The City College of New York

Computer Organization Lab

CSC 34300

(Spring 2024)

LAB 1

2 to 1 Multiplexer

Mahir Asef

Instructor: Albi Arapi

Date: 02.21.2024

<u>Table of Contents</u>

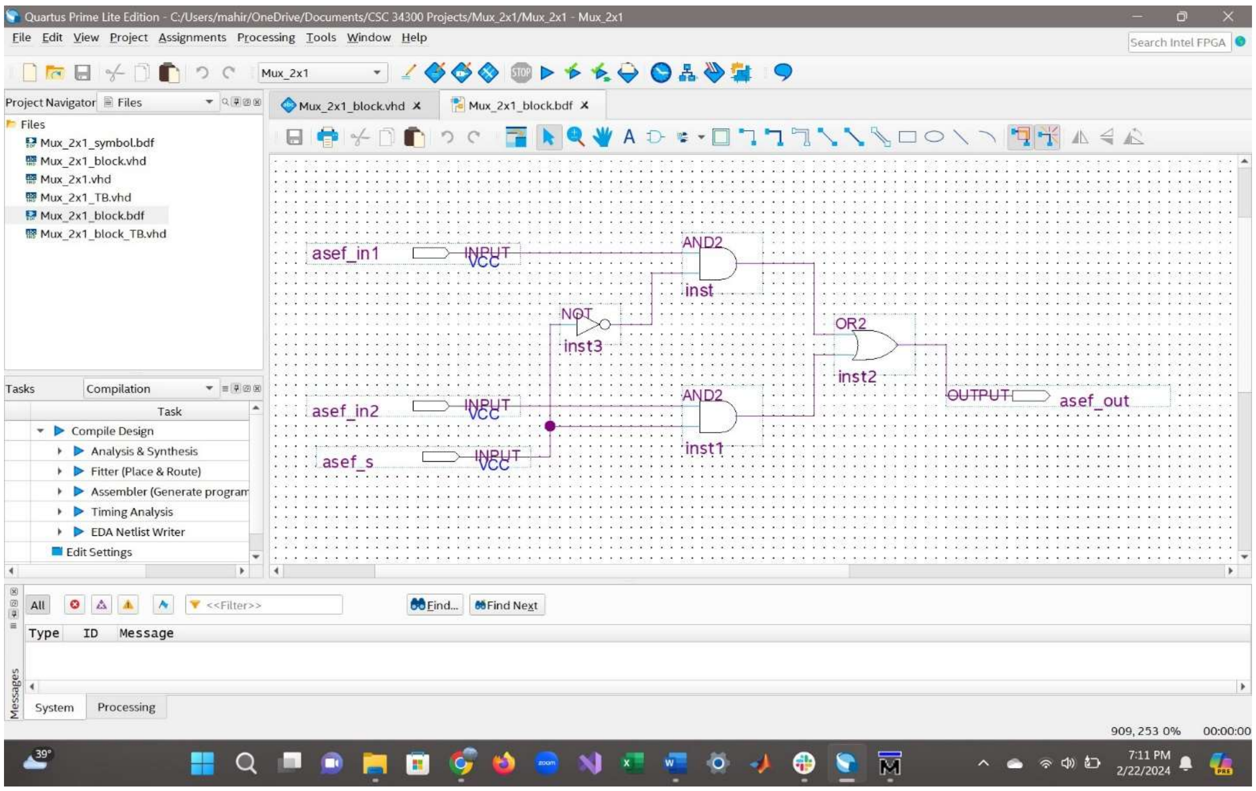
Objective:

The purpose of this lab is to design a 2:1 multiplexer in Quartus Digital Circuit and test our design by simulation of waveform in Modelsim. By the end of this lab, we should be able to design schematic of a 2:1 multiplexer using appropriate gates and from our design we will be able to formulate the input-output truth table for the 2:1 multiplexer as well as create Boolean function of the design. This lab will also require us to use our truth table and Boolean function to manually create a 2:1 multiplexer in VHDL hardware programming language. Towards the end of the lab, we should be able to use our design to create a simple symbol diagram in Quartus. In the final part of the assignment, we will learn how to create testbenches for our gate design, the gate design VHDL code and the truth table VHDL code in order to simulate them in Modelsim to verify all our programs and design which will conclude our experiment.

Functionality and Specifications:

Part 1:

Gate/Schematic Design:



Part 2:

S = Control Input

I0 = Dependent input for S = 0

I1 = Dependent input for S = 1

F = output for 2:1 Multiplexer

Truth Table:

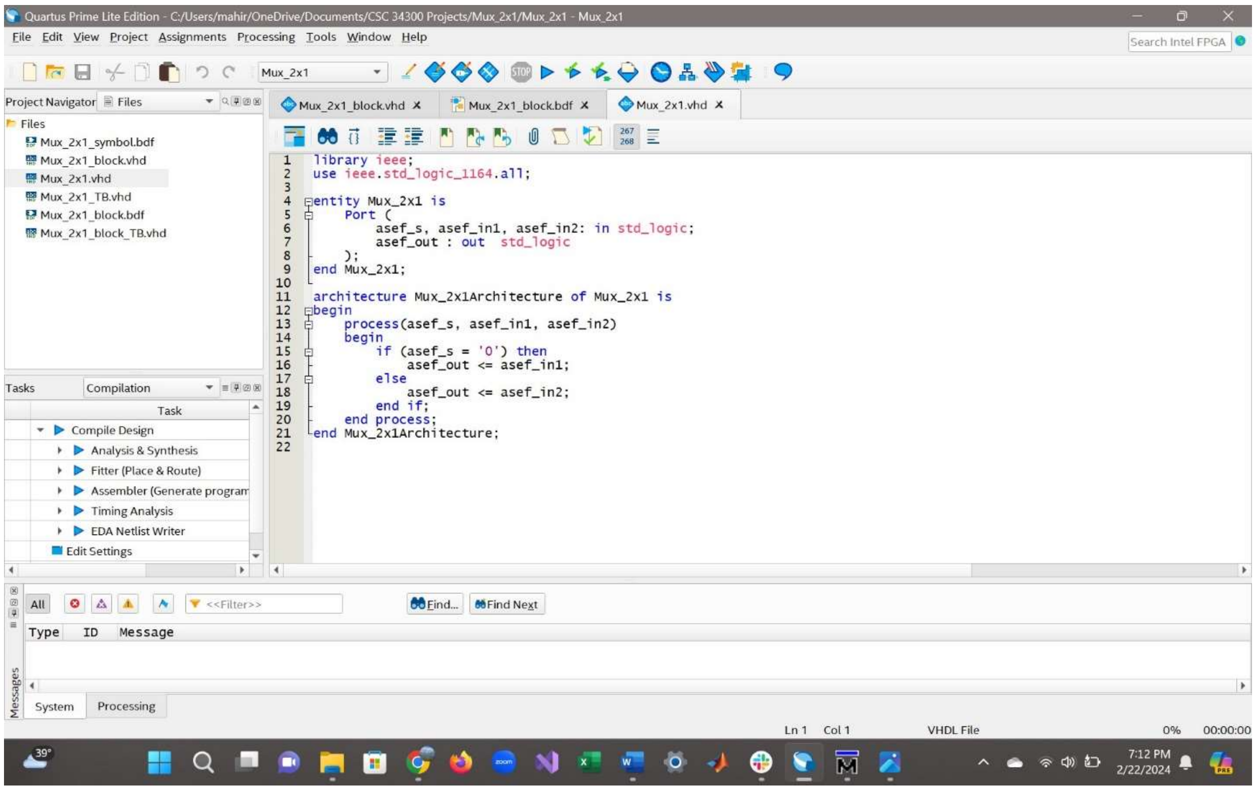
S	I0	I1	F
0	0	X	0
0	0	X	1
1	X	0	0
1	X	1	1

Boolean Function:

F = (S’. I0) + (S.I1)

Part 3:

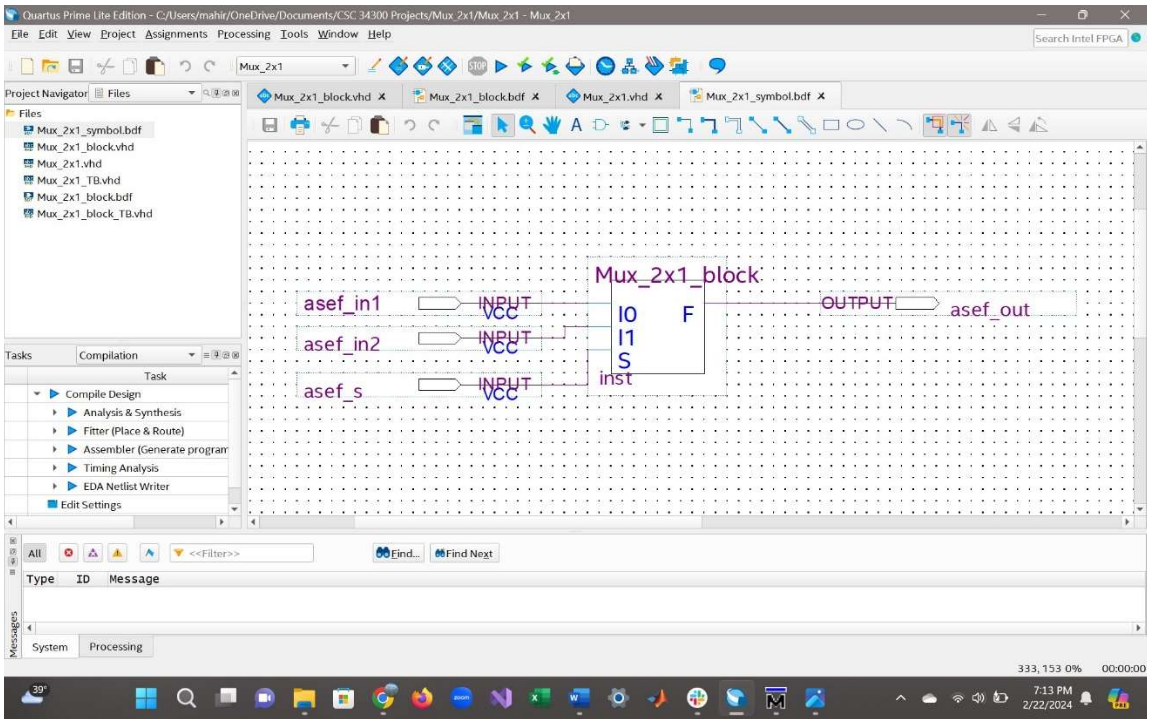
VHDL Code for Truth Table:



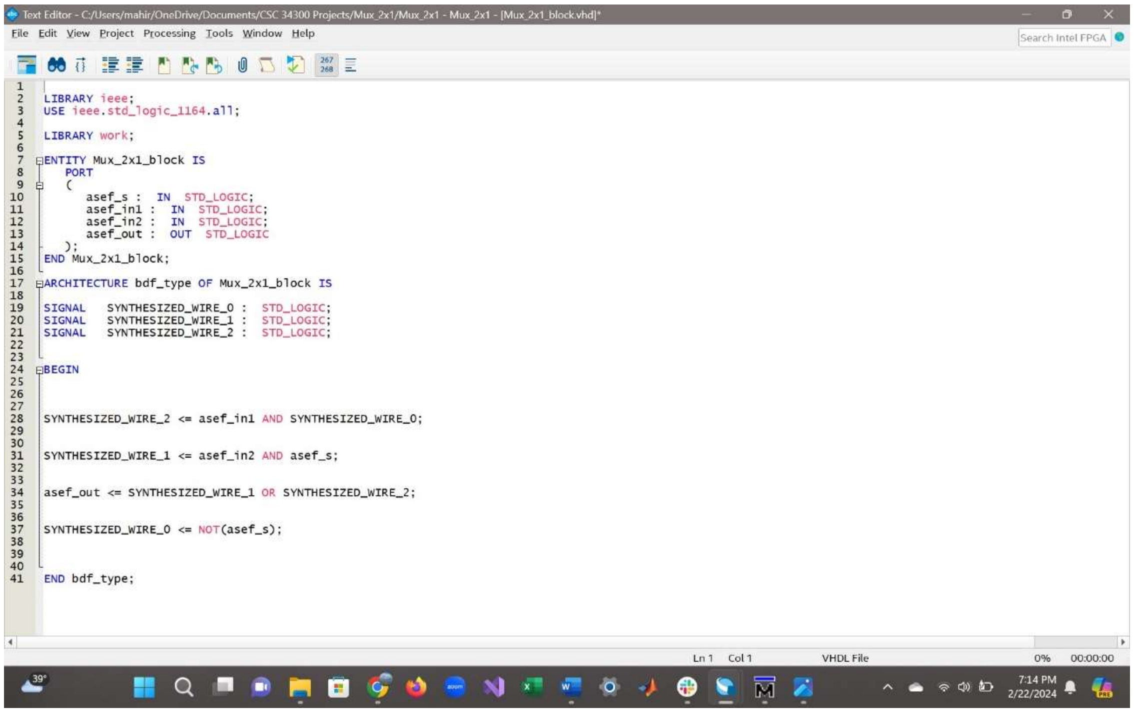
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Mux_2x1 is
    Port (
        asef_s, asef_in1, asef_in2: in std_logic;
        asef_out : out  std_logic
    );
end Mux_2x1;

architecture Mux_2x1Architecture of Mux_2x1 is
begin
    process(asef_s, asef_in1, asef_in2)
    begin
        if (asef_s = '0') then
            asef_out <= asef_in1;
        else
            asef_out <= asef_in2;
        end if;
    end process;
end Mux_2x1Architecture;

```

Part 4:

Symbol Block Design:



Part 5:

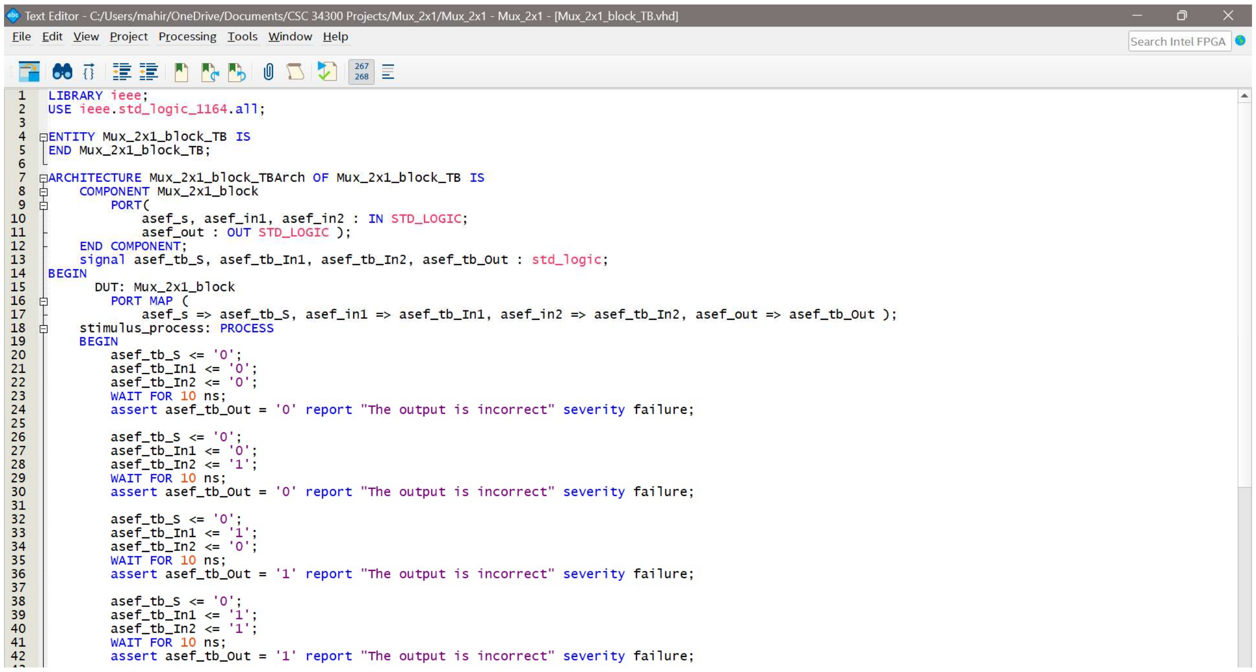

Text Editor - C:/Users/mahir/OneDrive/Documents/CSC 34300 Projects/Mux_2x1/Mux_2x1 - Mux_2x1 - [Mux_2x1_block.vhd]*

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;

ENTITY Mux_2x1_block IS
    PORT
    (
        asef_s :  IN  STD_LOGIC;
        asef_in1 :  IN  STD_LOGIC;
        asef_in2 :  IN  STD_LOGIC;
        asef_out :  OUT  STD_LOGIC
    );
END Mux_2x1_block;

ARCHITECTURE bdf_type OF Mux_2x1_block IS

SIGNAL   SYNTHESIZED_WIRE_0 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_1 :  STD_LOGIC;
SIGNAL   SYNTHESIZED_WIRE_2 :  STD_LOGIC;


BEGIN



SYNTHESIZED_WIRE_2 <= asef_in1 AND SYNTHESIZED_WIRE_0;


SYNTHESIZED_WIRE_1 <= asef_in2 AND asef_s;


asef_out <= SYNTHESIZED_WIRE_1 OR SYNTHESIZED_WIRE_2;


SYNTHESIZED_WIRE_0 <= NOT(asef_s);



END bdf_type;
```

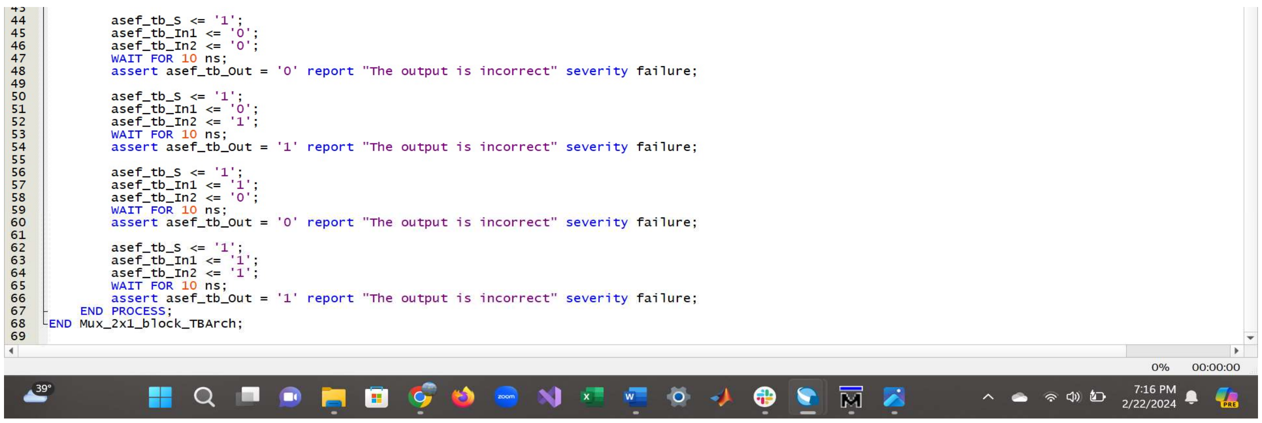
Comparing the VHDL generated code to our user created truth table code, we can see that they follow the same VHDL model which is dataflow model however the convention varies at different places of the program. The entity declaration and the port declaration within remains the same. However, the generated code doesn't declare the output signal in the architecture which we have done for the truth table code. The generated code also calls our signals SYTHASIZED_WIRES and when the process begins instead of using conditional (if-else) logic that we have used to create our own code, the generated code seems to make direct comparison between signals which also takes more steps (at least 4 steps as shown above). In our code with conditional statement, we can optimize our code to handle two operations at once (For S =0 and For S = 1).

Part 6:

Block/Schematic Testbench:

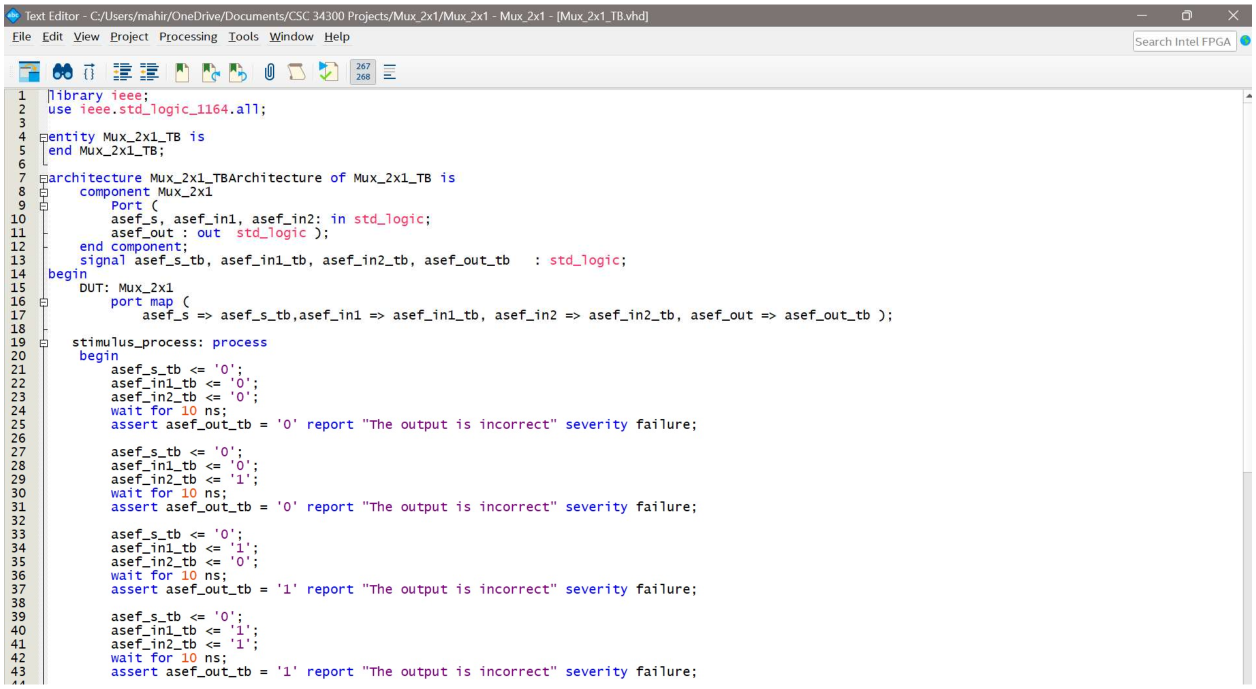

Text Editor - C:/Users/mahir/OneDrive/Documents/CSC 34300 Projects/Mux_2x1/Mux_2x1 - Mux_2x1 - [Mux_2x1_block_TB.vhd]
File Edit View Project Processing Tools Window Help
Search Intel FPGA

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Mux_2x1_block_TB IS
END Mux_2x1_block_TB;

ARCHITECTURE Mux_2x1_block_TBArch OF Mux_2x1_block_TB IS
    COMPONENT Mux_2x1_block
        PORT(
            asef_s, asef_in1, asef_in2 : IN STD_LOGIC;
            asef_out : OUT STD_LOGIC );
    END COMPONENT;
    signal asef_tb_S, asef_tb_In1, asef_tb_In2, asef_tb_Out : std_logic;
BEGIN
      DUT: Mux_2x1_block
        PORT MAP (
            asef_s => asef_tb_S, asef_in1 => asef_tb_In1, asef_in2 => asef_tb_In2, asef_out => asef_tb_Out );
    stimulus_process: PROCESS
    BEGIN
        asef_tb_S <= '0';
        asef_tb_In1 <= '0';
        asef_tb_In2 <= '0';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '0' report "The output is incorrect" severity failure;

        asef_tb_S <= '0';
        asef_tb_In1 <= '0';
        asef_tb_In2 <= '1';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '0' report "The output is incorrect" severity failure;

        asef_tb_S <= '0';
        asef_tb_In1 <= '1';
        asef_tb_In2 <= '0';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '1' report "The output is incorrect" severity failure;

        asef_tb_S <= '0';
        asef_tb_In1 <= '1';
        asef_tb_In2 <= '1';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '1' report "The output is incorrect" severity failure;
```

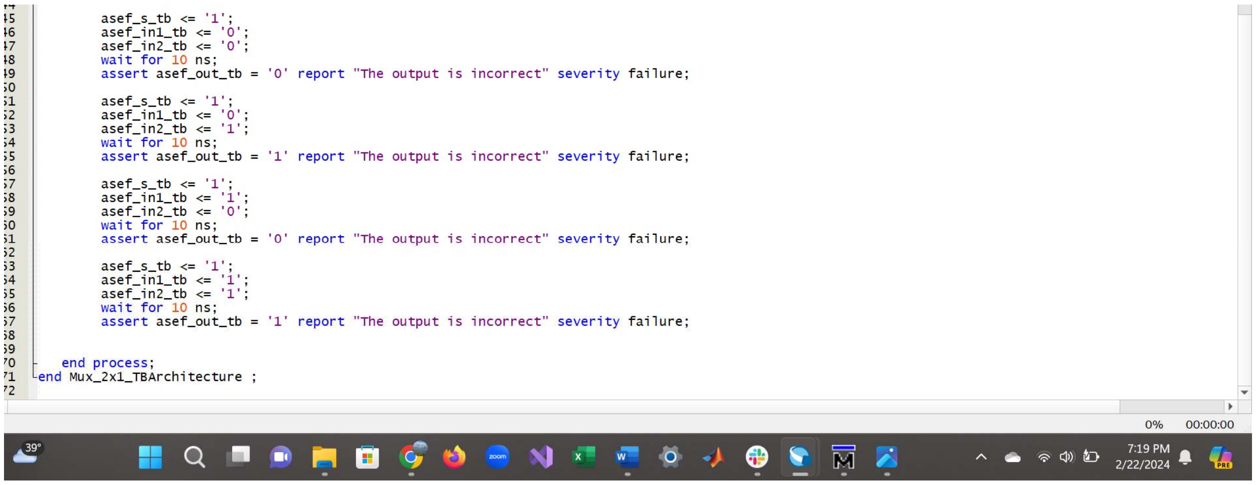


```vhdl

        asef_tb_S <= '1';
        asef_tb_In1 <= '0';
        asef_tb_In2 <= '0';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '0' report "The output is incorrect" severity failure;

        asef_tb_S <= '1';
        asef_tb_In1 <= '0';
        asef_tb_In2 <= '1';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '1' report "The output is incorrect" severity failure;

        asef_tb_S <= '1';
        asef_tb_In1 <= '1';
        asef_tb_In2 <= '0';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '0' report "The output is incorrect" severity failure;

        asef_tb_S <= '1';
        asef_tb_In1 <= '1';
        asef_tb_In2 <= '1';
        WAIT FOR 10 ns;
        assert asef_tb_Out = '1' report "The output is incorrect" severity failure;
    END PROCESS;
END Mux_2x1_block_TBArch;

```

Truth Table VHDL Code Testbench:


Text Editor - C:/Users/mahir/OneDrive/Documents/CSC 34300 Projects/Mux_2x1/Mux_2x1 - Mux_2x1 - [Mux_2x1_TB.vhd]
File Edit View Project Processing Tools Window Help
Search Intel FPGA

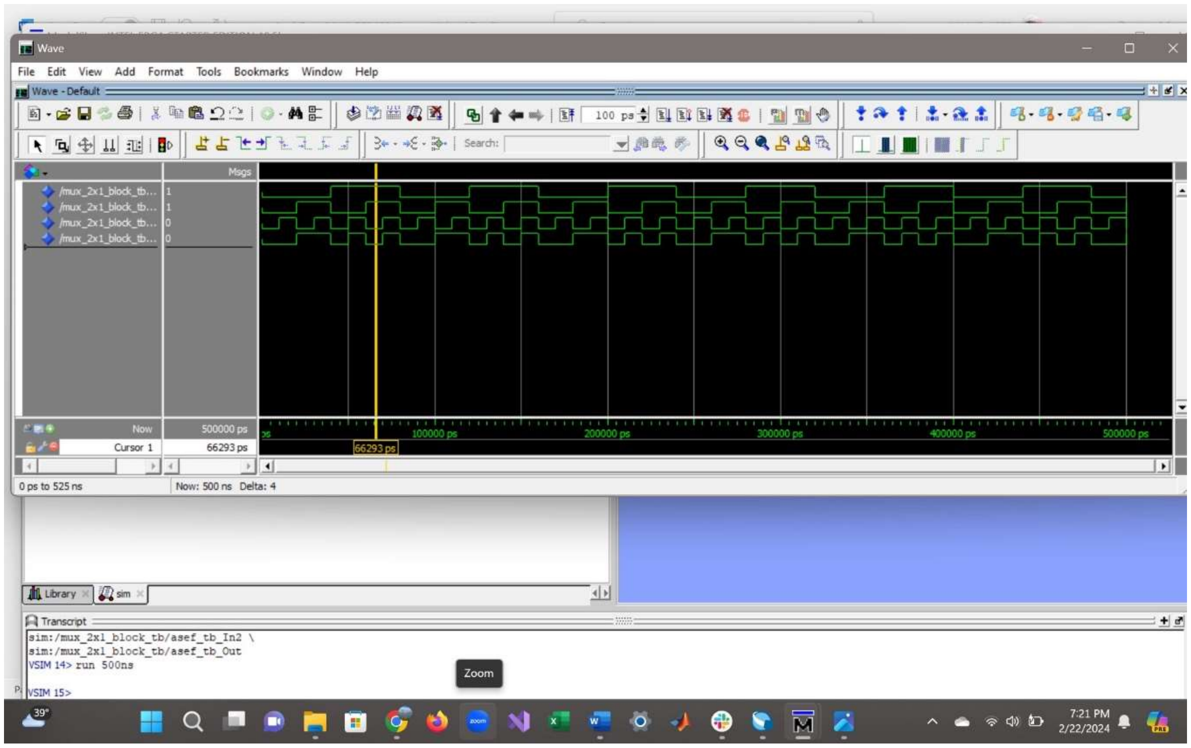
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Mux_2x1_TB is
end Mux_2x1_TB;

architecture Mux_2x1_TBArchitecture of Mux_2x1_TB is
    component Mux_2x1
        Port (
        asef_s, asef_in1, asef_in2: in std_logic;
        asef_out : out  std_logic );
    end component;
    signal asef_s_tb, asef_in1_tb, asef_in2_tb, asef_out_tb   : std_logic;
begin
    DUT: Mux_2x1
        port map (
            asef_s => asef_s_tb,asef_in1 => asef_in1_tb, asef_in2 => asef_in2_tb, asef_out => asef_out_tb );

   stimulus_process: process
    begin
        asef_s_tb <= '0';
        asef_in1_tb <= '0';
        asef_in2_tb <= '0';
        wait for 10 ns;
        assert asef_out_tb = '0' report "The output is incorrect" severity failure;

        asef_s_tb <= '0';
        asef_in1_tb <= '0';
        asef_in2_tb <= '1';
        wait for 10 ns;
        assert asef_out_tb = '0' report "The output is incorrect" severity failure;

        asef_s_tb <= '0';
        asef_in1_tb <= '1';
        asef_in2_tb <= '0';
        wait for 10 ns;
        assert asef_out_tb = '1' report "The output is incorrect" severity failure;

        asef_s_tb <= '0';
        asef_in1_tb <= '1';
        asef_in2_tb <= '1';
        wait for 10 ns;
        assert asef_out_tb = '1' report "The output is incorrect" severity failure;
```



```vhdl
        asef_s_tb <= '1';
        asef_in1_tb <= '0';
        asef_in2_tb <= '0';
        wait for 10 ns;
        assert asef_out_tb = '0' report "The output is incorrect" severity failure;

        asef_s_tb <= '1';
        asef_in1_tb <= '0';
        asef_in2_tb <= '1';
        wait for 10 ns;
        assert asef_out_tb = '1' report "The output is incorrect" severity failure;

        asef_s_tb <= '1';
        asef_in1_tb <= '1';
        asef_in2_tb <= '0';
        wait for 10 ns;
        assert asef_out_tb = '0' report "The output is incorrect" severity failure;

        asef_s_tb <= '1';
        asef_in1_tb <= '1';
        asef_in2_tb <= '1';
        wait for 10 ns;
        assert asef_out_tb = '1' report "The output is incorrect" severity failure;


    end process;
end Mux_2x1_TBArchitecture ;
```

Simulations:

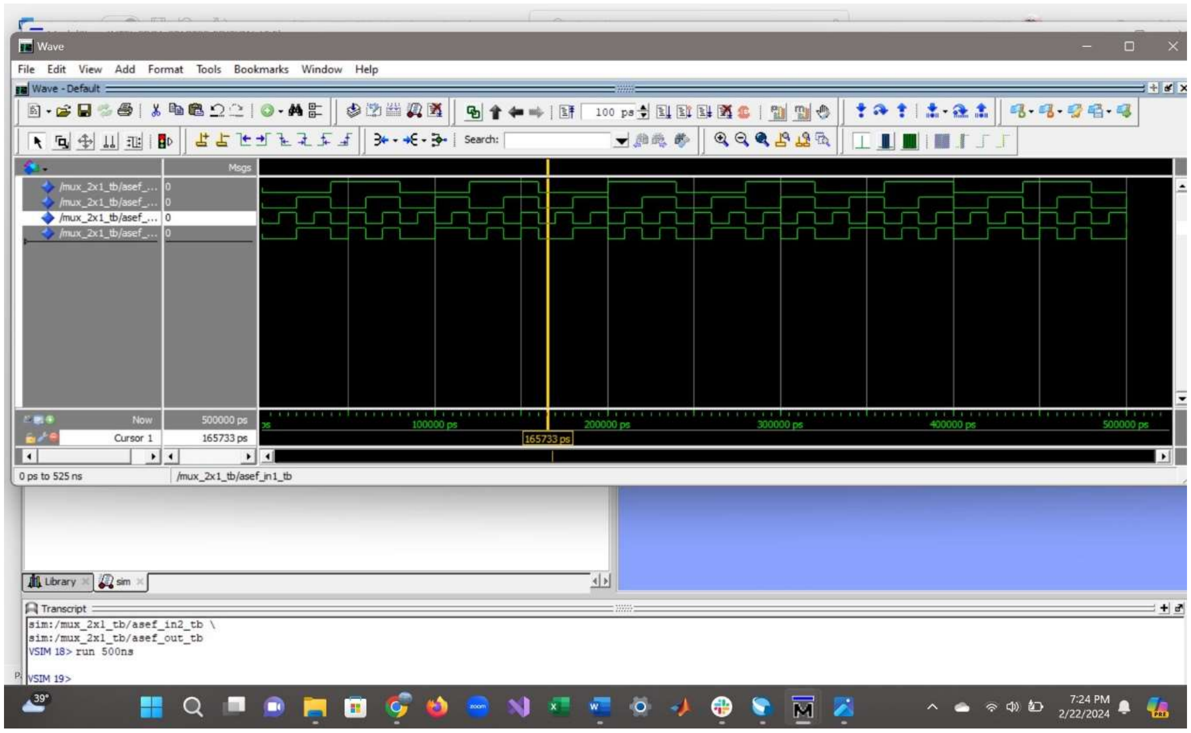
Schematic Design ModelSim Waveform:



```
/mux_2x1_block_tb...  1
/mux_2x1_block_tb...  1
/mux_2x1_block_tb...  0
/mux_2x1_block_tb...  0

Now       500000 ps
Cursor 1  66293 ps

0 ps to 525 ns    Now: 500 ns  Delta: 4
```

```
sim:/mux_2x1_block_tb/asef_tb_In2 \
sim:/mux_2x1_block_tb/asef_tb_Out
VSIM 14> run 500ns

VSIM 15>
```

Truth Table VHDL Code ModelSim Waveform:



```
/mux_2x1_tb/asef_...  0
/mux_2x1_tb/asef_...  0
/mux_2x1_tb/asef_...  0
/mux_2x1_tb/asef_...  0

Now       500000 ps
Cursor 1  165733 ps

0 ps to 525 ns    /mux_2x1_tb/asef_in1_tb
```

```
sim:/mux_2x1_tb/asef_in2_tb \
sim:/mux_2x1_tb/asef_out_tb
VSIM 18> run 500ns

VSIM 19>
```

Conclusion:

We have reached the end of our 2:1 multiplexer experiment. By now, we have done all the experiments necessary. First, we started by creating a block diagram for 2:1 multiplexer in Quartus using a NOT gate, two AND gates and one OR gate. We have attached 3 inputs and 1 output to it as well. Following the diagram, then we created our truth table for MUX as well as the Boolean expression of the truth table which was shown in part 2 above. Then we moved back to Quartus to build the MUX from the truth table using VHDL code. We followed the behavioral model of VHDL to create our MUX design program. Next up, we have generated a VHDL program file from our block diagram that we created in part 1 and compared that VHDL model with our own and identified all the similarities and differences as explained above. Following our block diagram we then created a symbol diagram and defined our inputs and outputs there as well. Once we have designed all our models, we started writing testbench in VHDL code for both our truth table model as well as our gate/schematic model. Finally, we have used ModelSIm to create a work library that we compiled with our truth table VHDL, and block generated VHDL as well as their corresponding testbench VHDL. Our final step was to simulate the MUX that we have designed using gate and programmed using code to verify the functionality of our design. As we ran the simulation, we were able to prove that our designed 2:1 multiplexer indeed works as intended as shown above. Overall, this lab was a great success in learning 2:1 MUX as well as getting familiarized with how Quartus builds model that we eventually simulate and run design verification with using ModelSim.