Please attach this cover sheet to your lab report

ECE324 Lab 7: Traffic Light

Name(s):

Alex Blake

Jameson Shaw

Exercise	Course outcome	Grade
Lab7 Demo	2.a, 2.d, 5.c, 7.b	/15
Lab7 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 are asked to design a finite state machine (FSM) using a sequential logic circuit to implement a traffic light controller. This builds upon all the previously learned material students have covered in the class including synchronous logic design. Students then will upload their synthesized projects into an FPGA discovery board to verify the design works on physical hardware.

Procedure

Before starting anything else, create a new Vivado project and add all the downloaded files required from the professor. Once these are added, synthesize the project and verify it synthesizes and works on the board. Note that the PWM signal driving the RBG LEDs are controlled by the 16bit value represented by the switches.

The first modification needed is to add sensors to the street lights. This is done by added the 4 buttons (BTNR, BTNL, BTNU, and BTND) in the constraint file. Now that they are enabled, 'or' the inputs of BTNL and BTNR and 'or' BTNU and BTND and have their input signals go through a shift register for metastability. Now modify the code so that if "greenA" is active for at least 6 seconds and the sensor detects Road B (and doesn't sense Road A) switch to 'yellowA'. 'greenB' should do the same, but if both sensors are detected, give priority to Road A.

Verify the above changes work by testing the buttons during each of the above situations. If these worked, continue to the next step.

The final step for this lab is to implement the flashing red state. To be more specific, its two states that need to be added to the FSM. It will be setup so that when BTNC is pressed, the FSM will change to this flashing red loop immediately and stay flashing until the button is pressed again. The FSM should also be defaulted to flashing red when it is powered on.

Make sure that the BTNC is enabled in the constraints file. After that, use a shift register for metastability. Unlike the other buttons, this button requires debounce because it does not need to be held down like the others do. This is accomplished by sending the input through a debounce module and a rising edge detector.

The BTNC should switch any state to the flashRedOn state. In this state, the red lights are turned on, delayed for 1 second, and switch to flashRedOff. This state turns all lights off and waits 1 second and then loops back to flashRedOn. This loop continues until it is broken by the input of BTNC.

Verify that the code uploads properly and works on hardware. When showing the professor, make sure to set the PWM signal to 25% for credit. This completes the lab.

Results

After downloading the files and setting up the Vivado profile, the code synthesized and was implemented to the board and worked exactly as expected.

The implementation of sensors to the circuit went smooth as well. The addition of the shift registers was easy to do and using those new sensor inputs were used within the corresponding green states so that they change the timing. One issue we had with this section was that we misunderstood the directions of this part and realized after the instructor informed us that Road

A is always supposed to take priority over Road B if on Road B with both sensors working. This was a simple fix and was more of a confusion from the instructions than the coding itself.

Adding the two red flashing states was a bit more complex only because it required all the states to have an additional 'if' statement to catch if that signal goes high from the rising edge detector of the debounced input from BTNC. We had no issues here and this feature worked exactly as expected.

The overall utilization of hardware from this project can be found in Figure 1. Overall utilization for everything other than IO ports were less than 10%, so there is quite a bit more room for complexity and scale of projects.

Resource	Utilization	Available	Utilization %
LUT	69	63400	0.11
LUTRAM	3	19000	0.02
FF	107	126800	0.08
IO	42	210	20.00
BUFG	1	32	3.13

Figure 1: Utilization from Implementation of Project to FPGA Board

Worst Negative Slack (WNS): 5.059 ns

Figure 2: Worst Negative Slack of Project

The project was implemented with a slack of 5.059ns (Figure 2). Using this, the following frequency could be used.

$$5.059\,ns = Worst\ Period$$

$$\frac{1}{5.059ns} = 197.67MHz$$

This means that the project could be implemented with a 197.67MHz clock on the FPGA board before stability becomes an issue. This means our use of a 100MHz clock gives us plenty of room so we do not hit that cap and that all of our logic blocks should update before the clock cycle hits them so there is no propagation issues.

Conclusion

This lab was another great way to test students' ability to create a Vivado project and how to use sequential logic and case statements to create a finite-state machine. This street light controller circuit tested all the material covered in class so far and was great practice. These skills will be useful in the future for future labs and the final project for the class.