CSM152A – Lab 1

Names: Jahan Kuruvilla Cherian; Kevin Xu

Group: Lab 6 – Group 9

UID: 104436427; 704303603

Date Performed: 4/12

TA: Babak Moatamed

1.0 INTRODUCTION

The aim of this lab was to introduce and familiarize oneself with Verilog and the Xilinx ISE along with FPGA design with the focus being on the Nexsys 3 board. The first part of the experiment concentrated on understanding the simulations that Xilinx can provide you with along with understanding Verilog and test benching, through the use of given combinational module files. For the lab we focused on the combination gates muxed module, where we ran the simulation and implemented the design on the Nexsys 3 board. The second part of the lab required us to use some template Verilog to create a sequential 4-bit counter through gate level implementation and behavioral implementation. We had to run simulations for both versions of the counters and write our own test benches to do so. In addition, we created schematics for both of these using the RTL tool in Xilinx. Finally, we had to create from the ground up an implementation of a clock divider that we implemented onto the Nexsys 3 board. This required us to create the module from scratch and allowed us to understand how to use IMPACT to download the bitstream files onto the FPGA.

2.0 EXPERIMENT

2.1 Combinational Gates Muxed

Given the schematic shown in Figure 1 from the lab manual, we were asked to run the source code with the test bench provided to run a simulation of this design in Xilinx.

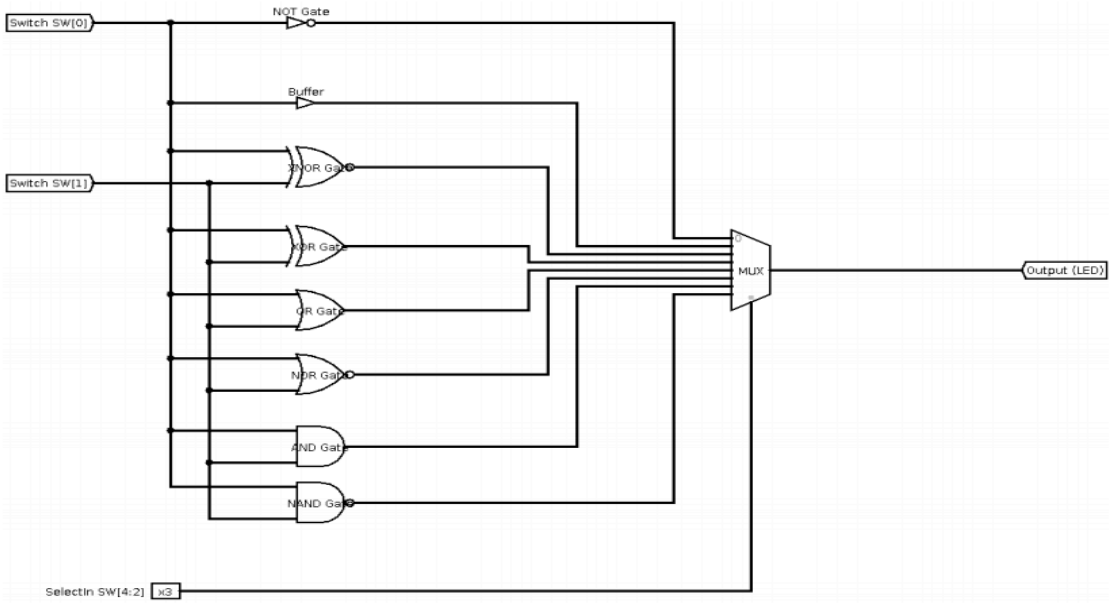

Figure 1: Design schematic of a combinational gates multiplexer. This module allows one to select different gates through the use of different switches to control the resulting outputs. The switches were mapped onto the Nexsys board using the UCF file provided to us.

Running the simulation within Xilinx required us to import the test bench code provided and run the UUT (unit under testing) Verilog module. The resulting simulation gave us the waveform shown in Figure 2. We see that sw[4:2] were our selection bits for the multiplexer; sw[0] controlled the input to the NOT gate and the Buffer, and was used as the secondary input for the other gates shown in Figure 1; sw[1] was the primary input for the 6 gates such as XNOR, XOR, NAND, AND OR and NOR gates shown in Figure 1. These switches were respectively mapped to physical switches on the Nexsys 3 board through the UCF file provided to us.

Figure 2: Xilinx simulation output for the combinational gates muxed, running on a clock with the timescale set to 1*ns*. As we can see the test bench varies the switch bits (sw[4:2]) which results ina different output that is reflected through the use of the led. This screenshot demonstrate a small time slice of the waveform produced by this simulation.

Upon implementing the module, we had to initially generate the bitstream file that came about by importing the UCF file. This essentially aided Xilinx in aptly mapping the variables used in the Verilog file to the respective physical components on the FPGA board. After generating the bitstream file we opened the IMPACT software and programmed the FPGA board with this bitstream file. The result was a demo that showcased the design highlighted in Figure 1.

2.2 Sequential 4-bit Counter – Gate Level

The schematic shown in Figure 3 demonstrates the module we had to implement for the sequential 4-bit counter. This schematic demonstrates the gate level behavior required and so the first part of the lab was to write a Verilog module along with a test bench that worked at the gate level, that is using individual wires and focusing on bitwise operations.

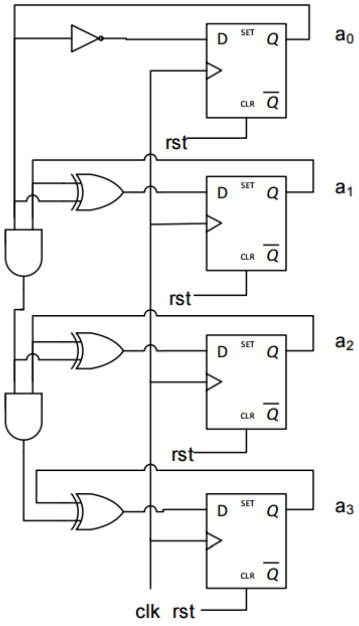

Figure 3: Sequential 4-bit counter design provided that demonstrates the working of the counter at a gate level. This schematic was implemented through gate and behavioral level Verilog, and the resulting simulation and RTL based schematics are provided.

The resulting waveform shown in Figure 4 represents the how the counter incremented each individual bit at every nanosecond time scale.

a0 represents our Least Significant Bit while *a3* represents our Most Significant Bit, and we programmed the Verilog module to increment each bit respectively at the positive edge of the clock running.

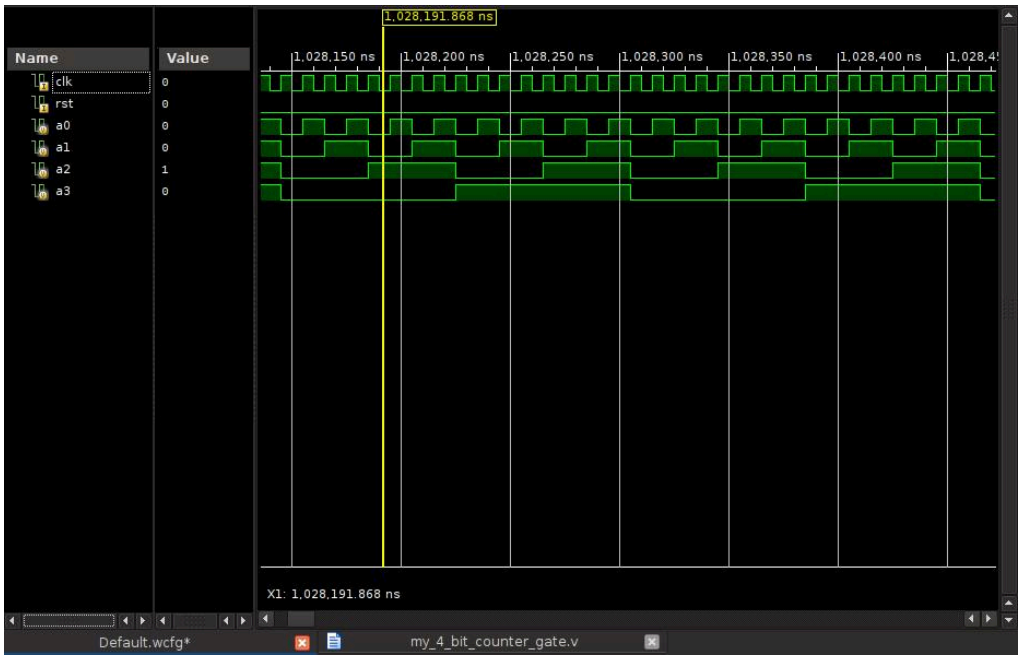

Figure 4: Xilinx simulation output for the 4-bit counter at a gate level, running on a clock with the timescale set to 1*ns*. The test bench runs through each instance flipping the clock which in turn makes the Verilog module increment the individual bits at the positive edge.

Using the RTL tool provided by Xilinx we were able to create the schematic that Xilinx believes is the correct implementation. This can be seen in Figure 5. For the most part we notice that this schematic is

very similar to the one given in Figure 3, which shows that when programming at a gate level with Verilog, we get very close to the actual desired implementation.

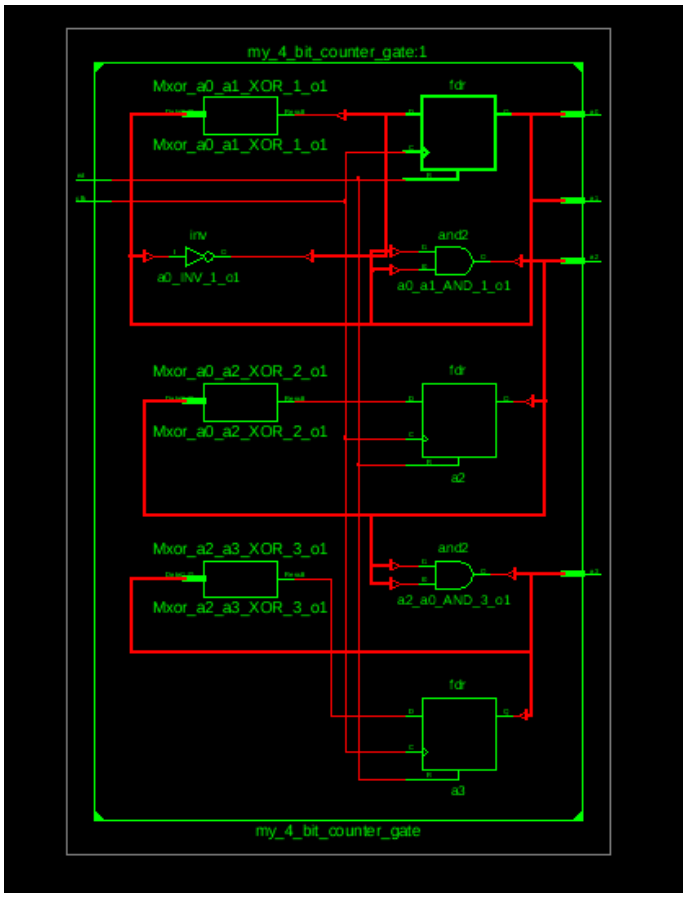

Figure 5: RTL based schematic that is used by Xilinx in its simulation of the module. We notice that for the most part this design sticks to the original but with a few extra modifications such as the use of only 3 *D*-flip flops, but still outputs 4 bits with the same logic gates in use.

2.3 Sequential 4-bit Counter – Behavioral

With this implementation of the counter we notice that the programming in Verilog was a lot easier since we were allowed to use simple arithmetic operations that Verilog uses to abstract away the lower level gates. The output is the exact same, but the simulation and the waveform as can be seen in Figure 6 show the results in somewhat of a different way. The register *a*[3:0] represents what our individual wires did in the gate level implementation, and it stores the result of the sum for every nanosecond.

Figure 7 shows us the RTL understanding of the design, which it just assumes as a generic in built counter module. This – as expected by behavioral implementation – abstracts away the details about the gates and flip-flops involved in the actual design of the module.

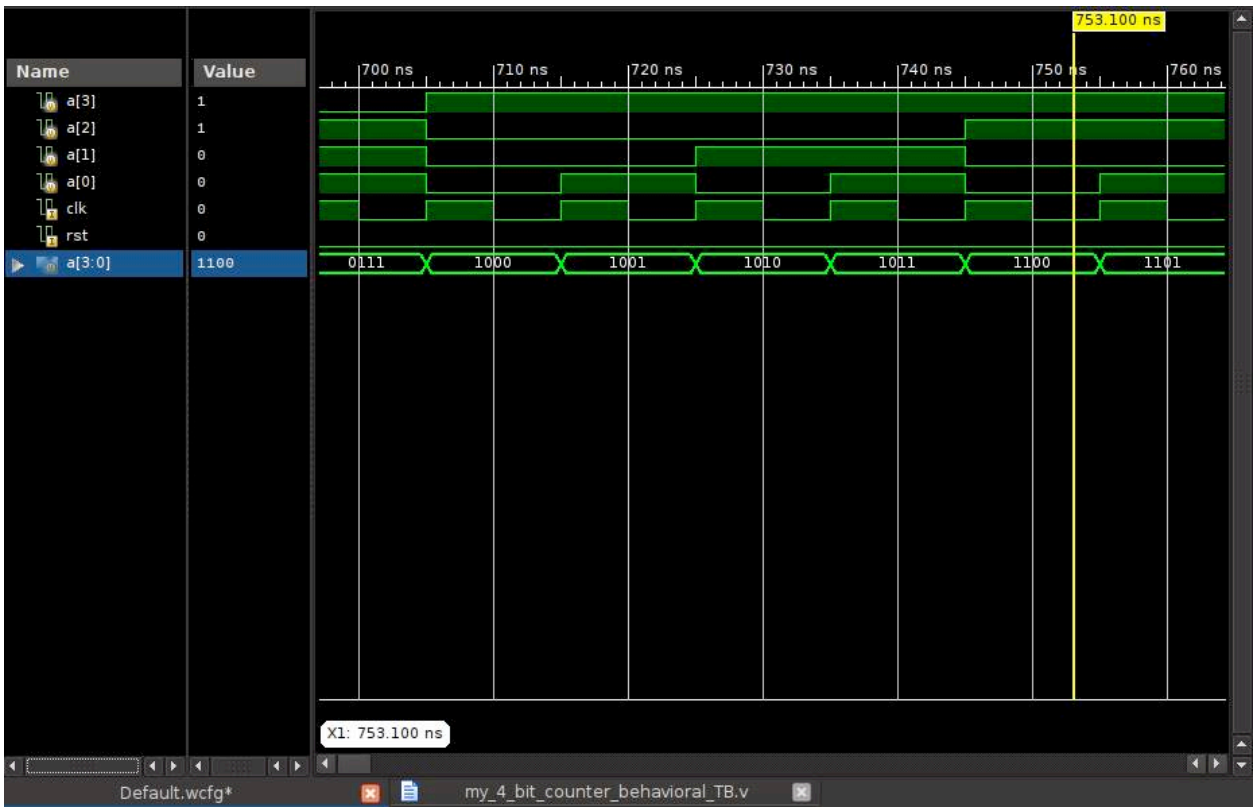

Figure 6: Xilinx simulation output for the 4-bit counter as a behavioral implementation, with the timescale set to *1ns* with the changes in the register that stores the count happening at every positive edge of the clock.

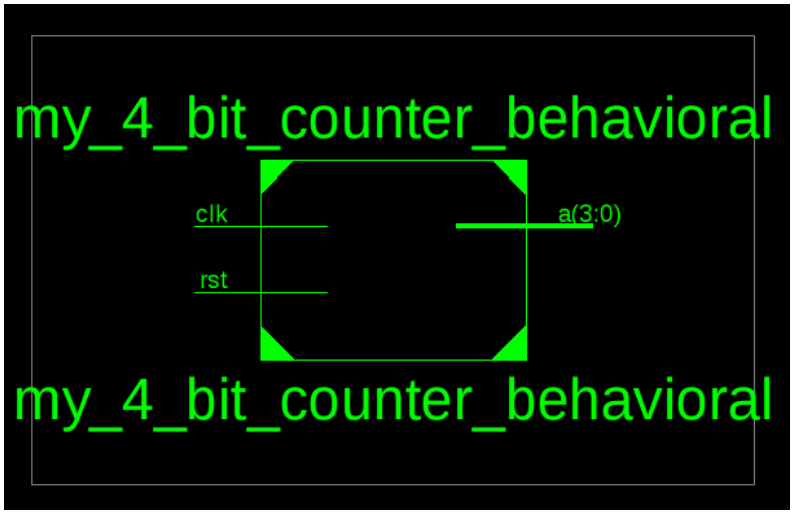

Figure 7: RTL based schematic of the behavioral counter. We notice that this is exactly the same as representing the counter as a block module in design, because the behavioral aspect of Verilog and Xilinx abstract away the more complex innards of the counter.

2.4 Clock Divider

The aim of this final section of the lab was to create a frequency divider by focusing on the master clock of the Nexsys 3 board. This master clock runs at a 100MHz, and we were given the task of reducing the clock frequency to 1Hz. The output of this division was to be reflected through the flashing of an LED on the Nexsys 3 board. The only part we had to be careful on was that though we had to go from 10^8 down to 1, we had to use a division factor of $5x10^7$ so that the clock would go up and down within this one cycle. We wrote the behavioral Verilog with the use of a custom made comparator that was used to see when to flip our 'new_clk' which represented the LED. Using a generic Nexsys 3 UCF file, we commented out all the lines that didn't affect our module and focused on correctly mapping our 'new_clk' to a valid LED on the board, and mapping the master clock to our input variable clk.

We used the IMPACT software (similar to what was done in part 1 of the lab) to download the generated bitstream file onto the FPGA and program it to flicker the LED on and off every second.

3.0 CONCLUSION

This lab taught us how to effectively navigate and use the Xilinx ISE for implementation, simulation, schematic design, and onboard functionality using IMPACT. It also taught us to write behavioral and gate level Verilog for the respective modules required. With every Verilog module, we wrote a corresponding test bench with which we could confirm our design through the simulation and waveform Xilinx provides us with.