Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

LabL

Objective:

Using comparators (1-bit, 2-bit, 8-bit), this lab covers the following:

1. Introduction to VHDL, Design Comparator, and test it.
2. Introduction to ModelSim for circuit simulation (Learn How to use Model-Sim to simulate comparator)
3. Testing of circuits using a test bench file
4. Your task: Assignment for you to perform once completed the design, simulation and testing part of this lab tutorial. It also includes an outline for you to use in your lab report.

VHDL is intended for describing and modeling a digital system at various levels and is an extremely complex language. Then, we will use the design tool ModelSim to design, and verify our VHDL designs. Modelsim is widely used in industry

NOTE: In this lab you will design 1-bit, 2-bit and 8-bit comparators; however, we will only show you how to do the simulation. The 2-bit and 8-bit comparators follow the exact same procedure and are left for you to execute.

Additional Tasks:

1. **Optimize the 1 bit comparator VHDL code shown on page 3, to replace lines 12,13,14 with ONE Boolean operation!**
2. Optimize other comparators accordingly.
3. Design and test using Model_Sim a 8 bit comparator in VHDL.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

1. INTRODUCTION TO VHDL

VHDL stands for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. It is a language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. For VHDL primer please refer to this link

http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html

Basic VHDL Example:

1-BIT COMPARATOR

One-bit Comparator Design

A one bit comparator is a circuit that can be used to compare two one-bit signals. The comparator outputs a '1' if the input signals are equal; otherwise, the comparator outputs a '0'.

As introduction to this course, we use a simple comparator to illustrate the skeleton of a VHDL program. The description uses only logical operators and represents a gate-level combinational circuit, which is composed of simple logic gates.

Input		Output
i0	i1	Eq
0	0	1
0	1	0
1	0	0
1	1	1

Figure 5. Truth table of a 1-bit comparator

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

The truth table can be expressed by the following equation:

$$Eq = i0 . i1 + i0' . i1'$$

The entire code for the 1-bit comparator is shown below:

equal.vhd
```
 1 Library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity equal is
 5        port ( I0, I1 : in std_logic;
 6                Eq    : out std_logic);
 7 end equal;
 8
 9 architecture arch of equal is
10        signal p0, p1 : std_logic;
11        begin
12        EQ <= p0 or p1;
13        p0 <= (not I0) and (not I1);
14        p1 <= I0 and I1;
15 end arch;
```

As we said before, VHDL is case insensitive, which means that upper and lowercase letters can be used interchangeably, and free formatting, which means that spaces and blank lines can be inserted freely.

Lines 1 and 2 tell the compiler (in our case Quartus and ModelSim) which libraries to use. The libraries contain precompiled VHDL code. For example ieee.std_logic_1164.all contains the code for the 'or' function, and without it the VHDL compiler would generate and error on line 12.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

Lines 4 to 7 are the entity declaration statements. The entity declaration outlines the input and output signals of the circuit. The mode term can be in or out, which indicates that the corresponding signals flow "into" or "out of" of the circuit. It can also be ***inout***, for bidirectional signals.

Lines 9 to 15 are the architecture statements. The architecture body describes functions of the circuit. The architecture may include signals which we have used in line 10. We need the signals to store the value of first product and second product. You will understand signals more as you do more coding. The main description, encompassed between **begin** and **end**, contains three concurrent statements. Unlike a program in C language, in which the statements are executed sequentially, concurrent statements are like circuit parts that operate in parallel. The signal on the left-hand side of a statement can be considered as the output of that part, and the expression specifies the circuit function and corresponding input signals.

Notice that to translate our equation into VHDL code, we represented the term *i0 . i1* as *p0*, and the term *i0' . i1'* as term **p1**. Then we say ***Eq = p0 + p1***. We can, of course, represent the equation in just one line of VHDL code; however, as our codes grow larger, it is better to break our assignments to make the code more readable and modular. This notion will become apparent as you work on bigger projects.

==NOTE: Just like we have an *Eq* output, we can also have a *notEq* output that notifies when the inputs being compared are not equal. This is left for you as an exercise after you complete the tutorial part of this lab. See the YOUR TASK section towards the end of this lab for further explanation.==

2. INTRODUCTION TO MODELSIM

2.1 ModelSim installation:

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL and Verilog. Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

For this part you must have ModelSim installed in your computer. Altera gives you the choice to download ModelSim as bundle when you download Quartus from their website.

- If you followed our tutorial on installing Quartus and ModelSim, you are ready to go and you can ignore the bullet point bellow. (Alternatively, if you haven't installed any of these programs,

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

please follow our tutorial on Quartus and ModelSim download and setup, then come back to this lab).

- If you have Quartus installed but not ModelSim, you can download it independently from their website, just make sure you choose the same version of Quartus you already have in your computer before starting download. Go to:

  http://dl.altera.com/?edition=subscription&product=modelsim_ae#tabs-2

The following shows the download page from Altera. It is important that you know which version of Quartus you have installed in your computer. If you don't know, open up Quartus, then go to **Help > About Quartus II**
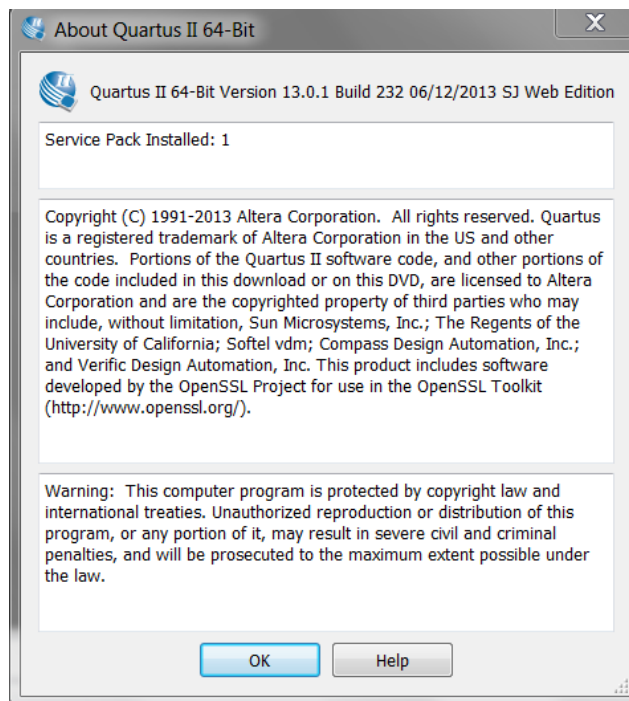
Figure 6. Screenshot of our Quartus version.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

Our version is 13.0 service pack 1, yours may be different.

The next screenshot shows the download page for the Quartus software. Notice that at the top it says Subscription Edition although you may have downloaded the Free Web edition. This is irrelevant: it doesn't matter if it is part of the FREE or SUBSCRIPTION edition, ModelSim as a component is the same for both editions. Make sure you are logged into your Altera account (since you installed Quartus, we assume you have an account already with them).

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

Figure 7. ModelSim download page

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

2.1 ModelSim basics:

1. Open ModelSim. To run ModelSim, go to terminal and type vsim. This tutorial was done on a Windows environment, so we will show the corresponding Windows system screenshots but other operating systems should follow similar steps.

   If you are on Windows go to **Start > Run**. Type in "cmd" (without quotes) in the input field, then hit **Enter**. Alternatively, you could also search for Command Prompt in the search field in the Start Menu.
   If you are on Linux, right click your desktop and click Terminal.
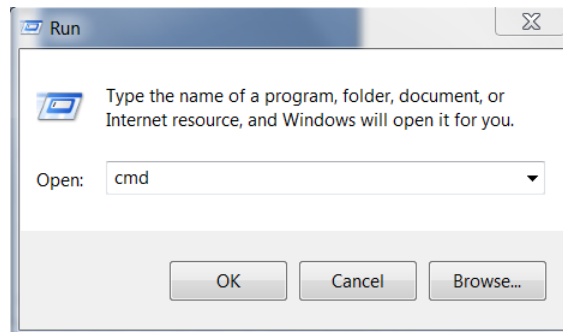
Figure 8. Run Command in Windows

2. In terminal type "vsim" (without the quotes).

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021
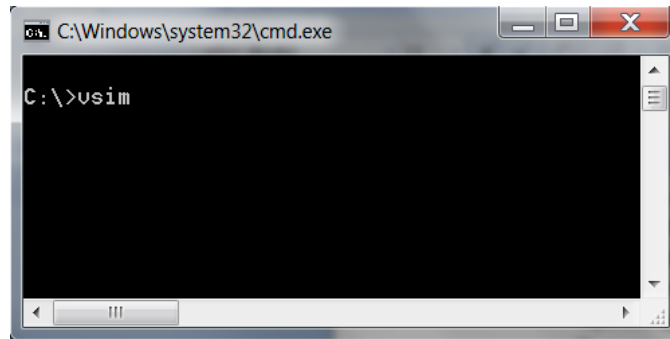
Figure 9. Terminal and ModelSim start command

You are now taken to the main screen of ModelSim, a Welcome splash screen will appear and you are now ready to start testing your circuit designs.

The initial screen of ModelSim is shown below:
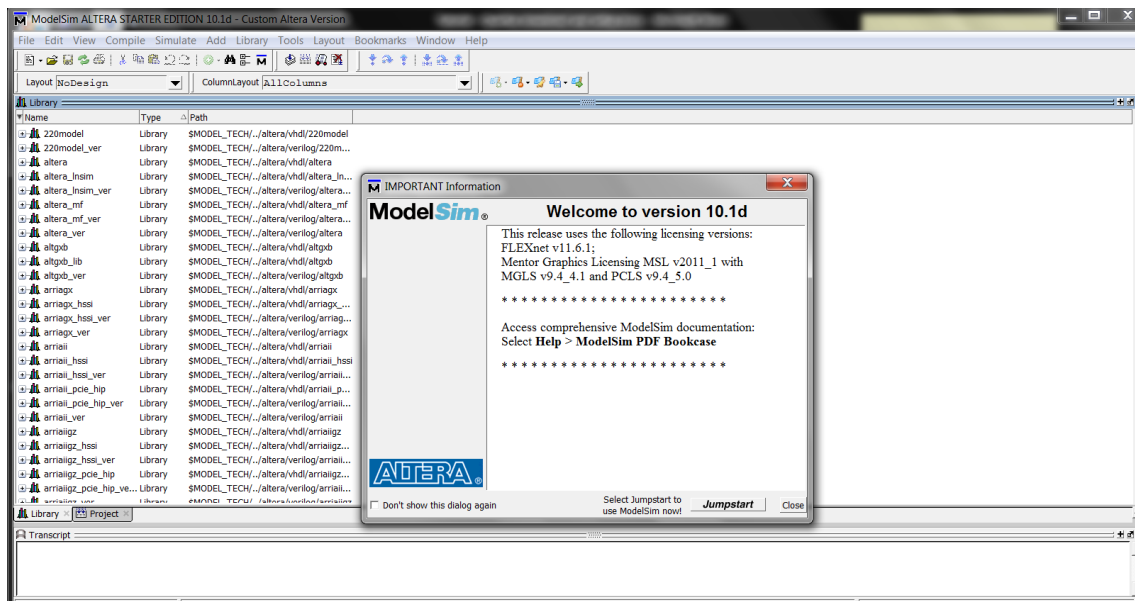
Figure 10. ModelSim Welcome screen

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

3. When the Welcome screen appears, hit Close. Now go to Now go to File > New > Project. If a message appears asking if you want to close the current project just accept.
4. The Create New Project dialog appears, enter a name for your project, you can call it **one_bit_comparator**. See figure 11.
5. Browse a location for your new project.
6. In the Default Library Name, if empty, call it **work**.
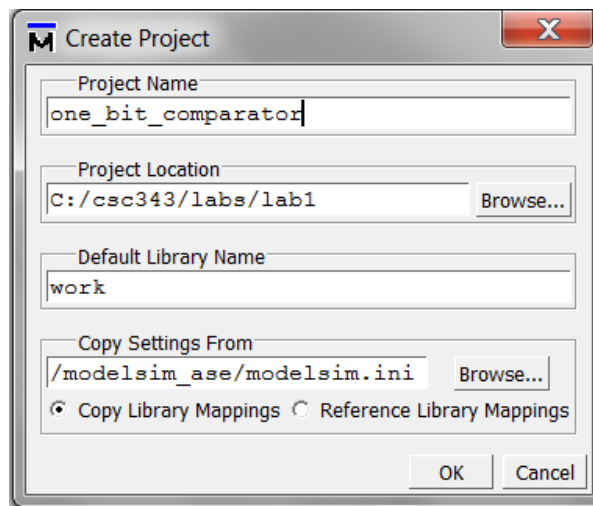
Figure 11. Create New Project dialog in ModelSim

7. The Add items to project dialog appears (See figure 12). In the previous section we described the comparator we are going to implement and we gave you the VHDL code for it. Right now we are going to create a new VHDL file and copy-paste the given code to it. Click **Create New File**.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021
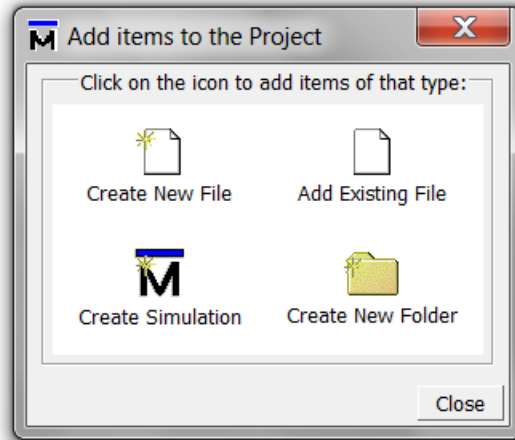
Figure 12. Add items dialog

8.   The Create Project File dialog appears, see figure below. In the File Name field enter **equal**, then hit OK.
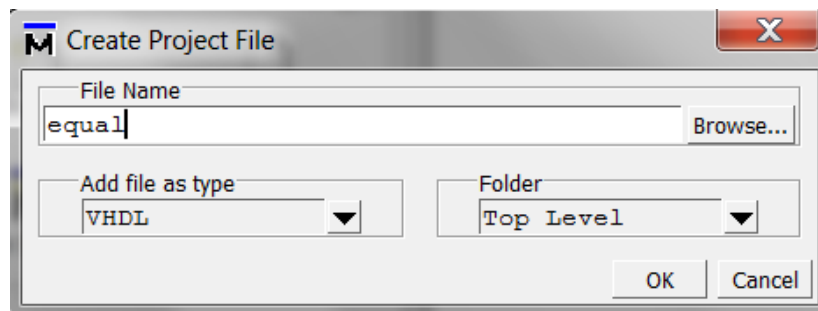
Figure 13. Create Project File dialog

9.   Right click the file equal.vhd you just included, and then choose **Edit**.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021
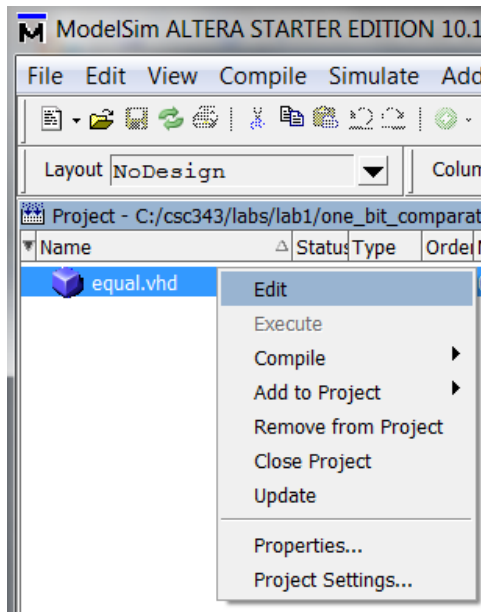
Figure 14. Project navigator in ModelSim with the equal.vhd file

10. A panel with a code editor appears. Copy and paste the code given in the previous section of this lab for the one-bit comparator, called equal.vhd

11. Save this file by going to **File > Save**.

12. In the project navigation panel, right click the equal.vhd file and choose **Compile > Compile Selected**

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021
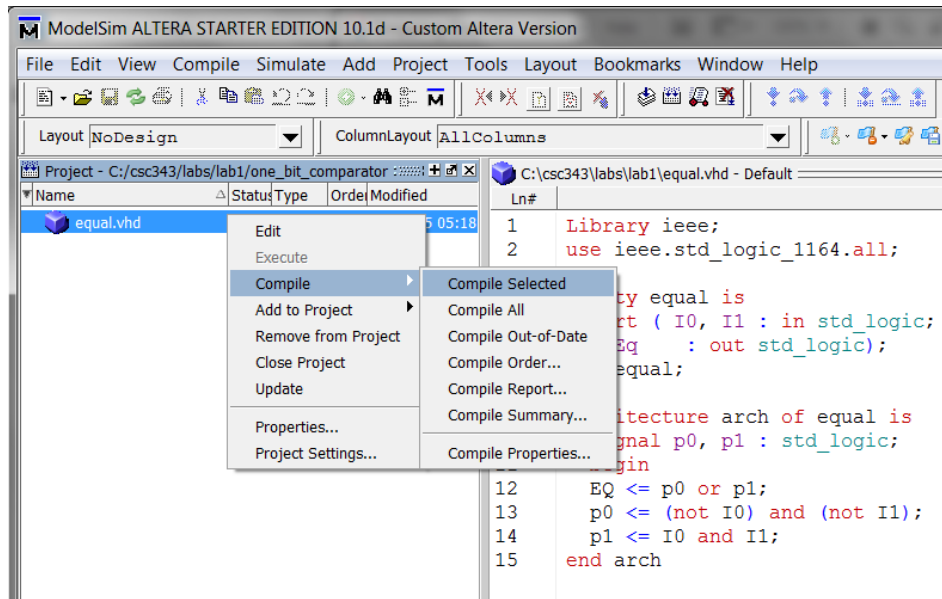
Figure 15. Code editor in ModelSim and Compilation options

13. At the bottom of the main ModelSim window you should see a Transcript panel, it will output a message to notify you if the compilation was successful or of any errors. Also, next to the name of your file you should see a **Green Checkmark** if the file is compiled or a **Blue Question Mark** if it is not.
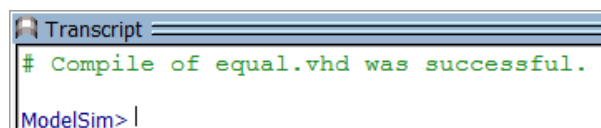
Figure 16. Transcript panel – Compilation successful

14. Go to **Simulate > Start Simulation...**

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
*Instructor: Professor   Izidor Gertner*

February 10, 2021

15. A dialog appears, browse the "work" Library which is (by default) the library we told ModelSim to store the file we wanted to have imported. Select the **equal** entity and click OK.
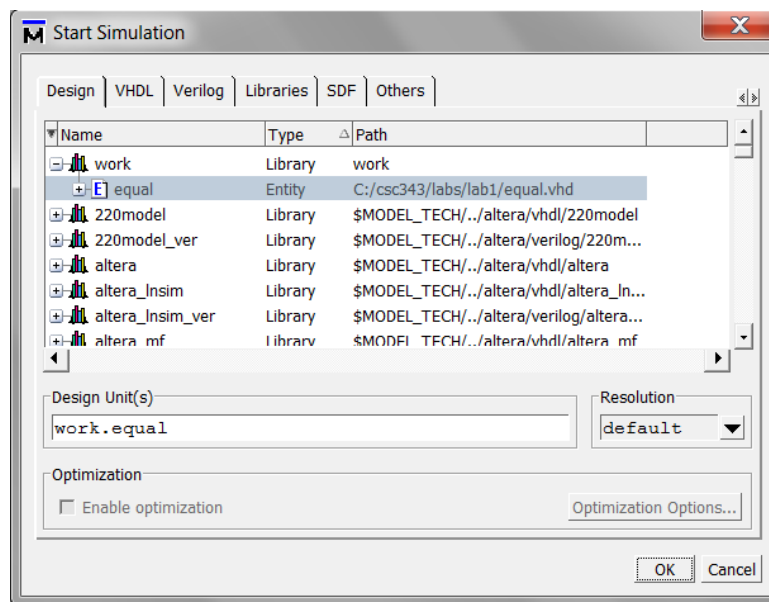
Figure 17. Choosing equal entity which will be simulated

Note: at this stage you are allowed to select multiple files, let's say if you had a one-bit-comparator and a two-bit comparator and want to simulate and compare both, if you have the VHDL files ready in your work Library you can select both and they will be included in your simulation configuration. For now we only have a equal.vhd file which is our one-bit comparator, so we are adding only one file.

16. ModelSim should arrange its layout automatically to allow for simulation. At this point you should have something similar to figure 18. With your mouse, select the one-bit comparator inputs (i0 and i1) and output (Eq) and drag them to the wave panel as you see in the picture:

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

*Instructor: Professor Izidor Gertner*

February 10, 2021
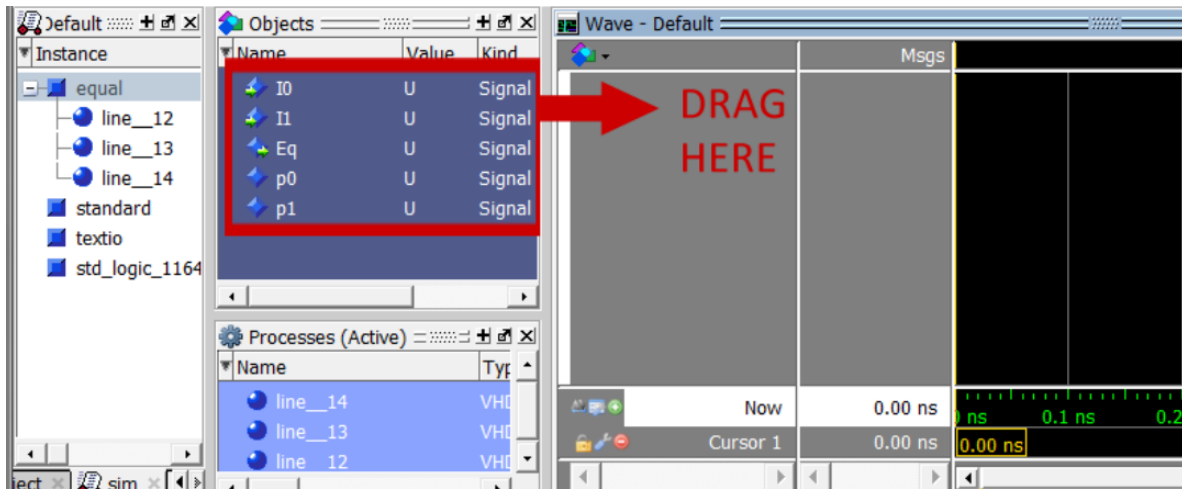
Figure 18a. Dragging inputs and outputs to waveform panel

Alternatively, you could also right click on the name of the VHDL file as shown in Figure 18b and click **Add Wave**
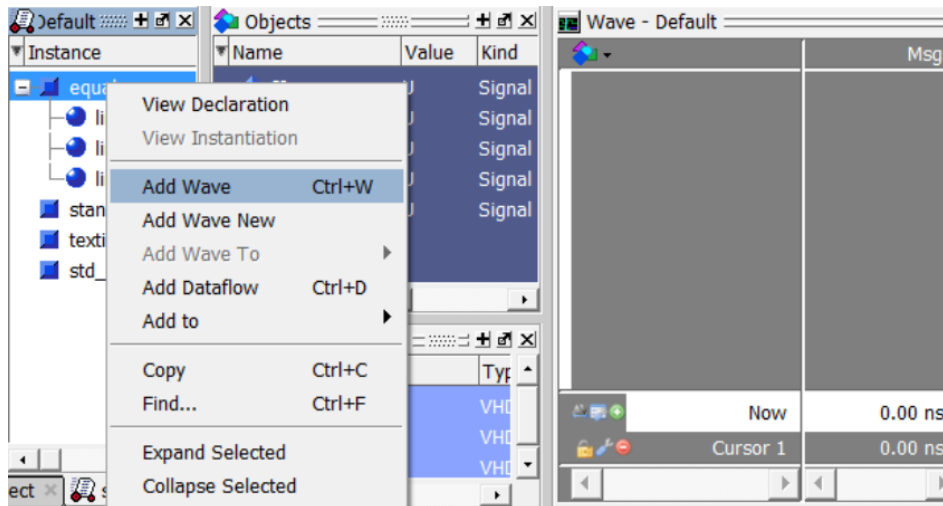
Figure 18b. Adding inputs and outputs through menu to waveform panel

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

17. Now that you have your inputs and outputs in the waveform, it is time to give them values to simulate. Note that a "U" (Undefined) appears to every input/output of your wave list. This means that no value has been set for these. To give a value to an input signal, right click the first input (i0) then select **Force**.
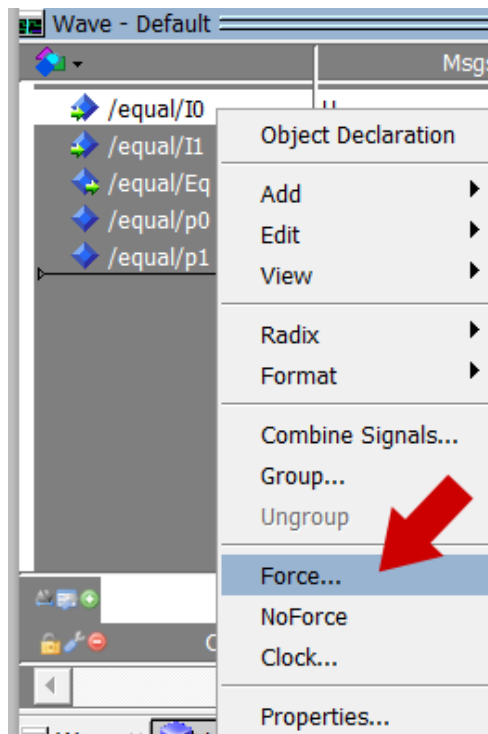
Figure 19. Force values option

18. When the dialog appears, change the Value from U (undefined) to 0. Click OK.

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021
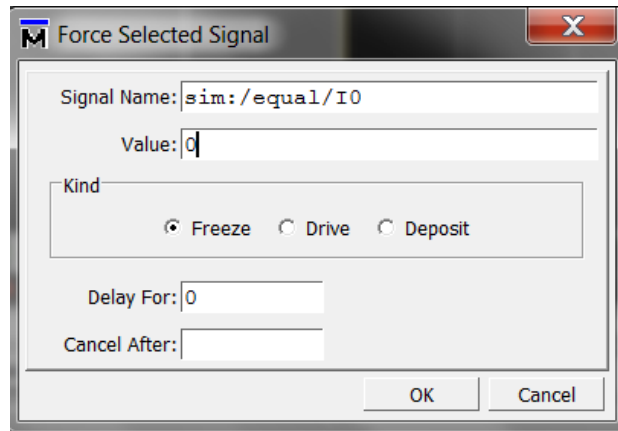
Figure 20. Force Selected Signal dialog

19. Do the same for input i1. Right click it, choose **Force**, and give it a value of 0 as well.

20. Next, it is time to simulate given the initial values of i0=0 and i1=0. Hit F9, hit the Run button (not Run All) or go to Simulation > Run > Run 100.

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021
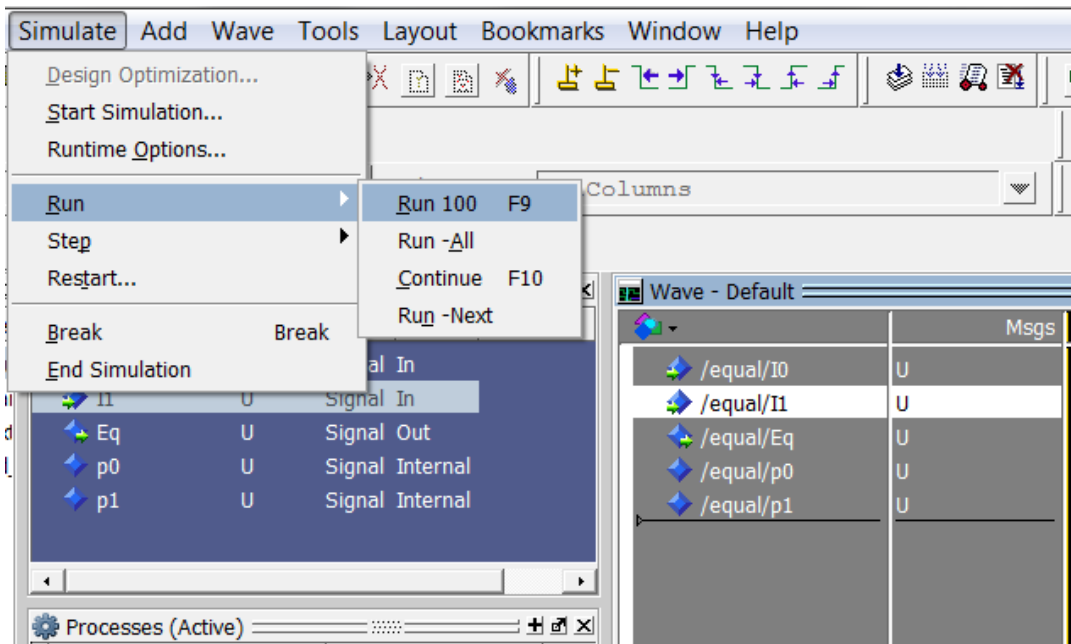
Figure 21. Run options

In case you need to restart your simulation, and thus, clear the input values you assigned, you can do so by clicking the Restart button. When the restart dialog appears hit **OK**.

Figure 22. Run/Restart buttons in Menu bar

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

## *(Detailed version)*
Instructor: Professor   Izidor Gertner

February 10, 2021

21. You should obtain the following result in the waveform panel after your first 100ps run shown in figure 23. Notice how i0 and i1 have a value of 0, Eq has a value of 1 now, which is expected since in this case i0=i1.
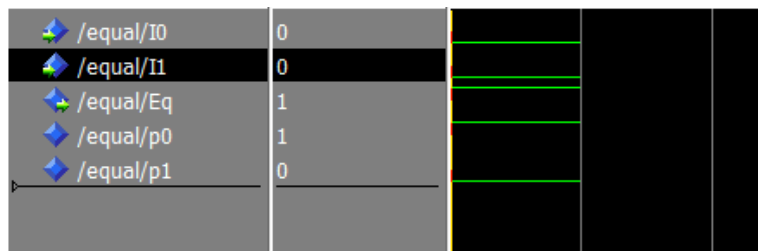
Figure 23. Result of first run after forcing values i0=0, i1=0.
Note Eq=1 since the values of the inputs are equal

22. Keep running the simulation with different values. This time right click on i0, select **Force** and give it a value of 1. Then hit F9 or hit the run button. You should obtain the following waveform:
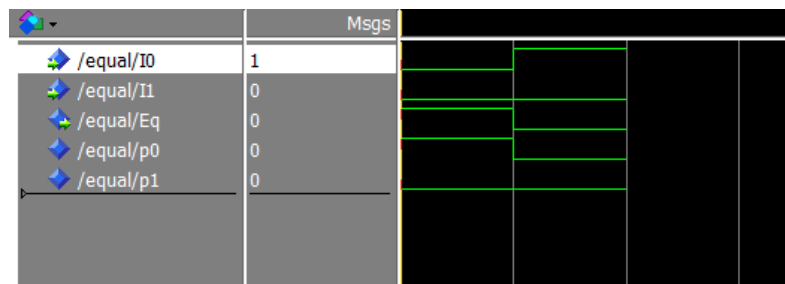
Figure 24. Waveform result after second run

Note that i0 has the value of 1 that we assigned, i1 kept its value of 0 from the prior stage of the simulation that we did in the previous step, and Eq has a value of 0 now, which is expected since i0 and i1 do not match.

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: *Professor   Izidor Gertner*

February 10, 2021

23. Finish the simulation by testing all possible value combinations of i0 and i1 according to the truth table given at the beginning of this lab. We are now going to create a test file in ModelSim, so do not close your project.

24. Close the simulation by going to **Simulate > End Simulation**.

Figure 25. Ending simulation

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

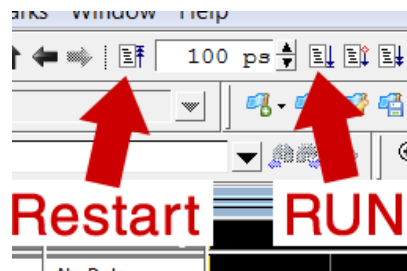*Instructor: Professor   Izidor Gertner*

February 10, 2021

3. USING A TEST BENCH IN MODELSIM

3.1 Test bench idea

After the code is developed, it can be simulated in a host computer to verify the correctness of the circuit operation and can be synthesized to a physical device (DE2 board for example). As we showed you in the previous section, a simulation can be done without a test bench but it is more professional and exact to use a test bench because you will have better control over your simulation.

3.1 Test file

We create a special program, known as a test bench, to imitate a physical lab bench. Here's the test file for the one-bit comparator:

test_equal.vhd
```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity test_equal is
 5 end test_equal;
 6
 7 architecture arch_test of test_equal is
 8 component equal
 9 port ( I0, I1 : in std_logic;
10         Eq    : out std_logic );
11 end component;
12
13 signal p1, p0, pout : std_logic;
14 signal error        : std_logic := '0';
15 begin
16 uut: equal port map (I0 => p0, I1 => p1, Eq => pout);
17 process
18 begin
19 p0 <= '1';
20 p1 <= '0';
21 wait for 1 ns;
22 if (pout = '1') then
23         error <= '1';
```

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

```
24 end if;
25 wait for 200 ns;
26 p0 <= '1';
27 p1 <= '1';
28 wait for 1 ns;
29 if (pout = '0') then
30         error <= '1';
31 end if;
32 wait for 200 ns;
33 p0 <= '0';
34 p1 <= '1';
35 wait for 1 ns;
36 if (pout = '1') then
37         error <= '1';
38 end if;
39 wait for 200 ns;
40 p0 <= '0';
41 p1 <= '0';
42 wait for 1 ns;
43 if (pout = '0') then
44         error <= '1';
45 end if;
46 wait for 200 ns;
47
48 if (error = '0') then
49         report "No errors detected. Simulation successful" severity
50 failure;
51         else
52                 report "Error detected" severity failure;
53 end if;
54
55 end process;
56 end arch_test;
```

Read and try to understand the test file as much as you can. This test file declares "equal" (line 8) as a component and uses it without defining – it assumes that "equal" is defined somewhere else and we are just using this functionality (line 16) to test for specific cases or values. For example, you see that p0 and p1 are given specific values at certain time intervals and, assuming we know what value that should be produced, we test the result for the opposite value. If that incorrect result is reached, then we know we have an error and a message is displayed to the user.

Laboratory Project 2
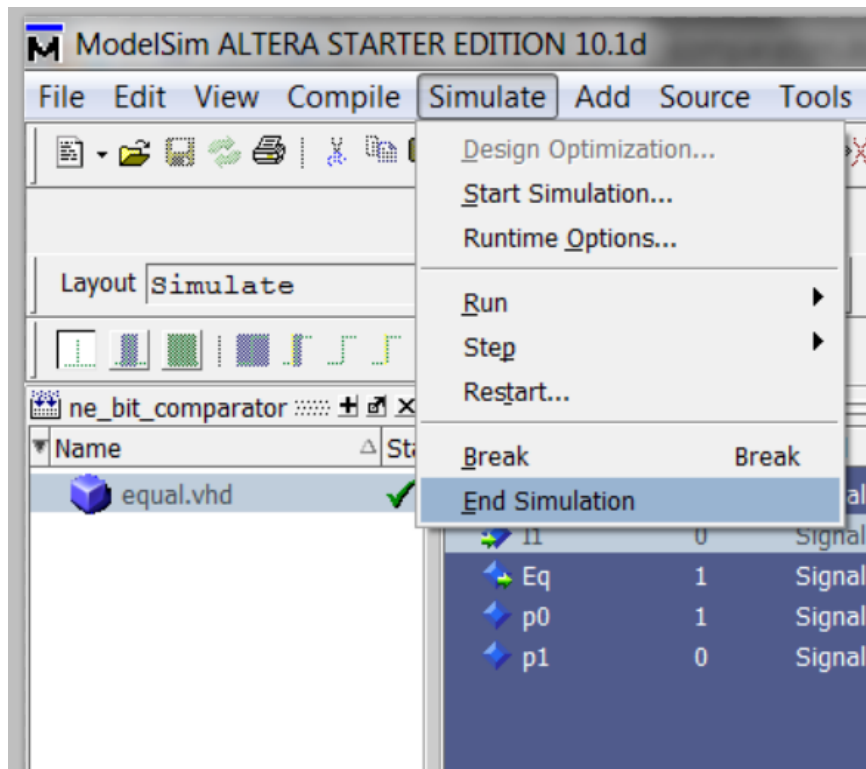# Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

3.3 Running the simulation

1. With the one_bit_comparator project open, click on the Project tab (see figure26). Afterwards, Right click on an empty space in the project navigation panel. Then select **Add to project > New File…**
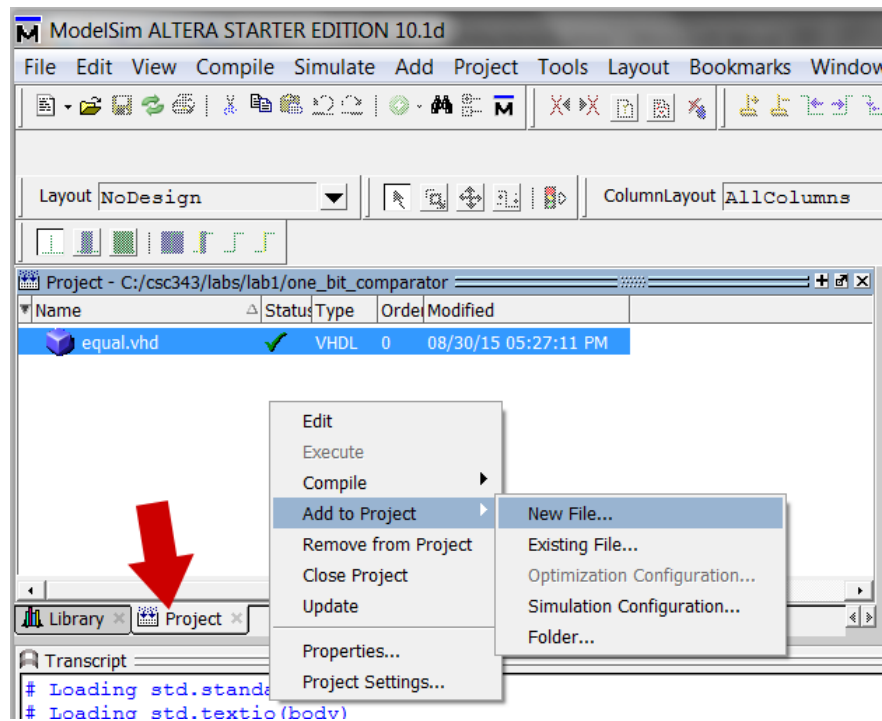
Figure 26. Creating a new VHDL file

2. The Create Project file dialog appears. In the input field for File Name enter **test_equal**, then hit OK.

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021
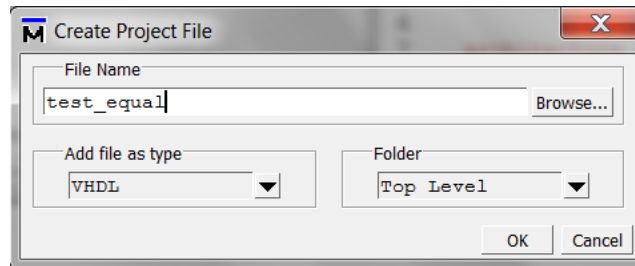
Figure 27. Giving a name to our new file

3. The newly created file appears in the project panel. Right click the test_equal file and choose **Edit**.
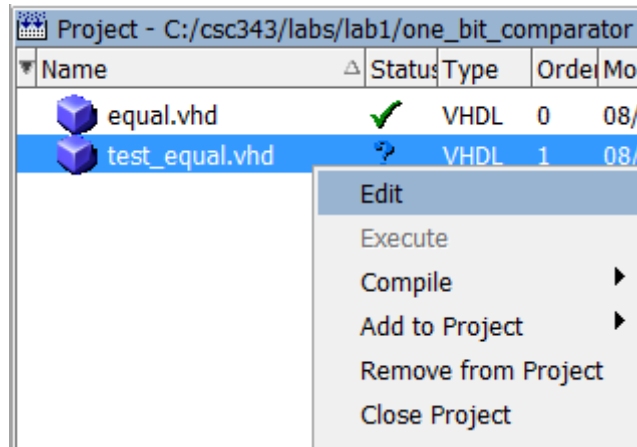
Figure 28. Editing the file

4. The code editor panel appears, copy and paste the code we gave you for the test_equal.vhd file.

5. Save the changes to this file. Go to **File > Save**.

6. Before we simulate, we need to compile this file. **Right click** on test_equal.vhd, then choose **Compile > Compile Selected**.

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor Izidor Gertner

February 10, 2021

7. The question mark that was there previously has turned into a green checkmark. This means our file has been compiled correctly and you should be ready for simulation. If you run into compilation problems, make sure you are copying the code correctly.
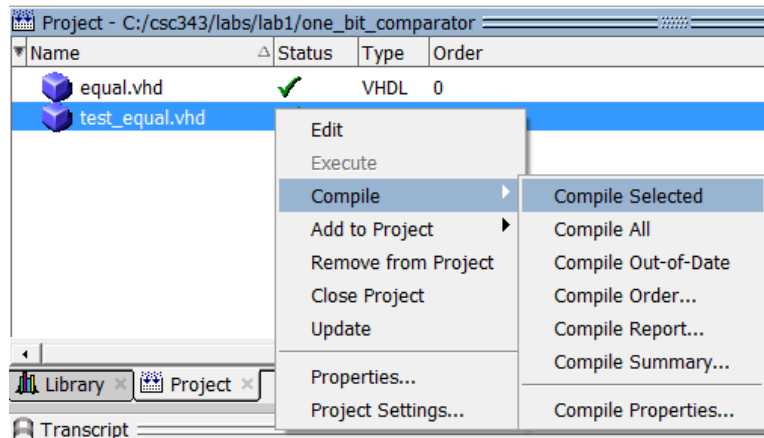
Figure 29. Compiling the test_equal.vhd file

8. Time to simulate. Go to **Simulation > Start Simulation**. The Simulation dialog appears so you can choose which entity to simulate. Expand the work library we created in the previous section, then select test_equal and click **OK**.
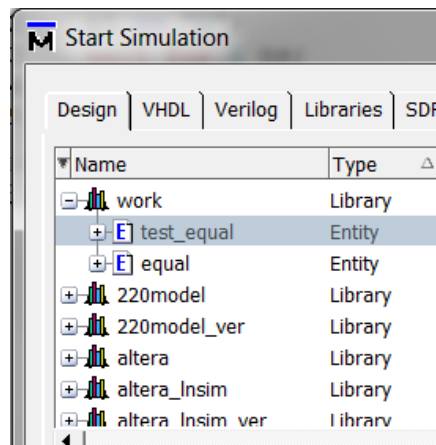
Figure 30. Choosing the test_equal entity

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

9. Starting the simulation will configure ModelSim to the simulation layout. This will give you the screen shown below:
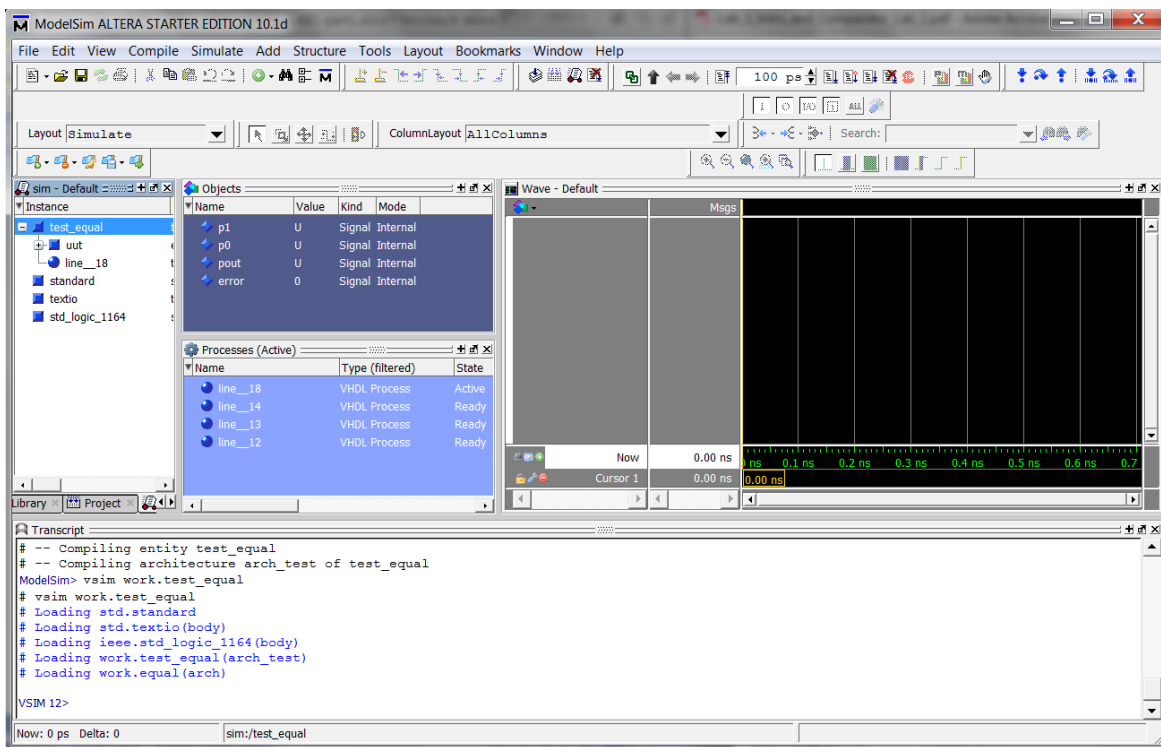
Figure 31. Simulation environment for test_equal

10. This should be familiar to you now. It is time to add the inputs and outputs as we did for equal.vhd file. Right click on test_equal in the instance window and select **Add Wave**. Alternatively, you can also right-click **test_equal** and do select **Add to > Wave > All items in region**. Your screen should look similar to the picture below with inputs and outputs populating the waveform screen on the right pane.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021
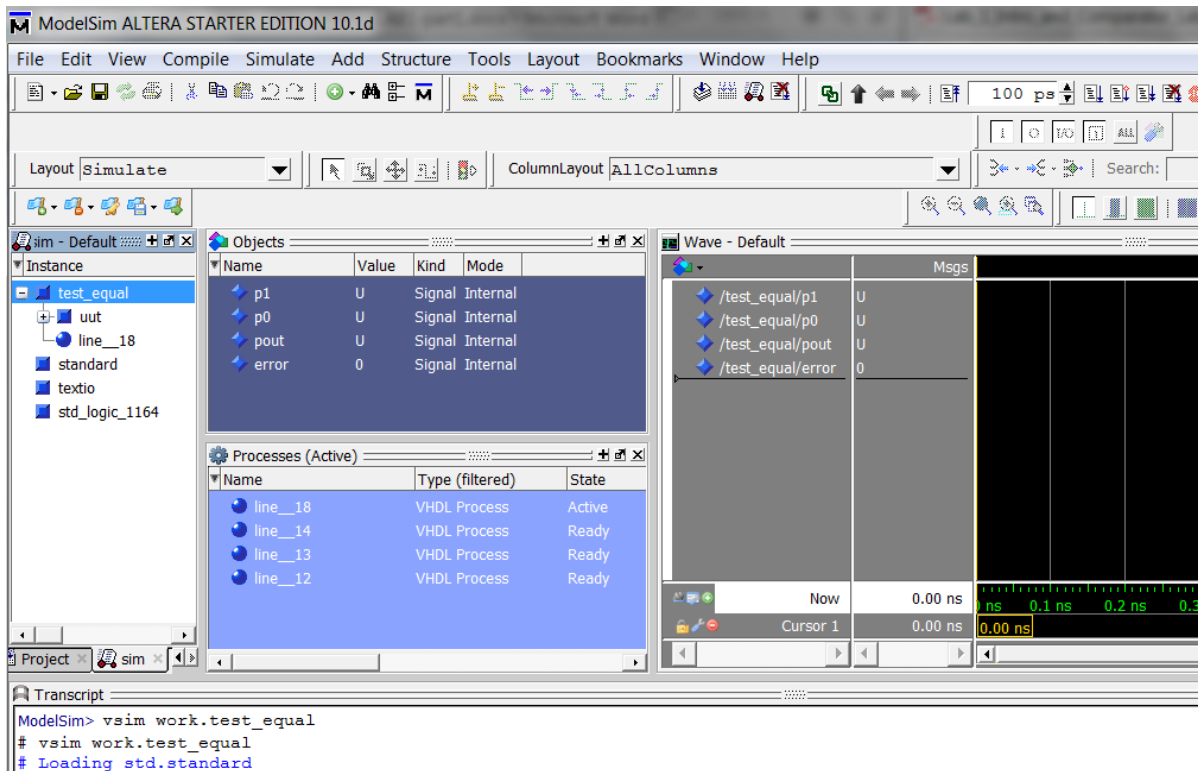
Figure 32. Inputs and outputs added for simulation

11. To run the simulation, click on the "Run All" icon. Normally a simulation will run until the "Break" icon is pressed, however in our test file the simulation is forced to terminate after a short period of time.

Figure 33. Run All and Break buttons in Main ModelSim Menu bar

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

12. After simulating your design you will see a waveform in the wave window, you may have to close the ModelSim text editor to see it, or click on the Wave tab at the bottom of the pane (see figure below). You can zoom in or zoom out to observe your results. To see all the values of your run right click anywhere in the black pane and choose **Zoom Full** (see figure).
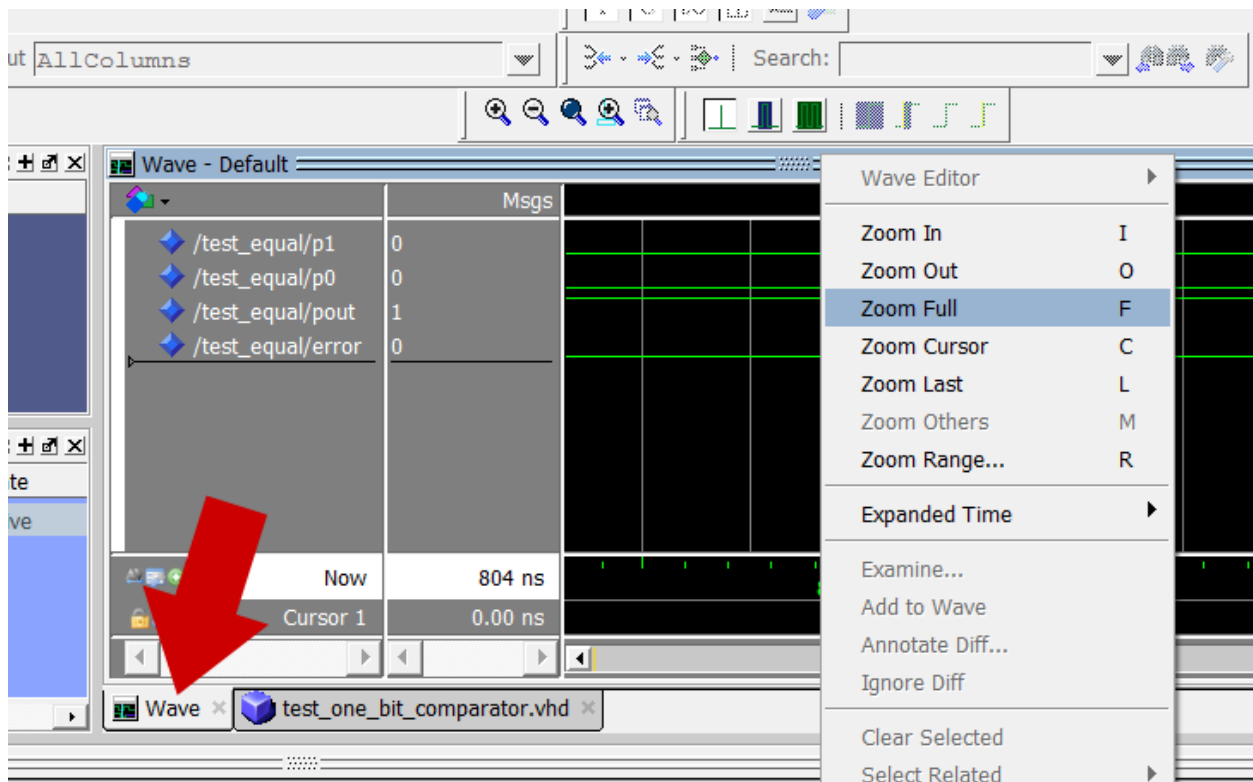
Figure 34. The Wave tab and full zoom of your waveform

The values you have gotten may not be exactly like the picture below. Analyze that your results are correct based on the truth table. Use the yellow scrollbar to scroll through the values. In the Msgs section you see the values each input/output takes (0 or 1) as you scroll through the waveform.

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021
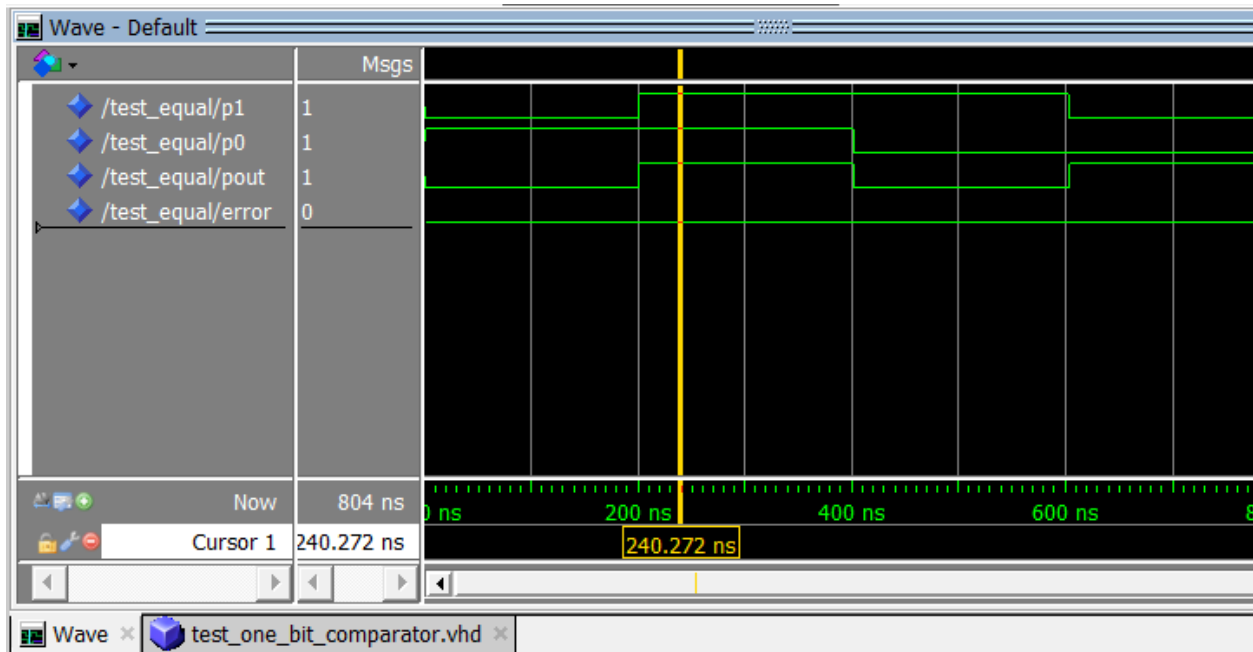
Figure 35. Completed waveform simulation of Test bench

The test code also produces a report in the transcript window on the bottom of the screen. If no errors are found it report that simulation was successful, else is report that an error was detected. You can use the error signal in the waveform to see at what exact point the error was found. Here is how a successful simulation would look like.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

Figure 36. Succesful simulation transcript

2-BIT COMPARATOR

Now that we have written our first piece of VHDL code we want to make it a bit more complicated. Now we want to make a two-bit comparator. The program will take two 2-bit inputs and compare them and check if their equal or not. This can be done in two ways:

- Logic (logic-vector) inputs/outputs
- Port Maps

 You will implement the two-bit comparator in both ways and at the end you will understand their differences.

Note: For the 2-bit and 8bit comparator you will perform the exact same procedure we showed for the 1-bit comparator. You will write your VHDL code, run the waveform simulation and test in ModelSim, mount your design in Quartus, etc.

Using Logic Vectors:

If you draw out the truth table for the two-bit comparator you will be able to derive the logical statement given below:

$$aeqb = (a1'.b1').(a0'.b0') + (a1'.b1').(a0.b0) + (a1.b1).(a0'b0') + (a1.b1).(a0.b0)$$

Fill out the missing parts in the code below using the logical expression above - notice that aeqb can be expressed as aeqb = p0 + p1 +p2 +p3

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
*Instructor: Professor   Izidor Gertner*

February 10, 2021

two_bit_equal.vhd

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity two_bit_equal is
 5        port ( a, b    : in std_logic_vector(1 downto 0);
 6                   aeqb    : out std_logic);
 7 end two_bit_equal;
 8
 9 architecture arch of two_bit_equal is
10 signal p0, p1, p2, p3 : std_logic;
11 begin
12 aeqb <= p0 or p1 or p2 or p3;
13 p0 <= --Enter you code here;
14 p1 <= --Enter you code here;
15 p2 <= --Enter you code here;
16 p3 <= --Enter you code here;
17 end arch;
```

The only difference in this code is that inputs a and b are declared as "STD_LOGIC_VECTOR(1 downto 0)" which means it is a vector of size two.

Now to test our design we need a test file. Again the test file will be given to you but try to understand how it works.

test_two_bit_equal.vhd

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity test_two_bit_equal is
 5 end test_two_bit_equal;
 6
 7 architecture arch_test of test_two_bit_equal is
 8 component two_bit_equal
 9 port ( a, b    : in std_logic_vector(1 downto 0);
10          aeqb    : out std_logic);
11 end component;
12
```

CS 343,   Spring 2021

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

```vhdl
13 signal p1, p0   : std_logic_vector(1 downto 0);
14 signal pout     : std_logic;
15 signal error    : std_logic := '0';
16 begin
17 uut: two_bit_equal port map(a => p0, b=> p1, aeqb => pout);
18 process
19 begin
20 p0 <= "00";
21 p1 <= "00";
22 wait for 1 ns;
23 if (pout = '0') then
24        error <= '1';
25 end if;
26 wait for 200 ns;
27 p0 <= "01";
28 p1 <= "00";
29 wait for 1 ns;
30 if (pout = '1') then
31        error <= '1';
32 end if;
33 wait for 200 ns;
34 p0 <= "01";
35 p1 <= "11";
36 wait for 1 ns;
37 if (pout = '1') then
38        error <= '1';
39 end if;
40 wait for 200 ns;
41 p0 <= "11";
42 p1 <= "00";
43 wait for 1 ns;
44 if (pout = '1') then
45        error <= '1';
46 end if;
47 wait for 200 ns;
48 p0 <= "11";
49 p1 <= "11";
50 wait for 1 ns;
51 if (pout = '0') then
52        error <= '1';
53 end if;
54 wait for 200 ns;
55 p0 <= "10";
56 p1 <= "11";
57 wait for 1 ns;
```

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

```
58 if (pout = '1') then
59         error <= '1';
60 end if;
61 wait for 200 ns;
62 p0 <= "10";
63 p1 <= "10";
64 wait for 1 ns;
65 if (pout = '0') then
66         error <= '1';
67 end if;
68 wait for 200 ns;
69 p0 <= "11";
70 p1 <= "01";
71 wait for 1 ns;
72 if (pout = '1') then
73         error <= '1';
74 end if;
75 wait for 200 ns;
76
77 if (error = '0') then
78         report "No errors detected. Simulation successful" severity
79 failure;
80         else
81               report "Error detected" severity failure;
82 end if;
83 end process;
84 end arch_test;
```

Using Port Maps:

As you can see writing the two-bit comparator using logical expressions could get a bit tiresome and confusing especially as we increase the number of bits. There is an easier way to do this using "PORT MAPS". What port map does is that it uses other components that we or others have created and connects them together to make something bigger. For example to make a two-bit comparator we will connect 2 one-bit comparators making life much easier and simpler.
Below is the code for the two-bit comparator with the use of port maps:

two_bit_equal_port.vhd

```
 1 library ieee;
```

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

```vhdl
 2 use ieee.std_logic_1164.all;
 3
 4 entity two_bit_equal_port is
 5 port (
 6         a, b: in std_logic_vector(1 downto 0);
 7         aeqb : out std_logic);
 8 end two_bit_equal_port;
 9
10 architecture arch of two_bit_equal_port is
11
12         --component declaration...we are telling the compiler
13         --which components we want to use from the library.
14         component equal
15         port (
16                 I0, I1: in std_logic;
17                 Eq : out std_logic);
18         end component;
19         signal e0,e1: std_logic;
20
21         begin
22
23         --instantiates two one-bit comparators
24         H1: equal port map(i0=>a(0), i1=>b(0), eq=>e0);
25         H2: equal port map(i0=>a(1), i1=>b(1), eq=>e1);
26
27         --a and b are equal if individual bits are equal.
28         aeqb <= e0 and e1;
29
30 end arch;
```

Observe the syntax of using port maps because it is very important as we continue with more advanced projects. Port maps act as wires connecting different components or black boxes together. H1 and H2 used in the code are just names for the components; you can name them anything you want.
To test this design use the same **test_two_bit_equal** test file given above but change the name of the component to **two_bit_equal_port** since you are simulating this file instead. Simulate your code and analyze your results.

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor Izidor Gertner

February 10, 2021

8-BIT COMPARATOR

Using the two-bit comparator example, write the VHDL code for an eight-bit comparator. Your code should look very similar to the code from the **two_bit_equal_port** file, with the only difference that instead of using two instances of the "equal" component, you will use eight instances of the one-bit comparator. Use the following test code to verify your results.

test_eight_bit_equal_port_map.vhd

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity test_eight_bit_equal_port_map is
 5 end test_eight_bit_equal_port_map;
 6
 7 architecture arch_test of test_eight_bit_equal_port_map is
 8
 9 component eight_bit_equal_port_map
10   port ( a, b : in std_logic_vector(7 downto 0);
11          aeqb  : out std_logic);
12 end component;
13
14 signal p1, p0 : std_logic_vector(7 downto 0);
15 signal pout   : std_logic;
16 signal error  : std_logic := '0';
17 begin
18   uut: eight_bit_equal_port_map port map(a => p0, b => p1, aeqb => pout);
19   process
20     begin
21       p0 <= "00000000";
22       p1 <= "00000000";
23       wait for 1 ns;
24       if (pout = '0') then
25         error <= '1';
26       end if;
27       wait for 200 ns;
28       p0 <= "01010101";
29       p1 <= "00010101";
```

Laboratory Project 2

Introduction to VHDL, ModelSim and Quartus using Comparators

(Detailed version)

Instructor: Professor   Izidor Gertner

February 10, 2021

```
30          wait for 1 ns;
31          if (pout = '1') then
32            error <= '1';
33          end if;
34          wait for 200 ns;
35          p0 <= "01100101";
36          p1 <= "11111001";
37          wait for 1 ns;
38          if (pout = '1') then
39            error <= '1';
40          end if;
41          wait for 200 ns;
42          p0 <= "11110011";
43          p1 <= "00010100";
44          wait for 1 ns;
45          if (pout = '1') then
46            error <= '1';
47          end if;
48          wait for 200 ns;
49          p0 <= "11001100";
50          p1 <= "11001100";
51          wait for 1 ns;
52          if (pout = '0') then
53            error <= '1';
54          end if;
55          wait for 200 ns;
56          p0 <= "10010001";
57          p1 <= "11100111";
58          wait for 1 ns;
59          if (pout = '1') then
60            error <= '1';
61          end if;
62          wait for 200 ns;
63          p0 <= "10111001";
64          p1 <= "10111001";
65          wait for 1 ns;
66          if (pout = '0') then
67            error <= '1';
68          end if;
69          wait for 200 ns;
70          p0 <= "11010011";
71          p1 <= "01101001";
72          wait for 1 ns;
73          if (pout = '1') then
74            error <= '1';
```

Laboratory Project 2
Introduction to VHDL, ModelSim and Quartus using Comparators
(Detailed version)
Instructor: Professor   Izidor Gertner

February 10, 2021

```
75        end if;
76        wait for 200 ns;
77
78        if (error = '0') then
79          report "No errors detected. Simulation successful" severity
80          failure;
81        else
82          report "Error detected" severity failure;
83        end if;
84
85   end process;
86 end arch_test;
```