МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Черниговский государственный технологический университет

# СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ QUARTUS II

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для самостоятельного изучения и подготовки
к лабораторным работам по дисциплине “Большие интегральные микросхемы микропроцессоров и программируемой логики”
для студентов направления подготовки 6.050102 “Компьютерная инженерия”

УТВЕРЖДЕНО
На заседании кафедры
информационных и
компьютерных систем
Протокол № 8 от 19.03.2013г.

Чернигов ЧГТУ 2013

Система автоматизованого проектування Quartus II. Методичні вказівки для самостійного вивчення та підготовки до лабораторних робіт з дисципліни “Великі інтегральні схеми мікропроцесорів і програмованої логіки” для студентів за напрямом підготовки 6.050102 “Комп'ютерна інженерія”/ Укл. Вервейко О.І., Заровський Р.В., Красножон О.В. - Чернігів: ЧДТУ. – 2013. – 87с. Рос. мовою.

Составители: Вервейко Александр Иванович, кандидат технических наук, доцент кафедры информационных и компьютерных систем
Заровский Руслан Владиславович, кандидат технических наук, доцент кафедры промышленной электроники и радиоэлектронных аппаратов
Красножон Алексей Васильевич, ассистент кафедры информационных и компьютерных систем

Ответственный за выпуск: Казимир Владимир Викторович, заведующий кафедрой информационных и компьютерных систем, доктор технических наук, профессор

Рецензент: Иванец Сергей Анатольевич, кандидат технических наук, доцент кафедры промышленной электроники и радиоэлектронных аппаратов Черниговского государственного технологического университета

**Содержание**

## Предисловие

При разработке специализированных цифровых устройств уже давно используют высокотехнологическую базу – программируемые логические интегральные схемы (ПЛИС) или СБИС ПЛ (programmable logic device – PLD). СБИС ПЛ используются в различных областях для создания специализированных контроллеров, в системах телекоммуникаций, цифровой обработке сигналов и т.д. СБИС ПЛ оказываются вне конкуренции при создании высокопроизводительных специализированных устройств, ориентированных на аппаратную реализацию. Аппаратное решение различного рода задач обеспечивает распараллеливание процесса обработки и увеличение производительности в десятки раз по сравнению с программным решением. А использование СБИС ПЛ обеспечивает такую же гибкость модификации, как у любых программных решений. Развитие элементной базы СБИС ПЛ позволило создавать на кристалле стандартные процессорные ядра и решать практически любые задачи по постарению программно-аппаратных систем на одной микросхеме с использованием единых средств проектирования и отладки.

Разработчик специализированного цифрового устройства, используя средства САПР СБИС ПЛ, в привычной ему форме (схемы, текстовое описание) задает требуемое устройство и получает программирующий файл, который используется для конфигурации ПЛИС. Программирование заключается в задании нужных свойств функциональным преобразователям и установлении необходимых связей между ними. Такой цикл проектирования/изготовления занимает незначительное время и изменения могут вноситься на любой стадии разработки устройства. Внедрение новых средств проектирования на начальном этапе практически не требует материальных затрат благодаря низкой стоимости микросхем и наличию бесплатных полнофункциональных версий САПР.

Фирма Altera производит СБИС ПЛ нескольких принципиально различающихся семейств: MAX3000A, MAX7000, MAX9000, FLEX10K, ACEX1K, APEX20K, Stratix, а также ряд других семейств.

СБИС ПЛ, входящие в эти семейства отличаются:

– степенью интеграции (логической емкостью);

– архитектурой функционального преобразователя;

– организацией внутренней структуры СБИС и структуры матрицы соединений функциональных преобразователей;

– типом используемого программируемого элемента;

– наличием внутренней оперативной памяти;

– наличием специализированных встроенных модулей.

Современная элементная база предполагает использование новых технологий проектирования и современных средств проектирования. Фирма Altera предлагает разработчикам систему автоматизированного проектирования цифровых устройств на базе микросхем программируемой логики Quartus II. Это САПР, поддерживающий работу со всеми новыми семействами СБИС ПЛ,

обеспечивает доступ ко всем ресурсам микросхемы и позволяет проводить проектирование программно-аппаратных систем любой сложности.

# 1 ОСНОВНЫЕ СВЕДЕНИЯ О САПР QUARTUS II

Программное обеспечение Quartus II фирмы Altera представляет полную мультиплатформенную среду проектирования, которая включает в себя все этапы проектирования цифрового устройства на ПЛИС при помощи различных языков описания цифровых устройств. Программное обеспечение Quartus II включает в себя средства для всех этапов проектирования с применением ПЛИС как FPGA, так и CPLD структур. На рисунке 1.1 представлена общая структурная схема проектирования в среде Quartus II.

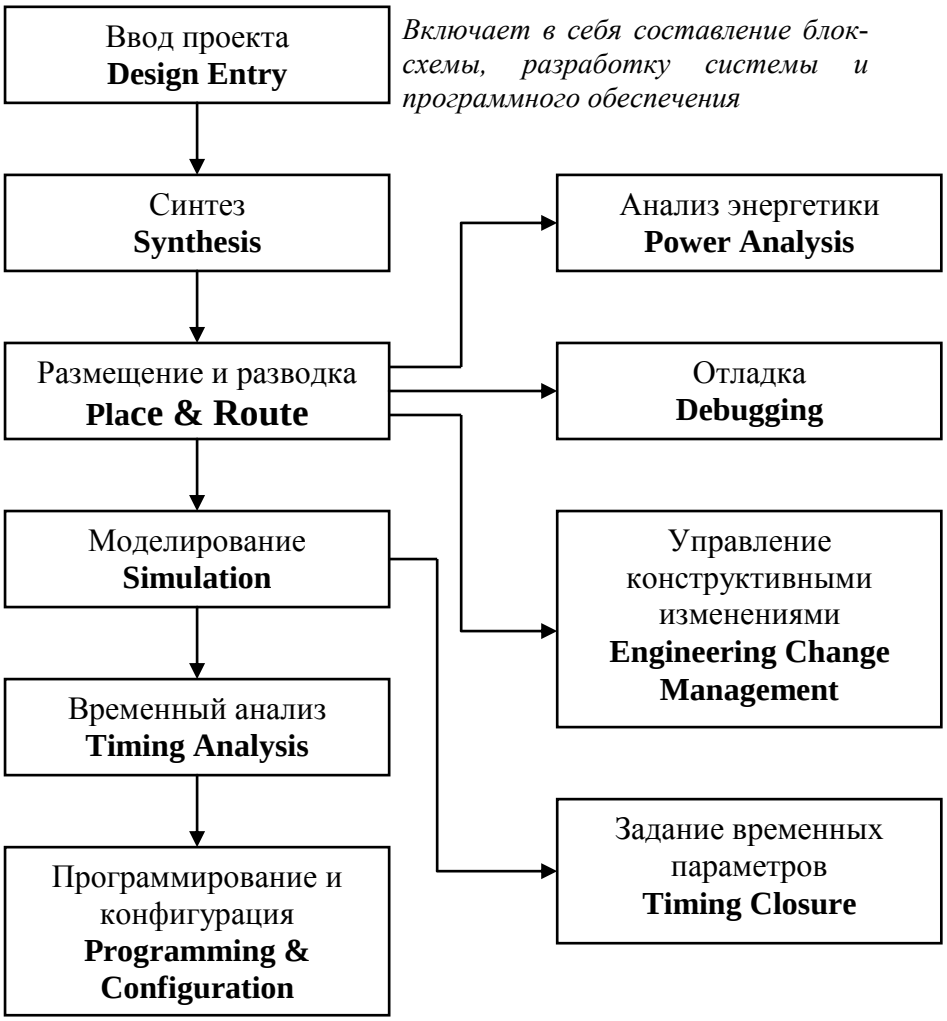



Рисунок 1.1 – Общая структурная схема проектирования в среде Quartus II

Quartus II относится к САПР ПЛИС (EDA Tool), то есть поддерживает весь цикл проектирования цифровых устройств на основе программируемой и реконфигурируемой логики, например, ПЛИС семейства MAX3000, Flex10K, APEX20K, ACEX1K, Cyclon.

САПР Quartus II содержит в своем составе:

– различные средства разработки цифровых устройств (графический, текстовый, схемный редакторы и т.д.);

– компилятор и редактор для размещения разработанной схемы на логику целевого устройства;

– средства анализа временных характеристик устройства (время задержки сигнала между входом и выходом, максимальная тактовая частота работы устройства и т.д.);

– редактор временных диаграмм для тестирования и отладки разрабатываемого устройства;

– программатор для переноса конфигурации устройства из проекта в СБИС ПЛ.

Аналогами и конкурентами Quartus II являются MAX+PLUS II той же фирмы Altera (предыдущая версия Quartus II), Xilinx ISE фирмы Xilinx и Active HDL.

Дополнительно Quartus II предоставляет возможность использовать графический интерфейс пользователя, EDA или интерфейс командной строки в каждом этапе разработки устройства. Можно использовать какой-либо интерфейс для всего процесса проектирования или различные настройки для каждого отдельного этапа разработки.

## 1.1 Возможности САПР Quartus II

Основными возможностями САПР Quartus II являются:

– различные способы ввода поведенческих и структурных описаний устройств;

– интегрированные средства помощи для создания сложных проектов MegaWizard® & SOPC;

– подсистема синтеза;

– подсистема размещения внутренних ресурсов и разводки СБИС;

– подсистема моделирования;

– подсистема временного анализа и анализа потребляемой энергии;

– подсистема программирования СБИС;

– подсистема оптимизации быстродействия проекта – LogicLock™;

– подсистема поддержки интеграции с другими средствами автоматизации проектирования – NativeLink®;

– система проектирования блоков цифровой обработки сигналов – DSP Builder;

– интегрированные средства разработки программного обеспечения для встраиваемых микроЭВМ;

– поддержка использования IP-модулей;

– встроенные средства отладки СБИС в составе системы SignalTap® II & SignalProbe™;

– поддержка операционных систем Widows, Solaris и Linux;

– поддержка различных схем лицензирования (nodelocked, network).

## 1.2 Особенности САПР Quartus II

Средство разработки Quartus II – это следующий шаг в проектировании устройств с высокой степенью интеграции, включая разработку законченных систем на одном программируемом кристалле (System-on-a-programmable-chip (SOPC)).

Quartus II объединяет в себе проектирование, синтез, размещение элементов, трассировку соединений и верификацию, связь с системами проектирования других производителей.

Разработка систем на кристалле требует от разработчиков эффективной командной работы. Изменения в одной части проекта должно иметь минимальное влияние на других членов команды. Программное обеспечение Quartus II – это наиболее комплексная среда для разработки систем на кристалле SOPC, доступная в настоящее время. Новые особенности СПАР Quartus II.

**LogicLock** – это новая блочная методология проектирования, доступная исключительно в программном обеспечении Quartus II. Quartus II совместно с LogicLock – единственное программное обеспечение для разработки устройств на основе программируемой логики, которое включает в себя блочную методологию проектирования как стандартную функцию. Это помогает увеличить эффективность работы разработчиков, снизить время проектирования и верификации. LogicLock позволяет проектировать и проверять каждый модуль отдельно. Разработчики могут объединять готовые модули в проект верхнего уровня, сохраняя производительность каждого модуля в процессе объединения. LogicLock снижает время разработки и верификации, поскольку каждый модуль оптимизируется только один раз.

**NativeLink** – позволяет осуществлять связь между средством разработки Quartus II и программным обеспечением других производителей. NativeLink позволяет средствам синтеза сторонних производителей преобразовывать свои примитивы напрямую в примитивы устройств Altera. Прямое преобразование сокращает время компиляции и освобождает от использования дополнительных библиотек трансляций преобразований, которые могут ограничить производительность, достигнутую средствами проектирования сторонних производителей. Процесс разработки NativeLink позволяет разработчикам использовать Quartus II для размещения элементов, а средства проектирования других производителей - для оптимизации стратегий синтеза.

Технология размещения элементов и трассировки соединений **PowerFit** в программном обеспечении Quartus II использует временные параметры, заданные разработчиком, для оптимального составления схемы и размещения логических элементов. Интеллектуальный алгоритм трассировки по временным параметрам в программном обеспечении Quartus II уделяет первостепенное внимание соединениям, критичным к временным параметрам. Критичные к временным параметрам соединения оптимизируются в первую очередь, для уменьшения задержек и достижения максимальной производительности ($f_{MAX}$). Дальнейшее улучшение параметра $f_{MAX}$ достигается использованием новейшей

архитектуры, такой как в семействе устройств Stratix. Эта передовая технология размещения элементов и трассировки соединений помогает пользователям программного обеспечения Quartus II достичь максимальной производительности и обладает самым малым временем компиляции проекта среди подобных средств разработки.

**Верификация**. Проверка или верификация проекта может оказаться самой продолжительной стадией в процессе разработки высокопроизводительных систем на кристалле. Однако, используя Quartus II, можно сократить время верификации, поскольку это программное обеспечение обладает набором собственных средств верификации, интегрированных с последними средствами верификации сторонних фирм.

**Анализ**. Altera разработала два метода, для того, чтобы помочь разработчикам проанализировать состояние внутренних точек и входов/выходов устройства. Это отладочное средство SignalProbe и логический анализатор SignalTap. Технологии SignalTap и SignalProbe могут работать совместно со средствами синтеза сторонних производителей и не требуют внесения изменений в исходный HDL файл проекта.

Доступная в последних версиях программного обеспечения Quartus II технология аппаратной отладки **SignalProbe** позволяет пользователям последовательно соединять внутренние точки устройства со свободными зарезервированными выводами для анализа с помощью осциллографа или логического анализатора.

Для многих разработчиков, которые используют корпуса BGA с большим количеством входов/выходов, верификация системного уровня занимает очень много времени и иногда сильно затруднена. Логический анализатор **SignalTap** производит верификацию, с помощью интеграции функциональности логического анализатора в программном обеспечении. SignalTap позволяет разработчикам собрать данные с любых внутренних точек и входов/выходов устройства в режиме реального времени при работе системы. Quartus II вставляет в проект мегафункцию, содержащую логический анализатор. Данные собираются и сохраняются в блоках встроенной памяти устройства и направляются в программное обеспечение Quartus II через загрузочный кабель. Разработчики также могут подать внутренние сигналы на выводы устройства для дальнейшего мониторинга. Логический анализатор SignalTap позволяет существенно снизить время верификации, что позволяет в более короткие сроки выпускать новые продукты.

Программное обеспечение Quartus II включает технологию **PowerGauge** - первое интегрированное средство анализа энергопотребления. Средство анализа PowerGauge использует файлы, созданные в процессе моделирования для того, чтобы связать оценку потребления энергии с заданными параметрами устройства. Используя симулятор Quartus II или симуляторы сторонних производителей, интегрированный анализатор энергопотребления позволяет потребителям Altera установить и оптимизировать потребление энергии на более ранней стадии процесса разработки.

## 2 ПОНЯТИЕ ПРОЕКТА В САПР QUARTUS II

Под термином «проект» в рамках САПР Quartus II понимается набор файлов, связанных с проектируемым модулем, в котором выделяются две группы файлов:

– логические файлы, описывающие алгоритм работы устройства (**Design Files**);

– вспомогательные файлы(**Ancilary Files**). Проект может содержать один логический файл либо несколько логических файлов, образующих иерархическое описание проектируемого модуля.

При иерархическом описании среди множества логических файлов различают:

– файл верхнего уровня в иерархии описаний (**Top-level Design File**);

– файлы нижних (одного или нескольких) уровней иерархии (**Low-level Design files**).

В файле верхнего уровня иерархии задается архитектура модуля, определяется набор модулей, входящих в его состав как компоненты, и их взаимосвязь. Описания этих модулей содержатся в логических файлах более низкого уровня иерархии. В их состав, в свою очередь, в виде компонентов также могут входить модули, описания которых приведены в логических файлах еще более низкого уровня иерархии, и т. д.

Имя проекта должно совпадать с именем модуля верхнего уровня в иерархии описаний, а, следовательно, и именем логического файла, в котором хранится его описание. Имена модулей нижних уровней иерархии, в свою очередь, должны совпадать с именами файлов, в которых они описаны.

Логический файл – это файл одного из следующих типов:

– **Block Design File** (стандартное расширение – **.bdf**). Файл содержит блок-схему устройства, созданную в рамках САПР Quartus II;

– **Graphic Design File** (стандартное расширение – **.gdf**). Файл содержит принципиальную электрическую схему, созданную в рамках САПР Quartus II;

– **AHDL Text Design File** (стандартное расширение – **.tdf**).Файл содержит текстовое описание модуля на языке AlteraHDL;

– **State Machine File**(стандартное расширение – **.smf**). Файл содержит графическое или табличное описание цифрового автомата;

– **VHDL Design File** (стандартное расширение – **.vhd**). Файл содержит текстовое описание модуля на языке VHDL;

– **Verilog Design File** (стандартное расширение – **.v**). Файл содержит текстовое описание модуля на языке Verilog HDL;

– **Orcad Schematic Files** (стандартное расширение – **.sch**). Файл содержит схему, созданную в рамках САПР ORCAD;

– **EDIF Input Files** (стандартное расширение – **.edf**). Файл содержит описание в формате EDIF 200 или 300;

– Xilinx Netlist Format File (стандартное расширение – .xnf).Файл содержит описание модуля, полученное в рамках САПР фирмы Xilinx.

Вспомогательные файлы хранят дополнительную информацию о проекте. Их имена совпадают с именем проекта.

# 3 СТРАТЕГИЯ ПРОЕКТИРОВАНИЯ

Среда проектирования Quartus II позволяет реализовать стратегии восходящего либо нисходящего проектирования.

Обе стратегии подразумевают использование поведенческих и структурных описаний модулей. При структурном описании модуль представляется в виде совокупности взаимосвязанных компонентов более низкого уровня в иерархии описаний. При поведенческом же описании задается алгоритм работы модуля.

Восходящее проектирование применимо в том случае, когда для создаваемого устройства имеется детальное структурное описание (обычно – принципиальная схема на микросхемах средней степени интеграции), выполненное в элементном базисе, отличном от имеющегося в распоряжении разработчика СБИС.

При этом разработчик решает следующие задачи:

– создание функциональных аналогов элементов, использованных в заданном структурном описании;

– отладка созданных компонентов;

– сборка созданных компонентов в единый модуль;

– моделирование и отладка устройства в целом.

Таким образом, в процессе проектирования разработчик сначала создает модули нижнего уровня в иерархии описаний, а затем – модуль верхнего уровня. Отсюда и название стратегии проектирования.

Стратегия нисходящего проектирования применяется в том случае, когда задан алгоритм работы (поведенческое описание) создаваемого устройства и набор системных требований (максимальная тактовая частота работы, задержка распространения сигналов от входов до выходов, потребление энергии, стоимость и т. д.). При этом поведенческое описание может быть как формализованным (блок схема алгоритма, граф, таблица переходов и выходов и т. д.), так и неформализованным (словесное описание). Реализация нисходящего проектирования базируется на итерационном выполнении структурной декомпозиции.

Упрощенно, ориентируясь на возможности САПР Quartus II, процедура нисходящего проектирования выглядит следующим образом:

– разработка архитектуры СБИС. Поведенческое описание преобразуется в структурное (блок-схему), элементами которого являются архитектурные модули;

– архитектурные модули либо описываются на поведенческом уровне (например, с помощью языка AHDL), либо осуществляется их структурная декомпозиция и создается структурное описание, элементами которого являются функциональные модули. Далее процедура итерационно повторяется до тех пор, пока все функциональные модули не будут описаны;

– после этого осуществляется функциональное моделирование модулей, имеющих поведенческие описания;

– функциональное моделирование модулей, имеющих структурное описание (модули, имеющие поведенческое описание, входят в них как компоненты);

– моделирование и отладка устройства в целом.

Таким образом, в процессе проектирования разработчик опускается с верхнего уровня иерархии описаний, уровня СБИС к нижним уровням. Отсюда и название стратегии проектирования.

Следует отметить, что стратегия нисходящего проектирования имеет безусловные преимущества как по временным затратам на разработку, так и по качеству проработки проекта.

Независимо от выбранной стратегии проектирования для задания структур и алгоритмов работы модулей целесообразно использование текстового описания, созданного на языках описания аппаратуры высокого уровня, например AlteraHDL (AHDL).

# 4 ВВОД ОПИСАНИЯ ПРОЕКТА

Проект в САПР Quartus II представляет собой полный набор файлов проекта, файлов назначения, файлов моделирования, системных установок и информации об иерархической структуре проекта.

## 4.1 Запуск САПР Quartus II

Запуск САПР выполняется либо с использованием иконки, расположенной на рабочем столе компьютера, либо из меню Пуск: **Пуск** => **Все программы** => **Altera** => **Quartus II**. Для запуска из командной строки введите Quartus и нажмите клавишу Enter (ввод).

После запуска на экране будет открыто главное окно проекта. Менеджер САПР Quartus II состоит из нескольких областей:

– заголовок окна (название проекта и его рабочая папка);

– меню менеджера САПР;

– панель инструментов;

– навигатор проекта (отображает иерархию проекта, файлы проекта и команды быстрого запуска);

– окно состояния процедуры компиляции проекта;

– окно сообщений (выводит информацию о выполнении операций, ошибках и предупреждениях);

– главное окно (отображает отчеты, файлы проекта и другие элементы проекта).

## 4.2 Определение имени проекта и его основных параметров

Для создания нового проект САПР Quartus II имеет встроенный мастер создания проекта (**New Project Wizard**). Для создания проекта с помощью мастера **New Project,** необходимо выполнить следующие действия:

– в рабочей области жесткого диска создать каталог, в котором будет храниться проект;

– в меню **File (**операции с файлами) выбрать опцию **New Project Wizard** (мастер создания проекта). Откроется окно **New Project** (новый проект). При первом запуске отображается окно заставки, в котором отражена вся возможная последовательность действий, сопровождающая открытие нового проекта. Для перехода к первому окну создания нового проекта следует нажать клавишу Next. В окне первого этапа создания проекта указывается имя рабочего каталога проекта (тот который был создан на предыдущем этапе), текущее имя проекта и имя файла верхнего уровня иерархии (таблица 4.1 ).

Таблица 4.1 – Назначение полей ввода

| Поле ввода | Назначение |
|---|---|
| Рабочий каталог проекта | Каталог должен содержать все файлы проекта и другие, связанные с данным проектом файлы. Если вводится имя несуществующего каталога, Quartus II создаст его автоматически. Для выбора существующего каталога, необходимо нажать на кнопку «…», расположенную справа поля ввода |
| Имя проекта | Задается имя файла верхнего уровня проекта |
| Имя блока верхнего уровня иерархии проекта | Quartus II автоматически создает установки компиляции и моделирования для блока, указанного в этом окне. После создания проекта можно добавить другие объекты верхнего уровня иерархии проекта и создать для них установки компиляции и моделирования при помощи меню **Processing**. Заданное имя проекта автоматически присваивается блоку верхнего уровня иерархии. Однако допустимо задавать имена, отличные от имени блока верхнего уровня иерархии |

– второе окно этапа создания проекта предназначено для подключения к проекту ранее созданных файлов. Назначения полей окна описаны в таблице 4.2;

Таблица 4.2– Назначение полей ввода

| Поле ввода | Назначение |
|---|---|
| Имя файла | Выберите имена файлов, которые необходимо подключить к проекту. Нажмите кнопку **Add All,** чтобы добавить в рабочий каталог проекта все файлы.<br>Если нет файлов проекта в других каталогах или файлах, имя которых отлично от имени проекта, то добавлять файлы в этом окне не обязательно. |
| Выбор дополнительных библиотек | Если проект включает библиотеки специализированных функций, укажите путь к ним. |

– если отсутствуют дополнительные файлы, которые должны быть подключены к проекту, список файлов в окне **File Name** будет пуст. Путь к дополнительно подключаемым файлам и их имена могут быть найдены при помощи кнопки **Browse** – (…) (обзор);

– для завершения операции подключения дополнительных файлов следует нажать кнопку **Next;**

– следующее, третье, окно предназначено для выбора семейства и типа ПЛИС, которая будет использована для компиляции проекта. На этом этапе нужно выбрать ПЛИС семейства FLEX10K, поскольку от типа ПЛИС напрямую зависит время компиляции устройства;

– для завершения операции выбора ПЛИС следует нажать кнопку **Next**;

– четвертое окно дает возможность подключить к системе Quartus II при создании данного проекта других средств EDA (Electronic Design Automation – САПР электронной аппаратуры). Если разработка проекта не требует дополнительных средств EDA, не нужно отмечать ни каких дополнительных средств. Нажать кнопку **Next;**

– последнее, пятое окно содержит полную информацию о сделанных назначениях. При необходимости внесения исправлений, всегда можно вернуться к предыдущим окнам – клавиша **Back** (назад);

– для завершения создания нового проекта необходимо нажать кнопку – **Finish** (конец);

– после закрытия окна создания нового проекта в основном окне программы **Project Navigator** в строке **Hierarchies** появится имена используемого семейства ПЛИС и блока верхнего уровня разрабатываемого проекта. Одновременно в верхней части основного окна проекта так же появятся путь к проекту и названия проекта и его блока верхнего уровня иерархии.

### 4.3 Создание блок-схемы проекта

Создание блок-схемы проекта начинается с создания файла блок-схемы.

#### 4.3.1 Создание файла блок-схемы проекта

Для создания файла блок-схемы (графический файл ***.bdf - Block Design File**) необходимо выполнить следующую последовательность действий:

– в меню **File** (операции с файлами) выбирают команду **New** (создать новый файл). Появляется вкладка **New,** позволяющая выбрать тип создаваемого файла;

– в открывшемся окне необходимо выбрать вкладку **Device Design Files** (файлы проекта) и в открывшемся ниже окне выделить строку **Block Diagram/Schematic File** (файлы блок-схемы/схематические файлы). Нажав «OK»**,** откроется окно редактора блок-схемы;

– в меню **File** (операции с файлами) выбираем команду **Save As** и ввести имя записываемого файла**.** При необходимости поставьте галочку около сообщения **Add file to current project** (добавить файл в текущий проект) и запишите файл. В рабочем окне программы появится область с заданным названием файла, а в используемом каталоге проекта появится файл с расширением **.bdf.** Программа автоматически в качестве имени файла верхнего уровня проекта предлагает имя проекта.

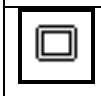
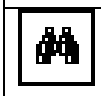
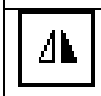
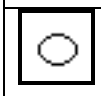
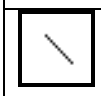
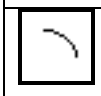
4.3.2 Ввод схемы проекта

Создание блок-схемы проекта выполняется с использованием панели инструментов главного окна программы. Назначение различных инструментов панели приведено в таблице 4.3.

Таблица 4.3 – Список пиктограмм

| **Режим** | **Назначение** |
|---|---|
| **1** | **2** |
| **Редактирование и опрос** | |
|  **Detach Window** **(**Отсоединение окна**)** | Отсоединение окна рабочей области от программного приложения и наоборот |
|  **Selection Tool** (Указатель выделения) | Инструмент выбора объектов для выполнения следующих операций. Отдельный объект выбирается щелчком мыши. Для добавления объекта в группу предварительно нажимают клавишу Shift |
|  **Text Tool** (Текст) | Инструмент для ввода текст. Нанесение на схему текстовых надписей: имен цепей, описаний моделей компонентов, комментариев |
|  **Symbol Tool** (Символ) | Инструмент для ввода символа функционального элемента (либо двойной щелчок в поле схемы) |
|  **Block Tool** (Блок) | Инструмент для ввода блока |
|  **Orthogonal Node Tool** **(**Проводник**)** | Инструмент для рисования ортогональных проводников (цепей) |
|  **Orthogonal Bus Tool** (Шина) | Инструмент для рисования ортогональных шин |
|  **Orthogonal Conduit Tool** (Канал) | Инструмент для рисования ортогональных каналов связи (передачи данных) |

Продолженибе таблицы 4.3

| 1 | 2 |
|---|---|
|  **Use Rubberbanding** («резиновая» связь) | Использование «резиновой» связи. Привязка объектов схемы к связям (ортогональным каналам, шинам и каналам) |
|  **Use Partial Line Selection** (частичный выбор линий) | Использование частичного выбора линий. Выбор необходимого участка связи не зависимо от ее размера (также нескольких связей одновременно) |
|  **Zoom Tool** (масштабирование) | Масштабирование объектов |
|  **Full Screen** (Полный экран) | Отображение проекта во весь экран |
|  **Find** (Поиск) | Поиск объектов |
|  **Flip Horizontal** (Отображение по горизонтали) | Горизонтально отображение объектов |
|  **Flip Vertical** (Отображение по вертикали) | Вертикальное отображение объектов |
|  **Rotate Left 90** (Поворот влево на 90°) | Поворот объектов влево на 90 градусов |
|  **Oval Tool** (Эллипс) | Инструмент для рисования эллипсов |
|  **Line Tool** (Линия) | Инструмент для рисования линий |
|  **Arc Tool** (Дуга) | Инструмент для рисования дуг |

### 4.3.3 Настройка и изменение свойств экрана

Если необходимо, в режиме редактора блок-схем можно изменить основные свойства главного окна программы. Для изменения настроек экрана в меню **Tools** (инструментальные средства) основного окна программы, необходимо выбрать команду **Options** или **Options for Block Editor…**

(Свойства редактора блок-схем графических обозначений**).** В открывшемся окне из списка **Category** необходимо выбрать строку **Block/Symbol Editor.** Данное окно позволяет редактировать свойства главного окна. Описание свойств приведено в таблице 4.4.

Таблица 4.4 – Свойства экрана

| Свойство | Назначение |
|---|---|
| **Show guidelines** | Показать сетку |
| **Guidelines spacing** | Шаг сетки |
| **Snap to grid (applies only to Symbol Editor**) | Привязка к сетке (только в режиме редактора символов) |
| **Use rubberbanding** | Использование «резиновых» связей |
| **Use partial line selection** | Использование частичного выбора линий связи |
| **Double click to show property sheet** | Разрешить вызов окна свойств двойным щелчком мыши |
| **Show parameter assignment** | Показывать значения параметров |
| **Show block I/0 tables** | Показывать для блоков таблицы входов-выходов |
| **Show mapper table** | Показывать отображение таблиц |
| **Show location** | Показывать разводку выводов и размещение |
| **Show I/O standard and reserve pin assignments** | Показывать стандартный ввод/выводы и резервные выводы разводки |
| **Show connection rectangle** | Показывать прямоугольники в местах соединения |
| **Show page breaks** | Показывать границы листа схемы |
| **Include a border when printing** | Включать границы при печати |
| **Automatically open as detached window** | Автоматически открывать как отдельное (отсоединенное от программного приложения)окно |

Изменить, используемые в редакторе цветовую гамму и шрифт, можно с помощью команд **Colors** и **Fonts** того же меню.

### 4.3.4 Создание функционального блока

Создание функциональных блоков осуществляется следующим образом:

– на панели инструментов необходимо выбрать клавишу **Block Tool** (**рисование блока**);

– нажать на белом поле редактора левую кнопку мыши, и удерживая ее, выделить прямоугольную область, размеры которой соответствуют вводимому блоку. Корректировать свои действия, можно используя команды **Undo** (отмена действия) и **Redo** (повтор действия) в меню **Edi**t (правка);

– выбрать на панели инструментов клавишу «указатель выделения» - в виде стрелки, навести указатель мыши на созданный блок и дважды щелкнуть левой кнопкой мыши. Открывается диалоговое окно **Block Properties** (свойства блока). Оно предназначено для ввода имени проекта и обозначения его входных и выходных сигналов. Также вызвать окно **Block Properties** можно и не выходя из режима «ввод блока». Для этого, достаточно наведя курсор на нужный блок нажать правую клавишу мыши. В открывшемся меню надо выбрать строку **Block Properties;**

– выбрать вкладку **General** (основные) и в открывшемся окне в первой строке **Name** (имя) ввести имя блока. Имя во второй строке вкладки оставляют без изменения ( Instance name - inst);

– выбрать вкладку **I/Os**. В окно **Name (**имя**)** ввести имя первого входного вывода блока**.** В окне **Type (**тип) указать тип вывода. Нажать клавишу **Add** (добавить);

– в нижерасположенном окне (**Existing block I/0s**) появится соответствующая запись;

– аналогичным образом ввести и имена остальных выходов и выходов данного блока. Также можно ввести имена всех выводов, через запятую. При этом необходимо следить за правильным назначением функции выводов: вход (**Input**), выход (**Output**) или двунаправленный вывод (**Bidir**)**.** Данные свойства выбираются из вкладке **Type** (тип);

– после окончания ввода имен всех выводов данного блока необходимо нажать «ОК». После этого введенные названия входов и выходов появятся в таблице редактируемого блока.

Аналогичным образом вводятся имена входов и выходов всех блоков проекта**.**

После ввода имен выводов функциональных блоков может оказаться, что из-за малых размеров блоков они полностью не отображаются в таблицах. Увеличить размеры блоков можно двумя способами. Первый – выбрать на панели инструментов «указатель выделения» (стрелка), щелкнуть по нужном блоке левой кнопкой мыши и растянуть его до нужного размера за точки на обрамляющем его контур. Второй – выделить вышеуказанным способом нужный блок и выбрать в меню **Edit (**правка) команду **AutoFit**. Блок автоматически увеличится до нужных размеров.

При необходимости, разработанные блоки можно расставить на поле главного окна программы наиболее удобным образом. Для этого на панели инструментов необходимо выбрать инструмент «указатель выделения». Данный инструмент также можно выбрать, нажав клавишу **ESC.** Навести указатель мыши на передвигаемый блок, нажать левую кнопку мыши и, удерживая ее, переместить блок на новое место.

После выполнения всех описанных действий необходимо записать проект (команда **Save** в меню **File).**

### 4.3.5 Создание входных и выходных выводов

Для создания входных и выходных выводов на разрабатываемой блок-схеме необходимо на панели инструментов выбрать инструмент **Symbol Tool** (Ввод символа функционального элемента). Появляется окно с названием **Symbol** (символ).

В левой части окна необходимо последовательно указать путь к библиотеке с нужными примитивами: **altera/81/quartus/libraries/primitives/pin/input**. В правой части окна появится изображение выбранного примитива. В данном случае это входной вывод **input**. После нажатия клавиши «ОК» выбранный символ появится в основном поле программы. При таком вводе автоматически включается режим **Repeat-insert-mode** (повторного ввода) при котором один символ можно вставить в несколько мест проекта. Введенный символ привязывается к курсору. Теперь при нажатии левой кнопки мыши символ вводится на указанное в данный момент место схемы. Далее его можно перевести в другое место схемы и там его аналогичным способом зафиксировать. Для завершения вставки достаточно нажать на клавиатуре клавишу ESC или на правую кнопку мыши. Появляется окно, в котором можно выбрать одну из двух команд: **Insert Here** (вставить здесь) и (отмена). При выборе команды **Insert Here** процесс вставки продолжается, при выборе **Cancel** процесс введения элемента завершается.

Аналогичным образом вводятся все выводы, необходимые для создания проекта. После окончания ввода всех выводов необходимо перезаписать файл проекта.

### 4.3.6 Назначение имен выводов

Присвоение имен входным и выходным выводам производится в следующей последовательности:

– навести указатель на требуемый вывод и дважды щелкнуть левой кнопкой мыши;

– на экране появится окно **Pin Properties** (свойства выводов). В данном окне выбрать вкладку **General** (основные свойства);

– в строке **Pin name(s)** (имя вывода) ввести имя данного вывода.

– в строке **Default value** (значение по умолчанию) выставляется исходное значение переменной, либо логическая единица (**VCC**), либо логический ноль (**GND**);

– вкладки **Format** (формат) позволяет изменить цвета линий и текста;

– завершается назначение нажатием клавиши «ОК».

По аналогии вводятся имена всех входных и выходных выводов проекта.

4.3.7 Выполнение соединений блоков устройства

Связи между функциональными узлами устройства могут осуществляться:

– сигнальными линиями (одиночными проводниками) – команда меню инструментов **Orthogonal Node Tool (**рисование ортогональных проводников);

– шинами – команда меню инструментов **Orthogonal Bus Tool** (рисование ортогональных шин);

– каналами передачи данных – команда меню инструментов **Orthogonal Conduit Tool** (рисование ортогональных каналов связи).

Кроме этого большинство соединений между узлами (функциональными блоками) можно выполнить с помощью **Selection and Smart Drawing Tool** (автоматического указателя выделения). Данный указатель (стрелка) автоматически переключается в режим рисования необходимых линий связи при наведении на обозначение любого вывода или границу блока. Например, после наведения стрелки на обозначение вывода, автоматически включается указатель рисования ортогональных линий, а при наведении на границу блока – режим рисования ортогональных шин. Эти же режимы можно выбрать и на панели инструментов.

Система Quartus II автоматически создает связи между соединенными блоками, а так же между сигналами шины и входами-выходами блоков. Таким образом, шина выступает в роли канала для любого числа сигналов.

Рисование линии шины или канала выполняется в следующей образом. На панели инструментов выбирается необходимый инструмент для рисования связи, например, **Orthogonal Bus Tool**. Для определения начала шины щелкнуть левой кнопкой курсора на входной вывод, который необходимо соединить**.** Далее, удерживая левую кнопку мыши, переместить указатель до границы необходимого блока, а затем до следующего блока, если это необходимо**.** При соединении шины с любым из блоков на краю последнего автоматически появится символ **mapper** (распределителя). Распределитель позволяет назначать входные и выходные порты блока определенным сигналам шины. Аналогично соединяют выходной вывод**.**

Проверить выполненные соединения можно, выполнив следующую последовательность действий:

– в режиме **Selection Tool** навести указатель на нужный участок канала связи;

– нажать правую клавишу мыши и в появившемся списке выбрать команду **Properties** (свойства). Откроется окно **Conduit Properties** (свойства канала);

– в открывшемся окне выбрать команду **Signal** (сигналы). Появляется окно, в котором перечислены все сигналы, проходящие через данный канал связи.

4.3.8 Разводка сигналов между блоками

САПР Quartus II предоставляет различные пути разводки сигналов (**map signals**) между отдельными блоками (**blocks**) и функциональными элементами (**simbols**) схемы. Описание методов разводки представлено в таблице 4.5.

Таблица 4.5 – Методы разводки сигналов между блоками

| **Метод разводки** | **Описание** |
|---|---|
| Автоматическое соединение **(smart)** | Если имена сигналов входов/выходов (**I/Os**) одного блока совпадают с именами соответствующих сигналов другого блока, то соединение формируются автоматически. Не надо рисовать линии этих шин или каналов. Необходимо лишь указать, какие сигналы блоков должны оставаться не присоединенными. Это необходимо для предотвращения автоматического выполнения их соединения |
| Соединение по именам проводов или шин **(connection by name)** | Если имена соединяемых вход/выходов (**I/Os**) блоков различны, их можно соединить с каналом или шиной по имени. Для этого необходимо назначить каналу имя, соответствующее имени присоединяемого входа/выхода. Этот способ удобен для соединения каналов без обрисовки их линий на схеме. Для создания соединения между такими каналами и блоками сначала необходимо проверить, что бы каждый такой канал соединялся с блоком одним концом и не имел физического соединения с другим блоком или функциональным элементом.<br>Соединение двух каналов по имени можно осуществить, присвоив им одинаковые имена (полагая, что эти имена соответствуют именам соединенных входов/выходов блоков). Таким же способом можно соединять по имени и отдельные проводники |
| Задание соединений в явном виде с помощью распределителей сигналов (**mappers**) | Если имена входов/выходов соединяемых блоков различны, но блоки физически соединены – распределение сигналов можно задать в явном виде. Для этого необходимо присвоить имена выводов блоков сигналам шины, а затем распределить сигналы по соединяемым выводам блоков |

При назначении имен ряду сигналов для обеспечения возможности их соединения по именам или логического соединения необходимо:

– в режиме указателя выделения (стрелка) выделить нужный проводник;

– нажать правую кнопку мыши и в открывшемся меню выбрать строку **Properties** (свойства);

– в появившемся меню **Node Properties** (свойства точки соединения) перейти на вкладку **General** (основная);

– в поле **Name** (имя канала) ввести имя необходимого сигнала;

– нажать кнопку «ОК» после чего сигнал автоматически добавится к каналу и его имя отобразится над ранее выделенным проводником.

Для обеспечения передачи сигналов между блоками в случае не совпадения их имен необходимо:

– в режиме указателя выделения навести курсор на указатель выходного вывода (**mapper**) так, чтобы рядом с курсором образовалось изображение таблицы и дважды щелкнуть левой кнопкой мыши;

– в появившемся окне с названием **Mapper Properties** (свойства карты соединений) выбрать закладку **General** (основные);

– в списке **Type** (тип) выбрать функциональное назначение данного вывода;

– перейти на вкладку **Mapping** (составление карты соединений) и в списке **I/0 on block** (входы и выходы блока) выбрать необходимый сигнал;

– в списке **Signals in conduit** (сигналы в канале) выбрать имя сигнала, который должен присутствовать в канале связи;

– для завершения соединения необходимо нажать кнопку **Add** (добавить). При этом введенная информация отобразится в поле **Existing mappings** (существующие соединения);

– для выхода их окна необходимо нажать кнопку «ОК». При этом на блок-схеме разрабатываемого устройства появится соответствующая таблица, отображающая введенную информацию о назначенных соединениях;

– сохранить файл проекта, используя команду **Save** (сохранить).

# 5 СОЗДАНИЕ ОПИСАНИЙ ОТДЕЛЬНЫХ БЛОКОВ ПРОЕКТА

Для того, чтобы создать файлы описания отдельных блоков проекта предварительно необходимо выполнить следующую последовательность действий:

– запустить САПР Quartus II;

– вызвать меню **File** (Файл) и в нем выбрать команду **Open Project** (открыть проект);

– в открывшемся меню в строке «тип файлов» выбрать расширение файла, соответствующее проекту (**Quartus II Project File (*.qpf; *.quartus; *.qar)**), в строке «папка» указать путь к проекту, а в свободном поле выбрать имя проекта и нажать кнопку «открыть»;

– во втором открывшемся окне выбрать имя нужного файла верхнего уровня проекта и нажать «открыть»;

– в главном окне программы появится выбранный файл верхнего уровня проекта, содержащий блок-схему устройства;

– выделите левой кнопкой мыши на блок-схеме проекта блок, описание которого необходимо ввести и нажмите правую кнопку мыши;

– в появившемся окне выделите курсором строку **Create Design File from Selected Block** (создать файл проекта для выделенного блока) и нажать левую кнопку мыши;

– появившееся окно предоставляет возможность ввести либо поведенческое описание алгоритма работы блока на языках высокого уровня (**AHDL**, **VHDL** или **Verilog HDL**), либо вариант его схемотехнического представления (**Schematic**). В зависимости от выбранного типа описания Quartus II создаст файл описания блока с соответствующим расширением. Имя создаваемого файла автоматически задается как имя первоначально выделенного блока. При выборе описания **AHDL** будет создан файл ***.tdf**, при **VHDL** – файл ***.vhd**, при **Verilog HDL** – ***.v**, а при выборе **Schematic – *.bdf**. Необходимо обратить внимание, что в меню должен быть установлен флаг около надписи **Add the new design file to the current project** (добавить новый файл к текущему проекту). После выбора типа описания файла, необходимо нажать кнопку «ОК»;

– система Quartus II соответствующей надписью, появляющейся на экране, сообщает об успешном открытии нужного файла и открывает его в своем главном окне. Теперь можно вводить описание блока.

## 5.1 Общие принципы схемотехнического описания поведения блока

Процесс создания схемотехнического описания сводится к составлению в графическом редакторе из имеющихся в библиотеке системы Quartus II стандартных примитивов логической схемы блока. Для этого необходимо выполнить стандартную процедуру, состоящую из следующих действий:

– сформулировать алгоритм работы блока;

– по выполненному описанию формализовать алгоритм работы блока и записать соответствующие функцию алгебры логики (ФАЛ);

– по ФАЛ, с использованием имеющихся примитивов, нарисовать логическую схему устройства и ввести ее в окно программы.

Схемный редактор интегрирован в САПР Quartus II.

При создании схемы могут использоваться:

– простейшие логические элементы, триггеры, выводы и другие примитивы;

– мегафункции, модули созданные фирмой Altera;

– ранее созданные в текстовом или графическом редакторах компоненты.

## 5.2 Общие принципы описания с использованием языков высокого уровня

Языки описания аппаратуры **(Hardware Description Languages)** являются формальной записью, которая может быть использована на всех этапах разработки цифровых электронных систем. Это возможно потому, что язык легко воспринимается как машиной, так и человеком. Он может использоваться на этапах проектирования, верификаций, синтеза и тестирования аппаратуры, также как и для передачи данных о проекте, его модификации и сопровождения. Языки описания аппаратуры долгое время были прерогативой довольно узкого класса разработчиков специализированных интегральных схем. С появлением такой элементной базы, как программируемые логические интегральные схемы (ПЛИС), резко расширился круг пользователей, заинтересованных в использовании современных способов описания проекта.

Языки описания аппаратуры можно условно разделить на языки высокого и низкого уровня. К первым принято относить VHDL и Verilog HDL, ко вторым - AHDL, Abel HDL и ряд других. Языки высокого уровня позволяют обеспечить определенную мобильность описания при миграции на другую элементную базу, в то время как языки низкого уровня ориентированы на использование архитектурных особенностей ПЛИС конкретных производителей.

Возможности текстового ввода описания поведения блока:

– нумерация линий;

– использование заготовок языковых конструкций;

– отображение ключевых слов выбранными цветами;

– подсказка о необходимости сохранения файла.

# 6 КОМПИЛЯЦИЯ ПРОЕКТА

Компилятор САПР Quartus II состоит из ряда модулей, выполняющих следующие функции:

– проверка проекта на наличие ошибок;

– логический синтез;

– размещение и разводка проекта в ПЛИС;

– генерация выходных файлов для моделирования проекта;

– анализ временных характеристик;

– программирование.

В начале компиляции проекта из него извлекается информация об иерархических связях между составляющими его файлами и описание проекта проверяется на наличие основных ошибок. Затем создается организационная карта проекта и все файлы преобразуются в единую базу данных, с которой в последствие и будет работать система.

Компиляция может выполняться с учетом заданных требований, к которым относятся:

– обеспечение требуемых временных характеристик проекта;

– увеличение быстродействия;

– оптимизация используемых ресурсов ПЛИС.

Компилятор создает файлы для программирования и конфигурирования ПЛИС фирмы Altera.

Промежуточные и окончательные результаты компиляции в системе Quartus II можно посмотреть в окне **Compilation Report** (отчет о компиляции).

Программирование и конфигурирование ПЛИС фирмы Altera может быть выполнено как с помощью встроенных средств САПР, так и с использованием стандартных промышленных средств программирования.

## 6.1 Настройка компилятора

Система Quartus II позволяет выполнить компиляцию, как всего проекта, так и любой его составляющей части.

При настройке компилятора определяются:

– компилируемая часть проекта (**Compilation focus**);

– тип компиляции;

– семейство и тип ПЛИС;

– дополнительные параметры компиляции.

При создании нового проекта система Quartus II по умолчанию устанавливает значения всех необходимых параметров. Параметры, заданные по умолчанию, можно переопределить в соответствие с поставленными требованиями. Кроме того, есть возможность выбора различных параметров настройки непосредственно при выполнении компиляции.

Ниже рассматривается методика настройки основных параметров компиляции, включающая:

– просмотр основных параметров компилятора;

– определение семейства и типа ПЛИС;
– определение режима компиляции;
– определение и настройка параметров логического синтеза и разводки;
– определение параметров верификации проекта на этапе компиляции.

Компилятор системы Quartus II имеет модульный характер. В него входят следующие модули (модули, помеченные звездочкой (*) являются опциональными и их наличие зависит от настроек):

– модуль анализа и синтеза проекта (**Analysis & Synthesis**);
– модуль размещения «сборщик» (**Fitter**);
– модуль транслятора программатора (**Assembler**);
– модуль временного анализа (**Timing Analyzer**);
– помощник проектирования(**Design Assistant**)*;
– редактор списка соединений (**EDA Netlist Writer**);
– интерфейс базы данных компилятора (**Compiler Database Interface**) *.

Полную компиляцию проекта можно запустить, выбрав в меню **Processing** (обработка) команду **Start Compilation** (пуск компилятора) или **Ctrl+L**. При этом будут последовательно запущены все модули компилятора. Нужные модули компилятора также можно запускать по отдельности. Для этого необходимо выбрать пункт **Start** (пуск) меню **Processing** (обработка) и затем, выбрать команду для модуля, который необходимо запустить, через подменю **Start**.

Модули компилятора так же можно запустить, используя меню **Processing** (обработка) и выбрав в нем команду **Compiler Tool** (свойства компилятора). На экране появиться окно компилятора.

Нажатие кнопки **Start** (пуск) приводит к полной компиляции проекта. После выполнения всего цикла полной компиляции, Quartus II сообщает о её завершении и о количестве найденных ошибок и сделанных предупреждений по поводу проекта.

Этапы типичной компиляции в системе Quartus II представлены на рисунке 6.1.

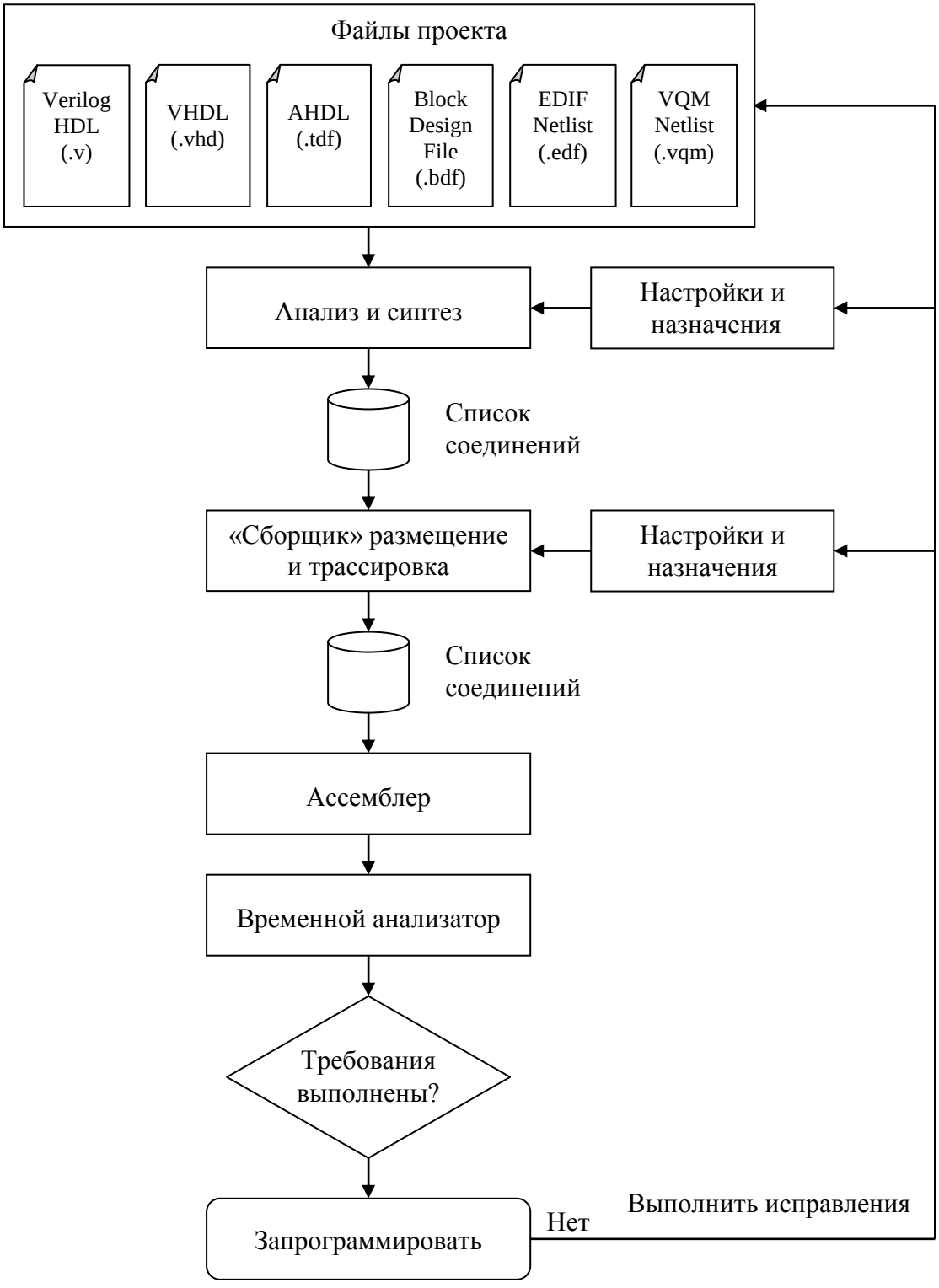
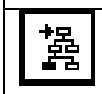
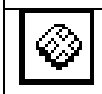



Рисунок 6.1 – Этапы типовой компиляции в Quartus II

Запустить отдельный модуль компилятора можно используя кнопки, размещенные под названиями соответствующих модулей (таблица 6.1).

Таблица 6.1 – Компиляция модулей проекта

| **Модуль компиляции** | **Назначение** | |
|---|---|---|
| **1** | 2 | |
| **Analysis & Synthesis** | Модуль анализа и синтеза проекта проверяет файлы дизайна на ошибки и затем строит базу данных, которая интегрирует все файлы дизайна в иерархию. Также модуль синтезирует и оптимизирует дизайн. В конце этот модуль производит технологическое соответствие Вашей разработки устройству, в котором она должна быть запрограммирована | |
| |  **Start Analysis & Synthesis** | Запуск модуля анализа и синтеза проекта |
| |  **Analysis & Synthesis Settings** | Открыть окно настоек модуля анализа и синтеза |
| |  **Synthesis Report** | Открыть файл отчетности модуля анализа и синтеза |
| |  **Hierarchy Project Top** | Открыть верхний файл иерархии проекта |
| **Fitter** | «Сборщик» помещает логику дизайна в микросхему. Перед тем как использовать этот модуль, необходимо выполнить анализ и синтез | |
| |  **Start Fitter** | Запуск модуля «сборщик» |
| |  **Fitter Settings** | Открыть окно настроек модуля «сборщик» |
| |  **Fitter Report** | Открыть файл отчетности модуля «сборщик» |
| |  **Chip Planner (Floorplan & Chip Editor)** | Открыть топологическую структуру проекта, его размещение в ПЛИС |
| **Assembler** | Ассемблер завершает обработку проекта, превращая то, что сгенерировал сборщик в образ для программирования устройства в форме одного или нескольких файлов | |
| |  **Start Assembler** | Запуск модуля ассемблера |
| |  **Device & Pin Options** | Открыть окно настройки устройства и выводов |

Продолжение таблицы 6.1

| 1 | 2 | |
|---|---|---|
| |  **Assembler Report** | Открыть файл отчетности модуля ассемблера |
| |  **Programmer** | Открыть окно программатора |
| **Classic Timing Analyzer** | Временной анализатор анализирует, отлаживает и утверждает временную производительность всей логики в дизайне. Прежде чем выполнять этот модуль, необходимо выполнить анализ и синтез, а также запустить сборщик. Однако имеется возможность произвести временной анализ на раннем этапе без сборки | |
| |  **Start Classic Timing Analyzer** | Запуск модуля временного анализатора |
| |  **Classic Timing Analyzer Settings** | Открыть окно настроек модуля временного анализатора |
| |  **Timing Analyzer Report** | Открыть файл отчетности модуля временного анализатора |
| |  **Timing Analyzer Summary** | Открывает результирующее окно временного анализатора |

Список в левой части окна отчета компилятора (Compilation Report) позволяет получить более детальный отчет о выполненной компиляции.

6.1.1 Просмотр и настройка основных свойств компилятора

Просмотреть и настроить основные параметры компилятора можно либо с использованием общего окна **Settings** (назначений параметров проекта), которое можно вызвать из окна **Assignments** (назначения) командой **Settings** (установки), либо непосредственно по модулям компилятора, с использованием окна **Compiler Tool** (свойства компилятора). Оба метода позволяют открыть окна свойств соответствующих модулей компилятора и, при необходимости, ввести соответствующие изменения в параметры компиляции. В левой части отрывающегося окна показано общее дерево возможных параметров проекта, а в его правой части – параметры выбранного раздела назначений.

6.1.2 Определение семейства и типа ПЛИС

Выбрать семейство и конкретный тип ПЛИС, предназначенный для реализации проекта, можно используя общее окно установок проекта. Оно

вызывается из окна **Assignments** (назначения) либо командой **Settings** (установки), тогда в открывшемся в левой части окна дереве назначений проекта необходимо выбрать строку **Device** (прибор), либо командой **Device**. Результатом этих действий будет открытие окна с названием **Device.**

В верхней части данного окна необходимо выбрать семейство используемой ПЛИС (окно с надпись **Family** (семейство)).

Выбор конкретного типа ПЛИС приведен к автоматическому переключению в разделе меню **Target Device** (программируемый прибор). Будет выбран указатель, свидетельствующий о выборе для реализации проекта интегральной схемы конкретного типа (**Specific device selected in "Available devices" list**). Если конкретный тип ПЛИС не задается, то в разделе **Target Device** (программируемый прибор**)** следует выбрать строку **Auto device selected by the Fitter from the "Available devices" list** (автоматический выбор модулем размещения компилятора ПЛИС из доступного для выбранного семейства списка приборов).

Правее окна со списком доступных приборов, под надписью **Show in "Available devices" list** (ввести в список доступных приборов**),** можно выбрать тип корпуса (**Package**), количество выводов (**Pin count**) и градацию быстродействия (**Speed Grade**) используемой ПЛИС. Если эти параметры специально не оговариваются, то во всех окнах можно задать строку **Any** (произвольно).

Следует отметить, что выбор конкретных параметров ПЛИС, производимый в разделе **"Available devices" list** приводит к изменению списка доступных приборов в окне **Target Device.** Поэтому можно либо выбрать тип прибора из списка **Target Device,** задав во всех строках раздела **Show in "Available devices" list** выбор произвольных параметров (**Any),** либо произвести в разделе **Show in "Available devices" list** выбор конкретных параметров ПЛИС, что приведет к сужению множества доступных приборов. Таким образом, если в разделе **Show in "Available devices" list** задать указанные для ПЛИС параметры, то в списке доступных приборов останется только этот прибор. Выбор параметров в окне **Show in "Available devices" list** удобен в случае если уже известны конструктивные параметры разрабатываемого устройства.

В разделе **Show in "Available devices" list** дополнительно указано напряжения питания ядра выбранной ПЛИС (**Core voltage**).

## 6.2 Выполнение компиляции проекта

Компилятор во время работы использует сделанные ранее установки. Компилятор автоматически распознает и обрабатывает все файлы, относящиеся к объекту компиляции. Это файлы с расширениями:

– ***.inc** – файлы включения, содержащие описание функций на языке AHDL;

– ***.mif** – файлы инициализации памяти;

– ***.hex** – файлы инициализации памяти в шестнадцатеричном формате Intel;

– ***.psf, *.esf, *.and, *.csf** – файлы, содержащие параметры проекта и компиляции.

Генерируемые в процессе компиляции предупреждения (**Warnings**) и сообщения об ошибках (**Error messages**) отображаются в окне процессора сообщений (**Messages**).

### 6.2.1 Запуск компилятора

Для запуска компилятора необходимо в меню **Processing** (обработка) выбрать команду **Start Compilation** (запустить компиляцию). Начинается процесс компиляции. При этом появляются окна: **Module** (модуль), в котором отражается процесс прохождения компиляции и фиксируется затраченное на это время и **Compilation Report** (отчет о компиляции).

Процесс компиляции выполняется в фоновом режиме. Поэтому при длительной компиляции возможна работа с другими окнами САПР Quartus II или другими запущенными под операционной системой программами.

После завершения компиляции на экране появляется соответствующая надпись, свидетельствующая о завершении процесса и количестве найденных ошибок и информационных сообщений.

Для завершения процесса компиляции необходимо нажать кнопку «ОК».

### 6.2.2 Локализация источника сообщений

В течение компиляции в окне процессора сообщений (**Messages**) появляются информационные сообщения, сообщения об ошибках и предупреждения, сделанных компилятором по мере анализа проекта. Это записи относятся к определенному месту в файле проекта или в другом исходном файле.

Для того, что бы найти (локализовать) часть проекта, являющуюся источником сообщения необходимо раскрыть соответствующее сообщение, щелкнув по значку «+». Далее необходимо два раза щелкнуть по интересующему Вас сообщению, после чего в главном окне системы появится файл, к которому относится данное сообщение. К тому же на экране будет цветом выделен непосредственный источник сообщения.

### 6.2.3 Просмотр отчета о компиляции

Вся информация о проведенной компиляции проекта выводится в окне **Compilation Report** (отчет о компиляции). Непосредственно, после завершения компиляции в правой части окна появляется сводный отчет (**Flow Summary**). Отчет содержит в себе информацию о дате и времени проведения компиляции и версии системы Quartus II**,** именах контрольного и файла верхнего уровня проекта, семействе и типе используемой ПЛИС, типе компиляции (промежуточная или окончательная), удовлетворяет ли проект заданным

временным параметрам и количестве использованных логических блоков, выводов и ячеек памяти.

В левой части окна присутствует организованный по иерархическому древовидному принципу каталог сообщений отдельных блоков компилятора. Здесь можно почерпнуть полную информацию о всех этапах компиляции, включая предварительно сделанный установки и результаты работы отдельных блоков компилятора. Дополнительно предоставляется возможность распечатки выбранных частей отчета.

# 7 МОДЕЛИРОВАНИЕ ПРОЕКТА

Моделирование (**Simulation**) позволяет определить реакцию разработанного проекта на заданное входное воздействие, то есть позволяет убедиться в правильности его функционирования.

Исходными данными для моделирования являются внешние воздействия, заданные в виде некоторого входного вектора (набора кодовых слов). Подсистема моделирования (**Simulator**) САПР Quartus II, в соответствие с алгоритмом проекта, синтезирует выходные сигналы, соответствующие его реакции на заданное входное воздействие, которая очень близка к реакции запрограммированной ПЛИС. В типовых задачах разработчик задает наборы входных векторов и анализирует полученные в результате моделирования выходные сигналы.

В зависимости от поставленной цели подсистема моделирования позволяет выполнить:

– функциональное моделирование проекта (**Functional Simulation**) при котором проверяется правильность описания и логического функционирования проекта;

– моделирование с учетом временных параметров реальной ПЛИС (**Timing Simulation**), позволяющее проверить не только правильность логического функционирования проекта, но и его работу с учетом реальных параметров выбранной ПЛИС в самых жестких условиях эксплуатации.

## 7.1 Создание вектора входных воздействий

Файлы вектора входных воздействий в системе Quartus II могут задаваться в виде:

– описания в графической форме (некоторых временных диаграмм) с использованием редактора временных диаграмм (**Waveform Editor**) – файлы ***.vwf (Vector Waveform Files);**

– описание в текстовом виде при помощи векторных файлов (**Vector File**) – файлы ***.vec**.

### 7.1.1 Создание временных диограмм

Создание файла (***.vwf**), содержащего временные диаграммы, выполняется в следующей последовательности:

– в меню **File** (файл) выбирается команда **New** (новый);

– в открывшемся окне **New** (новый) выбрать закладку **Veryfication/Debugging Files,** в которой выделить строку **Vector Waveform File** (файл вектора временных диаграмм) и нажать кнопку «ОК»;

– открывается пустое окно редактора временных диаграмм с именем по умолчанию **Waveform1. vwf**;

– в окне **Edit** (редактировать) выбрать команду **End Time (в**ремя окончания) и в открывшемся окне указать время окончания моделирования

(длительность интервала моделирования и единицу измерения времени). Нажать «ОК»;

– созданный файл необходимо сохранить, используя команду **Save As** (записать как) меню **File** (файл). Программа автоматически предложит сохранить файл с именем, совпадающим с именем файла верхнего уровня проекта, присвоив ему расширение **.vwf**;

– для завершения процесса создания файла необходимо нажать кнопку **Save** (сохранить**).** При этом необходимо обратить внимание на наличие флажка около надписи **Add file to current project** (добавить файл к текущему проекту). Если флажок поставлен, то система автоматически присоединяет созданный файл к текущему проекту;

– для удобства, на поле временных диаграмм нанесена временная сетка, предназначенная для визуальной привязки сигналов к конкретным временным интервалам. Используя команду **Grid Size** (шаг сетки) меню **Edit (р**едактировать) можно изменить ее шаг (период) повторения (**Period**), начальную фазу (**Phase**) и относительную длительность каждого из полупериодов (**Duty cycle**).

### 7.1.2 Добавление входных, выходных и промежуточных сигналов

Введение входных и выходных сигналов, которые есть в проекте, в файл вектора входных воздействий осуществляется следующим образом:

– в окне **Edit** (редактировать) выбрать строку **Insert** => **Insert Node or Bas** (вставить узел или шину);

– в открывшемся окне **Insert Node or Bas** (вставить узел или шину) нажать кнопку **Node Finder** (система поиска узлов);

– открывается окно системы поиска узлов проекта (**Node Finder**), позволяющее ввести в файл временных диаграмм узлы текущего проекта). В подокне **Lock** (искать) должно быть указано имя верхнего файла проекта, или файла, моделирование которого необходимо выполнить. В подокне **Fitter** (система поиск) нужно указать какие выводы нужно искать. Если необходимо вставить в файл временных диаграмм все узлы проекта, в этом подокне необходимо выбрать команду **Pins: all** (выводы все);

– для отображения оговоренных условиями поиска выводов необходимо нажать кнопку **List** (список);

– в левой части окна под заголовком **Nodes Found** (найденные узлы проекта) появляется список найденных узлов проекта;

– для того чтобы найденные узлы были введены в файл временных диаграмм, их необходимо переместить в правое подокно, которое называется **Selected Nodes** (выбранные узлы). Для этой цели служат расположенные между подокнами **Nodes Found** (найденные узлы проекта) и **Selected Nodes** (выбранные узлы) кнопки (таблица 7.1).

Таблица 7.1 – Список пиктограмм

| Кнопка | Назначение |
|---|---|
| > **Copy to Selected Nodes list** | Копировать в список выбранных узлов. Переместить выделенный узел из подокна **Nodes Found** (найденные узлы проекта) в подокно **Selected Nodes** (выбранные узлы) |
| >> **Copy all to Selected Nodes list** | Копировать все в список выбранных узлов. Переместить все узлы из подокна **Nodes Found** (найденные узлы проекта) в подокно **Selected Nodes** (выбранные узлы) |
| < **Remove from Selected Nodes list** | Удалить из списка выбранных узлов. Переместить выделенный узел из подокна **Selected Nodes (в**ыбранные узлы) в подокно **Nodes Found** (найденные узлы проекта) |
| << **Remove all from Selected Nodes list** | Удалить все из списка выбранных узлов. Переместить все узлы из подокна **Selected Nodes** (выбранные узлы) в подокно **Nodes Found** (найденные узлы проекта) |

– после перемещения в правое подокно всех необходимых при моделировании узлов необходимо нажать кнопку «ОК». Появится окно **Insert Node or Bus** (вставить узел или шину), в котором тоже необходимо нажать кнопку «ОК». После этого в файле временных диаграмм проекта появляются оси для всех вышеуказанных сигналов;

– оси файла временных диаграмм, предназначенные для введения входных сигналов остаются пустыми (не заполненными). Оси, предназначенные для отображения выходных сигналов, соответствуют неопределенным значениям;

– на временных диаграммах присутствует вертикальная линия временного маркера, который изображен в виде сплошной цветной линии. Положение этого маркера можно изменять, используя курсор. Значение сигналов, соответствующее текущему положению курсора, отображается в столбце с именем **Master Time Bar**. Если на временных диаграммах необходимо отметить некоторые базовые моменты времени, это можно сделать, используя команду **Insert Time Bar** (ввести временную метку) меню **Edit** (редактировать). Открывается окно **Insert Time Bar** (ввести временную метку). В данном окне необходимо выбрать время, соответствующее базовому моменту времени и единицу его измерения. После нажатия кнопки «ОК» линия, соответствующая введенному времени, появится на временных диаграммах. Теперь при перемещении маркерной линией над ней будет отображаться текущее время моделирования, а над линиями базовых моментов времени, их расстояние (длительность временного интервала) от маркерной линии;

– при необходимости, любую из введенных линий базовых моментов времени, можно преобразовать в маркерную линию. Для этого на расположенный в верхней части линии необходимо навести курсор и нажать правую кнопку мыши. Появляется окно, позволяющее:

– **Delete** – удалить линию времени;

– **Make Master Time Bar** – использовать данную линию как маркерную;

– **Insert Time Bar** – ввести линию времени;

– **Time Bar Organizer** – вызвать окно органайзера временных линий, позволяющее переназначить основные параметры линий времени;

– **Zoom –** выполнить масштабирование временных диаграмм.

Рассмотренная методика позволяет ввести в файл временных диаграмм любое количество используемых при моделировании проекта входных и выходных сигналов.

Используя окно **Insert Node or Bus** (вставить узел или шину) можно произвольно ввести в файл временных диаграмм нужные сигналы. Для этого достаточно в графе **Name** (имя) ввести название требуемого сигнала, а в последующих строках определить его основные параметры.

### 7.1.3 Редактирование временных диаграмм входных сигналов

Следует отметить, что редактированию могут быть подвержены только входные сигналы, присутствующие в текущем проекте. Необходимый для моделирования вектор входных воздействий вводится путем задания для выбранных временных интервалов логических уровней, соответствующих значениям входной переменной в заданном узле проекта. Для этого необходимо выполнить следующую последовательность действий:

– выбрать требуемый входной узел;

– если необходимо задать одно и то же значение логического сигнала на всем интервале моделирования, необходимо щелкнуть левой кнопкой мыши в левом части окна файла временных диаграмм. В этом случае цветом будет выделена вся ось сигнала;

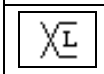
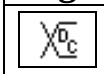
– если необходимо задать значение логического сигнала на определенном интервале моделирования, в правой части окна курсор необходимо установить на ось требуемого сигнала в точке начала задания логического уровня (сигнала) и, удерживая левую кнопку мыши, переместить курсор в конец требуемого временного интервала. В этой точке кнопку мыши следует отпустить. В результате цветом буде выделен только требуемый временной интервал;

– после выделения нужной области сигнала в левой части главного окна системы Quartus II появляется набор инструментов для введения входных воздействий, причем, каждый вариант входного воздействия обозначен соответствующей пиктограммой (таблица 7.2).

Таблица 7.2 – Список пиктограмм

| **Пиктограмма** | **Назначение** |
|---|---|
| **1** | **2** |
|  **Overwrite Uninitialized** (Ctrl+Alt+U) | Очистить сигнал. Задание неинициализированного логического уровня |

Продолжение таблицы 7.2

| 1 | 2 |
|---|---|
|  **Overwrite Forcing Unknown** (Ctrl+Alt+X) | Задание неопределенного логического уровня сигнала |
|  **Overwrite Forcing Low** (Ctrl+Alt+0) | Задание сигнала логического «0» (низкий логический уровень) |
|  **Overwrite Forcing High** (Ctrl+Alt+1) | Задание сигнала логической «1» (высокий логический уровень) |
|  **Overwrite High Impedance** (Ctrl+Alt+Z) | Задание сигнала с высоким импедансом (полное сопротивление) |
|  **Overwrite Weak Unknown** (Ctrl+Alt+W) | Задание слабого неопределенного логического уровня сигнала |
|  **Overwrite Weak Low** (Ctrl+Alt+L) | Задание слабого низкого логического уровня сигнала |
|  **Overwrite Weak High** (Ctrl+Alt+H) | Задание слабого высокого логического уровня сигнала |
|  **Overwrite Don’t Care** (Ctrl+Alt+D) | Задание не имеющего значения логического уровня сигнала |
|  **Overwrite Invert** (Ctrl+Alt+I) | Инвертирование сигнала. Для того чтобы инвертировать сигнал или группы сигналов необходимо, в меню, которое появляется после нажатия правой кнопки мыши, выбрать пункт **Value** (значение) => **Invert** (инвертировать) либо выбрать пиктограмму на панели инструментов **Overwrite Invert**. Инвертирование логического уровня производиться от низкого **low** (0) к высокому **high** (1) и от высокого к низкому уровню логики. Слабый высокий **weak high** (H) инвертируется в слабый низкий **weak low** (L), и слабый низкий в слабый высокий. Не заданный **undefined** (X), высокий импеданс **high impedance** (Z), неинициализированные **uninitialized** (U), слабый неизвестный **weak unknown** (W) и не имеющий значения **don't care** (DC) логические уровни остаются без изменений |

Продолжение таблицы 7.2

| 1 | 2 |
|---|---|
| **Overwrite Count Value** (Ctrl+Alt+V) | Величина значения применяется для задания сигнала с шагом приращения на определенном интервале времени. Вместо того чтобы вручную редактировать значения для каждого узла C**ount Value** автоматически создает подсчета значения для сигнала. Эта функция позволяет указать начальное значение, временной интервал увеличения и когда остановить подсчет. Также можно настроить вид кодирования сигнала и шаг инкремента. Используется для задания сигнала синхронизации |
| **Overwrite Clock** (Ctrl+Alt+K) | Использование функции часов позволяет автоматически генерировать временную диаграииу. Используется для задания тактового сигнала |
| **Overwrite Arbitrary Value** (Ctrl+Alt+B) | Задание произвольного значения логического уровня сигнала |
| **Overwrite Random Value** (Ctrl+Alt+R) | Задание случайного значения логического уровня сигнала. Случайная величина позволяет генерировать случайные значения одного выбранного сигнала, нескольких или группы сигналов. Возможно, генерировать случайные значения на каждом интервале сетки, каждой половине интервала сетки, на случайном интервале или через определенные промежутки времени |

Рассмотрим пример задания сигнала синхронизации.

В появившемся окне **Count Value** на первой вкладке **(Counting)** можно выбрать:

– **Radix** – используемая при отображении сигнала система исчисления;

– **Start Value** – исходное значение сигнала;

– **Increment by** – шаг сигнала;

– **Count Type** – вид кодирования сигнала.

В окне также отображено конечное значение сигнала (значение после изменения) **End Value**;

Вторая вкладка окна выбора параметров сигнала синхронизации **(Timing)** позволяет задать его временные параметры:

– **Start Time** и **End Time** – время начала и окончания действия сигнала соответственно. Эти времена определяют интервал действия задаваемого сигнала;

– **Count every** – время изменения значения сигнала;

– множитель времени изменения сигнала. Использование данного множителя позволяет увеличить время изменения сигнала синхронизации (фактически его период) в целое число раз.

– ввод сигнала синхронизации завершается нажатием кнопки «ОК».

Аналогичным образом вводятся все необходимые для моделирования входные сигналы.

## 7.2 Определение параметров моделирования

Система Quartus II позволяет выполнить моделирование, как всего проекта, так и его любой составной части. Типовые параметры по умолчанию задаются системой моделирования автоматически при создании нового проекта. При необходимости, эти параметры можно отредактировать. Для этой цели служить окно **Simulator Tool** (свойства системы моделирования), вызываемое из раздела **Processing** (обработка) главного меню системы.

Данное окно позволяет определить следующие параметры моделирования проекта:

– выбрать тип моделирования проекта (**Simulation mode**) – функциональное (**Functional**) или учитывающее временные параметры выбранного типа ПЛИС (**Timing**);

– выбрать имя файла, содержащего вектор входного воздействия для моделирования (**Simulation Input**). По умолчанию, это имя файла верхнего уровня проекта. При необходимости, имя требуемого файла можно найти, используя кнопку в правой части окна имени;

– раздел **Simulation Period** (область моделирования) позволяет задать моделирование либо на всем заданном интервале (**Run simulation until all vector stimuli are used**), либо задать время окончания моделирования не совпадающее со временем, определенном в файле временных диаграмм (**End simulation at…**). В последнем случае задаются значение времени окончания моделирования и единица его измерения;

– раздел **Simulation options** (варианты выбора условий моделирования) позволяет определить режимы для моделирования (таблица 7.3).

Таблица 7.3 – Режимы моделирования

| Режим | Описание |
|---|---|
| **Automatically add pins to simulation output waveform** | Функция (режим) автоматического добавления к временным диаграммам выходных выводов проекта |
| **Check outputs** | Режим проверки выходов |
| **Overwrite simulations input file with simulations results** | Режим перезаписи исходного файла моделирования с учетом результатов его выполнения. Если этот режим не задан, то файл входных воздействий сохраняется неизменным, то есть одновременно существует и исходный файл и файл с результатами моделирования |

## 7.3 Выполнение моделирования проекта

Запустить процесс моделирования проекта можно двумя различными способами:

– в меню **Processing** (обработка) вызвать команду **Start Simulation** (запуск моделирования) или **Ctrl+I**;

– в меню **Processing** (обработка) вызвать команду **Simulation Tools** (свойства системы моделирования) и в открывшемся окне нажать кнопку **Start** (пуск).

В обоих случаях начинается процесс моделирования, который, в случае успешного выполнения, заканчивается появлением окна – результата моделирования.

При первом способе запуска процесс моделирования заканчивается отображением в главном окне системы файла с отчетом о моделировании (**Simulation Report**) в левой части которого приведены временные диаграммы, поясняющие работу проекта при заданных входных воздействиях.

При втором методе запуска для получения требуемых временных диаграмм необходимо нажать кнопку **Open** (открыть).

Следует помнить, что последовательность действий, выполняемая при запуске процесса моделирования, зависит от ее исходных установок. Если выполняется функциональное моделирования, то перед запуском системы моделирования необходимо создать список соединений проекта. Для этого в окне **Simulation Tools** (свойства системы моделирования) необходимо нажать кнопку **Generate Functional Simulations Netlist** (создать файл со списком соединений для функционального моделирования). После его создания запускается система моделирования. Если выполняется моделирование с учетом временных параметров выбранной ПЛИС, создание файла со списком соединений не требуется и сразу запускается система моделирования.

Моделирование проекта выполняется в фоновом режиме. Поэтому во время моделирования возможна работа с другими окнами системы или с другими программами, работающими под операционной системой.

Во время моделирования, так же как и на интервале компиляции, работает процессор сообщений, формируя информационные сообщения, предупреждения и сообщения об ошибках.

При анализе полученных результатов, измеряя длительности различных процессов, удобно использовать линии времени, а так другую информацию, содержащуюся в файле отчета о моделировании (**Simulation Report**).

Если в поле временных диаграмм или названий сигналов нажать правую кнопку мыши, появится окно, позволяющее редактировать временные диаграммы. Доступны операции копирования, удаления создания групп сигналов и т.д.

# 8 ПРОЕКТИРОВАНИЕ ЭЛЕКТРОННЫХ УСТРОЙСТВ

## 8.1 Общие методические рекомендации по выполнению проектирования

Изучение дисциплины “СБИС ПЛ” эффективно лишь тогда, когда наряду с овладением теорией студенты в условиях проведения лабораторного практикума приобретают навыки синтеза цифровых устройств различного назначения с помощью САПР, применяемых в промышленности.

Предлагаемые методические указания посвящены изучению методов проектирования электронных устройств с помощью САПР Quartus II, которая отличается от других известных САПР:

– простотой и удобством в обращении, особенно при графическом вводе принципиальной схемы;

– доступностью для пользователя, мало знакомого с вычислительной техникой;

– наличием основных видов анализа электронных устройств;

– развитой библиотекой компонентов.

В ходе выполнения лабораторных работ студенты должны:

– освоить методики синтеза электронных устройств;

– получить знания о методах проектирования электронных устройств на персональных ЭВМ;

– глубже усвоить основы теории электронных устройств и методы их расчета.

Подготовка к лабораторной работе предусматривает обязательное изучение студентами теоретического материала.

Лабораторную работу рекомендуется начинать с проектирования примера схемы цифрового устройства, приведенного в каждой лабораторной работе. Затем следует приступить к исследованию:

– синтезировать цифровое электронное устройство с учетом установленного варианта задания;

– задать параметры и пределы моделирования цифрового устройства;

– исследовать влияние типа ПЛИС на поведение исследуемой характеристики.

– Отчет к каждой лабораторной работе должен содержать:

– название лабораторной работы;

– цель работы;

– краткие теоретические сведения;

– порядок выполнения работы;

– результаты исследования и анализа параметров и характеристик исследуемого устройства;

– особенности функционирования САПР Quartus II, выявленные в ходе выполнения лабораторной работы;

– выводы.

## 8.2 Лабораторная работа №1. САПР Quartus II

**Цель работы:** изучить основные параметры и возможности САПР Quartus II, а также основные типы триггеров и временные диаграммы их работы.

### 8.2.1 Порядок выполнения работы

1 Запустите САПР Quartus II.

2 Изучите основные возможности САПР Quartus II.

3 Исследуйте назначение основных команд, относящихся к созданию чертежей электрических принципиальных схем и анализу их работы.

4 Изучите основные методы построения основных типов триггеров.

5 Синтезируйте схему исследования триггера с учетом установленного варианта задания по таблице 8.1.

Таблица 8.1 – Варианты заданий

| № варианта | Тип триггера |
|---|---|
| 1 | jkff |
| 2 | dff |
| 3 | tff |
| 4 | srff |
| 5 | jkffe |
| 6 | dffe |
| 7 | tffe |
| 8 | srffe |
| 9 | jkff |
| 10 | dff |
| 11 | tff |
| 12 | srff |

6 Создайте схему для исследования триггера в САПР Quartus II.

На рисунке 8.1 в качестве примера приведена схема для исследования RS-триггера.

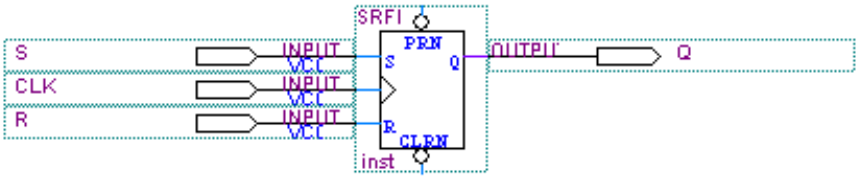



Рисунок 8.1 – Схема исследования RS-триггера

7 Исследуйте временные диаграммы работы триггера при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.2 в качестве примера приведены временные диаграммы работы RS-триггера.

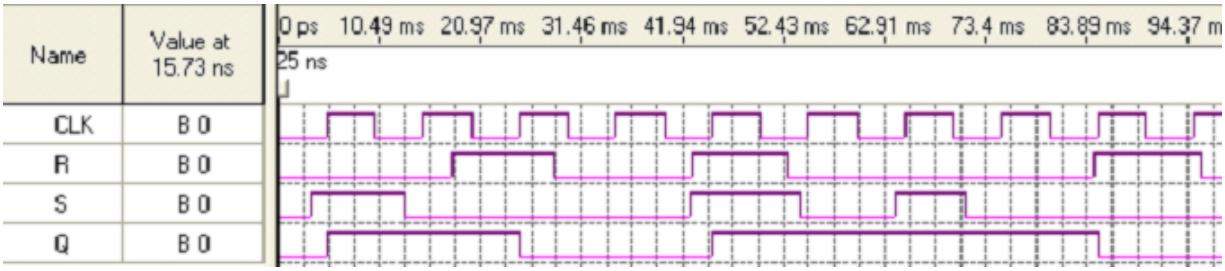



Рисунок 8.2 – Временные диаграммы работы RS-триггера

8 В соответствии с вариантом задания, приведенным в таблице 8.2, синтезируйте схему для исследования триггера с дополнительными входами.

Таблица 8.2 – Варианты заданий

| № варианта | Тип триггера | Дополнительные входы |
|---|---|---|
| 1 | jkffe | load |
| 2 | dffe | load |
| 3 | tffe | load |
| 4 | srffe | load |
| 5 | jkff | enable, load |
| 6 | dff | enable, load |
| 7 | tff | enable, load |
| 8 | srff | enable, load |
| 9 | jkffe | load |
| 10 | dffe | load |
| 11 | tffe | load |
| 12 | srffe | load |

На рисунках 8.3 и 8.4 приведены результаты исследования D-триггера с дополнительными входами enable и load.

9 Приведите комментарии к назначению и описанию работы дополнительных входов разработанного триггера.

10 Определите значения времени установления входных сигналов, времени удержания сигнала, времен задержек распространения, setup time, hold time, а также максимально допустимой частоты тактового сигнала для двух различных типов микросхем ПЛИС (Assignments->Device).

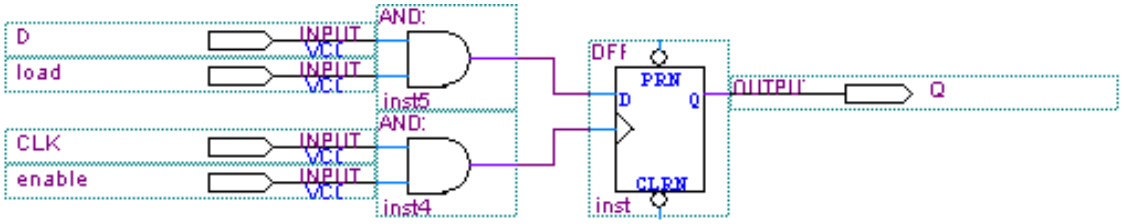



Рисунок 8.3 – Схема исследования D-триггера с дополнительными входами enable и load

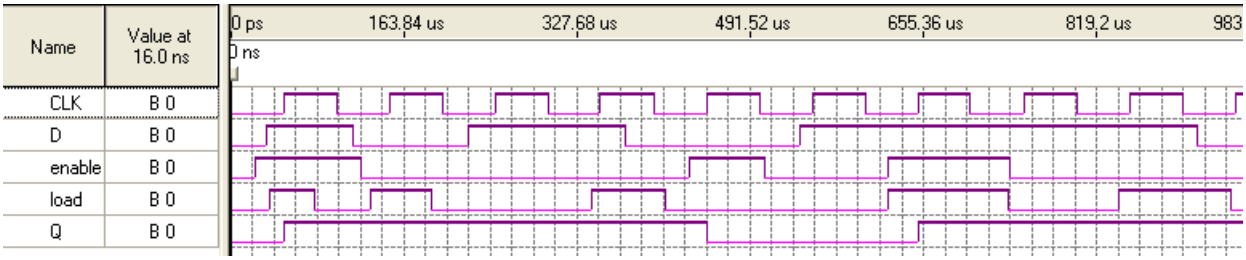



Рисунок 8.4 – Временные диаграммы работы D-триггера с дополнительными входами enable и load

### 8.2.2 Контрольные вопросы

1 Дайте определения триггера.
2 Классификация триггеров.
3 Назначение входов триггера S, R, D, T, J, K, C, V.
4 Основные динамические параметры триггеров.
5 D- и T-триггеры. Временные диаграммы их работы.
6 JK- триггеры. Временные диаграммы их работы.

### 8.3 Лабораторная работа №2. Счетчики

**Цель работы:** изучить временные диаграммы работы счетчиков, построенных по различным схемам.

8.3.1 Порядок выполнения работы

1 Изучите до начала выполнения лабораторной работы методики синтеза счетчиков.

2 Соберите схему проверки стандартного счетчика, тип которого в зависимости от установленного варианта задания приведен в таблице 8.3.

Таблица 8.3 – Варианты заданий

| Номер варианта | Тип счетчика |
|---|---|
| 1 | 74190 |
| 2 | 74191 |
| 3 | 74192 |
| 4 | 74193 |
| 5 | 74196 |
| 6 | 74197 |
| 7 | 74160 |
| 8 | 74161 |
| 9 | 74162 |
| 10 | 74163 |
| 11 | 74190 |
| 12 | 74191 |

3 Создайте схему для исследования заданного счетчика в САПР Quartus II.

На рисунке 8.5 в качестве примера приведена схема для исследования счетчика 74160.

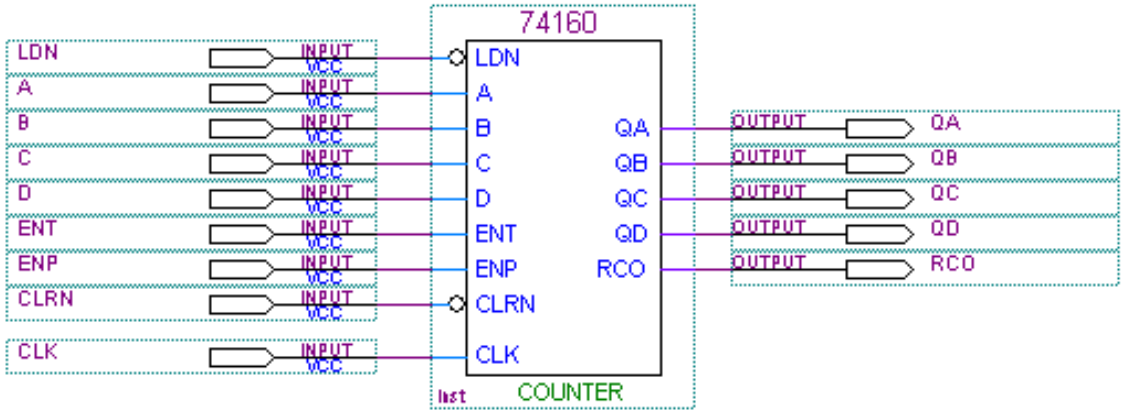



Рисунок 8.5 – Схема исследования счетчика 74160

4 Исследуйте временные диаграммы работы счетчика при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.6 в качестве примера приведены временные диаграммы работы построенного счетчика.

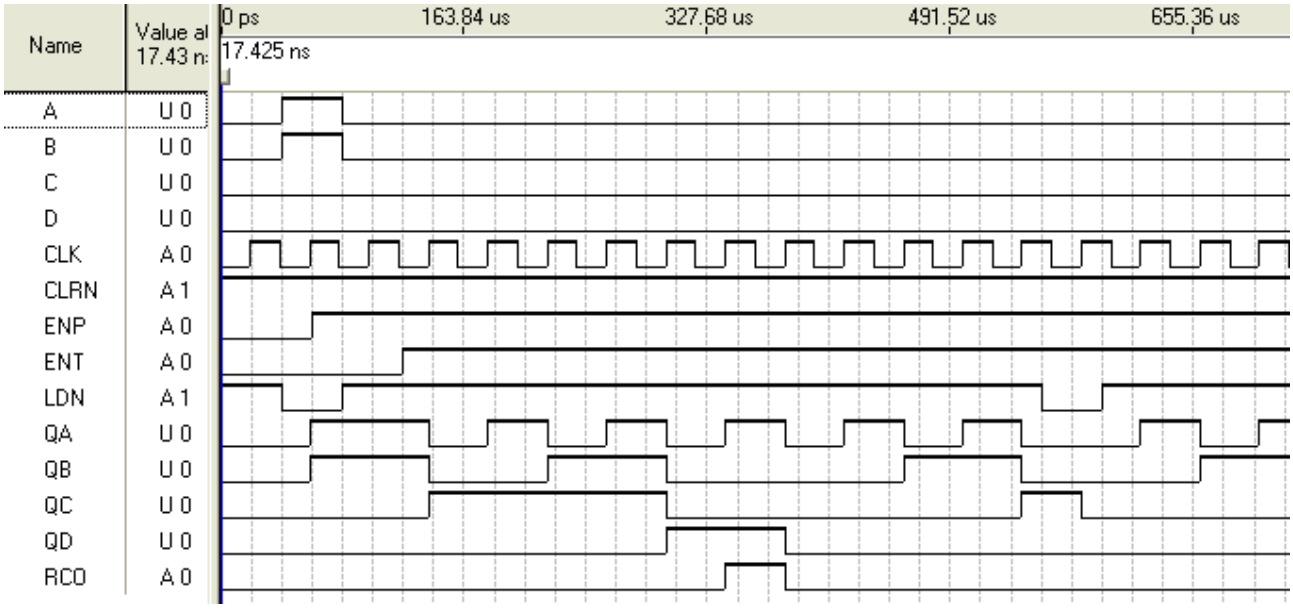



Рисунок 8.6 – Временные диаграммы работы счетчика, построенного на базовой микросхеме 74160

5 Исследуйте и приведите в отчете назначение всех входов и выходов базовой микросхемы счетчика.

6 Синтезируйте схему четырехразрядного счетчика с учетом требований, приведенных в таблице 8.4.

Таблица 8.4 – Варианты заданий

| **№ варианта** | **Тип счетчика** | **Триггер** |
|---|---|---|
| 1 | Асинхронный вычитающий с последовательным переносом | jkff |
| 2 | Асинхронный суммирующий с последовательным переносом | dff |
| 3 | Асинхронный реверсивный с последовательным переносом | tff |
| 4 | Синхронный суммирующий со сквозным переносом | jkff |
| 5 | Синхронный вычитающий со сквозным переносом | dff |

Продолжение таблицы 8.4

| № варианта | Тип счетчика | Триггер |
|---|---|---|
| 6 | Синхронный реверсивный со сквозным переносом | tff |
| 7 | Синхронный суммирующий со сквозным переносом | jkff |
| 8 | Асинхронный вычитающий с последовательным переносом | jkff |
| 9 | Асинхронный вычитающий со сквозным переносом | dff |
| 10 | Синхронный суммирующий с параллельным переносом | tff |
| 11 | Синхронный реверсивный с параллельным переносом | jkff |
| 12 | Асинхронный суммирующий с последовательным переносом | dff |
| Примечание. Указанный триггер обязательно должен входить в состав разрабатываемого счетчика. Допускается дополнительно использовать любые типы логических микросхем | | |

7 Создайте схему проверки синтезированного счетчика.

На рисунке 8.7 в качестве примера приведена схема проверки асинхронного вычитающего трехразрядного счетчика со сквозным переносом на триггере dff.

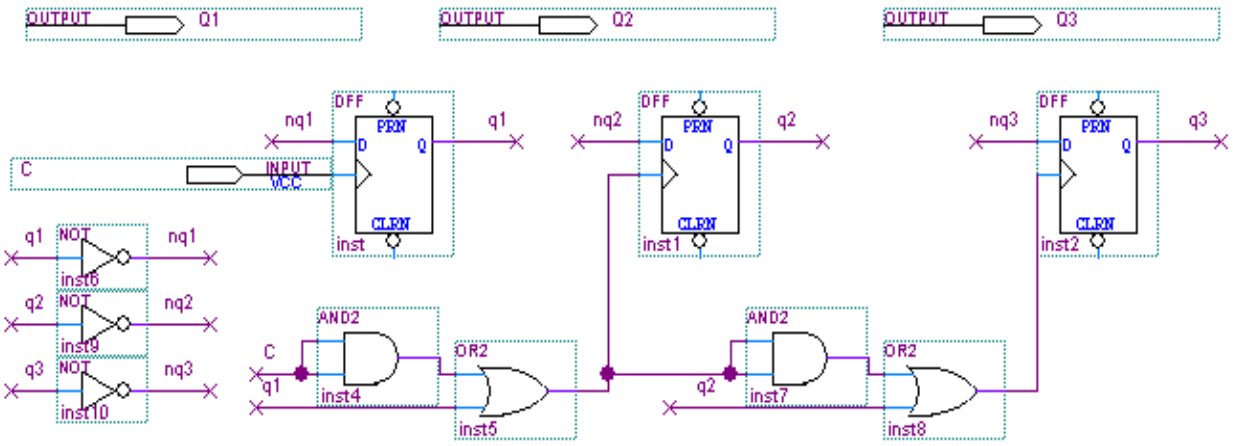



Рисунок 8.7 – Схема проверки асинхронного вычитающего счетчика со сквозным переносом на триггере dff

8 Исследуйте временные диаграммы работы построенного счетчика при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.8 в качестве примера приведены временные диаграммы работы вычитающего трехразрядного счетчика со сквозным переносом на триггере dff.

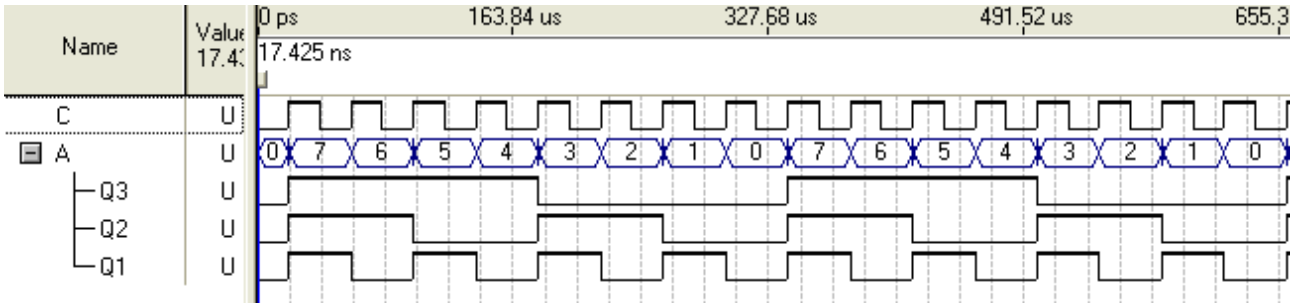



Рисунок 8.8 – Временные диаграммы асинхронного вычитающего счетчика со сквозным переносом на триггере dff

8.3.2 Контрольные вопросы

1 Дайте определение счетчика.
2 Назовите основные специфические параметры счетчиков.
3 Классификация счетчиков, принципы их построения.
4 Приведите схему и дайте краткую характеристику (укажите преимущества, недостатки и особенности работы) следующих типов счетчиков:

a) суммирующие;
b) вычитающие;
c) реверсивные;
d) асинхронные с последовательным переносом;
e) асинхронные со сквозным переносом;
f) синхронные со сквозным переносом;
g) синхронные с параллельным переносом;
h) синхронные с параллельным переносом на D-триггерах;
i) синхронные с параллельным переносом на JK-триггерах.

## 8.4 Лабораторная работа №3. Делители с произвольным постоянным коэффициентом деления

**Цель работы:** изучить временные диаграммы работы делителей с произвольным постоянным коэффициентом деления (ДПКД), построенных по различным схемам.

### 8.4.1 Порядок выполнения работы

1 Изучите до начала выполнения лабораторной работы методики синтеза делителей.

2 Синтезируйте схему исследования делителя в соответствии с установленным вариантом задания по таблице 8.5.

Таблица 8.5 – Варианты заданий

| № варианта | Метод синтеза | Коэффициент деления | Базовая микросхема |
|---|---|---|---|
| 1 | 2 | 5 | 74197 |
| 2 | 1 | 6 | 74196 |
| 3 | 2 | 7 | 74193 |
| 4 | 1 | 9 | 74192 |
| 5 | 2 | 5 | 74191 |
| 6 | 1 | 6 | 74190 |
| 7 | 2 | 7 | 74190 |
| 8 | 1 | 9 | 74191 |
| 9 | 2 | 5 | 74192 |
| 10 | 1 | 6 | 74193 |
| 11 | 2 | 7 | 74196 |
| 12 | 1 | 9 | 74197 |

Примечание. В таблице 8.5 методам синтеза условно присвоены следующие номера:

1 – метод исключения лишних состояний счетчика;

2 – метод искусственного насчёта импульсов в счетчике.

3 Создайте схему для исследования делителя в САПР Quartus II.

На рисунке 8.9 в качестве примера приведена схема для исследования делителя, выполненного на базовой микросхеме 74191 с коэффициентом деления 4.

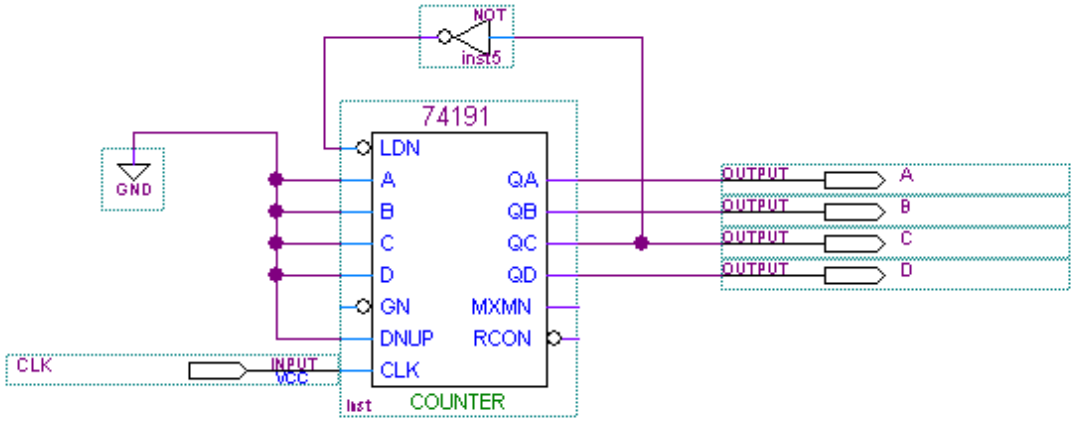



Рисунок 8.9 – Схема исследования делителя, выполненного на базовой микросхеме 74191

4 Исследуйте временные диаграммы работы делителя при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.10 в качестве примера приведены временные диаграммы работы построенного делителя.

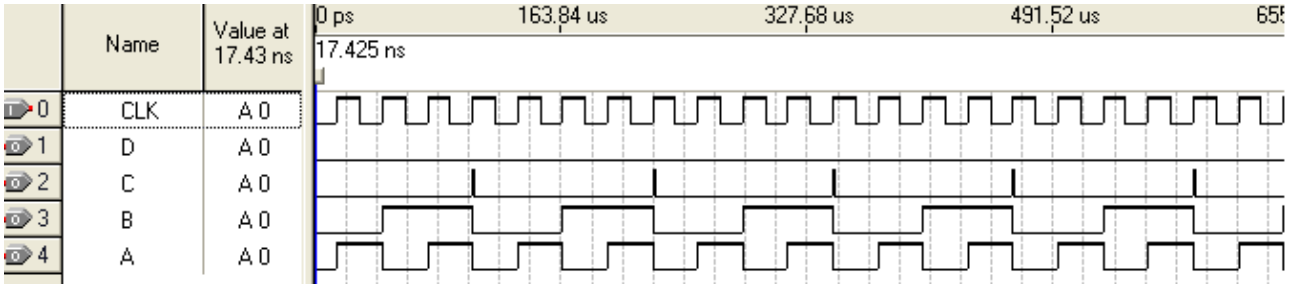



Рисунок 8.10 – Временные диаграммы работы делителя выполненного на базовой микросхеме 74191

5 Исследуйте и приведите в отчете назначение всех входов и выходов базовой микросхемы делителя.

6 Синтезируйте схему ДПКД по 4-му методу построения делителей (синтез счетчиков с использованием теории цифровых автоматов) в соответствии с требованиями, приведенными в таблице 8.6.

Таблица 8.6 – Варианты заданий

| **Номер варианта** | **Коэффициент деления** | **Тип триггера** |
|---|---|---|
| 1 | 9 | dff |
| 2 | 10 | jkff |
| 3 | 11 | tff |
| 4 | 9 | jkff |
| 5 | 10 | tff |
| 6 | 11 | dff |
| 7 | 9 | tff |
| 8 | 10 | jkff |
| 9 | 11 | dff |
| 10 | 9 | tff |
| 11 | 10 | dff |
| 12 | 11 | jkff |

В таблицах 8.7-8.11 приведены результаты разработки делителя на 6 на базе триггера jkff (в таблице 8.7 приведена расширенная таблица истинности работы делителя, в таблицах 8.8–8.11 даны карты Карно для нахождения сигналов на управляющих входах).

Таблица 8.7 – Таблица истинности работы делителя

| **Номер импуль-са** | **Текущее состояние** | | | **Следующее состояние** | | | **Управляющие сигналы** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Q3 | Q2 | Q1 | Q3 | Q2 | Q1 | J3 | K3 | J2 | K2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | X |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 | X |
| 2 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | X | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | X | 1 |
| 4 | 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | X |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 | X | 1 | 0 | X |

Таблица 8.8 – Карта Карно для J3

| Q3\Q2Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | X | X | X | X |

J3=Q2Q1

Таблица 8.9 – Карта Карно для K3

| Q3\Q2Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | X | X |
| 0 | 0 | 1 | X | X |

K3=Q1

Таблица 8.10 – Карта Карно для J2

| Q3\Q2Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | X | X |
| 1 | 0 | 0 | X | X |

J2=~Q3Q1

Таблица 8.11 – Карта Карно K2

| Q3\Q2Q1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | 1 | 0 |
| 1 | X | X | X | X |

K2=Q1

7 Создайте схему проверки синтезированного ДПКД.

На рисунке 8.11 в качестве примера приведена схема проверки делителя на 6.

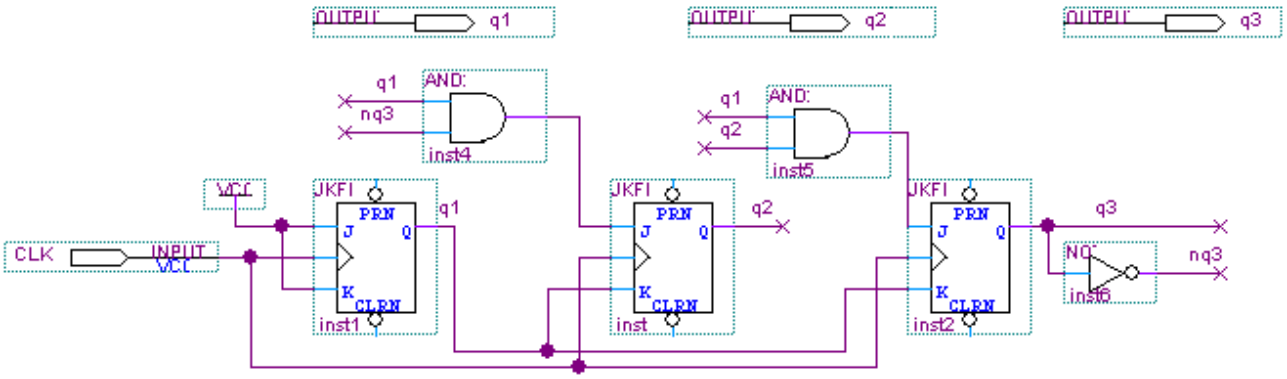



Рисунок 8.11 – Схема проверки делителя на 6

8 Исследуйте временные диаграммы работы построенного делителя при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.12 в качестве примера приведены временные диаграммы работы построенного делителя на 6.

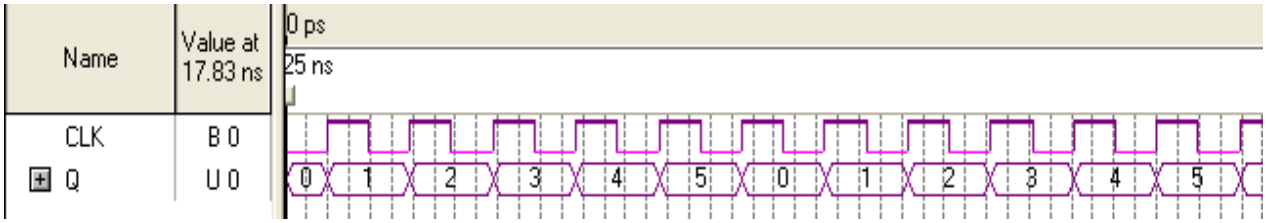



Рисунок 8.12 – Временные диаграммы работы делителя на 6

8.4.2 Контрольные вопросы

1 Дайте определение делителя.

2 Поясните методики синтеза ДПКД.

3 ДПКД на базе метода исключения лишних состояний. Укажите преимущества и недостатки.

4 ДПКД на базе метода искусственного принудительного насчета импульсов. Укажите преимущества и недостатки.

5 ДПКД на основе вычитающих счетчиков. Укажите преимущества и недостатки.

6 ДПКД на основе теории цифровых автоматов. Укажите преимущества и недостатки.

## 8.5 Лабораторная работа №4. Регистры

**Цель работы:** изучить временные диаграммы работы регистров, построенных по различным схемам.

### 8.5.1 Порядок выполнения работы

1 Изучите до начала выполнения лабораторной работы методики синтеза регистров.

2 Соберите схему проверки стандартного регистра, тип которого в зависимости от установленного варианта задания приведен в таблице 8.12.

Таблица 8.12 – Варианты заданий

| Номер варианта | Тип регистра |
|---|---|
| 1 | 74173 |
| 2 | 74174 |
| 3 | 74174b |
| 4 | 74175 |
| 5 | 74173 |
| 6 | 74174 |
| 7 | 74174b |
| 8 | 74175 |
| 9 | 74173 |
| 10 | 74174 |
| 11 | 74174b |
| 12 | 74175 |

3 Создайте схему для исследования указанного по варианту регистра в САПР Quartus II.

На рисунке 8.13 в качестве примера приведена схема для исследования регистра 74174b.


Рисунок 8.13 – Схема исследования регистра 74174b

4 Исследуйте временные диаграммы работы регистра при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.14 в качестве примера приведены временные диаграммы работы построенного регистра.

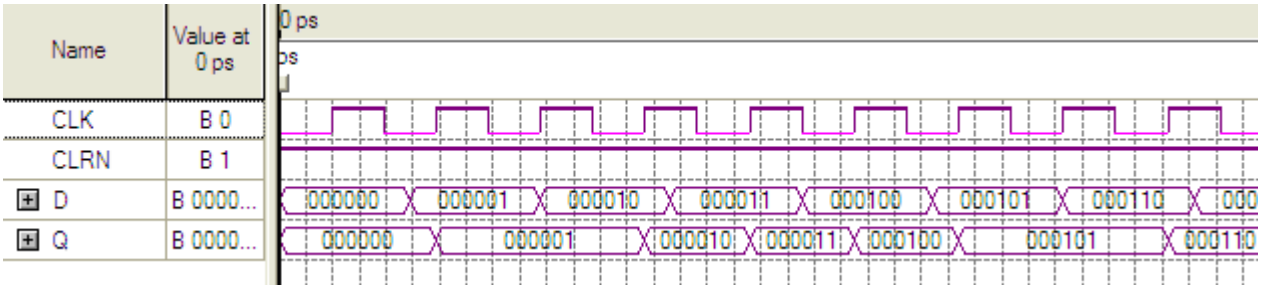



Рисунок 8.14 – Временные диаграммы работы регистра построенного на базовой микросхеме 74174b

5 Исследуйте и приведите в отчете назначение всех входов и выходов базовой микросхемы регистра.

6 Синтезируйте схему четырехразрядного регистра с учетом требований, приведенных в таблице 8.13.

Таблица 8.13 – Варианты заданий

| № варианта | Тип регистра | Триггер |
|---|---|---|
| 1 | Параллельно-параллельный | dff |
| 2 | Параллельно-последовательный | tff |
| 3 | Последовательно-параллельный | jkff |
| 4 | Последовательно-последовательный | dff |
| 5 | Параллельно-параллельный | tff |
| 6 | Параллельно-последовательный | jkff |
| 7 | Последовательно-параллельный | dff |
| 8 | Последовательно-последовательный | tff |
| 9 | Параллельно-параллельный | jkff |
| 10 | Параллельно-последовательный | dff |
| 11 | Последовательно-параллельный | dff |
| 12 | Последовательно-последовательный | tff |
| Примечание. Базовый триггер обязательно должен входить в состав разрабатываемого регистра. Допускается дополнительно использовать любые типы логических микросхем | | |

7 Создайте схему проверки синтезированного регистра.

На рисунке 8.15 в качестве примера приведена схема проверки трехразрядного параллельно-последовательного регистра построенного на триггерах jkff.

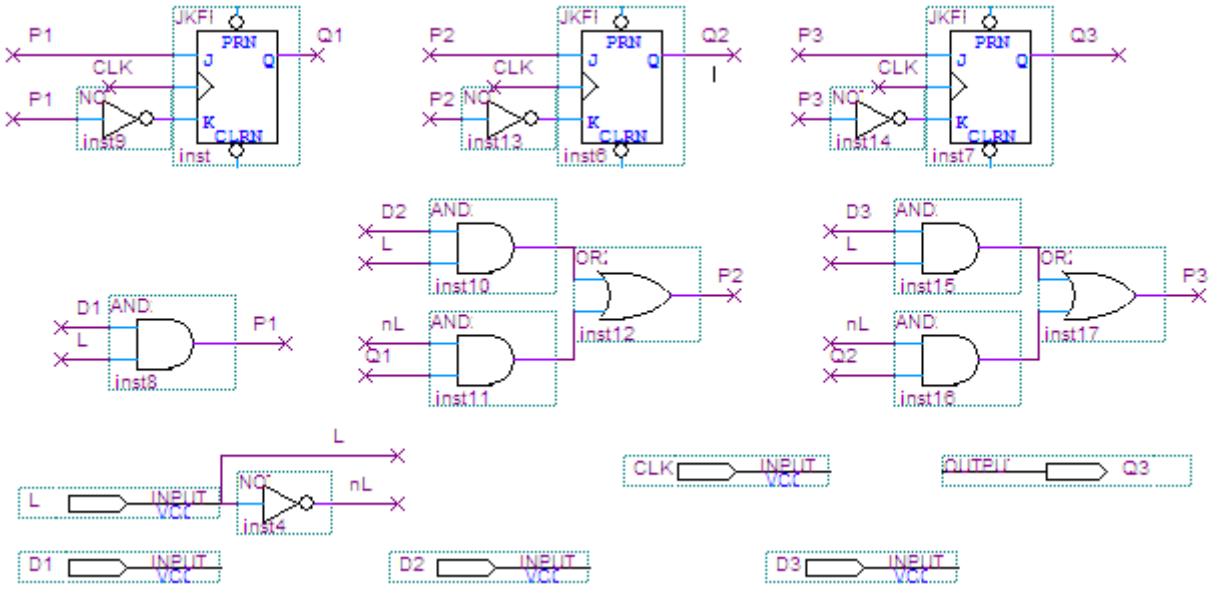



Рисунок 8.15 – Схема проверки трехразрядного параллельно-последовательного регистра построенного на триггерах jkff

8 Исследуйте временные диаграммы работы построенного регистра при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.16 в качестве примера приведены временные диаграммы работы трехразрядного параллельно-последовательного регистра построенного на триггерах jkff.

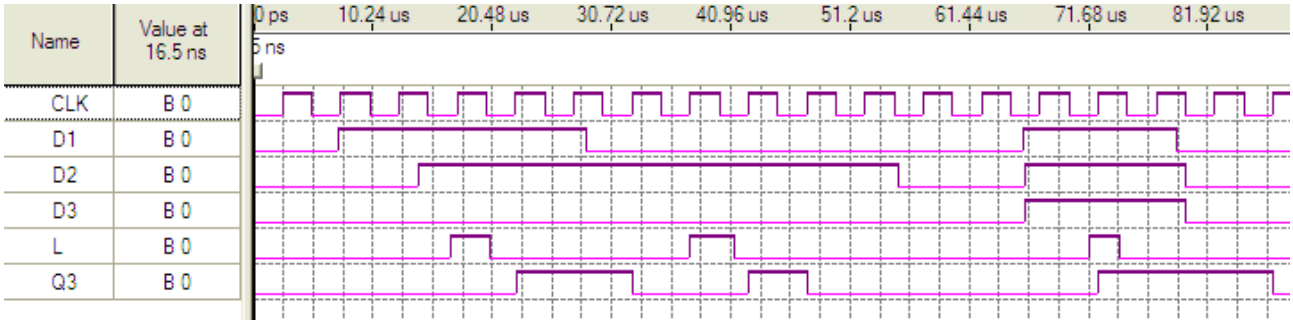



Рисунок 8.16 – Временные диаграммы трехразрядного параллельно-последовательного регистра построенного на триггерах jkff

8.5.2 Контрольные вопросы

1 Дайте определение регистра.
2 Принципы построения регистров.
3 Классификация регистров.
4 Основные параметры регистров.
5 Приведите схему и диаграммы работы, а также дайте краткую характеристику (укажите преимущества, недостатки, особенности работы) регистров следующих типов:

а) параллельные регистры на основе RS- триггеров;

b) параллельные регистры на основе D- триггеров. Какие D- триггеры (динамические или статические) рекомендуется использовать и почему?
c) последовательные регистры на основе JK- триггеров. Укажите требования к временным параметрам;
d) последовательные регистры на основе D- триггеров. Укажите требования к временным параметрам;
e) последовательно-параллельные регистры на основе JK- триггеров;
f) последовательно-параллельные регистры на основе D- триггеров;
g) параллельно-последовательные регистры на основе JK- триггеров;
h) параллельно-последовательные регистры на основе D- триггеров.

6 Универсальный регистр.

## 8.6 Лабораторная работа №5. Комбинированные устройства

**Цель работы:** разработать устройство, состоящее из нескольких компонентов и проверить его работоспособность.

### 8.6.1 Порядок выполнения работы

1 Изучите до начала выполнения лабораторной работы методики построения цифровых автоматов, комбинационных и последовательных устройств.

2 Изучите методику построения и использования компонентов в САПР Quartus II. Разрабатываемое устройство будет состоять из трех компонентов: конечного автомата, последовательного и комбинационного устройства.

3 Соберите схему первого компонента устройства – комбинационной схемы, в соответствии с вариантом задания, приведенным в таблице 8.14.

Таблица 8.14– Варианты заданий

| **Номер варианта** | **Входы** | **Реализуемая функция** |
|---|---|---|
| 1 | A, B, C, D | (A or B and C) xor D |
| 2 | A, B, C, D | (A and B or nC) or D |
| 3 | A, B, C, D | A xor B and C or D |
| 4 | A, B, C, D | (A or nB) and (C or D) |
| 5 | A, B, C, D | nA or (B and C xor D) |
| 6 | A, B, C, D | (nA or nB or C) and D |
| 7 | A, B, C, D | A and nB or C and D |
| 8 | A, B, C, D | A and nB or C and nD |
| 9 | A, B, C, D | nA and nB xor nC and D |
| 10 | A, B, C, D | (A or nB) or (nC or D) |
| 11 | A, B, C, D | nA and (nB and nC xor D) |
| 12 | A, B, C, D | (A and nB and C) or D |

Разработанный компонент должен иметь 4 входа и один выход. Выходная функция должна реализовывать функцию от четырех аргументов в соответствии с вариантом.

4 Исследуйте и приведите в отчете временные диаграммы работы компонента при произвольных фазовых соотношениях входных сигналов.

5 Создайте условное графическое обозначение созданного компонента для применения его в проекте, для чего выполните команду: Create/Update/Create Symbol Files for Current File.

6 Соберите схему второго компонента устройства – конечного автомата (с наложением) в соответствии с вариантом задания, приведенным в таблице 8.15.

Таблица 8.15 – Варианты заданий

| **Номер варианта** | **Вход установки** | **Метод создания** | **Тип автомата** | **Последова-тельность** |
|---|---|---|---|---|
| 1 | Асинхронный | 1 | Мура | 1001 |
| 2 | Низкий активный уровень | 2 | Мили | 0110 |
| 3 | Высокий активный уровень | 2 | Мура | 0010 |
| 4 | Синхронный | 1 | Мили | 0100 |
| 5 | Низкий активный уровень | 1 | Мура | 1000 |
| 6 | Высокий активный уровень | 2 | Мили | 1011 |
| 7 | Асинхронный | 2 | Мура | 1010 |
| 8 | Синхронный | 2 | Мили | 1100 |
| 9 | Высокий активный уровень | 1 | Мура | 1110 |
| 10 | Асинхронный | 1 | Мили | 1101 |
| 11 | Синхронный | 1 | Мура | 1111 |
| 12 | Низкий активный уровень | 2 | Мили | 1011 |
| Примечание – Методам создания конечного автомата условно присвоены номера:<br>– 1 – создание в графическом редакторе;<br>– 2 – разработка с помощью State Machine Wizard | | | | |

Автомат должен определять искомую последовательность и выдавать логическую «1» в случае ее обнаружения.

Рассмотрим пример создания в графическом редакторе конечного автомата, который выделяет последовательность 110:

Для создания цифрового автомата в САПР Quartus II с применением графического редактора следует выполнить следующие операции:

– создайте новый проект;

– откройте новый файл State Machine File (вкладка Design Files);

– введите в таблицы входных и выходных контактов необходимые сигналы и их имена (см. рисунок 8.17);

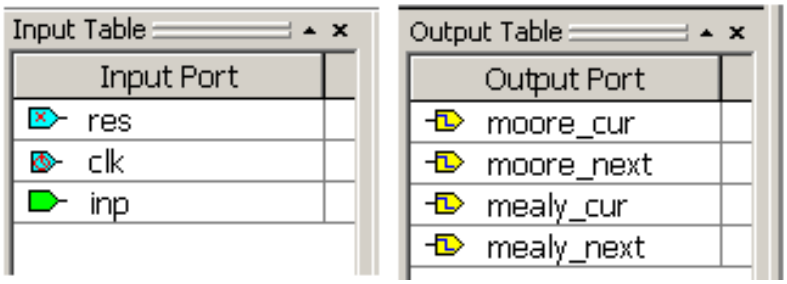



Рисунок 8.17 – Таблицы входных и выходных контактов

– определите во вкладке General окна State Table вид синхронизации сигнала установки (res) и его активный уровень (см. рисунок 8.18);

State Table

| | Option | Setting |
|---|---|---|
| 1 | Reset Mode | Synchronous |
| 2 | Reset Active Level | Active High |

General | States | Inputs | Outputs | Transitions | Actions

Рисунок 8.18 – Таблица параметров сигнала синхронизации

– укажите состояния цифрового автомата и дуги переходов между ними;

– укажите на дугах переходов выражения, определяющие условия переходов между состояниями цифрового автомата (см. рисунок 8.19 );

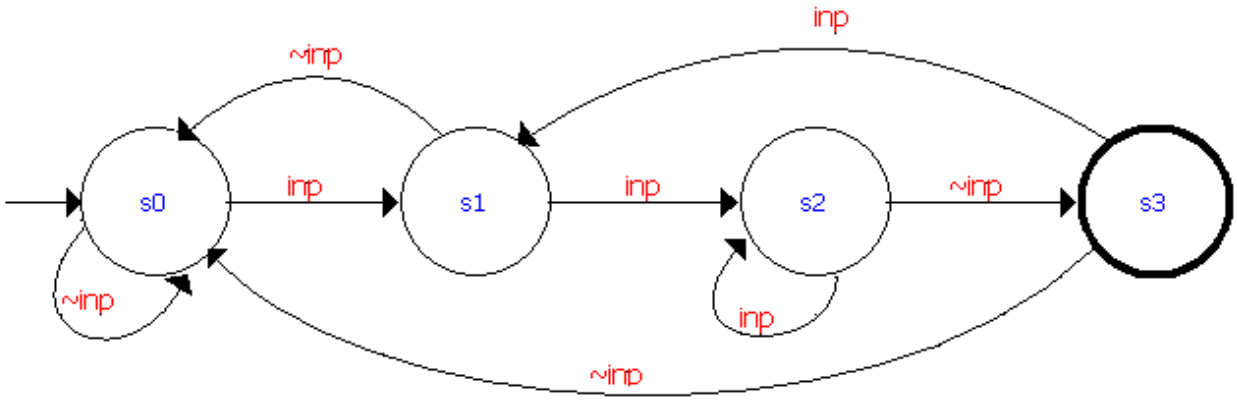



Рисунок 8.19 – Диаграмма состояний цифрового автомата

– определите во вкладках Outputs и Actions окна State Table алгоритм формирования выходного сигнала (см. рисунок 8.20);

State Table

| | Output Port | Registered | Output State |
|---|---|---|---|
| 1 | moore_cur | Yes | Current clock cycle |
| 2 | moore_next | Yes | Next clock cycle |
| 3 | mealy_cur | Yes | Current clock cycle |
| 4 | mealy_next | Yes | Next clock cycle |

General | States | Inputs | **Outputs** | Transitions | Actions

State Table

| | Output Port | Output Value | In State | Additional Conditions | Output State |
|---|---|---|---|---|---|
| 1 | moore_cur | 1 | s3 | | Current clock cycle |
| 2 | moore_next | 1 | s3 | | Next clock cycle |
| 3 | mealy_cur | 1 | s3 | inp | Current clock cycle |
| 4 | mealy_next | 1 | s3 | inp | Next clock cycle |

General | States | Inputs | Outputs | Transitions | **Actions**

Рисунок 8.20 – Таблицы параметров выходного сигнала

– укажите в окне Generate Others Files тип файла для автоматического создания условного графического обозначения цифрового автомата. Для выбора окна щелкните правой кнопкой “мыши” в любом месте графического редактора. Щелкните левой кнопкой “мыши” на кнопке “OK” (см. рисунок 8.21);

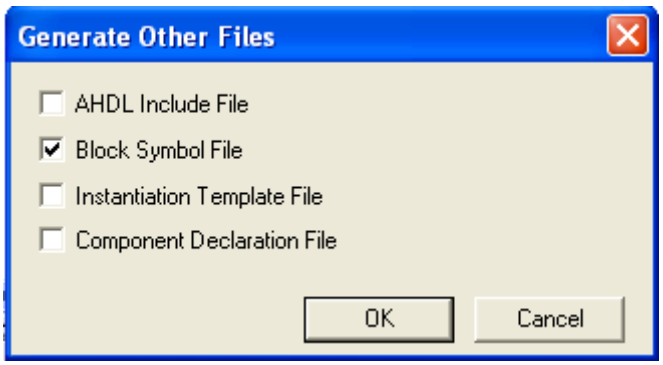



Рисунок 8.21 – Окно выбора автоматического создания УГО

– укажите язык VHDL для описания цифрового автомата (см. рисунок 8.22);

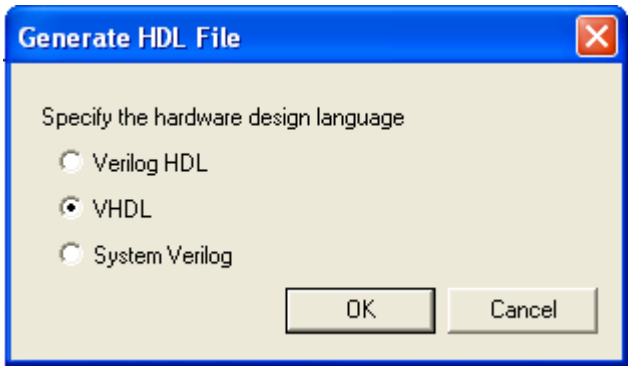



Рисунок 8.22 – Таблица выбора языка описания

– выполните автоматическую компиляцию описания цифрового автомата на языке VHDL;

– проверьте и откорректируйте, при необходимости, описание цифрового автомата;

– выполните компиляцию проекта;

– проверьте работоспособность цифрового автомата по результатам анализа временных диаграмм (см. рисунок 8.23) .

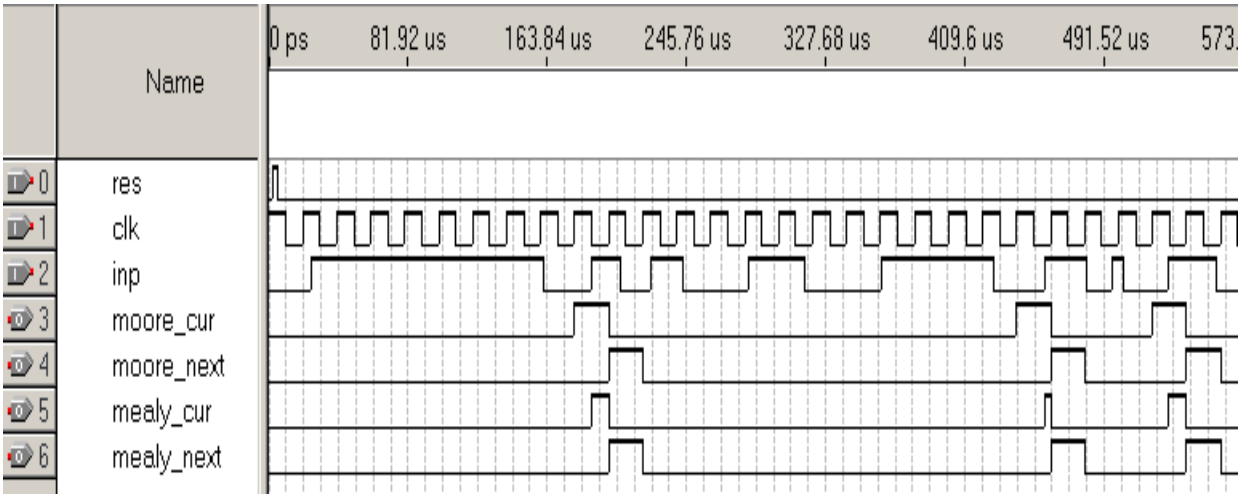



Рисунок 8.23 – Временные диаграммы работы цифрового автомата

Для создания цифрового автомата в САПР Quartus II с применением мастера создания цифровых автоматов следует выполнить следующие операции:

– создайте новый проект;

– откройте новый файл State Machine File (вкладка Design Files);

– запустите State Machine Wizard;

– выберите в диалоговом окне «State Machine Wizard» опцию «Create a new state machine design» ; (см. рисунок 8.24)

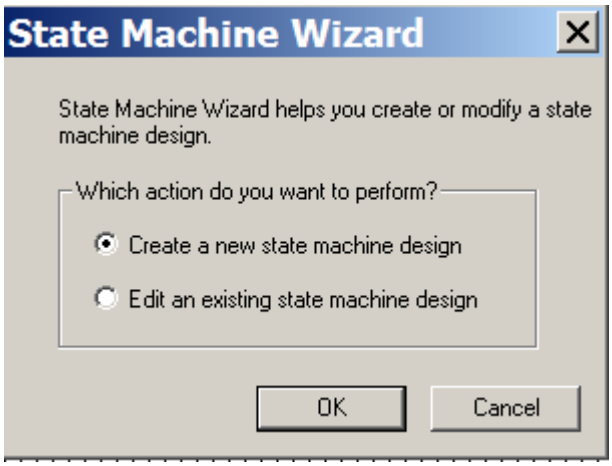



Рисунок 8.24 – Таблица выбора создания нового автомата

– выберите на первой странице тип автомата (Synchronous или Asynchronous), активный уровень сигнала Reset и тип выходного контакта (Register) (см. рисунок 8.25);

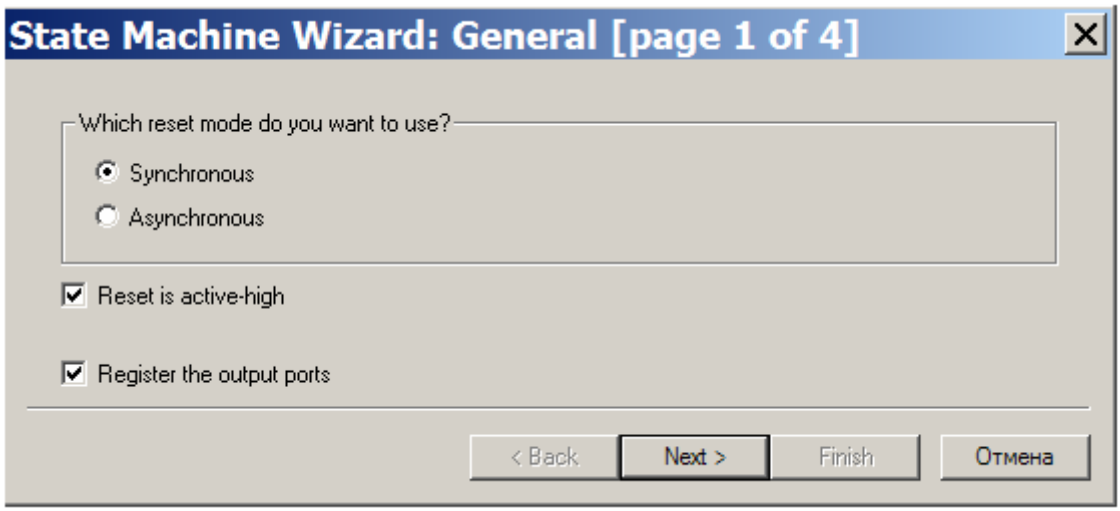



Рисунок 8.25 – Таблица выбора параметров автомата

– укажите в таблице имена состояний цифрового автомата и входных контактов, а также условия переходов между состояниями цифрового автомата (см. рисунок 8.26);

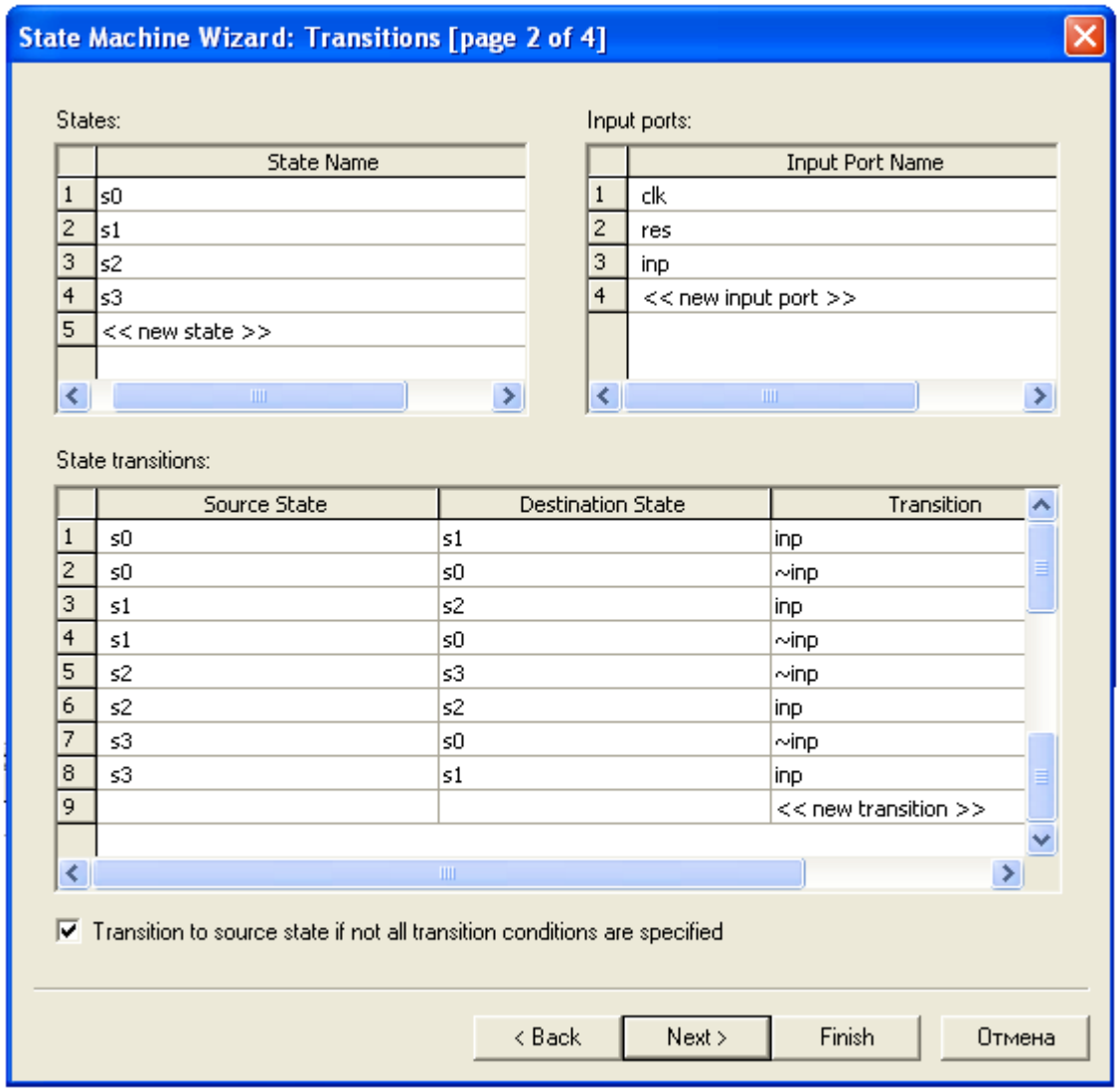



Рисунок 8.26 – Таблица выбора сигналов и состояний цифрового автомата

– определите алгоритм формирования выходного сигнала(см. рисунок 8.27) ;

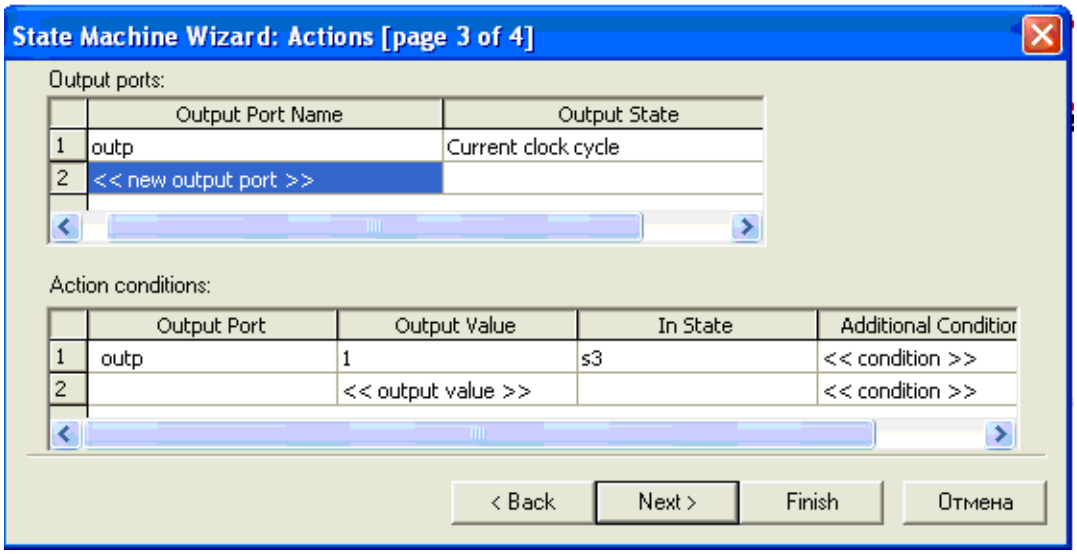



Рисунок 8.27 – Таблица выбора алгоритм формирования выходного сигнала

– проконтролируйте правильность ввода имен состояний, а также входных и выходных контактов (см. рисунок 8.28);

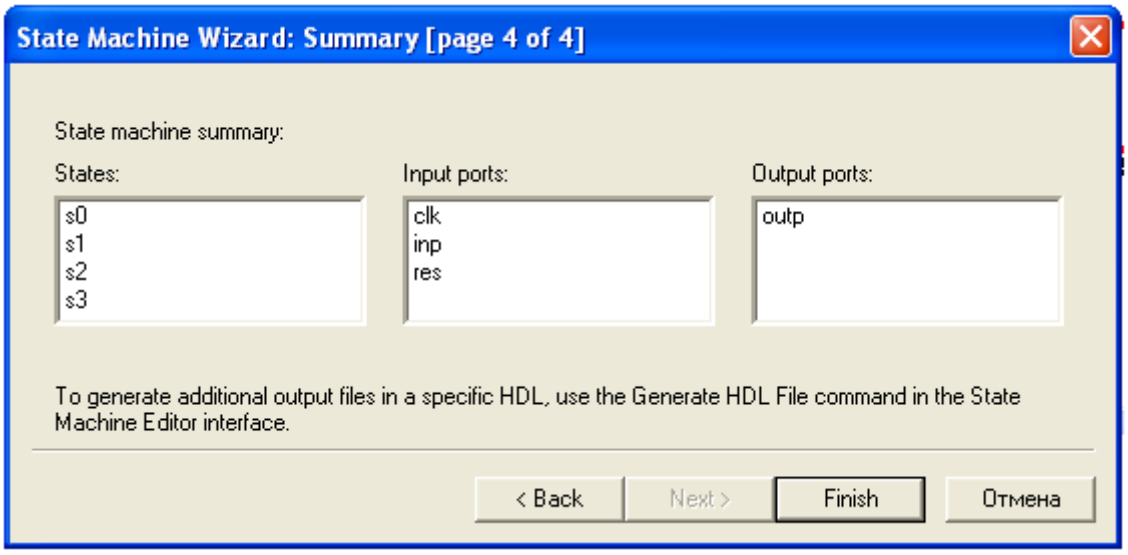



Рисунок 8.28 – Таблица контроля состояний, входных и выходных сигналов цифрового автомата

– проконтролируйте диаграмму состояний цифрового автомата (см. рисунок 8.29);

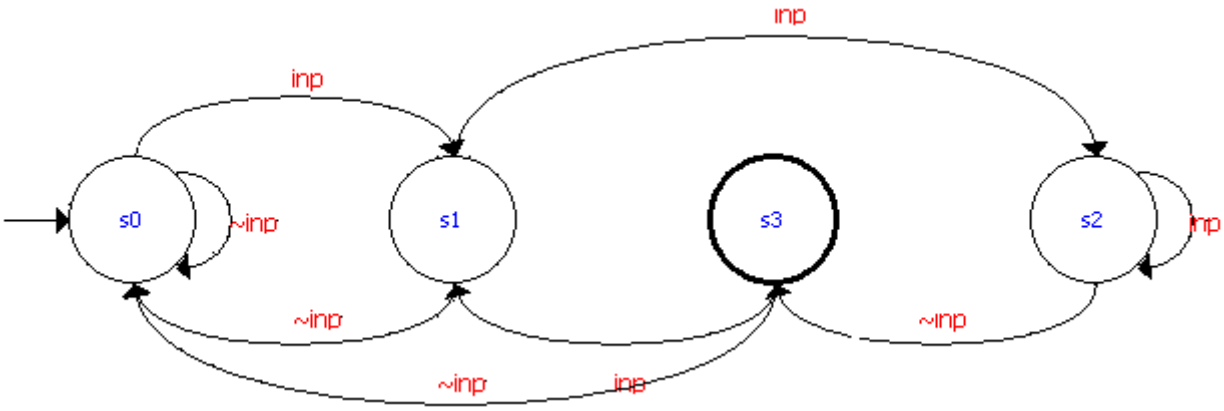



Рисунок 8.29 – Диаграмма состояний цифрового автомата

– укажите тип файла для автоматического создания условного графического обозначения цифрового автомата и язык VHDL для его описания;

– выполните автоматическую компиляцию описания цифрового автомата на языке VHDL;

– проверьте и откорректируйте, при необходимости, описание цифрового автомата;

– выполните компиляцию проекта;

– проверьте работоспособность цифрового автомата по результатам анализа временных диаграмм.

6 Исследуйте и приведите в отчете временные диаграммы работы разработанного автомата при произвольных фазовых соотношениях входных сигналов.

7 Соберите схему счетчика в соответствии с вариантом задания, приведенным в таблице 8.16.

Таблица 8.16– Варианты заданий

| Номер варианта | Коэффициент пересчета | Тип триггера |
|---|---|---|
| 1 | 3 | jkff |
| 2 | 4 | dff |
| 3 | 5 | jkff |
| 4 | 6 | dff |
| 5 | 3 | dff |
| 6 | 4 | jkff |
| 7 | 5 | dff |
| 8 | 6 | jkff |
| 9 | 3 | dff |
| 10 | 2 | tff |
| 11 | 4 | dff |
| 12 | 6 | jkff |

Разрабатываемый счетчик должен содержать один счетный вход и один выход. Выход счетчика должен сигнализировать о достижении требуемого коэффициента пересчета выдачей сигнала с уровнем логической единицы.

8 Исследуйте и приведите в отчете временные диаграммы работы разработанного счетчика и проверьте правильность его функционирования.

9 Соберите схему из трех созданных компонентов, соединенных в следующей последовательности: комбинационное устройство, конечный автомат, последовательное устройство (см. рисунок 8.30).

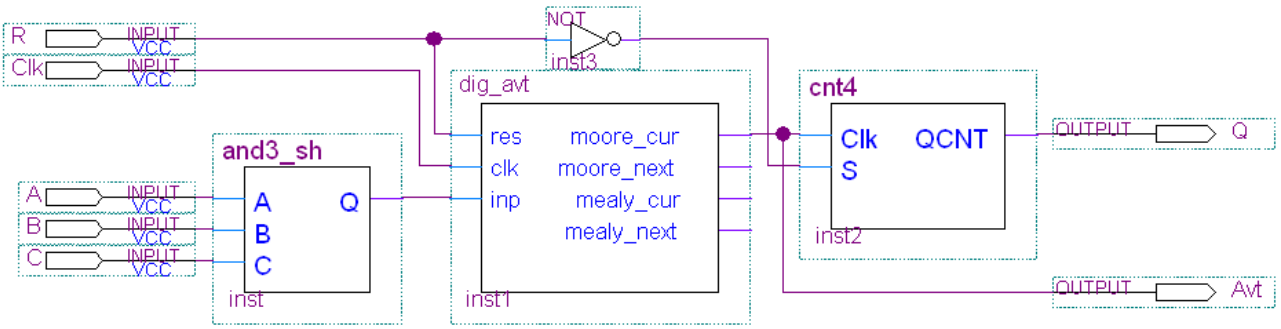



Рисунок 8.30 – Схема комбинированного устройства

10 Исследуйте и приведите в отчете временные диаграммы работы разработанного устройства и проверьте правильность его функционирования (см. рисунок 8.31).

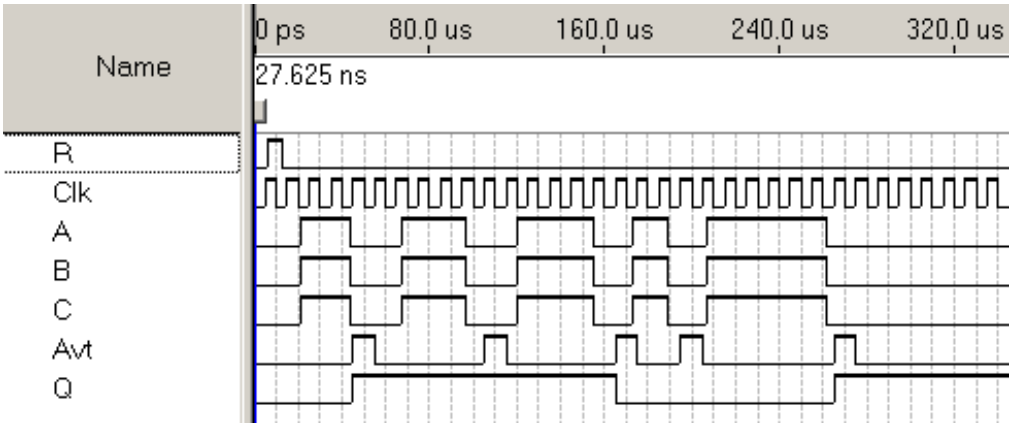



Рисунок 8.31 – Временные диаграммы работы комбинированного устройства

### 8.6.2 Контрольные вопросы

1 Дайте определение автомата.
2 Какие виды автоматов вы знаете?
3 Дайте определение комбинационного устройства. Приведите примеры.
4 Какое устройство называется последовательным? Примеры.
5 Какими способами можно создать условное графическое обозначение компонента в САПР Quartus II?

## 8.7 Лабораторная работа №6. Запоминающие устройства

**Цель работы:** изучить временные диаграммы работы запоминающих устройств, построенных по различным схемам.

### 8.7.1 Порядок выполнения работы

1 Изучите до начала выполнения лабораторной работы методики построения оперативных и постоянных запоминающих устройств.

2 Изучите методику создания при помощи утилиты MegaWizard Plug-in Manager компонентов и их использования в САПР Quartus II.

3 Соберите схему для исследования постоянного запоминающего устройства в САПР Quartus II согласно варианта задания, приведенного в таблице 8.17.

Для создания постоянного запоминающего устройства в САПР Quartus II при помощи утилиты MegaWizard Plug-in Manager следует выполнить такие операции:

– создайте новый проект;

– выберите в качестве элементной базы для реализации любую из микросхем ПЛИС, обязательно принадлежащих только семейству Cyclone (например, EP1C3T100A8);

– создайте новый Memory Initialization File (выберите необходимый пункт из меню File -> New);

– задайте необходимые значения для запрашиваемых параметров (например: Word size: 8, Number of words: 32) ;

– заполните открывшийся файл необходимыми данными, которые будут содержаться в ячейках ПЗУ (в качестве примера в таблице 8.18 приведено заполнение файла LpmRom.mif);

– сохраните заполненный файл инициализации памяти (рекомендуется располагать его в каталоге проекта) и закройте его;

– создайте новый Block Diagram/Schematic File (выберите необходимый пункт из меню File -> New);

– выберите пункт Insert Symbol из меню Edit (аналогичное действие производится по двойному нажатию мышки в любом пустом месте Block Diagram/Schematic File;

– найдите и выберите из раскрывающегося списка в левой части появившегося окна параметризированную мегафункцию lpm_rom, которая находится в разделе megafunctions -> storage стандартных библиотек САПР Quartus II как это показано на рисунке 8.32);

– завершите подтверждение выбора нажатием кнопки OK;

Таблица 8.17– Варианты заданий

| **Номер варианта** | **Наличие триггеров защёлки для выходных сигналов** | **Разрядность N выходной шины данных, бит** | **Количество N-битных слов памяти** | **Способ тактирования операций записи/чтения** |
|---|---|---|---|---|
| 1 | есть | 4 | 32 | общий тактовый сигнал |
| 2 | нет | 6 | 64 | раздельные сигналы inclock и outclock |
| 3 | есть | 8 | 32 | раздельные сигналы inclock и outclock |
| 4 | нет | 10 | 64 | раздельные сигналы inclock и outclock |
| 5 | есть | 12 | 32 | общий тактовый сигнал |
| 6 | нет | 14 | 64 | общий тактовый сигнал |
| 7 | есть | 15 | 32 | раздельные сигналы inclock и outclock |
| 8 | нет | 13 | 64 | раздельные сигналы inclock и outclock |
| 9 | есть | 11 | 32 | раздельные сигналы inclock и outclock |
| 10 | нет | 9 | 64 | общий тактовый сигнал |
| 11 | есть | 7 | 32 | общий тактовый сигнал |
| 12 | нет | 5 | 64 | общий тактовый сигнал |

Таблица 8.18 – Содержимое файла инициализации ПЗУ

| **Addr** | **+0** | **+1** | **+2** | **+3** | **+4** | **+5** | **+6** | **+7** |
|---|---|---|---|---|---|---|---|---|
| 0 | 0F | F0 | 11 | 22 | 33 | 44 | 55 | 66 |
| 8 | 77 | 88 | 99 | AA | BB | CC | DD | EE |
| 16 | FF | 00 | 01 | 02 | 03 | 04 | 05 | 06 |
| 24 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E |

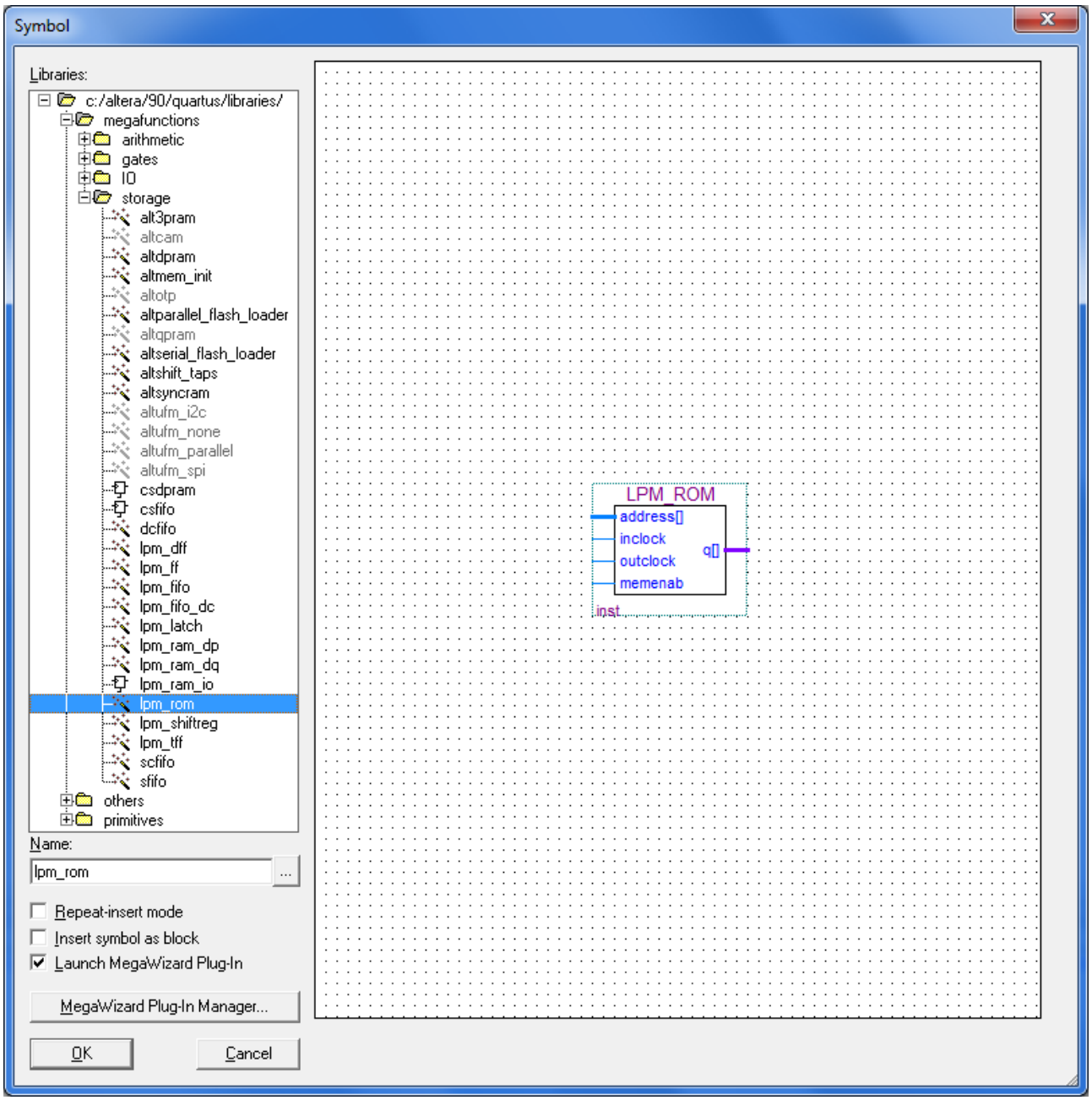



Рисунок 8.32 – Выбор параметризированной мегафункции lpm_rom

– задайте имя создаваемого компонента (см. рисунок 8.33), соблюдая правила именования, в открывшемся окне утилиты MegaWizard Plug-in Manager (см. рисунок 8.33) при необходимости и в случае, если САПР автоматически не выполнила эту операцию (например, не задала имя по умолчанию lpm_rom0);

– установите необходимое значение в списке выбора языка описания компонента (настоятельно рекомендуется использовать язык VHDL);

– нажмите кнопку Next >;

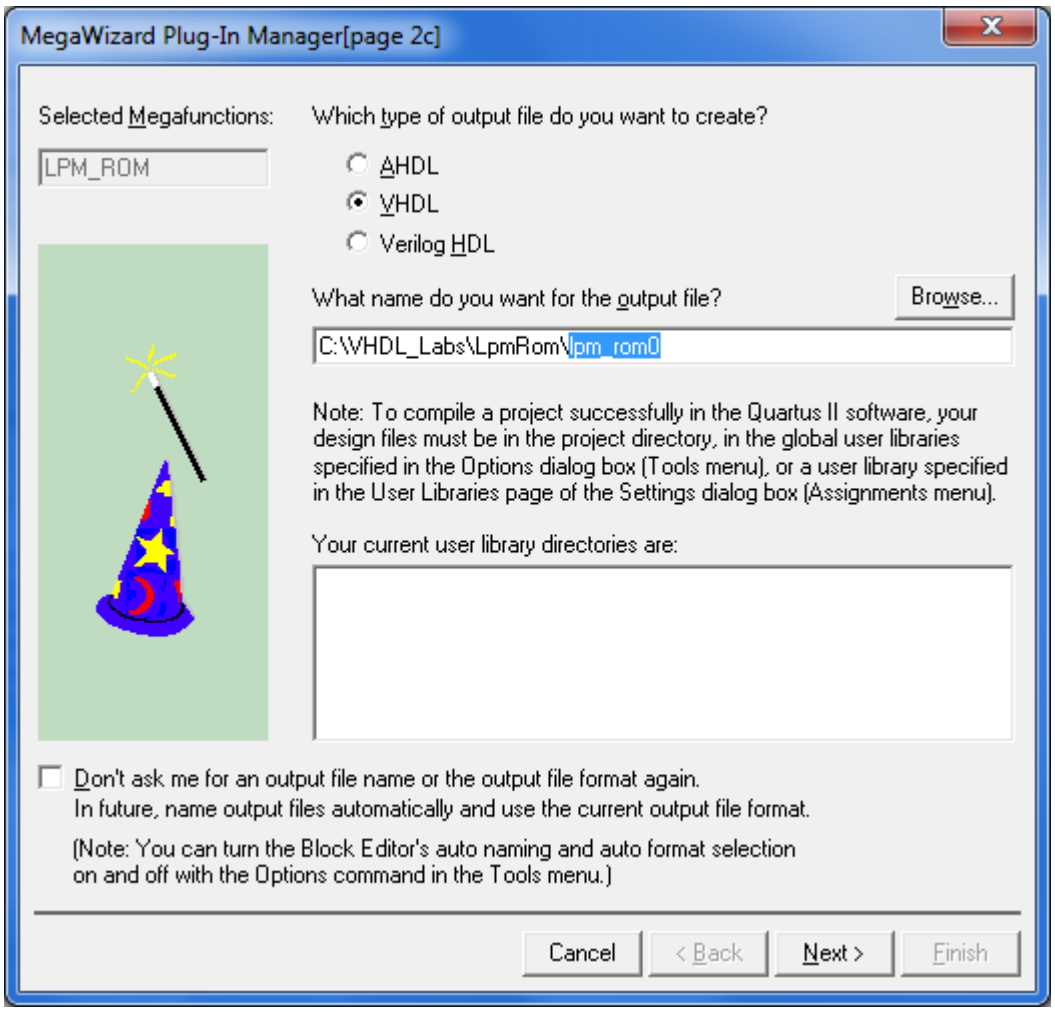



Рисунок 8.33 – Первый шаг создания ПЗУ

– установите необходимое значение в списке выбора разрядности выходной шины данных ПЗУ, как это показано на рисунке 8.34 (в качестве примера было выбрано значение 8 бит);

– установите необходимое значение в списке выбора общего объёма памяти (см. рисунок 8.34), которое задаётся как количество N-битных слов, где N – разрядность выходной шины данных ПЗУ (в качестве примера было выбрано значение 32);

– выполните проверку соответствия между заданными значениями разрядности и объёма памяти, которые указывались при создании Memory Initialization File, и теми, которые установлены в окне утилиты MegaWizard Plug-in Manager;

– внесите необходимые изменения содержимого Memory Initialization File, в случае возникновения указанных выше несоответствий параметров, поскольку при этом не гарантируется корректная работа ПЗУ (причём, САПР Quartus II не генерирует какие-либо сообщения об ошибках или предупреждения);

– установите значение Auto в списке выбора типа блока памяти (в этом случае компилятор и трассировщик самостоятельно учтут все особенности внутренней реализации ПЗУ для выбранной микросхемы ПЛИС);

– установите необходимое значение в списке выбора способа тактирования операций с памятью: Single clock – общий тактовый сигнал, по которому осуществляется защёлкивание входного адреса ячейки ПЗУ и чтение данных из ячейки с заданным адресом; input, output clocks – раздельные сигналы для защёлкивания адреса ячейки и чтения данных соответственно (в качестве примера был выбран второй способ тактирования операций);

– выполните повторную проверку соответствия всех заданных параметров;

– нажмите кнопку Next >;

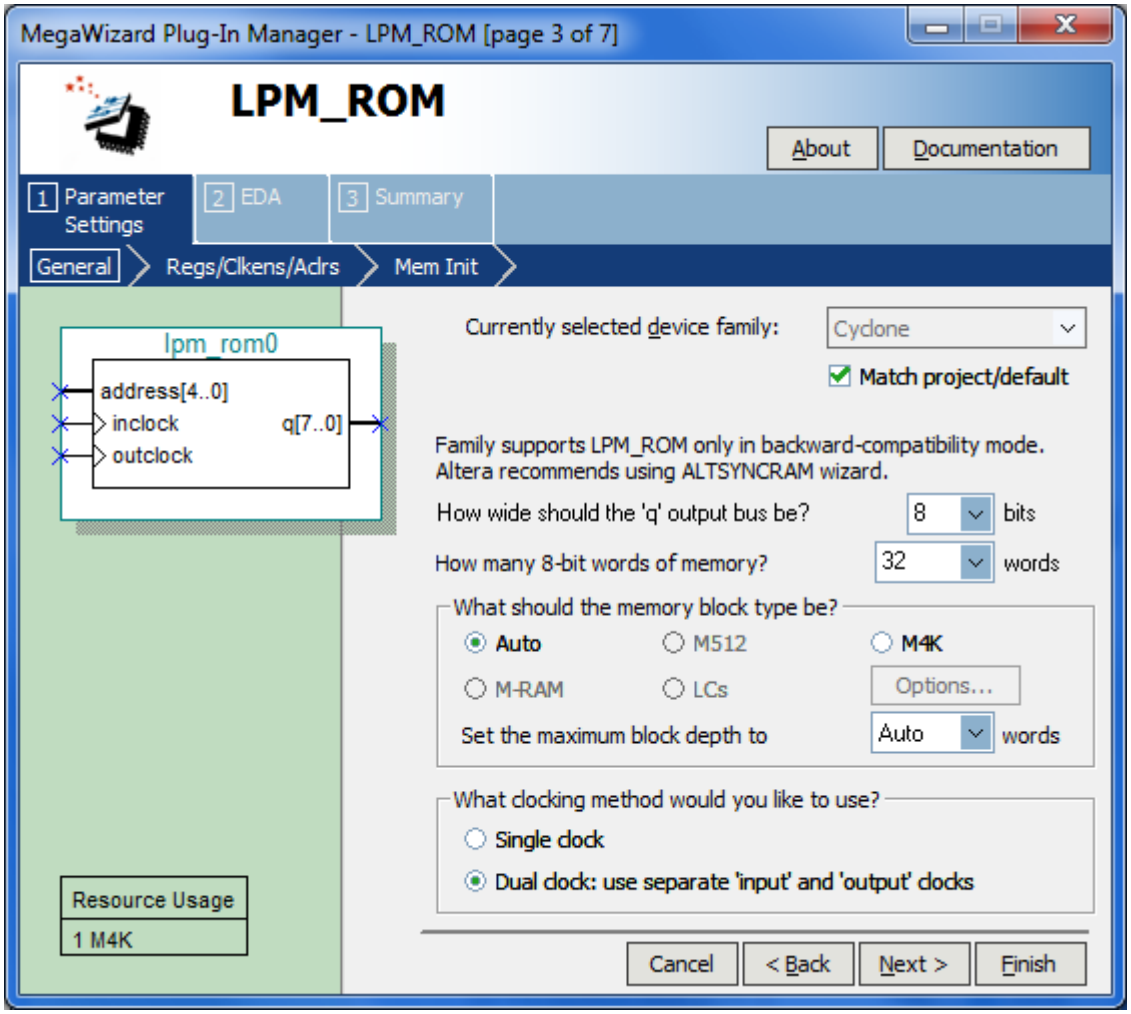



Рисунок 8.34 – Второй шаг создания ПЗУ

– установите (см. рисунок 8.35) соответствующее значение в списке выбора вариантов установки триггеров-защёлок для выходных портов сигналов (создаваемое в качестве примера ПЗУ снабжено соответствующими триггерами), для входных портов значение данной опции задано по умолчанию, его изменять нельзя, поэтому все входные сигналы имеют соответствующие триггеры-защёлки;

– нажмите кнопку Next >;

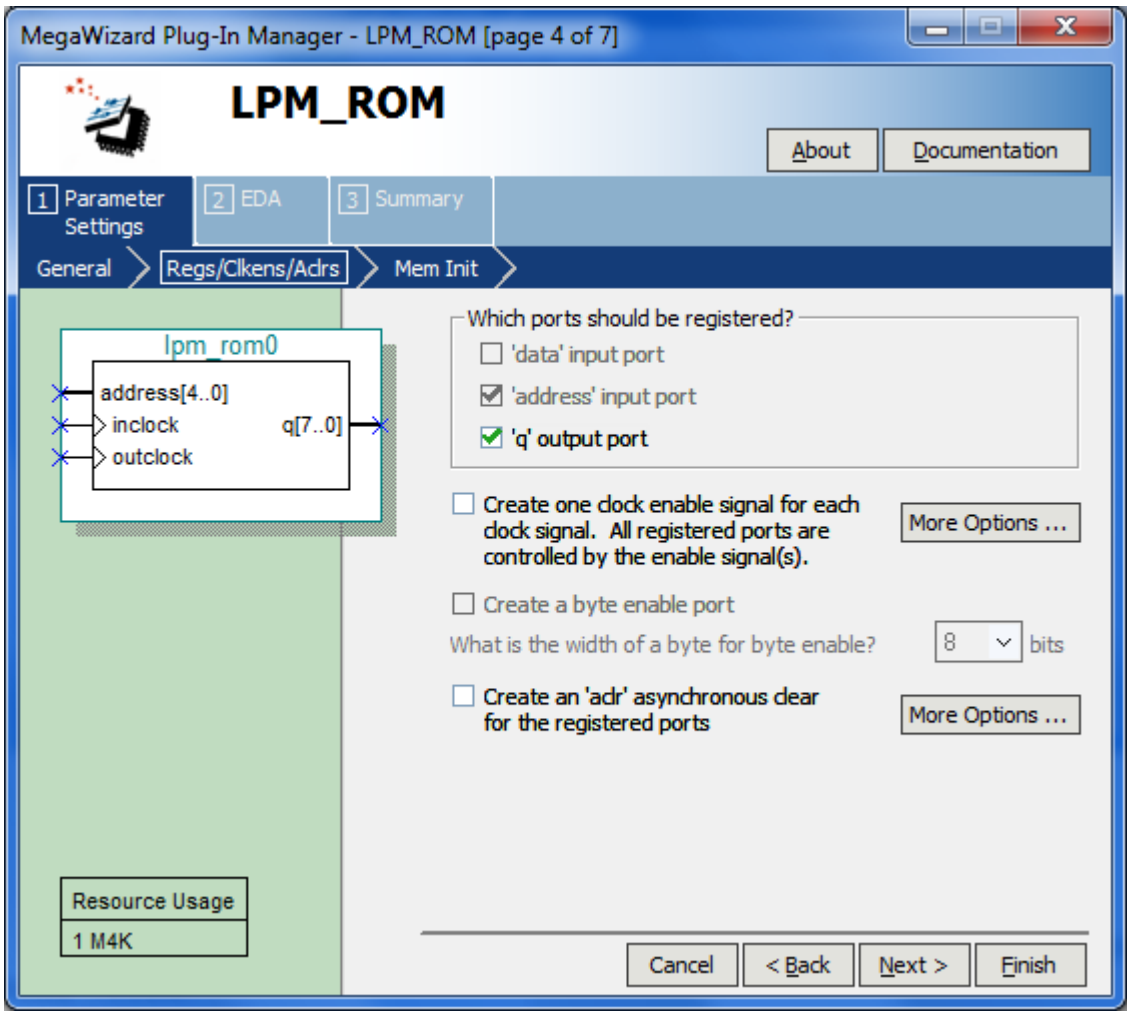



Рисунок 8.35 – Третий шаг создания ПЗУ

– укажите в поле File name: путь, по которому размещён созданный ранее Memory Initialization File, как показано на рисунке 8.36 (./LpmRom.mif означает, что этот файл находится в каталоге проекта);

– выберите из соответствующего списка расширение файла *.mif (а не *.hex!) в процессе поиска пути к Memory Initialization File;

– нажмите кнопку Next >;

– нажмите кнопку Next > внизу появившегося окна (не следует вносить сюда никаких изменений);

– нажмите кнопку Finish, после появления окна, изображённого на рисунке 8.37, для завершения процесса создания компонента ПЗУ с помощью утилиты MegaWizard Plug-in Manager в САПР Quartus II.

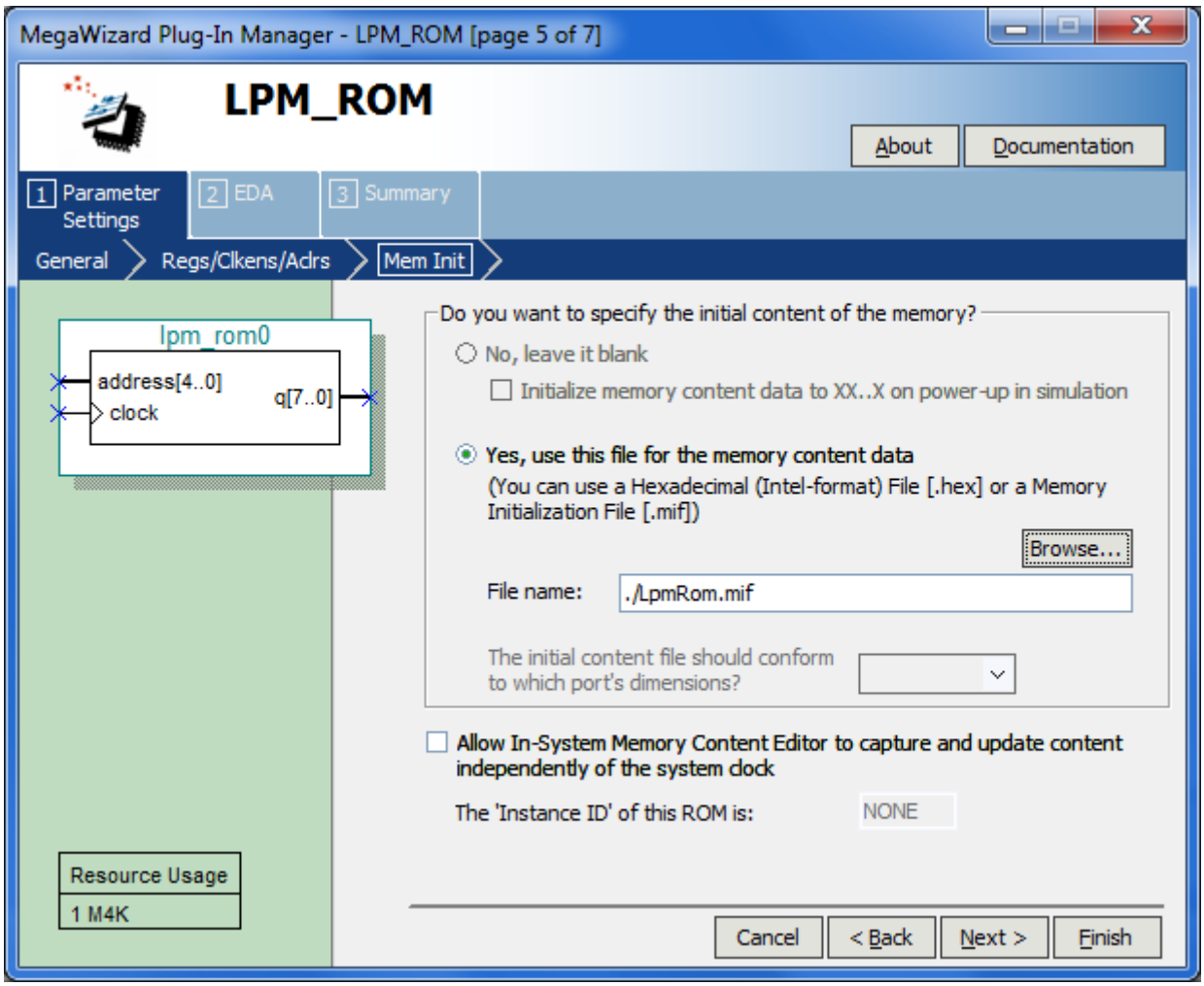



Рисунок 8.36 – Четвёртый шаг создания ПЗУ

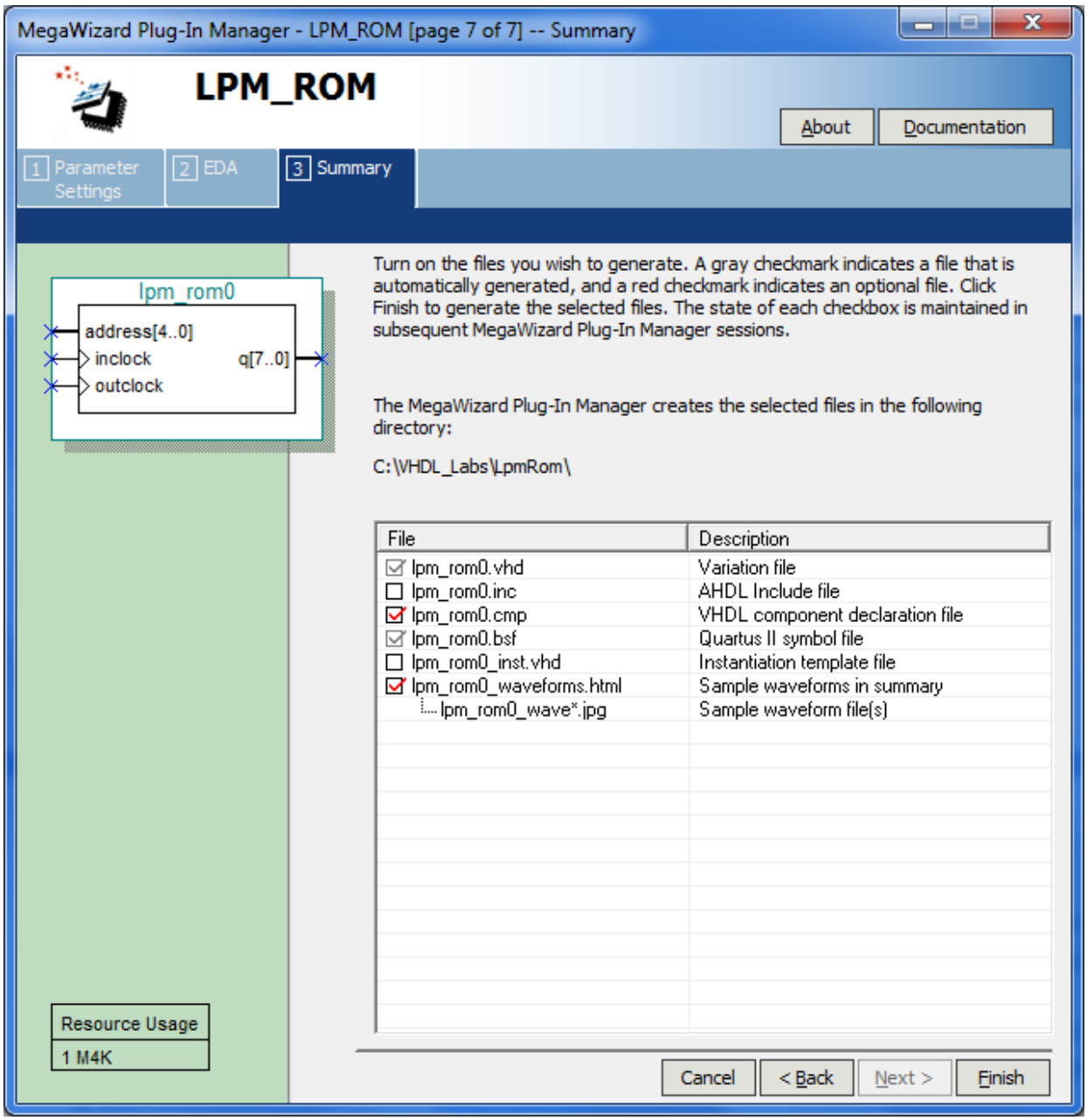



Рисунок 8.37 – Итоговое окно создания ПЗУ

На рисунке 8.38 в качестве примера приведена схема для исследования синтезированного при помощи утилиты MegaWizard Plug-in Manager постоянного запоминающего устройства на базе параметризируемой мегафункции lpm_rom.

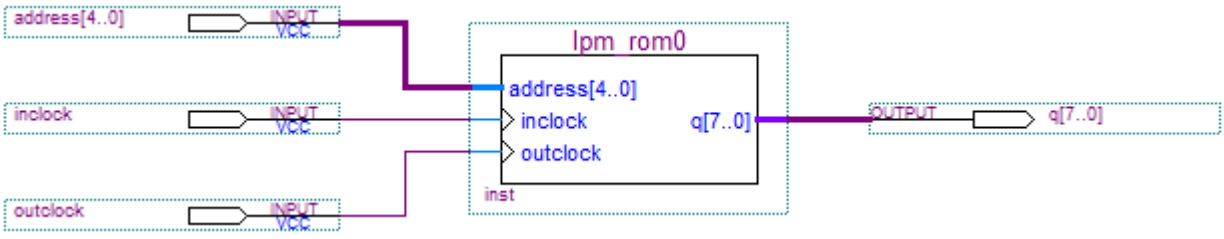



Рисунок 8.38 – Схема исследования созданного ПЗУ

4 Исследуйте временные диаграммы работы синтезированного ПЗУ при произвольных фазовых соотношениях входных сигналов.

На рисунке 8.39 в качестве примера приведены временные диаграммы работы ПЗУ, процесс создания которого был описан выше.

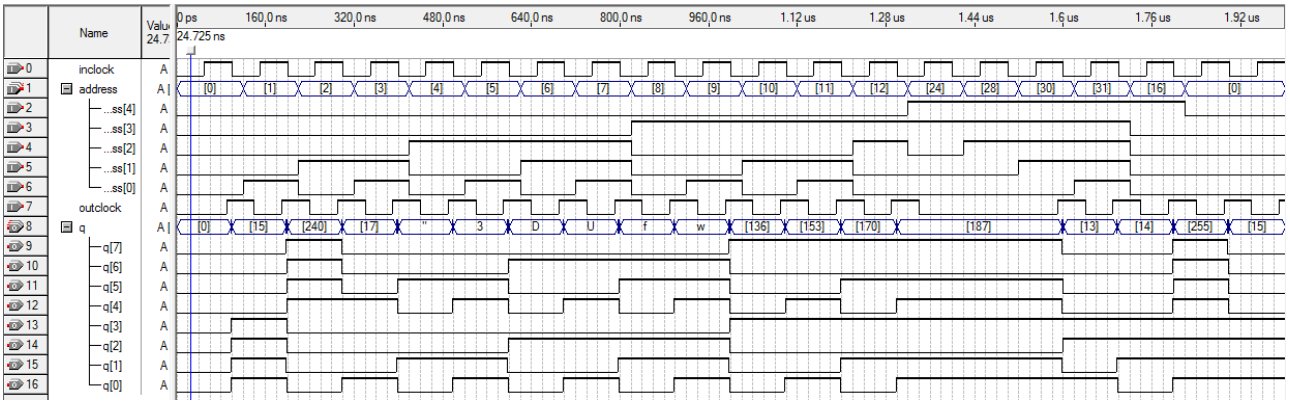



Рисунок 8.39 – Временные диаграммы работы синтезированного ПЗУ

У компонента ПЗУ, созданного в качестве примера, выходная шина данных имеет соответствующие триггеры-защёлки. Это означает, что чтение данных из ячейки ПЗУ с заданным адресом будет выполняться за 2 такта (при отсутствии выходных триггеров чтение выбранной ячейки памяти происходит за 1 такт): в первом такте происходит выборка данных из ячейки ПЗУ, а во втором – эти данные защёлкиваются выходными триггерами. Сказанное выше касается так же и ОЗУ. Однако запись данных в ОЗУ происходит всегда за 1 такт соответствующего сигнала и не зависит от наличия регистров защёлки входных сигналов.

Из анализа временных диаграмм можно сделать вывод, что по каждому переднему фронту импульса на входе inclock происходит защёлкивание адреса ячейки памяти, который установлен на шине address[4..0]. По каждому переднему фронту импульса на входе outclock происходит защёлкивание данных из соответствующей ячейки (значения ячеек памяти заданы при помощи LpmRom.mif) выходными триггерами-защёлками, т.е., по сути, происходит чтение данных. Поэтому при отсутствии импульсов на входе outclock данные на выходе ПЗУ не изменяются (даже при наличии сигналов защёлки адреса).

5 Соберите схему для исследования однопортового оперативного запоминающего устройства на основании параметризируемой мегафункции lpm_ram_dp, согласно варианта задания, приведенного в таблице 8.19.

6 Исследуйте временные диаграммы работы синтезированного ОЗУ при произвольных фазовых соотношениях входных сигналов.

7 Соберите схему для исследования оперативного запоминающего устройства на основании параметризируемой мегафункции lpm_ram_dq, согласно варианта задания, приведенного в таблице 8.20.

8 Исследуйте временные диаграммы работы синтезированного ОЗУ при произвольных фазовых соотношениях входных сигналов.

Таблица 8.19 – Варианты заданий

| **Номер варианта** | **Разрядность входной и выходной шин данных, бит** | **Общий объём памяти, бит** | **Наличие триггеров защёлки для выходных сигналов** | **Способ тактирования операций записи/чтения** |
|---|---|---|---|---|
| 1 | 4 | 128 | есть | общий тактовый сигнал |
| 2 | 8 | 256 | нет | раздельные сигналы read и write |
| 3 | 16 | 512 | нет | раздельные сигналы input и output |
| 4 | 16 | 128 | нет | раздельные сигналы read и write |
| 5 | 8 | 256 | есть | раздельные сигналы input и output |
| 6 | 4 | 512 | есть | оощий тактовый сигнал |
| 7 | 8 | 128 | нет | раздельные сигналы input и output |
| 8 | 16 | 256 | нет | раздельные сигналы read и write |
| 9 | 4 | 512 | есть | общий тактовый сигнал |
| 10 | 16 | 128 | есть | раздельные сигналы input и output |
| 11 | 4 | 256 | есть | общий тактовый сигнал |
| 12 | 8 | 512 | нет | раздельные сигналы read и write |
| Примечание. В качестве элементной базы для реализации ОЗУ выбирать любую микросхему ПЛИС, принадлежащую только семейству Cyclone.<br>Для вариантов задания №1 – №6 обязательно осуществить начальную инициализацию содержимого памяти, привести в отчёте содержимое файла инициализации. | | | | |

Таблица 8.20 – Варианты заданий

| **Номер варианта** | **Разрядность N входной и выходной шин данных, бит** | **Количество N-битных слов памяти** | **Наличие триггеров защёлки для выходных сигналов** | **Способ тактирования операций записи/чтения** |
|---|---|---|---|---|
| 1 | 5 | 128 | есть | общий тактовый сигнал |
| 2 | 6 | 128 | нет | раздельные сигналы input и output |
| 3 | 7 | 128 | есть | общий тактовый сигнал |
| 4 | 8 | 128 | нет | раздельные сигналы input и output |
| 5 | 9 | 64 | есть | общий тактовый сигнал |
| 6 | 10 | 64 | нет | раздельные сигналы input и output |
| 7 | 11 | 64 | есть | общий тактовый сигнал |
| 8 | 12 | 64 | нет | раздельные сигналы input и output |
| 9 | 13 | 32 | есть | общий тактовый сигнал |
| 10 | 14 | 32 | нет | раздельные сигналы input и output |
| 11 | 15 | 32 | есть | общий тактовый сигнал |
| 12 | 16 | 32 | нет | раздельные сигналы input и output |
| Примечание. В качестве элементной базы для реализации ОЗУ выбирать любую микросхему ПЛИС, принадлежащую только семейству Cyclone. Для вариантов задания №7 – №12 обязательно осуществить начальную инициализацию содержимого памяти, привести в отчёте содержимое файла инициализации. | | | | |

### 8.7.2 Контрольные вопросы

1 Дайте определение запоминающего устройства.

2 Какие виды запоминающих устройств Вы знаете?

3 Дайте определение постоянного запоминающего устройства. Приведите примеры их применения в технике.

4 Дайте определение оперативного запоминающего устройства. Приведите примеры их применения в технике.

5 Каким образом можно создать условное графическое обозначение компонента в САПР Quartus II?

6 Каким образом осуществляется синтез запоминающего устройства заданного типа при помощи утилиты MegaWizard Plug-in Manager?

# 9 САМОСТОЯТЕЛЬНАЯ РАБОТА

## 9.1 Цель и задачи дисциплины

Дисциплина "Большие интегральные схемы микропроцессоров и программируемой логики" входит в число дисциплин по выбору ВУЗа для образовательно-профессиональной программы высшего образования по направлению подготовки 050102 "Компьютерная инженерия".

Целью изложения дисциплины является получение теоретической базы, которая необходима при усвоении прикладных вопросов проектирования и решении конкретных задач исследования технических систем. Особенности проектируемых систем обусловливают разнообразие подходов к разработке различных микропроцессорных систем. Поэтому основной упор в курсе сделан на проектировании систем традиционной архитектуры фон Неймана, что соответствует требованиям квалификационной характеристики специалиста.

После изучения дисциплины студенты должны знать:

1) принципы работы современных БИС микропроцессоров (МП) и программируемой логики (ПЛ);
2) существующие средства построения соответствующих программных объектов;
3) современные методы тестирования процессорных систем;
4) основные тенденции развития современных БИС;
5) современную техническую базу: БИС микропроцессоров и программируемой логики;
6) архитектурные характеристики современных БИС и уметь применять их при проектировании электронных схем;
7) основы технологии и программно-технические средства разработки и отладки электронных схем на базе современных микропроцессоров и ПЛИС.

В результате освоения учебной дисциплиной студенты должны уметь:

1) разрабатывать структуры встраиваемых систем на основе микропроцессоров и программируемой логики;
2) программировать встраиваемые системы;
3) исследовать функционирование встраиваемых систем;
4) исследовать быстродействие встраиваемых систем;
5) применять современные принципы построения электронных схем с применением больших интегрированных схем микропроцессоров и программируемой логики.

Значение дисциплины для реализации требований квалификационной характеристики специалиста и изучения следующих дисциплин заключается в том, что дисциплина охватывает широкий круг вопросов, связанных с разработкой, программированием и исследованием микропроцессорных устройств и создает базу, необходимую для выполнения квалификационной работы.

Дисциплина является базовой для дисциплин "Архитектура компьютеров", "Метрология и измерительная техника", "Технологии

проектирования компьютерных систем", "Проектирование специализированных КС", "Проектирование встроенных КС".

9.2 Содержание дисциплины

9.2.1 Введение

Предмет, цель и задачи курса «БИС МП и ПЛ». Краткий обзор современных архитектур ЭВМ: транспьютер, нейрокомпьютер. Понятие интерфейса. Специализированные интегральные схемы (СПИС). Классификация СПИС.

9.2.2 Общая характеристика микропроцессорного комплекта К1810

Общие сведения о микропроцессоре К1810ВМ86. Назначение выводов микропроцессора. Структура микропроцессора: операционное устройство, управляющее устройство, устройство шинного интерфейса. Сегментация памяти и вычисления адресов, организация ввода-вывода. Общие сведения о системе прерываний. Процедура обслуживания прерываний. Функционирование процессора в минимальном режиме. Структурная схема ЦП на основе ВМ86. Система команд микропроцессора К1810ВМ86.

9.2.3 Общие сведения и технические характеристики специализированного процессора ввода-вывода К1810ВМ89 (СПВВ)

Структура СПВВ: устройство управления, арифметико-логическое устройство, внутренние регистры, блок шинного интерфейса. Использование СПВВ в местной и удаленной конфигурациях. Схема подключения СПВВ в местной конфигурации. Схема подключения СПВВ в удаленной конфигурации. Система команд СПВВ.

9.2.4 Периферийные устройства

Генератор тактовых импульсов К1810ГФ84 (ГТИ). Схема формирования тактовых импульсов. Схема формирования сигнала RESET. Схема формирования сигнала готовности READY. Буферные регистры и шинные формирователи (БР, ШФ). Функциональные схемы БР К580ИР82 и ШФ К580ВА86. Контроллер системной шины К1810ВГ88 (КСШ). Назначение выводов. Структура КСШ ВГ88. Функционирование КСШ: режим работы с системной шиной, режим работы с шиной ввода-вывода. Схема подключения КСШ в режиме системной шины. Программируемый таймер К1810ВИ54. Структурная схема таймера, назначение выводов. Режимы работы таймера (0,1,2,3,4,5). Программирование таймера.

9.2.5 Программируемый контроллер прерываний К1810ВМ59 (ПКП)

Программируемый контроллер прерываний К1810ВМ59 (ПКП). Описание функционирования. Регистры запросов прерываний и

обслуживаемые прерывания, блок приоритетов, регистр масок. Работа ПКП с ЦП 80186. Программирование контроллера: слова команд инициализации, слова операционных команд. Режимы работы ПКП: режим специального маскирования, режим каскадирования, режим запроса.

### 9.2.6 Структура микропрограммных устройств

Микропрограммный принцип управления операциями. Однокристальный микроконтроллер КР1816ВЕ48. Структурная схема и назначение выводов. Карты распределения внутренней памяти микроконтроллера ВЕ48. Режимы работы микроконтроллера с внешней памятью команд и данных. Режим выполнения программы по шагам. Организация системы прерываний микроконтроллера КР1816ВЕ48. Микроконтроллер КР1816ВЕ51. Структурная схема ВЕ51. Организация прерываний ВЕ51. Порты ввода-вывода микроконтроллера ВЕ51. Особый режим работы микроконтроллера КР1816ВЕ51.

### 9.2.7 Программируемые логические интегральные схемы

Понятие программируемых логических интегральных схем (ПЛИС). Преимущества применения специализированных интегральных схем. Основные области применения ПЛИС. Методы построения программируемых логических функций: методы плавких и наращиваемых перемычек. Методы построения программируемых логических функций на основе запоминающих устройств. Структура ПЛИС. Классификация специализированных интегральных схем. Классификация ПЛИС по структурному признаку. Классификация ПЛИС по набору критериев. Классификация ПЛИС по набору критериев: архитектура простейшего функционального преобразователя.

### 9.2.8 Структура программируемых логических интегральных схем

Структура программируемой логической матрицы (ПЛМ). Структура простейшего функционального преобразователя СБИС ПЛУ. Структура простейшего функционального преобразователя с архитектурой Look-up-table. Структуры одноуровневых и многоуровневых СБИС ПЛ.

### 9.2.9 ПЛИС фирмы Altera Corporation. Семейство FLEX10K

ПЛИС фирмы Altera Corporation. Общая характеристика семейств. Структура СБИС ПЛ семейства FLEX10K. Глобальная программируемая матрица соединений. Логический блок. Структура схемы коммутации. Логический элемент. Режим каскадного наращивания. Цепочки переноса. Схема управления установкой/сбросом триггера. Реконфигурируемый модуль памяти. Элементы ввода-вывода. Основные параметры СБИС семейства FLEX 10K.

9.2.10 Конфигурирование и реконфигурирование ПЛИС. Реализация алгоритмов ЦОС на основе специализированных БИС

Конфигурирование и реконфигурирование ПЛИС. Использование последовательных ПЗУ для конфигурирования СБИС ПЛ семейства FLEX10K. Аппаратные средства конфигурирования и реконфигурирования ПЛИС. Порт JTAG. Реализация алгоритмов ЦОС на основе специализированных БИС. Реализация алгоритмов ЦОС на основе цифровых сигнальных процессоров общего назначения. Реализация алгоритмов ЦОС на базе ПЛИС.

9.3 Распределение объема самостоятельной работы

Распределение объема самостоятельных работ указано в таблице 9.1.

Таблица 9.1 - Распределение объема самостоятельных работ по видам работ

| Вид работы | Объем, часов |
|---|---|
| Подготовка к лабораторным занятиям | 12 |
| Подготовка отчетов по лабораторным роботам | 8 |
| Подготовка к тестам | 14 |
| Выполнение расчетно-графической работы | 10 |
| Работа с конспектом лекций, методическими указаниями, основной и дополнительной литературой | 10 |
| Всего за курс | 54 |

9.4 Экзаменационные вопросы

1. Классификация специализированных интегральных схем. Преимущества применения специализированных интегральных схем.
2. Понятие программируемых логических интегральных схем.
3. Основные области применения ПЛИС.
4. Методы построения программируемых логических функций: методы плавких и наращиваемых перемычек.
5. Методы построения программируемых логических функций на основе запоминающих устройств.
6. Структура ПЛИС.
7. Классификация ПЛИС по структурному признаку.
8. Классификация ПЛИС по набору критериев.
9. Классификация ПЛИС по набору критериев: архитектура простейшего функционального преобразователя.
10. Структура программируемой логической матрицы (ПЛМ).
11. Структура простейшего функционального преобразователя СБИС ПЛУ.
12. Структура простейшего функционального преобразователя с архитектурой Look-up-table.
13. Структуры одноуровневых и многоуровневых СБИС ПЛ.
14. ПЛИС фирмы Altera Corporation. Общая характеристика семейств.
15. Структура СБИС ПЛ семейства FLEX10K.

16. Семейство FLEX10K: глобальная программируемая матрица соединений.
17. Семейство FLEX10K: логический блок.
18. Семейство FLEX10K: структура схемы коммутации.
19. Семейство FLEX10K: логический элемент.
20. Семейство FLEX10K: режим каскадного наращивания.
21. Семейство FLEX10K: цепочки переноса.
22. Семейство FLEX10K: схема управления установкой/сбросом триггера.
23. Семейство FLEX10K: реконфигурируемый модуль памяти.
24. Семейство FLEX10K: элементы ввода-вывода.
25. Основные параметры СБИС семейства FLEX 10K.
26. Конфигурирование и реконфигурирование ПЛИС.
27. Использование последовательных ПЗУ для конфигурирования СБИС ПЛ семейства FLEX10K.
28. Аппаратные средства конфигурирования и реконфигурирования ПЛИС.
29. Порт JTAG
30. Реализация алгоритмов ЦОС на основе специализированных БИС.
31. Реализация алгоритмов ЦОС на основе цифровых сигнальных процессоров общего назначения.
32. Реализация алгоритмов ЦОС на базе ПЛИС.
33. Понятие интерфейса. Классификация интерфейсов: параллельный и последовательный интерфейсы.
34. Общие сведения о микропроцессор К1810ВМ86. Назначение выводов микропроцессора.
35. Структура микропроцессора ВМ86: операционное устройство, управляющее устройство, устройство шинного интерфейса.
36. Сегментация памяти и вычисления адресов, организация ввода-вывода.
37. Общие сведения о системе прерываний. Процедура обслуживания прерываний.
38. Функционирование процессора ВМ86 в минимальном режиме.
39. Структурная схема ЦП на основе ВМ86.
40. Система команд микропроцессора К1810ВМ86.
41. Общие сведения и технические характеристики арифметического сопроцессора К1810ВМ87 (АСП).
42. Структура АСП: операционное устройство, устройство шинного интерфейса.
43. Функционирование АСП: пассивный и активный режимы.
44. Временная диаграмма работы АСП в пассивном и активном режимах.
45. Система команд АСП ВМ87.
46. Общие сведения и технические характеристики специализированного процессора ввода-вывода К1810ВМ89 (СПВВ).
47. Структура СПВВ: устройство управления, арифметико-логическое устройство, внутренние регистры, блок шинного интерфейса.
48. Использование СПВВ в местной и удаленной конфигурациях.
49. Схема подключения СПВВ в местной конфигурации.
50. Схема подключения СПВВ в удаленной конфигурации.

51. Система команд СПВВ.
52. Генератор тактовых импульсов К1810ГФ84 (ГТИ).
53. Схема формирования тактовых импульсов.
54. Схема формирования сигнала RESET.
55. Схема формирования сигнала готовности READY.
56. Буферные регистры и шинные формирователи (БР, ШФ). Функциональные схемы БР К580ИР82 и ШФ К580ВА86.
57. Контроллер системной шины К1810ВГ88 (КСШ). Назначение выводов.
58. Структура КСШ ВГ88.
59. Функционирование КСШ: режим работы с системной шиной, режим работы с шиной ввода-вывода.
60. Схема подключения КСШ в режиме системной шины.
61. Контроллер динамичной памяти К1810ВТ03 (КДП).
62. Организация памяти в микропроцессорной системе на базе комплекта К1810.
63. Структурная схема и назначение выводов КДП.
64. Функционирование КДП: цикл регенерации, цикл проверки, цикл чтения, цикл записи.
65. Режимы работы КДП: режим 16К, режим 64К. Применение КДП.
66. Контроллер прямого доступа к памяти К1810ВТ37 (КПДП). Структурная схема и назначение выводов.
67. Функционирование в режиме ПДП. Цикл ожидания, активный цикл, режим одиночного обмена, режим блочного обмена, режим запроса обмена, режим каскадирования.
68. Типы обмена КПДП: чтение, запись, проверка. Режимы память-память и автоинициализация.
69. Описание регистров КПДП. Программирования контроллера.
70. Программируемый контроллер прерываний К1810ВМ59 (ПКП). Описание функционирования.
71. Регистры запросов прерываний и обслуживаемых прерываний, блок приоритетов, регистр маскирования.
72. Работа ПКП с ЦБ 80186.
73. Программирование контроллера: слова команд инициализации, слова операционных команд.
74. Режимы работы ПКП: режим специального маскирования, режим каскадирования, режим запроса.
75. Программируемый таймер К1810ВИ54. Структурная схема таймера, назначение выводов.
76. Режимы работы таймера (0,1,2,3,4,5). Программирование таймера.
77. Контроллер накопителя на гибком магнитном диске К580ВГ72.
78. Структурная схема и назначение выводов контроллера К580ВГ72.
79. Программирование и функционирование контроллера.
80. Структура микропрограммных устройств.
81. Микропрограммный принцип управления операциями.

82. Однокристальный микроконтроллер КР1816ВЕ48. Структурная схема и назначение выводов.
83. Карты распределения внутренней памяти микроконтроллера ВЕ48.
84. Режимы работы микроконтроллера с внешней памятью команд и данных.
85. Режим выполнения программы по шагам.
86. Организация системы прерываний микроконтроллера КР1816ВЕ48.
87. Микроконтроллер КР1816ВЕ51. Структурная схема ВЕ51.
88. Организация прерываний ВЕ51.
89. Порты ввода-вывода микроконтроллера ВЕ51.
90. Особый режим работы микроконтроллера КР1816ВЕ51.

## РЕКОМЕНДОВАННАЯ ЛИТЕРАТУРА


1. Introduction to the Quartus II Software. – San Jose: Altera Corporation, 2009. – 229 с.
2. Quartus II Handbook Version 9.0. – San Jose: Altera Corporation, 2009. – 2490 с.
3. Антонов А.П., Мелехин В.Ф., Филиппов А.С. Обзор элементной базы фирмы ALTERA. - Санкт-Петербург: ЭФО, 1997. - 139 с.
4. Казеннов Г.Г. Основы проектирования интегральных схем и систем. – М.: БИНОМ. Лаборатория знаний, 2005. – 295с.
5. Комолов Д.А., Мяльк Р.А., Зобенко А.А., Филоппов А.С. Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II. Краткое описание самоучиель. – М.: ИП РадиоСофт, 2002. – 352с.: ил.
6. Соловьев В.В. Проектирование цифровых систем на основе программируемых логических интегральных схем. 2-е изд., стереотип. – М.: Горячая линия – Телеком, 2007. – 636 с.
7. Стешенко В.Б. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов. - М.: Додека, 2000. - 127 с.
8. Стешенко В.Б. ПЛИС фирмы Altera: элементная база, система проектирования и языки описания аппаратуры. – М.: Издательский дом «Додэка XXI», 2002. – 576 с.