МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра космического приборостроения и систем связи

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_9_» _____ 2024 г.

ПРОЕКТИРОВАНИЕ СИСТЕМ НА БАЗЕ ПЛИС

Методические указания по выполнению лабораторной работы
для студентов направления подготовки
11.04.02 Инфокоммуникационные технологии и системы связи

Курск 2024

УДК 621.396

Составители: И.Г. Бабанин, Е.Ю. Бабанина

Рецензент

Доктор технических наук, старший научный сотрудник,
заведующий кафедрой *В.Г. Андронов*

Проектирование систем на базе ПЛИС : методические указания по выполнению лабораторной работы / Юго-Зап. гос. ун-т; сост.: И. Г. Бабанин, Е.Ю. Бабанина. – Курск, 2024.– 41 с.

Содержат теоретические сведения о проектировании систем на базе ПЛИС компании Xilinx, а также рекомендации по выполнению лабораторной работы..

Методические указания соответствуют требованиям ФГОС ВО по направлению подготовки 11.04.02 Инфокоммуникационные технологии и системы связи, рабочим учебным планам по направлению подготовки 11.04.02.

Предназначены для студентов по направлениям подготовки 11.04.02.

Текст печатается в авторской редакции

Подписано печать 9.04.24 Формат 60x841/16.
Усл. печ. л. 2,38. Уч.-изд. л. 2,16. Тираж 100 экз. Заказ 602. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94

1 Цель работы

Изучить основные принципы работы с программным пакетом Xilinx ISE на основе проектирования простейших цифровых схем.

2 Краткие теоретические сведения

2.1 Программный пакет Xilinx ISE

2.2 Создание нового проекта

Для создания нового проекта необходимо запустить программу ISE Project Navigator с использованием ярлыка на рабочем столе, либо главного системного меню (Рисунок 1).

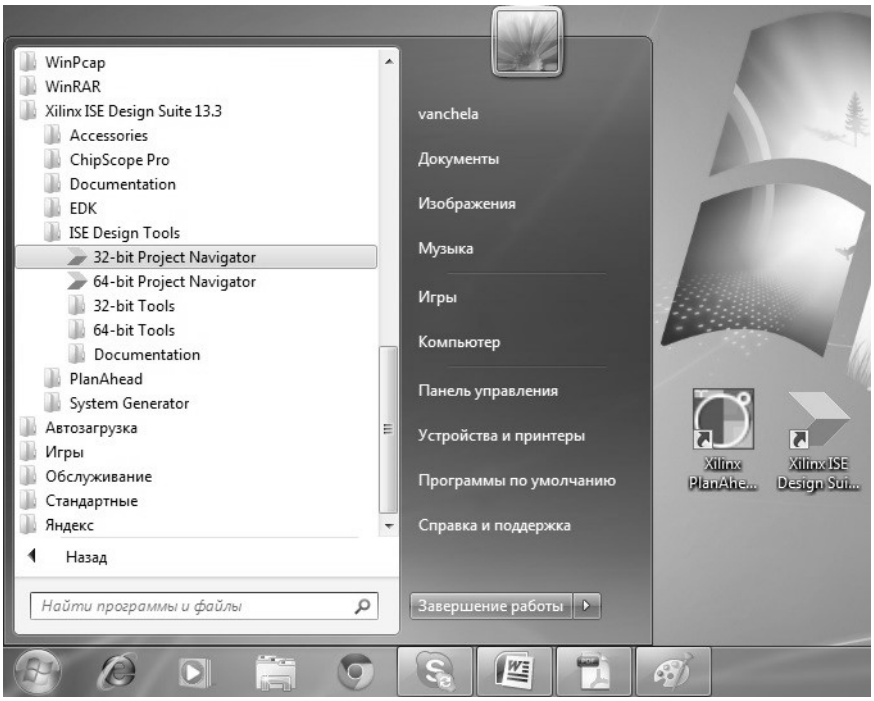


Рисунок 1 – Запуск навигатора проектов Xilinx ISE

В результате будет открыто основное окно (Навигатор проектов ISE), содержащее данные предыдущего проекта. Как показано на рисунке 2, при стандартных настройках интерфейс пользователя представляет собой комбинацию типичных для интегрированных сред проектирования окон:

- окно исходных модулей, т.е. непосредственно навигатор проекта;
- окно документов;
-окно процессов (проектных процедур), которые могут быть выполнены для модуля, выбранного в окне навигатора проекта;
- окно консоли сообщений о ходе выполнения проектных процедур и их результатах.

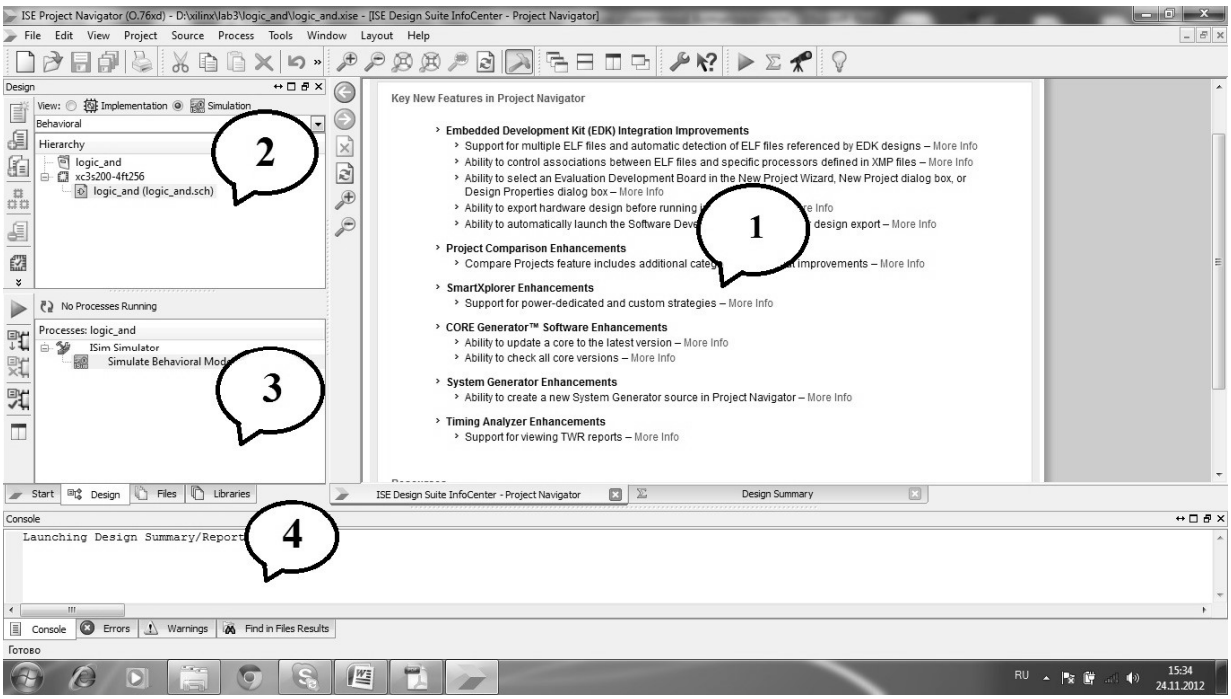


Рисунок 2 – Главное окно интерфейса пользователя ISE Project Navigator, где 1- окно документов; 2- навигатор проекта; 3- окно процессов; 4- окно сообщений

В свою очередь, окна могут иметь вкладки, раскрывающиеся при нажатии соответствующих кнопок, расположенных внизу окна. Сверху расположены главное меню и панель инструментов проекта, обозначенных пиктограммами и дублирующими вызов некоторых основных функций вкладок главного меню. Каждый пункт главного меню открывает всплывающий список соответствующих функций, перечень приведен в таблице 1 [1].

Таблица 1 – Перечень основных функций главного меню ISE Project Navigator

Пункт меню	**Соответствующие инструменты подменю**	
File	**Работа с файлами, функции печати и завершения работы**	
	Создать новый проект	New Project…
	Открыть уже существующий проект	Open Project…
	Открыть один из примеров проекта	Open Example…
	Просмотреть содержимое проекта	Project Browser…
	Скопировать проект	Copy Project…
	Закрыть проект	Close Project
	Создать файл в проекте	New
	Открыть файл	Open…
	Закрыть файл	Close
	Сохранить файл	Save
	Сохранить файл как…	Sava As…
	Сохранить все открытые файлы в	Save All

	проекте	
		Print Preview…
	Распечатать проект	Print…
	Открыть последний файл	Recent Files
	Открыть последний проект	Recent Project
	Выход	Exit
Edit	**Работа со встроенным редактором HDL- кода и функции настройки конфигурации навигатора проекта**	
	Отменить действие	Undo
	Вернуть отмененное действие	Redo
	Вырезать	Cut
	Копировать	Copy
	Вставить	Paste
	Удалить	Delete
	Найти…	Find…
		Find Next
		Find Previous
		Find&Replace
	Найти в данном файле	Find in Files
	Ввести комментарий	Comment
	Преобразовать	Convert
	Вставить файл	Insert file
	Перейти к…	Go to
	Переименовать	Rename
		Show Next Result
		Show Previous Result
	Использовать шаблон кода языка	Language template…
	Выбрать всё	Select all
		Message Filters…
	Настройка конфигурации навигатора проекта	Preference…
View	**Функции настройки режимов просмотра**	

Выбираем пункт меню File -> New Project. Откроется диалоговое окно New Project Wizard (Рисунок 3).

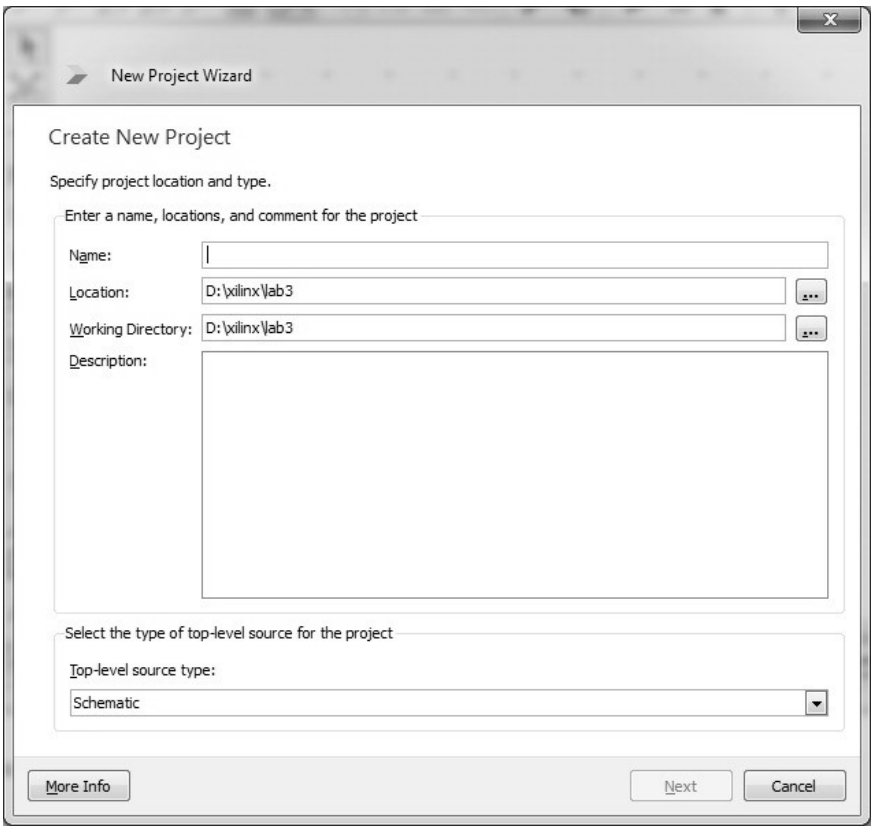


Рисунок 3 – Диалоговое окно New Project Wizard

В строке Name необходимо указать название создаваемого проекта (lab_pr_1), в строках Location, Working Directory путь к рабочим файлам (D:/lab_pr_xilinx/gr_IT-11b/lab_pr_1).

САПР Xilinx ISE требует, чтобы в пути к рабочим файлам проекта не было пробелов и символов кириллицы.

В этом же окне задается формат представления «главного модуля проекта» (Top-level source type):

- HDL (Hardware Description Language) – текстовый файл на языке описания аппаратуры;

- Schematic – графическое изображение принципиальной электрической схемы, составленной из стандартных библиотечных модулей, и модулей, добавляемых разработчиком в виде других графических схем или файлов на HDL [1];

- EDIF, NGC/NGO – устройство представляется в виде готовых списков связей, разработанных ранее в САПР Xilinx ISE или с помощью иных программных инструментов. Маршруты, основанные на EDIF и NGC/NGO, представляют интерес в том случае, если в ПЛИС выполняется устройство, приобретенное в виде IP- ядра. В такой проект невозможно внести несанкционированные изменения, или восстановить его схему, имея NGC- представление [1].

Для выполнения данного лабораторного практикума установить режим Schematic.

Для перехода к следующему диалоговому окну производим клик по кнопке Next.

В открывшемся окне выполняются следующие действия:

- выбор используемой микросхемы или отладочной платы в проекте (для облегчения поиска заданной ПЛИС в данном окне предусмотрены следующие типы классификации: категория продукта (Product Category), серия (Family), модель (Device), тип корпуса (Package), класс скорости (Speed));

- выбор инструмента синтеза (Synthesis Tool), программы для моделирования (Simulator), языка описания аппаратуры (Preferred Language), спецификации параметров в проекте (Property Specification in Project File), стандарта источника анализа VHDL (VHDL Source Analysis Standard);

- вкл/выкл фильтра сообщений (Enable Massage Filtering).

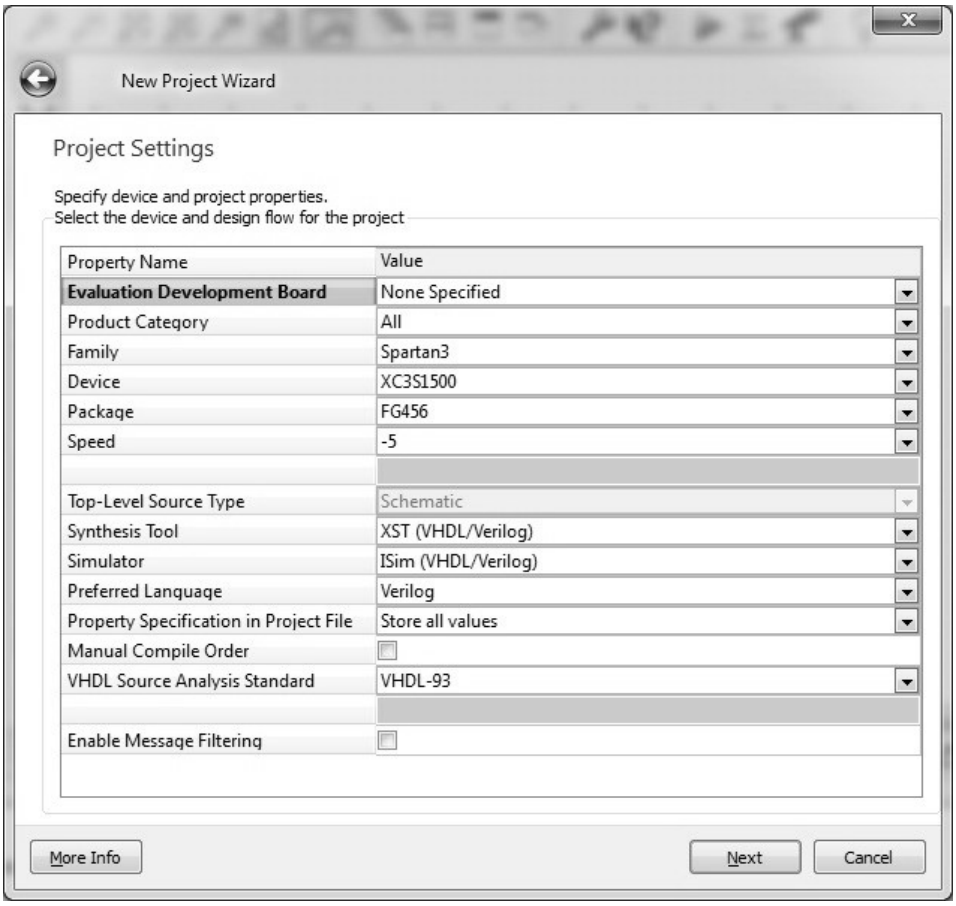


Рисунок 4 – Диалоговое окно выбора ПЛИС и настройки параметров САПР

Вид правильно заполненных полей для данного практикума представлен на рисунке 4. Изменение параметров окна можно выполнить на любой стадии проекта.

При клике на кнопку Next открывается окно, представленное на рисунке 5, для проверки установленных значений в создаваемом проекте.

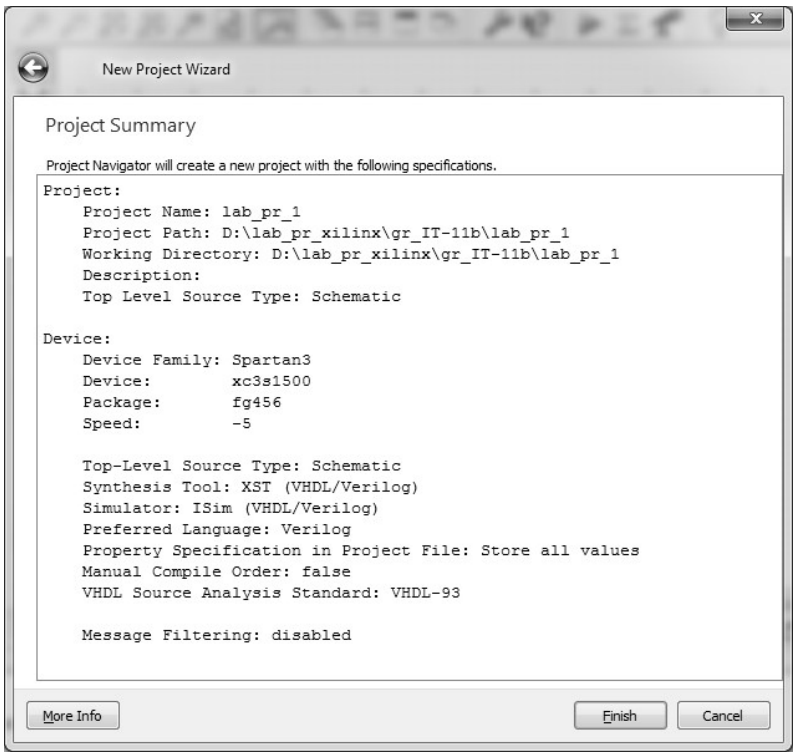


Рисунок 5 – Диалоговое окно проверки установленных значений

Кнопка Finish производит завершение начальных установок по созданию цифрового устройства на базе ПЛИС.

2.3 Схемотехническое проектирование цифрового устройства

Для создания схемы необходимо в окне навигатора проектов найти имя проекта и с помощью контекстного меню (нажатием правой кнопки мыши) выбрать пункт New Source (Рисунок 6), или же вызвать функцию Project -> New Source (Рисунок 7).

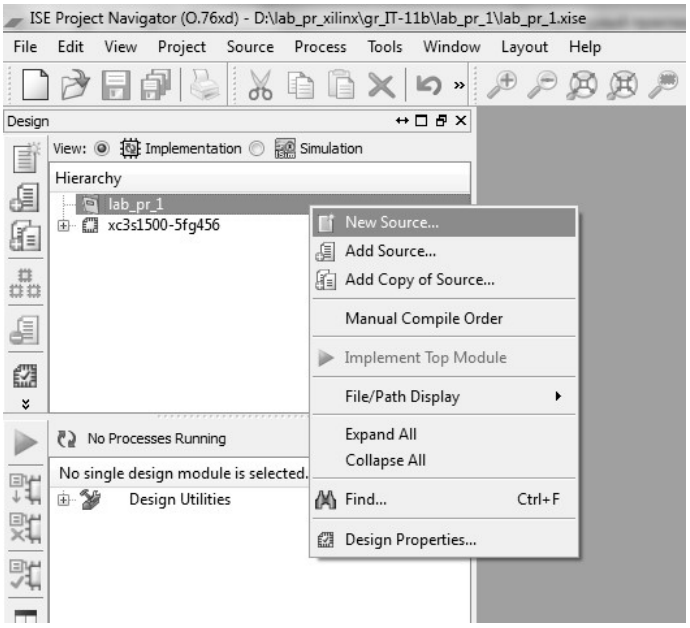


Рисунок 6 – Контекстное меню для выбора пункта New Source

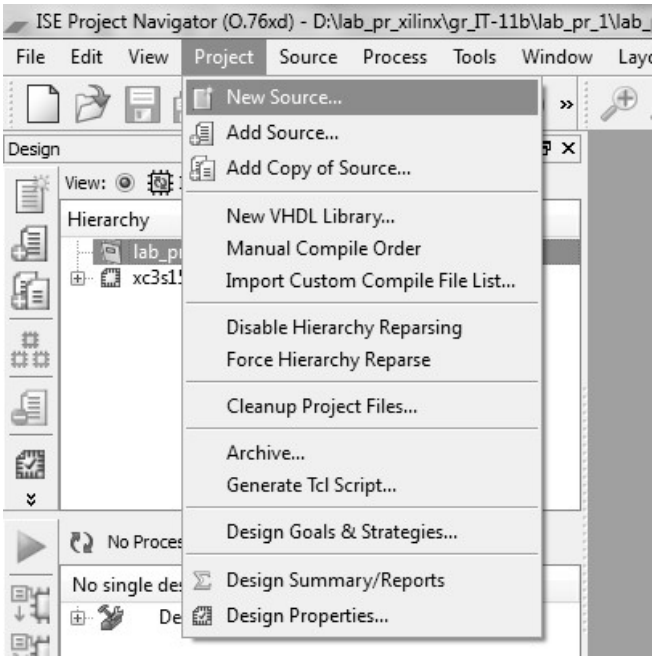


Рисунок 7 – Меню Project

В появившемся окне New Source Wizard выбрать форму представления верхнего уровня модуля (Schematic), дать ему название (logic_1) и добавить в проект (Рисунок 8).

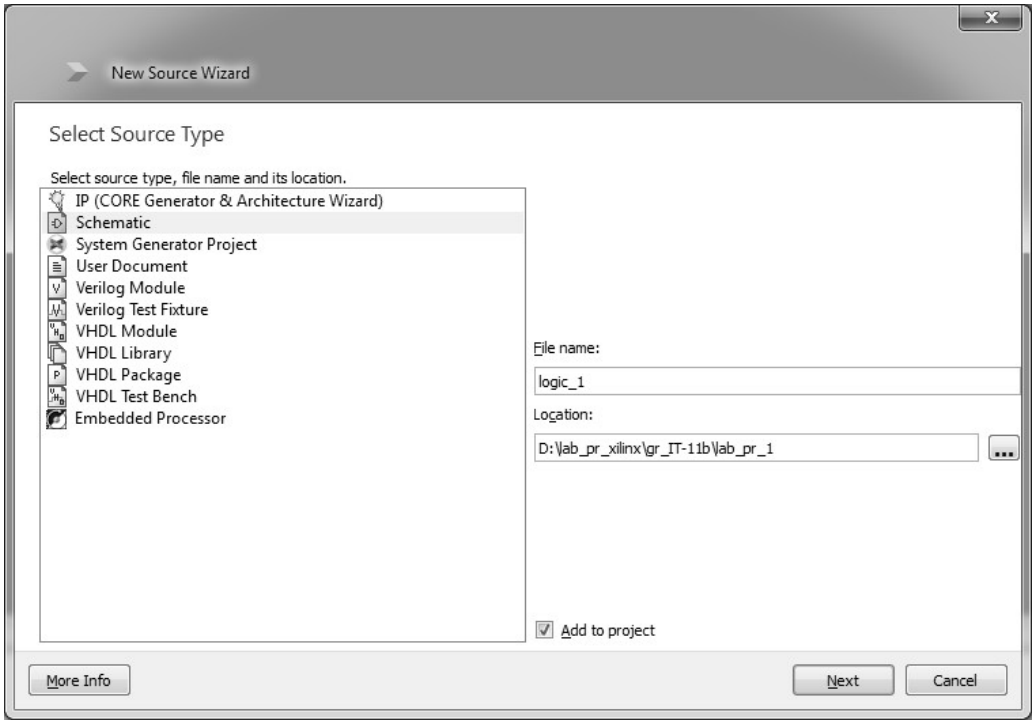


Рисунок 8 – Диалоговое окно создания нового модуля проекта

Для перехода к следующему окну производим клик по кнопке Next. В открывшемся окне содержится отчёт о создании нового модуля (Рисунок 9).

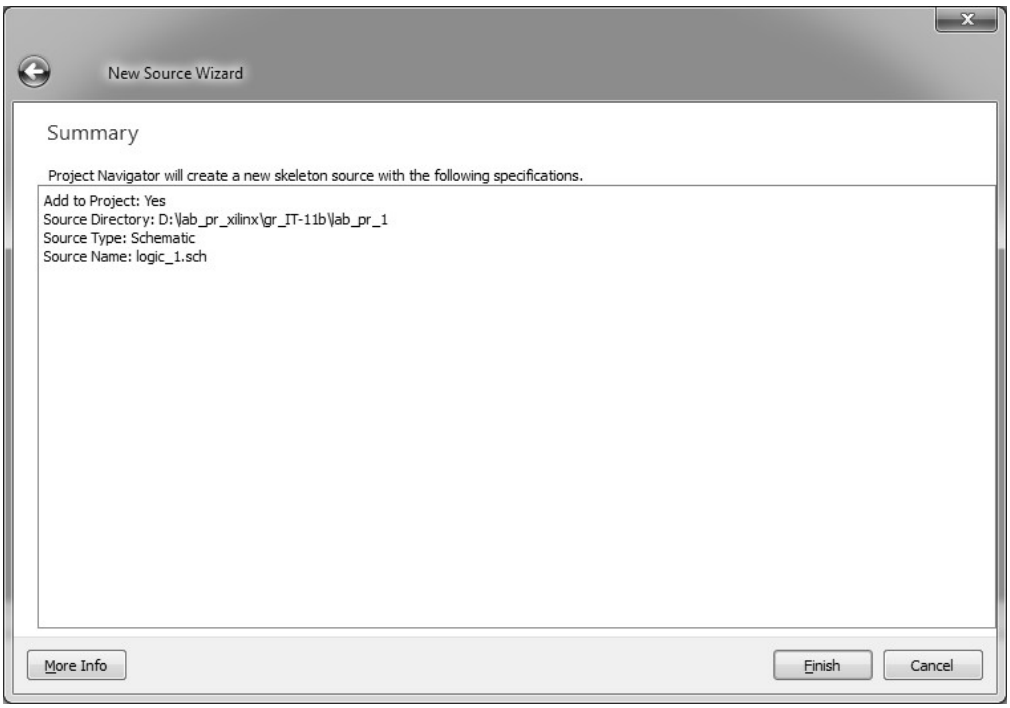


Рисунок 9 – Отчёт о создании нового модуля

По нажатию кнопки Finish навигатор проекта приобретает вид, представленный на рисунке 10.

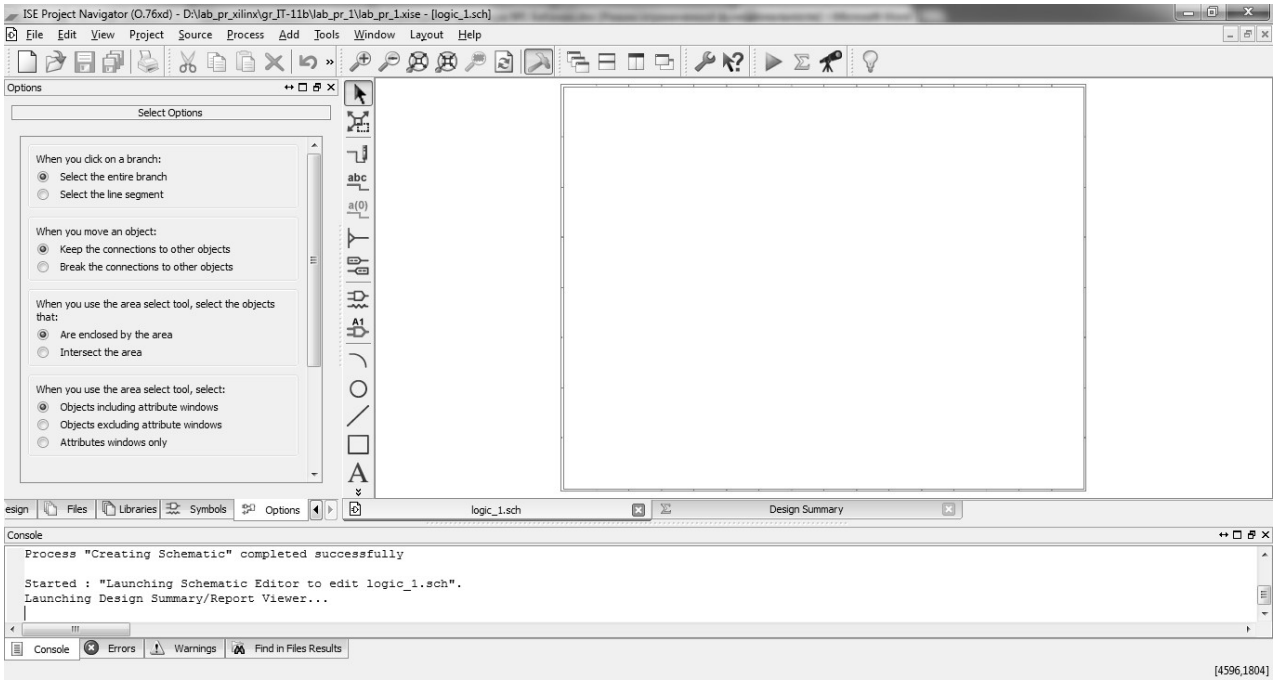


Рисунок 10 – Графический редактор в САПР Xilinx ISE

Поскольку для верхнего уровня проекта выбран тип модуля в виде принципиальной схемы (расширение файла модуля ***.sch**), то соответствующее ему окно документа представляет собой графический редактор принципиальных схем ECS (Engineering Capture Schematic). Слева открыта вкладка Option, предназначенная для настроек режимов выбора

объектов графического редактора с помощью курсора. Назначение переключателей этой вкладки разъясняет таблица 2 [1].

Таблица 2 – Параметры настройки выбора объектов в редакторе ECS

When you click on a branch:	**При выборе проводника:**
Select the entire branch	выделять весь проводник;
Select the line segment	выделять сегмент линии.
When you move an object:	**При перемещении объекта:**
Keep the connection to other objects	сохранять соединения с другими объектами;
Break the connections to other objects	разрывать соединения с другими объектами.
When you use the area select tool, select the objects that:	**При выборе области, выделять объекты:**
Are enclosed by the area	полностью расположенные в области;
Intersect the area	пересекающие область.
When you use the area select tool, select:	**При выборе области выделять:**
Objects including attribute windows	объекты, включая окна атрибутов;
Objects excluding attribute windows	объекты, исключая атрибутов;
Attributes windows only	только окна атрибутов.

В лабораторном практикуме рекомендуем установить переключатели согласно рисунку 11.

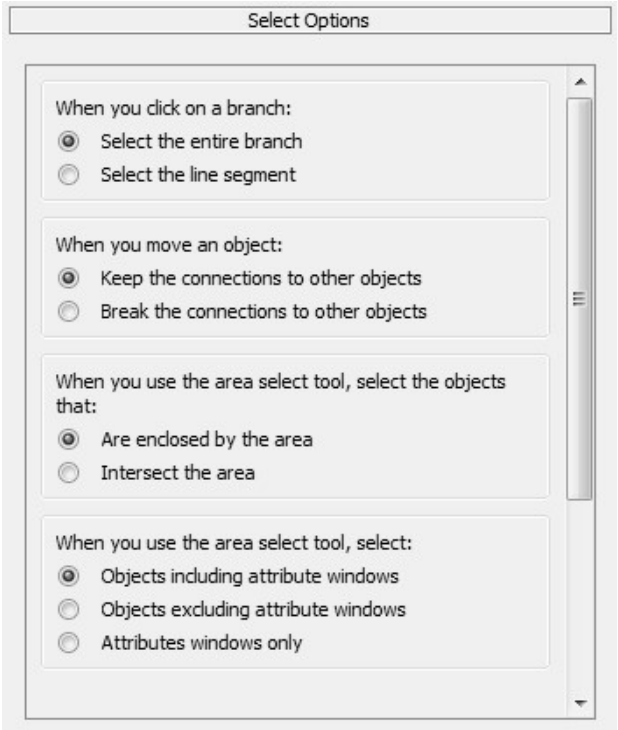


Рисунок 11 – Рекомендуемые параметры настройки для выполнения лабораторного практикума

Панель инструментов графического редактора ECS содержит стандартные для Windows- приложений кнопки-пиктограммы работы с файлами и кнопки специальных операций для редактирования принципиальных схем. Назначение этих кнопок поясняется таблицей 3 [1].

Таблица 3 – Основные кнопки панели инструментов редактора схем

Пиктограмма	**Назначение элементов управления**
	Порядок размещения дополнительных окон ECS
	Отображение/скрытие окна сообщений
	Вызов библиотеки шаблонов
	Изменение масштаба отображения схемы, переход к общему виду всего листа, выбор фрагмента для отображения
	Управление режимом «Рисование схемы» и «Курсор»
	Добавление проводника (шины)
	Создание, изменение количества проводников и имени шины
	Переименование выбранной шины
	Добавление отвода от шины
	Добавление маркеров ввода/вывода
	Выбор и размещение цифрового компонента
	Добавление надписи
	Элементы оформления схемы
	Поворот и зеркальное отображение схемы
	Проверка правильности изображения схемы
	Переходы к уровням иерархии проекта
	Вызов справки

Для добавления элемента схемы, в общем случае, можно использовать контекстное меню Add -> Symbol или кнопку «  » (выбор элемента) [2]. После чего из списка, представленного на рисунке 12, выбрать соответствующий элемент из заданной библиотеки и поместить на схему его графическое изображение.

Рисунок 12 – Выбор элемента из библиотеки logic

При добавлении проводника необходимо нажать кнопку «  » и настроить следующие параметры, представленные на рисунке 13:

1) При добавлении проводника в графический редактор (When you add a wire:):

- использовать автотрассировку между указанными точками (Use the Autorouter to add one or more line segments between the points you indicate);
- использовать ручной метод добавления сегментов линии между заданными точками (Use the Manual method to add single line segments between the points you indicate).

2) При соединении контакта элемента и шины (When a wire connects a symbol pin and a bus):

- автоматически добавлять отвод между шиной и проводником (Auromatically add a bus tap between the bus and the wire).

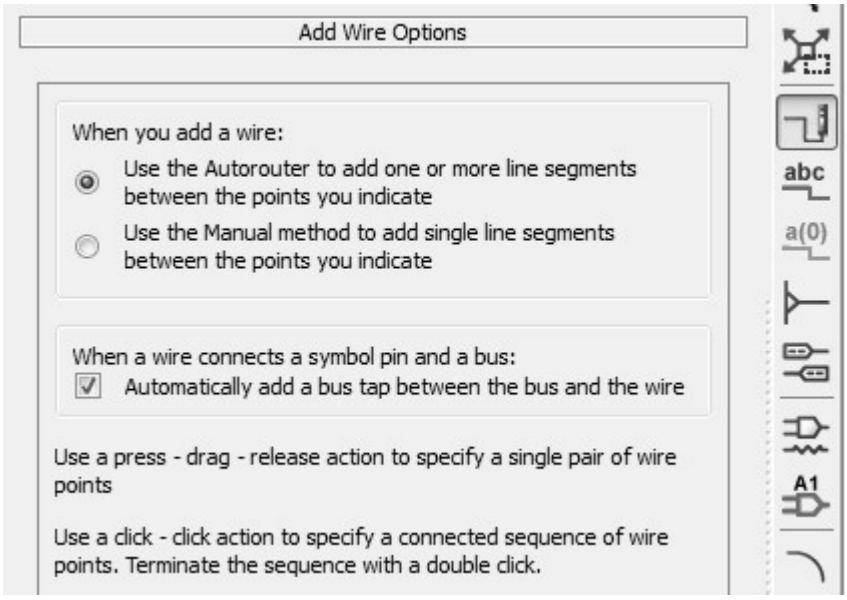


Рисунок 13 – Настройки параметров при добавлении проводника

После этого в поле графического редактора ECS нужно кликнуть левой клавишей мыши по ножке элемента или в свободном поле, тем самым установив начальную точку проводника, затем каждый аналогичный клик мыши будет устанавливать конец сегмента. Для окончания рисования проводника необходимо нажать клавишу Esc. В случае если конец проводника соединен с портом элемента окончание рисования завершается автоматически. Необходимо заметить, в случае установления начальной точки на элементе у которого входные/выходные порты представлены в виде шины, то при последующих кликах будет создаваться шина.

Для создания шины из нарисованного проводника необходимо перевести редактор в режим «Курсор» и по проводнику кликнуть правой клавишей мыши, тем самым вызвав контекстное меню. После чего в появившемся меню необходимо выбрать строку Rename Selected Net… В открывшемся диалоговом окне выбирается пункт «Переименовать ветви всей цепи (Rename the Branch's Net)», в противном случае будет переименована одна заданная ветвь. В ниже стоящей строке указывается название шины и в круглых скобках количество проводников. По желанию может быть установлена «галка» для отображения названия редактируемого объекта в графическом редакторе (Рисунок 14). По завершению проделанных изменений нажимается кнопка OK, в случае нажатия Cancel изменения не будут сохранены.

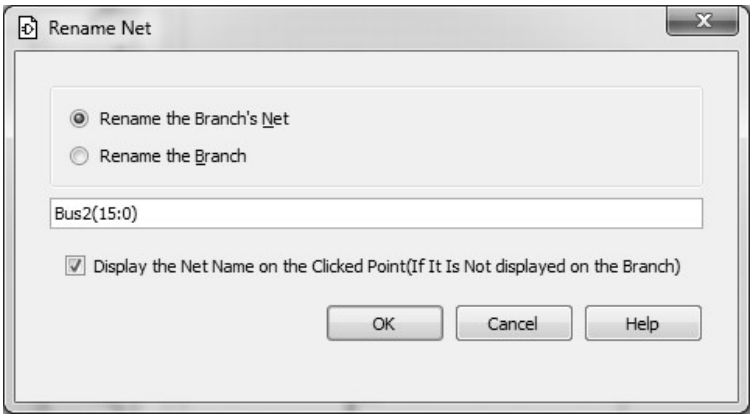


Рисунок 14 – Диалоговое окно настроек объекта

Для добавления отвода от заданной шины нужно нажать кнопку «  », выделить нужную шину левым кликом мыши. После этого в настройках параметров необходимо задать направление отвода и кликом по левой клавиши мыши в графическом редакторе создать отвод (Рисунок 15). Затем после добавления всех отводов нужно соединить их с соответствующими элементами и указать к каким проводникам в шине они подключены. Для этого двойным кликом мыши по отводу вызвать диалоговое окно, где необходимо выбрать строку в соответствии с рисунком 16. В строке «Name» ввести название шины и в круглых скобках номер подключаемого проводника. Нажать Apply, затем OK.

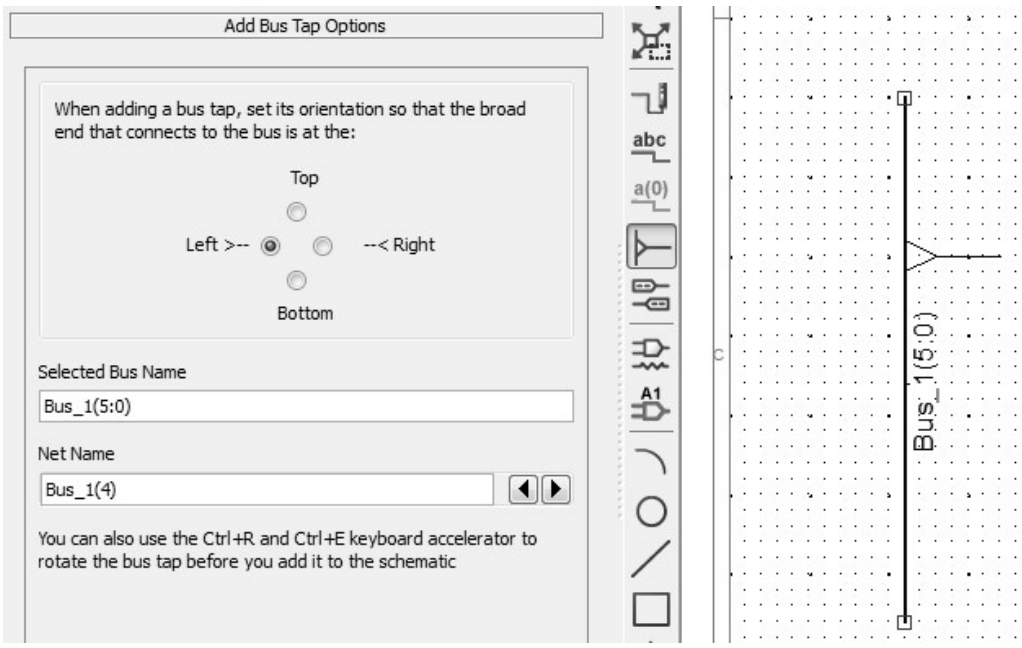


Рисунок 15 – Настройки параметров отвода от шины

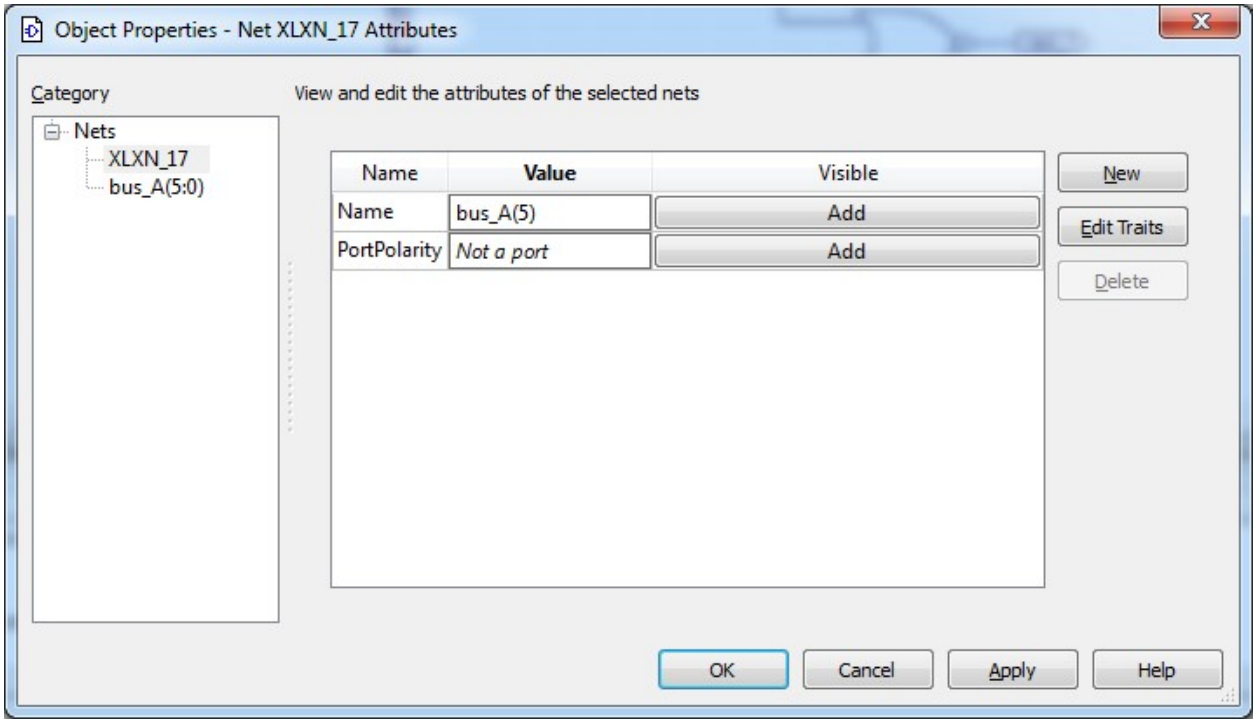


Рисунок 16 – Подключение отвода к соответствующему проводнику в шине

По окончанию создания цифрового устройства в графической среде все входные и выходные порты «навешивают» специальными маркерами, которые впоследствии должны быть подключены в выводам микросхемы ПЛИС. Для произведения данной процедуры производится нажатие кнопки «  » и выбирается из окна настроек соответствующий параметр маркера (Рисунок 17).

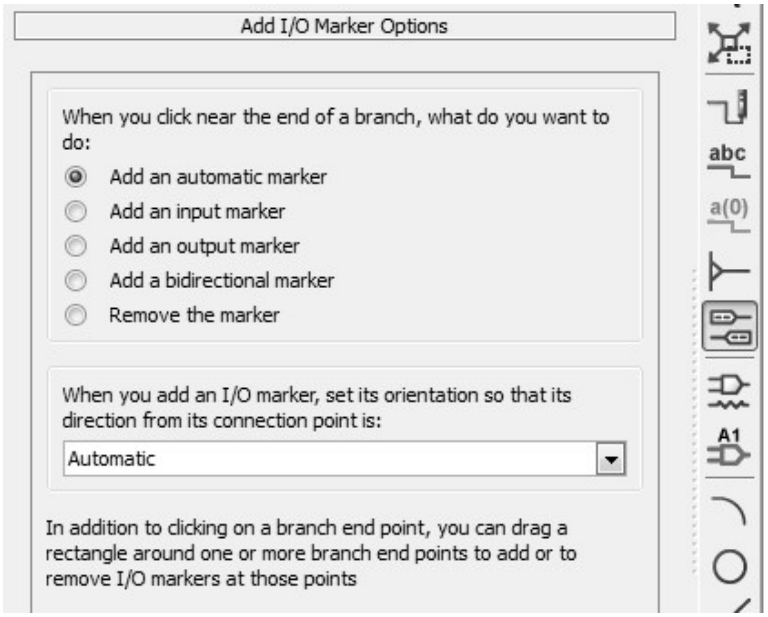


Рисунок 17 – Диалоговое окно параметров маркеров ввода/вывода

В таблицу 4 сведен перевод основных параметров маркеров из рисунка 17.

Таблица 4 – Настройки параметров маркеров ввода/вывода

When you click near the end of a branch, what do you want to do:	**При нажатии на конец ветви, что вы хотите сделать:**
Add an automatic marker	Автоматически добавить маркер
Add an input marker	Добавить входной маркер
Add an output marker	Добавить выходной маркер
Add a bidirectional marker	Добавить двунаправленный маркер
Remove the marker	Удалить маркер
When you add an I/O marker, set it's orientation?	При добавлении маркера какую ориентацию хотите задать?

После установления соответствующих параметров необходимо переместить курсив мыши в рабочую область редактора ECS и кликнуть мышью по выбранному порту схемы. Для окончательного оформления схемы рекомендуется дать маркерам входов и выходов собственные контекстные имена, отражающие назначение подключенным к ним сигналов. Для этого можно использовать контекстное меню маркеров, изменив имя порта (Рисунок 18) или свойства выделенного объекта схемы, изменив имя узла (Рисунок 19) [1].

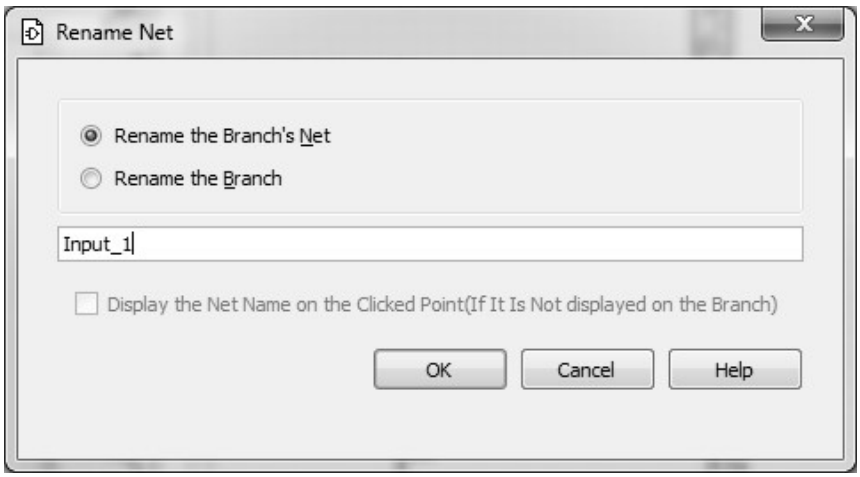


Рисунок 18 – Пример редактирования имени порта вывода

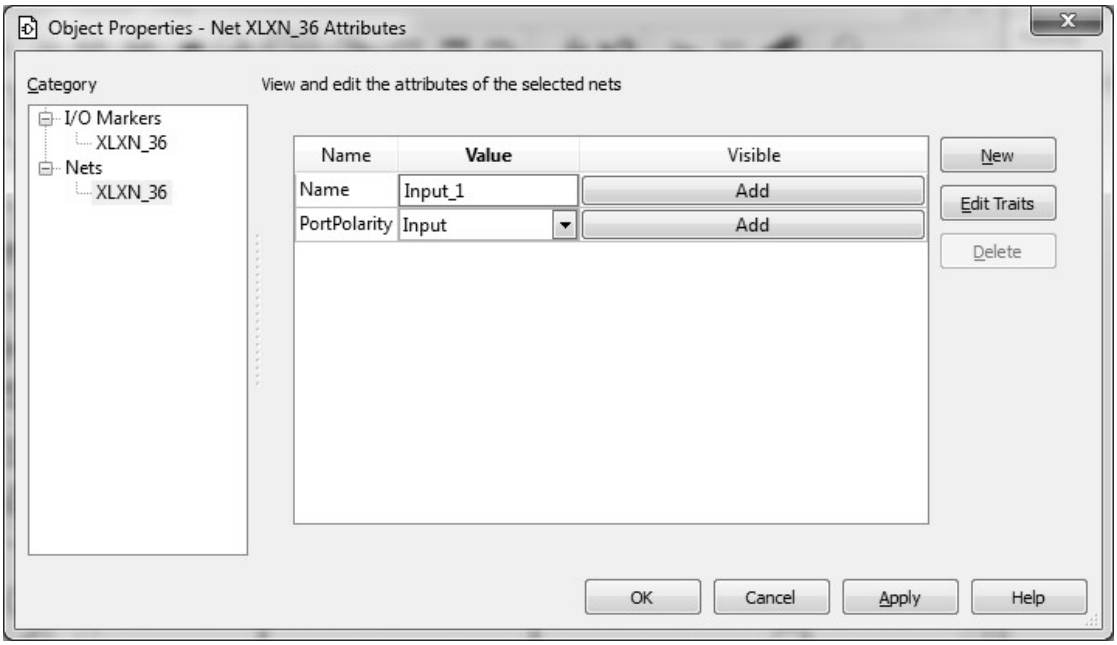


Рисунок 19 – Пример редактирования свойств порта вывода

2.4 Проектирование модуля на HDL- языке

Для проектирования цифрового устройства на языке Verilog (VHDL) нужно создать новый модуль (Project -> New Source…) и добавить его в проект (Рисунок 20).

Название Verilog-модуля в проекте лабораторного практикума – logic_2.

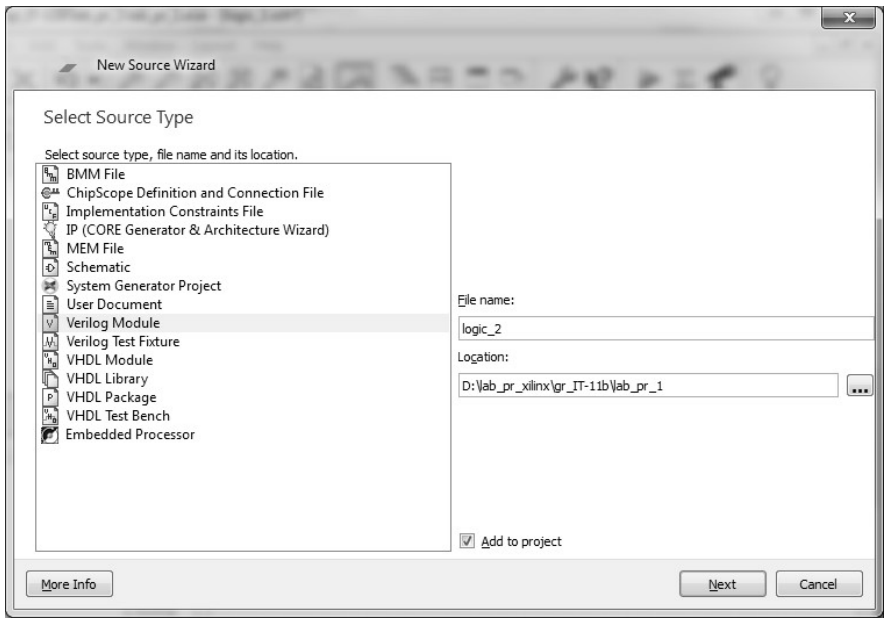


Рисунок 20 – Диалоговое окно по созданию модуля на языке Verilog

На следующем шаге следует определить интерфейс создаваемого логического модуля, указав параметры его соединений с другими модулями, т.е. имена и типы его портов. Пример назначения параметров модуля приведен на рисунке 21.

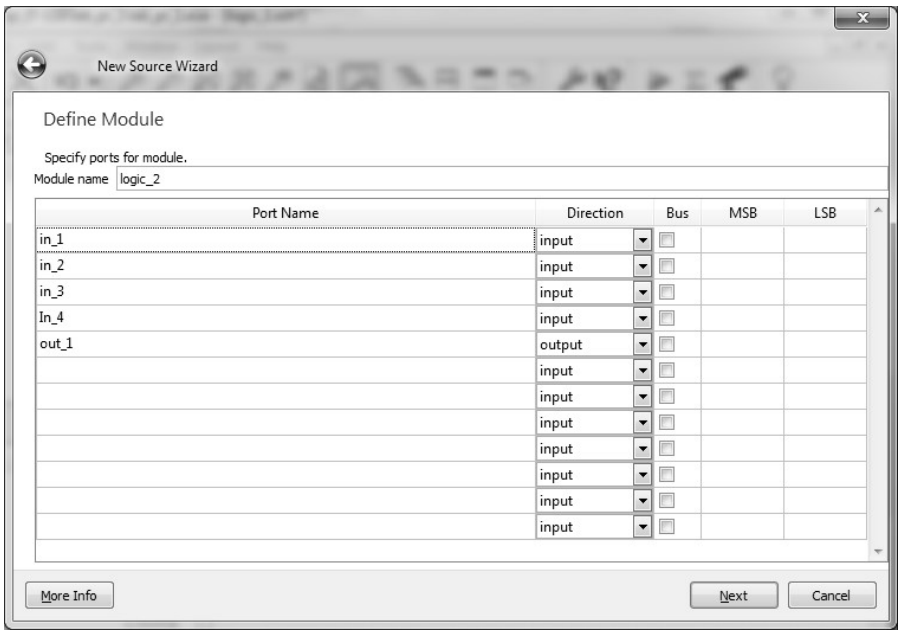


Рисунок 21 – Пример настройки параметров интерфейса модуля в САПР ISE

Далее следует нажать кнопку Next и проверить правильность введенных данных, изучив соответствующий отчёт (Рисунок 22) [1].

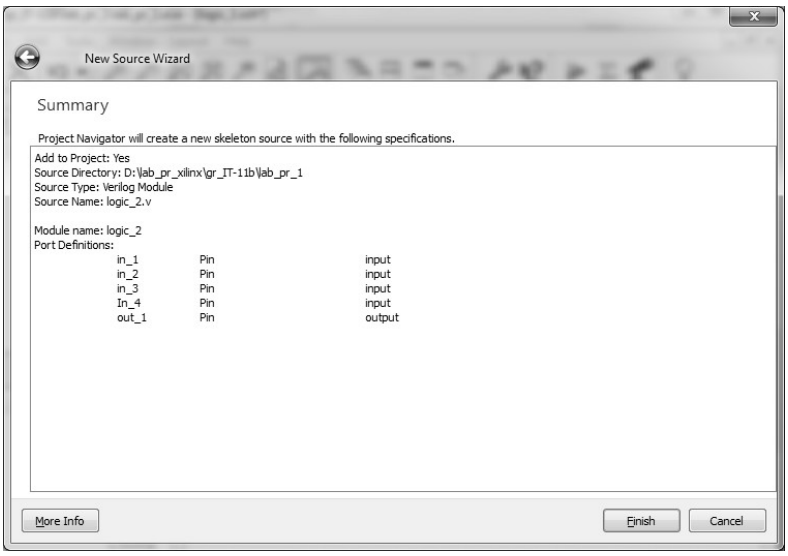


Рисунок 22 – Отчёт проверки установленных значений

После нажатия кнопки Finish, в навигаторе проекта появится новый модуль, а в окне документов станет доступным редактирование шаблона кода его описания. Работа с редактором кода аналогична работе с любым текстовым редактором. При вводе стандартных функций и директив языка для удобства работы цвет текста автоматически изменяется [1].

Для выполнения данного лабораторного практикума необходимо в созданный шаблон вписать строки, которые будут задавать функцию устройства в соответствии с заданием.

Пример (Рисунок 23):

```
assign pr_1 = ~(in_1 | in_2); // элемент 2ИЛИ-НЕ
assign pr_2 = in_3 & in_4; // элемент 2И
assign out_1 = pr_1 | pr_2; // элемент ИЛИ
```

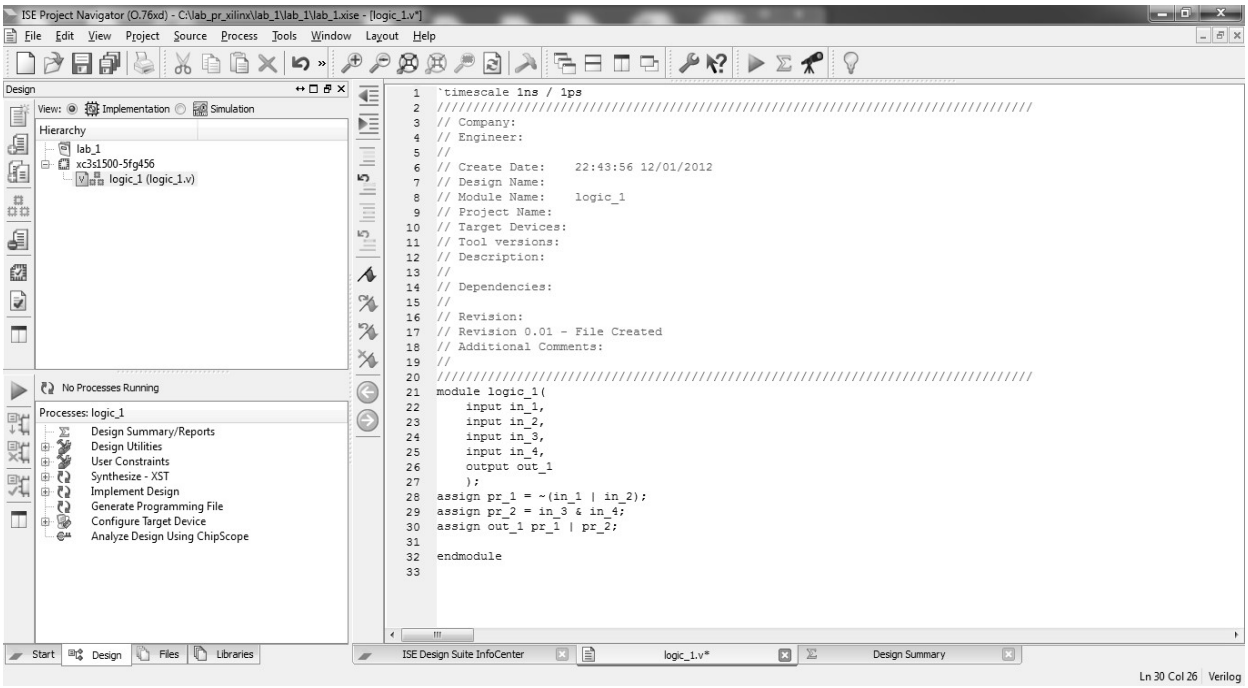


Рисунок 23 – Пример цифрового устройства, написанного на Verilog

После редактирования код необходимо сохранить (File->Save) и выполнить процедуру проверки синтаксиса. Для этого нужно в окне процессов раскрыть строку Synthesize – XST нажав на «+» и вызвать приложение Check Syntax двойным кликом мыши (Рисунок 24).

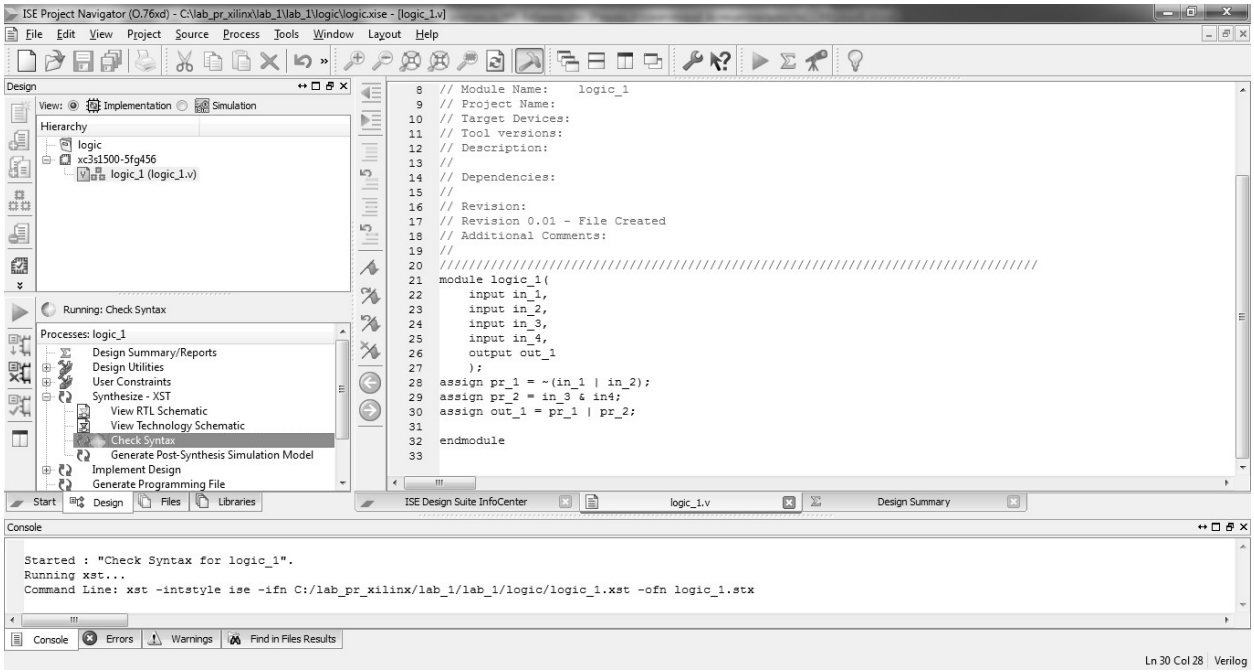


Рисунок 24 – Процесс проверки на синтаксис написанного кода на VerilogHDL

Сведения обо всех найденных ошибках отображаются в окне сообщений. Свидетельством отсутствия ошибок является в окне процессов значок «», в противном случае появляется «».

2.5 Создание символьного представления HDL- модуля и включение его в схемотехническое решение

Для создания символьного представления модуля и включения его в принципиальную схему следует выполнить следующие действия.

1) Выделить в окне исходных модулей проекта строку с названием модуля «logic_2 (logic_2.v)», после чего в окне Process откроется в список процессов, доступных для данного модуля.

2) Дважды щелкнуть на строке Design Utilites -> Create Schematic Symbol, проверить, что в окне консоли появилось сообщение об успешном выполнении компиляции графического символа логического модуля. Важно проверить корректность выполнения данной операции после каждого изменения кода проектируемого модуля и перезаписи соответствующего текстового файла кода, т.к. при ошибках САПР ISE будет всегда использовать в проекте последнюю корректно оттранслированную версию

компонента. Впоследствии это послужит источником ошибок в работе всего проекта, который будет трудно обнаружить (Рисунок 25).

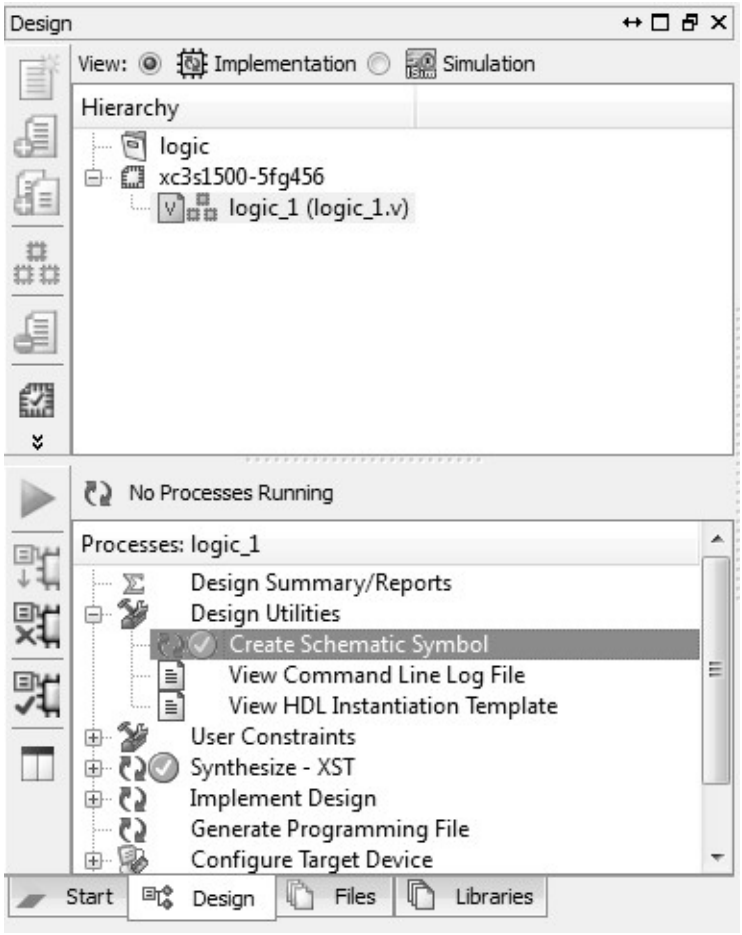


Рисунок 25 – Успешно созданный символьный элемент HDL- модуля

3) Открыть из окна модулей процесса документ с принципиальной схемой проекта, раскрыть закладку Symbol и убедиться, что в списке библиотечных элементов проекта в категории «<-- All symbols -->» появился модуль logic_2 (Рисунок 26). Компоненты, созданные пользователем, сохраняются в отдельной группе, название которой соответствует пути к папке проекта. Поэтому, в частности, нужно, чтобы проект размещался вне общей папки с САПР ISE. Чтобы поместить символьное представление модуля на принципиальную схему нужно кликнуть левой кнопкой мыши на его наименовании и захватив мышкой, перенести на схему [1].

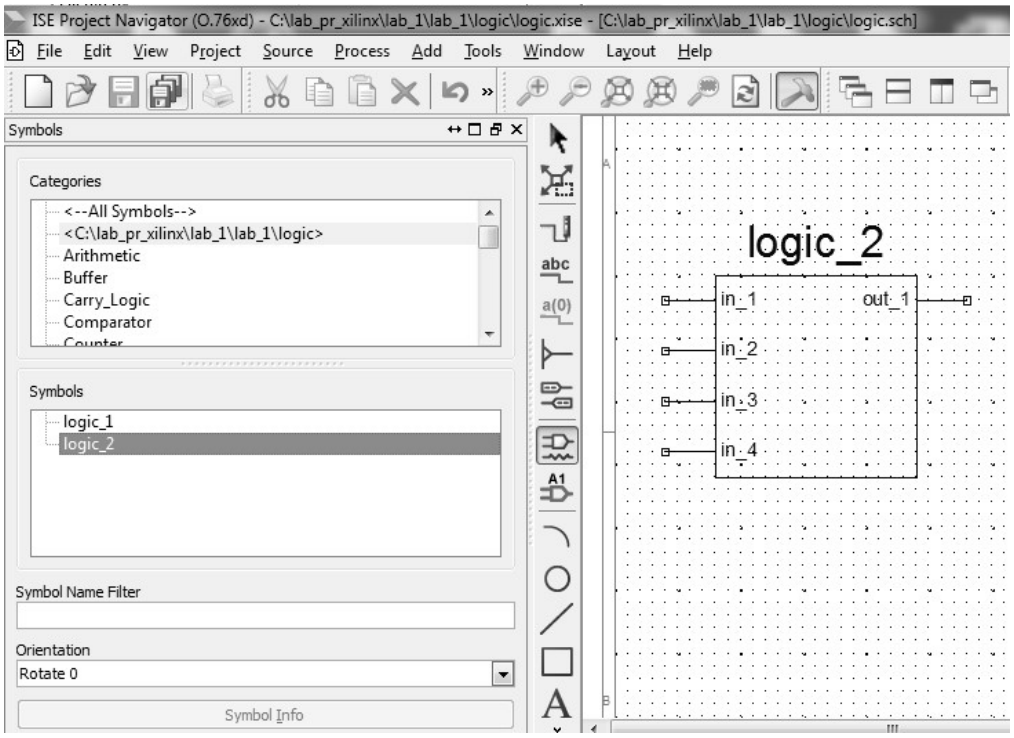


Рисунок 26 – Пример размещение в графическом редакторе созданного символьного модуля на основе HDL- описания

2.6 Размещение разработанного устройства в ПЛИС

На этапе размещения устройства в конкретной ПЛИС, пользователь может просмотреть как реализован проект на уровне описания цепей сигналов и сохранения их состояний в регистрах (Register Transfer Level, RTL-уровень). Для этого нужно вызвать из раздела списка Synthesize –XST процедуру View RTL Schematic и включить, как показано на рисунке 27, второй пункт в окне представления результатов [1].

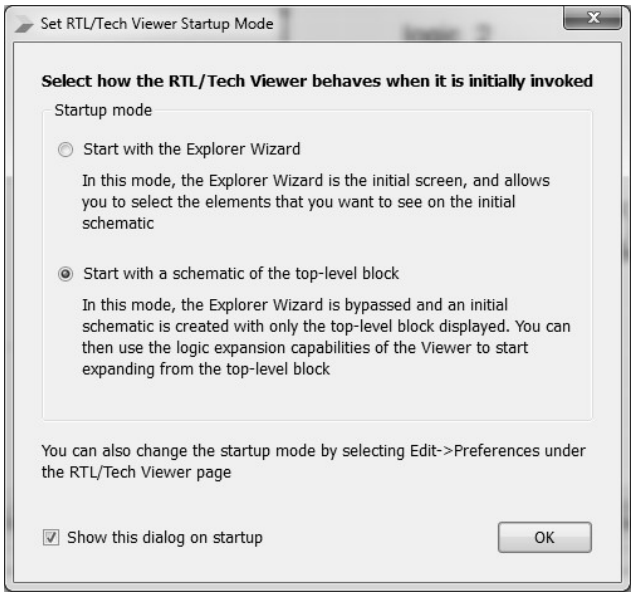


Рисунок 27 – Диалоговое окно представления результатов выполнения процедуры Synthesize –XST - View RTL Schematic

После нажатия OK появится окно разрабатываемого устройства logic_1 в виде модуля на верхнем уровне иерархии (Рисунок 28).

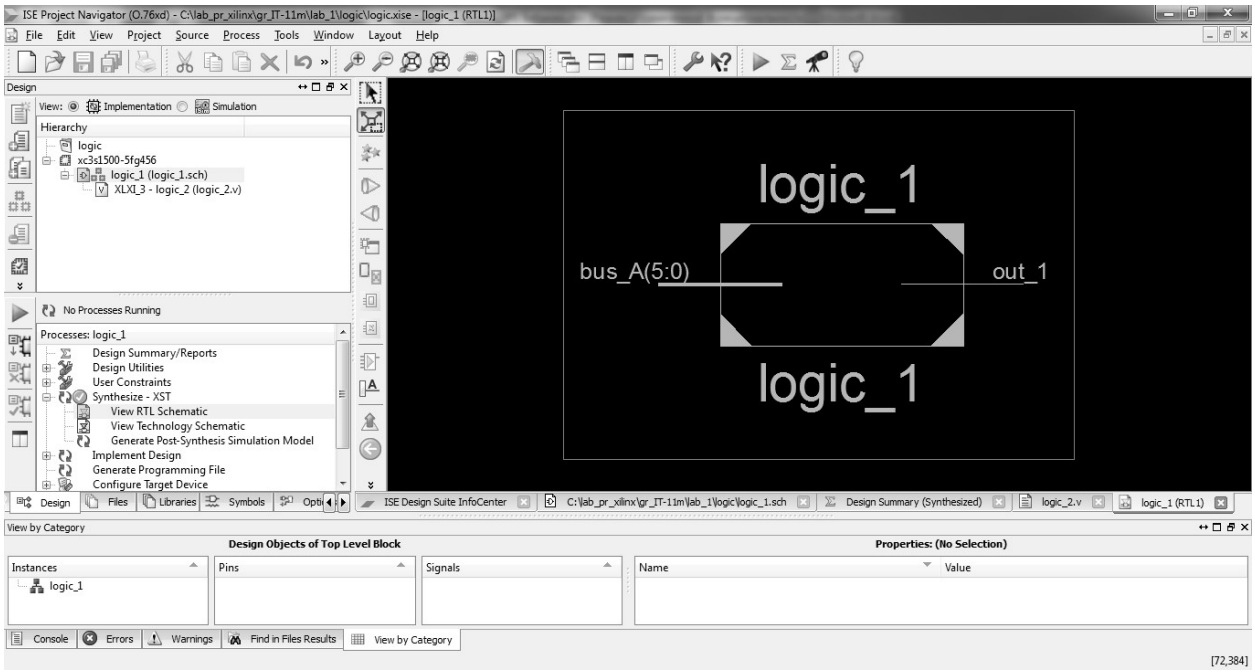


Рисунок 28 – Разрабатываемое устройство logic_1 на верхнем уровне иерархии RTL

При двойном клике мыши по интересующему элементу появляется модуль на нижнем уровне иерархии (Рисунок 29).

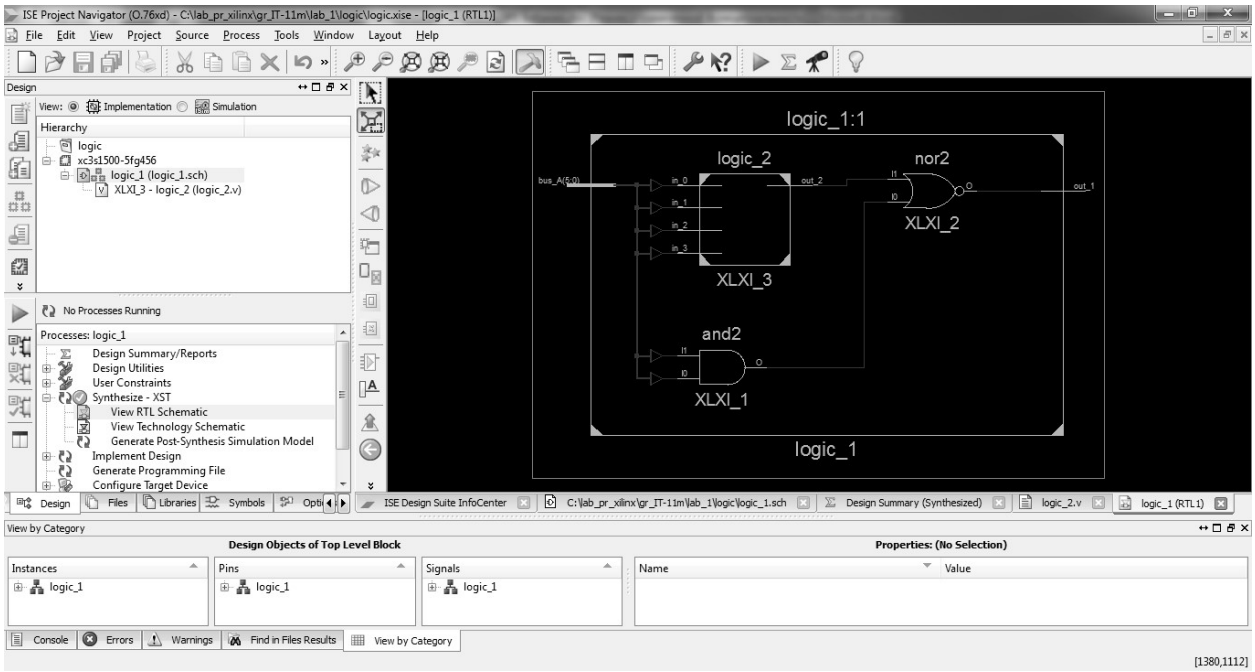


Рисунок 29 – Разрабатываемое устройство logic_1 на нижнем уровне иерархии RTL

При двойном клике по строке, показанной на рисунке 30, можно узнать как транслятор ISE распределил ресурсы в ПЛИС (Рисунок 31)

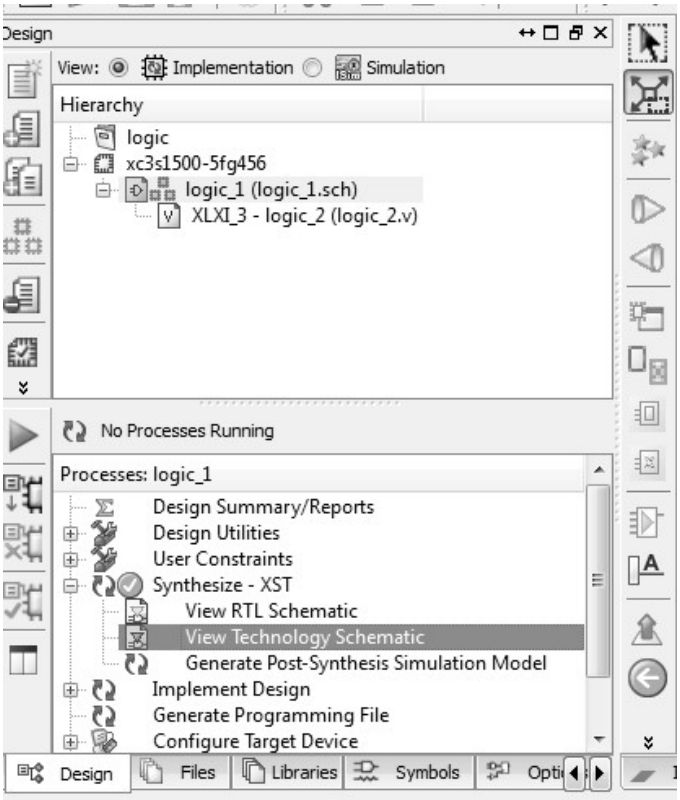


Рисунок 30 – Вызов схемотехнического представления устройства logic_1 в ПЛИС

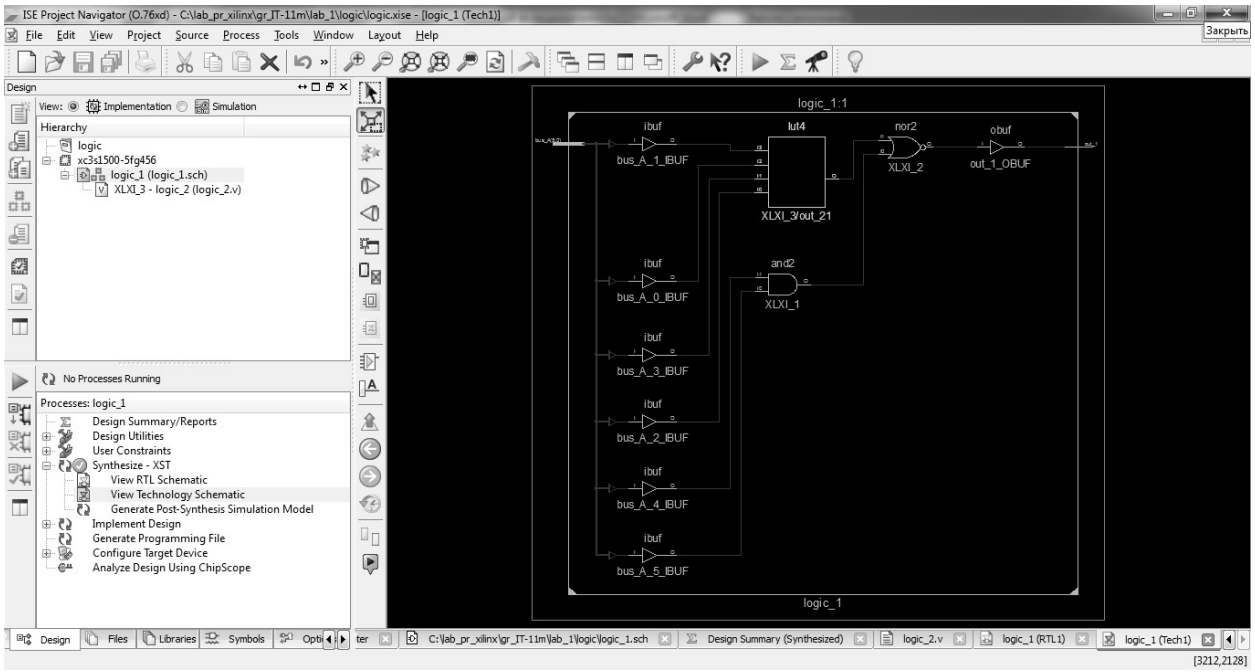


Рисунок 31 – Схемотехническое представление устройства logic_1 в ПЛИС

При двойном клике левой клавишей мыши по элементу logic_2 откроется диалоговое окно, где можно узнать информацию о распределении ресурсов данного модуля в ПЛИС (Рисунок 32), его карту Карно (Рисунок

33), таблицу истинности (Рисунок 34), математическое представление (Рисунок 35).

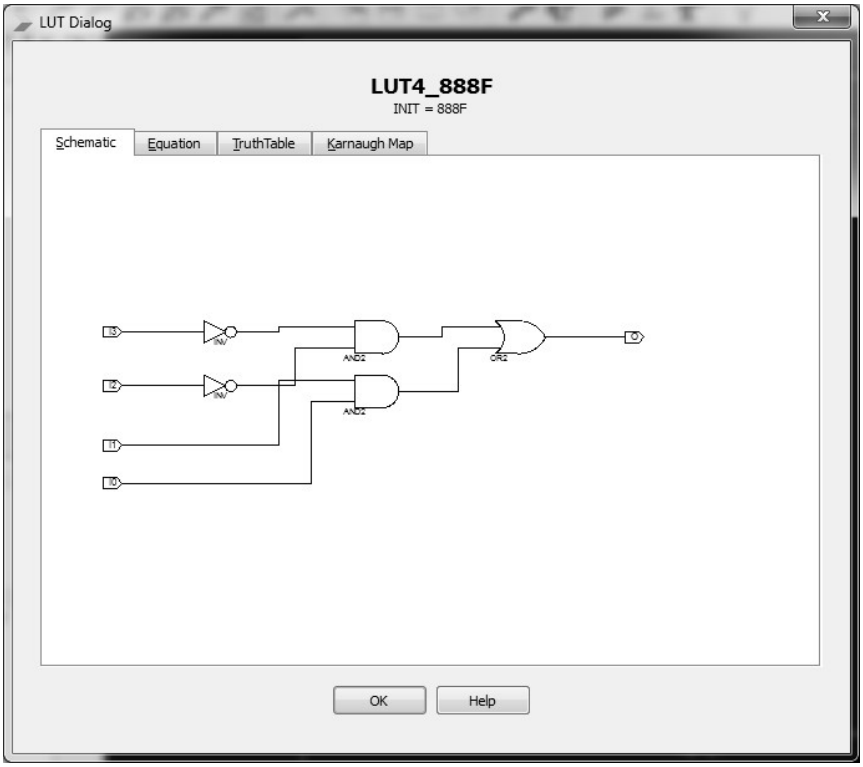


Рисунок 32 – Схема распределения ресурсов в ПЛИС модуля logic_2

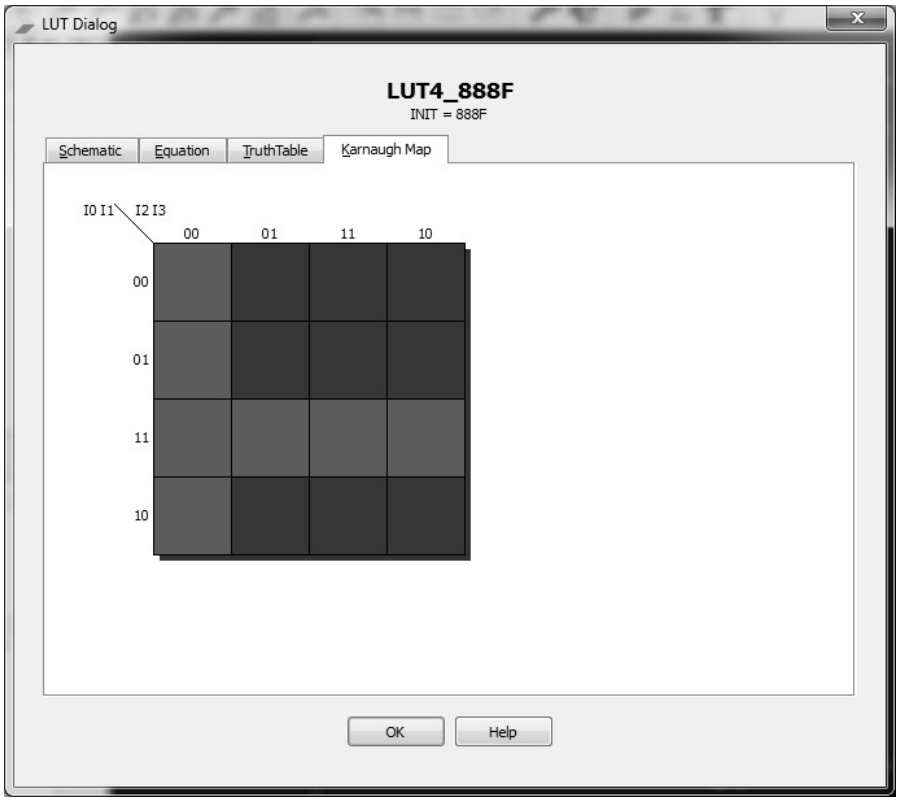


Рисунок 33 – Карта Карно модуля logic_2

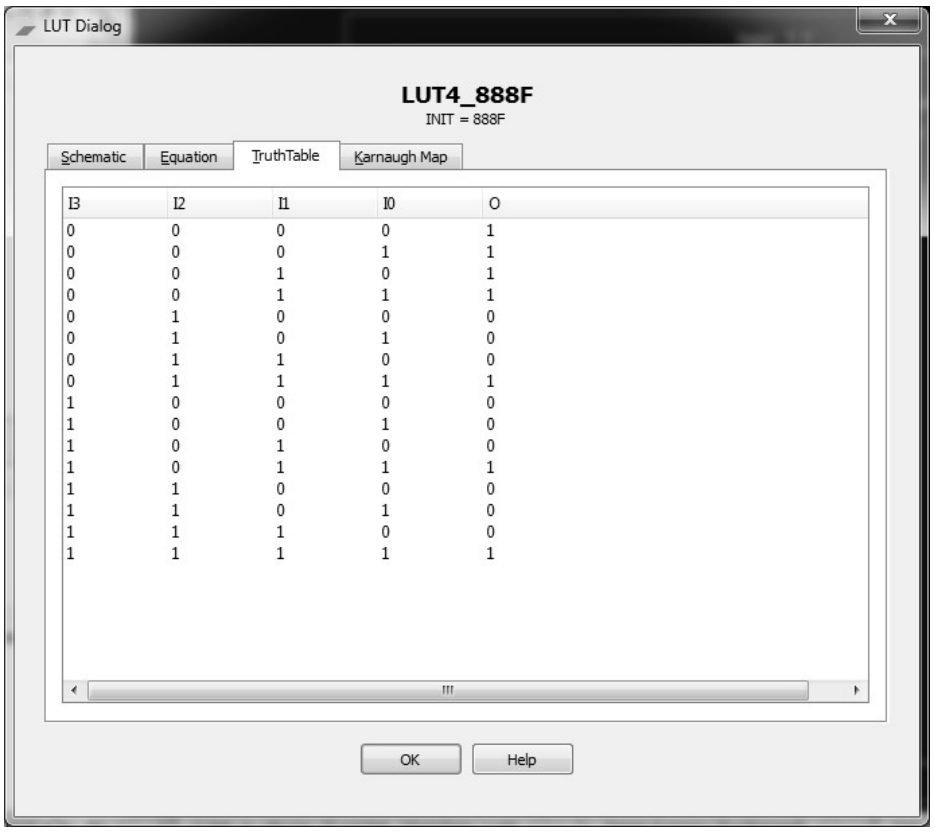


Рисунок 34 – Таблица истинности модуля logic_2

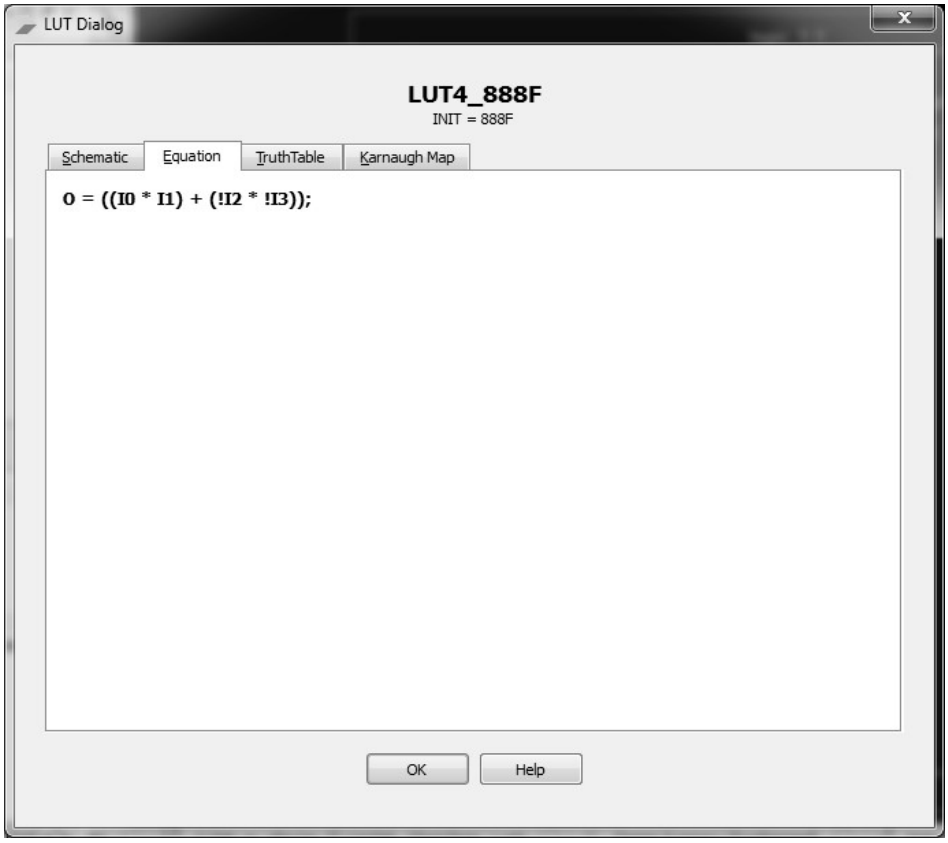


Рисунок 35 – Математическое представление модуля logic_2, где «*» - операция ИЛИ; «+» - операция И; «!» - операция НЕ

2.7 Создание тестового модуля и функциональное моделирование разработанного цифрового устройства

Этап функционального моделирования позволяет выполнить предварительную верификацию проекта. На этой стадии отсутствует информация о значениях задержек распространения сигналов, поэтому при функциональном моделировании можно обнаружить только логические и синтаксические в описании разрабатываемого устройства. Для функционального моделирования проекта применяется библиотека UniSim Library, элементы которой имеют единичные задержки [3].

При моделировании используется подход, основанный на создании и применении специальной моделирующей программы - «испытательного стенда» (testbench). Для этого моделируемое (тестируемое) устройство (в англоязычной литературе - UUT, Unit Under Test) представляется своим синтезируемым кодом (внутренний модуль испытательного стенда), а для проверки его поведения в различных условиях создаются описания тестовых воздействий («моделирующий код», т.е. внешний модуль испытательного стенда) [1].

Для создания набора тестовых воздействий необходимо выполнить следующие действия.

1) Выбрать в окне описания проекта файл «logic».
2) Для создания нового набора тестовых воздействий выбрать пункт меню Project-> New Source.
3) В открывшемся окне выбрать тип исходного файла «Verilog Test Fixture» и ввести имя для создаваемого набора «tb_logic» в поле File Name (Рисунок 36).

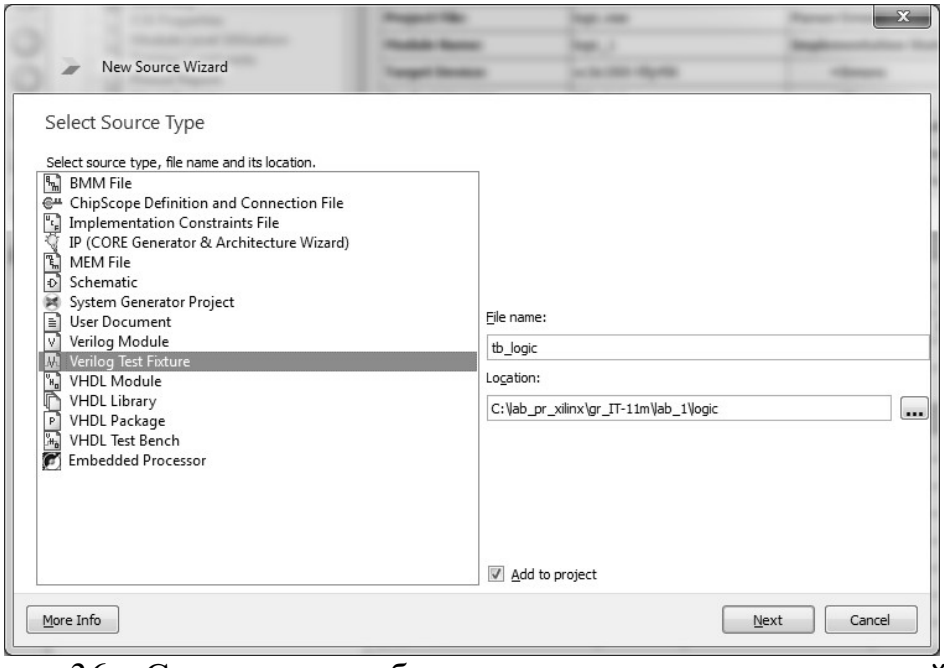


Рисунок 36 – Создание «тестбенча» для проектируемого устройства

4) Нажать кнопку Next.

5) В открывшемся окне привязок необходимо выбрать файл с описанием устройства, к которому будет привязан набор тестовых воздействий. Для выполнения данного лабораторного практикума необходимо выбрать файл «logic_1» (Рисунок 37).

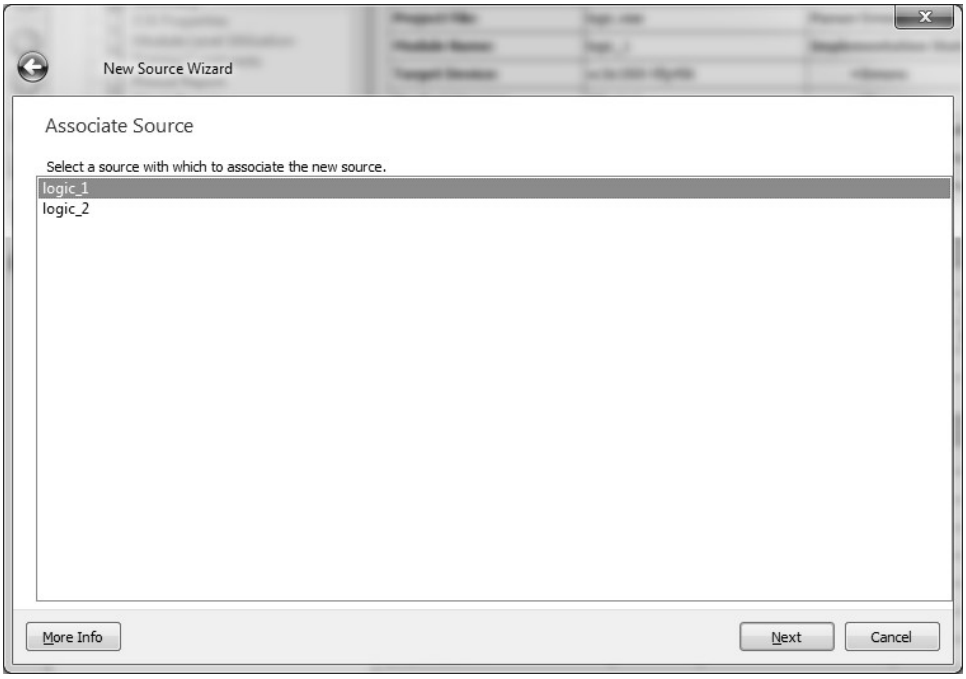


Рисунок 37 – Выбор устройства к которому будет привязан набор тестовых воздействий

6) Нажать кнопку Next.

7) В окне резюме проверить соответствие файла с исходным описанием устройства и привязанного к нему набора тестовых воздействий (Рисунок 38).

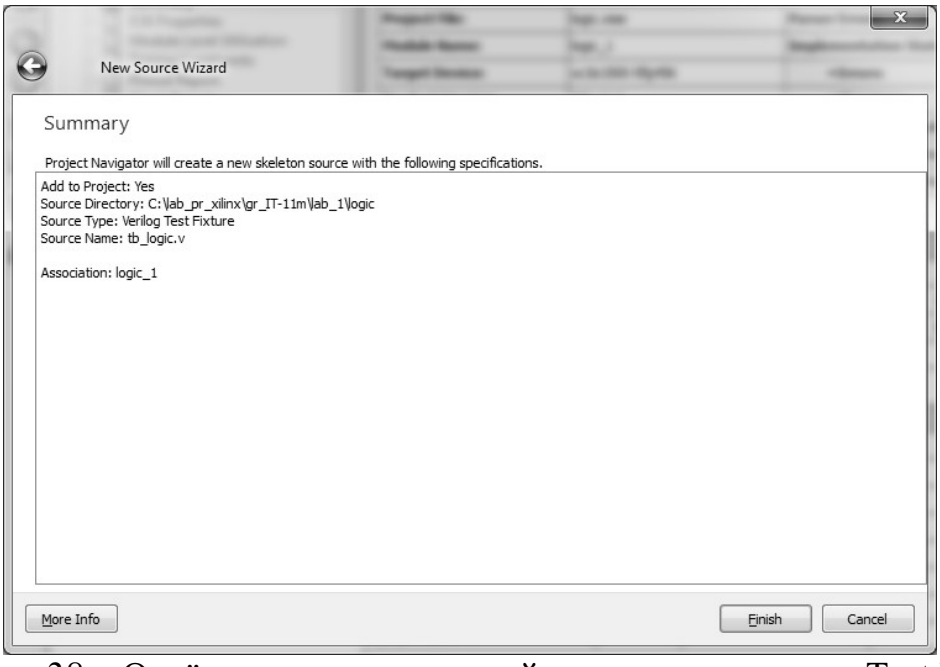


Рисунок 38 – Отчёт о выполненных действиях по созданию Test Bench

8) Нажать кнопку Finish [3].

9) В созданном Verilog Test Fixture модуле содержится код для тестирования разрабатываемого устройства (Рисунок 38)

.

```
1  // Verilog test fixture created fr
2
3  `timescale 1ns / 1ps
4
5  module logic_1_logic_1_sch_tb();
6
7  // Inputs
8     reg [5:0] bus_A;
9
10 // Output
11    wire out_1;
12
13 // Bidirs
14
15 // Instantiate the UUT
16    logic_1 UUT (
17       .out_1(out_1),
18       .bus_A(bus_A)
19    );
20 // Initialize Inputs
21    `ifdef auto_init
22        initial begin
23       bus_A = 0;
24    `endif
25 endmodule
26
```

Рисунок 38 – Сгенерированный test bench для разрабатываемого устройства

Данный исходный код содержит:

- директиву трансляции `timescale, которая указывает транслятору, что нужно задать условное время моделирования 1ns с шагом дискретизации 1ps;

- модуль верхнего уровня иерархии test bench:

module logic_1_logic_1_sch_tb ();

…

endmodule;

- определения и обозначения входных и выходных сигналов тестового модуля:

reg [5:0] bus_A;

wire out_1;

- модуль тестируемого устройства:

logic_1 UUT (

.out_1 (out_1),

.bus_A (bus_A)

);

- директивы условной трансляции, выполняющие функции задания начальных значений переменным модуля верхнего уровня [1]:

`ifdef auto_init
initial begin
bus_A = 0;
`endif

Для создания последовательности входных данных (с целью получения таблицы истинности испытуемого модуля) код необходимо изменить согласно рисунку 39.

```
1  // Verilog test fixture created from s
2
3  `timescale 1ns / 1ps
4
5  module logic_1_logic_1_sch_tb();
6
7  // Inputs
8     reg [5:0] bus_A;
9
10
11 // Output
12    wire out_1;
13
14 // Bidirs
15
16 // Instantiate the UUT
17    logic_1 UUT (
18       .out_1(out_1),
19       .bus_A(bus_A)
20    );
21 // Initialize Inputs
22
23    initial |
24    begin
25    bus_A = 0;
26    #320 bus_A[5]=~bus_A[5];
27    #320 $finish;
28    end
29
30    always #10 bus_A[0]=~bus_A[0];
31    always #20 bus_A[1]=~bus_A[1];
32    always #40 bus_A[2]=~bus_A[2];
33    always #80 bus_A[3]=~bus_A[3];
34    always #160 bus_A[4]=~bus_A[4];
35
36 endmodule
```

Рисунок 39 – Test Bench для проверки логики работы разрабатываемого устройства

Рассмотрим подробнее описание команд выше представленного текста на языке аппаратного программирования Verilog.

Ключевые слова input и output определяют направление выводов: вход и выход соответственно. Для определения назначения ключевого слова wire следует отметить понятие сигнал (signal). Сигналы – это электрические импульсы, которые передаются по проводам (wire) между логическими элементами схемы. Провода переносят информацию, не производя над ней никаких вычислений. В цифровой схеме сигналы важны для передачи двоичных данных [5].

Один из базовых типов источника сигнала в языке Verilog – это цепь или проводник, wire. Таким образом, если у вас есть арифметическое или логическое выражение, вы можете ассоциировать результат выражения с именованным проводником и позже использовать его в других выражениях. Это немного похоже на переменные, только их (как провода в схеме) нельзя пересоединить на лету, нельзя поменять назначение. Значение проводника (wire) – это функция того, что присоединено к нему.

Еще существует другой тип источника сигнала называемый регистр: reg. Регистр reg в языке Verilog скорее обозначает переменную, которая может хранить значение, чем аппаратный регистр. Тип reg используют при поведенческом (behavioral) и процедурном описании цифровой схемы. Если регистру постоянно присваивается значение комбинаторной (логической) функции, то он ведет себя точно как проводник (wire). Если же регистру присваивается значение в синхронной логике, например по фронту сигнала тактовой частоты, то ему, в конечном счете, будет соответствовать физический D-триггер или группа D-триггеров.

D-триггер – это логический элемент способный запоминать один бит информации. В англоязычных статьях D-триггер называют flipflop [4].

Блоки типа always названы в силу того, что они реализуют некий процесс, который может повторяться многократно. В литературе их часто называют always- блоками по ключевому слову, определяющему такой блок. Вкратце можно сказать следующее- в подобном блоке всегда есть некоторое выражение, в правой части которого стоит переменная, блок выполняет процесс всегда (always), когда эта переменная изменяет свое значение или происходит какое-либо событие (event). Формальный синтаксис always-блока приведен ниже:

always [<event control>]
[begin | fork] [:<block_name>]
<block_statements>;
[end | join]

Опциональное поле event_control содержит т.н. список чувствительных блоков, т.е. список событий (events) при возникновении которых блок начнет выполняться. Если список отсутствует, он создается автоматически по именам переменных в правой части выражений block_statements.

Если в блоке только одно выражение, то ключевые слова begin-end и fork-join можно опустить (хотя и оставить их вполне можно). Какую именно пару можно применять и когда более подробно описано в книгах [6, 7]. Чаще

всего применяется пара begin-end, что определяет блок с последовательным выполнением операторов.

Блоки установки начальных состояний вводятся ключевым словом initial. Главное отличие таких блоков от always-блоков в том, что они выполняются один раз в самом начале моделирования, на момент времени 0. Эти блоки не поддерживаются системами синтеза, но широко используются в test bench- модулях. Формальный синтаксис такого модуля выглядит так:

initial
[begin | fork] [:<block_name>]
<block_statements>;
[end | join]

Списка чувствительности в таких блоках нет, поскольку он выполняется только один раз в начале моделирования. Как видно из его названия, наиболее часто эти блоки применяются для установки начальных состояний. Однако вводя задержку и операторы присвоения, можно сформировать некоторую временную диаграмму. В одном моду может быть несколько initial- блоков, их количество не ограничивается. Чаще всего в блоках используется пара begin-end, хотя пара fork-join тоже допустима.

Задержки в описании обозначают #<time>. Данный тип команды показывает на какое время задержать выполнение какой-либо операции. Значение <time> имеет условный временной интервал. Величина условный единицы задается директивой трансляции `timescale. По умолчанию она равна 1 пс.

Системные функции служат различным целям, например, управление процессом моделирования, ввод/вывод форматированных данных, получение информации о реальном времени какого-либо события в общем они представляют мощный и удобный инструментарий разработчика Системные функции никогда не применяются в синтезируемых кодах, поскоку чаще всего оперируют с несинтезируемыми объектами. В написанном коде системной функцией является $finish, которая необходима для остановки процесса моделирования.

Более подробно с Verilog HDL можно ознакомиться изучив книги [6,7].

10) После написания testbech необходимо проверить код на синтаксические ошибки. Для этого нужно сохранить файл, убедиться, что включен режим Simulation и выбрана схема проверки функционального поведения устройства (Behavioral) (Рисунок 40).

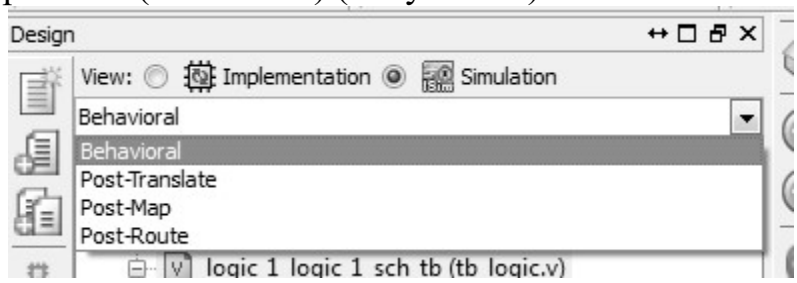


Рисунок 40 – Проверка установленных значений

Затем в окне процессов раскрыть вкладку ISim Simulator и двойным кликом нажать на строку Behavioral Check Syntax (Рисунок 41).

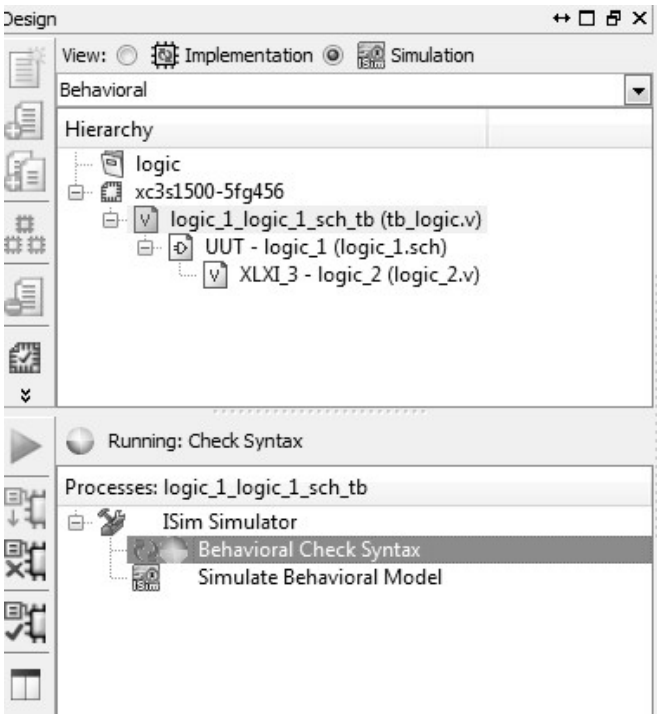


Рисунок 41 – Процесс проверки синтаксических ошибок в TestBench

В случае успешной проверки появится напротив надписи Behavioral Check Syntax зеленый кружок с галочкой, в противном случае красный кружок с крестиком.

11) Для симуляции в окне процессов кликаем два раза мышью по строке Simulate Behavioral Model. В результате должно открыться окно программы iSim (Рисунок 42, 43).

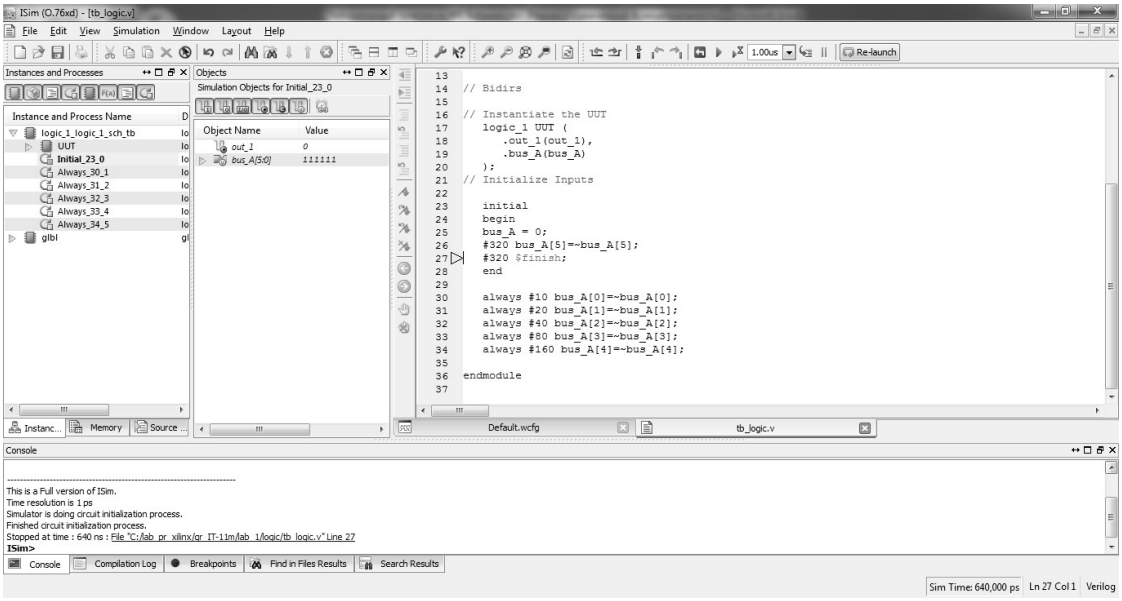


Рисунок 42 – Окно встроенного средства моделирования iSim

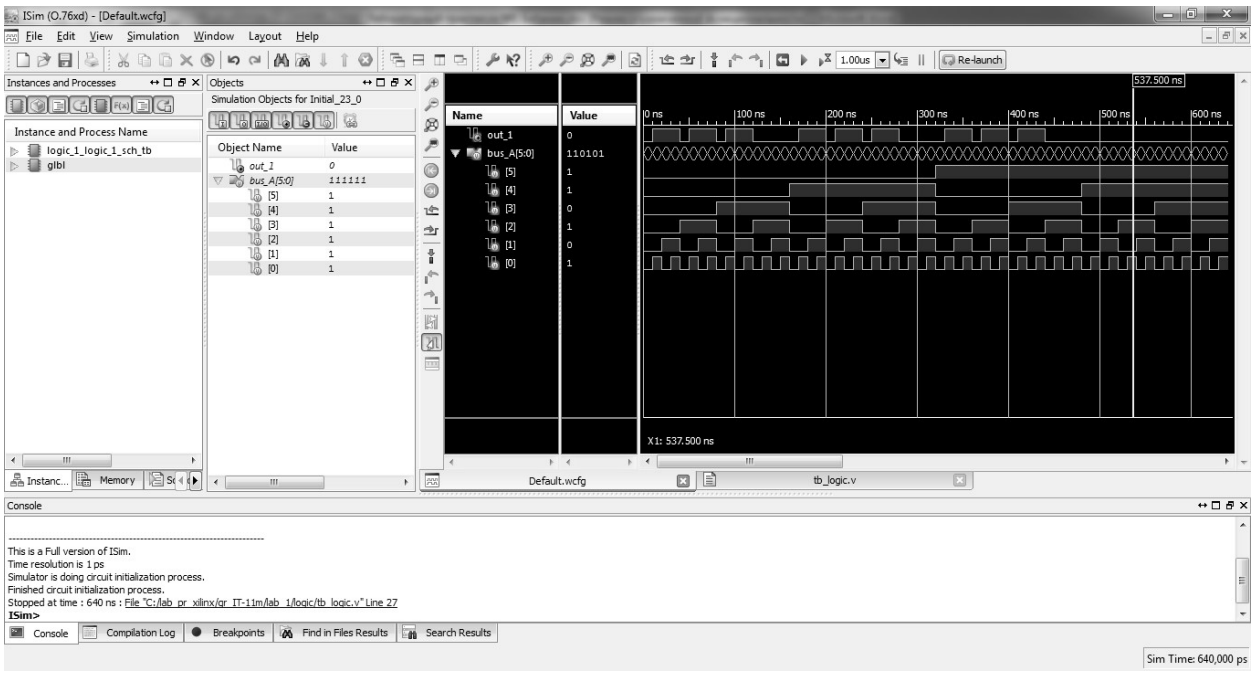
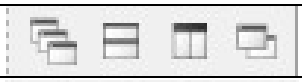
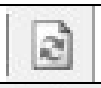
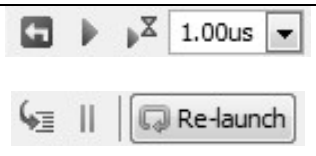


Рисунок 43 – Временные диаграммы тестируемого устройства

Инструмент моделирования iSim содержит следующие элементы управления: главное меню, палитру инструментов, окно имен классов и процессов проекта (Instance and Process Name), окно объектов моделирования (Object Name) и окно временной диаграммы с результатами моделирования.

Функции программы симулятора снабжены контекстными всплывающими подсказками. Основные функции палитры инструментов, необходимые для анализа временной диаграммы перечислены в таблице 5.

Таблица 5 – Основные функции палитры инструментов

Пиктограммы	**Назначение элементов управления**
	Вызов справочный системы, информация о выделенном объекте
	Настройка размещения окон программы
	Отображение текущего вида диаграммы – изменение масштаб, отображение всей диаграммы, отображение фрагмента между маркерами
	Перерисовать все окна
	Перейти к предыдущему или следующему моменту изменения состояния диаграммы, добавить маркер, сделать активным предыдущий или следующий маркер
	Управление запуском и остановкой временной диаграммы: сброс симуляции, постоянная работа, выполнить моделирование до момента, указанного на панели инструментов, задание общего времени моделирования, выполнить по шагам, остановить моделирование и перезапустить сначала

	Перейти к началу, перейти к концу
	Поменять местами маркеры
	Показывать моменты изменения состояния (переходы)
	Поместить скользящую шкалу

При работе с данным приложением следует иметь в виду, что при необходимости внести коррективы в текстовый файл, например, для изменения времени событий, следует в окне Instance and Process Name щелчком мыши на названии текстового файла открыть редактор кода, внести необходимые изменения, запомнить файл, снова проверить его синтаксис. После этого в окне документов нужно открыть временную диаграмму, перезапустить ее симулирование. При закрытии программы симулирования и переходе к интерфейсу навигатора проекта нужно сохранить в проекте новую редакцию текстового файла.

Другая последовательность действий при необходимости внесения изменений в файл текстовых воздействий: закрыть программу iSim, выполнить редактирование текстового файла и проверку его синтаксиса, а затем снова вызвать процесс Simulate Behavioral Model.

Полученную при моделировании временную диаграмму изменения состояний сигналов (цепей и переменных) можно сохранить в проекте для последующего просмотра. Соответствующий файл имеет расширение .wcfg [1].

2.8 Описание подключения разработанного цифрового устройства к внешним элементам проектируемой системы по отношению к ПЛИС

Файл проектных ограничений (Implementation Constrain File) служит для подключения выводов микросхемы ПЛИС к проектируемому устройству. Добавить в проект его можно двумя способами:

1) С помощью мастера создания новых проектных модулей (New Source), и последующего вызова процедуры Edit Constrains (Text), открывающей текстовый файл для редактирования.

2) В окне Process путем выбора пункта User Constrains и запуска одного из процессов I/O PlanAhead (Рисунок – 44) или Floorplan Area/IO/Logic (PlanAhead) в списке процессов, доступных для модуля верхнего уровня.

Для выполнения данного лабораторного практикума примем 2 вариант.

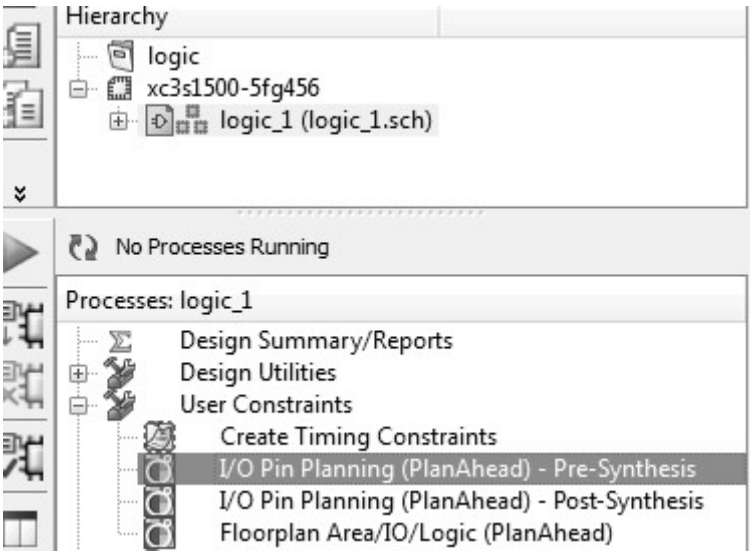


Рисунок 44 – Вызов PlanAhead для подключения разрабатываемого устройства к выводом ПЛИС

После произведения двойного клика мышью по I/O Pin Planning – Pre-Synthesis откроется окно, показанное на рисунке 45.

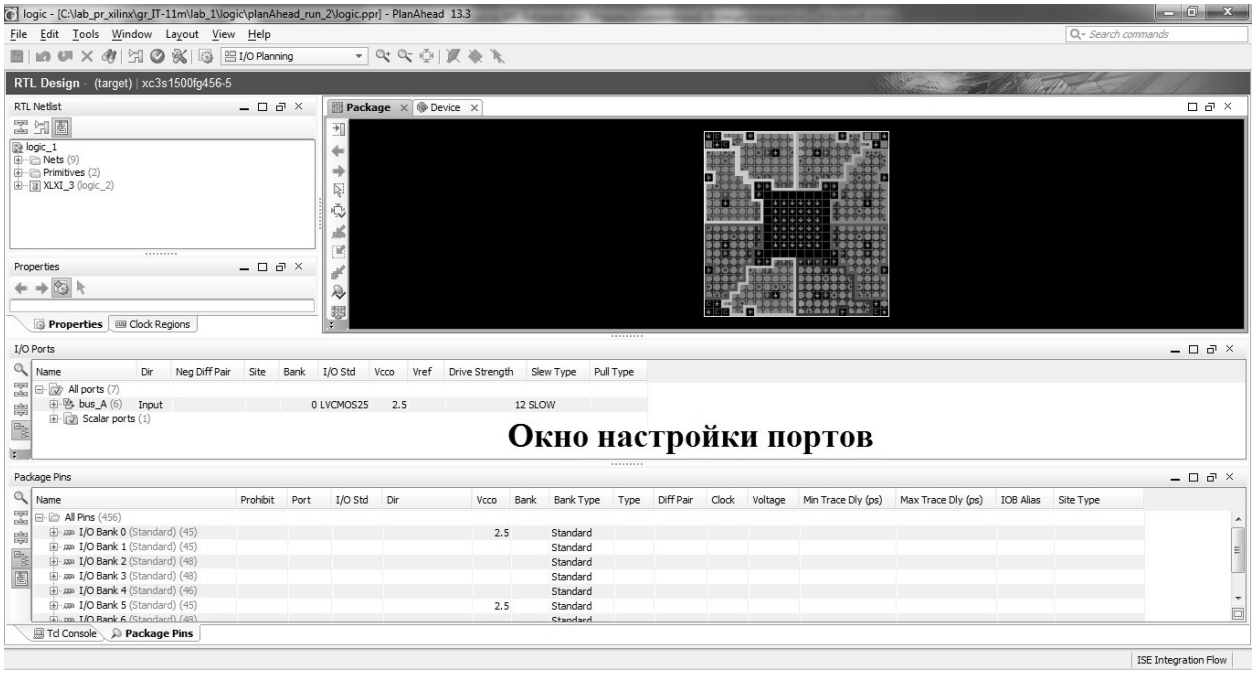


Рисунок 45 –Диалоговое окно PlanAhead

Для произведения подключений в окне настройки портов необходимо в столбце Site назначить порты ПЛИС соответствующим входам/выходам проектируемого устройство. Назначение портов произвести согласно рисунку 46.

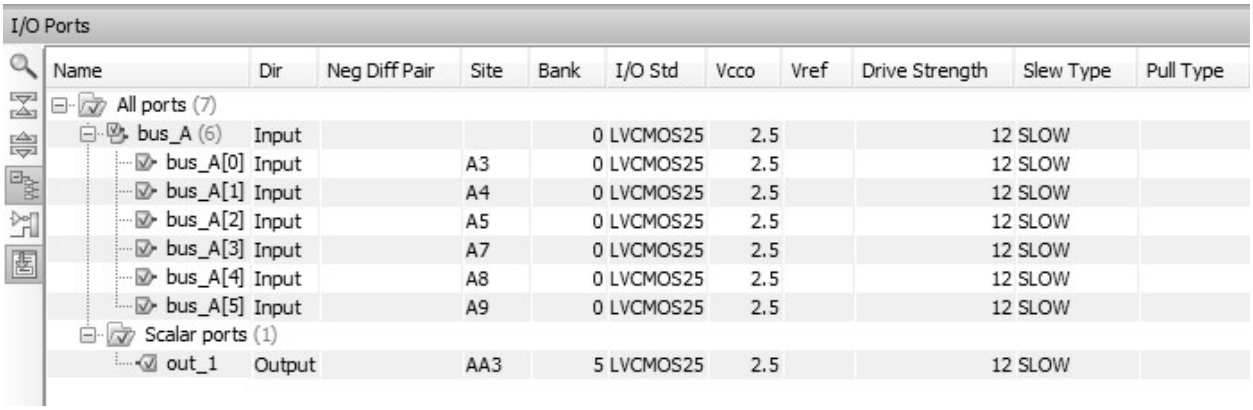


Рисунок 46 – Окно настройки портов ПЛИС

Далее необходимо сохранить данные настройки с помощью Fili--> Save File. В результате будет создан файл с расширением .usf.

Для того, чтоб открыть данный файл и убедиться в верности подключений выводом нужно открыть ISE Project Navigator и произвести двойной клик мышью в окне процессов по строке Edit Constraints (Text), представленной на рисунке 47.

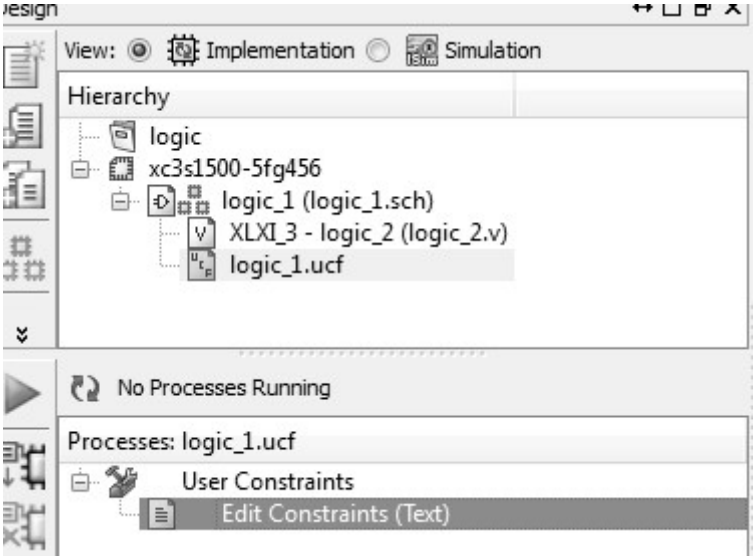


Рисунок 47 – Запуск Edit Constraints (Text)

Появится текстовый файл с кодом:

```
1
2     # PlanAhead Generated physical constraints
3
4     NET "bus_A[0]" LOC = A3;
5     NET "bus_A[1]" LOC = A4;
6     NET "bus_A[2]" LOC = A5;
7     NET "bus_A[3]" LOC = A7;
8     NET "bus_A[4]" LOC = A8;
9     NET "bus_A[5]" LOC = A9;
10    NET "out_1" LOC = AA3;
```

Команда NET "bus_A[1]" LOC = A4 означает, что сигнал проводника 1 шины bus_A подключен к выводу ПЛИС A4.

Для команды NET "out_1" LOC = AA3 – сигнал out_1 соединен с контактом на ПЛИС AA3.

В случае нахождения несоответствия код необходимо исправить и произвести сохранения файла.

2.9 Загрузка проекта в ПЛИС

После трассировки и построения внутренней структуры необходимо сформировать файл прошивки ПЛИС. Для этого производится в окне модулей выбор проекта logic_1 и двойным кликом мыши в окне процессов запускается процесс генерации соответственного файла (Generate Programming File) (Рисунок 48).

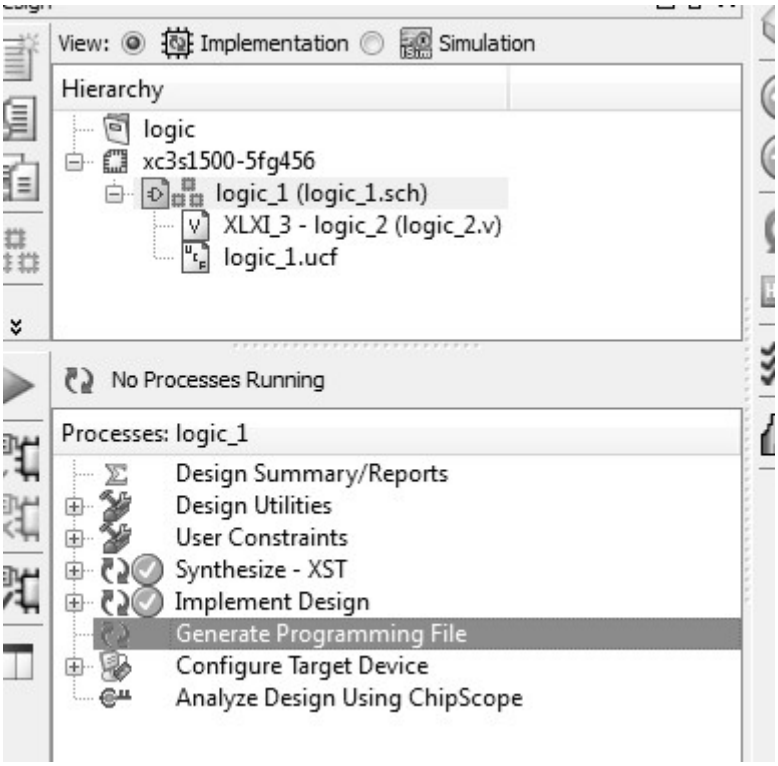


Рисунок 48 – Вызов процесса для генерирования файла прошивки

При успешном создании файла прошивки с расширением .bit напротив надписи процесса появится зеленый кружок с галкой.

Для прошивки файла программы в ПЛИС (или во внешнюю загрузочную память) необходимо в области действий над проектом запустить пункт Configure Target Device (Рисунок 49).

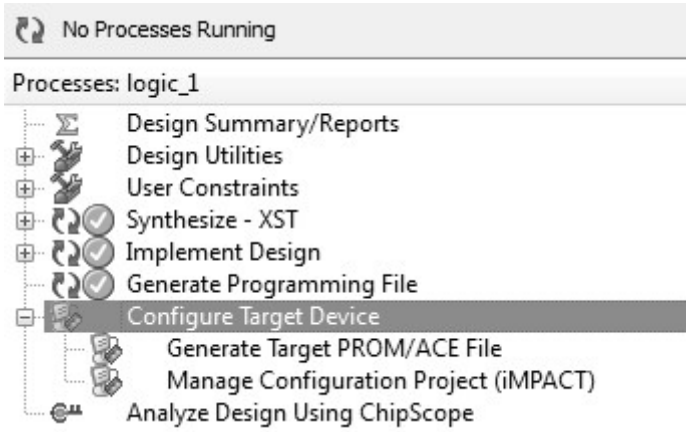


Рисунок 49 – Диалоговое окно процессов с выбранной строкой для осуществления запуска процесс прошивки ПЛИС

3 Задание на лабораторный практикум

Реализовать, отладить цифровую схему согласно рисунку х на базе ПЛИС Spartan 3 xc3c1500-5fg456, а также подготовить документацию разработанного проекта.

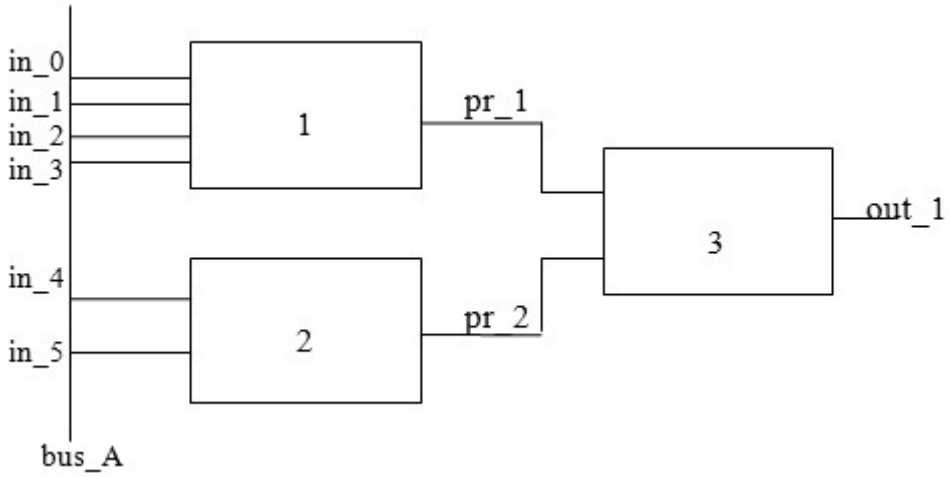


Рисунок х – Структурная схема цифрового устройства

Блок 1 должен быть выполнен на Verilog- языке, а блоки 2-3 описаны схемно. Логика функционирования каждого из блоков представлена в таблице 3.1.

Таблица 3.1 – Описание блоков для различных вариантов

Номер варианта	Блок 1	Блок 2	Блок 3
1	(in_1 И in_2) И (in_3 И in_4)	in_4 И in_5	pr_1 И pr_2
2	(in_1 ИЛИ in_2) И (in_3 И in_4)	in_4 ИЛИ	(НЕ pr_1) И

		in_5	pr_2
3	(in_1 И in_2) И (in_3 ИЛИ in_4)	НЕ (in_4 ИЛИ in_5)	(НЕ pr_1) И (НЕ pr_2)
4	(in_1 ИЛИ in_2) И (in_3 ИЛИ in_4)	(НЕ in_4) ИЛИ in_5	pr_1 И (НЕ pr_2)
5	(in_1 ИЛИ in_2) ИЛИ (in_3 ИЛИ in_4)	in_4 ИЛИ НЕ(in_5)	pr_1 ИЛИ (НЕ pr_2)
6	(in_1 И in_2) ИЛИ (in_3 И in_4)	НЕ (in_4 И in_5)	(НЕ pr_1) И pr_2
7	(in_1 ИЛИ in_2) ИЛИ (in_3 И in_4)	(НЕ in_4) И in_5	pr_1 ИЛИ pr_2
8	(in_1 И in_2) ИЛИ (in_3 ИЛИ in_4)	in_4 И (НЕ in_5)	НЕ (pr_1 ИЛИ pr_2)
9	(НЕ(in_1 И in_2)) И (in_3 И in_4)	НЕ (in_4 ИЛИ in_5)	(НЕ pr_1) ИЛИ pr_2
10	(in_1 И in_2) И (НЕ(in_3 И in_4))	(НЕ in_4) ИЛИ in_5	(НЕ pr_1) И (НЕ pr_2)
11	НЕ((in_1 И in_2) И (in_3 И in_4))	in_4 ИЛИ НЕ(in_5)	(НЕ pr_1) ИЛИ pr_2
12	((НЕ in_1) ИЛИ in_2) И (in_3 И in_4)	НЕ (in_4 И in_5)	pr_1 ИЛИ (НЕ pr_2)
13	(in_1 И in_2) И ((НЕ in_3) И in_4)	in_4 И in_5	pr_1 И pr_2
14	НЕ((in_1 ИЛИ in_2) И (in_3 И in_4))	in_4 ИЛИ in_5	pr_1 ИЛИ pr_2
15	(in_1 И in_2) И (НЕ(in_3 ИЛИ in_4))	НЕ (in_4 ИЛИ in_5)	pr_1 ИЛИ (НЕ pr_2)
16	(НЕ(in_1 И in_2)) И (НЕ(in_3 И in_4))	(НЕ in_4) ИЛИ in_5	НЕ (pr_1 ИЛИ pr_2)
17	(НЕ(in_1 И in_2)) И (НЕ(in_3 И in_4))	in_4 ИЛИ НЕ(in_5)	(НЕ pr_1) И pr_2
18	(in_1 И (НЕ(in_2))) И (in_3 И in_4)	НЕ (in_4 И in_5)	pr_1 И pr_2
19	(in_1 И (НЕ(in_2))) И (in_3 ИЛИ in_4)	(НЕ in_4) ИЛИ in_5	(НЕ pr_1) И (НЕ pr_2)
20	(in_1 И (НЕ(in_2))) ИЛИ (in_3 И in_4)	in_4 И in_5	pr_1 И (НЕ pr_2)

4 Список использованной литературы

1) Тарасов И.Е., Певцов Е.Ф. Основы проектирования цифровых устройств с использованием языка Verilog: Учебное пособие. / Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный