

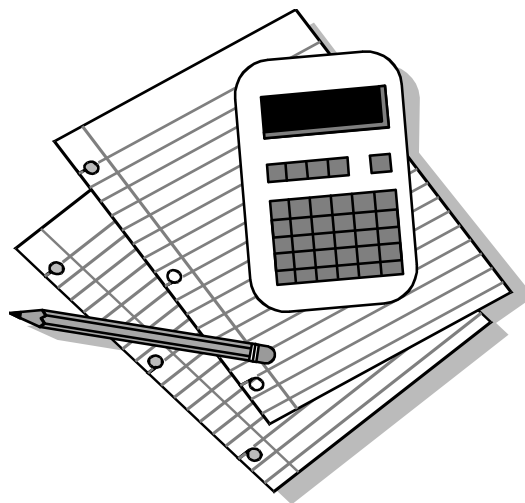


Кафедра конструирования
и производства радиоаппаратуры

Д.В. Озёркин

СХЕМОТЕХНИКА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Компьютерный лабораторный практикум для студентов
специальности 210201 «Проектирование и технология радио-
электронных средств» и 160905 «Техническая эксплуатация
транспортного радиооборудования»



ТОМСК 2012

Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

УТВЕРЖДАЮ

Заведующий кафедрой КИПР

_____ В.Н. ТАТАРИНОВ

“ ____ ” _____ 20__ г.

Д.В. Озёркин

СХЕМОТЕХНИКА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Компьютерный лабораторный практикум для студентов
специальности 210201 «Проектирование и технология радио-
электронных средств» и 160905 «Техническая эксплуатация
транспортного радиооборудования»

2012

Рецензент: профессор кафедры КИПР ТУСУР, д.т.н. Татаринов В.Н.

Технический редактор: доцент кафедры КИПР ТУСУР, к.т.н. Озёркин Д.В.

Озёркин Д.В.

Схемотехника компьютерных технологий. Компьютерный лабораторный практикум для студентов специальностей 210201 «Проектирование и технология радиоэлектронных средств» и 160905 «Техническая эксплуатация транспортного радиооборудования».

Томск: Томский государственный университет систем управления и радиоэлектроники, 2012. – 185 с.

В настоящем компьютерном лабораторном практикуме по дисциплине «Схемотехника компьютерных технологий» основным «инструментом» для выполнения лабораторных работ выступает программа схемотехнического моделирования MicroCAP.

Основное назначение лабораторного практикума – изучение принципов построения и проектирования функциональных узлов и устройств ЭВМ, а также цифровой автоматики.

© Озёркин Д.В., 2012

© Кафедра КИПР Томского
государственного университета систем
управления и радиоэлектроники, 2012

СОДЕРЖАНИЕ

НАЗНАЧЕНИЕ КОМПЬЮТЕРНОГО ЛАБОРАТОРНОГО ПРАКТИКУМА.....	6
ОСОБЕННОСТИ КОМПЬЮТЕРНОГО ЛАБОРАТОРНОГО ПРАКТИКУМА.....	8
СВЕДЕНИЯ ДЛЯ СТУДЕНТОВ, ОБУЧАЮЩИХСЯ С ПРИМЕНЕНИЕМ ДИСТАНЦИОННЫХ ТЕХНОЛОГИЙ.....	13
ОБЩИЕ МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОГО ПРАКТИКУМА.....	15
ТИПОВЫЕ ПРИЕМЫ РАБОТЫ В MICROSOFT, НЕОБХОДИМЫЕ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ ЗАДАНИЙ	18
1 ЛАБОРАТОРНАЯ РАБОТА №1 – СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ	52
1.1 Цель работы	52
1.2 Порядок выполнения работы	52
1.3 Канонические формы представления логических функций. Минимизация логических функций. Синтез логических устройств в заданных базисах.....	52
1.4 Пример минимизации логической функции и перехода в заданный базис	61
1.5 Лабораторное задание.....	70
1.6 Контрольные вопросы	70
1.7 Варианты заданий	71
2 ЛАБОРАТОРНАЯ РАБОТА №2 – ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ	73
2.1 Цель работы	73
2.2 Порядок выполнения работы	73
2.3 Синтез недоопределенных логических функций. Синтез логических устройств с несколькими выходами	73
2.4 Пример синтеза логических устройств	81
2.5 Лабораторное задание.....	88
2.6 Контрольные вопросы	89
2.7 Варианты заданий	89
2.7.1 Варианты заданий для минимизации недоопределенных логических функций	89
2.7.2 Варианты заданий для минимизации системы логических функций	90
3 ЛАБОРАТОРНАЯ РАБОТА №3 – УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ.....	93

3.1	Цель работы	93
3.2	Порядок выполнения работы	93
3.3	Способы настройки универсальных логических модулей.....	93
3.4	Пример различных способов настройки УЛМ.....	101
3.5	Лабораторное задание.....	111
3.6	Контрольные вопросы	111
3.7	Варианты заданий	112
4	ЛАБОРАТОРНАЯ РАБОТА №4 – ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ.....	115
4.1	Цель работы	115
4.2	Порядок выполнения работы	115
4.3	Методика проектирования автоматов с памятью	115
4.4	Пример проектирования автомата на основе JK-триггеров ...	121
4.5	Лабораторное задание.....	129
4.6	Контрольные вопросы	129
4.7	Варианты заданий	130
5	ЛАБОРАТОРНАЯ РАБОТА №5 – АЛЬТЕРНАТИВНЫЕ СПОСОБЫ ПРОЕКТИРОВАНИЯ АВТОМАТОВ С ПАМЯТЬЮ	131
5.1	Цель работы	131
5.2	Порядок выполнения работы	131
5.3	Автоматы с памятью на основе мультиплексного управления	131
5.4	Автоматы с памятью, имеющие кодирование состояний типа «1 из N».....	135
5.5	Примеры проектирования трехразрядного цифрового автомата.....	139
5.6	Лабораторное задание.....	148
5.7	Контрольные вопросы	148
5.8	Варианты заданий	149
6	ЛАБОРАТОРНАЯ РАБОТА №6 – ПРОЕКТИРОВАНИЕ ДВОИЧНО-КОДИРОВАННЫХ СЧЕТЧИКОВ С ПРОИЗВОЛЬНЫМ МОДУЛЕМ.....	150
6.1	Цель работы	150
6.2	Порядок выполнения работы	150
6.3	Способы построения двоично-кодированных счетчиков	150
6.4	Примеры проектирования двоично-кодированного счетчика с произвольным модулем	158
6.5	Лабораторное задание.....	165
6.6	Контрольные вопросы	166
6.7	Варианты заданий	167
	СПИСОК ЛИТЕРАТУРЫ	168
	ПРИЛОЖЕНИЕ 1 – СПРАВОЧНЫЕ СВЕДЕНИЯ О ПРОГРАММЕ СХЕМОТЕХНИЧЕСКОГО	

МОДЕЛИРОВАНИЯ MICROCAP	169
ПРИЛОЖЕНИЕ 2 – НЕКОТОРЫЕ СООБЩЕНИЯ ОБ ОШИБКАХ, ВЫДАВАЕМЫЕ ПРОГРАММОЙ MICROCAP .	176
ПРИЛОЖЕНИЕ 3 – ПРИМЕР ОФОРМЛЕНИЯ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ.....	179
ПРИЛОЖЕНИЕ 4 – СООТВЕТСТВИЕ УСЛОВНЫХ ГРАФИЧЕСКИХ ОБОЗНАЧЕНИЙ НЕКОТОРЫХ ЭРЭ В РОССИИ И ЗА РУБЕЖОМ	187

НАЗНАЧЕНИЕ КОМПЬЮТЕРНОГО ЛАБОРАТОРНОГО ПРАКТИКУМА

К настоящему времени наиболее совершенные принципы и средства взаимодействия человека с окружающим миром обеспечила цифровая техника. Ее наименее избыточный алфавит – двухуровневые символы, которыми оказалось возможным представить (кодировать) любую информацию – привел к созданию чрезвычайно точных, надежных, малогабаритных и функционально-наращиваемых устройств.

Использование в цифровой технике двухсимвольного алфавита привело к созданию новых, исключительно эффективных методов передачи, хранения и преобразования сигналов, к новым средствам обработки информации – компьютерным технологиям. Под этим словосочетанием понимают технологию обработки с использованием современных средств цифровой техники и ее вершины – вычислительной техники. Так родились основанные на новых принципах современные технологии: связи (цифровая связь и цифровое телевидение), обнаружения (цифровая радиолокация и цифровая навигация), вычислений и автоматического управления (электронно-вычислительная техника) и т.д.

Цифровая техника стоит на трех «китах». Первый «кит» - теорема о дискретизации, сформулированная и доказанная в 1933 г. академиком В.А. Котельниковым. В этой теореме теоретически обоснована возможность получения цифрового эквивалента (цифрового образа) аналогового сигнала, хранить, передавать и обрабатывать который оказалось значительно проще и точнее, чем осуществлять аналогичные действия над аналоговым сигналом.

Второй «кит» – алгебра логики (булева алгебра, названная так в честь ее автора – ирландского математика Дж. Буля). Получившая дальнейшее развитие в основополагающих трудах А.Н. Колмогорова, К. Шеннона, В.И. Шестакова и других ученых, алгебра логики позволила поставить анализ и синтез цифровых схем на прочный математический фундамент.

Третий «кит» - импульсная техника, из которой цифровая техника заимствовала многие принципы, элементы и устройства.

Цифровые устройства обладают рядом преимуществ перед аналоговыми: огромная степень интеграции, составляющая десятки миллионов транзисторов в одной микросхеме, чрезвычайно низкая погрешность, малая зависимость от параметров окружающей среды.

Области применения цифровой техники поистине безграничны. К сказанному ранее можно добавить, что в настоящее время до 90% всех разрабатываемых устройств – цифровые. Со знанием цифровой техники будущий инженер окажется востребованным в любой области и сумеет внести вклад в ее развитие.

При изучении электротехнических дисциплин в последние годы широко применяются программы и системы, позволяющие проводить схемотехни-

ческое моделирование. Внедрение в учебный процесс компьютерного моделирования обусловлено рядом причин. Среди них можно выделить такие, как:

- экономическая целесообразность. Модернизация устаревшего оборудования лабораторий в ряде случаев дороже оборудования компьютерного класса;

- возможность исследования схем, содержащих самые современные компоненты электронных схем;

- возможность проведения исследования схем и их компьютерного моделирования в домашних условиях;

- возможность дистанционного обучения;

- программные продукты, используемые при моделировании электронных схем, как правило, не требуют специальных настроек и нестандартного оборудования. Для установки таких программ и систем могут быть использованы компьютеры простых конфигураций.

В настоящем лабораторном практикуме для целей моделирования используется программный комплекс MicroCAP 7. Выбор указанного программного продукта не случаен, а продиктован следующими основными причинами:

- многолетний опыт применения различных версий программного продукта в учебном процессе;

- минимальные аппаратные требования к персональному компьютеру, о чем уже говорилось выше;

- исключительно схемотехническая направленность программного продукта, сбалансированность для целей настоящего лабораторного практикума. Следует отметить, что другие программные комплексы – OrCAD, MultiSim (EWB), Protel, PCAD – наряду со схемотехническим моделированием обладают возможностями топологического проектирования. В данном случае этот факт можно считать недостатком, поскольку указанные программные комплексы гораздо сложнее в освоении студентами.

Основное назначение лабораторного практикума – изучение принципов построения и проектирования функциональных узлов и устройств ЭВМ, а также цифровой автоматики. Кроме этого, компьютерный лабораторный практикум способствует приобретению студентами специальности 210201 и 160905 серьезных практических навыков в схемотехническом моделировании цифровых устройств. Практикум может также использоваться студентами других специальностей, учебный план которых имеет дисциплины, связанные с изучением цифровой электроники.

ОСОБЕННОСТИ КОМПЬЮТЕРНОГО ЛАБОРАТОРНОГО ПРАКТИКУМА

Компьютерный лабораторный практикум базируется на материале, изучаемом в лекционных курсах «Схемотехника компьютерных технологий» специальности 210201 и «Схемотехника-2» специальности 160905. Заметим, что дисциплина «Схемотехника компьютерных технологий» относится к регионально-вузовским компонентам учебного плана; дисциплина «Схемотехника» является специальной (СД.05) и входит в государственной образовательный стандарт ГОС ВПО 653300-2000.

Согласно учебных планов УП 210201 (200800)-01, УП 160905 (201300)-01, УП 210201 (200820).з-2003, УП 201300.з-01, УП 200800.ДД-2002, УП 200800.ЗД-2002, УП 200820-2001*2004 ИФ, количество часов, выделенных на выполнение лабораторных работ, различается в зависимости от специальности и формы обучения. Общий список тем лабораторных занятий, включенных в настоящий лабораторный практикум, состоит из шести работ:

1. Синтез комбинационных логических устройств.
2. Особые случаи синтеза комбинационных логических устройств.
3. Универсальные логические модули на основе мультиплексоров.
4. Проектирование цифровых автоматов на *JK*-триггерах.
5. Альтернативные способы проектирования автоматов с памятью.
6. Проектирование двоично-кодированных счетчиков с произвольным модулем.

Конкретные сведения о количестве и темах лабораторных работ для каждой из специальностей и форм обучения приведены ниже в таблице.

Таблица

СПЕЦИАЛЬНОСТЬ	ФОРМА ОБУЧЕНИЯ	КОЛИЧЕСТВО РАБОТ	НАЗВАНИЯ РАБОТ
210201	ОЧНАЯ, В ТОМ ЧИСЛЕ С ОБУЧЕНИЕМ В ФИЛИАЛАХ	6	<p>1. СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ.</p> <p>2. ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ.</p> <p>3. УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ.</p> <p>4. ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА <i>JK</i>-ТРИГГЕРАХ.</p> <p>5. АЛЬТЕРНАТИВНЫЕ СПОСОБЫ</p>

			ПРОЕКТИРОВАНИЯ АВТОМАТОВ С ПАМЯТЬЮ. 6. ПРОЕКТИРОВАНИЕ ДВОИЧНО- КОДИРОВАННЫХ СЧЕТЧИКОВ С ПРОИЗВОЛЬНЫМ МОДУЛЕМ.
--	--	--	--

Окончание таблицы

СПЕЦИ- АЛЬНОС ТЬ	ФОРМА ОБУЧЕ НИЯ	КОЛИЧЕ СТВО РАБОТ	НАЗВАНИЯ РАБОТ
210201	ЗАОЧНА Я	4	1. СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 2. ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 3. УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ. 4. ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ.
210201	ЗАОЧНА Я С ДИСТА НЦИОН НОЙ ТЕХНО ЛОГИЕ Й	4	1. СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 2. ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 3. УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ. 4. ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ.
210201	ДИСТА НЦИОН НАЯ	6	1. СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 2. ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 3. УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ. 4. ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ. 5. АЛЬТЕРНАТИВНЫЕ СПОСОБЫ ПРОЕКТИРОВАНИЯ АВТОМАТОВ С ПАМЯТЬЮ. 6. ПРОЕКТИРОВАНИЕ ДВОИЧНО- КОДИРОВАН-НЫХ СЧЕТЧИКОВ С ПРОИЗВОЛЬНЫМ МОДУЛЕМ.
160905	ОЧНАЯ	4	1. СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 2. ОСОБЫЕ СЛУЧАИ СИНТЕЗА

			КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 3. УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ. 4. ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ.
160905	ЗАОЧНА Я	2	1. СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ. 2. ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ.

Описанию лабораторных работ предшествует раздел по демонстрации всех типовых приемов, которые необходимы для выполнения лабораторных заданий. Для удобства такие типовые приемы озаглавлены и пронумерованы. В описаниях лабораторных работ даются соответствующие ссылки на типовые приемы, для того чтобы не акцентировать внимание на технических подробностях при изложении основного материала. Демонстрационный раздел компьютерного лабораторного практикума можно рассматривать и как краткое учебное пособие по моделированию цифровых устройств в MicroCAP и как справочный материал. Раздел снабжен максимально подробными разъяснениями и иллюстрациями.

В описаниях лабораторных работ включены две обязательные части: краткие теоретические сведения по теме работы и методический пример по выполнению лабораторного задания. Краткие теоретические сведения, предваряющие лабораторное задание, позволяют студентам вспомнить материал лекционного курса без специального поиска нужного раздела в учебном пособии. Методический пример к каждой работе представляет собой аналог соответствующего лабораторного задания. Таким образом, задача студентов сводится к повторению представленного методического примера, но по своему варианту задания. Методические примеры демонстрируют определенный «сценарий» проведения лабораторных исследований:

1. Построение математической модели цифрового устройства с использованием аппарата алгебры логики (булевой алгебры).

2. Реализация схмотехнического решения в заданном логическом базисе или на заданной элементной базе.

3. Анализ полученного схмотехнического решения в программе моделирования MicroCAP с целью проверки правильности функционирования.

Лабораторные работы содержат в своем составе различное число зависимых или независимых друг от друга лабораторных заданий – от одного до четырех.

Например, лабораторная работа №3 состоит из трех независимых заданий, объединяет которые изучение способов реализации универсальных ло-

гических модулей на основе мультиплексоров. Лабораторные работы №4 и №5, напротив, взаимосвязаны общим вариантом задания и фактически являются одной работой по изучению методов проектирования цифровых автоматов.

В конце компьютерного лабораторного практикума приведены дополнительные справочные материалы:

- сведения о программе схемотехнического моделирования MicroCAP 7;
- правила оформления отчета по лабораторной работе;
- соответствие условных графических обозначений элементов в России и за рубежом.

СВЕДЕНИЯ ДЛЯ СТУДЕНТОВ, ОБУЧАЮЩИХСЯ С ПРИМЕНЕНИЕМ ДИСТАНЦИОННЫХ ТЕХНОЛОГИЙ

Для студентов, обучающихся с применением дистанционных технологий, необходимо обратиться к предыдущему разделу и найти в таблице количество лабораторных работ и их темы в соответствии с Вашей специальностью и формой обучения. Категорически запрещается изменять темы лабораторных работ по своему усмотрению. Порядок выполнения лабораторных работ решающего значения не имеет. Для выполнения лабораторных работ следует иметь:

1. Настоящий лабораторный практикум.
2. Персональный компьютер, практически любой аппаратной конфигурации.
3. Дистрибутивный архив с программой MicroCAP 7.

Источники приобретения программного обеспечения в России широко известны и поэтому здесь не комментируются. Если в Вашем случае возникли принципиальные трудности с поиском указанного программного продукта, то в качестве варианта можно обратиться за помощью к преподавателю информатики ТУСУРа, который проводит выездные сессии в Вашем регионе.

Не следует пользоваться демонстрационными версиями программы MicroCAP (Evaluation version), которые доступны для бесплатного копирования с Web-сервера фирмы-разработчика. В демонстрационных версиях введены некоторые ограничения, в частности, в библиотеке элементов отсутствуют многие цифровые микросхемы. Это обстоятельство не позволяет использовать демонстрационные версии программы для выполнения лабораторных работ.

Выполнять лабораторные задания можно в более ранних полнофункциональных версиях MicroCAP. Заметим, что интерфейс и функциональные возможности ранних версий программы несколько отличаются от MicroCAP 7.

Номер варианта выбирается по общепринятым правилам в дистанционном образовании по формуле:

$$V = (25 \cdot k) \operatorname{div} 100,$$

где V – искомый номер варианта (при $V = 0$ выбирается номер варианта 25); k – две последние цифры номера Вашей зачетной книжки; div – операция целочисленного деления.

При отправке отчетов по лабораторным работам следует указывать свой номер варианта. Пример оформления отчета по лабораторной работе приведен в Приложении 3. Отчет по лабораторной работе обязательно должен быть в виде твердой копии, при этом допускается как рукописный, так и печатный текст отчета. Аргументами в пользу представления отчета в виде твердой копии являются следующие причины:

- исключение возможности получения компьютерных вирусов;
- снятие проблем технического характера, связанных с поврежденными архивами или архивами неизвестного формата;
- снятие проблем, связанных с прочтением документа или его части из-за применения нестандартных средств оформления;
- снятие проблем, связанных с проверкой, рецензированием и исправлениями отчета, а особенно его графической части.

Распространенной ошибкой в отчетах является неполное оформление графиков. Графики функций всегда должны иметь единицы измерения по осям, правильный масштаб, поясняющие подписи, дополнительные построения (если это требует лабораторное задание). В связи с этим не нужно бояться на полученной твердой копии графика наносить от руки дополнительную информацию. Ответы на контрольные вопросы содержатся в теоретической части описания лабораторной работы.

Вопросы **технического** характера, возникающие при выполнении лабораторных работ, Вы можете отправить по электронному адресу *kipr_depart@main.tusur.ru*.

ОБЩИЕ МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОГО ПРАКТИКУМА

Предполагается, что студент должен обладать начальными практическими навыками работы в программе MicroCAP, полученные им в курсе «Информатика». В связи с этим в настоящем лабораторном практикуме остались вне рассмотрения следующие вопросы:

- состав и назначение пунктов меню;
- состав и назначение элементов панели инструментов в режимах изображения схемы и анализа графиков;
- приемы изображения электрических схем;
- виды схемотехнического анализа и запуск на моделирование.

Если Вы по какой-либо причине не владеете перечисленными знаниями и навыками работы, то целесообразно обратиться сначала к [1], а затем уже к настоящему лабораторному практикуму. Перед выполнением лабораторных заданий рекомендуется ознакомиться с разделом «Типовые приемы работы в MicroCAP, необходимые для выполнения лабораторных заданий»; полезно также воспроизвести примеры из этих разделов на компьютере.

Приведем несколько практических советов при работе в программе схемотехнического моделирования MicroCAP.

Программа MicroCAP сохраняет результаты работы в файлы с расширением *.cir. Распространенной ошибкой является попытка сохранить непосредственно графики зависимостей, полученные в результате моделирования. Такая попытка будет неудачна, поскольку в режиме отображения графиков пункт меню *Save* всегда заблокирован.

Следует помнить, что основная информация, которая сохраняется в рабочем файле *.cir – это не графики функций, а электрическая схема вместе с индивидуальными настройками и установками на проведение какого-либо моделирования. Таким образом, всегда нужно сначала закрыть окно с графиками зависимостей, а затем уже приступать к сохранению информации и выходу из программы. Если во время следующего сеанса работы открыть сохраненный файл – получим первоначальные графики зависимостей.

Если есть необходимость перенести изображение графика функции в какое-либо приложение, например, в текстовый редактор Microsoft Word, то проще всего это сделать через буфер обмена по команде *Edit/Copy to Clipboard/Copy the Visible Portion of Window in BMP Format*.

Полезно при работе в программе MicroCAP сохранять промежуточные результаты своей работы с периодичностью в 15-20 минут для того, чтобы не потерять всю информацию. Особенно важно это перед первым запуском на моделирование – в случае неправильного задания параметров иногда возникает аварийная ситуация и программа MicroCAP «зависает».

При выполнении лабораторных работ следует стремиться, чтобы каждое новое частное задание располагалось в отдельном файле, даже если схема включения одинакова. В противном случае, можно очень легко запутаться в многочисленных установках параметров на моделирование, которые в каждом новом задании имеют индивидуальные особенности. Несоблюдение этого правила может привести к тому, что вместо ожидаемого результата появится совсем иной.

По сравнению с моделированием аналоговых устройств схемотехнический анализ цифровых устройств в программе MicroCAP имеет некоторые специфические особенности, о которых необходимо упомянуть.

По общим признакам анализ цифровых устройств в MicroCAP содержит много условностей в представлении электрической схемы и напоминает функционально-логическое моделирование [2]. Часто при изображении электрической схемы цифрового устройства в MicroCAP используют набор либо логических примитивов (вентилей), либо секций. Такие наборы не имеют законченного физического воплощения (корпуса) и являются фрагментами интегрального исполнения микросхемы. Иногда используемые в электрической схеме логические примитивы вообще не имеют прямых аналогов в составе интегральной микросхемы. Например, цифровая электрическая схема, состоящая из элементов И, ИЛИ, НЕ, как правило, абстрактна и служит для проверки математической модели, поскольку в реальных микросхемах используется базис И-НЕ, ИЛИ-НЕ.

По правилам программы MicroCAP для цифровых схем цепи питания и «земли» не только не изображаются на чертеже, но даже не упоминаются в текстовом виде (в отличие, например, от систем топологического проектирования). Программа MicroCAP при составлении списка соединений электрической схемы производит автоматическое подсоединение таких цепей без участия пользователя. В связи с этим для цифровых схем на поле чертежа не требуется размещать символы общих проводников и источников питания. С другой стороны, практически все цифровые схемы, моделируемые в MicroCAP, должны иметь на входе источник сигнала – это может быть источники информационного сигнала и/или источники служебных сигналов (асинхронный сброс, синхронизация и т.д.).

Моделирование в MicroCAP различного рода импульсных помех в цифровых схемах не предусмотрено, поэтому нет необходимости использовать фильтрующие конденсаторы, согласующие линии и т.д.

Применительно к цифровой элементной базе, используемой в MicroCAP, имеется возможность ввести ненулевые или изменить существующие времена задержки прохождения сигналов и, тем самым, провести анализ «гонок сигналов» и аномальных состояний в цифровой схеме, оценить статические риски.

В программе MicroCAP нет специального графического средства для изображения линий групповой связи (шин), однако каждый из проводников предполагаемой шины можно именовать, например, A0, A1, ..., что фактиче-

ски и сформирует шину. При изображении электрической схемы в MicroCAP допускаются разрывы проводников при условии, что им присвоены уникальные имена. В этом случае, все отрезки проводников, имеющие одинаковые имена, образуют глобальную электрическую цепь (см. прием № 5 следующего раздела).

Выводы электрорадиоэлементов должны быть подключены к электрическим проводникам. Исключением из этого правила могут являться цифровые источники сигналов и условные графические обозначения интегральных микросхем, не все выводы которых используются в данной конкретной схеме. Большинство цифровых схем, моделируемых в MicroCAP, не требует подключения к выходу нагрузки – эта особенность справедлива, если речь идет о логических выходах или выходах с третьим состоянием.

Из всех доступных в MicroCAP видов анализа наибольший практический интерес для цифровых схем представляет временной вид анализа (Transient Analysis) с наглядным представлением форм сигналов и их взаимного положения. В режиме по постоянному току (Dynamic DC) можно оценить логические состояния цифровых элементов в начальный момент времени.

При составлении символических и математических выражений в MicroCAP указание на какой-либо элемент производится по его позиционному обозначению на электрической схеме. Несоответствие позиционного обозначения элемента на схеме и в выражении всегда вызывает сообщение об ошибке.

Сообщения об ошибках выводятся в программе MicroCAP на английском языке, поэтому не всегда информативны и понятны. В любом случае следует внимательно прочесть такое сообщение и постараться устранить причину, вызвавшую появление ошибки. В Приложении 2 приведены некоторые (типичные) сообщения об ошибках программы MicroCAP.

ТИПОВЫЕ ПРИЕМЫ РАБОТЫ В MICROCAP, НЕОБХОДИМЫЕ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ ЗАДАНИЙ

Прием №1. Работа с источником логических констант Fixed Digital.

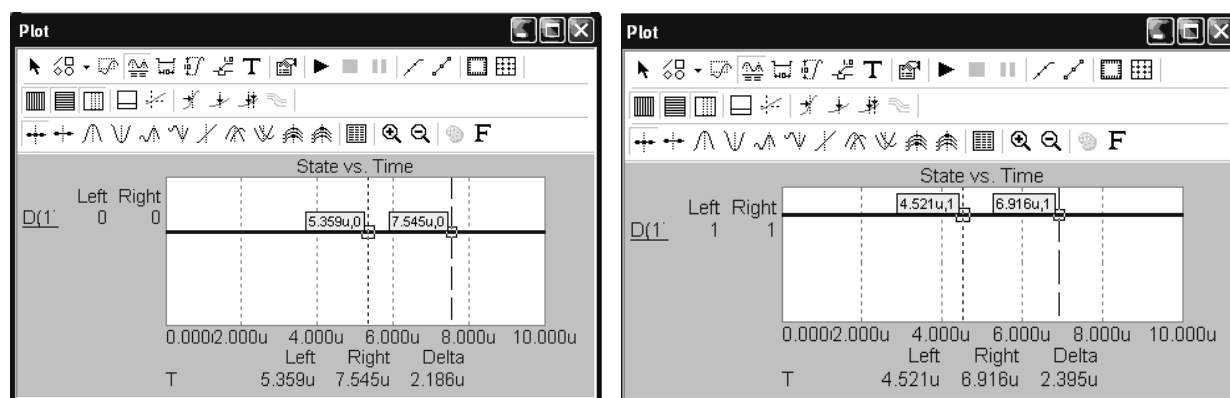
Наиболее простой источник Fixed Digital предназначен для установки в цифровых схемах уровней логической единицы или логического нуля. Для размещения источника на поле чертежа применяется команда *Component/Digital Primitives/Stimulus Generators/Fixed Digital*. В появляющемся диалоговом окне свойств элемента в строке **Value** указывают одно из двух возможных состояний источника: **0** или **1**.

На поле чертежа (рисунок 1) позиционное обозначение для такого источника не выводится, отображается только установленное значение логической константы.



Рисунок 1 – Условное графическое обозначение источника логических констант

Естественно, что в течение всего сеанса работы цифровой схемы значение источника Fixed Digital не меняется. Вообще правильность задания параметров любого цифрового сигнала можно оперативно оценить нажатием на кнопку **Plot** в диалоговом окне свойств. При этом появляется окно с временными диаграммами цифрового сигнала (рисунок 2).




а)

б)

а) уровень логического нуля; б) уровень логической единицы

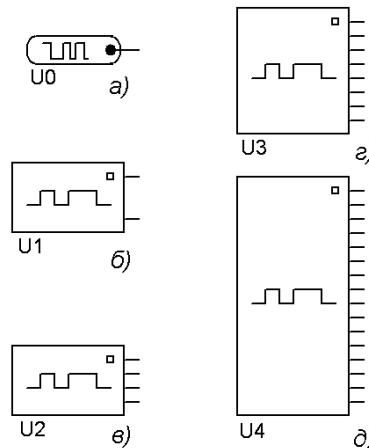
Рисунок 2 – Временные диаграммы для источника логических констант

Проверить количественные параметры цифрового сигнала удобно в режиме электронного курсора. Для перехода в этот режим достаточно нажать пиктограмму  на панели инструментов в верхней части окна. При включении режима появляются изображения курсоров с указанием текущих координат

нат. Курсоры можно скачкообразно перемещать в любую точку графика нажатием левой или правой кнопок мыши. Уровень логического сигнала отображается слева от графика, причем значения даются отдельно для левого (Left) и правого (Right) курсора. Например, из рисунка 2, а видно, что левый и правый курсор находятся на одном и том же уровне логического нуля (**Left 0; Right 0**). Более подробно работа с графиками в режиме электронного курсора рассмотрена в приеме №4.

Прием №2. Работа с цифровыми источниками сигнала Stim1, Stim2, Stim4, Stim8, Stim16.

Источники сигналов Stim1, Stim2, Stim4, Stim8, Stim16 предназначены для выработки цифрового кода, представленного, соответственно, в 1, 2, 4, 8 и 16 разрядах цифрового слова. Размещение на поле чертежа графических образов источников сигнала происходит по команде *Component/Digital Primitives/Stimulus Generators/Stim X*, где вместо X может быть число 1, 2, 4, 8, 16. Маленький квадратик на графических образах источников – это ключ, обозначающий расположение старшего разряда (рисунок 3).



а) Stim1; б) Stim2; в) Stim4; г) Stim 8; д) Stim 16

Рисунок 3 – Условные графические образы цифровых источников сигнала

Заметим, что в цифровых схемах старшие разряды многоразрядных источников сигнала иногда не используются, например, если фактическая разрядность схемы меньше разрядности источника сигнала. В этом случае старшие разряды остаются на поле чертежа не подключенными; каких-либо сообщений об ошибках в программе MicroCAP этот факт не вызывает.

В однотипных диалоговых окнах свойств источников Stim1, Stim2, Stim4, Stim8, Stim16 указывают информацию о выбранной системе счисления и о параметрах сигнала. Происходит это, соответственно, в строках **FORMAT** и **COMMAND**.

Допустимые значения в строке **FORMAT** – это одна или несколько цифр из множества {1, 3, 4}; они интерпретируются как показатели степени числа 2 и задают двоичную ($2^1 = 2$), восьмеричную ($2^3 = 8$) или шестнадцатеричную ($2^4 = 16$) систему счисления для последующего описания сигнала.

Предположим, что в схеме используется четырехразрядный источник Stim4. Путем несложных рассуждений можно перечислить варианты представления сигнала для данного источника в строке **FORMAT**:

1. **1111** – четыре одноразрядных подслова, в каждом из которых используется алфавит {0, 1}.

2. **13** или **31** – комбинация одноразрядного и трехразрядного подслов с алфавитом {0, 1} и {0, 1, 2, 3, 4, 5, 6, 7}, соответственно.

3. **4** – одно четырехразрядное слово, в котором используется алфавит {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}.

Из представленных вариантов видно, что количество цифр в записи определяет количество цифровых подслов; сумма цифр в записи обязательно должна совпадать с разрядностью источника цифрового сигнала. В общем случае для источника цифрового сигнала Stim X , где $X \in \{1, 2, 4, 8, 16\}$, последовательность действий такова:

а) выбирается система счисления n для описания сигнала:

$$n \in \{1, 3, 4\};$$

б) в выбранной системе счисления определяется количество цифровых подслов l для исходного слова X :

$$l = X \operatorname{div} n,$$

где div – операция целочисленного деления;

в) если остаток от деления $X \bmod n \neq 0$, то подбираются цифры $k_i \in \{1, 3, 4\}$, так чтобы:

$$n \cdot l + \sum_i k_i = X;$$

г) в строку **FORMAT** записывают форму представления сигнала:

$$\underbrace{nn \dots nk_1 k_2 \dots}_l$$

При выборе строки **COMMAND** диалогового окна в его нижней части появляется область для написания и редактирования параметров цифрового сигнала. Описание параметров цифрового сигнала представляет собой произвольную комбинацию нескольких следующих строк:

.DEFINE <имя сигнала>

+ <*t*> <значение>

+ **LABEL** <имя метки>

+ <*t*> **GOTO** <имя метки> <*n*> **TIMES**

+ <*t*> **GOTO** <имя метки> **UNTIL GT** <значение>

+ <*t*> **GOTO** <имя метки> **UNTIL GE** <значение>

+ <*t*> **GOTO** <имя метки> **UNTIL LT** <значение>

+ <*t*> **GOTO** <имя метки> **UNTIL LE** <значение>

+ <*t*> **INCR BY** <значение>

+ <*t*> **DECR BY** <значение>

Тестовая директива **.DEFINE** предназначена для присвоения уникального имени текущему цифровому сигналу. При написании имени сигнала буквы русского алфавита не допускаются. Строка с директивой **.DEFINE** яв-

ляется обязательной и всегда располагается в начале. Символы «+» в первой позиции последующих строк обозначают продолжение описания, т.е. служат признаком переноса с предыдущей строки.

Переменная t определяет моменты времени, в которые задаются новые значения цифрового кода сигнала. Если перед значением переменной t имеется символ «+» (не путать с аналогичным символом в первой позиции строки), то это значение времени является относительным, измеренным от предыдущего момента. В случае отсутствия символа «+» считается, что переменная t обозначает абсолютное время – время, прошедшее от начала действия сигнала. Например, запись **+7** говорит о том, что относительно предыдущего момента времени прошло 7 секунд; запись **7** – что прошло 7 секунд с начала сеанса работы.

В записи числовых значений времени допускаются специальные суффиксы для обозначения кратных и дольных единиц секунды (см. Приложение 1). Суффикс «s» является необязательным и имеет смысл «секунда». Все суффиксы записываются слитно с их числовыми значениями. Например, записи 7ns и 7n – равноценны и обозначают время в 7 наносекунд, причем «n» является резервированным суффиксом программы MicroCAP.

<Значение> определяет новую величину цифрового кода сигнала. При записи новой величины пользуются теми же алфавитами и в том же порядке, как это было оговорено выше в строке **FORMAT**. Например, если в строке **FORMAT** было **31**, то запись цифрового кода сигнала **51** является допустимой, а запись **52** – ошибочной, поскольку второе подслово должно использовать алфавит {0, 1}.

Заметим, что в программе MicroCAP все системы счисления имеет расширенный алфавит, где помимо констант, существует следующие допустимые значения:

- R – передний фронт импульса;
- F – задний фронт импульса (в шестнадцатеричной системе не используется);
- X – неопределенное состояние вывода;
- Z – третье состояние вывода;
- ? – случайное значение.

Оператор **LABEL** устанавливает метку, название которой <имя метки> следует через пробел (буквы русского алфавита не допускаются). Метка используется при организации циклов совместно с оператором перехода **GOTO**. Оператор **GOTO** передает управление на строку, следующую за оператором **LABEL**.

Переменная < n > задает количество повторяющихся циклов **GOTO**; значение $n = -1$ задает бесконечное повторение цикла. Резервированное слово **TIMES** имеет смысл «количество раз».

Условный оператор **UNTIL** имеет смысл «до тех пор, пока не» и в сочетании с оператором перехода **GOTO** образует циклическую структуру с

предусловием. Ключевые слова **GT** (greater than), **GE** (greater than or equal), **LT** (less than), **LE** (less than or equal) являются аналогами математических операций отношения $>$, \geq , $<$, \leq , соответственно.

Операторы инкрементирования **INCR BY** и декрементирования **DECR BY** служат для увеличения или уменьшения текущего значения цифрового кода сигнала на величину <значение>.

Проиллюстрируем работу с цифровыми источниками сигнала на нескольких примерах.

Пример 1. Требуется задать одиночный отрицательный импульс, передний фронт которого соответствует моменту времени 10 нс, а задний фронт – моменту времени 20 нс.

Решение. Размещаем на поле чертежа одnorазрядный источник цифрового сигнала Stim1. В диалоговом окне свойств источника в строке **FORMAT** указываем значение **1**, т.е. выбираем двоичную систему счисления. При выборе строки **COMMAND** в появляющейся внизу области для редактирования записываем листинг:


```
.DEFINE EXAMPLE1
```

```
+ 0ns 1
```

```
+ 10ns 0
```

```
+ 20ns 1
```

Описание сигнала состоит из четырех строк. В первой строке записывается имя сигнала (*Example1*), последующие строки определяют характерные моменты времени, в которые происходит смена логического уровня сигнала.

Нажатие в диалоговом окне кнопки **Plot** приводит к появлению временной диаграммы (рисунок 4). Убедиться в правильности задания количественных параметров сигнала можно в режиме электронного курсора, нажав пиктограмму .

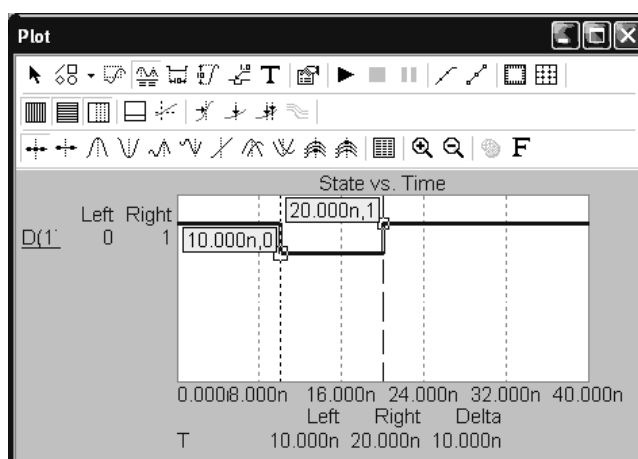


Рисунок 4 – Временная диаграмма одиночного отрицательного импульса

Замечание. В ряде случаев после нажатия на кнопку **Plot** может появиться временная диаграмма с неудачным масштабом. Слишком большой или слишком маленький масштаб временной диаграммы не позволяет наблюдать характерные особенности цифрового сигнала. Выходом из создавшей

ситуации служит ручное масштабирование. Для этого необходимо сделать двойной щелчок на любом числовом значении временной оси. В появившемся диалоговом окне **Properties** переходят на вкладку **Scales and Formats**. В блоке настроек **X** параметр **Grid Spacing** устанавливают равным нулю, а параметр **Range High** подбирают вручную так, чтобы обеспечивалось нормальное визуальное восприятие цифрового сигнала. Для данного примера Range High = **4e-8** или 40 нс.

Пример 2. По условию предыдущего примера требуется составить описание сигнала с использованием относительных значений времени.

Решение. Отличие от предыдущего случая заключается в трех последних строках описания:

.DEFINE EXAMPLE2

+ +0ns 1

+ +10ns 0

+ +10ns 1

Здесь запись +10ns обозначает относительное время, прошедшее с предыдущего момента. Поскольку описания сигналов в примере 1 и примере 2 равноценны, то и временные диаграммы у них одинаковы (см. рисунок 4).

Пример 3. Требуется задать одноразрядную бесконечную последовательность импульсов с одинаковыми длительностями паузы и импульса, равными 5 нс. В начальный момент времени уровень логического сигнала равен нулю.

Решение. Размещаем на поле чертежа одноразрядный источник цифрового сигнала Stim1. В диалоговом окне свойств источника в строке **FORMAT** указываем значение 1. При выборе строки **COMMAND** в появляющейся внизу области для редактирования записываем листинг:

.DEFINE EXAMPLE3

+ +0ns 0

+ LABEL begin

+ +5ns 1

+ +5ns 0

+ +5ns GOTO begin -1 TIMES

В представленном листинге используются временные промежутки 5 нс. В третьей строке листинга с помощью оператора **LABEL** введена метка с названием **begin**. В последней строке листинга с помощью оператора **GOTO** организован переход на метку, причем значение -1 задает такой переход бесконечное количество раз. Приведенное описание позволяет получить требуемый по условию сигнал (рисунок 5).

Пример 4. Требуется задать двухразрядную бесконечную последовательность импульсов, образующую циклический двоичный код: 00 → 01 → 10 → 11 → 00 → Длительности паузы и импульса в младшем разряде равны 5 нс.

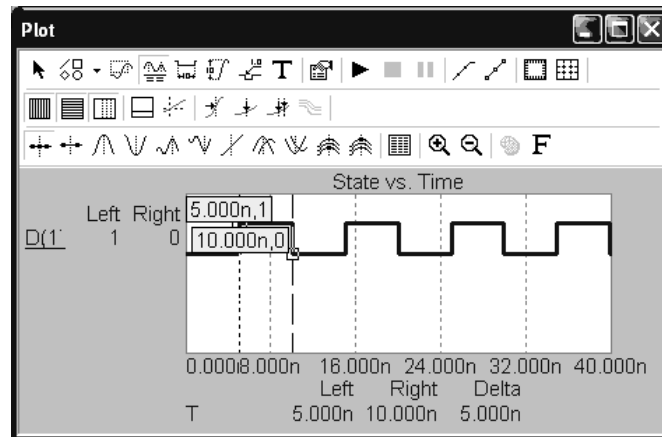


Рисунок 5 – Временная диаграмма бесконечной последовательности импульсов

Решение. Решить поставленную задачу можно несколькими способами. Покажем наиболее наглядный способ, не претендующий, однако, на оптимальность.

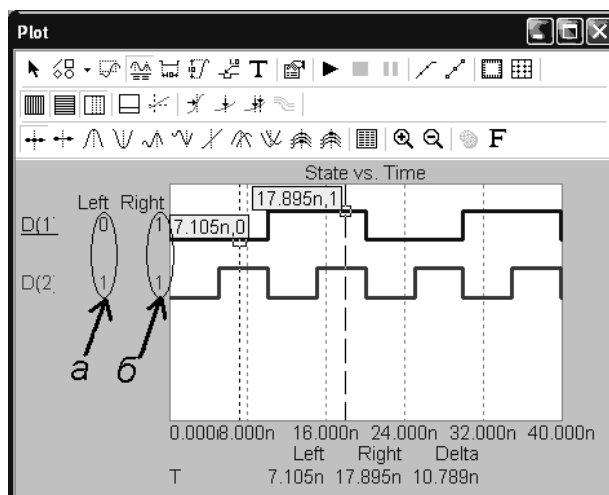
Размещаем на поле чертежа двухразрядный источник цифрового сигнала Stim2. В диалоговом окне свойств источника в строке **FORMAT** указываем значение **11**, т.е. исходное цифровое слово поделено на два подслова, в каждом из которых применяется двоичная система счисления. После выбора строки **COMMAND** записываем листинг:

```
.DEFINE EXAMPLE4
+ LABEL begin
+ +0ns 00
+ +5ns 01
+ +5ns 10
+ +5ns 11
+ +5ns GOTO begin -1 TIMES
```

В представленном листинге через временные интервалы в 5 нс перечислены в порядке возрастания все возможные состояния двухразрядного цифрового слова. В конце листинга с помощью оператора **GOTO** задан бесконечный повтор этих состояний.

Проверку правильности описания сигнала можно провести в режиме электронного курсора (рисунок 6), последовательно продвигая курсор слева направо и анализируя состояния разрядов. Заметим, что в окне **Plot**, в отличие от общепринятых правил, старший разряд обозначается как **D(1)**, следующий разряд – **D(2)** и т.д.

Пример 5. Требуется задать восьмиразрядный сигнал, который отражает последовательность шестнадцатеричного кода: 01 → 02 → 03 → 04 → 05 → 06 → F0 → F1. Длительность состояния $(06)_{16}$ должна составлять 20 нс. Последнее состояние $(F1)_{16}$ длится неограниченно долго. Длительность всех других состояний составляет по 10 нс.



а) цифровое состояние 01, отмеченное левым курсором;

б) цифровое состояние 11, отмеченное правым курсором

Рисунок 6 – Временная диаграмма двухразрядного сигнала с естественным порядком счета

Решение. Размещаем на поле чертежа условное графическое обозначение восьмиразрядного источника сигнала Stim8. В строке **FORMAT** указываем значение **44**, т.к. по условию задан шестнадцатеричный код.

После выбора строки **COMMAND** составляем листинг описания сигнала:

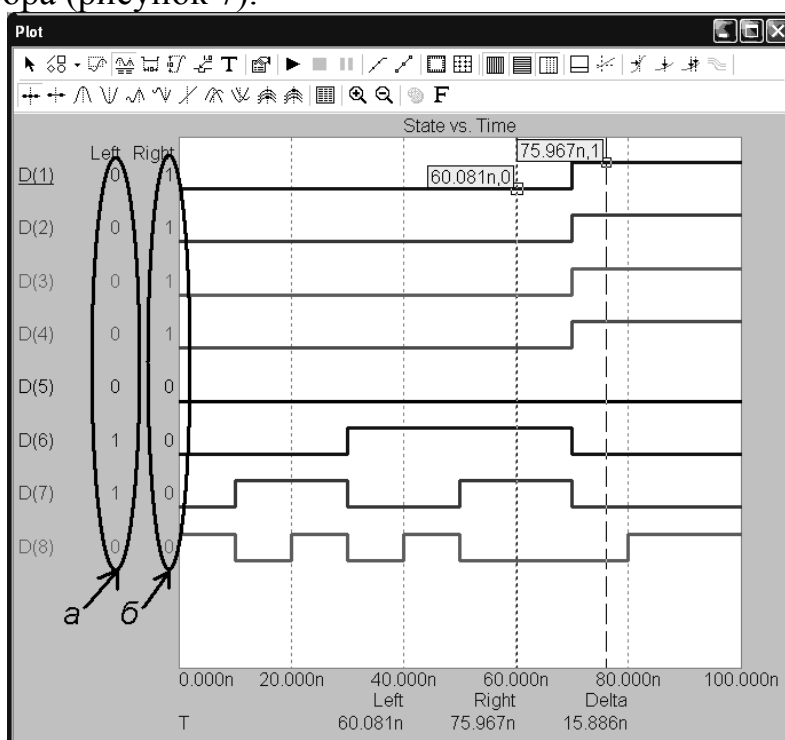
```
.DEFINE EXAMPLE5
+ LABEL begin
+ +0ns INCR BY 01
+ +10ns GOTO begin UNTIL GE 06
+ +10ns F0
+ +10ns F1
```

В третьей строке листинга использован оператор **INCR BY** для последовательного увеличения состояний на единицу. В четвертой строке оператор **GOTO** имеет условие прекращения цикла **UNTIL GE 06**. Если адаптировать условие прекращения цикла на русский язык, то будем иметь «повторять до тех пор, пока нет состояния больше или равного 6». Избавляясь от отрицания в логическом суждении, можно сказать «повторять, пока состояние меньше 6».

Проведем анализ представленного листинга. В нулевой момент времени состояние цифрового сигнала сразу становится 01. Далее через каждые 10 нс пять раз происходит хождение по циклу и последовательное увеличение состояний на единицу. В момент времени $t = 50$ нс происходит очередное увеличение состояния до значения 06. Затем, согласно листингу, еще через 10 нс, т.е. в момент $t = 60$ нс, условие **UNTIL GE 06** перестает выполняться. Управление переходит к пятой строке листинга, где сказано, что через 10 нс, т.е. в момент $t = 70$ нс, устанавливается состояние F0. Таким образом, предыдущее

состояние 06 длилось $\Delta t = 70 - 50 = 20$ нс. Последняя строка листинга устанавливает значение F1, которое длится неограниченно долго.

Проверка правильности описания сигнала проводится в режиме электронного курсора (рисунок 7).



- а) цифровое состояние $(00000110)_2 = (6)_{16}$, отмеченное левым курсором;
 б) цифровое состояние $(11110000)_2 = (F0)_{16}$, отмеченное правым курсором

Рисунок 7 – Временная диаграмма восьмиразрядного сигнала с состояниями 01 → 02 → 03 → 04 → 05 → 06 → F0 → F1

Следует отметить, что функциональные возможности окна **Plot** по представлению цифровых сигналов ограничены. Несмотря на то, что в описании сигнала использован шестнадцатеричный код, его графическое представление возможно только поразрядно в виде импульсных последовательностей, где D(1) – старший разряд, D(8) – младший. От указанного недостатка свободно окно результатов моделирования во временной области **Transient Analysis**, которое появляется после успешного завершения схемотехнического анализа цифровой схемы. В этом окне возможно представление многоразрядных сигналов в четырех базовых системах счисления. Более подробно об этом говорится в приеме №3.

Пример 6. Требуется задать восьмиразрядный сигнал, состоящий из двух независимых тетрад (четырёхразрядных подслов). Старшая тетрада должна быть представлена в двоичном коде, младшая тетрада – в шестнадцатеричном. Последовательность смены состояний в старшей тетраде: 0000 → 0001 → 0010 → 0011 → → 1111 → 0000. Для младшей тетрады первые четыре состояния должны принимать случайные значения; начиная с пятого со-

стояния – значения детерминированы: $4 \rightarrow 5 \rightarrow \dots \rightarrow F \rightarrow 0$. Длительность состояний составляет 100 нс.

Решение. Размещаем на поле чертежа условное графическое обозначение восьмиразрядного источника сигнала Stim8. В строке **FORMAT** указываем значение 11114, т.к. по условию старшая и младшая тетрады представлены в различных кодах. После выбора строки **COMMAND** записываем листинг:

.DEFINE EXAMPLE6

+0ns 0000?

+100ns 0001?

+200ns 0010?

+300ns 0011?

+400ns 01004

+500ns 01015

+600ns 01106

+700ns 01117

+800ns 10008

+900ns 10019

+1000ns 1010A

+1100ns 1011B

+1200ns 1100C

+1300ns 1101D

+1400ns 1110E

+1500ns 1111F

+1600ns 00000

Входной язык программы MicroCAP предусматривает использование операторов **INCR BY** и **DECR BY** только для цифровых слов, все разряды которых представлены в одинаковой системе счисления. По этой причине для данного примера организация циклических структур невозможна и применяется табличное представление сигнала. Символ «?» из расширенного алфавита обозначает случайное значение в выбранной системе счисления.

Оценить форму сигнала в поразрядном представлении можно в окне **Plot** (рисунок 8).

Внимание! Воспроизведение примера 6, связанного со случайными значениями в младшей тетраде, приводит к неповторяющимся результатам. Причина этого – разные наборы значений, вырабатываемых генератором случайных чисел при каждом запуске.

Прием №3. Моделирование цифровых схем во временной области.

Моделирование цифровых схем в MicroCAP во временной области сопровождается широкими функциональными возможностями по способам обработки и представления результатов. Проиллюстрируем некоторые из этих возможностей, вновь обратившись к примерам 4, 5, 6, разобранным в приеме №2.

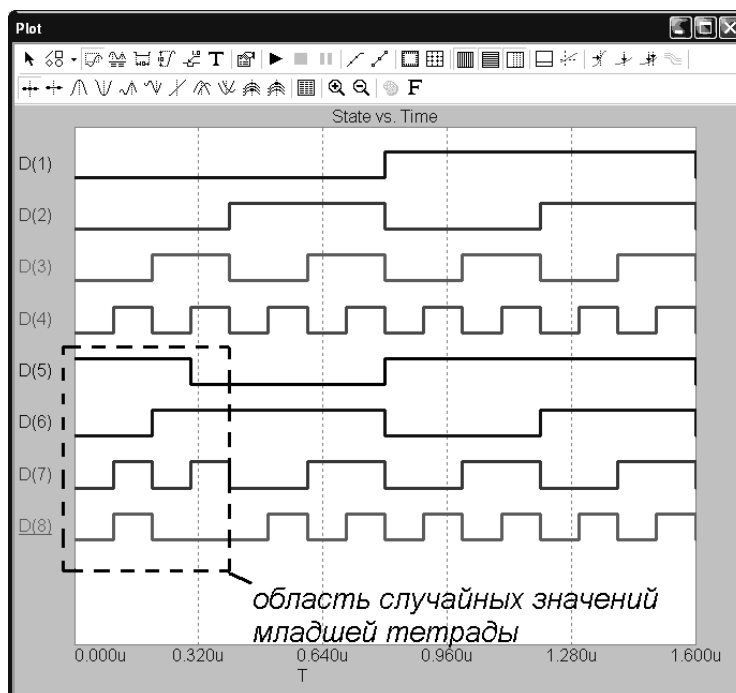



Рисунок 8 – Временная диаграмма восьмиразрядного сигнала с четырьмя случайными значениями в младшей тетраде

Пусть на поле чертежа размещен двухразрядный источник Stim2 со всеми внутренними параметрами из примера 4 (рисунок 9). С точки зрения схемотехнического анализа, источник Stim2 – это простейшая цифровая схема, состоящая из одного элемента. Перед началом моделирования во временной области определим номера контрольных точек, в которых требуется наблюдать сигнал. Для схемы примера 4 контрольные точки – это выходы цифрового источника сигнала Stim2. По умолчанию программа MicroCAP присваивает в произвольном порядке номера для всех контрольных точек на схеме. Обычно информация о номерах скрыта от пользователя; увидеть ее можно, нажав на пиктограмму  панели инструментов.

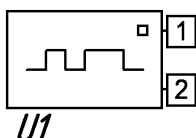


Рисунок 9 – Номера контрольных точек цифрового источника Stim2

Моделирование во временной области начинается по команде *Analysis/Transient...*. В появляющемся диалоговом окне **Transient Analysis Limits** задают параметры, необходимые для проведения моделирования (рисунок 10).

В строке ввода **Time Range** указывают значение **40n** (40 нс) – такой же диапазон, как на временной диаграмме примера 4. Строка ввода **Maximum Time Step** должна содержать нулевое значение, в этом случае программа MicroCAP автоматически подберет необходимый шаг по времени. Опцию **Auto Scale Ranges** рекомендуется включить для автоматического масштабирования.

ния временной диаграммы на координатной плоскости. В нижней части диалогового окна располагается таблица, в которой указывают, сколько предполагается получить координатных плоскостей, временных диаграмм, какие величины будут откладываться по осям абсцисс и ординат.

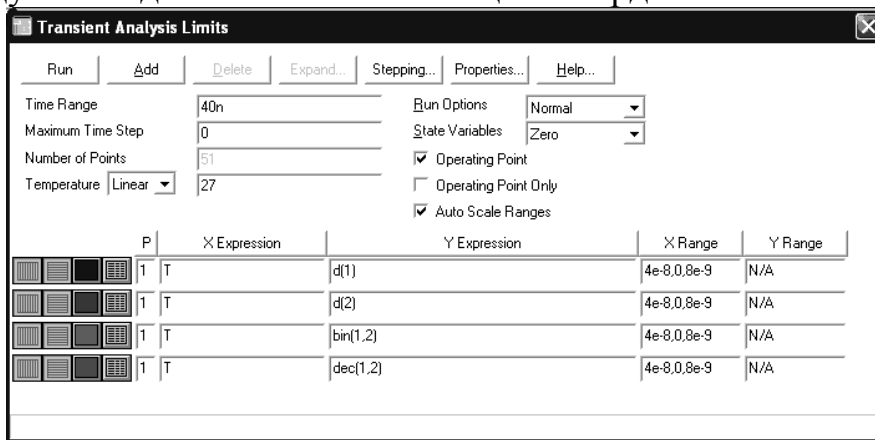


Рисунок 10 – Диалоговое окно задания параметров моделирования во временной области

Для нашего случая таблица должна содержать четыре строки. Добавить недостающую или убрать лишнюю строку можно с помощью кнопок **Add** и **Delete** в верхней части окна. В ячейки таблицы заносят следующую информацию:

P	X EXPRESSION	Y EXPRESSION
1	T	D(1)
1	T	D(1)
1	T	BIN(1,2)
1	T	DEC(1,2)

Учитывая, что опция **Auto Scale Ranges** активирована, столбцы **X Range** и **Y Range** в таблице не заполняются. Столбец **P** определяет количество координатных плоскостей одновременно выводимых на экран монитора. Если во всех ячейках столбца **P** содержится единица, то значит все временные диаграммы будут отражены на одной координатной плоскости. Столбец **X Expression** определяет, какая переменная или выражение будет откладываться по оси абсцисс. Переменная **T** – это резервированное обозначение времени в программе MicroCAP. В столбце **Y Expression** указывают, что будет откладываться по оси ординат.

Выражение вида d(...) обозначает логический уровень цифрового сигнала, где вместо многоточия ставят номер или имя интересующей контрольной точки. В нашем случае – это контрольные точки 1 и 2 (см. рисунок 9), т.е. **d(1)** и **d(2)**. В диалоговом окне **Transient Analysis Limits** существует возможность для представления многоразрядных цифровых сигналов в четырех системах счисления:


- bin(A,B,C,D,...) – двоичное представление цифрового слова, образованного разрядами A, B, C, D, ...; где A, B, C, D – номера или имена кон-

трольных точек. Перечисление идет в порядке убывания старшинства разрядов, т.е. А – старший разряд;

- oct(A,B,C,D,...) – восьмеричное представление цифрового слова;
- hex(A,B,C,D,...) – шестнадцатеричное представление цифрового слова;

- dec(A,B,C,D,...) – десятичное представление цифрового слова.

В последних двух строках таблицы добавлены выражения для представления двухразрядного сигнала в двоичной и десятичной системах счисления: **bin(1,2)** и **dec(1,2)**. Порядок перечисления номеров контрольных точек в выражении напрямую зависит от расстановки этих номеров на схеме. Так, на рисунке 9 старший разряд, имеющий ключ в виде квадратика, обозначен номером 1 и поэтому в скобках выражений он стоит на первом месте.

Для правильного представления на экране временных диаграмм необходимо убедиться, что внешний вид первых двух графических кнопок слева от столбца **P** таков: . Графический вид этих кнопок обозначает, что оси абсцисс и ординат представлены в линейном масштабе. Нажатие на кнопку **Run** в диалоговом окне приводит к построению временных диаграмм цифрового источника Stim2 (рисунок 11).

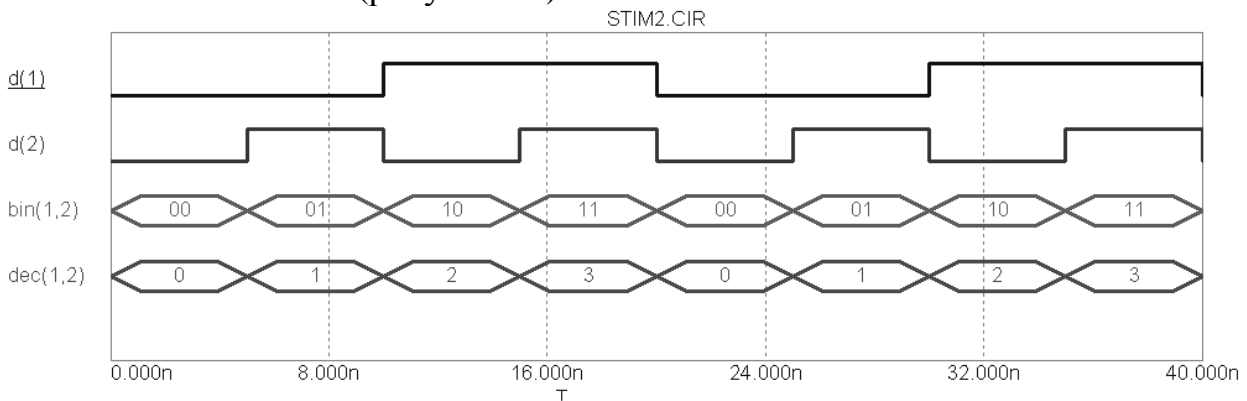
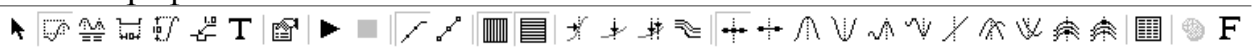


Рисунок 11 – Модифицированная временная диаграмма для примера 4

Действуя аналогичным образом, можно получить модифицированные временные диаграммы для примеров 5 и 6 (рисунки 12, 13). На этих диаграммах, помимо импульсных последовательностей, имеется шестнадцатеричная и десятичная форма представления восьмиразрядного сигнала: **hex(1,2,3,4,5,6,7,8)** и **dec(1,2,3,4,5,6,7,8)**.

Прием №4. Анализ временных диаграмм в режиме электронного курсора.

Программа MicroCAP имеет специальную панель инструментов для работы с графиками.



Эта панель доступна только во время отображения графиков на экране. Действия пиктограмм дублируются также пунктами меню *Options* и *Scope*, если график представлен как результат схемотехнического моделирования.

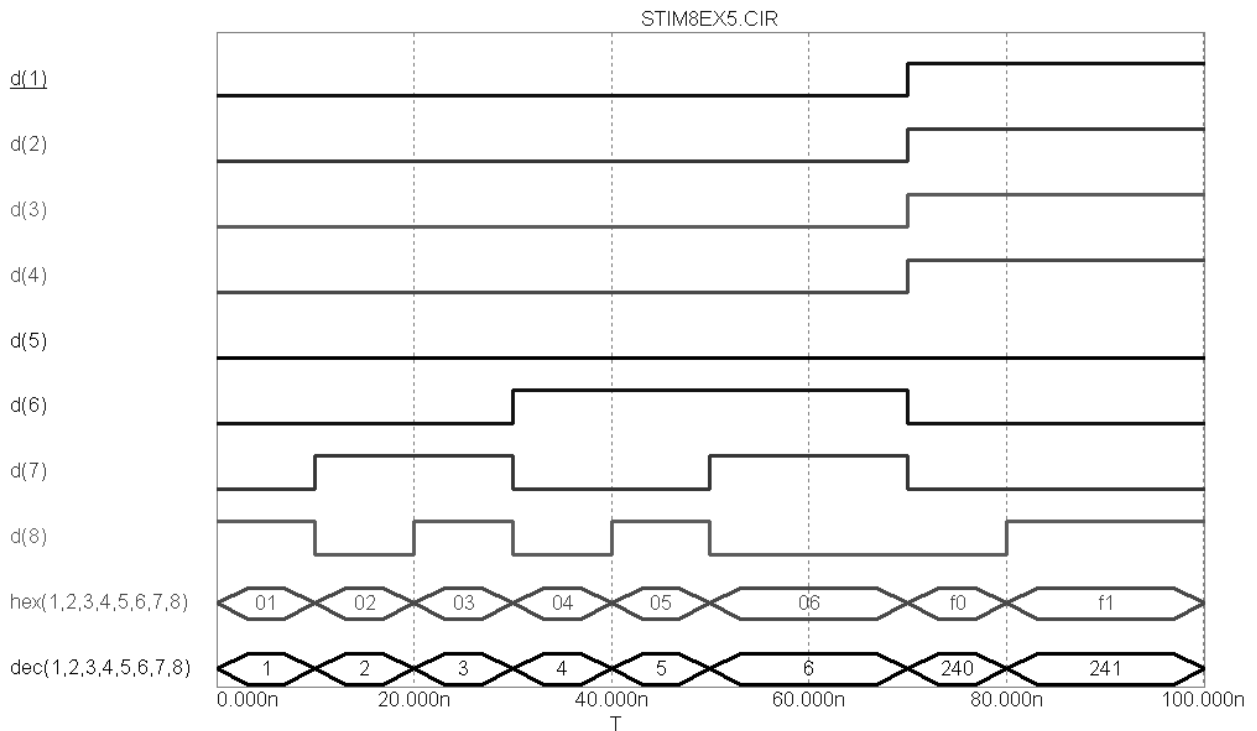


Рисунок 12 – Модифицированная временная диаграмма для примера 5

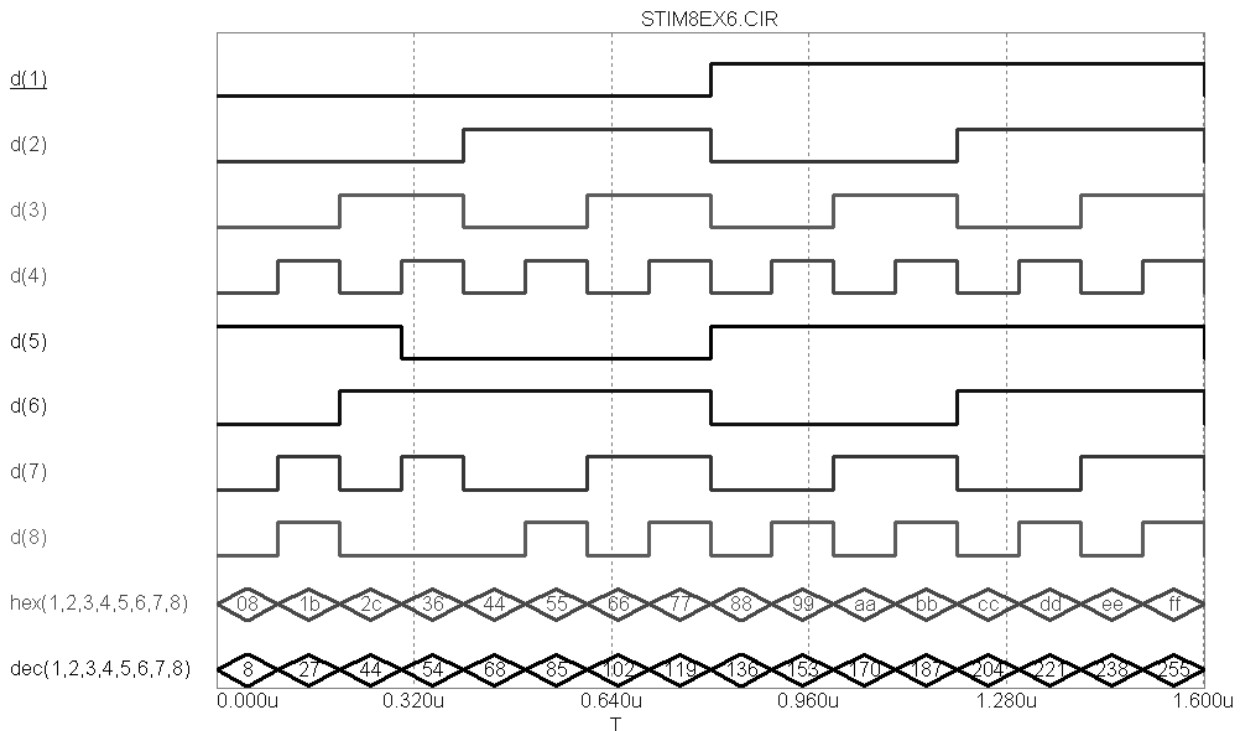







Рисунок 13 – Модифицированная временная диаграмма для примера 6


Поочередное нажатие на первые семь пиктограмм обуславливает следующие режимы работы с графиками:

1.  - режим выбора объектов (Select Mode), изображенных на экране;
2.  - режим масштабирования (Scale Mode), позволяет увеличивать произвольные фрагменты графиков; при этом следует сначала с помощью

мыши протянуть воображаемый прямоугольник над тем фрагментом, который нужно увеличить;

3.  - режим электронного курсора (Cursor Mode), позволяет считать координаты одной или двух точек на графике.

4.   - режимы измерения по горизонтали и вертикали (Horizontal Tag Mode и Vertical Tag Mode); позволяют оценивать расстояние между двумя выбранными точками графика;

5.  - режим нанесения на график значений координат выбранной точки (Tag Mode);

6. **T** - текстовый режим (Text Mode); позволяет наносить произвольный текст на графики, в том числе и по-русски.

Наиболее часто востребован режим электронного курсора. При его включении в крайней левой и правой позициях временной диаграммы появляются изображения курсоров с указанием текущих координат. Курсоры можно скачкообразно перемещать в любую точку временной диаграммы нажатием левой или правой кнопок мыши. Более плавного перемещения левого курсора можно добиться с помощью кнопок $\langle \rightarrow \rangle$ и $\langle \leftarrow \rangle$. Передвижение правого курсора осуществляется сочетанием клавиш $\langle \text{Shift} \rangle + \langle \rightarrow \rangle$ и $\langle \text{Shift} \rangle + \langle \leftarrow \rangle$. Электронные курсоры ассоциированы с той диаграммой, чье наименование в текущий момент подчеркнуто слева от координатной плоскости. Например, на рисунке 14 подчеркнуто hex(1,2,3,4,5,6,7,8). Переход электронных курсоров на какую-либо временную диаграмму происходит щелчком мыши по наименованию.

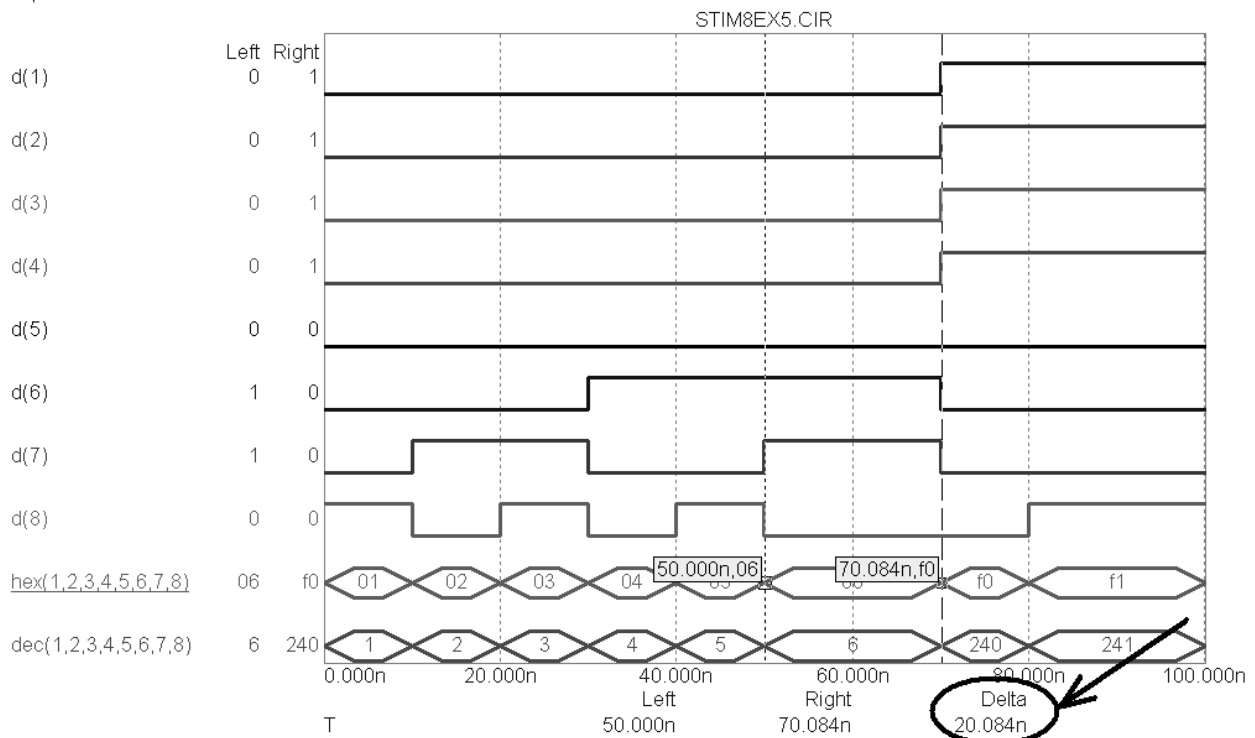



Рисунок 14 – Длительность состояния $(06)_{16}$, измеренная в режиме электронного курсора

Для точного позиционирования электронного курсора в заданную точку на диаграмме удобно пользоваться диалоговым окном **Go To X**. Диалоговое окно вызывается либо командой *Scope/Go To X*, либо нажатием на пиктограмму  на панели инструментов. В строке ввода **Value** одноименной вкладки указывают значение аргумента (момента времени) и нажимают на кнопку **Left** или **Right**. Кнопки **Left** и **Right** перемещают в заданную позицию X левый или правый курсор, соответственно.

В нижней части под координатной плоскостью располагается однострочная таблица, в которой размещаются значения аргумента, откладываемого по оси X (время). В колонках таблицы располагается информация:

- **Left** – значение момента времени, помеченного левым курсором;
- **Right** – значение момента времени, помеченного правым курсором;
- **Delta** – разность значений координат курсора.

Предположим, что на временной диаграмме примера 5 (рисунок 12), необходимо экспериментальным путем найти длительность состояния $(06)_{16}$. Поставленная задача выполняется в режиме электронного курсора:


1. Нажатием на пиктограмму  на панели инструментов переходят в режим электронного курсора.

2. Однократным щелчком мыши выбирают и делают активной временную диаграмму с наименованием **hex(1,2,3,4,5,6,7,8)**. Признаками активности являются подчеркнутое наименование и расположение электронных курсоров на этой диаграмме.

3. Левый электронный курсор с помощью кнопок $\langle \rightarrow \rangle$, $\langle \leftarrow \rangle$ или левого щелчка мыши устанавливают на левой границе состояния $(06)_{16}$, правый электронный курсор с помощью кнопок $\langle \text{Shift} \rangle + \langle \rightarrow \rangle$, $\langle \text{Shift} \rangle + \langle \leftarrow \rangle$ или правого щелчка мыши – на правой границе.

4. В нижней части окна под координатной плоскостью анализируют значение **Delta**.

Из рисунка 14 видно, что разница Delta между моментами времени, отмеченными левым и правым курсором, составляет 20 нс. Этот результат согласуется с теоретически предсказанным (см. прием №2, пример 5).

Замечание. Такой же результат можно получить, если воспользоваться режимом измерения по горизонтали . Протягивая указатель мыши при нажатой левой кнопке от левой до правой границы состояния $(06)_{16}$, будем иметь размерную стрелку с указанием расстояния.

Прием №5. Глобальные электрические цепи. Именованние электрических цепей.

Во многих электротехнических САПР существует понятие глобальной электрической цепи. Применительно к MicroCAP, глобальной можно считать любую электрическую цепь, которая отвечает двум требованиям:

1. Электрическая цепь состоит из нескольких отрезков проводников, визуальнo не связанных между собой.

2. Все отрезки проводников, принадлежащие глобальной электрической цепи, должны иметь одинаковое, данное пользователем, наименование.

Способ размещения на поле чертежа глобальных электрических цепей особенно удобен при изображении цифровых схем, потому что позволяет в ряде случаев упростить их «чтение» и понимание. Присвоение имени проводнику происходит двойным щелчком мыши на любом его участке. В появляющемся диалоговом окне указывают новое имя цепи.

Иногда именование проводников необходимо, чтобы присвоить уникальное (осмысленное) название тем контрольным точкам на схеме, в которых предполагается наблюдать сигнал. Обычно это контрольные точки, относящиеся ко входу или выходу схемы, линии групповой связи (шины). Несмотря на то, что программа MicroCAP в автоматическом режиме расставляет номера контрольных точек, их осмысленные названия позволяют проще записывать выражения в диалоговых окнах моделирования. Символы русского алфавита в названиях контрольных точек или цепей недопустимы.

Вновь обратимся к двухразрядному цифровому источнику сигнала (рисунок 9), моделирование которого проведено в приеме №3, а задание внутренних параметров источника – в приеме №2.

Модифицируем эту простейшую схему, состоящую из одного элемента. Присоединим к каждому выводу источника короткий отрезок проводника, затем изобразим еще пару проводников, не связанных с предыдущими (рисунок 15). Каждой паре, состоящий из присоединенного и свободного проводника, присвоим названия: **A1** и **A0**. Название **A1** должно соответствовать паре проводников, принадлежащих старшему разряду источника Stim2. Таким образом, модифицированная схема имеет две глобальные электрические цепи **A1** и **A0**, каждая из которых состоит из двух, визуальнo не связанных, проводников.

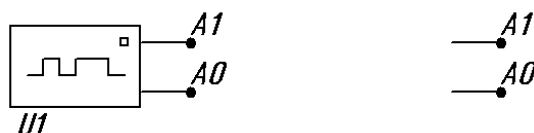


Рисунок 15 – Иллюстрация именовании электрических цепей

Моделирование схемы во временной области имеет небольшое отличие от описанного в приеме №3. В таблице диалогового окна **Transient Analysis Limits** теперь имеется возможность при записи выражений столбца **Y Expression** пользоваться именами контрольных точек, присвоенными нами на схеме:

P	X EXPRESSION	Y EXPRESSION
1	T	D(A1)
1	T	D(A0)
1	T	BIN(A1,A0)
1	T	DEC(A1,A0)

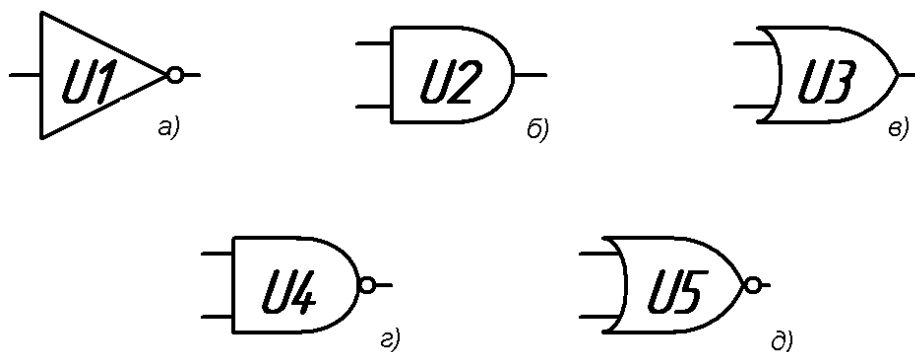
После нажатия на кнопку **Run** получим идентичные рисунку 11 временные диаграммы.

Замечание. Факт присвоения проводникам имен не отменяет обращение к ним по номерам, расставленным программой MicroCAP в автоматическом режиме.

Прием №6. Работа с логическими элементами И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ.

Библиотека программы MicroCAP насчитывает большое количество цифровых примитивов (логических вентилях), с полным перечнем которых можно ознакомиться в [1]. Для выполнения лабораторных заданий потребуются лишь небольшая их часть, а именно, логические элементы НЕ, И, ИЛИ, И-НЕ, ИЛИ-НЕ. Рассмотрим особенности работы с перечисленными элементами.

В связи с американским происхождением программы MicroCAP, условные графические обозначения логических элементов выполнены в библиотеке по стандарту США (рисунок 16). Для сравнения в Приложении 4 находится таблица соответствий условных графических обозначений элементов по ЕСКД и по стандарту США.



а) НЕ; б) И; в) ИЛИ; г) И-НЕ; д) ИЛИ-НЕ

Рисунок 16 – Условные графические обозначения некоторых логических элементов в программе MicroCAP

Для элементов И, ИЛИ, И-НЕ, ИЛИ-НЕ, помимо вариантов с двумя входами, в библиотеке доступны также 3, 4, 5 и 9-входные разновидности. Для размещения на поле чертежа перечисленных выше логических элементов служат команды:

- *Component/Digital Primitives/Standard Gates/Inverters/Inverter;*
- *Component/Digital Primitives/Standard Gates/And Gates/And X;*
- *Component/Digital Primitives/Standard Gates/Or Gates/Or X;*
- *Component/Digital Primitives/Standard Gates/Nand Gates/Nand X;*
- *Component/Digital Primitives/Standard Gates/Nor Gates/Nor X;*

где вместо символа X могут быть цифры 2, 3, 4, 5, 9, обозначающие количество входов.

Замечание. Из теории известно [3], что логический элемент НЕ может иметь только один вход.

В появляющемся после размещения элемента диалоговом окне свойств необходимо указать единственный параметр – название модели динамики,

т.е. временной модели распространения цифрового сигнала. Во всех лабораторных заданиях предполагается использование только идеальной модели динамики **D0_GATE** с нулевыми временами задержки сигнала. Для установки модели следует выбрать строку **TIMING MODEL** диалогового окна, а затем выбрать название из списка справа. При этом можно убедиться, что в нижней части диалогового окна все параметры модели **D0_GATE** имеют нулевое значение. Обычно после указания модели динамики для первого размещаемого элемента, все последующие элементы уже по умолчанию (без дополнительного запроса) имеют такую же модель.

Проведем исследование элементов НЕ, И, ИЛИ, И-НЕ, ИЛИ-НЕ в составе простейших схем.

Элемент НЕ. Составим электрическую схему из двух элементов: цифрового источника сигнала Stim1 и элемента НЕ (рисунок 17). В качестве внутренних параметров источника Stim1 можно взять описание бесконечной импульсной последовательности из примера 3 приема №2. Для большей правдоподобности изменим в описании сигнала наносекунды на микросекунды, т.е. заменим суффикс *n* на *u*:

```
.DEFINE EXAMPLE3
+ +0us 0
+ LABEL begin
+ +5us 1
+ +5us 0
+ +5us GOTO begin -1 TIMES
```

Модель динамики для элемента НЕ – **D0_GATE**. Входную и выходную контрольные точки обозначим как *in* и *out*.

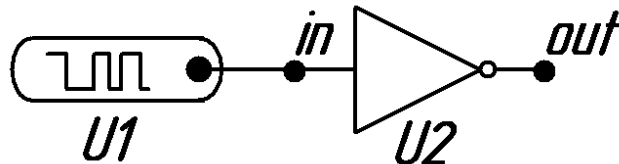


Рисунок 17 – Схема исследования элемента НЕ

В диалоговом окне задания на моделирование **Transient Analysis Limits** указываем:

- Time Range **20u** (20 мкс);
- Maximum Time Step **0**.

Рекомендуется включить опцию **Auto Scale Ranges**. В таблице диалогового окна **Transient Analysis Limits** приводят сведения:

P	X EXPRESSION	Y EXPRESSION
1	T	D(IN)
1	T	D(OUT)

Анализ временной диаграммы (рисунок 18), позволяет сделать вывод, что элемент НЕ работает в соответствие с таблицей истинности [3].

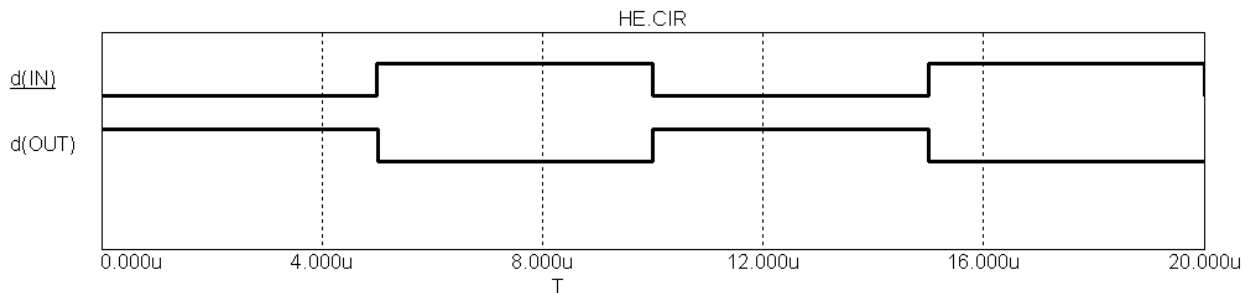


Рисунок 18 – Временная диаграмма работы элемента НЕ

Элемент И. Составим электрическую схему из цифрового источника сигнала Stim2 и двухвходового элемента И (рисунок 19). В качестве прототипа описания сигнала для источника Stim2 удобно взять пример 4 приема №2. В этом примере рассматривалась организация бесконечной последовательности состояний $(00)_2 \rightarrow (01)_2 \rightarrow (10)_2 \rightarrow (11)_2$. Снова заменим наносекунды на микросекунды в описании сигнала:

```
.DEFINE EXAMPLE4
+ LABEL begin
+ +0us 00
+ +5us 01
+ +5us 10
+ +5us 11
+ +5us GOTO begin -1 TIMES
```

Модель динамики для элемента И – D0_GATE. В схеме исследования элемента И имеются две входные контрольные точки *in1* и *in2*.

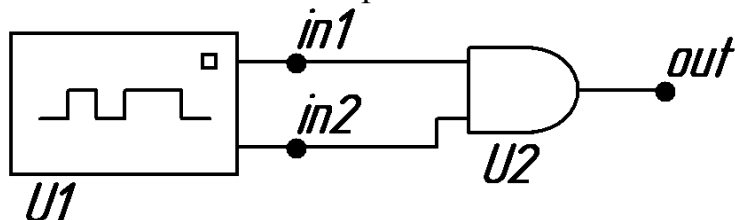


Рисунок 19 – Схема исследования элемента И

По сравнению с предыдущим исследованием, в таблице диалогового окна **Transient Analysis Limits** добавляется одна строка:

P	X EXPRESSION	Y EXPRESSION
1	T	D(IN1)
1	T	D(IN2)
1	T	D(OUT)

Анализ временной диаграммы (рисунок 20), позволяет сделать вывод, что элемент И работает в соответствии с таблицей истинности [3].

Если на схеме, изображенной на рисунке 19, последовательно заменять элемент И на ИЛИ, И-НЕ, ИЛИ-НЕ, то можно провести еще три исследования работы указанных элементов. Результаты исследований (рисунки 21-23) также согласуются с теорией [3].

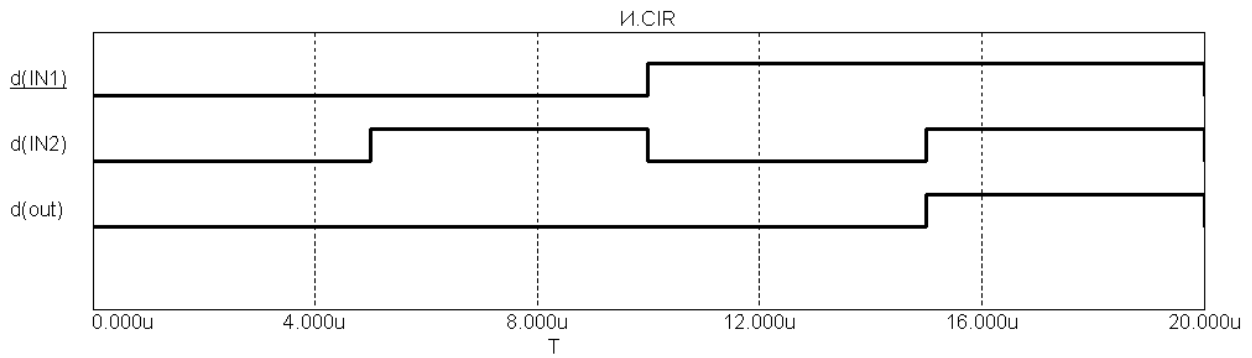


Рисунок 20 – Временная диаграмма работы элемента И

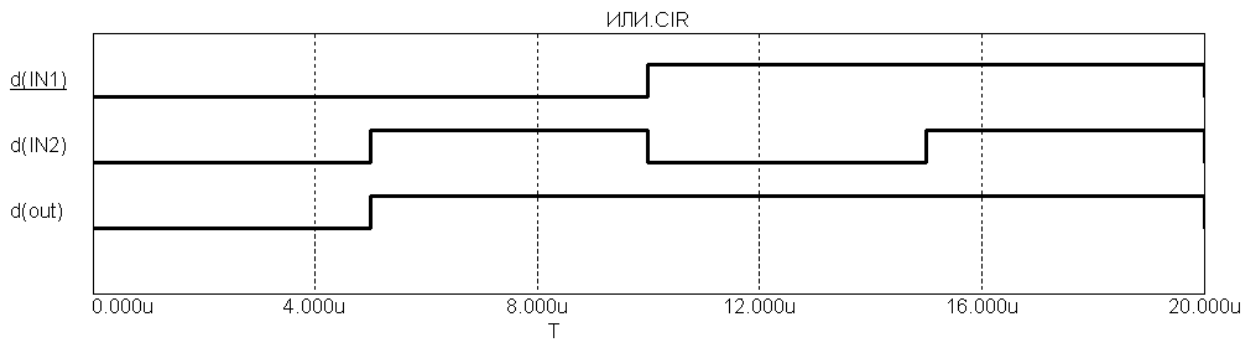


Рисунок 21 – Временная диаграмма работы элемента ИЛИ

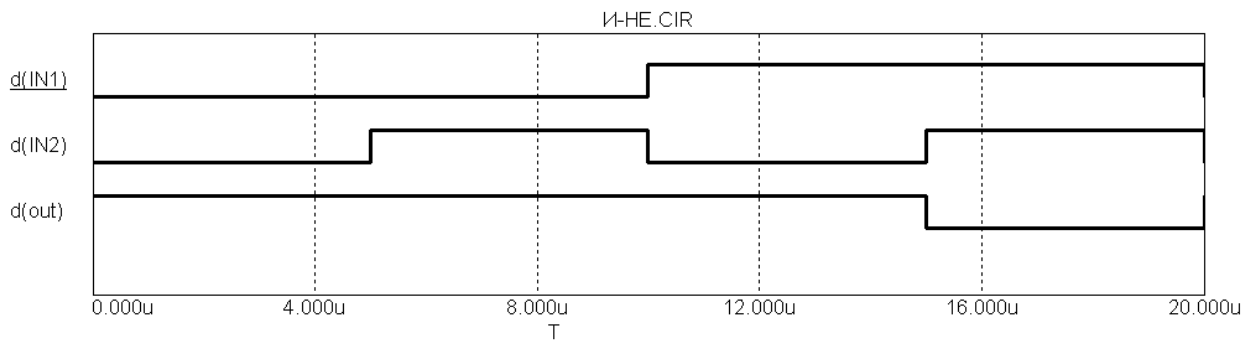


Рисунок 22 – Временная диаграмма работы элемента И-НЕ

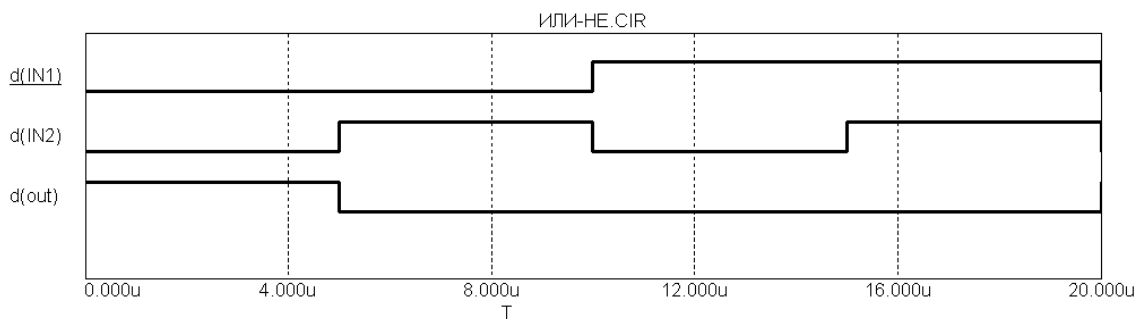


Рисунок 23 – Временная диаграмма работы элемента ИЛИ-НЕ

Прием №7. Применение мультиплексоров в цифровых схемах.

Известно [3], что мультиплексор коммутирует на выход у один из входов D_0, D_1, D_2, \dots , который выбирается (адресуется) двоичным кодом на адресных входах A_0, A_1, A_2, \dots .

Присутствующие в библиотеке MicroCAP многочисленные модели мультиплексоров, в отличие от рассмотренных выше логических элементов, имеют реальные прототипы в виде серийно выпускаемых интегральных микросхем. Такие модели уже содержат весь необходимый набор внутренних параметров, включая и времена задержки прохождения сигналов. Номенклатура мультиплексоров в MicroCAP только зарубежного производства, но при необходимости по справочникам можно найти отечественный аналог используемой микросхемы.

Модели мультиплексоров в библиотеке MicroCAP не образуют тематической группы, а рассеяны среди других цифровых элементов, что вызывает при поиске определенные неудобства. Учитывая это, при размещении на поле чертежа мультиплексоров удобно пользоваться окном поиска компонентов **Find Component**. Диалоговое окно вызывается по команде *Edit/Find Component*, в нем указывают название модели мультиплексора и нажимают на кнопку **Find**.

Рассмотрим принцип действия мультиплексора на примере простой схемы, предназначенной для преобразования параллельного кода в последовательный (рисунок 24). Схема состоит из цифрового источника сигнала Stim4, источников логических констант Fixed Digital и селектора мультиплексора на 8 каналов 74151A (отечественный аналог КМ155КП7).

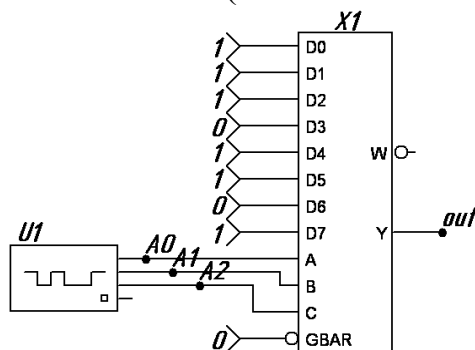


Рисунок 24 – Схема преобразования параллельного кода в последовательный

Размещение мультиплексора на поле чертежа можно сделать двояко: либо с помощью команды *Component/Digital Library/74xx120-/148-/74151A*; либо с помощью диалогового окна **Find Component**, указав в нем название микросхемы 74151A. На условном графическом обозначении мультиплексора имеются несколько выводов:

- D0...D7 – информационные входы;
- A, B, C – адресные входы; причем A – младший разряд;
- GBAR – инверсный вход разрешения работы;

- Y – прямой выход мультиплексора;
- W – инверсный выход мультиплексора.

На информационные входы D0...D7 мультиплексора подано цифровое слово $(10110111)_2$. С помощью цифрового источника Stim4 организована последовательность состояний с естественным порядком счета $0 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 7$. Такая последовательность дает возможность поочередной смены адресных кодов, а значит, позволяет поочередно обращаться к информационным входам. В результате данное цифровое слово $(10110111)_2$ будет выводиться последовательно, начиная с младшего разряда. На вход GBAR разрешения работы мультиплексора подан активный уровень логического нуля. Заметим, что в схеме на рисунке 24 старший разряд источника Stim4 не используется. Для организации последовательности состояний $0 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 7$ составим листинг описания в диалоговом окне свойств источника:

```
.DEFINE INPUT
+ 0us 0
+ LABEL begin
+ +1us INCR BY 1
+ +1us GOTO begin -1 TIMES
```

Присвоим имена A2, A1, A0 адресным линиям и out – прямому выходу мультиплексора.

В диалоговом окне задания на моделирование **Transient Analysis Limits** указывают:

- Time Range **8u** (8 мкс) – из расчета восьми состояний длительностью по 1 мкс;
- Maximum Time Step **0**.

Опцию **Auto Scale Ranges** рекомендуется включить.

В таблице диалогового окна приводят сведения:

P	X EXPRESSION	Y EXPRESSION
1	T	OCT(A2,A1,A0)
1	T	BIN(OUT)

Для удобства восприятия результатов адресный сигнал записан в таблице в восьмеричной системе счисления, а выходной сигнал – в двоичной.

Из рисунка 25 видно, что на выходе схемы организован последовательный вывод цифрового слова $(10110111)_2$, начиная с младшего разряда.

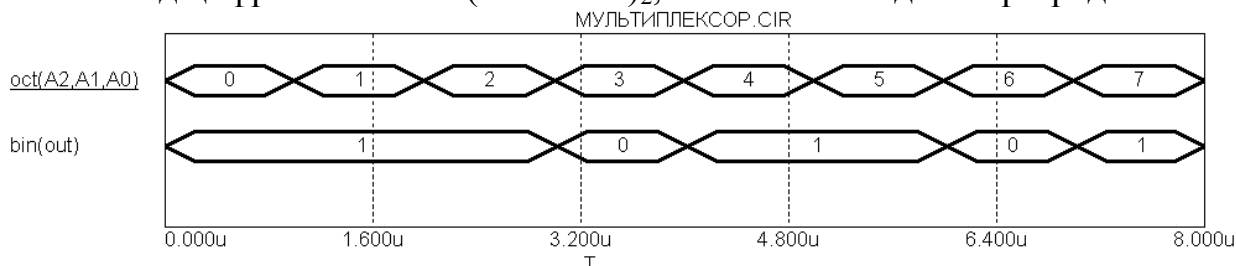


Рисунок 25 – Временная диаграмма последовательного вывода слова $(10110111)_2$, начиная с младшего разряда

Прием №8. Работа с логическим преобразователем программы Electronics Workbench (MultiSim).

Программа схемотехнического моделирования Electronics Workbench – разработка фирмы Interactive Image Technologies (Канада). Особенностью программы является наличие контрольно-измерительных приборов, по внешнему виду и характеристикам, приближенных к их промышленным аналогам. Панель контрольно-измерительных приборов в зависимости от версии программы может находиться и выглядеть по-разному. Так, для версии 2005 года MultiSim 8 панель контрольно-измерительных приборов плавающая (рисунок 26) и содержит большое количество пиктограмм: цифровой мультиметр, функциональный генератор, осциллограф и т.д.

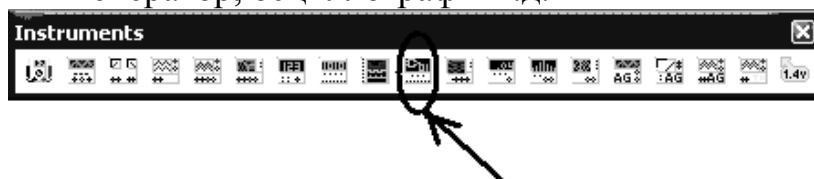


Рисунок 26 – Плавающая панель контрольно-измерительных приборов Electronics Workbench

Для выполнения лабораторных заданий представляет практический интерес виртуальный прибор под названием логический преобразователь (Logic Converter), отмеченный на рисунке 26 стрелкой. Внешний вид пиктограммы этого прибора в более ранних версиях программы иной, однако всплывающие подсказки названий приборов позволяют легко найти его в любых версиях.

Общий порядок работы с приборами таков: нажимается пиктограмма требуемого прибора, после чего на поле чертежа размещается его образ. Условное графическое обозначение логического преобразователя приведено на рисунке 27.

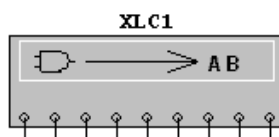


Рисунок 27 – Условное графическое обозначение логического преобразователя в Electronics Workbench

Для приведения прибора в рабочее (развернутое) состояние необходимо дважды щелкнуть указателем мыши по его графическому образу.

Внешний вид логического преобразователя показан на рисунке 28, на панели которого имеются:

- в верхней части – индикаторы восьми входов *A, B, C, D, E, F, G, H* и одного выхода *Out*;
- в средней части – таблица истинности;
- в правой части – кнопки управления процессом преобразования **Conversions**;
- в нижней части – строка ввода/вывода логических выражений.

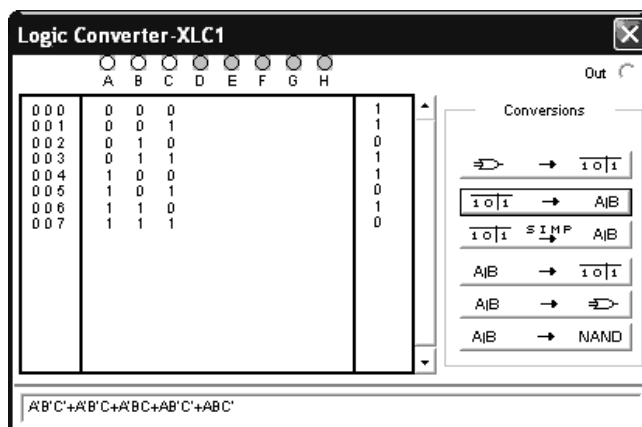


Рисунок 28 – Панель логического преобразователя

Рассмотрим некоторые полезные преобразования, выполняемые этим прибором.

Получение совершенной дизъюнктивной нормальной формы (СДНФ). Пусть описание функционирования логического устройства задано таблицей истинности (таблица 1), где x_2, x_1, x_0 – аргументы логической функции.

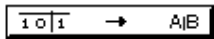
Таблица 1 – Исходная таблица истинности

X_2	X_1	X_0	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Требуется получить СДНФ по исходной таблице истинности.

Решение. Сделаем замену переменных: $x_2 \rightarrow A; x_1 \rightarrow B; x_0 \rightarrow C$. На лицевой панели логического преобразователя однократным щелчком мыши активируем три разряда A, B, C , после чего в средней части автоматически появится заготовка таблицы истинности из восьми возможных комбинаций аргументов. В левой части таблицы истинности имеется столбец с порядковыми номерами комбинаций, нумерация ведется с нуля.

Щелчком мыши устанавливаем значения логической функции в столбце *Out* логического преобразователя по таблице 1. Каждый щелчок мыши приводит к циклическому изменению значений $0 \rightarrow 1 \rightarrow X \rightarrow 0 \rightarrow \dots$, где X – неопределенное состояние.

После заполнения таблицы истинности значениями логической функции (см. рисунок 28) нажимаем вторую сверху кнопку  блока

Conversions. Получившийся результат – СДНФ логической функции - появляется в строке ввода/вывода:

$$A'B'C' + A'B'C + A'BC + AB'C' + ABC',$$

где символ ' обозначает инверсию.

Переходя к исходным переменным, имеем:

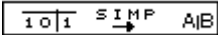
$$y = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1x_0 + x_2\bar{x}_1\bar{x}_0 + x_2x_1\bar{x}_0.$$

Замечание 1. В таблице истинности логического преобразователя младшим разрядом всегда считается крайний правый из активных разрядов.

Замечание 2. Строки комбинаций аргументов в таблице истинности логического преобразователя представляют собой возрастающую последовательность двоичных чисел от $(00\dots0)_2$ до $(11\dots1)_2$.

Если первоначально таблица истинности дана в ином виде, ее следует обязательно переписать на бумаге в соответствии с замечаниями 1 и 2.

Получение минимальной дизъюнктивной нормальной формы (МДНФ). Описание функционирования логического устройства задано таблицей истинности (таблица 1). Требуется получить МДНФ по исходной таблице истинности.

Решение. Аналогично предыдущему случаю заполняем таблицу истинности логического преобразователя. Нажимаем третью сверху кнопку  блока **Conversions**. Получившийся результат – МДНФ логической функции – появляется в строке ввода/вывода:

$$A'B' + A'C + AC'.$$

Переходя к исходным переменным, имеем:

$$y = \bar{x}_2\bar{x}_1 + \bar{x}_2x_0 + x_2\bar{x}_0.$$

Преобразование логического выражения к МДНФ. Часто исходным видом логической функции является не таблица истинности, а логическое выражение, причем в произвольной форме.

Пусть дано логическое выражение вида:

$$y = x_1x_0 + \bar{x}_2\bar{x}_1\bar{x}_0 + x_2\bar{x}_1x_0.$$

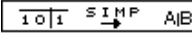
Требуется представить выражение в виде МДНФ.

Решение. Сделаем замену переменных $x_2 \rightarrow A$; $x_1 \rightarrow B$; $x_0 \rightarrow C$, т.к. работа с логическим преобразователем возможна только в алфавите $\{A, B, C, D, E, F, G, H\}$. В строке ввода/вывода логического преобразователя после замены переменных запишем:

$$(BC + A'B'C' + AB'C)'$$

Все символы, включая знак ', должны быть набраны в латинском регистре. Запись вида $(\dots)'$, обозначает инверсию всего выражения, находящегося в скобках.

Нажатие четвертой сверху кнопки  блока **Conversions** приводит к восстановлению таблицы истинности для записанного выражения.

Затем, нажимая на третью сверху кнопку  блока **Conversions**, в строке ввода/вывода получаем выражение в МДНФ:

$$AB' + B'C + BC'.$$

Переходя к исходным переменным, имеем:

$$y = x_2\bar{x}_1 + \bar{x}_1x_0 + x_1\bar{x}_0.$$

Прием №9. Применение JK-триггеров в цифровых схемах.

Известно, что триггеры – элементарные автоматы, содержащие собственно элемент памяти (фиксатор) и схему управления [3]. Триггеры типа JK универсальны, имеют входы установки (J) и сброса (K), подобные входам триггера RS. В отличие от последнего, JK-триггер допускает ситуацию с одновременной подачей сигналов на оба эти входа ($J = K = 1$). В этом режиме JK-триггер работает как счетный триггер относительно третьего (тактового) входа.

В библиотеке электрорадиоэлементов MicroCAP номенклатура JK-триггеров представлена шестью наименованиями, из которых пять являются моделями реальных интегральных микросхем ТТЛ-схемотехнологии, а один триггер – абстрактный логический примитив.

С целью изучения общих свойств JK-триггеров имеет смысл рассмотреть именно абстрактный примитив, не связанный с каким-либо реальным прототипом. Размещение на поле чертежа такого триггера происходит по команде *Component/Digital Primitives/Edge-Triggered Flip-Flops/JKFF*. В диалоговом окне свойств триггера JKFF укажем идеальную временную модель – с нулевыми задержками распространения сигнала. Для этого выбирают строку **TIMING MODEL** и в появляющемся справа списке находят название модели **D0_EFF**.

Триггер JKFF представляет собой синхронную разновидность с инверсным динамическим входом. Напомним, что инверсный динамический вход – это вход управления триггера с разрешением на переключение при изменении тактового сигнала с единичного значения на нулевое.

Условный графический образ синхронного триггера JKFF имеет 7 выводов (см. рисунок 29):

- J – информационный вход установки;
- K – информационный вход сброса;
- CLKB – инверсный динамический вход (реагирует только на отрицательный фронт синхроимпульса);
- PREB – инверсный вход предустановки триггера в начальное состояние (пассивная единица для этого входа означает нулевое состояние прямого выхода Q);
- CLRБ – инверсный вход сброса;
- Q – прямой выход;
- QB – инверсный выход.

Проиллюстрируем работу JK-триггера в четырех режимах и, соответственно, в четырех схемах включения (см. рисунки 29, 31, 34, 36):

1. Счетный режим.
2. Режим хранения.
3. Режим установки.
4. Режим сброса.

Общими элементами в нижеследующих схемах включения кроме *JK*-триггера являются цифровые источники *Stim 1* для синхронизации и сброса, а также источник *Fixed Digital* пассивной логической единицы для предварительной установки *JK*-триггера в нулевое состояние.

Параметры сигнала синхронизации *Clock* взяты нами из приема № 6, где рассматривалась организация бесконечной импульсной последовательности с периодом в 10 мкс:

```
.DEFINE CLOCK
+ +0us 0
+ LABEL begin
+ +5us 1
+ +5us 0
+ +5us GOTO begin -1 TIMES
```

Сигнал сброса *Reset* представляет собой отрицательный строб-импульс длительностью 1 мкс и возникающий через 1 мкс после начала анализа во временной области. На входном языке *MicroCAP* такой сигнал записывается следующим образом:

```
.DEFINE RESET
+ 0us 1
+ 1us 0
+ 2us 1
```

Напомним, что оперативно оценить правильность заданного сигнала можно, нажав на кнопку **Plot** диалогового окна свойств источника *Stim1* (см прием №2).

Счетный режим. Для реализации счетного режима *JK*-триггера на информационные входы *J* и *K* подана логическая единица (рисунок 29). В схеме включения даны наименования контрольным точкам, в которых предполагается наблюдать сигналы: *clock*, *reset*, *out*, *nout*.

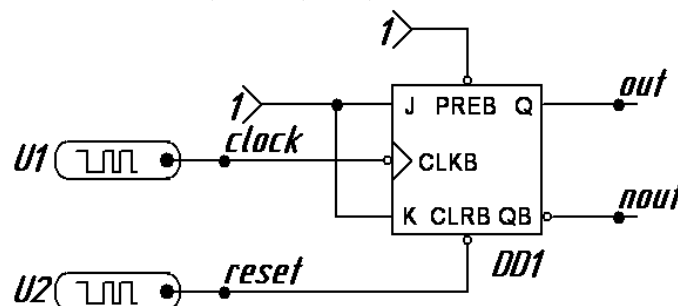


Рисунок 29 – Схема включения *JK*-триггера для счетного режима

В диалоговом окне задания на моделирование **Transient Analysis Limits** укажем длительность временного вида анализа в 100 мкс: **Time Range 100u**.

Кроме этого, в таблице диалогового окна перечислим наименования контрольных точек на схеме:

P	X EXPRESSION	Y EXPRESSION
1	T	D(RESET)
1	T	D(CLOCK)
1	T	D(OUT)
1	T	D(NOUT)

Из временной диаграммы (рисунок 30) видно, что приход каждого нового заднего фронта синхроимпульса *Clock* ведет к изменению прежнего состояния триггера на противоположное: $Q_H = \bar{Q}_{CT}$ при $J = K = 1$. Так реализуется счетный режим *JK*-триггера.

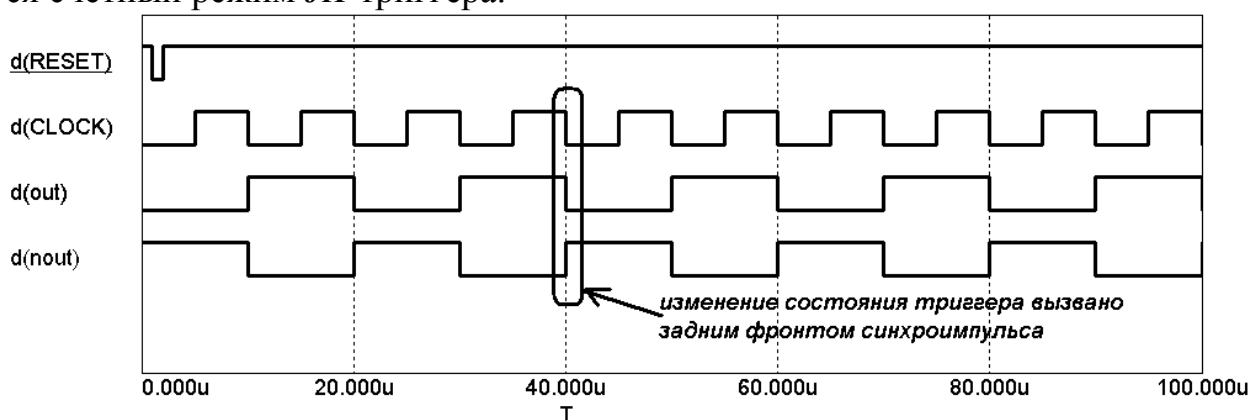


Рисунок 30 – Временная диаграмма счетного режима *JK*-триггера

Замечание. В счетном режиме частота изменения состояния триггера в 2 раза меньше частоты следования синхроимпульсов:

$$f_Q = \frac{f_{CLOCK}}{2}.$$

Режим хранения. Комбинация из двух пассивных сигналов, поданных на информационные входы *JK*-триггера, приводит к возникновению режима хранения. До тех пор, пока не появится хотя бы один активный сигнал на входе, *JK*-триггер будет сохранять свое последнее выходное состояние: $Q_H = Q_{CT}$ при $J = K = 0$. Чтобы продемонстрировать отдельно режим хранения нуля и единицы, реализуем два случая, которые будут отличаться только параметрами входного сигнала *In*. Вначале предположим, что в схеме включения (рисунок 31) входной сигнал *In* имел уровень логической единицы первые 47 мкс, а затем возник уровень логического нуля:

.DEFINE IN

+ 0us 1

+ 47us 0

Другими словами, первые 47 мкс имеет место счетный режим (т.к. сигнал *In* подается одновременно на оба входа), а затем – режим хранения.

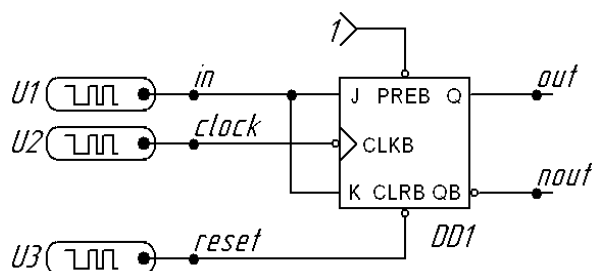


Рисунок 31 – Схема включения JK -триггера в режиме хранения

По временной диаграмме счетного режима (рисунок 30) видно, что в момент времени 47 мкс выходное состояние триггера было $Q = 0$. Следовательно, с возникновением режима хранения выходное состояние $Q = 0$ будет сохраняться неограниченно долго, в чем можно убедиться по временной диаграмме на рисунке 32.

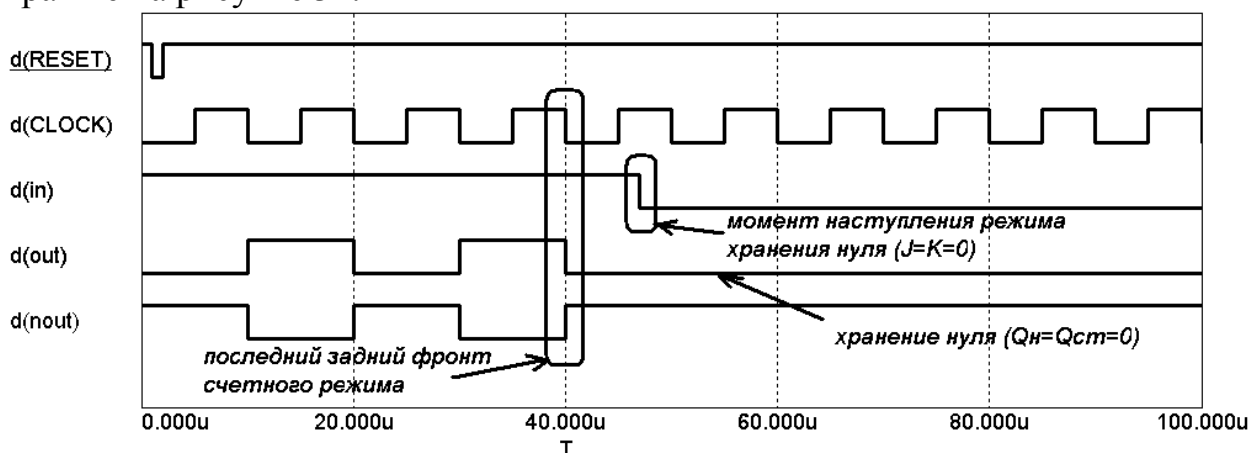


Рисунок 32 – Временная диаграмма хранения нуля JK -триггера

Для второго случая сигнал In изменяет свое логическое состояние единицы на ноль в момент времени 57 мкс:

.DEFINE IN

+ 0us 1

+ 57us 0

По временной диаграмме счетного режима (рисунок 30) можно видеть, что в этот момент времени выходное состояние триггера $Q = 1$. Рассуждая аналогично предыдущему случаю, приходим к выводу о возникновении режима хранения единицы (рисунок 33).

Замечание. Режим хранения для синхронных триггеров возникает также при отсутствии сигнала синхронизации, например при постоянном уровне сигнала $Clock$.

Режим установки. Комбинация активного ($J = 1$) и пассивного ($K = 0$) информационных сигналов для JK -триггера означает переход в режим установки. Вне зависимости от старого выходного состояния триггера, его новое состояние будет единичным: $Q_H = 1$ при $J = 1$ и $K = 0$. Схема включения для демонстрации режима установки содержит два источника логических кон-

стант Fixed Digital, которые присоединены к информационным входам J и K (рисунок 34).

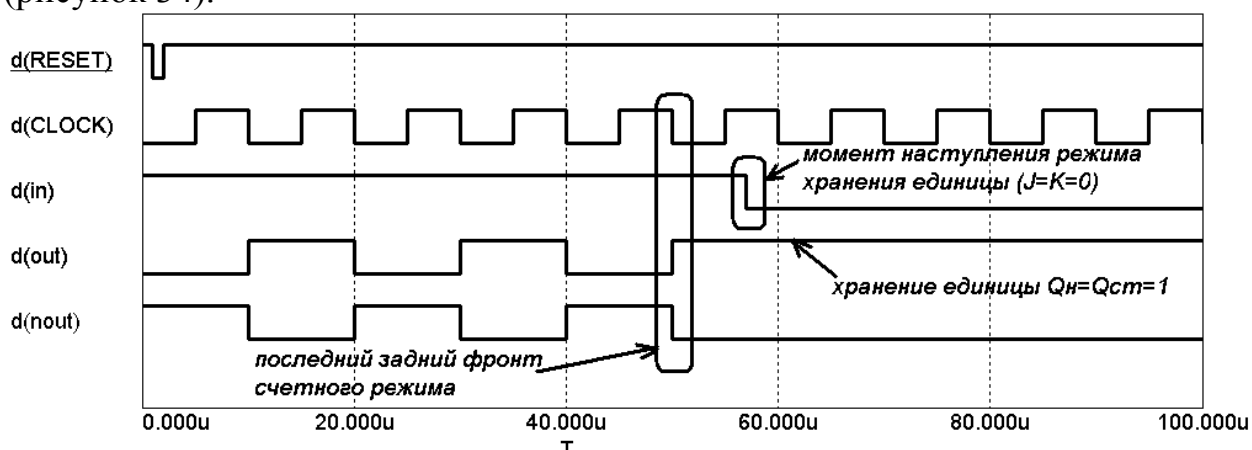


Рисунок 33 – Временная диаграмма хранения единицы JK -триггера

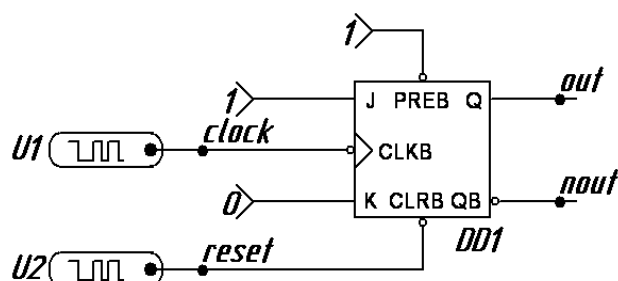


Рисунок 34 – Схема включения JK -триггера в режиме установки

В результате приход первого заднего фронта синхроимпульса приведет к возникновению в JK -триггере режима установки (рисунок 35).

Режим сброса. Режим сброса противоположен режиму установки как по физическому принципу действия, так и по комбинации активного и пассивного сигналов. Если $J = 0$ и $K = 1$, то новое состояние триггера будет безусловно нулевым: $Q_H = 0$. Схема включения для демонстрации режима сброса несколько сложнее, чем для режима установки (рисунок 36).

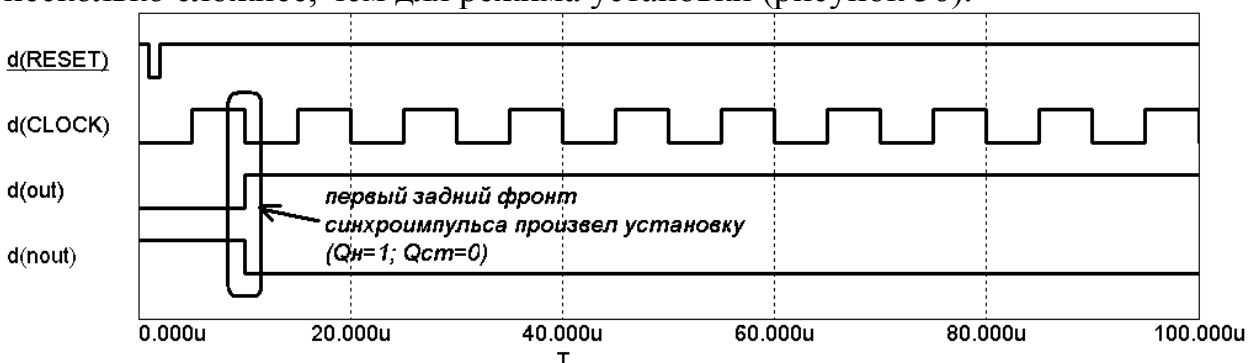


Рисунок 35 – Временная диаграмма режима установки JK -триггера

Предполагается, что первоначально имеет место счетный режим, из которого триггер переходит в режим сброса в момент времени 37 мкс. Для этой

цели в схеме введен источник сигнал In , присоединенный ко входу J и имеющий следующее описание:

```
.define IN  
+0us 1  
+37us 0
```

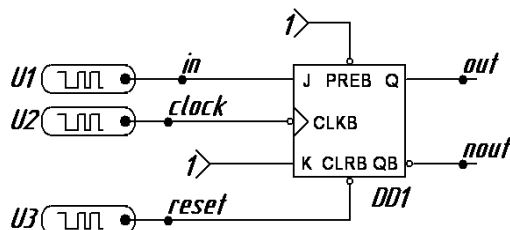


Рисунок 36 – Схема включения JK -триггера в режиме сброса

По временной диаграмме счетного режима (рисунок 30) видно, что в момент времени 37 мкс выходное состояние триггера единичное: $Q = 1$. Значит, первый после наступления режима сброса ($J = 0$ и $K = 1$) задний фронт синхроимпульса сбросит единичное состояние JK -триггера в ноль (рисунок 37).

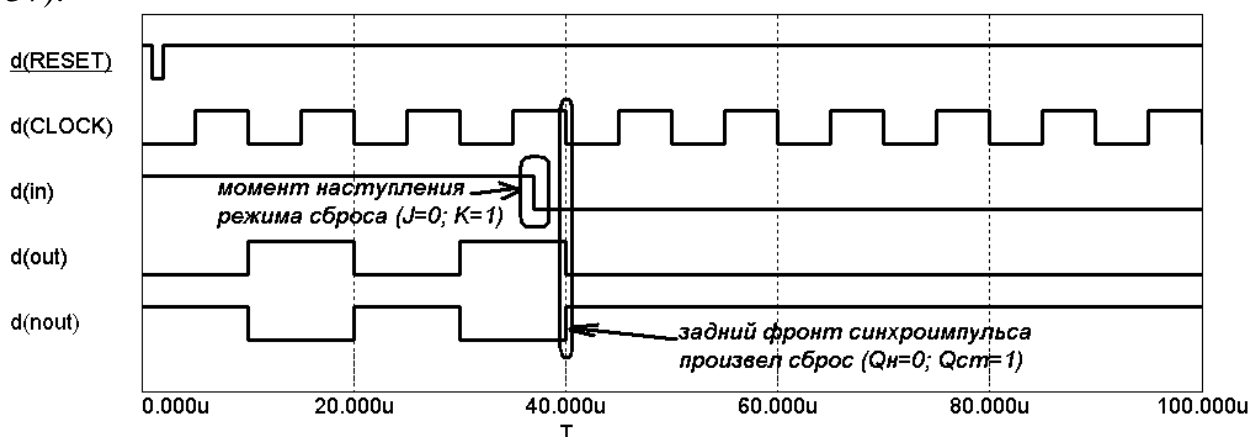


Рисунок 37 – Временная диаграмма режима сброса JK -триггера

Прием № 10. Применение D -триггеров в цифровых схемах.

D -триггер обычно снабжен только одним информационным входом. Это вход D , информация с которого по определению входа переписывается на выход триггера только по сигналу синхронизации [3]. Из сказанного следует, что D -триггер может быть только синхронным. Так как информация на выходе D -триггера остается неизменной вплоть до прихода очередного импульса синхронизации, такой триггер иногда называют защелкой.

В библиотеке электрорадиоэлементов MicroCAP представлено большое количество моделей D -триггеров, имеющих реальные прототипы в виде серийно выпускаемых интегральных микросхем. Среди них имеются D -триггеры как управляемые уровнем, так и управляемые фронтом. Для концептуального и поискового моделирования целесообразно применять логические примитивы, не имеющие реальных прототипов, в частности D -триггер с управлением уровнем LATCH и D -триггер с управлением фронтом DFF.

Рассмотрим подробнее особенности работы с D -триггером DFF, поскольку в лабораторных заданиях предполагается использование D -триггеров только с управлением фронтом.

Условное графическое обозначение D -триггера DFF наносится на поле чертежа по команде *Component/Digital Primitives/Edge-Triggered Flip-Flops/DFF*. В диалоговом окне свойств триггера DFF можно указать идеальную временную модель с нулевыми задержками распространения сигнала. Для этого, выбрав строку **TIMING MODEL**, необходимо найти в окне справа значение **D0_EFF**.

Условное графическое обозначение D -триггера насчитывает шесть выводов (см. рисунок 38):

- D – информационный вход триггера;
- PREB – инверсный вход предустановки триггера в начальное состояние (подача на него пассивного единичного уровня устанавливает выход триггера в нулевое состояние);
- CLRБ – инверсный вход сброса;
- CLK – прямой динамический вход синхронизации (триггер DFF воспринимает информацию на входе только в момент возникновения *переднего* фронта синхроимпульса);
- Q – прямой выход триггера;
- QB – инверсный выход триггера.

В отличие от JK -триггера D -триггер имеет только два режима работы: режим передачи данных и режим хранения. Продемонстрировать оба режима работы можно на простой схеме включения (рисунок 38). На схеме присутствуют собственно D -триггер, три источника Stim 1 для задания информационного сигнала, сигнала синхронизации и сигнала сброса, а также источник Fixed Digital для подачи пассивной единицы на вход предустановки.

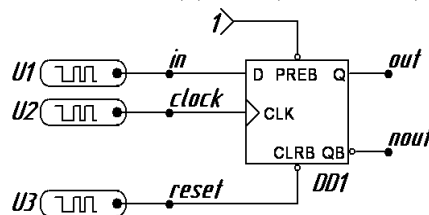


Рисунок 38 – Схема включения D -триггера

Параметры сигнала сброса полностью идентичны сигналу *Reset*, описанному в приеме №9. Предположим, что информационный сигнал представляет собой чередующуюся последовательность нулей и единиц с длительностью каждого состояния в 10 мкс:

```
.DEFINE IN
+ +0us 0
+ LABEL begin
+ +10us 1
+ +10us 0
```

+ +10us GOTO begin -1 TIMES

При этом на вход синхронизации первые 160 мкс от начала анализа поступают синхроимпульсы с периодом в 10 мкс, затем подача синхроимпульсов прекращается:

.DEFINE CLOCK

+ +0us 0

+ LABEL begin

+ +5us 1

+ +5us 0

+ +5us GOTO begin 15 TIMES

На схеме включения обозначены контрольные точки, в которых будет наблюдаться сигнал: *in*, *clock*, *reset*, *out*, *nout*.

В диалоговом окне задания на моделирование **Transient Analysis Limits** укажем длительность анализа во временной области в 200 мкс: Time Range **200u**. В таблице диалогового окна перечислим сигналы, подлежащие выводу на временную диаграмму:

P	X EXPRESSION	Y EXPRESSION
1	T	D(RESET)
1	T	D(CLOCK)
1	T	BIN(IN)
1	T	BIN(OUT)
1	T	BIN(NOUT)

На временной диаграмме (рисунок 39) можно видеть, что до момента времени 160 мкс имеет место режим передачи данных. Чередующаяся последовательность нулей и единиц передается на выход триггера, причем передача каждого нового состояния начинается только в момент возникновения переднего фронта синхроимпульса *Clock*. После прохождения 160 мкс от начала анализа синхроимпульсы прекращаются, что вызывает режим хранения. Последним переданным состоянием была единица ($Q = 1$), которая затем сохраняется в триггере неограниченно долго.

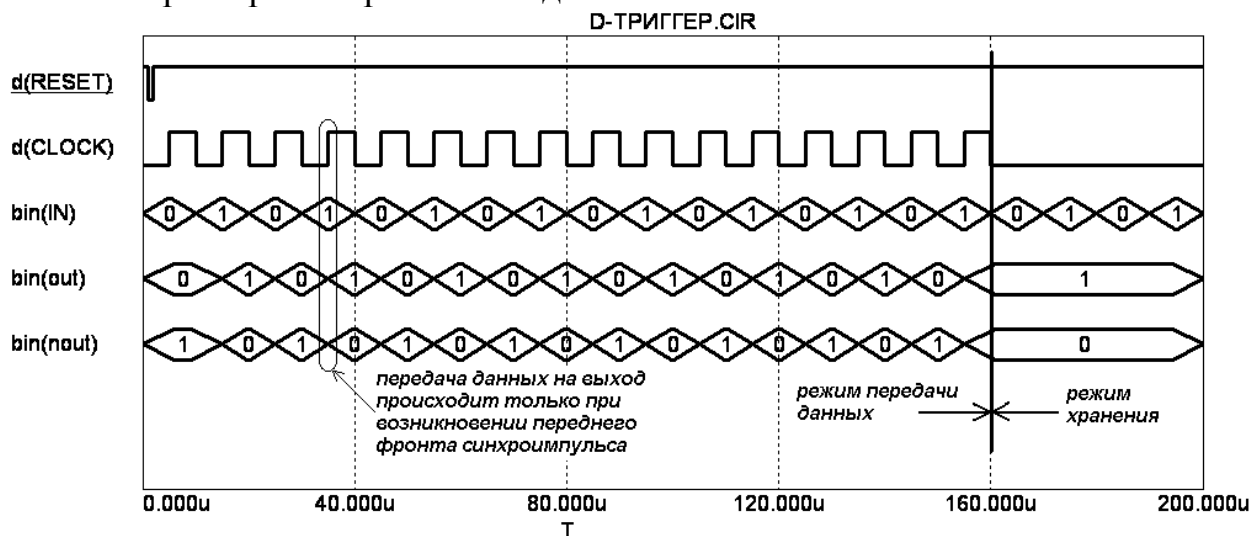


Рисунок 39 – Временная диаграмма работы D-триггера

1 ЛАБОРАТОРНАЯ РАБОТА №1 – СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ

1.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

- 1) знакомство с основными этапами синтеза комбинационных логических устройств;
- 2) изучение способов минимизации логических функций, представленных в совершенной дизъюнктивной форме;
- 3) приобретение навыков перехода в логический базис элементов И-НЕ, ИЛИ-НЕ;
- 4) построение структурных схем логических устройств по заданным функциям алгебры логики.

1.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с методическим примером, приведенным в пункте 1.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

1.3 Канонические формы представления логических функций.

Минимизация логических функций. Синтез логических устройств в заданных базисах

Синтез логического устройства распадается на несколько этапов. На первом этапе функцию, заданную в словесной, табличной или другой формах, требуется представить в виде логического выражения с использованием некоторого базиса. Дальнейшие этапы сводятся к получению минимальных форм функций, обеспечивающих при синтезе наименьшее количество электронного оборудования и рациональное построение функциональной схемы устройства. Для первого этапа обычно используется базис И, ИЛИ, НЕ независимо

от базиса, который будет использован для построения логического устройства.

Для удобства последующих преобразований приняты следующие две исходные канонические формы представления функций: совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ).

Дизъюнктивной нормальной формой (ДНФ) называется такая форма представления функции, при которой логическое выражение функции строится в виде дизъюнкции ряда членов, каждый из которых является простой конъюнкцией аргументов или их инверсией. Примером ДНФ может служить выражение:

$$f(x_3, x_2, x_1) = x_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_2. \quad (1.1)$$

Приведем форму представления функции, не являющуюся ДНФ. Например, функция

$$f(x_3, x_2, x_1) = x_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \overline{x_3 \cdot x_2}$$

представлена не в ДНФ, так как последний член не является простой конъюнкцией аргументов. Также не является ДНФ следующая форма представления функции:

$$f(x_3, x_2, x_1) = x_1 \cdot (\bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2) + x_3 \cdot x_2.$$

Если в каждом члене ДНФ представлены все аргументы (или их инверсии) функции, то такая форма называется СДНФ. Выражение (1.1) не является СДНФ, так как в нем лишь третий член содержит все аргументы функции.

Для перехода от ДНФ к СДНФ необходимо в каждый из членов, в которых представлены не все аргументы, ввести выражение вида $x_i + \bar{x}_i$, где x_i — отсутствующий в члене аргумент. Так как $x_i + \bar{x}_i = 1$, такая операция не может изменить значений функции. Покажем переход от ДНФ к СДНФ на примере простого выражения:

$$f(x_3, x_2, x_1) = x_1 + \bar{x}_3 \cdot x_2.$$

Добавление в члены выражений вида $x_i + \bar{x}_i$ приведет к функции:

$$\begin{aligned} f(x_3, x_2, x_1) &= x_1 \cdot (x_2 + \bar{x}_2) \cdot (x_3 + \bar{x}_3) + \bar{x}_3 \cdot x_2 \cdot (x_1 + \bar{x}_1) = \\ &= x_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1. \end{aligned}$$

На основании тождества алгебры логики $x + x = x$ имеем:

$$\bar{x}_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 = \bar{x}_3 \cdot x_2 \cdot x_1.$$

Отсюда после приведения подобных членов:

$$f(x_3, x_2, x_1) = x_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1,$$

т.е. имеем СДНФ.

Если исходная функция задана в табличной форме, то СДНФ может быть получена непосредственно. Пусть задана функция в форме таблицы 1.1.

Для этой функции СДНФ имеет вид:

$$f(x_3, x_2, x_1) = \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1. \quad (1.2)$$

Каждый член в (1.2) соответствует некоторому набору значений аргументов, при котором $f(x_3, x_2, x_1)$ равна 1. Каждый из наборов аргументов, при которых $f(x_3, x_2, x_1)$ равна 1 (третий, четвертый, шестой и восьмой столбцы наборов), обращает в единицу соответствующий член выражения (1.2), вследствие чего и вся функция оказывается равной единице.

Таблица 1.1 – Таблица истинности логической функции трех аргументов

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Можно сформулировать следующее правило записи СДНФ функции, заданной таблицей истинности. Необходимо записать столько членов в виде конъюнкций всех аргументов, сколько единиц содержит функция в таблице. Каждая конъюнкция должна соответствовать определенному набору значений аргументов, обращающему функцию в единицу, и если в этом наборе значение аргумента равно нулю, то в конъюнкцию входит инверсия данного аргумента. Следует отметить, что любая функция имеет единственную СДНФ.

Структурная схема логического устройства может быть построена непосредственно по канонической форме (например, СДНФ) реализуемой функции. Получающаяся при этом схема для функции (1.2) показана на рисунке 1.1. Недостаток такого метода построения структурных схем, обеспечивающего в общем правильное функционирование устройства, состоит в том, что получающиеся схемы чаще всего неоправданно сложные, требуют использования большого числа логических элементов, имеют низкие экономичность и надежность. Во многих случаях удается так упростить логическое выражение, не изменив функции, что соответствующая структурная схема оказывается существенно более простой. Методы такого упрощения функции называются методами *минимизации функций*.

Наиболее просто и наглядно *задача минимизации логической функции* решается с использованием ее *кубического представления*. Такое представление использует ограниченное число символов и поэтому применяется при автоматизации процессов логического проектирования цифровых интегральных схем.

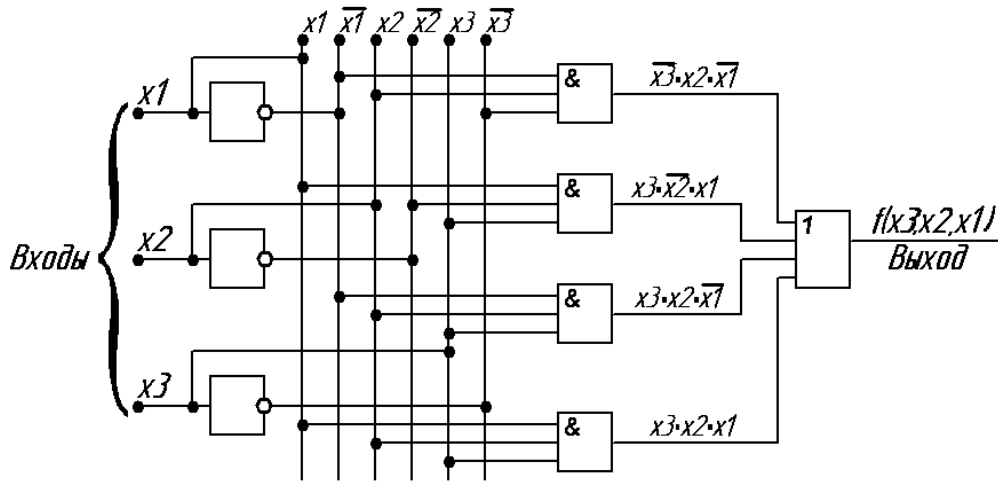
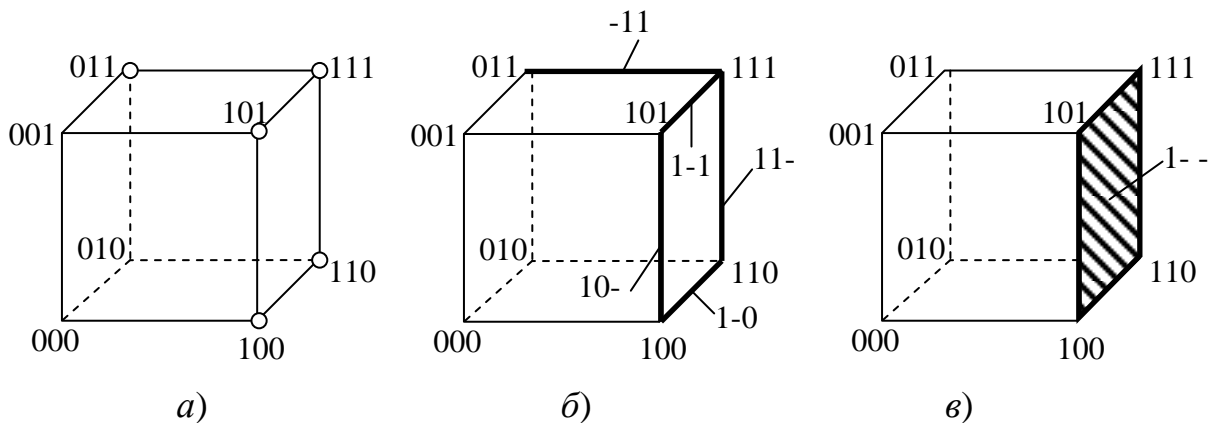


Рисунок 1.1 – Структурная схема устройства, построенная по СДНФ

Основой кубической формы является представление каждого набора входных переменных в качестве n -мерного вектора. Вершины этих векторов геометрически могут быть представлены как вершины n -мерного куба. Отмечая точками вершины векторов, для которых логическая функция равна единице, получаем геометрическое представление функции в виде куба. Пусть дана логическая функция трех аргументов:

$$f(x_3, x_2, x_1) = x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot x_2 \cdot x_1.$$

Графический образ данной функции в виде куба представлен на рисунке 1.2, а.



а) нулевой кубический комплекс; б) единичный кубический комплекс;
в) двоичный кубический комплекс

Рисунок 1.2 – Геометрическое представление логической функции

Очевидно, что наборы переменных, расположенные на концах ребер куба, отличаются только одной переменной. Такие наборы (коды) принято называть *соседними*. Каждую вершину куба, в которой функция принимает единичное значение, называют *нулевым кубом* (0-кубом). Записывается 0-куб последовательностью образовавших его входных переменных, т.е. кодом, соответствующим единице в таблице истинности. Множество нулевых кубов

образуют нулевой кубический комплекс. Например, $\mathbf{K}_0 = (011, 100, 101, 110, 111)$ (рисунок 1.2, *a*).

Если два нулевых куба комплекса \mathbf{K}_0 отличаются только по одной координате (переменной), т.е. два набора переменных, для которых логическая функция равна единице, являются соседними, то они образуют единичный куб (1-куб). Геометрически это соответствует ребру исходного n -мерного куба (рисунок 1.2, *b*), 1-куб записывается последовательностью общих элементов образовавших его 0-кубов с прочерком несовпадающих элементов. Множество единичных кубов образуют единичный кубический комплекс. Например, $\mathbf{K}_1 = (-11, 10-, 1-0, 11-, 1-1)$.

Аналогично, если два единичных комплекса \mathbf{K}_1 отличаются только по одной координате (переменной), то эти единичные кубы образуют двоичный куб (2-куб). Геометрически это соответствует грани исходного n -мерного куба (рисунок 1.2, *в*). 2-куб также записывается последовательностью общих элементов, образовавших его 1-кубов с прочерком несовпадающих элементов, а множество двоичных кубов образуют двоичный кубический комплекс. Например, $\mathbf{K}_2 = (1- -)$. И так далее.

Из сказанного следует, что размерность куба (его *ранг*) определяется числом несовпадающих координат, т.е. числом прочерков в его записи. Объединение кубических комплексов $\mathbf{K}_0, \mathbf{K}_1, \dots, \mathbf{K}_m$ для логической функции n переменных образует ее кубический комплекс:

$$\mathbf{K}(f) = \cup (\mathbf{K}_0, \mathbf{K}_1, \dots, \mathbf{K}_m).$$

Например, кубический комплекс логической функции, представленный на рисунке 1.2, имеет вид:

$$\mathbf{K}(f) = (011, 100, 101, 110, 111, -11, 10-, 1-0, 11-, 1-1, 1- -).$$

Из кубического комплекса $\mathbf{K}(f)$ всегда можно выделить множество кубов $\mathbf{\Pi}(f)$, таких, что каждый член комплекса \mathbf{K}_0 , т.е. вершина куба будет включен по крайней мере в один куб из множества $\mathbf{\Pi}(f)$. Множество кубов $\mathbf{\Pi}(f)$ называется *покрытием* комплекса $\mathbf{K}(f)$ или покрытием логической функции. Очевидно, что для любой логической функции существует несколько ее покрытий. В свою очередь, каждому покрытию $\mathbf{\Pi}(f)$, так же как и самому комплексу, соответствует своя дизъюнктивная нормальная форма, получаемая логическим суммированием логических произведений, соответствующих выделенным кубам функции.

Например, нулевой кубический комплекс (рисунок 1.2, *a*) включает все вершины куба, поэтому образует покрытие функции:

$$\mathbf{\Pi}_1(z) = \mathbf{K}_0 = (011, 100, 101, 110, 111).$$

Все вершины куба включаются также в единичный кубический комплекс \mathbf{K}_1 , поэтому он также образует покрытие функции:

$$\mathbf{\Pi}_2(z) = \mathbf{K}_1 = (-11, 10-, 1-0, 11-, 1-1).$$

Перебирая сочетания кубов различных рангов можно получить следующие покрытия функции:

$$\mathbf{\Pi}_3(z) = (011, 11-, 10-);$$

$$\mathbf{\Pi}_4(z) = (-11, 1-1, 1-0);$$

$$\mathbf{\Pi}_5(z) = (011, 1- -);$$

$$\mathbf{\Pi}_6(z) = (-11, 1- -) \text{ и так далее.}$$

Соответствующие указанным покрытиям ДНФ имеют вид:

$$f_1(x) = \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1;$$

$$f_2(x) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 + x_3 \cdot \bar{x}_1 + x_3 \cdot x_2 + x_3 \cdot x_1;$$

$$f_3(x) = \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot x_2 + x_3 \cdot \bar{x}_2;$$

$$f_4(x) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot \bar{x}_1;$$

$$f_5(x) = \bar{x}_3 \cdot x_2 \cdot x_1 + x_3;$$

$$f_6(x) = x_2 \cdot x_1 + x_3.$$

Сложность полученной таким образом ДНФ принято характеризовать понятием «цена покрытия» ($\mathbf{\Pi}_{\Pi}$), которая равна сумме цен всех кубов, составляющих данное покрытие $\mathbf{\Pi}(f)$: $\mathbf{\Pi}_{\Pi} = \sum \mathbf{\Pi}_{\mathbf{K}}$. В свою очередь, цена одного r -куба функции n -переменных определяется как разность полного числа входных переменных и ранга соответствующего куба, т.е. равна числу переменных в соответствующей дизъюнкции: $\mathbf{\Pi}_{\mathbf{K}} = n - r$. Так, для функции трех переменных цена 0-куба равна трем, а 2-куба – единице.

В соответствие со сказанным, задача минимизации логической функции сводится к поиску покрытия $\mathbf{\Pi}(f)$ кубического комплекса $\mathbf{K}(f)$, имеющего минимальную цену.

Например, среди перечисленных выше покрытий выражение $\mathbf{\Pi}_6(z) = (-11, 1- -)$ имеет минимальную цену:

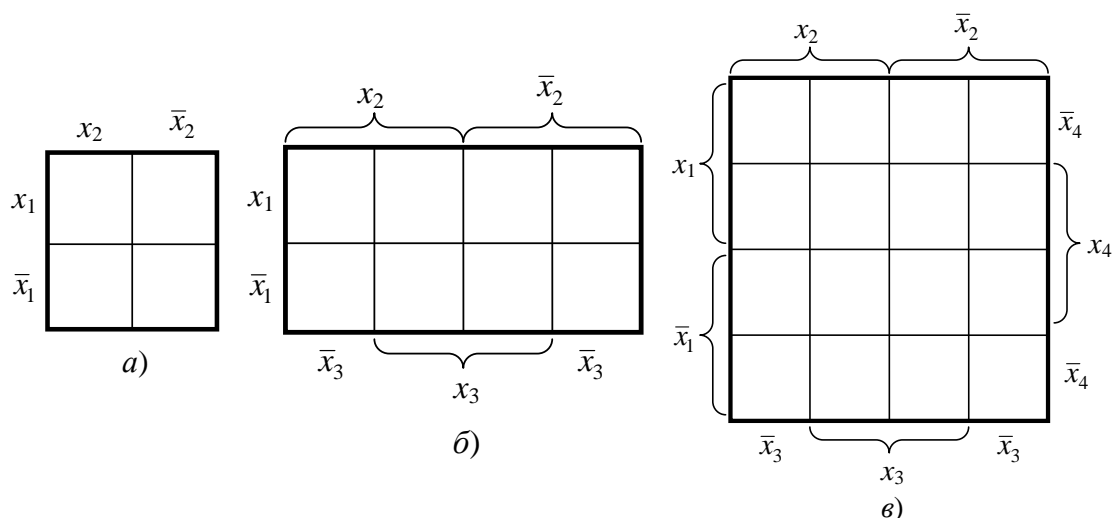
$$\mathbf{\Pi}_{\mathbf{K}1} = 3 - 1 = 2; \mathbf{\Pi}_{\mathbf{K}2} = 3 - 2 = 1; \mathbf{\Pi}_{\Pi} = \sum \mathbf{\Pi}_{\mathbf{K}} = 2 + 1 = 3.$$

Следовательно, логическая функция $f_6(x) = x_2 \cdot x_1 + x_3$ является минимальной дизъюнктивной нормальной формой (МДНФ).

Метод минимизации функции с помощью карт Вейча обеспечивает простоту получения результатов. Он используется при минимизации относительно несложных функций (с числом аргументов до пяти) ручным способом. Этот метод требует изобретательности и не может быть применен для решения задачи минимизации с помощью ЭВМ. Карта Вейча представляет собой определенную форму таблицы истинности. Таблицы, представленные на рисунке 1.3, являются картами Вейча для функций двух, трех и четырех аргументов.

Каждая клетка карты соответствует некоторому набору значений аргументов. Этот набор аргументов определяется присвоением значения логической I буквам, на пересечении строк и столбцов которых расположена клетка. Так, в карте функций четырех аргументов (рисунок 1.3, в) клетки первой строки соответствуют следующим комбинациям значений аргументов:

- первая клетка $x_1 = 1, x_2 = 1, \bar{x}_3 = 1$ ($x_3 = 0$), $\bar{x}_4 = 1$ ($x_4 = 0$);
- вторая клетка $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0$;
- третья клетка $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$;
- четвертая клетка $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0$.



a) две переменные; *б)* три переменные; *в)* четыре переменные
Рисунок 1.3 – Карты Вейча для функций нескольких переменных

Число клеток карты равно числу всех возможных наборов значений аргументов 2^n (n – число аргументов функций). В каждую из клеток карты записывается значение функции на соответствующем этой клетке наборе значений аргументов. Пусть логическая функция задана в виде таблицы истинности (таблица 1.2).

Таблица 1.2 – Таблица истинности логической функции трех аргументов

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Тогда таблица истинности этой функции в форме карты Вейча может быть представлена рисунком 1.4.

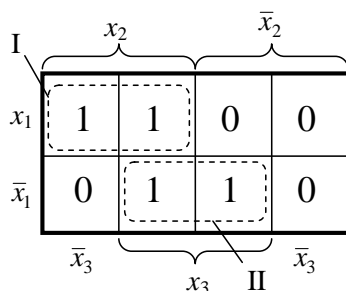


Рисунок 1.4 – Карта Вейча для логической функции трех переменных

Карта Вейча определяет значения функции на всех возможных наборах значений аргументов и является таблицей истинности. Карты Вейча компактны, но главное их достоинство состоит в следующем. При любом переходе из одной клетки в соседнюю вдоль столбца или строки изменяется значение лишь одного аргумента функции. Следовательно, если в паре соседних клеток содержится 1 , то над соответствующими им членами канонической формы может быть проведена операция склеивания. Таким образом, облегчается поиск склеиваемых членов.

Сформулируем правила получения МДНФ функций с помощью карт Вейча.

Все клетки, содержащие 1 , объединяются в замкнутые области. При этом каждая область должна представлять собой прямоугольник с числом клеток 2^k , где $k = 0, 1, 2, \dots$. Значит, допустимое число клеток в области $1, 2, 4, 8, \dots$. Области могут пересекаться, и одни и те же клетки могут входить в разные области. Затем проводится запись выражения МДНФ функции. Каждая из областей в МДНФ представляется членом, число букв в котором на k меньше общего числа аргументов функции n (т.е. равно $n - k$). Каждый член МДНФ составляется лишь из тех аргументов, которые для клеток соответствующей области имеют одинаковое значение (без инверсии, либо с инверсией).

Таким образом, при охвате клеток замкнутыми областями следует стремиться, чтобы число областей было минимальным (при этом минимальным будет число членов в МДНФ функции), а каждая область содержала возможно большее число клеток (при этом минимальным будет число букв в членах МДНФ функции).

Рассмотрим минимизацию с помощью карты Вейча функции трех аргументов, представленной в таблице 1.2. Все клетки, содержащие 1 , охватываются двумя областями. В каждой из областей 2^1 клеток, для них $n - k = 3 - 1 = 2$, и эти области в МДНФ будут представлены членами, содержащими по две буквы. Первой области соответствует член $x_2 \cdot x_1$ (аргумент x_3 здесь не присутствует, так как для одной клетки этой области он имеет значение без инверсии, для другой – с инверсией). Второй области соответствует член $x_3 \cdot \bar{x}_1$. Следовательно, МДНФ функции:

$$f(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_1.$$

При охвате клеток замкнутыми областями допускается сворачивание в цилиндр карты Вейча с объединением ее противоположных вертикальных сторон. В силу этого крайние клетки столбца таблицы рассматриваются как соседние и могут быть объединены в общую область. Карту Вейча функции четырех переменных можно представить объемной фигурой тора. Для такой карты допустимо сворачивание с объединением и вертикальных и горизонтальных сторон исходной развертки.

Прием уменьшения фактического числа входов логического элемента базируется на использовании тождества булевой алгебры $x + x = x$ или $x \cdot x = x$, поэтому на несколько входов логического элемента можно подавать одну и ту же логическую переменную. Следствием сказанного является важный практический вывод: если на все входы n -входового элемента И-НЕ или ИЛИ-НЕ подать один и тот же логический сигнал, то относительно этого сигнала элемент превращается в инвертор.

Построенные в соответствии с (1.3) и (1.4) схемы логических устройств приведены на рисунке 1.5.

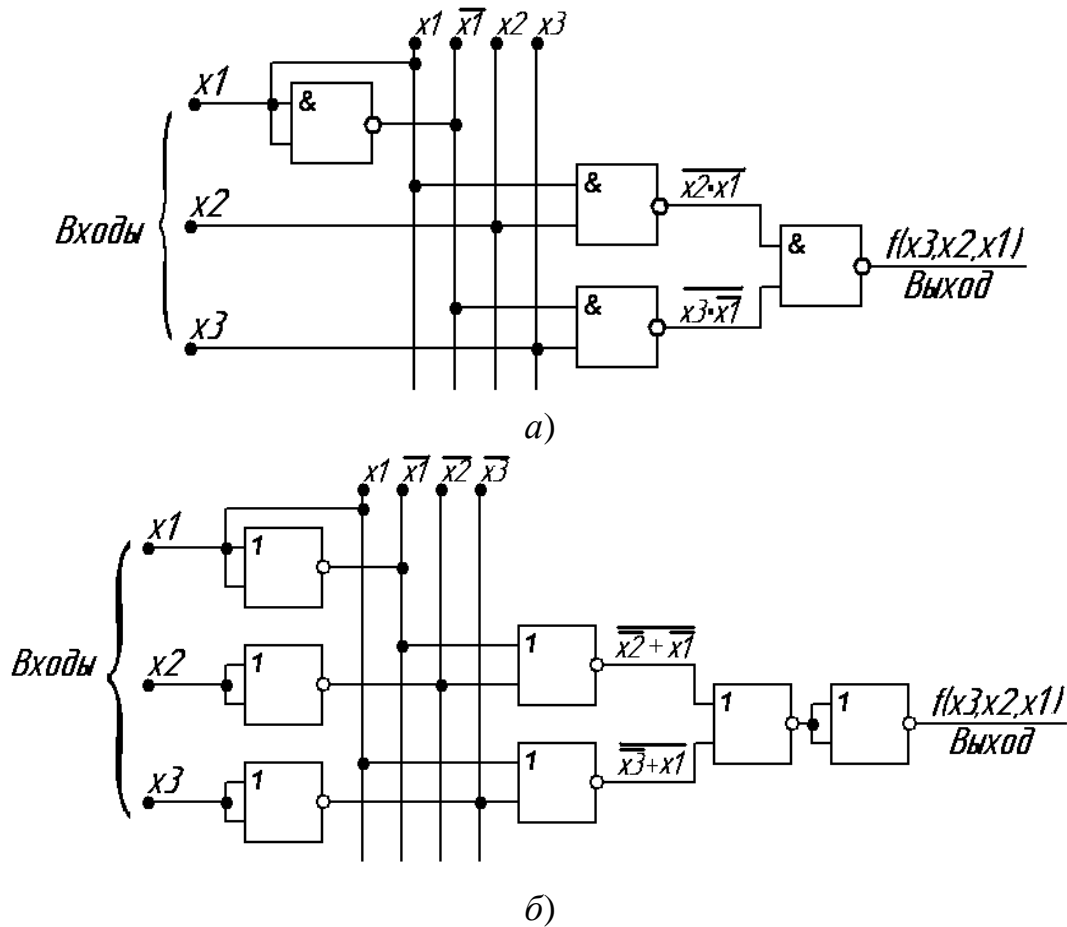


Рисунок 1.5 – Логические схемы устройств в базисе И-НЕ (а), ИЛИ-НЕ (б)

1.4 Пример минимизации логической функции и перехода в заданный базис

Логическая функция трех аргументов $f(x_3, x_2, x_1)$ задана в виде таблицы истинности (таблица 1.3).

Требуется для данной функции:

- построить структурную схему логического устройства на основе совершенной дизъюнктивной нормальной формы;

- построить структурную схему логического устройства на основе минимальной дизъюнктивной нормальной формы;
- построить структурную схему логического устройства в базисе логических элементов И-НЕ;
- построить структурную схему логического устройства в базисе логических элементов ИЛИ-НЕ.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Таблица 1.3 – Таблица истинности логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

I этап. Синтез логического устройства на основе СДНФ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP...».

Каждый из наборов аргументов, при которых логическая функция $f(x_3, x_2, x_1)$ равна единице (4, 6, 7, 8 строки таблицы истинности), является дизъюнктивным членом СДНФ. Следовательно, совершенная дизъюнктивная нормальная форма имеет вид:

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

Аппаратная реализация функции $f(x_3, x_2, x_1)$ в СДНФ потребует три инвертора (для выработки сигналов $\bar{x}_3, \bar{x}_2, \bar{x}_1$); четыре элемента 3И; один элемент 4ИЛИ. Для построения логической схемы необходимо логические элементы, предназначенные для выполнения операций, располагать от входа в порядке, определенном булевым выражением.

Структурная схема логического устройства на основе СДНФ, подготовленная в программе MicroCAP, представлена на рисунке 1.6.

Схема, предназначенная для моделирования в MicroCAP, дополнена четырехразрядным источником цифрового сигнала $U1$. Трехразрядных источников сигнала, как это требует наша схема, в MicroCAP не предусмотрено. Если число аргументов данной функции равно $n = 3$, то число различных сочетаний (наборов) аргументов составляет $2^n = 8$. Следовательно, для воспроизведения в MicroCAP всех сочетаний аргументов потребуется 8 тактов (циклов). Предположим, что длительность одного такта составляет $t = 1$ мкс, тогда

общая продолжительность анализа по времени будет $\sum_{i=1}^8 t_i = 8$ мкс. В нашем примере старший вывод источника $U1$ не используется, поскольку область определения логической функции находится в диапазоне $[0; 7]_{16}$.

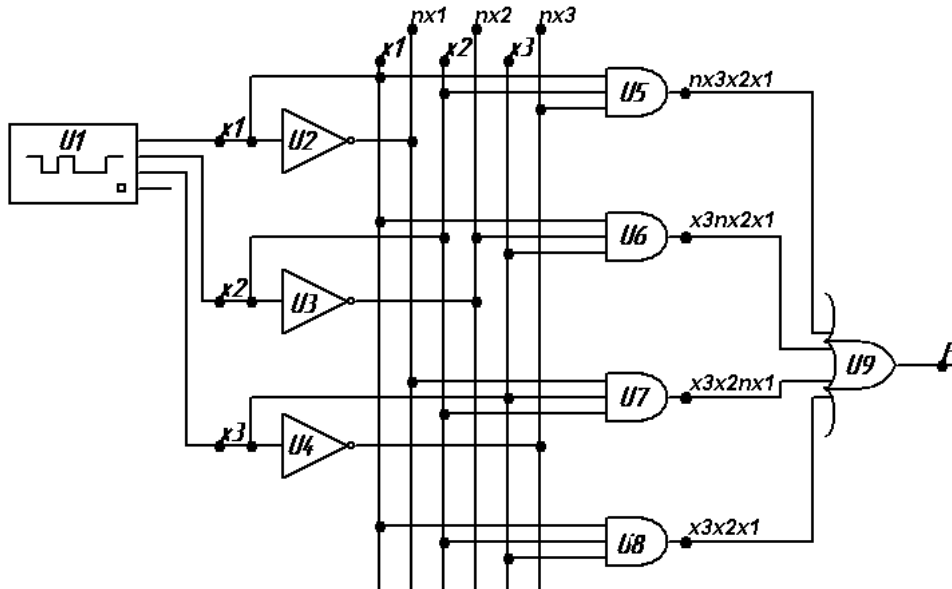


Рисунок 1.6 – Структурная схема логического устройства на основе СДНФ

В диалоговом окне свойств Stim 4 для источника $U1$ указывают информацию о выбранной системе счисления и о параметрах цифрового сигнала. В частности, в строке **FORMAT** указывают значение 4. Задать параметры цифрового сигнала для строки **COMMAND** можно несколькими способами. Рассмотрим наиболее простой способ – написание бесконечного программного цикла увеличения цифрового слова на единицу в каждом такте. При достижении максимального значения в выбранной системе счисления (число F) в следующем такте происходит обнуление. Листинг такой программы приведен ниже.

```
.DEFINE INPUT
+ 0us 0
+ LABEL begin
+ +1us INCR BY 1
+ +1us GOTO begin -1 TIMES
```

Правильность задания параметров цифрового сигнала оперативно проверим нажатием на кнопку **Plot** в диалоговом окне Stim 4 (рисунок 1.7). Временная зависимость разрядов D(2), D(3), D(4) представляет собой изменение трехразрядного слова согласно кодовым комбинациям таблицы истинности (таблица 1.3): 000, 001, 010, 011 и т.д.

В диалоговых окнах свойств логических элементов НЕ, ЗИ, 4 ИЛИ в строке **TIMING MODEL** необходимо выбрать из списка справа имя

D0_GATE. Выбранное имя соответствует идеальной модели динамики распространения цифрового сигнала – с нулевыми временами задержки.

На рисунке 1.6 входные зажимы электрической схемы обозначены как x_1 , x_2 , x_3 , а выход схемы – как F . В программе MicroCAP не предусмотрен символ инверсии. По этой причине пользователь должен самостоятельно придумать символьное обозначение инвертированных сигналов своей схемы. В нашем примере для именования инвертированных сигналов применена приставка n . Например, имя nx_1 на схеме означает инверсию сигнала x_1 , т.е. \bar{x}_1 .

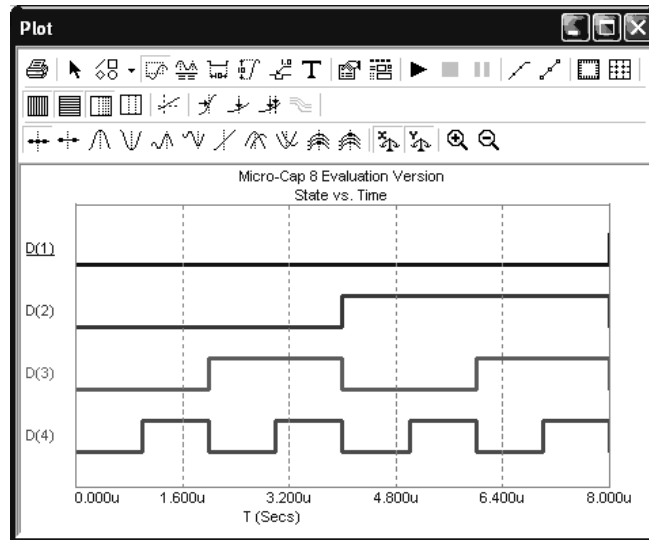


Рисунок 1.7 – Временная диаграмма цифрового сигнала

Моделирование во временной области проводится по команде *Analysis/Transient...* В появляющемся диалоговом окне **Transient Analysis Limits** указывают:

- в строке ввода Time Range **8u** – общая продолжительность временного вида анализа;
- нажатием на кнопку **Add** устанавливаются четыре табличные строки в нижней части диалогового окна;
- в столбце **P** всех строк таблицы записывается **1** – первый и единственный номер окна графика;
- в столбце **X Expression** всех строк таблицы записывается переменная **T** – резервированная переменная для обозначения времени;
- в столбце **Y Expression** в каждой из строк записывается по одному из выражений **d(x1)**, **d(x2)**, **d(x3)**, **d(f)**.

Опцию **Auto Scale Range** рекомендуется включить.

Нажатие на кнопку **Run** в диалоговом окне приводит к построению временной диаграммы работы устройства (рисунок 1.8). Нижняя зависимость $d(f)$ на графике представляет собой значения логической функции. Сравнивая их с таблицей истинности (таблица 1.3, правый столбец) приходим к выводу об адекватности проведенного моделирования.

II этап. Синтез логического устройства на основе МДНФ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP...».

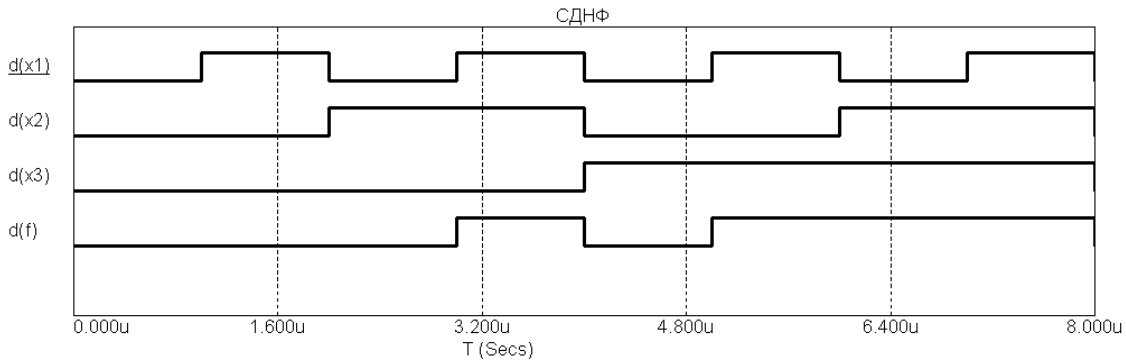
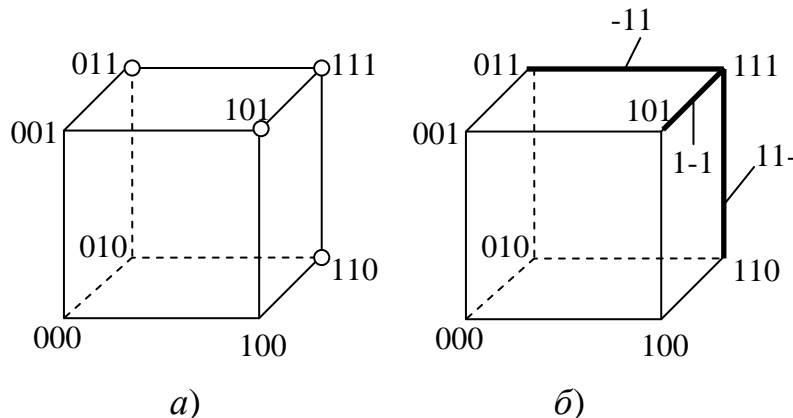


Рисунок 1.8 – Временная диаграмма работы схемы в СДНФ

Для проведения второго этапа исследования воспользуемся логической функцией в СДНФ, полученной на I этапе:

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

Графический образ такой логической функции представлен в виде куба на рисунке 1.9.



a) нулевой кубический комплекс; *б)* единичный кубический комплекс
Рисунок 1.9 – Геометрическое представление логической функции

Множество нулевых кубов образуют нулевой кубический комплекс $\mathbf{K}_0 = (011, 101, 110, 111)$. Множество единичных кубов образуют единичный кубический комплекс $\mathbf{K}_1 = (-11, 1-1, 11-)$. Двоичного кубического комплекса для данной функции не существует, поскольку нет ни одной грани, полностью образованной единичными кубами.

Кубический комплекс логической функции имеет вид:

$$\mathbf{K}(f) = (011, 101, 110, 111, -11, 1-1, 11-).$$

Нулевой кубический комплекс (рисунок 1.9, *a*) включает все вершины куба, поэтому образует покрытие функции:

$$\Pi_1(z) = \mathbf{K}_0 = (011, 101, 110, 111).$$

Все вершины куба включаются также в единичный кубический комплекс \mathbf{K}_1 (рисунок 1.9, *б*), поэтому он также образует покрытие функции:

$$\Pi_2(z) = \mathbf{K}_1 = (-11, 1-1, 11-).$$

Перебирая сочетания кубов различных рангов можно получить следующие покрытия функции:

$$\Pi_3(z) = (011, 1-1, 11-);$$

$$\Pi_4(z) = (101, -11, 11-);$$

$$\Pi_5(z) = (110, -11, 1-1);$$

$$\Pi_6(z) = (111, -11, 1-1, 11-) \text{ и так далее.}$$

Среди перечисленных выше (и существующих вообще) покрытий выражение $\Pi_2(z) = \mathbf{K}_1 = (-11, 1-1, 11-)$ имеет минимальную цену. Докажем это утверждение по формулам:

$$\Pi_{Ki} = n - r; \quad \Pi_{\Pi} = \sum \Pi_{Ki},$$

где n – количество переменных логической функции; r – количество прочерков в записи i -ого куба (ранг куба).

Для покрытия $\Pi_2(z)$ имеем:

$$\Pi_{K1} = 3 - 1 = 2; \quad \Pi_{K2} = 3 - 1 = 2; \quad \Pi_{K3} = 3 - 1 = 2; \quad \Pi_{\Pi} = \sum \Pi_{Ki} = 2 + 2 + 2 = 6.$$

Все остальные покрытия имеют бóльшую цену. Следовательно, логическая функция $f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2$, соответствующая покрытию $\Pi_2(z)$, является минимальной дизъюнктивной нормальной формой (МДНФ).

Проверим полученный результат другим способом – минимизацией с помощью карты Вейча. Карта Вейча для исходной функции в СДНФ представлена на рисунке 1.10. В каждую из клеток карты записывается значение функции на соответствующем этой клетке наборе значений аргументов.

	x_2		\bar{x}_2	
I				
x_1	1	1	1	0
\bar{x}_1	0	1	0	0
	\bar{x}_3	x_3	\bar{x}_3	
	III			

Рисунок 1.10 – Карта Вейча для исходной функции в СДНФ

Все клетки, содержащие единицы, охватываются тремя прямоугольными областями. В каждой из областей 2^1 клеток. Поскольку области могут пересекаться, то одни и те же клетки входят в разные области. Первой области соответствует член $x_2 \cdot x_1$; второй области – $x_3 \cdot x_1$; третьей области – $x_3 \cdot x_2$. Следовательно, МДНФ функции:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

Напомним, что при охвате клеток замкнутыми областями следует стремиться, чтобы число областей было минимальным (при этом минимальным будет число членов в МДНФ функции), а каждая область содержала возможно большее число клеток (при этом минимальным будет число букв в членах МДНФ функции).

Аппаратная реализация логической функции в МДНФ потребует три элемента 2И и один элемент 3ИЛИ. Структурная схема логического устройства на основе МДНФ, подготовленная в программе MicroCAP, представлена на рисунке 1.11.

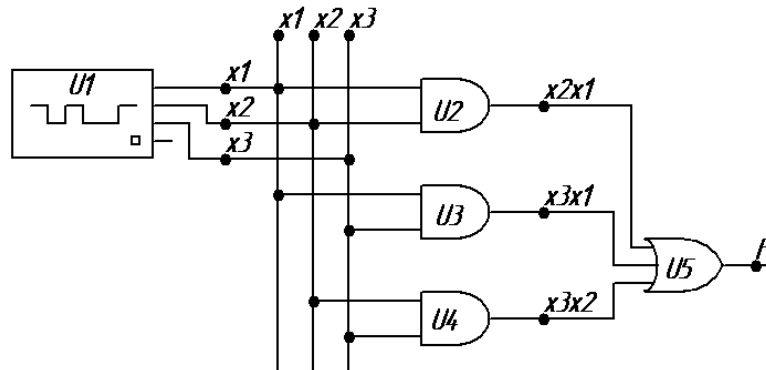


Рисунок 1.11 - Структурная схема логического устройства на основе МДНФ

Временной анализ схемы в MicroCAP проводится аналогично I этапу; результаты анализа представлены на рисунке 1.12. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

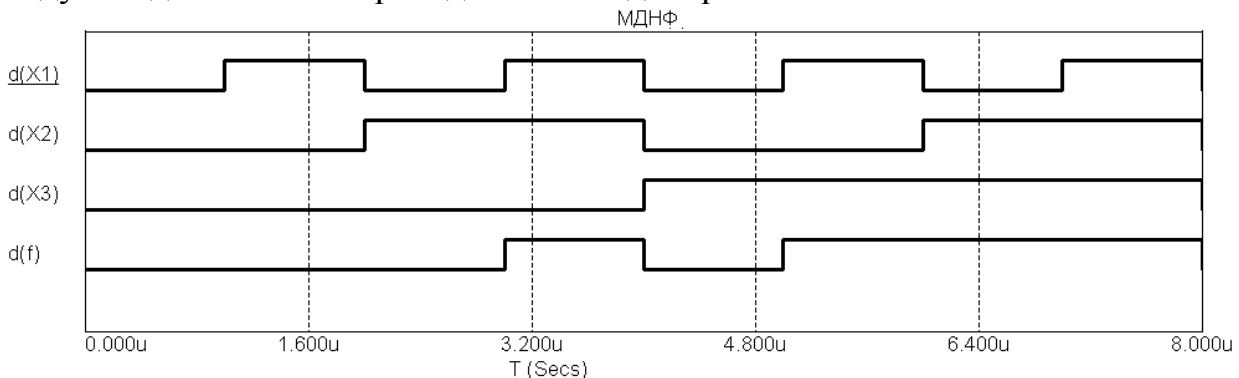


Рисунок 1.12– Временная диаграмма работы схемы в МДНФ

III этап. Синтез логического устройства в базисе И-НЕ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP...».

Для проведения третьего этапа исследования воспользуемся логической функцией в МДНФ, полученной на II этапе:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

Если требуется привести логическую функцию к базису логических элементов И-НЕ, то специальными приемами функция преобразуется к виду, содержащему только операции логического умножения и инверсии. Для перехода к базису И-НЕ выполняются следующие действия:

- дважды инвертируется правая часть выражения:

$$f(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2}};$$

- проводится преобразование по теореме де Моргана:

$$f(x_3, x_2, x_1) = \overline{x_2 \cdot x_1 \cdot x_3 \cdot x_1 \cdot x_3 \cdot x_2}.$$

Следовательно, преобразованная логическая функция в базисе логических элементов И-НЕ выглядит как:

$$f_{\text{И-НЕ}}(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1 \cdot x_3 \cdot x_1 \cdot x_3 \cdot x_2}}.$$

Аппаратная реализация такой логической функции потребует три элемента 2И-НЕ и один элемент 3И-НЕ. Структурная схема логического устройства в базисе И-НЕ, подготовленная в программе MicroCAP, представлена на рисунке 1.13.

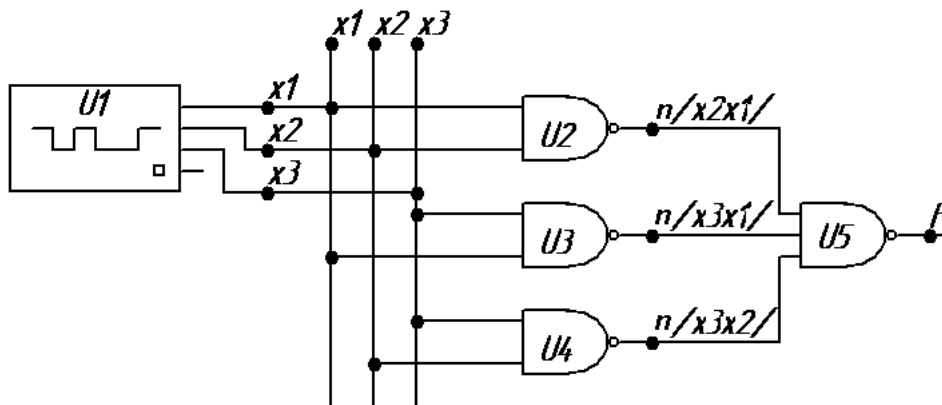


Рисунок 1.13 - Структурная схема логического устройства в базисе И-НЕ

Временной анализ схемы в MicroCAP проводится аналогично I этапу; результаты анализа представлены на рисунке 1.14. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

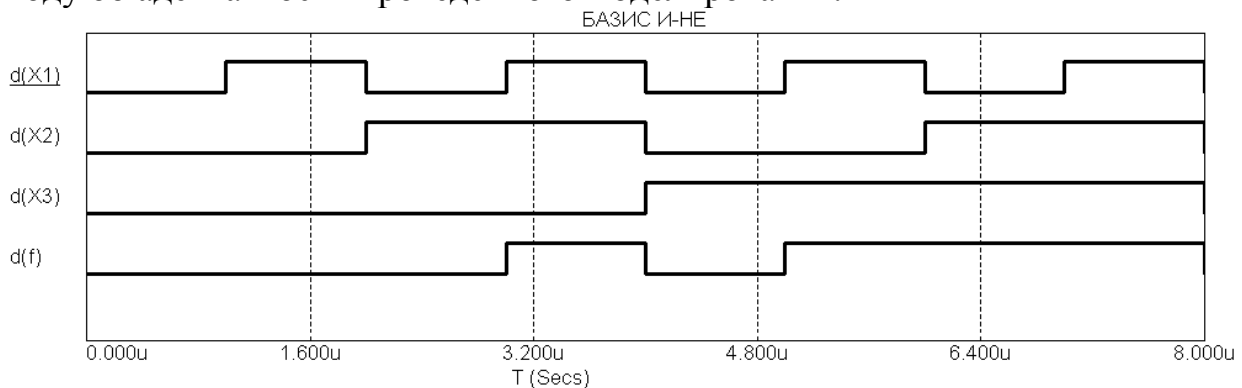


Рисунок 1.14 – Временная диаграмма работы схемы в базисе И-НЕ

IV этап. Синтез логического устройства в базисе ИЛИ-НЕ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP...».

Для проведения четвертого этапа исследования воспользуемся логической функцией в МДНФ, полученной на II этапе:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

Если требуется привести логическую функцию к базису логических элементов ИЛИ-НЕ, то специальными приемами функция преобразуется к виду, содержащему только операции логического сложения и инверсии. Для перехода к базису ИЛИ-НЕ выполняются следующие действия:

- инвертируются обе части выражения:

$$\overline{f(x_3, x_2, x_1)} = \overline{x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2};$$

- каждый дизъюнктивный член выражения дважды инвертируются:

$$\overline{\overline{\overline{x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2}}};$$

- проводится преобразование по теореме де Моргана:

$$\overline{\overline{\overline{x_2 + \overline{x_1} + \overline{x_3} + \overline{x_1} + \overline{x_3} + \overline{x_2}}}};$$

- для получения неинвертированного значения функции f на выходе устройства добавляется инвертор в базисе ИЛИ-НЕ:

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

Таким образом, преобразованная логическая функция в базисе логических элементов ИЛИ-НЕ выглядит как:

$$f_{\text{ИЛИ-НЕ}}(x_3, x_2, x_1) = \overline{\overline{\overline{\overline{\overline{\overline{x_2 + \overline{x_1} + \overline{x_3} + \overline{x_1} + \overline{x_3} + \overline{x_2}}}}}}}}.$$

Аппаратная реализация такой логической функции потребует семь элемента 2ИЛИ-НЕ, среди которых четыре элемента нужны для выработки инверсных сигналов ($\overline{x_1}, \overline{x_2}, \overline{x_3}, f$), и один элемент 3ИЛИ-НЕ. Напомним, что если на все входы n -входного элемента ИЛИ-НЕ подать один и тот же логический сигнал, то относительно этого сигнала элемент превращается в инвертор. Структурная схема логического устройства в базисе ИЛИ-НЕ, подготовленная в программе MicroCAP, представлена на рисунке 1.15.

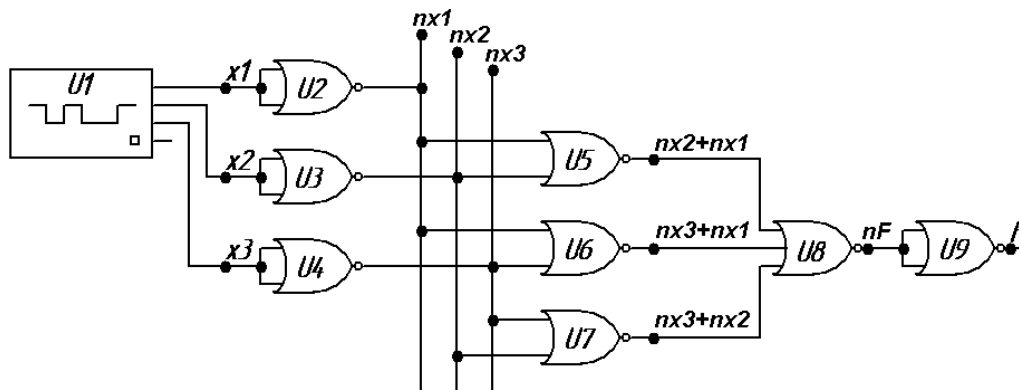


Рисунок 1.15 - Структурная схема логического устройства в базисе ИЛИ-НЕ

Временной анализ схемы в MicroCAP проводится аналогично I этапу; результаты анализа представлены на рисунке 1.16. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

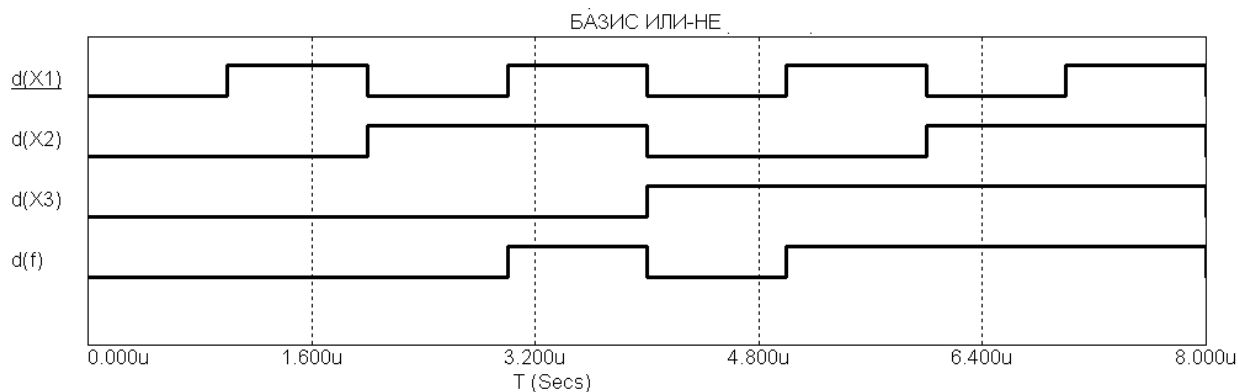


Рисунок 1.16 - Временная диаграмма работы схемы в базисе ИЛИ-НЕ

1.5 Лабораторное задание

Повторить методический пример, приведенный выше, по исходным данным Вашего варианта. **Критерием правильности** проведенного исследования является совпадение временных зависимостей выходного сигнала для всех схем. Временная зависимость выходного сигнала, в свою очередь, должна повторять значения логической функции в таблице истинности.

1.6 Контрольные вопросы

1. В чем заключаются этапы синтеза логического устройства?
2. Что такое дизъюнктивная нормальная форма?
3. Как происходит переход от дизъюнктивной нормальной формы к совершенной дизъюнктивной нормальной форме?
4. Каково правило записи совершенной дизъюнктивной нормальной формы для функции заданной таблично?
5. В чем недостаток структурных схем, построенных по СДНФ?
6. Что такое нулевой, единичный, двоичный куб?
7. Что такое ранг куба?
8. Что такое кубический комплекс?
9. Что такое покрытие логической функции?
10. Что такое цена покрытия логической функции?
11. Что такое карта Вейча?

1.7 Варианты заданий

Для всех вариантов задания общими являются три столбца таблицы истинности (таблица 1.4), которые представляют собой восемь возможных сочетаний аргументов логической функции $f(x_3, x_2, x_1)$. Варианты значений логической функции приведены в таблице 1.5. Выбрав соответствующий своему варианту столбец в таблице 1.5, необходимо присоединить его в качестве четвертого столбца к таблице 1.4 и получить, таким образом, полноценную таблицу истинности.

Таблица 1.4 – Сочетания аргументов логической функции

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Таблица 1.5 – Варианты значений логической функции $f(x_3, x_2, x_1)$

	НОМЕРА ВАРИАНТОВ												
	1	2	3	4	5	6	7	8	9	10	11	12	13
ЗНАЧЕНИЯ	1	1	1	1	1	1	1	1	0	1	0	0	0
	0	1	1	1	0	0	0	0	0	0	1	1	0
	1	0	1	1	1	1	1	1	1	0	1	0	1
	0	0	0	1	0	0	0	0	1	1	0	1	0
	1	1	0	0	1	0	1	0	1	1	1	1	0
	0	0	0	0	1	0	0	1	0	0	0	0	1
	1	1	1	0	0	1	0	1	1	1	1	1	1
	0	0	0	0	0	1	1	0	0	0	0	0	1

Окончание на следующей странице

Окончание таблицы 1.5

	НОМЕРА ВАРИАНТОВ											
	<i>14</i>	<i>15</i>	<i>16</i>	<i>17</i>	<i>18</i>	<i>19</i>	<i>20</i>	<i>21</i>	<i>22</i>	<i>23</i>	<i>24</i>	<i>25</i>
ЗНАЧЕНИЯ	0	1	0	0	0	0	1	1	0	1	1	1
	1	0	0	0	0	1	1	0	0	1	0	1
	0	0	0	1	0	0	0	1	1	1	0	1
	0	0	0	1	1	1	1	0	1	0	1	0
	0	0	1	1	1	0	0	1	1	1	0	0
	1	1	1	1	1	1	1	0	1	0	1	0
	1	1	1	0	1	0	0	1	1	0	1	1
	1	1	1	0	0	1	1	1	0	1	1	1

2 ЛАБОРАТОРНАЯ РАБОТА №2 – ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ

2.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

- 1) знакомство с процессом минимизации логических функций по методу Квайна;
- 2) изучение принципов доопределения логических функций с последующим нахождением минимальной формы;
- 3) изучение принципов минимизации систем логических функций на основе импликантных матриц;
- 4) приобретение практических навыков по использованию операций склеивания и поглощения.

2.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 2.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

2.3 Синтез недоопределенных логических функций. Синтез логических устройств с несколькими выходами

Для минимизации логических функций часто используются методы, обладающие однозначностью алгоритма, что является предпосылкой применения ЭВМ. К таким методам относится, в частности, метод Квайна. Метод Квайна позволяет представлять функции в дизъюнктивной нормальной форме с минимальным числом членов и минимальным числом букв в членах. Этот метод содержит два этапа преобразования выражения функции: на первом этапе осуществляется переход от канонической формы (СДНФ) к *сокращенной форме*, на втором этапе – переход от сокращенной формы логического выражения к *минимальной форме*.

Получение сокращенной формы. Пусть заданная функция f представлена в СДНФ. Переход к сокращенной форме основан на последовательном применении двух операций: *операции склеивания* и *операции поглощения*. Для выполнения операции склеивания в выражении функции выявляются пары членов вида $w \cdot x$ и $w \cdot \bar{x}$, различающихся лишь тем, что один из аргументов в одном из членов представлен без инверсии, а в другом – с инверсией. Затем проводится склеивание таких пар членов: $w \cdot x + w \cdot \bar{x} = w \cdot (x + \bar{x}) = w$, и результаты склеивания w вводятся в выражение функции в качестве дополнительных членов. Далее выполняется операция поглощения. Она основана на равенстве $w + w \cdot z = w \cdot (1 + z) = w$ (член w поглощает член $w \cdot z$). При проведении этой операции из логического выражения вычеркиваются все члены, поглощаемые членами, которые введены в результате операции склеивания. Операции склеивания и поглощения выполняются последовательно до тех пор, пока это возможно.

Члены сокращенной формы называются *простыми импликантами* функции.

Получение минимальной формы. Сокращенная форма может содержать лишние члены, исключение которых из выражения не повлияет на значение функции. Дальнейшее упрощение логического выражения достигается исключением из выражения лишних членов. В этом заключается содержание второго этапа минимизации. Покажем этот этап минимизации логического выражения на примере построения логического устройства для функции, заданной в таблице 2.1.

Таблица 2.1 – Таблица истинности логической функции

X_4	X_3	X_2	X_1	$F(X_4, X_3, X_2, X_1)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

СДНФ этой функции:

$$f(x_4, x_3, x_2, x_1) = \bar{x}_4 \cdot \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_4 \cdot x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_4 \cdot x_3 \cdot x_2 \cdot \bar{x}_1 + \\ + \bar{x}_4 \cdot x_3 \cdot x_2 \cdot x_1 + x_4 \cdot \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_4 \cdot x_3 \cdot x_2 \cdot x_1.$$

Для получения сокращенной формы проводим операции склеивания и поглощения, а также используем теорему булевой алгебры $a + F \cdot \bar{a} = a + F$:

$$f(x_4, x_3, x_2, x_1) = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 \cdot (\bar{x}_4 + x_4) + \bar{x}_4 \cdot x_3 \cdot x_2 \cdot (\bar{x}_1 + x_1) + \bar{x}_4 \cdot x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \\ + x_4 \cdot x_3 \cdot x_2 \cdot x_1 = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_4 \cdot x_3 \cdot x_2 + \bar{x}_4 \cdot x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_4 \cdot x_3 \cdot x_2 \cdot x_1 = \\ = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_4 \cdot x_3 \cdot (x_2 + \bar{x}_2 \cdot \bar{x}_1) + x_4 \cdot x_3 \cdot x_2 \cdot x_1 = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_4 \cdot x_3 \cdot x_2 + \\ + \bar{x}_4 \cdot x_3 \cdot \bar{x}_1 + x_4 \cdot x_3 \cdot x_2 \cdot x_1 = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot (\bar{x}_4 + x_4 \cdot x_1) + \bar{x}_4 \cdot x_3 \cdot \bar{x}_1 = \\ = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_4 \cdot x_3 \cdot x_2 + x_3 \cdot x_2 \cdot x_1 + \bar{x}_4 \cdot x_3 \cdot \bar{x}_1.$$

Выражение (2.1) представляет собой сокращенную форму логического выражения заданной функции, а члены его являются простыми импликантами функции. Переход от сокращенной формы к минимальной осуществляется с помощью импликантной матрицы, приведенной в таблице 2.2.

В столбцы импликантной матрицы вписываются члены СДНФ заданной функции, в строки – простые импликанты функции, т.е. члены сокращенной формы логического выражения функции. Отмечаются крестиками столбцы членов СДНФ, поглощаемых отдельными простыми импликантами.

Таблица 2.2 – Импликантная матрица

Простая импликанта	$\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	$\bar{x}_4 x_3 \bar{x}_2 \bar{x}_1$	$\bar{x}_4 x_3 x_2 \bar{x}_1$	$\bar{x}_4 x_3 x_2 x_1$	$x_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	$x_4 x_3 x_2 x_1$
$\bar{x}_3 \bar{x}_2 \bar{x}_1$	⊗				⊗	
$\bar{x}_4 x_3 x_2$			x	x		
$x_3 x_2 x_1$				⊗		⊗
$\bar{x}_4 x_3 \bar{x}_1$		⊗	⊗			

В приведенной матрице, например, простая импликанта $\bar{x}_4 x_3 \bar{x}_1$ поглощает члены $\bar{x}_4 x_3 \bar{x}_2 \bar{x}_1$ и $\bar{x}_4 x_3 x_2 \bar{x}_1$, поэтому во втором и третьем столбцах последней строки поставлены крестики. Импликанты, которые не могут быть лишними и, следовательно, не могут быть исключены из сокращенной формы, составляют *ядро*. Входящие в ядро импликанты легко определяются по импликантной матрице. Для каждой из них имеется хотя бы один столбец, перекрываемый только данной импликантой.

В рассматриваемом примере ядро составляют импликанты $\bar{x}_3 \bar{x}_2 \bar{x}_1$; $x_3 x_2 x_1$; $\bar{x}_4 x_3 \bar{x}_1$, потому что только ими перекрываются первый, второй, пятый и шестой столбцы матрицы. Ядро выделено в матрице скругленными прямоугольниками.

Для получения минимальной дизъюнктивной формы необходимо, в

первую очередь, включить простые импликанты, составляющие ядро функции. Затем, достаточно выбрать из импликант, не входящих в ядро, такое минимальное их количество с минимальным количеством букв в каждой из этих импликант, которое обеспечит перекрытие всех столбцов, не перекрытых членами ядра.

В рассматриваемом примере не требуется включать дополнительные импликанты, поскольку ядро функции перекрывает все столбцы. Следовательно, МДНФ заданной функции выглядит как:

$$f_{\text{МДНФ}}(x_4, x_3, x_2, x_1) = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1 + \bar{x}_4 \cdot x_3 \cdot \bar{x}_1.$$

Синтез недоопределенных логических функций. По условиям работы логического устройства некоторые наборы значений аргументов могут оказаться запрещенными для данного устройства и никогда не появиться на его входах. В этом случае функция задана не на всех наборах аргументов. Такие функции будем называть *недоопределенными*.

При синтезе логического устройства, реализующего не полностью заданную функцию, допустимо задаваться произвольными значениями функции на запрещенных наборах аргументов. При этом в зависимости от способа задания этих значений функции минимальная форма может оказаться простой или более сложной. Таким образом, возникает проблема целесообразного определения функции на запрещенных наборах аргументов.

Может быть использован следующий способ получения минимальной формы не полностью заданной функции f :

а) записывается СДНФ функции f_0 , полученной из функции f заданием значения 0 на всех запрещенных наборах аргументов;

б) записывается СДНФ функции f_1 , полученной из функции f заданием значения 1 на всех запрещенных наборах аргументов;

в) функция f_1 приводится к сокращенной форме (к форме, содержащей все простые импликанты);

г) составляется импликантная таблица из всех членов функции f_0 и простых импликант функции f_1 ;

д) искомая минимальная форма составляется из простых импликант функции f_1 , поглощающих все члены СДНФ функции f_0 .

Рассмотрим применение данного метода к минимизации недоопределенной функции, приведенной в таблице 2.3.

Методом Квайна приводим функцию f_1 к сокращенной форме:

$$\begin{aligned} f_1(x_3, x_2, x_1) &= \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \\ &+ x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1 = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_2 \cdot \bar{x}_1 \cdot (\bar{x}_3 + x_3) + \bar{x}_2 \cdot x_1 \cdot (\bar{x}_3 + x_3) + \\ &+ x_2 \cdot x_1 \cdot (\bar{x}_3 + x_3) = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_2 \cdot \bar{x}_1 + \bar{x}_2 \cdot x_1 + x_2 \cdot x_1 = \bar{x}_1 \cdot (\bar{x}_3 \cdot \bar{x}_2 + x_2) + \\ &+ x_1 \cdot (\bar{x}_2 + x_2) = \bar{x}_1 \cdot (\bar{x}_3 + x_2) + x_1 = \bar{x}_3 + x_2 + x_1. \end{aligned}$$

Составляем импликантную матрицу (таблица 2.4).

Таблица 2.3 – Таблица истинности недоопределенной логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	---
0	0	1	1
0	1	0	1
0	1	1	---
1	0	0	0
1	0	1	---
1	1	0	---
1	1	1	1

Таблица 2.4 – Импликантная матрица

ПРОСТЫЕ ИМПЛИКАНТЫ ФУНКЦИИ F_1	$\bar{x}_3\bar{x}_2x_1$	$\bar{x}_3x_2\bar{x}_1$	$x_3x_2x_1$
\bar{x}_3	×	×	
X_2		×	×
X_1	×		×

Минимальная форма логического выражения может быть получена тремя способами:

$$f(x_3, x_2, x_1) = \bar{x}_3 + x_2 \text{ или}$$

$$f(x_3, x_2, x_1) = x_2 + x_1 \text{ или}$$

$$f(x_3, x_2, x_1) = \bar{x}_3 + x_1.$$

Рассмотрим минимизацию той же функции методом карт Вейча (рисунок 2.1). При минимизации функции данным методом следует на запрещенных наборах аргументов задавать такие значения, при которых клетки со значением 1 охватываются минимальным числом областей с максимальным числом клеток в каждой области. Применительно к рассматриваемой функции такое доопределение может быть осуществлено тремя различными способами, представленными на рисунке 2.2. Они приводят к полученным выше выражениям МДНФ функции.

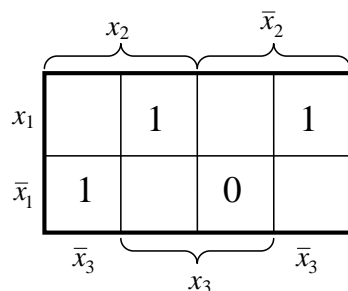


Рисунок 2.1 – Карта Вейча для недоопределенной логической функции

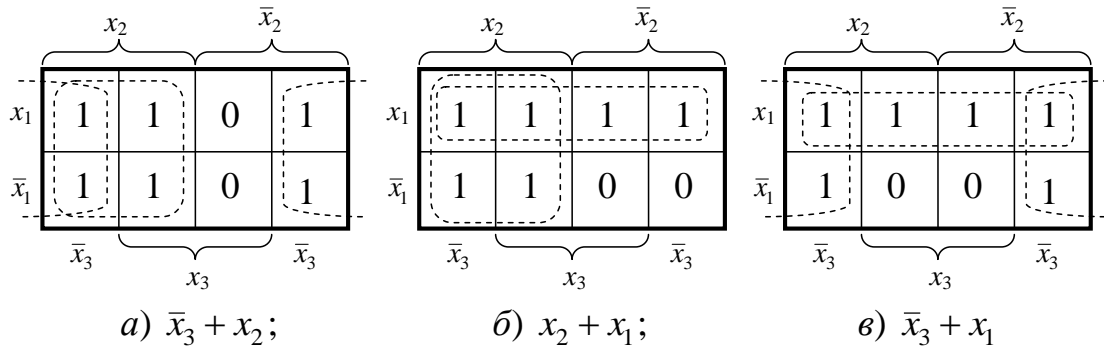


Рисунок 2.2 – Различные способы доопределения логической функции

Синтез логических устройств с несколькими выходами. Пусть синтезируемое логическое устройство имеет n входов и m выходов (рисунок 2.3). На каждом из выходов должна быть сформирована определенная функция входных переменных. Эта задача могла бы быть решена синтезированием отдельно действующих узлов, каждый из которых реализовывал бы определенную выходную функцию. Однако, если даже каждый из этих узлов будет построен минимальным образом, в целом логическое устройство может оказаться не минимальным. Действительно, такое устройство могло бы быть минимизировано путем использования общих элементов в нескольких узлах, реализующих различные выходные функции.

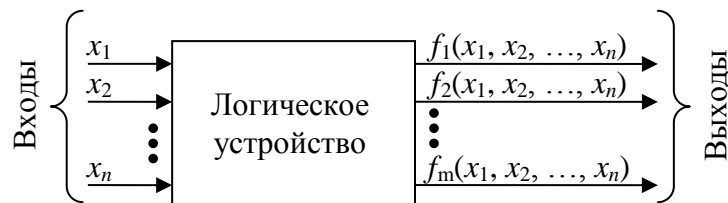


Рисунок 2.3 – Логическое устройство с несколькими выходами

Из этих соображений приведение каждой из выходных функций к минимальной форме не является условием получения минимального в целом устройства. При минимизации устройства в целом некоторые из функций могут оказаться представленными в неминимальной форме.

Принцип получения минимальной формы устройства сводится к нахождению минимального набора членов с минимальным числом входящих в них букв, достаточного для получения всех формируемых устройством функций.

Метод построения минимальных логических устройств с несколькими выходами рассмотрим на примере реализации устройства, способ функционирования которого задан в таблице 2.5.

Записываем в первый столбец таблицы 2.6 наборы аргументов, на которых хотя бы одна из выходных функций имеет значение 1. Во второй столбец таблицы 2.6 в качестве признака записываем функции, принимающие значение 1 при данном наборе аргументов.

Таблица 2.5 – Таблица истинности логического устройства с тремя выходами

X_3	X_2	X_1	$F_1(X_3, X_2, X_1)$	$F_2(X_3, X_2, X_1)$	$F_3(X_3, X_2, X_1)$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Таблица 2.6 – Признаки наборов аргументов

Наборы аргументов	Признаки наличия единицы	Признак поглощения таблицей 2.7
$\bar{x}_3 \bar{x}_2 x_1$	$f_2 f_3$	✓
$\bar{x}_3 x_2 \bar{x}_1$	$f_2 f_3$	
$\bar{x}_3 x_2 x_1$	f_1	✓
$x_3 \bar{x}_2 \bar{x}_1$	$f_1 f_3$	✓
$x_3 \bar{x}_2 x_1$	$f_2 f_3$	✓
$x_3 x_2 \bar{x}_1$	$f_1 f_3$	✓
$x_3 x_2 x_1$	f_1	✓

Далее проводится операция склеивания, и получающиеся при этом члены заносятся в таблицу 2.7, рядом с членами записываются признаки в виде функций, общих в признаках той пары членов таблицы 2.6, склеиванием которых они получены.

Таблица 2.7 – Признаки наборов аргументов после склеивания

ИМПЛИКАНТЫ	ПРИЗНАКИ
$\bar{x}_2 \cdot x_1$	$F_2 F_3$
$x_2 \cdot \bar{x}_1$	F_3
$x_2 \cdot x_1$	F_1
$x_3 \cdot \bar{x}_2$	F_3
$x_3 \cdot \bar{x}_1$	$F_1 F_3$
$x_3 \cdot x_2$	F_1

Так, склеивание членов таблицы 2.6 $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ ($f_1 f_3$) и $x_3 \cdot x_2 \cdot \bar{x}_1$ ($f_1 f_3$) приводит в таблице 2.7 к члену $x_3 \cdot \bar{x}_1$ ($f_1 f_3$); склеивание членов $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ ($f_1 f_3$) и $x_3 \cdot \bar{x}_2 \cdot x_1$ ($f_2 f_3$) приводит к члену $x_3 \cdot \bar{x}_2$ (f_3) и т.д. Операция склеивания не проводится над членами, в признаках которых не имеется общих функций.

Далее реализуется операция поглощения членами таблицы 2.7 членов таблицы 2.6. Операция поглощения может проводиться лишь над членами, имеющими одинаковую комбинацию функций в признаках. Указанные операции склеивания и поглощения повторяются до тех пор, пока это возможно. Так, в таблице 2.6 галочками отмечены члены, поглотившиеся в результате членами таблицы 2.7.

Затем составляется импликантная матрица (таблица 2.8). Поскольку один из членов таблицы 2.6 не поглотился, его записывают в таблицу 2.8 наряду с простыми импликантами. Определяется набор импликант, обеспечивающий перекрытие всех столбцов импликантной матрицы (таблица 2.9).

Таблица 2.8 – Импликантная матрица

Импликанты	$\bar{x}_3 \bar{x}_2 x_1$		$\bar{x}_3 x_2 \bar{x}_1$		$\bar{x}_3 x_2 x_1$		$x_3 \bar{x}_2 \bar{x}_1$		$x_3 \bar{x}_2 x_1$		$x_3 x_2 \bar{x}_1$		$x_3 x_2 x_1$	
	f_2	f_3	f_2	f_3	f_1		f_1	f_3	f_2	f_3	f_1	f_3	f_1	
$\bar{x}_2 x_1$ ($f_2 f_3$)	×	×							×	×				
$x_2 \bar{x}_1$ (f_3)				×								×		
$x_2 x_1$ (f_1)					×									×
$x_3 \bar{x}_2$ (f_3)								×		×				
$x_3 \bar{x}_1$ ($f_1 f_3$)							×	×			×	×		
$x_3 x_2$ (f_1)											×			×
$\bar{x}_3 x_2 \bar{x}_1$ ($f_2 f_3$)			×	×										

Таблица 2.9 – Набор импликант, обеспечивающих перекрытие

ИМПЛИКАНТЫ	ПРИЗНАКИ
$\bar{x}_2 \cdot x_1$	$F_2 F_3$
$\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$	$F_2 F_3$
$x_2 \cdot x_1$	F_1
$x_3 \cdot \bar{x}_1$	$F_1 F_3$

Записываем для выходных функций логические выражения, составленные из этих импликант, в признаках которых содержатся заданные функции:

$$f_1(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_1;$$

$$f_2(x_3, x_2, x_1) = \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1;$$

$$f_3(x_3, x_2, x_1) = \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1.$$

Легко убедиться, что выражение для функции $f_3(x_3, x_2, x_1)$ не является минимальным. Минимальная для этой функции форма:

$$\begin{aligned} f_3(x_3, x_2, x_1) &= \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 = \\ &= \bar{x}_1 \cdot (\bar{x}_3 \cdot x_2 + x_3) + \bar{x}_2 \cdot x_1 = x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 + \bar{x}_2 \cdot x_1 \end{aligned}$$

невыгодна для всей системы из-за члена $x_2 \cdot \bar{x}_1$, так как он не присутствует в функциях $f_1(x_3, x_2, x_1)$ и $f_2(x_3, x_2, x_1)$.

На рисунке 2.4 приведена структурная схема логического устройства, обеспечивающего заданное таблицей 2.5 функционирование. Как видно из схемы, ряд элементов участвует в формировании нескольких функций.

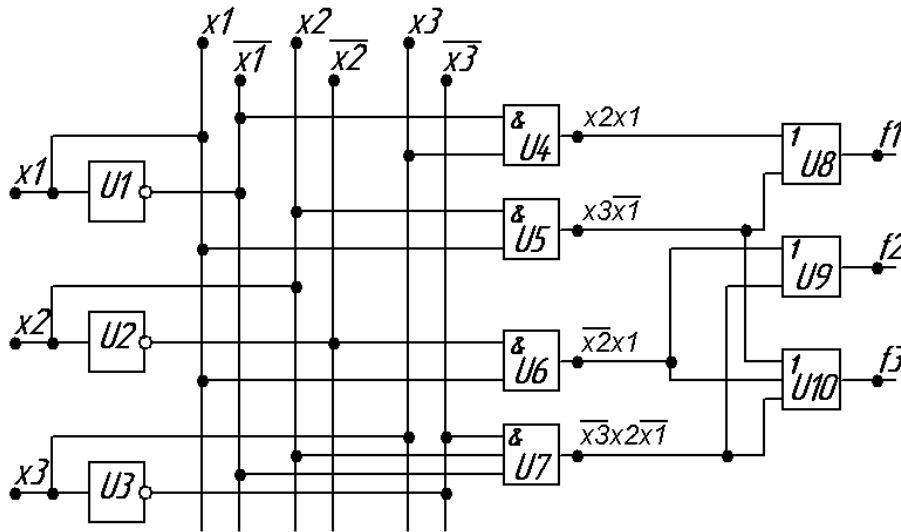


Рисунок 2.4 – Структурная схема логического устройства с тремя выходами

2.4 Пример синтеза логических устройств

Недоопределенная логическая функция трех аргументов $f(x_3, x_2, x_1)$ и система логических функций трех аргументов

$$\begin{cases} g_1(x_3, x_2, x_1); \\ g_2(x_3, x_2, x_1); \\ g_3(x_3, x_2, x_1), \end{cases}$$

заданы в виде таблиц истинности (таблицы 2.10, 2.11).

Для логической функции $f(x_3, x_2, x_1)$ требуется:

- провести доопределение на запрещенных наборах аргументов с целью получения минимально возможной формы;
- построить структурную схему логического устройства в произвольном базисе по найденной минимальной форме.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Для системы логических функций

$$\begin{cases} g_1(x_3, x_2, x_1); \\ g_2(x_3, x_2, x_1); \\ g_3(x_3, x_2, x_1), \end{cases}$$

требуется:

- провести минимизацию аппаратных ресурсов устройства, которое будет реализовывать заданную систему;
- построить структурную схему логического устройства в произвольном базисе по найденной минимальной форме.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Таблица 2.10 – Таблица истинности недоопределенной логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	---
0	0	1	0
0	1	0	1
0	1	1	---
1	0	0	1
1	0	1	---
1	1	0	---
1	1	1	1

Таблица 2.11 – Таблица истинности системы логических функций

X_3	X_2	X_1	$G_1(X_3, X_2, X_1)$	$G_2(X_3, X_2, X_1)$	$G_3(X_3, X_2, X_1)$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	1	0

I этап. Доопределение логической функции на запрещенных наборах аргументов.

Пусть логическая функция $f_0(x_3, x_2, x_1)$ описывает процесс доопределения исходной таблицы истинности (таблица 2.10) значениями 0, тогда логическая функция $f_1(x_3, x_2, x_1)$ – процесс доопределения значениями 1.

Логическое выражение функции $f_0(x_3, x_2, x_1)$ в СДНФ выглядит как:

$$f_0(x_3, x_2, x_1) = \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1.$$

Аналогичное выражение функции $f_1(x_3, x_2, x_1)$ записывается как:

$$f_1(x_3, x_2, x_1) = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \\ + x_3 \cdot \bar{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1.$$

Логическая функция $f_1(x_3, x_2, x_1)$, представленная в СДНФ, преобразуется к сокращенной форме по методу Квайна. Напомним, что для преобразования к сокращенной форме используются три теоремы булевой алгебры:

1. $w \cdot z + w \cdot \bar{z} = w \cdot (z + \bar{z}) = w$ – теорема склеивания;
2. $w + w \cdot z = w \cdot (1 + z) = w$ – теорема поглощения;
3. $a + F \cdot \bar{a} = a + F$.

Процесс преобразования к сокращенной форме выглядит следующим образом:

$$f_1(x_3, x_2, x_1) = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \\ + x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1 = \bar{x}_2 \cdot \bar{x}_1 \cdot (\bar{x}_3 + x_3) + \bar{x}_3 \cdot x_2 \cdot (\bar{x}_1 + x_1) + x_3 \cdot x_2 \cdot (\bar{x}_1 + x_1) + \\ + x_3 \cdot \bar{x}_2 \cdot x_1 = \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot \bar{x}_2 \cdot x_1 = \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot (x_2 + \bar{x}_2 \cdot x_1) = \\ = \bar{x}_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 = x_2 \cdot (\bar{x}_3 + x_3) + \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_1 = \\ = x_2 + \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_1 = x_2 + \bar{x}_1 + x_3 \cdot x_1 = x_2 + \bar{x}_1 + x_3.$$

Составляется импликантная матрица (таблица 2.12) функции $f_0(x_3, x_2, x_1)$ и простых импликант $f_1(x_3, x_2, x_1)$. Напомним, что крестиками отмечаются те столбцы членов СДНФ, которые поглощаются отдельными простыми импликантами.

Таблица 2.12 – Импликантная матрица

ПРОСТЫЕ ИМПЛИКАНТЫ ФУНКЦИИ F_1	$\bar{x}_3 x_2 \bar{x}_1$	$x_3 \bar{x}_2 \bar{x}_1$	$x_3 x_2 x_1$
X_3		×	×
X_2	×		×
\bar{x}_1	×	×	

Минимальная форма логического выражения функции может быть получена тремя способами:

$$f(x_3, x_2, x_1) = x_3 + \bar{x}_1 \text{ или} \\ f(x_3, x_2, x_1) = x_2 + \bar{x}_1 \text{ или} \\ f(x_3, x_2, x_1) = x_3 + x_2. \quad (2.2)$$

II этап. Построение структурной схемы логического устройства с одним выходом.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP...».

По условию лабораторного задания структурная схема логического устройства может быть реализована в произвольном базисе. Среди вариантов минимальных форм в (2.2) наиболее выгодным с точки зрения аппаратной реализации является последнее выражение. Аппаратная реализация логической функции $f(x_3, x_2, x_1) = x_3 + x_2$ потребует применения лишь одного элемента 2ИЛИ. Нетрудно, убедиться, что перевод любого из выражений (2.2) в базисы И-НЕ, ИЛИ-НЕ для данного случая невыгоден – вновь полученные выражения будут более громоздкими.

Структурная схема логического устройства на основе минимально возможной формы $f(x_3, x_2, x_1) = x_3 + x_2$, подготовленная в программе MicroCAP, представлена на рисунке 2.5.

Особенности установки параметров для источника цифрового сигнала и прочие аспекты, связанные с моделированием схемы во временной области, подробно рассмотрены в методическом примере лабораторной работы №1.

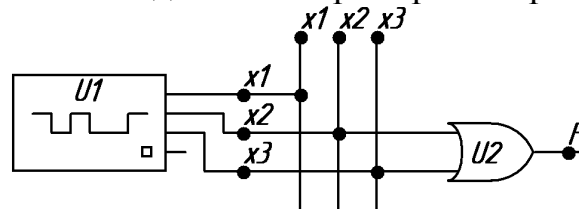


Рисунок 2.5 – Структурная схема логического устройства по минимально возможной форме

Нижняя зависимость $d(f)$ на временной диаграмме (рисунок 2.6) представляет собой значения логической функции $f(x_3, x_2, x_1)$. Сравнивая значения функции на 2, 3, 5 и 8-ом временных интервалах с соответствующими строками таблицы истинности (таблица 2.10), приходим к выводу об адекватности проведенного моделирования. На остальных временных интервалах значения функции являются факультативными (необязательными). Эти временные интервалы соответствуют запрещенным наборам аргументов таблицы истинности.

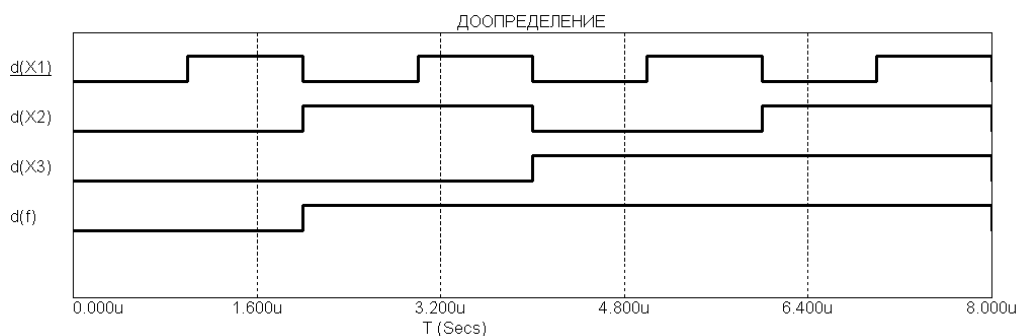


Рисунок 2.6 – Временная диаграмма работы схемы в минимально возможной форме

III этап. Минимизация системы логических функций.

Система логических функций трех аргументов

$$\begin{cases} g_1(x_3, x_2, x_1); \\ g_2(x_3, x_2, x_1); \\ g_3(x_3, x_2, x_1), \end{cases}$$

задана в виде таблицы истинности (таблица 2.11). В первый столбец таблицы 2.13 записываются наборы аргументов, на которых хотя бы одна из выходных функций имеет значение 1. Во второй столбец в качестве признака записываются функции, принимающие значение 1 при данном наборе аргументов.

Таблица 2.13 – Признаки наборов аргументов

Наборы аргументов	Признаки наличия единицы	Признак поглощения таблицей 2.14
$\bar{x}_3 \bar{x}_2 \bar{x}_1$	$g_1 g_2 g_3$	√
$\bar{x}_3 \bar{x}_2 x_1$	g_3	√
$x_3 \bar{x}_2 \bar{x}_1$	$g_1 g_2 g_3$	√
$x_3 \bar{x}_2 x_1$	g_1	√
$x_3 x_2 \bar{x}_1$	$g_1 g_2 g_3$	√
$x_3 x_2 x_1$	g_2	√

Проведем операцию склеивания над набором аргументов таблицы 2.13. Напомним, что операция склеивания возможна только над наборами аргументов, в признаках которых имеются общие функции. В операции склеивания участвуют следующие пары:

1. $\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$ и $\bar{x}_3 \cdot x_2 \cdot x_1$;
2. $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ и $x_3 \cdot \bar{x}_2 \cdot x_1$;
3. $x_3 \cdot x_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot x_1$;
4. $\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$;
5. $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$.

Результаты операций склеивания заносятся в первый столбец таблицы 2.14, во втором столбце – общие признаки склеенной пары.

Таблица 2.14 – Результаты операций склеивания

ИМПЛИКАНТЫ	ПРИЗНАКИ
$\bar{x}_3 \cdot x_2$	G_3
$x_3 \cdot \bar{x}_2$	G_1
$x_3 \cdot x_2$	G_2
$x_2 \cdot \bar{x}_1$	$G_1 G_2 G_3$
$x_3 \cdot \bar{x}_1$	$G_1 G_2 G_3$

Проведем операцию поглощения членами таблицы 2.14 членов таблицы 2.13. Операция поглощения возможна только над членами, имеющими *одинаковую* комбинацию функций в признаках. Следовательно, в операции поглощения участвуют пары:

1. $\bar{x}_3 \cdot x_2$ и $\bar{x}_3 \cdot x_2 \cdot x_1$;
2. $x_3 \cdot \bar{x}_2$ и $x_3 \cdot \bar{x}_2 \cdot x_1$;
3. $x_3 \cdot x_2$ и $x_3 \cdot x_2 \cdot x_1$;
4. $x_2 \cdot \bar{x}_1$ и $\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$;
5. $x_3 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$;
6. $x_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$ или $x_3 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$.

По результатам проведения операции следует, что все члены таблицы 2.13 поглотились. По этой причине в импликантной матрице (таблица 2.15) в первом столбце перечислены только члены из таблицы 2.14. В противном случае, следовало бы дописать непоглощенные члены таблицы 2.13 в первый столбец таблицы 2.15 (см. пример раздела 2.3). В остальных столбцах матрицы записаны члены таблицы 2.13 до их поглощения.

Таблица 2.15 – Импликантная матрица

Импликанта	$\bar{x}_3 x_2 \bar{x}_1$			$\bar{x}_3 x_2 x_1$	$x_3 \bar{x}_2 \bar{x}_1$			$x_3 \bar{x}_2 x_1$	$x_3 x_2 \bar{x}_1$			$x_3 x_2 x_1$
	g_1	g_2	g_3	g_3	g_1	g_2	g_3	g_1	g_1	g_2	g_3	g_2
$\bar{x}_3 x_2 (g_3)$			x	x								
$x_3 \bar{x}_2 (g_1)$					x			x				
$x_3 x_2 (g_2)$										x		x
$x_2 \bar{x}_1 (g_1 g_2 g_3)$	x	x	x						x	x	x	
$x_3 \bar{x}_1 (g_1 g_2 g_3)$					x	x	x		x	x	x	

В импликантной матрице крестиками отмечены столбцы членов, поглощаемых отдельными импликантами с учетом общих функциональных признаков.

Определим набор импликант, обеспечивающий перекрытие всех столбцов матрицы (таблица 2.16).

Таблица 2.16 – Набор импликант, обеспечивающий перекрытие

ИМПЛИКАНТЫ	ПРИЗНАКИ
$x_2 \cdot \bar{x}_1$	$G_1 G_2 G_3$
$\bar{x}_3 \cdot x_2$	G_3
$x_3 \cdot \bar{x}_1$	$G_1 G_2 G_3$
$x_3 \cdot \bar{x}_2$	G_1
$x_3 \cdot x_2$	G_2

Заметим, что в данном примере перекрытие столбца $x_3 \cdot x_2 \cdot \bar{x}_1$ возможно двумя равноценными способами: импликантой $x_2 \cdot \bar{x}_1$ (как это сделано в таблице 2.15) или импликантой $x_3 \cdot \bar{x}_1$.

Запишем для выходных функций логические выражения, составленные из импликант таблицы 2.16, в признаках которых содержатся заданные функции:

$$\begin{aligned} g_1(x_3, x_2, x_1) &= x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2; \\ g_2(x_3, x_2, x_1) &= x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 + x_3 \cdot x_2; \\ g_3(x_3, x_2, x_1) &= x_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_1. \end{aligned} \quad (2.3)$$

Анализ полученных выражений позволяет сделать вывод: импликанты $x_2 \cdot \bar{x}_1$ и $x_3 \cdot \bar{x}_1$ присутствуют одновременно во всех логических функциях. Это позволяет значительно сократить (минимизировать) общее количество логических элементов при синтезе устройства.

IV этап. Построение структурной схемы логического устройства с тремя выходами.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP...».

По условию лабораторного задания структурная схема логического устройства может быть реализована в произвольном базисе. Аппаратная реализация системы логических функций (2.3) потребует 11 логических элементов. В частности, три элемента НЕ нужны для выработки инверсных сигналов \bar{x}_3 , \bar{x}_2 , \bar{x}_1 ; пять элементов 2И требуются для соответствующих операций логического умножения; три элемента 3ИЛИ будут формировать окончательный вид логических функций – дизъюнкцию конъюнктивных членов.

Структурная схема логического устройства на основе системы (2.3), подготовленная в программе MicroCAP, представлена на рисунке 2.7. Все технические подробности, связанные с изображением схемы в MicroCAP и с последующим моделированием во временной области, были рассмотрены ранее.

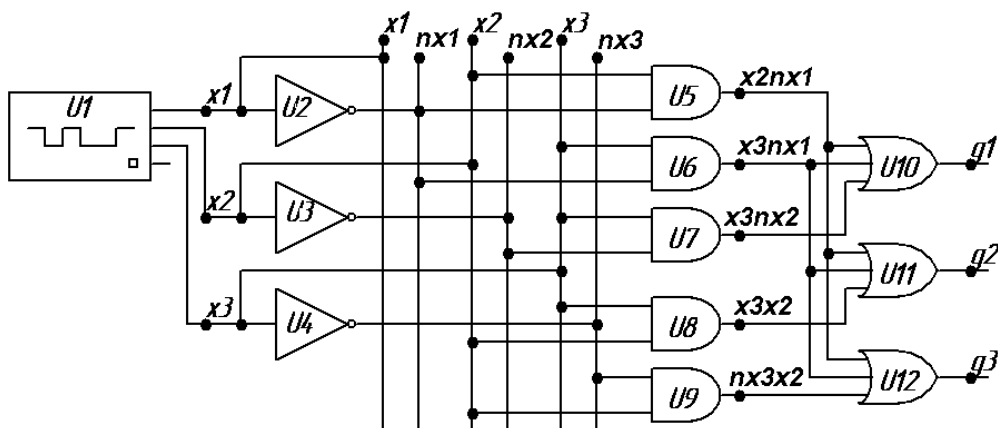


Рисунок 2.7 – Структурная схема логического устройства с тремя выходами

Временная диаграмма работы устройства с тремя выходами показана на рисунке 2.8. На графике для удобства восприятия и анализа, помимо одиночных сигналов, присутствуют временные зависимости входной $x_3x_2x_1$ и выходной $g_3g_2g_1$ шин данных.

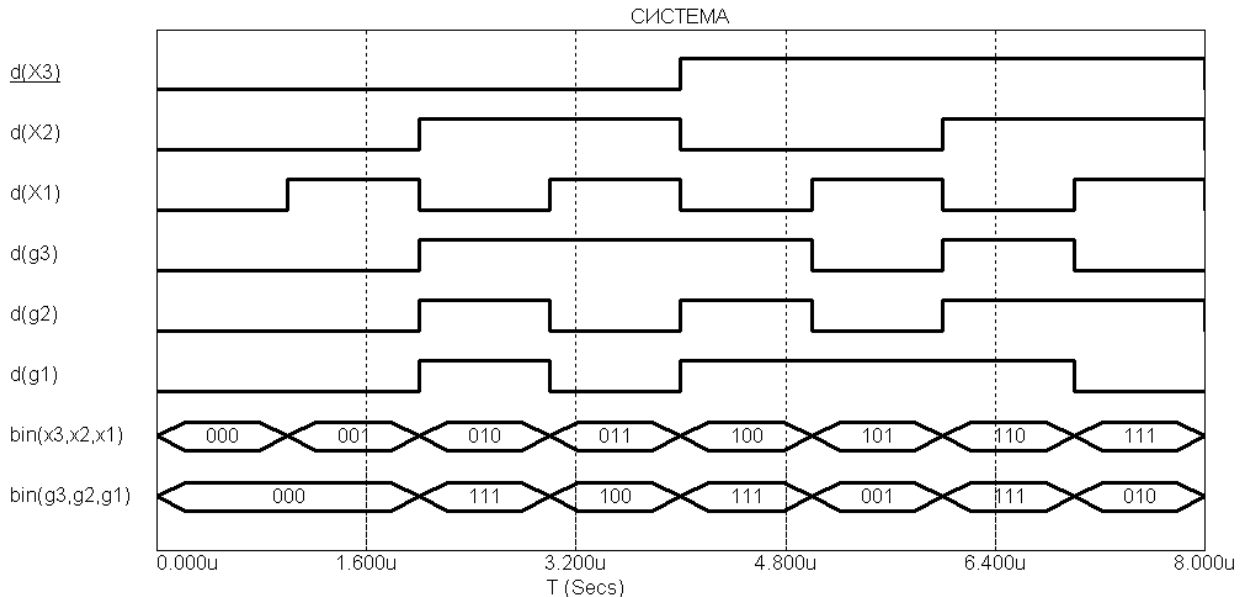


Рисунок 2.8 – Временная диаграмма работы логического устройства с тремя выходами

2.5 Лабораторное задание

Повторить методический пример, приведенный выше, по исходным данным Вашего варианта. **Критерием правильности** проведенного исследования является совпадение значений выходного сигнала на различных временных интервалах в MicroCAP с соответствующими строками таблицы истинности.

2.6 Контрольные вопросы

1. В чем заключаются основные этапы минимизации логических функций по методу Квайна?
2. Что такое операция склеивания?
3. Что такое операция поглощения?
4. Чем отличается сокращенная форма представления логической функции от минимальной?
5. Что такое простая импликанта?
6. Каков принцип заполнения импликантной матрицы?
7. Что такое ядро сокращенной формы представления логической функции?
8. Что нужно сделать, чтобы на основе заполненной импликантной матрицы получить минимальную форму логической функции?
9. Что такое недоопределенная логическая функция?
10. В чем особенность минимизации системы логических функций?

2.7 Варианты заданий

2.7.1 Варианты заданий для минимизации недоопределенных логических функций

Для всех вариантов задания общими являются три столбца таблицы истинности (таблица 2.17), которые представляют собой восемь возможных сочетаний аргументов логической функции $f(x_3, x_2, x_1)$. Варианты значений логической функции приведены в таблице 2.18. Выбрав соответствующий своему варианту столбец в таблице 2.18, необходимо присоединить его в качестве четвертого столбца к таблице 2.17 и получить, таким образом, полноценную таблицу истинности.

Таблица 2.17 – Сочетания аргументов логической функции

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Таблица 2.18 – Варианты значений логической функции $f(x_3, x_2, x_1)$

	НОМЕРА ВАРИАНТОВ												
	1	2	3	4	5	6	7	8	9	10	11	12	13
ЗНАЧЕНИЯ	–	–	1	–	1	–	1	1	–	–	0	0	0
	–	1	–	1	–	0	0	–	–	0	1	–	–
	–	–	–	1	–	–	1	1	–	–	1	–	1
	–	0	–	1	0	–	0	–	1	1	–	1	–
	1	–	–	0	1	–	–	0	–	–	–	–	–
	0	0	0	–	1	0	–	–	0	0	–	0	1
	1	–	1	–	–	1	–	–	1	1	1	1	1
	0	0	0	–	–	1	–	0	0	–	–	–	–

Окончание таблицы 2.18

	НОМЕРА ВАРИАНТОВ											
	14	15	16	17	18	19	20	21	22	23	24	25
ЗНАЧЕНИЯ	0	1	–	0	–	0	1	1	1	–	0	–
	–	0	0	0	–	–	0	1	1	0	–	0
	0	–	–	–	0	–	1	0	1	–	–	–
	0	0	–	–	1	1	–	–	–	–	–	–
	–	–	1	1	–	–	–	–	–	–	–	–
	–	1	1	–	1	–	–	–	–	1	0	1
	–	–	–	0	–	0	0	0	0	0	1	1
	1	–	1	–	0	1	–	–	–	1	1	1

2.7.2 Варианты заданий для минимизации системы логических функций

Для всех вариантов задания общими являются три столбца таблицы истинности (таблица 2.19), которые представляют собой восемь возможных сочетаний аргументов для логических функций $g_1(x_3, x_2, x_1)$, $g_2(x_3, x_2, x_1)$, $g_3(x_3, x_2, x_1)$. Варианты значений логических функций приведены в таблице 2.20. Выбрав соответствующие своему варианту три столбца в таблице 2.20, необходимо присоединить их к таблице 2.19 и получить, таким образом, полную таблицу истинности.

Таблица 2.19 – Сочетания аргументов логических функций

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Таблица 2.20 – Варианты значений логических функций $g_1(x_3, x_2, x_1)$, $g_2(x_3, x_2, x_1)$, $g_3(x_3, x_2, x_1)$

	НОМЕРА ВАРИАНТОВ																		
	1			2			3			4			5			6			
ФУНКЦИИ	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	
ЗНАЧЕНИЯ ФУНКЦИЙ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	
	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	
	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	
	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1
	1	1	1	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	
	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	

Продолжение таблицы 2.20

	НОМЕРА ВАРИАНТОВ																	
	7			8			9			10			11			12		
ФУНКЦИИ	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
ЗНАЧЕНИЯ ФУНКЦИЙ	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	1
	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	0	0	1	0	1	1	1	1	0	1	0	1	0	1	0	1	0	0
	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Продолжение на следующей странице

Продолжение таблицы 2.20

ФУНКЦИИ	НОМЕРА ВАРИАНТОВ																		
	13			14			15			16			17			18			
	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	
ЗНАЧЕНИЯ ФУНКЦИЙ	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	1	
	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	
	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	0	
	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1	1	1	
	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	

Продолжение таблицы 2.20

ФУНКЦИИ	НОМЕРА ВАРИАНТОВ																	
	19			20			21			22			23			24		
	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
ЗНАЧЕНИЯ ФУНКЦИЙ	0	1	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0
	1	1	1	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1
	1	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0
	0	0	0	1	0	1	0	1	1	1	1	0	0	1	1	1	0	1
	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0
	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	0	1	1
	0	0	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0
	1	1	0	0	1	1	1	1	0	1	1	0	1	1	1	1	0	1

Окончание таблицы 2.20

ФУНКЦИИ	НОМЕР ВАРИАНТА		
	25		
	G_1	G_2	G_3
ЗНАЧЕНИЯ ФУНКЦИЙ	0	1	0
	1	0	0
	0	1	1
	1	0	0
	0	0	1
	1	1	1
	0	1	0
	1	1	0

3 ЛАБОРАТОРНАЯ РАБОТА №3 – УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ

3.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

- 1) знакомство с принципом действия мультиплексора, его режимами работы, назначением выводов;
- 2) изучение некоторых способов настройки универсальных логических модулей на основе мультиплексоров;
- 3) приобретение навыков разложения логических выражений по Шеннону для случая двух переменных;
- 4) отработка приемов по составлению остаточных функций для универсальных логических модулей.

3.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 3.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

3.3 Способы настройки универсальных логических модулей

Универсальные логические модули (УЛМ) на основе мультиплексоров относятся к устройствам, настраиваемым на решение той или иной задачи. Универсальность их состоит в том, что для заданного числа аргументов можно настроить УЛМ на любую функцию. Известно, что общее число функций n аргументов выражается как 2^{2^n} . С ростом n число функций растет чрезвычайно быстро. Хотя практический интерес представляет не все существующие функции, возможность получить любую из огромного числа функций свидетельствует о больших перспективах применения УЛМ.

Первый способ настройки УЛМ. Первым способом настройки, используемым в УЛМ, является фиксация некоторых входов. Для этого способа

справедливо следующее соотношение между числом аргументов и числом настроечных входов. Пусть число аргументов n и требуется настройка на любую из функций. Тогда число комбинаций для кода настройки, равное числу функций, есть 2^{2^n} . Для двоичного кода число комбинаций связано с разрядностью кода выражением 2^m , где m – разрядность кода. Приравнявая число воспроизводимых функций к числу комбинаций кода настройки, имеем для числа настроечных входов соотношение $m = 2^n$.

Полученному выражению отвечает соотношение между числом входов разного типа для мультиплексора. При этом на адресные входы следует подавать аргументы функции, а на информационные входы – сигналы настройки (рисунок 3.1). Таким образом, для использования мультиплексора в качестве УЛМ следует изменить назначение его входов.

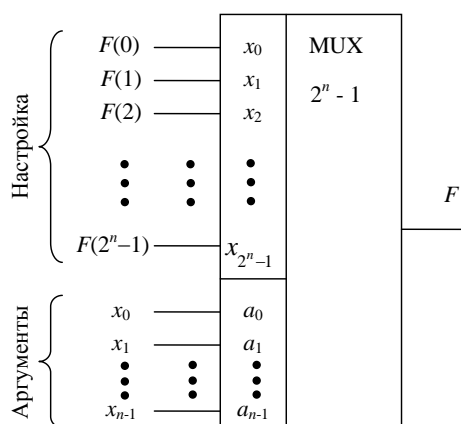


Рисунок 3.1 – Схема использования мультиплексора в качестве УЛМ

Рисунок 3.1 иллюстрирует возможность воспроизведения с помощью мультиплексора любой функции n аргументов. Действительно, каждому набору аргументов соответствует передача на выход одного из сигналов настройки. Если этот сигнал есть значение функции на данном наборе аргументов, то задача решена. Разным функциям будут соответствовать разные коды настройки. Алфавитом настройки будет $\{0, 1\}$ – настройка осуществляется константами 0 и 1. На рисунке 3.2 показан пример воспроизведения функции неравнозначности $x_1 \oplus x_2$ с помощью мультиплексора «4 – 1».

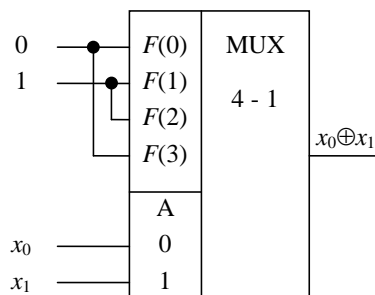


Рисунок 3.2 – Воспроизведение функции при настройке константами

Большое число настроечных входов затрудняет реализацию УЛМ. Для УЛМ, расположенных внутри кристалла, можно вводить код настройки последовательно в сдвигающий регистр, к разрядам которого подключены входы настройки. Тогда внешним входом настройки будет всего один, но настройка будет занимать не один такт, а 2^n тактов. Возможны и промежуточные последовательно-параллельные варианты ввода кода настройки.

Второй способ настройки УЛМ. Большое число входов настройки наталкивает на поиск возможностей их уменьшения. Такие возможности существуют и заключаются в расширении алфавита настроечных сигналов. Если от алфавита $\{0, 1\}$ перейти к алфавиту $\{0, 1, \tilde{x}_i\}$, где \tilde{x}_i - литерал одного из аргументов, то число входов аргументов сократится на единицу, а число настроечных входов – вдвое. Напомним, что под литералом переменной понимается либо сама переменная, либо ее инверсия. Перенос одного из аргументов в число сигналов настройки не влечет за собою каких-либо схемных изменений. На том же оборудовании будут реализованы функции с числом аргументов на единицу больше, чем при настройке константами.

Для нового алфавита код настройки находится следующим образом. Аргументы, за исключением \tilde{x}_i , подаются на адресующие входы, что соответствует их фиксации в выражении для искомой функции, которая становится функцией единственного аргумента \tilde{x}_i . Эту функцию, которую назовем остаточной, и нужно подавать на настроечные входы.

Если искомая функция зависит от n аргументов и в число сигналов настройки будет перенесен один из аргументов, то возникает n вариантов решения задачи, т.к. в сигналы настройки может быть перенесен любой аргумент. Спрашивается, какой именно аргумент целесообразно переносить в сигналы настройки? *Здесь можно опираться на рекомендацию: в настроечные сигналы следует переводить аргумент, который имеет минимальное число вхождений в импликанты функции.* В этом случае будут максимально использованы как бы внутренние логические ресурсы мультиплексора, а среди сигналов настройки увеличится число констант, что и считается благоприятным для схемной реализации УЛМ.

Проиллюстрируем сказанное примером воспроизведением функции трех аргументов $F = x_3x_2x_1 + \bar{x}_3\bar{x}_2$. Минимальное число вхождений в выражение функции имеет переменная x_1 , которую и перенесем в число сигналов настройки. Остаточная функция определится таблицей 3.1.

Например, при $x_3 = 0$ и $x_2 = 0$ имеем $F_{OCT} = 0 \cdot 0 \cdot x_1 + \bar{0} \cdot \bar{0} = 1$. При $x_3 = 0$ и $x_2 = 1$ имеем $F_{OCT} = 0 \cdot 1 \cdot x_1 + \bar{0} \cdot \bar{1} = 0$ и т.д.

Схема УЛМ приведена на рисунке 3.3.

По пути расширения алфавита сигналов настройки можно идти и дальше, но при этом понадобятся дополнительные логические схемы, воспроизводящие остаточные функции, которые будут уже зависеть более чем от одного аргумента.

Таблица 3.1 – Таблица истинности остаточной функции

X_3	X_2	$F_{\text{ост}}$
0	0	1
0	1	0
1	0	0
1	1	X_1

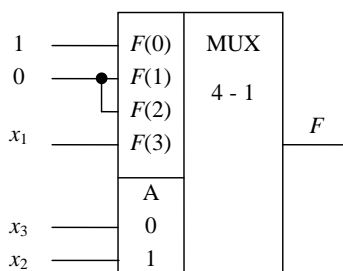


Рисунок 3.3 – Перенос одного аргумента в число сигналов настройки

Если в сигналы настройки перевести два аргумента, то дополнительные логические схемы будут двухвходовыми вентилями, что мало усложняет УЛМ и может оказаться приемлемым решением. В этом случае для сохранения универсальности УЛМ мультиплексору нужно предпослать блок выработки остаточных функций, в котором формируются все функции двух переменных (за исключением констант 0, 1 и литералов самих переменных, которые не требуется вырабатывать). Такой блок показан на рисунке 3.4. Пример реализации функции $F = x_2x_1 + \bar{x}_4x_3$ при алфавите настройки $\{0, 1, \tilde{x}_1, \tilde{x}_2\}$ показан на рисунке 3.5. Таблица остаточной функции для этого примера приведена в таблице 3.2.

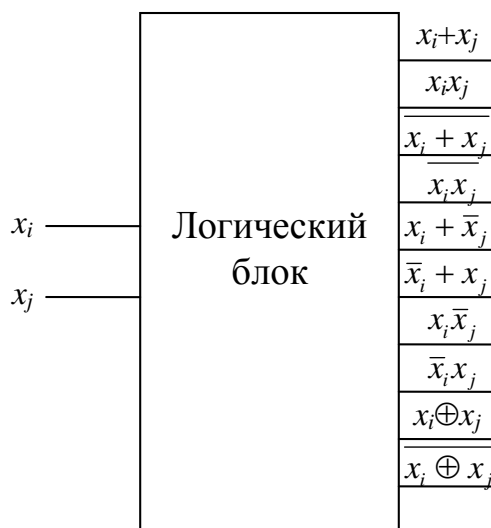


Рисунок 3.4 – Логический блок выработки сигналов настройки УЛМ с переносом двух аргументов в сигналы настройки

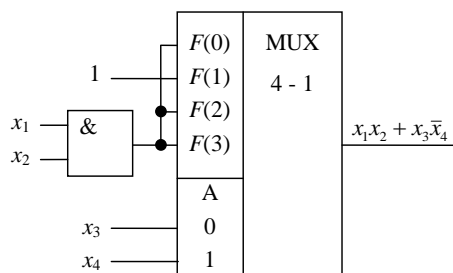


Рисунок 3.5 – Схема воспроизведения функции четырех аргументов на мультиплексоре «4 – 1»

Таблица 3.2 – Таблица истинности остаточной функции

X_4	X_3	$F_{\text{ост}}$
0	0	X_2X_1
0	1	1
1	0	X_2X_1
1	1	X_2X_1

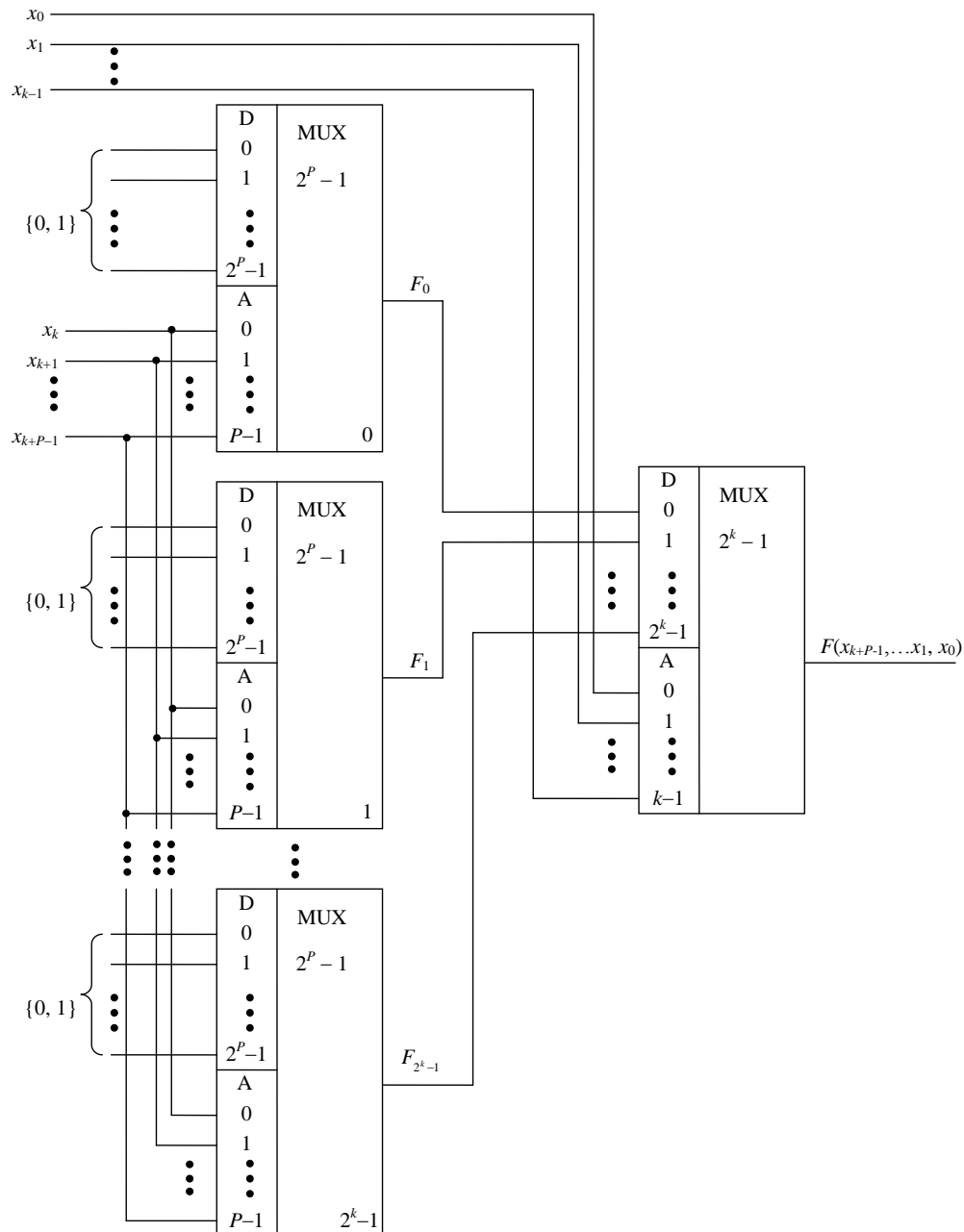
Пирамидальные структуры УЛМ. Дальнейшее расширение алфавита настройки за счет переноса трех и более переменных в сигналы настройки требует вычислений остаточных функций трех или более переменных. Вычисление таких остаточных функций с помощью мультиплексоров приводит к пирамидальной структуре (рисунок 3.6), в которой мультиплексоры первого яруса реализуют остаточные функции, а мультиплексор второго яруса вырабатывает искомую функцию.

Показанная пирамидальная структура – каноническое решение, которое приводит к нужному результату, но не претендует на оптимальность. Дело в том, что варианты построения схем из нескольких мультиплексоров для воспроизведения функций многих переменных разнообразны, но алгоритм поиска оптимального по затратам оборудования или какому-либо другому критерию отсутствует. Имеются работы, в которых найдены решения более высокого качества, но это результаты изобретений, касающиеся частных случаев, и не относятся к регулярному методу поиска структур.

При чисто электронной настройке константами 0 и 1 схема воспроизводит функцию n аргументов, где $n = k + p$, причем k – число аргументов, подаваемых на мультиплексор второго яруса, p – число аргументов, от которых зависят остаточные функции, воспроизводимые мультиплексорами $0 \dots (2^k - 1)$ первого яруса.

Для уменьшения аппаратных затрат в схеме следует стремиться к минимизации числа мультиплексоров в столбце, т.е. минимизации k и, соответственно, максимальным p , поскольку их сумма $k + p$ постоянна и равна n .

Сигналы настройки для мультиплексоров первого яруса можно искать разными способами:



**Рисунок 3.6 – Структура УЛМ,
построенного на нескольких мультиплексах**

1. Подстановкой (фиксацией) наборов аргументов, подаваемых на адресные входы мультиплексов для получения остаточных функций и, далее, сигналов настройки. Этот способ уже рассмотрен (таблицы 3.1, 3.2).

2. С помощью разложения функции по Шеннону. Это разложение можно произвести по разному числу переменных. По одному из аргументов разложение имеет вид:

$$F(x_{n-1}, \dots, x_1, x_0) = \bar{x}_0 \cdot F(x_{n-1}, \dots, x_1, 0) + x_0 \cdot F(x_{n-1}, \dots, x_1, 1).$$

Справедливость такого разложения видна из подстановки в него значений $x_0 = 0$ и $x_0 = 1$, что дает непосредственно функции $F(x_{n-1}, \dots, x_1, 0)$ и $F(x_{n-1}, \dots, x_1, 1)$.

Разложение функции по двум аргументам:

$$F(x_{n-1}, \dots, x_1, x_0) = \bar{x}_1 \bar{x}_0 \cdot F(x_{n-1}, \dots, x_2, 0, 0) + \bar{x}_1 x_0 \cdot F(x_{n-1}, \dots, x_2, 0, 1) + \\ + x_1 \bar{x}_0 \cdot F(x_{n-1}, \dots, x_2, 1, 0) + x_0 x_1 \cdot F(x_{n-1}, \dots, x_2, 1, 1)$$

и, наконец, разложение по k аргументам:

$$F(x_{n-1}, \dots, x_1, x_0) = \bar{x}_{k-1} \bar{x}_{k-2} \dots \bar{x}_1 \bar{x}_0 \cdot F(x_{n-1}, \dots, x_k, 0, \dots, 0, 0) + \\ + \bar{x}_{k-1} \bar{x}_{k-2} \dots \bar{x}_1 x_0 \cdot F(x_{n-1}, \dots, x_k, 0, \dots, 0, 1) + \dots \\ \dots + x_{k-1} x_{k-2} \dots x_1 x_0 \cdot F(x_{n-1}, \dots, x_k, 1, \dots, 1, 1) = \\ = \bar{x}_{k-1} \bar{x}_{k-2} \dots \bar{x}_1 \bar{x}_0 \cdot F_0 + \bar{x}_{k-1} \bar{x}_{k-2} \dots \bar{x}_1 x_0 \cdot F_1 + \dots + x_{k-1} x_{k-2} \dots x_1 x_0 \cdot F_{2^{k-1}},$$

где

$$F_0 = F(x_{n-1}, \dots, x_k, 0, \dots, 0, 0), \\ F_1 = F(x_{n-1}, \dots, x_k, 0, \dots, 0, 1), \\ \dots \\ F_{2^{k-1}} = F(x_{n-1}, \dots, x_k, 1, \dots, 1, 1).$$

Структура формул разложения полностью соответствует реализации двухъярусных УЛМ. В первом ярусе реализуются функции F_i , ($i = 0, \dots, 2^k - 1$), зависящие от $n - k$ аргументов, которые используются как настроечные для второго яруса, мультиплексор которого воспроизводит функцию k аргументов.

3. Сигналы настройки можно получить непосредственно из таблицы истинности функции. **Для удобства просмотра таблицы ее следует записать так, чтобы аргументы, переносимые в сигналы настройки, играли роль младших разрядов в словах-наборах аргументов.** Пусть имеется функция четырех переменных $f(x_3, x_2, x_1, x_0)$, и переменная x_3 считается старшим разрядом вектора аргументов. Пусть, далее, функция задана перечислением наборов аргументов, на которых она принимает единичные значения, причем заданы десятичные значения этих наборов: 3, 4, 5, 6, 7, 11, 15. Заметим, что аналитическое значение этой функции имеет вид $F = x_0 x_1 + x_2 \bar{x}_3$. Значения функции сведены в таблицу 3.3.

При электронной настройке УЛМ константами 0 и 1 требуется мультиплексор размерности «16 – 1», на настроечные входы УЛМ подаются значения самой функции из таблицы.

При переносе литерала \tilde{x}_0 в сигналы настройки (алфавит настройки $\{0, 1, \tilde{x}_0\}$) требуется найти остаточную функцию, аргументами которой является вектор переменных x_3, x_2, x_1 . Каждая комбинация этих переменных встречается в двух смежных строках таблицы. Просматривая таблицу по смежным парам строк, можно видеть, что остаточная функция соответствует другой таблице (таблица 3.4).

Таблица 3.3 – Таблица истинности функции четырех аргументов

X_3	X_2	X_1	X_0	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Таблица 3.4 – Таблица истинности остаточной функции трех аргументов

X_3	X_2	X_1	F
0	0	0	0
0	0	1	X_0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	X_0
1	1	0	0
1	1	1	X_0

Для реализации этого варианта УЛМ достаточен мультиплексор «8 – 1», но для перестройки на другую функцию потребуется не только смена кода настройки, но и коммутация входов настройки для подачи литералов переменной на другие настроечные входы.

При переносе в сигналы настройки двух переменных (\tilde{x}_0 и \tilde{x}_1) для поиска остаточных функция следует просмотреть четверки смежных строк таблицы с неизменными наборами x_3x_2 – аргументами, подаваемыми на адресные входы УЛМ. Этот просмотр приводит к следующей таблице истинности (таблица 3.5).

Из таблицы видно, что для воспроизведения функции достаточно использовать мультиплексор «4 – 1» с дополнительным конъюнктом для получения произведения x_1x_0 . Но при перестройке на другую функцию потре-

буются и другие функции двух переменных, т.е. универсальный логический модуль должен включать в свой состав дополнительный логический блок (см. рисунок 3.4).

Таблица 3.5 – Таблица истинности остаточной функции двух аргументов

X_3	X_2	F
0	0	X_1X_0
0	1	1
1	0	X_1X_0
1	1	X_1X_0

3.4 Пример различных способов настройки УЛМ

Логические функции $f(x_2, x_1, x_0)$, $g(x_3, x_2, x_1, x_0)$ и $h(x_4, x_3, x_2, x_1, x_0)$ заданы в виде следующих булевых алгебраических выражений:

$$f(x_2, x_1, x_0) = x_0 + \bar{x}_2x_1; \quad (3.1)$$

$$g(x_3, x_2, x_1, x_0) = x_2x_1x_0 + \bar{x}_3; \quad (3.2)$$

$$h(x_4, x_3, x_2, x_1, x_0) = x_2x_0 + x_4\bar{x}_3x_1. \quad (3.3)$$

Для логической функции $f(x_2, x_1, x_0)$ требуется аппаратно реализовать ее на основе УЛМ первым способом – фиксацией информационных входов сигналами настройки с алфавитом $\{0, 1\}$.

Для логической функции $g(x_3, x_2, x_1, x_0)$ требуется аппаратно реализовать ее на основе УЛМ вторым способом – переносом двух аргументов в сигналы настройки и алфавитом $\{0, 1, \tilde{x}_1, \tilde{x}_0\}$.

Для логической функции $h(x_4, x_3, x_2, x_1, x_0)$ требуется аппаратно реализовать ее третьим способом – на основе пирамидальной двухъярусной структуры УЛМ.

Правильность аппаратной реализации каждым способом проверить в программе MicroCAP путем сравнения с проверочными схемами. Под проверочными схемами будем понимать структурные схемы в базисе НЕ, И, ИЛИ, выполненные непосредственно по выражениям (3.1) – (3.3).

I этап. Синтез логического устройства на основе УЛМ с алфавитом настройки $\{0, 1\}$.

При выполнении этапа исследования использованы приемы №1 – 7 раздела «Типовые приемы работы в MicroCAP...».

Логическое устройство должно быть реализовано на основе УЛМ, поэтому целесообразно, чтобы окончательным видом данной функции $f(x_2, x_1, x_0)$ являлась СДНФ. СДНФ содержит все адреса, по которым нужно хранить единичные значения функции. Набор аргументов СДНФ есть адрес конкретного информационного входа УЛМ.

Для перевода функции $f(x_2, x_1, x_0)$ в СДНФ следует конъюнктивные члены, не содержащие переменной x_i , умножить на равную единице дизъюнкцию $(x_i + \bar{x}_i)$:

$$\begin{aligned} f(x_2, x_1, x_0) &= x_0 + \bar{x}_2 x_1 = x_0(x_2 + \bar{x}_2)(x_1 + \bar{x}_1) + \bar{x}_2 x_1(x_0 + \bar{x}_0) = \\ &= (x_2 x_0 + \bar{x}_2 x_0)(x_1 + \bar{x}_1) + \bar{x}_2 x_1 x_0 + \bar{x}_2 x_1 \bar{x}_0 = \\ &= x_2 x_1 x_0 + \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + \bar{x}_2 \bar{x}_1 x_0 + \bar{x}_2 x_1 x_0 + \bar{x}_2 x_1 \bar{x}_0 = \\ &= x_2 x_1 x_0 + \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + \bar{x}_2 \bar{x}_1 x_0 + \bar{x}_2 x_1 \bar{x}_0. \end{aligned}$$

Используя полученную СДНФ функции, восстановим таблицу истинности (таблица 3.6). Анализ таблицы 3.6 позволяет сформулировать словесное правило настройки УЛМ константами: на информационные входы D1, D2, D3, D5, D7 мультиплексора следует подать логическую единицу, на остальные информационные входы – логический ноль. При этом адресные входы мультиплексора подключаются к цифровому источнику сигналов x_2, x_1, x_0 .

Таблица 3.6 – Таблица истинности функции $f(x_2, x_1, x_0)$

X_2	X_1	X_0	$F(X_2, X_1, X_0)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Структурная схема логического устройства на основе УЛМ с алфавитом настройки $\{0, 1\}$, подготовленная в программе MicroCAP, представлена в верхней части рисунка 3.7. В нижней части приведена проверочная схема, реализованная непосредственно по (3.1).

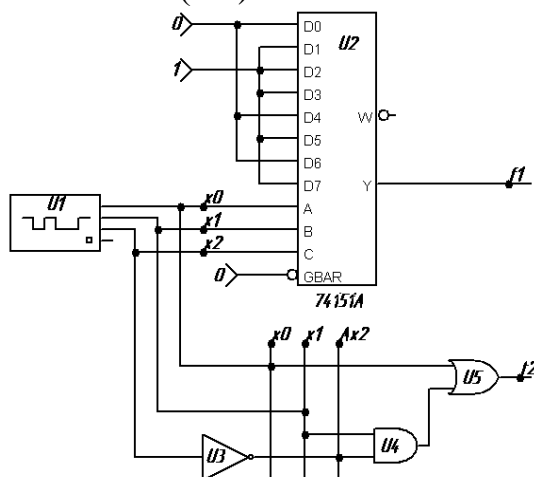
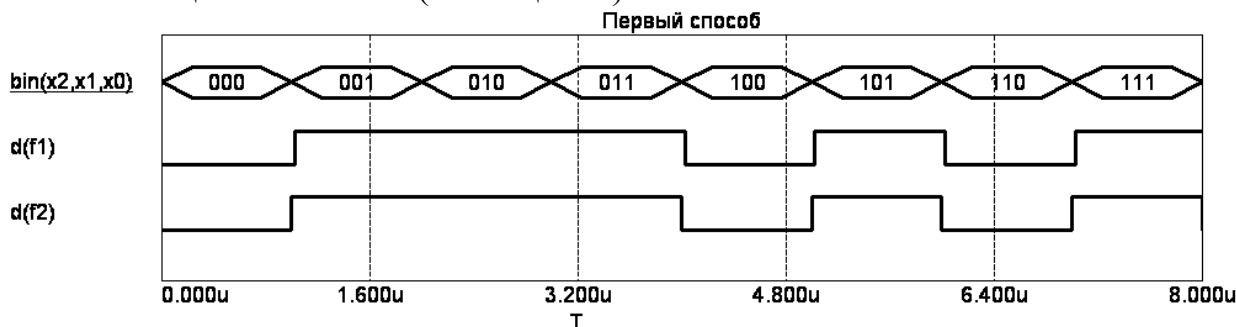


Рисунок 3.7 – Структурная и проверочная схемы логического устройства по первому способу настройки

В схеме применен селектор-мультиплексор на 8 каналов 74151А (отечественный аналог КМ155КП7). Временная зависимость входных сигналов x_2 , x_1 , x_0 на рисунке 3.8 представлена в виде состояния соответствующей трехразрядной шины.

Результаты моделирования говорят о совпадении в двух схемах временной зависимости выходного сигнала устройства и соответствии этого сигнала таблице истинности (таблица 3.6).



$d(f1)$ – на основе УЛМ; $d(f2)$ – на основе проверочной схемы

Рисунок 3.8 – Временная диаграмма функции $f(x_2, x_1, x_0)$

II этап. Синтез логического устройства на основе УЛМ с алфавитом настройки $\{0, 1, \tilde{x}_1, \tilde{x}_0\}$.

При выполнении этапа исследования использованы приемы №1 – 7 раздела «Типовые приемы работы в MicroCAP...».

Если в сигналы настройки УЛМ перевести два аргумента исходной логической функции, то число входов аргументов (адресные входы мультиплексора) сократится на два, а число настроечных входов – в четыре раза. При этом дополнительные логические схемы будут двухходовыми вентилями, что мало усложняет УЛМ и оказывается приемлемым решением.

Поскольку в исходной логической функции $g(x_3, x_2, x_1, x_0) = x_2 x_1 x_0 + \bar{x}_3$ все аргументы встречаются по одному разу, можно произвольно выбрать аргументы, подлежащие переносу в сигналы настройки, например, x_1 и x_0 . Для двух оставшихся аргументов x_3 и x_2 проведем верификацию четырех возможных сочетаний.

$$\text{При } x_3 = 0 \text{ и } x_2 = 0 \text{ имеем } g(x_3, x_2) = 0 \cdot x_1 x_0 + \bar{0} = 1.$$

$$\text{При } x_3 = 0 \text{ и } x_2 = 1 \text{ имеем } g(x_3, x_2) = 1 \cdot x_1 x_0 + \bar{0} = 1.$$

$$\text{При } x_3 = 1 \text{ и } x_2 = 0 \text{ имеем } g(x_3, x_2) = 0 \cdot x_1 x_0 + \bar{1} = 0.$$

$$\text{При } x_3 = 1 \text{ и } x_2 = 1 \text{ имеем } g(x_3, x_2) = 1 \cdot x_1 x_0 + \bar{1} = x_1 x_0.$$

По результатам приведенной верификации составим таблицу истинности остаточной функции (таблица 3.7). Из таблицы 3.7 видно, что для аппаратной реализации исходной функции $g(x_3, x_2, x_1, x_0)$ потребуется мультиплексор «4 – 1», на адресные входы которого поступают сигналы x_3, x_2 ; причем

младший разряд x_2 должен поступать обязательно на младший адресный вход. На информационные входы D0...D2 будут поданы логические константы 1, 1, 0, соответственно. К информационному входу D3 подключается логический элемент 2И, на входы которого поступают сигналы x_1, x_0 .

Таблица 3.7 – Таблица истинности остаточной функции

X_3	X_2	$G(X_3, X_2)$
0	0	1
0	1	1
1	0	0
1	1	$X_1 \cdot X_0$

По словесному описанию можно разработать в программе MicroCAP структурную схему логического устройства на основе УЛМ (рисунок 3.9, верхний фрагмент). На нижнем фрагменте рисунка приведена проверочная схема, реализованная непосредственно по выражению (3.2).

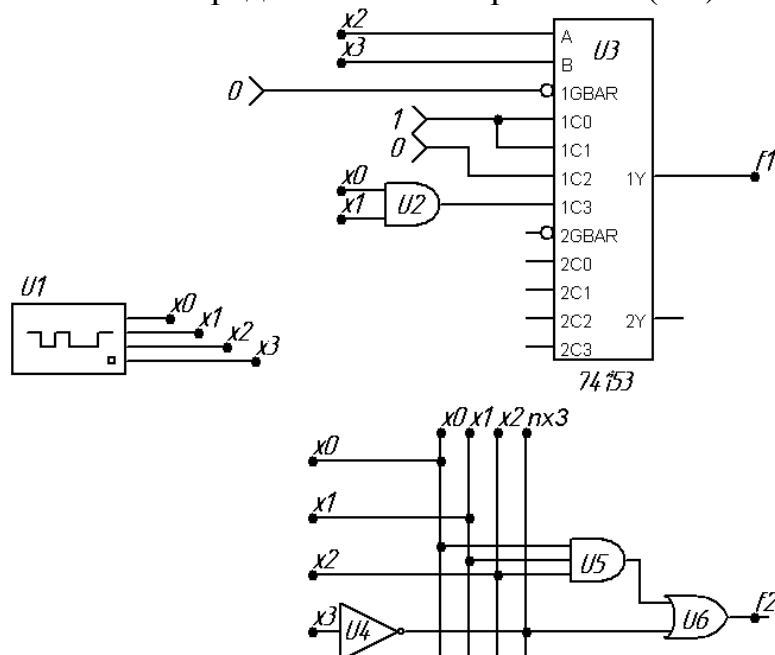
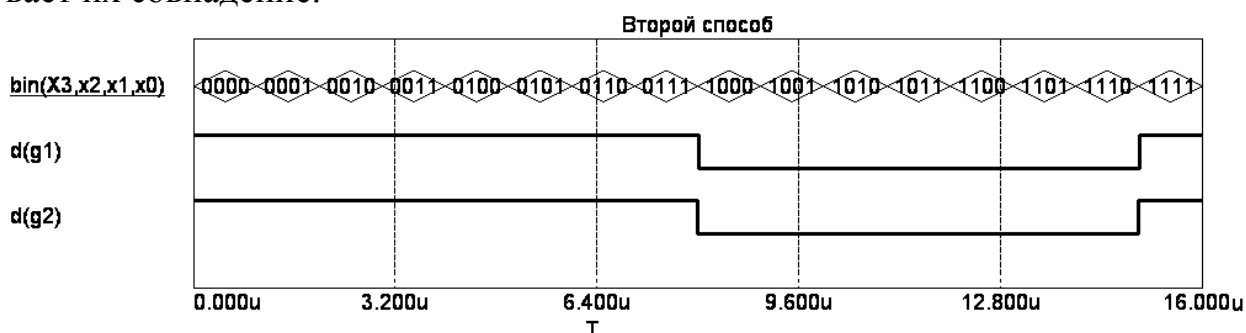


Рисунок 3.9 - Структурная и проверочная схемы логического устройства по второму способу настройки

В схеме применен сдвоенный селектор-мультиплексор 74153 типа «2 × (4 – 1)» (отечественный аналог К155КП2). Вторая независимая часть мультиплексора в схеме не используется. Для размещения графического образа на поле чертежа применяется команда *Component/Digital Library/74xx120-/153-/74153*. Установка каких-либо параметров в диалоговом окне свойств мультиплексора не требуется – он уже ассоциирован со своей математической моделью. На графическом образе мультиплексора:

- группы 1C0...1C3 и 2C0...2C3 – информационные входы первой и второй независимых частей мультиплексора;
- A, B – адресные входы, причем A – младший разряд;
- инверсные входы 1GBAR и 2GBAR – входы разрешения работы для первой и второй частей мультиплексора;
- выходы 1Y и 2Y – выходы первой и второй частей мультиплексора.

Временные зависимости выходного сигнала для двух схем представлены на рисунке 3.10. Входной сигнал изображен в виде четырехразрядной шины $x_3x_2x_1x_0$. Сравнительный анализ выходных сигналов в двух схемах показывает их совпадение.



$d(g1)$ – на основе УЛМ; $d(g2)$ – на основе проверочной схемы

Рисунок 3.10 – Временная диаграмма функции $g(x_3, x_2, x_1, x_0)$

III этап. Синтез логического устройства на основе пирамидальной структуры УЛМ.

При выполнении этапа исследования использованы приемы №1 – 7 раздела «Типовые приемы работы в MicroCAP...».

По условию логическая функция пяти аргументов $h(x_4, x_3, x_2, x_1, x_0)$ должна быть аппаратно реализована в виде двухъярусной пирамидальной структуры УЛМ. Для таких структур основным уравнением является:

$$n = k + p,$$

где n – количество аргументов исходной функции; k – число аргументов, подаваемых на мультиплексор второго яруса; p – число аргументов, от которых зависят остаточные функции, воспроизводимые мультиплексорами первого яруса.

Вообще числа k и p могут быть выбраны произвольно, однако напомним, что следует стремиться к *разумным* минимальным значениям k . Зададимся значением $k = 2$, тогда $p = n - k = 3$. Учтем также, что каноническое решение (пирамидальная структура) предполагает подачу на мультиплексоры второго яруса аргументов, играющих роль младших разрядов (см. п. 3.3). Этой информации достаточно, чтобы предложить словесное описание пирамидальной структуры УЛМ для нашего примера.

Во втором ярусе находится мультиплексор типа «4 – 1». Число подаваемых на него аргументов определено нами как $k = 2$; имена аргументов будут x_1x_0 ; число информационных входов $2^k = 2^2 = 4$.

В первом ярусе находятся четыре мультиплексора типа «8 – 1». Число мультиплексоров первого яруса равно числу информационных входов мультиплексора второго яруса. Число подаваемых аргументов на каждый мультиплексор первого яруса $p = 3$; имена аргументов $x_4x_3x_2$; количество информационных входов каждого мультиплексора $2^p = 2^3 = 8$.

Сигналы настройки для мультиплексоров первого яруса найдем с помощью разложения по Шеннону. Согласно приведенному выше описанию во втором ярусе УЛМ действует два аргумента x_1x_0 . Значит, требуется провести разложение по этим двум аргументам, чтобы получить четыре остаточные функции для мультиплексоров первого яруса.

$$\begin{aligned} h(x_4, x_3, x_2, x_1, x_0) &= x_2x_0 + x_4\bar{x}_3x_1 = \bar{x}_1\bar{x}_0 \cdot (x_2 \cdot 0 + x_4\bar{x}_3 \cdot 0) + \\ &+ \bar{x}_1x_0 \cdot (x_2 \cdot 1 + x_4\bar{x}_3 \cdot 0) + x_1\bar{x}_0 \cdot (x_2 \cdot 0 + x_4\bar{x}_3 \cdot 1) + x_1x_0 \cdot (x_2 \cdot 1 + x_4\bar{x}_3 \cdot 1) = \\ &= \bar{x}_1\bar{x}_0 \cdot h_0 + \bar{x}_1x_0 \cdot h_1 + x_1\bar{x}_0 \cdot h_2 + x_1x_0 \cdot h_3. \end{aligned}$$

Внимание! Порядок следования наборов аргументов, по которым ведется разложение, имеет большое значение. Наборы аргументов должны образовывать возрастающую последовательность двоичных чисел, символическую запись которых отражают наборы. Для нашего примера порядок следования наборов аргументов $\bar{x}_1\bar{x}_0$, \bar{x}_1x_0 , $x_1\bar{x}_0$, x_1x_0 в разложении соответствует естественному порядку счета $(00)_2$, $(01)_2$, $(10)_2$, $(11)_2$.

Представим в явном и отдельном виде остаточные функции:

$$\begin{cases} h_0(x_4, x_3, x_2) = 0, \\ h_1(x_4, x_3, x_2) = x_2, \\ h_2(x_4, x_3, x_2) = x_4\bar{x}_3, \\ h_3(x_4, x_3, x_2) = x_2 + x_4\bar{x}_3. \end{cases}$$

Настройка мультиплексора, воспроизводящего остаточную функцию h_0 , не представляет затруднений. На все информационные входы этого мультиплексора подается логический ноль.

Остаточную функцию $h_1(x_4, x_3, x_2) = x_2$ представим в СДНФ.

$$\begin{aligned} h_2(x_4, x_3, x_2) &= x_2 = x_2(x_3 + \bar{x}_3)(x_4 + \bar{x}_4) = (x_3x_2 + \bar{x}_3x_2)(x_4 + \bar{x}_4) = \\ &= x_4x_3x_2 + x_4\bar{x}_3x_2 + \bar{x}_4x_3x_2 + \bar{x}_4\bar{x}_3x_2. \end{aligned}$$

Наборы аргументов СДНФ отражают следующие числа:

$$\begin{aligned} x_4x_3x_2 &\Rightarrow (111)_2 = (7)_{10}; \\ x_4\bar{x}_3x_2 &\Rightarrow (101)_2 = (5)_{10}; \\ \bar{x}_4x_3x_2 &\Rightarrow (011)_2 = (3)_{10}; \\ \bar{x}_4\bar{x}_3x_2 &\Rightarrow (001)_2 = (1)_{10}. \end{aligned}$$

Значит, для мультиплексора, воспроизводящего функцию h_1 , на входы D1, D3, D5 и D7 подается логическая единица, на остальные входы – логический ноль.

Аналогично для остаточной функции $h_2(x_4, x_3, x_2) = x_4\bar{x}_3$ имеем:

$$h_2(x_4, x_3, x_2) = x_4 \bar{x}_3 = x_4 \bar{x}_3 (x_2 + \bar{x}_2) = x_4 \bar{x}_3 x_2 + x_4 \bar{x}_3 \bar{x}_2$$

Наборы аргументов СДНФ:

$$x_4 \bar{x}_3 x_2 \Rightarrow (101)_2 = (5)_{10};$$

$$x_4 \bar{x}_3 \bar{x}_2 \Rightarrow (100)_2 = (4)_{10}.$$

Для мультиплексора, воспроизводящего функцию h_2 , на входы D4 и D5 подается логическая единица, на остальные входы – логический ноль.

В случае остаточной функции $h_3(x_4, x_3, x_2) = x_2 + x_4 \bar{x}_3$ имеем:

$$\begin{aligned} h_3(x_4, x_3, x_2) &= x_2 + x_4 \bar{x}_3 = x_2 (x_4 + \bar{x}_4) (x_3 + \bar{x}_3) + x_4 \bar{x}_3 (x_2 + \bar{x}_2) = \\ &= (x_4 x_2 + \bar{x}_4 x_2) (x_3 + \bar{x}_3) + x_4 \bar{x}_3 x_2 + x_4 \bar{x}_3 \bar{x}_2 = \\ &= x_4 x_3 x_2 + \bar{x}_4 x_3 x_2 + x_4 \bar{x}_3 x_2 + \bar{x}_4 \bar{x}_3 x_2 + x_4 \bar{x}_3 x_2 + x_4 \bar{x}_3 \bar{x}_2 = \\ &= x_4 x_3 x_2 + \bar{x}_4 x_3 x_2 + x_4 \bar{x}_3 x_2 + \bar{x}_4 \bar{x}_3 x_2 + x_4 \bar{x}_3 \bar{x}_2. \end{aligned}$$

Наборы аргументов СДНФ:

$$x_4 x_3 x_2 \Rightarrow (111)_2 = (7)_{10};$$

$$\bar{x}_4 x_3 x_2 \Rightarrow (011)_2 = (3)_{10};$$

$$x_4 \bar{x}_3 x_2 \Rightarrow (101)_2 = (5)_{10};$$

$$\bar{x}_4 \bar{x}_3 x_2 \Rightarrow (001)_2 = (1)_{10};$$

$$x_4 \bar{x}_3 \bar{x}_2 \Rightarrow (100)_2 = (4)_{10}.$$

Для мультиплексора, воспроизводящего функцию h_3 , на входы D1, D3, D4, D5, D7 подается логическая единица, на остальные входы – логический ноль.

На основе проведенного анализа пирамидальной двухъярусной структуры УЛМ в программе MicroCAP составлена структурная схема логического устройства (рисунок 3.11), реализующего функцию $h(x_4, x_3, x_2, x_1, x_0)$. Проверочная схема устройства, составленная непосредственно по выражению (3.3), приведена отдельно на рисунке 3.12.

В первом ярусе пирамидального УЛМ использованы мультиплексоры 74151А; во втором ярусе задействована верхняя часть сдвоенного мультиплексора 74153. Наличие в схеме пяти аргументов потребовало применения восьмиразрядного цифрового источника сигнала Stim8. Пятиразрядного источника в программе MicroCAP не предусмотрено.

Установка параметров для источника Stim8 несколько отличается от аналогичных действий для Stim4. В диалоговом окне свойств Stim8 в строке **FORMAT** следует указать значение **44**. Две четверки трактуются как две тетрады с шестнадцатеричной ($2^4 = 16$) системой представления. Очевидно, что две тетрады образуют байт с диапазоном значений $[0, FF]_{16}$. Для нашего примера используется только часть диапазона $[0, 1F]_{16}$. При выборе строки **COMMAND** следует записать короткий программный листинг в нижней части диалогового окна:

```
.DEFINE INPUT  
+ 0us 00
```

+ LABEL begin
+ +1us INCR BY 01
+ +1us GOTO begin -1 TIMES

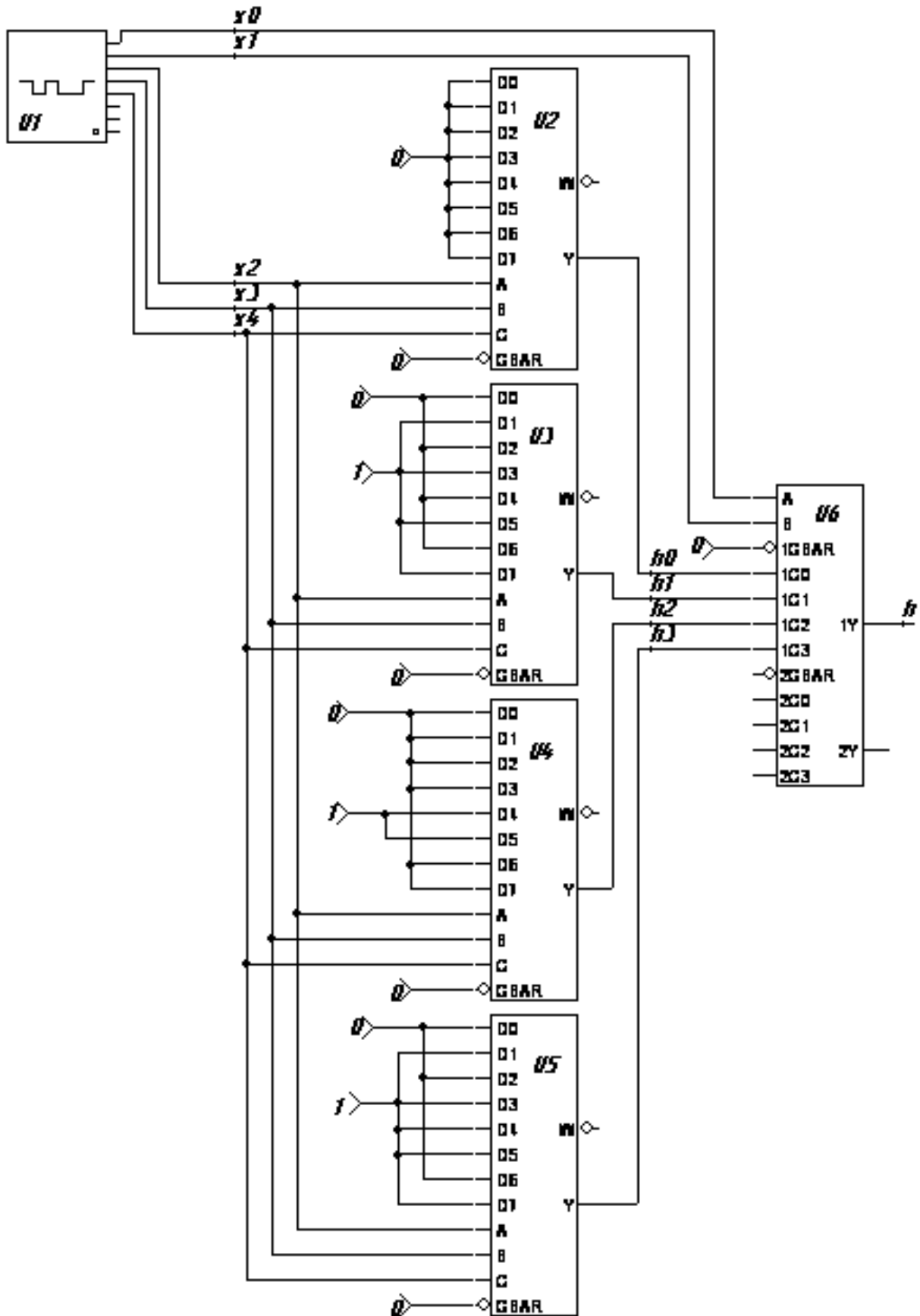


Рисунок 3.11 – Структурная схема логического устройства по третьему способу настройки

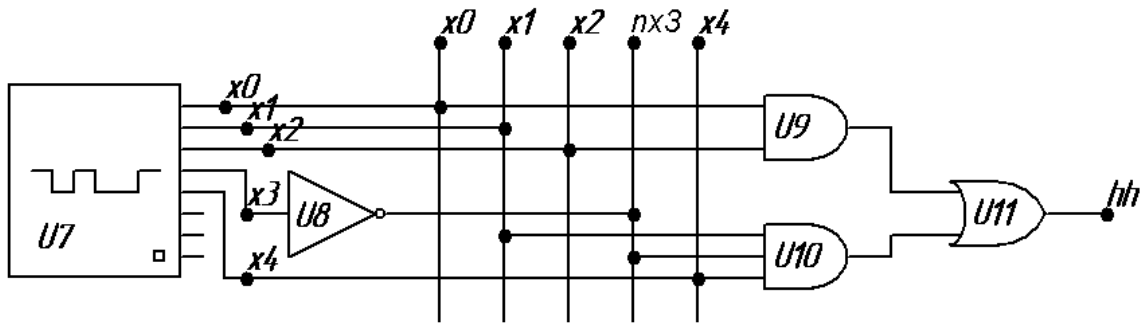
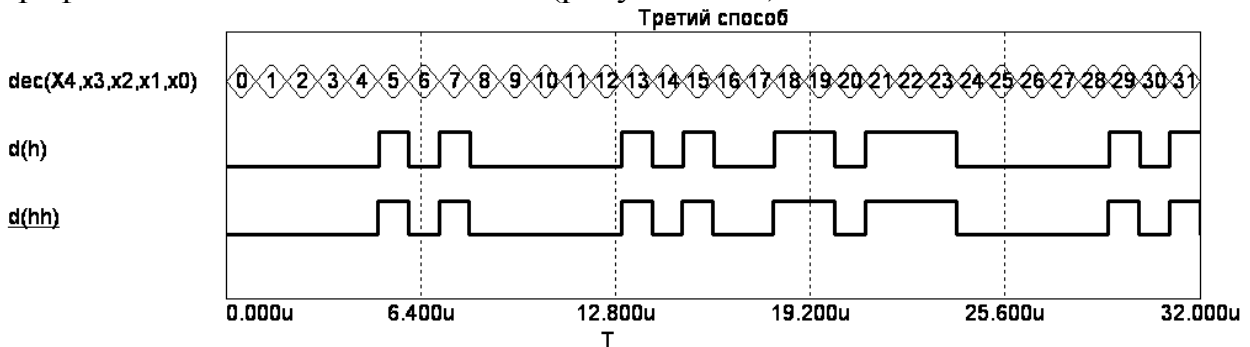


Рисунок 3.12 – Проверочная схема логического устройства по третьему способу настройки

Особое внимание при разработке схемы следует обратить на строгое соответствие старшинства разрядов аргументов и старшинства адресных входов мультиплексора. В пирамидальной структуре аргументы располагаются в порядке убывания старшинства как x_4, x_3, x_2, x_1, x_0 . Старшинство адресных входов мультиплексора убывает как С, В, А для первого яруса и В, А – для второго.

Логическая функция пяти аргументов $h(x_4, x_3, x_2, x_1, x_0)$ для отображения всех своих значений требует $2^5 = 32$ временных интервала (цикла). Общая длительность временного анализа схемы составляет $32 \cdot T$; где T – длительность одного цикла. Если в диалоговом окне свойств источника Stim8 было определено, что $T = 1$ мкс, то в строке ввода **Time Range** диалогового окна **Transient Analysis Limits** следует указать **32u**.

Состояния пятиразрядной шины аргументов $x_4x_3x_2x_1x_0$ представлены на графике в виде десятичных чисел (рисунок 3.13).



$d(h)$ – на основе УЛМ; $d(hh)$ – на основе проверочной схемы

Рисунок 3.13 – Временная диаграмма функции $h(x_4, x_3, x_2, x_1, x_0)$

Из рисунка видно, что временные зависимости выходных сигналов для двух схем взаимно совпадают, значит III этап исследования проведен адекватно.

3.5 Лабораторное задание

Повторить методический пример, приведенный выше, по исходным данным Вашего варианта. *Критерием правильности* проведенного исследования является совпадение временных зависимостей выходного сигнала для структурной и проверочной схемы на I, II и III этапах.

3.6 Контрольные вопросы

1. Сравнение структурных (на основе УЛМ) и проверочных схем показывает, что проверочные схемы гораздо проще. В чем тогда заключается преимущество аппаратной реализации логической функции с помощью УЛМ?
2. Что подается на входы мультиплексора при использовании его в качестве УЛМ?
3. Что такое алфавит настройки?
4. Что такое литерал?
5. Какой именно аргумент целесообразно переносить в сигналы настройки УЛМ?
6. Каково правило уменьшения аппаратных затрат для случая пирамидальной двухъярусной структуры УЛМ?
7. Что такое разложение по Шеннону?
8. Что такое остаточная функция?

3.7 Варианты заданий

Таблица 3.8 – Варианты задания для первого способа настройки УЛМ

НОМЕР ВАРИАНТА	ЛОГИЧЕСКАЯ ФУНКЦИЯ $F(X_2, X_1, X_0)$
1	$\bar{x}_0 + x_2\bar{x}_1$
2	$x_1x_0 + x_2x_1 + x_2x_0$
3	$x_1x_0 + \bar{x}_2\bar{x}_1 + x_2x_0$
4	$x_2x_1x_0 + \bar{x}_2\bar{x}_1\bar{x}_0$
5	$\bar{x}_1x_0 + x_2x_0$
6	$x_2 + x_1 + x_0$
7	$x_2 + \bar{x}_1 + x_0$
8	$\bar{x}_1\bar{x}_0 + x_2x_1$
9	$x_2\bar{x}_0 + \bar{x}_2x_1$
10	$\bar{x}_0 + \bar{x}_2x_1$
11	$x_2 + x_1 + \bar{x}_0$
12	$x_1x_0 + \bar{x}_2\bar{x}_1 + \bar{x}_2x_0$
13	$x_2x_1 + \bar{x}_1\bar{x}_0 + \bar{x}_2\bar{x}_1$
14	$x_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0$
15	$\bar{x}_2 + x_1 + x_0$
16	$x_2x_1 + x_2\bar{x}_0$
17	$\bar{x}_2x_1x_0 + x_2\bar{x}_1\bar{x}_0$
18	$\bar{x}_2 + \bar{x}_1 + \bar{x}_0$
19	$x_1\bar{x}_0 + \bar{x}_2x_1 + x_2x_0$
20	$\bar{x}_2 + \bar{x}_1\bar{x}_0$
21	$\bar{x}_2x_1 + x_1\bar{x}_0 + \bar{x}_2x_0$
22	$\bar{x}_2x_1\bar{x}_0 + x_2\bar{x}_1\bar{x}_0$
23	$\bar{x}_2\bar{x}_1 + \bar{x}_1\bar{x}_0$
24	$x_2 + \bar{x}_1 + \bar{x}_0$
25	$x_2 + \bar{x}_1x_0$

Таблица 3.9 – Варианты задания для второго способа настройки УЛМ

НОМЕР ВАРИАНТА	ЛОГИЧЕСКАЯ ФУНКЦИЯ $G(X_3, X_2, X_1, X_0)$
1	$x_1x_0 + x_2x_0 + x_3$
2	$x_0 + x_1 + \bar{x}_2 + x_3$
3	$x_2\bar{x}_1x_0 + x_3$
4	$x_3\bar{x}_2\bar{x}_1x_0$
5	$x_0 + \bar{x}_1 + x_2 + \bar{x}_3$
6	$x_1x_0 + x_3x_2$
7	$\bar{x}_1x_0 + x_3\bar{x}_2$
8	$x_0 + x_2x_1 + x_3x_2$
9	$x_0 + \bar{x}_2\bar{x}_1 + x_3x_2$
10	$x_2x_1x_0 + x_3x_2x_1$
11	$x_2x_1x_0 + \bar{x}_3\bar{x}_2\bar{x}_1$
12	$x_2\bar{x}_1x_0 + \bar{x}_3x_2\bar{x}_1$
13	$x_3x_2x_1x_0 + x_3\bar{x}_2\bar{x}_1x_0$
14	$x_1\bar{x}_0 + x_3x_2$
15	$x_0 + \bar{x}_2x_1 + \bar{x}_3$
16	$x_1x_0 + x_2x_0 + x_3x_0$
17	$x_1\bar{x}_0 + \bar{x}_2x_0 + x_3\bar{x}_0$
18	$x_2x_1x_0 + x_3x_1x_0$
19	$\bar{x}_0 + \bar{x}_1 + \bar{x}_2 + \bar{x}_3$
20	$x_1x_0 + \bar{x}_2\bar{x}_0 + x_3$
21	$\bar{x}_3 + x_2 + x_1 + \bar{x}_0$
22	$x_3\bar{x}_2x_0 + \bar{x}_1$
23	$\bar{x}_3x_2x_1\bar{x}_0$
24	$x_3\bar{x}_0 + x_2\bar{x}_1$
25	$x_3\bar{x}_1x_0 + x_2x_1\bar{x}_0$

Таблица 3.10 – Варианты задания для третьего способа настройки УЛМ

НОМЕР ВАРИАНТА	ЛОГИЧЕСКАЯ ФУНКЦИЯ $H(X_4, X_3, X_2, X_1, X_0)$
1	$x_1x_0 + x_4x_2x_0 + x_3$
2	$x_0 + x_1 + \bar{x}_2 + x_3 + x_4$
3	$x_2\bar{x}_1x_0 + x_4x_3$
4	$x_3x_2x_1x_0 + \bar{x}_4$
5	$x_0 + \bar{x}_1 + x_2 + \bar{x}_3 + x_4$
6	$x_1x_0 + x_3x_2 + x_4$
7	$\bar{x}_1x_0 + x_3\bar{x}_2 + x_4$
8	$x_0 + x_2x_1 + x_4x_3x_2$
9	$x_0 + \bar{x}_2\bar{x}_1 + x_4x_3x_2$
10	$x_2x_1x_0 + x_3x_2x_1 + x_4x_3x_2$
11	$x_2x_1x_0 + \bar{x}_3\bar{x}_2\bar{x}_1 + x_4x_3x_2$
12	$x_2\bar{x}_1x_0 + \bar{x}_3x_2\bar{x}_1 + x_4\bar{x}_3x_2$
13	$x_3x_2x_1x_0 + \bar{x}_4x_2\bar{x}_1x_0$
14	$x_1\bar{x}_0 + x_3x_2 + \bar{x}_4$
15	$x_0 + \bar{x}_2x_1 + x_4\bar{x}_3$
16	$x_4x_0 + x_1x_0 + x_2x_0 + x_3x_0$
17	$\bar{x}_4x_0 + x_1\bar{x}_0 + \bar{x}_2x_0 + x_3\bar{x}_0$
18	$x_2x_1x_0 + x_3x_1x_0 + x_4x_1x_0$
19	$\bar{x}_0 + \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4$
20	$\bar{x}_4x_1 + x_3\bar{x}_2x_1 + x_0$
21	$\bar{x}_4 + x_3 + \bar{x}_2 + x_1 + \bar{x}_0$
22	$x_4\bar{x}_3\bar{x}_2 + \bar{x}_1x_0$
23	$x_4\bar{x}_3\bar{x}_2x_1 + x_0$
24	$\bar{x}_4\bar{x}_3 + \bar{x}_2\bar{x}_1 + \bar{x}_0$
25	$x_3\bar{x}_2\bar{x}_1x_0 + \bar{x}_4x_2x_1\bar{x}_0$

4 ЛАБОРАТОРНАЯ РАБОТА №4 – ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ

4.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

- 1) знакомство с особенностями проектирования синхронных автоматов с памятью;
- 2) исследование структуры автоматов Мура;
- 3) приобретение навыков двоичного кодирования состояний для синхронного автомата;
- 4) приобретение навыков по нахождению функций возбуждения для JK-триггеров.

4.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 4.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

4.3 Методика проектирования автоматов с памятью

Узлы и устройства, которые содержат элементы памяти, относятся к классу автоматов с памятью (АП). Наличие элементов памяти (ЭП) придает АП свойство иметь некоторое внутреннее состояние Q , определяемое совокупностью состояний всех элементов памяти. В зависимости от внутреннего состояния (далее называемого просто состоянием), АП различно реагирует на один и тот же вектор входных сигналов X . Воспринимая входные сигналы при определенном состоянии, АП переходит в новое состояние и вырабатывает вектор выходных переменных Y . Таким образом, для АП:

$$Q_n = f(Q, X),$$
$$Y = \varphi(Q, X),$$

где Q_n и Q – состояния АП после и до подачи входных сигналов (индекс «n» от слова «новое»).

Переходы АП из одного состояния в другое начинаются с некоторого исходного состояния Q_0 , задание которого также является частью задания автомата. Следующее состояние зависит от Q_0 и поступивших входных сигналов X . В конечном счете, текущее состояние и выходы автомата зависят от начального состояния и всех векторов X , поступавших на автомат в предшествующих сменах входных сигналов. Таким образом, вся последовательность входных сигналов определяет последовательность состояний и выходных сигналов. Это объясняет название «последовательностные схемы», также применяемое для обозначения АП.

Структурно АП отличаются от КЦ наличием в их схемах обратных связей, вследствие чего в них проявляются свойства запоминания состояний (достаточно вспомнить схемы триггерных элементов, где указанная особенность проявляется очень наглядно).

Автоматы с памятью в каноническом представлении разделяют на две части: *память* и *комбинационную цепь*. На входы КЦ подаются входные сигналы и сигналы состояния АП. На ее выходе вырабатываются выходные сигналы и сигналы перевода АП в новое состояние.

Принципиальным является деление АП на *асинхронные* и *синхронные*. В асинхронных (рисунок 4.1) роль элементов памяти играют элементы задержки, через которые сигналы состояния передаются на входы КЦ, чтобы совместно с новым набором входных переменных определить следующую пару значений Y и Q на выходе. Элементы АП переключаются здесь под непосредственным воздействием изменений информационных сигналов. Скорость распространения процесса переключений в цепях асинхронного автомата определяется собственными задержками элементов.

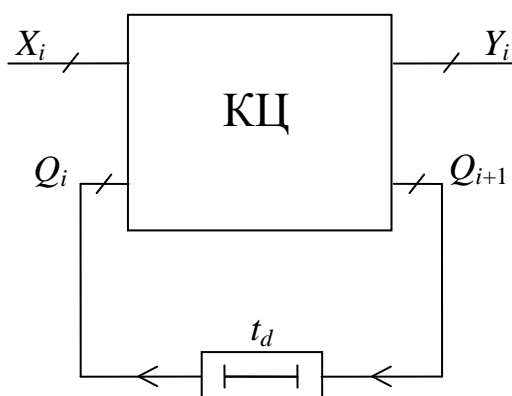


Рисунок 4.1 – Асинхронный автомат с памятью

В синхронном АП (рисунок 4.2) имеются специальные синхросигналы (тактирующие импульсы) S , которые разрешают элементам памяти прием данных только в определенные моменты времени. Элементами памяти служат синхронные триггеры. Процесс обработки информации упорядочивается во

времени, и в течение одного такта возможно распространение процесса переключения только в строго определенных пределах тракта обработки информации.

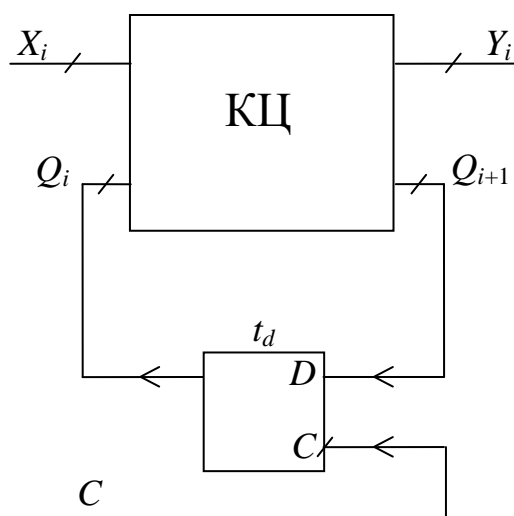


Рисунок 4.2 – Синхронный автомат с памятью

Практическое применение асинхронных автоматов существенно затруднено сильным влиянием на их работу задержек сигналов в цепях АП, создающих статические и динамические риски, гонки элементов памяти (неодновременность срабатывания ЭП даже при одновременной подаче на них входных сигналов) и др. В итоге характерным свойством асинхронного автомата является то, что при переходе из одного устойчивого состояния в другое он обычно проходит через промежуточные нестабильные состояния. Нельзя сказать, что методы борьбы с нежелательными последствиями рисков и гонок в асинхронных АП отсутствуют, но все же обеспечение предсказуемого поведения АП – сложная проблема. В сложных АП асинхронные схемы встречаются редко, а в простейших схемах применяются. Примером могут служить асинхронные *RS*-триггеры.

В синхронных автоматах каждое новое состояние устойчиво и переходные временные состояния не возникают. Концепция борьбы с последствиями рисков и гонок в синхронных автоматах проста – прием информации в элементы памяти разрешается только после завершения в схеме переходных процессов. Это обеспечивается параметрами синхроимпульсов, задающих интервалы времени для завершения тех или иных процессов. В сравнении с асинхронными, синхронные АП значительно проще в проектировании.

На сегодняшний день и достаточно длительную перспективу основным путем построения АП следует считать применение тактирования, т.е. синхронных автоматов.

В работах отечественных и зарубежных ученых разрабатывается направление, называемое проектирование самосинхронизирующихся устройств, в которых тактовые импульсы следуют с переменной частотой, за-

висящей от длительности реального переходного процесса в схеме. Однако перспективность этого направления еще не вполне ясна.

В теории автоматов проводится их классификация по ряду признаков. В схемотехнике преобладают автоматы Мура, выходы которых являются функциями только состояния автомата. Для этого автомата:

$$Q_n = f(Q, X),$$

$$Y = \varphi(Q).$$

Зависимость выходов и от состояния автомата и от вектора входных переменных свойственна автоматам Мили.

Некоторые функциональные узлы принадлежат к числу автономных автоматов, которые не имеют информационных входов, и под действием тактовых сигналов переходят из состояния в состояние по алгоритму, определяемому структурой автомата.

Проектирование автоматов. Проектирование АП содержит следующие этапы:

- исходное задание функционирования;
- формализованное задание функционирования;
- минимизация состояний;
- кодирование состояний;
- составление таблицы переходов;
- определение функций возбуждения элементов памяти (триггеров);
- минимизация функций возбуждения триггеров;
- переход к базису выбранной для реализации схемотехнологии;
- составление логической схемы;
- сборка и проверка автомата.

Исходное задание функционирования может иметь различную форму, в том числе и словесную. От нее переходят к формализованному заданию – таблицам, формулам, диаграммам состояния и т.п. Далее выполняются минимизация и кодирование состояний автомата, в результате чего получается таблица переходов, на основании которой можно найти функции возбуждения триггеров.

Минимизация и кодирование состояний в общем случае задача, решение которой может потребовать значительных усилий, однако при проектировании узлов ЭВМ и цифровой автоматики она чаще всего проста, и ее решение подсказывается самой формулировкой задания на проектирование. Традиционно *широко применяется кодирование состояний автомата двоичными кодами*, при котором триггеры используются в схеме экономно. Для некоторых новых СБИС программируемой логики, снабженных большим числом триггеров, экономия их числа при построении автомата несущественна. Для таких случаев применение кодирования кодами «1 из N » может быть предпочтительным, т.к. приводит к более быстродействующим схемам, хотя и требующим значительного числа триггеров.

Функции возбуждения триггеров, обеспечивающие переходы АП из одного состояния в другое, реализуются его комбинационной частью. Они, как сказано в перечислении этапов проектирования, минимизируются и переводятся в базис выбранных средств реализации автомата. Это положение следует понимать в широком смысле, поскольку в зависимости от средств реализации КЦ требования к формам представления функций возбуждения могут существенно различаться. Точнее можно говорить о приведении функций к виду, удобному для воспроизведения данными средствами.

После выполнения указанных действий можно получить логическую схему АП. Заканчивается процесс проверкой работы узла с помощью моделирования или макетирования.

Рассмотрим более подробно методику проектирования автоматов, содержащих триггеры (рисунок 4.3).

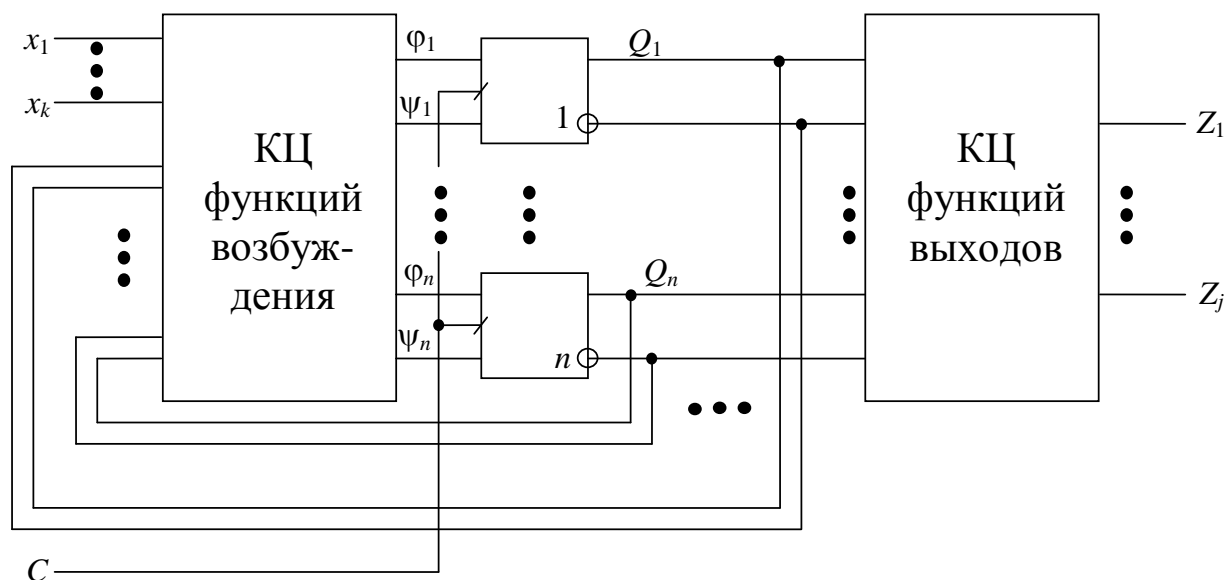


Рисунок 4.3 – Структурная схема автомата Мура

В тактируемых автоматах элементами памяти служат синхронные триггеры, причем любой автомат можно построить на любом типе триггера (D , JK , RS , T и др.).

При двоичном кодировании состояний автомата число триггеров в его схеме равно:

$$n = \lceil \log_2 N \rceil,$$

где N – число состояний автомата; $\lceil \rceil$ – знак округления до ближайшего справа целого числа.

При кодировании кодом «1 из N » число триггеров равно числу состояний автомата $n = N$, т.к. каждому состоянию соответствует один триггер в единичном состоянии при нулевом состоянии остальных.

Будем считать, что закон функционирования автомата определен, и кодирование его состояний произведено. Значит, известна последовательность состояний триггеров, принимаемых ими в каждом такте под управлением

входных сигналов x_1, x_2, \dots, x_k и текущего состояния Q_1, Q_2, \dots, Q_n . Предмет синтеза – получение функций возбуждения φ_i и ψ_i для каждого входа всех триггеров, обеспечивающих необходимые переходы автоматов.

Функции выхода для автоматов Мура зависят только от состояния автомата, поэтому нахождение выходов Z_1, Z_2, \dots, Z_j осуществляется комбинационной схемой, на которую подаются только выходы триггеров Q_1, Q_2, \dots, Q_n . Все триггеры тактируются общим синхросигналом C . После завершения выработки функций возбуждения комбинационной схемой поступает очередной тактовый сигнал C , переводящий триггеры в новое состояние.

Вид функций возбуждения зависит от логического типа триггеров. Поэтому одним из средств синтеза служат «словари» для триггеров.

При поиске функций возбуждения триггеров вначале составляется таблица (таблица 4.1), содержащая приведенные ниже данные.

Таблица 4.1 – Функции возбуждения триггеров

ВХОДЫ В МОМЕНТ ВРЕМЕНИ T				СОСТОЯНИЯ ТРИГГЕРОВ								НЕОБХОДИМЫЕ СИГНАЛЫ НА ВСЕХ ВХОДАХ КАЖДОГО ТРИГГЕРА					
				Q (СТАРОЕ)				Q_H (НОВОЕ)									
X_1	X_2	...	X_k	Q_1	Q_2	...	Q_N	Q_1	Q_2	...	Q_N	φ_1	ψ_1	...	φ_N	ψ_N	

Столбцы $\varphi_1, \psi_1, \dots, \varphi_n, \psi_n$ определяют функции возбуждения триггеров.

Многовариантность реализаций автомата связана с выбором типа триггеров и комбинационной части.

Относительно наиболее распространенных типов триггеров JK и D можно отметить следующее. Триггер типа JK обладает более развитыми логическими функциями, поэтому для него функции возбуждения в среднем более просты, но число их вдвое больше, чем для триггера D . Что же даст более простое решение, заранее неизвестно.

Комбинационная часть автомата может быть построена на логических элементах, мультиплексорах, ИС программируемой памяти, программируемых логических матрицах и т.д.

Состояния автомата можно кодировать двоичными кодами, кодами «1 из N » и др.

Автомат можно построить, приспособив к необходимому функционированию типовую ИС среднего уровня интеграции (счетчик, сдвигающий регистр), добавив к ним специально спроектированную логическую часть.

4.4 Пример проектирования автомата на основе JK-триггеров

Требуется спроектировать трехразрядный автомат на основе JK-триггеров с двумя режимами работы, управляемый входным сигналом M . При $M = 0$ автомат должен работать как двоичный счетчик с модулем счета 8, а при $M = 1$ – как счетчик в коде Грея. Комбинационные цепи автомата реализовать в базисе И-НЕ. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенного схемотехнического решения подтвердить результатами моделирования в программе MicroCAP.

Код Грея используется в системах контроля цифровых устройств, преобразователях механических перемещений в цифровой код и т.д. При переходе от предыдущей кодовой комбинации к следующей в коде Грея изменяется только один разряд. На рисунке 4.4 представлена диаграмма состояний трехразрядного кода Грея.

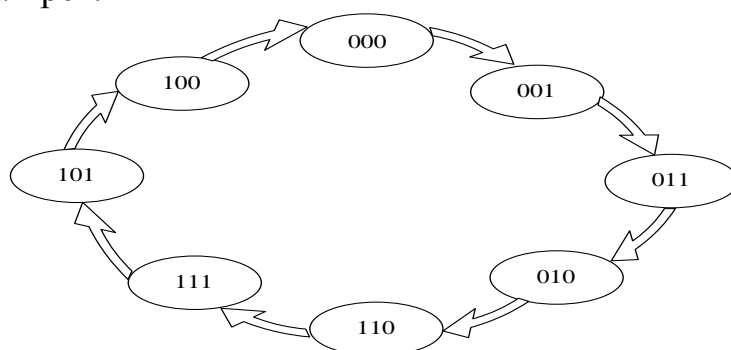


Рисунок 4.4 – Диаграмма состояний трехразрядного кода Грея

1 этап. Описание работы цифрового автомата.

На первом этапе проектирования цифрового автомата целесообразно представить его функционирование с помощью диаграммы состояний. Преимущества такого способа представления – наглядность и простота первоначального восприятия. Условимся, что переходы автомата на диаграмме состояний будут обозначены следующим образом:

- для режима двоичного счетчика как \overline{M} , что соответствует $M = 0$;
- для режима счетчика в коде Грея как M , что соответствует $M = 1$.

Восемь возможных состояний ($2^3 = 8$) цифрового автомата следует расположить на диаграмме в естественном порядке их наступления, характерном для какого-либо режима. Пусть базовая последовательность состояний на диаграмме соответствует режиму двоичного счетчика. Тогда дополнив рисунок переходами, отражающими последовательность кода Грея, получим полную диаграмму состояний двухрежимного цифрового автомата (рисунок 4.5).

Используя понятия теории графов, можно сказать, что представленная диаграмма состояний – это направленный граф, т.к. направления переходов заданы. Каждый переход на диаграмме или ребро графа соединяет старое (предыдущее) состояние автомата и новое (последующее) состояние; причем

новое состояние в такой паре всегда указано стрелкой. Перебор всех ребер графа и анализ состояний позволяет составить таблицу истинности цифрового автомата для двух режимов функционирования (таблица 4.2).

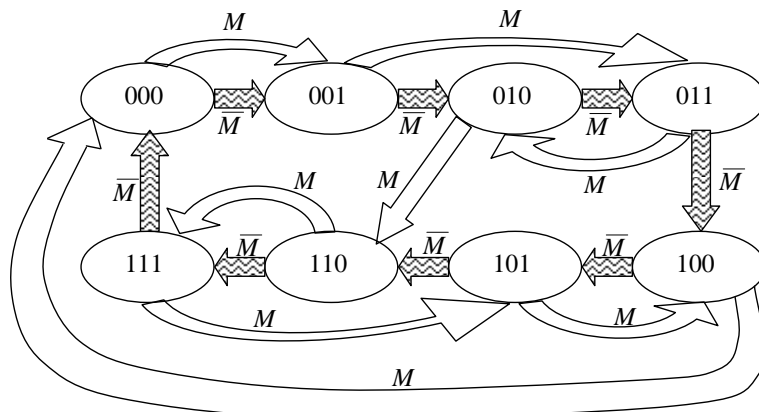


Рисунок 4.5 – Диаграмма состояний двухрежимного трехразрядного автомата

Таблица 4.2 – Таблица истинности двухрежимного цифрового автомата

ВХОДНОЙ УПРАВЛЯЮЩИЙ СИГНАЛ	ИСХОДНОЕ СОСТОЯНИЕ			НОВОЕ СОСТОЯНИЕ		
	Q_2	Q_1	Q_0	Q_{2H}	Q_{1H}	Q_{0H}
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	1	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	1	1
1	1	1	1	1	0	1

Правило заполнения таблицы истинности. Для каждого режима заполнения таблицы истинности следует начинать с блока Q_2, Q_1, Q_0 перечислением кодовых комбинаций аргументов в порядке естественного счета двоичной системы счисления: 000, 001, ..., 111. Затем, учитывая, что кодовые комбинации в блоке Q_2, Q_1, Q_0 представляют собой старые значения состоя-

ний автомата, необходимо найти для каждого такого состояния по диаграмме соответствующее новое значение и записать правее в ячейки Q_{2H} , Q_{1H} , Q_{0H} . Бессистемное (произвольное) расположение пар состояний в таблице истинности оказывается критичным для некоторых способов синтеза цифровых автоматов, например, для синтеза автоматов с мультиплексным управлением.

II этап. Составление функций переходов JK-триггеров.

При выполнении этапа исследования использован прием №8 раздела «Типовые приемы работы в MicroCAP...».

В таблице истинности (таблица 4.2) новые состояния автомата Q_{2H} , Q_{1H} , Q_{0H} являются значениями трех логических функций, а входной управляющий сигнал M и старые состояния Q_2 , Q_1 , Q_0 можно рассматривать как аргументы логических функций:

$$\begin{aligned} Q_{2H} &= f(M, Q_2, Q_1, Q_0); \\ Q_{1H} &= g(M, Q_2, Q_1, Q_0); \\ Q_{0H} &= h(M, Q_2, Q_1, Q_0). \end{aligned} \quad (4.1)$$

Пользуясь таблицей истинности, каждую из функций (4.1) можно записать сначала в СДНФ, а затем упростить до МДНФ. Автоматизируем этот процесс с помощью логического преобразователя Electronics Workbench.

$$\begin{aligned} Q_{2H} &= \bar{M} \cdot \bar{Q}_2 Q_1 Q_0 + \bar{M} \cdot Q_2 \bar{Q}_1 \bar{Q}_0 + \bar{M} \cdot Q_2 \bar{Q}_1 Q_0 + \bar{M} \cdot Q_2 Q_1 \bar{Q}_0 + \\ &+ M \cdot \bar{Q}_2 Q_1 \bar{Q}_0 + M \cdot Q_2 \bar{Q}_1 Q_0 + M \cdot Q_2 Q_1 \bar{Q}_0 + M \cdot Q_2 Q_1 Q_0 = \\ &= \bar{M} \cdot \bar{Q}_2 Q_1 Q_0 + \bar{M} \cdot Q_2 \bar{Q}_1 + \bar{M} \cdot Q_2 \bar{Q}_0 + M \cdot Q_1 \bar{Q}_0 + M \cdot Q_2 Q_0. \end{aligned} \quad (4.2)$$

$$\begin{aligned} Q_{1H} &= \bar{M} \cdot \bar{Q}_2 \bar{Q}_1 Q_0 + \bar{M} \cdot \bar{Q}_2 Q_1 \bar{Q}_0 + \bar{M} \cdot Q_2 \bar{Q}_1 Q_0 + \bar{M} \cdot Q_2 Q_1 \bar{Q}_0 + \\ &+ M \cdot \bar{Q}_2 \bar{Q}_1 Q_0 + M \cdot \bar{Q}_2 Q_1 \bar{Q}_0 + M \cdot \bar{Q}_2 Q_1 Q_0 + M \cdot Q_2 Q_1 \bar{Q}_0 = \\ &= \bar{M} \cdot \bar{Q}_1 Q_0 + M \cdot \bar{Q}_2 Q_0 + Q_1 \bar{Q}_0. \end{aligned} \quad (4.3)$$

$$\begin{aligned} Q_{0H} &= \bar{M} \cdot \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + \bar{M} \cdot \bar{Q}_2 Q_1 \bar{Q}_0 + \bar{M} \cdot Q_2 \bar{Q}_1 \bar{Q}_0 + \bar{M} \cdot Q_2 Q_1 \bar{Q}_0 + \\ &+ M \cdot \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + M \cdot \bar{Q}_2 \bar{Q}_1 Q_0 + M \cdot Q_2 Q_1 \bar{Q}_0 + M \cdot Q_2 Q_1 Q_0 = \\ &= \bar{M} \cdot \bar{Q}_0 + M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1. \end{aligned} \quad (4.4)$$

III этап. Нахождение функций возбуждения для входов JK-триггеров.

При выполнении этапа исследования использован прием №8 раздела «Типовые приемы работы в MicroCAP...».

Функции переходов триггеров Q_{iH} (4.2 – 4.4) выражены, помимо прочих аргументов, также через свое старое состояние \tilde{Q}_i . Это обстоятельство позволяет провести непосредственный сравнительный анализ функций переходов и характеристического уравнения JK-триггера, чтобы получить функции возбуждения для входов J и K каждого триггера. Известно [3], что характеристическое уравнение JK-триггера в общем виде записывается как:

$$Q_{iH} = J_i \bar{Q}_i + \bar{K}_i Q_i. \quad (4.5)$$

Значит, для сравнительного анализа необходимо, чтобы функции переходов (4.2 – 4.4) были представлены в виде:

$$Q_{iH} = \alpha_i \bar{Q}_i + \beta_i Q_i, \quad (4.6)$$

где сомножители α_i и β_i уже не содержат переменных \bar{Q}_i и Q_i .

Если воспользоваться разложением Шеннона (см. лабораторную работу №3) по одной переменной, то можно достаточно просто преобразовать функции переходов (4.2 – 4.4) к форме (4.6). В качестве переменной, по которой ведется разложение, выступает старое состояние Q_i .

$$\begin{aligned} Q_{2H} &= \bar{M} \cdot \bar{Q}_2 Q_1 Q_0 + \bar{M} \cdot Q_2 \bar{Q}_1 + \bar{M} \cdot Q_2 \bar{Q}_0 + M \cdot Q_1 \bar{Q}_0 + M \cdot Q_2 Q_0 = \\ &= \bar{Q}_2 (\bar{M} \cdot \bar{0} \cdot Q_1 Q_0 + \bar{M} \cdot 0 \cdot \bar{Q}_1 + \bar{M} \cdot 0 \cdot \bar{Q}_0 + M \cdot Q_1 \bar{Q}_0 + M \cdot 0 \cdot Q_0) + \\ &\quad + Q_2 (\bar{M} \cdot \bar{1} \cdot Q_1 Q_0 + \bar{M} \cdot 1 \cdot \bar{Q}_1 + \bar{M} \cdot 1 \cdot \bar{Q}_0 + M \cdot Q_1 \bar{Q}_0 + M \cdot 1 \cdot Q_0) = \\ &= \bar{Q}_2 (\bar{M} \cdot Q_1 Q_0 + M \cdot Q_1 \bar{Q}_0) + Q_2 (\bar{M} \cdot \bar{Q}_1 + \bar{M} \cdot \bar{Q}_0 + M \cdot Q_1 \bar{Q}_0 + M \cdot Q_0) = \\ &= \bar{Q}_2 (\bar{M} \cdot Q_1 Q_0 + M \cdot Q_1 \bar{Q}_0) + Q_2 (\bar{M} \cdot \bar{Q}_1 + \bar{M} \cdot \bar{Q}_0 + M \cdot Q_1 + M \cdot Q_0). \end{aligned}$$

$$\begin{aligned} Q_{1H} &= \bar{M} \cdot \bar{Q}_1 Q_0 + M \cdot \bar{Q}_2 Q_0 + Q_1 \bar{Q}_0 = \\ &= \bar{Q}_1 (\bar{M} \cdot \bar{0} \cdot Q_0 + M \cdot \bar{Q}_2 Q_0 + 0 \cdot \bar{Q}_0) + Q_1 (\bar{M} \cdot \bar{1} \cdot Q_0 + M \cdot \bar{Q}_2 Q_0 + 1 \cdot \bar{Q}_0) = \\ &= \bar{Q}_1 (\bar{M} \cdot Q_0 + M \cdot \bar{Q}_2 Q_0) + Q_1 (M \cdot \bar{Q}_2 Q_0 + \bar{Q}_0) = \\ &= \bar{Q}_1 (\bar{M} \cdot Q_0 + \bar{Q}_2 Q_0) + Q_1 (M \cdot \bar{Q}_2 + \bar{Q}_0). \end{aligned}$$

$$\begin{aligned} Q_{0H} &= \bar{M} \cdot \bar{Q}_0 + M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1 = \\ &= \bar{Q}_0 (\bar{M} \cdot \bar{0} + M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1) + Q_0 (\bar{M} \cdot \bar{1} + M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1) = \\ &= \bar{Q}_0 (\bar{M} + M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1) + Q_0 (M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1) = \\ &= \bar{Q}_0 (\bar{M} + \bar{Q}_2 \bar{Q}_1 + Q_2 Q_1) + Q_0 (M \cdot \bar{Q}_2 \bar{Q}_1 + M \cdot Q_2 Q_1). \end{aligned}$$

Сравнивая преобразованные функции переходов Q_{2H} , Q_{1H} , Q_{0H} с характеристическим уравнением (4.5), находим функции возбуждения для входов J_i и K_i :

$$J_2 = \bar{M} \cdot Q_1 Q_0 + M \cdot Q_1 \bar{Q}_0 \quad (4.7)$$

$$K_2 = \frac{\bar{M} \cdot \bar{Q}_1 + \bar{M} \cdot \bar{Q}_0 + M \cdot Q_1 + M \cdot Q_0}{\bar{M} \cdot \bar{Q}_1 + \bar{M} \cdot \bar{Q}_0 + M \cdot Q_1 + M \cdot Q_0} \quad (4.8)$$

$$J_1 = \bar{M} \cdot Q_0 + \bar{Q}_2 Q_0 \quad (4.9)$$

$$K_1 = \frac{M \cdot \bar{Q}_2 + \bar{Q}_0}{M \cdot \bar{Q}_2 + \bar{Q}_0} \quad (4.10)$$

$$J_0 = \bar{M} + \bar{Q}_2 \bar{Q}_1 + Q_2 Q_1 \quad (4.11)$$

$$K_0 = \overline{M \cdot \overline{Q_2 Q_1}} + \overline{M \cdot Q_2 Q_1} \quad (4.12)$$

Для дальнейшего практического использования удобно, чтобы функции возбуждения входов K_i (4.8, 4.10, 4.12) были представлены в дизъюнктивной нормальной форме (ДНФ). Напомним, что ДНФ называется такая форма представления функции, при которой логическое выражение функции строится в виде дизъюнкции ряда членов, каждый из которых является простой конъюнкцией аргументов или их инверсией. Функции возбуждения входов J_i (4.7, 4.9, 4.11) уже представлены в ДНФ, поэтому преобразования не требуют. Для перехода от форм (4.8, 4.10, 4.12) к ДНФ вновь воспользуемся логическим преобразователем Electronics Workbench.

$$K_2 = \overline{\overline{M \cdot \overline{Q_1}} + \overline{M \cdot \overline{Q_0}} + \overline{M \cdot Q_1} + \overline{M \cdot Q_0}} = \overline{M \cdot Q_1 Q_0} + \overline{M \cdot \overline{Q_1} \overline{Q_0}}. \quad (4.13)$$

$$K_1 = \overline{M \cdot \overline{Q_2} + \overline{Q_0}} = \overline{M \cdot Q_0} + \overline{Q_2 Q_0}. \quad (4.14)$$

$$K_0 = \overline{M \cdot \overline{Q_2} \overline{Q_1} + \overline{M \cdot Q_2 Q_1}} = \overline{M} + \overline{Q_2 Q_1} + \overline{Q_2 \overline{Q_1}}. \quad (4.15)$$

По условию комбинационные цепи автомата должны быть реализованы в базисе И-НЕ. Функции возбуждения триггеров J_i и K_i (4.7, 4.9, 4.11, 4.13 – 4.15) преобразуем к заданному базису. Для этого воспользуемся правилом (см. лабораторную работу №1): для перехода к базису И-НЕ нужно дважды инвертировать все выражение, а затем применить теорему де Моргана $\overline{x + y} = \overline{x} \cdot \overline{y}$.

$$J_2 = \overline{\overline{M \cdot Q_1 Q_0} + \overline{M \cdot Q_1 \overline{Q_0}}} = \overline{\overline{M \cdot Q_1 Q_0} \cdot \overline{M \cdot Q_1 \overline{Q_0}}} \quad (4.16)$$

$$K_2 = \overline{\overline{M \cdot Q_1 Q_0} + \overline{M \cdot \overline{Q_1} \overline{Q_0}}} = \overline{\overline{M \cdot Q_1 Q_0} \cdot \overline{M \cdot \overline{Q_1} \overline{Q_0}}} \quad (4.17)$$

$$J_1 = \overline{\overline{M \cdot Q_0} + \overline{Q_2 Q_0}} = \overline{\overline{M \cdot Q_0} \cdot \overline{Q_2 Q_0}} \quad (4.18)$$

$$K_1 = \overline{\overline{M \cdot Q_0} + \overline{Q_2 Q_0}} = \overline{\overline{M \cdot Q_0} \cdot \overline{Q_2 Q_0}} \quad (4.19)$$

$$J_0 = \overline{\overline{M} + \overline{Q_2 \overline{Q_1}} + \overline{Q_2 Q_1}} = \overline{\overline{M} \cdot \overline{Q_2 \overline{Q_1}} \cdot \overline{Q_2 Q_1}} \quad (4.20)$$

$$K_0 = \overline{\overline{M} + \overline{Q_2 Q_1} + \overline{Q_2 \overline{Q_1}}} = \overline{\overline{M} \cdot \overline{Q_2 Q_1} \cdot \overline{Q_2 \overline{Q_1}}} \quad (4.21)$$

IV этап. Синтез автомата на JK-триггерах и элементах И-НЕ.

При выполнении этапа исследования использованы приемы №1, 2, 3, 4, 5, 6, 9 раздела «Типовые приемы работы в MicroCAP...».

По результатам выполнения предыдущих этапов исследования подготовлены все необходимые сведения для синтеза автомата. Сформулируем словесное описание. Основой цифрового автомата являются три JK-триггера – по количеству разрядов цифрового кода. К каждому из входов J_i , K_i присоединяется комбинационная цепь, реализованная в базисе И-НЕ согласно (4.16 – 4.21). Номер индекса i однозначно определяет взаимное соответствие функции возбуждения и входа триггера. Схема автомата должна быть дополнена вспомогательными цепями синхронизации и сброса.

По предложенному словесному описанию на рисунке 4.6 представлена структурная схема цифрового автомата, подготовленная в программе Micro-

САР. В схеме JK -триггер $X2$, предназначенный для старшего разряда, расположен слева, далее триггеры идут в порядке убывания старшинства разрядов. Элемент И-НЕ $U2$ исполняет роль инвертора для выработки сигнала \bar{M} . Чтобы избежать чрезмерной сложности в восприятии схемы, применен скрытый способ прокладки электрических проводников. Отрезки проводников, принадлежащие одной и той же цепи, имеют одинаковые названия.

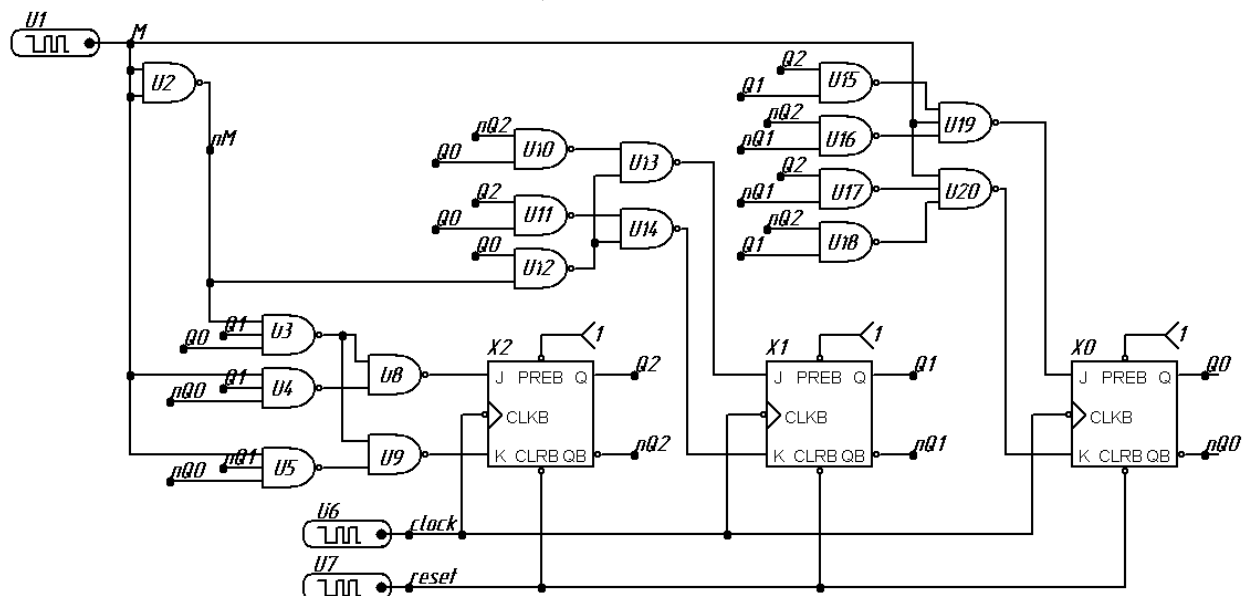


Рисунок 4.6 – Структурная схема цифрового автомата на JK -триггерах

Триггеры $X2$, $X1$, $X0$ – синхронные типа $JKFF$ с инверсным динамическим входом, рассмотренные в приеме №9.

Для цифровых источников $U1$, $U6$, $U7$ требуется задать параметры, определяющие поведение сигнала управления, синхросигнала и сигнала сброса, соответственно. Сначала определим длительность временного вида анализа. По условию частота импульсов синхронизации $f = 100$ кГц, значит период следования $T = 10$ мкс, причем в каждом периоде возникает новое состояние автомата. Согласно диаграмме (рисунок 4.5) имеем в общей сложности для двух режимов количество состояний $n = 16$. Общая длительность τ временного анализа:

$$\tau = n \cdot T = 16 \cdot 10 = 160 \text{ мкс.}$$

При исследовании цифрового автомата важно убедиться, что после восьмого (последнего) состояния в каждом режиме он возвращается в исходное положение. Например, по диаграмме (рисунок 4.5) в режиме $M = 0$ за состоянием 111 должно наступать состояние 000; в режиме $M = 1$ за состоянием 100 должно наступать состояние 000. Для контроля этих переходов необходимо добавить еще по одному периоду следования синхроимпульса в каждом режиме. Окончательная длительность τ_{OK} временного анализа:

$$\tau_{OK} = \tau + 2T = 160 + 2 \cdot 10 = 180 \text{ мкс.}$$

Установим параметры синхросигнала $Clock$. Пусть начало периода синхросигнала сопровождается нулевым уровнем, через половину периода $T/2$

= 5 мкс возникает передний фронт импульса и, как следствие, единичный уровень. Приход еще через 5 мкс следующего периода с нулевым уровнем сформирует отрицательный фронт импульса (рисунок 4.7).

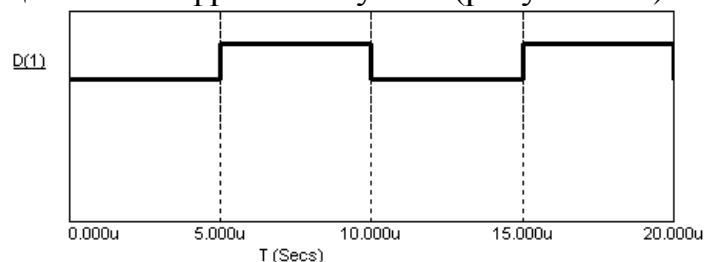


Рисунок 4.7 – Временная диаграмма синхросигнала

В диалоговом окне свойств источника *U6* в строке **FORMAT** указывают значение **1**. При выборе строки **COMMAND** задают форму синхросигнала в нижней части диалогового окна:

```
.DEFINE CLOCK
+ 0us 0
+ LABEL=begin
+ +5us INCR BY 1
+ +5us GOTO begin -1 TIMES
```

Установим параметры сигнала управления *M*. Будем исходить из предположения, что первую половину временного анализа будет занимать режим двоичного счетчика ($M = 0$); вторую половину – режим счетчика в коде Грея ($M = 1$, рисунок 4.8).

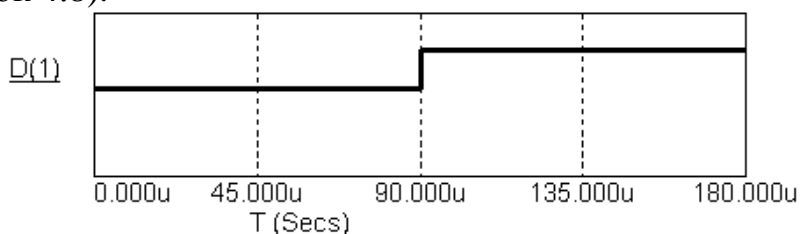


Рисунок 4.8 – Временная диаграмма сигнала управления

В диалоговом окне свойств источника *U1* в строке **FORMAT** указывают значение **1**. При выборе строки **COMMAND** задают форму сигнала управления:

```
.DEFINE M
+ 0us 0
+ 90us 1
```

Установим параметры сигнала сброса *Reset*. Сигнал сброса должен представлять собой строб-импульс, появляющийся в начальные моменты действия того или иного режима работы цифрового автомата. Учитывая инверсный тип входа *CLRB*, строб-импульс сброса имеет нулевой активный уровень и единичный пассивный. Длительность строб-импульса можно принять как десятую часть периода следования синхросигнала $T/10 = 1$ мкс; момент возникновения фронта строб-импульса примем как +1 мкс от начала ре-

жима работы. Цифровой автомат, имеющий два режима работы, дважды сбрасывается сигналом *Reset* в моменты $t_1 = 1$ мкс и $t_2 = \frac{\tau_{OK}}{2} + 1 = 91$ мкс (рисунок 4.9).

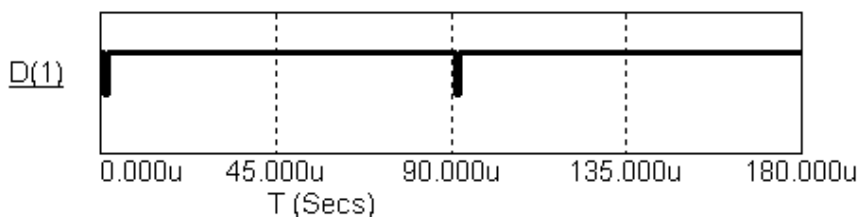


Рисунок 4.9 – Временная диаграмма сигнала сброса

В диалоговом окне свойств источника *U7* в строке **FORMAT** указывают значение **1**. При выборе строки **COMMAND** задают форму сигнала сброса:

.DEFINE RESET

+ **0us 1**

+ **1us 0**

+ **2us 1**

+ **91us 0**

+ **92us 1**

В таблицу диалогового окна **Transient Analysis Limits** заносится следующая информация:

P	X	Y
	EXPRESSION	EXPRESSION
1	T	D(M)
1	T	D(CLOCK)
1	T	D(RESET)
1	T	D(Q0)
1	T	D(Q1)
1	T	D(Q2)
1	T	BIN(Q2,Q1,Q0)

В последней строке таблицы записано двоичное представление трехразрядной шины данных, составленной из проводников *Q2*, *Q1*, *Q0*.

Результат исследования трехразрядного двухрежимного цифрового автомата представлен в виде временной диаграммы на рисунке 4.10. Сравнивая последовательность состояний трехразрядной шины *bin(Q2,Q1,Q0)* с диаграммой на рисунке 4.5, можно убедиться в адекватности проведенного исследования.

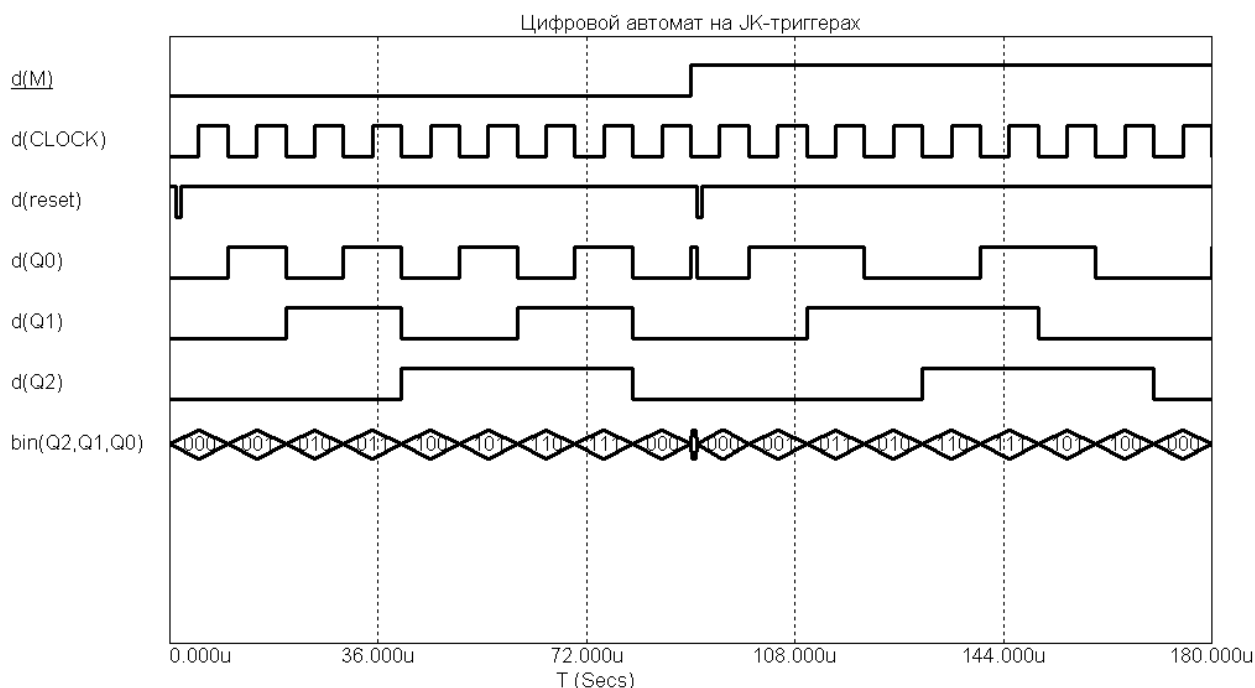


Рисунок 4.10 – Временная диаграмма работы двухрежимного трехразрядного автомата

4.5 Лабораторное задание

Требуется спроектировать трехразрядный автомат на основе *JK*-триггеров с двумя режимами работы, управляемый входным сигналом M . При $M = 0$ автомат должен работать как двоичный счетчик с модулем счета 8, а при $M = 1$ – согласно приведенной в варианте задания последовательности состояний. Комбинационные цепи автомата реализовать в базисе И-НЕ. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенного схемотехнического решения подтвердить результатами моделирования в программе MicroCAP.

4.6 Контрольные вопросы

1. Что такое последовательностные схемы?
2. В чем основное отличие автоматов с памятью от комбинационных цепей?
3. В чем заключается отличие асинхронных и синхронных автоматов с памятью?
4. Для чего служат тактирующие импульсы в синхронных автоматах с памятью?

Продолжение на следующей странице

5. Каков основной недостаток асинхронных автоматов в плане их практического использования?

6. Что такое автоматы Мили и автоматы Мура?

7. Какие существуют способы кодирования состояний автоматов?

8. На основе каких типов триггеров наиболее часто реализуют цифровые автоматы?

9. Как определить число триггеров, необходимых для синтеза цифрового автомата, если N – число состояний автомата?

4.7 Варианты заданий

№ ВАРИА НТА	ПОСЛЕДОВАТЕЛЬНОСТЬ СОСТОЯНИЙ ПРИ $M = 1$
1	000 → 010 → 011 → 111 → 101 → 001 → 110 → 100 → 000
2	000 → 011 → 100 → 110 → 001 → 111 → 101 → 010 → 000
3	000 → 100 → 101 → 011 → 111 → 001 → 110 → 010 → 000
4	000 → 101 → 110 → 100 → 001 → 111 → 010 → 011 → 000
5	000 → 110 → 111 → 101 → 100 → 001 → 010 → 011 → 000
6	000 → 111 → 001 → 011 → 010 → 100 → 110 → 101 → 000
7	000 → 001 → 010 → 111 → 011 → 110 → 100 → 101 → 000
8	000 → 010 → 100 → 001 → 111 → 110 → 011 → 101 → 000
9	000 → 011 → 101 → 100 → 001 → 010 → 111 → 110 → 000
10	000 → 100 → 110 → 111 → 101 → 001 → 010 → 011 → 000
11	000 → 101 → 111 → 011 → 110 → 001 → 010 → 100 → 000
12	000 → 110 → 011 → 101 → 111 → 001 → 100 → 010 → 000
13	000 → 111 → 010 → 100 → 001 → 101 → 011 → 110 → 000
14	000 → 001 → 100 → 111 → 110 → 101 → 010 → 011 → 000
15	000 → 010 → 101 → 011 → 001 → 100 → 111 → 110 → 000
16	000 → 011 → 110 → 100 → 101 → 111 → 001 → 010 → 000
17	000 → 100 → 111 → 011 → 001 → 010 → 101 → 110 → 000
18	000 → 101 → 001 → 111 → 010 → 100 → 011 → 110 → 000
19	000 → 110 → 011 → 001 → 010 → 100 → 101 → 111 → 000
20	000 → 111 → 100 → 001 → 010 → 011 → 101 → 110 → 000
21	000 → 001 → 111 → 101 → 011 → 100 → 110 → 010 → 000
22	000 → 010 → 001 → 111 → 101 → 110 → 011 → 100 → 000
23	000 → 011 → 001 → 100 → 111 → 010 → 110 → 101 → 000
24	000 → 100 → 001 → 101 → 011 → 010 → 110 → 111 → 000
25	000 → 101 → 100 → 010 → 001 → 011 → 110 → 111 → 000

5 ЛАБОРАТОРНАЯ РАБОТА №5 – АЛЬТЕРНАТИВНЫЕ СПОСОБЫ ПРОЕКТИРОВАНИЯ АВТОМАТОВ С ПАМЯТЬЮ

5.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

- 1) знакомство с методом синтеза перестраиваемых автоматов с памятью на основе мультиплексного управления;
- 2) знакомство с методом синтеза автоматов с памятью, в которых используется кодирование состояний кодами «1 из N »;
- 3) изучение работы сдвигающих регистров;
- 4) изучение работы шифраторов, преобразующих унитарный код в двоичные, двоично-десятичные коды, коды Грея.

5.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 5.5.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

5.3 Автоматы с памятью на основе мультиплексного управления

Математическая модель. Рассмотрим простейший способ реализации перестраиваемого автомата с памятью. Пусть задано множество автоматных графов. Каждый из графов характеризуется числом состояний (вершин), числом входных переменных x_1, \dots, x_n и числом выходных переменных U_1, \dots, U_p . Пусть максимальное число состояний среди всех графов равно R , максимальное число входных переменных равно n , максимальное число выходных переменных равно p . Построим $(m + p)$ универсальных логических модулей (УЛМ), каждый из которых реализует все функции от $(n + m)$ переменных, где $m = \log_2 R$. Соединим их так, как показано на рисунке 5.1. На этом построение перестраиваемого автомата с памятью закончено. Двойные линии на рисунке 5.1 означают, что все входы x_1, \dots, x_n и выходы Q_1, \dots, Q_m соединены с

адресными входами каждого УЛМ. Сигналы z_1, \dots, z_k – это настроечные переменные, заменяемые при настройке функциями $f_1(x_1, \dots, x_n), \dots, f_k(x_1, \dots, x_n)$, где k – число различных функций, используемых для настройки.

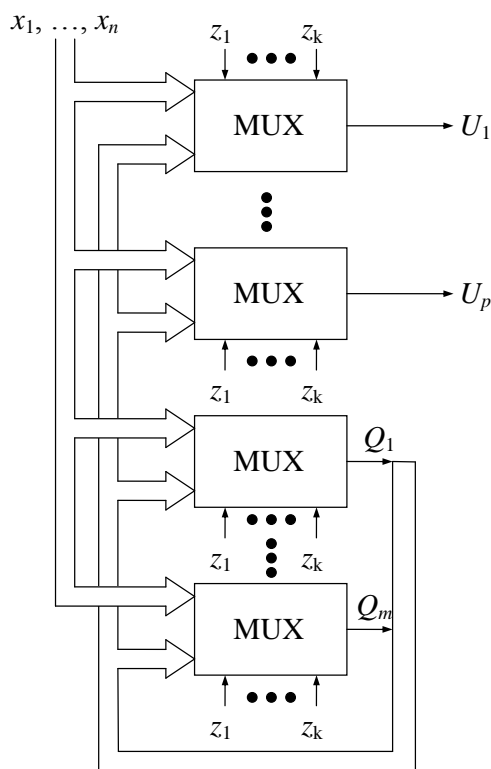


Рисунок 5.1 – Блок-схема перестраиваемого автомата с памятью (элементы памяти условно не показаны)

Построенный перестраиваемый автомат может реализовать любой автоматный граф, имеющий не более чем R состояний, n входных переменных, p выходных переменных. Действительно, пусть задан некоторый автоматный граф, удовлетворяющий этому условию. Обозначим его входной алфавит (x_1, \dots, x_n) , выходной алфавит (U_1, \dots, U_p) . Закодируем вершины графа значениями дополнительных переменных (q_1, \dots, q_m) , где $m = \log_2 R$. В качестве элементов памяти выберем D -триггеры. Тогда по графу переходов с закодированными вершинами можно построить систему функций возбуждения $\{Q\}$ и выходов $\{U\}$:

$$\begin{aligned}
 Q_1 &= Q_1(x_1, \dots, x_n, q_1, \dots, q_m), \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 Q_m &= Q_m(x_1, \dots, x_n, q_1, \dots, q_m), \\
 U_1 &= U_1(x_1, \dots, x_n, q_1, \dots, q_m), \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 U_p &= U_p(x_1, \dots, x_n, q_1, \dots, q_m).
 \end{aligned}$$

Эта система содержит $(m + p)$ функций, и каждая из них может быть реализована с помощью одного УЛМ от $(m + n)$ переменных. Перестраиваемый автомат, показанный на рисунке 5.1, состоит из $(m + p)$ УЛМ, каждый из которых реализует любую функцию от $(m + n)$ переменных. Эти УЛМ в совокупности могут реализовать любую систему из m функций переходов и p функций выходов. Таким образом, существует простой способ аппаратной реализации перестраиваемого автомата с памятью.

Аппаратная модель. Структура с мультиплексорами на входах триггеров отличается концептуальной простотой и наглядностью, для ее проектирования не требуется разработка логических преобразователей, обеспечивающих необходимые переходы автомата. Задача решается, в сущности, табличным методом. Переменные состояния, снимаемые с триггеров, и входные сигналы образуют слово, служащее для мультиплексора адресным входом. По этому адресу в каждом мультиплексоре выбирается переменная (0 или 1), необходимая для перевода D -триггера в новое состояние. Ясно, что при этом данные для информационных входов мультиплексоров берутся прямо из таблицы переходов ($D_i = Q_{iH}$).

Достоинство структуры – легкость перестройки автомата на новый алгоритм работы, недостаток – быстрый рост размерности мультиплексоров с ростом числа состояний и входов автомата. Структура с мультиплексорным управлением триггерами показана на рисунке 5.2. Входные сигналы x_1, \dots, x_n и значения разрядов слова старого состояния Q_1, \dots, Q_m образуют управляющее (адресное) входное слово мультиплексора, по которому выбираются значения разрядов нового слова состояния. Поступление тактового импульса вводит новое слово состояния в триггеры.

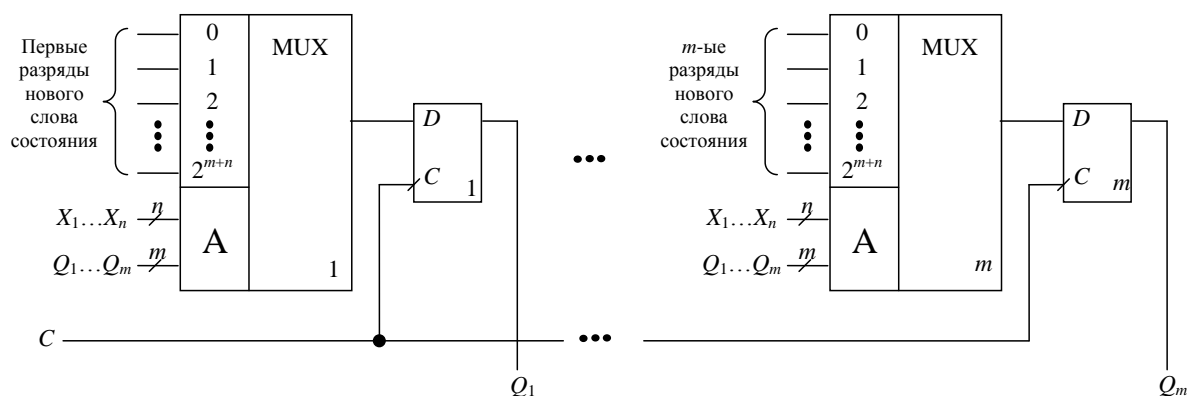


Рисунок 5.2 – Структура автомата на триггерах с мультиплексным управлением

Рассмотрим простой пример. Пусть цифровой автомат имеет три состояния, которые можно закодировать как 00, 01, 10. Последовательность наступления таких состояний циклическая и отражена автоматным графом на рисунке 5.3.

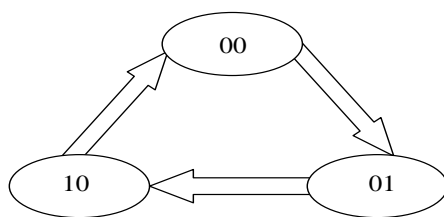


Рисунок 5.3 – Граф переходов цифрового автомата

Условимся, что младшие разряды кодовых состояний будут обозначены Q_0 , старшие разряды – Q_1 . Новые состояния каждого перехода будут обозначены дополнительным индексом «н»: $Q_{0н}$, $Q_{1н}$. Рассматривая каждый переход по отдельности (его новое и старое состояния), можно составить таблицу истинности (таблица 5.1).

Таблица 5.1 – Таблица истинности цифрового автомата

Q_1	Q_0	$Q_{1н}$	$Q_{0н}$
0	0	0	1
0	1	1	0
1	0	0	0

Таблица истинности 5.1 дает исчерпывающую информацию для синтеза цифрового автомата на основе мультиплексного управления. Количество разрядов для обозначения кодовых состояний автомата $m = 2$, значит в состав устройства должны входить два мультиплексора и два элемента памяти (рисунок 5.4). В качестве элементов памяти выбраны D -триггеры.

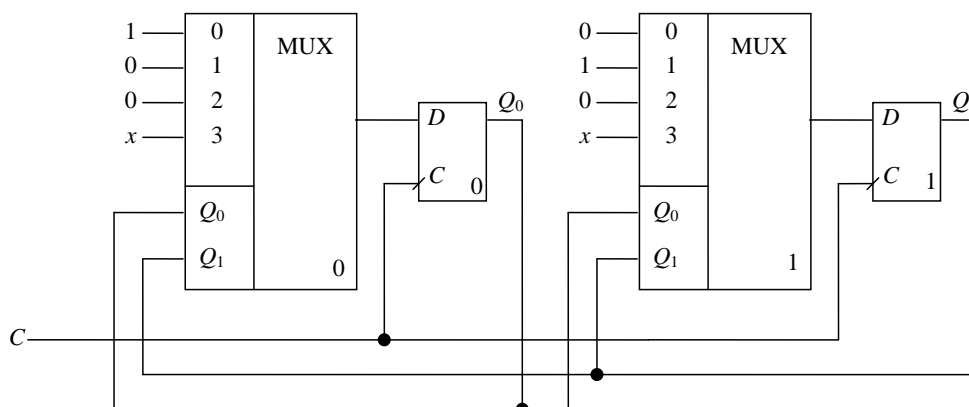


Рисунок 5.4 – Логическая схема цифрового автомата с мультиплексным управлением

Адресная шина мультиплексоров состоит всего из двух разрядов – Q_1Q_0 . Линии $X_1 \dots X_n$, изображенные в шине адреса на рисунке 5.2, в данном случае отсутствуют, поскольку не требуется перестраивать автомат на другой алгоритм работы. Если количество адресных линий мультиплексоров $m = 2$, то количество информационных входов равно $2^m = 4$. Легко заметить, что мультиплексоры нулевого и первого разряда имеют на информационных вхо-

дах столбцы констант, совпадающие с соответствующими столбцами Q_{0H} и Q_{1H} таблицы истинности. Так как граф переходов содержит только три состояния, то четвертый по счету информационный вход мультиплексоров не используется. На этих входах неопределенное (любое) значение x . Переход от одного состояния к другому осуществляется с помощью тактирующих импульсов C . Предполагается, что в начальный момент времени сигнал сброса (на схеме не показан) устанавливает D -триггеры в нулевое состояние, так что $Q_0 = 0$; $Q_1 = 0$.

5.4 Автоматы с памятью, имеющие кодирование состояний типа «1 из N »

Цифровые автоматы с памятью, имеющие структуру кодирования состояний типа «1 из N », состоят из трех функциональных узлов: сдвигающего регистра, шифраторов перевода исходного (1 из N) кода в требуемый код и мультиплексора (рисунок 5.5). Требуемый код может быть двоичным, двоично-десятичным, кодом Грея и т.д. в зависимости от поставленной задачи. В общем случае унитарный код «1 из N », а точнее, как следует из представленного рисунка, «1 из 2^n », может быть подвергнут 2^k различным кодовым преобразованиям. Для этой цели вводятся 2^k шифраторов перевода исходного кода в требуемый – F_1, F_2, \dots, F_{2^k} . По общей шине к каждому шифратору поступает исходный код «1 из 2^n » от сдвигающего регистра. Сделаем допущение о том, что, несмотря на разнообразие кодовых преобразований в шифраторах, разрядность выходного кода неизменна и равна n . Выбор необходимого в текущий момент времени кода $a_n a_{n-1} \dots a_1$ из множества доступных происходит с помощью мультиплексора. Управляющее (адресное) слово $x_k x_{k-1} \dots x_1$ связывает выход мультиплексора с одним из информационных каналов F_1, F_2, \dots, F_{2^k} на входе.

Рассмотрим более подробно первые два функциональных узла – сдвигающий регистр и шифратор.

Параллельный (сдвигающий) регистр является, как правило, универсальным и может выполнять все доступные для регистров микрооперации. Для этого разрядные схемы, входящие в его состав, соединены между собой. На рисунке 5.6 показан цифровой автомат, который состоит из m последовательно соединенных D -триггеров, функции возбуждения которых имеют вид:

$$D_1 = x, D_r = Q_{r-1}, r = 2, 3, \dots, m. \quad (5.1)$$

Из соотношения (5.1) вытекает, что информация, которая сохраняется в некотором такте в триггере Q_{r-1} , передается в следующем такте в триггер Q_r , т.е. происходит сдвиг информации от триггера к триггеру. Такие автоматы называются регистрами сдвига и используются для сдвига m -разрядных чисел в одном направлении. Значение входного сигнала x , которое отвечает некоторому такту, появляется на выходе регистра сдвига Q_m через m тактов.

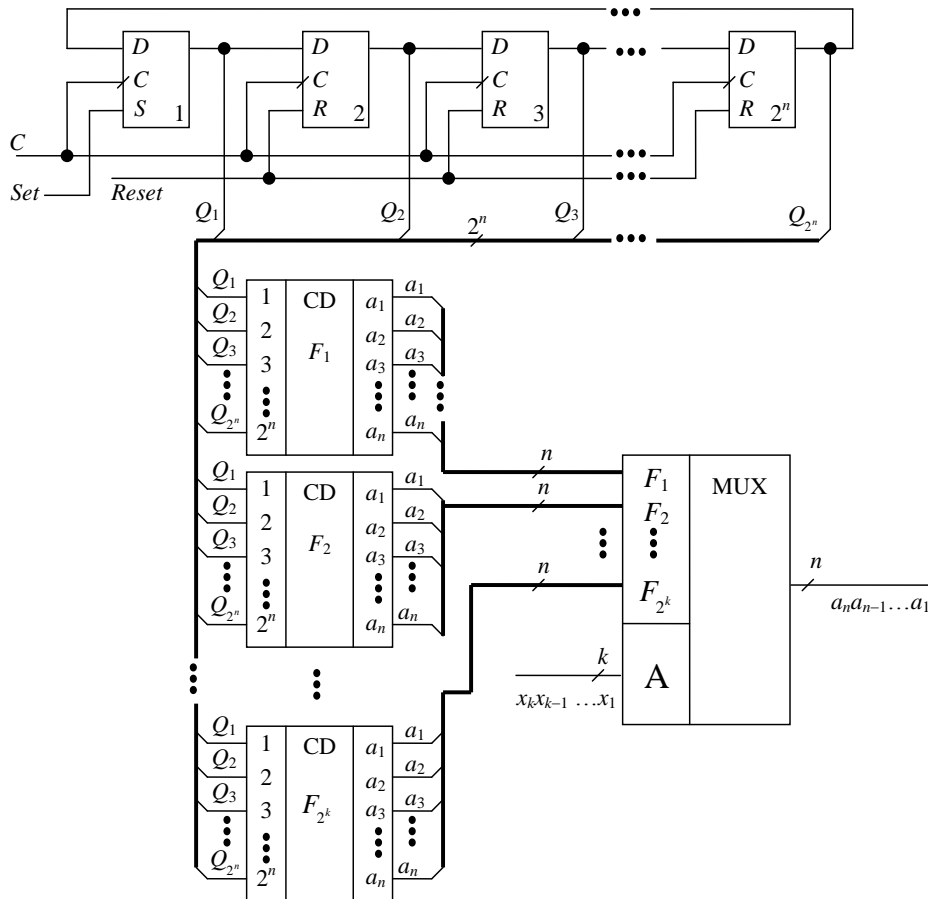


Рисунок 5.5 – Логическая схема автомата с памятью, имеющего кодирование состояний «1 из N »

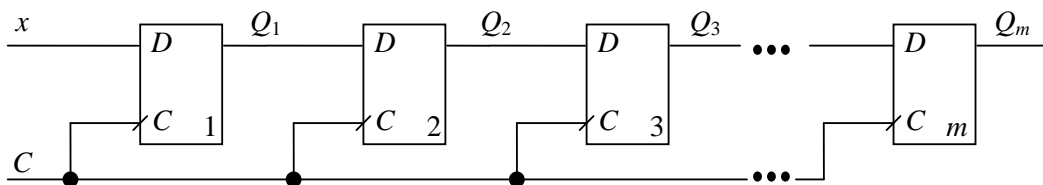


Рисунок 5.6 – Регистр сдвига

Если Q_m – старший разряд, то имеет место сдвиг в сторону старших разрядов или влево. Если же Q_m считать младшим разрядом, то будет иметь место сдвиг в сторону младших разрядов, или вправо. **Заметим, что направление сдвига в регистрах (влево или вправо) определяется не по расположению триггеров на схеме, а исходя из того, что в записи цифрового слова старшие разряды располагаются слева, а младшие справа.** Кроме основного назначения (сдвиг чисел) регистры сдвига используются и для сдвига нечисловой информации (например, при построении на них счетчиков).

Разновидностью сдвигающего регистра, применяемого в структуре кодирования «1 из N » (рисунок 5.5), является *кольцевой регистр*. Записанное в регистр слово содержит всего один активный уровень сигнала. При сдвигах

активный уровень перемещается с одного выхода на другой, циркулируя в кольце. Число выходов кольцевого регистра равно количеству возможных состояний автомата с памятью. Недостаток кольцевого регистра – потеря правильного функционирования при сбое. Если в силу каких-либо причин слово в регистре исказится, то возникшая ошибка станет постоянной. Схема кольцевого регистра не обладает свойством автозапуска.

Специфическая ситуация может возникнуть при установке исходного состояния автомата. Если набор триггеров выполнен как единая ИС, то сброс сигналом *Reset* переведет *все* триггеры в нулевое (пассивное) состояние, тогда как первый слева (по рисунку) триггер должен получать единичное (активное) состояние. Для создания эквивалента нужной ситуации можно взять выход левого триггера с инверсного вывода, что после сброса даст на выходах регистра состояние $100\dots 0$. Чтобы не изменилось функционирование схемы, на вход левого триггера также следует подавать инвертированный сигнал (рисунок 5.7).

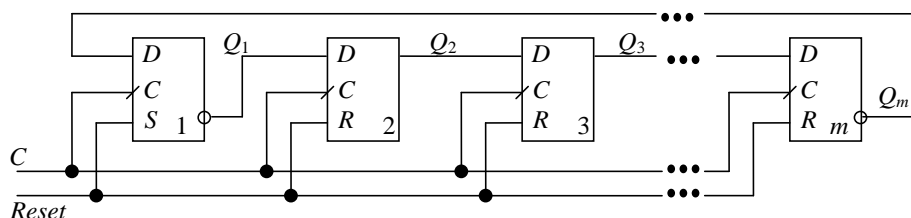


Рисунок 5.7 – Схема установки автомата в исходное состояние при использовании кода «1 из N»

Среди шифраторов наиболее распространены *двоичные шифраторы*, выполняющие перевод унитарного кода «1 из N » в двоичный код, т.е. реализуют микрооперацию, обратную микрооперации двоичных дешифраторов. Входам шифратора последовательно присваиваются значения десятичных чисел, поэтому подача активного логического сигнала на один из его входов воспринимается шифратором как подача соответствующего десятичного числа. Этот сигнал преобразуется на выходе шифратора в двоичный код. Согласно сказанному, если шифратор имеет n выходов, число его входов должно быть не более чем 2^n . Шифратор, имеющий 2^n входов и n выходов, называется *полным*. Если число входов шифратора меньше 2^n , он называется *неполным*.

Поясним работу шифратора на примере преобразователя десятичных чисел от 0 до 9 в двоично-десятичный код. Таблица истинности, соответствующая этому случаю, имеет вид представленный ниже (таблица 5.2).

Так как число входов данного устройства меньше $2^n = 16$, имеем неполный шифратор. Используя таблицу для Q_3, Q_2, Q_1, Q_0 , можно записать следующие выражения:

$$\begin{aligned} Q_3 &= x_8 + x_9, \\ Q_2 &= x_4 + x_5 + x_6 + x_7, \\ Q_1 &= x_2 + x_3 + x_6 + x_7, \\ Q_0 &= x_1 + x_3 + x_5 + x_7 + x_9. \end{aligned} \quad (5.2)$$

Полученная система характеризует работу шифратора. Логическая схема устройства, соответствующая системе (5.2), приведена на рисунке 5.8.

Таблица 5.2 – Таблица истинности шифратора

X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

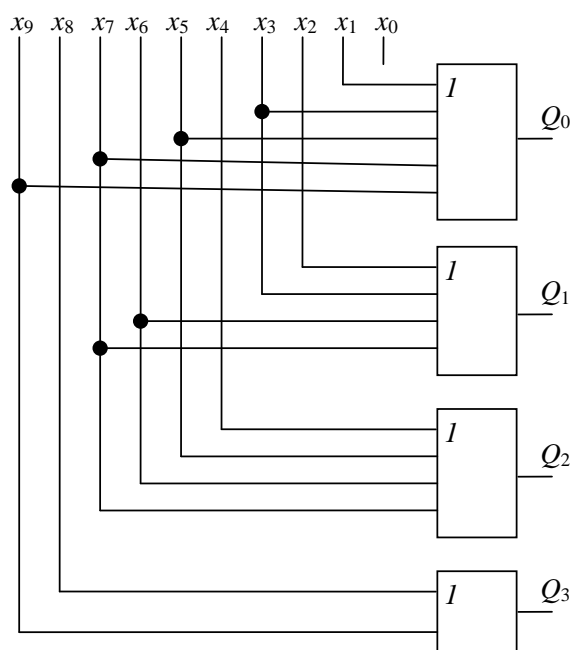


Рисунок 5.8 – Логическая схема шифратора десятичных чисел

Нетрудно заметить, что в шифраторе рассматриваемого типа сигнал, подаваемый на вход x_0 , не используется. Поэтому отсутствие сигнала на любом из входов x_1, \dots, x_9 трактуется схемой как наличие на входе нулевого сигнала.

Структура с кодированием типа «1 из N » содержит максимальное число триггеров, т.к. в ней для каждого состояния предусматривается специальный триггер, находящийся в активном состоянии (пусть это будет состояние 1) при пассивном состоянии всех остальных. Рост числа триггеров усложняет автомат, но, одновременно с этим, резко упрощается логика, обеспечивающая

переходы автомата. Поэтому сложность автомата в целом может оказаться приемлемой.

Кодирование состояний автомата кодом «1 из N » (в английском языке ONE, One Hot Encoding) рекомендуется для ряда современных СБИС программируемой логики, т.к. дает простой метод построения автомата высокого быстродействия, имеющего во входных цепях триггеров мало уровней логики. Метод ONE устраняет много логических схем, требуемых для декодирования состояний. Набор триггеров образует структуру типа сдвигающего регистра – узла, допускающего эффективное размещение и трассировку в топологии СБИС.

5.5 Примеры проектирования трехразрядного цифрового автомата

Требуется спроектировать цифровой автомат двумя способами: на основе мультиплексного управления и на основе кодирования состояний автомата кодами «1 из N ». Трехразрядный автомат должен иметь два режима работы (см. лабораторную работу №4). Если управляющий сигнал $M = 0$, то автомат работает как двоичный счетчик с модулем счета 8, при $M = 1$ автомат работает в коде Грея. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

I этап. Синтез цифрового автомата на основе мультиплексного управления.

При выполнении этапа исследования использованы приемы №1-7, 10 раздела «Типовые приемы работы в MicroCAP...».

Диаграмма состояний двухрежимного цифрового автомата (рисунок 5.9), а также таблица истинности (таблица 5.3) взяты нами без изменений из методического примера лабораторной работы №4.

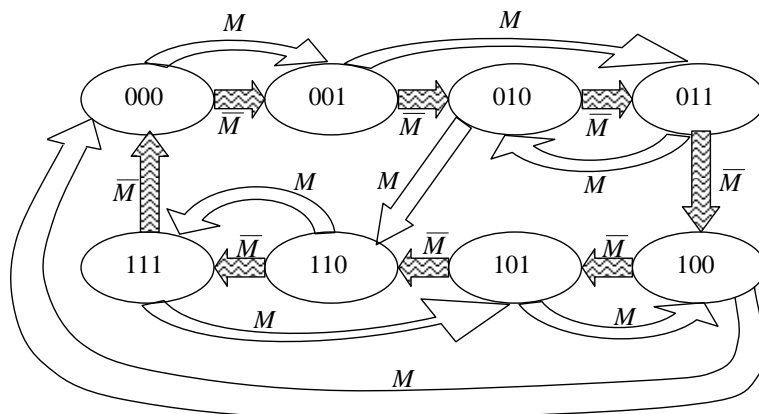


Рисунок 5.9 – Диаграмма состояний двухрежимного трехразрядного автомата

Таблица 5.3 – Таблица истинности двухрежимного цифрового автомата

ВХОДНОЙ УПРАВЛЯЮЩИЙ СИГНАЛ	ИСХОДНОЕ СОСТОЯНИЕ			НОВОЕ СОСТОЯНИЕ		
	Q_2	Q_1	Q_0	Q_{2H}	Q_{1H}	Q_{0H}
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	1	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	1	1
1	1	1	1	1	0	1

Конкретная реализация автомата для рассматриваемого примера (рисунок 5.10) не требует особых пояснений. Заметим лишь, что мультиплексор старшего разряда на этом рисунке расположен слева. Столбцы цифровых констант на информационных входах мультиплексоров берутся непосредственно из таблицы истинности 5.3 (Q_{2H} , Q_{1H} , Q_{0H}). При нулевых исходных состояниях триггеров и управляющем сигнале $M = 0$ на адресные входы мультиплексоров поступает код 0000 и на входах триггеров формируется комбинация сигналов 001. Поступление тактового импульса вводит эту комбинацию в триггеры. Теперь адресом для мультиплексоров становится комбинация 0001, по которой с них снимается комбинация 010, поступающая по разрешению следующего тактового импульса в триггеры. Так реализуется режим двоичного счетчика.

Изменение управляющего сигнала M дает смену режима работы автомата. Если, например, при слове состояния 010 сигнал становится единичным, то адрес мультиплексоров изменяется с 0010 на 1010 и с их выводов снимается комбинация 110, соответствующая следующему состоянию при работе счетчика в коде Грея.

Структура и работа автомата отличаются большой наглядностью, переход к другому алгоритму функционирования требует только смены сигналов на информационных входах мультиплексоров.

Представленная на рисунке 5.10 схема цифрового автомата требует некоторой доработки в плане подготовки ее для схмотехнического моделирования в программе MicroCAP. Для этого рассмотрим логическую схему на рисунке 5.11, которая несколько отличается от исходной.

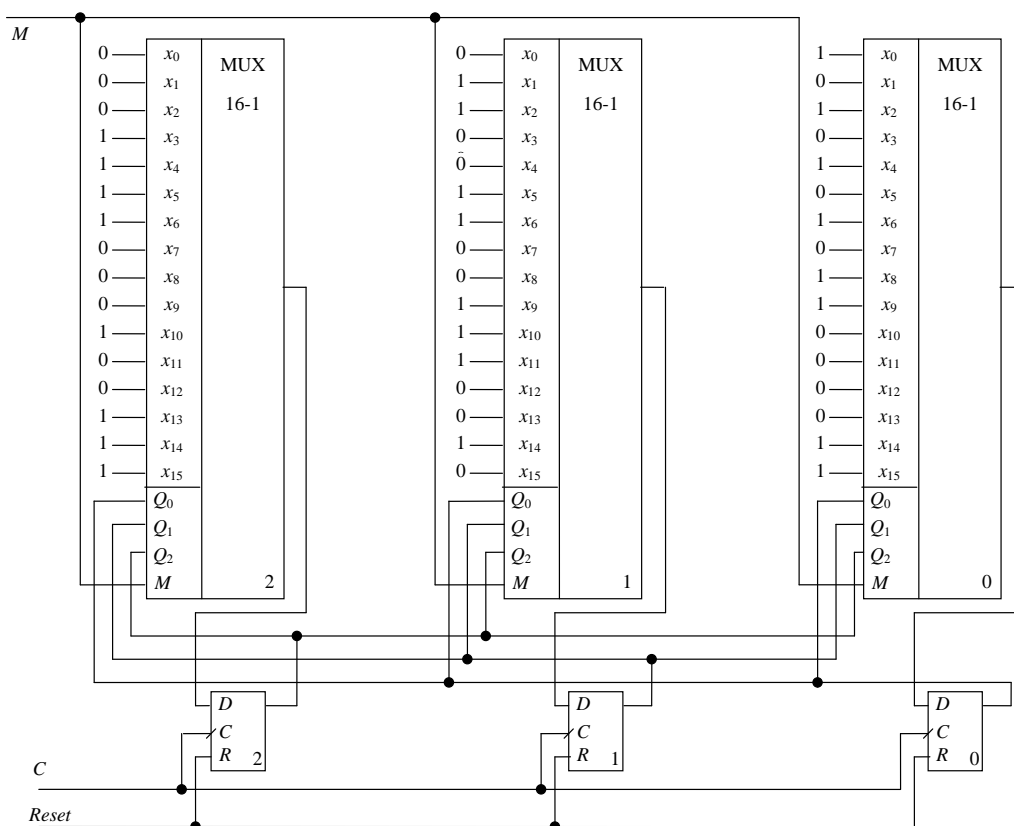


Рисунок 5.10 – Логическая схема автомата с мультиплексорами на входах триггеров

Основой схемы являются три 16-входовых мультиплексора 74150 (отечественный аналог К155КП1), которые размещаются на поле чертежа по команде *Component/Digital Library/74xx120/148-/74150*. Устанавливать какие-либо параметры в диалоговом окне свойств мультиплексора не требуется. Условное графическое изображение мультиплексора 74150 включает следующие выводы:

- E0...E15 – информационные входы;
- A, B, C, D – адресные входы; причем вход A является младшим разрядом;
- GBAR – инверсный вход разрешения работы мультиплексора;
- W – инверсный выход мультиплексора.

Учитывая, что мультиплексоры оснащены инверсными выходами, в схему введены также инверторы U5, U7, U9. Элементы памяти цифрового автомата – это D-триггеры U4, U6, U8 типа DFF, рассмотренные в приеме №10

Значения цифровых констант на схеме устанавливаются с помощью элементов Fixed Digital. Чтобы не затруднять чтение и понимание схемы,

применен способ скрытой прокладки проводников между выходами *D*-триггеров и адресными входами мультиплексов.

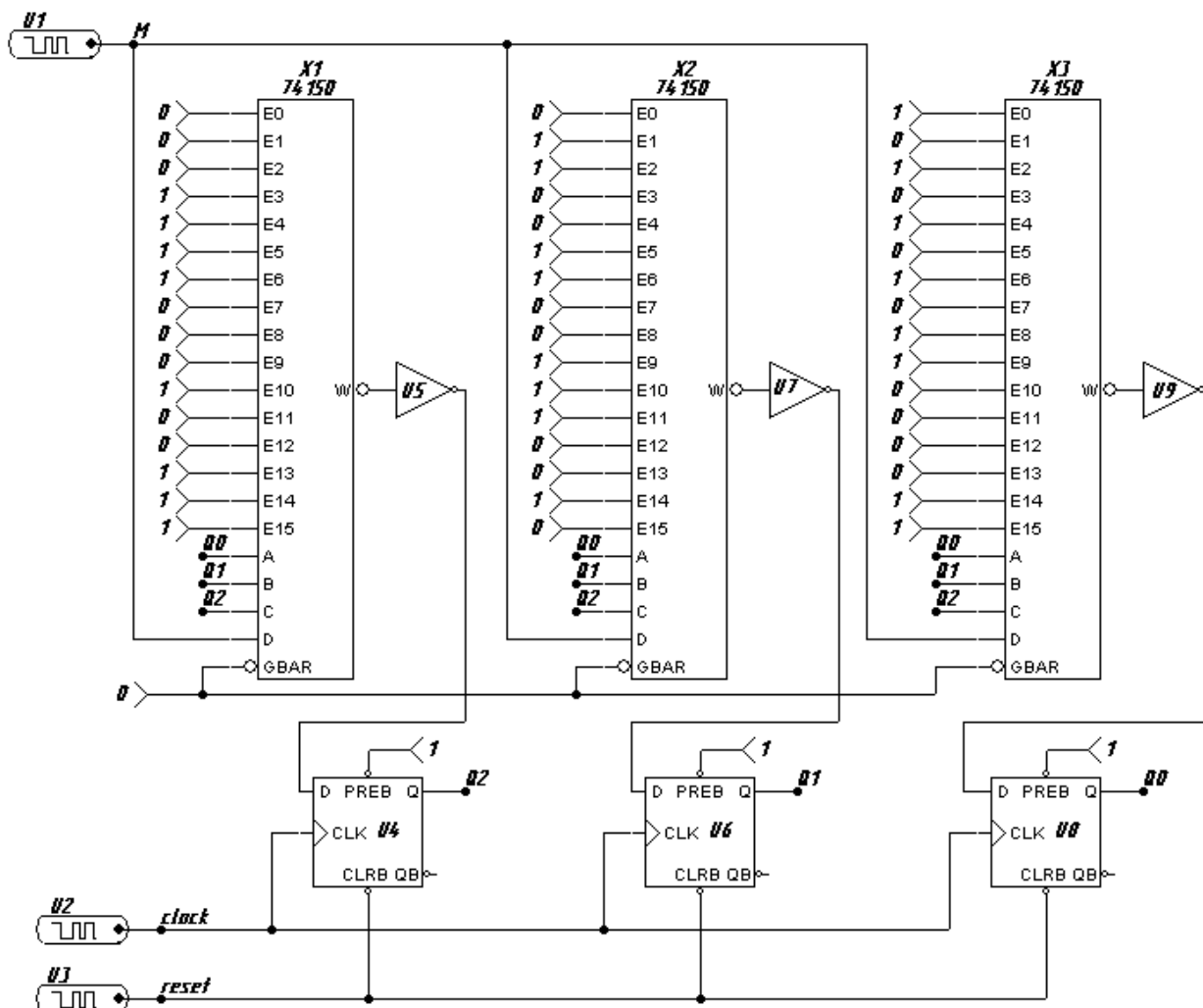


Рисунок 5.11 – Логическая схема автомата с мультиплексным управлением, подготовленная в программе MicroCAP

Применяемые в схеме *D*-триггеры активируются *передним* фронтом синхроимпульсов. Если оставить без изменений (относительно прошлой работы) параметры входных сигналов *M*, *Clock*, *Reset*, то возникнет ситуация, когда на каждом из временных интервалов в 90 мкс будут возникать вместо ожидаемых девяти 10 состояний автомата. Наиболее простой способ устранения такой ситуации – инвертирование формы синхросигнала *Clock* (рисунок 5.12). Для этого во второй строке листинга с описанием параметров синхросигнала следует изменить константу 0 на константу 1:

```
.DEFINE CLOCK
+ 0us 1
+ LABEL begin
+ +5us INCR BY 1
+ +5us GOTO begin -1 TIMES
```

Особенности установки параметров для сигналов управления M и сброса $Reset$ подробно рассмотрены в лабораторной работе №4; в настоящей работе они остаются без изменений. Общая длительность временного вида анализа составляет 180 мкс.

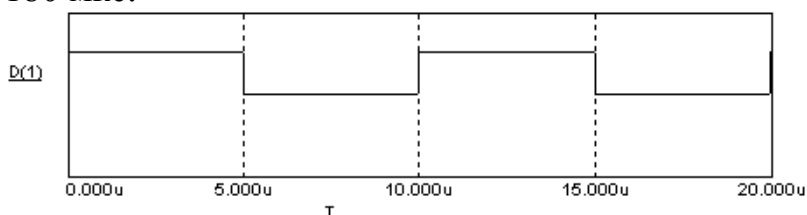


Рисунок 5.12 – Временная диаграмма синхросигнала

Моделирование схемы во временной области, а также анализ полученных результатов, не содержат принципиальных отличий от аналогичных действий в методическом примере лабораторной работы №4. Сравнивая последовательность состояний трехразрядной шины $bin(Q2, Q1, Q0)$ на рисунке 5.13 с результатами предыдущей работы, можно убедиться в адекватности проведенного исследования.

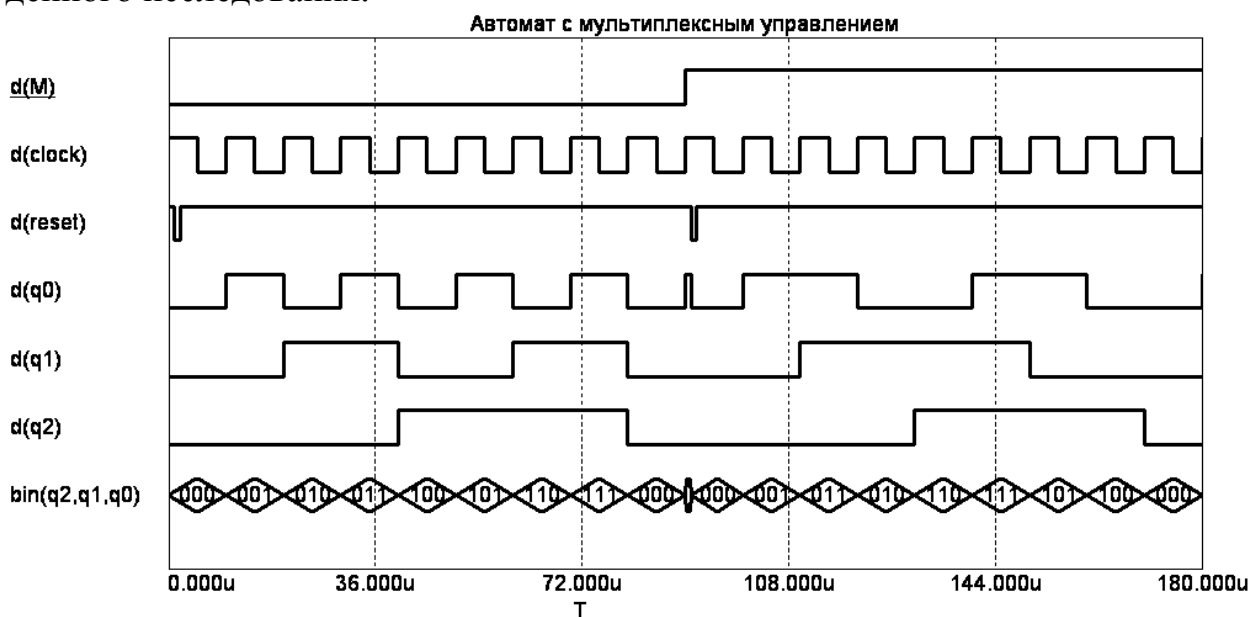


Рисунок 5.13 – Временная диаграмма работы автомата с мультиплексным управлением

II этап. Синтез цифрового автомата на основе кодирования состояний кодами «1 из N».

При выполнении этапа исследования использованы приемы №1-6, 10 раздела «Типовые приемы работы в MicroCAP...».

Для рассматриваемого примера автомат реализуется структурой (рисунок 5.14) с числом триггеров 8. Переход в следующее состояние происходит как переход единственной единицы из триггера в соседний триггер, что осуществляется сдвигающим регистром. В структуре цифрового автомата присутствуют шифраторы для перевода кода «1 из N» в двоичный код, либо в код

0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Таблица 5.5 – Таблица истинности шифратора кода Грея

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	1	0	1
1	0	0	0	0	0	0	0	1	0	0

Анализируя таблицы, легко заметить, что следование каноническому правилу записи логической функции в виде СДНФ приведет к неоправданно сложному выражению для разрядов Q_2 , Q_1 , Q_0 . Поэтому в данном случае, когда в левой части таблицы $x_7 \dots x_0$ представлен код «1 из N » для записи логических функций применяется более простое правило. Например, по таблице 5.4 находим, что в двоичном разряде Q_2 присутствует единица, когда в коде «1 из N » активными являются десятичные числа 7 ИЛИ 6 ИЛИ 5 ИЛИ 4. Переходя к выражениям булевой алгебры, имеем:

$$Q_2 = x_7 + x_6 + x_5 + x_4.$$

Полная система уравнений для случаев $M = 0$ (двоичный код) и $M = 1$ (код Грея) имеет вид:

$$M = 0 \Rightarrow \begin{cases} Q_2 = x_7 + x_6 + x_5 + x_4; \\ Q_1 = x_7 + x_6 + x_3 + x_2; \\ Q_0 = x_7 + x_5 + x_3 + x_1, \end{cases}$$

$$M = 1 \Rightarrow \begin{cases} Q_2 = x_7 + x_6 + x_5 + x_4; \\ Q_1 = x_5 + x_4 + x_3 + x_2; \\ Q_0 = x_6 + x_5 + x_2 + x_1. \end{cases}$$

Полученные системы характеризуют работу шифраторов. Выражения для Q_2 одинаковы, значит для шифрации старшего разряда служит один и тот же элемент. Шифраторы разрядов Q_1 и Q_0 представлены четырьмя отдельными логическими элементами.

Мультиплексоры реализованы в виде набора логических элементов НЕ, 2-2-И-2ИЛИ. Такой способ выбран, поскольку для переключения всего двух каналов (двоичный код/код Грея) применение функционально законченного блока мультиплексора неоправданно. В зависимости от управляющего сигнала M , цифровая константа I подается на левые входы либо одного, либо другого вентиля 2И каждого из элементов 2-2-И-2ИЛИ. Подача логической еди-

ницы равносильна подключению выхода соответствующего шифратора к разрядному выходу всего устройства.

Рассмотрим отличительные особенности логической схемы автомата (рисунок 5.15), подготовленной для моделирования в MicroCAP.

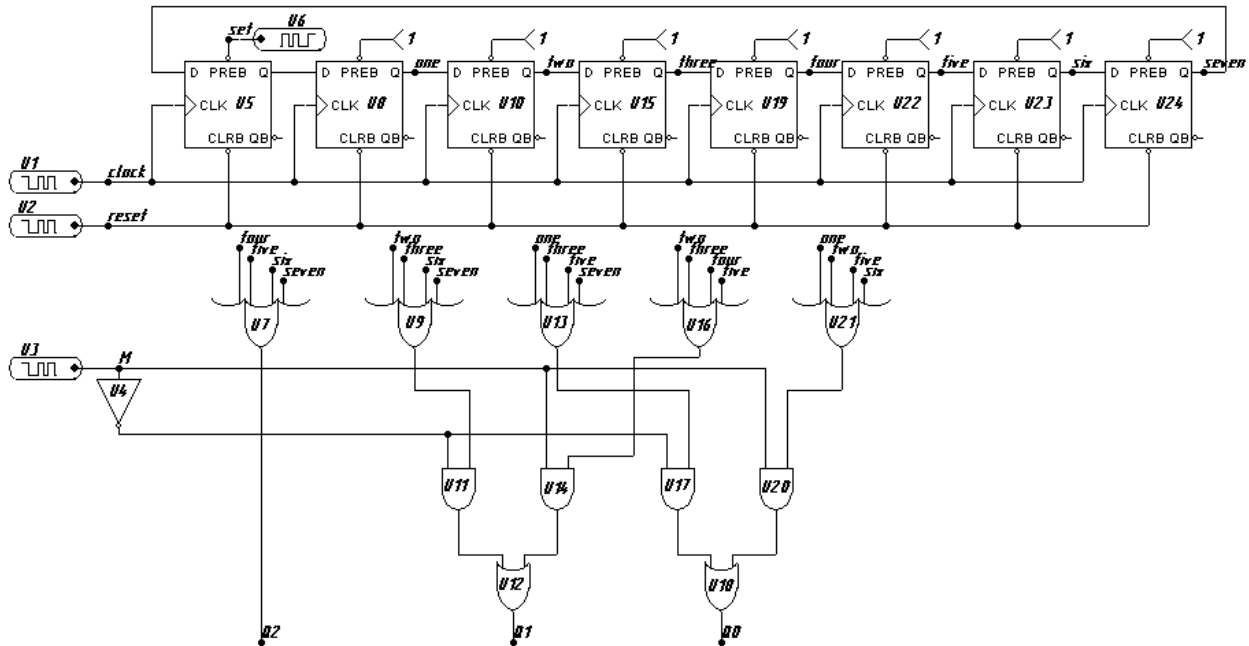
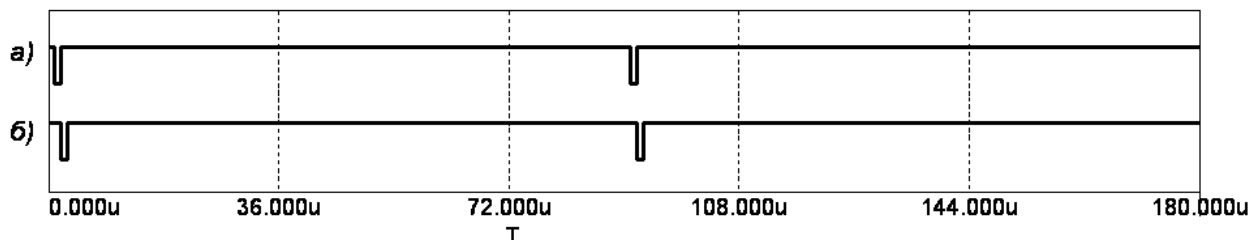


Рисунок 5.15 – Логическая схема автомата с кодированием «1 из N», подготовленная в программе MicroCAP

Помимо управляющих источников синхронизации *Clock*, сброса *Reset* и выбора режима *M*, в схеме присутствует также источник *U6* для записи активного уровня сигнала в крайний левый триггер в начале каждого цикла. Ранее говорилось о том, что для кольцевых регистров важно в начальный момент времени записать активный уровень сигнала, который затем под воздействием синхросигналов будет непрерывно циркулировать по кольцу. В данном случае предлагается запись активного уровня реализовать в виде подачи на вход предустановки левого триггера отрицательного строб-импульса. Последовательно возникающие друг другом два строб-импульса (сброса и записи единицы) подготавливают автомат к работе в каждом цикле. Временная диаграмма таких сигналов представлена на рисунке 5.16.



a) сигнал сброса; *б)* сигнал записи единицы

Рисунок 5.16 – Временная диаграмма сигналов сброса и записи единицы

Ниже приведен листинг с описанием параметров сигнала записи единицы *Set* и для сравнения листинг сигнала *Reset* (из прошлой лабораторной работы). Можно заметить, что сигнал *Set* смещен во времени относительно сигнала *Reset* на 1 мкс.

.DEFINE SET

+ 0uS 1
+ 2uS 0
+ 3us 1
+ 92us 0
+ 93us 1

.DEFINE RESET

+ 0us 1
+ 1us 0
+ 2us 1
+ 91us 0
+ 92us 1

Замечание. Схема, подготовленная для моделирования в MicroCAP (рисунок 5.15), имеет текстовое, а не числовое обозначение номеров разрядов регистра и входов шифраторов: *one, two, ..., seven*. Сделано это не случайно. Следует помнить, что программа MicroCAP автоматически и на свое усмотрение проставляет номера контрольных точек на электрических схемах. По умолчанию эта информация скрыта от пользователя. Таким образом, дополнительное числовое обозначение каких-либо контрольных точек схемы приводит к конфликтной ситуации номеров, проставленных программой и пользователем. Наиболее простое решение проблемы заключается в текстовом именовании контрольных точек по мнемоническому правилу, например, написание числительных английского языка.

Моделирование схемы во временной области, а также анализ полученных результатов, не содержат принципиальных отличий от аналогичных действий на предыдущем этапе работы. Сравнивая последовательность состояний трехразрядной шины *bin(Q2,Q1,Q0)* на рисунке 5.17 с результатами прошлого этапа, можно убедиться в адекватности проведенного исследования.

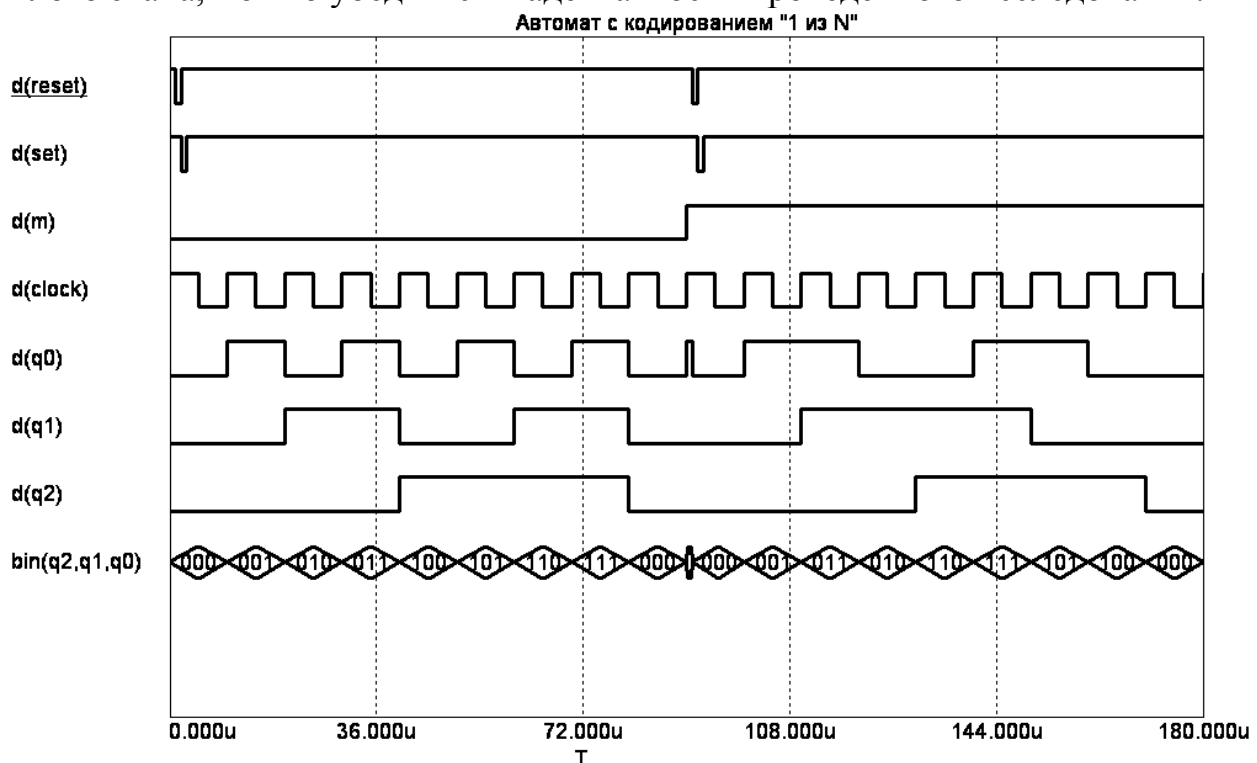


Рисунок 5.17 – Временная диаграмма работы автомата с кодированием «1 из N »

5.6 Лабораторное задание

Требуется спроектировать цифровой автомат двумя способами: на основе мультиплексного управления и на основе кодирования состояний автомата кодами «1 из N ». Трехразрядный автомат должен иметь два режима работы (см. лабораторную работу №4). Если управляющий сигнал $M = 0$, то автомат работает как двоичный счетчик с модулем счета 8, при $M = 1$ – согласно приведенной в варианте задания последовательности состояний. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

5.7 Контрольные вопросы

1. В чем заключается достоинство структуры автоматов с памятью на основе мультиплексного управления?
2. Какие сигналы образуют управляющее (адресное) слово мультиплексоров в структуре автоматов с мультиплексным управлением?
3. Для какой цели присутствуют D -триггеры в структуре автоматов с мультиплексным управлением?
4. Откуда берется информация для записи столбца цифровых констант на информационные входы мультиплексоров в структуре автоматов с мультиплексным управлением?
5. Какие функциональные узлы входят в состав структуры автоматов с кодированием «1 из N »?
6. В чем заключается основной недостаток кольцевого регистра?
7. Каким образом можно реализовать запись единицы в первый триггер кольцевого регистра, если весь набор выполнен как единая ИС?
8. Что такое полный шифратор?
9. Почему автоматы с кодированием «1 из N » обладают наилучшим быстродействием по сравнению с другими структурами автоматов?

5.8 Варианты заданий

№ ВАРИА НТА	ПОСЛЕДОВАТЕЛЬНОСТЬ СОСТОЯНИЙ ПРИ $M = 1$
1	000 → 010 → 011 → 111 → 101 → 001 → 110 → 100 → 000
2	000 → 011 → 100 → 110 → 001 → 111 → 101 → 010 → 000
3	000 → 100 → 101 → 011 → 111 → 001 → 110 → 010 → 000
4	000 → 101 → 110 → 100 → 001 → 111 → 010 → 011 → 000
5	000 → 110 → 111 → 101 → 100 → 001 → 010 → 011 → 000
6	000 → 111 → 001 → 011 → 010 → 100 → 110 → 101 → 000
7	000 → 001 → 010 → 111 → 011 → 110 → 100 → 101 → 000
8	000 → 010 → 100 → 001 → 111 → 110 → 011 → 101 → 000
9	000 → 011 → 101 → 100 → 001 → 010 → 111 → 110 → 000
10	000 → 100 → 110 → 111 → 101 → 001 → 010 → 011 → 000
11	000 → 101 → 111 → 011 → 110 → 001 → 010 → 100 → 000
12	000 → 110 → 011 → 101 → 111 → 001 → 100 → 010 → 000
13	000 → 111 → 010 → 100 → 001 → 101 → 011 → 110 → 000
14	000 → 001 → 100 → 111 → 110 → 101 → 010 → 011 → 000
15	000 → 010 → 101 → 011 → 001 → 100 → 111 → 110 → 000
16	000 → 011 → 110 → 100 → 101 → 111 → 001 → 010 → 000
17	000 → 100 → 111 → 011 → 001 → 010 → 101 → 110 → 000
18	000 → 101 → 001 → 111 → 010 → 100 → 011 → 110 → 000
19	000 → 110 → 011 → 001 → 010 → 100 → 101 → 111 → 000
20	000 → 111 → 100 → 001 → 010 → 011 → 101 → 110 → 000
21	000 → 001 → 111 → 101 → 011 → 100 → 110 → 010 → 000
22	000 → 010 → 001 → 111 → 101 → 110 → 011 → 100 → 000
23	000 → 011 → 001 → 100 → 111 → 010 → 110 → 101 → 000
24	000 → 100 → 001 → 101 → 011 → 010 → 110 → 111 → 000
25	000 → 101 → 100 → 010 → 001 → 011 → 110 → 111 → 000

6 ЛАБОРАТОРНАЯ РАБОТА №6 – ПРОЕКТИРОВАНИЕ ДВОИЧНО-КОДИРОВАННЫХ СЧЕТЧИКОВ С ПРОИЗВОЛЬНЫМ МОДУЛЕМ

6.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

- 1) ознакомление со способами исключения лишних состояний цифровых автоматов для построения счетчиков с произвольным модулем;
- 2) приобретение навыков построения счетчиков с произвольным модулем счета методом модификации межразрядных связей и методом управления сбросом;
- 3) изучение принципа работы параллельных счетчиков прямого и обратного счета;
- 4) закрепление навыков минимизации недоопределенных логических функций.

6.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 6.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

6.3 Способы построения двоично-кодированных счетчиков

Счетчики с модулем, не равным целой степени числа 2, т.е. с произвольным модулем, реализуются на основе нескольких методов.

Для построения счетчика с произвольным модулем M берется разрядность $n = \lceil \log_2 M \rceil$, где $\lceil \rceil$ – знак округления до ближайшего справа целого числа. Иными словами, исходной структурой как бы служит двоичный счетчик с модулем 2^n , превышающим заданный и ближайший к нему. Такой двоичный счетчик имеет $2^n - M = L$ лишних (неиспользуемых) состояний, подлежащих исключению.

Способы *исключения лишних состояний* многочисленны, и для любого M можно предложить множество реализаций счетчика. Исключая некоторое число первых состояний, получим ненулевое начальное состояние счетчика, что приводит к отсутствию естественного порядка счета и регистрации в счетчике кода с избытком. Исключение последних состояний позволяет сохранить естественный порядок счета. Сложность обоих вариантов принципиально одинакова, поэтому далее будем ориентироваться на схемы с естественным порядком счета. Состояния счетчиков во всех случаях предполагаем закодированными двоичными числами, т.е. будем рассматривать двоично-кодированные счетчики.

В счетчиках с исключением последних состояний счет ведется обычным способом вплоть до достижения числа $M - 1$. Далее последовательность переходов счетчика в направлении роста регистрируемого числа должна быть прервана, и следующее состояние должно быть нулевым. При этом счетчик будет иметь M внутренних состояний (от 0 до $M - 1$), т.е. его модуль равен M .

Остановимся на двух способах построения счетчиков с произвольным модулем: *модификации межразрядных связей* и *управлении сбросом*. При построении счетчика с модифицированными межразрядными связями последние, лишние, состояния исключаются непосредственно из таблицы функционирования счетчика. При этом после построения схемы обычным для синтеза автоматов способом получается счетчик, специфика которого состоит в нестандартных функциях возбуждения триггеров, и, следовательно, в нестандартных связях между триггерами, что и объясняет название способа. Схема получается как специализированная, изменение модуля счета требует изменение самой схемы, т.е. легкость перестройки с одного модуля на другой отсутствует. В то же время реализация схемы счетчика может оказаться простой.

При управлении сбросом выявляется момент достижения содержимым счетчика значения $M - 1$. Это является сигналом сброса счетчика в следующем такте, после чего начинается новый цикл. Этот вариант обеспечивает легкость перестройки счетчика на другие значения модуля, т.к. требуется изменять лишь код, с которым сравнивается содержимое счетчика для выявления момента сброса.

Построение счетчика первым способом. Простейшим счетчиком является счетчик по модулю 2, представляющий собой T -триггер при $T = 1$. Действительно, при $T = 1$ новое состояние триггера можно записать как [3]:

$$Q_H = QT + \bar{Q}T = \bar{Q},$$

где Q – старое состояние триггера; T – значение сигнала на информационном входе.

Для случая синхронного T -триггера новое состояние записывается как [3]:

$$Q_H = (QT + \bar{Q}T)C + \bar{C}Q = C\bar{Q} + \bar{C}Q,$$

где C – значение синхросигнала.

Следовательно, состояния триггера 0 и 1 циклически изменяются при каждом изменении тактового сигнала C . Граф переходов счетчика по модулю 2 представлен на рисунке 6.1.

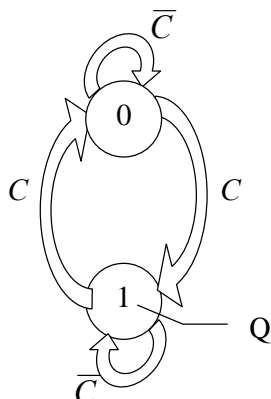
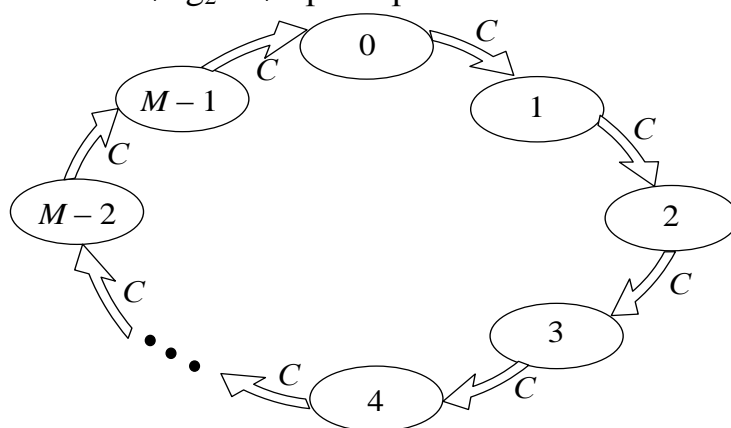


Рисунок 6.1 – Граф переходов счетчика по модулю 2

На рисунке 6.2 показан общий случай – граф переходов счетчика по модулю M , внутренние состояния которого обозначены числами от 0 до $M - 1$. Как уже было сказано, для получения M разных состояний необходимо использовать не менее $n = \lceil \log_2 M \rceil$ триггеров.



**Рисунок 6.2 – Граф переходов счетчика по модулю M
(переходы при $C = 0$ не показаны)**

Способ кодирования внутренних состояний счетчика может быть произвольным. Важно только, чтобы все внутренние состояния были разные. В общем случае от выбранного способа кодирования внутренних состояний автомата зависит его сложность. Закодируем внутренние состояния счетчика значениями выходных сигналов $n = \lceil \log_2 M \rceil$ триггеров $Q_0, Q_1, \dots, Q_{n-2}, Q_{n-1}$, так как это показано на рисунке 6.3.

На основании рисунка 6.3 составляется таблица истинности (таблица 6.1).

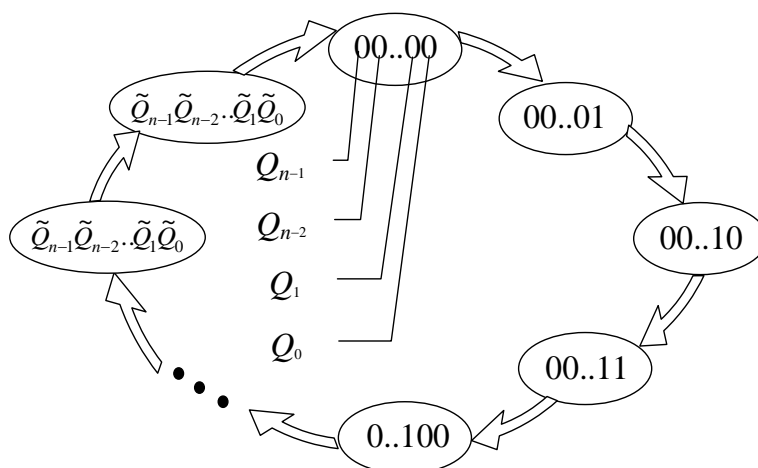


Рисунок 6.3 – Кодированный граф переходов счетчика по модулю M

Таблица 6.1 – Таблица истинности счетчика по модулю M

Q_{N-1}	Q_{N-2}	...	Q_1	Q_0	$Q_{H,N-1}$	$Q_{H,N-2}$...	$Q_{H,1}$	$Q_{H,0}$
0	0	...	0	0	0	0	...	0	1
0	0	...	0	1	0	0	...	1	0
0	0	...	1	0	0	0	...	1	1
0	0	...	1	1	0	0	...(1)	0	0
...

Для синтеза счетчиков, как и любых цифровых автоматов, можно использовать триггеры любых типов: D , T , JK . Сложность автомата в общем случае зависит от используемых типов триггеров. Следует иметь в виду, что в одном и том же автомате можно использовать триггеры разных типов. Выполним синтез счетчика по модулю M из триггеров типа JK . Для этого нужно найти функции возбуждения J_i и K_i ($i = 0, 1, \dots, n - 1$). Из функции переходов JK -триггеров [3] вытекает, что

$$Q_{Hi} = J_i \bar{Q}_i + \bar{K}_i Q_i$$

логическое выражение с двумя неизвестными J_i и K_i , которое нужно решить относительно этих неизвестных. Пусть $Q_i = 0$, тогда $Q_{Hi} = J_i \bar{0} + \bar{K}_i 0$. Из этого уравнения следует, что $J_i = Q_{H,i}$, а $K_i = X$ – произвольное значение. Пусть теперь $Q_i = 1$, тогда $Q_{Hi} = J_i \bar{1} + \bar{K}_i 1$. Из данного уравнения следует, что $J_i = X$, а $K_i = \bar{Q}_{Hi}$. Объединив оба решения при $Q_i = 0$ и $Q_i = 1$, получим:

$$J_i = \begin{cases} Q_{H,i}, & \text{если } Q_i = 0; \\ X, & \text{если } Q_i = 1. \end{cases} \quad (6.1)$$

$$K_i = \begin{cases} \bar{Q}_{H,i}, & \text{если } Q_i = 1; \\ X, & \text{если } Q_i = 0. \end{cases} \quad (6.2)$$

Дополним исходную таблицу истинности 6.1 столбцами функций возбуждения J_i и K_i . В результате получим расширенную таблицу истинности (таблица 6.2).

Таблица 6.2 – Расширенная таблица истинности счетчика по модулю M

Q_{N-1}	Q_{N-2}	...	Q_1	Q_0	$Q_{H,N-1}$	$Q_{H,N-2}$...	$Q_{H,1}$	$Q_{H,0}$	J_{N-1}	K_{N-1}	J_{N-2}	K_{N-2}	...	J_1	K_1	J_0	K_0
0	0	...	0	0	0	0	...	0	1	0	X	0	X	...	0	X	1	X
0	0	...	0	1	0	0	...	1	0	0	X	0	X	...	1	X	X	1
0	0	...	1	0	0	0	...	1	1	0	X	0	X	...	X	0	1	X
0	0	...	1	1	0	0	...(1)	0	0	0	X	0	X	...	X	1	X	1
...

Дополнительные столбцы функций возбуждения J_i и K_i заполняются согласно системам уравнений (6.1), (6.2). Например, для столбцов J_{n-1} и K_{n-1} имеем следующие закономерности. В четырех ячейках столбца Q_{n-1} размещены нули, следовательно, в четыре ячейки столбца J_{n-1} нужно скопировать новые значения из столбца $Q_{H,n-1}$, т.е. нули. По этой же причине в четыре ячейки столбца K_{n-1} нужно записать произвольные значения X .

После заполнения расширенной таблицы истинности необходимо составить логические выражения для функций возбуждения J_i и K_i . Цель этого этапа – нахождение наиболее простых форм логических выражений для функций J_i и K_i , выраженных через базис старых состояний триггеров:

$$J_i = f(Q_{n-1}, Q_{n-2}, \dots, Q_1, Q_0) \quad (6.3)$$

$$K_i = g(Q_{n-1}, Q_{n-2}, \dots, Q_1, Q_0) \quad (6.4)$$

Очевидно, что задача нахождения минимальных форм функциональных зависимостей (6.3) и (6.4) аналогична задаче минимизации недоопределенных логических функций (см. лабораторную работу №2), поскольку значения функций J_i и K_i определены не на всех наборах аргументов.

Одно из условий минимизации недоопределенных логических функций – первоначальное ее представление в виде СДНФ. Заметим, что в общем случае для описания счетчика по модулю M в таблице истинности содержится меньшее количество строк, чем требуется для записи СДНФ:

$$M < 2^n, \quad (6.5)$$

где $n = \lceil \log_2 M \rceil$ – количество аргументов.

Учитывая неравенство (6.5), дополним таблицу истинности счетчика по модулю M количеством строк равным $2^n - M$, в каждой из которых значение логических функций J_i и K_i принимает неопределенное значение X (таблица 6.3).

Дальнейший синтез минимальных форм недоопределенных логических функций J_i и K_i ничем не отличается от последовательности, рассмотренной в лабораторной работе №2:

а) записывается СДНФ функции f_0 , полученной из функции f заданием значения 0 на всех запрещенных наборах аргументов;

б) записывается СДНФ функции f_1 , полученной из функции f заданием

значения 1 на всех запрещенных наборах аргументов;

в) функция f_1 приводится к сокращенной форме (к форме, содержащей все простые импликанты);

г) составляется импликантная таблица из всех членов функции f_0 и простых импликант функции f_1 ;

д) искомая минимальная форма составляется из простых импликант функции f_1 , поглощающих все члены СДНФ функции f_0 .

Таблица 6.3 – Таблица истинности счетчика по модулю M с количеством строк 2^n

Q_{N-1}	Q_{N-2}	...	Q_1	Q_0	$Q_{Н,N-1}$	$Q_{Н,N-2}$...	$Q_{Н,1}$	$Q_{Н,0}$	J_{N-1}	K_{N-1}	J_{N-2}	K_{N-2}	...	J_1	K_1	J_0	K_0
0	0	...	0	0	0	0	...	0	1	0	X	0	X	...	0	X	1	X
0	0	...	0	1	0	0	...	1	0	0	X	0	X	...	1	X	X	1
0	0	...	1	0	0	0	...	1	1	0	X	0	X	...	X	0	1	X
0	0	...	1	1	0	0	...(1)	0	0	0	X	0	X	...	X	1	X	1
...
1	1	...	1	0	X	X	...	X	X	X	X	X	X	...	X	X	X	X
1	1	...	1	1	X	X	...	X	X	X	X	X	X	...	X	X	X	X

Полученные в результате минимизации логические выражения (6.3) и (6.4) определяют аппаратную структуру комбинационных цепей на входах соответствующих триггеров (рисунок 6.4).

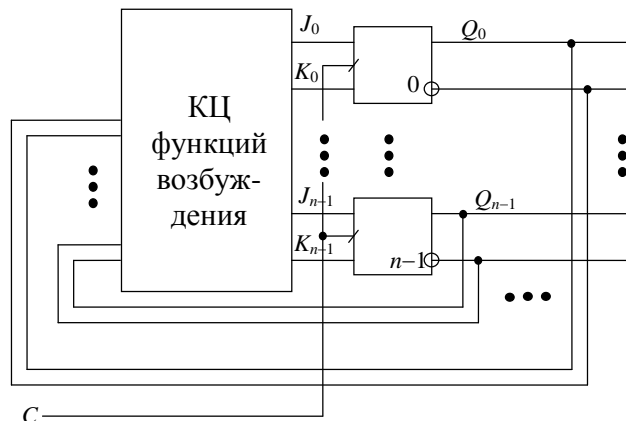


Рисунок 6.4 – Обобщенная структурная схема счетчика по модулю M

Построение счетчика вторым способом. Второй способ построения счетчиков с произвольным модулем позволяет изменить модуль счета очень простым приемом, не требующим изменений самой схемы счетчика. Сначала рассмотрим этот способ применительно к синхронному счетчику с параллельным переносом и счетом в прямом направлении (рисунок 6.5).

Функции возбуждения двоичного счетчика указанного типа, как известно [3], имеют вид $J_i = K_i = Q_0 Q_1 \dots Q_{i-1}$ (причем в младшем разряде $J_0 = K_0 = 1$). Введем в эти функции сигнал сброса R , изменив их следующим образом:

$$J_i = (Q_0 Q_1 \dots Q_{i-1}) \bar{R},$$

$$K_i = J_i + R.$$

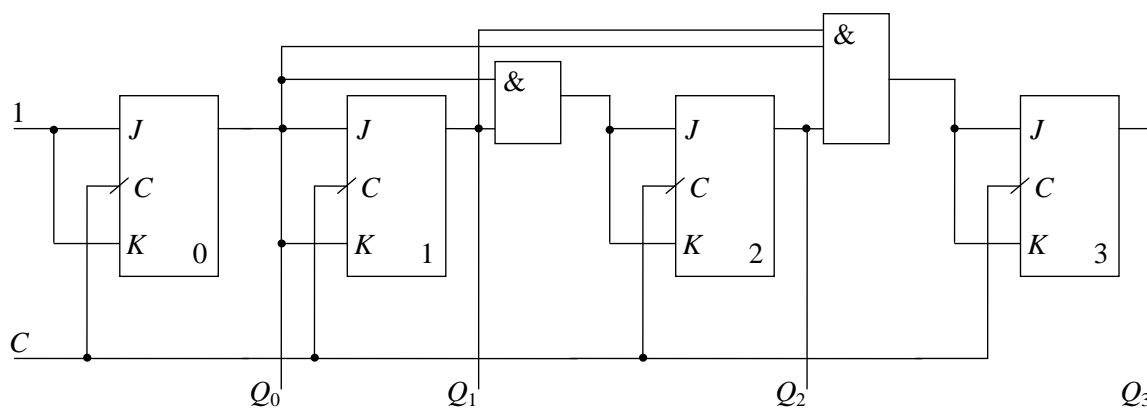
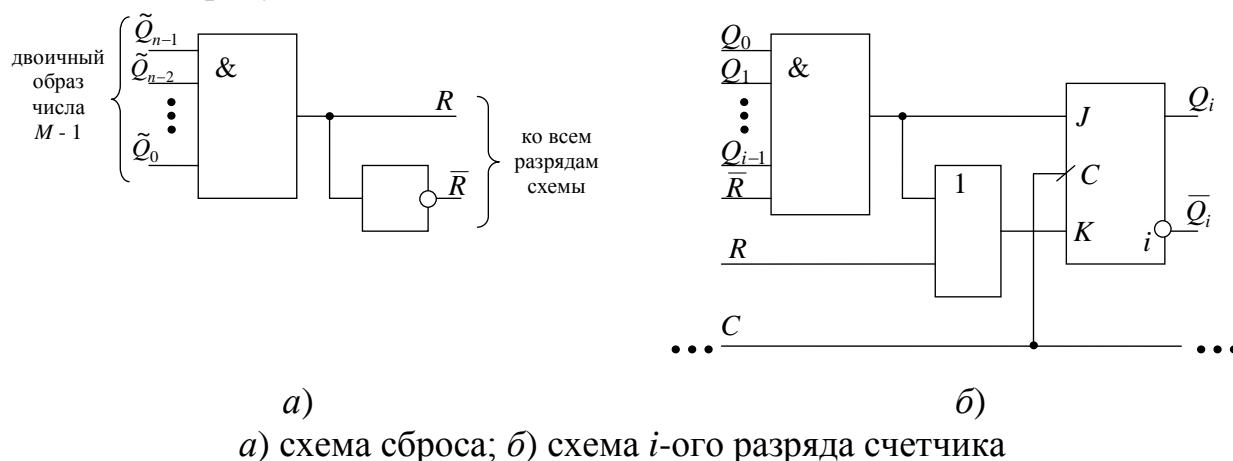


Рисунок 6.5 – Схема параллельного счетчика прямого направления

Пока сигнал сброса отсутствует ($R = 0$), функции J_i и K_i не отличаются от соответствующих функций двоичного счетчика. Когда сигнал R приобретает единичное значение, все функции J_i становятся нулевыми, K_i – единичными, что заставляет все триггеры сброситься по приходу следующего такта.

Если сигнал R появится как следствие возникновения в счетчике числа $M - 1$, то будет реализована последовательность счета $0, 1, 2, \dots, M - 1, 0, \dots$, т.е. счетчик с модулем M .

Схемы всех разрядов счетчика с управляемым сбросом не зависят от модуля счета. Кроме разрядных схем, счетчик содержит один конъюнктор, вырабатывающий сигнал сброса при достижении содержимым счетчика значения $M - 1$ (рисунок 6.6).



а) схема сброса; б) схема i -ого разряда счетчика

Рисунок 6.6 – Фрагмент схемы счетчика с управляемым сбросом

Если, например, имеется четырехразрядный счетчик, и на входы конъюнктора выработки сигнала сброса подключены выходы триггеров, как показано на рисунке 6.7, то сброс произойдет после достижения счетчиком числа $(1001)_2 = (9)_{10}$, т.е. счетчик будет работать как двоично-десятичный.

Несколько сложнее реализуется способ управляемого сброса применительно к синхронному параллельному счетчику с обратным счетом. Известно [3], что основное отличие параллельного счетчика с обратным счетом от счетчика с прямым счетом заключается в разном подключении входов триг-

геров к выходам предыдущих. В частности, для схемы счетчика на JK -триггерах сигнал i -ого разряда снимается с инверсного выхода JK -триггера \bar{Q}_i и подается затем на входы триггера следующего $i+1$ разряда (рисунок 6.8).

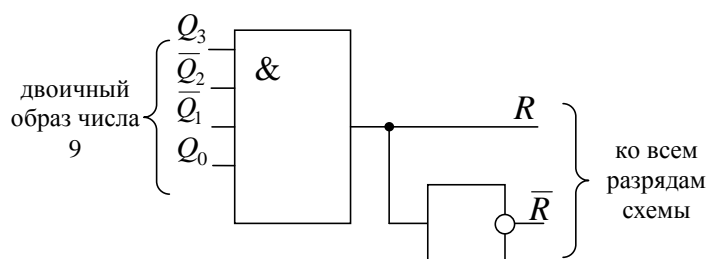


Рисунок 6.7 – Схема выработки сигнала сброса для двоично-десятичного счетчика

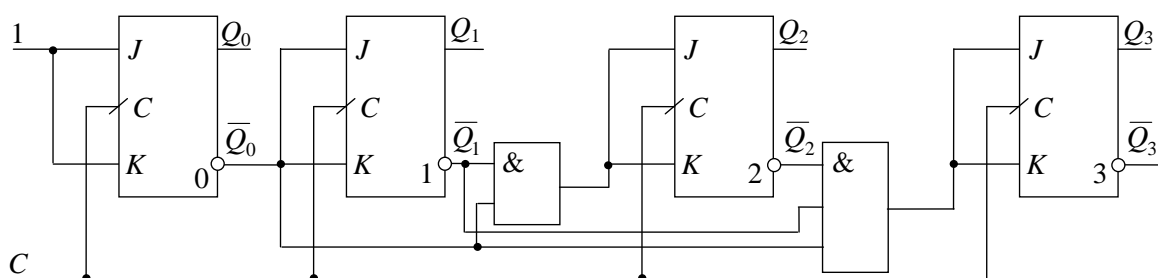


Рисунок 6.8 – Схема параллельного счетчика обратного направления

Функции возбуждения счетчика с обратным направлением счета будут иметь вид $J_i = K_i = \bar{Q}_0 \bar{Q}_1 \dots \bar{Q}_{i-1}$ (в младшем разряде по-прежнему $J_0 = K_0 = 1$). Пусть последовательность обратного счета выглядит как следующий ряд чисел:

$$M-1, M-2, \dots, 1, 0, M-1, \dots$$

Тогда, в отличие от прямого счета, необходимо ввести сигнал установки Set . Сигнал Set по достижению счетчиком числа 0 устанавливает на n триггерах двоичный аналог числа $M-1$ для нового цикла счета (рисунок 6.9).

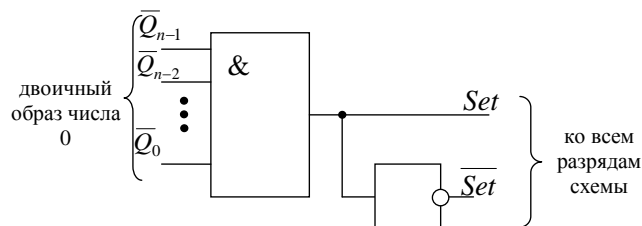


Рисунок 6.9 – Схема выработки сигнала установки

В общем случае двоичный аналог десятичного числа $M-1$ может содержать y единиц и z нулей:

$$(M-1)_{10} = (\tilde{Q}_{n-1} \tilde{Q}_{n-2} \dots \tilde{Q}_1 \tilde{Q}_0)_2;$$

$$\sum_{i=0}^{n-1} (\tilde{Q}_i |_{\tilde{Q}_i=1}) = y; \quad \sum_{i=0}^{n-1} (\tilde{Q}_i |_{\tilde{Q}_i=0}) = z.$$

Следовательно, разрядные схемы триггеров должны иметь две разновидности: случай установки в разряд значения 0 и случай установки значения 1. Легко заметить, что случай установки в i -ый разряд триггера логического нуля аналогичен случаю выработки сигнала сброса для счетчика прямого направления:

$$J_i = (\overline{Q_0} \overline{Q_1} \dots \overline{Q_{i-1}}) \overline{Set}, \quad (6.6)$$

$$K_i = J_i + Set, \quad (6.7)$$

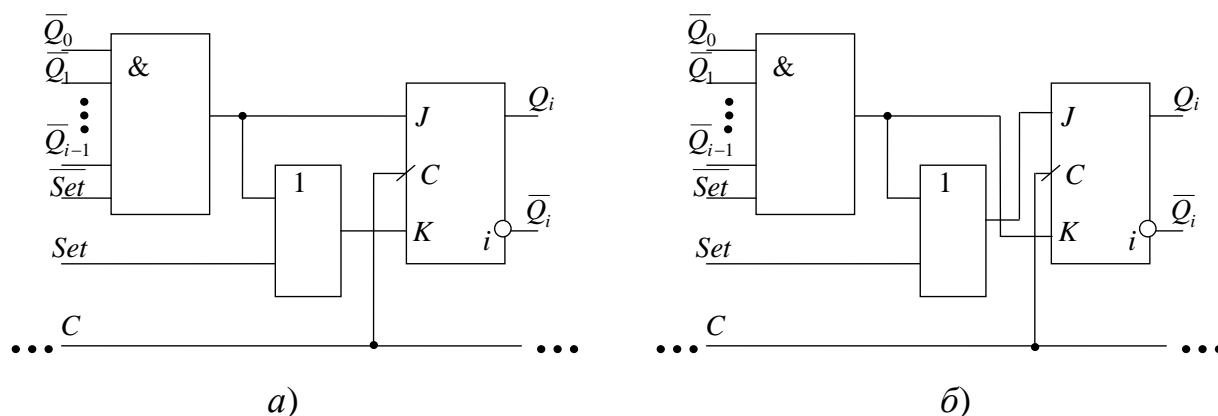
где $\overline{Q_0}, \overline{Q_1}, \dots, \overline{Q_{i-1}}$ – сигналы, снимаемые с инверсных выходов JK -триггеров.

Для установки в разряд триггера значения логической единицы следует в выражениях (6.6), (6.7) поменять местами функции возбуждения J_i и K_i :

$$K_i = (\overline{Q_0} \overline{Q_1} \dots \overline{Q_{i-1}}) \overline{Set},$$

$$J_i = K_i + Set.$$

Разрядные схемы триггеров для двух случаев представлены на рисунке 6.10.



а) установка в разряд логического нуля;
б) установка в разряд логической единицы

Рисунок 6.10 – Разрядные схемы триггеров для счетчика обратного направления

6.4 Примеры проектирования двоично-кодированного счетчика с произвольным модулем

Требуется спроектировать автомат-счетчик с модулем счета $M = 15$ двумя способами: на основе модификации межразрядных связей и на основе управления сбросом. Направление счета – прямое от 0 до 14. Элементная база – триггеры типа JK и логические элементы И, ИЛИ, НЕ. Частота тактовых импульсов синхронизации $f = 100$ кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

1 этап. Синтез счетчика на основе модификации межразрядных связей.

При выполнении этапа исследования использованы приемы №1-6, 8, 9 раздела «Типовые приемы работы в MicroCAP...».

Для построения счетчика с модулем счета $M = 15$ вычислим разрядность n цифрового автомата:

$$n = \lceil \log_2 M \rceil = \lceil \log_2 15 \rceil = \lceil 3.908 \rceil = 4.$$

Значит, исходной структурой для синтеза устройства будет служить двоичный счетчик с модулем $2^n = 16$. Такой счетчик имеет лишние состояния L , подлежащие исключению, в количестве:

$$L = 2^n - M = 16 - 15 = 1.$$

Рассмотрим последовательность заполнения таблицы истинности для счетчика (таблица 6.4). Сначала запишем заданную по условию последовательность счета в десятичной и двоичной системах счисления:

$$(0)_{10} \rightarrow (1)_{10} \rightarrow (2)_{10} \rightarrow (3)_{10} \rightarrow (4)_{10} \rightarrow (5)_{10} \rightarrow (6)_{10} \rightarrow (7)_{10} \rightarrow (8)_{10} \rightarrow (9)_{10} \rightarrow (10)_{10} \rightarrow \\ \rightarrow (11)_{10} \rightarrow (12)_{10} \rightarrow (13)_{10} \rightarrow (14)_{10}; \\ (0000)_2 \rightarrow (0001)_2 \rightarrow (0010)_2 \rightarrow (0011)_2 \rightarrow (0100)_2 \rightarrow (0101)_2 \rightarrow (0110)_2 \rightarrow (0111)_2 \rightarrow \\ \rightarrow (1000)_2 \rightarrow (1001)_2 \rightarrow (1010)_2 \rightarrow (1011)_2 \rightarrow (1100)_2 \rightarrow (1101)_2 \rightarrow (1110)_2.$$

Таблица 6.4 – Таблица истинности счетчика со строкой лишнего состояния

Q_3	Q_2	Q_1	Q_0	$Q_{н.3}$	$Q_{н.2}$	$Q_{н.1}$	$Q_{н.0}$	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	0	0	0	0	X	1	X	1	X	1	0	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

Заполним сверху вниз левый блок таблицы $Q_3Q_2Q_1Q_0$ в соответствие с приведенной последовательностью двоичных состояний. Затем заполним средний блок $Q_{н.3}Q_{н.2}Q_{н.1}Q_{н.0}$, рассматривая в каждой строке левого блока $Q_3Q_2Q_1Q_0$ двоичный код как старое состояние и находя по приведенной последовательности новое состояние. Под жирной горизонтальной чертой для левого блока $Q_3Q_2Q_1Q_0$ добавим все неиспользуемые состояния. В нашем случае – это единственное состояние $(15)_{10} = (1111)_2$. Во все остальные ячейки

под жирной горизонтальной чертой должны быть занесены неопределенные состояния X . Общее количество строк таблицы в итоге должно быть 2^n .

При заполнении правого блока $J_3K_3J_2K_2J_1K_1J_0K_0$ воспользуемся формулами для функций возбуждения JK -триггера (6.1) и (6.2):

$$J_i = \begin{cases} Q_{H.i}, & \text{если } Q_i = 0; \\ X, & \text{если } Q_i = 1, \end{cases}$$

$$K_i = \begin{cases} \bar{Q}_{H.i}, & \text{если } Q_i = 1; \\ X, & \text{если } Q_i = 0, \end{cases}$$

где Q_i – старое состояние i -ого триггера; $Q_{H.i}$ – новое состояние; X – неопределенное (произвольное) значение.

Поиск минимальных форм для функций возбуждения J_i и K_i аналогичен минимизации недоопределенных логических функций (см. лабораторную работу №2). Рассмотрим на примере функции возбуждения J_3 последовательность действий по поиску минимальной дизъюнктивной нормальной формы.

Доопределим столбец J_3 значениями логического нуля с последующей записью СДНФ $J_{3,0}$:

$$J_{3,0} = \bar{Q}_3Q_2Q_1Q_0.$$

Доопределим столбец J_3 значениями логической единицы с последующей записью СДНФ $J_{3,1}$. Затем, пользуясь логическим преобразователем Electronics Workbench, преобразуем СДНФ в МДНФ:

$$J_{3,1} = \bar{Q}_3Q_2Q_1Q_0 + Q_3\bar{Q}_2\bar{Q}_1\bar{Q}_0 + Q_3\bar{Q}_2\bar{Q}_1Q_0 + Q_3\bar{Q}_2Q_1\bar{Q}_0 + Q_3\bar{Q}_2Q_1Q_0 + \\ + Q_3Q_2\bar{Q}_1\bar{Q}_0 + Q_3Q_2\bar{Q}_1Q_0 + Q_3Q_2Q_1\bar{Q}_0 + Q_3Q_2Q_1Q_0 = Q_2Q_1Q_0 + Q_3.$$

Внимание! Для вариантов с обратным счетом возникает специфическая ситуация при использовании логического преобразователя Electronics Workbench. Напомним (см. прием №8), что наборы аргументов в таблице истинности логического преобразователя *всегда* представляют собой возрастающую последовательность двоичных чисел от $(00\dots0)_2$ до $(11\dots1)_2$, в то время как заполненная вручную таблица 6.4 для вариантов с обратным счетом образует иной порядок. Выход из создавшейся ситуации – переписать на бумаге таблицу 6.4, так чтобы наборы аргументов составляли возрастающую последовательность. Естественно, что при этом меняется не только набор аргументов, но и остальная информация, расположенная в строке правее.

Составим импликантную матрицу (таблица 6.5) для функции $J_{3,0}$ и для простых импликант (слагаемых) функции $J_{3,1}$. Крестиками отмечаются те столбцы членов СДНФ, которые поглощаются отдельными простыми импликантами (слагаемыми).

Таблица 6.5 – Импликантная матрица

ПРОСТЫЕ ИМПЛИКАНТЫ ФУНКЦИИ $J_{3,1}$	$\bar{Q}_3Q_2Q_1Q_0$
$Q_2Q_1Q_0$	×
Q_3	

Минимальная форма логического выражения функции J_3 состоит из простой импликанты, которая поглотила функцию $J_{3,0}$:

$$J_3 = Q_2 Q_1 Q_0. \quad (6.8)$$

Аналогичным образом могут быть получены остальные минимальные формы логических функций возбуждения $K_3, J_2, K_2, J_1, K_1, J_0, K_0$:

$$K_3 = Q_2 Q_1; \quad (6.9)$$

$$J_2 = Q_1 Q_0; \quad (6.10)$$

$$K_2 = Q_1(Q_0 + Q_3); \quad (6.11)$$

$$J_1 = Q_0; \quad (6.12)$$

$$K_1 = Q_0 + Q_3 Q_2; \quad (6.13)$$

$$J_0 = \overline{Q_3} + \overline{Q_2} + \overline{Q_1}; \quad (6.14)$$

$$K_0 = 1. \quad (6.15)$$

Выражения (6.8) – (6.15) определяют структурный состав комбинационных цепей на входах JK -триггеров. Аппаратная реализация счетчика с модулем 15 (рисунок 6.11) сходна с реализацией цифровых автоматов с памятью.

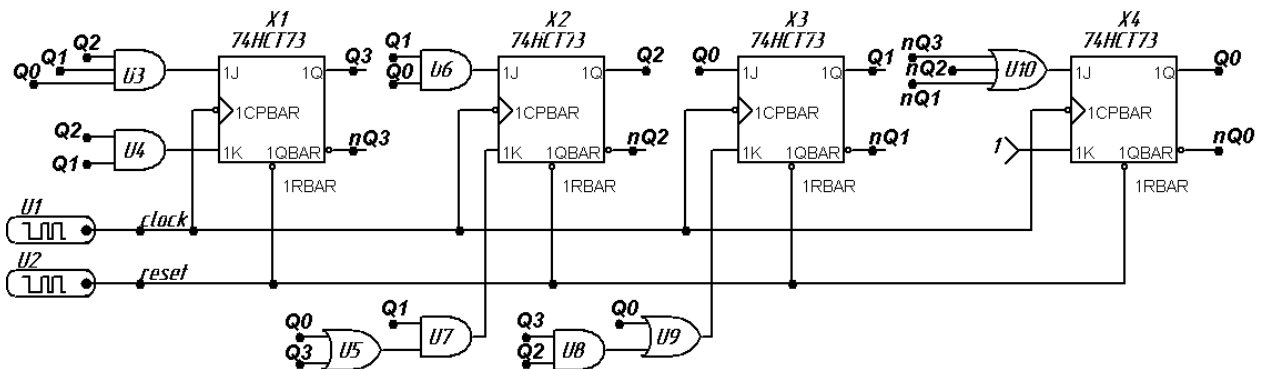


Рисунок 6.11 – Структурная схема счетчика с модулем 15 (способ модификации межразрядных связей)

В схеме применяется модель JK -триггера, имеющая реальный прототип – интегральную микросхему зарубежного производства 74HC73. В корпусе микросхемы 74HC73 содержится две независимые секции JK -триггеров, обладающие инверсным динамическим входом синхронизации. На поле чертежа каждая секция указанной микросхемы размещается по команде *Component/Digital Library/74xx42-/60-/74HC73*.

Параметры сигнала синхронизации *Clock* можно взять без изменений из методического примера лабораторной работы №4. Сигнал сброса *Reset* должен представлять собой строб-импульс, появляющийся в начальный момент работы счетчика. Учитывая инверсный вход сброса $QBAR$ JK -триггера, строб-импульс имеет нулевой активный уровень и единичный пассивный. Длительность строб-импульса примем как десятую часть периода следования синхросигнала $T/10 = 1$ мкс; момент возникновения фронта строб-импульса примем как +1 мкс от начала режима работы.

В диалоговом окне свойств источника $U2$ в строке **FORMAT** указывают значение **1**. При выборе строки **COMMAND** задают форму сигнала сброса:

.DEFINE RESET

+ **0us 1**

+ **1us 0**

+ **2us 1**

Рассчитаем длительность временного вида анализа. Частота тактовых импульсов синхронизации по условию $f = 100$ кГц, значит период следования

$T = \frac{1}{f} = 10 \text{ мкс}$. Количество состояний N счетчика равно модулю счета M :

$$N = M = 15.$$

При анализе в MicroCAP важно убедиться, что по достижению счетчиком максимального значения $(M - 1) = 14$ наступают новые циклы счета. По этой причине продлим временной вид анализа еще на 5 периодов следования синхроимпульсов. Тогда окончательная длительность t анализа составляет:

$$t = M \cdot T + 5 \cdot T = 15 \cdot 10 + 5 \cdot 10 = 200 \text{ мкс}.$$

В диалоговом окне свойств моделирования **Transient Analysis Limits** указывают:

- в строке ввода **Time Range** – длительность анализа **200u**;
- в таблице – наименование функций, отображаемых на графике.

P	X EXPRESSION	Y EXPRESSION
1	T	D(CLOCK)
1	T	D(RESET)
1	T	D(Q0)
1	T	D(Q1)
1	T	D(Q2)
1	T	D(Q3)
1	T	DEC(Q3,Q2,Q1,Q0)

В последней строке таблицы с помощью ключевого слова *dec* формируется четырехразрядная шина выходного сигнала счетчика. На рисунке 6.12 представлена временная диаграмма, из которой следует, что счетчик работает в прямом направлении от 0 до 14, т.е. удовлетворяет поставленному заданию.

II этап. Синтез счетчика на основе управления сбросом.

При выполнении этапа исследования использованы приемы №1-6, 9 раздела «Типовые приемы работы в MicroCAP...».

Исходной структурой для синтеза является схема синхронного счетчика с параллельным переносом. Для обеспечения прямого направления счета необходимо, чтобы входы триггеров соединялись с прямыми выходами триггеров младших разрядов (рисунок 6.13). Количество триггеров в схеме вычисляется аналогично:

$$n = \lceil \log_2 M \rceil = \lceil \log_2 15 \rceil = \lceil 3.908 \rceil = 4.$$

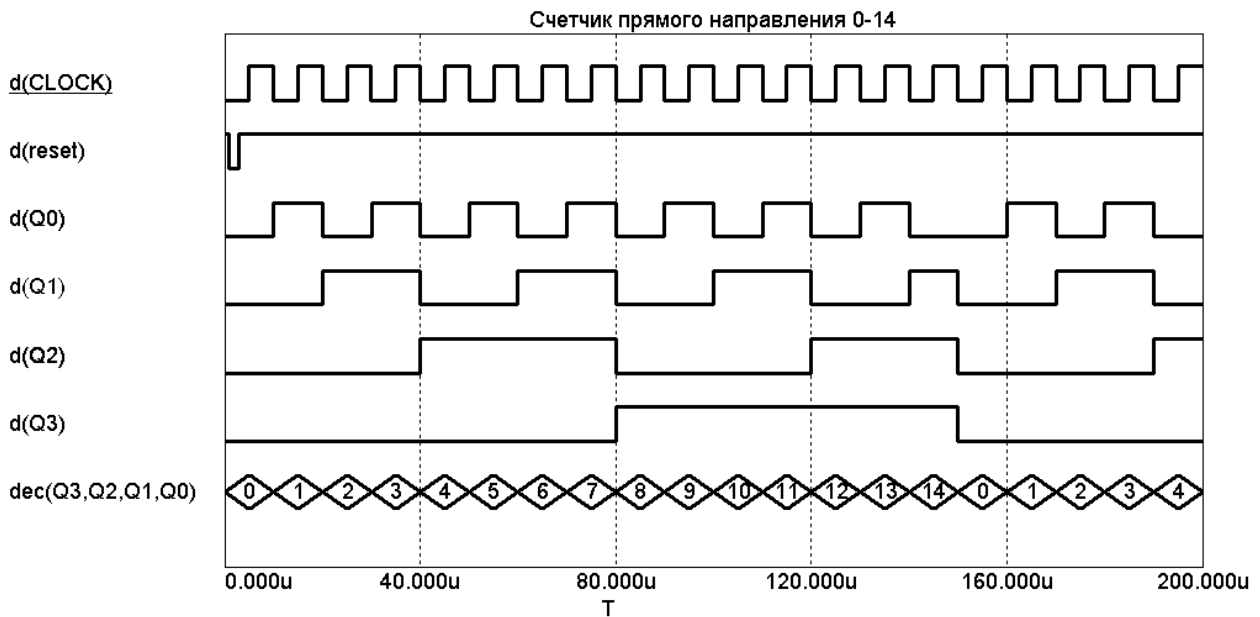


Рисунок 6.12 – Временная диаграмма работы счетчика с модулем 15 (способ модификации межрядных связей)

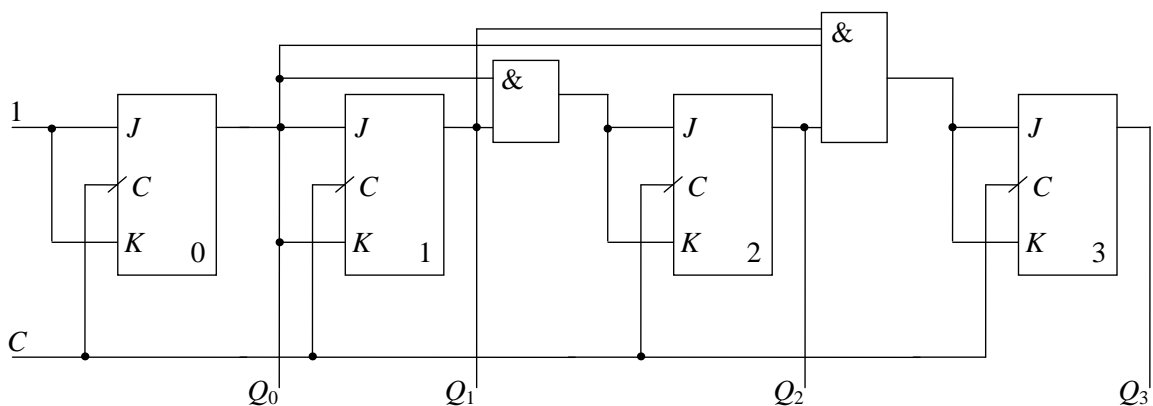


Рисунок 6.13 – Исходная схема счетчика с параллельным переносом и модулем счета 16

Функции возбуждения счетчика по модулю 16 равны:

$$\begin{cases} J_3 = K_3 = Q_0 Q_1 Q_2; \\ J_2 = K_2 = Q_0 Q_1; \\ J_1 = K_1 = Q_0; \\ J_0 = K_0 = 1. \end{cases} \quad (6.16)$$

Счетчик прямого направления с произвольным модулем предполагает наличие сигнала сброса R и, соответственно, схему выработки такого сигнала. Введем в исходные функции возбуждения J_i и K_i (6.16) сигнал сброса R , так чтобы:

$$\begin{aligned} J_i &= (Q_0 Q_1 \dots Q_i) \bar{R}, \\ K_i &= J_i + R, \end{aligned}$$

где $i = 0, 1, 2, 3$.

Пока сигнал сброса отсутствует ($R = 0$), функции J_i и K_i не отличаются от функций счетчика (6.16) с модулем 16. Когда сигнал R приобретает единичное значение, все функции J_i становятся нулевыми, а K_i – единичными. Это заставляет все триггеры сброситься.

По условию модуль счета должен быть $M = 15$, значит сигнал сброса R появляется при возникновении в счетчике числа $(M - 1)_{10} = (14)_{10} = (1110)_2$. Кодовая комбинация аргументов, соответствующая числу $(1110)_2$, есть $Q_3Q_2Q_1\bar{Q}_0$. Учитывая вышесказанное, схема выработки сигнала сброса может быть реализована на основе элемента 4И и элемента НЕ (рисунок 6.15). Сигнал сброса $R = 1$ появляется на выходе только в случае возникновения на входе схемы кода 1110.

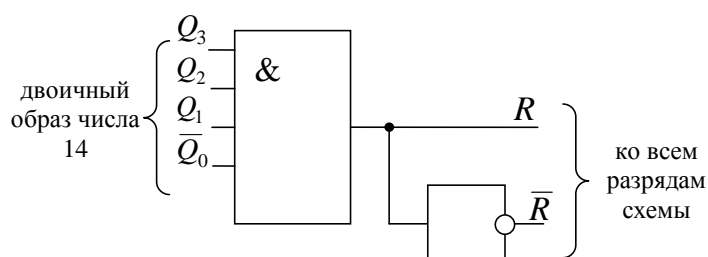


Рисунок 6.15 – Схема выработки сигнала сброса

Разрядные схемы триггеров счетчика необходимо дополнить элементом 2ИЛИ с целью учета сигнала сброса R . Кроме этого, в конъюнкторах каждого разряда следует добавить вход для подачи инверсного сигнала \bar{R} . Общий вид разрядной схемы триггера представлен на рисунке 6.16.

Совмещая схему выработки сигнала сброса и все разрядные схемы триггеров, получим новое схемотехническое решение – счетчик с модулем $M = 15$, управляемый сигналом сброса (рисунок 6.17).

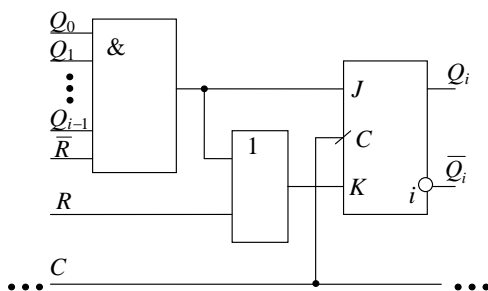


Рисунок 6.16 – Разрядная схема триггера

Заметим, что в синтезированной схеме триггеры младших разрядов расположены слева, а триггеры старших разрядов – справа. Эта особенность обусловлена структурным свойством двоичных счетчиков, содержащих триггер младшего разряда на входе схемы, т.е. слева. Технические подробности, связанные с моделированием схемы на рисунке 6.17, здесь не комментируются, поскольку аналогичны предыдущему этапу. Временные диаграммы сигнала

лов счетчика, выполненного на основе управления сбросом, показаны на рисунке 6.18.

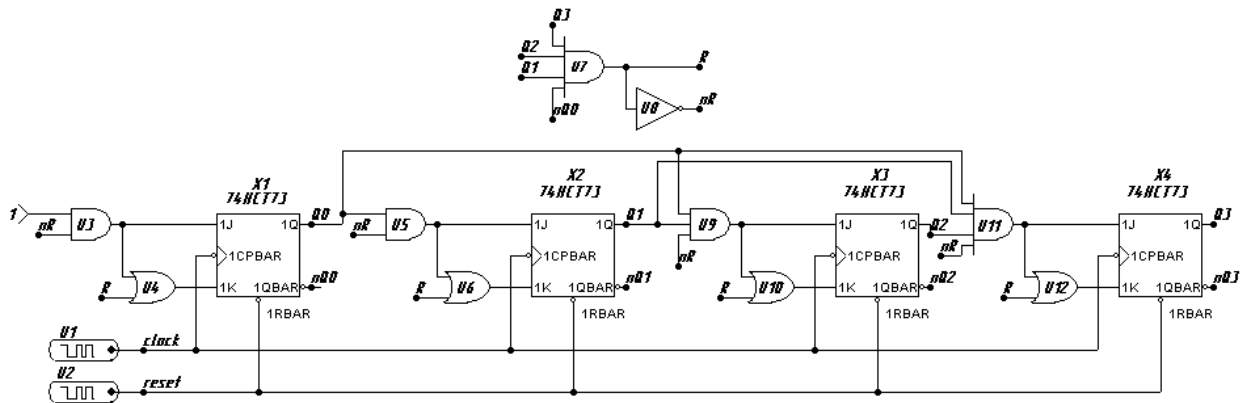


Рисунок 6.17 – Структурная схема счетчика с модулем 15 (способ управления сбросом)

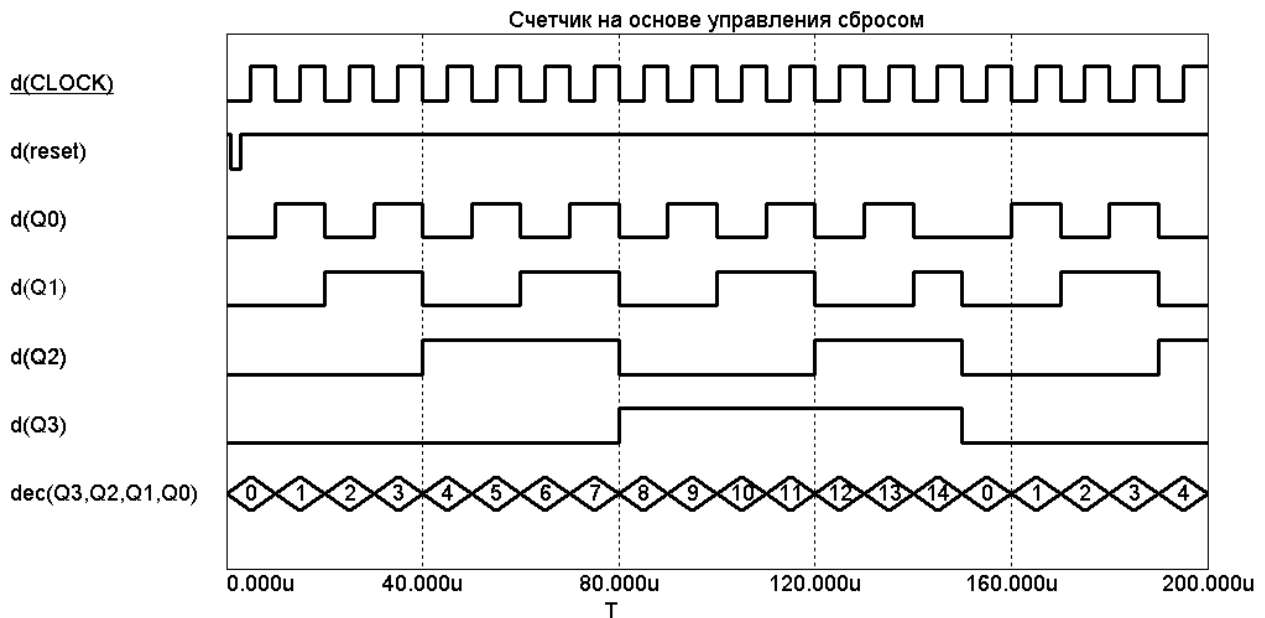


Рисунок 6.18 – Временные диаграммы работы счетчика с модулем 15 (способ управления сбросом)

Анализ временных зависимостей, их сравнение с результатами предыдущего этапа позволяет сказать об адекватности проведенного исследования.

6.5 Лабораторное задание

Варианты 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25. Спроектировать автомат-счетчик двумя способами: на основе модификации межразрядных связей и на основе управления сбросом. Направление счета – прямое, модуль счета – согласно варианту задания. Элементная база – триггеры типа *JK* и логические элементы И, ИЛИ, НЕ. Частота тактовых импульсов синхронизации

$f = 100$ кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

Варианты 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24. Спроектировать автомат-счетчик двумя способами: на основе модификации межразрядных связей и на основе управления начальным состоянием. Направление счета – обратное, модуль счета – согласно варианту задания. **Особенности способа управления начальным состоянием для счетчиков обратного направления рассмотрены в конце пункта 6.3.** Элементная база – триггеры типа JK и логические элементы И, ИЛИ, НЕ. Частота тактовых импульсов синхронизации $f = 100$ кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

6.6 Контрольные вопросы

1. Что такое счетчик с произвольным модулем счета M ?
2. Как вычислить необходимое число триггеров для реализации двоично-кодированного счетчика с модулем счета M ?
3. Какие виды триггеров можно применять для построения двоично-кодированных счетчиков?
4. Какова цель нахождения логических выражений для функций возбуждения J_i и K_i ?
5. Для чего нужно дополнять таблицу истинности счетчика по модулю M строками лишних состояний?
6. Что такое импликантная таблица (матрица)?
7. Чему равны функции возбуждения для синхронного счетчика с параллельным переносом и счетом в прямом направлении?
8. В чем отличие реализации счетчика на основе управляемого сброса и на основе управления начальным состоянием?
9. Какие разновидности разрядных схем триггеров возможны для счетчика на основе управления начальным состоянием?

6.7 Варианты заданий

№ ВАРИАНТА	НАПРАВЛЕНИЕ СЧЕТА	МОДУЛЬ СЧЕТА	ДИАПАЗОН ЗНАЧЕНИЙ
1	ПРЯМОЕ	5	$[0; 4]_{10}$
2	ОБРАТНОЕ	5	$[4; 0]_{10}$
3	ПРЯМОЕ	6	$[0; 5]_{10}$
4	ОБРАТНОЕ	6	$[5; 0]_{10}$
5	ПРЯМОЕ	7	$[0; 6]_{10}$
6	ОБРАТНОЕ	7	$[6; 0]_{10}$
7	ПРЯМОЕ	9	$[0; 8]_{10}$
8	ОБРАТНОЕ	9	$[8; 0]_{10}$
9	ПРЯМОЕ	10	$[0; 9]_{10}$
10	ОБРАТНОЕ	10	$[9; 0]_{10}$
11	ПРЯМОЕ	11	$[0; 10]_{10}$
12	ОБРАТНОЕ	11	$[10; 0]_{10}$
13	ПРЯМОЕ	12	$[0; 11]_{10}$
14	ОБРАТНОЕ	12	$[11; 0]_{10}$
15	ПРЯМОЕ	13	$[0; 12]_{10}$
16	ОБРАТНОЕ	13	$[12; 0]_{10}$
17	ПРЯМОЕ	14	$[0; 13]_{10}$
18	ОБРАТНОЕ	14	$[13; 0]_{10}$
19	ПРЯМОЕ	17	$[0; 16]_{10}$
20	ОБРАТНОЕ	17	$[16; 0]_{10}$
21	ПРЯМОЕ	18	$[0; 17]_{10}$
22	ОБРАТНОЕ	18	$[17; 0]_{10}$
23	ПРЯМОЕ	19	$[0; 18]_{10}$
24	ОБРАТНОЕ	19	$[18; 0]_{10}$
25	ПРЯМОЕ	20	$[0; 19]_{10}$

СПИСОК ЛИТЕРАТУРЫ

1. Разевиг В.Д. Схемотехническое моделирование с помощью Micro-CAP 7. – М.: Горячая линия – Телеком, 2003. – 368 с.
2. Алексеев В.П., Озёркин Д.В. Системный анализ и методы научно-технического творчества: Уч. пособие. – Томск: Издательство ИОА СО РАН, 2003. – 304 с.
3. Опадчий Ю.Ф. и др. Аналоговая и цифровая электроника (Полный курс): Учебник для вузов / Ю.Ф.Опадчий, О.П.Глудкин, А.И.Гуров; Под ред. О.П.Глудкина. – М.: Горячая Линия – Телеком, 2002. – 768 с.
4. Калабеков Б.А. Цифровые устройства и микропроцессорные системы: Учебник для техникумов связи. – М.: Горячая линия – Телеком, 2002. – 336 с.
5. Фролкин В.Т., Попов Л.Н. Импульсные и цифровые устройства: Учеб. пособие для вузов. – М.: Радио и связь, 1992. – 336 с.
6. Зельдин Е.А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре. – Л.: Энергоатомиздат. Ленингр. отд-ние, 1986. – 280 с.
7. Потемкин И.С. Функциональные узлы цифровой автоматики. – М.: Энергоатомиздат, 1988. – 320 с.
8. Браммер Ю.А. Цифровые устройства: Учебное пособие для вузов /Ю.А.Браммер, И.Н.Пашук. – М.: Высшая школа, 2004. – 229 с.
9. Пупырев Е.И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, Главная редакция физико-математической литературы, 1984. – 192 с.
10. Бойко В.И. и др. Схемотехника электронных систем. Цифровые устройства. – СПб.: БХВ-Петербург, 2004. – 512 с.
11. Micro-CAP 7.0. Electronic Circuit Analysis Program. Reference Manual – Sunnyvale: Spectrum Software, 2001. – 698 p.
12. Micro-CAP 7.0. Electronic Circuit Analysis Program. User's Guide – Sunnyvale: Spectrum Software, 2001. – 238 p.

ПРИЛОЖЕНИЕ 1 – СПРАВОЧНЫЕ СВЕДЕНИЯ О ПРОГРАММЕ СХЕМОТЕХНИЧЕСКОГО МОДЕЛИРОВАНИЯ MICROCAP

При создании принципиальных схем используются числа, переменные и математические выражения. Числовые значения параметров компонентов представляются в виде:

- действительных чисел с фиксированным десятичным знаком (**обратим внимание, что в качестве десятичного знака в программе MicroCAP используется точка**). Например, сопротивление 2,5 кОм, записывается как 2500, а емкость 1 мкФ как 0.000001;

- действительных чисел с плавающим десятичным знаком - научная нотация. Например, емкость 1 мкФ может быть записана как 1E-6;

- действительных чисел с плавающим десятичным знаком - инженерная нотация, согласно которой различные степени числа 10 обозначаются следующими суффиксами:

F фемто	10^{-15}	M милли	10^{-3}
P пико	10^{-12}	K кило	10^3
N нано	10^{-9}	MEG мега	10^6
U микро	10^{-6}	G гига	10^9
		T тера	10^{12}

Для экономии места на осях X, Y графиков результатов моделирования малая буква "m" обозначает 10^{-3} , большая буква "M" – 10^6 (вместо MEG). Во всех остальных случаях большие и малые буквы не различаются.

Например, сопротивление 1,5 МОм может быть записано как 1.5MEG, 1.5meg или 1500K, емкость 1 мкФ как 1U или 1uF. В последнем примере показано, что для большей наглядности после стандартных суффиксов допускается помещать любые символы, которые при интерпретации чисел не будут приниматься во внимание. **Пробелы между числом и буквенным суффиксом не допускаются.**

В программе Micro-CAP ряд констант и переменных имеют стандартные значения:

T – время в секундах;

F – частота в герцах;

DCINPUT1 – первая варьируемая переменная в DC-анализе;

E – $\text{EXP}(1) = 2,718281828$;

S – комплексная частота, равная $2 \cdot \text{PI} \cdot \text{J}$;

GMIN – минимальная проводимость ветви, задаваемая в диалоговом окне Options/Global settings;

PI – число $\pi = 3.14159265389795$;

TEMP – температура компонентов в градусах Цельсия;

V_T – температурный потенциал p - n -перехода, при $TEMP = 27 \text{ }^\circ\text{C}$
 $V_T = 25,86419 \text{ мВ}$;

J – корень квадратный из -1 ;

T_{min} – начальный момент времени расчета переходных процессов;

T_{max} – конечный момент времени расчета переходных процессов;

F_{min} – начальная частота расчета частотных характеристик;

F_{max} – конечная частота расчета частотных характеристик;

P_{GT} – общая мощность, генерируемая в схеме;

P_{ST} – общая мощность, запасаемая в схеме;

P_{DT} – общая рассеиваемая в схеме мощность.

Номера точек, присваиваемые программой MicroCAP автоматически, представляют собой целые числа, например 0, 2, 25. Кроме того, пользователь может присвоить любой точке имя в виде текстовой алфавитно-цифровой переменной, начинающейся с буквы или символа "_" и содержащей не более 50 символов, например A1, Out, Reset. В математических выражениях могут использоваться следующие переменные:

D(A)	Логическое состояние на проводнике A
V(A)	Напряжения в точке A (напряжения измеряются относительно узла "земли", которой программа присваивает номер 0)
V(A,B)	Разность потенциалов между точками A и B
V(D1)	Напряжение между выводами устройства D1
I(D1)	Ток через устройство D1
I(A,B)	Ток через ветвь между точками A и B
IR(Q1)	Ток, втекающий в вывод R устройства Q1
VRS(Q1)	Напряжение между выводами R и S устройства Q1
CRS(Q1)	Емкость между выводами R и S устройства Q1
QRS(Q1)	Заряд емкости между выводами R и S устройства Q1
R(R1)	Сопrotивление резистора R1
C(X1)	Емкость конденсатора или диода X1
Q(X1)	Заряд конденсатора или диода X1
L(X1)	Индуктивность катушки индуктивности или сердечника X1
X(L1)	Магнитный поток в катушке индуктивности или сердечнике L1
B(L1)	Магнитная индукция сердечника L1
H(L1)	Напряженность магнитного поля в сердечнике L1
RND	Случайное число с равномерным законом распределения на отрезке [0, 1]
ONoise	Корень квадратный из спектральной плотности выходного напряжения
INoise	Корень квадратный из спектральной плотности входного напряжения, равной $ONoise / \text{коэффициент передачи по мощности}$

PG(V1)	Мощность, генерируемая источником V1
PS(X1)	Реактивная мощность, накапливаемая в устройстве X1
PD(D1)	Мощность, рассеиваемая в устройстве D1

В этом перечне символы A и B обозначают номера точек схемы, D1 - имя компонента с двумя выводами или управляемого источника, Q1 - имя любого активного устройства или линии передачи. Символы R и S заменяются аббревиатурами выводов устройств согласно следующей таблице:

Устройство	Аббревиатуры выводов	Названия выводов
МОП-транзистор (MOSfets)	D,G,S,B	Сток, затвор, исток, подложка
Полевой транзистор (Jfets)	D,G,S	Сток, затвор, исток
Арсенид-галлиевый транзистор (GaAsfets)	D,G,S	Сток, затвор, исток
Биполярный транзистор (BJT)	B,E,C,S	База, эмиттер, коллектор, подложка
Статически индуцированный биполярный транзистор (IGBT)	C,G,E	Коллектор, затвор, эмиттер
Линия передачи (Tran. Line)	AP, AM, BP, BM	Вход+, вход-, выход+, выход-

Например, следующие выражения означают: $I(R1)$ – ток через резистор R1; $R(Rload)$ – сопротивление резистора Rload; $IC(VT1)$ – ток коллектора биполярного транзистора VT1; $VBE(Q1)$ – напряжение между базой и эмиттером биполярного транзистора Q1.

Приведем список обозначений переменных типа напряжение, ток, емкость и заряд для всех компонентов:

Компонент	Напряжение	Ток	Емкость	Заряд
Резистор	V	I	Нет	Нет
Конденсатор	V	I	C	Q
Индуктивность	V	I	Нет	Нет
Диод	V	I	C	Q
Линия передачи	VAP, VAM, VBP, VBM	IA, IB	Нет	Нет
Биполярный транзистор	VBE, VBC, VEB, VEC, VCB, VCE	IB, IE, IC	CBE, CBC	QBE, QBC

Компонент	Напряжение	Ток	Емкость	Заряд
Биполярный транзистор с выводом подложки	VBE, VBC, VBS, VEB, VEC, VES, VCB, VCE, VBS	IB, IE, IC, IS	CBE, CBC, CCS	QBE, QBC, QCS
МОП-транзистор	VGS, VGD, VGB, VDS, VDG, VDS, VSG, VSD, VSB, VBG, VBD, VBS	IG, IS, ID, IB	CGS, CGD, CGB, CBD, CBS	QGS, QGD, QGB, QBD, QBS
Полевой транзистор	VGS, VGD, VSG, VSD, VDG, VDS	IG, IS, ID	CGS, CGD	QGS, QGD
Арсенид-галлиевый транзистор	VGS, VGD, VSG, VSD, VDG, VDS	IG, IS, ID	CGS, CGD	QGS, QGD
Источники тока или напряжения	V	I	Нет	Нет

После имени переменной в скобках указывается позиционное обозначение компонента. Например, напряжение затвор-исток МОП-транзистора M1 обозначается как VGS(M1).

При указании переменных, выводимых на графиках при проведении моделирования, возможно использовать следующие математические операции.

Арифметические операции

+ сложение;

– вычитание;

* умножение;

/ деление;

DIV целочисленное деление;

MOD остаток целочисленного деления.

Тригонометрические функции от действительных и комплексных величин (x – действительная, z – комплексная величина)

Exp(x) – экспонента;

Ln(x) – натуральный логарифм |x|;

Log(x) или Log10(x) – десятичный логарифм |x|;

Sin(x) – синус, x в радианах;

Cos(x) – косинус, x в радианах;

$\text{Tan}(x)$ – тангенс, x в радианах;
 $\text{Asin}(x)$ – арксинус;
 $\text{Acos}(x)$ – арккосинус;
 $\text{Atn}(x)$ или $\text{Arctan}(x)$ – арктангенс;
 $\text{Atan2}(y, x) = \text{Atn}(y/x)$;
 $\text{Sinh}(z)$ – гиперболический синус;
 $\text{Cosh}(z)$ – гиперболический косинус;
 $\text{Tanh}(z)$ – гиперболический тангенс;
 $\text{Coth}(z)$ – гиперболический котангенс.

Прочие функции от действительных и комплексных величин

$\text{ABS}(y)$ – абсолютное значение y ;
 $\text{SQRT}(y)$ – корень квадратный из модуля y ;
 $\text{SGN}(y)$ – знак числа y ;
 $\text{POW}(y, x)$ – степенная функция комплексных величин обозначаемая как y^x ;
 $\text{PWR}(y, x)$ – действительная часть степенной функции y^x ;
 $**$ – степенная функция, например $5**2=25$;
 $\text{PWRs}(y, x)$ – действительная часть степенной функции y^x ;
 $\text{FACT}(n)$ – факториал целого числа n ;
 $\text{JN}(n,z[,m])$ – функция Бесселя n -го порядка первого рода комплексного аргумента z , полученная суммированием первых m членов ряда; по умолчанию $m = 10$;
 $\text{J0}(z)$ – функция Бесселя нулевого порядка первого рода комплексного аргумента z , аналогичная $\text{JN}(0,z,10)$;
 $\text{J1}(z)$ – функция Бесселя первого порядка первого рода комплексного аргумента z , аналогичная $\text{JN}(1,z,10)$;
 $\text{YN}(n,z[,m])$ – функция Бесселя n -го порядка второго рода комплексного аргумента z , полученная суммированием первых m членов ряда; по умолчанию $m = 10$;
 $\text{Y0}(z)$ – функция Бесселя нулевого порядка второго рода комплексного аргумента z , аналогичная $\text{YN}(0,z,10)$;
 $\text{Y1}(z)$ – функция Бесселя нулевого порядка второго рода комплексного аргумента z , аналогичная $\text{YN}(1,z,10)$;
 $\text{Series}(n,n1,n2,z)$ – текущая сумма ряда комплексной функции $z = z(n)$ при изменении n от $n1$ до $n2$;
 RND – случайные числа на отрезке $[0, 1]$ с равномерным законом распределения;
 $\text{STP}(x)$ – функция единичного скачка, равная 1 при $x > 0$ и равная 0 при $x < 0$;
 $\text{IMPULSE}(y)$ — Импульсная функция, равная y и площадь которой равна 1;
 $\text{TABLE}(x,x_1,y_1,x_2,y_2,\dots,x_n,y_n)$ – табличная зависимость функции y от x . Переменная x должна быть определена как параметр с помощью директивы

.define. Задаются координаты точек (x_i, y_i) , в промежуточных точках используется линейная интерполяция. Если $x < x_i$, то $y = y_1$, если $x > x_n$, то $y = y_n$;

Waveform(имя файла,y) – импорт функции y из файла *имя файла*, имеющего стандартный формат Micro-CAP; в этот файл пользователя (User source) могут быть записаны дискретизированные результаты моделирования, если на закладке Save Waveforms команды **Properties (F10)** выбрать из списка имя переменной и ввести имя файла *.USR;

IMPORT(имя_файла,y) – импорт функции y из файла. Текстовый файл должен иметь формат SPICE или Micro-CAP; в него помещается таблица значений переменных, в качестве которых может быть время (T), частота (F), напряжение источника напряжений ($V(\text{имя источника})$), ток источника тока ($I(\text{имя источника})$), и выражение для y ;

SUM(y,x[,start]) – текущий интеграл от переменной y по переменной x ; начальное значение x равно *start*;

SD(y[,start]) – текущий интеграл от переменной y по времени T при анализе переходных процессов, по частоте F при АС-анализе или по переменной DCINPUT1 при DC-анализе; начальное значение независимой переменной равно *start*;

DD(y) – производная y по времени T при анализе переходных процессов, по частоте F при АС-анализе частотных характеристик и по переменной DCINPUT1 при DC-анализе по постоянному току;

RMS(y[,start]) – текущее среднеквадратичное отклонение переменной y при интегрировании по времени T при анализе переходных процессов, по частоте F при АС-анализе частотных характеристик и по переменной DCINPUT1 при DC-анализе по постоянному току; начальное значение независимой переменной равно значению *start*;

AVG(y[,start]) – текущее среднее значение переменной y при интегрировании по времени T при анализе переходных процессов, по частоте F при АС-анализе частотных характеристик и по переменной DCINPUT1 при DC-анализе по постоянному току; начальное значение независимой переменной равно значению *start*;

DEL(y) – приращение процесса $y(t)$ относительно предыдущей точки при расчете переходных процессов. Производная рассчитывается как отношение двух таких операторов, например производная dy/dt равна DEL(y)/DEL(t);

SDT(y) – текущий интеграл процесса $y(t)$ относительно времени T, начиная от $T = T_{\min}$;

DDT(y) – производная процесса $y(t)$ относительно времени T;

DIFA(u,v[,d]) – сравнение значений двух функций u и v во всех дискретных точках при расчете переходных процессов. DIFA присваивается значение 1, если во всех точках абсолютное значение разности функций меньше величины d , в противном случае присваивается 0. Параметр d необязательный, по умолчанию полагается $d = 0$;

$DIFD(u, v, [d])$ – сравнение значений двух логических u и v во всех дискретных точках при расчете переходных процессов. DIFA присваивается значение 1, если во всех точках значения функций отличаются друг от друга, в противном случае присваивается 0. В течение первых d секунд после начала расчета переходных процессов сравнение не проводится. Параметр d необязательный, по умолчанию полагается $d = 0$;

$DB(z)$ – величина в децибелах, равная $20 \cdot \log(|z|)$;

$RE(z)$ – действительная часть z ;

$IM(z)$ – мнимая часть z ;

$MAG(z)$ – модуль z . При построении графиков допустимо просто указать z ;

$PH(z)$ – фаза z в градусах;

$GD(z)$ – групповое время запаздывания.

ПРИЛОЖЕНИЕ 2 – НЕКОТОРЫЕ СООБЩЕНИЯ ОБ ОШИБКАХ, ВЫДАВАЕМЫЕ ПРОГРАММОЙ MICROCAP

В этом Приложении приведены некоторые сообщения об ошибках, возникающие при моделировании в программе MicroCAP 7, а также их краткое толкование. Список сообщений об ошибках упорядочен по алфавиту.

1. Can't find subckt Не найдена макромодель

Ошибка возникает, когда в библиотеке электрорадиоэлементов MicroCAP отсутствует макромодель, используемая в схеме. Типичная причина появления такой ошибки – использование демонстрационной версии MicroCAP, в которой существенно сокращена библиотека элементов по сравнению с полнофункциональной версией. См. раздел «Сведения для студентов, обучающихся с применением дистанционных технологий». Для устранения ошибки следует воспользоваться полнофункциональной версией программы.

2. Can't use '...' in function 'D' on arg. no. '1'. Нельзя использовать '...' в функции 'D' в качестве первого аргумента.

Обычно такое сообщение возникает как следствие неправильно указанной информации в таблице диалогового окна Transient Analysis Limits в столбце Y Expression. Типичная причина – на электрической схеме забыли указать имя контрольной точки, а в таблице диалогового окна Transient Analysis Limits сослались на эту точку. Возможен вариант с разным написанием имени контрольной точки на схеме и в таблице. Для устранения ошибки следует дать правильное имя контрольной точке на электрической схеме. См. прием №5 раздела «Типовые приемы работы в MicroCAP...».

3. Duplicate definition of part Повторное позиционное обозначение элемента

Ошибка возникает когда на электрической схеме присутствует два или более элементов, имеющих одинаковое позиционное обозначение. Например, два разных резистора с позиционным обозначением R1. Для устранения ошибки следует сменить повторное позиционное обозначение.

4. Expecting '...' but found '...' after value '...'. Должно быть '...', однако найдено '...' после величины '...'.

Сообщение, связанное с ошибкой описания макромодели элемента, в частности, мультиплексора или триггера. Макромодель представляет собой определенный текст на входном языке формата SPICE. Структура описания макромодели жестко регламентирована в плане следования резервированных слов, значений аргументов и т.д. Любое неквалифицированное вмешательство в описание макромодели вызывает сообщение об ошибке при запуске на моделирование. Для устранения ошибки следует либо переписать папку LIBRARY с другого компьютера, либо заново переустановить MicroCAP.

5. Extra text '...' found. Найден дополнительный текст '...'.

Ошибка может возникать в самых различных ситуациях. Общий смысл этого сообщения – программой найден какой-либо текст, назначение которого не определено. Типичная причина появления такой ошибки – на схеме присутствует источник цифрового сигнала, однако текстовое описание сигнала в диалоговом окне свойств источника не приведено. Для устранения ошибки следует на входном языке программы MicroCAP описать цифровой сигнал. См. прием №2 «Типовые приемы работы в MicroCAP...».

6. Failed to converge in specified number of iterations in time=... . *Прекращение сходимости за указанное число итераций в момент времени*

Ошибка, возникающая при работе вычислительных алгоритмов программы MicroCAP. Типичная причина возникновения – неправильно изображенная электрическая схема. В частности, для мультиплексоров или триггеров может иметь место замыкание их выходов. Для устранения ошибки следует внимательно проверить изображенную на экране электрическую схему, найти неправильный фрагмент и перерисовать его.

7. Illegal character in label '...'. Непредусмотренная буква в метке '...'. .

Ошибка возникает, если в позиционных обозначениях ЭРЭ на схеме используются буквы русского алфавита. Возможно также появление такого сообщения, если в строках ввода диалогового окна Transient Analysis Limits имеются недопустимые значения, например, текстовое значение вместо цифрового. Для устранения ошибки следует внимательно проверить все введенные значения и исправить, вызывающие это сообщение.

8. Illegal time range. Непредусмотренный временной диапазон.

Ошибка возникает при неправильном указании временного диапазона в диалоговом окне Transient Analysis Limits. Для устранения ошибки следует исправить введенное значение.

9. Invalid value '...'. Непредусмотренное значение '...'. .

Ошибка возникает, когда введенное пользователем значение не входит в область определения какого-либо параметра. Например, такое сообщение возникнет, если в описании одноразрядного цифрового сигнала указать значение FF. Для устранения ошибки следует проверить допустимость введенных Вами значений и исправить недопустимые.

10. Must specify Part Должно быть Элемент

См. ошибку №4.

11. No label. Нет метки.

Ошибка описания цифрового источника сигнала, где используются метки для многократного повтора программного блока. Ошибка появляется в случае разного написания одной и той же метки или ее отсутствия в программном блоке. Для устранения ошибки следует верно составить программный блок. См. прием №2 «Типовые приемы работы в MicroCAP...».

12. No Model Name Given. Математической модели ЭРЭ не дано наименование.

Ошибка, характерная для размещения на поле чертежа многопараметрических ЭРЭ, описание которых содержится в математических моделях. Для

устранения ошибки следует указать имя модели, если оно уже есть – уточнить правильность написания.

13. Stimulus pin count does not match format statement. *Число выводов источника сигнала не соответствует формату описания сигнала.*

Ошибка описания цифрового источника сигнала, когда значения параметра FORMAT (система счисления) не соответствуют информации указанной в строке COMMAND (описание сигнала). Например, ошибка может возникнуть если в строке FORMAT указано значение 1 (двоичная система счисления), а в описании сигнала используются значения АВ. Для устранения ошибки следует исправить информацию либо в строке FORMAT, либо в строке COMMAND. См. прием №2 «Типовые приемы работы в MicroCAP...».

14. Unknown command found ‘...’ in stim. *Найдена неизвестная команда ‘...’ в описании источника сигнала.*

Ошибка описания цифрового источника сигнала, когда встречается неизвестный программе MicroCAP оператор в программном блоке. Типичная причина появления ошибки – неправильно записанный пользователем оператор. Для устранения ошибки следует исправить такой оператор в соответствии с правилами, изложенными в приеме №2 «Типовые приемы работы в MicroCAP...».

15. Unknown identifier ‘...’ in ‘...’. *Неизвестный идентификатор ‘...’ в ‘...’.*

Ошибка возникает, когда в символических выражениях или позиционных обозначениях ЭРЭ встречается неизвестный (нерезервированный) в программе MicroCAP символ, например, буква русского алфавита. Для устранения ошибки следует исправить неправильно введенное значение.

**ПРИЛОЖЕНИЕ 3 – ПРИМЕР ОФОРМЛЕНИЯ ОТЧЕТА ПО
ЛАБОРАТОРНОЙ РАБОТЕ**

Министерство образования и науки РФ

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

КАФЕДРА КОНСТРУИРОВАНИЯ И ПРОИЗВОДСТВА РАДИОАППАРАТУРЫ (КИПР)

ЛАБОРАТОРНАЯ РАБОТА №1

«СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ»
по дисциплине «Схемотехника компьютерных технологий»

Выполнил
студент группы 233-1
_____ А.В.Хвостов
15 марта 2012 г.

Проверил
доцент каф. КИПР
_____ Д.В.Озёркин
« » _____ 2012 г.

Томск, 2012

Цели работы

1. Знакомство с основными этапами синтеза комбинационных логических устройств.
- 2) Изучение способов минимизации логических функций, представленных в совершенной дизъюнктивной форме.
- 3) Приобретение навыков перехода в логический базис элементов И-НЕ, ИЛИ-НЕ.
- 4) Построение структурных схем логических устройств по заданным функциям алгебры логики.

Лабораторное задание

Условие. Требуется для заданной таблично функции (таблица 1):

- построить структурную схему логического устройства на основе совершенной дизъюнктивной нормальной формы;
- построить структурную схему логического устройства на основе минимальной дизъюнктивной нормальной формы;
- построить структурную схему логического устройства в базисе логических элементов И-НЕ;
- построить структурную схему логического устройства в базисе логических элементов ИЛИ-НЕ.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Таблица 1 – Таблица истинности логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Выполнение.

I этап. Синтез логического устройства на основе СДНФ.

Совершенная дизъюнктивная нормальная форма таблично заданной функции имеет вид:

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

Аппаратная реализация функции $f(x_3, x_2, x_1)$ в СДНФ потребует три инвертора (для выработки сигналов $\bar{x}_3, \bar{x}_2, \bar{x}_1$); четыре элемента 3И; один элемент 4ИЛИ. Структурная схема логического устройства на основе СДНФ, подготовленная в программе MicroCAP, представлена на рисунке 1.

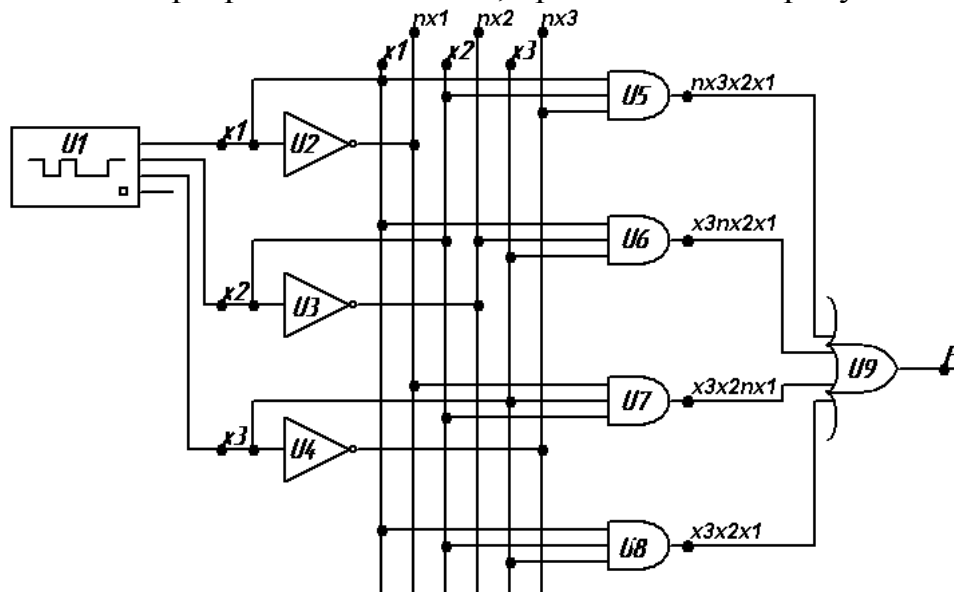


Рисунок 1 – Структурная схема логического устройства на основе СДНФ

Временная диаграмма логической функции $f(x_3, x_2, x_1)$ приведена на рисунке 2. Сравнивая ее с таблицей истинности (таблица 1, правый столбец) приходим к выводу об адекватности проведенного моделирования.

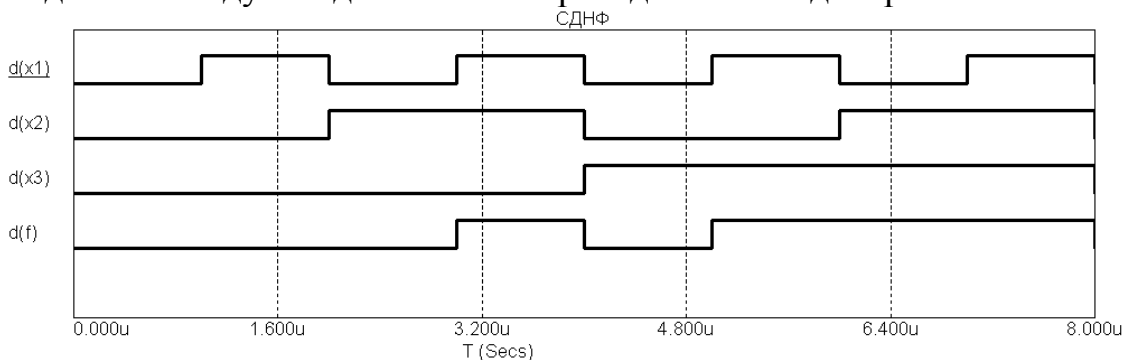


Рисунок 2 – Временная диаграмма работы схемы в СДНФ

II этап. Синтез логического устройства на основе МДНФ.

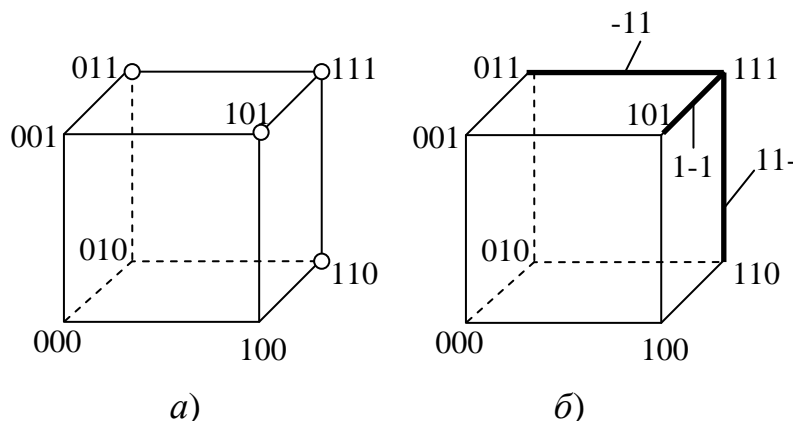
Для проведения второго этапа исследования воспользуемся логической функцией в СДНФ, полученной на I этапе:

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

Графический образ такой логической функции представлен в виде куба на рисунке 3.

Множество нулевых кубов образуют нулевой кубический комплекс $\mathbf{K}_0 = (011, 101, 110, 111)$. Множество единичных кубов образуют единичный кубический комплекс $\mathbf{K}_1 = (-11, 1-1, 11-)$. Двоичного кубического комплекса для

данной функции не существует, поскольку нет ни одной грани, полностью образованной единичными кубами.



а) нулевой кубический комплекс; б) единичный кубический комплекс
Рисунок 3 – Геометрическое представление логической функции

Кубический комплекс логической функции имеет вид:

$$\mathbf{K}(f) = (011, 101, 110, 111, -11, 1-1, 11-).$$

Нулевой кубический комплекс (рисунок 3, а) включает все вершины куба, поэтому образует покрытие функции:

$$\mathbf{\Pi}_1(z) = \mathbf{K}_0 = (011, 101, 110, 111).$$

Все вершины куба включаются также в единичный кубический комплекс \mathbf{K}_1 (рисунок 3, б), поэтому он также образует покрытие функции:

$$\mathbf{\Pi}_2(z) = \mathbf{K}_1 = (-11, 1-1, 11-).$$

Перебирая сочетания кубов различных рангов можно получить следующие покрытия функции:

$$\mathbf{\Pi}_3(z) = (011, 1-1, 11-);$$

$$\mathbf{\Pi}_4(z) = (101, -11, 11-);$$

$$\mathbf{\Pi}_5(z) = (110, -11, 1-1);$$

$$\mathbf{\Pi}_6(z) = (111, -11, 1-1, 11-) \text{ и т.д.}$$

Среди перечисленных выше (и существующих вообще) покрытий выражение $\mathbf{\Pi}_2(z) = \mathbf{K}_1 = (-11, 1-1, 11-)$ имеет минимальную цену.

Для покрытия $\mathbf{\Pi}_2(z)$ имеем:

$$\mathbf{\Pi}_{\mathbf{K}_1} = 3 - 1 = 2; \mathbf{\Pi}_{\mathbf{K}_2} = 3 - 1 = 2; \mathbf{\Pi}_{\mathbf{K}_3} = 3 - 1 = 2; \mathbf{\Pi}_{\mathbf{\Pi}} = \sum \mathbf{\Pi}_{\mathbf{K}_i} = 2 + 2 + 2 = 6.$$

Все остальные покрытия имеют бóльшую цену. Следовательно, логическая функция $f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2$, соответствующая покрытию $\mathbf{\Pi}_2(z)$, является минимальной дизъюнктивной нормальной формой (МДНФ).

Проверим полученный результат другим способом – минимизацией с помощью карты Вейча. Карта Вейча для исходной функции в СДНФ представлена на рисунке 4. МДНФ функции:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

Аппаратная реализация логической функции в МДНФ потребует три элемента 2И и один элемент 3ИЛИ. Структурная схема логического устройства на основе МДНФ, подготовленная в программе MicroCAP, представлена на рисунке 5.

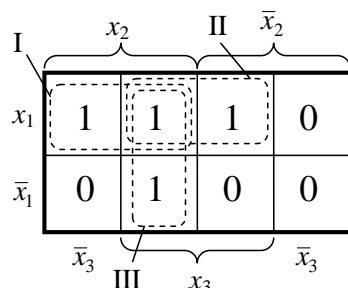


Рисунок 4 – Карта Вейча для исходной функции в СДНФ

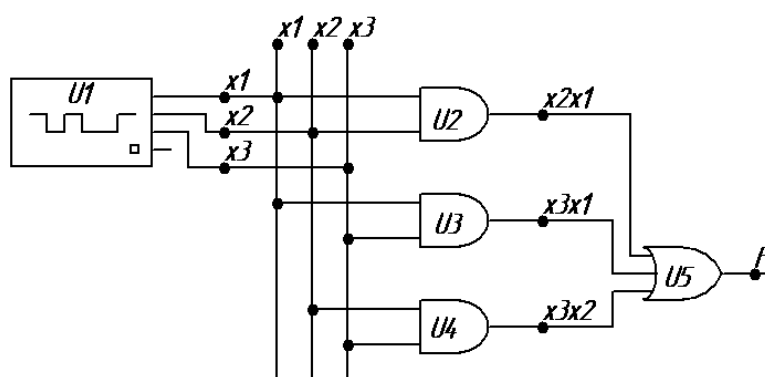


Рисунок 5 - Структурная схема логического устройства на основе МДНФ

Результаты анализа во временной области представлены на рисунке 6. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

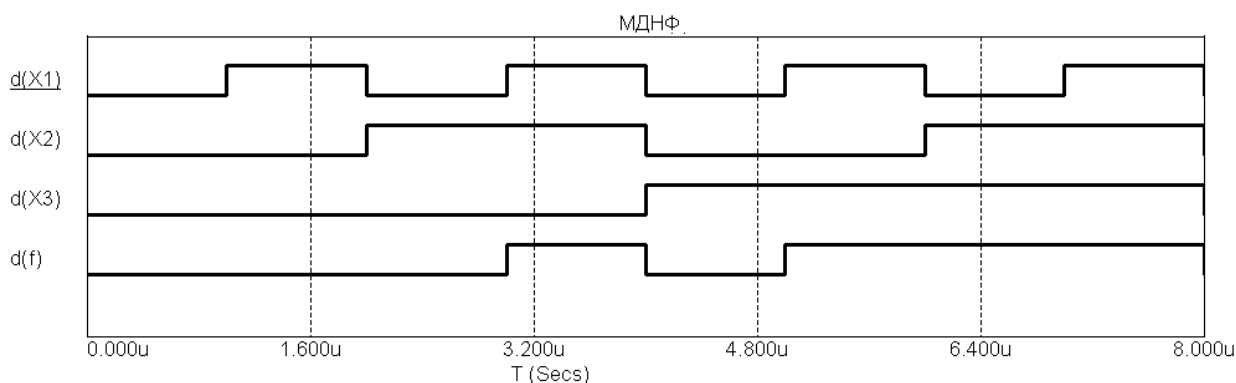


Рисунок 6 – Временная диаграмма работы схемы в МДНФ

III этап. Синтез логического устройства в базе И-НЕ.

Для проведения третьего этапа исследования воспользуемся логической функцией в МДНФ, полученной на II этапе:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

Для перехода к базису И-НЕ выполняются следующие действия:

- дважды инвертируется правая часть выражения:

$$f(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2;$$

- проводится преобразование по теореме де Моргана:

$$f(x_3, x_2, x_1) = \overline{x_2 \cdot x_1 \cdot x_3 \cdot x_1 \cdot x_3 \cdot x_2}.$$

Следовательно, преобразованная логическая функция в базисе логических элементов И-НЕ выглядит как:

$$f_{\text{И-НЕ}}(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1 \cdot x_3} \cdot \overline{x_1 \cdot x_3 \cdot x_2}}.$$

Аппаратная реализация такой логической функции потребует три элемента 2И-НЕ и один элемент 3И-НЕ. Структурная схема логического устройства в базисе И-НЕ, подготовленная в программе MicroCAP, представлена на рисунке 7.

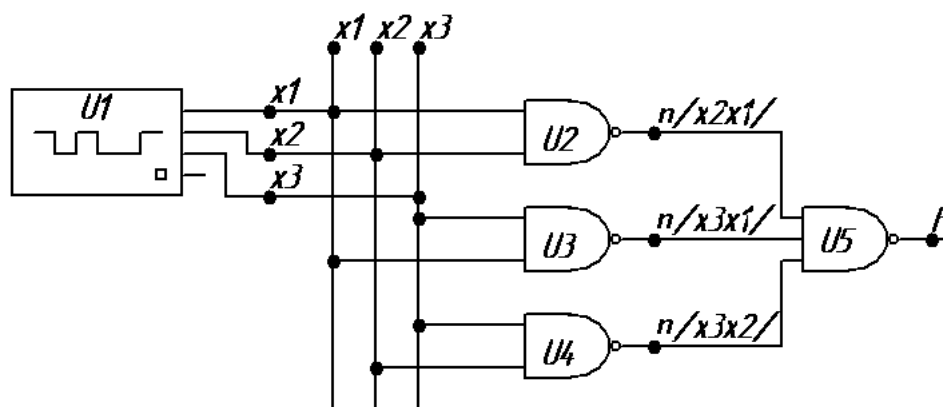


Рисунок 7 - Структурная схема логического устройства в базисе И-НЕ

Результаты анализа во временной области представлены на рисунке 8. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

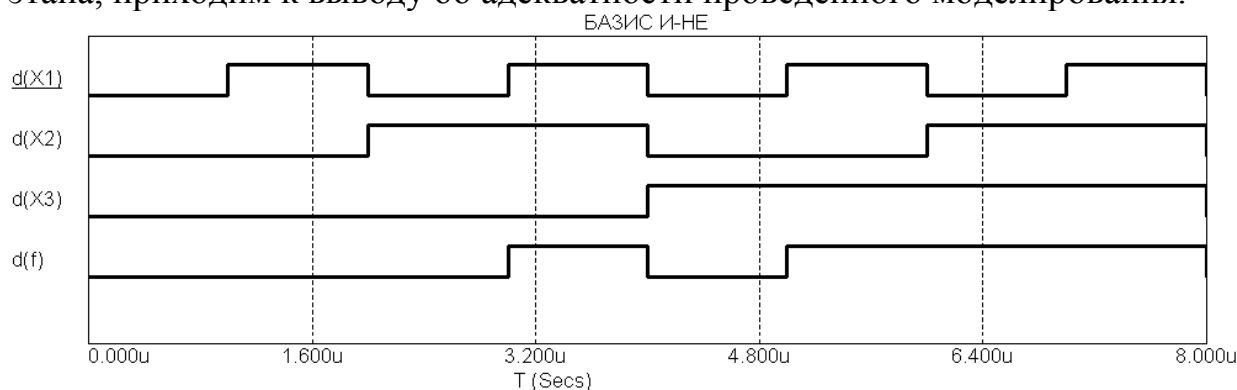


Рисунок 8 – Временная диаграмма работы схемы в базисе И-НЕ

IV этап. Синтез логического устройства в базисе ИЛИ-НЕ.

Для проведения четвертого этапа исследования воспользуемся логической функцией в МДНФ, полученной на II этапе:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

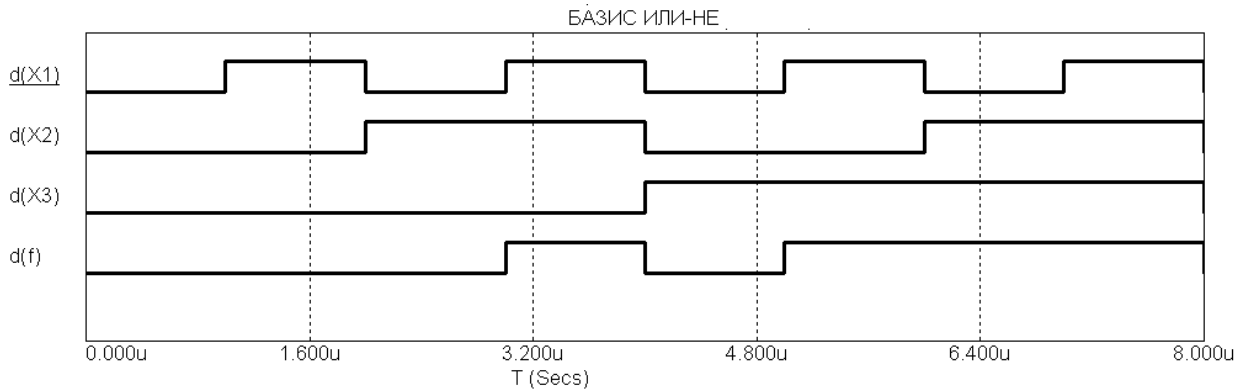


Рисунок 10 - Временная диаграмма работы схемы в базисе ИЛИ-НЕ

Выводы:

1. Существует несколько способов представления логических функций: в виде таблиц истинности, в виде кубических комплексов, в виде булевых выражений и т.д.

2. Совершенная дизъюнктивная нормальная форма в силу своего громоздкого вида не используется для практических целей цифровой схемотехники, она лишь служит промежуточным вариантом для перехода от табличного представления к булевому выражению.

3. Переход к тому или иному базису (И-НЕ, ИЛИ-НЕ) определяется в каждом конкретном случае схемотехнологией цифровых интегральных схем, на которых предполагается аппаратно реализовать логическое устройство.

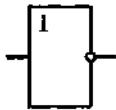
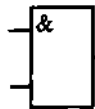
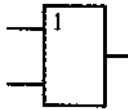
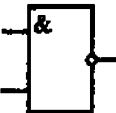
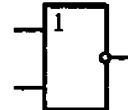
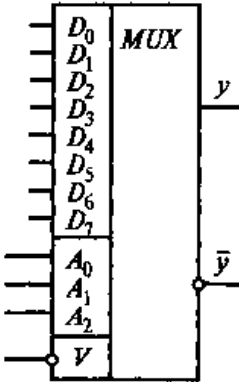
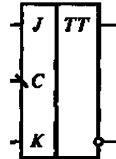
Ответы на контрольные вопросы

1. *В чем заключаются этапы синтеза логического устройства?* На первом этапе функцию, заданную в словесной, табличной или другой формах, представляют в виде логического выражения с использованием некоторого базиса. Дальнейшие этапы сводятся к получению минимальных форм функций, обеспечивающих при синтезе наименьшее количество электронного оборудования и рациональное построение функциональной схемы устройства.

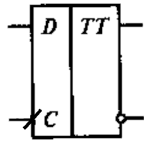
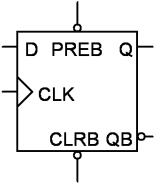
2.

**ПРИЛОЖЕНИЕ 4 – СООТВЕТСТВИЕ УСЛОВНЫХ ГРАФИЧЕСКИХ
ОБОЗНАЧЕНИЙ НЕКОТОРЫХ ЭРЭ В РОССИИ И ЗА РУБЕЖОМ**

Таблица

НАЗВАНИЕ ЭРЭ	ОБОЗНАЧЕНИЯ, ПРИНЯТЫЕ В ЕСКД		ОБОЗНАЧЕНИЯ, MICRO
	СИМВОЛ ПОЗИЦИОННОГО ОБОЗНАЧЕНИЯ	УСЛОВНОЕ ГРАФИЧЕСКОЕ ОБОЗНАЧЕНИЕ	СИМВОЛ ПОЗИЦИОННОГО ОБОЗНАЧЕНИЯ
ИНВЕРТОР НЕ	<i>DD</i>		<i>U</i>
ЭЛЕМЕНТ 2И	<i>DD</i>		<i>U</i>
ЭЛЕМЕНТ 2ИЛИ	<i>DD</i>		<i>U</i>
ЭЛЕМЕНТ 2И-НЕ	<i>DD</i>		<i>U</i>
ЭЛЕМЕНТ 2ИЛИ-НЕ	<i>DD</i>		<i>U</i>
МУЛЬТИПЛЕКСОР «8-1»	<i>DD</i>		<i>X</i>
JK-ТРИГГЕР С ИНВЕРСНЫМ ДИНАМИЧЕСКИМ ВХОДОМ СИНХРОНИЗАЦИИ	<i>DD</i>		<i>U</i>

Окончание таблицы

<p><i>D</i>-ТРИГГЕР С ПРЯМЫМ ДИНАМИЧЕСКИМ ВХОДОМ СИНХРОНИЗАЦИИ</p>	<p><i>DD</i></p>		<p><i>U</i></p>	
---	------------------	---	-----------------	---

Учебное издание

Озёркин Денис Витальевич

Схемотехника компьютерных технологий

Компьютерный лабораторный практикум

Формат 60×84 1/16. Усл. печ. л. 10,9

Тираж 50 экз. Заказ

Отпечатано в Томском государственном университете
систем управления и радиоэлектроники.

634050, Томск, пр. Ленина, 40. Тел. (3822) 533018.