Digital Logic Design 3441 – Spring 2021

Lab #7: Implementing MUX Circuits and Counters Using SimUAid

Cody Betzner, Thomas Nguyen

Overview: Goals of the Lab

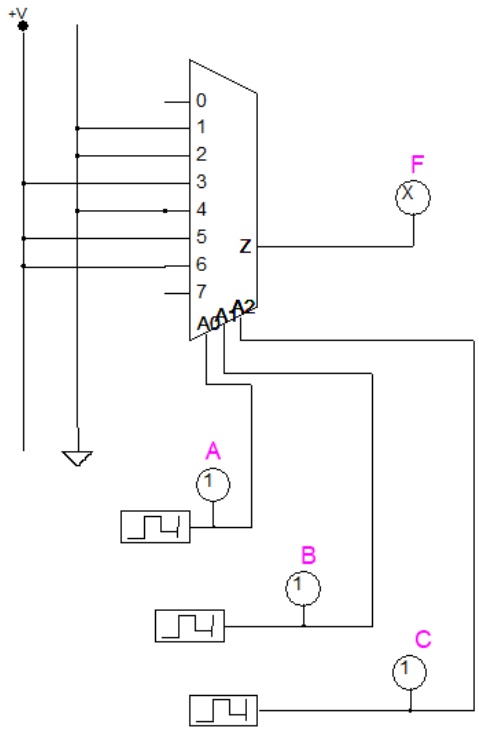
The first goal of this lab is to simulate the behavior of a standard MUX given a specific equation in POS, which was then converted to SOP. This was done to demonstrate how multiple inputs can be inputted into a MUX and how an output can be formed from this. The second goal of the lab is to simulate a sequential counter using D flip flops and then implementing it using actual hardware on a breadboard. We were tasked with building a circuit around a given sequence and used K Maps and truth tables to derive the sequence of electronics needed.

Part 1: Building Function in SOP form using 8-to-1 MUX

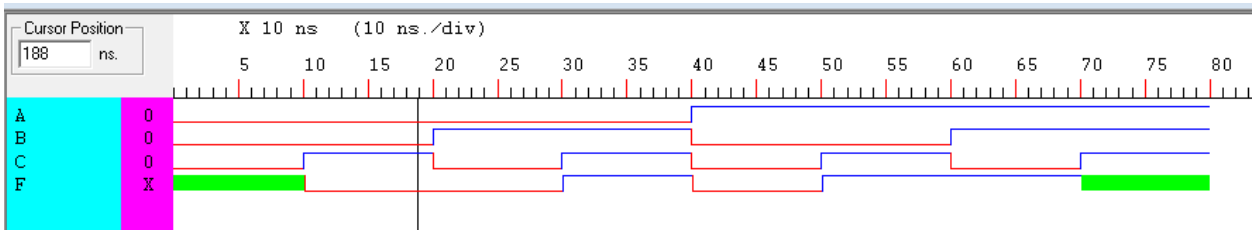
F(A,B,C) = AC + BC + ABA

A

BC	0	1
00	X	1
01	0	1
11	1	X
10	0	1



*don't cares disconnected (not connected to 0 or 1)



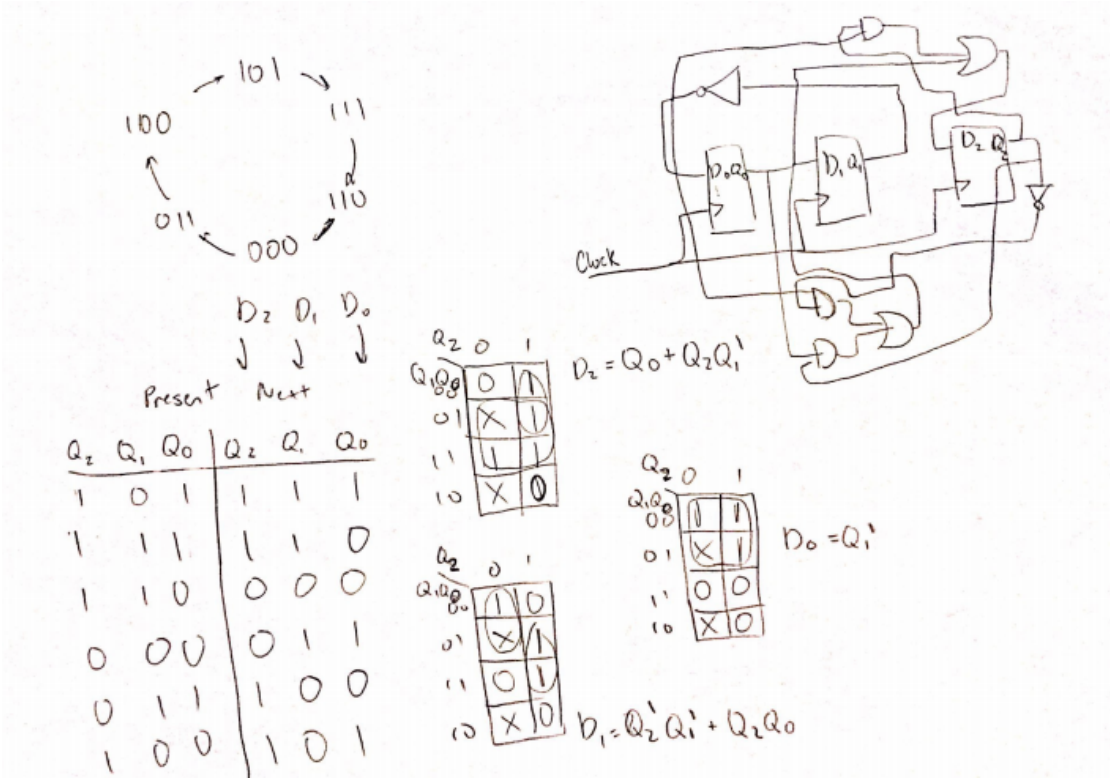
1. A multiplexer is a device that takes in a group of data inputs and a group of control inputs and outputs it as a single data line. In Digital Logic Design, an MUX can take in digital signals and create an output based on the design of that specific MUX.
2. In order to implement a function that has 5 variables you can either connect a number of MUXs together, or you can use an 32:1 MUX that has 5 control inputs. Possible combinations including:
    - One 8:1 MUX (3 total control inputs) + One 4:1 MUX (2 total control inputs)

- Two 4:1 MUX (4 total control inputs) + One 2:1 MUX (1 total control input)
- One 8:1 MUX (3 total control inputs) + Two 2:1 MUX (2 total control inputs)

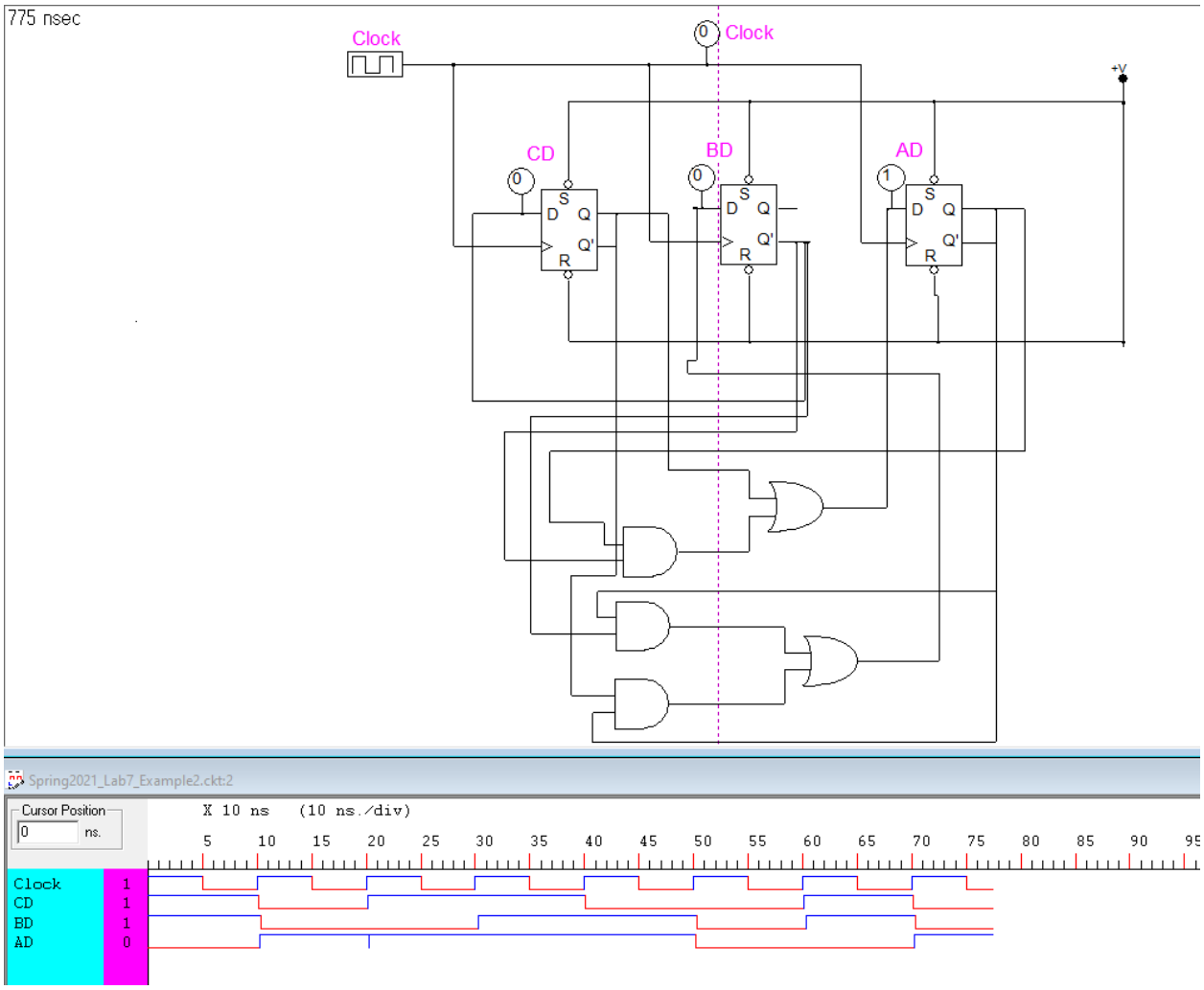
A MUX higher than 8:1 isn't needed but a 32:1 MUX can be used.

Part 2: Building a Sequential Counter in SimUAid

Work:



SimuAid Circuit and Timing Diagram (AB is MSB and CD is LSB):



3. The combinations not used in the counter are ‘don’t-cares’ and are not used in the counter because their values do not change the final product. The ‘don’t cares’ are unstable and eventually change to the final values shown in the sequential counter. In order to prevent them from showing up in the output, one can use pull down resistors.
4. The D flip flop has S and R input so they can be forced to the set or reset state. The output is high when S is pulled high and the clock signal is 1. The output is low when R is pulled high and the clock signal is 1. We must keep them high so the circuit is not over written by the set or reset state.

Part 3: Counter Circuit Hardware Implementation

Cody's link:

https://drive.google.com/file/d/1GXkVnOoDu8CQhhJRBIdpKvFNKOOwDJ8R/view?usp=sharing

Thomas's link:

https://drive.google.com/file/d/1uiN7LCbwSn9ZVne6KSYZi3bpO6Gp4DmP/view?usp=sharing

Conclusion

In this lab, we learned about the behavior of a MUX and how it can take in multiple inputs and output a given function. Additionally, we learned how to implement a sequential counter using D flip flops, AND gates, and OR gates. Given a sequence, we learned how to correctly derive equations for each aspect of the circuit. We also learned how to affirm our sequence was working by studying the timing diagrams.