Introduction to VHDL, ModelSim, and Quartus using Comparators

MdShahid Bin Emdad

February 27th, 2022

CSC 34200/34300

TABLE OF CONTENTS

Objective:

The objective of this assignment is to design and compile 1-bit, 2-bit and 8-bit comparators by using tutorial code from the given assignment. We will also have to do simulation and testing it through the testbench code given by the professors. To make the assignment more challenging, we are instructed to make our code optimized.

Description of Specifications, and Functionality:

The digital system I used in this assignment is Quartus Prime 20.1.1 and ModelSimSetup-20.1.1. There two packages needed are cyclonev and cyclonevi (both versions are 20.1.1). In the VHDL editor, I wrote my VHDL code to get the circuit output and then used Modelsim to simulate and run my circuit over time (ns unit).

I. 1-bit comparator normal & Optimized

In figure 1, I create the 1-but comparator and named the entity "***Emdad_02_27_22_equal***" and save the project as "***Emdad_02_27_22_equal_1_bit_comparator***".

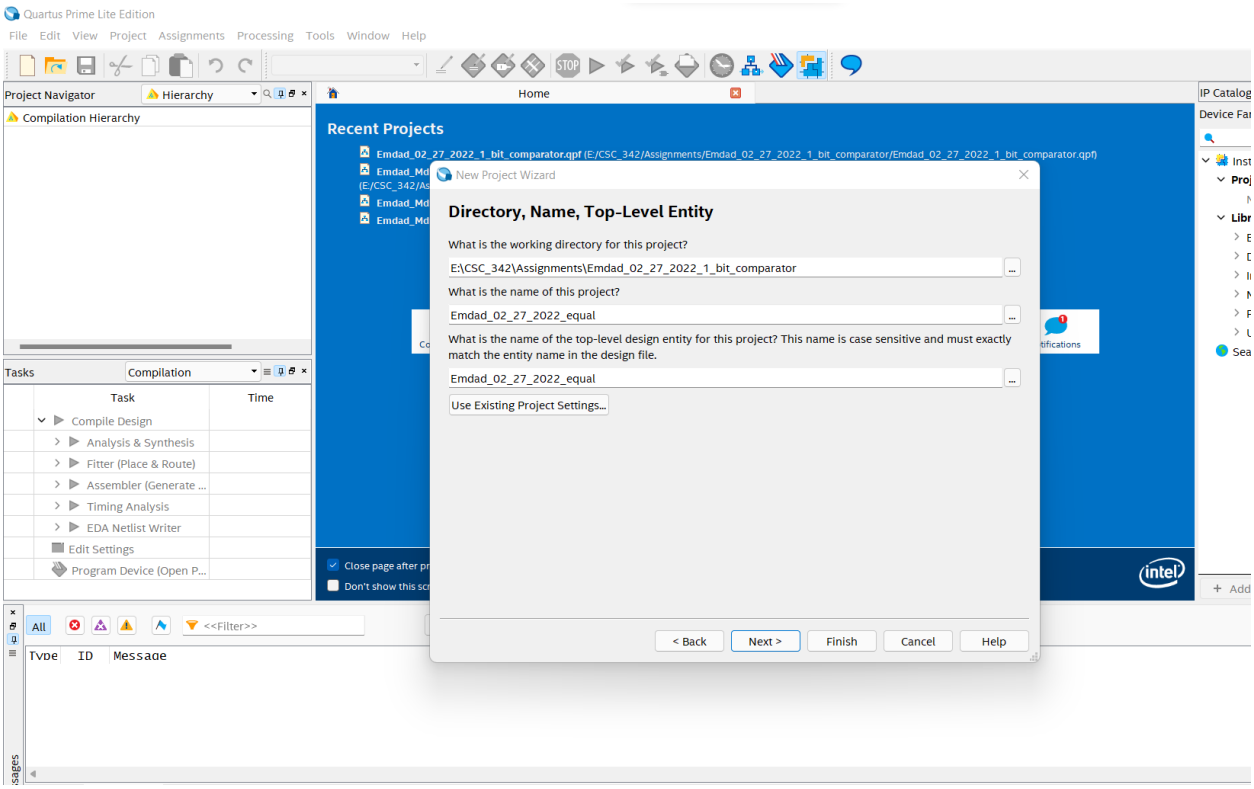

Figure 1: creating the project

In figure 2, we can see a summary of creating the project successfully.

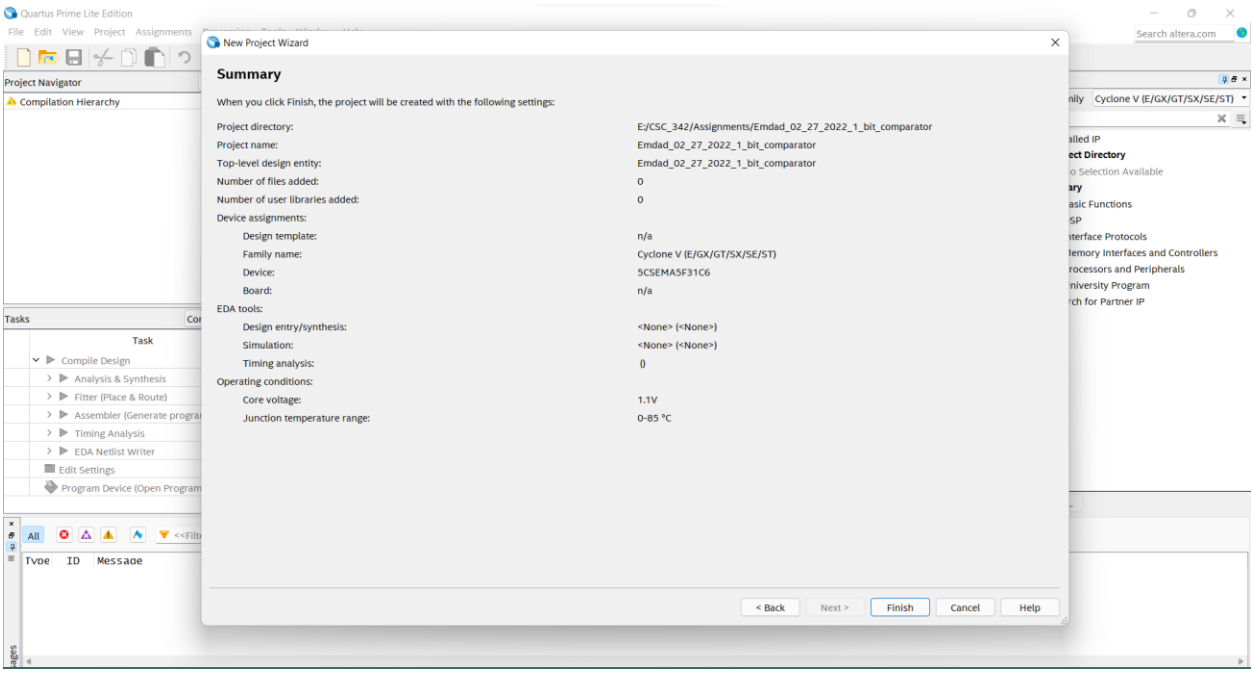

Figure 2: Project Summary Report

In figure 3, I created a VHDL file and inserted code for 1-bit comparator.

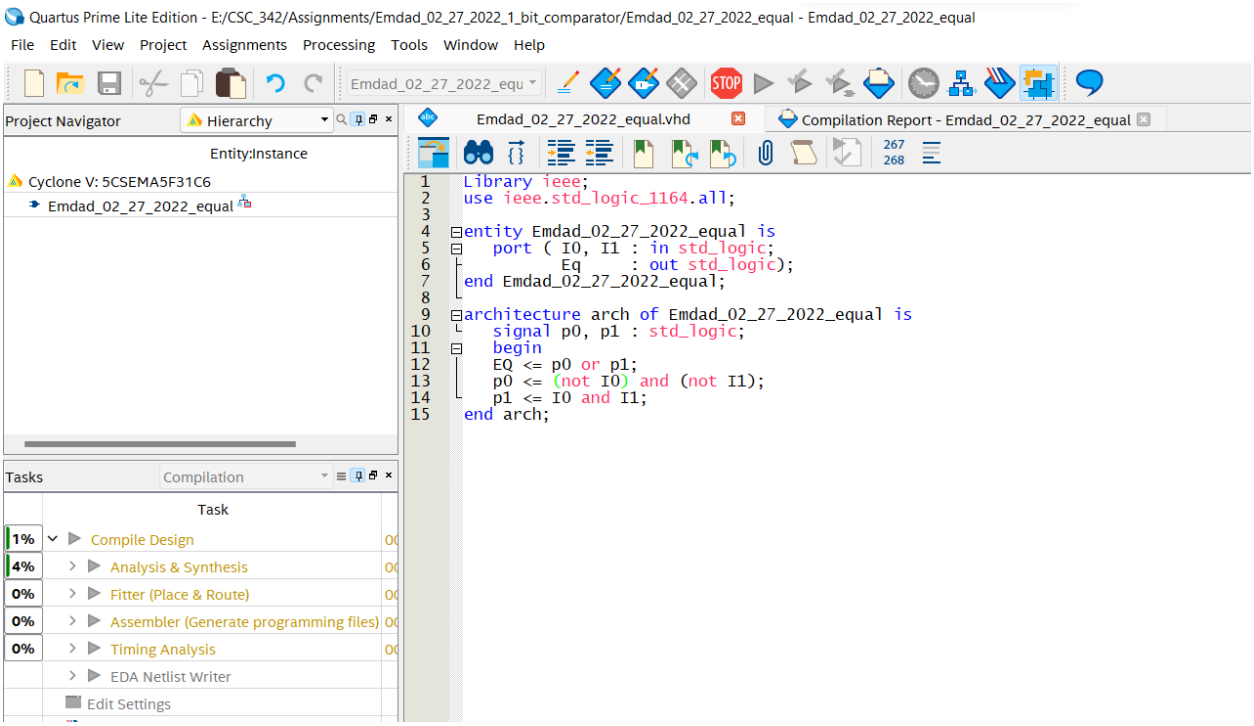

Figure 3: 1 bit comparator VHDL file

In figure 4, we can see the compilation successful report below.

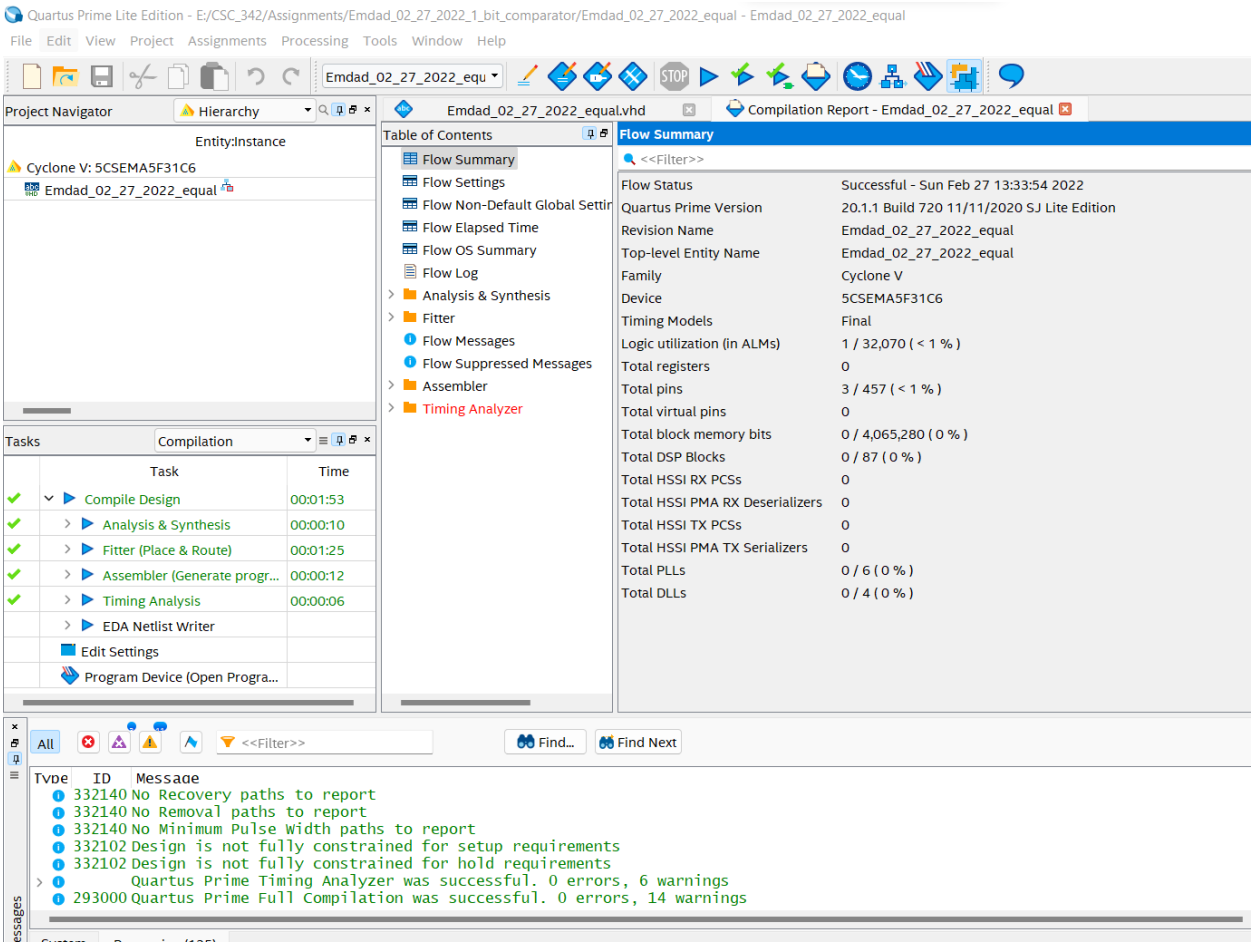

Figure 4: Compilation Successful

In figure 5, we can see modelsim and I see inserted the test benchcode to get the simulation for the 1-bit comparator and also test the code.

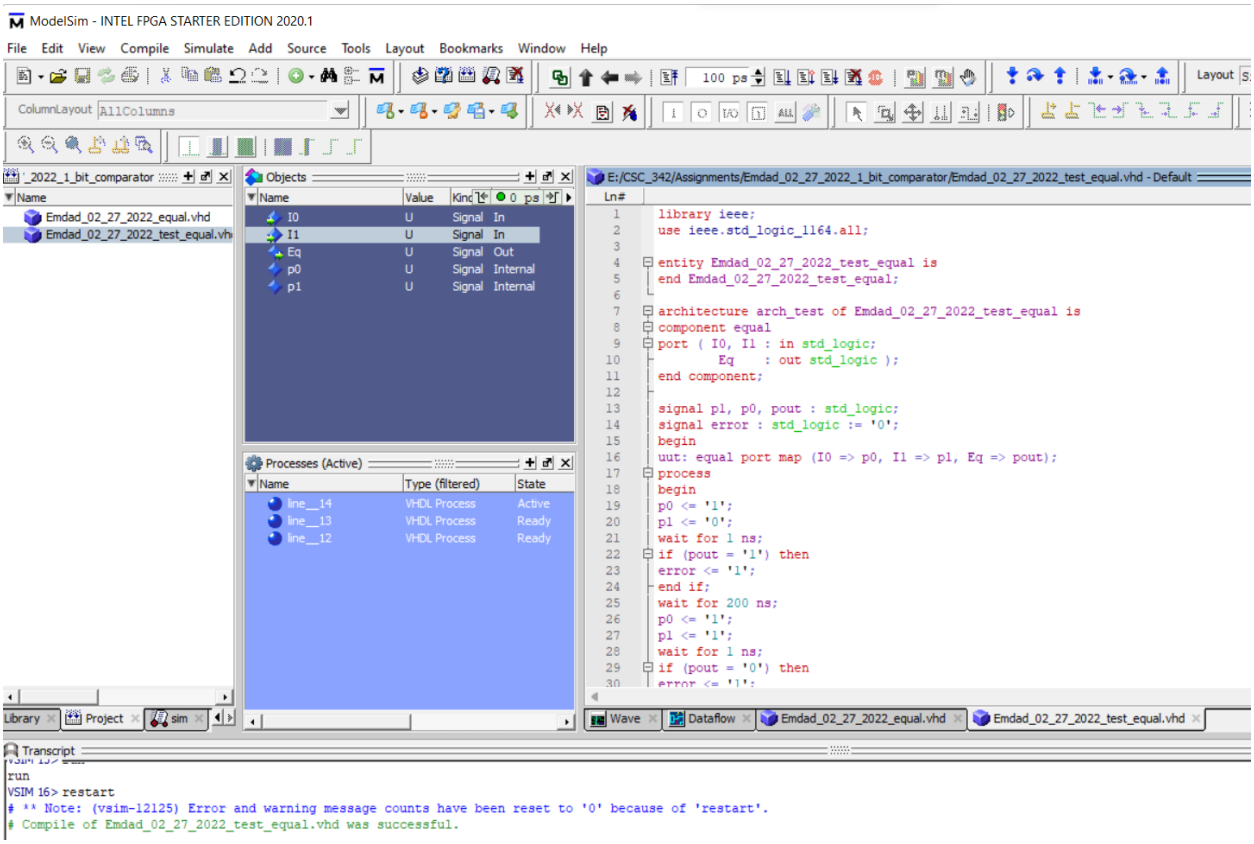

Figure 5: Testbench compilation

In figure 6, we can see the simulation output given by the testbench code and it tested good.

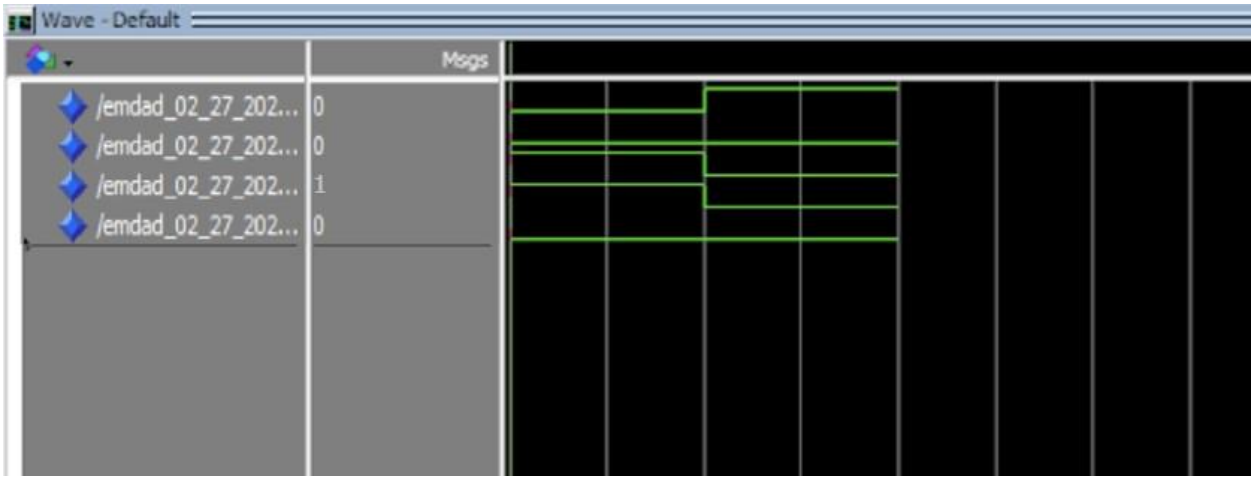

Figure 6: Simulation Output

In figure 7, I optimized the code by shorted the main function. I made it to one line and no signals needed. It ran successfully.

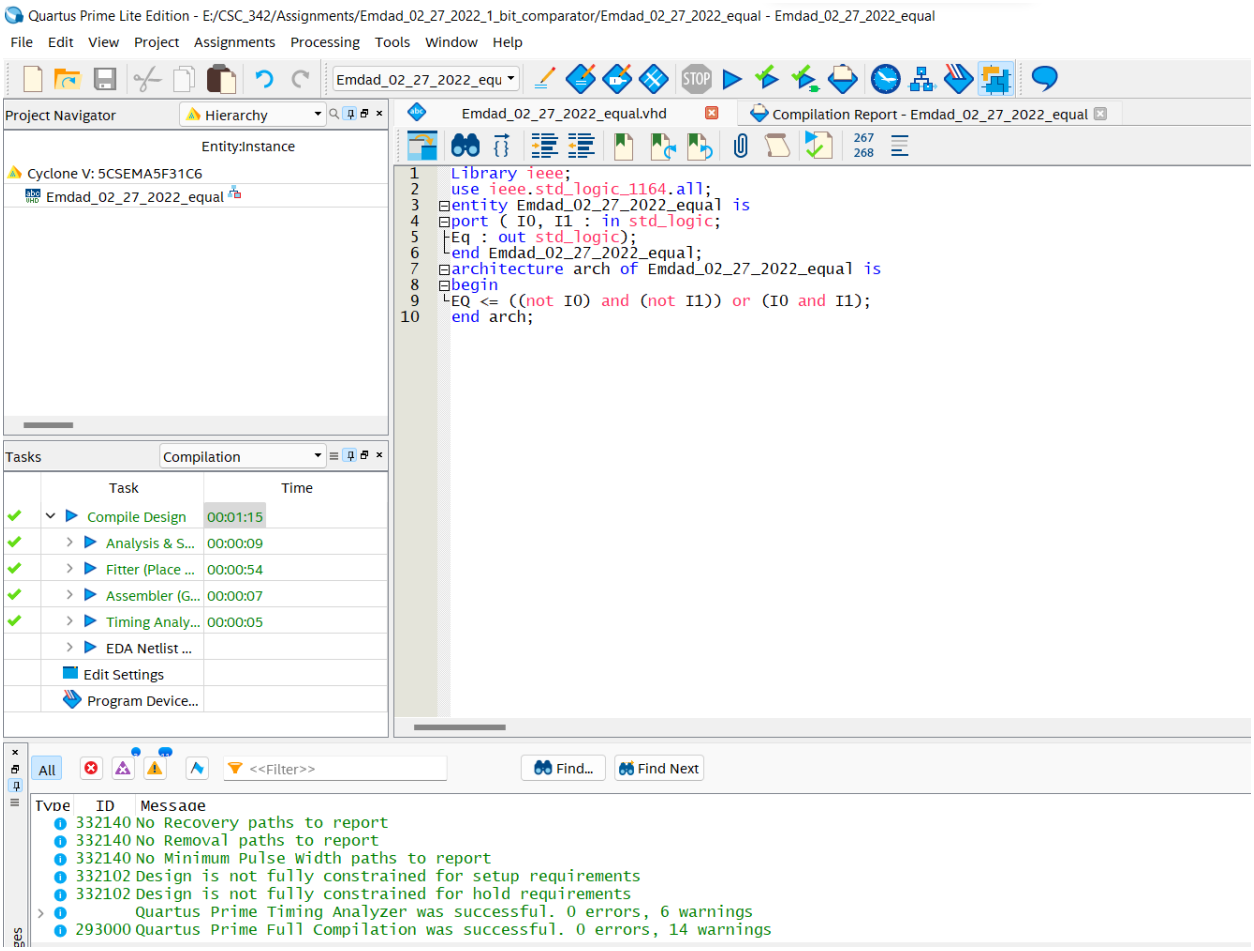

Figure 7: Optimized Code

II. 2-bit comparator normal, ports & Optimized:

In figure 8, we can see I inserted the code the 2-bit comparator in the modelsim and it compiled successfully.

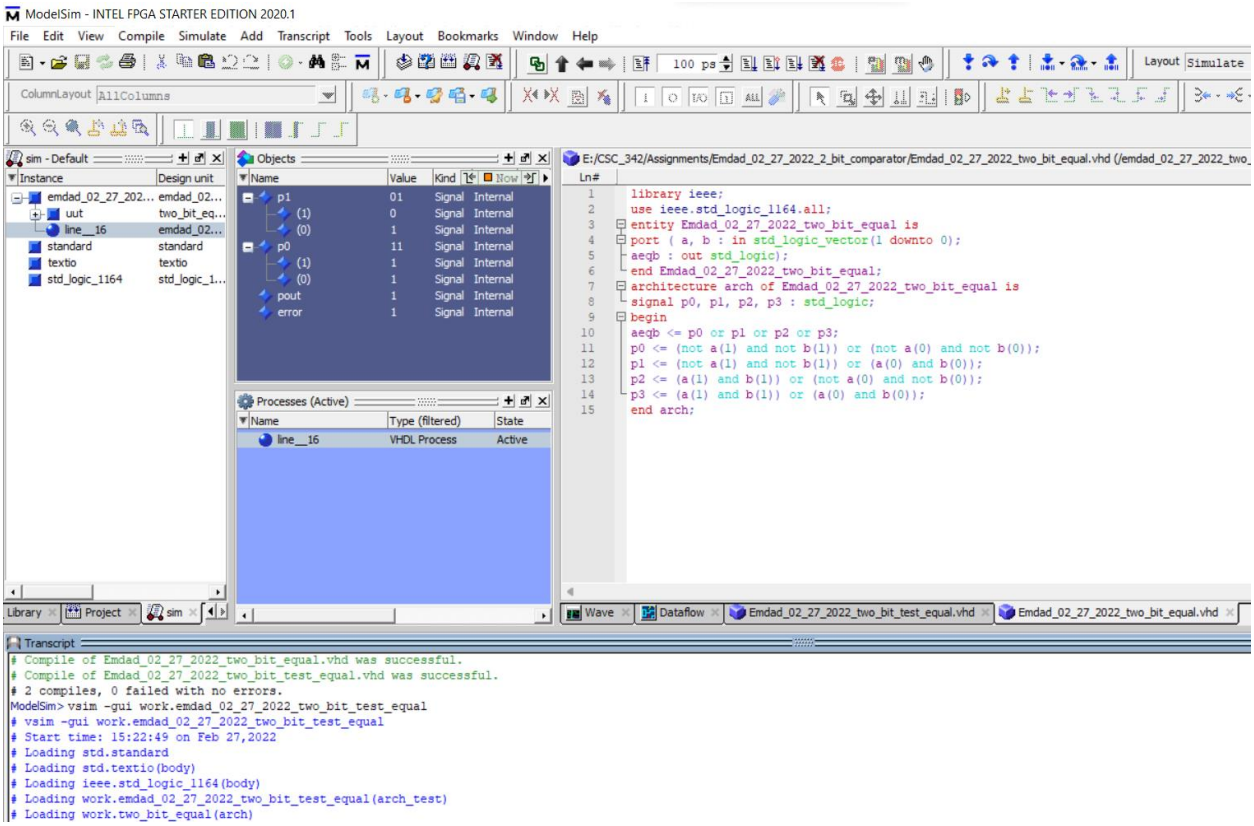

Figure 8: 2-bit comparator compiled successful

In figure 9, we can see the test-bench which got modified from the tutorial for the 2-bit comparator code.

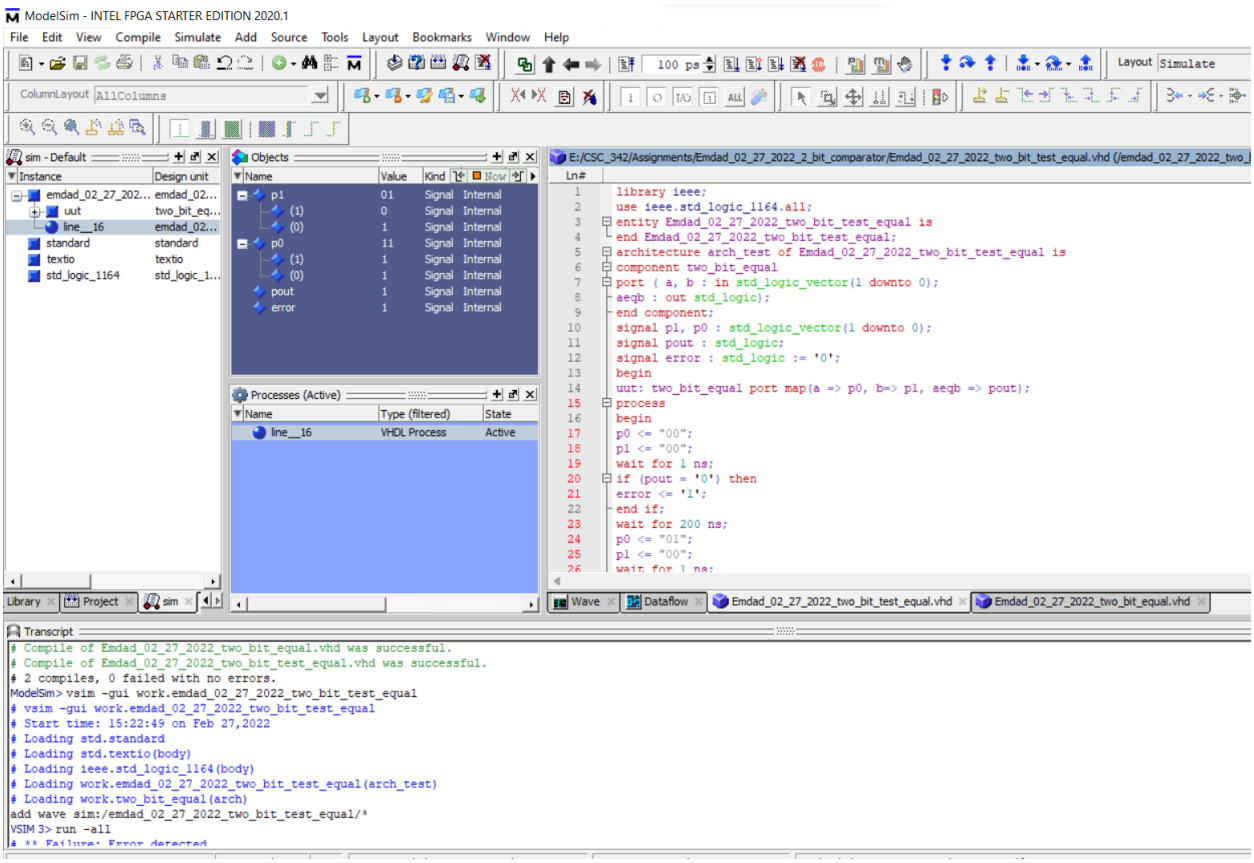

Figure 9: 2-bit comparator code compilation

In figure 10, we can see the simulation output and tested the code good.

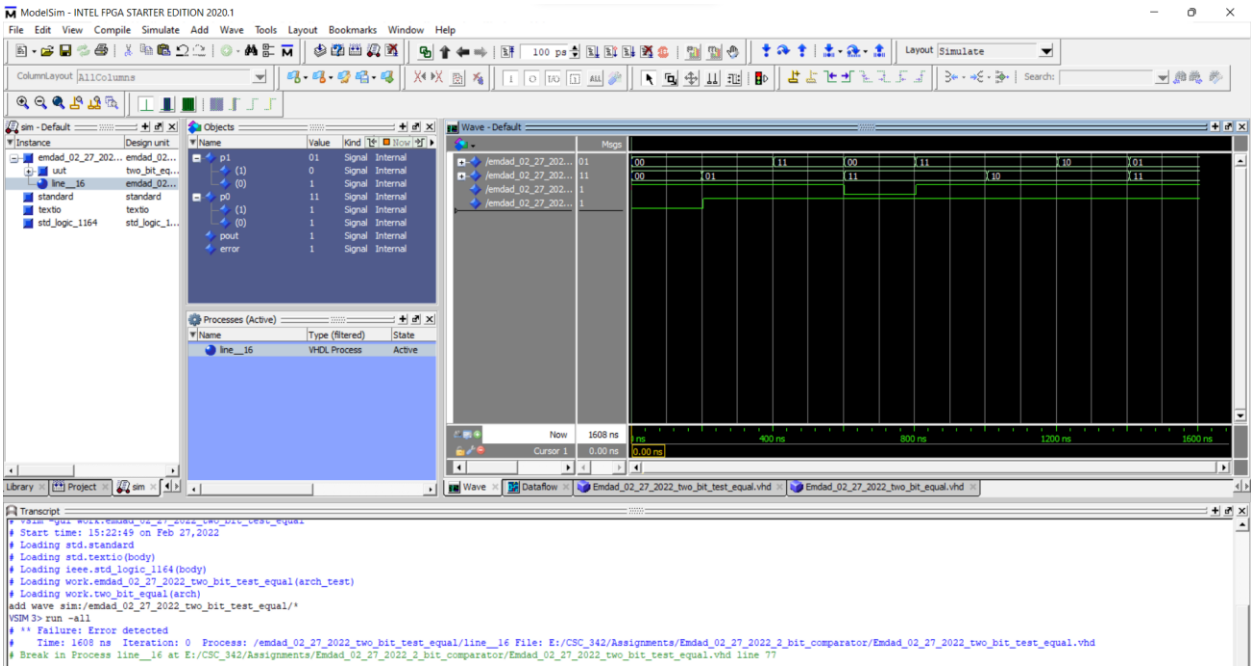

Figure 10: Simulation Output

In figure 11, we can see that the code got modified and 1-bit comparator component was being used.

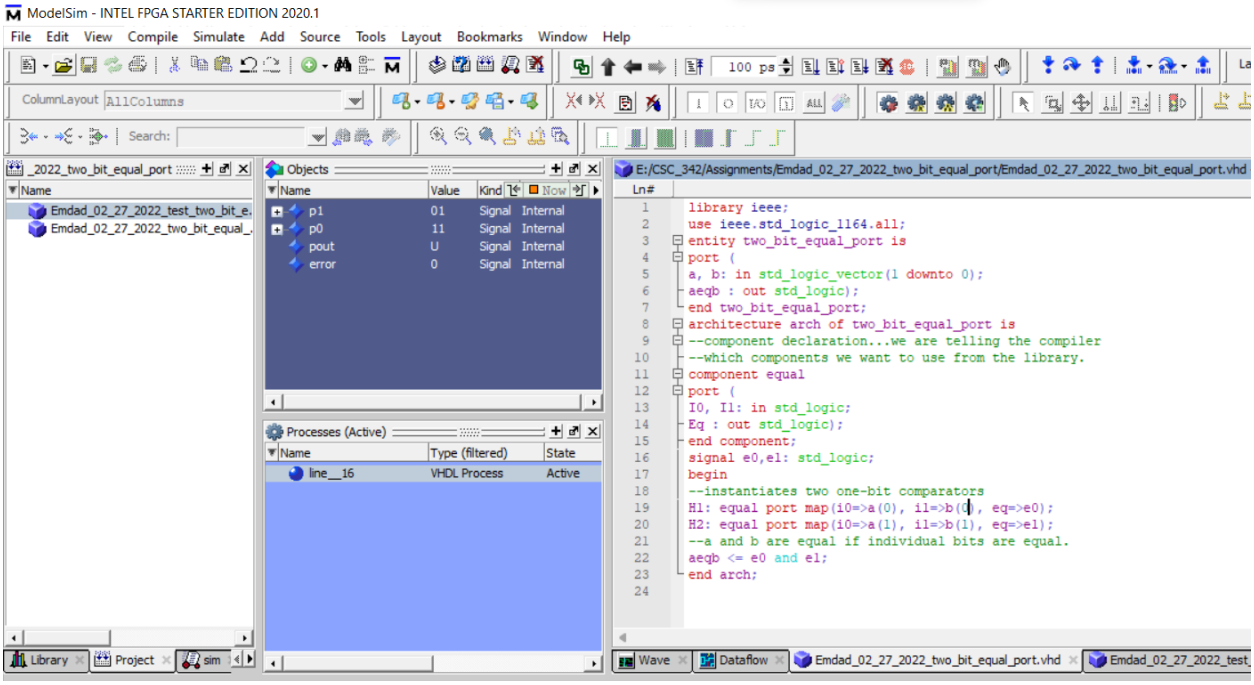

Figure 11: 2-bit comparator port

In figure 12, we can see the testbench for 2-bit comparator port.

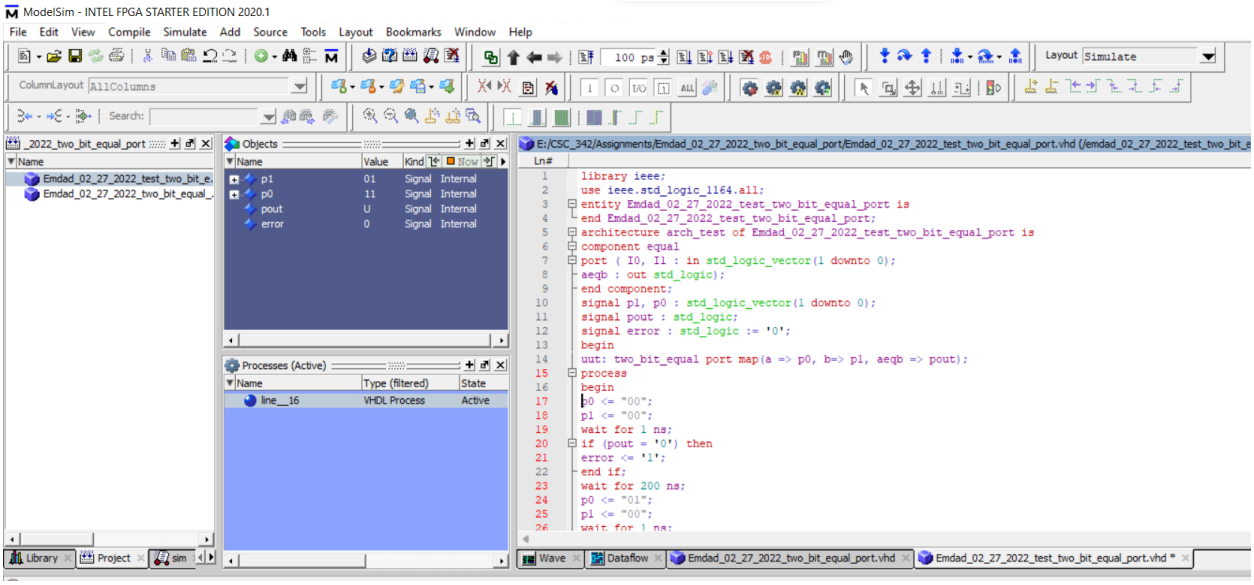

Figure 12: testbench for 2-bit comparator port.

In figure 13, we can see that I reduced the code line to optimize the code which ran successfully.

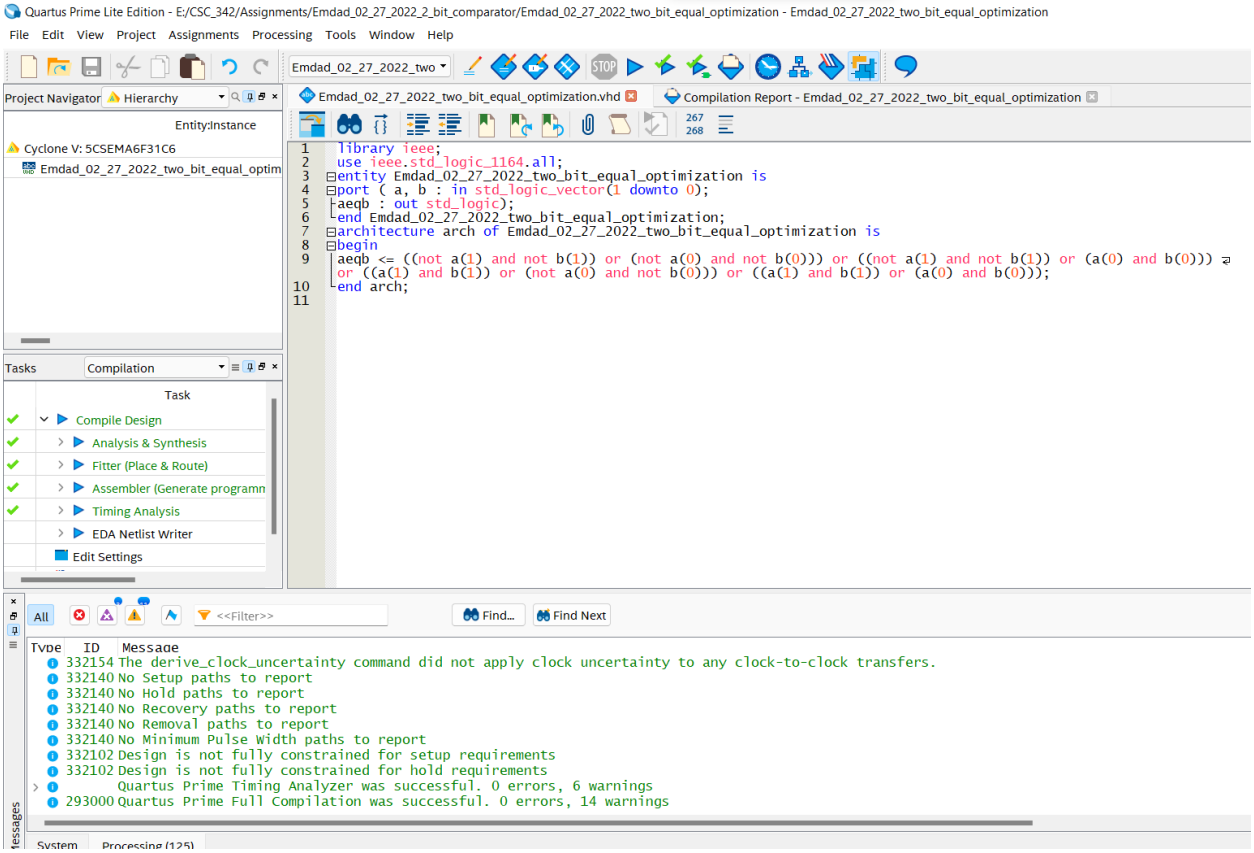

Figure 13: Optimized code

III. 8-bits comparator normal, port & optimization

In figure 14, we can see the 8-bits comparator code and compiled succesfully.

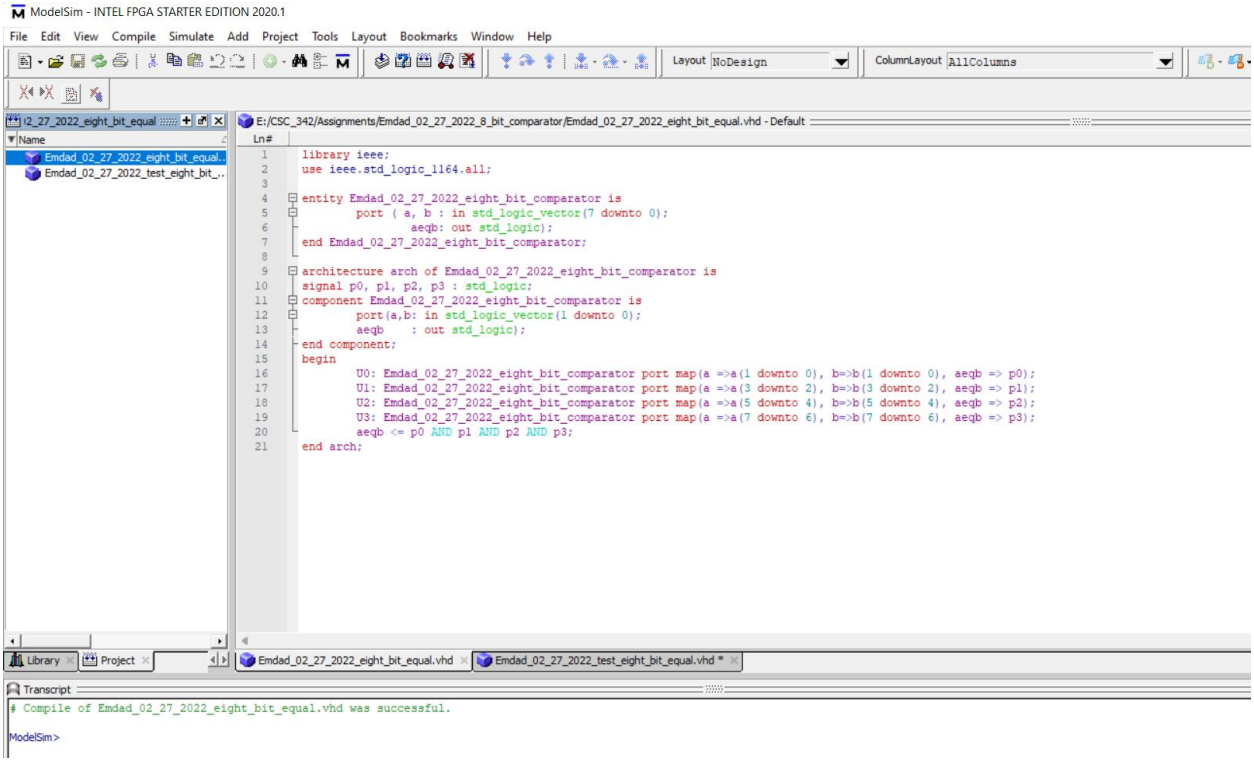

Figure 14: 8-bits comparator code

In figure 15, we can see the modified for the 8-bits comparator testbench to get correct simulation.

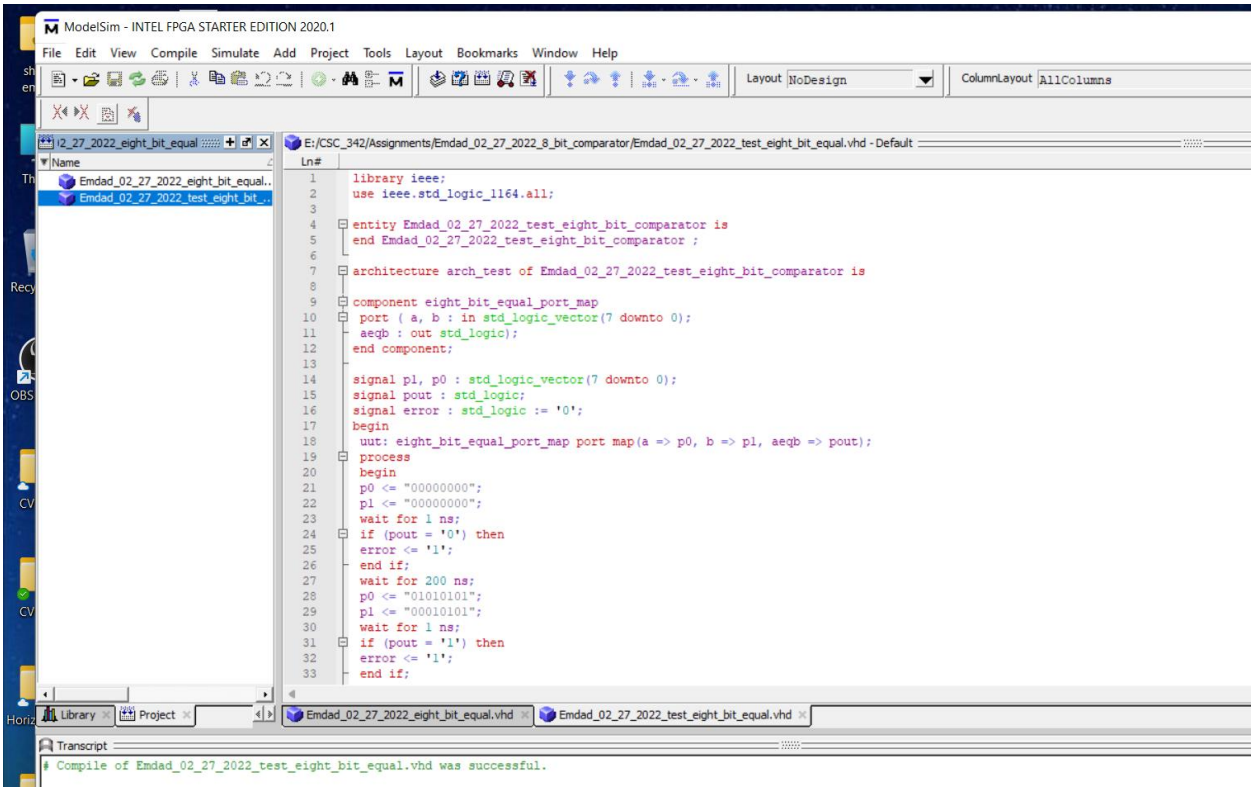

Figure 15: Testbench for 8-bit comparator

In figure 16, we can see the final output for the 8-bits comparator with the simulation.

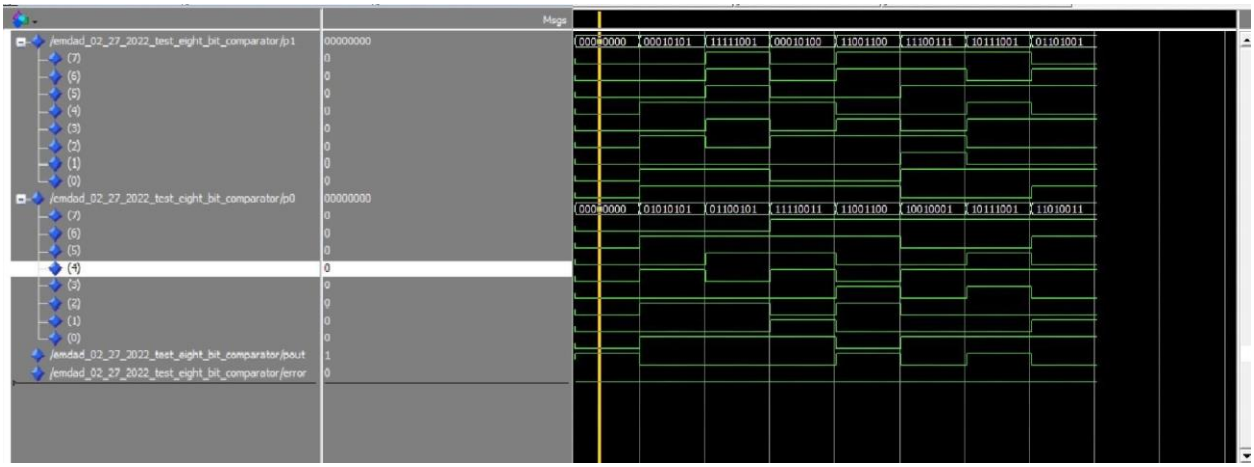

Figure 16: Simulation Output for 8-bits comparator

In figure 17, I optimized the code by lessing the code line and it compiled successfully.

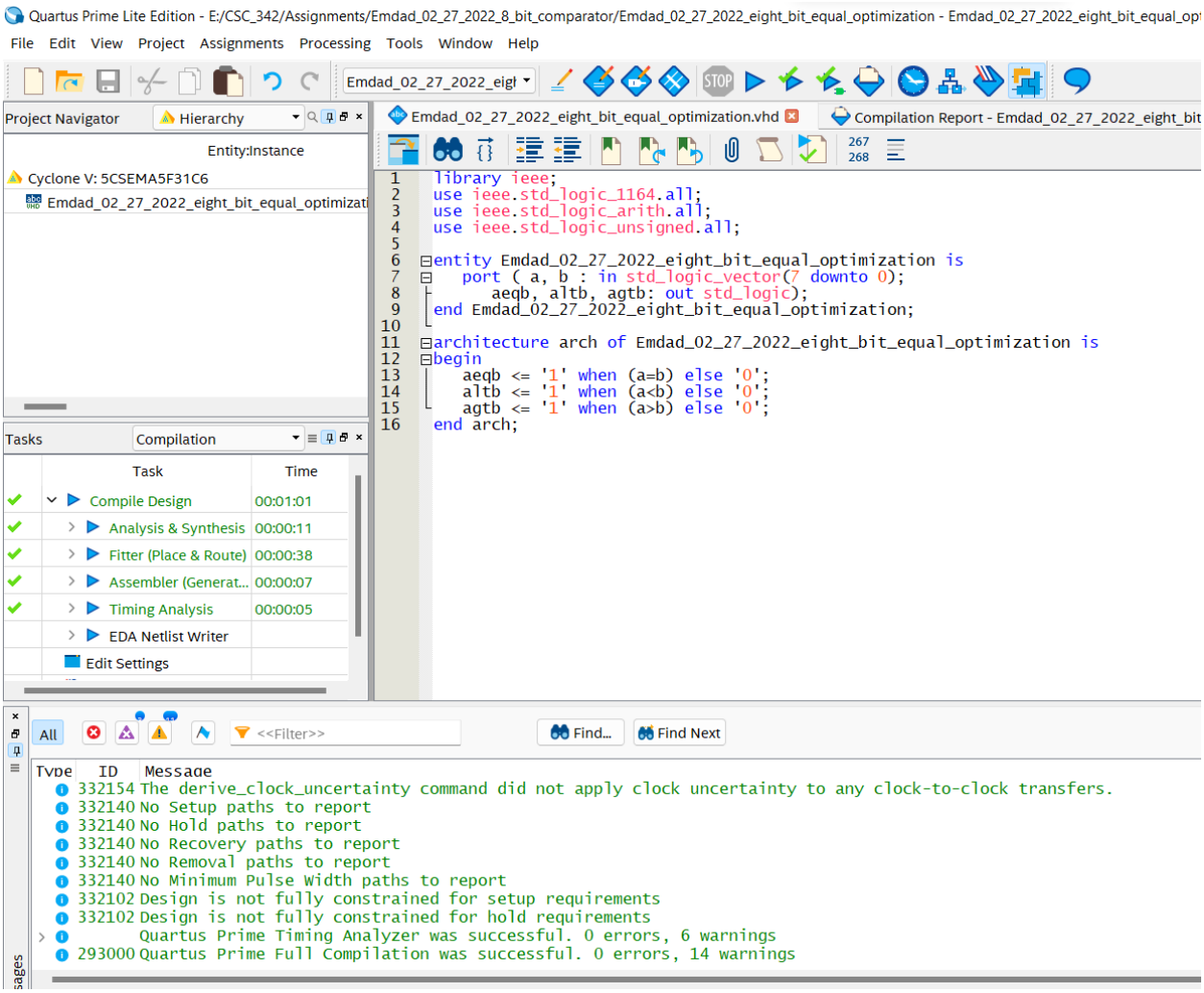

Figure 17: Optimized code for 8-bits comparator

Explanation:

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

For 1-bit comparator, I used the above truth table to do the actual code and then used the testbench the to check my simulation and then reviewed the simulation's output with the above truth table and it matched which tested my code good.

The truth table can be expressed by the following equation:

$$Eq = i0 \cdot i1 + i0' \cdot i1'$$

To optimize the code by removing lines 12-14, we need a Boolean expression. I used this above Boolean given in the assignment sheet and applied that in my code and then compiled and tested it and it gave me correct result.

Next, I moved to the 2-bits comparator. I used the same format from the 1-bit comparator. The half of the code was given by the professor and a Boolean expression was provided as well. I had to make my code from there. Below, we can see the expression.

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

We have to know some of the information before proceeding to the code from this expression:

- The ' indicate NOT and the . Indicate an AND.
- The + indicate an OR gate
- The a0 means the first bit of a
- The a1 means the second bit of a
- Same with the b0 and b1

So, for example, we will get something similar like this and it continues:

('a1.'b1) + (a0.b0) would be `(not a(1) and not b(1)) or (a(0) and b(0))`

After that, I moved to port maps part where I took help from google and modified the port information as needed from the assignment sheet. Then, I used the testbench to check the code and it compiled and simulation came out good successfully.

To optimize the code, I followed the Boolean expression and got rid of the 4(U0, U1, U2, U3) cases and it compiled successfully. The simulation output was same.

Finally, I moved to the 8-bits comparator. I used the 1bit comparator idea and implemented there. I had the 2 signals and implemented the code that. Then I moved to the port-maps and checked my code with simulation, and it ran correctly and successfully. I also optimized the code by change the code logic and reduced to 3 conditional statements.

Conclusion:

This assignment was helpful to learn as it was an introduction to VHDL, design Comparator, and how to test code. It was also an Introduction to ModelSim for circuit simulation and learn how to simulate comparator in ModelSim. I also learned how to test the circuits by using test bench files. The operations and condition logics were helpful to learn easily. I relearned and reviewed the concept again and which will help me in the course further.