Chisom Adabanya
101142680

Lab 1 – Multiplexers in Multisim

Lab Section: L1O – Monday 2:30pm - 5:30pm

1.0 Introduction

The main purpose of this lab was to design a mux-demux circuit, operating as a type of telephose switch. It was built with the aim of handling 4 phone calls and give out 8 outputs at the other end of the circuit. This is done using logic gate combinations. These combinations help form a multiplexer and a demultiplexer.

The multiplexer handles the 4 input phone calls and the demultiplexer handles the 8 listening output receivers. The multiplexer and demultiplexer were first designed in the prelab.

After the mux-demux logic circuits were designed in the prelab, each individual logic gate was specifically placed to ensure the right outcome. The design done during the prelab was then successfully tested during the lab using multisim and the results of these labs were described using the LED light symbols in multisim. Every possible input was tested and comparing the lights to the truth table shows that the logic gate combinations were correct.

2.0 Specifications

There are a total of 9 inputs (counting the S_0 and S_1 signals, as well as the P_0, P_1 and P_2 signals), and a total of 8 outputs. The 4 actual inputs were labelled W through Z and the 8 outputs were labelled A through H. All the inputs and outputs were 1-bit binary numbers, either 0 or 1. The design was simulated with multisim, and the inputs and outputs were set from subcircuits in which the logic gates and the sources were placed. There are very few limitations, whether size (only that I was running out of space on the multisim page for the demultiplexer because it needed so many gates), power, time, or any of the sort.

3.0 Design

Firstly, I thought of how many inputs we needed (4) and this guided my choices. I decided to use several 2-input AND and OR gates as this was the most convenient way to begin the design. Then the next problem was deciding how many control signals I needed to successfully implement these gates. I found that I needed 2 control signals: S_0 and S_1. The signal S_0 had 2 inverted inputs and 2 regular inputs and the signal S_1 had 1 regular and 1 inverted output. This was hand drawn and is shown in figure 1 below:

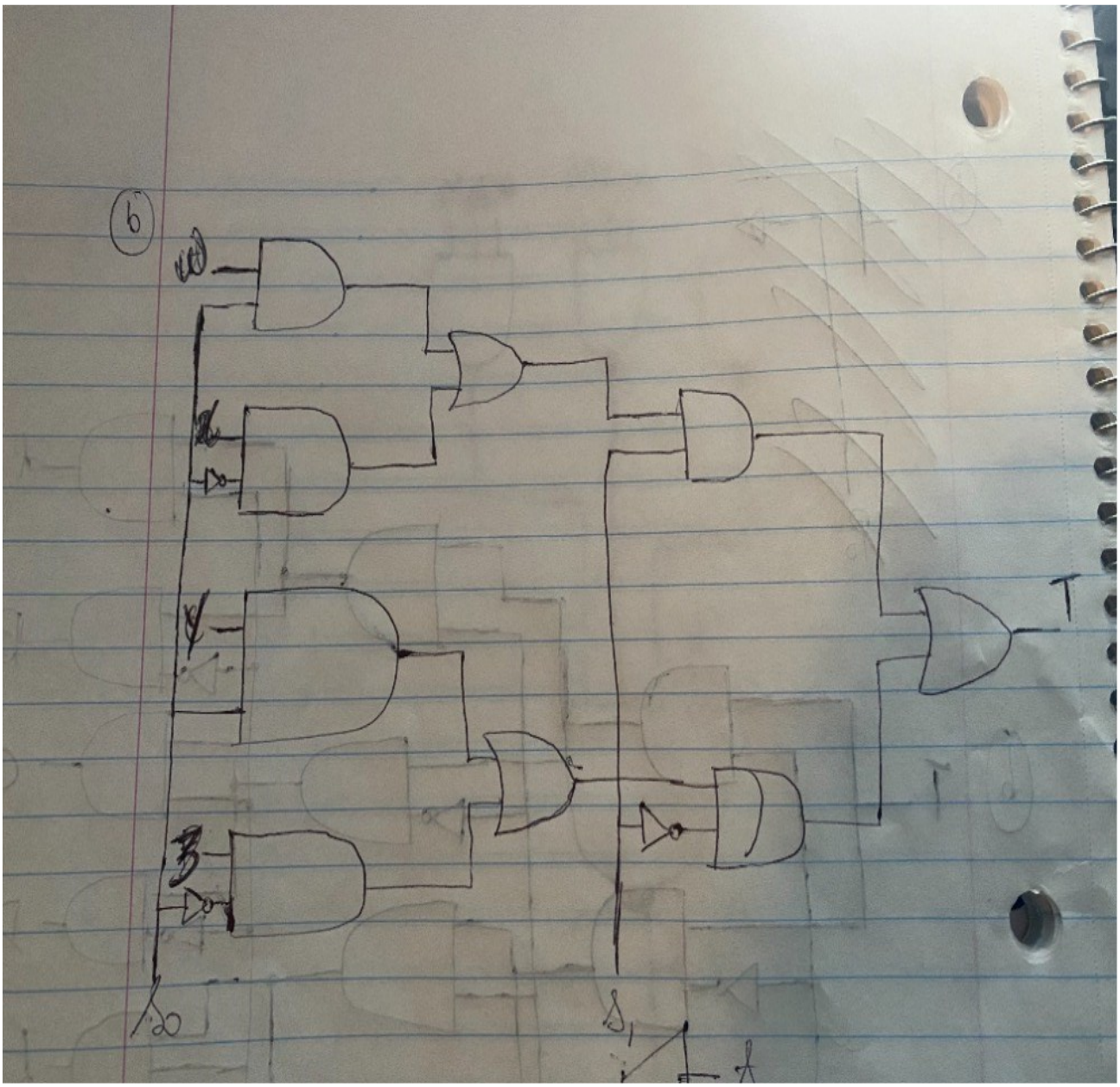

Figure 1: Multiplexer with And and OR Gates

These gates, along with their control signal inputs (S_0 and S_1) made it possible to achieve 1 output (T) from 4 inputs (W, X, Y, Z) using a multiplexer.

The completed logic gate design to be implemented into multisim had to go through DeMorgan's laws of conversion first. The end result after all the permutations meant that I was able to create 4 3-input NAND gates. The control inputs had the test inputs on a 4-bit truth table; The input S_0 having the values (0,0,1,1) and S_1 having the values (0,1,0,1) in order to properly represent all possible outputs from the 4 original inputs. This is shown in figure 2 below:

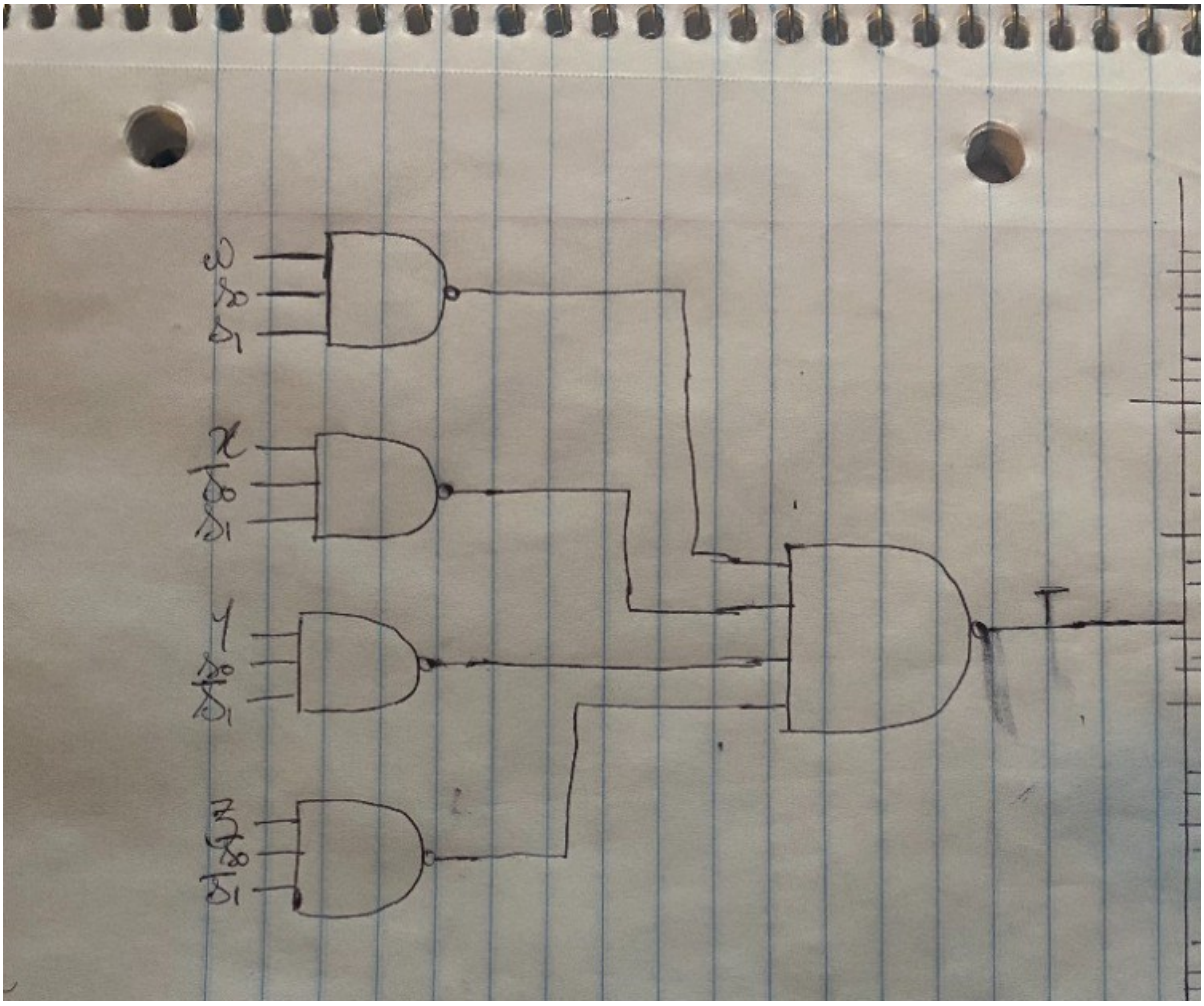

Figure 2: Multiplexer with NAND Gates

After the multiplexer was completed, I then began to build the demultiplexer circuit, originally with one input (T) and 3 control signals (P_0, P_1 and P_2) using AND gates. The number of required outputs guided my decisions, and the final results of the constructed logic circuit with AND gates is shown in figure 3 below:

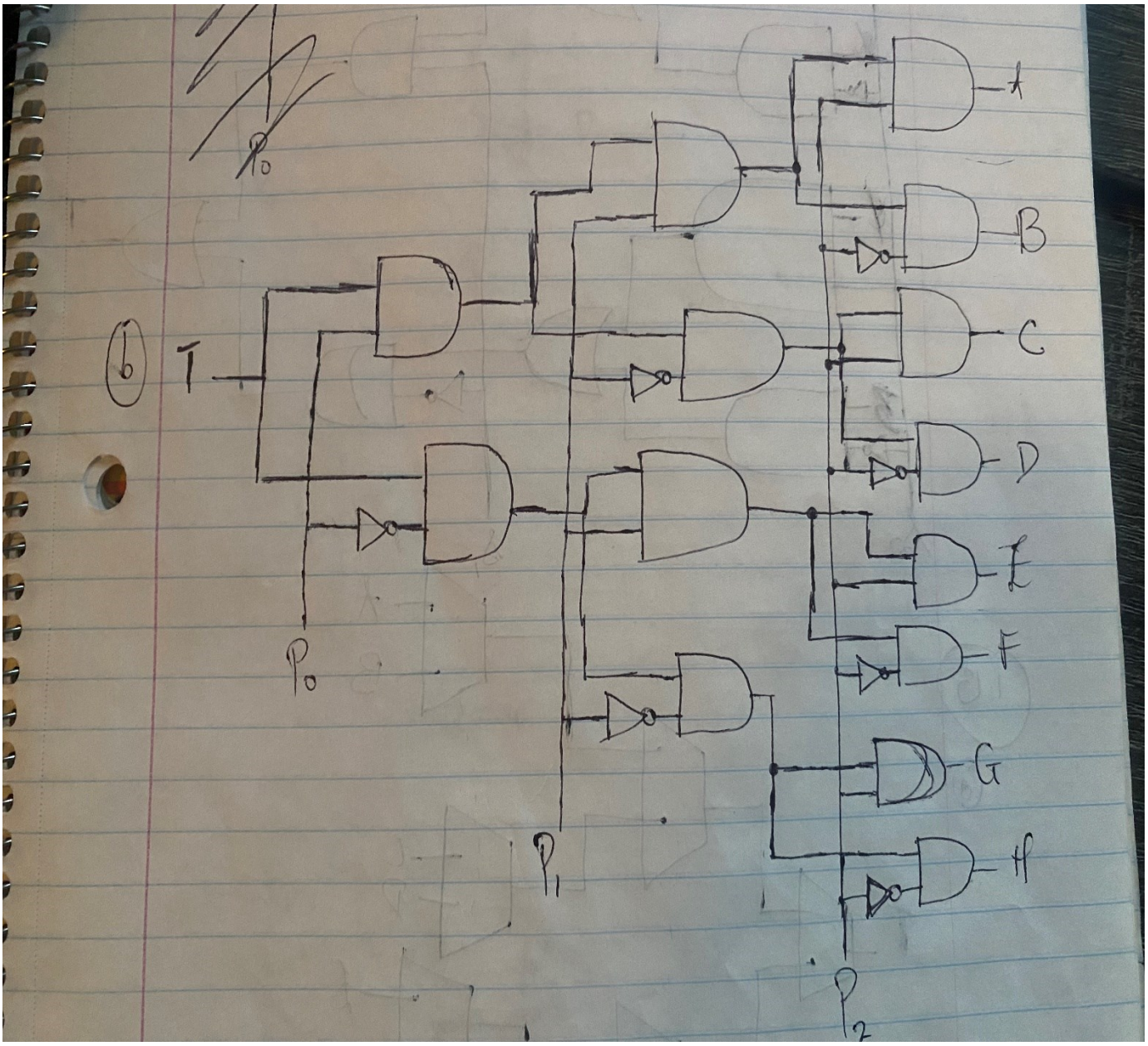

Figure 3: Demultiplexer with AND Gates

After getting the circuit with AND gates, I then proceeded to convert the gates using DeMorgan's theorem and this resulted in 8 separate 4-input NAND gates. The 3 control inputs had the testing values of a test 8-bit, 3-input truth table in order to achieve all possible outputs; The input P_0 had values of (0,0,0,0,1,1,1,1), The input P_1 had values of (0,0,1,1,0,0,1,1) and the input P_2 had the values (0,1,0,1,0,1,0,1). This is shown in figure 4 below:

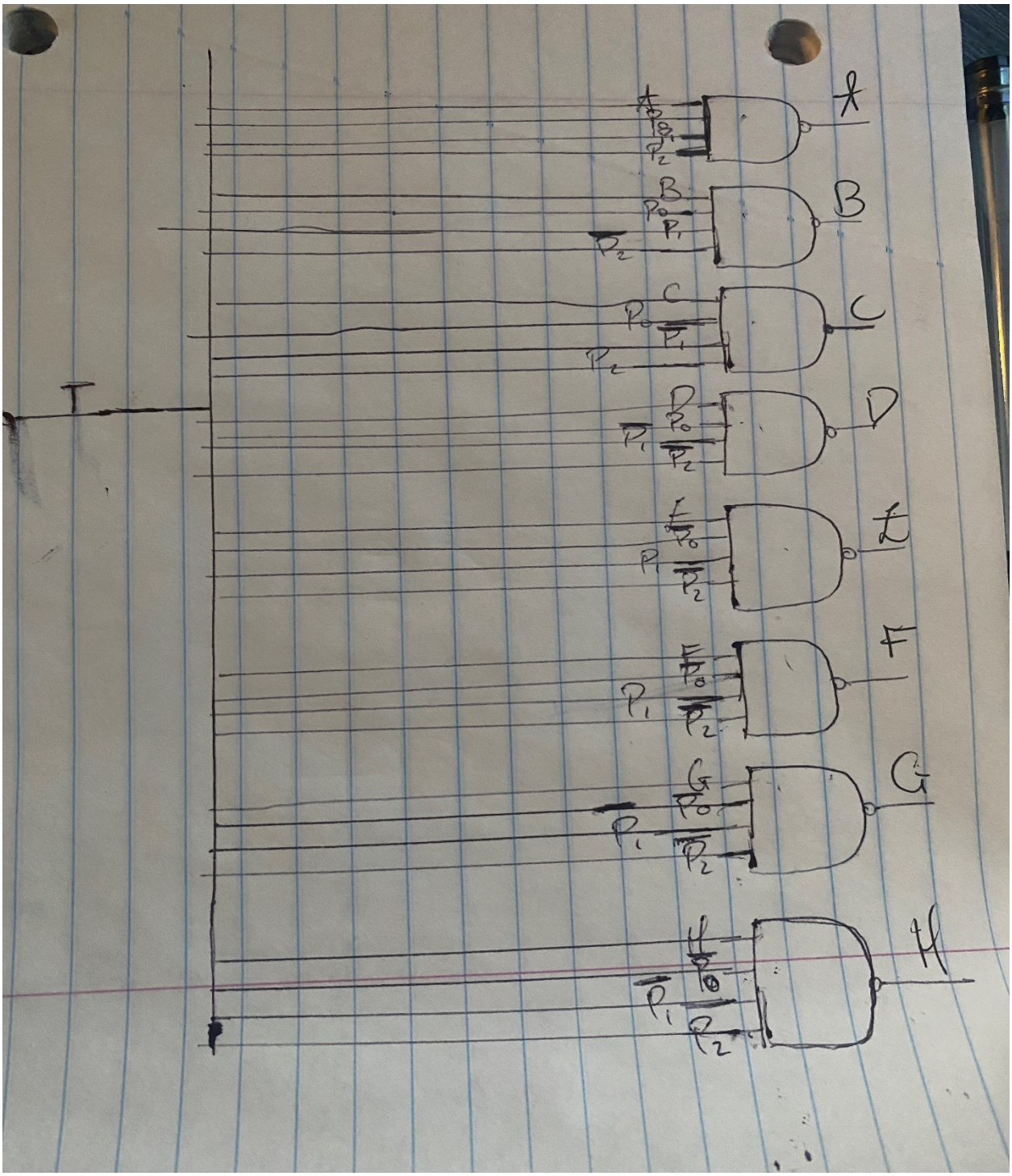

Figure 4: Demultiplexer with NAND Gates

4.0 Implementation and Testing

In order to implement the gates properly, I had to go over the designed gates a bit, and smooth things over, as the design process was a little rough. I used inverters to represent the 1 value and the regular connections to represent the 0 value. The design, at the NAND gate level, was implemented through multisim and the figures 5 and 6 below show how multisim subcircuits allow the implementation of logic gate combinations to get a single output or multiple outputs.

Figure 5 below shows how I implemented the 3 input NAND gates, as the subcircuit within the multiplexer. This was done from the multiplexer design and achieves a single output:

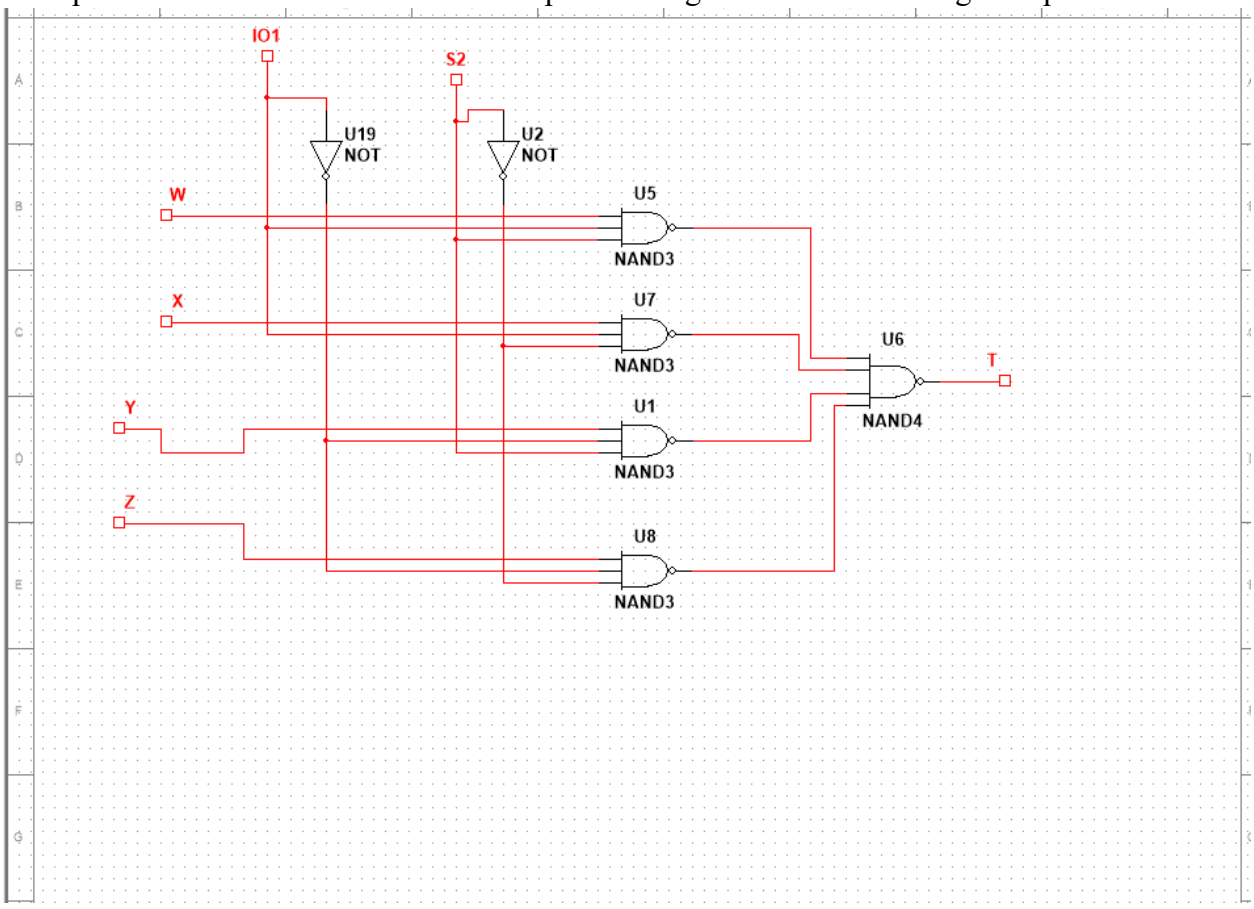

Figure 5: Multiplexer with NAND Gates in multisim

Figure 6 below shows how I implemented the 4-input NAND gates in the demultiplexer as the subcircuit within it. This was done from the demultiplexer design and achieves a 8 separate outputs:

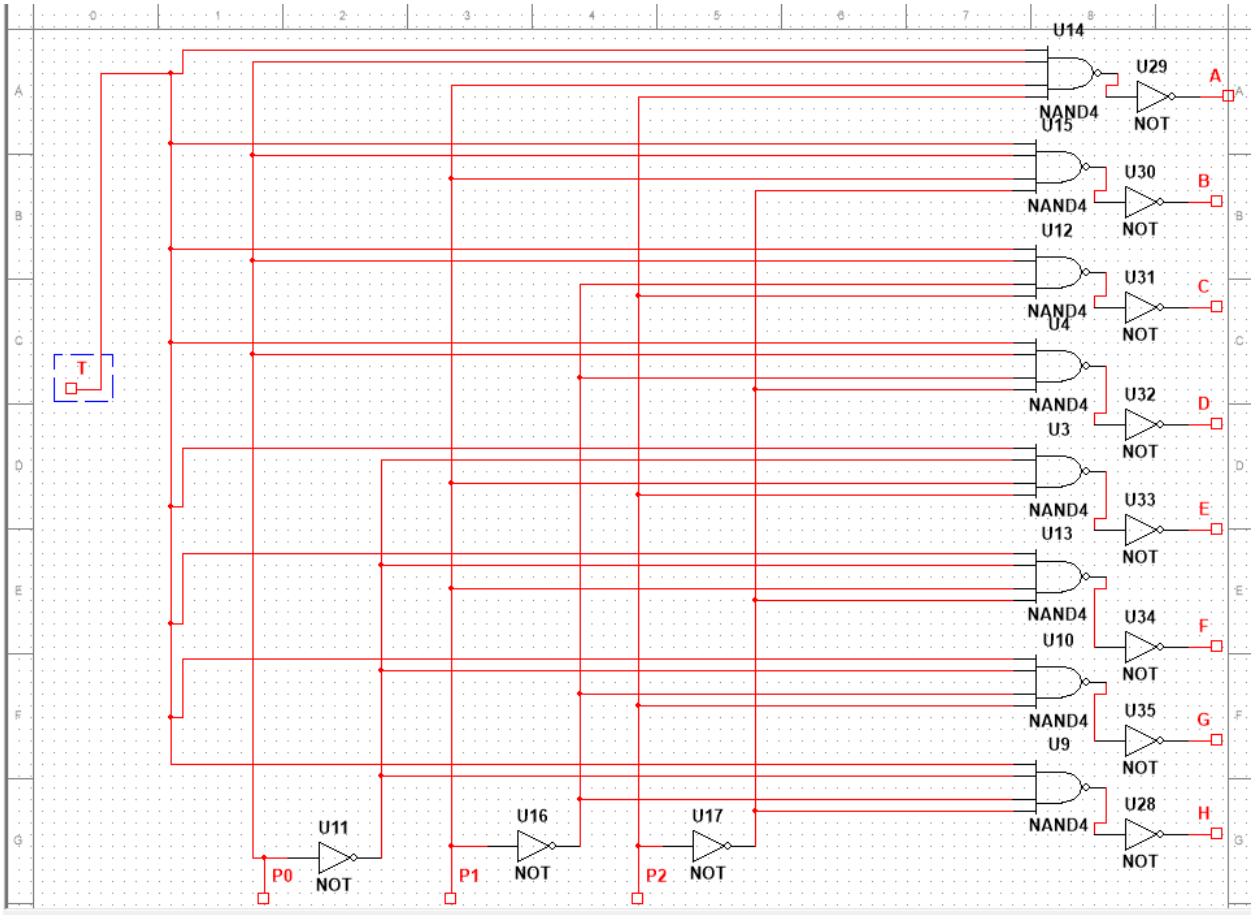

Figure 6: Demultiplexer with NAND Gates in multisim

The circuit was tested with truth tables and the outputs were recorded to ensure that their values were true. I had a few issues with the output at first, but this was because I made an error and did not put in the inverters at the outputs in figure 6. After I did this, the circuit began to work as it was meant to. 2 tests I ran are shown below in figures 7 and 8:

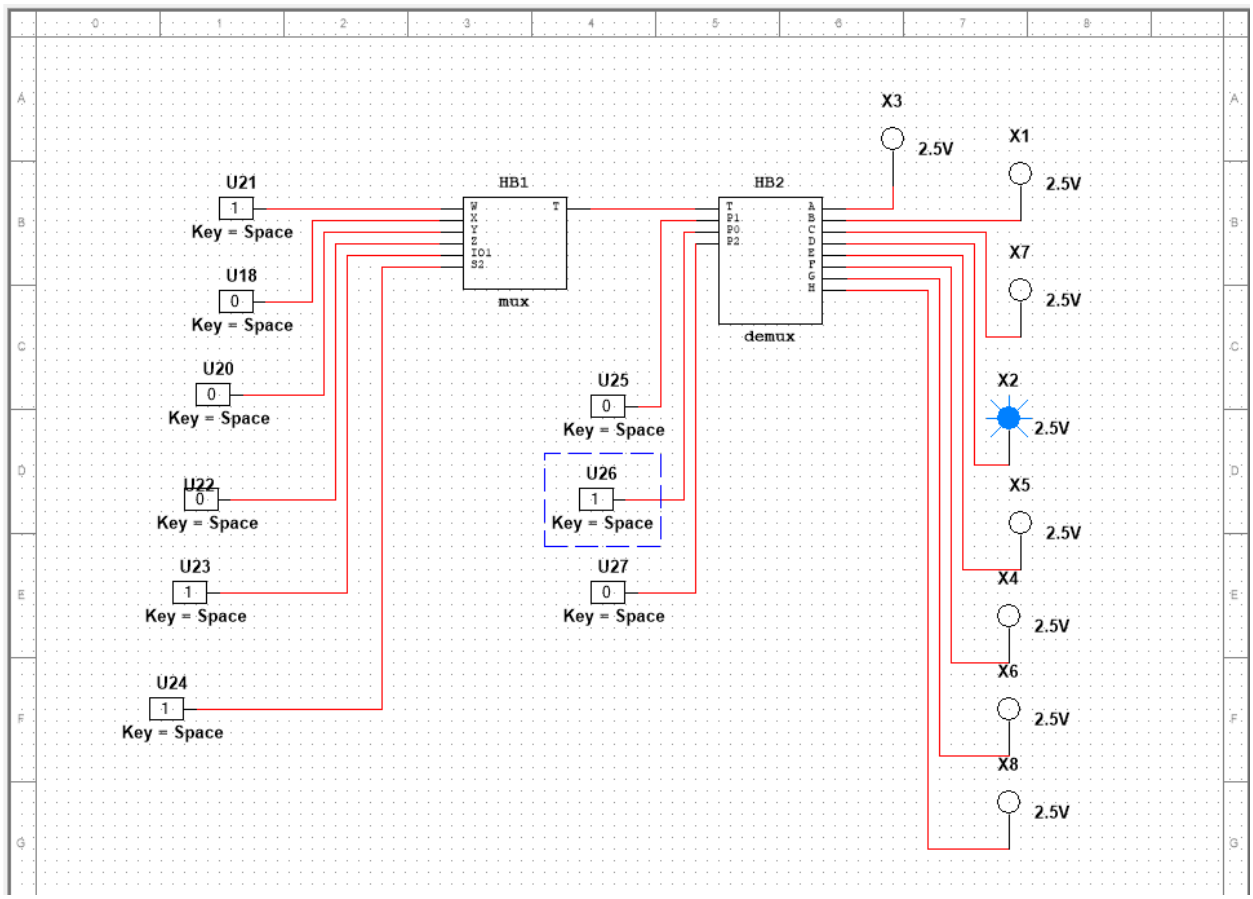

Figure 7: Full Circuit showing 1 output working

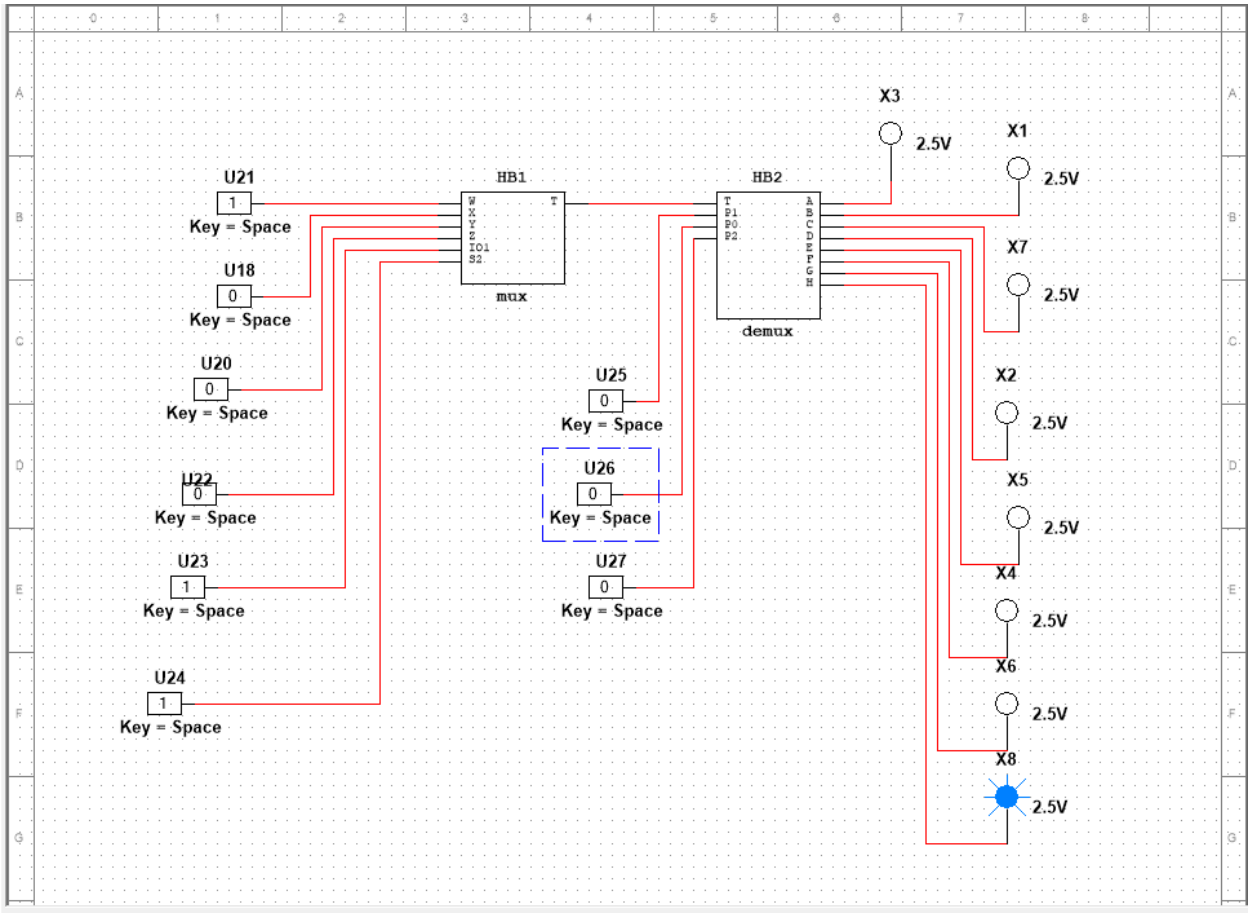

Figure 8: Full Circuit showing a different working output

5.0 Conclusion

The main purpose of this lab was to design a circuit that could operate very similarly to a telephone switch. This was achieved with the use of a multiplexer and a demultiplexer, both constructed with logic circuits and combinations that were built into their subcircuits. They were built to handle 4 phone calls (inputs) and be able to give outputs to 8 receivers at the other end of the circuit. In order to simulate this properly, multiple arbitrary control signals were used in both the multiplexer (S signals) and the demultiplexer (P signals) in order to simulate all the possible outputs from the circuit.

6.0 Appendix

See Prelab submission under 101142680.