

Date: Oct 18, 2022

Laboratory Practices

FPGA Design Workshop

Table of Contents

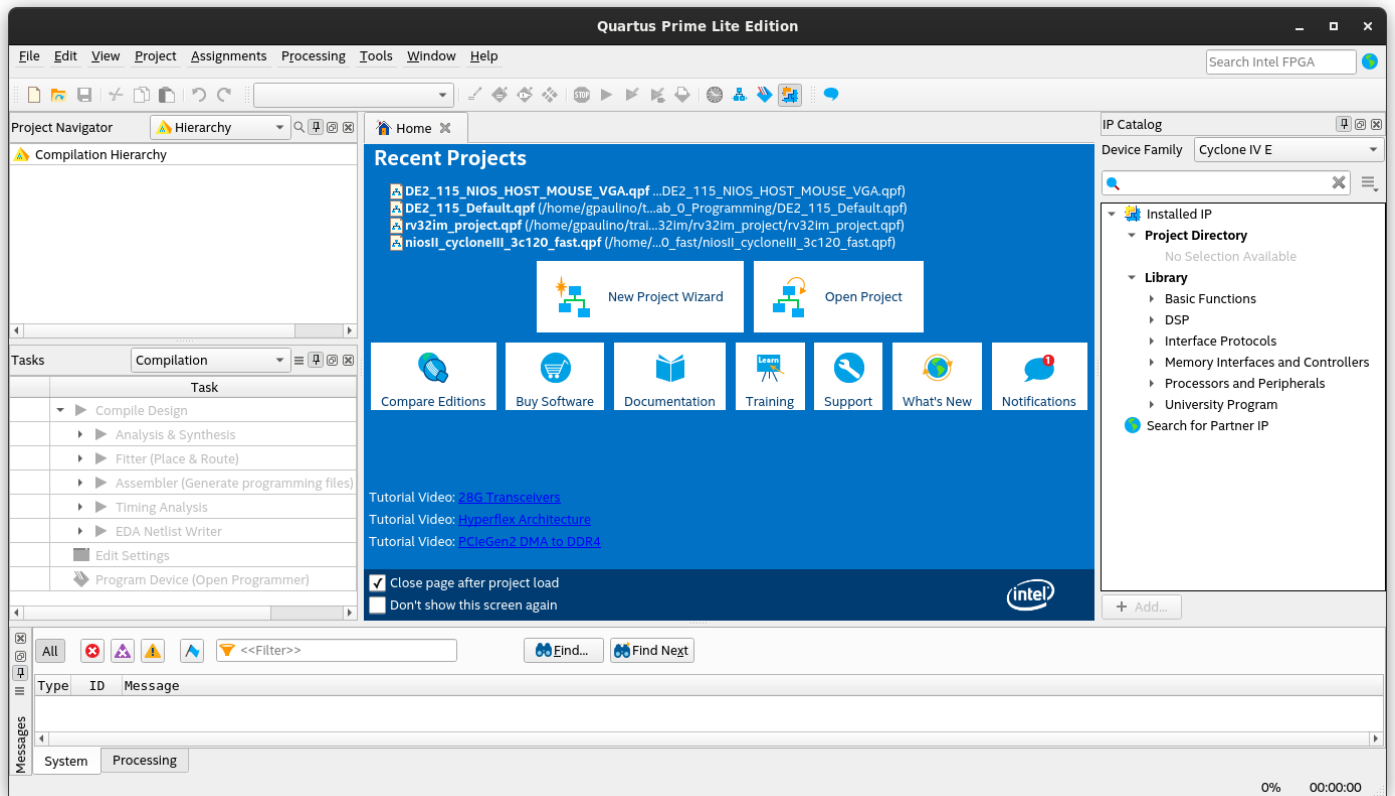
| | |
|--|-----------|
| Lab Practice 0 | |
| Programming the Board | 2 |
| Lab Practice 1 | |
| Introduction to Hardware Description Language | 3 |
| 1.1 Getting Started | 4 |
| 1.2 Starting a New Project | 5 |
| 1.3 Pin Assignment | 13 |
| 1.4 Design Entry Using Verilog Code | 17 |
| 1.4.1 LED Controller design | 18 |
| 1.4.2 Simulating the Designed Circuit | 20 |
| 1.4.3 Programming and Configuring the FPGA Device | 20 |
| 1.4.3.1 JTAG Programming | 20 |
| Lab Practice 2 | |
| Finite State Machines | 21 |
| 2.1 Starting a New Project | 22 |
| 2.2 Pin Assignment | 23 |
| 2.3 Power Sequencer | 24 |
| Lab Practice 3 | |
| Soft-Core Processors | 26 |
| 3.1 Getting Started | 27 |
| 3.2 Starting a New Project | 28 |
| 3.3 Pin Assignment | 28 |
| 3.4 Platform Designer | 29 |
| 3.4.1 Instantiation of the Module Generated by the Platform Designer | 35 |
| 3.5 Nios II Software Build Tools for Eclipse | 35 |
| References | 37 |

Lab Practice 0

Programming the Board

Follow the instructions:

1. Open the software Quartus Lite



2. Open project in Quartus, and find the project file inside the folder:
 - a. DE2-115
 - i. DE2_115_Lab_0_Programming > DE2_115_Default.qpf
 - b. Mercurio IV
 - i. Mercurio_IV_Lab_0_Programming > quartus > Labx1.qpf
3. Compile the design
 - a. Processing > Start Compilation
4. Plug USB cable from PC host to DE2-115 board
5. Ensure that power is applied to the board
6. Program the Device
 - a. Tools > Programmer
 - i. Hardware Setup should identify the board
 - ii. Find the *.sof file from this project, and use JTAG mode
 - iii. Click Start

For more detailed steps, follow the user manual from vendors.

Lab Practice 1

Introduction to Hardware Description Language

1.1 Getting Started

This tutorial presents an introduction to the Quartus software tool, and to Intel's FPGA programmable logic design workflow (see Figure 1.1). It makes use of the Verilog design entry method, in which the user specifies the desired circuit in the Verilog hardware description language. The last step in the design process involves configuring the designed circuit in an actual FPGA device. [1]

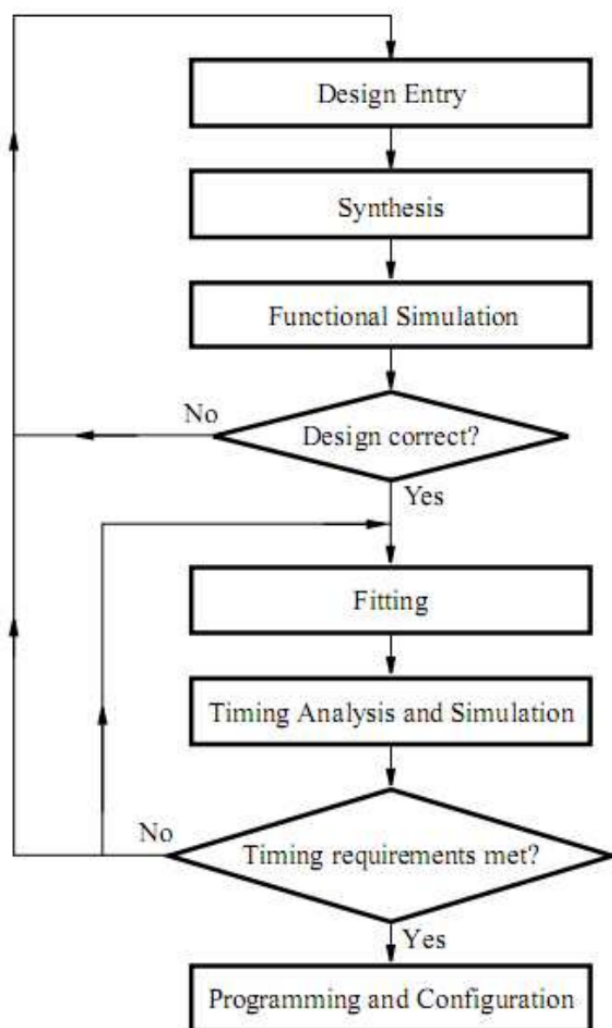


Fig. 1.1 - Typical FPGA design workflow. Source: [1]

The FPGA design workflow involves the following steps:

Design Entry – the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as Verilog or VHDL.

Synthesis – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip.

Functional Simulation – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues.

Fitting – the Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs.

Timing Analysis – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit.

Timing Simulation – the fitted circuit is tested to verify both its functional correctness and timing.

Programming and Configuration – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections.

1.2 Starting a New Project

Each logic circuit, or sub circuit, being designed with Quartus II software is called a project. The software works on one project at a time and keeps all information for that project in a single directory (folder) in the file system.

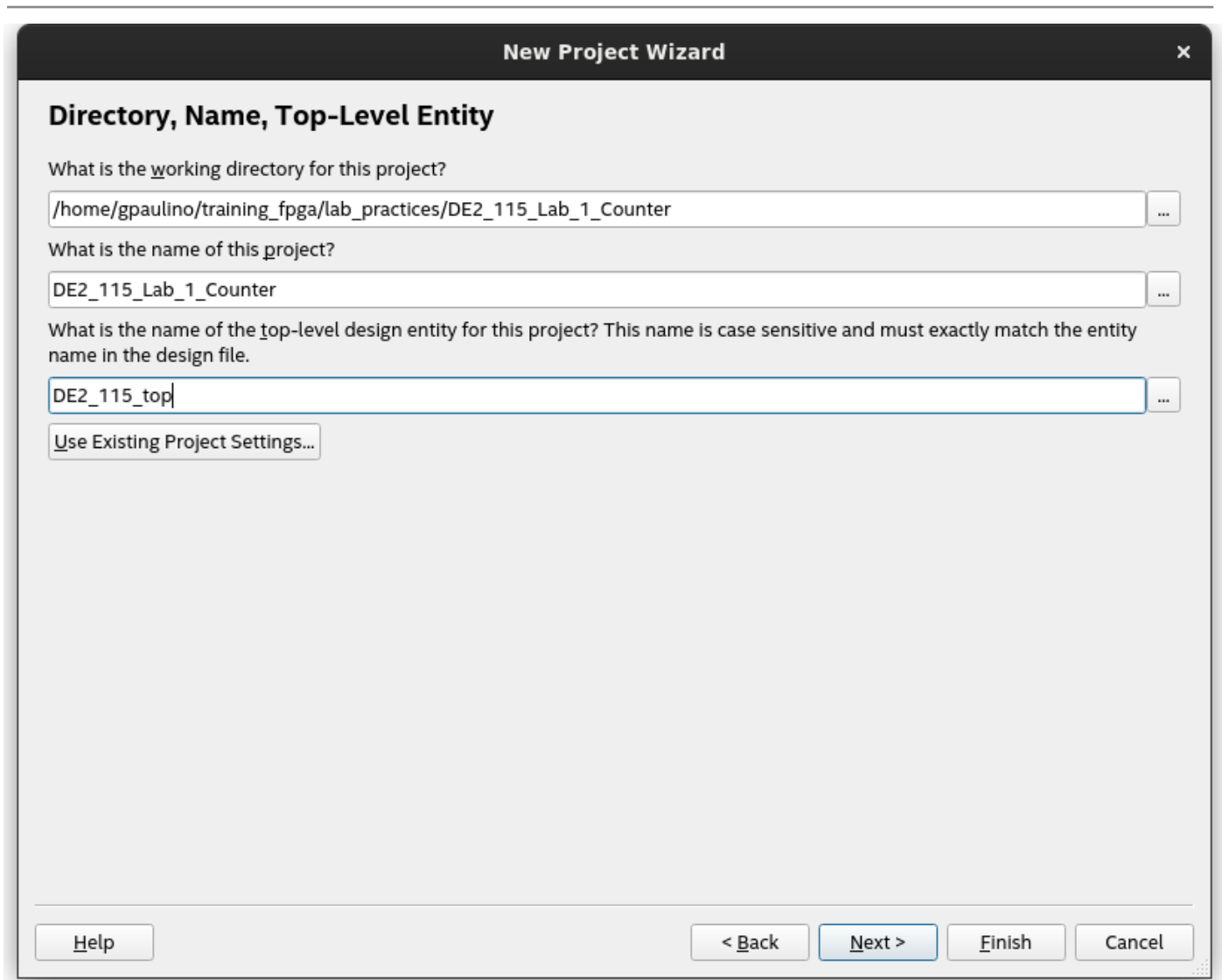
To hold the design files for this tutorial, we will use a one of the directories (*depending on board*):

- DE2_115_Lab_1_Counter
- Mercurio_IV_Lab_1_Counter

Start the Quartus software.

To start working on a new design we first have to define a new design project. Create a new project:

1. Select File > New Project Wizard
 - a. Name of this project
 - i. DE2_115_Lab_1_Counter
 - ii. Mercurio_IV_Lab_1_Counter
 - b. Top-level design entity
 - i. DE2_115_top
 - ii. MercurioIV_top
2. Empty Project
3. Add All
4. Select the family and device
5. No third-party tools should be used
6. Review the summary



The image shows a 'New Project Wizard' dialog box with a dark title bar. The main area is light gray and contains three text input fields with labels and a 'Use Existing Project Settings...' button. The first field is for the working directory, the second for the project name, and the third for the top-level design entity. Each field has a small '...' button to its right. At the bottom, there are five buttons: 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

New Project Wizard ✕

Directory, Name, Top-Level Entity

What is the working directory for this project?

/home/gpaulino/training_fpga/lab_practices/DE2_115_Lab_1_Counter ...

What is the name of this project?

DE2_115_Lab_1_Counter ...

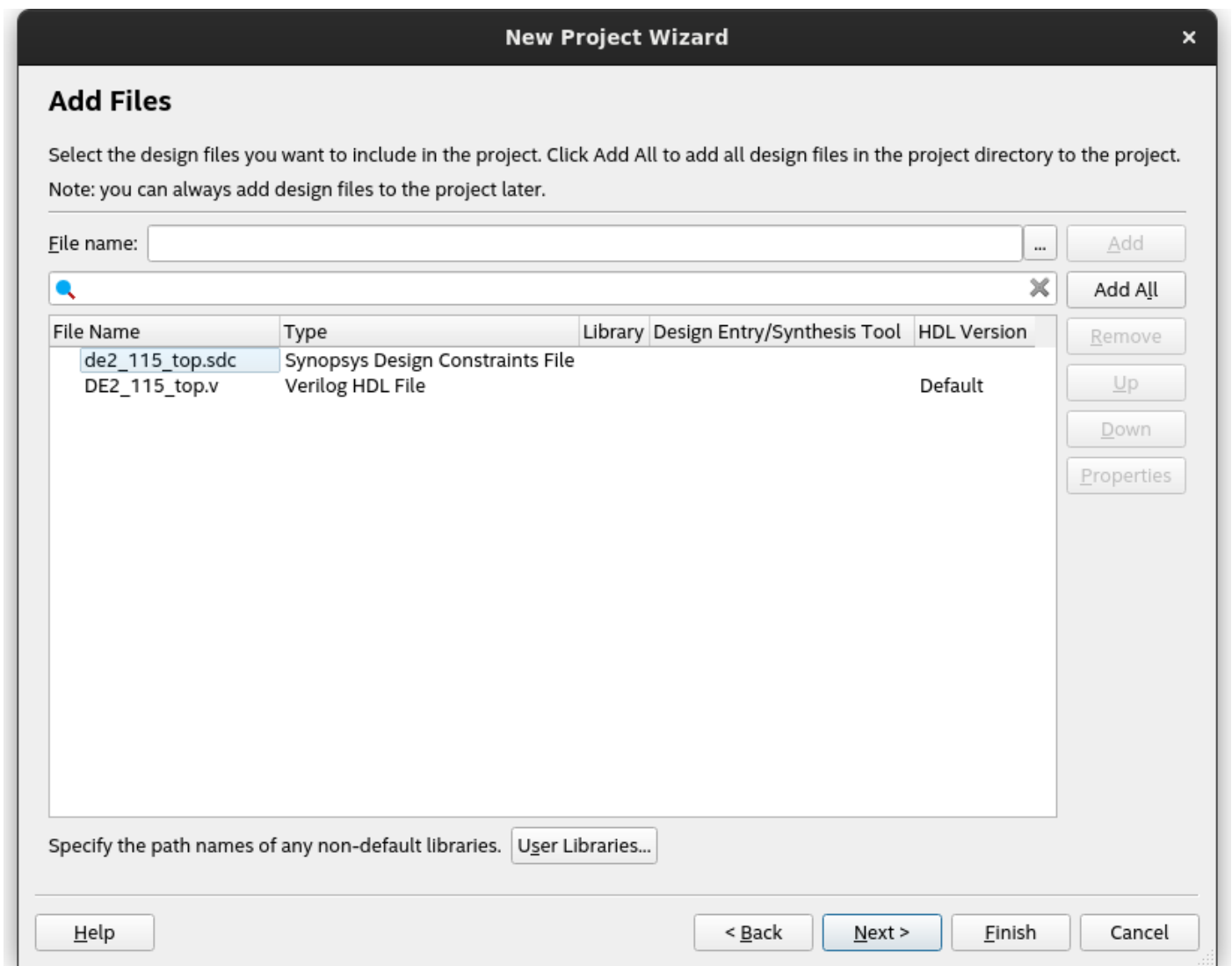
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

DE2_115_top ...

Use Existing Project Settings...

Help < Back Next > Finish Cancel

Add all the Files:



Choose the device family and a specific device for **DE2-115**: EP4CE115F29C7

New Project Wizard

✕

Family, Device & Board Settings

Device

Board

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.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: Cyclone IV E

Device: 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:

Any

Core speed grade:

Any

Name filter:

☒ Show advanced devices

Available devices:

| Name | Core Voltage | LEs | Total I/Os | GPIOs | Memory Bits | Embedded multiplier 9-bit element |
|---------------|--------------|--------|------------|-------|-------------|-----------------------------------|
| EP4CE75F29C7 | 1.2V | 75408 | 427 | 427 | 2810880 | 400 |
| EP4CE75F29C8 | 1.2V | 75408 | 427 | 427 | 2810880 | 400 |
| EP4CE75F29I7 | 1.2V | 75408 | 427 | 427 | 2810880 | 400 |
| EP4CE75U19I7 | 1.2V | 75408 | 293 | 293 | 2810880 | 400 |
| EP4CE115F23C7 | 1.2V | 114480 | 281 | 281 | 3981312 | 532 |
| EP4CE115F23C8 | 1.2V | 114480 | 281 | 281 | 3981312 | 532 |
| EP4CE115F23I7 | 1.2V | 114480 | 281 | 281 | 3981312 | 532 |
| EP4CE115F29C7 | 1.2V | 114480 | 529 | 529 | 3981312 | 532 |
| EP4CE115F29C8 | 1.2V | 114480 | 529 | 529 | 3981312 | 532 |
| EP4CE115F29I7 | 1.2V | 114480 | 529 | 529 | 3981312 | 532 |
| EP4CE6E22C8L | 1.0V | 6272 | 92 | 92 | 276480 | 30 |
| EP4CE6E22C9L | 1.0V | 6272 | 92 | 92 | 276480 | 30 |

Help

< Back

Next >

Finish

Cancel

Choose the device family and a specific device for **Mercurio IV**: EP4CE30F23C7

New Project Wizard

✕

Family, Device & Board Settings

Device

Board

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.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: Cyclone IV E

Device: 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:

Any

Core speed grade:

Any

Name filter:

☒ Show advanced devices

Available devices:

| Name | Core Voltage | LEs | Total I/Os | GPIOs | Memory Bits | Embedded multiplier 9-bit element |
|--------------|--------------|-------|------------|-------|-------------|-----------------------------------|
| EP4CE30F19A7 | 1.2V | 28848 | 193 | 193 | 608256 | 132 |
| EP4CE30F23A7 | 1.2V | 28848 | 329 | 329 | 608256 | 132 |
| EP4CE30F23C6 | 1.2V | 28848 | 329 | 329 | 608256 | 132 |
| EP4CE30F23C7 | 1.2V | 28848 | 329 | 329 | 608256 | 132 |
| EP4CE30F23C8 | 1.2V | 28848 | 329 | 329 | 608256 | 132 |
| EP4CE30F23C7 | 1.2V | 28848 | 329 | 329 | 608256 | 132 |

Help

< Back

Next >

Finish

Cancel

There is no need to select a third-party tool for design or simulation. Just click **Next**.

New Project Wizard

EDA Tool Settings

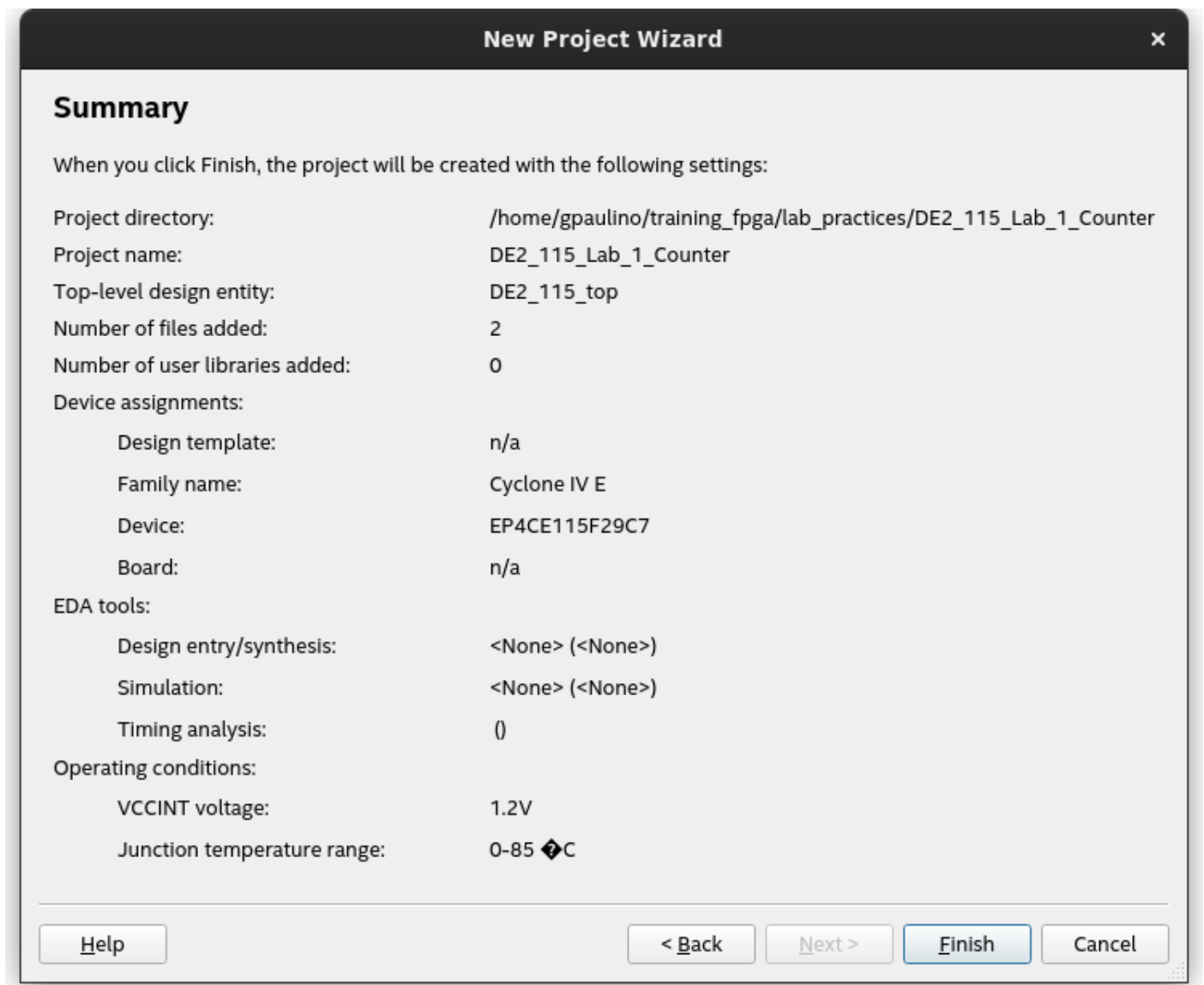
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

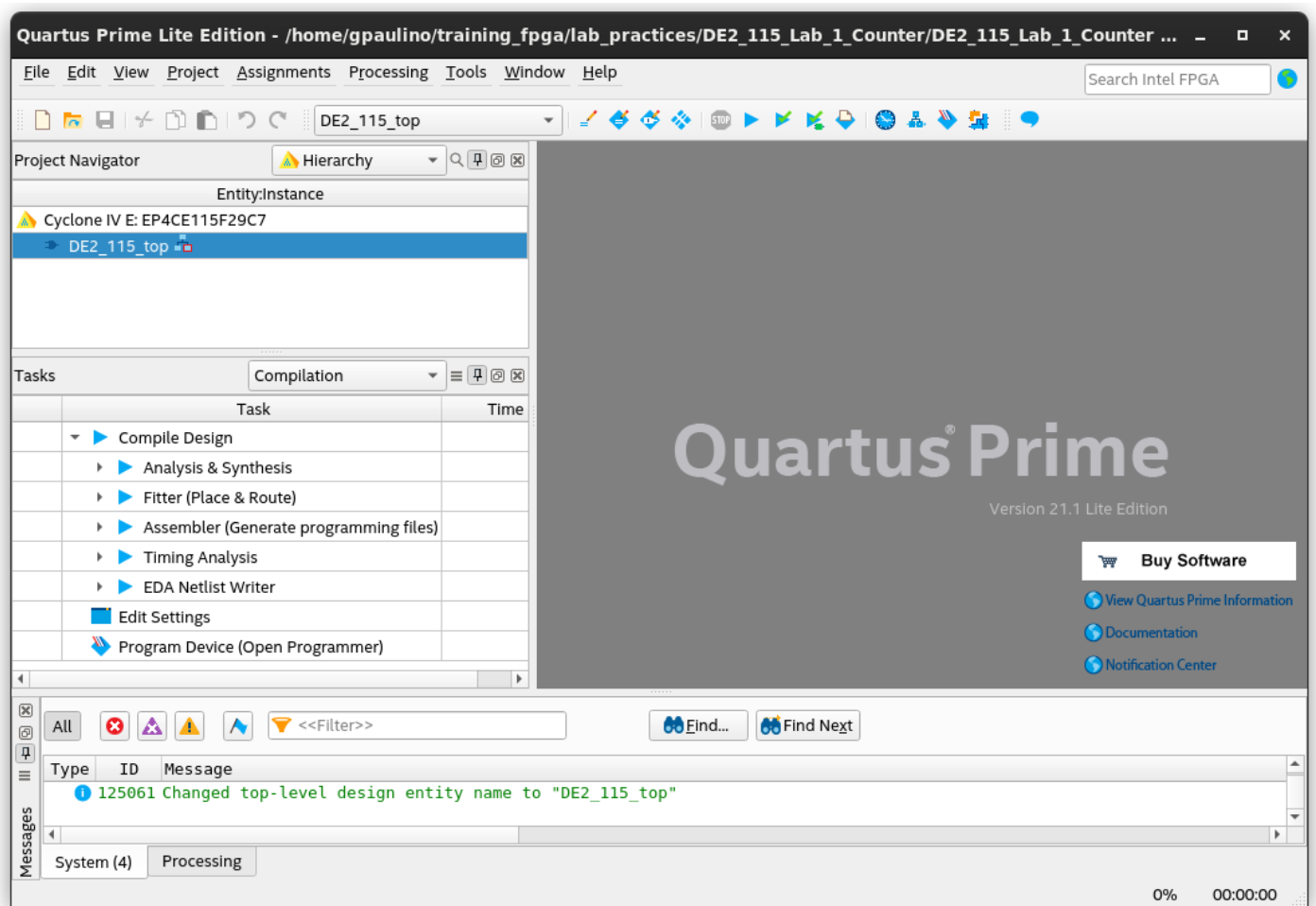
| Tool Type | Tool Name | Format(s) | Run Tool Automatically |
|------------------------|------------------|-----------|--|
| Design Entry/Synthesis | <None> | <None> | <input type="checkbox"/> Run this tool automatically to synthesize t |
| Simulation | <None> | <None> | <input type="checkbox"/> Run gate-level simulation automatically af |
| Board-Level | Timing | <None> | |
| | Symbol | <None> | |
| | Signal Integrity | <None> | |
| | Boundary Scan | <None> | |

Help < Back Next > Finish Cancel

A summary of the chosen settings appears on the screen. Just press **Finish**.

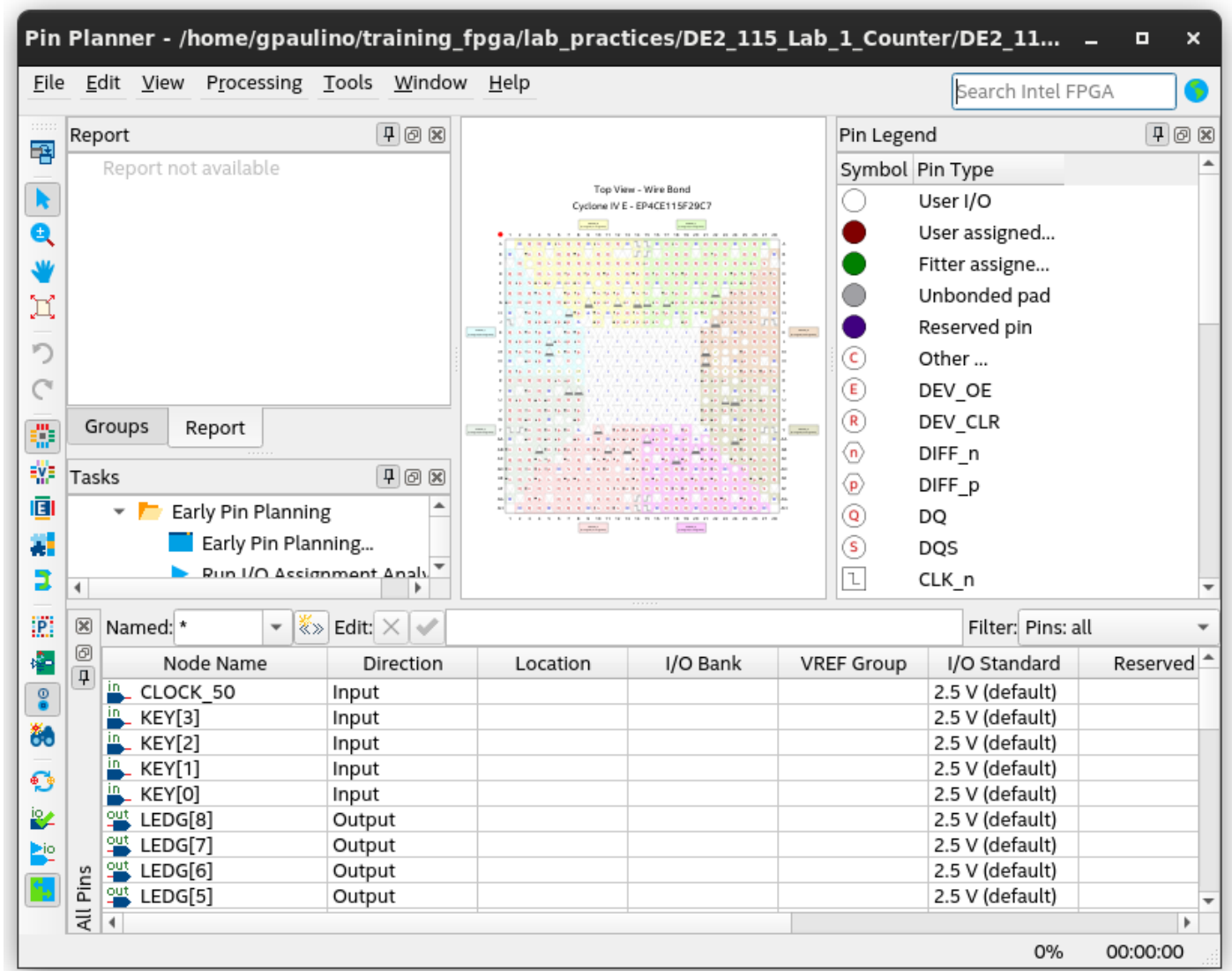


Quartus software tool:



1.3 Pin Assignment

Open "Pin Planner" tool:



After the assignment, you can start the I/O Assignment Analysis.

Note: remember to export the pinout assignment file to be used for the next laboratory practice.

Assign the pinout for **DE2-115**: EP4CE115F29C7

| To | Direction | Location | I/O Bank | VREF Group | I/O Standard |
|----------|-----------|----------|----------|------------|--------------|
| CLOCK_50 | Input | PIN_Y2 | 2 | B2_N0 | 3.3-V LVTTL |
| KEY[3] | Input | PIN_R24 | 5 | B5_N0 | 2.5 V |
| KEY[2] | Input | PIN_N21 | 6 | B6_N2 | 2.5 V |
| KEY[1] | Input | PIN_M21 | 6 | B6_N1 | 2.5 V |
| KEY[0] | Input | PIN_M23 | 6 | B6_N2 | 2.5 V |
| LEDG[8] | Output | PIN_F17 | 7 | B7_N2 | 2.5 V |
| LEDG[7] | Output | PIN_G21 | 7 | B7_N1 | 2.5 V |
| LEDG[6] | Output | PIN_G22 | 7 | B7_N2 | 2.5 V |
| LEDG[5] | Output | PIN_G20 | 7 | B7_N1 | 2.5 V |
| LEDG[4] | Output | PIN_H21 | 7 | B7_N2 | 2.5 V |
| LEDG[3] | Output | PIN_E24 | 7 | B7_N1 | 2.5 V |
| LEDG[2] | Output | PIN_E25 | 7 | B7_N1 | 2.5 V |
| LEDG[1] | Output | PIN_E22 | 7 | B7_N0 | 2.5 V |
| LEDG[0] | Output | PIN_E21 | 7 | B7_N0 | 2.5 V |
| LEDR[17] | Output | PIN_H15 | 7 | B7_N2 | 2.5 V |
| LEDR[16] | Output | PIN_G16 | 7 | B7_N2 | 2.5 V |
| LEDR[15] | Output | PIN_G15 | 7 | B7_N2 | 2.5 V |
| LEDR[14] | Output | PIN_F15 | 7 | B7_N2 | 2.5 V |
| LEDR[13] | Output | PIN_H17 | 7 | B7_N2 | 2.5 V |
| LEDR[12] | Output | PIN_J16 | 7 | B7_N2 | 2.5 V |
| LEDR[11] | Output | PIN_H16 | 7 | B7_N2 | 2.5 V |
| LEDR[10] | Output | PIN_J15 | 7 | B7_N2 | 2.5 V |
| LEDR[9] | Output | PIN_G17 | 7 | B7_N1 | 2.5 V |
| LEDR[8] | Output | PIN_J17 | 7 | B7_N2 | 2.5 V |
| LEDR[7] | Output | PIN_H19 | 7 | B7_N2 | 2.5 V |
| LEDR[6] | Output | PIN_J19 | 7 | B7_N2 | 2.5 V |
| LEDR[5] | Output | PIN_E18 | 7 | B7_N1 | 2.5 V |
| LEDR[4] | Output | PIN_F18 | 7 | B7_N1 | 2.5 V |
| LEDR[3] | Output | PIN_F21 | 7 | B7_N0 | 2.5 V |
| LEDR[2] | Output | PIN_E19 | 7 | B7_N0 | 2.5 V |
| LEDR[1] | Output | PIN_F19 | 7 | B7_N0 | 2.5 V |
| LEDR[0] | Output | PIN_G19 | 7 | B7_N2 | 2.5 V |

Pin Planner - /home/gpaulino/training_fpga/lab_practices/DE2_115_Lab_1_Counter/DE2_115_Lab_1_Counter - DE2_115_top

File Edit View Processing Tools Window Help

Search Intel FPGA

Report not available

Groups Report

Tasks

- Early Pin Planning
 - Early Pin Planning...
 - Run I/O Assignment Analy...
 - Export Pin Assignments...

Top View - Wire Bond
Cyclone IV E - EP4CE115F29C7

Pin Legend

| Symbol | Pin Type |
|--------|-------------------|
| ○ | User I/O |
| ● | User assigned... |
| ● | Fitter assigne... |
| ○ | Unbonded pad |
| ● | Reserved pin |
| C | Other ... |
| E | DEV_OE |
| R | DEV_CLR |
| n | DIFF_n |
| p | DIFF_p |
| Q | DQ |
| S | DQS |
| L | CLK_n |
| f | CLK_p |

Named: * Edit: X <<new node>>

Filter: Pins: all

| Node Name | Direction | Location | I/O Bank | VREF Group | Fitter Location | I/O Standard | Reserved | Current Strength | Slew Rate | Differential Pair |
|-----------|-----------|----------|----------|------------|-----------------|--------------|----------|------------------|-------------|-------------------|
| CLOCK_50 | Input | PIN_Y2 | 2 | B2_N0 | PIN_Y2 | 3.3-V LVTTTL | | 8mA (default) | | |
| KEY[3] | Input | PIN_R24 | 5 | B5_N0 | PIN_R24 | 2.5 V | | 8mA (default) | | |
| KEY[2] | Input | PIN_N21 | 6 | B6_N2 | PIN_N21 | 2.5 V | | 8mA (default) | | |
| KEY[1] | Input | PIN_M21 | 6 | B6_N1 | PIN_M21 | 2.5 V | | 8mA (default) | | |
| KEY[0] | Input | PIN_M23 | 6 | B6_N2 | PIN_M23 | 2.5 V | | 8mA (default) | | |
| LEDG[8] | Output | PIN_F17 | 7 | B7_N2 | PIN_F17 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[7] | Output | PIN_G21 | 7 | B7_N1 | PIN_G21 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[6] | Output | PIN_G22 | 7 | B7_N2 | PIN_G22 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[5] | Output | PIN_G20 | 7 | B7_N1 | PIN_G20 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[4] | Output | PIN_H21 | 7 | B7_N2 | PIN_H21 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[3] | Output | PIN_E24 | 7 | B7_N1 | PIN_E24 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[2] | Output | PIN_E25 | 7 | B7_N1 | PIN_E25 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[1] | Output | PIN_E22 | 7 | B7_N0 | PIN_E22 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDG[0] | Output | PIN_E21 | 7 | B7_N0 | PIN_E21 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[17] | Output | PIN_H15 | 7 | B7_N2 | PIN_H15 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[16] | Output | PIN_G16 | 7 | B7_N2 | PIN_G16 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[15] | Output | PIN_G15 | 7 | B7_N2 | PIN_G15 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[14] | Output | PIN_F15 | 7 | B7_N2 | PIN_F15 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[13] | Output | PIN_H17 | 7 | B7_N2 | PIN_H17 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[12] | Output | PIN_J16 | 7 | B7_N2 | PIN_J16 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[11] | Output | PIN_H16 | 7 | B7_N2 | PIN_H16 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[10] | Output | PIN_J15 | 7 | B7_N2 | PIN_J15 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[9] | Output | PIN_G17 | 7 | B7_N1 | PIN_G17 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[8] | Output | PIN_J17 | 7 | B7_N2 | PIN_J17 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[7] | Output | PIN_H19 | 7 | B7_N2 | PIN_H19 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[6] | Output | PIN_J19 | 7 | B7_N2 | PIN_J19 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[5] | Output | PIN_E18 | 7 | B7_N1 | PIN_E18 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[4] | Output | PIN_F18 | 7 | B7_N1 | PIN_F18 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[3] | Output | PIN_F21 | 7 | B7_N0 | PIN_F21 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[2] | Output | PIN_E19 | 7 | B7_N0 | PIN_E19 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[1] | Output | PIN_F19 | 7 | B7_N0 | PIN_F19 | 2.5 V | | 8mA (default) | 2 (default) | |
| LEDR[0] | Output | PIN_G19 | 7 | B7_N2 | PIN_G19 | 2.5 V | | 8mA (default) | 2 (default) | |

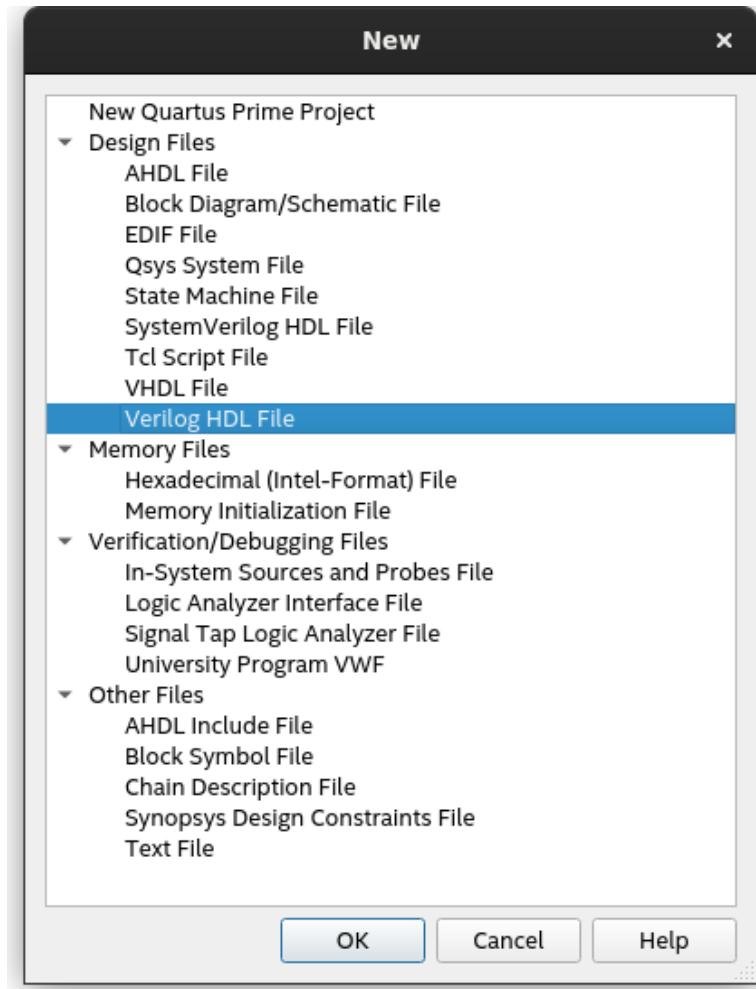
0% 00:00:00

Assign the pinout for **Mercurio IV**: EP4CE30F23C7

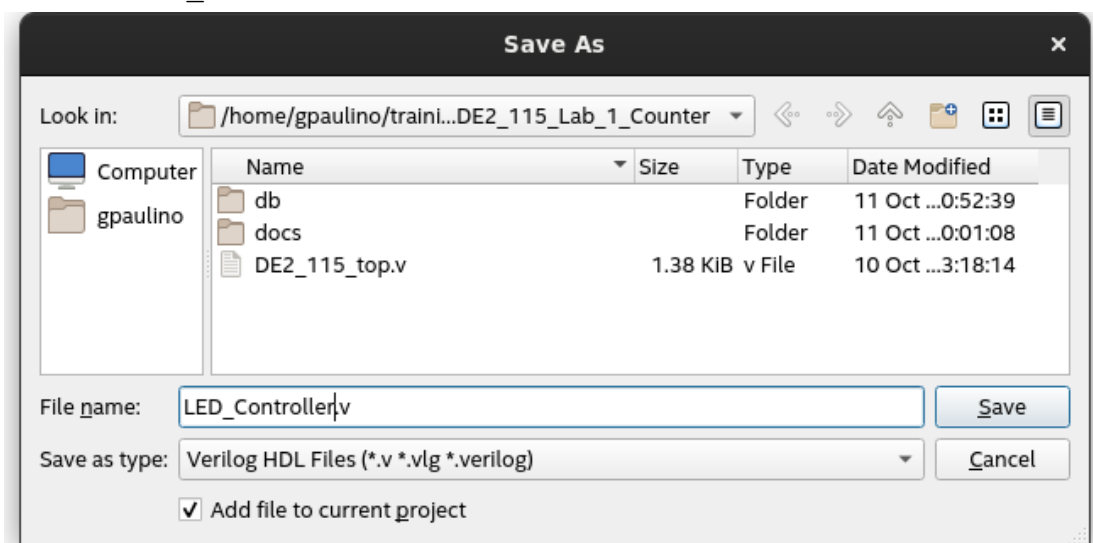
| To | Direction | Location | I/O Bank | VREF Group | I/O Standard |
|--------------|-----------|----------|----------|------------|--------------|
| CLOCK_50 MHz | Input | PIN_T1 | 2 B2_N0 | | PIN_T1 |
| KEY[11] | Input | PIN_Y17 | 4 B4_N0 | | PIN_Y17 |
| KEY[10] | Input | PIN_U17 | 4 B4_N0 | | PIN_U17 |
| KEY[9] | Input | PIN_W15 | 4 B4_N1 | | PIN_W15 |
| KEY[8] | Input | PIN_W19 | 5 B5_N3 | | PIN_W19 |
| KEY[7] | Input | PIN_W17 | 4 B4_N0 | | PIN_W17 |
| KEY[6] | Input | PIN_V15 | 4 B4_N2 | | PIN_V15 |
| KEY[5] | Input | PIN_U21 | 5 B5_N1 | | PIN_U21 |
| KEY[4] | Input | PIN_W20 | 5 B5_N3 | | PIN_W20 |
| KEY[3] | Input | PIN_U16 | 4 B4_N0 | | PIN_U16 |
| KEY[2] | Input | PIN_U22 | 5 B5_N2 | | PIN_U22 |
| KEY[1] | Input | PIN_U20 | 5 B5_N2 | | PIN_U20 |
| KEY[0] | Input | PIN_V22 | 5 B5_N2 | | PIN_V22 |
| LEDM_C[4] | Output | PIN_L8 | 1 B1_N2 | | PIN_L8 |
| LEDM_C[3] | Output | PIN_J8 | 1 B1_N1 | | PIN_J8 |
| LEDM_C[2] | Output | PIN_K8 | 1 B1_N2 | | PIN_K8 |
| LEDM_C[1] | Output | PIN_J6 | 1 B1_N1 | | PIN_J6 |
| LEDM_C[0] | Output | PIN_J7 | 1 B1_N2 | | PIN_J7 |
| LEDM_R[7] | Output | PIN_H11 | 8 B8_N1 | | PIN_H11 |
| LEDM_R[6] | Output | PIN_G8 | 8 B8_N3 | | PIN_G8 |
| LEDM_R[5] | Output | PIN_F8 | 8 B8_N3 | | PIN_F8 |
| LEDM_R[4] | Output | PIN_F9 | 8 B8_N3 | | PIN_F9 |
| LEDM_R[3] | Output | PIN_G9 | 8 B8_N3 | | PIN_G9 |
| LEDM_R[2] | Output | PIN_E9 | 8 B8_N1 | | PIN_E9 |
| LEDM_R[1] | Output | PIN_C8 | 8 B8_N1 | | PIN_C8 |
| LEDM_R[0] | Output | PIN_F10 | 8 B8_N2 | | PIN_F10 |

1.4 Design Entry Using Verilog Code

Create a New Verilog HDL file:



The filename will be: `LED_Controller.v`



1.4.1 LED Controller design

For the LED controller, a digital circuit with the following behavior must be implemented:

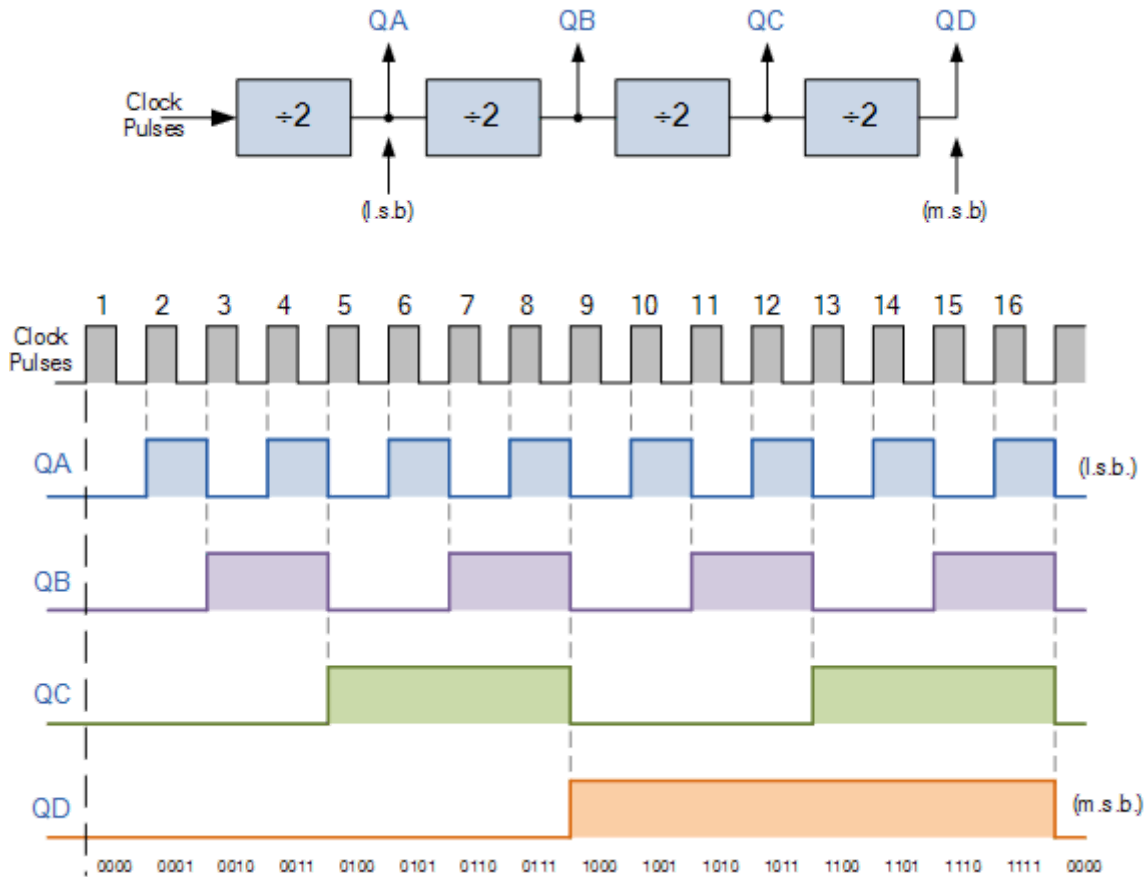


Fig. 1.2 - Clock Frequency Divider.

When considering a 27-bit logic vector:

$$f_{CLK} = 50 \text{ MHz}$$

$$T = 1/f_{CLK} = 1/50 \text{ MHz} = 20 \text{ ns}$$

$$T_{X-2} = T \times (2^{23}) = 0.167 \text{ s}$$

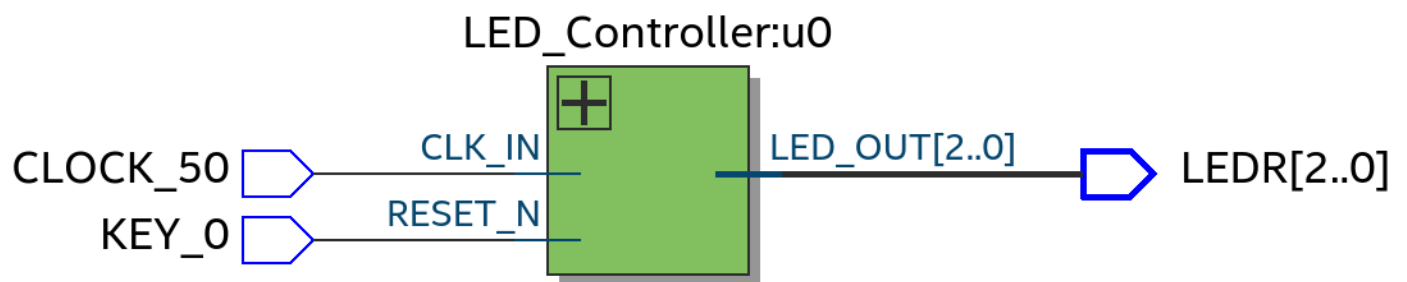
$$T_{X-1} = T \times (2^{24}) = 0.335 \text{ s}$$

$$T_{LED[0]} = T \times (2^{25}) = 0.671 \text{ s}$$

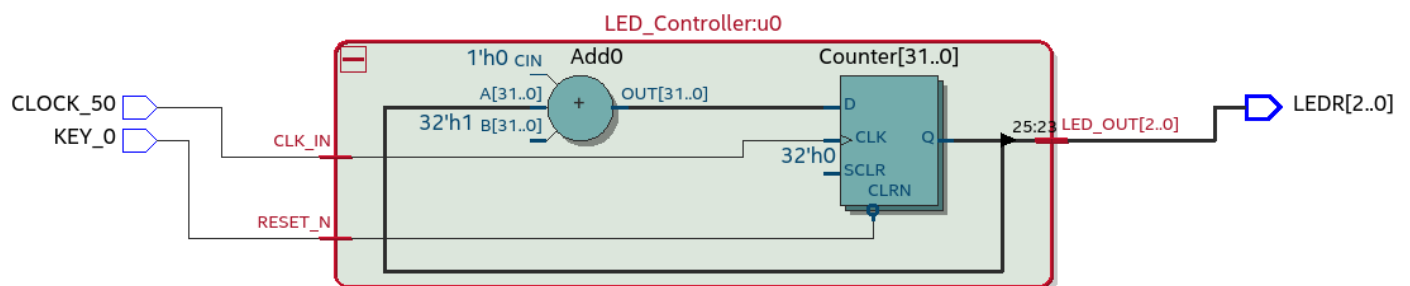
$$T_{LED[1]} = T \times (2^{26}) = 1.34 \text{ s}$$

$$T_{LED[2]} = T \times (2^{27}) = 2.68 \text{ s}$$

The HDL should describe a LED Controller.



The circuit will be an adder which increments a register by the input clock.



1.4.2 Simulating the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE2-115 board, it is prudent to simulate it to ascertain its correctness.

1.4.3 Programming and Configuring the FPGA Device

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus Compiler's Assembler module.

DE2-115

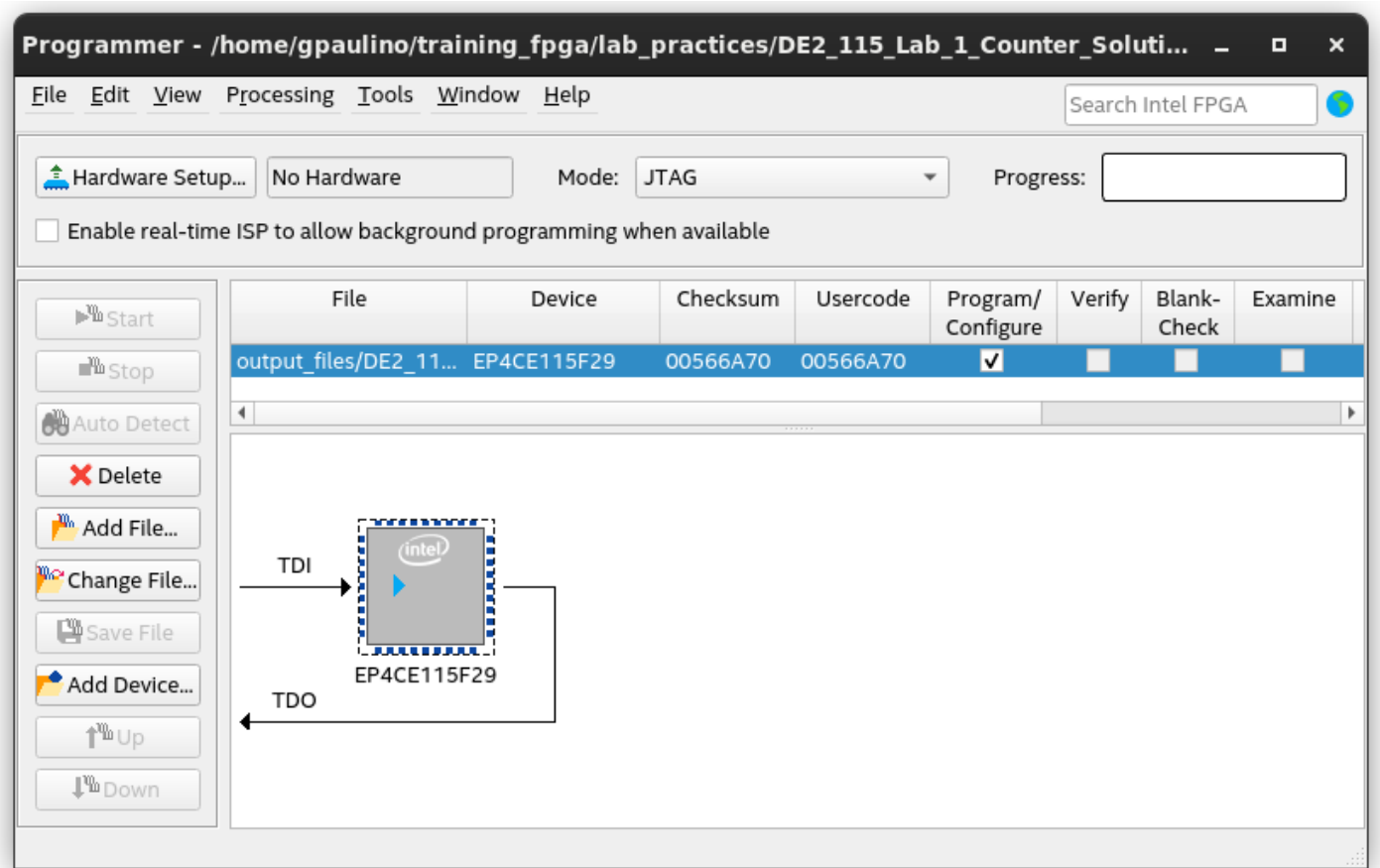
The choice between the two modes is made by the RUN/PROG switch on the DE2-115 board. The RUN position selects the JTAG mode, while the PROG position selects the Active Serial (AS) mode - which is the flash memory used to store the configuration data.

Mercurio IV

In case of Mercurio IV board, the PROG FPGA position selects the JTAG mode, while the PROG FLASH position selects the AS mode.

1.4.3.1 JTAG Programming

The programming and configuration task is performed using the JTAG mode for this practice.



Lab Practice 2

Finite State Machines

2.1 Starting a New Project

To hold the design files for this tutorial, we will use a one of the directories (*depending on board*):

- DE2_115_Lab_2_Sequencer
- Mercurio_IV_Lab_2_Sequencer

To start working on a new design we first have to define a new design project. Create a new project:

1. Select File > New Project Wizard
 - a. Name of this project
 - i. DE2_115_Lab_2_Sequencer
 - ii. Mercurio_IV_Lab_2_Sequencer
 - b. Top-level design entity
 - i. DE2_115_fsm_top
 - ii. MercurioIV_fsm_top
2. Empty Project
3. Add All
4. Select the family and device
 - a. DE2-115: EP4CE115F29C7
 - b. Mercurio IV: EP4CE30F23C7

New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

/home/gpaulino/training_fpga/lab_practices/DE2_115_Lab_2_Sequencer

What is the name of this project?

DE2_115_Lab_2_Sequencer

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.

DE2_115_fsm_top

Use Existing Project Settings...

2.2 Pin Assignment

For this project, use the CSV file generated from the previous **Lab #1**.

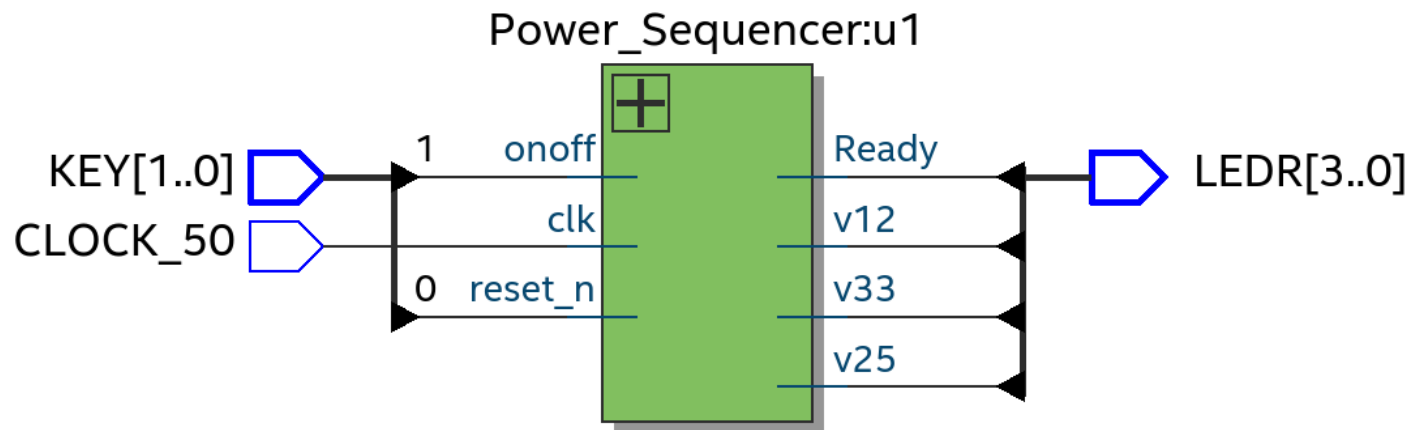
Select Assignments > Import Assignemnts> Select the CSV file.

The screenshot shows the Pin Planner tool interface for the Cyclone IV E - EP4CE115F29C7 device. The main window displays the Top View - Wire Bond. The Pin Legend on the right lists various pin types and their symbols. The table below shows the pin assignments for the device.

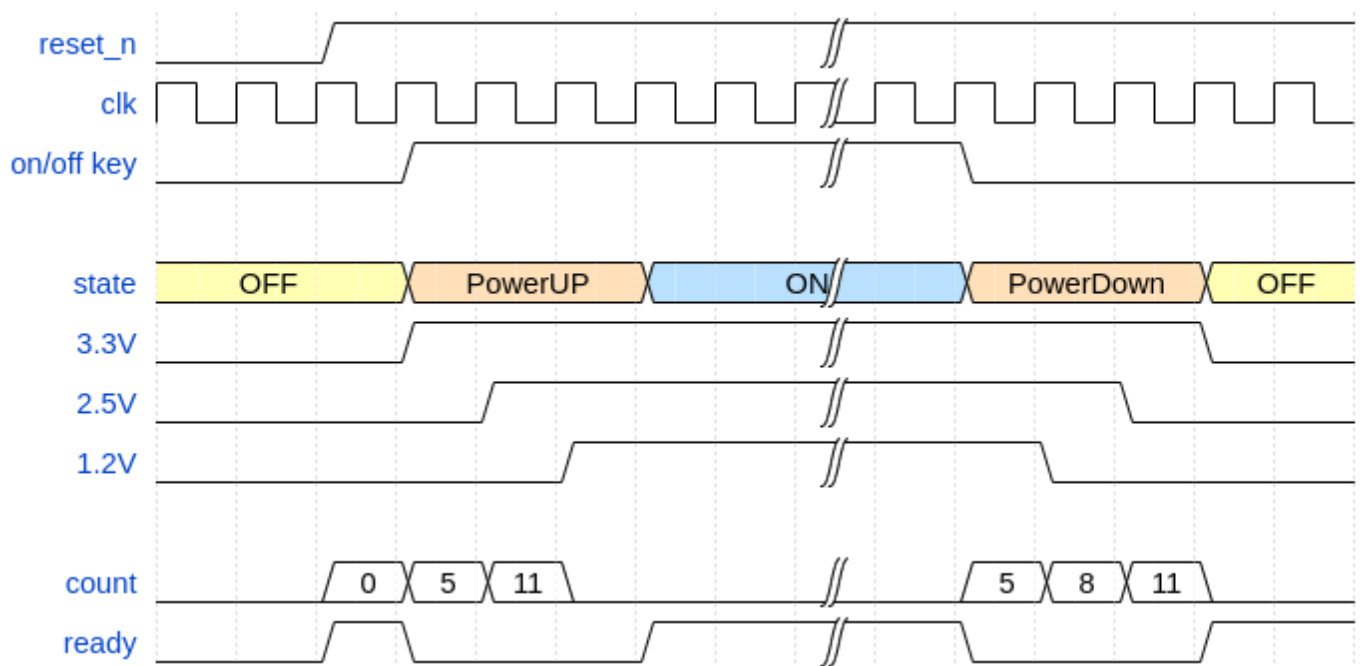
| Node Name | Direction | Location | I/O Bank | VREF Group | Fitter Location | I/O Standard | Reserved | Current Strength | Slew Rate | Differential Pair | Tri-state Preservation |
|-----------|-----------|----------|----------|------------|-----------------|--------------|----------|------------------|-------------|-------------------|------------------------|
| CLOCK_50 | Input | PIN_Y2 | 2 | B2_N0 | PIN_Y2 | 3.3-V LVTTTL | | 8mA (default) | | | |
| KEY[3] | Input | PIN_R24 | 5 | B5_N0 | PIN_R24 | 2.5 V | | 8mA (default) | | | |
| KEY[2] | Input | PIN_N21 | 6 | B6_N2 | PIN_N21 | 2.5 V | | 8mA (default) | | | |
| KEY[1] | Input | PIN_M21 | 6 | B6_N1 | PIN_M21 | 2.5 V | | 8mA (default) | | | |
| KEY[0] | Input | PIN_M23 | 6 | B6_N2 | PIN_M23 | 2.5 V | | 8mA (default) | | | |
| LEDG[8] | Output | PIN_F17 | 7 | B7_N2 | PIN_F17 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[7] | Output | PIN_G21 | 7 | B7_N1 | PIN_G21 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[6] | Output | PIN_G22 | 7 | B7_N2 | PIN_G22 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[5] | Output | PIN_G20 | 7 | B7_N1 | PIN_G20 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[4] | Output | PIN_H21 | 7 | B7_N2 | PIN_H21 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[3] | Output | PIN_E24 | 7 | B7_N1 | PIN_E24 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[2] | Output | PIN_E25 | 7 | B7_N1 | PIN_E25 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[1] | Output | PIN_E22 | 7 | B7_N0 | PIN_E22 | 2.5 V | | 8mA (default) | 2 (default) | | |
| LEDG[0] | Output | PIN_E21 | 7 | B7_N0 | PIN_E21 | 2.5 V | | 8mA (default) | 2 (default) | | |

2.3 Power Sequencer

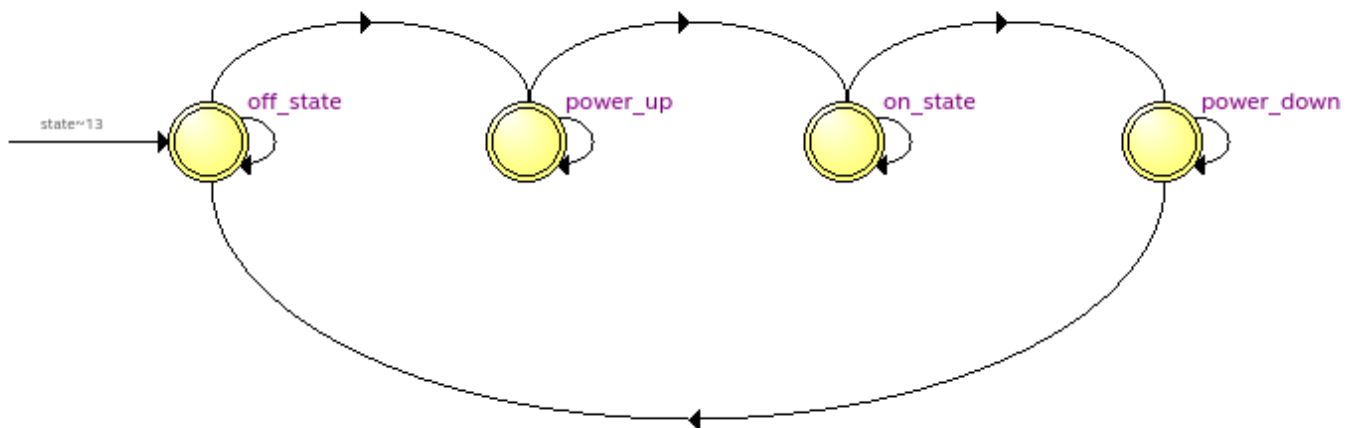
The circuit should describe a module:



Waveform for 3 power domains:



Finite State Machine



Transition table:

| Source State | Destination State | Condition |
|--------------|-------------------|---------------|
| off_state | power_up | (onoff) |
| off_state | off_state | (!onoff) |
| on_state | on_state | (onoff) |
| on_state | power_down | (!onoff) |
| power_down | power_down | counter < 11 |
| power_down | off_state | counter >= 11 |
| power_up | on_state | counter >= 11 |
| power_up | power_up | counter < 11 |

Note: try using the “Signal Tap Logic Analyzer” to verify the circuit behavior after programming the board.

Lab Practice 3

Soft-Core Processors

3.1 Getting Started

Altera's Nios II is a soft processor, defined in a hardware description language, which can be implemented in Altera's FPGA devices by using the Quartus (such as in the Figure 3.1). [3]

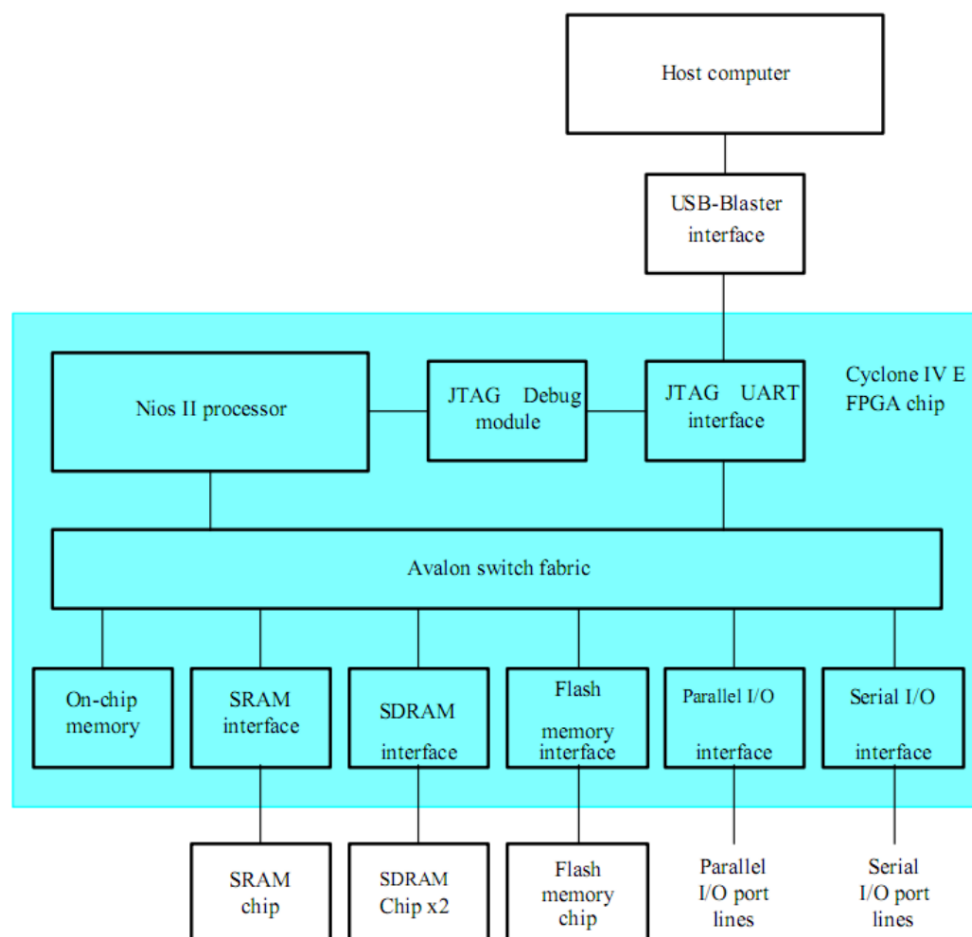


Fig. 3.1 - Nios II system implemented on the DE2-115 board. Source: [3]

The Nios II processor and the interfaces needed to connect to other chips on the DE2-115 board are implemented in the Cyclone IV E FPGA chip. These components are interconnected by means of the interconnection network called the Avalon Switch Fabric.

The memory blocks in the Cyclone IV E device can be used to provide an on-chip memory for the Nios II processor. The SRAM, SDRAM and Flash memory chips on the DE2-115 board are accessed through the appropriate interfaces. Parallel and serial input/output interfaces provide typical I/O ports used in computer systems. A special JTAG UART interface is used to connect to the circuitry that provides a Universal Serial Bus (USB) link to the host computer. This circuitry and the associated software is called the USB-Blaster. Another module, called the JTAG Debug module, is provided to allow the host computer to control the Nios II system. It makes it possible to perform operations such as downloading programs into memory, starting and stopping execution, setting breakpoints, and collecting real-time execution trace data.

3.2 Starting a New Project

To hold the design files for this tutorial, we will use a one of the directories (*depending on board*):

- DE2_115_Lab_3_SoftCore
- Mercurio_IV_Lab_3_SoftCore

To start working on a new design we first have to define a new design project. Create a new project:

5. Select File > New Project Wizard
 - a. Name of this project
 - i. DE2_115_Lab_3_SoftCore
 - ii. Mercurio_IV_Lab_3_SoftCore
 - b. Top-level design entity
 - i. DE2_115_nios_top
 - ii. MercurioIV_nios_top
6. Empty Project
7. Add All
8. Select the family and device
 - a. DE2-115: EP4CE115F29C7
 - b. Mercurio IV: EP4CE30F23C7

3.3 Pin Assignment

For this project, a complete CSV file with the pinout was provided.

Select Assignments > Import Assignemnts> Select the CSV file.

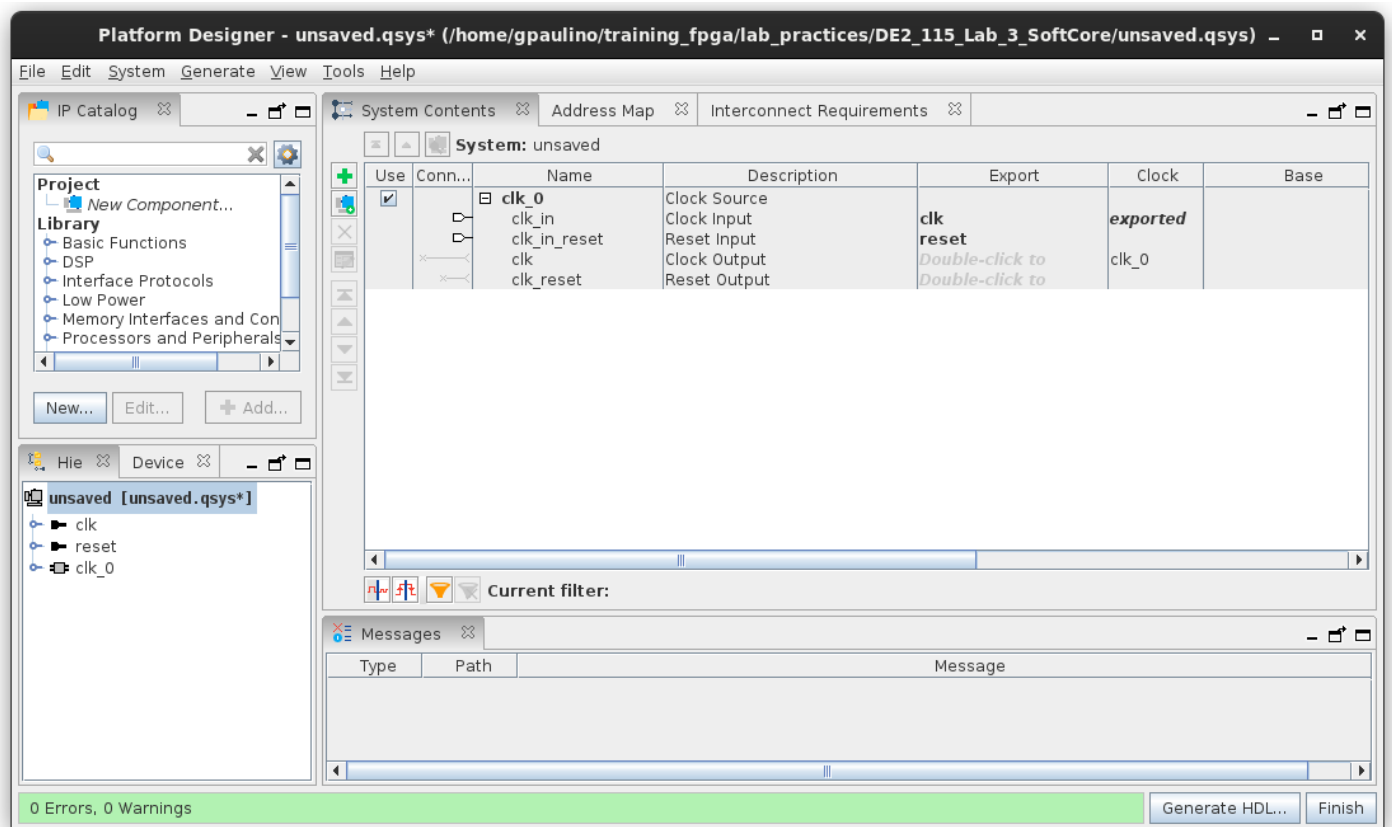
3.4 Platform Designer

To implement the system in Figure 3.1, we have to instantiate the following functional units:

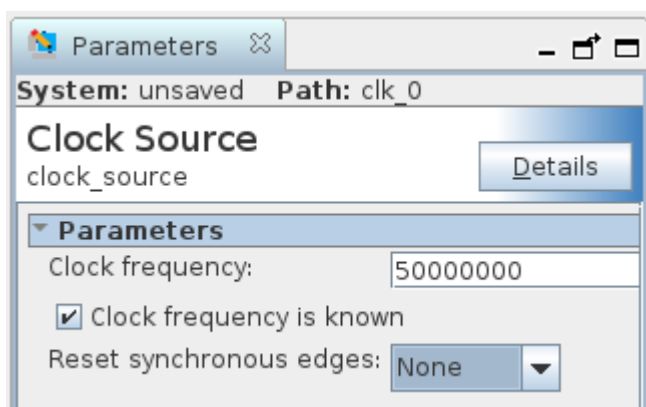
- **Nios II processor**, which is referred to as a Central Processing Unit (CPU)
- **On-chip memory**, which consists of the memory blocks in the Cyclone IV E chip; we will specify a 4-Kbyte memory arranged in 32-bit words
- Two parallel **I/O interfaces**
- **JTAG UART interface** for communication with the host computer

To define the desired system, perform the following steps:

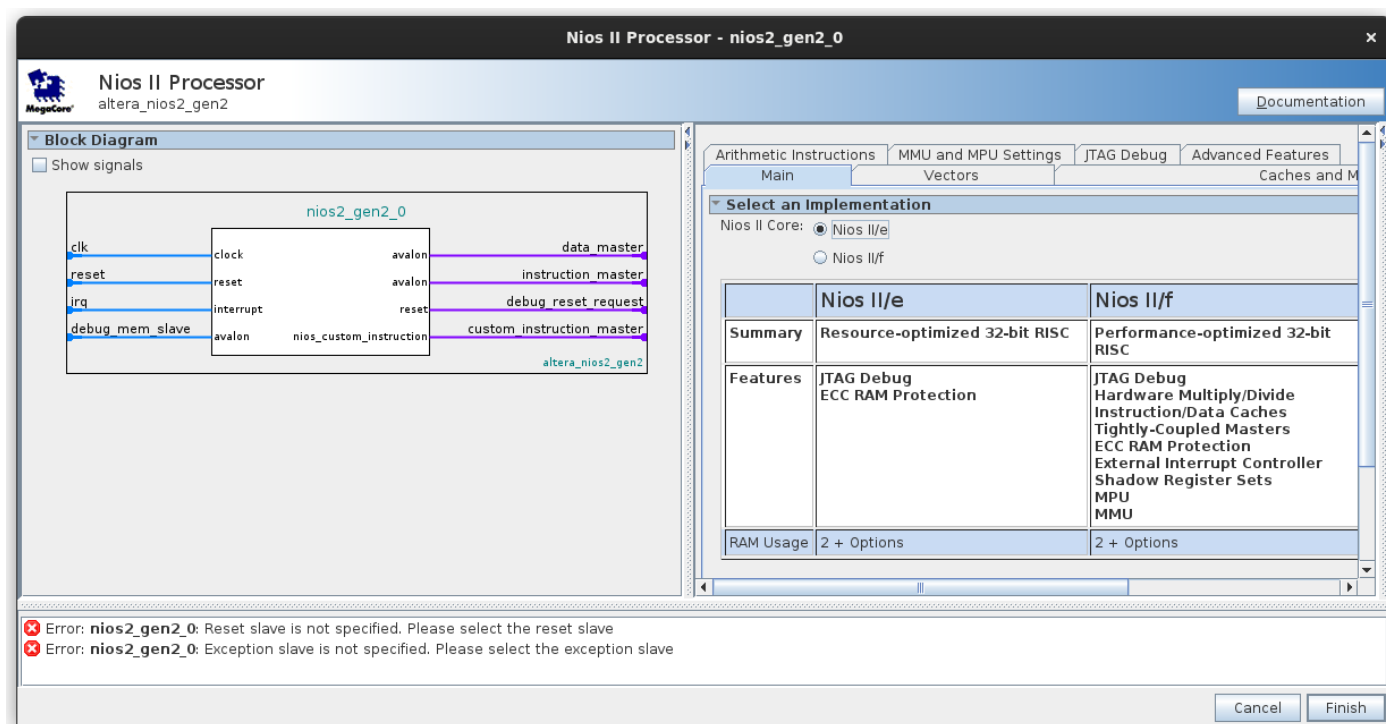
Select Tools > Platform Designer



If not already included in this list, specify a clock named clk_0 with the source designated as External and the frequency set to 50.0 MHz.



On the left side select **Processors and Peripherals** > **Embedded Processors** > **Nios II Processor** and click Add.



Choose **Nios II/e** which is the simplest version of the processor and click Finish.

Note: There may be some warnings or error messages displayed in the Messages window (at the bottom of the screen), because some parameters have not yet been specified. Ignore these messages as we will provide the necessary data later.

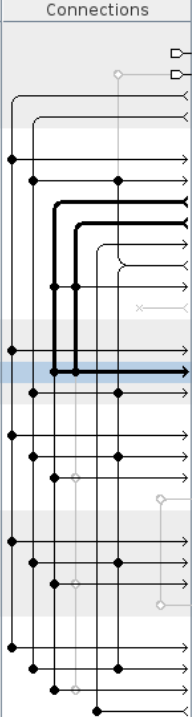
To specify the **on-chip memory** perform the following:

- Select **Basic Functions** > **On-Chip Memory** > **On-Chip Memory (RAM or ROM) Intel FPGA IP** and click Add
- Do not change the default settings
- Click Finish

| Connections | Name | Description | Export |
|-------------|-----------------------------|----------------------------------|------------------------|
| | [-] clk_0 | Clock Source | |
| | clk_in | Clock Input | clk |
| | clk_in_reset | Reset Input | reset |
| | clk | Clock Output | <i>Double-click to</i> |
| | clk_reset | Reset Output | <i>Double-click to</i> |
| | [-] nios2_gen2_0 | Nios II Processor | |
| | clk | Clock Input | <i>Double-click to</i> |
| | reset | Reset Input | <i>Double-click to</i> |
| | data_master | Avalon Memory Mapped Master | <i>Double-click to</i> |
| | instruction_master | Avalon Memory Mapped Master | <i>Double-click to</i> |
| | irq | Interrupt Receiver | <i>Double-click to</i> |
| | debug_reset_req... | Reset Output | <i>Double-click to</i> |
| | debug_mem_slave | Avalon Memory Mapped Slave | <i>Double-click to</i> |
| | custom_instructi... | Custom Instruction Master | <i>Double-click to</i> |
| | [-] onchip_memory... | On-Chip Memory (RAM or ROM... | |
| | clk1 | Clock Input | <i>Double-click to</i> |
| | s1 | Avalon Memory Mapped Slave | <i>Double-click to</i> |
| | reset1 | Reset Input | <i>Double-click to</i> |
| | [-] pio_0 | PIO (Parallel I/O) Intel FPGA IP | |
| | clk | Clock Input | <i>Double-click to</i> |
| | reset | Reset Input | <i>Double-click to</i> |
| | s1 | Avalon Memory Mapped Slave | <i>Double-click to</i> |
| | external_conne... | Conduit | <i>Double-click to</i> |
| | [-] pio_1 | PIO (Parallel I/O) Intel FPGA IP | |
| | clk | Clock Input | <i>Double-click to</i> |
| | reset | Reset Input | <i>Double-click to</i> |
| | s1 | Avalon Memory Mapped Slave | <i>Double-click to</i> |
| | external_conne... | Conduit | <i>Double-click to</i> |
| | [-] jtag_uart_0 | JTAG UART Intel FPGA IP | |
| | clk | Clock Input | <i>Double-click to</i> |
| | reset | Reset Input | <i>Double-click to</i> |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | <i>Double-click to</i> |
| | irq | Interrupt Sender | <i>Double-click to</i> |

Note that the Platform Designer automatically chooses names for the various components. The names are not necessarily descriptive enough to be easily associated with the target design, but they can be changed.

Observe that the base and end addresses of the various components in the designed system have not been properly assigned. These addresses can be assigned by the user, but they can also be assigned automatically by the Platform Designer tool. We will choose the latter possibility. However, we want to make sure that the on-chip memory has the base address of zero. Double-click on the Base address for the on-chip memory in the Platform Designer window and enter the address 0x00000000. Then, lock this address by clicking on the adjacent lock symbol. Now, let the Platform Designer assign the rest of the addresses by selecting **System > Auto-Assign Base Addresses** (at the top of the window).



| Connections | Name | Description | Export | Clock | Base | End | IRQ |
|-------------|--------------------|----------------------------------|-----------------|----------|--------|--------|--------|
| | clk_0 | Clock Source | | | | | |
| | clk_in | Clock Input | clk | exported | | | |
| | clk_in_reset | Reset Input | reset | | | | |
| | clk | Clock Output | Double-click to | clk_0 | | | |
| | clk_reset | Reset Output | Double-click to | | | | |
| | nios2_gen2_0 | Nios II Processor | | | | | |
| | clk | Clock Input | Double-click to | clk_0 | | | |
| | reset | Reset Input | Double-click to | [clk] | | | |
| | data_master | Avalon Memory Mapped Master | Double-click to | [clk] | | | |
| | instruction_master | Avalon Memory Mapped Master | Double-click to | [clk] | | | |
| | irq | Interrupt Receiver | Double-click to | [clk] | | IRQ 0 | IRQ 31 |
| | debug_reset_req... | Reset Output | Double-click to | [clk] | | | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to | [clk] | 0x1800 | 0x1fff | |
| | onchip_memory2_0 | On-Chip Memory (RAM or ROM...) | | | | | |
| | clk1 | Clock Input | Double-click to | clk_0 | | | |
| | s1 | Avalon Memory Mapped Slave | Double-click to | [clk1] | 0x0000 | 0x0fff | |
| | reset1 | Reset Input | Double-click to | [clk1] | | | |
| | switches | PIO (Parallel I/O) Intel FPGA IP | | | | | |
| | clk | Clock Input | Double-click to | clk_0 | | | |
| | reset | Reset Input | Double-click to | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Double-click to | [clk] | 0x2000 | 0x200f | |
| | leds | PIO (Parallel I/O) Intel FPGA IP | | | | | |
| | clk | Clock Input | Double-click to | clk_0 | | | |
| | reset | Reset Input | Double-click to | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Double-click to | [clk] | 0x2010 | 0x201f | |
| | jtag_uart_0 | JTAG UART Intel FPGA IP | | | | | |
| | clk | Clock Input | Double-click to | clk_0 | | | |
| | reset | Reset Input | Double-click to | [clk] | | | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to | [clk] | 0x2020 | 0x2027 | |
| | irq | Interrupt Sender | Double-click to | [clk] | | | |

The behavior of the Nios II processor when it is reset is defined by its reset vector. It is the location in the memory device from which the processor fetches the next instruction when it is reset. Similarly, the exception vector is the memory address of the instruction that the processor executes when an interrupt is raised. To specify these two parameters, perform the following:

- Right-click on the nios2_gen2_0 component, and then select Edit
- In the Vectors tab, select onchip_memory2_0.s1 to be the memory device for both reset and exception vectors
- Do not change the default settings for offsets
- Observe that the error messages dealing with memory assignments shown will now disappear
- Click Finish

| Reset Vector | |
|--------------------------|---------------------|
| Reset vector memory: | onchip_memory2_0.s1 |
| Reset vector offset: | 0x00000000 |
| Reset vector: | 0x00000000 |
| Exception Vector | |
| Exception vector memory: | onchip_memory2_0.s1 |
| Exception vector offset: | 0x00000020 |
| Exception vector: | 0x00000020 |

So far, we have specified all connections inside our nios_system circuit. It is also necessary to specify connections to external components, which are switches and LEDs in our case. To accomplish this, double click on Double-click to export (in the Export column of the System Contents tab) for external_connection of the switches PIO, and type the name switches. Similarly, establish the external connection for the lights, called leds.

The complete system is depicted as follows:

| Use | Connections | Name | Description | Export | Clock | Base | End | IRQ |
|-------------------------------------|--------------------|----------------------------------|--------------|--------|----------|------|-----|-----|
| <input checked="" type="checkbox"/> | clk_0 | clk_in | Clock Source | clk | exported | | | |
| <input checked="" type="checkbox"/> | clk_in_reset | Reset Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | clk_reset | Clock Output | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | reset | Reset Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | data_master | Avalon Memory Mapped Master | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | instruction_master | Avalon Memory Mapped Master | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | irq | Interrupt Receiver | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | debug_reset_req... | Reset Output | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | debug_mem_slave | Avalon Memory Mapped Slave | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | custom_instruct... | Custom Instruction Master | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | onchip_memory... | On-Chip Memory (RAM or ROM) | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | clk1 | Clock Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | s1 | Avalon Memory Mapped Slave | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | reset1 | Reset Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | switches | PIO (Parallel I/O) Intel FPGA IP | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | clk | Clock Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | reset | Reset Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | s1 | Avalon Memory Mapped Slave | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | external_conec... | Conduit | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | leds | PIO (Parallel I/O) Intel FPGA IP | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | clk | Clock Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | reset | Reset Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | s1 | Avalon Memory Mapped Slave | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | external_conec... | Conduit | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | jtag_uart_0 | JTAG UART Intel FPGA IP | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | clk | Clock Input | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | avalon_jtag_slave | Avalon Memory Mapped Slave | | | clk_0 | | | |
| <input checked="" type="checkbox"/> | irq | Interrupt Sender | | | clk_0 | | | |

Messages:

| Type | Path | Message |
|------|----------------------------|--|
| Info | 2 Info Messages | |
| Info | nios_processor.pio_0 | PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation. |
| Info | nios_processor.jtag_uart_0 | JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board |

0 Errors, 0 Warnings

Generate HDL... Finish

Having specified all components needed to implement the desired system, it can now be generated. **Save** the specified system; where you can use the name nios_processor (or nios_system). Then, select **Generate > Generate HDL**. Select None for the option Simulation > Create simulation model, because in

this tutorial we will not deal with the simulation of hardware. Click Generate on the bottom of the window. When successfully completed, the generation process produces the message “Generate Completed”.

Exit the Platform Designer tool to return to the main Quartus window.

3.4.1 Instantiation of the Module Generated by the Platform Designer

The Platform Designer tool generates a Verilog module that defines the desired Nios II system. The following code will be used to instantiate the system module in the top design:

```
nios_processor u0 (
    .clk_clk          (<connected-to-clk_clk>),          //      clk.clk
    .reset_reset_n    (<connected-to-reset_reset_n>),    //      reset.reset_n
    .switches_export  (<connected-to-switches_export>),  // switches.export
    .leds_export       (<connected-to-leds_export>)       //      leds.export
);
```

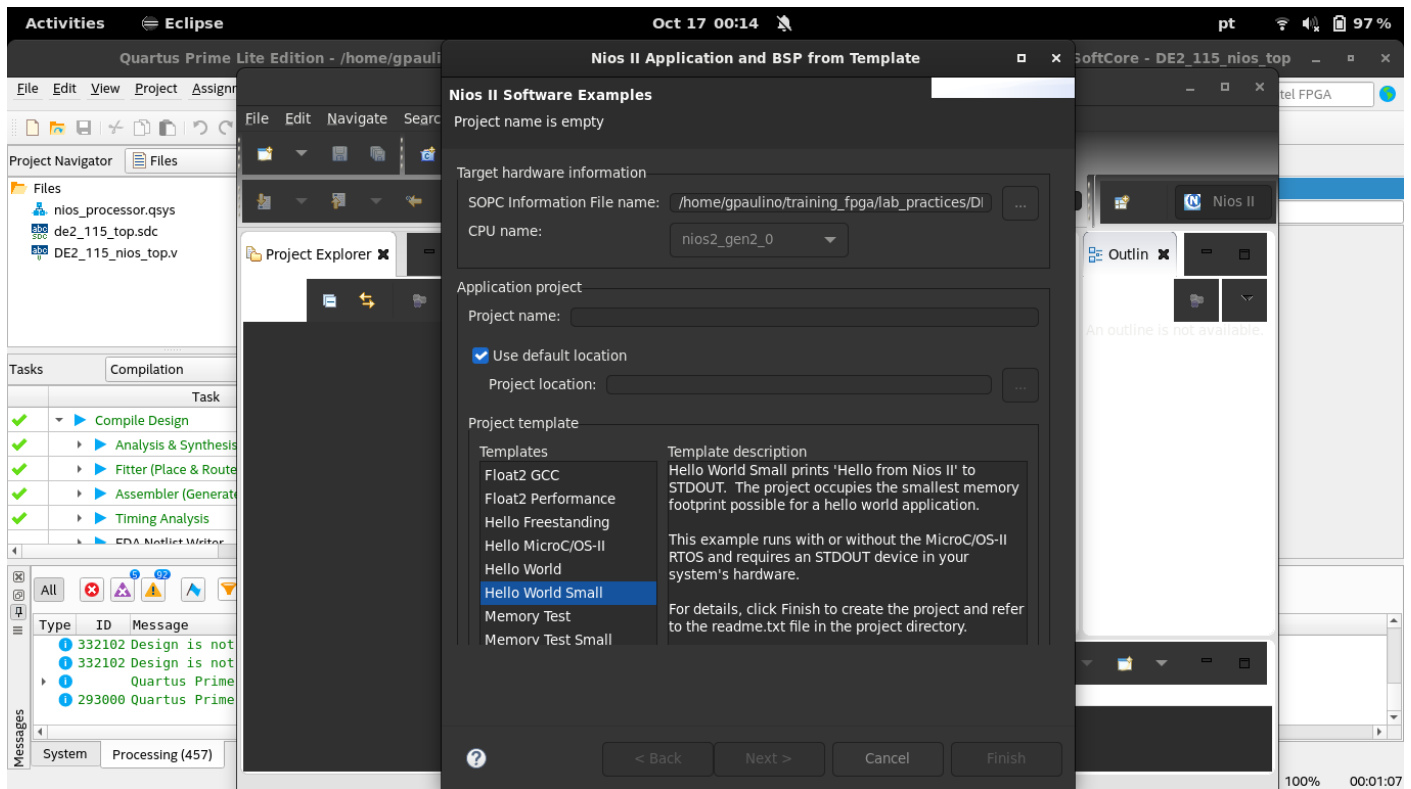
3.5 Nios II Software Build Tools for Eclipse

A parallel I/O interface generated by the Platform Designer tool is accessible by means of registers in the interface. Depending on how the PIO is configured, there may be as many as four registers. One of these registers is called the Data register. In a PIO configured as an input interface, the data read from the Data register is the data currently present on the PIO input lines. In a PIO configured as an output interface, the data written (by the Nios II processor) into the Data register drives the PIO output lines. If a PIO is configured as a bidirectional interface, then the PIO inputs and outputs use the same physical lines. In this case there is a Data Direction register included, which determines the direction of the input/output transfer. In our unidirectional PIOs, it is only necessary to have the Data register. The addresses assigned by the Platform Designer tool are 0x00002010 for the Data register in the PIO called switches and 0x00002000 for the Data register in the PIO called LEDs.

Our application task is very simple. A pattern selected by the current setting of slider switches has to be displayed on the LEDs.

In Quartus, open the **Tools > Nios II Software Build Tools for Eclipse**.

Select **File > Nios II Application and BSP from Template**. And choose the SOPC Information file as the “nios_processor.sopcinfo”. The template used can be “Hello Wold Small”.



Nios II C-language program that implements our task:

```
#include "sys/alt_stdio.h"

#define switches (volatile char *) 0x0002010
#define leds (char *) 0x0002000

int main()
{
    alt_putstr("Hello from Nios II!\n");

    /* Event loop never exits. */
    while (1)
        *leds = *switches;

    return 0;
}
```

References

- [1] Altera Corporation. **Quartus II Introduction Using Verilog Design**. 2010
- [2] Altera Corporation. **Introduction to the Altera SOPC Builder Using Verilog Design**. 2010