Traffic Light controller

ECE 2700 fall 2018

Austin Berger, Tyler Kavanagh, Remington Davids, Santiago Michel)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: AustinBerger@okland.edu, Tdkavanagh@oakland.edu, Remingtondavids@oakland.edu,
Svillarrealmich@oakland.edu

***Abstract*—The purpose of this project is to model a traffic light controller for a four-way intersection using the Nexys A7 FPGA Board that we used to control LEDs that are on a breadboard to represent the traffic lights. Which gave us a real life example of how a digital system works. VHDL was used to code the Nexys A7 FPGA board which control the LED lights. The VHDL code will control the timing of when the traffic lights change color. A finite state machine was used to control the color of the LEDs based on a timer and switch. While doing this project one of the major findings was that it was difficult to get the LED lights to all work at once and display the same color. The best way to do this project is to first come up with the pattern of the traffic lights which we did using a state table. Then the second part would be to figure out how to translate that pattern into VHDL code that can be implemented on the FPGA. Then the third part is to set up the hardware and bug out the issues. The traffic light is a very interesting final project because it gives us a better understanding on how traffic light works.**

I. Introduction

Traffic lights play an important role in keeping drivers safe and getting them to their destinations in a timely fashion. Most know the basics when it comes to traffic lights, but behind the scenes there is often a lot more than meets the eye. When designing an intersection, engineers need to determine the amount of time each light stays a certain color based on numerous factors. Does one road have much more traffic than the other? Perhaps having a longer green light for that road is appropriate. Do certain road conditions such as a steep decline require a greater stopping distance? A longer yellow light may be in order. These are just some of the many factors at play when a traffic system is being implemented. The intention of this project was for our group to explore a real-life example that had to with digital logic and we picked the traffic light controller. The most challenging part of this project was using the counter and fsm to control the lights and getting them to display the same color for the same amount of time. The things we learn during this project that was not taught in class were how to use the pmod ports which we used to link our LEDS to the Nexys A7 FPGA board. We also used a counter and fsm which we learn in class, so we knew how to use them. We also added a manual sensor into our VHDL code to act like a real sensor. Which when the sensor sees a car it would change the light for the traffic coming and we did this manual for a better demonstration. Also with a sensor in our traffic light it makes traffic flow more efficiently.

II. Methodology

This is the body of your report. Here you explain how you designed your project.

A. State table

The first part we needed to figure out when doing this project was how we were going to design our project and how we were going to change the lights. We did this by creating a state table to demonstrate what type of color signal the road would be getting. which in this case we used green, yellow, and red for the colors? The state table also gave us a better understanding of how many states we would need. After it reaches state six it stops until our manual switch is switched we did this to represent a sensor that would make traffic flow more efficiently.

North/south			East/west			State
R	Y	G	R	Y	G	
0	0	1	1	0	0	S1
0	1	0	1	0	0	S2
1	0	0	1	0	0	S3
1	0	0	0	0	1	S4
1	0	0	0	1	0	S5
1	0	0	1	0	0	S6

Figure 1: State table for the different states.

B. FSM

The finite state machine was the key part of the VHDL code because it let us control the order of state which the lights change color in. It changes the states based on the inputs that it receives and controls the time it stays on that

state. For instance, if the state is under a green light condition (s1) one of the lights will change to yellow upon entering the next state (s2). In our VHDL code we used if and when statements which controls the state it is in. Which is shown below in figure 3. For example, the state machine will say that the green light will run for 30 seconds and after the 30 seconds then turn to yellow for 10 seconds and finally red for a minimum for 60 seconds. The fsm is the main reason why are traffic light changes colors in the order it does

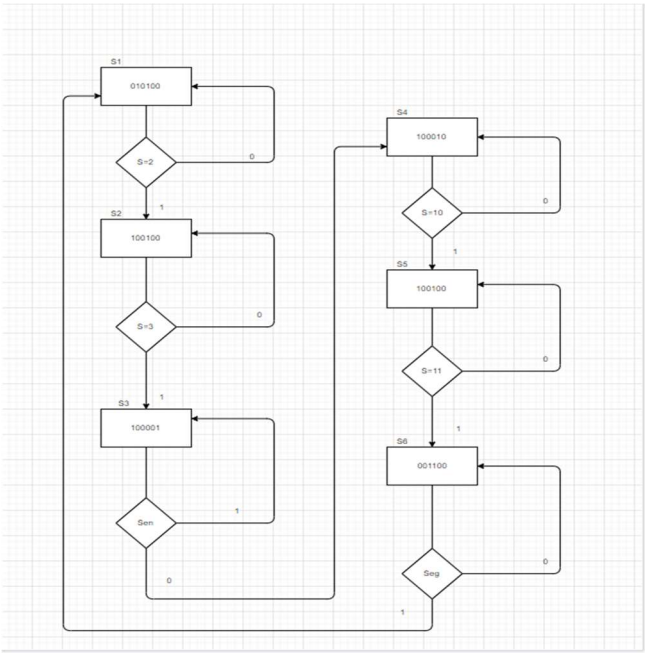

Figure two: State diagram of FSM

```
when S1 =>
    if sen = '1' then y <= S2; else y <= S1; end if;
when S2 =>
    if s = "0101" then y <= S3; else y <= S2; end if;
when S3 =>
    if s = "0111" then y <= S4; else y <= S3; end if;
when S4 =>
    if sen = '0' then y <= S5; else y <= S4; end if;
when S5 =>
    if s = "0101" then y <= S6; else y <= S5; end if;
when S6 =>
    if s = "0111" then y <= S1; else y <= S6; end if;
```

Figure three: Piece of FSM VHDL Code

C. *Counter*

The counter was used to demonstrate special traffic light states. To reach these states the counter was used to increment each time the FSM would pass a certain state. Once the FSM reaches a certain value the counter will automatically change the FSM state and stay there for some time.

D. *Top file*

The top file VHDL code is used to put all the different types of code and combine them in into one into one file. The top file and constraint file are the two main file that connect the VHDL code to the Nexys A7 FPGA board which allows use the Switches a pmod ports.

E. *Hardware*

When our VHDL code was done we programed a Nexys A7 Artix- 7 100T FPGA board. We then used Arduino wires to wire the FPGA board to our leds from the JA pmod port. We used two bread board, 12 LEDS and 12 resistors. We connected the resistor to the positive side from the pmod port which gives a logic signal to the LEDS. We then set up the ground for the pmod port which grounded our leds. We then used one switch of the FPGA board to represent a sensor.

III. Experimental Setup

After the VHDL was completed and working properly we set up our constraint file so we could wire and connect them to give them the appropriate logic signal. We used the 6 logic ports from the JA pmod ports to connect the wires to our LEDS. This will give them the logic signal that the need for them to change colors all at once and at the right time. After we made sure the program was working correctly with our leds. We created a testbench to show our inputs and how is cycles through the states shown below in figure four.

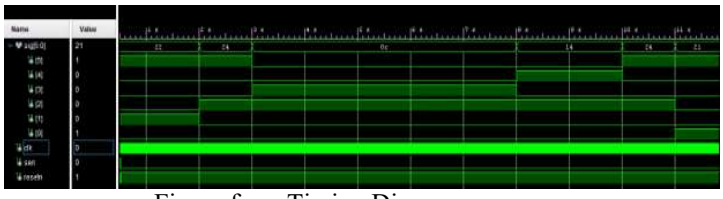

Figure four: Timing Diagram

IV. Results

The first time that the bitstream was uploaded to the FPGA the cycle of the traffic lights was too fast, it was discovered that the counter used in the system was not counting full seconds, to fix this more bits were added to the counter to account for the 100 MHz signal. The current code and hardware setup works as intended, the traffic lights for the main road are green unless a car needs to cross on the secondary road, if this occurs the lights on the main road will switch to yellow then red and after a few seconds the traffic lights on the secondary road will turn green, and after all cars in the secondary road have crossed the system goes back to the original state. A picture below shows the functioning traffic controller system.

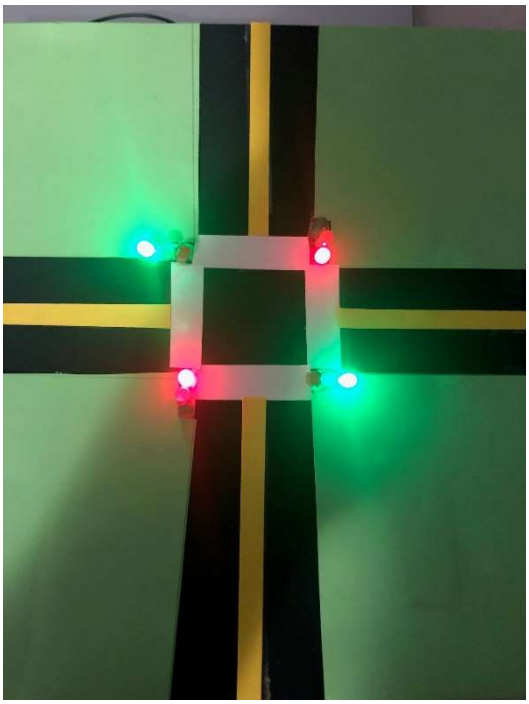

Figure five: Traffic light

CONCLUSIONS

The biggest takeaway from this experiment for our team was how a clock coincides with a Finite state machine to change multiple states of a traffic light. The most intricate part of this lab was to work out the bugs within our code in the Xilinx software. This experiment taught us how to wire the output pins from the Nexys A7 Artix-7 100T FPGA board, and then relay them through multiple wires to a breadboard. Being that we never really worked with the pmod output pins on the FPGA board became a bit of a struggle for us. Future experiments dealing with a traffic light controller can be made more user-friendly and have a better aspect of what these traffic lights do daily. We could have added an Emergency detection system for when emergency vehicles are coming or even make a backup system where all the lights would change to blinking red if there was a power outage or system issue for to show a better aspect of what real traffic lights would do in these cases as improvements to our project. Overall, we built a functioning traffic system controller and learn how digital design makes a big impact on a functioning traffic light controller.

REFERENCES

[1] Llamocca, Daniel. VHDL Coding for FPGAs, www.secs.oakland.edu/~llamocca/VHDLforF PGAs.html

[2] Nexys A7 Reference Manual. (n.d.). Retrieved from https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-

.