



**California State University, Bakersfield**  
**Computer & Electrical Engineering & Computer Science**  
**ECE 3220: Digital Design with VHDL**  
**Laboratory 1**

The purpose of this lab is to familiarize the student with the concepts of

1. Schematic capture, i.e. capturing a circuit design composed of gates using Quartus II
2. To compile the design
3. To generate a Vector Waveform File to simulate the circuit
4. To connect the inputs and outputs of the circuit to LEDS and switches on the DE2 Board
5. Download the design to the FPGA on the DE2 board
6. To test the design on the DE2 board
7. To permanently program the DE2 board with a logic function.

# 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 9.0 SP2; if other versions of the software are used, some of the images may be slightly different (but the principle is the same).

**Contents:**

Typical CAD flow

Getting started

Starting a New Project

Schematic Design Entry

Compiling the Design

Pin Assignment

Simulating the Designed Circuit

Programming and Configuring the FPGA Device

Testing the Designed Circuit

Computer Aided Design (CAD) software makes it easy to implement a desired logic circuit by using a programmable logic device, such as a field-programmable gate array (FPGA) chip. A typical FPGA CAD flow is illustrated in Figure 1.

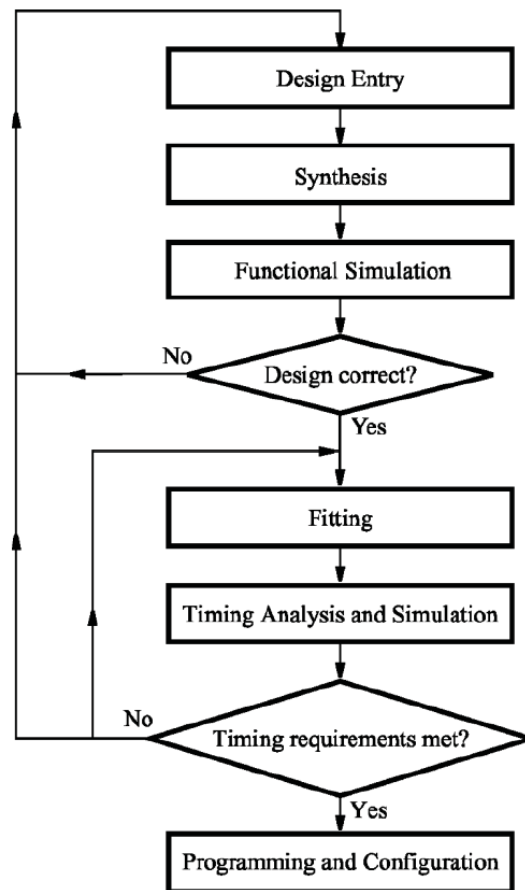


Figure 1. Typical CAD flow.

The CAD flow involves the following steps:

- **Design Entry** – the desired circuit is specified either by means of a **schematic diagram**, or by using a hardware description language, such as **Verilog** or **VHDL**.
- **Synthesis** – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip
- **Functional Simulation** – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues

- **Fitting** – the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in the actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs.
- **Timing Analysis** – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit.
- **Timing Simulation** – the fitted circuit is tested to verify both its functional correctness and timing.
- **Programming and Configuration** – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections.

This tutorial introduces the basic features of the Quartus II software. It shows how the software can be used to design and implement a circuit specified by means of a schematic diagram. It makes use of the graphical user interface to invoke the Quartus II commands. Doing this tutorial, the reader will learn about:

- Creating a project
- Entering a schematic diagram
- Synthesizing a circuit from the schematic diagram
- Fitting a synthesized circuit into an Altera FPGA
- Assigning the circuit inputs and outputs to specific pins on the FPGA
- Simulating the designed circuit
- Programming and configuring the FPGA chip on Altera's DE2 board

## 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.

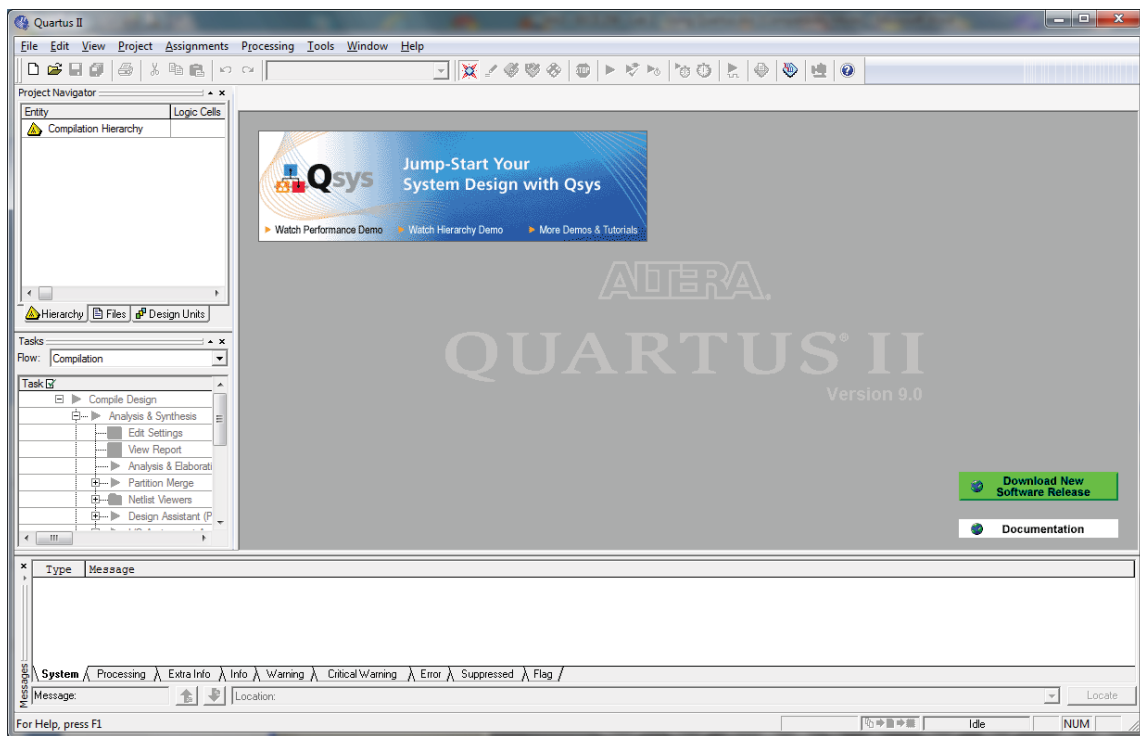


Figure 2. The main Quartus II V9.0 SP2 display.

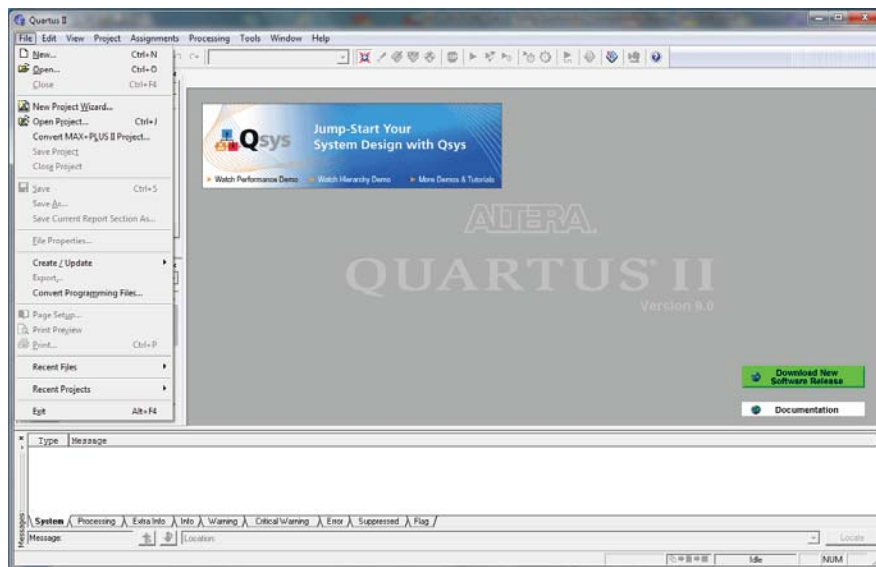


Figure 3. An example of the File menu.

For some commands it is necessary to access two or more menus in sequence. We use the convention **Menu1 > Menu2 > Item** to indicate that to select the desired command the user should first click the left mouse button on **Menu1**, then within this menu click on **Menu2**, and then within **Menu2** click on **Item**. For example, **File > Exit** uses the mouse to exit from the system. Many commands can be invoked by clicking on an icon displayed in one of the toolbars. To see the command associated with an icon, position the mouse over the icon and a tooltip will appear that displays the command name.

## 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. To get some idea of the extent of documentation provided, it is worthwhile for the reader to browse through the **Help** menu. For instance, selecting **Help > How to Use Help** gives an indication of what type of help is provided.

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. While using most applications, pressing the **F1** function key on the keyboard opens a Help display that shows the commands available for the application.

## 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 "Quartus II Project" Wizard** to reach the window in Figure 4, which asks for the name and directory (folder) of the project.

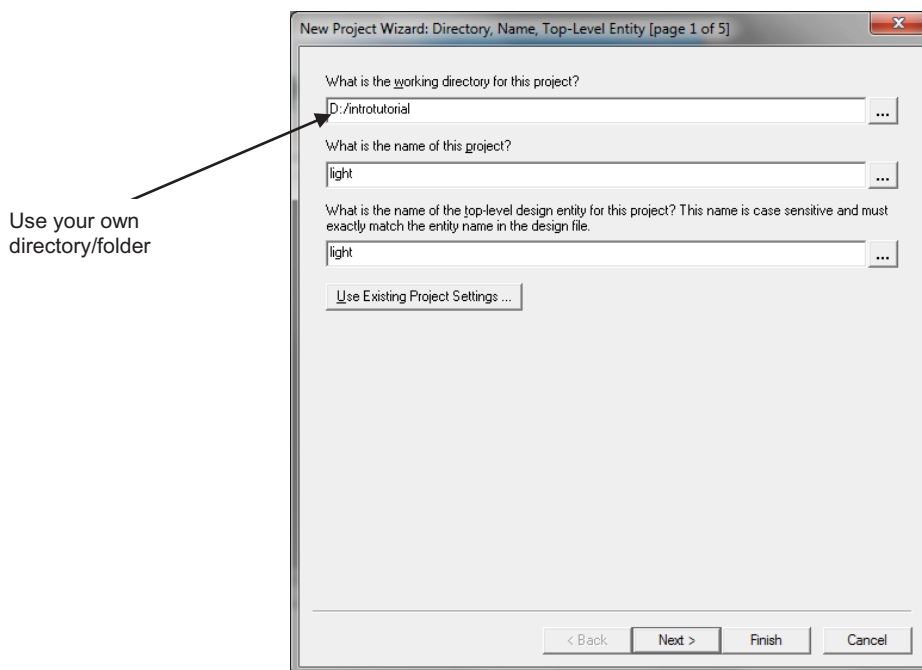


Figure 4. Creation of a new project.

2. Set the working directory to be *introtutorial*; of course, you are advised to use some other directory such as Z: if working in the lab on a networked systems, or C: if you prefer. The project must have a name, which is usually the same as the **top-level design entity** that will be included in the project. Choose *light* as the name for both the project and the top-level entity, as shown in Figure 4. Press **Next**.

Since we have not yet created the directory *introtutorial*, Quartus II software displays the pop-up box in Figure 5 asking if it should create the desired directory. Click **Yes**, which leads to the window in Figure 6.

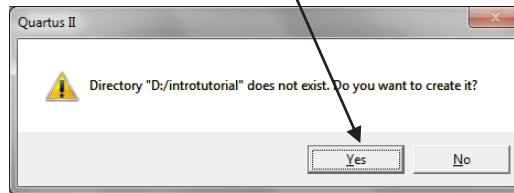


Figure 5. Quartus II software can create a new directory for the project.

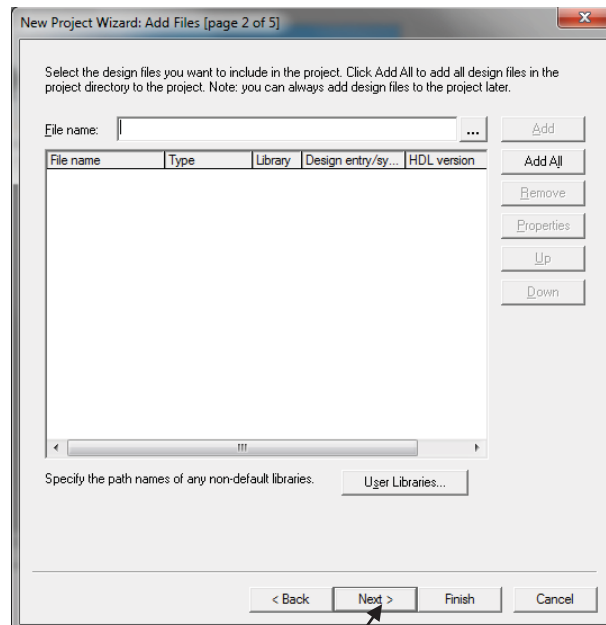


Figure 6. The wizard can include user-specified design files.

3. 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 Figure 7.



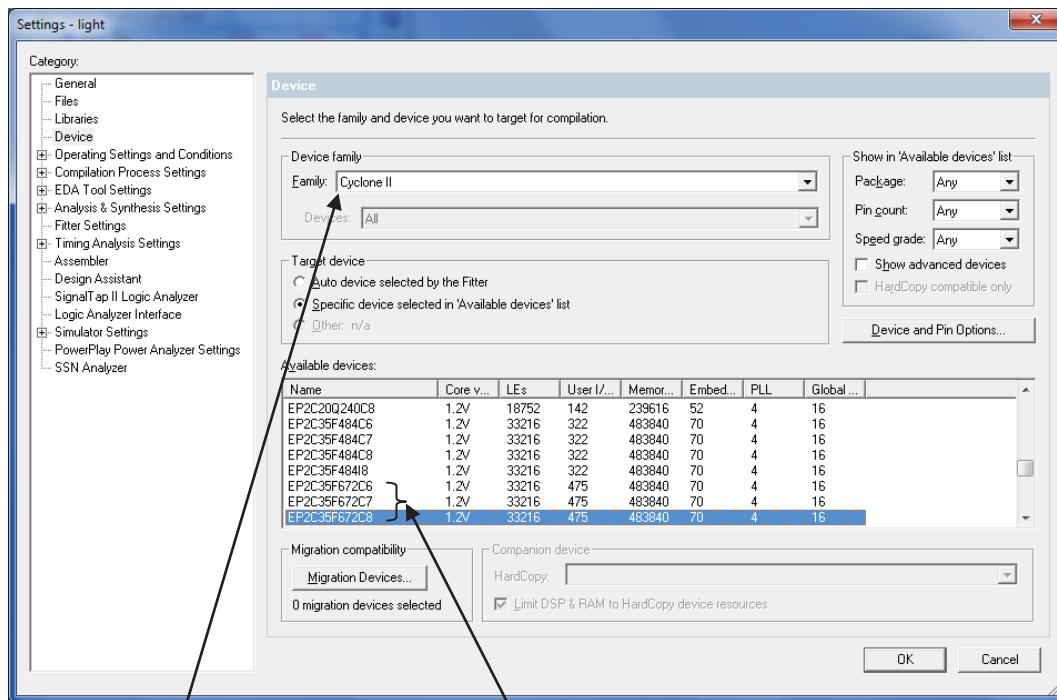


Figure 7. Choose the device family and a specific device **EP2C35F672C8** or **EP2C35F672C6** for DE2 Boards. **Inspect your board to find out what chip is fitted** and select accordingly.

- We have to specify the type of device in which the designed circuit will be implemented. Choose **Cyclone™ II** as the target device family. We can let Quartus II software select a specific device in the family, or we can choose the device explicitly. We will take the latter approach. From the list of available devices, choose the device called **EP2C35F672C8** or **EP2C35F672C6** whichever is the device **fitted** to your DE2 board. Press **Next**, which opens the window in Figure 8.

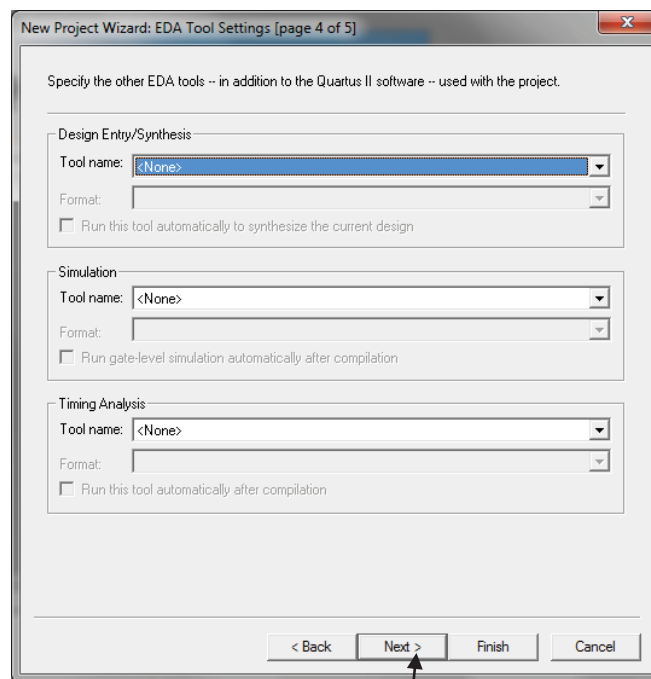


Figure 8. Other EDA tools can be specified.

5. The user can specify any third-party 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. This term is used in Quartus II messages that refer to third-party tools, which are the tools developed and marketed by companies other than Altera. Since we will rely solely on Quartus II tools, we will not choose any other tools. Press [Next](#).
6. A summary of the chosen settings appears in the screen shown in Figure 9.

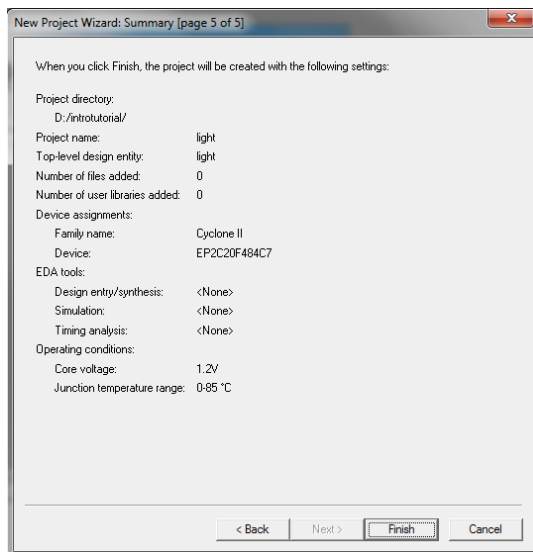


Figure 9. Summary of the project settings Note.

Press Finish, which returns to the main Quartus II window, but with *light* specified as the new project, in the display title bar, as indicated in Figure 10.

Note Project

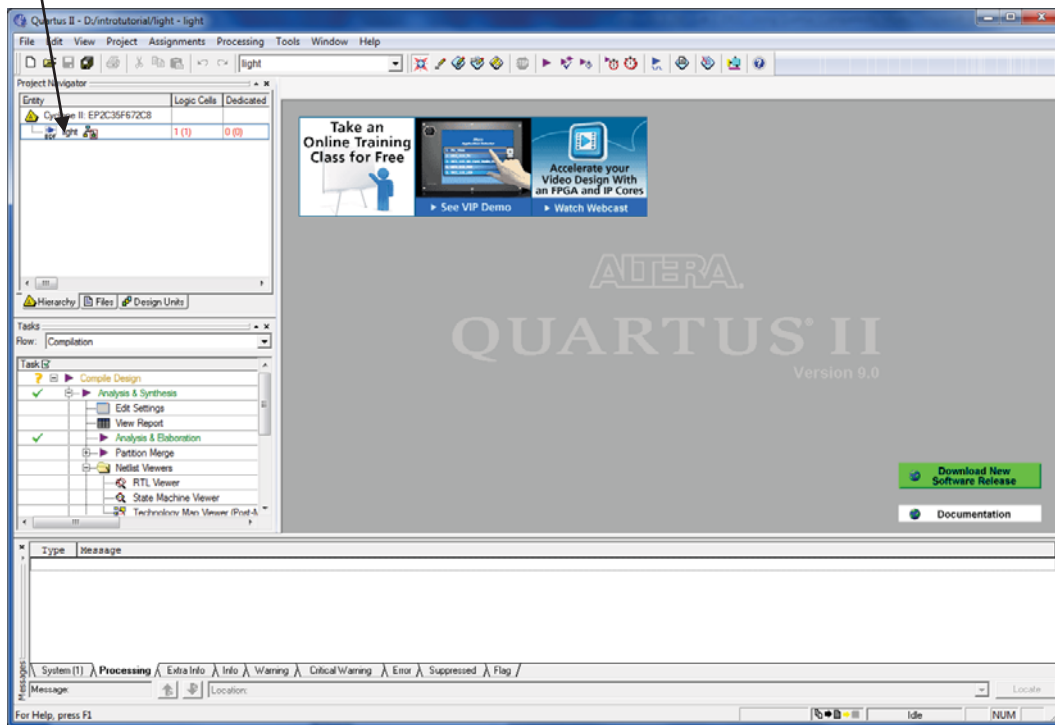


Figure 10

### 3 Design Entry Using the Graphic Editor

As a design example, we will use the two-way light controller circuit shown in Figure 11. 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.

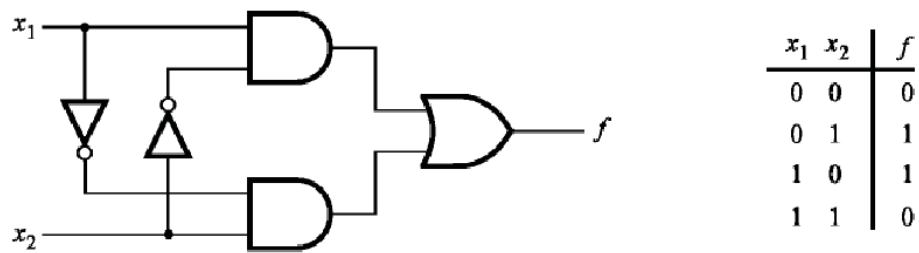


Figure 11. The light controller circuit.

The Quartus II Graphic Editor can be used to specify a circuit in the form of a block diagram. Select **File > New**, choose **Block Diagram/Schematic File**, (to get the window in Figure 12) and click **OK**. This opens the Graphic Editor window.

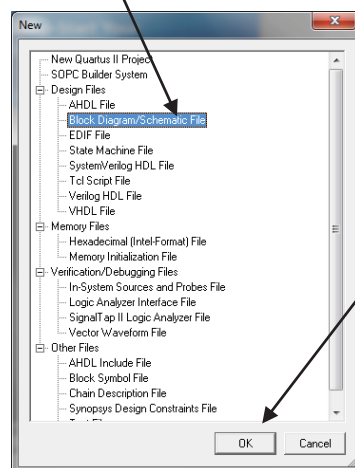


Figure 12. Choose to prepare a block diagram.

The first step is to specify a name for the file that will be created. Select **File > Save As** to open the pop-up box depicted in Figure 13.

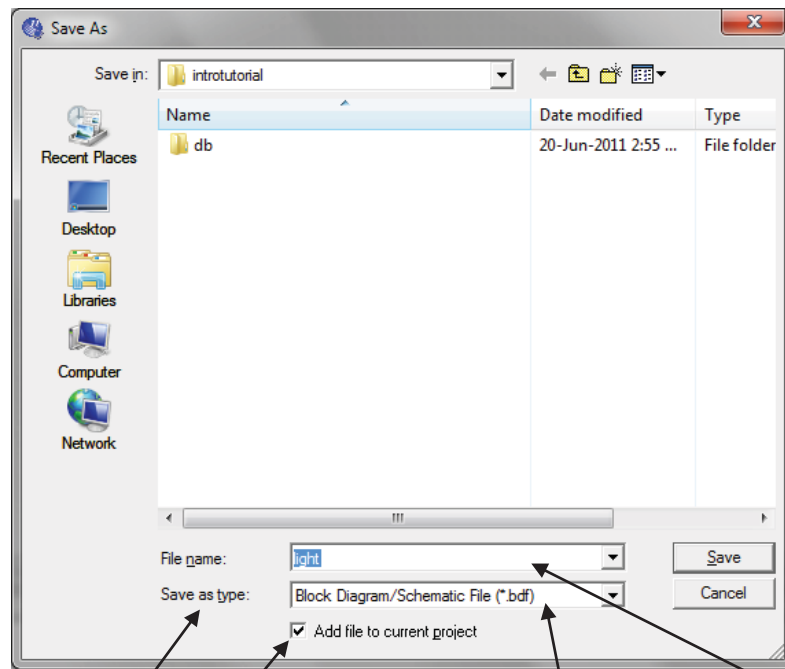


Fig 13 – Save file

In the box labeled **Save as type** choose **Block Diagram/Schematic File (\*.bdf)**. In the box labeled File name type *light*, to match the name given in Figure 4, which was specified when the project was created. Put a checkmark in the box **Add file to current project**. Click **Save**, which puts the file into the directory *introtutorial* and leads to the Graphic Editor window displayed in Figure 14.

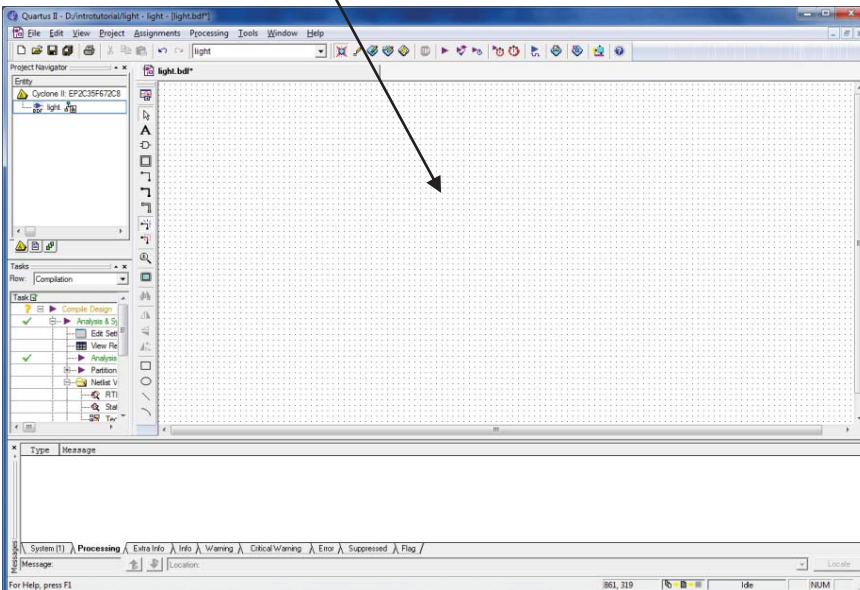

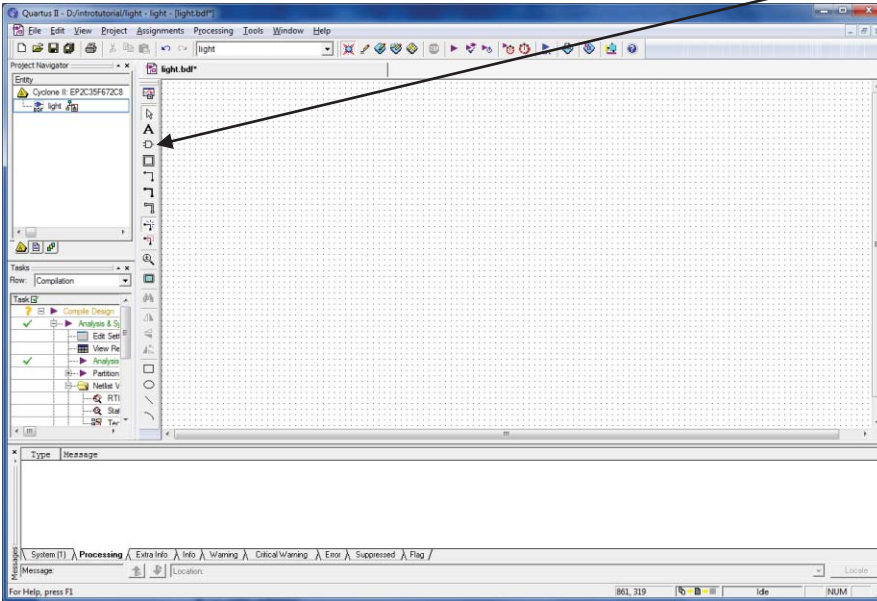


Fig 14

### 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. Double-click on the blank space in the Graphic Editor window, or click on the icon  in the toolbar that looks like an **AND** gate.



A pop-up box like that in Figure 15 will appear.

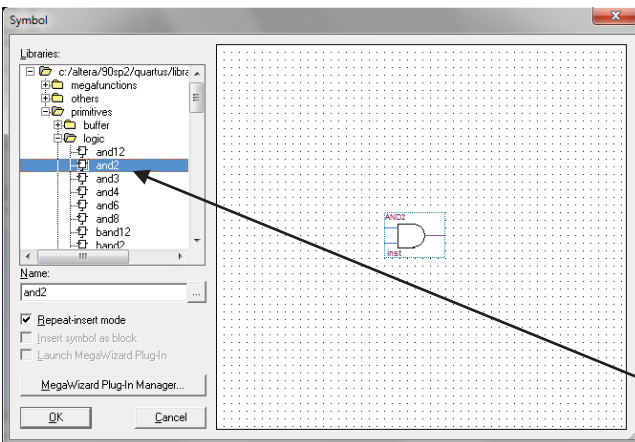



Fig 15

Expand the hierarchy in the Libraries box as shown in the figure. First expand **libraries**, then expand the library **primitives**, followed by expanding the library **logic** which comprises the logic gates. Select **and2**, which is a two-input AND gate, and click OK. Now, the AND gate symbol will appear in the Graphic Editor window. Using the mouse, move the symbol to a desirable location and click to place it there. Import the second AND gate, which can be done simply by positioning the mouse pointer over the existing AND-gate symbol, right-clicking, and dragging to make

a copy of the symbol. (NOTE Press **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.

Next, select **or2** from the library and import the OR gate into the diagram. Then, 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 in Figure 16.

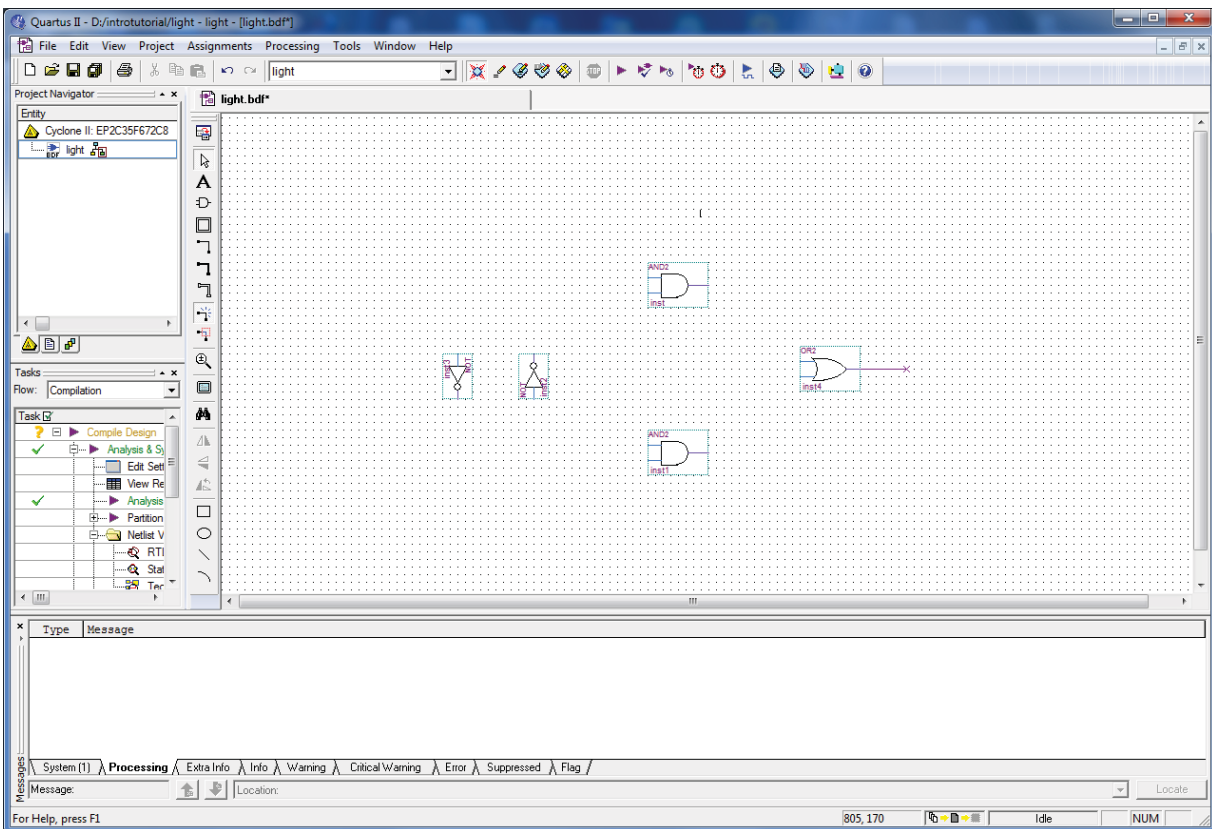
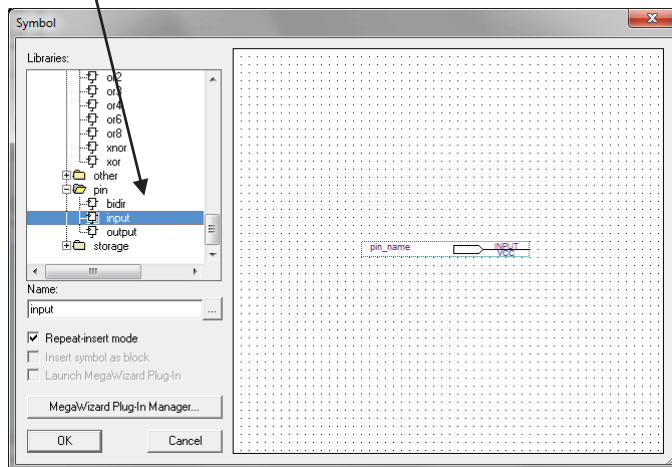


Figure 16. Import the gate symbols into the Graphic Editor window.

### 3.2 Importing Input and Output Symbols

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*.



Import two instances of the **input pin** and one instance of the **output pin**, to obtain the image in Figure 17. Don't worry about the **names** given to the pins, we will change those in a moment.

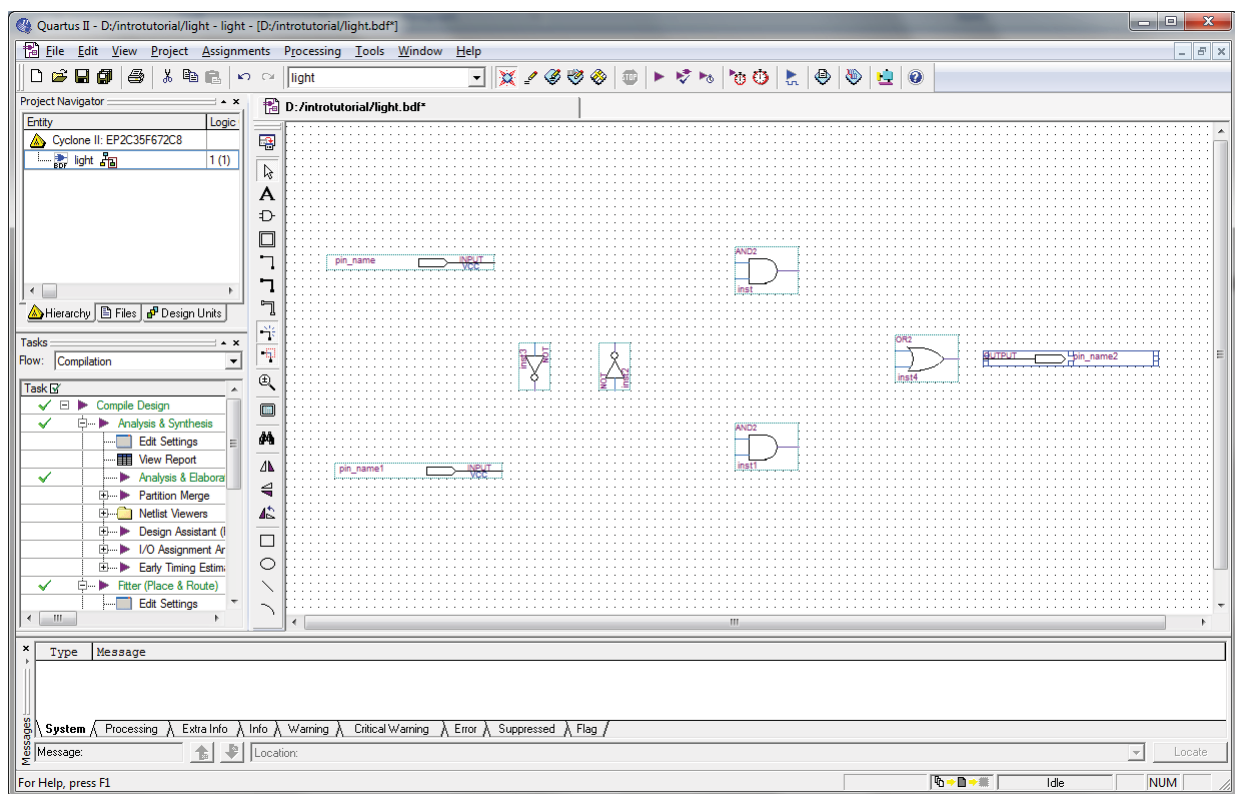




Figure 17. Import the input and output pins.

Assign names to the input and output symbols as follows. Point to the word *pin\_name* on the top input symbol and **double-click** the mouse. The dialog box in Figure 18 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*

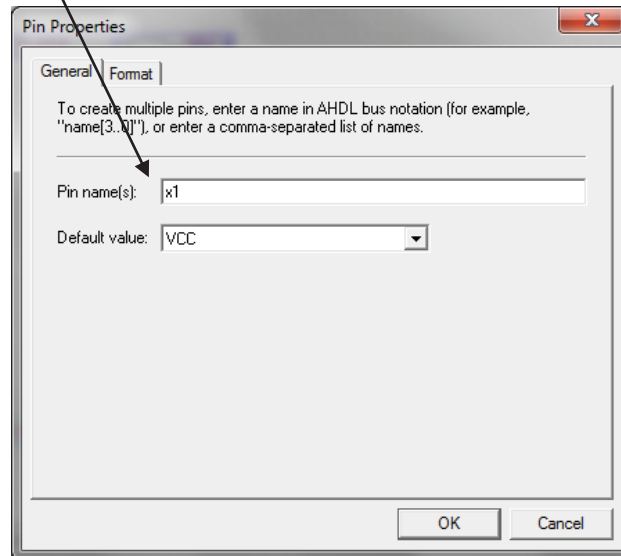



Figure 18. Naming of a pin.

### 3.3 Connecting Nodes with Wires

The symbols in the diagram have to be connected by drawing lines (wires). Click on the icon in the toolbar to activate the Orthogonal Node Tool. Position the mouse pointer over the right edge of the **x1** input pin. Click and hold the mouse button and drag the mouse to the right until the drawn line reaches the pinstub on the top input of the AND gate. Release the mouse button, which leaves the line connecting the two pinstubs. Next, draw a wire from the input pinstub of the leftmost NOT gate to touch the wire that was drawn above it. Note that a dot will appear indicating a connection between the two wires.

Use the same procedure to draw the remaining wires in the circuit. If a mistake is made, a wire can be selected by clicking on it, and removed by pressing the **Delete key** on the keyboard. Upon completing the diagram, click on the icon  to activate the Selection and Smart Drawing Tool. Now, changes in the appearance of the diagram can be made by selecting a particular symbol or wire and either moving it to a different location or deleting it. The final diagram is shown in Figure 19; **save it**.

Note you can move wires and gates around by clicking on them and dragging them with the mouse or using the arrows keys on the keyboard

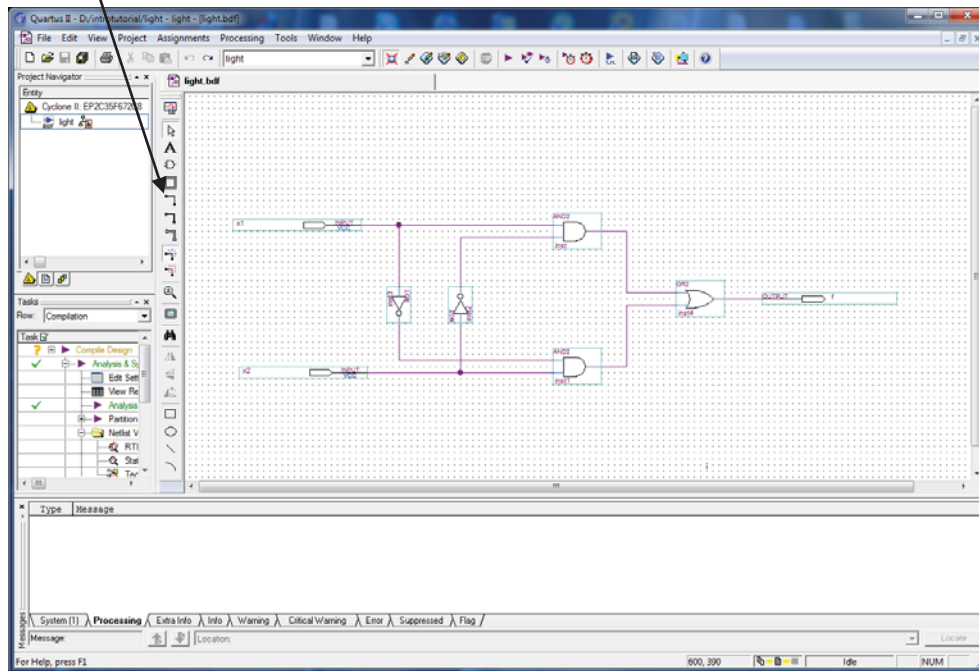
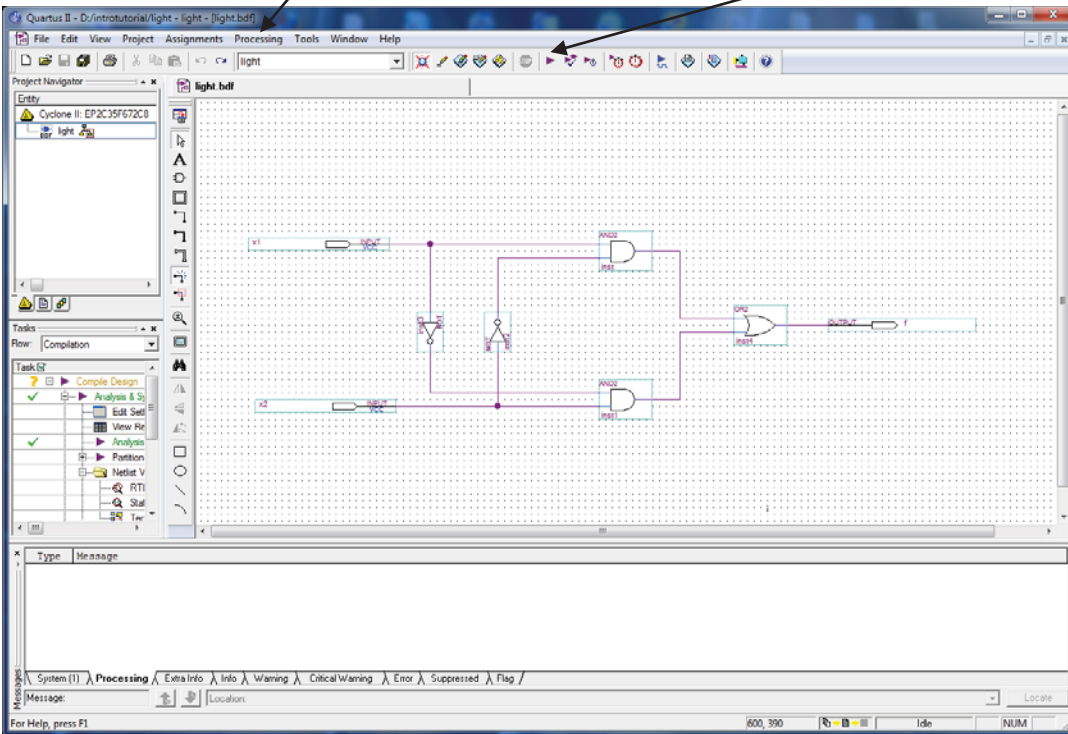


Fig 19 - Final Diagram

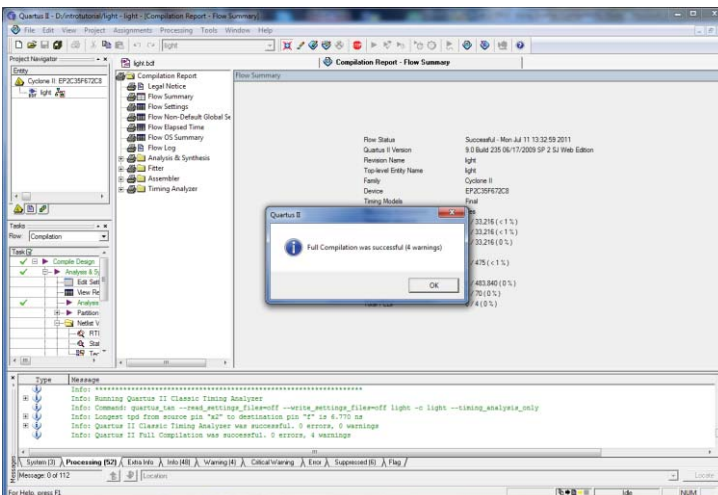
## 4 Compiling the Designed Circuit

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 target chip (the FPGA that we download to on the DE2 board). These tools are controlled by the application program called the *Compiler*.

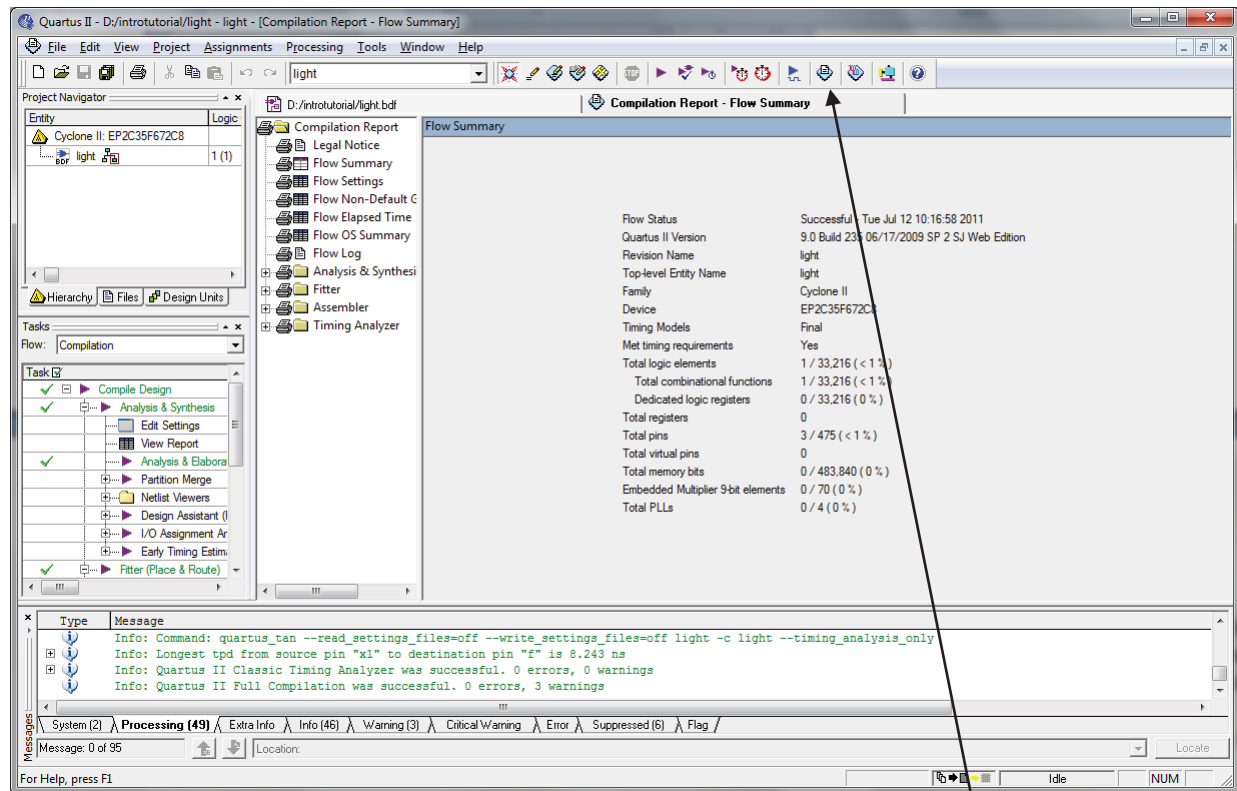
Run the Compiler by selecting **Processing > Start Compilation**, or by clicking on the toolbar icon . (See below)




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 a pop-up box. (See below)



Acknowledge it by clicking OK, which leads to the Quartus II display below. In the message window, at the bottom of the figure, various messages are displayed. In case of errors, there will be appropriate messages given. **Warnings are OK** and can be ignored for the moment.




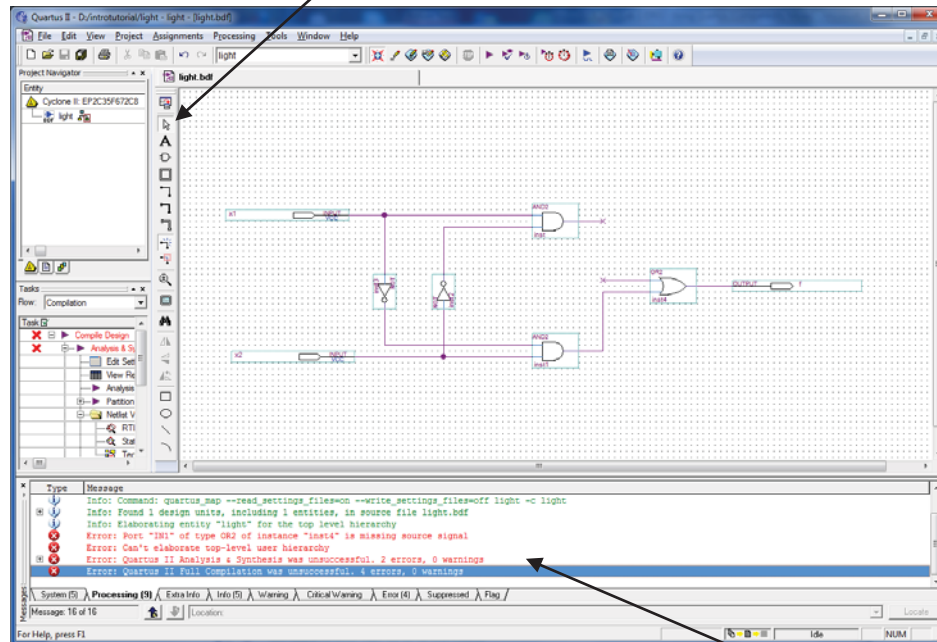
When the compilation is finished, a compilation report is produced (see above) A window showing this report is opened automatically. The window can be resized, maximized, or closed in the normal way and it can be opened at any time either by selecting **Processing > Compilation Report** or by clicking on the icon . The report includes a number of sections listed on the left side of its window. The Figure above displays the Compiler Flow Summary section, which indicates that only one logic element and three pins are needed to implement this tiny circuit on the selected FPGA chip.


## 4.1 Errors

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.

If the Compiler does not report zero errors, then there is at least one mistake in the schematic entry. In this case a message corresponding to each error found will be displayed in the Messages window. Double-clicking on an error message will highlight the offending part of the circuit in the Graphic Editor window. Similarly, the Compiler may display some warning messages. Their details can be explored in the same way as in the case of error messages. The user can obtain more information about a specific error or warning message by selecting the message and pressing the F1 function key.

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



Compile the erroneous design by clicking on the compile icon . A pop-up box will ask if the changes made to the *light.bdf* file should be saved; click **Yes**. After trying to compile the circuit, Quartus II software will display a pop-up box indicating that the compilation was not successful. Acknowledge it by clicking **OK**. The errors are listed in the bottom window, you will have to **correct** all errors before proceeding further, so put the wire back in place and recompile until you have no errors.

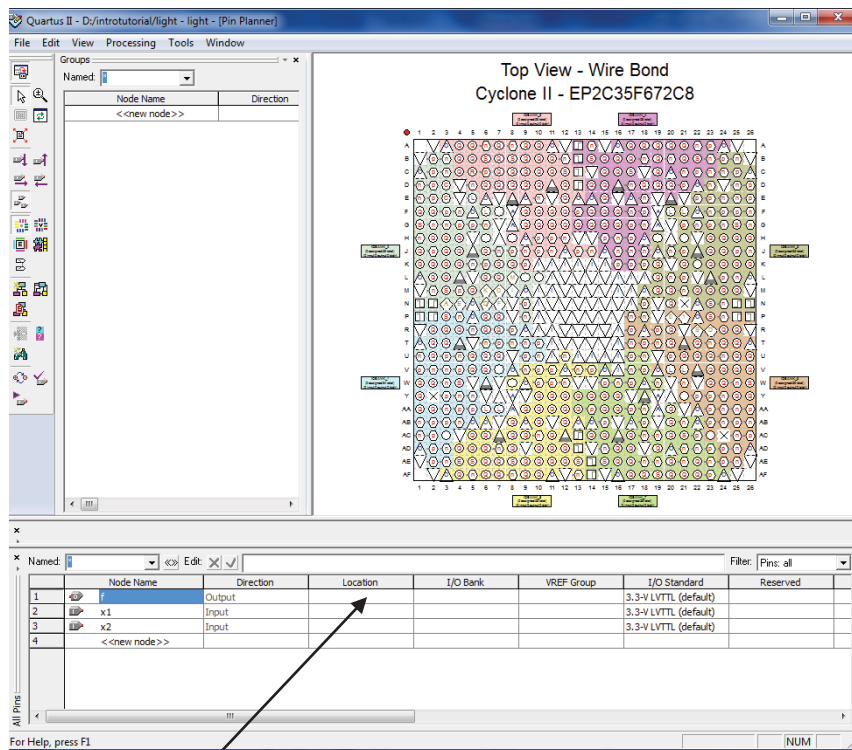
## 5 Physical Pin Assignment

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 between certain FPGA pins and the other components on the board such as the LEDs, switches, 7 segment displays etc, so we have to assign the signals from our circuit (i.e.  $x_1$ ,  $x_2$ ,  $f$ ) to the physical pins on the FPGA 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** and  $x_2 = SW_1 = \text{PIN}_N26$ , respectively (See page 28 in the DE2 User Manual on the course web-site).

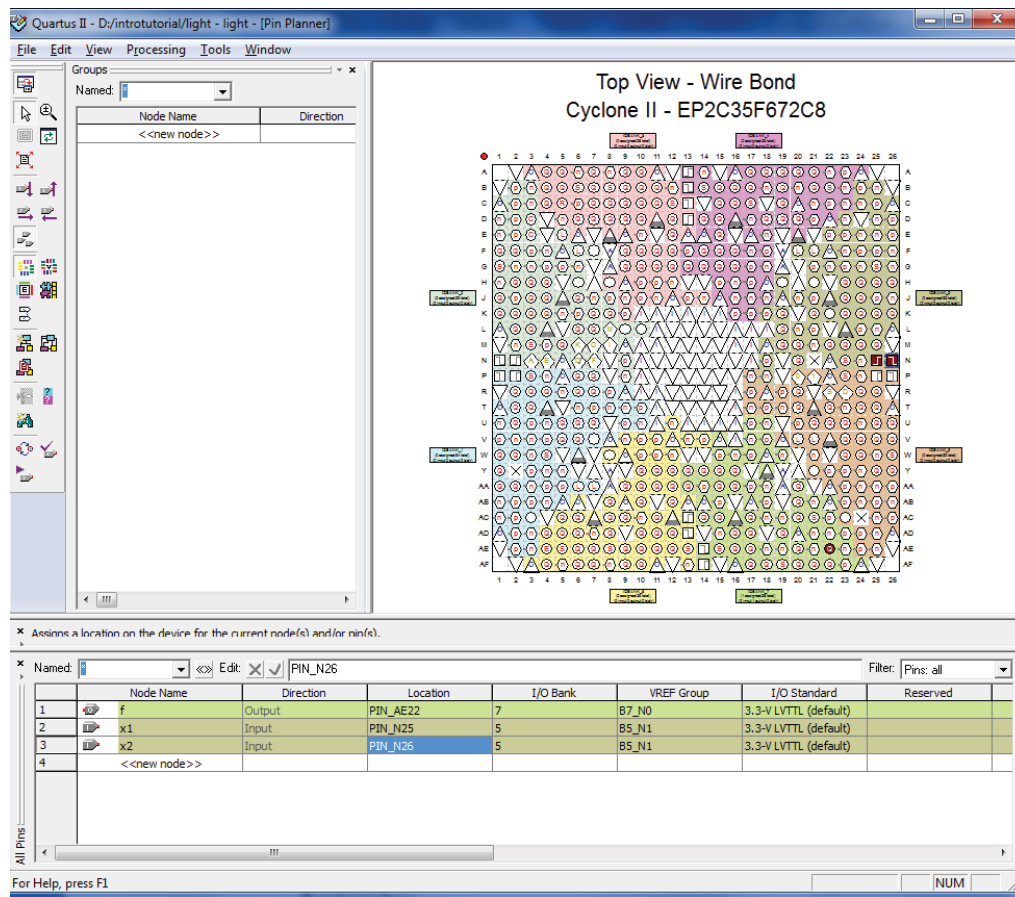
We will connect the output  $f$  to the green light-emitting diode labeled  $LEDG_0$  which is hardwired to the FPGA pin **PIN\_AE22**. (See page 29 in the DE2 User Manual on the course website). To map the pins shown on our circuit diagram to physical pins on the FPGA that are hard wired to the switches and LEDs we require, right mouse click on a pin in the circuit schematic diagram and when the pop-up menu appears select 'Locate' -> 'Locate in Pin Planner'

The Pin Planner is opened as shown below listing all the inputs/outputs of our circuit



In the 'Location' cell to the right of each pin name, enter the FPGA pin that we want that signal to connect to on the FPGA. To make an assignment of circuit pin to physical FPGA pin, type the physical FPGA pin name into the cell next to the signal, 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 in the Figure below



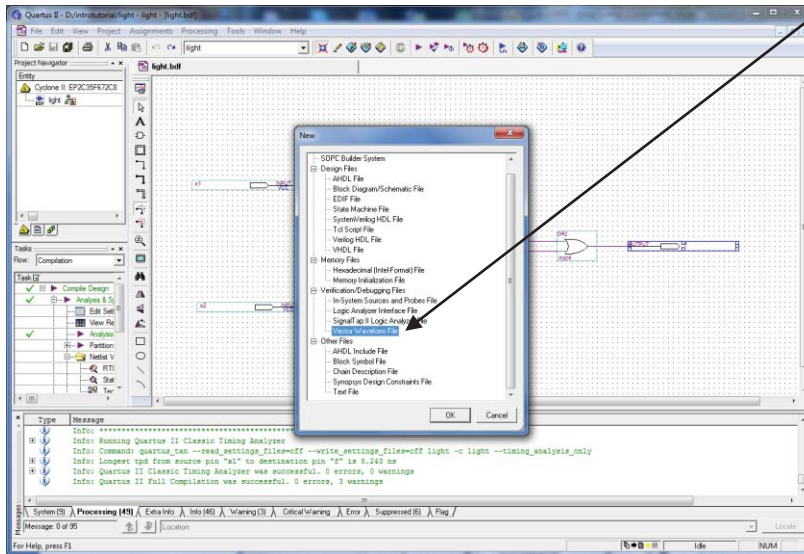
Now you have made the pin assignments, **close** the Pin Planner which will save your pin assignments and **IMPORTANTLY** **Recompile the circuit**, so that it will be compiled with the correct pin assignments.



## 6 Simulating the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE2 board, it is prudent to simulate it to ascertain 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 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 II Waveform Editor to draw the test vectors, as follows:

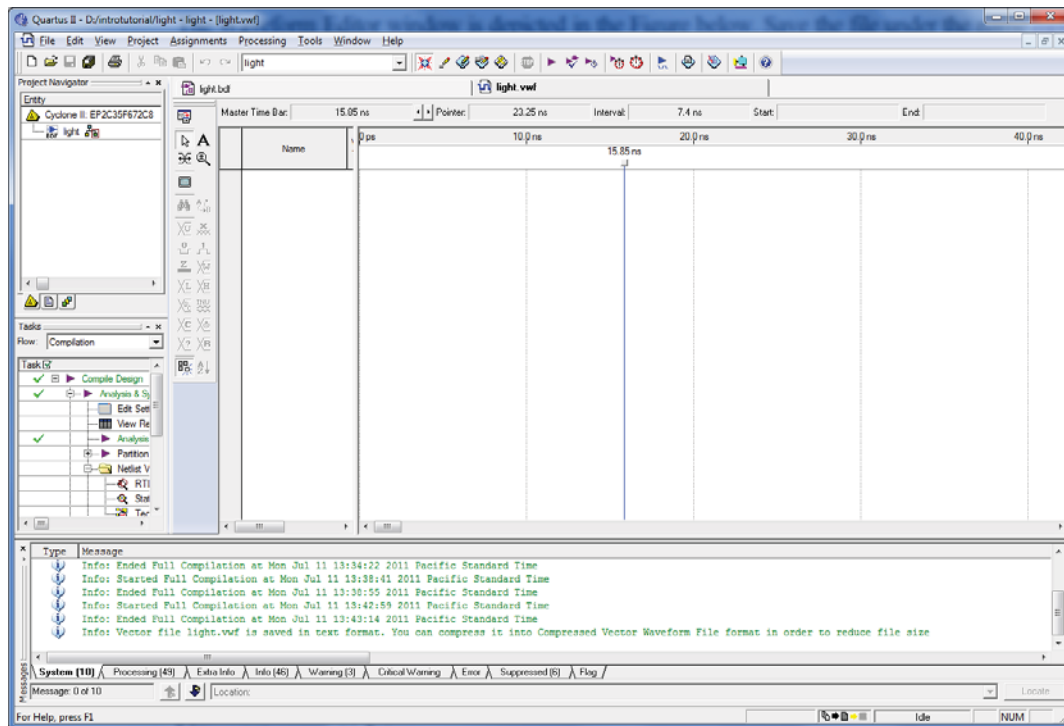
1. Open the Waveform Editor window by selecting **File > New** and select **Vector Waveform File**



Creating a new Vector Waveform File.



The Waveform Editor window is depicted 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 **Edit > End Time** and entering 200 ns in the dialog box that pops up.

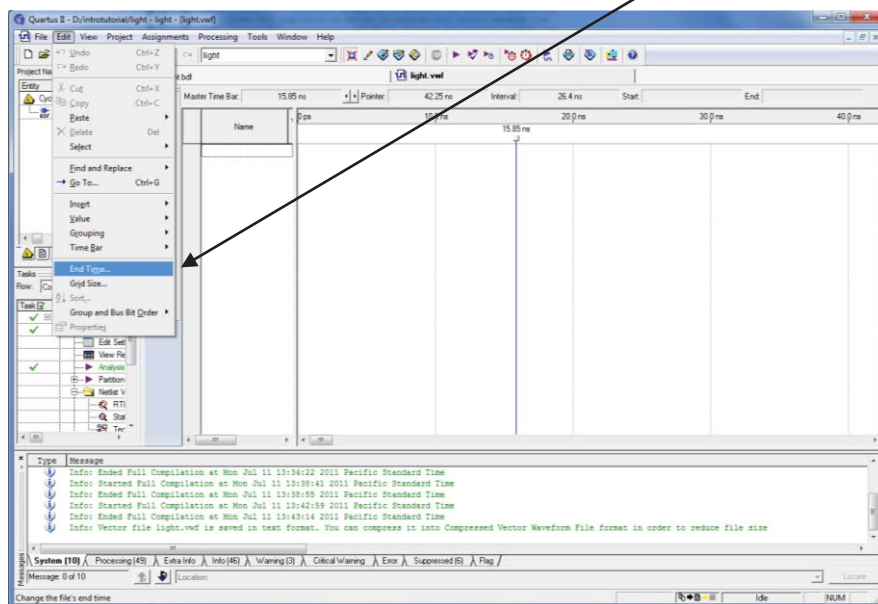
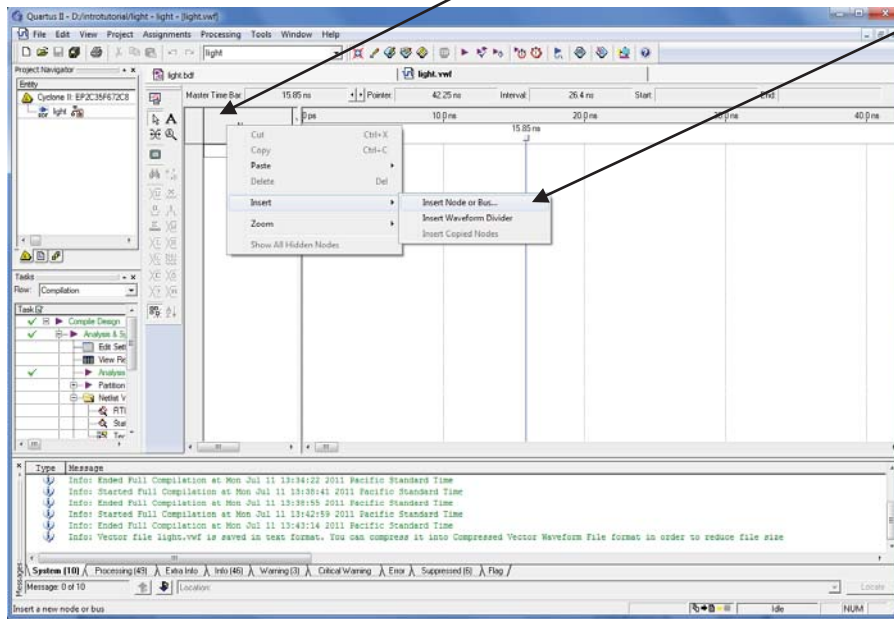


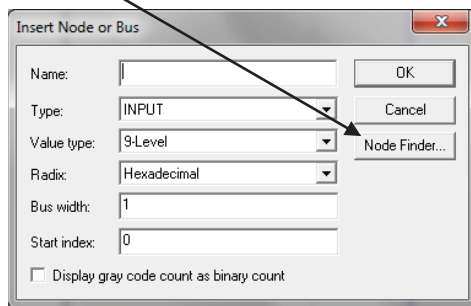
Figure 33. The augmented Waveform Editor window.

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.

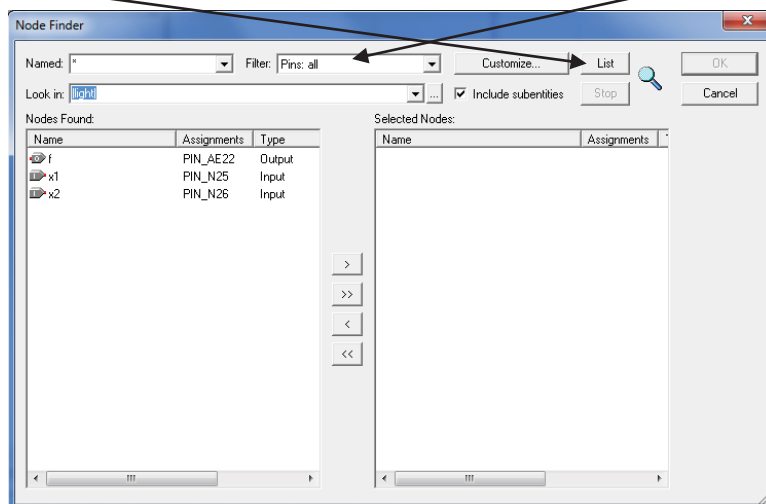
Right mouse click in the area of the waveform file **here**, (or click menu **Edit > Insert** ) and select **Insert Node** or **Bus** as shown below.



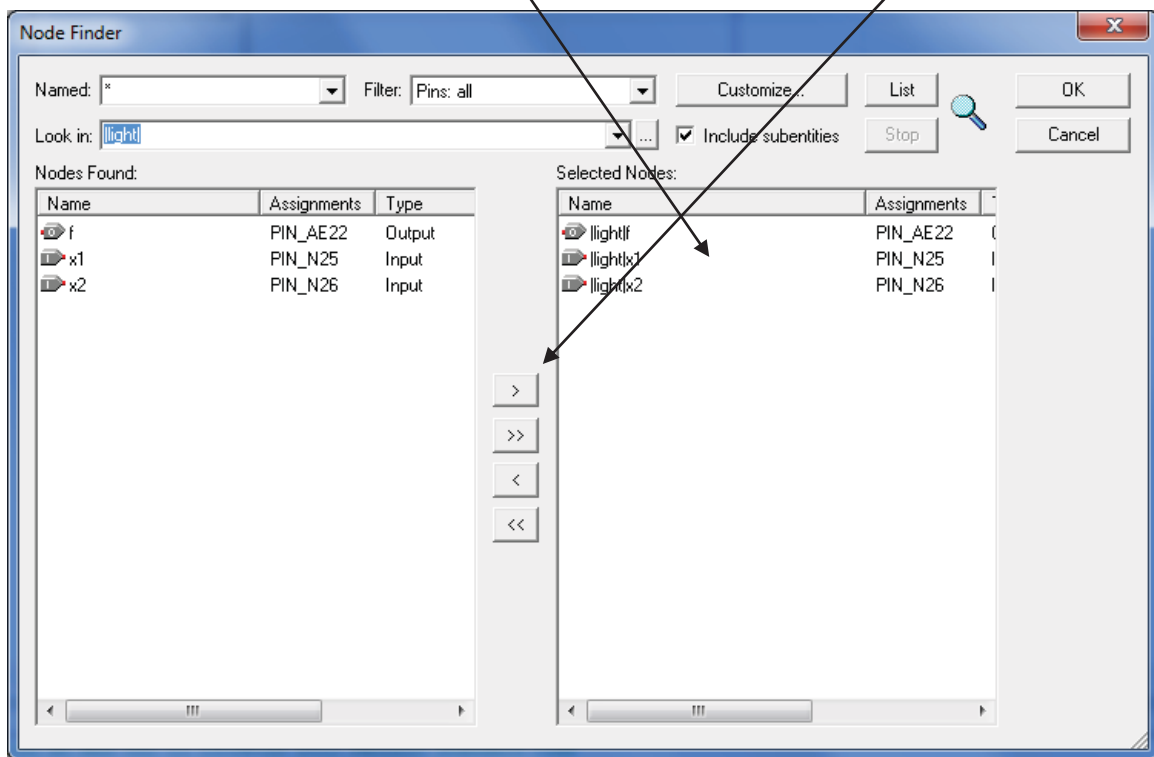
It is possible to type the name of a signal (pin) into the Name box, but it is easier to click on the button labeled **Node Finder** as shown below.



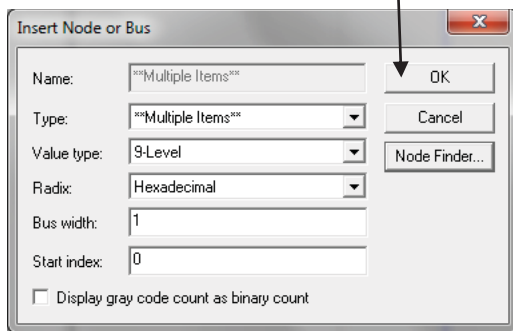
This will open the **Node Finder** utility. The Node Finder utility has a filter used to indicate what type of nodes are to be found. Since we are interested in input and output pins, set the **filter to Pins: all**. Click the **List** button to find the input and output nodes as indicated on the left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



Click on each of the 3 signals  $f$ ,  $x1$ ,  $x2$  in the Nodes Found box and then click the arrow  $\rightarrow$  sign to add it to the **Selected Nodes** box on the right side of the figure as shown below. Then click **OK**

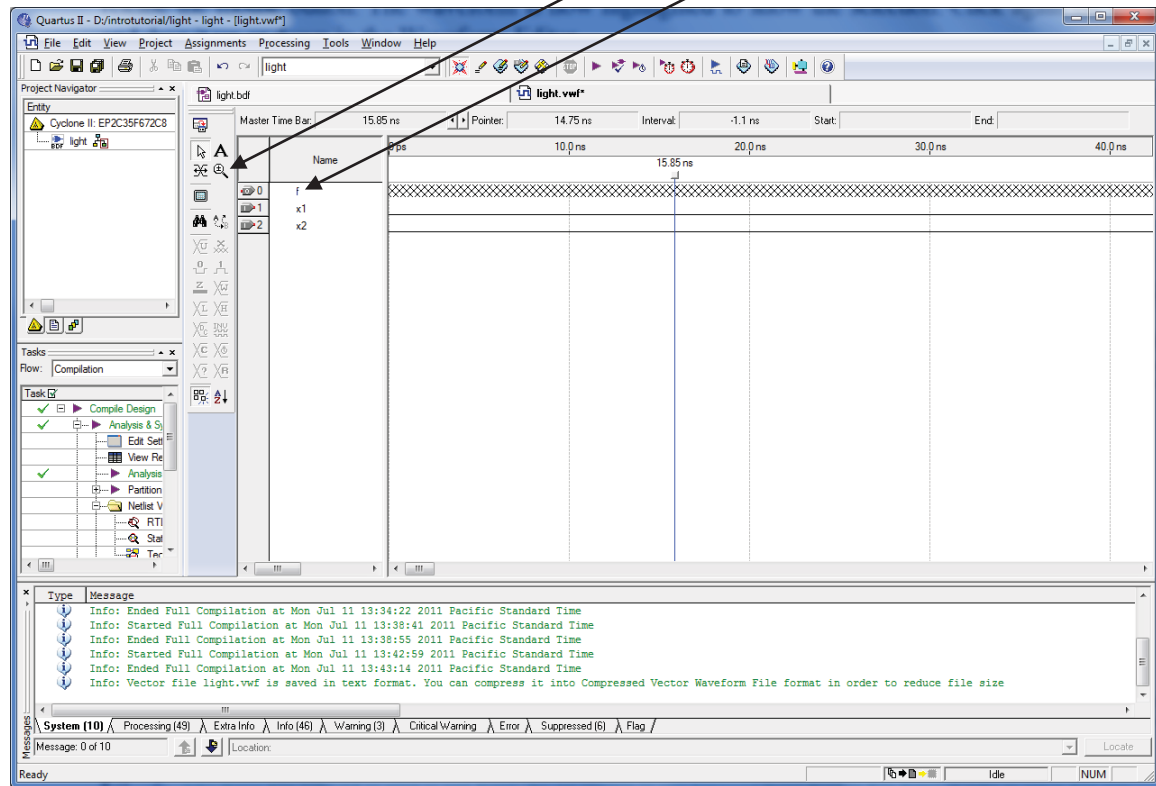


When this dialog box appears, click **OK**



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.

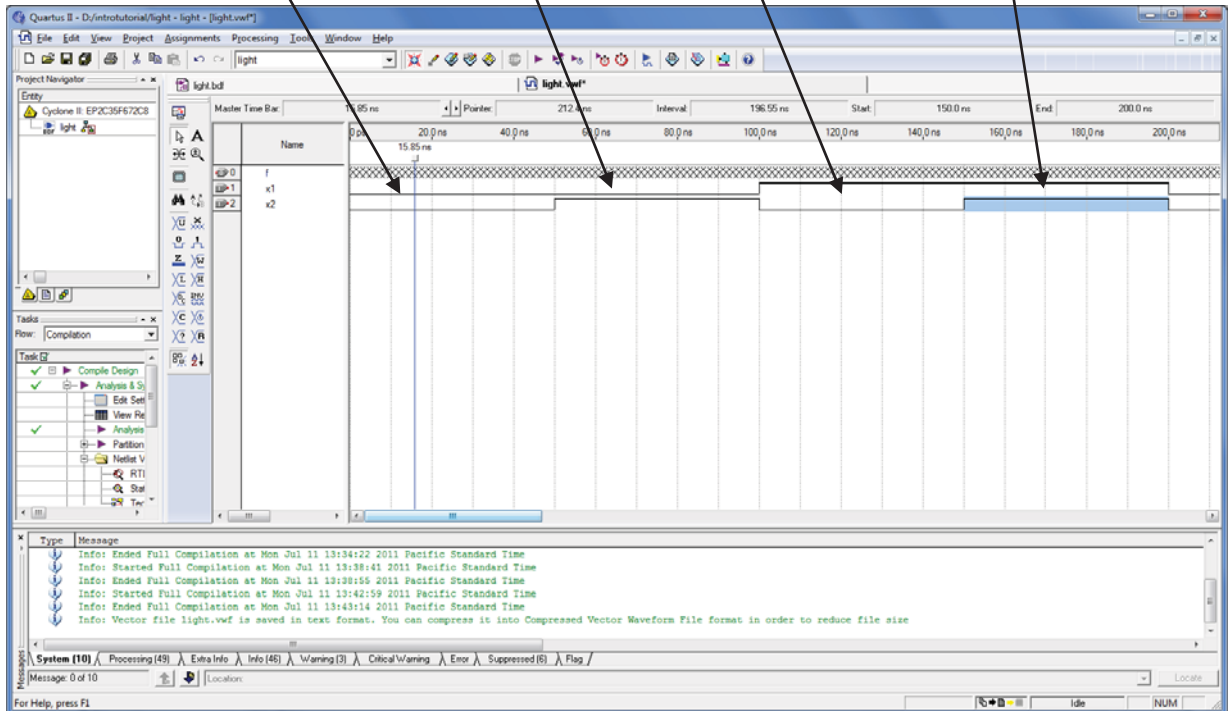
To zoom **in** or **out** on the waveform file, select the **magnifying tool** and use the right or left mouse buttons to click on the waveform file



4. 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 outputs for a set of inputs). 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. However, for our tiny circuit we can simulate all four input valuations for a 2 input logic function, i.e. the input values, **00, 01, 10, 11**. We will use four 50-ns time intervals to apply the four test vectors.

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**, 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.

Set  $x1$  to 0 in the time interval **0 to 100 ns**, which is probably already set by default. Next, set  $x1$  to 1 in the time interval **100 to 200 ns**. Do this by pressing 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. Make  $x2 = 1$  from **50 to 100 ns** and also from **150 to 200 ns**, which corresponds to the truth table. This should produce the image in the Figure below. Thus we have inputs **00** for time 0-50ns, input **01** for time 50-100ns, **10** for time 100-150ns and **11** for time 150-200ns. Observe that the output  $f$  is displayed as having an **unknown** value at this time, which is indicated by a hashed pattern; its value will be determined during simulation. Save the file



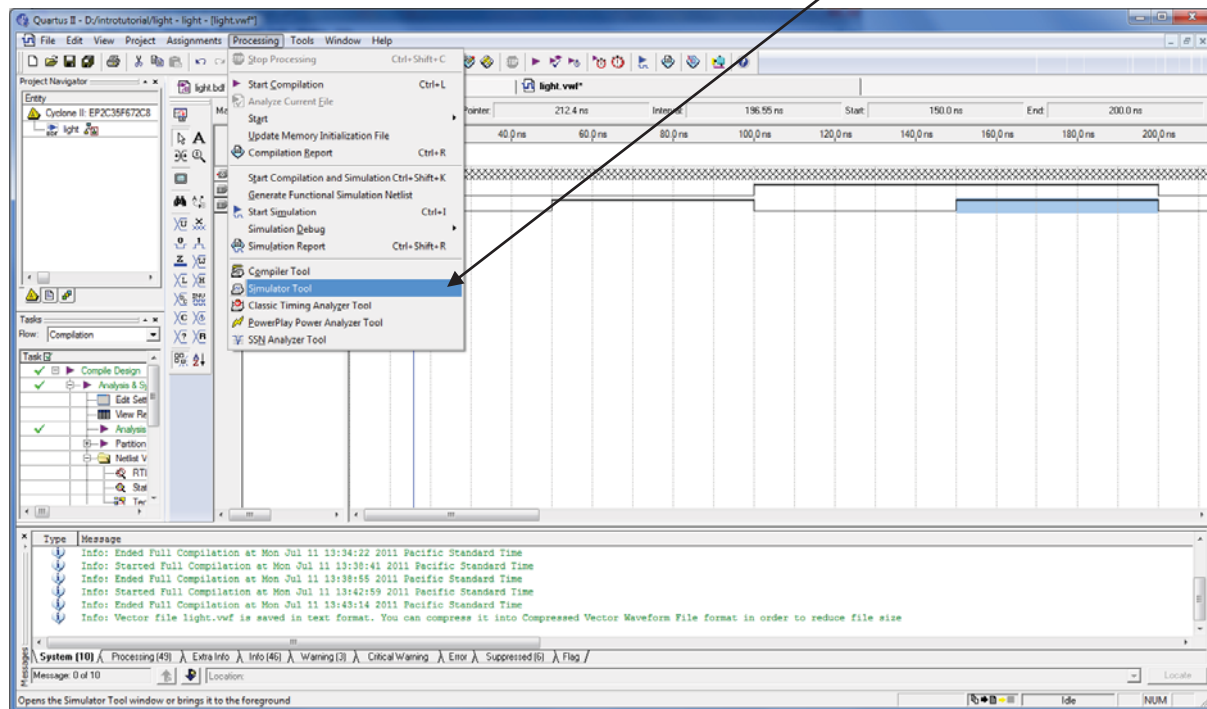
## 6.1 Performing the Simulation

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 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 timing. This takes much less time, because the simulation can be performed simply by using the logic expressions that define the circuit.

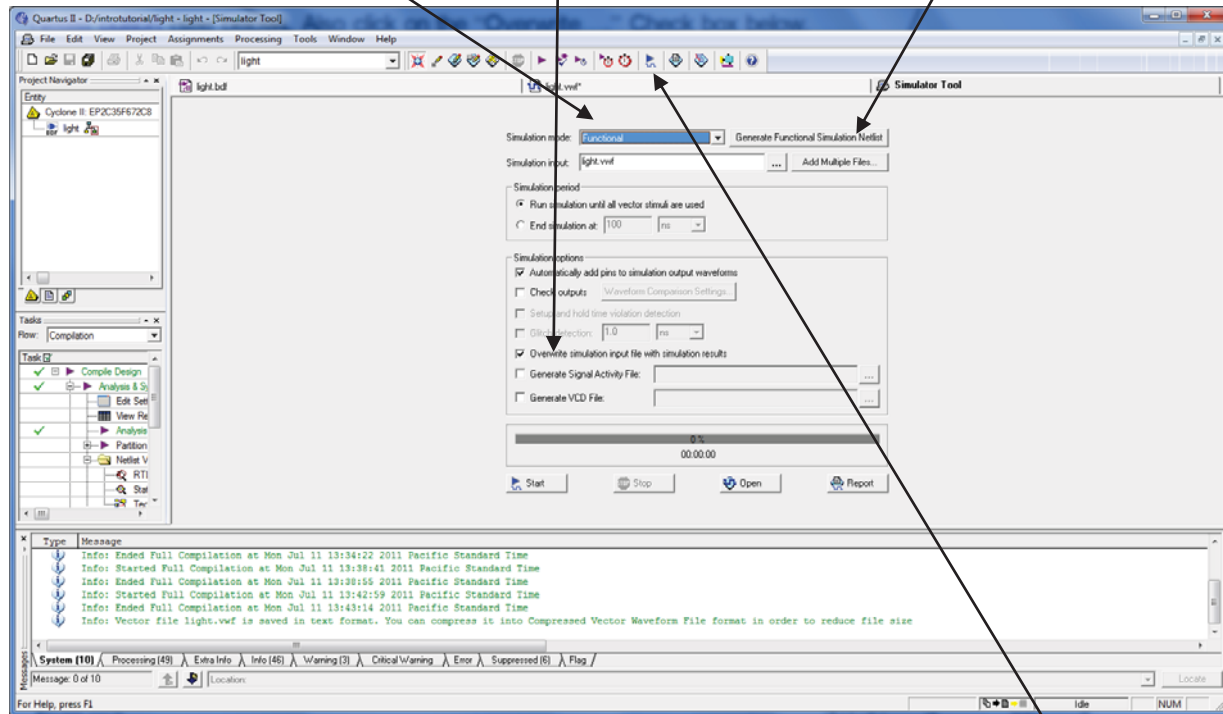
### 6.1.1 Functional Simulation


To perform the **functional simulation**, open the Simulator Tool/Window (menu **Processing->Simulator tool**) as shown below



The simulator window opens – maximize it.

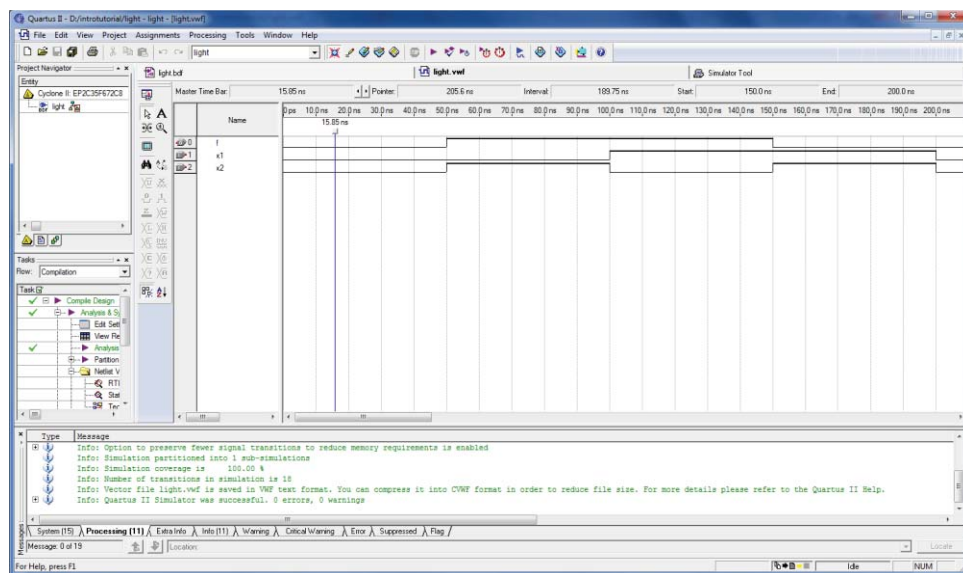
Now choose **Functional** in the **Simulation mode** window and then press the **Generate Functional Simulation Netlist** button (see below). Also click on the “**Overwrite....**” Check box below.



A simulation run is started by selecting menu **Processing > Start Simulation**, or by pressing the icon . At the end of the simulation, Quartus II software indicates its successful completion and displays a Simulation Report. If your report window does not show the entire simulation time range, click on the report window to select it and **choose View > Fit in Window**.

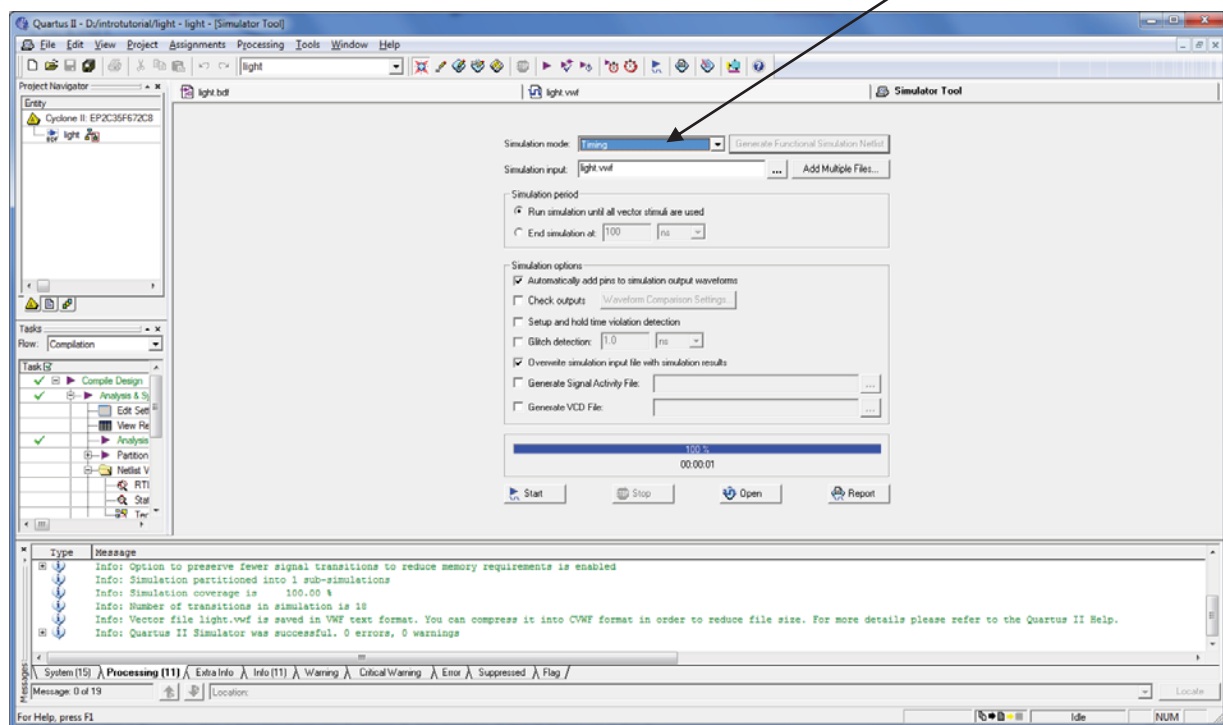
Observe that the output  $f$  is as shown below.





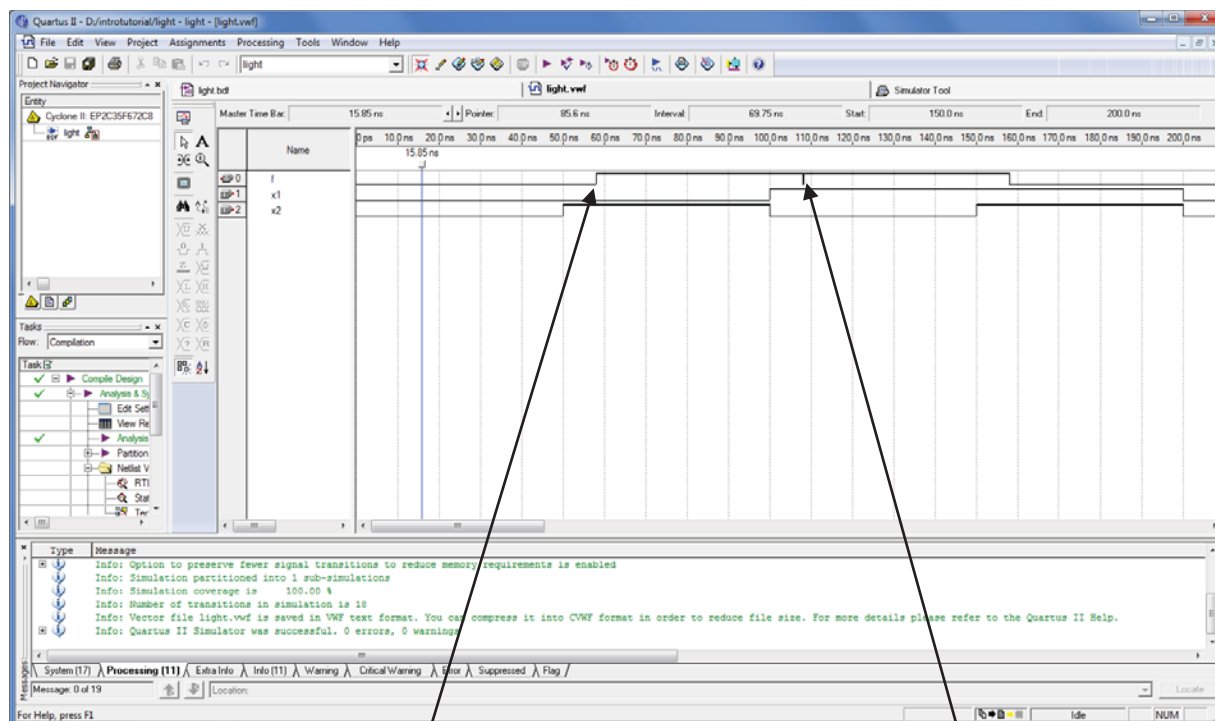
### 6.1.2 Timing Simulation

Having ascertained that the designed circuit is **functionally** correct, we should now perform the timing simulation to see how it will behave when it is actually implemented in the chosen FPGA device, i.e. taking account of propagation delays in wires and logic gates. In the simulator tool/window, set the simulation to **Timing** (there is **no** need to generate a Functional Simulation Netlist now so it is not available to be clicked).



Re-simulate and observe the resultant simulation below





Observe that there is a **delay of about 8 ns** in producing a change in the signal  $f$  from the time when the input signals,  $x_1$  and  $x_2$ , change their values. This delay is due to the propagation delays in the **logic elements** and the **wires** in the FPGA device. Notice also some glitches or momentary ‘spikes’ on the output ‘ $f$ ’ at time 108ns and 210ns (this is normal).

## 7 Programming and Configuring the FPGA Device

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus II Compiler’s Assembler module. Altera’s DE2 board allows the configuration to be done in two different ways, known as **JTAG** and **AS** modes.

The configuration data 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**. If this driver is not already installed, **consult the course web-site** or the tutorial *Getting Started with Altera’s DE2 Board* for information about installing the driver. Before using the board, make sure that the USB cable is properly connected and turn on the power supply switch on the board.

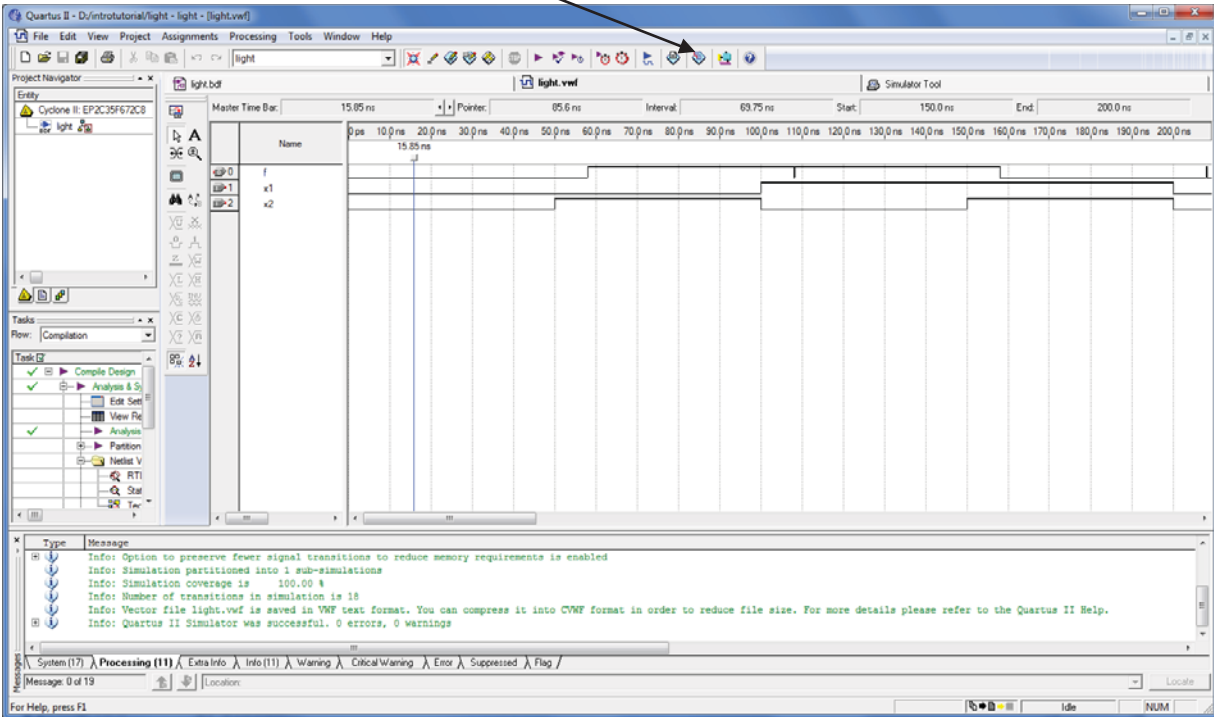
In the JTAG mode, the configuration data is loaded directly into the FPGA device. The acronym JTAG stands for Joint Test Action Group. This group defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. If the FPGA is configured in this manner, it will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

The second possibility is to use the **Active Serial (AS)** mode. In this case, a memory chip on the DE2 board is used to store the configuration data downloaded by Quartus. When power is applied to the DE2, the data from the memory chip is loaded automatically into the FPGA (which takes about  $\frac{1}{2}$  sec). In this way, you **don’t** have to use Quartus to download the configuration 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.

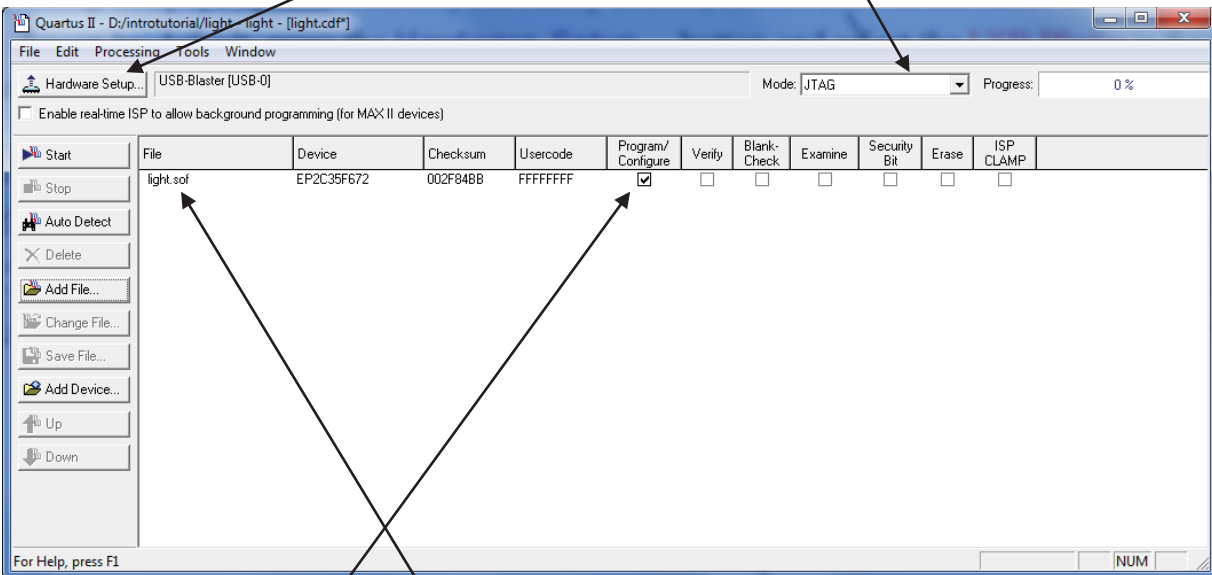
The choice between the two modes or download is made by the **RUN/PROG** switch on the DE2 board. The **RUN** position selects the JTAG mode, while the **PROG** position selects the **AS** mode.

## 7.1 JTAG Programming

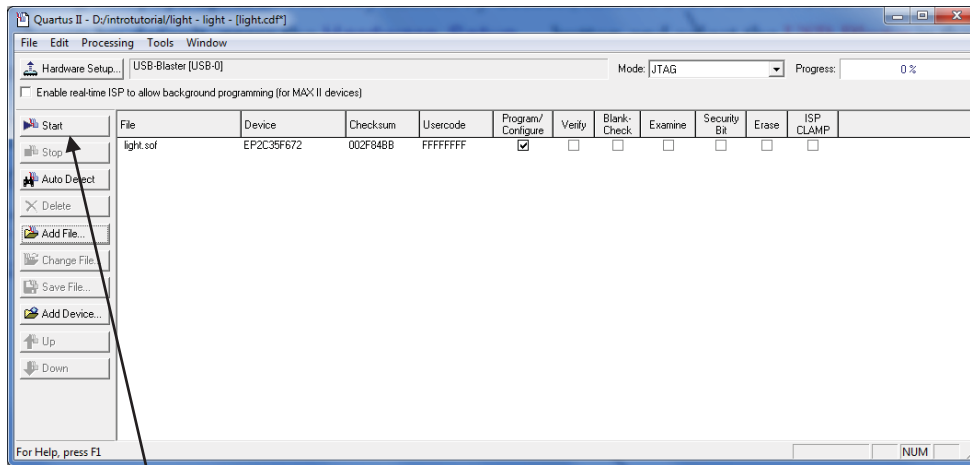
The programming and configuration task is performed as follows. Flip the **RUN/PROG** switch on the DE2 into the **RUN** position. Click on the Programmer button below



The following dialog box pops up. If not already chosen by default, select **JTAG** in the Mode box. Also, 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*).



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. This is a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device. 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.

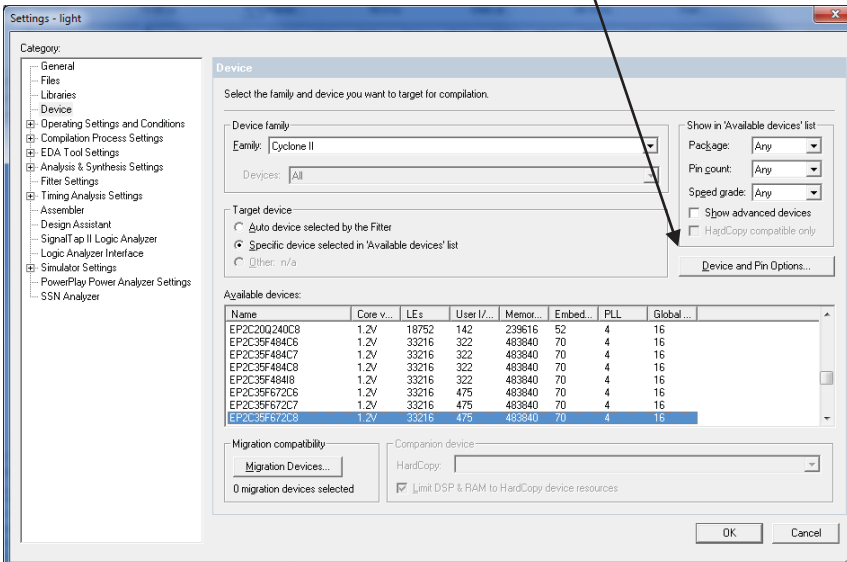


Now, press **Start** button in the window. An LED on the DE2 board will light up when the configuration data has been downloaded successfully. If you see an error reported by Quartus II software indicating that programming failed, check to ensure that the board is properly powered on.

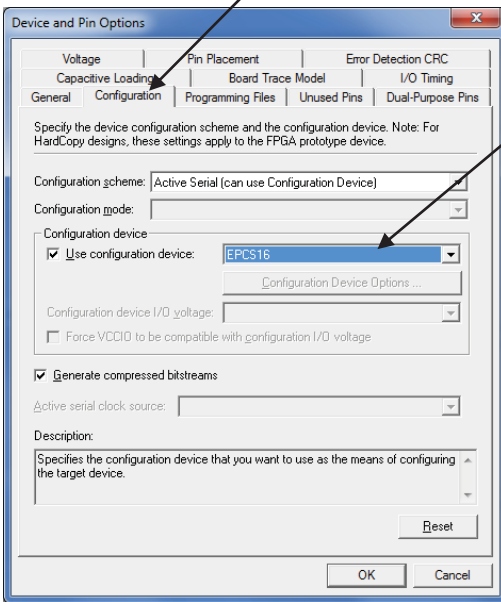
**(NOTE SKIP SECTION 7.2 for the MOMENT and goto section 8, you can return to it later)**

## 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 **EP2C35F672C6**. To specify the required configuration device, select menu **Assignments > Device**, which leads to the window below. Click on the **Device and Pin Options** button

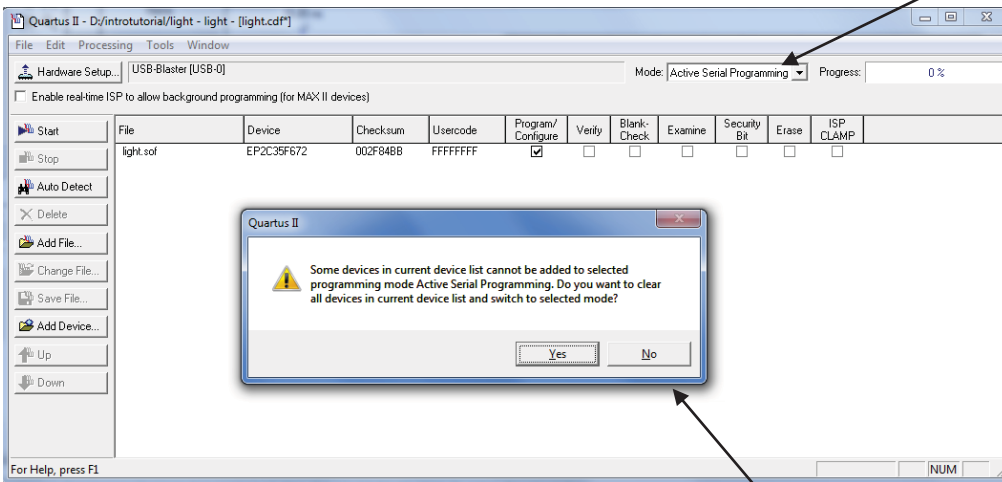


Now, click on the **Configuration** tab as shown below and select the **EPCS16** device

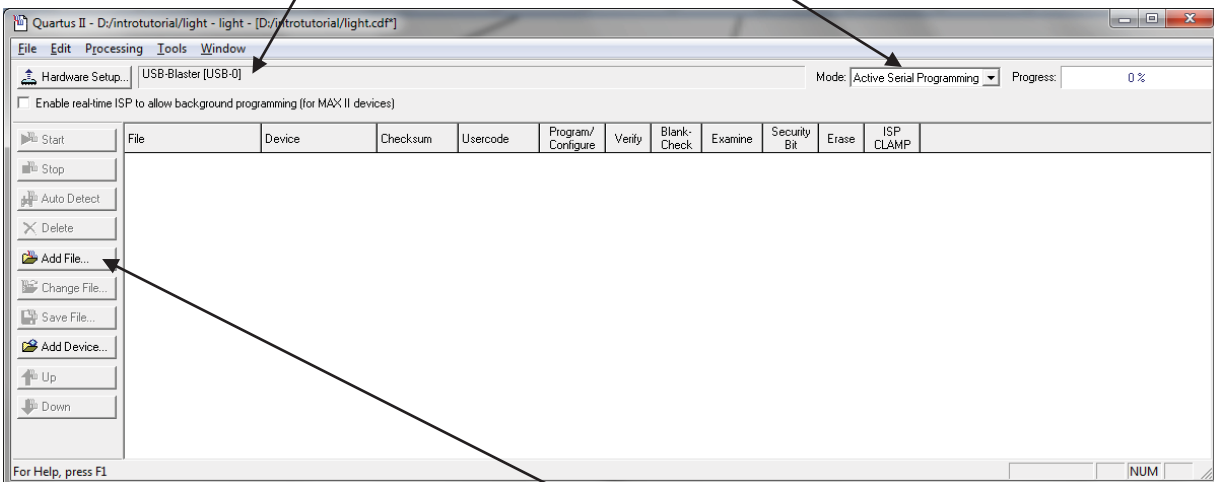


Now you **MUST RECOMPILE** the designed circuit. The rest of the procedure is similar to the one described above for the JTAG mode.

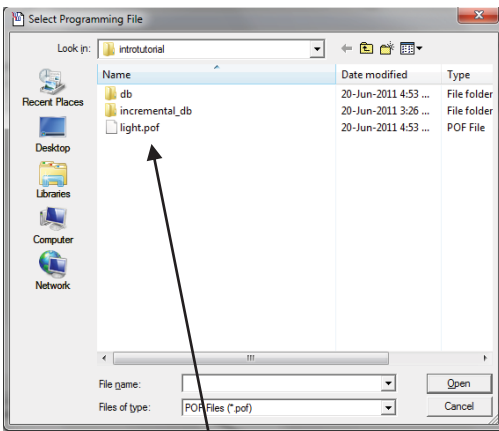
Select the **Programmer** tool to reach the window below. In the Mode box select **Active Serial Programming**.



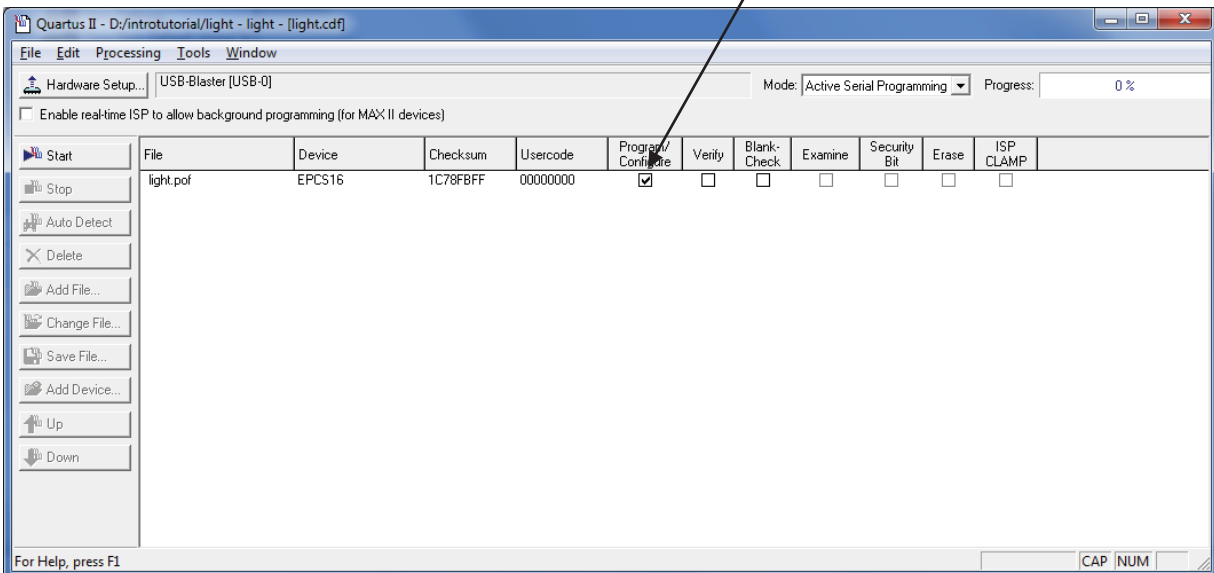
If you are changing the mode from the previously used **JTAG** mode, the pop-up box above will appear, asking if you want to clear all devices. Click **Yes**. Now, the Programmer window shown below will appear. Make sure that the Hardware Setup indicates the **USB-Blaster** and **Active Serial Programming**



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

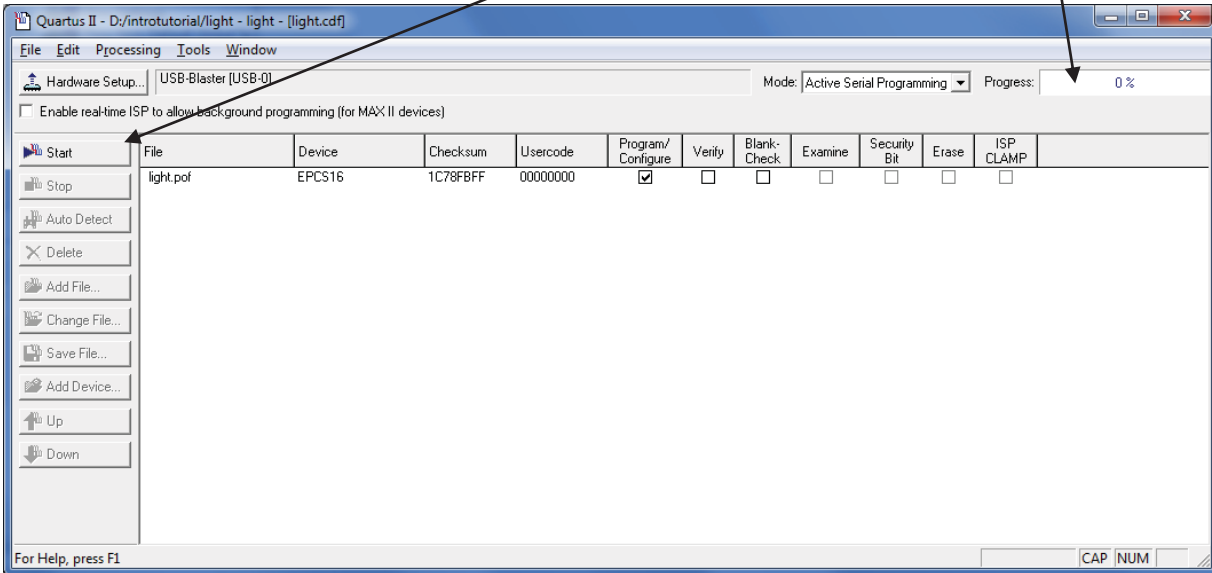


Select the file *light.pof* in the directory *introtutorial* and click **Open**. As a result, the configuration file *light.pof* will be listed in the window. Make sure the Program/Configure check box is ticked.



This is a binary file produced by the Compiler's Assembler module, which contains the data to be loaded into the **EPCS16** configuration device. The extension *.pof* stands for **Programmer Object File**.

Flip the **RUN/PROG** switch on the DE2 board to the **PROG** position. Press **Start** in the window. An LED on the board will light up when the configuration data has been downloaded successfully. Also, the **Progress** box will indicate when the configuration and programming process is completed. **Expect the programming to take over a minute (it may even appear to be doing nothing for up to 1 minute, but will then proceed rapidly).**



## 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  $x_1$  and  $x_2$ , by setting the corresponding states of the switches  $SW_1$  and  $SW_0$ . Verify that the circuit implements the truth table in Figure 12.

If you want to make changes in the designed circuit, first close the Programmer window. Then make the desired changes in the Block Diagram/Schematic file, compile the circuit, and program the board as explained above.