



## Familiar with Altera Quartus II Software

### 1. Getting Started:

To start the Quartus II software:

Click Right Button of the mouse > **Terminal** and type:

```
cd
cd^digital
source^.cshrc_linux
quartus&
```

(^ = space)

You should see a display similar to the one in Figure 1.1.

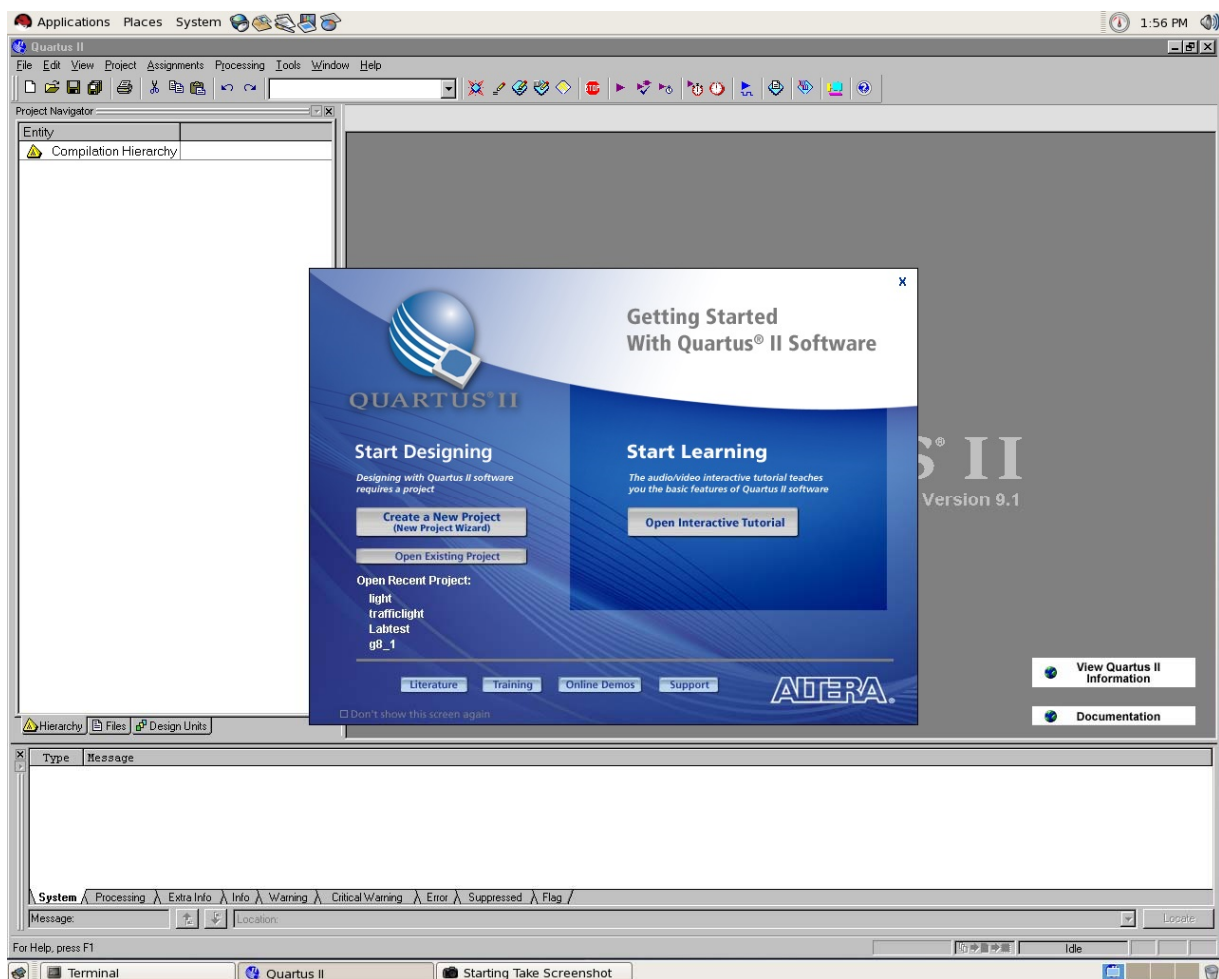


Figure 1.1 The main Quartus II display

## 2. Starting a New Project:

Every design being done with Quartus II software is called a *project*. To hold the design files for this lab, we will use a directory *lab1*. The example for this lab is a simple circuit for two-way light control.

Create a new project as follows:

1. Select **Creat a New Project (New Project Wizard)** or **File > New Project Wizard** to reach the window in Figure 1.2, which asks for the name and directory of the project.

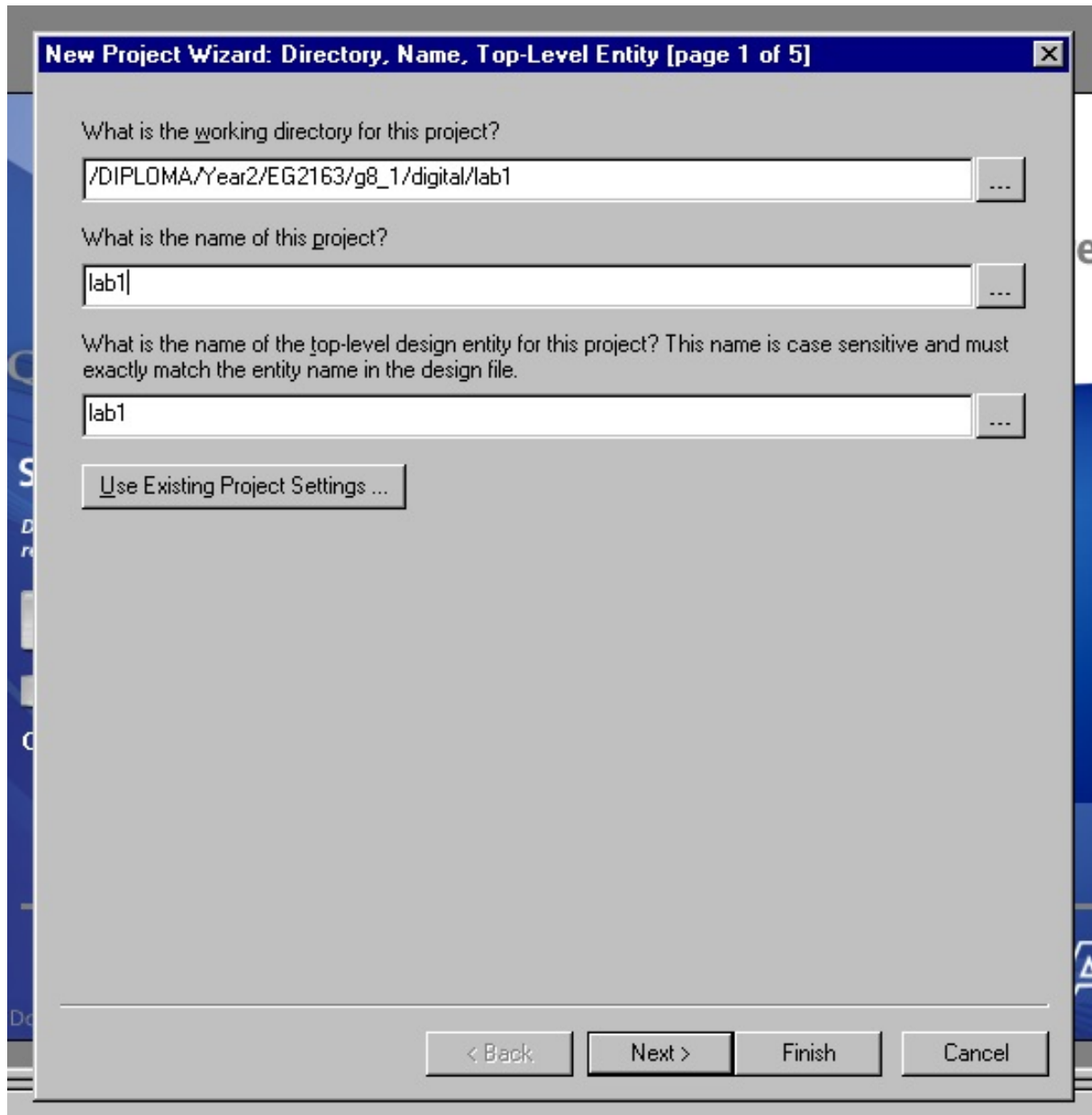


Figure 1.2 Creation of a new project

Set the working directory to be `.../digital/lab1`. 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 *lab1* as the name for both the project and the top-level entity, as shown in Figure 1.2. Click **Next**.

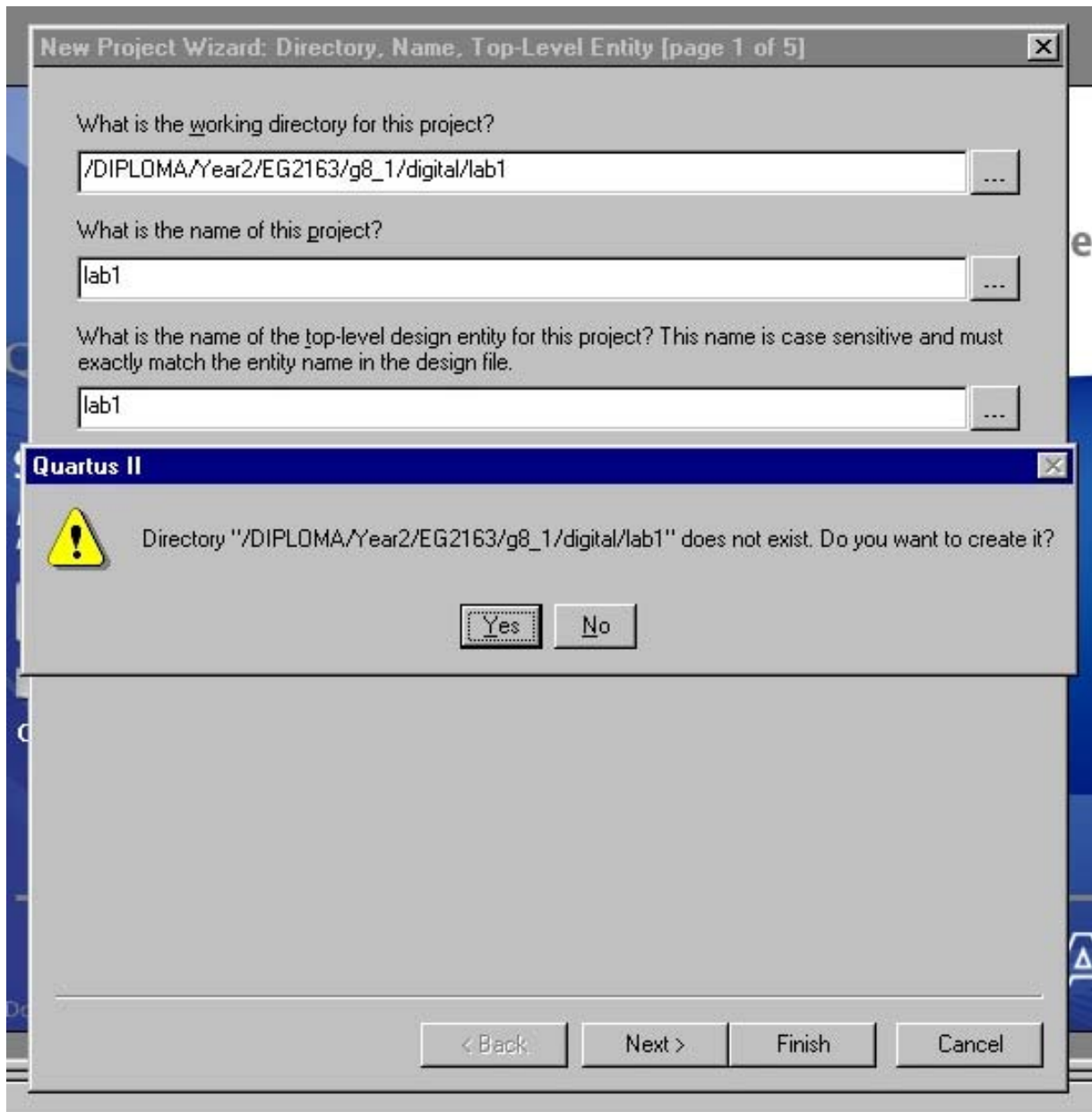


Figure 1.3 Creation of a new project

Click *Yes* to create the directory *lab1*.

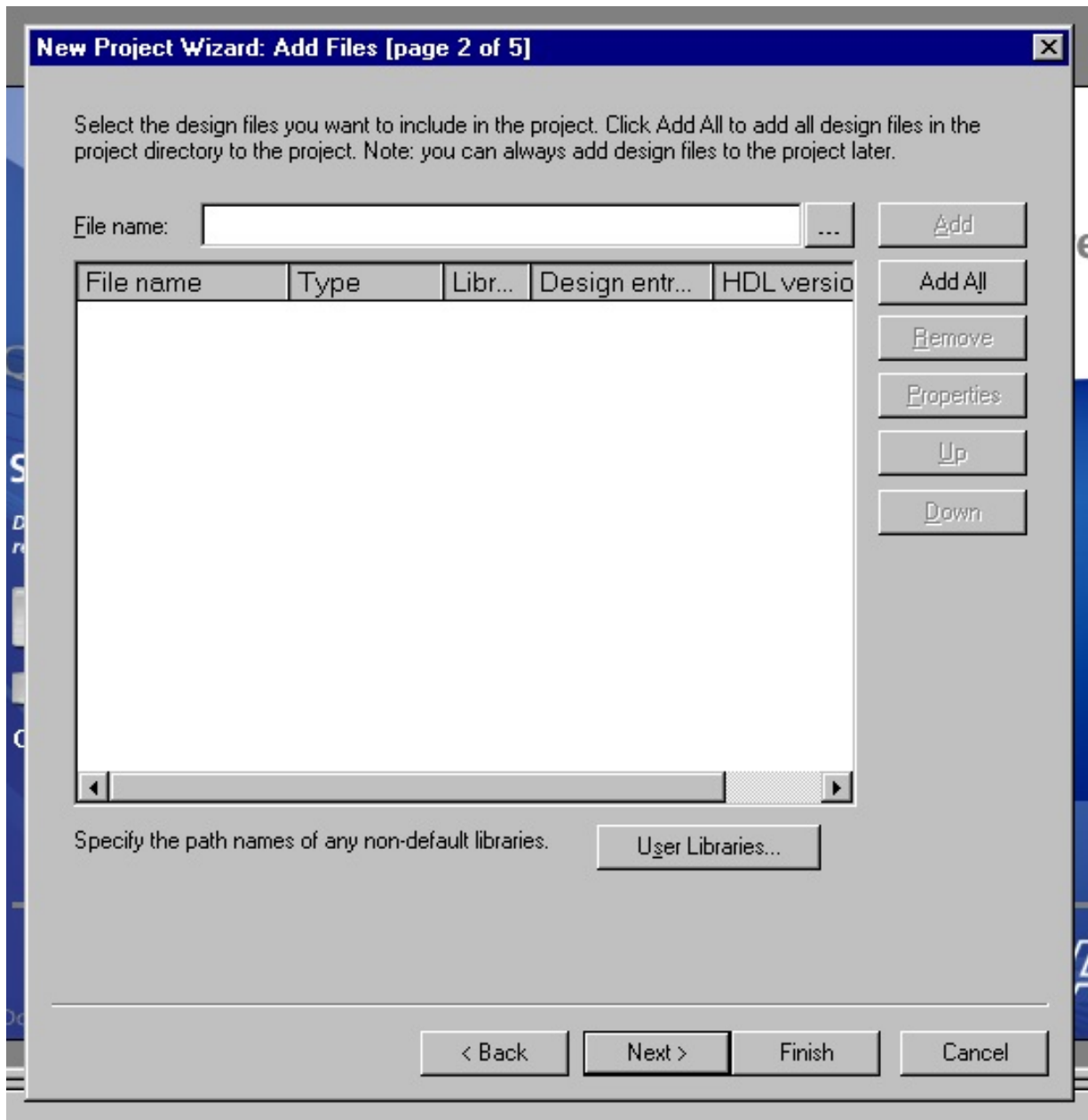


Figure 1.4 The wizard can include user-specified design files

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 1.5.

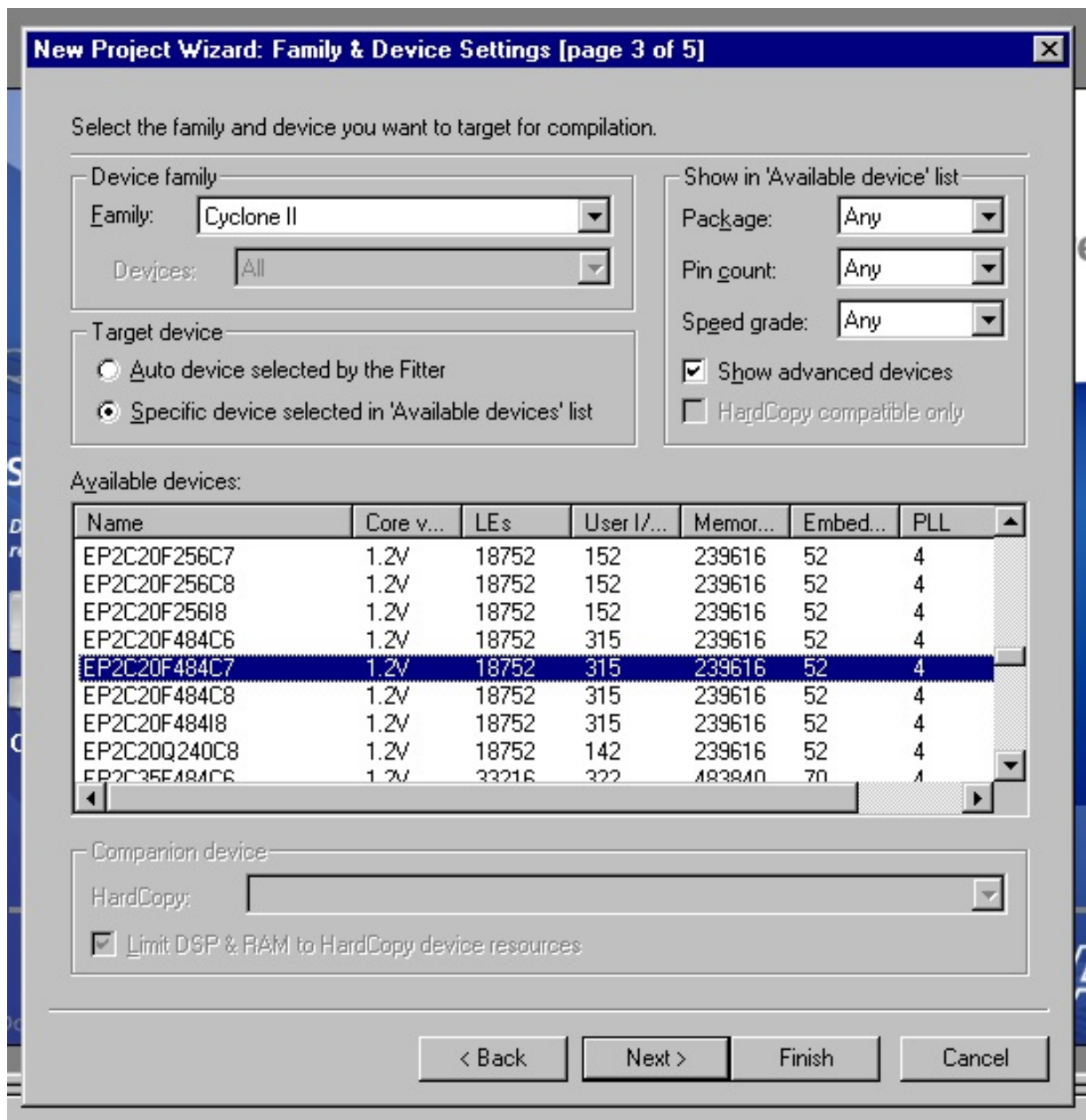


Figure 1.5 Choose the device family and a specific device

We have to specify the type of device in which the designed circuit will be implemented. Choose **Cyclone II** as the target device family. From the list of available devices, choose the device called **EP2C20F484C7** which is the FPGA used on Altera's DE1 board. Click **Next**, which opens the window in Figure 1.6.

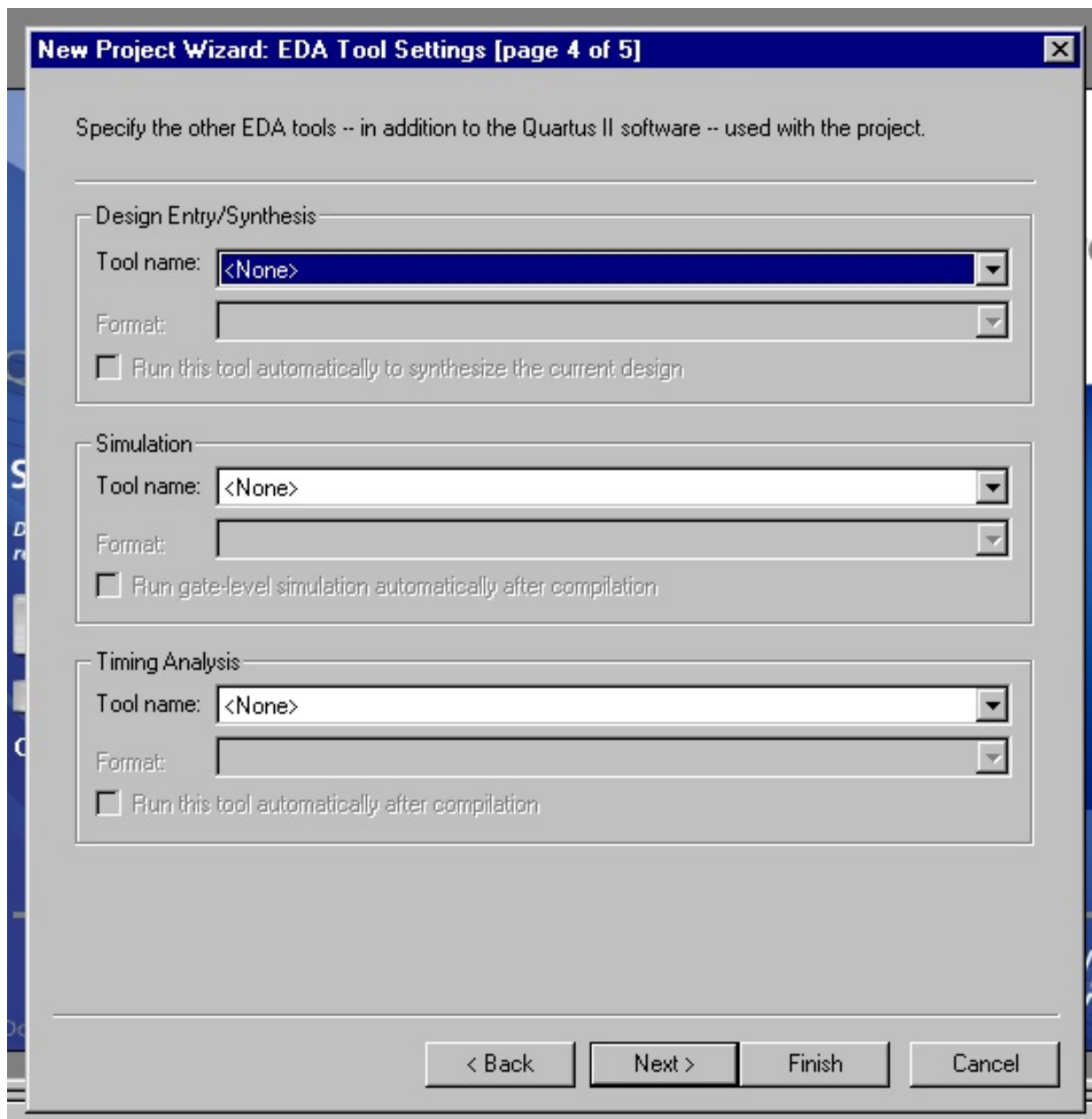


Figure 1.6 Other EDA tools can be specified

The user can specify any third-party tools that should be used. Since we will rely solely on Quartus II tools, we will not choose any other tools. Click *Next*. A summary of the chosen settings appears in the screen shown in Figure 1.7.

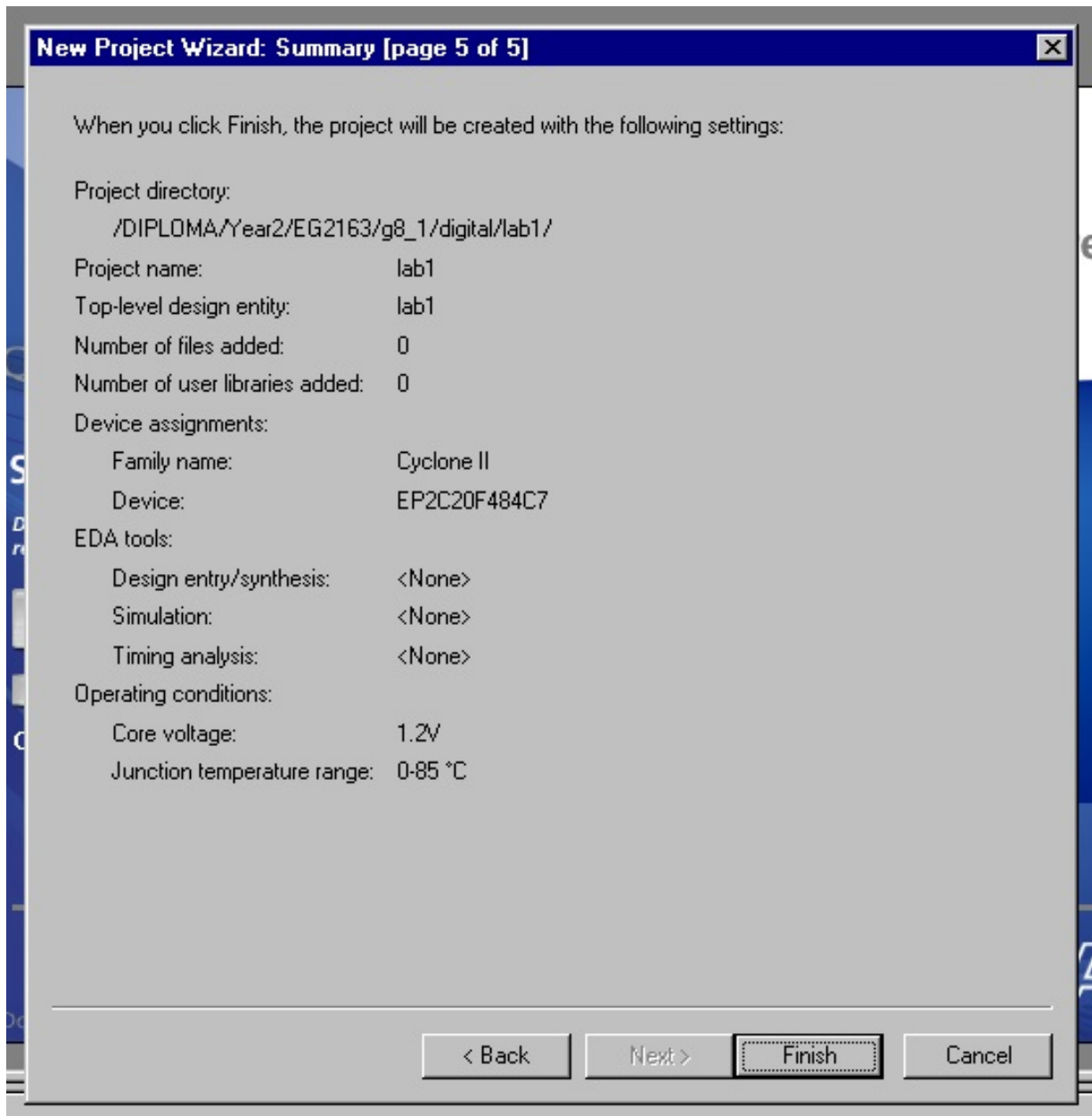
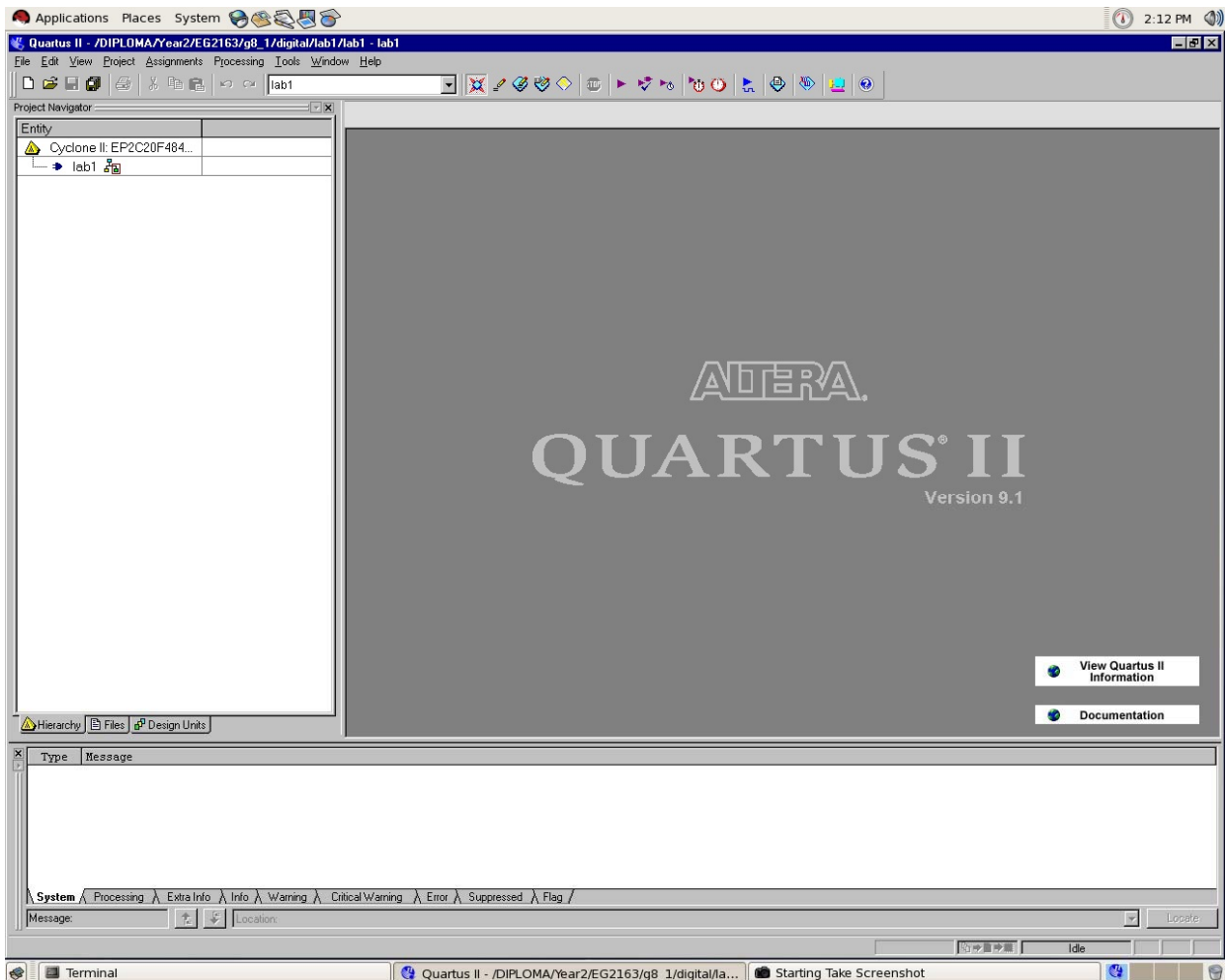


Figure 1.7 Summary of the project settings

Click **Finish**, which returns to the main Quartus II window, but with *lab1* specified as the new project, in the display title bar, as indicated in Figure 1.8.





Click **OK** if the below pop-up appear (ignore the pop-up).

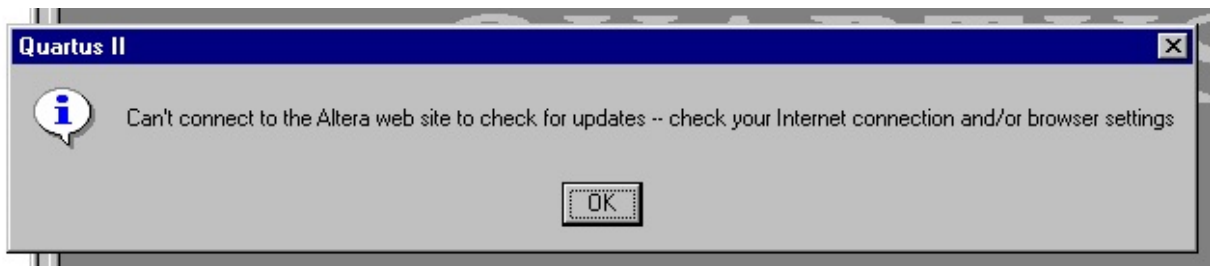


Figure 1.8 The Quartus II display for the created project

3. Schematic Design Entry:

As a design example, we will use the circuit shown in Figure 1.9.

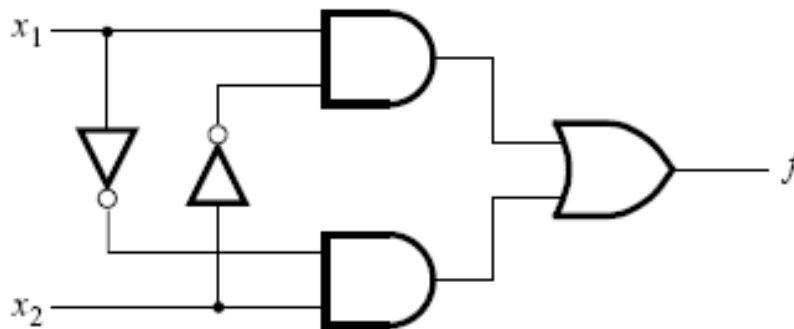


Figure 1.9 Design example

Select **File > New** to get the window in Figure 1.10, choose **Block Diagram/Schematic File**, and Click **OK**. This opens the Graphic Editor window.

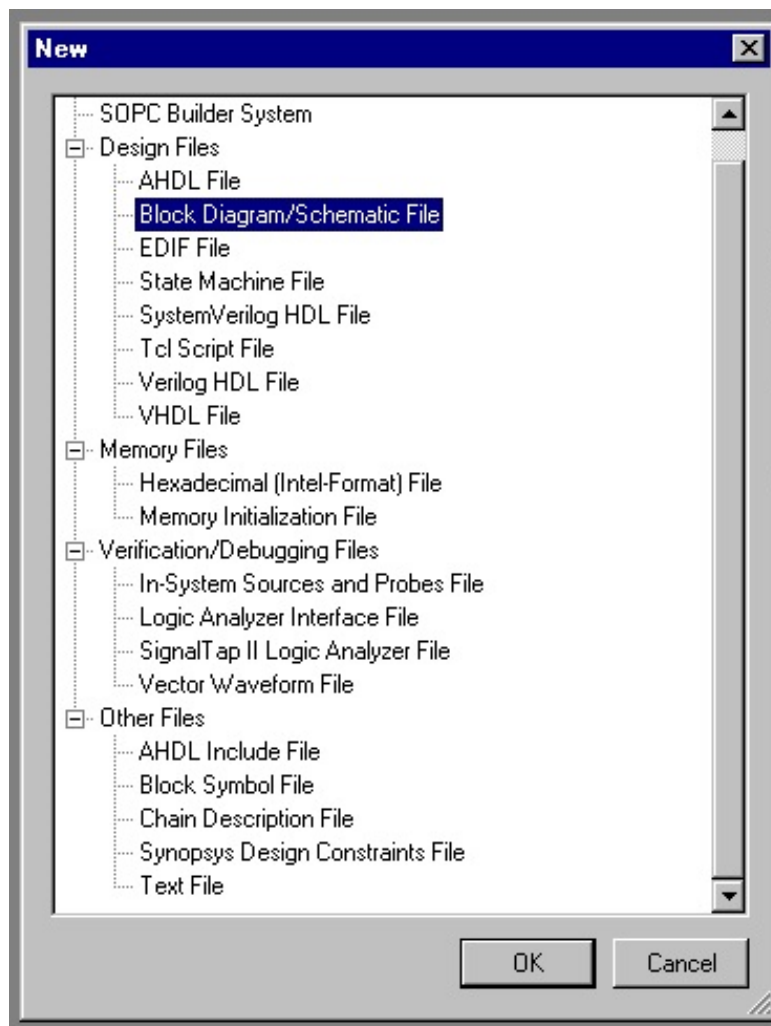


Figure 1.10 Choose to prepare a Block Diagram/Schematic File

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 1.11. In the box labeled (Save as type:) choose Block Diagram/Schematic File (\*.bdf). In the box labeled (File name:) type *lab1*, to match the name given in Figure 1.2, 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 *lab1* and leads to the Graphic Editor window displayed in Figure 1.12.

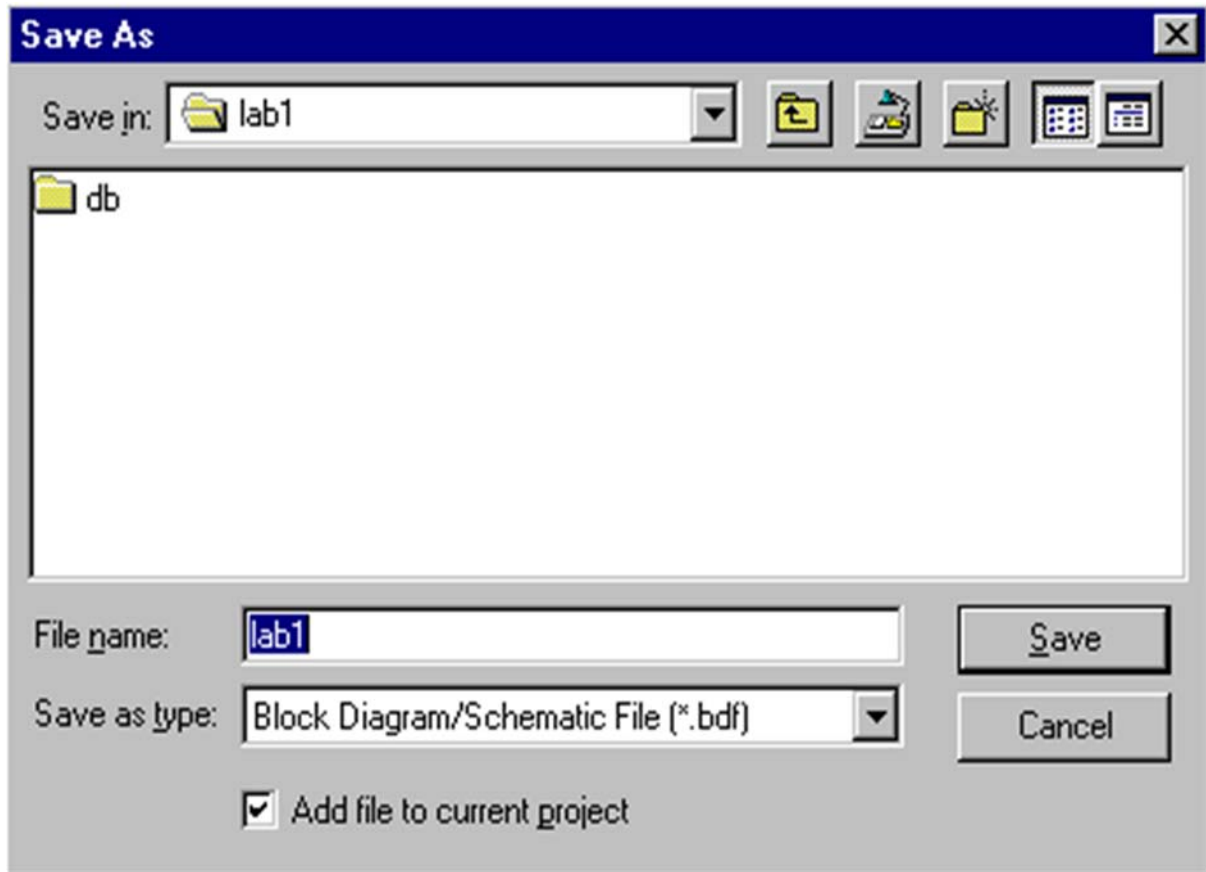


Figure 1.11 Specify a name for Block Diagram/Schematic File

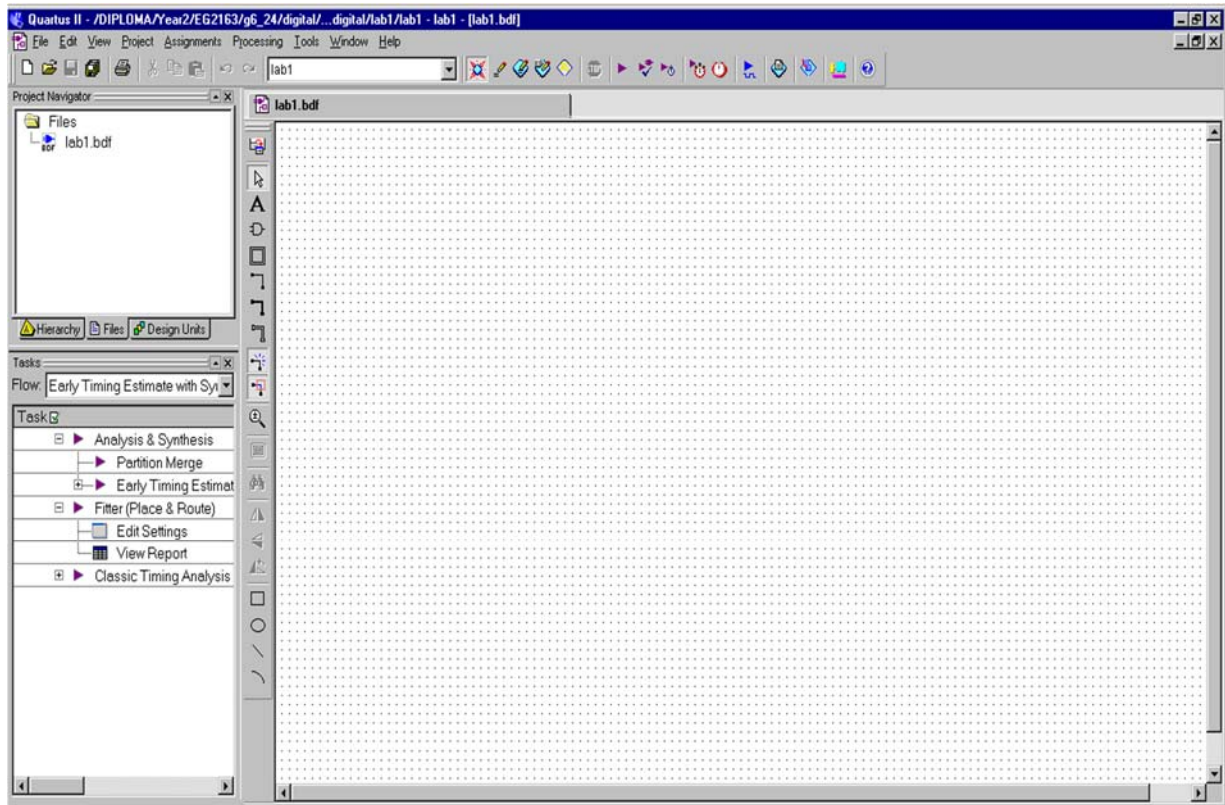



Figure 1.12 Graphic Editor window

To add a symbol: Double-click on the blank space in the Graphic Editor window, or Click on the Symbol Tool icon  in the toolbar. A pop-up box in Figure 1.13 will appear. Expand *libraries* > *primitives* > *logic* which comprises the logic gates. Select *and2*, which is a two-input AND gate. You can also type *and2* in the search bar to locate the symbol. Click **OK**, and now the AND gate symbol will appear in the Graphic Editor window.

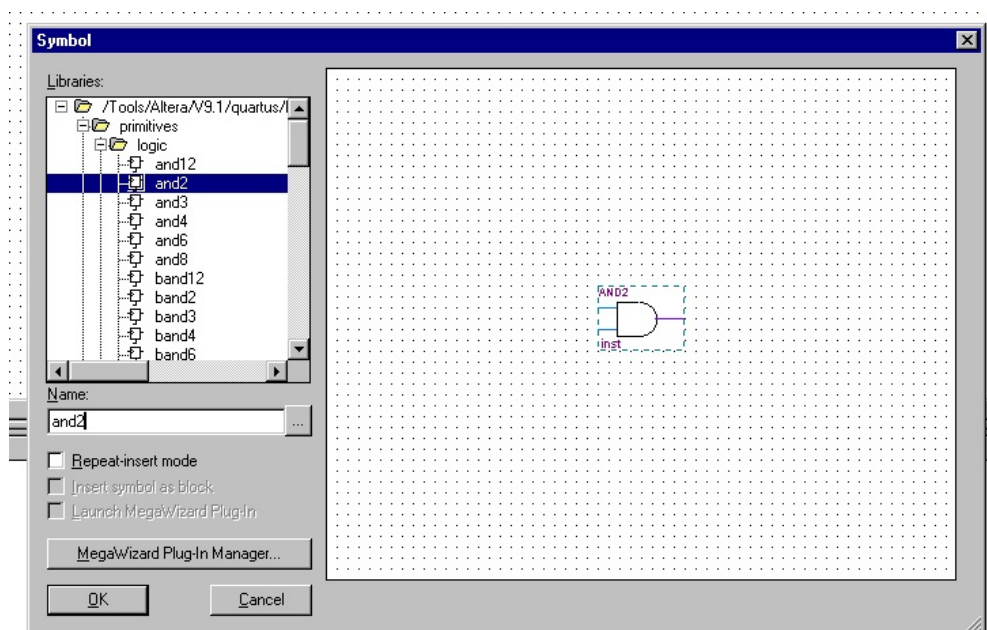



Figure 1.13 Choose a symbol from the library

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. 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 “Rotate left 90” icon . NOT gate must be selected first you can click the rotate icon. Arrange the gates as shown in Figure 1.14.

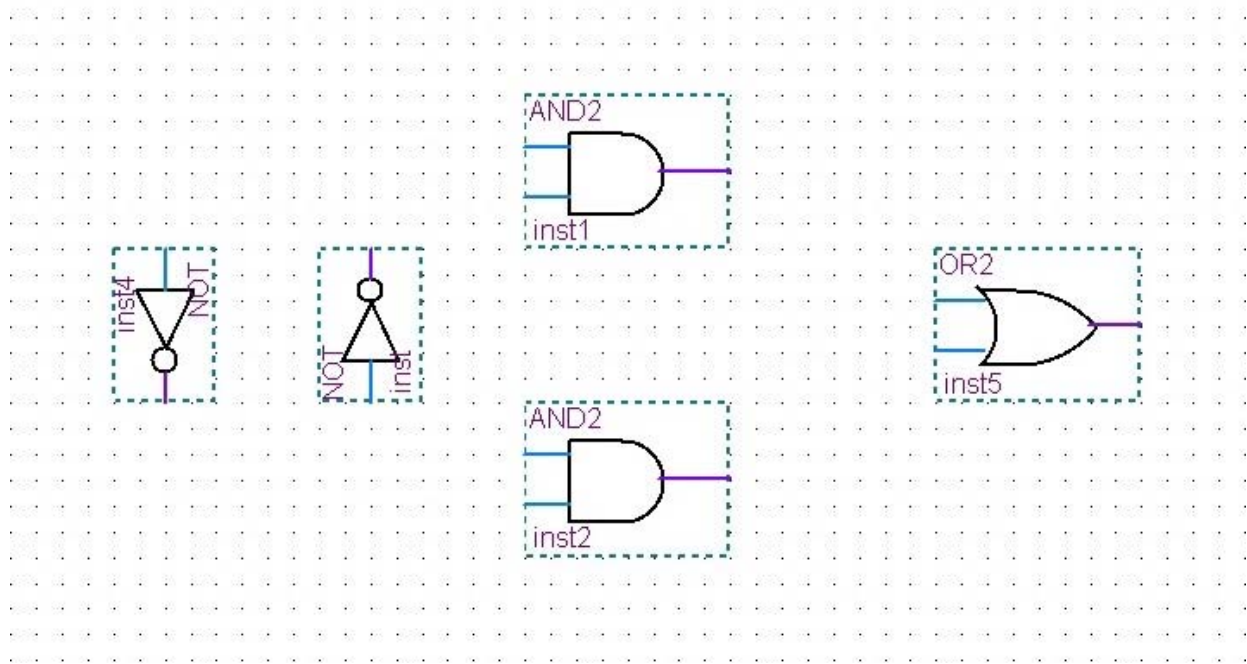


Figure 1.14 Import the gate symbols into the Graphic Editor window

Having entered the logic-gate symbols, it is now necessary to enter the input and output pins of the circuit. Use the same procedure as for importing the pins, but choose the pin symbols from *libraries > primitives > pin*. Import two instances of the input pin and one instance of the output pin, to obtain the circuit in Figure 1.15.

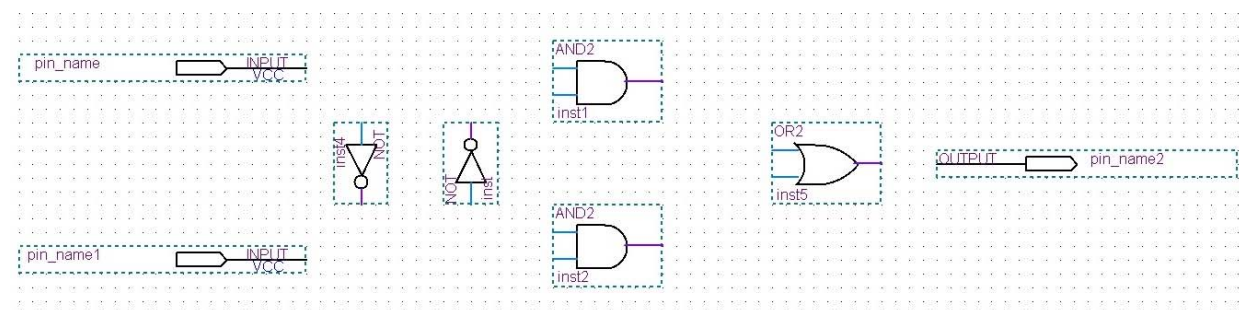


Figure 1.15 Import the input and output pins

Assign names to the input and output pins as follows. Point to the word *pin\_name* on the top input pin and double-click the mouse. The dialog box in Figure 1.16 will appear. Type the pin name, *x1*, and Click **OK**. Similarly, assign the name *x2* to the other input pin and *f* to the output pin.

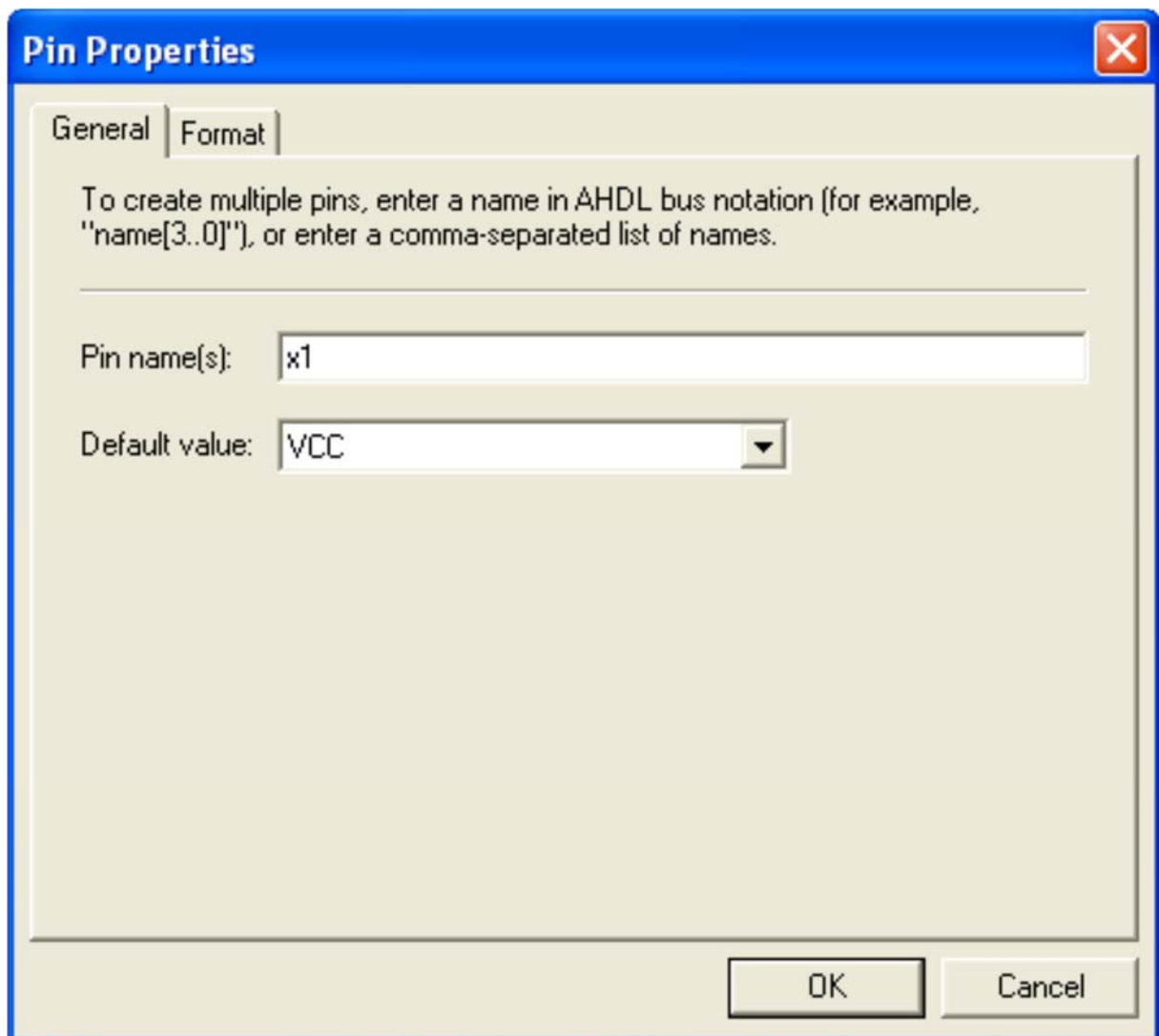




Figure 1.16 Naming of a pin

The symbols in the diagram have to be connected by drawing 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 wire reaches the pin stub on the top input of the AND gate. Release the mouse button, which leaves the wire connecting the two pin stubs. Use the same procedure to draw the remaining wires in the circuit. Note that a dot will appear indicating a connection between the two wires. 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 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 1.17; save it.

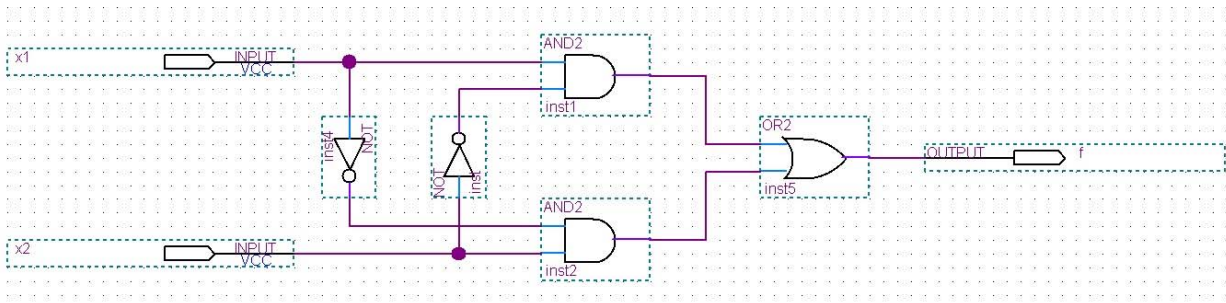



Figure 1.17 The completed schematic diagram

The entered schematic diagram file, *lab1.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. 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  that looks like a purple triangle.

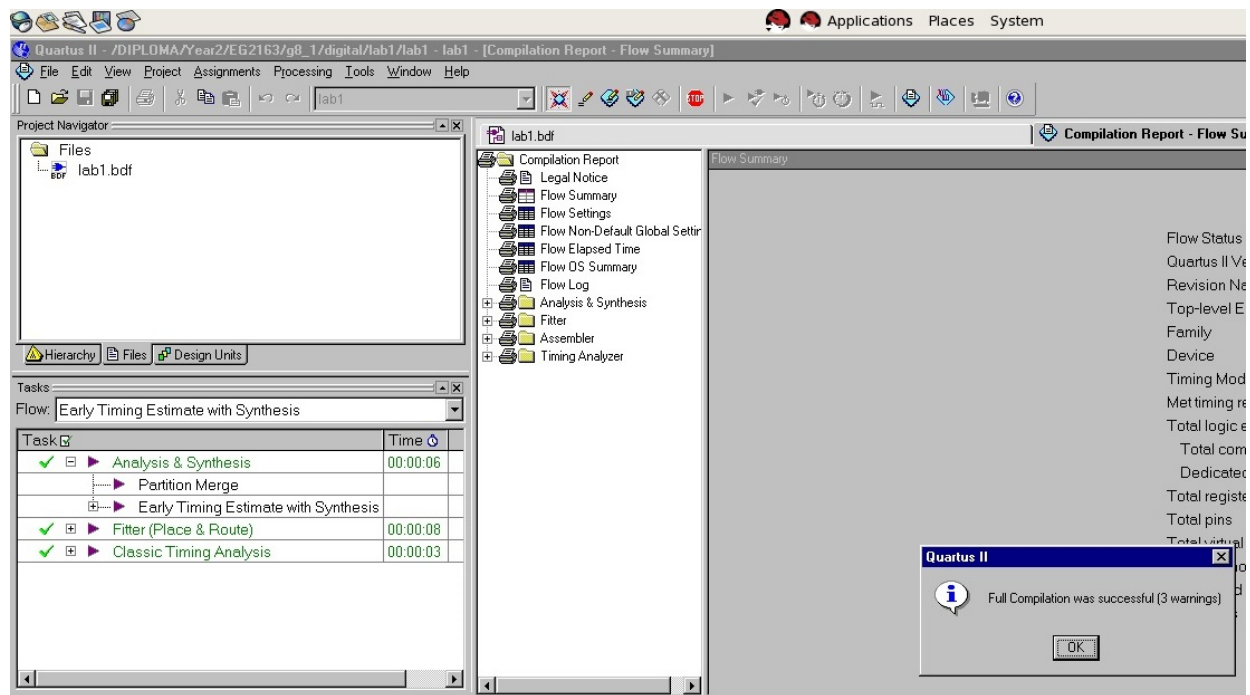


Figure 1.18 Display after a successful compilation

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. Successful (or unsuccessful) compilation is indicated in a pop-up box as shown in Figure 1.18. Acknowledge it by clicking **OK**.

#### 4. Simulating the Designed Circuit:

Before implementing the designed circuit in the FPGA chip on the DE1 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:

Open the Waveform Editor window by selecting **File > New**, choose **Vector Waveform File** as shown in Figure 1.19 and Click **OK**.

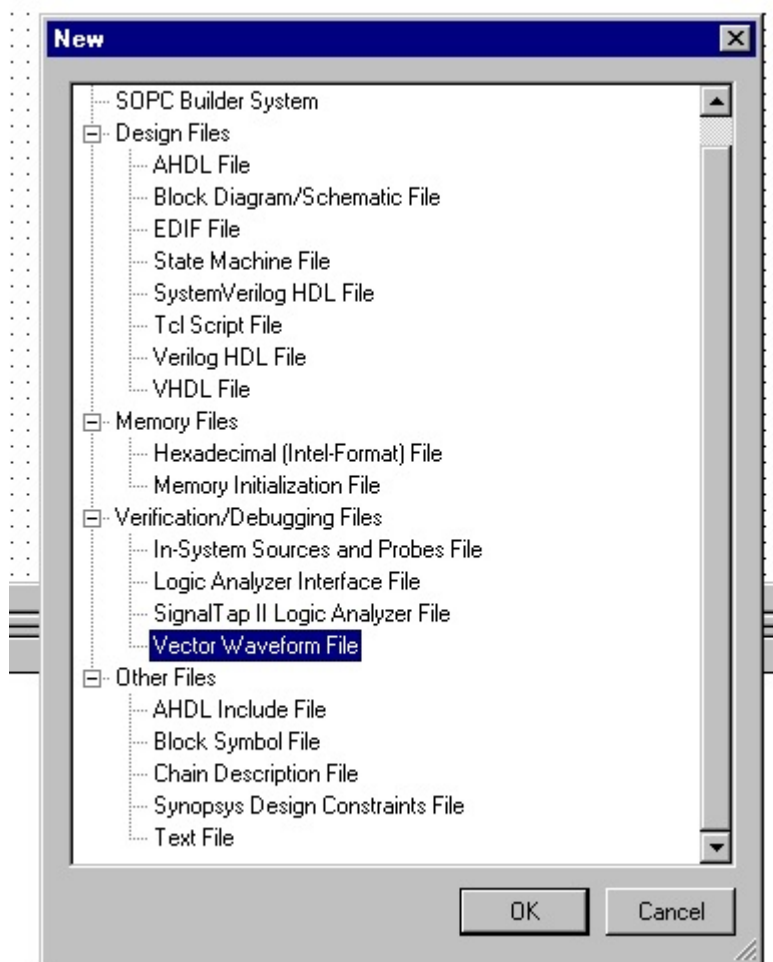


Figure 1.19 Choose to prepare a test vector file.

The Waveform Editor window is depicted in Figure 1.20. Save the file under the name *lab1.vwf*; note that this changes the name in the displayed window. Set the desired simulation to run from 0 to 200 ns by selecting **Edit > End Time** and entering **200 ns** in the dialog box that pops up. Next set the grid size to 50 ns by clicking **Edit > Grid Size** and entering **50 ns** in the dialog box that pops up. Select **View > Fit in Window** to display the entire simulation range of 0 to 200 ns in the window, as shown in Figure 1.20.



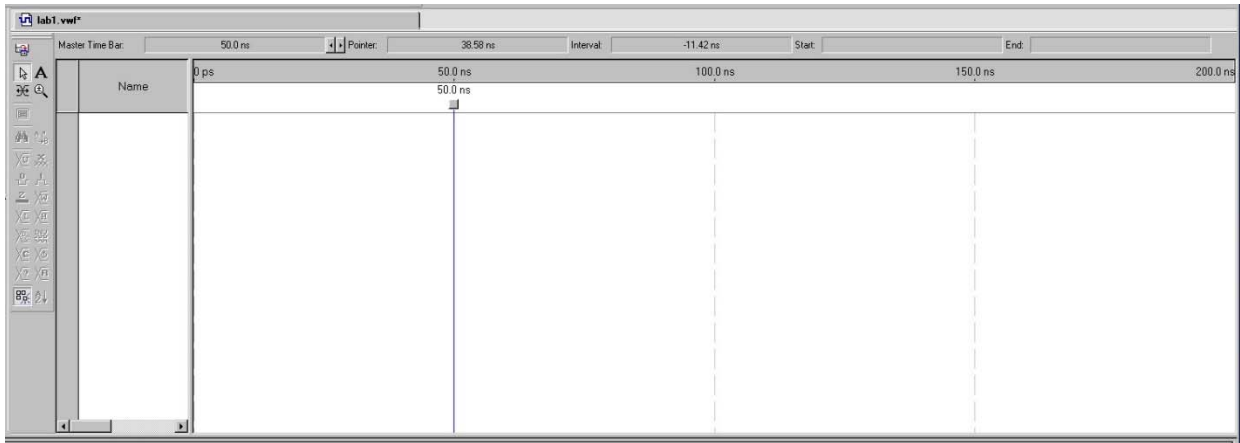


Figure 1.20 The Waveform Editor window

We want to include the input and output nodes of the circuit to be simulated. Click **Edit > Insert > Insert Node or Bus** to open the window in Figure 1.21. It is easier to click on the button labeled Node Finder to open the window in Figure 1.22. 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.

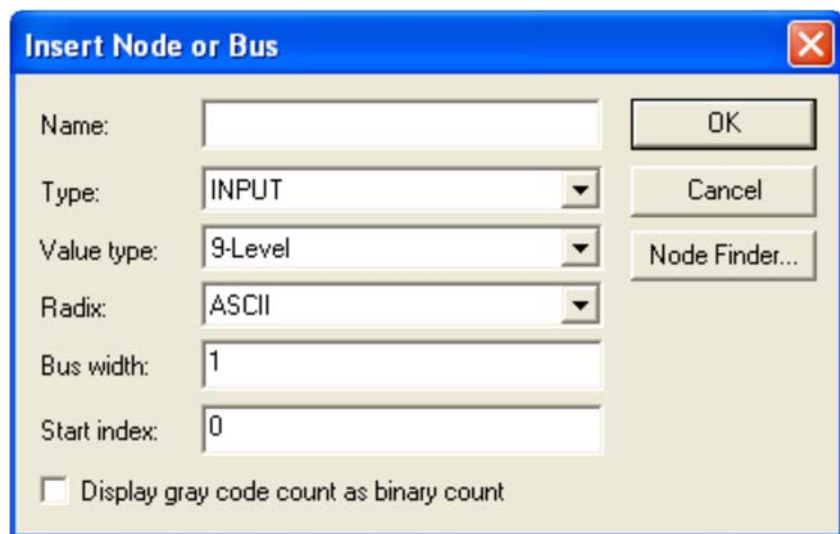


Figure 1.21 The Insert Node or Bus dialogue

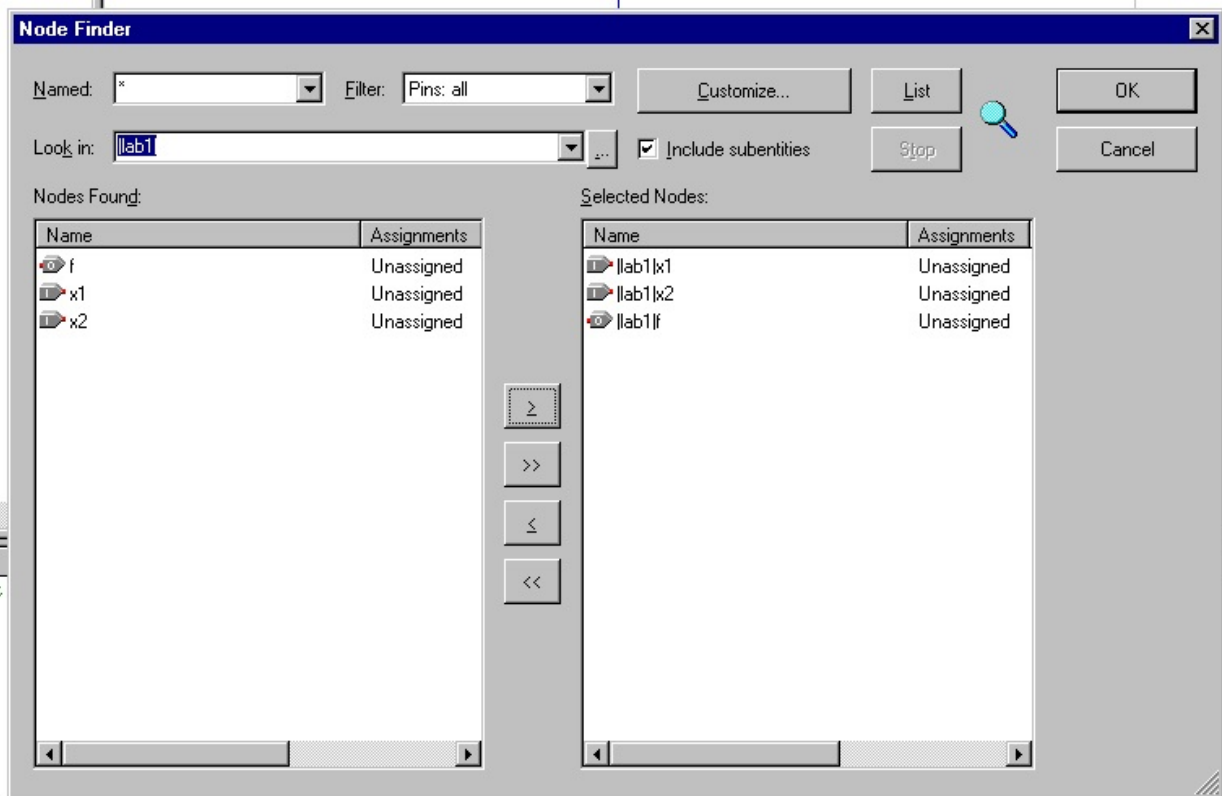


Figure 1.22 Selecting nodes to insert into the Waveform Editor

Click on the *x1* signal in the Nodes Found box in Figure 1.22, and then click the  $\geq$  sign to add it to the Selected Nodes box on the right side of the figure. Do the same for *x2* and *f*. You may remove the signal in the Selected Nodes box by selecting it and click the  $\leq$  sign. Click **OK** to close the *Node Finder* window, and then Click **OK** to close *Insert Node or Bus* window. This leaves a fully displayed Waveform Editor window, as shown in Figure 1.23. If you did not select the nodes in the same order as displayed in Figure 1.23, 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.

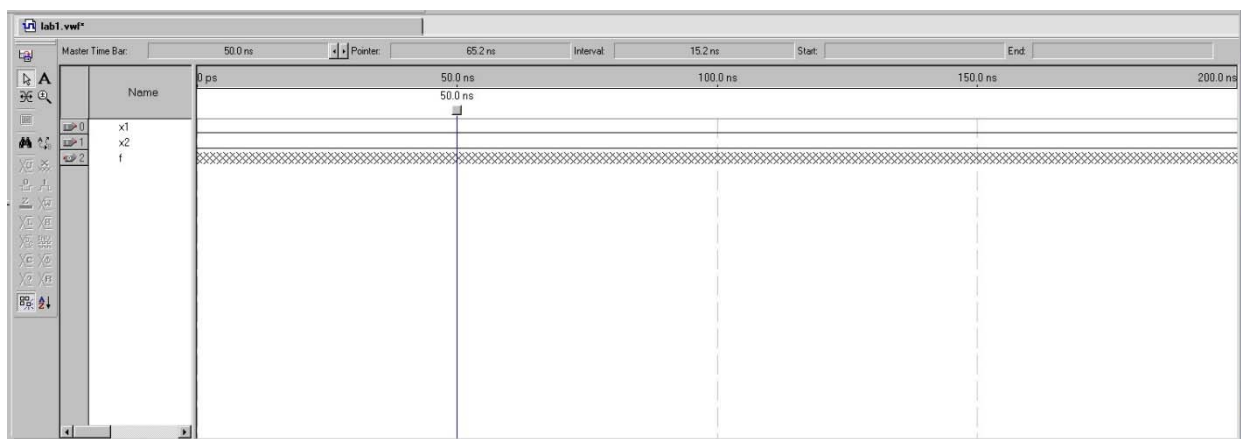




Figure 1.23 The nodes needed for simulation.

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. To make it easy to draw the desired waveforms, the Waveform Editor displays (by default) vertical guidelines and provides a drawing feature that snaps on these lines (which can otherwise be invoked by choosing **View > Snap to Grid**) or enable the snap to grid icon .

Observe also a solid vertical line, which can be moved by pointing to its top and dragging it horizontally. This reference line is used in analyzing the timing of a circuit; move it to the  $time = 0$  position.

For our tiny circuit we can simulate all four input valuations. We will use four 50 ns time intervals to apply the four test vectors. We can generate the desired input waveforms by using the **Waveform Editing Tool** . This should produce the image in Figure 1.24. 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.

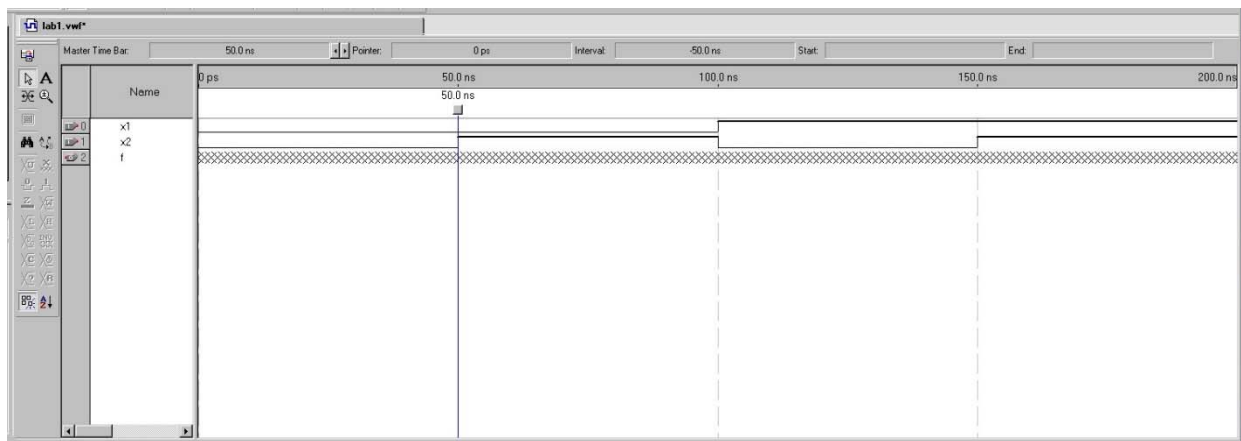



Figure 1.24 Setting of test values.

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, thus causing no 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. This takes much less time, because the simulation can be performed simply by using the logic expressions that define the circuit.

To perform the functional simulation, select **Assignments > Settings** to open the Settings window. On the left side of this window click on Simulator Settings to display the window in Figure 1.25, choose **Functional** as the simulation mode, and Click **OK**.

The Quartus II simulator takes the inputs and generates the outputs defined in the *lab1.vwf* file. Before running the functional simulation it is necessary to create the required netlist, which is done by selecting **Processing > Generate Functional Simulation Netlist**. A simulation run is started by **Processing > Start Simulation**, or by using the icon .

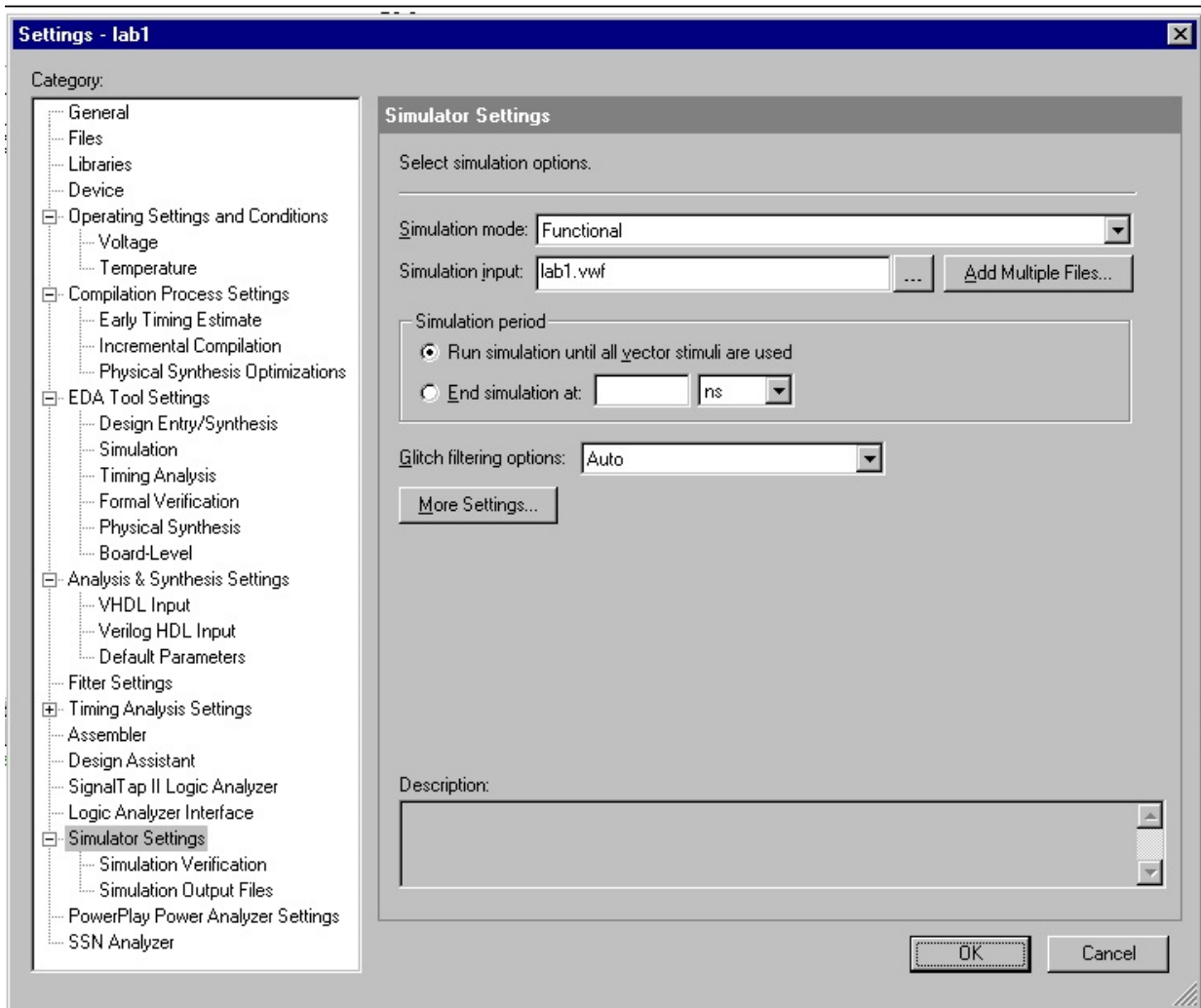


Figure 1.25 Specifying the simulation mode

At the end of the simulation, Quartus II software indicates its successful completion and displays a Simulation Report illustrated in Figure 1.26. 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 the output *f*.

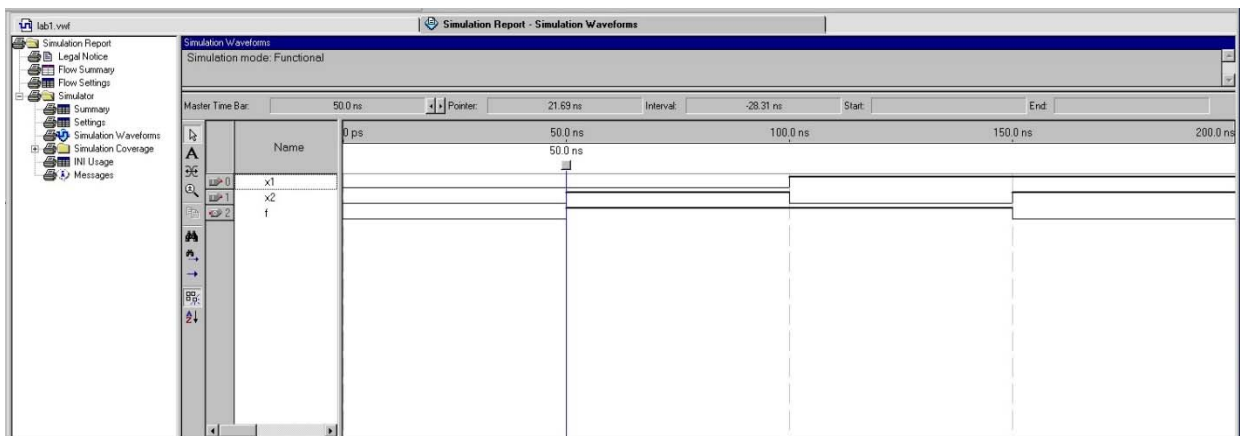


Figure 1.26 The result of functional simulation