ECSE-323
Digital System Design

Lab #1 – ***Using the Altera Quartus II Software*** Winter 2017

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.

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
- Generate a functional simulation netlist
- Create a Vector Waveform File (for simulation inputs)
- Do functional simulation of a circuit

Table of Contents

This lab consists of the following stages:

1. Introduction
2. Startup and creation of the project framework
3. Creating a schematic diagram design file for a comparator circuit
4. Performing functional simulation of the comparator circuit
5. Sketching a block diagram and laying out the schematic for a modulo-13 circuit
6. Functional simulation of the modulo-13 circuit
7. Writeup of the lab report and uploading to myCourses

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. There will also be some physical measurements required, using the lab station oscilloscopes.

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 WebCT student presentation area. This report will generally describe the circuits designed during the lab.

The lab project for this term will be to develop an electronic version of a super fun party game.

More details on the project will be given in future labs.

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

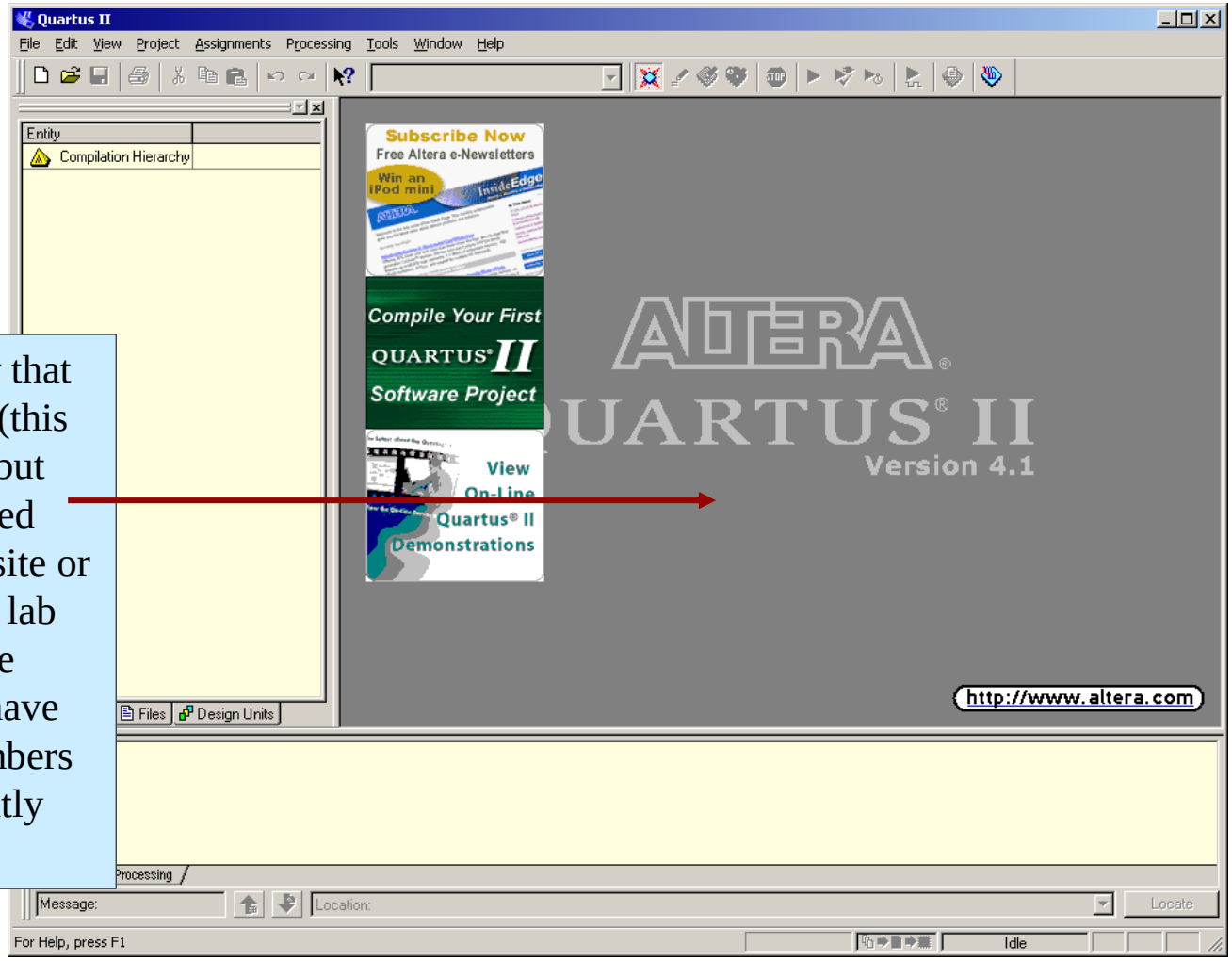
2. Startup and creation of the project framework

In this course you will be using commercial FPGA design software, the Altera ***Quartus II*** program. A slightly restricted version of this 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 most recent 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.

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 4.1 but versions downloaded from Altera's web site or the versions on the lab computers or on the textbook CD will have higher version numbers and may look slightly different)

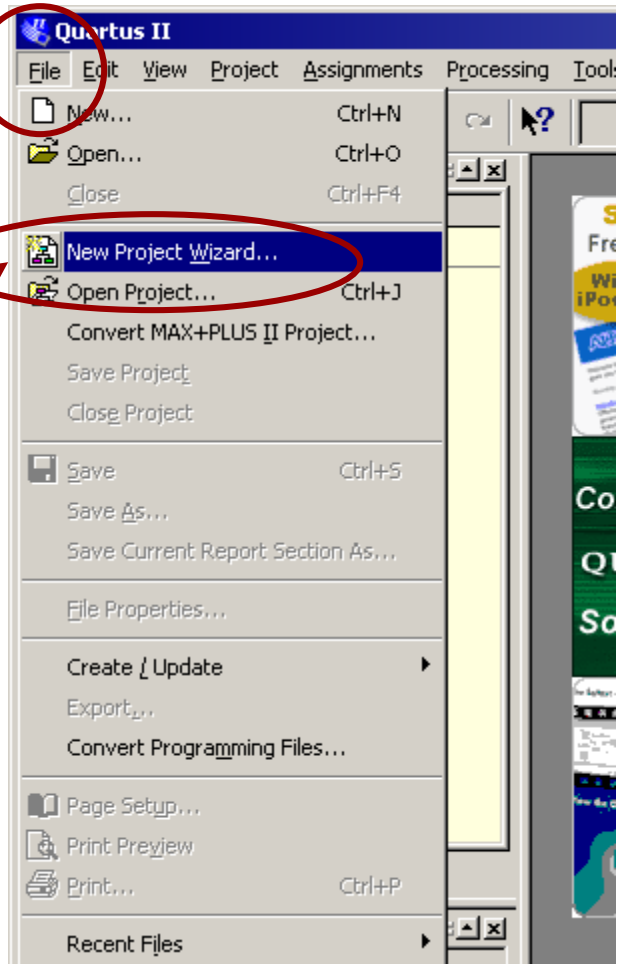


The Altera ***Quartus II*** program employs a ***project-based approach***. The goal of a Quartus project is to develop a hardware implementation of a specific function, targeted to an FPGA (*Field Programmable Gate Array*) device.

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 vector 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.

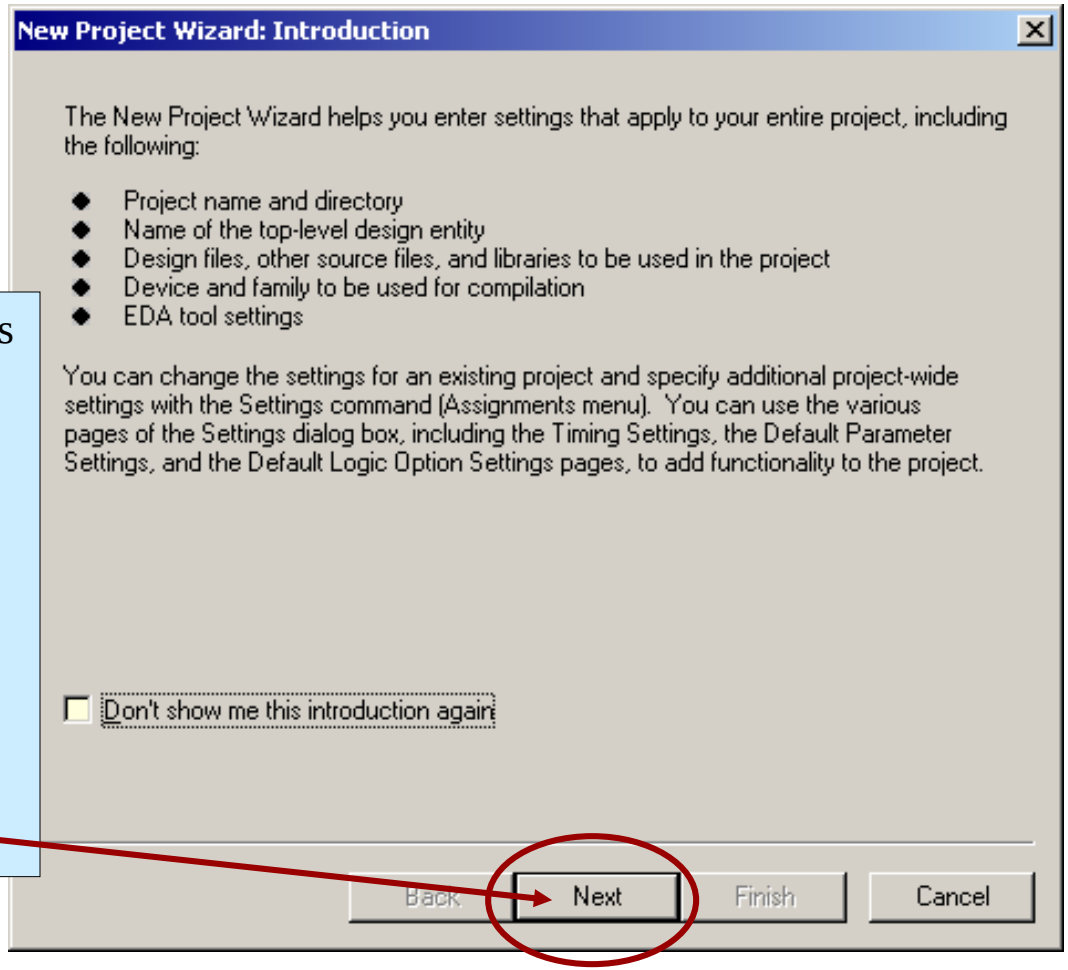
To run the Project Wizard, click on the "***File***" menu and select the "***New Project Wizard***" entry.



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.

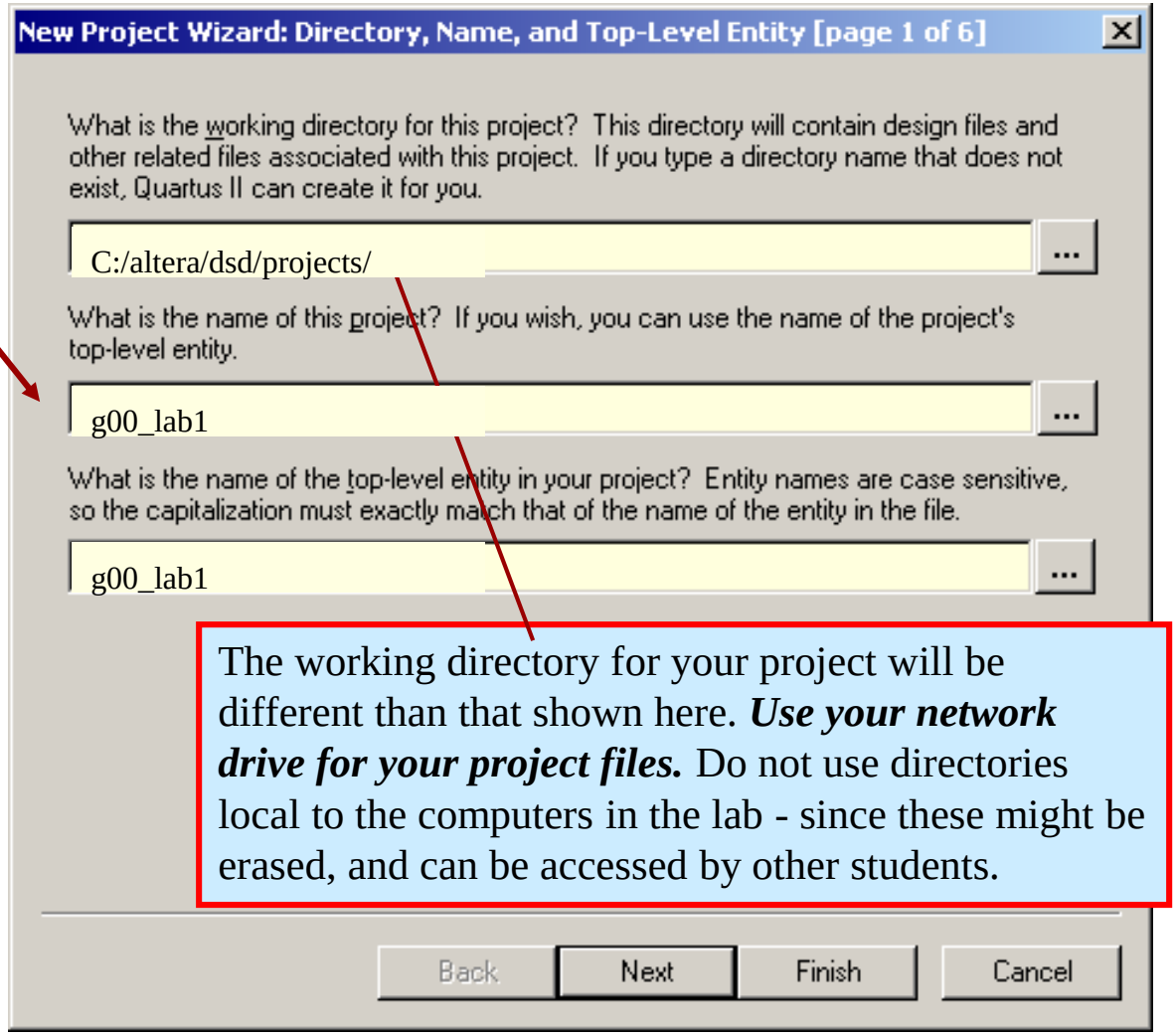


In the second window you should give the project the following name:

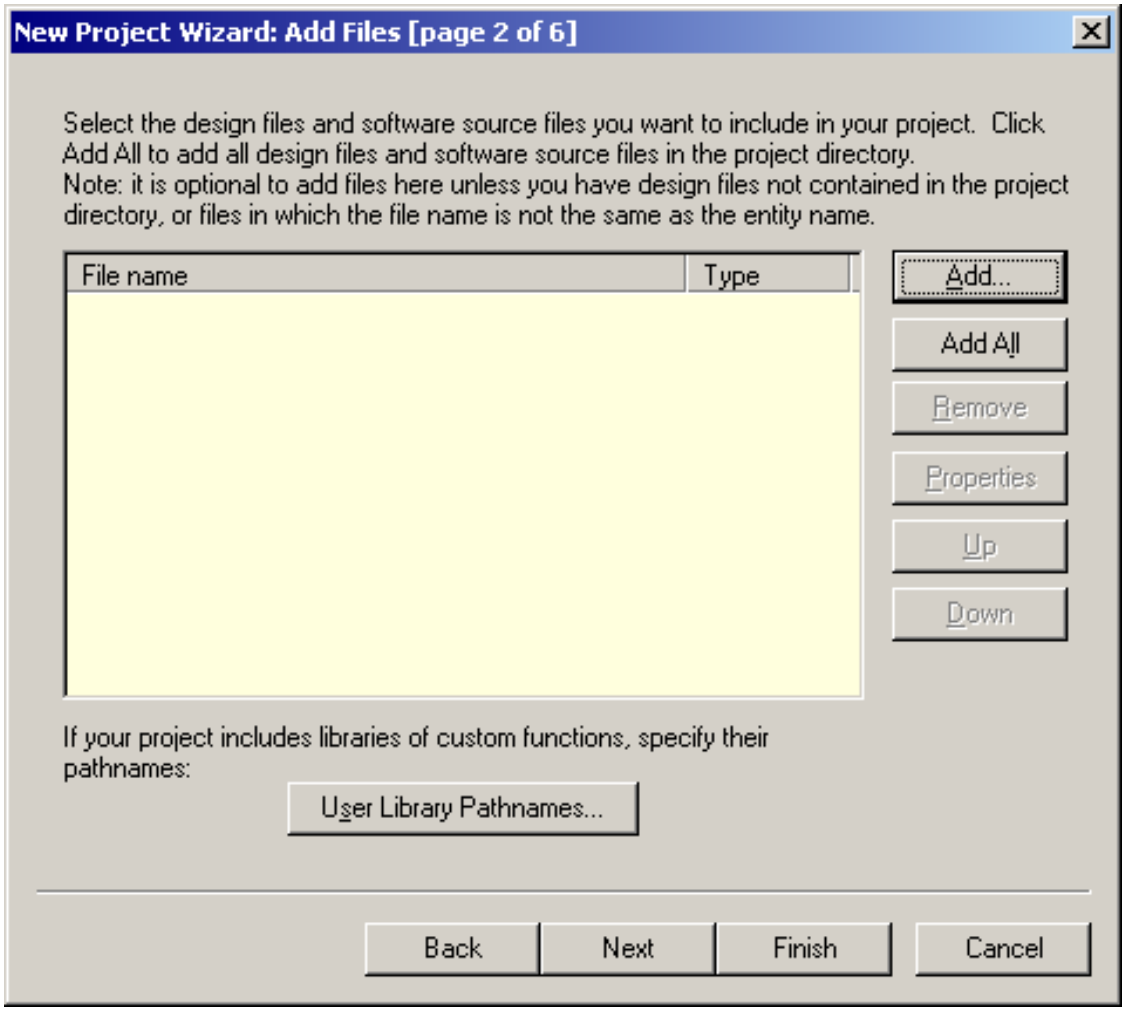
gNN_lab1

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.

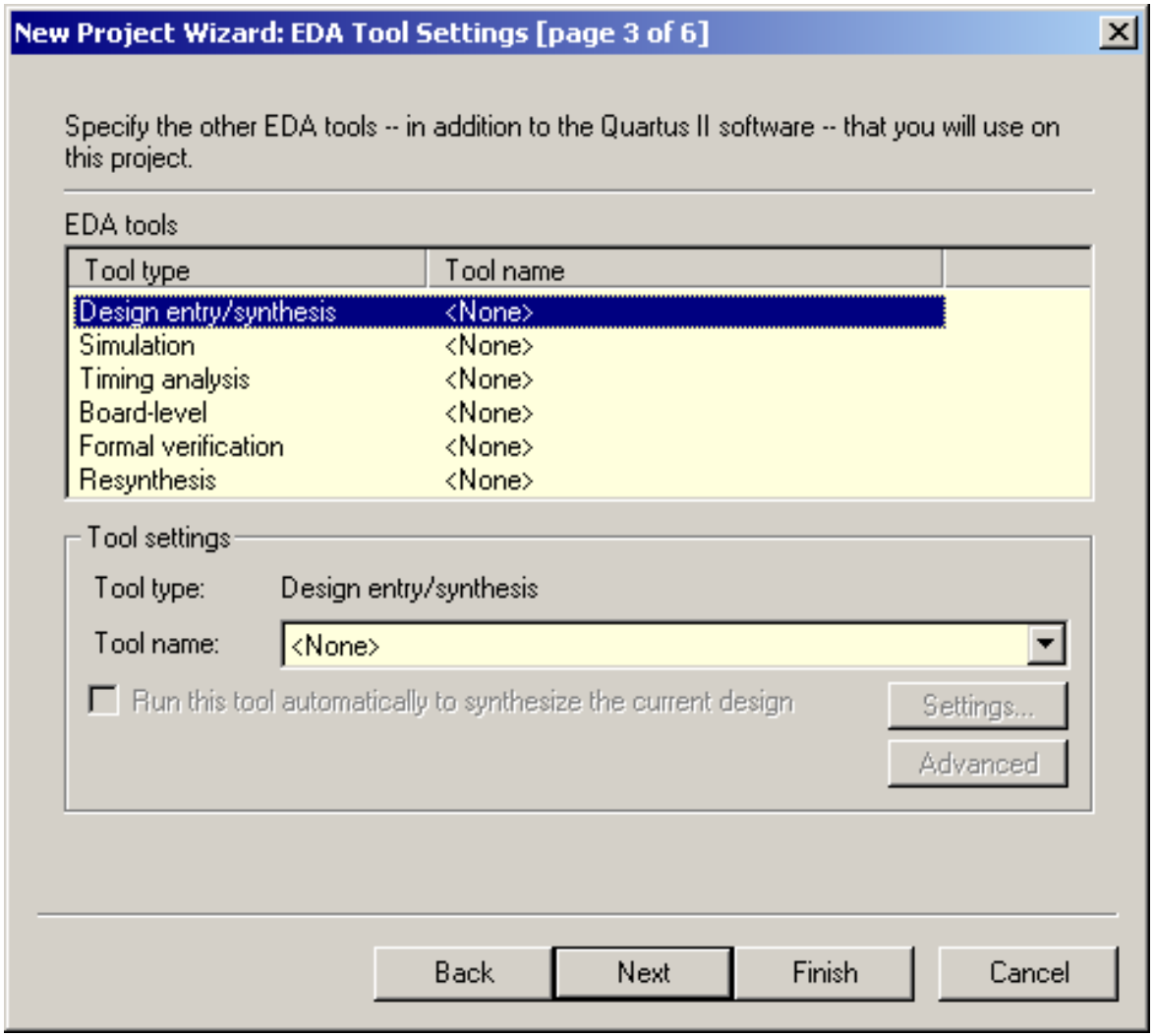
Note: ***g00_*** will be used for designs developed by the course instructors



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.



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



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

We will not be using any tools other than those provided by the Quartus II program, so just click on "Next".

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 part number:

EP2C20F484C7

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

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, we will start by designing a simple ***7-bit comparator*** circuit, which we will name ***gNN_comp7*** (where, as with the project name, NN is to be replaced with the group number).

The gNN_comp7 circuit has two 7-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(6) xnor B(6)) and (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.

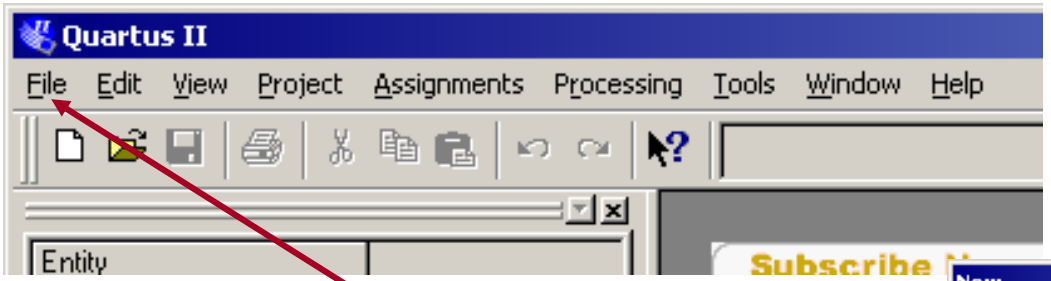
In this course, you will learn two methods for describing circuits to be implemented:

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

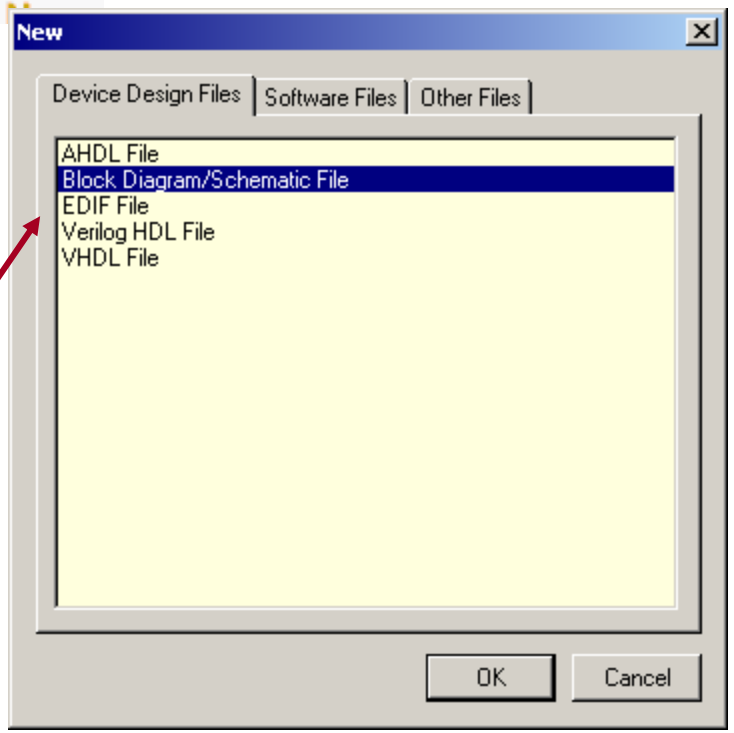
Schematic diagrams are graphical descriptions of the circuit, while VHDL uses textual descriptions.

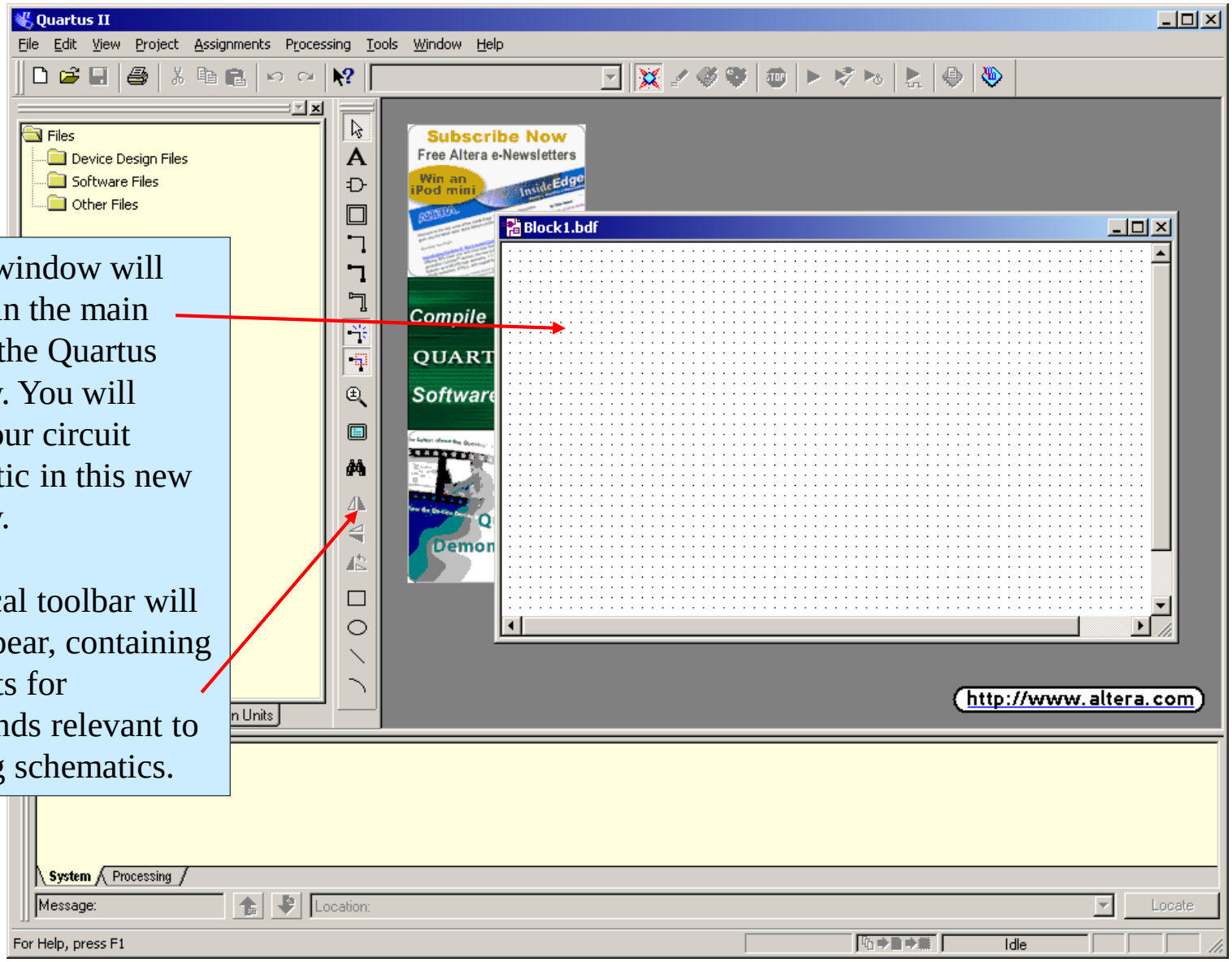
In this lab you will learn how to describe a circuit via a schematic diagram. You will learn how to describe circuits with VHDL in the next lab.

Now create a schematic diagram ***design file*** for the 7-bit comparator. Start by opening a new block diagram window…



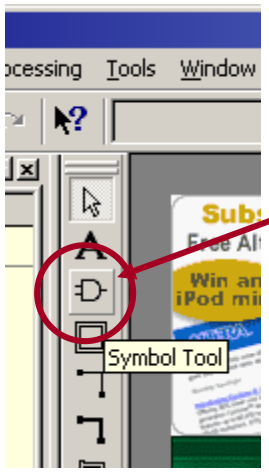
1. Click and hold on the "File" menu of the main Quartus window.
2. Select the "New" menu item. The dialog box shown at right will popup.
3. In the dialog box, select "Block Diagram/Schematic File" and click on "OK".



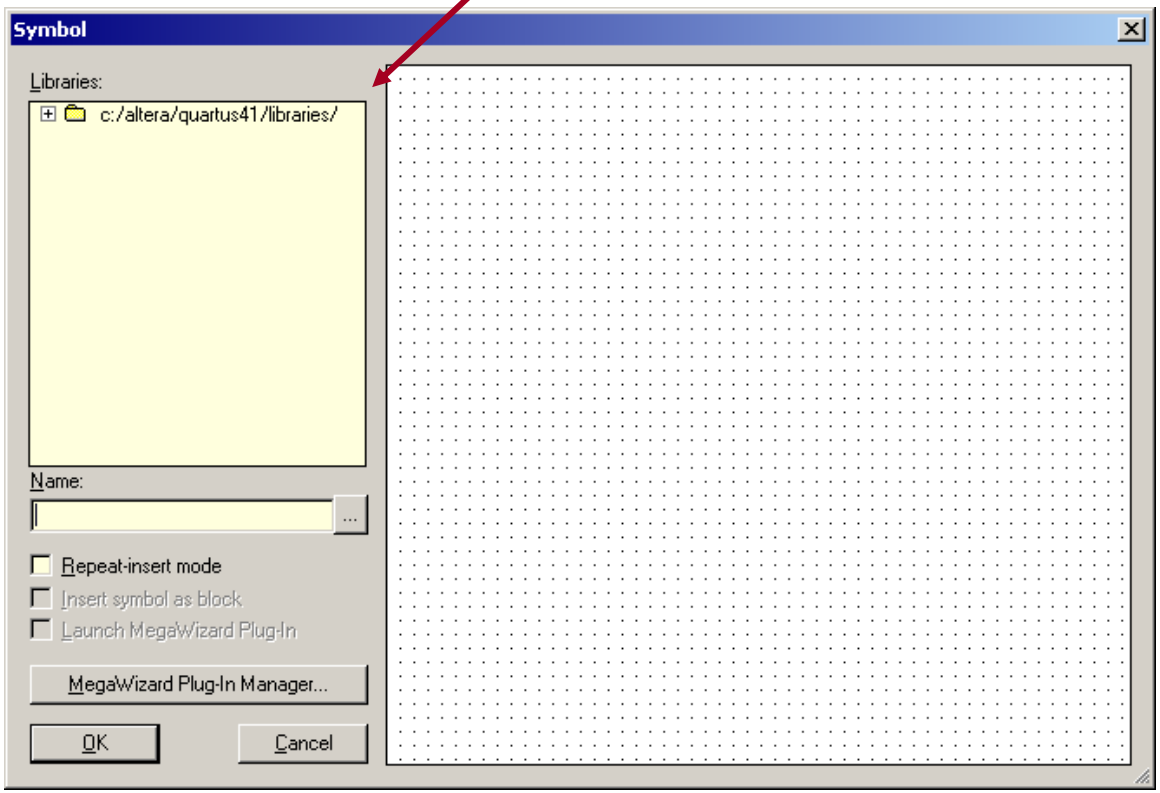


A new window will appear in the main area of the Quartus window. You will draw your circuit schematic in this new window.

A vertical toolbar will also appear, containing shortcuts for commands relevant to drawing schematics.

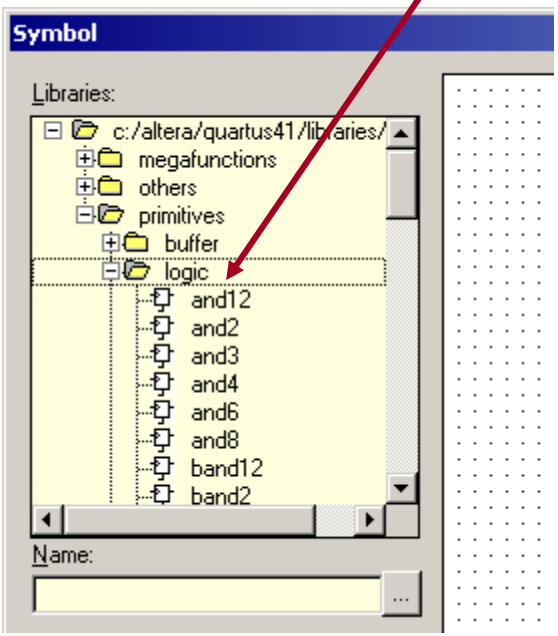


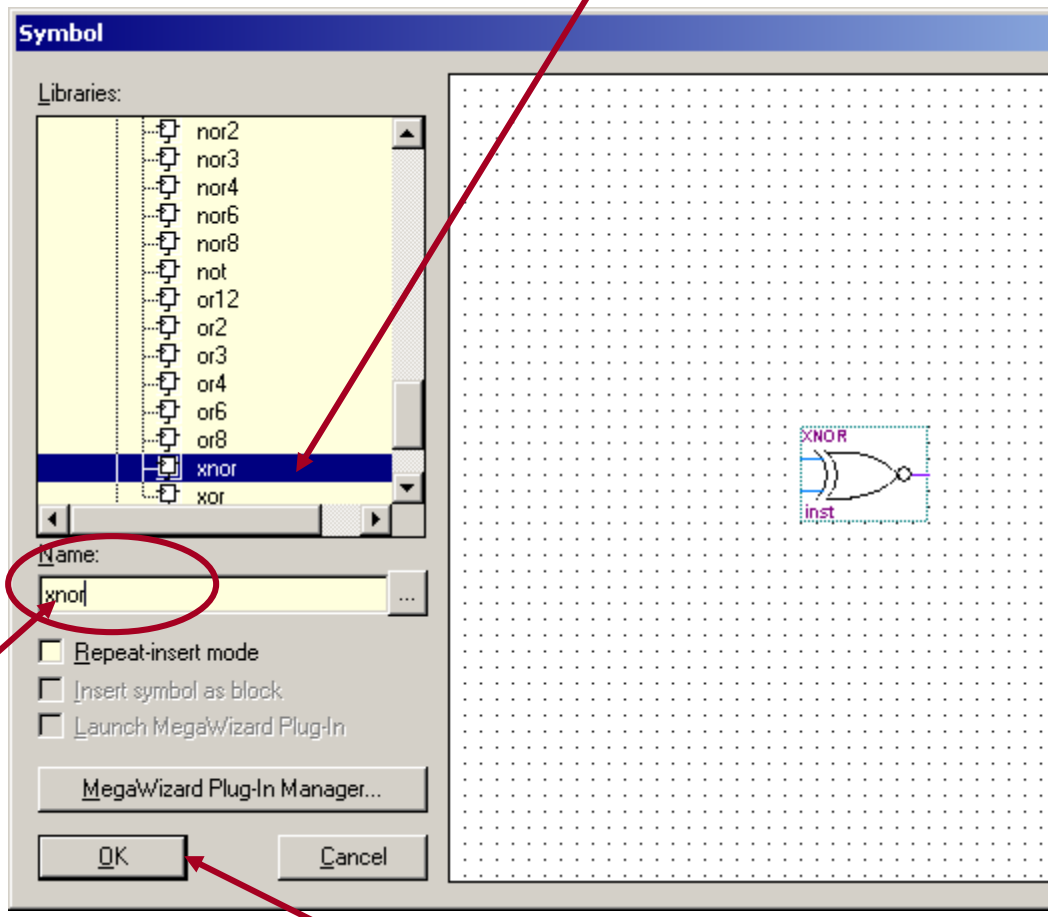
Click on the toolbar item that is shaped like an AND gate. This will bring up the "***symbol***" window that will allow you to select which symbol to add to your schematic drawing.



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

We want to add an XNOR gate symbol to our 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.



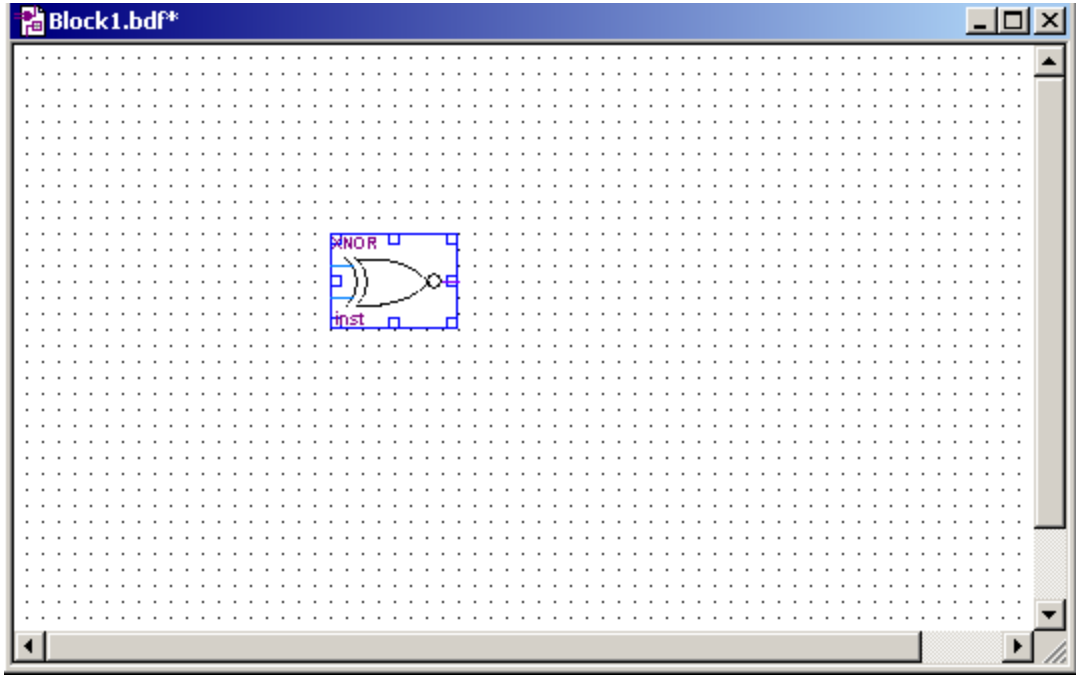


The second way is to type the symbol name directly into the Name box. You do not have to select the directory - the program will search the directory tree for the symbol.

Finally, click on OK to complete the selection.

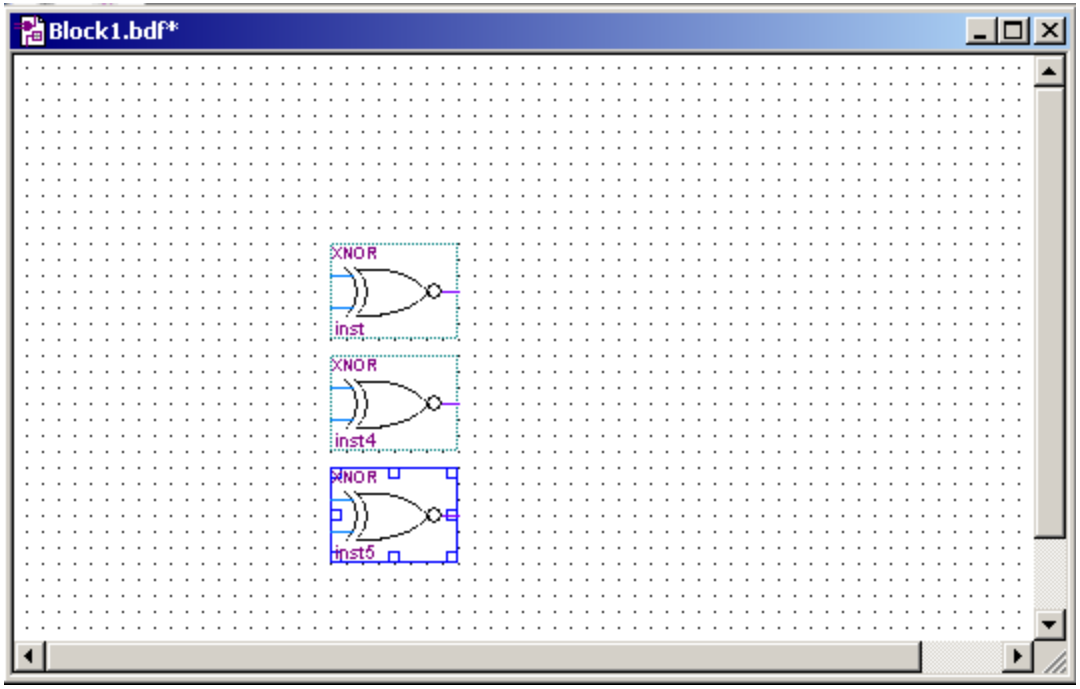
After clicking on OK in the Symbol window, a floating image of an XNOR gate will appear in the schematic window. As you move the mouse, the symbol will follow. Position the symbol where you would like it to be and left-click the mouse. The symbol will now be placed onto the schematic.

Your schematic should now look something like the figure below:



Our design for the 7-bit comparator requires seven instances of the XNOR gate. We could repeat the symbol search process six more times, but there is an easier, faster way to enter the other six XNOR gates.

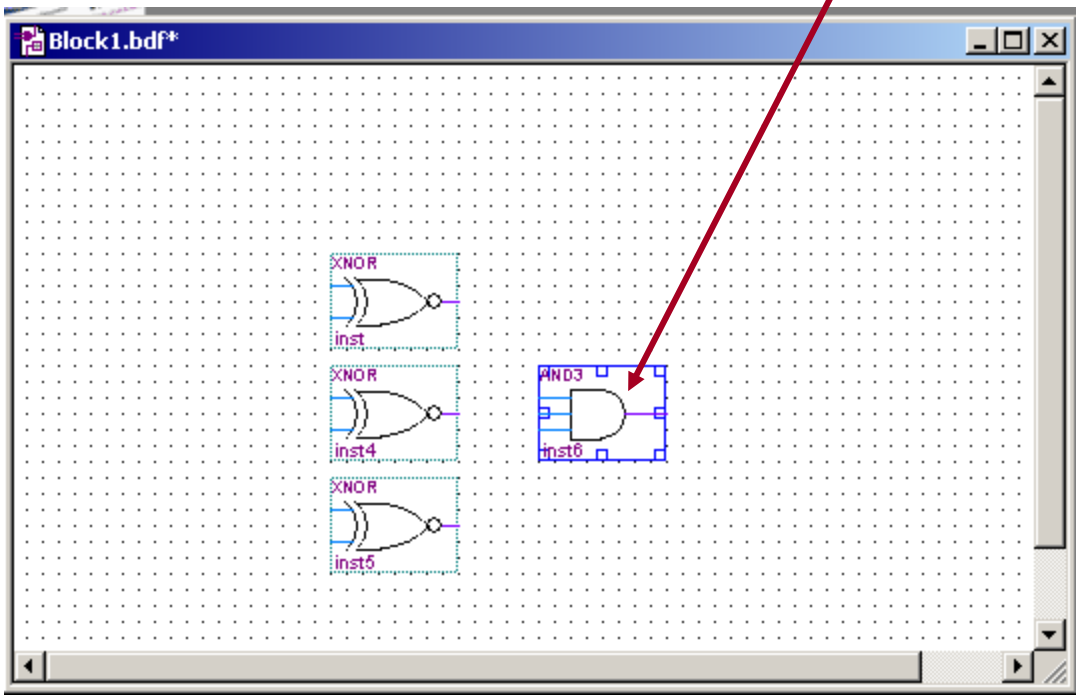
The faster way is to use ***copy-and-paste***. To ***copy*** a symbol, left-click on it, to select the symbol, and then press **Control-c** on the computer keyboard. Then left-click the mouse with the cursor positioned at the location where you want the new symbol instance to be placed. Then press **Control-v**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



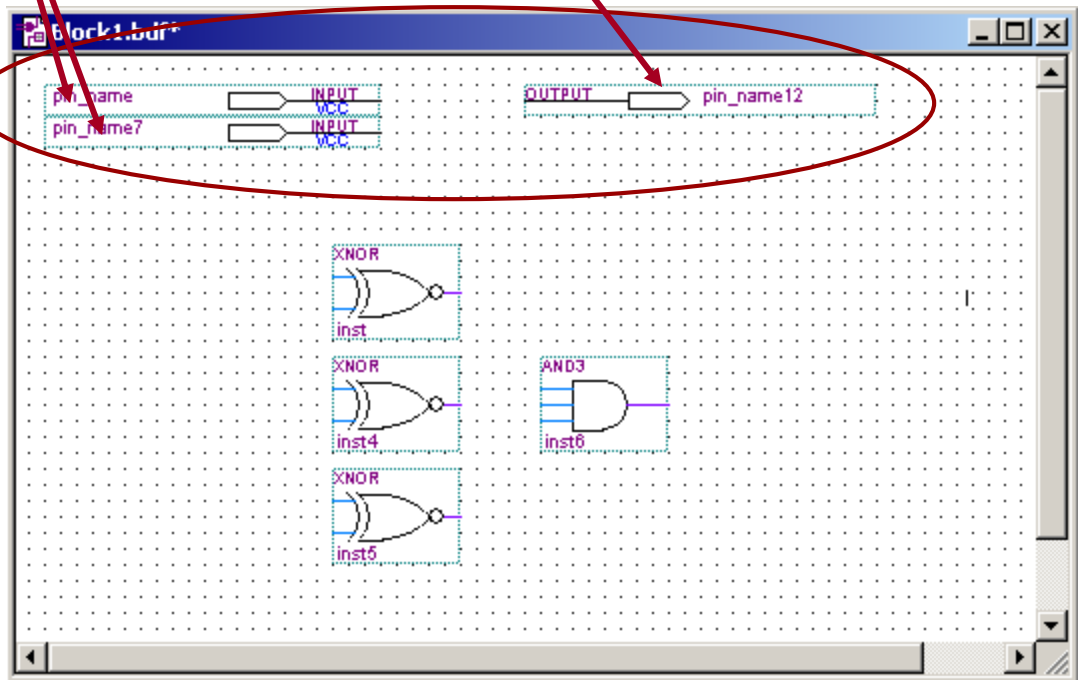
Do this copy-and-paste twice, to end up with 3 XNOR gates arranged as shown. Then select the group of three and repeat the copy-and-paste, to leave 6 XNOR gates. Then add one more to make 7.

You need to AND together the outputs of the 7 XNOR gates. This can be done with three 3-input AND gates. (there are other ways as well)

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



Before connecting the gates, we 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.



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

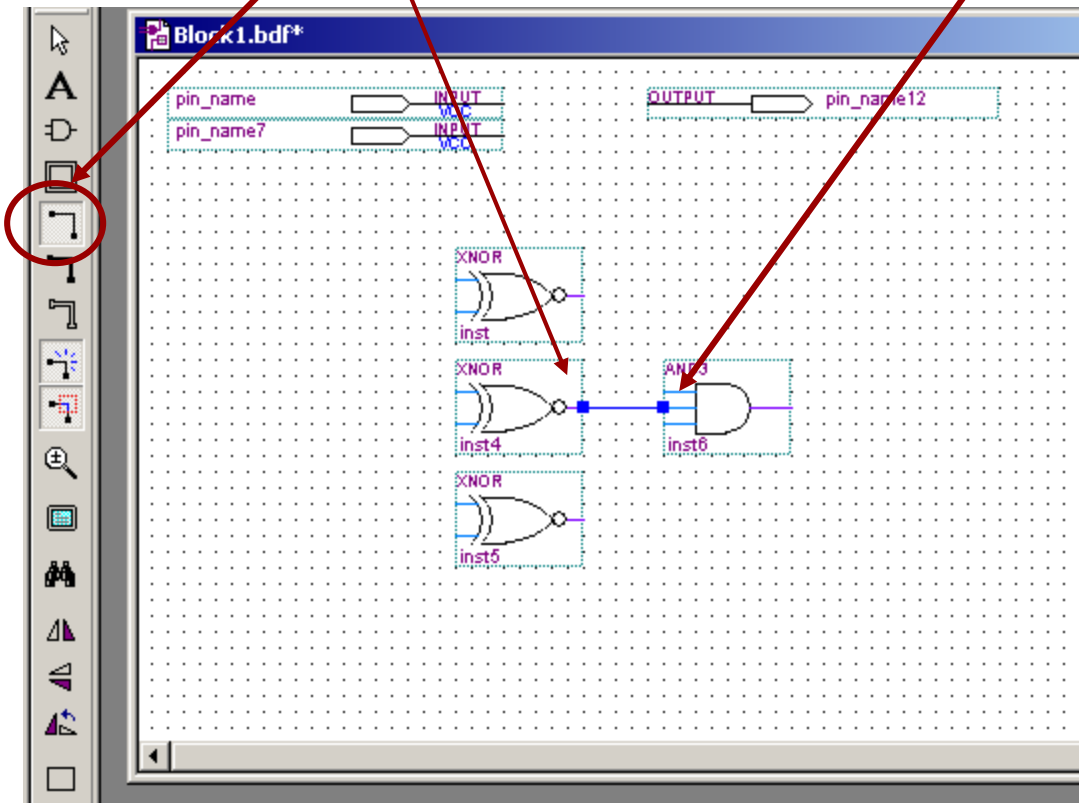
There are two ways that we can connect nodes in the schematic:

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

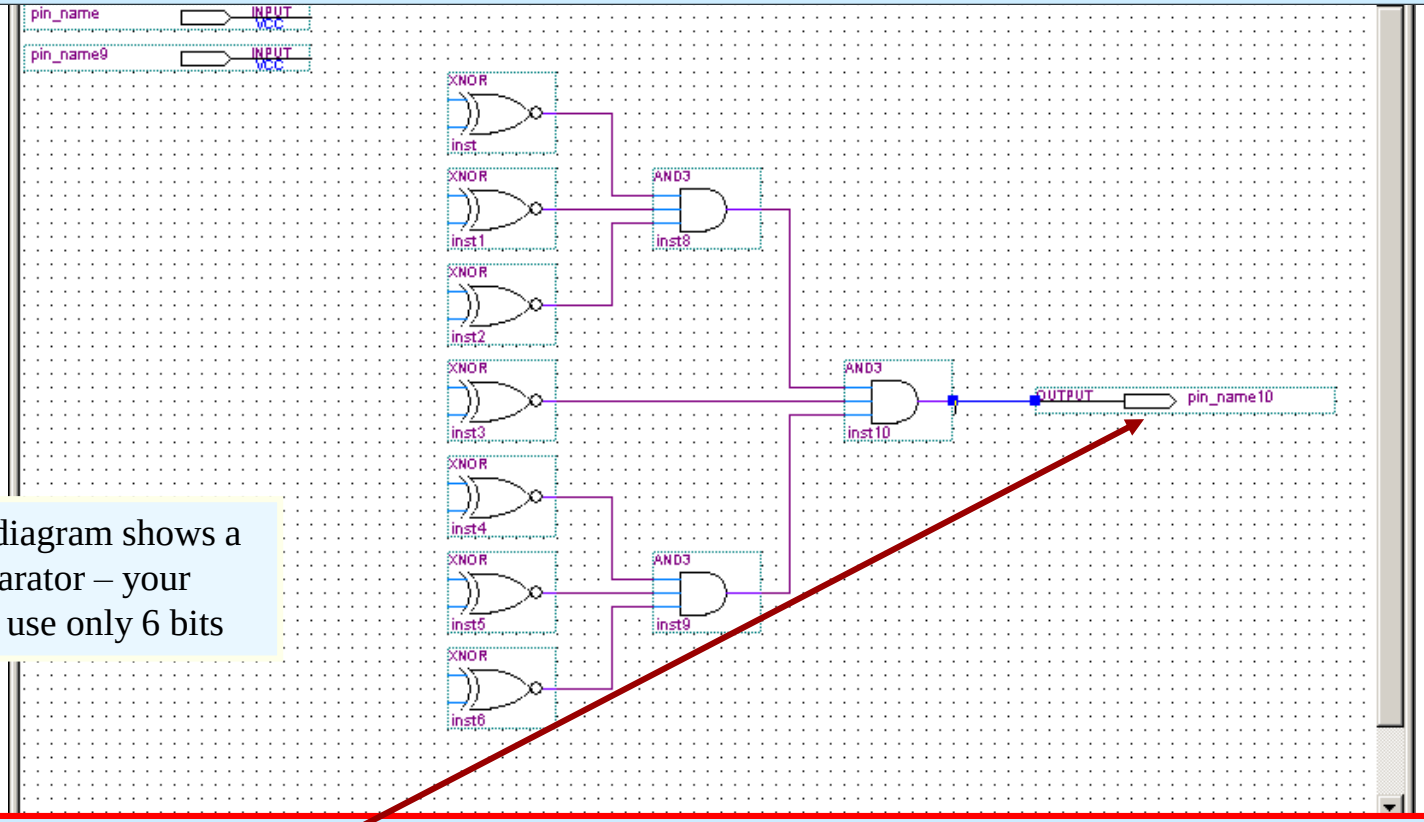
The first approach is best when making short connections between neighboring symbols, or when you want to make it immediately obvious from looking at the schematic that two nodes are connected.

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.

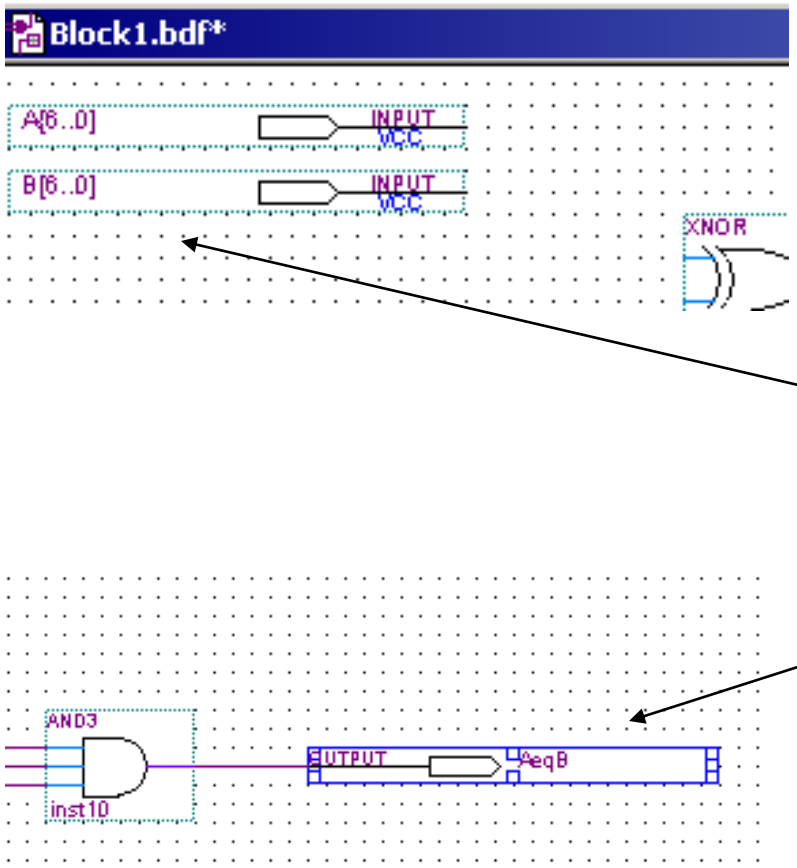


Practice drawing lots of wires between various places. You can delete a wire by selecting the "Selection and Smart Drawing Tool" (the arrow shaped icon), clicking on the wire to select it, and then pressing the Delete key. You can move symbols by selecting them (with the arrow tool) and dragging them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



Note: this diagram shows a 7-bit comparator – your circuit will use only 6 bits

Drag the output pin symbol to the right of the third 3-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.



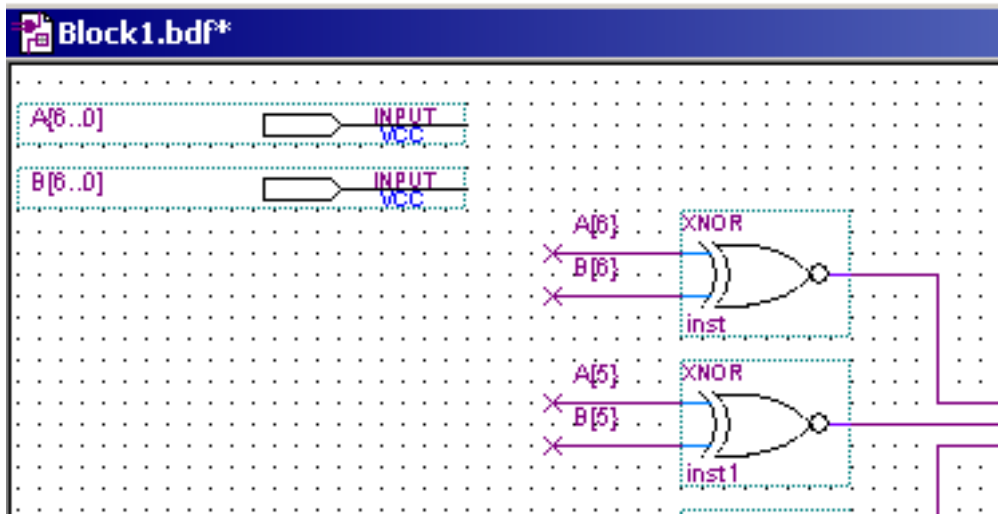
In order to use node naming as a connection technique, we must name the nodes we want connected. Let us begin by naming the input busses. To do this, double-click on the left-most part of one of the input symbol. This should highlight the input name (this can be a bit tricky, so get the TA to help if 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[6..0] and B[6..0]. Also name the output as AeqB.

The name A[6..0] means that A is a bus with 7 wires, having indices of 6,5,4,…,0. The node named A[6] 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.

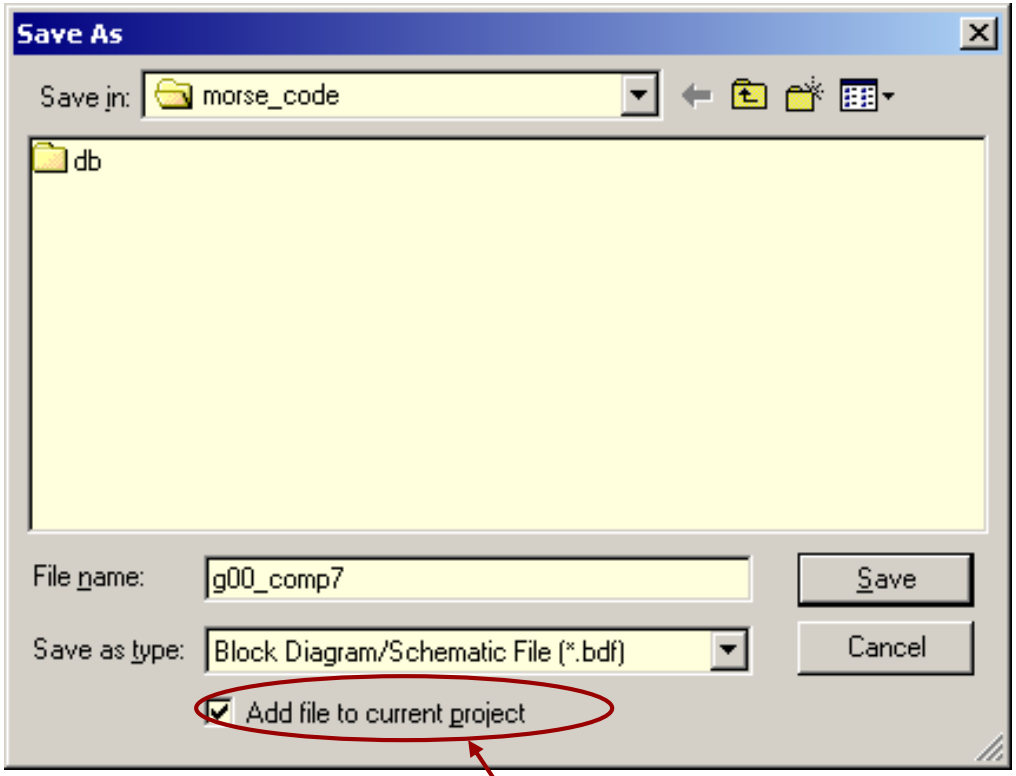
Since we have given the wire connected to the first input of the top XNOR gate the name A[6], it will automatically be connected to the MSB of the input symbol named A[6..0]



This completes the entry of the schematic for the g00_comp7 circuit. ***Show your schematic to the TA and have him or her sign your grade sheet.***

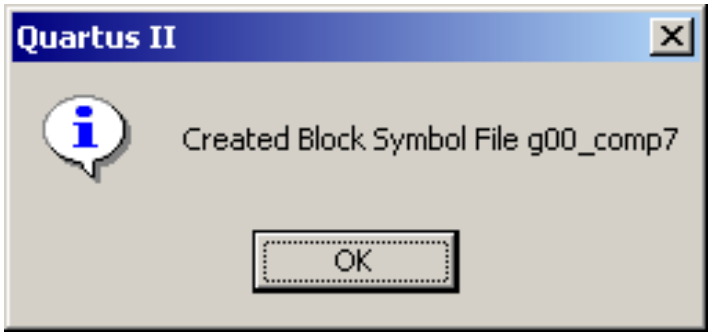
You should now save the design, using the "***Save As***" item in the ***File*** menu.
Actually, you can do this at any time, and it is a good habit to save regularly - just in case.

Call your file "***gNN_comp7.bdf*** " where ***NN*** is replaced by your group number.



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

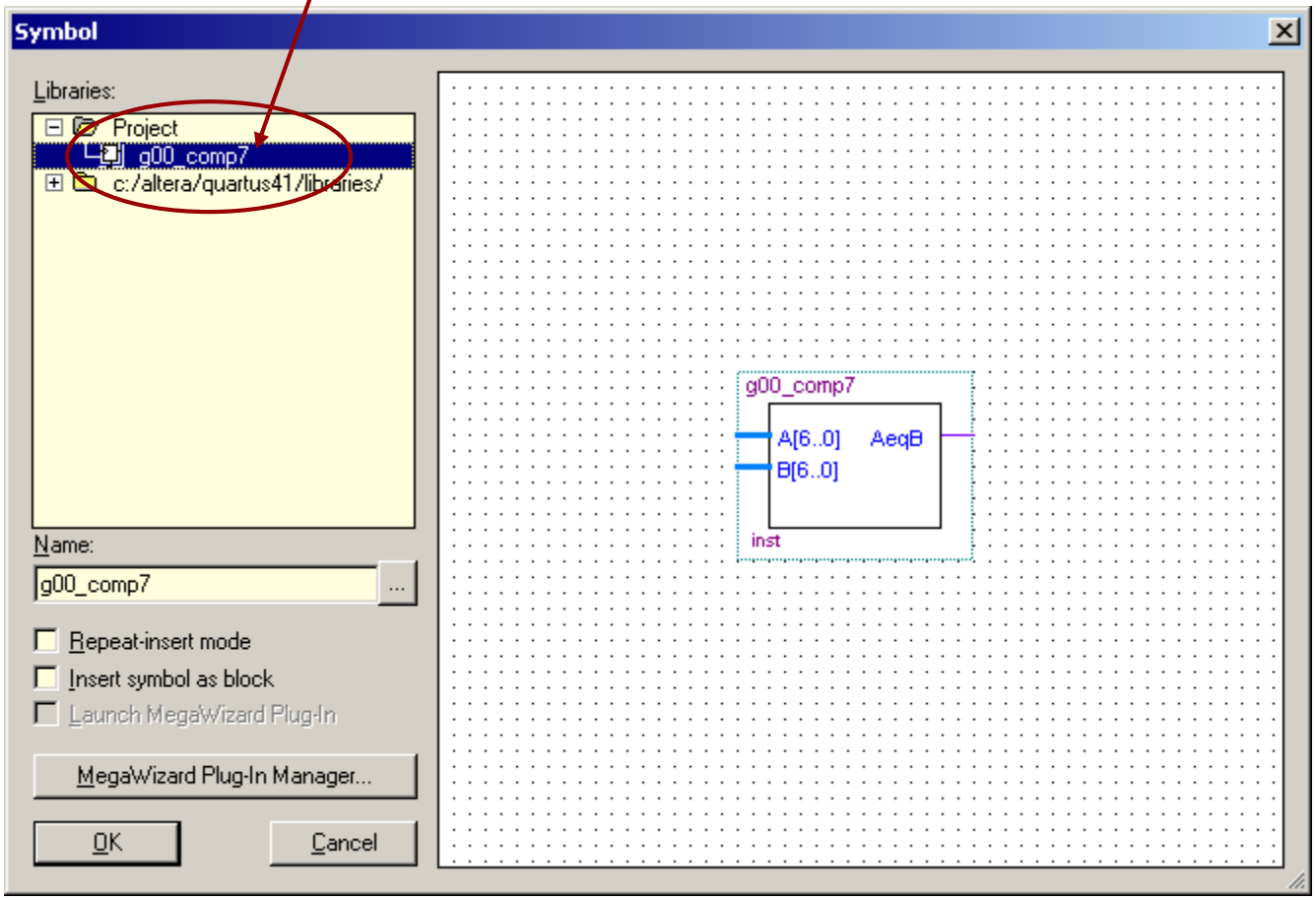
So that the circuit gNN_comp7 can be used as a component in other circuits, we need to create a ***symbol*** for it. This is done by selecting the "***Create/Update***" menu item in the "***File***" menu, and then selecting the "***Create Symbol Files for Current File***" menu item. This will produce the following message if everything was OK:



TIME CHECK

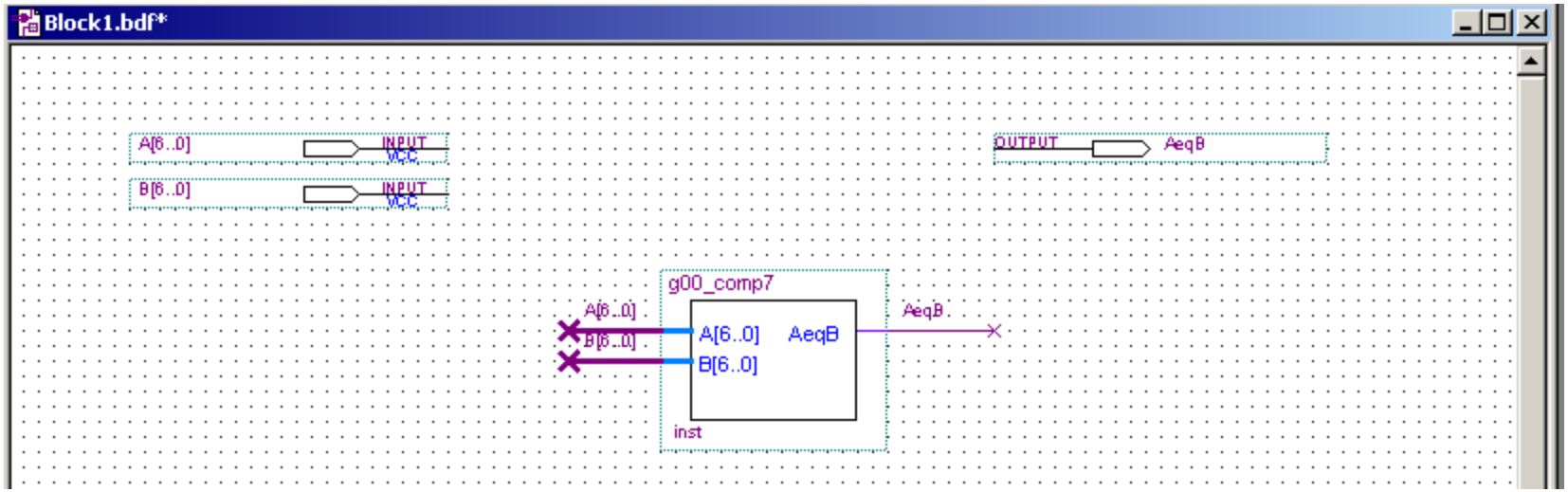
You should be at least this far at the end of your *first* 2-hour lab period.

Open a new block diagram file, and insert an instance of your gNN_comp7 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).



4. Performing Functional Simulation of the Project

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

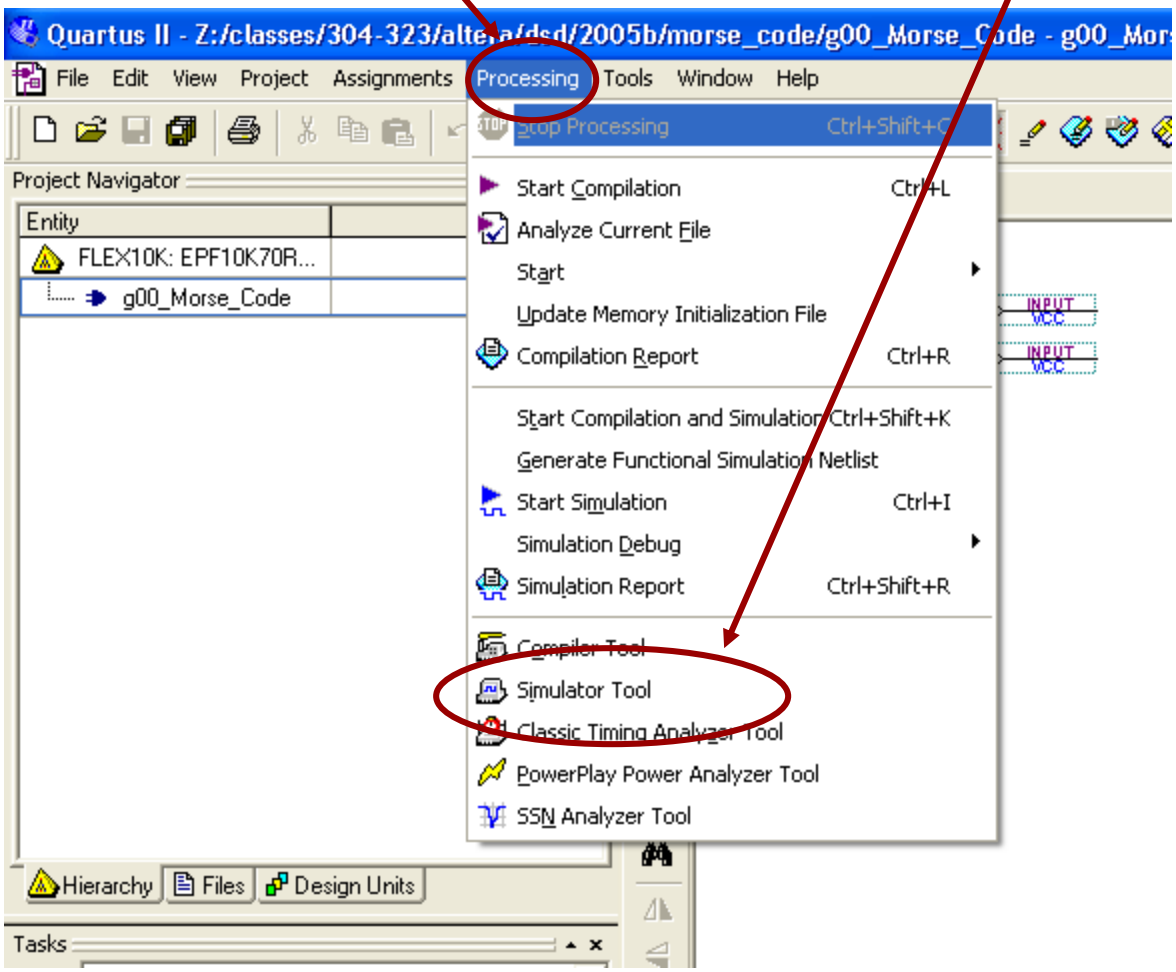
The purpose of simulation is generally two-fold:

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

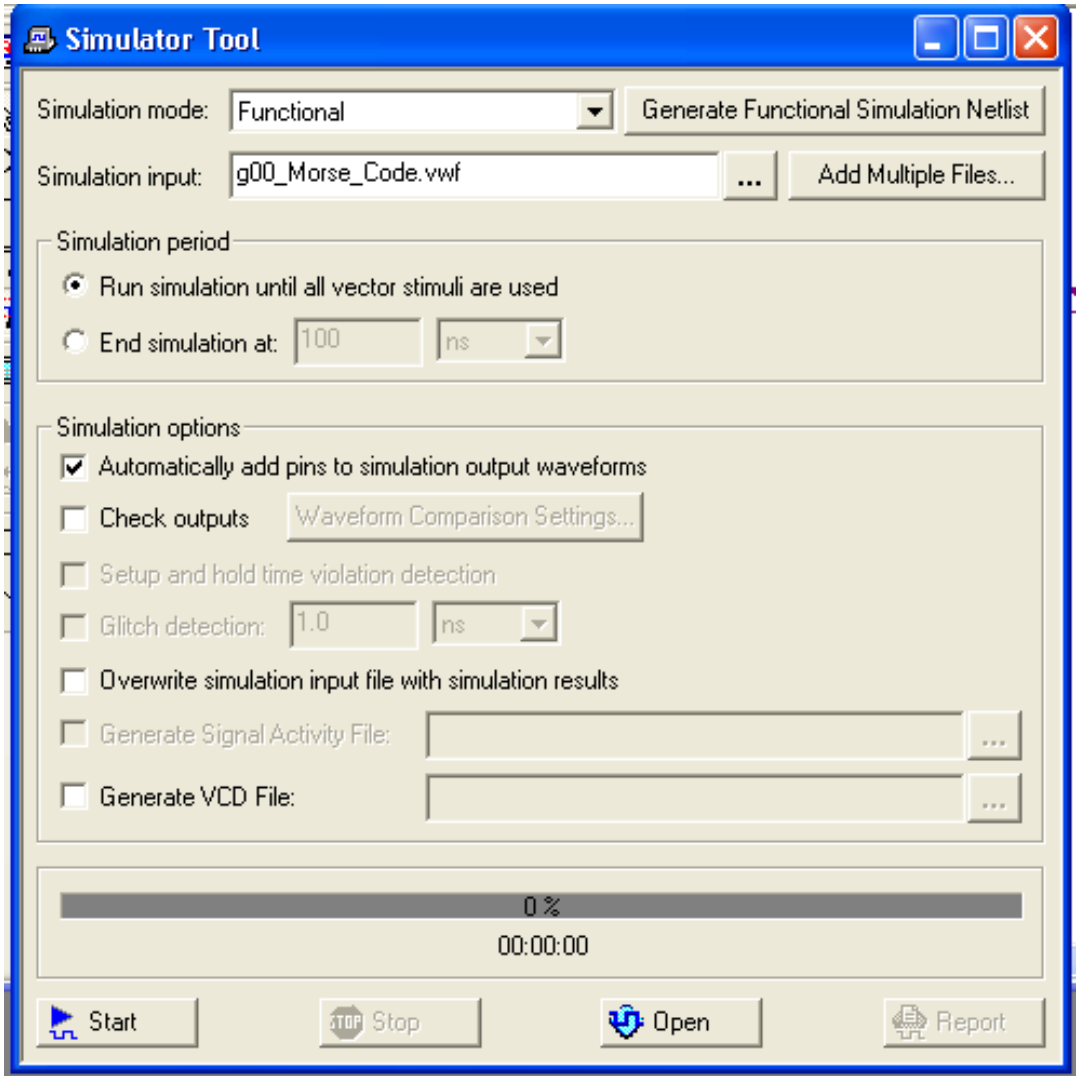
In the first case, we are only interested in the functionality of our implementation. We do not care about propagation delays and other timing issues. Because of this, we do not have to map our design to a target hardware. This type of simulation is called ***functional simulation.*** *This is the type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

The other form of simulation is called timing simulation, or hardware 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.

To begin setting up to do a functional simulation, select the "***Simulator Tool***" item in the "***Processing***" menu.



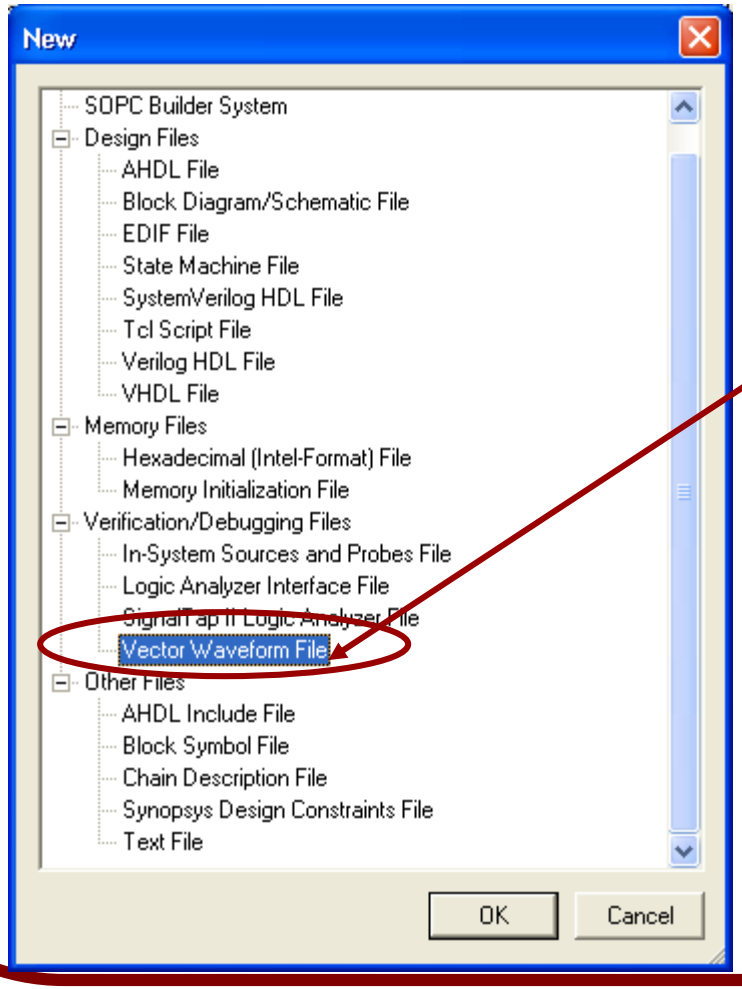
The "***Simulator Tool***" window will appear…



Click on the "***Generate Functional Simulation Netlist***" button in the simulator tool window. This will analyze the circuitry in the project and extract a set of logical equations which describe the functionality of the circuit.

When running a simulation it is useful to have a means of setting the inputs to certain patterns, and of observing the outputs' responses to these inputs.

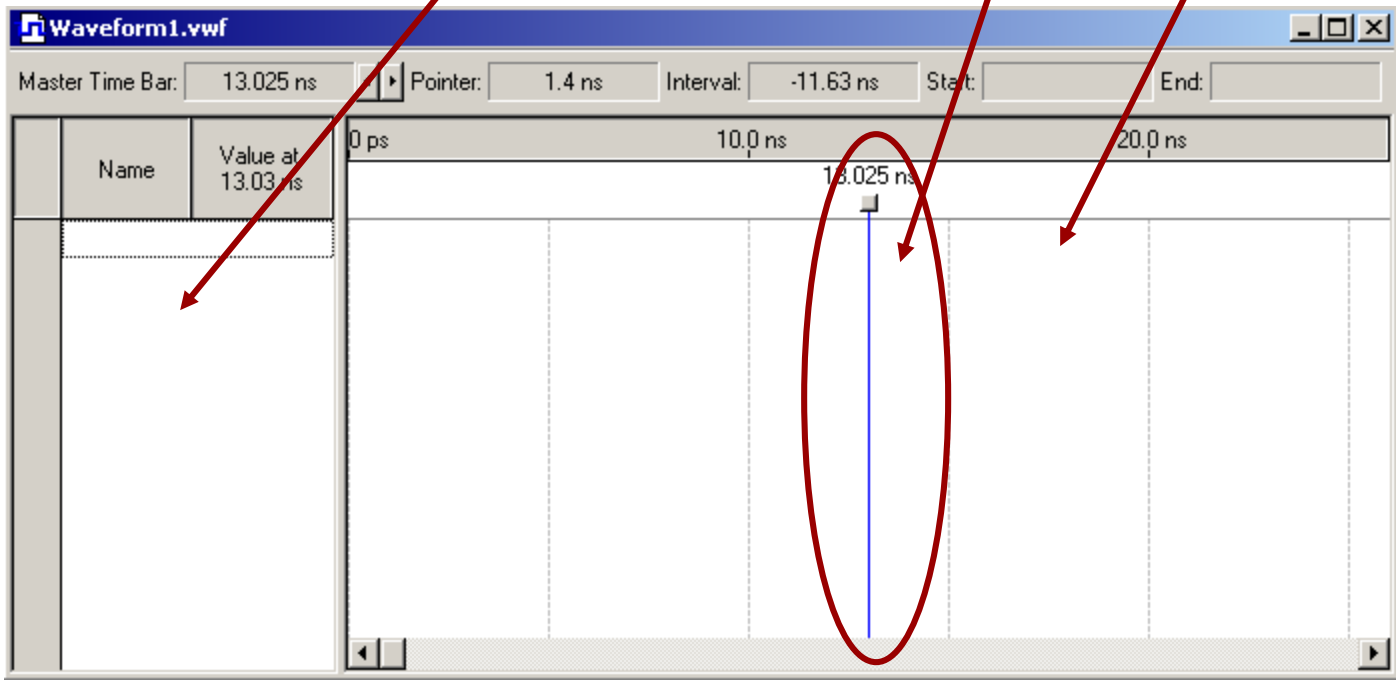
In Quartus, these tasks are done through graphical display of a "***Vector Waveform File***" (.vwf). To create a new vwf file, select the "***New***" command from the "***File***" menu.



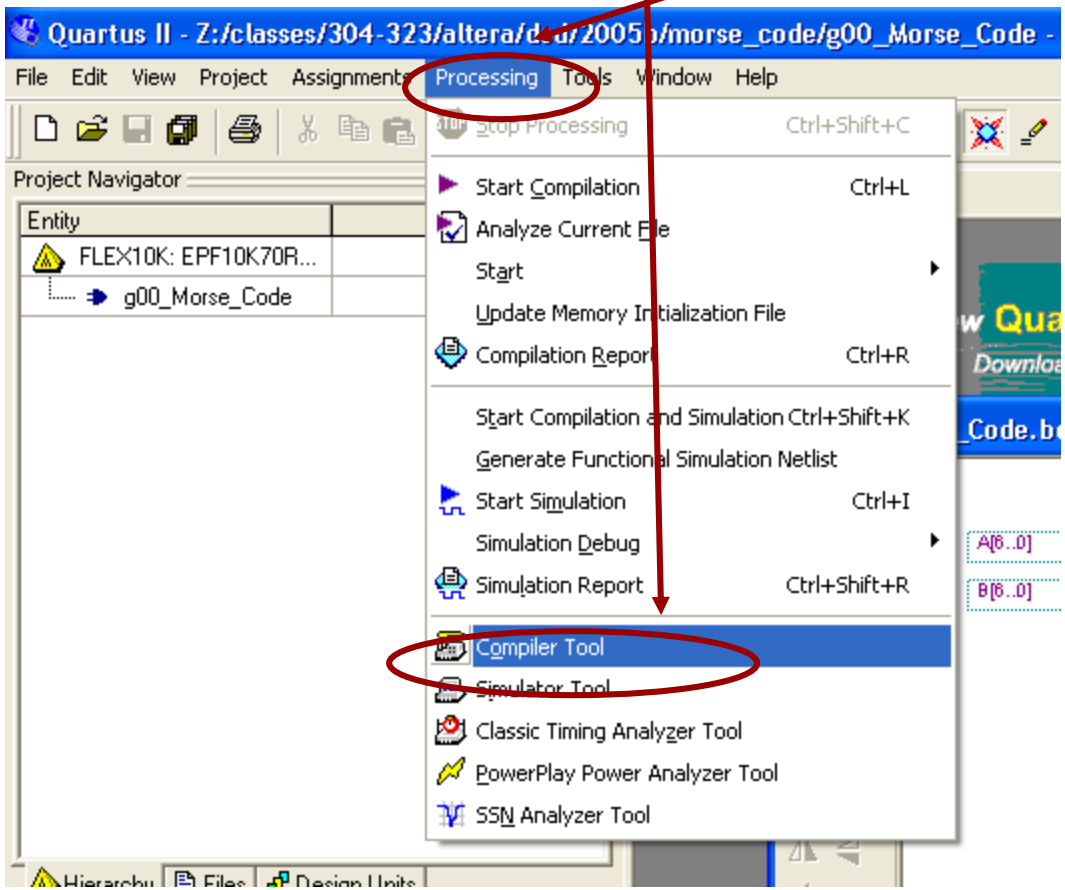
Select the "Vector Waveform File" entry and click on OK.

An empty waveform editor window will appear, as shown below.
It contains two primary display panes - the left hand pane shows the name of the nodes or busses being displayed, and their value at the time indicated by the time-bar, and the right hand pane shows the waveform - which is the node value as a function of time.

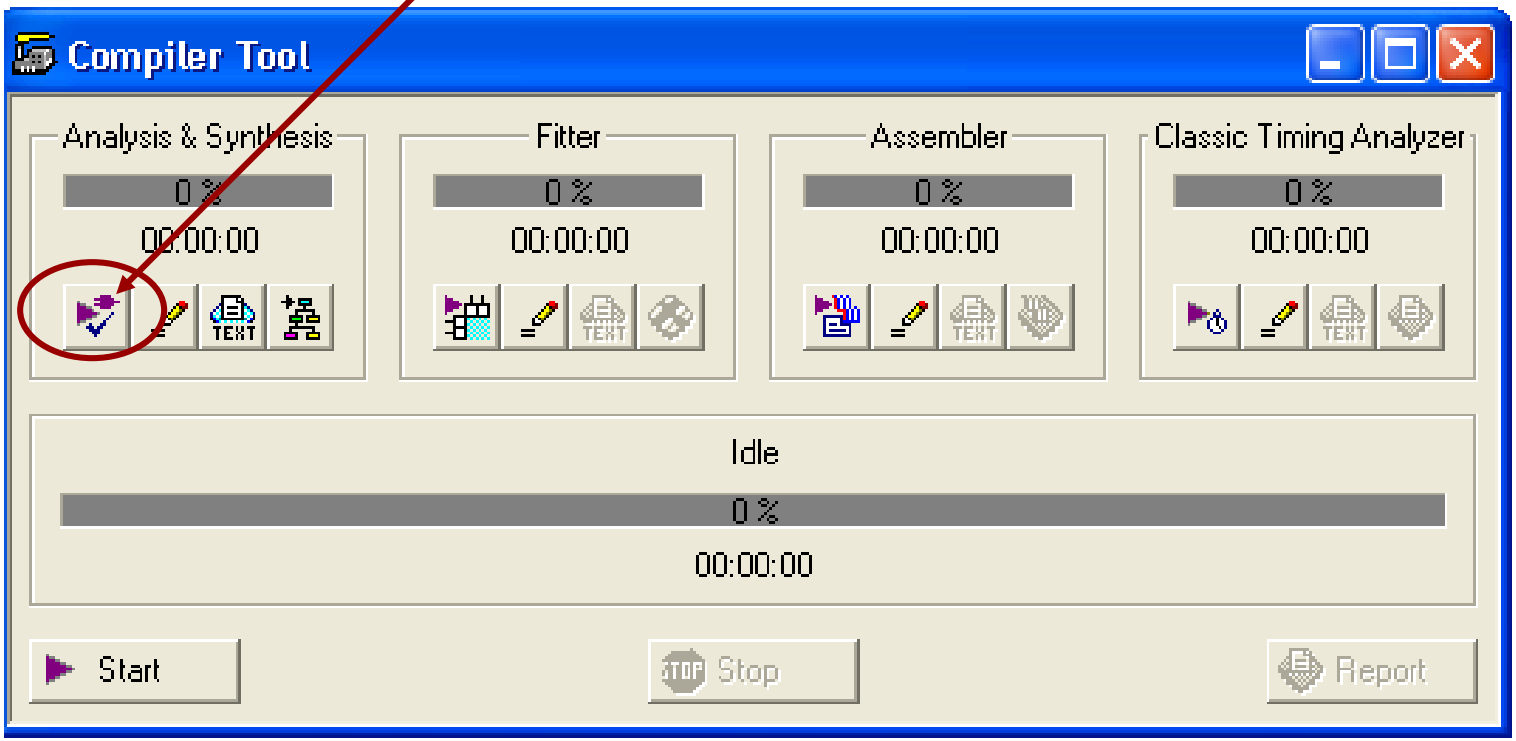
When the VWF is first created, the waveform window will be empty. So, the next thing we must do is specify the nodes and busses that are to be displayed…



Before specifying the nodes to be displayed in the simulator, we must do a ***Compilation*** of the design. To do this, select "***Compiler Tool***" from the "***Tools***" menu.

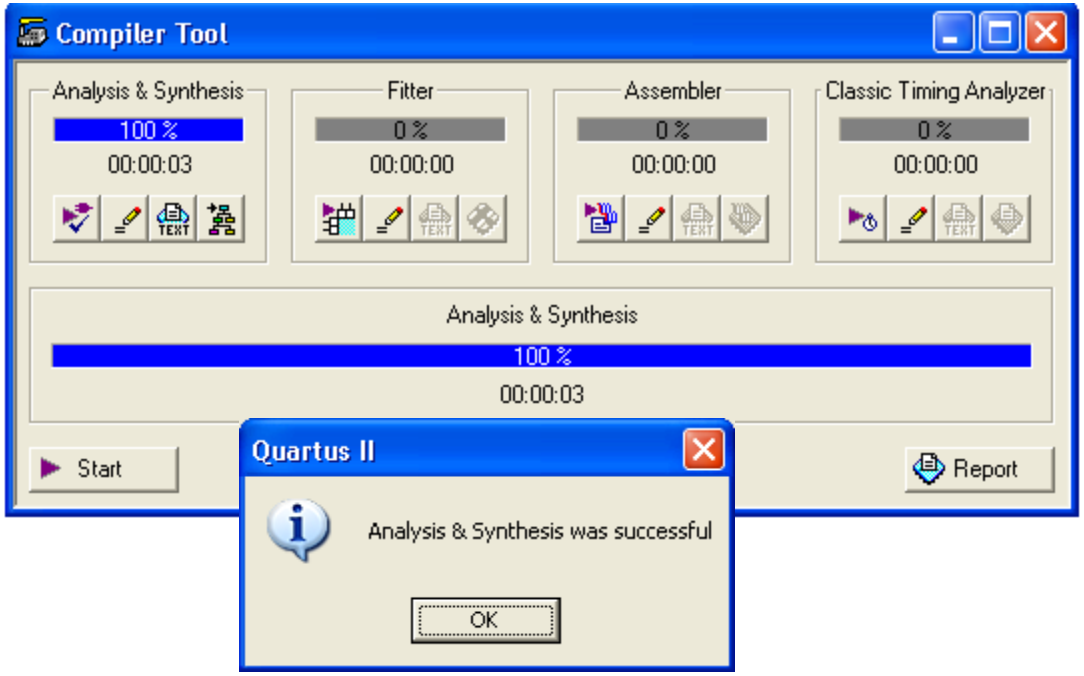


The compiler window will appear. In order to do a functional simulation we do not need to do a complete compilation. We only need to do the first part, Analysis & Synthesis. To run the Analysis/Synthesis, click on the left-most icon in the Analysis & Synthesis block.



On completion the compiler window should look like that shown below.

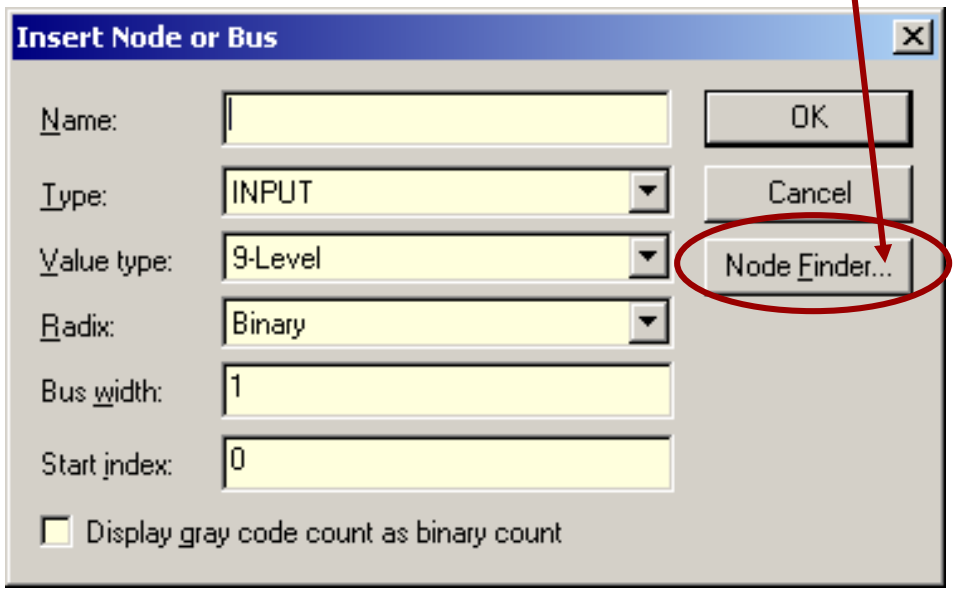
You can now go back to the simulator window and insert some nodes.



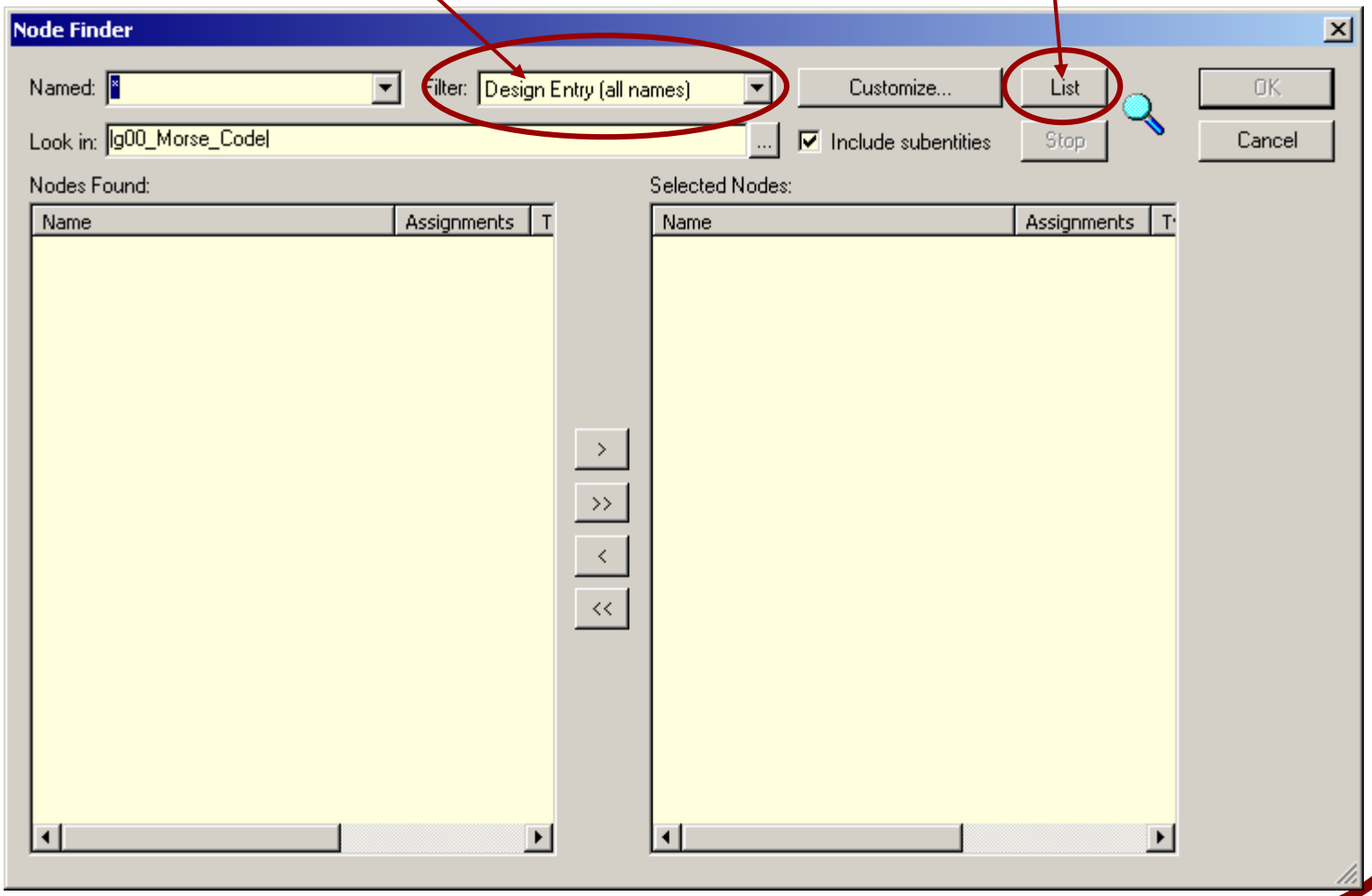
To insert some nodes and busses into the waveform editor window, position the mouse cursor over the left part of the Waveform editor window (the part which says "Name" and "Value"). ***Right***-click the mouse, and select the "Insert Node or Bus…" menu item. A dialog window will popup as shown below.

If you know the name of the node that you want to add then you can type it in to this window. But at the beginning, when you have a lot of nodes to enter, or when you don't know the name of the node, it is better to use the "***Node Finder***".

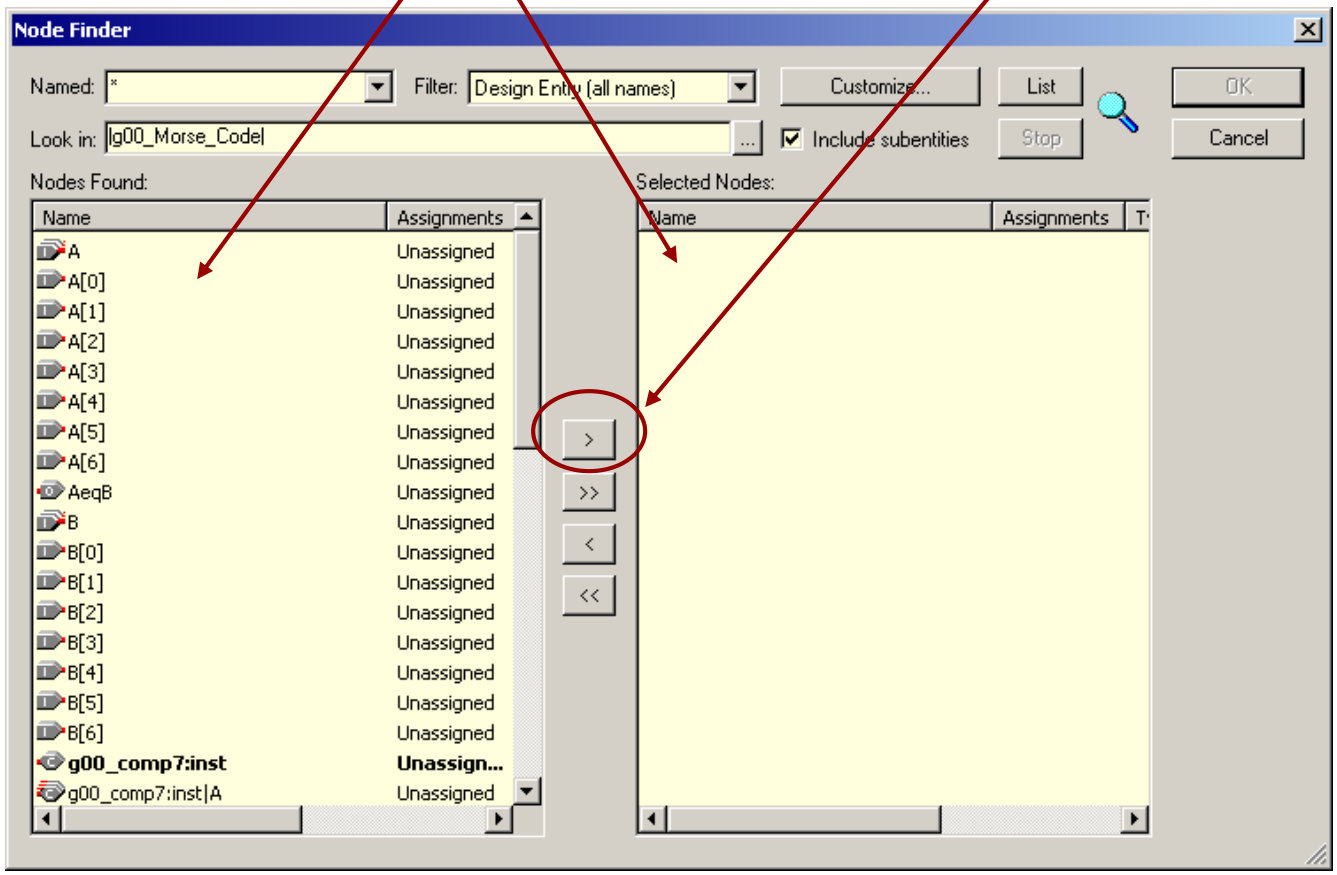
To use the node finder, click on the Node Finder button…



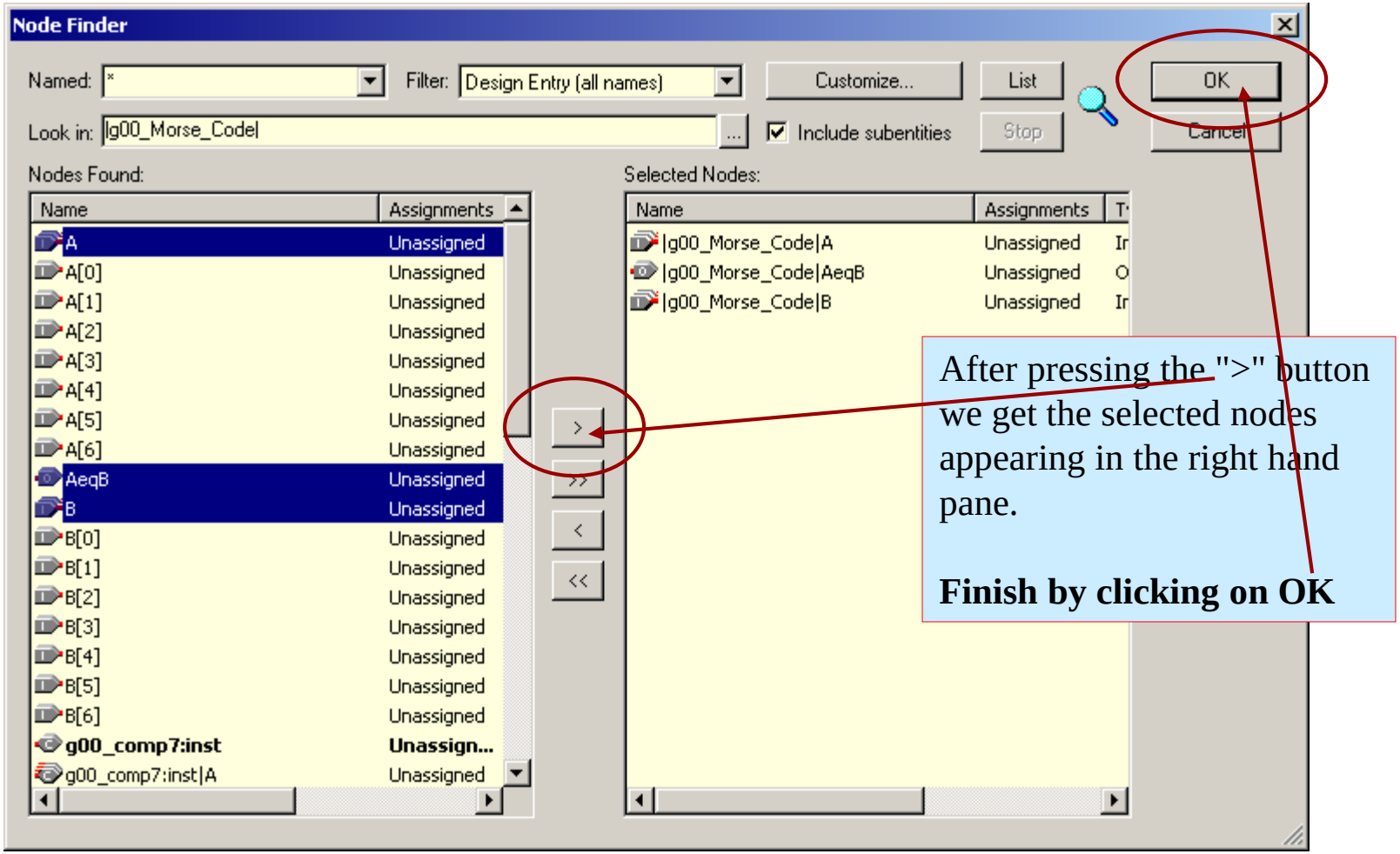
When you click on the Node Finder button, the following window appears. Make sure that the Filter: is set to "***Design Entity (all names)***", and click on the "***List***" button.



When you click on the List button, the "***Nodes Found***" pane on the left side of the Node Finder window is filled with all of the nodes in the design entity. You can now select individual nodes to be shown in the simulation. Selection of multiple nodes is done in the same way as selecting multiple files in the Windows File Explorer (i.e. ctrl-click to select). Once you have all the files you want selected, click on the ">" button to transfer these to the "***Selected Nodes***" pane.

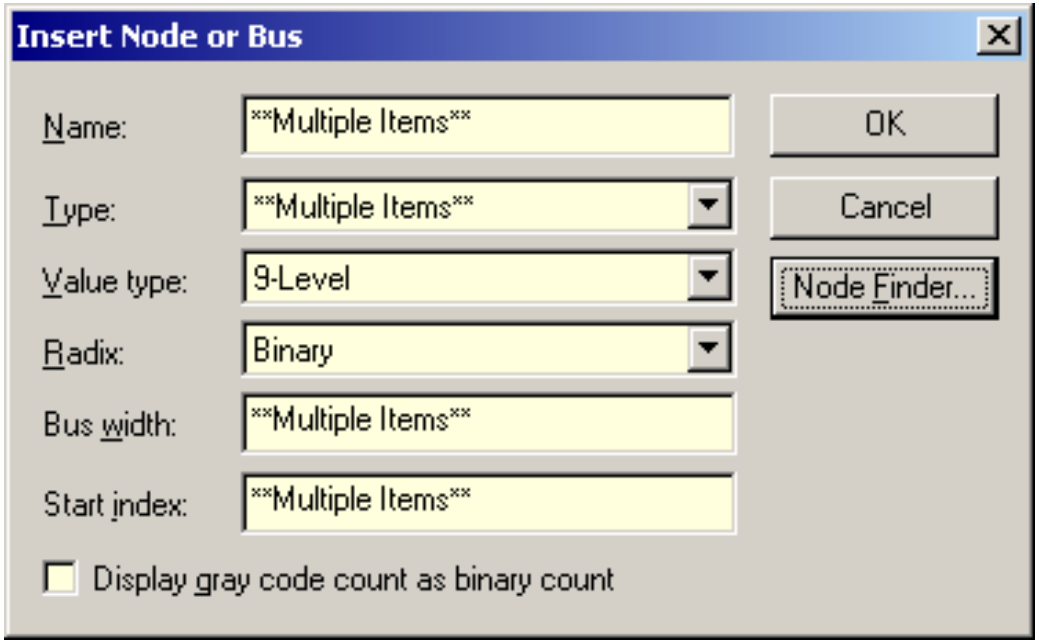


For this case, the only nodes we care about during simulation are the input busses A and B, and the output node AeqB. We could have also selected the individual bus nodes, A[0], A[1], A[2], etc., but it is more convenient to treat them as a whole bus.



Returning from the Node Finder dialog, the "Insert Node or Bus" window looks as shown in the figure below.

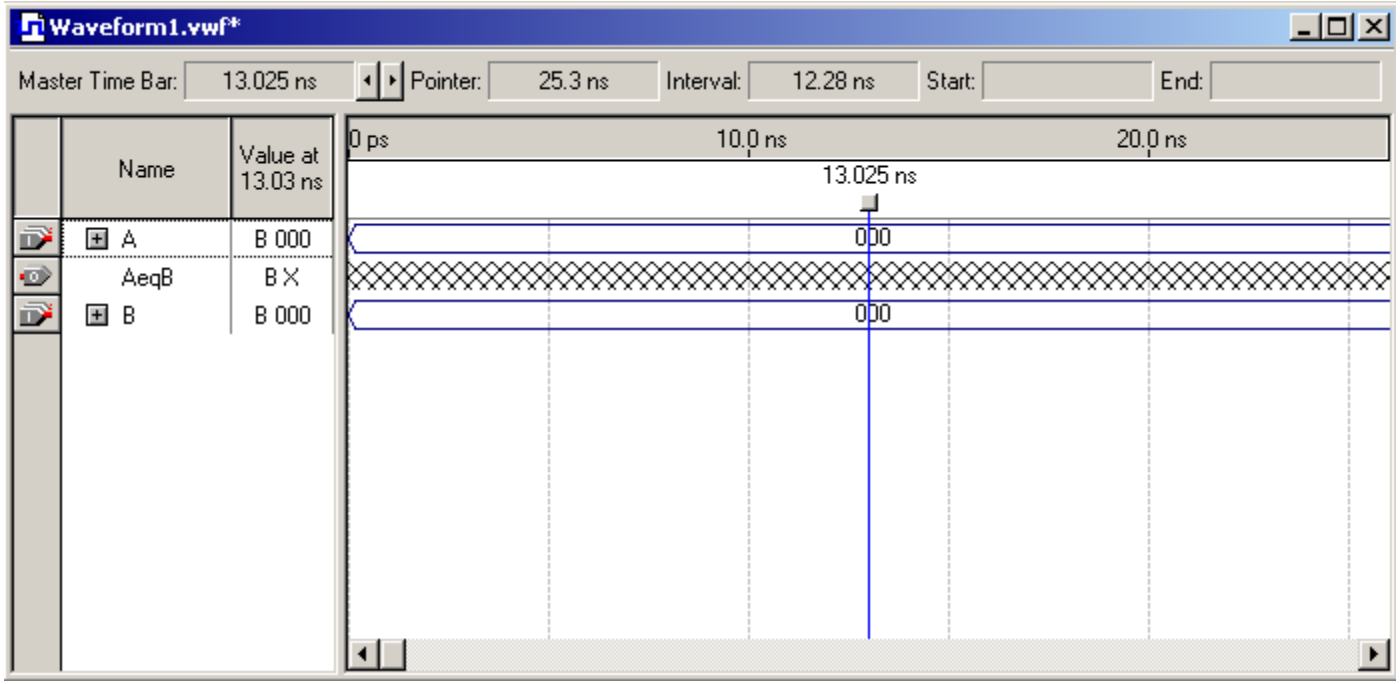
Click on OK to return to the Simulator window.



After insertion of the selected nodes, the Simulator window now looks as in the figure shown below. Note that A and B are both set to zero and the value of AeqB is X, indicating an undefined value (as is expected, since we have yet to run the simulation).

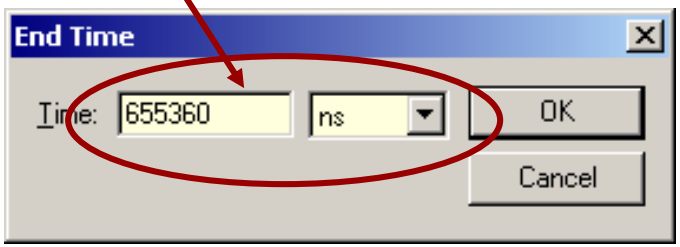
The next step is to specify test input waveforms for the input signals A and B.

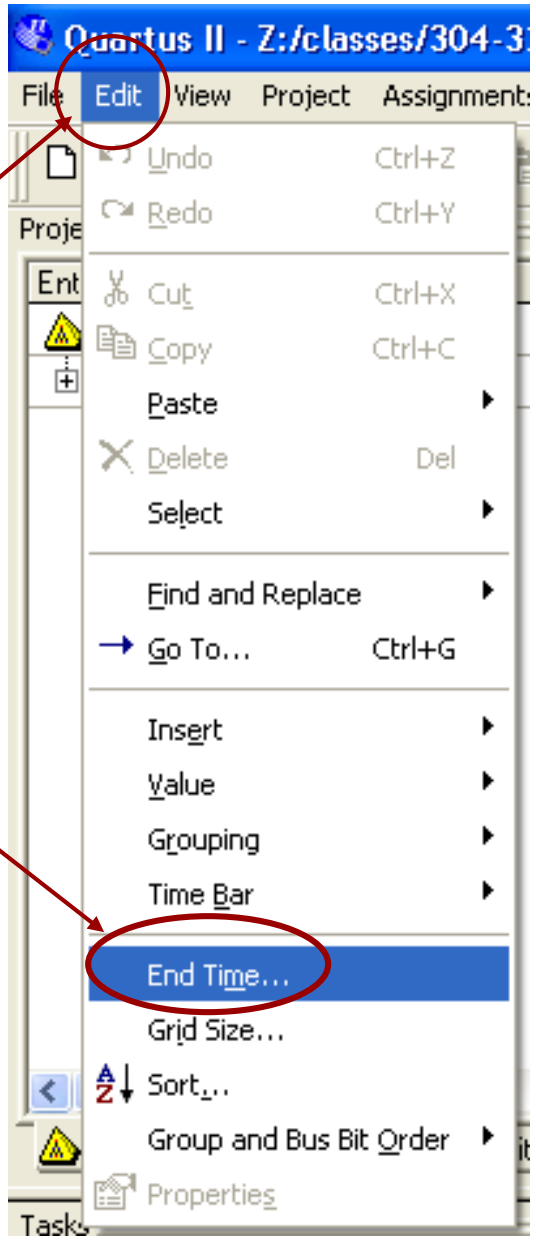
There are 2^14 possible input patterns (16,384), which is a lot to test, but the computer is doing all of the work.



First, set the end time for the simulation, by selecting the "***End Time***" item in the ***Edit*** menu. A window will popup, allowing you to specify the time at which the simulation will end.

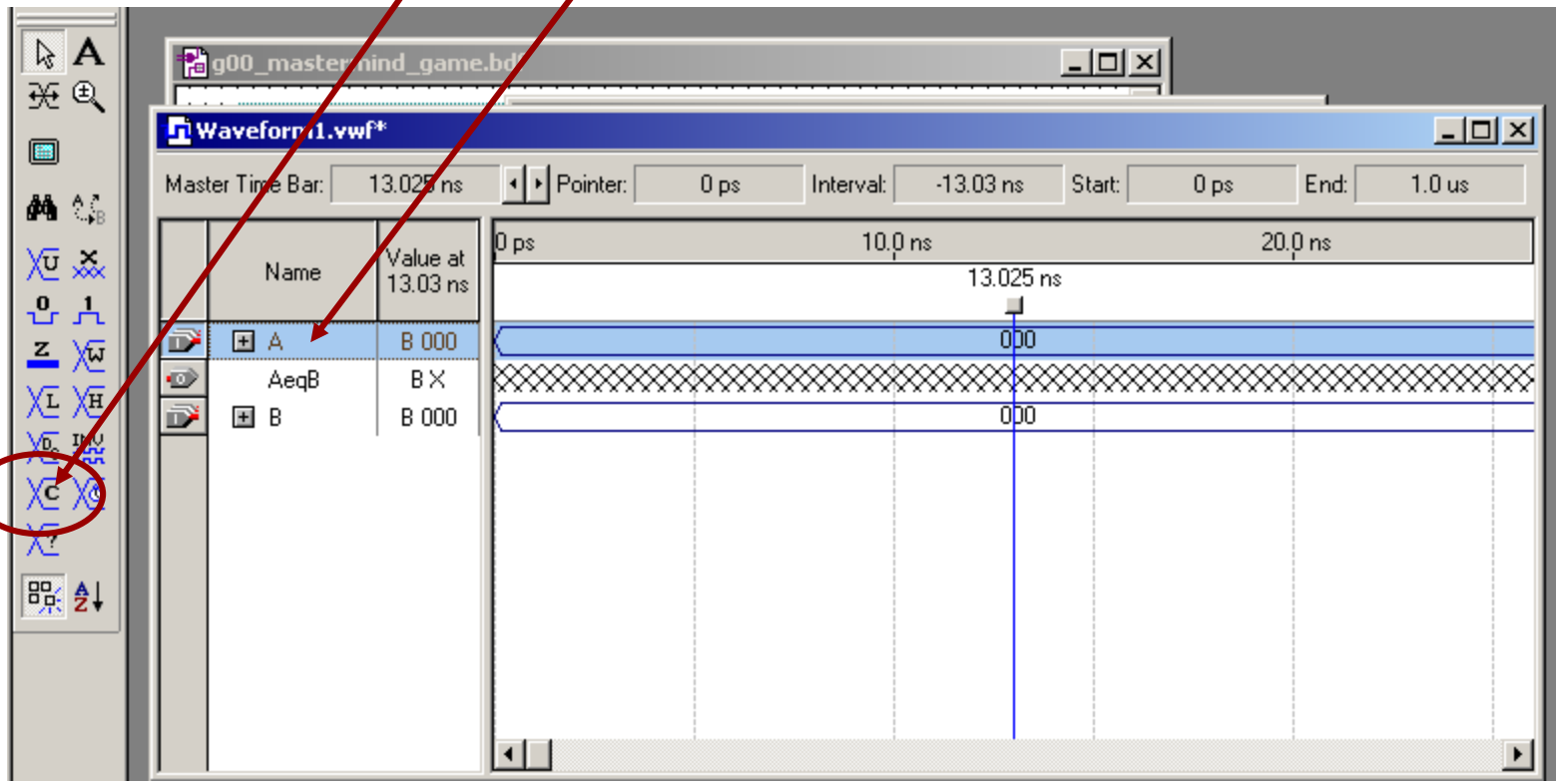
Set the time to 655360 ns 655360 ns (which is 2^14 x 40ns).



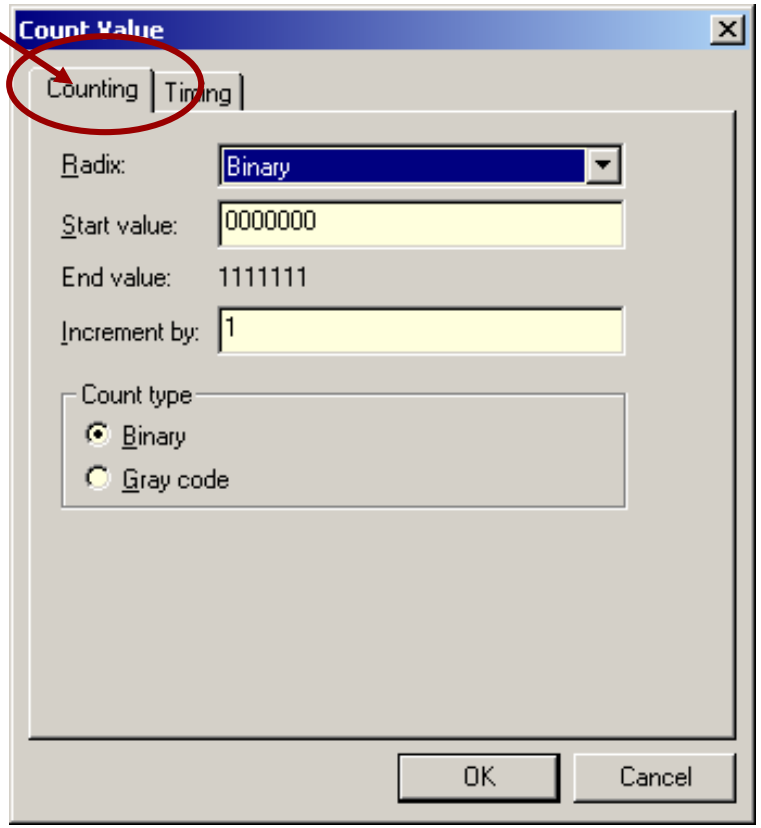


Next, select bus A by left-clicking on its name in the left-hand pane. When it is selected, the entire row will be highlighted.

Now, click on the "Count Value" button in the left-hand vertical toolbar. This is the button with the "C" inside of a pair of crossed lines.
This will allow you to fill the waveform for A with a binary count sequence.

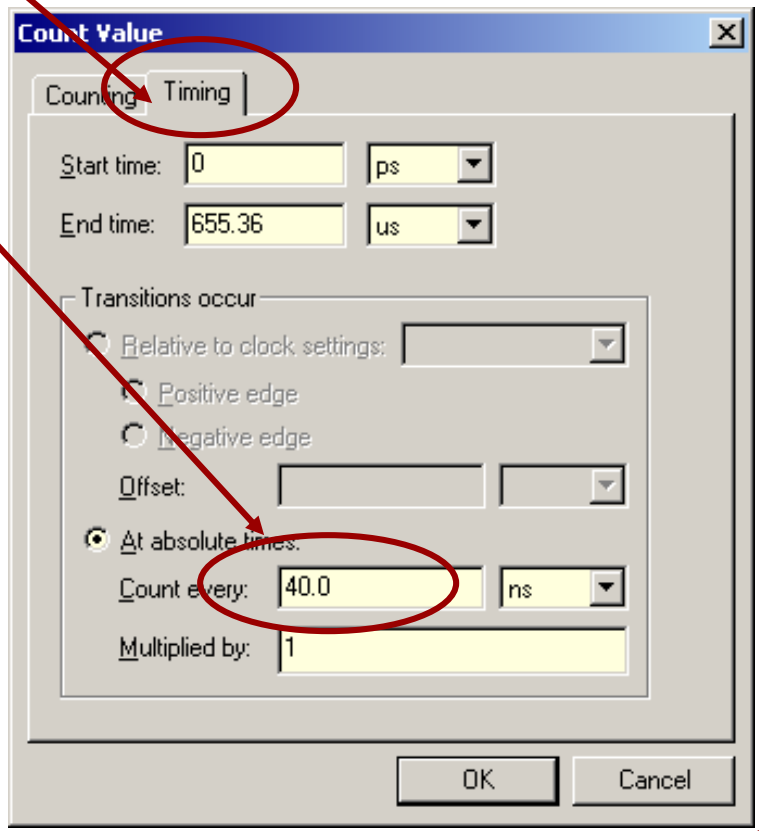


Clicking on the "Count Value" toolbar button brings up a dialog window with two tabs. Select the "Counting" tab. The window shown in the figure below will appear. Set the "Start value" to 0000000 and the "Increment by" value to 1 as shown (these are the default values).

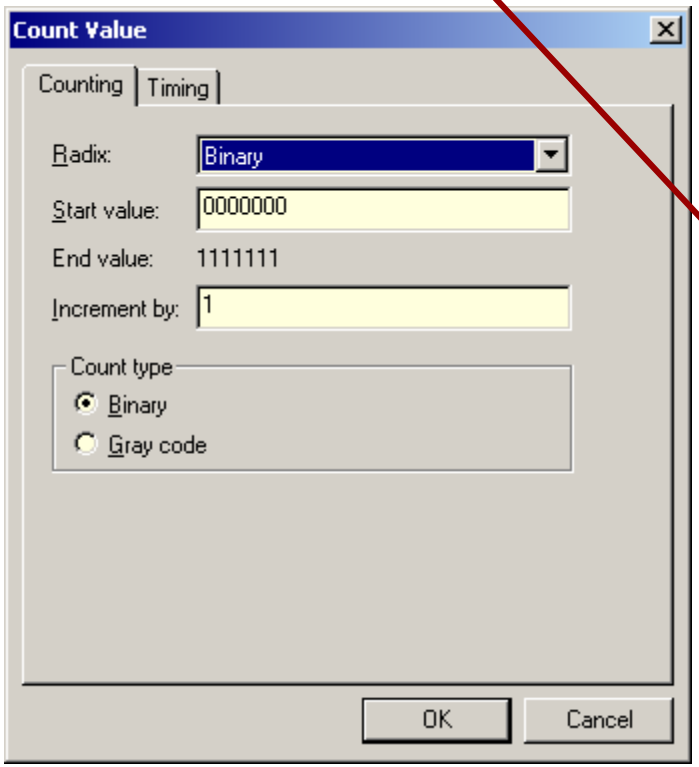


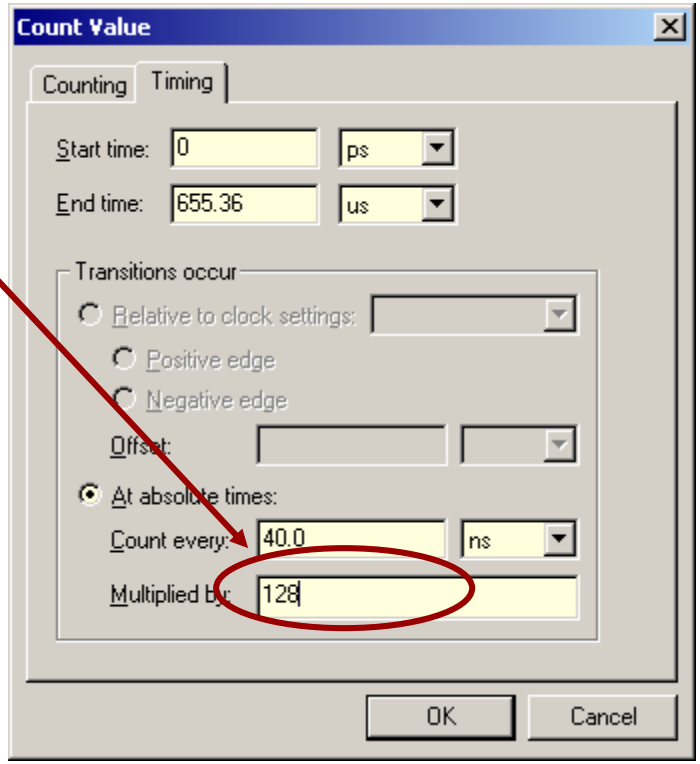
Then select the "Timing" tab. The window shown below will appear. Set the "Count every" setting to 40.0 ns as shown. This will create a waveform which changes every 40 ns.

Click on OK to finish.

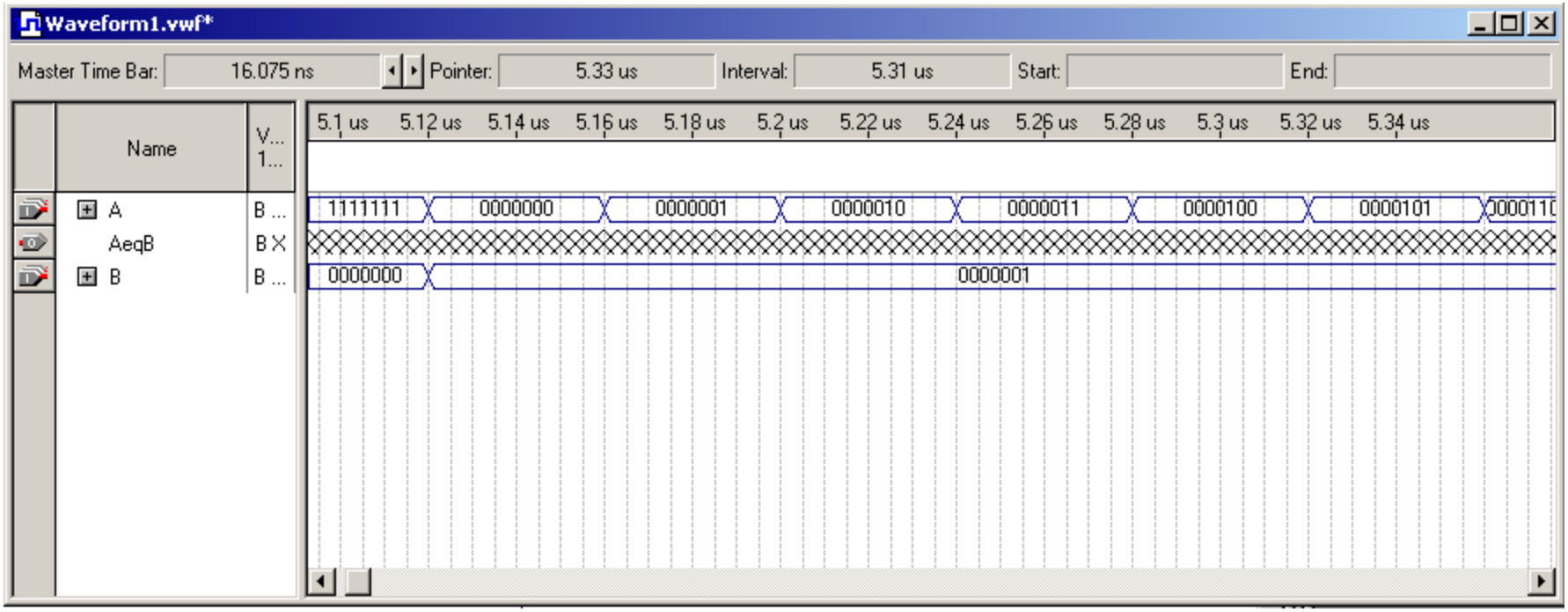


Repeat for bus B. The settings should be the same, except that in the Timing tab view, set the "***Multiplied by***" setting to ***128***. This means that the count sequence for B will change every 5120 ns, which is 128 (2^7) times slower than for A. In this way you get a sequence of the 2^14 possible different values for the combination of A and B.



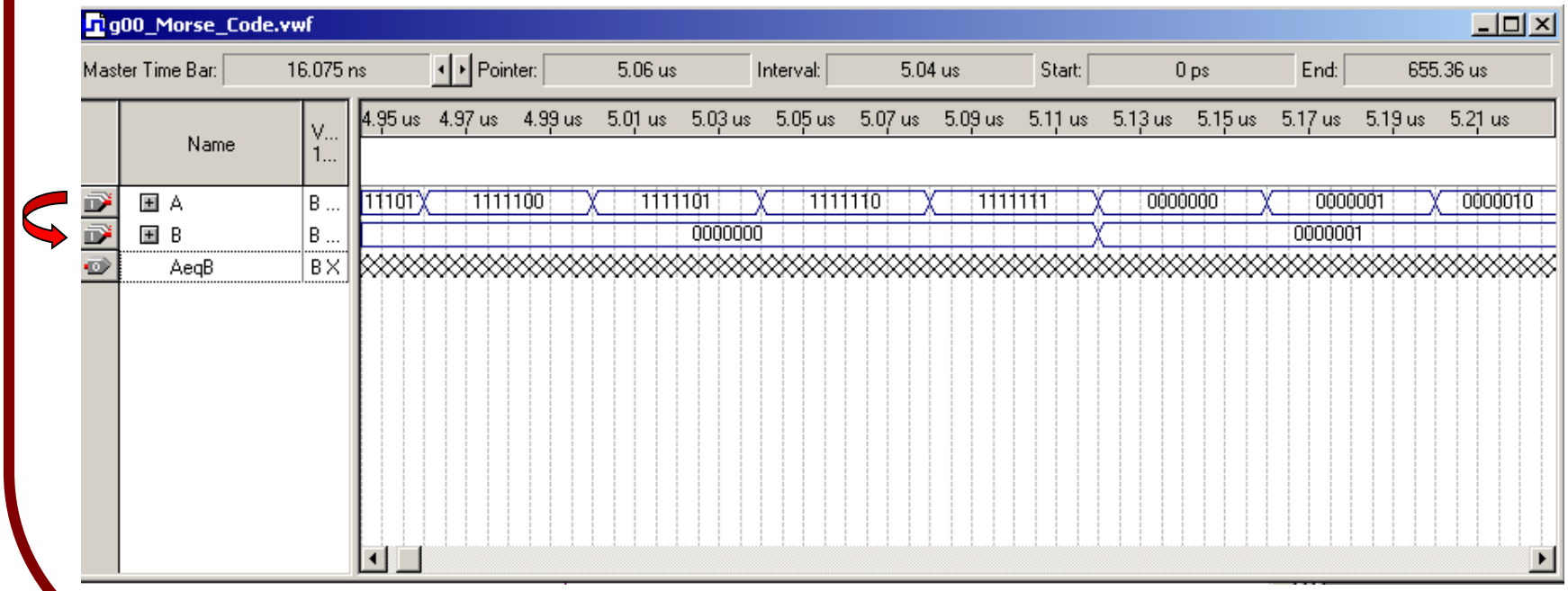


After specifying the count values, the waveforms for A and B will look similar to that shown in the figure below. Note that B changes more slowly than A, as desired, and that all 16,384 possible combinations of the values of A and B are produced.



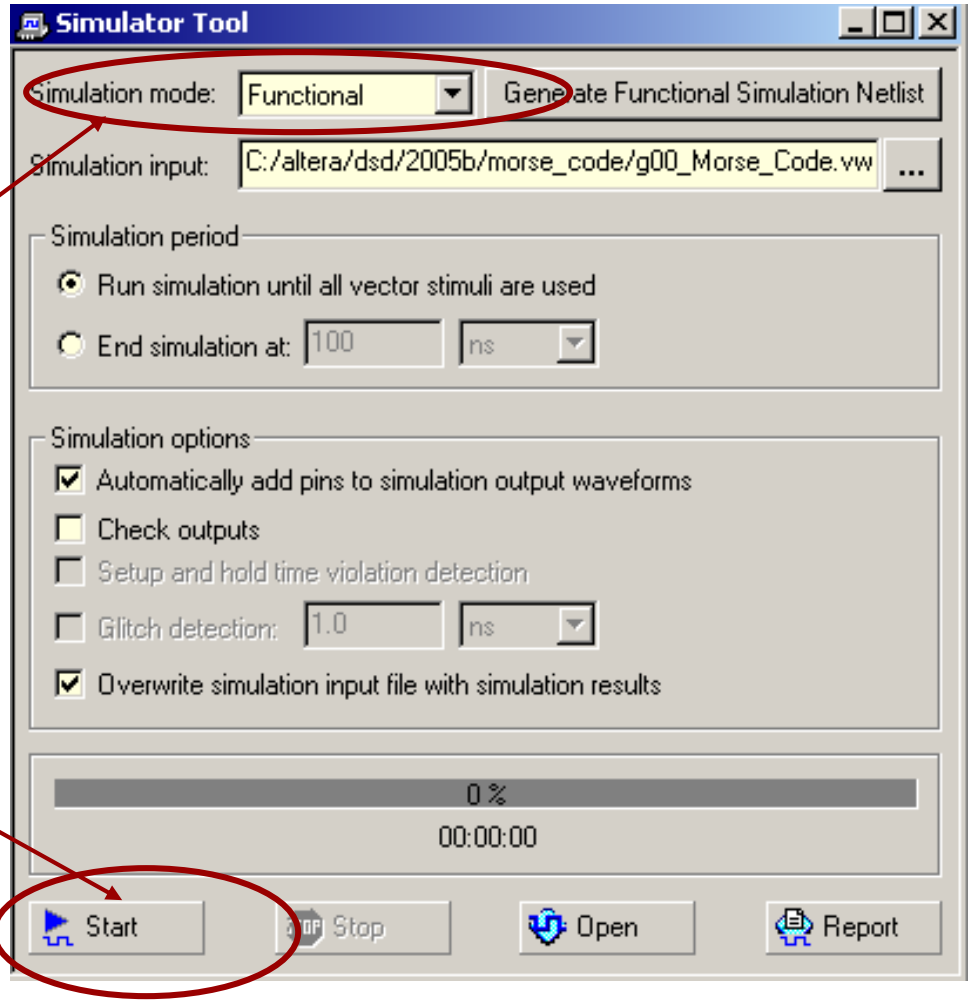
Select "Save As" from the file menu, and save the waveform file as "gNN_lab1.vwf".

We can rearrange the ordering of the waveforms displayed in the simulation window. This is done by selecting the waveform to be moved, then by left-clicking on it, and dragging it to the desired new position, while keeping the mouse button held down. In the diagram below, we have swapped the positions of waveforms B and AeqB, so that waveforms A and B are adjacent.

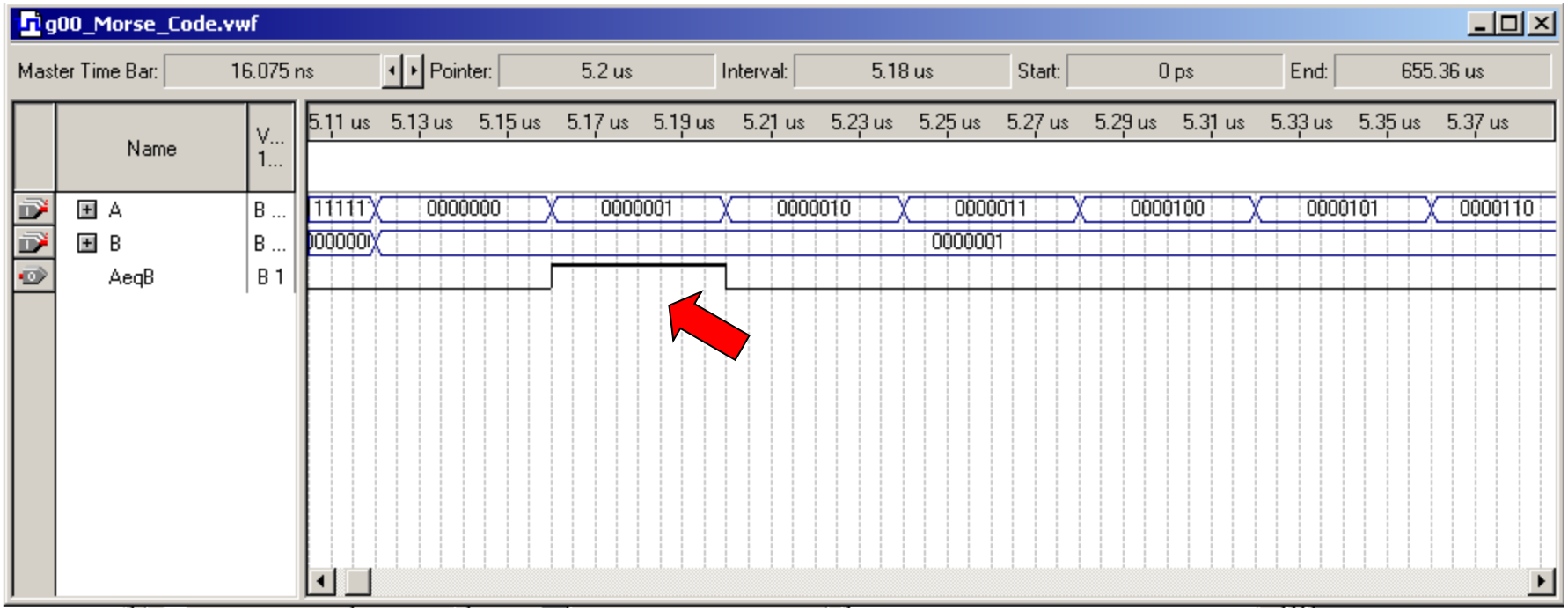


Once your waveforms have been setup, return to the Simulator Tool window. Make sure that the Simulation mode is set to "***Functional***", then click on the Start button to run the simulation.

(Note: you may have to re-run the "Generate Function Simulation Netlist" before starting the simulation)



After the simulation finishes, the output node AeqB, which was previously undefined, should now be filled in with properly defined values. Look at the values for AeqB and see that they are correct, given the inputs A and B (i.e. it should be high when A=B)



Show your simulation results to the TA and have him sign your grade sheet

TIME CHECK

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

5. Design of a Modulo-13 Circuit

For the course project you will need a circuit that will take in a 6-bit input, *A*, and provide a 4-bit output, *Amod13*, which represents the modulo, base 13, of the input.

The modulo 13 of A is given by the following formula:

$$Amod13(A) = A - floor(A/13)*13$$

where *floor(x)* is the largest integer that is less than or equal to *x*.

For example, if *A=37*, then *A/13=2.846…*, and *floor(A/13)=2*.
Thus *Amod13(37) = 37-2*13 =11*.

Your circuit should also have a 3-bit output *floor13*, which outputs *floor(A/13)*.

To implement the modulo circuit you should use a couple of useful tricks to simplify the computations.

Trick #1 - *circuits for implementing* ***division*** *operations tend to be large (many gates) and slow. Thus, you should avoid using division if possible, except for division by numbers which are powers of 2. Integer division by 2^N is performed simply by right-shifting N places.*

e.g. 203/8 = 11001011/1000 = shift_right(11001011,3) = 00011001 = 25

But we have to divide by 13, which is not a power of 2! What do we do???

Trick #2 – *instead of dividing by 13, we can multiply by (1/13). Then, approximate (1/13) by a ratio of integers, where the denominator is a power of 2. Use the smallest numbers that will give a good approximation. In this case we can approximate (1/13) by (5/64), (which uses the smallest possible numbers that yield an accuracy sufficient for our needs).*

So, to compute *floor(A/13)* we need only compute *shift_right(A*5,6).* Since the shifting operation gets rid of the fractional part of the division, the floor operation comes for free.

e.g. floor(37/13) = shift_right('100101' * '101',6) = shift_right('10111001',6) = '10' = 2

Once we have computed floor(A/13) we need to multiply by 13 and then subtract the product from A. So, we have to do two multiplications in total – once by 5 and once by 13. This leads us to our final trick for this lab.

Trick #3 *– Multipliers can be quite large circuits as well, and can often be simplified using a* ***shift-and-add*** *technique, especially when one of the multiplicands is a constant (as is the case here). For example, in computing A*5, we observe that 5 = '101' in binary, so that A*5 = A*4 + A*1. Multiplication by a power of 2, e.g. 2^N, can be done simply by shifting left N places. Thus we can implement A*5 by shifting A two places to the left and adding this to the unshifted version of A. Similarly for computing X*13, observe that 13 = '1101', so we can implement this by shifting X three places to the left, and adding it to X shifted two places to the left and then to the unshifted version of X.*

So, we have seen how to implement the modulo-13 (and floor(A/13)) operation without actually using any division or multiplication circuits. All we need are some adder circuits and some shifting left and right (which is just a matter of connecting the correct bits to the adder inputs).

Your first order of business should be to design an adder circuit that you can use in your implementation of the modulo circuit. Just make one adder module, with the number of input and output bits determined by the maximum that you will need (not all adders will need to have the same number of bits, but to simplify your design you can assume that they are all the same size). Implement the adder using a ripple-carry design, such as that shown in figures 5.5 and 5.6 of the textbook (on page 257).

Once you have the correct design for the adder module sketched out, fire up the Quartus program. Using the techniques you learned in the first part of the lab, create a new bdf file into which you will draw the schematic diagram for the circuit. Call your circuit block ***gNN_adder*** where NN is your group number.

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

Please note that you will also need to perform a ***subtraction*** in implementing the modulo-13 operation. You can use your adder module for this, but the input for the *subtrahend* (the number that is being subtracted) must be in ***2's-complement*** form.

See figure 5.13 on page 266 of the textbook for detail on the circuitry needed to form the 2's-complement.

6. Functional simulation of the adder circuit

Once you have the schematic of the adder module entered into Quartus, create a symbol for it. Then carry out a functional simulation of the circuit, using the same approach used earlier to simulate the 7-bit comparator circuit. Test all possible input cases, and display all output bits.

After simulation, verify that your circuit works properly for all input patterns that you presented. If the circuit gives erroneous results, you will have to go back to your schematic and check for errors.

(note: There may be some brief glitches near the input transitions. You can ignore these and just focus on the values attained after settling down.)

When you have completed the simulation, show the resulting vector waveform display to your TA and have him or her sign your grade sheet.

7. Completion of the Modulo-13 Circuit

Once you have the adder module functioning properly, and the design for the modulo-13 module sketched out, create a new .bdf file and draw the schematic diagram for the complete Modulo-13 module.

Call your circuit block ***gNN_Modulo_13***.

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

TIME CHECK

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

8. Functional simulation of the Modulo-13 circuit

Once you have the schematic of the Modulo_13 circuit entered into Quartus, create a symbol for it. Then carry out a functional simulation of the circuit, using the same approach used earlier to simulate the 7-bit comparator circuit. Test all 64 possible input cases, and display all 7 output bits.

After simulation, verify that your circuit works properly for all input patterns that you presented. If the circuit gives erroneous results, you will have to go back to your schematic and check for errors.

(note: There may be some brief glitches near the input transitions. You can ignore these and just focus on the values attained after settling down.)

When you have completed the simulation, show the resulting vector waveform display to your TA and have him or her sign your grade sheet.

TIME CHECK

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

NOTE: If you finish ahead of time start work on Lab 2!

9. Writeup the Lab Report .

Write up a short report describing the ***gNN_Modulo_13*** circuit that you designed in this lab. This report should be done in html or pdf, or in Microsoft Word (***pdf format is preferred!***).

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_Modulo_13***) 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 (Submission details to follow).

10. Submit the Lab Report to WebCT .

The lab report, and all associated design 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 2017.

Group Number:_____.

Group Member Name:_____. Student Number:_____.

Group Member Name:_____. Student Number:_____.

Marks			TA Signatures
	1.	Schematic diagram for the 7-bit comparator	_____.
	2.	Waveform Vector File display for the 7-bit comparator	_____.
	3.	Schematic diagram for the adder circuit	_____.
	4.	Waveform Vector File display for the adder circuit	_____.
	5.	Schematic diagram for the Modulo-13 circuit	_____.
	6.	Waveform Vector File display for the Modulo-13 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.