Four Way Traffic Lights

ECE 2700 Winter 2021

List of Authors (Justin Akhahon, John Antosh, Faristina Jackson, Daorsa Dulaj)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-Mails: justinakhahon@oakland.edu , jantosh@oakland.edu , ddulaj@oakland.edu , fjackson@oakland.edu

***Abstract*- In this project, we constructed a functional traffic signal for a standard city four-way intersection. The LEDs are intended to operate at a traffic intersection assuming traffic is going north, south, east, and west. We accomplished this by using VHDL code to implement our traffic signals, and programming VHDL code onto a Digilent Nexys 4 DDR board to verify our design.**

I. Introduction

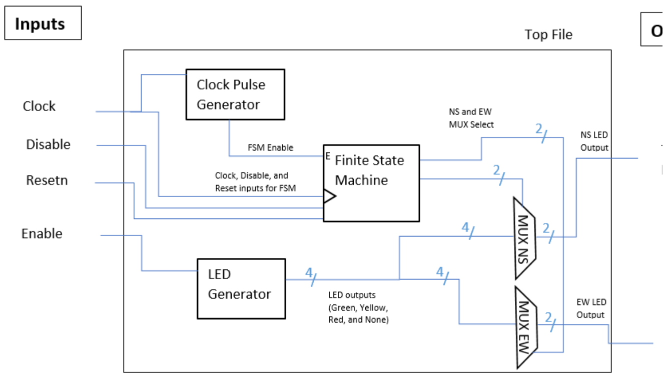
Since the introduction of the first traffic signal in 1868, traffic signals have always been used at road intersections to control the flow of traffic. In today's time traffic signals have been implemented in almost every major city around the world, not only does it control the flow of traffic, but it protects the commutes of pedestrians daily by using these specific colors (green, yellow, and red). Even though it is nearly impossible to prevent traffic jams from occurring, we can still innovate ways to reduce the duration of traffic jams. Especially those that occur on busy road intersections. Instead of implementing more roads that may cause more traffic, we can innovate our traffic signals to reduce the duration of traffic. For this project, our team designed a model traffic signal. This model will replicate the basic idea of a traffic signal at an intersection. This project also involves the use of an integrated circuit with various components, and a test bench circuit simulation which will help us examine our circuit and debug it so it will function successfully.

II. Methodology

In this project we created a top file and other corresponding files to support our system. The top file controls the output of our system which are two LEDs, one representing a road along the North/South direction and the other representing a road along the East/West direction.

Our first component is the clock pulse generator which takes the internal clock signal of our Nexys board as an input and transforms it into an output signal of 1 every half a second. The next component is a finite state machine. Our FSM takes in the clock pulse generator output as an enable input. In this way, the FSM functions every 0.5 seconds instead of every 10ns. The FSM, along with an embedded counter, manages the states of our system. The counter is used to control how long the states last and the FSM outputs are used to select which color the LEDs will have. We created an LED generator to generate our top file outputs. This component has the top files Enable as an input and four outputs that represent each color the circuit will produce. These outputs are fed into two multiplexers. These multiplexers receive their select input from our FSM. In this way, the FSM controls which of the four inputs in each multiplexer will become outputs. These outputs are then displayed on the Nexys board as LEDs.

Figure 1 (Block Design)



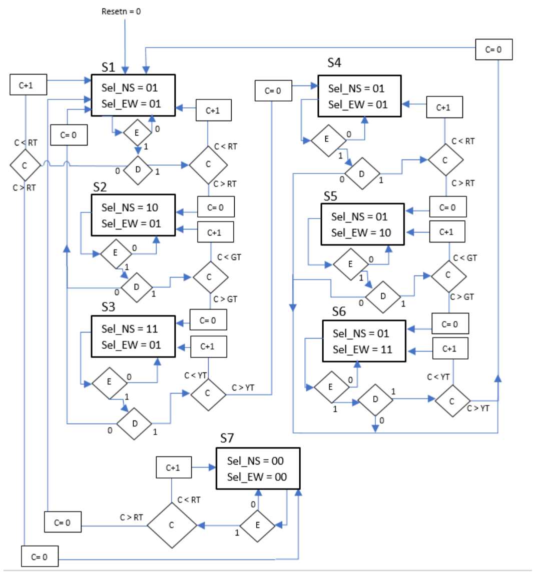
A. Finite State Machine:

The finite state machine also known as (FSM) are used to represent systems with a finite number of states. In our project we took our input signals of our function and used that to control our output functions. By using an FSM we were able to define our sequential logic for our output signals.

Our FSM has three inputs: Enable, Disable, and Resetn. The Enable input is received from the clock pulse generator. This in essence causes the FSM to either increment its embedded counter or change the state when the enable is activated every 0.5 seconds. The Disable input is received from the top file input and it causes the FSM to oscillate between the first and seventh states. This, in conjunction with the rest of the circuit, was used to emulate the effect of a blinking red LED. The Resetn input is received from the top file input and it causes the FSM to remain in state 1 (solid red LEDs).

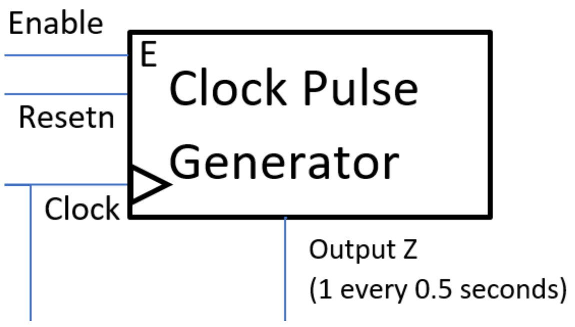
Our FSM has two outputs. Each output is a set of two binary numbers. These two sets represent what colors the final LED outputs should be.
(Green: 10, Red: 01, Yellow: 11, None:00)

Figure 2 (FSM State Diagram)

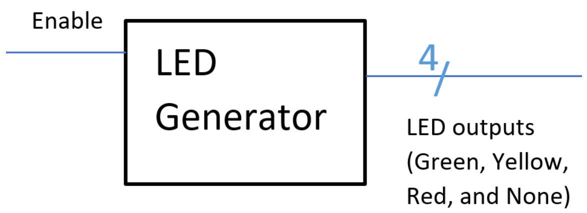


Pulse Generator:

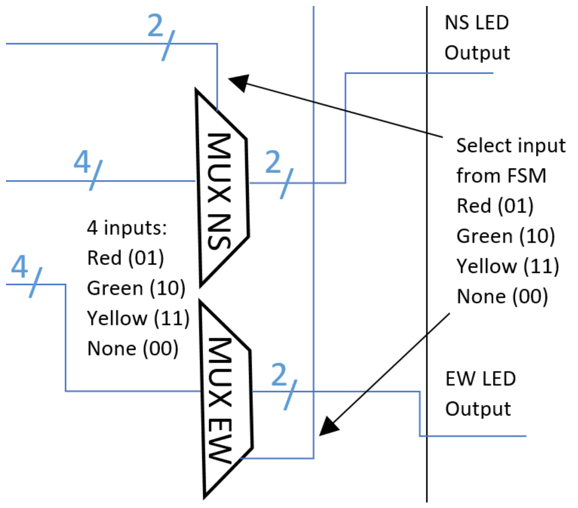
The pulse generator was used in our project to generate an output signal of 1 every half a second. With the integration of "if then" statements our group was able to implement a counter that increased on every clock tick. After our counter reached its limit, the output of the pulse generator would be 1. This essentially allowed us to convert an output of 1 every 10ns to an output of 1 every 0.5 seconds.



Led Generator:
The led generator was used in our project to generate all our desired led outputs: green, red, yellow, and none. This component's only input is Enable. This input controlled whether the generator would generate the normal outputs, or if Enable was set to 0, it would generate "none" for every output.



Multiplexer: We used two multiplexers to control which color our LEDs would be at any given time. The inputs from the LED generator and the select inputs from the FSM were designed to match. (E.g. if the select input from the FSM is 10 then the multiplexer would choose the input 10 from the LED generator)
Each multiplexer has an output that is connected to the top file LED output.



Test Bench: Lastly before we uploaded our program onto our board, we had to test it in our test bench file to make sure that everything was working properly. This test which was carried out in our test bench file was utilized to amend mistakes that occurred in our coding process.

III. Experimental Setup

Software: For this project, our group used VHDL to program our system. This is a programming language that we have been taught to use in our ECE 2700 class. Besides the normal setup, we had to set the configuration bank voltage to ground and the configuration voltage to 1.8V to avoid damaging the circuit.

Hardware: The VHDL code that was used in this project was constructed on a program called Vivado. This code was further programmed onto a Digilent Nexys 4 DDR board which computed our code. We used two LEDs on the board as our outputs and three switches as our Enable, Disable, and Resetn inputs. The expected results from our circuit is a functioning, easily identifiable sequence of lights that represent indications for traffic flow and multiple switches to set these lights into either an off state, a blinking red state, or a solid red state.

IV. Results

After multiple attempts of revising our code, we successfully got our code to run correctly on our board. After multiple group meetings and having limitations of resources we found it easier to limit our project scope to having a basic led behavioral sequence function and then add on the disable feature afterwards. Our group had some help with coding by using some references from Dr. Llamocca. Our group also was able to implement the use of a finite state machine as the main driver of the circuit function. This benefited us greatly for our project and was an important topic for the class. Our FSM state diagram was especially helpful when stepping through our circuit logic process and debugging issues.

V. Conclusions

Designing and building a model traffic signal wasn’t as easy as it appeared. Iterating through the debugging process was stressful when it failed, but rewarding when all the pieces began to come together. This project helped expand our group's experience with critical thinking, design, and VHDL code. We learned the benefit of using components such as a pulse generator and a finite state machine. The integration of these systems helped our project run smoothly. Even though we went through many challenges, our group was able to get a better understanding of the class material because of this project.

Some improvements that we could have made are to add more functionality for traffic lights that usually occur during different times of the day. We also could have implemented different time lengths for intersections that were experiencing heavier traffic by adding sensors to our project.

Overall, the design functioned as intended and taught us valuable lessons about critical thinking, the debugging process, and working as a group on a design project.

Reference

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