VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY – UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING
DEPARTMENT OF ELECTRONICS
--oOo—

LABORATORY MANUAL

EXPERIMENT ON ALTERA DE10 STANDARD KIT

ABOUT THE MANUAL

This document is intended to serve as a lab manual for students enrolled in Digital Design Lab at HCMC University of Technology. All the Lab Experiments are designed for using VHDL to implement a system on Altera DE10 Standard Board.

There are one introduction lab and five main labs in this courses:

Digital Design Lab	
Lab 0	Review of digital design using FPGAs
Lab 1	Adders, Subtractors And Multipliers
Lab 2	Finite State Machines
Lab 3	A Simple Processor
Project	An Enhanced Processor
Lab 4	A Processor Intergrated Parallel Input/Output Interface
Lab 5	A Processor Intergrated UART
Project	Final Project

In order to complete the lab on time, all students are required to do prelabs before each class.

Lab Manual:

EXPERIMENT ON ALTERA DE10 STANDARD KIT

I. DE10 STANDARD FPGA BOARD

1. Peripherals:

The Altera DE10 Standard board is equipped with variety peripherals so that users can implement various applications in digital logic and computer organization… Cyclone V FPGA chip, connecting with SDRAM, VGA digital to output converter…, allow users to configure variety of applications.

In this lab, all experiments are simple, we use switches and push buttons for supplying inputs while LEDs and 7-Segment LED display the outputs.

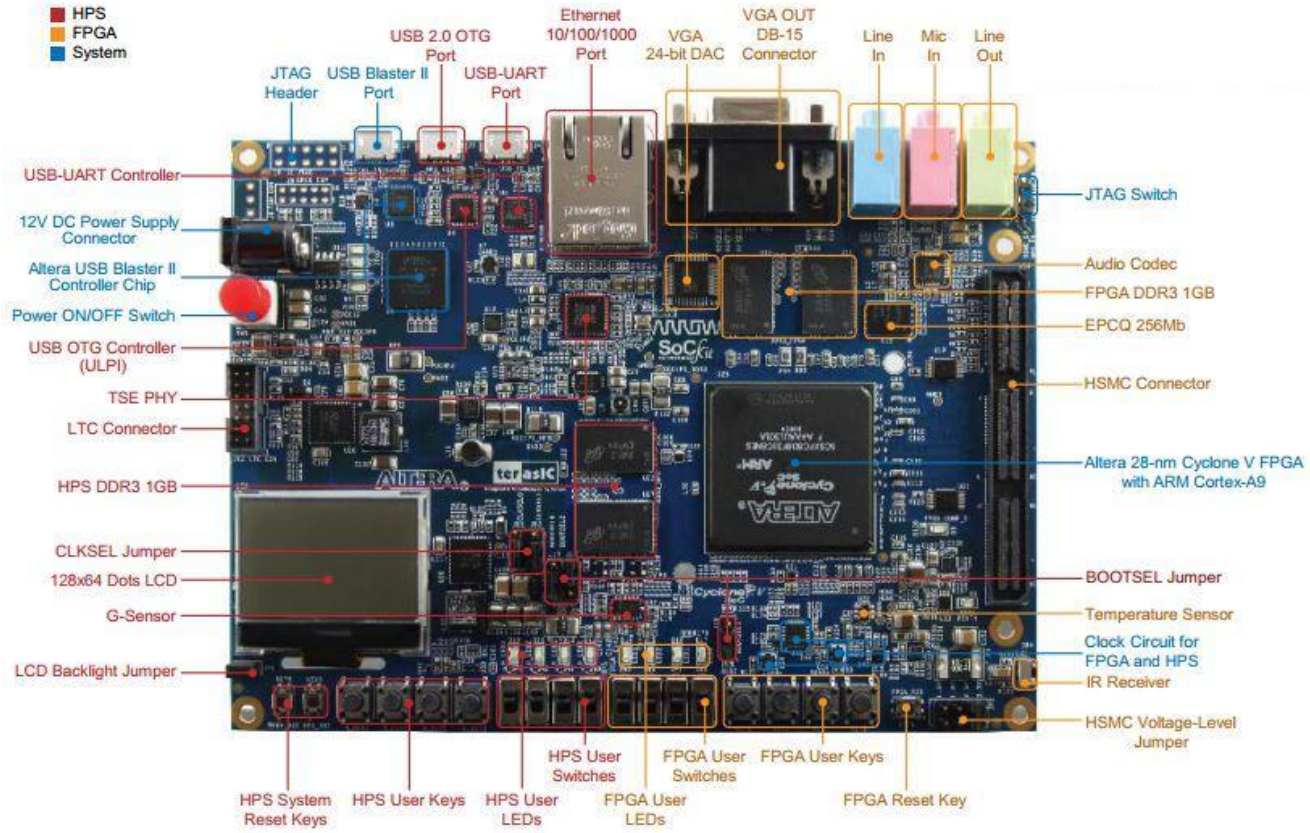


Figure 1: DE10 Standard Board

2. Components:

The DE10-Standard board has many features that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects. Figure 2 is the block

diagram of the board. All the connections are established through the Cyclone V SoC FPGA device to provide maximum flexibility for users. Users can configure the FPGA to implement any system design.

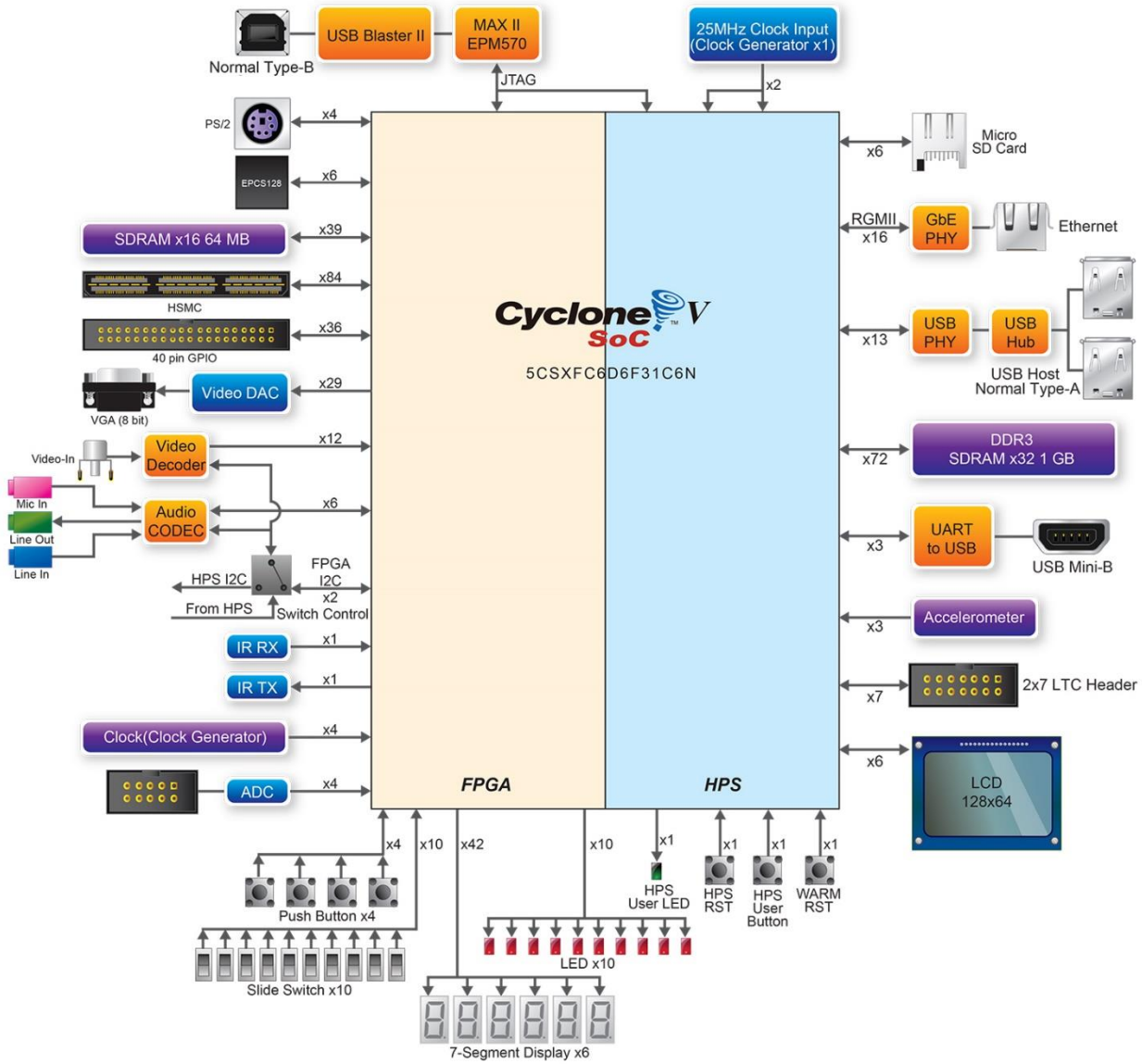


Figure 2: Block diagram of DE10-Standard

Detailed information about Figure 2 are listed below:

- FPGA Device

- Cyclone V SoC 5CSXFC6D6F31C6N Device
- Dual-core ARM Cortex-A9 (HPS)
- 110K programmable logic elements
- 5,761 Kbits embedded memory
- 6 fractional PLLs
- 2 hard memory controllers

➢ Configuration and Debug

- Quad serial configuration device – EPCS128 on FPGA
- Onboard USB-Blaster II (normal type B USB connector)

➢ Memory Device

- 64MB (32Mx16) SDRAM on FPGA
- 1GB (2x256Mx16) DDR3 SDRAM on HPS
- Micro SD card socket on HPS

➢ Communication

- Two port USB 2.0 Host (ULPI interface with USB type A connector)
- UART to USB (USB Mini-B connector)
- 10/100/1000 Ethernet
- PS/2 mouse/keyboard
- IR emitter/receiver
- I2C multiplexer

➢ Connectors

- One HSMC (Configurable I/O standards 1.5/1.8/2.5/3.3V)
- One 40-pin expansion headers
- One 10-pin ADC input header
- One LTC connector (one Serial Peripheral Interface (SPI) Master, one I2C and one GPIO interface)

➢ Display

- 24-bit VGA DAC
- 128x64 dots LCD Module with Backlight

➢ Audio
- 24-bit CODEC, Line-in, Line-out, and microphone-in jacks

➢ Video Input
- TV decoder (NTSC/PAL/SECAM) and TV-in connector

➢ ADC
- Interface: SPI
- Fast throughput rate: 500 KSPS
- Channel number: 8
- Resolution: 12-bit
- Analog input range : 0 ~ 4.096

➢ Switches, Buttons, and Indicators
- 5 user Keys (FPGA x4, HPS x1)
- 10 user switches (FPGA x10)
- 11 user LEDs (FPGA x10, HPS x 1)
- 2 HPS reset buttons (HPS_RESET_n and HPS_WARM_RST_n)
- Six 7-segment displays

➢ Sensors
- G-Sensor on HPS

➢ Power
- 12V DC input

3. Examples of advanced demonstrations on DE10 board:

➢ TV box application

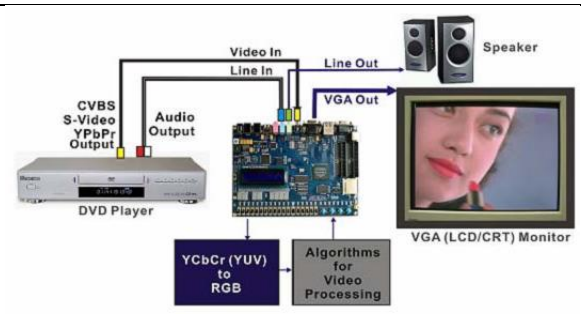


Figure 3: TV box

- USB Paintbrush

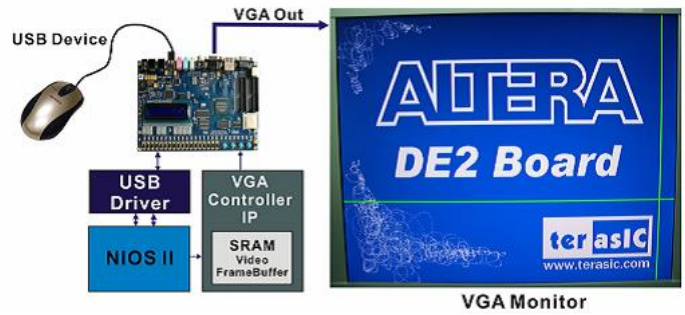


Figure 2: USB Paintbrush

- A Karaoke Machine and SD Card Music Player

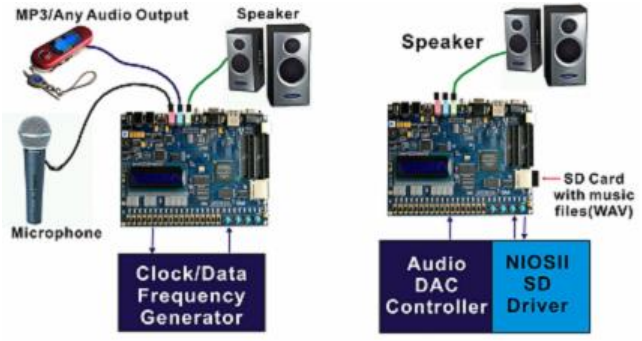


Figure 3: Karaoke Machine and SD Card Music Player

II. QUARTUS 18.1 INTRODUCTION: INSTALLATION AND USAGE

Quartus 18.1 – developed by Intel – is one of the softwares going with the Kit, which provide interface between Kit and users. Users may download Quartus 18.1 at Intel website (FPGA Software Download Center (intel.com)) and get free license after registration. Fig.5 demonstrates software's interface and its basic windows. For more information, user may access "Help" tool on the Toobars.

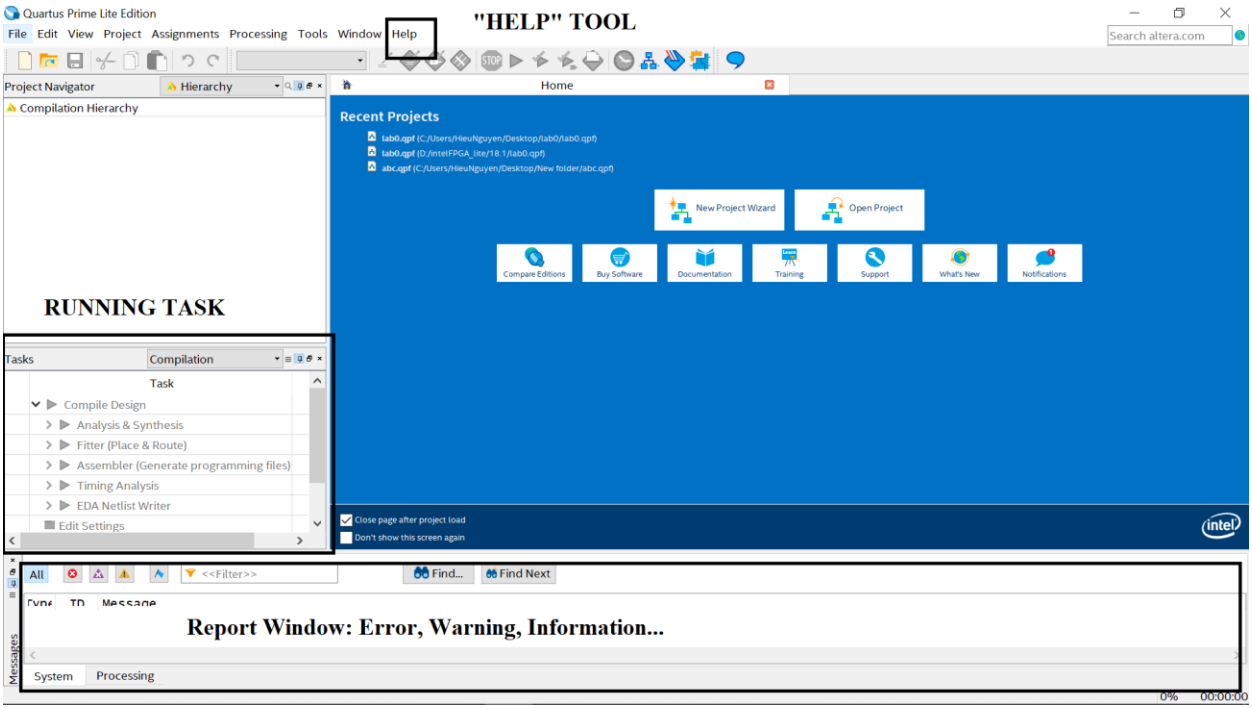


Figure 5: Quartus 18.1 GUI (Graphic User Interface)

1. Quartus 18.1 Installation

Download Quartus 18.1 from Intel website and run Quartus.exe to install the software.

2. Creating design using Quartus 18.1

The following steps describe the basic flow to create a design using Quartus 18.1 GUI:

- Step 1: Creating a new project, compose VHDL code.
- Step 2: Simulating VHDL code to verify the design function (Verification Tools of Quartus/ QuestaSim).

- Step 3: Synthesizing VHDL code using Altera – Quartus tools.
- Step 4: Using Programmer (Quartus' tool) to implement the design on FPGA.
- Step 5: Applying inputs to and observing outputs from our circuit using the peripherals (like switches, buttons, LEDs, etc) on the DE10 board.

III. SAMPLE LAB: LED control using Switches on FPGA DE10

1. ***Create the project:*** Run Quartus 18.1 and follow these steps

Step 1: Create the project

- Choose **File/New Project Wizard** on toolbars to reach the window in Fig.7. You can skip this window in subsequent projects by checking the box ***Don't show me this introduction again***. Press ***Next*** to get the window shown in Fig.8.

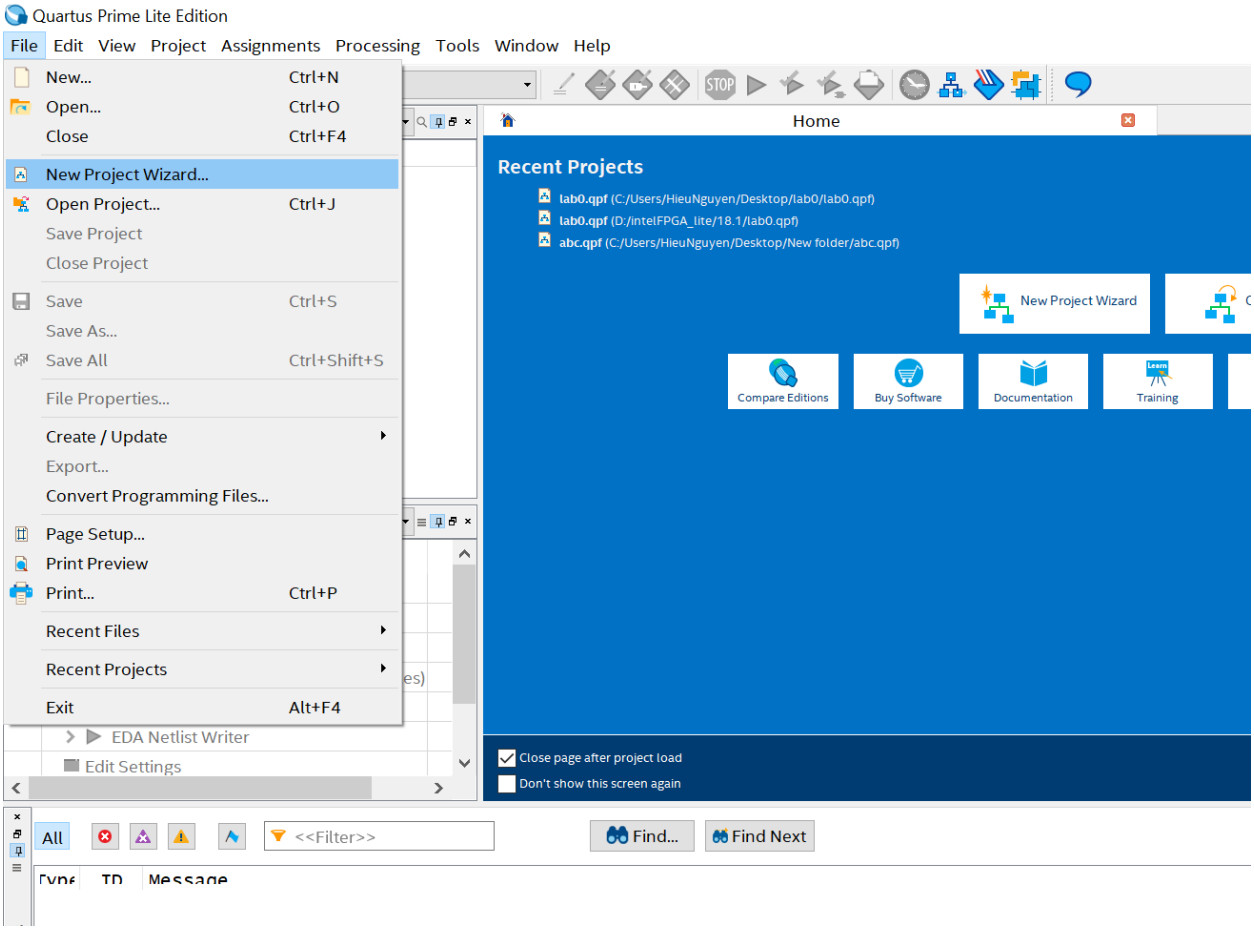


Figure 4: Create a new project

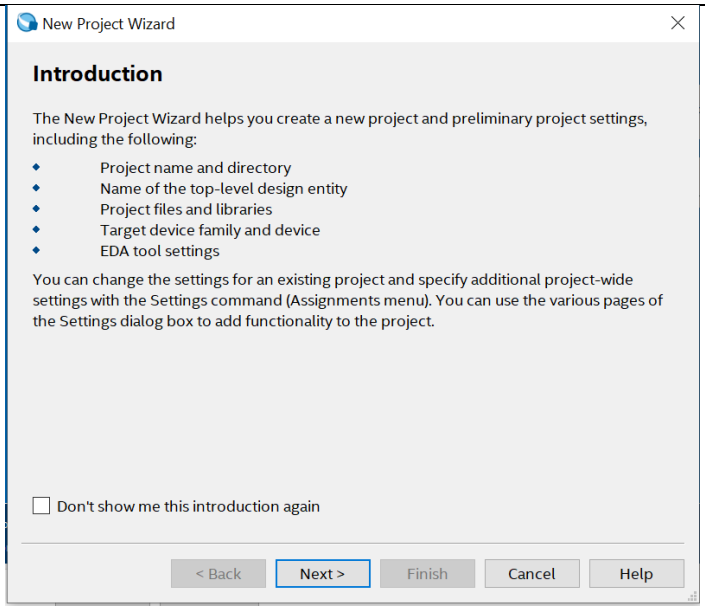


Figure 5: Introduction Interface

- **Step 2:** Choose the working directory and project's name.

  In Fig. 8, project's name is set "test01" and the working directory is "D:\intelFPGA_lite\Project". (Student may change the name and directory).

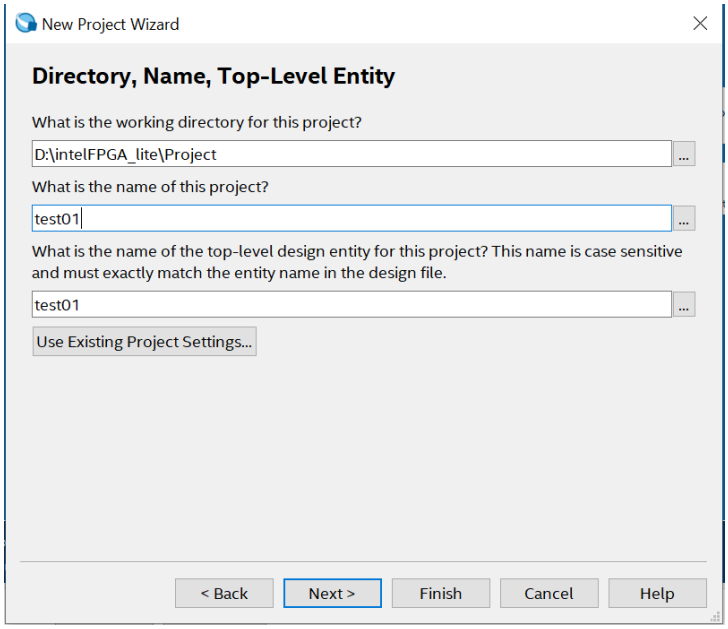


Figure 6: Working directory and project's name settings.

- **Step 3:** Create Project Type and Add files to the project

- In Project Type window, choose ***Empty Project*** and click ***Next***.

- The wizard makes it easy to specify which existing files (if any) should be included in the project. Assuming that we do not have any existing files, click ***Next***, which leads to the window in Fig.9

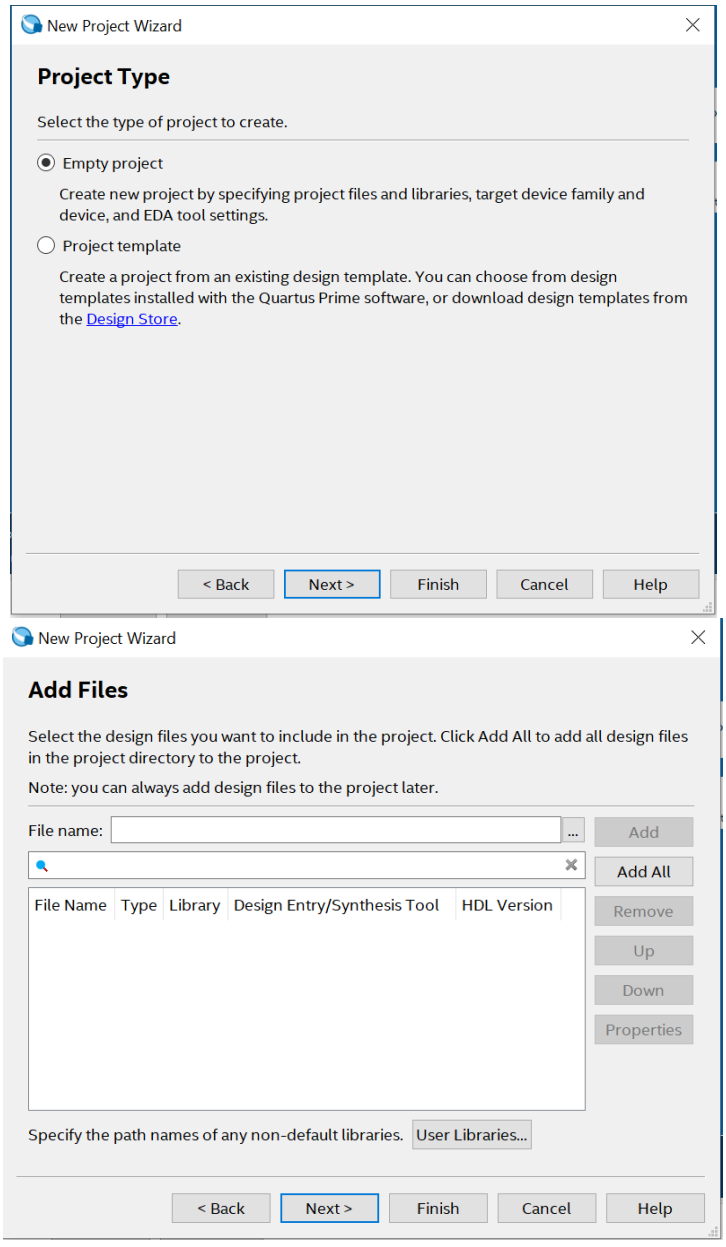


Figure 7: Add file

- **Step 4:** Choose the device

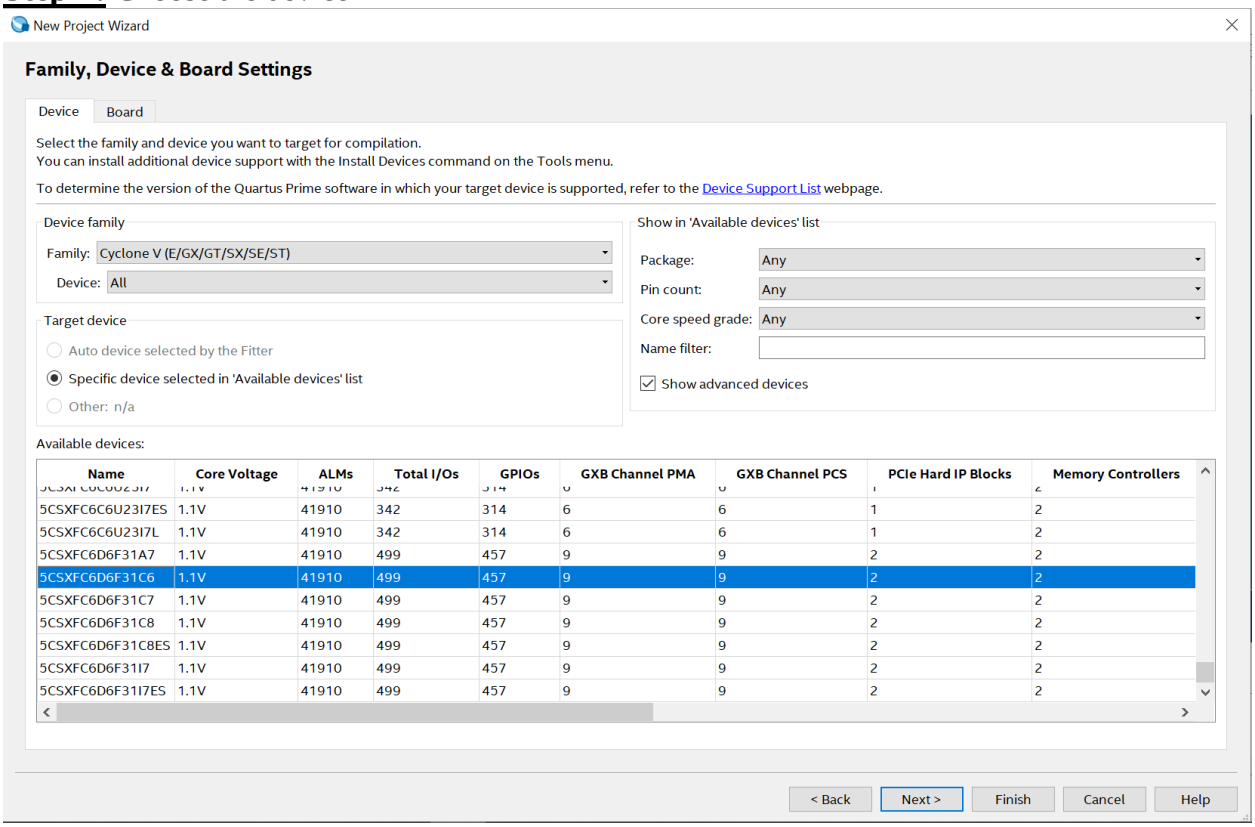


Figure 8: Choose Family & Device Settings.

In this lab, we choose:

Device family: **Cyclone V**

Available device: **5CSXFC6D6F31C6**

Notes: Quartus 18.1 is intended for communicate variety of FPGA devices, remember to select the correct device information.

After completing device settings, click ***Next*** to following steps. Users can specify any third-party tools (EDA tools), however in this lab we rely solely on Quartus 18.1 tools and skip this step.

A summary of the chosen settings appears in the screen shown in Fig.11. Press ***Finish***, which return to the main Quartus 18.1 window, but with *test01* specified as the new project, in the display title bar, as indicated in Fig.12.

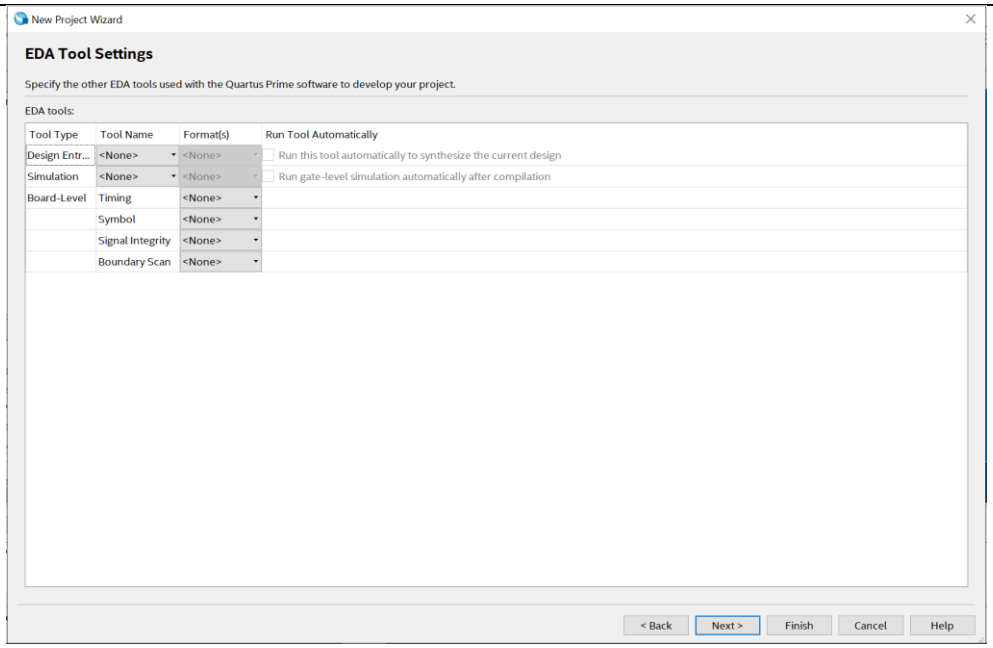


Figure 9 EDA tool settings

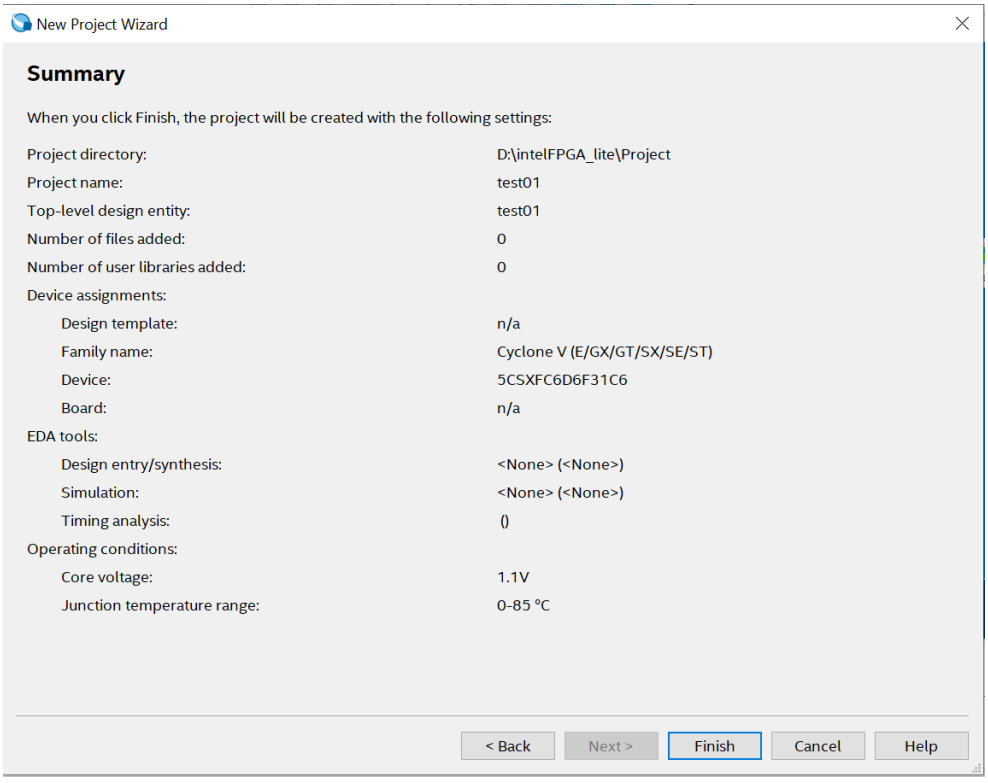


Figure 10: Project summary

- **Step 5:** Create HDL source code file

Choose ***File → New*** as in Fig.13, select **VHDL file** (Design file list) in the dialog (Fig.14). Press ***OK*** to complete this step.

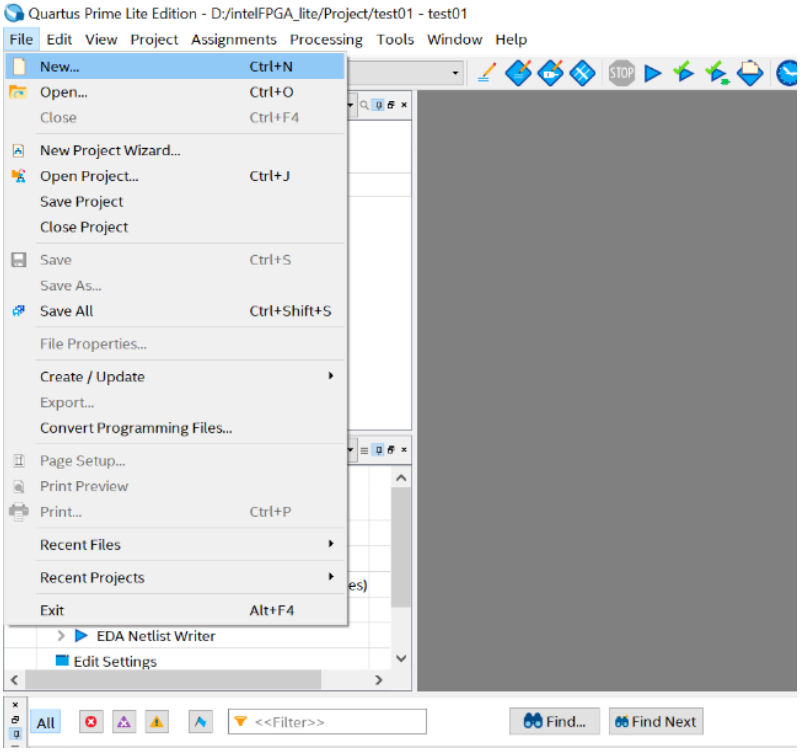


Figure 11: Create HDL source code file

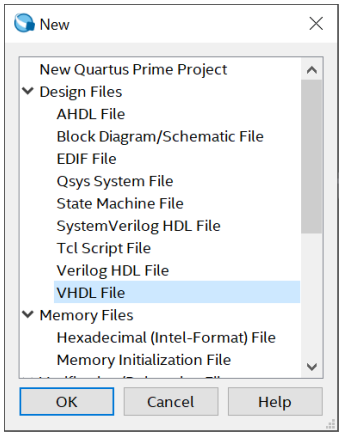


Figure 12: Choose to prepare a VHDL file

- **Step 6:** Compose HDL (VHDL/Verilog HDL) design code.

In the Quartus 18.1 Text Editor, enter the HDL source code into it. The following VHDL source code describes the circuit in which SW0 and SW1 on DE2 kit control LEDR0 and LEDR1 via FPGA.

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

entity test01 is
port (
   SW: in std_logic_vector (1 downto 0) ;
   LEDR : out std_logic_vector (1 downto 0)
);
end entity;

architecture behavior of test01 is
begin
   LEDR (1 downto 0) <= SW (1 downto 0) ;
end architecture;
```

Verilog code:

```
module test01 (SW, LEDR);
   input [1:0] SW;
   output [1:0] LEDR;

   assign LEDR = SW;
endmodule
```

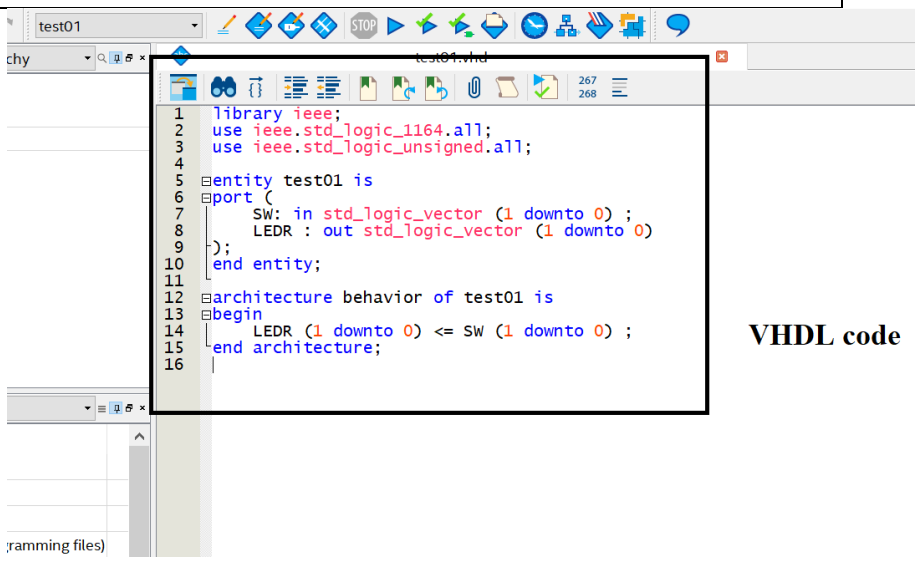


Figure 13: VHDL Text Editor

Save the file by typing ***File → Save As***.

Notes: Ensure that:

- The source code name is similar to the project name.
- Design file stays in project directory (Fig.16)

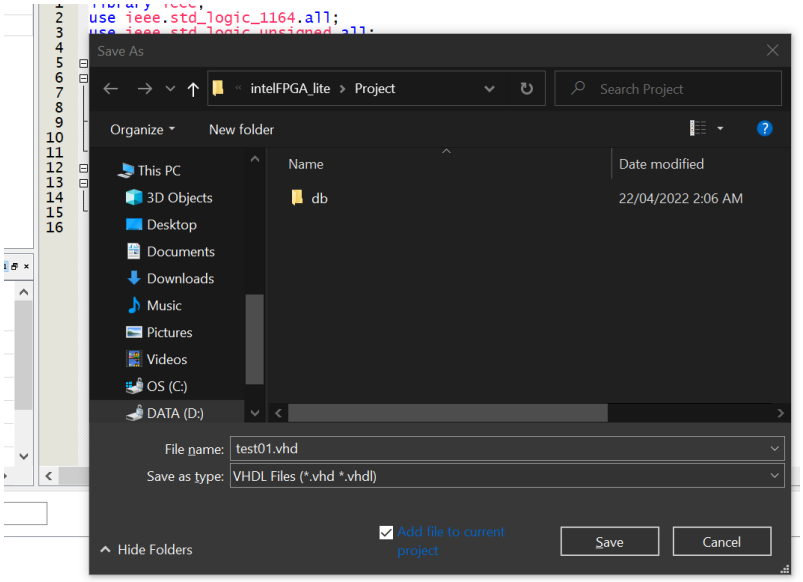


Figure 14: Save VHDL source code file

- **Step 7:** Compile the design

Press Start Compilation button on toolbars (Fig. 17) or choose ***Processing → Start compilation***.

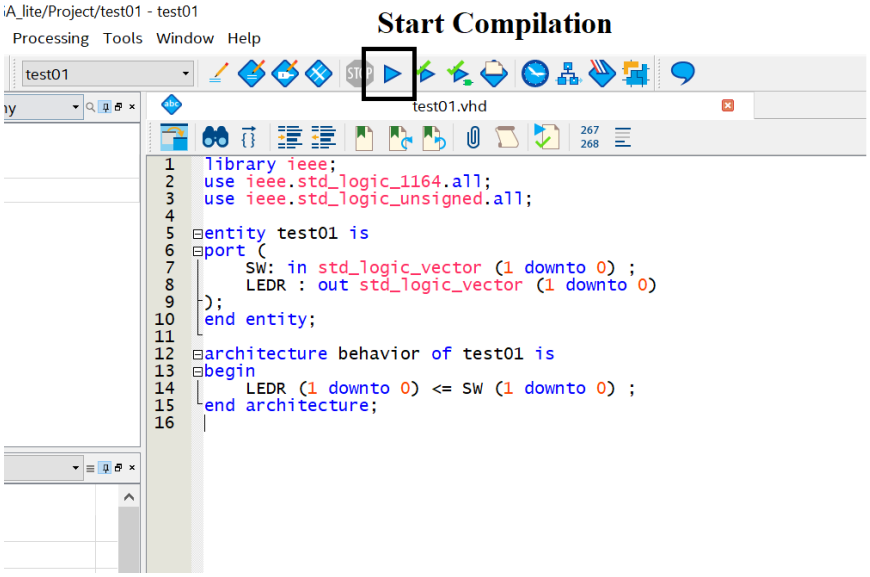


Figure 17: Start Compilation button

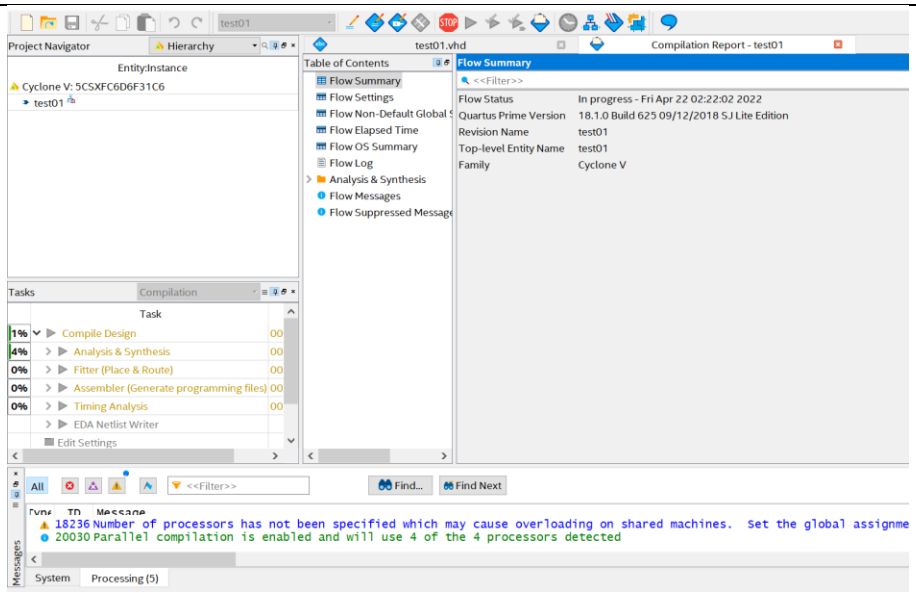


Figure 18: Compilation process

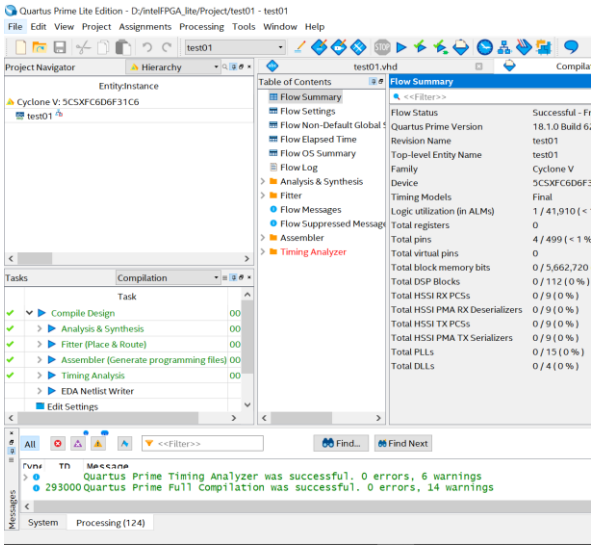


Figure 19: Successful Compilation

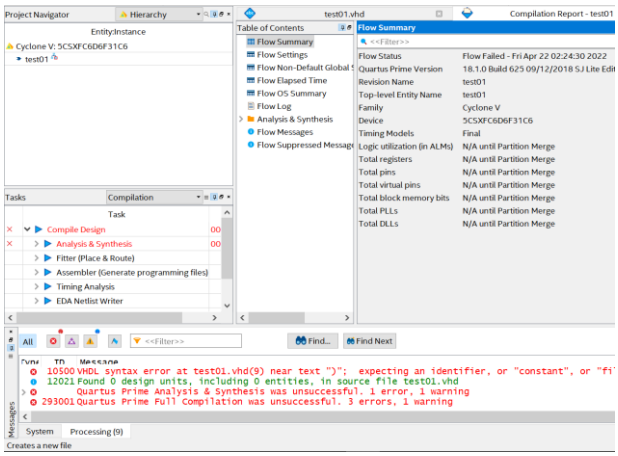


Figure 20: Unsuccessful compilation

When the compilation ends, a pop-up box appears as in Fig.19. If the design is correct, one of the messages will state that the compilation is successful and that there are no errors, only Warning and Info messages can be seen.

If the Compiler does not report zero errors, then there is at least one mistake. In this case, a message corresponding to each error found will be displayed in the Messages window. Double-clicking on an error message, devine the problem and fix it.

In this lab as well as some simple design, users may skip Warning messages; however, Warning need to be considered carefully and some Errors may be discovered.

In Fig. 21, the compilation report shows that the design use 4 pins only (SW0, SW1, LEDR0, LEDR1).

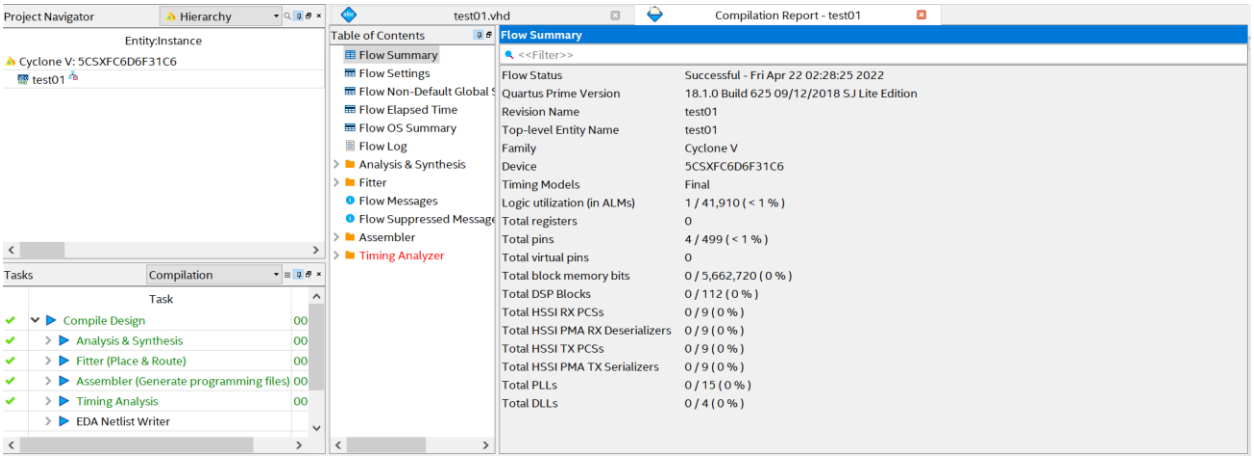


Figure 21: Compilation report

Steps in Compilation process:

- **Analysis & Synthesis:** Syntax Error analysis and converts the design into parts which are available within the selected FPGA. Parts which are available are typically flip flops, memory blocks, look-up tables and adders; and sometimes multipliers and other complex support parts.

- **Filters:** This places the parts within an FPGA, connects them together and to the input and output pins, and optimizes the layout for the user goals (typically speed). A design may require hundreds, or thousands of CLBs (Combinational Logic Elements), LEs (Logic Elements) or LUTs (Look-Up Tables).

- **Assembler:** Converts the fitted design into a file which can be used to program the FPGA. The file has one of following format: Programmer Object Files (.pof), SRAM Object Files (.sof), Hexadecimal (Intel-Format) Output Files (.hexout),Tabular Text Files (.ttf), and Raw Binary Files (.rbf).

- **TimeQuest Timing Analysis:** analyze the performance of all logic in your design and help to guide the Fitter to meet timing requirements. You can use the information generated by the timing analyzer to analyze, debug, and validate the timing performance of your design.

After compilation, you can get the Gate-level design by using ***Tools → Netlist Viewer → RTL viewer*** (Fig. 22).

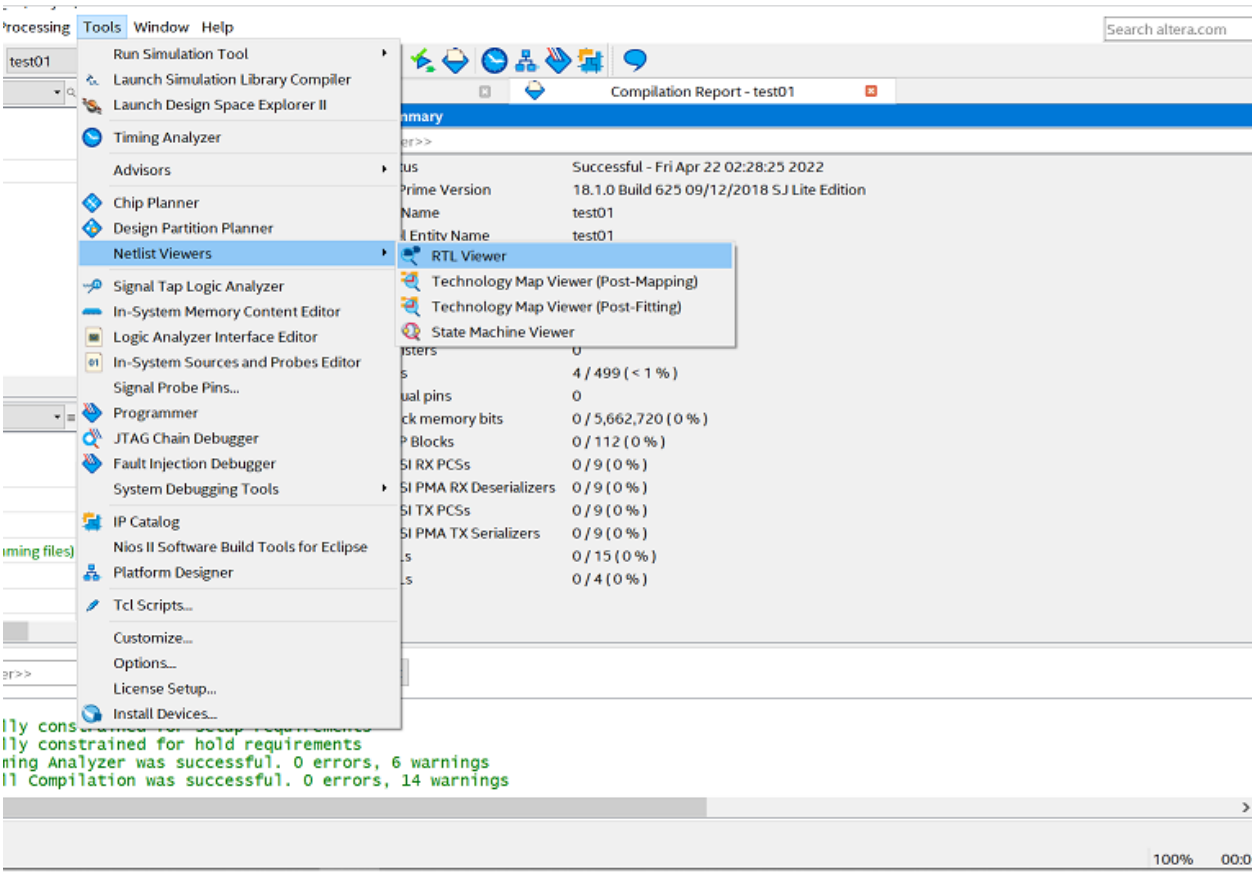


Figure 152: RTL viewer tools

- **<u>Step 8:</u>** Assign pins

Pin assignment is the process which maps the input and output signals of your design to physical pins of the FPGA chip available on the hardware board. Since each peripheral is connected physically to a certain pin in the FPGA on the board, we should make sure the tool exports our inputs and outputs to the peripherals we target for use.

In this project, we use two toggle switches, labeled SW0 and SW1 as the inputs; these switches

are connected to the FPGA pins AB30 and Y27, respectively. The output is LEDR0 and LEDR1 which is connected to the pin AA24 and AB23. This name PIN can be found in file "***DE10-Standard_User_manual.pdf*** " (Fig. 23).

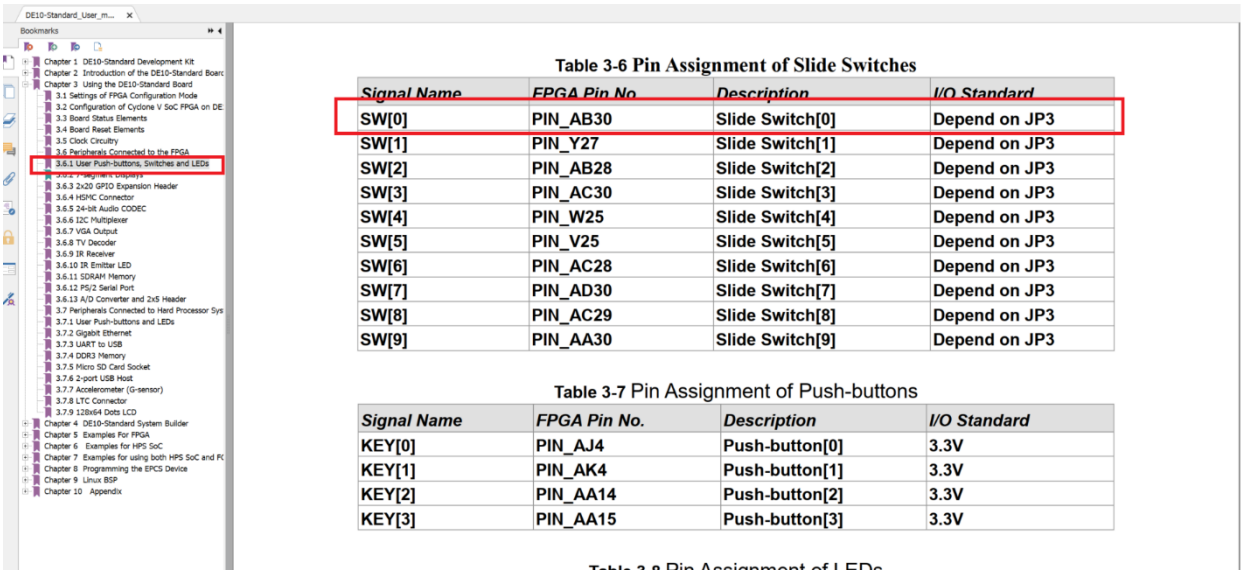


Table 3-6 Pin Assignment of Slide Switches

Signal Name	FPGA Pin No.	Description	I/O Standard
SW[0]	PIN_AB30	Slide Switch[0]	Depend on JP3
SW[1]	PIN_Y27	Slide Switch[1]	Depend on JP3
SW[2]	PIN_AB28	Slide Switch[2]	Depend on JP3
SW[3]	PIN_AC30	Slide Switch[3]	Depend on JP3
SW[4]	PIN_W25	Slide Switch[4]	Depend on JP3
SW[5]	PIN_V25	Slide Switch[5]	Depend on JP3
SW[6]	PIN_AC28	Slide Switch[6]	Depend on JP3
SW[7]	PIN_AD30	Slide Switch[7]	Depend on JP3
SW[8]	PIN_AC29	Slide Switch[8]	Depend on JP3
SW[9]	PIN_AA30	Slide Switch[9]	Depend on JP3

Table 3-7 Pin Assignment of Push-buttons

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY[0]	PIN_AJ4	Push-button[0]	3.3V
KEY[1]	PIN_AK4	Push-button[1]	3.3V
KEY[2]	PIN_AA14	Push-button[2]	3.3V
KEY[3]	PIN_AA15	Push-button[3]	3.3V

Figure 23: Parameter PIN name

Click ***Assignments → Pin Planner*** (Fig 24) to open Pin Planner window.

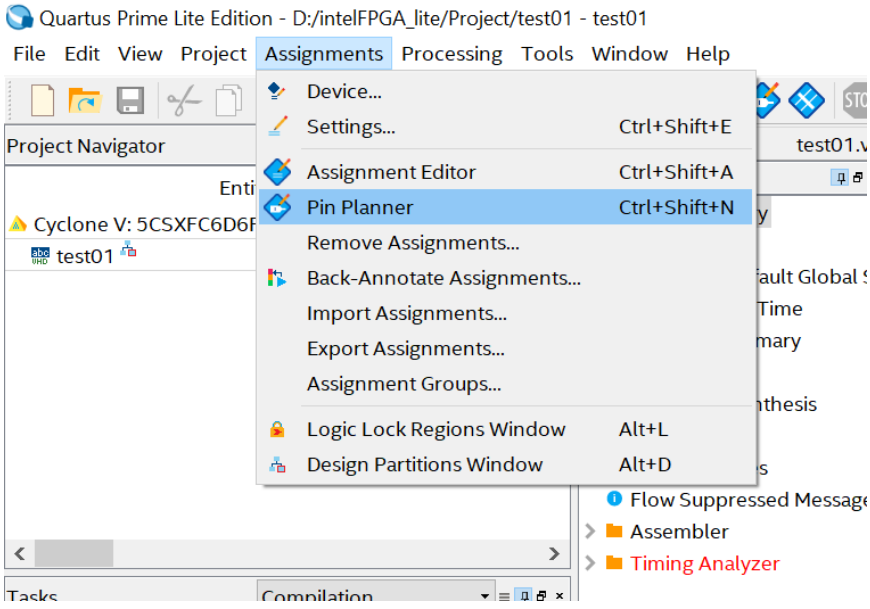


Figure 24: Open Pin Planner.

In Pin Planner Window, at colum Location, click and input Pin name (Fig. 25). After input, the result in shown in Fig. 26.

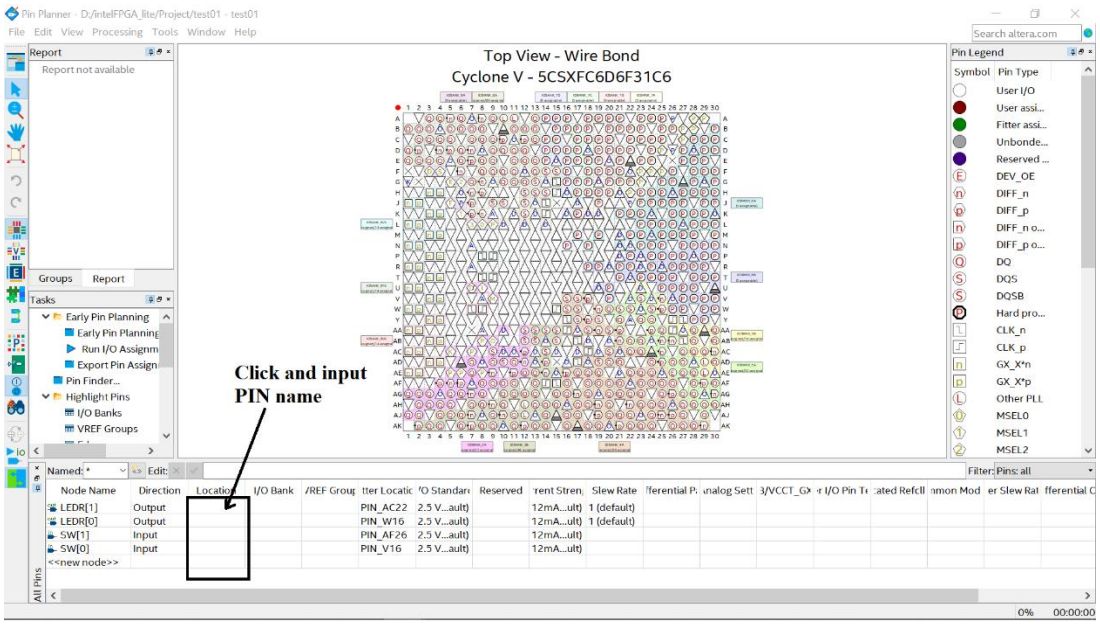


Figure 25: Pin Planner Window.

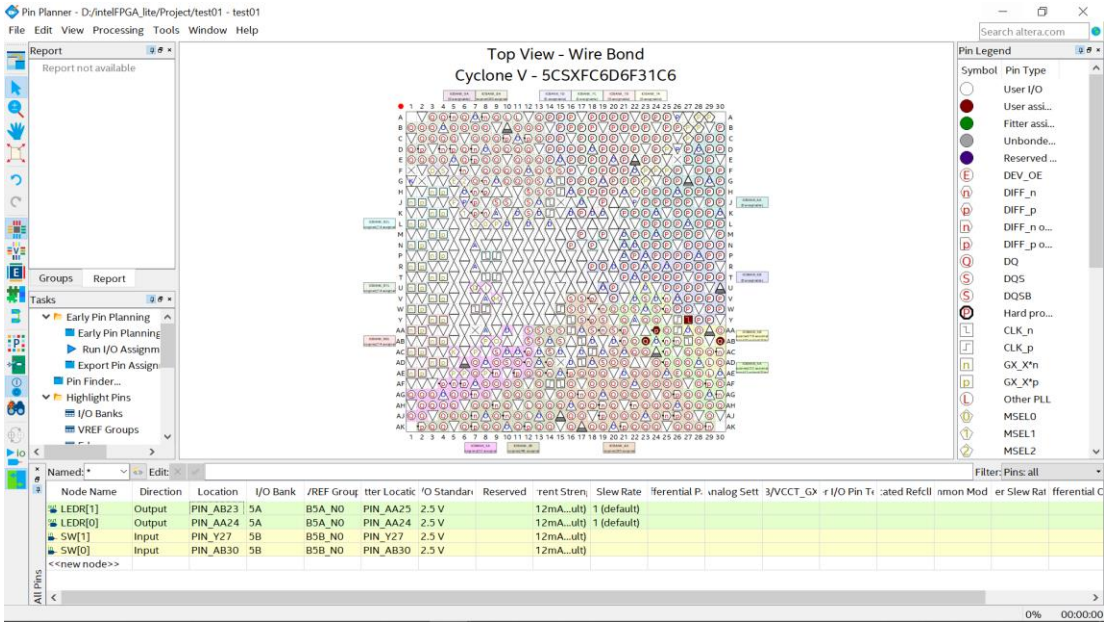


Figure 26: After Input Pin Name.

After this step, recompile the project.

2. Create the project:

Before implementing the designed circuit in the FPGA chip on the DE10 board, it is prudent to simulate it to ascertain its correctness. Quartus 18.1 software includes a simulation tool that can be used to simulate the behavior of a designed circuit.

Before the circuit can be simulated, it is necessary to create the desired waveforms, called *test vectors*, to represent the input signals. It is also necessary to specify which outputs, as well as possible internal points in the circuit, the designer wishes to observe. The simulator applies the test vectors to a model of the implemented circuit and determines the expected response. We will use the Quartus 18.1 Waveform Editor to draw the test vectors, as follows:

- **Step 1:** Create the Waveform file

Click ***File → New***, choose University Program VWF file in "Verification/Debugging files" tab.

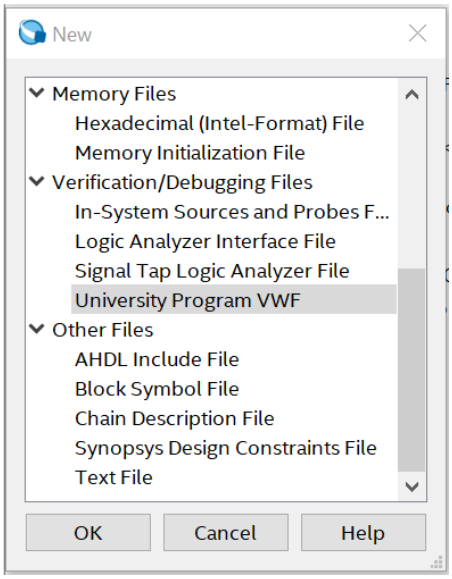


Figure 27: Create waveform file

- **Step 2:** Insert input signal into waveform file

Select ***Edit → Insert → Insert Node or Bus***…, and then click on the ***Node Finder***… button. In the filter tab scroll up and select Pins: assigned, then click the ***List*** button to show all input pins. Click on the ">" arrow to add input pins SW(0), SW(1) and output pins LEDR[0], LEDR[1] to the Selected Nodes column, then click OK.

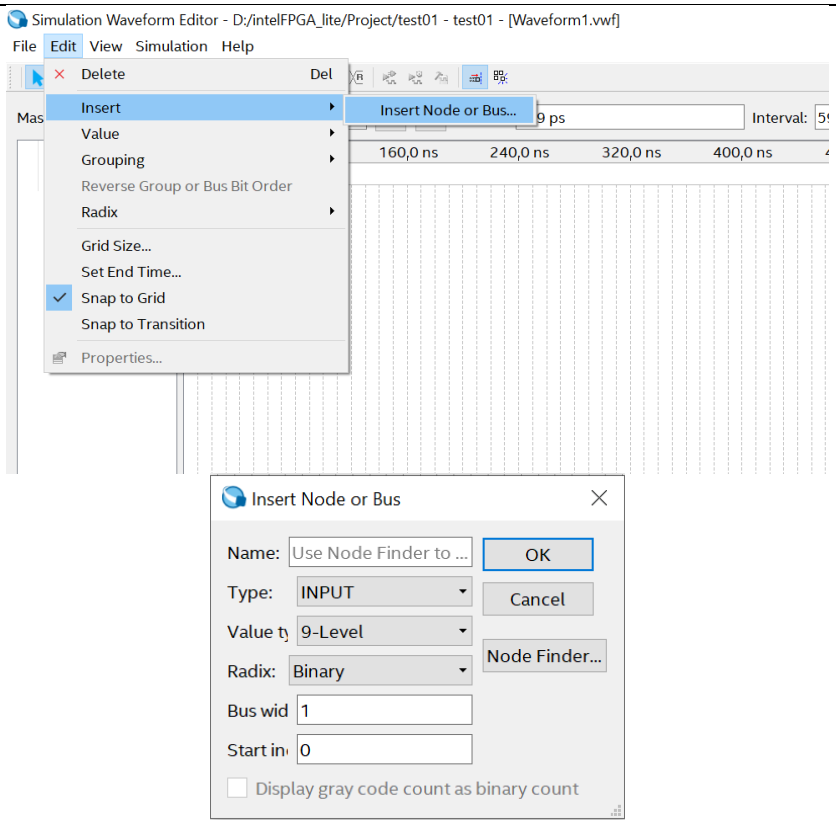


Figure 28: Insert Node or Bus

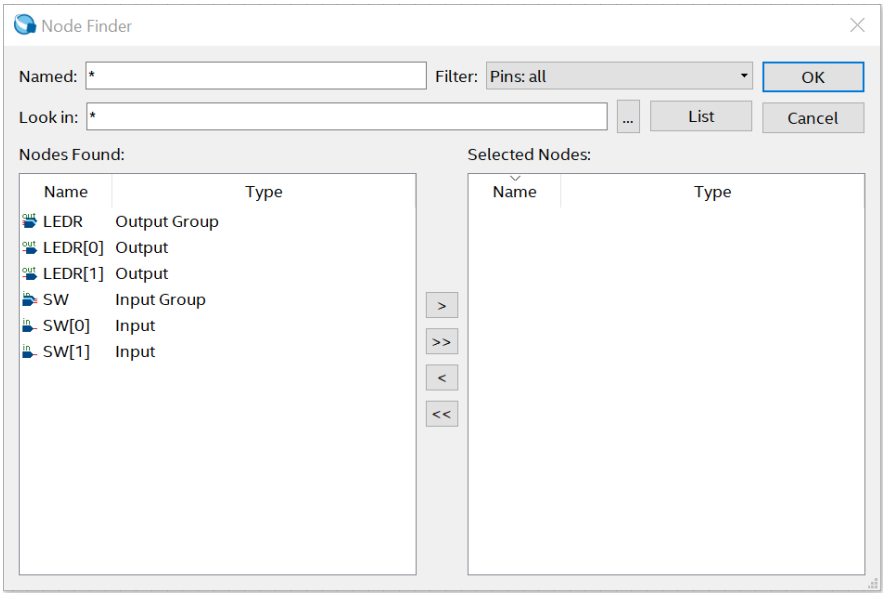


Figure 16 Node Finder dialog

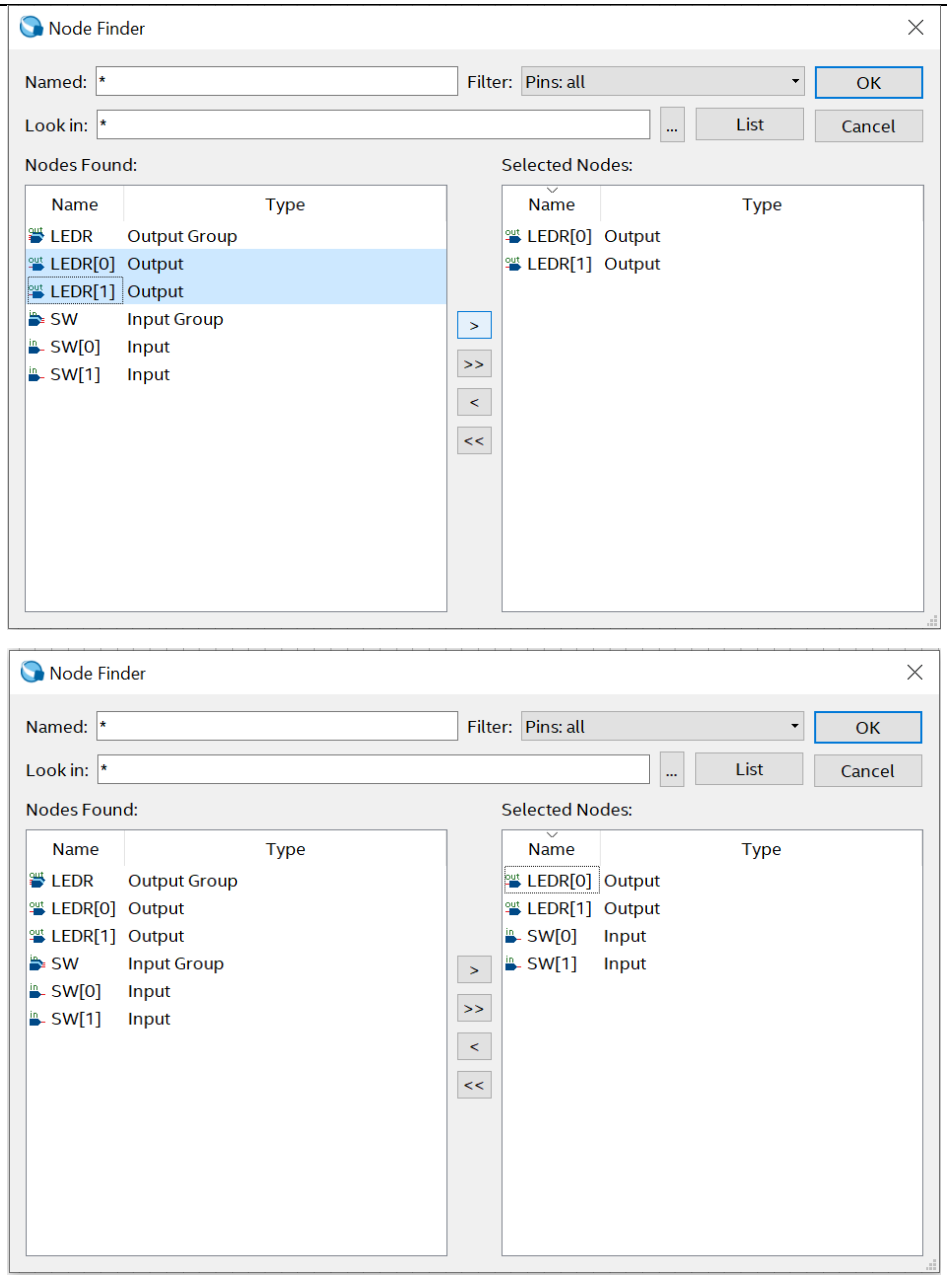


Figure 17: Add pin to Selected Nodes comlumn by pressing > button

- **<u>Step 3:</u>** Set input waveforms

The waveforms can be drawn using the Selection Tool, which is activated by selecting the icon in the toolbar, or the Waveform Editing Tool, which is activated by the icon .

To simulate the behavior of a large circuit, it is necessary to apply a sufficient number of input

valuations and observe the expected values of the outputs. In a large circuit the number of possible input valuations may be huge, so in practice we choose a relatively small (but representative) sample of these input valuations.

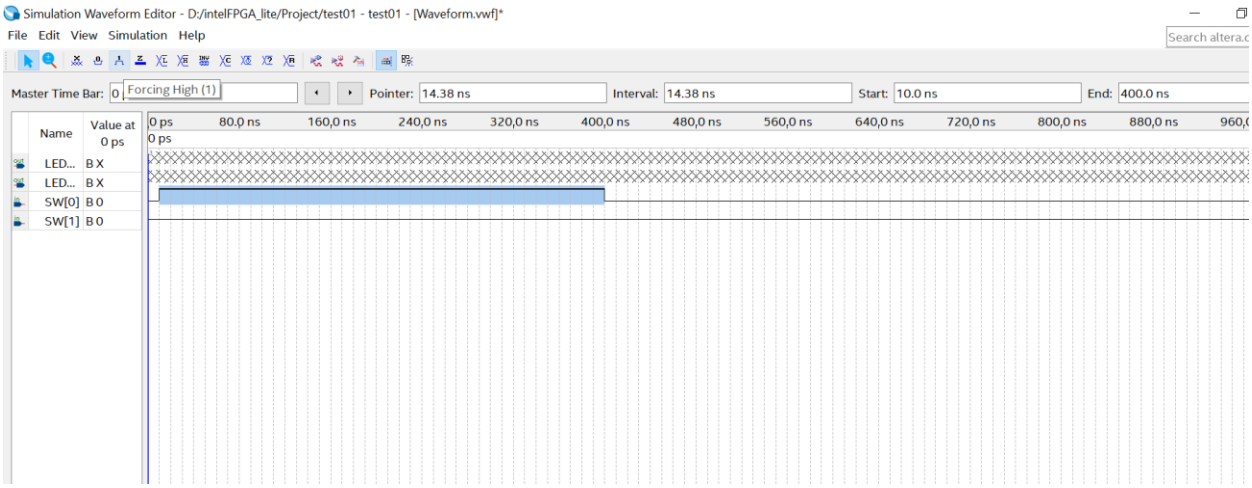


Figure 18: Setting of the test values

The input waveform could be generated as follow: Click on the waveform name for the SW0 node. Once a waveform is selected, the editing commands in the Waveform Editor can be used to draw the desired waveforms. Commands are available for setting a selected signal to 0, 1, unknown (X), high impedance (Z), don't care (DC), inverting its existing value (INV), or defining a clock waveform. Each command can be activated by using the Edit > Value command, or via the toolbar for the Waveform Editor. The Edit menu can also be opened by right-clicking on a waveform name.

In order to set SW0 in the time interval 0 to 10ns, press the mouse at the start of the interval and dragging it to its end, which highlights the selected interval, and choosing the logic value 1 in the toolbar.

Save the file in the project directory.

- **Step 4:** Set simulation period:

Select ***Edit →Set End time…*** (Fig.32) to set simulation period, fill in the "Time" box and click OK.

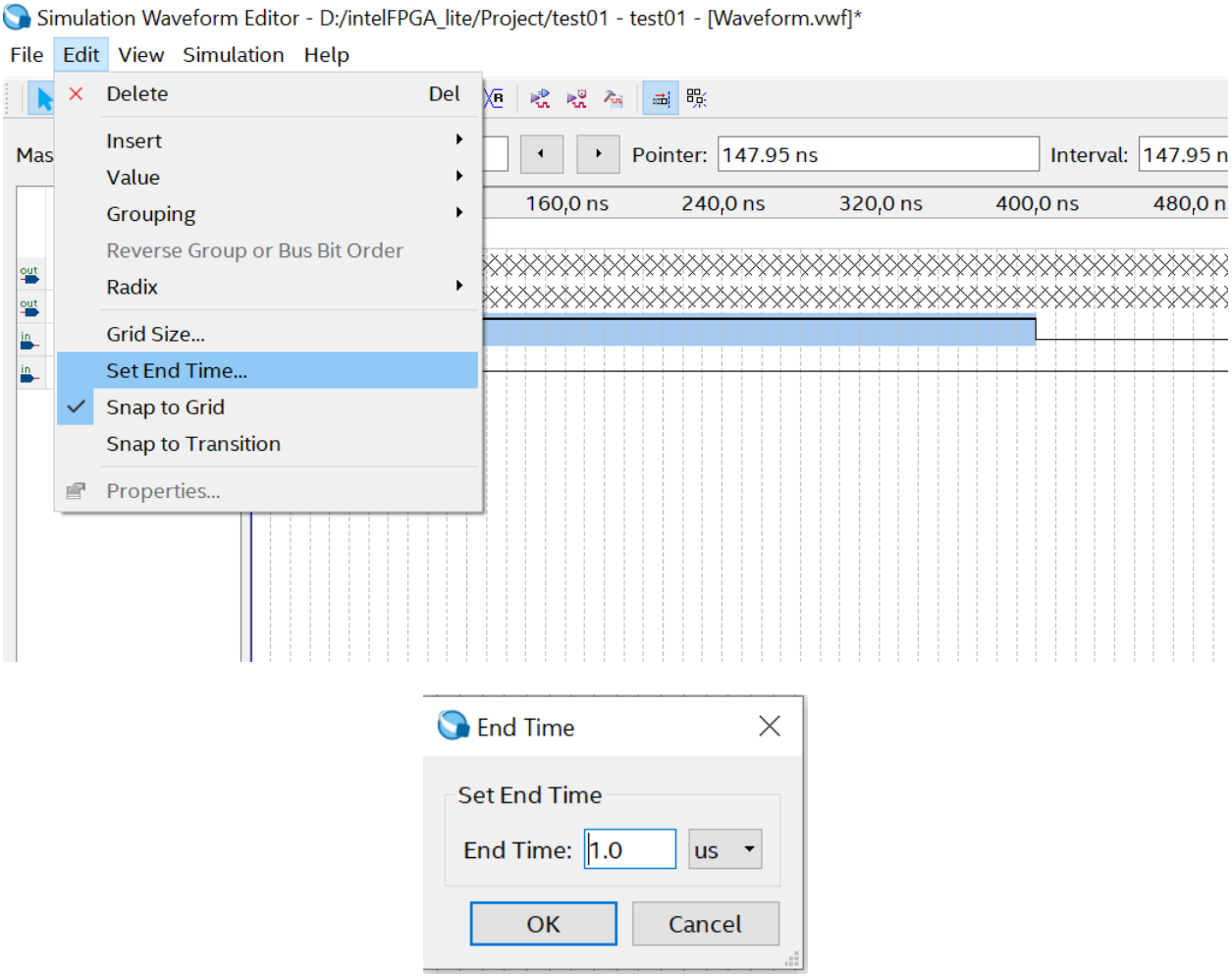


Figure 19: Setting simulation period

- **<u>Step 5:</u>** Run simulation

Choose ***Simulation → Run Functional Simulation Netlist*** to start simulation. We can see that output LEDR is equal to input SW as desired (Figure 33).

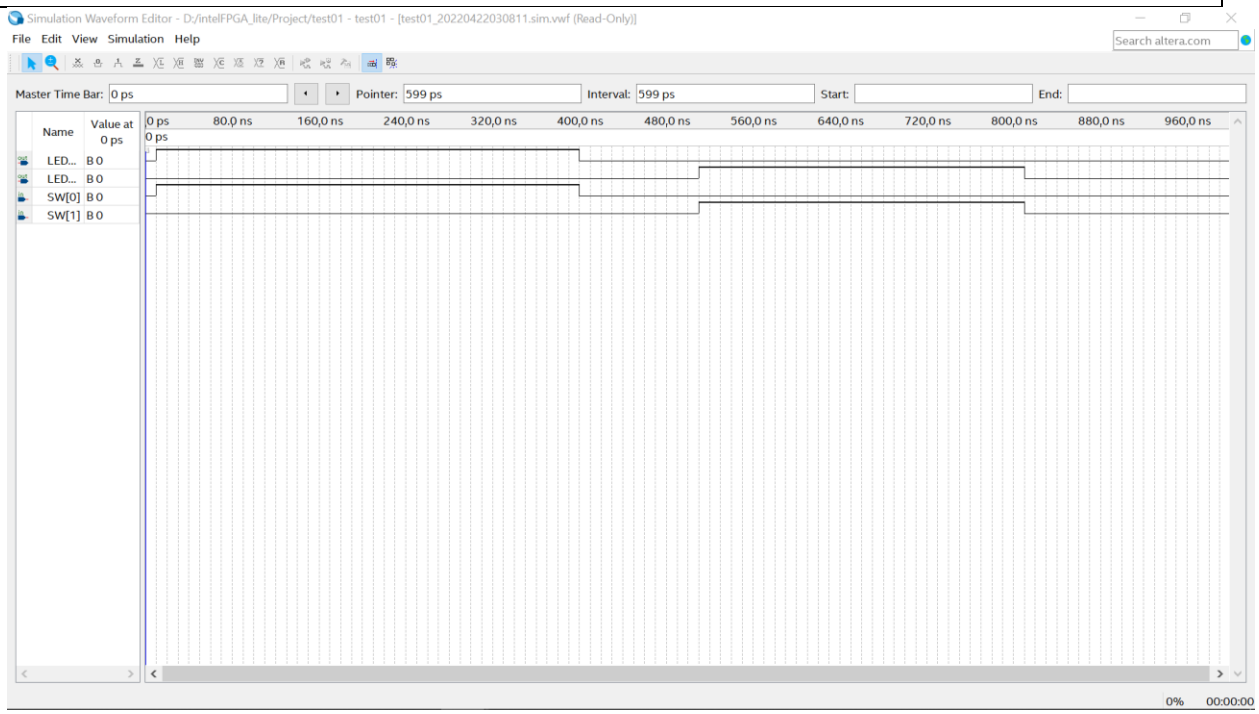


Figure 20: Successful simulation

3. Programming the FPGA:

The configuration data is transferred from the host computer (which runs the Quartus 18.1 software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board. To use this connection, it is necessary to have the USB-Blaster driver installed. Before using the board, make sure that the USB cable is properly connected and turn on the power supply switch on the board as in Fig.34.

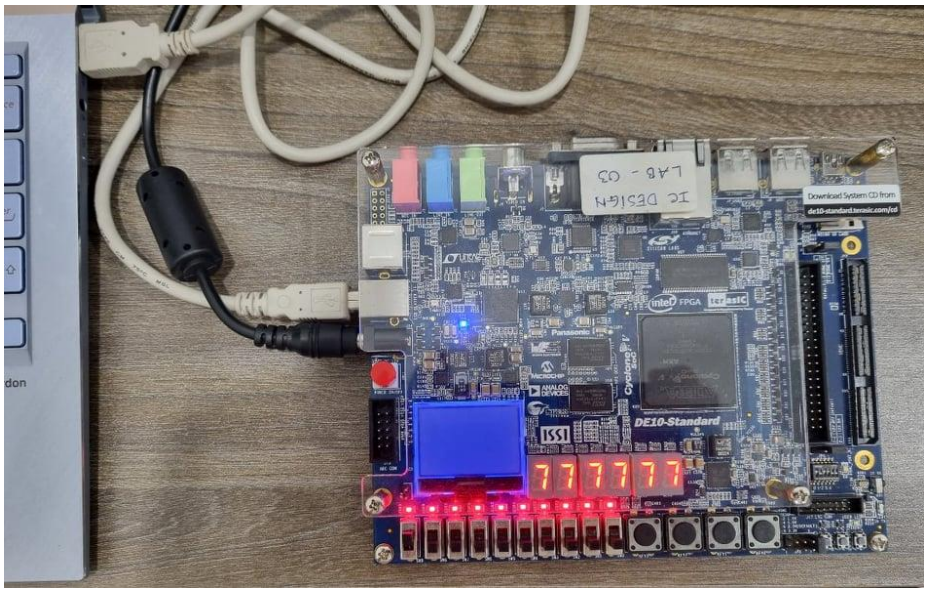


Figure 21: FPGA and computer connection

Select ***Tools*** > ***Programmer*** to reach the window in Fig.35.

There are some points needed to pay attention:

- In box "**HardwareSetup**", choose driver "**USB-Blaster**".

- Click ***Auto Detect***, choose ***5CSXFC6D6,*** click ***OK***.

- Choose the FPGA device (5CSXFC6D6), then click ***Change File***.

- If the file test01.sof is not already listed, choose **Folder output_files**. The extension .sof stands for SRAM Object File, a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device.

- Box **Program/Configure** have to be checked.

- Press Start to configure FPGA kit. If the Progress bar displays **100% (Successful)**, the configuration data has been downloaded successfully.

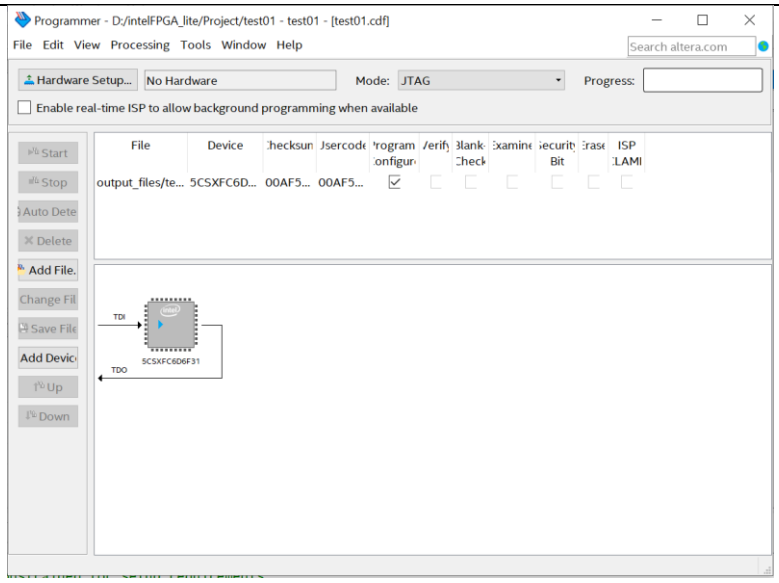
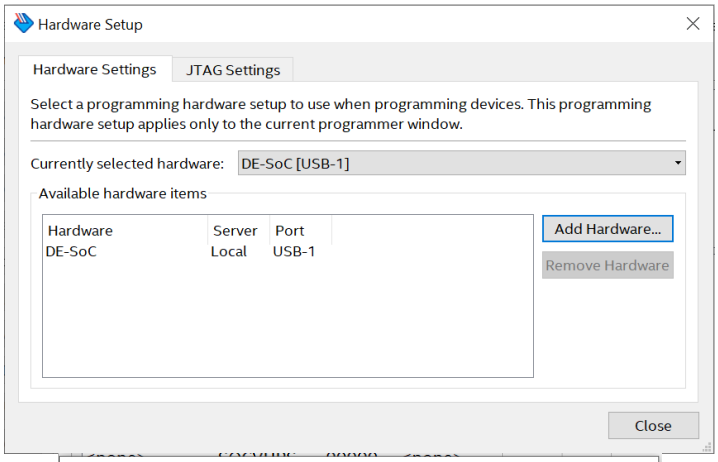
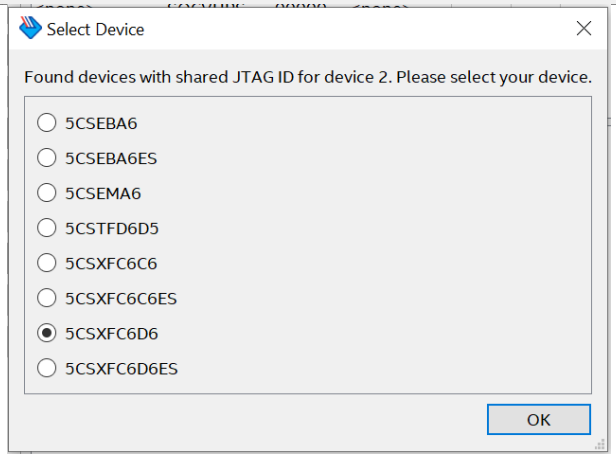


Figure 22: The Programmer window





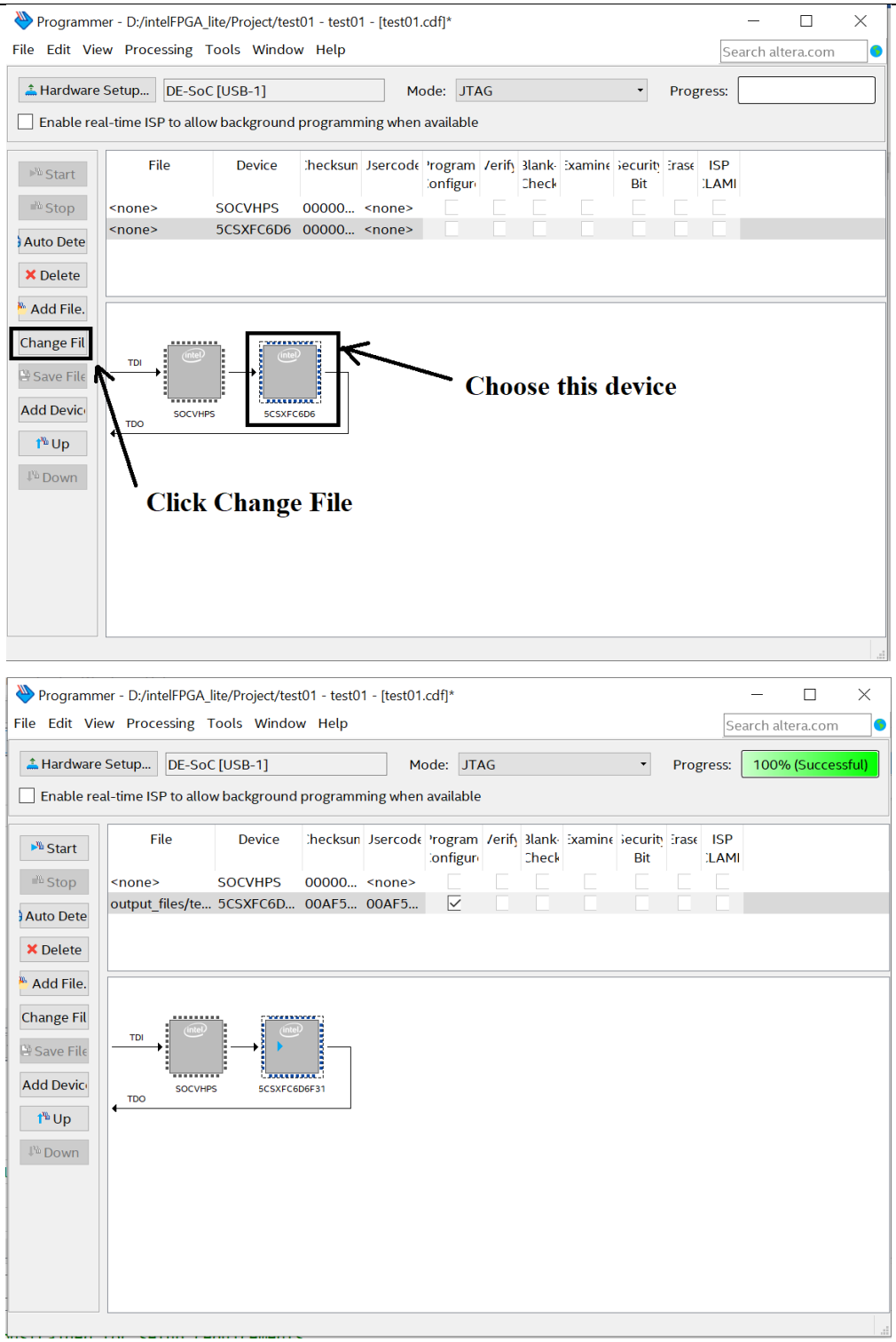


Figure 23: Successful configuration

It is observed that when the SW0 is on, LEDR0 lights up and vice versa. The SW1 controls LEDR1 in the same way.

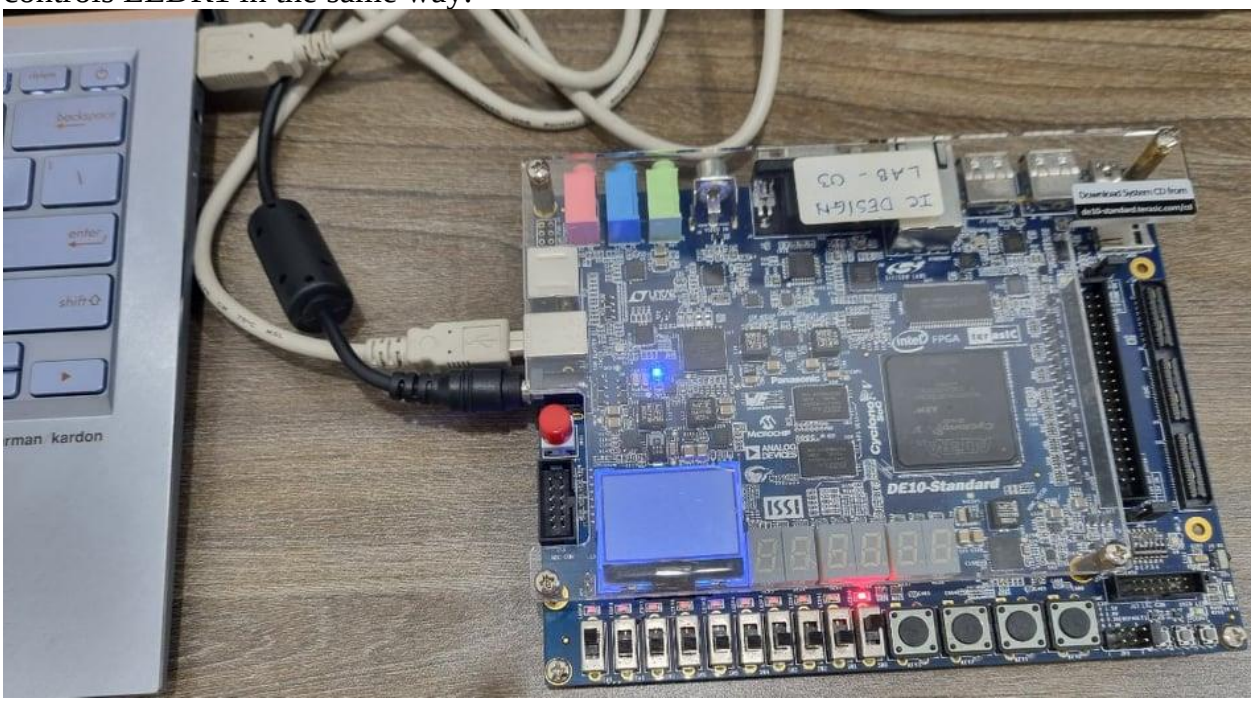


Figure 24: Verify the project.