

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

М. В. Качинский, В. Ю. Герасимович, А. В. Станкевич

**ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ
НА ИНТЕГРАЛЬНЫХ МИКРОСХЕМАХ.
КУРСОВОЕ ПРОЕКТИРОВАНИЕ**

*Рекомендовано УМО по образованию в области информатики
и радиоэлектроники в качестве пособия для специальности
1-40 02 02 «Электронные вычислительные средства»*

Минск БГУИР 2017

УДК 621.3.049.77-027.31(076)
ББК 32.844.1-02я73
К30

Рецензенты:

кафедра радиофизики и цифровых медиа технологий
Белорусского государственного университета
(протокол №12 от 27.04.2017);

главный научный сотрудник
государственного научного учреждения «Объединенный институт
проблем информатики Национальной академии наук Беларуси»,
доктор технических наук, профессор Г. И. Алексеев

Качинский, М. В.

К30 Проектирование цифровых устройств на интегральных микросхемах. Курсовое проектирование : пособие / М. В. Качинский, В. Ю. Герасимович, А. В. Станкевич. – Минск : БГУИР, 2017. – 66 с. : ил.
ISBN 978-985-543-378-2.

Содержит методические указания по выполнению курсового проекта по дисциплине «Основы проектирования электронных вычислительных средств», включая типовые задания и примеры их выполнения. В пособии рассматриваются также вопросы схемотехнического моделирования в среде проектирования Xilinx ISE.

УДК 621.3.049.77-027.31(076)
ББК 32.844.1-02я73

ISBN 978-985-543-378-2

© Качинский М. В., Герасимович В. Ю.,
Станкевич А. В., 2017
© УО «Белорусский государственный
университет информатики
и радиоэлектроники», 2017

Содержание

ВВЕДЕНИЕ	4
1 ЦЕЛИ И ЗАДАЧИ КУРСОВОГО ПРОЕКТИРОВАНИЯ	5
2 ОСНОВНЫЕ ТРЕБОВАНИЯ К КУРСОВОМУ ПРОЕКТУ	5
2.1 Тематика курсового проектирования.....	5
2.2 Задание по курсовому проектированию и исходные данные	6
2.3 Содержание и объем курсового проекта	6
2.4 Организация курсового проектирования и защита проекта	7
3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА	8
3.1 Общие указания.....	8
3.2 Особенности проектирования цифровых устройств на интегральных микросхемах	9
3.2.1 Типы выходных каскадов цифровых элементов микросхем малой и средней степени интеграции	9
3.2.2 Помехи по цепям питания в схемах цифровых устройств.....	15
3.2.3 Типовые ситуации при проектировании цифровых устройств на интегральных микросхемах	17
3.3 Примеры решения задач по проектированию цифровых устройств при выполнении заданий курсового проекта	21
4 СХЕМОТЕХНИЧЕСКОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ ПРОЕКТИРОВАНИЯ XILINX ISE	31
4.1 Основы работы в среде проектирования <i>Xilinx ISE</i>	31
4.1.1 Создание проекта.....	31
4.1.2 Симуляция моделируемой схемы с помощью создания файла тестового воздействия.....	39
4.1.3 Симуляция моделируемой схемы с помощью ручного задания значений входных сигналов	43
4.2 Примеры моделирования схем в среде проектирования <i>Xilinx ISE</i>	45
4.2.1 Комбинационный узел на основе мультиплексора.....	46
4.2.2 Пересчетное устройство	49
4.2.3 Сдвиговый регистр	51
4.2.4 Моделирование устройств на основе сумматоров.....	53
ПРИЛОЖЕНИЕ А (ОБЯЗАТЕЛЬНОЕ) ПЕРЕЧЕНЬ ТИПОВЫХ ЗАДАНИЙ КУРСОВОГО ПРОЕКТА	59
А.1 Дешифраторы, демультимплексоры, шифраторы, преобразователи кодов..	59
А.2 Мультиплексоры	61
А.3 Счетчики, пересчетные устройства.....	61
А.4 Регистры	63
А.5 Сумматоры и арифметическо-логические устройства (АЛУ).....	64
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	65

ВВЕДЕНИЕ

Дисциплина «Основы проектирования электронных вычислительных средств» занимает одно из ведущих мест в подготовке специалистов в области проектирования электронных вычислительных средств (ЭВС). При изучении этой дисциплины ставится задача формирования у студентов технического мышления, обучения будущих инженеров практическим навыкам проектирования цифровых устройств на интегральных микросхемах (ИМС). С целью закрепления этих навыков выполняется курсовой проект, который является необходимым условием для успешной работы над курсовыми проектами по другим дисциплинам, а также дипломным проектом.

Для проектирования ЭВС требуются специалисты, хорошо знающие принципы работы цифровых устройств, базовые элементы цифровой электроники, типовые схемы их включения, способы построения типовых цифровых узлов. При этом стать настоящим специалистом по разработке цифровых устройств невозможно без овладения основами цифровой схемотехники. Разработчик обязан понимать логику работы таких базовых компонентов цифровой схемотехники, как логические элементы, триггеры, регистры, дешифраторы, мультиплексоры, счетчики и т. д. Кроме того, он должен знать типовые схемы включения этих компонентов и основы их корректной работы в составе цифровых устройств. Такие знания необходимы, даже если разрабатывается устройство на базе микросхем с программируемой логикой или микропроцессоров.

Курсовой проект по дисциплине «Основы проектирования электронных вычислительных средств» посвящен основам цифровой схемотехники, ее основным методам, подходам и приемам. Он не претендует на то, чтобы охватить весь круг вопросов проектирования цифровых систем, но в то же время, позволяет достаточно глубоко освоить методы проектирования сравнительно простых цифровых устройств. После выполнения данного курсового проекта студент сможет как уверенно разбираться в работе готовых цифровых устройств, так и проектировать новые устройства и системы. Выполнение курсового проекта дает студенту тот необходимый минимум знаний и умений, который должен иметь и которым должен свободно и активно пользоваться каждый профессиональный разработчик цифровой аппаратуры.

В качестве учебного базиса для выполнения курсового проекта выбрано хорошо зарекомендовавшее себя функционально полное семейство микросхем SN74, разработанных в различные годы американской фирмой Texas Instruments и выпускаемых различными производителями электронных компонентов, например, *Fairchild/ON Semiconductor, Toshiba Semiconductor and Storage, Motorola* (аналоги – серии К555, КР1533, КР1554 и др. [1]–[4], отечественные аналоги выпускаются ОАО «ИНТЕГРАЛ» – ТТЛШ серии КР, ЭКР, ЭКФ1533XXXX, IN74LSXXXN (D, DW), КМОП серии IN74ACXXXN (D, DW), IN74ACTXXXN (D, DW), IN74HCXXXN (D, DW), IN74HCTXXXN (D, DW), IN74VHCXXXD (DW), IN74VHCTXXXD (DW), IN74LVXXXN (D, DW) <http://integral.by/ru/products/standartnye-cifrovye-logicheskie-ims>).

1 ЦЕЛИ И ЗАДАЧИ КУРСОВОГО ПРОЕКТИРОВАНИЯ

Курсовой проект по дисциплине «Основы проектирования электронных вычислительных средств» имеет следующие цели и задачи:

- закрепить, углубить и систематизировать теоретические знания, полученные при изучении основных разделов дисциплины;
- получить практические навыки самостоятельной работы при решении комплекса задач по проектированию цифровых устройств комбинационного и последовательностного типа с использованием микросхем малой и средней степени интеграции путем выполнения самостоятельной разработки по заданному индивидуальному заданию;
- научить пользоваться специальной, справочной и другой нормативно-технической литературой, действующими стандартами;
- получить навыки по оформлению текстовой и графической документации согласно требованиям государственных стандартов и стандарта предприятия СТП 01–2017 «Дипломные проекты (работы). Общие требования»;
- подготовить студента к выполнению курсовых проектов по другим дисциплинам специальности, а также к дипломному проектированию.

2 ОСНОВНЫЕ ТРЕБОВАНИЯ К КУРСОВОМУ ПРОЕКТУ

2.1 Тематика курсового проектирования

Курсовой проект по дисциплине «Основы проектирования электронных вычислительных средств» должен представлять собой самостоятельное решение комплекса задач по проектированию цифровых устройств комбинационного и последовательностного типа с использованием микросхем малой и средней степени интеграции. Тема курсового проекта: «Проектирование цифровых устройств на интегральных микросхемах».

В курсовом проекте необходимо выполнить пять заданий по разработке электрических принципиальных схем цифровых устройств различного типа на заданной элементной базе с заданным критерием оптимизации. В качестве критерия оптимизации используется условие минимального количества корпусов ИМС, выбранных для построения схемы цифрового устройства.

В качестве заданий курсового проекта рекомендуется разработка следующих цифровых устройств комбинационного и последовательностного типа:

- 1) дешифраторы, демультимплексоры, шифраторы, преобразователи кодов;
- 2) мультимплексоры;
- 3) счетчики и пересчетные устройства;
- 4) регистры;
- 5) сумматоры и арифметико-логические устройства.

Перечень типовых заданий курсового проекта приведен в приложении А.

2.2 Задание по курсовому проектированию и исходные данные

Индивидуальное задание по курсовому проектированию выдается руководителем проекта студентам очной формы обучения в первые две недели после начала семестра.

Задание оформляется на специальном бланке. Заданием по курсовому проектированию предусматривается разработка электрических принципиальных схем заданных цифровых устройств, их схемотехническое моделирование с целью подтверждения правильности проектирования. В задание включаются:

- 1) тема проекта;
- 2) сроки сдачи студентом законченного проекта;
- 3) исходные данные к проекту;
- 4) содержание пояснительной записки (перечень вопросов, подлежащих разработке);
- 5) перечень графического материала;
- 6) календарный график работы над проектом.

В качестве исходных данных для выполнения курсового проекта задается назначение разрабатываемого цифрового устройства и порядок его функционирования (выполняемая устройством функция).

В курсовом проекте должны быть проработаны следующие вопросы:

- анализ условия задания;
- обзор и анализ теоретических сведений, необходимых для выполнения проектирования устройства, выбор метода построения устройства;
- проектирование устройства – выбор типа используемых интегральных микросхем, разработка электрической принципиальной схемы в соответствии с условием минимального количества корпусов ИМС, использованных для построения цифрового устройства, описание работы устройства по принципиальной схеме;
- схемотехническое моделирование разработанной схемы устройства.

В состав графической части курсового проекта должны входить электрические принципиальные схемы устройств для каждого задания.

2.3 Содержание и объем курсового проекта

Курсовой проект должен содержать проектировочную часть, помещаемую в пояснительной записке, и графическую часть, оформленную комплектом схем.

При изложении материала в пояснительной записке рекомендуется придерживаться следующего его расположения:

- титульный лист;
- реферат;
- задание по курсовому проекту;
- содержание;
- введение;

- задание №1;
- задание №2;
- задание №3;
- задание №4;
- задание №5;
- параметры использованных в курсовом проекте микросхем;
- заключение;
- список использованных источников;
- приложения;
- ведомость документов.

Общий объем пояснительной записки должен составлять 25–35 страниц. Графическая часть должна содержать 5 листов формата А4 или А3 в зависимости от сложности схемы. Пояснительная записка и графический материал выполняются только с использованием средств вычислительной техники, оформляются в соответствии с требованиями, приведенными в СТП 01–2017.

2.4 Организация курсового проектирования и защита проекта

На выполнение курсового проекта по дисциплине «Основы проектирования электронных вычислительных средств» отводится примерно 13 учебных недель. Работа над курсовым проектом является самостоятельной работой студента, проводимой под руководством и контролем руководителя проектирования. В установленное графиком время студент консультируется у своего руководителя. Время работы над курсовым проектом разбивается на 5 этапов, в соответствии с количеством подлежащих разработке устройств. Студент обязан после каждого этапа проектирования представлять руководителю выполненные расчеты, схемотехнические решения и другие материалы на проверку. Руководитель проверяет сделанную работу, указывает ошибки, разъясняет недоработанные места и дает рекомендации по их исправлению. Объем выполненной студентом работы по каждому этапу оценивается руководителем проектирования в процентах от общего объема проектирования. Явка на опроцентовку студентов строго обязательна. Законченный курсовой проект, подписанный студентом, представляется руководителю в срок, установленный календарным планом. Выполненный курсовой проект может быть сдан на проверку руководителю до срока, указанного в календарном плане. Руководитель проверяет полноту представленных материалов, соответствие их заданию, выясняет готовность проекта к защите и по согласованию со студентом устанавливает дату защиты. В случае неготовности курсового проекта либо необходимости внести поправки студенту предоставляется дополнительный срок (с конкретным указанием требуемых исправлений). После внесения исправлений и доработки курсового проекта студент повторно представляет руководителю курсовой проект для проверки и защиты, но не позднее, чем за три дня до защиты. Устранение недостатков, отмеченных руководителем, контролируется комиссией в процессе защиты курсового проекта.

Защита курсового проекта осуществляется перед комиссией и включает в себя доклад в течение 5–10 мин. и ответы на вопросы членов комиссии. В докладе студент должен изложить задачи проектирования и их реализацию, привести обоснование принятых технических решений, кратко охарактеризовать каждый лист графического материала. Принятые в курсовом проекте технические решения должны сопровождаться выводами. Доклад должен быть дополнен электронной презентацией, показывающей результаты схемотехнического моделирования разработанных в курсовом проекте схем устройств. В заключении студент должен отразить степень соответствия разработанного проекта требованиям задания на проектирование. Студенту на защите могут задаваться любые вопросы по теме курсового проекта. По выступлению студента и его ответам на вопросы комиссия судит о его умении правильно и доходчиво излагать результаты своей работы, поэтому выступление рекомендуется подготовить заранее. Комиссия оценивает результаты защиты каждого курсового проекта и принимает решение об отметке, учитывая при этом полноту представленного материала, обоснованность принятых решений, содержание доклада, ответы на вопросы, соблюдение требований стандартов к графическим и текстовым документам.

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА

3.1 Общие указания

Методические указания преследуют цель облегчить студентам выполнение проекта, намечают основные направления работы над курсовым проектом. При этом не исключается самостоятельный выбор студентом путей решения тех или иных задач курсового проекта, применения других обоснованных методов проектирования.

Рекомендуется начинать проектирование с составления плана работы над проектом, в котором необходимо предусмотреть изучение литературы, повторение некоторых дисциплин, разработку электрических принципиальных схем заданных цифровых устройств, описание работы устройств по принципиальной схеме, выполнение схемотехнического моделирования разработанных схем устройств, оформление пояснительной записки, выполнение схем и т. д. Студент самостоятельно составляет развернутый календарный план выполнения курсового проекта с учетом контрольных точек, оговоренных заранее руководителем проекта. В этом случае обеспечивается самоконтроль, равномерное распределение нагрузки по этапам проектирования, что оказывает положительное влияние на качество разработки и облегчает работу.

Разработка устройства должна оцениваться с точки зрения эффективности затрат. В качестве критерия оптимизации используется условие минимального количества корпусов интегральных микросхем, выбранных для построе-

ния схемы цифрового устройства. В этом случае студент должен просмотреть всю заданную серию, выбрать различные методы минимизации (объединение единиц на картах Карно, объединение нулей, минимизация систем логических функций, построение скобочной формы логической функции и т. д.), которые позволят преобразовать минимальные ДНФ к виду, требующему минимального числа корпусов ИМС. Для лучшего понимания этих положений необходимо рассмотреть особенности проектирования цифровых устройств на интегральных микросхемах [2]–[8].

3.2 Особенности проектирования цифровых устройств на интегральных микросхемах

3.2.1 Типы выходных каскадов цифровых элементов микросхем малой и средней степени интеграции

Цифровые элементы, реализуемые в микросхемах малой и средней степени интеграции (логические элементы, триггеры, буферные элементы, типовые комбинационные и последовательностные узлы) могут иметь выходы следующих типов:

- 1) логические;
- 2) с тремя состояниями;
- 3) с открытым коллектором (стоком).

Наличие различных типов выходов объясняется различными условиями работы элементов в логических цепях, в магистрально-модульных структурах микропроцессорных средств и т. д.

Логический выход [6]. Логический выход формирует два уровня выходного напряжения U_0 и U_1 . Он имеет малое значение выходного сопротивления. Это делается для того, чтобы логический выход мог развивать большие токи для перезаряда емкостных нагрузок и, следовательно, получения высокого быстродействия элемента. Такой тип выхода имеют большинство элементов, используемых в цифровых устройствах.

Схемы логических выходов элементов ТТЛ и КМОП реализуются по принципу двухтактных каскадов. Оба фронта выходного напряжения формируются с участием активных транзисторов, работающих противофазно, что обеспечивает малые выходные сопротивления при любом направлении переключения выхода (рисунок 1, а).

Логические выходы имеют две особенности, о которых необходимо помнить при проектировании цифровых устройств на ИМС.

Первая особенность таких выходов состоит в том, что их нельзя соединять параллельно. Это создает логическую неопределенность, т. к. в точке соединения выходов, находящихся в различных логических состояниях, не будет нормального уровня напряжения. Кроме того, в этом случае возникает уравнительный ток, который из-за малых значений выходных сопротивлений может

достигать достаточно большой величины, способной вывести из строя элементы выходной цепи.

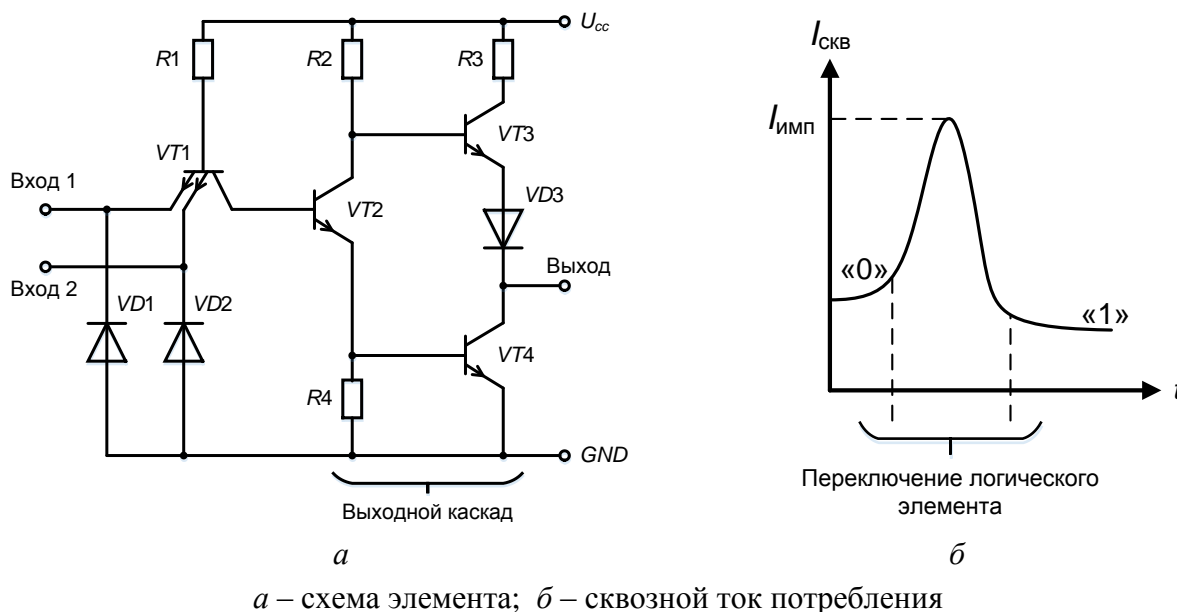


Рисунок 1 – Базовый логический элемент И-НЕ стандартной серии ТТЛ

Вторая особенность логического выхода заключается в том, что при переключении из одного логического состояния в другое через оба транзистора от источника питания на общую точку («землю») протекают короткие импульсы тока. В статических состояниях таких токов нет, т. к. транзисторы $VT3$ и $VT4$ (см. рисунок 1, а) работают в противофазе, и один из них всегда закрыт. Однако в переходном процессе из-за некоторой несинхронности переключения транзисторов возникает кратковременная ситуация, при которой проводят оба транзистора. При этом через транзисторы протекает короткий импульс сквозного тока значительной величины (см. рисунок 1, б).

Выход с тремя состояниями [6]. Выход с тремя состояниями (типа ТС или Z) кроме логических состояний 0 и 1 имеет состояние «отключено» (называемое Z-состоянием), в котором выходной ток пренебрежимо мал. В это состояние выход элемента переводится специальным управляющим сигналом, который обычно обозначается как EZ (*Enable Z-state*) или OE (*Output Enable*).

В качестве примера на рисунке 2 показана схема логического элемента (ЛЭ) ТТЛ, дополненная управляющим входом EZ . Управляющий вход EZ подключен к базе дополнительного транзистора $VT2'$, коллектор которого подключен к нагрузке $R2$ транзистора $VT2$. При отсутствии разрешения $EZ = 0$ транзистор $VT2'$ закрыт, позволяя нормально работать выходному каскаду элемента. При наличии разрешения $EZ = 1$ напряжение на коллекторе $VT2'$ близко к нулю и оба транзистора выходного каскада $VT3$ и $VT4$ заперты. Выходное сопротивление запертых транзисторов велико и выход логического элемента находится в третьем «отключенном» состоянии. Поэтому это состояние ЛЭ часто называют *высокоимпедансным*.

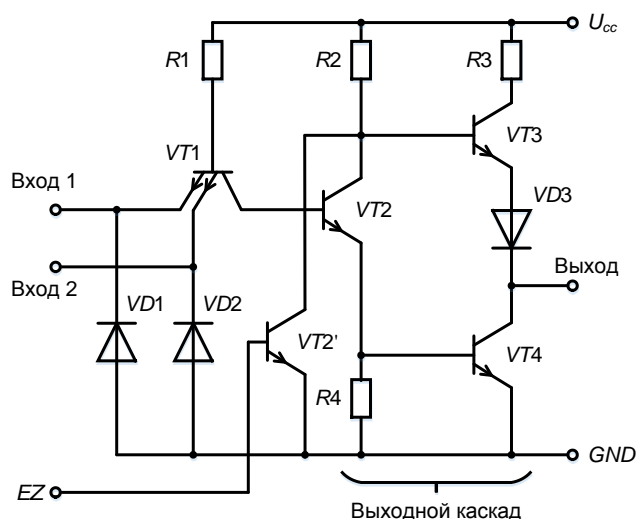


Рисунок 2 – Схема логического элемента ТТЛ с выходом типа ТС

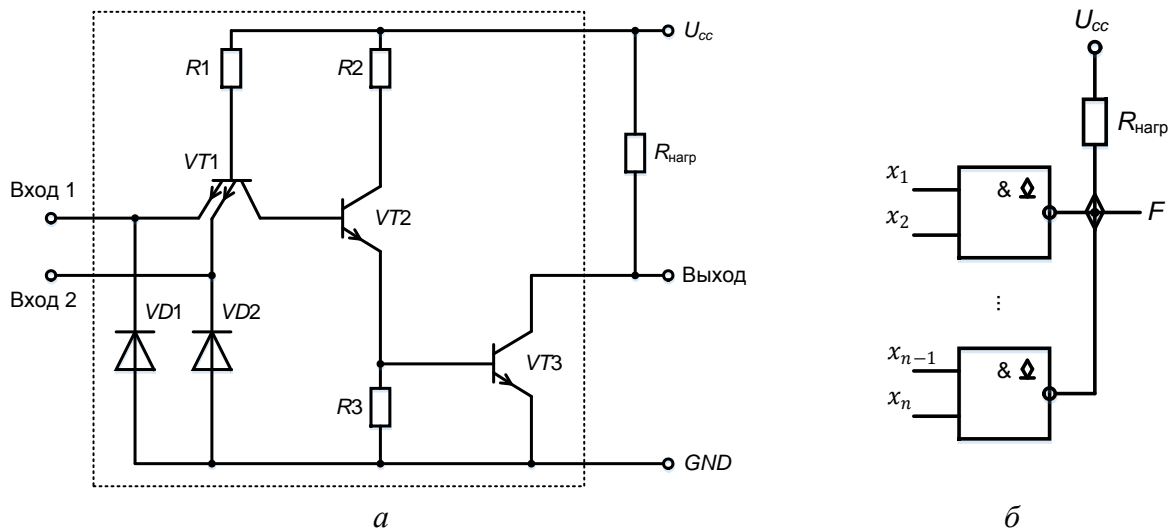
Сигнал EZ управления выходом типа ТС переводит выходной каскад элемента в Z -состояние. В отличие от него сигнал OE управляет нормальным режимом выхода элемента. При наличии разрешения ($OE = 1$) выход элемента работает как обычно, т. е. находится в состоянии 0 или 1, в зависимости от выполняемой элементом логической операции. При отсутствии разрешения ($OE = 0$) выход переходит в состояние «отключено». Выходы типа ТС обычно отмечаются в обозначениях элементов значком треугольника или буквой Z .

Выходы типа ТС можно соединять параллельно при условии, что в любой момент времени активным является только один из них. В этом случае отключенные выходы не мешают активному выходу формировать нормальные значения сигнала в точке соединения выходов. Эта возможность позволяет применять элементы с выходом типа ТС в магистрально-модульных структурах микропроцессорных и других систем, где несколько источников сигналов поочередно пользуются одной и той же линией связи.

Выход с открытым коллектором (стоком) [6]. Выход с открытым коллектором (типа ОК) или стоком (типа ОС) имеет выходной каскад, заканчивающийся одиночным транзистором, коллектор или сток которого не соединен ни с какими цепями внутри элемента (рисунок 3, а).

Транзистор управляется схемой элемента так, что может находиться в насыщенном (для МОП-транзистора просто открытом) или запертом состоянии. Насыщенное (открытое) состояние соответствует логическому нулю, запертое – логической единице. Насыщенное состояние транзистора обеспечивает на выходе напряжение U_0 (малое напряжение насыщения «коллектор-эмиттер», $U_{кэп}$). Запирание транзистора какого-либо уровня напряжения на выходе элемента не создает, т. к. коллектор транзистора не подключен ни к каким цепям внутри элемента. При этом выход имеет фактически неизвестный «плавающий» потенциал. Поэтому при запирании транзистора для формирования высокого уровня напряжения U_1 на выходе элемента с открытым коллектором требуется подключать внешний нагрузочный резистор $R_{нагр}$ (или другую нагрузку), со-

единенный с источником питания (см. рисунок 3, а). В обозначениях элементов с выходом типа ОК (OC) после символа функции ставится ромб с чертой снизу (см. рисунок 3, б).



а – схема элемента; б – параллельное соединение выходов типа ОК

Рисунок 3 – Логический элемент ТТЛ с выходом типа ОК

Несколько выходов типа ОК можно соединять параллельно, подключая их к общему для всех выходов внешнему резистору (см. рисунок 3, б). При этом можно получить режим поочередной работы элементов на общую линию, если активным сделать только один элемент, а выходные транзисторы всех остальных элементов перевести в запертое состояние. Особенностью выходов типа ОК является возможность активной работы одновременно нескольких элементов, выходы которых соединены параллельно. В этом случае реализуется дополнительная логическая операция, которую называют операцией *монтажной логики*. При реализации операции монтажной логики высокое напряжение на общем выходе возникает только при запираии выходных транзисторов всех элементов. Насыщение выходного транзистора хотя бы одного из элементов снижает выходное напряжение до уровня $U_0 = U_{кэн}$. Таким образом, для получения логической единицы на выходе операции монтажной логики необходимо обеспечить единичное состояние выходов всех элементов, т. е. выполняется монтажная операция И для положительной логики. При этом для отрицательной логики реализуется операция монтажное ИЛИ.

Элементы с выходами типа ОК, также как и элементы с выходами типа ТС, можно использовать в магистрально-модульных структурах микропроцессорных и других систем, где несколько источников сигналов поочередно пользуются одной и той же линией связи. При этом требуется разрешать или запрещать работу того или иного элемента. Для элементов типа ТС это делается с помощью специального сигнала разрешения. Для элементов типа ОК в качестве входа разрешения может быть использован один из обычных входов логического элемента. Например, для элемента И-НЕ, подавая уровень логического 0 на

Чтобы выходной ток элемента с ОК не превысил допустимого значения, следует соблюдать следующее условие

$$I_{\text{вых.0}} = I_R + nI_{\text{вх.0}} \leq I_{\text{вых.0.max}} \quad (2)$$

Ток I_R , протекающий через нагрузочный резистор, определяется в соответствии с законом Ома, следующим выражением:

$$I_R = \frac{U_{CC} - U_0}{R}, \quad (3)$$

где U_{CC} – напряжение источника питания; U_0 – напряжение логического 0 на выходе элемента с ОК; R – сопротивление нагрузочного резистора.

Подставив выражение (3) для тока I_R , протекающего через нагрузочный резистор, в формулу (2), получим соотношение, определяющее ограничение снизу сопротивления R нагрузочного резистора:

$$R \geq \frac{U_{CC} - U_0}{I_{\text{вых.0.max}} - nI_{\text{вх.0.max}}}. \quad (4)$$

Ограничение сверху величины сопротивления R нагрузочного резистора связано с необходимостью обеспечивать достаточно высокий уровень напряжения U_1 , формируемый в схеме при запертом состоянии выходного транзистора элемента с ОК:

$$U_1 \geq U_{\text{вых.1.min}} \quad (5)$$

На рисунке 4, б показаны токи, протекающие в этом случае через нагрузочный резистор. Из рисунка видно, что уровень напряжения U_1 , формируемый в схеме при запертом состоянии выходного транзистора элемента с ОК, определяется выражением

$$U_1 \geq U_{CC} - I_R R. \quad (6)$$

При этом через нагрузочный резистор протекает ток I_R , складывающийся из входных токов $I_{\text{вх.1}}$ логических элементов $ЛЭ_1 - ЛЭ_n$ за вычетом выходного тока I_Z запертого выходного транзистора элемента с ОК:

$$I_R = nI_{\text{вх.1.max}} - I_Z. \quad (7)$$

Из выражений (5) и (7) следует

$$U_{CC} - U_{\text{вых.1.min}} \geq I_R R. \quad (8)$$

Подставив выражение (7) для тока I_R , протекающего через нагрузочный резистор, в формулу (8), получим соотношение, определяющее ограничение сверху сопротивления R нагрузочного резистора:

$$R \leq \frac{U_{CC} - U_{\text{вых.1.min}}}{nI_{\text{вх.1.max}} - I_Z}. \quad (9)$$

Из полученного диапазона значений сопротивления R нагрузочного резистора выбирается некоторое конкретное значение. При этом выбор вблизи нижней границы улучшает быстродействие схемы, а выбор вблизи верхней границы уменьшает потребляемую схемой мощность.

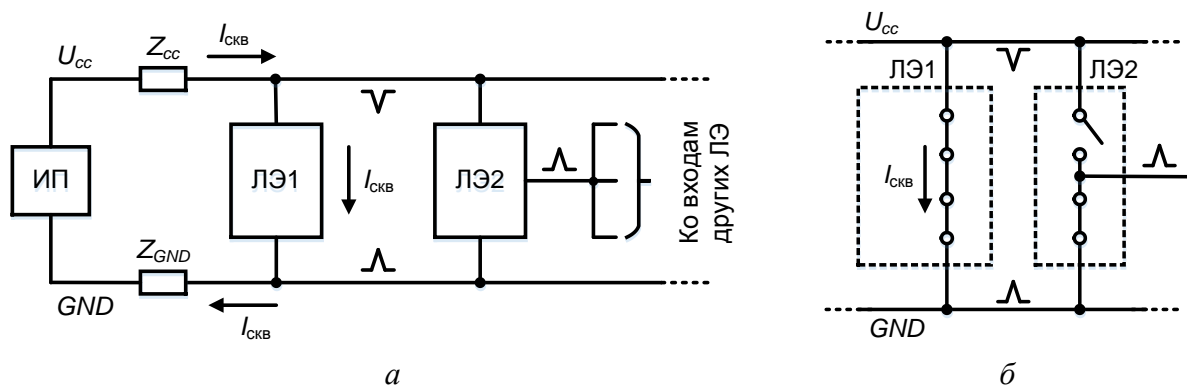
3.2.2 Помехи по цепям питания в схемах цифровых устройств

Типовой проблемой при проектировании схем цифровых устройств на интегральных микросхемах является наличие помех по цепям питания, вызванных, в частности, протеканием токовых импульсов в цепях питания ИМС. Такие помехи могут привести к сбою в работе цифрового устройства [4], [6].

В реальных схемах цифровых устройств при переключениях элементов в цепях питания ИМС возникают кратковременные импульсные токи, которые появляются из-за перезаряда емкостей нагрузки и протекания сквозных токов в выходных каскадах цифровых элементов. Эти токи создают на линии питания (общей точки) импульсы помех, которые через выходные транзисторы попадают на выход подключенных рядом элементов. В результате помехи, возникшие в цепях питания, попадают в схему цифрового устройства и далее распространяются по ее логическим цепям, изменяя логику функционирования устройства.

Для определенности рассмотрим далее механизм создания помех сквозными токами [5]. Аналогичные процессы происходят и при протекании токов перезаряда нагрузочных емкостей.

На рисунке 5 приведена схема, поясняющая механизм появления импульсных помех, связанный с протеканием сквозного тока при переключении логического элемента. На рисунке 5, *а* показаны два подключенных рядом логических элемента ЛЭ1 и ЛЭ2, причем переключается элемент ЛЭ1, а элемент ЛЭ2 находится в состоянии логического нуля. Для лучшего понимания процесса проникновения помехи на выход логического элемента на рисунке 5, *б* выходные транзисторы логических элементов условно представлены ключами, находящимися в соответствующем положении в зависимости от режима, в котором находится выходной каскад элемента.



а – механизм создания импульсов помех в цепях питания;
б – путь проникновения помехи на выход логического элемента

Рисунок 5 – Схема, поясняющая механизм появления импульсных помех, связанный с протеканием сквозного тока при переключении логического элемента

Из рисунка 5 видно, что импульс сквозного тока $I_{\text{СКВ}}$ переключающегося элемента ЛЭ1 протекает через выходные транзисторы (условно показанные замкнутыми ключами) от источника питания (ИП) U_{CC} на общую точку схемы GND через цепи питания, имеющие полные сопротивления Z_{CC} и Z_{GND} .

Современные источники питания, выполняющие стабилизацию выходного напряжения, имеют очень низкое выходное сопротивление за счет применения в своих схемах глубоких отрицательных обратных связей. Однако такие обратные связи инерционны и не обеспечивают качественной фильтрации коротких импульсных помех, т. к. для них источник питания не обеспечивает того низкого уровня выходного сопротивления, которое обеспечивается в статическом режиме. При этом основную часть сопротивлений Z_{CC} и Z_{GND} составляют индуктивности линий, на которых и происходит падение напряжения при протекании импульса сквозного тока $I_{\text{СКВ}}$. Протекание сквозного тока создает на линии питания отрицательный импульс помехи, а на линии общей точки – положительный. Эти импульсы через транзисторы выходного каскада элемента ЛЭ2, подключенного вблизи элемента ЛЭ1, поступают на его выход. Если элемент ЛЭ2 находится в состоянии логического нуля, как это показано на рисунке 5, б, то его выход через открытый нижний выходной транзистор связан с линией GND , откуда положительный импульс помехи попадает на выход элемента. Если элемент ЛЭ2 будет находиться в состоянии логической единицы, то через открытый верхний выходной транзистор на его выход пройдет отрицательный импульс помехи с линии питания U_{CC} . Далее импульс помехи с выхода элемента ЛЭ2 поступает на входы других элементов. Если величина импульса помехи повышает/понижает уровень напряжения на входах этих элементов до порогового, то это может привести к переключению элементов в неправильное состояние, что в свою очередь может привести к неправильному переключению и других связанных уже с ними элементов схемы. Таким образом, импульс помехи может распространяться по обычным сигнальным цепям схемы цифрового устройства.

Так как причины возникновения помех рассмотренного типа определяются свойствами самих элементов, то при проектировании схем цифровых устройств необходимо принимать меры для снижения их влияния на работу устройства. На практике обычно применяют простые пассивные методы, которые реализуются как на конструктивном, так и на схемотехническом уровне.

Конструктивные методы заключаются в обеспечении минимального значения сопротивления Z_{GND} : линии цепей питания делаются утолщенными, для их реализации отводят целые слои многослойных печатных плат и т. д.

На схемотехническом уровне применяются следующие основные меры:

- в выходные цепи элементов добавляют резисторы с небольшими сопротивлениями, ограничивающие сквозные токи и токи перезаряда емкостей;
- на выходах элементов включают развязывающие каскады для ограничения емкостных нагрузок;
- используют фильтрацию напряжения в цепях питания ИМС.

Для фильтрации напряжения в цепях питания ИМС между линиями U_{CC} и GND включают фильтрующие конденсаторы C_{ϕ} (рисунок 6), имеющие низкое сопротивление для коротких импульсов. С помощью таких конденсаторов в схеме создаются дополнительные пути для протекания импульсов сквозного тока и тока перезаряда емкостей, минуя сопротивление Z_{CC} . Поэтому импульсы помех на линиях питания не возникают и, следовательно, они не попадают на выходы элементов. Такие конденсаторы должны подключаться к выводам питания корпусов ИМС как можно более короткими проводниками.

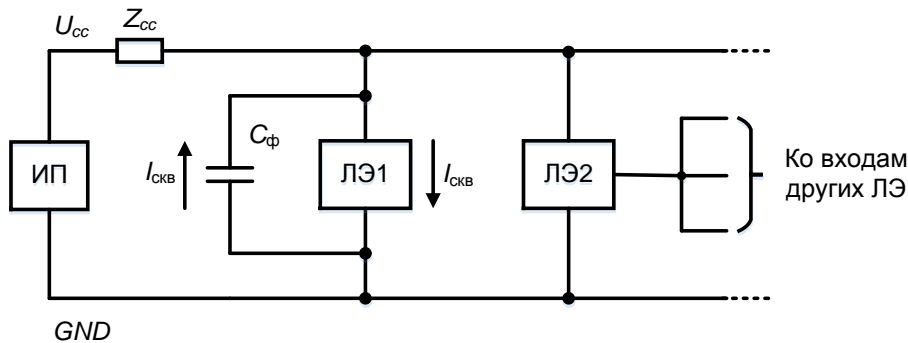


Рисунок 6 – Пути протекания импульсов сквозного тока при использовании фильтрующего конденсатора

3.2.3 Типовые ситуации при проектировании цифровых устройств на интегральных микросхемах

В общем случае процесс проектирования цифрового устройства на интегральных микросхемах включает два этапа [7]. На первом этапе разрабатывается логическая схема устройства, обеспечивающая его функционирование в соответствии с заданным законом. При этом решается задача синтеза логической схемы. Решение задачи синтеза логической схемы осуществляется путем выполнения следующей последовательности шагов:

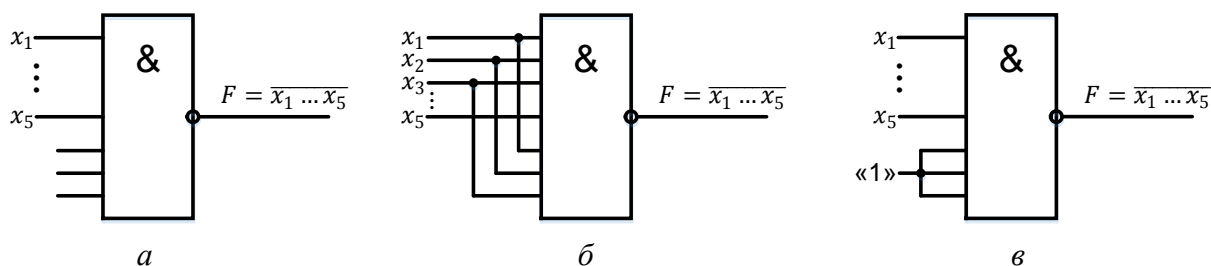
- 1) составление математического описания процесса функционирования разрабатываемого устройства в виде системы логических функций;
- 2) получение минимальной формы для каждой логической функции;
- 3) переход от логических функций к логической схеме.

На втором этапе разработанная логическая схема реализуется с использованием стандартных ИМС той или иной серии, т. е. строится принципиальная схема устройства. Каждый логический элемент схемы сопоставляют с некоторым физическим элементом некоторой стандартной ИМС. При этом в качестве критерия оптимизации используется условие минимального количества корпусов интегральных микросхем, выбранных для построения схемы цифрового устройства.

В общем случае при переходе от логической схемы к принципиальной возможны ситуации несовпадения элементов логической схемы и элементов, которые имеются в заданной стандартной серии ИМС, используемых для построения схемы устройства. Рассмотрим основные типовые ситуации, возника-

ющие при реализации логической схемы устройства с использованием стандартных ИМС.

Наличие у имеющихся элементов неиспользуемых входов [6]. Например, логическая схема содержит элемент И-НЕ с пятью входами. В стандартных сериях нет микросхем, которые содержали бы элементы И-НЕ с пятью входами. Однако есть ИМС, содержащая элемент И-НЕ с восемью входами. Поэтому в такой ситуации приходится выбирать элемент с восемью входами. В этом случае у элемента оказывается три лишних входа, т. е. незадействованных в данной схеме (рисунок 7, а). При этом возникает вопрос, что делать в данной ситуации – оставить незадействованные входы неподключенными либо подсоединить их к каким-то цепям схемы.



а – незадействованные входы оставлены неподключенными;
 б – незадействованные входы подсоединены к задействованным входам;
 в – на незадействованные входы подана логическая константа (логическая 1)

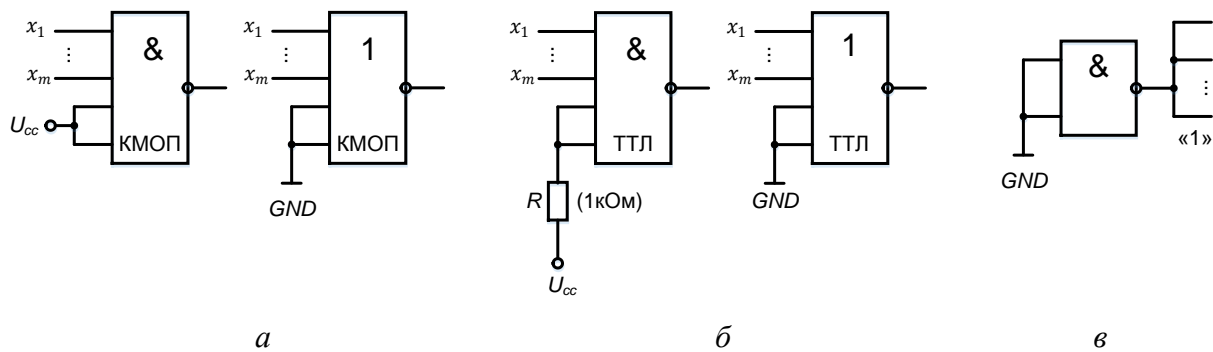
Рисунок 7 – Возможные варианты подключения неиспользуемых входов

Данный вопрос решается с учетом конкретного типа технологии используемых ИМС. Если незадействованные входы оставить неподключенными (см. рисунок 7, а), то это может привести к изменению логики работы схемы. ИМС КМОП имеют очень большие входные сопротивления, поэтому на неподключенные входы легко наводятся паразитные потенциалы, которые могут изменить состояние элемента. В ИМС ТТЛ при оставленных неподключенными входах ухудшаются параметры быстродействия элементов. Поэтому незадействованные в схеме входы никуда не подсоединенными не оставляют.

Таким образом, незадействованные входы необходимо подключать к каким-либо цепям схемы. С точки зрения логики работы элемента такие входы можно подсоединить к задействованным входам (см. рисунок 7, б) или подать на них логические константы – логический 0 или 1 (см. рисунок 7, в). Однако подключение незадействованных входов к задействованным вызывает увеличение нагрузки на выходы элементов, являющихся источниками сигнала для задействованных входов. Это может привести к их перегрузке. Кроме того, увеличение нагрузки на выходе может привести к увеличению задержки элемента.

В результате правильным решением по отношению к незадействованным в схеме входам является подсоединение их к логическим константам (логическим 0 или 1), не изменяющим работу элемента для задействованных входов (рисунок 8). При этом уровни напряжений U_1 и U_0 для КМОП совпадают с уровнями U_{CC} (напряжение питания) и общей точки схемы. Поэтому неисполь-

зюемые входы ИМС КМОП подключают к линиями U_{CC} и GND (см. рисунок 8, а). Для ТТЛ уровень напряжения U_1 на 1,5–2 В ниже уровня напряжения питания U_{CC} . Поэтому подключение входа к напряжению питания U_{CC} может вызвать его пробой. Для предотвращения пробоев неиспользуемые входы подключают к линии U_{CC} через резисторы с сопротивлением $R = 1-10$ кОм (см. рисунок 8, б). Уровень логической единицы можно получать с помощью инвертора (см. рисунок 8, в). Если в качестве инвертора использовать специальный элемент с повышенной нагрузочной способностью, то к его выходу можно подсоединять до 30 входов.



а – для КМОП; б – для ТТЛ; в – схема формирования логической единицы

Рисунок 8 – Рекомендуемые варианты подключения неиспользуемых входов

Наличие в корпусе ИМС неиспользуемых элементов [6]. В корпусах ИМС стандартных серий размещается определенное количество элементов. Это количество определяется числом входов и выходов элементов, размещенных в корпусе, а также типом корпуса и числом имеющихся у него выводов. Например, ИМС стандартных серий малой степени интеграции размещаются в корпусах, имеющих 14 выводов. В таком корпусе можно разместить, например, как показано на рисунке 9, четыре двухвходовых логических элемента И-НЕ (И, ИЛИ, ИЛИ-НЕ).

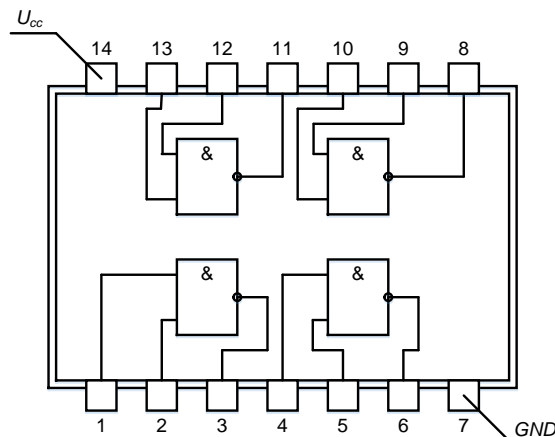


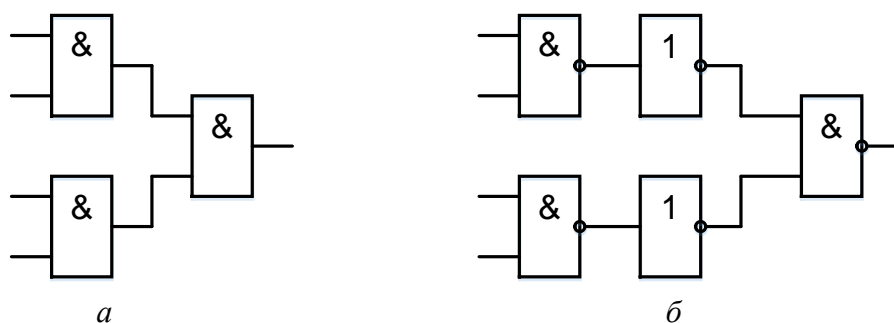
Рисунок 9 – Размещение логических элементов И-НЕ в корпусе ИМС SN74ALS00 (KP1533ЛА3)

Поэтому при переходе от логической схемы к принципиальной возможна ситуация, когда не все элементы, имеющиеся в корпусе ИМС, используются для построения схемы устройства. При этом проблема заключается в том, что неиспользованные элементы также подключены к напряжению питания, которое является общим для всего корпуса. В этом случае возникает вопрос, что делать с неиспользованными элементами – оставить их неподключенными, либо подсоединить их к каким-то цепям схемы.

Принципиально неиспользованные элементы можно оставить неподключенными к каким-либо цепям схемы. Однако, если мощности, потребляемые элементом в состояниях 0 и 1, не одинаковые, то имеет смысл установить неиспользуемый элемент в состояние минимальной мощности, подав на его входы соответствующие константы.

У имеющихся в ИМС логических элементов не хватает необходимого числа входов [6]. Обычно в корпусах ИМС стандартных серий размещают логические элементы с одинаковым числом входов. Например, в корпусе размещено четыре двухвходовых элемента, три трехвходовых или два элемента с четырьмя входами. Поэтому может возникнуть ситуация, когда в ИМС имеются незадействованные элементы, но они имеют число входов меньше, чем у элементов логической схемы, которые требуется реализовать.

Например, в логической схеме требуется реализовать элементы с четырьмя входами, а в ИМС незадействованными остались только двухвходовые элементы. В этом случае встает задача наращивания числа входов. Для логических операций И, ИЛИ эта задача решается просто. Для получения нужного числа входов берется несколько элементов с меньшим числом входов, выходы которых объединяются далее элементом того же типа (рисунок 10, а). Для операций И-НЕ, ИЛИ-НЕ задача наращивания числа входов решается аналогичным способом, но выходы элементов с меньшим числом входов объединяются далее элементом того же типа через промежуточные инверторы, как показано на рисунке 10, б.



a – для элементов И; *б* – для элементов И-НЕ

Рисунок 10 – Схема наращивания числа входов

3.3 Примеры решения задач по проектированию цифровых устройств при выполнении заданий курсового проекта

В данном подразделе приведены примеры решения задач по проектированию схем комбинационных и последовательностных цифровых устройств при выполнении заданий курсового проекта [1], [2], [9], [10].

3.3.1. Построить преобразователь кода 2421 в код Грея на логических элементах серии SN74ALS (КР1533).

Решение.

Работа преобразователя кода 2421 в код Грея описывается таблицей истинности (таблица 1).

Таблица 1 – Таблица истинности преобразователя кода 2421 в код Грея

Код 2421	Код Грея	Код 2421	Код Грея
$x_3x_2x_1x_0$	$y_3y_2y_1y_0$	$x_3x_2x_1x_0$	$y_3y_2y_1y_0$
0000	0000	1011	0111
0001	0001	1100	0101
0010	0011	1101	0100
0011	0010	1110	1100
0100	0110	1111	1101

Нанесем функции y_i на карты Карно, добавив шесть избыточных наборов кода 2421 (рисунок 11).

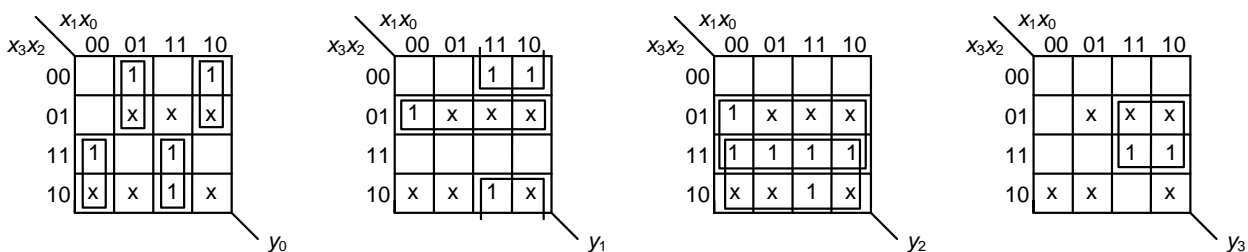


Рисунок 11 – Карты Карно для преобразователя кода 2421 в код Грея

По картам Карно можно записать логические выражения для выходных функций кода Грея:

$$\begin{aligned}
 y_0 &= x_3\bar{x}_1\bar{x}_0 \vee \bar{x}_3\bar{x}_1x_0 \vee x_3x_1x_0 \vee \bar{x}_3x_1\bar{x}_0 = \\
 &= \bar{x}_1(x_3 \oplus x_0) \vee x_1(\overline{x_3 \oplus x_0}) = x_1 \oplus x_3 \oplus x_0, \\
 y_1 &= \bar{x}_2x_1 \vee \bar{x}_3x_2 = \overline{\overline{\bar{x}_2x_1}} \vee \overline{\overline{\bar{x}_3x_2}}, \\
 y_2 &= x_3 \vee x_2 = \overline{\overline{x_3}} \vee \overline{\overline{x_2}} = \overline{\overline{x_3\bar{x}_2}}, \quad y_3 = x_2x_1 = \overline{\overline{x_2\bar{x}_1}}.
 \end{aligned}
 \tag{10}$$

Из выражений (10) видно, что преобразователь кода 2421 в код Грея можно реализовать на двух ИМС SN74ALS00 (КР1533ЛА3) и одной ИМС SN74ALS86 (КР1533ЛП5) (рисунок 12).

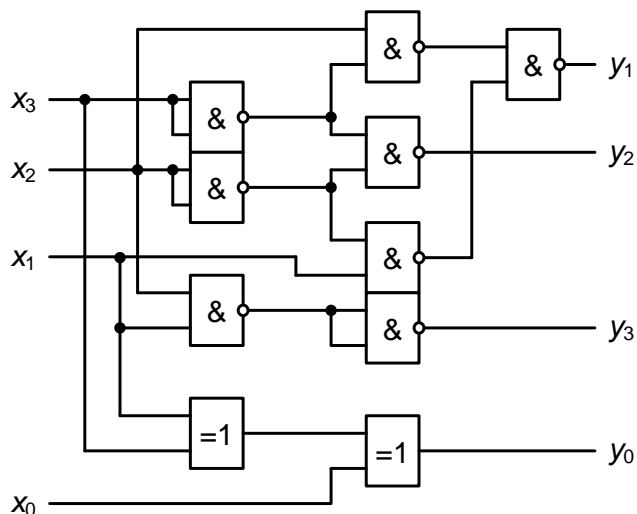


Рисунок 12 – Преобразователь кода 2421 в код Грея

3.3.2. Построить только на одной микросхеме SN74ALS153 (КР1533КП2) функциональный узел, реализующий логическую функцию F трех переменных, заданную таблицей истинности (таблица 2).

Таблица 2 – Таблица истинности логической функции F (к примеру 3.3.2)

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

Решение.

Микросхема SN74ALS153 (КР1533КП2) представляет собой сдвоенный селектор-мультиплексор «1 из 4» с общими входами выбора данных (адресными входами) и отдельными входами стробирования. Мультиплексор имеет два адресных входа (входы выбора данных SED1, SED2), а по условию задачи необходимо реализовать булеву функцию трёх переменных. Для решения этой задачи следует две из трех переменных подать на входы выбора данных SED1, SED2, а третью переменную использовать для определения значения, которое нужно подавать на входы данных D_i мультиплексора.

Если нанести на карту Карно логическую функцию F , то после минимизации получим $F = \overline{A}C \vee B\overline{C} \vee A\overline{C}$. Из полученной минимальной формы функ-

ции F следует, что переменная B встречается без инверсии. Это означает, что переменную B можно подавать на входы данных D_i мультиплексора без дополнительного инвертора. Поэтому для подключения к адресным входам мультиплексора выбираем переменные A и C , которые встречаются с инверсией. Таблицу 2 для удобства можно представить в виде таблицы 3, поменяв местами переменные B и C и сгруппировав наборы парами так, чтобы переменные A и C имели одинаковые значения.

Таблица 3 – Видоизмененная таблица истинности логической функции F

A	C	B	F	D_i
0	0	0	0	$D_0 = B$
0	0	1	1	
0	1	0	1	$D_1 = 1$
0	1	1	1	
1	0	0	1	$D_2 = 1$
1	0	1	1	
1	1	0	0	$D_3 = 0$
1	1	1	0	

Подадим переменную A на адресный вход SED2, а переменную C – на адресный вход SED1 мультиплексора. Рассматривая таблицу 3 по две строки и сравнивая значение переменной B со значением функции F определим, что нужно подавать на входы данных D_i мультиплексора (рисунок 13).

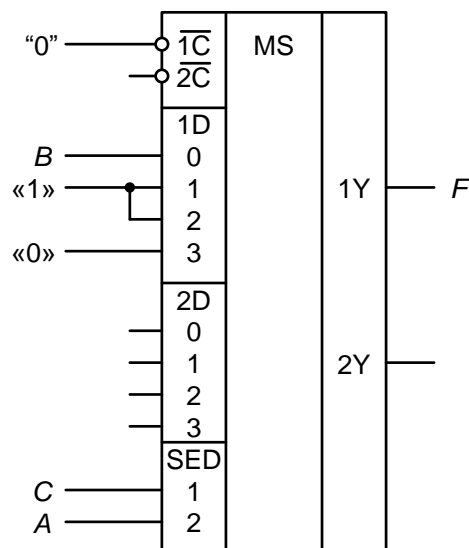


Рисунок 13 – Реализация логической функции F на мультиплексоре (к примеру 3.3.2)

Можно не составлять таблицу 3, а в выражение логической функции F подставлять адресные наборы AC и определять значение, которое нужно подавать на соответствующий вход данных D_i мультиплексора. Результаты такой подстановки удобно оформить в виде таблицы 4. Например, подставим набор $A = C = 0$. Тогда $D_0 = \bar{0}0 \vee B\bar{0} \vee 0\bar{0} = B$.

Таблица 4 – Результаты подстановки наборов AC в выражение функции F

A	C	D_i
0	0	$D_0 = \bar{0}0 \vee B\bar{0} \vee 0\bar{0} = B$
0	1	$D_1 = \bar{0}1 \vee B\bar{1} \vee 0\bar{1} = 1$
1	0	$D_2 = \bar{1}0 \vee B\bar{0} \vee 1\bar{0} = 1$
1	1	$D_3 = \bar{1}1 \vee B\bar{1} \vee 1\bar{1} = 0$

3.3.3. Построить только на одной микросхеме SN74ALS153 (КР1533КП2) комбинационный узел, реализующий логическую функцию F трех переменных, заданную таблицей истинности (таблица 5).

Таблица 5 – Таблица истинности логической функции F (к примеру 3.3.3)

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

Решение.

Данная задача решается аналогично предыдущей. Однако возможна ситуация, когда все три переменные встречаются в минимальной форме функции с инверсией. Это означает, что любую из трех переменных нужно будет подавать на входы данных D_i мультиплексора через дополнительный инвертор.

Нанесем функцию F на карту Карно. После минимизации получим $F = \bar{A}\bar{B} \vee \bar{A}C \vee B\bar{C}$. В этом выражении все переменные встречаются в инверсном виде. Поэтому любые две переменные можно выбрать для подключения к адресным входам мультиплексора. При этом при определении значения, которое нужно подавать на входы данных D_i мультиплексора может потребоваться дополнительно инвертор для подключения третьей переменной.

Разобьем таблицу 5 по две строки. Если подать переменную A на SED2, переменную B на SED1 и сравнить значение переменной C со значением функции F , то легко определить, что нужно подавать на входы данных D_i мультиплексора (таблица 6).

Из таблицы 6 видно, что на вход данных мультиплексора D_3 переменную C нужно подавать через инвертор.

Поскольку микросхема SN74ALS153 (КР1533КП2) представляет собой двоянный селектор-мультиплексор «1 из 4», то инверсию переменной C можно получить на втором выходе мультиплексора-селектора и не использовать дополнительный инвертор.

Таблица 6 – Значения, подключаемые на входы данных мультиплексора

A	B	C	F	D_i
0	0	0	0	$D_0 = C$
0	0	1	1	
0	1	0	1	$D_1 = 1$
0	1	1	1	
1	0	0	1	$D_2 = 1$
1	0	1	1	
1	1	0	1	$D_3 = \bar{C}$
1	1	1	0	

На рисунке 14 показана функциональная схема комбинационного узла, реализующего заданную логическую функцию, в которой для инверсии переменной C используется второй мультиплексор ИМС SN74ALS153 (КР1533КП2).

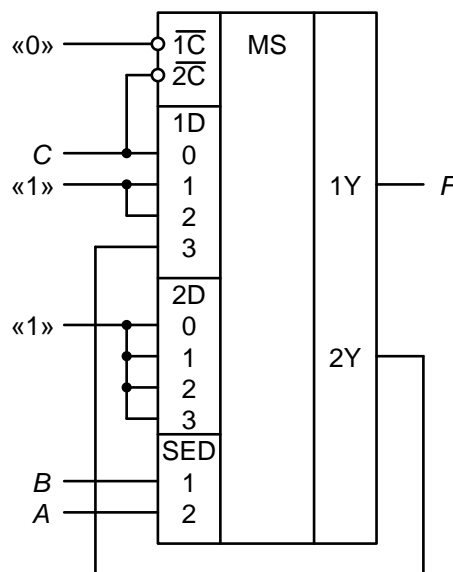


Рисунок 14 – Реализация логической функции F на мультиплексоре (к примеру 3.3.3)

3.3.4. Разработать преобразователь 4-разрядного прямого кода целого числа $y_3y_2y_1y_0$ (y_3 – знаковый разряд) в дополнительный $F_3F_2F_1F_0$ (F_3 – знаковый разряд) на ИМС SN74ALS181 (КР1533ИП3) и инверторе той же серии.

Решение.

Микросхема SN74ALS181 (КР1533ИП3) представляет собой 4-разрядное арифметическо-логическое устройство (АЛУ). АЛУ выполняет 16 арифметических и логических операций с двумя 4-разрядными операндами. Тип операции определяется комбинацией сигналов на входах выбора SE_3 – SE_0 . Выбор между логическими и арифметическими операциями осуществляется с помощью специального входа MO . При высоком уровне напряжения на входе MO отключаются переносы между разрядами, и микросхема выполняет логические операции. При низком уровне напряжения на этом входе АЛУ выполняет арифметические операции. Микросхема имеет вход переноса \overline{CR}_n , уровень напряжения на котором влияет на выполнение арифметических операций.

Для выполнения преобразования прямого кода в дополнительный необходимо выбрать четырнадцатую операцию, подав на входы выбора SE3–SE0 двоичный код 1110. В этом случае будет выполняться либо логическая операция $A \vee B$ (при MO = 1), либо арифметическая операция $(A \vee \bar{B}) + A + \overline{CR}_n$ (при MO = 0). Поэтому, если подать на вход MO инверсное значение знакового разряда входного целого числа \bar{y}_3 , на вход \overline{CR}_n – низкий уровень напряжения, а в качестве операндов на входы A3–A0 – двоичный код 0000, на входы B3–B0 – значащие разряды входного целого числа с нулевым значением знакового разряда $0y_2y_1y_0$, то на выходе АЛУ получим дополнительный код входного числа.

При $y_3 = 0$ (входное число положительное) на входе MO будет высокий уровень напряжения (MO = 1) и в АЛУ будет выполняться логическая операция

$$\begin{aligned} A \vee B &= (a_3 \vee b_3)(a_2 \vee b_2)(a_1 \vee b_1)(a_0 \vee b_0) = 0 \vee B = \\ &= (0 \vee 0)(0 \vee y_2)(0 \vee y_1)(0 \vee y_0) = 0y_2y_1y_0 = y_3y_2y_1y_0. \end{aligned} \quad (11)$$

При $y_3 = 1$ (входное целое число отрицательное) на входе MO будет низкий уровень напряжения (MO = 0). Учитывая, что на вход \overline{CR}_n подан низкий уровень напряжения $\overline{CR}_n = 0$, в АЛУ выполняется арифметическая операция

$$\begin{aligned} (A \vee \bar{B}) + A + 1 &= (a_3 \vee \bar{b}_3)(a_2 \vee \bar{b}_2)(a_1 \vee \bar{b}_1)(a_0 \vee \bar{b}_0) + A + 1 = \\ &= (0 \vee \bar{B}) + 0 + 1 = (0 \vee \bar{0})(0 \vee \bar{y}_2)(0 \vee \bar{y}_1)(0 \vee \bar{y}_0) + 1 = \\ &= 1\bar{y}_2\bar{y}_1\bar{y}_0 + 1 = y_3\bar{y}_2\bar{y}_1\bar{y}_0 + 1. \end{aligned} \quad (12)$$

Анализ выражений (11) и (12) показывает, что на выходах F_i АЛУ получаем дополнительный код входного целого числа $y_3y_2y_1y_0$.

Функциональная схема преобразователя 4-разрядного прямого кода целого числа в дополнительный приведена на рисунке 15.

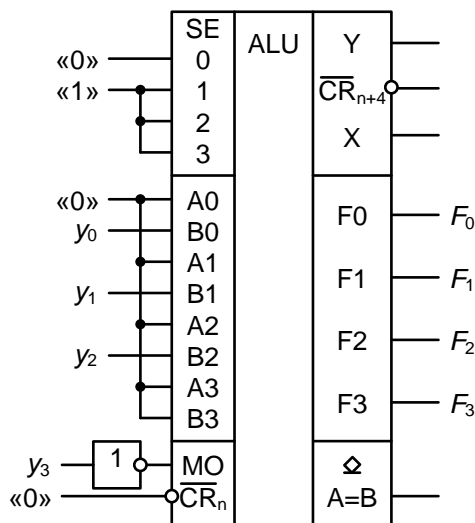


Рисунок 15 – Преобразователь 4-разрядного прямого кода в дополнительный на ИМС SN74ALS181 (КР1533ИП3) и инверторе

3.3.5. На одной ИМС SN74ALS112 (КР1533ТВ9) и логических элементах той же серии построить генератор чисел 7 – 2 – 4 – 11.

Решение.

Микросхема SN74ALS112 (КР1533ТВ9) представляет собой два JK-триггера, срабатывающих по отрицательному фронту тактового сигнала, с входами сброса и предустановки.

Для построения генератора нужна последовательная схема на четыре состояния (два триггера), к выходам которой подключается выходная логика, имеющая четыре выхода $y_3y_2y_1y_0$, т. к. для кодирования числа 11 необходимо четыре двоичных разряда.

Четыре состояния можно получить в схеме регистра сдвига, суммирующего счетчика, вычитающего счетчика или пересчетного устройства, обеспечивающего любую заданную последовательность состояний. Выберем схему вычитающего двухразрядного счётчика. Выходы $y_3y_2y_1y_0$ определим по таблице истинности (таблица 7).

Таблица 7 – Таблица истинности выходной логики генератора

Q_2	Q_1	y_3	y_2	y_1	y_0
0	0	0	1	1	1
1	1	0	0	1	0
1	0	0	1	0	0
0	1	1	0	1	1

$$y_0 = \overline{Q_2}, \quad y_1 = \overline{Q_2} \overline{Q_1}, \quad y_2 = \overline{Q_1}, \quad y_3 = \overline{Q_2} Q_1 = \overline{\overline{\overline{Q_2} Q_1}}. \quad (13)$$

Из выражений (13) видно, что выходная логика генератора может быть реализована на одной ИМС SN74ALS00 (КР1533ЛА3).

Функциональная схема генератора чисел приведена на рисунке 16.

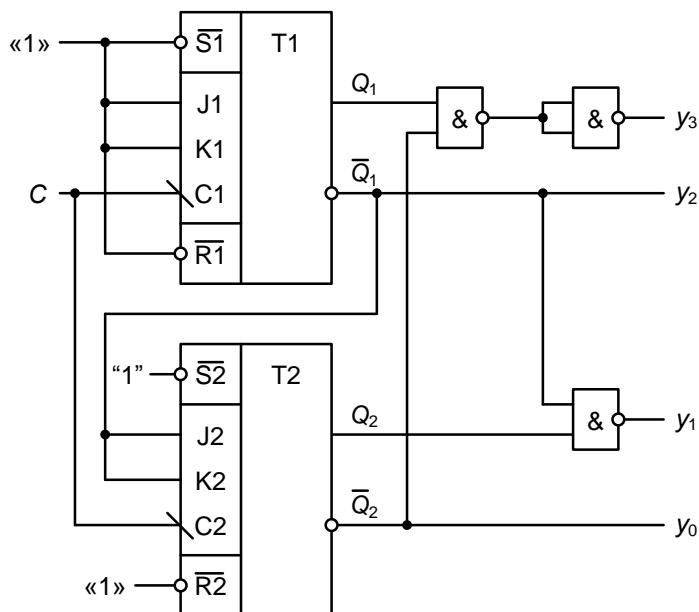


Рисунок 16 – Функциональная схема генератора чисел 7 – 2 – 4 – 11

3.3.6. На двух ИМС SN74ALS109 (КР1533ТВ15) и логических элементах этой же серии построить генератор чисел 7 – 2 – 4 – 11. Построить полный граф состояний генератора.

Решение.

Микросхема SN74ALS109 (КР1533ТВ15) представляет собой два *JK*-триггера, срабатывающих по положительному фронту тактового сигнала, со входами сброса и предустановки. Особенностью *JK*-триггеров, входящих в состав микросхемы, является то, что они имеют инверсный вход *K*.

С помощью управляющей таблицы *JK*-триггера (таблица 8) составим таблицу переходов генератора чисел (таблица 9).

Таблица 8 – Управляющая таблица *JK*-триггера

$Q^t \rightarrow Q^{t+1}$	<i>J</i>	<i>K</i>
0 → 0	0	×
0 → 1	1	×
1 → 0	×	1
1 → 1	×	0

При составлении таблицы переходов генератора чисел следует помнить о том, что *JK*-триггеры, входящих в состав микросхемы SN74ALS109 (КР1533ТВ15), имеют инверсный вход *K*.

Таблица 9 – Таблица переходов генератора чисел

Q_3	Q_2	Q_1	Q_0	J_3	\bar{K}_3	J_2	\bar{K}_2	J_1	\bar{K}_1	J_0	\bar{K}_0
0	1	1	1	0	×	×	0	×	1	×	0
0	0	1	0	0	×	1	×	×	0	0	×
0	1	0	0	1	×	×	0	1	×	1	×
1	0	1	1	×	0	1	×	×	1	×	1

По таблице переходов генератора чисел (см. таблицу 9) составляем карты Карно (рисунок 17) для сигналов J_i и \bar{K}_i , подключаемым к информационным входам *JK*-триггеров.

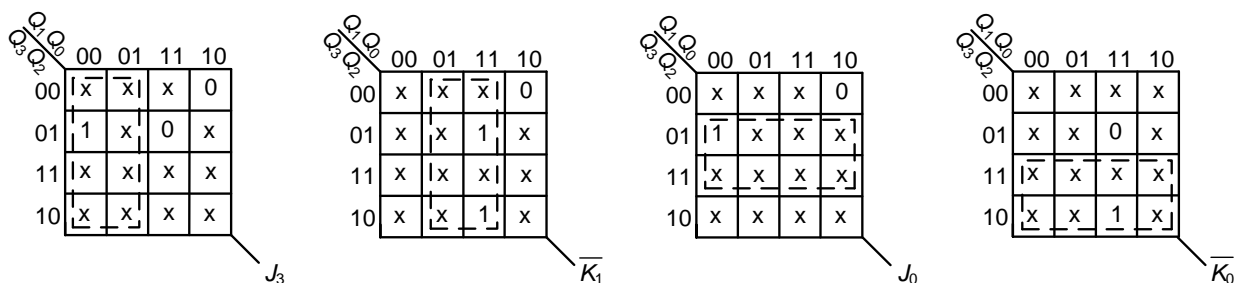


Рисунок 17 – Карты Карно для сигналов J_i и \bar{K}_i , подключаемым к информационным входам *JK*-триггеров генератора чисел

По картам Карно составляем логические выражения для сигналов J_i и \bar{K}_i :

$$\begin{aligned} J_3 &= \bar{Q}_1, J_2 = 1, J_1 = 1, J_0 = Q_2, \\ \bar{K}_3 &= 0, \bar{K}_2 = 0, \bar{K}_1 = Q_0, \bar{K}_0 = Q_3. \end{aligned} \quad (14)$$

Для построения полного графа состояний генератора составим таблицу переходов для избыточных состояний (таблица 11). Для составления таблицы 11 выпишем $16 - 4 = 12$ избыточных состояний, по уравнениям (14) заполним значения сигналов J_i и \bar{K}_i , а затем по таблице переключения JK -триггера (таблица 10) определим следующие состояния триггеров генератора в момент времени $t + 1$. При составлении таблицы переходов для избыточных состояний генератора чисел следует помнить о том, что JK -триггеры, входящие в состав микросхемы SN74ALS109 (КР1533ТВ15), имеют инверсный вход K .

Полный граф состояний генератора чисел показан на рисунке 18.

Таблица 10 – Таблица переключения JK -триггера

J	K	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	\bar{Q}^t

Таблица 11 – Таблица переходов для избыточных состояний генератора чисел

Q^t	t												$t+1$				
	Q_3	Q_2	Q_1	Q_0	J_3	\bar{K}_3	J_2	\bar{K}_2	J_1	\bar{K}_1	J_0	\bar{K}_0	Q_3	Q_2	Q_1	Q_0	Q^{t+1}
0	0	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	14
1	0	0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	14
3	0	0	1	1	0	0	1	0	1	1	0	0	0	1	1	0	6
5	0	1	0	1	1	0	1	0	1	1	1	0	1	0	1	0	10
6	0	1	1	0	0	0	1	0	1	0	1	1	0	0	0	1	1
8	1	0	0	0	1	0	1	0	1	0	0	1	0	1	1	0	6
9	1	0	0	1	1	0	1	0	1	1	0	1	0	1	1	1	7
10	1	0	1	0	0	0	1	0	1	0	0	1	0	1	0	0	4
12	1	1	0	0	1	0	1	0	1	0	1	1	0	0	1	1	3
13	1	1	0	1	1	0	1	0	1	1	1	1	0	0	1	1	3
14	1	1	1	0	0	1	1	0	1	0	1	1	0	0	0	1	1
15	1	1	1	1	0	0	1	0	1	1	1	1	0	0	1	1	3

Из рисунка 18 видно, что граф состояний генератора состоит из двух частей. Первая часть (на рисунке 18 верхняя) включает заданную последовательность состояний 7 – 2 – 4 – 11 плюс три избыточных состояния 5, 9, 10 – рабочая часть графа. Эта часть не имеет тупиковых состояний и при сбое из любого избыточного состояния за конечное число тактов работы вернется к генерации заданной последовательности состояний. Вторая часть (на рисунке 18 нижняя)

является нерабочей частью графа состояний генератора, т. к. состоит только из избыточных состояний. При попадании состояния в результате сбоя в эту часть таблицы генератор не вернется к генерации заданной последовательности состояний 7 – 2 – 4 – 11, т. к. эта часть имеет два тупиковых состояния 1 – 14, в которые генератор попадает при сбое за конечное число тактов работы. Поэтому в закон функционирования генератора необходимо внести изменения, обеспечивающие переход из одного из тупиковых состояний в рабочую часть графа. Для этого достаточно, например, в таблице переходов для избыточных состояний (см. таблицу 11) в строке, соответствующей переходу из состояния 14, для функции \overline{K}_3 значение 1 изменить на 0. В результате генератор из состояния 14 будет переходить в состояние 9, а не возвращаться в состояние 1 (на рисунке 18 показано штриховой линией). Тем самым генератор вернется к генерации заданной последовательности состояний.

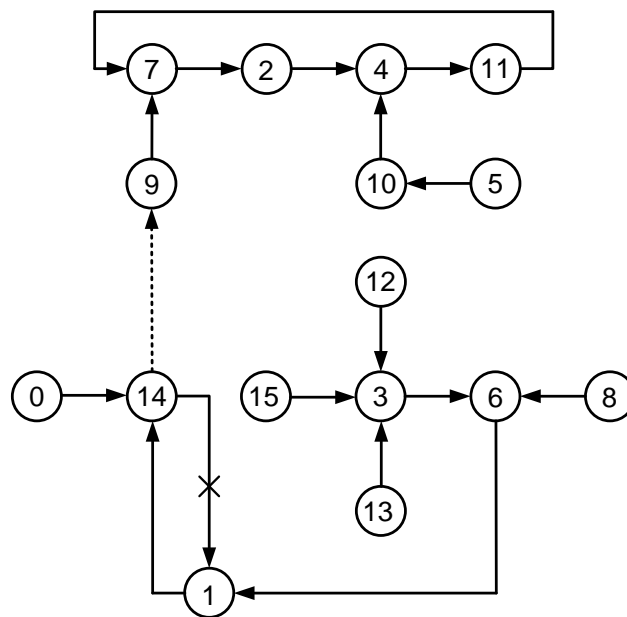


Рисунок 18 – Полный граф состояний генератора чисел 7 – 2 – 4 – 11

Для получения нового логического выражения для сигнала \overline{K}_3 составим карту Карно (рисунок 19).

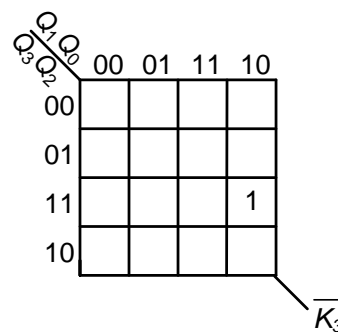


Рисунок 19 – Карта Карно для сигнала \overline{K}_3

По карте Карно составляем логическое выражение для сигнала \bar{K}_3 :

$$\bar{K}_3 = Q_3 Q_2 Q_1 \bar{Q}_0. \quad (15)$$

Функциональная схема генератора чисел приведена на рисунке 20.

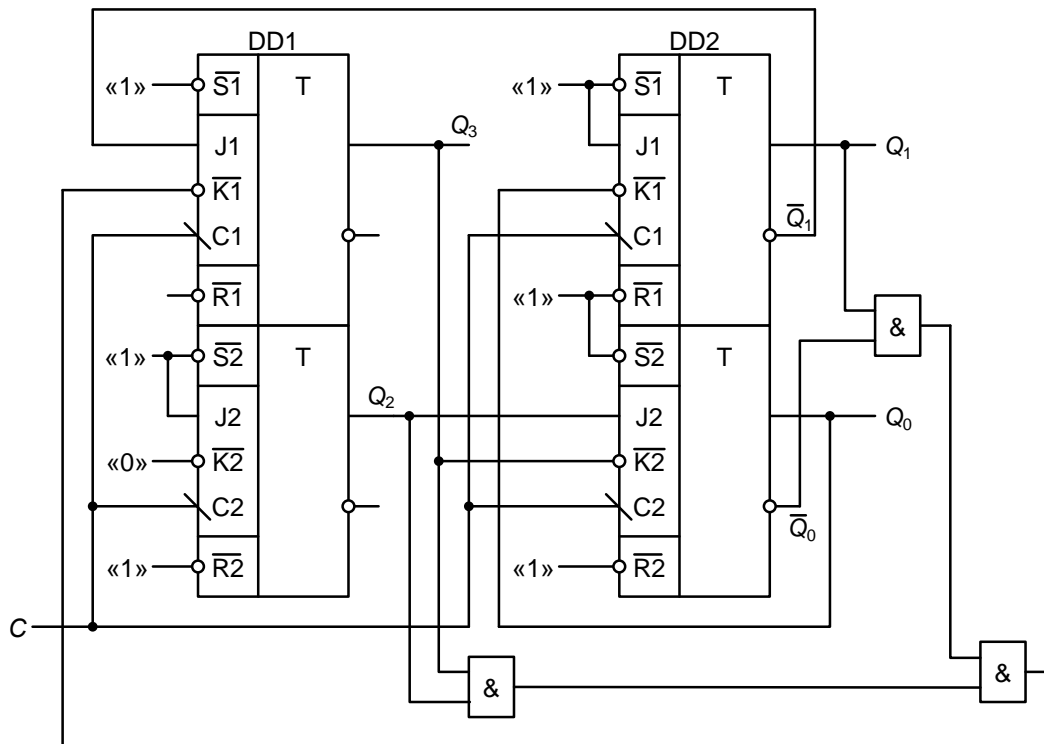


Рисунок 20 – Функциональная схема генератора чисел 7 – 2 – 4 – 11 на четырех JK-триггерах микросхем SN74ALS109 (КР1533ТВ15)

4 СХЕМОТЕХНИЧЕСКОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ ПРОЕКТИРОВАНИЯ XILINX ISE

4.1 Основы работы в среде проектирования Xilinx ISE

4.1.1 Создание проекта

После запуска среды Xilinx ISE появится главное окно программы, также называемое «Навигатор проектов» («ISE Project Navigator»).

Основными элементами интерфейса данного окна являются (рисунок 21):

- 1) меню программы и панель инструментов для быстрого доступа к наиболее часто используемым функциям;
- 2) область быстрого доступа к командам открытия/загрузки проекта;
- 3) область отображения ранее открывавшихся проектов;
- 4) область консоли программы, в которой будут отражены информационные сообщения по мере работы над проектом, к примеру, предупреждения и ошибках в схеме.

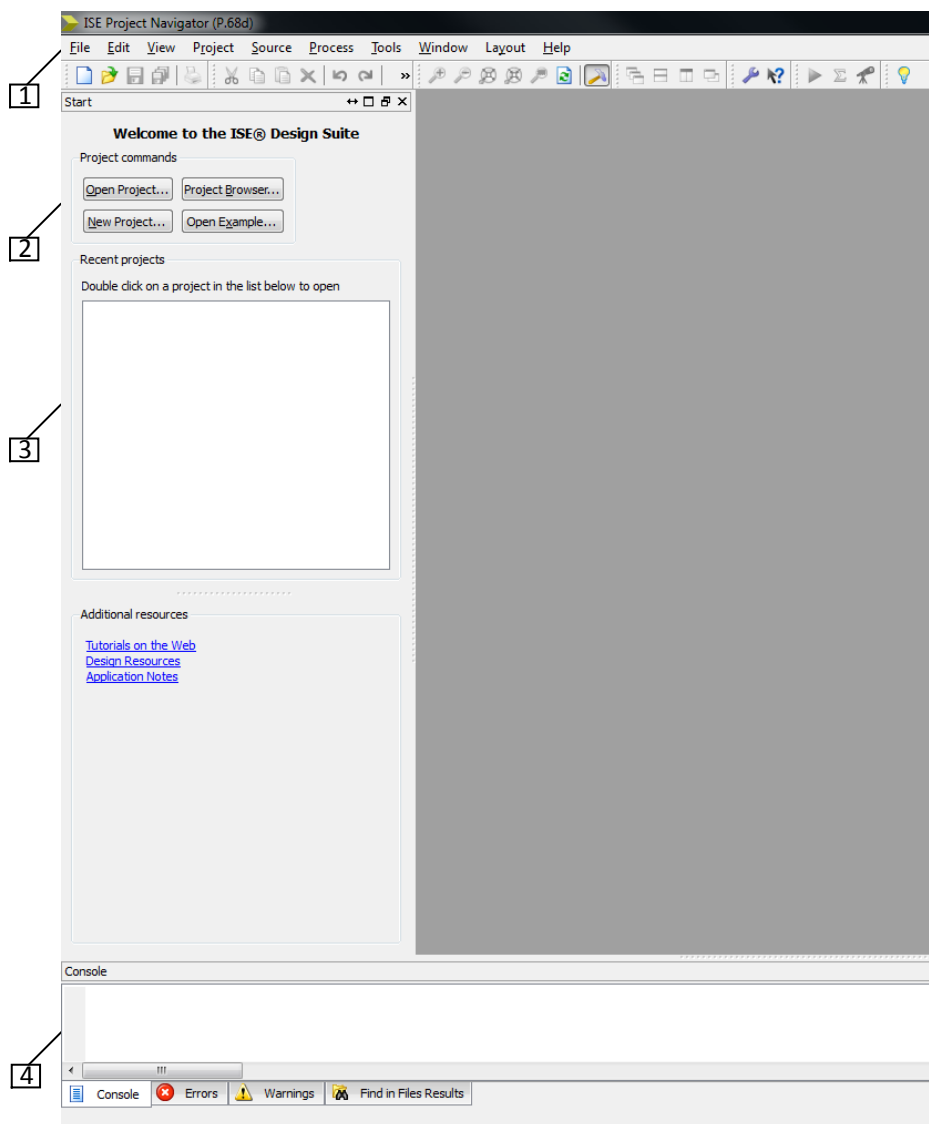


Рисунок 21 – Основные элементы интерфейса главного окна среды проектирования *Xilinx ISE*

Для создания нового проекта в среде *Xilinx ISE* можно воспользоваться пунктом меню *File* → *New Project* либо соответствующей кнопкой в области быстрого доступа *New Project*. Загрузка уже созданного проекта осуществляется аналогичным способом: пункт меню *File* → *Open Project* либо кнопкой *Open Project* из области быстрого доступа.

В качестве демонстрационного проекта далее будет реализована комбинационная схема, описываемая следующей функцией (16):

$$f = \bar{x}_3 \bar{x}_1 \vee x_2 x_1 x_0 \vee x_3 \bar{x}_2 x_1 \bar{x}_0. \quad (16)$$

После нажатия кнопки *New Project* откроется окно *New Project Wizard* – пошаговый мастер создания проектов (рисунок 22).

В данном окне необходимо указать название проекта, месторасположение и рабочую папку для проекта (см. рисунок 22, область 1). В нижней части окна мастера создания проектов находится выпадающее меню *Top-level source type*,

которое должно быть установлено в пункт *Schematic* – создание проекта на основе схмотехнического моделирования (см. рисунок 22, область 2).

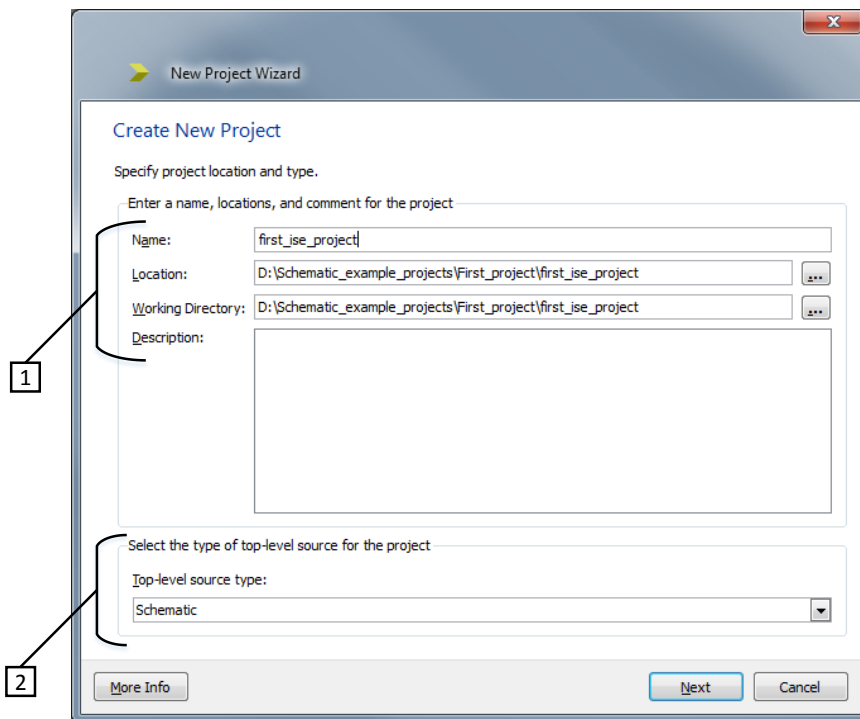


Рисунок 22 – Окно создания нового проекта (*New Project Wizard*)

После выставления настроек в данном окне и нажатия кнопки *Next* в мастере создания проекта откроется второе окно, отвечающее за настройки аппаратной платформы, на базе которой будет производиться моделирование, и выбор программы-симулятора для данного проекта (рисунок 23).

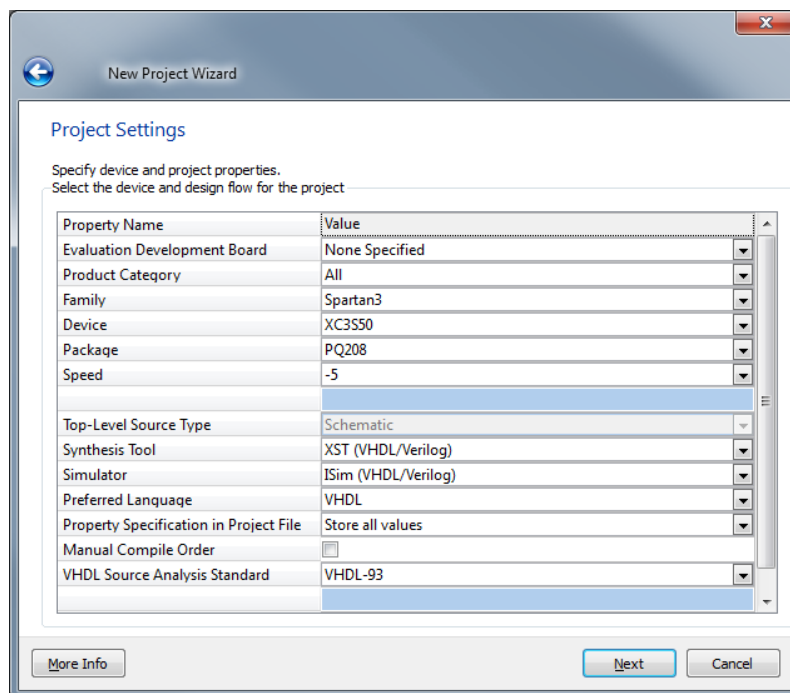


Рисунок 23 – Окно настройки создаваемого проекта

На рисунке 23 показаны рекомендуемые для работы над заданиями курсового проектирования настройки, а именно: семейство кристаллов (*Family*), на базе которого будет проводиться моделирование, – *Spartan 3*; непосредственно устройство (*Device*) – *XC3S50*. Что касается выбора симулятора (пункт *Simulator*), то вариант выбора на рисунке 23 носит не рекомендательный, а обязательный характер – симулятор *ISim*, поскольку данное программное обеспечение является встроенным в среду проектирования *Xilinx ISE*. В дальнейшем изложении материала по работе с данной средой будет использоваться именно симулятор *ISim*.

После выбора необходимых настроек аппаратной платформы и выбора инструмента симуляции, по нажатию кнопки *Next* откроется окно с суммарной информацией о создаваемом проекте (рисунок 24). Если все настройки, отображенные в данном окне, корректны, то после нажатия кнопки *Finish* произойдет создание пустого проекта.

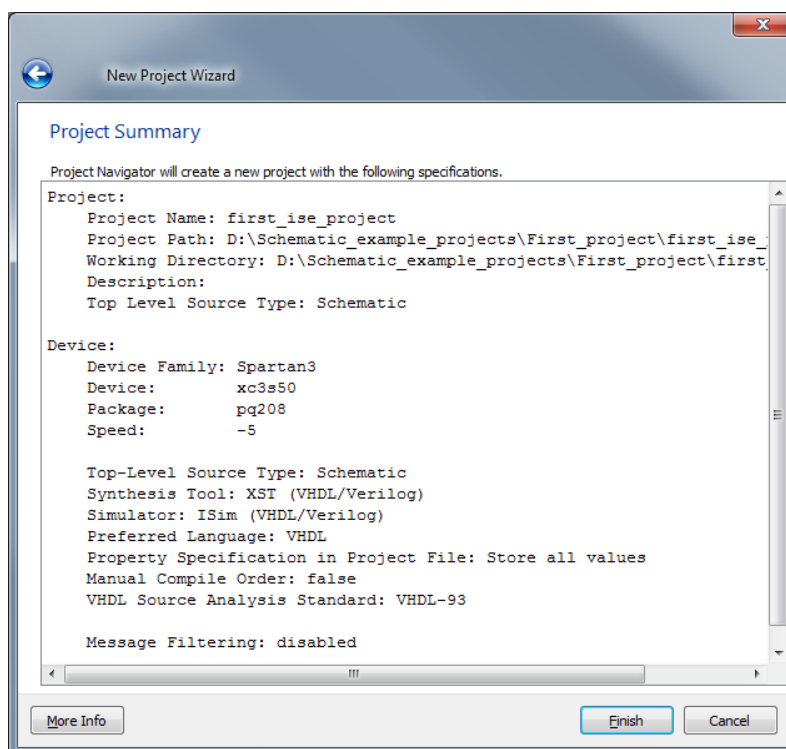


Рисунок 24 – Суммарная информация о создаваемом проекте

После создания проекта, главное окно видоизменится следующим образом: в левой части окна появятся две области, отражающие иерархию файлов проекта (*Hierarchy*), и ниже – различные действия с проектом (рисунок 25).

Для начала работы и построения схемы моделируемого устройства необходимо создать первый файл проекта. Это можно сделать тремя способами:

- 1) выбрать пункт меню *Project* → *New Source*;
- 2) щелкнуть правой кнопкой мыши в левой верхней области главного окна с надписью *Empty Project* и выбрать в выпадающем контекстном меню пункт *New Source*;

3) нажать верхнюю пиктограмму слева от области *Empty Project* (см. рисунок 25, выделена овалом).

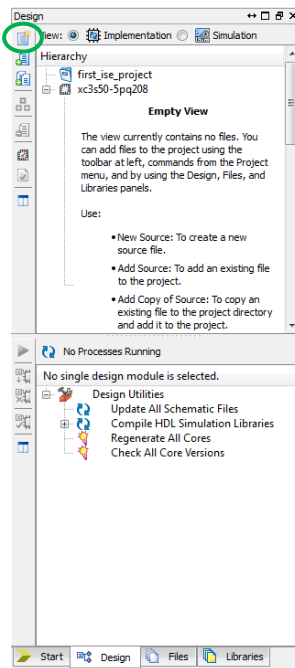


Рисунок 25 – Изменения в главном окне среды проектирования после создания проекта

В любом из приведенных выше вариантов откроется окно (рисунок 26) пошагового мастера создания нового файла проекта (схемы).

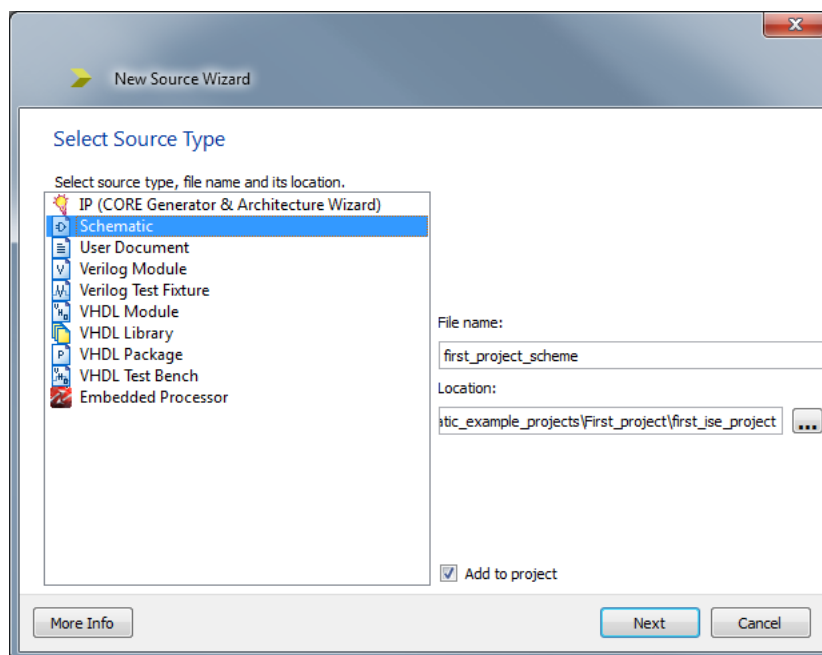


Рисунок 26 – Окно мастера создания нового файла проекта (схемы)

В данном окне необходимо выбрать тип создаваемого исходного файла к проекту *Schematic* для осуществления схмотехнического моделирования. Также на этом шаге указывается имя исходного файла. Поле *Location* заполняется автоматически (в нем указывается рабочая директория проекта). После нажатия

кнопки *Next* появится окно с суммарной информацией о создаваемом файле схемы, аналогичное изображенному на рисунке 24. При нажатии в этом окне кнопки *Finish* будет создан новый файл схемы для текущего проекта, что отобразится в области иерархии проекта (рисунок 27) вместо надписи «*Empty Project*». Так же в центральной области главного окна появится поле создания и редактирования схемы моделируемого устройства, а в области, приведенной на рисунке 25, внизу, добавятся две новые вкладки: *Symbols* и *Options*. Теперь задача разработчика – размещение необходимых компонентов в редакторе схемы и соединение их между собой.

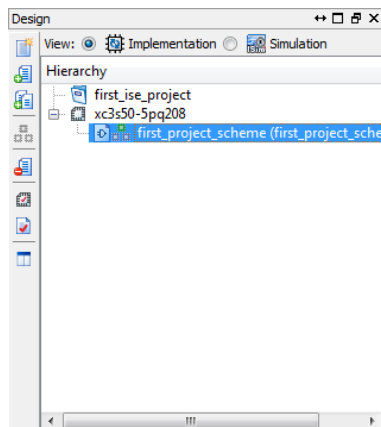


Рисунок 27 – Область иерархии проекта с созданным файлом схемы

В библиотеке среды проектирования *Xilinx ISE* находится большое количество компонентов, которые позволяют произвести моделирование схем различного уровня сложности. Данные компоненты находятся на вкладке *Symbols* (рисунок 28, область 1).

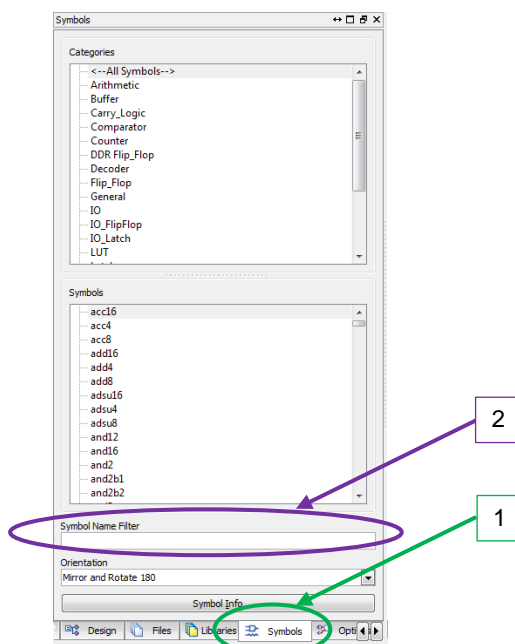


Рисунок 28 – Вкладка *Symbols*

Как видно из рисунка 28, все компоненты разбиты на категории (логические элементы, арифметические, триггеры и т. д.). Для более удобного поиска необходимого элемента библиотеки есть строка фильтра по имени компонента (*Symbol Name Filter*, область 2 на рисунке 28). Под строкой фильтрации имени компонентов расположен пункт выбора пространственной ориентации компонентов, где доступен выбор поворота выбранного элемента на четыре угла (0, 90, 180, 270°), а также зеркального отражения с одновременным поворотом.

Для реализации моделируемой схемы сначала необходимо расположить на поле создания и редактирования логические элементы. Для построения схемы, заданной формулой (16) необходимы элементы: 2И (*AND2*), 3И (*AND3*), 4И (*AND4*), 3ИЛИ (*OR3*) и 3 инвертора – НЕ (*INV*). Все перечисленные элементы находятся в категории *Logic*. Щелчком левой кнопки мыши на названии соответствующего элемента происходит его выбор. Затем, при наведении курсора мыши на поле редактирования схемы, можно увидеть контур выбранного компонента. Повторным щелчком левой кнопки мыши произойдет установка выбранного элемента на схему. На рисунке 29 показано расположение логических элементов для реализации схемы рассматриваемого проекта.

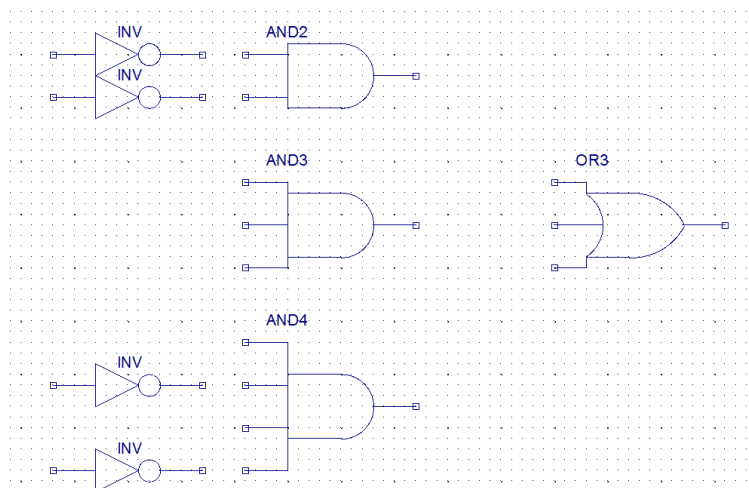



Рисунок 29 – Расположение элементов схемы


Для того, что бы поменять позицию логического элемента на листе схемы можно обратиться к контекстному меню, раздел *Symbol* – там находятся команды зеркального отражения и поворота (*Mirror* и *Rotate*, соответственно). Также, в данном пункте контекстного меню находится пункт вызова справочной информации о компоненте – *Symbol Info*, в котором содержится описание, а также таблица истинности для большинства элементов.

После того, как все необходимые логические элементы из библиотеки *Xilinx ISE* расположены на поле схемы, необходимо произвести редактирование схемы: соединение всех элементов, а также обозначение входных и выходных разрядов. Инструменты для осуществления данных действий находятся в пункте меню *Add*, а также расположены на панели быстрого доступа, находящейся слева от поля редактирования схемы (рисунок 30).



Рисунок 30 – Панель быстрого доступа к инструментам редактирования схемы

Для соединения логических элементов друг с другом нужно воспользоваться инструментом *Add Wire* (, см. рисунок 30, третья сверху пиктограмма). Для соединения двух элементов необходимо после выбора данного инструмента навести курсор на квадратное поле, которым заканчивается вход/выход логического элемента, щелкнуть левую кнопку мыши. Далее, навести курсор мыши на целевой вход/выход элемента (аналогичное квадратное поле) и щелкнуть левой кнопкой мыши на нем. При этом, произойдет создание связи между необходимыми элементами схемы. В случае, если нужно разделить одну связь на несколько логических элементов (общий вход, к примеру), то можно щелкнуть левой кнопкой мыши в любом свободном месте необходимой линии соединения и провести связь, как это описано выше, к целевому логическому элементу.

Для того, что бы обозначить входные и выходные разряды на схеме необходимо воспользоваться инструментом *Add I/O Marker* (, см. рисунок 30, седьмая сверху пиктограмма; либо пункт меню *Add*). Для простановки данного маркера необходимо щелкнуть левой кнопкой мыши на квадратном поле входа/выхода логического элемента либо необходимой линии связи. По умолчанию, среда проектирования сама расставляет имена портов входов/выходов. Для их переименования необходимо щелкнуть на маркере правой кнопкой мыши и выбрать пункт контекстного меню *Rename Port*. После этого появится окно со строкой ввода имени порта входа/выхода.

На рисунке 31 изображена моделируемая схема со всеми связями и обозначенными входными и выходными разрядами.

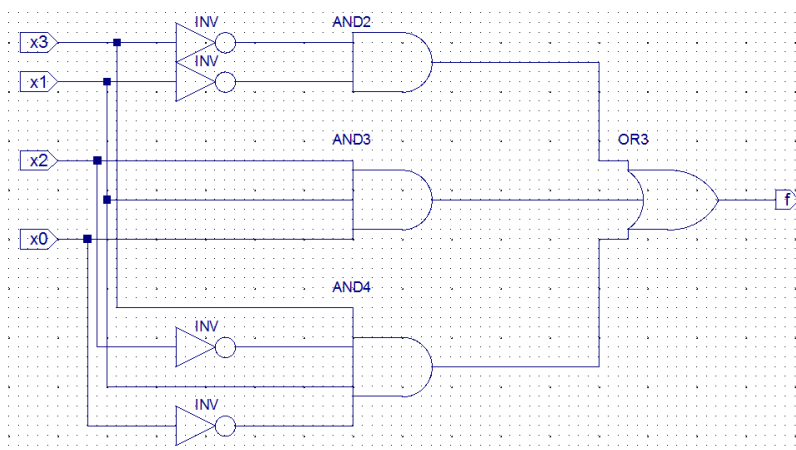


Рисунок 31 – Моделируемая схема

В случае, если необходимо отредактировать схему, а именно, связи между логическими элементами, то однократным щелчком левой кнопки мыши можно выделить нужную связь. Существует два варианта выделения линий связи, переключение между которыми находится на вкладке *Options* (см. рисунок 28, справа от вкладки *Symbols*): выбор линии и всех ответвлений от нее либо выбор только нажатого сегмента данной связи. Также на данной вкладке находятся некоторые другие опции редактирования схемы, например, настройка события, возникающего при перемещении логического элемента по схеме (оставлять связи с ним либо нет).

4.1.2 Симуляция моделируемой схемы с помощью создания файла тестового воздействия

После создания схемы и внесения в нее необходимых правок необходимо ее сохранить и приступить к симуляции ее работы. Для этого в левой области главного экрана среды проектирования над иерархией проекта необходимо переместить переключатель на пункт *Simulation* (рисунок 32, выделено овалом).

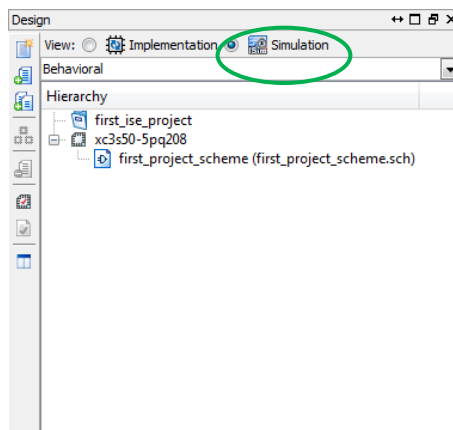


Рисунок 32 – Переключение в режим симуляции работы устройства

Симуляция представляет собой генерацию выходных значений моделируемой схемы как отклик на различные наборы значений входных сигналов. Для того, что бы реализовать последовательную подачу сигналов на входы схемы необходимо создать тестовый файл, в котором будут находиться данные входные значения. Создание тестового файла аналогично созданию файла-схемы: через пункт меню *Project* → *New Source* либо с помощью соответствующей пиктограммы (см. рисунок 25). После совершения данного действия появиться мастер пошагового создания файла (см. рисунок 26), однако теперь необходимо выбрать пункт *VHDL Test Bench*. После совершения необходимых шагов среда *Xilinx ISE* создаст текстовый шаблон тестового файла.

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;

ENTITY first_project_scheme_first_project_scheme_sch_tb IS
END first_project_scheme_first_project_scheme_sch_tb;

ARCHITECTURE behavioral OF first_project_scheme_first_project_scheme_sch_tb IS

    COMPONENT first_project_scheme
    PORT (x3          : IN  STD_LOGIC;
          x1          : IN  STD_LOGIC;
          x2          : IN  STD_LOGIC;
          x0          : IN  STD_LOGIC;
          f           : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL x3          : STD_LOGIC;
    SIGNAL x1          : STD_LOGIC;
    SIGNAL x2          : STD_LOGIC;
    SIGNAL x0          : STD_LOGIC;
    SIGNAL f           : STD_LOGIC;

BEGIN

    UUT: first_project_scheme PORT MAP(
        x3 => x3,
        x1 => x1,
        x2 => x2,
        x0 => x0,
        f  => f
    );

    -- *** Test Bench - User Defined Section ***
    tb : PROCESS
    BEGIN
        WAIT; -- will wait forever
    END PROCESS;
    -- *** End Test Bench - User Defined Section ***

END;
```

Как видно из шаблона тестового файла, среда проектирования автоматически определила сигналы, которые были обозначены на схеме как входные и

выходные. Задачей разработчика является инициализация последовательности подаваемых входных наборов так, чтобы система симуляции автоматически определила значение выходного сигнала и отобразила его зависимости от входных сигналов на временной диаграмме. Для этого в области, границы которой обозначены строками `-- *** Test Bench - User Defined Section ***` и `-- *** End Test Bench - User Defined Section ***`, между зарезервированными словами *BEGIN* и *END PROCESS* необходимо задать все желаемые наборы значений входных сигналов и указать, в течение какого интервала времени симуляции они будут поступать на схему. Так как разрабатываемая в примере схема является функцией от четырех переменных, то для полной ее симуляции необходимо описать 16 двоичных наборов таблицы истинности. Фрагмент тестового файла, отвечающий за данную симуляцию, представлен в таблице 12.

Таблица 12 – Входные наборы для демонстрационной схемы

Наборы 0000–0011	Наборы 0100–0111	Наборы 1000–1011	Наборы 1100–1111
x3 <= '0'; x2 <= '0'; x1 <= '0'; x0 <= '0'; wait for 50 ns;	x3 <= '0'; x2 <= '1'; x1 <= '0'; x0 <= '0'; wait for 50 ns;	x3 <= '1'; x2 <= '0'; x1 <= '0'; x0 <= '0'; wait for 50 ns;	x3 <= '1'; x2 <= '1'; x1 <= '0'; x0 <= '0'; wait for 50 ns;
x3 <= '0'; x2 <= '0'; x1 <= '0'; x0 <= '1'; wait for 50 ns;	x3 <= '0'; x2 <= '1'; x1 <= '0'; x0 <= '1'; wait for 50 ns;	x3 <= '1'; x2 <= '0'; x1 <= '0'; x0 <= '1'; wait for 50 ns;	x3 <= '1'; x2 <= '1'; x1 <= '0'; x0 <= '1'; wait for 50 ns;
x3 <= '0'; x2 <= '0'; x1 <= '1'; x0 <= '0'; wait for 50 ns;	x3 <= '0'; x2 <= '1'; x1 <= '1'; x0 <= '0'; wait for 50 ns;	x3 <= '1'; x2 <= '0'; x1 <= '1'; x0 <= '0'; wait for 50 ns;	x3 <= '1'; x2 <= '1'; x1 <= '1'; x0 <= '0'; wait for 50 ns;
x3 <= '0'; x2 <= '0'; x1 <= '1'; x0 <= '1'; wait for 50 ns;	x3 <= '0'; x2 <= '1'; x1 <= '1'; x0 <= '1'; wait for 50 ns;	x3 <= '1'; x2 <= '0'; x1 <= '1'; x0 <= '1'; wait for 50 ns;	x3 <= '1'; x2 <= '1'; x1 <= '1'; x0 <= '1'; wait;

В таблице 12 приведены все входные наборы таблицы истинности логической функции четырех переменных. После имени входного сигнала указан знак назначения «<=», который показывает системе симуляции, что данный сигнал устанавливается в соответствующее значение. Само двоичное значение указывается в одиночных кавычках после знака назначения. После задания значений каждого набора помещается строка «*wait for 50 ns;*», которая указывает системе симуляции интервал времени, в течение которого необходимо сохранить указанные выше значения входных сигналов. После финального набора 1111 помещается оператор «*wait;*», который указывает, что данное значение необходимо продержать до конца процесса симуляции.

Аналогичный фрагмент тестового файла может быть использован для получения временных диаграмм других комбинационных схем с заменой, если требуется, названий и количества входных сигналов.

Для финального этапа моделирования – построения временной диаграммы, необходимо сохранить написанный тестовый файл и в левой области главного окна программы двойным щелчком левой кнопки мыши на пункте *Simulate Behavioral Model* запустить процесс симуляции работы моделируемой схемы (рисунок 33).

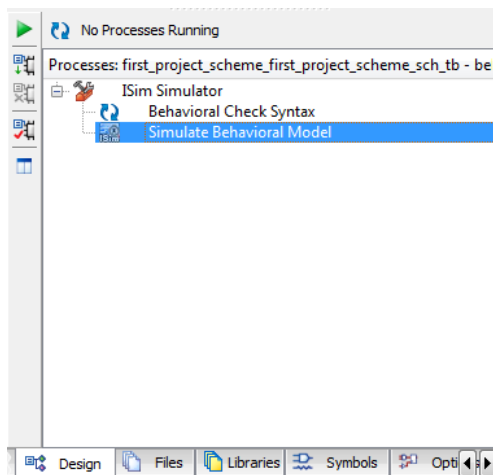


Рисунок 33 – Запуск процесса симуляции работы моделируемой схемы

После этого откроется окно программы симуляции *ISim*, в котором будет отображена временная диаграмма функционирования моделируемой схемы (рисунок 34).

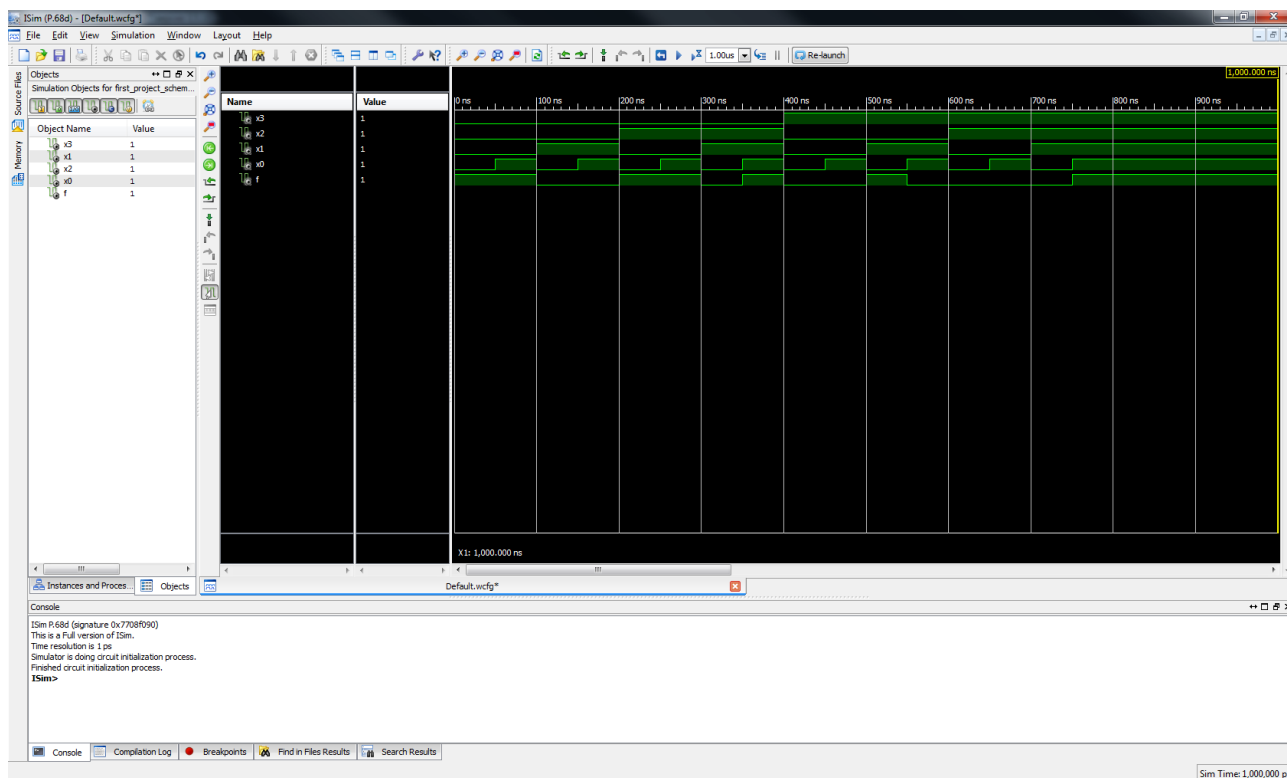


Рисунок 34 – Окно отображения временной диаграммы моделируемой схемы

Центральную область окна, показанного на рисунке 34, занимает непосредственно временная диаграмма. Левее данной области располагаются столбец с текущими значениями входных/выходных сигналов – *Value* и столбец *Name*, в котором отображаются имена всех литералов (входных и выходных маркеров схемы). Для проверки правильности построения и функционирования демонстрационной схемы можно поочередно подставлять входные наборы в формулу (16) и сверять значение функции с соответствующими значениями временной диаграммы. В общем случае, разработка устройства связана с синтезом таблицы истинности (либо в задании она указана в качестве исходных данных), и временная диаграмма является прямым отражением этой таблицы. Временную диаграмму в окне программы симуляции можно масштабировать для более детализированного изучения. Масштаб можно изменять либо обратившись в пункт меню *View→Zoom* и выбрав в нем необходимые действия, либо с помощью соответствующих данному меню пиктограмм на панели инструментов, а также правее столбца с перечнем имен входных/выходных переменных. Для более удобной группировки сигналов их позволено перетаскивать. Для этого необходимо щелкнуть правой кнопкой мыши на имени сигнала в области *Name* и, не отпуская её, произвести перетаскивания объекта.

4.1.3 Симуляция моделируемой схемы с помощью ручного задания значений входных сигналов

Другим способом симуляции работы моделируемой схемы является ручное задание значений входных сигналов. Для этого необходимо перейти в режим симуляции (*Simulation*) таким же образом, как это описано выше в пункте 4.1.2 (см. рисунок 33). Затем, установив указатель мыши на имени моделируемой схемы в области иерархии проекта (*Hierarchy*), двойным щелчком левой кнопки на пункте *Simulate Behavioral Model* в нижней левой области окна вызвать программу симуляции *ISim*. После открытия окна программы симуляции необходимо в области *Name* удалить те имена сигналов, которые являются внутренними для моделируемой схемы, т. е. оставить только входные и выходные сигналы. Для рассматриваемого примера это сигналы с именами x_3, x_2, x_1, x_0, f . Для этого нужно щелкнуть правой кнопкой мыши по названию внутреннего сигнала и в контекстном меню выбрать пункт *Delete* либо выделить сигнал однократным щелчком левой кнопкой мыши и нажать клавишу *Delete (Del)* на клавиатуре. Следующий шаг – перезапуск процесса симуляции. Для этого в пункте меню *Simulation* нужно выбрать пункт *Restart*.

После выполнения описанных выше действий можно приступить к ручному заданию параметров каждому входному сигналу индивидуально. Щелчком правой кнопки мыши на имени сигнала вызвать контекстное меню и в нём выбрать пункт *Force Clock* (рисунок 35).

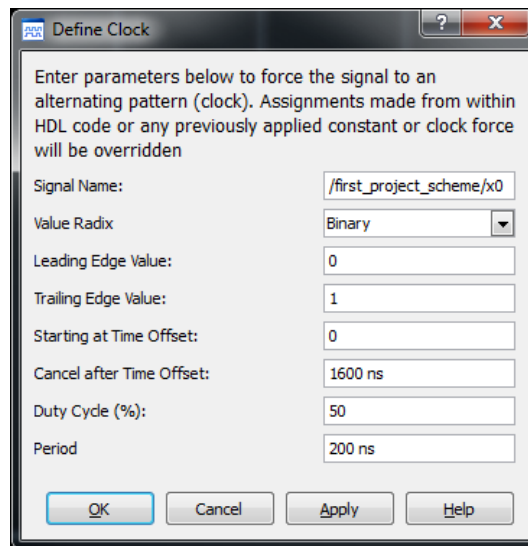


Рисунок 35 – Окно настройки параметров сигнала

В появившемся окне осуществляется настройка параметров каждого сигнала в отдельности. Поскольку для симуляции необходимо чередовать значения двоичных нуля и единицы через определенные интервалы времени, в данном окне будет происходить настройка интервалов чередования.

Поля в окне настройки параметров сигнала имеют следующие значения:

Signal Name – имя настраиваемого сигнала;

Value Radix – основание системы счисления для задания значения данного сигнала. В выпадающем списке для моделируемой схемы необходимо выбрать *Binary* – двоичная система счисления;

Leading Edge Value – в данном случае, это значение начала отсчета при моделировании. Поскольку первым набором является двоичный эквивалент десятичной цифры ноль, то это значение для всех сигналов выставляется в ноль;

Trailing Edge Value – в данном случае, это значение, сменяющее предыдущее состояние сигнала. Поскольку настраивается двоичный сигнал с временным чередованием нуля и единицы, в этом поле необходимо указать «1»;

Starting at Time Offset – начало симуляции (во времени). Рекомендуется оставить значение по умолчанию – ноль;

Cancel after Time Offset – время завершения симуляции. Поскольку для каждого сигнала смена нуля единицей – это периодический процесс, то рекомендуется рассчитать значение данного параметра, отталкиваясь от значения периода для младшего разряда набора. В частности, в рассматриваемом примере симулируется работа устройства, описываемого логической функцией четырех переменных, следовательно, младший разряд восемь раз принимает нулевое значение и столько же – единичное для того, чтобы перебрать всю таблицу истинности для четырех переменных. Если установить длительность каждого двоичного значения равным 100 нс (ns), то время завершения симуляции составит 1600 нс. В данном поле временное значение вносится с указанием единиц измерения;

Duty Cycle – коэффициент заполнения периода, который показывает, какую часть периода значение сигнала будет оставаться нулем, а какую – единицей. Рекомендуется оставить значение по умолчанию – 50 %;

Period – период сигнала. Для того чтобы, как описано выше, длительность каждого двоичного значения составляла 100 нс, длительность всего периода должна быть равна 200 нс (при условии, что параметр *Duty Cycle* установлен по умолчанию в 50 %).

Вышеприведенные настройки необходимо сделать для каждого входного сигнала моделируемой схемы. В данном примере для всех сигналов все поля, кроме *Period*, будут совпадать. Что касается периода, то с возрастанием номера разряда набора на единицу в сторону старшего, данное значение удваивается, т. е. для x_0 – это 200 нс, x_1 – 400 нс, x_2 – 800 нс, x_3 – 1600 нс.

После полной настройки всех входных сигналов необходимо произвести запуск симуляции. Это можно сделать через пункт меню *Simulation* → *Run All*. После этого в окне программы симуляции отобразится временная диаграмма, приведенная на рисунке 36.

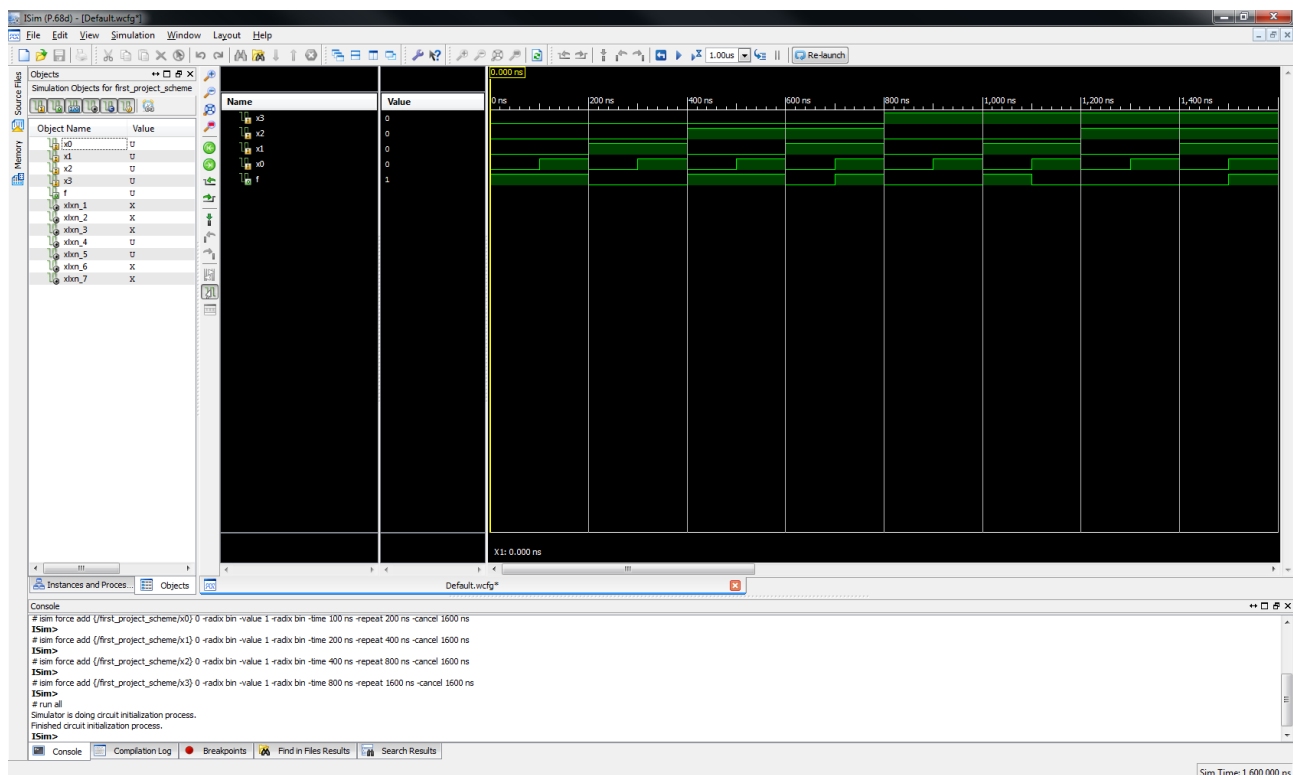


Рисунок 36 – Окно отображения временной диаграммы моделируемой схемы

Подробно изучить возможности и порядок работе в среде проектирования *Xilinx ISE* можно с помощью обучающего руководства фирмы *Xilinx* [14].

4.2 Примеры моделирования схем в среде проектирования *Xilinx ISE*

4.2.1 Комбинационный узел на основе мультиплексора

Рассмотрим реализацию на основе мультиплексора «1 из 4» логической функции трёх переменных, заданной таблицей истинности (таблица 13).

Таблица 13 – Таблица истинности реализуемой логической функции

x_2	x_1	x_0	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

При реализации комбинационного узла на основе мультиплексора необходимо определить, какие входные переменные логической функции будут поступать на адресные, а какие на информационные входы мультиплексора.

Для того чтобы определить, в каком случае не будет нужен дополнительный инвертор, нужно построить карту Карно и записать по ней минимальную форму функции. После этого, анализируя выражение, нужно узнать, какая из переменных входит во все конъюнкции без инверсии – ее можно подавать на один из информационных входов мультиплексора, тогда оставшиеся переменные нужно подключить к адресным входам.

Карта Карно для заданной логической функции представлена на рисунке 37.

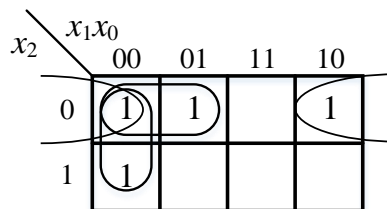


Рисунок 37 – Карта Карно для заданной логической функции

После минимизации получим следующую минимальную форму заданной логической функции: $f = \bar{x}_2\bar{x}_0 \vee \bar{x}_2\bar{x}_1 \vee \bar{x}_1\bar{x}_0$. Из данного выражения видно, что все переменные встречаются в инверсном виде. Поэтому для реализации функции потребуется дополнительный инвертор. Поскольку в данном случае не имеет значения, какие переменные подавать на адресные входы мультиплексора, то к ним подключим переменные x_2 и x_1 , а переменную x_0 будем использовать для определения значений, подключаемых к информационным входам. Для этого разобьем таблицу 13 на четыре части по две строки. Сравнивая значение переменной x_0 со значением функции f , определим, что нужно подавать на входы данных мультиплексора (таблица 14).

Таблица 14 – Значения, подключаемые на входы данных мультиплексора

x_2	x_1	x_0	f	D
0	0	0	1	$D0 = 1$
0	0	1	1	
0	1	0	1	$D1 = \bar{x}_0$
0	1	1	0	
1	0	0	1	$D2 = \bar{x}_0$
1	0	1	0	
1	1	0	0	$D3 = 0$
1	1	1	0	

Схема моделируемого устройства представлена на рисунке 38.

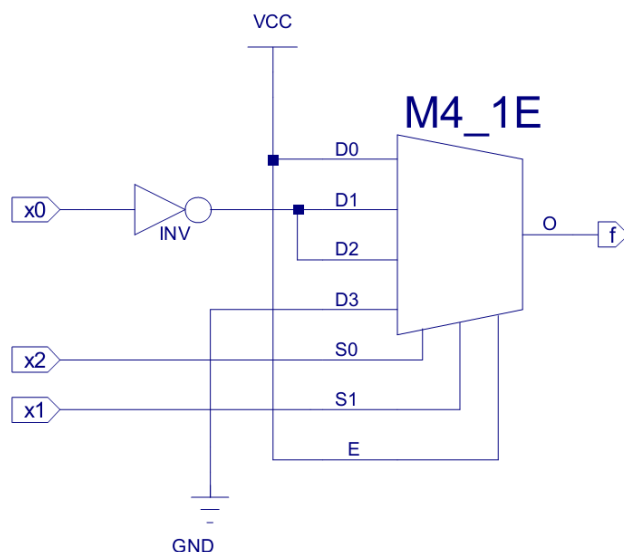


Рисунок 38 – Моделируемая схема комбинационного узла на основе мультиплексора

Мультиплексоры в среде проектирования *Xilinx ISE* находятся в категории *Mux*. Для моделирования схемы на информационный вход $D0$ и на служебный – E необходимо подать логическую единицу, т. е. напряжение высокого уровня. Для этого, в категории *General* есть компонент VCC . Что касается логического нуля (низкий уровень), то данный элемент находится в той же категории, под названием GND . Таблица истинности для используемого мультиплексора приведена в пункте контекстного меню *Symbol* → *Symbol Info*.

Для выполнения процесса симуляции моделируемого устройства необходимо подготовить тестовый файл со следующим исходным текстом.

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.vcomponents.ALL;
ENTITY MUX_based_function_MUX_based_function_sch_tb IS
END MUX_based_function_MUX_based_function_sch_tb;
ARCHITECTURE behavioral OF MUX_based_function_MUX_based_function_sch_tb IS
    COMPONENT MUX_based_function
        PORT( x2 :      IN      STD_LOGIC;
              x1      :      IN      STD_LOGIC;
              x0      :      IN      STD_LOGIC;

```

```

        f :      OUT      STD_LOGIC);
END COMPONENT;

SIGNAL x2      :      STD_LOGIC;
SIGNAL x1      :      STD_LOGIC;
SIGNAL x0      :      STD_LOGIC;
SIGNAL f :      STD_LOGIC;

BEGIN

    UUT: MUX_based_function PORT MAP(
        x2 => x2,
        x1 => x1,
        x0 => x0,
        f => f
    );

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN

    x2 <= '0';
    x1 <= '0';
    x0 <= '0';
    wait for 50 ns;

    x2 <= '0';
    x1 <= '0';
    x0 <= '1';
    wait for 50 ns;
    x2 <= '0';
    x1 <= '1';
    x0 <= '0';
    wait for 50 ns;

    x2 <= '0';
    x1 <= '1';
    x0 <= '1';
    wait for 50 ns;

    x2 <= '1';
    x1 <= '0';
    x0 <= '0';
    wait for 50 ns;

    x2 <= '1';
    x1 <= '0';
    x0 <= '1';
    wait for 50 ns;

    x2 <= '1';
    x1 <= '1';
    x0 <= '0';
    wait for 50 ns;

    x2 <= '1';
    x1 <= '1';
    x0 <= '1';
    wait;
END PROCESS;
-- *** End Test Bench - User Defined Section ***

```


END;

На рисунке 39 показана временная диаграмма симуляции моделируемого устройства.

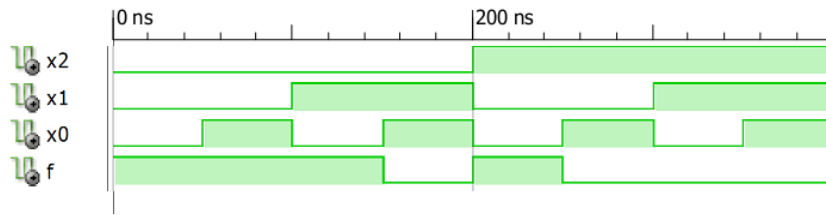


Рисунок 39 – Временная диаграмма симуляции моделируемого устройства

4.2.2 Пересчетное устройство

Рассмотрим моделирование генератора чисел 7 – 2 – 4 – 11, построенного на базе четырех JK-триггеров, синтез которого был описан в примере 3.3.6 (см. подраздел 3.3).

Поскольку в библиотеке элементов среды проектирования *Xilinx ISE* нет триггеров с инверсным входом *K*, то для моделирования перед каждым соответствующим входом необходимо установить инвертор. Схема моделируемого устройства приведена на рисунке 40.

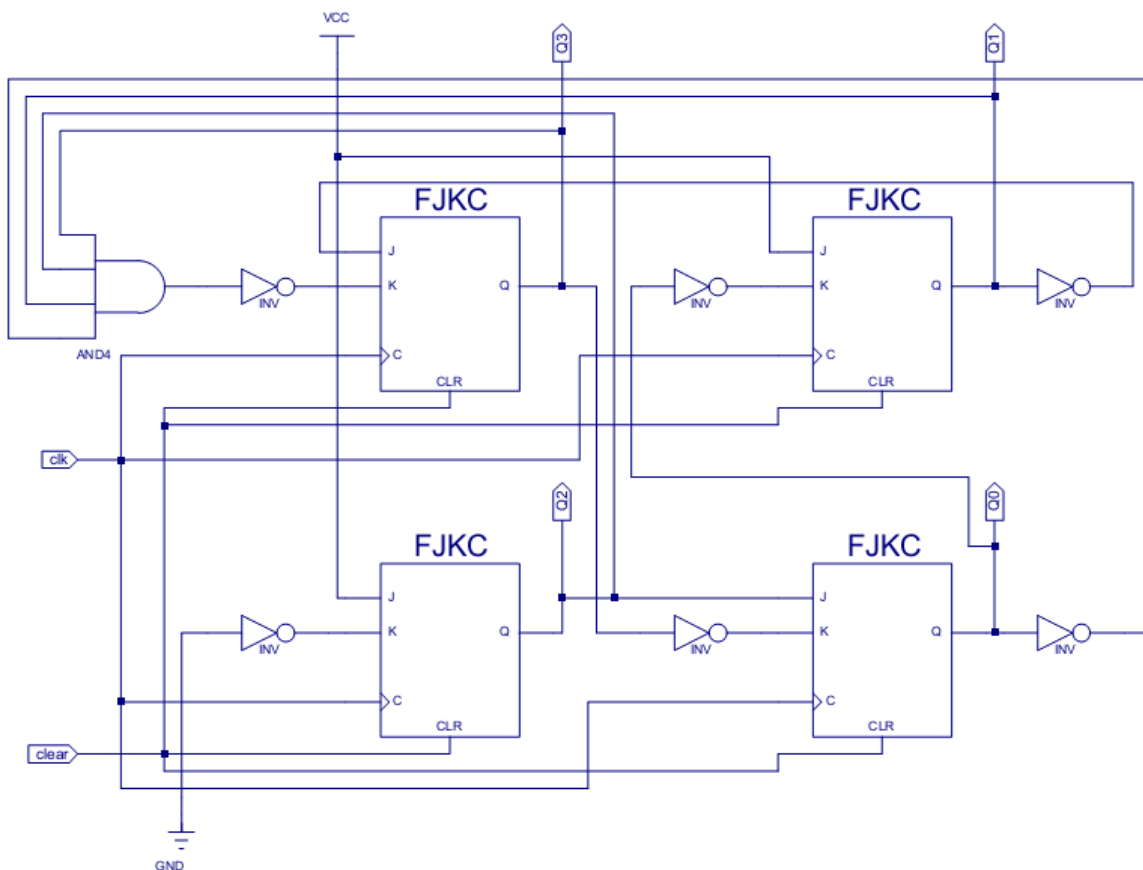


Рисунок 40 – Моделируемая схема генератора чисел 7 – 2 – 4 – 11

Для ее построения были использованы JK-триггеры, которые находятся в категории *Flip_Flop* среды проектирования. Таблица переходов и назначения

входа *CLR* описана в контекстном меню в пункте *Symbol* → *Symbol Info*. Вход *CLR* является асинхронным сбросом (очистка, *clear*), который по высокому уровню, подаваемому на него, производит очистку триггеров и устанавливает значение низкого уровня на выходе триггера *Q*. Данная схема имеет два входных сигнала: описанный выше сигнал сброса и сигнал синхронизации (сигнал тактовых импульсов *clk*).

Для симуляции работы моделируемой схемы необходимо в файле тестового воздействия описать процесс генерации тактового сигнала. Это делается путем объявления константы с именем *clk_period*, значение которой задает длительность одного периода тактового сигнала. Кроме этого, после зарезервированного слова *BEGIN* нужно добавить блок, который отвечает непосредственно за генерацию всего тактового сигнала. Для этого сигналу тактового импульса назначаются значения нуля и единицы с интервалом в половину длительности периода синхронизации.

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY Generator_scheme_Generator_scheme_sch_tb IS
END Generator_scheme_Generator_scheme_sch_tb;
ARCHITECTURE behavioral OF Generator_scheme_Generator_scheme_sch_tb IS

    COMPONENT Generator_scheme
    PORT( clk      :      IN      STD_LOGIC;
          clear    :      IN      STD_LOGIC;
          Q3 :      OUT      STD_LOGIC;
          Q2 :      OUT      STD_LOGIC;
          Q0 :      OUT      STD_LOGIC;
          Q1 :      OUT      STD_LOGIC);
    END COMPONENT;

    SIGNAL clk      :      STD_LOGIC;
    SIGNAL clear    :      STD_LOGIC;
    SIGNAL Q3      :      STD_LOGIC;
    SIGNAL Q2      :      STD_LOGIC;
    SIGNAL Q0      :      STD_LOGIC;
    SIGNAL Q1      :      STD_LOGIC;

    constant clk_period : time := 10 ns;    --длительность периода
                                           --тактового сигнала

BEGIN

    --блок, отвечающий за генерацию тактового импульса
    process
    begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    UUT: Generator_scheme PORT MAP(
        clk => clk,
        clear => clear,

```

```

        Q3 => Q3,
        Q2 => Q2,
        Q0 => Q0,
        Q1 => Q1
    );

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
    clear <= '0';
    WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Временная диаграмма симуляции моделируемого устройства показана на рисунке 41.

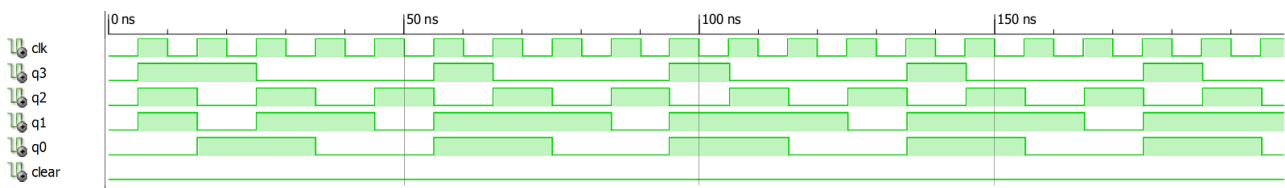


Рисунок 41 – Временная диаграмма симуляции генератора чисел 7 – 2 – 4 – 11

4.2.3 Сдвиговый регистр

Рассмотрим моделирование сдвигового регистра, имеющего последовательность состояний 1 – 3 – 7 – 6 – 4. Для построения устройства будут использоваться три последовательно включенных *D*-триггера. Задача синтеза такого регистра сдвига сводится к получению логической функции обратной связи для формирования необходимого входного значения для триггера младшего разряда. Для этого построим таблицу истинности для функции обратной связи сдвигового регистра с указанием избыточных наборов (таблица 15).

Таблица 15 – Таблица истинности функции обратной связи регистра сдвига

Q_2	Q_1	Q_0	D_r
0	0	1	1
0	1	1	1
1	1	1	0
1	1	0	0
1	0	0	1
0	0	0	1
0	1	0	X
1	0	1	X

Для минимизации функции обратной связи сдвигового регистра по таблице истинности строим карту Карно (рисунок 42).

	$Q_1 Q_0$		
	00	01	11
Q_2	0	1	X
	1	X	0

Рисунок 42 – Карта Карно для функции обратной связи сдвигового регистра

По карте Карно записываем минимальную форму функции обратной связи сдвигового регистра: $D_r = \overline{Q_2} \vee \overline{Q_1} = \overline{Q_2 Q_1}$.

Схема моделируемого сдвигового регистра показана на рисунке 43.

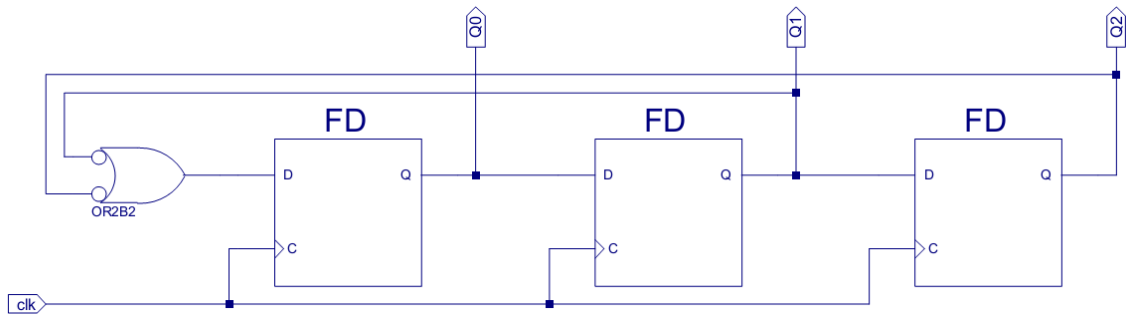


Рисунок 43 – Моделируемая схема сдвигового регистра

Для моделирования регистра сдвига можно использовать файл тестового воздействия из пункта 4.2.2, оставив только один входной сигнал – сигнал тактовых импульсов (*clk*).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY Register_scheme_Register_scheme_sch_tb IS
END Register_scheme_Register_scheme_sch_tb;
ARCHITECTURE behavioral OF Register_scheme_Register_scheme_sch_tb IS
```

```
COMPONENT Register_scheme
PORT( clk      : IN   STD_LOGIC;
      Q0       : OUT  STD_LOGIC;
      Q1       : OUT  STD_LOGIC;
      Q2       : OUT  STD_LOGIC);
END COMPONENT;
```

```
SIGNAL clk      : STD_LOGIC;
SIGNAL Q0       : STD_LOGIC;
SIGNAL Q1       : STD_LOGIC;
SIGNAL Q2       : STD_LOGIC;
```

```
constant clk_period : time := 10 ns;
BEGIN
```

```
process
begin
```

```

        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;
    UUT: Register_scheme PORT MAP(
        clk => clk,
        Q0 => Q0,
        Q1 => Q1,
        Q2 => Q2
    );

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
    WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Временная диаграмма симуляции работы моделируемого сдвигового регистра приведена на рисунке 44.

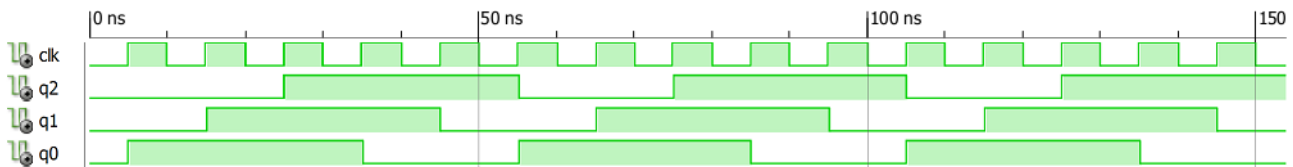



Рисунок 44 – Временная диаграмма симуляции сдвигового регистра

4.2.4 Моделирование устройств на основе сумматоров

Сумматоры, как и другие элементы, выполняющие арифметические операции находятся в среде проектирования *Xilinx ISE* в категории *Arithmetic*. Модели устройств суммирования представлены 4-, 8- и 16-разрядными примитивами. Принцип работы с ними аналогичен принципу, описанному в предыдущих примерах, за исключением 8- и 16-разрядных сумматоров, поскольку для них входы операндов и выход результата оформляются в виде шины. Поэтому в данном примере дополнительно будет показано, каким образом оформляются шины и как сделать подключение линии связи к ним.

Для моделирования будем использовать 8-разрядный сумматор. Согласно спецификации на данный сумматор (контекстное меню, пункт *Symbol* → *Symbol Info*), элемент содержит три входа: операнд *A* (8 разрядов), операнд *B* (8 разрядов), вход переноса *CI*, выход для результата *S* (8 разрядов), а также выходы переноса *CO* и переполнения *OFL*. Работа с входом распространения переноса не отличается от подключения логических элементов, описанных в предыдущих примерах, поскольку это обычный сигнал. Для моделирования на него будет подаваться низкий уровень (цепь *GND*). Чтобы подать на входы сумматора слагаемые, необходимо использовать шинное подключение. Реализуется это аналогично подключению обычного сигнала с помощью инструмента *Add Wire*,

однако после формирования линии связи ей необходимо присвоить имя. Это можно сделать с помощью пункта меню *Add* → *Net Name* либо соответствующего значка на панели инструментов (, см. рисунок 30). После выбора данного инструмента автоматически отобразится вкладка настройки опций, в которой необходимо задать корректное имя шины (латинские буквы и номер старшего и младшего разрядов в круглых скобках через знак двоеточия), как показано на рисунке 45.

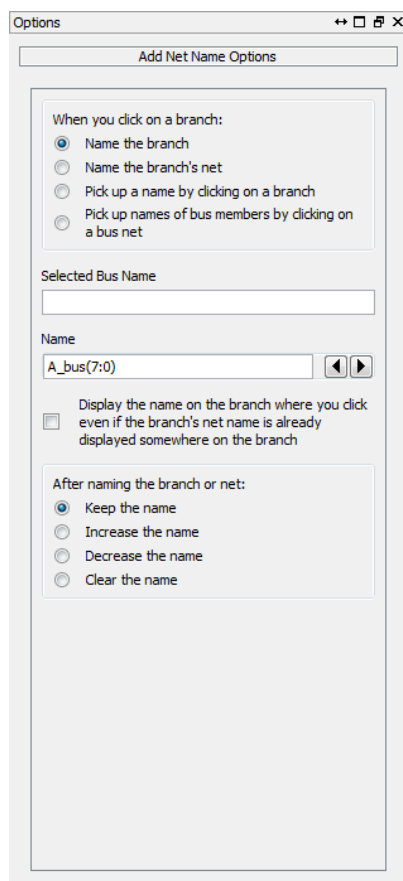



Рисунок 45 – Задание имени шины

После настройки имени, необходимо щелчком левой кнопки мыши в любом месте шины установить заданную метку. Далее на оконечной части шины необходимо установить маркер входов/выходов (инструмент *Add I/O Marker*). В отличие от работы с обычными линиями связи, в случае с шинами имена входных и выходных маркеров сформируются автоматически в зависимости от имени шины.

Для того, чтобы выделить сигнал конкретного разряда из общей шины в среде проектирования *Xilinx ISE* в пункте меню *Add* есть инструментарий *Bus Tap*. Также его можно вызвать с помощью соответствующего значка на панели инструментов (, см. рисунок 30). После выбора данного инструмента в правой области окна отобразятся его свойства (рисунок 46).

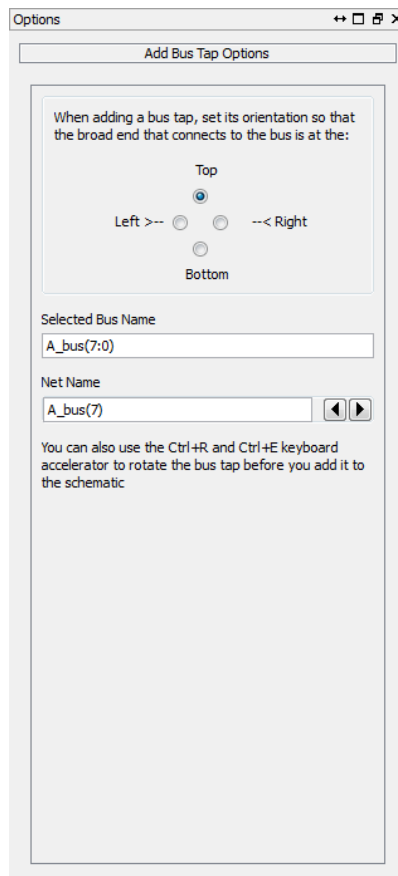


Рисунок 46 – Настройка инструмента *Bus Tap*

Для того, что бы в поле *Selected Bus Name* автоматически отразилось имя шины, от которой необходимо вывести сигнал, нужно навести на нее курсор и щелкнуть левой кнопкой мыши. В области *Net Name* будет указан разряд шины, который будет отделен от нее. Для того, что бы его отредактировать, можно воспользоваться кнопками со стрелками справа от области либо откорректировать значения прямо в строке, выставив нужный номер разряда с помощью клавиатуры. Так же в области настройки данного инструмента можно выбрать его пространственную ориентацию относительно шины. Далее, когда элемент разряда шины настроен, для его установки необходимо щелкнуть левой кнопкой мыши на свободное место около целевой шины на моделируемой схеме. Элемент широкой частью автоматически закрепится к нужной шине, а к противоположной части можно подключать любой элемент, на вход которого должен поступать данный сигнал.

На рисунке 47 приведена схема, демонстрирующая использование сумматора, подключение шин и выделение из шины одного разряда.

На схеме шинам входных операндов назначены имена *A_bus(7:0)* и *B_bus(7:0)*, шине выходного результата – *S_bus(7:0)*. Кроме сумматора в схему добавлен логический элемент И, на входы которого подаются младшие разряды входных шин (операндов).

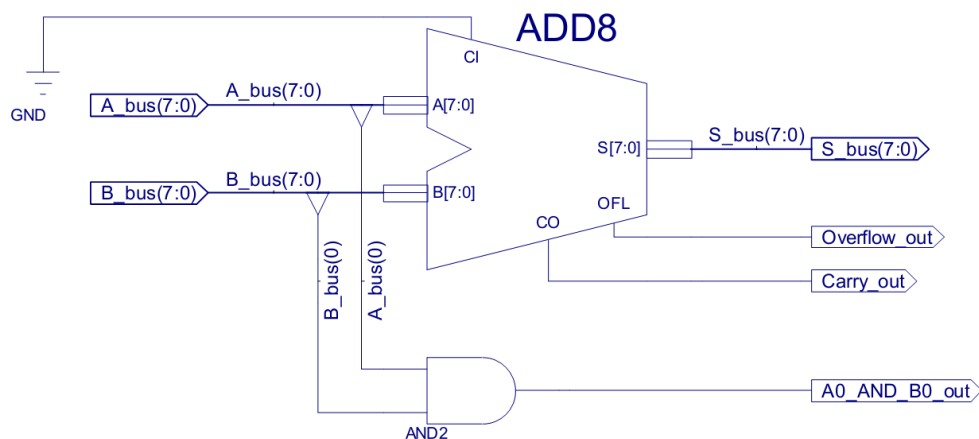


Рисунок 47 – Схема подключения сумматора

Для выполнения процесса симуляции сумматора необходимо подготовить тестовый файл со следующим исходным текстом. В примере показано, каким образом происходит назначение конкретного значения многоразрядному сигналу – поразрядно, между кавычками.

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY Sum_and_bus_project_Sum_and_bus_project_sch_tb IS
END Sum_and_bus_project_Sum_and_bus_project_sch_tb;
ARCHITECTURE behavioral OF Sum_and_bus_project_Sum_and_bus_project_sch_tb IS

    COMPONENT Sum_and_bus_project
    PORT( A_bus      :    IN      STD_LOGIC_VECTOR (7 DOWNTO 0);
          B_bus      :    IN      STD_LOGIC_VECTOR (7 DOWNTO 0);
          S_bus      :    OUT     STD_LOGIC_VECTOR (7 DOWNTO 0);
          Overflow_out :    OUT   STD_LOGIC;
          Carry_out  :    OUT   STD_LOGIC;
          A0_AND_B0_out :    OUT  STD_LOGIC);
    END COMPONENT;

    SIGNAL A_bus      :    STD_LOGIC_VECTOR (7 DOWNTO 0);
    SIGNAL B_bus      :    STD_LOGIC_VECTOR (7 DOWNTO 0);
    SIGNAL S_bus      :    STD_LOGIC_VECTOR (7 DOWNTO 0);
    SIGNAL Overflow_out :    STD_LOGIC;
    SIGNAL Carry_out  :    STD_LOGIC;
    SIGNAL A0_AND_B0_out :    STD_LOGIC;

BEGIN

    UUT: Sum_and_bus_project PORT MAP(
        A_bus => A_bus,
        B_bus => B_bus,
        S_bus => S_bus,
        Overflow_out => Overflow_out,
        Carry_out => Carry_out,
        A0_AND_B0_out => A0_AND_B0_out
    );

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN

    A_bus <= "00101010";

```



```

B_bus <= "01010110";
wait for 50 ns;

A_bus <= "10100001";
B_bus <= "11011010";
wait for 50 ns;

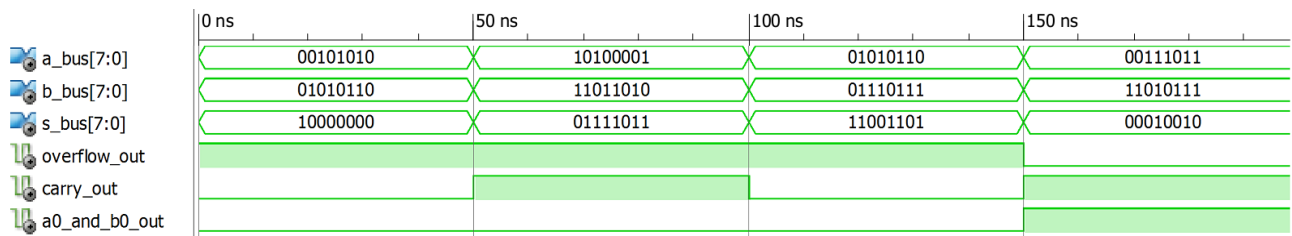
A_bus <= "01010110";
B_bus <= "01110111";
wait for 50 ns;

A_bus <= "00111011";
B_bus <= "11010111";
wait; -- will wait forever

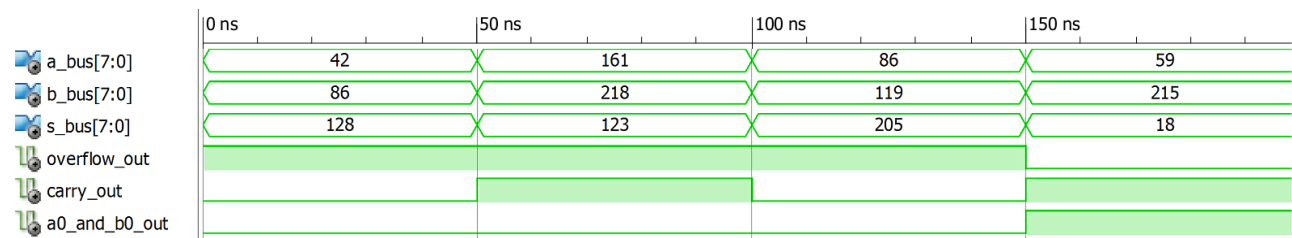
END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

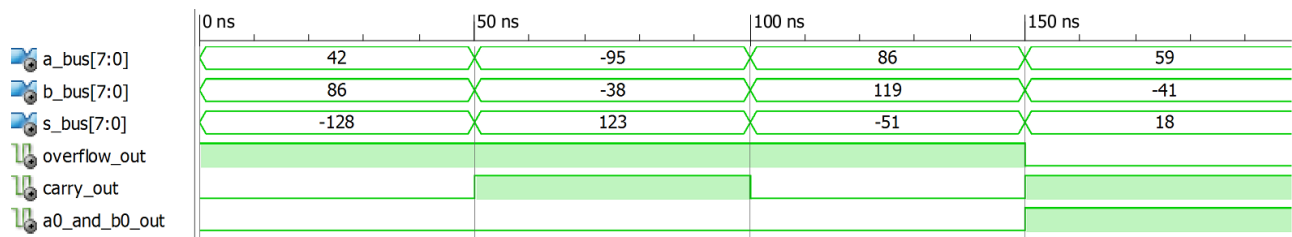
На рисунке 48 приведена временная диаграмма процесса симуляции работы сумматора.



a



б



в

a – операнды и результат в двоичной форме;

б – операнды и результат в десятичной форме: интерпретируются как беззнаковые числа;

в – операнды и результат в десятичной форме: интерпретируются как числа со знаком в дополнительном коде

Рисунок 48 – Временная диаграмма симуляции работы сумматора

На временной диаграмме, кроме вычисления суммы входных операндов, показано формирование выходных сигналов переполнения и распространения переноса в зависимости от того, как интерпретируются входные операнды и результат арифметической операции. Входные операнды для данного сумматора могут интерпретироваться как беззнаковые числа, так и числа со знаком в дополнительном коде. В случае беззнаковых входных операндов наличие переполнения на выходе нужно проверять по сигналу выходного переноса *CO*, для знаковых чисел – по сигналу переполнения *OFL*.

На временной диаграмме также показан выходной сигнал конъюнктора (*a0_and_b0_out*), на входы которого подаются младшие разряды входных слагаемых.

ПРИЛОЖЕНИЕ А

(обязательное)

Перечень типовых заданий курсового проекта

А.1 Дешифраторы, демультимплексоры, шифраторы, преобразователи кодов

А.1.1 На базе ИМС SN74ALS155 (КР1533ИД4) и логических схем этой же серии построить полный двухступенчатый дешифратор с прямыми выходами ([9], с. 3; [11], с. 120–128).

А.1.2 Каскадным включением только дешифраторов ИМС SN74ALS155 (КР1533ИД4) построить 4-разрядный полный двоичный дешифратор ([9], с. 3–4).

А.1.3 Построить 6-разрядный полный двоичный дешифратор на базе ИМС SN74ALS154 (КР1533ИД3) и инверторов той же серии ([2], с. 100–103).

А.1.4 На базе ИМС SN74LS155 (КР1533ИД4) реализовать демультимплексор «1 в 16». Привести таблицу истинности ([9], с. 4).

А.1.5 Построить 4-разрядный полный двоичный дешифратор на базе дешифраторов ИМС SN74LS145 (К555ИД10) и инверторов той же серии ([2], с. 100–103).

А.1.6 Реализовать с помощью дешифратора ИМС SN74ALS154 (КР1533ИД3) и элементов И-НЕ той же серии логическую функцию $y = x_1x_2x_3 \vee x_1\bar{x}_2 \vee \bar{x}_1x_2\bar{x}_3x_4$ ([6], с. 86–87; [9], с. 8).

А.1.7 Построить преобразователь трехразрядного двоичного кода в код Грея на элементах серии SN74ALS (КР1533), выполняющих функции: а) И-НЕ; б) И-ИЛИ-НЕ ([9], с. 8–9; [11], с. 113–114).

А.1.8 Построить схему преобразования двоично-десятичного кода 8421 в код семисегментного индикатора на элементах И-НЕ серии SN74ALS (КР1533) ([11], с. 115–117).

А.1.9 Построить шифратор унитарного кода в двоично-десятичный код 8421 на элементах серии SN74ALS (КР1533): а) И-НЕ; б) ИЛИ-НЕ ([9], с. 5–6).

А.1.10 Построить схему двоичного компаратора чисел $A = a_1a_2a_3a_4$ и $B = b_1b_2b_3b_4$ с выходами $F_{A=B}$, $F_{A<B}$, $F_{A>B}$ ([6], с. 104–105). Схему реализовать на элементах И-НЕ серии SN74ALS (КР1533).

А.1.11 Для 4-разрядного информационного слова $A = a_3a_2a_1a_0$ построить на элементах серии SN74ALS (КР1533) схему кодирования и декодирования на основе кода Хемминга с минимальным кодовым расстоянием, равным 4 ([6], с. 112–116).

А.1.12 В соответствии с номером варианта построить преобразователь кодов ([9], с. 8–9), заданный таблицами А.1 и А.2 на ИМС серии SN74ALS (КР1533). Учесть при минимизации избыточные наборы. Входной и выходной коды, заданные в таблице А.1, выбираются из таблицы А.2. Например, код вход-выход 1-2 из таблицы А.1 в варианте 1 означает в соответствии с таблицей А.2 входной код 8421 (столбец 1), выходной код 7421 (столбец 2).

Таблица А.1 – Варианты преобразователей кодов (к заданию А.1.12)

Номер варианта	Код вход-выход	Номер варианта	Код вход-выход	Номер варианта	Код вход-выход	Номер варианта	Код вход-выход
1	1-2	19	2-4	37	3-7	55	5-7
2	2-1	20	4-2	38	7-3	56	7-5
3	1-3	21	2-5	39	3-8	57	5-8
4	3-1	22	5-2	40	8-3	58	8-5
5	1-4	23	2-6	41	3-9	59	5-9
6	4-1	24	6-2	42	9-3	60	9-5
7	1-5	25	2-7	43	4-5	61	6-7
8	5-1	26	7-2	44	5-4	62	7-6
9	1-6	27	2-8	45	4-6	63	6-8
10	6-1	28	8-2	46	4-4	64	8-6
11	1-7	29	2-9	47	4-7	65	6-9
12	7-1	30	9-2	48	7-4	66	9-6
13	1-8	31	3-4	49	4-8	67	7-8
14	8-1	32	4-3	50	8-4	68	8-7
15	1-9	33	3-5	51	4-9	69	7-9
16	9-1	34	5-3	52	9-4	70	9-7
17	2-3	35	3-6	53	5-6	71	8-9
18	3-2	36	6-3	54	6-5	72	9-8

Таблица А.2 – Двоичные коды (к заданию А.1.12)

Десятичная цифра А	Код 8421	Код 7421	Код 5421	Код 2421	Код Грея	Код с избытком 3 «А + 3»	Дополнение до 9 «9 – А»	Дополнение до 10 «10 – А»	Код Джонсона
	1	2	3	4	5	6	7	8	9
0	0000	0000	0000	0000	0000	0011	1001	1010	00000
1	0001	0001	0001	0001	0001	0100	1000	1001	00001
2	0010	0010	0010	0010	0011	0101	0111	1000	00011
3	0011	0011	0011	0011	0010	0110	0110	0111	00111
4	0100	0100	0100	0100	0110	0111	0101	0110	01111
5	0101	0101	1000	1011	0111	1000	0100	0101	11111
6	0110	0110	1001	1100	0101	1001	0011	0100	11110
7	0111	1000	1010	1101	0100	1010	0010	0011	11100
8	1000	1001	1011	1110	1100	1011	0001	0010	11000
9	1001	1010	1100	1111	1101	1100	0000	0001	10000

А.2 Мультиплексоры

А.2.1 Реализовать на базе двух ИМС SN74LS151 (КР1533КП7) и логических схем той же серии мультиплексор «1 из 16» ([9], с. 10–11).

А.2.2 Реализовать на базе четырех мультиплексоров ИМС SN74LS151 (КР1533КП7) и логических схем той же серии мультиплексор «1 из 32» без стробирования по параллельной схеме наращивания каналов ([9], с. 11).

А.2.3 Реализовать только на базе ИМС SN74LS151 (КР1533КП7) мультиплексор «1 из 24» со стробированием ([9], с. 11).

А.2.4 Построить *JK*-триггер с помощью мультиплексора ИМС SN74ALS153 (КР1533КП2) и *D*-триггера ИМС SN74ALS74 (КР1533ТМ2) ([9], с. 11–12).

А.2.5 В соответствии с номером варианта (таблица А.3) построить только на одной ИМС SN74ALS153 (КР1533КП2) комбинационный узел, реализующий заданную логическую функцию трех переменных ([2], с. 113–116; [6], с. 98–102; [9], с. 16–17; [11], с. 129–134).

Таблица А.3 – Варианты логических функций (к заданию А.2.5)

<i>ABC</i>	<i>F</i> ₁	<i>F</i> ₂	<i>F</i> ₃	<i>F</i> ₄	<i>F</i> ₅	<i>F</i> ₆	<i>F</i> ₇	<i>F</i> ₈	<i>F</i> ₉	<i>F</i> ₁₀	<i>F</i> ₁₁	<i>F</i> ₁₂	<i>F</i> ₁₃	...	<i>F</i> ₂₅₄
000	0	0	0	0	0	0	0	0	0	0	0	0	0	...	1
001	0	0	0	0	0	0	0	0	0	0	0	0	0	...	1
010	0	0	0	0	0	0	0	0	0	0	0	0	0	...	1
011	0	0	0	0	0	0	0	0	0	0	0	0	0	...	1
100	0	0	0	0	0	0	0	1	1	1	1	1	1	...	1
101	0	0	0	1	1	1	1	0	0	0	0	1	1	...	1
110	0	1	1	0	0	1	1	0	0	1	1	0	0	...	1
111	1	0	1	0	1	0	1	0	1	0	1	0	1	...	0

А.2.6 На базе ИМС SN74ALS153 (КР1533КП2) построить устройство циклического сдвига 4-разрядного двоичного кода ([11], с. 135–137).

А.2.7 Построить устройство выборки четырех разрядов $x_1x_2x_3x_4, \dots, x_{29}x_{30}x_{31}x_{32}$ из 32-разрядного двоичного кода $x_1 - x_{32}$ на селекторах-мультиплексорах ИМС SN74LS151 (КР1533КП7) ([11], с. 135–137).

А.2.8 Реализовать мажоритарный элемент для трех переменных на мультиплексоре ИМС SN74ALS153 (КР1533КП2) ([11], с. 128–134).

А.3 Счетчики, пересчетные устройства

А.3.1 Привести схему, реализующую на счетчике ИМС SN74ALS193 (КР1533ИЕ7) модуль счета в соответствии с одним из вариантов 3–224 (модули счета соответственно 3–224) в режиме: а) суммирования; б) вычитания; в) с квазисинхронной загрузкой ([2], с. 247; [9], с. 22–23). Нарисовать несколько тактов временной диаграммы до и после момента параллельного занесения информации.

А.3.2 На базе двух *JK*-триггеров одной ИМС SN74ALS112 (КР1533ТВ9) и логических схем той же серии построить один из генераторов чисел: а) 3 – 2 – 8 – 12; б) 1 – 0 – 7 – 11; в) 2 – 5 – 14 – 1; г) 4 – 9 – 3 – 6;

д) 1 - 13 - 6 - 4; е) 10 - 7 - 2 - 5; ж) 2 - 13 - 8 - 7; з) 6 - 1 - 2 - 0; и) 7 - 11 - 0 - 5; к) 14 - 0 - 6 - 9; л) 8 - 2 - 5 - 11; м) 9 - 0 - 8 - 10 ([9], с. 23-24).

А.3.3 На базе четырех *JK*-триггеров двух ИМС SN74ALS109 (КР1533ТВ15) и логических схем той же серии построить один из генераторов чисел задачи А.3.2 ([9], с. 24–26).

А.3.4 На основе счетчика ИМС SN74ALS193 (КР1533ИЕ7) построить пересчетное устройство, принимающее в цикле состояния 0 - 1 - 2 - 3 - 5 - 6 - 7 ([2], с. 237–250). Нарисовать временную диаграмму.

А.3.5 Построить схему пересчетного устройства с модулем 5 на ИМС серии SN74ALS (КР1533) в соответствии с вариантом: а) 0 - 1 - 2 - 3 - 4; б) 3 - 4 - 5 - 6 - 7; в) 2 - 3 - 4 - 5 - 6; г) 0 - 1 - 5 - 6 - 7. Привести временную диаграмму ([2], с. 237–250).

А.3.6 Построить схему управляемого суммирующего счетчика с параллельным переносом, в котором при управляющем сигнале $a = 0$ модуль счета $k = 8$, а при $a = 1$ модуль счета $k = 5$ ([12], с. 47, 65). Использовать *JK*-триггеры и логические элементы серии SN74ALS (КР1533).

А.3.7 Построить на ИМС SN74ALS74 (КР1533ТМ2) и логических элементах той же серии схему 4-разрядного асинхронного счетчика с собственной остановкой на заданном входном 4-разрядном двоичном коде. Для повторного цикла счета его необходимо установить в «0». Привести временную диаграмму ([2], с. 237–250; [9], с. 30–31).

А.3.8 На *JK*-триггерах и логических элементах серии SN74ALS (КР1533) построить синхронный счетчик с коэффициентом счета $K = 32 - N$, где N - входной 5-разрядный двоичный код ([10], с. 92–99). Привести временную диаграмму.

А.3.9 На *D*-триггерах и элементах «исключающее ИЛИ» ИМС серии SN74ALS (КР1533) построить n -разрядное пересчетное устройство на сдвиговом регистре с количеством состояний $K = 2^n - 1$ для одного из вариантов 3 - 12, у которых соответственно $n = 3 - 12$. Добавить логические элементы для выхода из нуля. Привести несколько тактов временной диаграммы для моментов изменения сигналов на выходах элементов «исключающее ИЛИ» ([9], с. 26–29; [10], с. 134–139).

А.3.10 На *D*-триггерах и логических элементах ИМС серии SN74ALS (КР1533) построить n -разрядное пересчетное устройство на сдвиговом регистре (максимальное количество состояний $K_{max} = 2^n - 1$) с модулем счета K (метод скачка) в соответствии с номером варианта $K = 4 - 62$ ([9], с. 26–29; [10], с. 134-139). Привести граф состояний.

А.3.11 На ИМС серии SN74ALS (КР1533) построить 3-разрядный счетчик, который при управляющем входе $M = 1$ работает как двоичный суммирующий счетчик, а при $M = 0$ - как счетчик в коде Грея ([9], с. 29–30).

А.3.12 На ИМС серии SN74ALS (КР1533) построить распределитель импульсов на базе счетчика Джонсона и преобразователя кода Джонсона в код «1 из N ». Привести временную диаграмму ([6], с. 255–258; [10], с. 131–134).

А.4 Регистры

А.4.1 На ИМС SN74ALS175 (КР1533ТМ8) и логических схемах той же серии реализовать выполнение поразрядных логических операций дизъюнкции и конъюнкции при передаче двоичного кода с регистра на регистр ([6], с. 237).

А.4.2 На триггерах ИМС SN74ALS74 (КР1533ТМ2) и логических схемах той же серии построить на основе сдвигового регистра генератор чисел 0 - 1 - 2 - 5 - 3 - 7 - 6 - 4 ([9], с. 31-32; [10], с. 120-127; [13], с. 228-233).

А.4.3 На регистре ИМС SN74LS295 (КР1533ИР16) и логических схемах той же серии построить генератор чисел 2 - 3 - 3 - 5 - 3 ([9], с. 32-33).

А.4.4 На ИМС серии SN74ALS (КР1533) построить 3-разрядный регистр сдвига, имеющий одну из следующих последовательностей состояний: а) 0 - 1 - 2 - 5 - 3 - 6 - 4 - 0; б) 0 - 1 - 2 - 4 - 0; в) 0 - 1 - 2 - 5 - 3 - 7 - 6 - 4 - 0; г) 0 - 1 - 3 - 6 - 5 - 2 - 4 - 0; д) 0 - 1 - 3 - 6 - 5 - 2 - 4 - 0 ([9], с. 31-32; [10], с. 120-127; [13], с. 228-233).

А.4.5 На элементах серии SN74ALS (КР1533) построить 4-разрядный регистр сдвига ([9], с. 31-32; [10], с. 120-127; [13], с. 228-233), имеющий одну из следующих последовательностей состояний:

- а) 0 - 1 - 3 - 7 - 14 - 13 - 10 - 4 - 8 - 0;
- б) 0 - 1 - 3 - 6 - 12 - 9 - 2 - 4 - 8 - 0;
- в) 0 - 1 - 2 - 5 - 11 - 7 - 14 - 12 - 8 - 0;
- г) 0 - 1 - 2 - 4 - 9 - 3 - 6 - 12 - 8 - 0;
- д) 0 - 1 - 3 - 7 - 15 - 14 - 13 - 10 - 4 - 8 - 0;
- е) 0 - 1 - 3 - 7 - 14 - 13 - 11 - 6 - 12 - 8 - 0;
- ж) 0 - 1 - 2 - 5 - 11 - 7 - 15 - 14 - 12 - 8 - 0;
- з) 0 - 1 - 2 - 4 - 9 - 3 - 7 - 14 - 12 - 8 - 0;
- и) 0 - 1 - 3 - 7 - 15 - 14 - 13 - 11 - 6 - 12 - 8 - 0;
- к) 0 - 1 - 2 - 5 - 11 - 7 - 14 - 13 - 10 - 4 - 8 - 0;
- л) 0 - 1 - 2 - 4 - 9 - 3 - 7 - 15 - 14 - 12 - 8 - 0;
- м) 0 - 1 - 3 - 7 - 14 - 13 - 10 - 5 - 11 - 6 - 12 - 8 - 0;
- н) 0 - 1 - 3 - 6 - 13 - 10 - 5 - 11 - 7 - 14 - 12 - 8 - 0;
- о) 0 - 1 - 2 - 5 - 11 - 7 - 15 - 14 - 13 - 10 - 4 - 8 - 0;
- п) 0 - 1 - 3 - 6 - 13 - 10 - 5 - 11 - 7 - 15 - 14 - 12 - 8 - 0;
- р) 0 - 1 - 3 - 6 - 13 - 10 - 4 - 9 - 2 - 5 - 11 - 7 - 14 - 12 - 8 - 0;
- с) 0 - 1 - 3 - 6 - 13 - 10 - 5 - 11 - 7 - 15 - 14 - 12 - 9 - 2 - 4 - 8 - 0;
- т) 0 - 1 - 3 - 6 - 13 - 10 - 4 - 9 - 2 - 5 - 11 - 7 - 15 - 14 - 12 - 8 - 0.

А.4.6 На ИМС серии SN74ALS (КР1533) построить схему 4-разрядного счетчика Джонсона, исключив состояние 1111 из счетной последовательности. Начертить временную диаграмму, характеризующую его работу в течение восьми тактовых импульсов. Привести временную диаграмму ([6], с. 255-258; [10], с. 131-134).

А.4.7 На ИМС SN74ALS109 (КР1533ТВ15) и логических элементах той же серии построить схему 2-разрядного реверсивного сдвигающего регистра. Привести временную диаграмму ([11], с. 179-182).

А.4.8 На ИМС SN74LS165 (КР1533ИР9) и логических элементах той же серии построить преобразователь 7-разрядного параллельного кода в последовательный ([6], с. 234–237).

А.5 Сумматоры и арифметическо-логические устройства (АЛУ)

А.5.1 На ИМС SN74LS283 (К555ИМ6) построить 8-разрядный сумматор дополнительного кода (восьмой разряд знаковый). Определить его быстродействие ([2], с. 131–135; [10], с. 65–66).

А.5.2 На ИМС SN74ALS181 (КР1533ИП3), SN74ALS182 (КР1533ИП4) построить 12-разрядный сумматор с параллельным переносом. Обеспечить правильность формирования сигналов P и G на выход ИМС SN74ALS182 (КР1533ИП4) для дальнейшего наращивания 12-разрядных секций. Определить быстродействие ([2], с. 135–141).

А.5.3 Используя ИМС АЛУ SN74ALS181 (КР1533ИП3), построить преобразователь 8-разрядного прямого кода целого числа со знаком в дополнительный ([2], с. 141–145; [9], с. 20–22).

А.5.4 Используя ИМС АЛУ SN74ALS181 (КР1533ИП3), построить преобразователи: а) прямого хода в обратный; б) обратного кода в прямой; в) дополнительного кода в обратный; г) обратного кода в дополнительный ([2], с. 141–145; [9], с. 20–22).

А.5.5 Используя ИМС АЛУ SN74ALS181 (КР1533ИП3) построить сумматор по модулю q : а) $q = 3$; б) $q = 5$; в) $q = 7$; г) $q = 11$; д) $q = 13$ ([2], с. 203–204).

А.5.6 На ИМС серии SN74ALS (КР1533) построить 4-разрядный последовательный сумматор ([6], с. 119–120; [11], с. 141).

А.5.7 На ИМС SN74LS283 (К555ИМ6) и логических элементах той же серии построить 12-разрядный групповой сумматор с параллельным переносом ([6], с. 126–128).

А.5.8 На ИМС SN74LS283 (К555ИМ6) и логических элементах той же серии построить 8-разрядный сумматор с условным переносом ([6], с. 128).

А.5.9 На трех ИМС SN74LS283 (К555ИМ6) построить пороговую схему, $n = 9$, $k = 6$ ([2], с. 145–146; [9], с. 18–19).

А.5.10 На четырех ИМС SN74LS283 (К555ИМ6) построить мажоритарный элемент для $n = 13$ ([2], с. 146; [9], с. 19–20).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Петровский, И. И. Логические ИС КР1533, КР1554 : справочник. В 2 ч. / И. И. Петровский [и др.] ; пер. с англ. – М. : ТОО «БИНОМ», 1993. – Ч. 1 – 254 с. ; Ч. 2 – 498 с.
- [2] Пухальский, Г. И. Проектирование дискретных устройств на интегральных микросхемах : справочник / Г. И. Пухальский, Т. Я. Новосельцева. – М. : Радио и связь, 1990. – 304 с.
- [3] Шило, В. Л. Популярныe цифровые микросхемы : справочник / В. Л. Шило. – 2-е изд., испр. – Челябинск : Металлургия, Челябинское отд., 1989. – 352 с.
- [4] Никитин, В. А. Схемотехника интегральных схем ТТЛ, ТТЛШ и КМОП : учеб. пособие / В. А. Никитин. – М. : НИЯУ МИФИ, 2010. – 64 с.
- [5] Хэррис, Д. М. Цифровая схемотехника и архитектура компьютера / Д. М. Хэррис, С. Л. Хэррис ; пер. с англ. – М. : ДМК Пресс, 2017. – 771 с.
- [6] Угрюмов, Е. П. Цифровая схемотехника / Е. П. Угрюмов. – 3-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2010. – 816 с.
- [7] Савельев, А. Я. Основы информатики : учеб. для вузов / А. Я. Савельев. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2001. – 328 с.
- [8] Проектирование цифровых вычислительных машин : учеб. пособие для студентов вузов / С. А. Майоров [и др.] ; под ред. С. А. Майорова. – М. : Высш. шк., 1972. – 344 с.
- [9] Проектирование цифровых устройств на интегральных микросхемах : метод. пособие по курсу «Основы проектирования ЭВС» для студ. спец. 1-40 02 02 «Электронные вычислительные средства» дневной формы обуч. / Г. В. Таранов [и др.]. – Минск : БГУИР, 2006. – 35 с.
- [10] Голдсуорт, Б. Проектирование цифровых логических устройств / Б. Голдсуорт ; пер. с англ. М. В. Сергиевского ; под ред. Ю. И. Топчеева. – М. : Машиностроение, 1985. – 288 с.
- [11] Зубчук, В. И. Справочник по цифровой схемотехнике / В. И. Зубчук, В. П. Сигорский, А. Н. Шкуро. – Киев : Тэхника, 1990. – 448 с.
- [12] Гольденберг, Л. М. Цифровые устройства и микропроцессорные системы. Задачи и упражнения : учеб. пособие для вузов / Л. М. Гольденберг, В. А. Малев, Г. Б. Малько. – М. : Радио и связь, 1992. – 256 с.
- [13] Алексенко, А. Г. Микросхемотехника : учеб. пособие для вузов / А. Г. Алексенко, И. И. Шагурин. – 2-е изд., перераб. и доп. – М. : Радио и связь, 1990. – 496 с.
- [14] ISE In-Depth Tutorial (UG695, v14.1) [Электронный ресурс]. – 2012. – Режим доступа : https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf.
- [15] СТП 01–2017. Дипломные проекты (работы). – Минск : БГУИР, 2017.

Учебное издание

**Качинский Михаил Вячеславович
Герасимович Вадим Юрьевич
Станкевич Андрей Владимирович**

**ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ
НА ИНТЕГРАЛЬНЫХ МИКРОСХЕМАХ.
КУРСОВОЕ ПРОЕКТИРОВАНИЕ**

ПОСОБИЕ

Редактор *Е. С. Юрец*
Корректор
Компьютерная правка, оригинал-макет

Подписано в печать . Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. . Уч-изд. л. . Тираж 50 экз. Заказ 335.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6