Review Laboratory Exercise: MUX Basic Circuit Design and Testing

Zi Xuan Li

The Grove School of Engineering, The City College of New York

CSC 34300 5DE[43223]: Computer Systems Design Laboratory

Professor Gertner, TA Albi Arapi

February 20th, 2024

Table of Contents

Objective……..3

Functionality and Specifications……..3

What is the combinational logic function (Boolean function) of the circuit? …….3

What is the VHDL code (Boolean function) of the circuit?.....3

Include a screenshot of your design steps and of final the circuit……...…4

Simulation……..……..5

Include a screenshot of the vector waveform output file for all possible inputs…….6

Draw a truth table from the waveforms and explain how you to derive it……..6

Write a Boolean function for each exercise…….7

Conclusion……...7

Objective: The goal of this exercise is to design and verify the functionality of our own 2:1 Multiplexer using Quartus and ModelSim.

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

The combinational logic function for a 2:1 Multiplexer determines the output based on the select signal. Given input signals A, B, S where S is the select signal. The function selects A or B as the output based on the value of S.

$$Y = (A \text{ AND } (\text{NOT } S)) \text{ OR } (B \text{ AND } S)$$

What is the VHDL code (Boolean function) of the circuit?

```
library ieee;
use IEEE.STD_LOGIC_1164.ALL;

entity p3_2to1_Multiplexer is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Sel : in STD_LOGIC;
           Y : out STD_LOGIC);
end p3_2to1_Multiplexer;

architecture Behavioral of p3_2to1_Multiplexer is
begin
    process (A, B, Sel)
    begin
        if (Sel = '0') then
            Y <= A;
        else
            Y <= B;
        end if;
    end process;
end Behavioral;
```

Include a screenshot of your design steps and of final the circuit.

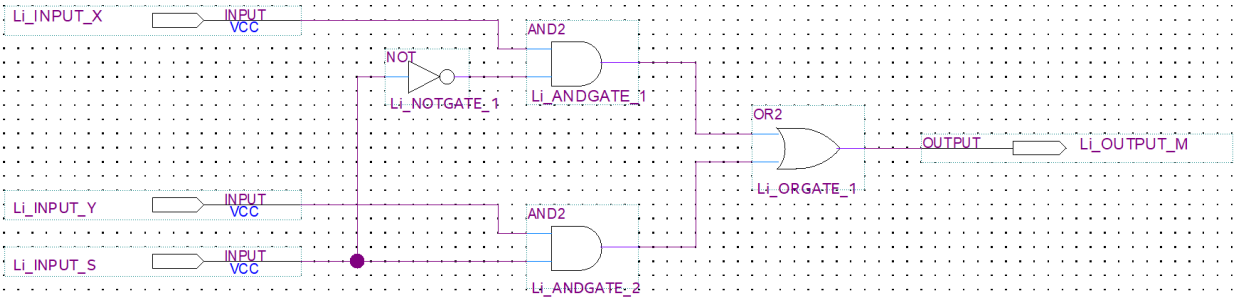

Figure 1: This is my block diagram file for Part 1 of the exercise. Only AND, OR, and NOT gates were used to recreate the functionality of a 2:1 Multiplexer.

```
1   library ieee;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity p3_2to1_Multiplexer is
5       Port ( A : in STD_LOGIC;
6              B : in STD_LOGIC;
7              Sel : in STD_LOGIC;
8              Y : out STD_LOGIC);
9   end p3_2to1_Multiplexer;
10
11  architecture Behavioral of p3_2to1_Multiplexer is
12  begin
13      process (A, B, Sel)
14      begin
15          if (Sel = '0') then
16              Y <= A;
17          else
18              Y <= B;
19          end if;
20      end process;
21  end Behavioral;
22
```

Figure 2: This is my VHDL code for Part 3 of the exercise that I wrote to recreate the functionality of a 2:1 Multiplexer.

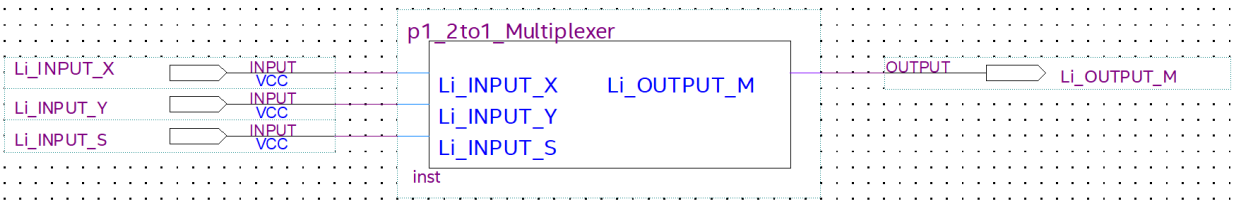

Figure 3: This is my block diagram file for Part 4 of the exercise. This symbol was created using the bdf from figure 1.

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  LIBRARY work;
5
6  ENTITY p5_2to1_Multiplexer IS
7     PORT
8     (
9        Li_INPUT_Y :  IN  STD_LOGIC;
10       Li_INPUT_X :  IN  STD_LOGIC;
11       Li_INPUT_S :  IN  STD_LOGIC;
12       Li_OUTPUT_M :  OUT  STD_LOGIC
13    );
14 END p5_2to1_Multiplexer;
15
16 ARCHITECTURE bdf_type OF p5_2to1_Multiplexer IS
17
18 SIGNAL   SYNTHESIZED_WIRE_0 :  STD_LOGIC;
19 SIGNAL   SYNTHESIZED_WIRE_1 :  STD_LOGIC;
20 SIGNAL   SYNTHESIZED_WIRE_2 :  STD_LOGIC;
21
22
23 BEGIN
24
25
26
27 SYNTHESIZED_WIRE_2 <= Li_INPUT_X AND SYNTHESIZED_WIRE_0;
28
29
30 SYNTHESIZED_WIRE_1 <= Li_INPUT_Y AND Li_INPUT_S;
31
32
33 SYNTHESIZED_WIRE_0 <= NOT(Li_INPUT_S);
34
35
36
37 Li_OUTPUT_M <= SYNTHESIZED_WIRE_1 OR SYNTHESIZED_WIRE_2;
38
39
40 END bdf_type;
```

Figure 4: This is the VHDL code for Part 5 of the exercise that was automatically generated by Quartus using the bdf from Figure 1.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

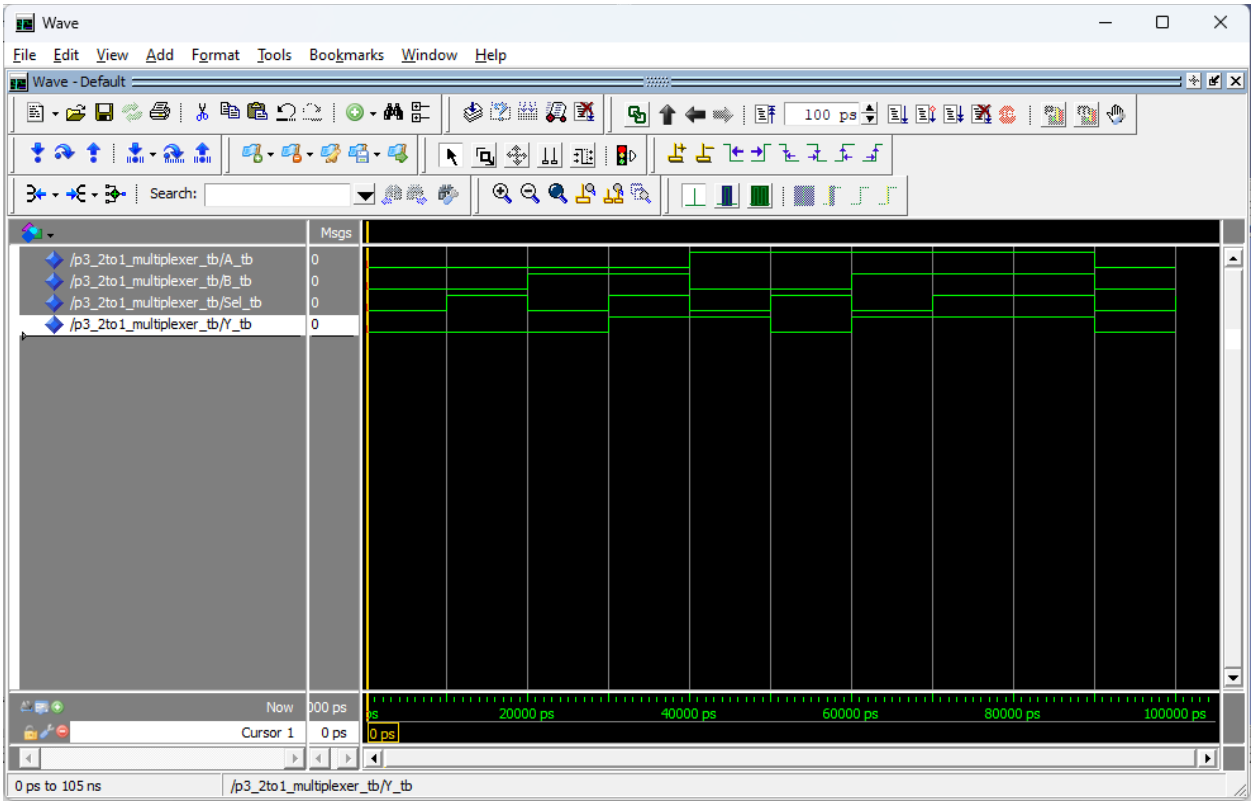

Figure 5: This is the vector waveform output from ModelSim based on the testbench that I designed for the 2:1 Multiplexer.

Draw a truth table from the waveforms and explain how you used the waveform data to derive it.

The waveform data for each segment corresponds to an input that is being tested. This is because in the test bench, I mapped A to A_tb, B to B_tb, S to Sel_tb, and Y to Y_tb. Knowing this, all I had to do was read the waveform data to derive the output Y given any combination of input A, B, and S. To determine whether the input was high (1) or low (0), I looked at the waveform's height in the simulation.

Input A	Input B	Input S	Output Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Write a Boolean function for each exercise.

The Boolean function for each exercise is the same because we're dealing with the same multiplexer throughout the exercise. Nevertheless, the Boolean function is as follows:

$$Y = AS' + YS$$

Conclusion: This exercise offered a practical exploration of digital circuit design and simulation techniques through the creating and testing of a 2:1 Multiplexer. By constructing the mux in Quartus and verifying its functionality in ModelSim, we gained an understanding of the behavior of a 2:1 MUX, implemented the MUX using both gates and VHDL code, as well as verifying the correctness of designs through waveform simulation sin ModelSim.