Лабораторная работа № 1.

Часть 1 Обзор САПР Quartus II

1.1. Назначение и возможности Quartus II

Пакет Quartus II представляет собой автоматизированную систему сквозного проектирования цифровых устройств на кристаллах ПЛИС фирмы Altera [1]. Он предоставляет пользователю широкие возможности по вводу описаний проекта, логическому синтезу, компиляции проекта, программированию ПЛИС, функциональному и временному моделированию, временному анализу и анализу потребляемой мощности проекта, реализации внутрисистемной отладки.

В Quartus II используется удобный графический интерфейс и простая в применении справочная система, содержащая всю необходимую для выполнения проектирования информацию. Также пакет позволяет использовать командную строку для выполнения каждого этапа проектирования. Причем, в зависимости от предпочтений пользователя, графический интерфейс или командная строка могут использоваться как для выполнения отдельных этапов, так и для всего проекта в целом.

Пакет Quartus II интегрирует в себе большое количество программных модулей, предназначенных для выполнения различных этапов проектирования. Задание параметров и выполнение типовых команд выполняется в отдельных модулях одинаково, что значительно облегчает работу пользователя. Редакторы исходных файлов проекта (графический, текстовый, редактор символов, содержимого модулей памяти, временных диаграмм, конечных автоматов) используют одинаковые подходы и приёмы, а также похожие оконные формы,

применяемые при создании и редактировании исходных файлов с описанием модулей проектируемого устройства.

В одном иерархическом проекте можно сочетать использование различных типов описания исходных файлов модулей проекта, подбирая наиболее подходящий тип, для каждого модуля.

В состав стандартной библиотеки Quartus II входит большое количество базовых элементов, включая мегафункции и макрофункции. Составной частью мегафункций являются операционные устройства, созданные по стандарту библиотеки параметризируемых модулей (LPM – library of parameterized modules).

Значительная часть мегафункций разработана фирмой Altera. Они описаны на языке низкого уровня и оптимизированы для применения в ПЛИС компании Altera. Остальная часть разработана компаниями партнерами. Применение мегафункций в проектах пользователя значительно расширит возможности проектирования и ускорит выполнение проекта.

Пакет Quartus II содержит средство ***SOPC*** (System on programmable chip) ***Builder***, предназначенное для проектирования реализуемых на кристалле процессорных систем [4]. С помощью этого средства можно легко создавать конфигурируемые процессорные ядра, реализовывать на кристалле различные контроллеры, а также значительную часть периферийного оборудования.

1.2. Понятие проекта в Quartus II

Под термином «проект» в Quartus II понимается набор файлов, связанных с проектируемым устройством, и набор соответствующих библиотек [4].

Файлы могут быть двух типов – логические и вспомогательные. Логические файлы описывают поведение или структуру отдельных модулей проектируемого устройства. К ним относятся файлы с текстовым описанием на языках описания аппаратуры (HDL, Hardware Description Language), файлы с графическим представлением схем, файлы с представлением отдельных модулей в виде конечных автоматов. Вспомогательные файлы содержат дополнительную информацию о проектируемом устройстве. Большинство вспомогательных файлов не содержит описания логики проекта. Некоторые из них автоматически создаются приложением Quartus II, некоторые вводятся пользователем. Примерами вспомогательных файлов являются файлы установок и назначений (**.qsf**), символьные файлы (**.bsf**), файлы отчетов (**.rpt**).

Как правило, проект содержит несколько логических файлов, образующих иерархическое описание создаваемого устройства. Один из файлов представляет описание устройства на верхнем уровне иерархии. Он является модулем верхнего уровня.

Если описание устройства содержит единственный логический файл, то по умолчанию он является модулем верхнего уровня описания.

Каждый проект размещается в отдельной папке и имеет своё собственное имя, которое назначается пользователем при создании нового проекта. Автоматически при создании нового проекта создаются файлы с расширением **.qpf** (quartus project file) и **.qsf** (quartus settings file). Первый из этих файлов содержит номер версии Quartus II, используемой для создания проекта, дату создания и название активной версии проекта. Пример файла с расширением **.qpf** приведен на рис.1.2.1.

```
# --------------------------------------------------------------------------
#
# Quartus II
# Version 9.1 Build 222 10/21/2009 SJ Full Version
# Date created = 13:23:39  April 20, 2011
#
# --------------------------------------------------------------------------

QUARTUS_VERSION = "9.1"
DATE = "13:23:39  April 20, 2011"

# Revisions

PROJECT_REVISION = "new_project"
```

Рис. 1.2.1. Содержимое файла new_project.qpf

Второй файл содержит установки и назначения. Пример файла с расширением **.qsf** приведен на рис.1.2.2.

```
# Quartus II
# Version 9.1 Build 222 10/21/2009 SJ Full Version
# Date created = 13:23:39  April 20, 2011
#
# ------------------------------------------------------------------------- #
#
# Notes:
#
# 1) The default values for assignments are stored in the file:
#              new_project_assignment_defaults.qdf
#    If this file doesn't exist, see file:
#              assignment_defaults.qdf
#
# 2) Altera recommends that you do not modify this file. This
#    file is updated automatically by the Quartus II software
#    and any changes you make may be lost or overwritten.
#
# ------------------------------------------------------------------------- #


set_global_assignment -name FAMILY "Cyclone II"
set_global_assignment -name DEVICE EP2C70F896C6
set_global_assignment -name TOP_LEVEL_ENTITY new_project
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 9.1
set_global_assignment -name PROJECT_CREATION_TIME_DATE "13:23:39  APRIL 20, 2011"
set_global_assignment -name LAST_QUARTUS_VERSION 9.1
set_global_assignment -name USE_GENERATED_PHYSICAL_CONSTRAINTS OFF -section_id eda_blast_fpga
set_global_assignment -name SEARCH_PATH "e:\\altera\\pr22/"
set_global_assignment -name DEVICE_FILTER_SPEED_GRADE FASTEST
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
```

Рис. 1.2.2. Содержимое файла new_project.qsf

Он представляет собой текстовый файл, использующий синтаксис языка tcl. Поэтому файл **.qsf** можно редактировать в любом текстовом редакторе. При создании проекта пользователь может воспользоваться

существующими в пакете Quartus II установками и назначениями, либо использовать их из ранее созданного проекта, либо выполнить свои собственные.

В процессе работы над проектом, пользователь может изменить установки и назначения. В этом случае Quartus II позволяет создавать разные версии проекта. Каждой версии назначается собственное имя. Пользователь может выполнить компиляцию проекта для разных его версий и на основе сравнительного анализа результатов компиляции разных версий проекта, выбрать ту версию, которая ему более подходит.

При создании версии проекта, автоматически создается файл <revision_name>_assignment_defaults.qdf и размещается в рабочей папке проекта, где revision_name – имя версии. Он содержит установки и назначения, которые будут использоваться по умолчанию.

Для работы с проектом в менеджере проекта Quartus II используется команда **Project**, выпадающее меню которой приведено на рис. 1.2.3.

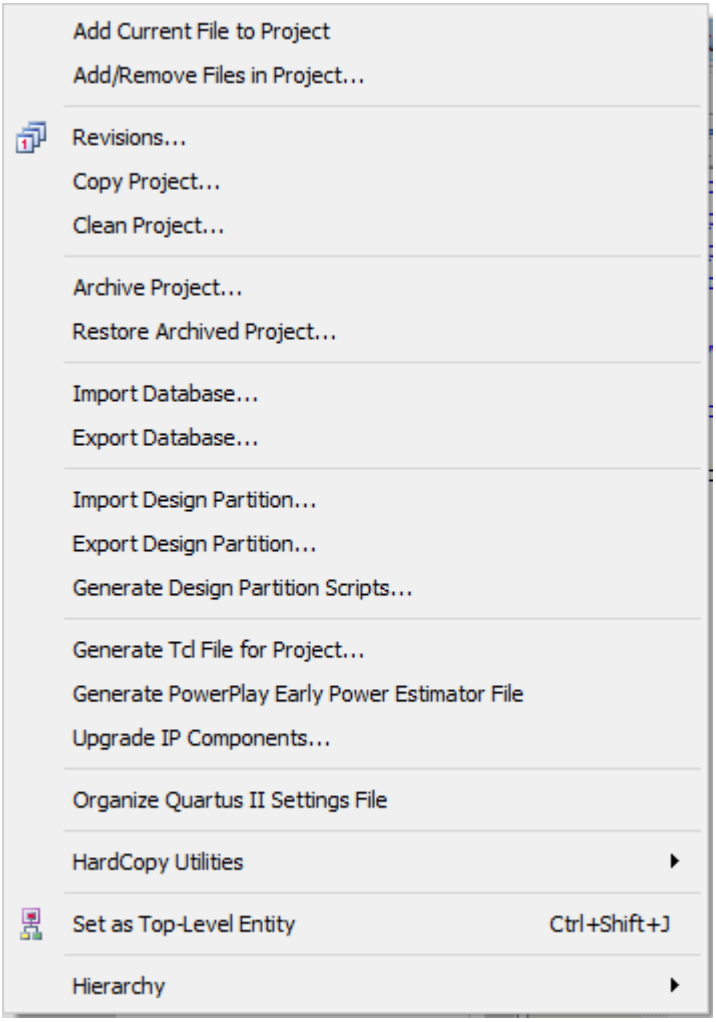


Рис. 1.2.3. Меню команды **Project** менеджера проекта

С помощью команд из этого меню к проекту можно добавлять либо удалять отдельные файлы, создавать новую версию проекта, создавать копии проекта, архивировать проект, восстанавливать проект из архива, импортировать или экспортировать базу данных и так далее.

1.3. Процедура проектирования в Quartus II

Процедура проектирования устройств на ПЛИС включает в себя следующие этапы [1].

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.**

Графическое представление проекта в пакете Quartus II может создаваться, как в базисе библиотечных элементов, так и в базисе графических символов проектировщика. В качестве компонентов проекта Quartus II позволяет использовать IP (Intellectual Properties) ядра, представляющие собой единицы интеллектуальной собственности. Эти блоки являются полностью синтезируемыми, перемещаемыми и могут по желанию проектировщика располагаться в разных частях кристалла. Примерами IP ядер являются процессорные ядра NIOS II и большой набор различных контроллеров периферийных устройств.

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. Такие возможности позволяют представлять проект 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**.

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.

Quartus II допускает использование в проектах отдельных компонентов, созданных в системах автоматизации проектирования сторонних производителей. Такими компонентами могут быть и IP ядра. В этом случае, файл описания, как правило, представляет собой таблицу соединений конфигурируемых логических блоков и таблиц соответствия, выполненную до этапа размещения и разводки элементов. Соответствующие файлы имеют расширение **.edf**.

2. 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.

2.1. Анализ и синтез. 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 (lut), триггеры, защелки, логические элементы, блоки встроенной памяти, встроенные умножители. Рис. 1.3.1. демонстрирует реализацию этапа анализа и синтеза. 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].

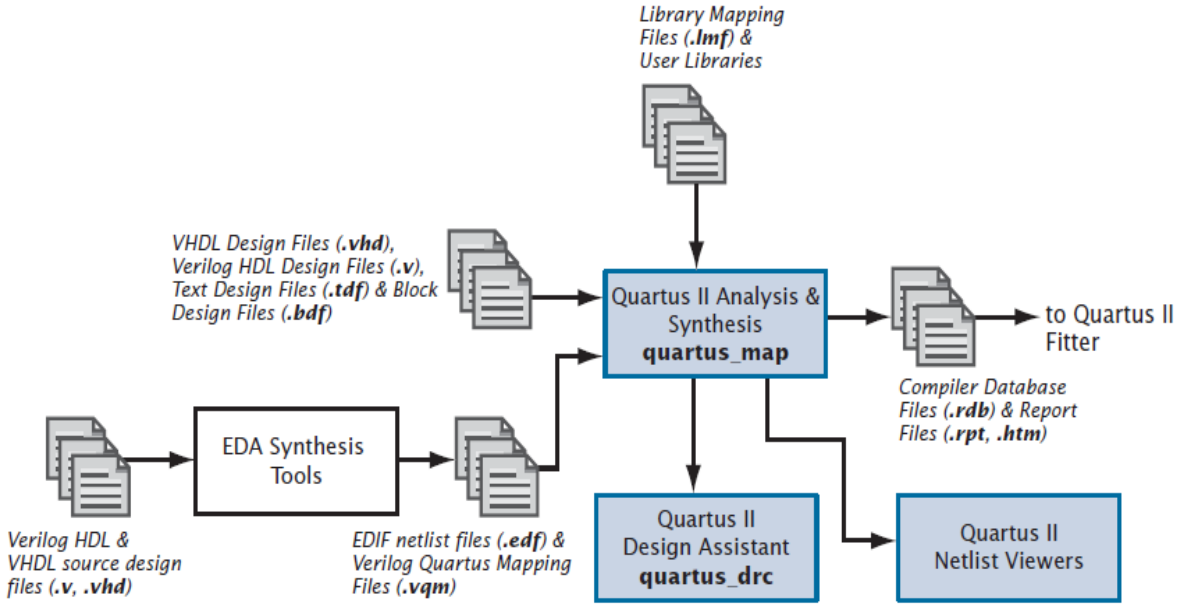


Рис. 1.3.1. Этап анализа и синтеза проекта в Quartus II

Пакет Quartus II допускает использование средств синтеза сторонних производителей САПР. В этом случае, созданные проектировщиком файлы описания проекта на языке VHDL или Verilog, должны быть преобразованы средствами синтеза в файл соединений в формате EDIF (**.edf**) или файл Verilog Quartus Mapping File (**.vqm**), которые затем могут быть использованы в пакете Quartus II (см. рис. 1.3.1.).

3. Функциональное 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. Существует два подхода к генерации внешних, относительно проекта, воздействий. Первый подход заключается в формировании воздействий путем задания

последовательности входных сигналов в редакторе временных диаграмм. Второй подход состоит в написании специальной тестирующей программы с помощью одного из языков HDL. При использовании второго подхода тестируемый объект представляется как структурный компонент, соединенный с одной стороны с генератором тестовых воздействий, с другой стороны с анализатором реакций. Пакет Quartus II поддерживает реализацию обоих подходов. Верификация может быть выполнена как для отдельных частей проекта, так и для проекта в целом. Для выполнения этапа верификации также могут привлекаться средства моделирования, разработанные сторонними производителями САПР.

2.2. Размещение и трассировка. 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. Рис. 1.3.2. демонстрирует реализацию этапа размещения и трассировки. 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].

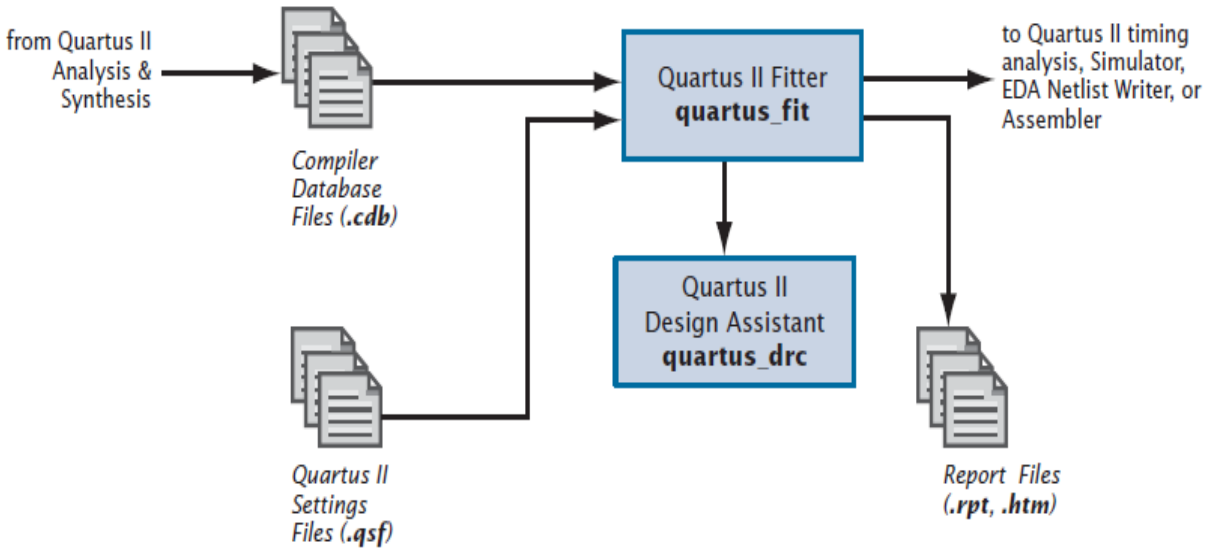


Рис. 1.3.2. Этап размещения и трассировки проекта в Quartus II

При выполнении этапа размещения и трассировки пользователь может задать свои собственные назначения. Модуль размещения и трассировки реализует их, а затем выполняет оптимизацию оставшейся части проекта. После выполнения монтажа, пользователь может просмотреть результаты размещения и трассировки с помощью специального средства ***Chip Planner*** и, в случае необходимости, изменить некоторые назначения.

Результатом компиляции проекта в пакете Quartus II является загрузочный файл, т.е. конфигурационная информация для выбранной микросхемы ПЛИС или для загрузочного ПЗУ этой интегральной схемы. Также создается файл отчета, содержащий информацию, как о процессе компиляции, так и о его результатах.

В пакете Quartus II имеется специальное средство ***Resource Property Editor***, называемое редактором топологии, с помощью которого можно вручную вмешаться в процесс компиляции и изменить структуру проекта на кристалле, например, с целью повышения производительности проектируемого устройства.

2.3. Временной анализ. Проверка соответствия реализованного проекта требованиям быстродействия.

2.4. Анализ потребляемой мощности. Проверка соответствия реализованного проекта требованиям по потребляемой мощности.

4. Временное моделирование на вентильном уровне. Осуществляется проверка правильности функционирования проекта после выполнения этапов синтеза, размещения и трассировки.

5. Программирование ПЛИС. Выполняется загрузка конфигурационной информации в программируемый кристалл, посредством чего реализуется пользовательский проект.

6. Тестирование и отладка проекта в составе системы. Проводятся комплексные испытания реализованного проекта, в результате чего принимается решение о соответствии проекта техническому заданию и завершении проектирования. В противном случае принимается решение о доработке проекта.

1.4. Графический пользовательский интерфейс Quartus II. Навигатор проекта

Окно навигатора проекта содержит страницы со следующими закладками: **Hierarchy**, **Files**, **Design Units**.

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. Для вызова меню настройки необходимо выбрать модуль проекта, щелкнув по его имени левой кнопкой мыши, а затем нажать правую кнопку мыши. На рис. 1.5.2.1. приведен пример иерархической структуры некоторого проекта.

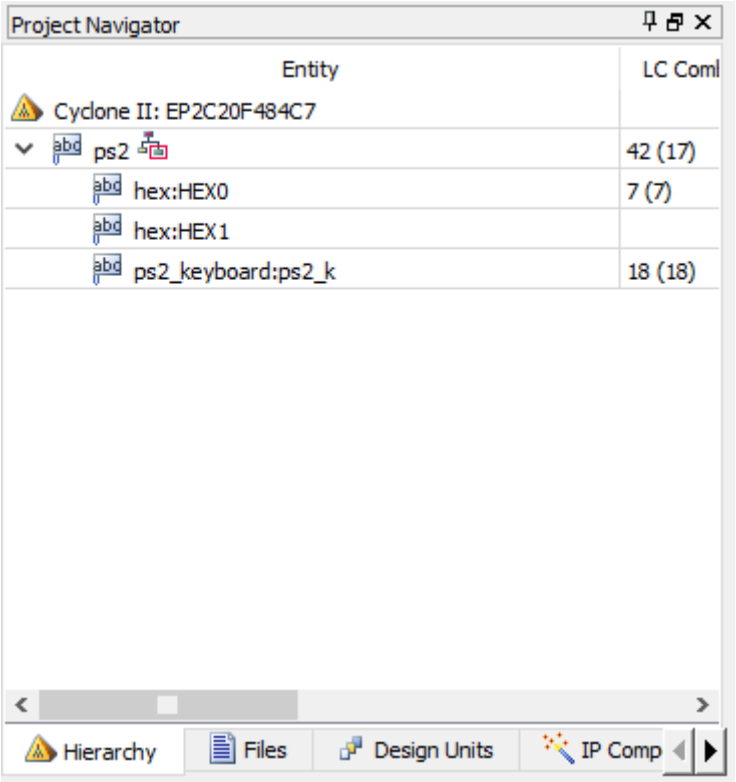


Рис. 1.5.2.1. Окно навигатора проекта

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 1.5.2.2. 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.

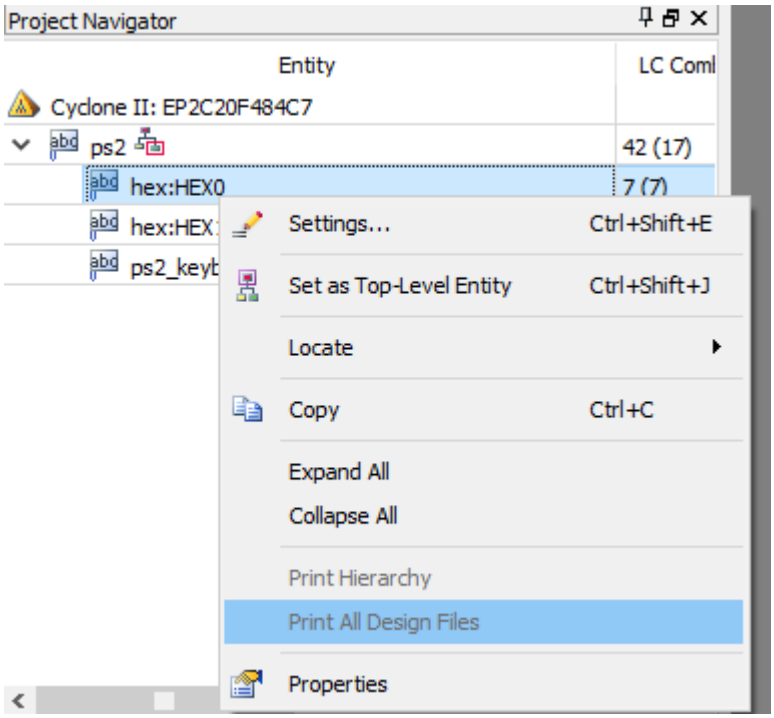


Рис. 1.5.2.2. Страница **Files** навигатора проекта с контекстно-зависимым меню

Страница **Design Units** навигатора отображает все компоненты проекта, использованный способ описания проекта, язык описания и файл с описанием компонента (см. рис. 1.5.2.3).

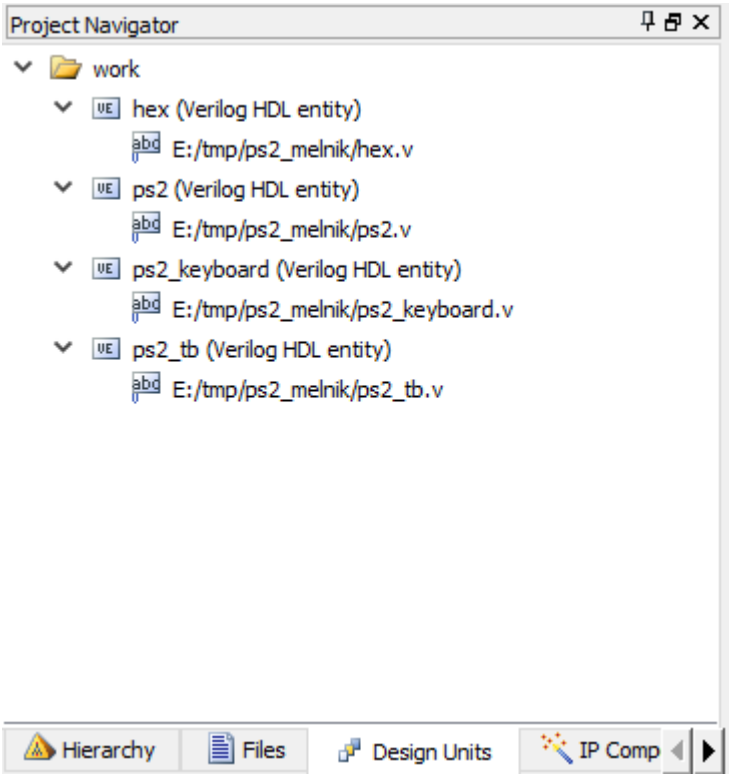


Рис. 1.5.2.3. Страница **Design Units** навигатора проекта

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.

1.5. Создание нового проекта

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. 1.6.1.

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.

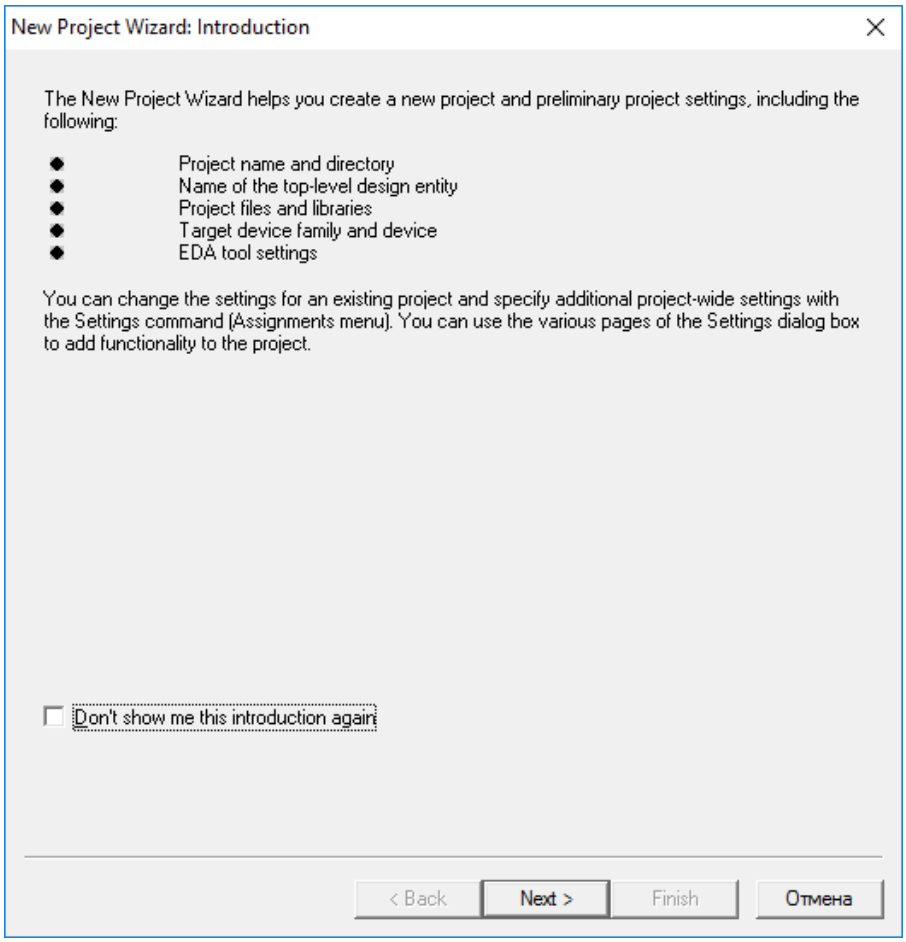


Рис. 1.6.1. Вводное окно **NPW**

В следующем окне [page1] **NPW** , представленном на рис. 1.6.2, выберите рабочую папку для размещения проекта, задайте имя проекта и определите модуль верхнего уровня. 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 пользователь может задать установки, которые будут использоваться по умолчанию в его

новом проекте. Причем можно использовать установки либо из последнего открытого в Quartus II проекта, либо из указанного пользователем проекта.

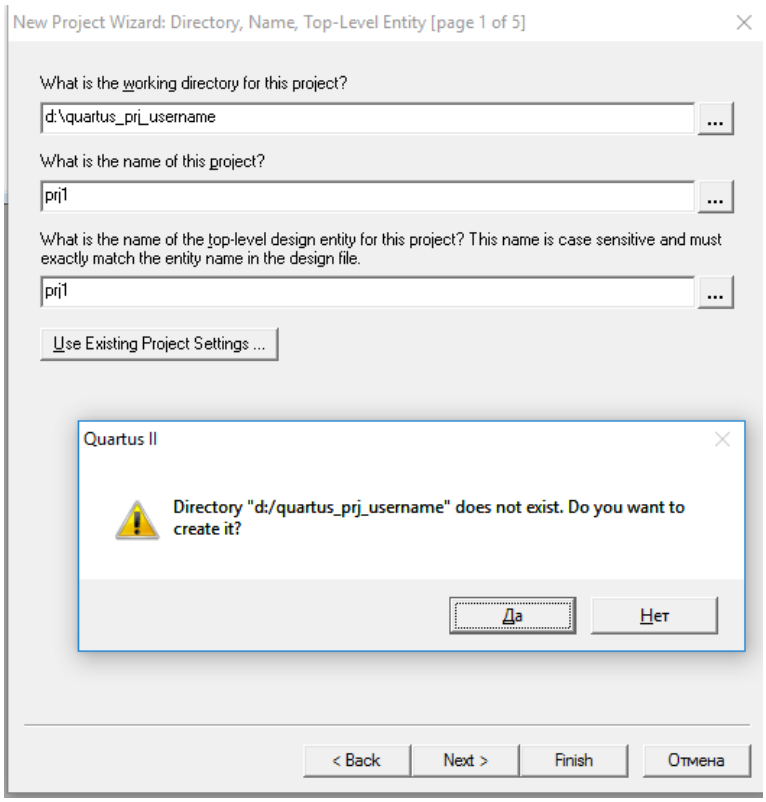


Рис. 1.6.2. Окно указания рабочей папки, имени проекта и модуля верхнего уровня

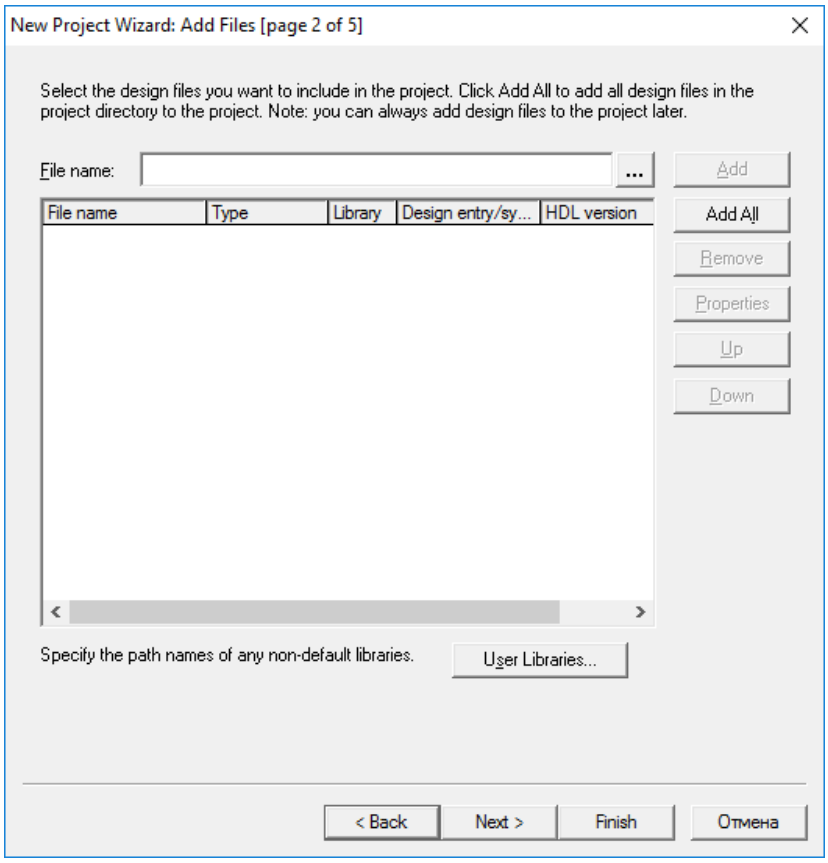


Рис. 1.6.3. Окно **NPW** включения в проект файлов и библиотек

В следующем окне [page2] **NPW**, показанном на рис 1.6.3, 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** пользователь может добавить к своему проекту либо выделяемые им файлы по отдельности, либо сразу все файлы. Также он может удалить файлы, посмотреть их свойства, либо переставить местами. Порядок файлов имеет значение при компиляции проекта.

В рассматриваемом окне **NPW** с помощью кнопки **User Libraries** пользователь может 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. 1.6.4.

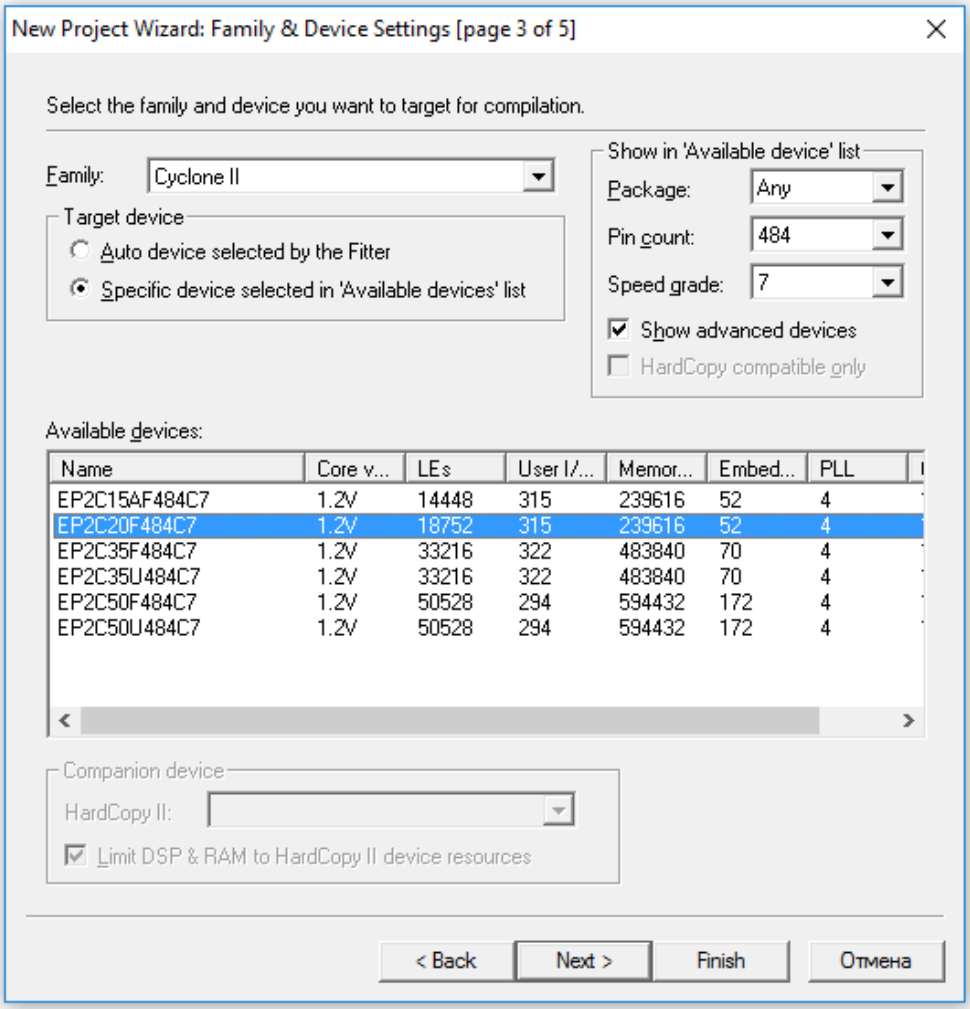


Рис. 1.6.4. Окно **NPW** задания целевого кристалла ПЛИС

В этом окне пользователь выбирает ПЛИС, на которой будет реализован проект. Причем кристалл может быть выбран пользователем из предлагаемого списка, либо автоматически назначен компоновщиком. В случае выбора кристалла пользователем, 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. 1.6.5. На рис. 1.6.4. приведен пример задания типа кристалла ПЛИС Cyclone EP2C70F484C7.

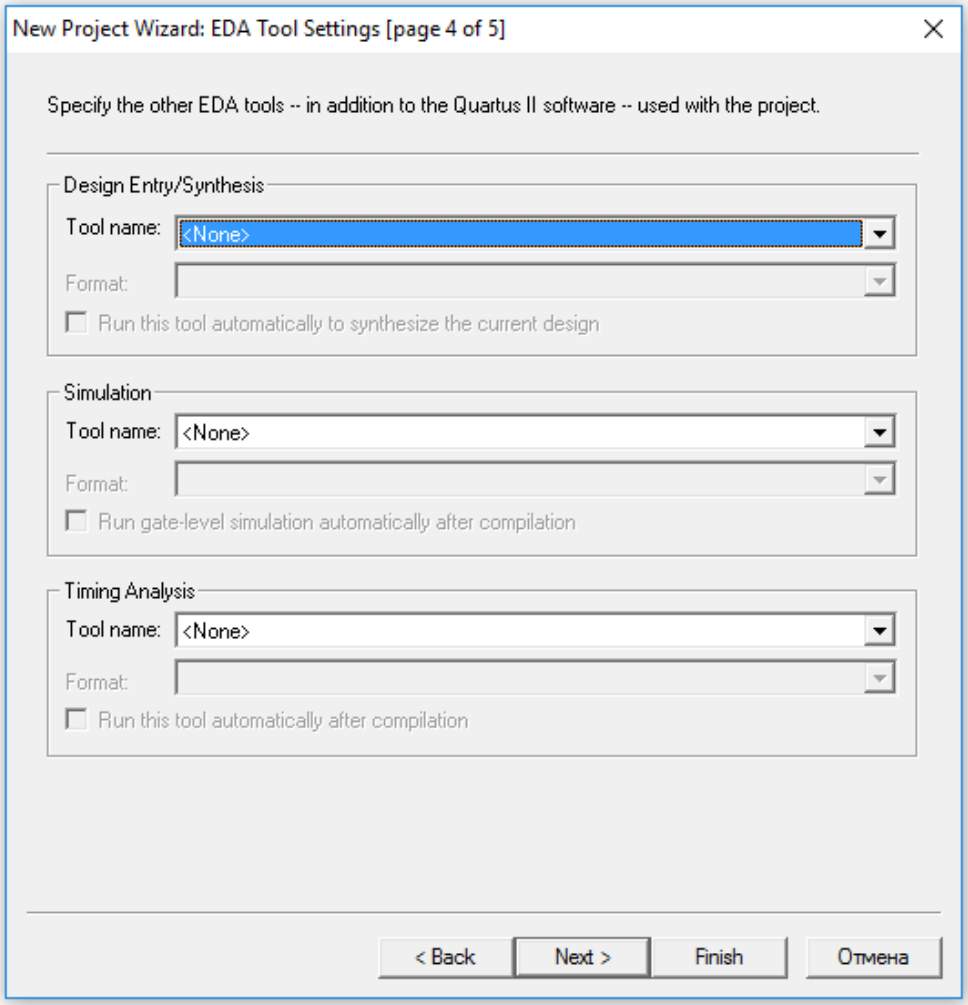


Рис. 1.6.5. Окно **NPW** декларации об использовании средств автоматизации сторонних производителей

В следующем окне [page 4] **NPW** пользователь может определить средства автоматизации проектирования сторонних производителей, которые он планирует использовать для ввода проекта, синтеза, моделирования или временного анализа. В последующем эти средства будут вызываться непосредственно из пакета Quartus II. Пример окна на рис. 1.6.5. соответствует случаю, когда дополнительные средства автоматизации проектирования использовать не планируется.

С помощью кнопки **Next** перейдите к завершающему окну [page5] **NPW**, которое показано на рис. 1.6.6.

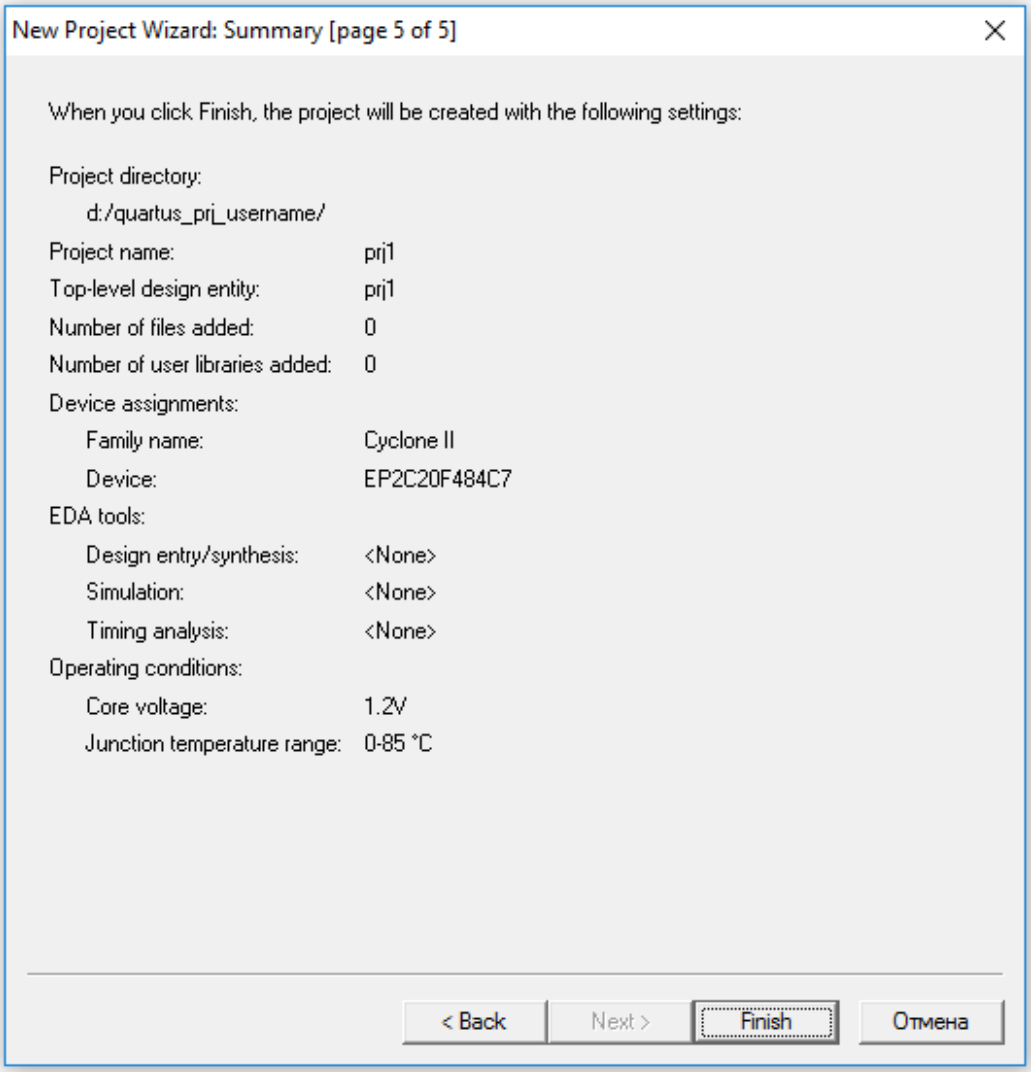


Рис. 1.6.6. Окно **NPW** с выполненными установками для создаваемого проекта

В этом окне приведена итоговая информация по сделанному пользователем выбору на предыдущих шагах **NPW**, а также информация по условиям работы выбранного кристалла. Нажатие на кнопку **Finish** завершит работу **NPW**. Если теперь открыть окно навигатора проекта, то на странице **Hierarchy** появится тип выбранного кристалла ПЛИС и имя

модуля верхнего уровня, а на странице **Files** - добавленные к проекту файлы.

Утилита **NPW** позволяет на каждом шаге своего выполнения вернуться к предыдущим шагам, и в случае необходимости, скорректировать выполненные действия.

Часть 2 Ввод описания проекта в Quartus II

2.1. Назначение и возможности текстового редактора

Пакет Quartus II поддерживает ввод описаний отдельных модулей, созданных с использованием языков описания аппаратуры (HDL). Допустимыми языками в пакете Quartus II являются языки VHDL, Verilog, Altera HDL и System Verilog [5-7]. Файлы, содержащие описания на соответствующих языках, имеют расширение **.vhd**, **.v**, **.tdf** и **.sv** соответственно. В Quartus II эти типы файлов можно комбинировать с другими типами файлов описаний при создании иерархических проектов.

Для создания и редактирования файлов с текстовым описанием проектируемых модулей предназначен текстовый редактор пакета Quartus II. Он также может быть использован для ввода и редактирования любых других текстовых файлов, включая файлы, созданные в пакете Quartus II. Примером являются командные файлы, созданные на языке Tcl. Такие файлы имеют расширение **.tcl**. Командные файлы могут быть взяты из существующих проектов, и после редактирования использованы в создаваемом проекте.

Дополнительно в текстовом редакторе можно создавать и редактировать файлы с исходным текстом программ на языке ассемблера (имеют расширение **.a**, **.asm**), на языке Си (**.c**), Си++ (**.cpp**), файлы включения (си/си++ include files (**.h**)) и любые другие ASCII файлы.

В Quartus II допускается использование и других текстовых редакторов для создания и редактирования текстовых файлов. Однако текстовый редактор Quartus II предоставляет ряд дополнительных полезных возможностей.

-Нумерация строк в текстовом файле. Наличие номеров позволит облегчить поиск ошибок при проверке синтаксической правильности созданного текста.

-Использование заготовок языковых конструкций (шаблонов). Эта возможность позволит вставлять в текстовые файлы как любые, поддерживаемые Quartus II конструкции используемого языка HDL, так и реализованные фирмой Altera логические функции, включая примитивы и мегафункции, в виде отдельных компонентов. Аналогичным образом можно включать и определенные пользователем логические функции.

-Возможность интегрирования созданных текстовых файлов в иерархическую структуру проекта, содержащего разнотипные описания отдельных компонентов. Так, например, для текстового описания модуля с помощью команды *Create/Update* из меню **File** можно создать условное графическое обозначение (файл с расширением **.bsf**), которое будет помещено в рабочую библиотеку проекта и в последующем может быть использовано, как элемент на более высоком уровне иерархии при схемотехническом проектировании. Аналогичным образом, с помощью такой же команды, может быть создан на языке VHDL файл декларации описанного модуля в виде компонента, который затем может быть использован в VHDL проекте более высокого уровня иерархии.

-Окрашивание языковых конструкций в тексте, что значительно повысит наглядность представления исходных программ.

-Комментирование отдельных фрагментов в тексте.

-Поиск и замена фрагментов текста.

-Поиск и выделение цветом парных скобок.

-Создание и управление закладками. Такая возможность позволит упростить процесс последующей отладки созданного текстового описания модуля.

-Свертывание и развертывание фрагментов текста, что позволит структурировать текстовое описание создаваемого модуля.

-Возможность отсоединения - присоединения окна текстового редактора от главного окна менеджера проекта.

- Наличие средств проверки синтаксической корректности создаваемого текстового описания. Причем в этом случае, для вывода сообщений о найденных ошибках используется отдельное окно сообщений менеджера проекта. В окне же текстового редактора в это время можно наблюдать выделяемые цветом строки, содержащие ошибки, что позволит их легко локализовать.

- Возможность получения оперативной справки, как по командам текстового редактора, так и по ключевым словам, операторам, конструкциям используемого языка HDL, так и по примитивам и мегафункциям, используемым фирмой Altera.

После внесения изменений в текстовый файл к его имени в окне текстового редактора добавляется символ *. Когда же пользователь выполнит сохранение редактируемого файла, Quartus II сохранит предыдущую версию файла в той же директории, с тем же именем, но с расширением **.bak**. Такая возможность позволит, в случае необходимости, вернуться к предыдущей версии текстового файла.

Вызов текстового редактора в пакете Quartus II осуществляется путем открытия текстового файла с помощью пиктограммы , или команды *Open* из меню **File**, на панели инструментов менеджера проекта, либо с помощью контекстно-зависимого меню, открываемого в окне с закладкой **Files** навигатора проекта.

2.2. Создание нового файла с текстовым описанием

Для создания нового текстового файла выполните следующие шаги.

1. Щелкните левой клавишей мыши по кнопке **File** на панели инструментов менеджера проекта.

2. В появившемся меню выберите команду *New*, выполнив щелчок мышью по соответствующей строке (см. рис. 2.2.2.1.).

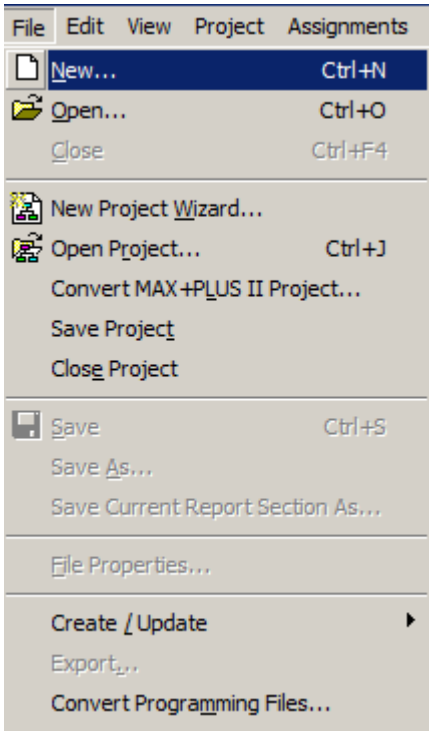


Рис. 2.1.2.1. Фрагмент выпадающего меню команды **File** менеджера проекта

3. Появится окно **New**, изображенное на рис. 2.2.2.2.

4. Укажите тип вновь создаваемого файла. Как уже упоминалось в предыдущем разделе, в Quartus II можно создавать текстовые файлы на языках AHDL, System Verilog, Tcl, Verilog, VHDL.

5. На рис. 2.2.2.2. приведен пример выбора строки *Verilog HDL File*. После этого нажмите кнопку **OK** в нижней части панели.

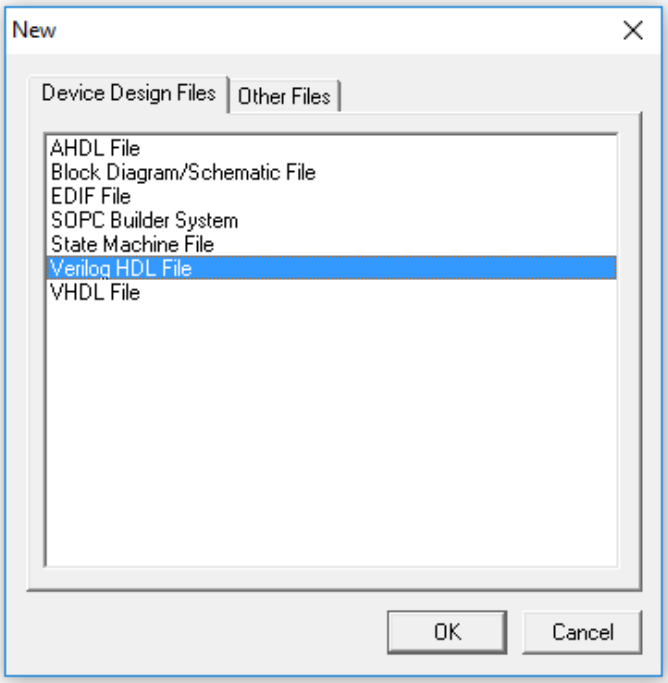


Рис. 2.1.2.2. Окно выбора типа создаваемого файла

6. Чтобы назначить создаваемому файлу собственное имя, выполните команду *Save As* из меню **File** менеджера проекта.

7. В поле «*Имя файла*» введите назначаемое имя и нажмите кнопку **Сохранить**.

2.3. Окно текстового редактора

Вид окна текстового редактора Quartus II, вновь созданного файла, приведен на рис. 2.2.3.1.

Рис. 2.1.3.1. Окно текстового редактора вновь созданного файла

В верхней части, на закладке окна, содержится имя редактируемого файла. Вдоль левой стороны окна расположена панель инструментов текстового редактора. Справа от неё расположено поле номеров строк. Для разрешения или запрещения использования этого поля предназначена соответствующая кнопка инструментальной панели редактора.

Рядом с полем номеров строк расположено поле, в котором используются значки ⊞ (разворачивания) и ⊟ (сворачивания) структурных блоков текстового описания. Их использование в текстовом редакторе позволит значительно повысить наглядность текстового файла.

Назначение кнопок инструментальной панели текстового редактора следующее.

- кнопка отсоединения - присоединения окна. С помощью этой кнопки можно отсоединить окно текстового редактора. То есть представить редактируемый файл в отдельном окне, которое стандартным

образом может быть увеличено до полного экрана монитора. Повторный щелчок по этой кнопке вернёт окно на прежнее место, в главное окно менеджера проекта.

- кнопка поиска. С помощью этой кнопки можно осуществлять поиск отдельных фрагментов в тексте.

- кнопка замены.

- кнопка перехода к парной скобке.

Если установить курсор мыши на открывающейся скобке и нажать эту кнопку, то курсор мыши переместиться к парной закрывающейся скобке. При этом сами скобки будут выделены зеленым цветом.

- кнопка вставки отступления.

- кнопка уменьшения отступления.

Можно выделить фрагмент текста и с помощью этих кнопок выполнить табуляцию текста либо убрать её.

- кнопка установки закладки.

С помощью этой кнопки можно разместить в тексте закладки. Закладка представляет собой кружок голубого цвета, расположенный в поле номеров строк.

- кнопка перехода к следующей закладке.

- кнопка перехода к предыдущей закладке.

- кнопка удаления всех закладок в текущем файле.

- кнопка удаления всех закладок во всех файлах.

- кнопка вставки файла.

- кнопка вставки шаблона языковой конструкции. Указания по использованию этой кнопки содержатся в следующем разделе описания.

- кнопка запуска синтаксического анализа текущего текстового файла.

- кнопка использования номеров строк в текстовом файле. С помощью этой кнопки можно включить / выключить нумерацию строк в текстовом файле.

- кнопка включения / выключения инструмента горизонтальной прокрутки в окне текстового редактора.

 - кнопка включения / выключения показа линий табуляции текста.

- кнопка включения отображения подряд идущих пробелов горизонтальными стрелками.

- кнопка комментирования выделенного фрагмента текста.

- кнопка отказа от комментирования выделенного фрагмента текста.

- включение/отключение автозаполнения текста.

2.4. Использование шаблонов в текстовом описании

Заготовки языковых конструкций помогают значительно облегчить ввод исходного текста описания на одном из возможных языков. Заготовки представляют собой шаблоны с готовыми фрагментами кода. Причем части кода, которые должны быть определены пользователем, выделены черным цветом и заключены в остроугольные скобки (< >).

Чтобы вызвать окно с шаблонами, необходимо нажать кнопку на панели инструментов текстового редактора, либо выполнить команду *Insert Template* из меню **Edit** менеджера проекта. Вид появившегося окна представлен на рис 2.2.4.1.

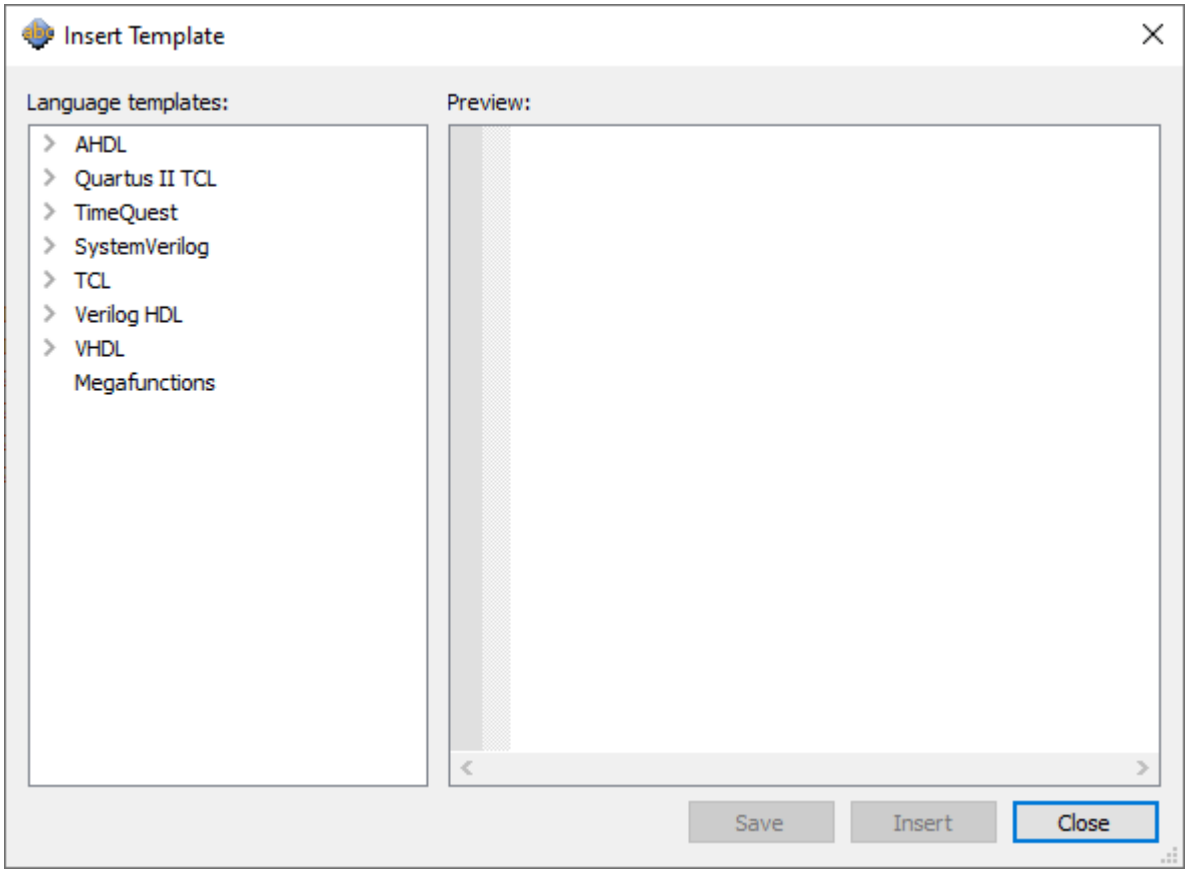


Рис. 2.1.4.1. Окно вставки шаблона

Окно состоит из двух частей. В левой части содержатся шаблоны, представленные в виде иерархической структуры. В правой части представляется фрагмент кода, соответствующий выбранному шаблону. Причем в этом окне можно не только просматривать шаблон, но и редактировать его. На верхнем уровне шаблоны разбиты по категориям в зависимости от используемого языка описания. Имеются также группы *TimeQuest* и *Megafunctions*. В категории *Megafunctions* содержатся шаблоны декларации компонентов проекта. Чтобы раскрыть содержимое каждой категории необходимо щелкнуть левой кнопкой мыши по значку ⊞ рядом с названием категории.

Внутри категории используется еще несколько уровней иерархии классификации шаблонов. Имеются следующие группы.

1. Шаблоны полных проектов (Full Designs) функциональных узлов, таких как блоки памяти, сдвиговые регистры, конечные автоматы, арифметические устройства.

2. Шаблоны базовых конструкций языка описания аппаратуры, таких как элементы проекта, раздел деклараций, параллельные и последовательные операторы языка, выражения.

3. Шаблоны описаний логических элементов, включая регистры, защелки, элементы с тремя состояниями.

4. Шаблоны синтезируемых атрибутов.

5. Шаблоны с фрагментами кода, демонстрирующие использование в текстовом описании примитивов из стандартной библиотеки пакета Quartus II.

Чтобы проверить синтаксическую корректность созданного текстового файла, нажмите кнопку на панели инструментов текстового редактора. При отсутствии ошибок будет выведено сообщение об успешном завершении этапа анализа текущего файла, как показано на рис

2.2.4.2, и в главном окне менеджера проекта появится отчет о выполнении анализа.

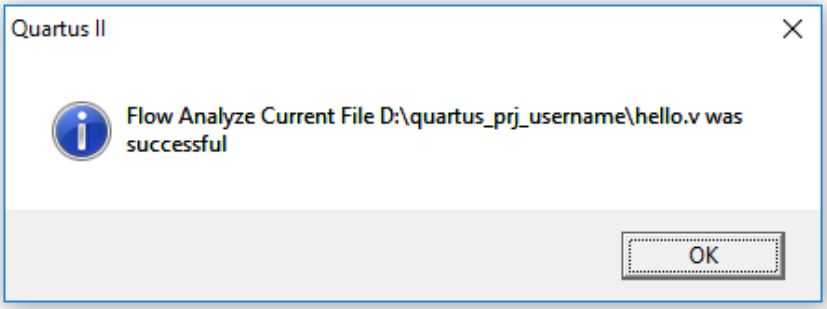


Рис. 2.1.4.2. Окно сообщения об успешном завершении анализа текущего файла

При обнаружении в процессе анализа текущего файла синтаксических ошибок на экран будет выведено соответствующее сообщение с указанием количества обнаруженных ошибок (см. рис. 2.1.4.3.).

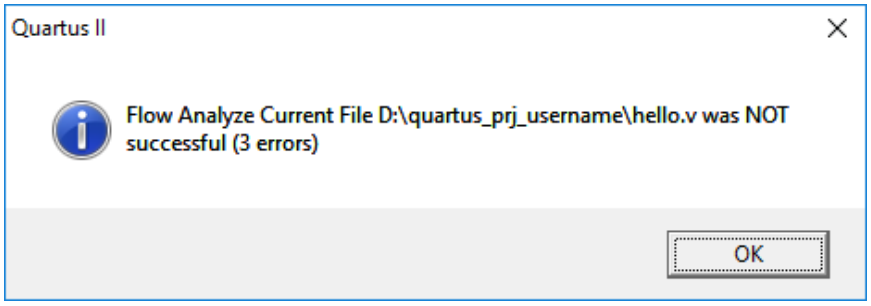


Рис. 2.1.4.3. Окно сообщения об обнаружении ошибок в процессе анализа текущего файла

Причем в окне сообщений менеджера проекта, будет выведена красным цветом информация о выявленных ошибках с указанием их типа и номеров содержащих их строк в текстовом файле. Если двойным щелчком мыши щелкнуть по сообщению об ошибке, то в окне

редактируемого файла строка, содержащая ошибку, выделится синим цветом, что позволит быстро локализовать и устранить ошибку. На рис. 2.1.4.4. приведен пример вывода сообщений об обнаруженных ошибках

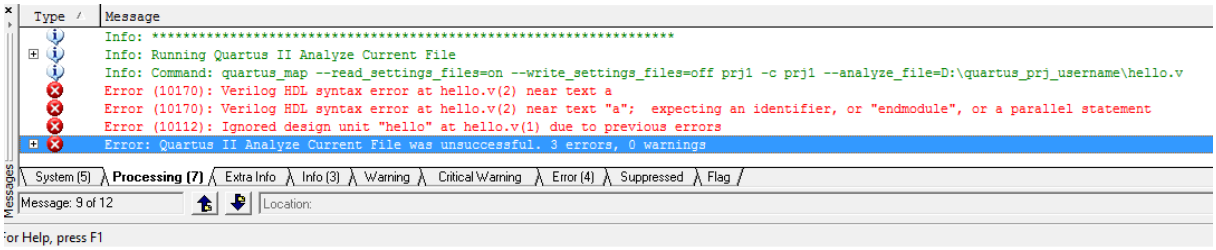


Рис. 2.1.4.4. Пример вывода сообщений об обнаруженных ошибках

Часть 3 Моделирование проекта

Одним из этапов проверки правильности функционирования описанного устройства является поведенческое моделирование.

Имеем файл example.v и тестбенч для него – example_tb.v.

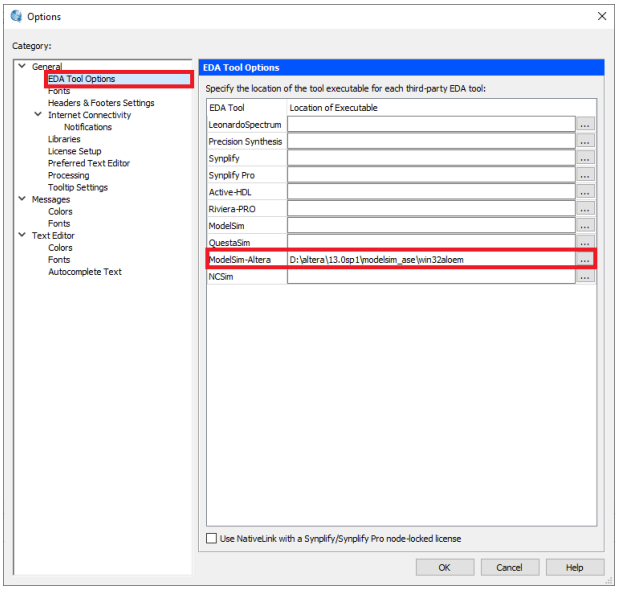
example.v

```
1   module example(in, clk, out);
2   input in;          //информационный вход
3   input clk;         //тактовый сигнал
4   output reg out;    //выход
5
6   always @(posedge clk) begin
7       out <= in;
8   end
9
10  endmodule
```

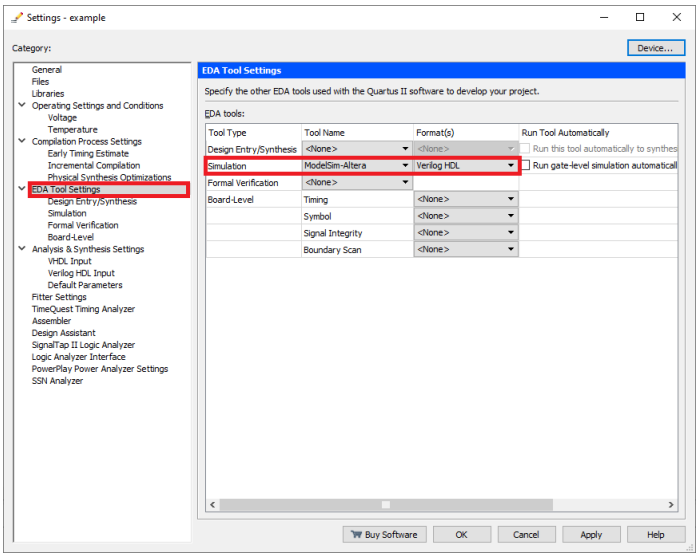
example_tb.v

```
1   `timescale 1ns/1ps
2   module example_tb;
3   wire out;
4   reg in;
5   reg clk;
6       example uut(.in(in),
7                   .clk(clk),
8                   .out(out)
9                   );
10
11  initial begin
12      in = 1'b0;
13      clk = 1'b1;
14      #25 in = 1'b1;
15      #20 in = 1'b0;
16      #20 in = 1'b1;
17      #20 $finish;
18  end
19
20  always #5 clk = !clk;
21  endmodule
```

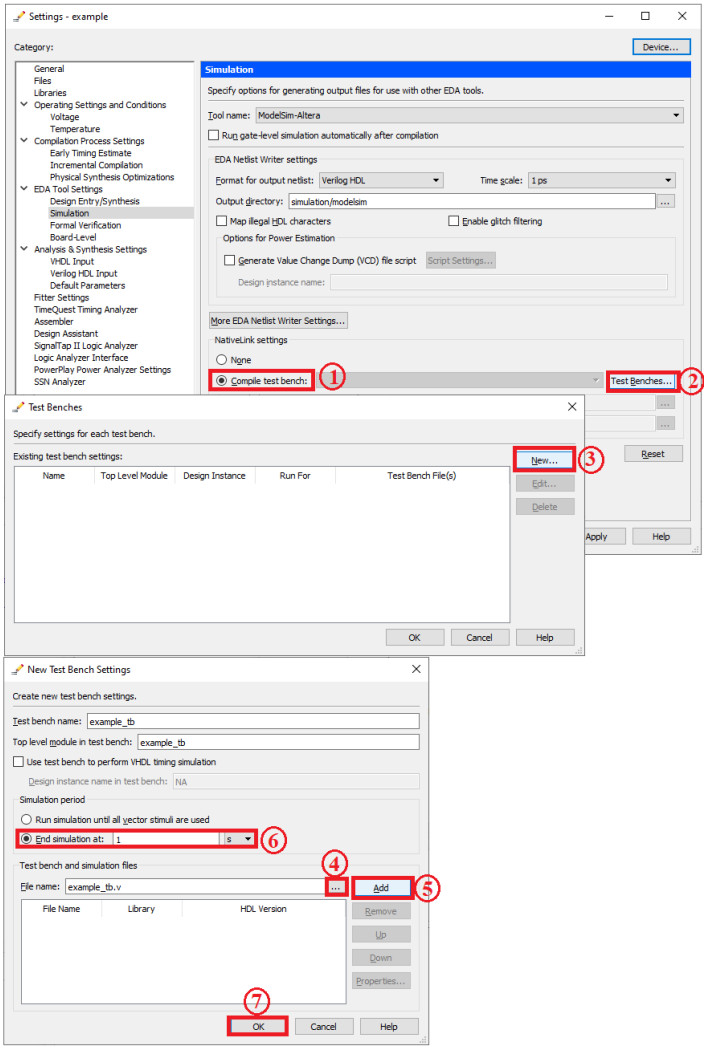
Укажем путь расположения папки с исполняемым файлом симулятора ModelSim-Altera. Tools –> Options –> EDA Tool Option



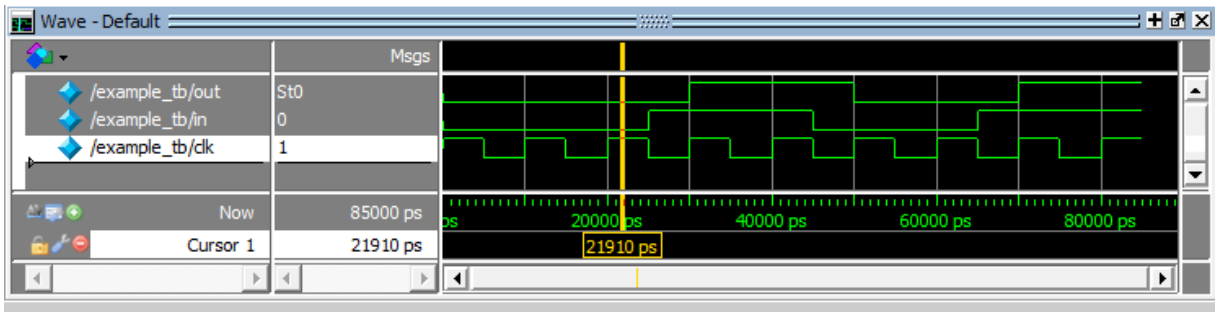
Укажем ModelSim-Altera в качестве инструмента для моделирования. Assignments –> Settings –> EDA Tool Settings. Формат выберем Verilog HDL.



Теперь необходимо настроить параметры моделирования. Заходим в Assignments –> Settings –> EDA Tool Settings –> Simulation. Выбираем Compile test bench, жмем кнопку “Test Benches”. В появившемся окне нажимаем “New…”. В следующем окне необходимо задать путь к файлу тестбенча и добавить его.



Теперь все готово для моделирования. Нажимаем Tools –> Run Simulation Tool –> Run Simulation. После этого запустится программа ModelSim, которая скомпилирует необходимые для моделирования файлы и построит диаграммы на основе заданных в тестбенче сигналов.



После изменения Verilog-файлов не обязательно повторять все эти шаги, достаточно в поле transcript выбрать нажатием клавиши «вверх» скрипт моделирования *.do и нажать «Enter». Моделирование перезапустится с новыми исходными данными.

Часть 4 Компиляция проекта

4.1. Настройка параметров и запуск компиляции проекта

Как было сказано в разделе 1.3. настоящего описания, процедура полной компиляции проекта в Quartus II включает последовательное выполнение следующих этапов: синтаксический анализ и синтез, размещение и трассировка, формирование конфигурационного файла, временной анализ. Все эти этапы представлены в окне задач менеджера проекта, приведенном на рис 4.1.1.

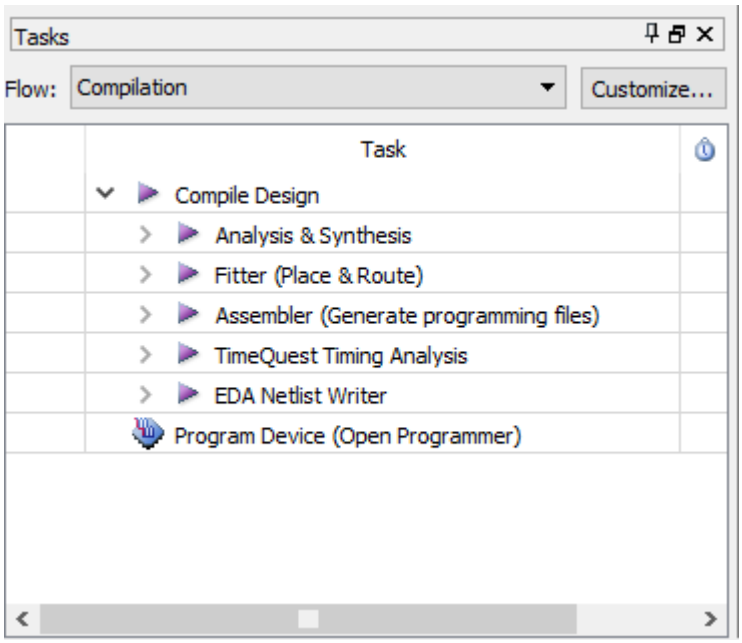


Рис. 4.1.1. Составные этапы компиляции проекта

В свою очередь, отдельные этапы процесса компиляции проекта могут содержать ряд вложенных самостоятельных процедур. Чтобы их увидеть, щелкните мышью по значку ⊞ рядом с названием этапа. На рис.

4.1.2. приведено окно задач, в котором представлены все процедуры, входящие в этап анализа и синтеза проекта. Как видно из рисунка, этот этап включает процедуру анализа назначений контактов ввода - вывода, процедуру выполнения назначений помощником проекта (Design Assistant), процедуру ранней оценки временных характеристик проекта и другие.

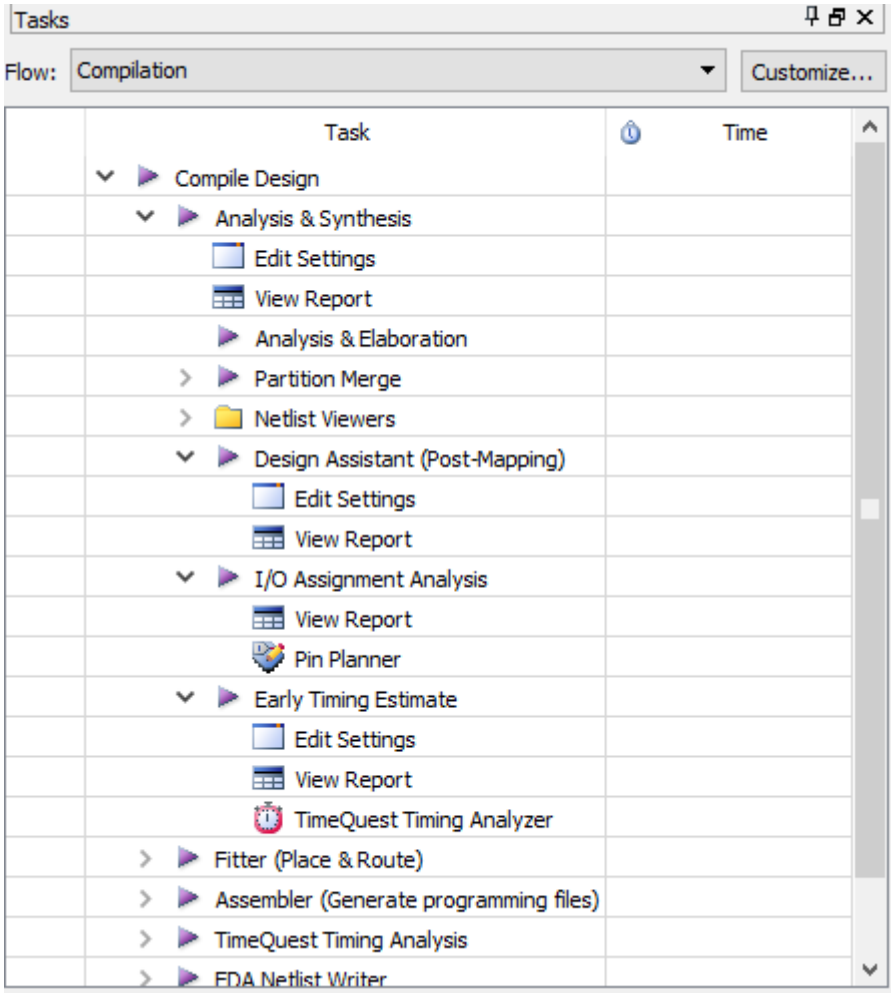


Рис. 4.1.2. Процедуры, запускаемые при выполнении анализа и синтеза проекта

Те процедуры, которые содержат рядом с названием значок ▶, могут быть выполнены путем двойного щелчка мышью по их названию. Однако, в ряде случаев, их выполнение потребует выполнения предыдущих

процедур процесса компиляции. В окне задач содержатся строки *Edit Settings* (см. рис.4.1.2.), двойной щелчок мышью по которым, приведет к появлению на главной панели менеджера проекта окна с установками для соответствующих процедур. Пользователь может выполнить редактирование установок, используя эти окна. На рис. 4.1.3. приведен пример окна с установками для этапа анализа и синтеза проекта.

Как видно из рис. 4.1.3., в качестве критерия оптимизации проекта выбрано быстродействие проекта. Двойной щелчок мышью по строке *View Report* приведет к появлению на главной панели менеджера проекта окна с отчетом о выполнении соответствующей части процесса компиляции. На рис. 4.1.4. приведено окно с отчетом о выполнении анализа и синтеза проекта struct_avt, рассмотренного в разделе 2.

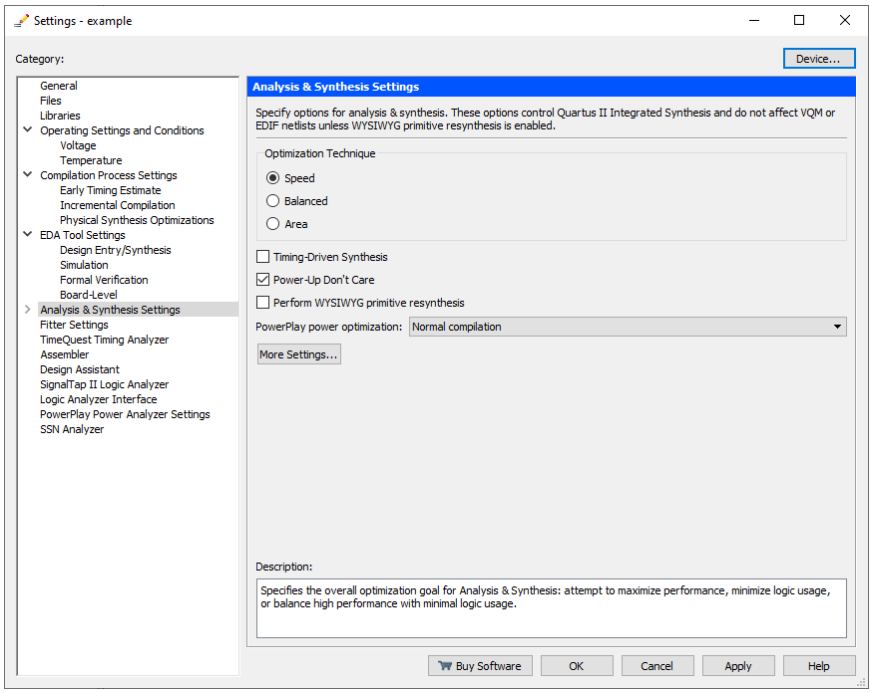


Рис. 4.1.3. Окно установок для этапа анализа и синтеза проекта

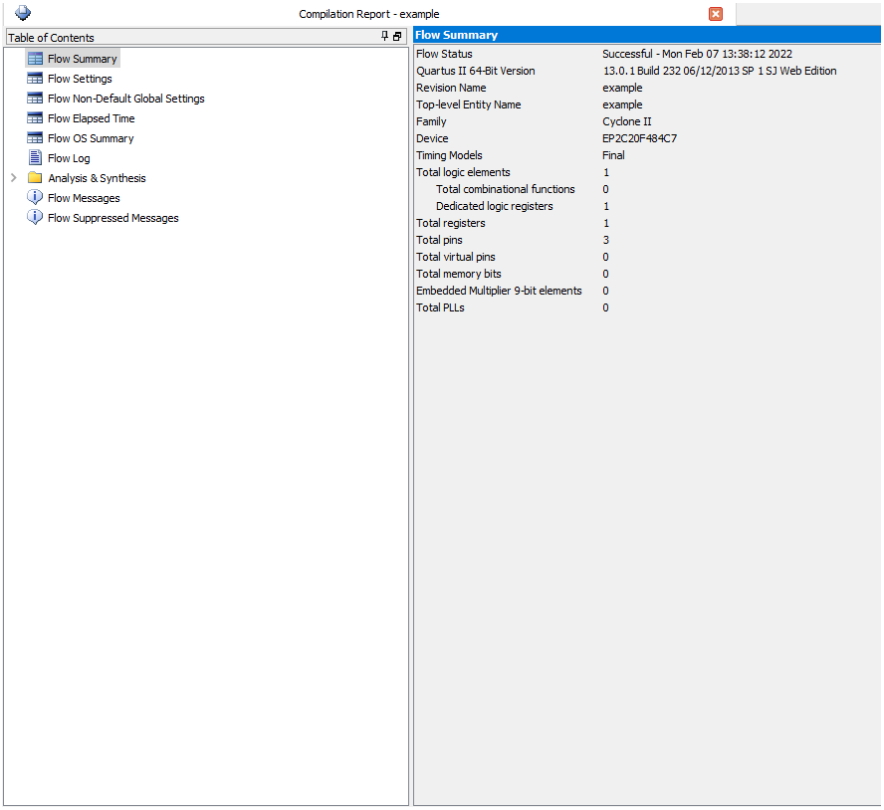


Рис. 4.1.4. Окно с отчетом о выполнении этапа анализа и синтеза проекта

Для настройки параметров каждого этапа компиляции проекта также может быть использована пиктограмма на инструментальной панели или команда *Settings…* из меню **Assignments** менеджера проекта.

Выполнить можно как полную компиляцию всего проекта, так и отдельных составляющих этапов. Для запуска процесса полной компиляции проекта используется пиктограмма на панели инструментов или команда *Start Compilation* из меню **Processing** менеджера проекта.

Для запуска процесса выполнения отдельных этапов или шагов компиляции используется команда *Start* из выпадающего меню **Processing** менеджера проекта. При наведении курсора мыши на эту строку появляется еще одно выпадающее меню, приведенное на рис. 4.1.5., в котором следует указать выполняемый этап компиляции.

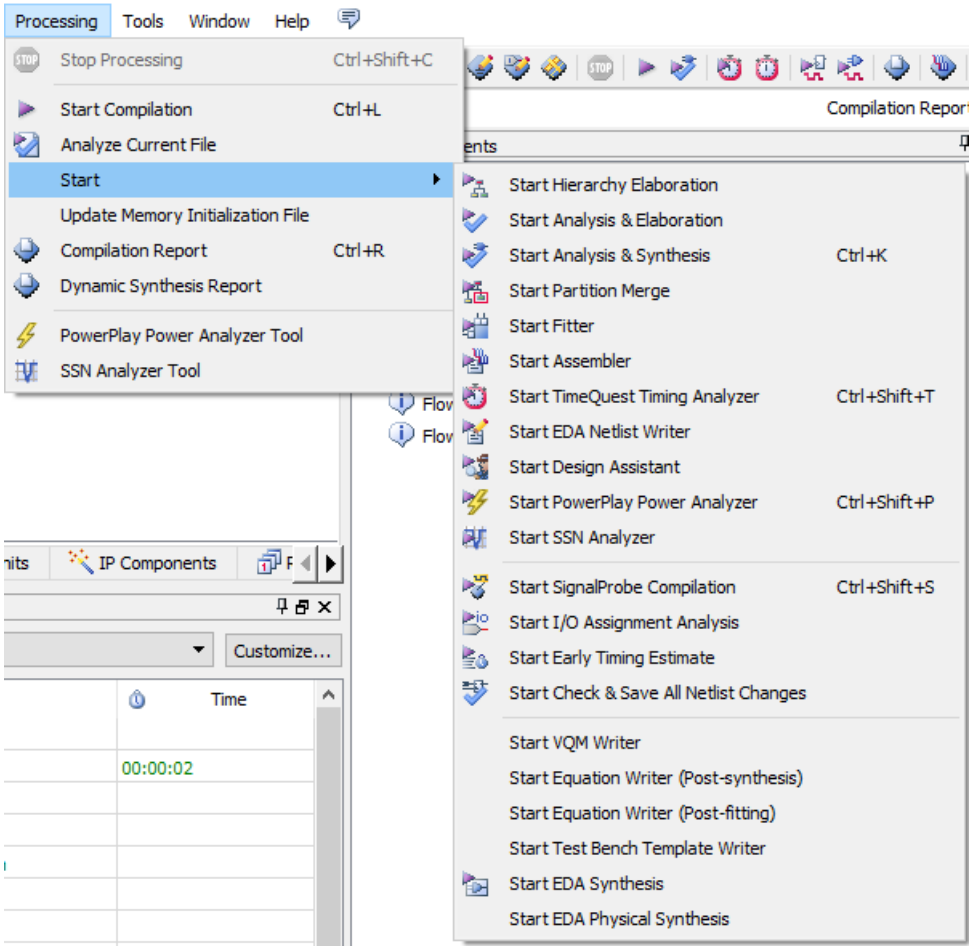


Рис. 4.1.5. Фрагмент меню **Processing** менеджера проекта и выпадающее меню команды *Start*

Для индикации процесса выполнения компиляции или отдельных его этапов может быть использовано окно состояния (Status) менеджера проекта. Чтобы его активировать выполните команду *View>Utility Windows>Status*. Вид окна **Status** приведен на рис. 4.1.6. В поле **Progress%** динамически отображается ход выполнения отдельных этапов компиляции с указанием процента выполненной работы. Это же поле одновременно отображается в окне задач, слева от названия выполняемого этапа компиляции, а справа от названия в поле **Time** указывается время

выполнения этапа компиляции, которое также одновременно отражается в одноименном поле в окне **Status**.

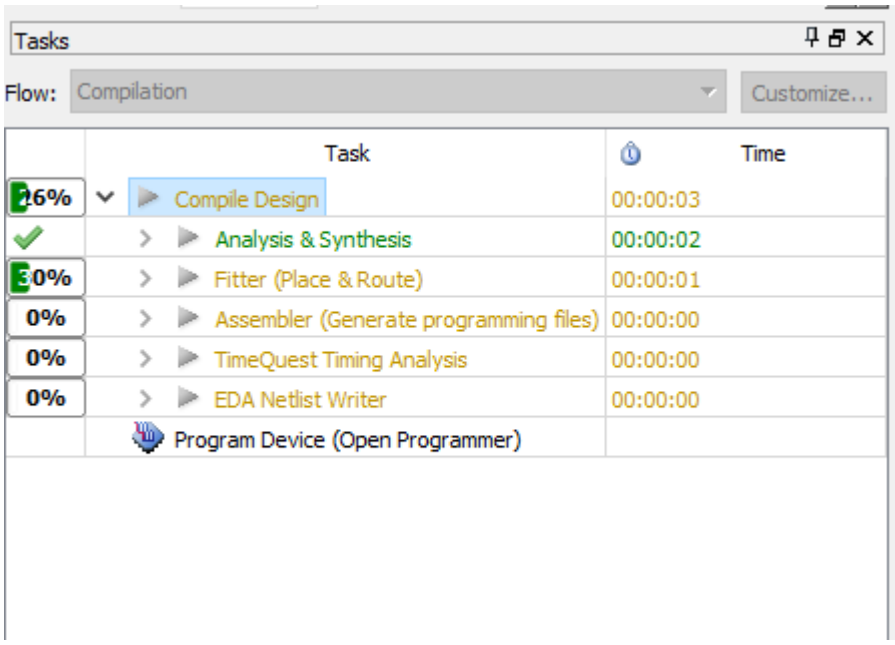


Рис. 4.1.6. Окно состояния менеджера проекта

После успешного завершения этапа компиляции на экран выводится соответствующее сообщение (см. рис. 4.1.7.), а в окне задач выполненный этап изображается зеленым цветом, и слева от его названия устанавливается зеленая галочка.

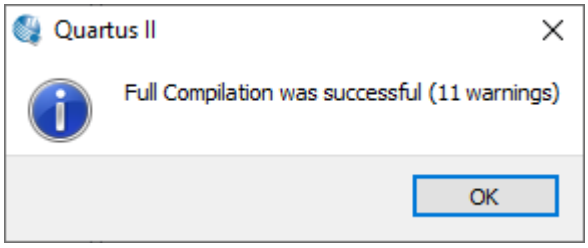


Рис. 4.1.7. Окно с сообщением об успешном завершении компиляции проекта

В окне сообщений менеджера проекта отображается информация о ходе выполнения компиляции, включая предупреждения и сообщения об ошибках. При этом используется следующее соглашение. Сообщения об успешном выполнении шагов компиляции выводятся зеленым цветом и начинаются со слова Info, предупреждения выводятся синим цветом и начинаются со слова Warning, ошибки выводятся красным цветом и начинаются со слова Error.

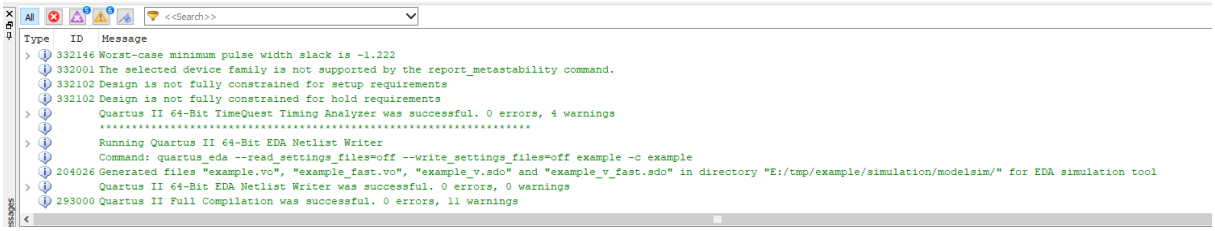


Рис. 4.1.8. Окно сообщений менеджера проекта

На рис. 4.1.8. приведен пример окна сообщений менеджера проекта. В этом окне используется прием закладок, позволяющий путем щелчка мыши по закладке, открыть соответствующую страницу. С помощью закладки Flag можно пометить отдельные сообщения флагом (см. рис. 4.1.8.). С помощью закладки Suppressed можно скрыть появление некоторых сообщений. Использование этой возможности может пригодиться в тех случаях, когда сообщение проанализировано, и вывод его больше не требуется.

Закладки Error, Critical Warning, Warning позволяют открыть страницы с сообщениями об ошибках, критических предупреждениях и предупреждениях, соответственно.

Объектом компиляции всегда является модуль верхнего уровня. Поэтому для выполнения компиляции модуля нижележащего уровня, необходимо объявить его модулем верхнего уровня. Сделать это можно с помощью закладок Hierarchy и Files в окне навигатора проекта, с

использованием контекстно-зависимого меню, в котором следует выполнить команду *Set as Top Level Entity* (см. раздел 1.5.).

4.2. Отображение результатов компиляции проекта

В состав пакета Quartus II входят средства, позволяющие наглядно представить отчет о результатах компиляции проекта, выполнить анализ представленного отчета и внести некоторые изменения в результат компиляции проекта.

Отчет о результатах компиляции появляется сразу после завершения компиляции проекта. Он размещается в главном окне менеджера проекта и состоит из двух частей, как показано на рис. 4.2.1. В левой части приводится иерархическая структура отчета, где информация по выполнению отдельных этапов компиляции находится в соответствующих папках, а в правой части – сам отчет. По умолчанию, в правой части выводится обобщенная информация о результатах компиляции проекта. Чтобы просмотреть другие разделы отчета, нужно в левой части, с помощью щелчка мыши, выбрать интересующий раздел отчета, который отобразится в правой части окна.

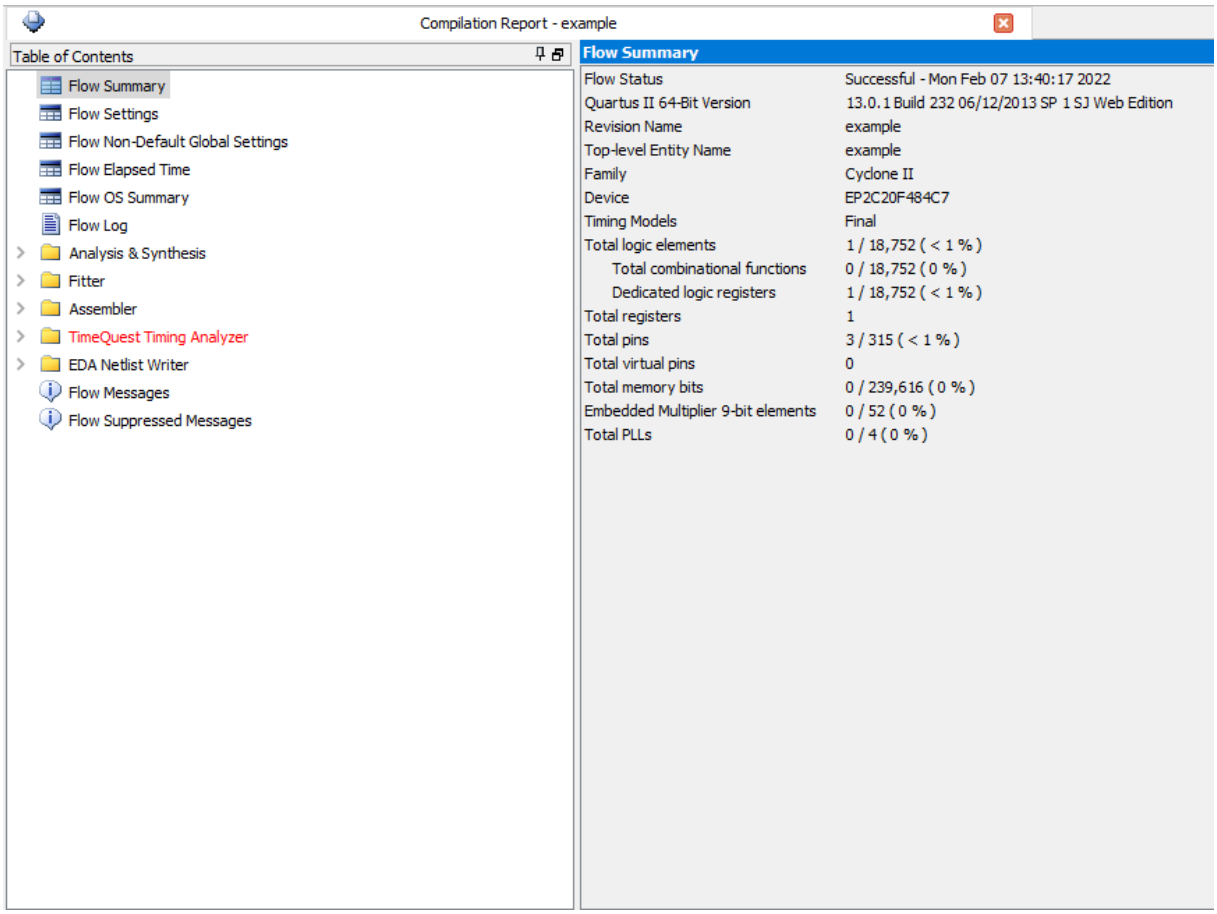


Рис. 4.2.1. Окно отчета о компиляции (раздел Flow Summary)

Окно с отчетом может быть свернуто, развернуто, закрыто стандартным образом. Оно может быть вновь открыто в любое время либо выполнением команды *Processing>Compilation Report*, либо щелчком мыши по пиктограмме на панели инструментов менеджера проекта.

Информация из отчета также доступна в текстовом виде в рабочей папке проекта в файлах <project_name>.fit.rpt и <project_name>.map.rpt.

Ниже приводится описание разделов отчета, которые, являются наиболее информативными для разработчика.

Окно с разделом отчета об используемых ресурсах кристалла для некоторого проекта приведено на рис 4.2.2. Чтобы его просмотреть, раскройте папку *Analysis & Synthesis* в левой части окна отчета с помощью щелчка мыши по значку ⊞, рядом с именем папки. Затем щелкните мышью

по строке *Resource Usage Summary*, после чего появится окно, изображенное на рис. 4.2.2.

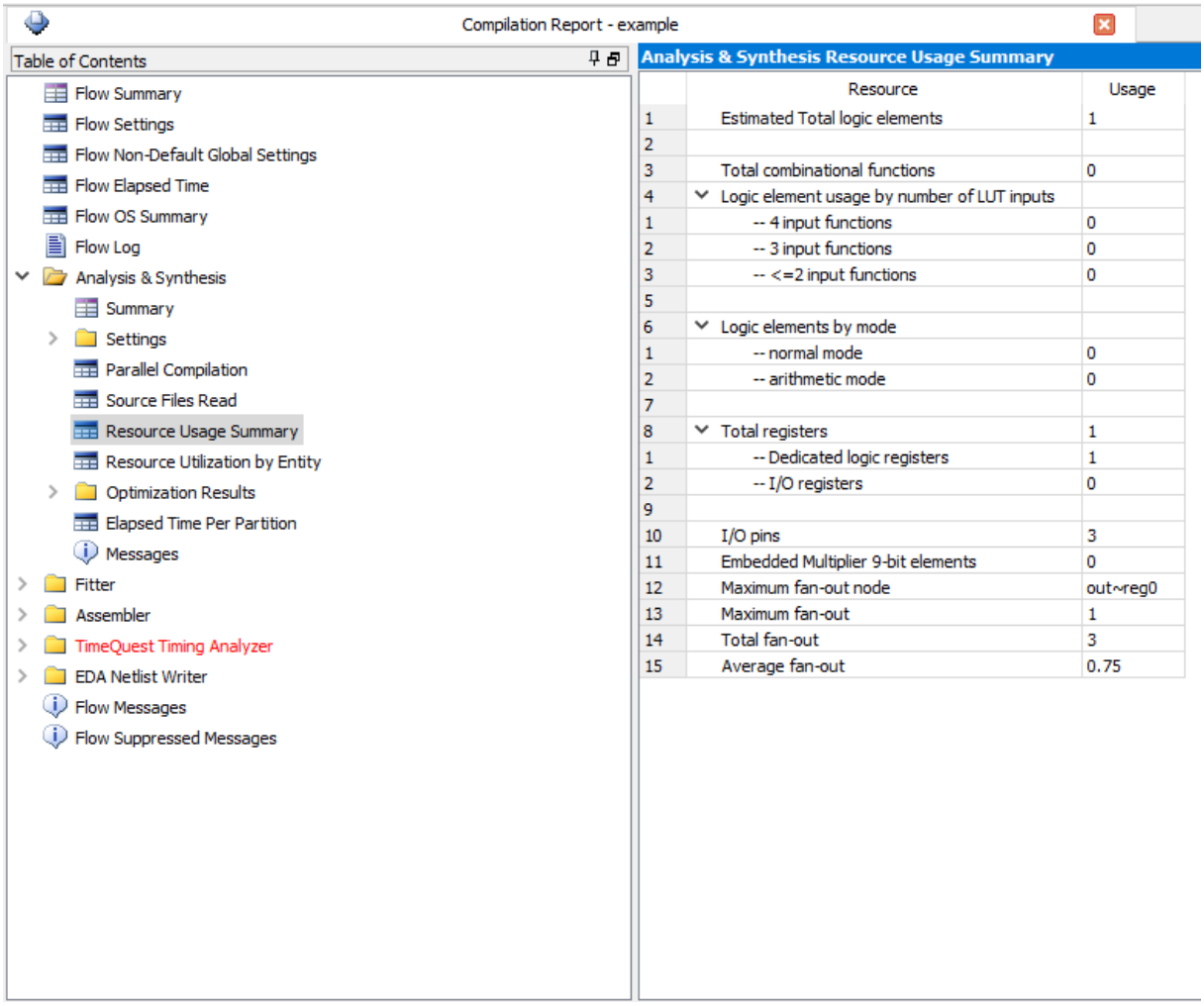
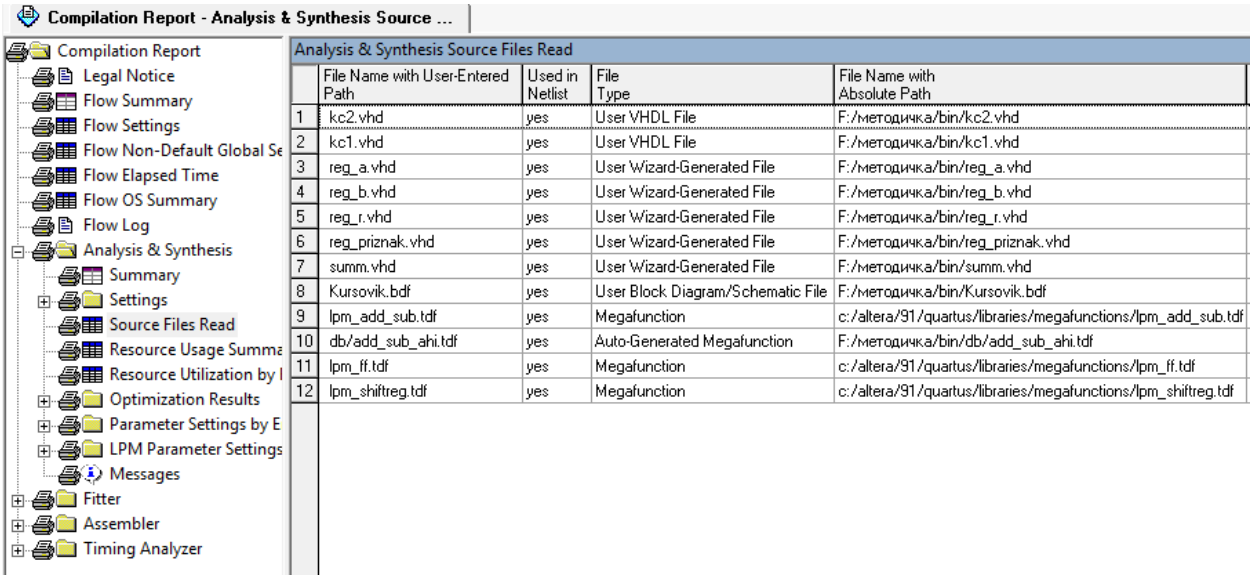


Рис. 4.2.2. Окно отчета об используемых ресурсах кристалла

На рис. 4.2.3. изображен пример раздела отчета для того же проекта, отражающий использованные в процессе компиляции исходные файлы проекта. Чтобы открыть это окно, необходимо щелкнуть мышью по строке *Source Files Read* в разделе отчета из папки *Analysis & Synthesis*.

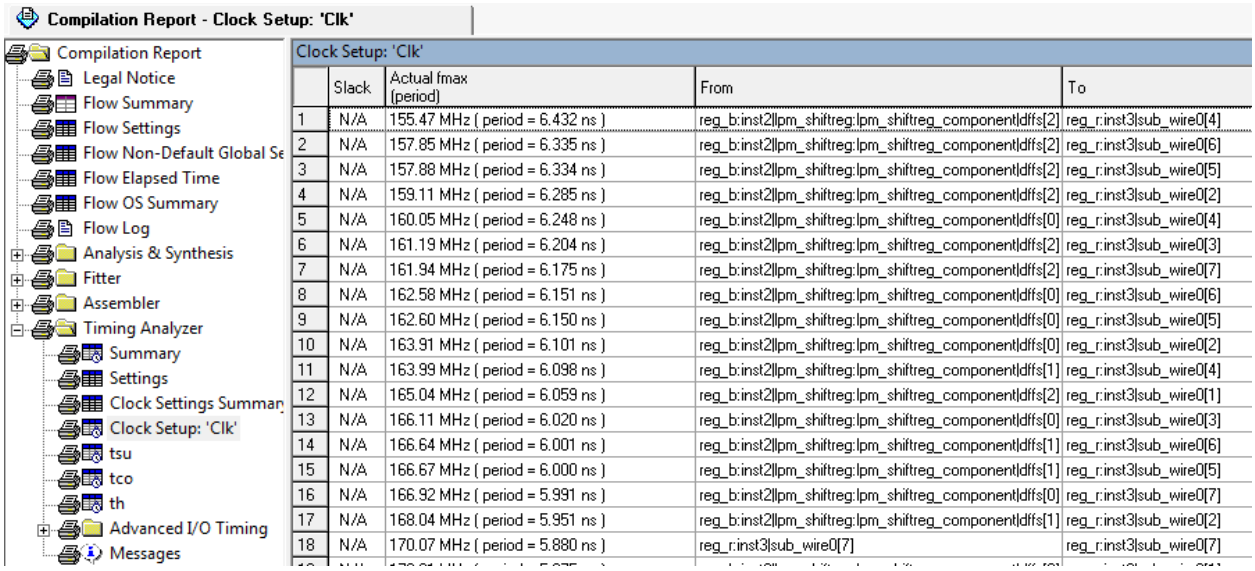


	File Name with User-Entered Path	Used in Netlist	File Type	File Name with Absolute Path
1	kc2.vhd	yes	User VHDL File	F:/методичка/bin/kc2.vhd
2	kc1.vhd	yes	User VHDL File	F:/методичка/bin/kc1.vhd
3	reg_a.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_a.vhd
4	reg_b.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_b.vhd
5	reg_r.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_r.vhd
6	reg_priznak.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_priznak.vhd
7	summ.vhd	yes	User Wizard-Generated File	F:/методичка/bin/summ.vhd
8	Kursovik.bdf	yes	User Block Diagram/Schematic File	F:/методичка/bin/Kursovik.bdf
9	lpm_add_sub.tdf	yes	Megafunction	c:/altera/91/quartus/libraries/megafunctions/lpm_add_sub.tdf
10	db/add_sub_ahi.tdf	yes	Auto-Generated Megafunction	F:/методичка/bin/db/add_sub_ahi.tdf
11	lpm_ff.tdf	yes	Megafunction	c:/altera/91/quartus/libraries/megafunctions/lpm_ff.tdf
12	lpm_shiftreg.tdf	yes	Megafunction	c:/altera/91/quartus/libraries/megafunctions/lpm_shiftreg.tdf

Рис. 4.2.3. Окно отчета с исходными файлами проекта

В этом окне отчета приводится список всех логических файлов, как пользовательских, так и библиотечных, использованных в процессе компиляции проекта. Причем указаны типы файлов и папки, в которых они хранятся.

На рис. 4.2.4. изображен пример раздела отчета для того же проекта, содержащий информацию о результатах временного анализа проекта. Чтобы открыть этот раздел, следует щелкнуть мышью по значку ⊞, рядом с именем папки *Timing Analyzer*. Последовательно открывая вложенные разделы отчета, разработчик может получить исчерпывающую информацию о результатах временного анализа проекта. Так, из отчета он определит максимальную частоту, на которой может работать проект, с указанием критического пути, времена предустановки и времена удержания сигналов, используемых в проекте.



	Slack	Actual fmax (period)	From	To
1	N/A	155.47 MHz (period = 6.432 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[4]
2	N/A	157.85 MHz (period = 6.335 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[6]
3	N/A	157.88 MHz (period = 6.334 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[5]
4	N/A	159.11 MHz (period = 6.285 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[2]
5	N/A	160.05 MHz (period = 6.248 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[0]	reg_r:inst3|sub_wire0[4]
6	N/A	161.19 MHz (period = 6.204 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[3]
7	N/A	161.94 MHz (period = 6.175 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[7]
8	N/A	162.58 MHz (period = 6.151 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[0]	reg_r:inst3|sub_wire0[6]
9	N/A	162.60 MHz (period = 6.150 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[0]	reg_r:inst3|sub_wire0[5]
10	N/A	163.91 MHz (period = 6.101 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[0]	reg_r:inst3|sub_wire0[2]
11	N/A	163.99 MHz (period = 6.098 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[1]	reg_r:inst3|sub_wire0[4]
12	N/A	165.04 MHz (period = 6.059 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[2]	reg_r:inst3|sub_wire0[1]
13	N/A	166.11 MHz (period = 6.020 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[0]	reg_r:inst3|sub_wire0[3]
14	N/A	166.64 MHz (period = 6.001 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[1]	reg_r:inst3|sub_wire0[6]
15	N/A	166.67 MHz (period = 6.000 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[1]	reg_r:inst3|sub_wire0[5]
16	N/A	166.92 MHz (period = 5.991 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[0]	reg_r:inst3|sub_wire0[7]
17	N/A	168.04 MHz (period = 5.951 ns)	reg_b:inst2|lpm_shiftreg:lpm_shiftreg_component|dffs[1]	reg_r:inst3|sub_wire0[2]
18	N/A	170.07 MHz (period = 5.880 ns)	reg_r:inst3|sub_wire0[7]	reg_r:inst3|sub_wire0[7]

Рис. 4.2.4. Окно отчета с результатами временного анализа проекта

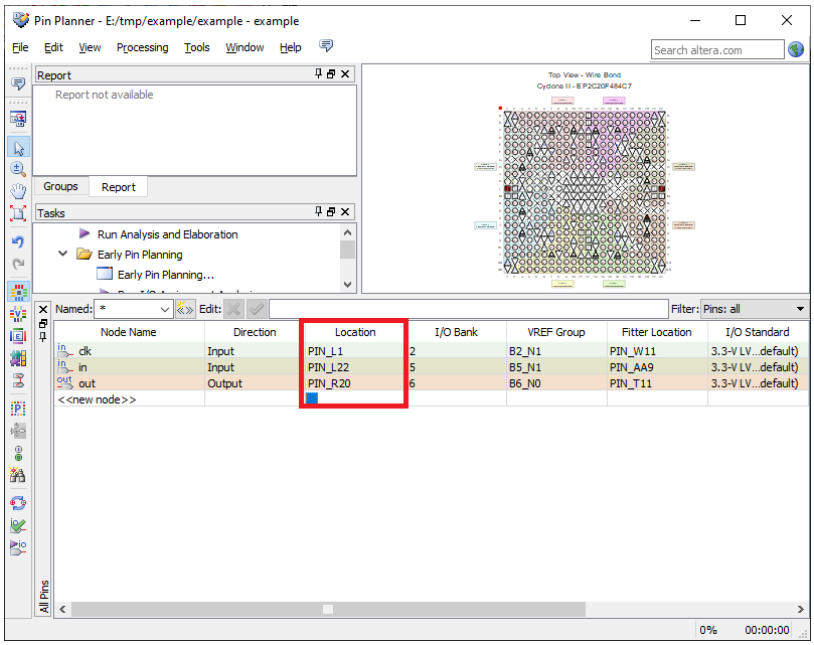
Для просмотра внутреннего представления проекта используется средство Quartus II, называемое ***Netlist Viewers***. С помощью этого средства можно просматривать реализацию проекта на программируемом кристалле на разных уровнях абстракции. Так, ***RTL Viewer*** позволяет просматривать результаты компиляции проекта на уровне регистровых передач. ***Technology Map Viewer*** позволяет просматривать технологическую карту проекта, то есть его реализацию с использованием таких ресурсов кристалла, как функциональные преобразователи (LUT), так и триггеры. Для просмотра результатов компиляции модулей проекта, описанных в виде конечного автомата, предназначено средство ***State Machine Viewer***.

Для просмотра и анализа результатов компиляции проекта предназначено средство ***Chip Planner***, для выполнения анализа и внесения изменений в результаты компиляции – средство ***Resource Property Editor***.

4.3. Задание соответствия портов модуля и выводам ПЛИС

После завершения компиляции проекта необходимо задать соответствие между портами модуля верхнего уровня и выводами

микросхемы ПЛИС. Для этого служит инструмент Pin Planner, открыть который можно из вкладки Assignments.



В нем отображены порты скомпилированного модуля верхнего уровня. Вам необходимо задать каждому из этих портов прописать название вывода ПЛИС.

Часть 5 Программирование кристалла ПЛИС

Успешное завершение компиляции проекта сопровождается созданием конфигурационной информации для выбранной микросхемы ПЛИС. Причем Quartus II создает два конфигурационных файла. Один файл с названием проекта и расширением **.sof** предназначен для программирования кристалла ПЛИС. Другой файл с расширением **.pof** предназначен для загрузки конфигурационного ПЗУ, которое в свою очередь, будет конфигурировать кристалл ПЛИС после подачи на него питания.

Для выполнения программирования кристаллов СБИС в состав пакета Quartus II входит соответствующая утилита. Чтобы её вызвать щелкните мышью по пиктограмме на панели инструментов менеджера проекта, либо выполните команду *Programmer* из меню **Tools** как показано на рис. 5.1.

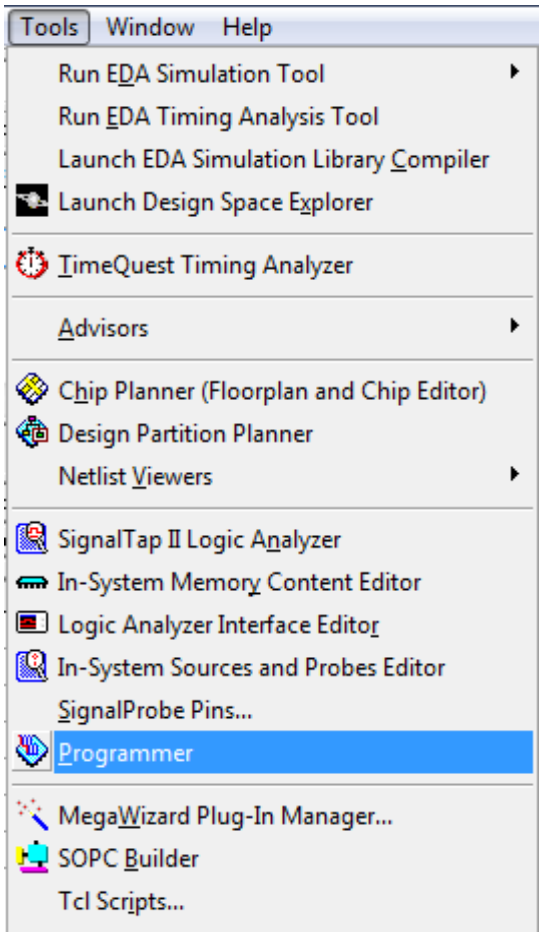


Рис. 5.1. Выпадающее меню команды **Tools** менеджера проекта

Появится окно утилиты программирования, приведенное на рис 5.2. Основное окно на панели (см. рис.5.2.) предназначено для отображения файлов с конфигурационной информацией.

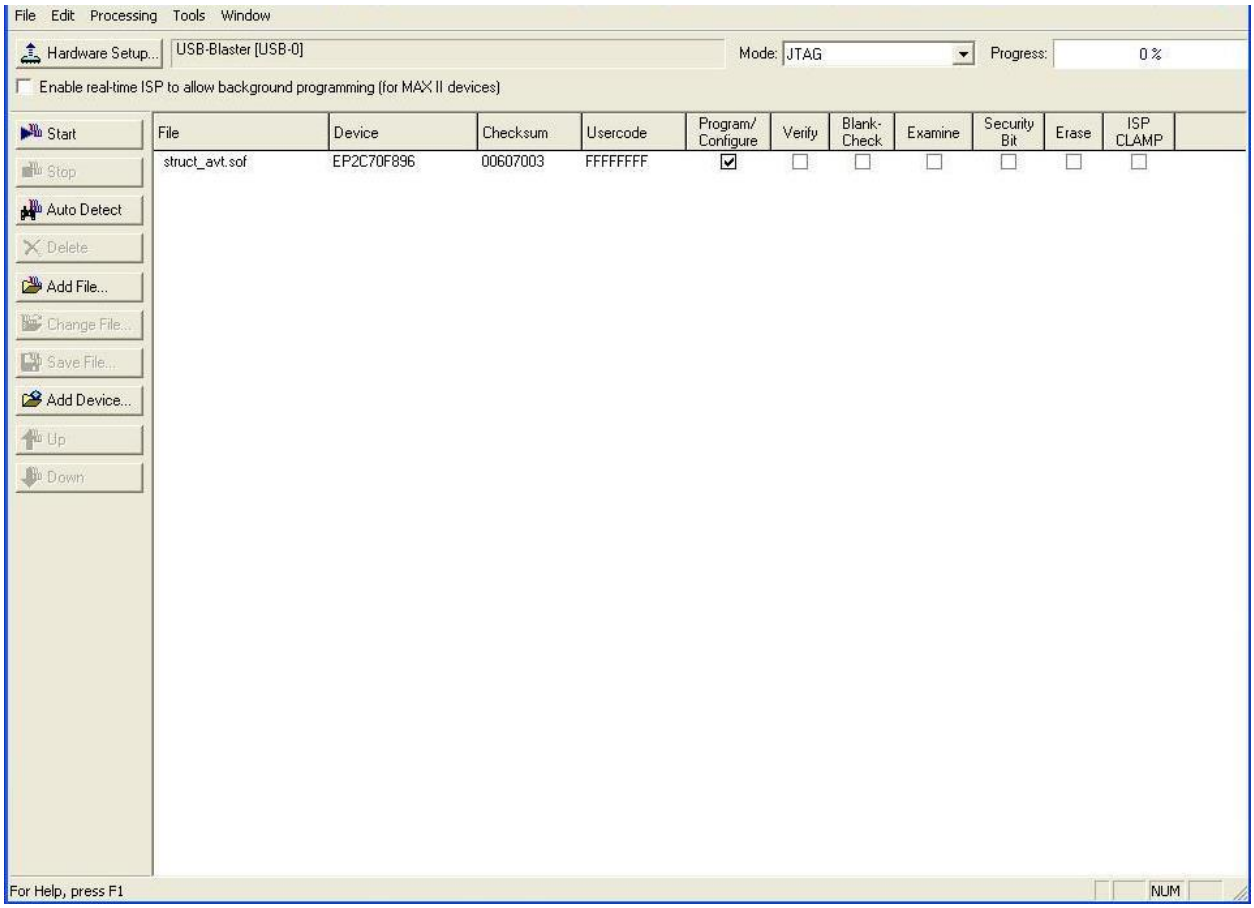
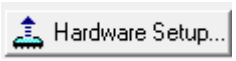


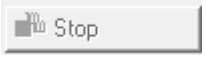
Рис. 5.2. Окно утилиты программирования кристалла

Вдоль левой стороны окна расположены инструментальные кнопки. Их назначение следующее.



- установка аппаратных средств программирования

запуск программирования кристалла



- останов программирования кристалла

- обнаружение микросхемы ПЛИС в JTAG цепи

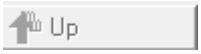
- удаление файла из списка

- добавление файла в список

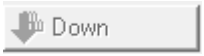
- изменение файла

- сохранение файла

- добавление ПЛИС в JTAG цепь



- выделение файла в списке, соседнего сверху



- выделение файла в списке, соседнего снизу

Нажатие первой кнопки из приведенного списка вызовет появление на экране монитора окна **Hardware Setup**, приведенного на рис. 5.3. Первая страница этого окна с закладкой Hardware Setting позволит выбрать подходящий программатор из представленного в большом окне списка. В лабораторных стендах DE0, DE2-70, Minidilab, Dilab используется программатор USB Blaster. Причем, в первых трех стендах он является встроенным, а в четвертом – установлен в коммуникационном соединителе. Поэтому при заполнении окна следует указать именно его.

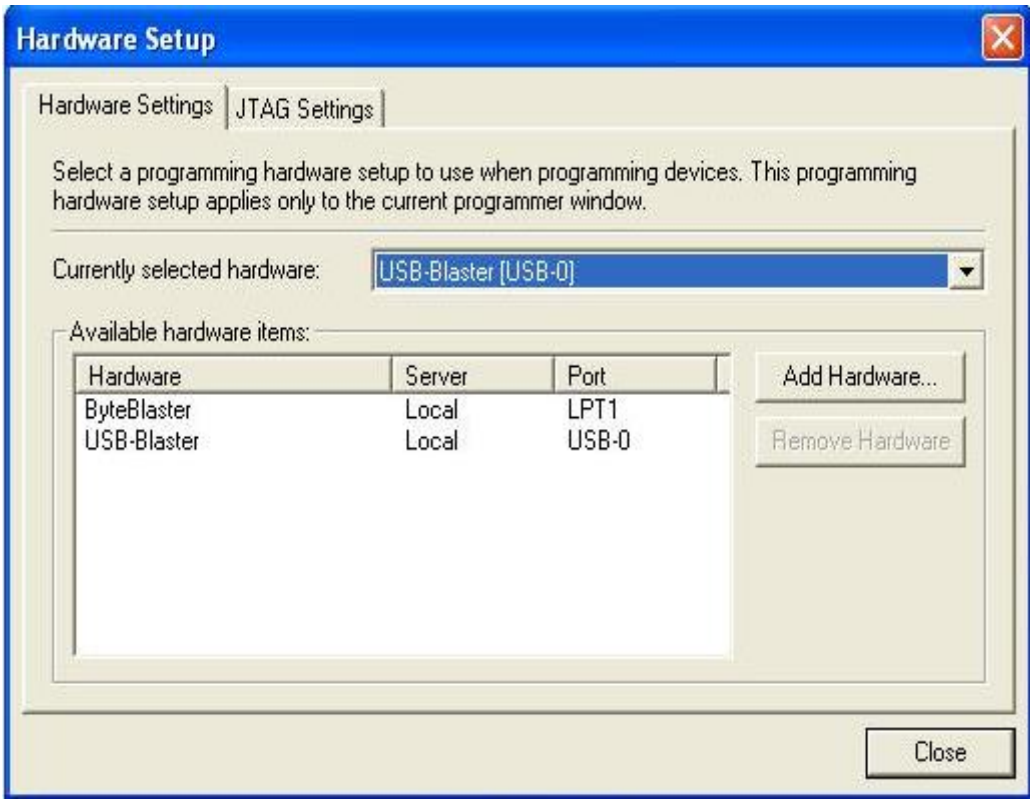


Рис. 5.3. Окно настройки программатора

Вторая страница, с закладкой JTAG Settings позволит выделить JTAG серверы, которые следует добавить или удалить из представленного списка.

Чтобы выполнить программирование кристалла ПЛИС выполните следующие действия.

1. Добавьте конфигурационный файл из файловой системы инструментального компьютера в основное поле окна программатора, 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%.

4. Закройте окно программатора обычным образом.

Ниже приведены таблицы соответствия выводов ПЛИС элементам управления и индикации платы, которые потребуются при выполнении лабораторных работ.

Таблица 8.5.1 Соответствие выводом микросхемы кнопкам переключателей.

Signal	FPGA Pin	Descriptio
SW[0	PIN L22	Toggle
SW[1	PIN L21	Toggle
SW[2	PIN M22	Toggle
SW[3	PIN V12	Toggle
SW[4	PIN W12	Toggle
SW[5	PIN U12	Toggle
SW[6	PIN U11	Toggle
SW[7	PIN M2	Toggle
SW[8	PIN M1	Toggle
SW[9	PIN L2	Toggle

Таблица 8.5.2 Соответствие выводом микросхемы тактовым кнопкам.

Signal	FPGA Pin	Descriptio
KEY[0	PIN R22	Pushbutton[
KEY[1	PIN R21	Pushbutton[
KEY[2	PIN T22	Pushbutton[
KEY[3	PIN T21	Pushbutton[

Таблица 8.5.3 Соответствия выводом микросхемы светодиодам.

Signal	FPGA Pin	Description

LEDR[0]	PIN_R2	LED Red[0]
LEDR[1]	PIN_R1	LED Red[1]
LEDR[2]	PIN_U	LED Red[2]
LEDR[3]	PIN_Y	LED Red[3]
LEDR[4]	PIN_T1	LED Red[4]
LEDR[5]	PIN_V	LED Red[5]
LEDR[6]	PIN_Y	LED Red[6]
LEDR[7]	PIN_U	LED Red[7]
LEDR[8]	PIN_R1	LED Red[8]
LEDR[9]	PIN_R1	LED Red[9]
LEDG[0]	PIN_U	LED Green[0]
LEDG[1]	PIN_U	LED Green[1]
LEDG[2]	PIN_V	LED Green[2]
LEDG[3]	PIN_V	LED_Green[3]
LEDG[4]	PIN_W	LED_Green[4]
LEDG[5]	PIN_W	LED_Green[5]
LEDG[6]	PIN_Y	LED_Green[6]
LEDG[7]	PIN_Y	LED_Green[7]

Таблица 8.5.4 Соответствия выводом микросхемы выводам семисегментных индикаторов.

Signal	FPGA Pin	Description
HEX0[0	PIN_J2	Seven-Segment segment 0[0]
HEX0[1	PIN_J1	Seven-Segment segment 0[1]
HEX0[2	PIN_H2	Seven-Segment segment 0[2]
HEX0[3	PIN_H1	Seven-Segment segment 0[3]
HEX0[4	PIN_F2	Seven-Segment segment 0[4]
HEX0[5	PIN_F1	Seven-Segment segment 0[5]

HEX0[6	PIN_E2	Seven-Segment segment 0[6]
HEX1[0	PIN_E1	Seven-Segment segment 1[0]
HEX1[1	PIN_H6	Seven-Segment segment 1[1]
HEX1[2	PIN_H5	Seven-Segment segment 1[2]
HEX1[3	PIN_H4	Seven-Segment segment 1[3]
HEX1[4	PIN_G3	Seven-Segment segment 1[4]
HEX1[5	PIN_D2	Seven-Segment segment 1[5]
HEX1[6	PIN_D1	Seven-Segment segment 1[6]
HEX2[0	PIN_G5	Seven-Segment segment 2[0]
HEX2[1	PIN_G6	Seven-Segment segment 2[1]
HEX2[2	PIN_C2	Seven-Segment segment 2[2]
HEX2[3	PIN_C1	Seven-Segment segment 2[3]
HEX2[4	PIN_E3	Seven-Segment segment 2[4]
HEX2[5	PIN_E4	Seven-Segment segment 2[5]
HEX2[6	PIN_D3	Seven-Segment segment 2[6]
HEX3[0	PIN_F4	Seven-Segment segment 3[0]
HEX3[1	PIN_D5	Seven-Segment segment 3[1]
HEX3[2	PIN_D6	Seven-Segment segment 3[2]
HEX3[3	PIN_J4	Seven-Segment segment 3[3]
HEX3[4	PIN_L8	Seven-Segment segment 3[4]
HEX3[5	PIN_F3	Seven-Segment segment 3[5]
HEX3[6	PIN_D4	Seven-Segment segment 3[6]

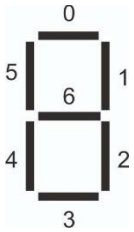


Рис. 8.1.2. Индексы и расположение сегментов индикатора.

Диоды на семисегментном индикаторе загораются при подаче на них низкого напряжения (0 - горит, 1 - не горит).

Таблица 8.5.5 Соответствия выводов микросхемы источникам тактового сигнала.

Signal	FPGA Pin	Description
CLOCK_2	PIN_D	27 MHz clock
CLOCK_5	PIN_L1	50 MHz clock
CLOCK_2	PIN_B1	24 MHz clock
EXT_CL	PIN_M	External (SMA)

Задание лабораторной работы.

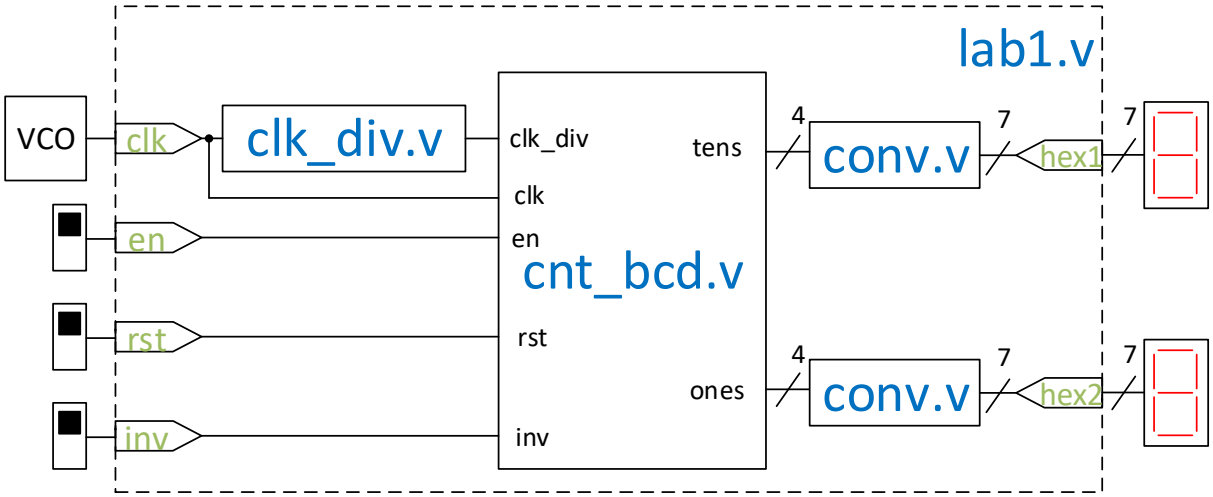


Рис. 9 Блок-схема устройства.

Выполнив описание устройства на языке Verilog и, прошив ПЛИС, продемонстрировать корректную работу устройства.