ECSE-323
Digital System Design

Lab #1 – ***Using the Altera Quartus II and ModelSim Software*** Fall 2015

Introduction

In this lab you will learn the basics of the *Altera Quartus II* FPGA design software through following a step-by-step tutorial, and use it to design two simple combinational logic circuits. You will also learn the basics of digital simulation using the *ModelSim* simulation program.

Learning Outcomes .

After completing this lab you should know how to:

- Startup the Altera Quartus II software
- Create the framework for a new project
- Layout a schematic block/gate diagram of a logic circuit
- Name and connect nodes and busses using the block diagram editor
- Design a simple combinational circuit using VHDL
- Do functional simulation of a circuit using ModelSim

Table of Contents .

This lab consists of the following stages:

1. Introduction

The lab portion of the course consists of 5 parts, each two weeks long. The bulk of the work will be done using the Altera Quartus II software running on the computers in the lab. Throughout the 5 lab experiments, you will develop all of the building blocks for the system, and integrate them into a complete user-friendly system, using an FPGA development board.

Each lab will have a number of items that the lab groups must demonstrate to a TA. The TA will then sign the appropriate entry on the lab grade sheet (the grade sheet can be found at the end of the lab description, and should be printed out). Items that needed to be demonstrated to the TAs are indicated in the lab description with the pencil-paper-checkmark icon. Once you have obtained all the necessary signatures on the grade-sheet, give it to one of the TAs, who will then pass it on to the course instructor for recording the grade.

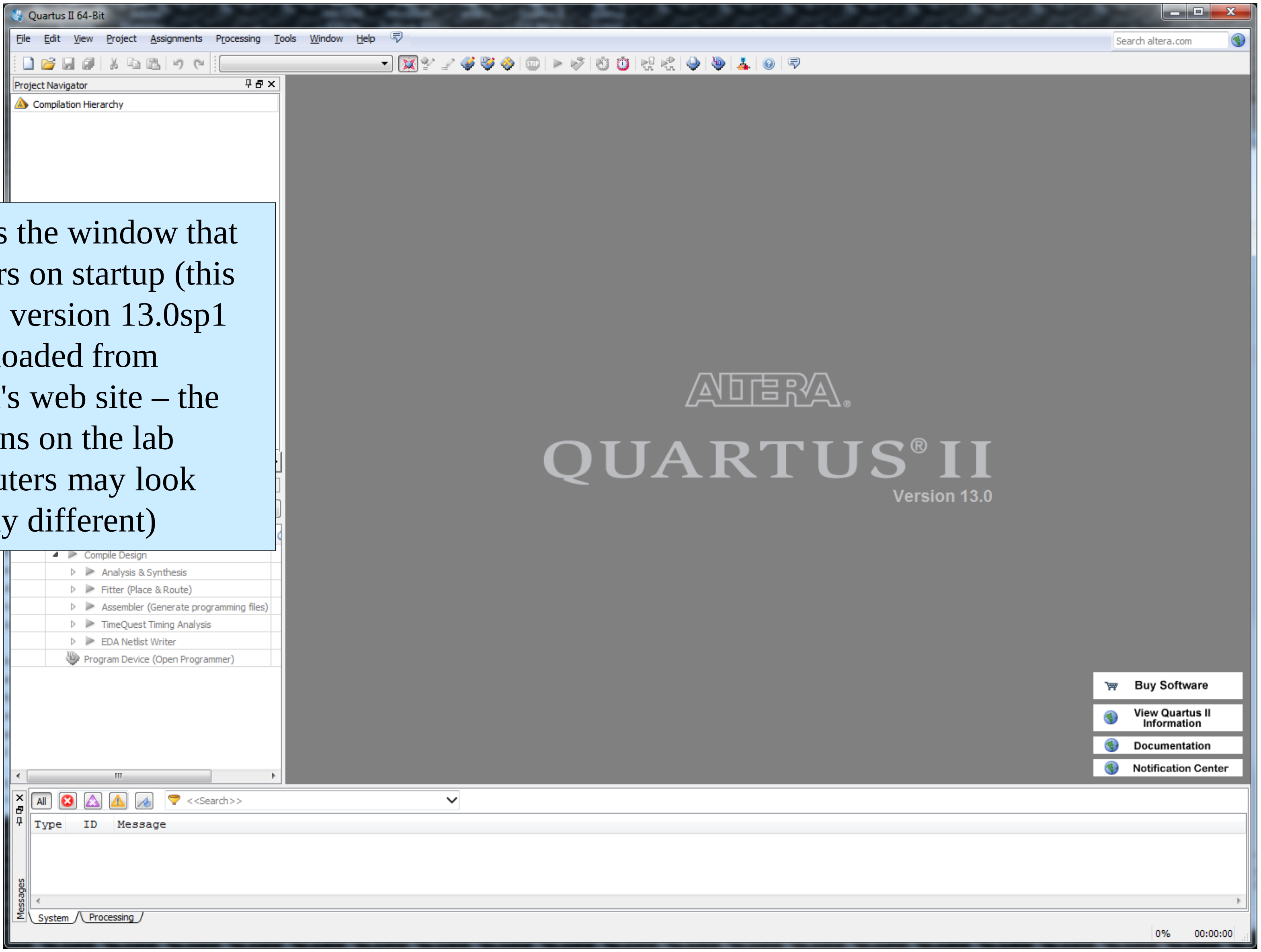
After each lab each group must submit, electronically, a lab report to the course myCourses web site. This report will generally describe the circuits designed during the lab.

2. Startup and creation of the project framework

In this course you will be using commercial FPGA design software, the Altera ***Quartus II*** program and the Mentor Graphics ***ModelSim*** simulation program. A slightly restricted version of the Quartus program is contained on the CD that comes with the course textbook, and can also be downloaded from the Altera web site (preferred, since the Altera site will have the required version). The program restrictions will not affect any designs you will be doing in the course, so you can install the program on your personal computer, so that you can work on your project outside of the lab. You should use version **13.0sp1** of the program, as this is the latest version that supports the prototyping board (the DE1) that you will be using in the lab.

The purpose of this first lab is to familiarize yourself with the Quartus program, by going through a step-by-step tutorial on its use. You will use it to develop two simple circuits using schematic capture techniques.

To begin, start the ***Quartus II*** program by either by selecting the program in the Windows Start menu or double-clicking on the desktop shortcut icon (if one exists).

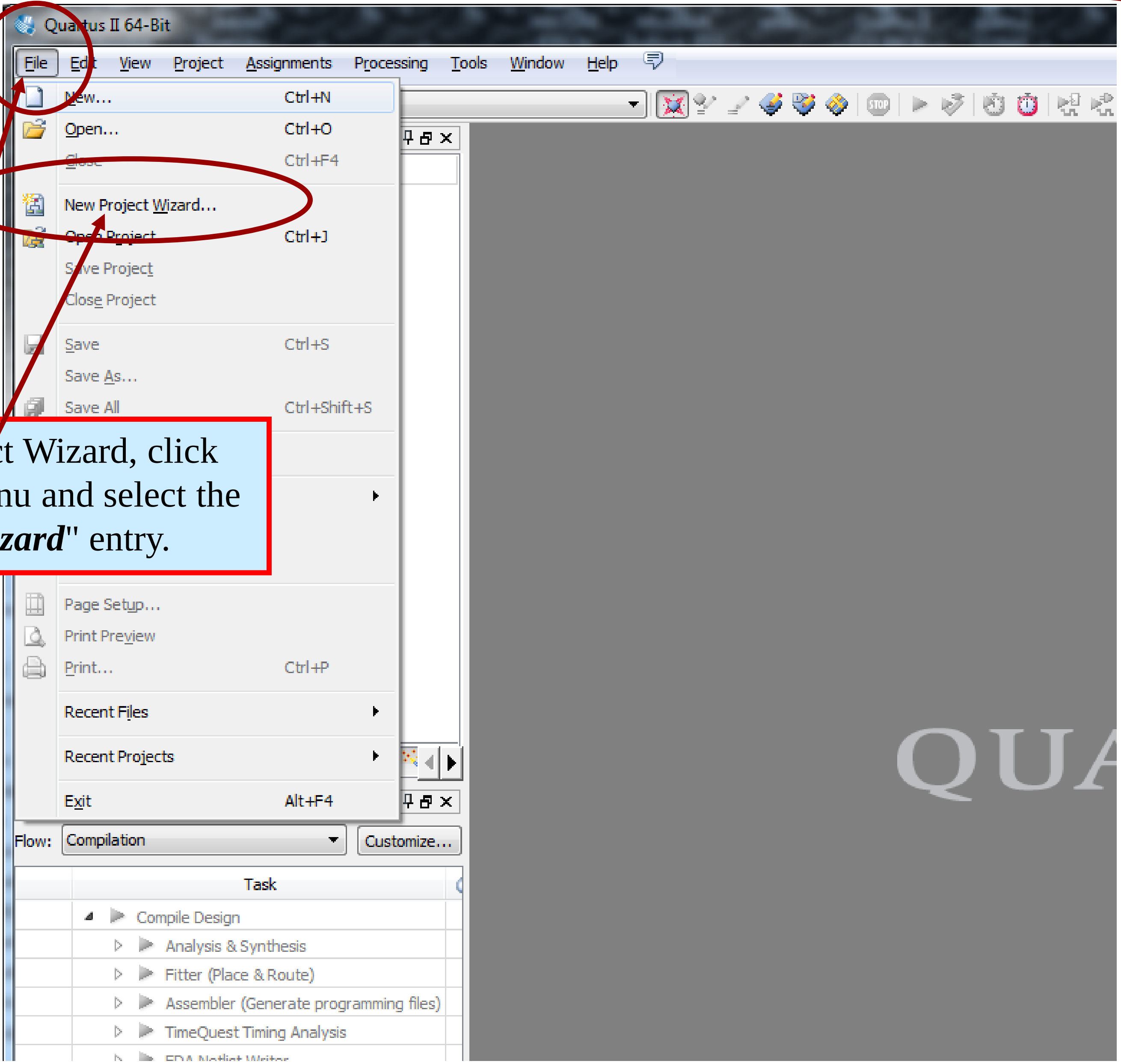


This is the window that appears on startup (this shows version 13.0sp1 downloaded from Altera's web site – the versions on the lab computers may look slightly different)

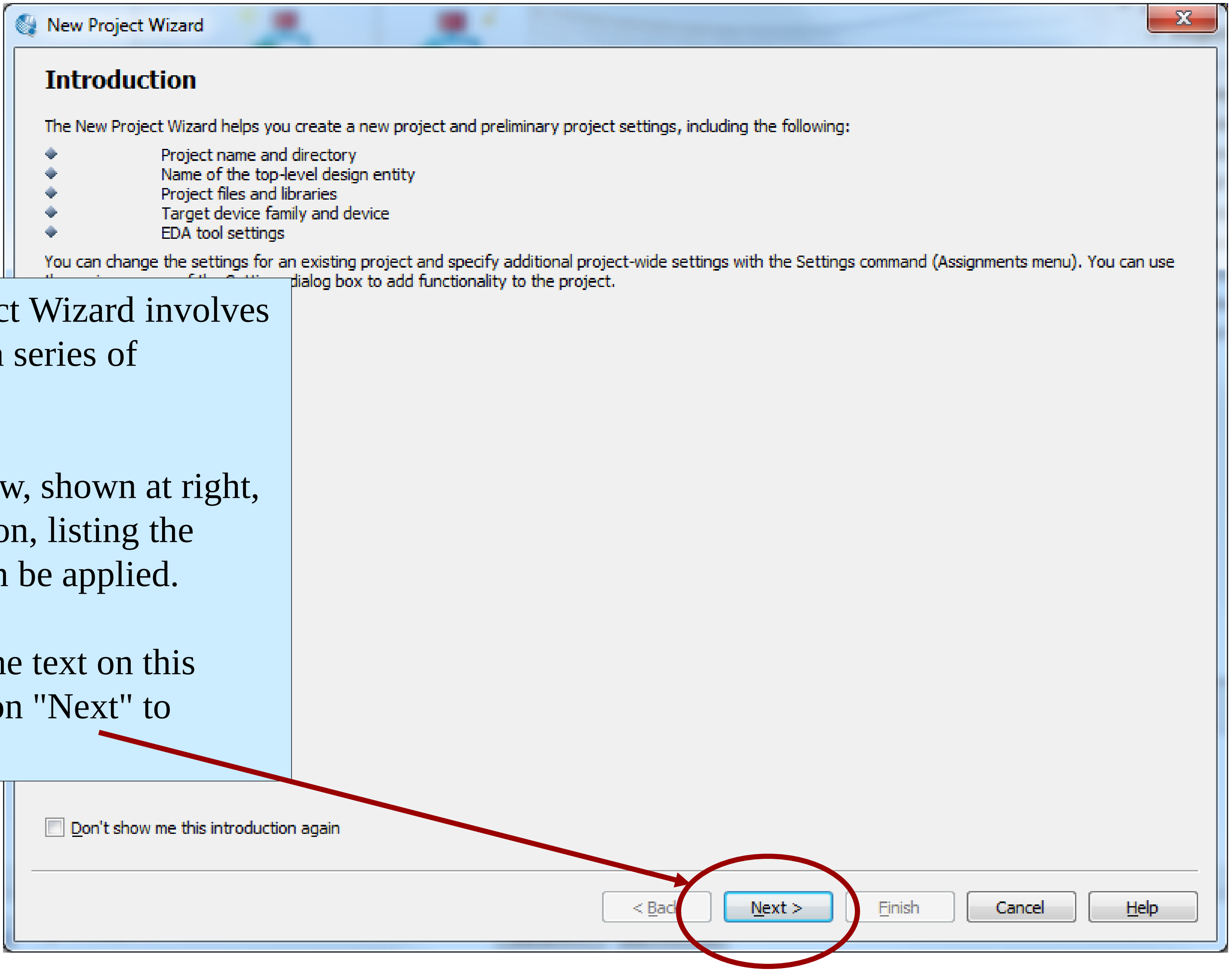
The Altera ***Quartus II*** program employs a ***project-based approach***. The goal of a Quartus project is to develop a hardware implementation of a specific function, targeted to an FPGA (*Field Programmable Gate Array*) device.

Typically, the project will involve a (large) number of different circuits, each designed individually, or taken from circuit libraries. Project management is therefore important. The Quartus II program aids in the project management by providing a project framework, that keeps track of the various components of the project, including ***design files*** (such as schematic block diagrams or VHDL descriptions), ***simulation files***, ***compilation reports***, ***FPGA configuration or programming files***, project specific program settings and assignments, and many others.

The first step in designing a system using the Quartus II approach is therefore to create the project framework. The program simplifies this by providing a "Wizard" which guides you through a step-by-step setting of the most important options.



To run the Project Wizard, click on the "***File***" menu and select the "***New Project Wizard***" entry.



New Project Wizard

Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

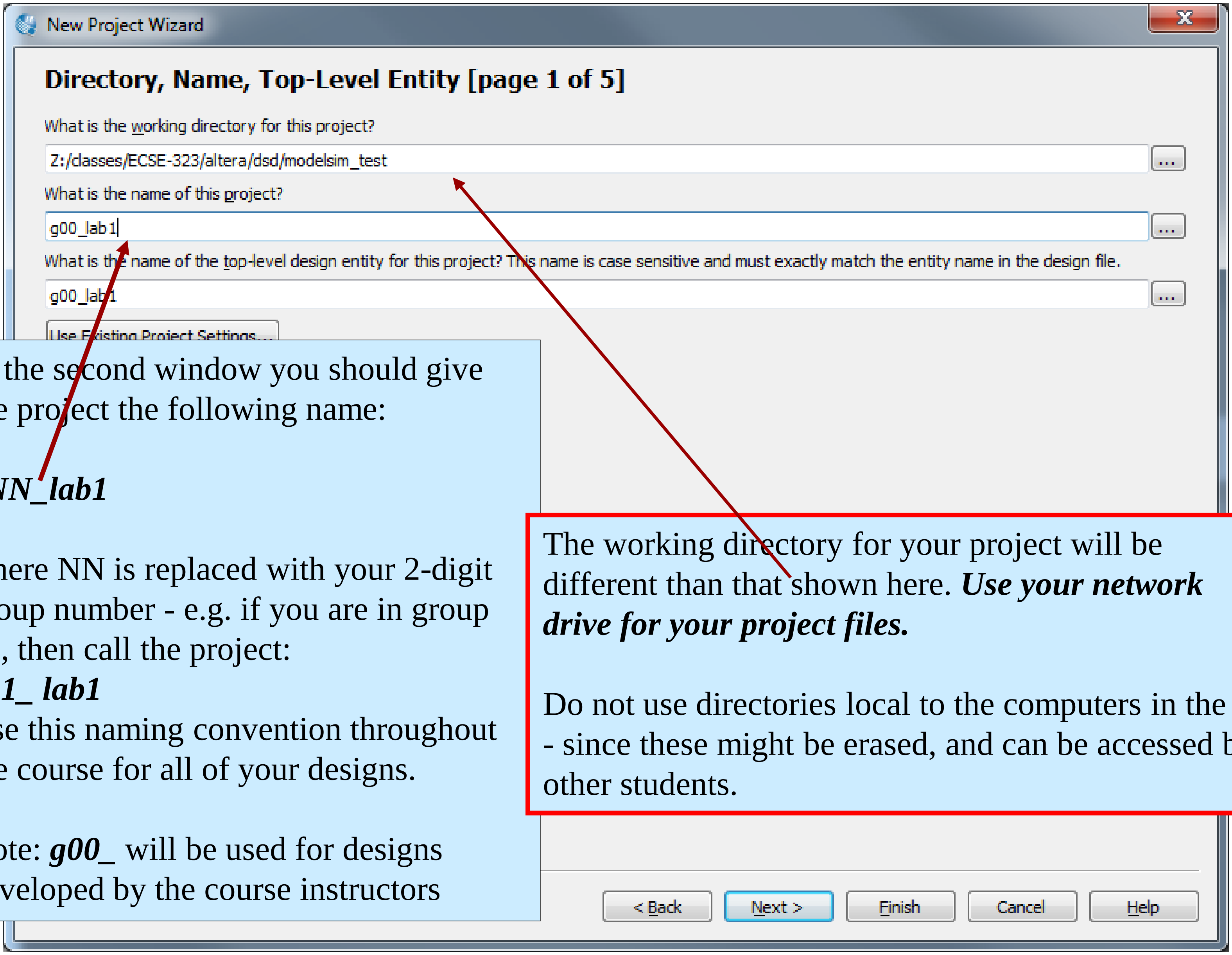
You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

Don't show me this introduction again

The New Project Wizard involves going through a series of windows.

The first window, shown at right, is an introduction, listing the settings that can be applied.

After reading the text on this window, click on "Next" to proceed.



New Project Wizard

Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

Z:/classes/ECSE-323/altera/dsd/modelsim_test

What is the name of this project?

g00_lab1

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

g00_lab1

In the second window you should give the project the following name:

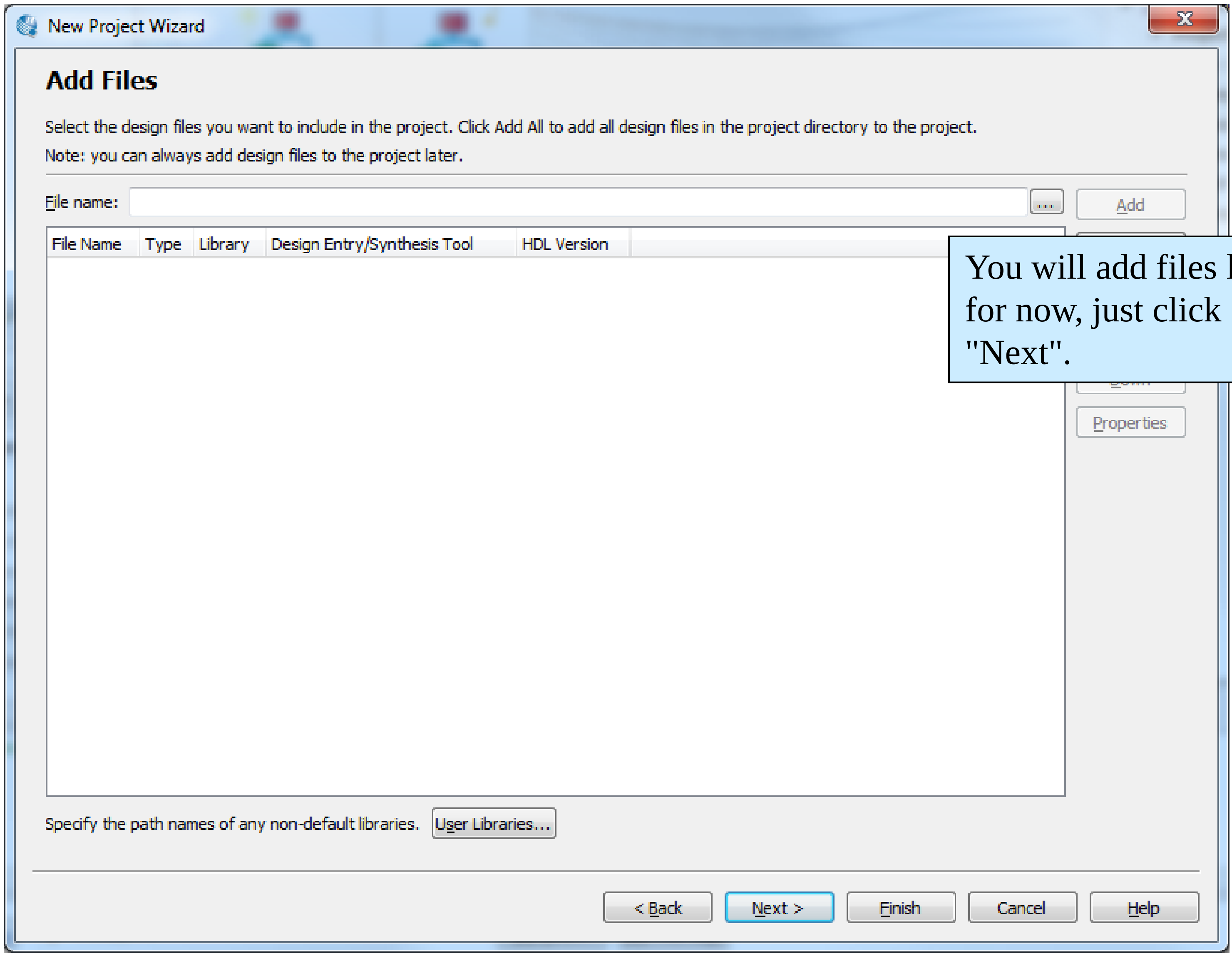
gNN_lab1

where NN is replaced with your 2-digit group number - e.g. if you are in group 21, then call the project:
g21_ lab1
Use this naming convention throughout the course for all of your designs.

Note: ***g00_*** will be used for designs developed by the course instructors

The working directory for your project will be different than that shown here. ***Use your network drive for your project files.***

Do not use directories local to the computers in the lab - since these might be erased, and can be accessed by other students.



New Project Wizard

Add Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

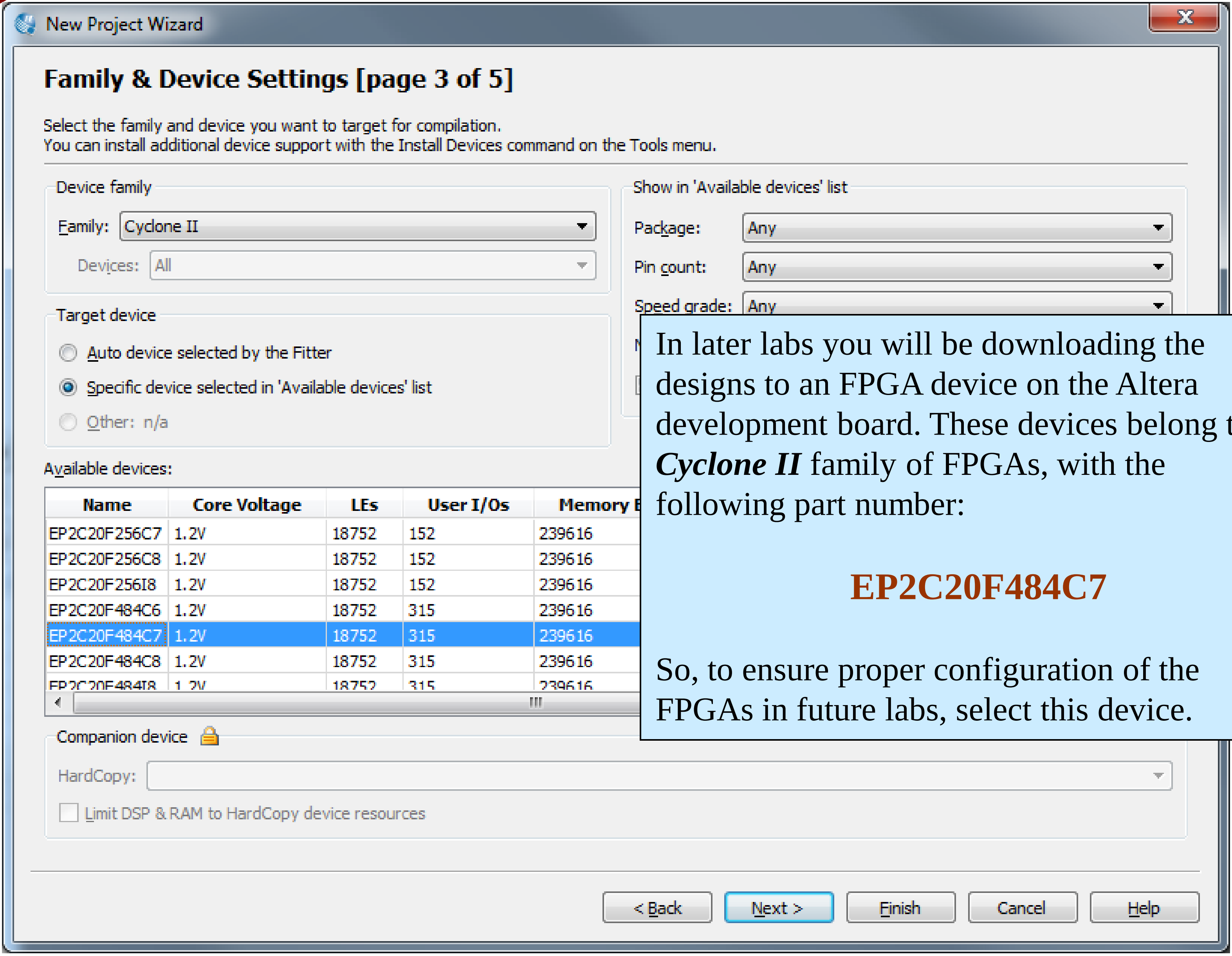
Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

Specify the path names of any non-default libraries.

You will add files later, so for now, just click on "Next".



New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family

Family: Cyclone II

Devices: All

Show in 'Available devices' list

Package: Any

Pin count: Any

Speed grade: Any

Target device

Auto device selected by the Fitter [✓ Specific device selected in 'Available devices' list] Other: n/a

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory B
EP2C20F256C7	1.2V	18752	152	239616
EP2C20F256C8	1.2V	18752	152	239616
EP2C20F256I8	1.2V	18752	152	239616
EP2C20F484C6	1.2V	18752	315	239616
EP2C20F484C7	1.2V	18752	315	239616
EP2C20F484C8	1.2V	18752	315	239616
EP2C20F484I8	1.2V	18752	315	239616

Companion device

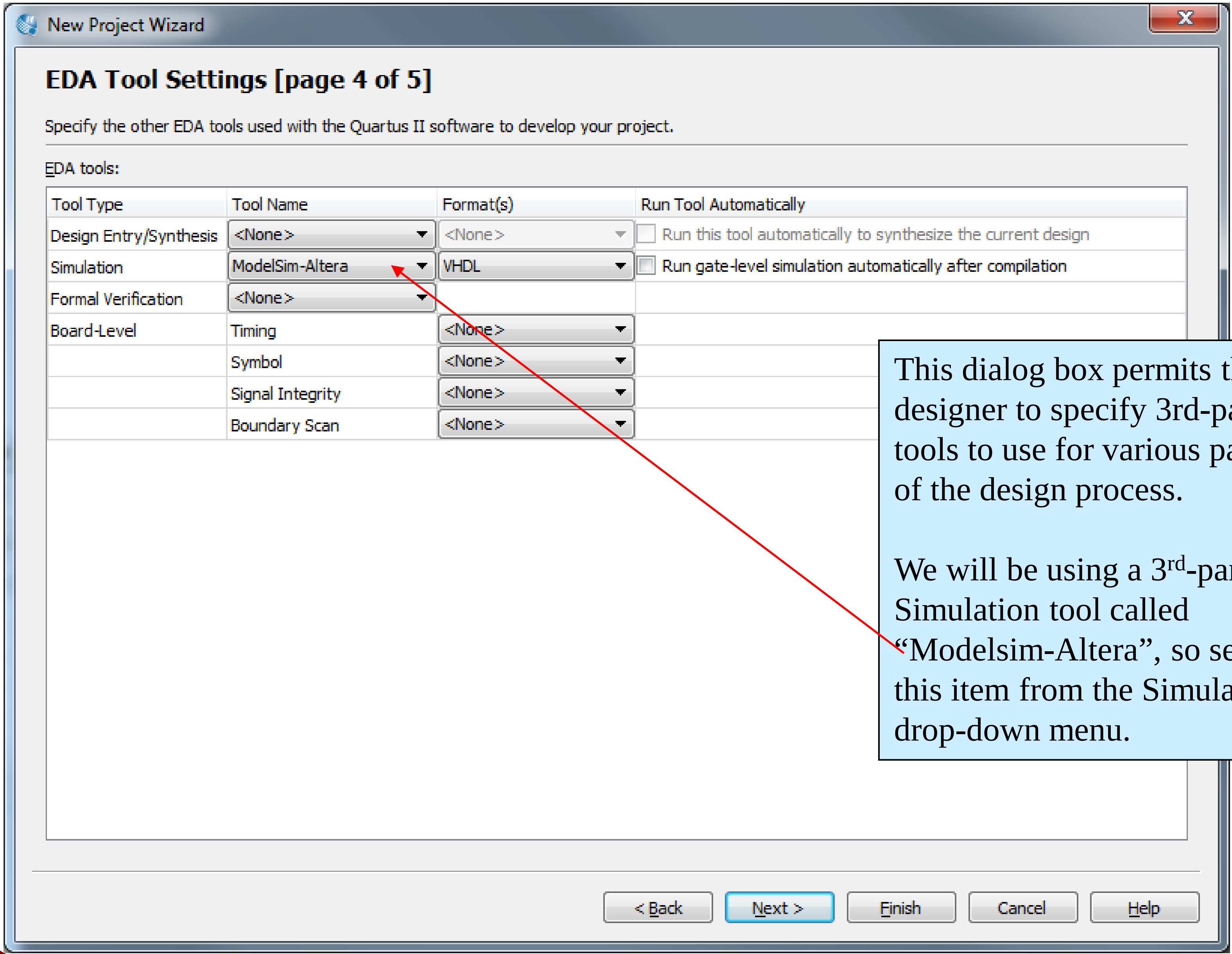
HardCopy:

Limit DSP & RAM to HardCopy device resources

In later labs you will be downloading the designs to an FPGA device on the Altera development board. These devices belong to the ***Cyclone II*** family of FPGAs, with the following part number:

EP2C20F484C7

So, to ensure proper configuration of the FPGAs in future labs, select this device.



New Project Wizard

EDA Tool Settings [page 4 of 5]

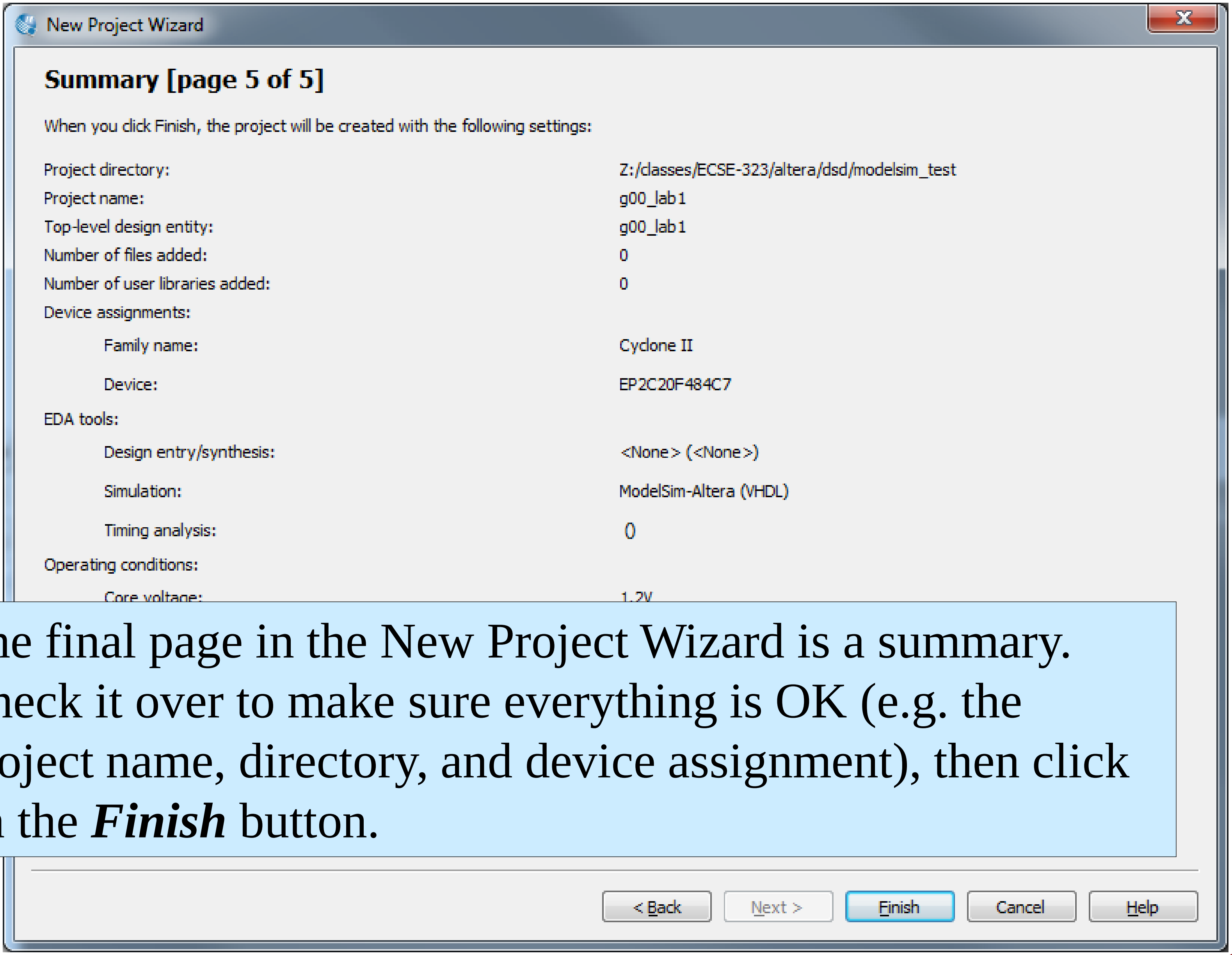
Specify the other EDA tools used with the Quartus II software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synthesis	<None>	<None>	Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	VHDL	Run gate-level simulation automatically after compilation
Formal Verification	<None>		
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

This dialog box permits the designer to specify 3rd-party tools to use for various parts of the design process.

We will be using a 3rd-party Simulation tool called “Modelsim-Altera”, so select this item from the Simulation drop-down menu.



Summary [page 5 of 5]

When you click Finish, the project will be created with the following settings:

Project directory:	Z:/classes/ECSE-323/altera/dsd/modelsim_test
Project name:	g00_lab1
Top-level design entity:	g00_lab1
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Family name:	Cyclone II
Device:	EP2C20F484C7
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	ModelSim-Altera (VHDL)
Timing analysis:	0
Operating conditions:	
Core voltage:	1.2V

The final page in the New Project Wizard is a summary. Check it over to make sure everything is OK (e.g. the project name, directory, and device assignment), then click on the ***Finish*** button.

3. Creating a schematic diagram design file

To get some practice with Quartus, you will start by designing a simple ***6-bit comparator*** circuit, which you should name ***gNN_comp6*** (where, as with the project name, NN is to be replaced with the group number).

The gNN_comp6 circuit has two 6-bit inputs, **A** and **B**, and a single 1-bit output, **AeqB**.
The output is to be high when the two inputs have exactly the same values, and be low otherwise.

The boolean equation for the output in terms of the input is easy to derive, and is:

```
AeqB = (A(5) xnor B(5)) and (A(4) xnor B(4)) and (A(3) xnor B(3))
                      and (A(2) xnor B(2))
            and (A(1) xnor B(1)) and (A(0) xnor B(0))
```

Verify for yourself that this boolean equation does in fact represent the desired function.

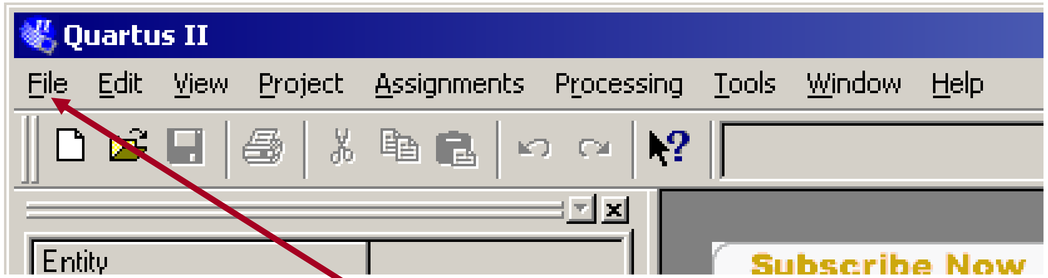
In this course, you will learn two methods for describing circuits in Quartus to be implemented on an FPGA:

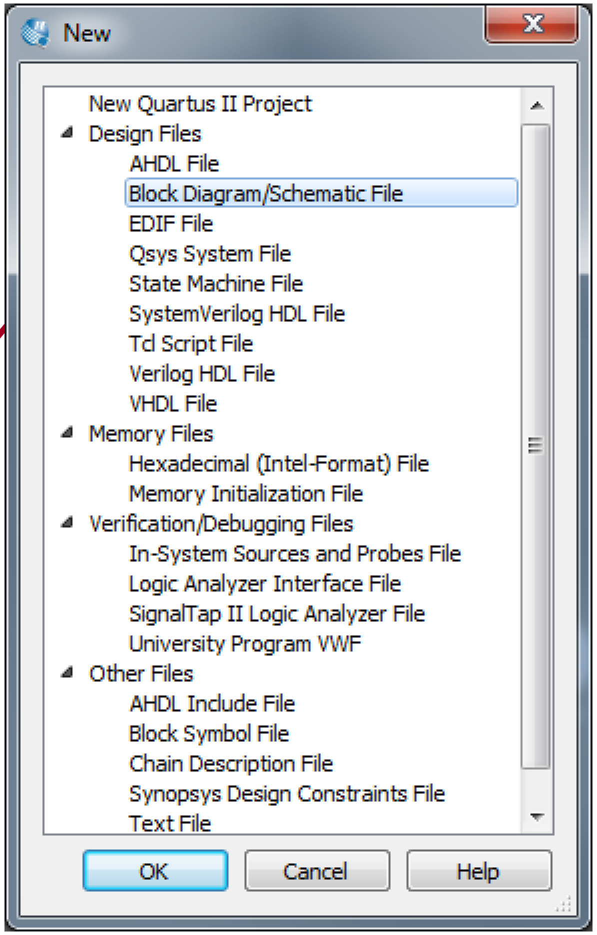
- ***gate-level schematic diagrams***
- ***VHDL descriptions***.

Schematic diagrams are graphical descriptions of the circuit, while VHDL uses textual descriptions. For most of the designs in the course you will use VHDL, but schematic entry is often useful for making the top level designs.

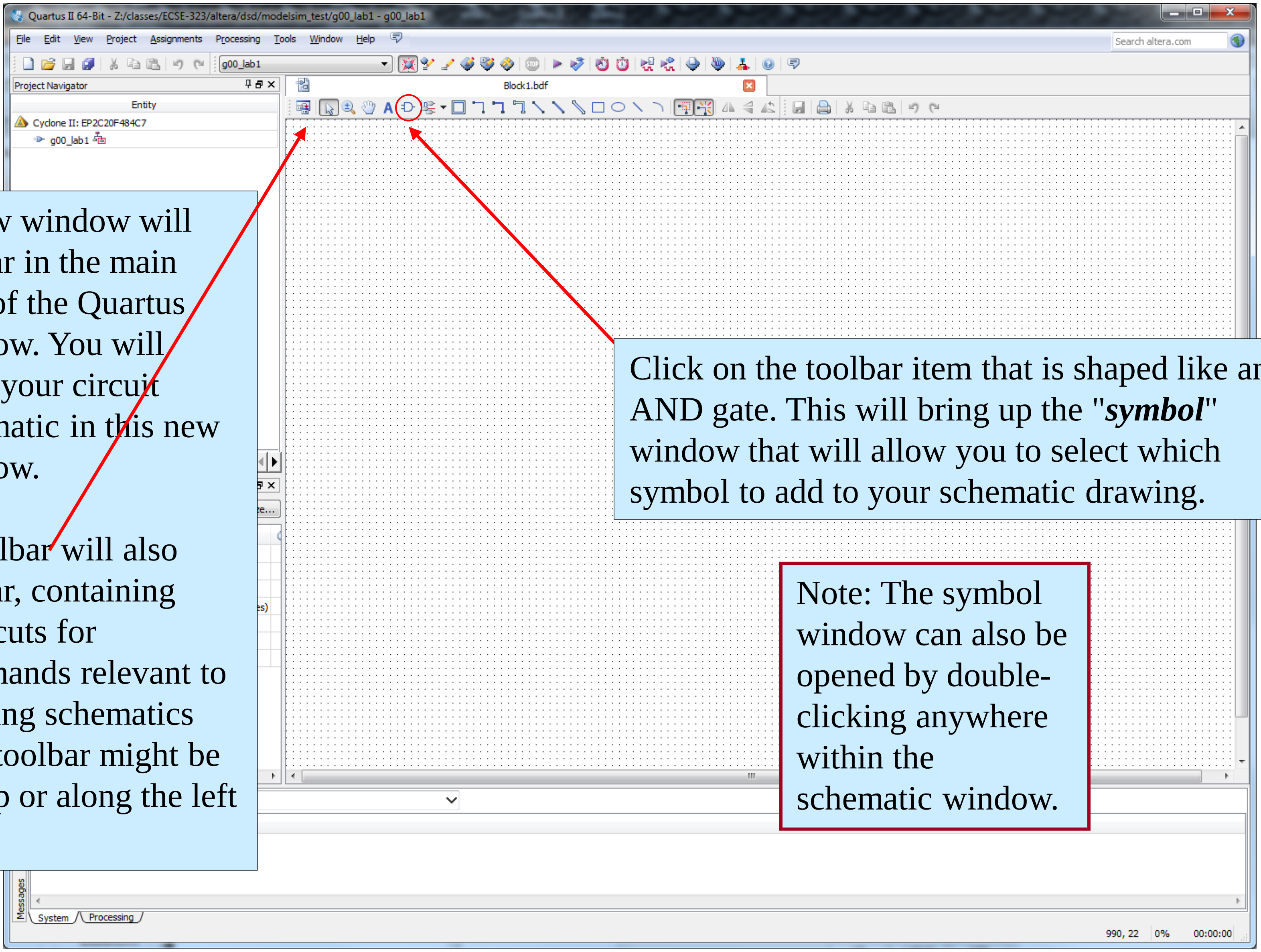
In this lab you will learn how to describe a circuit via a schematic diagram.
You will learn how to describe circuits with VHDL in the second part of the lab.

Let us now create a schematic diagram ***design file*** for the 6-bit comparator.
Start by opening a new block diagram window.


Quartus II
File Edit View Project Assignments Processing Tools Window Help
Entity
Subscribe Now



1. Click and hold on the "File" menu of the main Quartus window.
2. Select the "New" menu item. The dialog box shown at right will popup.
3. In the dialog box, select "Block Diagram/Schematic File" and click on "OK".


Quartus II 64-Bit - Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_lab1 - g00_lab1
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Cyclone II: EP2C20F484C7
g00_lab1
Block1.bdf
System
Processing
Messages
990, 22
0%
00:00:00

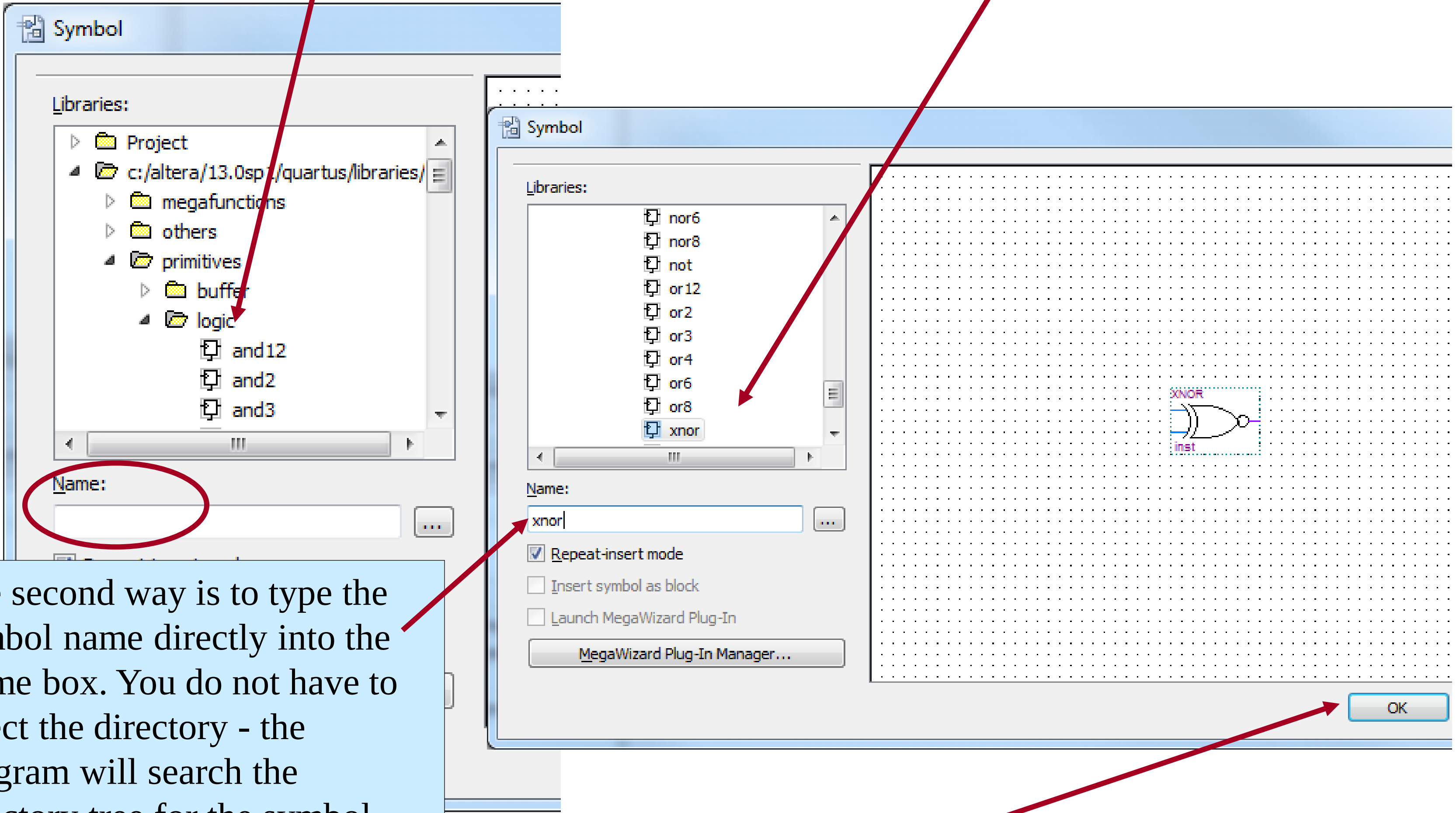
A new window will appear in the main area of the Quartus window. You will draw your circuit schematic in this new window.

A toolbar will also appear, containing shortcuts for commands relevant to drawing schematics (this toolbar might be on top or along the left side).

Click on the toolbar item that is shaped like an AND gate. This will bring up the "***symbol***" window that will allow you to select which symbol to add to your schematic drawing.

Note: The symbol window can also be opened by double-clicking anywhere within the schematic window.

You will want to add an XNOR gate symbol to your schematic. This can be done in two ways. The first way is to expand the library directory to the primitives/logic directory and then scroll down to the xnor item and select it.

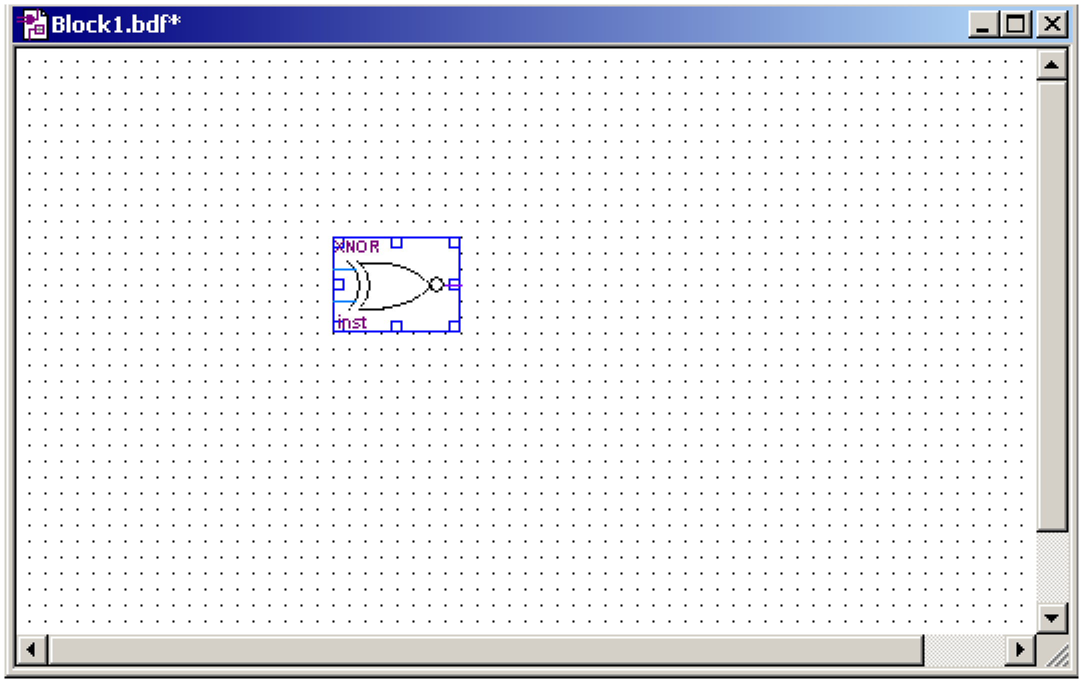


The second way is to type the symbol name directly into the Name box. You do not have to select the directory - the program will search the directory tree for the symbol.

Finally, click on OK to complete the selection.

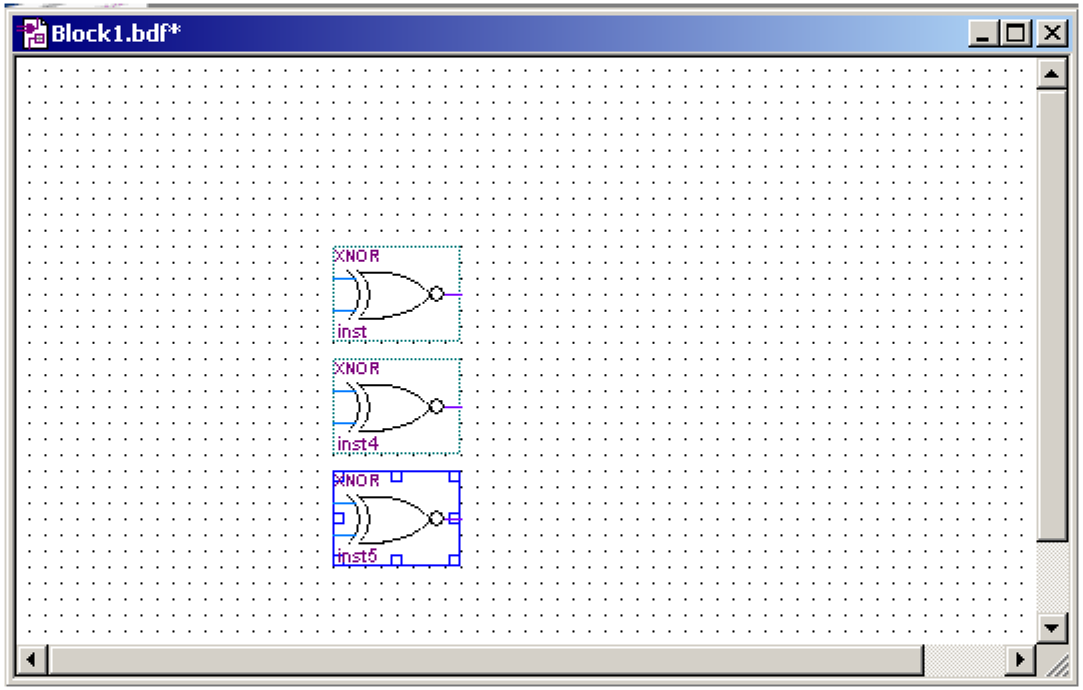
After clicking on OK in the Symbol window, a floating image of an XNOR gate will appear in the schematic window. As you move the mouse, the symbol will follow. Position the symbol where you would like it to be and left-click the mouse. The symbol will now be placed onto the schematic.

Your schematic should now look something like the figure below:



Your design for the 6-bit comparator requires six instances of the XNOR gate.
You could repeat the symbol search process five more times, but there is an easier, faster way to enter the other five XNOR gates.

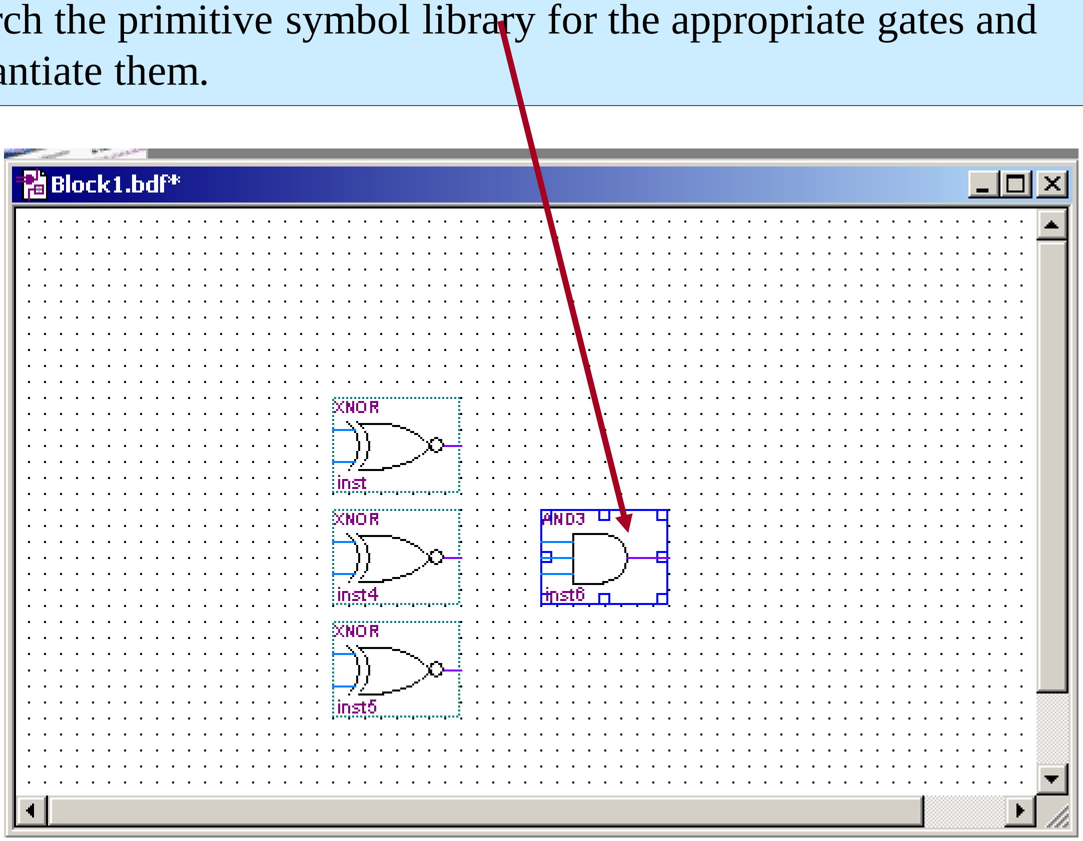
The faster way is to use ***copy-and-paste***. To ***copy*** a symbol, left-click on it, to select the symbol, and then press **Control-c** on the computer keyboard. Then left-click the mouse with the cursor positioned at the location where you want the new symbol instance to be placed. Then press **Control-v** on the computer keyboard to ***paste*** the symbol at this location.



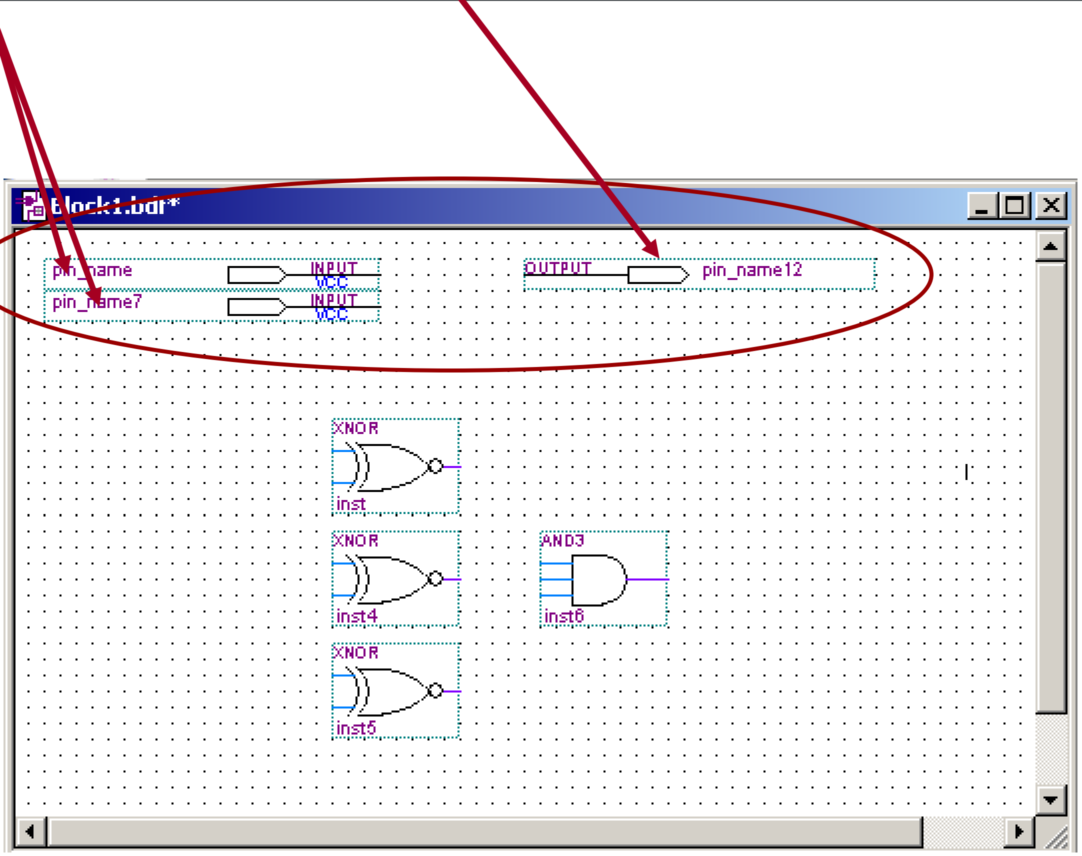
Do this copy-and-paste twice, to end up with 3 XNOR gates arranged as shown. Then select the group of three and repeat the copy-and-paste, to leave 6 XNOR gates.

You need to AND together the outputs of the 6 XNOR gates. This can be done with two 3-input AND gates and one 2-input AND gate.

Search the primitive symbol library for the appropriate gates and instantiate them.



Before connecting the gates, you should enter the input and output ports. This is done by instantiating symbols named "***input***" and "***output***". We need 2 input ports and 1 output port.



You now need to ***connect*** all of the gates together and hook them up to the input and output ports.

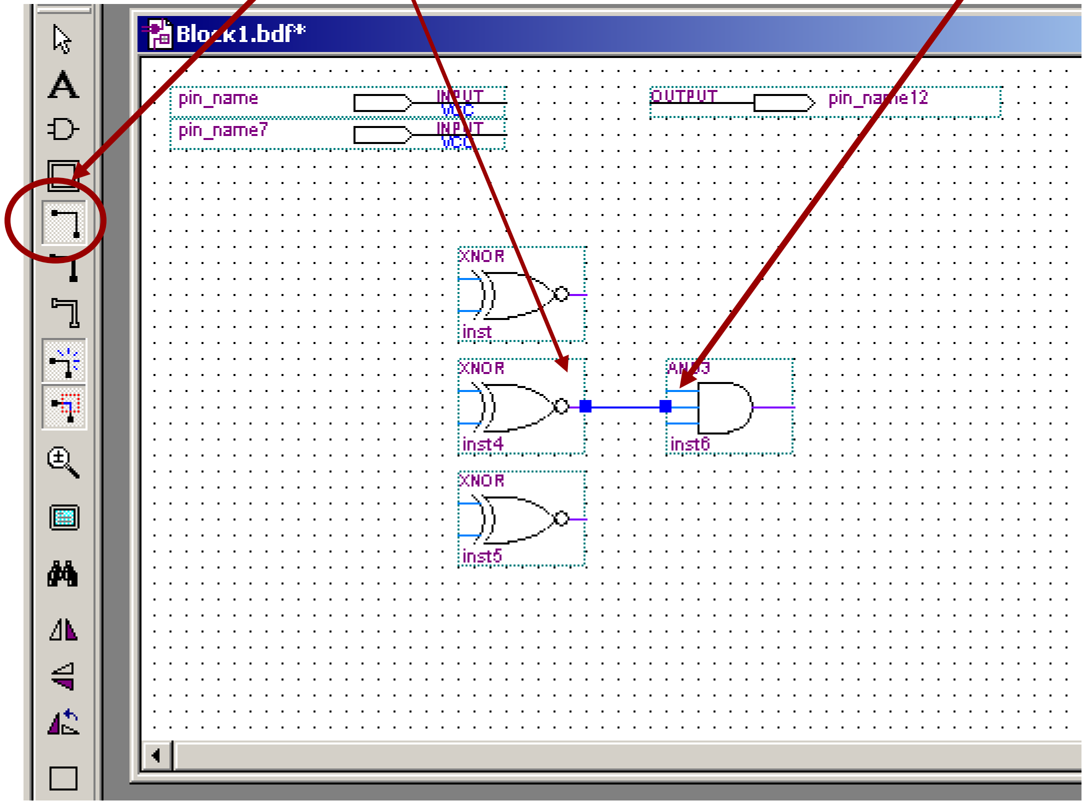
There are two ways that you can connect nodes in the schematic:

1. Drawing ***wires*** between two nodes using the ***orthogonal node tool***.
2. Giving two different nodes the same ***name*** or ***label***.

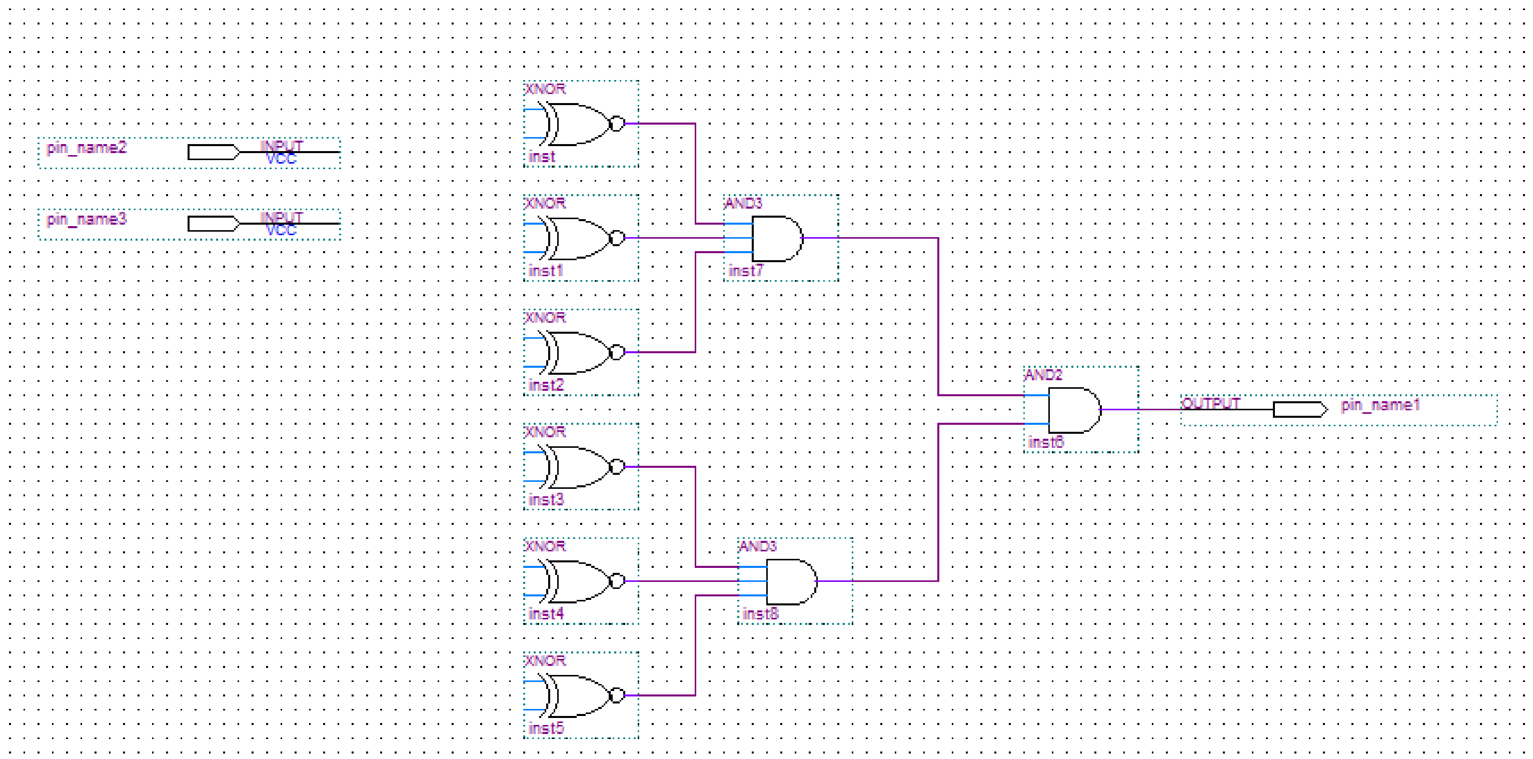
The first approach is best when making short connections between neighboring symbols, or when you want to make it immediately obvious from looking at the schematic that two nodes are connected.

The second approach is the most convenient way of connecting busses (which are collections of individual nodes or wires). The software assumes that two nodes with the same name are electrically connected.

To draw ***wires*** between two nodes using the ***orthogonal node tool*** first select the tool from the toolbar to the left of the window. This tool allows to draw wires which run horizontally or vertically. When this tool is selected, the cursor changes into a crosshair. To draw a wire, position the cursor over the place where you want the wire to begin, and left-click the mouse. Then, while holding the mouse button down, drag the cursor to the place where you want the wire to end, then release.



Practice drawing lots of wires between various places. You can delete a wire by selecting the "Selection and Smart Drawing Tool" (the arrow shaped icon), clicking on the wire to select it, and then pressing the Delete key. You can move symbols by selecting them (with the arrow tool) and dragging them to the desired location.



Drag the output pin symbol to the right of the 2-input AND gate, and wire up the AND gates to the outputs of the XNOR gates. Don't wire up the inputs of the XNOR gates, instead, we will connect these using node naming.

You can connect nodes without explicitly drawing in wires to connect them, by giving each node the same name.

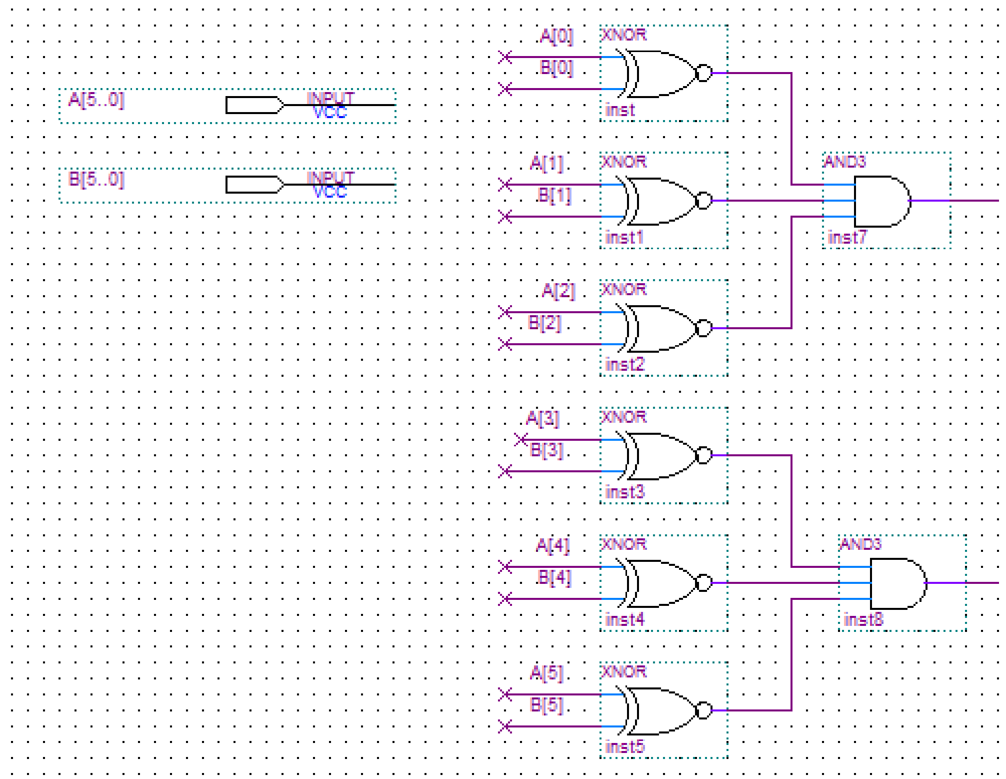
In order to use node naming as a connection technique, you must give names, or labels, to the nodes to be connected. Begin by naming the input busses. To do this, double-left-click on the left-most part of one of the input symbol. This should highlight the input name (this can be a bit tricky, so get the TA to help if you can't manage to select the name field). You can also set the name by right-clicking on the symbol and selecting the "***Properties***" menu item, and filling in the name field there.

Name the inputs as A[5..0] and B[5..0].
Also name the output as AeqB.

The name A[5..0] means that A is a bus with 6 wires, having indices of 5,4,…,0. The node named A[5] corresponds to the Most-Significant-Bit (MSB) of A, and A[0] to the Least-Significant-Bit (LSB) of A.

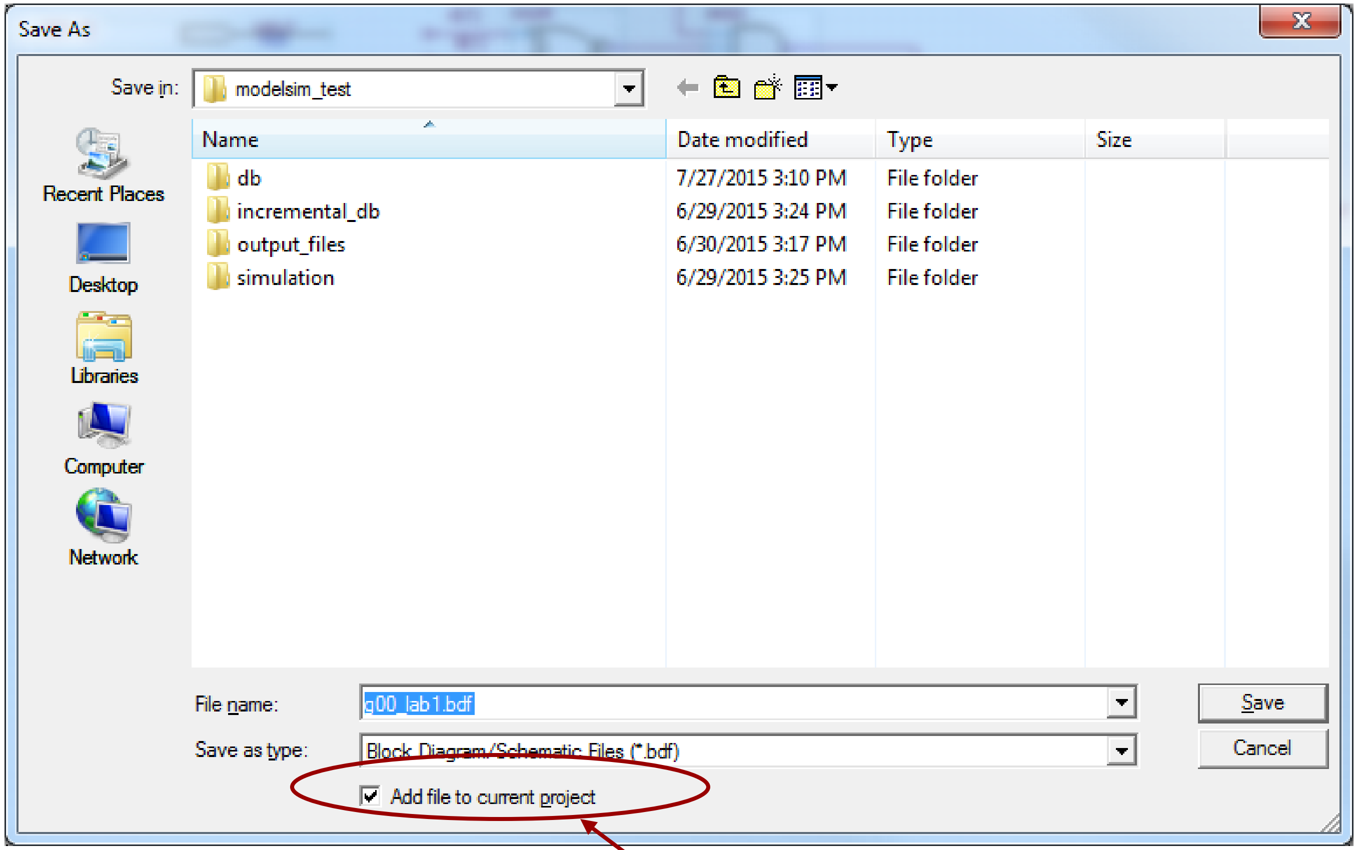
Use node naming to connect the XNOR gates to the input symbols. To do this, connect short wires to the XNOR input points, as shown in the figure to the right. Now, immediately after drawing a wire, type in the desired name. You can also select the wire later and type in the name then, or select the wire and right-click and select Properties and change the name in the Properties dialog box.

Since you have given the wire connected to the first input of the top XNOR gate the name A[5], it will automatically be connected to the MSB of the input symbol named A[5..0]



You should now save the design, using the "***Save As***" item in the ***File*** menu.
Actually, you can do this at any time, and it is a good habit to save regularly - just in case.

Call your file "***gNN_comp6.bdf*** " where ***NN*** is replaced by your group number.



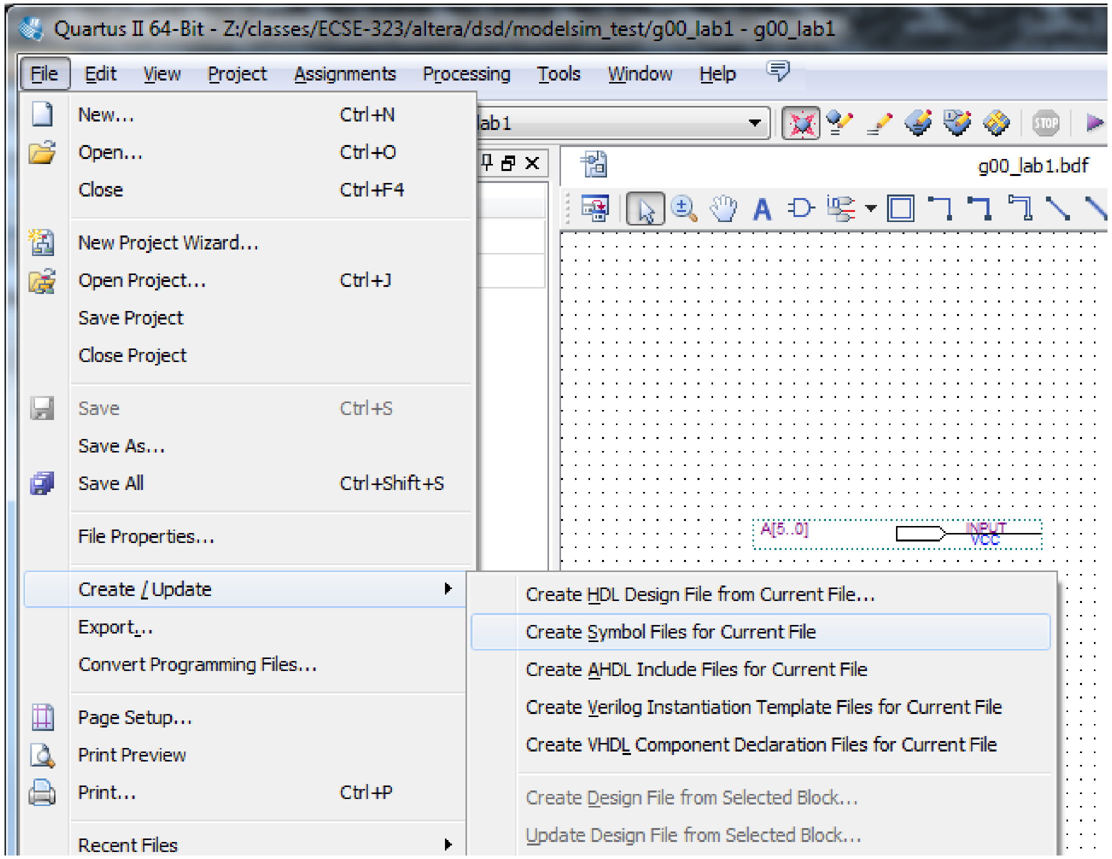
Name	Date modified	Type	Size
db	7/27/2015 3:10 PM	File folder	
incremental_db	6/29/2015 3:24 PM	File folder	
output_files	6/30/2015 3:17 PM	File folder	
simulation	6/29/2015 3:25 PM	File folder	

File name: g00_lab1.bdf
Save as type: Block Diagram/Schematic Files (*.bdf)
☑ Add file to current project

Make sure that the "Add file to current project" box is selected.

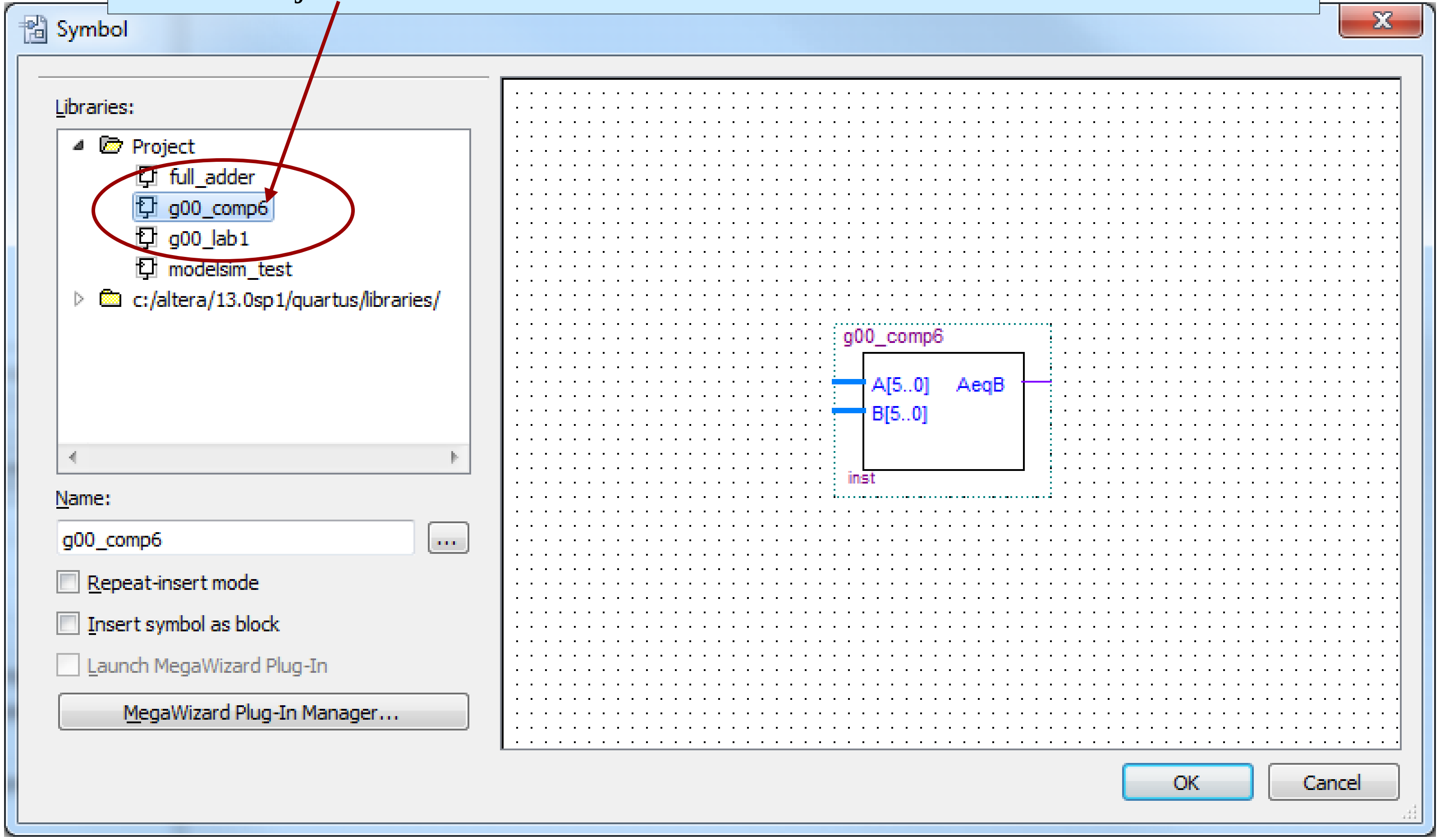
So that the circuit gNN_comp6 can be used as a component in other circuits, we need to create a ***symbol*** for it. This is done by selecting the "***Create/Update***" menu item in the "***File***" menu, and then selecting the "***Create Symbol Files for Current File***" menu item.

A window will popup asking for the filename. Just use the one it suggests, which will be gNN_comp6.bsf and click OK. Another window will popup saying that the block symbol file was created. Click on OK once more.


Quartus II 64-Bit - Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_lab1 - g00_lab1
File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Create HDL Design File from Current File...
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
Create Design File from Selected Block...
Update Design File from Selected Block...
g00_lab1.bdf
A[5..0]
INPUT
VCC

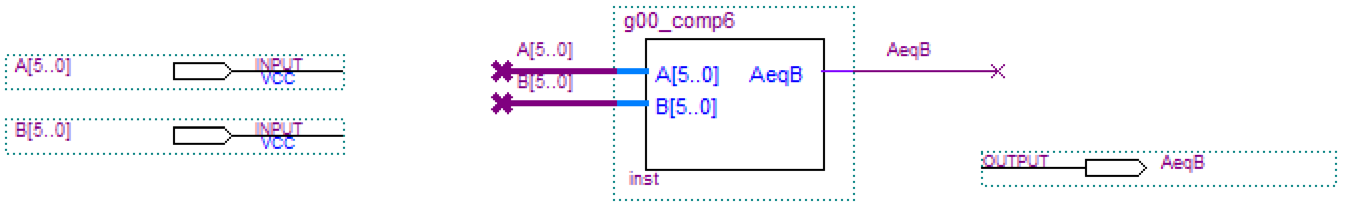
This completes the entry of the schematic for the g00_comp6 circuit. ***Show your schematic to the TA and have him or her sign your grade sheet.***

Open a new block diagram file, and insert an instance of your gNN_comp6 block. Note that this block now appears in the Libraries list, under the folder "Project".



Add some inputs and outputs and connect them using bus Naming, as shown in the figure below.

Save the file as "***gNN_lab1.bdf***" (the name of this file should be the same as the name you gave to your project when you ran the New Project Wizard at the beginning of the lab).



This is somewhat pointless, since we could have made the g00_comp6 the top level entity. But, typically, the top level entity will contain many different modules connected together to implement the overall system. So this example serves to show how to insert modules you designed into a schematic.

Note that in capturing this schematic screenshot, I turned off the background dots. You should do the same whenever capturing a screenshot for your reports.
To do this, select the View toolbar item and uncheck the “Show Guidelines” entry.

TIME CHECK

You should be at least this far at the end of your *first* 2-hour lab period.

4. Simulation of the Project using ModelSim

Once you have your circuit described in a schematic diagram (or, in later labs, in a VHDL description) you should simulate it.

The purpose of simulation is generally two-fold:

1. To determine if the circuit performs the desired function
2. To determine if timing constraints are met

In the first case, we are only interested in the functionality of our implementation. We do not care about propagation delays and other timing issues. Because of this, we do not have to map our design to a target hardware. This type of simulation is called ***functional simulation.*** *This is the type of simulation we will learn about in this lab.*

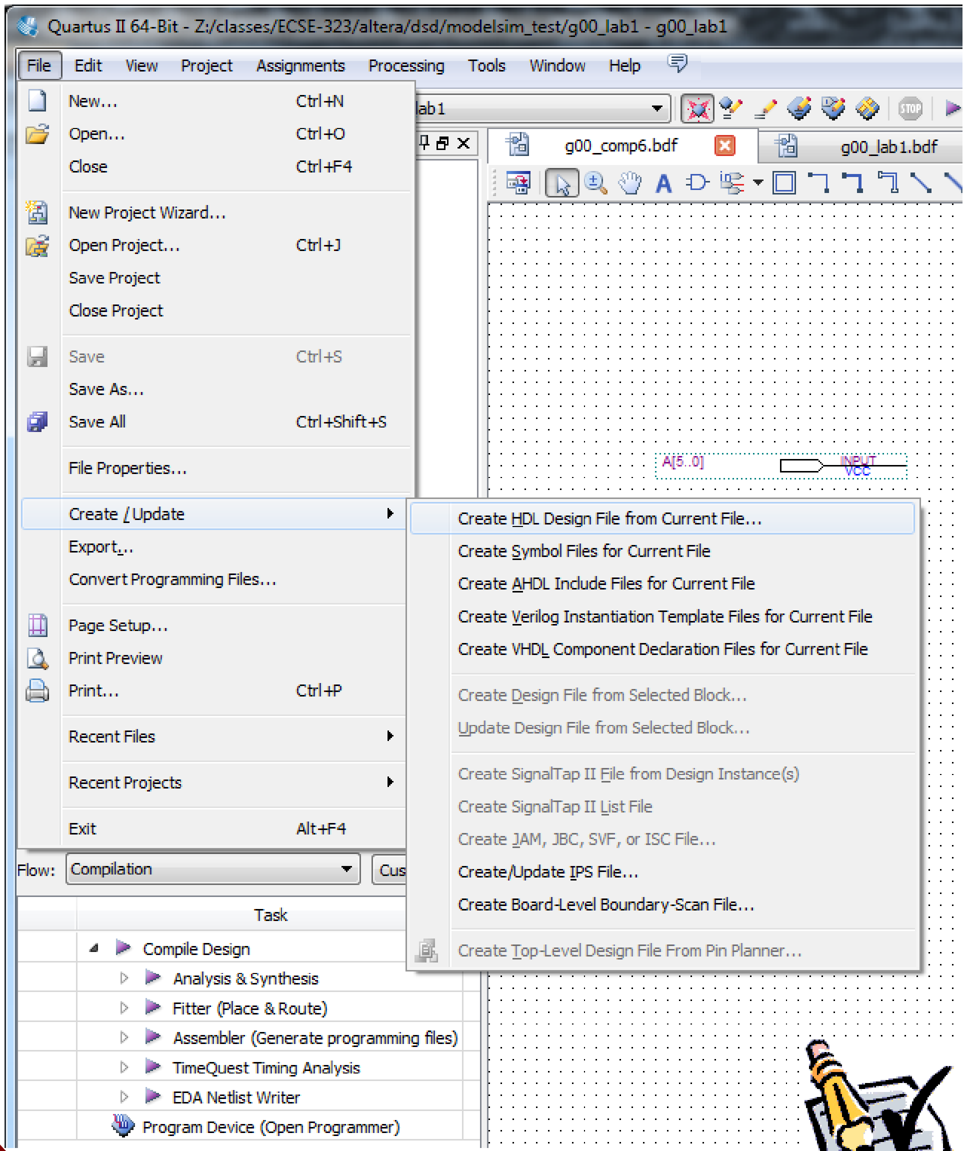
The other form of simulation is called timing simulation. It requires that the design be mapped onto a target device, such as an FPGA. Based on the model of the device, the simulator can predict propagation delays, and provide a simulation that takes these into account. Thus, the timing simulation may produce results that are quite different from the purely functional simulation.

In this course, we will be using the ***Modelsim*** simulation software, created by the company Mentor Graphics (actually we will use a version of it specific to Quartus, called Modelsim-Altera).

The Modelsim software operates on an HDL description of the circuit to be simulated, written either in VHDL, Verilog, or System-Verilog. **You will use VHDL**.

Modelsim doesn't understand schematic diagrams, so you need to convert your schematic diagram to an HDL description. You could just write the VHDL description directly, but you already have a schematic so it would be nice if you could use that. Fortunately, Quartus has a built-in capability of converting schematic diagrams to a VHDL description, so you can use that to convert your comparator schematic to VHDL.

In future labs, you will just write everything in VHDL right from the start, so there will be no need for schematic conversions.


Quartus II 64-Bit - Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_lab1 - g00_lab1
File Edit View Project Assignments Processing Tools Window Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Recent Projects
Exit Alt+F4
Create HDL Design File from Current File...
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
Create Design File from Selected Block...
Update Design File from Selected Block...
Create SignalTap II File from Design Instance(s)
Create SignalTap II List File
Create JAM, JBC, SVF, or ISC File...
Create/Update IPS File...
Create Board-Level Boundary-Scan File...
Create Top-Level Design File From Pin Planner...
g00_comp6.bdf
g00_lab1.bdf
A[5..0]
INPUT VCC
Flow: Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Program Device (Open Programmer)

Go back and display the schematic capture window for your comparator circuit.

To convert this schematic to a VHDL description that the simulator can use, select FILE/Create/Update/Create HDL Design File from Current File.

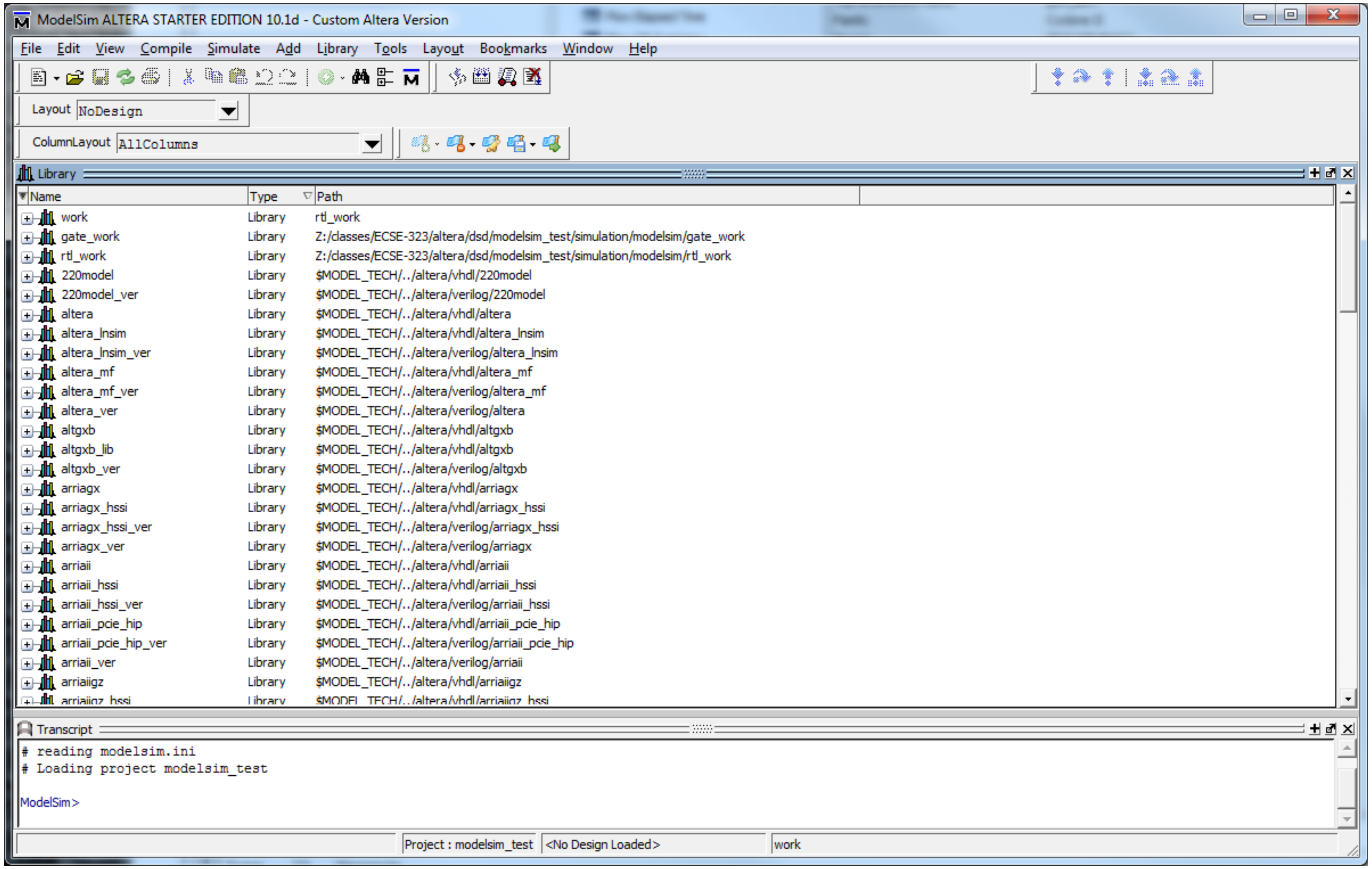
This will do the conversion, and create a file named gNN_comp6.vhd

Open it in the Quartus editor and take a look at how it describes the circuit.

Repeat the conversion process for the top level schematic gNN_lab1 to generate the vhdl file gNN_lab1.vhd

Take a look at this vhdl file. You will see that it is mainly a component instantiation.
Show the VHDL file to the TA.

Double-click on the ModelSim desktop icon to startup the ModelSim program. A window similar to the one shown below will appear.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript

Name	Type	Path
work	Library	rtl_work
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/gate_work
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/rtl_work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220model
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_lnsim
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_lnsim
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_mf
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog/altgxb
arriagx	Library	$MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/arriagx_hssi
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriagx_hssi
arriagx_ver	Library	$MODEL_TECH/../altera/verilog/arriagx
arriaii	Library	$MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_hssi
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pcie_hip
arriaii_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_pcie_hip
arriaii_ver	Library	$MODEL_TECH/../altera/verilog/arriaii
arriaiigz	Library	$MODEL_TECH/../altera/vhdl/arriaiigz
arriaiigz_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaiigz_hssi

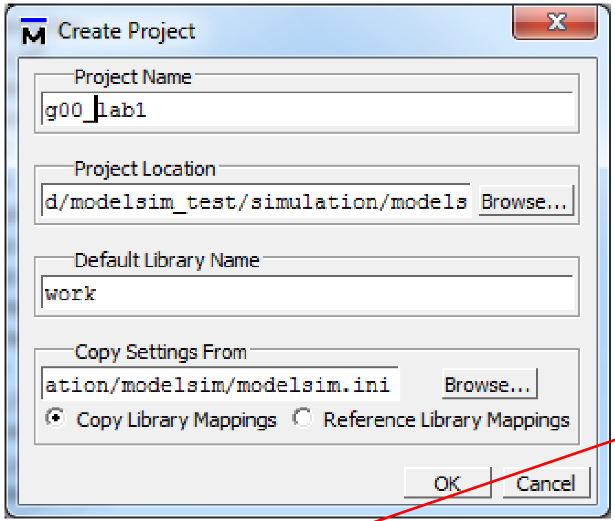
```
# reading modelsim.ini
# Loading project modelsim_test

ModelSim>
```

Project : modelsim_test   <No Design Loaded>   work

Select FILE/New/Project and, in the window that pops up, give the project the name “gNN_lab1”

Click OK.


Create Project

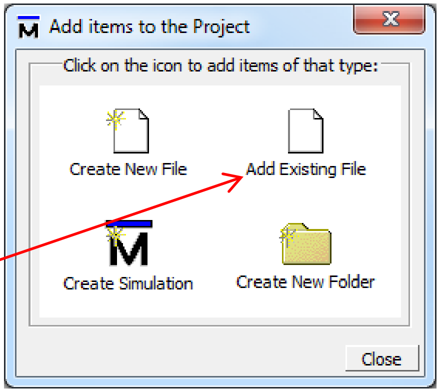
Project Name: g00_lab1

Project Location: d/modelsim_test/simulation/models

Default Library Name: work

Copy Settings From: ation/modelsim/modelsim.ini

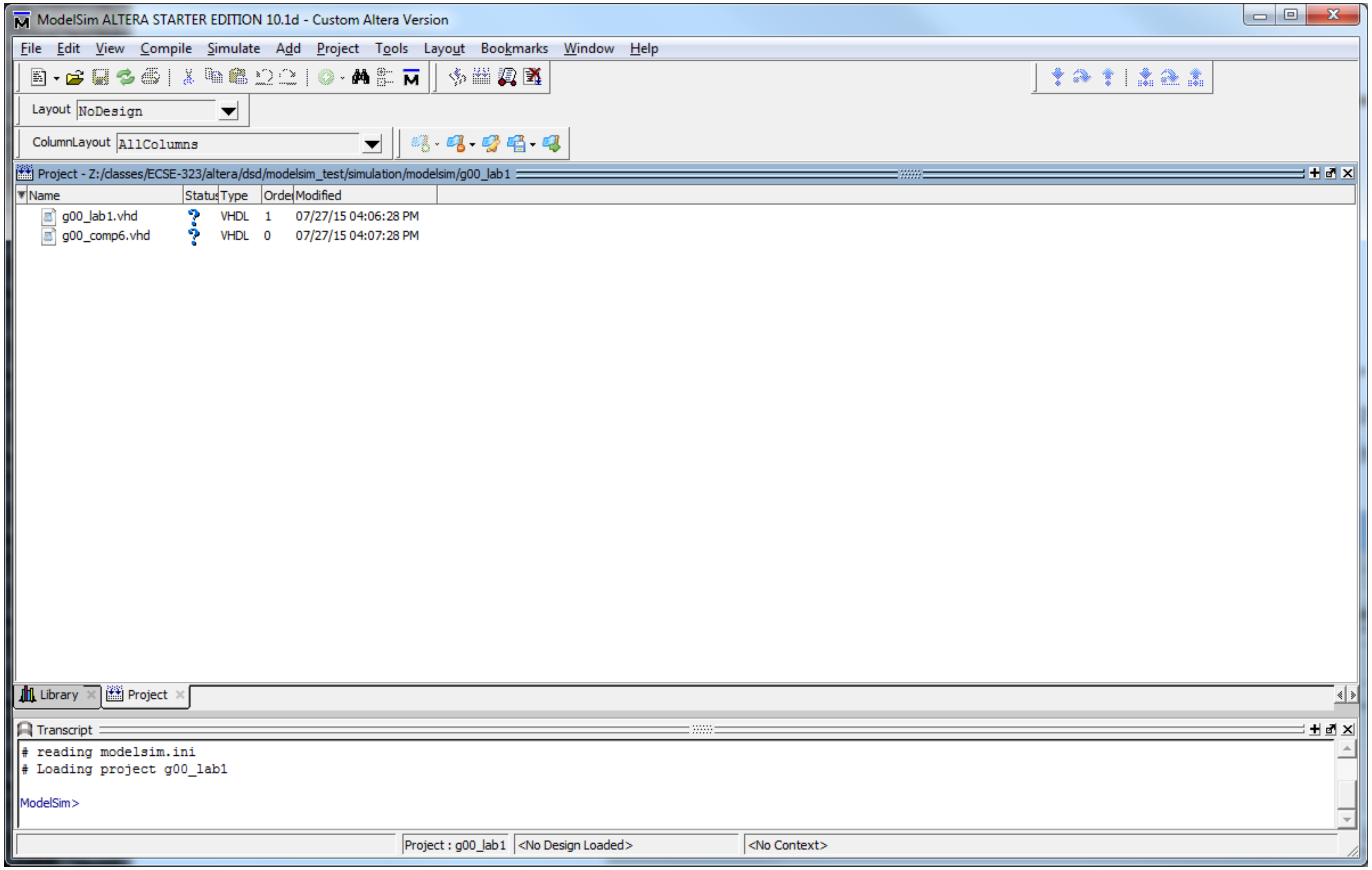
[✓ Copy Library Mappings] Reference Library Mappings


Add items to the Project

Click on the icon to add items of that type:

Another dialog box will appear, allowing you to add files to the project. Click on “Add Existing File” and select the two VHDL files that were generated earlier (gNN_comp6.vhd and gNN_lab1.vhd). You can also add files later.

The ModelSim window will now show these vhdl files in the Project pane.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Project - Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/g00_lab1
Library
Project
Transcript
Project : g00_lab1
<No Design Loaded>
<No Context>

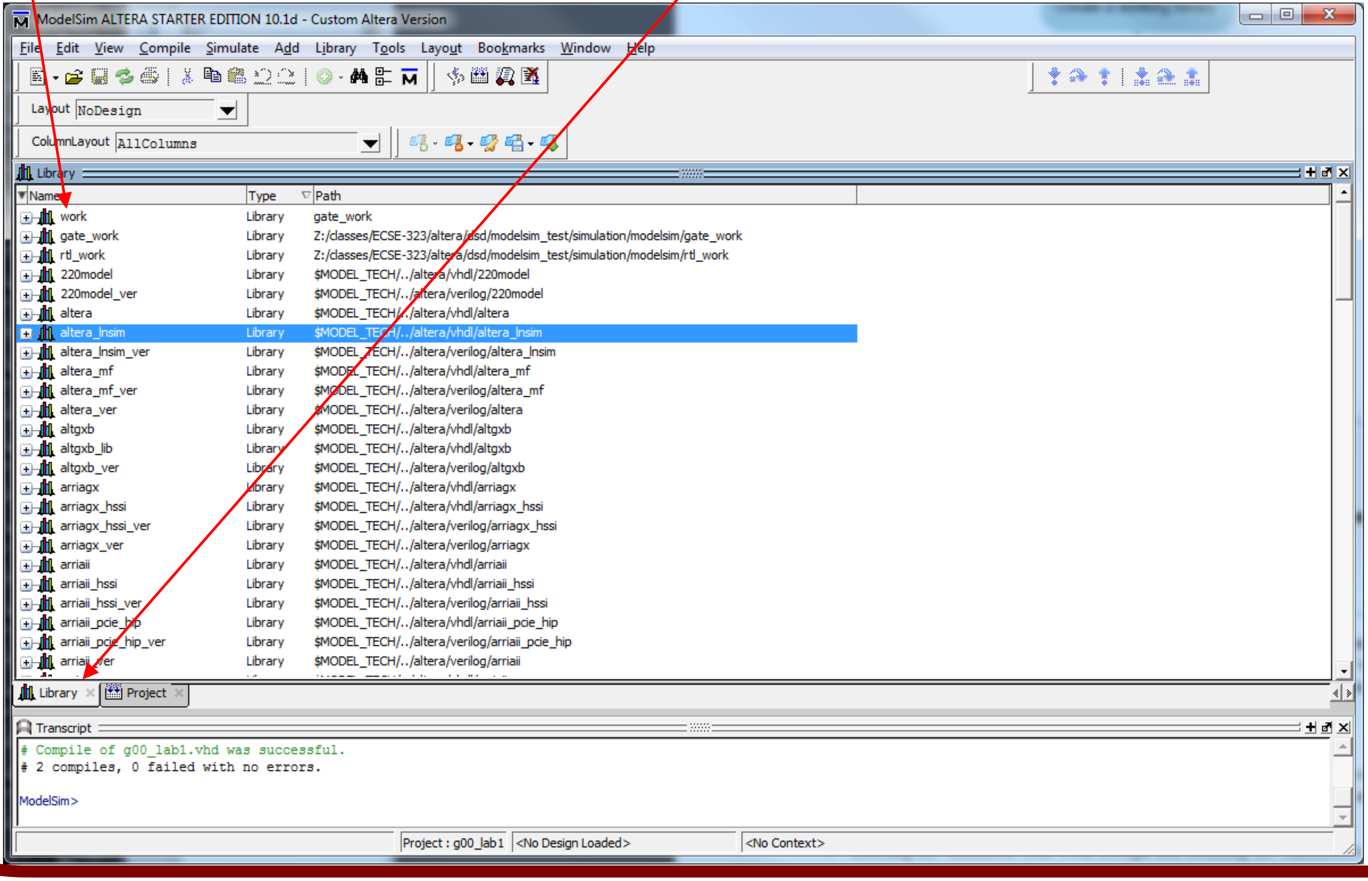
Name	Status	Type	Order	Modified
g00_lab1.vhd	?	VHDL	1	07/27/15 04:06:28 PM
g00_comp6.vhd	?	VHDL	0	07/27/15 04:07:28 PM

```
# reading modelsim.ini
# Loading project g00_lab1

ModelSim>
```

In order to simulate the design, ModelSim must analyze the VHDL files, a process known as *compilation*.

The compiled files are stored in a *library*. By default, this is named “work”. You can see this library in the “library” pane of the ModelSim window.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Library Project
Transcript

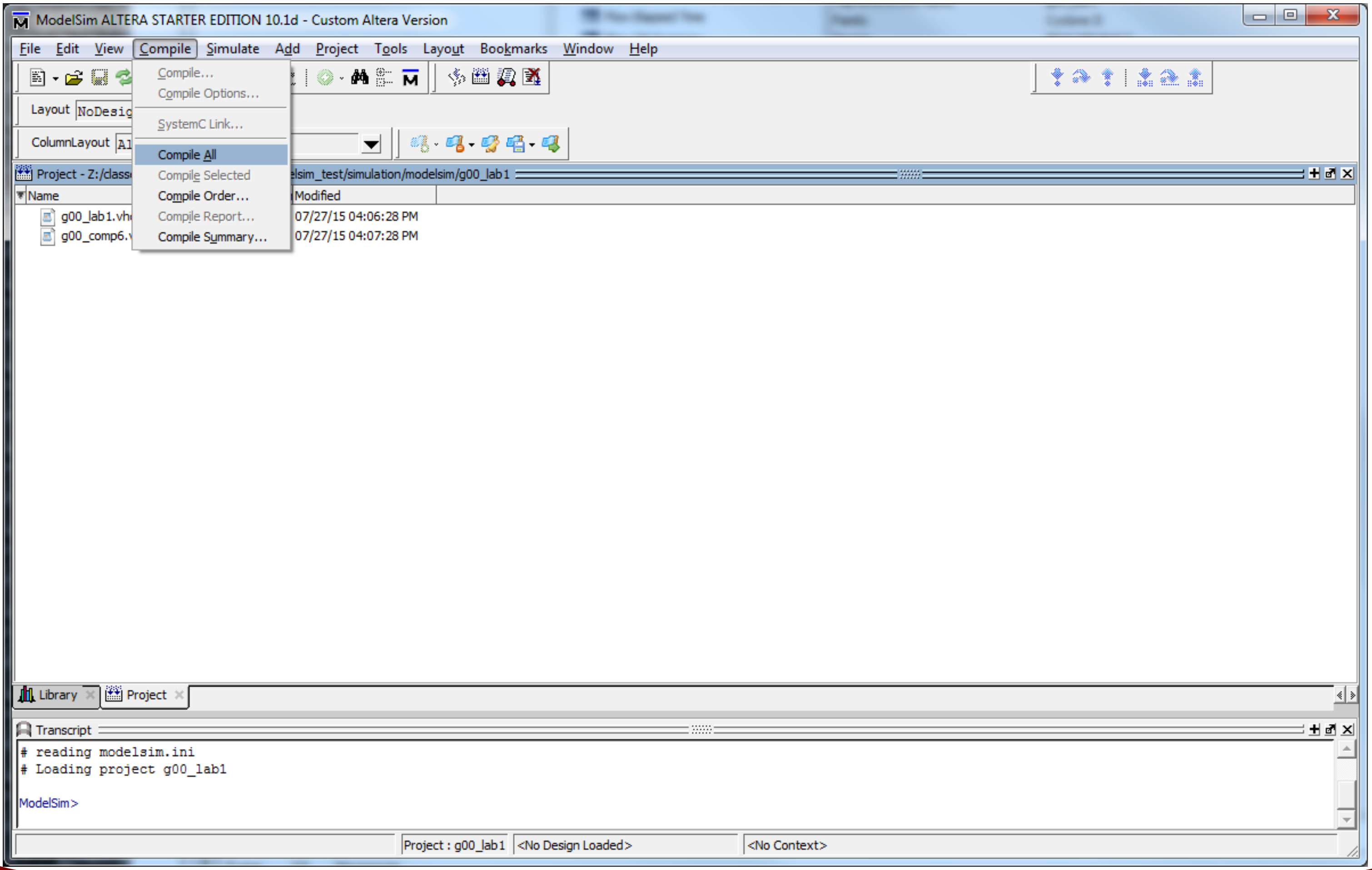
Name	Type	Path
work	Library	gate_work
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/gate_work
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/rtl_work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220model
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_lnsim
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_lnsim
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_mf
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog/altgxb
arriagx	Library	$MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/arriagx_hssi
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriagx_hssi
arriagx_ver	Library	$MODEL_TECH/../altera/verilog/arriagx
arriaii	Library	$MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_hssi
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pcie_hip
arriaii_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_pcie_hip
arriaii_ver	Library	$MODEL_TECH/../altera/verilog/arriaii

```
# Compile of g00_lab1.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Project : g00_lab1 <No Design Loaded> <No Context>

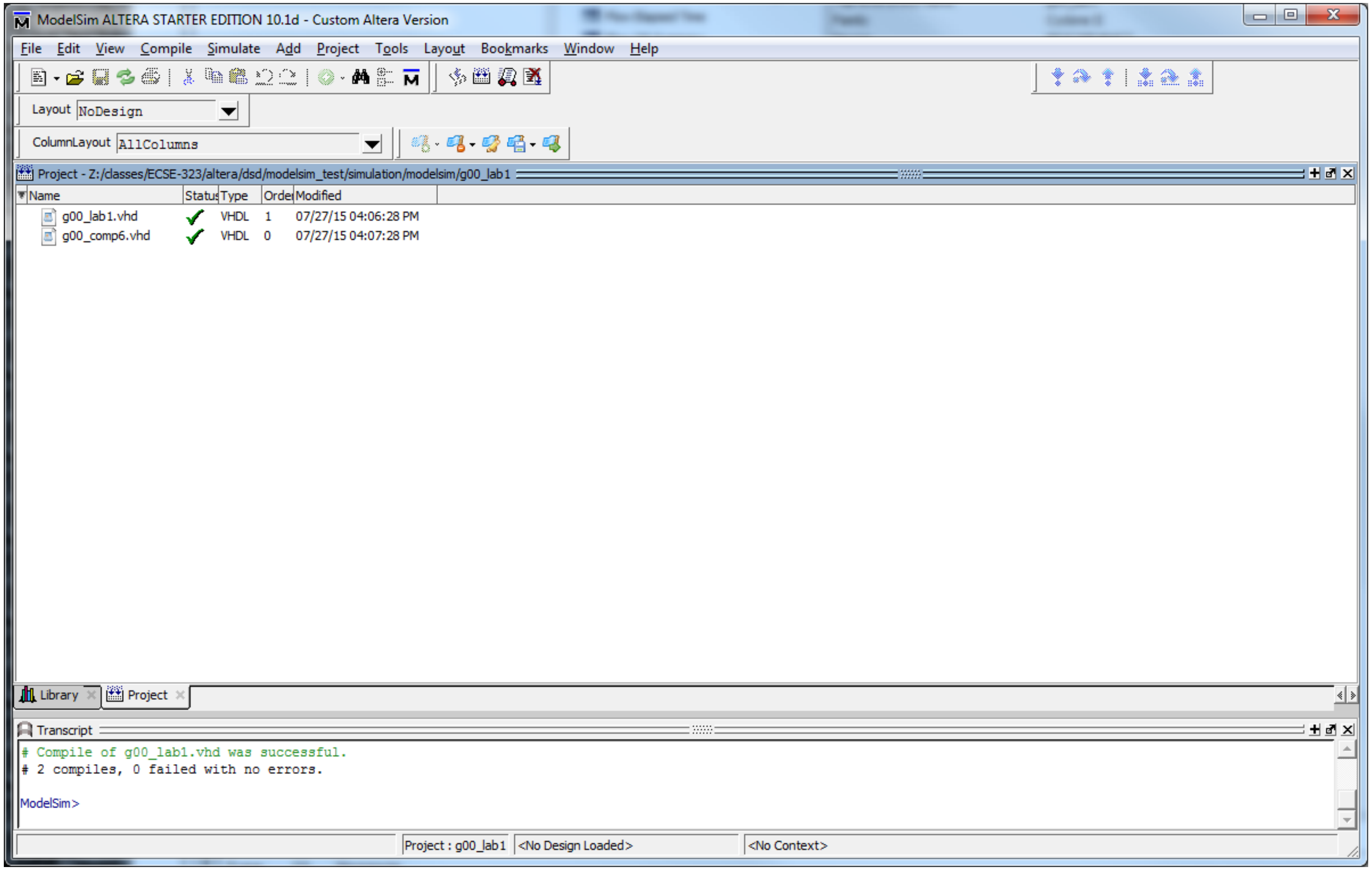
The question marks in the Status column in the Project tab indicate that either the files haven't been compiled into the project or the source file has changed since the last compilation. To compile the files, select **Compile > Compile All** or right click in the Project window and select **Compile > Compile All**.



```
# reading modelsim.ini
# Loading project g00_lab1

ModelSim>
```

If the compilation is successful, the question marks in the Status column will turn to check marks, and a success message will appear in the Transcript pane.


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Project - Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/g00_lab1
Library
Project
Transcript

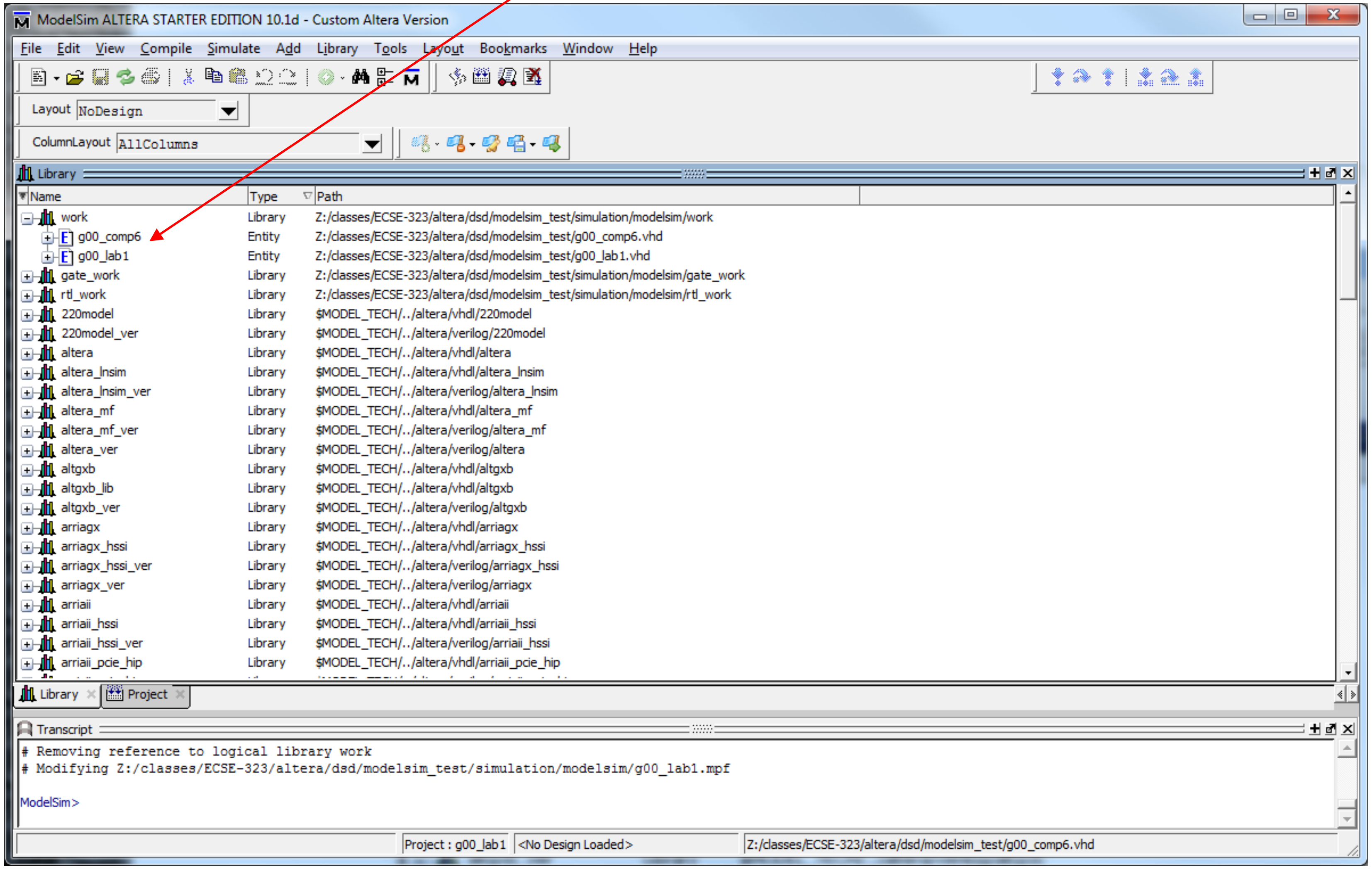
Name	Status	Type	Order	Modified
g00_lab1.vhd	✓	VHDL	1	07/27/15 04:06:28 PM
g00_comp6.vhd	✓	VHDL	0	07/27/15 04:07:28 PM

```
# Compile of g00_lab1.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Project : g00_lab1 <No Design Loaded> <No Context>

The compiled vhdl files will now appear in the library "work".


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Library
Project
Transcript

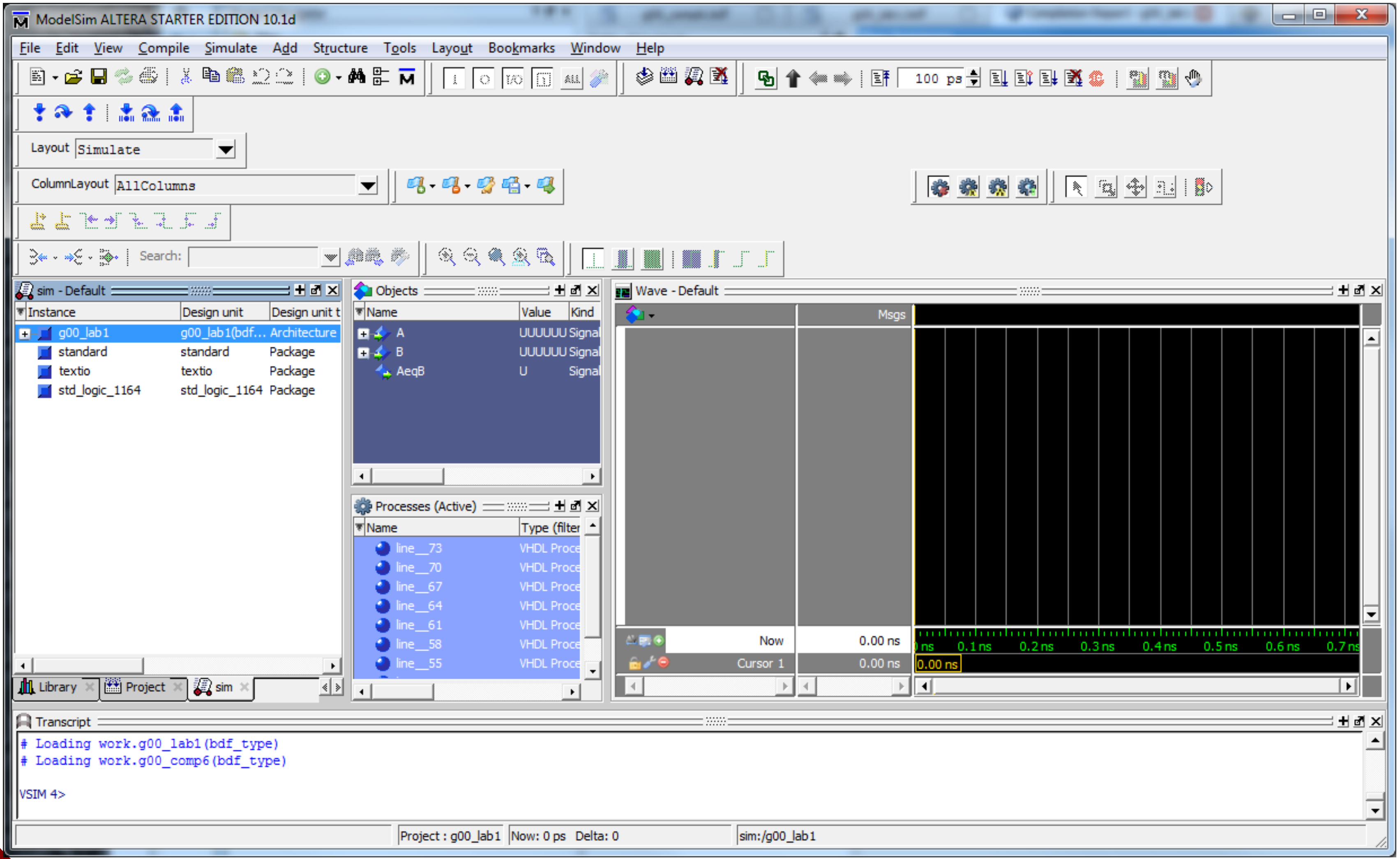
Name	Type	Path
work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/work
g00_comp6	Entity	Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_comp6.vhd
g00_lab1	Entity	Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_lab1.vhd
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/gate_work
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/rtl_work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220model
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_lnsim
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_lnsim
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_mf
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog/altgxb
arriagx	Library	$MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/arriagx_hssi
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriagx_hssi
arriagx_ver	Library	$MODEL_TECH/../altera/verilog/arriagx
arriaii	Library	$MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_hssi
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pcie_hip

```
# Removing reference to logical library work
# Modifying Z:/classes/ECSE-323/altera/dsd/modelsim_test/simulation/modelsim/g00_lab1.mpf

ModelSim>
```

Project : g00_lab1 <No Design Loaded> Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_comp6.vhd

In the library window, double-click on gNN_lab1. This will open up a bunch of windows which will be used in doing the simulation of the gNN_lab1 module.


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Structure Tools Layout Bookmarks Window Help
Layout Simulate
ColumnLayout AllColumns
sim - Default
Objects
Wave - Default
Processes (Active)
Transcript
Library
Project
sim

Instance	Design unit	Design unit t
g00_lab1	g00_lab1(bdf…	Architecture
standard	standard	Package
textio	textio	Package
std_logic_1164	std_logic_1164	Package

Name	Value	Kind
A	UUUUUU	Signal
B	UUUUUU	Signal
AeqB	U	Signal

Name	Type (filter
line__73	VHDL Proce
line__70	VHDL Proce
line__67	VHDL Proce
line__64	VHDL Proce
line__61	VHDL Proce
line__58	VHDL Proce
line__55	VHDL Proce

```
# Loading work.g00_lab1(bdf_type)
# Loading work.g00_comp6(bdf_type)

VSIM 4>
```

Project : g00_lab1 Now: 0 ps Delta: 0 sim:/g00_lab1

You are not quite ready to start the simulation yet!

Notice that, in the “Objects” window, the signals A and B (which are the inputs to the gNN_comp6 module) have the value “UUUUUU”. This means that all of the inputs are *undefined*. If you ran the simulation now, the outputs would also be undefined.

So you need to have a means of setting the inputs to certain patterns, and of observing the outputs' responses to these inputs.

In Modelsim, this is done by using a special VHDL entity called a ***Testbench***.

The testbench entity is unique in that it has NO inputs or outputs!

The testbench contains a single *component instantiation statement* that inserts the module to be tested (in this case the gNN_lab1 module), as well as some statements that describe how the test inputs are generated.

There are many ways to get timing information into a VHDL description.

A commonly used approach is to use a "**WAIT**" statement in a process block:

```
always : PROCESS
BEGIN
        A <= '0';
        WAIT FOR 2 ns;
        A <= '1';
        WAIT FOR 2 ns;
END PROCESS always;
```

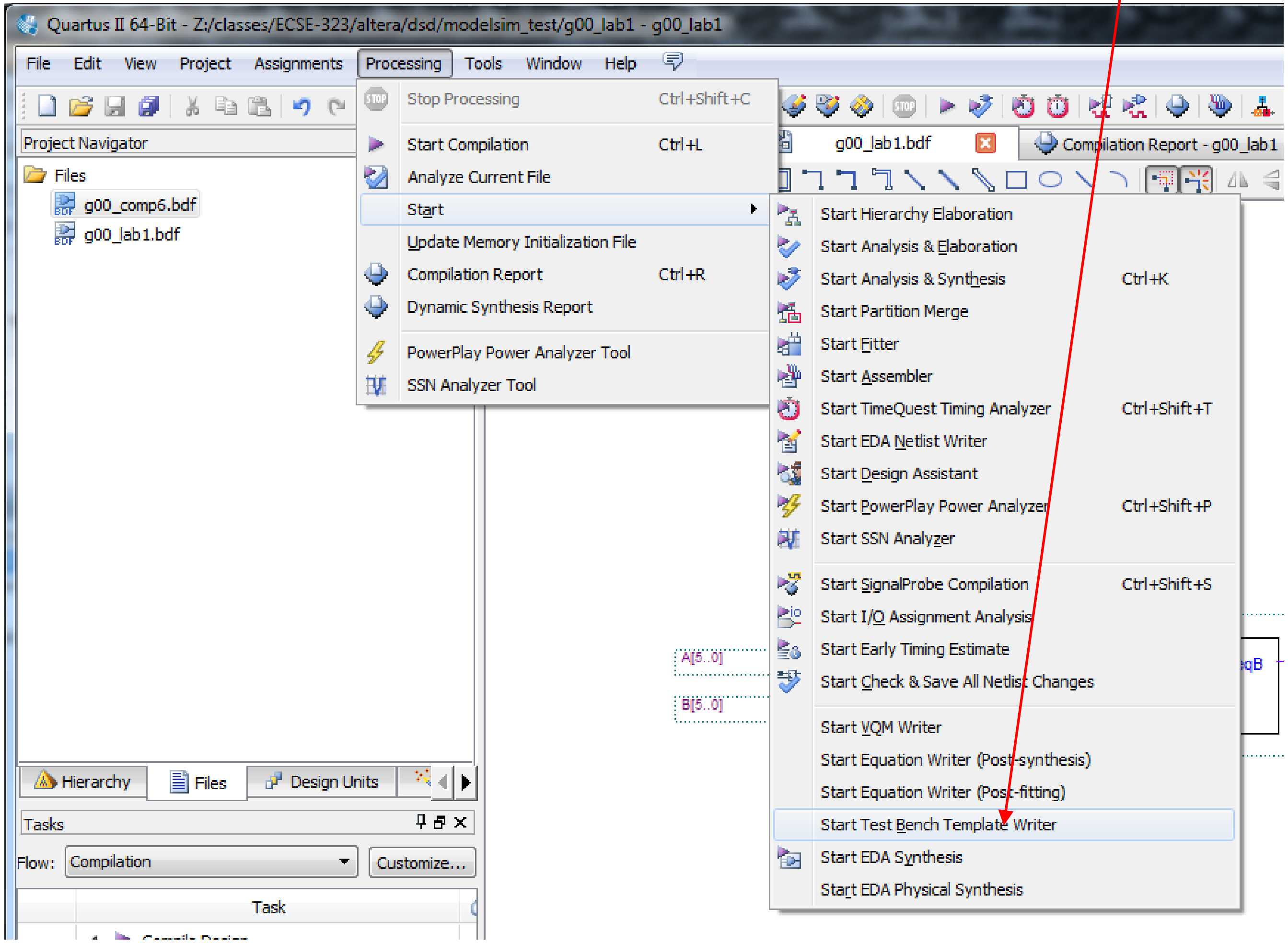
This describes a signal that oscillates between values of 0 and 1 with a period of 4 nanoseconds.

In later labs you will learn other ways of defining signal waveforms in VHDL testbenches. But for this lab, you should just use this approach.

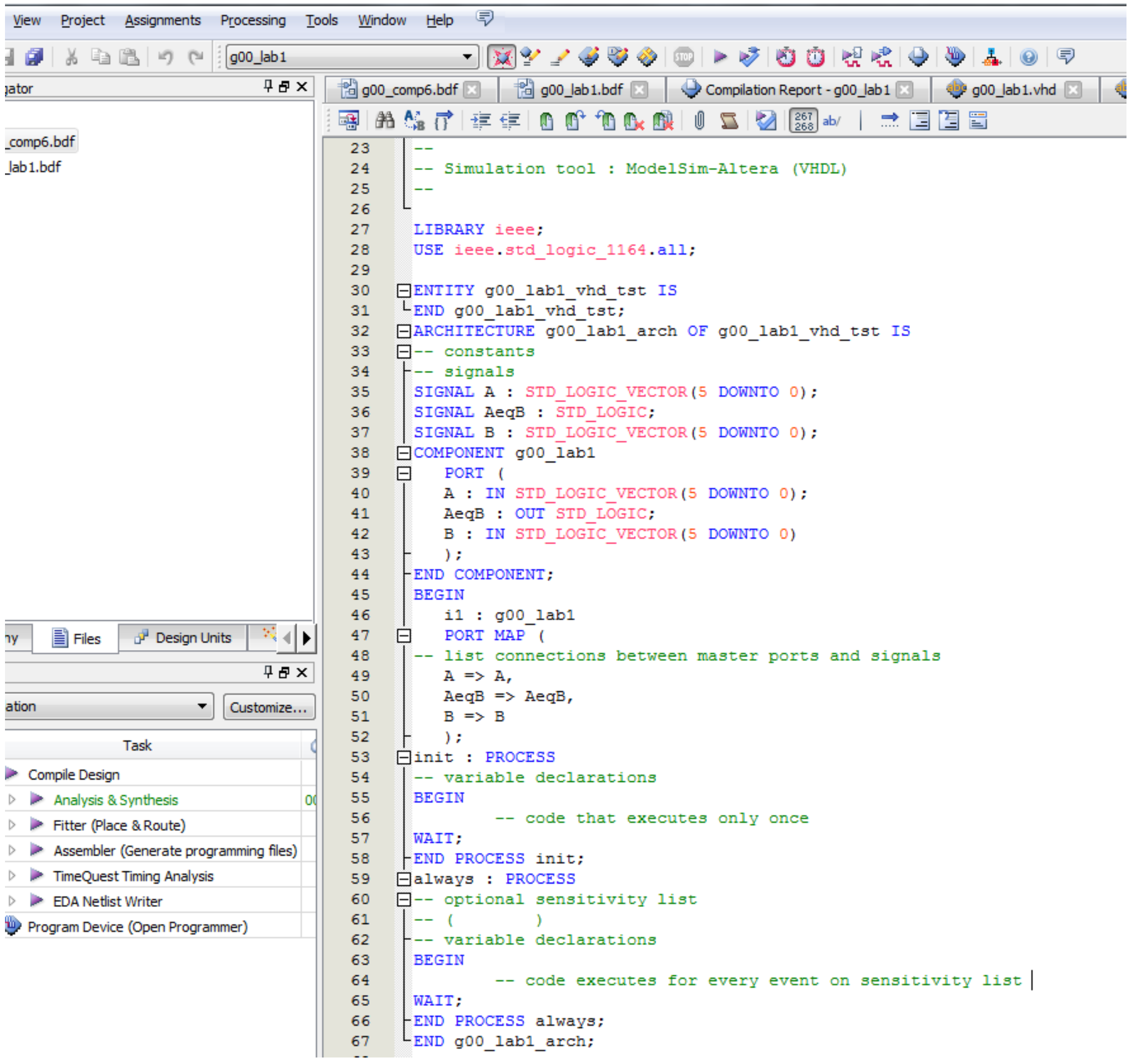
After you gain more experience you will be able to write VHDL testbenches from scratch. However, Quartus has a convenient built-in process, called the ***Test Bench Writer***, which produces a VHDL template from your design that will get you started.

Go back to the Quartus program, making sure that you have the gNN_lab1 project loaded.

Then, in the **Processing** toolbar item, select **Start/Start Test Bench Template Writer**


Quartus II 64-Bit - Z:/classes/ECSE-323/altera/dsd/modelsim_test/g00_lab1 - g00_lab1
File Edit View Project Assignments Processing Tools Window Help
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R
Dynamic Synthesis Report
PowerPlay Power Analyzer Tool
SSN Analyzer Tool
Start Hierarchy Elaboration
Start Analysis & Elaboration
Start Analysis & Synthesis Ctrl+K
Start Partition Merge
Start Fitter
Start Assembler
Start TimeQuest Timing Analyzer Ctrl+Shift+T
Start EDA Netlist Writer
Start Design Assistant
Start PowerPlay Power Analyzer Ctrl+Shift+P
Start SSN Analyzer
Start SignalProbe Compilation Ctrl+Shift+S
Start I/O Assignment Analysis
Start Early Timing Estimate
Start Check & Save All Netlist Changes
Start VQM Writer
Start Equation Writer (Post-synthesis)
Start Equation Writer (Post-fitting)
Start Test Bench Template Writer
Start EDA Synthesis
Start EDA Physical Synthesis
Project Navigator
Files
g00_comp6.bdf
g00_lab1.bdf
Compilation Report - g00_lab1
A[5..0]
B[5..0]
Hierarchy Files Design Units
Tasks
Flow: Compilation
Customize...
Task

This will generate a VHDL file named gNN_lab1.vht and place it in the simulation/modelsim directory. Open it up in Quartus. It will look something like this:


View Project Assignments Processing Tools Window Help
g00_lab1
g00_comp6.bdf
g00_lab1.bdf
Compilation Report - g00_lab1
g00_lab1.vhd
Files
Design Units
Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Program Device (Open Programmer)

```
--
-- Simulation tool : ModelSim-Altera (VHDL)
--

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY g00_lab1_vhd_tst IS
END g00_lab1_vhd_tst;
ARCHITECTURE g00_lab1_arch OF g00_lab1_vhd_tst IS
-- constants
-- signals
SIGNAL A : STD_LOGIC_VECTOR(5 DOWNTO 0);
SIGNAL AeqB : STD_LOGIC;
SIGNAL B : STD_LOGIC_VECTOR(5 DOWNTO 0);
COMPONENT g00_lab1
   PORT (
   A : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
   AeqB : OUT STD_LOGIC;
   B : IN STD_LOGIC_VECTOR(5 DOWNTO 0)
   );
END COMPONENT;
BEGIN
   i1 : g00_lab1
   PORT MAP (
-- list connections between master ports and signals
   A => A,
   AeqB => AeqB,
   B => B
   );
init : PROCESS
-- variable declarations
BEGIN
        -- code that executes only once
WAIT;
END PROCESS init;
always : PROCESS
-- optional sensitivity list
-- (        )
-- variable declarations
BEGIN
        -- code executes for every event on sensitivity list
WAIT;
END PROCESS always;
END g00_lab1_arch;
```

Note that the template already includes the instantiation of the gNN_lab1 component.

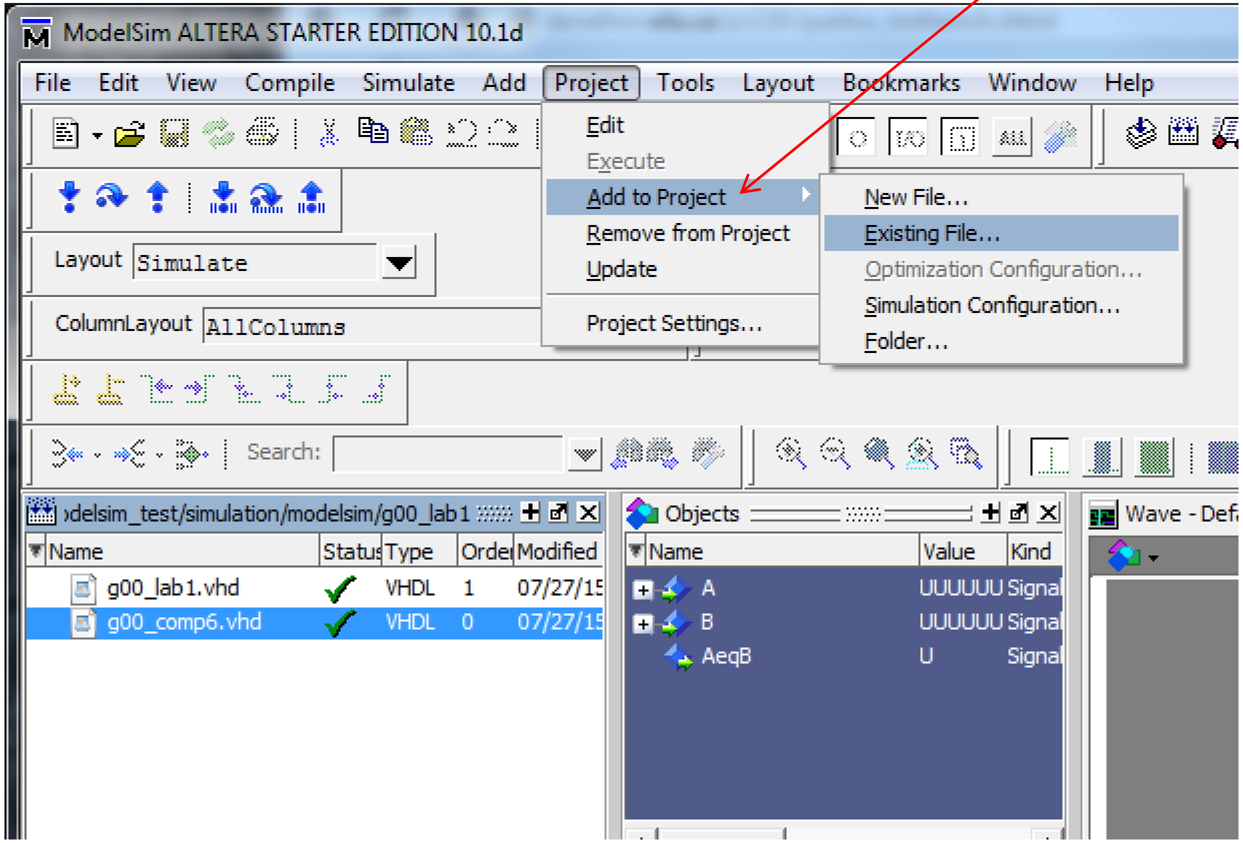
It also includes the skeletons of two process blocks, one labeled “*init*” and the other labeled “*always*”.

The init process block can be deleted. You should edit the “always” process block to suit your needs, so in this case it will be used to generate the A and B signal waveforms.

For a simple introductory example, replace the “always” process block with the following text:

```
always : PROCESS
BEGIN
                  A <= "000000";
                  B <= "000000";
                  wait for 10 ns;
                  A <= "101100";
                  wait for 15 ns;
                  B <= "101100";
                  wait for 5 ns;
                  A <= "111101";
                  wait for 20 ns;
                  B <= "111101";
                  wait; -- this waits forever…
END PROCESS always;
```

Once you have finished editing the testbench file, you need to add it to the project in ModelSim:


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Edit
Execute
Add to Project
Remove from Project
Update
Project Settings...
New File...
Existing File...
Optimization Configuration...
Simulation Configuration...
Folder...
Layout Simulate
ColumnLayout AllColumns
Search:
Objects
Wave - Def

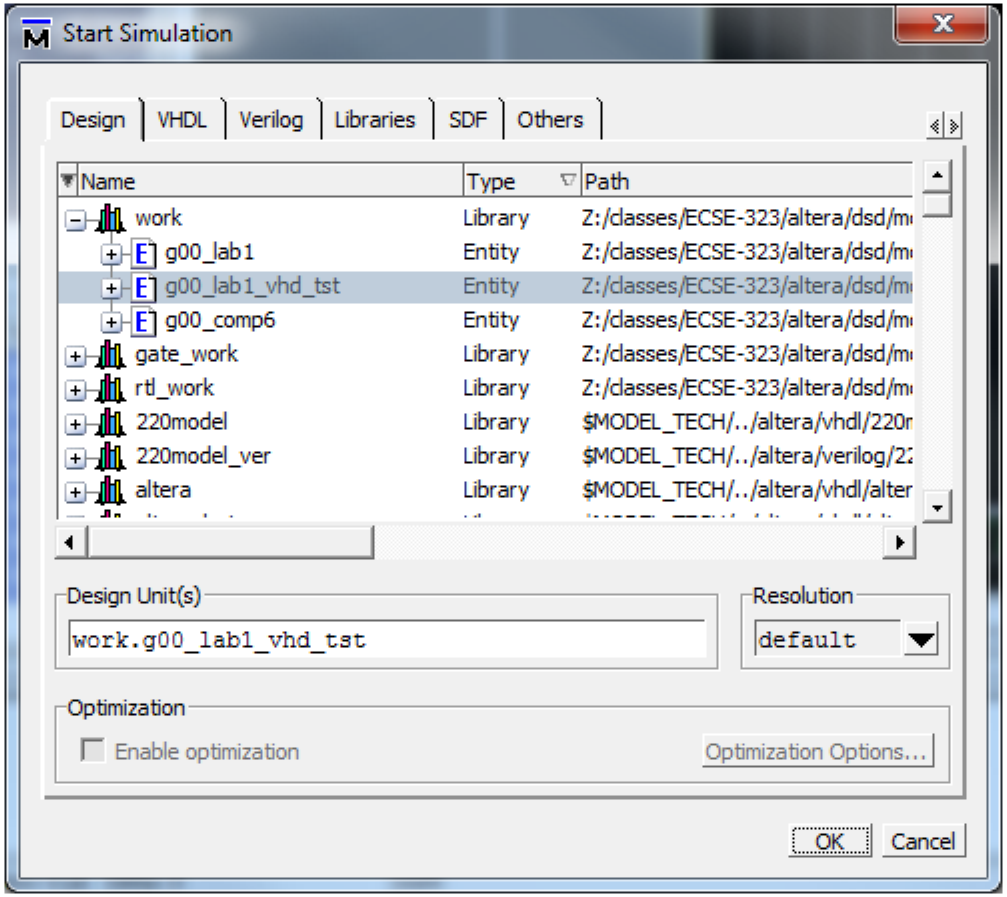
Name	Status	Type	Order	Modified
g00_lab1.vhd	✓	VHDL	1	07/27/15
g00_comp6.vhd	✓	VHDL	0	07/27/15

Name	Value	Kind
A	UUUUUU	Signal
B	UUUUUU	Signal
AeqB	U	Signal

Once the testbench file has been added to the project, you should select the testbench file in the Project pane, and click on Compile Selected from the Compile toobar item. This will compile the testbench file.

Now everything is ready for you to actually run a simulation!

Select “Start Simulation” from the Simulate toolbar item in the ModelSim program. The following window will popup:



Start Simulation

Tabs: Design | VHDL | Verilog | Libraries | SDF | Others

Name	Type	Path
work	Library	Z:/classes/ECSE-323/altera/dsd/m
g00_lab1	Entity	Z:/classes/ECSE-323/altera/dsd/m
g00_lab1_vhd_tst	Entity	Z:/classes/ECSE-323/altera/dsd/m
g00_comp6	Entity	Z:/classes/ECSE-323/altera/dsd/m
gate_work	Library	Z:/classes/ECSE-323/altera/dsd/m
rtl_work	Library	Z:/classes/ECSE-323/altera/dsd/m
220model	Library	$MODEL_TECH/../altera/vhdl/220
220model_ver	Library	$MODEL_TECH/../altera/verilog/2
altera	Library	$MODEL_TECH/../altera/vhdl/alter

Design Unit(s): work.g00_lab1_vhd_tst

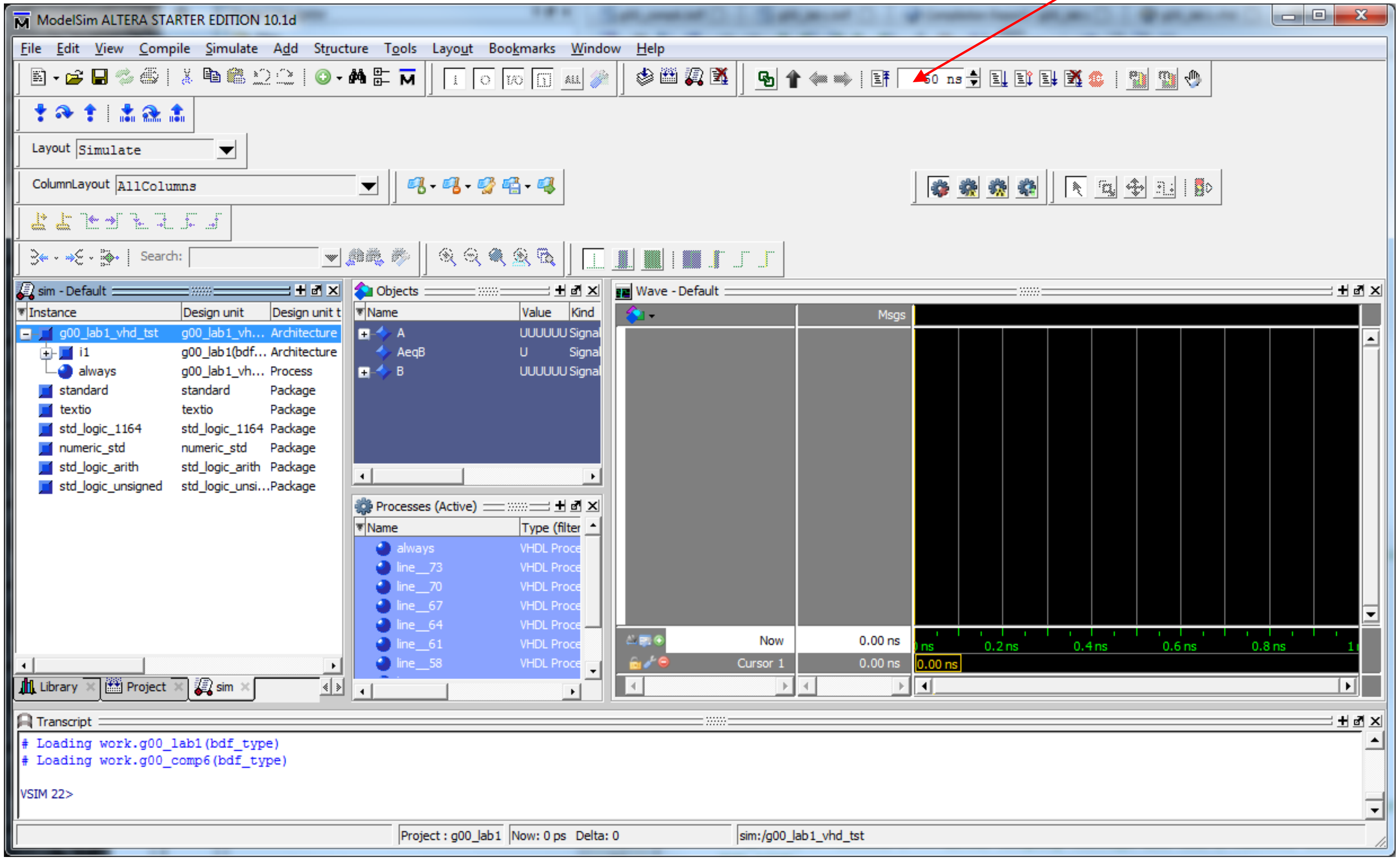
Resolution: default

Optimization: ☐ Enable optimization — Optimization Options...

OK | Cancel

Select the g00_lab1_tst entity and click on OK

The ModelSim window should now look like this. Enter a value of 60ns into the simulation length window.



Instance	Design unit	Design unit t
g00_lab1_vhd_tst	g00_lab1_vh...	Architecture
i1	g00_lab1(bdf...	Architecture
always	g00_lab1_vh...	Process
standard	standard	Package
textio	textio	Package
std_logic_1164	std_logic_1164	Package
numeric_std	numeric_std	Package
std_logic_arith	std_logic_arith	Package
std_logic_unsigned	std_logic_unsi...	Package

Name	Value	Kind
A	UUUUUU	Signal
AeqB	U	Signal
B	UUUUUU	Signal

Name	Type (filter
always	VHDL Proce
line__73	VHDL Proce
line__70	VHDL Proce
line__67	VHDL Proce
line__64	VHDL Proce
line__61	VHDL Proce
line__58	VHDL Proce

```
Now       0.00 ns
Cursor 1  0.00 ns
```

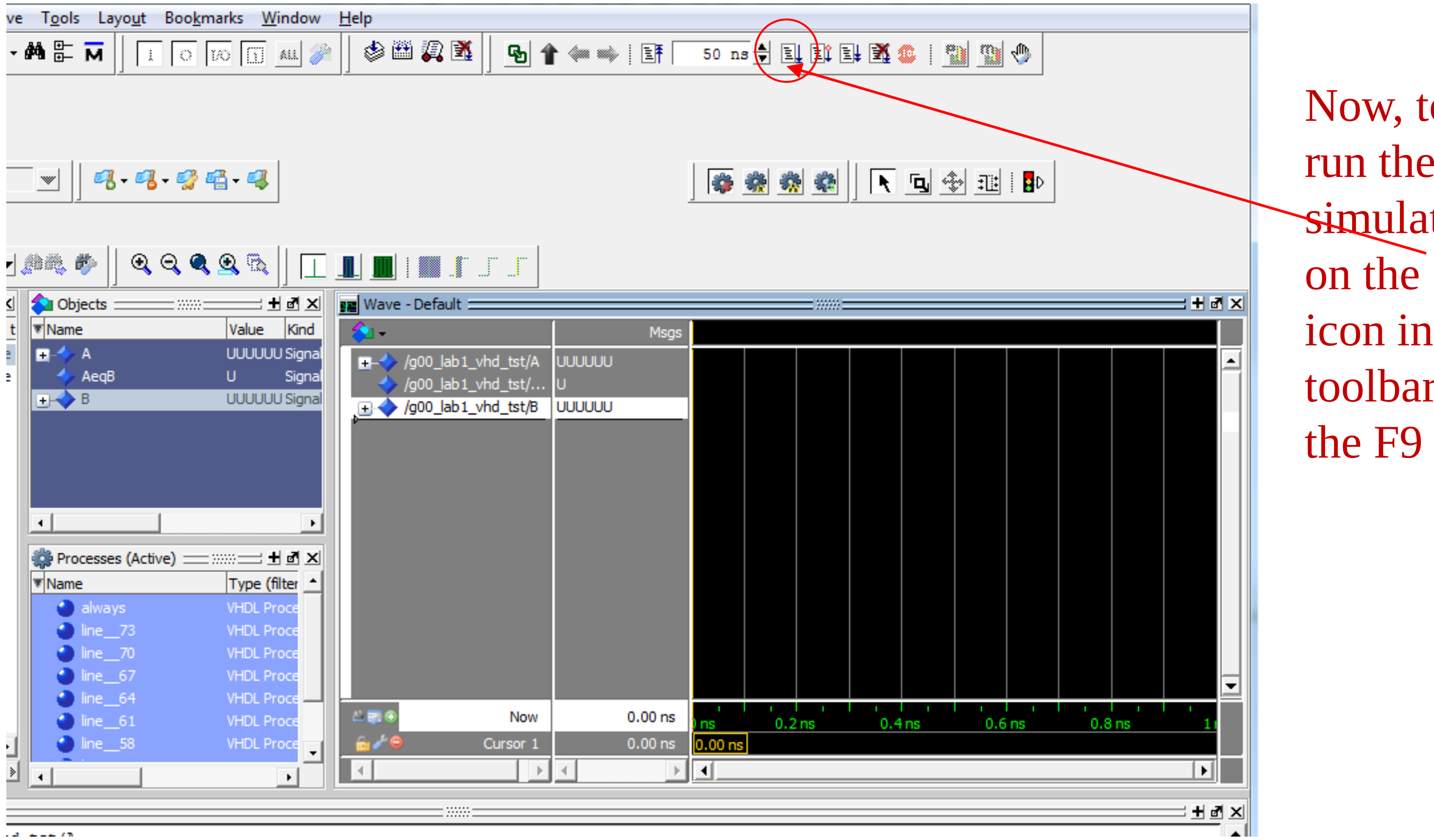
```
# Loading work.g00_lab1(bdf_type)
# Loading work.g00_comp6(bdf_type)

VSIM 22>
```

Project : g00_lab1 Now: 0 ps Delta: 0 sim:/g00_lab1_vhd_tst

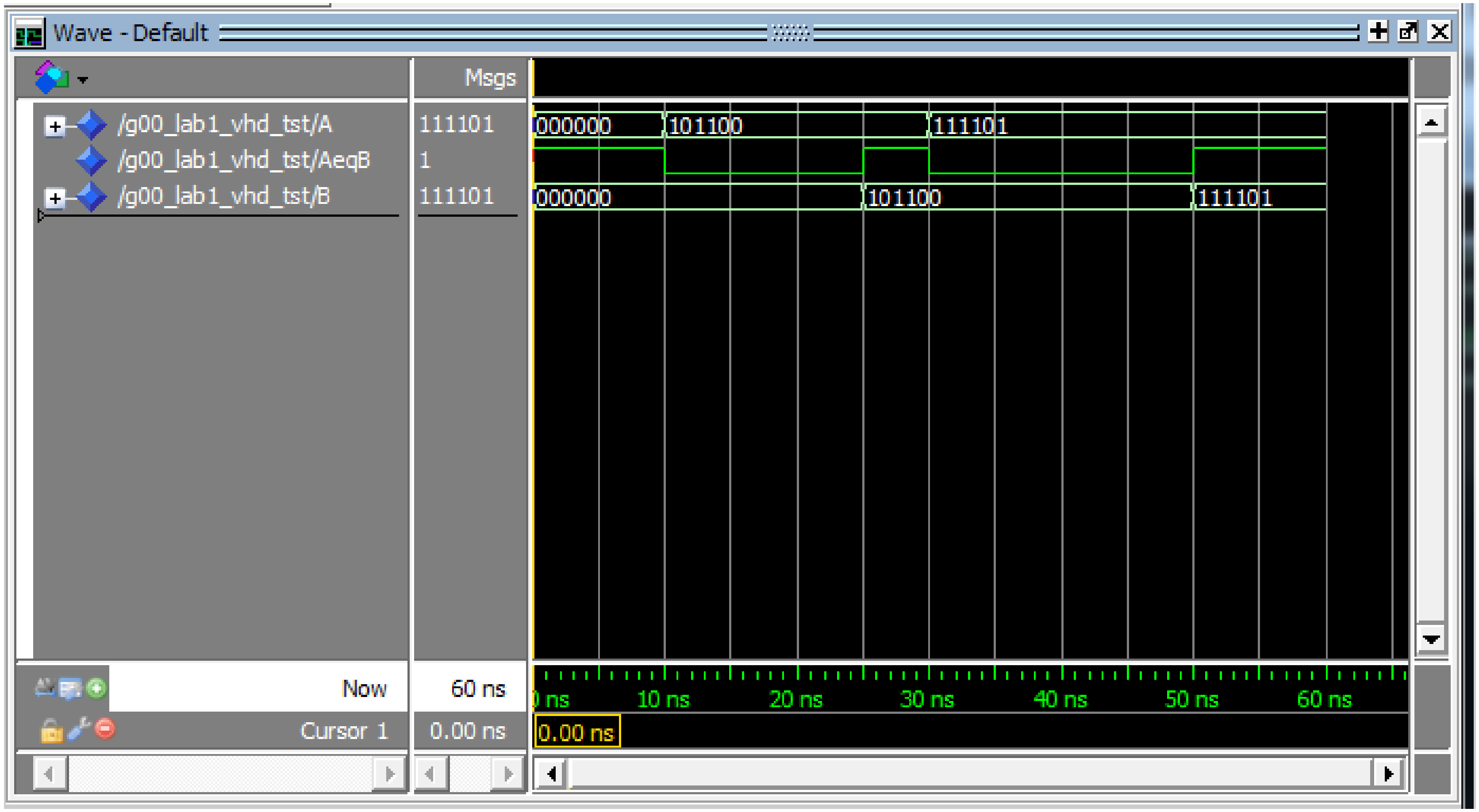
At first, the “Wave” window will not have any signals in it. You can drag signals from the “Objects” window by click on a signal, holding down the mouse button, and dragging the signal over to the Wave window. Do this for all three signals.

The Wave window will now look like this:



Now, to actually run the simulation, click on the “Run” icon in the toolbar (or press the F9 key).

Here is the output you should get (you can right-click in the right-hand pane and select "Zoom Full" to see the entire time range).

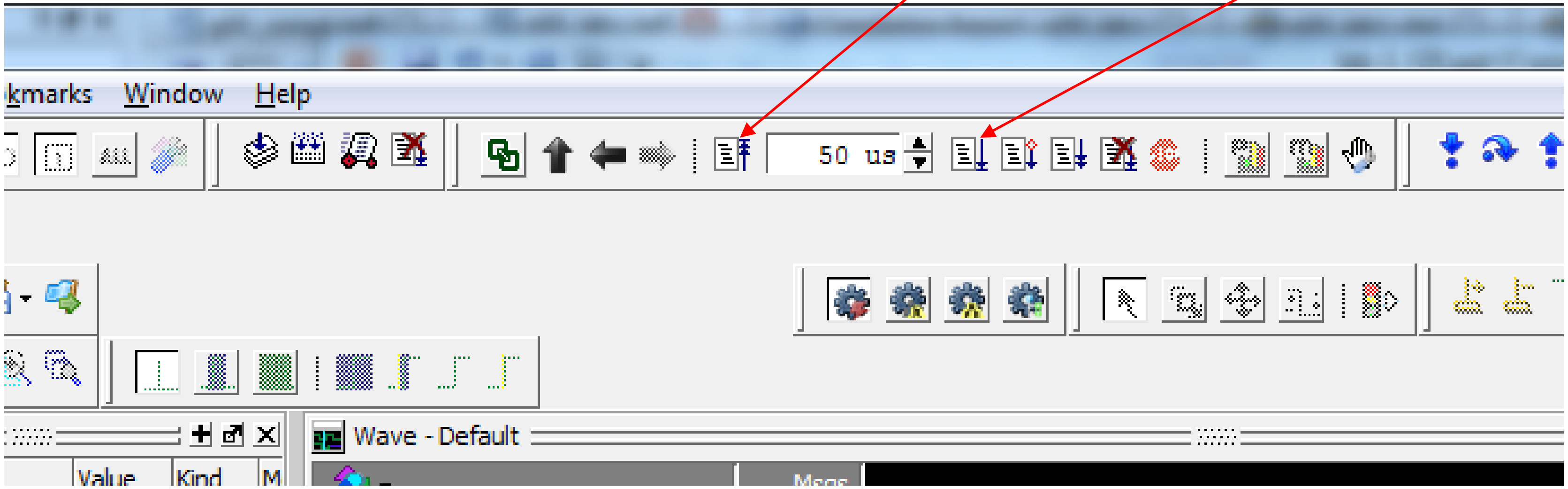


Show your simulation waveforms to the TA.

If you get an incorrect output waveform, you will have to go back and look at your design. If you make a correction to either of your schematic diagrams, you will have to re-run the conversion to VHDL (using the File/Create/Update/Create HDL Design File from Current File… command).

Then you will have to re-run the compilation of the changed files in ModelSim.

Finally, to rerun the simulation, first click on the “Restart” button, then click on the “Run” button.



TIME CHECK

You should be at least this far at the end of your *second* 2-hour lab period.

5. Exhaustive Testing of the Project using ModelSim

The simulation you ran in the previous part of the lab just had a couple of input signal transitions, and did not test all possible input patterns.

There are 2^12 or 4096 possible patterns, so complete testing of the circuit will require you to simulate all of these patterns.

If you used the approach used in the earlier testbench you would need thousands of statements in order to define each possible case. It is simpler to have a FOR LOOP, and increment the values of A and B in the loop to run through all the 4096 possible cases. This is done in the testbench shown on the next two pages.

The process block generates all of the possible input patterns using two nested FOR loops. The RANGE attribute is equivalent to specifying the loop range as over the minimum value of the signal to its maximum value. For example, **A'RANGE** is equivalent to **0 to 63**.

```
------- Testbench for validating and simulating the 6-input comparator circuit ---------------
--
-- put the designer(s) names here
-- put the date here
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all; -- added to the template

ENTITY g00_lab1_vhd_tst IS
END g00_lab1_vhd_tst;
ARCHITECTURE g00_lab1_arch OF g00_lab1_vhd_tst IS
-- constants
-- signals
SIGNAL A : STD_LOGIC_VECTOR(5 DOWNTO 0);
SIGNAL AeqB : STD_LOGIC;
SIGNAL B : STD_LOGIC_VECTOR(5 DOWNTO 0);

COMPONENT g00_lab1
          PORT (
          A : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
          AeqB : OUT STD_LOGIC;
          B : IN STD_LOGIC_VECTOR(5 DOWNTO 0)
          );
END COMPONENT;
```

Continued on next page…

```
-------------------------------------------------------------------------------
BEGIN
           i1 : g00_lab1 PORT MAP (A => A,AeqB => AeqB,B => B); --instantiate component being tested

generate_test : PROCESS
BEGIN
    FOR i IN 0 to 63 LOOP -- loop over all A values
    A <= std_logic_vector(to_unsigned(i,6)); -- convert the loop variable i to std_logic_vector
    FOR j IN 0 to 63 LOOP -- loop over all B values
        B <= std_logic_vector(to_unsigned(j,6)); -- convert the loop variable i to std_logic_vector

        WAIT FOR 10 ns; -- suspend process for 10 nanoseconds at the start of each loop

    END LOOP; -- end the j loop
    END LOOP; -- end the i loop

    WAIT; -- we have gone through all possible input patterns, so suspend simulator forever

END PROCESS generate_test;

END g00_lab1_arch;
-------------------------------------------------------------------------------
```

Modify your testbed file so that it matches that shown on the previous two pages.

In the ModelSim program recompile the testbed file and then restart and re-run the simulation.

Look at the simulated cases and check to see if they are correct. You don't need to look at all of them, as that would take a long time. Just look at some representative cases. Later in the course you will learn a method for having ModelSim do the complete error checking for you.

Show your simulation waveforms to the TA.

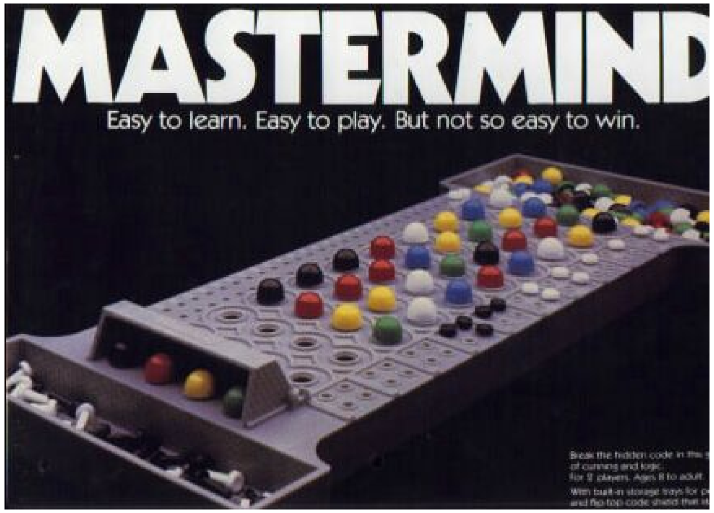
TIME CHECK

You should be at least this far at the end of your *third* 2-hour lab period.

The lab project for this term will be to develop an electronic version of the classic board game ***Mastermind***. This point of the game is to determine the colours of hidden pegs chosen by the opponent. You make guesses and the opponent states how many pegs you have guessed correctly, and how many pegs have the same colour as your guesses. Based on this feedback, you come up with a new guess. If you can guess the correct combination within a certain number of tries (say 7-10 or so) then you win.

More details on the Mastermind game will be given in future labs, but you can get a head-start by doing a search on the web for more information.

Throughout the 5 lab experiments, you will develop all of the building blocks for the Mastermind player, and integrate them into a complete user-friendly system, using an FPGA development board.



The image is from the web site of Toby Nelson (http://www.tnelson.demon.co.uk/mastermind/index.html)

6. Design of the ones-counter circuit .

As part of the Mastermind game scoring circuit, you will need a module that can count the number of '1' bits in a 4-bit input vector. For example, the input x = “1101” has 3 ones, so the output would be N = “011”.

Since there are 5 possible counts (0,1,2,3,4) the module will need to have a 3-bit output. The entity declaration for this module’s VHDL description should look like the following:

```
entity g00_num1s is
 port ( X     : in std_logic_vector(3 downto 0);
        num1s : out std_logic_vector(2 downto 0));
end g00_num1s;
```

The first step in the design of the ones-counter circuit is the derivation of the Boolean logic equations describing the circuit function. You should derive a Boolean expression for each output bits.

Use the Karnaugh map technique to find minimal sum-of-products forms for these expressions. As an example, the solution for the most-significant-bit is shown below:

N(2)

	$\overline{x1}\,\overline{x2}$	$\overline{x1}x2$	$x1x2$	$x1\overline{x2}$
$\overline{x3}\,\overline{x4}$	0	0	0	0
$\overline{x3}x4$	0	0	0	0
$x3x4$	0	0	$\textcircled{1}$	0
$x3\overline{x4}$	0	0	0	0

N(2) = (x1 . x2 . x3 . x4)

Fill in the Karnaugh maps for the other 2 output bits, N(0) and N(1). Find the minimal Sum-of-Products expression.

Show the completed K-maps and resulting minimal SOP forms to your TA and have him or her sign your grade sheet.

N(0)

	$\overline{x1}\,\overline{x2}$	$\overline{x1}x2$	$x1x2$	$x1\overline{x2}$
$\overline{x3}\,\overline{x4}$				
$\overline{x3}x4$				
$x3x4$				
$x3\overline{x4}$				

N(1)

	$\overline{x1}\,\overline{x2}$	$\overline{x1}x2$	$x1x2$	$x1\overline{x2}$
$\overline{x3}\,\overline{x4}$				
$\overline{x3}x4$				
$x3x4$				
$x3\overline{x4}$				

N(0) = ?

N(1) = ?

7. VHDL description of the ones-counter circuit

Once you have the Sum-of-Products form description of your circuit worked out, write the vhdl description for the circuit using three simple concurrent assignment statements, each describing one output bit.

Name the vhdl file "***gNN_num1s.vhd***" (where NN is your group number).

When you have completed the vhdl description, show it to your TA and have him or her sign your grade sheet.

8. Simulation of the ones-counter circuit

Once you have completed the vhdl description of the circuit, write a VHDL testbench for simulating the circuit.

Show the simulation testbench VHDL code to your TA.

Do a Modelsim simulation of the circuit, using the same approach as used earlier to simulate the 6-bit comparator circuit. Make sure that all 16 possible input patterns are tested.

When you have completed the simulation, show the resulting waveform display to your TA, demonstrating the correct outputs for all 16 input patterns.

TIME CHECK

You should be this far (i.e. have completed the lab) at the end of your *fourth* 2-hour lab period!

9. Writeup the Lab Report .

Write up a short report describing the ***gNN_num1s*** circuit that you designed in this lab. This report should be done in html or pdf, or in Microsoft Word (***pdf format is preferred!***). You do not need to report on the gNN_comp6 circuit.

The report must include the following items:

- A header listing the group number (and company name if you have one), the names and student numbers of each group member.
- A title, giving the name (e.g. ***g00_num1s***) of the circuit.
- A description of the circuit's function, listing the inputs and outputs. Provide a pinout or symbol diagram.
- A gate level schematic diagram of the circuit.
- A discussion of how the circuit was tested, showing representative simulation plots. How do you know the circuit works correctly?

The report is due one week after the end of the 2-week lab period (i.e. on October 2), at midnight.

10. Submit the Lab Report to WebCT .

The lab report, and all associated design files (by design files, I mean the schematic (.bdf) and vhdl (.vht and .vht) files) must be submitted, as an assignment to the myCourses site. Only one submission need be made per group (both students will receive the same grade!).

Combine all of the files that you are submitting into one *zip* file, and name the zip file gNN_LAB_1.zip (where NN is your group number).

Grade Sheet for Lab #1 Fall 2015.

Group Number:_______________.
Group Member Name:________________. Student Number:___________________.
Group Member Name:________________. Student Number:___________________.

Marks			TA Signatures
	1.	Schematic diagram for the 6-bit comparator	.
	2.	VHDL file for the 6-bit comparator	.
	3.	Initial partial simulation results for the 6-bit comparator	.
	4.	Complete simulation results for the 6-bit comparator	.
	5.	Boolean equations for the 1's counter circuit	.
	6.	VHDL description for the 1's counter circuit	.
	7.	Simulation Testbench VHDL for the 1's counter circuit	.
	8.	Simulation results for the 1's counter circuit	.

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.