Digital Designs

CSE 371 Lab 1 Report

Wenjie Ma

# Procedure

* Counter

For designing the counter, I had to get a clearer view of what the lab is trying to accomplish. From reading the specs, I get a clear understanding that there is only two states that exist in this parking sensor, which is EMPTY and FULL. It is empty at a counter of 0 and full at a counter of 25. This means that anything in between is a free for all. So I designed the counter in a way that will stop at 0 and cannot go negative while displaying empty. It will display full when the counter is at 25 and cannot go any higher. Anything in between, I allow it to increment and also decrement because it is within the range desired.

* Display

For the display, I realized that there are 25 states that the machine may want to display. I realized that it may be a hassle to write out 25 states on its own so I decided to create parameters that represent each integer state. Then separating the hex display numbers into counter 1 and counter 2. It will display hex displays onto each counter display.

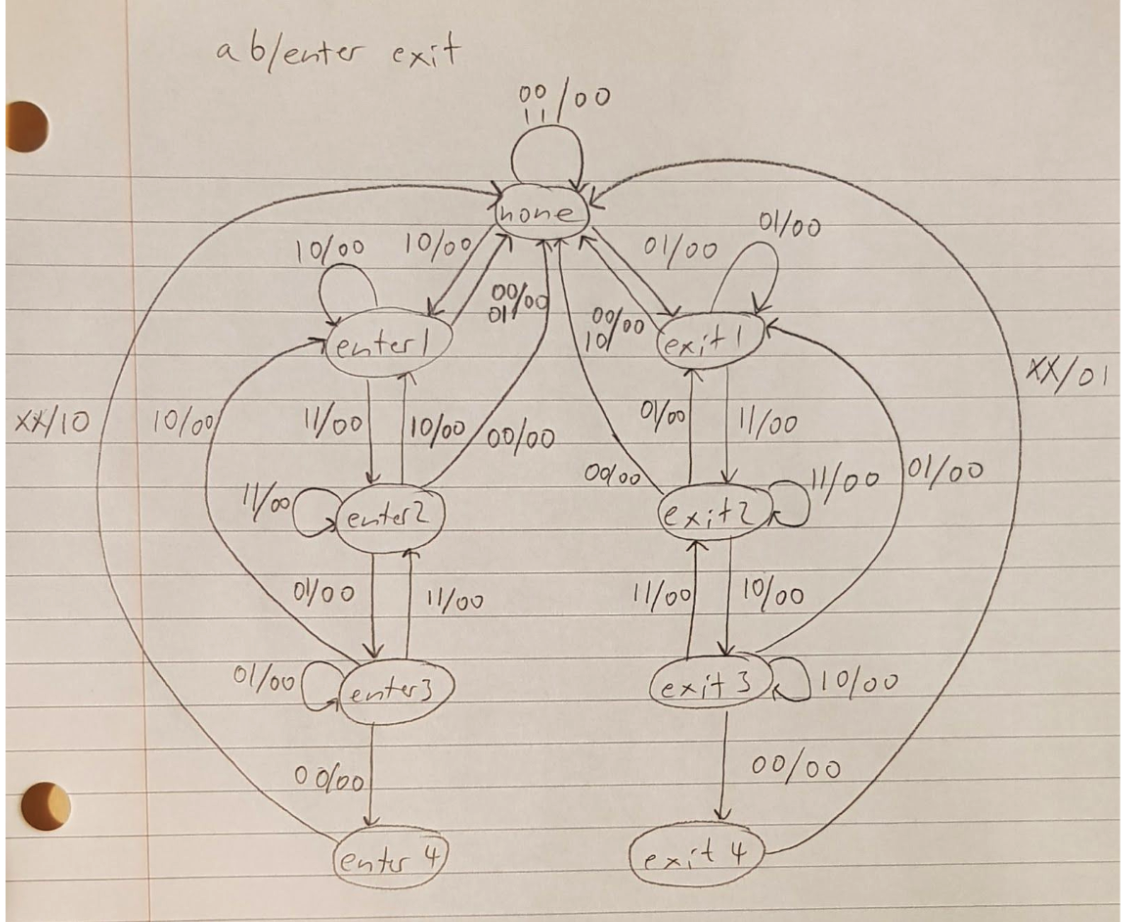
* Sensor

For the sensor, I realized that there are four states for the parking sensor in general. Which are empty, A, B, and A&B. This allowed me to create enum states for each and traverse through each when encountered with the particular button pressed.

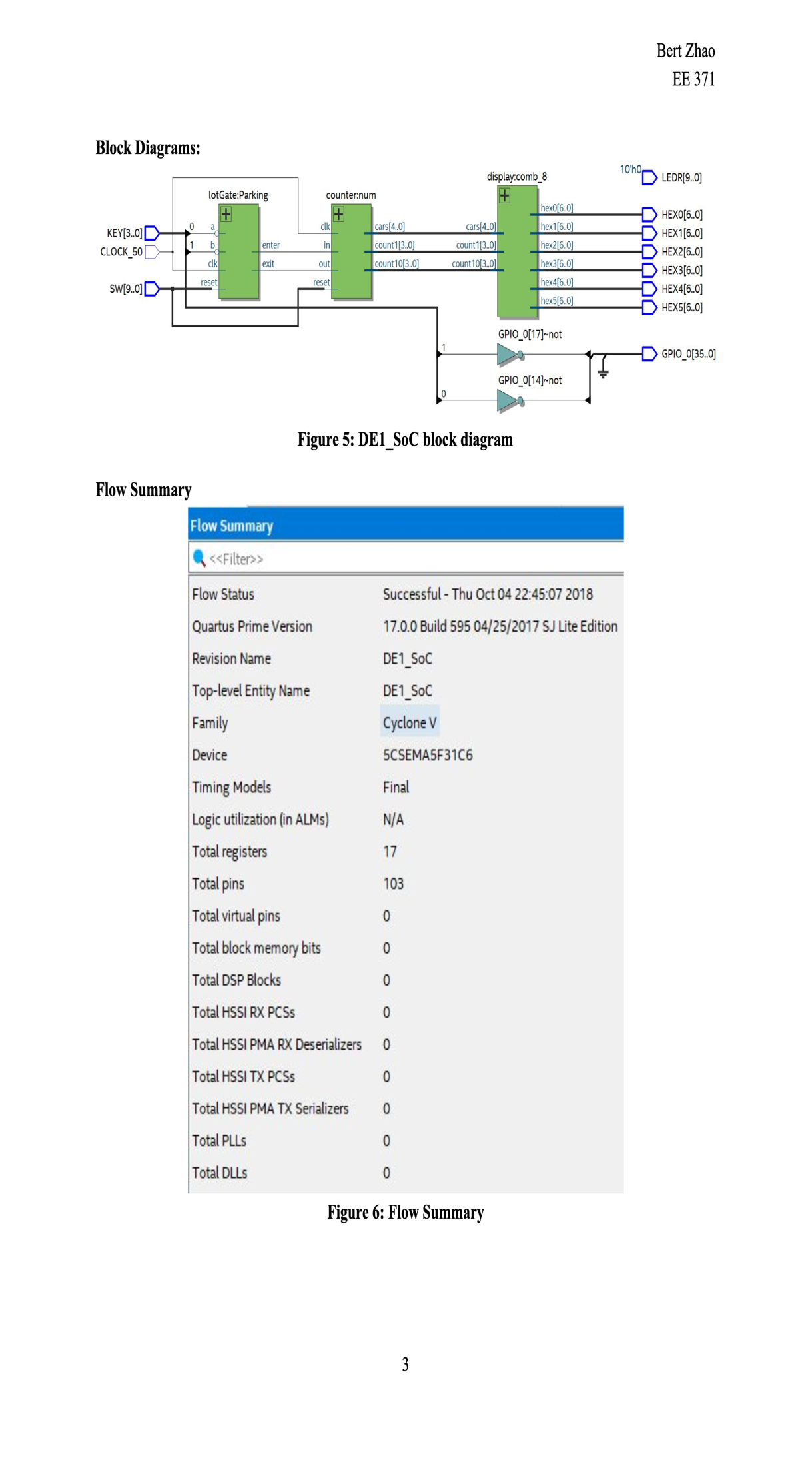
* DE1\_SoC

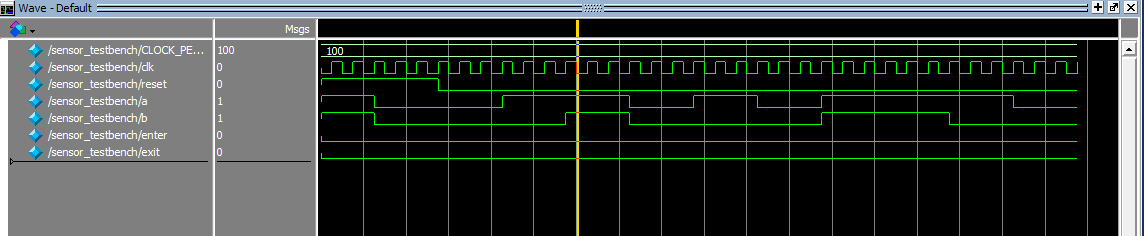
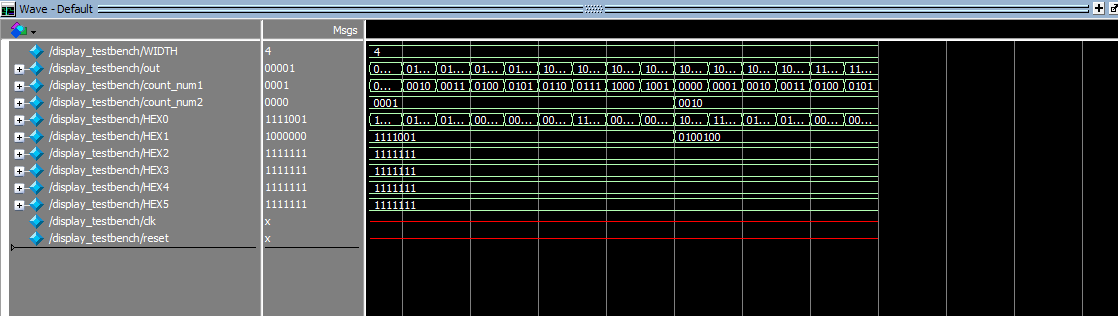
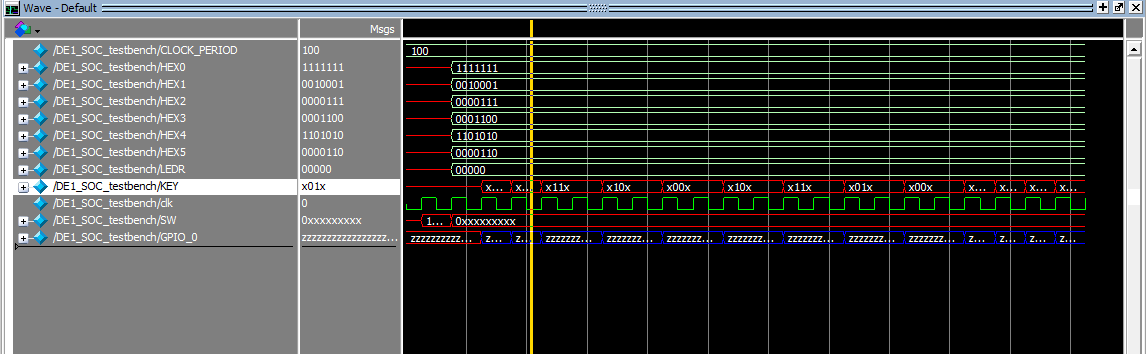
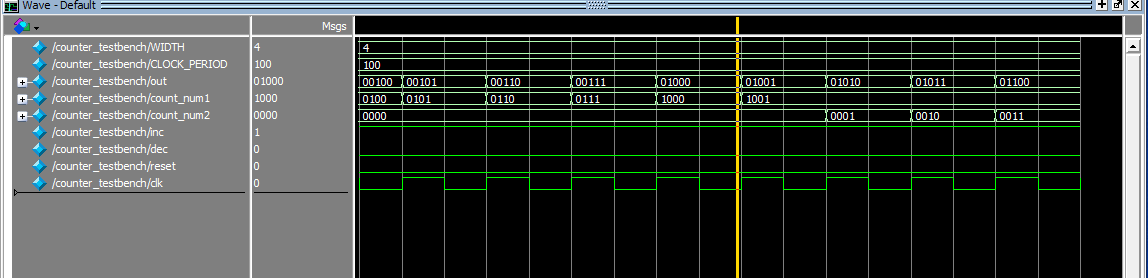
Designing the DE1\_SoC was quite hard for me, because I haven’t touched FPGA and its interaction with GPIO in a while. So I had to dig up old 271 stuff to get a quick rewrap of what I did previously. So I connected all the ports and buttons to the right modules and implemented them.

**Diagrams:**



**Figure 1. State Diagram**

****

**Figure 2. Block diagram**

**Figure 3, 4, 5, 6. ModelSim Simulation.**

I tested for the ability to change states in display.sv and counter.sv. By making sure that the Model Sim simulations would be able to change states with the change of input values; I knew that the program would be able to work on the FPGA.

The finished project would be able to increment and decrement within the range of 1-24. With the exception of 0 and 25 being able to show empty and full, respectively. This allows the user to know that it has reached the bounded ranges of both side.

## Problems faced & Feedback

For the lab, I spent about 15 hours on the lab. I feel like one of the tips that I learned was to learn how to build test benches with the generate statement. This allowed me to save a lot of time by skipping the process of manually typing every test case that I want to cover. One of the issues I had with the lab was figuring out how many states I want to include in the counter because I originally had over 5 and realized that was not the right way of doing it. This led me to think more about the state diagram and come up with a simpler and more direct solution that only had 4 states.