4 Way Traffic Controller

List of Authors (Fatin Kamash, Andrew Pettit, Alen Cehajic, Emmanuel Co)
Electrical and Computer Engineering Department
School of Engineering and Computer Science Oakland University, Rochester, MI
e-mails: fkamash@oakland.edu, akpettit@oakland.edu, acehajic@oakland.edu, eco@oakland.edu

***Abstract*—The purpose of this project is to design, model and create a traffic light that can command a four–way intersection and is able to direct a left turn lane. During the design process, it was found that a simple architecture would yield the best results. This project will contribute to the understanding of the digital logic within traffic light operation, as well as the analysis and production of digital logic systems in the future.**

I. Introduction

II. How do traffic lights work, and can they be made better? A 4-way traffic light will be created in order to gain more insight on their inner workings. This project will require the use of an FSM machine with binary counters, LUTs, and decoders. A VHDL code on FPGA is presented and uploaded to Nexys A7. This report will cover the methodology, experimental setup, results, and conclusions.

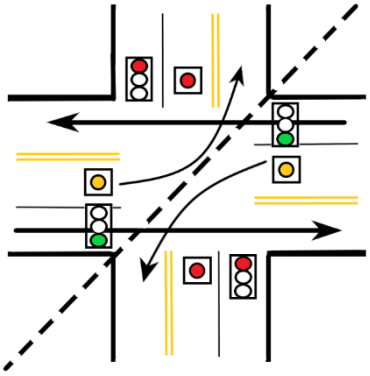
III. Methodology

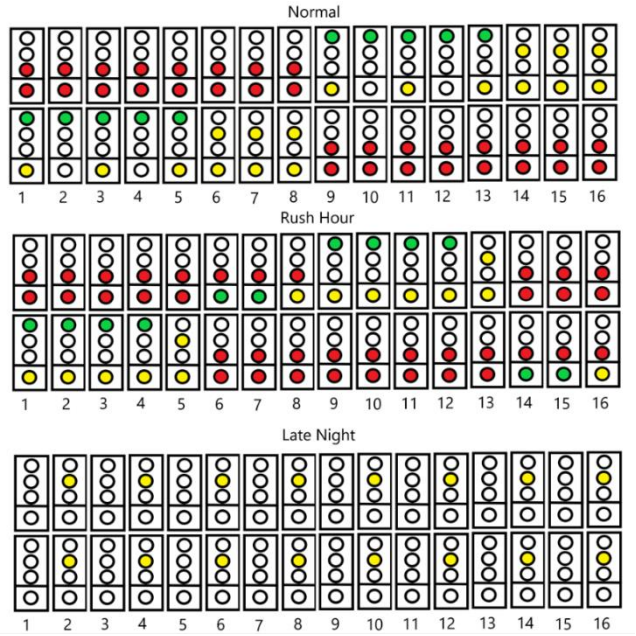
A. Using LUT'S

The first design problem was determining the number of LEDs needed and the total number of patterns required. Our design accounted for three modes of operations: Normal, Rush hour and Late night. Each mode of operation is represented by an LUT with 16 addresses. These addresses are the pattern of light that should be assigned to two adjacent intersections at a time at each count. To flip between these patterns, we needed a counter of 4 bits. 4 bits were needed to give us enough patterns until the three lights RGB Red, Green, and Yellow finished their cycle. The phases for the traffic controller were determined with a truth table that has 12 1-bit outputs, 1 output for each light. The truth table shows that there are 8 unique light combinations for traffic lights. Originally a 3 LED traffic light was considered but that was not effective because it did not take left turns into consideration.

B. Why using a 4 bit counter?

The original idea was to use a 3-bit counter that shifts between the different light patterns; this would not work for a traffic light because each pattern needs to have an addressable amount of time that it is activated. This problem was resolved by using a 4-bit counter and using the extra times slots to duplicate desired light pattern. For example, if a green light needs to be "ON" longer than the time allotted to one pattern by the counter, another identical address line (pattern) can be added after it in the LUT. This system allows for different light times while maintaining simplicity.



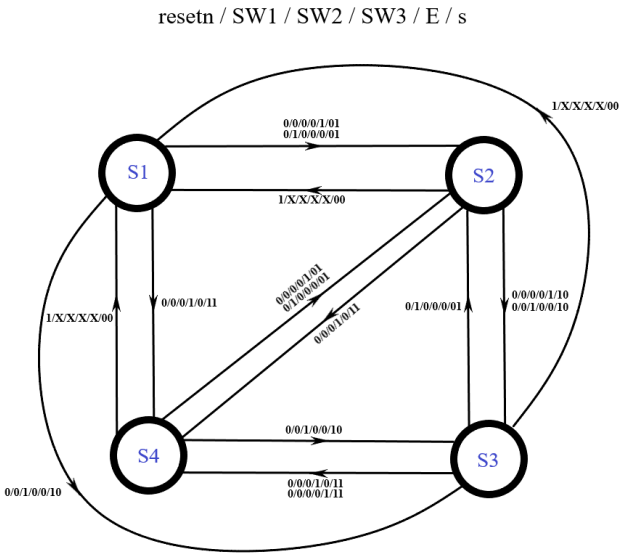


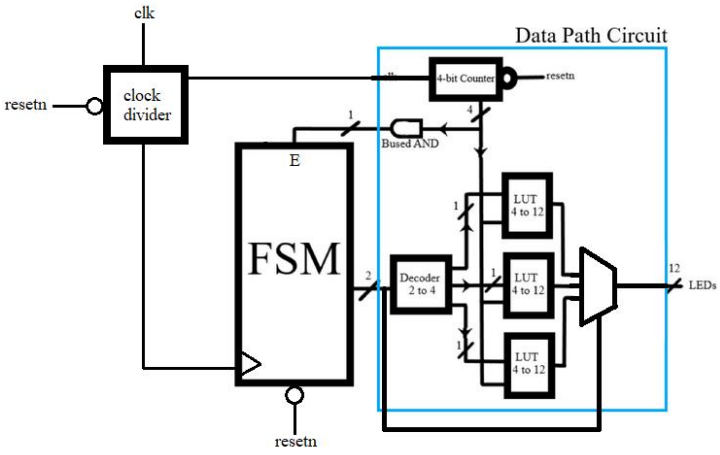
C. FSM machine

A Finite State Machine was used to flip between the four modes of operations. A state machine with Enable that is controlled by the counter will activate the machine and trigger the machine to transition to the next state. The output of the counter will be branched into two paths: one path to the FSM machine through an AND gate that will only be "ON" when receiving "1111" from the counter. the other output bus of the counter will travel directly to the LUTs to start counting and flipping between all patterns of a certain LUT.

We Needed four states to achieve this goal. The transition was seamlessly with the 4-bit counter. Each state is associated with an LUT. A 2-to-4 decoder was used at the output of the state machine to activate the associated LUT.

Switches were added to the FSM to provide a better functionality demonstration during the project presentation. The switches were able to manually shift states in real time while ignoring erroneous switch combinations. Here is the state diagram, and the result Block Diagram of the circuit:





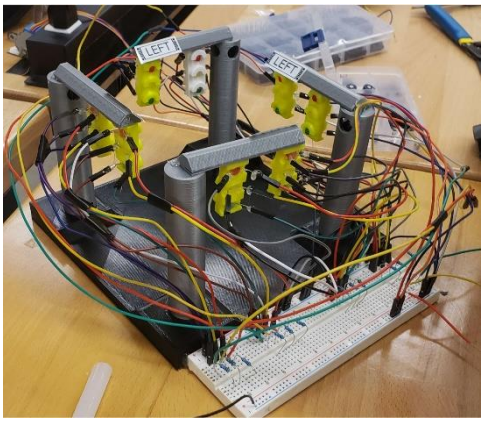
D. Final notes : clk divider and added switches?

We allotted about 2 seconds for each address line or pattern in the LUT. Which is about 32 seconds needed to cycle through all 16 patterns in each LUT. The counter module was assigned 0.5 seconds, but we needed the Clock divider to slow down the counting.

The circuit was designed so it can flip between State 2, State 3, and State 4 autonomously; however, we found it is more practical to integrate three switches into our FSM inputs. Each switch is assigned to turn on one mode of operation. For example, by flipping switch 1 On, the FSM will move to State 2 and output "01".

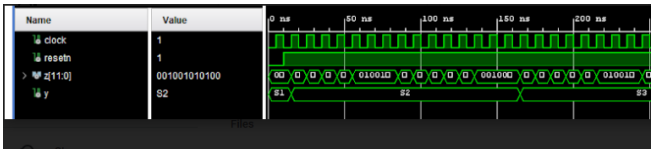
IV. Experimental Setup

The code was written using VHDL in Vivado software, The first step was using the on-board LEDs on the Nexys A7 to validate that our software works. Then a prototype of the external LEDs were created and connected to the external ports of the Nexys A7 and used the same program form step 1 to activate the prototype accordingly. Finally, we designed and created a 3D printed model of the intersection and connected the external LEDs from step 2.



V. Results

A simulation was run using the circuit and a test bench on Vivado. The traffic light controller was able to cycle through different traffic light patterns and respond to the off/on switch. The light pattern did not cycle until the previous cycle was complete. The traffic light controller was also able to respond to state switches properly. This shows that the finite state machine was able to read the Enable signal and the State Switches to change decoder inputs. Here is a picture of the simulation results.



A few issues needed to be addressed:

I. The simulation results were before the clock divider was added. The result was the LEDs would change at as high frequency that LEDs would be displayed as always on. The clock divider was added as an input to both the finite state machine and the 4-bit counter to slow the clock to approximately 0.5Hz.

II. When the switches were added as an input to the finite state machine, the output of the project

would prematurely change which LUT was being outputted due to using both the enable and the switches to switch stages of the finite state machine. The solution was to disable the enable input of the finite state machine whenever a physical switch was enabled.

III. If more than one switch was enabled at a time, the output of the project would seemingly at random switch which LUT was being outputted. The solution was to internally disable any switch input if more than one was enabled, this would result in the finite state machine to permanently remain in whatever stage it was previously to enabling more than one switch. The solution to this secondary problem was to activate the enable input for the finite state machine if more than one switch is activated at a time.

All these problems were resolved, and the end product was a 4-way intersection that cycles through all LUT tables through a timer and the ability to select which LUT table to use as per user input.

VI. Conclusions

Researching the design and creating a logical diagram proved to be more complicated than we initially expected. By looking into how speed limits affect the timing of traffic lights, we developed an estimate of how long each sequence lasts. The learning curve for this project was thinking of different logical sequences and choosing the most efficient one. Through investigating different 4-way intersections around Michigan, we concluded that our current design is the most efficient way in managing intersections with left turn lanes. Finally, we used the knowledge from this class about LUTs and binary counters and were able to program the NEMA A7 board. One improvement that could be made is the wiring of the physical project, the jumper wires would be replaced with soldered one, and wired inside the poles to make it professional looking. Ultimately, the research resulted in newfound knowledge and appreciation to the design of traffic systems.

VII References

[1] A. Pardo, “Finite State Machines explained,” *YouTube*, 30-Aug-2013. [Online]. Available: https://www.youtube.com/watch?v=hJIST1cEf6A&ab_channel=AbelardoPardo. [Accessed: 14-Nov-2022].

[2] B. Watson, “Sequential Logic Design Example (Traffic Lights),” *Sequential logic design example (traffic lights)*. [Online]. Available: http://www.barrywatson.se/dd/dd_sequential_logic_traffic_lights.html. [Accessed: 14-Nov-2022].

[3] D. Llamocca, “DIGITAL LOGIC DESIGN VHDL Coding for FPGAs,” *FINITE STATE MACHINES (FSMs)*, Oct. 2022.

[4] Intermation, “EP 063: Introduction to state machines: Designing a simple traffic signal,” *YouTube*, 26-Oct-2020. [Online]. Available: https://www.youtube.com/watch?v=gv5fQrD8XUo&authuser=1. [Accessed: 14-Nov-2022].

[5] nesoacademy, “Mealy and Moore State Machines,” *YouTube*, 17-Mar-2015. [Online]. Available: https://www.youtube.com/watch?v=0_OZKWdCixw&t=182s&ab_channel=NesoAcademy. [Accessed: 14-Nov-2022].