

CS342/CS343
Instructor: Professor Izidor Gertner
Spring 2022
Comparators Lab, Azwad Shameem, 2/28/2022

Table of Contents

Objective	2
Code:	3
Task 0:	3
Task 1	9
Task 1a)	9
Task 1b	13
Task 2	14
Task 2a	14
Task 2b	17
Task A1	18
Task A2	22
2-bit Comparator	22
8-bit Comparator	25
Task A3	29
Explanation	31
Conclusion	32

Objective

The objective of this assignment is to create VHDL code in Quartus and test the correctness of the VHDL code with waveforms in ModelSim. In order to accomplish this the assignment requires the design of 1 bit, 2 bit and 8 bit comparators using Quartus and the verification of correctness of each of comparators using the test bench file to create a waveform simulation in ModelSim. Furthermore, this assignment also intends for the practice of optimization of VHDL code in each of the 1 bit, 2 bit and 8 bit comparators by simplifying or reducing the lines of code of circuits that are already made and then testing that optimized code by verifying its correctness by utilizing the testbench's simulation waveform.

Code:

Task 0: Complete tutorial as listed below.

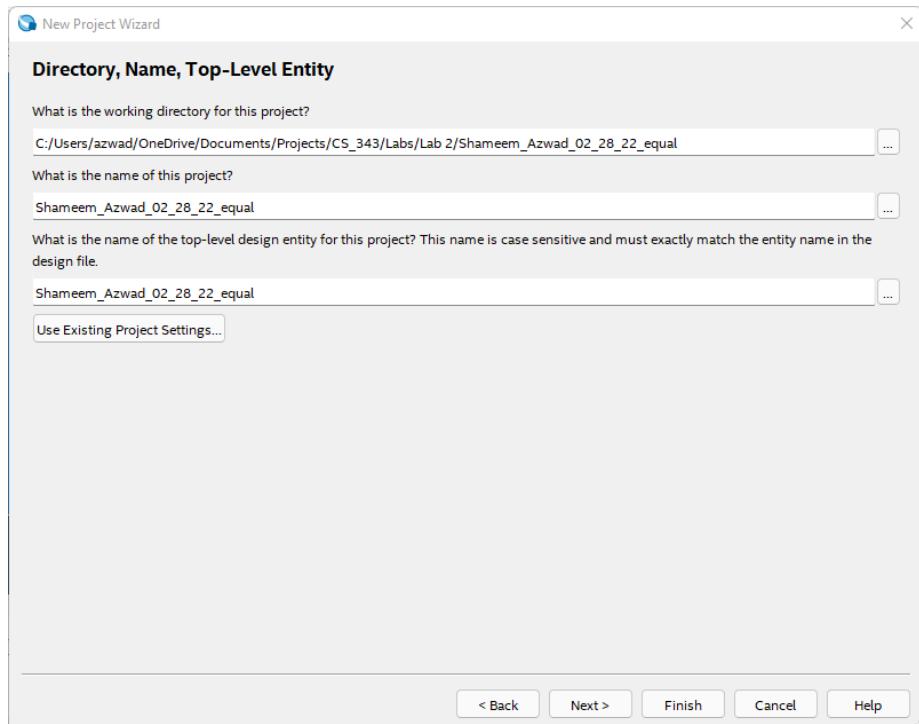


Figure 1: Creating project in Quartus

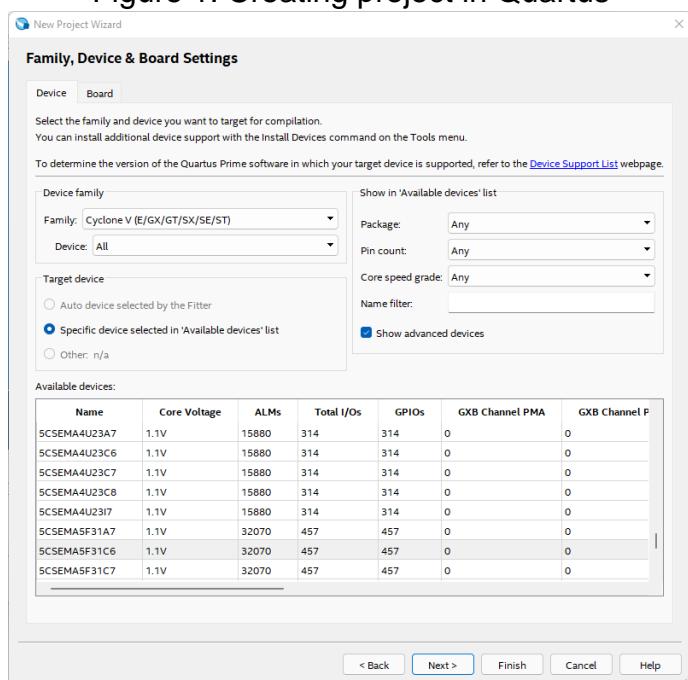


Figure 2: Device chosen

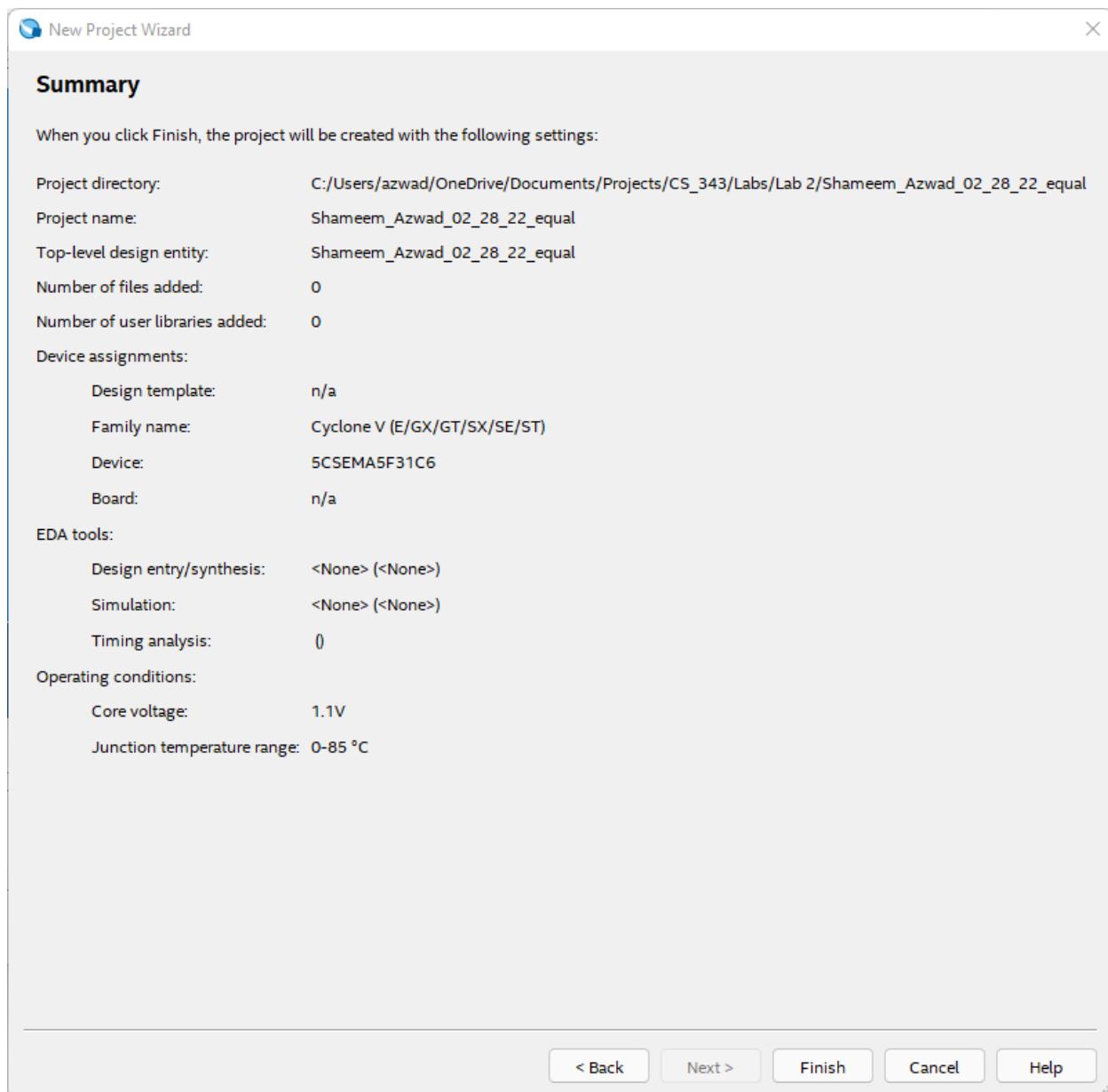


Figure 3: Project Summary

The screenshot shows the Quartus Prime Lite Edition interface. The main window displays the VHDL source code for a one-bit comparator:

```

library IEEE;
use ieee.std_logic_1164.all;

entity shameem_Azwad_02_28_22_equal is
    port (p0: in std_logic;
          p1: in std_logic;
          eq: out std_logic);
end entity shameem_Azwad_02_28_22_equal;

architecture arch of shameem_Azwad_02_28_22_equal is
begin
    eq <= p0 or p1;
    p0 <= (not p0) and (not p1);
    p1 <= p0 and p1;
end arch;

```

The Project Navigator pane shows the project structure under "Entity-Instance". The IP Catalog pane is visible on the right. The Tasks pane shows the compilation process, and the Messages pane shows the successful compilation report.

Figure 4: VHDL code for equal (one bit comparator)

The screenshot shows the Quartus Prime Lite Edition interface after compilation. The main window displays the "Compilation Report - Shameem_Azwad_02_28_22_equal" tab, which provides detailed statistics about the compilation process:

Category	Value
Flow Status	Successful - Fri Feb 25 16:10:09 2022
Quartus Prime Version	20.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	Shameem_Azwad_02_28_22_equal
Top-level Entity Name	Shameem_Azwad_02_28_22_equal
Family	Cyclone V
Timing Models	Final
Logic utilization (In ALMs)	1 / 32,070 (< 1 %)
Total registers	0
Total pins	3 / 457 (< 1 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSI RX PC56	0
Total HSI RX RX Deserializers	0
Total HSI TX PC56	0
Total HSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

The Project Navigator pane shows the project structure under "Entity-Instance". The IP Catalog pane is visible on the right. The Tasks pane shows the compilation process, and the Messages pane shows the successful compilation report.

Figure 5: VHDL code compiled successfully on Quartus

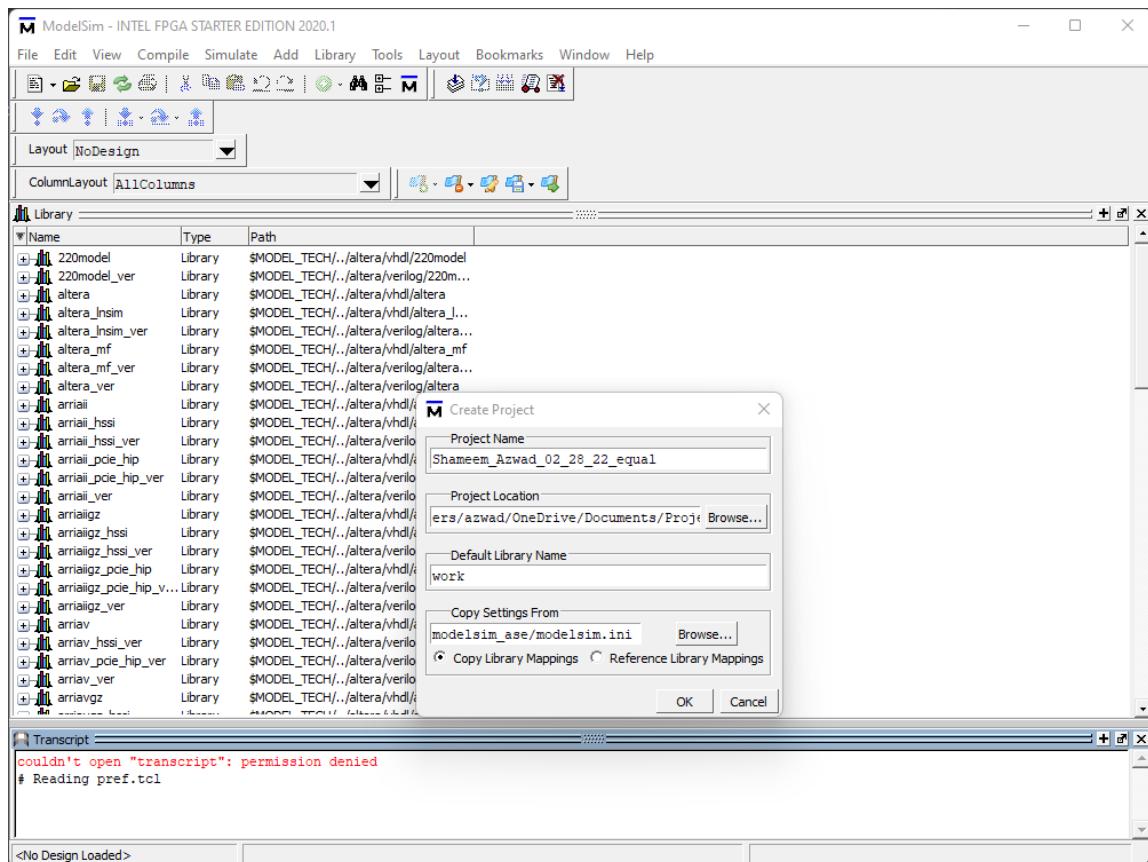


Figure 6: Project created in ModelSim

The screenshot shows the ModelSim interface with the 'Shameem_Azwad_02_28_22_equal' project loaded. The 'Library' tab is selected, showing the contents of the 'work' library. The 'Project' tab shows the project structure. The 'Transcript' tab displays the same permission denied error as in Figure 6. The main window displays the VHDL code for the entity:

```

library IEEE;
use ieee.std_logic_1164.all;

entity Shemeen_Azwad_02_28_22_equal is
    port (I0: in std_logic;
          I1: in std_logic;
          EQ: out std_logic);
end Shemeen_Azwad_02_28_22_equal;

architecture arch of Shemeen_Azwad_02_28_22_equal is
    signal p0: std_logic;
    signal p1: std_logic;
begin
    EQ <= p0 or p1;
    p0 <= (not I0) and (not I1);
    p1 <= I0 and I1;
end arch;

```

The status bar at the bottom shows 'In: 17 Col: 9' and 'Project: Shameen_Azwad_02_28_22_equal <No Design Loaded> <No Context>'.

Figure 7: VHDL code in ModelSim

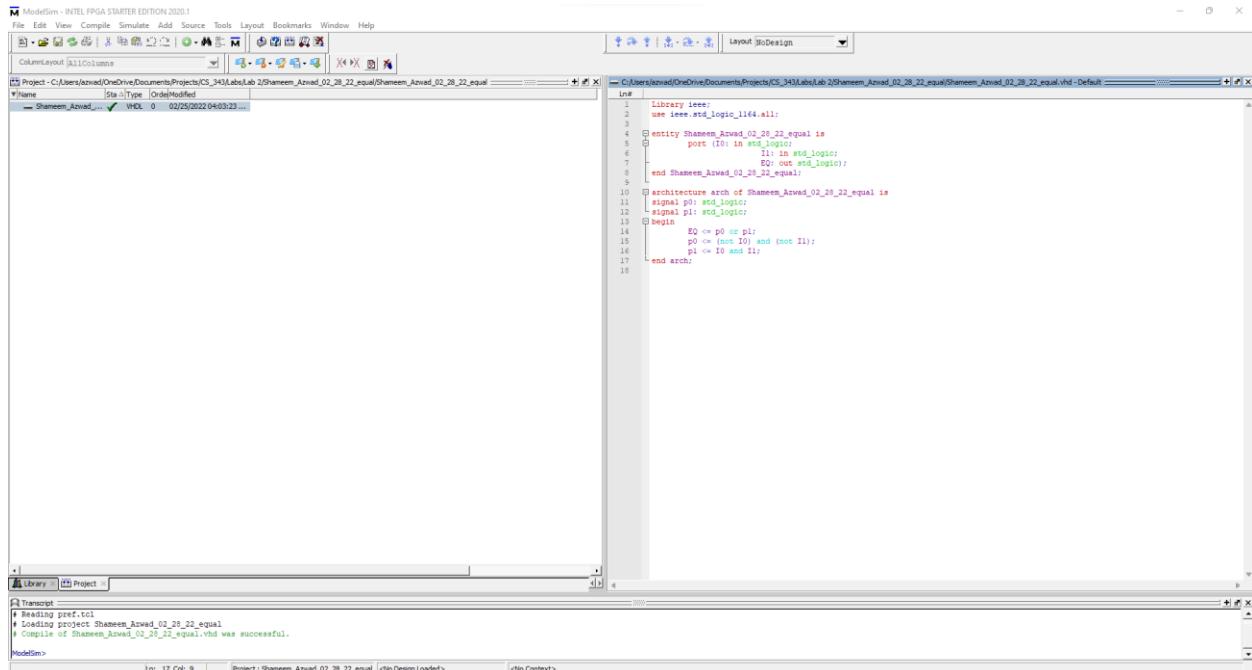


Figure 8: VHDL code compiled successfully in ModelSim.

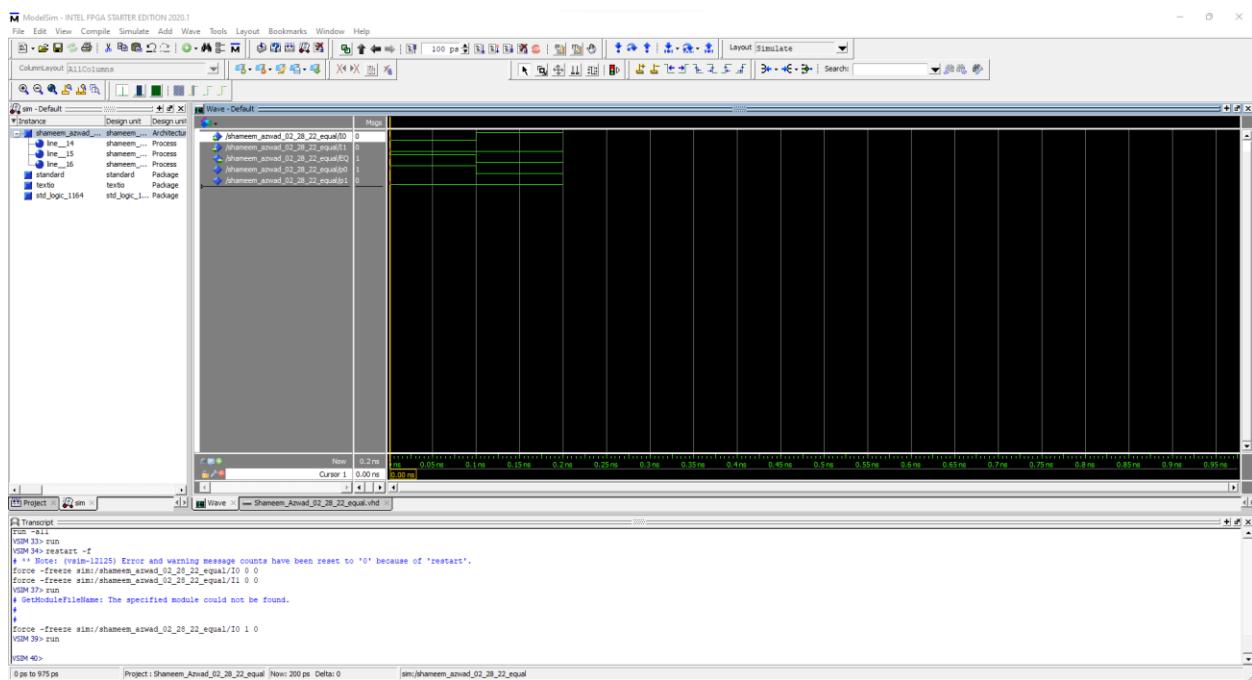


Figure 9: Simulation following tutorial of equal

The above image follows the tutorial in where we are using forced values for the inputs in order to test the circuit.

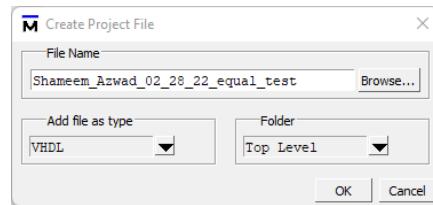


Figure 10: Creating test bench file

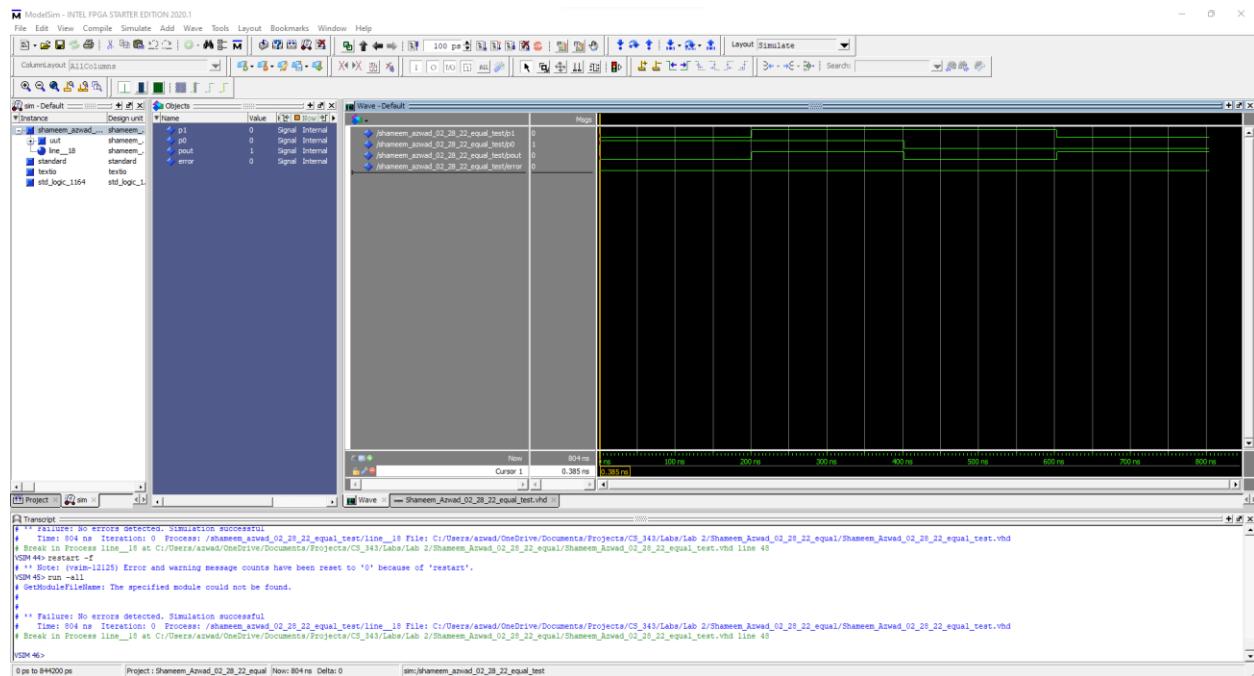


Figure 11: Simulation using the test bench

The above image utilizes the test bench in order to test the correctness of the circuit. In fact, the waveform follows the truth table of a one bit comparator which proves its correctness.

Task 1

Task 1a) write VHDL code for 2 bit comparator compile it

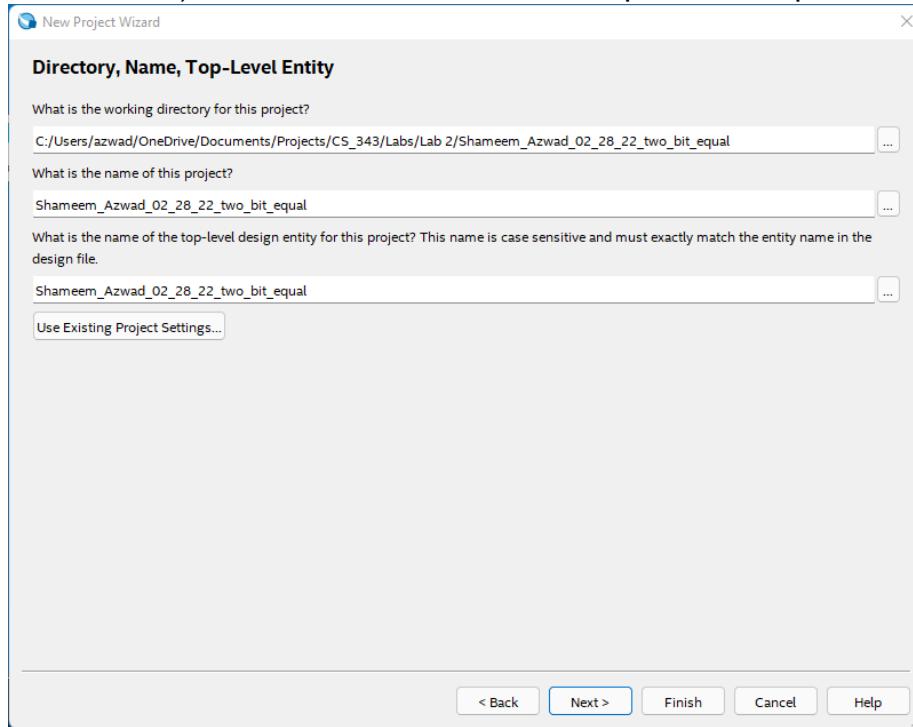


Figure 12: Two Bit Equal Project created

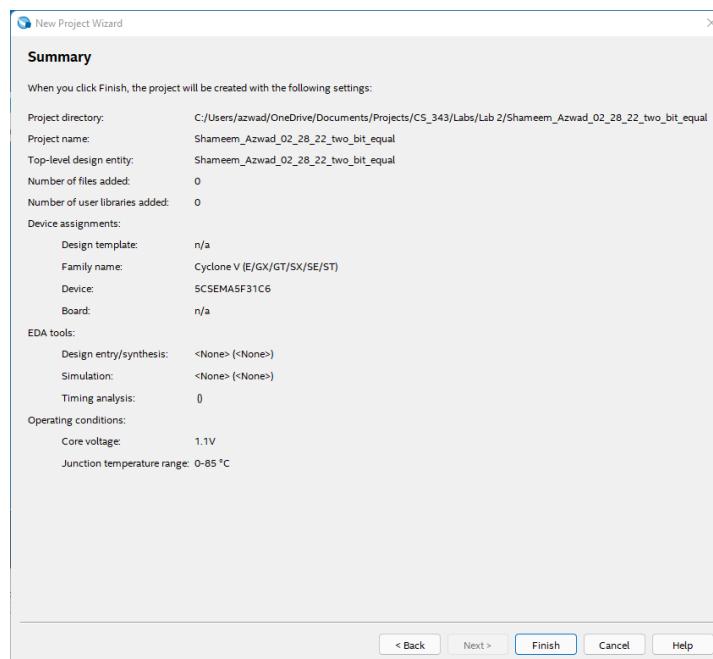


Figure 13: Project Summary for Two Bit Equal

The screenshot shows the Quartus Prime Lite Edition interface with the following details:

- Title Bar:** Quartus Prime Lite Edition - C:/Users/azwad/OneDrive/Documents/Projects/CS_343/Labs/Lab 2/Shameem_Azwad_02_28_22_two_bit_equal/Shameem_Azwad_02_28_22_two_bit_equal - Shameem_Azwad_02_28_22_two_bit_equal
- Project Navigator:** Shows an EntityInstance named "shameem_Azwad_02_28_22_two_bit_equal" under "Cyclone V: SCSEMASF31C6".
- Editor Area:** Displays the VHDL code for the entity "Shameem_Azwad_02_28_22_two_bit_equal.vhd". The code defines an entity with two inputs (a, b) and one output (weqb). It contains a single architecture block with four output assignments (p0, p1, p2, p3) based on the logic levels of the inputs.

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3
4  entity shameem_Azwad_02_28_22_two_bit_equal is
5    port (a : in STD_LOGIC_VECTOR(1 DOWNTO 0);
6          b : in STD_LOGIC_VECTOR(1 DOWNTO 0);
7          weqb : out STD_LOGIC);
8  end shameem_Azwad_02_28_22_two_bit_equal;
9
10 architecture arch of shameem_Azwad_02_28_22_two_bit_equal is
11 begin
12   process (a, b)
13   begin
14     p0 <= (NOT A(1) AND NOT B(1)) AND (NOT A(0) AND NOT B(0));
15     p1 <= (NOT A(1) AND NOT B(1)) AND (A(0) AND B(0));
16     p2 <= (A(1) AND NOT B(1)) AND (NOT A(0) AND NOT B(0));
17     p3 <= (A(1) AND B(1)) AND (A(0) AND B(0));
18   end process;
19 end arch;

```

- Tasks Panel:** Shows a list of tasks: Compile Design, Analysis & Synthesis, Fitter (Place & Route), Assembler (Generate program), Timing Analysis, and EDA Netlist Writer. The first task is marked as completed.
- Messages Panel:** Shows a table with columns Type, ID, and Message. It displays 125 processing messages.
- IP Catalog:** Shows sections for Installed IP, Project Directory, and Library, with various options like Basic Functions, DSP, Interface Protocols, etc.

Figure 14: VHDL code for two bit equal

The equation used to deduce the logic for p0, p1, p2, p3 was

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

This equation allows us to see that each plus sign is an OR so p0, p1, p2, p3 have to be the values in between the plus signs. In addition, we know that the ' sign means NOT and the . sign must be an AND.

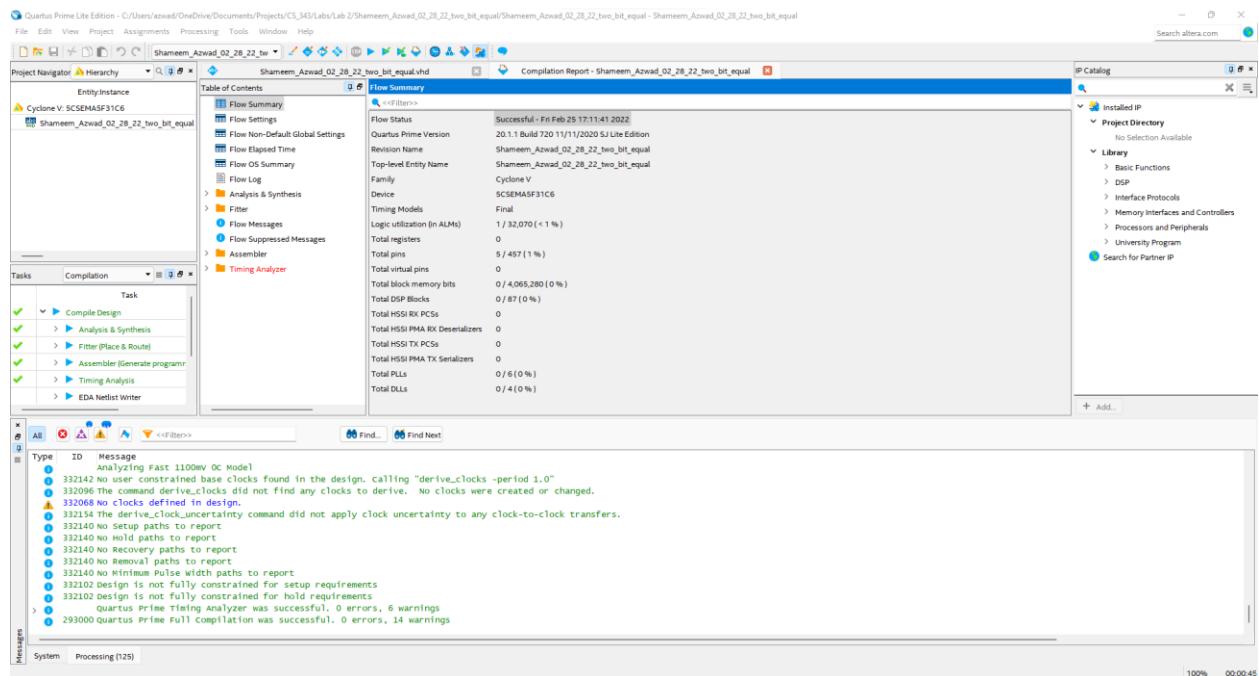


Figure 15: VHDL code compiled successfully in Quartus

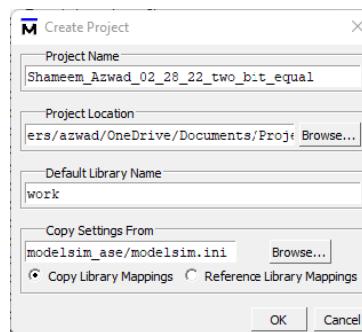


Figure 16: Two bit equal Project created on ModelSim

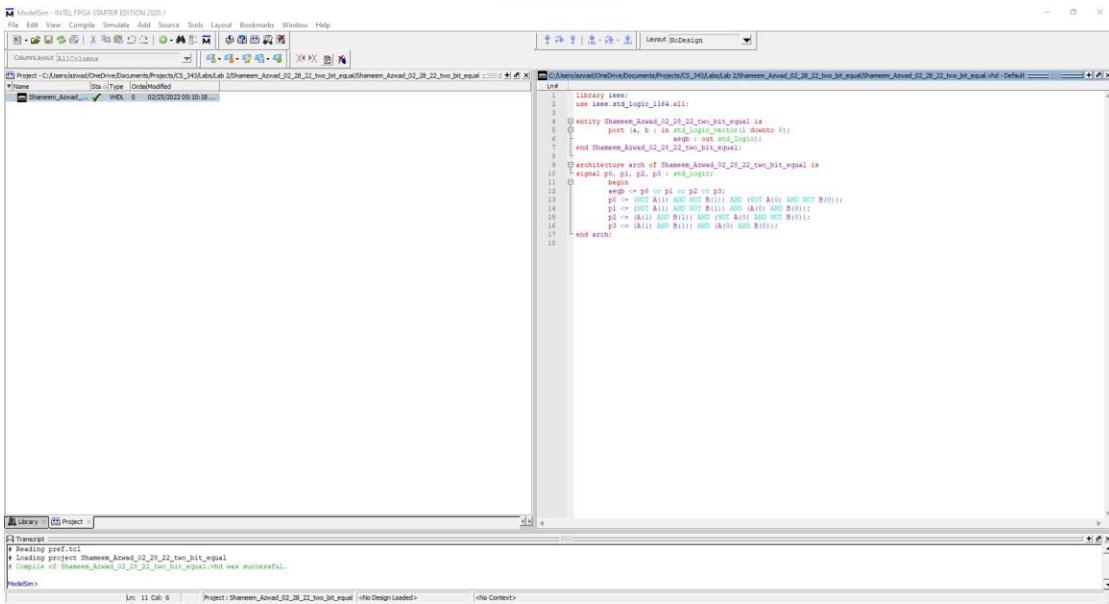


Figure 17: VHDL code compiled successfully on ModelSim

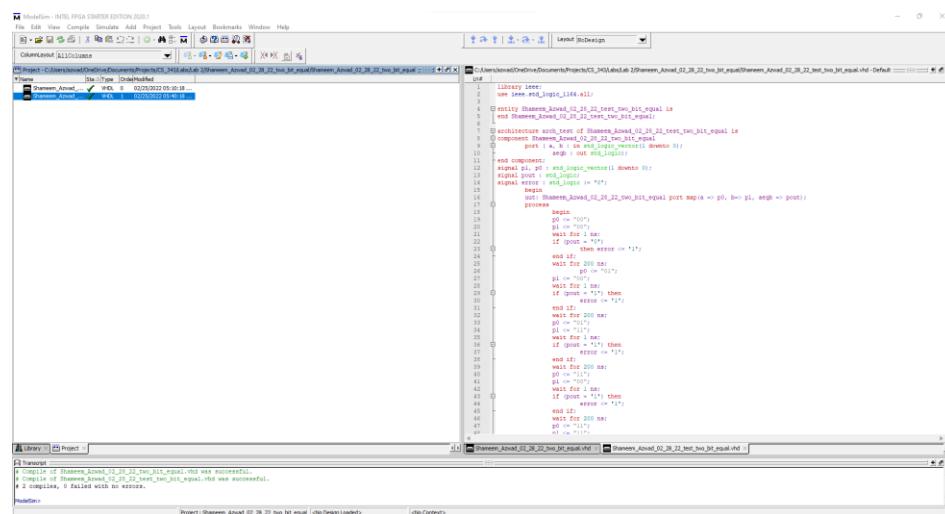


Figure 18: testbench code for two bit equal compiled

Task 1b) Verify correctness of 2 bit comparator using Model-SIM using tutorial.

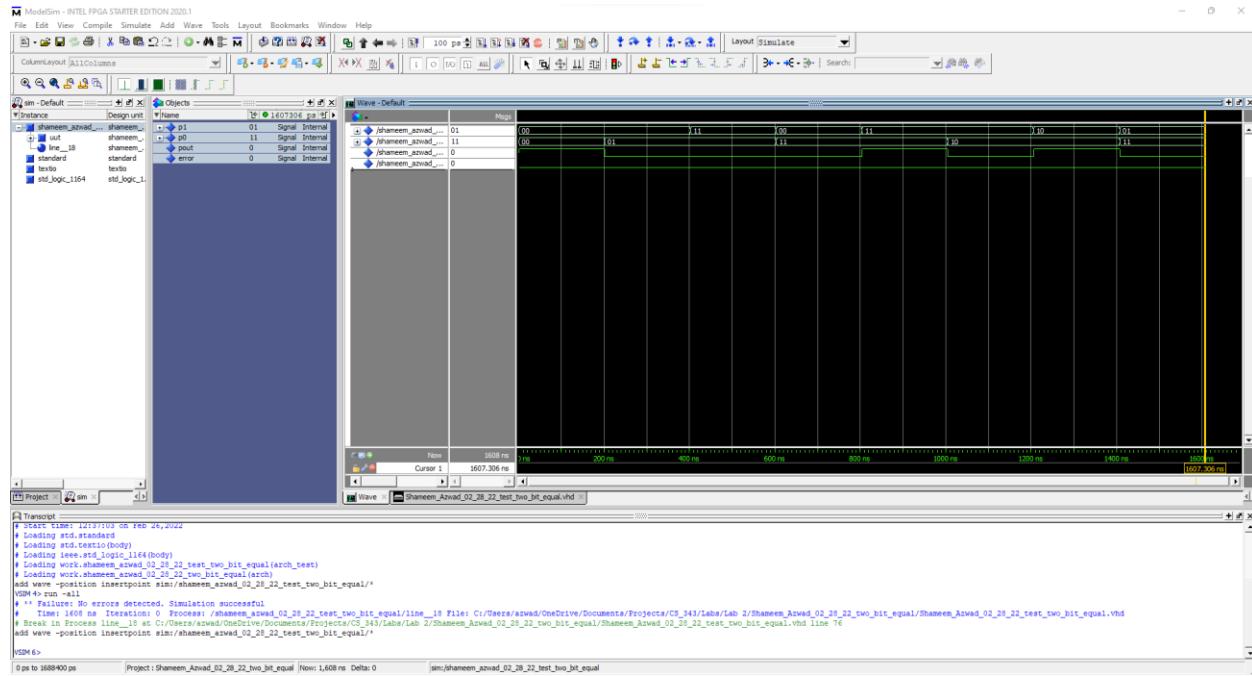


Figure 19: Simulation for two bit equal

This waveform shows the correctness of the two bit equal.

Task 2

Task 2a) write VHDL code for 8 bit comparator compile it

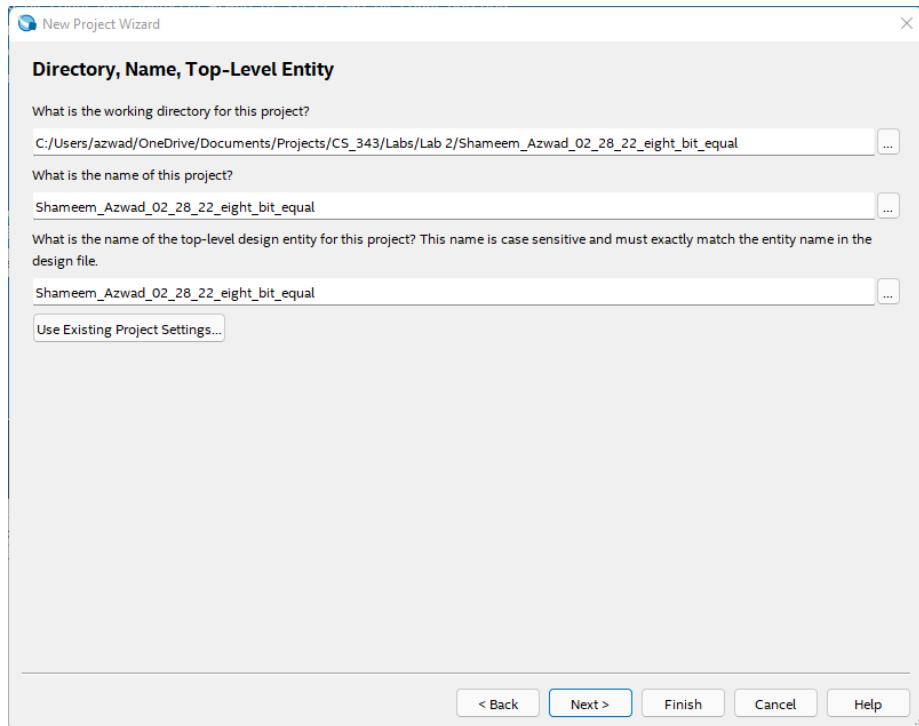


Figure 20: 8 bit comparator project creation

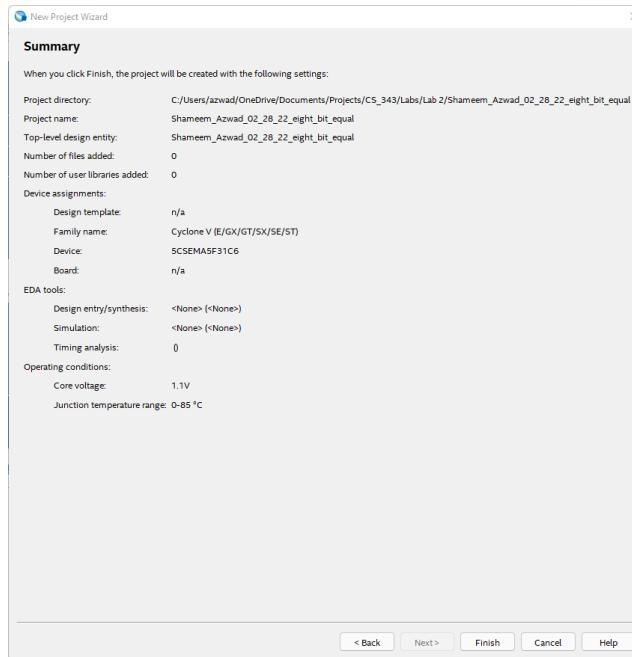


Figure 21: Project Summary

```

library IEEE;
use ieee.std_logic_1164.all;

entity cmp is
    port(a : in STD_LOGIC_VECTOR(7 downto 0);
         b : in STD_LOGIC_VECTOR(7 downto 0);
         aeqb : out STD_LOGIC);
end cmp;

architecture arch of cmp is
begin
    aeqb <= eq1 AND eq2 AND eq3 AND eq4 AND eq5 AND eq6 AND eq7;
end arch;

```

Figure 22: VHDL Code for the 8 bit comparator

The 8 bit comparator utilizes eight 1 bit comparators and ANDs them together.

The screenshot shows the Quartus Prime Lite Edition interface. The top menu bar includes File, Edit, View, Project, Assignments, Processing, Tools, Window, and Help. The main window displays the VHDL code for a 1-bit comparator:

```

library IEEE;
use IEEE.STD.TEXT.all;
use IEEE.STD.LOGIC.all;

entity Shameem_Azad_02_28_22_eight_bit_equal is
    port (a: in STD_LOGIC;
          b: in STD_LOGIC;
          p0: out STD_LOGIC);
end Shameem_Azad_02_28_22_eight_bit_equal;

architecture arch of Shameem_Azad_02_28_22_eight_bit_equal is
begin
    process(a, b)
        begin
            p0 <= (not a) and (not b);
            p1 <= a and b;
    end process;
end arch;

```

The IP Catalog panel on the right lists various IP components. The bottom pane shows the Messages window, which is currently empty.

Figure 23: VHDL code for the 1 bit comparator that is used as a component

The screenshot shows the Quartus Prime Lite Edition interface with the compilation report open. The top menu bar includes File, Edit, View, Project, Assignments, Processing, Tools, Window, Help, and a Compilation tab. The main window displays the compilation report for the file "Shameem_Azad_02_28_22_eight_bit_equal.vhd".

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Filter
- Flow Messages
- Flow Suppressed Messages
- Assembler
- Timing Analyzer

Flow Summary

Parameter	Value
Flow Status	Successful - Sat Feb 26 18:43:31 2022
Quartus Prime Version	20.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	Shameem_Azad_02_28_22_eight_bit_equal
Top-level Entity Name	Shameem_Azad_02_28_22_eight_bit_equal
Family	Cyclone V
Device	SCSEMASF31C6
Timing Models	Final
Logic utilization (in ALMs)	4 / 32,070 (< 1 %)
Total registers	0
Total pins	17 / 457 (4 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Block	0 / 87 (0 %)
Total HSI5 RX PC5s	0
Total HSI5 PMA RX Deserializers	0
Total HSI5 TX PC5s	0
Total HSI5 PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Messages

The Messages window at the bottom shows the following log entries:

- Analyzing File "Shameem_Azad_02_28_22_eight_bit_equal.vhd"
- 332142 No setup constrained base clocks found in the design. Calling "derive_clocks" -period 1.0"
- 332096 The command derive_clocks did not find any clocks to derive. No clocks were created or changed.
- 332068 No clocks defined in design.
- 332154 The derive_clock_uncertainty command did not apply clock uncertainty to any clock-to-clock transfers.
- 332140 No setup paths to report
- 332140 No hold paths to report
- 332140 No removal paths to report
- 332140 No minimum pulse width paths to report
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
- 293000 quartus Prime Full Compilation was successful. 0 errors, 16 warnings

Figure 24: VHDL code successfully compiled on Quartus

Task 2b) Verify correctness of 8 bit comparator using Model-SIM

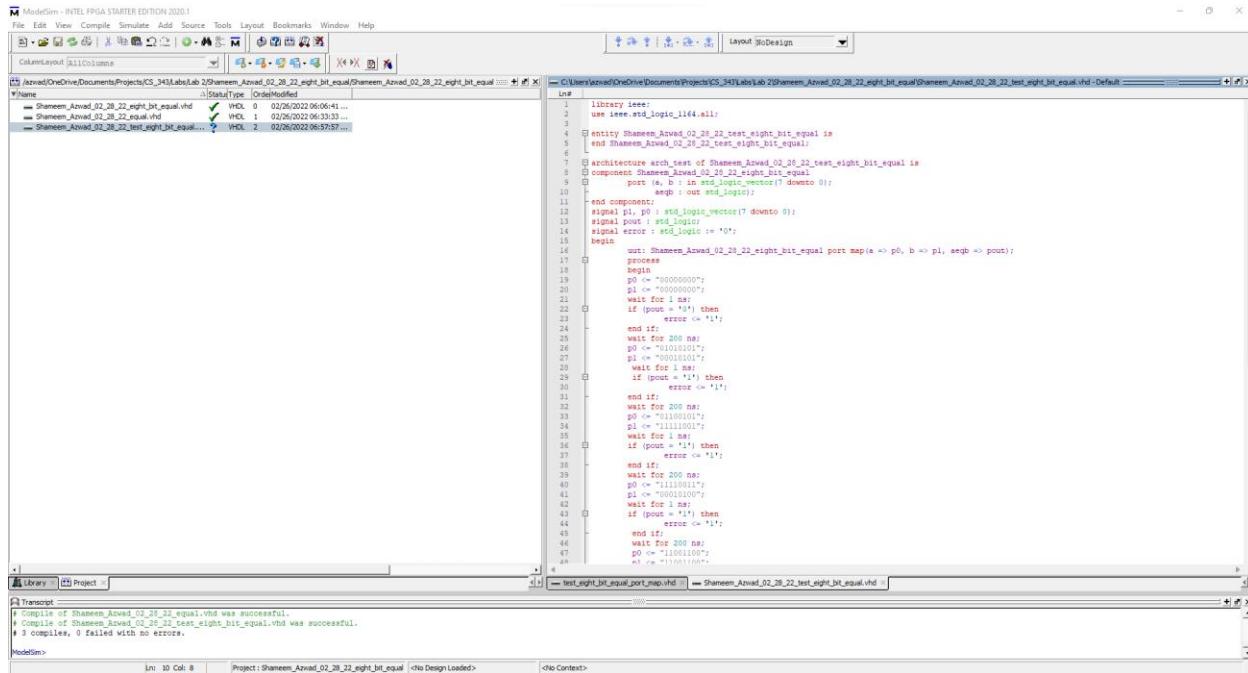


Figure 25: VHDL code successfully compiled on ModelSim

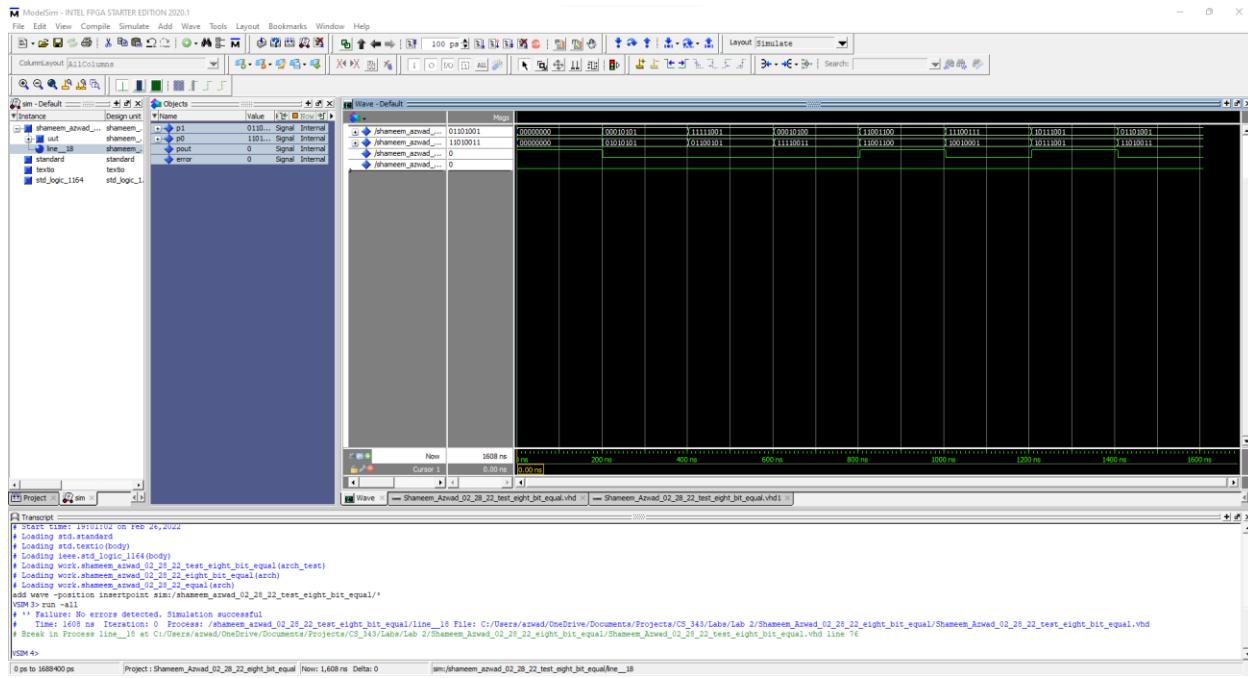


Figure 26: Eight bit comparator simulation waveform

Waveform correctly follows the outputs intended, which shows its correctness.

Task A1) Optimize the 1 bit comparator VHDL code shown on page 3, to replace lines 12,13,14 with ONE Boolean operation

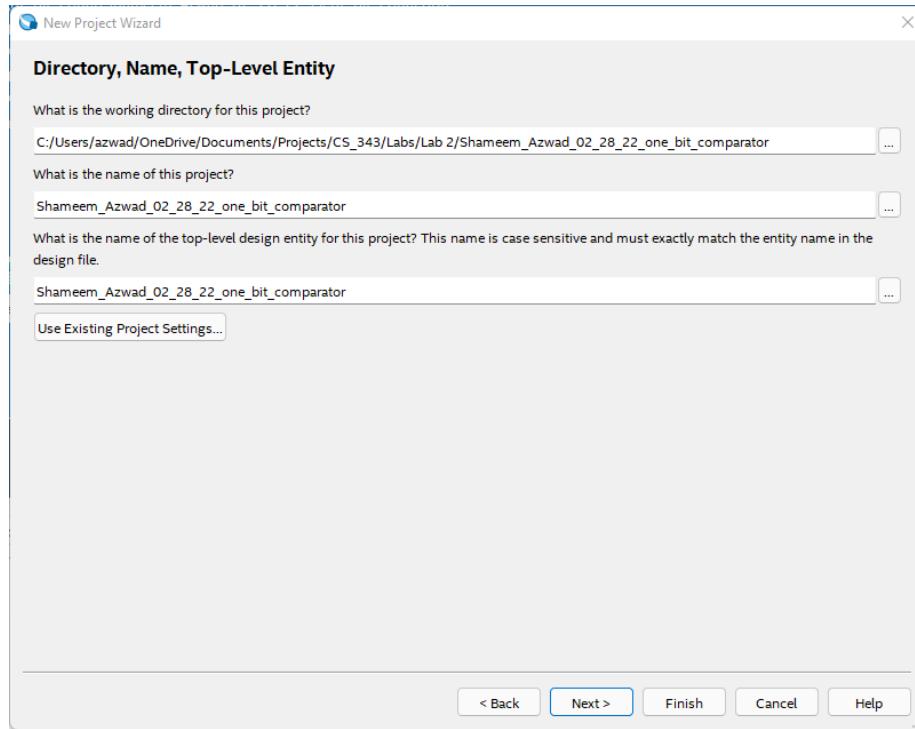


Figure 27: Creating new project for 1 bit comparator (this version is the optimized one)

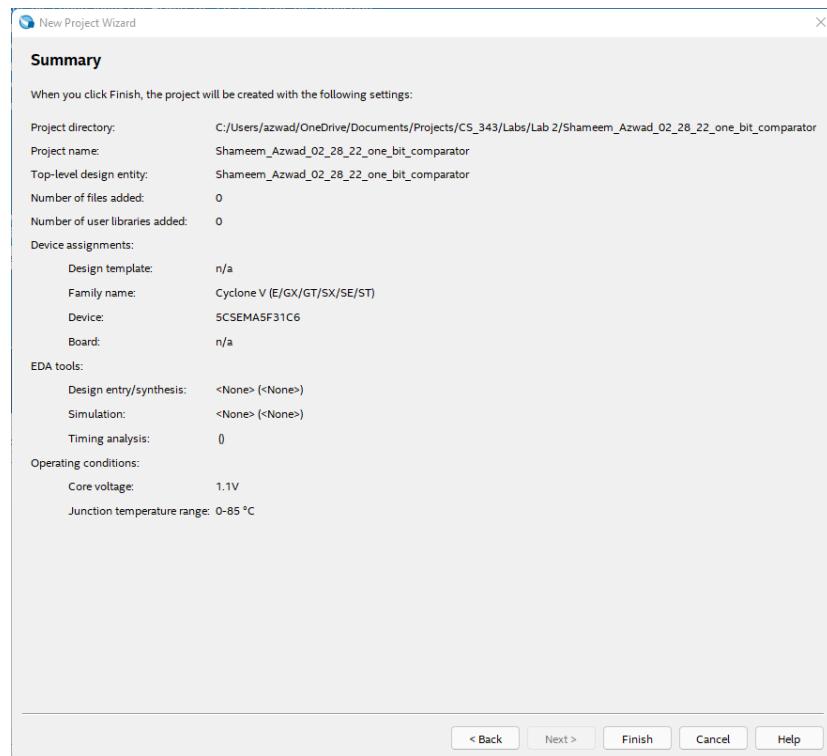


Figure 28: Project Summary

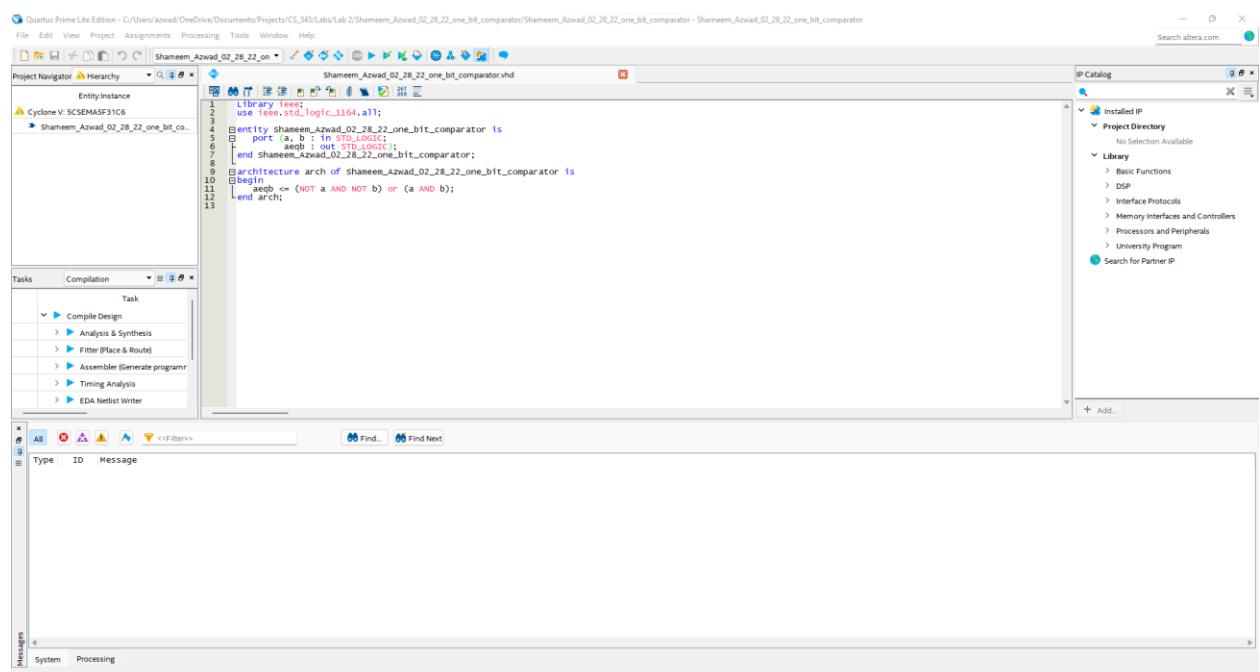


Figure 29: Optimized 1 bit comparator.

Replaced lines 12,13,14 with ONE Boolean operation.

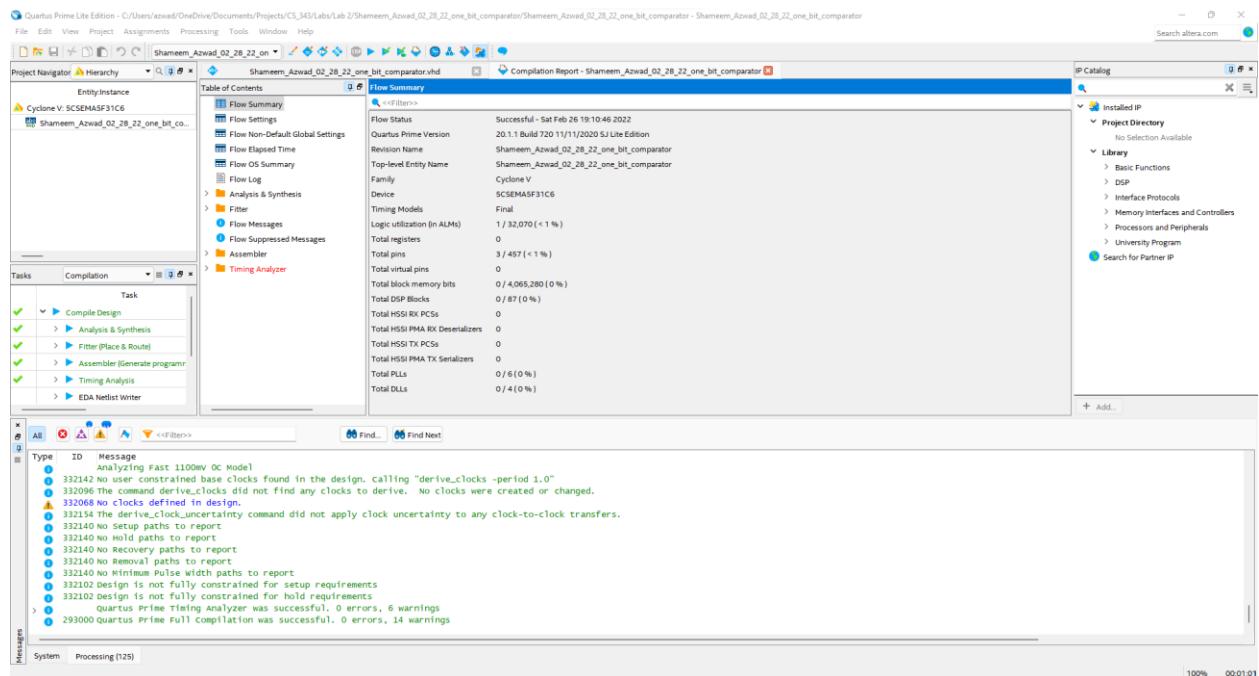


Figure 30: VHDL code compiled successfully on Quartus

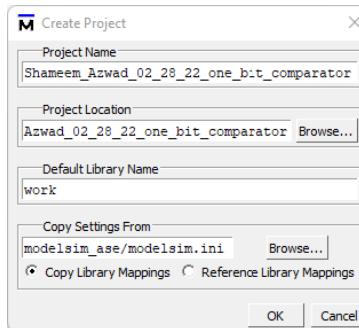


Figure 31: Project created on ModelSim

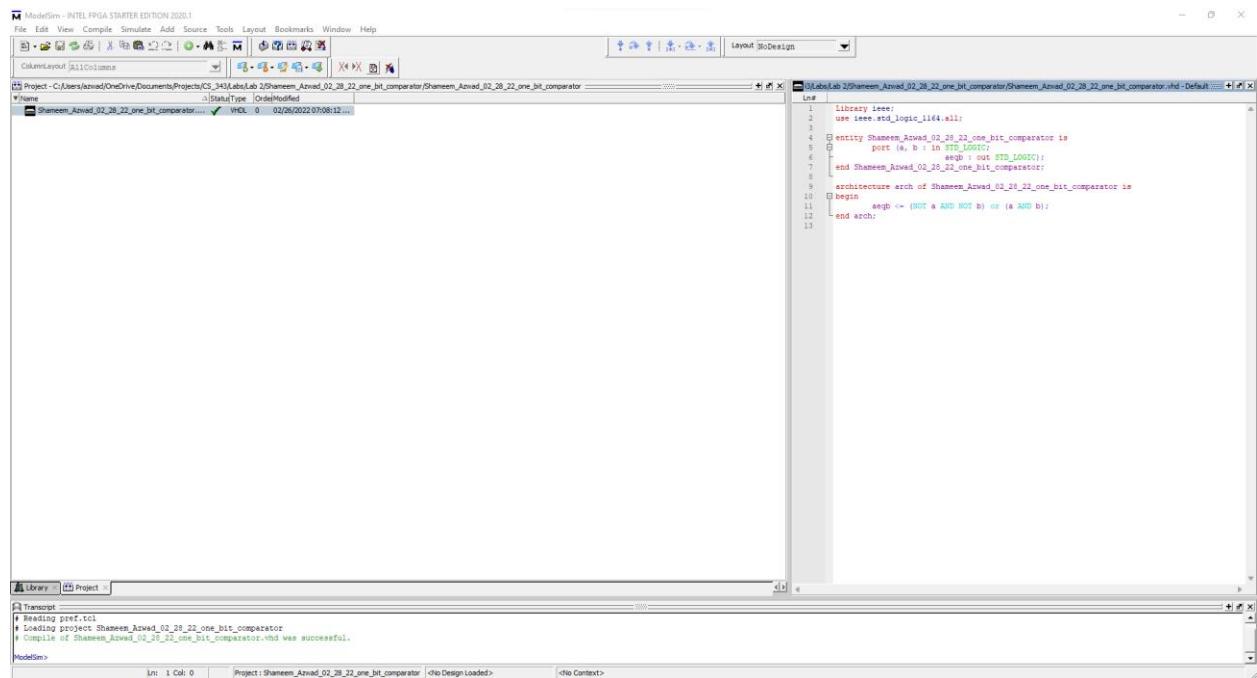


Figure 32: Project compiled successfully on ModelSim

Task A2) Optimize other comparators accordingly.

2-bit Comparator

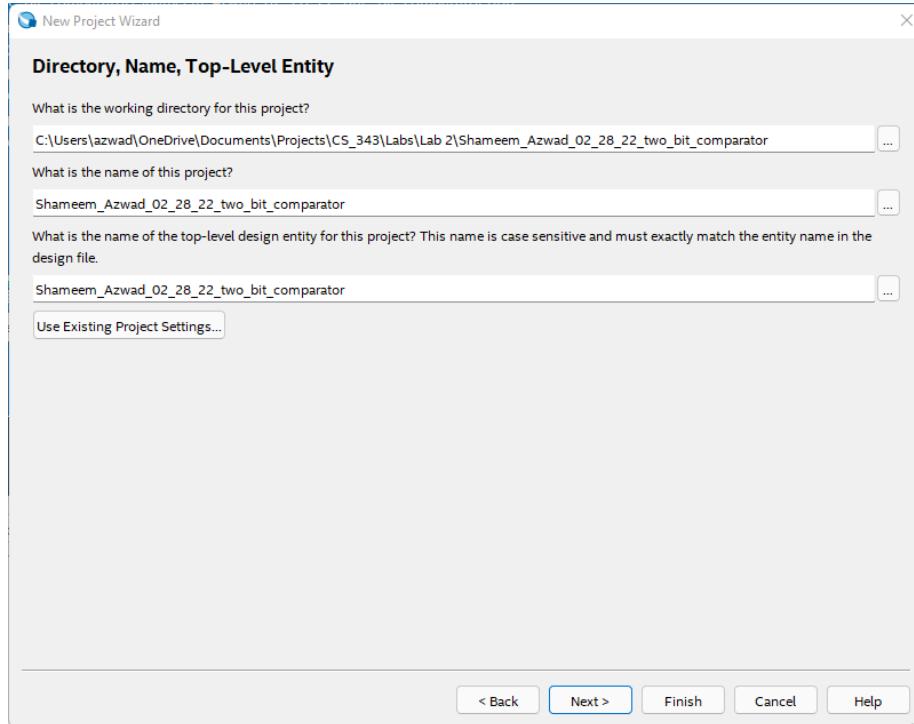


Figure 33: Optimized two-bit comparator created

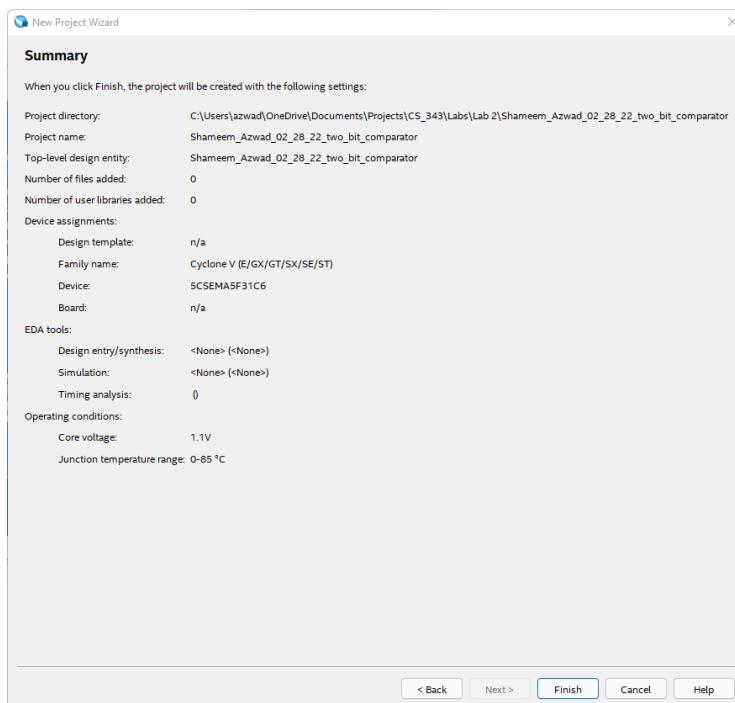


Figure 34: Project Summary

```

library IEEE;
use IEEE.STD.TEXTIO.all;
entity shameem_Azawad_02_28_22_two_bit_comparator is
    port (a, b : in STD_LOGIC_VECTOR(1 downto 0);
          result : out STD_LOGIC);
end entity;
architecture arch of shameem_Azawad_02_28_22_two_bit_comparator is
begin
    result <= ((NOT a(1) AND NOT b(1)) AND (NOT a(0) AND NOT b(0))) OR ((NOT a(1) AND NOT b(1)) AND (a(0) AND b(0))) OR ((a(1) AND b(1)) AND (NOT a(0) AND NOT b(0)));
end arch;

```

Figure 35: Optimized 2 bit comparator code.

Optimization is done by replacing multiple lines of Boolean operations with a one line Boolean expression.

Compilation Report - shameem_Azawad_02_28_22_two_bit_comparator

Flow Summary

Flow Settings	Successful - Sat Feb 26 19:43:06 2022
Flow Non-Default Global Settings	Quartus Prime Version 20.1 Build 720 11/11/2020 SJ Lite Edition
Flow Elapsed Time	Revision Name shameem_Azawad_02_28_22_two_bit_comparator
Flow OS Summary	Top-level Entity Name shameem_Azawad_02_28_22_two_bit_comparator
Flow Log	Family Cyclone V
Timing Models	Device SCSEMASF31C6
Logic Utilization (In ALMs)	Final 1 / 32,070 (< 1 %)
Total Registers	0
Total Pins	5 / 457 (1 %)
Total Virtual Pins	0
Total Block Memory Bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCIs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCIs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Messages

Type ID Message

- 332142 No recoverable constraint found in design. calling "derive_clocks -period 1.0"
- 332096 The command derive_clocks did not find any clocks to derive. No clocks were created or changed.
- 332068 No clocks defined in design.
- 332154 The derive_clock_uncertainty command did not apply clock uncertainty to any clock-to-clock transfers.
- 332140 No setup paths to report
- 332140 No hold paths to report
- 332140 No recovery paths to report
- 332140 No removal paths to report
- 332140 Minimal pulse width paths to report
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements

293000 quartus prime timing analyzer was successful. 0 errors, 6 warnings

System Processing (125)

Figure 36: Optimized VHDL code compiled successfully on Quartus.

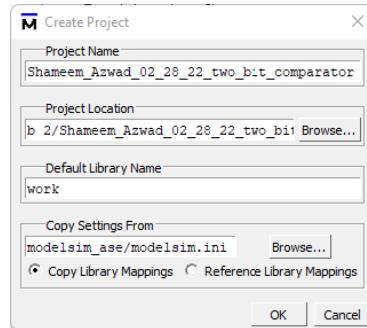


Figure 37: Project on ModelSim

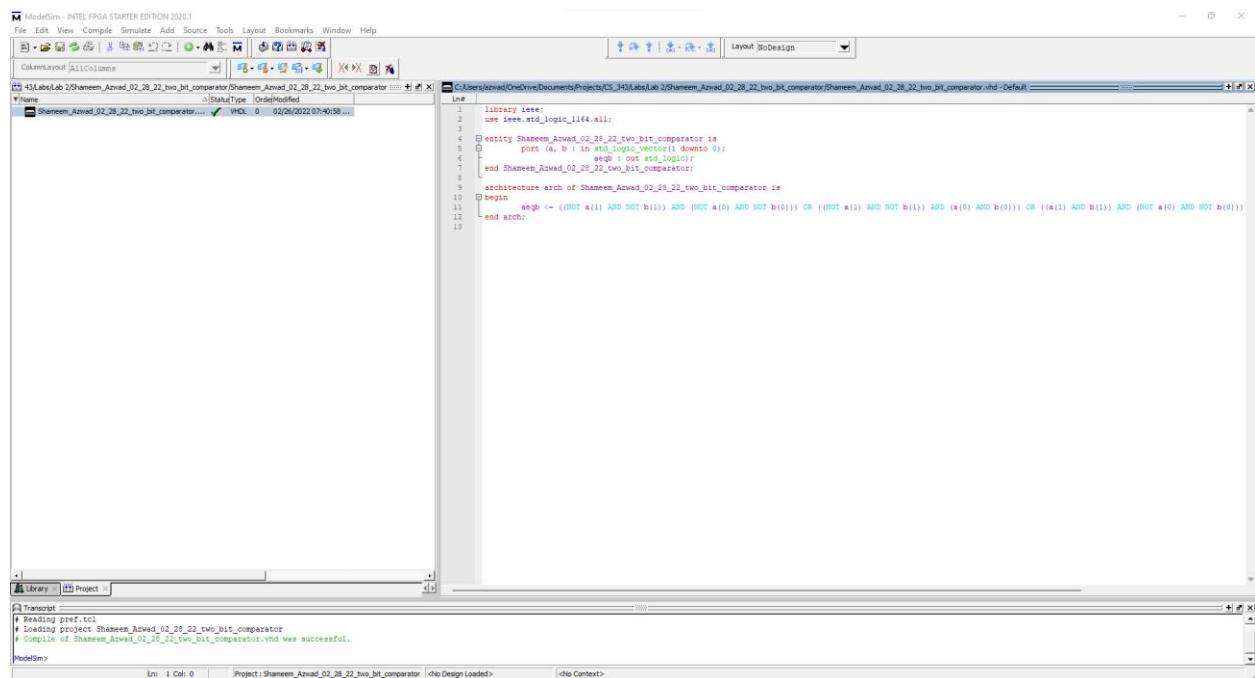


Figure 38: VHDL code compiled successfully on ModelSim

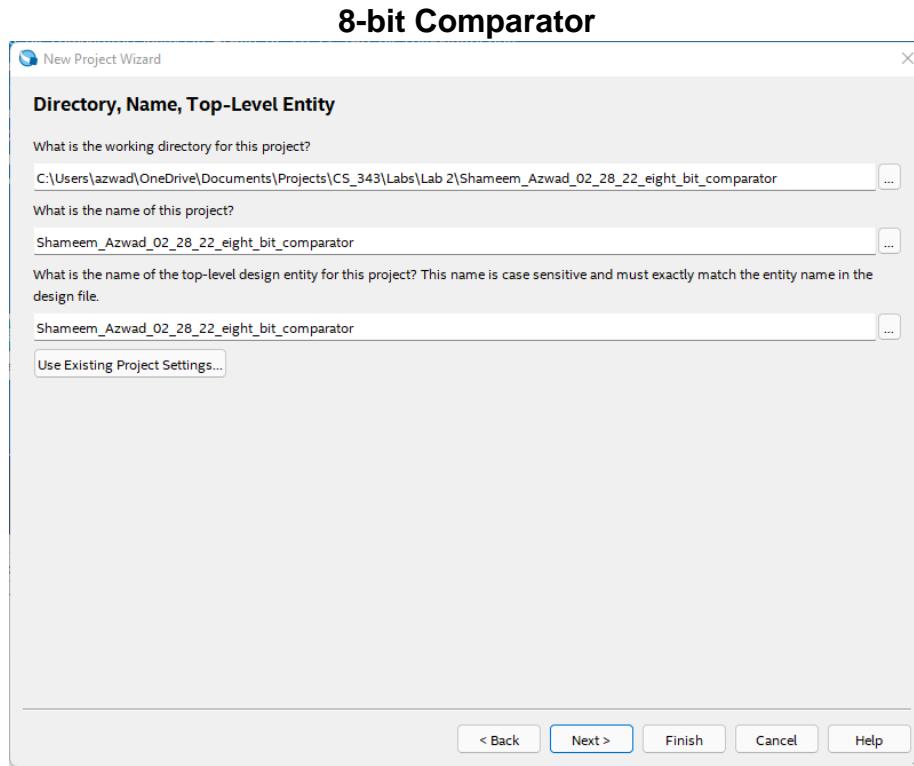


Figure 39: Project Created for Optimized eight bit comparator

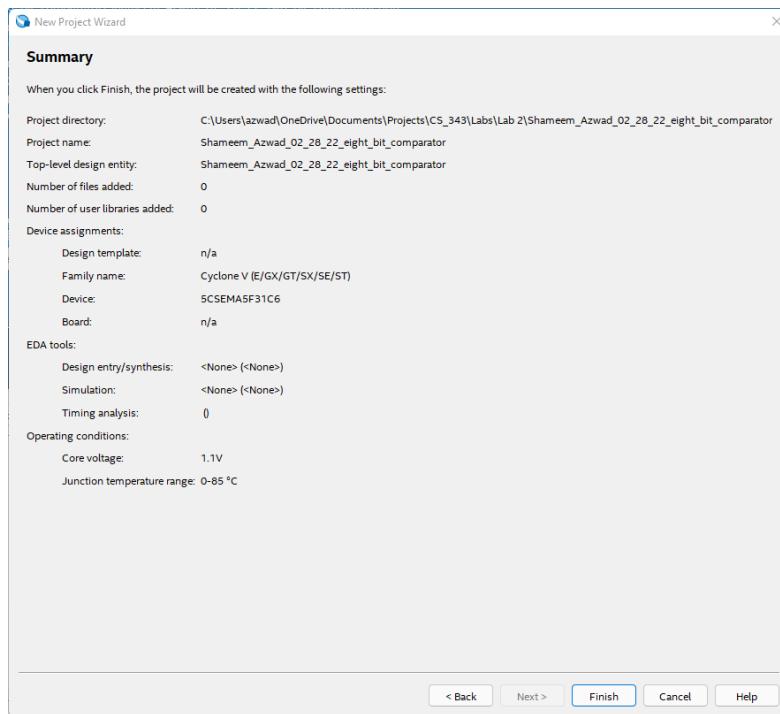


Figure 40: Project Summary

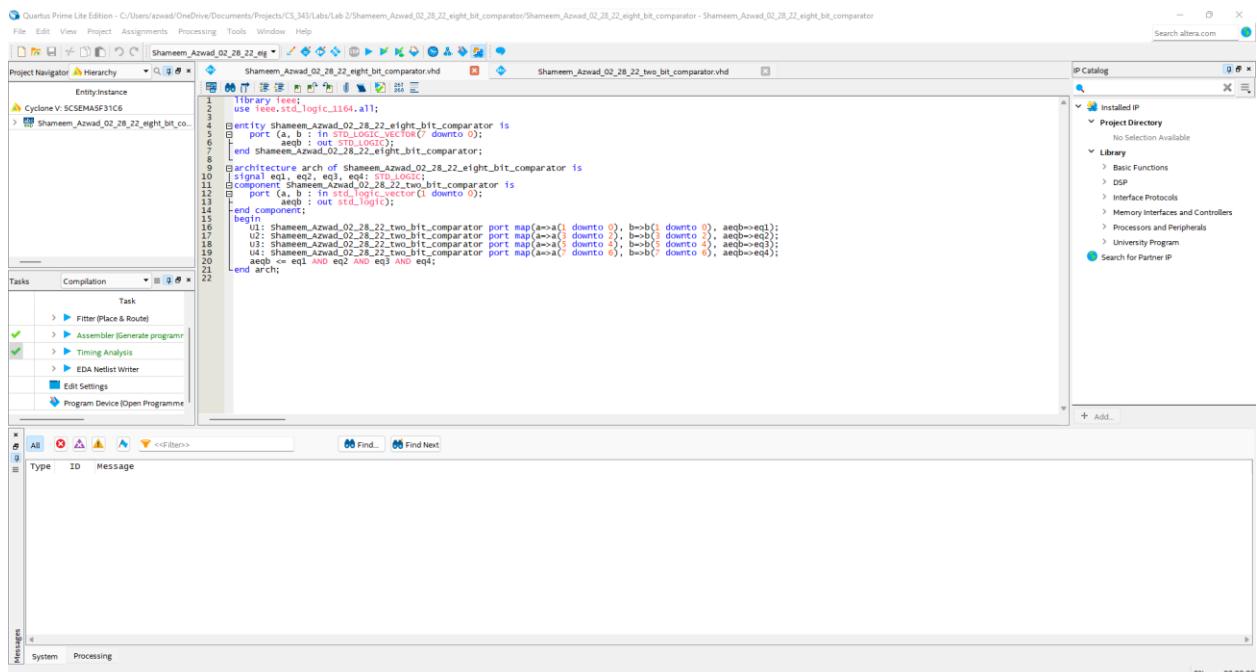


Figure 41: Optimized Eight bit Comparator.

Optimization is done by using the optimized two bit comparator instead of one bit which cuts down the code by 4 lines because you need 4 two bit comparators instead of 8 one bit comparators.

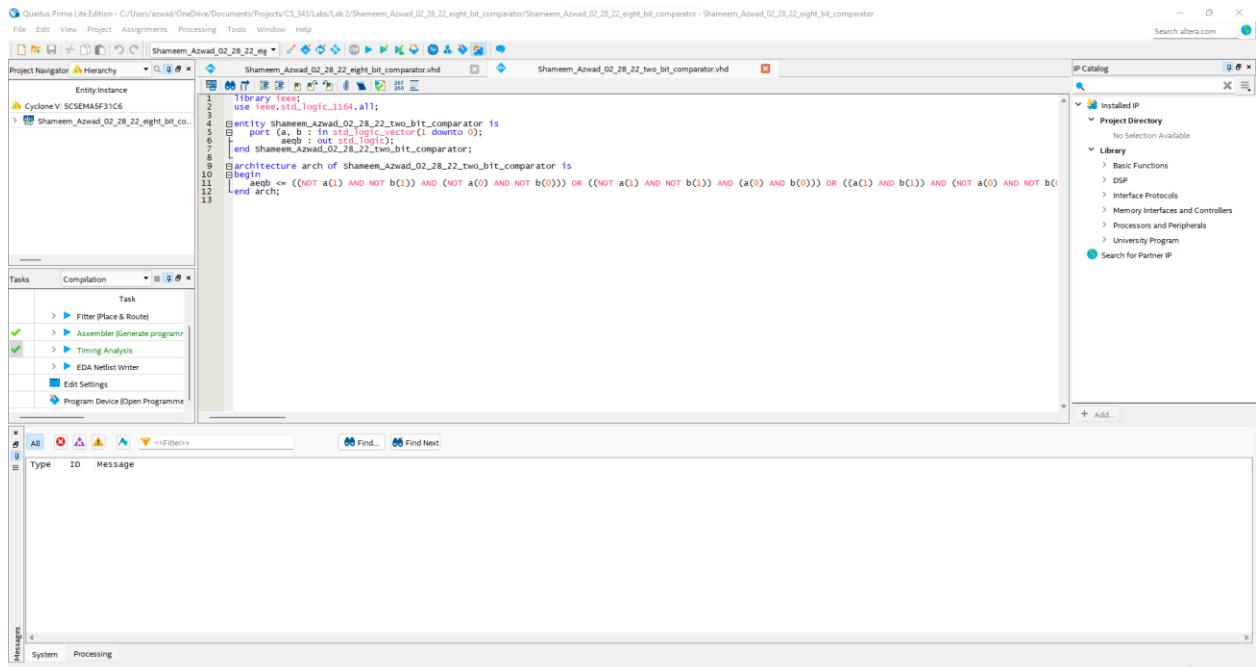


Figure 42: Utilizing Optimized two bit comparator as a component in the optimized eight bit comparator

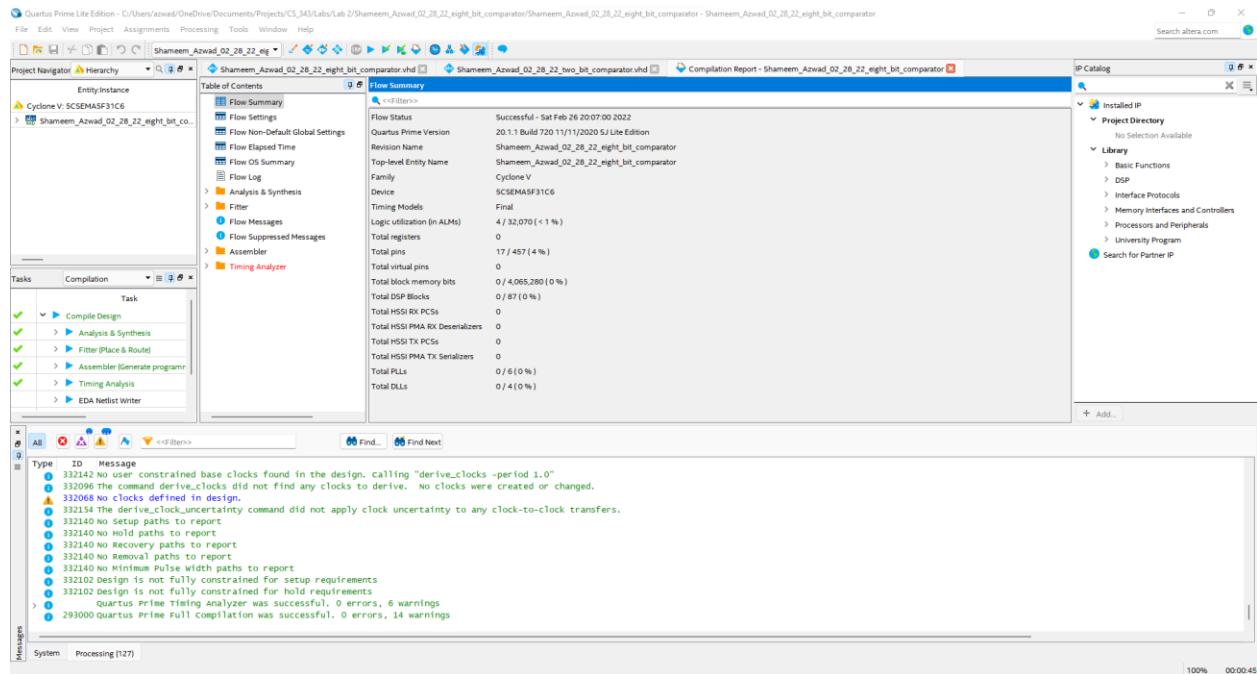


Figure 43: Optimized VHDL code successfully compiled on Quartus

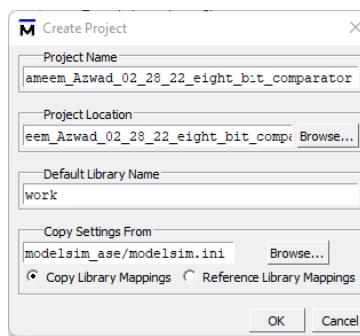
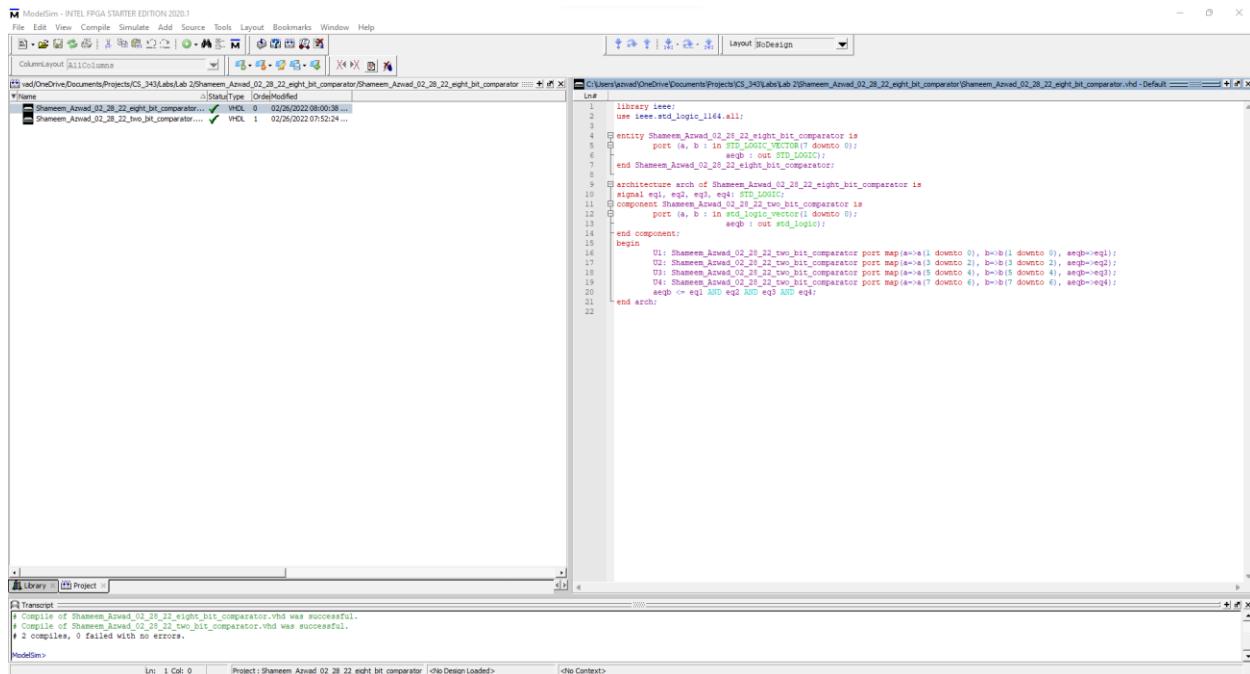


Figure 44: Project created on ModelSim



The screenshot shows the ModelSim interface with several windows open:

- Filemenu - INTEL FPGA STARTER EDITION 2020.1**: The main menu bar.
- ColumnLayout [All Columns]**: A panel showing two VHDL files: "Shameem_Azad_02_28_22_eight_bit_comparator.vhd" and "Shameem_Azad_02_28_22_two_bit_comparator.vhd". Both files are marked as modified (VHDL 0 and VHDL 1).
- Project [Shameem_Azad_02_28_22_eight_bit_comparator]**: A project browser showing the file structure.
- Editor [Shameem_Azad_02_28_22_eight_bit_comparator.vhd - Default]**: The main code editor window displaying the VHDL code for an 8-bit comparator. The code defines a library IEEE, uses STD_LOGIC_VECTOR, and describes an entry for the 8-bit comparator with ports for inputs A and B, and outputs SGEQ and SGBR.
- Terminal [Terminal]**: A terminal window showing the compilation results:


```
# Compilation of Shameem_Azad_02_28_22_eight_bit_comparator.vhd was successful.
# Compilation of Shameem_Azad_02_28_22_two_bit_comparator.vhd was successful.
# 2 compiles, 0 failed with no errors.
```
- ModelSim**: The bottom status bar indicating the current project is "Shameem_Azad_02_28_22_eight_bit_comparator" and the design is loaded.

Figure 45: VHDL code compiled successfully on ModelSim

Task A3) Design and test using ModelSim optimized 2 and 8 bit comparators in VHDL.

Testing optimized 2 bit comparator

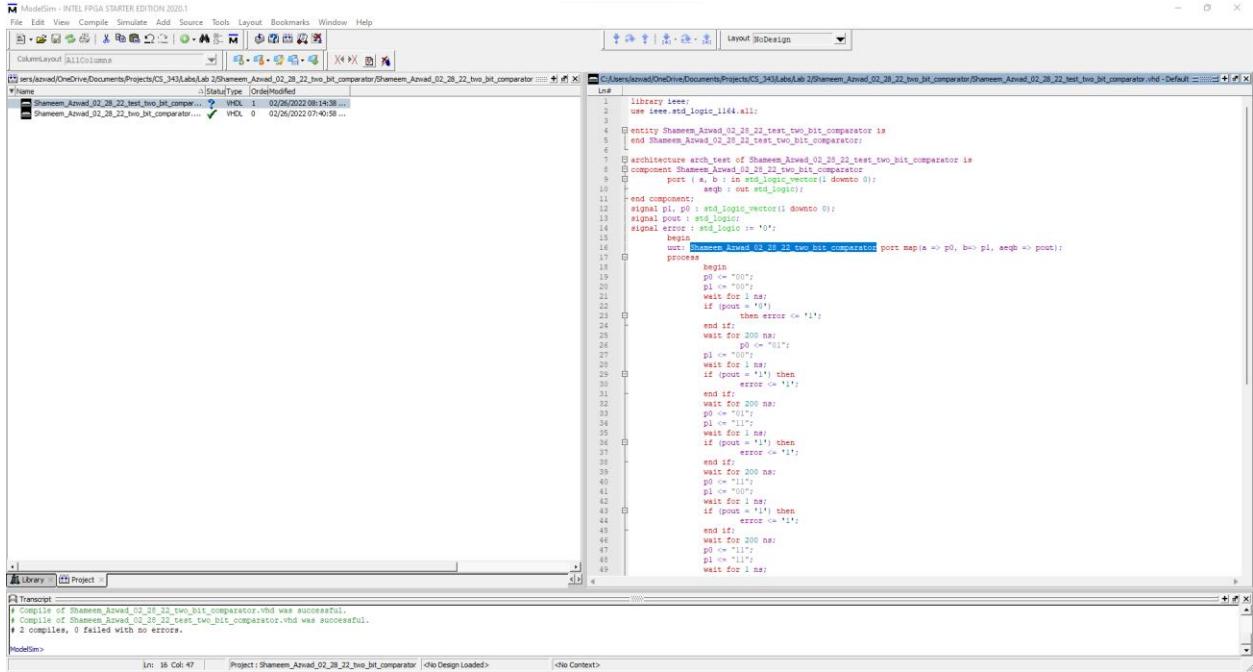


Figure 46: Testbench for optimized 2 bit comparator

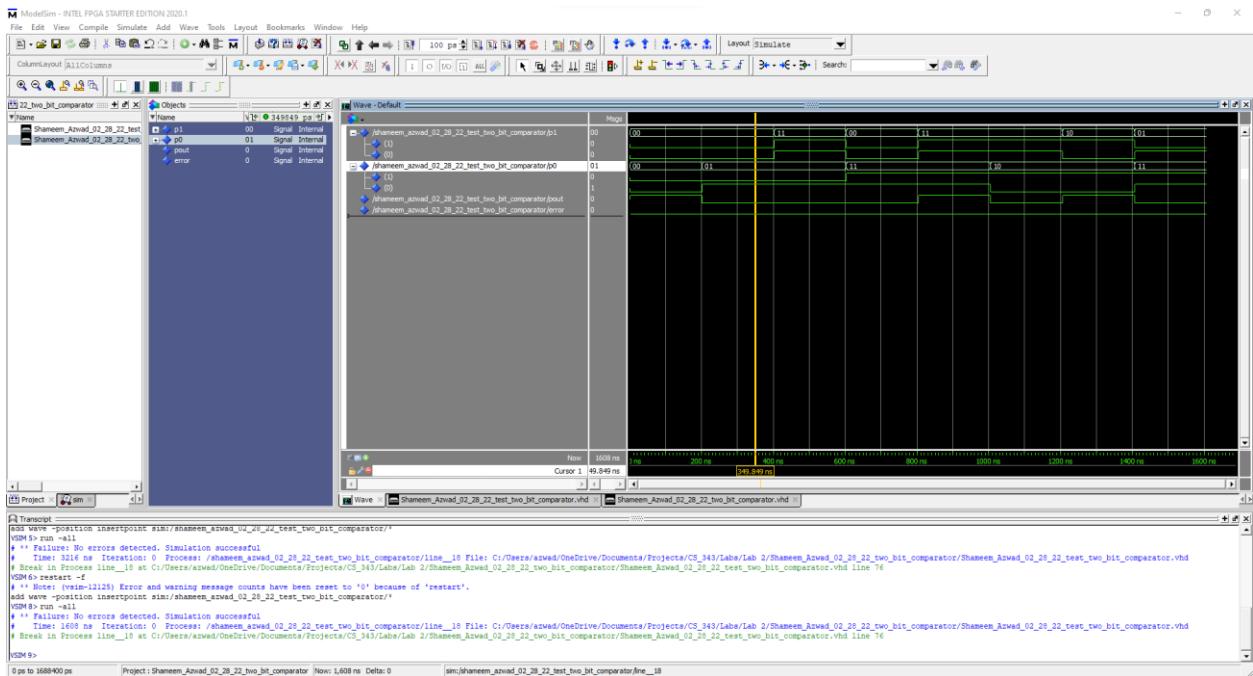


Figure 47: Simulation for optimized 2 bit comparator

8 bit Comparator

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Shameem_Azad_02_28_22_eight_bit_comparator is
end Shameem_Azad_02_28_22_eight_bit_comparator;
architecture arch of Shameem_Azad_02_28_22_eight_bit_comparator is
component Shameem_Azad_02_28_22_eight_bit_comparator
port (a, b : in STD_LOGIC_VECTOR(7 downto 0);
      srgb : out STD_LOGIC);
end component;
signal p1, p0 : STD_LOGIC_VECTOR(7 downto 0);
signal pout : STD_LOGIC;
signal error : STD_LOGIC := '0';
begin
  uut: Shameem_Azad_02_28_22_eight_bit_comparator port map(a => p0, b => p1, srgb => pout);
  process
begin
  p0 <= "00000000";
  p1 <= "00000000";
  for i in 0 to 7 loop
    if (pout = "0") then
      error <= '1';
    end if;
    wait for 200 ns;
    p0 <= "01010101";
    p1 <= "00101010";
    wait for 1 ns;
    if (pout = "1") then
      error <= '1';
    end if;
    wait for 200 ns;
    p0 <= "01100101";
    p1 <= "00110101";
    wait for 1 ns;
    if (pout = "1") then
      error <= '1';
    end if;
    wait for 200 ns;
    p0 <= "00101101";
    p1 <= "00011101";
    wait for 1 ns;
    if (pout = "1") then
      error <= '1';
    end if;
    wait for 200 ns;
    p0 <= "11001100";
    p1 <= "11001100";
    wait for 1 ns;
  end loop;
end process;
end;
  
```

Transcript

```

# Compilation of Shameem_Azad_02_28_22_eight_bit_comparator.vhd was successful.
# Compilation of Shameem_Azad_02_28_22_test_eight_bit_comparator.vhd was successful.
# 3 compilations, 0 failed with no errors.

```

Figure 48: Testbench for optimized 8 bit comparator

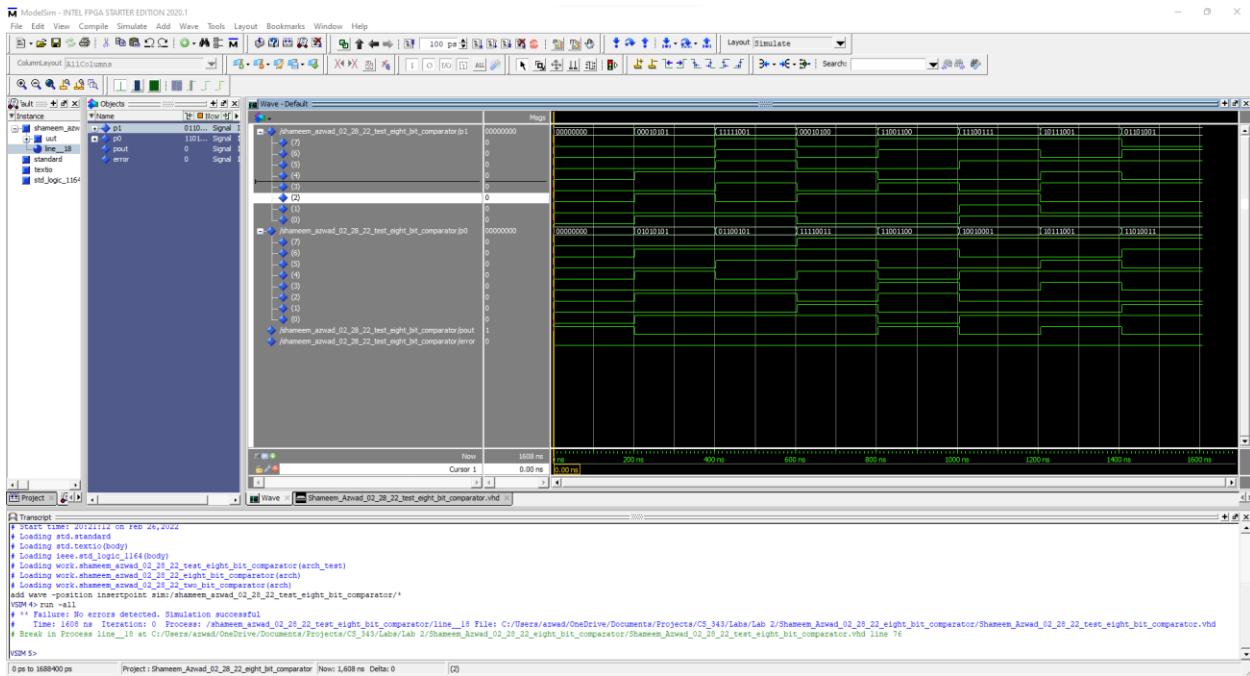


Figure 49: Simulation wavelength for optimized 8 bit comparator

Explanation

Task 0) The tutorial for the equal circuit has been followed. The code and images show that the VHDL code successfully compiles in Quartus and ModelSim for both the circuit and the testbench. The testbench simulation is also correct because it follows truth table accordingly.

Task 1) The VHDL code for the 2 bit comparator and its testbench compiles successfully in both Quartus and ModelSim. The correctness of the 2 bit comparator was tested using the ModelSim and the testbench from the tutorial. The waveform obtained from running the testbench simulation verifies the correctness of the 2 bit comparator.

Task 2) The VHDL code for the 8 bit comparator and its testbench compiles successfully in both Quartus and ModelSim. The 8 bit comparator utilizes the 1 bit comparator as components. Furthermore, the 1 bit comparator is correct because it followed the truth table for the comparator. In addition, the correctness of the 8 bit comparator was tested using the ModelSim and the testbench from the tutorial. The waveform obtained from running the testbench simulation verifies the correctness of the 8 bit comparator.

Task A1) The 1 bit comparator used to have three lines of Boolean operations, which was optimized by merging lines 12, 13, 14 into one line. The optimization was completed merging variables p0's and p1's logic. Essentially, the Boolean operation for p0 and p1 was put in place of the variables in line 12.

Task A2) The 2 bit comparator and the 8 bit comparator were also optimized accordingly. In the case of the optimized *2 bit comparator*, the variables p0, p1, p2, p3 which all had different Boolean operations which took up a line of space each. In order to optimize this the same process as the 1 bit comparator was taken. The Boolean operations was merged lines 12, 13, 14, 15, 16 into a one line Boolean operation which reduced the code by 4 lines of code. Therefore, the optimized 2 bit comparator was optimized because it gained a reduction of 4 lines of code. In the case of the optimized *8 bit comparator* the comparator utilized components of the optimized 2 bit comparators instead of 1 bit comparators, which made the circuit use four optimized 2 bit comparators instead of eight 1 bit comparators. Doing this allowed the optimized 8 bit comparator to reduce 4 lines of code, which optimized the 8 bit comparator by reducing the number of lines of code.

Task A3) The optimized 2 bit comparator and the optimized 8 bit comparator was designed and tested using the ModelSim testbench and the waveforms still match the correct outputs just like the non-optimized versions. The waveforms for both comparators still match the correct simulation results and have no error. This shows that the comparators have been designed and tested for correctness.

Conclusion

The lab was very beneficial to gain a better understanding of optimization VHDL coding because it made the designer first create a component using the given equation and then made them think about how to optimize by reducing lines of code.

Furthermore, the lab had a great effect on understanding the benefits of using components with either port maps or logic vectors and how to optimize. Essentially, the lab was a great way to gain more practice with VHDL in general and a way to learn not only about designing circuits but also optimizing circuits and even after they are made.