Lab 1: Counters and Simulations

Eric Liu

Objectives

In this lab, our goal is to learn how to use our DE10 FPGA development boards. We will do this by creating circuits in Quartus and flashing the output onto the FPGAs so we can observe their behavior. We will also be using ModelSim to simulate the output of our circuits, which will be a useful skill in later labs: it will be much easier to debug the output of our circuits with a simulation than to try and figure out what's going on using only the FPGA.

Design and Test Procedure

The design of the circuits in this lab were simply provided to us in the lab manual.

Part 1:

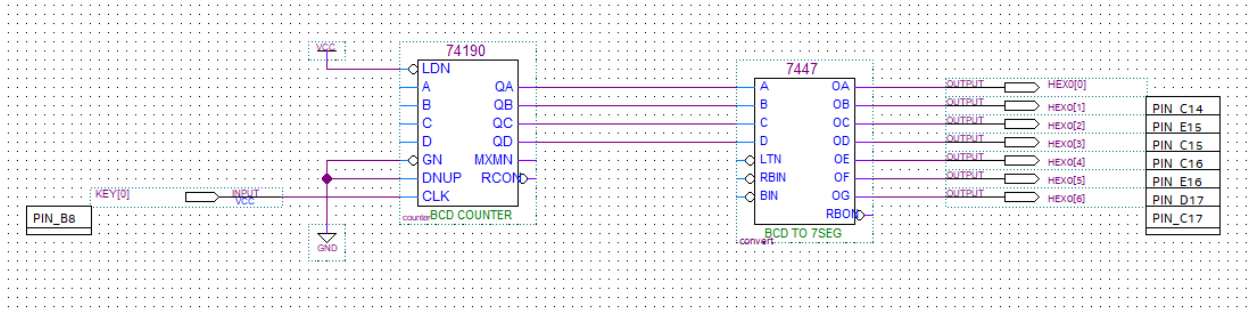

Fig. 1: Initial circuit schematic, in Quartus.

Here, we have two integrated circuits: a binary-coded decimal counter and a converter that takes the number in the counter and converts it to outputs that can then be shown on a 7-segment display. The outputs of the counter (the 4 bits of the counter's number) are wired to the inputs of the converter, and the outputs of the converter are wired to `HEX0[0]` to `HEX0[6]`, which correspond to the pins of the 7-segment display. The counter is also connected to ground, Vcc, and `KEY[0]`, which is a push button on the dev board. This means that each time the button is pressed, the counter will increase by one (unless it's at 9, in which case it'll overflow back to 0).

To test this circuit, we can simply compile it and run it on the FPGA, and make sure that the number shows up on the display and that it increases when the button is pushed down.

Part 2:

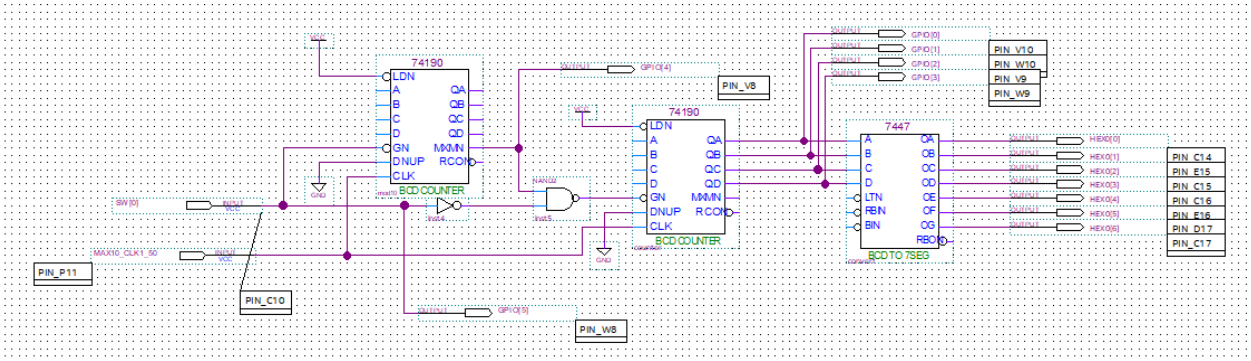

Fig. 2: Expanded circuit with two counters.

This is a modified version of the circuit from part one. Now we have two counters, one of which is only there to throttle the clock. The second counter is disabled by default, but every time the first counter overflows, the second counter is enabled by the NAND gate. Effectively, this makes the frequency of the second counter 5 MHz instead of 50 MHz, since it only fires every 10 clock cycles. The ground inputs of both counters are connected to a switch, which “turns off” both circuits when driven high. So we should expect the 7 segment display to freeze on the current number of the second counter if we turn on the switch. There are also some new outputs connected, `GPIO[0]` to `GPIO[5]`, which are to help us when we visualize the circuit later.

To test the circuit, we first compiled and ran the program on the FPGA. However, it’s almost impossible to actually tell if the circuit is working properly, because the display is toggling way too fast. We can only see if the counter freezes to a “random” number between 0 - 9 when the switch is turned on, but this isn’t a very good method of verification because most of the operation of the circuit isn’t proven by this.

Instead, we will be simulating this circuit in ModelSim so that we can see a graph of the signals in respect to time. This will make it much more obvious whether or not each component in our circuit is working properly, because we can track the outputs with perfect accuracy:

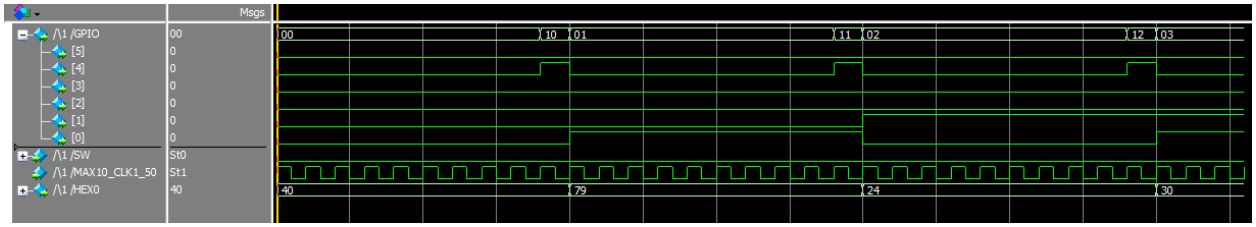

Fig. 3: 22 clock cycles of waveforms in ModelSim.

Now we can see that our counter is indeed incrementing as expected. Although, it’s a bit hard to see because the second counter is actually only the bottom 4 bits of GPIO (0 - 3), so the hex number that is displayed isn’t accurate (it also accounts for the overflow bit of the first counter).

Results and Answers to Questions

Part 1:

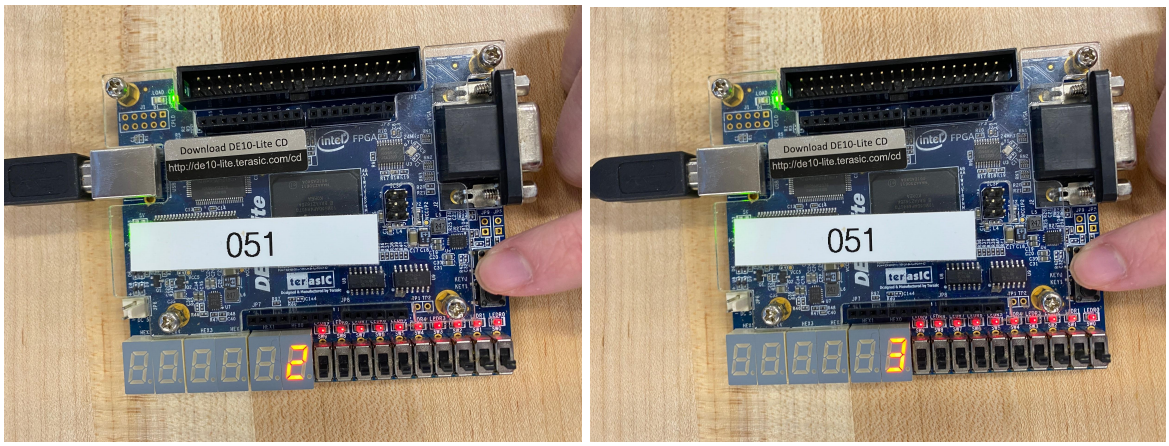

Fig. 4: Counter before and after pushing down the push button.

Part 2:

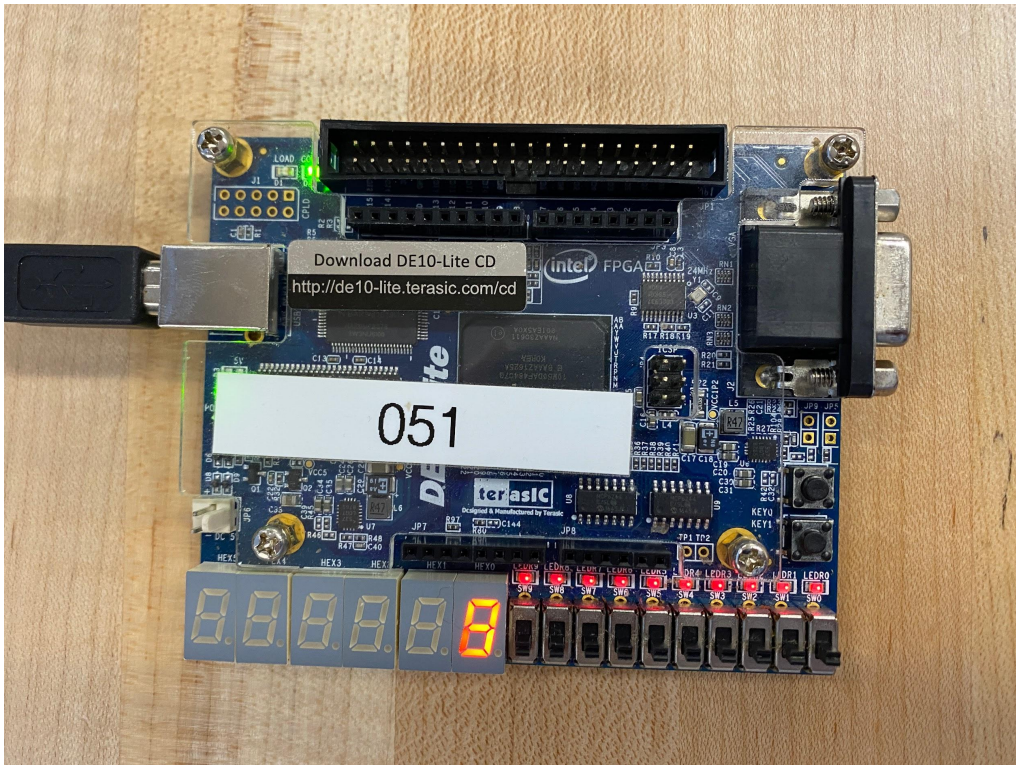

Fig. 5: The FPGA counting up rapidly.

While it was hard to see in real life, it's actually somewhat visible that the FPGA is counting up in Figure 5 because the exposure of the camera captures the "density" of the light. You can see that the left bottom segment is less bright than the other ones, which probably suggests that it captured the transition between 8 and 9.

Conclusions

I learned a lot from this lab. This was my first time working with an FPGA and using Quartus to create schematics to run on the FPGA, and I got to understand a lot about how each component worked in the circuit by copying the schematic and running the circuit to see what the components behaved like. I also got to simulate the circuit in ModelSim and twiddle with it to see how the signals looked at different times, which also helped my understanding.

A couple things went wrong in this lab. First, SystemBuilder wasn't installed on the lab computer when I was working on the lab, and I wasn't sure where to find it. I ended up googling the name and found a download link for it, but I should've been looking under the Canvas files since it's documented under the software installation guide. Also, Quartus had some trouble running ModelSim because it requires you to specify the path to the executable, despite being installed in the same package. To fix this, I went under Tools > Options > EDA Tool Options and typed in the path to ModelSim manually, after which the lab guide worked fine. I also had a bit of trouble installing the driver for the DE10 on my personal laptop, as the driver that came with Quartus refused to install. For this, I also googled it and found an article from Terasic that solved my issues.

I don't think I could've improved upon my design and test procedures, because I simply copied the design and test stuff from the lab manual.

Overall, this lab was very helpful in learning how to use the tools that will be used in future labs.