Lab Report #1: AND-OR-NOT Circuits

Authors:
Anurag Koushik(ID:1806274, akoushik),
James Wright (ID:1801348, jtwrigh1)

Course Number: D22

Lab Instructors: Md Anik Hossain, Mohammad Hadi Masoumi

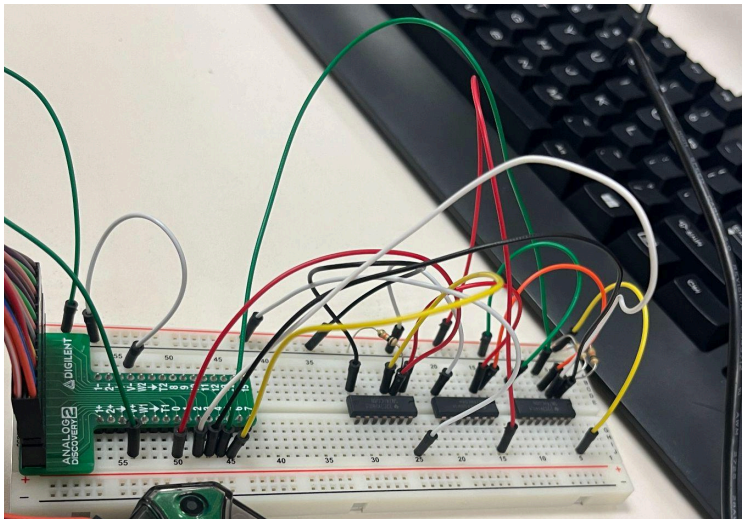
Date of Lab: 2024-10-15

Abstract:

In this lab, we explored the operations, combinations, and implementations of AND, NOT, and OR logic gates. The main goal of this lab was to implement the pre-lab design created from the truth table on page 3 of the lab manual[2]. We did this by simplifying the truth table provided to create Karnaugh Maps(K-Maps), which allowed us to create a simplified circuit diagram using AND and NOT gates. Using this method we were successfully able to make a circuit and implement it using Waveform and Vivado software. In conclusion circuit construction is possible, but we learned that each chip can only be present for one phase of the circuit. This means you can't re-visit a chip from a previous phase since it throws off the timing of the whole circuit.

Introduction:

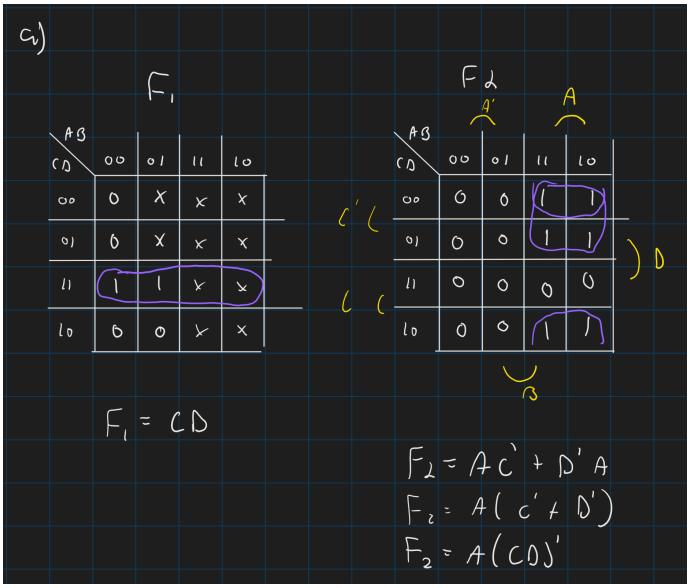
Figure 1: Set Up Circuit



Logic circuits are networks of electrical elements that take input values in binary (ones and zeros) and compute them in a specific way to produce an output value in binary. These elements include gates like AND, OR, and NOT which take binary inputs and make a binary output dependent on the inputs. For this lab we will only be dealing with gates stated above because they are simple and easy to use, just note that there are more types of gates than the 3 above. The AND gate requires both inputs to be 1 to produce an output of 1, otherwise the output is 0. The OR gate outputs a 1 if one of the inputs is a 1 while the other is a zero, otherwise the output is 0. The NOT gate inverts the signal input as its output, therefore inputting a 0 outputs a 1. Multiple logic gates of the same type are packaged together onto an integrated circuit (IC or microchip) which then are able to be used. In **Figure 1** there are three black rectangles on the breadboard which are microchips. Applications of these microchips could inclued creating complex circuit networks to complete computations quickly and efficiently. This would save everyone time and energy which could be spent more efficiently. The objective of this lab was to create the pre lab design corresponding to the truth in the lab 1 manual [2, p.3]. This lab demonstrates how a large table of values can be manipulated into a K-Maps which from there can be turned into a simple circuit diagram. K-Maps are super helpful tools that are used to organize truth tables in such a way that it makes it very easy to pick out the important parts thus creating a simpler circuit diagram. See **Figure 2 & 3** for drawing of the K-map and circuit diagram that was devised for this lab.

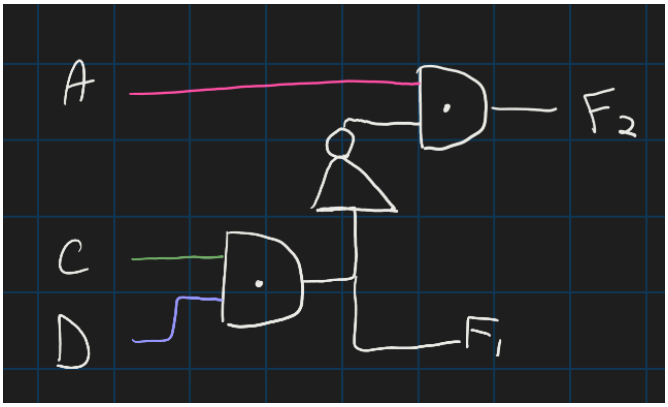
Design Section:

For this lab we were tasked with creating a circuit from the prelab truth table. Our constraints for the lab were to implement our designed circuit using only components found in the ECE equipment kit. The circuit needed to take 4 inputs coming from the analog discovery 2's Digital Input/Output(DIO) ports and output 2 signals. Our design can be seen in **Figures 1 & 2**. We went with a 3 chip design as seen in **Figure 3** due to our pre lab circuit diagram. This design was able to take the 4 inputs given and manipulate them into the desired two outputs. Note that this design could have been simplified system would have run slightly faster and simpler.



↑Figure 2: K-Maps and Final Equation

↓Figure 3: Final circuit diagram



Experimental Procedure and Equipment:

Equipment:

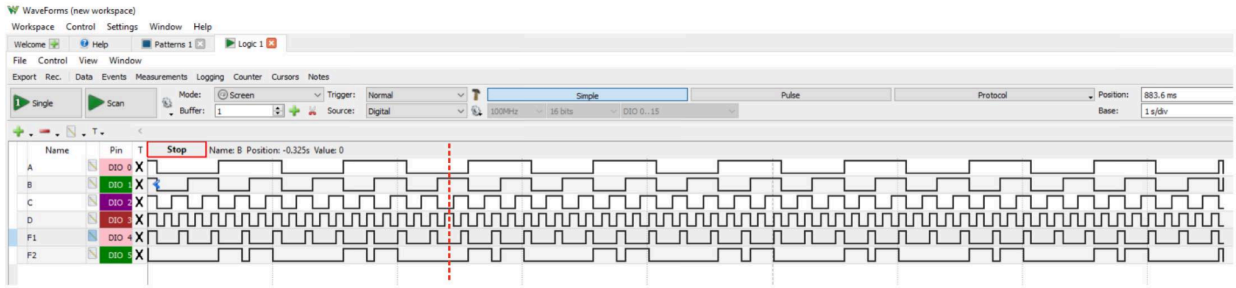
In this lab we used a total of 2 SN74HC08N AND gate chips, 1 SN74HC04N NOT gate chips. This along with some wires, 1 analog discovery 2, and 3 10 kΩ pull down resistors to make sure our signal was clean. See **Figure 1** for more detail and a visual description.

Procedure:

We did the simplification for the logic relationship for outputs F1 and F2 based on the inputs A, B, C and D using K-maps. After that we prepared a schematic diagram as seen in **Figure 3** of the circuit we then moved on to implementing the circuit on the breadboard. This started with setting up the Analog discovery 2 with the correct DIO labeled and defined. Next we placed 2 AND gate microchips and NOT gate microchips on the breadboard. For each microchip we connected the ground to the V- and the Voltage in to V+. From there we used jumper wires to connect from the Analog discovery 2 to the microchips. Looking at the schematics of the two microchips used we can tell were the inputs and outs from the drawings.[3][4] After we were done setting up the circuit, we used the program WaveForms to verify the functionality of the selected cases to make sure we match the expected outputs. We used the patterns window in the waveforms software to configure the inputs and monitor the output in real time. The observation screenshots are in the results section. After we verified the outputs with the expected ones, we moved on to VHDL for implementation and simulation on the Zybo Z7 FPGA board. We created a new project and implemented the logic equations for F1 and F2 from the template we were given from the Lab 1

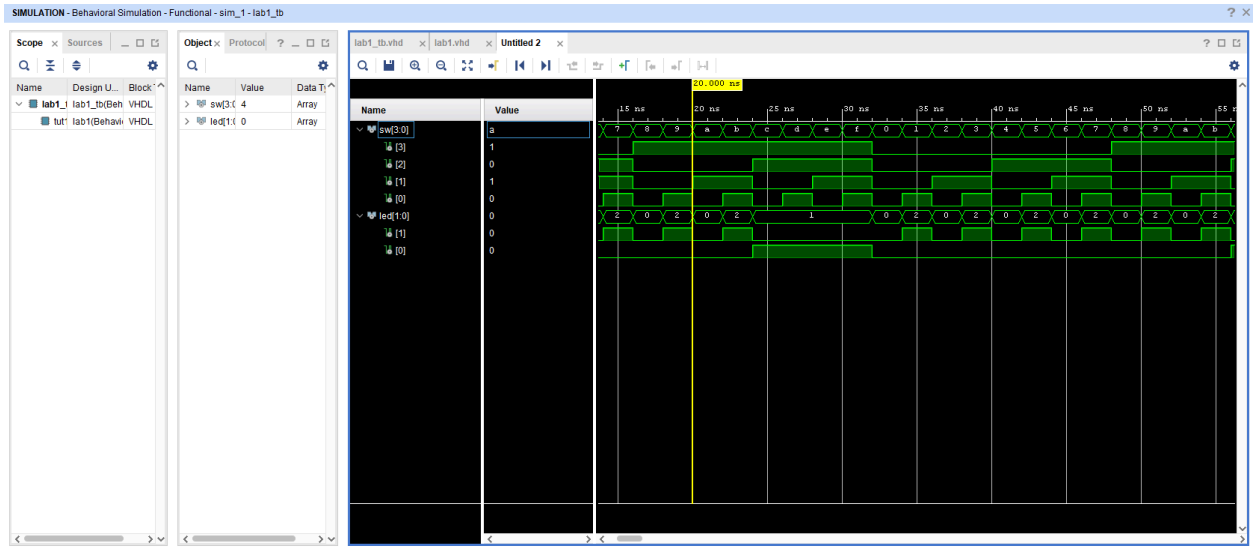
Manual. There, we assigned the switches as inputs A, B, C and D and the corresponding LEDs as outputs for F1 and F2. After that we ran the simulation to ensure our VHDL implementation was matching the expected outputs and that there were no errors in the code. When the code was verified and showed no errors, we compiled the code and downloaded it into the Zybo Z7 FPGA board. After the compiling and downloading was done, we tested the functionality through observing the LEDs by switching on and off different switches.

Results:



↑Figure 4: WaveForms patterns window

Waveforms was used to monitor the outputs based on the inputs A, B, C and D. For example, like on the red dotted line shown, on the line A, B, C and D are 0101 respectively, and the output of F1 and F2 are 00 which match the truth table shown in the appendix 1.



↑Figure 5: VHDL simulation window

After implementing the logic equations in the template code that was given to us, we simulated the code and this is the simulation window of VHDL. We used this simulation to check if our code is operating correctly and matches the expected outcomes. Similarly enough, the simulation matched the expected outputs. We mapped switches (sw) from 0 to 1 chronologically to the inputs A, B, C and D respectively and the outputs F1 and F2 to LEDs (led) 0 and 1 respectively.

So for example at the yellow line, A, B, C and D are 0101 and the outputs 00 matching the one in the Waveforms software.
All the other values were checked from the truth table and both the outputs of Waveforms and the VHDL software matched.

Zybo Z7 FPGA Board: After the simulation test in VHDL passed, we downloaded the code in the FPGA board. In the picture the switches go from 0 to 4 to the right as well as the LEDs, going from 0 to 4 to the right. In the image, the switches 0 and 1 are in the on position which means A and B are 11 and C and D are 00 as they are in the off position. In this case the LED 0 is in the on position (1) and LED 1 is in the off position (0) which matches the truth table as well as all the simulations that we performed.

Discussion:

This lab we built a circuit that takes 4 inputs and outputs 2 signals. Our results show that as the different cases get tested the circuit reacts and produces the expected output. One major observation that we learned was the importance of timing in a digital logic circuit. A microchip can only be used in one phase of the circuit. Since each microchip has multiple logic gates on the IC we thought that you could loopback after going through the not gate seen in **Figure 3**. Below is **Figure 5** which shows the thought process of how we were going to implement the two microchips setup. Since CD goes from the AND gate microchip to the NOT gate chip then back to the same AND gate microchip this causes timing issues with the circuit. This limitation ultimately led us to using a 3 chip design.

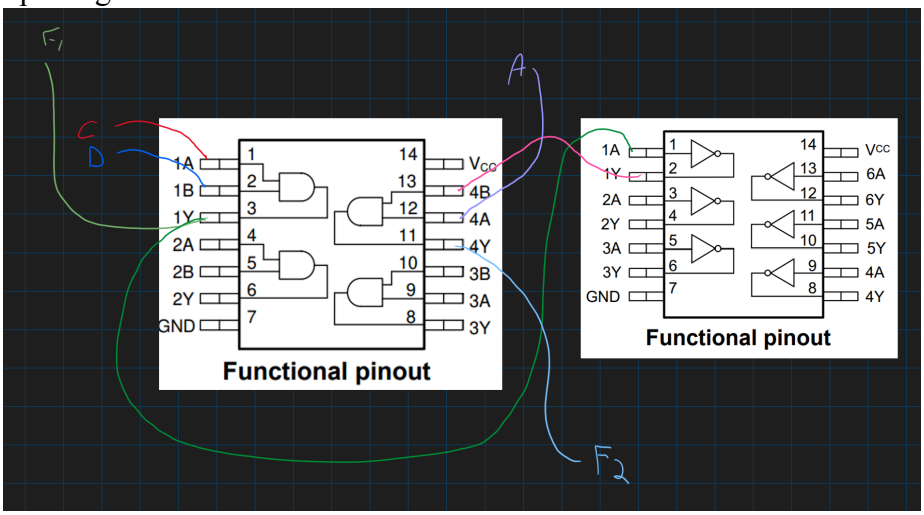


Figure 5: 2 Microchip design that didn't work

→

Conclusion:

In conclusion we were able to successfully build a circuit that produced the correct graphs as seen above. Expectantly the chips worked as intended and were easy to use once set up. Unexpectedly was our issues with the timing of the microchips in the wrong phase of the circuit that was explained in the **Abstract & Results.** We believe that to improve this circuit we could have used a NAND gate instead of an AND to make the circuit slightly faster and more cost effective.

References:

[1] Department of Electrical and Computer Engineering, *General Lab Report Style Guide,* University of Alberta, Fall 2017. [Accessed: October 25, 2024]

[2] Department of Electrical and Computer Engineering, *ECE_210_LAB_1_revised,* University of Alberta, Fall 2014. [Accessed: October 25, 2024]

[3] Texas Instruments, *CD74HCT04 High-Voltage CMOS Logic Hex Inverter*, datasheet, Oct. 2019. Accessed: Oct. 28, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/cd74hct04.pdf

[4] Texas Instruments, *CD74HCT08 High-Speed CMOS Logic Quad 2-Input AND Gate*, datasheet, Oct. 2019. Accessed: Oct. 28, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/cd74hct08.pdf

Appendix:

Truth table:

Inputs				Output 1	Output 2
A	B	C	D	F1	F2
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	x	0
0	1	0	1	x	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	x	1
1	0	0	1	x	1
1	0	1	0	x	1
1	0	1	1	x	0
1	1	0	0	x	1
1	1	0	1	x	1
1	1	1	0	x	1
1	1	1	1	x	0