Shikha Mody

Jivan Gubbi

Hayley Martinez

Brendon Ng

Armaan Singh

Bradley Mont

COM SCI M152A Lab 5

TA: Logan Kuo

Lab 0 Report

Note: Because one of the workstations was not functional, we worked in a large group of 6 for Lab 0. This was approved by our TA, Logan Kuo. Normal groups: (Shikha Mody, Brendon Ng, Armaan Singh) and (Jivan Gubbi, Hayley Martinez, Bradley Mont).

**Introduction**

This lab was meant to introduce us to the tools and software we will be using in the class. There were three main parts to this lab: the first was simulating sample Verilog code to verify the software and hardware were working properly, the second was to implement a four-bit counter in a lower level and more modern version, and the third was to implement a clock divider to adjust the frequency of a clock to flash an LED on an FPGA board.

**Design description**

For the design of lab 0, we first ran the sample project provided to us to familiarize ourselves with the Verilog environment, then we made our own 4-bit counter, modified it to a modern version, and finally created a clock divider which could maintain a one hertz frequency.

To run the sample project, we downloaded the source files for the combinational circuit from CCLE and set up the environment. Then we implemented the code onto our FPGA board to ensure our environment was working correctly.

Next we implemented the naive 4-bit counter. We started with the base schematics of a 4-bit counter circuit using D flip-flops provided to us. Each bit in the counter was stored as a register within the module, a0 being the least significant, and a3 being the most significant bit. In addition, our module used a reset signal, named rst, to start the counter from 0 again. The counter incremented by one at each clock cycle. To do this, the a0 bit was flipped, and each subsequent bit was flipped depending on whether the previous bit overflowed. For example, if a0 changed from 1 to 0 (i.e. overflowed), then a1 would be flipped, and so on till a3.

The modern version of the 4-bit counter, instead of relying on 4 single bit registers, used a single 4-bit register a[3:0]. This allowed us to increment the entire register by one much more simply as seen below:

a <= a + 1’b1

This allows us to abstract away some of the underlying boolean logic to produce code which is easier to read and less prone to bugs.

The clock divider took the faster clock from the UCF file and slowed it down to 1Hz. Our new clock's frequency needed to be 100000000 times slower than the original clock. This means every 100000000 positive edges of the original clock should correspond to a positive edge of our divided clock. To do this, we keep a counter that is incremented at every positive edge of the original clock. We negate our divided clock every 50000000 positive edges of the original clock since we need to negate for every positive edge *and* negative edge of the divided clock (negate at twice the frequency/half the period of the desired posedge frequency)

**Simulation documentation**

For our first simulation, we simply ran the sample combinational circuit given to us in the lab specifications. Since we made no modifications to the code (module and test bench) provided on CCLE, we did not include the code in our submission for the lab. Through simulating the waveforms for this circuit, we gained valuable experience with Xilinx ISE software and Verilog in general. We now feel much more comfortable with the process of digital design for future labs.

After becoming comfortable with combinational circuits in our first simulation, our second simulation involved sequential circuits. Specifically, we simulated a simple 4-bit counter using two different implementations. One implementation used gate / flip-flop level logic, and the other implementation took advantage of a higher level abstraction provided by the Verilog HDL. To make sure all functionality was there, we utilized clock cycles and a reset trigger as well. For testing, we forced the clock to be a certain rate, and we also forced the reset to act as a clock cycle to ensure that all 4-bit numbers were being properly displayed and that the reset signal was functioning correctly.

Finally, we simulated a clock divider as our last simulation. We also directly implemented the clock divider on the FPGA board. To ensure consistency and functionality of the reset signal, we slowed down the clock rate and ran our code for longer time intervals.

**Conclusion**

During Lab 0, we ran into a few major issues. The actual software itself on our computer crashed and stopped working during the lab, and the computer threw a “no more disk space available” error, so we lost all our work and were forced to restart the computer in order to fix the issue, but restarting the computer also didn’t work. Because of this, our TA, Logan Kuo, told us to skip the simulation portion and move straight to the FPGA implementation because our simulation on our computer wasn’t working.

‘

Ultimately, this lab helped introduce us to the tools and technology that we will be using throughout this class. Now, we are much more comfortable with the overall process of development, simulation, and implementation despite the errors we encountered.