Design Entry, Simulation, and Emulation using the Altera Quartus II® and ModelSim® CAD Software and the Terasic DE2-115 Rapid Prototyping Platform

B. Earl Wells and Dennis Hite

Introduction

Objectives:

The Fundamentals of Digital Design class (CPE 322) you will use the Altera Quartus II ® computer aided design software to create a schematic diagram or a Verilog representation of logic designs that you have created. You will also often use the integrated Mentor Graphics' Altera ModelSim® Starter Edition simulator package to aid you in understanding and verifying the functionality of your design. In the co-requisite Advanced Digital Design Lab (CPE 324) you will emulate your design by downloading it into a Field Programmable Gate Array, FPGA, that is present on the DE2-115 Rapid Prototyping Platform.

What is the purpose of this Altera Quartus II®/ModelSim® CAD Software?

The Altera Quartus II ® CAD development software is a fully integrated programmable logic design environment. This easy-to-use tool supports a large array of Altera® programmable device families and works in both the Microsoft Windows® PC and Linux environments. The tool supports design capture and has the Altera ModelSim Starter Edition® as a plug in. The ModelSim® software tool supports digital logic simulation and timing analysis.

How can I get access to the Software?

There are two options for getting accessing the software. One is to use the software that is already installed in the Advanced Logic Design Laboratory in Room 226 of the Engineering building. The software in this laboratory is the full commercial 13.0 Sp1 version.
There is also a web version of Quartus II that you can download and run on your own PC. You can download it from the Altera web site at

http://dl.altera.com/13.0sp1/?edition=web

Then make sure you have chosen Version 13.0 Sp1 and chose your operating system. Then select your download method making sure that the Quartus II, ModelSim-Altera Editions, and Devices

Cyclone IV E are checked which should be included on the defaults. The Web Edition software contains the same basic functionality as the commercial version but it is limited in terms of the devices that are supported and the size of your designs. This option is mentioned only for your convenience and UAH is not liable for any activities concerning the use of the software that you download.

Getting Started

There are many Quartus® II file types, three of the most common types include:

Block Description Files (bdf) = schematic or block diagram
Block Symbols File (bsf) = symbol or schematic component
Hardware Description File (tdf, v, vhd) = AHDL, Verilog, or VHDL, -- alternative higher level ways of capturing your design.

To complete the simulation exercises in this course, you will be concerned mainly with creating Block Description Files (bdf) and Verilog Hardware Description Files (v). Both type of files can appear in the same design. A bdf file is where you will place your logic schematic that represents your design. A Verilog file is where you place your textural description of your design that is written in the Verilog hardware description language. A given design can contain a hierarchy of block description and Verilog files.

After you have created your design you will need to compile it. At this point the design can be simulated to determine its functionality. This can be done using the same computer that you entered it via a schematic/block diagram and or hardware description language. Simulation requires that you drive the inputs of your design in some predetermined and meaningful manner at specified points in time. This is called providing an input *stimulus* to the simulated design. It also requires that you select and view the resulting output that you want to observe. The simulation program will produce an output that should mimic the behavior of the actual design. This is then compared with the expected output. Simulation outputs can be viewed as a timing diagram or in a textural manner. The stimulus can be provided by creating a testbench element of your design that provides the inputs and outputs to your design (this is often expressed using a hardware description language such as Verilog) or through stimulus generation commands that are part of the ModelSim® Simulation Environment.

I. Setting up the work environment

This section describes the steps necessary to utilize the Quartus II/ModelSim® environment within the UAH ECE Department's EB 226 Advanced Logic Design lab. Individual student implementations should work in a similar manner. *Throughout this tutorial it is assumed that Windows ® maps your flash drive to the device labeled* ***e:****. If this is not the case you should substitute the actual drive letter for the* ***e:*** *labeling presented here.*

2 Before entering the Quartus II ® CAD software place your personal USB flash drive in the computers USB port. This is where you are to store your design files for the labs in this course. It is suggested that you use a USB flash thumb drive that has a capacity of 512 Mbytes or greater. Make sure that you are careful with this drive and that you exit the Quartus II ® software and properly stop the drive before you remove it from the PC or you may have loss of data. It is the student's responsibility to properly backup his/her flash drive. Do not put your personal files on the disk drive of the laboratory PCs.

3 To open the Quartus II ® CAD tool package, first double click with the left mouse button on the Altera Quartus II ® icon 

on your desktop. Make sure to select version 13.0 or higher. Note that the exact version number of Quartus II may not match the one shown here but this tutorial should still be valid for your version of the software. You will then see a welcome screen that is similar to that in Figure 1.1.

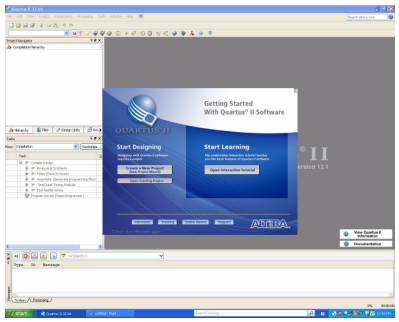


Figure 1.1

4 Next left-click on the *New Project Wizard* which is under the *File* Menu.

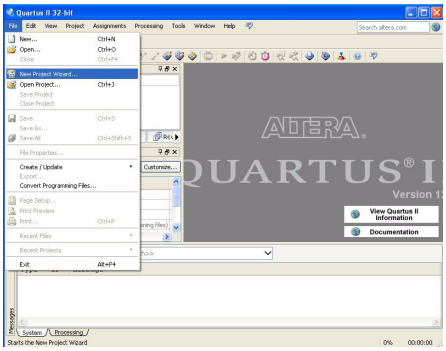


Figure 1.2

5 An information window similar to the one shown in Figure 1.3 will appear, left-click on *Next* to continue.

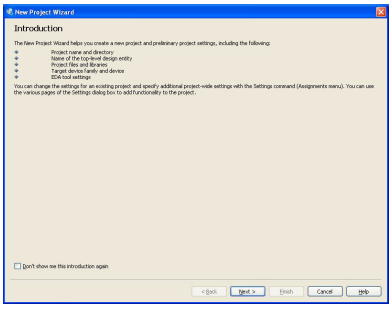


Figure 1.3

6 A dialog box similar to the one shown in Figure 1.4 will then appear. In the first field of it enter the name of the working directory such as **e:\example1** as shown in the figure. Then enter the name of the project. The project contains all the pertinent information about the design. In this case enter the name **comb_logic** in the following field. Notice that as you enter this name it also is being entered in the bottom field where will be used to indicate your top-level design file. The Quartus II tool supports hierarchical design methodologies that allows the user to use many files to represent the entire design. In this example we will only use a single top-level design file which will have the same name as the project. This is the default so simply left-click on *Next* after you have entered the name of the project in the second field

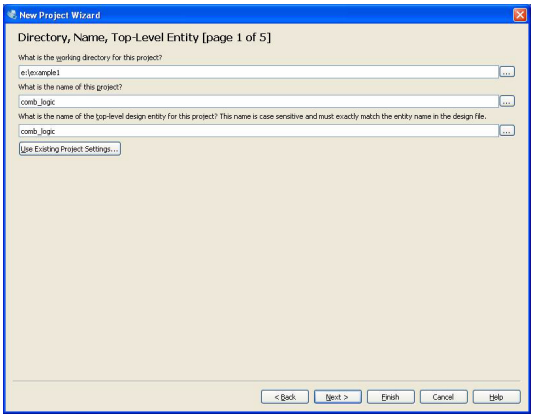


Figure 1.4

7 If the working folder (**e:\example1** in this example) is not present then a warning message window will appear asking you if you want to create the file. In this case simply left-click on

the *Yes* button to continue. The next *New Project Wizard* window will then give you the opportunity to add more files to the project (as shown in Figure 1.5).

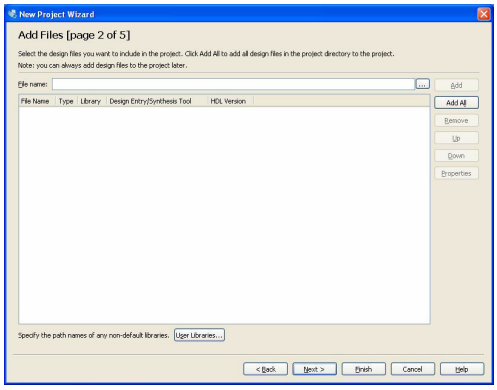


Figure 1.5

You do not want to do that at this time because you have not yet created the schematic file (you have no files to add). Simply left-click on *Next* button to continue to the next screen.

8 The *New Project Wizard* will then display the *Family & Device Setting* window which will allow you to select a programmable logic device to configure with your design (i.e. load your design into). You are to select the Cyclone IV E family, and the EP4CE115F29C7 device as shown in Figure 1.6.

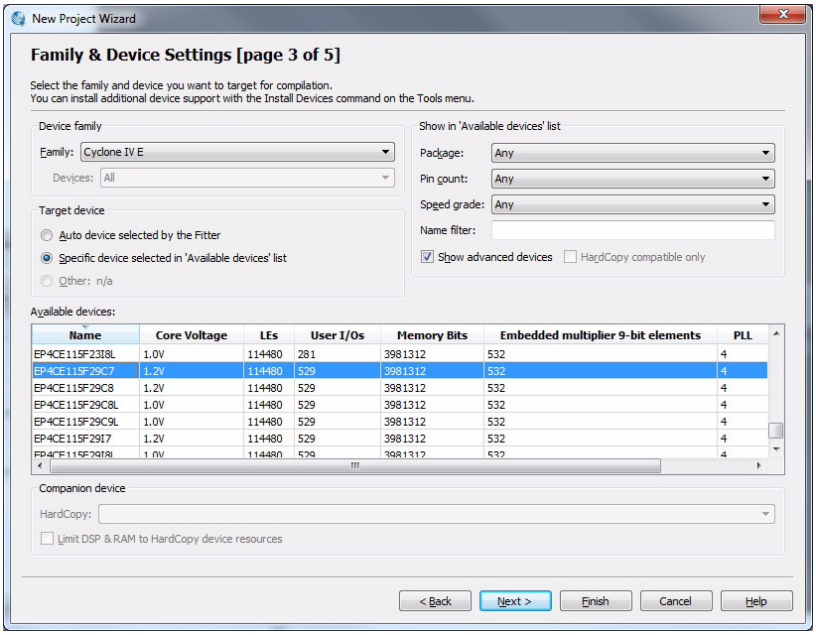


Figure 1.6

After the *Family*, and *Target Device* is selected you should left-click on the *Next* button which should advance you to the *EDA Tool Settings* window as shown in Figure 1.7.

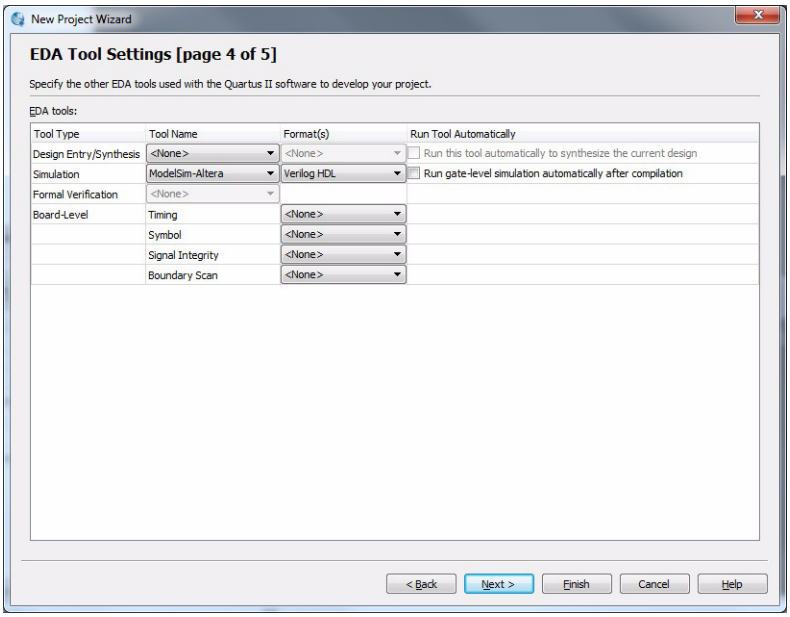


Figure 1.7

9 The *EDA Tool Settings* window is where you can specify the configuration of the "plug-in" type CAD tools you would like to use as part of your design flow. In this case we would like to employ the customized version of ModelSim® that incorporates the Cyclone IV E Altera library and you would like to specify that this tool utilize Verilog as its simulation environment. This requires that you select the **ModelSim-Altera** option on the *Simulation* row as the *Tool Name* and on this same row then select the **Verilog HDL** option as shown in Figure 1.7. After this is done left-click on the *Next* button to bring up the *Summary* window as shown in Figure 1.8 which is the final *New Project Wizard* window.

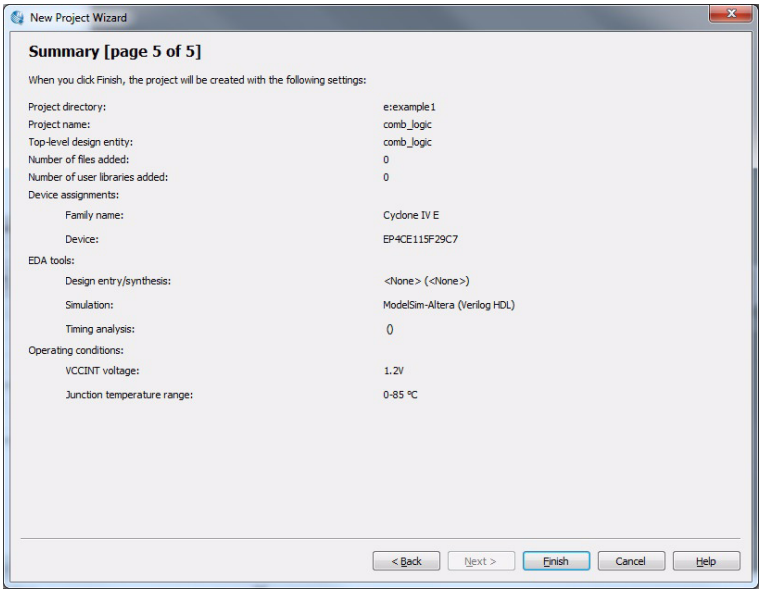


Figure 1.8

10 This highlights the major options which you chose previously for this design including the project directory, project name, FPGA device number, and the EDA tools setup. If this is correct left-click on the *Finish* button, otherwise click on the *Back* button to make the necessary corrections.

II. Design Entry

Part A: Creating a Schematic or Block Diagram Representation of your Design

Entering the Block Editor

1 After you have exited the *New Project Wizard*, may choose to enter your design as a schematic diagram. To do this, first select *New* from the *File* menu (or click on the upper left icon on the top tool bar of the Quartus II window). This will bring up the file type box that is shown in Figure 2.1. Then choose the *Block Diagram/Schematic File* option and press the *OK* button.

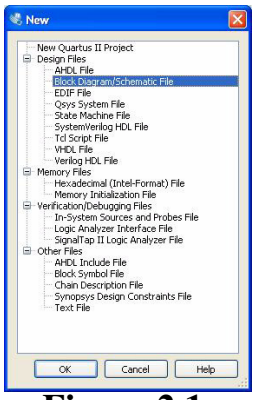


Figure 2.1

2 This will bring up the *Block Editor* which is shown in Figure 2.2. On the right hand side of the block editor is the location of the graphics window work area where you will create your schematic. The name of this graphics window is tentatively set by the software as **Block1.bdf** but this will be renamed to the name of your project when you save your schematic the first time. Double-click with the left mouse button somewhere in the *Block Editor's* work area or left-click on the icon to select a given component to enter into your design (you can also select the *Insert Symbol* option from the *Edit* Menu).

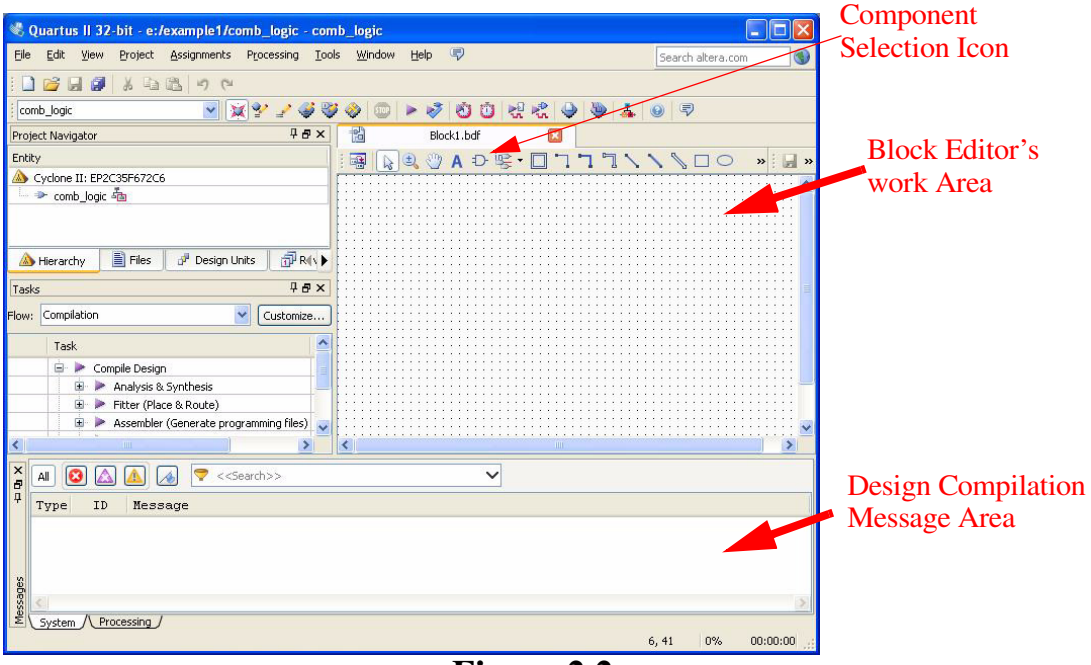


Figure 2.2

Entering Components

1 Before you can enter a component, you will need to choose it from the appropriate component library of symbols. In the previous step you brought up the *Symbol Window* which has a list of such library components in the left hand side of the window. To reveal the components expand upon the *c:/altera/.../quartus/libraries* entry by left-clicking on the ⊞ symbol next to it. This will reveal the ***megafunctions***, ***other***, and ***primitives*** component library symbol elements where:

megafunctions: (larger customizable parameterizable logic components such as multipliers, adders, decoders, multiplexers, counters, and special i/o components)
others: (7400 equivalents, SSI, legacy MSI components)
primitives: (gates, flip-flops, I/O pins, VCC, GND symbols, etc.

To enter the **comb_logic** example choose a 2-input AND gate from the ***primitives*** library. You can do this by expanding the ***primitives*** library to reveal the ***logic*** sub-library as shown in Figure 2.3 (i.e. select *c:\altera\...\quartus\libraries\primitives\logic*). Once the *and2* component is selected left-click the *OK* button and then left-click the place in the *Block Editor* graphics window work area where you would like to put the component as shown in Figure 2.4.

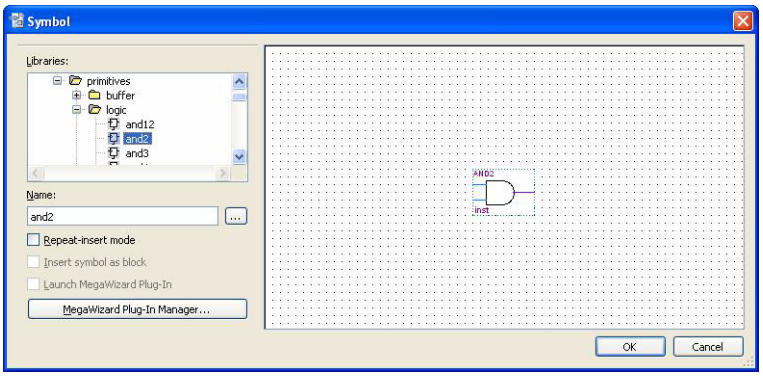


Figure 2.3

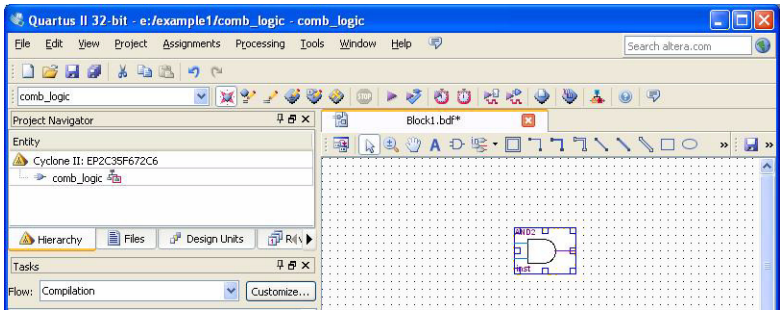


Figure 2.4

2 In the **comb_logic** example this design is a two-level logic schematic composed of three 2-input AND gates that all feed into a single 3-input OR gate. Once you have selected the first AND gate you can cut-and-paste a copy in the normal windows manner. First, using your mouse, right-click on the *and2* gate that you just added then scroll down and then right-click on **Copy** as shown in Figure 2.5.

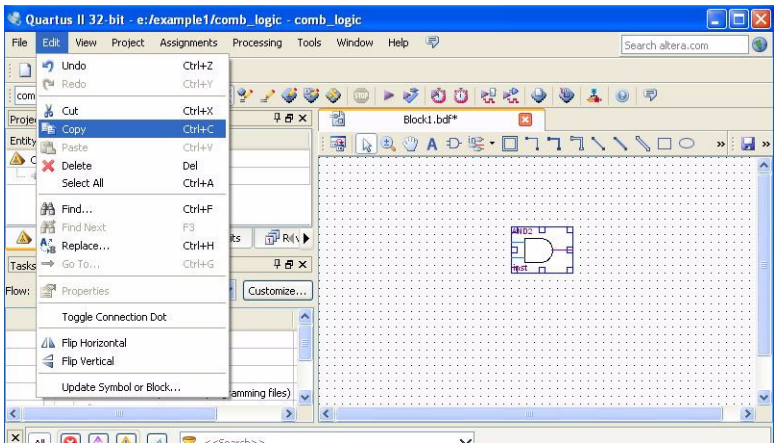


Figure 2.5

3 Then choose a position below the original *and2* gate, right-click, then scroll down and left-click on **Paste** as shown in Figures 2.6-2.7 (*ctrl+x* and *ctrl+v* also work).

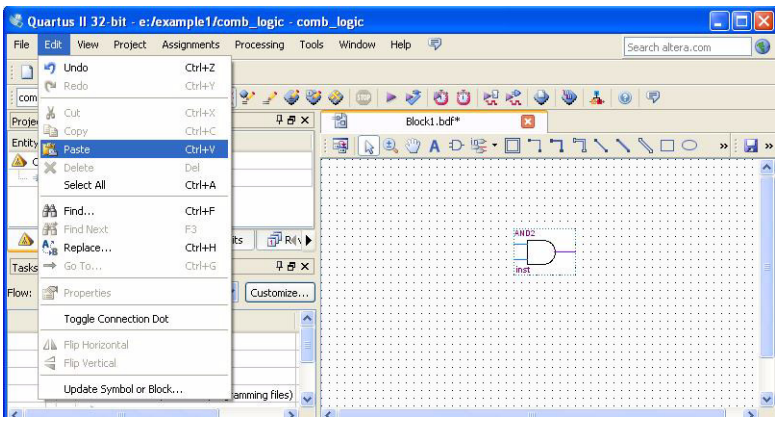


Figure 2.6

4

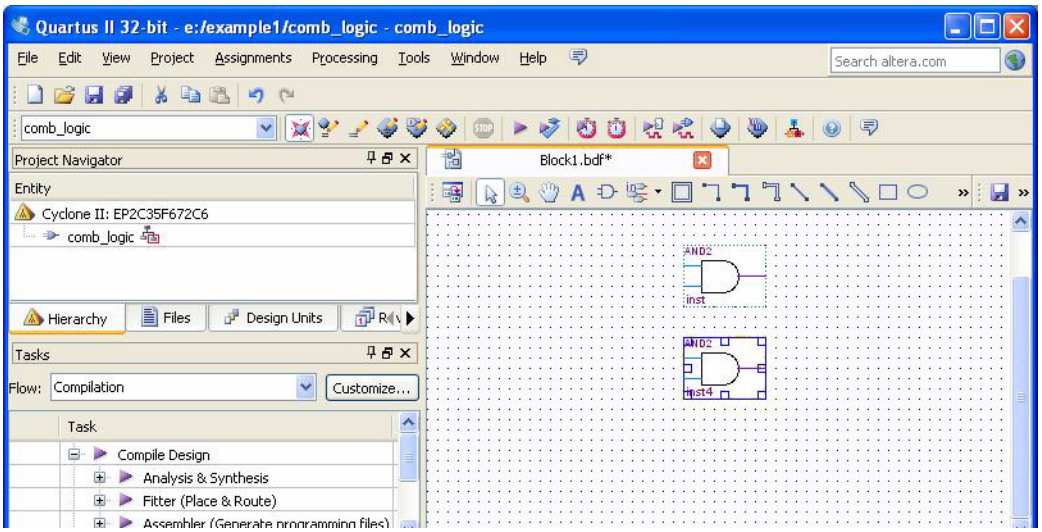


Figure 2.7

5 Now repeat the procedure to add a third *and2* gate. You can move components by selecting the component as you hold down the left mouse button and drag the symbol to where you want it to go.

6 Now add a 3-input OR gate (*or3*) to the right of the second (middle) *and2* gate. Use the same procedure you used to select and add your first *and2* gate.

7 Next, wire the outputs of all three *and2* gates to the inputs of the *or3* gate. To add the first wire, point to the output pin of the first *and2* gate, click the left mouse button and hold it down while dragging the wire (line) toward the top input of the *or3* gate. Once the wire is touching the *or3* input pin release the mouse button. Make sure that there are no visible gaps in the wire. Figure 2.8 shows how your design should look after wiring the first *and2* gate to the *or3* gate.

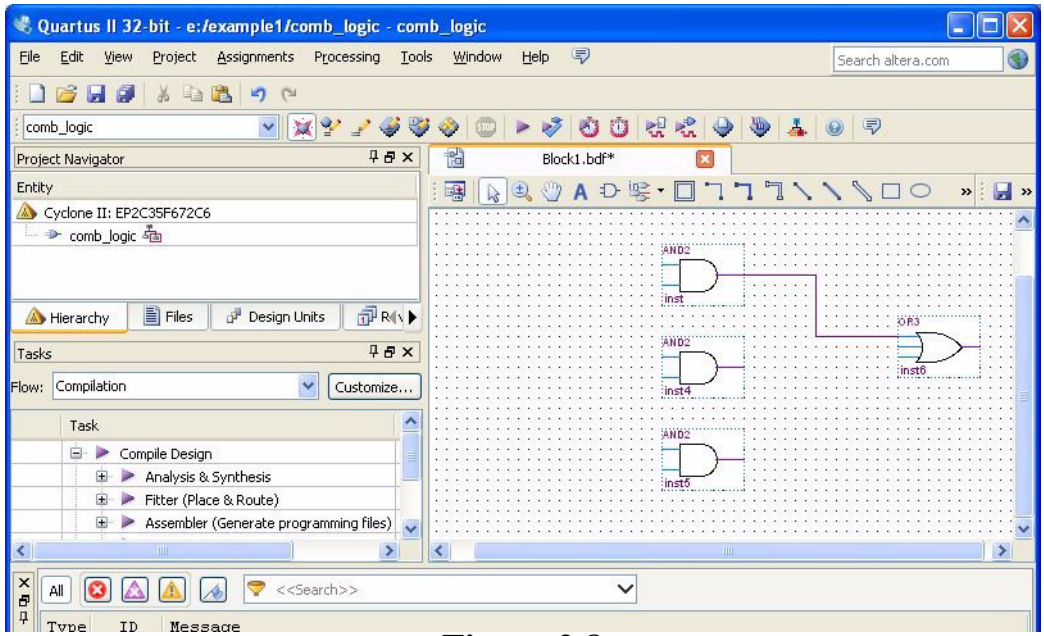


Figure 2.8

8 Repeat the procedure and connect the remaining outputs of the two remaining *and2* to the unattached *or3* inputs. Note that components and wires may be relocated as needed by dragging them with the mouse.

Defining Input and Outputs

1 Your logic design will have one or more input and outputs that define its interface with the outside world. In Quartus II® these I/O lines must be wired-from or wired-into special components that are named *input*, *output*, or *bidir*. It is also desirable to give these I/O components descriptive/user defined names. To accomplish this first double-click using your left mouse button on an empty space within the graphics window work area.

2 Next, select the special *input* component from the Altera component library. You can do this by expanding the ***primitives*** library to reveal the ***pin*** sub-library (i.e. select *c:\altera\...\quartus\libraries\primitives\pin*). Then select the *input* component and left-click on the *OK* button as shown in Figure 2.9.

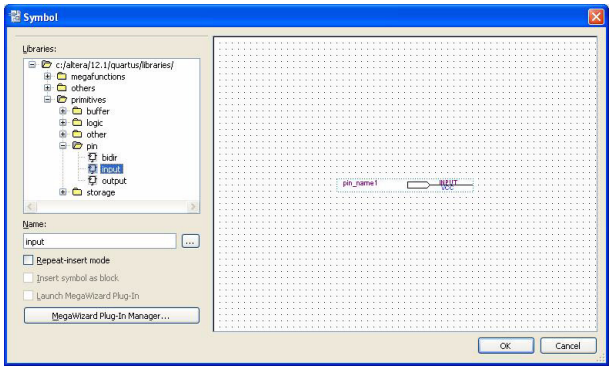


Figure 2.9

3 Then left-click on the place in the graphics window work area of the *Block Editor* where you would like to place the component.

4 Make five copies of the *input* component and connect all six of these components to the three *and2* components using wires in the manner described previously.

5 Next select an *output* component from the same ***pin*** sub-library (i.e. select *c:\altera\...\quartus\libraries\primitives\pin*)

6 Connect the *output* component to the output of the *or3* gate.

When you are finished all of your inputs and outputs should be wired as shown in Figure 2.10.

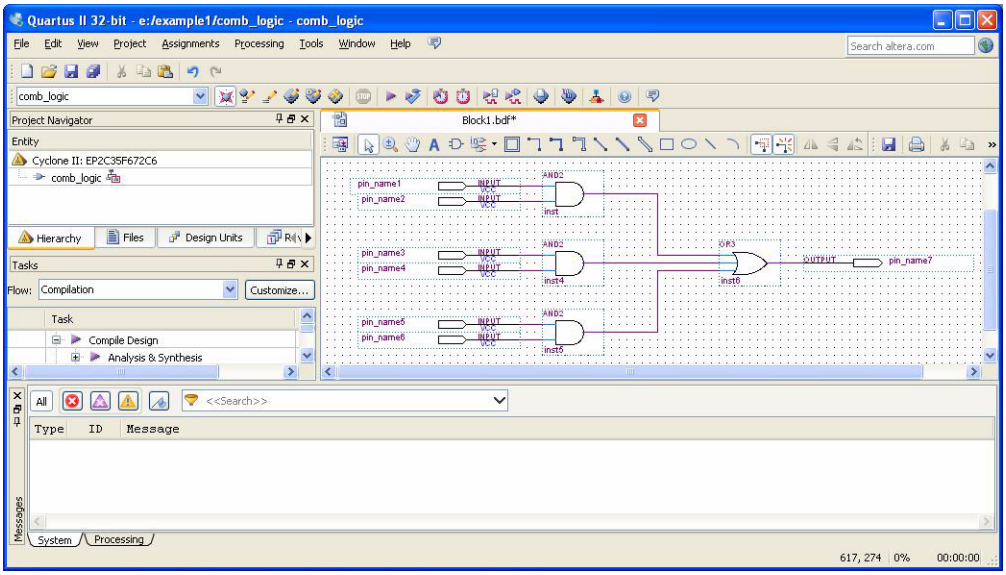


Figure 2.10

7 The next step is to give user defined names to your inputs and output. For this example give your inputs the names **I0** through **I5** by double-clicking on the "pin_name" text of each input symbol and then typing the appropriate new name. In the same manner label the *output* component **O**. If you miss the *pin_name* area of the *input* or *output* components but click on the component itself a *Pin Properties* dialog box will appear and the name can be entered in the *Pin name(s):* field as shown in Figure 2.11. In this case, just click on the *OK* button after entering the name.

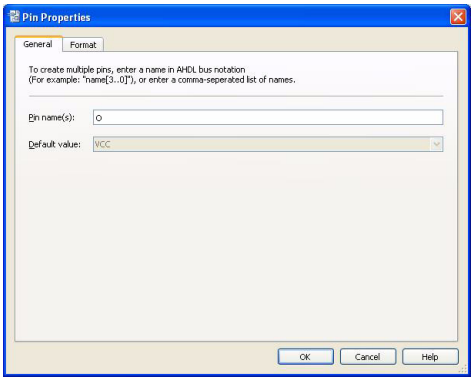


Figure 2.11

The final schematic is shown in Figure 2.12.

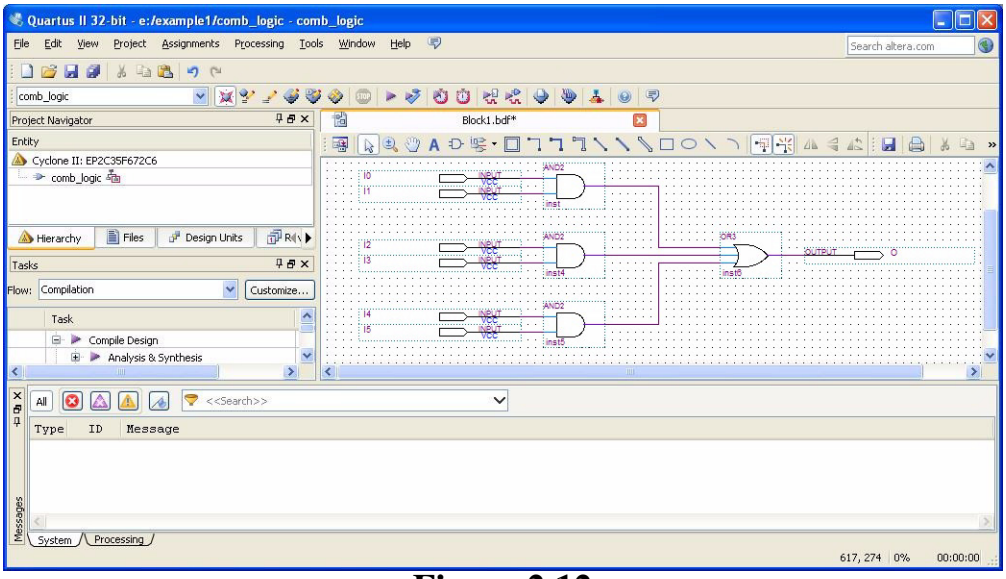


Figure 2.12

8 Now you need to save the schematic (bdf) file and include it in your project. To accomplish this select *Save* from the *File* menu or left-click on the Icon on the tool bar. This will bring up the *Save As* window as shown in Figure 2.13. Notice that the default file name is the same as the project name (**comb_logic**) and the *Add file to current project* option is automatically selected. This is what you desire so simply accept the defaults and left-click on the *Save* button.

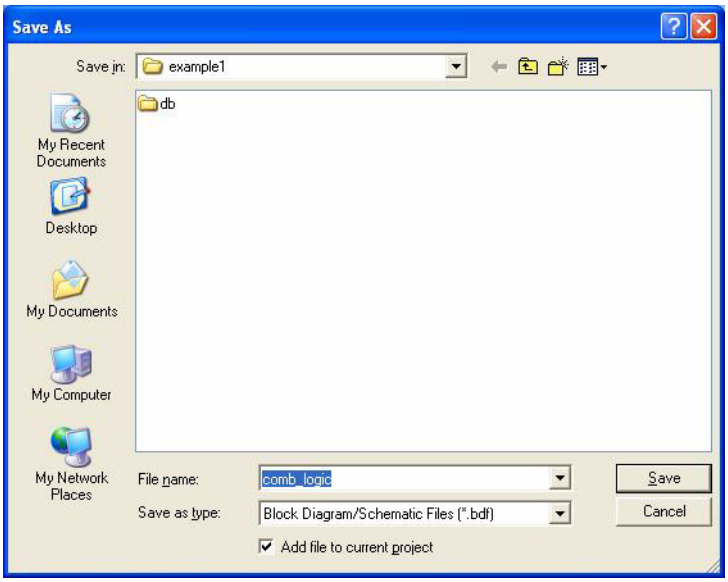


Figure 2.13

Congratulations, you have now completed your first schematic block diagram.

Creating a Design Hierarchy in you Schematic Representation

The schematic representation of Figure 2.12 can be considered to be a “flat” design because it utilizes only low-level logic primitives which are interconnected together on a single sheet to express the desired functionality. While this is an acceptable way to express very simple designs, for most situations creating a flat design would be too complicated, and error prone and even if successful would result in a representation that would be impossible to understand or maintain. One solution to this complexity management problem is to employ functional decomposition techniques that partitions the design along a multi-level design hierarchy. At the top level there can be a basic block diagram of the system where each block represents a high-level function. As one transcends to the next level by tranversing “into” one of the blocks then one will view a schematic of the subsystem that is represented by that block. As one transverses down the hierarchy one would expect to eventually find low-level schematics with all components for that module being described at a primitive component level (AND OR, NOR, etc.).
It should be noted that within this paradigm commonly used logic configurations are often encapsulated into modules that are called components. Each time this logic configuration is needed then a copy of that component can be placed in the design (instantiated) at the specified point in the hierarchy from a library in the same manner as the primitives. Components can come in prepackaged libraries or the user can add them to the current project library. Common components include multiplexers, decoders, encoders, latches, buffers, ROMs, RAMs, ALUs, and even CPUs.

To demonstrate how to create a hierarchical design let us now assume that we would like to use our entire previous design as a component building block. While the simplicity of this function does not lend itself to widespread use the same steps should be followed when creating other more useful component modules or blocks in a multi-level hierarchy.

1 Enter your existing project and open up a new bdf file. To do this left-click on *New* which is under the *File* Menu as shown in Figure 12.14 (or click on the upper left icon on the upper tool bar of the Quartus II window).

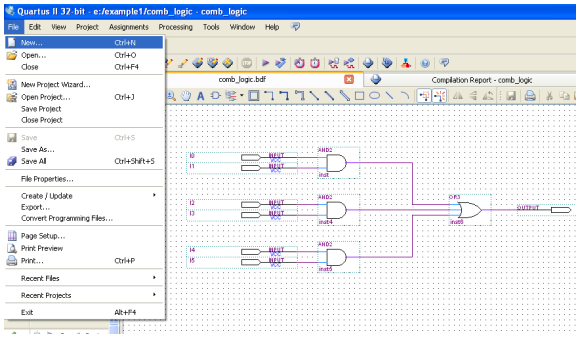


Figure 2.14

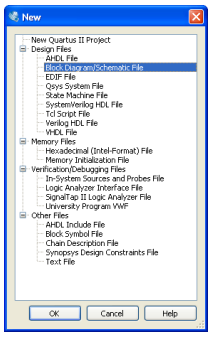


Figure 2.15

The *New* window should appear as shown in figure 2.15. Chose the *Block Diagram/Schematic File* option and press the *Ok* button. A new block editor tab should appear as shown in Figure 2.16. It is tentatively given a default name (**Block2.bdf**).

2 The next step is to enter the logic associated with the new block you are creating. In general you will create a schematic in a similar way that you did for your flat designs, entering components and I/O pins and interconnecting them using the Quartus II methods described previously. The major difference is that your I/O pins will become component pins, not general I/O pins with the component pins matching the names that you have given these pins.

3 For this example, to place all the functionality of your original design into this block without reentering it from scratch you can simply click on the **comb_logic.bdf** tab. Then cut your original top level design into the clipboard. After which you can select the new block tab (**Block2.bdf**) and then paste the design into the work area of the new block. One way to do this is to click on the upper left-hand corner of your design that is under the comb_logic.bdf tab with your left mouse button while holding down the <*ctrl*> key and bring the mouse down to the right hand corner after which you release the mouse button. All elements should be highlighted. Then you can cut the item and place it in your clipboard. Do this by pressing <*ctrl*> *X* or through the edit menu. Then go to the new block (**Block2.bdf**) tab and paste your design into this block.

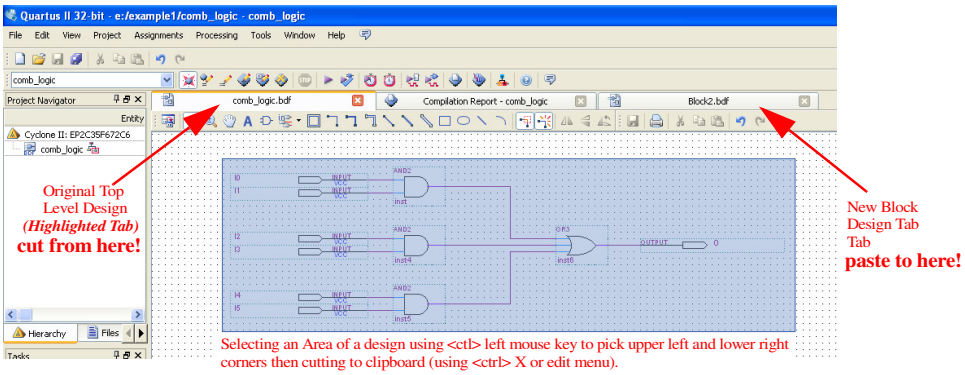


Figure 2.16

4 The next step is to save this new block file and include it in your project. To accomplish this select *Save* from the *File* menu or left-click on the Icon on the tool bar. This will bring up the *Save As* window as shown in Figure 2.17.

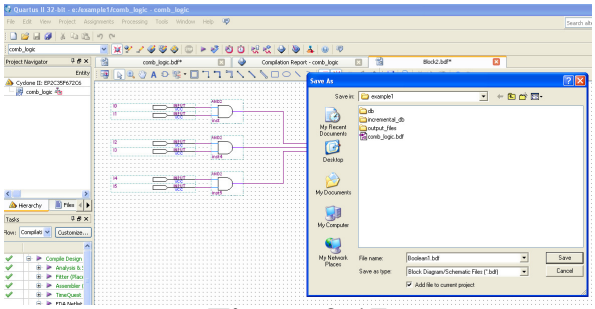


Figure 2.17

Initially the default name, **Block2.bdf**, should be highlighted. Change it to something more meaningful such as **Boolean1.bdf** as shown in the figure. This will also become the name of the component module. Make sure that the *Add file to current project* box is checked and click *Save*. The block tab's name should now appear as Boolean1.bdf and the asterisk should have disappeared indicating that it has been saved. The module has now been saved and added to the project but before it can be used as a component a symbol will have to be created.

5 The next step is to create a component symbol for this block. To do this first make sure that the block tab (**Boolean1.bdf** in the example's case) is selected. Then chose the *Create Symbol Files for Current File* option that is under the *Create/Update* submenu of the *File* menu as shown in Figure 2.18.

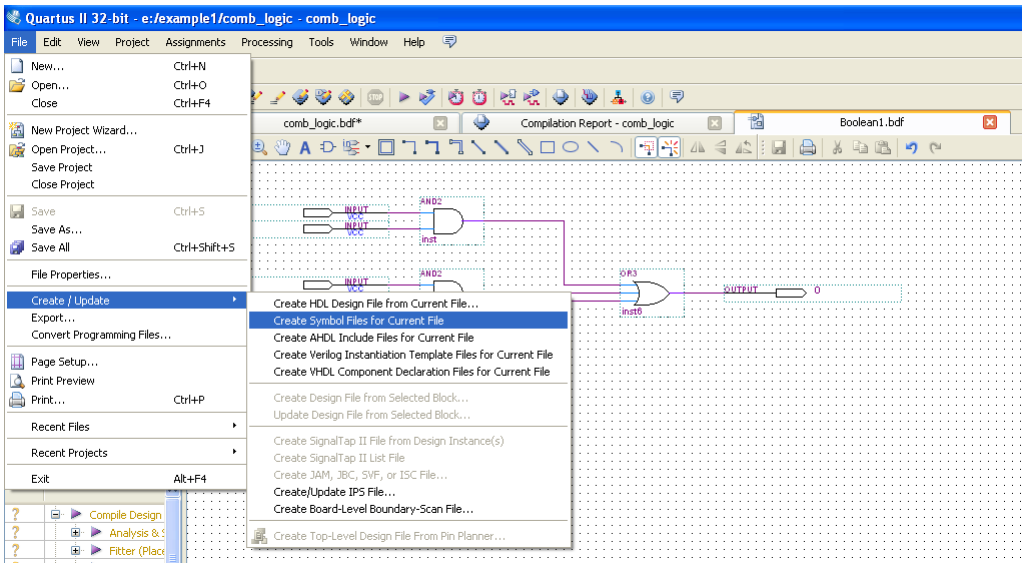


Figure 2.18

This will bring up the *Create Symbol File* dialog box as shown in Figure 2.19. By default the symbol name will be the same as the module file name (without the .bdf extension). Click the *Save* button to continue.

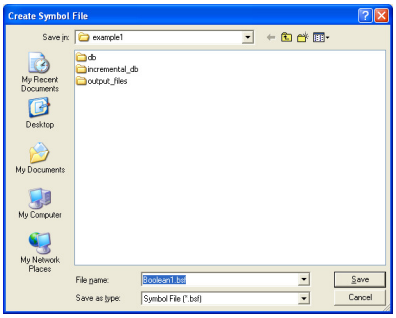


Figure 2.19

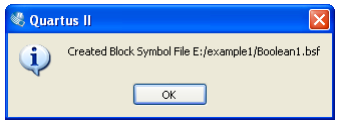


Figure 2.20

If there are no errors then Quartus II will generate a status window as shown in Figure 2.20 that indicates that the Block Symbol file was successfully created. Click *Ok* to continue.

6 The next step is to actually use the component that you have created in the design. This is sometimes called instantiating the component. In general to do this requires that you go to the place within your design hierarchy that you need the component and then simply select the component from your project library that is associated with your design. The component will be interconnected to others in the normal manner.
In this example we would like to replace our original design at the top level with the new component that we have just created. To do this go back to the **comb_logic.bdf** tab. It now has an asterisk because we have modified the original top level design by cutting the circuitry. Double click in the white space or left-click on the icon in the normal manner to reveal the new component you have created (you can also select the *Insert Symbol* option from the *Edit* Menu). You should now see a *Project* folder above the other folders in the library area as shown in Figure 2.21.

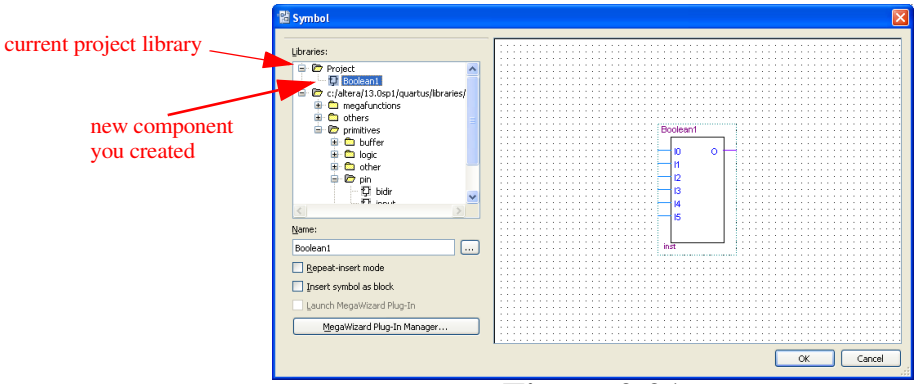


Figure 2.21

Expand this folder by clicking the ⊞ symbol. You should see the new component **Boolean1**. Click on this and click on the *OK* button. You can now place this in your top level schematic as you would any other component. Notice the pin names on the symbol are the same as your pin names on the lower level of the design. To create an equivalent two level hierarchical design that is equivalent in functionality to your original flat one simply connect up the inputs and the output to I/O connector components (input and output) and name them in the same way as you did previously. Figure 2.22 illustrates the resulting implementation.

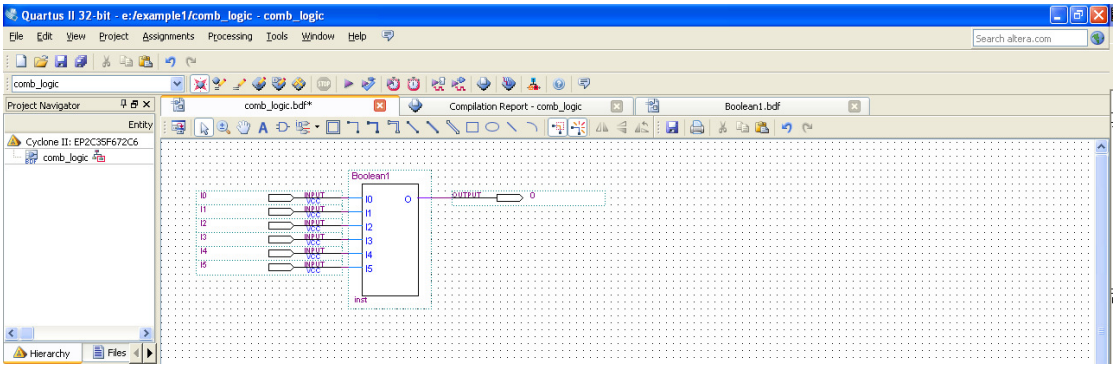


Figure 2.22

After you have saved this file you can transcend down the hierarchy by simply double clicking with your left mouse button on the individual components. In this case, double clicking on the **Boolean1** component will cause you to descend down to its gate level representation that you entered previously making active its tab in the bdf editor. This is true of any design component regardless of whether it is user entered or part of a preexisting library.

Part B: Creating a Verilog HDL Representation of your Design

Often you may desire to create a design that is written entirely in the Verilog hardware description language to take advantage of this language's portability, flexibility, and ability to model hardware at multiple levels of abstraction. To use this method of design capture follow the following steps.

Entering the Verilog Model Editor

1 First create your project using the *New Project Wizard* as described in Part 1 of this tutorial.

2 You will then use a special text editor to allow you to enter your design as a Verilog HDL module. To do this, first select *New* from the *File* menu. This will bring up the file type box that is shown in Figure 2.23. Then highlight the *Verilog HDL File* option and press the *OK* button.

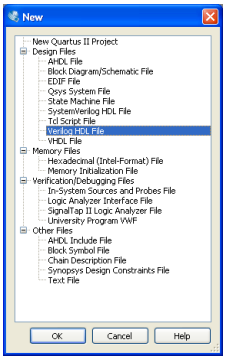


Figure 2.23

3 This will bring up the *Verilog Text Editor* as shown in Figure 2.24.

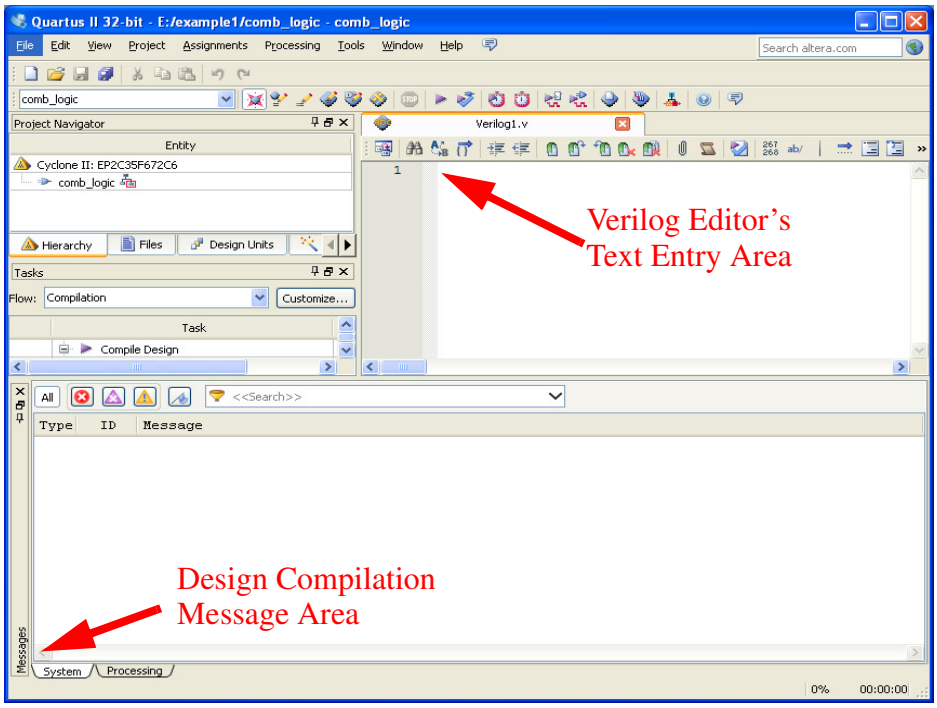


Figure 2.24

On the upper right hand side of this editor is the location of the text entry area where you will create your Verilog HDL model. The name of this text window is tentatively set by the software as *Verilog1.v* but this will automatically be renamed to the name of your project when you save your design for the first time. To begin entering your model simply left-click with your mouse into the *Verilog Editor's* work area. For the **comb_logic** example a *continuous assignment* style Verilog representation is shown in Figure 2.25 below. This is equivalent in functionality to the schematic capture version of this design that was presented in Figure 2.12 in the previous section.

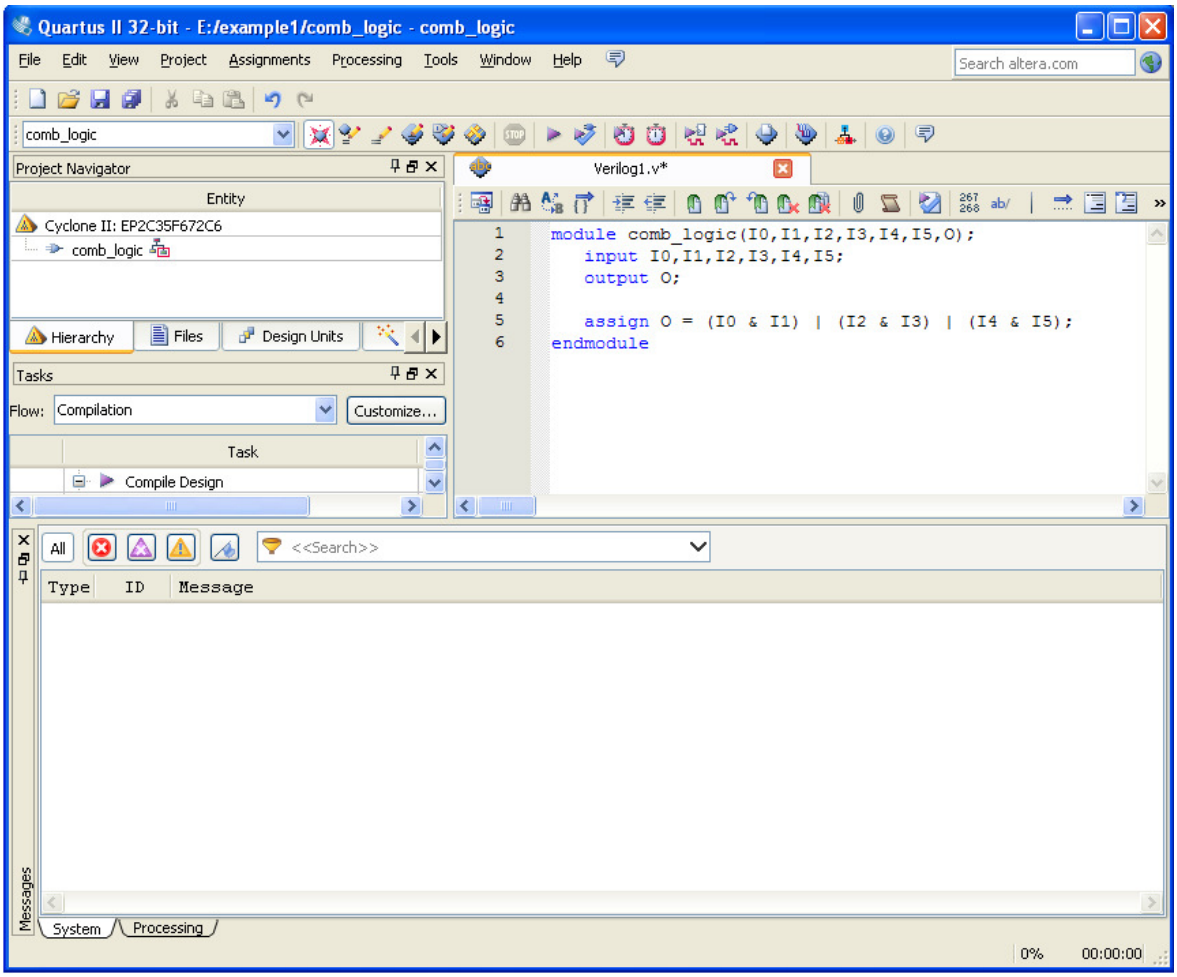


Figure 2.25

Note that the keywords in Verilog are highlighted in blue. User defined names are shown in black. If this is the top level of your model, the signals that are declared as module ***input***, ***output*** or inout will automatically form the inputs and outputs of your design of your targeted FPGA representation. There is no need to name it further. This will be the simulation signal name and the emulation pin name that you will use during pin assignment during the configuration. ***In the past it has been a requirement that the top-level module name of the Verilog model be the same a your Quartus II project name***. This requirement has been relaxed but it is probably good design practice to keep this original convention.

4 The next step is to save the Verilog model as a v file and include it in your project. To accomplish this select *Save* from the *File* menu or left-click on the Icon on the tool bar. This will

bring up the *Save As* window as shown in Figure 2.26. Notice that for the Verilog version of the **comb_logic** example the default file name is the same as the project name (**comb_logic**) and the *Add file to current project* option is automatically selected. This is what you desire so simply accept the defaults and left-click on the *Save* button.

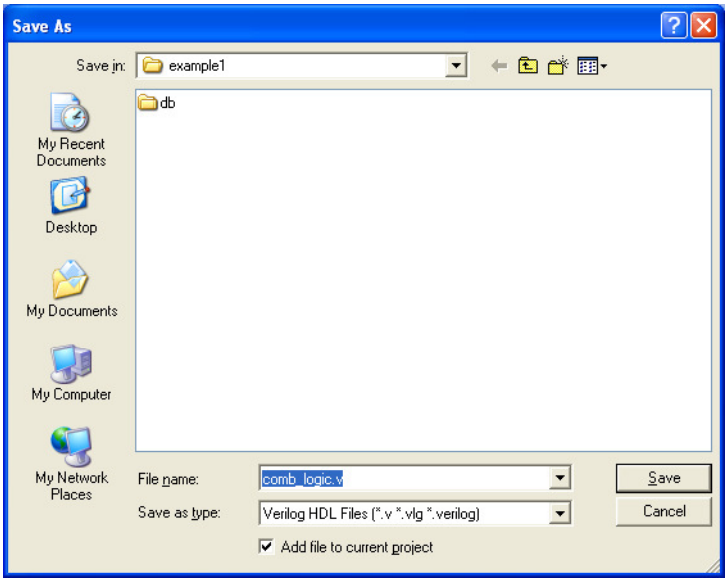


Figure 2.26

Congratulations, you have now completed your first Verilog design. You are now ready to compile the project. You may have to re-enter this step if there are errors or warnings or if you discover errors in your design during the simulation, timing analysis, or emulation phases of the design process

Part C: Creating a Hybrid Schematic Capture/Verilog HDL Design

It is common practice to incorporate both schematic capture and HDL modules within the same design. Such a design by its very nature is hierarchal. While it is possible to have Verilog HDL as your top module and then to instantiate bdf derived components within Verilog itself the more common approach is to create a block diagram representation at the top level and to then instantiate within this framework modules that are represented in Verilog HDL. This section highlights how one would proceed with the later approach -- creating a hybrid schematic capture/Verilog HDL representation where the top level of the design is a schematic. In many ways this is done in the same manner as was discussed in the subsection entitled “Creating a Design Hierarchy in you Schematic Representation” of Part A but with Verilog HDL techniques used to create new modules instead of Schematic Capture techniques. The steps to follow are outlined below:

Entering the Verilog Model Editor

1 First create your project using the *New Project Wizard* as described in Part 1 of this tutorial.

2 Enter your design as a schematic diagram. To do this, first select *New* from the *File* menu (or click on the upper left icon on the top tool bar of the Quartus II window). This will bring up the file type box that is shown in Figure 2.27. Then choose the *Block Diagram/Schematic File* option and press the *OK* button.

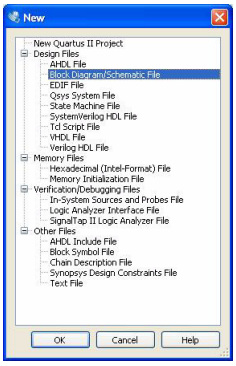


Figure 2.27

3 This step involves saving the top level schematic design file. This design file will eventually contain the highest level components and modules that will make up your design but many of these modules and components may not be present the first time you enter your design. You must first create these modules and add their symbols to the project library. At this point in time you can go ahead as save the file which has the same name as the project that you created in Part 1 of this tutorial. Quartus II will not allow you to save a blank version of the file so you can enter some of the necessary components such as the *input*, *output*, or *bidir* connectors. After this is done select *Save* from the *File* menu or left-click on the Icon on the tool bar. This will bring up the *Save As* window as shown in Figure 2.28. Notice that the default file name is the same as the project name (for this base example, **comb_logic.bdf**) and the *Add file to current project* option is automatically selected. This is what you desire so simply accept the defaults and left-click on the *Save* button.

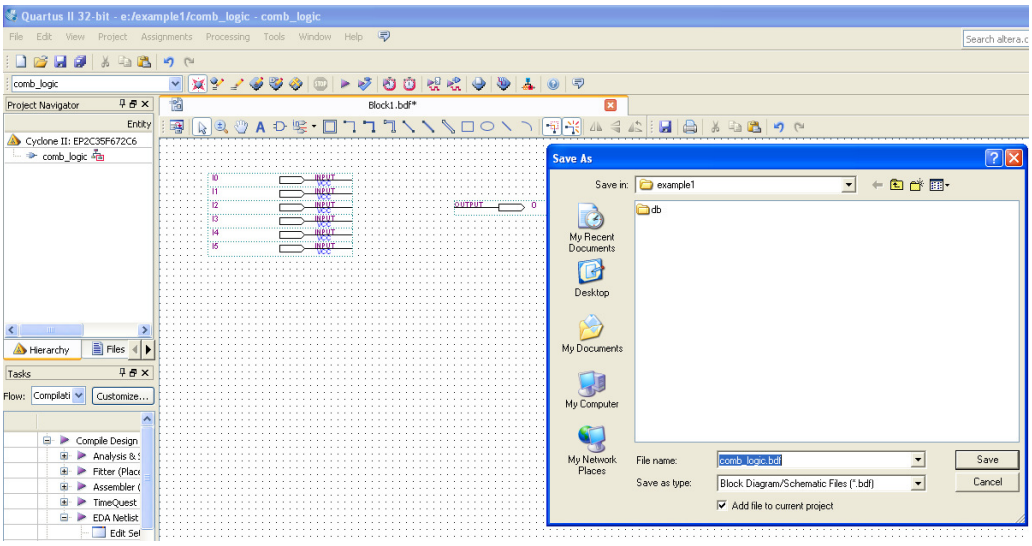


Figure 2.28

4 The next step involves entering a new Verilog HDL or schematic capture representation of each specific building block module. In the case of a Verilog HDL module select *New* from the *File* menu. This will bring up the file type box that is shown in Figure 2.29. Then highlight the *Verilog HDL File* option and press the *OK* button.

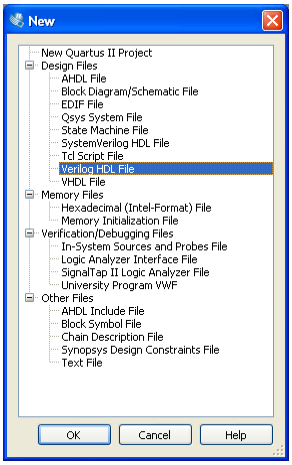


Figure 2.29

Then under the tab that contains the Verilog text editor enter you design giving the top-level module a unique name. For the example design we will give the module the name Boolean2. In all other aspects this Verilog file is the same as in the Verilog only representation of Part B as shown in Figure 2.30.

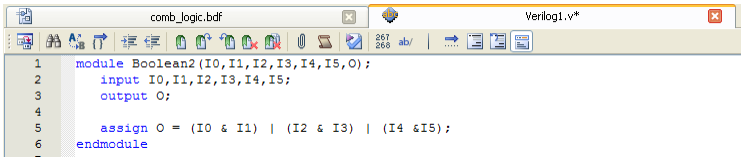


Figure 2.30

5 The next step is to save this new Verilog HDL file and include it in your project. To accomplish this select *Save* from the *File* menu or left-click on the Icon on the tool bar. This will bring up the *Save As* window as shown in Figure 2.31.

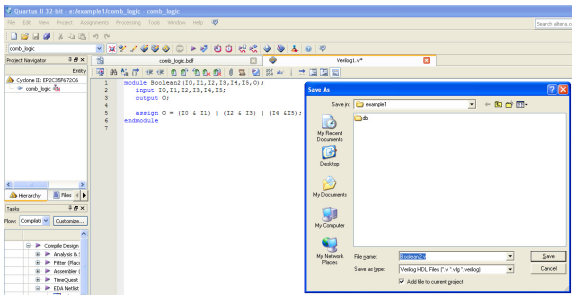


Figure 2.31

Initially the default name, **Verilog1.v**, should be highlighted. Change it to something more meaningful such as **Boolean2.v** as shown in the figure. This will also become the name of the component module. Make sure that the *Add file to current project* box is checked and click *Save*. The modules tab's name should now appear as **Boolean2.v** and the asterisk should have disappeared indicating that it has been saved. The module has now been saved and added to the project but before it can be used as a component a symbol will have to be created.

6 The next step is to create a component symbol for this module. To do this first make sure that the module tab (Boolean2.v in this case) is selected. Then chose the *Create Symbol Files for Current File* option that is under the *Create/Update* submenu of the *File* menu as shown in Figure 2.32.

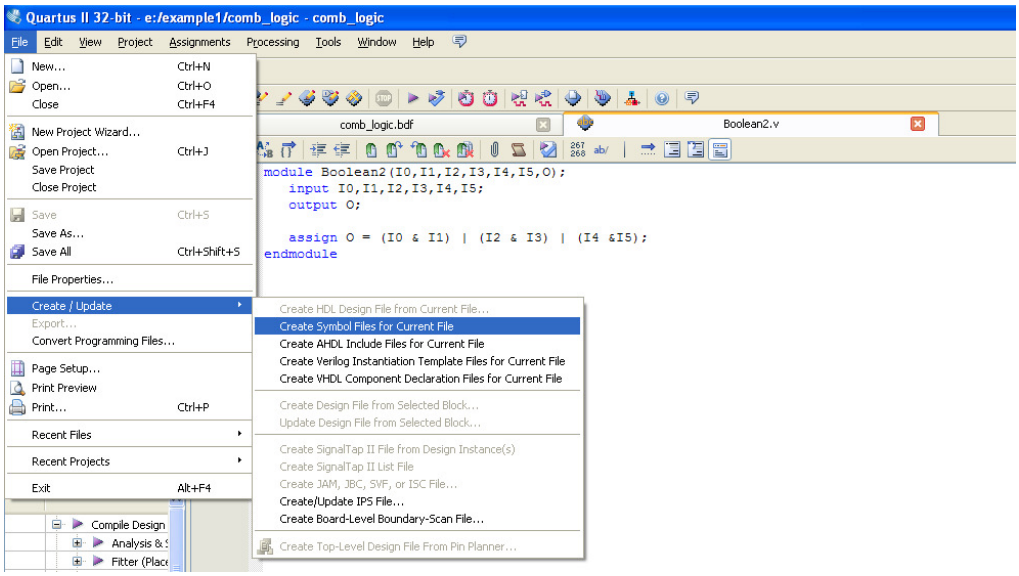


Figure 2.32

If there are no errors then Quartus II will generate a status window as shown in Figure 2.33 that indicates that the symbol file was successfully created. Click *Ok* to continue.

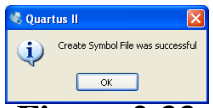


Figure 2.33

7 The next step is to actually use the component that you have created in the design. This is sometimes called instantiating the component. In general to do this requires that you go to the place within your block diagram/schematic hierarchy that you need the component and then simply select the component from your project library that is associated with your design. The component will be interconnected to others in the normal manner.
In this example we would like to use this component within our top level schematic. To do this click on the **comb_logic.bdf** tab. Then double click in the white space in the design window or left-click on the icon in the normal manner to reveal the new component you have created (you can also select the *Insert Symbol* option from the *Edit* Menu). You should now see a *Project* folder above the other folders in the library area as shown in Figure 2.34.

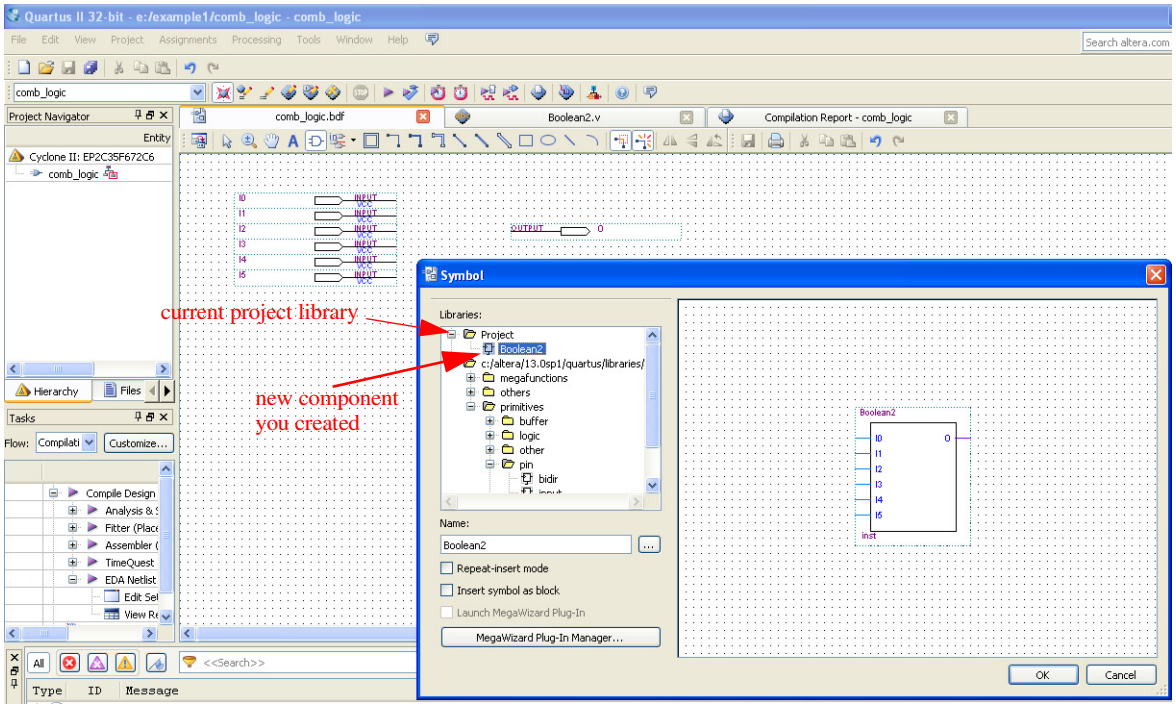


Figure 2.34

Expand this folder by clicking the ⊞ symbol. You should see the new component **Boolean1**. Click on this and click on the *OK* button. You can now place this in your top level schematic as you would any other component. Notice the pin names on the symbol are the same as the wire names of your inputs and output in your Verilog HDL module.
To create an equivalent two level hierarchical design that has the same functionality as the schematic or Verilog VHDL only implementation you only need to connect up the inputs and the output to I/O connector components (input and output) and name them in the same way as you did previously. Figure 2.35 illustrates the resulting implementation.

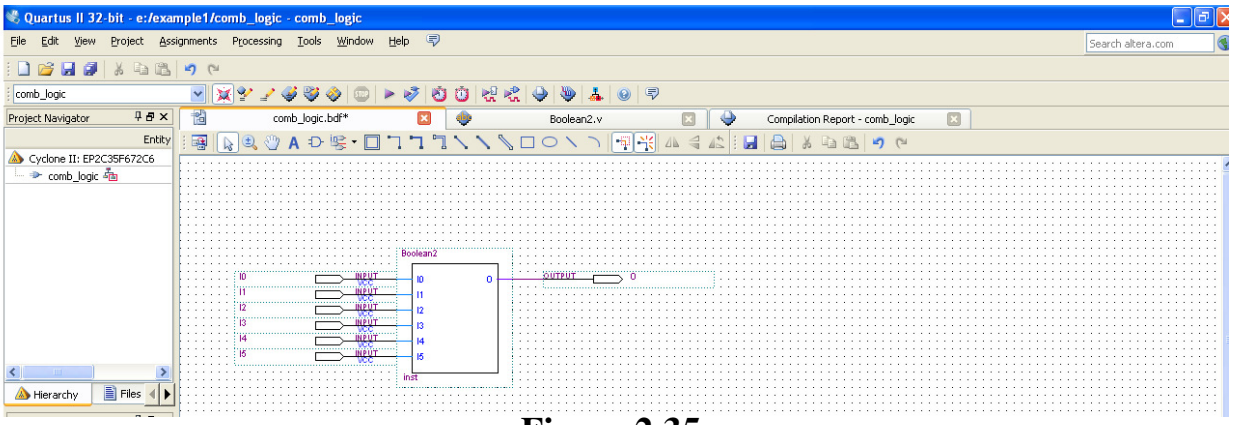


Figure 2.35

After you have saved this file you can transcend down the hierarchy by simply double clicking with your left mouse button on the individual components. In this case, double clicking on the **Boolean2** component will cause you to descend down to its Verilog HDL representation that you entered previously making active its tab in the Verilog HDL editor. This is true of any design component regardless of whether it is user entered or part of a preexisting library (unless it has been projected by encryption).

III. Compiling the Project

1 After you enter your schematic, Verilog, or hybrid representation of your design you will need to compile your project before you can begin simulation or configure external hardware to emulate your design. Compiling the project creates the internal information that the simulator needs to understand the logic makeup of your circuit. It also creates a file that can be used to load the design in the selected programmable logic device. To compile your project, first left-click on the ▶ Icon on the tool bar or select the *Start Compilation* which can be found under the *Processing* menu. The compilation process will then begin and a window similar to the one shown in Figure 3.1 will appear.

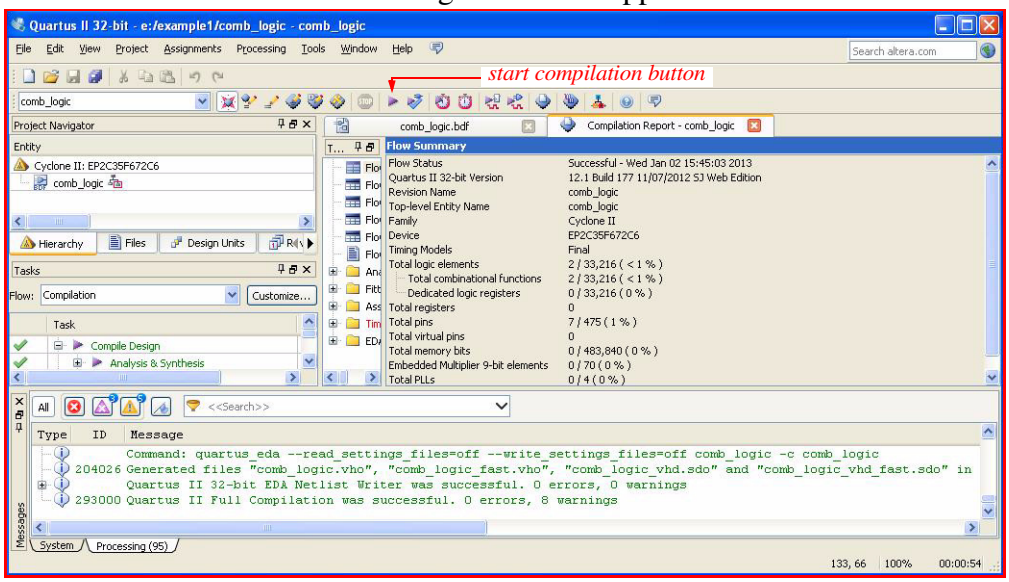


Figure 3.1

2 After the compilation is complete if there are no errors you should see a window similar to that shown in Figure 3.2. Left-click on the *OK* button to continue.

Figure 3.2

Note: you can review the warnings (and any compilation errors) by clicking on the *Warning* (or *Error*) tab at the bottom of the *Message Window* for both schematic capture and Verilog HDL design entry methods. Warnings need to be evaluated closely to determine if they are important to your design. In many cases warnings can be ignored. Warnings are classified into two major categories based upon their likely severity with the classification of critical being the warnings that most likely are real problems. Warnings can often be ignored but errors always have to be fixed before you can proceed to the other phases of the design flow (timing analysis, simulation, and design configuration). If you have a compilation error first examine the error message. Double-clicking on it should bring up the source of the error. This could be a component on a schematic or an offending line in a Verilog model.

Figure 3.3 shows an explanation of the three critical warnings generated for the **comb_logic** schematic capture example and in this case none of them will affect the simulation of our design. (Note a near identical set of warnings are also issued for the Verilog model of this design.)

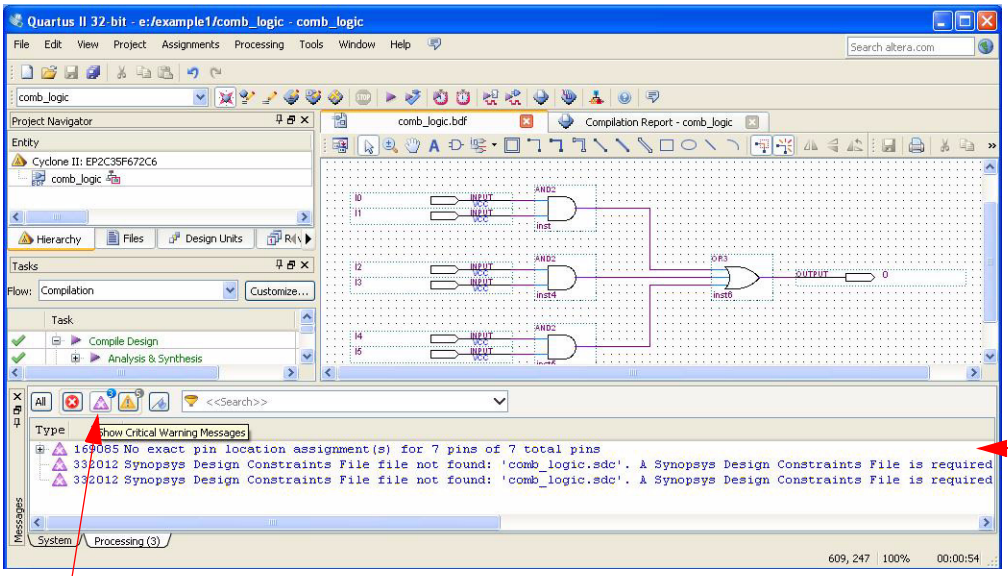


Figure 3.3

By clicking on the critical warning button the warning shown under the processing tab of the *Design Compilation Message* area are shown separately from the other warnings. The first warning tells us that we have not specified specific pin locations for our input and output pins (**I0-I5** and **O**). This is ok if all we wish to do is simulation but if we want to actually download our design into an FPGA these pins will need to be assigned. Right now the Quartus II tool has randomly assigned them for us. Again, this is fine if we want to perform an initial functional simulation but we will have to make this assignment before any detailed post place-and-route timing analysis is performed or our design is actually implemented in existing reconfigurable hardware such as the DE2-115 board. The other two warnings can be ignored if we are not performing timing constraint analysis.

When compilation errors or warnings that must be fixed occur, they can be fixed by modifying the schematic (bdf) file or the Verilog Model (v) file. You can access the schematic or Verilog HDL file again simply by left-clicking on the appropriate tab in the main Quartus II® window (either the **comb_logic.bdf** or **comb_logic.v** file for this example depending on the specific design entry method employed).

IV. Simulating Your Design

Simulation is a process through which the functionality of a design can be evaluated using computer software. Simulation has the advantage that it often allows one to determine the correct operation of a design without prototyping it in hardware. Simulation also allows one to verify timing and to test for conditions which may be destructive if the design were actually implemented in hardware. One of the major drawbacks of simulation, though, is the amount of time that it takes to simulate complex circuitry, which can be several orders of magnitude slower than the actual execution time of the corresponding hardware. It must also be remembered that simulation only approximates the behavior of actual circuitry to a certain level of fidelity. The real operation of an actual circuit may differ. With that said, though, simulation is a very powerful tool that should be utilized early an often within the design cycle.

4.1 Preparing Schematic Representations for the ModelSim® Simulation

For schematic-based designs to fully utilize the capabilities of the Altera/ModelSim® simulator it is necessary to direct Quartus II® to automatically create a Verilog HDL file for each block description (bdf) file that is present in the design. The design project will then need to be modified so that it utilizes these automatically-generated Verilog HDL source files instead of the corresponding block description files. The modification of the project only needs to occur once but if modifications are then made to any of the schematic files then it is the responsibility of the user to make sure that they direct Quartus II® to update the corresponding Verilog HDL file. If a design utilizes only Verilog HDL then one can proceed directly to simulation. Otherwise one should perform the following steps;

5 *Creating Verilog HDL file(s) from one or more block diagram files.* This is accomplished by first opening up in the active window tab the corresponding schematic/block diagram (bdf) file. Then select the *Create HDL file from Current File* option that is under the *Create/ Update* submenu that is under the File menu as shown in Figure 4.1 for the schematic **comb_logic** example.

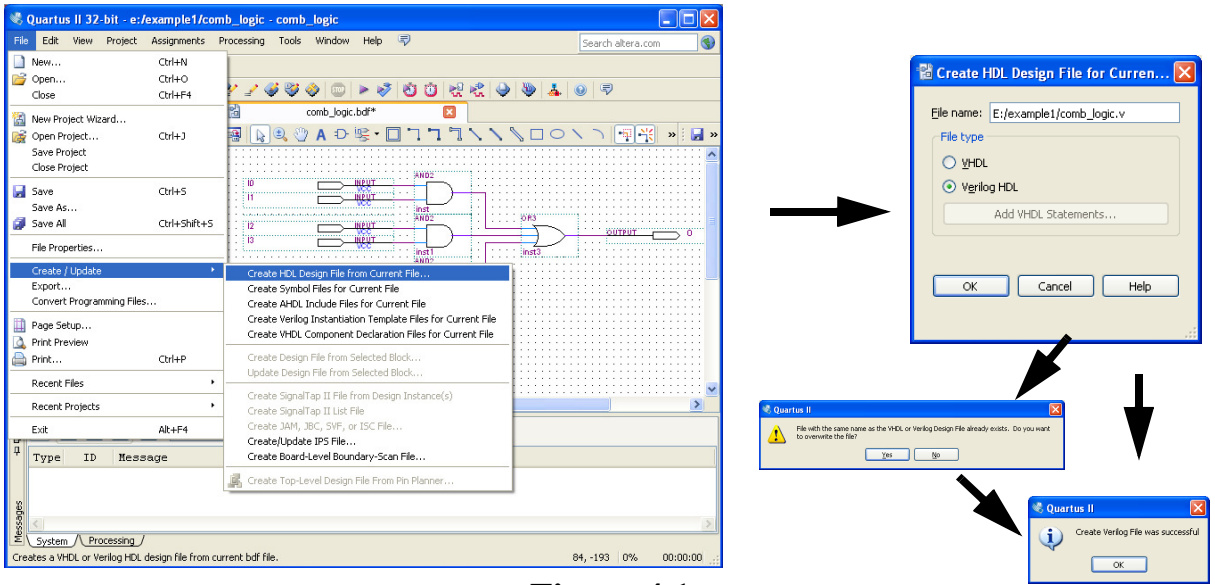


Figure 4.1

A dialog box will appear that ask the user to select an HDL language. Choose *Verilog HDL* and click *OK*. If the block diagram file (bdf) file has been previously modified and the corresponding Verilog HDL file is now to be updated a message will appear indicating that a file already exists with the same name and will be overwritten if you continue. Click *Yes* here. If the block diagram file is correct in some versions of Quartus II the *Create Verilog File was successful* dialog box will appear. Click on *OK* to continue. ***It is important to note that updating the corresponding Verilog HDL file does not occur automatically and it is up to the user to update this file using this procedure each time a block description file is modified***. If this does not occur the changes in the block description file will not be reflected in your simulation.

6 *Setting up your project to reference the HDL files.* After the Verilog HDL files have been generated from each block diagram files in the schematic, then the project profile will need to be updated so that they reference the Verilog HDL versions of these modules and not the block diagram version. This should only have to be once for each block diagram (bdf) file. To access the design project select *Add/Remove Files in Project* option that is under the *Project* menu as shown in Figure 4.2 below.

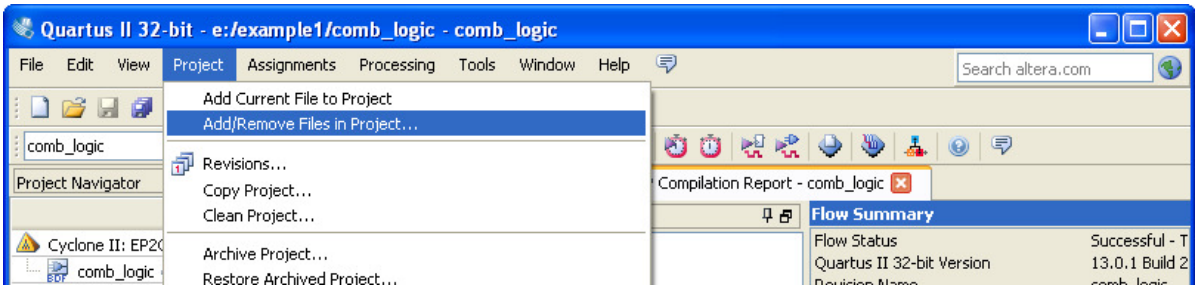


Figure 4.2

This will bring up the general project *Settings* window as shown in Figure 4.3. The *File* category should be highlighted in the manner shown in the figure. For each schematic/block description file (*.bdf) in the design one should select that file from the list and then click on the *Remove* button. When this occurs the file name should disappear from the list. (Note the file though still exists and can be accessed by the user using through the *Open* option of the *File* menu, or by clicking on the icon, but it is now no longer part of the project). After the block description file has been deleted from the project, the Verilog version of the module should then be added. This version will have the same file name as the original block description file (bdf) but will possess the verilog (.v) extension. One can browse for the file using the button next to the *File name* text box or just enter the name in the text area directly. When the name of the Verilog HDL version of the module has been entered successfully then click on the *Add* button. When all of the Verilog HDL files have been selected in this way to replace their corresponding block diagram equivalents then click on the *OK* button at the bottom of the *Settings* window to continue. At this point the design is ready to simulate. (It can also be downloaded into the Altera DE2-115 platform -- but converting to an HDL representation is not required for this to occur.)

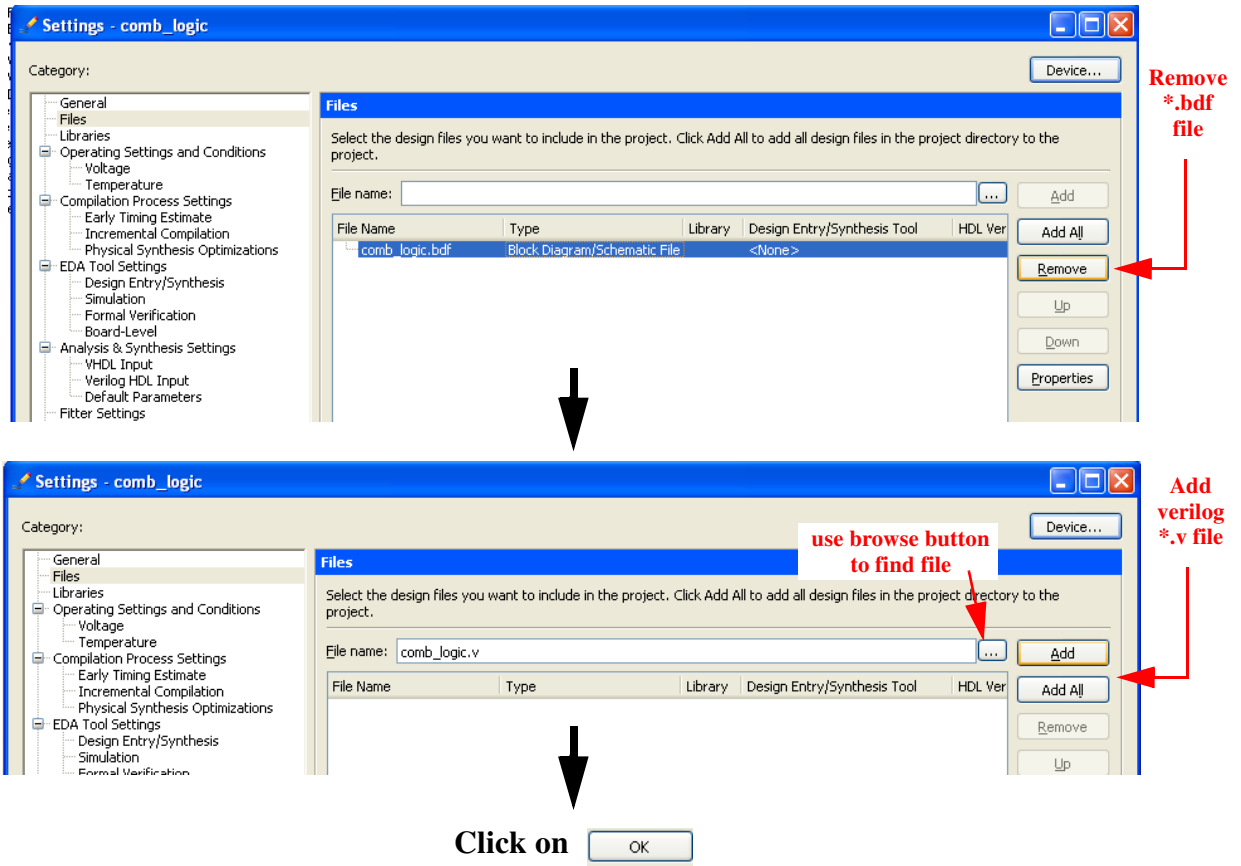


Figure 4.3

4.2 Basic RTL Simulation using ModelSim® -- an Overview

Register Transfer Level, RTL, simulations (sometimes called functional simulations) are used to verify the logical correctness of a design. This type of simulation does not account for delays that exist in the final implementation, but assumes that all logic and wiring delays are zero. The simulation is very useful for determining design issues early in the design cycle even before the final hardware implementation of the design is determined. In this section we will highlight how an RTL simulation can be performed on the simple combinational design that was successfully entered in Section II of this document. These steps are outlined below:

1 *Entering ModelSim® from Quartus II®.* After you have entered your design and successfully compiled it as discussed in Section III* (i.e. no Error Messages and the Warning Messages that were generated make sense within the framework of your design) then you are ready to enter the ModelSim® simulation environment to simulate it. To enter the ModelSim® environment left-click on the RTL Simulation Icon, on the tool panel as shown in Figure 4.4

*Actually RTL design does not require that you perform a full post synthesis place-and-route compilation. Only an analysis and elaboration is required. Bypassing the full post-synthesis/ place-and-route process allows for a faster turn around time between design entry and verification for HDL-based designs. This could be very important for large designs but for the relatively small designs in this class, full compilation occurs at an acceptable speed and produces all the necessary files needed by ModelSim® for both RTL and post-synthesis/ **place and route** simulations so that is why full computation is recommended.

or from the *Tools* menu select the *Run Simulation Tool* submenu and then select the *RTL Simulation* option.

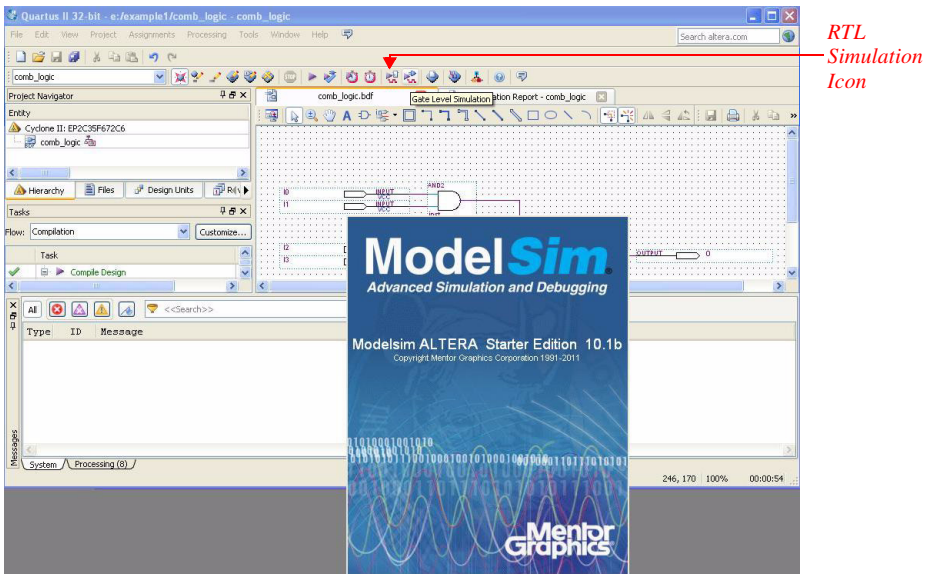


Figure 4.4

The first time you enter either a RTL or Gate-Level Simulation a dialog box may appear asking which simulation language you desire. If this occurs chose the Verilog HDL as your simulation language as shown in Figure 4.5. Then click on the *OK* button to continue.

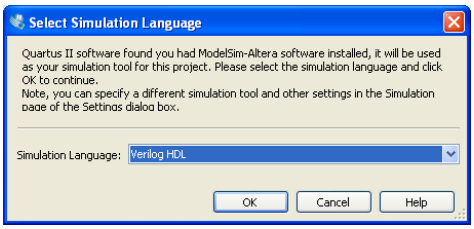


Figure 4.5

The next window should be the *ModelSim Simulator Window* as shown in Figure 4.6 below.

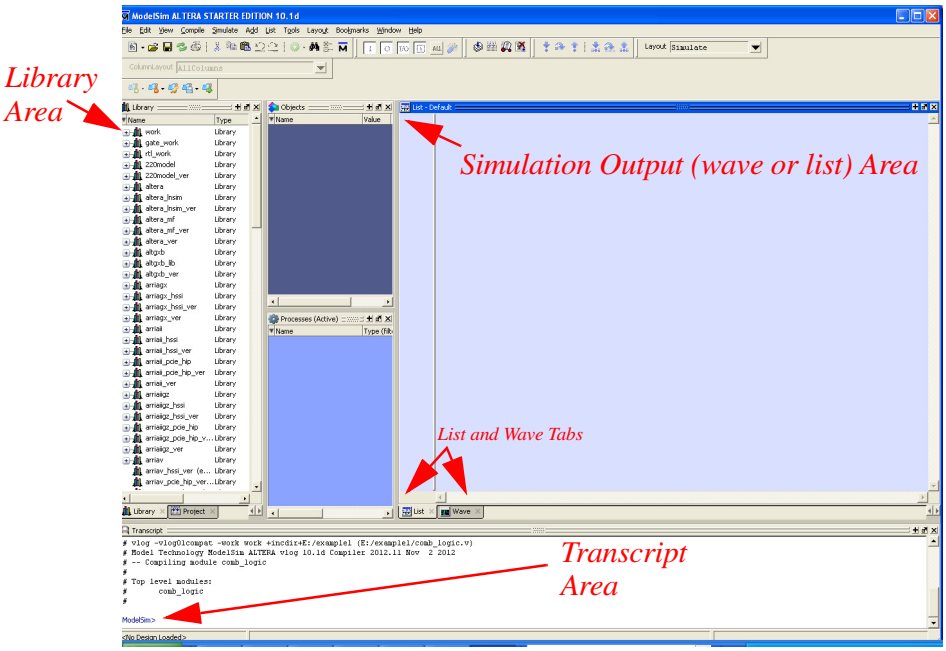


Figure 4.6

During this process various generic packages and libraries will be automatically added to your simulation work space. This also starts the simulation elaboration process for your design that you entered using Quartus II. By default, the ModelSim® program is set up to allow you to view the underlying Verilog HDL objects. This allows you to better understand the overall hierarchical structure of your design and its associated libraries. The two major areas that we will be using in this tutorial are the *Simulation Output Window* and the *Transcript Window*. The Simulation Output Window is where ModelSim® directs the simulation output. Output can be viewed in a textual list format or as a graphical waveform. These output areas are accessible by clicking on the corresponding *List* and the *Wave* tabs. The *Transcript Area* is where information about the simulation is provided by ModelSim® and where you can enter simulation commands. Most of the commands also have an equivalent sequence of menu operations. The base simulation excitation commands are presented in this tutorial, instead of the menu command sequences because of their portability across different versions of ModelSim®.

In should also be noted that each of these windows are initially docked in place in the manner shown in Figure 4.6 but they can be undocked at any time by clicking on the Icon. If you undock them they will form separate windows. In this case be cautioned to make sure you know where in your workspace these windows reside. Experience has shown that they can sometimes get lost or forgotten because the system does not automatically bring them to the front when data is updated. The next step is to initialize the simulation. This is done by selecting the *Start Simulation* option from the *Simulation* menu that is located on the ModelSim® toolbar as shown in Figure 4.7.

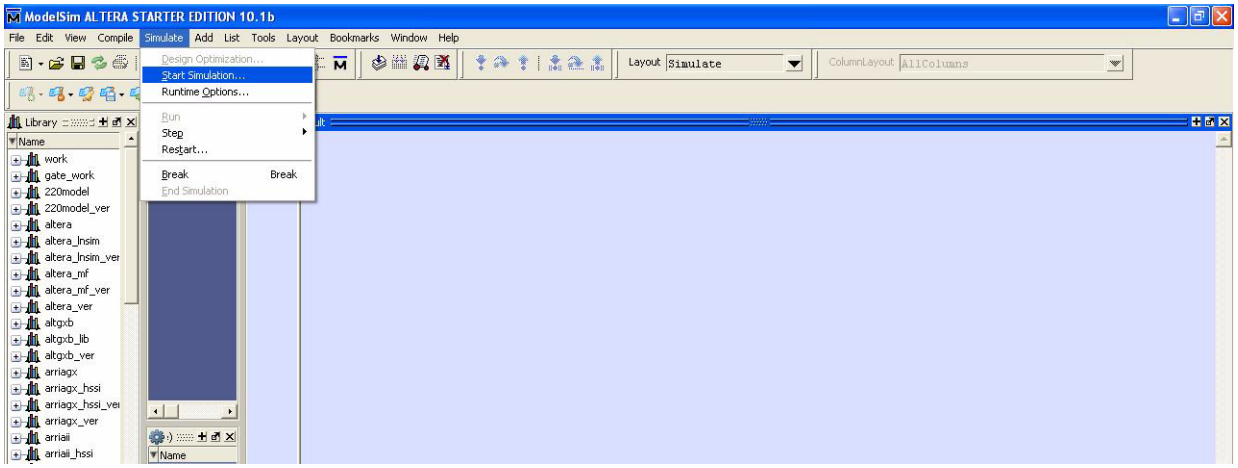


Figure 4.7

2 *Simulation Initialization.* This brings up the *Start Simulation Window* that allows you to select the top-level element of your design. This element is under the *Design* tab under the *work* library as shown in Figure 4.8. Expand *work* to reveal the sublibrary components. You should see a name that is the same as your top-level module which is **comb_logic** for the example we have been discussing. Left-click on this entity and then click on the *OK* button. In cases where you have more than one architecture for your design (such as might be the case for a design entered in VHDL) then you will need to transverse further until you find that specific architecture and selected it.

.

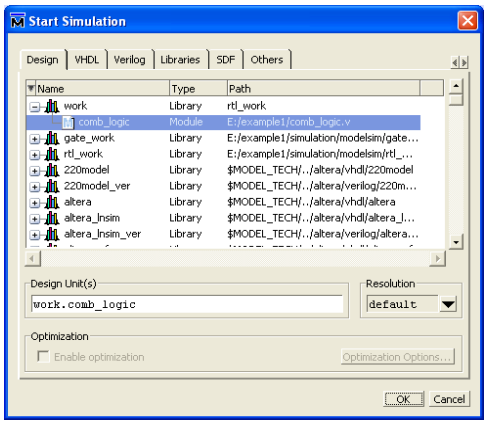


Figure 4.8

This executes the ModelSim® simulation *vcom* compiler/elaborator for your design and places your design in the default work library. It then starts the underlying *vsim* simulator which operates in an interactive mode throughout the simulation process. The simulation is now ready to accept additional simulation commands that you can give by typing them in the *Transcript Area* as shown in Figure 4.9.

.

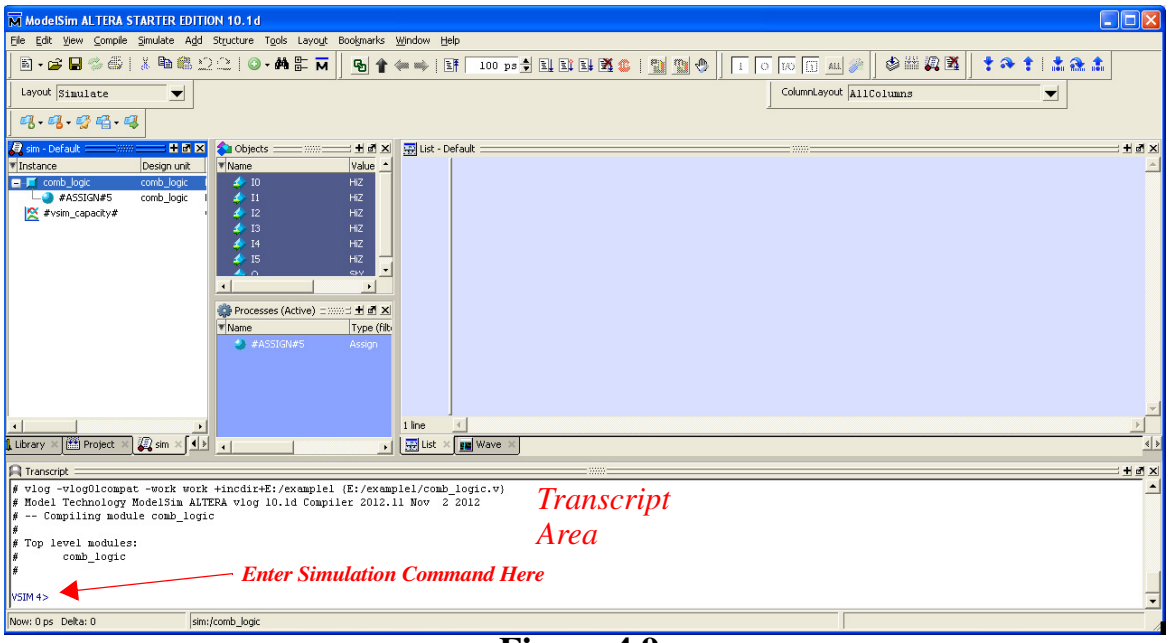


Figure 4.9

3 *Pin and Signal Monitoring*. Before you actually execute the simulation (and possibly as the simulation progresses) you need to determine which pins or signals you want to observe. This will allow the simulator to record only the important activities and events that are pertinent. In most real-world scenarios recording and viewing all possible internal events in not practical or desirable. To select the signals you would like to view you can use the ModelSim® *add* commands. The *add list* command adds the subsequent set of pin name and signals so that they will be monitored in textural form in the *List* window. The *add wave* command selects the set of signals and other objects to be displayed in a similar manner but allows them to be viewed as a waveform in the *Wave* window. To monitor the input and out-

put signals in as a waveform and as textural data you can issue both the *add list* and *add wave* commands before you run your simulation. To do this you need to place the set of I/O pin names of your top-level schematic or the signal names of your top-level module of your Verilog HDL model. Each of these names should be separated by spaces and the order that you place them is the order they will appear in your waveform and textual listing. For the **comb_logic** example the following two commands can be entered on separate lines in the *Transcript Window* at the VSIM prompt to monitor the six inputs and one output.

add list I0 I1 I2 I3 I4 I5 O
add wave I0 I1 I2 I3 I4 I5 O

If you enter a name that does not represent a valid object a red error message will be interactively posted in the *Transcript Window* to alert you of your mistake. To correct it simply enter a correct *add* command.

4 Stimulus Generation. The next step is to drive the inputs of your simulation in a manner that allows you to evaluate it. This can be done with the *force* command. The first argument of the force command is the name of the I/O pin or signal object. The next set of arguments represent the logic value you desire the pin or signal to take on followed by the time, relative to the current point in the simulation, that it should take on that value. If a time is not entered then it takes on the value specified from the current point in time and holds that value throughout the simulation run. The time should be expressed as an integer followed by the appropriate time unit (such as ms, us, ns, or ps). Each logic value/time pair should be separated from one another with commas.
For example, if the desired input stimulus for the six inputs in our **comb_logic** case is given by Table 4.1

Table 4.1: Desired Input Stimulus for *comb_logic* Example

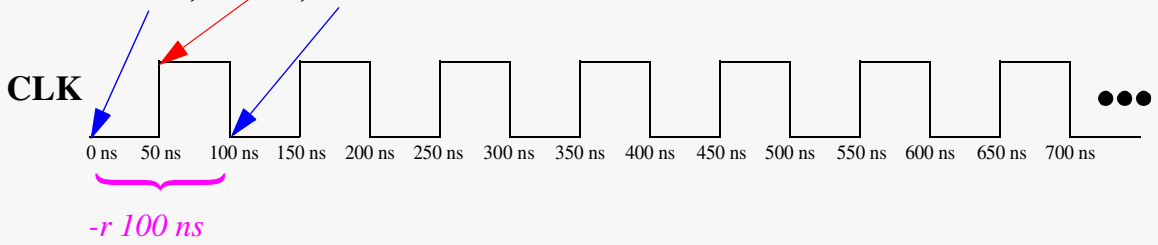
Input	0 ns	80ns	160 ns	240 ns	320 ns	400 ns	480 ns	560 ns
I0	0	1	0	0	0	0	0	0
I1	1	1	0	0	0	0	0	0
I2	0	0	0	1	0	0	0	0
I3	0	0	1	1	0	0	0	0
I4	0	0	0	0	0	1	0	0
I5	0	0	0	0	0	1	1	0

then the following set of *force* commands can be used to specify this stimulus.

force I0 0 0 ns, 1 80 ns, 0 160 ns
force I1 1 0 ns, 0 160 ns
force I2 0 0 ns, 1 240 ns, 0 320 ns
force I3 0 0 ns, 1 160 ns, 0 320 ns
force I4 0 0 ns, 1 400 ns, 0 480 ns
force I5 0 0 ns, 1 400 ns, 0 560 ns

Generating Clocks: Although the use of a clock signal is not needed in this example because it is a strictly combinational design it is important to note that the *force* command can also be used to generate periodic patterns such as a clock by using the -r option as shown below.

force CLK 0 0 ns, 1 50 ns, 0 100 ns -r 100 ns



Here the -r option specifies that the pattern that has been present for the past 100 ns (starting at time 100 ns) be repeated indefinitely giving a 50% duty cycle clock with a period of 100 ns that will run the duration of the simulation.

In general input stimulus can be created using these ModelSim commands or within the hardware description language (such as Verilog or VHDL) itself. One advantage to using the ModelSim commands are that they are portable across designs that employ different design capture methods (schematic and HDLs). A major disadvantage is that hardware description languages are much more robust in terms of the degree to which stimulus patterns can be made to be responsive to simulation output and can make use of built in constructs that enhance timing verification. After stimulus generation the next step is to actually run the simulation.

5 *Running the Simulation*. To finally run the simulation you should enter the *run* command in the *Transcript Area*. The argument for this command is an integer that specifies how long the simulation is to execute from its current point before the simulation. During simulation runs no additional simulation commands can be executed but once the simulation is over it is placed in the interactive mode allowing it to accept new simulation commands. (It should be noted that the simulator is event driven and it is possible for the simulation to end its execution before the specified end time if it runs out of events.)
For the **comb_logic** example the command to execute the simulation for 560 ns from the current point in time is simply

run 560 ns

When the simulation is complete, the results can be view and analyzed, and new simulation commands can be entered. These commands always take effect at the current point in simulation time unless the simulation is reset. You must reset a simulation in order to start again from the beginning. To do this you can enter the *restart -f* command in the *Transcript Area*. reset a simulation. If a design change is needed then one should exit ModelSim® and then make the necessary changes in the Quartus II®, recompile the design and then reenter the simulation in the same manner as before.

Viewing your Simulation Results: After the simulation is run you will want to view the results. To view as a waveform timing diagram make sure to select the *Waveform* window by clicking on the *Wave* tab as shown in Figure 4.10 (or activate this window by left clicking on it if you undocked it from the main ModelSim® window). The default time unit is a ps and

you may be representing your input stimulus in terms of ns. This means that at the end of the simulation the waveform may be at a point and scale where it does not illustrate any useful information. To change this you can use the *pan* and *zoom* options. You can *pan* to different points in your waveform by using the left and right scroll bars at the bottom of the window. You can *zoom* in and out using the and Icons located on the toolbar that is above the waveform display. There are also many other useful zoom type viewing zoom options which are highlighted in Table 4.2 below:

Table 4.2: Waveform Zoom Options

Zoom Command	Description	Key Toolbar Icon
Zoom In	View a smaller range of time on display (more detail)	I or
Zoom Out	View a larger range of time values on display (less detail)	O or
Zoom Full	Fits the entire waveform into the display area	F or
Zoom Cursor	Zoom in centered around cursor line location on waveform	C or
Zoom Last	Restore last zoom point	L
Zoom Range	View in the display area a specified time value range (user supplied start point and end point)	R

The waveform for the **comb_logic** example is shown in Figure 4.10

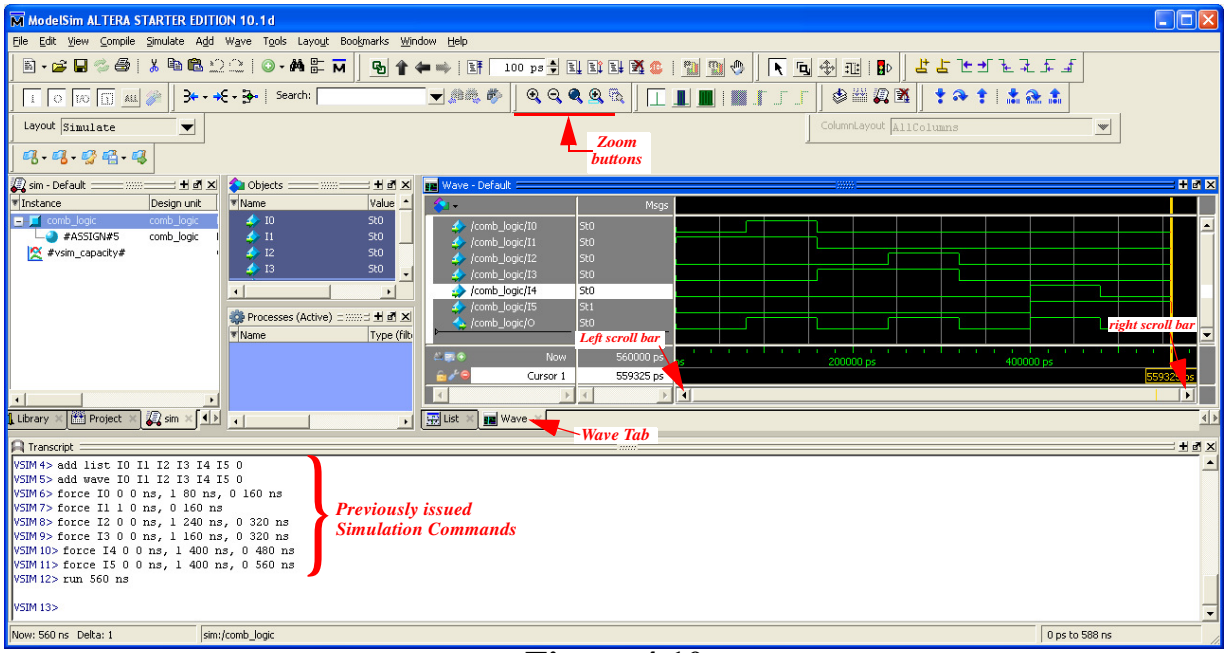


Figure 4.10

It should be noted here that for this purely functional simulation the output **O** responds to the six inputs **I0 - I5** in an "ideal" manner in that all propagation delays associated with the logic and wiring is assumed to be zero. Register Transfer Level (RTL) Simulations such as this only reflect the expected functional behavior of the system and does not take timing into account.

To view this information in a textual form you should click on the *List* tab as shown in Figure 4.11 (or activate this window by left clicking on it if you undocked it from the main Modelsim® window). Along the top of the window you will see the specific pin/signal

names in the same order that you expressed in the ***add list*** statement before you ran the simulation. Small arrows from these point to the column that is associated with each signal. The left side of the window represents the time in ps that the signal takes on these values. The default for this window is to produce a new output line each time one or more of the signals that were specified in the ***add list*** command has an event. An event is defined as when that signal actually changes its state from one value to another (such as going from a logic 0 to 1). Also along the left hand side is a column that is labeled delta. This represents the specific delta delay that is associated with the signal obtaining its value. A *delta delay* is a zero weighted value that is used internally by the event driven simulator to allow a sequential simulation algorithm to simulate the concurrent operation associated with real hardware. The actual value that the pin/signal will take on is the one associated with the final (highest number) delta delay.

The list output for the ***comb_logic*** example is shown in Figure 4.11

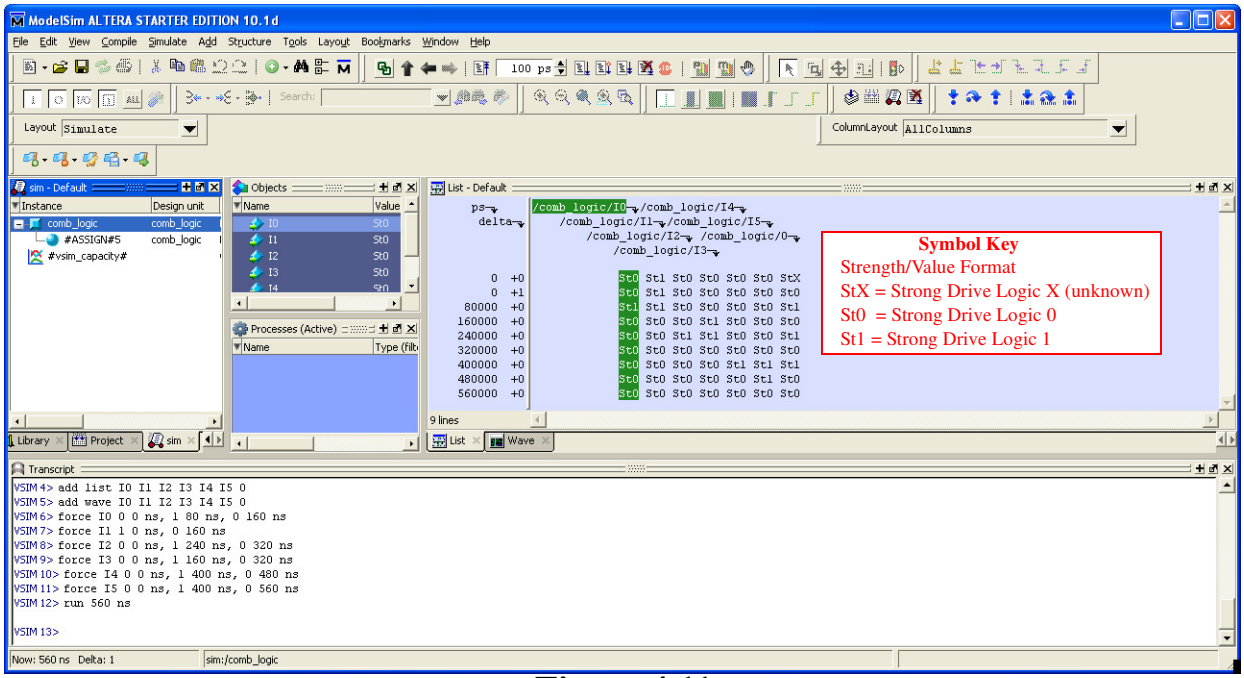


Figure 4.11

In this listing we see one logic strength and three different types of logic values for the **O** output. These strengths and values correspond to the Verilog standard. The logic strength for O in this simulation is always a *St* which indicates that a strong logic drive at the specified logic value. This strength is the next to highest, with the highest being the power supply. It overrides any other strength which is of a lower level such as capacitive or pull up strengths.The logic values displayed include the standard Logic ‘0’ and Logic ‘1’ false and true Boolean values. It also includes at the beginning of the simulation an ‘X” which in this case indicates

that the signal at the beginning was uninitialized. In general there are seven levels of logic strengths and four logic values. These are highlighted in Table 4.3.

Table 4.3: Possible Strengths and Logic Values

Logic Strengths

Highest ↑

Level	Strength	Strength Indicator
7	Supply Drive	Su
6	Strong Drive	St
5	Pull Drive	Pu
4	Large Capacitive	La
3	Weak Drive	We
2	Medium Capacitive	Me
1	Small Capacitive	Sm
0	High Impedance	HiZ

↓ Lowest (0 = no strength)

Logic Values

Description	Logic Value Indicator
zero, low, or FALSE	0
one, high, or TRUE	1
high impedance (tri-state or floating)	z or Z
unknown or uninitialized	x or X

It should be noted that the textural listing actually gave us more information than the wave-form one but is harder to read. In cases where one desires only to see the final values of each signal at a particular point in time and not display the artificial delta delays then one can change the list preferences. To do this first click on the *List Window* or the *List* tab. Then go to the *List* menu that is located above the top tool bar and chose the *List Preferences* option as shown in Figure 4.12

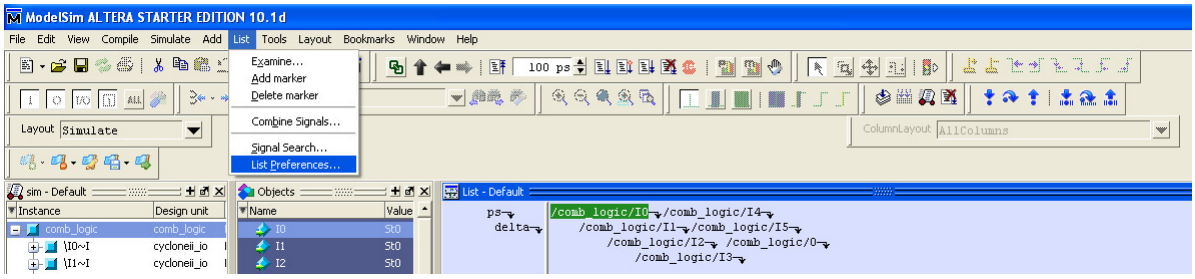


Figure 4.12

This will bring up the *Modify Display Properties Window* as shown in Figure 4.13.

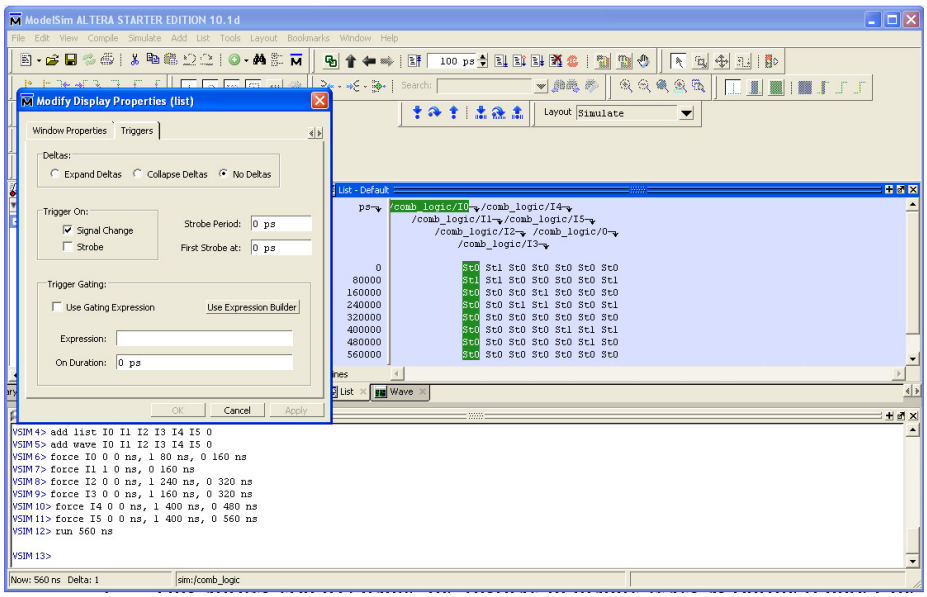


Figure 4.13

This allows you to change the display in many ways as outlined under the *Window Properties* and *Triggers* tab. The display shown in Figure 4.12 was made with the *Expanded Deltas* option turned on. To eliminate the appearance of these delta delays in your listing you can select the no deltas option that is under the *Triggers* tab under the *Modify Display Properties Window* and then left-click on the *Apply* button. The result of this compressed listing is also shown in Figure 4.13

4.2 Gate-level Timing Simulation using ModelSim® -- an Overview

Gate-level simulations not only demonstrate the logical aspects of a design but are also used to evaluate the operation of the design under the presence of the real-world delays that result form capacitive effects that cause the internal logic components and the wiring paths not to respond immediately when their stimulus is changed. This type of simulation is a more realistic view of the actual implementation but is much more complex, requiring longer simulation execution times than pure functional simulation. The timing delays associated with the paths and the components come from the actual characteristics of the physical implementation of the final design when it is implemented within its final targeted hardware environment. In the case of FPGA's these delays include the effects of the internal place and route operations. In the Quartus II/ModelSim® environment, gate-level timing simulations are performed in almost identical manner as the RTL simulations discussed in Section 4.1. In this section we will highlight how an full gate-level timing simulation can be performed on the simple combinational design that was successfully entered in Section II of this document. These steps are outlined below:

1 *Entering ModelSim® from Quartus II®.* After you have entered your design and successfully compiled it as discussed in Section III* (i.e. no Error Messages and the Warning Messages that were generated make sense within the framework of your design) then you are ready to enter the ModelSim® simulation environment to simulate it. To enter the ModelSim® environment left-click on the RTL Simulation Icon, on the tool panel as shown in Figure 4.14 or from the *Tools* menu select the *Run Simulation Tool* submenu and then select the *RTL Simulation* option.

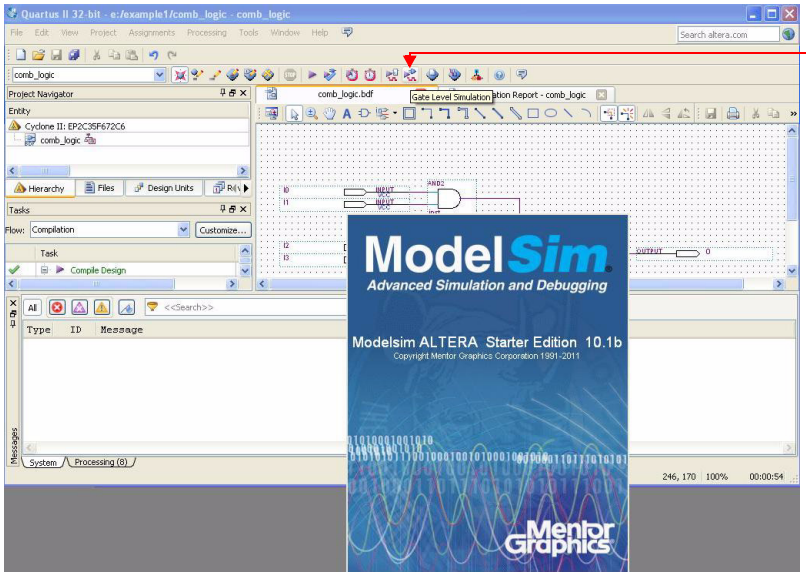


Figure 4.14

The first time you enter either a Gate-Level Simulation a dialog box may appear asking which simulation language you desire. If this occurs chose *Verilog HDL* as your simulation language as shown in Figure 4.15. Then click on the *OK* button to continue.

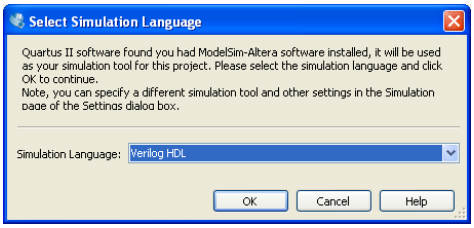


Figure 4.15

Before ModelSim® is launched *Quartus II* will ask you in the *EDA Gate Level Simulation* window to select the timing model you desire. Altera has provided three such timing models for the Cyclone IV E EP4CE115F29C7 FPGA that is being used on the DE2-115 rapid prototyping board. Internal FPGA timing is affected by such parameters as the fabrication process, voltage level, and the operating temperature. These three timing models were selected by Altera to represent extreme conditions in the multi-dimensional FPGA parameter space.Your design should be robust enough to function correctly for all three models. To proceed select the timing model you desire as shown in Figure 4.16 and then left-click on the *Run* button.

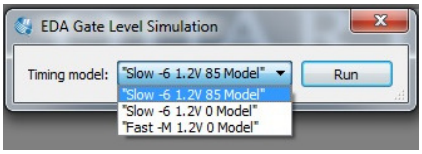


Figure 4.16

The next window should be the *ModelSim Simulator Window* as shown in Figure 4.17 below.

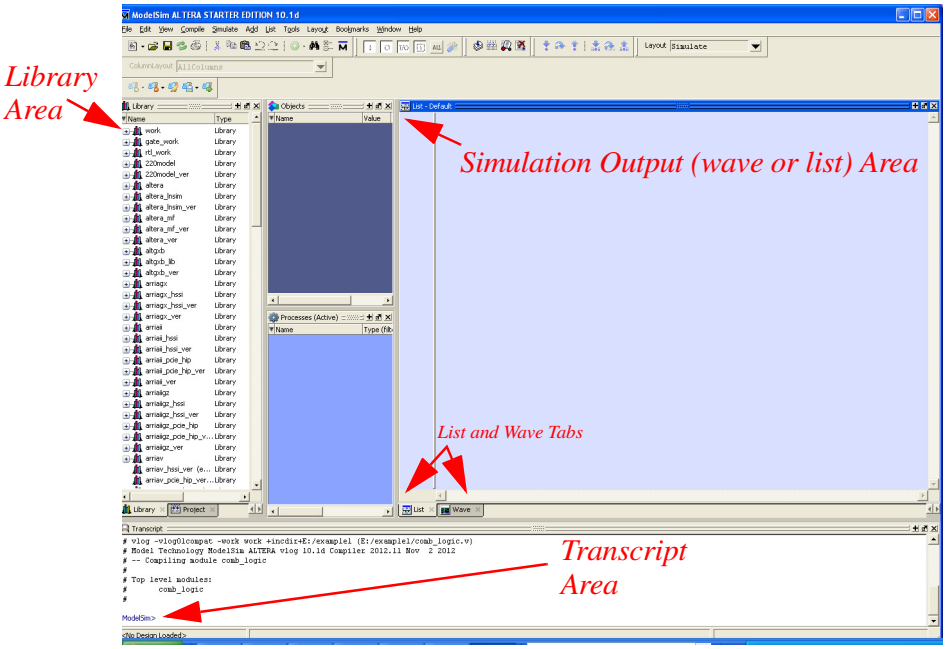


Figure 4.17

During this process various generic packages and libraries will be automatically added to your simulation workspace. This also starts the simulation elaboration process for your design that you entered using Quartus II. By default, the ModelSim® program is set up to

allow you to view the underlying Verilog HDL objects. This allows you to better understand the overall hierarchical structure of your design and its associated libraries. The two major areas that we will be using in this tutorial are the *Simulation Output Window* and the *Transcript Window*. The Simulation Output Window is where ModelSim® directs the simulation output. This output can be viewed in a textual list format or as a graphical waveform. These output areas are accessible by clicking on the corresponding *List* and the *Wave* tabs. The *Transcript Area* is where information about the simulation is provided by ModelSim® and where you can enter simulation commands. Most of the commands also have an equivalent sequence of menu operations. The base simulation excitation commands are presented in this tutorial, instead of the menu command sequences because of their portability across different versions of ModelSim®.

In should also be noted that each of these windows are initially docked in place in the manner shown in Figure 4.17 but they can be undocked at any time by clicking on the Icon. If you undock them they will form separate windows. In this case be cautioned to make sure you know where in your workspace these windows reside. Experience has shown that they can sometimes get lost or forgotten because the system does not automatically bring them to the front when data is updated. The next step is to initialize the simulation. This is done by selecting the *Start Simulation* option from the *Simulation* menu that is located on the ModelSim® toolbar as shown in Figure 4.18.

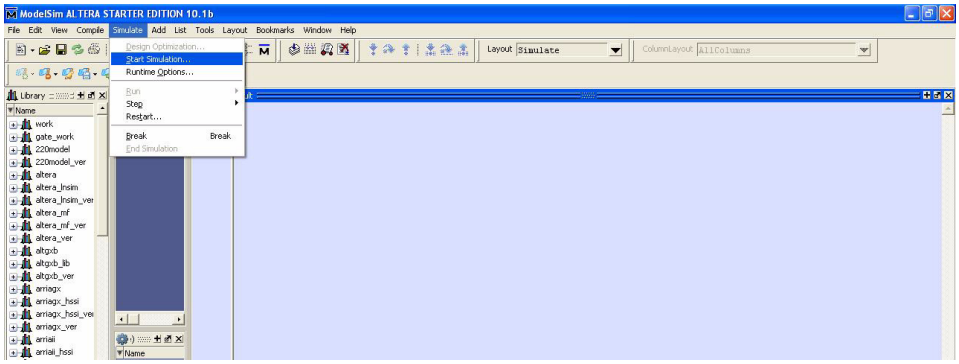
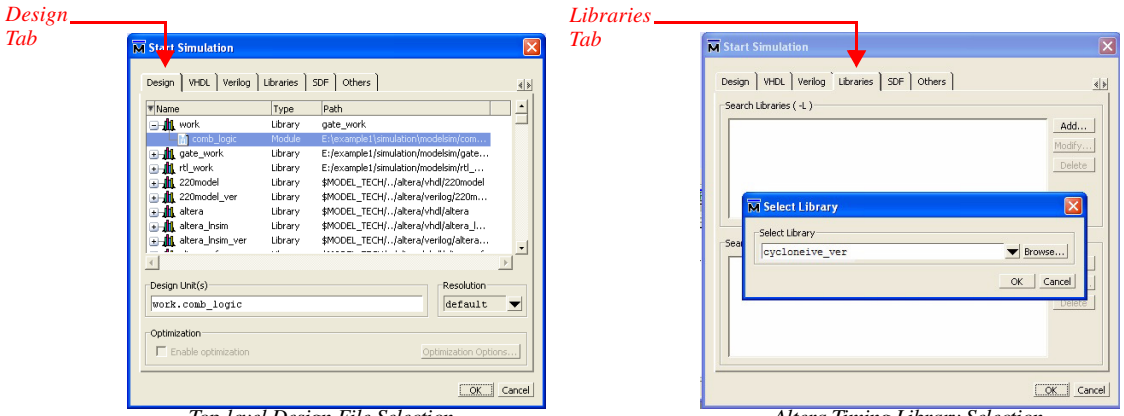


Figure 4.18

2 *Simulation Initialization.* This brings up the *Start Simulation Window* that allows you to select the top-level element of your design. This element is under the *Design* tab under the *work* library as shown in Figure 4.19. Expand *work* to reveal the sublibrary components. You should see a name that is the same as your top-level module which is **comb_logic** for the example we have been discussing. Left-click on this entity. Then select the *Libraries* Tab and click on the *Add..* button that is on the right side of the *Search Libraries (L)* portion of the window. This should launch the *Select Library* dialog box. Enter **cycloneive_ver** which is the ModelSim® timing library for the Altera Cyclone IV E family of devices one of which corresponds to the FPGA device used on the DE2-115 board. After this is done left click the *OK* button on both the *Select Library* dialog box and the *Start Simulation Window.*



Top-level Design File Selection *Altera Timing Library Selection*

Figure 4.19

This executes the ModelSim® simulation *vcom* compiler/elaborator for your design and places your design in the default work library. It then starts the underlying *vsim* simulator which operates in an interactive mode throughout the simulation process. The simulation is now ready to accept additional simulation commands that you can give by typing them in the *Transcript Area* as shown in Figure 4.20.

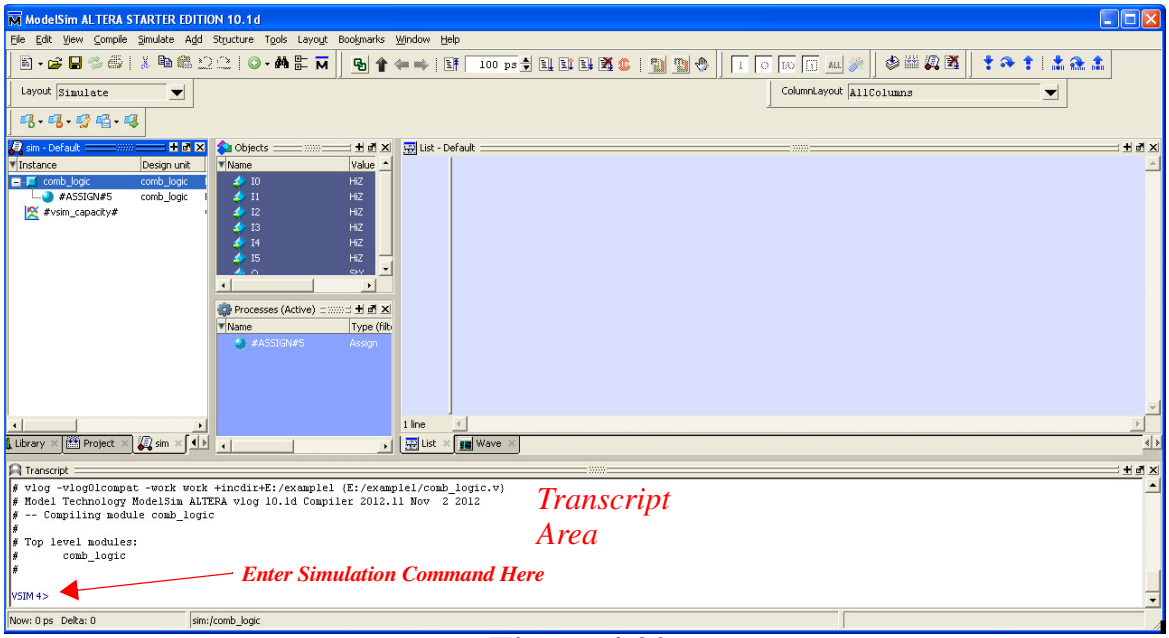


Figure 4.20

3 *Pin and Signal Monitoring*. Before you actually execute the simulation (and possibly as the simulation progresses) you need to determine which pins or signals you want to observe. This will allow the simulator to record only the important activities and events that are pertinent. In most real-world scenarios recording and viewing all possible internal events in not practical or desirable. To select the signals you would like to view you can use the ModelSim® *add* commands. The *add list* command adds the subsequent set of pin name and signals so that they will be monitored in textural form in the *List* window. The *add wave* command selects the set of signals and other objects to be displayed in a similar manner but allows them to be viewed as a waveform in the *Wave* window. To monitor the input and output signals in as a waveform and as textural data you can issue both the *add list* and *add wave* commands before you run your simulation. To do this you need to place the set of I/O

pin names of your top-level schematic or the signal names of your top-level module of your Verilog HDL model. Each of these names should be separated by spaces and the order that you place them is the order they will appear in your waveform and textual listing. For the **comb_logic** example the following two commands can be entered on separate lines in the *Transcript Window* at the VSIM prompt to monitor the six inputs and one output.

add list I0 I1 I2 I3 I4 I5 O
add wave I0 I1 I2 I3 I4 I5 O

If you enter a name that does not represent a valid object a red error message will be interactively posted in the *Transcript Window* to alert you of your mistake. To correct it simply enter a correct *add* command.

4 Stimulus Generation. The next step is to drive the inputs of your simulation in a manner that allows you to evaluate it. This can be done with the *force* command. The first argument of the force command is the name of the I/O pin or signal object. The next set of arguments represent the logic value you desire the pin or signal to take on followed by the time, relative to the current point in the simulation, that it should take on that value. If a time is not entered then it takes on the value specified from the current point in time and holds that value throughout the simulation run. The time should be expressed as an integer followed by the appropriate time unit (such as ms, us, ns, or ps). Each logic value/time pair should be separated from one another with commas.
For example, if the desired input stimulus for the six inputs in our **comb_logic** case is given by Table 4.2

Table 4.4: Desired Input Stimulus for *comb_logic* Example

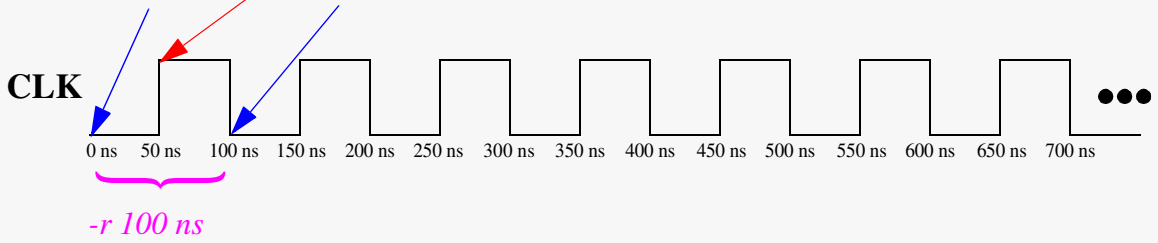
Input	0 ns	80ns	160 ns	240 ns	320 ns	400 ns	480 ns	560 ns
I0	0	1	0	0	0	0	0	0
I1	1	1	0	0	0	0	0	0
I2	0	0	0	1	0	0	0	0
I3	0	0	1	1	0	0	0	0
I4	0	0	0	0	0	1	0	0
I5	0	0	0	0	0	1	1	0

then the following set of *force* commands can be used to specify this stimulus.

force I0 0 0 ns, 1 80 ns, 0 160 ns
force I1 1 0 ns, 0 160 ns
force I2 0 0 ns, 1 240 ns, 0 320 ns
force I3 0 0 ns, 1 160 ns, 0 320 ns
force I4 0 0 ns, 1 400 ns, 0 480 ns
force I5 0 0 ns, 1 400 ns, 0 560 ns

Generating Clocks: Although the use of a clock signal is not needed in this example because it is a strictly combinational design it is important to note that the *force* command can also be used to generate periodic patterns such as a clock by using the -r option as shown below.

force CLK 0 0 ns, 1 50 ns, 0 100 ns -r 100 ns



Here the -r option specifies that the pattern that has been present for the past 100 ns (starting at time 100 ns) be repeated indefinitely giving a 50% duty cycle clock with a period of 100 ns that will run the duration of the simulation.

In general, input stimulus can be created using these ModelSim® commands or within the hardware description language (such as Verilog or VHDL) itself. One advantage to using the ModelSim® commands are that they are portable across designs that employ different design capture methods (schematic and HDLs). A major disadvantage is that hardware description languages are much more robust in terms of the degree to which stimulus patterns can be made to be responsive to simulation output and can make use of built in constructs that enhance timing verification. After stimulus generation the next step is to actually run the simulation.

5 *Running the Simulation*. To finally run the simulation you should enter the *run* command in the *Transcript Area*. The argument for this command is an integer that specifies how long the simulation is to execute from its current point before the simulation. During simulation runs no additional simulation commands can be executed but once the simulation is over it is placed in the interactive mode allowing it to accept new simulation commands. (It should be noted that the simulator is event driven and it is possible for the simulation to end its execution before the specified end time if it runs out of events.)
For the **comb_logic** example the command to execute the simulation for 560 ns from the current point in time is simply

run 560 ns

When the simulation is complete, the results can be view and analyzed, and new simulation commands can be entered. These commands always take effect at the current point in simulation time unless the simulation is reset. You must reset a simulation in order to start again from the beginning. To do this you can enter the *restart -f* command in the *Transcript Area*. reset a simulation. If a design change is needed then one should exit ModelSim® and then make the necessary changes in the Quartus II®, recompile the design and then reenter the simulation in the same manner as before.

Viewing your Simulation Results: After the simulation is run you will want to view the results. To view as a waveform timing diagram make sure to select the *Waveform* window by clicking on the *Wave* tab as shown in Figure 4.10 (or activate this window by left clicking on it if you undocked it from the main ModelSim® window). The default time unit is a ps and

you may be representing your input stimulus in terms of ns. This means that at the end of the simulation the waveform may be at a point and scale where it does not illustrate any useful information. To change this you can use the *pan* and *zoom* options. You can *pan* to different points in your waveform by using the left and right scroll bars at the bottom of the window. You can *zoom* in and out using the and Icons located on the toolbar that is above the waveform display. There are also many other useful zoom type viewing zoom options which are highlighted in Table 4.5 below:

Table 4.5: Waveform Zoom Options

Zoom Command	Description	Key Toolbar Icon
Zoom In	View a smaller range of time on display (more detail)	I or
Zoom Out	View a larger range of time values on display (less detail)	O or
Zoom Full	Fits the entire waveform into the display area	F or
Zoom Cursor	Zoom in centered around cursor line location on waveform	C or
Zoom Last	Restore last zoom point	L
Zoom Range	View in the display area a specified time value range (user supplied start point and end point)	R

The waveform for the **comb_logic** example is shown in Figure 4.21. This is the waveform produced by the Altera -6 slow 0 model that was chosen in quartus when ModelSim® was launched by Quartus II®.

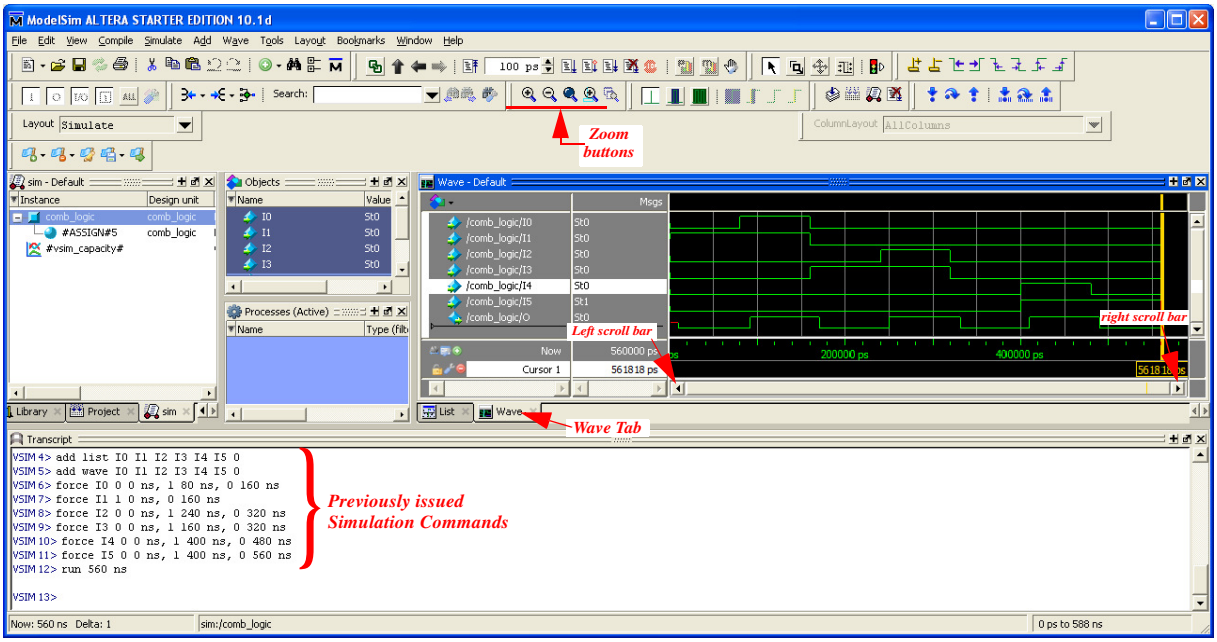


Figure 4.21

It should be noted here that the propagation delays in the output **O** at various points in time are evident in relationship to the six stimulus inputs **I0 - I5**. This differs from the response of the Register Transfer Level (RTL) Simulations which does not take timing into account but assumes zero component and wiring delays. These delays came from the Altera/ModelSim® library which in coordination with the FPGA layout "*back annotated*" these values into the simulation.

To view this information in a textual form you should click on the *List* tab as shown in Figure 4.22 (or activate this window by left clicking on it if you undocked it from the main ModelSim® window). Along the top of the window you will see the specific pin/signal names in the same order that you expressed in the ***add list*** statement before you ran the simulation. Small arrows from these point to the column that is associated with each signal. The left side of the window represents the time in ps that the signal takes on these values. The default for this window is to produce a new output line each time one or more of the signals that were specified in the ***add list*** command has an event. An event is defined as when that signal actually changes its state from one value to another (such as going from a logic 0 to 1 or from 1 to 0). Also along the left hand side is a column that is labeled delta. This represents the specific delta delay that is associated with the signal obtaining its value. A *delta delay* is a zero weighted value that is used internally by the event driven simulator to allow a sequential simulation algorithm to simulate the concurrent operation associated with real hardware. The actual value that the pin/signal will take on is the one associated with the final (highest number) delta delay.

The list output for the ***comb_logic*** example is shown in Figure 4.22

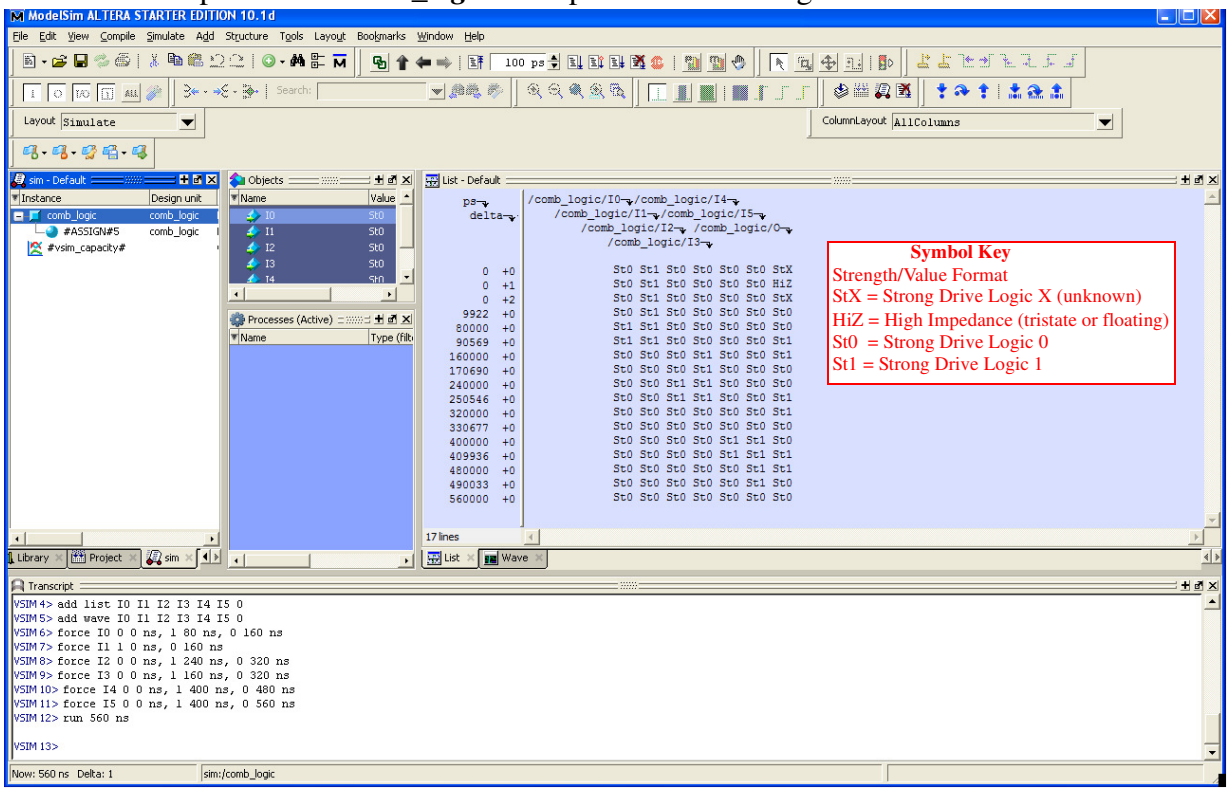


Figure 4.22

In this listing we see two logic strengths and four different types of logic values for the **O** output. These strengths and values correspond to the Verilog standard. The logic strength for **O** in this simulation is usually a *St* which indicates that a strong logic drive at the specified logic value. This strength is the next to highest, with the highest being the power supply. This strength will override any other strength which is of a lower level such as capacitive or pull up strengths.The second strength value is the *Hi* strength, which is the high impedance strength level. This level is the weakest of all strength levels. This level appears as an artifact of the simulation process at the second delta delay during initiation and never represents a

final output. The logic values displayed include the standard Logic ‘0’ and Logic ‘1’ false and true Boolean values. It also includes at the beginning of the simulation an ‘X” which in this case indicates that the signal at the beginning was uninitialized. A logic value of ‘Z’ also appears for one delta delay at the beginning of the simulation. This value indicates a high impedance tri-state or floating value. In general there are seven levels of logic strengths and four logic values. These are highlighted in Table 4.6.

Table 4.6: Possible Strengths and Logic Values

Logic Strengths

Highest

Level	Strength	Strength Indicator
7	Supply Drive	Su
6	Strong Drive	St
5	Pull Drive	Pu
4	Large Capacitive	La
3	Weak Drive	We
2	Medium Capacitive	Me
1	Small Capacitive	Sm
0	High Impedance	HiZ

Lowest
(0 = no strength)

Logic Values

Description	Logic Value Indicator
zero, low, or FALSE	0
one, high, or TRUE	1
high impedance (tri-state or floating)	z or Z
unknown or uninitialized	x or X

It should be noted that the textural listing actually gave us more information than the wave-form one but is harder to read. In cases where one desires only to see the final values of each signal at a particular point in time and not display the artificial delta delays then one can change the list preferences. To do this first click on the *List Window* or the *List* tab. Then go to the *List* menu that is located above the top tool bar and chose the *List Preferences* option as shown in Figure 4.23

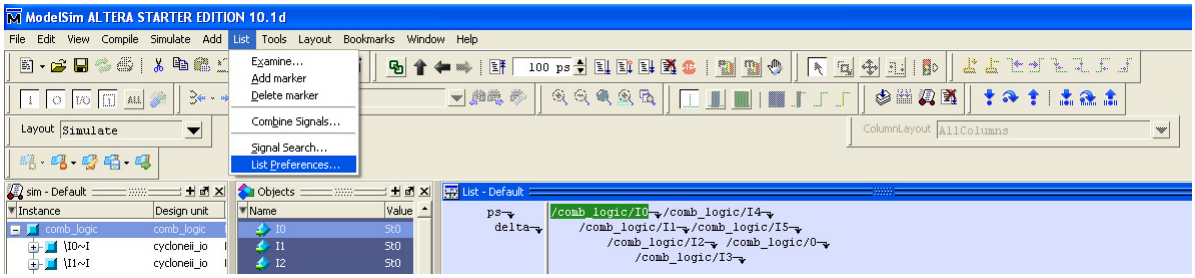


Figure 4.23

This will bring up the *Modify Display Properties Window* as shown in Figure 4.24.

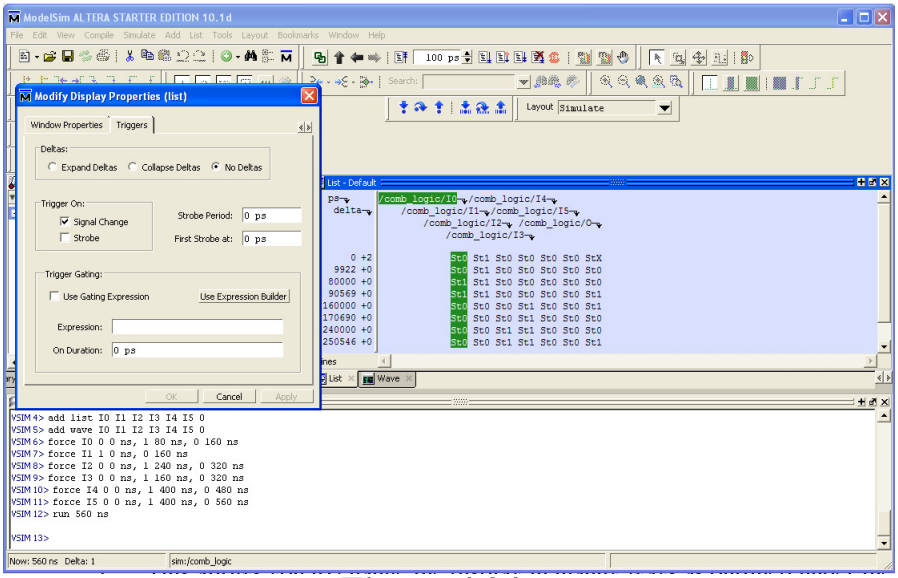


Figure 4.24

This allows you to change the display in many ways as outlined under the *Window Properties* and *Triggers* tab. The display shown in Figure 4.24 was made with the *Expanded Deltas* option turned on. To eliminate the appearance of these delta delays in your listing you can select the no deltas option that is under the *Triggers* tab under the *Modify Display Properties Window* and then left-click on the *Apply* button. The result of this compressed listing is also shown in Figure 4.24.

V. Configuring the Cyclone IV E FPGA on the Terasic DE2-115 Rapid Prototyping Platform

It is often desirable to actually implement the design in programmable logic. The DE2-115 Rapid Prototyping Board contains a Field Programmable Gate Array, FPGA, that will can be used to implement digital designs that range in size from a few gates to designs that are as complex as a system of embedded microprocessors. To illustrate this let us consider again the **comb_logic** example.

In this example, the inputs to the design can be driven, using external switches, and the output can be made to drive a single discrete LED. The DE2-115 board documentation specifies the pin numbers of the Cyclone IV E FPGA that are connected to the switches and the LEDs. A valid configuration for the **comb_logic** example might be to bind design pins **I0** through **I5** to Switch 0 through Switch 5 (*SW0* -- *SW5*) of the DE2-115 board and connect the output, *O*, to the green LED number 8 (LEDG8).

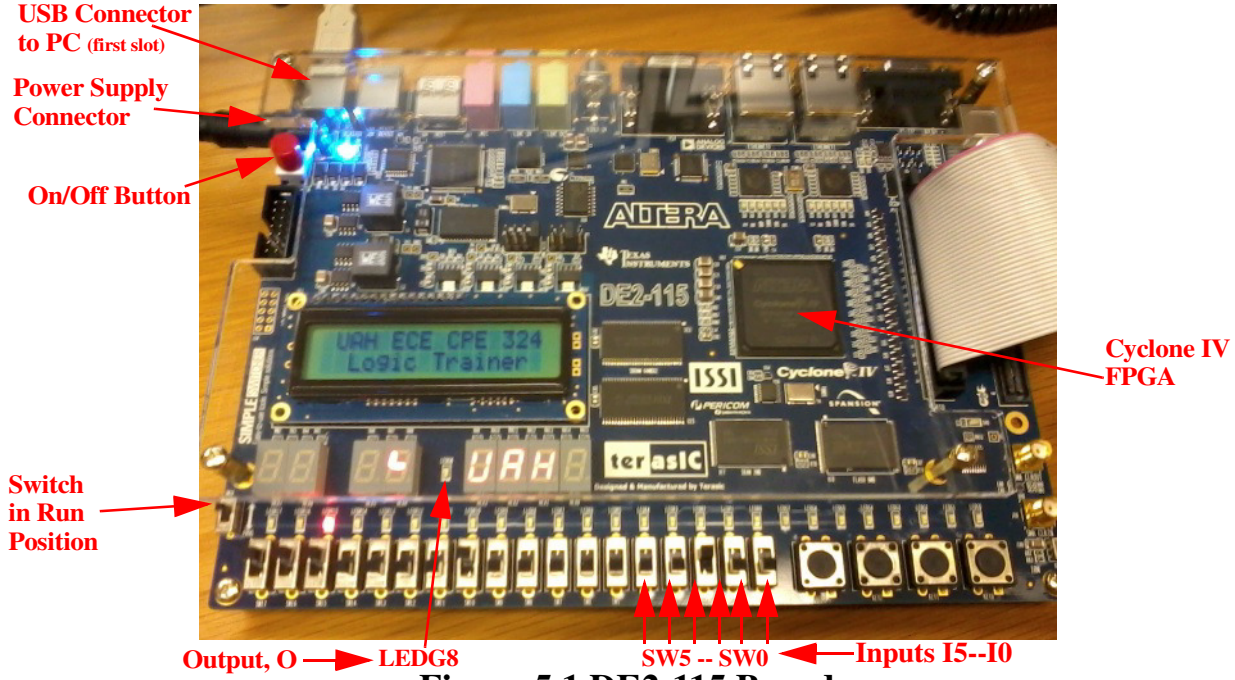


Figure 5.1 DE2-115 Board

1 To enable such a configuration of the DE2-115 board, one must first plug in its power supply cable, and plug in the USB cable to the port marked in Figure 5.1 (the other end of the cable must be inserted into the USB port of the computer that is running the Quartus II software). It is also important to make sure that the *On/Off* button is in the *On* position and that the *Run/Prog* switch is in the *Run* position.

Before the design can be downloaded into the Cyclone IV E FPGA on the DE2-115, you must assign the logical names of the inputs and output in your design to the corresponding FPGA pins that are connected to the switches and the LED. The DE2-115 manual has a complete listing of all of the pin connections that have been made between the Cyclone IV E FPGA and the other peripherals that are present on the DE2-115 board. This information should be present under the CPE 324 Info folder that is on the desktop of the laboratory computers.

2 As part of the project setup at the beginning of this document, you entered the specific Cyclone IV E FPGA device that is present on the DE2-115. The next step is to enter the specific pin numbers on this FPGA which you want to associate the logical I/O pins in your design. To accomplish this, first select *Pin Planner*, from the *Assignments* menu as shown in Figure 5.2..

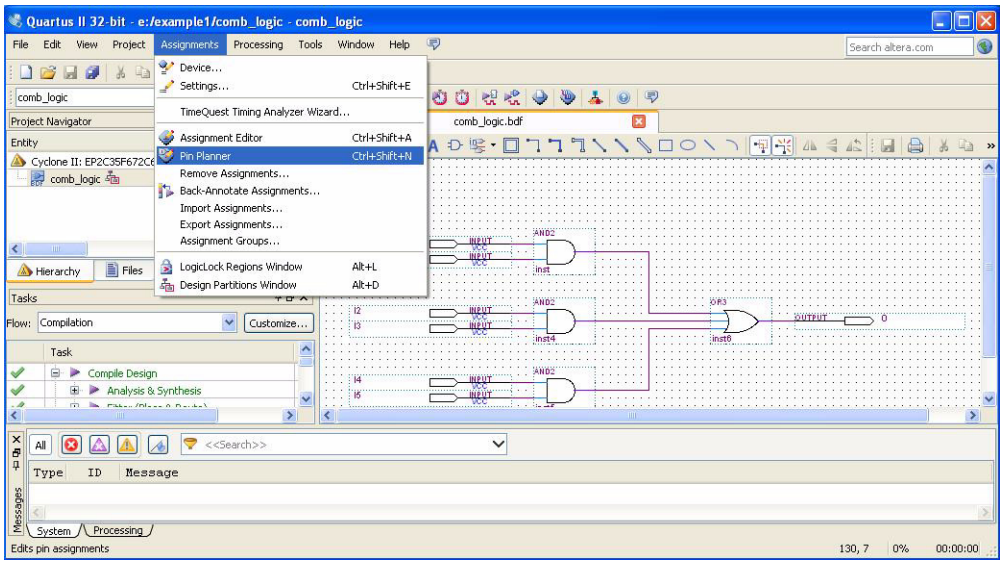


Figure 5.2

3 This will launch the *Planner Window.* as shown in Figure 5.3. The *Node Name* fields at the bottom of this window will contain the name that was assigned by you when naming the INPUTS and OUPUT connector components of your design. The *Location* field will be empty. This is where the user is allowed to place a specific external pin number of the FPGA that is to be associated with the logical input/output of the design. If the user does not enter a pin number, then the Quartus II software will select the pin numbers itself which is often undesirable.

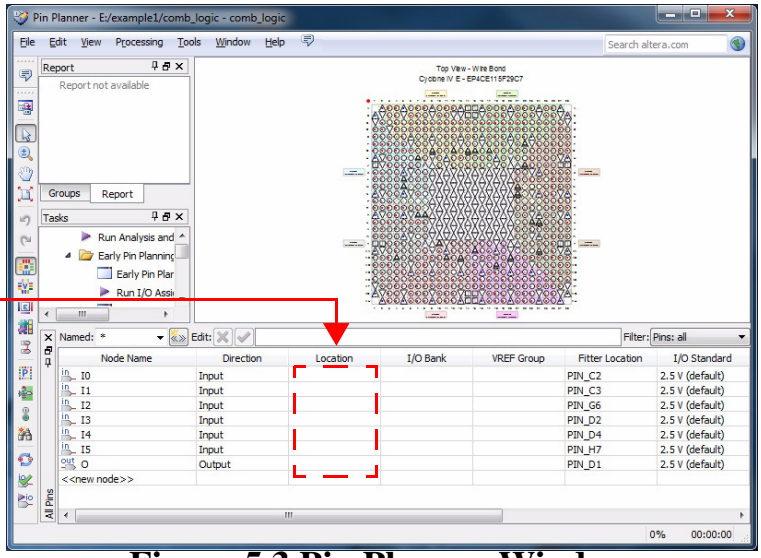


Figure 5.3 Pin Planner Window

4 In this example the pin numbers that correspond to the six switches, and the single LED are shown in Figure 5.4 after the correct FPGA pin numbers have entered on the *Pin Planner Window*. After entering these numbers you should then go back to the main Quartus II window and recompile the project.

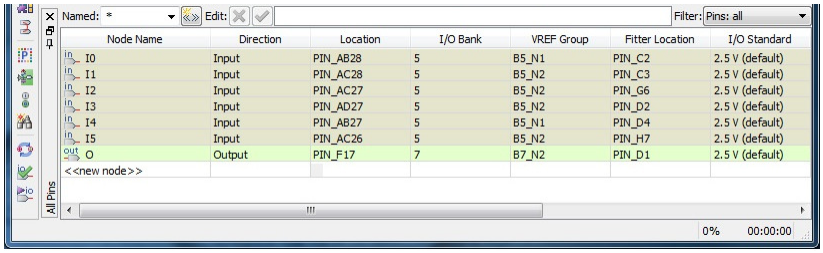


Figure 5.4 Pin Planner Window

5 To re-compile your project, left-click on the Icon or select the *Start Compilation* which can be found under the *Processing* menu. The compilation proceeds in the same manner as it did when the project was initially compiled. After the compilation has completed, the DE2-115 Cyclone II device can now be configured.

6 To configure the DE2-115 board with the example design, click on the Icon as shown in Figure 5.5, or select the *Programmer* option from the *Tools* menu.

FPGA Configuration (Programmer) Tool

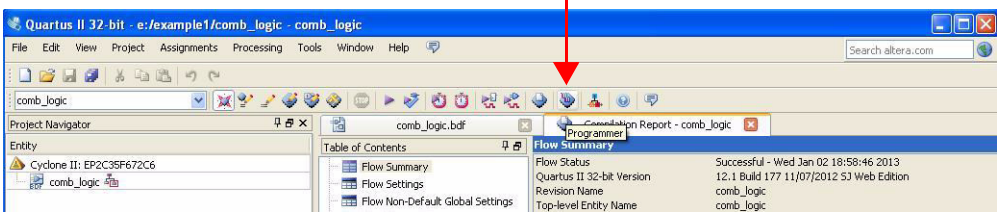


Figure 5.5 Invoking the Programmer Tool

This will launch a *Programmer Window* as shown in Figure 5.6.

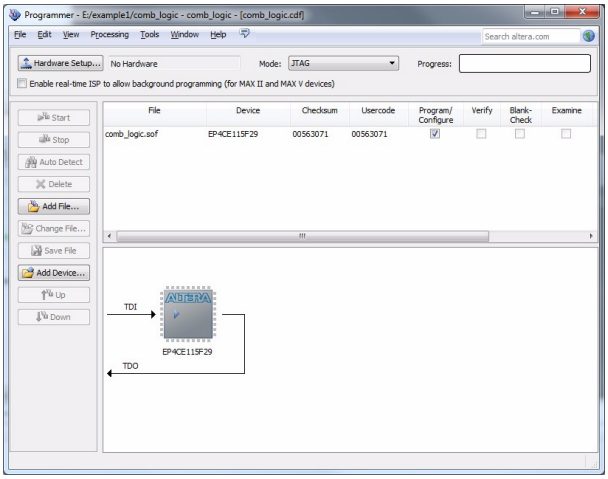


Figure 5.6 Programmer Window when no programming hardware is enabled

7 This window has a *File* field that should show the top-level file name of your project (but with a .sof extension). It also has a *Device* field that should match the EP4CE115F29 Cyclone IV E device that you have selected when you set up the project. There is also a *Program/Configure* box that should be checked. At the top of the window is a *Hardware Setup* button on the upper left-hand corner of the window. If the field next to it does not read *USB-Blaster,* then you will have to click on the *Hardware Setup* button before the DE2-115 can be programmed. (If the text next to the *Hardware Setup* button reads *USB-Blaster*, then the programming driver is already set up and does not need to be setup again. In this case, simply skip to step 9 and proceed).

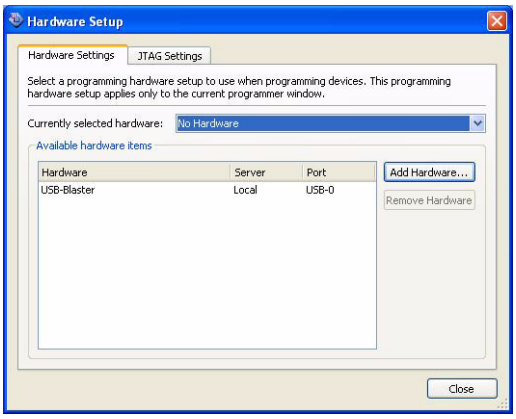


Figure 5.7 Hardware Setup Window

8 From the *Hardware Setup* window, choose, *USB-Blaster*, from the *Currently selected hardware:* list as shown in Figure 5.8. Then click on the *Close* button. This should result in the *Programmer* window now be displayed again.

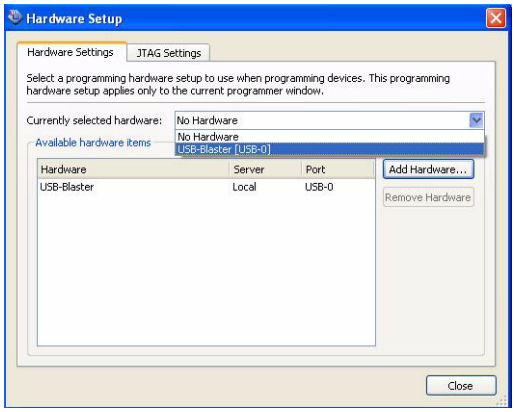


Figure 5.8 Selection of USB-Blaster

9 The *Programmer* window should now recognize the *USB-Blaster* and should have its *Start* button enabled. Make sure that the *Program/Configure* box is checked as shown in Figure 5.9 and that the DE2-115 board is turned on and properly connected to the PC through the USB cable as mentioned previously. Then click on the *Start* button to configure the DE2-115. The LEDs on the DE2-115 Board will flicker and after a few seconds the device should be configured. If there are problems with the configuration, error messages will appear in the status area at the bottom of the man Quartus II window. In this example, the design should now reside in the DE2-115's Cyclone IV E device, switches SW5-SW0 should control the design causing, LEDG8 to light up in the manner dictated by the Boolean function.

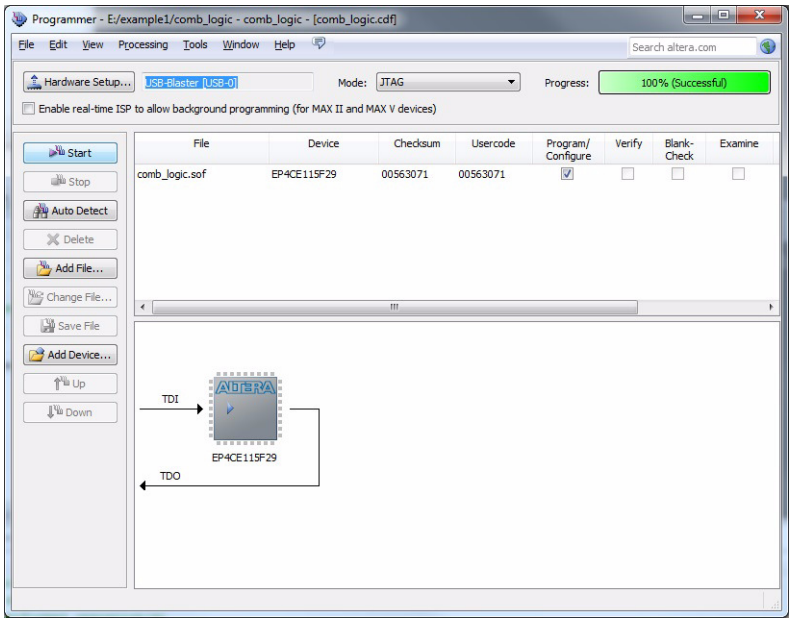


Figure 5.9 Programmer Window with USB-Blaster Enabled and successful DE2-115 Download