

АРХИТЕКТУРА ВМиС

Методические указания по выполнению лабораторных работ для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника» (профиль: «Вычислительные машины, комплексы, системы и сети»)

Введение

Широкое внедрение автоматики во все сферы человеческой деятельности, наблюдаемое в настоящее время, предъявляет жесткие требования к изделиям электронной техники. Это связано, с одной стороны, с возрастанием важности и сложности решаемых задач, а с другой стороны, необходимостью улучшения качественных характеристик, таких как: быстродействие, надежность, потребляемая мощность, габариты, стоимость и др. Одним из путей решения данной проблемы является использование новой элементной базы — программируемых логических интегральных схем (ПЛИС — Programmable Logic Device — PLD).

ПЛИС представляют собой интегральные схемы, обладающие гибкостью заказных БИС (больших интегральных схем) и доступностью традиционной "жесткой" логики. Современные ПЛИС характеризуются низкой стоимостью, высоким быстродействием, значительными функциональными возможностями, многократностью перепрограммирования, низкой потребляемой мощностью и др. Главным отличительным свойством ПЛИС является возможность их настройки на выполнение заданных функций самим пользователем, таким образом реализовывать даже достаточно сложные проекты в сжатые сроки в виде конкурентоспособных устройств и систем. По существу, разработка устройств на основе ПЛИС представляет собой новую технологию проектирования электронных схем, включая их изготовление и сопровождение.

Одним из доказательств перспективности рассматриваемой элементной базы служит ежегодное появление новых, имеющих более совершенную архитектуру, поколений ПЛИС, а также постоянно растущий объем выпуска ПЛИС такими производителями как ALTERA, Atmel, Xilinx, LATTICE и др., среди которых наиболее распространенными в нашей стране являются ПЛИС американской фирмы ALTERA.

В настоящее время фирма ALTERA выпускает такие семейства программируемых интегральных схем, как MAX 3000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000, FLEX 10K, APEX 20K, ACEX, Mercury и др. Основные

характеристики наиболее популярных из них приведены в таблице 1.

Семейства MAX представляют собой электрически перепрограммируемые микросхемы с энергонезависимой прошивкой, допускающие $10^4 - 10^6$ циклов программирования – стирания, в то время как микросхемы семейства FLEX изготовлены по технологии SRAM с загрузкой конфигурации при включении питания, что в свою очередь дает теоретически неограниченное число циклов программирования.

Таблица 1

Семейства ПЛИС					
Характеристики	MAX 7000E(S)	MAX 9000	FLEX 8000A	FLEX 10K	APEX20K
Архитектура	матрицы И-ИЛИ	матрицы И-ИЛИ	таблицы перекодировки	таблицы перекодировки	таблицы перекодировки
Логическая емкость ¹	600-5000	6000-12000	2500-16000	10000-100000	263000-26770000
Внутренняя память ²	нет	нет	нет	6-24 Кбит	53-540 Кбит
Число пользовательских выводов	36-164	60-216	68-208	59-406	252-780

Семейство микросхем программируемой логики ACEX 1K [4] производится на основе SRAM технологии с напряжением питания 2,5V. Микросхемы выпускаются с логической емкостью от 10 000 до 100 000 эквивалентных вентилях. Это семейство ПЛИС выпускаются в корпусах TQFP с количеством выводов 100, 144, 208 и в корпусах BGA с 256 и 484 выводами. Работая при напряжении питания 2,5 V устройства ACEX 1K позволяют реализовывать полностью 64 битные устройства. Это семейство микросхем имеет встроенное ОЗУ емкостью от 12 до 48 Kbit.

ALTERA выпускает три типа микросхем семейства APEx с напряжением питания 2,5 V (APEx 20K) и 1,8 V (APEx 20KE и APEx 20KC). Логическая емкость микросхем программируемой логики семейства APEx от 30 тысяч до

¹ единица измерения - число эквивалентных логических вентилях (вентилей типа 2И-НЕ);

² при использовании внутренней памяти доступные пользователю логические ресурсы СБИС не уменьшаются.

2,7 миллионов эквивалентных вентилях. Архитектура этих микросхем MultiCore основана на структуре нового поколения MegaLAB. Структура MegaLAB состоит из 16 логических блоков (LABs), которые в свою очередь состоят из 10 логических элементов, каждый из которых используется для реализации таблицы перекодировки (look-up-table – LUT), а также содержит усовершенствованные интегрированные блоки памяти объемом от 53 до 540 Kbit. Семейство APEX программируемых логических устройств обеспечивают гибкость и высокую логическую емкость, необходимую для разработки устройств типа систем на кристалле (system-on-a-programmable-chip – SOPC). Эти микросхемы позволяют разрабатывать устройства обработки 64 битных данных, и передачи данных с предельной скоростью передачи порядка 840 Mb/s, полностью удовлетворяют стандарту PCI.

«Система автоматизированного проектирования QUARTUS II»

САПР QUARTUS II представляет собой интегрированную среду для разработки цифровых устройств на базе программируемых логических интегральных схем (ПЛИС) фирмы ALTERA и обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

- создание проектов устройств;
- синтез структур и трассировку внутренних связей ПЛИС;
- подготовку данных для программирования или конфигурирования ПЛИС (компиляцию);
- верификацию проектов (функциональное моделирование и временной анализ);
- программирование или конфигурирование ПЛИС.

Ниже представлено главное окно программы (рис. 1.1), оно имеет стандартный интерфейс Windows-приложений. В заголовке окна программы указывается имя и путь последнего проекта, с которым велась работа.

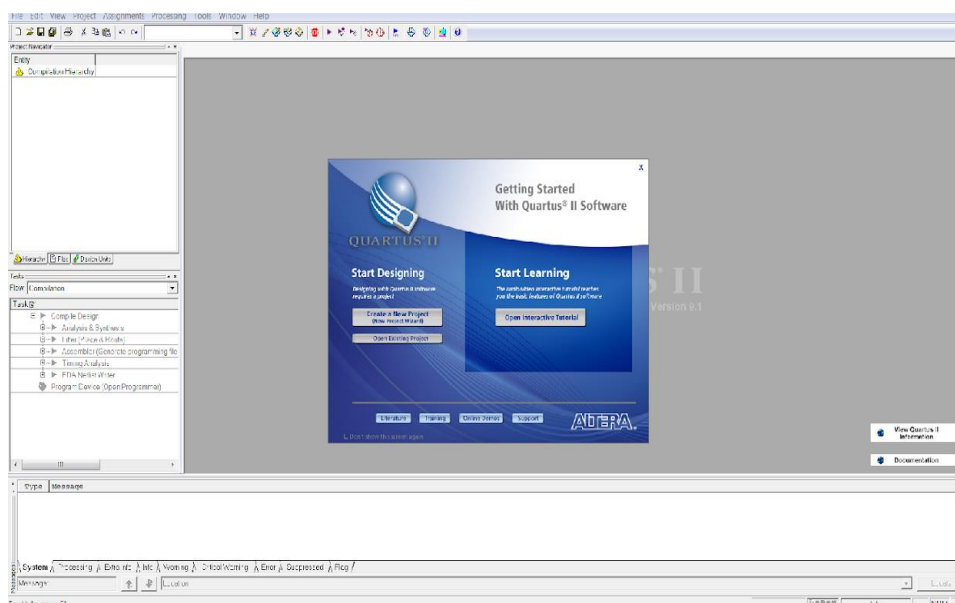


Рис. 1.1. Главное окно системы Quartus II

Приложения системы *QUARTUS II*. В состав пакета *QUARTUS II* входят следующие связанные между собой приложения, реализующие все этапы разработки цифровых устройств на ПЛИС фирмы ALTERA:

Приложения для ввода проектов (редакторы проектов):

Graphic Editor – графический редактор (рис. 1.2), предназначен для ввода проекта в виде схемы соединений символов элементов, извлекаемых из стандартных библиотек пакета либо из библиотеки пользователя. Открывается с помощью пункта *Block Diagram/Schematic File* меню *New*. Главным достоинством графического способа ввода проекта является его традиционность и наглядность, связанная с привычностью разработчиков к восприятию изображений схем.

Вставка символа производится двойным кликом левой кнопки мыши на свободном месте окна графического редактора. Введенные символы и группы символов можно копировать, удалять, поворачивать, перетаскивать в другую область окна обычным приемом “*Drag&Drop*”, а так же обмениваться с другими окнами через буфер обмена. Выводы элементов можно соединять сигнальными проводниками либо присваиванием одинаковых имен проводникам которые должны быть соединены.

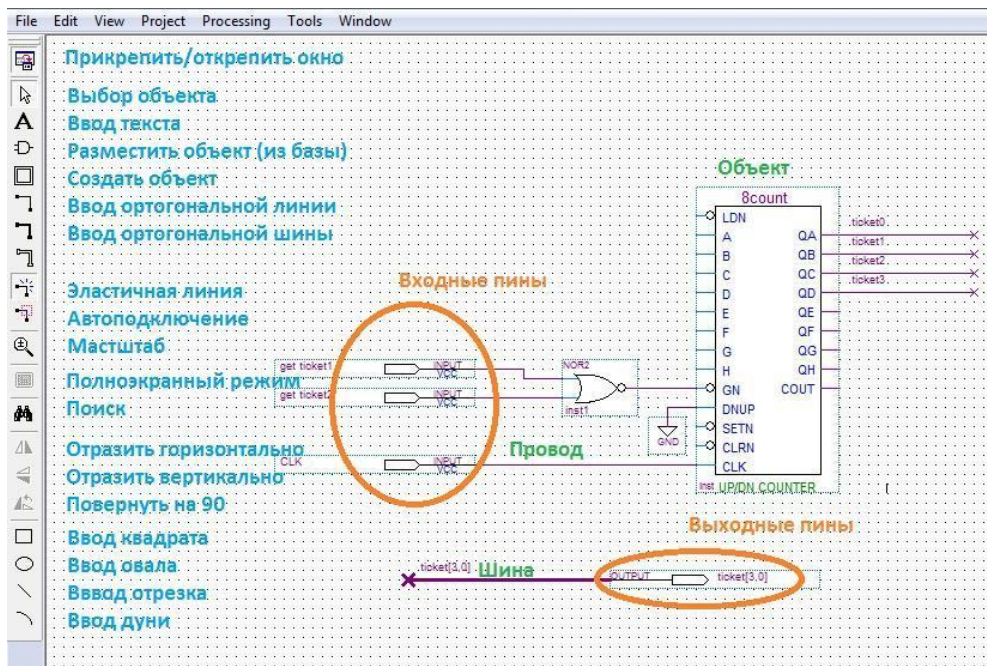


Рис. 1.2. Графический редактор

Simulation Waveform Editor – редактор временных диаграмм (некоторые разработчики называют это приложение сигнальным редактором), который выполняет двойную функцию: на этапе ввода обеспечивает ввод логики проекта в виде диаграмм (эпюр) состояний входов и выходов, а на этапе моделирования обеспечивает ввод диаграмм тестовых (эталонных) входных состояний моделируемого устройства и задание перечня тестируемых выходов. Открывается с помощью пункта *Vector Waveform File* меню *New*. Окно сигнального редактора представлено на рис. 1.3.

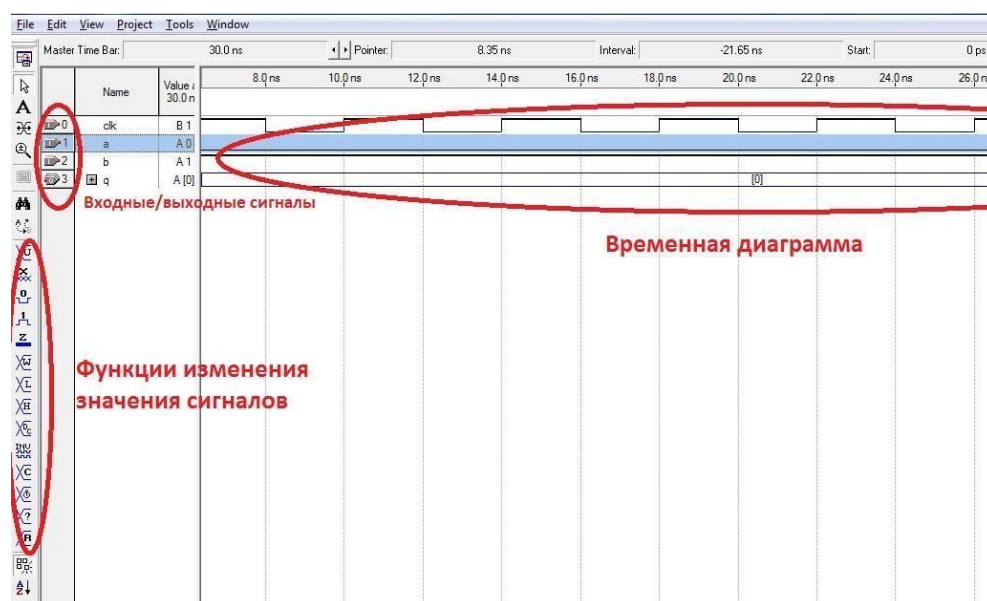


Рис. 1.3. Редактор временных диаграмм

Время в течении которого будет проводиться моделирование задается в меню “Edit”, пункт “Set End Time”, шаг временной сетки задается в меню “Edit”, пункт “Grid Size”.

Окно сигнального редактора имеет два поля, разделённых вертикальными линиями. Первое поле (“Name”), предназначено для ввода имени вывода и второе поле “Value” показаны состояния выводов, соответствующие положению специальной вертикальной визирной линии. Основное поле предназначено для задания требуемых состояний выводов, при этом используются инструменты с панели инструментов редактора, которая расположена вертикально вдоль левой стороны окна. Активизация панели инструментов происходит только в том случае, если выделен один из узлов. Чтобы выделить узел, необходимо щёлкнуть левой кнопкой мыши на имени узла, можно также выделить любой участок вдоль горизонтальной оси, при этом границы выделяемых участков привязываются к сетке.

Размещаются выводы при помощи всплывающего меню пункт “Insert Node” или меню Edit/Insert Node. Введенные выводы можно редактировать, перемещать, удалять, размножать (с обязательным редактированием имени или типа, если это необходимо).

Text Editor – текстовый редактор (рис. 1.4) является инструментом для создания текстовых файлов проекта на языках описания аппаратуры: AHDL (.tdf), VHDL (.vhd), Verilog HDL (.v). В этом текстовом редакторе можно работать также с произвольным файлом формата ASCII. Все перечисленные файлы проекта можно создавать в любом текстовом редакторе, однако данный редактор имеет встроенные возможности ввода файлов проекта, их компиляции и отладки с выдачей сообщений об ошибках и их локализацией в исходном тексте или в тексте вспомогательных файлов. Кроме того, существуют шаблоны языковых конструкций для AHDL, VHDL и Verilog HDL, выполнено окрашивание синтаксических конструкций. В данном редакторе можно вручную редактировать файлы назначений и конфигурации (.acf), а также делать установки конфигурации для компилятора, симулятора и временного анализатора.

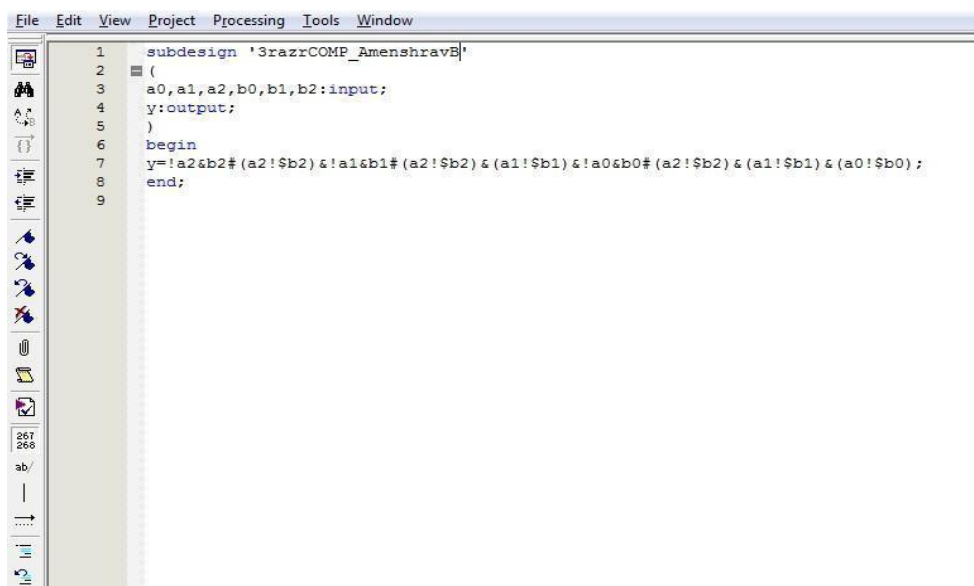



Рис. 1.4. Окно текстового редактора

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.

Окно навигатор проекта

Данное окно (**Project Navigator**) содержит страницы со следующими закладками: **Hierarchy**, **Files**, **Design Units**. (рис.1.5).

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

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 1.7. 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.

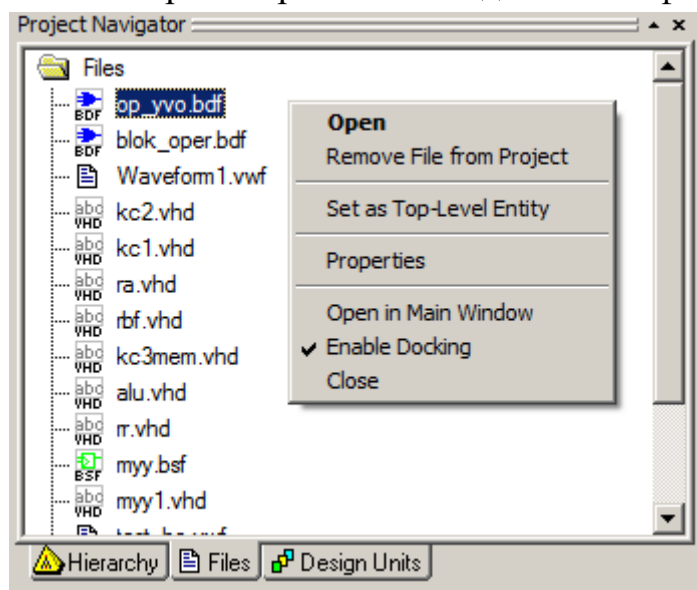


Рис. 1.7. Страница **Files** навигатора проекта

Страница **Design Units** навигатора отображает все компоненты проекта, использованный способ описания проекта, язык описания и файл с описанием компонента (рис. 1.8).

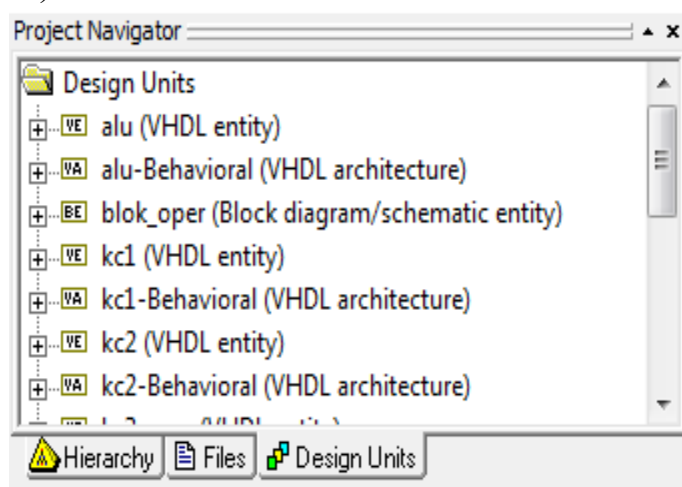


Рис. 1.8. Страница **Design Units** навигатора проекта

Окно состояния процедуры компиляции проекта В данном окне (**Tasks**)

показаны различные уровни компиляции в ходе проекта. **Компиляция проекта.** Компиляция представляет собой процесс преобразования описания проекта 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:

Анализ и синтез (Analysis & Synthesis). Составной частью процесса компиляции проекта является этап анализа и синтеза. Соответствующий модуль компилятора Quartus II строит базу данных проекта, которая объединяет все файлы описания проекта в единое целое с учетом иерархического представления проекта. Созданная база данных проекта в дальнейшем будет обновляться другими модулями компилятора до тех пор, пока не будет получен полностью оптимизированный проект. После своего создания база данных содержит только таблицу соединений проекта (netlist). После завершения полной компиляции – полностью оптимизированный, смонтированный проект, который используется для создания файлов, применяемых для временного моделирования, временного анализа, анализа потребляемой мощности и программирования кристалла.

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 HDL в их аппаратную реализацию на ресурсах кристалла таких, как функциональные преобразователи, триггеры защелки, 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).

Размещение и трассировка (Fitter). 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.

Временной анализ (Classic Timing Analysis). Проверка соответствия реализованного проекта требованиям быстродействия.

Запуск Quartus II и открытие проекта

Запустите САПР Quartus II. Появится главное окно менеджера проекта Quartus II, на фоне которого будет помещена панель (рис.2.1), с предложением либо пройти курс обучения, используя предложенное интерактивное руководство, либо открыть существующий проект, либо создать новый проект.

Чтобы создать новый проект, нажмите соответствующую кнопку, после чего будет запущен *New Project Wizard*.

Чтобы пройти курс обучения щелкните по соответствующей кнопке, чтобы открыть существующий проект нажмите кнопку **Open Existing Project**.

Кнопка **Literature** предназначена для получения информации по пакету Quartus II, включая подробное руководство по использованию пакета Quartus II. Также можно получить информацию обо всех типах ПЛИС, выпускаемых фирмой Altera.

Кнопка **Training** предназначена для выполнения, размещенных на сайте аудио-видео сеансов обучения в режиме онлайн.

Кнопка **Online Demos** позволяет наблюдать в режиме онлайн демонстрацию применения пакета Quartus II.

Кнопка **Support** позволяет получать техническую поддержку.

Чтобы открыть существующий проект, необходимо выполнить двойной щелчок мышью по файлу с именем проекта и расширением **.qpf**, либо использовать команду *Open Project* из меню **File** менеджера проекта. В этом же

меню содержится команда *Recent Project*, с помощью которой можно выбрать один проект из списка последних открывавшихся пользователем проектов.

Чтобы отказаться от предлагаемых действий закройте окно стандартным образом, используя значок **X** в правом верхнем углу окна.

Чтобы запретить вывод на экран этого окна при повторном запуске пакета, следует щелкнуть мышью в поле *Don't show this screen again*.

Для создания нового проекта используется утилита *New Project Wizard* (NPW). Чтобы её вызвать, выполните следующие шаги. Щелкните левой клавишей мыши по кнопке **File** на панели инструментов Quartus II. Затем в появившемся окне щелкните по строке *New Project Wizard*. Далее появится окно, представленное на рис. 2.2.



Рис. 2.1. Вводное окно пакета Quartus II

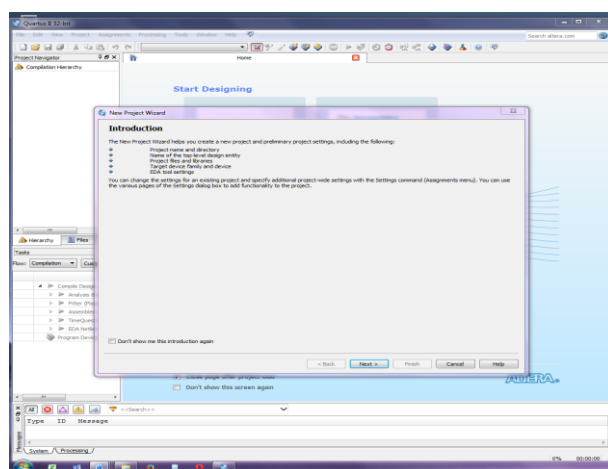



Рис. 2.2. Вводное окно NPW

В этом вводном окне перечислены шаги, которые необходимо выполнить для создания нового проекта. Для перехода к следующему окну **NPW** нажмите кнопку **Next** левой клавишей мыши.

В следующем окне [page1] **NPW**, представленном на рис. 2.3, выберите рабочую папку для размещения проекта. Укажите адрес папки, задайте имя проекта и определите модуль верхнего уровня. Если папка проекта еще не создана, QUARTUS II предложит создать новую папку. Кнопка  позволит воспользоваться браузером для задания вышеперечисленных параметров. Этап компиляции проекта всегда выполняется для модуля верхнего уровня. Поэтому, в случае необходимости, в последующем этот модуль может быть переопределен. На первой странице окна **NPW** имеется кнопка **Use Existing Project Settings**. С помощью этой кнопки можно задать установки, которые будут использоваться по умолчанию в новом проекте. Причем можно использовать установки либо из последнего открытого в Quartus II проекта, либо из указанного проекта.

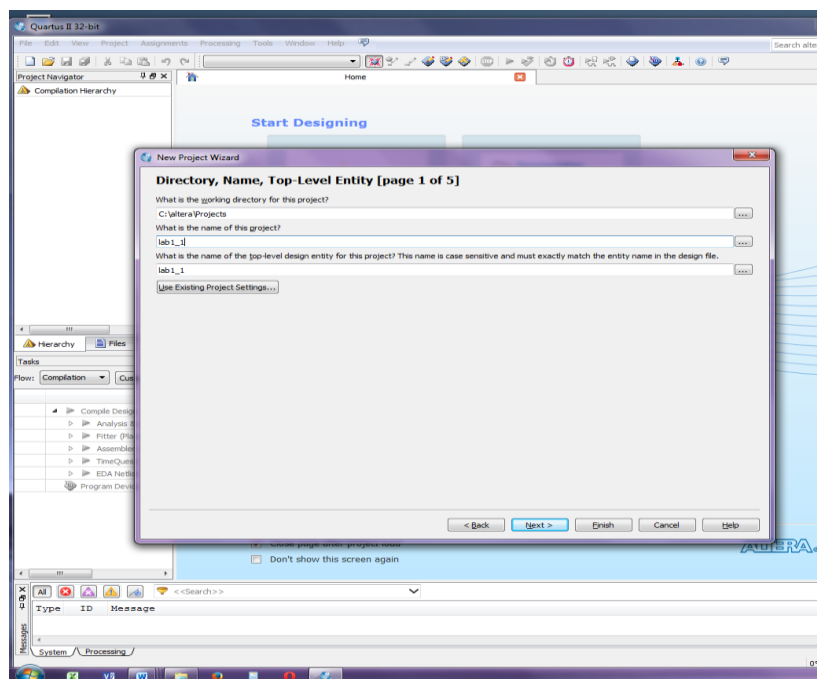


Рис. 2.3. Окно указания рабочей папки, имени проекта и модуля верхнего уровня

В следующем окне [page2] **NPW**, показанном на рис 2.4, добавьте к проекту необходимые файлы. Также как и в предыдущем окне, можно использовать кнопку браузера. Имена выбираемых файлов, с указанием пути доступа к ним, появляются в основном окне второй страницы **NPW**. С помощью кнопок **Add**, **Add All**, **Remove**, **Properties**, **Up**, **Down** можно добавить к своему проекту либо выделяемые файлы по отдельности, либо сразу все файлы. Также можно удалить файлы, посмотреть их свойства, либо переставить местами. Порядок файлов имеет значение при компиляции проекта.

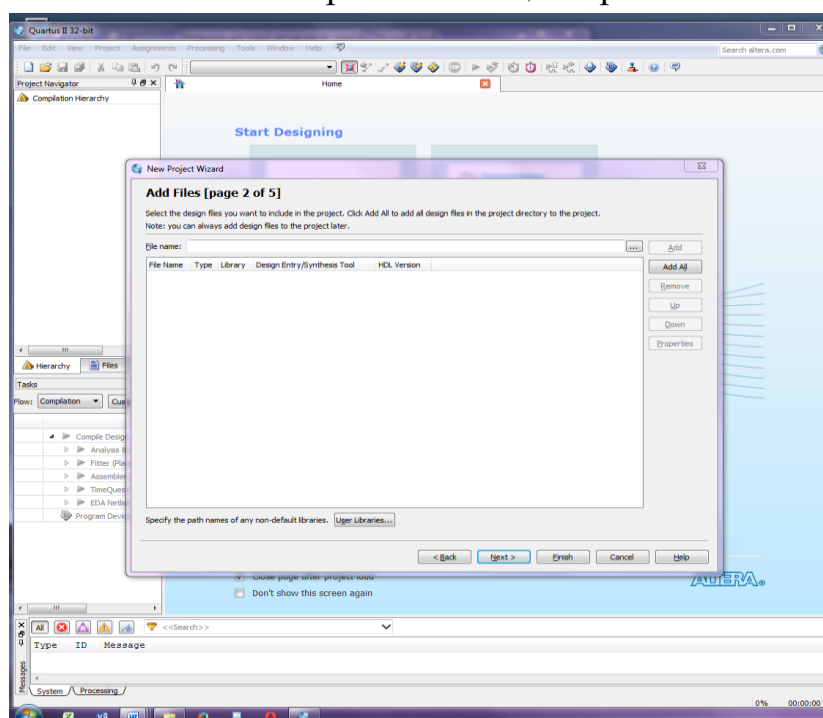


Рис. 2.4. Окно **NPW** включения в проект файлов и библиотек

В рассматриваемом окне **NPW** с помощью кнопки **User Libraries** можно добавить к проекту пользовательские библиотеки, содержащие файлы проекта на языках VHDL, Verilog, AHDL, файлы с определениями макрофункций, мегафункций и другие файлы проекта. Для перехода к следующему окну [page3] **NPW** нажмите кнопку **Next**. Вид открывшегося окна изображен на рис. 2.5.

В этом окне выбирается ПЛИС, на которой будет реализован проект. Причем кристалл может быть выбран из предлагаемого списка, либо автоматически назначен компоновщиком. В случае выбора кристалла, следует вначале выбрать его семейство в окне **Family**, затем определить тип корпуса интегральной схемы в окне **Package**, в окне **Pin Count** указать количество выводов кристалла и в окне **Speed grade** указать градацию быстродействия кристалла. Затем остается выбрать подходящий кристалл из списка в окне **Available Devices**, и с помощью кнопки **Next** перейти к следующему окну [page 4] **NPW**, показанному на рис. 2.6.

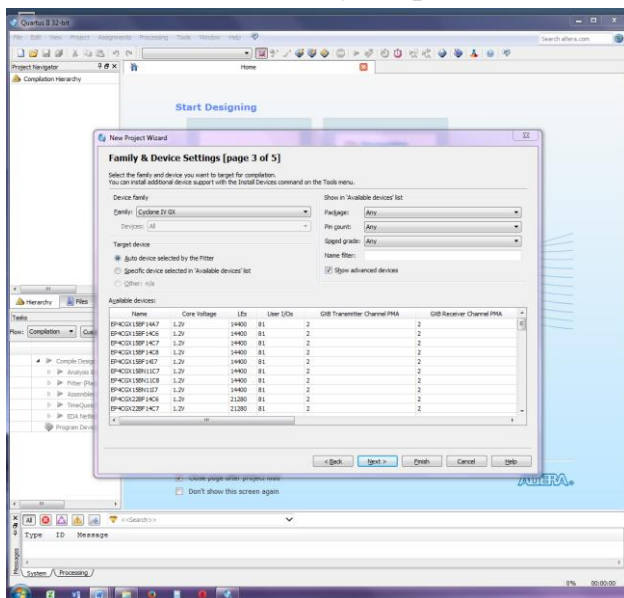


Рис. 2.5. Окно **NPW** задания целевого кристалла ПЛИС

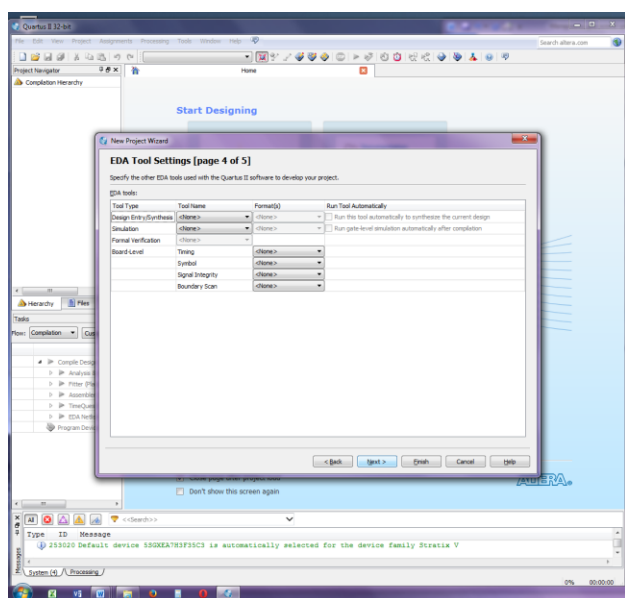


Рис. 2.6. Окно **NPW** декларации об использовании средств автоматизации сторонних производителей

В следующем окне [page 4] **NPW** можно определить средства автоматизации проектирования сторонних производителей, которые планируется использовать для ввода проекта, синтеза, моделирования или временного анализа. В последующем эти средства будут вызываться непосредственно из пакета Quartus II. Пример окна на рис. 2.6. соответствует случаю, когда дополнительные средства автоматизации проектирования использовать не планируется.

С помощью кнопки **Next** перейдите к завершающему окну [page5] **NPW**, которое показано на рис. 2.7.

Лабораторная работа №1

Графический ввод схемы и симуляция в САПР QUARTUS II

Цель работы:

Спроектировать логическую схему при помощи графического редактора САПР QUARTUS II. Исследовать работу схемы с использованием сигнального редактора САПР QUARTUS II.

1. Основные теоретические сведения:

Алгебра логики и основные логические элементы

Математической основой цифровой электроники и вычислительной техники является алгебра логики или булева алгебра (по имени английского математика Джона Буля).

В булевой алгебре независимые переменные или аргументы (X) принимают только два значения: «0» или «1». Зависимые переменные или функции (Y) также могут принимать только два значения: «0» или «1». Функция алгебры логики (ФАЛ) представляется в виде:

$$Y = F(X_1; X_2; X_3 \dots X_n)$$

Данная форма задания ФАЛ называется алгебраической.

Основными логическими функциями являются:

- логическое отрицание (инверсия):

$$Y = \overline{X}$$

- логическое сложение (дизъюнкция):

$$Y = X_1 + X_2 \quad \text{или} \quad Y = X_1 \vee X_2$$

- логическое умножение (конъюнкция):

$$Y = X_1 \cdot X_2 \quad \text{или} \quad Y = X_1 \wedge X_2$$

К более сложным функциям алгебры логики относятся:

- функция равнозначности (эквивалентности):

$$Y = X_1 \cdot X_2 + \overline{X_1} \cdot \overline{X_2} \quad \text{или} \quad Y = X_1 \sim X_2$$

- функция неравнозначности (сложение по модулю два):

$$Y = X_1 \cdot \overline{X_2} + \overline{X_1} \cdot X_2 \quad \text{или} \quad Y = X_1 \oplus X_2$$

- функция Пирса (логическое сложение с отрицанием):

$$Y = \overline{X_1 + X_2}$$

- функция Шеффера (логическое умножение с отрицанием):

$$Y = \overline{X_1 \cdot X_2}$$

Для булевой алгебры справедливы следующие законы и правила:

- распределительный закон:

$$X_1(X_2 + X_3) = X_1 \cdot X_2 + X_1 \cdot X_3$$

$$X_1 + (X_2 \cdot X_3) = (X_1 + X_2) \cdot (X_1 + X_3)$$

- правило повторения:

$$X \cdot X = X, \quad X + X = X$$

- правило отрицания:

$$X \cdot \overline{X} = 0, \quad X + \overline{X} = 1$$

- теорема де Моргана:

$$\overline{X_1 + X_2} = \overline{X_1} \cdot \overline{X_2}, \quad \overline{X_1 \cdot X_2} = \overline{X_1} + \overline{X_2}$$

- тождественности:

$$X \cdot 1 = X, \quad X + 0 = X, \quad X \cdot 0 = 0, \quad X + 1 = 1$$

Схемы, реализующие логические функции, называются логическими элементами. Основные логические элементы имеют, как правило, один выход (Y) и несколько входов, число которых равно числу аргументов $(X_1; X_2; X_3; \dots X_n)$. На электрических схемах логические элементы рисуют в виде прямоугольников с выводами для входных (слева) и выходных (справа) переменных. В середине прямоугольника изображается символ, обозначающий функциональное назначение элемента.

На рисунках 3.1 – 3.7 показаны логические элементы, реализующие рассмотренные ранее функции. Там же приведены так называемые таблицы состояний или таблицы истинности, которые описывают соответствующие логические функции в двоичном коде в виде состояний входных и выходных переменных. Таблица истинности является также табличным способом задания ФАЛ.

На рисунке 3.1 показан элемент «НЕ», который реализует функцию логического отрицания $Y = \overline{X}$.

X	Y
0	1
1	0

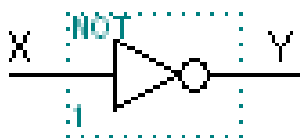
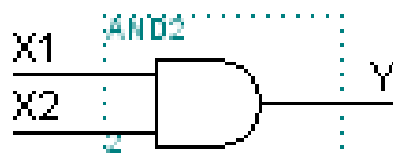
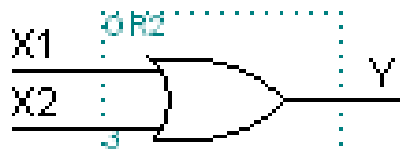


Рис. 3.1. Элемент «НЕ»

Элемент «ИЛИ» (рисунок 3.2) и элемент «И» (рисунок 3.3) реализуют функции логического сложения и логического умножения соответственно.

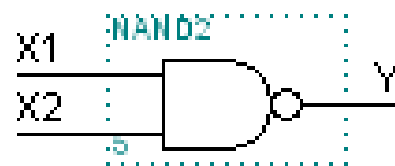
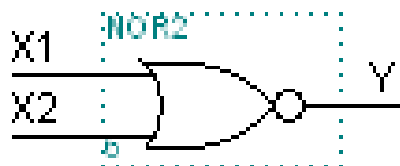


X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1
X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 3.2. Элемент «ИЛИ»

Рис. 3.3. Элемент «И»

Функции Пирса и функции Шеффера реализуются при помощи элементов «ИЛИ-НЕ» и «И-НЕ», приведенных на рисунках 3.4 и 3.5 соответственно.



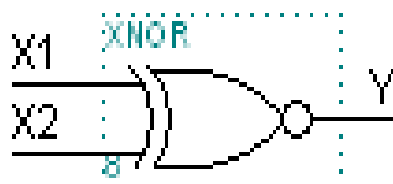
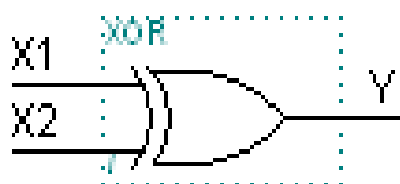
X1	X2	Y
----	----	---

0	0	1
0	1	0
1	0	0
1	1	0
X1	X2	Y
0	0	1
0	1	1
1	0	1
1	1	0

Рис. 3.4. Элемент «ИЛИ-НЕ»

Рис. 3.5. Элемент «И-НЕ»

Элемент Пирса можно получить последовательным соединением логических элементов «ИЛИ» и элемента «НЕ», а элемент Шеффера - в виде последовательного соединения логических элементов «И» и элемента «НЕ». На рисунках 3.6 и 3.7 показаны элементы «отрицающее ИЛИ» та «отрицающее ИЛИ-НЕ», которые реализуют функции неравнозначности и неравнозначности с отрицанием соответственно.



X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0
X1	X2	Y
0	0	1

0	1	0
1	0	0
1	1	1

Рис. 3.6. Элемент
«отрицающее ИЛИ»

Рис. 3.7. элемент
«отрицающее ИЛИ-НЕ»

Логические элементы, которые реализуют операции конъюнкции, дизъюнкции, функции Пирса и Шеффера, могут быть, в общем случае, **n**-входовыми. В таблице истинности такого элемента количество возможных комбинаций входных переменных N , в общем случае равняется: $N = 2^n$, где n – число входных переменных.

Логические элементы используются для построения интегральных микросхем, которые выполняют разнообразные логические и арифметические операции.

ФАЛ любой сложности можно реализовать при помощи обозначенных логических элементов. В качестве примера рассмотрим ФАЛ, заданную в алгебраической форме, в виде:

$$Y = X_1 + \overline{X_2} \cdot X_3$$

Для реализации заданной функции на элементах «И-НЕ» используют двойную инверсию функции (теорему де Моргана):

$$Y = \overline{\overline{Y}} = \overline{\overline{X_1 + \overline{X_2} \cdot X_3}} = \overline{\overline{X_1} \cdot \overline{\overline{X_2} \cdot X_3}}$$

Реализация проекта цифровой схемы в графическом редакторе САПР QUARTUS II.

Рассмотрим работу с графическим редактором САПР QUARTUS II на примере схемы, представленной на рис. 3.8.

$$OUT = ((AB + CD) + AD) + BD$$

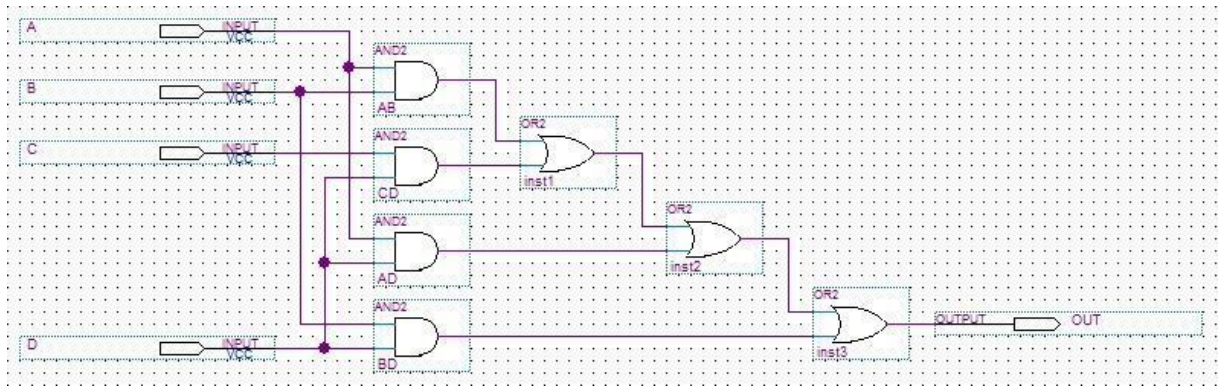


Рис. 3.8. Пример цифровой схемы

Запустив САПР QUARTUS II, создаем новый проект как указано в главе **Создание нового проекта**.

Далее при помощи меню File/New... создаем файл графического редактора (рис. 3.9).

В созданный файл вводим схему лабораторной работы. Для ввода элементов схемы воспользуемся, меню Enter Symbol (ввод символа), которое вызывается двойным щелчком левой кнопки мыши. В открывшемся окне (рис. 3.10) выбираем необходимую библиотеку примитивов (Иерархическое дерево также представлено на рисунке 3.10). Выбираем нужный элемент, нажимаем ОК и размещаем в нужном месте на рабочем поле.

После окончания ввода схемы сохраняем файл в предварительно созданную папку разрабатываемого проекта. Через меню File/Save As... (Рис. 2.11) сохраняем схему под выбранным именем, при этом расширение присваивается автоматически.

Если все сделано было правильно, то файл автоматически будет привязан к ранее созданному проекту

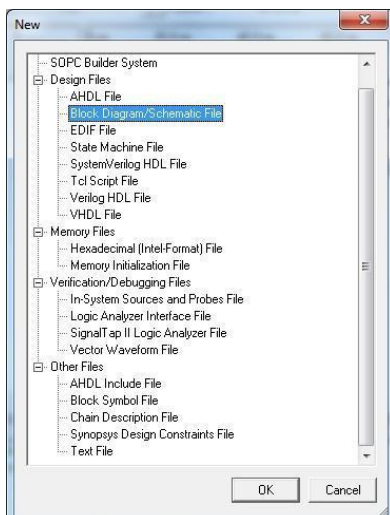


Рис. 3.9. Меню File/New...

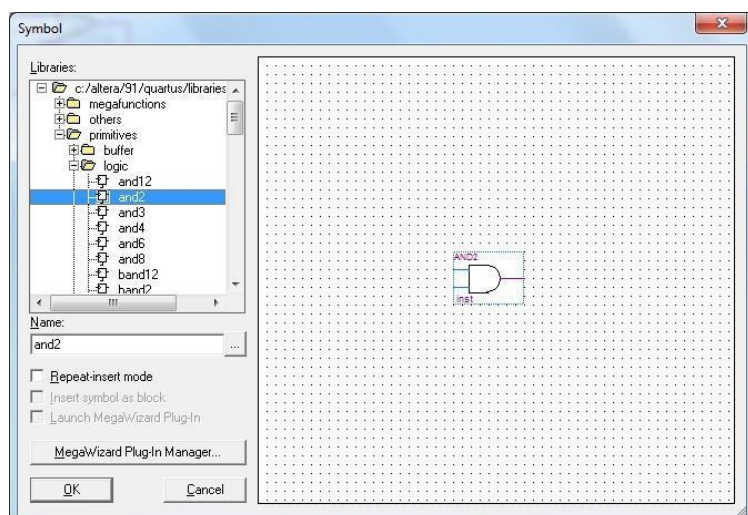


Рис. 3.10. Меню Enter Symbol

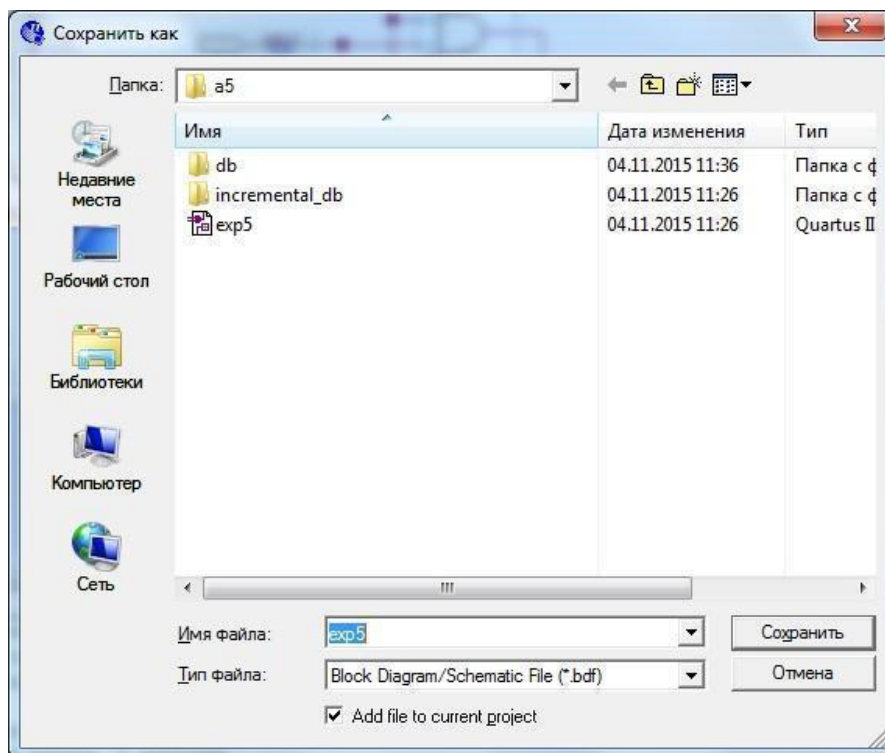


Рис. 3.11. Меню Save As...

Далее следует провести компиляцию проекта. Основные кнопки компилятора вынесены на панель задач (рис. 3.12).


В ней присутствуют пункты меню (слева на право).

- Остановка работы компилятора
- Запуск полной компиляции проекта (Также создает в папке проекта все необходимые файлы отчета, оптимизирует работу проекта)
- Компиляция «Анализ и синтаксис»
- Временная компиляция
- Начать работу Анализатора времени
- Анализатор времени
- Начать симуляцию (для временных диаграмм)
- Показать отчет о компиляции
- Меню программера

Более подробно о работе всех этих функций можно прочитать в главе **Компиляция проекта.**



Рис. 3.12 Кнопки компилятора на панели задач

На данный момент нам нужна кнопка *start Compilation* . Если не был назначен тип ПЛИС для компиляции проекта то система сама назначает подходящий тип микросхемы.

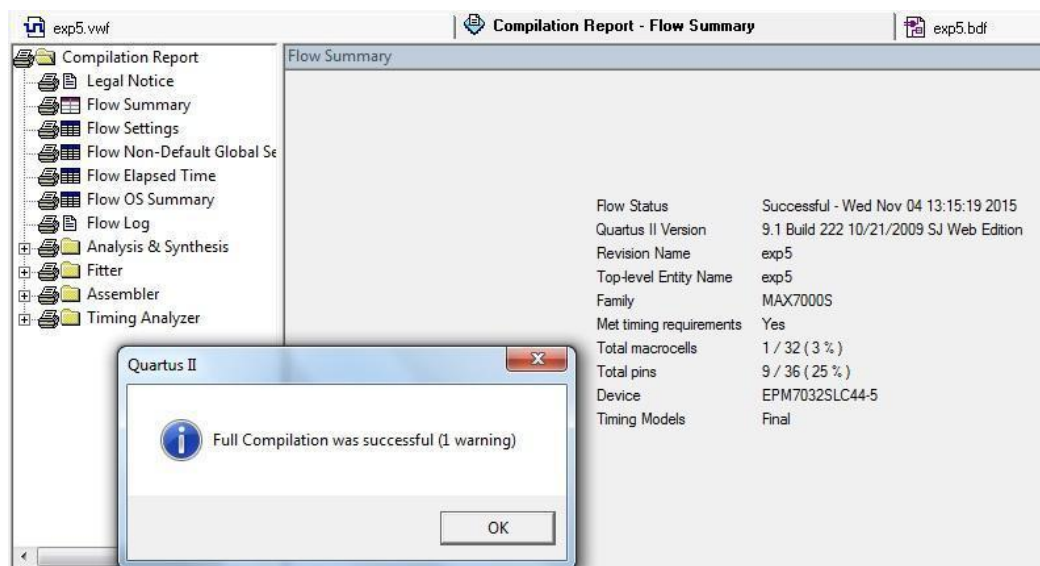


Рис. 3.13. Окно отчета компилятора

Также можно запускать отдельные пункты компиляции для экономии времени с помощью окна **Tasks** (Рис.3.14) путем двойного клика по нужным вариантам компиляции.

Task <input checked="" type="checkbox"/>	Time
Compile Design	
Analysis & Synthesis	00:00:04
Fitter (Place & Route)	00:00:03
Assembler (Generate programming files)	
Classic Timing Analysis	00:00:03
EDA Netlist Writer	
Program Device (Open Programmer)	

Рис. 3.14 Окно Tasks

Если все правильно сделано, то компилятор выведет окно с сообщением об успешной компиляции (рис.3.13) и отчет о совершенной работе

Чтобы посмотреть, как работает созданный проект лучше всего воспользоваться сигнальным редактором. Подробно он описан в главе введения «Основные редакторы для ввода/вывода проекта»

Перед тем как запустить сигнальный редактор, нужно создать специальный *netlist* для функциональной симуляции. Выбираем пункт меню *Processing\Generate Functional Simulation Netlist*.

Вообще симуляция бывает временная и функциональная.

Функциональная симуляция позволяет проверить именно логику работы. С ее помощью мы можем убедиться, что проект может и должен работать так как задумано. Прежде всего нужно делать именно функциональную симуляцию.

Временная симуляция позволяет увидеть сигналы с учетом всех возникающих задержек сигнала между элементами, входами и выходами. Временную симуляцию делают в последнюю очередь, уже после

функциональной симуляции, чтобы убедиться, что схема может работать на нужной заданной частоте.

Для небольших проектов, вполне достаточно делать только функциональную симуляцию. Ее чип CPLD весьма быстр, а внутренняя тактовая частота всего 5МГц. Так что с вероятностью 99,9% можно сказать, что если функциональная симуляция пройдена, то проект будет работать. Чтобы провести Функциональную симуляцию необходимо зайти в меню Assignments/Settings и в подменю Simulator Settings (рис. 3.15) выбрать в пункте Simulation mode режим Functional.

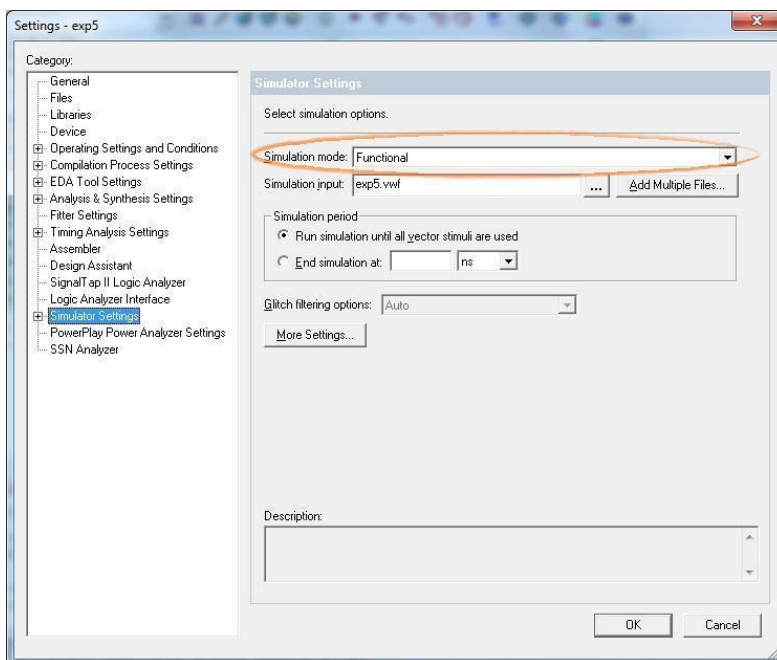


Рис. 3.15. Окно Simulation Settings.

После создания *Netlist* можно приступить к проверке работы схемы. Делаем это с помощью сигнального редактора. Для этого открываем сигнальный редактор (File/New...) на рисунке 1.3 пункт меню «Vector Waveform File» создаем (и сохраняем File/Save As...) файл с расширением .scf.

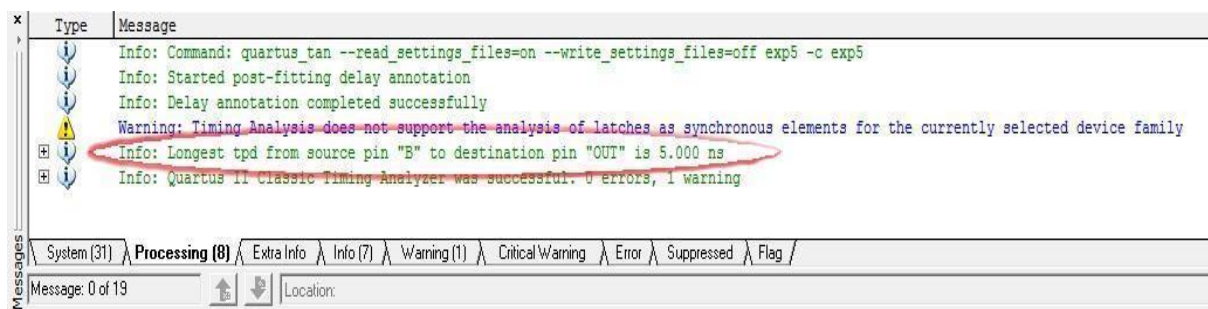


Рис. 3.16. Время прохода сигнала по максимально длинному маршруту с указанием этого маршрута

После Временной компиляции Quartus II в поле Messages выведет

сообщение о самом длинном маршруте и времени прохода сигнала по нему (рис. 3.16) Это нам необходимо для определения минимального времени отклика на изменения состояния сигналов.

В созданном файле при помощи меню Edit/Grid Size... шаг сетки моделирования (он должен быть больше минимального времени отклика). Далее двойным щелчком правой кнопки мыши на поле Name: вызываем меню Insert Node or bus (рис. 3.17), при помощи которого выбираем входы и выход схемы. Для более простого добавления лучше воспользоваться поиском (кнопка Node Finder). В открывшемся окне (Рис 3.18) можно найти необходимые пины, воспользовавшись фильтром или по маске и добавить их на Диаграмму.

Для входных выводов задаем их значения на протяжении необходимого времени моделирования. После того как входные значения заданы осталось только задать время окончания моделирования. Для этого идем в меню Edit/End Time... и задаем время окончания моделирования.

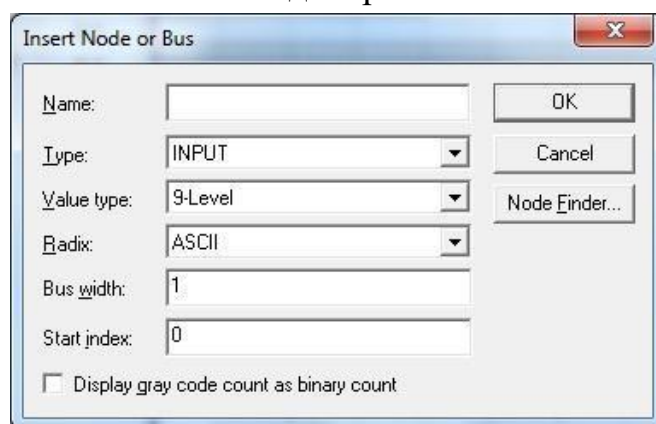


Рис. 3.17. Меню Insert Node or bus

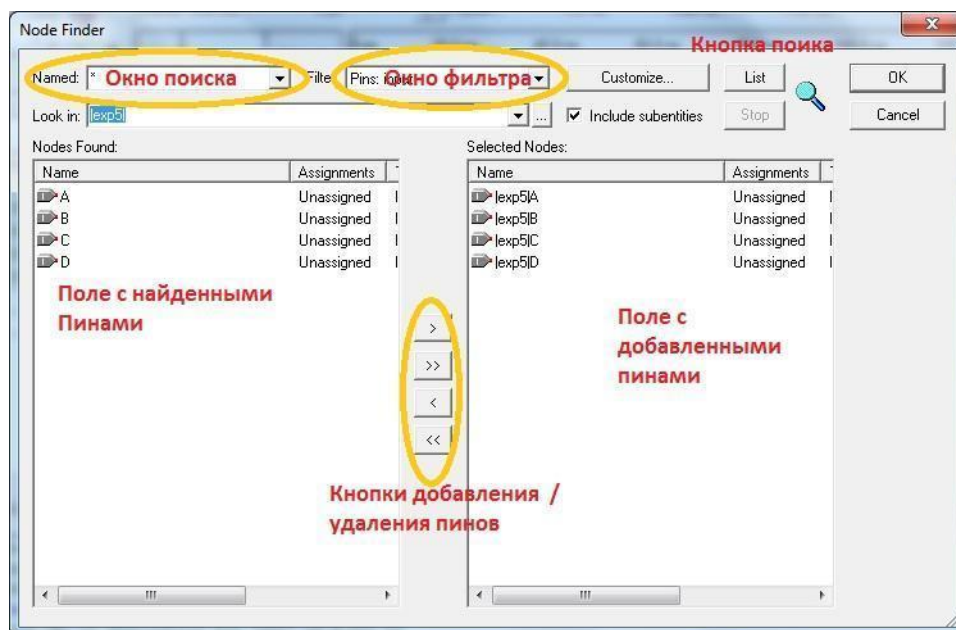


Рис. 3.18 Окно Node Finder

Результаты моделирования работы схемы лабораторной работы

представлены на рис. 3.19.

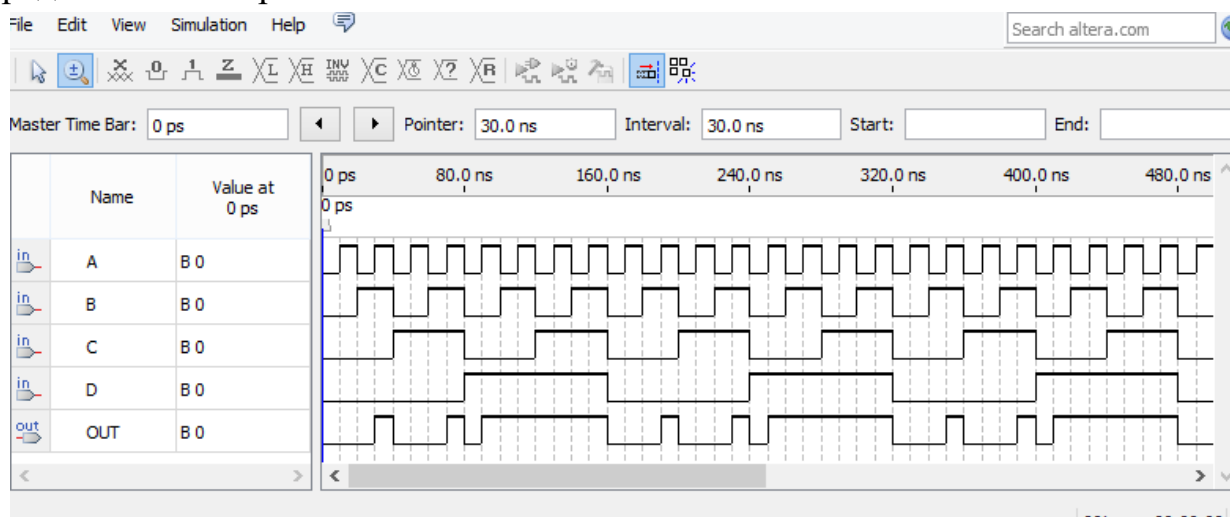


Рис. 3.19. Результаты моделирования работы схемы в сигнальном редакторе

Задание к выполнению лабораторной работы:

1. Изучить правила построения, принцип работы логических схем.
2. Синтезировать электрическую принципиальную схему логического устройства, описанного заданным преподавателем уравнением в алгебраической форме.
3. Нарисовать синтезированную схему в графическом редакторе САПР QUARTUS II.
4. Произвести симуляцию работы схемы. Зарисовать диаграммы работы и по ее результатам заполнить таблицу истинности смоделированной схемы.
5. Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы

1. Назовите основные логические (булевы) функции и изобразите элементы их реализующие. Для каждой из функции запишите таблицу истинности.
2. Какие логические элементы доступны в библиотеке примитивов графического редактора MAX+PLUS II?
3. Какие процессы протекают в системе при компиляции проекта?
4. Объясните результаты моделирования работы схемы лабораторной работы.

Варианты заданий для выполнения лабораторной работы №1

№ варианта	Задание
1	$Y = A+B+CD+AD+BD$
2	$Y = AB+CD+AB \oplus BD$
3	$Y = AD+\bar{C}+BD+\bar{D} \oplus ACB$
4	$Y = (AB+CD)AD+BD$
5	$Y = (A+D) \sim BC$
6	$Y = \bar{C}D+B(C \oplus A)$
7	$Y = (\bar{B}\bar{D}+CA) \oplus D$
8	$Y = B \oplus AC + \bar{A}\bar{D} + \bar{B}C$
9	$Y = CD(A+B)+AB(C+D)$
10	$Y = A\bar{C} + A\bar{B} + AC\bar{D} + \bar{B}\bar{C}$
11	Сумматор
12	Шифратор
13	Дешифратор
14	Мультиплексор
15	Компаратор

Лабораторная работа №2

Описание логических схем при помощи языка AHDL

Цель работы:

Приобретение основных навыков описания цифровых схем с помощью языка описания аппаратуры AHDL. Смоделировать логическую схему при помощи текстового редактора САПР QUARTUS II.

Основные теоретические сведения:

Язык описания аппаратуры AHDL разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС QUARTUS II. Файлы описания аппаратуры, написанные на языке AHDL, имеют расширение *.TDF (Text design file). Для создания TDF-файла можно использовать как текстовый редактор системы QUARTUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera.

Операторы и элементы языка AHDL являются достаточно мощным и универсальным средством описания алгоритмов функционирования цифровых устройств, удобным в использовании. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TDF-файлы, разработанные на языке AHDL, так и другие типы файлов.

При распределении ресурсов устройств разработчик может пользоваться командами текстового редактора или операторами языка AHDL для того, чтобы сделать назначения ресурсов и устройств. Кроме того, разработчик может только проверить синтаксис или выполнить полную компиляцию для отладки и запуска проекта. Любые ошибки автоматически обнаруживаются обработчиком сообщений и высвечиваются в окне текстового редактора.

Элементы языка AHDL. Резервированные ключевые слова.

Резервированные ключевые слова используются для следующих целей:

- для обозначения начала, конца и переходов в объявлениях языка AHDL;
- для обозначения predetermined констант, т.е. GND и VCC.

Ключевые слова можно использовать, как символические имена, только если они заключены в символы одинарных кавычках ('). Их можно также использовать в комментариях.

Altera рекомендует все ключевые слова набирать прописными буквами.

Список всех зарезервированных ключевых слов (Keywords) и идентификаторов (Identifiers) языка AHDL приведен в табл. 4.1 и табл. 4.2 соответственно:

Таблица 4.1

AND	ELSE	MOD	STATES
ASSERT	ELSIF	NAND	SUBDESIGN
BEGIN	END	NODE	TABLE
BIDIR	FOR	NOR	THEN
BITS	FUNCTION	NOT	TITLE
BURIED	GENERATE	OF	TO
CASE	GND	OPTIONS	TRI_STATE_NODE
CLIQUE	HELP_ID	OR	VARIABLE
CONNECTED_PINS	IF	OTHERS	VCC
CONSTANT	INCLUDE	OUTPUT	WHEN
DEFAULTS	INPUT	PARAMETERS	WITH
DEFINE	IS	REPORT	XNOR
DESIGN	LOG2	RETURNS	XOR
DEVICE	MACHINE	SEGMENTS	
DIV	MACRO	SEVERITY	

Таблица 4.2

CARRY	FLOOR	MCELL	TFFE
CASCADE	GLOBAL	MEMORY	TFF
CEIL	JKFFE	OPENDRN	TRI
DFFE	JKFF	SOFT	USED
DFF	LATCH	SRFFE	WIRE
EXP	LCELL	SRFF	X

Символы

Ниже в таблице 4.3 приведены символы, имеющие определенное значение

в языке AHDL. В этот перечень не включены символы, используемые в булевых выражениях как операторы и для операций сравнения.

Таблица 4.3

Символ	Функция
_ (подчеркивание)	Используемые пользователем идентификаторы
- (тире)	символы в символических именах
-- (два тире)	Начинает комментарий в стиле VHDL, который продолжается до конца строки
% (процент)	Заключает с двух сторон комментарий стиля AHDL
() (круглые скобки)	Закljučают и определяют последовательные имена групп. Закljučают имена выводов в секции подпроекта (Subdesign Section) и в прототипах функций. Закljučают (необязательно) входы и выходы таблиц в объявлении Truth Table. Закljučают состояния в объявлении цифрового автомата State Machine. Закljučают более приоритетные операции в булевых выражениях. Закljučают необязательные варианты в секции проекта Design Section (внутри объявления назначения ресурсов Assignment).
[] (квадратные скобки)	Закljučают диапазон значений в десятичном имени группы
'...' (одинарные кавычки)	Закljučают символические имена.
"..." (двойные кавычки)	Закljučают строку в объявлении названия Title Закljučают цифры в не десятичных номерах Закljučают путь в объявлении Include. Могут (необязательно) заключать имя проекта и устройства в секции проекта Design Section. Могут (необязательно) заключать имя в объявлении назначения клики графа Clique Assignment.
. (точка)	Отделяет символические имена переменных в макрофункции или примитиве от имен портов. Отделяет имя файла от расширения
... (многоточие)	Разделяет наименьшее и наибольшее значение в диапазонах

; (точка с запятой)	Заканчивает объявления и секции в языке AHDL
, (запятая)	Разделяет элементы последовательных групп и списков
: (двоеточие)	Отделяет символические имена от типов в объявлениях и назначениях ресурсов.
@ “собака”	Присваивает символические узлы вывода устройства и логическим ячейкам в объявлениях назначения ресурсов Resource Assignment
= (равенство)	Присваивает значения по умолчанию GND и VCC входам в секции подпроекта Subdesign. Присваивает установочные значения в вариантах. Присваивает значения состояниям в машине состояний. Присваивает значения в булевых уравнениях.
=> (стрелка)	Отделяет входы от выходов в объявлениях таблицы истинности Truth Table. Отделяет предложения с WHEN от булевых выражений в операторе Case.

Имена в кавычках и без кавычек

В языке AHDL есть три типа имен:

Символические имена – это определяемые пользователем идентификаторы. Они используются для обозначения следующих частей TDF:

- внутренних и внешних узлов (вершин);
- констант;
- переменных цифрового автомата, битов состояний, имен состояний;
- примеров (Instance).

Имена подпроекта(модуля) - это определяемые пользователем имена для файлов проекта более низкого уровня. Имя подпроекта должно быть таким же, как имя файла TDF.

Имена портов - это символические имена, идентифицирующие вход или выход примитива или макрофункции.

В файле .fit проекта могут появиться генерируемые компилятором имена выводов, с символом “тильда” (~). Этот символ зарезервирован для имен,

генерируемых компилятором, пользователю запрещается его использовать для обозначения имен выводов, узлов (вершин), групп (шин). Существуют две формы записи для всех трех типов имен (символических, подпроекта и портов): *в кавычках (')* и *без кавычек*.

Если разработчик создает символ по умолчанию для файла TDF, который включает в себя имена портов в кавычках, собственно кавычки не входят в имена выводов.

Числа в языке AHDL

В языке AHDL можно использовать десятичные, двоичные, восьмеричные и шестнадцатеричные числа в любой комбинации. В таблице 4.4 приведен синтаксис записи чисел в языке AHDL для каждой системы счисления.

Таблица 4.4

Система счисления	Значения
Десятичная	<последовательность цифр 0–9>
Двоичная	B"<последовательность из 0, 1, X>", где символ X обозначает безразличное значение
Восьмеричная	O"< последовательность цифр 0–7>" или Q"< последовательность цифр 0–7>"
Шестнадцатеричная	X"< последовательность цифр 0–9, букв A–F>" или H"< последовательность цифр 0–9, букв A–F>"

Булевы выражения

Булевы выражения состоят из операндов, разделенных логическими и арифметическими операторами и компараторами и (необязательно) сгруппированных с помощью круглых скобок. Выражения используются в булевых уравнениях, а также в других конструкциях языка, таких как операторы Case и If.

Существуют следующие применения булевых выражений:

- Операнд.

Пример: a, b[5..1], 7, VCC

- Встроенная в текст (in-line) ссылка (reference) на примитив или макрофункцию.
- Префиксный оператор (! или -), примененный к булеву выражению.

Пример: !c

- Два булевых выражения, разделенные двоичным (не префиксным) оператором.

Пример: d1 \$ d3

- Заключенное в круглые скобки булево выражение.

Пример: (!d1 & d3)

Результат каждого булева выражения должен иметь ту же ширину, что и узел или группа (в левой стороне уравнения), которому он, в конечном счете, присваивается.

Логические операторы

В таблице 4.5 приведены логические операторы для булевых выражений.

Таблица 4.5

Оператор:	Пример:	Описание:
!	!x	Инверсия (префиксное обращение)
NOT	NOT x	
&	x1 & x2	Логическое И
AND	x1 AND x2	
!&	a[3..1] !& b[5..3]	И-НЕ
NAND	a[3..1] NAND b[5..3]	
#	x1 # x2	Логическое ИЛИ
OR	x1 OR x2	
!#	c[8..5] !# d[7..4]	ИЛИ-НЕ
NOR	c[8..5] NOR d[7..4]	
\$	x1 \$ x2	Исключающее ИЛИ
XOR	x1 XOR x2	
!\$	x2 !\$ x4	Инверсия исключающего ИЛИ
XNOR	x2 XNOR x4	

Каждый оператор представляет собой логический вентиль с двумя входами; исключение составляет оператор NOT, являющийся префиксным инвертором. Для записи логического оператора можно использовать его имя или символ.

Выражения, в которых используются эти операторы, интерпретируются по-разному в зависимости оттого, что представляют собой операнды: одиночные узлы (вершины), группы или числа. Кроме того, выражения с оператором NOT интерпретируются не так как другие логические операторы.

Реализация проекта цифровой схемы в текстовом редакторе САПР QUARTUS

Рассмотрим работу с текстовым редактором САПР QUARTUS II на примере схемы демультиплексора.

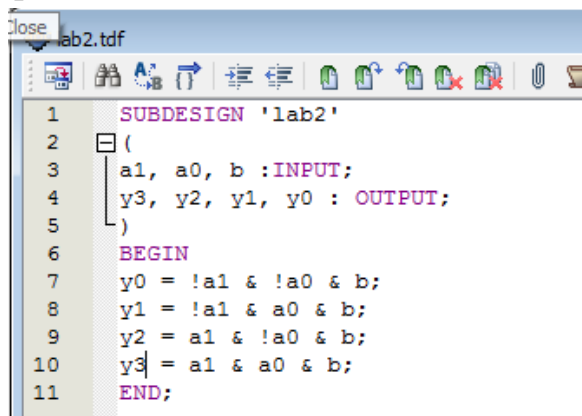
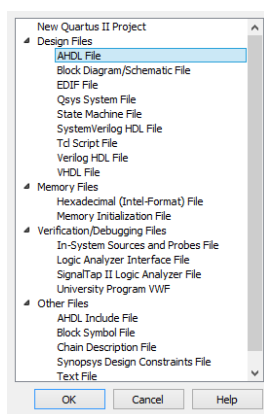


Рис. 4.1. Меню File/New... Рис. 4.2. Описание демультиплексора на языке AHDL

В созданный файл вводим описание демультиплексора на языке AHDL. Обратите внимание имя файла lab2.tdf должно совпадать с названием программы SUBDESIGN 'lab2'.

После окончания ввода, сохраняем файл в предварительно созданную средствами Windows папку разрабатываемого проекта (например: \lab2), Через меню File/Save As... (Рис. 4.3) сохраняем программу под выбранным именем (например: lab2), при этом расширение присваивается автоматически. После сохранения необходимо имя файла привязать к имени проекта. (Рис. 4.4).

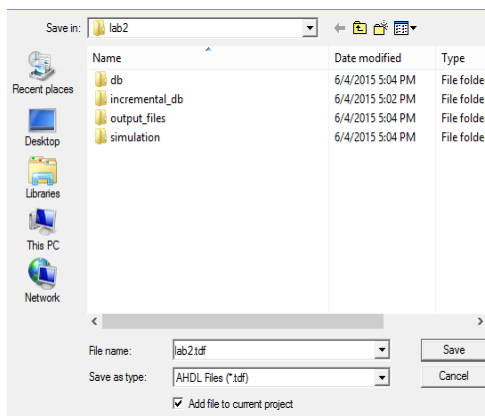


Рис. 4.3. Меню Save As...

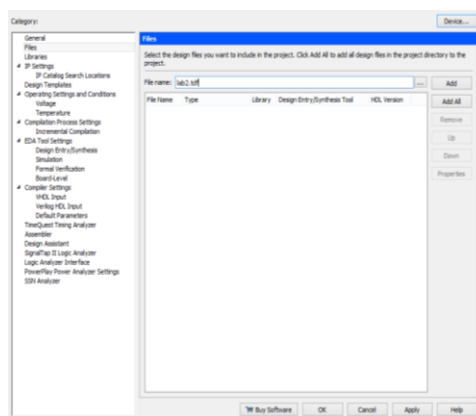


Рис. 4.4. Меню Settings

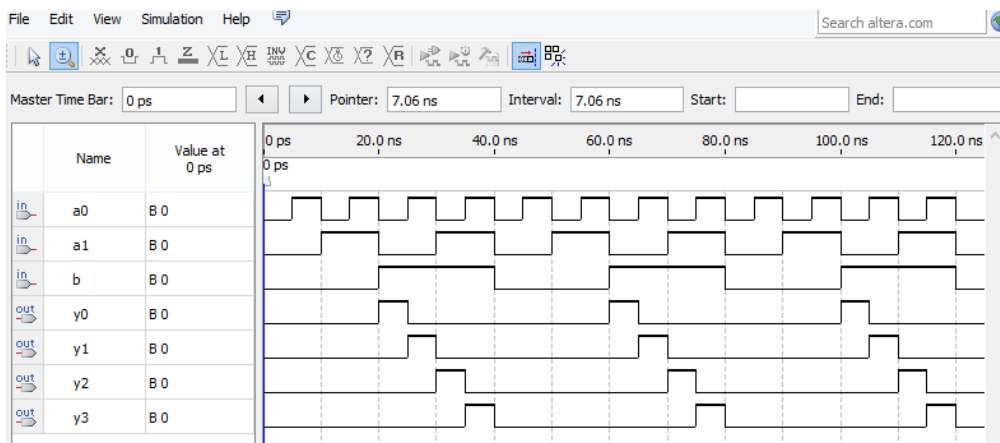


Рис. 4.5. Результаты моделирования работы программы в сигнальном редакторе

Задание к выполнению лабораторной работы:

1. Изучить основные элементы языка AHDL и правила описания логических схем.
2. Сделать описание электрической схемы заданной в предыдущей работе при помощи текстового редактора САПР QUARTUS II.
3. Произвести симуляцию работы схемы. Зарисовать диаграммы работы и по ее результатам заполнить таблицу истинности смоделированной схемы.
4. Сравнить результаты, полученные в ходе выполнения лабораторной работы с результатами, полученными в работе №1.
5. Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы

1. Что такое язык описания аппаратуры? Назовите существующие языки описания аппаратуры, в чем их отличие?
2. Назовите основные элементы языка AHDL, дайте их краткую характеристику.
3. Как описываются логические элементы в AHDL?

Лабораторная работа №3

Моделирование цифровых схем с использованием параметрических элементов

Цель работы:

Приобретение навыков использования параметрических элементов (LPM function) в САПР QUARTUS II, экспериментальное исследование счетчиков и регистров, построенных на их основе.

Основные теоретические сведения:

Счетчики и регистры Регистры и счетчики относятся к разряду цифровых устройств и являются одним из наиболее распространенных элементов вычислительной техники. Они широко используются для построения устройств ввода, вывода и хранения информации, а также для выполнения некоторых арифметических и логических операций.

Для построения счетчиков и регистров используются синхронные триггеры, переключение которых происходит только при наличии синхронизирующего сигнала (синхроимпульса) на входе С. Наиболее часто для построения регистров и счетчиков используется D-триггер, имеющий специальный информационный вход D, и динамический вход С (рис.5.1).

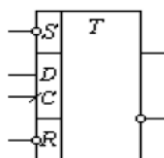


Рис. 5.1. D – триггер

Устройство, называемое счетчиком, предназначено для подсчета числа поступающих на вход сигналов (импульсов) в произвольной системе счисления. Двоичные счетчики строятся на основе триггеров, работающих в счетном режиме (Т - триггер или счетный триггер).

Счетный триггер может быть получен из универсального D - триггера путем соединения его инверсного выхода Q со входом D.

Счетный триггер и эюры сигналов, поясняющие его работу, представлены на рис.5.2.

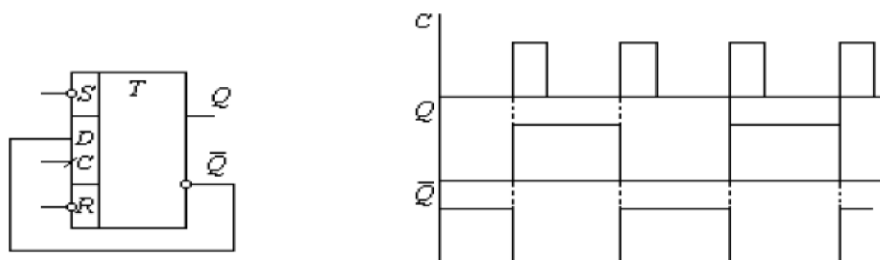


Рис. 5.2. Счетный триггер и его работа

У счетного триггера состояние выхода изменяется на противоположное при поступлении на вход C каждого очередного счетного импульса.

Функциональная схема и условное графическое обозначение двоичного счетчика с коэффициентом пересчета 2^3 представлена на рис. 5.3.

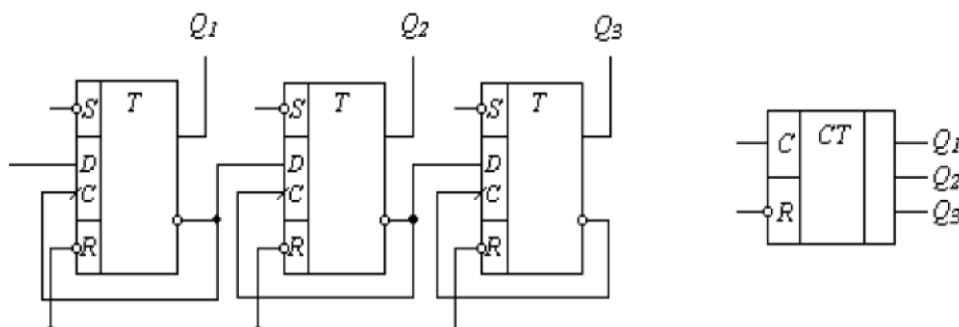


Рис. 5.3. Двоичный счетчик

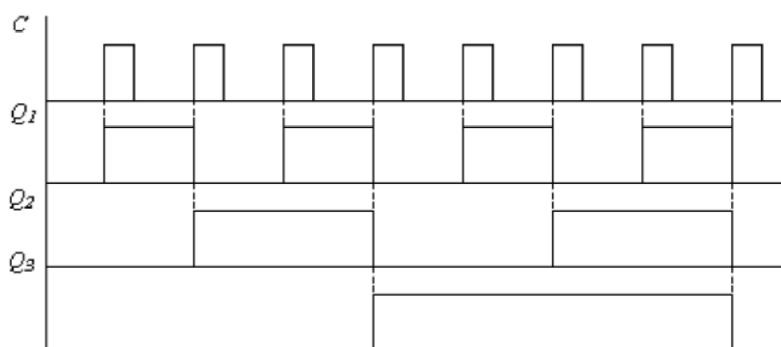


Рис. 5.4. Диаграммы работы двоичного счетчика

Каждый поступающий на вход счетчика импульс перебрасывает первый триггер в противоположное состояние (рис. 5.4). Сигнал с инверсного выхода предыдущего триггера является входным сигналом для последующего и, таким образом, комбинация сигналов на выходах Q1, Q2, Q3 будет соответствовать числу поступивших на вход счетчика импульсов, представленному в двоичном коде. Счетчик данного типа называется асинхронным счетчиком.

Если на счетный вход каждого последующего триггера счетчика подавать сигнал с прямого выхода предыдущего триггера, то счетчик будет производить операцию вычитания. Счетчики, способные выполнять функции сложения и вычитания, называются реверсивными.

Для построения счетчика с требуемым коэффициентом пересчета K_c , отличным от величины $2N$ (N - число двоичных разрядов счетчика), используется принудительный сброс счетчика в исходное состояние при достижении счетчиком числа K_c .

Устройство, называемое регистром, служит в основном для хранения чисел в двоичном коде при выполнении над ними различных арифметических и логических операций. С помощью регистров выполняются такие действия над числами, как передача их из одного устройства в другое, арифметический и логический сдвиг в сторону младших или старших разрядов, преобразование кода из последовательного в параллельный и наоборот и т.д.

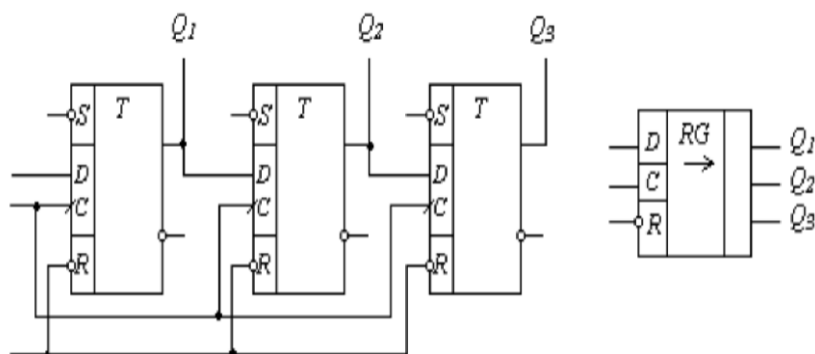


Рис. 5.5. Регистр сдвига

Функциональная схема и условно-графическое обозначение регистра сдвига представлены на рис. 5.5.

Последовательный информационный код поступит на вход D регистра. Импульс команды сдвига C подается одновременно на синхронизирующие входы всех триггеров регистра и переводит каждый триггер в состояние, в котором находился триггер предыдущего разряда. Таким образом, каждый импульс команды сдвига "продвигает" записываемое число на один разряд вправо.

При введении обратной связи в регистр сдвига, последний превращается в замкнутое кольцо, в котором под воздействием тактовых импульсов циркулирует введенная в регистр информация. Такие регистры называют кольцевыми счетчиками. Кодовая единица, введенная в первый триггер, циркулирует в течении всего времени существования тактовых импульсов, подаваемых на входы C всех триггеров счетчика. Приходящий тактовый импульс перебрасывает триггер, который был в состоянии 1, в состояние 0. Поскольку

выход Q этого триггера связан с входом D следующего триггера, то последний устанавливается в состояние 1 и т.д. Количество состояний такого счетчика равно числу триггеров.

Реализация проекта на параметрических элементах

Применение параметрических элементов САПР QUARTUS II в разработке проектов цифровых схем рассмотрим на примере реализации реверсивного счетчика разрядностью 4.

Создаем новый файл графического редактора и сохраняем его под определенным именем (например: lab3) в предварительно созданном каталоге \lab3. Двойным щелчком правой кнопки мыши открываем меню ввода символов (Symbol), выбираем библиотеку megafunctions/arithmetic и в ней выбираем lpm_counter. Для редактирования параметров и входов/выходов счетчика необходимо нажать правую кнопку мыши, выбрать меню Properties, откроется окно Symbol Properties. Во вкладке Ports(рис 5.6.1) выбрать необходимые входы/выходы счетчика, а во вкладке Parameter(рис 5.6.2) задать разрядность LPM_WIDTH и направление счета LPM_DIRECTION (в данном примере: вычитание).

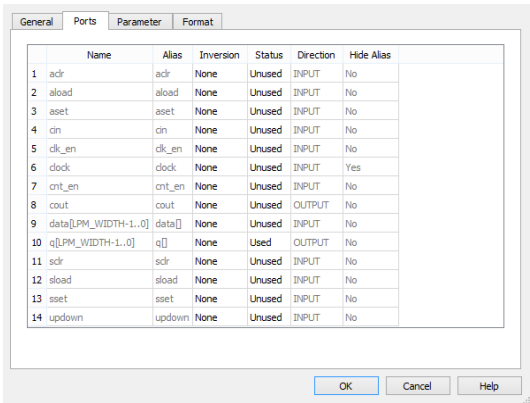


Рис 5.6.1 Вкладка Ports

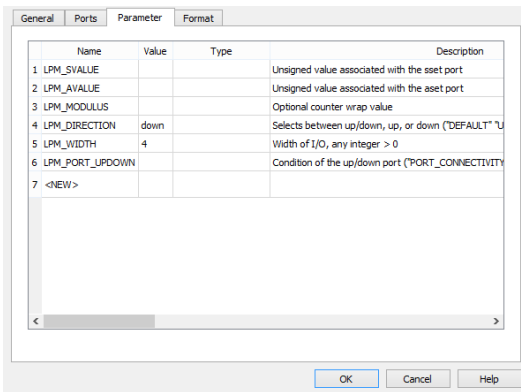


Рис 5.6.2 Вкладка Parameter

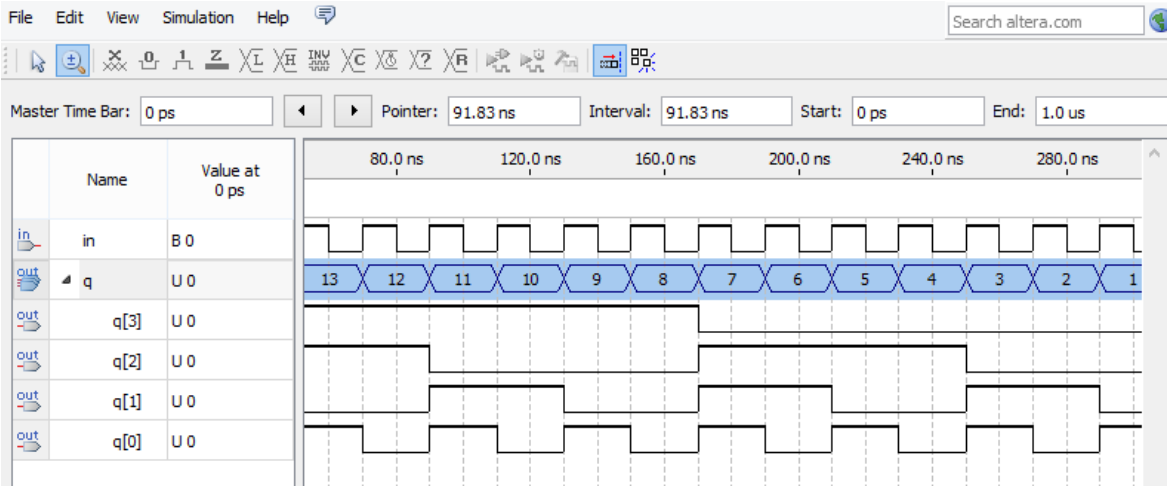


Рис. 5.7. Результат моделирования работы счетчика

Далее располагаем входные и выходные выводы схемы проекта. Когда схема создана, делаем проверку на предмет наличия ошибок ввода схемы, для чего запускаем компилятор.

Если компиляция прошла успешно, создаем файл симулятора для анализа работы счетчика. В созданном файле задаем входной (in) периодический сигнал с периодом следования импульсов в 20 нс. Сохраняем файл и запускаем симулятор. Результатом симуляции будет диаграммы работы счетчика, приведенные на рис. 5.7.

Описание некоторых параметрических элементов САПР QUARTUS II представлено в приложении.

Задание к выполнению лабораторной работы:

1. Изучить правила построения и принцип работы триггеров и построение на их основе логических схем.
2. Нарисовать электрическую схему по указанию преподавателя при помощи графического редактора САПР QUARTUS II.
3. Произвести симуляцию работы схемы, зарисовать диаграммы работы и по ее результатам заполнить таблицу истинности смоделированной схемы.
4. Спроектировать эту же электрическую схему, но с использованием параметрических элементов САПР QUARTUS II, проверить ее работу в сигнальном редакторе и оценить временные задержки в схеме.
5. Ответить на контрольные вопросы, оформить отчет о выполненной работе.

Контрольные вопросы

1. Объясните понятие «параметрический элемент». Какие параметрические элементы доступны в САПР QUARTUS II?
2. Объясните принцип работы счетчика построенного на триггерах. Какие типы счетчиков существуют?

3. Объясните назначение пунктов меню Edit Ports/Parameters.

4. Чем ограничивается максимальная скорость работы счетчика? Какова максимальная частота работы счетчика разработанного в ходе выполнения лабораторной работы?

Лабораторная работа №4

Счетчик с произвольным модулем счета

Цель работы:

Ознакомиться с САПР QUARTUS II фирмы Altera, получить практические навыки создания проектов по схемотехнике ЭВМ в САПР (ввод схем, компиляция и моделирование).

Задание:

1. Согласно своему варианту графа состояний автомата разработать функциональную электрическую схему цифрового программируемого устройства преобразования кодов.
2. Включить ЭВМ и запустить САПР QUARTUS II.
3. Создать проект, ввести разработанную схему, откомпилировать и отмоделировать её.
4. Проверить полученные результаты, сверив их с таблицей истинности устройства.

Порядок выполнения работы:

1. Получить № варианта состояний графа устройства (табл. 6.1).
2. На основе исходного графа состояний (рис. 6.1) и согласно своему варианту составить таблицу перекодировки состояний устройства в десятичном и двоичном коде.
3. Подставить новые значения состояний в исходный граф.
4. Составить таблицу истинности работы устройства.
5. По таблице истинности разработать функциональную электрическую схему устройства.

6. Включить ЭВМ.
7. Создать рабочую папку на рабочем столе Windows и дать ей название, совпадающее с фамилией выполняющего работу.
8. Запустить САПР QUARTUS II.
9. Выбрать пункт меню File/New/New Quartus II Project
10. В открывшемся окне в разделе New Project Wizard выбрать созданную папку для нового проекта; в разделе «What is the name of this project?» ввести имя проекта (например, lab4). Затем нажать «Finish».
11. Создать указанный файл проекта (lab4.bdf), выбрав пункт меню File/New/Design Files/Block Diagram...
12. Сохранить созданный файл под тем же именем (lab4.bdf), нажав Ctrl+S, а затем «ОК».
13. Выбрать пункт меню Assignments/Device... В разделе Device Family указать MAX II, выбрать в разделе Available Devices тип ПЛИС, применяемой в проекте – EPM240F100C4.
14. Нажать «ОК» и сохранить файл.
15. Графический ввод схемы. Для ввода элементов логических схем, обозначения входов/выходов производится двойной щелчок левой кнопки мыши.
16. В открывшемся окне можно указать логический элемент двумя способами: либо ввести имя элемента в строке ввода Name, если известно имя, либо выбрать его из библиотеки элементов, указав нужную библиотеку двойным щелчком в списке Libraries. В данном случае это библиотека */primitives, и из списка примитивов Symbol выбрать нужный.
17. Нажать «Ок». При этом выбранный элемент добавляется в схему. Для улучшения «читаемости» схемы элементы можно перетаскивать, вращать и переворачивать по горизонтали или вертикали. Для этого нужно выделить элементы щелчком левой кнопки мыши и затем, щелкнув правой кнопкой, в сплывающем меню выбрать пункт Flip Horizontal, Flip Vertical или Rotate by Degrees.
18. Соединение элементов можно выполнить «вытаскиванием» выводов (подвести курсор к выводу элемента, нажать левую кнопку графического манипулятора (мыши) и не отпуская её провести линию необходимой длины).
19. Добавление текста в схему: установить курсор на входе/выходе элемента, нажать и отпустить левую кнопку мыши, и ввести текст. Если текст расположен над линией («проводом»), то этот текст будет названием данного

«провода». «Провода» с одинаковым текстом считаются соединенными вместе.

20. Компиляция проекта. Компилятор запускается из меню Processing /Start Compilation.

21. Если схема составлена без ошибок, появится сообщение: «Quartus II Full Compilation was successful» (Полная компиляция Quartus II прошла успешно).

22. Создать файл симулятора для анализа работы счетчика по пути New/Verification.../University Program и сохранить файл с расширением vwf (lab4.vwf).

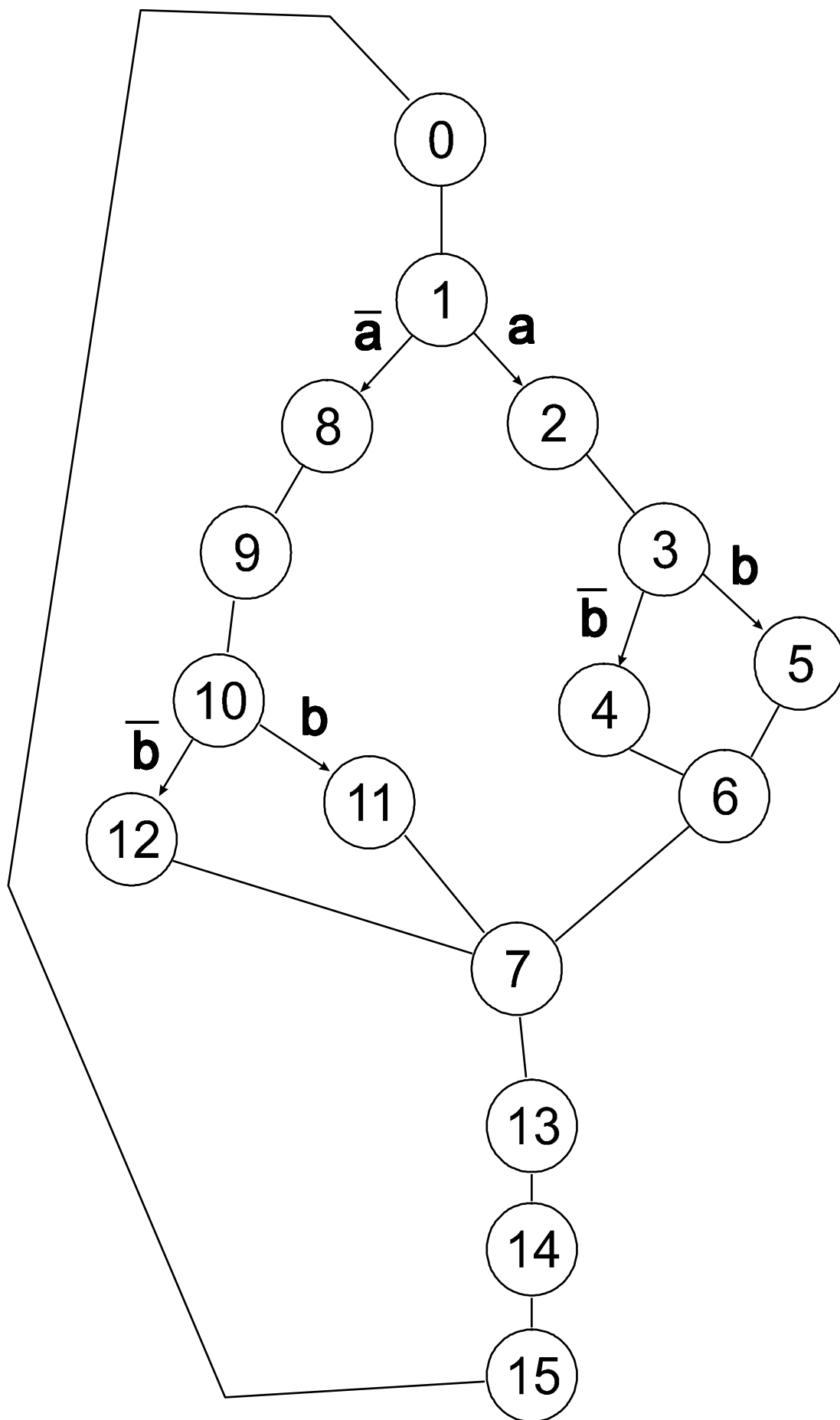


Рис. 6.1. Исходный граф

Таблица 6.1

Варианты состояний графа

№ вар.	Состояния графа															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	3	11	12	1	9	2	14	8	7	6	4	13	10	15	5
2	5	0	3	11	12	1	9	2	14	8	7	6	4	13	10	15
3	15	5	0	3	11	12	1	9	2	14	8	7	6	4	13	10
4	10	15	5	0	3	11	12	1	9	2	14	8	7	6	4	13
5	13	10	15	5	0	3	11	12	1	9	2	14	8	7	6	4
6	4	13	10	15	5	0	3	11	12	1	9	2	14	8	7	6
7	6	4	13	10	15	5	0	3	11	12	1	9	2	14	8	7
8	7	6	4	13	10	15	5	0	3	11	12	1	9	2	14	8
9	8	7	6	4	13	10	15	5	0	3	11	12	1	9	2	14
10	14	8	7	6	4	13	10	15	5	0	3	11	12	1	9	2
11	2	14	8	7	6	4	13	10	15	5	0	3	11	12	1	9
12	9	2	14	8	7	6	4	13	10	15	5	0	3	11	12	1
13	1	9	2	14	8	7	6	4	13	10	15	5	0	3	11	12
14	12	1	9	2	14	8	7	6	4	13	10	15	5	0	3	11
15	11	12	1	9	2	14	8	7	6	4	13	10	15	5	0	3
16	3	11	12	1	9	2	14	8	7	6	4	13	10	15	5	0
17	0	5	15	10	13	4	6	7	8	14	2	9	1	12	11	3
18	3	0	5	15	10	13	4	6	7	8	14	2	9	1	12	11
19	11	3	0	5	15	10	13	4	6	7	8	14	2	9	1	12
20	12	11	3	0	5	15	10	13	4	6	7	8	14	2	9	1
21	1	12	11	3	0	5	15	10	13	4	6	7	8	14	2	9
22	9	1	12	11	3	0	5	15	10	13	4	6	7	8	14	2
23	2	9	1	12	11	3	0	5	15	10	13	4	6	7	8	14
24	14	2	9	1	12	11	3	0	5	15	10	13	4	6	7	8
25	8	14	2	9	1	12	11	3	0	5	15	10	13	4	6	7
26	7	8	14	2	9	1	12	11	3	0	5	15	10	13	4	6
27	6	7	8	14	2	9	1	12	11	3	0	5	15	10	13	4
28	4	6	7	8	14	2	9	1	12	11	3	0	5	15	10	13
29	13	4	6	7	8	14	2	9	1	12	11	3	0	5	15	10
30	10	13	4	6	7	8	14	2	9	1	12	11	3	0	5	15

23. В окне «Insert Node or Bus» в подменю Node Finder в графе Named ввести названия входов и выходов схемы, для которых необходимо построить временные диаграммы. Для этого необходимо нажать кнопку List и выбрать в меню Nodes Found выбрать необходимые входы/выходы.

24. Зайти в меню Edit/Set End Time... и ввести время окончания построения временной диаграммы = 4 мкс.

25. Подать синхросигнал и установить необходимое состояние входов А и В на временной диаграмме.

26. В окне Simulation нажать «Run Functional Simulations»

27. В окне “Waveform editor” появятся временные диаграммы работы устройства. Если последовательность на шине q[3..0] совпадает с заданной, то задание выполнено правильно.

Пример выполнения Варианта №1

1. Составим таблицу перекодировки состояний автомата и их двоичный код (таблица 6.2).

2. Подставляем новые значения в граф состояний (рис.6.2).

3. Составим таблицу истинности автомата (таблица 6.3).

4. После получения таблицы истинности автомата создается функциональная схема в САПР QUARTUSII без минимизации (рис.6.3).

5. Временные диаграммы для этого варианта после компиляции проекта приведены на рис.6.4.

Таблица 6.2

Таблица перекодировки состояний автомата и их двоичный код

№ состояния	№ состояния из табл.1	Двоичный код q3,q2,q1,q0
0	0	0000
1	3	0011
2	11	1011
3	12	1100
4	1	0001
5	9	1001
6	2	0010
7	14	1110
8	8	1000
9	7	0111
10	6	0110

11	4	0100
12	13	1101
13	10	1010
14	15	1111
15	5	0101

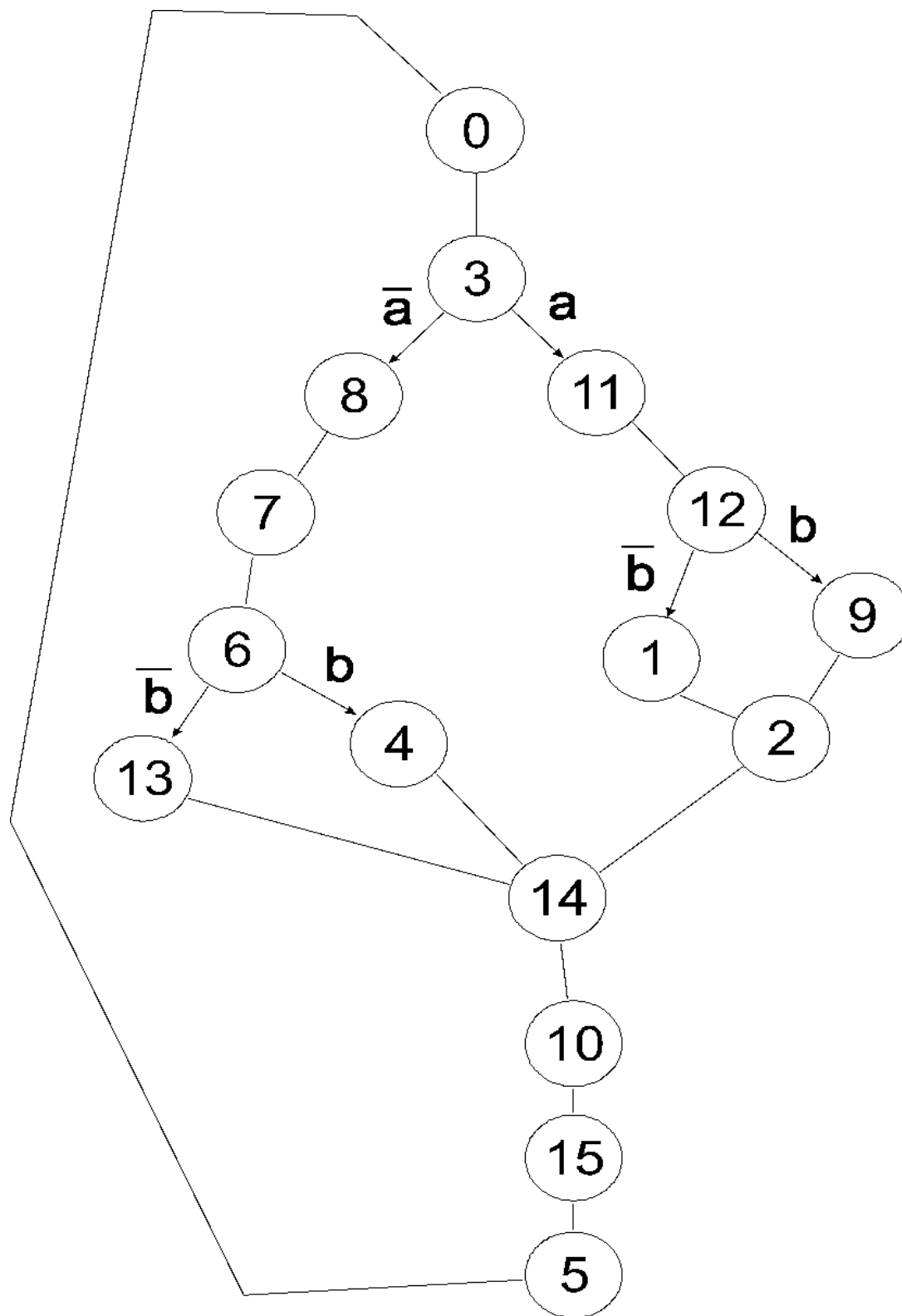
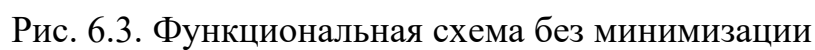


Рис.6.2. Граф полученный с учетом таблицы перекодировки

Таблица истинности автомата

старое состояние		условие	новое состояние	
№	код		№	Код
0	0000	-	3	0011
3	0011	A=0	8	1000
3	0011	A=1	11	1011
8	1000	-	7	0111
7	0111	-	6	0110
6	0110	B=0	13	1101
6	0110	B=1	4	0100
13	1101	-	14	1110
14	1110	-	10	1010
10	1010	-	15	1111
15	1111	-	5	0101
5	0101	-	0	0000
11	1011	-	12	1100
12	1100	B=0	1	0001
12	1100	B=1	9	1001
1	0001	-	2	0010
2	0010	-	14	1110
4	0100	-	14	1110
9	1001	-	2	0010



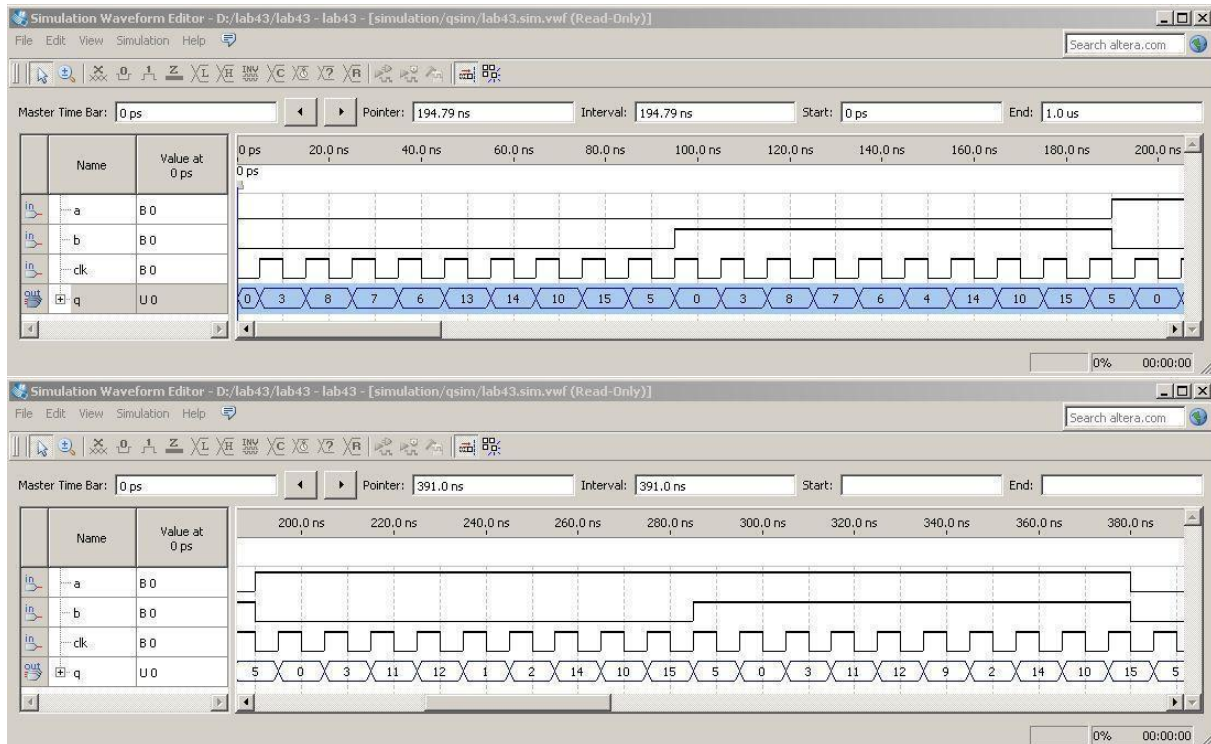


Рис. 6.4. Временные диаграммы

Параметрические элементы САПР QUARTUSII

Counter

<i>Входные выводы</i>	
Имя вывода	Описание
data []	Параллельный вход данных счетчика
clock	Вход счетных импульсов
clk_en	Разрешение синхронизации.
cnt_en	Разрешение счета
updown	Управление направлением счета (1 = сложение, 0 = вычитание)
aclr	Асинхронный сброс входов
aset	Асинхронная установка входов
aload	Асинхронная загрузка входов. Установка счетчика в значение data[].
sclr	Синхронный сброс входов. Сброс счетчика следующим тактовым импульсом
sset	Синхронная установка входов. Установка счета следующим тактовым импульсом.
sload	Синхронная загрузка входов. Загрузка в счетчик значения data[] следующим тактовым импульсом.

<i>Выходные выводы</i>	
Имя вывода	Описание
q []	Выход счетчика
eq [15..0]	Декодированный выход счетчика. Высокий активный уровень появляется в момент, когда счетчик достигает заданного значения.
cout	Перенос в старший разряд

<i>Параметры</i>	
Параметр	Описание
LPM_WIDTH	Разрядность счетчика или входных значений data[] и выходных q[].
LPM_DIRECTION	Может принимать значения “UP”, “DOWN” или “UNUSED”. Если этот параметр используется, то вход updown не должен быть подключен. Если вход updown не подключен, то значение LPM_DIRECTION по умолчанию – “UP”
LPM_MODULUS	Максимальный счет, плюс один. Число уникальных состояний в цикле счетчика. Если введенное значение больше, чем LPM_MODULUS параметр, поведение счетчика не определено.
LPM_AVALUE	Постоянное значение, которое загружается, когда aset высок. Если введенное значение больше чем <modulus>, поведение счетчика - неопределенный (X) логический уровень, где <modulus> - LPM_MODULUS. Параметр ограничен значением в 32 бита.
LPM_SVALUE	Постоянное значение, которое загружается по переднему фронту тактовых импульсов, когда sset или sconst высок. Должен использоваться, если sconst используется.
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL

Multiplier

Входные выходы	
Имя вывода	Описание
dataa[]	Множимое
datab[]	Множитель
sum[]	Частичная сумма
clock	Вход тактовых импульсов

clken	Разрешение использования тактового входа
aclr	Асинхронный сброс
<i>Выходные выводы</i>	
Имя вывода	Описание
result[]	result = dataa[] * datab[] + sum. The product LSB is aligned with the sum LSB.

<i>Параметры</i>	
Параметр	Описание
LPM_WIDTHA	Разрядность dataa[].
LPM_WIDTHB	Разрядность datab[].
LPM_WIDTHP	Разрядность result[].
LPM_WIDTHS	Разрядность sum[]. Обязателен, даже если порт суммы не используется.
LPM_REPRESENTATION	Тип выполняемого сравнения "SIGNED", "UNSIGNED", "UNUSED". Если значение не указано, то по умолчанию устанавливается "UNSIGNED"
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL
INPUT_A_IS_CONSTANT	Altera параметр. Принимает значения "YES", "NO", и "UNUSED". Если dataa [] связан с постоянным значением, устанавливая INPUT_A_IS_CONSTANT "YES" оптимизирует <i>multiplier</i> по использованию ресурсов и скорости. Если опущено, значение по умолчанию - "NO".
INPUT_B_IS_CONSTANT	Altera параметр. Принимает значения "YES", "NO", и "UNUSED". Если datab [] связан с постоянным значением, устанавливая INPUT_B_IS_CONSTANT "YES" оптимизирует <i>multiplier</i> по использованию ресурсов и скорости. Значение по умолчанию - "NO".

USE_EAB	Altera параметр. Принимает значения "ON", "OFF", и "UNUSED". Устанавливая параметр USE_EAB "ON" позволяет QUARTUSII использовать блоки дополнительных атрибутов, чтобы использовать 4 x 4 или (8 x значение константы) стандартные блоки в ACEX1K и FLEX10K устройствах.
LATENCY	Altera параметр. То же, что и LPM_PIPELINE. Параметр обеспечивает совместимости с QUARTUSII проектами версии ниже 7.0. Для всех новых проектов, используется параметр LPM_PIPELINE
MAXIMIZE_SPEED	Altera параметр. Возможные значения от 0 до 10. Если параметр используется то QUARTUSII пытается оптимизировать данную функцию lpm_mult для скорости, а не для уменьшения занимаемой области, и отменяет установку опции Optimize в диалоговом окне Global Project Logic Synthesis (меню Assign). Если MAXIMIZE_SPEED не использован, значение опции Optimize используется вместо него. Если установлено MAXIMIZE_SPEED - 6 или выше, компилятор оптимизирует мегафункции lpm_mult для более высокой скорости; если установлено - 5 или меньше, компилятор оптимизирует для уменьшения занимаемой области.

Comparator

Входные выводы	
Имя вывода	Описание
dataa[]	datab[] сравнивается с этим значением
datab[]	Значение с которым сравнивается dataa[]
clock	Вход тактовых импульсов

clken	Разрешение использования тактового входа
aclr	Асинхронный сброс
<i>Выходные выводы</i>	
Имя вывода	Описание
alb	“High” (1) если dataa[] < datab[]
aeb	“High” (1) если dataa[] == datab[]
agb	“High” (1) если dataa[] > datab[]
ageb	“High” (1) если dataa[] >= datab[]
aneb	“High” (1) если dataa[] != datab[]
aleb	“High” (1) если dataa[] <= datab[]

<i>Параметры</i>	
Параметр	Описание
LPM_WIDTH	Разрядность входов dataa[] и datab[]
LPM_REPRESENTATION	Тип выполняемого сравнения “SIGNED”, “UNSIGNED”, “UNUSED”. Если значение не указано, то по умолчанию устанавливается “UNSIGNED”
LPM_PIPELINE	
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файла проекта VHDL
CHAIN_SIZE	
ONE_INPUT_IS_CONSTANT	Специфический Altera - параметр. Принимает значения "YES", "NO", или "UNUSED". Обеспечивает большую оптимизацию, если один из входов постоянен. По умолчанию - "NO".

Adder Subtractor

Входные выводы	
Имя вывода	Описание
dataa[]	Первое слагаемое/ Уменьшаемое
datab[]	Слагаемое/ Вычитаемое

add_sub	Если “1” (high), операция = dataa[]+datab[] +cin. Если “0” (low), операция = dataa[]-datab[] +cin-1
clock	Вход тактовых импульсов
clken	Разрешение использования тактового входа
aclr	Асинхронный сброс

<i>Выходные выводы</i>	
Имя вывода	Описание
result[]	dataa[] +datab[] +cin или dataa[] -datab[] +cin-1.
cout	Обнаруживает переполнения в операциях "UNSIGNED".
overflow	Результат превышает доступную точность

<i>Параметры</i>	
Параметр	Описание
LPM_WIDTH	Разрядность входов dataa[],datab[],result[]
LPM_DIRECTION	Значения - "ADD", "SUB", и "UNUSED". Если не указано, значение по умолчанию "DEFAULT", в этом случае используется значение add_sub порта. Add_sub порт не может использоваться, если используется LPM_DIRECTION.
LPM_REPRESENTATION	Тип выполняемого сравнения “SIGNED”, ”UNSIGNED”, “UNUSED”. Если значение не указано, то по умолчанию устанавливается ”UNSIGNED”
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файла проекта VHDL
ONE_INPUT_IS_CONSTANT	Altera параметр. Принимает значения "YES", "NO", и "UNUSED". Обеспечивает большую оптимизацию, если один вход постоянный. Если не указано, значение

	по умолчанию - "NO"
MAXIMIZE_SPEED	Altera параметр. Возможные значения от 0 до 10. Если параметр используется то QUARTUSII пытается оптимизировать данную функцию lpm_mult для скорости, а не для уменьшения занимаемой области, и отменяет установку опции Optimize в диалоговом окне Global Project Logic Synthesis (меню Assign). Если MAXIMIZE_SPEED не использован, значение опции Optimize используется вместо него. Если установлено MAXIMIZE_SPEED – 6 или выше, компилятор оптимизирует мегафункции lpm_mult для более высокой скорости; если установлено - 5 или меньше, компилятор оптимизирует для уменьшения занимаемой области.

Absolute Value

Входные выводы	
Имя вывода	Описание
data []	Число со знаком

Выходные выводы	
Имя вывода	Описание
result[]	Абсолютное значение data [] .
overflow	

Параметры	
Параметр	Описание
LPM_WIDTHA	Разрядность data [] и result[]
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL

Divider

Входные выходы	
Имя вывода	Имя вывода
numer[]	Числитель
denom[]	Знаменатель
clock	Вход тактовых импульсов
clken	Разрешение использования тактового входа
aclr	Асинхронный сброс
Выходные выходы	
Имя вывода	Описание
quotient[]	Частное
remain[]	Остаток

Параметры	
Параметр	Описание
LPM_WIDTHN	Разрядность numer[] и quotient[].
LPM_WIDTHD	Разрядность denom[] и remain[].
LPM_NREPRESENTATION	Определяет параметр числителя “SIGNED” или “UNSIGNED” Сейчас поддерживается только “UNSIGNED”.
LPM_DREPRESENTATION	Определяет параметр знаменателя “SIGNED” или “UNSIGNED” Сейчас поддерживается только “UNSIGNED”
LPM_HINT	Позволяет определять специфические Altera-параметры в файлах проекта VHDL.
LPM_TYPE	Идентифицирует LPM имя файлах проекта VHDL

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Головков А., Пивоваров И., Кузнецов И. Компьютерное моделирование и проектирование радиоэлектронных средств. Учебник для вузов. Стандарт третьего поколения.- СПб.: 2015. – 208 с.
2. Соловьев В.В., Климович А. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Горячая линия - Телеком, 20011. – 376 с.
3. Стешенко В. ПЛИС фирмы ALTERA: элементная база, система проектирования и языки описания аппаратуры - М.: Додека, 2010. – 576 с.
4. Антонов А.П. Язык описания цифровых устройств AlteraHDL: Практический курс. – М.: ИП «Радиософт», 2013. – 224 с.
5. Ефремов Н.В. Введение в систему автоматизированного проектирования Quartus II. Учебное пособие. – М.: ГОУ ВПО МГУЛ, 2011. – 147 с.