МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И. РАЗАКОВА

Кафедра «Программное обеспечение компьютерных систем»

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ FPGA

Методические указания к лабораторным работам для студентов, обучающихся по направлению «Программная инженерия»

Бишкек – 2015

«Рассмотрено»
на заседании кафедры
«ПОКС»
Прот. № 13 от 1.06.2015 г.

«Одобрено»
Методическим советом
ФИТ
Прот. № 1 от 14.09.2015 г.

УДК 004.087

Составитель: Каткова С.Н.

Введение в программирование FPGA: Методические рекомендации к выполнению лабораторных работ для студентов направления «Программная инженерия»/ КГТУ им. И. Раззакова; Сост.: Каткова С.Н. / - Б.: ИЦ «Текник», 2015. - 52 с.

Представлены общие понятия дисциплины «Введение в программирование FPGA», теоретические сведения и примеры автоматизированного проектирования цифровых устройств в системе Quartus-II 9.0 Web Edition (фирмы Altera) на базе программируемых логических интегральных схем (ПЛИС) FPGA с использованием языка описания аппаратуры VHDL. Настоящее пособие позволит студентам быстро освоить пакет Quartus II и получить навыки практической работы в современной проектной среде.

Рецензент: к.т.н., доцент Аккозов А. Д.

Оглавление

Введение

Современные цифровые устройства обычно изготавливаются в виде одной интегральной микросхемы. Это может быть заказная микросхема (**ASIC**) или программируемая интегральная микросхема (ПЛИС, англ. programmable logic device, **PLD**). Заказные микросхемы характеризуются высокой стоимостью этапа разработки, поэтому для мелкосерийного и среднесерийного производства лучше подходят ПЛИС микросхемы.

ПЛИС – это электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования. Для программирования используются программаторы и отладочные среды, позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры: Verilog, VHDL, AHDL и др.

Программируемые логические интегральные схемы (ПЛИС) являются одними из самых перспективных элементов цифровой схемотехники. ПЛИС представляет собой кристалл, на котором расположено большое количество простых логических элементов. Изначально эти элементы не соединены между собой. Соединение элементов (превращение разрозненных элементов в электрическую схему) осуществляется с помощью электронных ключей, расположенных в этом же кристалле. Электронные ключи управляются специальной памятью, в ячейки которой заносится код конфигурации цифровой схемы. Таким образом, записав в память ПЛИС определенные коды, можно собрать цифровое устройство любой степени сложности (это зависит от количества элементов на кристалле и параметров ПЛИС). В отличие от микропроцессоров, в ПЛИС можно организовать алгоритмы цифровой обработки на аппаратном (схемном) уровне. При этом быстродействие цифровой обработки резко возрастает. Достоинствами технологии проектирования устройств на основе ПЛИС являются: минимальное время разработки схемы (нужно лишь занести в память ПЛИС конфигурационный код), в отличие от обычных элементов цифровой схемотехники здесь отпадает необходимость в разработке и изготовлении сложных печатных плат, быстрое преобразование одной конфигурации цифровой схемы в другую (замена кода конфигурации схемы в памяти), для создания устройств на основе ПЛИС не требуется сложное технологическое производство. ПЛИС конфигурируется с помощью персонального компьютера на столе разработчика. Потому иногда эту технологию называют «фабрикой на столе».

ПЛИС выпускается по двум основным технологиям:

CPLD — набор универсальных цифровых блоков;

CPLD (англ. complex programmable logic device — сложные программируемые логические устройства) содержат относительно крупные программируемые логические блоки — макроячейки, соединённые с внешними выводами внутренними шинами. Может применяться для расширения числа входов/выходов рядом с большими кристаллами, или для обработки сигналов в контроллере COM-порта.

FPGA (Field Programmable Gate Array) — это микросхема, программируемая пользователем. FPGA обычно используются для обработки сигналов, имеют больше логических элементов и более гибкую архитектуру, чем CPLD.

Она состоит из ячеек, часть из которых отвечает за реализацию элементарных функций (Configurable Logic Blocks CLB), а часть – за внутренние соединения (Programmable Switch Matrices PSM).

ПЛИС на FPGA - архитектуре основаны на ОЗУ и могут быть перепрограммированы бесконечное число раз. Но из-за этого у таких ПЛИС появляется существенный недостаток: при отключении питания программа теряется и для предотвращения ее потери нужен специальный конфигуратор. У **ПЛИС с CPLD-структурой** отсутствует проблема потери программы при отключении питания, т.к. они основаны на флэш-памяти.. Но вместе с тем у таких ПЛИС появляется другой недостаток: микросхема может быть перепрограммируема ограниченное число раз.

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Работа с инструментами среды проектирования Quartus II 9.0 на примере вентилей and, or, not и их таблиц истинности.

Цель работы: научиться на практике:

создавать графические схемы вентилей and, or, not с помощью графического редактора схем; компилировать проект; моделировать работу этих элементов и комбинационных схем с помощью программы Simulator; разрабатывать для них программы на языке описания аппаратуры VHDL с помощью текстового редактора; подключать входы и выходы схемы вентилей к ножкам реальной микросхемы на плате DEO; размещать вентили на кристалле FPGA и проводить их трассировку.

Теоретические сведения:

Процедура проектирования в Quartus II

Процедура проектирования устройств на ПЛИС включает в себя следующие этапы.

1. Ввод проекта.

На этом этапе разработчик вводит описание проекта и его частей. Проект или его части могут быть описаны традиционным способом в виде схемы, содержащей отдельные элементы, соединенные между собой цепями связи. Для создания и последующего редактирования таких описаний в пакете Quartus II используется графический редактор. Объектом его работы являются файлы с расширением **.bdf.**

Главным достоинством графического способа ввода проекта является его традиционность и наглядность, связанная с привычностью разработчиков к восприятию изображений схем [1].

В настоящее время все большую популярность приобретают языки описания аппаратуры (HDL) [2,3]. Они допускают описание проектируемого устройства с точки зрения, как его поведения, так и структуры. Достоинствами текстового описания проекта являются его компактность и относительная простота автоматизации любых преобразований, включая процесс создания описания, однозначность понимания и возможность переноса проектов в другие САПР.

2. Компиляция проекта.

Компиляция представляет собой процесс преобразования описания проекта (схемы) в его структурную реализацию на выбранном кристалле ПЛИС. Компиляции может подвергаться как весь проект, так и отдельные его фрагменты. В Quartus II компиляция всегда выполняется для модуля верхнего уровня (top level). Поэтому для компиляции отдельного компонента схемы необходимо предварительно объявить его модулем верхнего уровня. Компиляция включает выполнение нескольких этапов.

2.1. Анализ и синтез.

Модуль анализа и синтеза Quartus II выявляет синтаксические ошибки в проекте. Он проверяет логическую завершенность проекта, то есть *возможность объединения файлов описания проекта в единое целое*, и *возможность реализации проекта на выбранном кристалле ПЛИС*. Он также *преобразует конструкции используемого языка VHDL в их аппаратную реализацию на ресурсах кристалла* таких, как функциональные преобразователи (lut), триггеры, логические элементы, блоки встроенной памяти, встроенные умножители. Исходными файлами для выполнения этапа являются файлы с описанием модулей проекта на языках HDL (**.vhd**, **.v**, **.tdf**) и файлы со схемным представлением **.bdf**. На выходе получаются файлы отчета (**.rpt**, **.htm**) и созданная база данных (**.rdb**) [4].

2.2. Функциональное и временное моделирование проекта.

После завершения этапа синтеза проекта может быть выполнена верификация описания проекта. В основе верификации описания проекта лежит моделирование его работы при имитации различных внешних воздействий. Если при моделировании не учитываются задержки распространения сигналов, то такое моделирование называется *функциональным (functional)*. Если при моделировании учитываются задержки распространения сигналов, то такое моделирование называется *временным (timing)* . Выполняется этот этап с помощью программы ***Simulator***.

2.3. Программирование на языке описания аппаратуры VHDL.

Кроме традиционного способа описания схемы создают программу проекта в форме текстового описания алгоритмов функционирования его модулей в сочетании с текстовым описанием межмодульных соединений для сложных проектов. Для создания и последующего редактирования текстовых описаний частей проекта в Quartus II используется текстовый редактор. Допустимыми являются языки VHDL, Verilog, AHDL (Altera HDL), System Verilog [5-7]. Соответствующие текстовые файлы имеют расширения **.vhd**, **.v**, **.tdf**, **.sv**.

Что надо знать в первую очередь о языке описания аппаратуры? Первая и главная особенность состоит в том, что такой язык описывает не программу действий, а схему, которую можно реализовать в железе. Логические сигналы в логических схемах обрабатываются и передаются параллельно.

2.4. Размещение (имплементация) и трассировка (прошивка). Модуль компилятора Quartus II , реализующий этот этап проектирования, называется ***Fitter***. Он использует базу данных, созданную на предыдущем этапе модулем анализа и синтеза компилятора. Модуль ***Fitter*** осуществляет монтаж проекта в структуру выбранного кристалла программируемой логики. То есть, полученная на этапе синтеза модель полного представления проекта в техническом базисе кристалла ***отображается*** на внутренние ресурсы ПЛИС, которыми являются конфигурируемые логические блоки, блоки встроенной памяти, встроенные умножители и устанавливаются соответствующие соединения с помощью ресурсов ***трассировки*** кристалла. ***Модуль размещения и трассировки*** подбирает для каждой логической функции подходящее место на кристалле, с точки зрения уменьшения времени распространения сигнала, выполняет соответствующие соединения и назначения контактов ввода-вывода. Исходными файлами для выполнения этапа являются файлы с расширением **.cdb**, созданные после выполнения синтеза компилятором и файлы с установками, имеющие расширение **.qsf**. На выходе получаются файлы отчета о компиляции (**.rpt**, **.htm**) и обновленная база данных [4].

Порядок выполнения работы:

1. Создание папки проекта на диске компьютера

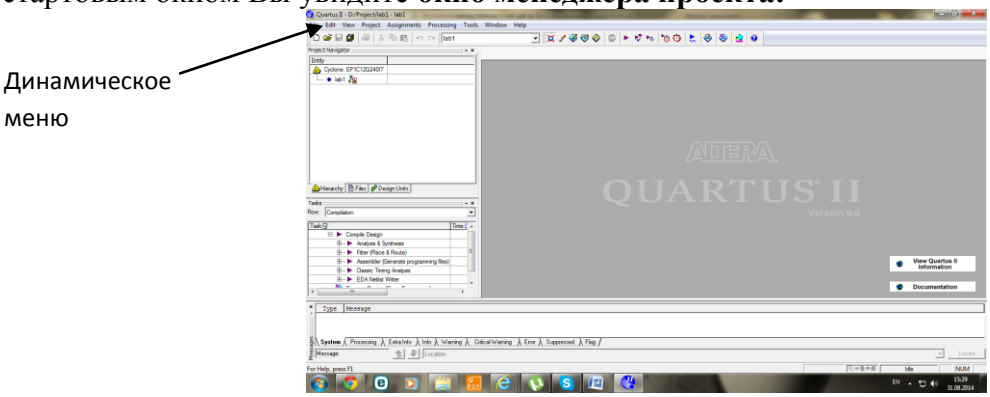
Создайте на диске D папку проекта, дайте папке проекта имя, включающее **Ваше имя** и **номер группы** английскими буквами, например: D/StasPI-1-13. В ней будете сохранять папки проектов.

2. Запуск Quartus-II

Щелкая по ярлыку программы на рабочем столе , запустите программу. Появится *стартовое окно* Quartus (Рис.1.1):

Рис.1.1.

Окно не появится, если стоит галочка напротив строки **Don't show thus screen again** в левом нижнем углу стартового окна. Галочку ставить не будем, просто закроем это окно. За стартовым окном Вы увидите **окно менеджера проекта:**



3. Создание проекта

Выберете в динамическом меню окна менеджера проекта команду создания нового проекта: **File=>New Project Wizard**, появится вводное окно (Рис.1.2.):

Рис.1.2.

- Нажмите **Next,** появится окно (Рис.1.3.).
- Щелкните на кнопе с тремя точками **№1** и введите путь к папке проекта, созданной в пункте 1: **D\ВашеИмя № группы**.
- Щелкните на кнопке **№2** и введите имя проекта **lab1.**
- Имя главного модуля проекта **lab1** в поле **№3** должно появиться автоматически и должно совпадать с именем проекта.

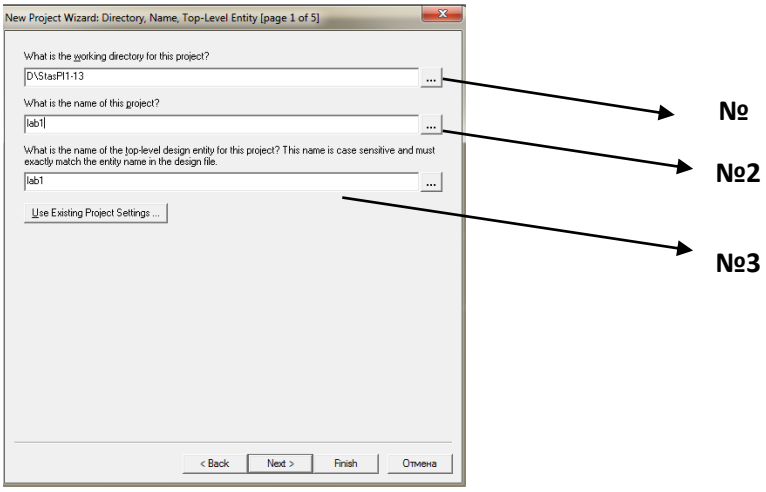


Рис.1.3.

Нажмите **Next.** В появившемся окне (рис. 1.4.) в поле **File Name** нужно ввести имя файла, которое следует импортировать в проект, но так как файлов у нас пока нет, заполнять его нет смысла.

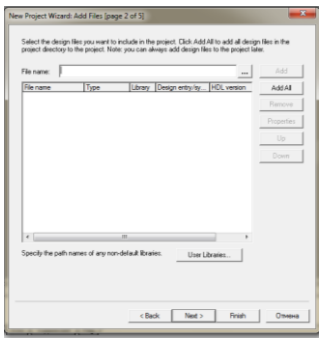


Рис.1.4.

Нажмите **Next**, в появившемся окне (Рис.1.5.) выберите в поле **Family** (семейство устройств) **Cyclone III** и тип устройства **EP3C16F484I7**, который используется в учебном стенде DEO (Terasic). На этом устройстве Вы будете размещать созданную Вами программу и тестировать её работу. В полях **Show in Available device list** оставим значение **any**, т.е. перечень устройств оставляем прежним, без изменения.

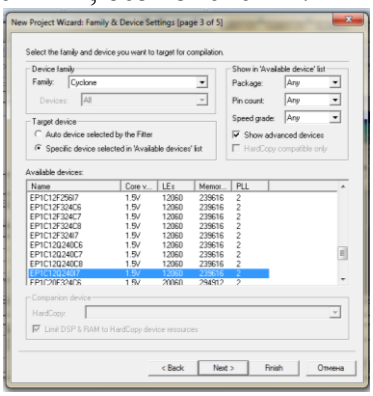


Рис.1.5.

Нажмите **Next,** появившееся окно (Рис.1.6.) пропускаем, оно пока не нужно.

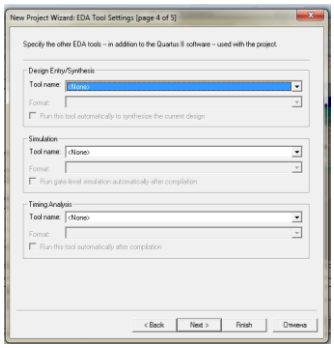


Рис.1.6.

Нажимаем **Next.** В окне (Рис. 1.7.) нажимаем **Finish.** В этом окне есть информация об имени Вашего проекта и имени файла верхнего уровня проекта, семействе и типе устройств, на которой Вы будете размещать созданные вами программы устройств.

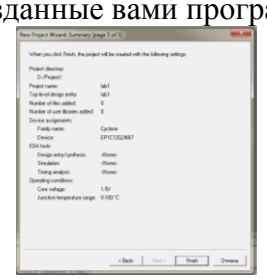


Рис. 1.7.

Таким образом, мы **создали пустой проект**. **Поздравляю!**

После 2-3х секундного ожидания Вы получите **окно менеджера проекта** Quartus II (Рис. 1.8 , стр. 12.):

Описание окна Менеджера проекта:

В левой части окна **менеджера проекта** находятся:

-навигатор проекта **Project Navigator** , который включает три страницы:

- **иерархическая структура проекта** (**Hierarchy**). Страница **Hierarchy** включает название целевого кристалла FPGA фирмы Altera, на котором будет реализован проект, а также отображает иерархическую структуру проекта. В ней отражен корневой модуль проекта (модуль верхнего уровня) и входящие в его состав модули. Навигатор может быть использован для выполнения установок для всего проекта и индивидуальных установок для каждого из модулей проекта.

- **используемые файлы** (**Files**). Страница **Files** навигатора проекта отображает логические файлы проекта, файлы с программами и выполняемые файлы проекта. Для выделения нужного файла используется левая клавиша мыши. Для вызова контекстно-зависимого меню используется правая кнопка мыши. Дважды щелкнув левой клавишей мыши по имени файла, можно открыть его в главном окне менеджера проекта Quartus II. Контекстно-зависимое меню изображено на рис 1.8.

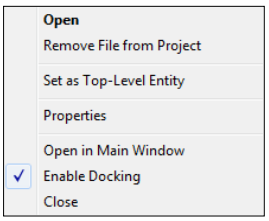


Рис.1.8.

Используя его можно открыть файл в главном окне Quartus II, удалить файл из проекта, объявить выделенный файл модулем верхнего уровня, а также выполнить ряд других действий, включая просмотр свойств выделенного файла.

- входящие в проект модули (**Design Units**). Страница **Design Units** навигатора отображает все компоненты проекта, использованный способ описания проекта, язык описания и файл с описанием компонента (Рис. 1.10.).

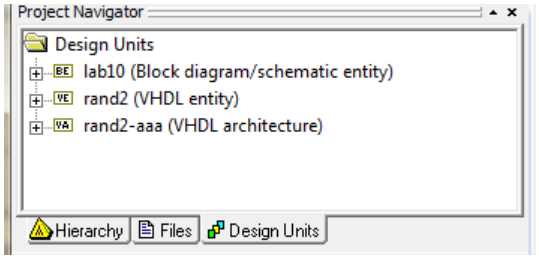


Рис.1.10

Для управления окном навигатора проекта используется пиктограмма  на панели инструментов менеджера проекта. Осуществляя щелчок мышью по этой пиктограмме, можно удалить или возвратить окно навигатора на экран дисплея.

- Под окном навигатора проекта расположено окно задач **Tasks**, содержащее список выполняемых процессов для выбранного этапа проектирования.

- В нижней части графического интерфейса Quartus II содержится окно сообщений ***Messages***, в котором выводится информация о выполненных шагах этапа проектирования, предупреждения и сообщения об ошибках.

4. Реализация проекта при помощи графического редактора схем

Ввод описания проекта включает создание графического файла схемы элемента или устройства. Создадим схему логического элемента &.

С помощью последовательности команд **File=>New** из строки меню окна менеджера проекта, появится окно **New** (Рис.1.11):

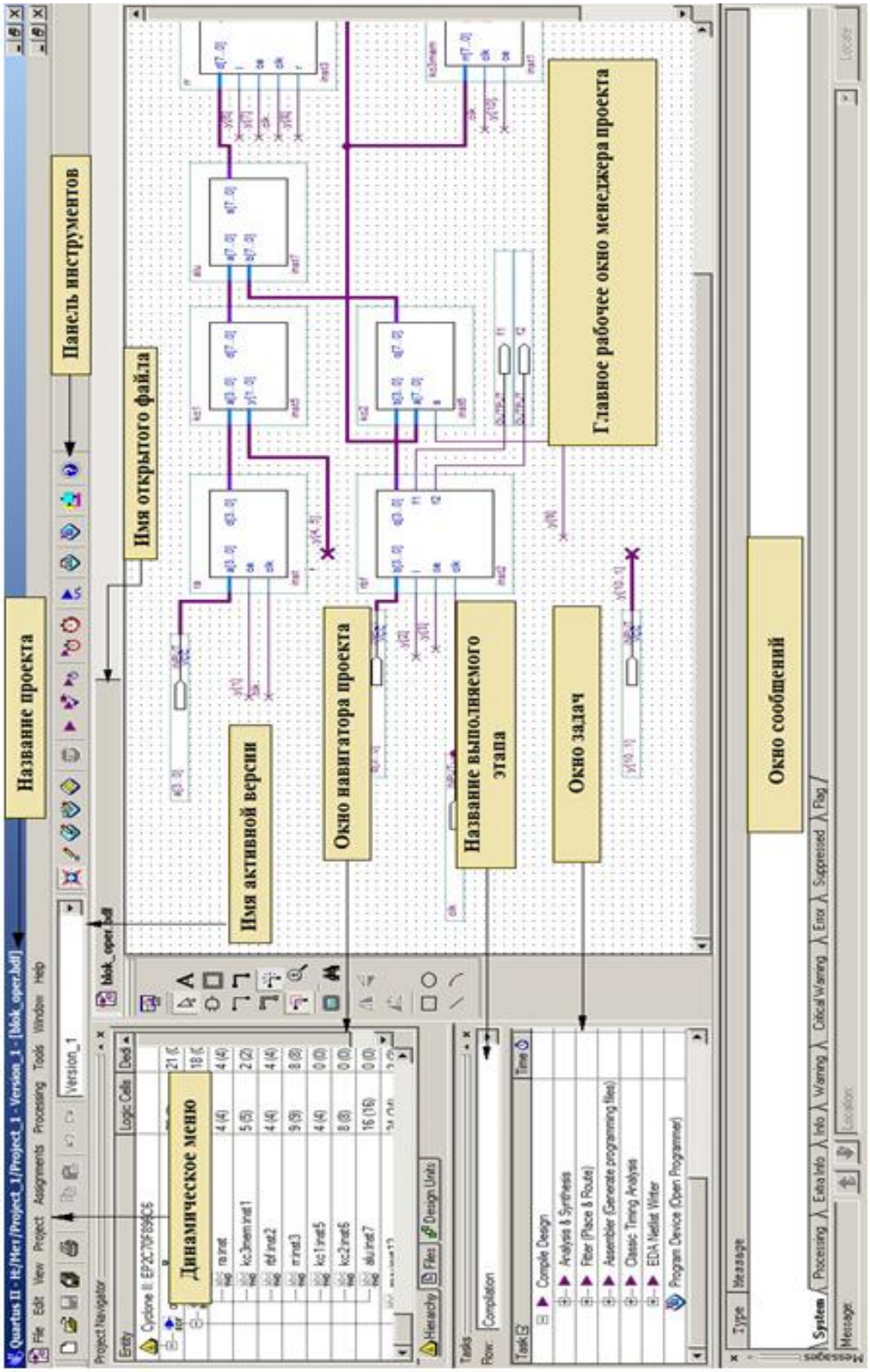


Рис 1 9

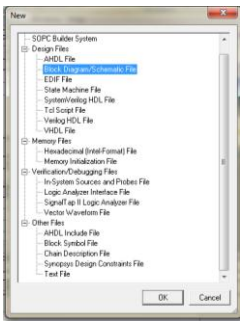


Рис.1.11

В окне **New** выбрать **Block Diagram/Schematic File**, появится окно редактора схем (Рис.1.12.) и в нем рабочее поле для размещения и редактирования схем:

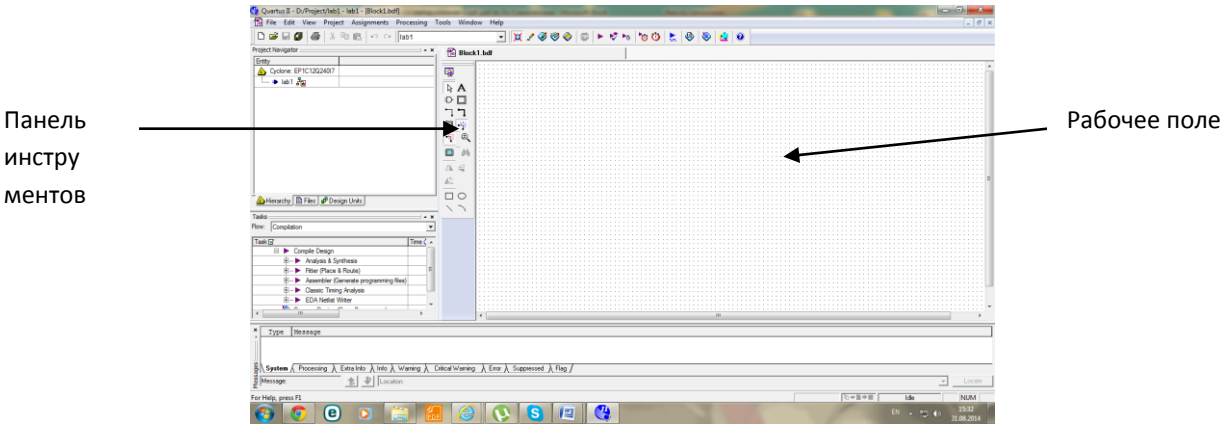


Рис.1.12.

Сохраните пустой файл проекта под именем **lab1.** Для этого откройте меню **Файл** окна менеджера проекта (Рис. 1.13).

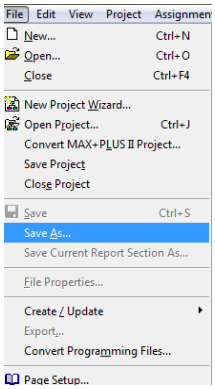


Рис. 1.13

В нем выберите **Save As**…. Появится окно **Сохранить как** (Рис.1.14.).

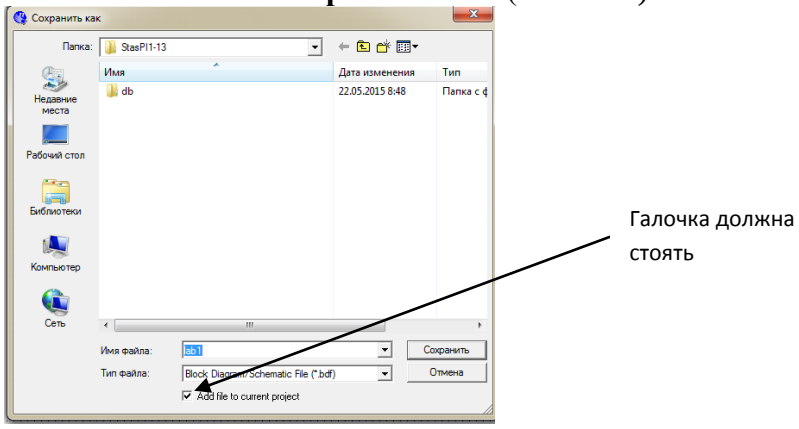


Рис.1.14

В поле **Имя файла** уже есть имя lab1. Щелкнете по кнопке **Сохранить**.

Откройте папку своего проекта на диске D, в ней находится сохраненный файл lab1 (Рис. 1.15.):

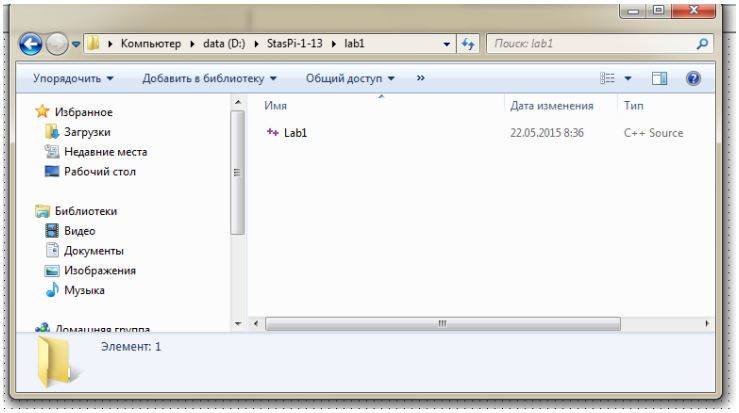


Рис. 1.15

На странице **Files** (1) навигатора проекта (2)появиться этот файл **lab1** (3) с расширением **.bdf** (Рис 1.16.):

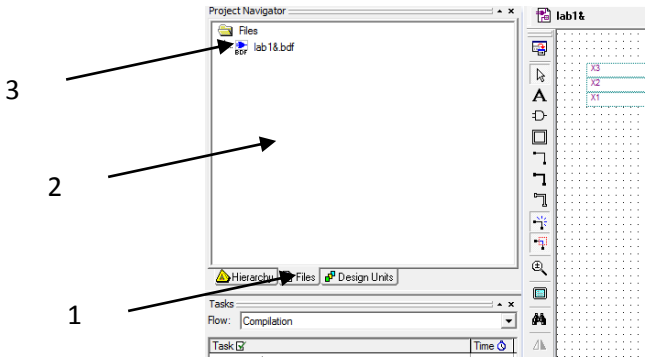


Рис.1.16.

Вы должны научиться создавать схему базового логического элемента **И (&).** Для этого выполните следующие пункты.

а) Поставьте этот элемент на рабочее поле, для этого щелкнете на панели инструментов (Рис. 1.12.), расположенной слева от рабочего поля, на кнопку **Simbol Tool** или просто щелкнем 2 раза по любому месту на рабочем поле, появится окно **Simbol (**Рис. 1.17.):

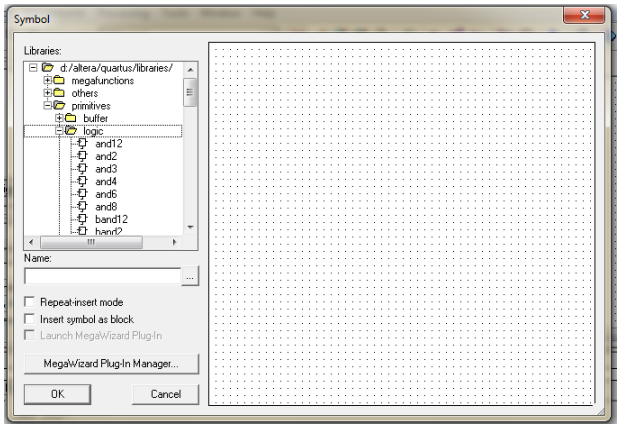


Рис. 1.17.

В окне **Simbol** в поле **Библиотеки (Libraries)** щелчком по плюсу откройте папку **primitives**, в ней подпапку **logic** и щелкните по **and2**, тем самым Вы выбрали логический элемент **&** на два входа. Появляется графическое изображение этого элемента на рабочем поле проекта (Рис. 1.18.):

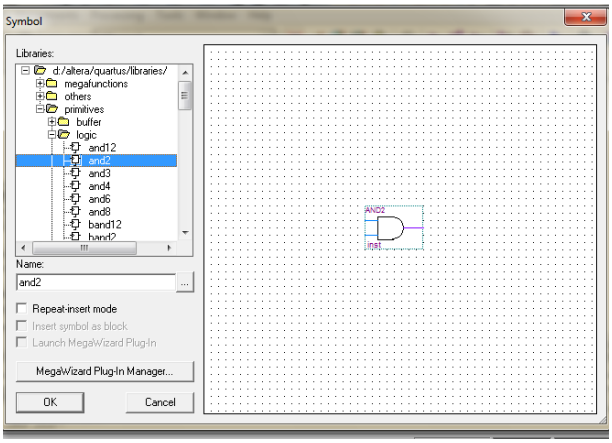


Рис. 1.18

Нажмите на **ОК** и поставьте этот элемент щелчком мышки на рабочее поле (Рис.1.19).

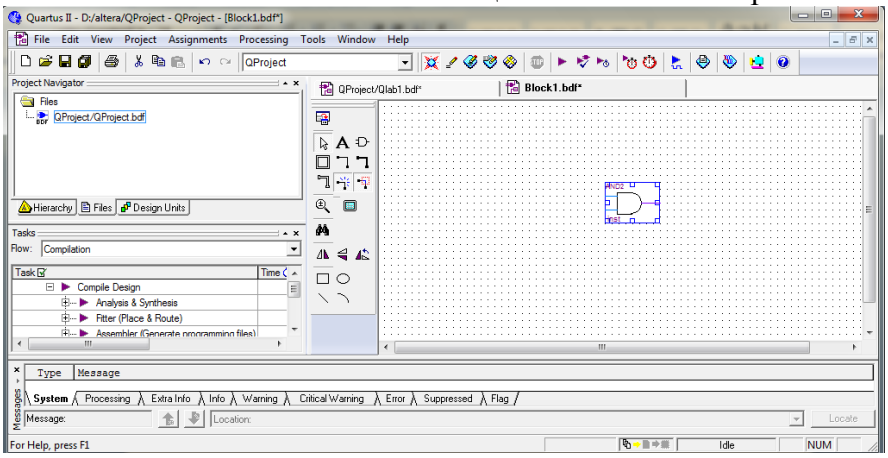


Рис.1.19

б) Поставьте на рабочее поле входы и выход для элемента AND2, для этого делаем 2 щелчка на рабочем поле, в окне **Simbol** выберите **primitives=>pin=>input, ОК**, подсоедините его на один из входов элемента, перемещая его левой кнопкой мыши по полю. Скопируйте его (ctrl C, ctrl V) и присоедините копию ко второму входу элемента AND2. Затем найдите в библиотеке выход **primitives=>pin=>output, ОК** и подсоединяем его к выходу элемента, в результате получите схему элемента AND2 (Рис.1.20):

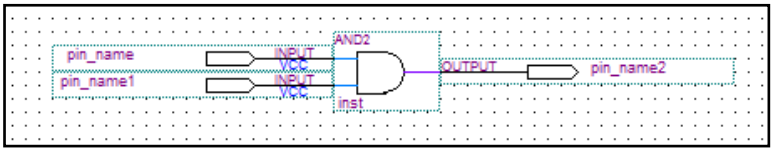


Рис.1.20

Переименуйте названия входов и выхода. Для этого щелкните 2 раза по изображению верхнего входа и введите имя **BUTTON0**. Можно это сделать другим способом: щелкнуть правой кнопкой мыши по входу и выбрать команду **Свойства**, там, в поле **имя** ввести новое имя. Нижнему входу дайте имя **BUTTON1**(Рис.1.48.), а выходу – **LEDG0**. (Рис.1.49.). Первая Ваша схема готова!

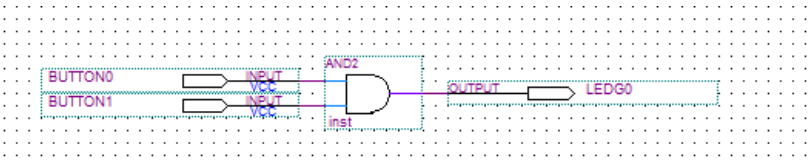


Рис.1.21

Сохраните файл с помощью команд динамического меню в менеджере проекта **File=>Save**. Схема сохранится под тем же именем **lab1.bdf**.

5. Анализ и синтез схемы.

Чтобы проверить, не сделали ли Вы ошибок при построении схемы, щелкните на панели инструментов по значку **Start Analisis & Sintesis**. Нас пока не интересует полная компиляция, которая используется при проверке программ. Если синтез прошел успешно появиться итоговое окно синтеза (Рис. 1.22.).

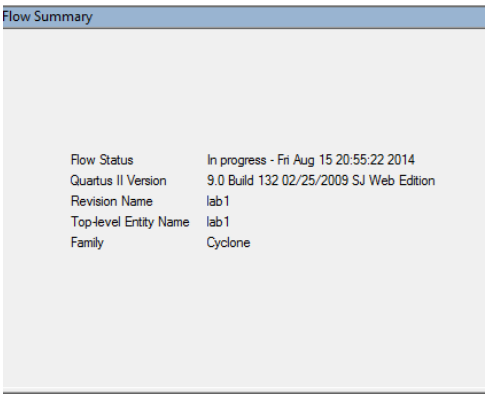


Рис. 1.22

И сообщение об успешном завершении процесса анализа и синтеза (Рис. 1.23).

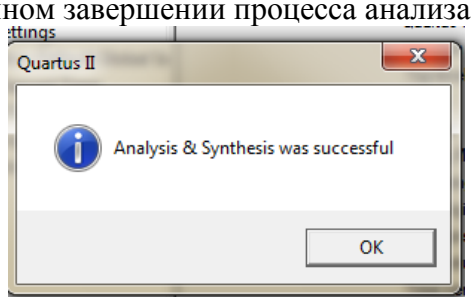


Рис. 1.23

6. Моделирование работы логического элемента & (И) с помощью модуля Simulator Tool

Проводим исследования работы этого элемента на соответствие таблице истинности с помощью программы **Simulator**, вмонтированной в основную программу Quartus. Для этого выполним команду **Processing=> Simulator Tool** окна менеджера проекта (Рис.1.24), появляется окно **Simulator Tool:**

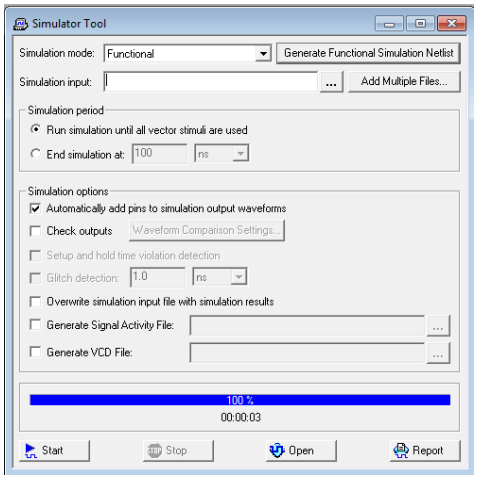
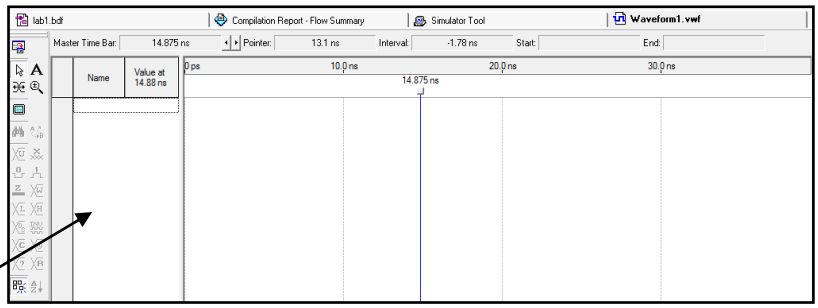


Рис.1.24

В поле **Simulation Mode** выберите режим **Functional** для отображения выходных сигналов без временных задержек по отношению к входным сигналам. Работу с этим окном продолжите позже. Сначала создайте файл для диаграммы. Для этого выберите **File=> New=>Vector Waveform file=>OK**, открылось окно симулятора (Рис.1.25).



Левое поле

Рис.1.25

Сделать 2 щелчка на левом поле симулятора (рис. 1.25), появилось окно **Insert Node or Bus** (рис.1.26):

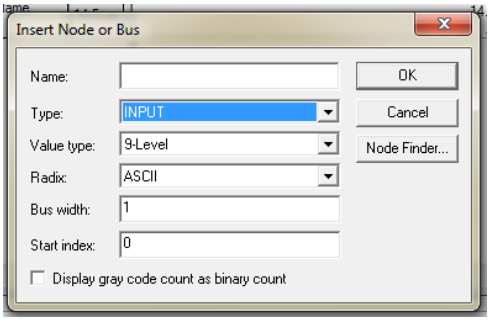


Рис.1.26

Выберите в поле **Type** input**, а затем Node finder, появляется окно** (Рис. 1.27), в котором выберите **Pins all=>list**, в левом окне появляется весь список входов и выходов, а затем с помощью кнопки >> перенесите весь список в правое поле (возможен ручной ввод названия вводов и вывода в поле Name) (рис.1.26):

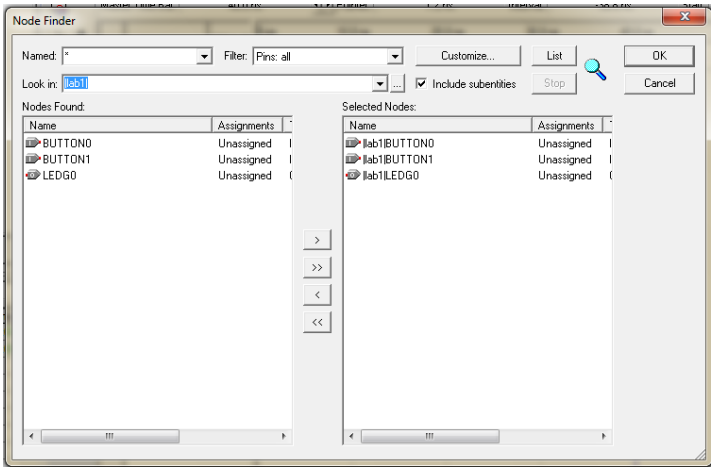


Рис. 1.27

Щелкните OK,OK. Сохраните файл File => Save As под именем lab1.vwf (Рис 1. 28):

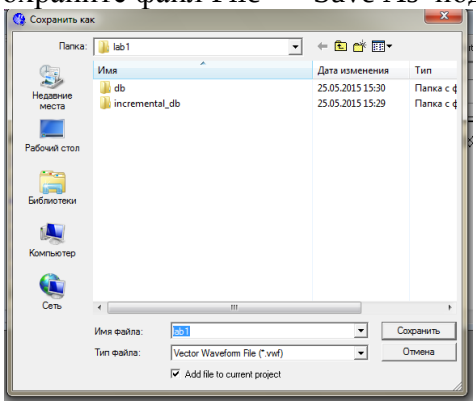


Рис 1.28

Имя файла **lab1.vwf** должно выйти автоматически в поле **Simulator input** окна **Simulator Tool**. Если этого не произошло сами, вручную впишите имя этого файла (Рис.1.29).

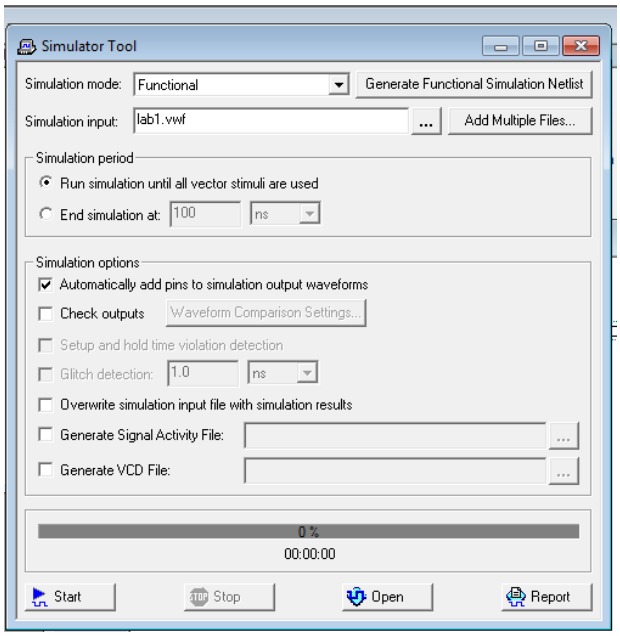


Рис.1.29

Нажмите на кнопку генерации **Generate Functional Simulation**. Выходит сообщение об успешном окончании генерации работы программы симулятора (Рис.1.30).

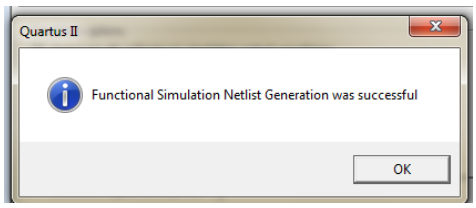


Рис.1.30

А затем на кнопку **Start** для запуска формирования отчета по симуляции. Тоже выходит сообщение об окончании этого этапа (Рис.1.31).

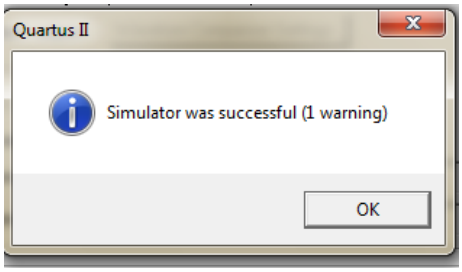


Рис.1.31

Нажмите на кнопку **Report,** появилось окно (Рис. 1.32). В этом окне видим результаты симуляции в виде временной диаграммы. На входы поданы нули, т.е. BUTTON0=0 и BUTTON1=0, на LEDG0 тоже - 0 и это верно в соответствии с таблицей истинности для элемента &.

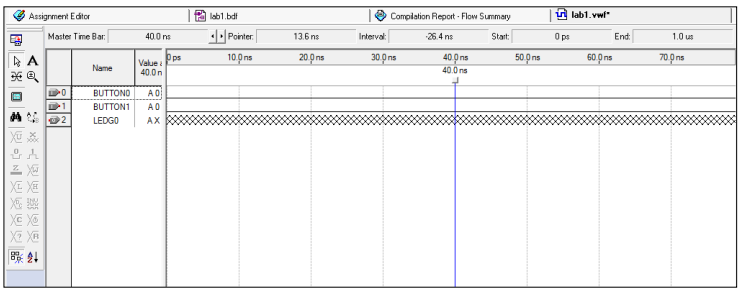


Рис.1.32

Подавайте разные значения на входы схемы, и анализируйте правильность работы логического элемента в соответствии с таблицей истинности, т.е. анализируйте значения входов BUTTON0, BUTTON1и соответствующие им значения выхода LEDG0 после симуляции.

Для этого установите BUTTON0в единицу на временном интервале от 20 до 30 ns. Для этого левой мышью выделите участок на линии сигнала BUTTON0 и нажмите на появившейся вертикальной панели инструментов кнопку **Forcing High(1)** (Рис. 1.33).

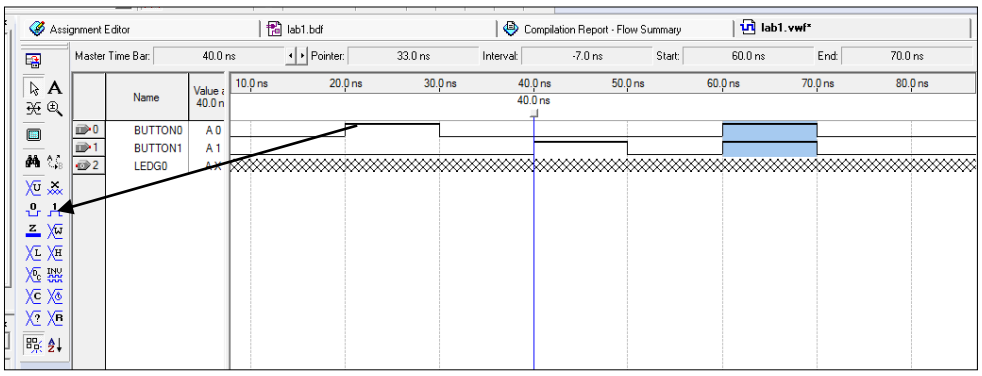


Рис. 1.33

Выделите участок на линии BUTTON1 на интервале 40-50 ns, нажмите **Forcing High (1),** далее установите вход BUTTON1 и вход BUTTON0в состояние единицы одновременно на интервале 60-70 наносекунд.

Запустите на выполнение программу Simulator (Симулятор), для этого нам нужно сделать только **Start и Report** (генерацию мы уже делали).

Итак, откройте вкладку **Simulator Tool**, нажмите в окне симулятора кнопку **Start**, по окончании процесса симуляции выходит сообщение об её успешном завершении (Рис.1.34).

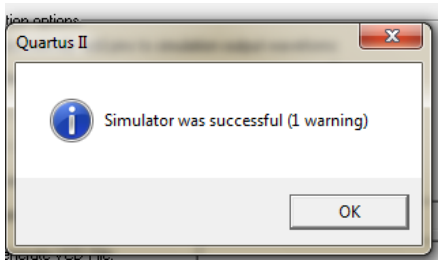


Рис. 1.34

Warning – это замечание, которое не считается ошибкой.

При нажатии кнопки **Report** Вы получите диаграмму работы Симулятора для логического элемента **& (И) (**Рис. 1.35):

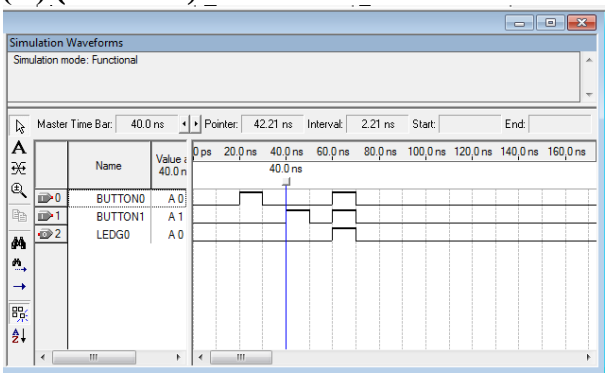


Рис. 1.35

Если надо изменить ширину столбцов, нажмите на **Ctrl** и **покрутите колесико**.

Проанализируем работу элемента на основе диаграммы:

На участке от 0 до 10 ns: LEDG0=BUTTON0 & BUTTON1 =0&0=0;

На участке от 10 до 30 ns: LEDG0= BUTTON0 & BUTTON1 =1&0=0;

На участке от 30 до 50 ns: LEDG0= BUTTON0 & BUTTON1 =0&1=0;

На участке от 50 до 60 ns: LEDG0= BUTTON0 & BUTTON1 =0&0=0;

На участке от 60 до 80 ns: LEDG0= BUTTON0 & BUTTON1 =1&1=1;

Вывод: элемент & работает должным образом, т.к. его работа соответствует таблице истинности для логической операции &.***Поздравляю, процесс моделирования работы логического элемента & успешно завершен!***

7. Реализация проекта при помощи языка описания аппаратуры VHDL

Теория: Интерфейс и архитектура объекта (элемента, схемы) проекта

Программа на языке VHDL состоит из *интерфейса* (entity) и *архитектуры* (architecture)**.**

В *интерфейсе* описывается связь между объектом проекта и его окружением. «Внутренность» объекта в entity не описывается, она подобна «черному ящику». Интерфейс включает только имя объекта и описания его портов (входов, выходов). Для сигналов, подаваемых и снимаемых с портов, указывается вид или режим, или направление сигнала: входной (in), выходной (out) и его тип, например: BIT, std_logic.

Архитектура объекта - это тело (architecture) объекта. В теле раскрывается внутренность «черного ящика». В архитектурном теле описываются функции (поведение) и внутренняя структура объекта проекта.

Пример программирования **объекта – элемента &** под именем **rand2.**

```
library ieee;
use ieee.std_logic_1164.all;
-----------------------------------
entity rand2 is                -- декларация имени объекта проекта
port ( BUTTON0, BUTTON1: in BIT;    -- декларация входных портов
LEDG0 : out BIT);                   -- декларация выходного порта
 end rand2;                    --окончание интерфейса
-----------------------------------
architecture functional of rand2 is  -- декларация архитектуры объекта
begin                                          --ключевое слово начала функции в ар
LEDG0<= BUTTON0 and BUTTON1;                  -- описание функции объекта
 end functional;
                               -- окончание архитектуры
```

Оператор <= означает передачу сигнала.

В тексте данной программы имеются комментарии. Комментарий начинается двумя смежными дефисами и продолжается до конца строки.

Для создания программного (конструкторского) файла в главном меню выберите **File→New…**. В появившемся окне **New** (рис. 1.36)

Рис. 1.36

выберите строку **VHDL File.** Это конструкторский файл на универсальном языке описания аппаратуры **VHDL** (*Very High Speed Integrated Circuit Hardware Description Language*). Конструкторский файл устройства – это главный файл проекта, он содержит функционально-логическое описание устройства.

Далее щелкните по кнопке "OK" в нижней части окна. В рабочем поле окна менеджера проекта появится окно созданного файла с именем **VHDL1.hdl** – это файл с текстовым (программным) представлением проекта (Рис. 1.37).

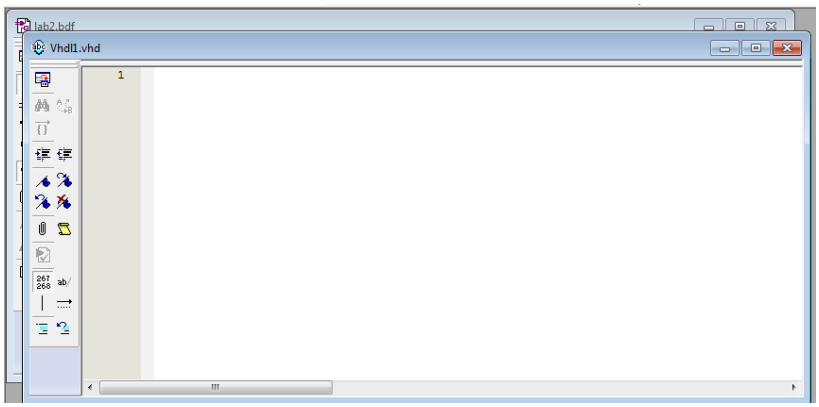


Рис. 1.37

Код программы элемента **&** копируем в текстовый файл VHDL1.hdl (Рис.1.38):

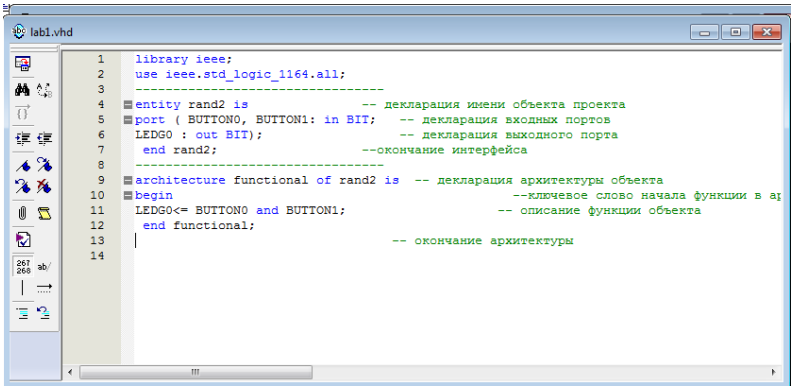


Рис.1.38

Файл **VHDL1.hdl** сохраните с именем, соответствующим имени проекта **lab1**. Для этого необходимо в главном меню выбрать **File→Save As…**. В появившемся диалоговом окне "***Сохранить как***" будет предложено сохранить файл с именем проекта и расширением "**vhd**". Следует принять предложение и щелкнуть по кнопке "***Сохранить***". В результате имя **VHDL1.hdl** файла заменится на имя проекта **lab1** (Рис.1.39).

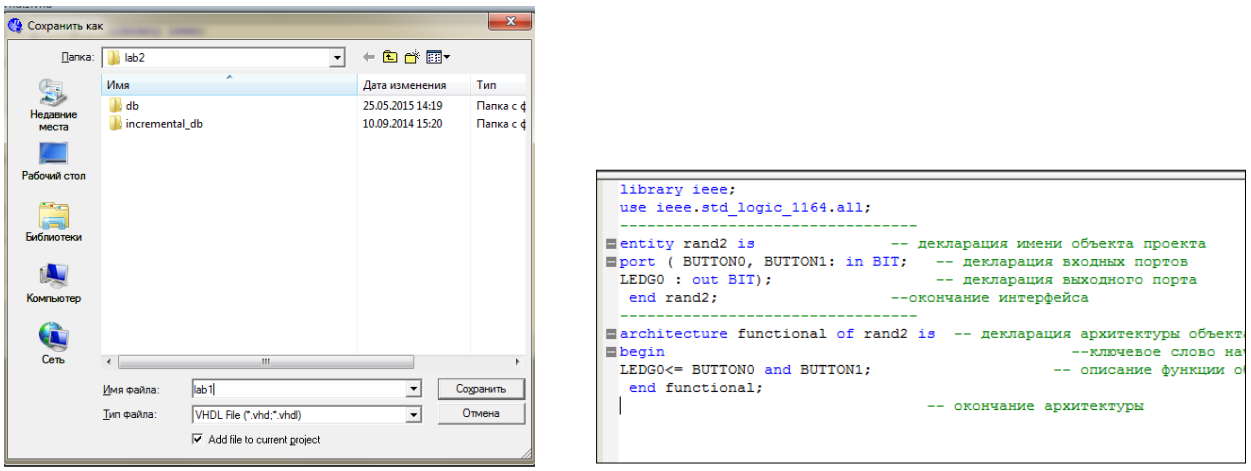


Рис.1.39

Теперь укажите программе, что это главный файл проекта: **Project => Set as Top-Level Entity** (Рис. 1.40):

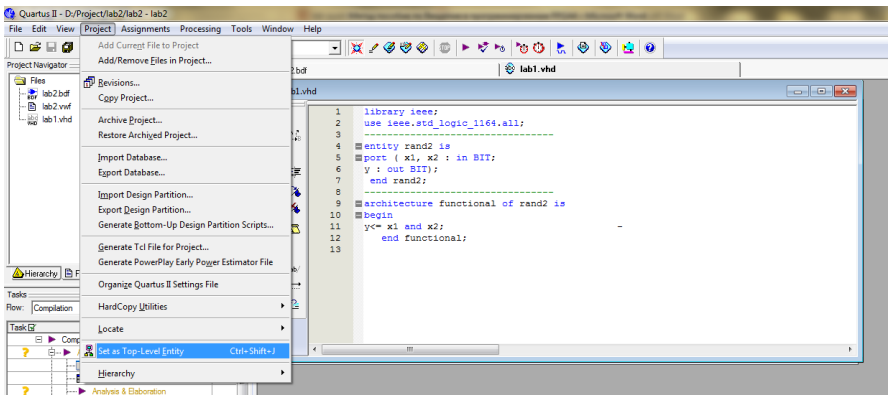


Рис. 1.40

Скомпилируйте проект: **Processing=>Start Compilation (Ctrl+L),** появится итоговое окно компиляции (Рис. 1.41)

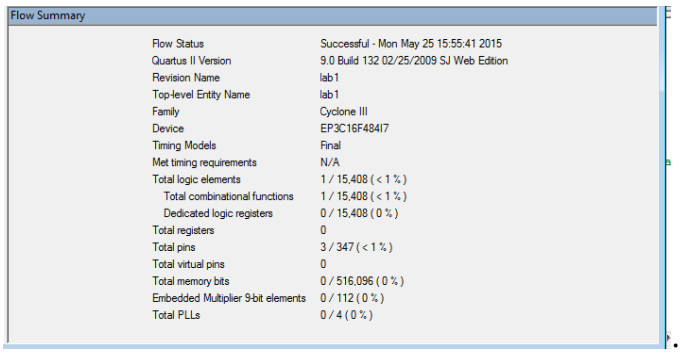


Рис. 1.41

Информация, размещенная в окне **Compilation Report - Flow Summary**, позволяет проверить состояние статуса компиляции, название проекта, выбор микросхемы, число используемых макроячеек и количество задействованных выводов.

8. Технология размещения и трассировки программы на ПЛИС FPGA.

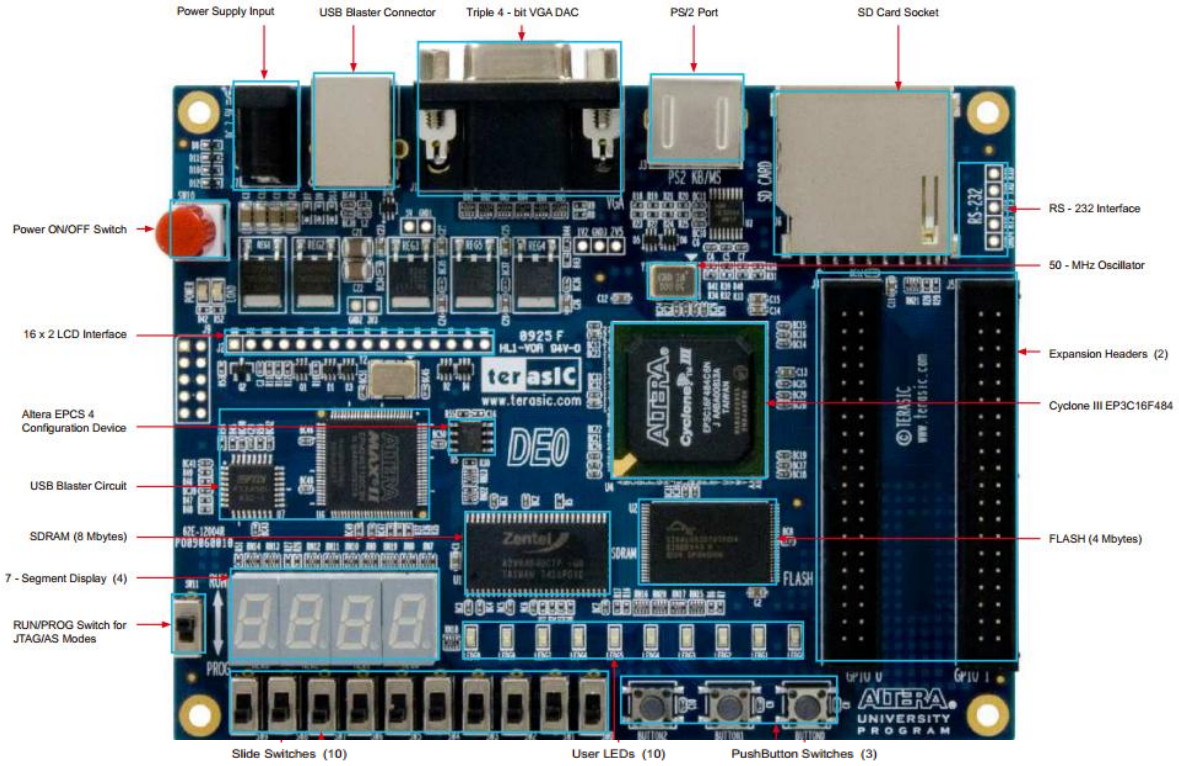


Рис.1.42 Плата DEO

Описание платы

Все лабораторные работы выполняются на плате, основой которого является ПЛИС семейства **Cyclon III EP3C16F484I**. Плата подключается к персональному компьютеру через порт USB. К **выводам** ПЛИС подключены 10 светодиодов (LEDG0-LEDG9), которые могут быть использованы как индикаторы логических уровней в различных точках схемы. Четыре 7-сегментных индикатора (NEXO_D0 - NEXO_D6) предназначены для индикации цифр от 0 до 9 на выводах создаваемой схемы. Элемент Oscillator является генератором тактовых импульсов с частотой 50 МГц. Три элемента BUTTON0, BUTTON1 и BUTTON2 представляют собой одиночные кнопки, каждая их которых на **входах** формирует два логических уровня (0 и 1). 10 переключателей SW0-SW9 также предназначены для формирования логических уровней на **входах** ПЛИС. Выводы конфигурирования ПЛИС подключены к разъему (USB – Blaster Connector) порта USB персонального компьютера.

Подключение платы

a) Соедините USB-порт компьютера с USB-BLASTEROM. Удостоверьтесь, что его драйвер установлен на компьютере. Должно появиться соответствующее сообщение.

b) Соедините 7.5-вольтовый адаптер с отладочной платой DE0 через порт Power Supply Input.

c) Соедините монитор VGA с портом VGA DE0.

d) Run/Prog - положение Run используется для текущей работы.

e) Включите напряжение, нажав переключатель input/output.

Установка DE0_ControlPanel

Плата DE0 идет со средством Control Panel, которое позволяет пользователям получать доступ к различным компонентам управления компьютера. Компьютер общается с платой через USB-соединение. Средство может использоваться, чтобы проверить функциональность компонентов управления или используется в качестве инструмента отладки.

Программное обеспечение DE0_ControlPanel расположено в папке “DE0_Control_panel” на CD-ROMe. Чтобы установить его, просто скопируйте целую папку на компьютер. Чтобы активировать DE0_ControlPanel, выполните следующие шаги:

1. Удостоверьтесь, что Quartus II и драйвер для USB -BLASTER установлены успешно на Вашем PC.

2. Соедините USB-кабель с портом USB -BLASTER, соедините 7.5-вольтовое электроснабжение, и включите выключатель питания.

3. Установите выключатель Run/ПРОГР в положение Run.

4. Начните установку DE0_ControlPanel на компьютер. Появляется интерфейс ControlPanel, показанный в рисунке 1.43.

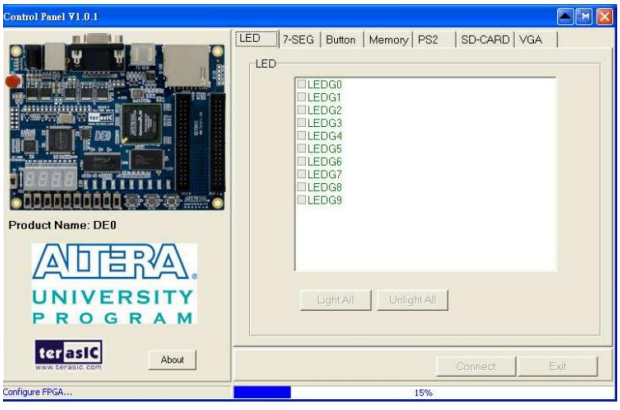


Рис. 1.43

Когда окно появится, оно автоматически загрузит бит файл потока **lab1.sof** в FPGA. *Панель **DE0_ControlPanel** теперь готова для использования*.

Проверка работоспособности светодиодов и кнопок устройства DEO с помощью панели DE0_ControlPanel

В окне Рис. 1.42 установите работоспособность светодиодов: откройте вкладку LED, выберите используемый в Вашей схеме светодиод **LEDG0** или все светодиоды, как на рисунке 1.44.

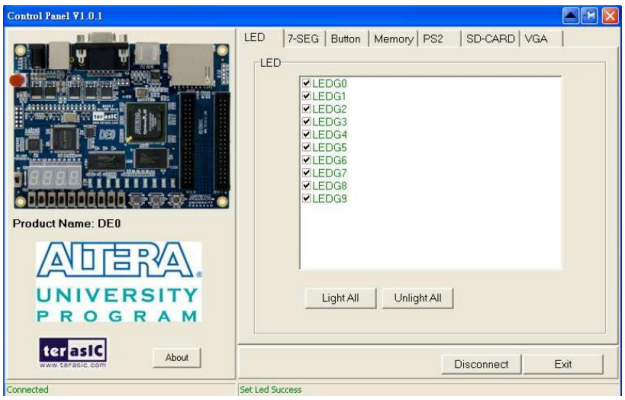


Рис.1.44

Выбрав вкладку **Button**, получим окно (Рис.1.45) для проверки функциональности кнопок и выключателей. Для этого нажмите кнопку **Connection**. Процесс контроля статуса кнопок показывают в окне GUI и обновляют в режиме реального времени. Нажмите **Stop**, чтобы закончить процесс контроля.

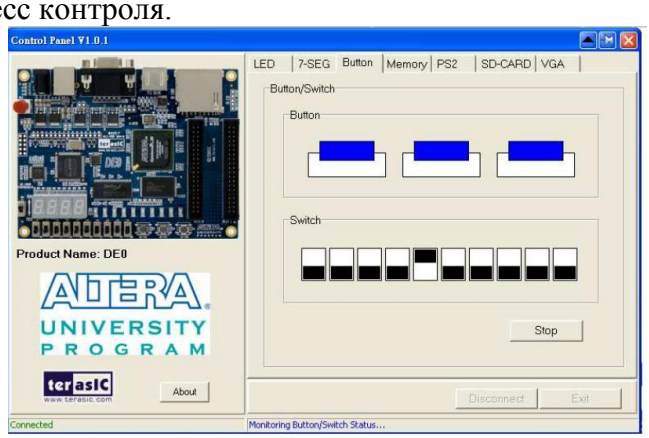


Рис.1.45

Выбрав вкладку 7- SEG, получим окно (Рис.1.46) для проверки функциональности четырех семисегментных индикаторов.

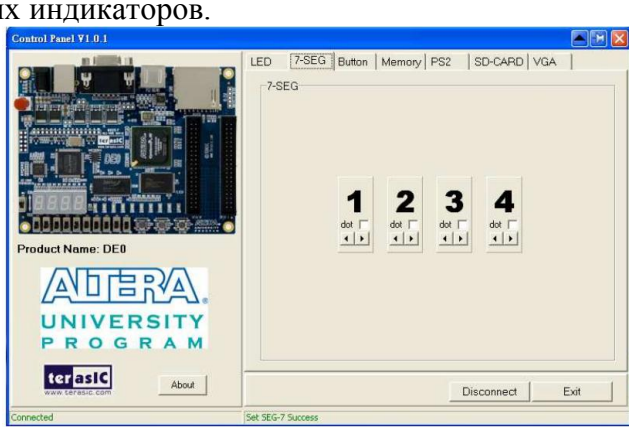


Рис.1.46

Назначение выводов микросхемы - связка Вашей схемы с реальным железом

Необходимые назначения выполним вручную, кликнув **Assigments→Pins**. В результате выведется окно планировщика выводов **Pin Planner** (рис. 1.47), в котором показан внешний вид (схема выводов) микросхемы и обозначен тип используемых выводов. В левой части окна дается перечень групп, а в нижней части – список входных и выходных переменных проекта. При подведении курсора к любому из выводов микросхемы выводится расшифровка его назначения.

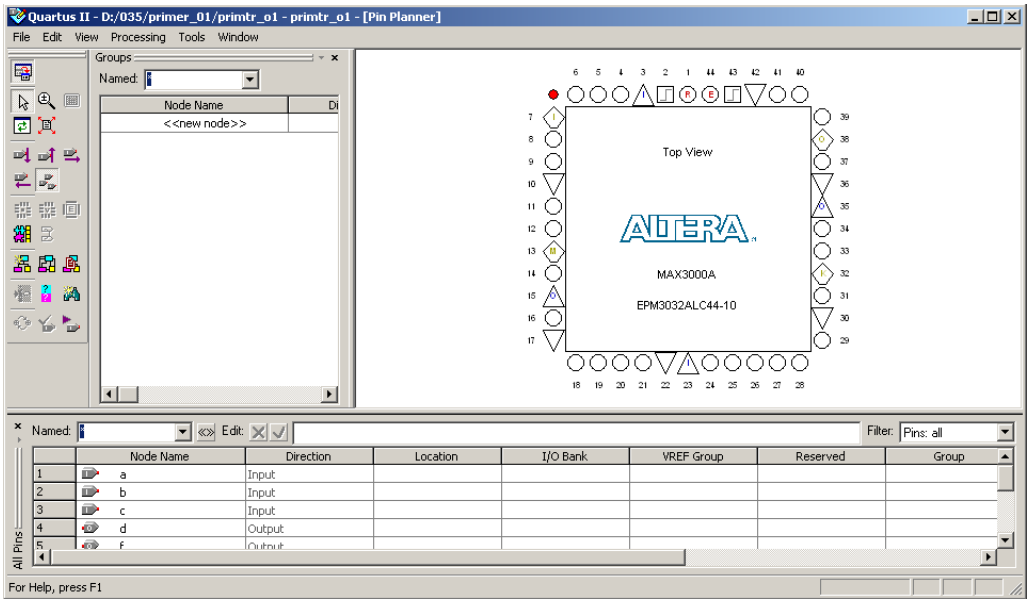


Рис. 1.47

Переменным проекта, показанным в нижней части окна, можно сделать назначение выводов микросхемы. Для этого в поле **Location** после двойного клика открывается меню, из которого можно выбрать номер назначаемого вывода. При переходе в другую ячейку поля **Location** ранее выполненное назначение вывода микросхемы фиксируется и назначенный вывод отмечается на схеме красным цветом. Аналогично можно сделать назначения других переменных проекта. После закрытия окна планировщика выводов все сделанные назначения сохраняются.

Мы сделаем в поле **Location** следующие назначения для входов **BUTTON0** и **BUTTON**1 в соответствии с картой назначений кнопок на плате DEO: **входу BUTTON0 назначим вывод H2, входу BUTTON1 вывод - G3,** (Рис. 1.48).

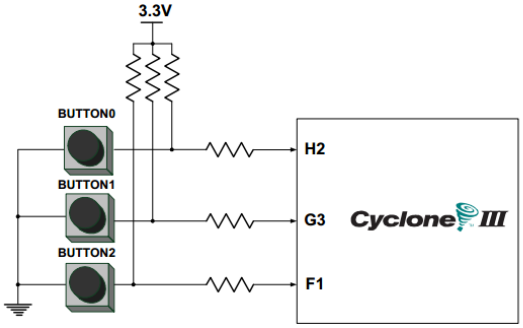


Рис. 1.48

Плата DE0 обеспечивает три кнопочных переключателя. Эти три кнопки под названием BUTTON0, BUTTON1, и BUTTON2 связаны непосредственно с Cyclon III FPGA. Каждый переключатель обеспечивает верхний уровень логический уровень (3.3 В), когда кнопка не нажата, и обеспечивает низкий логический уровень (0 В) когда кнопка нажата.

Для сборки исследуемой схемы необходимо также посмотреть карту подключения соответствующих переключателей и светодиодов к ПЛИС. Выходу **LEDG0** назначим вывод **J1**, соответствующий светодиоду **LEDG0** на основе карты назначений для диодов на плате DEO (Рис.1.49):

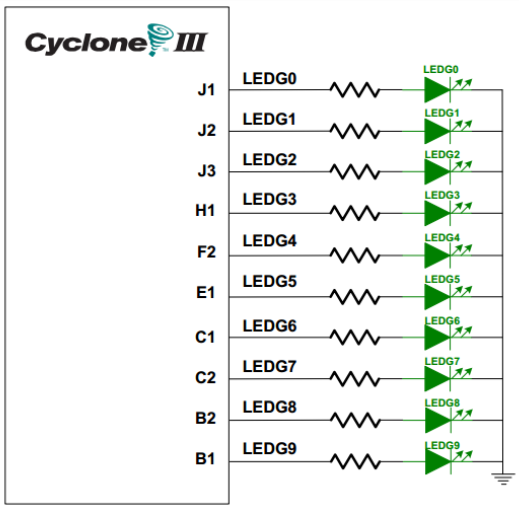


Рис.1.49

На плате DE0 есть 10 управляемых пользователем светодиодов. Каждый светодиод связан непосредственно с Cyclon III FPGA; светодиод горит при высоком логическом уровне (значение – логическая “1”), и гаснет при низком логическом уровне (значение - логический “0”).

В нашем случае две кнопки BUTTON0 и BUTTON1 объединены в логическую функцию И. Поэтому светодиод LEDG0 гаснет (переходит в ноль), если нажать хотя бы на одну кнопку (на входы поданы ноль и единица). И горит, когда не нажаты обе кнопки – на входы поданы две единицы. Эти результаты говорят о том, что работа схемы соответствует таблице истинности логическую функции И, а значит схема работает верно.

Для того чтобы назначения вступили в силу, следует **повторно выполнить компиляцию проекта!!!**

После выполнения компиляции, используя вызов из меню **Compilation Report→Fitter→Resource Section→All Package Pins**, можно посмотреть назначения выводов микросхемы. После компиляции в схеме **lab1.bdf** появились названия выводов схемы ПЛИС, назначенные её входам и выходам (рис.1.5).

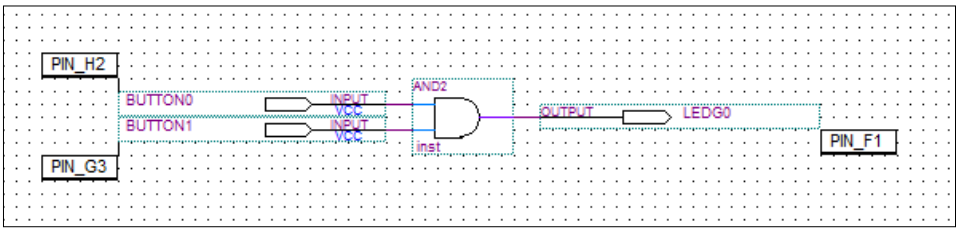


Рис. 1.5

Продолжите подключение схемы к реальной схеме ПЛИС. Для этого выполняем **Assignment** => **Assignment Editor**. И заполняем таблицу как показано ниже. Здесь для всех наших входов и выходов задаем расположение (Location), т.е. показываем на какую ножку микросхемы подключить вход или выход нашей схемы. Это мы уже сделали с помощью **Assignment** => **Pins.** Также для кнопок включаем подтягивающие резисторы (Рис.1.51).

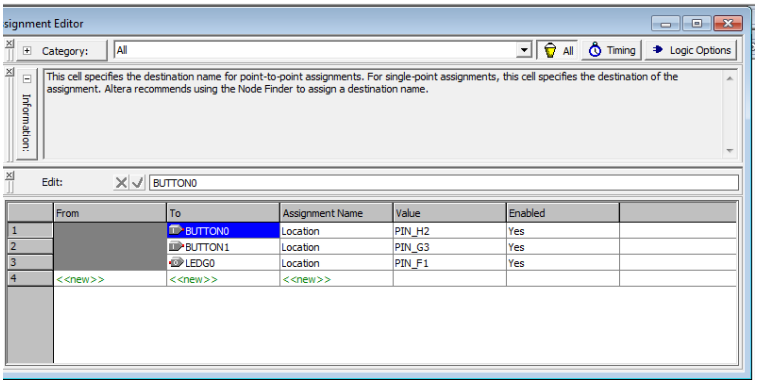


Рис.1.51

Выполните **Assignment => Devices… => Devices and Pin option… => Unused Pins,** укажите **«All input tri-stated»**, этим действием мы подключаем все неиспользуемые выводы микросхемы к нулю по питанию, и чтобы неиспользуемые выводы имели высокое входное сопротивление (Рис.1.52).

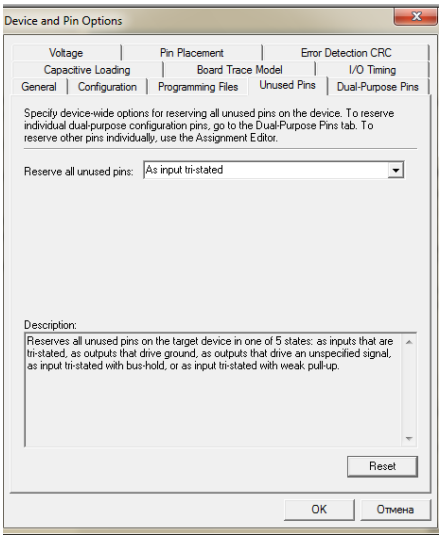


Рис.1.52

Проект готов можно повторить компиляцию!!!

Затем притяните к напряжению питания резистора, чтобы не нажатая кнопка давала на входе микросхемы четкую логическую единицу. Подключите внутреннее напряжение питания ко всем входам схемы пользователя. Для этого назначьте всем входам схемы с именами BUTTON0 и BUTTON1 настройку **Weak Pull Up Resistor**, для этого правой кнопкой щелкните по пустой строке и выберите **Node Finder** (Рис.1.53)

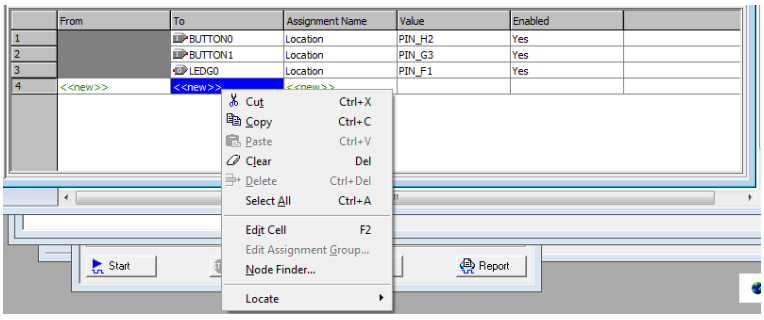


Рис.1.53

и перенесите **BUTTON0** в правое окно, **ОК (**рис. 1.54**).**

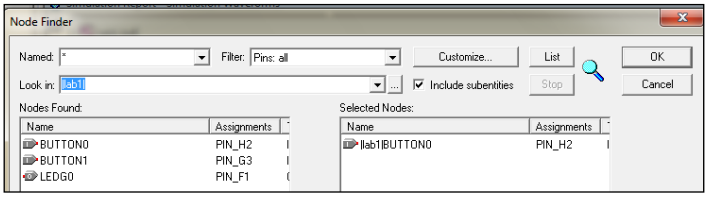


Рис. 1.54

В поле **Assignmen Name** введите **Weak Pull Up Resistor (**рис. 1.55**)**

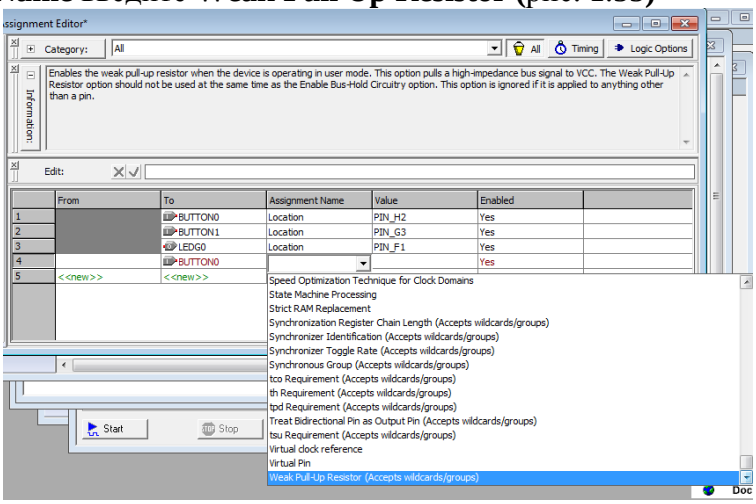


Рис. 1.55

Те же действия проделайте для кнопки BUTTON1. **Компилируйте проект!!!**

Загрузка схемы на плату FPGA и её прошивка

Установку и прошивку осуществим с помощью программатора USB-Blaster.

USB-Blaster – запускаем из **Quartus** утилиту «**Programmer**». Файл **lab1.sof** из папки **D\StasPi-1-13\lab1** и название схемы ПЛИС Вашего проекта автоматически добавлен в окно утилиты программирования кристалла (Рис.1.56).

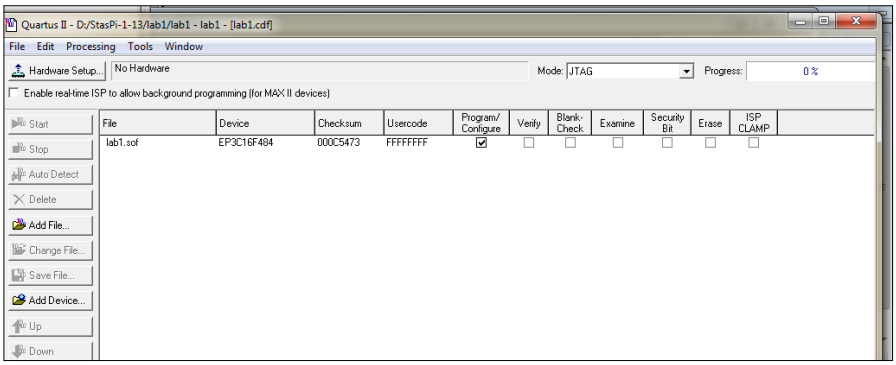
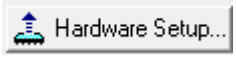


Рис.1.56

Вдоль левой стороны окна расположены инструментальные кнопки. Основные из них:



- установка аппаратных средств программирования

запуск программирования кристалла

- добавление файла в список

- добавление ПЛИС в JTAG цепь

Нажатие первой кнопки из приведенного списка вызовет появление на экране монитора окна **Hardware Setup**, приведенного на рис. 1.57. Первая страница этого окна с закладкой **Hardware Setting** позволит выбрать подходящий программатор из представленного в большом окне списка. В лабораторной плате DE0 используется программатор USB Blaster. Поэтому при заполнении окна для настройки программатора следует указать именно его (Рис.1.57).

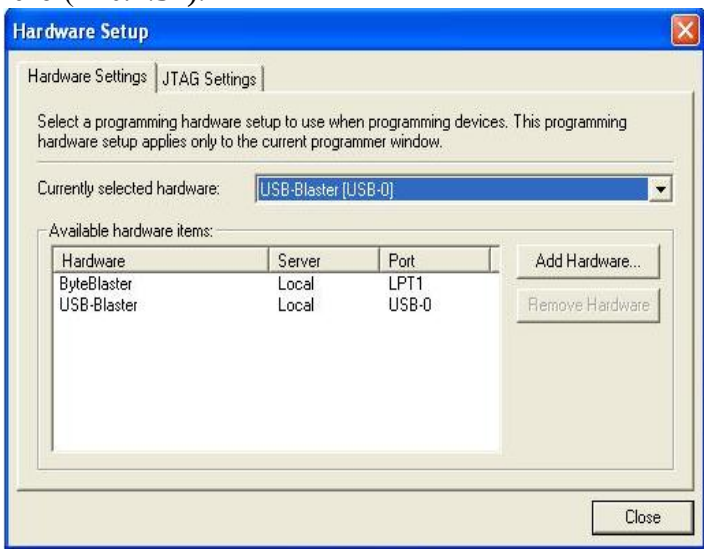


Рис.1.57

Вторая страница, с закладкой **JTAG Settings** позволит выделить **JTAG** серверы, которые следует добавить или удалить из представленного списка. Список мы не изменяем.

Программирование кристалла ПЛИС FPGA

1. Если конфигурационного файла lab1.sof нет в основном поле окна программатора, добавьте файл из Вашей папки, который находится на диске D компьютера, используя кнопку **Add File.**

2. Установите опцию ***Program/Configure***, щелкнув мышью в соответствующем окне в одноименном столбце.

3. Нажмите кнопку **Start**. После этого в окне **Progress** будет визуально отображаться процесс выполнения программирования кристалла. После успешного завершения этого процесса в окне **Progress** установится значение 100% (Рис. 1.58).

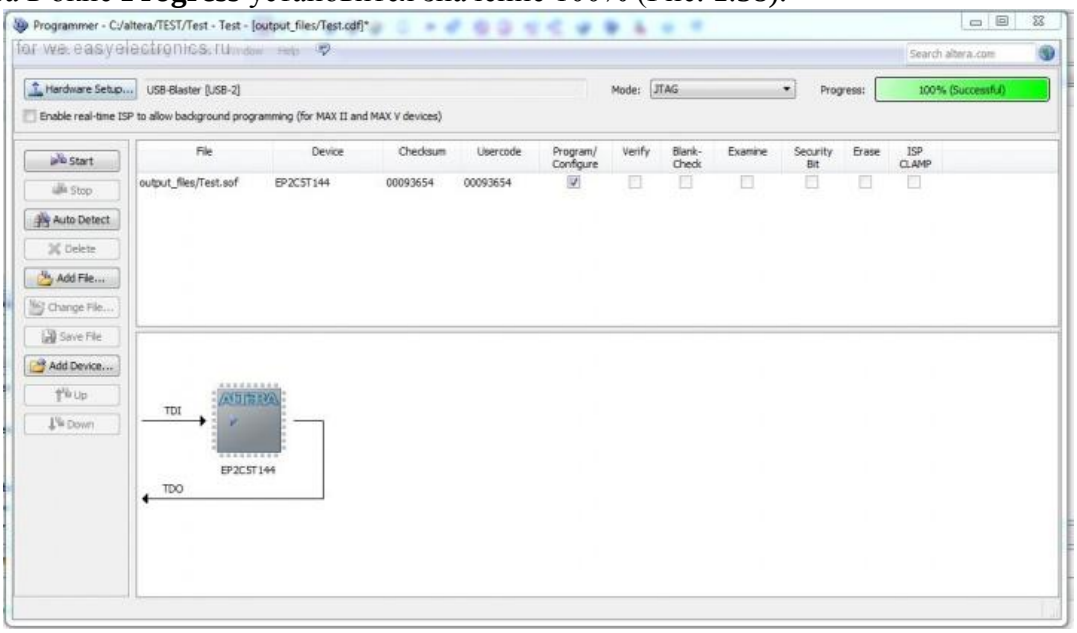


Рис. 1.58

Все бы хорошо, но как только мы отключим питание – наша программа из FPGA исчезнет, а схема ПЛИС не будет функционировать должным образом. Исправим ситуацию – зальем прошивку в конфигурационную память, откуда она будет загружаться в FPGA при включении. Выполняем **File=> Convert Programming Files…** (Рис. 1.59)

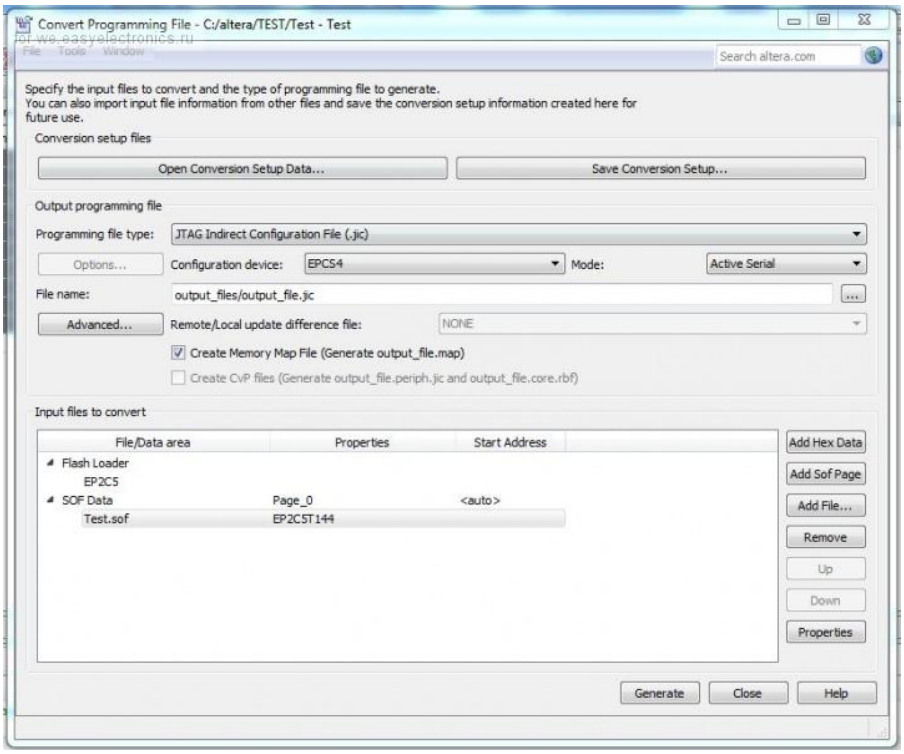


Рис. 1.59

Указываем выходной формат *.**jic**, добавляем устройство в **Flash Loader** и файл с прошивкой *.**sof** и нажимаем **Generate**. Теперь прошиваем **FPGA** с помощью сгенерированного файла **jic**, и наша программа не теряется при выключении питания (Рис. 1.6).

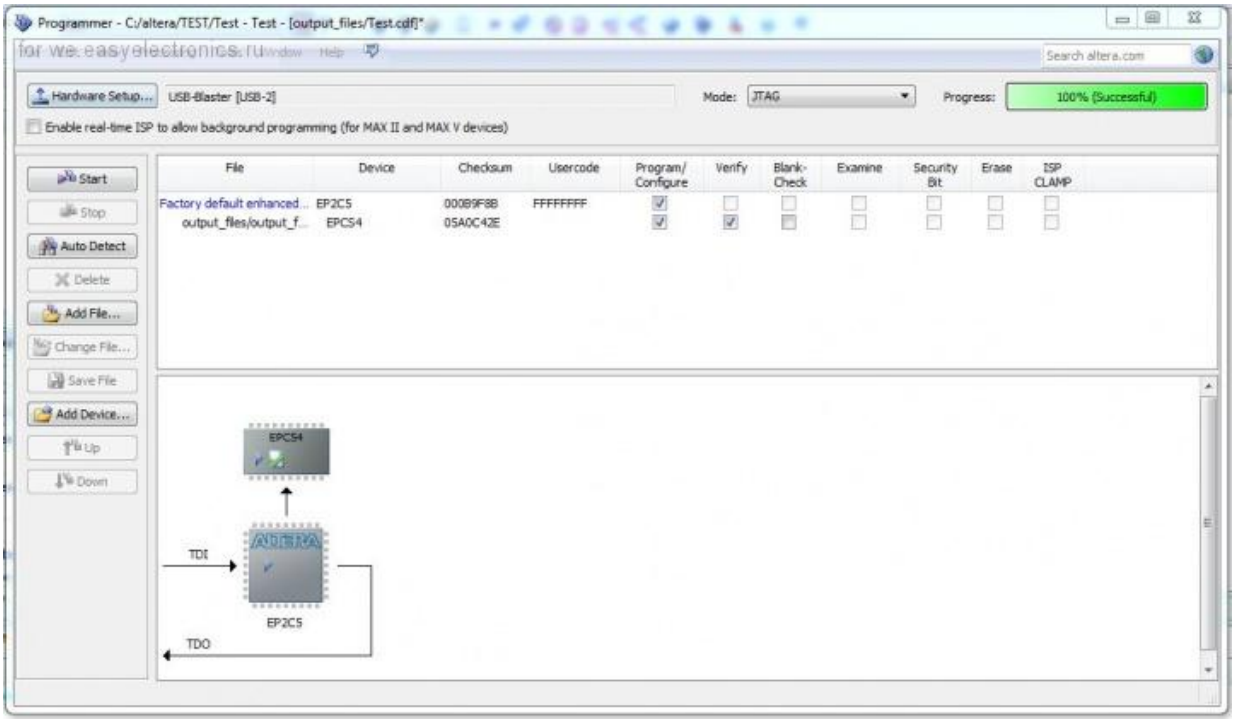


Рис. 1.6

Работает это следующим образом. Сначала в FPGA зашивается **Flash Loader** (Так сказать программатор, реализованный на ПЛИС), а потом с его помощью по интерфейсу **Active Serial** уже в конфигурационную микросхему зашивается наша конфигурация. Поэтому процесс программирования длится чуть дольше, чем обычно.

Проверка работы подключенной схемы элемента «and»

После успешной прошивки светодиод горит, это значит, что на нем единичный сигнал, соответствующий логической единице. По таблице истинности элемента and единичный сигнал появляется на выходе, если на оба входа подано по единице. Так оно и есть: на плате обе о**тжатых** кнопки находятся в состоянии единицы.

Но как только мы **нажимаем** одну из кнопок и тем самым подаем на неё сигнал 0 (при этом на двух кнопках одновременно оказываются сигналы 1 и 0), светодиод гаснет, то есть переходит в состояние нуль. Если нажать на обе кнопки одновременно, происходит то же самое: светодиод гаснет, то есть переходит в состояние нуль. Результаты этого эксперимента соответствуют логике работе элемента and.

Задание для самостоятельной работы

1. После загрузки файла конфигурации в ПЛИС провести исследование логического элемента. Для этого с помощью кнопок BUTTON0 и BUTTON1 последовательно установить возможные комбинации логических уровней на входах элемента «and». При этом каждый раз контролировать логический уровень на выходе элемента «and». Если светодиод LEDG0 светится – это логическая единица, иначе – логический ноль. По результатам исследования заполнить таблицу истинности для этого элемента.

Записать логическое выражение, соответствующее полученной таблице истинности.

2. Замените в схеме логического элемента И (and2) на элемент ИЛИ (or2), не трогая входы и выход. Выполните исследование этого элемента, в последовательности, описанной в пунктах 2-8 для элемента И.

3. Замените в схеме логического элемента ИЛИ (or2) на элемент НЕ (not) с одним входом и одним выходом. Выполните с этим элементом работу, описанную в пунктах 2-8 для элемента ИЛИ.

Содержание отчета

1. Цель работы.
2. Последовательность исследования работы логических элементов.
3. Последовательность создания программных файлов .vhdl.
4. Последовательность создания конфигурационных файлов ПЛИС в среде Quartus II.
5. Результаты исследования работы логических элементов на плате.
6. Таблицы истинности исследуемых логических элементов.
7. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Работа с комбинационными схемами.

Цель работы: Научиться синтезировать логические схемы по заданной таблице истинности.

Краткие теоретические сведения

Любая логическая схема без памяти полностью описывается таблицей истинности. Эта таблица является исходной информацией для синтеза схемы на основе логических элементов «И», «ИЛИ», «НЕ». Для разработки требуемого цифрового устройства сначала на основе таблицы истинности записывают его логическое выражение. Затем с целью упрощения цифрового устройства минимизируют его логическое выражение и далее разрабатывают схему, реализующую полученное логическое выражение. Логические выражения можно получить способами на основе совершенной дизъюнктивной нормальной формы (СДНФ);

СДНФ – это запись, в которой в конъюнкциях участвуют все переменные, причем в одной последовательности, а конъюнкции объединены дизъюнкциями.

Она составляется на основе таблицы истинности, например для схемы у которой на выходе единица, если на входы подается две и более единиц:

X3	X2	X1	У
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Находим строки в таблице, в которых у=1. Это строки 4, 6,7,8. Там, где 0 ставим знак инверсии. Для этих строк записываем собственные функции, Такая система функций называется системой собственных функций, а затем объединяем их в общую функцию для выхода У. Сколько выходов, столько и функций.

У= ¯x1x2x3∨ X1¯x2x3 ∨X1x2¯x3 ∨ X1x2x3

СКНФ составляется на основе таблицы истинности по правилу: **для каждого набора переменных, при котором функция равна 0, записывается сумма, в которой с отрицанием берутся переменные, имеющие значение 1, а дизъюнкции объединены конъюнкциями.**

Тогда общая функция для выхода У примет вид:

У= (¯x1+x2+x3) (x1¯+x2+x3) (x1+x2+¯x3) (x1+x2+x3)

На основе этих выражений можно составить схемы устройства, реализующие заданные функции.

<u>Задание</u>:

Начертите не минимизированную схему, полученную на основе СДНФ (рис. 2).Она включает четыре логических элемента на 3 входа – AND3, один логический элемент на четыре входа – OR4 , три входа и один выход.

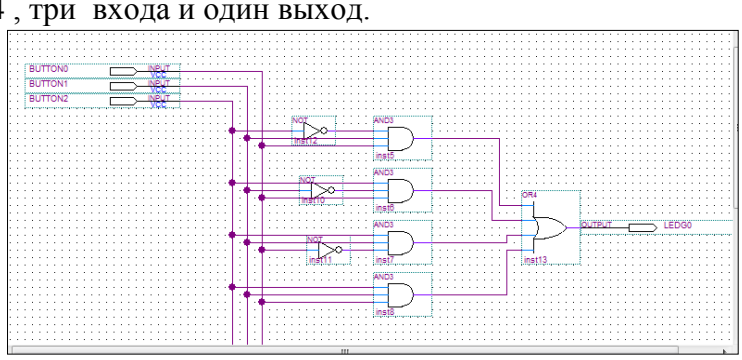


Рис.2

Входы имеют имена BUTTON0, BUTTON1, BUTTON2, выход - LEDG0 в соответствии с именами элементов на плате DEO, как в лабораторной работе №1. Все элементы соединены линиями.

2. Выполните её синтез, моделирование, анализ работы на основе временной диаграммы.

3. Минимизируйте эту схему с помощью **карты Карно** (Рис.2.1):

X1	**X2** ＼ **X3**	**0**	**1**
0	**0**	**0**	**0**
0	**1**	**0**	**1**
1	**1**	**1**	**1**
1	**0**	**0**	**1**

Рис.2.1

4. Постройте её на одном рабочем поле с неминимизированной, подсоедините к ней те же входы, выход минимизированной комбинационной схемы обозначьте **LEDG1**.

5. Смоделируйте работу схем. Получите диаграмму для нулевых значений входов схемы. Осуществите перебор всех значений входов. Сделайте это автоматически. Для этого выделите все входы и объедините их в группу (Рис.2.2).

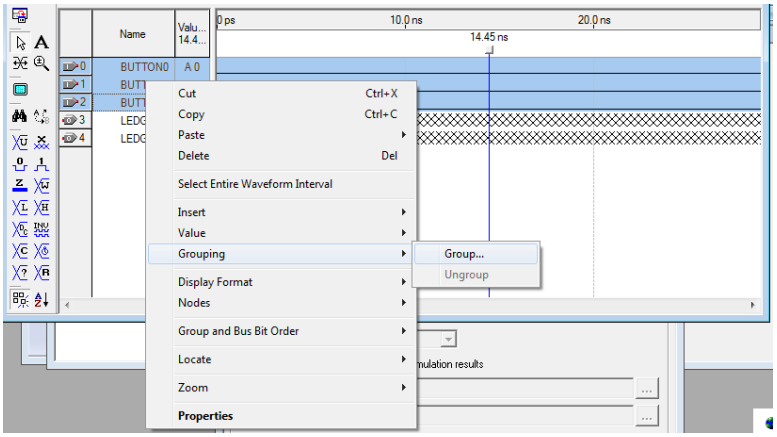


Рис.2.2

Дайте имя группе «X».

Рис.2.3

Теперь примените к группе полный перебор значений входных сигналов, тем самым обеспечим полный пересчет работы схемы, для этого выделим строку X, кнопку **Count Value:**

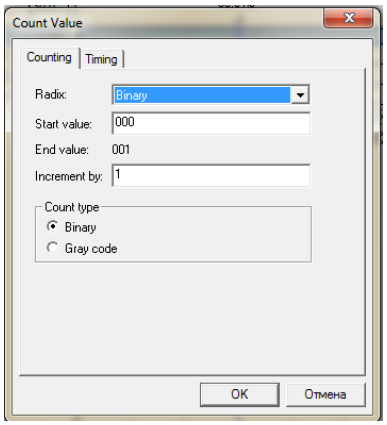


Рис.2.4

Выберите режим **Binary**. А на вкладке **Timing** задать **Count every** - 20 ns, OK.Сохраните диаграмму, нажав на дискету на панели инструментов.

6.Откройте вкладку **Simulator Tools**, сгенерируйте симуляцию, выполните **Start и Report**. Проанализируйте работу комбинационных схем - не минимизированной и минимизированной на основе общей диаграммы.

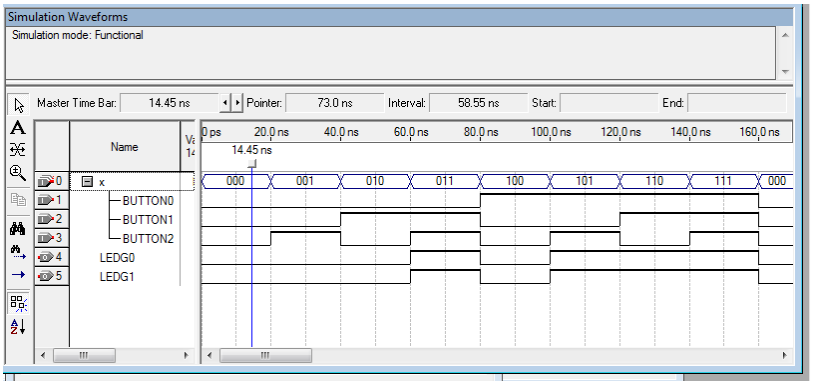


Рис.2.5

7. Выполните программирование этой комбинационной схемы, в последовательности, указанной в пункте 7 лабораторной работы №1 сначала для основной схемы, а потом для минимизированной.

8. Выполните размещение программы на ПЛИС в последовательности, описанной в пункте 8 лабораторной работы №1.

Когда будете выполнять назначение выводов в окне планировщика выводов **Pin Planner** (рис. 1.47), в соответствии с картой назначений кнопок на плате DEO, **входу BUTTON0 назначьте вывод H2, входу BUTTON1- вывод G3, входу BUTTON2 - вывод F1 (**рис.1.48**).** Выходу **LEDG0** назначьте вывод **J1**, выходу **LEDG1** - вывод **J2 (**рис.1.49**).** В файле **lab1** появятся номера реальных выводов, к которым подключены входы и выходы разработанной вами схемы.

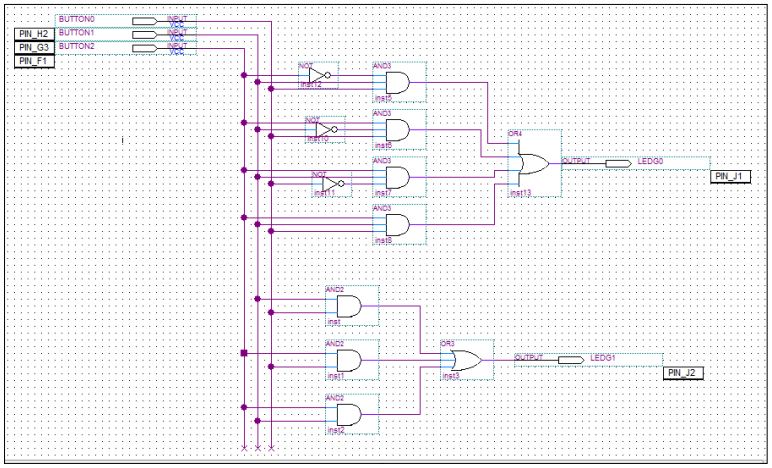


Рис.2.7

9. По результатам исследования заполнить таблицу 2.

BUTTON2	BUTTON1	BUTTON0	LEDG0	LEDG1
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Содержание отчета

1. Заданная таблица истинности.
2. Логическое выражение на основе СДНФ.
3. Минимизированное логическое выражение.
4. Схемы, синтезированные на основе СДНФ и в результате минимизации.
5. Таблицы истинности, полученные в результате исследования схем.
6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Работа с комбинационными схемами средней степени интеграции.

Цель работы: Создание и исследование работы схемы мультиплексора в среде Quartus II.

Последовательность работы:

1. Откройте **Quartus**. Создайте новый проект **File=>New Project Wizard =>lab3** в **своей папке**, созданной в лабораторной работе №1.

2. Создайте графический файл **lab3** комбинационной не минимизированной схемы мультиплексора на основе таблицы истинности Рис. 3.1:

A	X0	X1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Рис.3.1

X0	X1	\A	0	1
0	0			
0	1			**1**
1	1		**1**	1
1	0		**1**	

Рис.3.2

и логического выражения СНДФ: $\mathbf{У=A^{-}x_0x_1^{-} \vee A^{-}\, x_0x_1 \vee A\, x_0^{-}\, x_1 \vee A\, x_0x_1}$.

3. Создайте минимизированную схему мультиплексора на основе **карты Карно (Рис.3.2)** и СНДФ (напишите её самостоятельно). Подсоедините минимизированную схему к входам не минимизированной схемы на общем поле графического редактора.

4. Синтезируйте общую схему; с помощью симулятора получите временную диаграмму работы мультиплексора. Сравнить работу не минимизированной и минимизированной схем работы мультиплексора на основе диаграммы.

5. Создайте блок минимизированной схемы мультиплексора в редакторе блоков изображений.

6. Для этого создайте второй файл с расширением **.bdf** и поместите в него минимизированную схему. У Вас получилось два одинаковых файла. Сохраните этот файл с под именем **mx.bsf**.

7. В окне проекта появляется два файла lab2.dbf и мх. dbf. Откомпилируйте проект.

8. Откройте файл **mx.bsf,** выполните команду: **File=>Create/Update => Create Symbol Files for Current File:**

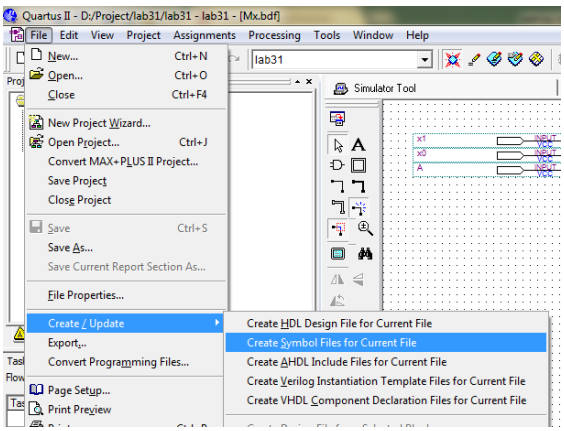


Рис.3.3

9. Откройте файл lab3.dbf. Сделайте 2 щелчка по рабочему полю этого файла, в окне Simbol появилась папка с Вашим проектом **Project**, в ней – Ваш файл **мх**. Щелкнете по **мх**. Появится блок мультиплексора. Нажмите ОК (Рис.3.4).

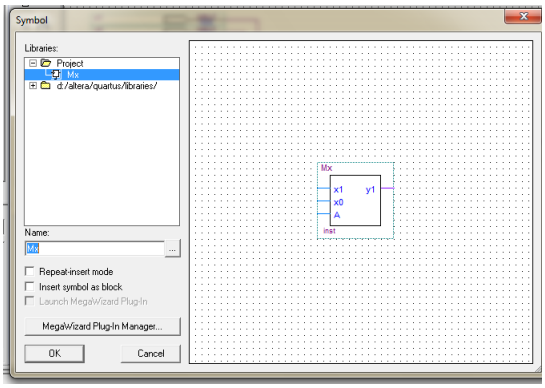


Рис.3.4

И вставьте этот блок **мх** на поле файла **lab3.dbf (**Рис.3.5).

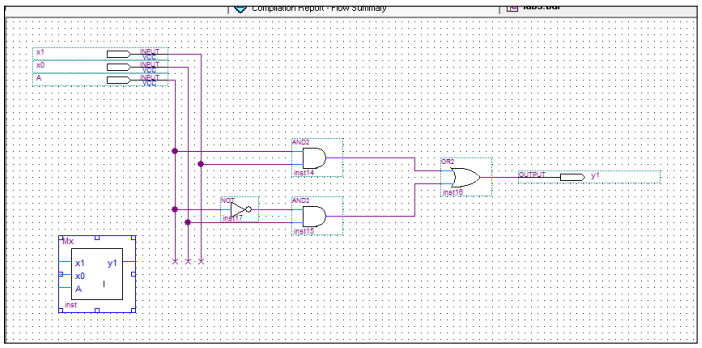


Рис.3.5

Щелкнете по блоку мультиплексора, откроется подробная минимизированная схема из файла **мх**. Оставим блок мультиплексора, минимизированную схему мультиплексора удалим, но входы и выходы не удаляйте, а подсоедините их к блоку мультиплексора (Рис.3.6). Сохраните файл **мх** в формате **.bsf.**

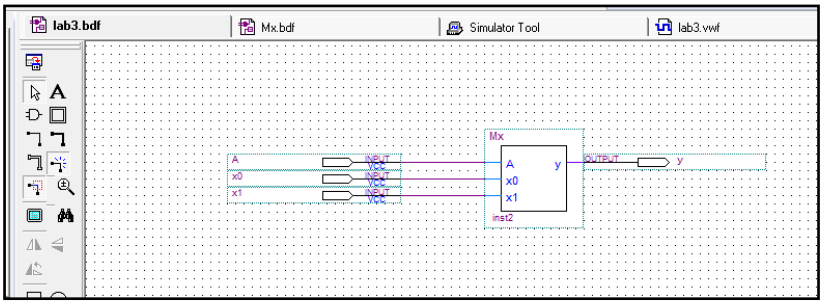


Рис.3.6

10. Смоделируйте работу блока мультиплексора в **Simulatore Tools.** Убедитесь, что он работает в соответствии с таблицей истинности.

11. Отредактируйте блок в редакторе: **Edit-selector – symbol** (Рис. 3.7).

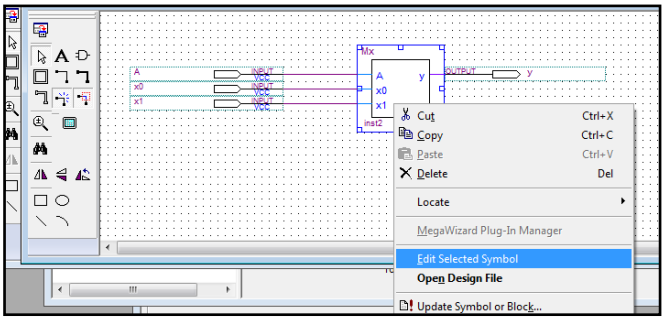


Рис. 3.7

Появляется поле редактора модулей

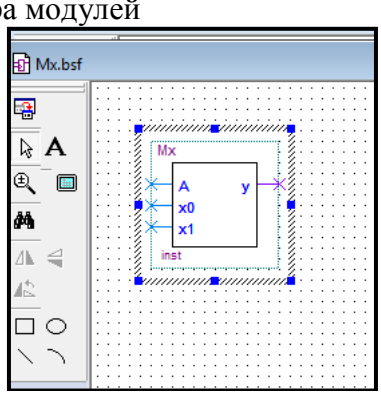


Рис.3.8

Редактируйте и приведите к гостовскому виду, используя инструменты на левой панели, например: **Line Tool**:

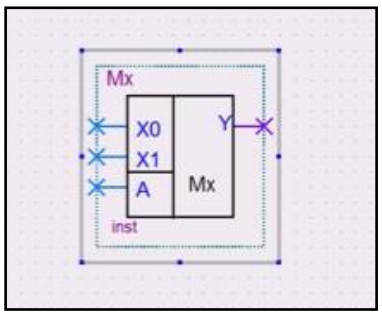


Рис.3.9

Сохранить отредактированный блок, нажав на изображение дискеты. В схеме мультиплексора заменим блок старый на отредактированный с помощью команды **Update symbol or block:**

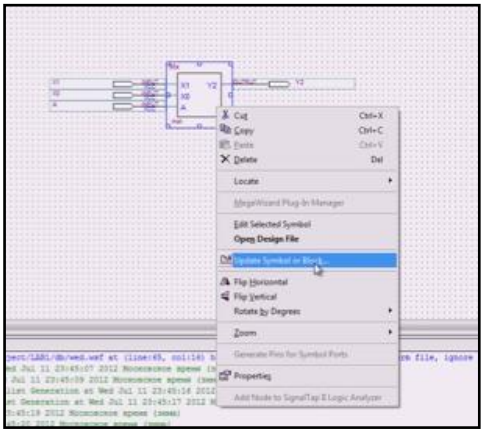


Рис.3.10

Далее в сообщении нажмите **ОК**:

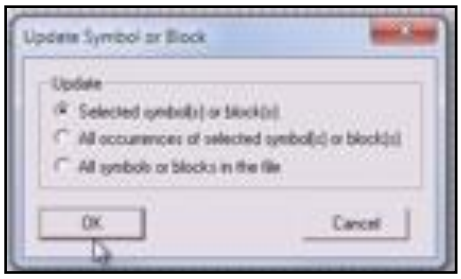


Рис.3.11

Получите обновленный блок мультиплексора:

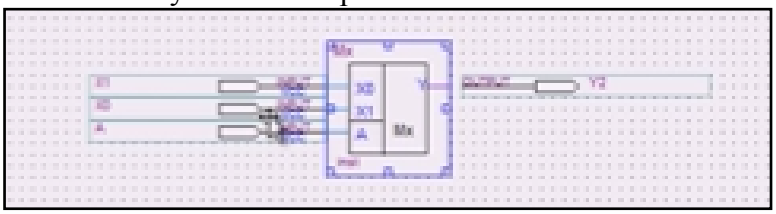


Рис.3.12

10. Синтезируйте блок, генерируйте симулятор, получите отчет по работе симулятора, на основе отчета проанализируйте работу блока мультиплексора.

11. Выполните программирование мультиплексора в последовательности, описанной в пункте 7 лабораторной работы 1 на основе следующего кода:

```
library ieee;
use ieee.std_logic_ll64.all;
entity busmux  is
port (
x0: in bit;
x1: in bit;
A: in bit;
y: out bit;
end;
architecture behavior of busmus is
begin
```

```
process (A)
begin
case A is
when ‘0’=> y <=  x0;
when ‘1’=> y <=  x1;
and case;
end process;
end behavior;
```

12. Выполните подключение мультиплексора к плате DEO и проверку правильности его работы как в пункте 8. Когда будете работать с окном планировщика выводов **Pin Planner** (рис. 1.44) сделайте в поле **Location** следующие назначения для входов **BUTTON**1 (x1), **BUTTON0 (x0)**, **BUTTON**2 (A) в соответствии с картой назначений кнопок на плате DEO: **входу BUTTON1 назначим вывод H2, входу BUTTON0 вывод - G3, входу BUTTON**2 – вывод **F1** (Рис. 1.48), а выходу **LEDG0 (y1) – J0 (**Рис. 1.49**).**

13. По результатам исследования заполнить таблицу 3.

Таблица 3

BUTTON2	**BUTTON0**	**BUTTON**1	**LEDG0**
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Отчет выполнить в той же форме, как в лабораторной №2

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Работа с комбинационными схемами средней степени интеграции.
Цель работы: Создание и исследование работы схемы дешифратора в среде Quartus II.

Логика работы дешифратора: на входы подаем номер того выхода y, на котором мы хотим видеть единицу.

Порядок работы:

1.Откройте **Quartus**. Создайте новый проект **File=>New Project Wizard** =>**lab4** в **своей папке**, созданной в лабораторной работе №1.

2. Создайте проект неполного дешифратора: **File=>New Project Wizard** =>**lab4** и файл верхнего уровня **lab4** на основе таблицы истинности (Рис.4.0):

BUTTON0	BUTTON1	LEDG0	LEDG1	LEDG2	LEDG3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Рис.4.0

3. Запрограммируйте простой дешифратор в соответствии со вторым вариантом потоковой формы описания архитектуры:

```
entity decoder is
port (  BUTTON0, BUTTON1:in bit;
LEDG0, LEDG1, LEDG2, LEDG3 out bit;
end decoder;

architecture arch of decoder is
begin
LEDG0<= (not BUTTON0) and (not BUTTON1);
LEDG1<= (not BUTTON1) and BUTTON0;
LEDG2<= (not BUTTON01) and BUTTON1;
LEDG3<= BUTTON0 and BUTTON1;
end arch;
```

4**.** На основе этой программы простого дешифратора напишите программу дешифратора с управляющим входом E.

5. Подключите дешифратор к реальной плате DEO в соответствии с пунктом **8** лабораторной работы №1. Когда будете работать с окном планировщика выводов **Pin Planner** (рис. 1.44), сделайте в поле **Location** следующие назначения для входов в соответствии с картой назначений кнопок на плате **DEO**: входу **BUTTON1** назначьте вывод **H2,** входу **BUTTON0** вывод **- G3 (**Рис.1.45**),** выходу **LEDG0 – J0, LEDG1–J1, LEDG2 – J2, LEDG3 – J3** (Рис. 1.46). Проанализируйте работу программы и схемы на учебном стенде DEO.

6. Заполните таблицу Рис.4.0 по результатам исследования работы на плате DEO .

Отчет выполнить, как в лабораторной №3.

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема: Работа с комбинационными схемами средней степени интеграции.

Цель работы: Создание и исследование работы схемы шифратора с управляющим входом разрешения работы **E** в среде **Quartus II**.

Логика работы шифратора*: *на выходе мы получаем код номера входа, на который подана 1.

1) В среде Quartus II создайте проект шифратора **File=>New Project Wizard** =>**lab5** и файл верхнего уровня **lab5.dbf.** На основе таблицы истинности:

X0	X1	X2	X3	У1	Y0	E
0	0	0	0	+	+	1
0	0	0	1	1	1	0
0	0	1	0	1	0	0
0	0	1	1	+	+	1
0	1	0	0	0	1	0
0	1	0	1	+	+	1
0	1	1	0	+	+	1
0	1	1	1	+	+	1
1	0	0	0	0	0	0
1	0	0	1	+	+	1
1	0	1	0	+	+	1
1	0	1	0	+	+	1
1	0	1	1	+	+	1
1	1	0	0	+	+	1
1	1	0	1	+	+	1
1	1	1	0	+	+	1
1	1	1	1	+	+	1

и карт Карно отдельно для выходов y0, y1 и E:

для y1:

x0	x1\x2x3	00	01	11	10
0	0	+	1	+	1
0	1	0	+	+	+
1	1	+	+	+	+
1	0	0	+	+	+

для y0:

x0	x1\x2x3	00	01	11	10
0	0	+	1	+	0
0	1	1	+	+	+
1	1	+	+	+	+
1	0	0	+	+	+

для E:

x0	x1\x2x3	00	01	11	10
0	0	1		1	
0	1		1	1	1
1	1	1	1	1	1
1	0		1	1	1

Напишите формулы СДНФ для всех карт Карно. На их основе постройте минимизированную схему шифратора в графическом редакторе в файле lab5.bdf.

3) Синтезируйте схему, с помощью симулятора смоделируйте работу шифратора, то есть получите временную диаграмму работы. На её основе проанализируйте работу шифратора.

4) Выполните программирование шифратора в последовательности, описанной в пункте 7 лабораторной работы №1.

5) Подключите схему к плате DEO. Когда будете работать с окном планировщика выводов **Pin Planner** (рис. 1.44) сделайте в поле **Location** следующие назначения для входов в соответствии с картой назначения переключателей (Рис.5) на плате DEO: **входу SW0** назначьте вывод **J6, входу SW1** вывод **– H5, входу SW2** – вывод **H6, входу SW2 -** вывод **G4** , выходам **LEDG0 – J0, LEDG1 – J1**.

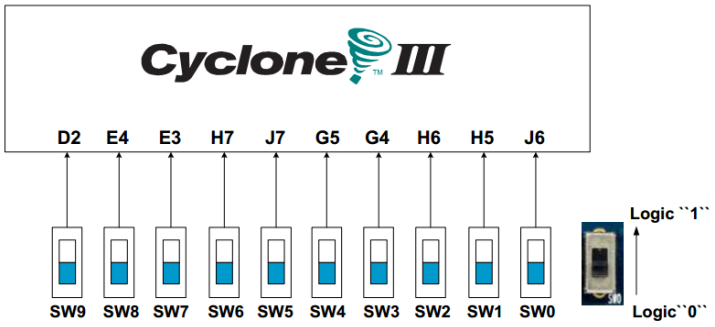


Рис.5

6) Проанализируйте работу программы и схемы на учебном стенде DEO и по результатам исследования заполните таблицу 5.

Таблица 5

SW0	**SW1**	**SW2**	**SW3**	**LEDG1**	**LEDG0**	E
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			

0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Отчет выполнить по схеме в лабораторной работе №4

ЛАБОРАТОРНАЯ РАБОТА № 6

Тема: работа с асинхронными триггерами.

Цель работы: создание и исследование работы асинхронного **RS -триггера** в среде Quartus

Теоретические сведения:

Триггер – это логическая схема с обратной связью, представляющая собой элементарное запоминающее устройство или устройство хранения.

Триггеры предназначены для запоминания двоичной информации. Использование триггеров позволяет реализовывать устройства оперативной памяти (то есть памяти, информация в которой хранится только на время вычислений). Однако триггеры могут использоваться и для построения некоторых цифровых устройств с памятью, таких как счётчики, преобразователи последовательного кода в параллельный или цифровые линии задержки.

Порядок выполнения работы:

1. В среде Quartus II создайте проект RS-триггера: **File=>New Project Wizard =>lab6** и графический файл верхнего уровня **lab6.bdf .**

2. Начертите в этом файле схему RS -триггера **(**Рис. 6) на основе таблицы истинности, по которой можно отследить логику работы триггера:

R	S	Q(t)	Q(t+1)	Режимы
0	0	0	0	хранения
0	0	1	1	
0	1	0	1	установки
0	1	1	1	
1	0	0	0	сброса
1	0	1	0	
1	1	0	+	запрещенный
1	1	1	+	

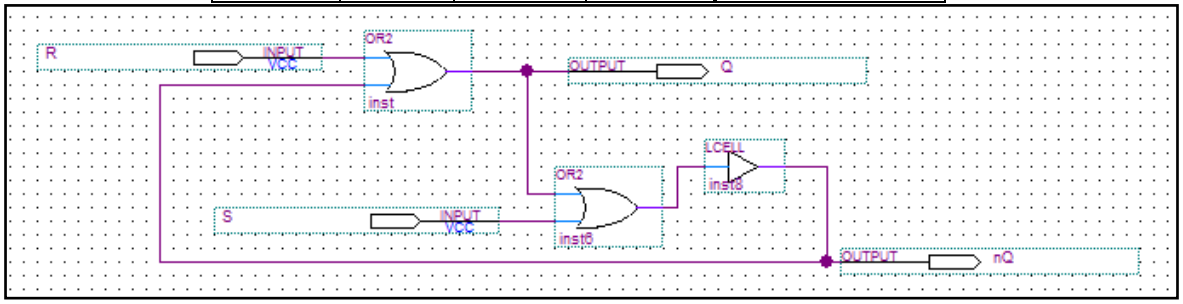


Рис.6

Минимизируйте эту схему с помощью:
карты Карно

R	S \Q	0	1
0	0		1
0	1	1	1
1	1	+	+
1	0		

и НСДФ: **У=nR ^Q V S**

3. Самостоятельно создайте минимизированную схему и расположите на одном рабочем поле с не минимизированной. Синтезируйте, смоделируйте и проанализируйте и сравните работу обеих схем.

4. Создайте блок для минимизированного RS-триггера в редакторе блоков изображений, как в пунктах 5-10 из лабораторной работы №2. Исследуйте работу блока RS - триггера на основе диаграммы. Сравните работу схемы и блока.

5. Выполните программирование RS-триггера как в пункте 7 лабораторной работы №1 на основе таблицы истинности.

```
entity my_ff  is
port R:in std_ligic;
S: in std_ligic;
Q: out std_ligic;
Q1: out std_ligic;
end my_ff;

architecture my_ff_code  of my_ff  is
begin
Q<= ‘1’ when R=‘0’ and S=‘1’
else ‘0’ when R=‘1’ and S=‘0’;
Q1<= ‘1’ when R=‘1’ and S=‘0’
else ‘0’ when R=‘0’ and S=‘1’;
end my_ff_code  ;
```

6. Выполните подключение схемы триггера к плате DEO как в пункте 8 лабораторной работы №1. Используя карту подключения двух входов через кнопки BUTTON0 (S) и BUTTON1 (R) и карту подключения двух выходов через диоды LEDG0 (Q0), LEDG1 (Q1), LEDG2 (Q2) .

7. Сделайте выводы о работе схемы.

8. Форма отчета как в лабораторной работе №5

ЛАБОРАТОРНАЯ РАБОТА № 7

Тема: Работа с асинхронными триггерами.

Цель работы: Создание и исследование работы универсального JK-триггера в среде Quartus II.

Порядок выполнения работы:

1.В той же папке проекта **lab6** создайте файл верхнего уровня **lab61.bdf .** Создайте в нем графический файл из двух схем **JK триггера** (Рис. 7) не минимизированной (на основе блока асинхронного RS-триггера) на основе таблицы истинности:

J	K	Q(t)	Q(t+1)	Режим
0	0	0	0	хранения
0	0	1	1	
0	1	0	0	сброса
0	1	1	0	

1	**0**	**0**	**1**	**установки**
1	**0**	**1**	**1**	
1	**1**	**0**	**1**	**инверсия**
1	**1**	**1**	**0**	

И минимизированной схемы, которую создайте с помощью карты Карно:

J	**K\Q**	**0**	**1**
0	**0**		**1**
0	**1**		
1	**1**	**1**	
1	**0**	**1**	**1**

и логического выражения СДНФ для выхода: **Q=J^nQ v nK^Q**

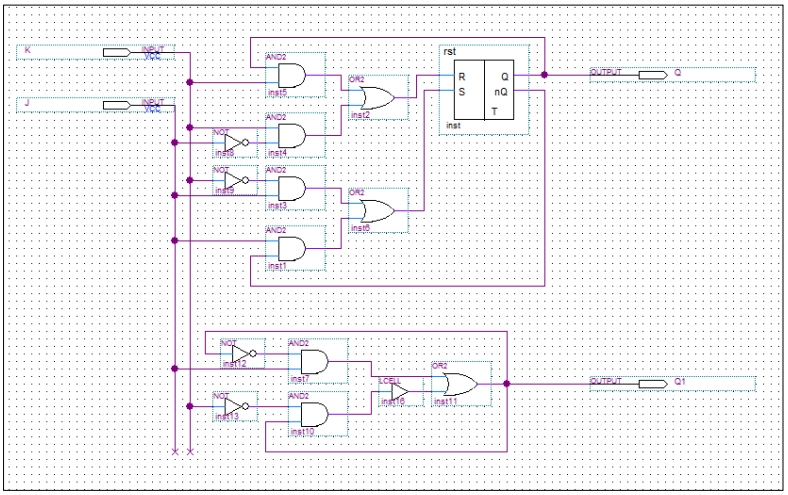


Рис.7

9. Синтезируйте, смоделируйте и проанализируйте работу и сравните работу двух схем JK-триггера.

10. Исследуйте работу блока минимизированного JK -триггера на основе диаграммы. Сделать вывод о правильности работы блока**.**

11. Выполните программирование JK -триггера на основе таблицы истинности как в пункте 7 лабораторной работы №1.

```
entity my_ff_code  is
port J:in std_ligic;
K: in std_ligic;
Q: out std_ligic;
Q1: out std_ligic;
end my_ff_code;

architecture my_ff  of my_ff_code  is
begin
Q<= ‘1’ when J=‘1’ and K=‘0’
else ‘0’ when K=‘1’ and J=‘0’;
Q1<= ‘1’ when J=‘1’ and K=‘0’
se ‘0’ when K=‘1’ and J=‘0’;
end my_ffcode;
```

12. Выполните подключение минимизированной схемы триггера к плате DEO как в пункте 8 лабораторной работы №1. Используя карту подключения двух входов через кнопки BUTTON0 (J) и BUTTON1 (K) и карту подключения двух выходов через диоды LEDG0 (Q0) и LEDG1 (Q1). Сделайте выводы о работе схемы на плате DEO.

ЛАБОРАТОРНАЯ РАБОТА № 8

Тема: Работа с асинхронными триггерами.

Цель работы: Создание и исследование работы асинхронного **D-триггера** в среде **Quartus**.

Порядок выполнения работы:

1. В той же папке проекта **lab6** создайте файл верхнего уровня **lab62** для **D - триггера**.
2. Создайте в нем микросхему D – триггеров на основе таблиц истинности

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

За основу возьмите асинхронный RS-триггер:

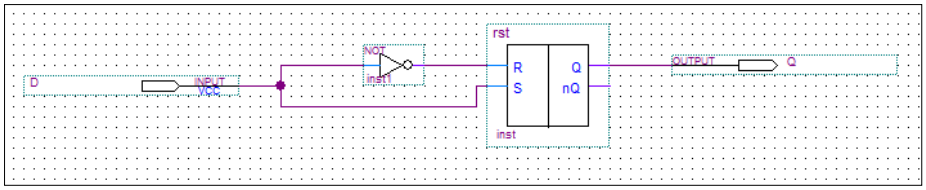


Рис.8.1

3.Самостоятельно минимизируйте схему с помощью Карты Карно:

D\Q	0	1
0	0	0
1	1	1

и логического выражения НСДФ: **Q=D**

4.Синтезируйте, моделируйте, проанализируйте, сравнит работу двух схем D-триггера.

5.Создайте блок минимизированного D-триггера. Исследуйте работу блока D - триггера на основе диаграммы. Сделать вывод о правильности работы блока**.**

6.Выполните программирование D -триггера на основе таблицы истинности как в пункте 7 лабораторной работы №1.

```
entity my_ff  is
port (
D:in std_ligic;
Q: out std_ligic;
end my_ff;

architecture my_ff_code  of my_ff  is
begin
Q<= ‘1’ when D=‘1’
```

else '0' when D='0';
end my_ffcode_code ;

7.Выполните подключение минимизированной схемы триггера к плате DEO как в пункте 8 лабораторной работы №1. Используя карту подключения входов через кнопки, Назначьте входу D - BUTTON0 и карту подключения выходов через диоды для выходов Q0 - LEDG0 и Q1 - LEDG1. Сделайте выводы о работе схемы.

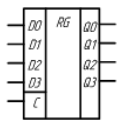
ЛАБОРАТОРНАЯ РАБОТА № 9

Тема: работа с регистровыми устройствами.
Цель работы: создание и исследование работы RG-регистра хранения в среде Quartus II.

Теоретические сведения:

Регистр хранения – это многоразрядный, многовходовой триггер.

Регистры предназначены для хранения и преобразования многоразрядных двоичных чисел. Для запоминания отдельных разрядов числа могут применяться триггеры различных типов. Одиночный триггер можно считать одноразрядным регистром. Занесение информации в регистр называется операцией записи. Операция выдачи информации из регистра – считывание. Перед записью информации в регистр, его необходимо обнулить.



Для записи информации в регистр на его входных выводах (D0-D3) нужно установить логические уровни, после чего на вход синхронизации (C) подать разрешающий импульс — логическую единицу. После этого на выходах Q0-Q3 появится записанное слово. Регистры запоминают входные сигналы только в момент времени, определяемый сигналом синхронизации.

Порядок выполнения работы:

1. В среде Quartus II создайте проект регистра хранения: **File=>New Project Wizard =>lab9** и графический файл верхнего уровня **lab9.bdf .**

2. Создайте в нем графическую схему регистра хранения RG (Рис.9.5)

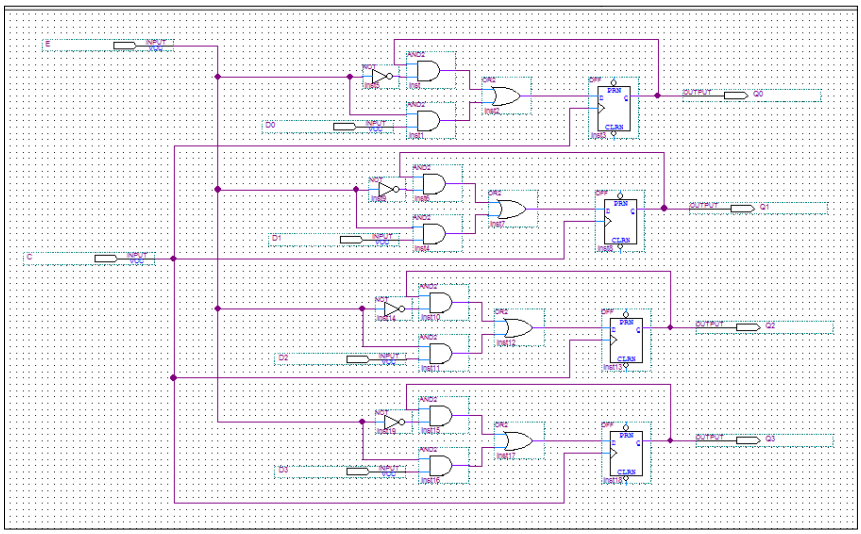


Рис.9,5

В этой схеме можно использовать стандартный блок D - триггера **dff** из папки **Storage** библиотеки **Primitives** окна **Simbol**.

В регистре хранения содержится 4 элементарных блока RG, подсоединенных к входу E – разрешение работы. Переименуйте последующие входы на D1, D2, D3. Переименуйте последующие выходы на Q1, Q2, Q3. Входы E и C – общие для всех элементов.

3. Сохраним схему, компилируем.

4. Создаем файл диаграммы *.vwf . Не забудьте сгруппировать входы D0,D1,D2,D3 и выходы Q0, Q1, Q2, Q3

На вход C подайте тактовый сигнал: кнопка **Overwrite Clock**:

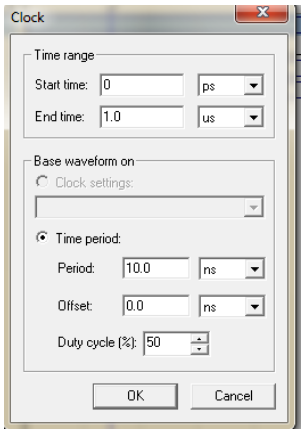


Рис.9.2

Нажмите OK.

Входы и выходы определяем в 16-ричной системе **Octal**:

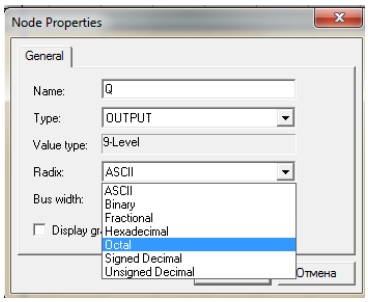


Рис.9.3

Подайте на вход число 4-х разрядное **Value-> Arbitrary Value**:

В появившемся окне:

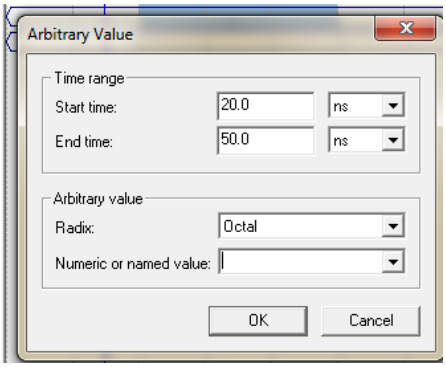


Рис.9.4

В пустое поле **Numeric or named value** введем любое число, например 5.

Но так как разрешения работы нет, число в регистр не запишется.

Подайте еще число на входы D, но на вход “разрешение работы” E подать единицу.

Повторите ввод других чисел на входы D при E=1 и E=0.

5. Промоделируйте схему. Проанализируем работу схемы регистра хранения.

Таблица истинности RG:

E	B	Q(t)	Q(t+1)	режимы
0	0	0	0	хранение
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	запись
1	0	1	0	
1	1	0	1	
1	1	1	1	

6. Выполните программирование т хранения как в пункте 7 лабораторной работы №1 на основе таблицы истинности.

```
entity registr1 is
port (
D:in std_logic_vector (3 downto 0);
clk: in std_logic;
Q: buffer std_logic_vector (3 downto 0)
);
end registr1;
architecture behav of registr1 is
begin
process(clk,D,E)
begin
if(rising_edge(slk)) then
if(E='1') then
Q<=D;
else
Q<=Q;
end if;
end if;
end process;
end behav;
```

.

7. Выполните подключение схемы триггера к плате DEO как в пункте 8 лабораторной работы №1. Используя карту подключения входов через переключатели и карту подключения выходов через диоды.

8. Сделайте выводы о работе схемы после подключения к реальной учебной плате DEO.

9. Выполнить отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Расскажите что такое ПЛИС.
2. Расскажите о преимуществах ПЛИС.
3. Что такое файл конфигурации ПЛИС?
4. Каким образом производится конфигурирование ПЛИС?
5. Назначение системы автоматизированного проектирования Quartus II.
6. Этапы создания проекта в системе Quartus II.

7. Работа с графическим редактором Quartus II.
8. Как производится подключение электрической схемы внутри ПЛИС к внешним выводам?
9. Расскажите, как устроена учебная плата.
10. Опишите схему исследования логических элементов.
11. Приведите условное графическое изображение основных логических элементов в российском стандарте и в системе Quartus II.
12. Изобразите таблицы истинности основных логических элементов.
13. Как задать перебор всех значений входов схемы?
14. Что такое таблица истинности?
15. Можно ли на основе таблицы истинности определить логику работы устройства?
16. Что такое метод Карно?
17. Как создать минимизированную логическую функцию СДНФ для выхода схемы на основе карты Карно?
18. Как создать минимизированную схему на основе СДНФ?
19. В чем состоит методика синтеза комбинационных схем?
20. Что такое комбинационные схемы средней степени интеграции?
21. В чем состоят принципы работы мультиплексора?
22. В чем отличие в работе дешифратора и шифратора?
23. Как часто используется асинхронный одноступенчатый RS-триггер? Приведите несколько примеров его практического использования.
24. Поясните, как работает асинхронный RS-триггер?
25. Какова логика работы синхронных JK и D - триггеров? Приведите примеры их практического использования.
26. Какой триггер используется в практике наиболее часто?
27. Какой триггер используется как базовый при создании других триггеров в ПЛИС?
28. Чем отличаются асинхронные триггера от синхронных?
29. Что такое тактовый сигнал?
30. Какой уровень сигнала является активным для синхронных триггеров?
31. Чем определяется разрядность регистров?
32. Назначение параллельного регистра.
33. Назначение последовательного регистра.
34. Объяснить принцип работы последовательного регистра.
35. Объяснить принцип работы параллельного регистра.
36. Для чего служат регистры?
37. Какие Вы знаете разновидности регистров?
38. Какие цифровые устройства Вы еще знаете?

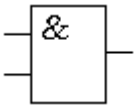
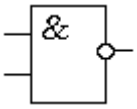
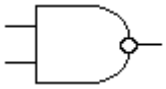
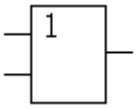
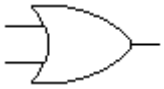
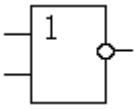
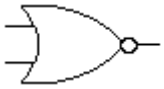
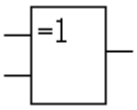
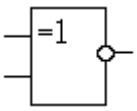
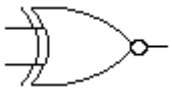
Название элемента	**Российское обозначение**	**Название элемента в Quartus**	**Обозначение в Quartus**
"И"		and	
"И-НЕ"		nand	
"ИЛИ"		or	
«ИЛИ-НЕ»		nor	
«ИСКЛЮЧАЮЩЕЕ ИЛИ»		xor	
«ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ»		xnor	

Таблица графических обозначений базовых логических элементов в соответствии с российским и международным ГОСТ.

Литература

1. Грушвицкий Р.И., Мурсаев А.Х. Угрюмов Е.П. Проектирование систем на микросхемах с программируемой структурой. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006. - 736с.

2. Бибило П. Н. Основы языка VHDL. Изд. 3-е, доп.- М.: Издательство ЛКИ, 2007. - 328с.

3. Поляков А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М.: СОЛОН-Пресс, 2003. -320с.

4. Quartus II Handbook Version 10.1. Интернет ресурс. http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf

5. Барри Уилкинсон. Основы проектирования цифровых схем. Изд. Дом «Вильямс». Москва-Санкт -Питербург-Киев, 2004.

Корректор ***Эркинбек к. Ж.***
Редактор ***Турдукулова А.К.***
Тех.редактор ***Кочоров А.Д.***

Подписано к печати 16.10.2015 г. Формат бумаги 60х84$^{1}/_{16}$.
Бумага офс. Печать офс. Объем 3 п.л. Тираж 50 экз. Заказ 417. Цена 51,3с.
Бишкек, ул. Сухомлинова, 20. ИЦ “Текник” КГТУ им. И.Раззакова, т.: 54-29-43
e-mail: *beknur@mail.ru*