Santa Clara University

Lab #4: Multiplexer Design

ELEN 21L 51306 Tuesday 2:15-5pm
May 2-5, 2017

Group 2
Erika Sweet
Yutong Li

Lab #4: Multiplexer Design

I. Objectives:

In this lab we will design and use a multiplexer as:

- A switch to connect different inputs to a single wire
- A building block for logic circuit.

II. Introduction:

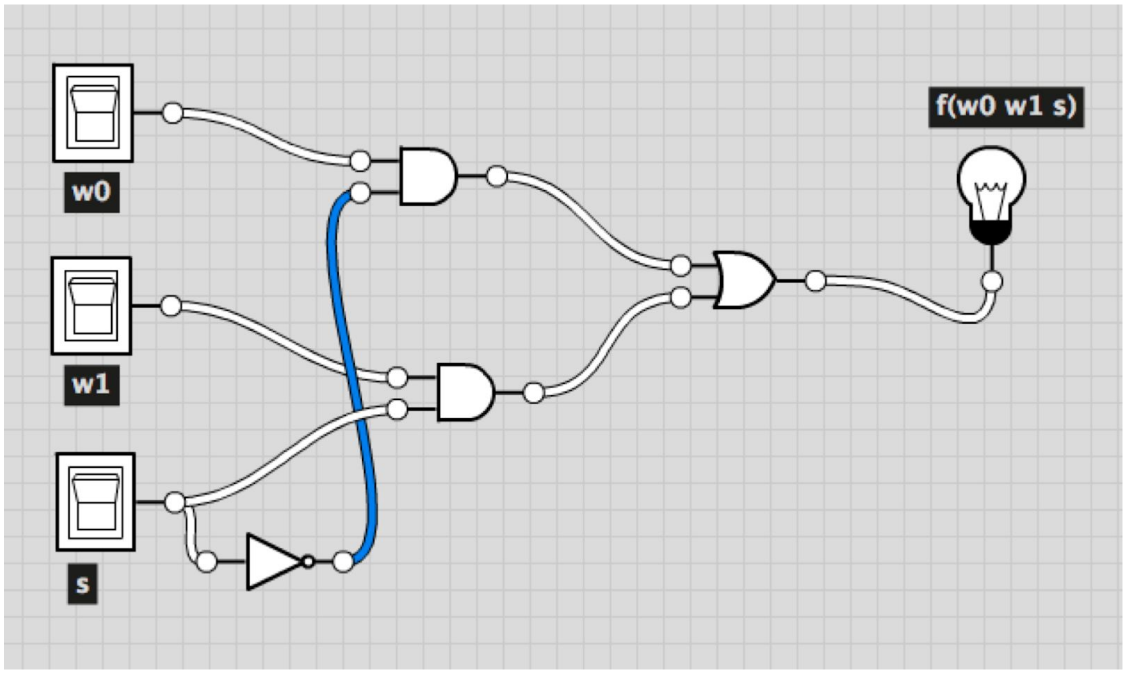
In this lab, we were able to successfully create multiplexers from logic gates and a larger multiplexer from the smaller multiplexers. We used more of the Altera Quartus II and by now are much more familiar with how the program works. We built a 2:1 MUX from a logic circuit consisting of one NOT gate, two AND gates, and one OR gate. We then used the 2:1 MUX to create an 8:1 MUX, and used that 8:1 MUX to create a circuit that outputs "1" whenever an odd number of inputs are "1", and "0" whenever an even number of outputs are "1". We tested all the circuits by using the waveform simulation and uploading to the FPGA board.

Truth Table 1: 2:1 MUX

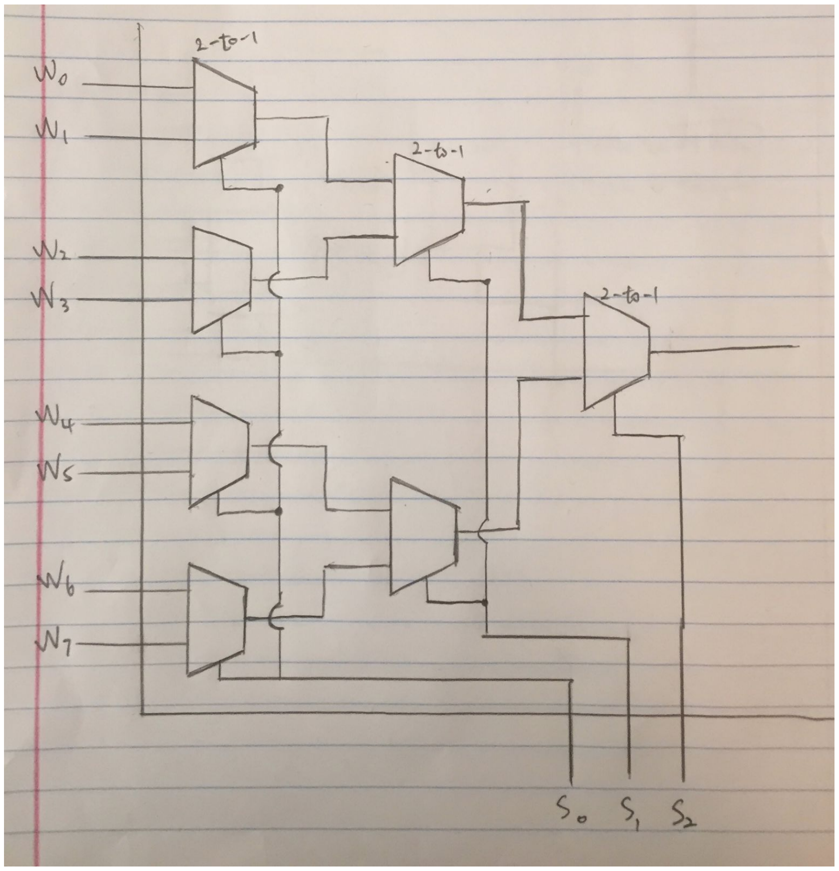
w0	**w1**	**s**	**f(w0, w1, s)**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

We used the logic from our truth table to build a 2:1 MUX. The design we created for our pre-lab was the same design that we ultimately implemented in lab.

Schematic 1: 2:1 MUX



Schematic 2: 8:1 MUX



We created an 8:1 MUX using 2:1 our MUX in a hierarchical manner.

III. **Procedure:**

Part 1: Circuit 1 - Multiplexer as a switch

In order to implement our design of a 2:1 MUX, we had to refer to our truth table and schematic from the pre-lab (Truth Table 1 and Schematic 1). We used input and output pins alongside AND, OR and NOT gates to replicate our system using Altera Quartus II, and manually dragged wires to connect each component (Schematic 3). Next, compiled our schematic and we ran a waveform simulation of our design (Simulation 1). The 2:1 MUX ran successfully when the switch (S) if off, w0 is passed through as the output and when S is on, w1 is passed through as the output. After we confirmed that our design was working correctly, we saved Schematic 3 as a symbol to be used in part 2.

Part 2: Circuit 2 - Hierarchical design of Multiplexers

Next, we used the 2:1 MUX that we built in part 1 to construct a 8:1 MUX (Schematic 4). We were able to load the 2:1 MUX as a symbol (similar to the AND, OR, and NOT gates) and layer them in a hierarchical manner. The first tier of 2:1 MUXs determines the least significant bit of the output, the second tier determines the middle bit of the output, and the third tier determines the most significant bit of the output. We then compiled our schematic and ran a wave simulation of our new design (Simulation 2 & Simulation 3). We had to test each combination of switches individually to ensure that the binary number created by the switches (s2, s1, s0) matched the input that was let through. In Simulation 2, s0, s1, and s2 are all 0, creating the binary number 000. Therefore, the output of the MUX is w0. In Simulation 3, s2 is 0, s1 is 1, and s0 is 1, creating the binary number 011. Therefore, the output of the MUX is w3. After we confirmed that our design was working correctly, we added an enable switch (Schematic 5). The enable switch ran through an 2-input AND gate with the output of the original MUX. When the enable switch was 1, the MUX functioned normally and when the enable switch was 0, the output was 0. We again tested our design using a waveform simulation. As demonstrated in Simulation 4, an enable switch of 0 forces the output of the MUX to be 0.

Part 3: Circuit 3 - Multiplexer as a logic building block

Finally, we saved our 8:1 MUX as a symbol and used it to create a circuit that had a output of 1 when an odd number of inputs were 1, and had an output of 0 when an even number of inputs were 1 (Schematic 6). Inputs w0 - w7 represent the 8 possible outputs of the circuit. We wired w1, w2, w4 and w7 to VCC because they correlate to the rows of the truth table that cause the output of the circuit to be 1. Likewise, we wired w0, w3, w5 and w6 to ground, because they correlate to the rows of the truth table that cause the output of the circuit to be 0. Therefore, whatever binary number is created with switches s2s1s0 will match the input that is let through. Once again we tested our schematic using a waveform simulation (Simulation 5 and Simulation 6). In Simulation 5 and Simulation 6, pin_name1 represents the output, sw[0] represents s0, sw[1] represents s1, sw[2] represents s2, and sw[3] represents the enable switch.. In Simulation 5, s2s1s0 = 001, so the output is w1. In Simulation 6, s2s1s0 = 011, so the output is w3.

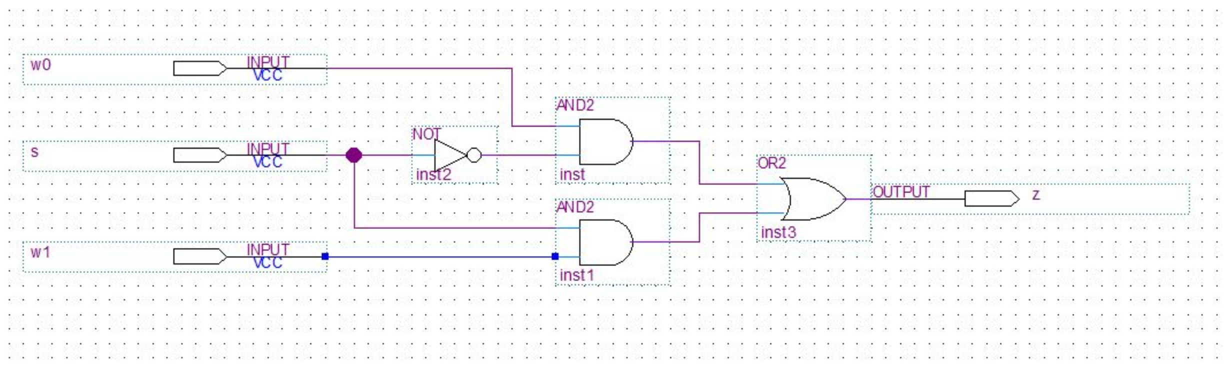
After we had confirmed that all 8 possible combinations of s2, s1, and s0 were working correctly, we downloaded our circuit onto our FPGA and successfully demonstrated it to the TA.

IV. Mistakes that we make:

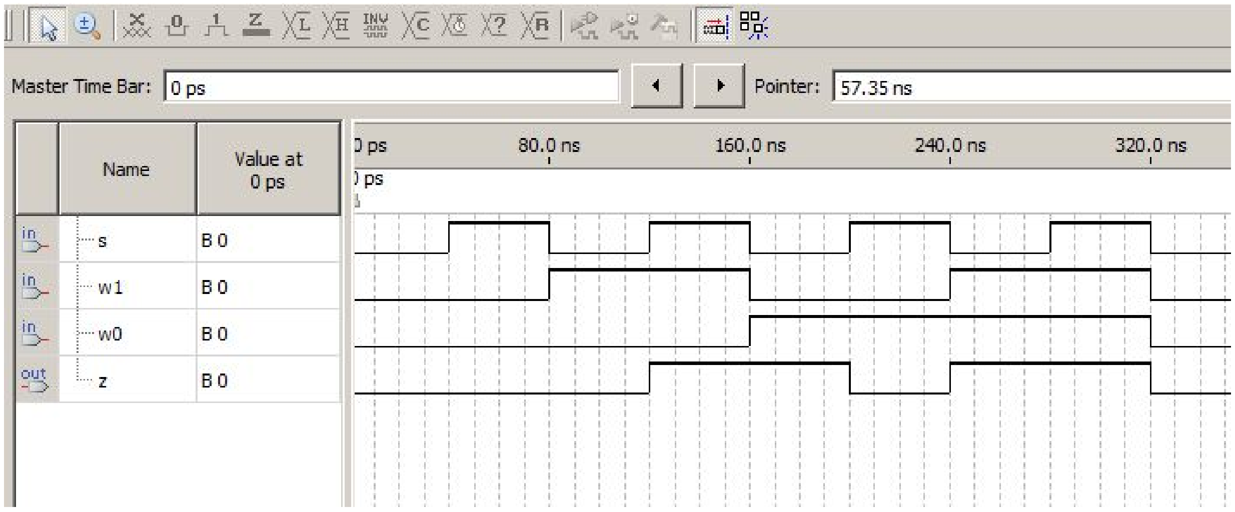
At first, we had trouble testing our 8:1 MUX as a waveform simulation. We set w0, w2, w4 and w6 to be 0, w1, w3, w5 and w7 to be one and s0, s1, and s2 to be random. We realized that this didn't work, because we couldn't differentiate between inputs that were set to the same value and therefore couldn't tell which input was being passed through to the output. Instead, we set w0 - w7 to be random and set s0, s1, and s2 to some combination of 0 and 1. Then we were successfully able to determine if our design was working by matching the waveforms of the input and output.

Furthermore, we originally didn't know how to hard wire out inputs of the even-odd circuit to VCC and ground. We thought that we had to use inputs and then manually set their values to be 0 or 1 when we were testing the design as a waveform simulation. We then realized that VCC and ground were input options and hard-wired w0-w7 to them instead.

Schematic 3: 2:1 MUX using Altera Quartus II

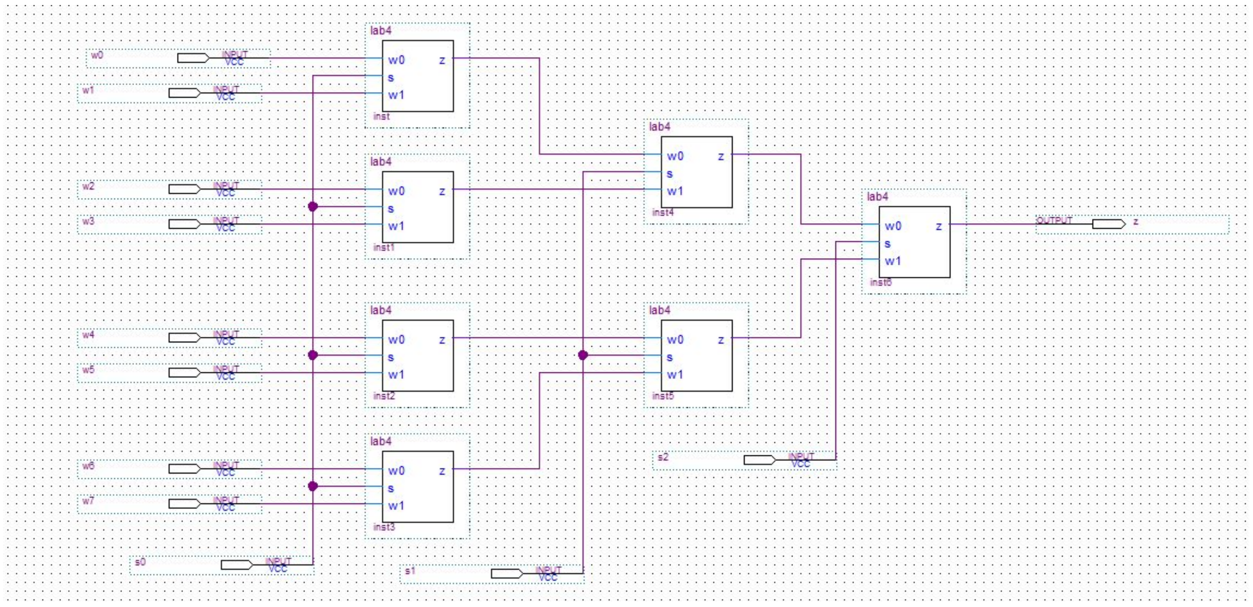


Waveform Simulation 1: 2:1 MUX

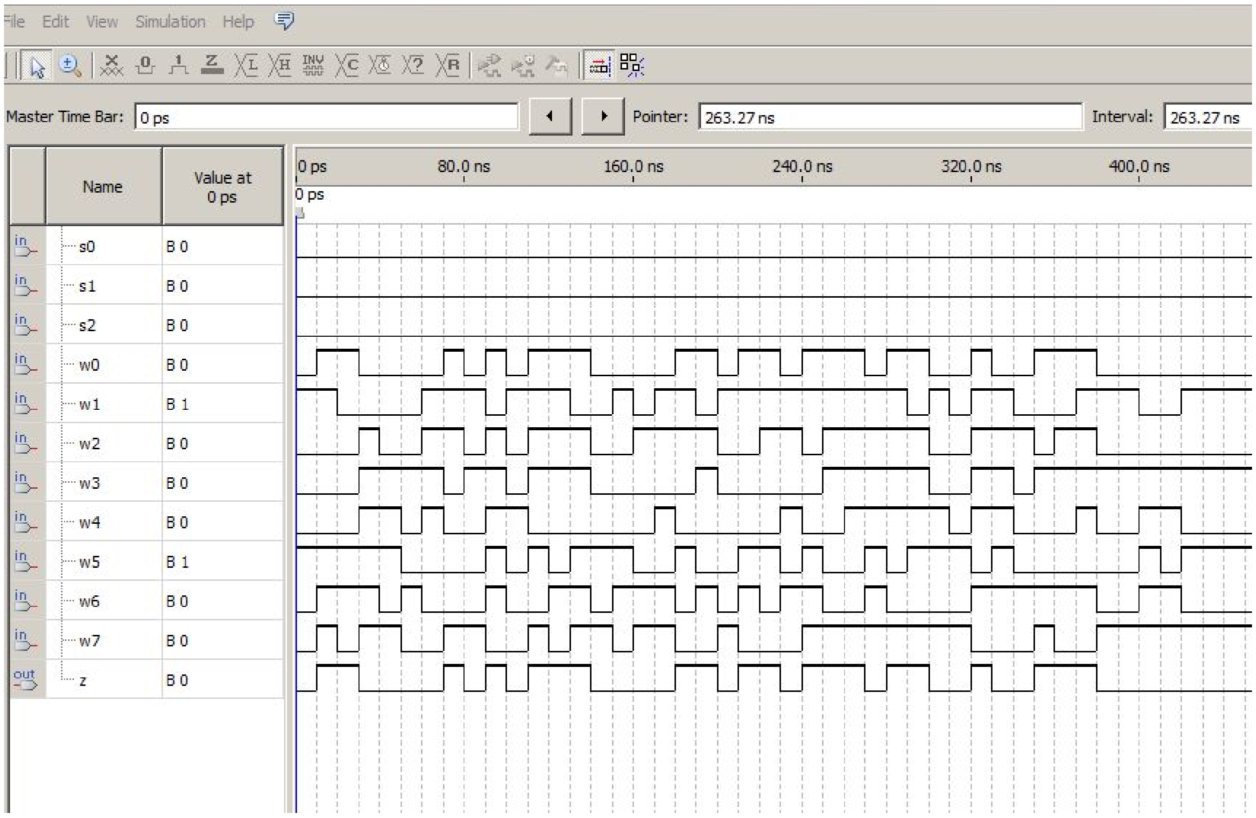


Name	Value at 0 ps
s	B 0
w1	B 0
w0	B 0
z	B 0

Schematic 4: 8:1 MUX using Altera Quartus II



Waveform Simulation 2: 8:1 MUX with w0 as Output



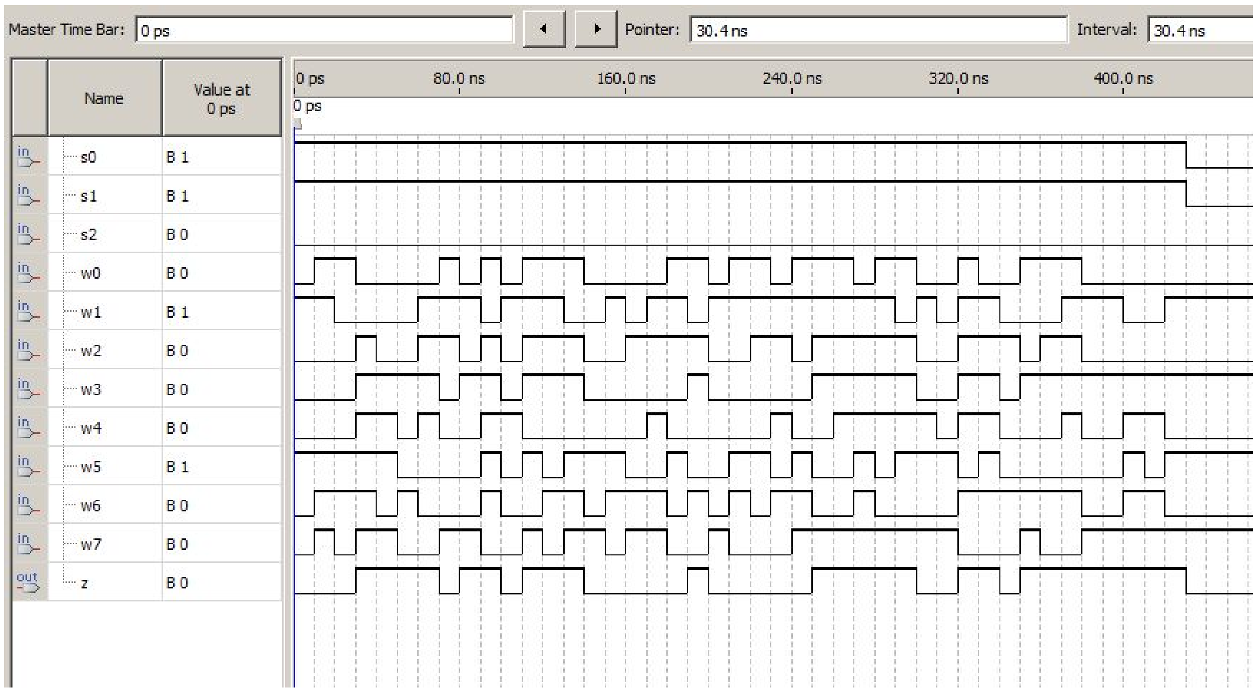
Master Time Bar: 0 ps

Pointer: 263.27 ns

Interval: 263.27 ns

	Name	Value at 0 ps
in	s0	B 0
in	s1	B 0
in	s2	B 0
in	w0	B 0
in	w1	B 1
in	w2	B 0
in	w3	B 0
in	w4	B 0
in	w5	B 1
in	w6	B 0
in	w7	B 0
out	z	B 0

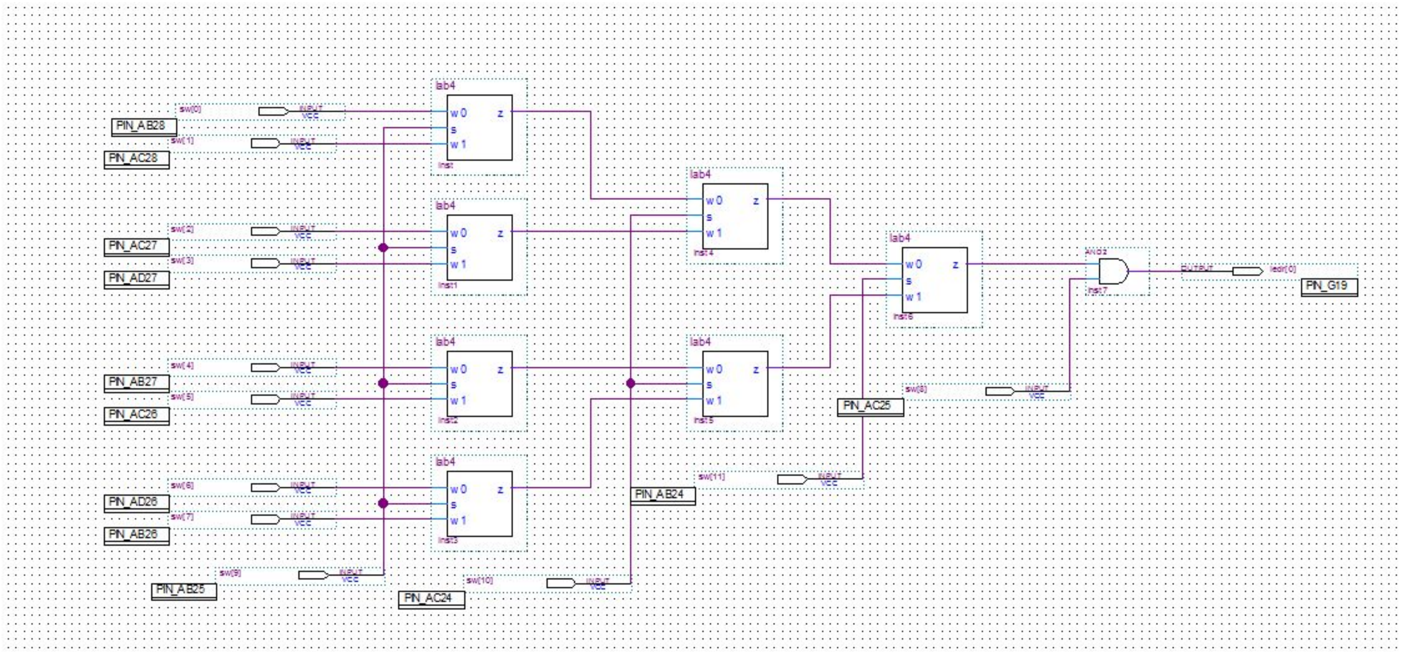
Waveform Simulation 3: 8:1 MUX with w3 as Output



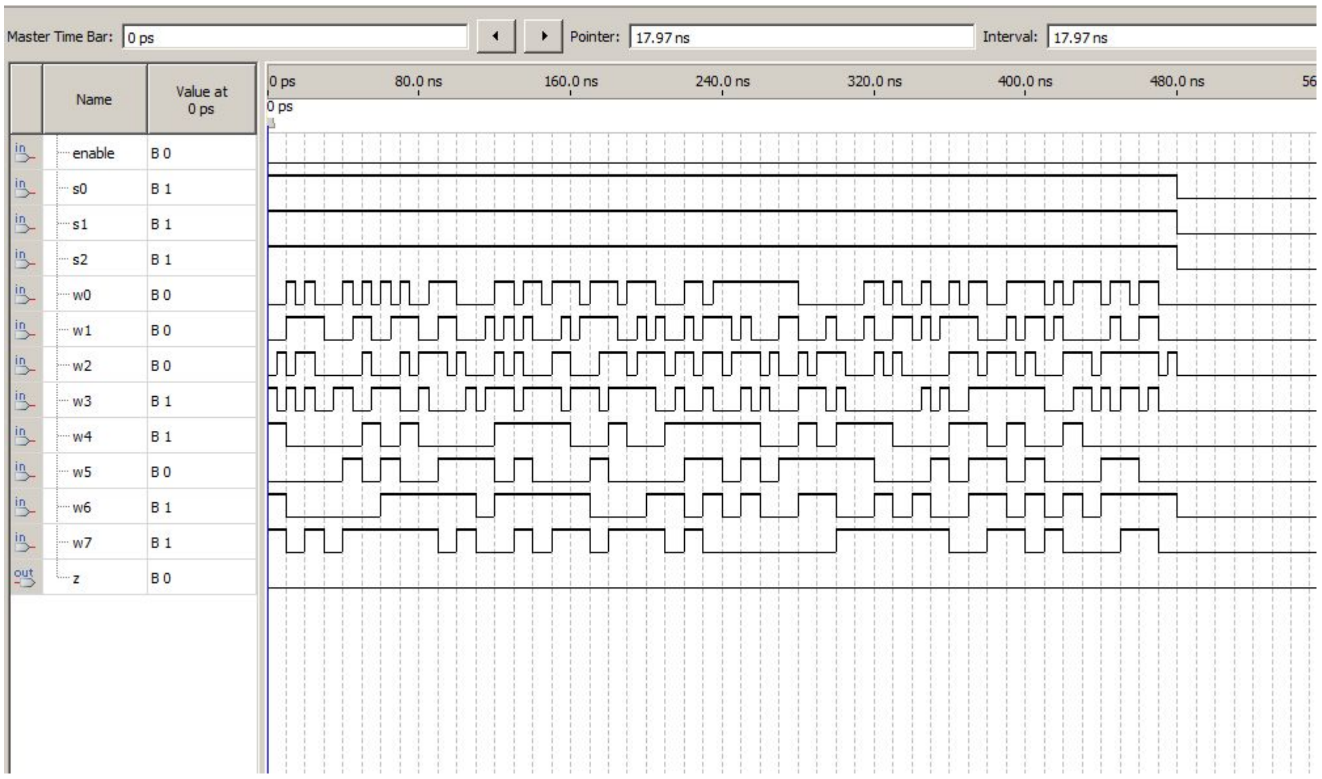
Master Time Bar: 0 ps   Pointer: 30.4 ns   Interval: 30.4 ns

	Name	Value at 0 ps
in	s0	B 1
in	s1	B 1
in	s2	B 0
in	w0	B 0
in	w1	B 1
in	w2	B 0
in	w3	B 0
in	w4	B 0
in	w5	B 1
in	w6	B 0
in	w7	B 0
out	z	B 0

Schematic 5: 8:1 MUX using Altera Quartus II with Enable Switch [sw8]



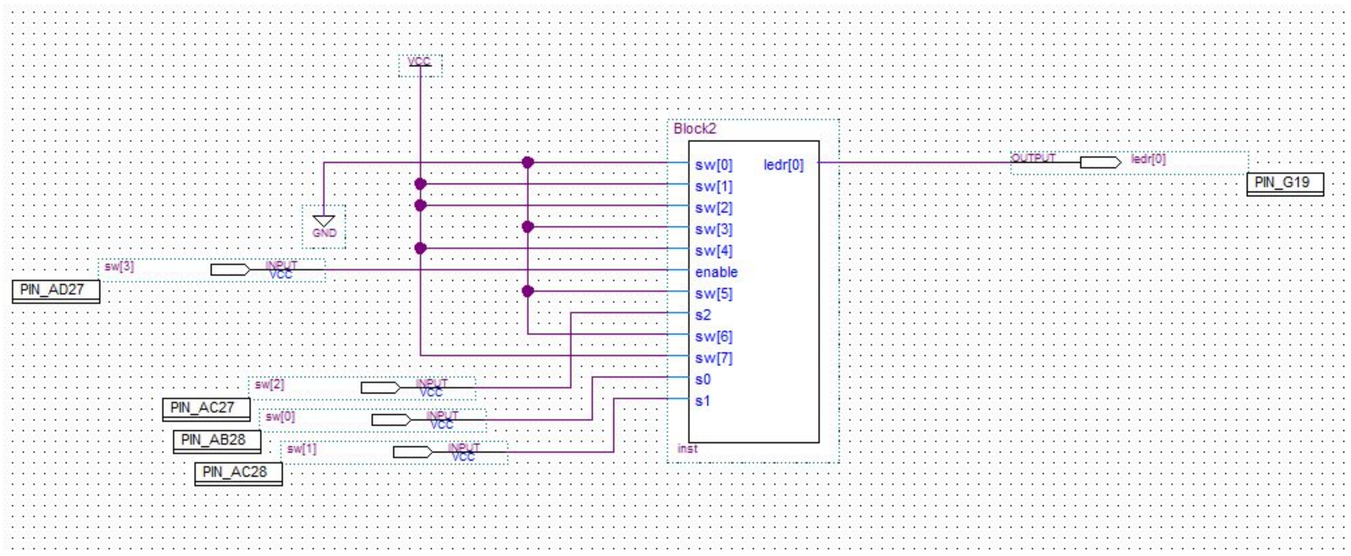
Waveform Simulation 4: 8:1 MUX with Enable Switch at 0



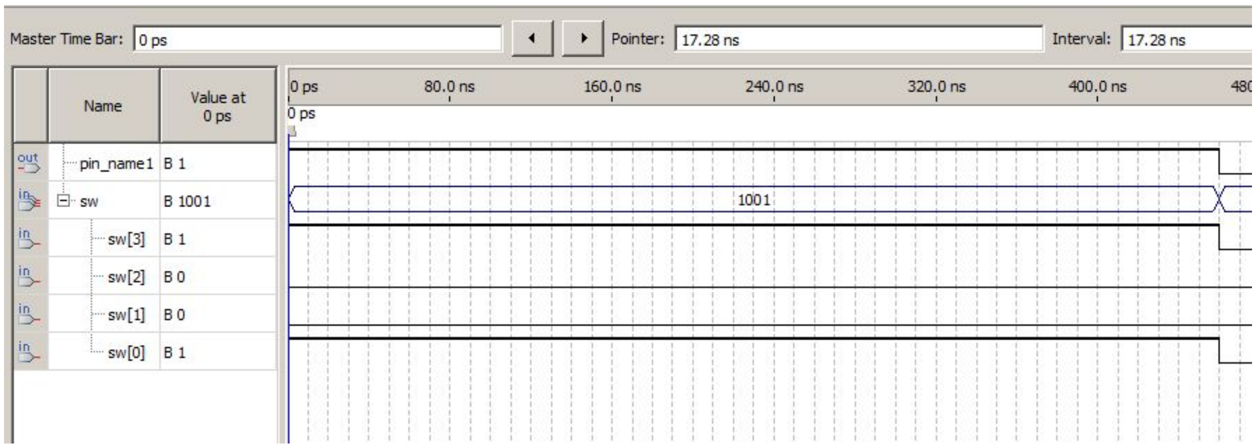
Master Time Bar: 0 ps
Pointer: 17.97 ns
Interval: 17.97 ns

Name	Value at 0 ps
enable	B 0
s0	B 1
s1	B 1
s2	B 1
w0	B 0
w1	B 0
w2	B 0
w3	B 1
w4	B 1
w5	B 0
w6	B 1
w7	B 1
z	B 0

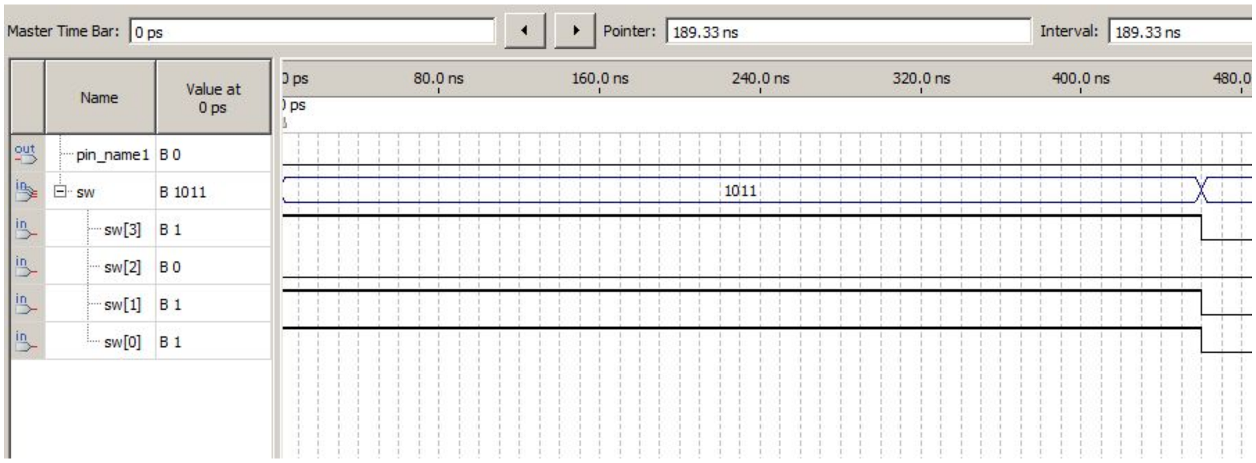
Schematic 6: Even-Odd (XOR) Circuit using Altera Quartus II



Waveform Simulation 5: Even-Odd (XOR) with s2s1s0 = 001



Waveform Simulation 6: Even-Odd (XOR) with s2s1s0 = 011

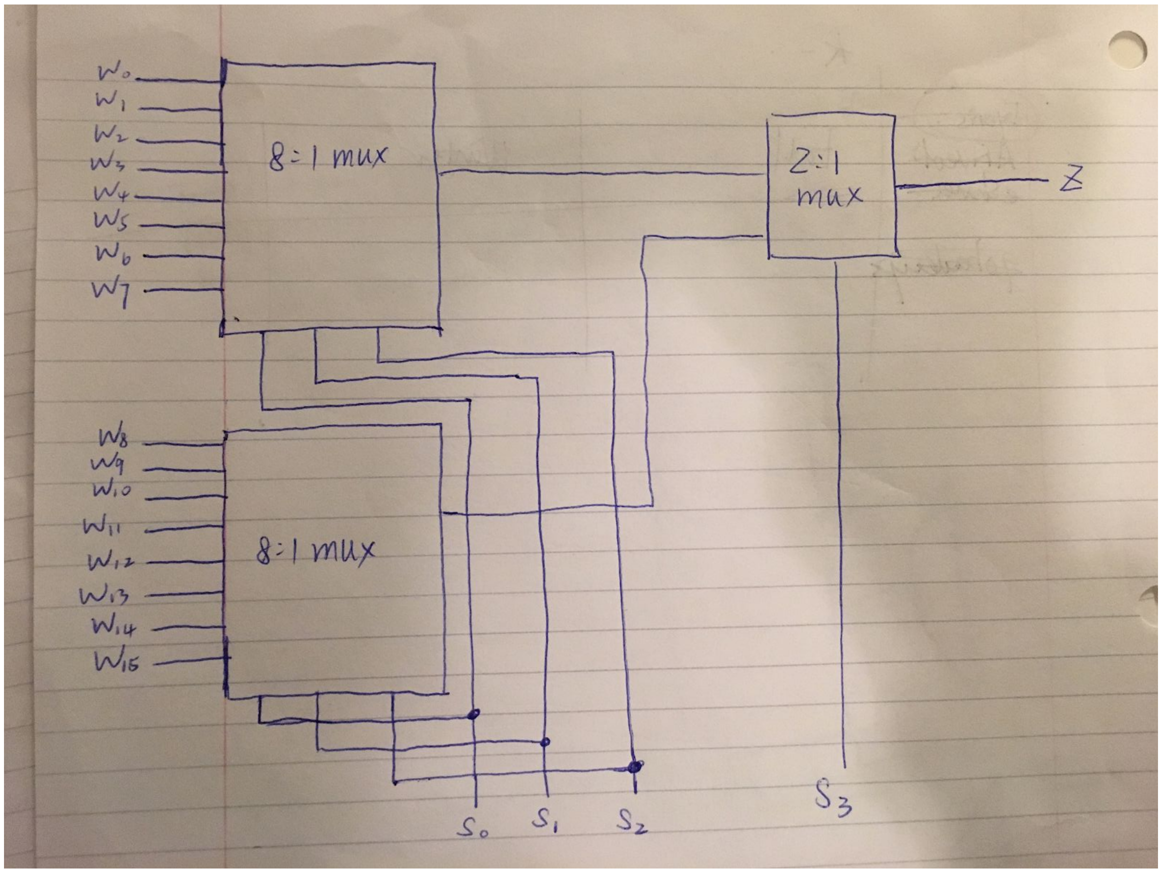


V. <u>Final question:</u>

1. Explain how you tested the operation of the 8:1 MUX. If you found problems with the circuit that required correction and a new download, explain what you observed that indicated there was a problem and describe how you fixed it.

First, we assign enable to be 1, so the output can be what we want it to be.
Then, we assigned numbers to s2, s1, and s0, for example, we let s2s1s0 be 101, which indicates 5 in decimal, which means the output will be w5.
Next, we generate the pattern all inputs randomly.
At last, in the waveform, the output is the same as the pattern of w5.
We designed the circuit perfectly, so we don't have any problem.

2. Show how you would design a 16:1 MUX using 8:1 and 2:1 MUXes.



I will design the 16:1 mux like this, with two 8:1 muxes and one 2:1 mux, and 4 switches.
For example, if the switches are 1011, the output will be w11.
If the switches are 0000, the output will be w0.

VI. Conclusion

The lab achieves its objective of familiarizing us with creating multiplexers in Quartus and using those multiplexers to create a switch connecting different inputs and creating larger multiplexers with smaller ones. We learned how to create symbols in Quartus and how to use those symbols in other schematics. Overall, the lab was a great learning experience for us in Quartus.

VII. Reference lists:

- Dr. Sally Wood, Dr. Samiha Mourad, Dr. Radhika Grover. *Laboratory #4: Multiplexer Design* . Spring 2017. Print
- ftp.altera.com/up/pub/Altera_Material/12.1/Tutorials/Schematic/Quartus_II_Introduction.pdf
- ftp.altera.com/up/pub/Altera_Material/13.1/Tutorials/Verilog/Quartus_II_Simulation.pdf
- ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2-115_User_Manual.pdf