Lab 2: 7-Segment Display

Eric Liu

Objectives

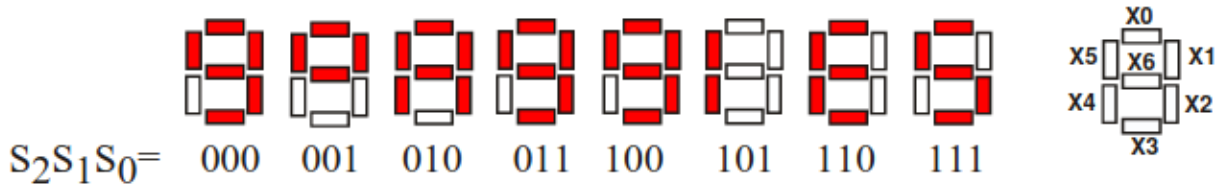


Fig. 1: 7-segment display with corresponding inputs.

In this lab, our goal is to find, minimize, and create circuits for the 7-segment display outputs shown in Figure 1. We will first be testing our circuits by creating the circuit in Quartus and simulating it in ModelSim. Then, we will be hooking up our circuit to switches on the FPGA and flashing the circuit so that we can verify the display looks correct on the actual board. In part two, we will replace the switches with a counter that's hooked up to one of the buttons on the FPGA so that we can cycle through the display outputs by clicking the button. We will verify our schematic again with this new circuit.

Design and Test Procedure

Prelab:

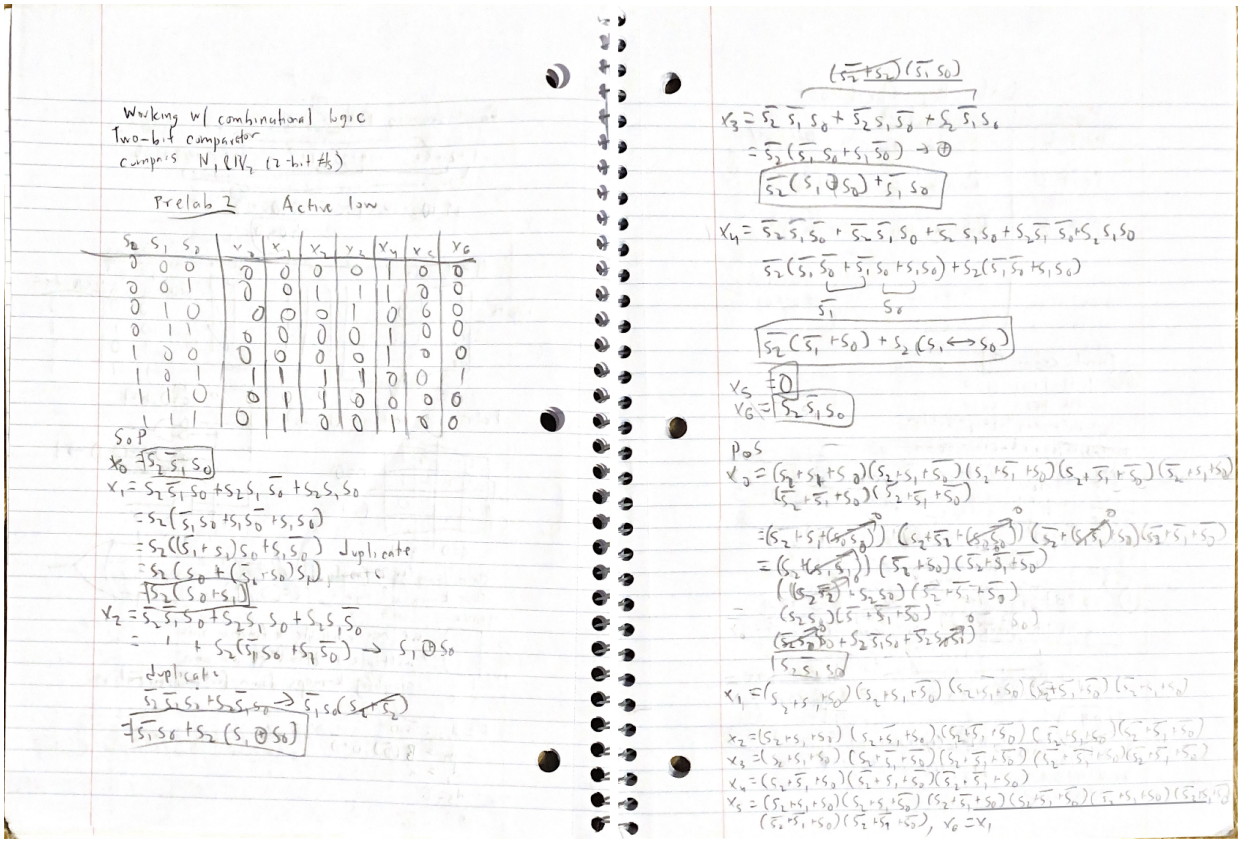


Fig. 2: Prelab simplifications

While I did try to follow the instructions for doing the prelab, I did not simplify the Product of Sums logic functions correctly and I had to redo it during the lab. I ended up using K-maps to do this:

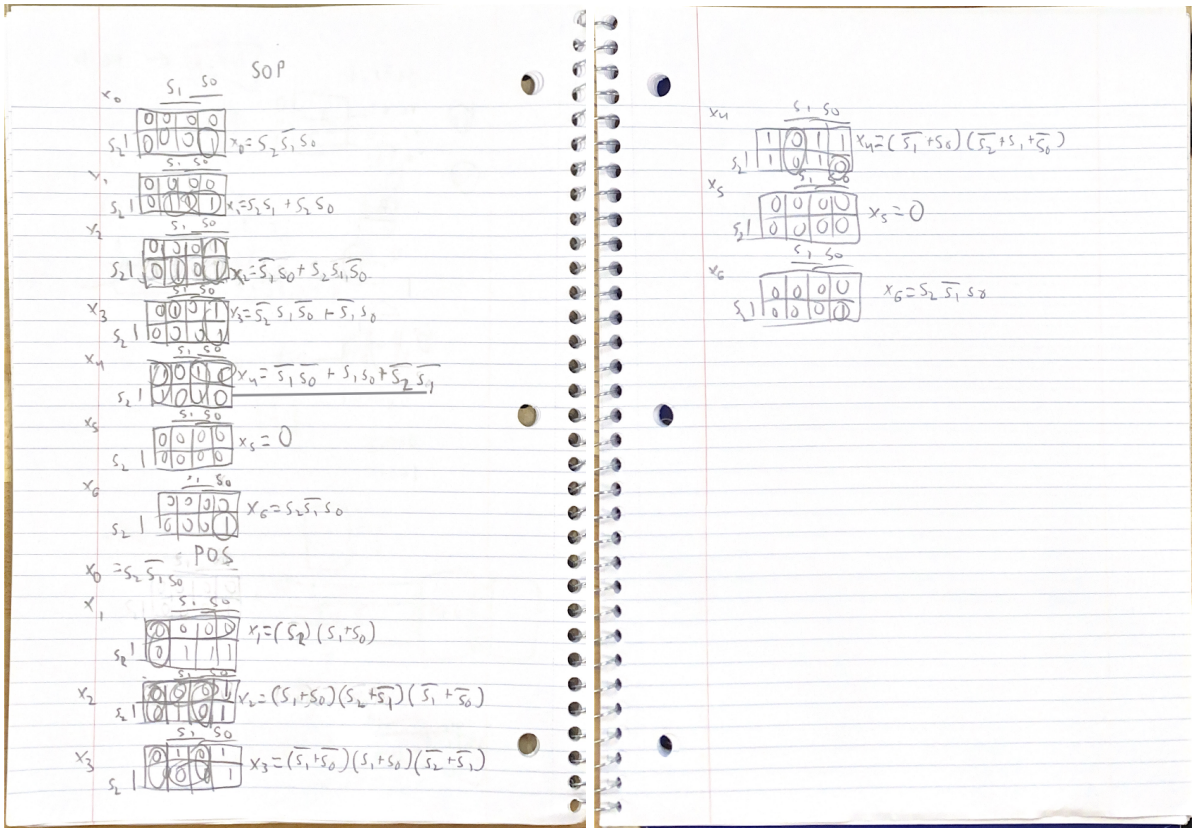


Fig. 3: K-map simplifications of each function (SOP and POS forms)

Part 1:

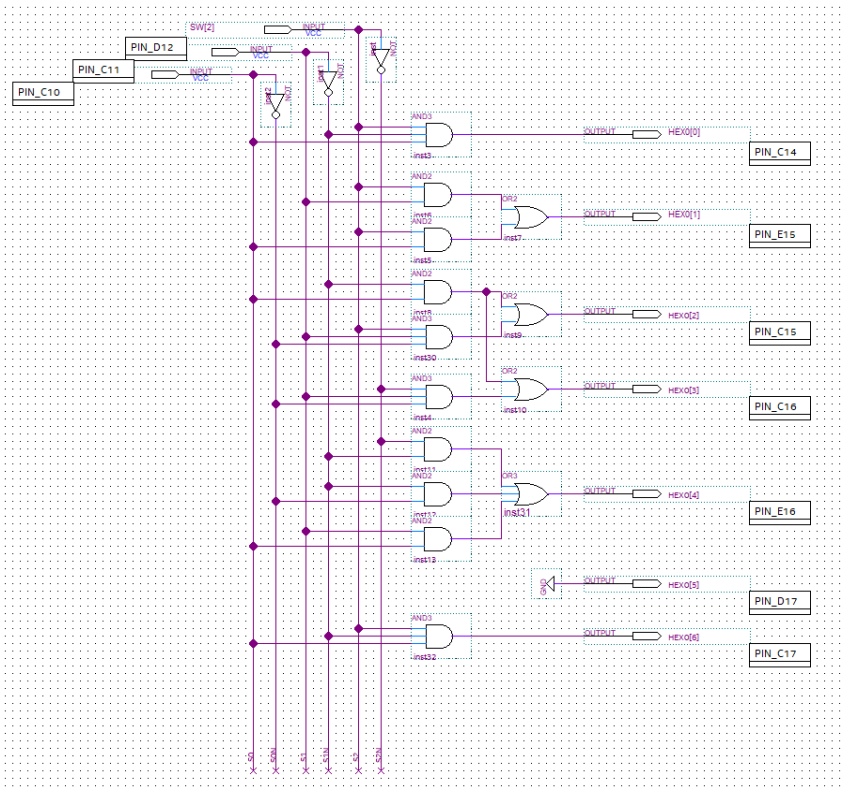


Fig. 4: Sum of Products variant of the part 1 circuit.

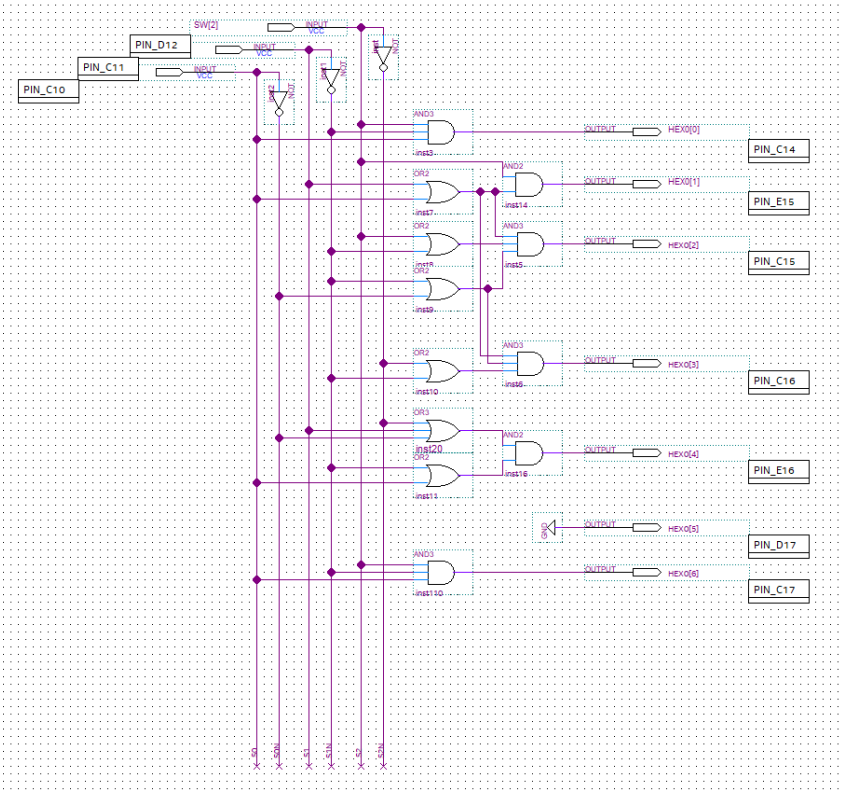


Fig. 5: Product of Sums variant of the part 1 circuit.

The simplified two-level logic used in both circuits were derived from K-maps, which were shown above. The ModelSim testing results are in the Results section below.

Part 2:

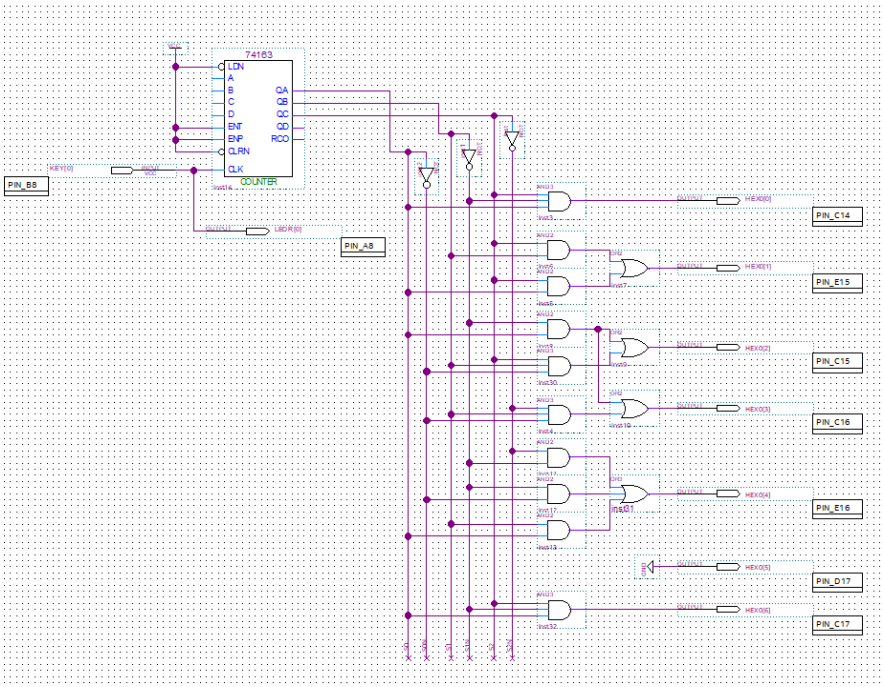


Fig. 6: Sum of Products variant of the part 2 circuit.

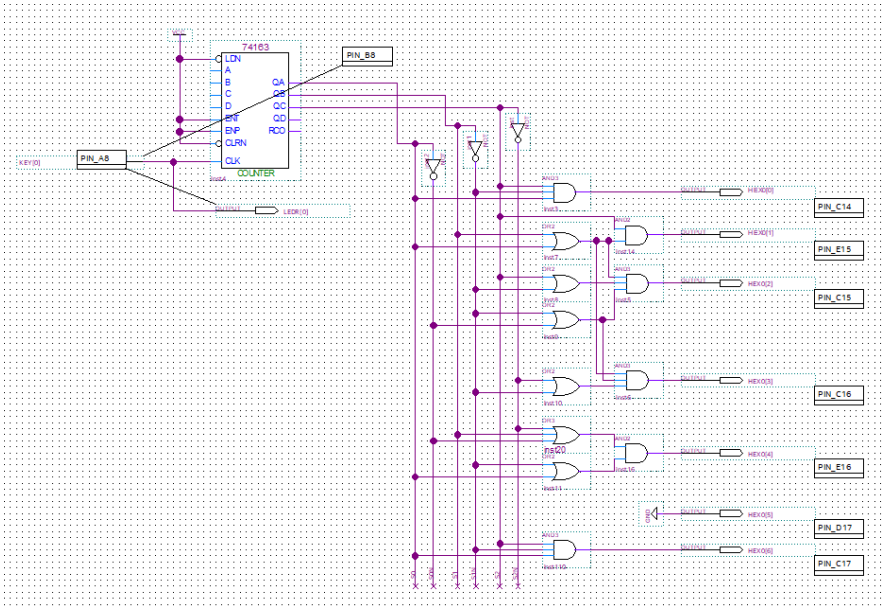


Fig. 7: Product of Sums variant of the part 2 circuit.

This part of the lab was pretty simple because it was basically just the work I did from part 1 but connected to a BCD counter and a button instead of three switch inputs. This way you don’t need to worry about how to configure the switches to increment the circuit and you only need to press the button to cycle through all of the possible outputs,

making it a lot easier to verify if the circuit worked correctly. Here's a picture of one of the outputs of the circuit:

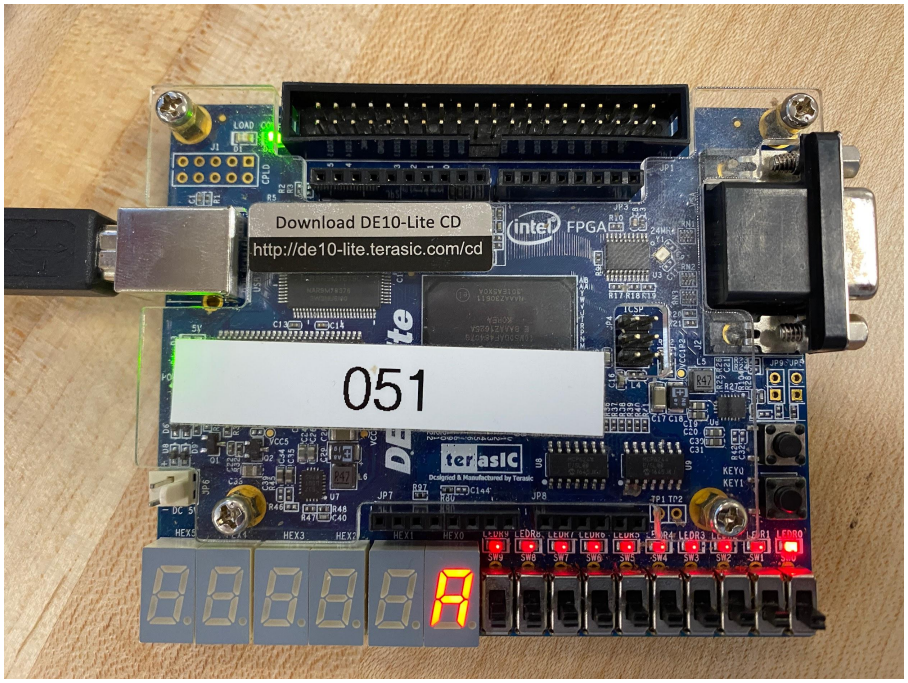


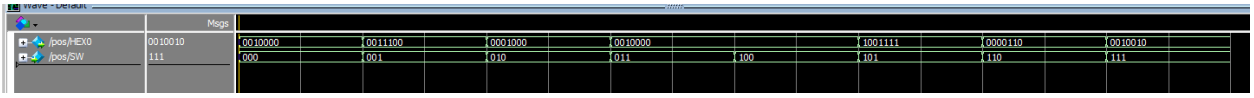
Fig. 8: Output of the SOP part 2 circuit with input 010.

As you can see, this matches the output specified in the diagram in the objectives section. The rest of the outputs matched as well, although I did not take a photo for each part. The ModelSim results from part 1 still hold, though.

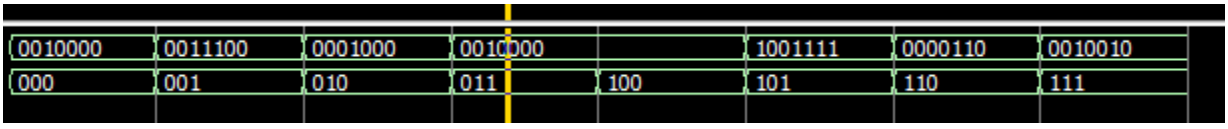
Results and Answers to Questions

Part 1:

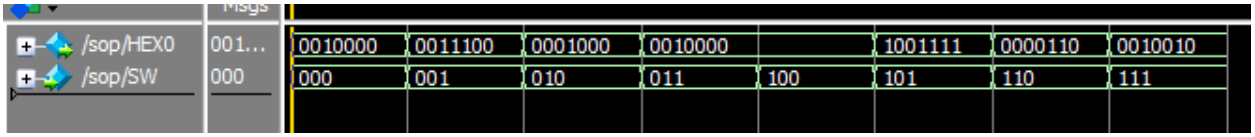
POS form:



Zoomed in:



SOP form:



The POS and SOP forms match, and they all make sense when referencing the diagram expected in the lab manual (Fig. 1).

Conclusions

In this lab, I learned more about designing circuits: first, planning out the circuit by drawing out a truth table and finding a relatively minimized form of the function that represents it, drawing out the circuit in Quartus in a neat way, testing the logic in a simulator, and then finally compiling the circuit onto my FPGA so I can test it on real hardware. I got to practice using K-maps, using Quartus circuit schematics, and using ModelSim to debug circuits.

A couple things went wrong in this lab. When I was transcribing my minimized circuits into a circuit diagram in Quartus, I actually messed up one of the 7-segment display outputs. I realized this after I performed the testing process in ModelSim, and then I simply went back to the schematic and saw that I had connected my gate to the wrong input wire. Afterwards, I tested the circuit in ModelSim again and everything worked perfectly. I also had a couple of issues with the schematic giving circuit components the same name twice, so Quartus had a little bit of trouble compiling the circuit. I simply renamed some of the circuit elements so that none of them had duplicate names, and then Quartus compiled the circuit smoothly.

I think I could've improved upon my design procedure by initially using a K-map. When doing the prelab, I simplified the circuit by using boolean algebra theorems, which isn't super consistent because it's easy to make a mistake and simplify the function wrong, or not reduce the function all the way down to its most simplified form. Using a K-map is much more consistent because the K-map does a lot of initial simplification for you, so I should've relied on that instead of realizing after the lab had started.

I think I also could've improved upon my testing procedure: the way that I had to change the test input in ModelSim was really slow and a bit clunky, and I imagine there's a way to autoincrement it with the scripting language and terminal that ModelSim provides. I wasn't sure how to do this though, so I simply manually changed the value driven by the pins each time to check. Next time, I hope to look into more automated testing so I don't need to do so much work to check my own work (and maybe even mess up checking my own work!)

The results of this lab were pretty self explanatory; I got the 7-segment display outputting the correct "digits" that were shown in the prelab and I didn't run into too many issues. However, I think this lab was pretty important because it helped us

practice our skills in designing basic combinational logic functions and reducing them to be more efficient, and overall it was a good learning experience.