4-Way Traffic Light Controller

ECE 2700, Fall 2022

List of Authors (Lionel Yousif, Amanpal Madahar, Nicholas Sinawi, Rana Kakooz)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: lionelyousif@oakland.edu, amadahar@oakland.edu, nsinawi@oakland.edu, rkakooz@oakland.edu

***Abstract*—A 4-way traffic light controller was constructed through the Vivado software. It was implemented through two Nexus A7 50T FPGA trainer boards. The main portion of the code was created using an FSM model with two counters and six states within the FSM. The counters were used to delay the time between state changes, therefore adding a longer delay when a traffic light is green and a shorter delay when the light is either in the red or yellow state. Once the VHDL code was tested, it was then uploaded to the board and PMOD headers were used to distribute the signals from the boards to the LED's on the breadboard.**

I. INTRODUCTION

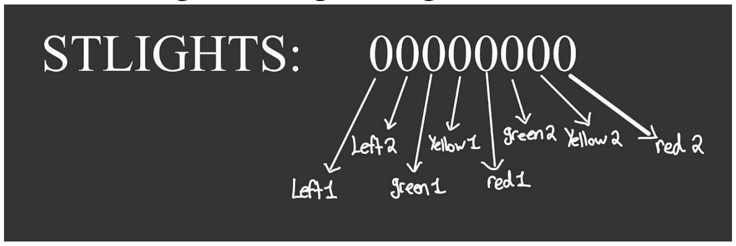
Traffic lights have improved much over the past few decades to help with the increase in vehicles and traffic all around the world. There are an infinite number of possible traffic light designs, whether it be sensor based or autonomous. The motivation for this design was to try and recreate an autonomous traffic light to see the manufacturing process behind them. The design of this traffic light controller is mainly architectural based. It is designed to work as a continuous traffic light controller that spans over four roads. It was discovered that the code can be duplicated in order to create a traffic light for the roads on the north and south, and east and west. There was no need to create a separate set of codes for the roads that were opposite of each other. Also a blue light is implemented into our design to constitute a yellow left turn signal. This project tested the skills learned in the labs and lectures within the class. The use of FSM's and counters shows the usage of the skills learned in class. Errors were then solved using the timing diagrams within the Vivado software. When creating the external components for the project, the group had to learn new methods of distributing the LED signals from the FPGA board to a breadboard with colored LED's. This design can be adjusted to real world conditions by applying changes to the counters. This will in turn allow for the design to be used within bigger LED's and a much larger traffic light.

II. METHODOLOGY

A. Finite State Machine

The FSM was a great source to use when coming up with the procedure and helped with the layout for VHDL for the 4 way traffic light controller. The FSM consists of six states each containing its own set of the sequence. The colored leds have an assigned binary number shown in Figure 1 below.

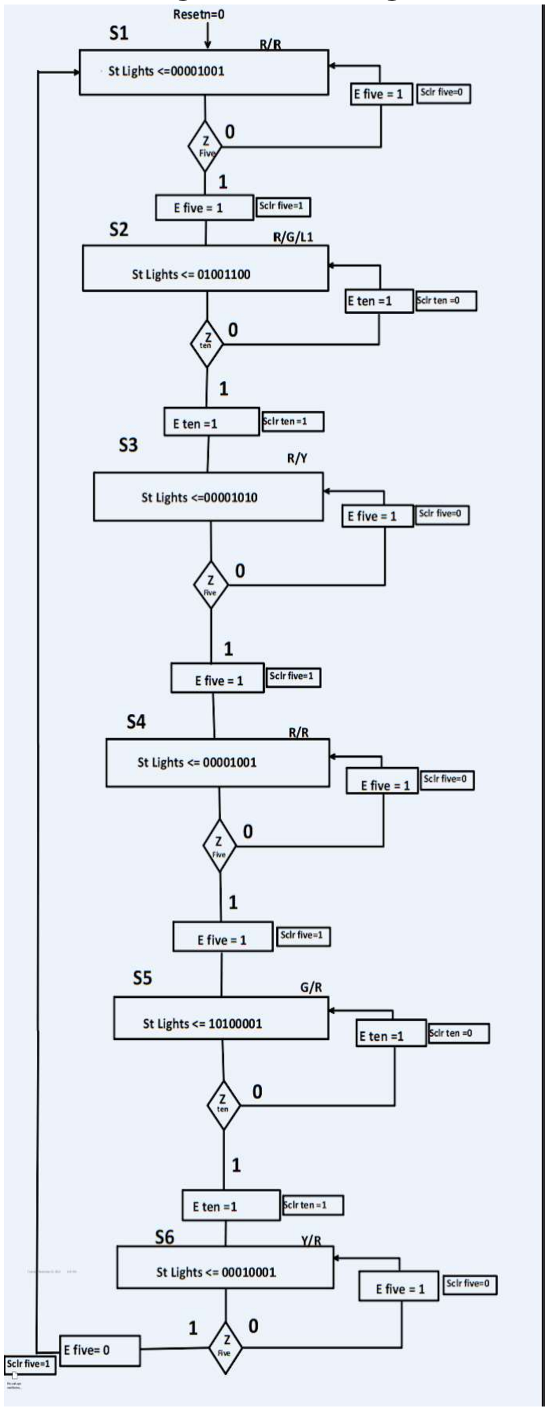
Figure 1: Stlights design



B. FSM

Starting with state one which will have all the lights at red/red. We have two counters, one counter is for five seconds the other for ten seconds in order for the sequence to move on to state 2 is if the counter has reached the time. The timer will be reset for it to move on to the next state and start counting up again according to the next state. More detailed look on FSM in Figure 2.

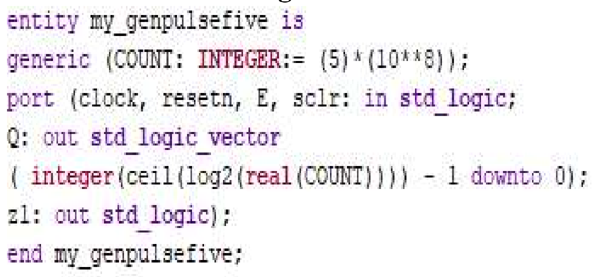
Figure 2: FSM Diagram



C. Pulse Generators

In order to create a delay between the traffic lights, the group decided to use code for a pulse generator. This code develops a pulse within the system for the LED's to turn off and on at specific time intervals. A snippet of the code is shown below in figure 1. This code was taken from Professor Llamoca's website and it was changed to meet the requirements of our project. The code uses an enable and sclr input and a z output. The main part of the code that was changed was within the snippet. The value that is put into COUNT is what affects the delay between states. The value in figure 3 is for the 0 to 5 second counter. So, in order for the counter to increase the delay during the green light state all that had to be done was change the value from 5 to 10. Also, the values of z and sclr were manipulated within the FSM to allow the counter to reset the counters to their first position so they can be used again within the next states. Within this project, the value of enable in both counters were set to 1 in the output process of the FSM.

Figure 3

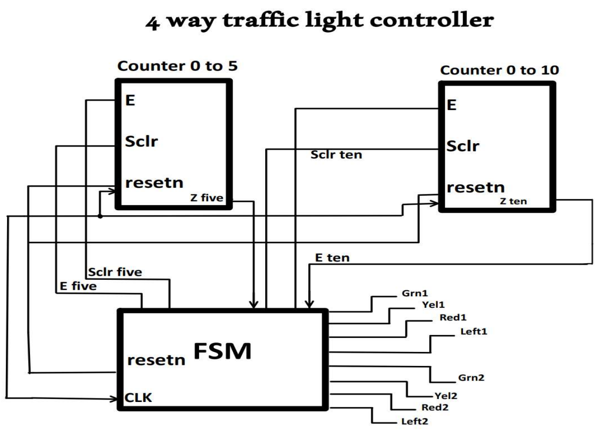


```
entity my_genpulsefive is
generic (COUNT: INTEGER:= (5)*(10**8));
port (clock, resetn, E, sclr: in std_logic;
Q: out std_logic_vector
( integer(ceil(log2(real(COUNT)))) - 1 downto 0);
z1: out std_logic);
end my_genpulsefive;
```

D. Testbench

The testbench is very essential when designing a project; it can help find errors within the code and develop a timing diagram that can also be used to solve errors. The testbench mainly takes the values from the top file and produces a diagram through the clock and reset inputs. Also, the values for each output light at the first state were placed into the testbench to begin the process. The testbench helped to solve a few early issues when determining which value in our stlights output went to each of the traffic light output values.

Figure 4: Circuit Diagram

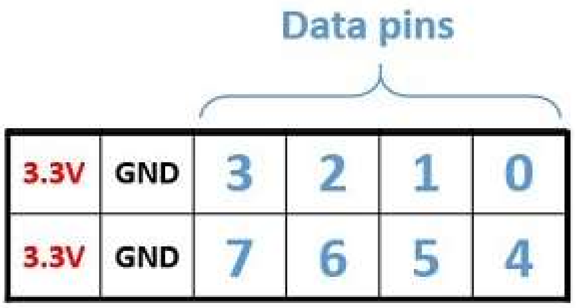


The circuit diagram shows the layout for the 4 way traffic light controller in an easy to follow layout from start to finish. Consisting of 2 counters that show what their inputs are and what outputs are going into the FSM for the FSM to cycle through its states and output the right combinations of light.

III. EXPERIMENTAL SETUP

The software used for this project was Vivado which utilized VHDL to synthesize and test the experiment. More specifically, a test bench was created to allow the experiment to be verified and later implemented onto a FPGA. Two Nexus A7 50T FPGAs were used in this project and wired to 4 different breadboards, each consisting of 4 LEDs. Each breadboard represented a street light with each LED symbolizing a color of the 4 street lights: red, green, yellow, and blue for the turning lane. To connect the LEDs of each breadboard to the FPGAs, the PMOD component on the boards were utilized in this case.

Figure 5: Pmod Headers



Data pins

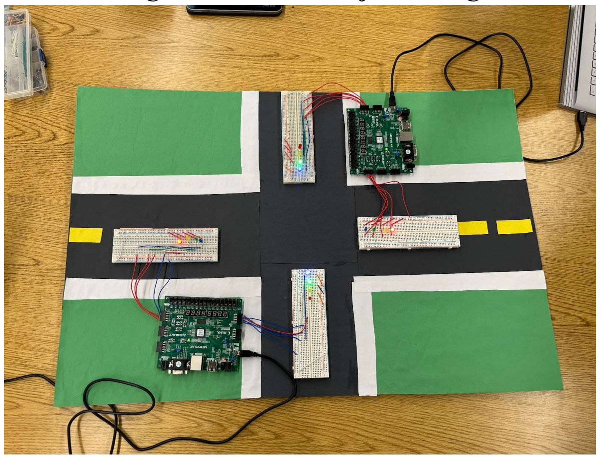
3.3V	GND	3	2	1	0
3.3V	GND	7	6	5	4

Each of the four LEDs were connected to data pins 3, 2, 1, and 0. And the GND pin was used to ground on each breadboard. The expected results are 4 working traffic lights with proper delays inserted to replicate a real life intersection.

IV. RESULTS

Upon completing the experiment the results that we obtained were as expected. These pictures will show the layout and the traffic lights at work.

Figure 6: Final Project Design



The picture above shows a snippet of the results that were expected.When one street has a green light to go for both directions and blue light is one for the left turns the street perpendicular is a red light hence the whole experiment is running accordingly. For the full demonstration of the experiment this video link shows the entire procedure, which includes a fairly busy intersection and a loop of the 4 way traffic light controller. Lights red and yellow have a duration of 5 seconds while green and blue have a duration of 10 seconds. The timing for the duration of these lights is low for demonstration purposes but of course when implementing in a real life scenario times can be bumped up accordingly.

Figure 7: Timing Diagram

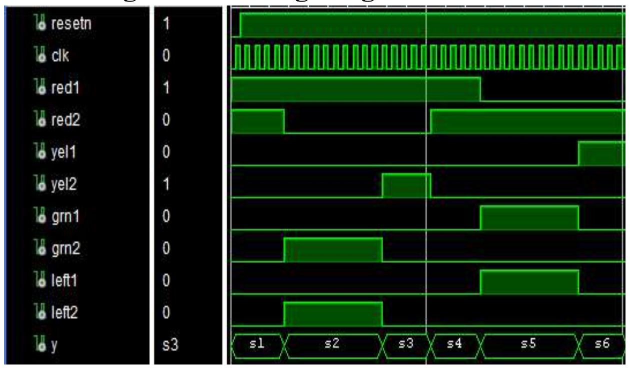


Figure 6 is a timing diagram that is a behavioral simulation of a 4 way traffic light controller. The simulation results were the most crucial because that would verify if the traffic light controller code is functioning correctly. The results in the behavioral simulation were as expected all throughout the simulation. Starting from state 1 as we can see red 1 and red 2 are both value one which means they are on. Continuing through the states the correct outputs have the correct values and time and no overlaps there for the timing diagram is working properly and to expectation.

V. CONCLUSION

The construction of the 4-way traffic light controller was successful in that it accurately mimicked a typical traffic light cycle. However, there are still many improvements that could be made. Further adjustments could be made to the counters to adjust the time between each state, allowing for more customizable and efficient traffic light cycles. Additionally, more complex systems could be implemented such as 'smart traffic lights' that detect traffic levels and adjust the traffic light cycle accordingly. Finally, the FSM model could be expanded to include more states and counters to create even more complex and customizable traffic light cycles.

REFERENCES

[1]D. Llamocca, *VHDL coding for fpgas*. [Online]. Available: http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html. [Accessed: 04-Dec-2022].