Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Физический факультет
Кафедра физической электроники

РАБОТА ПРОВЕРЕНА
Рецензент,
д.ф.-м.н., профессор
_____/И.И. Клебанов
«____»_____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
д.т.н., профессор
_____/С.Ю. Гуревич
«____»_____2018 г.

РАЗРАБОТКА ДОВЕРЕННОГО СОПРОЦЕССОРА НА ОСНОВЕ ПЛИС
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 11.04.04.2018.244. ПЗ ВКР

Нормоконтролер,
к.т.н., доцент
_____Н.С. Колмакова
«____»_____2018 года

Руководитель проекта,
д.ф.-м.н,профессор
_____К.И. Костромитин
«____»_____2018 года

Автор проекта,
студент группы ЕТ-263
_____А.С. Подберезко
«____»_____2018 года

Челябинск 2018

АННОТАЦИЯ

Подберезко А.С. Разработка доверенного сопроцессора на основе ПЛИС. - Челябинск: ЮурГУ, ИЕТН,Ф; ЕТ-263, 2018 г., 43с., 23 ил., библиогр. список – 40 наим.

Объектом исследования данной выпускной квалификационной работы являются возможность построения доверенного сопроцессора на основе ПЛИС.

Цель выпускной квалификационной работы – разработка доверенного сопроцессора на основе ПЛИС.

В ходе работы был проведён анализ представленных на рынке семейств ПЛИС и выбор необходимой модели для поставленной задачи, была изучена среда проектирования Quartus II (версия 13.0) был изучен интерфейс передачи данных RS-232 была спроектирована конечная схема устройства, проект скомпиллирован и загружен в память ПЛИС, так же была подтверждена правильность функционирования системы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Тема исследования – построение доверенного сопроцессора на основе ПЛИС.

Актуальность данной темы связана с тем, развитие IT технологий, повсеместная компьютеризация, появление смартфонов и иных высокотехнологичных устройств привело к тому, что всё больше и больше людей стремятся защитить свои данные от злоумышленников, а лоббирование спецслужбами своих интересов заставляет производителей аппаратной части электронных устройств делать их управляемо уязвимыми. Такие производители как Intel, AMD устанавливают в свои CPU дополнительные микропроцессоры (подсистема Intel Management Engine, Platform Security Processor) которые в свою очередь позволяют хакерам и спецслужбам перехватывать данные пользователя на самом низком уровне. Чтобы этого избежать есть различные варианты защиты, такие как:

- полная локализация и отрезка от внешнего мира своего устройства;

- использование криптозащиты (ключи от которой хранятся в " безопасных" областях) и может быть взломана;

- использование сопроцессоров вычислений (даже если будут перехвачены исходные данные, доступа к функции производящей вычисления и конечному результату у злоумышленника не будет).

Целью работы является:

– Создание доверенного сопроцессора на основе ПЛИС.

Для достижения поставленной цели необходимо было решить следующие за-дачи:

– анализ представленных на рынке семейств ПЛИС и выбор необходимой модели для решения поставленной задачи

– изучение среды проектирования Quartus II (версия 13.0) изучение интерфейса передачи данных RS-232

– проектирование конечной схема устройства

– компиляция проекта и его загрузка в память ПЛИС

– подтверждение правильности функционирования системы

Для решения поставленных задач наиболее оптимальным является проектирование устройства на плате разработчика (RZ-EasyFPGA A2.2 «Рис. 2.4») по следующим причинам:

– наличие на плате микросхемы памяти;

– кварцевого генератора частоты;

– наличие пользовательских кнопок, переключателей;

– наличие светодиодов, 8-ми сегментного дисплея;

– разъём DB9 или последовательный COM порт (который будет задействован в обмене данными между ПК и ПЛИС по протоколу RS-232);

– прочие разъёмы которые могут потребоваться как в ходе отладки, так и для последующих исследований;

– наличие в комплекте поставки программатора.

Все вышеперечисленные достоинства позволяют приступить сразу непосредственно к проектированию, без необходимости изготовления различных кабелей, переходников, и преобразователей.

1 КРАТКИЙ ОБЗОР ПРОИЗВОДИТЕЛЕЙ ПЛИС

1.1 Международные производители ПЛИС

Множество компаний в мире занято производством цифровых устройств на основе ПЛИС и использованием их в своих системах. Краткое описание основных производителей современных вычислительных систем на основе ПЛИС и комплектующих к ним.

Xilinx, Altera, Lattice Semiconductor, Actel, Atmel, Nallatech, Mitrionics, Alpha Data, QuickLogic, Achronix Semiconductor, MathStar, Rapid Prototypes, National Instruments, Sun Microsystems, SGI, Cray, MNB Technologies, CPU Tech, Exegy, Celoxica, XtremeData, Plurality.

Основанная в 1984 году американская компания Xilinx является одним из лидеров в области производства ПЛИС-микросхем. На данный момент у этой компании существует несколько серий выпускаемой аппаратуры для разного рода вычислений:

– **Virtex.** Высокопроизводительные ПЛИС на основе FPGA, призванные заменить специализированные интегральные схемы при решениях различных ресурсоемких задач.

– **Spartan**. Более дешевые и менее производительные ПЛИС FPGA, разработанные для использования в устройствах, рассчитанных на большие тиражи и невысокую стоимость комплектующих.

– **CoolRunner** и **XC9500**. Серии ПЛИС типа CPLD, предназначенных для использования в различных портативных устройствах - мобильных телефонах, GPS-навигаторах, КПК и т.д. Для микросхем данного типа главными критериями является минимизация размеров и потребляемой мощности.

Микросхемы данных серий применяются довольно широко: последнее семейство Virtex-5 из серии Virtex используется, например, в суперкомпьютерах Cray XT5h и NEC SX-9. Также ПЛИС FPGA являются альтернативой процессоров цифровой обработки сигналов, для чего в каждом семействе присутствуют модели со встроенными блоками для этой обработки.

Компания Altera является основным конкурентом компании Xilinx, причем по всем основным направлениям. Главное из них - это производство ПЛИС как типа FPGA, так и типа CPLD.

– **Stratix** высокопроизводительных микросхем типа FPGA - Stratix IV, работают на 40-нм архитектуре.

–**Cyclone** серия ПЛИС FPGA для менее ресурсоемких задач, а - серию

– **Arria** в качестве компромисса между производительными Stratix и недорогими Cyclone.

– **Max** для мобильных устройств на основе ПЛИС типа CPLD.

– **HardCopy** серия ASIC микросхем - разработанных в качестве специализированных аналогов соответствующих FPGA Stratix. В 2008 году микросхема Stratix III была отмечена наградой DesignVisionAward в области "Полупроводники и интегральные схемы".

Начиная с серии Stratix III, в ПЛИС используется технология Programmable Power Technology, которая позволяет варьировать режим работы и, соответственно, потребляемую мощность логических ячеек в зависимости от необходимости быстрого выполнения поставленной задачи.

Микросхемы компании Altera активно применяются во многих областях, например, на рынке беспроводных и проводных коммуникаций, в военных технологиях, в области телевещания, а также в различных мобильных устройствах.

LatticeSemiconductor только в 2002 году начала производство FPGA-микросхем, и на этом рынке она занимает всего порядка нескольких процентов. Однако LatticeSemiconductor является одним из лидеров в области производства CPLD и SPLD (simple PLD - более простые по сравнению с CPLD программируемые устройства) микросхем. На этом рынке компания предоставляет целый спектр ПЛИС различной направленности:

– CPLD общего назначения;

– CPLD с низкой потребляемой мощностью;

– CPLD с гибридной архитектурой серии MachXO - обладает некоторыми свойствами FPGA, что позволяет большей гибкости при программировании;

– CPLD серии ispXPLD 5000V/B/C, которая состоит из блоков Multi-FunctionBlock (MFB), каждый из которых может быть запрограммирован отдельно:

– как вычислительный блок;

– как блок памяти RAM или CAM;

– как блок, реализующий буфер типа FIFO.

– SPLD-микросхемы, используемые для проведения простых операций или в качестве связующих логических схем на плате.

Хотя компания LatticeSemiconductor появилась на рынке ПЛИС FPGA относительно недавно, она выпускает довольно большой ассортимент микросхем данного типа. Среди них стоит выделить микросхемы **FieldProgrammableSystemChip (FPSC)** - первой серии ПЛИС со встроенными ASIC ядрами, которые предназначены для реализации стандартных IP ядер - шинного интерфейса, высокоскоростного интерфейса или высокоскоростного трансивера; а также серию **Lattice XP2** и предназначенный для его программирования прикладной пакет ispLEVER, которым в 2008 году была присуждена премия DesignVisionAward в области "Средства программирования микросхем типа ASIC, FPGA и PLD".

1.2 Российские производители ПЛИС

Представительство российских компаний в данной области пока освещено не слишком широко, русскоязычная информация достаточно скудна. Информация о российских организациях, работающих в области ПЛИС.

НИИ МВС, ФГУП "НИИ КВАНТ", Инлайн Груп, Эфо, ИТМИВТ, Высокотехнологичные системы, НПП "Цифровые решения", DeverSYS.

- **НИИ МВС ЮФУ** Многопроцессорные системы с программируемой архитектурой, построенные на основе принципа модульного наращивания.
- **ФГУП "НИИ КВАНТ"** Кластеры из реконфигурируемых вычислительных модулей на базе специализированных плат с программируемыми логическими интегральными схемами.

2 ПОДГОТОВКА К РАБОТЕ

2.1 Установка программного обеспечения и драйверов устройств.

Для работы с данной ПЛИС нам нужно скачать официальную среду разработки - Quartus II Web Edition, она бесплатная. Заходим на официальный сайт, на момент написания самая новая версия 18.0, но нам для проекта будет достаточно версии 13.0.Выбираем согласно «рис.2.1» версию, операционную систему, в нашем случае - Windows

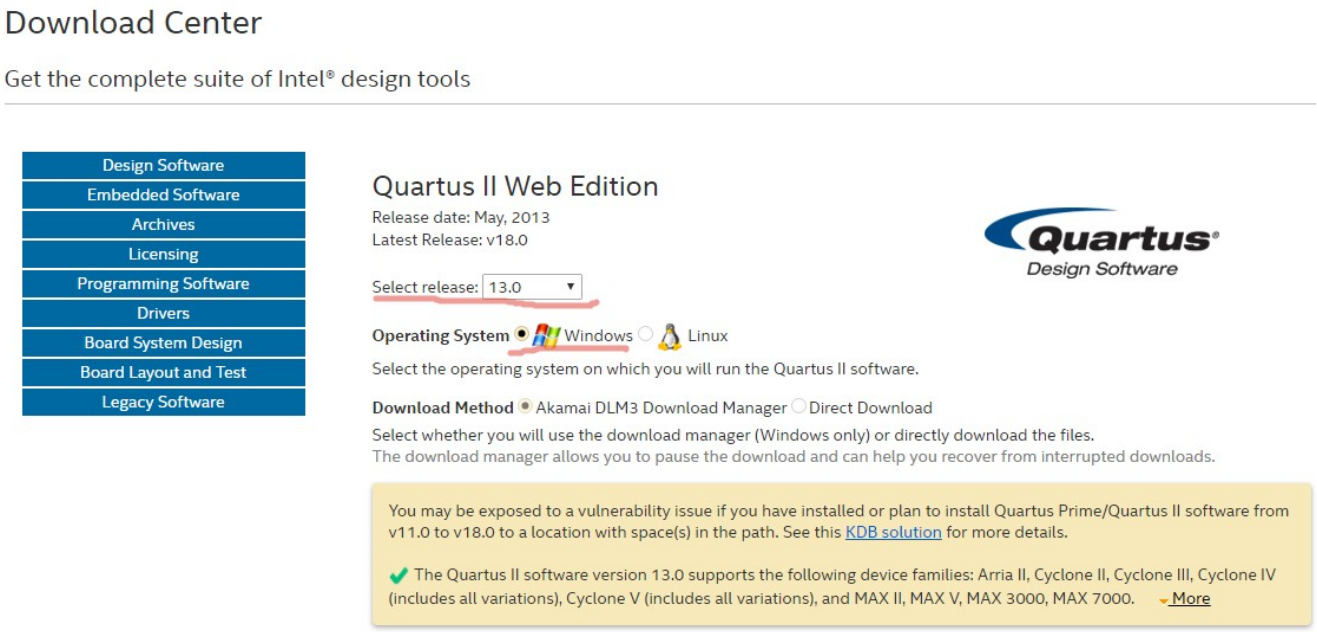


Рис. 2.1

Далее выбираем согласно «рис.1.2» необходимые файлы для скачивания:

– Quartus II Software (includes Nios II EDS);

– ModelSim-Altera Edition (includes Starter Edition);

– Cyclone II, Cyclone III, Cyclone IV device support (includes all variations);

– Quartus II Help.

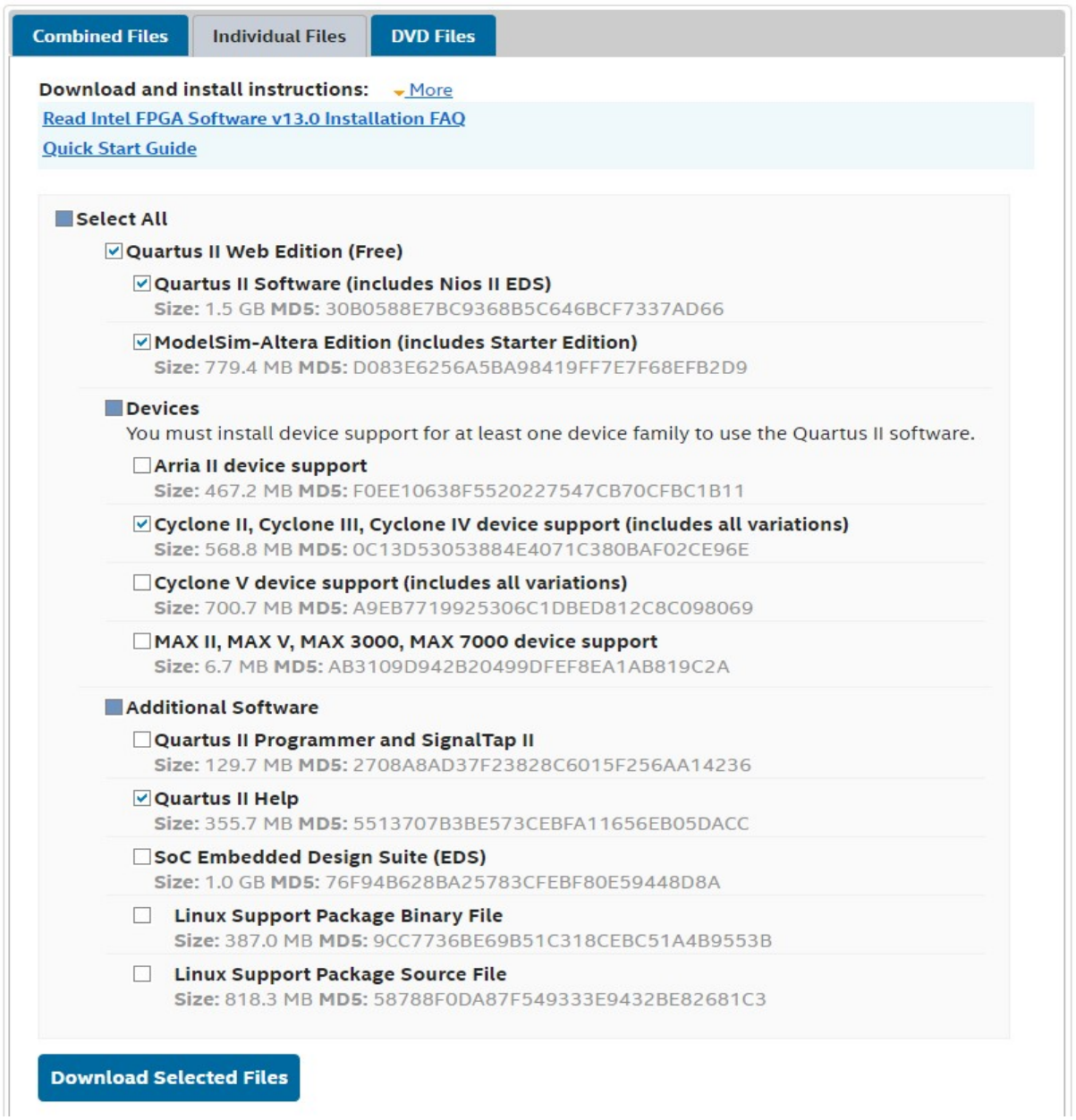


Рис. 2.2

Производим установку скачанных компонентов, рекомендуется папку Projects, в которой впоследствии будут храниться проекты создавать отдельно от папки программы.

После установки приступаем к установке драйверов устройств:

– устанавливаем драйвер на программатор Altera USB-Blaster

– Устанавливаем драйвер на адаптер usb 2.0 M to COM port «рис. 2.3»

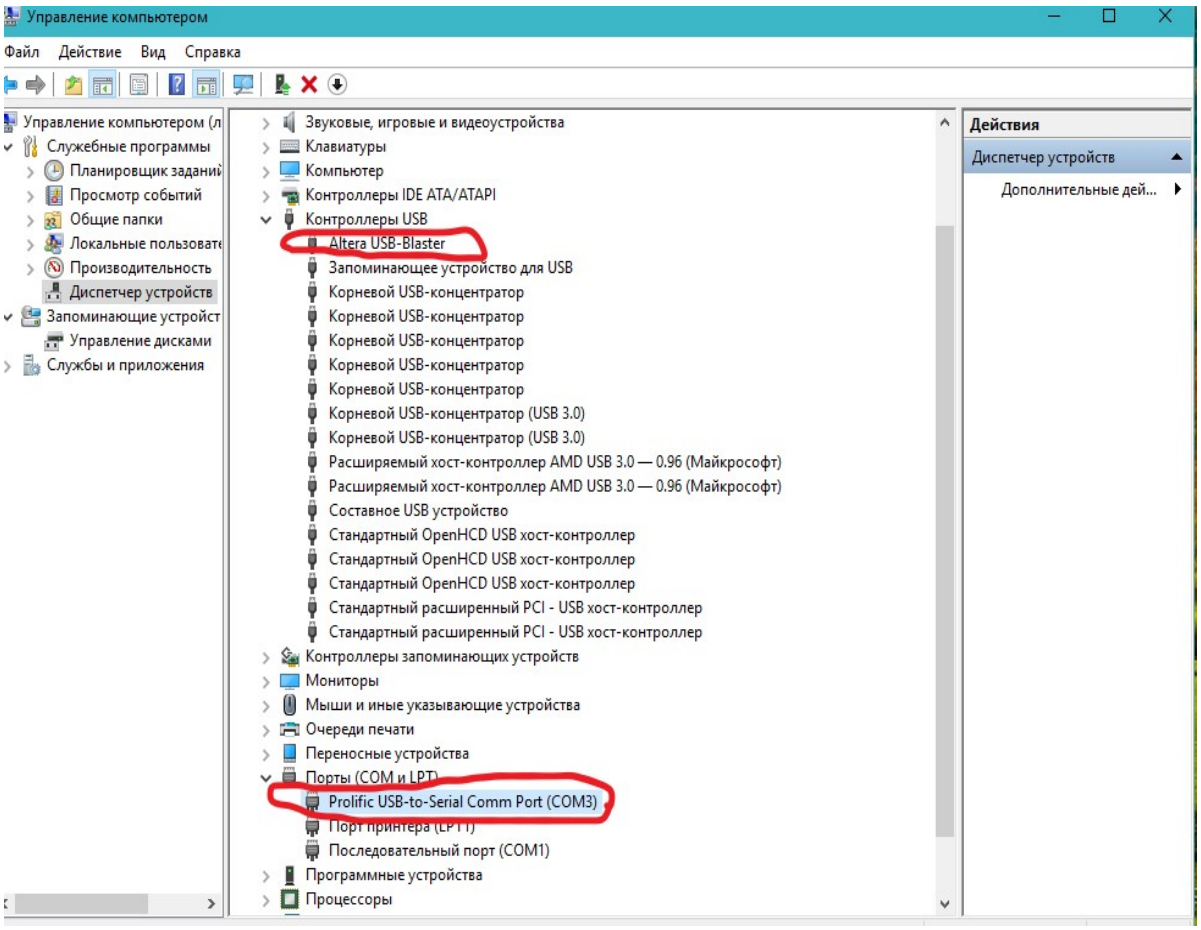


Рис. 2.3

2.2 Описание платы разработчика (RZ-EasyFPGA A2.2)

Далее подробно рассмотрим плату разработчика, комплект поставки изображён на « рис.2.4»

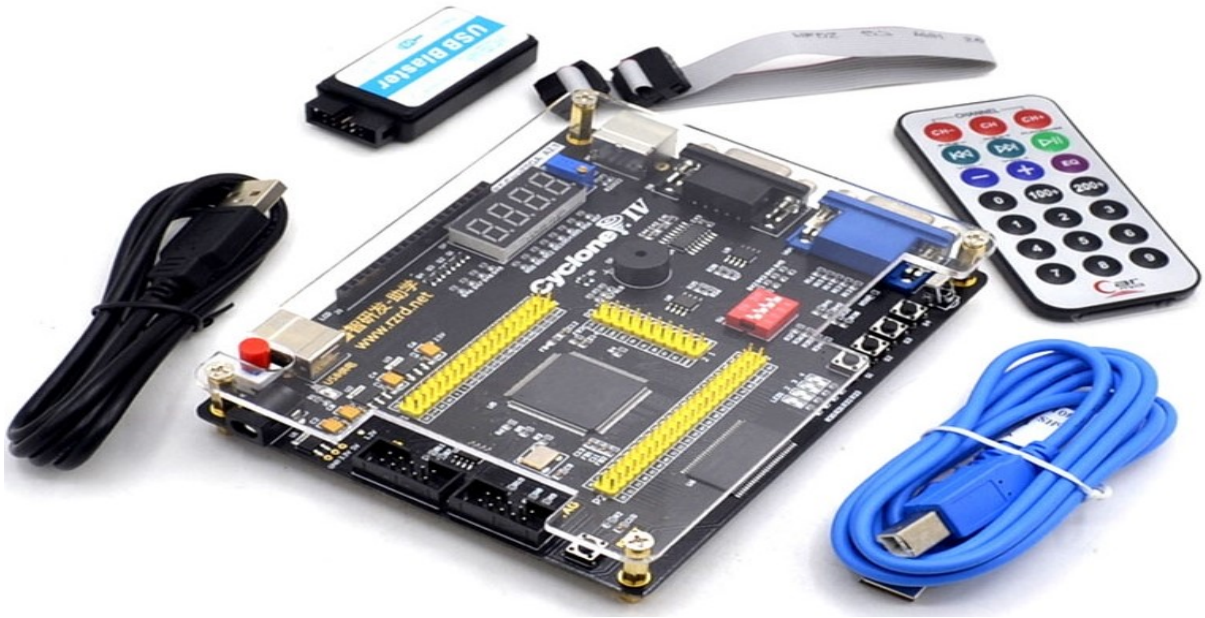


Рис. 2.4

На плате (RZ-EasyFPGA A2.2, « рис. 2.5») расположены (необходимые нам):

– микросхема памяти hynix SDRAM 64Mbit;

– кварцевый генератор частоты;

– пользовательские кнопки (4 шт.), переключатели(4 шт.);

– светодиоды красного цвета (4 шт.), 8-ми сегментный дисплей (4 разряда);

– разъём DB9 или последовательный COM порт (который будет задействован в обмене данными между ПК и ПЛИС по протоколу RS-232);

– прочие разъёмы, которые могут потребоваться как в ходе отладки, так и для последующих исследований;

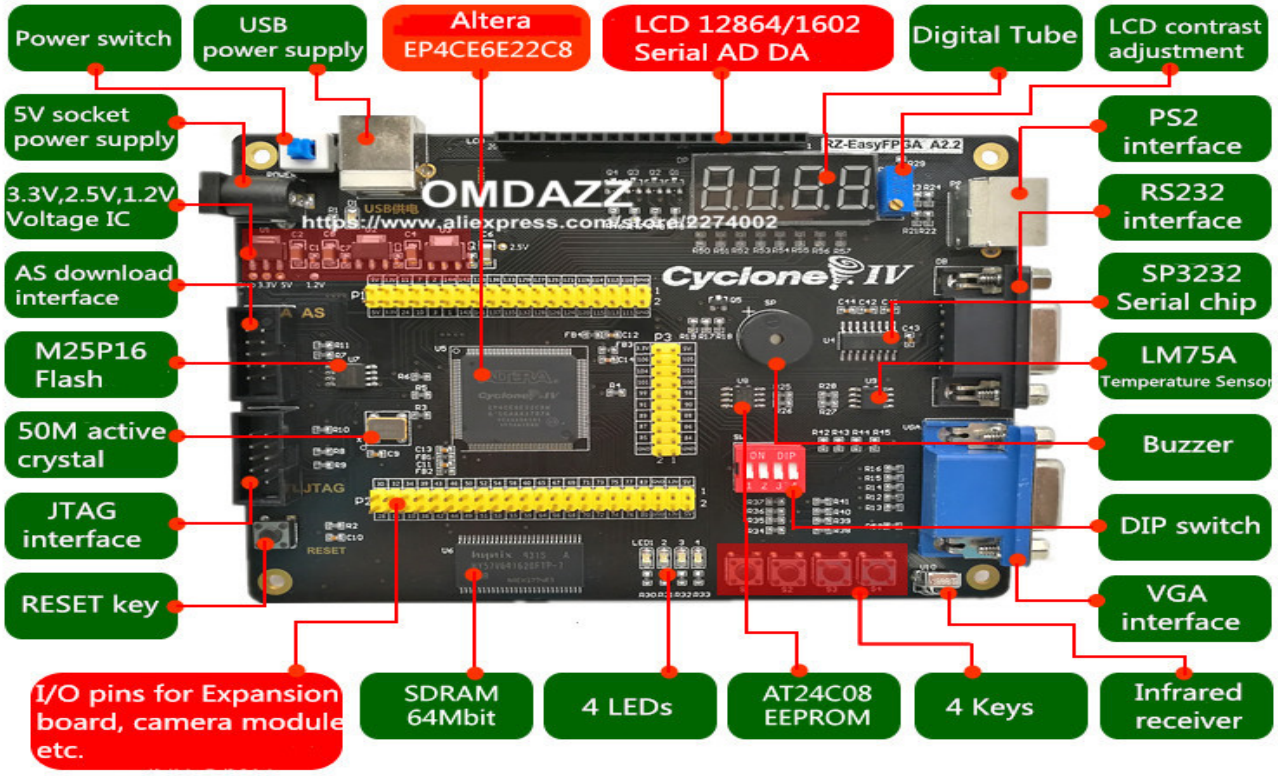

Power switch
USB power supply
Altera EP4CE6E22C8
LCD 12864/1602 Serial AD DA
Digital Tube
LCD contrast adjustment
5V socket power supply
3.3V,2.5V,1.2V Voltage IC
AS download interface
M25P16 Flash
50M active crystal
JTAG interface
RESET key
PS2 interface
RS232 interface
SP3232 Serial chip
LM75A Temperature Sensor
Buzzer
DIP switch
VGA interface
Infrared receiver
I/O pins for Expansion board, camera module etc.
SDRAM 64Mbit
4 LEDs
AT24C08 EEPROM
4 Keys
OMDAZZ
https://www.aliexpress.com/store/2274002
Cyclone IV

Рис. 2.5

3 VERILOG HDL СИНТАКСИС И КЛЮЧЕВЫЕ СЛОВА

3.1 Популярность языку Verilog придает простота синтаксиса, во многом совпадающего с языком программирования C, а также большие возможности при описании цифровых устройств и систем, как для синтеза, так и для моделирования, от уровня транзисторов до сложных иерархических структур. Язык Verilog предоставляет также возможности для своего расширения. Для этого служит механизм определения пользовательских примитивов UDP и язык программирования интерфейса PLI.

Типы данных:

Verilog содержит два базовых типа данных: wire и reg. Оба эти типа могут принимать 4 возможные значения при симуляции Verilog программы:

0

1

X — «неизвестное значение». Это значение используется только для симуляции, в реальной аппаратуре будет 0 или 1.

Z — «состояние высокого сопротивления», то есть отсутствие сигнала.

Тип wire используется для описания цепей, reg для регистров и переменных. Оба эти типа могут также быть использованы при описании многобитовых данных:

```
wire w1;
wire[31:0] bus; // 32-битовая шина
reg r1;
reg[7:0] bitvector; // 8-битовый регистр
```

Переменные типа *reg* имеют начальное значение 'X'. Цепи передают значения между регистрами. Если цепь не присоединена ни к какому регистру, она будет иметь значение 'Z'.

Verilog также содержит массивы, которые позволяют моделировать память:

```
    reg[31:0] memory[0:1023]; // 1024 слова памяти, каждое слово содер-
жит 32 бита.
```

Verilog содержит следующие типы данных:

- integer — то же самое, что «reg[31:0]»
- real
- time
- realtime

Verilog содержит два вида блоков, которые могут производить вычисления: «initial»-блок и «always»-блок.

«initial»-блок определяет, какие действия должны быть сделаны при старте программы. Этот блок не является синтезируемым и обычно используется для тестирования. Например:

```
module testbench;
  reg clock;
  reg[31:0] in1, in2;
  reg[63:0] out;

  // Тестируемый модуль
  multiplier mult(clock, in1, in2, out);

initial begin
  // Тестовые данные.
  in1 = 4;
  in2 = 20;

  // Подождать, пока результат будет готов.
  #10;

  // Вывести результат вычислений.
  $display("result=%d", out);

  $finish();
end
endmodule
```

Программа может содержать несколько «initial»-блоков, все они исполняются параллельно.

Операторы:

Тип	Символы	Выполняемая операция
Побитовые	~	Инверсия
	&	Побитовое AND
	|	Побитовое OR
	^	Побитовое XOR
	~^ или ^~	Побитовое XNOR (EQU)
Логические	!	NOT
	&&	AND
	||	OR
Редукция	&	Редуцированное AND
	~&	Редуцированное NAND
	|	Редуцированное OR
	~|	Редуцированное NOR
	^	Редуцированное XOR
	~^ или ^~	Редуцированное XNOR
Арифметические	+	Сложение
	-	Вычитание
	-	2’s complement
	*	Умножение
	/	Деление
	**	Экспонента (*Verilog-2001)
Отношение	>	Больше
	<	Меньше
	>=	Больше либо равно
	<=	Меньше либо равно
	==	Логическое равенство
	!=	Логическое неравно
	===	4-state логическое равенство
	!==	4-state логическое неравно
Сдвиг	>>	Логический сдвиг вправо
	<<	Логический сдвиг влево
	>>>	Арифметический сдвиг вправо (*Verilog-2001)
	<<<	Арифметический сдвиг влево (*Verilog-2001)
Сцепление	{ , }	Сцепление
Копирование	{n{m}}	Копирует m значение n раз
Условие	? :	Условие

4 СХЕМОТЕХНИЧЕСКИЙ РАЗДЕЛ

4.1 Структура кристалла cyclone IV EP4CE6E22C8N

Основные параметры СБИС Программируемой Логики (Cyclone IV) фирмы Altera: EP4CE6E22C8N.

При маркировке Altera применяет следующие обозначения EP4C_E_6_E_22_C_8_N «рис. 4.1»:

- EP4C – семейство Cyclone IV;
- E – улучшенная логика и память микросхемы;
- 6 – число логических элементов – 6272;
- E – тип корпуса EQFP (отличается от PQFP наличием заземляющего контакта на нижней поверхности микросхемы);
- 22 ширина микросхемы и число выводов нашей СБИС с каждой из 4 сторон всего – 144;
- C – рабочая температура микросхемы (T_J от 0° C до 85° C);
- 8 – класс скорости работы микросхемы;
- N– при изготовлении использовали безсвинцовые припои;
- линий ввода-вывода– 91;
- объем встроенной памяти – 270 кбит;
- число встроенных умножителей (разрядностью 18*18) – 15 шт.;
- число встроенных умножителей тактовых сигналов (PLL) – 2 шт.;
- объем файла конфигурации (без сжатия) – 3 Мбит.;
- кол-во блоков встроенного ОЗУ M9K (шт.) –30 ;
- кол-во глобальных цепей тактирования (шт.) – 10 ;
- поддерживаемые уровни напряжения ввода-вывода (В) 1,2; 1.5; 1,8; 2,5; 3,3;

Figure 1–3. Packaging Ordering Information for the Cyclone IV E Device

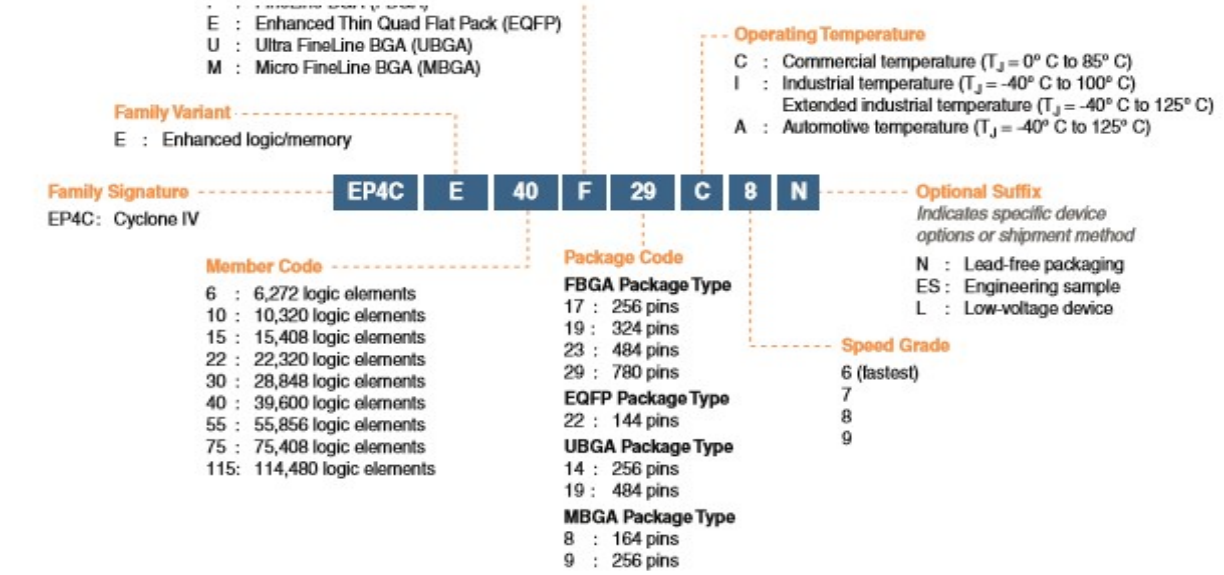


Рис. 4.1

Структура ячеек (Logic Elements and Logic Array Blocks) ПЛИС представлена на рисунке 4.2

Figure 2–4. Cyclone IV Device LAB Structure

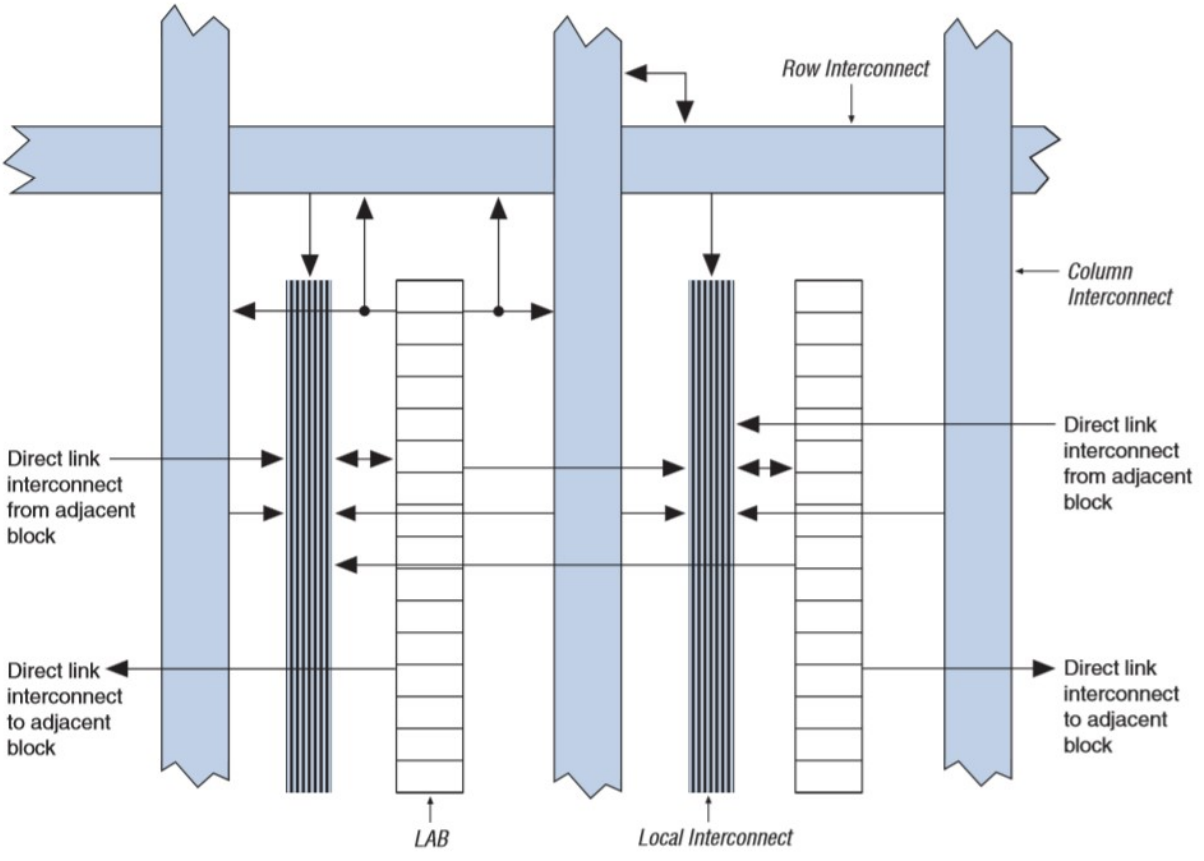


Рис. 4.2

4.2 Структурная схема сопроцессора представлена в виде нескольких масштабируемых звеньев. В простейшем случае мы имеем систему из:

- ПК (или иной источник входной информации);
- интерфейс передачи данных (rs-232);
- устройство сопроцессора логической частью которого является ПЛИС
- конечное устройство хранения (внешняя энергонезависимая память или другой ПК)

4.3 Создание проекта в среде проектирования Quartus II (13.0 версия)

При запуске программы мы видим приветственное окно «рис. 4.3»

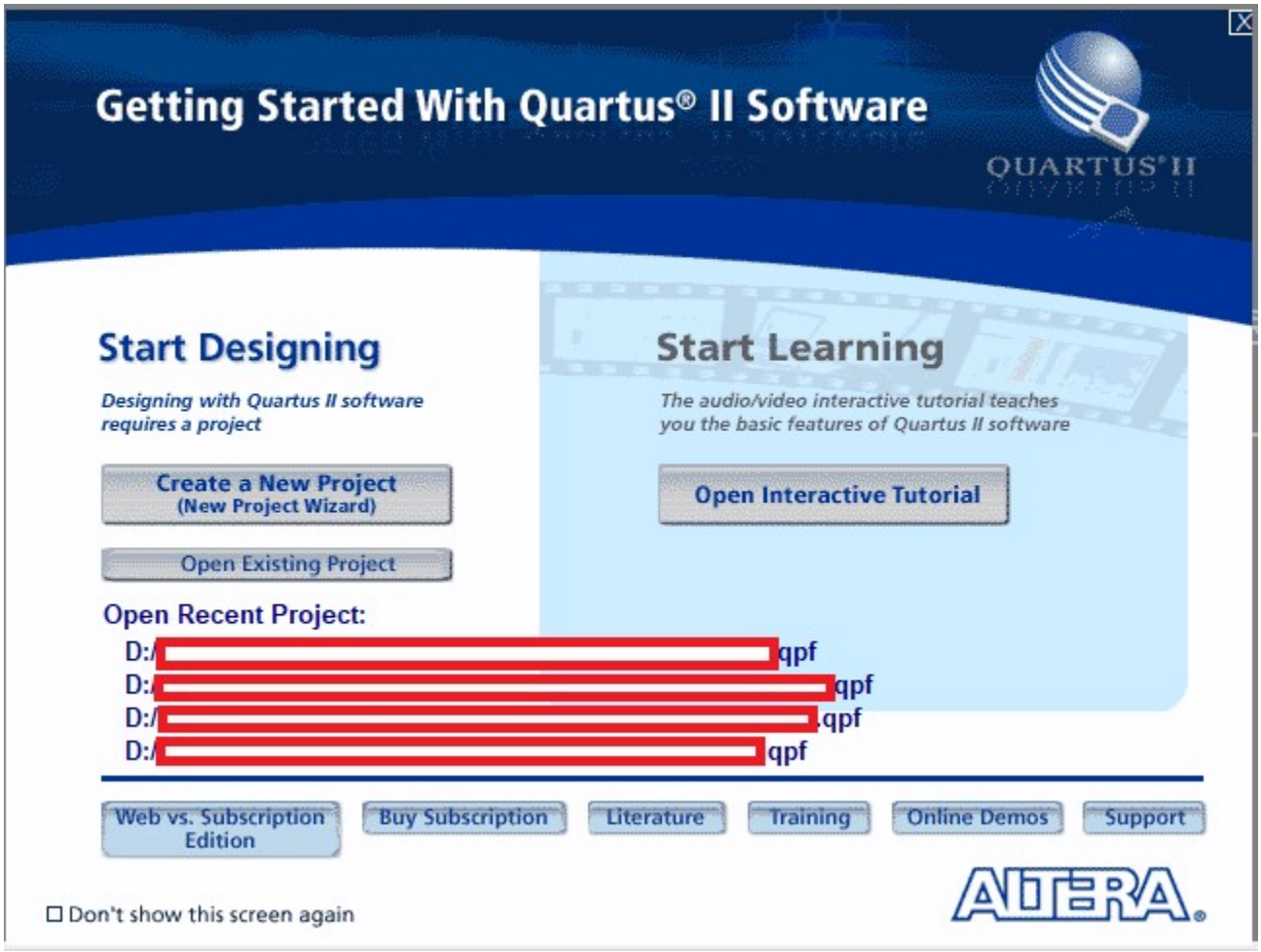


Рис. 4.3

в этом окне выбираем создание нового проекта или открытие существующего(Create New Project/Open Existing Project) . В первом случае у нас открыва-

ется мастер проектов «рис. 4.4» в первом окне которого говориться, что он поможет создать проект, в ходе создания которого вам предстоит :

- присвоить своему проекту название «рис. 4.5» и выбрать директорию сохранения;
- присвоить название главному файлу проекта;
- создать или добавить файлы проекта и библиотеки (при отсутствии готовых файлов это окно пропускаем клавишей «Next> »);
- выбрать семейство микросхемы ПЛИС «рис. 4.6» ;
- настроить среду моделирования»рис. 4.7».

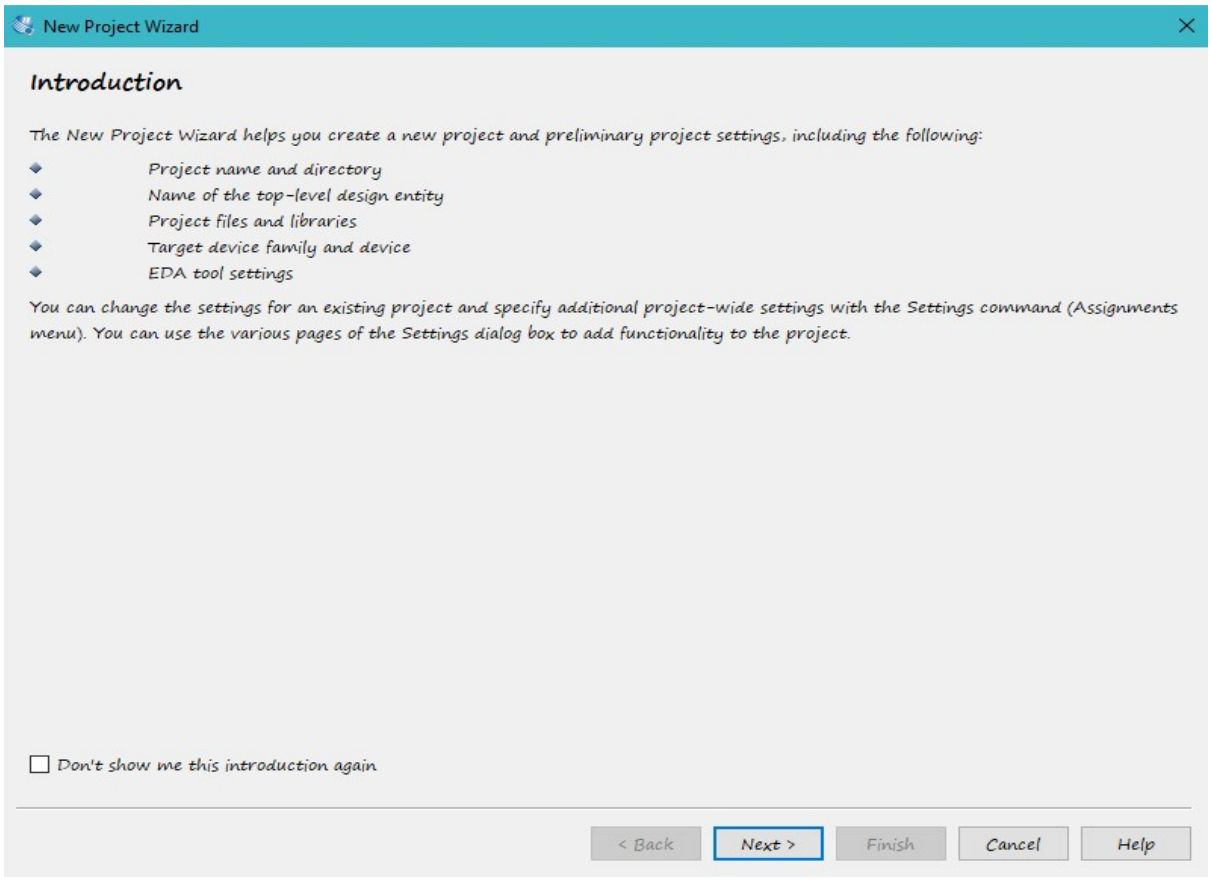


Рис. 4.4

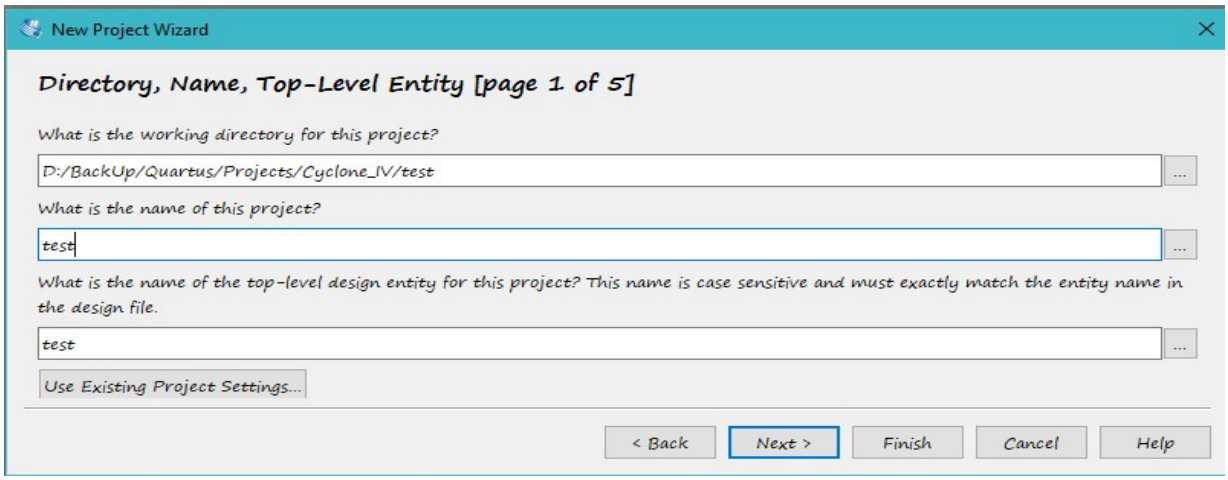

New Project Wizard
Directory, Name, Top-Level Entity [page 1 of 5]
What is the working directory for this project?
D:/BackUp/Quartus/Projects/Cyclone_IV/test
What is the name of this project?
test
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
test
Use Existing Project Settings...
< Back
Next >
Finish
Cancel
Help

Рис. 4.5

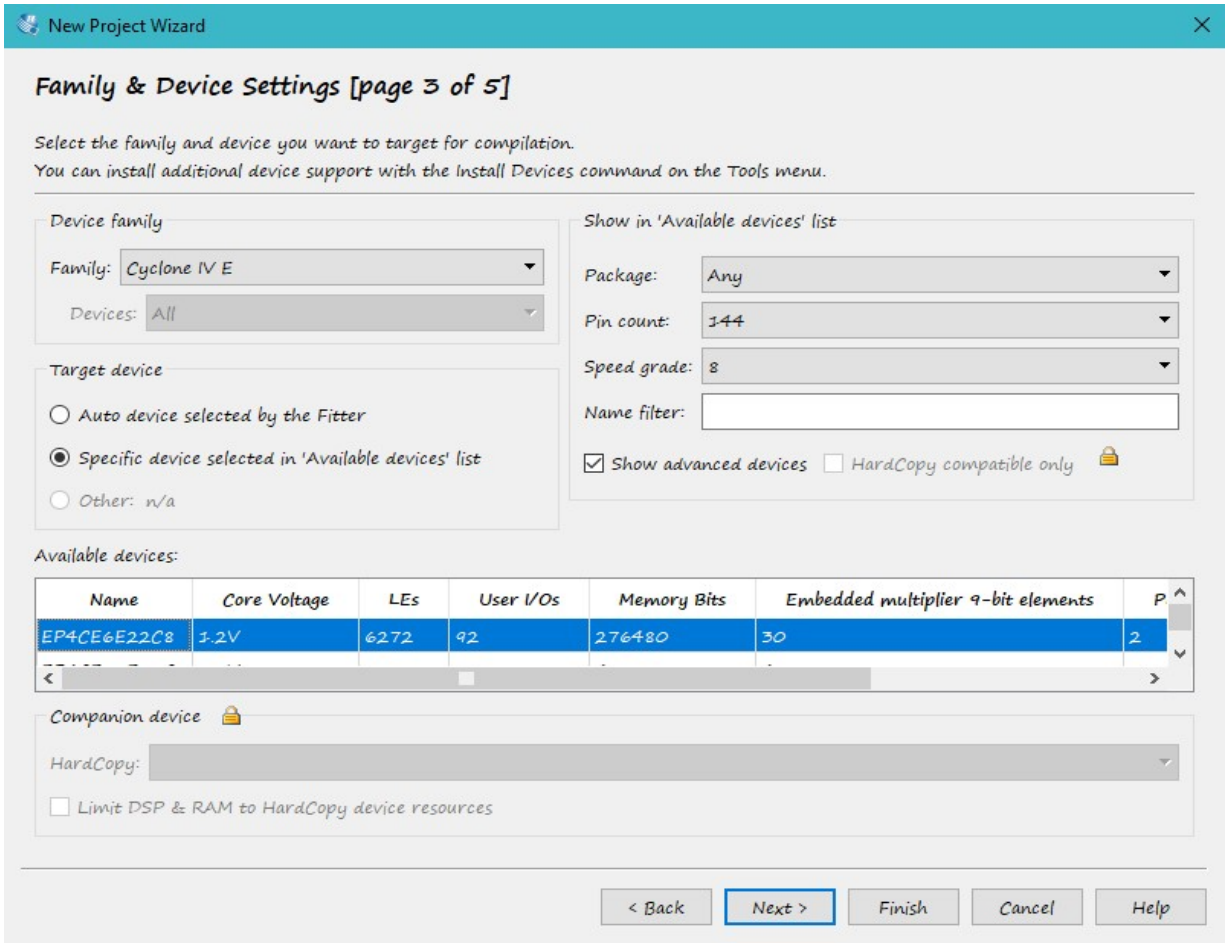

New Project Wizard
Family & Device Settings [page 3 of 5]
Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.
Device family
Family: Cyclone IV E
Devices: All
Target device
Auto device selected by the Fitter
Specific device selected in 'Available devices' list
Other: n/a
Show in 'Available devices' list
Package: Any
Pin count: 144
Speed grade: 8
Name filter:
Show advanced devices
HardCopy compatible only
Available devices:
Name
Core Voltage
LEs
User I/Os
Memory Bits
Embedded multiplier 9-bit elements
EP4CE6E22C8
1.2V
6272
92
276480
30
2
Companion device
HardCopy:
Limit DSP & RAM to HardCopy device resources
< Back
Next >
Finish
Cancel
Help

Рис. 4.6

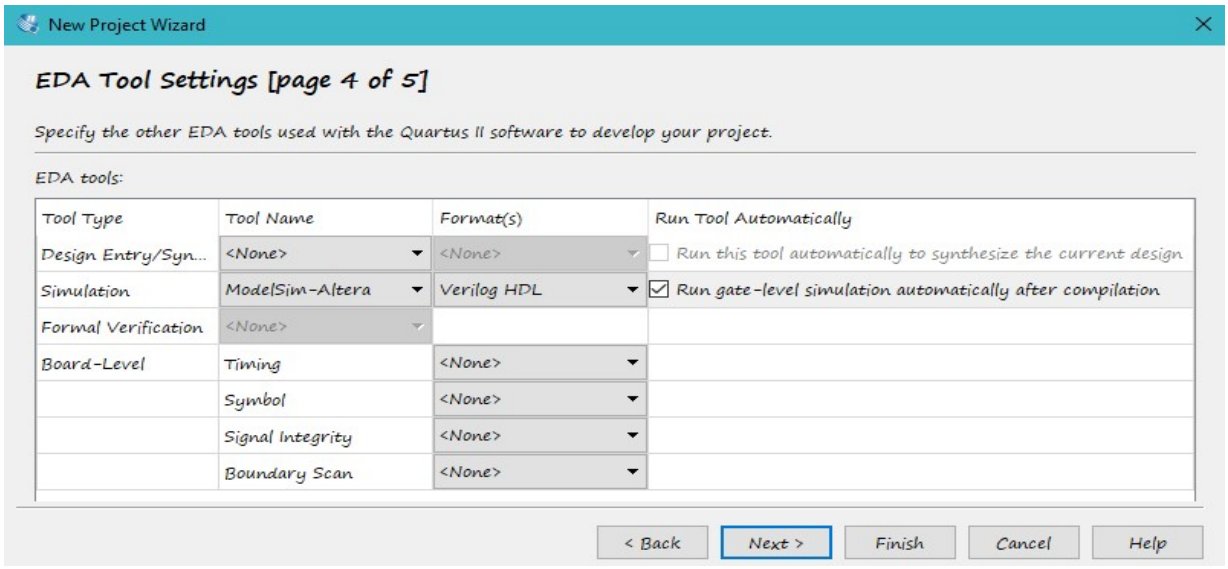

New Project Wizard
EDA Tool Settings [page 4 of 5]
Specify the other EDA tools used with the Quartus II software to develop your project.
EDA tools:
Tool Type
Tool Name
Format(s)
Run Tool Automatically
Design Entry/Syn...
<None>
<None>
Run this tool automatically to synthesize the current design
Simulation
ModelSim-Altera
Verilog HDL
Run gate-level simulation automatically after compilation
Formal Verification
<None>
Board-Level
Timing
<None>
Symbol
<None>
Signal Integrity
<None>
Boundary Scan
<None>
< Back
Next >
Finish
Cancel
Help

Рис. 4.7

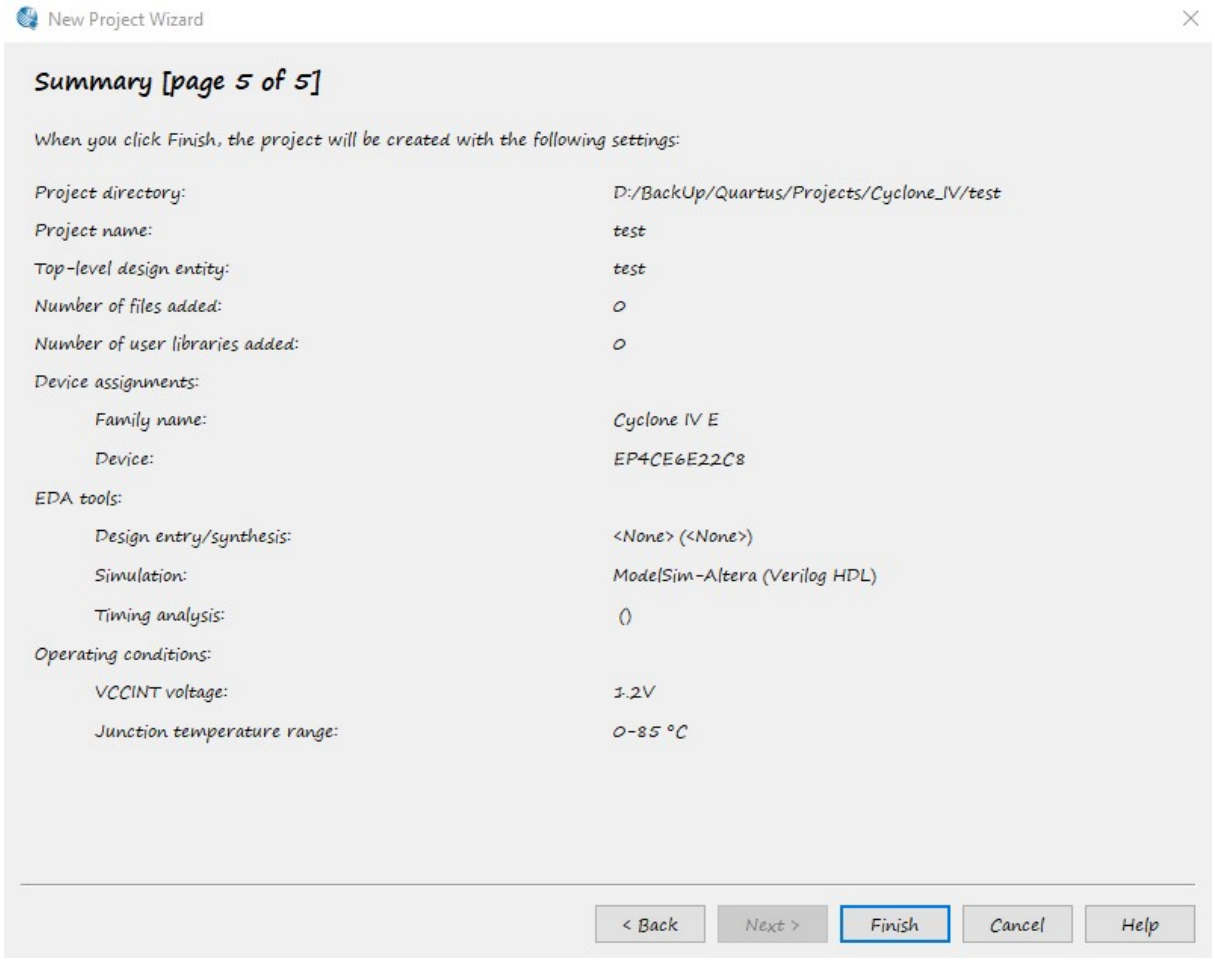

New Project Wizard
Summary [page 5 of 5]
When you click Finish, the project will be created with the following settings:
Project directory:
D:/BackUp/Quartus/Projects/Cyclone_IV/test
Project name:
test
Top-level design entity:
test
Number of files added:
0
Number of user libraries added:
0
Device assignments:
Family name:
Cyclone IV E
Device:
EP4CE6E22C8
EDA tools:
Design entry/synthesis:
<None> (<None>)
Simulation:
ModelSim-Altera (Verilog HDL)
Timing analysis:
()
Operating conditions:
VCCINT voltage:
1.2V
Junction temperature range:
0-85 °C
< Back
Next >
Finish
Cancel
Help

Рис. 4.8

В последнем окне проверяем правильно ли выбрана микросхема ПЛИС .

В левом окне переключаемся между файлами проекта, элементарными единицами проекта, компонентами.

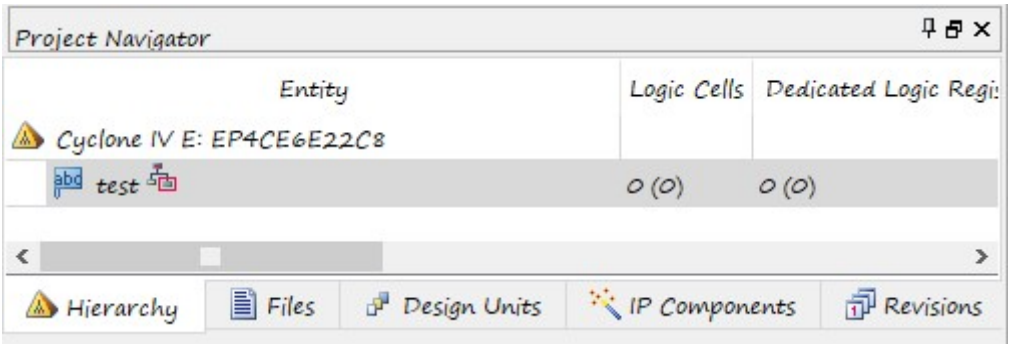
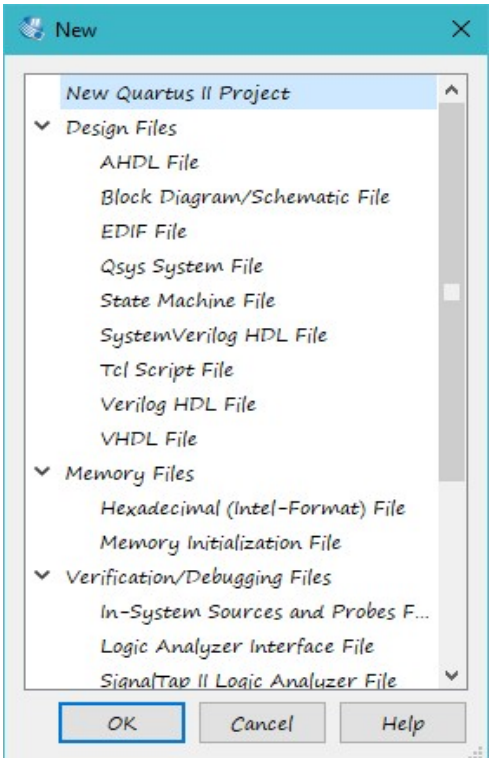


Рис. 4.9



Для создания нового файла в проекте нажимаем «File», выбираем в выпадающем списке пункт «New» или нажимаем сочетание клавиш «Ctrl+N».

Выбираем необходимое расширение(тип) файла (например Verilog HDL File).

Создаётся пустой файл с названием Verilog1.v в котором мы пишем проект, в ходе написания текст проекта автоматически подсвечивается как и в большинстве современных компиляторов.

Рис. 4.10

По завершению написания проекта нам необходимо произвести его компиляцию нажав на кнопку «Start Compilation» «рис. 4.11» или нажав сочетание клавиш «Ctrl+L».

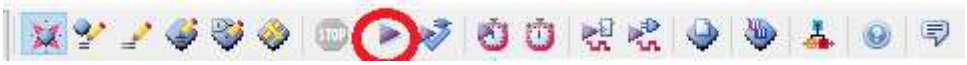


Рис. 4.11

В ходе которой Quartus II проверит проект на отсутствие ошибок и неразрешённых взаимосвязей, так же сообщит общее количество ошибок и предупреждений в проекте «рис. 4.12».

В появившемся окне «Flow Summary» «рис. 4.12» мы получаем краткий отчёт по использованным ресурсам ПЛИС

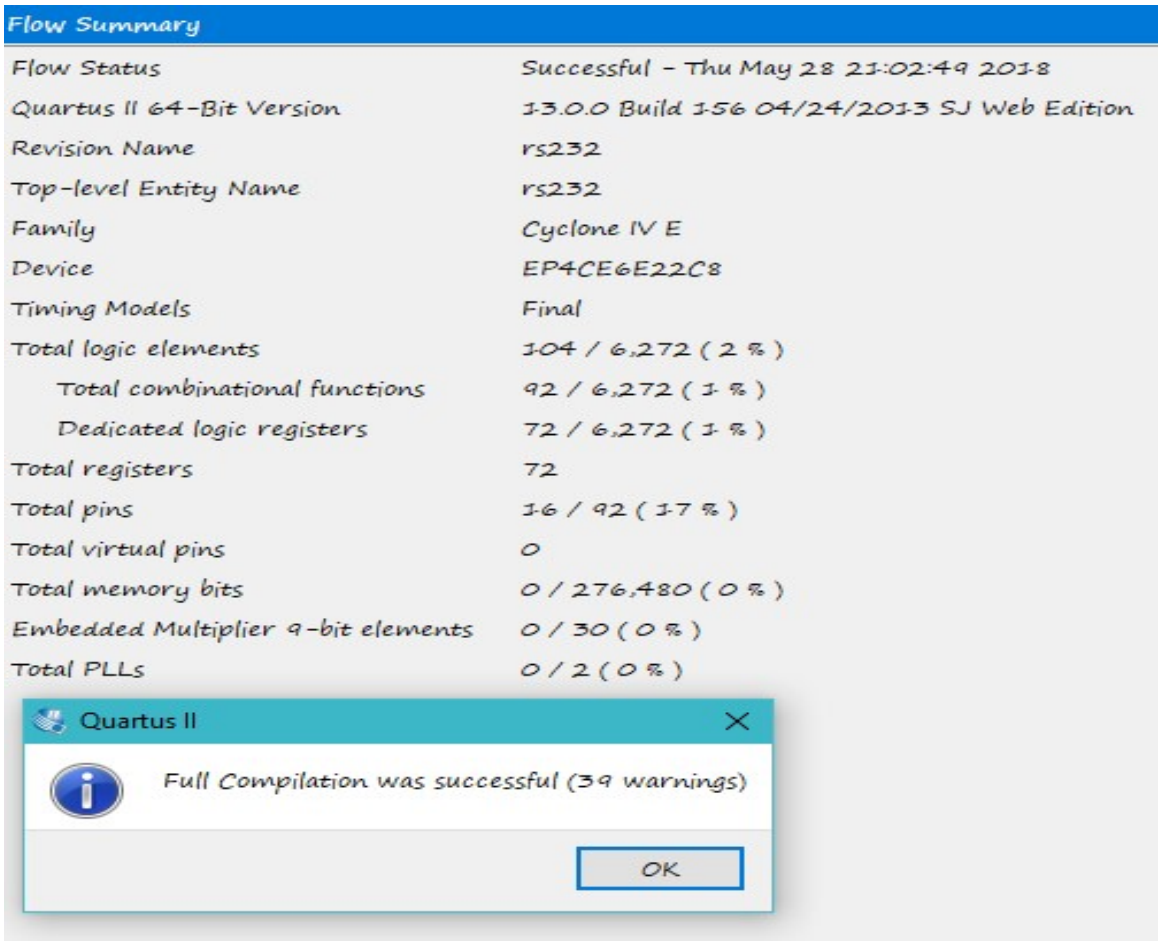


Рис. 4.12

4.4 Текст проекта приведён в таблицах 4.1-4.5

«Таблица 4.1»

```
1  ////////////////////////////////////////////////////////////////////
2  //|------------------------------------------------------------|//
3  //| Имя модуля: my_uart_top                                    |//
4  //| Имя файла: my_uart_top.qpf                                 |//
5  //| Функц. назначение: реализация rs-232 интерфейса в проекте|//
6  //| Автор : Подберезко А.С.                                    |//
7  //|------------------------------------------------------------|//
8  ////////////////////////////////////////////////////////////////////
9  module rs232(                          // начало головного модуля "rs232"
10                                clk,rst_n,      // объявление сигналов исполь-
   зуемых в модуле
11                                rs232_rx,rs232_tx      // --//--//--
12                                ,seg,n_dig
13                                );
14 //--Выходные порты-----------------------------------------------------------
15 output rs232_tx;                // RS232 исходящий сигнал передача данных, об-
   наруживается на pin_114
16 output [7:0] seg;
17 output [3:0] n_dig;
18 //--Входные порты------------------------------------------------------------
19 -
   input clk;                     //  50MHz входной тактовый сигнал кварцевого
20 генератора на плате, обнаруживается на pin_23
   input rs232_rx;                //  RS232 входящий сигнал приём данных, обна-
21 руживается на pin_115
22 input rst_n;                   //  сигнал сброса, обнаруживается на pin_25
23 //--Шины (цепи)внутри проекта-----------------------------------------------
   wire bps_start1,bps_start2;    // шина передачи данных о скорости на при-
24 ём/передачу
25 wire clk_bps1,clk_bps2;        // шина тактирования на приём/передачу
26 wire[7:0] rx_data;             // шина (8 бит)приёма данных
27 wire rx_int;           // шина инициализации разрешения/запрета приёмника
28 wire monitor;
29 //
30 wire [7:0] tx_data;
31 //
   speed_select   speed_rx(                    // вызов модуля "speed_select"
32 и работа с его сигналовами,
                  .clk(clk),                   // модуль устанавливает ско-
33 рость обмена данными
                  .rst_n(rst_n),               // в модуль speed select в in-
34 stance speed_rx в порт clk сигнала clk
                  .bps_start(bps_start1),      // использование сигналов для
35 модуля "my_uart_rx" - передача сигнала
```

```
                .clk_bps(clk_bps1)              // тактирования (clk) в порт
36  bps_start значения линии связи bps_start1
37              );
    my_uart_rx  my_uart_rx(                     // вызов модуля "my_uart_rx",
38  передача и получение от него сигналов
39              .clk(clk),
40              .rst_n(rst_n),
41              .rs232_rx(rs232_rx),
42              .rx_data(rx_data),
43              .rx_int(rx_int),
44              .clk_bps(clk_bps1),
45              .bps_start(bps_start1),
46              .monitor(monitor)
47              );
48  //
    speed_select  speed_tx(                     // вызов модуля "speed_select"
49  и получение от него сигналов
                .clk(clk),                      // дальнейшее использование
50  сигналов для модуля "my_uart_tx"
51              .rst_n(rst_n),
52              .bps_start(bps_start2),
53              .clk_bps(clk_bps2)
54              );
    my_uart_tx  my_uart_tx(                     // вызов модуля "my_uart_tx" и
55  получение от него сигналов
56              .tx_data(tx_data),
57              .clk(clk),
58              .rst_n(rst_n),
59              .rx_data(rx_data),
60              .rx_int(rx_int),
61              .rs232_tx(rs232_tx),
62              .clk_bps(clk_bps2),
63              .bps_start(bps_start2)
64              );
    digital_tube  digital_tube(                 // вызов модуля "speed_select"
65  и получение от него сигналов
                .seg(seg),                      // дальнейшее использование
66  сигналов для модуля "my_uart_tx"
67              .n_dig(n_dig),
68              .clk(clk),
69              .monitor(monitor)
70              );
    endmodule               // окончание головного модуля "rs232"
```

```
1  ////////////////////////////////////////////////////////////////////////
2  //|-------------------------------------------------------------------|//
3  //| начало модуля контроля сорости передачи данных    "speed_select"|//
4  //|-------------------------------------------------------------------|//
5  ////////////////////////////////////////////////////////////////////////
6  module speed_select(                              // начало модуля "speed_select"
7                                  clk,rst_n,      // объявление сигналов исполь-
   зуемых в модуле
8                                  bps_start,clk_bps
9                                  );
10 //--Входные порты----------------------------------------------------------
11 input clk;
12 input rst_n;
13 input bps_start;
14 //--Выходные порты----------------------------------------------------------
15 output clk_bps;// clk_bps
16 /*
17 parameter      bps9600         = 5207, // скорость 9600bps
18                                 bps19200        = 2603, // скорость 19200bps
19                                 bps38400        = 1301, // скорость 38400bps
20                                 bps57600        = 867,  // скорость 57600bps
21                                 bps115200       = 433;  // скорость 115200bps
22 parameter      bps9600_2       = 2603,
23                                 bps19200_2      = 1301,
24                                 bps38400_2      = 650,
25                                 bps57600_2      = 433,
26                                 bps115200_2 = 216;
27 */
28 `define         BPS_PARA                5207    //9600
29 `define         BPS_PARA_2              2603    //9600
30 reg[12:0] cnt;
31 reg clk_bps_r;
32 //---------------------------------------------------------------------------
33 reg[2:0] uart_ctrl;    // uart
34 //---------------------------------------------------------------------------
35 always @ (posedge clk or negedge rst_n)
36                         // always @(<sensitivity_list>) <statements>
37                         // <sensitivity_list> - это список всех входных
38                         // сигналов, к которым чувствителен блок. Это
39                         // список входных сигналов, изменение которых
40                         // влияет выходные сигналы этого блока.
41         if(!rst_n) cnt <= 13'd0;
42         else if((cnt == `BPS_PARA) || !bps_start) cnt <= 13'd0;
43                         // 13 разрядный десятичный счётчик обнуляем
44         else cnt <= cnt+1'b1;  // увеличаваем на 1 с каждым тактом
```

```
45  always @ (posedge clk or negedge rst_n)
46                          // Инициализация счётчика для скорости передачи?
47          if(!rst_n) clk_bps_r <= 1'b0;
48  //h&& bps_start
49          else if(cnt == `BPS_PARA_2 && bps_start) clk_bps_r <= 1'b1;
50                          // clk_bps_r
51          else clk_bps_r <= 1'b0;
52  assign clk_bps = clk_bps_r;
53  endmodule       // окончание модуля "speed_select"
```

«Таблица 4.3»

```
 1  ////////////////////////////////////////////
 2  //|--------------------------------------|//
 3  //| начало модуля приёмника   "my_uart_tx"|//
 4  //|--------------------------------------|//
 5  ////////////////////////////////////////////
 6  module my_uart_tx(                              // начало модуля "my_uart_rx"
    приёмника
 7                  clk,rst_n,      // объявление сигналов используемых в модуле
 8                  tx_data,
 9                  rx_data,rx_int,rs232_tx,
10                  clk_bps,bps_start
11                  );
12  //--Входные порты----------------------------------------------------------------
13  input clk;
14  input rst_n;
15  input clk_bps;          // clk_bps_r
16  input [7:0] rx_data;    // входящий 8-ми битный сигнал данных
17  input rx_int;           // входящий сигнал инициализации разрешения/запрета
    приёмника
18  //--Выходные порты---------------------------------------------------------------
19  output rs232_tx;
20  output bps_start;       // отправка сигнала о начале передачи данных[]
21  //--Внутренние переменные--------------------------------------------------------
22  reg rx_int0,rx_int1,rx_int2;   // объявление 3 регистров rx_int
23  wire neg_rx_int;               // объявление шины neg_rx_int
24  output [7:0] tx_data;
25  reg [7:0] tx_data;
26  //--постоянное присвоение при положительном фронте тактового сигнала и отри-
    цательном ----------------------------------------------------------------
27  //--фронте сигнала сброса--------------------------------------------------------
28  always @ (posedge clk or negedge rst_n) begin // начало блока присвоения
29          if(!rst_n) begin        // если отсутствует негативный фронт сигнала
    сброса, то:--------------------------------------------------------------------
30                  rx_int0 <= 1'b0; //инициализация 0(1-го) бита шины, присвое-
    ние в  0-ой бит 0
```

```
32                  rx_int1 <= 1'b0; //инициализация 1(2-го) бита шины, присвое-
    ние в  1-ой бит 0
33                  rx_int2 <= 1'b0; //инициализация 2(3-го) бита шины, присвое-
    ние во 2-ой бит 0
34          end
35  else begin                      // в противном случае
36                  rx_int0 <= rx_int;
37                  rx_int1 <= rx_int0;
38                  rx_int2 <= rx_int1;
39          end
40  end
41  assign neg_rx_int =  ~rx_int1 & rx_int2;
42  //-------------------------------------------------------------------------
43  reg bps_start_r;
44  reg tx_en;      //
45  reg[3:0] num;
46  //-------------------------------------------------------------------------
47  always @ (posedge clk or negedge rst_n) begin
48          if(!rst_n) begin
49                  bps_start_r <= 1'bz; // высокоимпедансное состояние (обрыв
    цепи)
50                  tx_en <= 1'b0;          // один двоичный разряд равен 0
51                  tx_data <= 8'd0;        //8-мь десятичных разрядов равны 0
52          end
53          else if(neg_rx_int) begin
54                  bps_start_r <= 1'b1;
55                  tx_data <= rx_data;
56                  tx_en <= 1'b1;
57          end
58          else if(num==4'd11) begin
59                  bps_start_r <= 1'b0;
60                  tx_en <= 1'b0;
61          end
62  end
63  assign bps_start = bps_start_r;
64  //-------------------------------------------------------------------------
65  reg rs232_tx_r;
66  always @ (posedge clk or negedge rst_n) begin
67          if(!rst_n) begin
68                  num <= 4'd0;
69                  rs232_tx_r <= 1'b1;
70          end
71  else if(tx_en) begin
72          if(clk_bps)     begin
73                  num <= num+1'b1;
74          case (num)
75                  4'd0: rs232_tx_r <= 1'b0;               // первый бит в переда-
```

```
    ваемом сигнале старт бит
76                  4'd1: rs232_tx_r <= tx_data[0];        // bit0 - первый значащий бит
77                  4'd2: rs232_tx_r <= tx_data[1];        // bit1
78                  4'd3: rs232_tx_r <= tx_data[2];        // bit2
79                  4'd4: rs232_tx_r <= tx_data[3];        // bit3
80                  4'd5: rs232_tx_r <= tx_data[4];        // bit4
81                  4'd6: rs232_tx_r <= tx_data[5];        // bit5
82                  4'd7: rs232_tx_r <= tx_data[6];        // bit6
83                  4'd8: rs232_tx_r <= tx_data[7];        // bit7 - последний значащий бит
84                  4'd9: rs232_tx_r <= 1'b1;              // последний бит в передаваемом сигнале бит окончания
85                  default: rs232_tx_r <= 1'b1;           // значение по умолчанию
86          endcase
87  end
88          else if(num==4'd11) num <= 4'd0;       //
89          end
90  end
91  assign rs232_tx = rs232_tx_r;
92  endmodule       // окончание модуля "my_uart_rx" передатчика
```

«Таблица 4.4»

```
1   ////////////////////////////////////////////
2   //|--------------------------------------|//
3   //| начало модуля передатчика "my_uart_rx"|//
4   //|--------------------------------------|//
5   ////////////////////////////////////////////
6   module my_uart_rx(      // начало модуля "my_uart_rx" передатчика
7           clk,rst_n,      // объявление сигналов используемых в модуле
8           rs232_rx,rx_data,rx_int,
9           clk_bps,bps_start,
10          monitor
11          );
12  //--Входные порты----------------------------------------------------------------
13  input clk;
14  input rst_n;
15  input rs232_rx; // RS232
16  input clk_bps;  // clk_bps
17  //--Выходные порты---------------------------------------------------------------
18  output bps_start;
19  output[7:0] rx_data;
20  output[7:0] monitor;
21  output rx_int;
22  //-------------------------------------------------------------------------------
```

```
23  reg rs232_rx0,rs232_rx1,rs232_rx2,rs232_rx3;
24  wire neg_rs232_rx;
25  always @ (posedge clk or negedge rst_n) begin
26          if(!rst_n) begin
27                  rs232_rx0 <= 1'b0;      // инициализируем двоичное значение
28                  rs232_rx1 <= 1'b0;      // на регистрах сдвига
29                  rs232_rx2 <= 1'b0;
30                  rs232_rx3 <= 1'b0;
31          end
32          else begin
33                  rs232_rx0 <= rs232_rx;
34                  rs232_rx1 <= rs232_rx0;
35                  rs232_rx2 <= rs232_rx1;
36                  rs232_rx3 <= rs232_rx2;
37          end
38  end
39          //<20ns-40ns
40          //40ns
41  assign neg_rs232_rx = rs232_rx3 & rs232_rx2 & ~rs232_rx1 & ~rs232_rx0;
42          //neg_rs232_rx
43  //----------------------------------------------------------------------------
44  reg bps_start_r;
45  reg[3:0] num;   //счётчик бит
46  reg rx_int;
47  always @ (posedge clk or negedge rst_n)
48          if(!rst_n) begin
49                  bps_start_r <= 1'bz;// высокоимпедансное состояние
    (обрыв цепи)
50                  rx_int <= 1'b0;
51          end
52          else if(neg_rs232_rx) begin     //rs232_rx
53                  bps_start_r <= 1'b1;
54                  rx_int <= 1'b1;
55          end
56          else if(num==4'd11) begin
57                  bps_start_r <= 1'b0;
58                  rx_int <= 1'b0;
59          end
60  assign bps_start = bps_start_r;
61  //----------------------------------------------------------------------------
62  reg[7:0] rx_data_r;             //сдвиговый регистр приёмника
63  //----------------------------------------------------------------------------
64  reg[7:0] rx_temp_data; //
65  always @ (posedge clk or negedge rst_n)
66          if(!rst_n) begin
67                  rx_temp_data <= 8'd0;
68                  num <= 4'd0;
```

```
69                  rx_data_r <= 8'd0;
70          end
71          else if(rx_int) begin
72                  if(clk_bps) begin                       //      начинаем считать
73                          num <= num+1'b1;// увеличиваем на 1 с каждым тактом
74                          case (num)              //описание числа
75                                  4'd1: rx_temp_data[0] <= rs232_rx;
76          //инициализация 1(2-го) бита шины, присвоение в  0-ой бит 0
77                                  4'd2: rx_temp_data[1] <= rs232_rx;     //1bit
78                                  4'd3: rx_temp_data[2] <= rs232_rx;     //2bit
79                                  4'd4: rx_temp_data[3] <= rs232_rx;     //3bit
80                                  4'd5: rx_temp_data[4] <= rs232_rx;     //4bit
81                                  4'd6: rx_temp_data[5] <= rs232_rx;     //5bit
82                                  4'd7: rx_temp_data[6] <= rs232_rx;     //6bit
83                                  4'd8: rx_temp_data[7] <= rs232_rx;     //7bit
84                                  default: ;
85                          endcase
86                  end
87                  else if(num == 4'd11) begin             //1+8+1(2)=11bit
88                          num <= 4'd0;                    //STOP
89                          rx_data_r <= rx_temp_data;      //rx_data
90                  end
91          end
92  assign rx_data = rx_data_r ;
93  assign monitor = rx_data ;
94  endmodule       // окончание модуля "my_uart_rx" передатчика
```

«Таблица 4.5»

```
1  ////////////////////////////////////////////////////////////////////
2  //|--------------------------------------------------------------|//
3  //| начало модуля отображения на цифровом табло "digital_tube"|//
4  //|--------------------------------------------------------------|//
5  ////////////////////////////////////////////////////////////////////
6  module digital_tube( monitor,
7          seg,n_dig,
8          clk
9          );
10 //=============================================================================
11 //                Входные/выходные сигналы
12 //=============================================================================
13 input clk;
14 input monitor;// входящий сигнал данных.
15 output reg [7:0] seg;  // подключение и хранение в регистре значения разряда
16 output reg [3:0] n_dig; // подключение и хранение в регистре номера разряда
17                        // совпадает с разрядностью приёмника.
18 reg [36:0] count;
19 reg [3:0] disp;
```

```
always @ (posedge clk )//Всегда при + фронте сигнала тактирования выполнять
следующее.
begin
 count = count + 1'b1;
 n_dig= 4'b0000;
end
always @ (count[24])
begin
disp = {count[29:26]};
end
always @ (monitor)
begin case (monitor)   // Загораются сегменты притянутые к земле (равные 0)
//=============================================================================
//                      Преобразование в 8-сегментный код
//=============================================================================
                4'h0 : seg = 8'hc0;    // "0" 1100_0000          _____
                4'h1 : seg = 8'hf9;    // "1" 1111_1001        /  A  \
                4'h2 : seg = 8'ha4;    // "2" 1010_0100       /\\_____//\
                4'h3 : seg = 8'hb0;    // "3" 1011_0000      |  |     |  |
                4'h4 : seg = 8'h99;    // "4" 1001_1001      |F |     | B|
                4'h5 : seg = 8'h92;    // "5" 1001_0010      |  |_____|  |
                4'h6 : seg = 8'h82;    // "6" 1000_0010       \//  G  \\/
                4'h7 : seg = 8'hf8;    // "7" 1111_1000       /\\_____//\
                4'h8 : seg = 8'h80;    // "8" 1000_0000      |  |     |  |
                4'h9 : seg = 8'h90;    // "9" 1001_0000      |E |     | C|
                4'ha : seg = 8'h88;    // "a" 1000_1000      |  |___ _|  |
                4'hb : seg = 8'h83;    // "b" 1000_0011       \//  D  \\/__
                4'hc : seg = 8'hc6;    // "c" 1100_0110         \_____// DP\
                4'hd : seg = 8'ha1;    // "d" 1010_0001               \__/
                4'he : seg = 8'h86;    // "e" 1000_0110
                4'hf : seg = 8'h8e;    // "f" 1000_1110
                                       //     .GFE_DCBA
endcase
end
endmodule
////////////////////////////////////////////////////////////////////////
//|-------------------------------------------------------------|//
//| окончание модуля отображения на цифровом табло "digital_tube"|//
//|-------------------------------------------------------------|//
////////////////////////////////////////////////////////////////////////
```

5 ПРОВЕРКА РАБОТОСПОСОБНОСТИ И ПРАВИЛЬНОСТИ РАБОТЫ СОПРОЦЕССОРА

5.1 Проверка работоспособности.

а) При первом включении платы разработчика (напряжение питания 5 В постоянного тока) мы должны увидеть загоревшийся красным светодиод D1, расположенный между разъёмами питания и кнопкой включения «рис 5.1» , а так же светодиоды LED1-LED4 расположенные в нижней части платы «см. рис. 2.5» должны начать загораться слева направо в режиме «бегущий огонь с накоплением»

Рис. 5.1

Это позволяет определить работоспособность после транспортирования таких элементов платы как:

- цепи питания;
- кварцевый генератор частоты;
- микросхема ПЛИС;
- микросхемы энергонезависимой памяти;
- пользовательские светодиоды;

При повторных включениях платы разработчика мы должны увидеть загоревшийся красным светодиод D1, расположенный между разъёмами питания и кнопкой включения «рис 5.1» , а так же (если мы в своих проектах не перезаписали свои данные в микросхему постоянной памяти) светодиоды LED1-LED4 расположенные в нижней части платы «см. рис. 2.4» аналогично первому включению.

б) Далее нам необходимо загрузить отлаженный и скомпилированный проект в микросхему ПЛИС для этого подключаем программатор ALTERA USB Blaster к разъёму «JTAG» «см. рис. 2.4, 2.5» . Делать это необходимо при отключенном питании платы, чтобы избежать повреждения микросхемы ПЛИС.

После подключения программатора к ПК и плате в ПО Quartus II нажимаем на кнопку «Programmer», «рис. 5.2»

Рис. 5.2

Далее в окне «Programmer» « рис. 5.3» нажимаем кнопку «Hardware Setup»

В окне «Hardware Setup» из выпадающего меню «Currently selected hardware» выбираем позицию USB-Blaster [USB-0] и закрываем окно.

В окне «Programmer» нажимаем кнопку «Add File» в открывшемся окне «Selest Programming File» в папке нашего проекта открываем папку «output_files» и выбираем файл прошивки.

В верхней рабочей области окна в строке с выбранным файлом отмечаем все галочки

В нижней рабочей области окна появляется пиктограмма чипа ALTERA «рис. 5.4» .

После всех перечисленных действий загорится кнопка «Start», нажимаем её и наблюдаем за прогрессом прошивки в окошке «Progress:» по завершению высветится надпись «100%» на зелёном фоне.

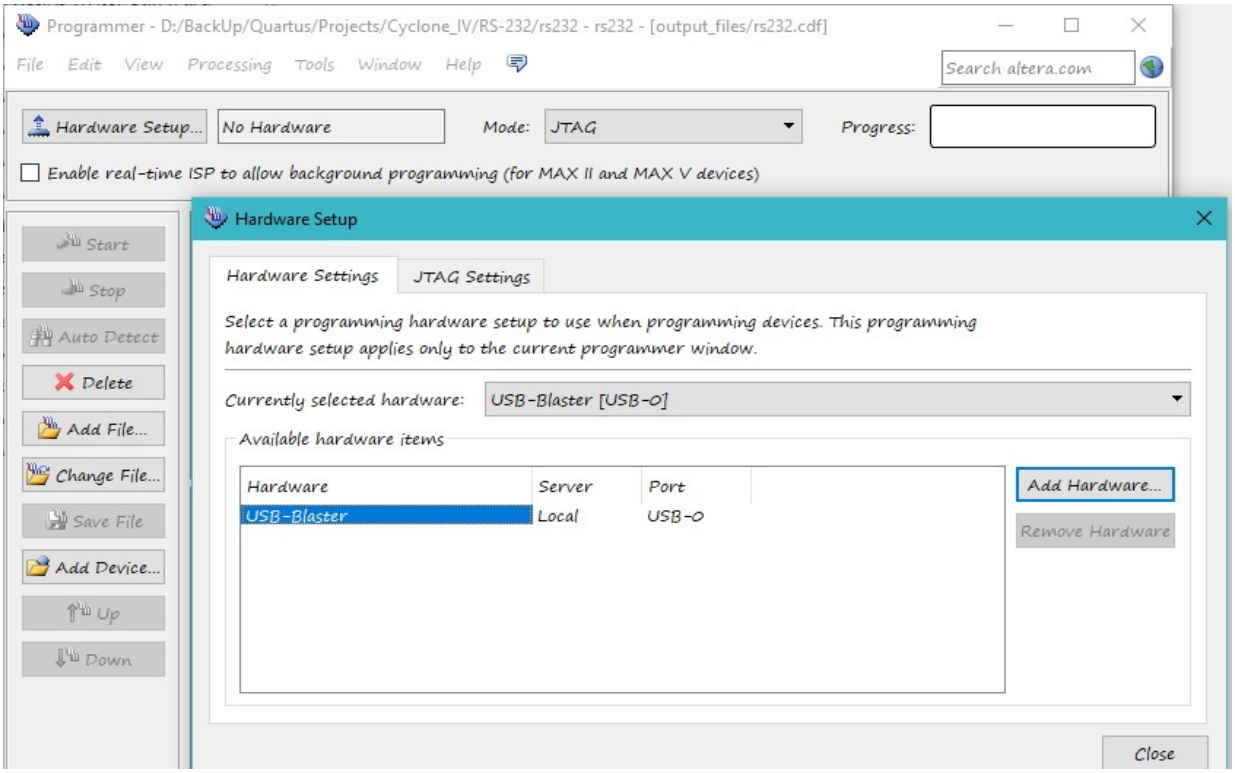
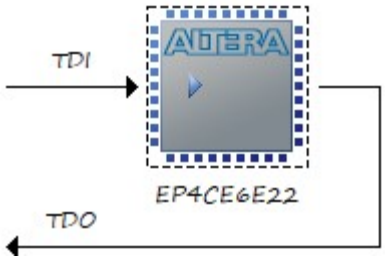


Рис. 5.3



Процесс прошивки успешно завершён.

Рис. 5.4

5.2 Проверка правильности выполнения написанного проекта.

Для проверки правильности выполнения проекта нам необходимо подключить плату к ПК с помощью адаптера usb 2.0 M to COM port или провода RS-232 – COM (при физическом наличии разъёма COM port на ПК) и запустить программу-терминал

Мы будем использовать PuTTY (/ˈpʌti/[2], Пати) — свободно распространяемый клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin. Также имеется возможность работы через последовательный порт.

Скачать программу можно с официального сайта, внешний вид программы представлен на «рис. 5.5»

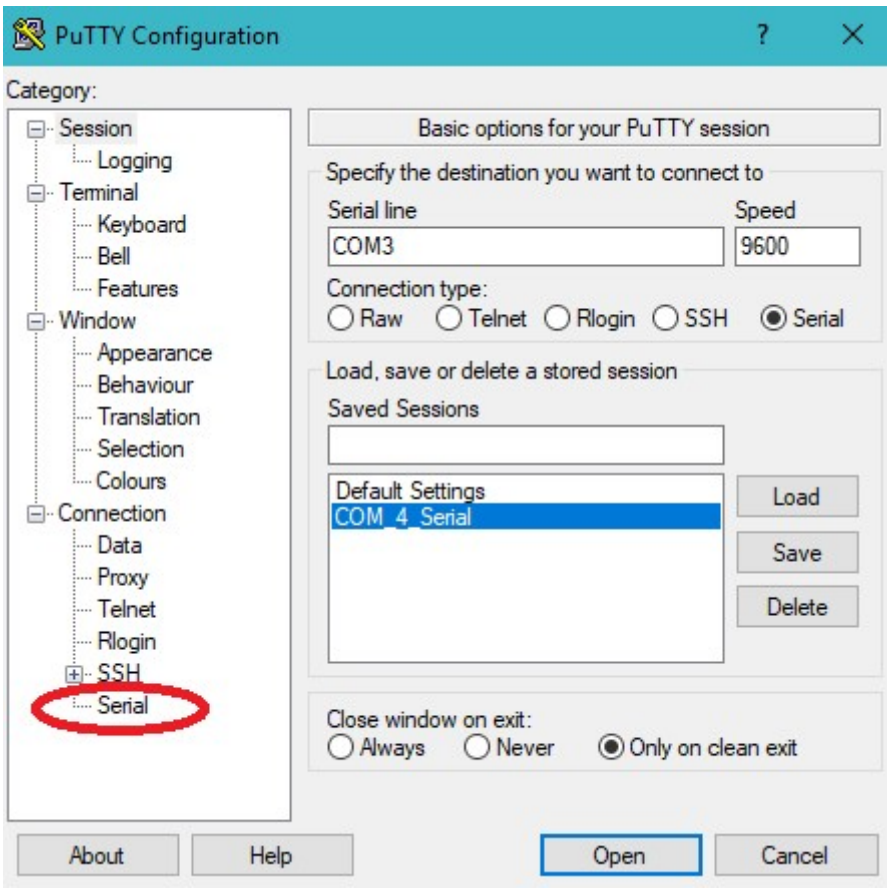


Рис. 5.5

Для проверки нам нужно выбрать режим «Serial», настройки режима находятся в нижней части древа (выделено красным)

Во вкладке «Serial» «рис. 5.6»изменяем номер порта (выделено красным) на тот, который определился в диспетчере устройств «см. рис. 2.3»

Далее нажимаем кнопку «Open» и у нас открывается консоль обмена данными между нашим ПЛИС и ПК «рис. 5.7»

В консоли мы вводим данные и отправляем их в ПЛИС, полученные данные автоматически отображаются на экране.

Критерием правильности написания проекта, его компиляции и выполнения сопроцессором является совпадение полученных данных с расчётными значениями полученными предварительно для каждой функции преобразования отдельно.

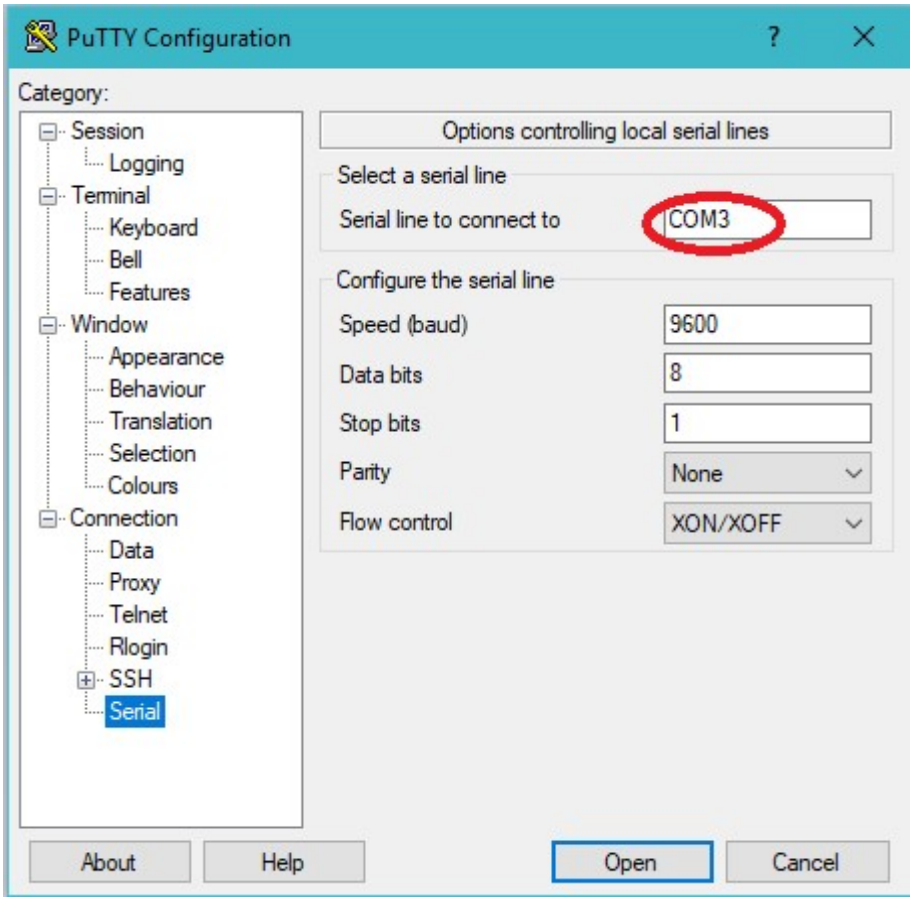


Рис. 5.6

Рис. 5.7

Функция задаётся в 93 строке «см. таблицу 4.3» присвоением вместо «`rx_data`» функции преобразования .

Для проверки правильности преобразования и обработки полученного сигнала функция преобразования заменяется на переприсваивание полученного значения отправляемому значению.

В таком случае мы на экране терминала видим те же числа, что и были введены для отправки на сопроцессор.

ЗАКЛЮЧЕНИЕ

В ходе работы был проведён анализ производителей ПЛИС на предмет выбора платы разработчика необходимой для выполнения проекта.

Был изучен в необходимом объёме язык проектирования цифровой аппаратуры Verilog HDL.

Была приобретена плата разработчика и необходимая периферия.

Была спроектирована структура ПЛИС позволяющая реализовать поставленные задачи.

Спроектированная структура была загружена в кристалл ПЛИС, была продемонстрирована работоспособность этой структуры и правильность работы.

БИБЛИОГРАФИЧЕСКИЙСПИСОК

Книги, журналы, печатные издания, ГОСТы

1. Стернхейм Э., Сингх Р., Триведи Я. Проектирование Цифровых Схем на Языке Описания Аппаратуры Verilog / с примечаниями Шевцова С. 1992) М.: 1992. – 45 с.

2. Bhasker J. Verilog HDL Synthesis. A Practical Primer, Allentown: Star Galaxy Publishing, 1998. - 218 p.

3. Brown S., Vranesic Z. Fundamentals of Digital Logic with Verilog Design// McGraw-Hill Science/Engineering/Math, 2013. — 864 p. — 3rd ed. — ISBN: 0073380547, 9780073380544/

4.Cavanagh J. Sequential Logic and Verilog HDL Fundamentals // CRC Press, 2016. — 858 p. — ISBN 9781498738224.

5. Cavanagh J. Verilog HDL Design Examples // Boca Raton: CRC Press, 2017. — 712 p.

6. Chu P.P. FPGA Prototyping By Verilog Examples// John Wiley & Sons, Inc., 2008. — 488 p.

7. Cilettii M.D. Advanced Digital Design With the Verilog HDL// New York: Pearson, 2011. - 986p.

8.Dolous. The Verilog Golden Reference Guide// Version 1.0, August 1996.- 151 p.

9. Hyde D.C. Computer Architecture Handbook on Verilog HDL// CSCI 320 Computer Architecture Handbook on Verilog HDL. By Dr. Daniel C. Hyde, Computer Science Department ,Bucknell University.1995, 32 p.

10.Lee J.M. A Practical Guide to Simulation and Synthesis in Verilog, Third Edition// Dordrecht: Kluwer Academic Publishers, 2002. - 356 p.

11.Lee M.J. Verilog Quickstart// A Practical Guide to Simulation and Synthesis in Verilog. 3rd edition. — Kluwer Academic Publishers, 2002. — 378 p. — eBook ISBN: 0-306-47680-0.

12.Lee W.F. Verilog Coding for Logic Synthesis// John Wiley & Sons, Inc., 2003. — 334 p.

13. Monk Simon. Programming FPGAs: Getting Started with Verilog// McGraw-Hill Education TAB, 2016. — 170 p. — ISBN 125964376X. — ISBN 978-1259643767.

14.Navabi Z. Verilog Digital System Design// Verilog digital system design / Zainalabedin Navabi. Includes bibliographical references and index. ISBN 0-07-047164-9. 1999 by The McGraw-Hill Companies, Inc. 477 p.

15.Nyasulu P., Knight J. Introduction to Verilog// Reference guide. - Carleton university, 32 p.

16. Palnitkar S. Verilog HDL: A Guide in Digital Design and Synthesis// Prentice Hall PTR, 2003. — 448 p.

17.Readler B.C. Verilog by Example: A Concise Introduction for FPGA Design// New York: Full Arc Press, 2011. — 114 p.

18. Sagdeo Vivek. The Complete Verilog Book // Boston/Dordrecht/London: Kluwer Academic Publishers, 1998. — 464 p.

19.Stine J.E. Digital Computer Arithmetic Datapath Design Using Verilog HDL// Springer US, 2004. — 181 p. — ISBN 978-1-4613-4725-5.

20. Sutherland S. Verilog HDL Quick Reference Guide// Based on the Verilog-2001 standard (IEEE Std 1364-2001). ISBN: 1-930368-03-8. Sutherland HDL, Inc., 2001, 56 p.

21.Tala D.K. Verilog Tutorial// 25-Oct-2003

22. Thomas D., Moorby P. The Verilog Hardware Description Language// 5th ed. — Springer, 2008. — 386 pages. — ISBN-10: 0387849300 ISBN-13: 978-0387849300

23.Williams John. Digital VLSI Design with Verilog // Springer, 2008. — 447 p. — e-ISBN 978-1-4020-8446-1.

24. Акчурин А.Д., Юсупов К.М. Программирование на языке// Verilog Учебное пособие. — Казань: Казанский федеральный университет, 2016. — 90 с.

25. В.В. Соловьёв Основы языка проектирования цифровой аппаратуры Verilog – М.: Горячая линия – Телеком, 2014. – 206 с.

2. Руководство пользователя GTKWave 3.3 Wave Analyzer User's Guide Updated December 5, 2016. This manual supports GTKWave 3.3.79 and higher versions.

26.Тарасов И.Е., Певцов Е.Ф. Программируемые логические схемы и их применение в схемотехнических решениях// Учебное пособие. - М., МГТУ МИРЭА, 2012. - 187 с.

Электронные ресурсы удаленного доступа (представленные в Интернете или внутренних сетях)

1. Алексеевский Петр Иванович Обучение студентов архитектуре вычислительных систем с использованием языка Verilog // Педагогическое образование в России. 2016. №7. URL: https://cyberleninka.ru/article/n/obuchenie-studentov-arhitekture-vychislitelnyh-sistem-s-ispolzovaniem-yazyka-verilog (дата обращения: 28.06.2018).

2.Емец Сергей Verilog — инструмент разработки цифровых электронных схем // Компоненты и Технологии. 2001. №14. URL: https://cyberleninka.ru/article/n/verilog-instrument-razrabotki-tsifrovyh-elektronnyh-shem-2 (дата обращения: 23.03.2018).

3. Каршенбойм Иосиф Краткий курс HDL. Часть 2. Описание языка Verilog // Компоненты и Технологии. 2008. №82. URL: https://cyberleninka.ru/article/n/kratkiy-kurs-hdl-chast-2-opisanie-yazyka-verilog (дата обращения: 28.04.2018).

4. Статья Графический дизайн или текст Verilog/VHDL?URL:https://marsohod.org/11-blog/251-sch-or-tx(дата обращения: 09.02.18)

5.Стешенко Владимир Школа схемотехнического проектирования устройств обработки сигналов. Занятие 12. Языки описания аппаратуры. Язык описания аппаратуры Verilog HDL // Компоненты и Технологии. 2001. №15. URL: https://cyberleninka.ru/article/n/shkola-shemotehnicheskogo-proektirovaniya-ustroystv-obrabotki-signalov-zanyatie-12-yazyki-opisaniya-apparatury-yazyk-opisaniya (дата обращения: 28.06.2018).

6.Основные производители современных ПЛИС-компьютеров и комплектующих к ним [Электронный ресурс]URL:

https://parallel.ru/fpga/vendors.html(дата обращения: 18.02.18)

7.Официальный сайт компании Intel FPGA URL:https://www.altera.com (дата обращения: 12.02.18)

8.Официальный сайт ООО "ЭФО" дистрибьютор и тренинг-партнер компании Intel FPGA в России URL:http://altera.ru (дата обращения: 04.02.18)

9.Официальный сайт АО "НПО "Электромашина" URL:

http://www.npoelm.ru/(дата обращения: 18.02.18)

10. Официальный сайт разработчиков программы PuTTY

URL:https://www.putty.org /(дата обращения: 17.04.18)

11. Официальный сайт компании Microsoft в России URL:https://www.microsoft.com/ru-ru// (дата обращения: 05.12.17)

12. Официальный сайт компании производителя платы разработчика RuiZhi YanFaa URL: http://rzrd.net/en// (дата обращения: 02.03.18)

13. Официальный сайт сообщества EasyElectronics.ru URL:http://we.easyelectronics.ru /(дата обращения: 15.01.18)

14.Официальный сайт системы Профессиональные справочные системы Техэксперт - электронный фонд правовой и нормативно-технической документации URL:http://www.cntd.ru