Please attach this cover sheet to your lab report

ECE324 Lab 5: Waterfall to Bouncing Ball

Name(s):

Alex Blake

Jameson Shaw

Exercise	Course outcome	Grade
Lab5 Demo	2.a, 2.d, 5.c, 7.b	/15
Lab5 Extra Credit	2.a, 2.d, 5.c, 7.b	/3
Lab5 Report	2.a, 5.c, 7.b	/25
	TOTAL:	/40

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.
2.d. Produce FPGA designs that meet specified needs.
5.c. Collaborate with individuals with diverse backgrounds, skills and perspectives.
7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction

In this lab, students were asked to take existing code and make modifications to it to ensure the sequential logic works as expected. This means making significant changes as some components are not synchronous. By the end, the project should be synchronous and have functioning sequential logic and be uploaded to a physical board and understanding what and how many resources are used on that board to produce the circuit.

Procedure

The first step is to obtain the code from the professor. This will include a main .sv file which will call upon a free run binary counter, mod m counter, universal binary counter, and the design constraint file for the Nexys4DDR

The files should come in a state where they can be synthesized and uploaded to the FPGA board and should function. Test this and make sure there were no issues to begin with. From here, many modifications will be made so it is important that the base version works correctly.

The code is given so that it simulates a water dropping from a high distance of 115 feet. The lab requires this to be changed from 115 feet to 4 feet. To do this, the free-run binary counter is replaced with a modulo counter. The modulo counter works by taking the modulo of the counter and each time that modulo is zero, its outputs timeBaseTick. This essentially allows any counter value to be used rather than just the number of bits like the free-run counter.

While the circuit has been simulating a circuit, it now needs to be changed so it acts like a ball bounding. First, replace the always block generating t with a universal binary counter. The counter should be synchronous, so the clock will be tied to the 100MHz clock. Syn_clr, load, and d are all set to 0; they are not required for this situation. Just like the flip flop that previously controlled t, it should increment only when timeBaseTick is high, so that is tied to the enable of the counter. Now the output q can be tied to t, which will now completely replace the flip flop. Since the t value is now an active output of the counter module, it should not be assigned to 0 when it is initialized; this is because its not storing a value like a flip flop now, but instead it is an active sequential logic output like a wire.

The last part of this step is to be able to change the direction of the counter to simulate the ball coming back. This is done by the up input, so create a simple logical up which will be controlled in an always_ff block. Within this block, up will change given max_tick or min_tick. Given these edge cases, up will change to either 0 or 1 to change direction. Synthesize the module and prove it works as expected. This concludes the lab.

Results

After downloading the code, the synthesis process went without a problem and the code worked exactly as expected on the board. This part was not required to get checked off, so we moved on after that.

Part 2 required a bit more work. Once we figured out how the equations worked to calculate timeBaseTick, it was simple to work backwards to solve for M to modulo to get a correct frequency.

To get 4 feet, the following was calculated:

$$4\,feet = 16t^2 \rightarrow t = 0.5seconds$$

$$T_{tbt} = \frac{0.5seconds}{2^{17}} = 3.81us$$

$$M_{tbt} = 3.81us(1x10^8) = 381$$

M_{tbt} is the value that the modulo counter will use to create timeBaseTick every $3.81us$ which will result in a fall-time equivalent to 4 feet.

The mod counter was then initialized with M=381 and its max_tick output driving timeBaseTick instead of the free run binary counter (now commented out). After synthesis, the code was implemented on the board and worked exactly as expected and was checked off by the instructor.

Now that we knew how to manipulate timeBaseTick, we had a good understanding of how the module worked and how it calculates the values to be displayed on the display. The next problem is to change water drop to a ball that bounces. This is best explained by Figure 1, where the drop acted like a sawtooth wave. Instead of the counter going up to max_tick and then resetting, it should go up to maxtick, then go back down to zero just like how the height of a bouncing ball would. To do this, the t is changed from an incrementing flip flop to a counter.

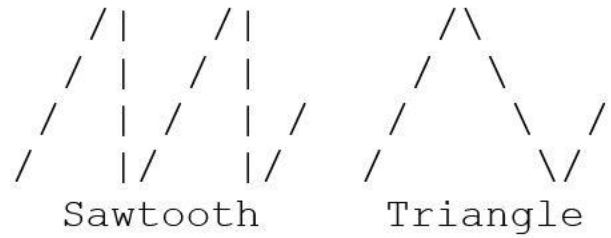

Figure 1: Sawtooth (Water Drop) vs Triangle (Ball Bouncing) (from Lab4 Handout)

Another problem is the up input. It needs to be changed outside of the module to change on min_tick and max-tick to change direction. This was done in a flip-flop always block with two if statements.

This was pretty simple minus the fact that we forgot to change the initialization of t so that it doesn't assign it to 0. This is a big problem; if t is set to 0, it is simultaneously an active output of a circuit, so its trying to be assigned like a flip-flop and a wire. This does not work and must be fixed.

Once that was fixed, the code was implemented and worked perfectly and was checked off by the professor. Figure 2 shows that 64 LUTs, 43 FF, and 1DSP were used to execute this circuit in the FPGA. This is minimal utilization compared to the available amount. This concludes the lab.

Resource	Utilization	Available	Utilization...
LUT	64	63400	0.10
FF	43	126800	0.03
DSP	1	240	0.42
IO	17	210	8.10
BUFG	1	32	3.13

Figure 2: Hardware Utilization of the Nexys4DDR

Conclusion

By modifying a circuit, students were able to use 16 LEDs on a Nexys4DDR to model a ball bouncing 4 feet. This lab was great practice on analyzing and modifying SystemVerilog modules as well as change asynchronous sequential logic to synchronous sequential logic. These skills are fundamental building blocks of designing hardware within SystemVerilog and will be useful in the future for solving harder problems.