**CSM152A-LAB 1**

Lab 5

Yichen Lyu

UID: 004940413

Apr 19, 2020

**1. INTRODUCTION**

This lab will explore how to design basic modules of Verilog and how to use the Xilinx ISE as a simulation and synthesis tool. There will be 4 modules to design, as well as their test bench. The first module is a combination circuit. The second module asks for the implementation of a 4-bit counter with D flip-flops, a design we have previously learned in M51A. The third module is a modern version of second module, in which we use a higher-level abstraction provided by Verilog. The fourth module is a clock divider, which utilizes a counter to divide the period of an existing clock and form a new clock.

**2. DESIGN DESCRIPTION**

**2.1 Combinational Circuit**

The design of the circuit is given in the project specification. It takes in an input [4:0] SW and outputs a 1-bit output LED. In my design, the MUX takes in a 8-bit input [7:0] mux and [4:2] SW as select.

A screenshot of a cell phone

Description automatically generated

Figure 1: Design of Combinational Circuit

**2.2 4-bit Counter**

The design of the counter is given in the project specification. It implements the counter using 4 D flip-flops. In my implementation, it takes in 2 1-bit inputs: CLK and RST, and has a 4-bit output OUT. Upon each clock cycle, when RST is 1b’1, OUT is reset to 4’b0. Otherwise, OUT is set according to the circuit design below.

A close up of text on a black background

Description automatically generated

Figure 2: Design of 4-bit Counter (D flip-flop)

**2.3 4-bit Counter Modern**

The code for the counter is given in the specification. It takes in 2 1-bit inputs: CLK and RST, and has a 4-bit output A. When RST is 1b’1, A is reset to 4’b0. Otherwise, every clock cycle, A will increase by 1.

**2.4 Clock Divider**

The clock divider divides a 1-kHz clock to 1-Hz: It divides the clock by 10,000. In my implementation, it takes in 2 inputs: CLK and RST, and outputs a new clock: DIV\_CLK. The module keeps a 14-bit counter. Once the value stored becomes 5,000, the divided clock changes phase from 0 to 1 or 1 to 0.

**2.5 “.UCF” Files**

“.ucf” file is user constraints file. It is a constraints file format that restricts the rules for the board. For example, when we want to connect pins to the switches and LEDs of a combinational circuitry, we need to have a .ucf file to specify how the inputs are mapped. We can use I/O Pin Planning (Plan Ahead) – Post Synthesis for Nexys3. When it is running, we can select inputs and drag them to certain pin location. After all inputs are mapped, we can save and exit. This will create a .ucf file which contains lines like NET “led[0]” LOC = U16, which assigns led[0] to the pin U16. (Duckworth)

**3. SIMULATION DOCUMENTATION**

**3.1 Combinational Circuit**

For the combinational circuit, I write a test bench that loops for 32 times, each increasing SW [4:0] by 1. In this way, we can test out the result from SW = 5’b00000 to SW = 5’b 11111. The simulated waveform is shown below, and I have confirmed by hand calculation that the output LED conforms to the circuit design in the specification.

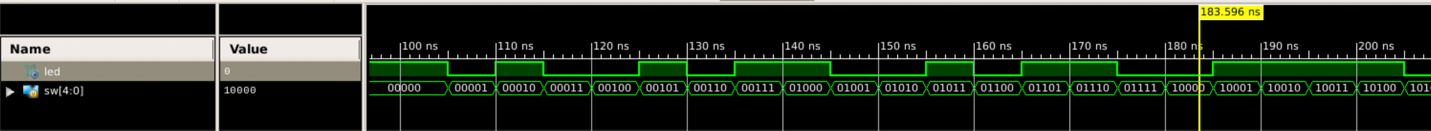
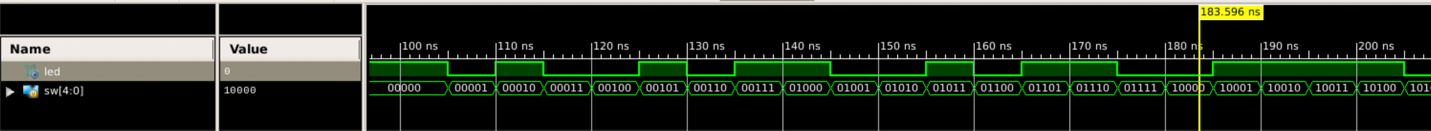


Figure 3: Combinational Circuit Waveform

**3.2 4-bit Counter**

For the 4-bit counter, I wrote a 100mHz clock and had it repeat for 100 times. Every time the clock flips, the counter will be incremented by 1.

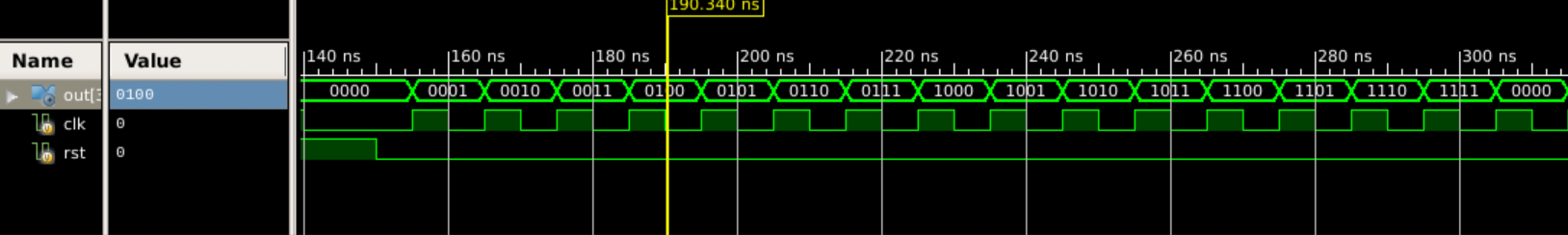


Figure 4: 4-bit Counter Waveform

**3.3 4-bit Counter Modern**

For the modern counter, I used the same test bench as the one for a D flip-flop counter. The simulated waveform looks the same.

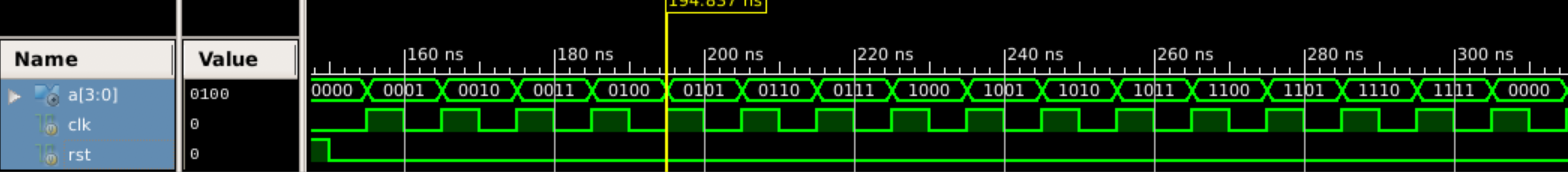


Figure 5: Modern 4-bit Counter Waveform

**3.4 Clock Divider**

The clock divider is implemented by counting the clock cycles of a 10kHz clock. Once it counts up to 5000, the divided clock flips. The test bench implements a 10kHz clock by waiting 50,000 ns before every flipping. The simulated waveform are as fallows: CLK\_1HZ has a period of 1,000 ms, which is a second, and CLK has a period of 100 ns, which makes its period 10kHz.

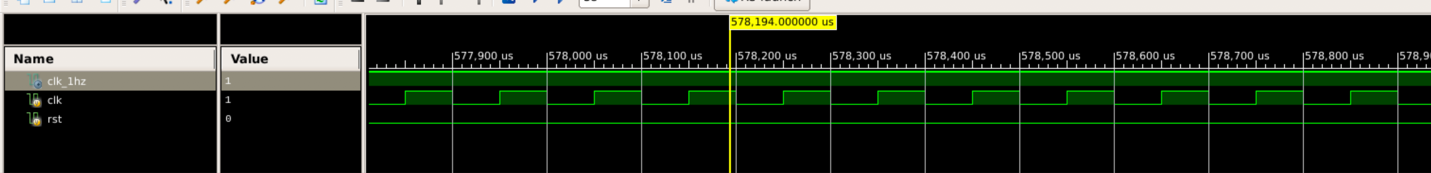


Figure 6: Clock Divider Waveform

**4. CONCLUSION**

This lab allows us to explore the functionality provided by Xilinx ISE and to understand basic design using Verilog, for both sequential and combinational logic. It also facilitates understanding of the structures of the modules as we are asked to write test bench and simulate waveform on Xilinx.

**5. REFERENCE**

Duckworth, Jim. “Nexys 3 board tutorial,” Aug 2011. WPI. http://users.wpi.edu/~rjduck/

Nexys3%20ISE%2013\_2%20Decoder%20Tutorial.pdf

**6. APPENDIX: Synthesis and Implementation Report**

A screenshot of a computer

Description automatically generated

A close up of a piece of paper

Description automatically generated

A close up of a piece of paper

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a computer

Description automatically generated