Abstract:

The objective of this lab was to construct a circuit using Multiplexers (MUX) and Demultiplexers (DEMUX). The pre-lab was divided into two parts. Part 1 tasked us with creating algebraic expressions for the MUX, DEMUX, and the indicator outputs by using the provided description of a MUX / DEMUX.[2, p. 5][3]. In part 2 we completed truth tables for the 5 inputs and 3 outputs, then created algebraic expressions from the truth tables.[2, p. 3, 13-15] Filling in the big truth tables allowed us to create individual K=Maps which were used to derive equations for each input.[2, p. 6] Overall the implementation of the circuits went smoothly and the coding of the board was moderately challenging.

Introduction:

The MUX and DEMUX are special microchips that contain multiple logic gates. A MUX is a basic microchip that has two types of inputs, data inputs and selector inputs, resulting in a single output. The data inputs contain data from the data sources, whereas the selector inputs select a data input to pass through to the output. For a number of selector inputs there are 2^n available data inputs. A DEMUX is similar to the MUX, but in reverse. There are still two types of inputs, a single data input and selector inputs, but there are multiple outputs. Likewise for DEMUX, for n number of selector inputs there are 2^n outputs. **Figure 1 & 2** contain the circuit diagram for both the MUX and DEMUX.

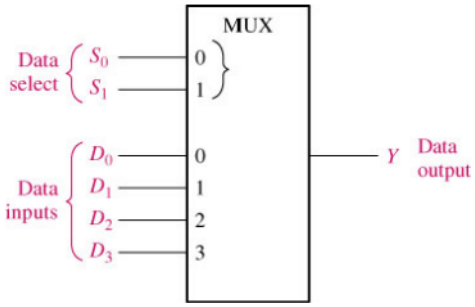
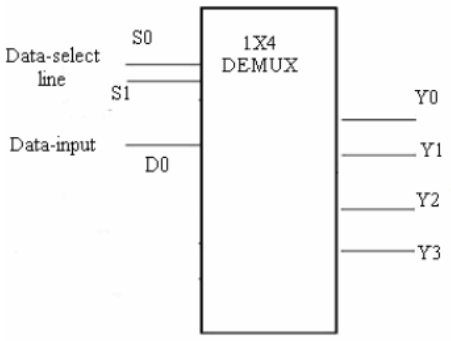


Figure 1↑: MUX Diagram

Figure 2↓: DEMUX Diagram



This lab aimed to understand how the MUX and the DEMUX worked, how to use them to compute and analyze given data, further using the VHDL programming language to create and run the MUX and DEMUX circuits. There are two parts to this lab. Part 1 utilized the algebraic equations found in the prelab to implement a system of MUX and DEMUX that correctly directed three different signals to three different end points. [2, p. 2] In part 2 the objective was to use different algebraic equations found in the prelab to create a system that analyzes given data and outputs the correct response. [2, p. 2] This lab demonstrated how using MUX and DEMUX can be used to significantly simplify circuits. The use of specific selector inputs gives more control in dictating the outcome of a circuit, and allows for major circuit simplification. The key takeaway of this lab is understanding how MUX and DEMUX work, and how to utilize specific selector inputs to direct signal flow towards a desired outcome.

Design Section:

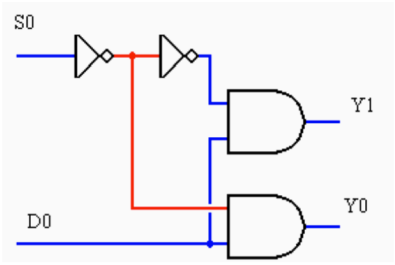
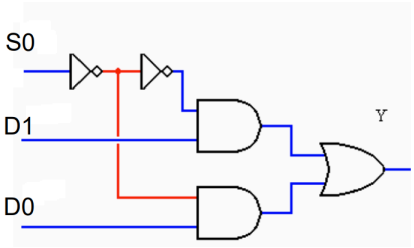


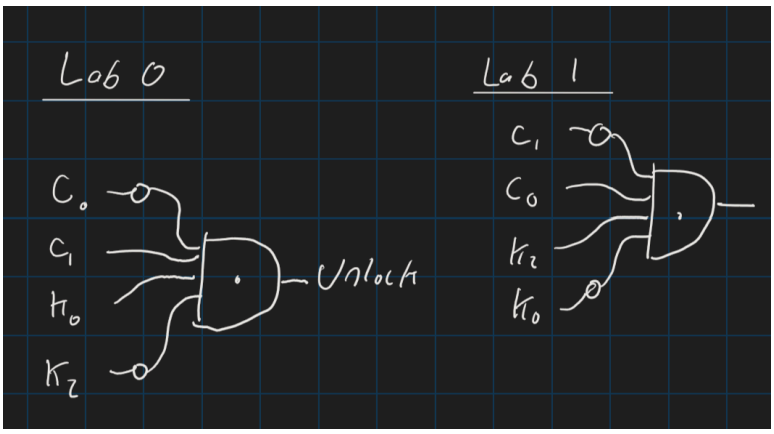
Figure 3↑: MUX Logic Gate Diagram

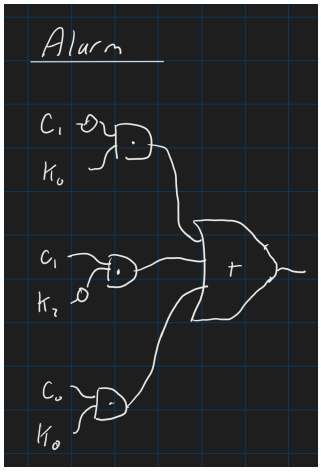
Figure4↓: DEMUX Logic Gare Diagram



For part 1 of the lab we were tasked to use MUX and DEMUX through the VHDL coding environment to control data flow to each endpoint as seen in the lab manual.[2, p. 5] In **Figure 1 & 2** the pin out of both the MUX and DEMUX can be seen. The logic expression and a visual of the internal components of both the MUX and DEMUX is shown in **Figure 3 & 4.** When coding in VHDL we consulted the diagram in the lab manual and concluded that there were three areas that the code needed to be applied.[2, p. [5] Area one was the MUX where we coded: $M = \overline{S(0)}\ \overline{S(1)}\ I(0)+S(0)\ \overline{S(1)}I(1)+\overline{S(0)}\ S(1)$ I(2) This code will select the correct output given the section inputs. The second Area was regarding the outputs of the DEMUX where we coded: $O(0) = \overline{DS(1)}\ \overline{DS(0)}M, O(1) = \overline{DS(1)}\ DS(0)M, O(2) = DS(1)\ \overline{DS(0)}M$. These statements represent each of the different outputs from the DEMUX. For part 2 of the lab we had to impediment the equations that were set up in the pre lab. These equations were found using the truth table and information in the lab manual to make K-maps that then equations could be constructed from.[2, p. 3,6] For the Lab 0 to be unlocked the following expression was coded: $Lab(0)Unlock = \overline{C(0)}\ C(1)K(0)\overline{K(2)}$ For the Lab 1 to be unlocked the following expression was coded: $Lab(1)Unlock = \overline{C(1)}C(0)K(2)\overline{K(0)}$. Finally, the alarm went off when using the following expression: $Alarm = C(1)\overline{K(0)}+C(1)K(2)+C(0)\overline{K(2)}+C(0)K(0)$. Note that if any of the above expressions equals one, the statement is true, otherwise it's false.

Figure 5↓: Lab 0, Lab 1, and Alarm Schematics





Experimental Procedure and Equipment:

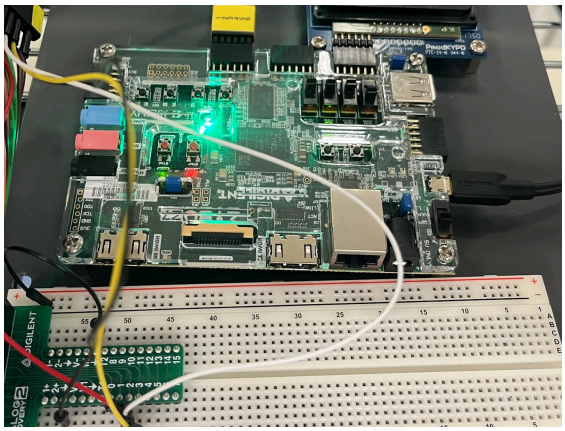


Figure 6↑: Set up for Part 1

Equipment: For this lab, we used the Zybo Z7 FPGA board to implement the designs created. We used the Vivado software for programming and simulating the scenarios and making sure the inputs match the truth table. Analog Discovery 2 was used for providing input signals and some miscellaneous tools such as breadboard, jumper wires were used in order to implement the designs.

Procedure: Firstly we used the template code that was given to us in the lab files to code a 3-input, 1-output MUX and a 1-input, 3-output DEMUX. For the MUX, 2 select signals (S0 and S1) should be used for the selection of one of the 3 inputs (I0, I1 and I2). Similarly, for the DEMUX, one of the 3 outputs (O0, O1 and O2) should be selected based on 2 select signals (DS0 and DS1). We coded the switches to the inputs and the LEDs to the outputs. For the simulation part, we used the test bench file provided to us in the eClass, (lab2_part1_tb), to simulate and verify the outputs match the truth table we completed in pre-lab. After the simulation outputs matched the truth table, we moved on to the implementation part. We synthesized the design and after that connected the Zybo Z7 FPGA board to the computer, and then implemented the synthesis on the board. After that, we toggled different switches on the board to test the outputs.

For part 2, we followed the same procedure of coding, except this time we had to code 2 Card code (C0 and C1) input and 3 keypad code (K0, K1 and K2) input. We defined the outputs of Lab0 Unlock, Lab1 Unlock and Alarm. The conditions that were provided to us were, Lab0 and Lab1 should unlock only when both a valid card code and a keypad code are inputs and the alarm should be activated when an invalid code combination is entered. We wrote a VHDL code, based on the template that was provided to us, mapping the card codes to specific buttons and the keypad codes to specific switches. Then we used the test bench file, (lab2_part2_tb), provided to us in the eClass to simulate the design. After we matched the design to the truth table, we moved on to synthesizing the design and implementing it on the Zybo Z7 board.

Results:

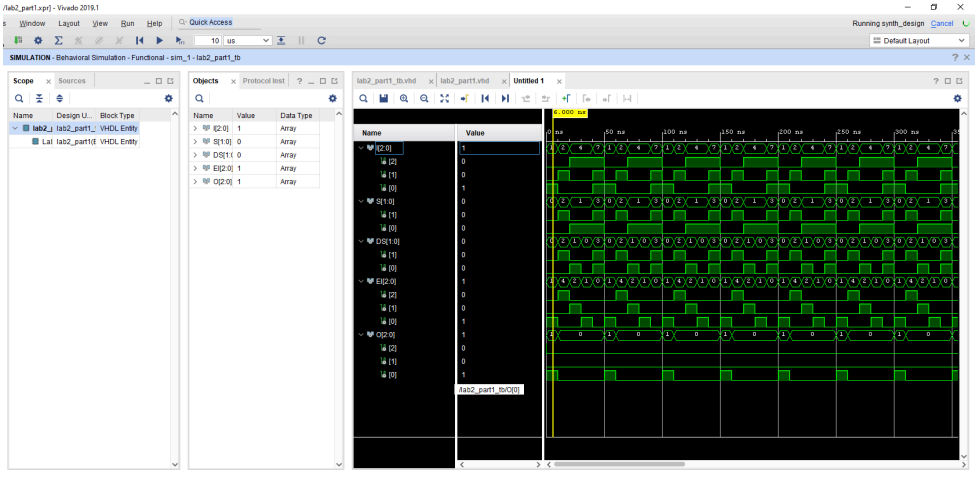
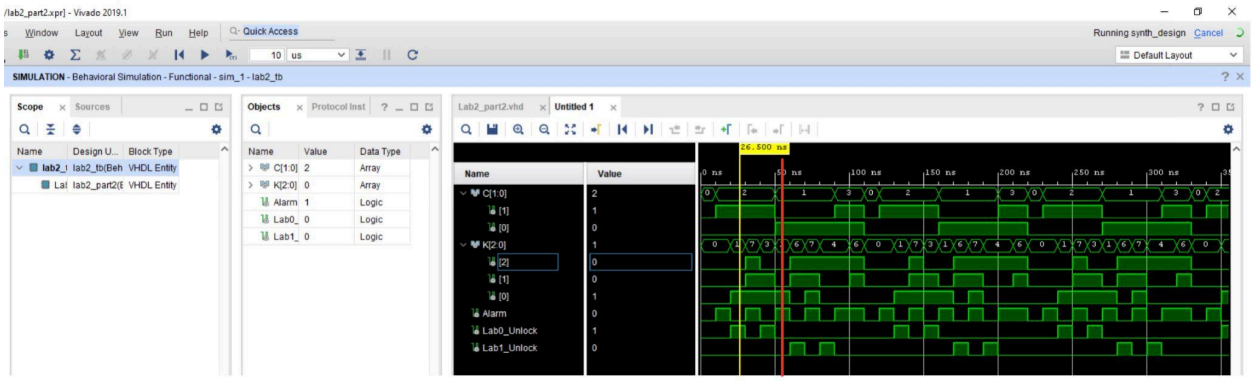
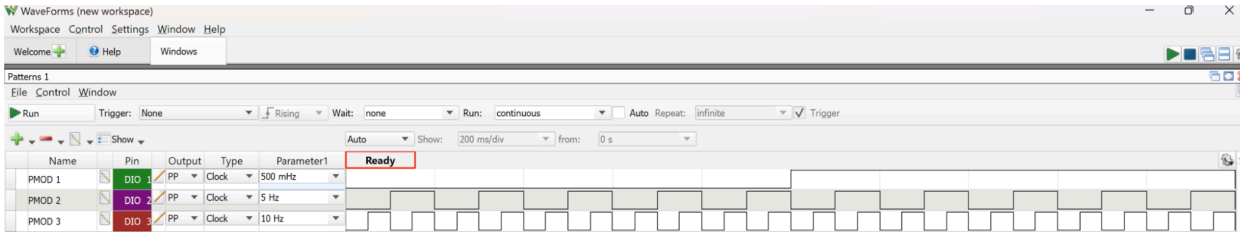


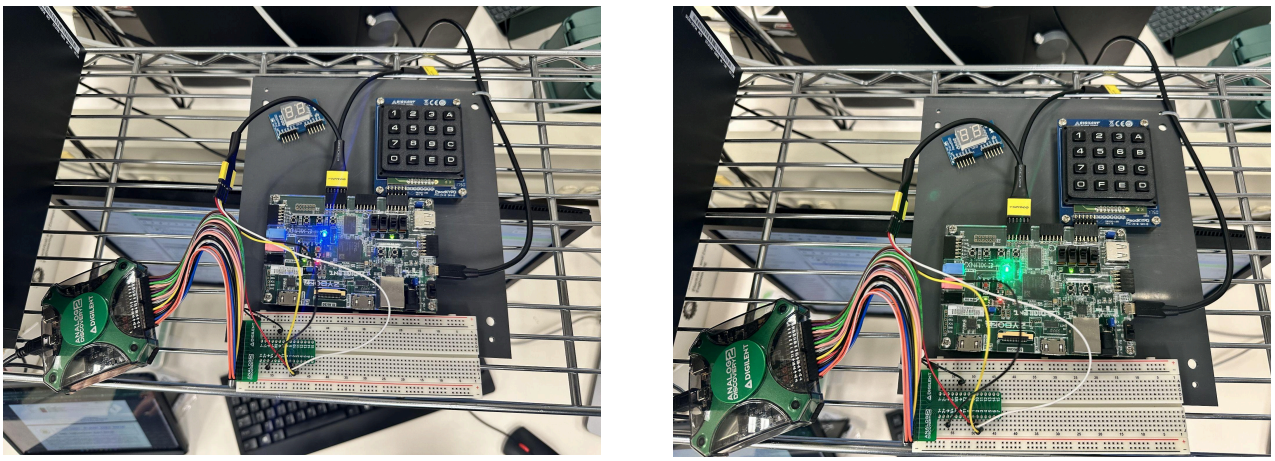
Figure 7: VHDL Simulation Window Part 1



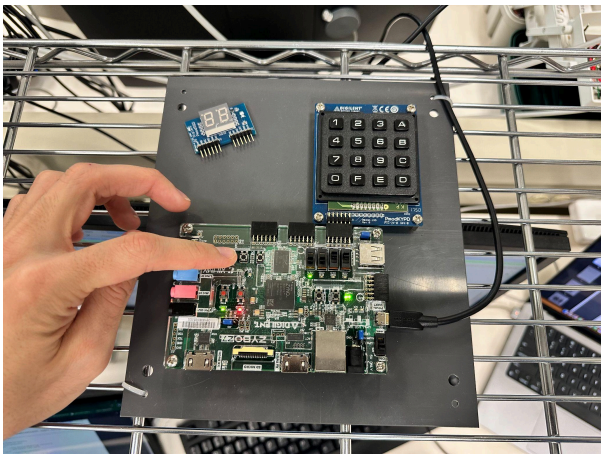
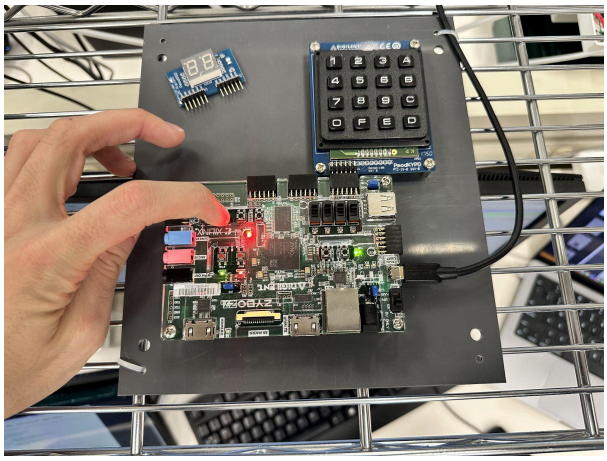
↑Figure 8: VHDL Simulation Window Part 2



↑Figure 9: Waveform setup for Analog Discovery 2



↑Figure 10: Examples of results from part 1 on the FPGA board

↑Figure 11: Examples of results from part 2 on the FPGA board

Discussion: Both the MUX and DEMUX and the Lab Access Control implementations worked as required. The MUX and DEMUX circuits were able to successfully multiplex and demultiplex the logic defined in our coding. The lab access control circuit also met its requirements of unlocking the Lab0 and Lab1 only when the correct card and the keypad code was entered and also the sounding the alarm when the keypad codes were unauthorized. When the MUX-select and DEMUX-select signals are in an unused state, the output can be unpredictable. When a select input does not match any of the defined input states in a MUX, the outputs may either be undefined or sometimes stay the last valid state. This is the same for DEMUX circuits, when the select outputs do not match any of the defined outputs, the outputs are unpredictable. The Xilinx FPGA board is really flexible and efficient which allows us to implement and program complex designs and the debugging is also really advanced and easier to do on the FPGA board. On the other hand, discrete ICs and breadboards are really good for hands-on experience but they are not as good at implementing complex designs and also are difficult to reprogram for changing designs whereas I can easily reprogram the FPGA board with a few clicks. Me and my lab partner both prefer the FPGAs for their speed and rapid reprogrammability and their ability to implement complex designs. For the Lab Access Control, in the results section in Figure 7, it is a waveform showing the simulation for the design we coded in VHDL. There are 2 lines, one yellow and one red, they are annotated to see if they match the truth table. For the yellow line, we can see that card 1 (C1) is valid as well as keypad code (K0), which in the truth table in the appendix shows that Lab0 will be unlocked. Similarly for the red line, card 0 (C0) is valid and keycode 0 (K0) is valid but it is not the corresponding keycode for card 0, as shown in the truth table, so the simulation triggers the alarm like it should which the truth table tells us. The lab access control system is overall effective as it takes both a valid card and a valid keycode combination to unlock Lab0 and Lab1 and trigger the alarm when one of these are invalid. This

system is simple enough to be used on a small scale, but might be a little cumbersome and complex to set up in larger facilities.

Conclusion:

In conclusion we were able to code the MUX and DEMUX circuits that were outlined in the lab manual. Implementing the code was moderately difficult but once the code was correct it was easy enough to simulate it and download it to the hardware. For next time more explanation on how the code needs to be structured would allow for less confusion.

References:

[1] Department of Electrical and Computer Engineering, *General Lab Report Style Guide,* University of Alberta, Fall 2017. [Accessed: November 11, 2024]

[2] Department of Electrical and Computer Engineering, *ECE_210_LAB_1_revised,* University of Alberta, Fall 2014. [Accessed: November 11, 2024]

[3] Department of Electrical and Computer Engineering, *Mux description.* [Accessed: November 11, 2024]

Appendix:

C0	C1	K0	K1	K2	Alarm	Lab0_Unlock	Lab1_Unlock	
0	0	0	0	0	0	0	0	
0	0	0	0	1	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	1	1	0	0	0	
0	0	1	0	0	0	0	0	
0	0	1	0	1	0	0	0	
0	0	1	1	0	0	0	0	
0	0	1	1	1	0	0	0	
0	1	0	0	0	1	0	0	
0	1	0	0	1	1	0	0	
0	1	0	1	0	1	0	0	
0	1	0	1	1	1	0	0	
0	1	1	0	0	0	1	0	Yellow Line
0	1	1	0	1	1	0	0	
0	1	1	1	0	0	1	0	
0	1	1	1	1	1	0	0	
1	0	0	0	0	1	0	0	
1	0	0	0	1	0	0	1	
1	0	0	1	0	1	0	0	
1	0	0	1	1	0	0	1	
1	0	1	0	0	1	0	0	Red line
1	0	1	0	1	1	0	0	
1	0	1	1	0	1	0	0	
1	0	1	1	1	1	0	0	
1	1	0	0	0	1	0	0	
1	1	0	0	1	1	0	0	
1	1	0	1	0	1	0	0	
1	1	0	1	1	1	0	0	
1	1	1	0	0	1	0	0	
1	1	1	0	1	1	0	0	
1	1	1	1	0	1	0	0	
1	1	1	1	1	1	0	0	