КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ФИЗИКИ
Кафедра радиоастрономии

ОСНОВЫ РАБОТЫ В СРЕДЕ QUARTUS II

Учебно-методическое пособие

Казань - 2017

УДК 621.396.075

Принято на заседании кафедры радиоастрономии КФУ
Протокол № 9 от 31 мая 2017 года

Рецензент:
доцент кафедры радиоэлектроники КФУ
кандидат физико-математических наук **И.А. Насыров**

Составители: **Акчурин А.Д., Юсупов К.М., Колчев А.А.**

ОСНОВЫ РАБОТЫ В СРЕДЕ QUARTUS II. – Казань: КФУ, 2017. – 49 с.

Пособие представляет собой введение в систему автоматизированного проектирования (САПР) Quartus II и дает общее представление о типичных этапах проектирования программируемых логических интегральных схем (ПЛИС), от идеи до готового изделия. Описаны два подхода к проектированию схем в ПЛИС: первый – графический ввод, когда пользователь рисует схемы цепей, используя готовые шаблоны, второй – кодовое описанием логических схем. Показаны способы передачи файлов конфигурации, содержащих разработанную логическую схему, с персонального компьютера на микросхему ПЛИС на примере отладочного комплекта DE2. Данное пособие предназначено для методической поддержки дисциплин «Лаборатория программируемой логики», «Лаборатория интеллектуальных датчиков» и «Лаборатория автоматизации систем научных измерений», изучаемых в бакалавриате радиофизического направления Института физики КФУ.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программируемая логическая интегральная схема (ПЛИС, ***p****rogrammable* ***l****ogic* ***d****evice, PLD*) — электронный компонент, используемый для создания цифровых интегральных схем. Функции ПЛИС, в отличие от обычных микросхем, задаются посредством программирования (проектирования), а не определяются при производстве. Для задания требуемой структуры цифрового устройства используются программатор и отладочная среда (IDE). В результате формируется принципиальная электрическая схема или программа на специальных языках описания аппаратуры: Verilog, AHDL, VHDL и др.

В настоящий момент существует множество производителей микросхем ПЛИС, и каждый из них имеет свою систему автоматизированного проектирования (САПР), но исторически сложилось так, что мы работаем с ПЛИС и САПР (Quartus II) фирмы Altera. САПР Quartus II является многофункциональной средой проектирования, которая содержит в себе наборы утилит, позволяющие разработать, верифицировать и запрограммировать проект для выбранного семейства микросхем ПЛИС.

Логическая схема, разработанная в программном обеспечении Quartus II, называется *проектом*. Проект может иметь *иерархическую структуру* или состоять из множества файлов-модулей, когда главный модуль содержит несколько дополнительных модулей, а каждый дополнительный модуль включает еще несколько файлов. В этом случае главный модуль называется *объектом верхнего уровня иерархии*. САПР Quartus II, помимо иерархической структуры, сохраняет в одном каталоге всю техническую информацию, касающейся выбранной микросхемы (например: назначение выводов, предыдущие варианты трассировки логической схемы, конфигурация проекта и т.д.). Эти файлы создаются в единой папке проекта, что удобно при копировании и переносе проектов в другие каталоги или на другие компьютеры.

В данном пособии рассмотрены действия по реализации проекта на примере одного файла без иерархически подчиненных файлов нижнего уровня (сложные проекты с иерархической структурой рассматриваться не будут). Рассмотрены общие сведения о ПЛИС, а также полная последовательность действий при разработке проекта в среде Quartus II, от создания логической схемы до ее ввода в конкретную микросхему (*конфигурация* ПЛИС). Для закрепления рассмотренного материала приведены контрольные задания, рекомендуемые для самостоятельного выполнения.

ОСОБЕННОСТИ АРХИТЕКТУРЫ ПЛИС

ПЛИС применяется для создания некоторой заданной схемы цифровых интегральных схем. Структура работы ПЛИС определяется путем программирования с помощью программатора и программного обеспечения.

В большинстве случаев микросхема ПЛИС состоит из набора перестраиваемых логических блоков, реализующих необходимую логическую функцию; программируемых переключателей, осуществляющих требуемое соединение логических ячеек; программируемых блоков ввода/вывода, обеспечивающих связь внешнего вывода микросхемы с внутренней структурой ПЛИС. В современных ПЛИС часто бывают встроены дополнительно блоки памяти, блоки DSP (Digital Signal Processor, цифровой сигнальный процессор) или умножители, PLL (Phase Locked Loop, фазовая автоподстройка частоты) и другие компоненты.

Если разрабатывается некоторый проект, то его разработчик может, как правило, не учитывать особенности элементов, из которых состоит ПЛИС. При этом он описывает желаемые функции проектируемой микросхемы или записывает текст в системе программного обеспечения (например, на языках описания аппаратуры Verilog или VHDL). Требуемую схему конструирует компилятор на основе известной внутренней структуры ПЛИС, который сам размещает элементы схемы по имеющимся конфигурируемым логическим блокам и соединяет эти блоки с помощью имеющихся программируемых линий связи.

Классификация ПЛИС по типу хранения конфигурации

Для физической реализации соединений между узлами ПЛИС фирмами – изготовителями используются однократно программируемые и перепрограммируемые устройства.

В первом случае в качестве токопроводящих связей могут использоваться как полевые транзисторы ЛИЗМОП, так и перемычки типа antifuse.

При использовании ЛИЗМОП транзисторов перемычка создается за счет перевода полевого транзистора в проводящее состояние, что осуществляется путем подачи программирующего импульса. Подействовав на такой транзистор ультрафиолетовым излучением, его можно вернуть в исходное, непроводящее состояние. Однако в однократно программируемых устройствах такой структуры кристаллы защищают от попадания света, и программирование в этом случае возможно только один раз.

При реализации технологии antifuse программирование осуществляется путем расплавления в нужных местах чипа специальных перемычек для образования нужной схемы. Устранить эту перемычку нельзя. Достоинством

таких ПЛИС является то, что они работают достаточно быстро (могут работать на больших частотах), меньше подвержены сбоям при радиации.

Для создания многократно программируемых токопроводящих перемычек в настоящее время используются как двухзатворные полевые транзисторы МНОП-типа, изготавливаемые по FLASH технологии, так и обычные.

В микросхемах типа flash-based хранение конфигурации происходит во внутренней FLASH памяти или памяти типа EEPROM. Перемычка создается за счет перевода полевого транзистора в проводящее состояние, что осуществляется путем подачи программирующего импульса. Такой подход позволяет многократно менять конфигурацию разрабатываемого устройства, сохраняя ее при выключенном питании. Однако у этого типа ПЛИС есть и недостатки. При реализация FLASH памяти внутри CMOS микросхемы требуется совместить два разных техпроцесса - это приводит к удорожанию производства микросхем. Кроме того, такие микросхемы, как правило, имеют ограниченное количество циклов перезаписи конфигурации.

В микросхемах с триггерной памятью для хранения конфигурации (SRAM-based) в качестве перемычки выступает обычный полевой транзистор, затвор которого подключается к триггеру. Высокий уровень сигнала на его выходе переводит транзистор в проводящее состояние, а низкий сохраняет запертое. Триггер устанавливается в требуемое состояние в ходе загрузки программы конфигурации и сохраняет его в течение времени, пока включено питание ПЛИС. Такое оперативное программирование может производиться неограниченное число раз. Однако в этом случае при включении питания необходимо запускать процесс инициализации конфигурирования, на что требуется определенное время. При этом на плате должен еще стоять загрузчик, специальная микросхема FLASH или микроконтроллер – все это приводит к увеличению стоимости изделия.

Конфигурируемые логические блоки

В документации компании Альтера для обозначения конфигурируемых логических блоков используется выражение **Logic Array Block (LAB)** – массив логики. У компании Xilinx в микросхемах ПЛИС есть аналогичные блоки - **Configurable Logic Block (CLB)**. Конфигурируемый логический блок – это базовый элемент в ПЛИС, в нем может быть выполнена какая-то простая логическая функция или реализовано хранение результата вычисления в регистрах (триггерах). ПЛИС состоит из прямоугольной матрицы конфигурируемых логических блоков (CLB), окруженных блоками ввода - вывода (Input/ Output Block, IOB). Между CLB располагаются программируемые трассировочные линии. Отдельные CLB не имеют индивидуальных выходов, соединенных с внешними выводами ПЛИС. Вместо этого функции преобразования сигналов выполняют выделенные

ресурсы - IOB. Между матрицей CLB и блоками ввода-вывода имеются отдельные межсоединения, которые и обеспечивают подключение внешних сигналов.

Сложность и структура CLB определяется производителем. CLB может быть простым, как отдельный транзистор, а может быть очень сложным, как процессор (это крайние точки реализации).

В первом случае требуется большое число программируемых связей, чтобы потом из отдельных транзисторов собрать требуемую схему. Во втором случае связей необходимо не так много, но теряется гибкость проектирования пользовательской схемы.

На практике конфигурируемый блок обычно достаточно сложен, чтобы можно было реализовать некоторую функцию, но и достаточно мал, чтобы поместить множество таких блоков внутри ПЛИС, обеспечивая гибкость проектирования.

Таким образом, выбор структуры конфигурируемого логического блока производителем ПЛИС – это всегда поиск компромисса по площади кристалла, по быстродействию, энергопотреблению и т. п.

Конфигурируемый логический блок может состоять из одного или нескольких базовых логических элементов. В англоязычной литературе это Basic Logic Element (BLE) или просто Logic Element (LE). В ПЛИС обычно используются так называемые LUT-based базовые логические элементы. Аббревиатура LUT расшифровывается как **Look-Up Table** или просто Lookup Table, что дословно можно перевести как "справочная таблица" или "таблица поиска". LUT – это метод реализации функции, в котором непосредственное вычисление заменяется поиском по таблице готовых решений. Применительно к ПЛИС это позволяет реализовать любую логическую функцию в виде памяти SRAM, где адрес – это аргумент, а содержимое ячейки – значение. Требуемая таблица истинности хранится в виде маски (LUT-mask) в соответствущей ячейке CRAM. С помощью мультиплексоров выбирается нужное значение. Мультиплексорами управляют сигналы входных портов. Для построения k-входовой LUT (k-LUT), которая реализует любую логическую функцию из k переменных, требуется 2^k бит SRAM и 2^{k-1} мультиплексоров. При таком подходе можно достаточно точно спрогнозировать время прохождения сигнала, и оно не будет зависеть от реализуемой логической функции. Эта важная особенность делает возможным временно́й анализ схемы.

Например, на рис.1 показан четырехбитный LUT в составе базового логического блока. Здесь четырехбитному числу на входе логической функции ставится в соответствие однобитный результат. Квадратики на рис. 1 обозначают программируемый элемент (SRAM) или регистр, где хранится прошивка для ПЛИС. Из рис. 1 видно, что для конфигурации 4-х битного LUT требуется 16 конфигурационных регистров. Содержимое этих

регистров определяет логическую функцию, реализованную внутри базового логического элемента.

Еще один конфигурационный регистр (на рис. 1 это одиночный квадратик справа), который определяет, нужно ли на выход базового логического элемента выдавать прямо значение с LUT или нужно выдать значение с LUT, зафиксированное в D-триггере. Фиксация и хранение данных в цифровых схемах нужна практически в любом проекте.

Из рассмотрения схемы на рис. 1 (как примера обычного базового логического элемента) видно, какая избыточность заложена внутри современного кристалла ПЛИС (SRAM-based). Ведь конфигурационные регистры (квадратики) прямо не доступны для использования в цифровом проекте. Они только служат для формирования пользовательской функции. Для одного D-триггера в пользовательском проекте требуется более 16 (иногда много больше) триггеров для хранения конфигурации ПЛИС.

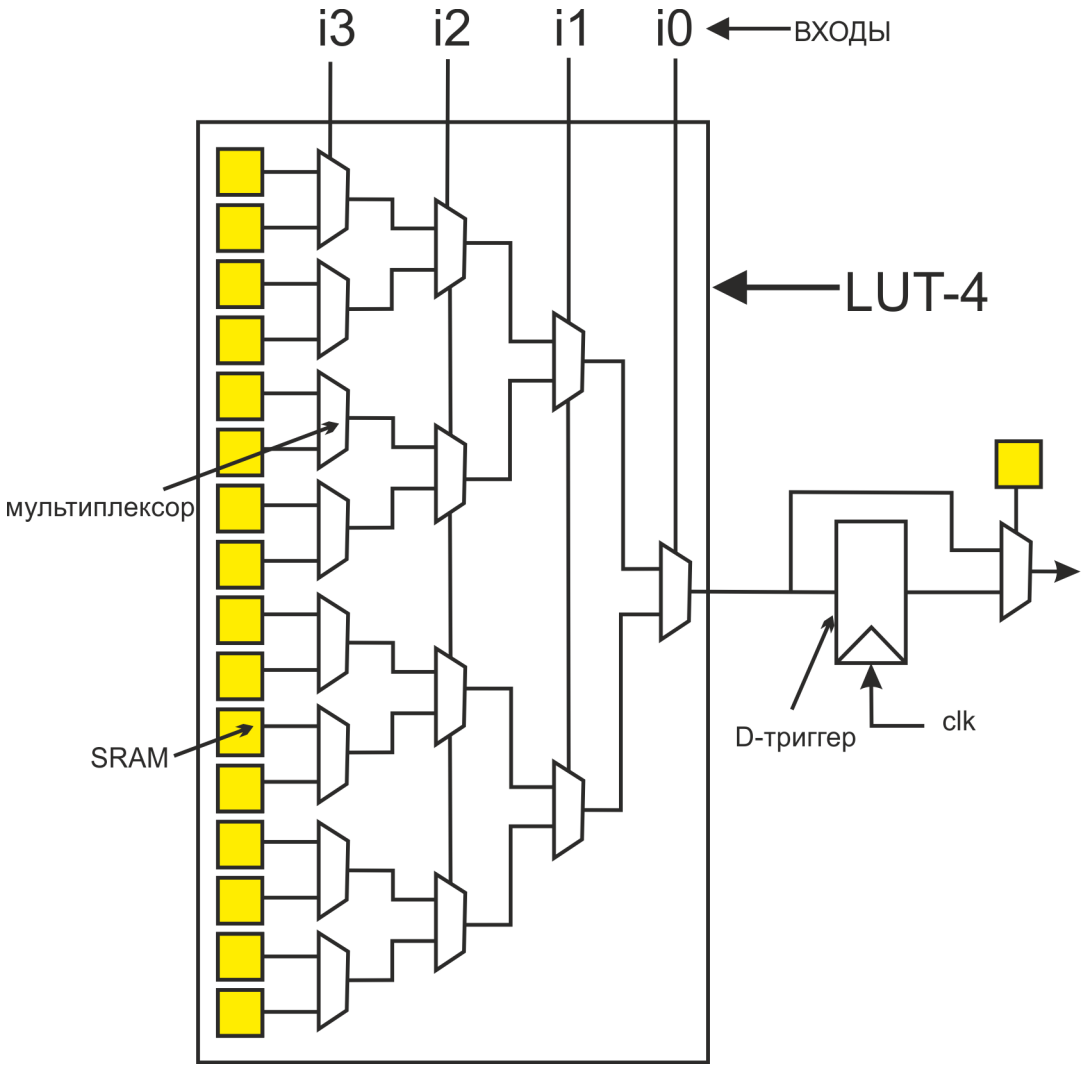


Рис. 1. Пример традиционного базового логического элемента

Иногда базовый логический элемент в других ПЛИС оказывается гораздо сложнее, чем показано на рис. 1. Ниже приведены некоторые примеры из документации на разные типы ПЛИС. На рис. 2 показан базовый логический элемент микросхемы CPLD MAX II компании Альтера. Здесь хорошо видны LUT и D-триггер хранения результата.

Ниже, на рис. 3, представлен базовый элемент микросхемы Cyclone III.

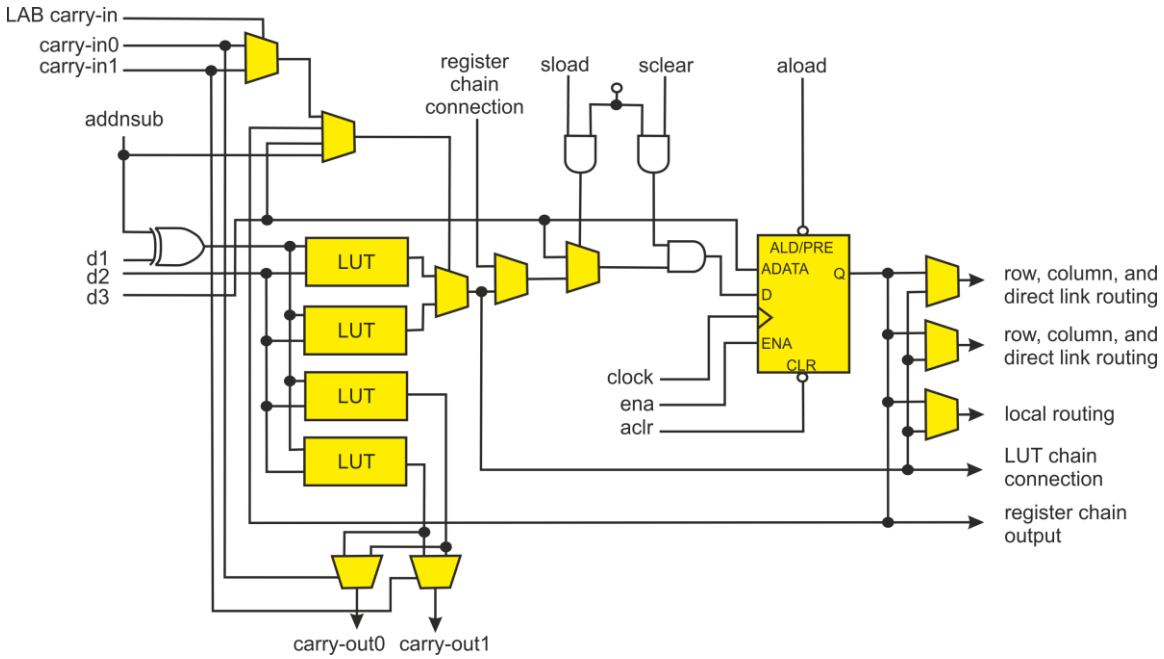


Рис. 2. Базовый логический элемент CPLD MAX II компании Альтера

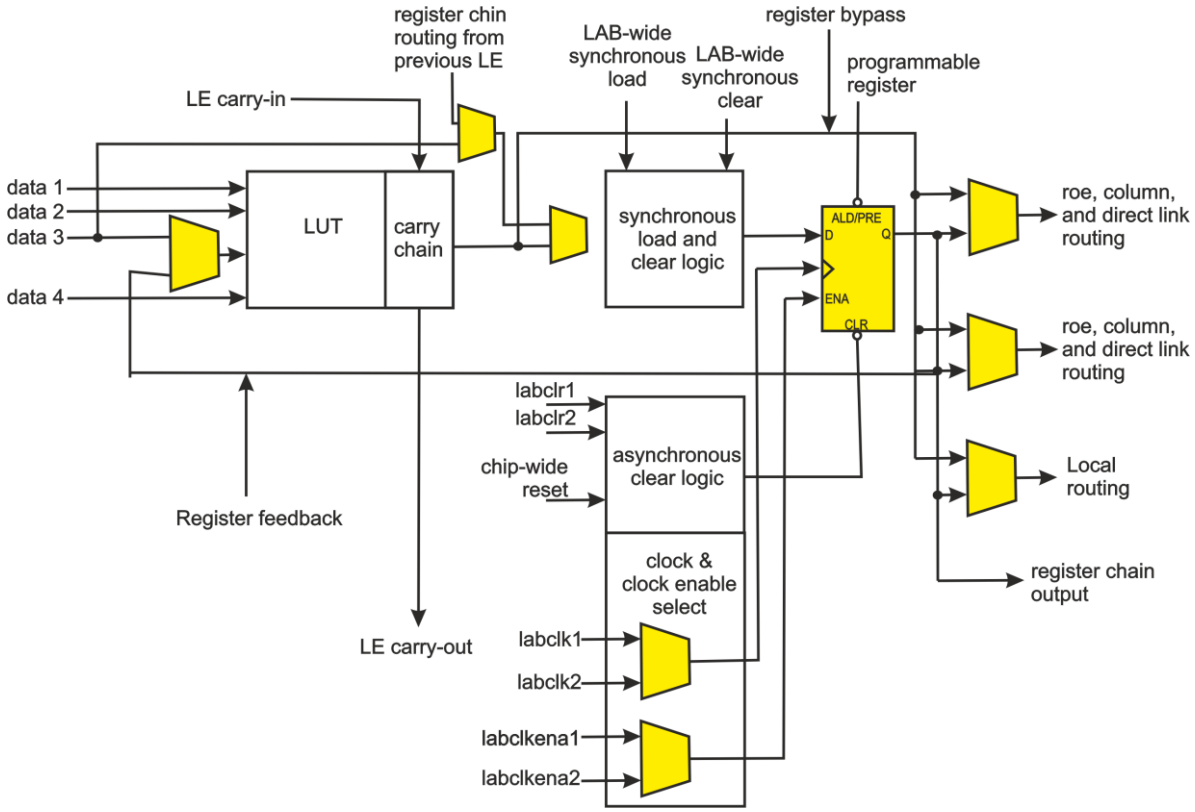


Рис. 3. Базовый логический элемент FPGA Cyclone III компании Альтера

В микросхемах компании Альтера в одном LAB может содержаться 10-16 LE.

В микросхемах компании Xilinx Virtex-6 базовый логический элемент – это так называемый Slice. В одном CLB всего два Slice. Зато один Slice – это довольно сложное устройство (рис.4): в одном CLB Virtex-6 имеется 8 LUT и 16 D-триггеров и еще кое-что плюс к этому.

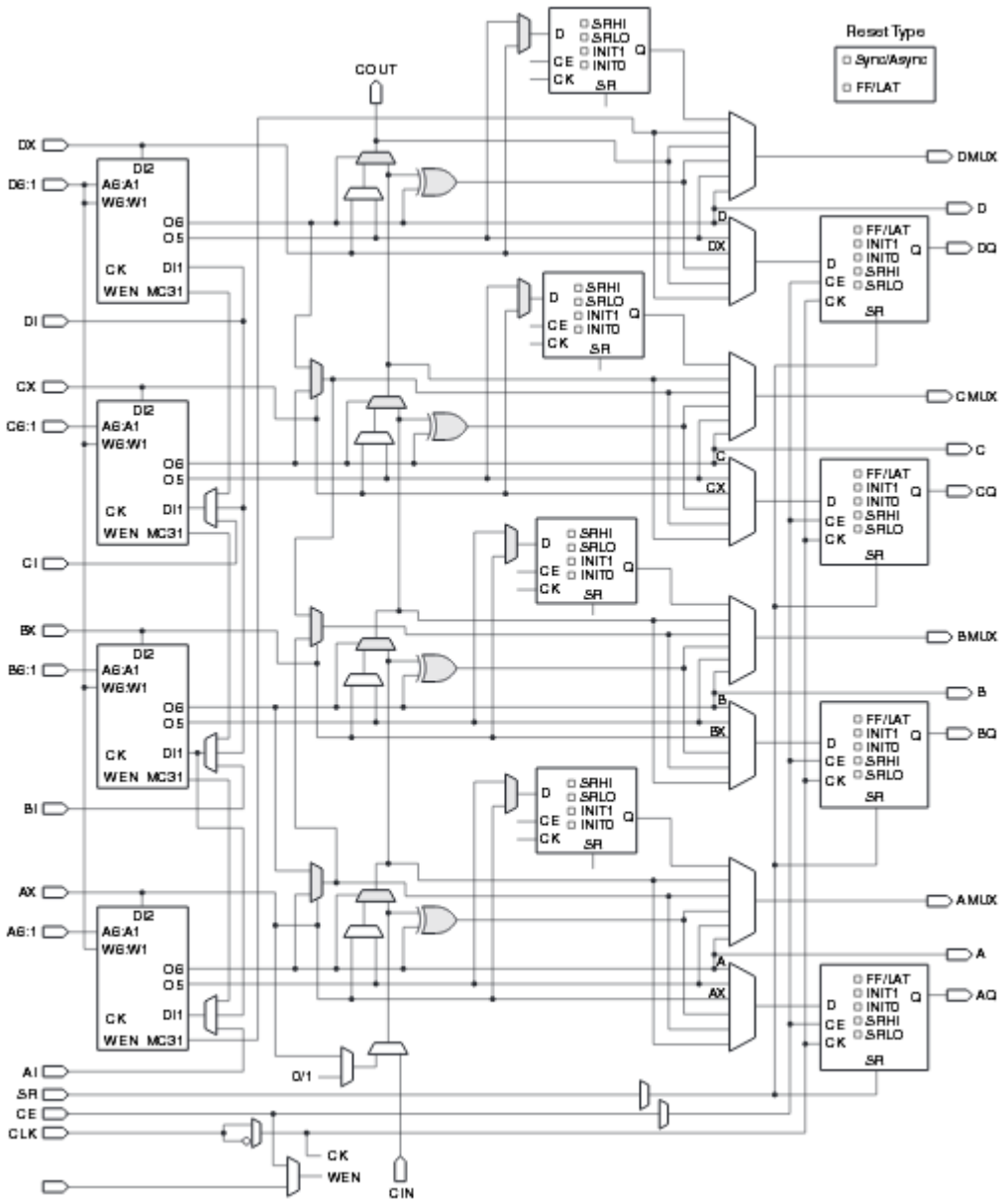


Рис. 4. Базовый элемент Xilinx Virtex-6 Slice

Примером более простой структуры базового логического элемента могут служить микросхемы FPGA компании Microsemi (бывшая Actel). На рис.5 представлен базовый логический элемент в микросхемах серии 40MX.

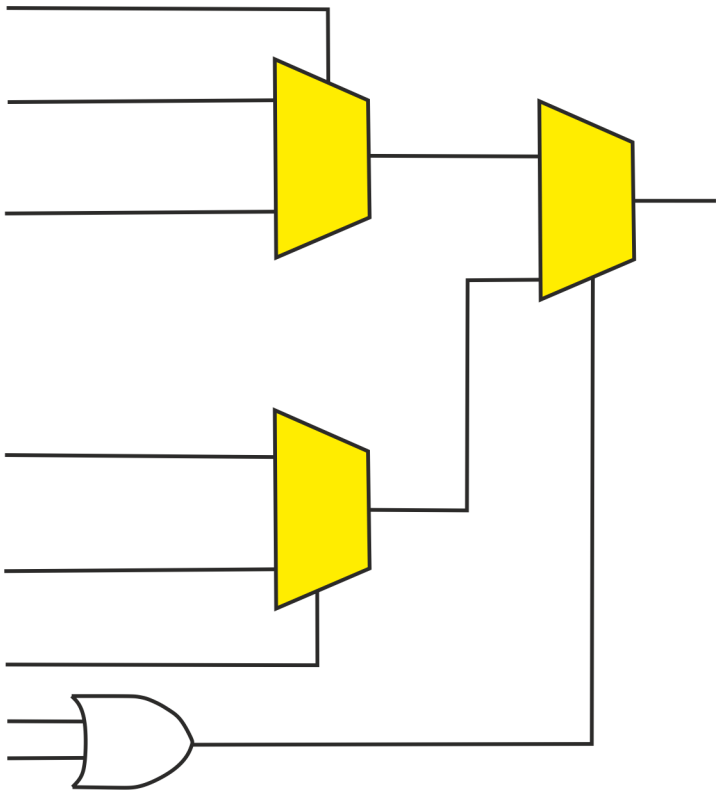


Рис. 5. Логический модуль микросхемы Microsemi 40MX

Логический модуль имеет восемь входов и один выход. Здесь не используется ни Look-Up Table, ни D-триггеры. Триггеры, как и остальная логика, формируются из Logic Module.

В микросхемах Microsemi связь между базовыми блоками обходится гораздо дешевле: серия 40MX является однократно программируемой. В ней межблочные связи «проплавляются» между соединяющими дорожками и позже не могут быть изменены. Нет регистров для временного хранения прошивки. В этих микросхемах нет программируемых переключателей, мультиплексоров, как в FPGA других типов. Здесь реализована технология antifuse – для производства таких микросхем используется модифицированный техпроцесс CMOS с дополнительными слоями для организации межблочных связей.

Программируемые связи между логическими блоками

Дополнительно к конфигурации логических блоков в ПЛИС необходимо программирование соединений между этими блоками. С этой целью в микросхемах FPGA используются отдельные конфигурационные коммутаторы (FPGA Routing Architecture и Programmable Routing Interconnect). В основном, такие соединения разделяют архитектуру ПЛИС на два типа: островная и иерархическая.

В ПЛИС островного типа блоки конфигурации одинаковы, а их совместное расположение с линиями связи и коммутационными узлами образуют матрицу (см. рис. 6).

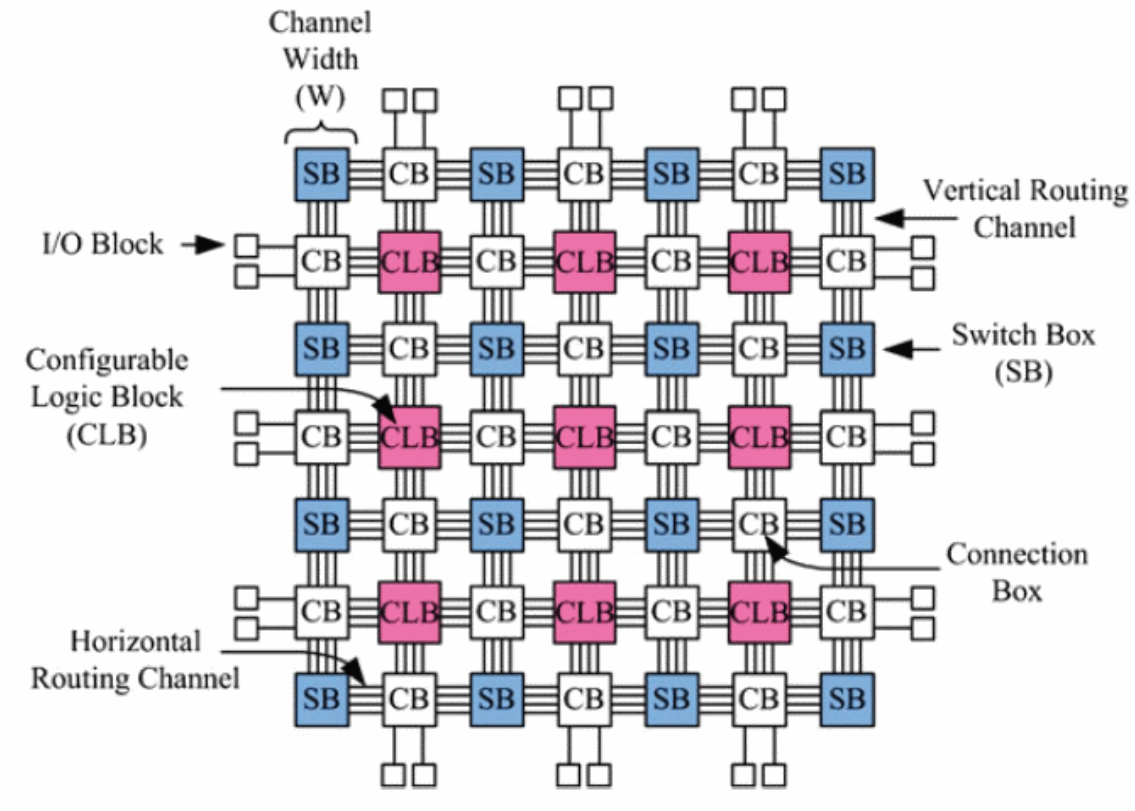


Рис. 6. Островная ПЛИС. CB – Connection Box, SB – Switch Box

Блоки на рис. 6 являются программируемыми мультиплексорами, выполняющими коммутацию между CLB посредством шин в микросхемах FPGA. В качестве примера таких микросхем могут служить семейства микросхем Altera: Cyclone и Stratix.

ПЛИС иерархического типа (см. рис. 7) рассчитаны на то, что отдельные блоки коммутируются друг с другом внутри ячейки, тем самым повышается скорость связей. В сложных проектах связь осуществляется путем коммутации соседних ячеек, т.е. сигнал сначала выходит из текущей ячейки и далее заходит в близлежащую, тем самым осуществляется иерархическая структура. ПЛИС с такой структурой имеют микросхемы Altera семейств: Flex10K, APEX.

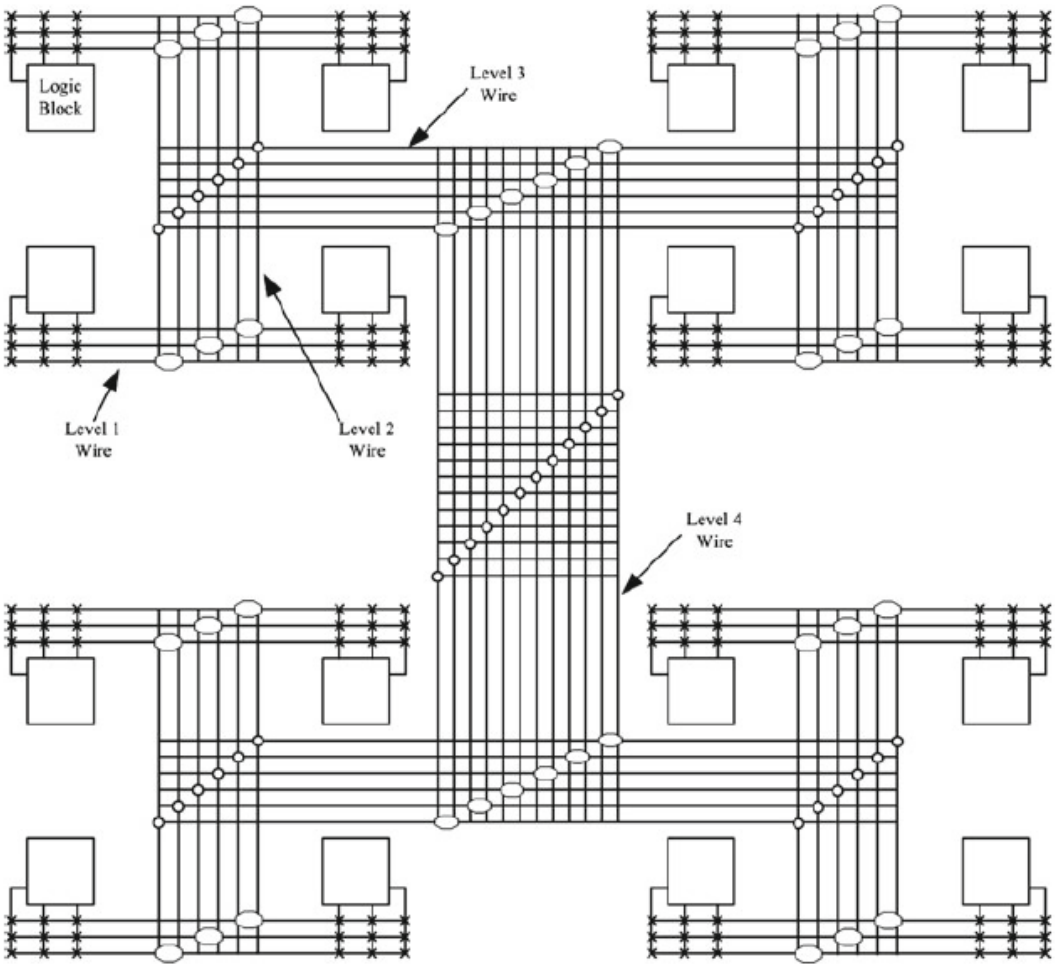


Рис. 7. Иерархическая ПЛИС

САПР для проектирования ПЛИС

САПР для проектирования ПЛИС, а именно компилятор (синтезатор логики и фиттер и ассемблер) – это, возможно, самая сложная часть всей ПЛИС технологии. Компилятор должен проанализировать пользовательский проект (схемы и текстовые описания на Verilog HDL или VHDL) и сгенерировать нетлист (netlist) – список всех элементов схемы и связи между ними. Netlist должен быть оптимизирован – логические функции нужно минимизировать, возможные дублированные регистры нужно удалить.

Затем компилятор должен вместить всю логику из netlist в имеющуюся архитектуру ПЛИС. Это делает фиттер (fitter). Он размещает логические элементы и выполняет трассировку связей между ними (процесс place and route). Сложность состоит в том, что один и тот же проект может быть размещен в ПЛИС разными способами, и этих способов миллионы. Некоторое размещение и трассировка оказываются лучше, другие хуже. Главный критерий качества полученной системы – максимальная частота, на которой сможет работать проект при данном размещении элементов и при данной трассировке связей. Здесь оказывают влияние длина связей между

логическими блоками и количество программируемых коммутаторов между ними.

Компилятор, зная архитектуру ПЛИС, по результатам работы дополнительно выдает отчет о времени прохождения сигналов от регистра до регистра. Эта информация часто бывает полезной для разработчика высокопроизводительных систем. Разработчик для ПЛИС имеет возможность давать некоторые советы компилятору: где, в каком месте кристалла лучше разместить тот или иной модуль проекта.

Выбирая для своего проекта конкретную модель микросхемы ПЛИС, разработчик в некоторой мере попадает в зависимость от производителя этой ПЛИС, так как должен в работе пользоваться программным обеспечением от этого же производителя.

Программное обеспечение компании Альтера: Quartus II. ПО Xilinx для проектирования для ПЛИС: ISE Suite, Vivaldo Design Suite. ПО компании Microsemi: Libero IDE, Libero SoC.

Программное обеспечение, компиляторы для ПЛИС – это одна из важнейших составляющих интеллектуальной собственности компаний-производителей ПЛИС.

В нашей работе мы рассматриваем ПЛИС Cyclone II компании Altera (микросхема EP2C35F672C6), а проектирование осуществляется в САПР Quartus II.

ОСНОВНЫЕ ЭТАПЫ ПРОЕКТИРОВАНИЯ В САПР QUARTUS II

Quartus II позволяет легко реализовать требуемую логическую схему в ПЛИС. Основные этапы проектирования в данной среде разработки показаны на рис. 8.

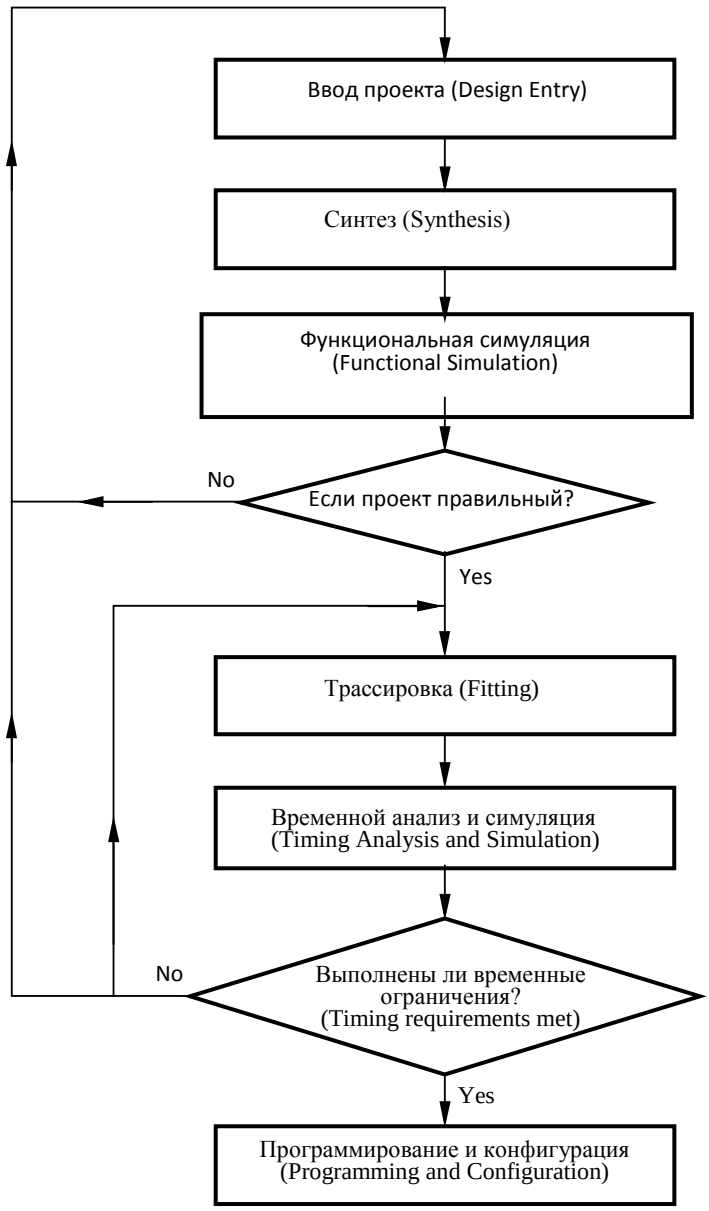


Рис. 8. Этапы проектирования ПЛИС в САПР Quartus II

Проектирование в САПР Quartus II включают следующие этапы:

• **Ввод проекта** (**Design Entry**) –желаемая схема проекта задается либо графическим способом, либо с использованием языков описания аппаратных средств, таких как Verilog HDL ,VHDL и др.

• **Синтез** (**Synthesis**) – вводимый проект синтезируется в схему, которая состоит из логических элементов (ЛЭ - LE) и логических блоков (ЛБ) в микросхеме ПЛИС.

• **Функциональное моделирование** (**Functional Simulation**) – синтезируемая схема тестируется на предмет корректности функционирования во встроенном симуляторе, который моделирует зависимость состояния (выбранных разработчиком) сигналов схемы от времени. Это моделирование не учитывает временные задержки сигналов (называемые *логическими гонками*) между ЛЭ/ЛБ микросхемы ПЛИС.

• **Трассировка** (**Fitting**) – инструмент трассировки САПР вычисляет оптимальное размещение и соединение ЛЭ/ЛБ, определенных в списке соединений (netlist) реальной микросхемы ПЛИС. Трассировщик также выбирает провода или «маршрут движения» в чипе для реализации необходимых связей между заданными ЛЭ/ЛБ.

• **Временной анализ** (**Timing Analysis**) – анализирует задержки распространения сигналов вдоль различных путей в трассируемой логической схеме для вычисления наличия/отсутствия логических гонок, чтобы сигналы с различных ЛБ приходили одновременно в тот или иной конечный узел схемы

• **Временное моделирование** (**Timing Simulation**) – схема тестируется для проверок функциональности и временных ограничений, но в отличие от функционального моделирования, здесь для обнаружения наличия или отсутствия логических гонок учитываются реальные задержки выбранных сигналов.

• **Программирование и конфигурация** (**Programming and Configuration**) - разработанная схема размещается в микросхеме ПЛИС путем программирования электронных связей между конфигурируемыми ЛЭ/ЛБ, что реализуется путем передачи конфигурационного файла с компьютера либо в микросхему ПЛИС, либо в дополнительную (не встроенную в ПЛИС) память, если такая имеется в отладочном комплекте.

Далее применение каждого этапа будет рассмотрено отдельно на примере проектирования простой логической схемы. Для этого необходимо выполнить следующую последовательность действий:

• Создание проекта
• Ввод логической схемы
• Синтез цепей в схеме
• Назначение входов и выходов схемы на выбранной микросхеме ПЛИС
• Моделирование разработанной логической схемы
• Программирование и настройка микросхемы ПЛИС на плате DE2

Начало работы в среде Quartus II

Первым шагом при реализации нового проекта логической схемы является создание директории для хранения файлов проекта. Запустите программу Quartus II, появится окно, похожее на рис. 9, которое состоит из нескольких окон, обеспечивающих доступ ко всем функциям программного обеспечения Quartus II. Большинство команд могут быть доступны с помощью набора меню, которое находится на панели инструментов. Выбрав кнопку с именем «File», открывается диалоговое окно, показанное на рис. 10.

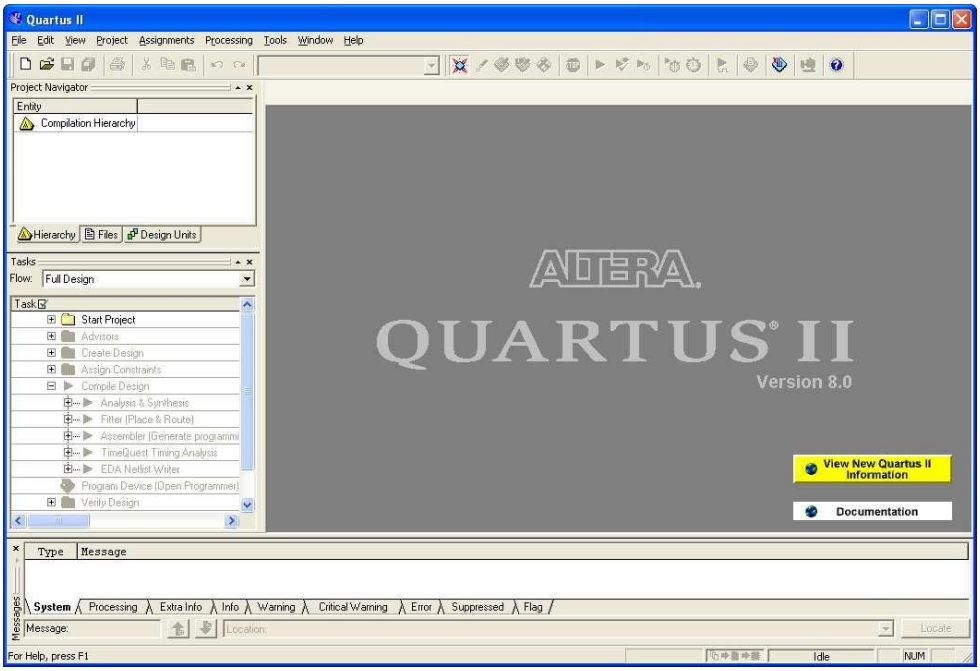


Рис. 9. Основное окно Quartus II

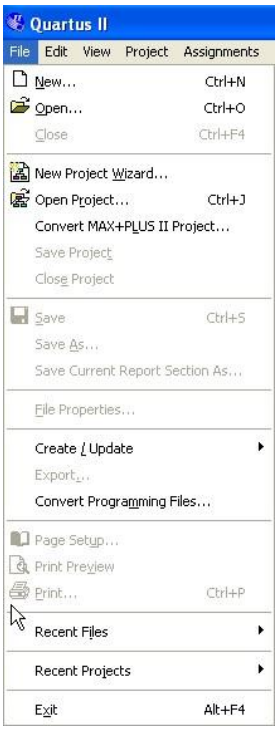


Рис. 10. Пример меню File

Создание нового проекта

Для начала работы необходимо создать новый проект:

1. В меню «File» выберите «New Project Wizard», появится окно, схожее с рис. 11, которое запрашивает имя и директорию проекта.

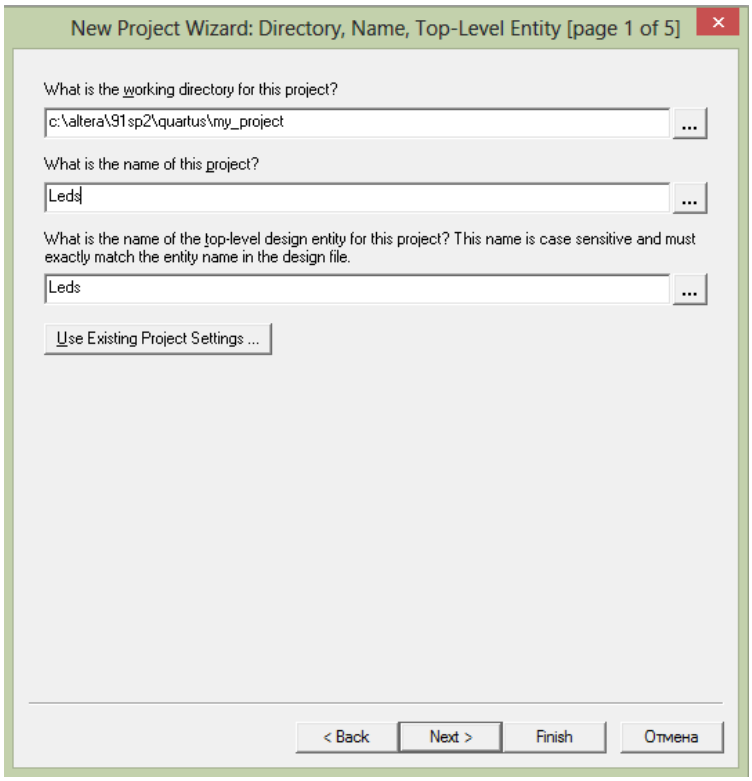


Рис. 11. Создание нового проекта

2. Выберите рабочий каталог, например: **my_project**. Удобно, когда название проекта имеет имя, совпадающее с объектом проекта верхнего уровня. В строках «What is the name of this project» и «What is the name of the top-level design entity for this project» введите «Leds» в качестве имени как самого проекта, так и имени объекта верхнего уровня (см. рис. 11). После нажатия кнопки «Next» появится всплывающее диалоговое окно (рис. 12), предлагающее создать необходимую папку. После нажатия кнопки «Да» появится окно, приведенное на рис. 13.

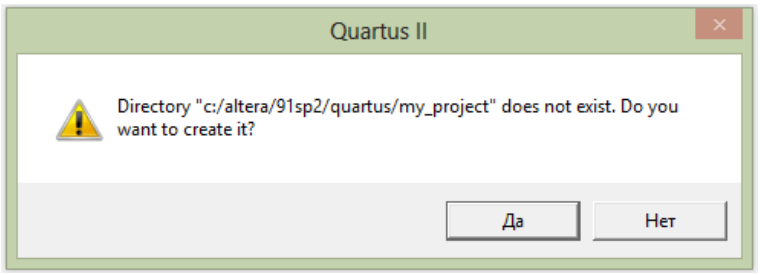


Рис. 12. Диалоговое окно создания нового каталога

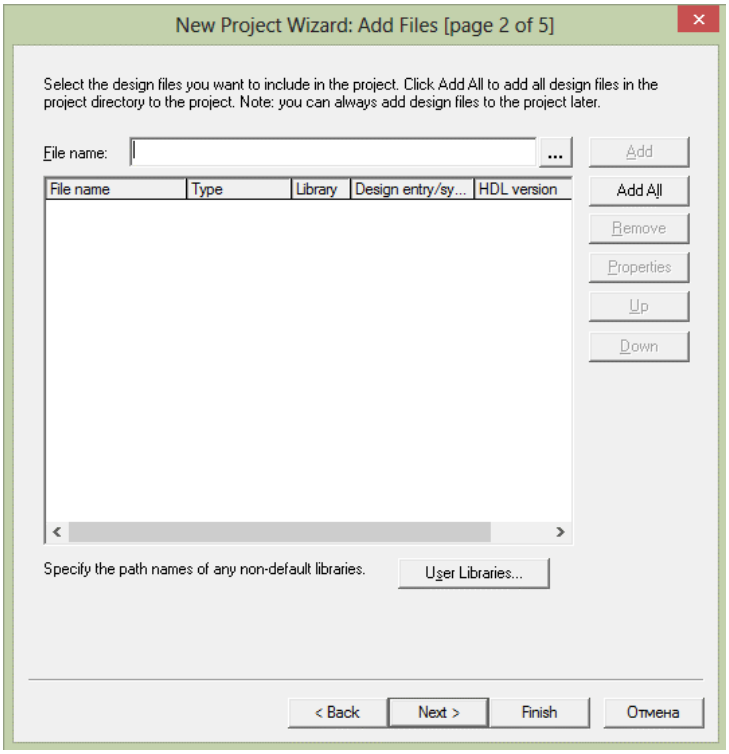


Рис. 13. Диалоговое окно подключения существующих файлов в проект

3. Если разработчик имеет опыт работы с ПЛИС и ранее создал какие-либо проектные файлы, то окно на рис. 13 позволяет легко указать и подключить в проект такие файлы. Если ранее созданных файлов проекта нет, нажмите «Next».

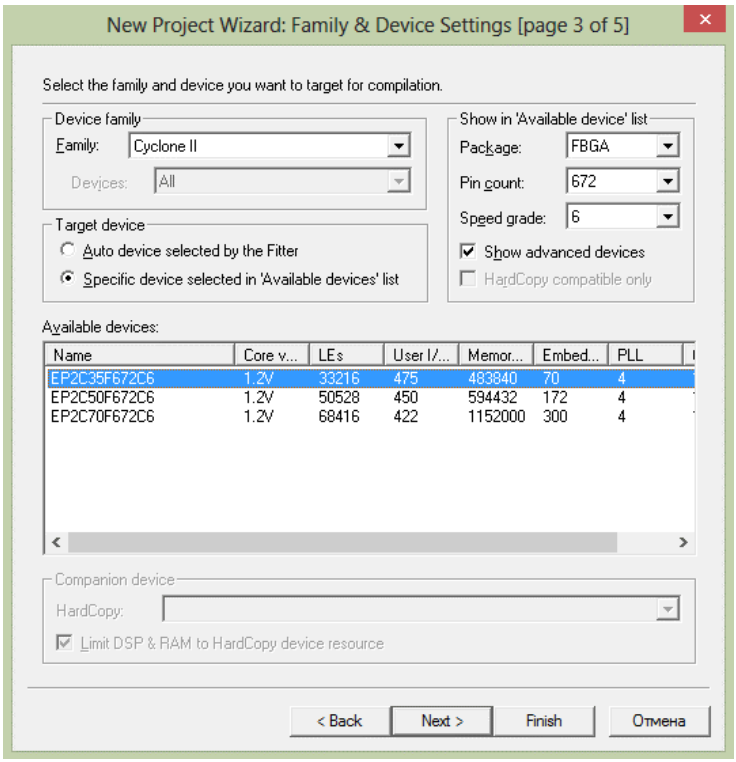


Рис. 14. Выбор микросхемы ПЛИС

4. В следующем диалоговом окне (рис.14) необходимо определить тип микросхемы, в которой будет реализован разработанный проект. В отладочном комплекте DE2 применена микросхема плис **EP2C35F672C6**, для ее выбора в разделе «Device family» выберите «Cyclone II». Для ускорения поиска выберите параметры необходимой микросхемы (Package – тип корпуса, Pin_count – количество выводов, Speed grade – оценка скорости), как это показано на рис. 14. Выберите микросхему **EP2C35F672C6**. После нажатия «Next» открывается окно, изображенное на рис. 15.

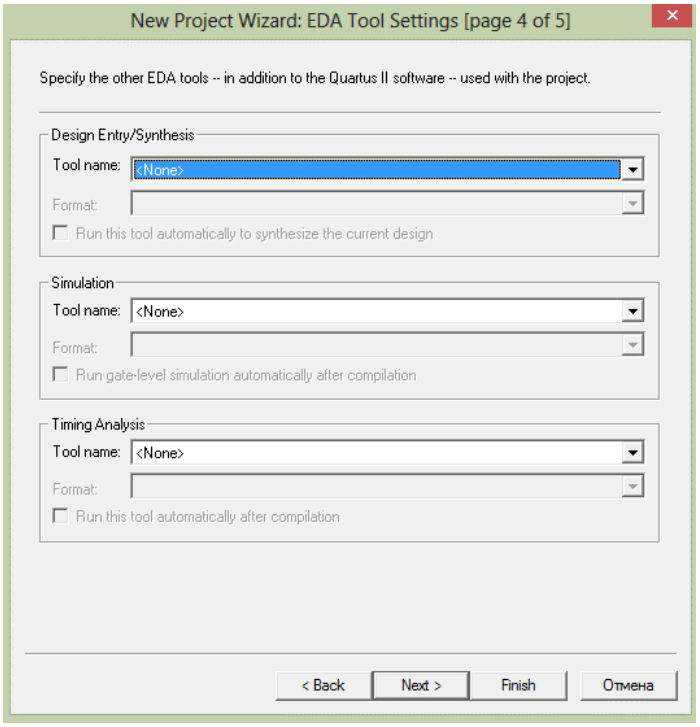


Рис. 15. Окно выбора инструментов моделирования сторонних производителей

5. В окне «EDA Tool Settings» (EDA – Electronic Design Automation, автоматизация проектирования электроники) на рис. 15 пользователь может указать любые утилиты моделирования сторонних производителей (например, ModelSim). В рамках данного методического пособия сторонние средства моделирования не будут использоваться, а будет применяться только встроенное в Quartus II 9 средство моделирования. Нажмите Next.

6. Краткая информация о выбранных настройках отображается в окне, показанном на рисунке 16. После нажатия кнопки Finish происходит возврат в основное окно Quartus II, с базовыми настройками вновь созданного проекта (рис. 17).

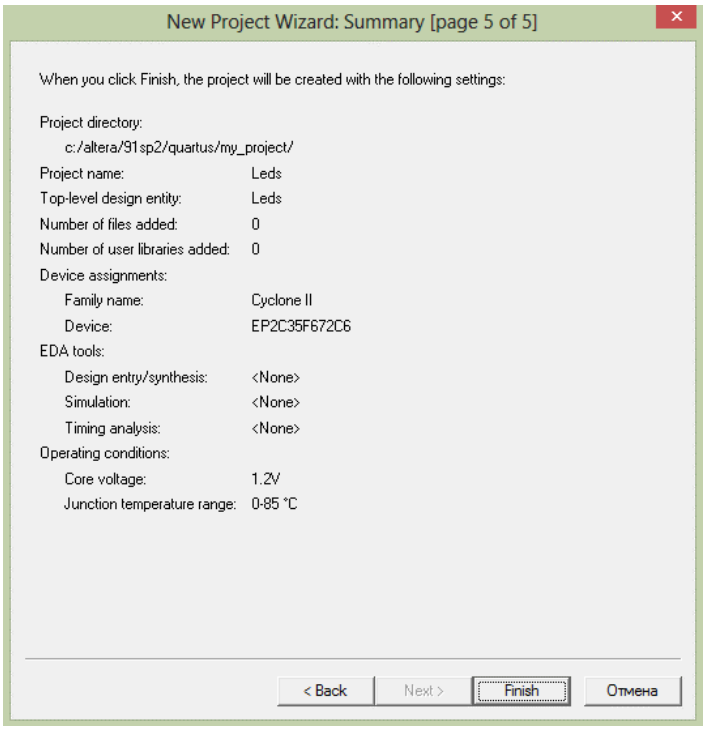


Рис. 16. Итоговое окно создания проекта

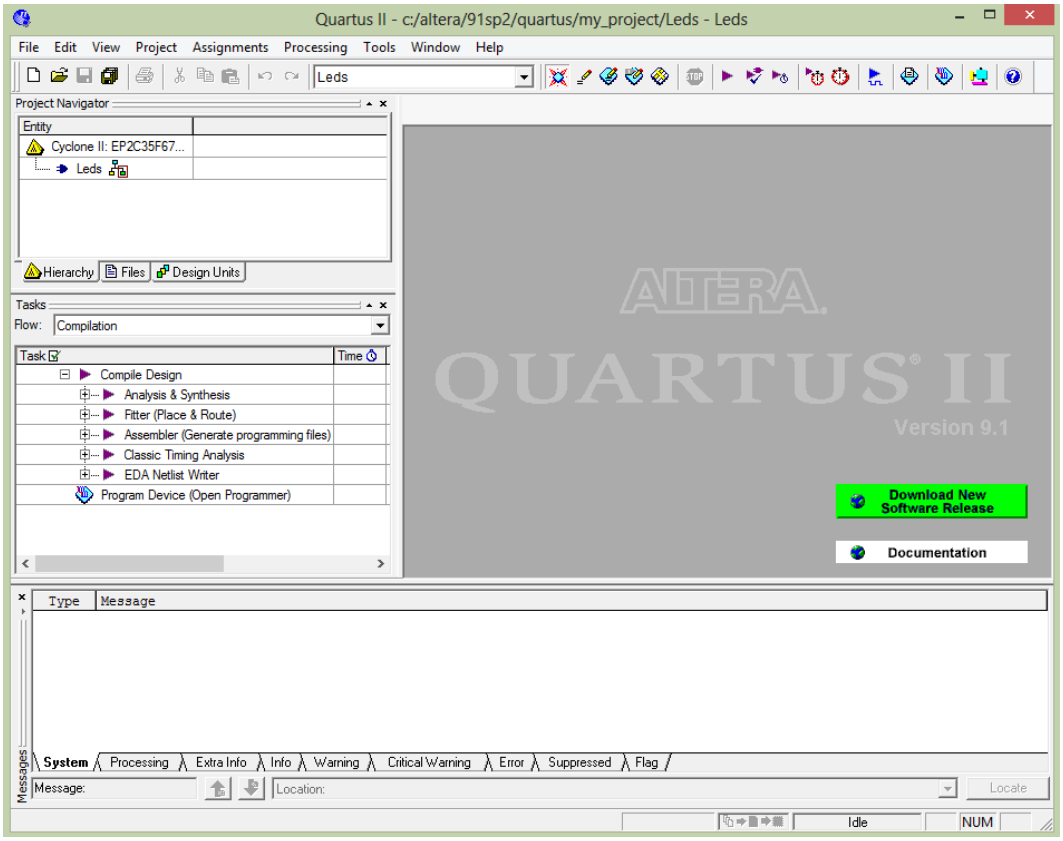
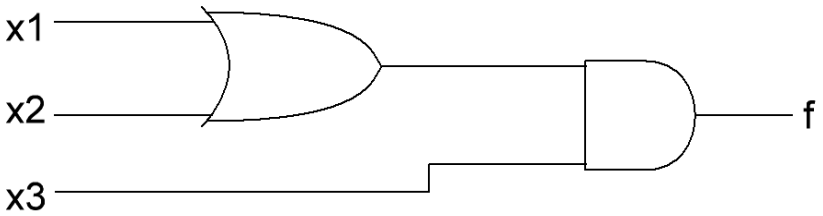


Рис. 17. Окно Quartus II для созданного проекта

Ввод проекта с помощью графического редактора

В качестве примера проектирования мы применим логическую схему управления светодиодом, показанную на рис. 18. Схема используется для управления одним светодиодом от любого из двух переключателей x1 и x2, а переключатель x3 является входом разрешения. Таблица истинности для схемы приведена на рис. 18 справа.



x1	x2	x3	f
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	1
0	0	1	1

Рис. 18. Логическая схема контроллера управления светодиодом

Чтобы приступить к выполнению схемы, нажмите кнопку «New» в основном окне Quartus II, в появившемся диалоговом окне (рис. 19) выберите «Block Diagram/Schematic File», далее нажмите «OK». Откроется окно графического редактора (рис. 20). В первую очередь необходимо указать имя файла, для этого в меню «File» выберите «Save as», появится окно (рис. 21), где в строке «Имя файла» напечатайте Leds, затем нажмите «Сохранить».

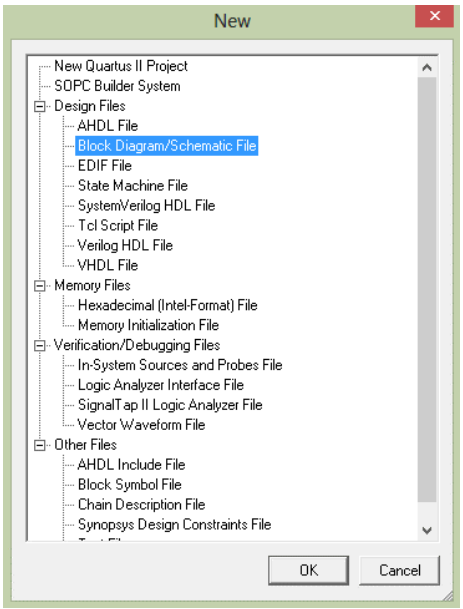


Рис. 19. Окно выбора новых файлов

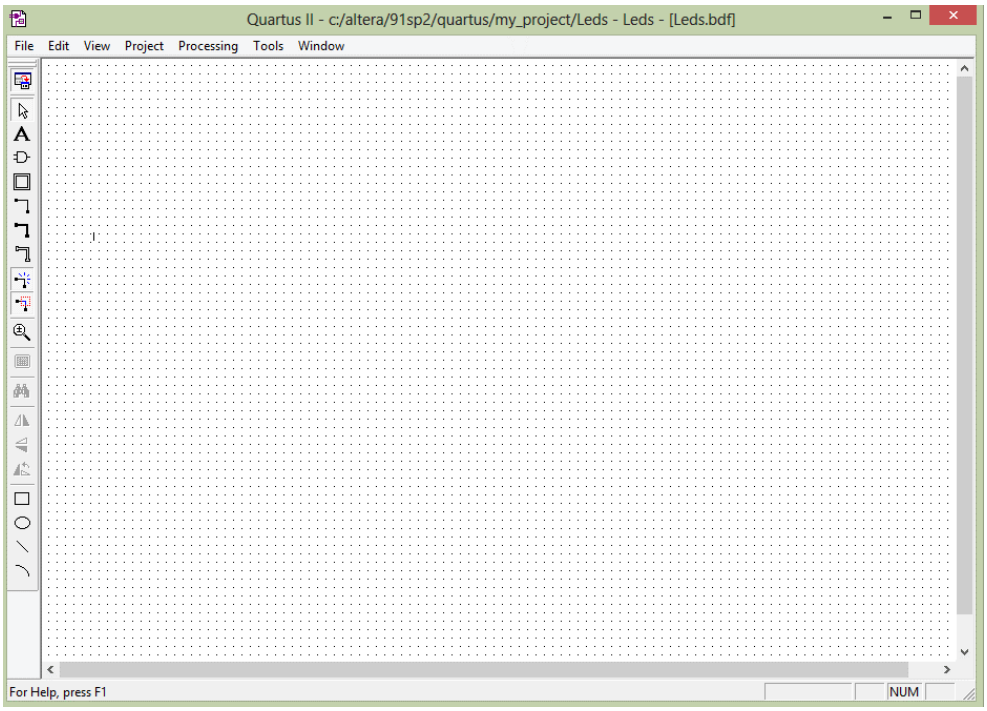


Рис. 20. Окно графического редактора

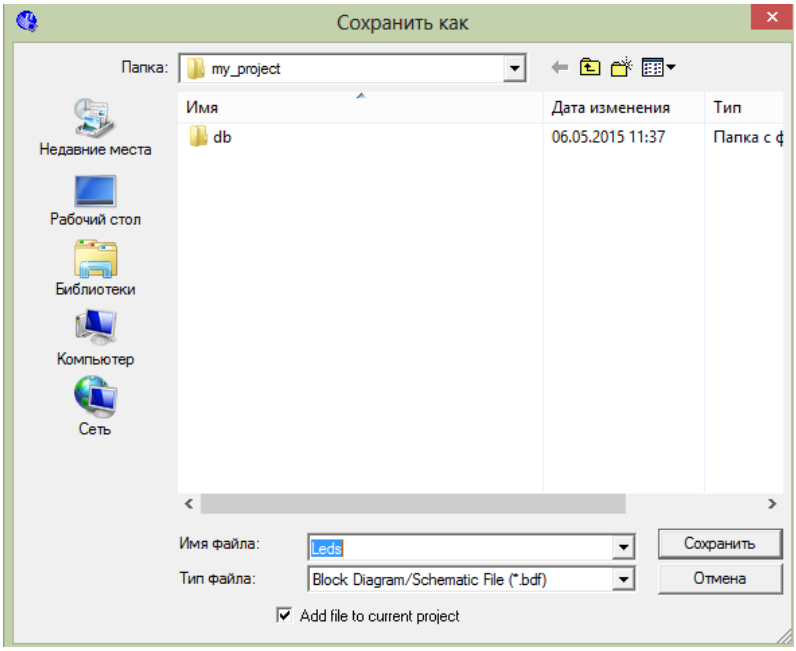


Рис. 21. Имя файла

Импорт символов логических вентилей

Графический редактор предоставляет набор библиотек с базовыми логическими элементами, которые могут быть импортированы в схему. Дважды щелкните на пустом месте в окне графического редактора или нажмите на кнопку в панели инструментов, которая выглядит как логическое «И». Появится всплывающее окно (рис. 22), выберите логический элемент and2 из иерархии библиотеки primitives/logic и нажмите кнопку «OK».

Теперь элемент появится в окне графического редактора, щелчком мыши разместите элемент в любом месте окна графического редактора. Таким же способом выберите элемент or2. Далее из библиотеки primitives/pin найдите элементы ввода/вывода «input» и «output» (рис. 23). При необходимости элементы можно поворачивать и отображать, пользуясь соответствующими инструментами.

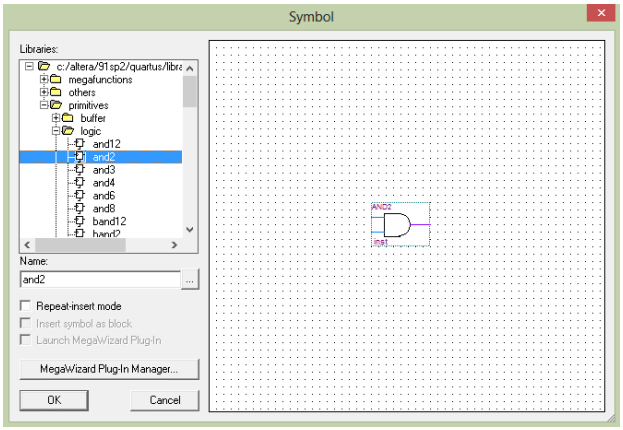


Рис. 22. Окно выбора логического элемента из библиотеки

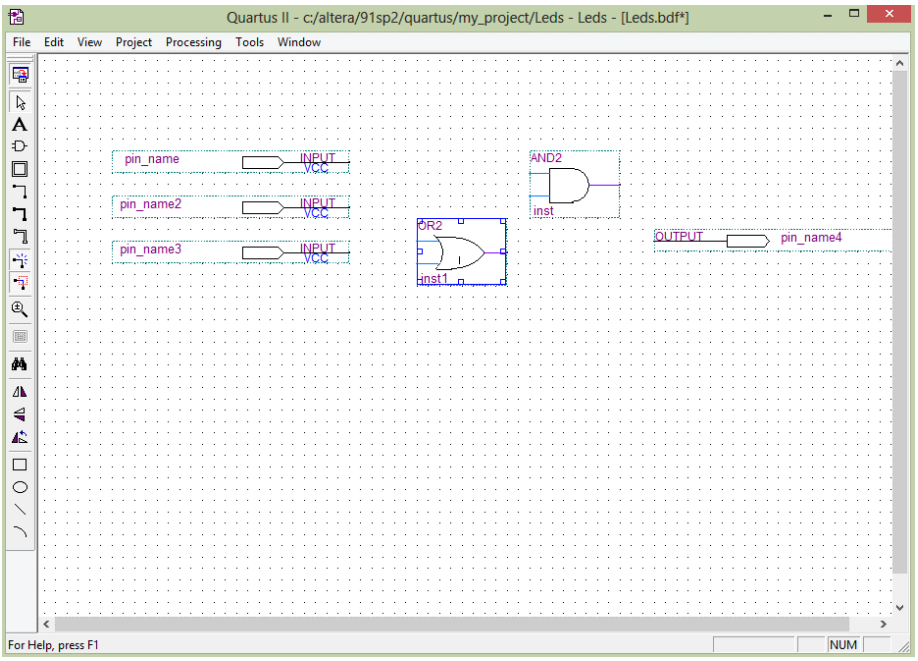


Рис. 23. Импорт элементов ввода/вывода в окне графического редактора

Назначение имен элементов ввода/вывода

После двойного щелчка по элементу ввода/вывода появляется окно свойств вывода (рис. 24), где в строке Pin name(s) назначаются имена элементов ввода/вывода. Введите имена элементов ввода/вывода в соответствии с рис. 11 (x1, x2, x3, f).

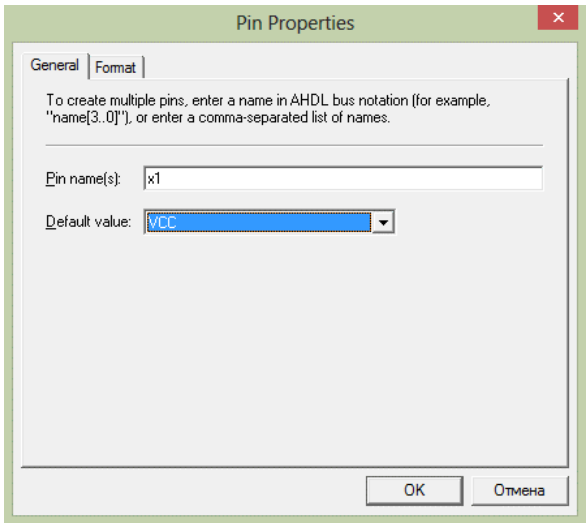


Рис. 24. Окно свойств элементов ввода/вывода

Подключение узлов с проводами

Обеспечение функциональности схемы требует соединения логических элементов с помощью линий или проводов. В панели инструментов нажмите на кнопку «Orthogonal Node Tool». Наведите указатель мыши на правый край элемента входа x1. Нажмите и удерживайте кнопку мыши, перетащите мышь вправо до тех пор, пока указатель не достигнет входа элемента or2, после чего отпустите. Если действия были выполнены правильно, то на соединительной линии не должно быть «крестиков», в случае их появления необходимо точнее доводить линию до выводов символьного элемента. Дорисуйте схему так, как это показано на рис. 25. Сохраните файл (File/Save).

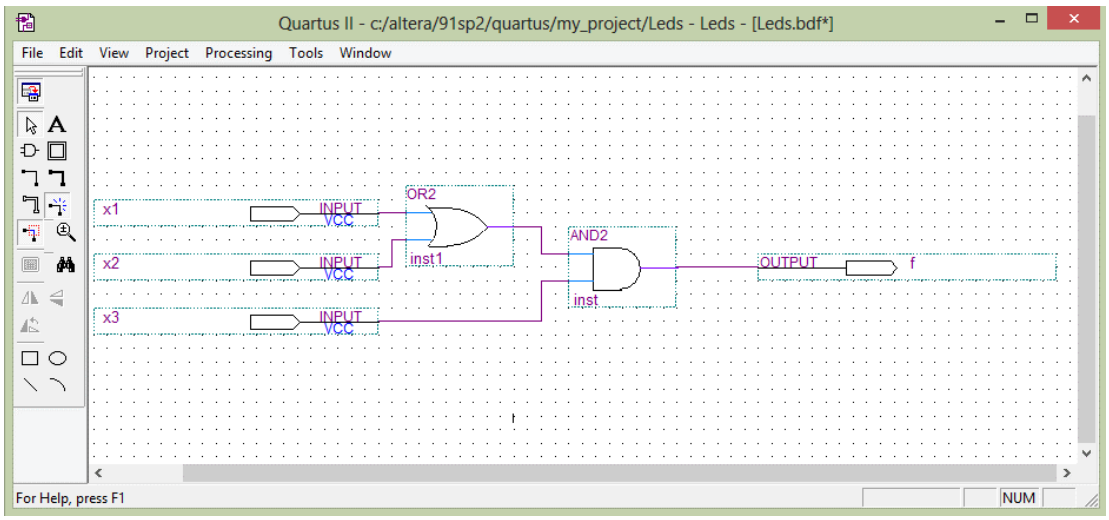


Рис. 25. Завершенная схема

Прежде чем приступить к компиляции данной схемы, создадим ту же самую схему контроллера управления светодиодом на языке описания

аппаратуры Verilog HDL, т.к. действия, необходимые для компиляции и моделирования проектов, будут идентичными.

Программный ввод проекта.

Для ввода проекта программным способом на основе языка Verilog HDL в меню Quartus II нажмите на кнопку «New», выберите «Verilog HDL File» (рис. 26).

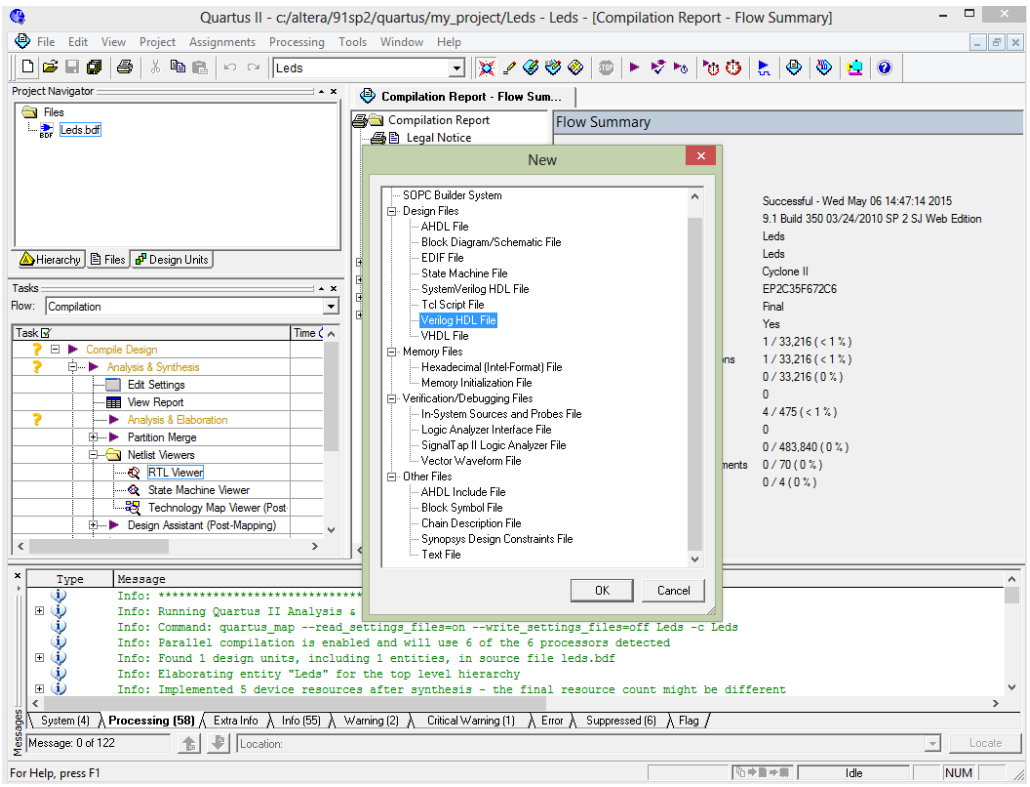


Рис. 26. Окно создания Verilog файла

В появившемся текстовом редакторе наберите Verilg-код, как показано на рис. 27. Verilog HDL код начинается с ключевого слова «module» и последующим его названием, в нашем примере «Leds», заканчивается файл ключевым словом «endmodule». После объявления модуля в круглых скобках через запятую идет последовательная декларация портов ввода/вывода, перед закрытием круглой скобки знаки препинания не ставятся. Круглая скобка закрывается точкой с запятой. Далее в конструкции непрерывного присваивания «assign» описывается код функции логической схемы одного вывода. Завершив написание кода, модуль необходимо сохранить (рис. 28); обратите внимание на то, что название сохраняемого фала должно совпадать с названием объявленного модуля.

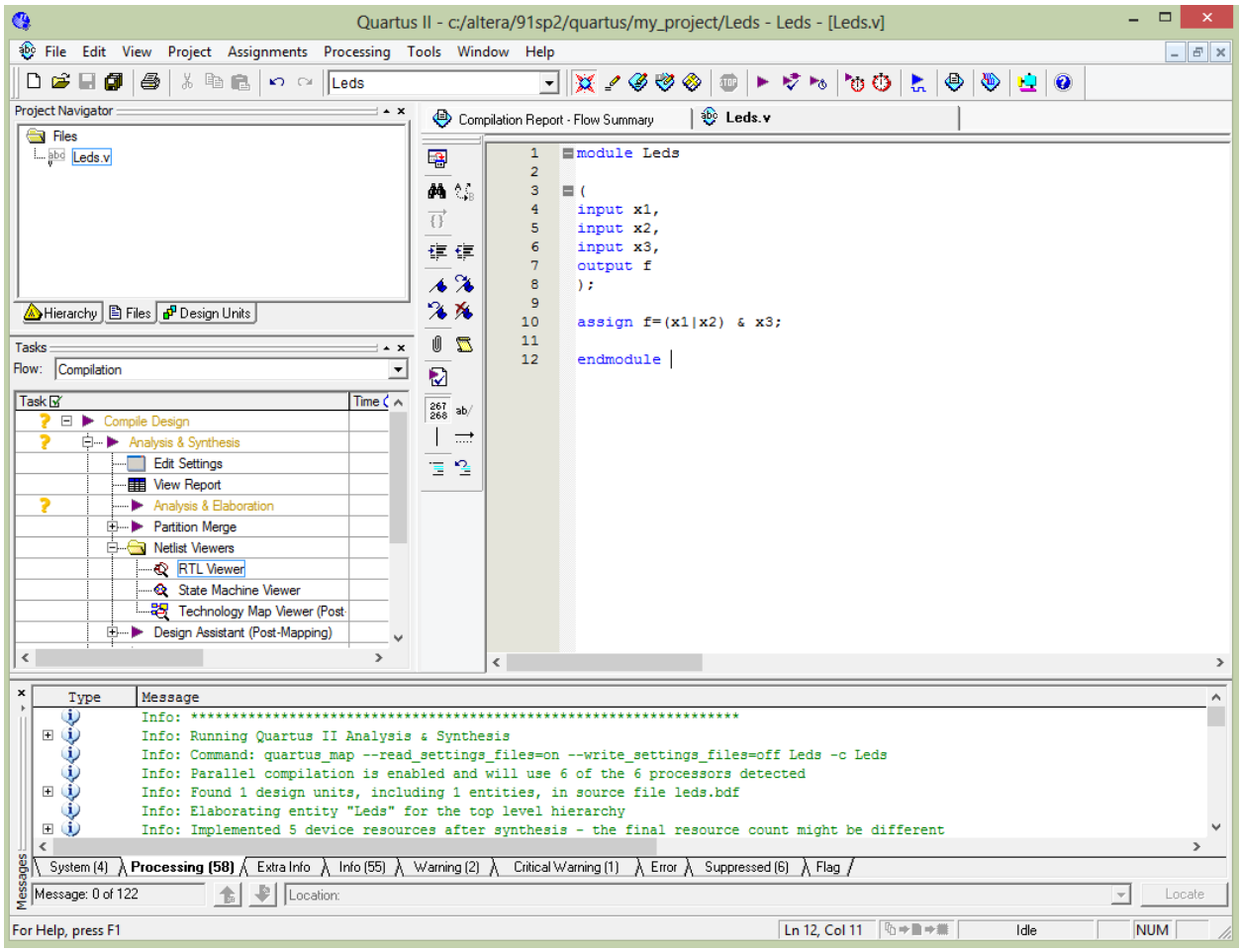


Рис. 27. Пример Verilog-кода контроллера управления светодиодом, соответствующий логической схеме на рис. 18

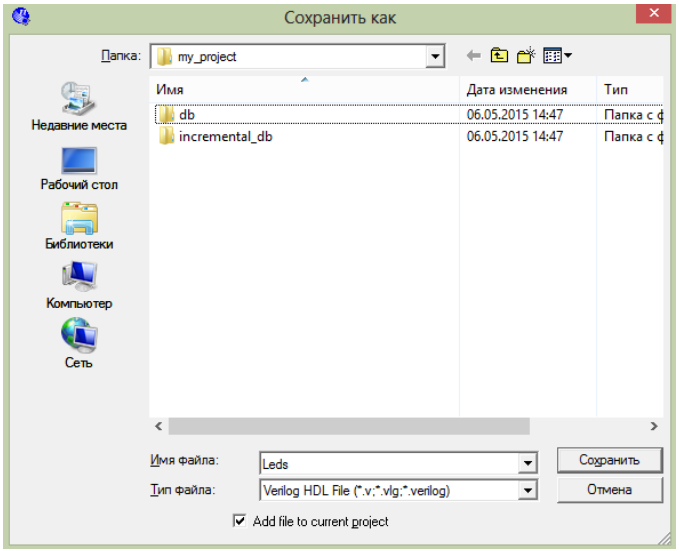


Рис. 28. Окно сохранения Verilog HDL файла

Компиляция разработанной схемы

Введенные вышеприведенными способами файлы Leds.bdf (или Leds.v) обрабатывается несколькими инструментами Quartus II, которые анализируют файл, синтезируют схему и генерируют ее реализацию на целевой микросхеме. Эти инструменты управляются прикладной программой под названием Compiler. Перед запуском компиляции необходимо указать файл верхнего уровня иерархии, для этого щелкните правой кнопкой мышки по файлу Leds.bdf (или Leds.v) в окне навигатора проекта (Project Navigator). В появившемся всплывающем окне выберите пункт меню «Set as Top-Level Entity». Далее можете приступить к компиляции, нажав кнопку «Run» в меню Quartus II или нажав на кнопку компиляции на панели инструментов (которая выглядит как фиолетовый треугольник). Quartus II последовательно выполняет процедуры в различных встроенных инструментах, за которыми можно проследить в левом окне «Task». По завершению компиляции всплывает окно отчета компиляции – «Compilation Report», где отображаются параметры проекта (рис.29). В случае наличия в компилируемых файлах ошибок Quartus II выводит их в окне «Message», расположенном в нижней части основного окна.

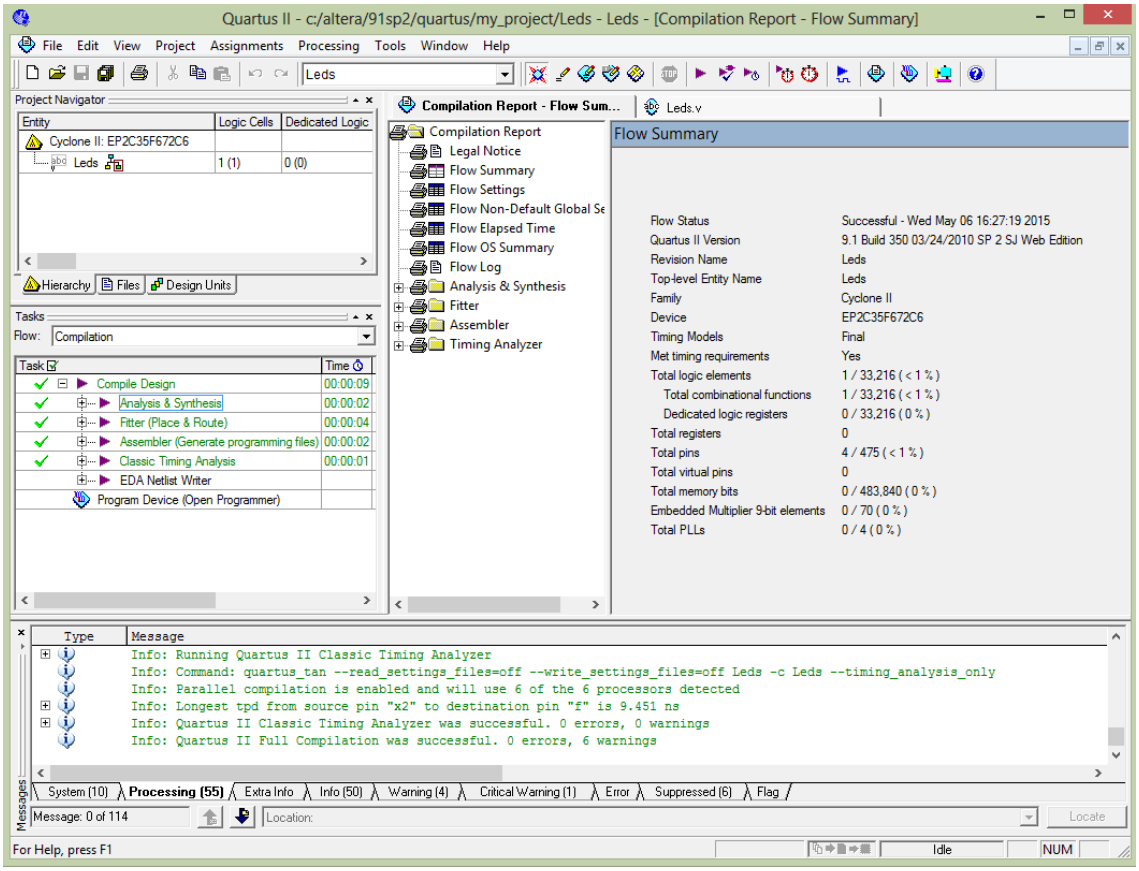


Рис. 29. Окно отчета компиляции

В приведенной компиляции Quartus II волен выбирать любые контакты на выбранной микросхеме ПЛИС в качестве входов и выходов. Тем не менее,

плата DE2 имеет строго выполненные соединения между контактами микросхемы ПЛИС и элементами на плате, поэтому после первой компиляции необходимо переназначить порты ввода/вывода.

Назначение контактов микросхемы ПЛИС

В меню «Tools» или на панели задач выберите «Pin Planer», откроется окно (рис. 30), визуализирующее расположение контактов микросхемы ПЛИС. Используемые в проекте элементы ввода/вывода отображены в нижней части окна в виде таблицы. Двойным щелчком в поле «Location» в строке с названием порта назначьте контакт микросхемы (f – **Pin_AE23**, x1 – **Pin_N25**, x2 – **Pin_N26**, x3 – **Pin_P25**), которые, согласно разводке платы DE2, подсоединяются к светодиоду LEDR0 и движковым переключателям SW0, SW1 и SW2. Назначенные контакты на изображении микросхемы подсвечивается черным. Для сохранения назначений в основном окне Quartus II нажмите на кнопку в виде стопки из трех дискеток.

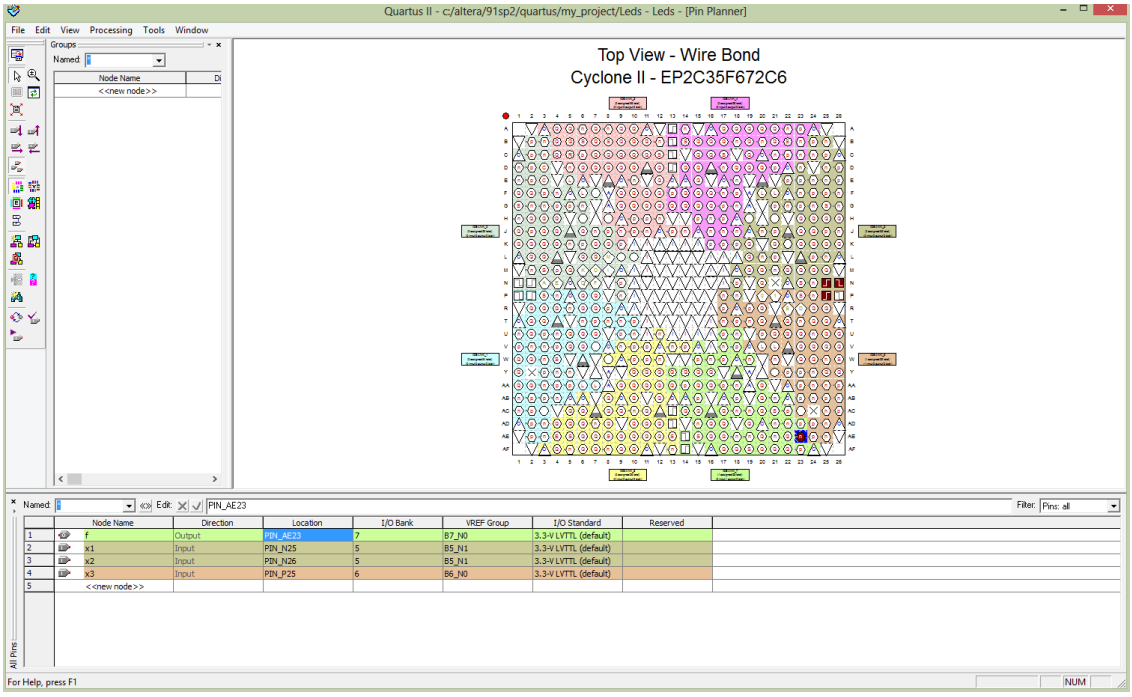


Рис. 30. Окно планировщика контактов

Для сохранения назначений контактов в отдельном файле выберите в окне «Pin Planer» меню «File/Export», наберите имя файла (Leds) и нажмите «Export» (рис. 31). Конечный файл Leds.csv будет содержать информацию о назначениях контактов и может быть импортирован в другие проекты.

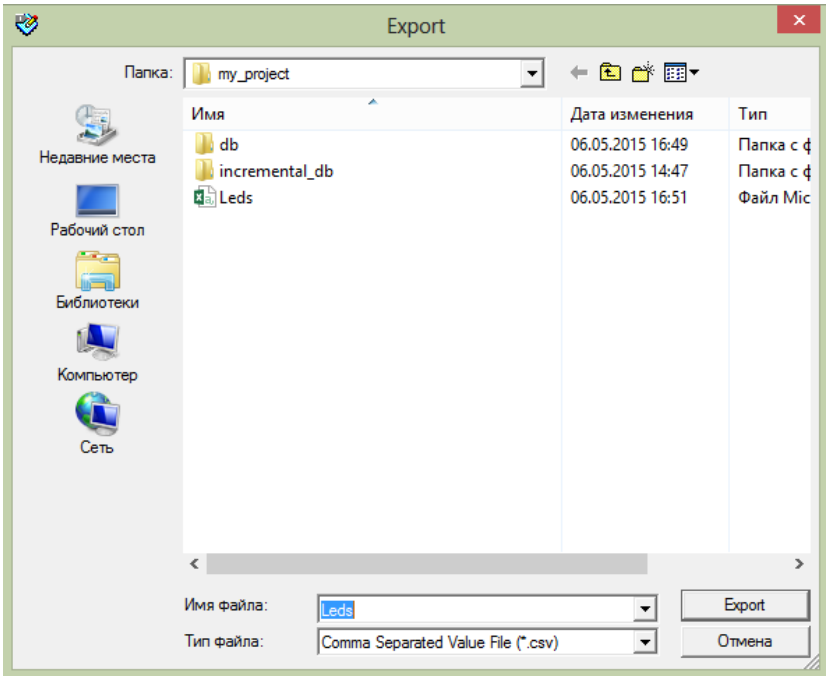


Рис. 31. Окно экспорта назначений портов ввода/вывода

Моделирование разработанной схемы

Прежде чем приступить к конфигурации микросхемы ПЛИС на плате DE2, целесообразно выполнить моделирование разработанной схемы для проверки правильности работы. САПР Quartus II включает в себя инструмент моделирования, который может быть использован для имитации поведения проектируемой схемы. Перед моделированием схемы необходимо создать требуемые временные диаграммы, называемые *тестовыми векторами*. Для этого необходимо указать элементы ввода/вывода, а также возможные внутренние точки схемы, которые разработчик хочет проанализировать. Инструмент моделирования или симулятор использует тестовые векторы в реализованной модели схемы и вычисляет ожидаемые отклики. Для ввода тестовых векторов в Quartus II 9 используется редактор Waveform:

1. В меню File/New выберите «Vector Waveform File» (рис. 32).

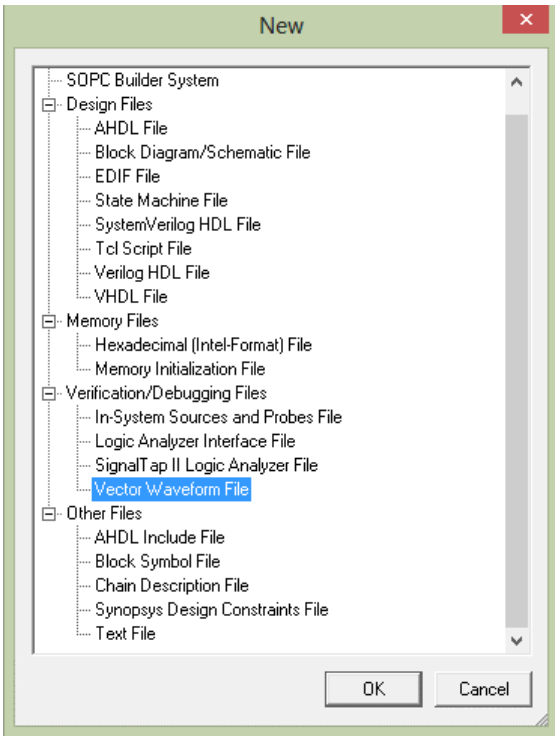


Рис. 32. Меню File/New

2. Окно Waveform Editor или редактора форм сигналов изображено на рис. 33. Задайте интервал времени, в течении которого будет производиться моделирование (в меню Edit/End Time…/Time-500 ns).

Сохраните файл под именем Leds.vwf. Настройте масштаб времени, выбрав пункт меню «View/Fit in Window». Настроенное окно Waveform Editor приведено на рис. 34.

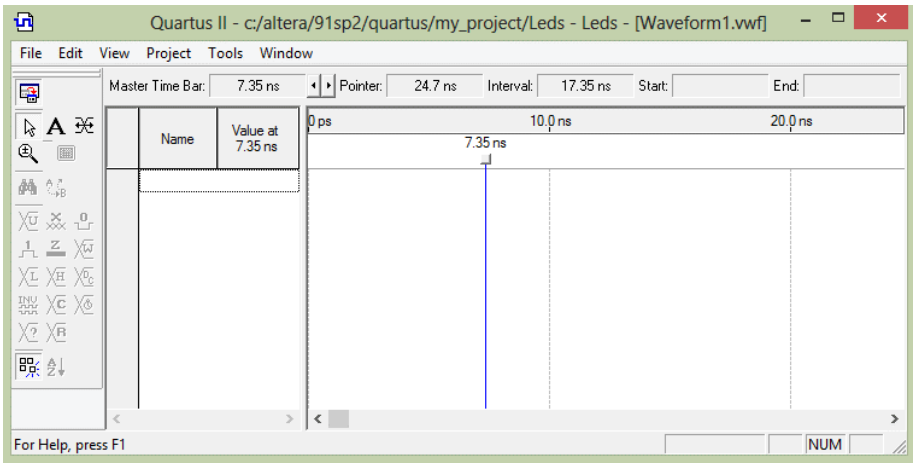


Рис. 33. Окно Waveform Editor

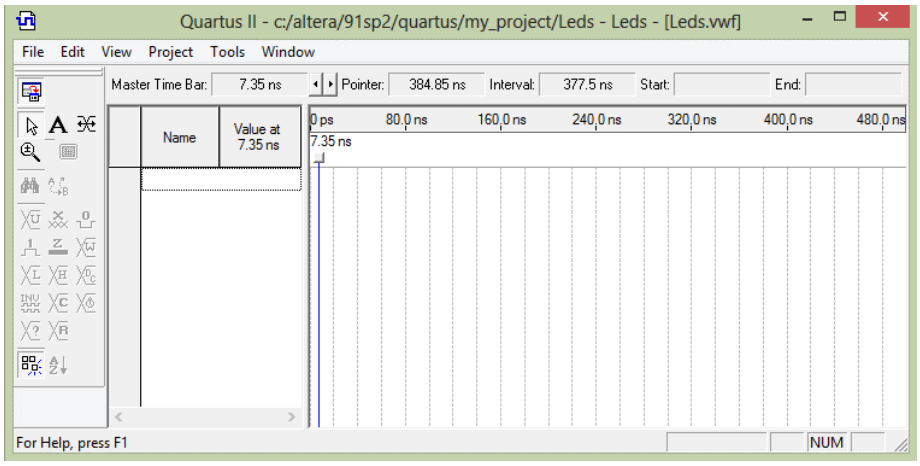


Рис. 34. Настроенное окно Waveform Editor

3. Двойным щелчком по белому полю в левой части окна Waveform Editor вызовите меню вставки узлов или шин – Insert Node or Bus (рис. 35). Нажмите кнопку «Node Finder», в строке «Filter» появившегося окна выбора узлов (рис. 36) выберите «Pins: all» и нажмите на кнопку «List». В левой части окна «Nodes Found» появятся порты ввода/вывода, использованные в проекте; нажав кнопку «>>», скопируйте эти порты в правое окно «Selected Nodes». Нажав кнопку «OK» сначала в окне «Node Finder», а затем и в окне вставки узлов или шин, перейдите в редактор форм сигналов, где уже появились выбранные порты (рис. 37).

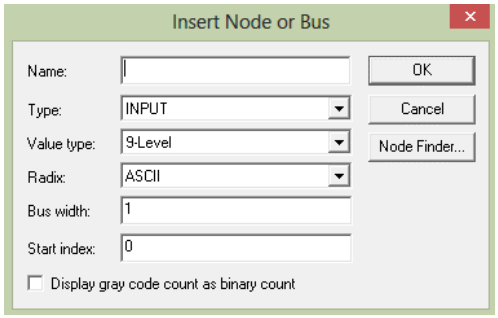


Рис. 35. Окно вставки узлов или шин

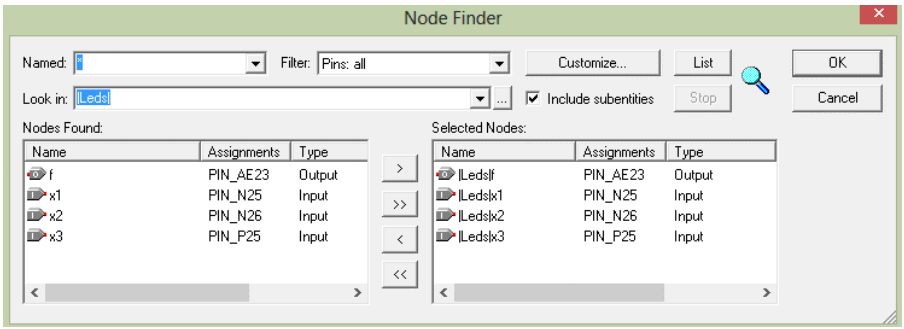


Рис. 36. Выбор портов для вставки в редактор форм сигналов

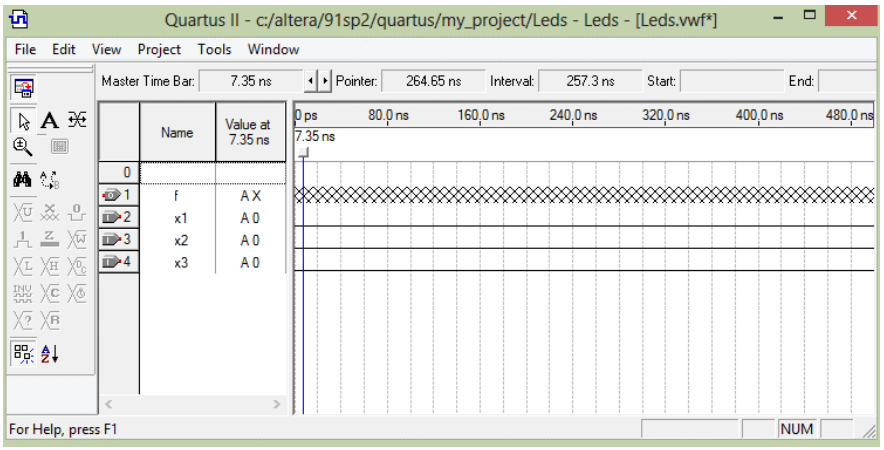


Рис. 37. Порты, необходимые для моделирования

4. После выбора элементов ввода/вывода, для моделирования необходимо ввести логические значения для входов x1, x2 и x3. Логические значения на выходе f будут сгенерированы автоматически в симуляторе. Для задания логических значений выделите мышкой необходимый отрезок времени на линии сигнала (или группы линий сигналов), обратите внимание на то, что инструменты в левой части окна стали активными, и выберите необходимый вам логические уровни «0» или «1». Задайте тестовые значения входов x1, x2 и x3 в соответствии с таблицей на рис. 11 так, как это показано на рис. 38. Сохраните файл.

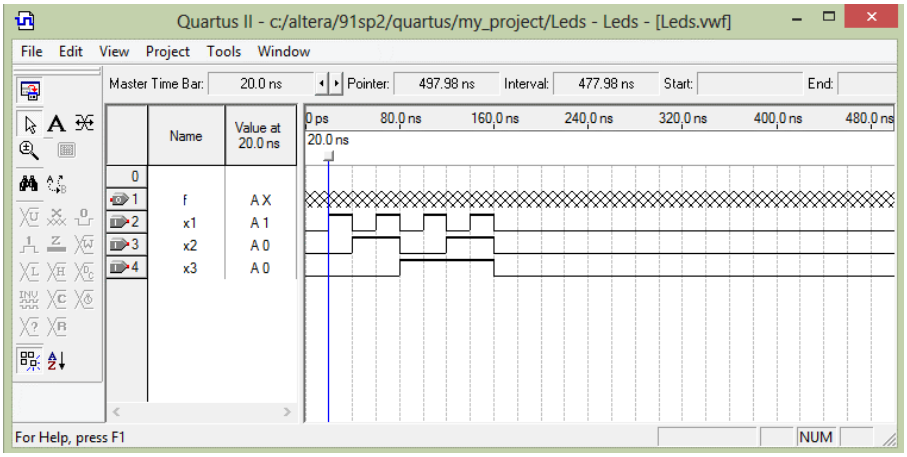


Рис. 38. Настройка тестовых значений

Проектируемая схема может быть смоделирована двумя способами. Проще всего предположить, что логические элементы и соединительные провода в ПЛИС идеальные, что подразумевает отсутствие задержек в распространении сигналов. Такое представление называется функциональным моделированием (functional simulation). Более сложный вариант моделирования учитывает все задержки распространения, и называется “временное моделирование” (timing simulation), которое не будет

рассматриваться в этом методическом пособии. Как правило, функциональное моделирование используется для проверки правильности функционирования схемы на первом этапе верификации и требует гораздо меньше временных затрат, т.к. моделирование выполняется на основе простых логических выражений, которые определяют схему.

Функциональное моделирование

Для выполнения функционального моделирования в основном окне Quartus II на панели инструментов выберите кнопку карандаша с двумя черточками , появится окно настроек проекта. В списке «Category» перейдите на свойство «Simulator Setting». В строке «Simulation mode» укажите «Functional» и нажмите кнопку «OK» (рис. 39). Прежде чем приступить к моделированию, необходимо сгенерировать список соединений для функционального моделирования в меню «Processing/Generate Functional Simulation Netlist»; после выполнения этого действия появится окно подтверждения генерации (рис. 40).

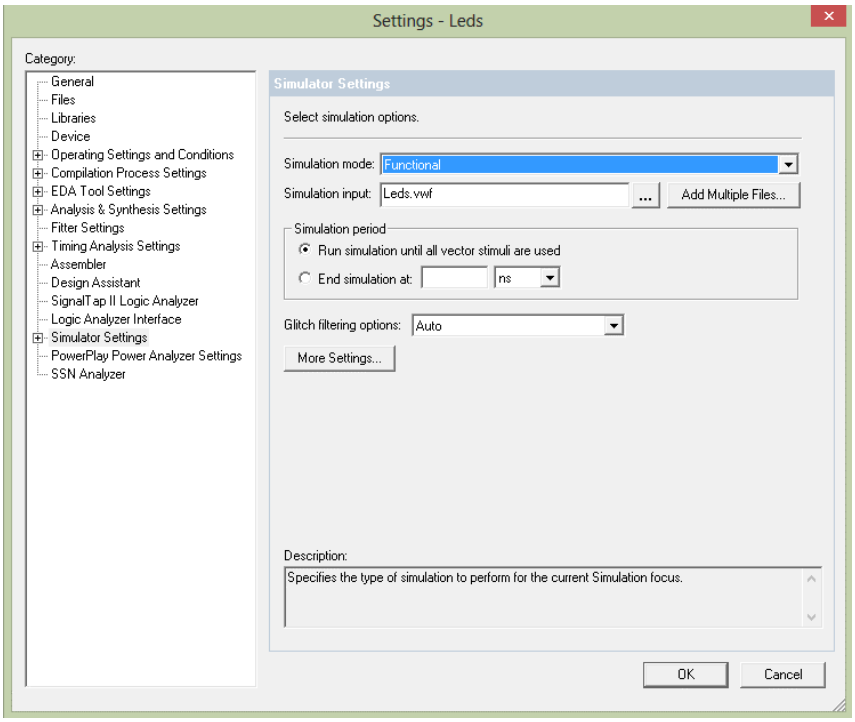


Рис. 39. Указание режима моделирования

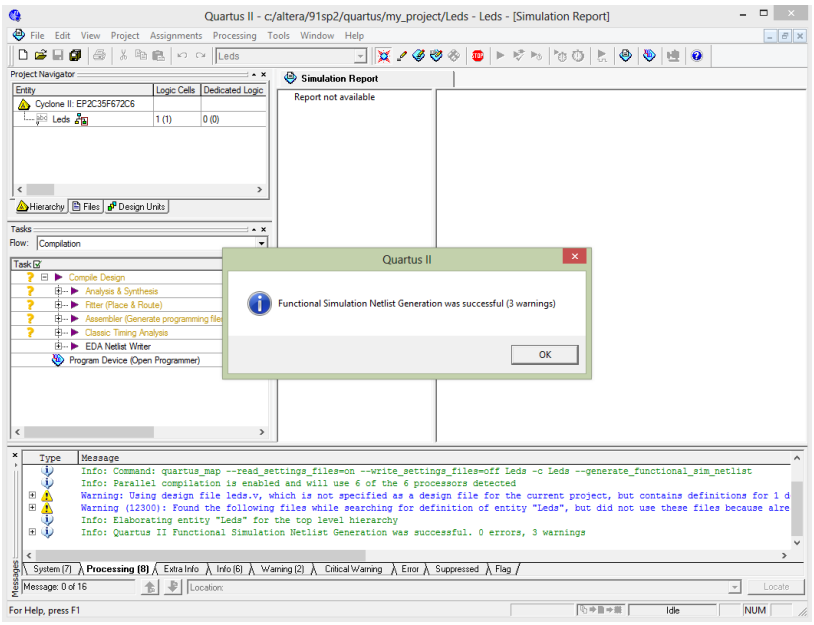


Рис. 40. Отчет генерации списка соединений для функционального моделирования

Для выполнения моделирования нажмите на кнопку синего треугольника с импульсами . Выполнив моделирование, Quartus II откроет окно «Simulation Waveform» с результатом вычислений, где может быть выполнена оценка правильности работы схемы (рис. 41)

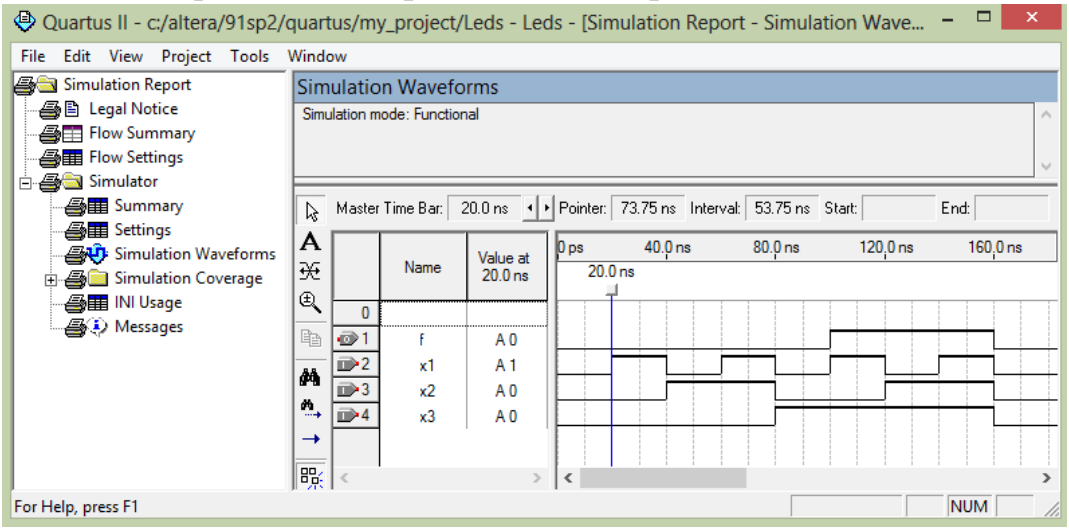


Рис. 41. Результат функционального моделирования

Программирование и конфигурация ПЛИС

Убедившись в правильности работы схемы, можно перейти к программированию и конфигурации микросхемы ПЛИС, что требует конфигурационного файла, генерируемого ассемблером Quartus II.

Отладочный комплект DE2 позволяет выполнить конфигурацию ПЛИС двумя различными способами:

1) JTAG (Joint Test Action Group) – предназначен для программирования ПЛИС файлами с расширением .sof (и др.) и интерактивной отладки проекта с использованием инструментов Quartus II (например: Signal Tap II, Logic Analyzer и др.). Название Joint Test Action Group является именем рабочей группы, разработавшей данный интерфейс на базе интерфейса IEEE 1149.1. Официальное название стандарта Standard Test Access Port and Boundary-Scan Architecture. В режиме JTAG конфигурационные данные загружаются непосредственно в микросхему ПЛИС. При этом конфигурация хранится только до тех пор, пока питание микросхемы остается включенным, после выключения питания информация о конфигурации теряется.
2) AS (Active Serial) – предназначен для программирования конфигурационного постоянного запоминающего устройства (ПЗУ), которая сохраняет данные конфигурации и подключена к микросхеме ПЛИС так (в этом случае Quartus II применяет файлы с расширением .pof), что после программирования ПЗУ данные загружаются в ПЛИС при включении питания или реконфигурации. Таким образом, нет необходимости конфигурировать ПЛИС с помощью программного обеспечения Quartus II каждый раз после включения питания микросхемы.

Выбор между этими двумя режимами производится с помощью переключателя RUN/PROG на плате DE2. Положением RUN выбирается режим JTAG, в то время как позиция PROG предназначена для выбора режима AS.

Данные конфигурации передаются от компьютера к отладочной плате DE2 с помощью устройства USB-Blaster, которое соединяет порт USB на компьютере с левым разъемом USB платы DE2. Прежде чем использовать это устройство, необходимо убедиться, что драйвер USB-Blaster установлен на компьютер. Если этот драйвер еще не установлен, зайдите в диспетчер устройств, правой кнопкой мыши щелкните на USB-blaster и выберите «Обновить драйвер». Выберите «Выполнить поиск драйверов на этом компьютере» и в папке установки Quartus II укажите директорию drivers (например: altera\91sp2\quartus\drivers\usb-blaster\).

Программирование ПЛИС в режиме JTAG

Подключите USB-кабель между компьютером и левым разъемом USB на плате DE2. Переведите переключатель RUN/PROG в положение RUN и включите питание DE2. В меню Quartus II выберите Tools/Programmer или

щелкните кнопку на панели инструментов. Появиться окно программирования ПЛИС (рис. 42). Здесь в опции «Hardware Setup» необходимо указать USB-Blaster (рис. 43), в строке «Mode» выбрать режим конфигурации JTAG, а в меню «Add File» включить конфигурационный файл Leds.sof (рис. 44).

Leds.sof – двоичный файл, созданный компилятором Qusrtus II, необходимый для конфигурирования микросхемы ПЛИС. Расширение .sof означает SRAM Object File. Отметим также, что выбранной микросхемой ПЛИС является EP2C35F672, которая используется на плате DE2. Поставьте галочку в поле Program/Configure, как показано на рисунке 44.

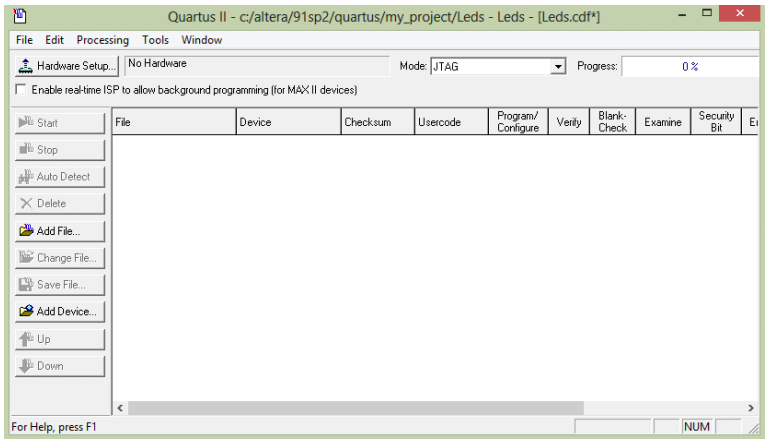


Рис. 42. Окно программатора

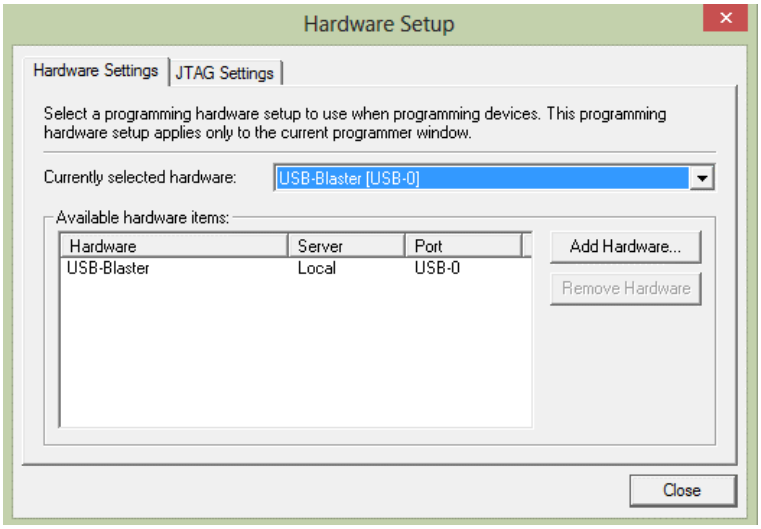


Рис. 43. Окно выбора устройств программирования

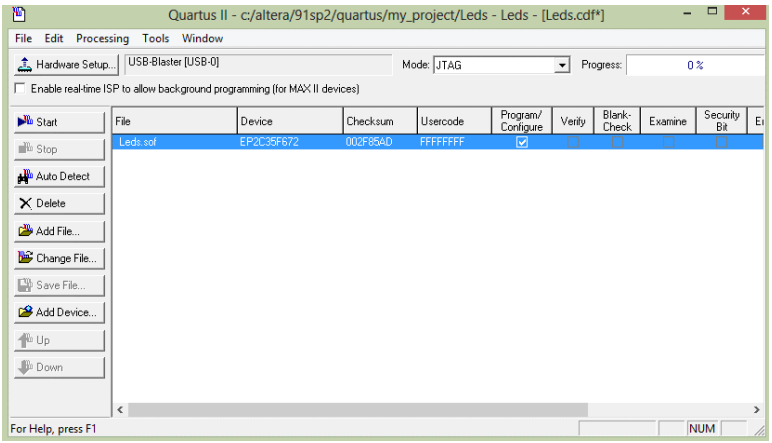


Рис. 44. Завершенный вид настройки окна программатора

Теперь нажмите на кнопку «Start» в окне на рис. 44, в поле «Progress» отображается состояние конфигурации в %. После успешного завершения программирования на плате DE2 загорается светодиод. Если Quartus II выдает ошибку при программировании ПЛИС, убедитесь в правильности выполнения описанных действий.

Программирование ПЛИС в режиме Active Serial

В этом случае данные конфигурации будут загружены в устройство конфигурации на плате DE2, которое называется EPCS16 (ПЗУ). Чтобы выбрать необходимую микросхему конфигурационного ПЗУ, зайдите в раздел «Settings/Devises» и кликните по кнопку «Devise and Pin Options» (рис. 45).

В окне на рис. 45 нажмите на кнопку «Device and Pin Options», что приведет к окну на рис. 46. Далее, выберите вкладку «Configuration» (появится окно настроек устройства конфигурации, показанное на рис. 47). В поле «Configuration device» поставьте галочку в разделе «Use configuration device» и выберите ПЗУ-EPCS16, нажмите кнопку ОК. По возвращении к окну на рис. 45 нажмите кнопку ОК. Перекомпилируйте разработанный ранее проект.

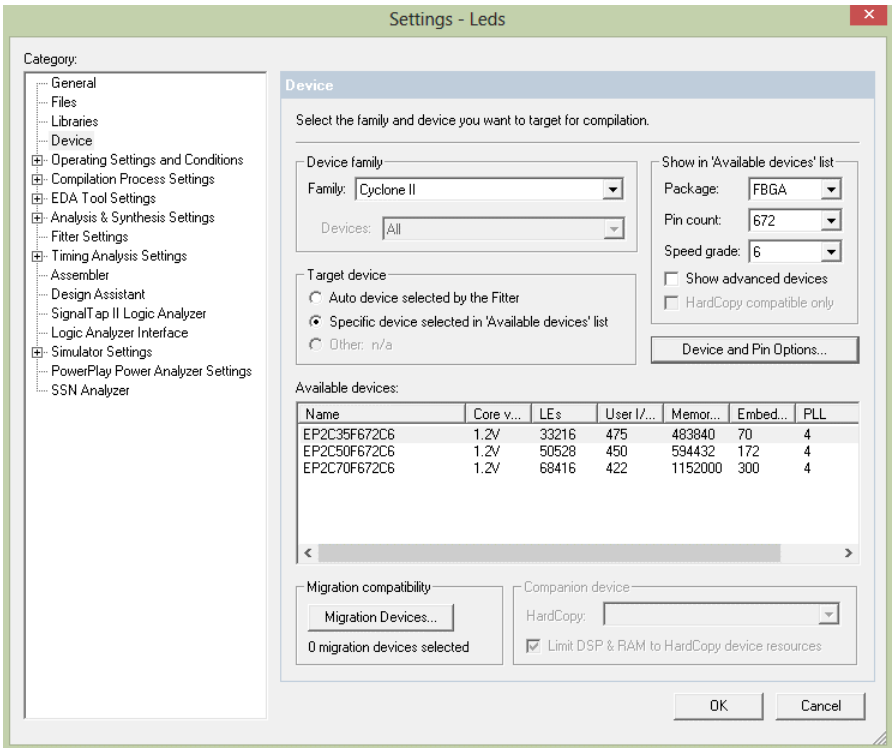


Рис. 45. Окно настроек устройства

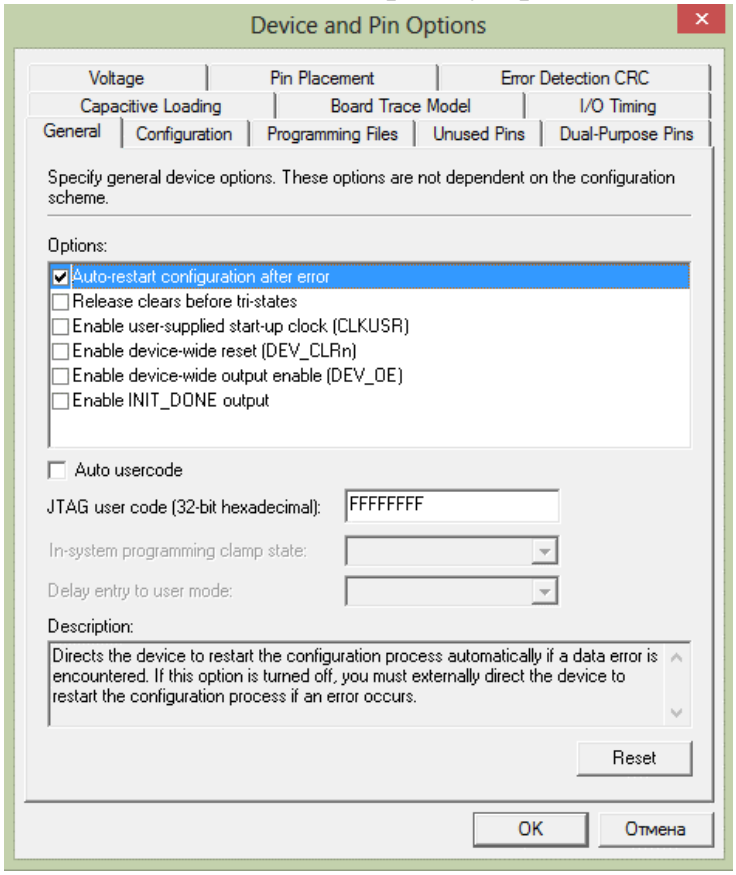


Рис. 46. Окно настроек устройства и контактов

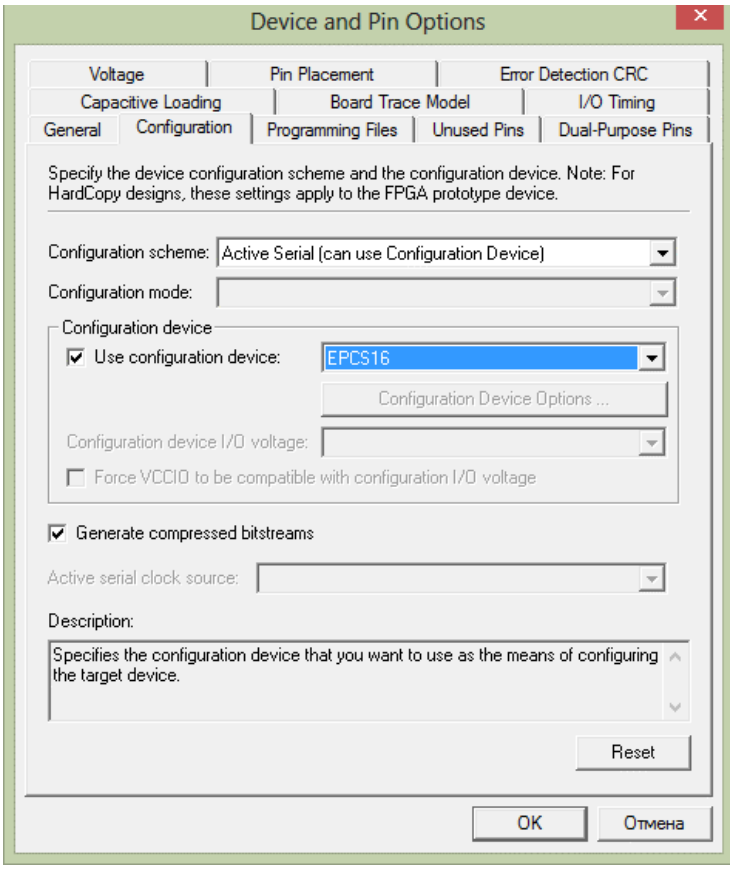


Рис. 47. Окно настроек устройства конфигурации

Дальнейшие действия процедуры программирования ПЛИС схожи с описанным выше режимом JTAG. На верхней панели основного окна Quartus II нажмите на кнопку «Programmer», появится окно конфигурации (рис. 48). В разделе «Mode» смените режим JTAG на «Active Serial Programming», после чего появится окно, схожее с рис. 49. Это окно предупреждает о необходимости сброса указанных ранее устройств для смены режима программирования. Нажмите кнопку «Да» или «Ok». Далее, нажав кнопку «Add File», выберите конфигурационный файл «Leds.pof» (рис. 50) и установите галочку в поле «Program/Configure» (как показано на рис. 51.). Теперь, переключив переключатель «Run/Prog» на плате DE2 в положение «Prog», запустите процесс программирования ПЗУ, нажав на кнопку «Start» в окне конфигурации устройства.

После завершения программирования, когда поле «Progress» отобразит 100%, выключите питание платы DE2, переключите переключатель «Run/Prog» в положение «Run» и снова включите питание платы DE2. В этом случае микросхема ПЛИС будет автоматически программироваться при каждом включении питания платы логической схемой «Leds», хранящейся в конфигурационном ПЗУ.

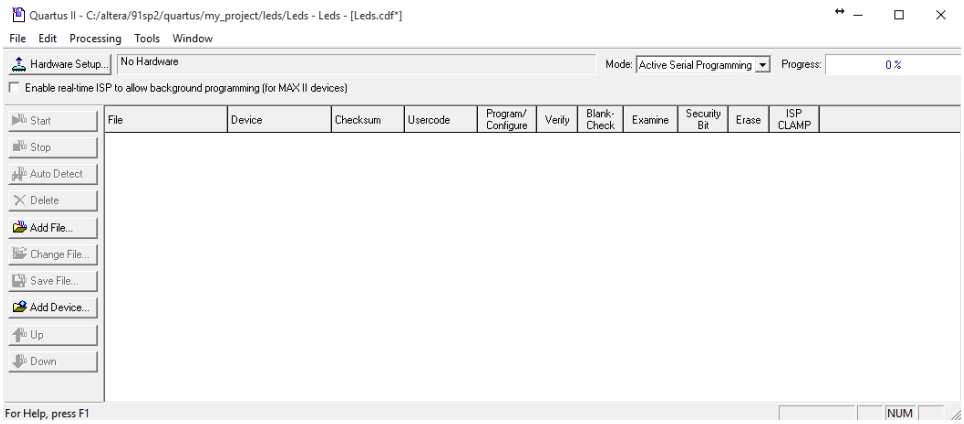


Рис. 48. Окно конфигурации устройства с выбором режима программирования «Active Serial Programming».

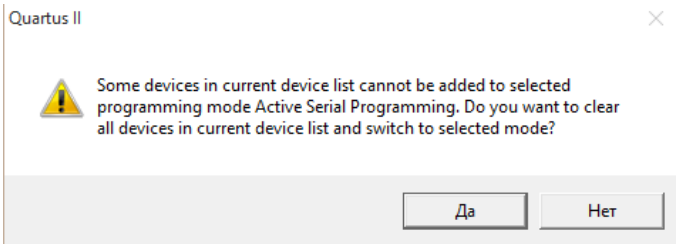


Рис. 49. Сброс ранее выбранных устройств

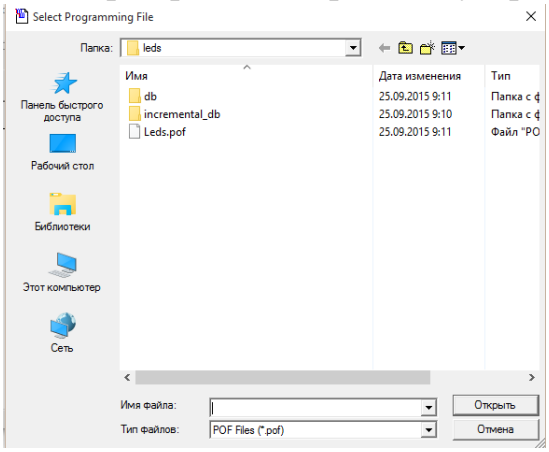


Рис. 50. Выбор файла конфигурации

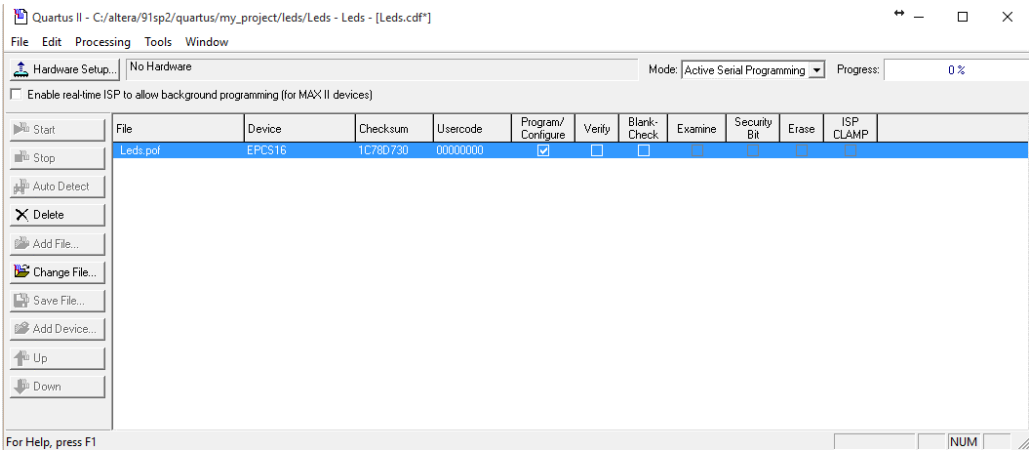


Рис. 51. Вид окна конфигурации устройства

Проверьте правильность работы вашей логической схемы.

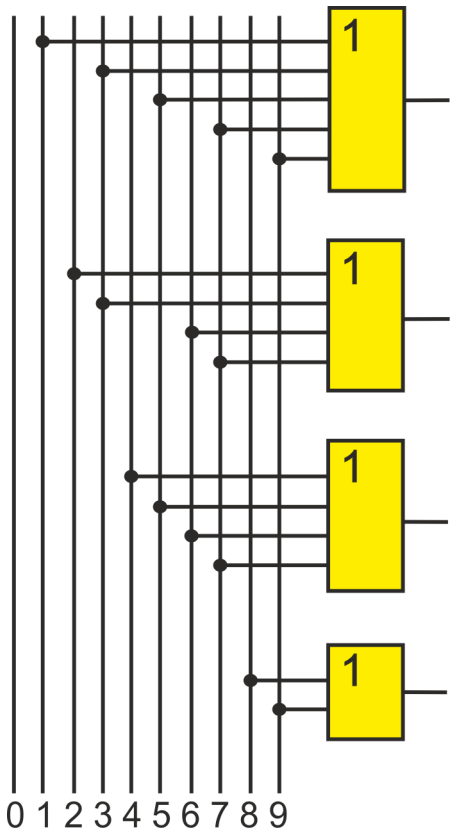
Если Вы хотите внести изменения в разработанную схему, сначала закройте окно конфигурации устройства. Затем сделайте необходимые изменения в схеме или коде, перекомпилируйте проект и запрограммируйте плату описанным выше способом.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

Ниже представлены 5 контрольных заданий, выполнение которых поможет закрепить рассмотренный выше материал.

1. Логическая схема шифратора

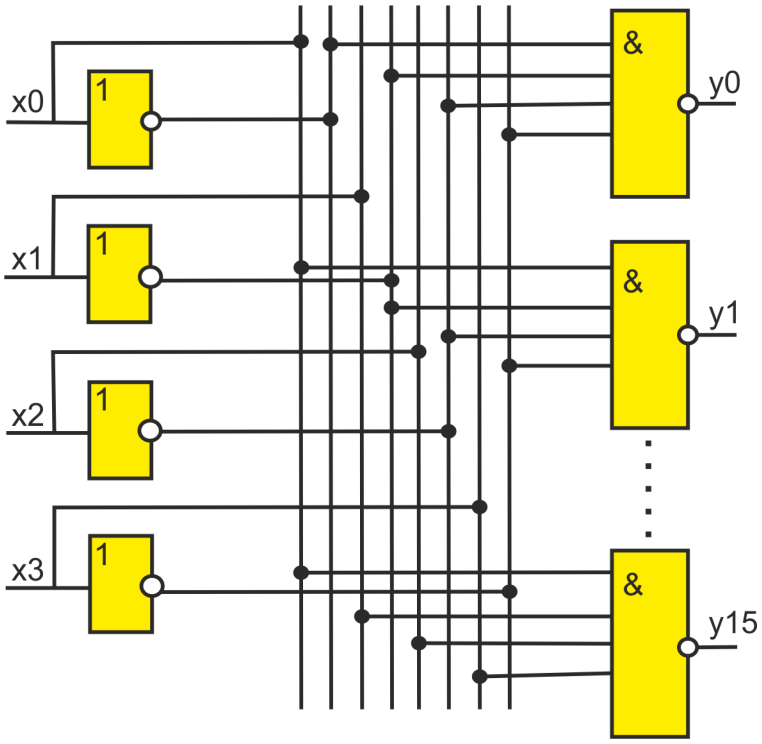
Спроектируйте в графическом редакторе Quartus II логическую схему шифратора 10x4, представленную на рисунке. Выполните симуляцию заданной схемы, представьте результат. Составьте таблицу истинности для данного шифратора.



Задать на информационных входах один или несколько активных уровней, а на стробирующем входе – разрешающий уровень. Убедиться в работоспособности шифратора, проверяя соответствие сигналов на выходах таблице истинности (с учетом того, что выходной двоичный код индицируется светодиодами в прямом виде). Меняя комбинации входных сигналов, проверить логику формирования сигналов, выполнив функциональное моделирование в Quartus II.

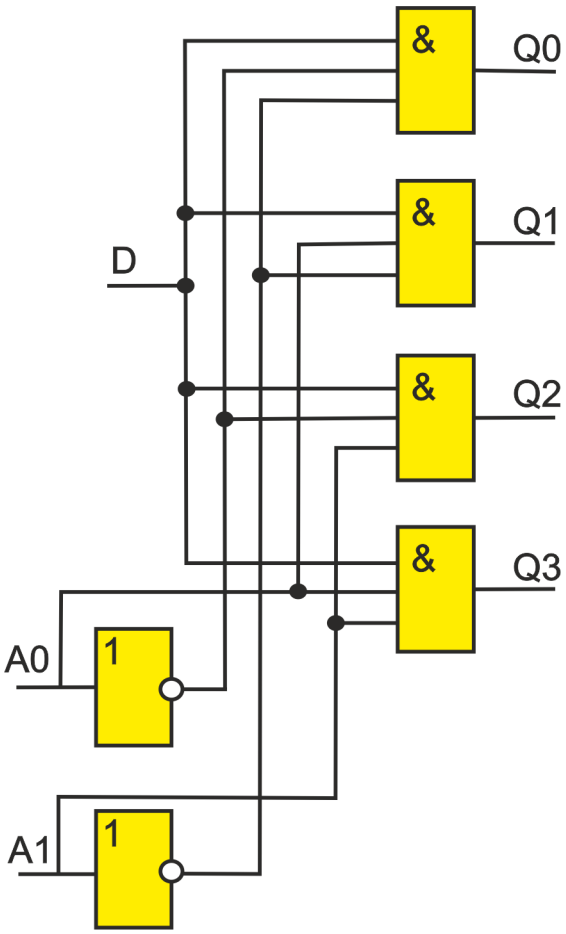
2. Логическая схема дешифратора

Спроектируйте в графическом редакторе Quartus II логическую схему дешифратора 4x16, представленную на рисунке. Составьте таблицу истинности для данного дешифратора.



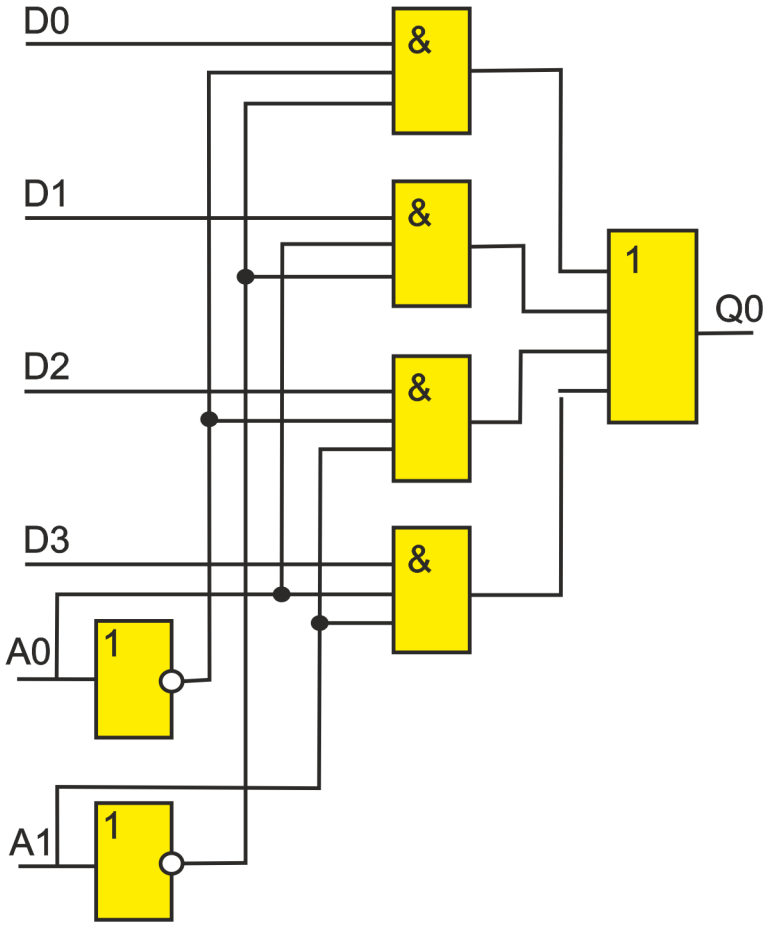
3. Логическая схема демультиплексора

Спроектируйте в графическом редакторе Quartus II логическую схему демультиплексора, представленную на рисунке. Составьте таблицу истинности для данного демультиплексора.



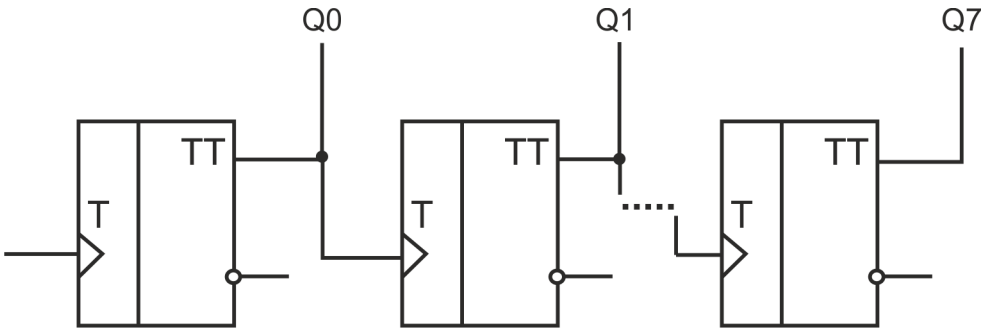
4. Логическая схема мультиплексора

Спроектируйте в графическом редакторе Quartus II логическую схему мультиплексора, представленную на рисунке. Составьте таблицу истинности для данного мультиплексора.



Проверить работоспособность мультиплексора 4→1, меняя входные коды на адресных линиях для заданной комбинации значений на информационных входах.

5. Спроектируйте в графическом редакторе Quartus II логическую схему восьмиразрядного счетчика, реализованную на Т-триггерах согласно рисунку ниже. Выполните функциональную симуляцию и проверьте работоспособность схемы.



ЛИТЕРАТУРА

1. Щука А. А. Электроника / А.А. Щука. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 751 с. — (Учебная литература для вузов).
2. Титце У. Шенк К. Полупроводниковая схемотехника. Том I. Издательство: "ДМК Пресс", 2009. – 832 с.
3. Стешенко В.Б. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов. – М.: ДОДЭКА, 2000. – 128 с.
4. Лехин С.Н. Схемотехника ЭВМ. – СПб.: БХВ-Петербург, 2010. – 652 с.
5. Введение в систему автоматизированного проектирования Quartus II: учебное пособие. – М.: ГОУ ВПО МГУЛ, 2011. – 147 с.
6. Акчурин А.Д., Шерстюков О.Н. Практикум по цифровой электронике. Издание второе, исправленное. Учебно-методическое пособие для студентов по специальности “радиофизика и электроника” Института физики. Казань, 2016. – 100 с.
   URL: http://kpfu.ru/portal/docs/F549825898/DigManual_2.pdf.
7. My First FPGA Design Tutoria
   URL: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/tt/tt_my_first_fpga.pdf