Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

М.С. Сахаров,
Д.В. Озеркин

ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ НА БАЗЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ В СРЕДЕ РАЗРАБОТКИ «QUARTUS»

Методические указания по лабораторным работам
для студентов направления 11.03.03 «Конструирование и технология электронных средств» и специальности 25.05.03 «Техническая эксплуатация транспортного радиооборудования»

Томск 2022

УДК 004.436.2
ББК 32.971
С 221

Рецензент:

Кривин Н.Н., доцент кафедры конструирования и производства радиоаппаратуры,
канд. техн. наук

Сахаров М.С., Озеркин Д.В.
С 221 Проектирование цифровых устройств на базе программируемых логических интегральных схем в среде разработки «Quartus»: метод. указания по лабораторным работам/ М.С. Сахаров, Д.В. Озеркин. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2022. – 38 с.

Данные методические указания являются частью учебно-методического комплекса и предназначены для подготовки и проведения лабораторных работ по дисциплине «Программируемые интегральные логические схемы». В методических указаниях содержится необходимый теоретический материал, методические рекомендации для выполнения лабораторных работ, а также варианты заданий для самостоятельного выполнения. Пособие содержит описание четырех лабораторных работ.

Методические указания предназначены для студентов всех форм обучения, обучающихся по специальности 25.05.03 «Техническая эксплуатация транспортного радиооборудования» и направлению подготовки бакалавров 11.03.03 «Конструирование и технология электронных средств». Могут использоваться студентами других специальностей и направлений подготовки.

Одобрено на заседании каф. конструирования и производства радиоаппаратуры, протокол № 16 от 24.03.2022

УДК 004.436.2
ББК 32.971

ОГЛАВЛЕНИЕ

1 ВВЕДЕНИЕ

Цифровые устройства на сегодняшний день являются неотъемлемой частью практически любых технических систем. Они известны достаточно давно, однако по мере развития материально-технической обеспечения для их построения использовалась разная элементная база: от электронно-вакуумных приборов до больших интегральных схем. Одним из последних продуктов развития цифровых устройств являются ***программируемые логические интегральные схемы*** (ПЛИС), которые представляют собой цифровые интегральные микросхемы, состоящие из универсальных программируемых логических элементов и программируемых соединений между этими элементами. В отличие от устройств, внутренняя функциональность которых жестко задана производителем, например, у логических микросхем или у микроконтроллеров,ПЛИС конфигурируется в лабораторных условиях непосредственно инженером, разрабатывающим цифровое устройство, что позволяет инженерам решать множество различных задач, связанных с проектирование цифровых устройств.Использование ПЛИС также позволяет легко модифицировать функционал уже существующего и используемого цифрового устройства.Таким образом, ключевая особенность ПЛИС заключается в том, что инженер может сам создавать архитектуру любой сложности из логических элементов, чтообеспечиваетвозможность сделать целый ряд проектов с использованием одной микросхемы.

Стандартный процесс проектирования на ПЛИС начинается с описанияпроектируемой системы, которое может быть структурным и алгоритмическим (поведенческим). В первом случаемогут быть использованы либографический редактор, в котором размещаются требуемые элементы структуры системы и связи между ними,либо языки описания аппаратуры, например,Verilog или VHDL. Во втором случае описывается поведение системы, при этом используется только язык описания оборудования.В независимости от способа описания системы на этом этапе создается цифровая схема, которая в дальнейшем записывается в ПЛИС. Затем идет процесс компиляции проекта, симуляция проекта, программирование и проверка работоспособности цифровой схемы при помощи отладочной платы ПЛИС. На рисунке 1.1 показаны основные этапы проектирования устройств на базе ПЛИС.

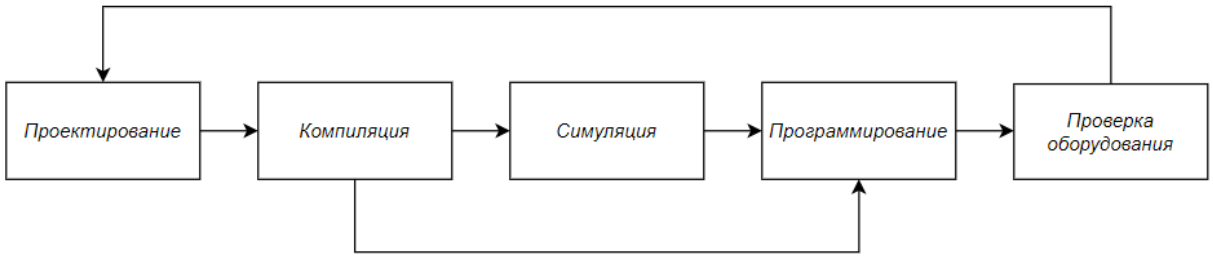


Рисунок 1.1 – Этапы проектирования ПЛИС

В данном курсе лабораторных работ для проектирования используется программное обеспечение «Quartus», обеспечивающее полный цикл для разработки и создания высокопроизводительных систем на кристалле. Оно объединяет в себе проектирование, синтез, размещение элементов, трассировку соединений и верификацию, связь с системами проектирования других производителей. При этом «Quartus» позволяет реализовать как структурное, так и алгоритмическое проектирование цифровых систем.

Целью данного цикла лабораторных работ является освоение методов синтезаи анализа цифровых устройств в среде разработки «Quartus».

2 ОПИСАНИЕ ЛАБОРАТОРНОГО МАКЕТА

В состав лабораторного макета входят:

1. Отладочная платаDE0-Nano-SoC Board, в состав которой входят микросхема класса «система на кристалле»AlteraCycloneV, содержащая в себе ПЛИС и микропроцессорный модуль на основе ядро ARMCortex-A9, в дальнейшем именуемый процессорным модулем (ПМ), и сопутствующие устройства, список которых приведен в таблице 2.1. На рисунке 2.1 изображен макет платы с местом расположения основных компонентов.
2. Блок питания отладочной платы с выходным напряжением 5В и максимальным током 2А.
3. Персональный компьютер с операционной системой Windows версии не ниже 7, с установленной средой разработки «Quartus» версии не ниже 14.1.
4. Кабель micro-USB для записи программ

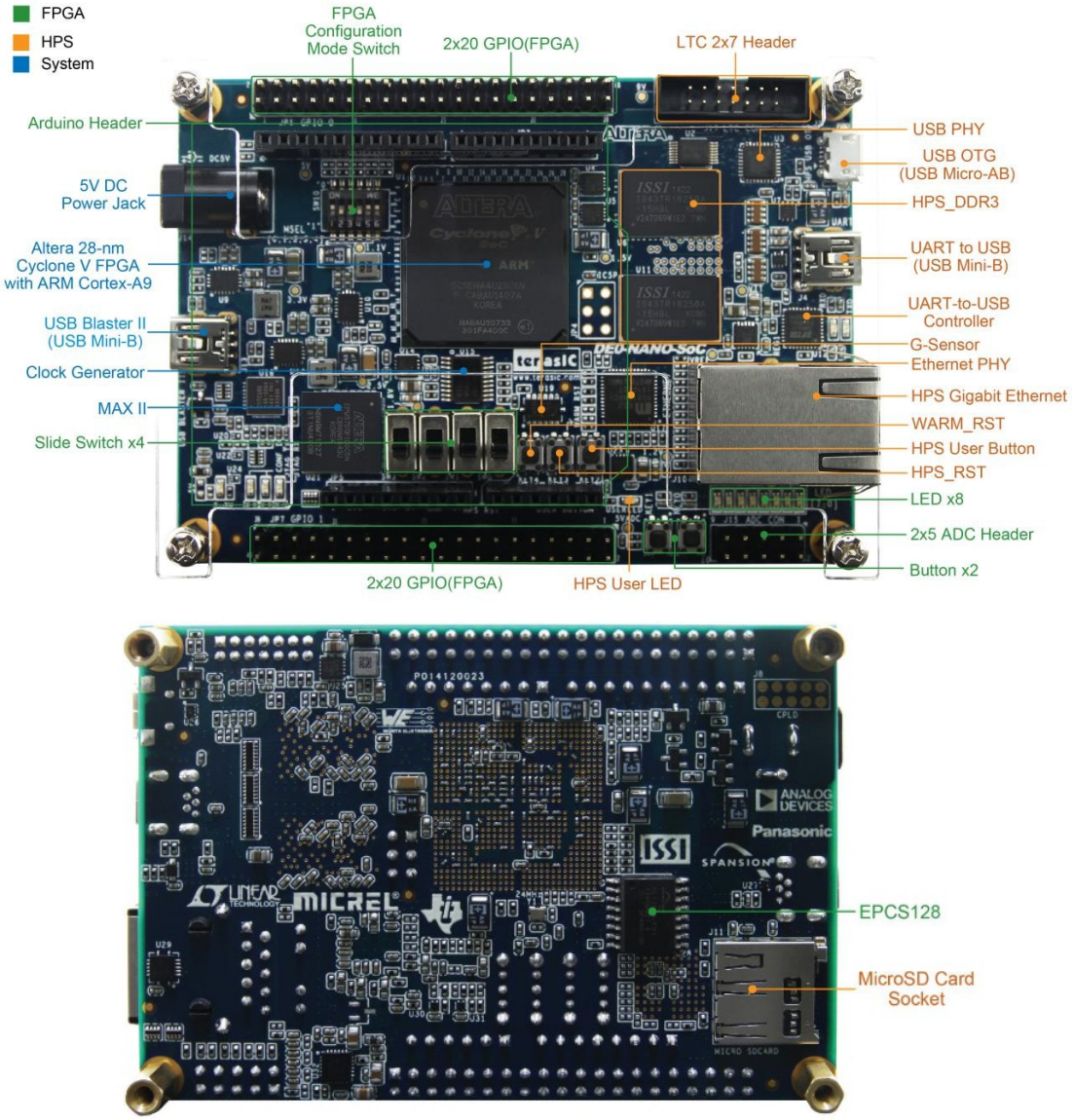


Рисунок 2.1– Макет отладочной платы DE0-Nano-SoC

Таблица 2.1 – основные блоки отладочной платы DE0-Nano-SoC с расшифровкой обозначений.

Altera 28-nm Cyclone V FPGA with ARM Cortex-A9		**AlteraCycloneV с ПЛИС и ПМ**
Модули для работы с ПЛИС	Arduino Header	Разъемы для состыковки с учебно-отладочной платой ArduinoUnoR3
	FPGA configuration mode switch	Переключатели для начальной конфигурации ПЛИС
	2*20 GPIO (FPGA)	Разъемы подключения к выводам общего назначения ПЛИС
	Slide switch *4	4 пользовательских переключателя
	Button *2	2 пользовательских кнопки
	2*5 ADC header	Разъем для подключения пользовательских сигналов к АЦП
	LED *8	8 пользовательских светодиодов
	EPCS128	Модуль памяти для хранения программы ПЛИС
Модули для работы с ПМ	LTC 2*7 Header	Разъем для подключения к последовательным интерфейсам
	USB PHY	Контролер пользовательского интерфейса USB
	USB OTG	Разъем пользовательского интерфейса USB
	HPS_DDR3	Модуль оперативной памяти
	UART to USB (USB mini-B)	Преобразователь интерфейсов UART в формат USB
	UART to USB controller	РазъемUSBдля преобразователя интерфейсов UART в формат USB
	G-sensor	Акселерометр
	Ethernet PHY	Контролер пользовательского интерфейса Ethernet
	HPS Gigabit Ethernet	Разъем пользовательского интерфейса Ethernet
	WARM_RST	Кнопка локального сброса ядра для отладки
	HPS_User_Button	Пользовательская кнопка для работы сПМ
	HPS_Reset	Кнопка общего сброса ПМ и связанных с ним устройств
	HPS User Led	Пользовательский светодиод для работы с ПМ
	Micro-SD card socket	Разъем для карты памяти Micro-SD
5V DC power jack		Разъем питания платы
USB Blaster II (USB mini-B)		Разъем для записи программ по интерфейсу USB
Clock Generator		Тактовый генератор
MAX II		Преобразователь USB-JTAG для записи программ и отладки проектов на физических носителях.

3 КРАТКОЕ ОПИСАНИЕ СРЕДЫ РАЗРАБОТКИ «QUARTUS»

«Quartus» представляет собой интегрированную среду для разработки цифровых устройств на базе программируемых логических интегральных схем (ПЛИС) фирмы Altera и обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

– создание проектов устройств;

– синтез структур и трассировку внутренних связей ПЛИС;

– подготовку данных для программирования или конфигурирования ПЛИС;

– верификацию проектов (функциональное моделирование и временной анализ);

– программирование или конфигурирование ПЛИС.

На официальном сайте фирмы Altera www.altera.com [1] можно найти наиболее полное описание системы «Quartus» и скачать бесплатную версию, в которой по сравнению с платной ограничена лишь скорость компиляции проектов, набор инструментов в обеих версиях одинаковый.

3.1 Создание проекта в «Quartus»

Для создания файла проекта следует выполнить следующие действия:

1. Выбрать среде«Quartus»опциюFile>New Project Wizard. откроется приветственное диалоговое окно проекта (рисунок 3.1).
2. Нажать Next.
3. Ввести следующую информацию о проекте:

a. ***Имя папки***, в которой будут храниться файлы проекта, например, *C:\My_design\my_first_fpga*.

b. ***Имя проекта***. Например, *fpgaTEST*. Имя объекта проектирования верхнего уровня (*thetop-leveldesignentity*) для этого проекта заполнится автоматически в соответствии с именем проекта (рисунок 3.2);

Важно: Имена файлов и директорий, открываемых в «*Quartus*», не должны содержать пробелов и нелатинских символов.

c. Нажать Next;

d. Назначить модель устройства FPGA для проектирования (рисунок 3.3);

Примечание. В данной лабораторной используется устройство *CycloneVFPGAwithARMCortex-A9*, но в поле «*Name*» следует выбрать название *5CSEMA4U23C6*, именно так называется микросхема, установленная на отладочной плате.

e. Нажать OK.

4. Появится основное окно проектирования (рисунке 3.4).

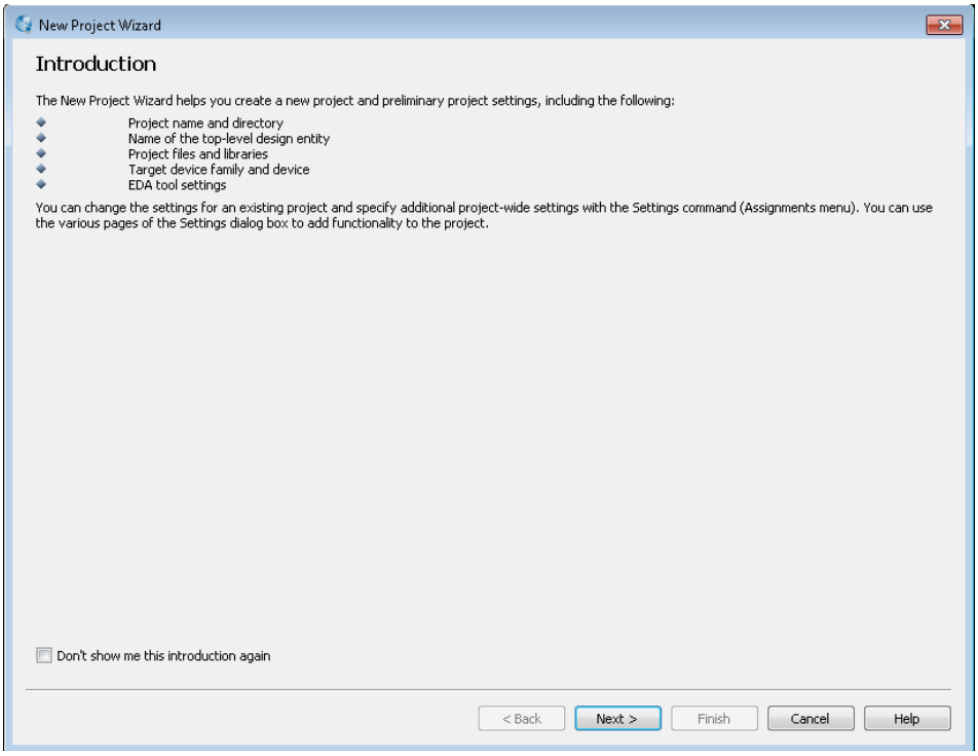


Рисунок 3.1 − Приветственное диалоговое окно проекта

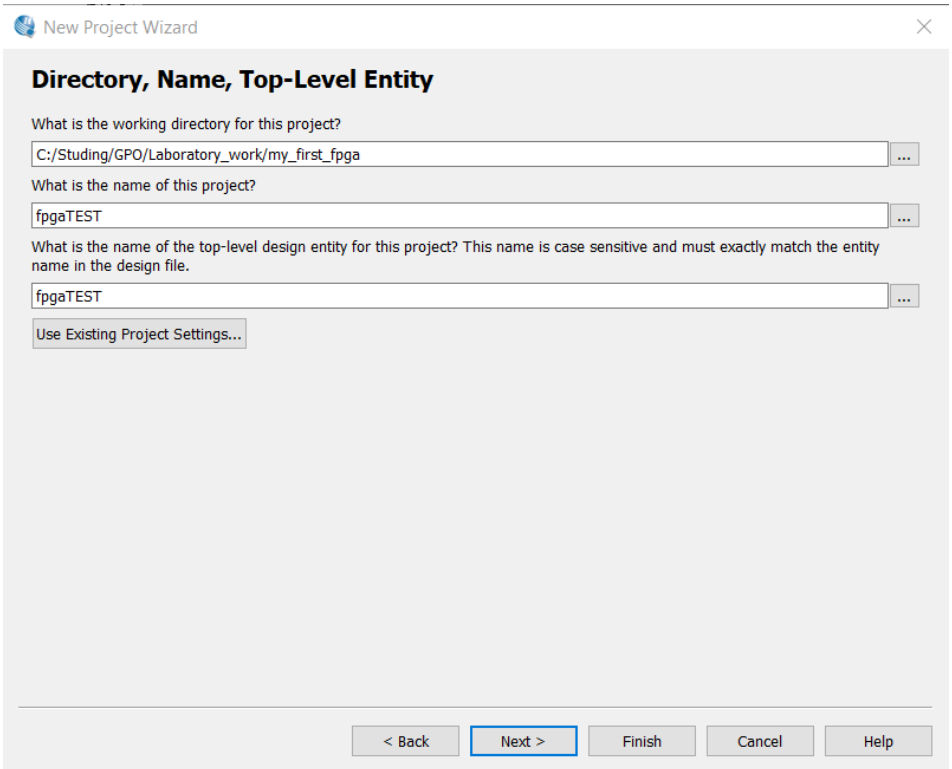


Рисунок 3.2 – Информация о проекте

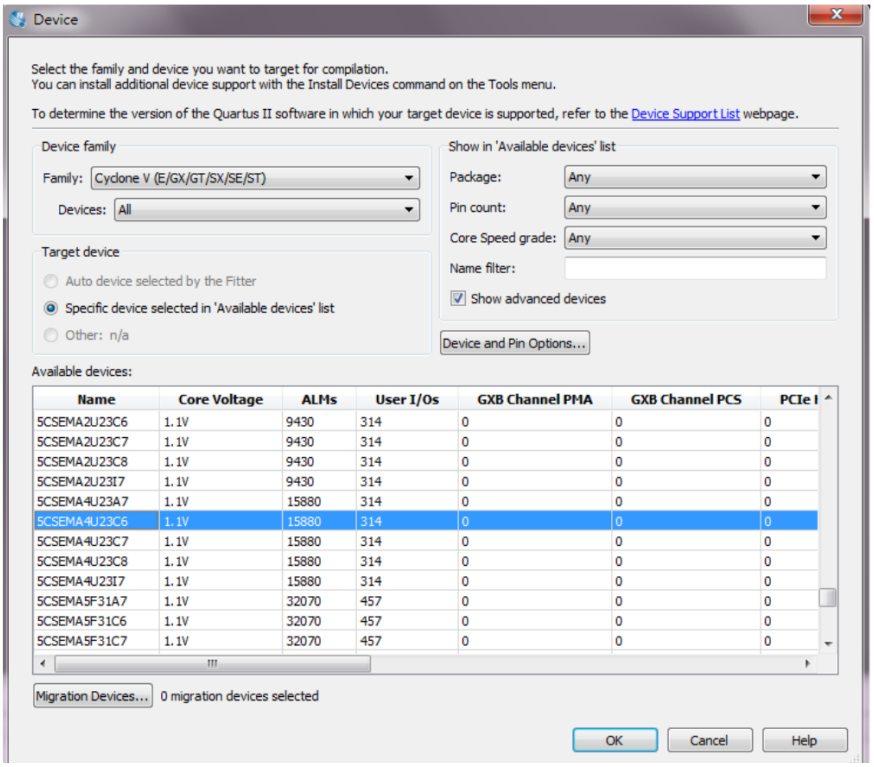


Рисунок 3.3 – Назначение устройства

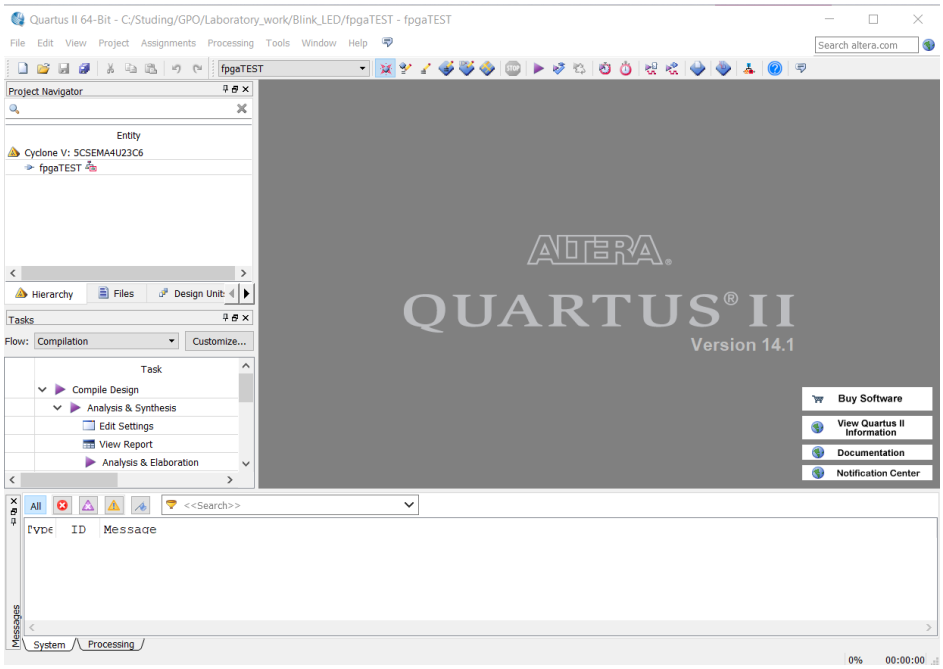


Рисунок 3.4 – Основное окно проектирования «Quartus»

3.2 Создание устройства в графическом редакторе

В данном разделе показано, как создавать цифровой устройствопри помощи встроенного графического редактора и библиотек элементов.

Для создания графического файла следует выполнить следующие действия:

1. Выбрать опцию File>New>BlockDiagram/Schematicfile, чтобы создать новый файл, появится соответствующее диалоговое окно (рисунок 3.5).

2. Нажать OK.

3. Выбрать опцию File>SaveAs и ввести в появившемся диалоговом окне (рисунок 3.6) следующую информацию:

- Имя файла, например *fpgaTEST_G*;
- Типфайла: BlockDiagram/Schematicfiles *(*.bdf)*.

4. Нажать «Сохранить». Полученный пустой файл готов для редактирования (рисунок 3.7)

5. Построить в графическом поле схему. Расположение панели инструментов, необходимых для построения схемы, указано на рисунке 3.8. Основные элементы находятся в библиотеке primitivities=>logic, входы и выходы – в библиотеке primitivities =>pin (рисунок 3.9). Доступ к редактированию имени осуществляется двойным нажатием левой кнопки мыши на имени элемента.

6. Сохранить файл, выбрав опцию File>Save или нажать комбинацию клавиш CTRL + S.

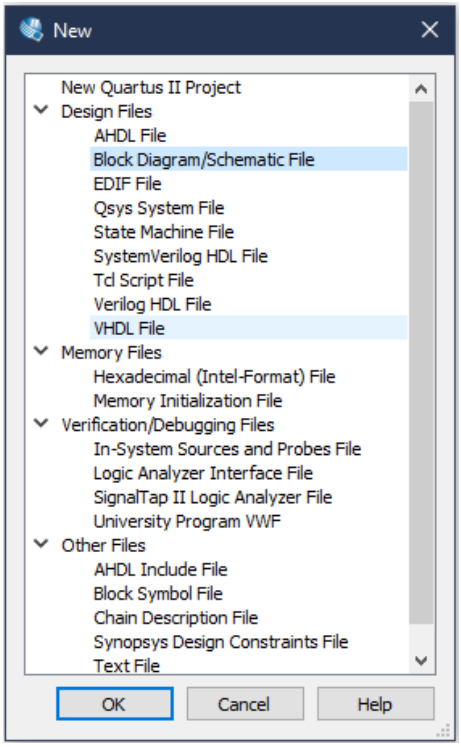


Рисунок 3.5 – Диалоговое окно создание файла

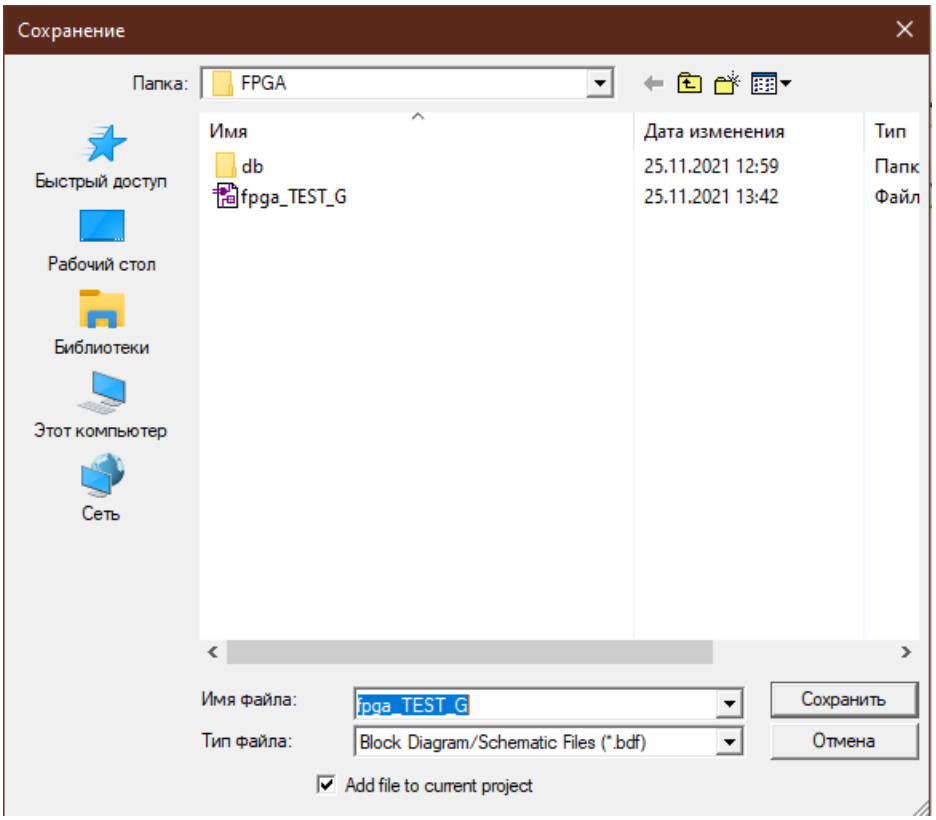


Рисунок 3.6 – Диалоговое окно сохранения графического файла

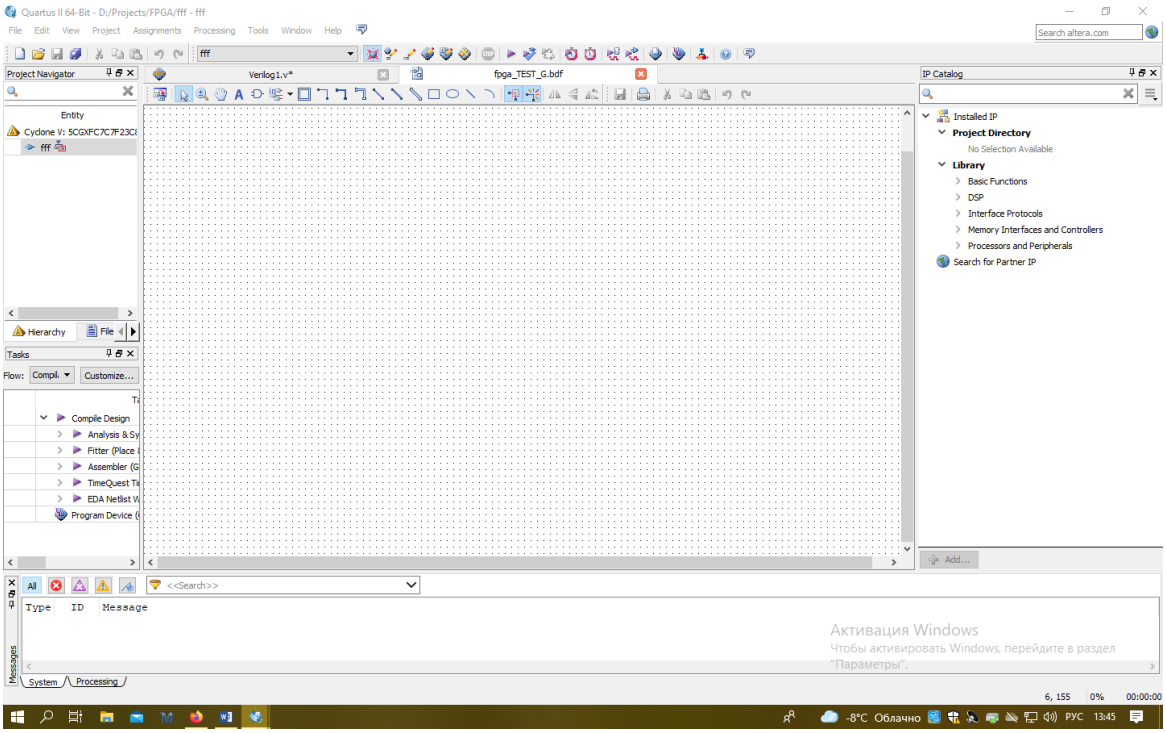


Рисунок 3.7–Диалоговое окно проекта после создания графического файла

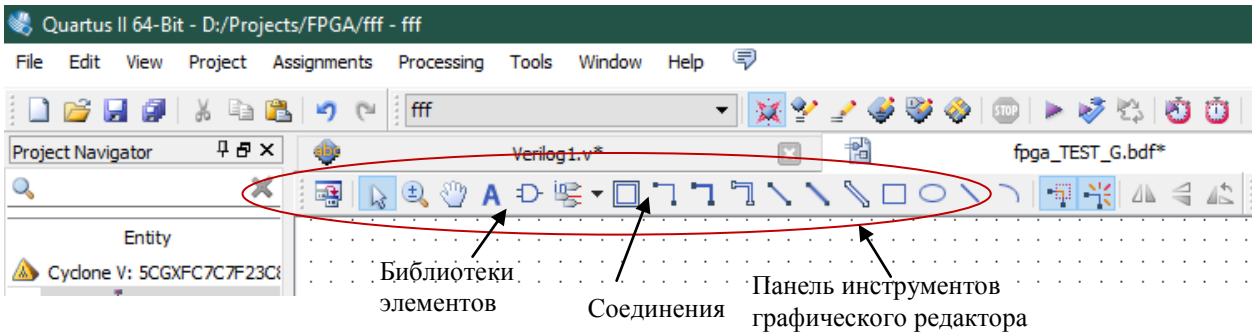


Рисунок 3.8 – Расположение панели инструментов графического редактора и основные инструменты для создания схемы.

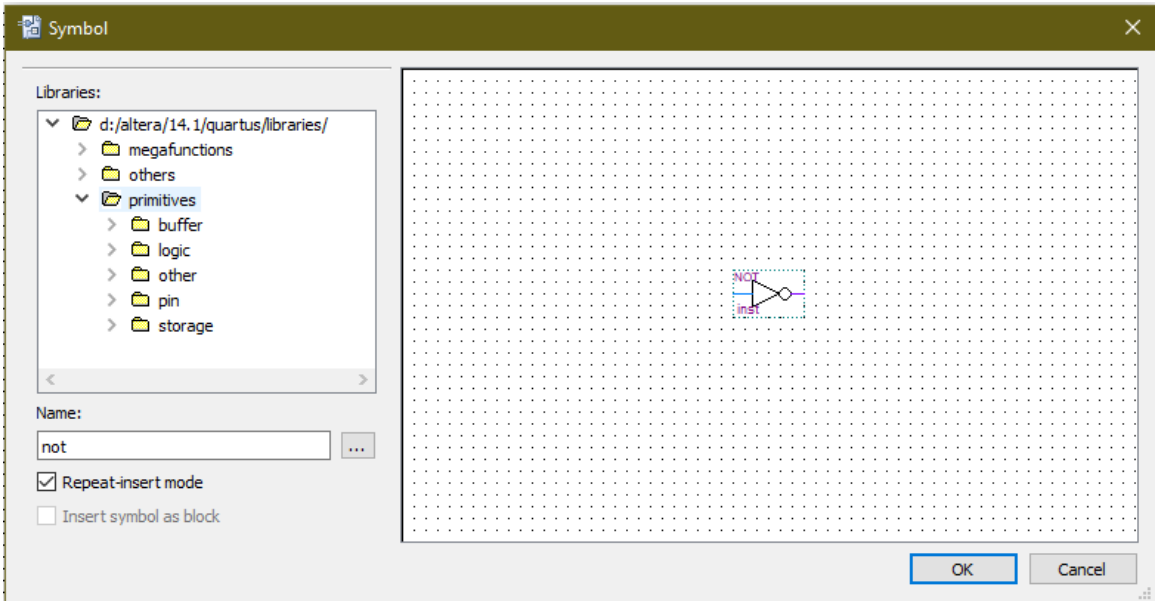


Рисунок 3.9 – Диалоговое окно библиотек элементов

3.3 Создание VHDL-файла для описания логики устройства

В данном разделе показано, как создавать цифровой устройство, применяя алгоритмический подход к проектированию, при помощи описания поведения устройства на языке VHDL.

Для создания VHDL-файла следует выполнить следующие действия:

1. Выбрать опцию File>New>VHDLFile,чтобы создать новый файл, появится соответствующее диалоговое окно (рисунок 3.10).
2. Нажать OK.
3. Выбрать опцию File>SaveAs и ввести в появившемся диалоговом окне (рисунок 3.11) следующую информацию:
   - Имя файла, например *fpgaTEST*;
   - Тип файла: *VHDLFiles (*.vhd*, .vhdl).*
4. Нажать «Сохранить». Полученный пустой файл готов для ввода кода на языке VHDL (рисунок 3.12)

5. Ввести в пустой файл *.vhd* требуемый текст описания:
6. Сохранить файл, выбрав опцию File>Save или нажать комбинацию клавиш CTRL + S.

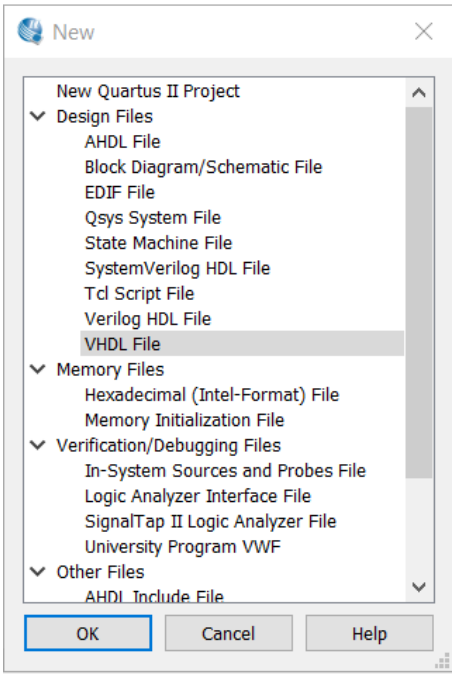


Рисунок 3.10– Диалоговое окно создание файла

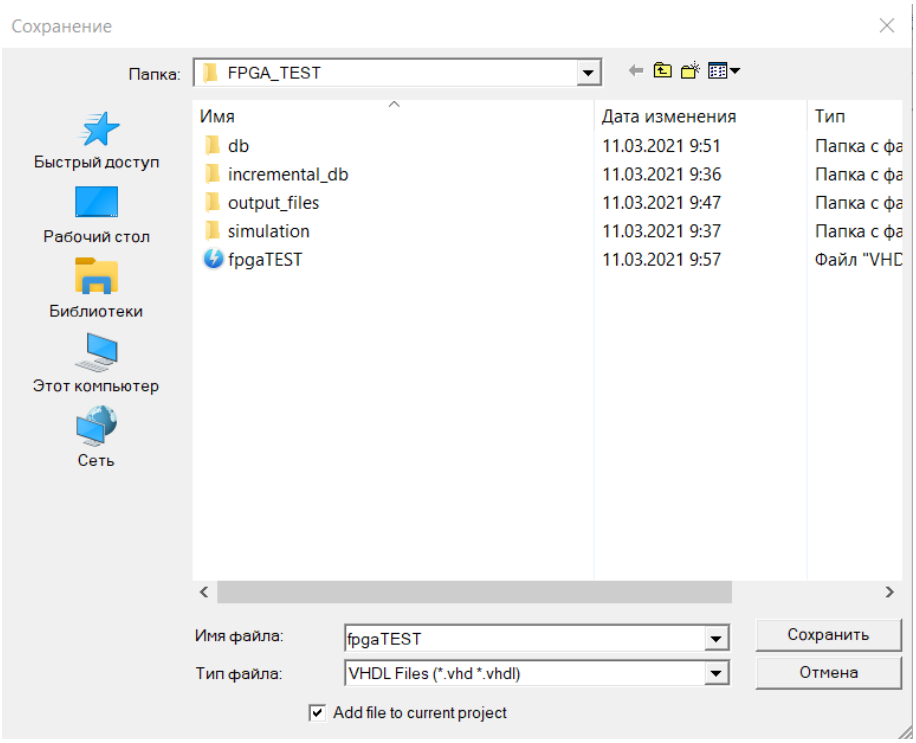


Рисунок 3.11– Диалоговое окно сохранения VHDLфайла

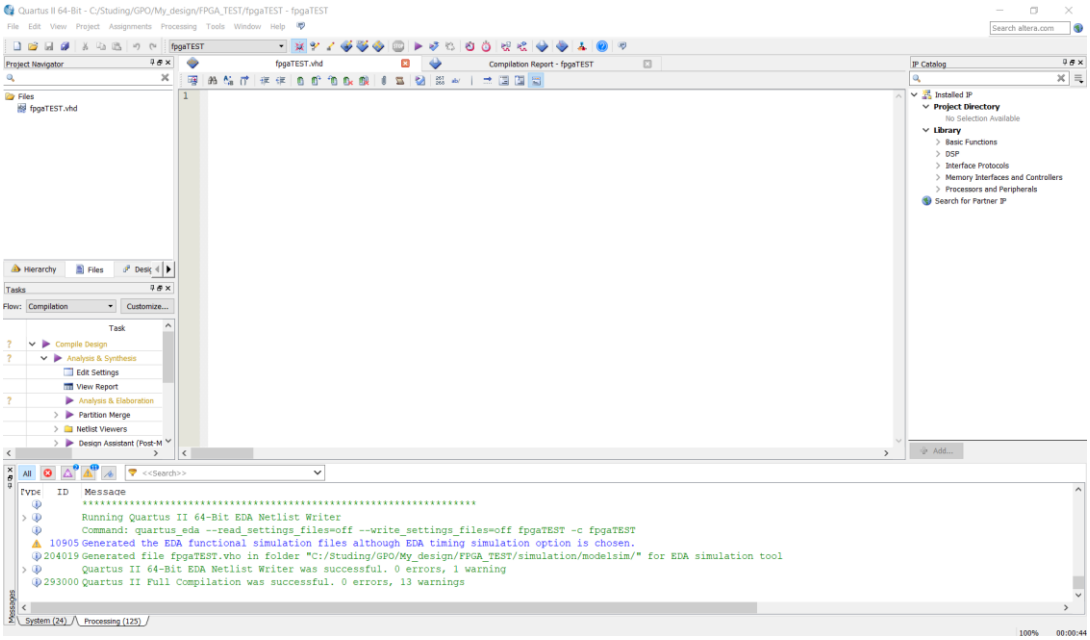


Рисунок 3.12 –Диалоговое окно проекта после создания VHDL-файла

3.4 Создание Verilog-файла для описания логики устройства

В данном разделе показано, как создавать цифровой устройство, применяя алгоритмический подход к проектированию, при помощи описания поведения устройства на языке Verilog.

1. Выбрать опциюFile>New>VerilogHDLFile, чтобы создать новый файл (рисунок 3.13).

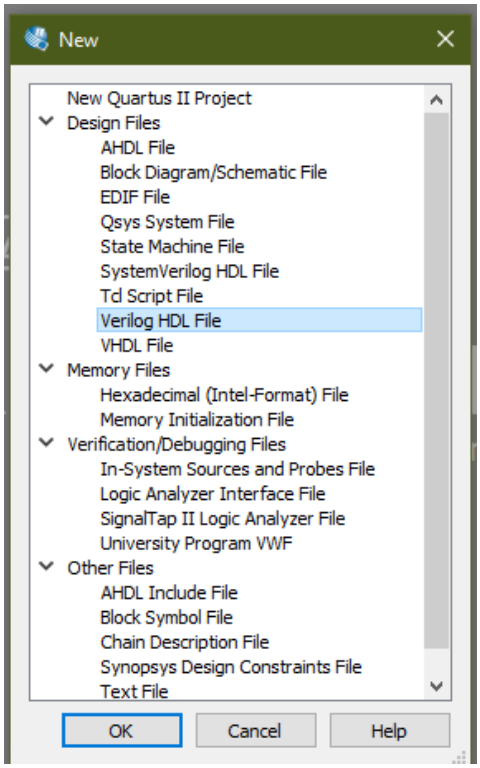


Рисунок 3.13 - Диалоговое окно создание файла

2. Нажать ОК.

3. ВыбратьFile>SaveAs и ввести появившемся окне (рисунок 3.14) следующую информацию:

- Имя файла, например *fpgaTEST_V*;
- Типфайла: *Verilog HDL Files (*.v*.vlg*.verilog).*

4. Нажать «Сохранить». Полученный пустой файл VerilogHDL готов для ввода кода.

5. Ввести в пустой файл.*v*требуемый текст описания*:*

6. Сохранить файл, выбрав опциюFile>Saveили нажать комбинацию клавиш Ctrl + S.

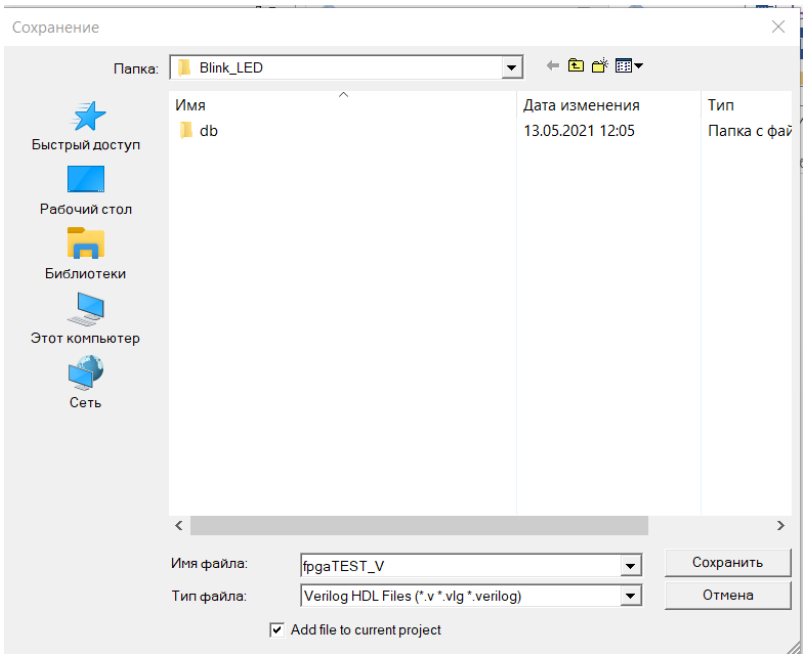


Рисунок 3.14 −Диалоговое окно сохранения VerilogHDL файла

3.5Назначение выводов микросхемы

При помощи языка описания аппаратуры или в графическом редакторе можно описать логику работы устройства, указав имена входных и выходных сигналов. Эти имена являются лишь логическими и не привязаны к конкретным выводам микросхемы. Для назначения требуемых функций выводам микросхемы в среде «Quartus» есть отдельный инструмент PinPlanner, который описан в данном разделе. Перед назначением выводов выполните следующие действия:

1. Выбрать Processing>Start>StartAnalysis&Elaboration для подготовки к назначению расположения выводов;

2. Нажать кнопку ОК в окне сообщения, которое появится после завершения анализа и разработки.

3. Выбрать Assignments>Pin Planner, после чего откроется планировщик выводов (Рисунок 3.15).

4. Рядом с каждым из двух имен узлов добавить координаты (имя вывода), как показано в Таблице 3.1, в которой отображены фактические значения выводов, используемые исходной отладочной платой DE0-Nano-SoC.

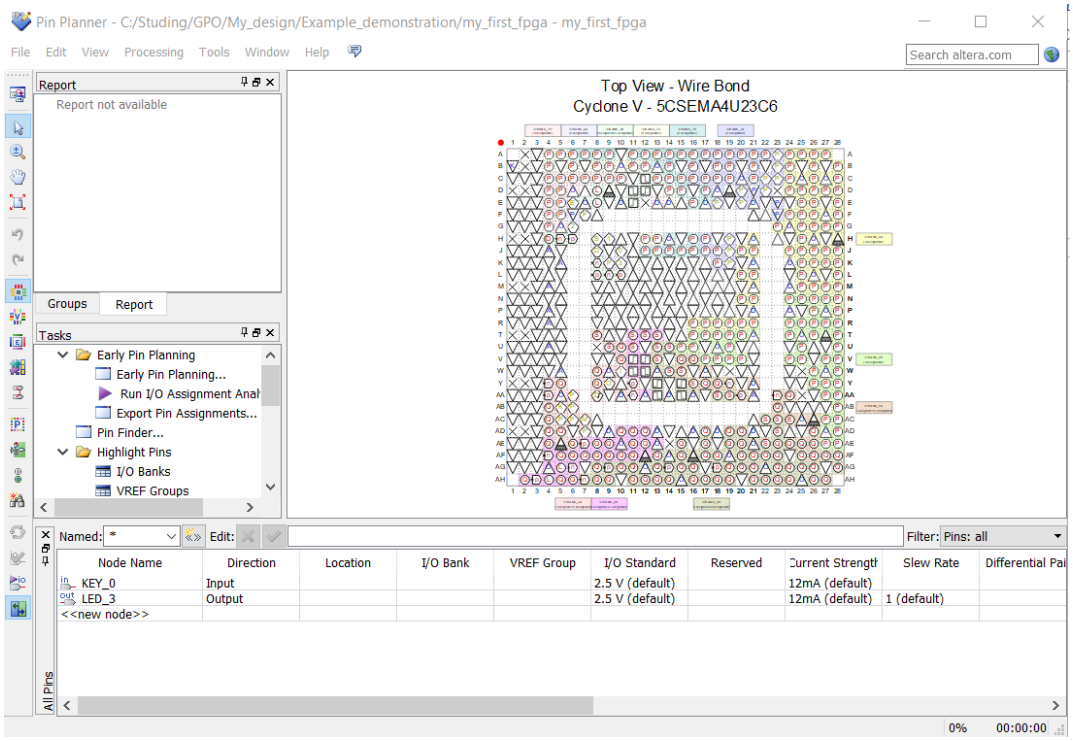


Рисунок 3.15 – Планировщик выводов (thePinPlanner)

Таблица 3.1 – Пример информации о фактических значениях выводов

Имя вывода	Расположение выводов на ПЛИС
KEY[0]	AH17
LED[3]	V15

Чтобы назначить фактическое расположение конкретного вывода, нужно дважды щелкнуть левой кнопкой мыши в столбце «Location», после чего откроется раскрывающийся список, из которого выбирается имя физического вывода (рисунок 3.16).

Также необходимо указать, как будут определены неиспользуемые выводы. Для этого нужно выбрать опцию Assignments>Device и в открывшемся окне нажать кнопку DeviceandPinOptions (Рисунок 3.17).

После в открывшемся окне Device and Pin Options слева нужно выбрать вкладку Unused Pins, а в графе Reserve all unused pins – Asinputtri-statedoption (Рисунок 3.18).

Далее нужно нажать OK дважды.

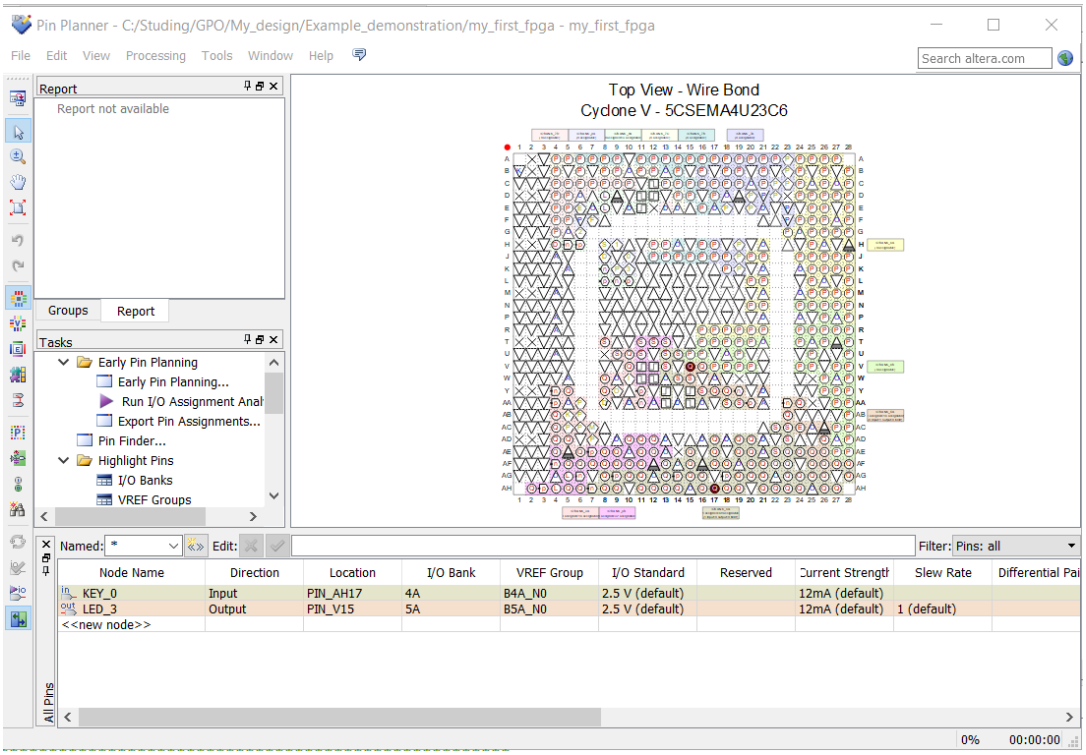


Рисунок 3.16– Окно Pin Planner после назначения выводов

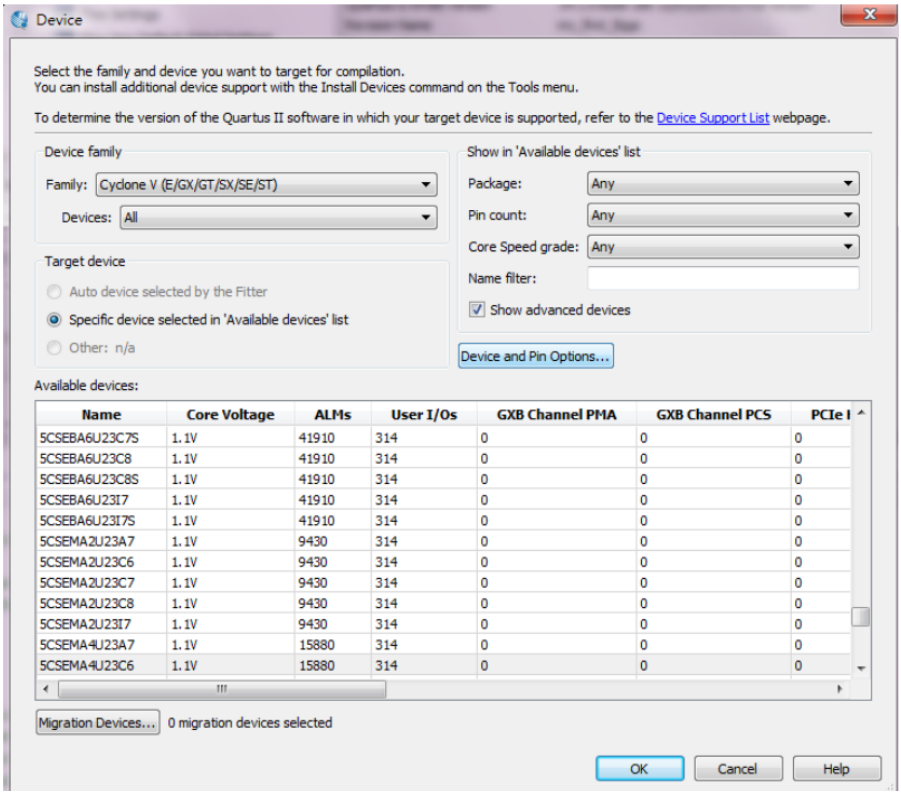


Рисунок 3.17 – Окно Device and Options

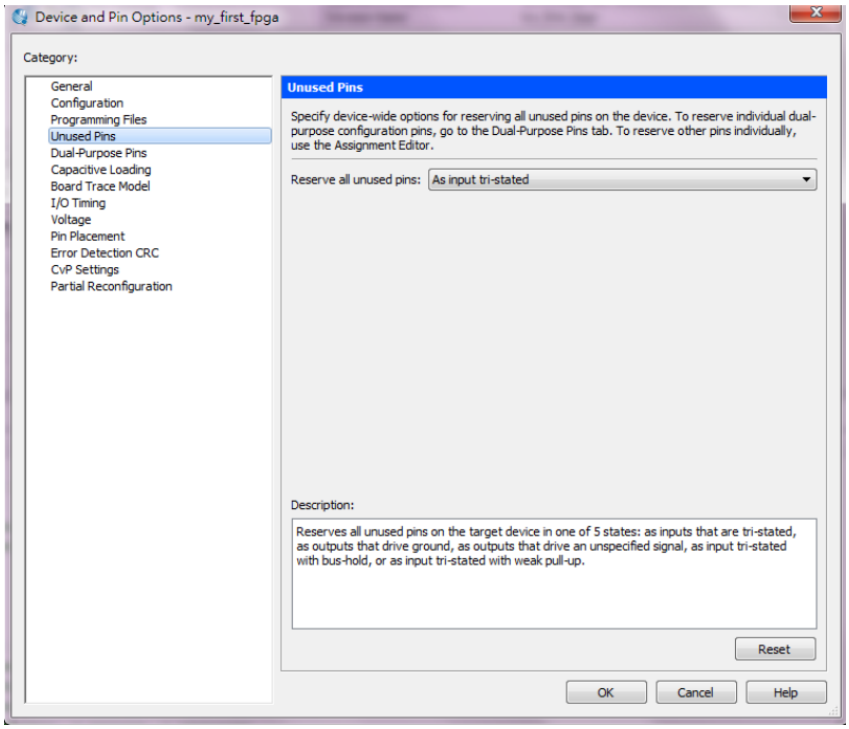


Рисунок 3.18 – Окно Device and Pin Options

3.6 Компилирование проекта

После того, как был создан полный проект и произведено назначение выводов, его необходимо скомпилировать. Для этого необходимо выбрать опцию Processing=>StartCompilation или кнопку Play на панели инструментов. Если «Quartus» спросит, сохранить ли изменения в файле, нужно нажать кнопку Yes.

В процессе компиляции проекта «Quartus»отображает информацию об выполняемых этапах, ошибках и предупреждениях (Рисунок 3.19).

Когда компиляция завершится, «Quartus» выводит сообщение. Нужно нажатьOK, чтобы закрыть диалоговое окно.

В окне «Quartus» *Messages* в случае правильно составленного проекта не должно отображаться никаких критических предупреждений (critical warnings).

*Примечание. «*Quartus» может отображать несколько предупреждений, указывающих на то, что информация о времени устройства является предварительной или что некоторые параметры на выводах (the I/O pins), используемых для светодиодов (LEDs), не были установлены.

Программное обеспечение «Quartus» предоставляет результаты компиляции на вкладке CompilationReport, как показано на рисунке 3.20.

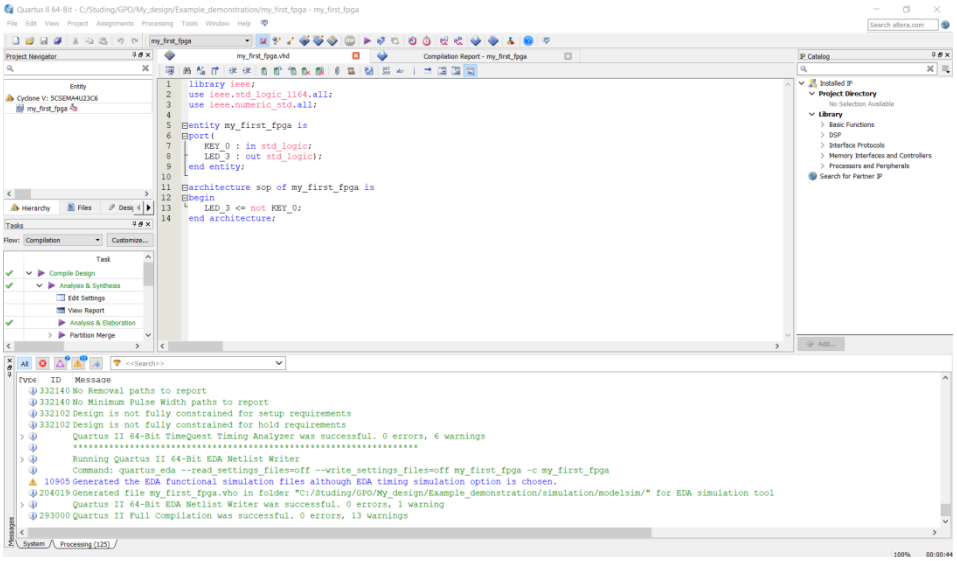


Рисунок 3.19 – Компиляция проекта

Flow Summary

Flow Status	Successful - Thu Mar 11 11:57:47 2021
Quartus II 64-Bit Version	14.1.0 Build 186 12/03/2014 SJ Web Edition
Revision Name	my_first_fpga
Top-level Entity Name	my_first_fpga
Family	Cyclone V
Device	5CSEMA4U23C6
Timing Models	Final
Logic utilization (in ALMs)	1 / 15,880 (< 1 %)
Total registers	0
Total pins	2 / 314 (< 1 %)
Total virtual pins	0
Total block memory bits	0 / 2,764,800 (0 %)
Total DSP Blocks	0 / 84 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 5 (0 %)
Total DLLs	0 / 4 (0 %)

Рисунок 3.20 − Окно «*CompilationReport*»

3.7 Запись программы в память отладочной платы

После компиляции и проверки проекта нужно запрограммировать (прошить) отладочную плату. Для этого необходимо загрузить файл с расширением .sof (который создается после компиляции и проверки проекта) в плату, используя вход USB-Blaster II. Чтобы настроить оборудование для программирования, нужно выполнить следующие действия:

a) Подключить кабель питания к плате и к электрической розетке;

b) ПодключитьUSB-minikJ13,другой конец USB-кабеля – к компьютеру.

На рисунке 3.21 показано правильное подключение кабеля питания и USB кабеля к отладочной плате DE0-Nano-SoC board.

Для программирования платы ПЛИС нужно выполнить следующие действия:

1. Выбрать опцию Tools>Programmer. Откроется окно программатора (TheProgrammer), которое изображено на рисунке 3.22.

2. НажатьHardware Setup.

3. Выбратьопцию DE0-Nano-SoC [USB-1] в разделе «Currently selected hardware» (рисунок 3.23).

4. Нажать Close.

5. Нажать кнопку AutoDetect, чтобы обнаружить все устройства в цепочке JTAGchain.

6. Выбрать5CSEMA4 и нажатьOK (рисунок 3.24).

7. И HPS, и FPGA будут перечислены на программаторе. Выбрать устройство FPGA (т.е. 5CSEMA4) и нажать кнопку ChangeFile, чтобы изменить .sof-файл.

8. Выбрать файл fpgaTEST.sof из каталога проекта (Рисунок 3.25, 3.26).

9. Поставить галочку в пункте Program/Configure и нажать кнопку Start, чтобы запрограммировать .sof-файл внутрь платы (Рисунок 3.27).

10. Скомпилированный объектный файл SRAM (.sof) загружается на плату.

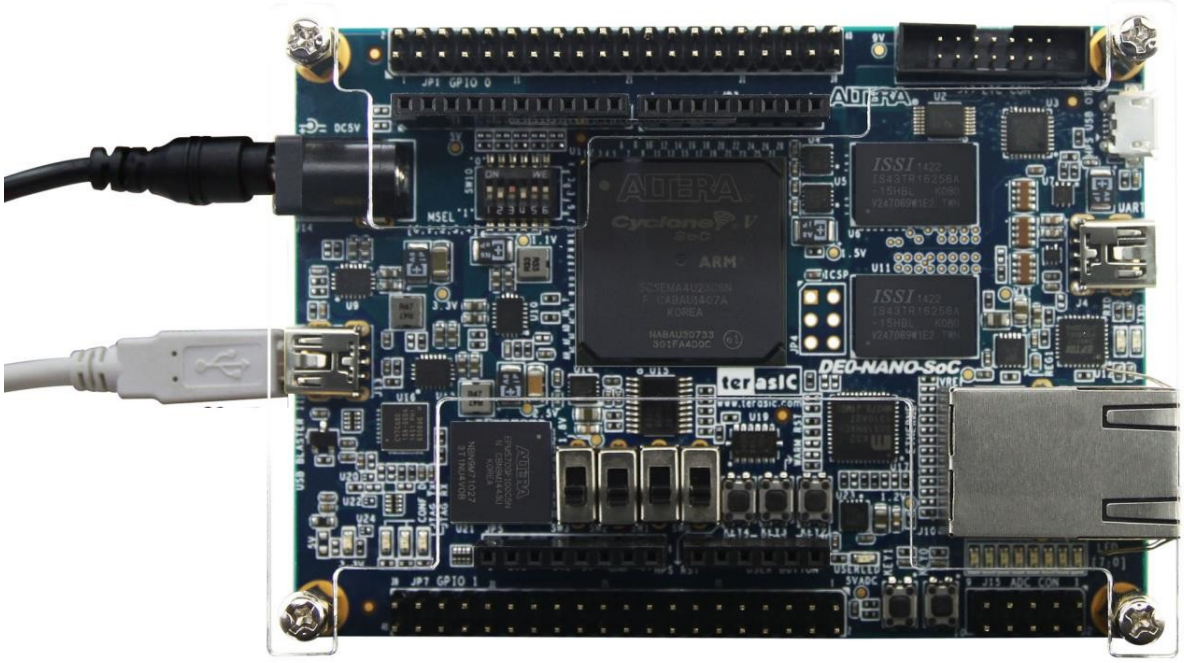


Рисунок 1.21– Подключение кабеля питания и USB к плате

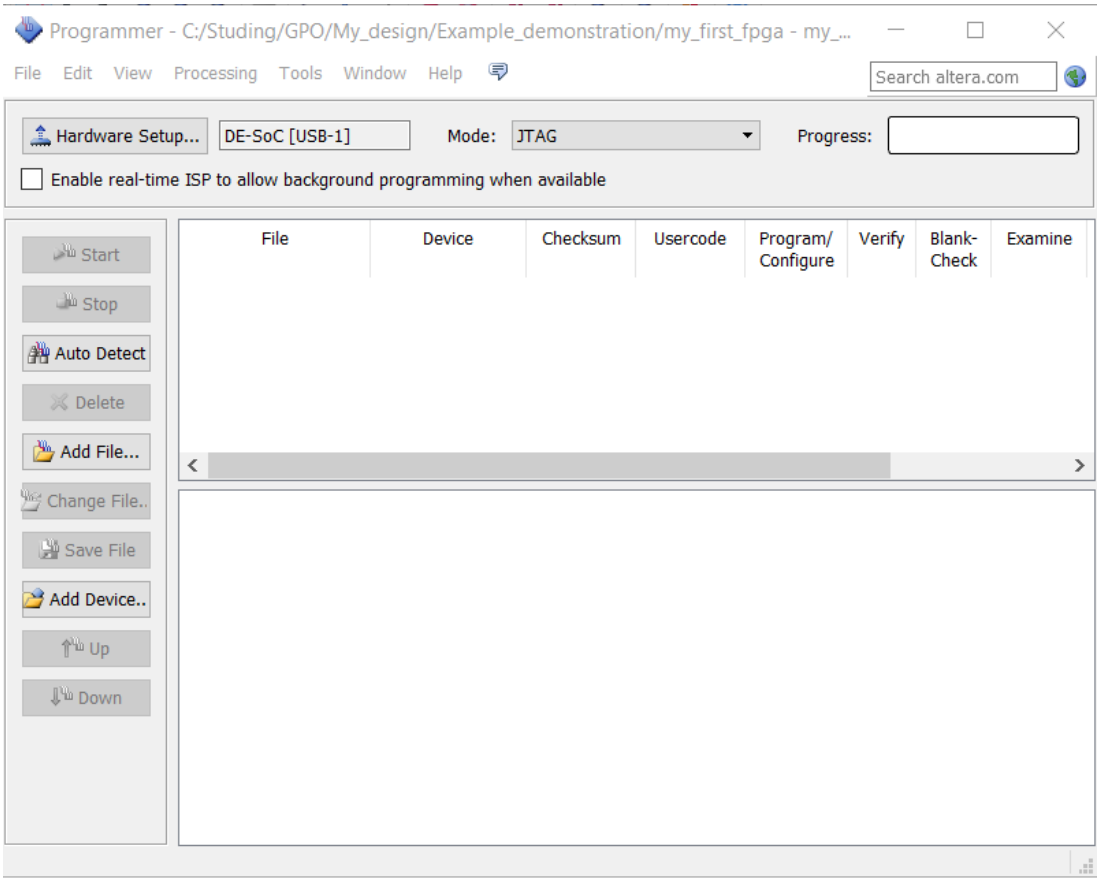


Рисунок 3.22 – Диалоговое окно «*Programmer*»

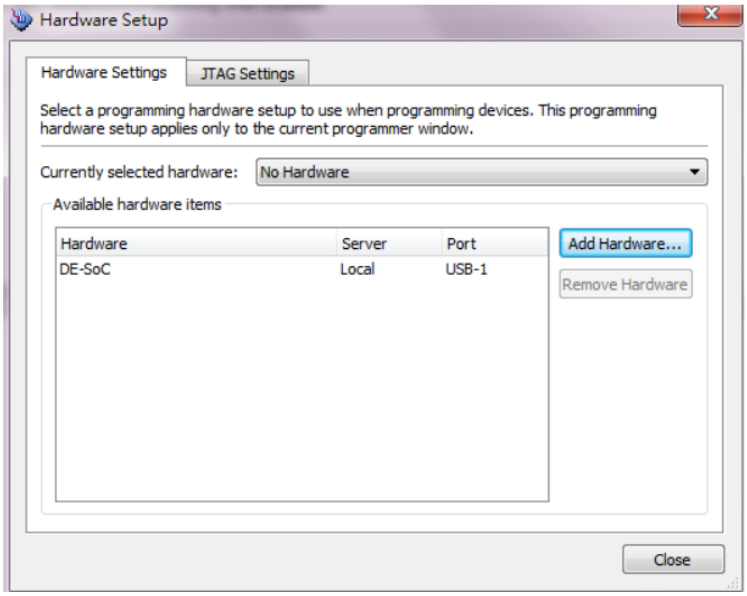


Рисунок 3.23 – Окно Hardware Setup

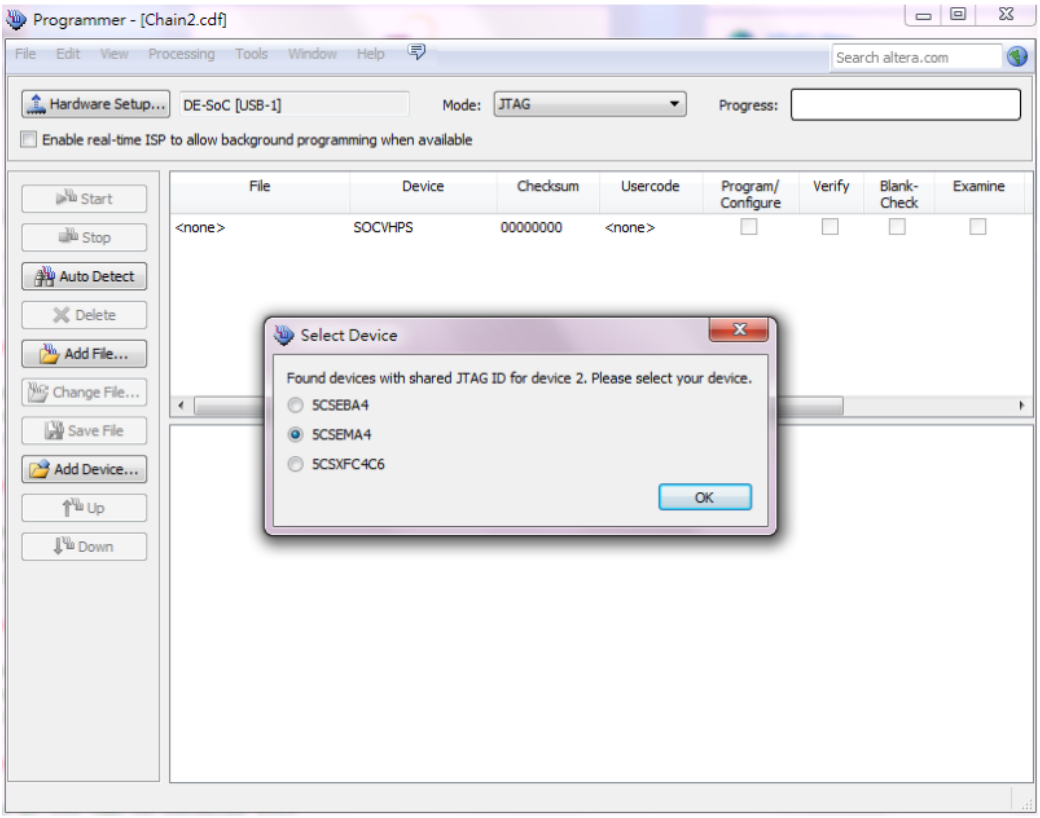


Рисунок 3.24 – Окно Select Device

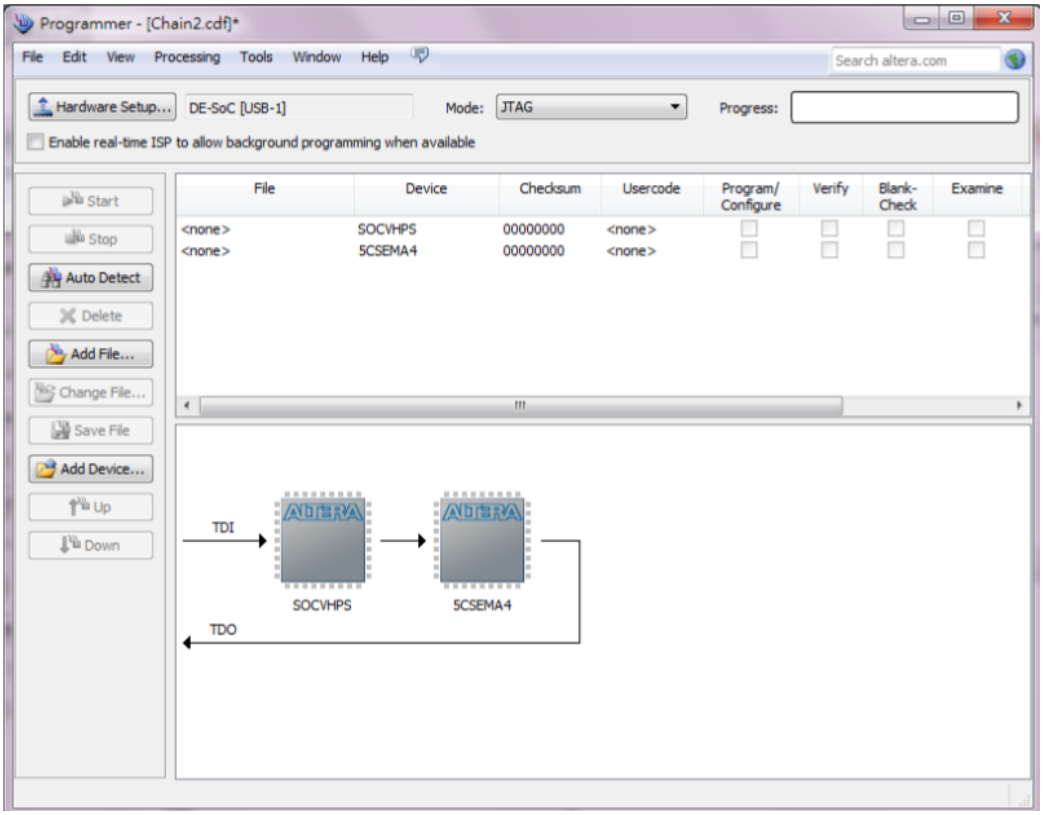


Рисунок 3.25 – Диалоговое окно «*Programmer*» после выбора программируемого устройства

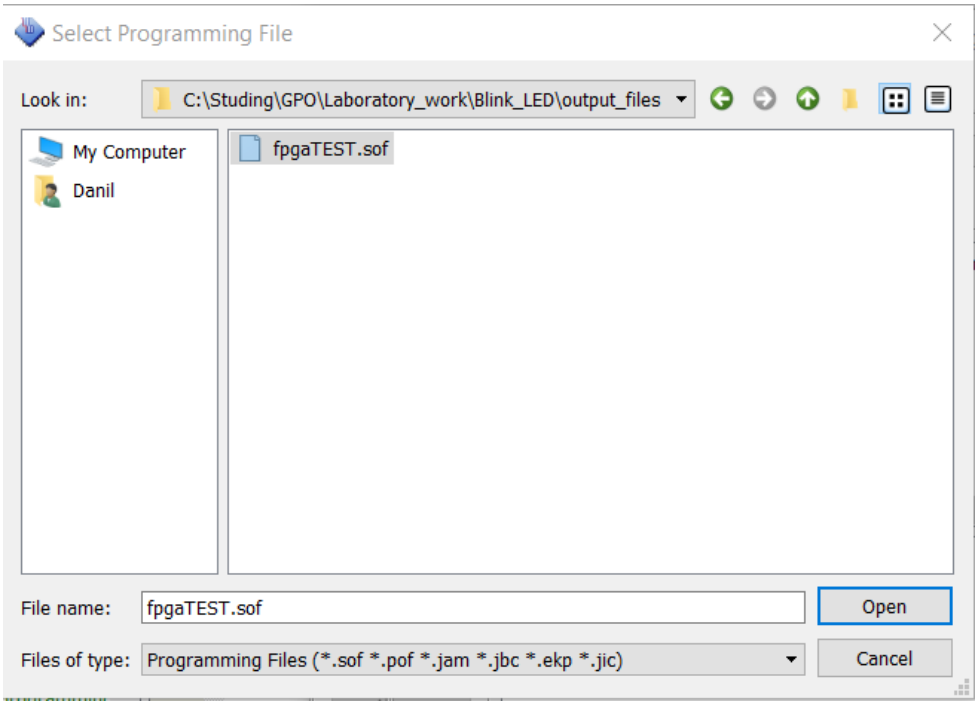


Рисунок 3.26 – Выбор файла конфигурации .sof

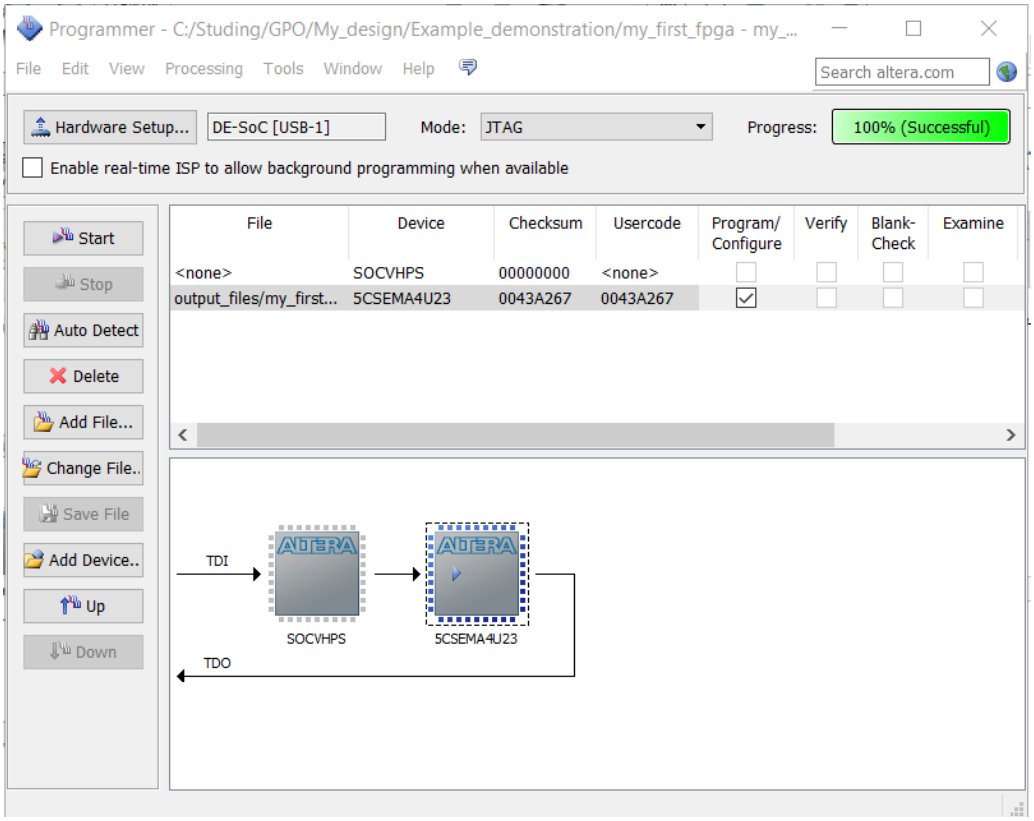


Рисунок 3.27 – Окно Programmer после завершения прошивания платы

4 ЛАБОРАТОРНАЯ РАБОТА №1«ПОЛУЧЕНИЕ ПЕРВОНАЧАЛЬНЫХ НАВЫКОВРАБОТЫС ПРОГРАММИРУЕМЫМИ ЛОГИЧЕСКИМИИНТЕГРАЛЬНЫМИ СХЕМАМИ»

4.1 Введение

Цель работы: ознакомление со средой разработки «Quartus» и создание первого проекта на ПЛИС.

Для получения первоначальных навыков работы с ПЛИС в работе рассматривается проектирование наиболее простых логических элементов.

Далее в работе рассмотрены три способа описания цифрового устройства, каждый из которых имеет свои достоинства и недостатки: графический, описание на языке VHDLи описание на языке Verilog.

Графический способ является наиболее наглядным и предпочтителен для описания цифровых схем, состоящих из небольшого количество типовых блоков любой сложности. Недостатком является то, что данный способ реализует только структурный подход к разработке устройств.

Язык описания аппаратуры VHDLявляется строго типизированным и имеет достаточно наглядный набор команд и операторов, что делает его наглядным и наиболее подходящим для знакомства с алгоритмическим подходом к проектированию аппаратуры. Недостатком является большое количество служебных ключевых слов, которые сильно загромождают код при описании сложных устройств.

Язык описания аппаратуры Verilogпо сравнению с VHDLявляется менее строгим, многие команды в нем имеют краткое символьное обозначение, что значительно сокращает объем текста, особенно при описании сложных устройств. Недостатком по сравнению с VHDLявляется необходимость наличия более высокой квалификации разработчикадля чтения и отладки кода.

4.2Порядок выполнения работы

4.2.1 Необходимо создать наиболее простое логическое устройство – инвертор (элемент НЕ), каждым из трех способов: средствами графического редактора, описанием на языке VHDLи описанием на языке SystemVerilog. Все три способа должны быть в разных проектах. Вход элемента следует назвать Key_0, выход – Led_3.

4.2.1.1 Для создания устройства средствами графического редактора нужно создать новый проект, потом создать графический файл и построить схему (см. п. 3.2), изображенную на рисунке 4.1:

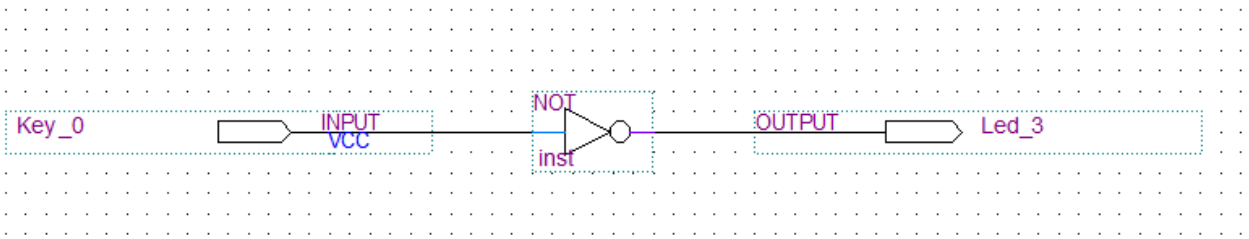


Рисунок 4.1 – Схема инвертора в графическом редакторе «Quartus»

4.2.1.2 Для создания устройства на языке VHDL нужно создать новый проект, потом создать VHDL-файл (см. п. 3.3)и ввести следующий текст:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fpga TEST is
port(
        KEY_0 :in std_logic;
        LED_3 :out std_logic);
endentity;

architecture sop of fpgaTEST is
begin
        LED_3 <=not KEY_0;
endarchitecture;
```

Примечание.Согласно правилам языка VHDL, наименование структуры *entity*должно совпадать с названием самого файла. В данном случае файл имеет название *fpgaTEST.vhd*, поэтому в *entity*пишется *fpgaTEST*.

4.2.1.3 Для создания устройства на языке VHDL нужно создать новый проект, потом создать Verilog-файл (см. п. 3.4)и ввести следующий текст:

```
module fpfaTEST_V (key0, led_3);
input key_0;
output led_3;

assign led_3=~key_0;
endmodule;
```

4.2.2 Назначить входы и выходы(см. п. 3.5). Информацию о соответствии кнопок и светодиодов можно получить в руководстве пользователя отладочной платы DE0-Nano-SoCBoard или у преподавателя.

4.2.3 Скомпилировать проект (см. п. 3.6)

4.2.4 Запрограммировать ПЛИС (см. п. 3.7)

4.2.5 Проверить правильность работы аппаратного обеспечения платы:

4.2.5.1 При нажатии кнопки KEY[0] на плате светодиод LED[3] должен загореться.

4.2.5.2 Все остальные рабочие ранее светодиоды на плате должны потухнуть.

4.3 Контрольные вопросы

1. Что такое программируемая логическая интегральная схема?
2. В чем заключается структурный подход к проектированию?
3. В чем заключается алгоритмический подход к проектированию?
4. В чем заключаются основные особенности языков описания аппаратуры по сравнению языками программирования высокого уровня для микропроцессоров?
5. В чем заключаются основные отличия языков VHDL и System Verilog?
6. Каким образом устанавливается соответствие между сигналами и физическими выводами ПЛИС?

5 ЛАБОРАТОРНАЯ РАБОТА №2 «СОЗДАНИЕ И ОТЛАДКА КОМБИНАЦИОННОГО ЛОГИЧЕСКОГО УСТРОЙСТВА»

5.1 Введение

Цель работы: синтез комбинационных схем и их реализация в ПЛИС средствами графического редактора и при помощи языков описания аппаратуры, а также моделирование их работы на ПЛИС с использованием элементов ввода и световой индикации.

Комбинационные логические устройства (*комбинационные схемы*) –это логические устройства, состояние выхода которых однозначно определяется набором входных сигналов. Это отличает комбинационную логику от последовательной логики, в рамках которой выходное значение зависит не только от текущего входного воздействия, но и от предыстории функционирования цифрового устройства, что предполагает наличие памяти, которая в комбинационной логике не предусмотрена. Комбинационная логика имеет очень широкое применение, в частности используется в вычислительных и коммуникационных системах для синтеза цифровых сигналов, для подготовки данных, которые подлежат сохранению, для коммутации цифровых сигналов, для реализации арифметических функций и многого другого. На практике сложные цифровые устройства обычно сочетают комбинационную и последовательную логику.

Математическим аппаратом для описания логических функций является булева алгебра. Логические функции могут быть как однозначными, так и многозначными, фактически представленными набором однозначных функций. Также логические функции могут быть полными, когда всем наборам значений аргументов соответствует определенное значение функции, и неполными, когда некоторым наборам значений аргументов могут соответствовать любые значения функции. В данной лабораторной работе рассматриваются полные однозначные функции.

Одним из характерных свойств логических функций является ярко выраженный полиморфизм, то есть одна и та же функция может быть записана множеством способов. В связи с этим одной из наиболее важнейших задач в процессе синтеза логических функция является их минимизация, то есть поиск такой формы записи, которая бы содержала в себе минимум аргументов и математических действий. Наиболее универсальным является метод Квайна, он применим к функциям с любым количеством аргументов, легко формализуется и автоматизируется, однако при минимизации вручную является весьма трудоемким и требует высокой степени внимательности, причем трудоемкость в зависимости от количества аргументов растет в геометрической прогрессии. Альтернативой методу Квайна является метод карт Карно или диаграмм Вейча, он является более наглядным и простым, однако применим при числе аргументов не более 4-х, что, впрочем, достаточно для решения широкого круга задач.

5.2 Ход работы

5.2.1 По варианту задания, данного в виде таблицы истинности (таблица5.1), необходимо минимизировать логическую функцию по заданному базису любым известным методом, для нечетных вариантов используется базисИ-НЕ, для четных – ИЛИ-НЕ.

Таблица 5.1 – Варианты заданий для синтеза комбинационной логической схемы.

N	Входные переменные				Варианты функцииf (x_1, x_2,x_3,x_4)																		
	x_1	x_2	x_3	x_4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1
2	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0	1	0	0	0	1	1
3	0	0	1	0	1	1	0	1	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	0
4	0	0	1	1	0	1	0	0	1	1	1	0	1	0	0	0	1	0	1	0	1	0	1	1
5	0	1	0	0	1	1	1	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	0
6	0	1	0	1	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	0	1	0	0	1
7	0	1	1	0	1	0	0	1	1	0	0	1	1	1	0	1	0	0	0	1	0	1	0	0
8	0	1	1	1	0	0	1	0	1	0	0	0	1	1	1	0	1	0	0	0	1	0	1	0
9	1	0	0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	1	1	0	0	1	0	1
10	1	0	0	1	0	0	1	0	1	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1
11	1	0	1	0	1	1	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	0	1	0
12	1	0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	1	1	1	0	1	0	1	1
13	1	1	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
14	1	1	0	1	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	1	1	1	0	1
15	1	1	1	0	1	0	0	1	0	0	0	1	1	1	0	1	1	0	0	1	0	1	1	0
16	1	1	1	1	0	0	1	0	1	0	0	0	1	0	1	0	1	0	0	1	1	1	1	0

5.2.2 Функция приводится к базису, для нечетных вариантов – И-НЕ, для четных – ИЛИ-НЕ.

5.2.3В графическом редакторе среды программирования «Quartus» рисуется схема и указываются наименования входных и выходных сигналов (см. пример на рис. 5.1) Основные логические элементы можно найти в библиотеке C:/ altera/ 14.1/ quartus/ libraries-primitivities-logic, а сам процесс расстановки элементов и установки соединений подробно рассмотрен в лабораторной №1 «Введение в программируемые логические интегральные схемы». Большинство программных пакетов, предназначенных для моделирования электронных схем, не поддерживают условно-графические обозначения, принятые в российских стандартах. Ниже приведено таблица соответствие между обозначениями, принятыми по ГОСТ 2.743-91 и применяемыми в среде разработки«Quartus» (таблица 5.2).

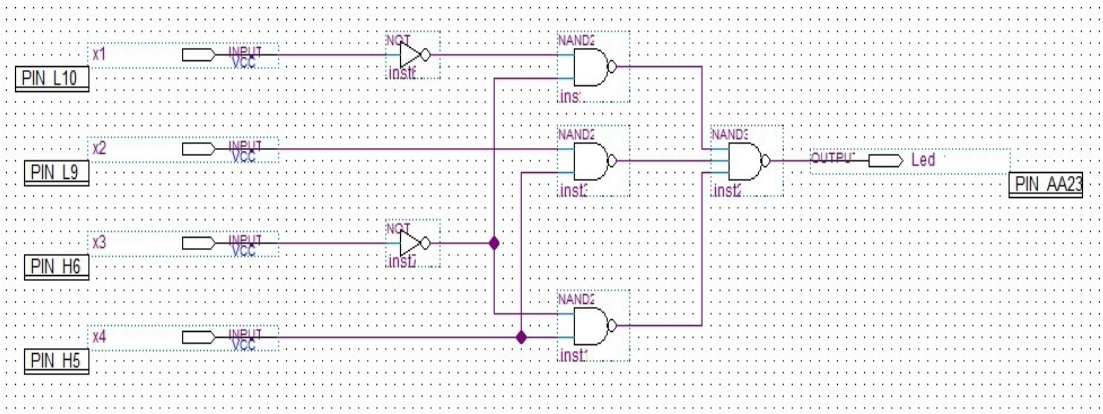
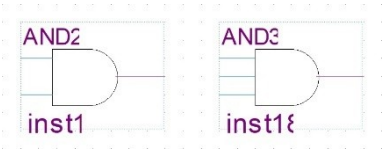
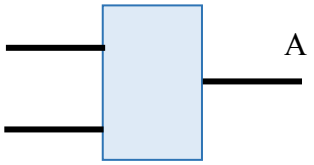
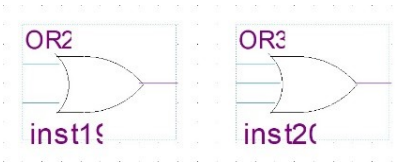
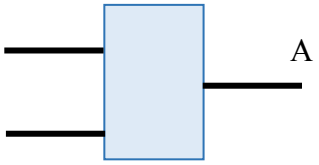
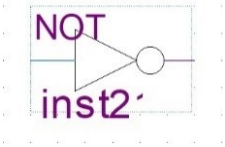
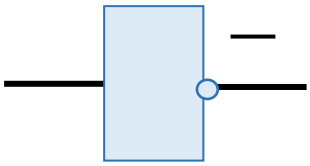
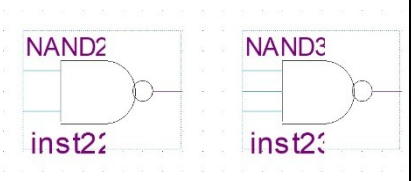
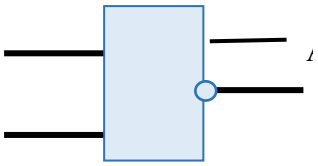
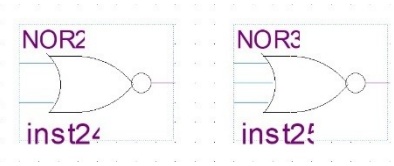
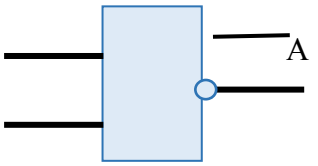
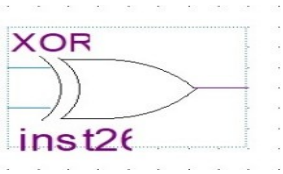
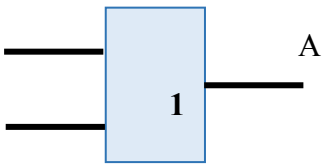
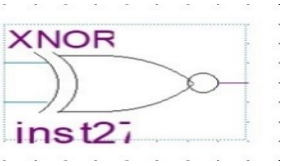
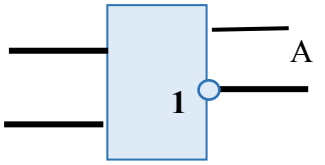


Рисунок 5.1 – Пример комбинационной схемы в «Quartus»

Таблица 5.2 – Типовые логические элементы и их условно-графические обозначения

Логические элементы	Графическое изображение всреде «Quartus»	Графическое изображение по ГОСТ 2.743-91
Конъюнктор (И)		
Дизъюнктор (ИЛИ)		
Инвертор (НЕ)		
И-НЕ		
ИЛИ-НЕ;		
Исключающее ИЛИ		
Эквивалентность (исключающее ИЛИ-НЕ).		

5.2.4 Назначается соответствие сигналов на схеме физическим выводам микросхемы при помощи инструмента Pin_Planner (рис. 5.2). Входные на отладочной плате формируются при помощи переключателей SW[0..4], выходные сигналы отображаются на светодиодах LED[0..7]. Более подробную информацию о соответствии переключателей и светодиодов физическим выводам микросхемы AlteraCycloneV можно найти в руководстве пользователя отладочной платы DE0-Nano-SoC, которое выдается преподавателем или доступно в электронном виде.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location
Led_1	Output	PIN_AA23	5A	B5A_N0	PIN_AA23
x1	Input	PIN_L10	8A	B8A_N0	PIN_L10
x2	Input	PIN_L9	8A	B8A_N0	PIN_L9
x3	Input	PIN_H6	8A	B8A_N0	PIN_H6
x4	Input	PIN_H5	8A	B8A_N0	PIN_H5
<<new node>>					

Рисунок 5.2 – Таблица соответствия сигналов и выводов в «Quartus»

5.2.5 Полученная схема записывается в ПЛИС, процесс программирования описан в лабораторной работе№1«Введение в программируемые логические интегральные схемы»

5.2.6 При помощи отладочной платы проверяется правильность построения логической функции, переключателями задается комбинация входных переменных, светодиоды отображают соответствующие сигналы на выходе.

5.2.7 Минимизированная функция описывается на языках VHDLи Verilog, повторяются пункты 5.2.4 – 5.2.6. Информацию о языках описания аппаратуры и примеры написания логических функций можно найти в литературе [1–3].

5.3 Контрольные вопросы

1.Что такое комбинационная схема?

2. Для чего нужно минимизировать логические функции?

3. Какими методами можно минимизировать логические функции?

4. Что значит минимизировать функцию по базису?

5. Как обозначается в графическом редакторе «Quartus» основные логические элементы?

6. Как обозначаются основные логические функции в языке VHDL?

7. Как обозначаются основные логические функции в языке Verilog?

6 ЛАБОРАТОРНАЯ РАБОТА №3 «СОЗДАНИЕ И ОТЛАДКА СЧЕТЧИКА»

6.1 Введение

Цель работы: синтез двоичного счетчика с предустановкой и его реализация в ПЛИС средствами графического редактора и при помощи языков описания аппаратуры счётчика, а также моделирование его работы на ПЛИС с использованием элементов ввода и световой индикации.

Счётчик является одним из наиболее важных узлов цифровой аппаратуры. Основной его функцией является подсчет количества импульсов, но в вместе с тем на его основе можно реализовать функции отсчета времени, деления частоты,формирования цифровых сигналов с заданными частотно-временными параметрами, измерения временных интервалов и частот и многие другие.

Счётчик— электронное устройство для подсчета количества импульсов, поступающих на его вход.

Единицей счёта является входной тактовый импульс (см. рис. 6.1), при этом срабатывание счетчика происходит в момент смены логического уровня импульса уровня. Счетчик может реагировать как на фронт – смену низкого уровня напряжения на высокий, так и на срез – смену с высокого уровня напряжения на низкий.

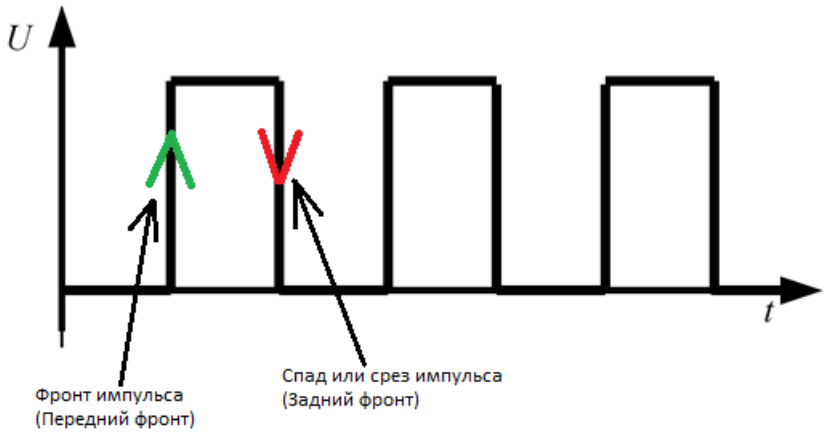


Рисунок 6.1 – Импульсы тактового генератора

Счетчики можно разделить по следующим признакам:

1. Разрядность – количество выходов счетчика, определяет максимальное количество состояний выходов:

$$X = 2^n, \quad (2.1)$$

где X – число состояний выходов, n – разрядность счётчика.

2. Система исчисления – двоичные и двоично-десятичные. В первом случае в виде двоичного кода представляется любое число, во втором своим двоичным кодом представлена каждая цифра десятичного числа.

3. Направление счета – суммирующие, вычитающие, двунаправленные

4. Возможность предустановки – счетчики с предустановкой начинают счет с ненулевого начального значения, следовательно, максимальное количество состояний выходов определяется формулой:

$$X = 2^n - D, \quad (2.2)$$

где D – начальное предустановленное значение.

5. Формат выхода – двоичный, когда код на выходе изменяется на единицу по приходу каждого импульса, или кольцевой, когда среди выходов только один имеет активный логический уровень, позиция которого меняется на одну по приходу очередного импульса. Кольцевой счетчик фактически совмещает в себе функции двоичного счетчика с дешифратором.

Кроме того, нередко счетчики имеют вход разрешения работы, при отсутствии активного логического уровня на этом входе выходы счетчика находятся в высокоимпедансном состоянии и счетчик выключен, а также вход сброса в начальное состояние.

6.2 Ход работы

Программирование ПЛИС можно осуществлять схематично (с помощью блок-схемы) или программно (при помощи языков описания аппаратуры). Для данной лабораторной не используется физический тактовый генератор, вместо него на вход тактирования счетчика будут подаваться одиночные импульсы от устройства ввода, чтобы была возможность наглядно продемонстрировать работу счетчика. Приняты следующие обозначения входных и выходных сигналов:

- CLK – тактирующий сигнал;
- RESET – сигнал сброса;
- D[0..3] – двоичный код предустановки;
- Y[0..4] – двоичный код на выходе счетчика.

Варианты заданий приведены в таблице 6.1. Для всех вариантов обязательно наличие сигнала сброса. При суммирующем счете счетчик должен сбрасываться в предустановленное значение, при вычитающем – в максимально возможное.

Таблица 6.1 – варианты заданий для синтеза двоичного счетчика.

№	Разрядность	Предустановка	Направление счета	Тип входа CLK
1	5	SW[0..3]+0x00	Вверх	Пофронту
2	5	SW[0..3]+0x10	Вниз	Пофронту
3	5	SW[0..3]+0x20	Вверх	Посрезу
4	5	SW[0..3]+0x30	Вниз	Посрезу
5	6	SW[0..3]+0x40	Вверх	Пофронту
6	6	SW[0..3]+0x50	Вниз	Пофронту
7	6	SW[0..3]+0x60	Вверх	Посрезу
8	6	SW[0..3]+0x70	Вниз	Посрезу
9	7	SW[0..3]+0x80	Вверх	Пофронту
10	7	SW[0..3]+0x90	Вниз	Пофронту
11	7	SW[0..3]+0xA0	Вверх	Посрезу
12	7	SW[0..3]+0xB0	Вниз	Посрезу
13	8	SW[0..3]+0xC0	Вверх	Пофронту
14	8	SW[0..3]+0xD0	Вниз	Пофронту
15	8	SW[0..3]+0xE0	Вверх	Посрезу
16	8	SW[0..3]+0xF0	Вниз	Посрезу

Независимо от метода синтеза устройство сначала определяются входные и выходные средства для демонстрации работы счетчика:

- кнопка KEY[0], имитирующая импульс тактового генератора CLK;
- кнопка KEY[1], генерирующая сигнал RESET;
- переключатели SW[0..3], генерирующие двоичный код предустановки;

- светодиоды LED[0..7], отображающие двоичный код на выходе счетчика.

6.2.1 Необходимо синтезировать счетчик средствами графического редактора в соответствии с заданием, приведенным в таблице 6.1.

6.2.1.1 Первый способ синтеза счетчика графическим способом – при помощи так называемых IP-компонентов – универсальных модулей, настраиваемых с помощью специальной встроенной утилиты MegaWizard. Для их загрузки необходимо в главном меню выбрать раздел «Инструменты» («Tools») и в нём выбрать пункт «IPCatalog». Соответствующее окно со списком библиотек появится в правой части (рисунок 6.2). Необходимо выбрать компонент LPM_COUNTER и нажать появившуюся под списком кнопку «Add». После этого среда в специальном диалоговом окне предложит сохранить компонент в виде файла в одном из выбранных форматов – VHDL или Verilog. Имя файла необходимо дописать в конце предложенного адреса. После нажатия кнопки «ОК» появится отдельное окно для настройки компонента (рисунок 6.3), в котором можно выбрать все необходимые для счетчика функции.

После выполненных действий появится только файл, описывающий счетчик, который появится в списке библиотек элементов в разделе «Project» (см. рисунки 3.9 и 3.10), откуда компонент данного типа уже можно вставить в поле проекта. К компоненту необходимо подключить соответствующие входы и выходы.

6.2.1.2 Второй способ синтеза счетчика графическим способом – построение схемы на базе D-триггеров. Примеры таких схем можно найти в литературе [4, 5]. D-триггер в библиотеке элементов обозначается как «dff».

6.2.3 Необходимо синтезировать счетчик согласно зданию в таблице 6.1 при помощи языков описания аппаратуры, как VHDL, так и Verilog. В теле программы должны присутствовать следующие переменные:

- counter – переменная с текущим значением счетчика
- maximum – переменная, ограничивающая максимальное значение счетчика.

Для помощи в написании программы на рисунке 6.4 приведена блок-схема, описывающая поведение счетчика. Примеры написания счетчиков можно найти в литературе [1–3].

6.2.4 Вне зависимости от способа синтеза следует расставить порты в PinPlanner в соответствии с таблицей 6.2, скомпилировать проект и запрограммировать плату.

Таблица 6.2. Соответствие сигналов и физических выводов.

Имя сигнала	Элемент, связанный с сигналом на плате	Тип вывода	Имяфизическогопорта
CLK	KEY[0]	Вход	PIN_AH17
Reset	KEY[1]	Вход	PIN_AH16
d[0..3]	SW[0..3]	Вход	PIN_L10 PIN_L9 PIN_H6 PIN_H5
y[0..7]	LED [0..7]	Выход	PIN_W15 PIN_AA24 PIN_V16 PIN_V15 PIN_AF26 PIN_AE26 PIN_Y16 PIN_AA23

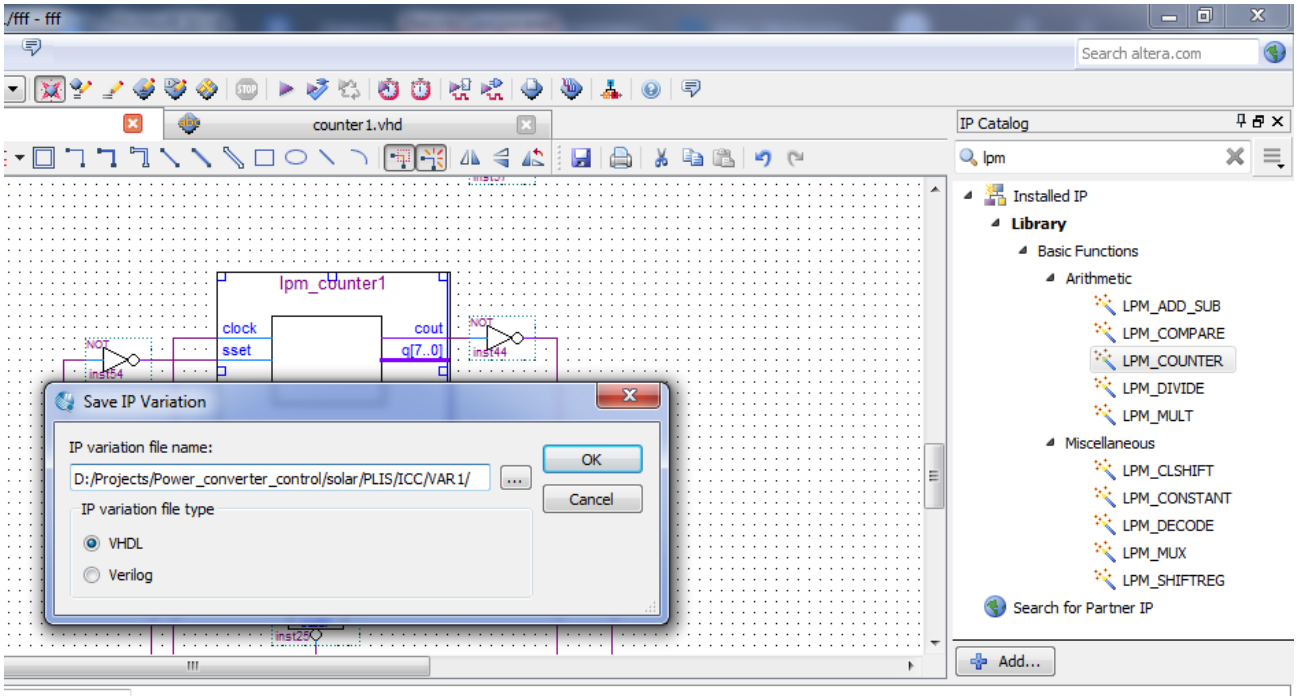


Рисунок 6.2 – Окна IP-компонентов.

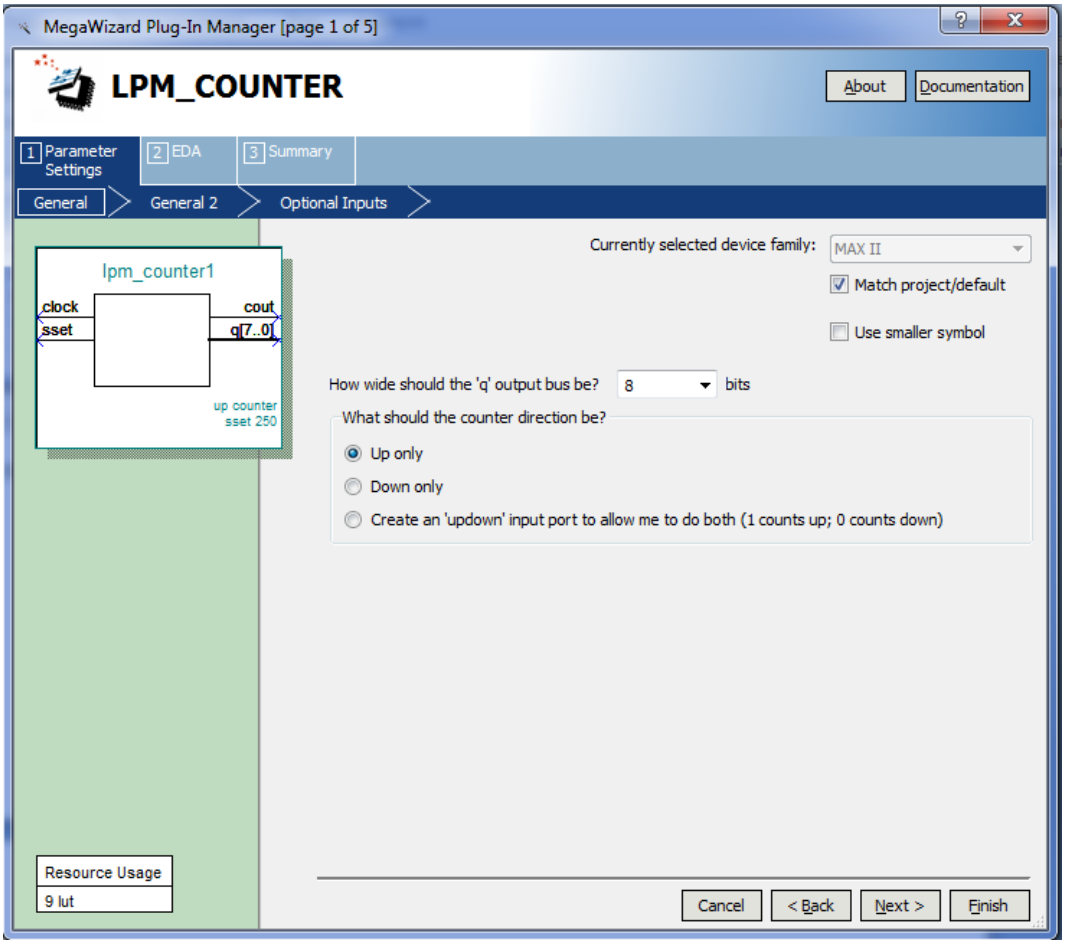


Рисунок 6.3– Окно настройки IP-компонента.

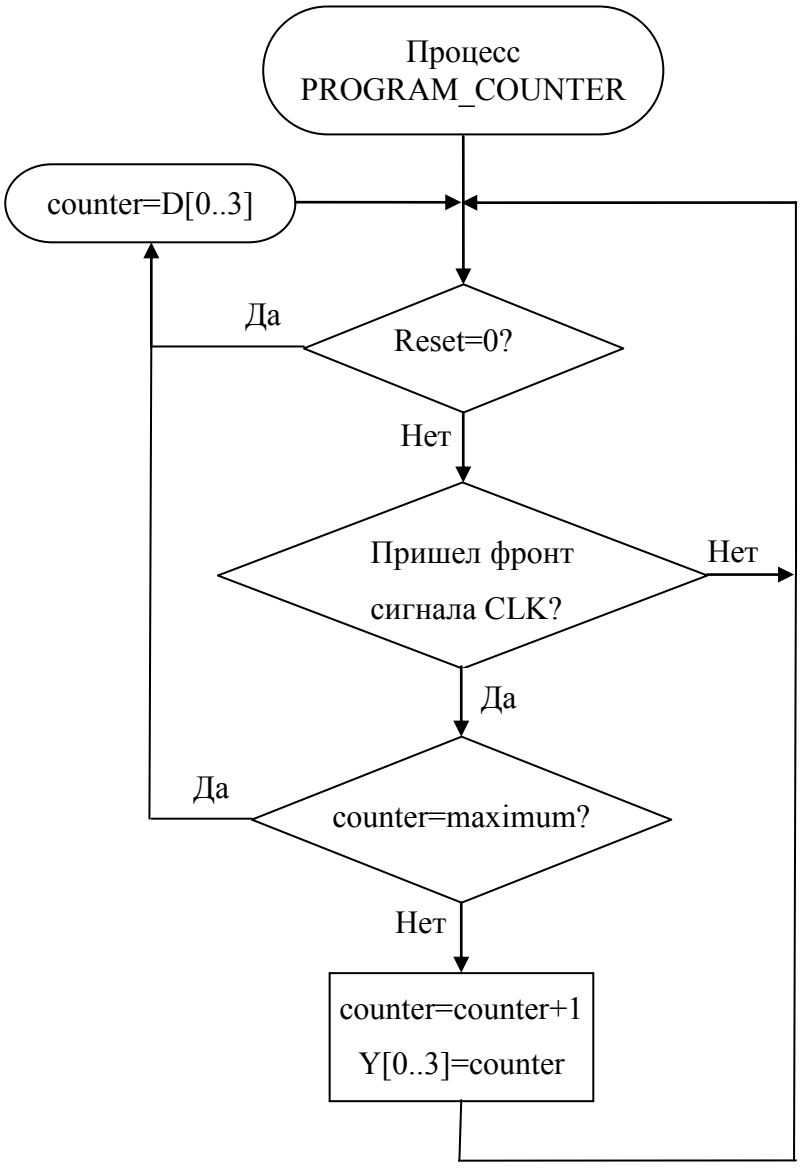


Рисунок 6.4 – Блок-схема работы счетчика

6.3 Контрольные вопросы

1. Что такое счётчик? Какие устройства можно реализовать на основе счётчика?
2. Что такое разрядность счётчика? В чём заключается смысл разрядности? Чем грозит выход за пределы разрядности?
3. Какие элементы управления применяются для счетчиков? Какие из них являются обязательными?
4. Описать алгоритм работы счётчика.

7 ЛАБОРАТОРНАЯ РАБОТА №4 «СОЗДАНИЕ ДЕЛИТЕЛЯ ЧАСТОТЫ МЕТОДАМИ ЯЗЫКОВ ОПИСАНИЯ АППАРАТУРЫ»

7.1 Введение

Цель работы: моделирование делителя частоты при помощи языков описания аппаратуры на ПЛИС с использованием средств цифровой индикации.

Делитель частоты является важным компонентом цифровых систем и применяется для формирования синхронизирующих и прочих сигналов для различных компонентов систем. Делители частоты также используются в синтезаторах частоты, радиочастотных блоках телекоммуникационных систем, устройствах синхронизации и генераторах развёрток телевизионных устройств. С точки зрения классификации цифровых устройств делители частоты, так же как и счетчики, относятся к триггерным автоматам, проектирование которых является задачей хоть и нетривиальной, но достаточно легко поддающейся формализации и, как следствие, автоматизации.

Фактически в качестве делителя частоты можно применять обычный счетчик с предустановкой, при этом сигнал с поделенной частотой берется с одного из двоичного кода на выходе счетчика. Такое подход оправдан в случае, когда требуется менять коэффициент деления в процессе эксплуатации устройства. В случае, если коэффициент деления является постоянным, использование счетчика является избыточным, и более выгодными являются более простые схемы на основе триггеров, в которых присутствует меньшее количество элементов и связей между ними. Как правило, при этом такая схема еще дополнительно выполняет функцию фазорасщепления, что является полезным именно для радиотехнических систем.

В любом случае при проектировании делителя частот разработчик сталкивается с задачей проектирования триггерных автоматов, которая является достаточно трудоемкой при использовании схемотехнических методов и значительно упрощается при использовании языков описания аппаратуры, в связи с чем последний способ преобладает при проектировании сложных устройств на основе ПЛИС.

7.2 Ход работы

Данная лабораторнаяработа выполняется на основе результатов, полученных в при синтезе счетчика методами языков описания аппаратуры, выполненного в лабораторной работе №3. При этом тактирующий сигнал должен подаваться от одного из генераторов, встроенных в отладочную плату. Кнопка сброса может быть использована по желанию для отладки устройства. Выходной сигнал подается на один из светодиодов. Выводы для кнопки сброса и светодиода можно взять из таблицы 6.1. Вывод для подачи тактового генератора берется из вариантов, представленных в таблице 7.1

Таблица 7.1. Соответствие сигналов тактовых генераторов и физических выводов.

Имя тактового сигнала	Частота, МГц	Имяфизическогопорта
FPGA_CLK1	50	PIN_V11
FPGA_CLK2	50	PIN_Y13
FPGA_CLK3	50	PIN_E11
HPS_CLK1	25	PIN_E20
HPS_CLK2	25	PIN_D20

Первоначально необходимо написать счётчик, с учётом того, что для процесса исходными являются сигналы от тактирующего генератора и кнопка сброса. В самом теле процесса необходимо создать вспомогательные переменные:

- counter – текущее значение счетчика
- divide_rate – коэффициент деления, фактически максимально значение счетчика.
- temp – временная переменная, меняющая свое значение после каждого достижения счетчиком максимального значения.

Блок-схема для помощи в организации кода описана на рисунке 7.1. Варианты заданий берутся в таблице 7.2. Делитель необходимо написать как на языке VHDL, так и на Verilog.

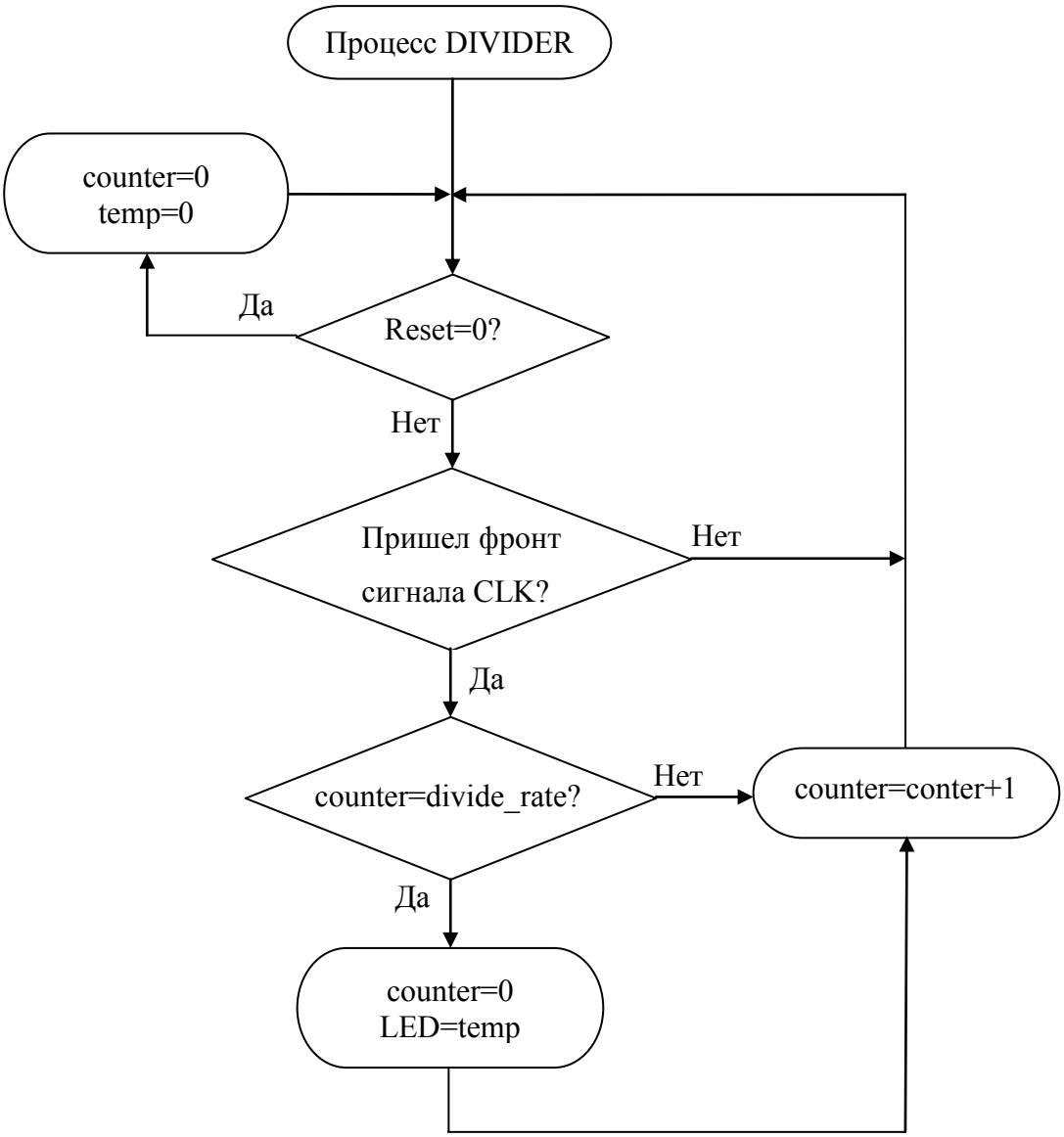


Рисунок 7.1 – Блок-схема делителя частоты

Таблица 7.2 Варианты заданий

Вариант	Коэффициентделения	Вариант	Коэффициентделения
1	2^{23}	12	2^{28}
2	2^{24}	13	2^{26}
3	2^{25}	14	2^{29}
4	2^{22}	15	2^{25}
5	2^{27}	16	2^{24}
6	2^{32}	17	2^{25}
7	2^{28}	18	2^{26}
8	2^{27}	19	2^{28}
9	2^{26}	20	2^{27}
10	2^{30}	21	2^{26}
11	2^{29}	22	2^{24}

7.3 Контрольные вопросы

1. Что такое делитель частоты? Где он используется?
2. На основе какого устройства реализуется делитель частоты?
3. Можно ли реализовать делитель частоты с дробным коэффициентом деления?
4. Какие основные средства управления присутствует в делителе частоты? Можно ли менять в процессе работы коэффициент деления?

СПИСОК ЛИТЕРАТУРЫ

1. Бибило П.Н. Основы языка VHDL – М.: СОЛОН-Р, 2002. – 224 с.

2. Поляков А.К. Языки VHDL и Verilog в проектировании цифровой аппаратуры – М. СОЛОН-Пресс, 2003. 320 с.

3. Сергиенко А.М. VHDL для проектирования вычислительных устройст – К ЧП «Корнейчук», ООО «ТИД ДС», 2003. – 208 с.

4. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx® с применением языка VHDL / И. Е. Тарасов. - М. : Горячая линия-Телеком, 2005. – 252 с.

5. Калабеков Б. А. Цифровые устройства и микропроцессорные системы : Учебник для средних специальных учебных заведений связи / Б. А. Калабеков. - 2-е изд., перераб. и доп. - М. : Горячая линия-Телеком, 2007. - 336 с.

6. Угрюмов, Евгений Павлович. Цифровая схемотехника : Учебное пособие для вузов / Е. П. Угрюмов. - 2-е изд., перераб. и доп. - СПб. : БХВ-Петербург, 2004. - 782 с.