Quartus Synthesis Tutorial

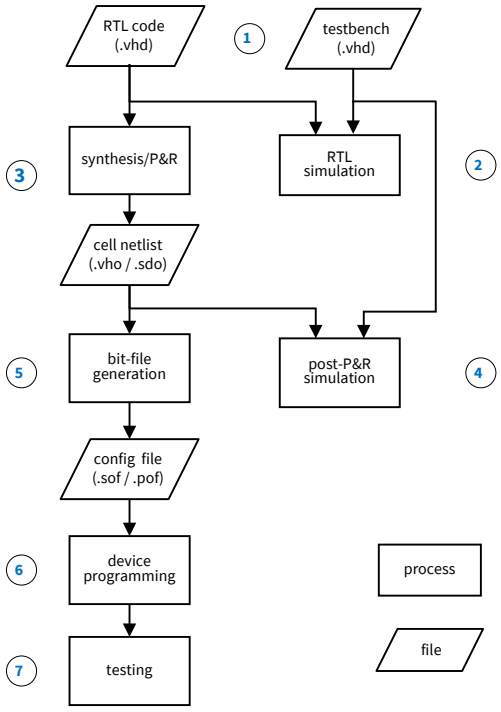
By Dr. P. Chu, Dept EECS, Cleveland State University

1 INTRODUCTION

Intel (formerly Altera) Quartus is a comprehensive EDA software package to develop and implement custom circuit for the Intel FPGA devices. The "lite" edition is free and can be downloaded from the Intel PSG (programmable solution group) site. The tutorial provides a quick overview of the synthesis flow. It is based Quartus Prime Lite Edition v18.1.

2 OVERVIEW OF DEVELOPMENT FLOW

The simplified FPGA development flow is



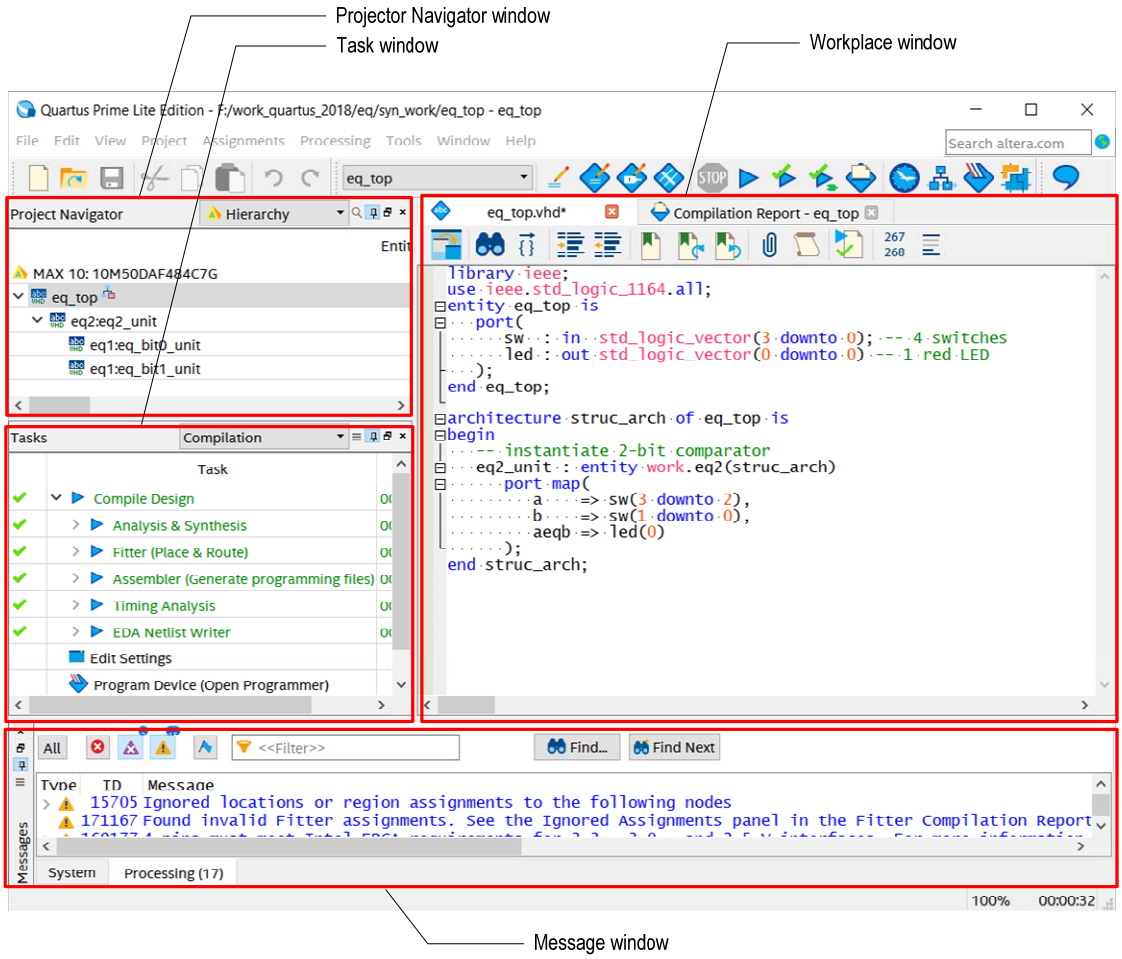
The basic steps are

1. Develop VHDL codes for RT-level design and testbench (both .vhd file).
2. Perform RT-level simulation.
3. Perform synthesis and placement-and-routing to software generate a cell-level netlist file (.vho) and timing delay file (.sdo).
4. Perform post-placement-and-routing simulation.
5. Generate FPGA configuration file (.sof or .pof).
6. Download the file (i.e., "program") the FPGA device.
7. Test the physical circuit.

The synthesis and implementation flow is on the left track, including Steps 1, 3, 5, 6, and 7. This tutorial focuses on this track.

3 OVERVIEW OF QUARTUS LAYOUT

The default Quartus GUI window is



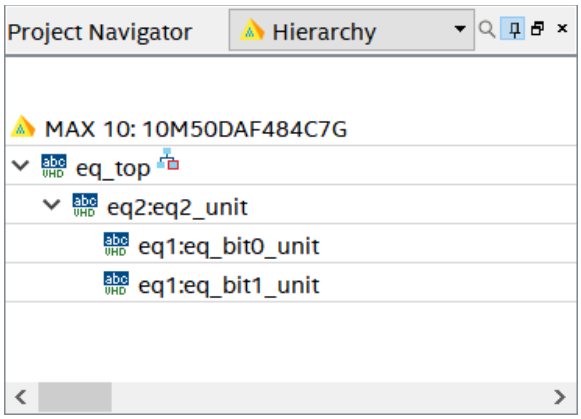
Its menu items and frequently used action icons are displayed on top. The remaining is divided into four smaller windows:

- Project Navigator window (top left)
- Tasks window (middle left)
- Messages window (bottom)
- Workplace area (top right)

Note that a window may contain multiple pages and the tabs at the top or bottom are used to select the desired page. Each window may be resized, moved, docked, or undocked.

3.1 PROJECT NAVIGATOR WINDOW

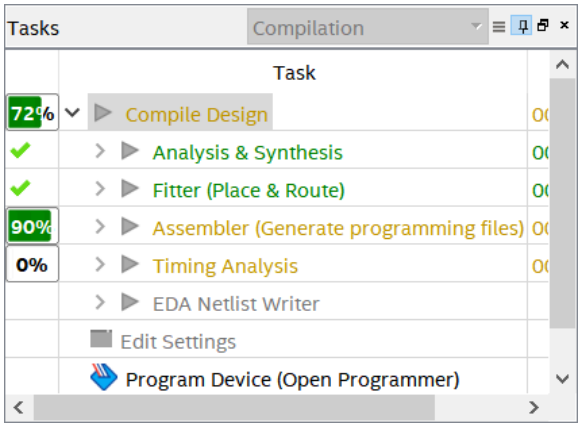
The Project Navigator window shows the design hierarchy, files, or design units associated with the project. The desired page can be selected from the pull-down menu on the top of the Project Navigator window. The hierarchy page displays the design in a hierarchical tree, starting with the top-level module. The lower-level modules can be expanded and hidden as needed. This information becomes available after we perform the initial analysis and elaboration process. The Files and Design Units pages list the design files and units in the projects, respectively. Double-clicking on a module, a file, or a unit opens the corresponding file in the Workplace area. The following is a sample Project Navigator window



It displays the hierarchy of the 2-bit comparator. The top row indicates the target FPGA device for the design.

3.2 TASKS WINDOW

The Tasks window allows a user to gather relevant processes (known as a *flow*) and show them in a flow-based layout. Several predefined flows are provided and can be selected from the Flow list. The default is the Compilation flow, which represents the left track of development flow in Section 1. Following is a snapshot:



It shows the processes as a flow from top to bottom. The left portion shows the progress of the process and a checkmark is placed when a process is successfully completed. The details of a process can be expanded or hidden as needed. We only use this flow in our Lab.

3.3 MESSAGES WINDOW

The Message window displays status messages, errors, warnings, etc. We can select the appropriate tab to get the desired information.

3.4 WORKPLACE AREA

The workplace area is the remaining area in the GUI window. It can contain multiple document windows, such as HDL code, reports, schematics, and so on. We can view and edit various types of files in this area.

4 QUARTUS DEVELOPMENT FLOW

The major steps are of developing flow are

1. Create a design project with HDL codes and constraints.
2. Compile the project.
3. Program the FPGA device.
4. Test the physical device.
5. The detailed procedure is demonstrated in the following subsections.

We use the 2-bit comparator discussed in Chapter 1 for the tutorial. The three files are

- eq1.vhd: 1-bit comparator
- eq2.vhd: 2-bit comparator composed of two 1-bit comparators
- eq2_top.vhd: top-level code with DE10-lite port names

The listing of these files are included in the Appendix.

In the Quartus II GUI, the same action can be invoked in multiple ways. For example, we can start the compiling process by selecting the proper menu item, clicking the icon on the top, or clicking the process in the Tasks window. We use the menu in the tutorial.

4.1 FILE STRUCTURE

Both synthesis and simulation generate a lot of intermediate and auxiliary files. It is a good practice to use separate folders for them. The recommended file structure for a simple project (say project #1) is

- ../proj1/src: folder for the HDL source files, constraint file, and test bench
- ../proj1/syn_work: working directory for synthesis
- ../proj1/sim_work: working directory for simulation

Only the files in the src folder need to be saved or backed up.

4.2 CREATE A DESIGN PROJECT

A Quartus project contains the basic information of a design, which includes the location of the working directory, the top-level entity, the source files, the target device, the constraints, and tool settings. There are several tasks associated with this step:

1 Create a project.
2 Assign a device.
3 Create new HDL files.
4 Check the code syntax.
5 Add existing HDL files.
6 Import a pin-assignment constraint file.

4.2.1 Create a project

During project creation, Quartus's New Project Wizard is invoked and it guides users through five pages to set up a new project. The initial project settings and relevant information are specified in this process. For flexibility, we manually update the setting later and only use the wizard to specify the location of the working directory and project name. A new project can be created as follows:

- Launch the Quartus program.
- In Quartus menu, select File ⇒ New Quartus Prime Project. Click the OK button. The New Project Wizard dialog appears.

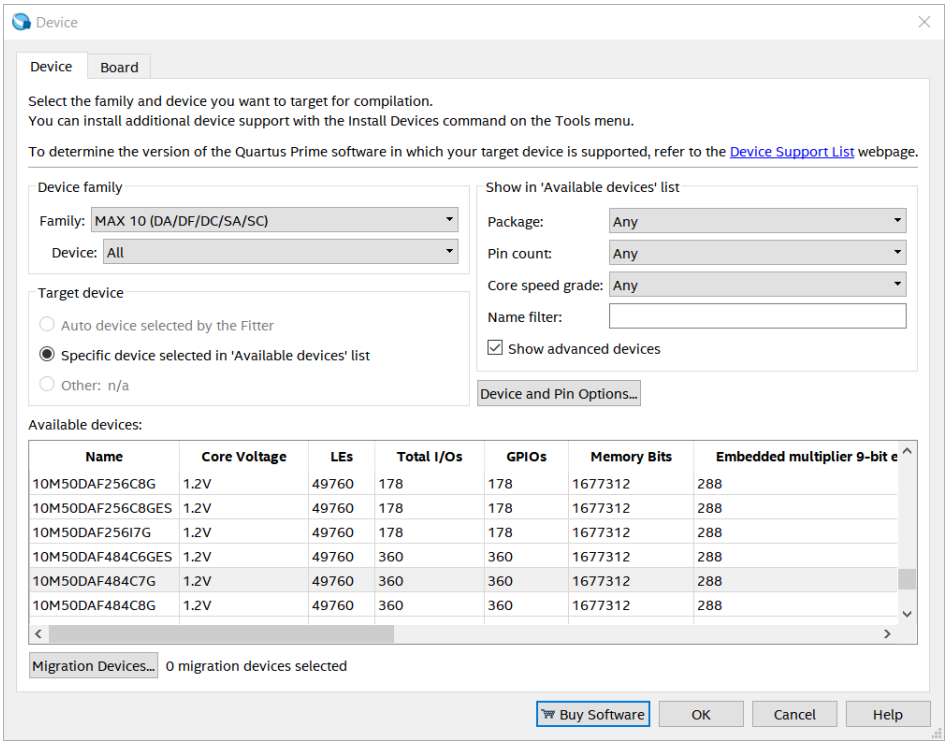
- Click the Next> button to go to the next page.
- In the working directory for this project field, navigate to or create a project folder and the working directory (.../eq_top/syn_work)
- Enter eq_top for the project name and the top-level entity name.
- Click the Finish button to exit the dialog.

If all the relevant information and files are known in advance, we can continue with the wizard and skip some of the subsequent tasks.

4.2.2 Assign a device

Since we plan to implement the design on the prototyping board, we must specify the board's FPGA chip as the target device. On the DE10-lite board, it is a MAX 10 10M50DAF484C7G device. We can specify the target device as follows:

- In the Quartus II menu, select Assignments ⇒ Device...
- The Device dialog appears



- Select MAX 10 in the Family field and click the Specific device selected in 'Available devices' list button.
- Scroll down the device list and select 10M50DAF484C7G.
- Click the OK button to complete the selection.

4.2.3 Create a new HDL file

A project may contain one or multiple HDL files. If a file does not exist, we must create a new source file. Quartus contains an HDL editor for this task. The following procedure we create a new file for the 1-bit comparator code. The procedure is:

- In Quartrus menu, select File ⇒ New. The New dialog appears.
- Select VHDL and then click the OK button.
- A new text editor window appears in the Workplace area.

- In Quartrus menu, select File ⇒ Save. The Save as dialog appears.
- Create a new folder .../src (for source code) and enter eq1.vhd in the file name: field.
- Check the Add file to current project box, and click the Save button.
- Enter the HDL code.
- In Quartrus menu, select File ⇒ save to save the file.

The Quartus HDL editor is language sensitive and colors the language constructs for clarity. It also provides a collection of *pre-defined templates* of various language segments to facilitate the code entry. To insert a template into a file, select Edit ⇒ Insert Template in Quartus menu, expand the VHDL row, and double-click the desired template.

4.2.4 Check the code syntax

After completing a new HDL file, we need to check the syntax of the code:

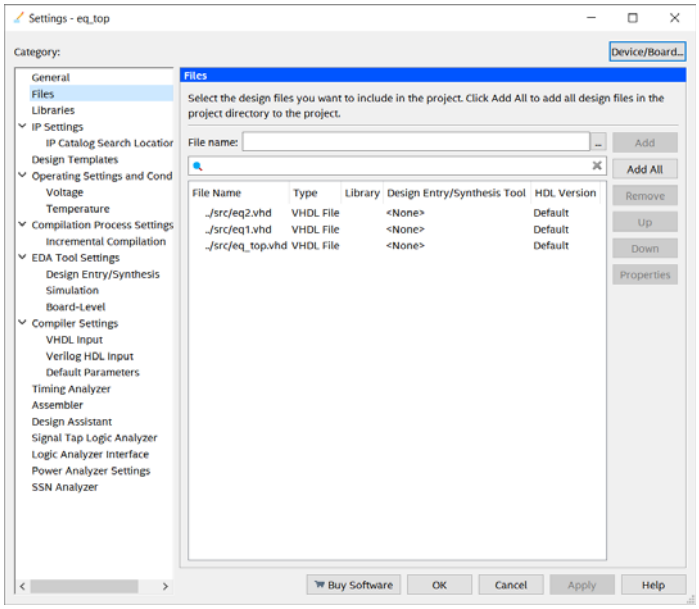
- Select the desired file window in the Workplace area.
- In Quartrus menu, select Processing ⇒ Analyze Current File.

The bottom Messages window displays the progress of the process and reports errors and warnings. Double-clicking an error message leads to the offending line in the file. We can correct the problem, save the file, and repeat the syntax checking process until all syntax errors are eliminated. The analysis process only checks the syntax of the current file and does not perform elaboration. Other errors may still occur when the entire project is compiled.

4.2.5 Add existing HDL files

A project usually contains multiple files and some files have been developed in previous projects or can be obtained from other sources. We can add existing HDL files to a project. The procedure is:

- In Quartrus menu, select Assignments ⇒ Settings.... The Settings dialog appears. In the left panel, click on the Files item.
- The Setting dialog shows the relevant file information in the right panel:



- Click the … button in the Files Name row and a Select File window appears.
- Navigate to the proper directory, select the file, and click the Open button to return to the Settings dialog.
- Click the Add button to add the file to the project.

- Repeat the process for multiple files.
- Click the OK button to complete the addition.

4.2.6 Import a pin-assignment constraint file

Constraints are certain conditions imposed on the synthesis and placement and routing processes. For our purposes, the primary type of constraint is the pin assignment of top-level I/O ports and the minimal clock rate for a sequential circuit. During the placement and routing process, an I/O port of the top-level module must be mapped to a physical pin of the FPGA device. Since the peripherals' I/O signals are already permanently connected to the designated FPGA's pins on the DE10-lite prototyping board, we must ensure that the HDL module's I/O ports are mapped to the corresponding pins.

In the eq2 circuit, we can connect the a and b ports to four slide switches and the aeqb port to an LED to verify the physical operation of the circuit. For the DE10-lite board, the corresponding pins are L22, L21, M22, V12, and R20. The pin assignment can be performed with Quartus's Assignment Editor, in which we can manually assign top-level I/O ports to FPGA's pins as well as specify the desired I/O standards. The process is tedious and error-prone, especially for a project with a large number of I/O ports.

A better alternative is to use a pin assignment file that defines all connected I/O peripheral signals of the DE10-lite board. The chu_de10_pin.csv file is created for this purpose. The port names defined in are similar to those on the DE10-lite board. Following are the names for the switches and LEDs:

- `sw:`.10-bit slide switches
- `btn:`.2-bit pushbutton switches
- `led:` 10-bit discrete LEDs
- `hex5`, `hex4`, …, `hex0`: six 7-segment (8-bit) LED displays

The best way to incorporate these pre-defined board I/O port names is to define a top-level "wrapping" VHDL file that maps the "logical" port names to the pre-defined names. The eq_top.vhd file in the appendix demonstrates the mapping.

The procedure to import the pin assignment file is:

- In Quartrus menu, select Assignments ⇒ Import Assignment... The Import Assignment dialog appears.
- Navigate to the proper directory, select the chu_de10_pin.csv file.
- Click the OK button to complete the process.

4.3 COMPILE THE PROJECT

There are several tasks in this step:

1. Specify the top-level module.
2. Perform synthesis and placement-and-routing.
3. Examine the compilation report.

The last task provides additional information about the design but can be omitted.

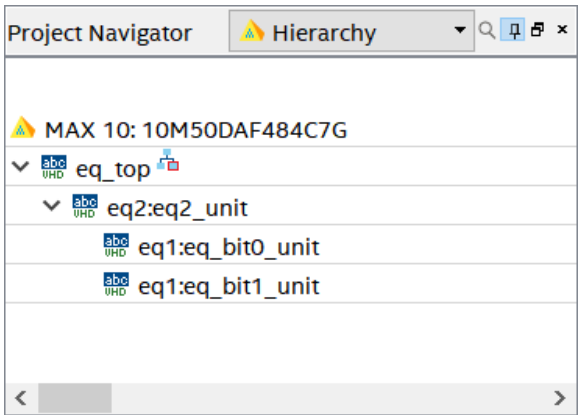
4.3.1 Specify the top-level module

In Quartus, the default top-level module name is set to be the top-level HDL file name. After adding and creating all HDL files, we can change the top-level module if needed. The procedure is:

- In Quartrus menu, select Assignments ⇒ Settings.... The Settings dialog appears.
- In the left panel, click the General item.
- Enter the top-level entity name, eq_top, in the Top-level entity field.

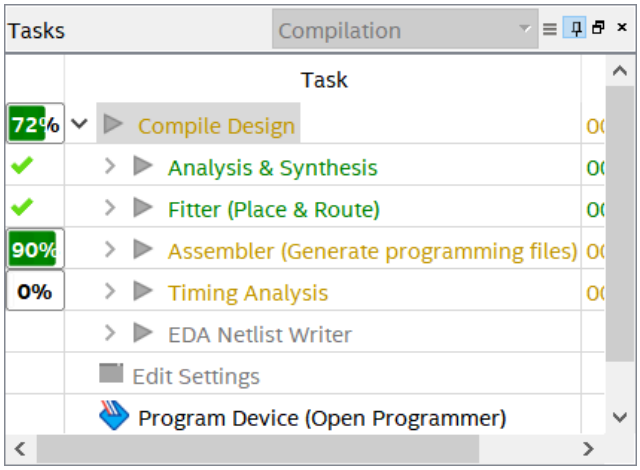
- Click the OK button to complete the process.

After compiling, Quartus analyzes the files based on the top-level module and establishes the hierarchical structure, which is displayed in the Project Navigator window



4.3.2 Perform synthesis and placement-and-routing

Compiling contains the processes to analyze design hierarchy, to perform elaboration, synthesis, and placement and routing, and to generate the configuration file. It can be invoked by selecting Processing ⇒ Start Compilation in Quartus menu. The progress of the compiling is displayed on the Tasks window:

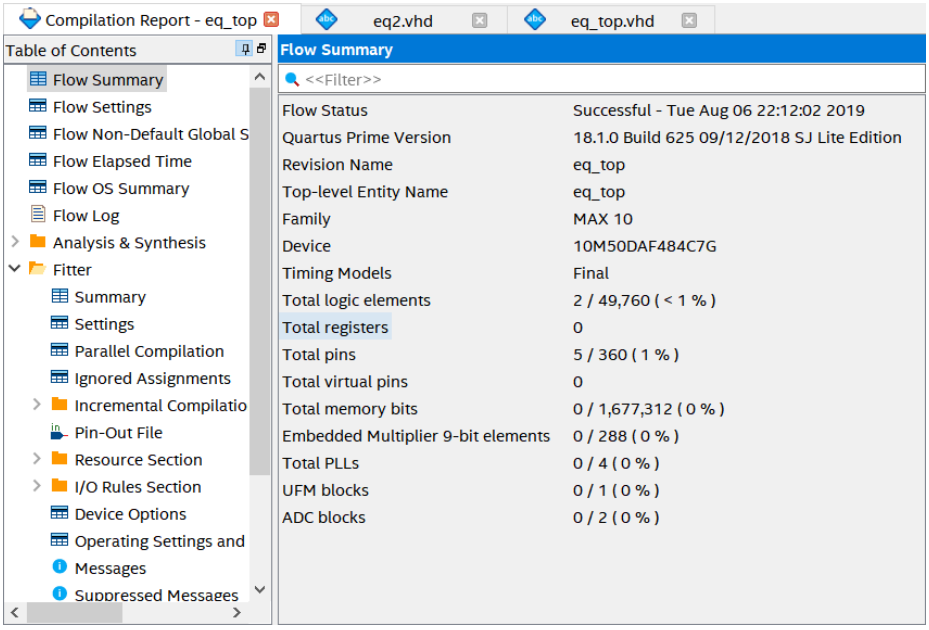


A green checkmark will be placed on the left if the corresponding process is successful.

Although the syntax of individual files is checked earlier, the code may contain constructs that cannot be synthesized or may lead to poor implementation (such as a combinational loop). The error and warning messages are displayed in the Messages window. We must correct the problems and repeat the compiling process if needed.

4.3.3 Examine the compilation report

After successful compilation, a report window is generated and opened automatically on the Workplace area:



It can also be invoked later by selecting Processing ⇒ Compilation Report in Quartus menu. The window contains information for the overall flow as well as detailed reports for individual processes. A list of more detailed reports can be obtained by clicking on the corresponding directory. The compilation report is quite comprehensive. For our purposes, the following information is of special interest:

- Resource utilization
- Use of I/O pins

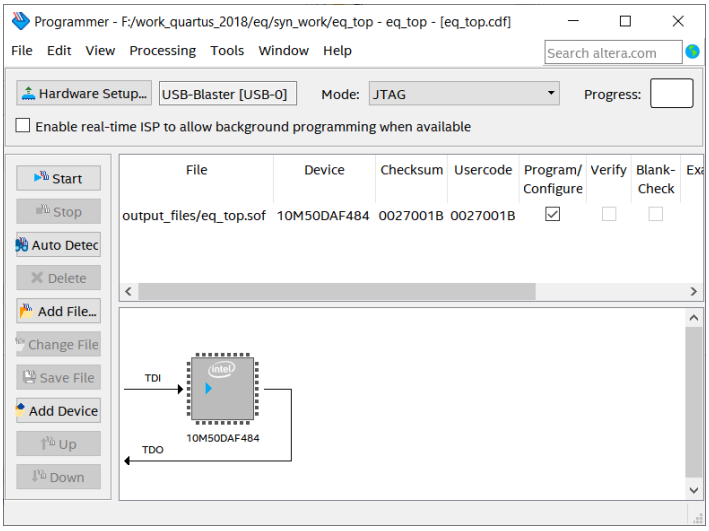
The resource utilization basically indicates the size of the resulting circuit in terms of the number of logic elements. It also shows the usage of various macro cells. This information can be found in the Flow summary report or the Fitter's Summary report. The previous report indicates that 2 logic elements (out of 49760) are used to synthesize the two-bit comparator.

Pin assignment is an error-prone task. To verify the pin locations, we can expand Fitter and then Resource Section and check the Input Pins and Output Pins reports for the pin assignment of HDL module's I/O ports.

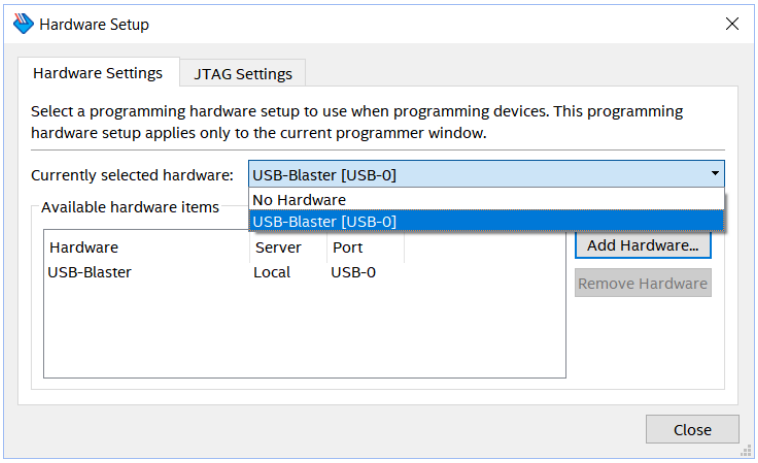
4.4 Program the FPGA device

The last step is to download the configuration file to the FPGA device on the DE10-lite board. An Intel MAX10 device is an SRAM based device but contains an on-chip flash memory module. At power-on, the data in flash memory is loaded to the configuration SRAM. To program the device's SRAM memory with the pre-designed FPro system configuration file, the steps are

- In Quartrus menu, select Tool ⇒ Programmer. The Quartus Prime Programmer window (a stand-alone utility) appears:



- Make sure that the Hardware Setup... field is set to be USB-Blaster. This field specifies the intended programming adapter. It may show No hardware initially. To correct this,
    - Click the Hardware Setup button to invoke the Hardware Setup dialog.



    - Click the Currently selected hardware pull-down menu, select USB-Blaster, and then click Close. Don't use the Add Hardware button.
- Make sure that the Mode field is set to JTAG.
- The default file, eq_top.sof, should already show up. If not, click the Add file... button on the left panel and navigate to select the eq_top.sof file. The type of FPGA device is detected from the file automatically.
- Make sure that the Program/Configure box is checked.
- Click the Start button to start the downloading process. The status is shown in the Progress bar on the top right and should display 100% (Successful) afterward.

1. Close the programmer window and return to Nios II Eclipse.

Since the flash memory can keep data after the power is removed (i.e., it is non-volatile), this task does not need to be repeated unless the baseline FPro system is modified.

4.5 TEST THE PHYSICAL DEVICE.

The FPGA device is programmed now. We can use the four slide switches and an LED to verify the operation of the circuit.

5 FURTHER READING

Quartus is a very complex software package. The tutorial just helps you "jump-start" the leaning process. Here are some documentation

- In Quartus, select Help ⇒ Help topics to learn an individual topic.
- Google "Quartus User Guides" to access the handbook. There are more than a dozen users guides on various Quartus subjects.

6 REFERENCE

- *FPGA Prototyping by VHDL Examples 2nd edition: Xilinx MicroBlaze MCS SoC*
- *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3*

7 APPENDIX: CODE LISTINGS

Following are the three files used in this tutorial.

```
-- File eq1.vhd
library ieee;
use ieee.std_logic_1164.all;
entity eq1 is
   port(
      i0, i1 : in  std_logic;
      eq     : out std_logic
   );
end eq1;

architecture sop_arch of eq1 is
   signal p0, p1 : std_logic;
begin
   -- sum of two product terms
   eq <= p0 or p1;
   -- product terms
   p0 <= (not i0) and (not i1);
   p1 <= i0 and i1;
end sop_arch;

-- File eq2.vhd
library ieee;
use ieee.std_logic_1164.all;
entity eq2 is
   port(
      a, b : in  std_logic_vector(1 downto 0);
```

```
      aeqb : out std_logic );
end eq2;

architecture struc_arch of eq2 is
   signal e0, e1 : std_logic;
begin
   -- instantiate two 1-bit comparators
   eq_bit0_unit : entity work.eq1(sop_arch)
      port map(
         i0 => a(0),
         i1 => b(0),
         eq => e0
      );
   eq_bit1_unit : entity work.eq1(sop_arch)
      port map(
         i0 => a(1),
         i1 => b(1),
         eq => e1
      );
   -- a and b are equal if individual bits are equal
   aeqb <= e0 and e1;
end struc_arch;

-- File eq_top.vhd
library ieee;
use ieee.std_logic_1164.all;
entity eq_top is
   port(
      sw  : in  std_logic_vector(3 downto 0); -- 4 switches
      led : out std_logic_vector(0 downto 0)  -- 1 red LED
   );
end eq_top;

architecture struc_arch of eq_top is
begin
   -- instantiate 2-bit comparator
   eq2_unit : entity work.eq2(struc_arch)
      port map(
         a    => sw(3 downto 2),
         b    => sw(1 downto 0),
         aeqb => led(0)
      );
end struc_arch;
```