

# Tutorial 1.1: Review of Quartus II

CPEN 391

University of British Columbia  
January 2015/16 Term 2

The purpose of this document is to very briefly describe how to use the Altera Quartus II software packages to design digital systems, and download them to the Altera DE2 board. This is a review of what you will have learned in EECE 259/353. If you recently took EECE 353 and are familiar with running Quartus II, you can skip this tutorial.

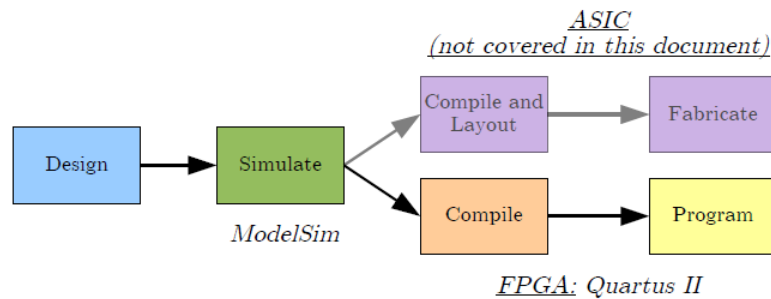


Figure 1: A typical digital design flow

A typical design flow may look like Figure 1. First, the designer will describe their circuit using either a Hardware Description Language (HDL) such as VHDL or via a schematic diagram (or a combination of both). Next, the designer will simulate their design to ensure that it functions correctly. In EECE 353, you used ModelSim built into Quartus. You are free to use ModelSim in this course as well, although you will find that most of the embedded system projects we will create do not lend themselves to efficient simulation.

The third stage is to compile the circuit, into physical hardware, such as logic gates. In this course, we will be exclusively targeting Field-Programmable Gate-Arrays, or FPGAs, as the implementation platform. For reference, and to highlight its similarity, the flow to build a custom ASIC chip is also shown. As in EECE 353, we will be using an industry-standard tool: Altera Quartus II.

Finally, we will then use Quartus II to program the FPGA chip with this output to actually realize the circuit in hardware. Tutorial 1.0 described how to install Quartus and the University Program Files. You should have installed the software (or be sitting at a departmental computer with the software installed) before working through this tutorial. Note that we will be using Version 13.0sp1 in this course; this may be different than the version you would have used in EECE 353.

# IMPORTANT NOTE

These notes have been modified from their original, previously written by Altera for use with their DE2 board.

## Quartus II Introduction Using Schematic Design

This tutorial presents an introduction to the Quartus<sup>®</sup> II CAD system. It gives a general overview of a typical CAD flow for designing circuits that are implemented by using FPGA devices, and shows how this flow is realized in the Quartus II software. The design process is illustrated by giving step-by-step instructions for using the Quartus II software to implement a very simple circuit in an Altera FPGA device.

The Quartus II system includes full support for all of the popular methods of entering a description of the desired circuit into a CAD system. This tutorial makes use of the schematic design entry method, in which the user draws a graphical diagram of the circuit. Two other versions of this tutorial are also available, which use the Verilog and VHDL hardware description languages, respectively.

The last step in the design process involves configuring the designed circuit in an actual FPGA device. To show how this is done, it is assumed that the user has access to the Altera DE2 Development and Education board connected to a computer that has Quartus II software installed. A reader who does not have access to the DE2 board will still find the tutorial useful to learn how the FPGA programming and configuration task is performed.

The screen captures in the tutorial were obtained using the Quartus II version 13.0; if other versions of the software are used, some of the images may be slightly different (but the principle is the same).

# 1 Getting Started

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 begin a new logic circuit design, the first step is to create a directory to hold its files. To hold the design files for this tutorial, we will use a directory *introtutorial*. The running example for this tutorial is a simple circuit for two-way light control.

Start the Quartus II software. You should see a display similar to the one in Figure 2. This display consists of several windows that provide access to all the features of Quartus II software, which the user selects with the computer mouse. Most of the commands provided by Quartus II software can be accessed by using a set of menus that are located below the title bar. For example, in Figure 2 clicking the left mouse button on the menu named File opens the menu shown in Figure 3. Clicking the left mouse button on the entry Exit exits from Quartus II software. In general, whenever the mouse is used to select something, the *left* button is used. Hence we will not normally specify which button to press. In the few cases when it is necessary to use the *right* mouse button, it will be specified explicitly.

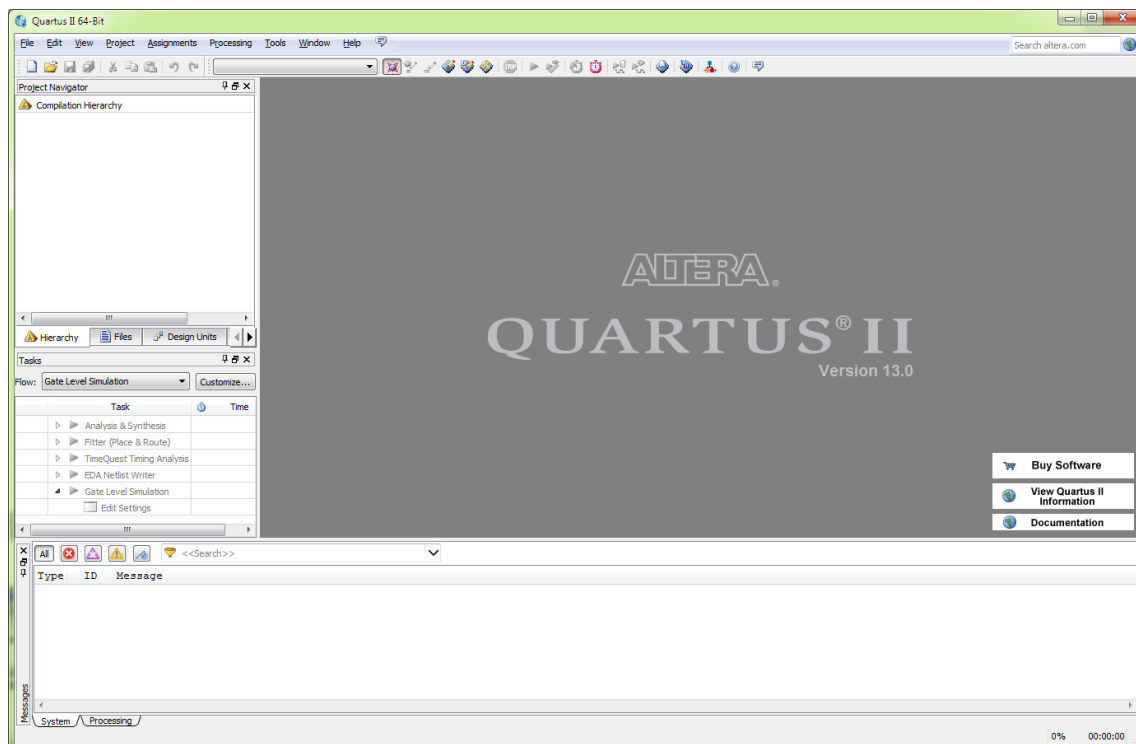


Figure 2. The main Quartus II V13.0 window.

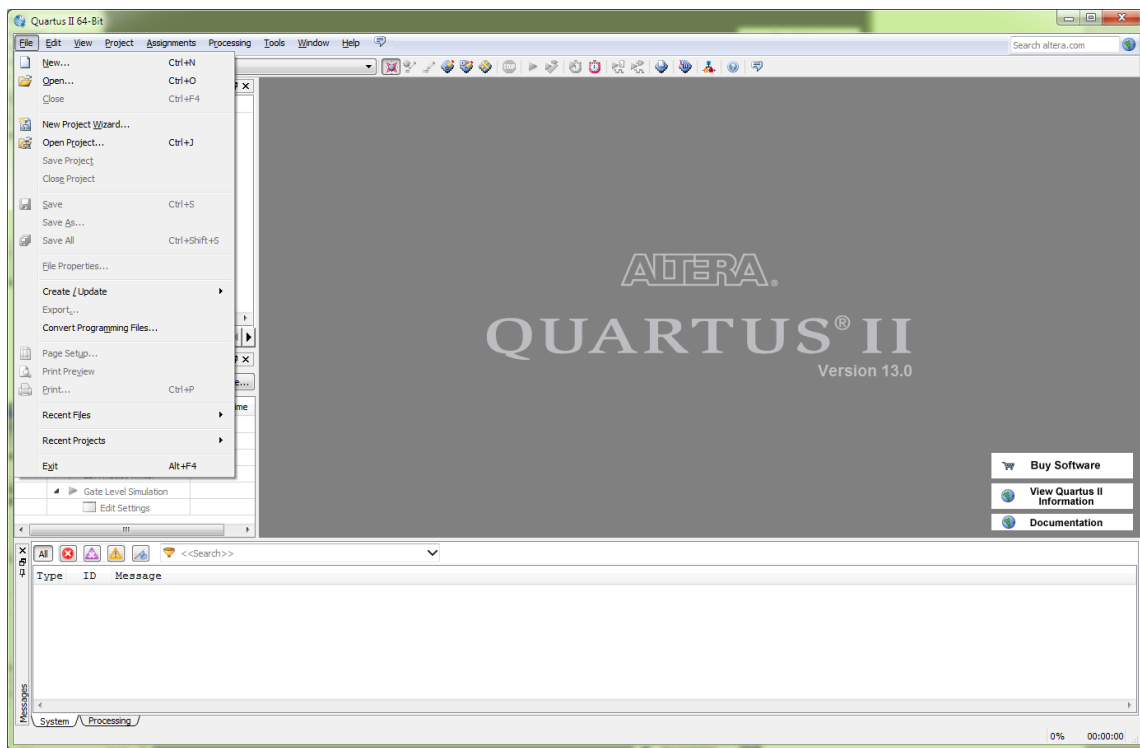


Figure 3. An example of the File menu.

## 1.1 Quartus II Online Help

Quartus II software provides comprehensive online documentation that answers many of the questions that may arise when using the software. The documentation is accessed from the menu in the **Help** window. The user can quickly search through the Help topics by selecting **Help > Search**, which opens a dialog box into which key words can be entered. Another method, context-sensitive help, is provided for quickly finding documentation for specific topics.

## 2 Starting a New Project

To start working on a new design we first have to define a new *design project*. Quartus II software makes the designer's task easy by providing support in the form of a *wizard*. Create a new project as follows:

1. Select **File > New Project Wizard** to reach the window in Figure 4, which asks for the name and directory (folder) of the project. This example uses the folder `c:\altera\13.0` (you should use a more appropriate folder such as one related to this course). Make the name of the project “light”.

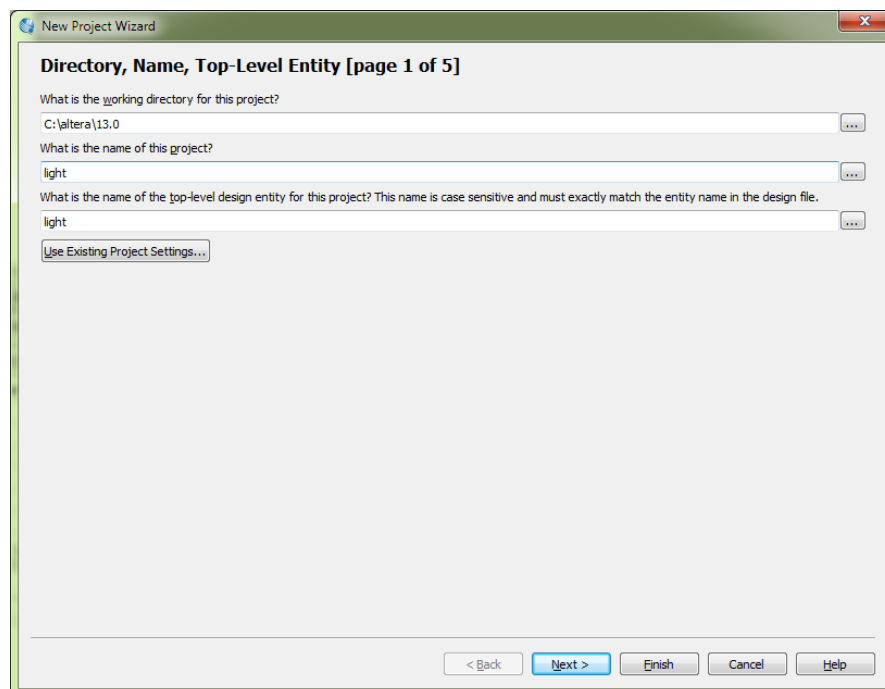


Figure 4. Creation of a new project.

2. Press the **Next** Key

Add this point the window in Figure 5 below will pop up asking if you would like to add any additional pre-existing files to the project (i.e. things we might have designed before and wish to reuse in this project). At this point we can add As this is our 1<sup>st</sup> project, just click [Next](#).

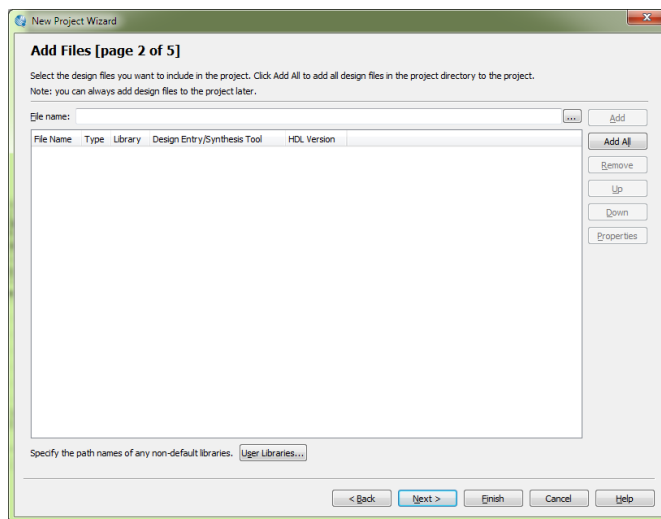
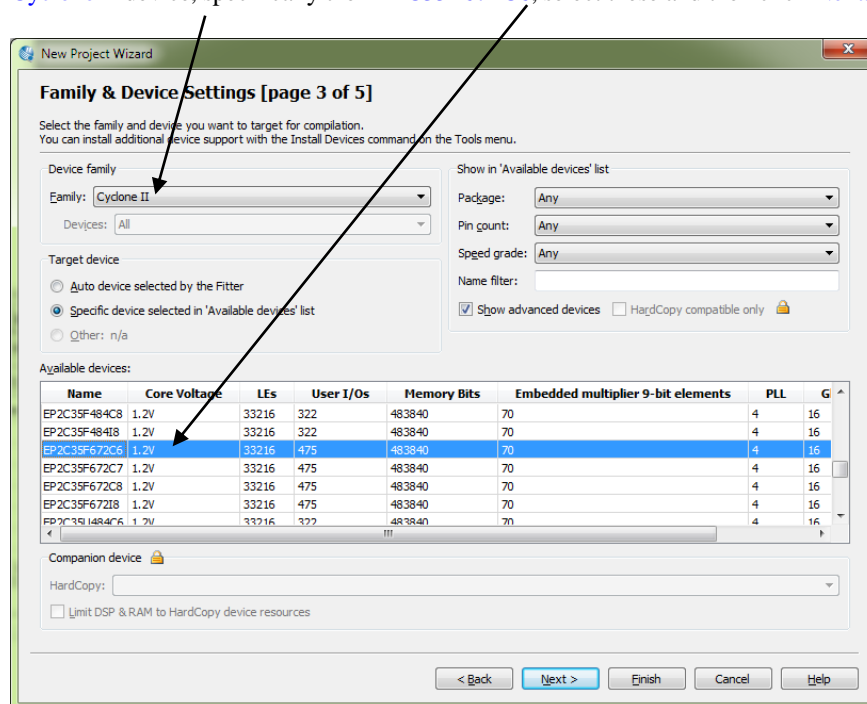
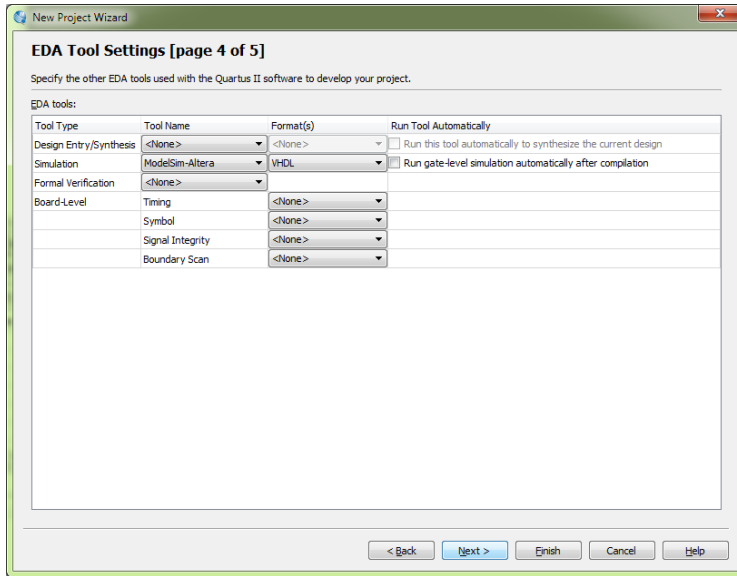


Figure 5. Quartus II software to add files to the project.

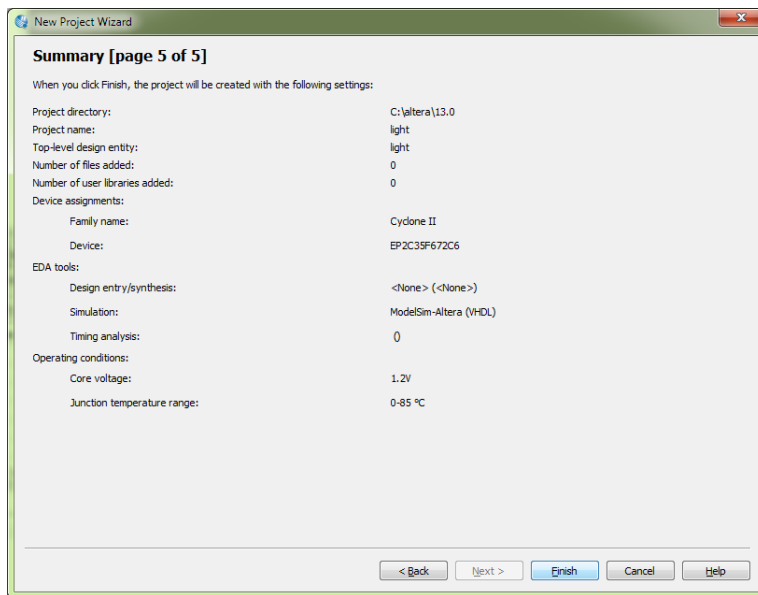
You will now be asked to choose the type of FPGA that is present on your development board. The DE2 board uses a [Cyclone II](#) device, specifically the [EP2C35F672C6](#), select these and then click [Next](#).



The following window pops up. The user can specify any third-party (non-Altera) EDA (Electronic Design Automation) tools that should be used. A commonly used term for CAD software for electronic circuits is *EDA tools*, where the acronym stands for Electronic Design Automation. Since we will rely solely on Altera's Quartus II tools, we will not choose any other tools. Press [Next](#).



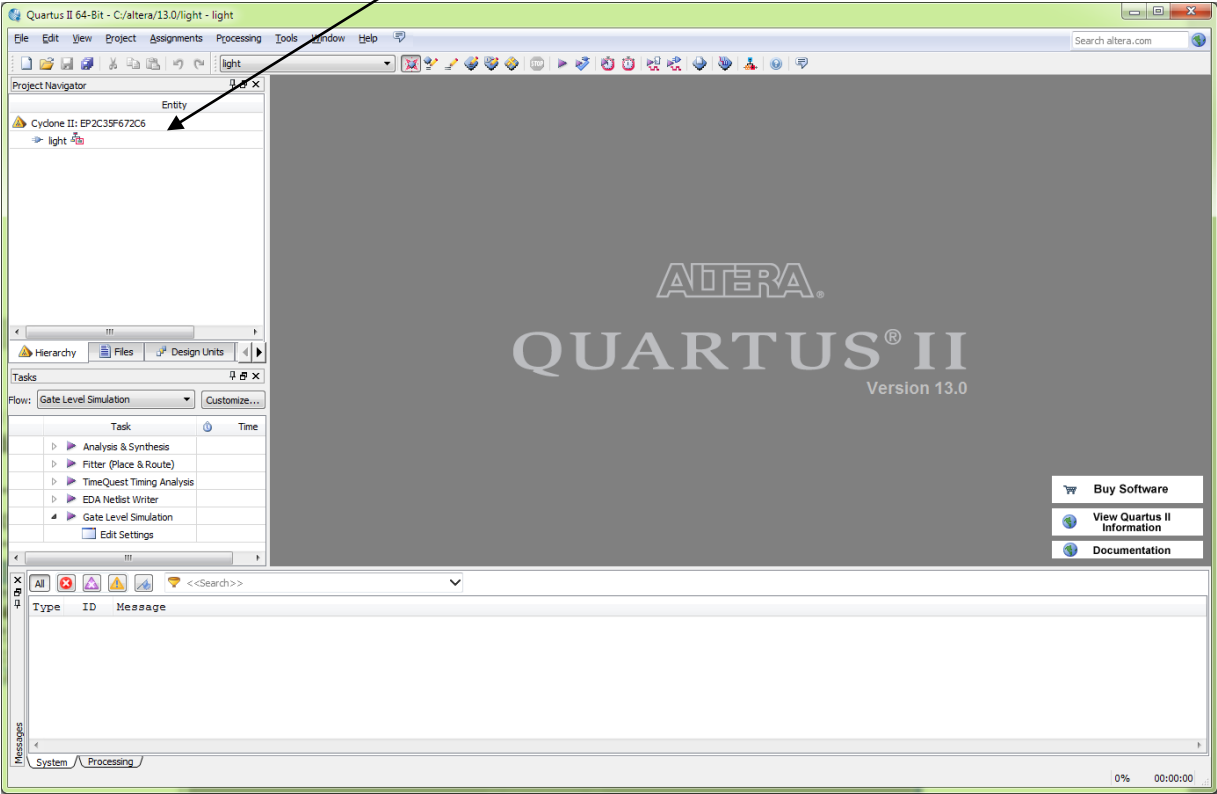
A summary of the chosen settings appears in the screen shown below



Press [Finish](#)

You will now be returned to the main Quartus II window, but with the project name *light* specified as the new project in the project navigator window

Note Project and Device Settings





### 3 Design Entry Using the Graphic Editor

As a design example of drawing a circuit based around logic gates, we will use the two-way light controller circuit shown below. The circuit can be used to control a single light from either of the two switches,  $x_1$  and  $x_2$ , where a closed switch corresponds to the logic value 1. The truth table for the circuit is also given in the figure. Note that this is just the **Exclusive-OR** function of the inputs  $x_1$  and  $x_2$ , but we will implement it using the gates shown.

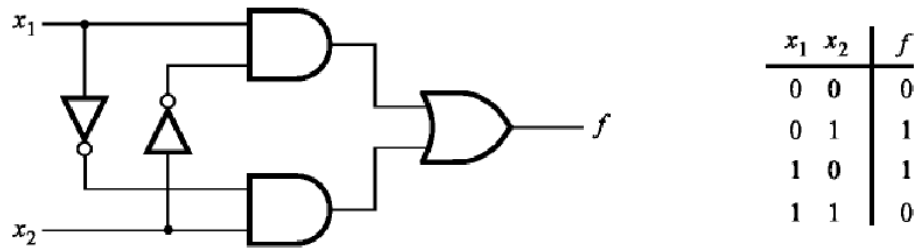
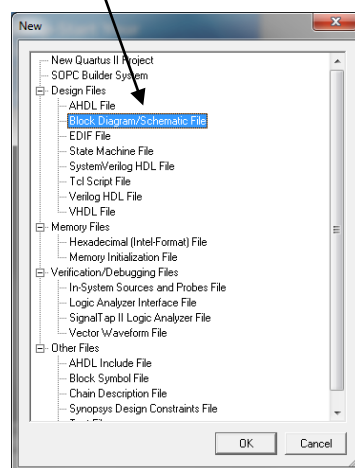
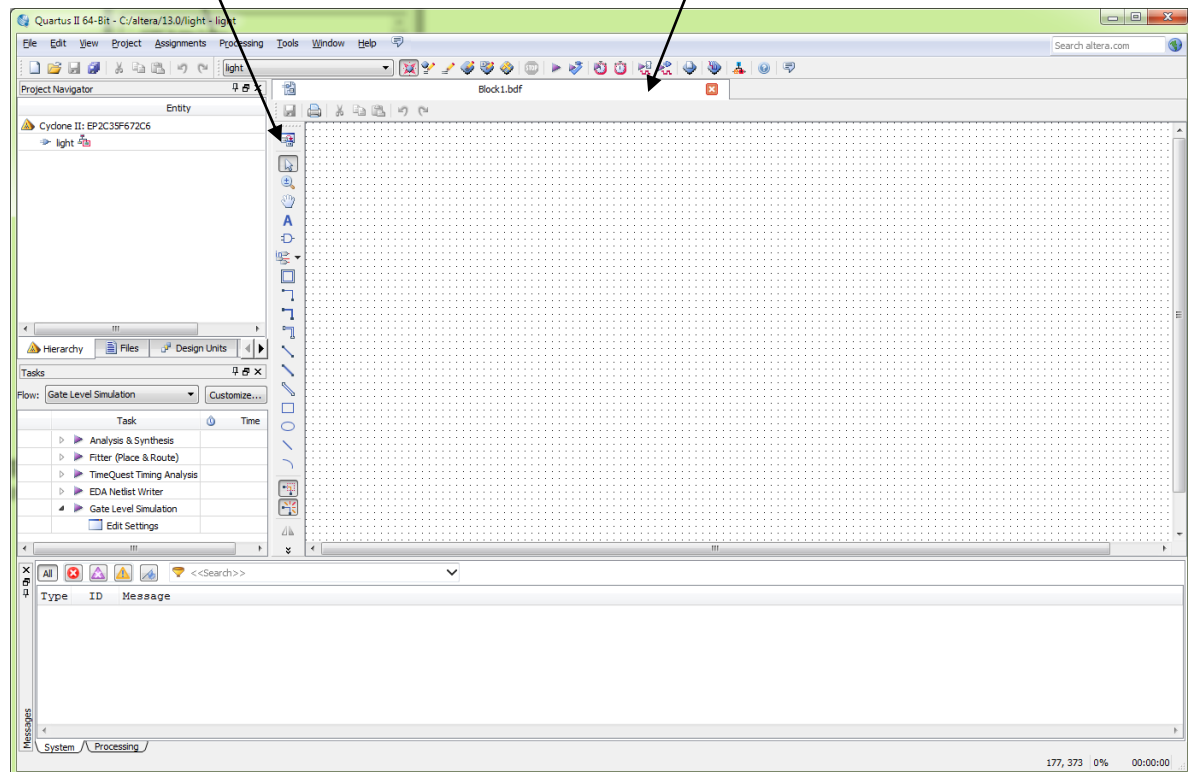


Figure 11. The light controller circuit.

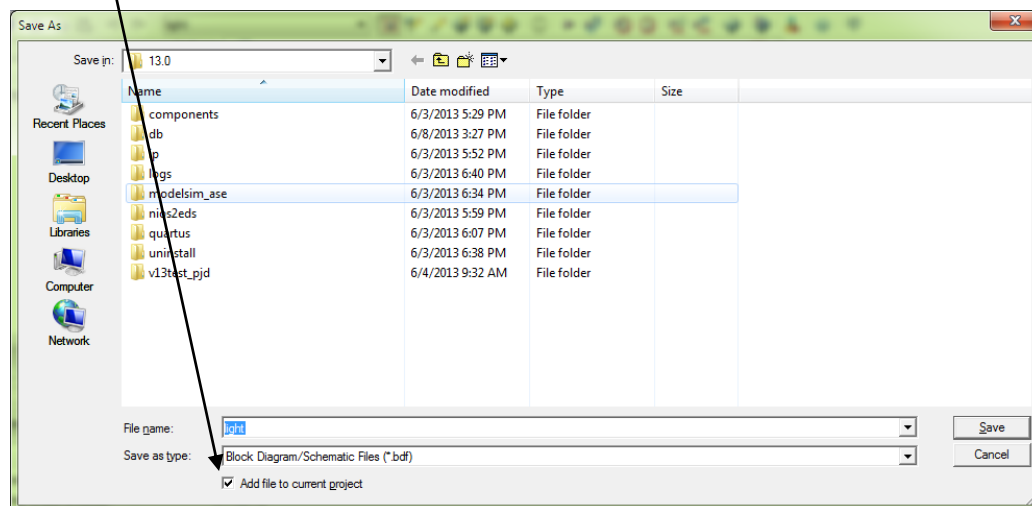
The Quartus II Graphic Editor can be used to specify a circuit in the form of a block diagram. Select **File > New**, the window below appears, choose **Block Diagram/Schematic File** and then click **OK**.



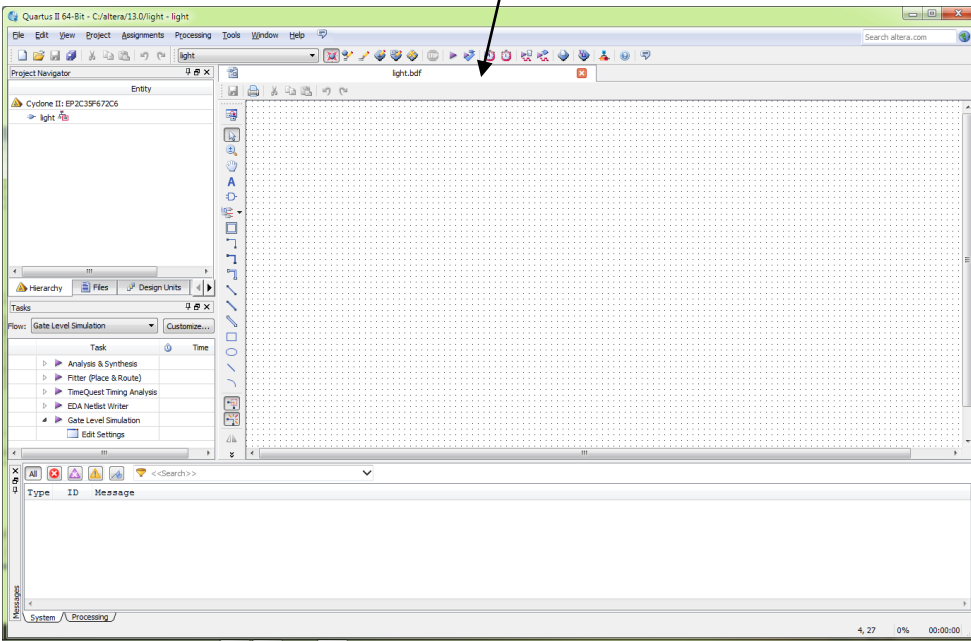
A new schematic diagram window is created with the default name **Block1.bdf**  
Note also the **tool bar buttons** that we can use to draw the circuit




The first step is to save the files with a more meaningful name. Select **File > Save As** to open the pop-up box depicted below. Enter the file name as **light**, (this name should match the name of the project we specified at the start. Note that the checkbox to add the file to the project should be ticked. Click **Save**

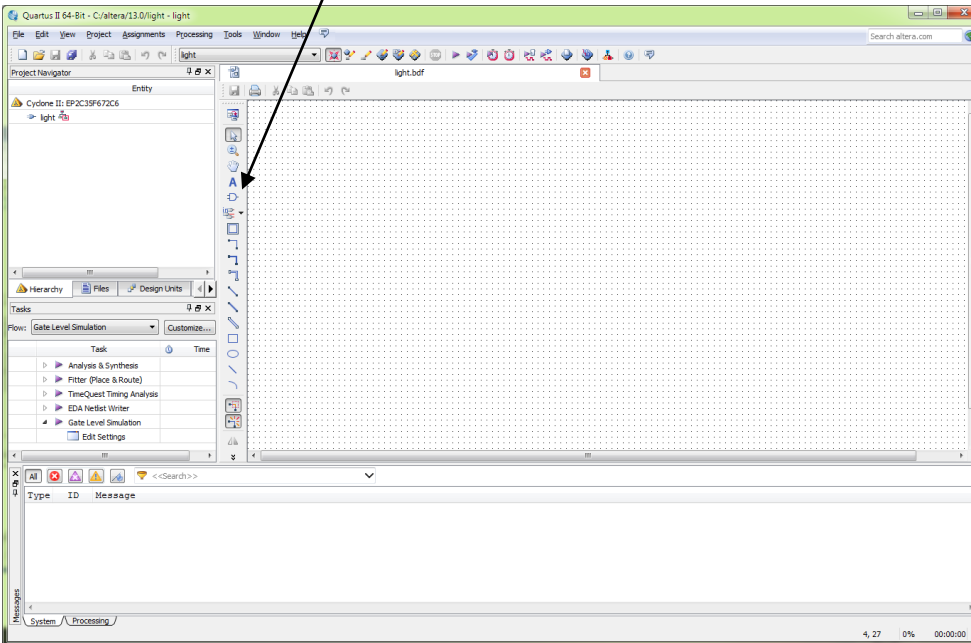


You will note the new file name in the main window below

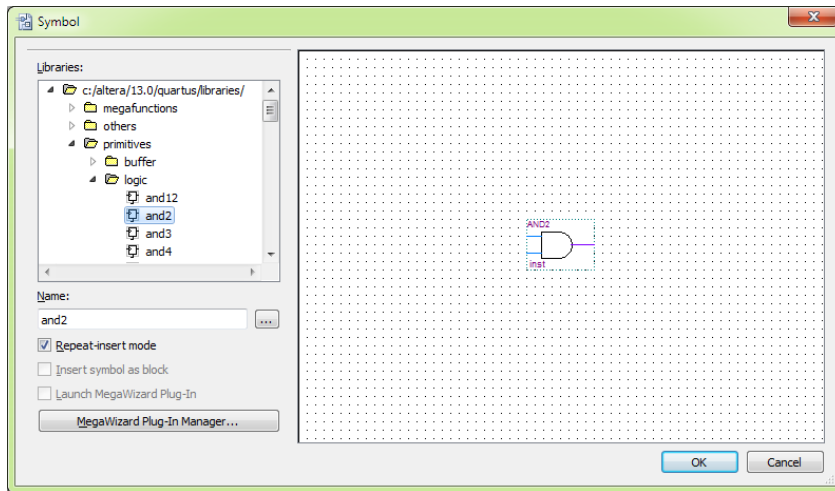


### 3.1 Importing Logic-Gate Symbols


The Graphic Editor provides a number of libraries which include circuit elements that can be imported into a schematic. Click on the icon  in the toolbar that looks like an AND gate.

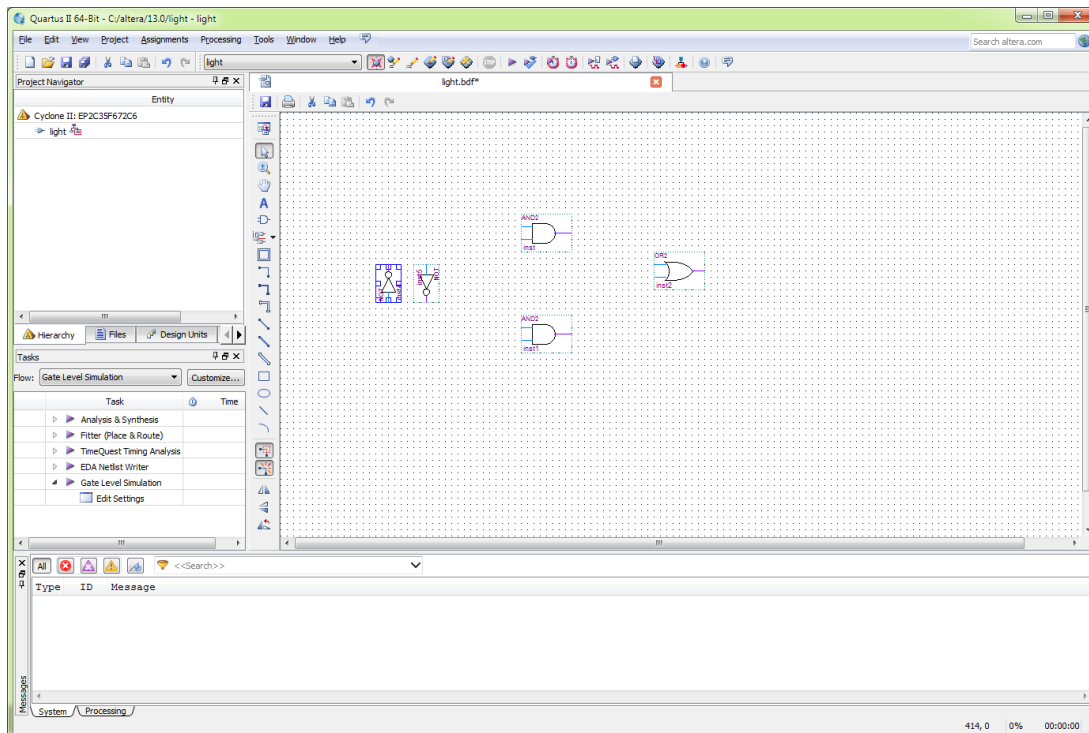


A pop-up box like the one shown below Expand the hierarchy in the Libraries box. First expand *libraries*, then expand the *primitives*, followed by *logic* and select *and2*, which is a two-input AND gate, and click **OK**



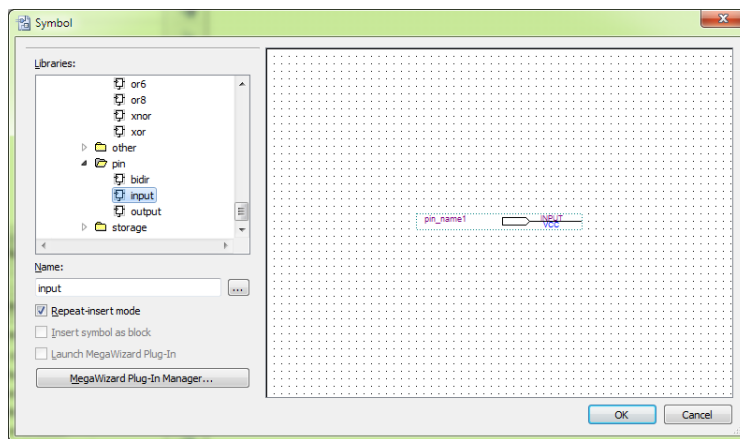
The above window will close and you can place the AND gate onto your schematic by click anywhere in the schematic diagram window. Click a **second time** to place a second **AND** gate symbol on your schematic diagram. Press the **ESC** key when you want to stop inserting gates. A symbol in the Graphic Editor window can be moved by clicking on it and dragging it to a new location with the mouse button pressed.

Using the same process as we used to select and place an AND gate, you can now select *or2* from the library and import a single **OR** gate into the diagram. Finally, select *not* and import **two** instances of the **NOT** gate. Rotate the NOT gates into proper position by using the right mouse button to click on the gate and then selecting “Rotate left 90” icon . Arrange the gates as shown below.

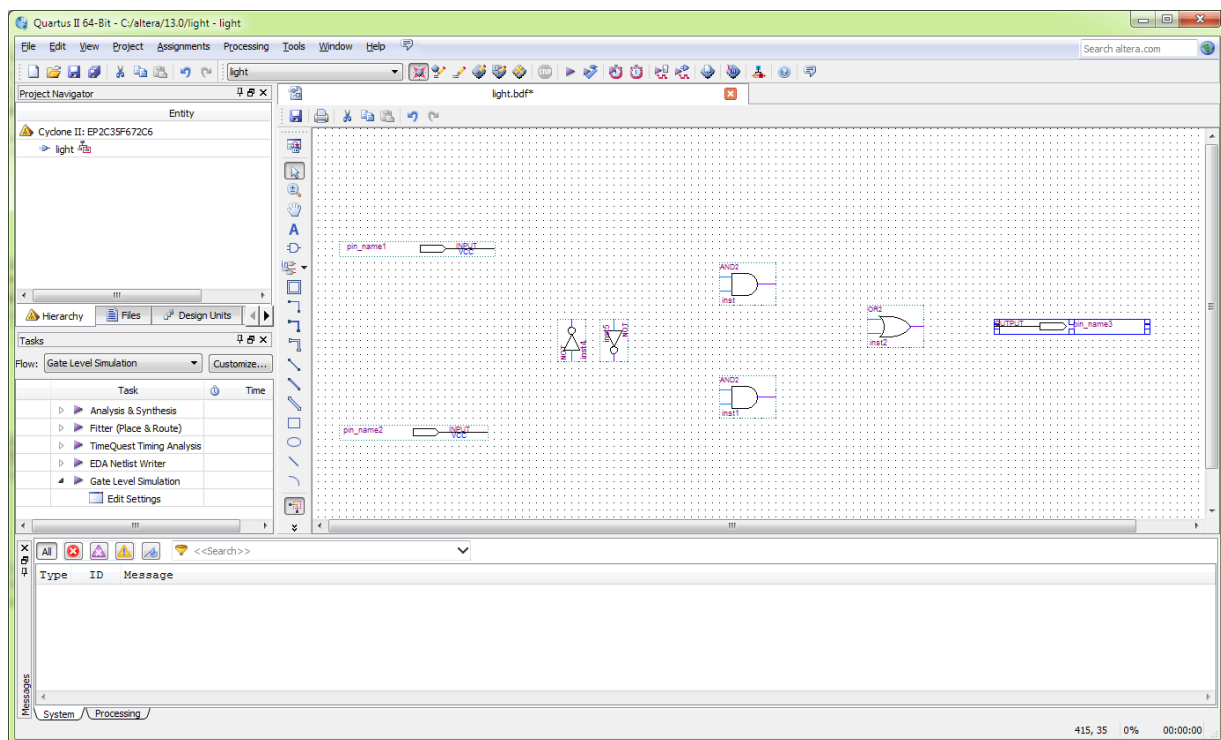


### 3.2 Importing Symbols for our Circuit Inputs and Outputs

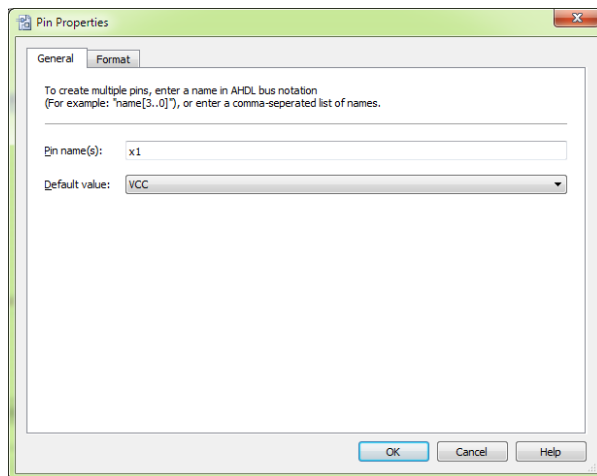
Having entered the logic-gate symbols, it is now necessary to enter the symbols that represent the **input and output ports** of the circuit. Use the same procedure as for importing the gates above, but choose the port symbols from the library *primitives/pin*.



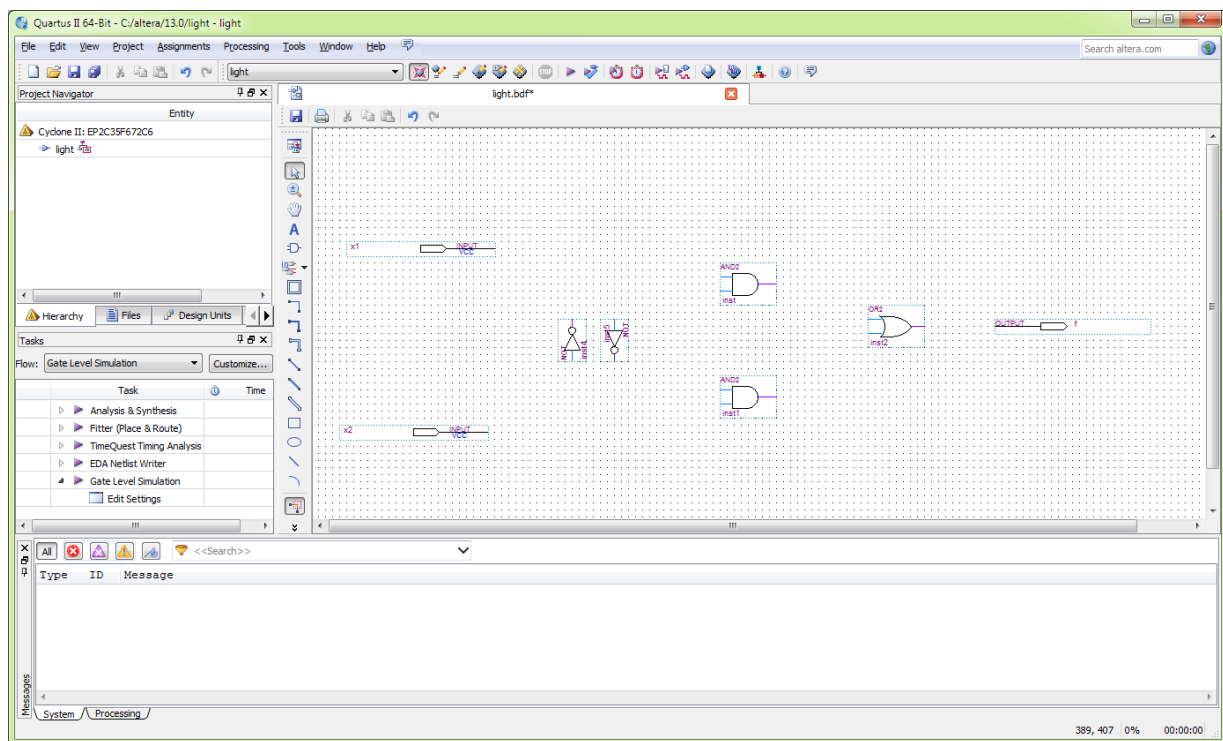
Place **two** instances of the **input pin** and **one** instance of the **output pin**, to obtain the image shown below. Don't worry about the **names** given to the pins, we will change those in a moment. Using the mouse, **Drag** and **position** the logic gates and input/output pins so that they look similar to that shown below



Assign names to the input and output symbols by **double-clicking** on them, one at a time, with the mouse. The dialog box below will appear. Type the pin name, **x1**, and click **OK**. Similarly, assign the name **x2** to the other input pin. Finally name the output pin to be **f**



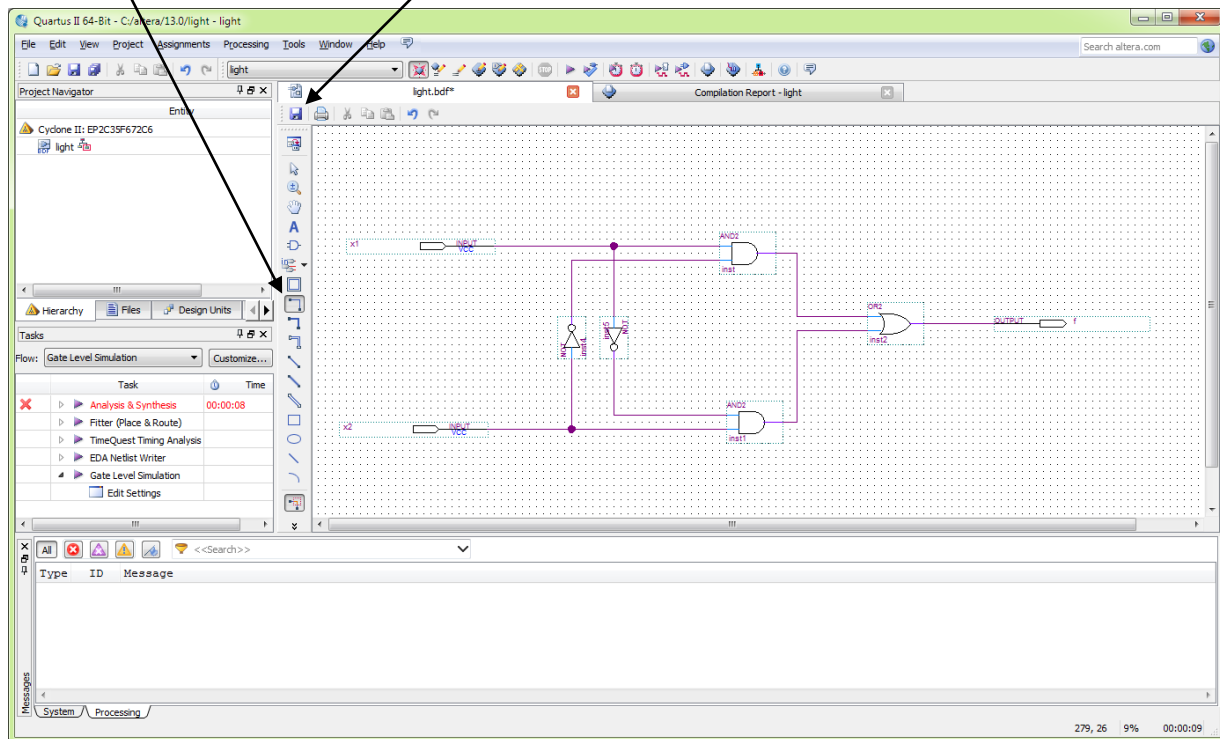
Your circuit will now look like this with the names for inputs and outputs correctly shown.




### 3.3 Connecting Nodes with Wires


The symbols in the diagram have to be connected by drawing lines (*representing electrical wires*). Click on the **Orthogonal Node Tool** icon in the toolbar. Click on either an **input** or an **output** to a **logic gate** or **pin**, and with the mouse button **held down** drag a wire to make the connection you require. Complete the circuit as shown below.

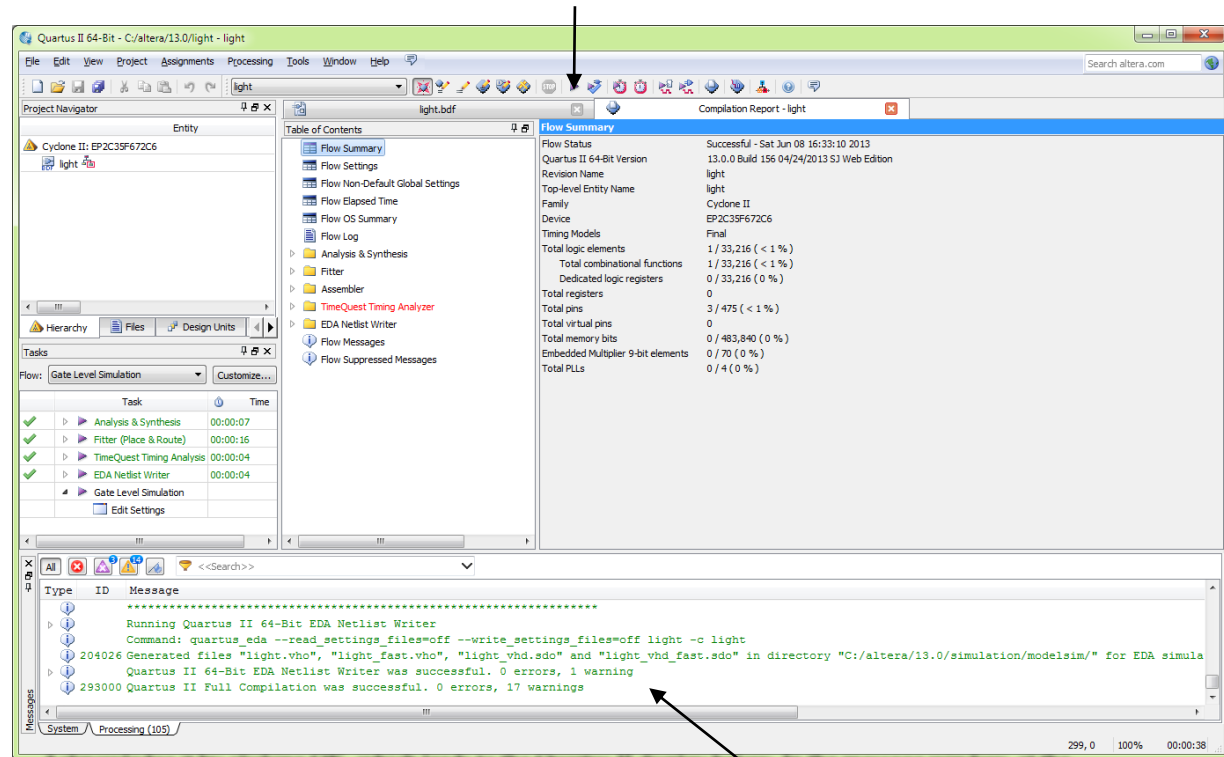
Try dragging a logic gate or pin around, you will see that the wires move with it, Once you are happy with the appearance of the circuit, now save it by clicking on the **save file** button.



Upon completing the diagram, click on the tool icon , to activate the **Selection Tool**.

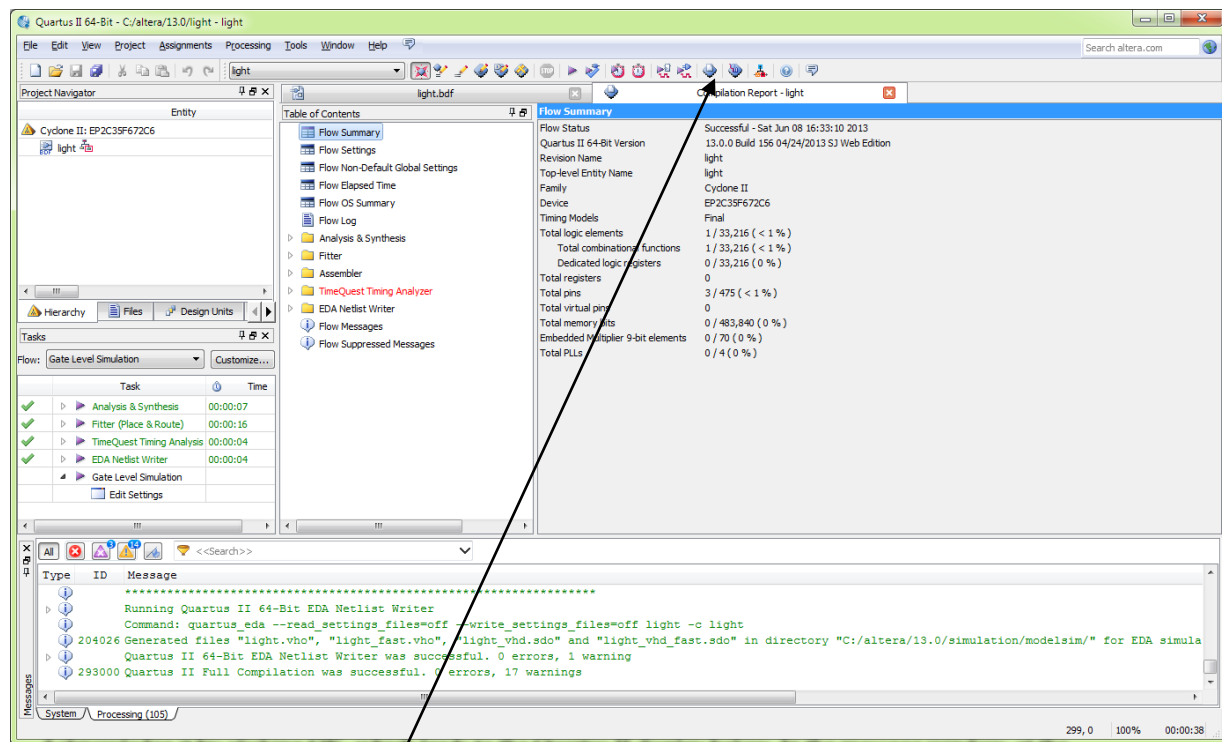
## 4 Compiling the Designed Circuit


The entered schematic diagram file, *light.bdf*, is processed by several Quartus II tools that analyze the file, synthesize the circuit, and generate an implementation of it for the FPGA that we can download to on the DE2 board. These tools are controlled by the application program called the *Compiler*. Run the Compiler by selecting menu **Processing > Start Compilation**, or by clicking on the toolbar icon . (See below)




As the compilation moves through various stages, its progress is reported in a window on the left side of the Quartus II display. Successful (or unsuccessful) compilation is indicated in the message window. **Warnings are OK** and can be ignored for the moment but errors need to be corrected.

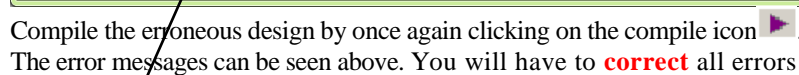




When the compilation is finished, a compiler flow summary report is produced (see above). It can be opened at any time by clicking on the icon . The report includes a number of sections listed on the left side of its window. The Flow Summary above indicates that only one logic element (not to be confused with logic gate) and three pins are needed to implement this tiny circuit on the selected FPGA chip.

Quartus II software displays messages produced during compilation in the Messages window. If the block diagram design file is correct, one of the messages will state that the compilation was successful and that there are no errors.

To see the effect of an error, remove the wire connecting the output of the top **AND** gate to the **OR** gate. To do this, click on the icon , right mouse the wire to be removed (to select it) and select **Delete**.



The error messages can be seen above. You will have to **correct** all errors and then recompile.

## 5 Physical Pin Assignment

During the compilation above, the Quartus II Compiler was free to choose any physical electrical pins on the selected FPGA to serve as inputs and outputs to/from our design. However, the DE2 board has hardwired connections made on the circuit board between certain pins FPGA on the FPGA and other components on the board such as the LEDs, switches, 7 segment displays etc, so we have to assign or map the signals that we created in our circuit above (i.e. the pins labelled  $x_1$ ,  $x_2$ ,  $f$ ) to the real/physical pins on the FPGA that are connected to the LEDs and Switches.

We will use two toggle switches on the DE2 board, labeled  $SW_0$  and  $SW_1$ , to provide the external inputs,  $x_1$  and  $x_2$  in our example circuit. These switches are connected to the DE2 FPGA as follows:

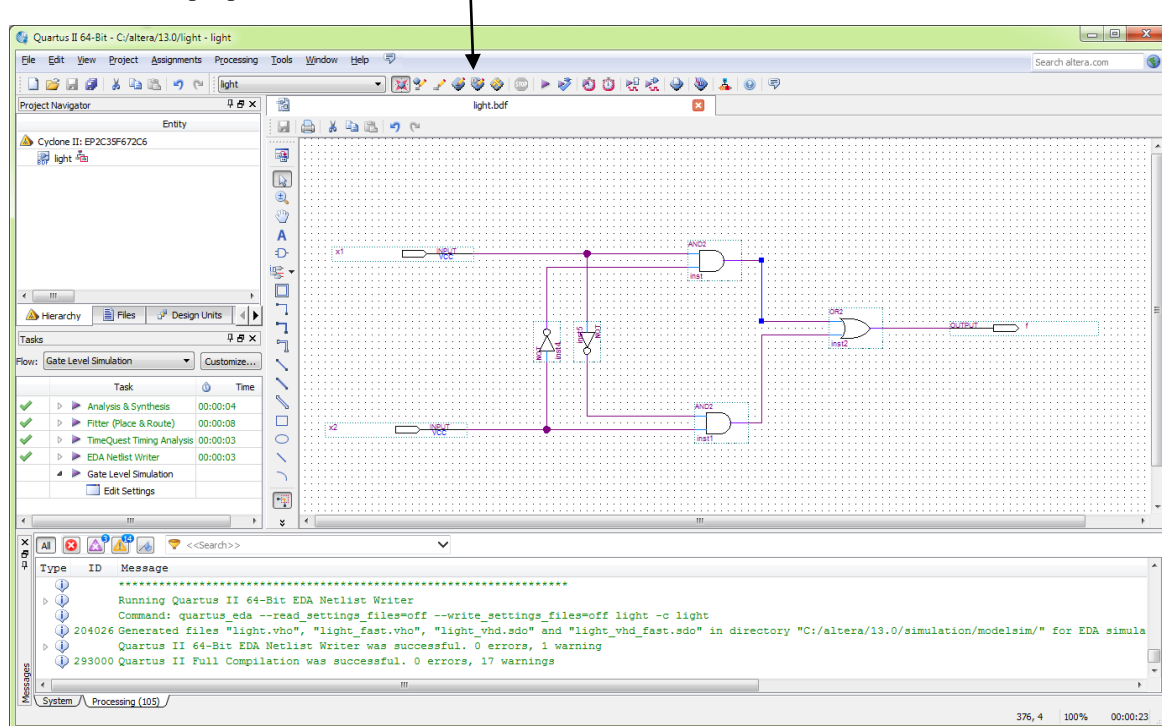
- $x_1$  will connect to  $SW_0$  which is physically connected to  $PIN\_N25$
- $x_2$  will connect to  $SW_1$  which is physically connected to  $PIN\_N26$

A summary of the pins on the FPGA and where they connect to on the DE2 board is contained in the DE2 user manual and also is available in summary for on the course web site

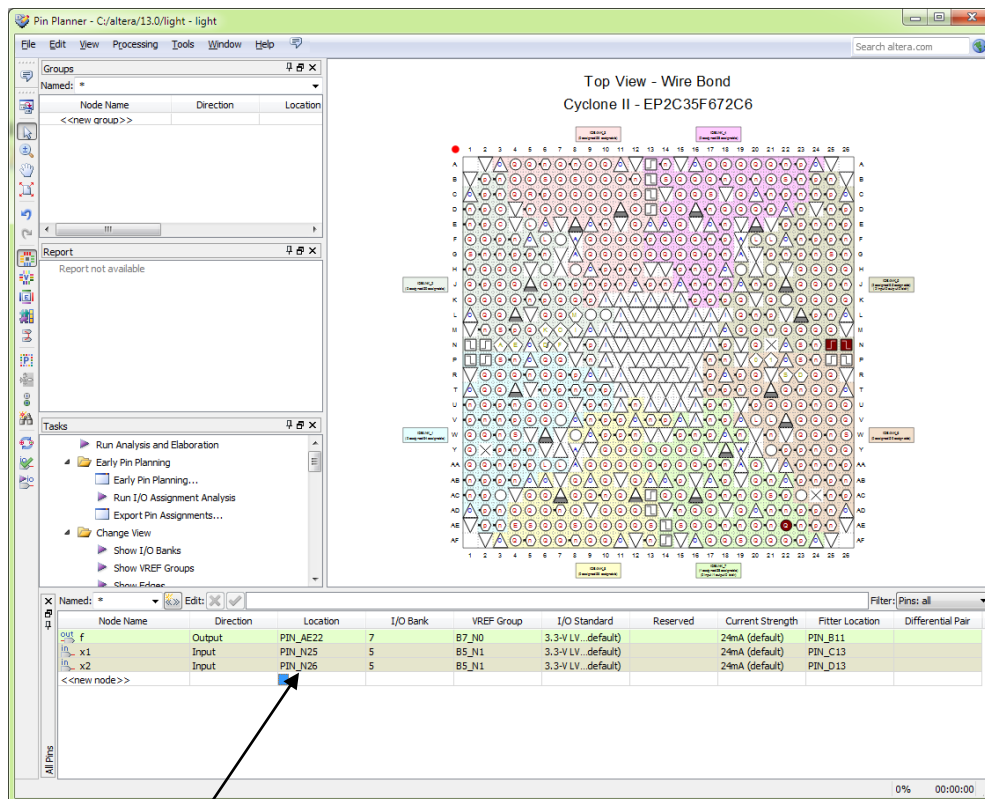
- $f$  will connect to the green light-emitting diode labeled  $LEDG_0$  which is physically connected to  $PIN\_AE22$ .

To map the pins shown on our schematic circuit diagram to the physical pins on the FPGA that are hard wired to the switches and LEDs, make sure your project has compiled without errors.

Now click on the pin planner button show below



The following window appears which shows the pins present on the underside of the Cyclone II chip that we selected when we started this project. At the **bottom** is a window that should be listing all the inputs and outputs from our circuit.



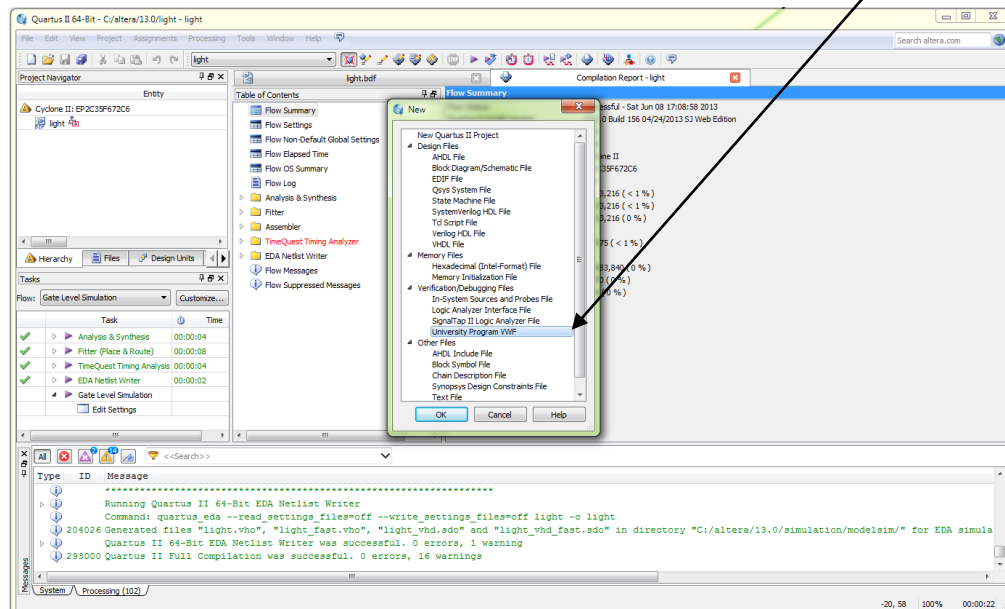
In the 'Location' cell to the right of each pin name, enter the FPGA pin that we want each signal to connect to on the FPGA. e.g. for output 'f' above, type in the name **AE22** (note you don't have to type in the full **PIN\_AE22** although you can if you want). Do the same for the other inputs **x1** and **x2** as shown above

Now you have made the pin assignments, **close** the Pin Planner which will automatically save your pin assignments and **\*\*\*\*\*IMPORTANTLY\*\*\*\*\* Recompile the circuit**, so that it will be compiled with the correct pin assignments.

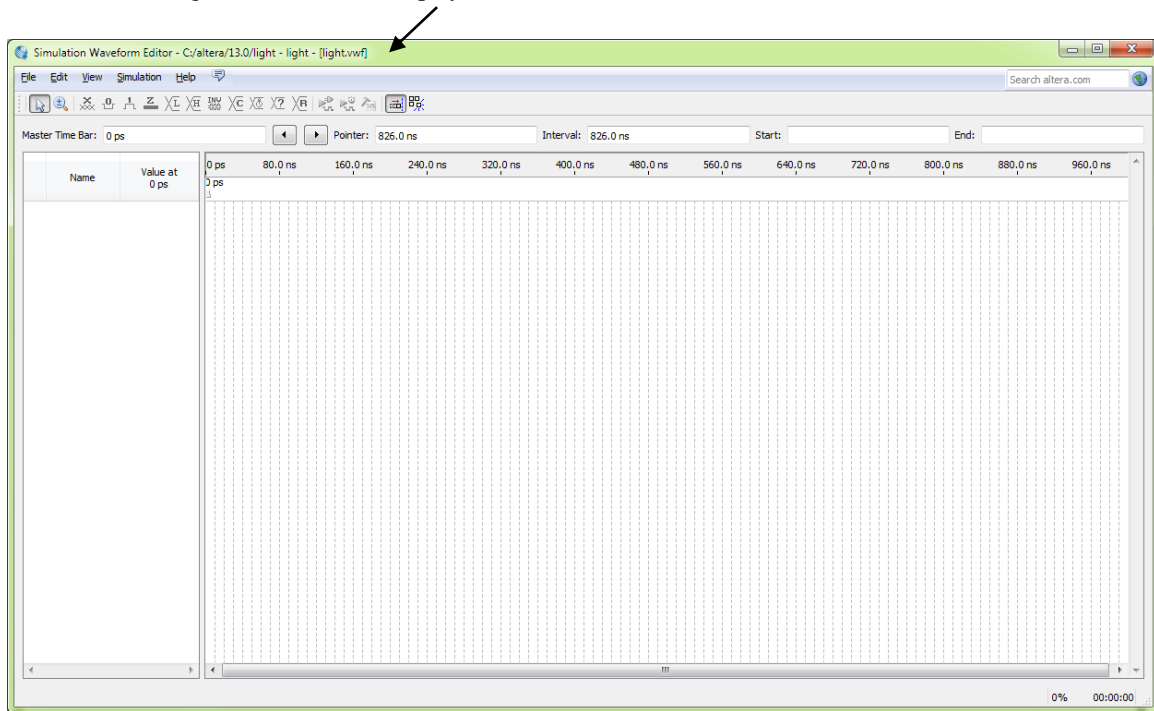
## 6 Simulating the Designed Circuit

Before downloading the compiled circuit in the FPGA chip on the DE2 board, it is prudent to simulate it to verify its correctness. Quartus II 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 our 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 II Waveform Editor to draw the test vectors, as follows:

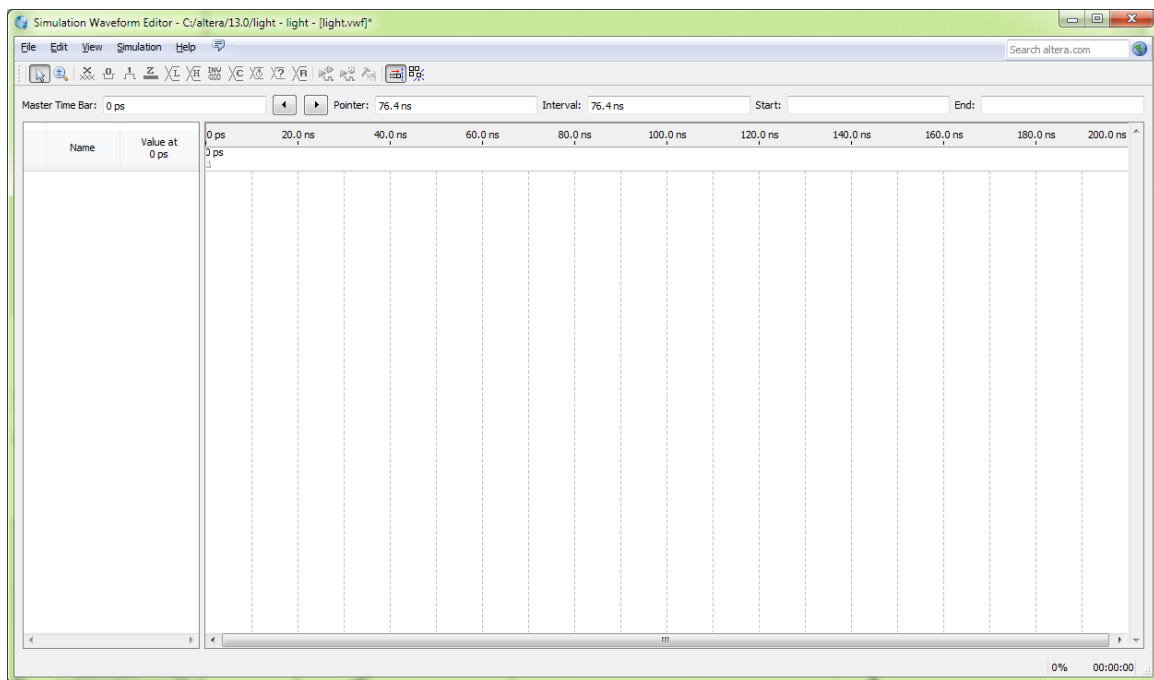
1. Open the Waveform Editor window by selecting **File > New** and select **University Program VWF** (short for **Vector Waveform File**)



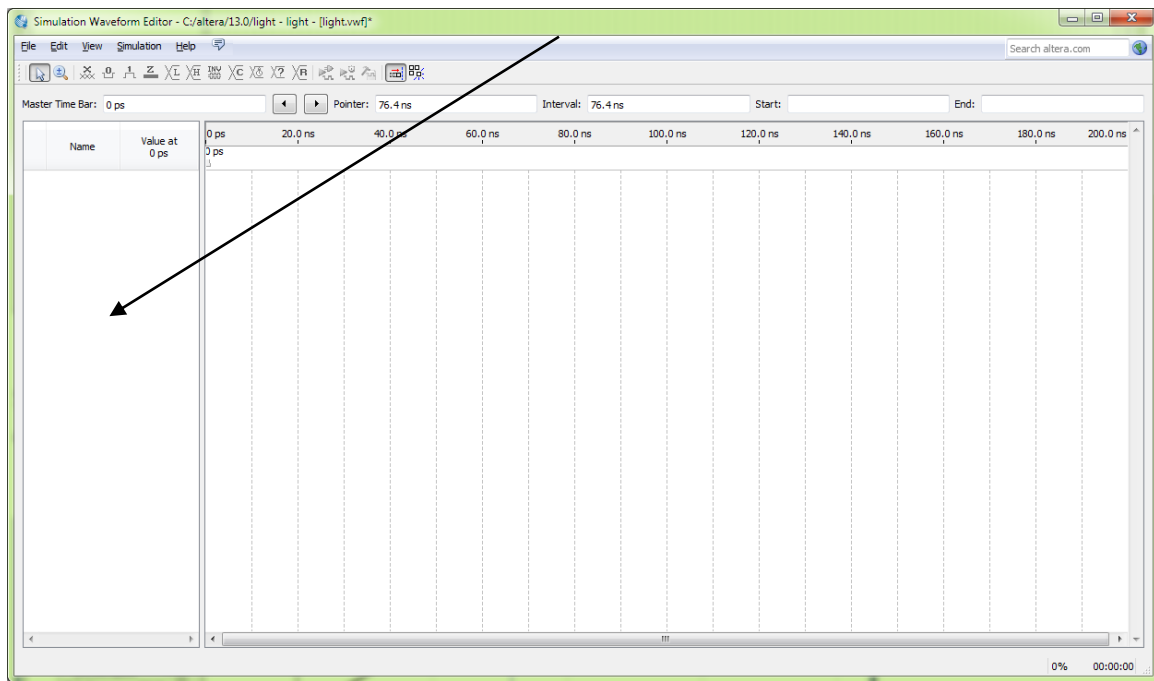
The Waveform Editor window is opened as shown in the Figure below. Save the file under the name *light.vwf*; note that this changes the name in the displayed window.



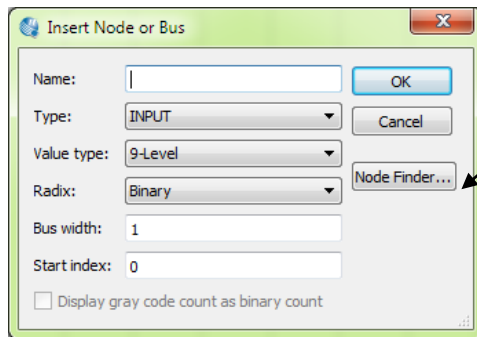
Set the desired simulation time to run from 0 to 200 ns by selecting menu **Edit > Set End Time** and entering 200 ns in the dialog box that pops up. The simulation window will now look like this



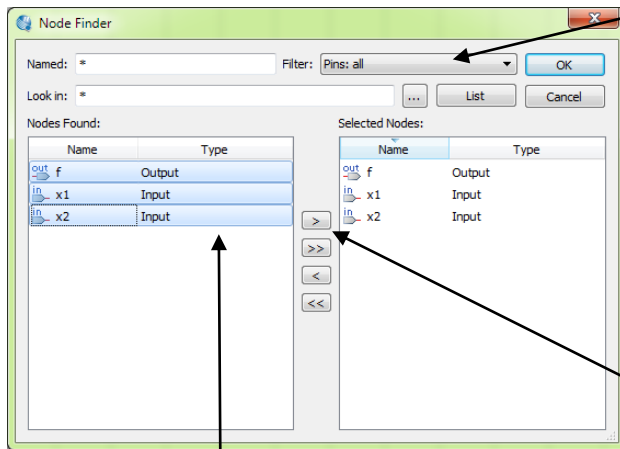
Next, we want to include the **input** and **output signals** (or **nodes**), (i.e. the signals '**x1**', '**x2**' and '**f**') from our circuit to be simulated.



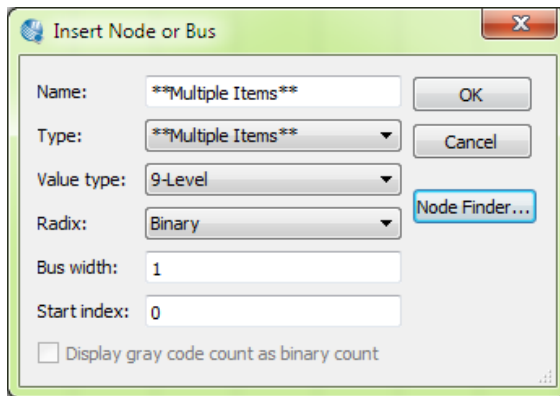
The following window pops up. Click on the button labeled **Node Finder**.



This will open the **Node Finder** utility. The Node Finder utility has a filter used to indicate what type of nodes are to be found. Since we are interested in input and output pins, set the filter to **Pins: all**. Click the **List** button to find the input and output nodes as indicated on the left side of the figure.



Click on each of the 3 signals *f*, *x1*, *x2* in the Nodes Found box and then click the '>' symbol sign to add it to the **Selected Nodes** box on the right side of the figure as shown below. Then click **OK**. When this dialog box appears, you can display click **OK**

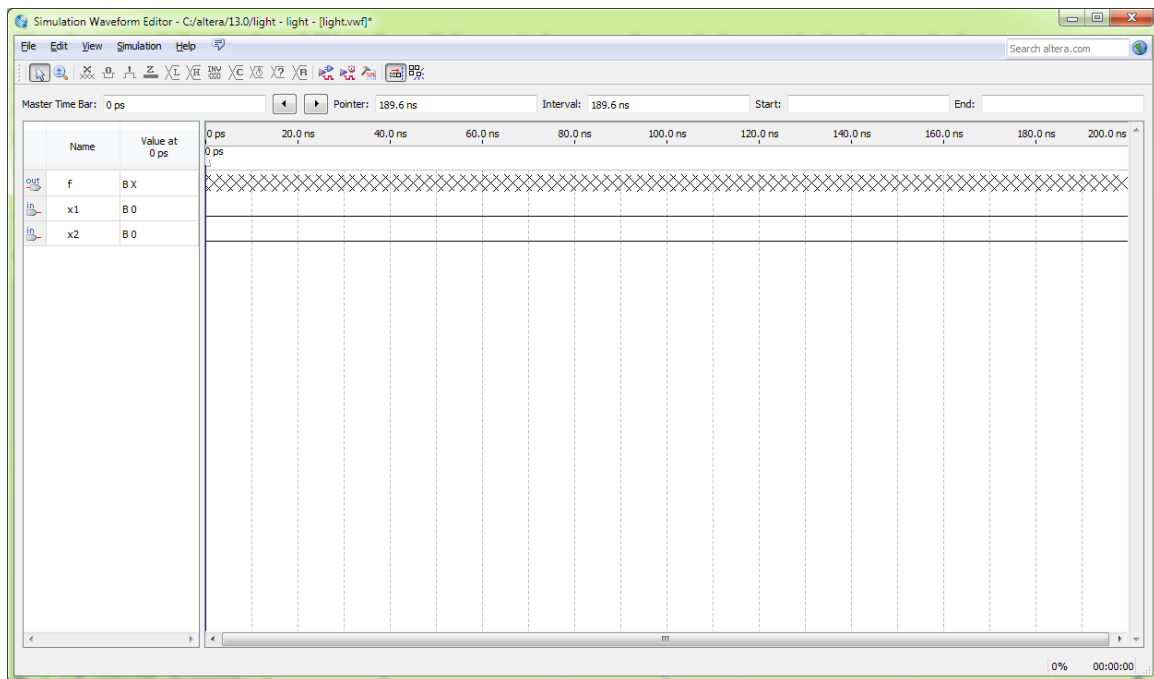




This leaves a fully displayed Waveform Editor window, as shown below with the 3 signals from our circuit. If you did not select the nodes in the same order as displayed in Figure 36, it is possible to rearrange them. To move a waveform up or down in the Waveform Editor window, click on the node name (in the Name column) and release the mouse button. The waveform is now highlighted to show the selection. Click again on the waveform and drag it up or down in the Waveform Editor.

To zoom **in** or **out** on the waveform file, select the **magnifying tool** and use the right or left mouse buttons to click on the waveform file

The cross hatching associated with the output signal 'f' means "unknown" because we have not yet run the simulator

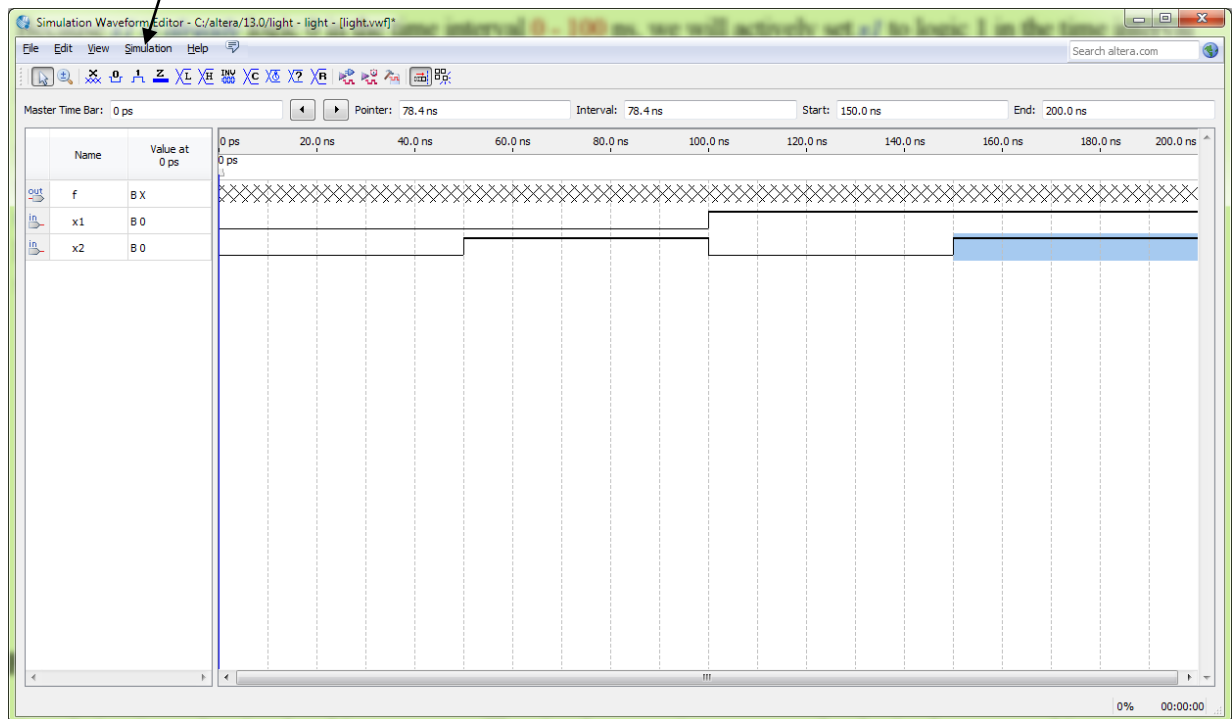


We will now specify the logic values to be used for the **input** signals *x1* and *x2* during simulation. The logic values at the **output** *f* will be generated **automatically** by the simulator (this is the purpose of simulation, to predict the simulated outputs for a set of inputs). To simulate the behavior of a large circuit, it is necessary to apply a sufficient number of input value combinations 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. However, for our tiny circuit we can simulate all four input values for a 2 input logic function, i.e. the input test patterns/values, **00**, **01**, **10**, **11**. We will use four **50-ns** time intervals to apply the four test patterns/values.

We can generate the desired input waveforms as follows. Click on the waveform name for the *x1* node. Once a waveform is selected, the toolbar buttons to the left of the waveform can be used to draw the desired waveforms. Commands are available for setting a selected signal to **0**, **1**, etc or defining a **clock** waveform.

Because *x1* is *already* logic 0 in the time interval **0 - 100 ns**, we will actively set *x1* to logic 1 in the time interval **100 - 200 ns**. Do this by pressing the mouse at the start of the time interval (100ns) on signal *x1* and dragging it to its end time of **200 ns**, which highlights the selected interval. Now press the **logic 1** button in the toolbar.



Having done this for *x1*, now make *x2* = 1 from **50 - 100 ns** and also from **150 - 200 ns**, which corresponds to the truth table. This should produce the image in the Figure above. Thus we have inputs **00** for time **0-50ns**, inputs **01** for time **50-100ns**, **10** for time **100-150ns** and **11** for time **150-200ns**.

Observe that the output *f* is still displayed as having an **unknown** value at this time, **Save the file**.

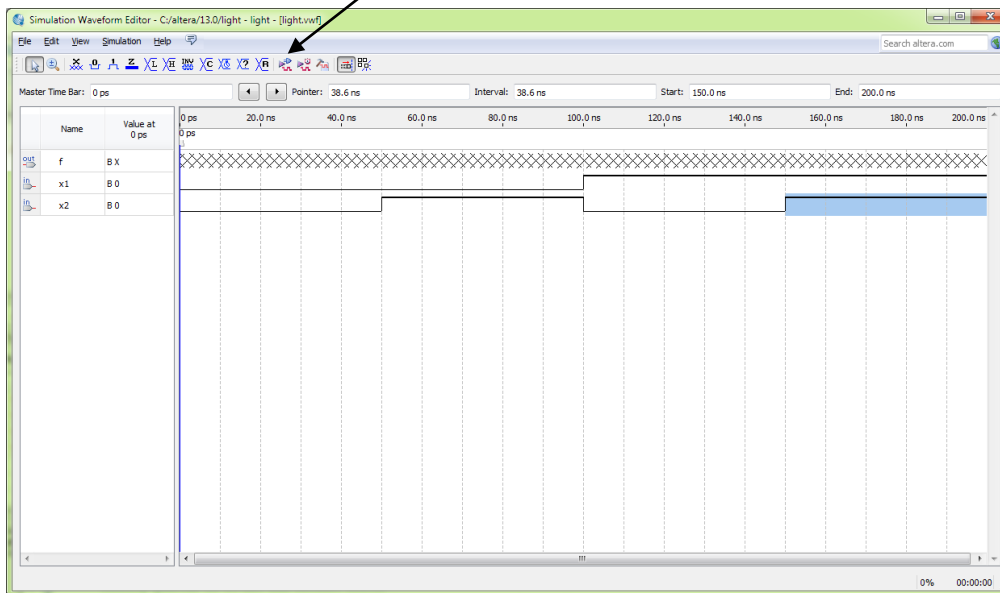
## 6.1 Performing the Simulation

A designed circuit can be simulated in two ways. The simplest way is to assume that logic elements and interconnection wires in the FPGA are **perfect**, and have **no time delay** in the propagation of signals through the circuit. This is called **functional simulation**.

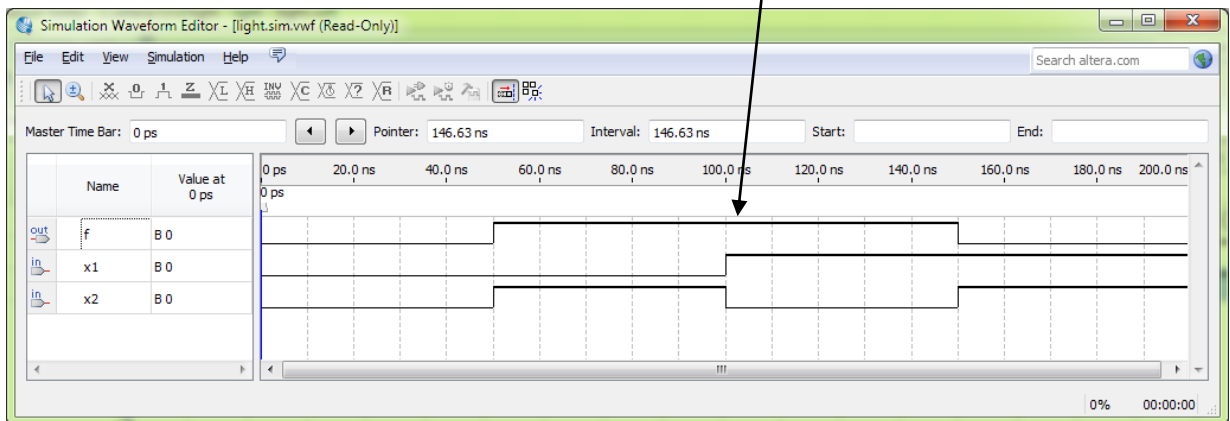
A more complex alternative is to take all propagation delays into account, which leads to **timing simulation**. Typically, **functional simulation** is used to verify the functional **correctness** of a circuit as it is being designed, i.e. it gives the correct 0's and 1's at the output for the corresponding 0's and 1's at the inputs, without regard to actual timing. This kind of simulation is faster so let's do that first.

### 6.1.1 Functional Simulation

To perform the **functional simulation**, click on the button below of click menu **Simulation->Run Functional Simulation**.

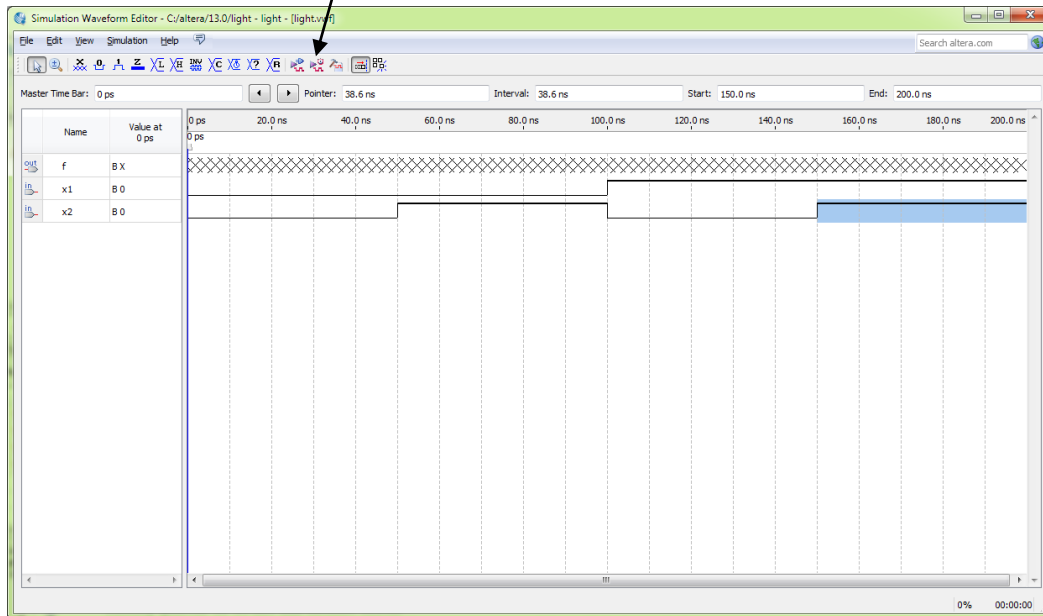


A new output simulator window opens as shown below. Notice how the output 'f' has been predicted by the simulator.

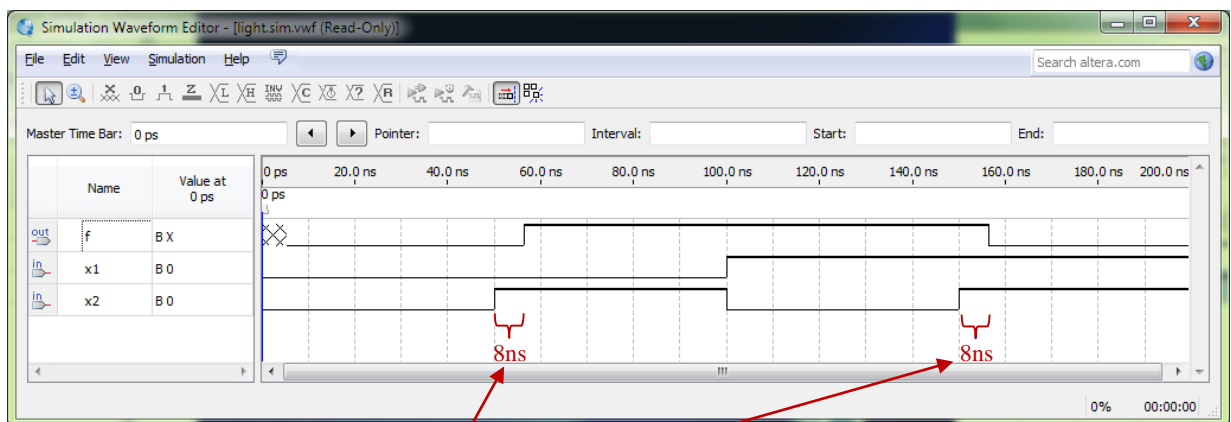


### 6.1.2 Timing Simulation

Having ascertained that the designed circuit is **functionally** correct, we should now perform a timing simulation to see how it will behave when it is actually implemented in the chosen FPGA device. This time, chose a Timing simulation by clicking on the button below or click menu **Simulation->Run Timing Simulation**



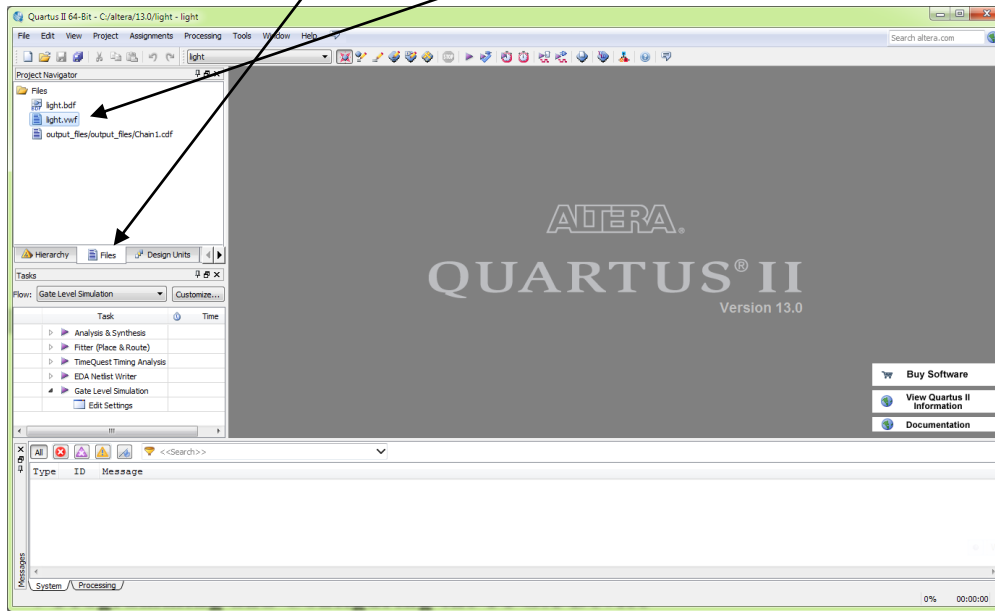
Notice how a similar simulation window is produced but it's not exactly the same as the functional version.



Observe that there is a **delay of about 8 ns** in producing a change in the output signal *f* after the input signals, *x1* or *x2*, change their values. This delay is due to the propagation delays in the **logic elements** and the **wires** in the FPGA device. This is a real simulation as it models the actual time delays associated with output signals when we download the design to the FPGA.

### 6.1.3 Re-Opening the Simulation waveform once it has been closed

Easiest way is to click on the files tab and select the light.vwf file as shown below (or you can search for it using the menu **File->Open**)



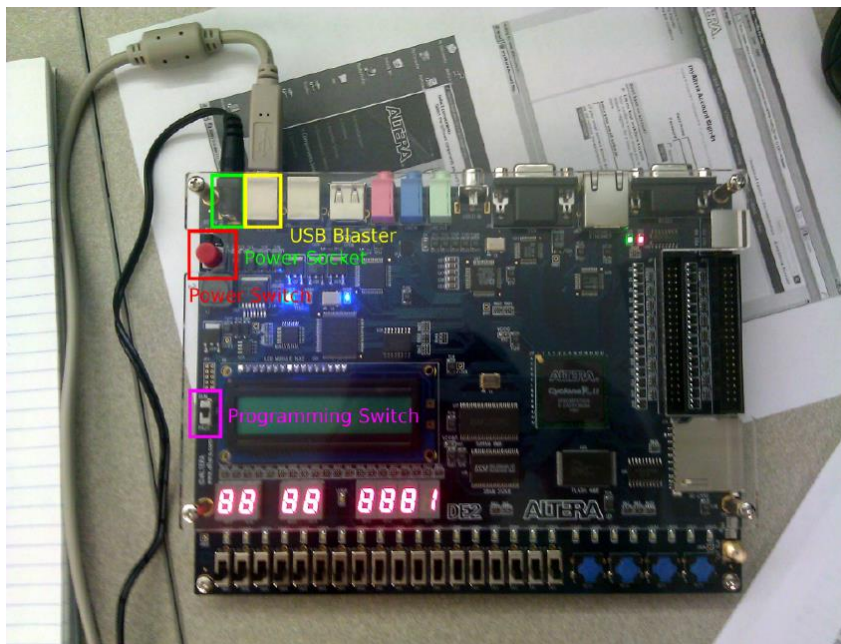
## 7 Programming and Configuring the FPGA Device

The FPGA device can be programmed and configured to implement the designed circuit above. The required programming can be done in two different ways, known as **JTAG** and **AS** modes. The choice between the two modes of download is made by the **RUN/PROG** programming switch on the DE2 board beneath the red power button on the L.H.S of the board. The **RUN** position (UP) selects the **JTAG** mode, while the **PROG** position (DOWN) selects the **AS** mode.

The programmed data for the FPGA is transferred from your PC to the board by means of a USB cable that connects a USB port on your PC 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, plug the DE2 power transformer into a wall outlet and into the DE2 board itself and turn on the power supply switch on the board (the **RED** power push button).

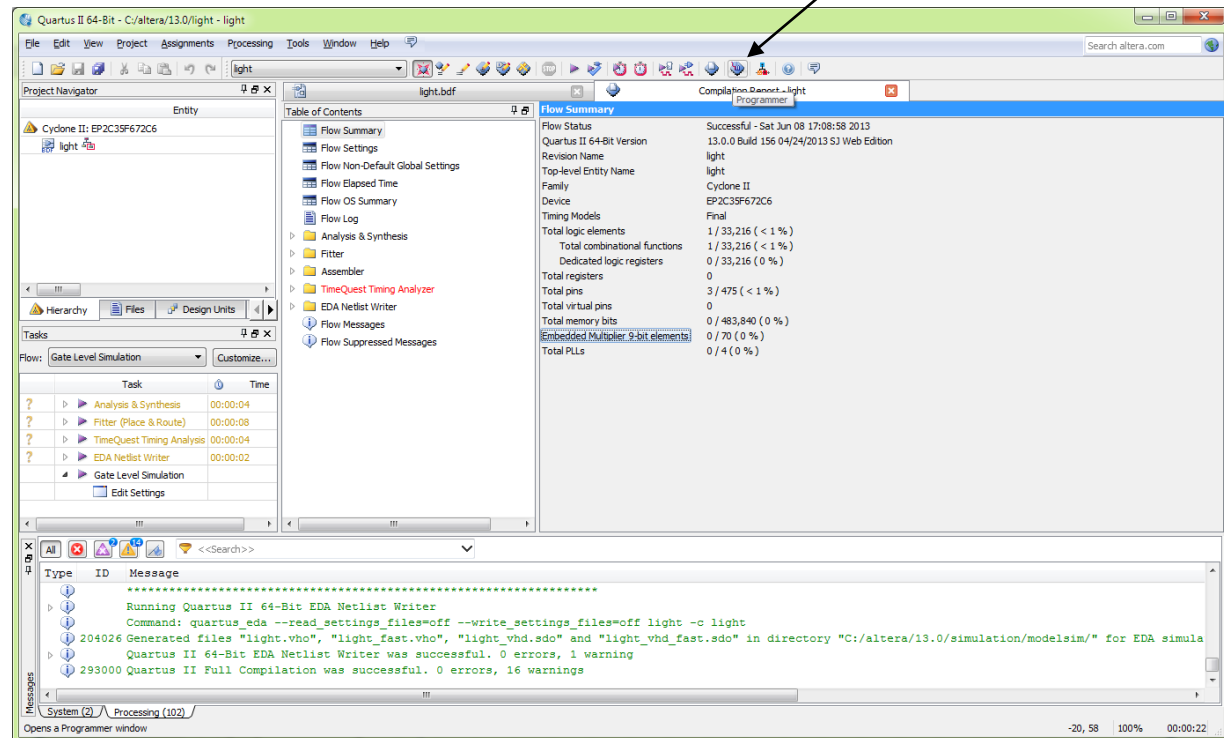
In **JTAG** mode, (Programming switch in the UP position) the configuration data is programmed directly into the FPGA device. The acronym JTAG stands for Joint Test Action Group. This group defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. If the FPGA is configured in this manner, it will retain its programming as long as the **power remains turned on**. The programming is lost when the power is turned off.

The second possibility is to use the **Active Serial (AS)** mode (Programming switch in the DOWN position). In this case, a memory chip on the DE2 board is used to store the programmed data for the FPGA. When power is applied to the DE2, the data from the memory chip is loaded automatically into the FPGA (which takes about ½ sec). In this way, you don't have to use Quartus to download the programmed data each time the DE2 board is turned on, thus the DE2 can be made to run “stand alone” without an attachment to your PC.



## 7.1 JTAG Programming

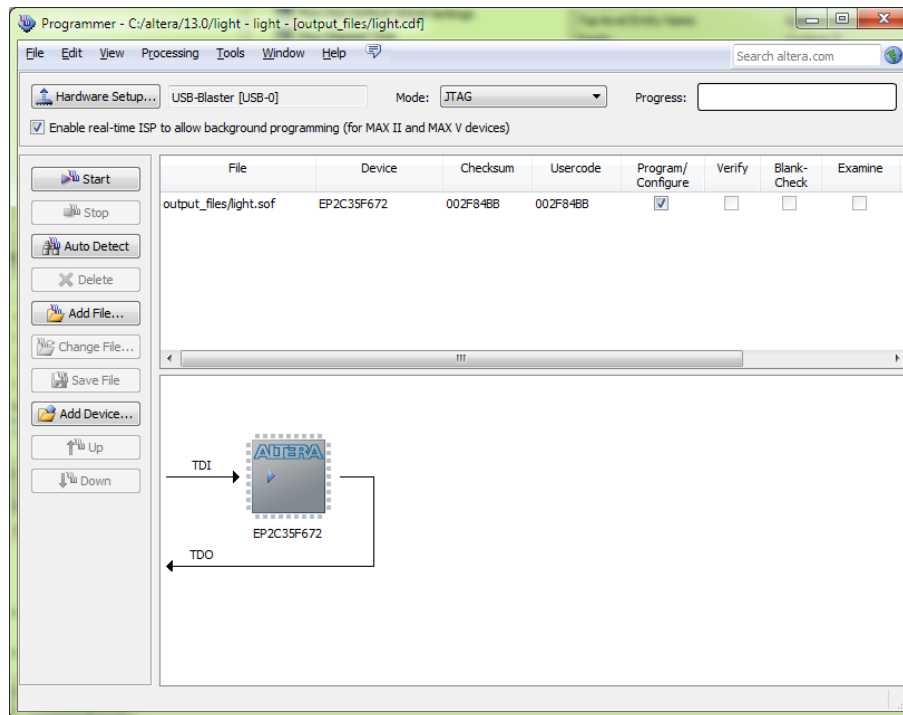
The programming and configuration task is performed as follows. Flip the **RUN/PROG** switch on the DE2 into the **RUN** position. Turn the DE2 board **off** then back **on** again. Click on the Programmer button below



The following dialog box pops up. If not already chosen by default, select **JTAG** in the Mode box. In addition if the USB-Blaster is not chosen by default, press the **Hardware Setup...** button and select the **USB-Blaster** in the window that pops up, as shown below (*you have to already have installed the USB blaster driver to do this*).

If you cannot get the USB blaster to appear,

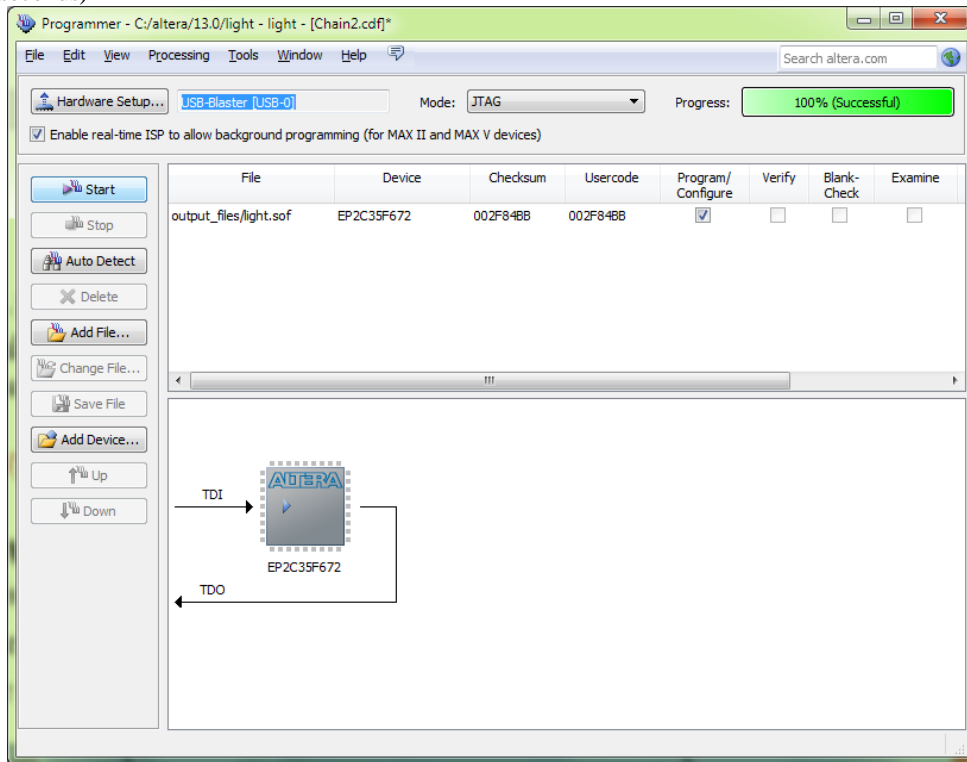
- Check that that your USB cable is connected from your Laptop to the USB blaster port on the DE2.
- Check that power is applied to the DE2 and it is turned on.
- Check your Laptop to make sure you have installed the Windows driver for the USB blaster (details are on the course web site).
- Cancel the programmer window. Now unplug the USB blaster cable from your laptop then plug it back in again. Start the programmer and see if you can select USB-Blaster from the Hardware setup button
- If none of these work, you have problems and you need to get help.



Observe that the configuration file *light.sof* is listed in the window. If the file is not already listed, then click the **Add File** button and select it. The extension *.sof* stands for SRAM Object File. Note also that the device selected is **EP2C35F672**, which is the FPGA device used on the DE2 board. Click on the **Program/Configure** check box to make sure it is selected.



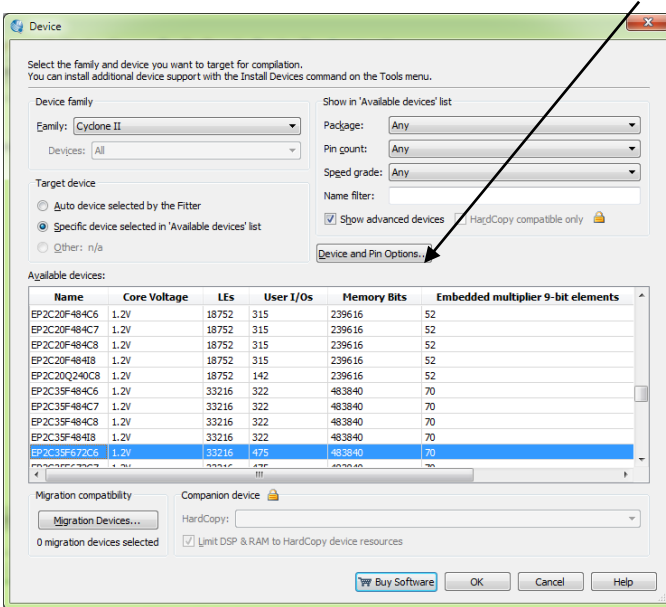
Now, press **Start** button in the window. The progress indicator will indicate when it done (it should only take a few seconds)



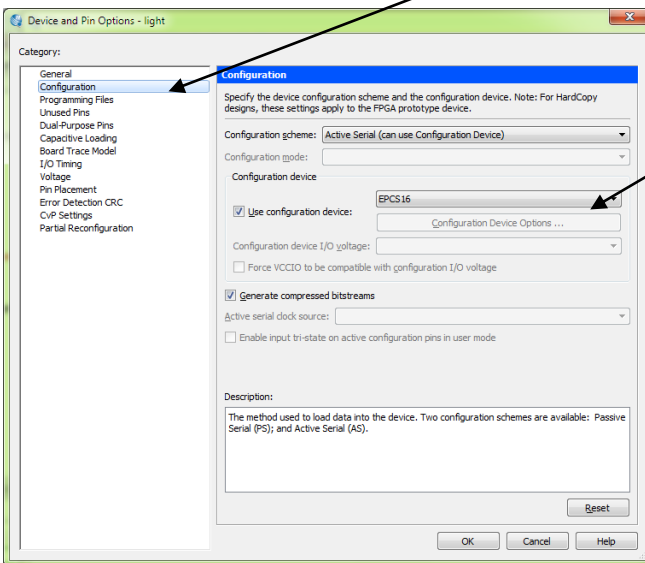
**(NOTE SKIP SECTION 7.2 for the MOMENT and goto section 8, you can return to it later)**

## 7.2 Active Serial Mode Programming

In this case, the configuration data has to be downloaded to a small memory chip on the DE2 board, which is identified by the name **EPCS16**. To specify the required configuration device, select Quartus main menu **Assignments > Device**, which leads to the window below. Click on the **Device and Pin Options** button.

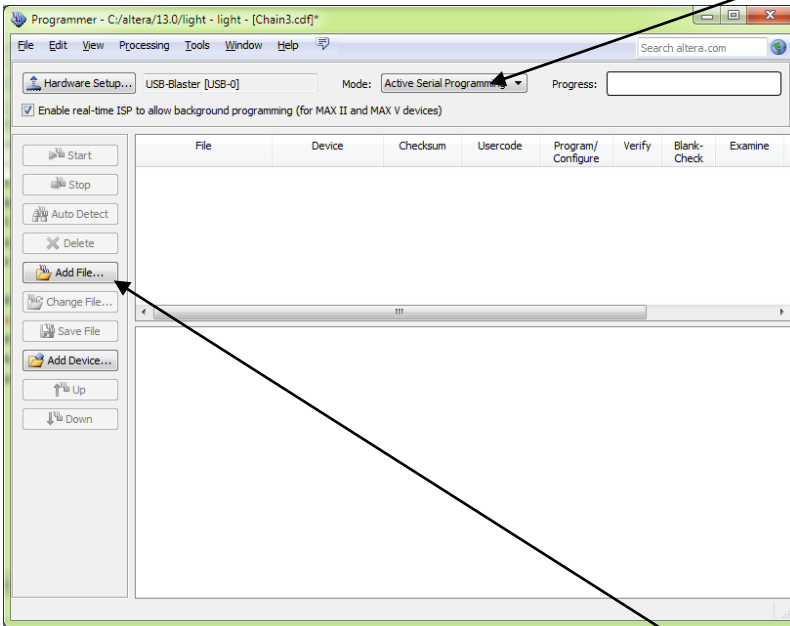


The following window appears. Click on the **Configuration** tab as shown below and select the **EPCS16** device

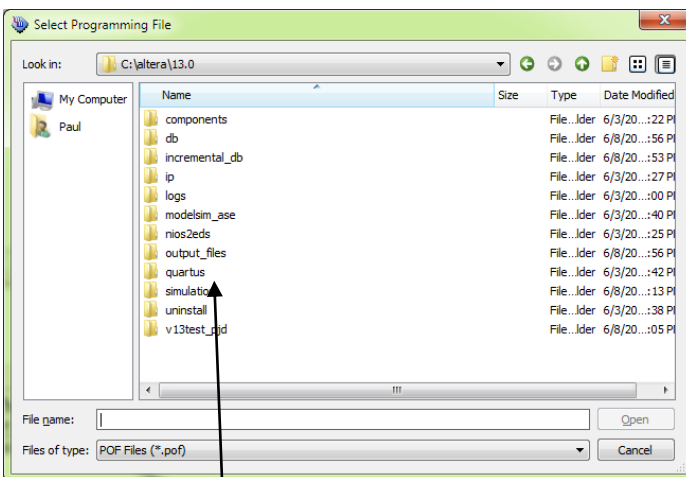


Now you **MUST RECOMPILE** the designed circuit. The rest of the procedure is similar to the one described above for the JTAG mode.

Select the **Programmer** tool to reach the window below. In the Mode box select **Active Serial Programming**.

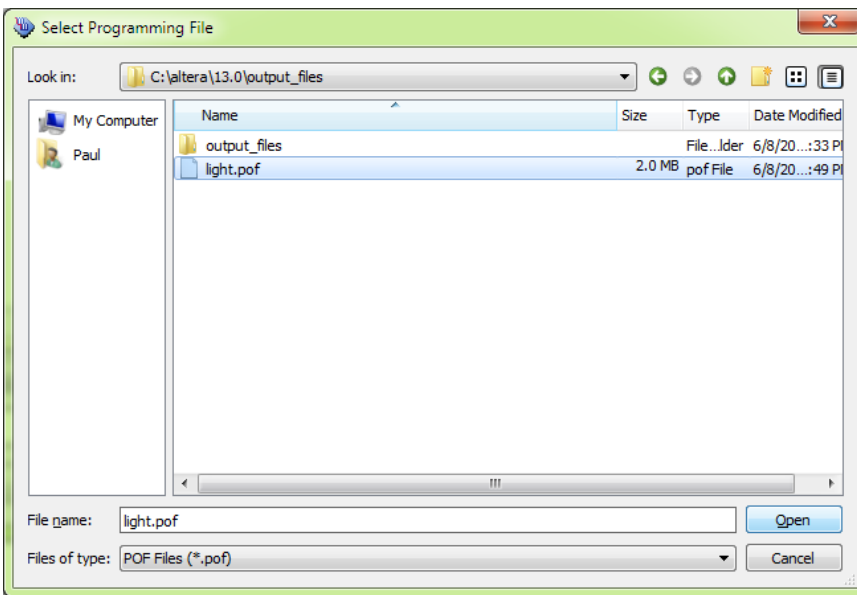


If the configuration (.pof file) is not already listed in the window, press **Add File**. The pop-shown below will appear.

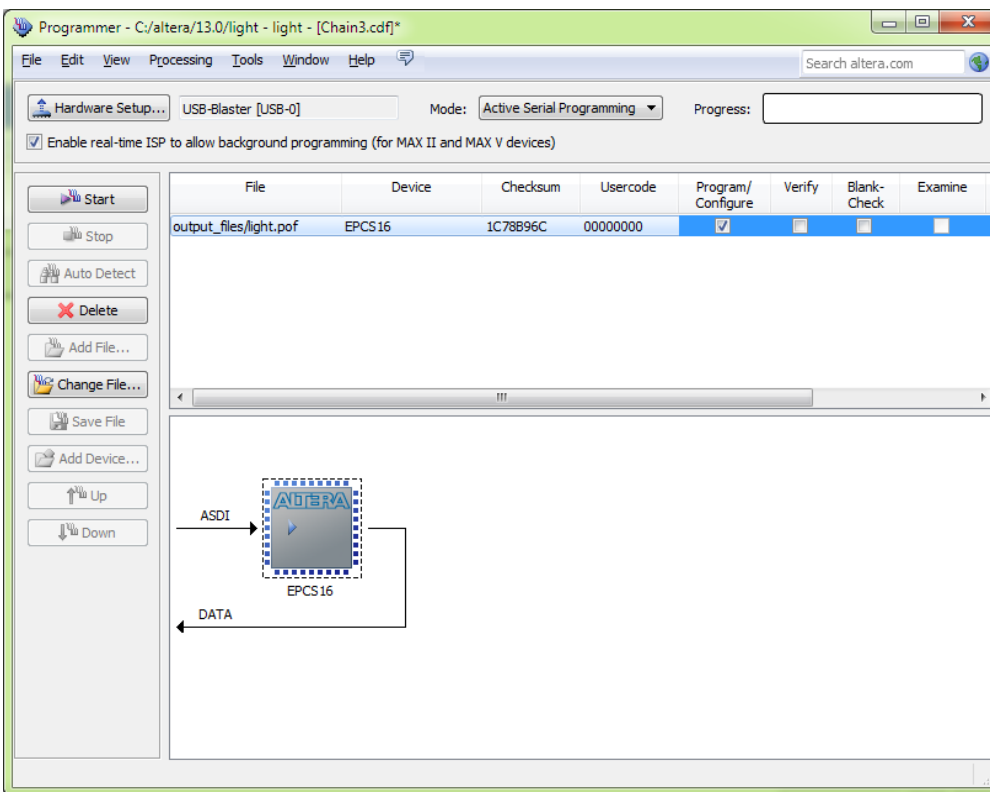


Select the **Output\_files** folder.

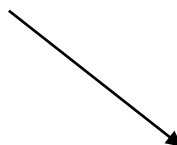
Then select light.pof from the window below and click **OPEN**



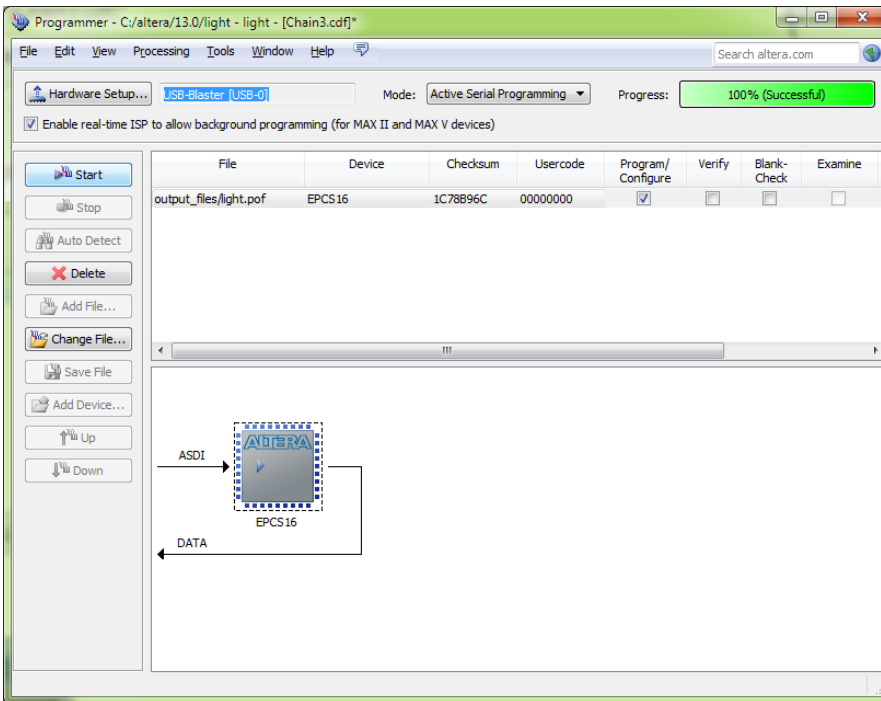
As a result, the configuration file *light.pof* (Programmer Object File) will be listed in the window. **Make sure the Program/Configure check box is ticked.**



Flip the **RUN/PROG** switch on the DE2 board to the **PROG** position, turn **OFF** then **ON** the DE2 board. Now press the



Start button in the window. The Progress box will indicate when the configuration and programming process is completed.



### Didn't work?

- Check the same things for JTag programming earlier, power, USB cable, driver for windows etc.
- Make sure the Program/Configure check box above is ticked
- Make sure the DE2 Run/Prog switch is set to Prog

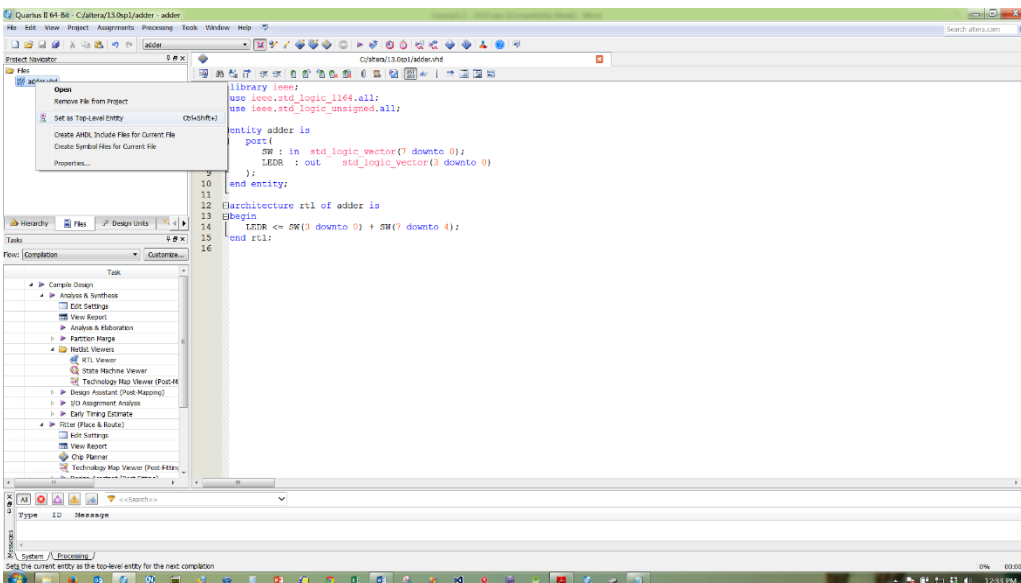
## 8 Testing the Designed Circuit

Having downloaded the configuration data into the FPGA device using either **JTAG** or **AS** mode, you can now test the implemented circuit. Flip the **RUN/PROG** switch on the DE2 board to **RUN** position. Try all four variations of the input variables **x1** and **x2**, by setting the corresponding states of the switches **SW1** and **SW0**. (UP = logic 1, DOWN = logic 0) Green LED 0 should come on when both switches are in different positions to each other and should be off when they are in the same position.

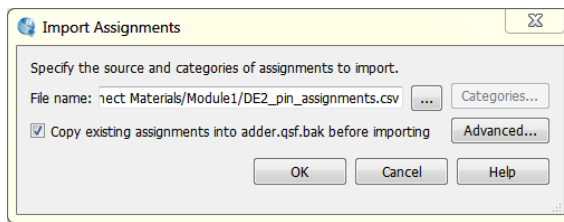
## 9 Simple Exercise for you to complete by yourself

There is VHDL file below for a simple 2 bit adder.

- Create a new Project in Quartus using the same settings as we did in the tutorial above, in particular the Cyclone II device **EP2K35F672C6**.
- Instead of creating a new BDF file (schematic diagram file), create a new VHDL file and past the "adder" code below into it. You will see it uses Switches and Red Leds on the DE2 as the inputs and outputs. Save the file using the name "adder" (to match the entity name below)
- Make sure the file is set as the "Top level entity" in Quartus by right clicking the file and selecting the option shown below



- Compile and simulate the file (don't download at this stage)
- Next, you need to tell Quartus exactly which I/O pins of the FPGA are connected to which signals in the adder. Recall that we have used the "SW" signal as the input for example, and "LEDR" as the output. We need to tell Quartus exactly which pins those signals map to on the DE2 PCB. This is a critically important step. Forgoing this task, or entering an incorrect assignments file, can damage the FPGA! Download the **DE2 pin assignments.csv** file from the Connect site.
- In Quartus II, select **Assignments->Import Assignments**, and select the downloaded file as shown below.



- Click OK. Ensure that a "Import Completed. 425 assignments were written (out of 425 read)" message is visible in the output window, as in Figure 7.
- This file contains pin assignments related to the DE2 board for the SW and LEDR pins specified in the VHDL file.
- Recompile the file
- Download the "adder.sof" file to the DE2 and check that it functions as a 2 bit adder

### VHDL File for 2 bit Adder

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

entity adder is
    port(
        SW      : in    std_logic_vector(7 downto 0);
        LEDR    : out   std_logic_vector(3 downto 0)
    );
end entity;

architecture rtl of adder is
begin
    LEDR <= SW(3 downto 0) + SW(7 downto 4);
end rtl;
```

