# 4-way Traffic Light

Nolan Mittison, Davio Mazzella
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: ndmittison@oakland.edu, daviomazzella@oakland.edu

***Abstract*—A 4-way traffic light is used to organize traffic and provide a safe way to cross an intersection. It was found that a breadboard could be used in conjunction with a Nexys A7 100-T to simulate an intersection. While working out how to implement two finite state machines, a multiplexer between both state machines allowed for an easy switch between the two modes. Overall, the project properly demonstrated how a traffic light is designed and implemented using VHDL code while incorporating a day/night system for a more comprehensive solution for traffic safety.**

## I. Introduction

This project is a 4-way traffic light consisting of forwards, backwards, left, and right directions. In addition, left turn arrows have been implemented for all four directions to increase safety. The physical lights are presented using breadboards wired with light emitting diodes driven by a Nexys A7 100-T. This project utilizes a counter that generates a pulse every one second to determine state for the traffic light's finite state machine. This traffic light uses two separate state machines to represent day and night.

This traffic light will create a safer driving environment, directing vehicles in a consistently safe, efficient manner. A period where a yellow light is active before each change to red, so as to give a driver time to slow to a stop. This controller was designed primarily with the safety of those drivers who may be navigating the intersection in mind.

## II. Methodology

Comparators were created in order to evaluate whether an appropriate amount of time had passed between light changes. This allows for proper timing while keeping the entire project driven off of one clock. A selector switch on the A7 was implemented in order to switch modes between day and night, whilst a breadboard display was used to give visual aid to the project (Figure 1). The greatest design challenge was in implementing two FSMs and getting them to work together without conflict.

### *A. Finite State Machine*

The 4-Way Traffic Light consists of two finite state machines. One controls the timing of the daylight mode (Figure 2), whilst the other controls the nighttime (Figure 3). Both finite state machines have twelve outputs each, connecting to a 2 to 1 multiplexor before continuing to their respective LEDs. The multiplexer allows for the switching between daylight and nighttime cycles with a switch on the Nexys Board.

### *B. Multiplexers*

To switch between modes, twelve 2-to-1 multiplexers were utilized. Both finite state machines are capable of separately generating signals to drive the traffic light's LEDs, requiring a way for the system to choose between the two. Multiplexers were the chosen method to accomplish this. All of the multiplexers are linked to the same mode switch, allowing all lights to be altered concurrently.

### *C. Clocks*

In order to properly change states in both cycles, a clock connected to five comparators was created. Whilst in day mode, the traffic light utilizes four, five, 15 , and 20 second comparators.When in night mode, only one comparator is used in order to create a blinking effect. This can be seen in figure 5, where a one second comparator is used. The counter sends out a pulse every second. The accumulator counts these seconds which are evaluated by the comparator to determine the state of the relevant FSM. Working with multiple comparators allows the program to follow a set path while minimizing the amount of code used. Use of a single counting circuit for both finite state machines eliminates the risk of the clocks becoming out of sync with one another. The process taken by the comparators can be seen within figure 2, where the timing diagrams affect each other as the counter is active.

## III. Experimental Setup

A test bench within the VHDL software was utilized to ensure the program worked properly before being implemented on hardware. To verify proper operation of the traffic controller within hardware, an A7 100t FPGA was used by connecting it to a breadboard with wires and representative LEDs attached as shown in Figure 1. Four breadboards were used to simulate a four-way intersection

with a left turn lane. Within the A7 100t FPGA board, two switches were used to start the cycle and change between day and night mode.

For software, Vivado was used to write and simulate the VHDL necessary to program the A7 FPGA. Generic components were borrowed from [1] in the construction of the project. Each generic component used is properly marked as such within its individual VHDL file. The state tables for both finite state machines are laid out in tables 1 and 2.

## IV. Results

Proper timing operation was observed through a behavioral simulation as shown in Figure 2. Initial versions could not be implemented in hardware due to a multiple driver issue. The problem was found to be in the way that the two FSMs were implemented. Both were trying to send conflicting signals to the same place. To solve this, a multiplexor controlled by the mode was implemented between them so that signals would only be used from the relevant machine. The video attached (Video of Working Project) shows the sequence used in the block diagram (Figure 3) when connected to the external hardware (Figure 1). Once both FSMs were correctly implemented within the project, the results shown matched exactly with expectations.

## Conclusions

This project demonstrates the utility of simple components when arranged meaningfully. It required seamless integration of the counter with the FSMs, while ensuring that the pulses generated by the counter were slow enough to allow the LEDs to change at a rate visible to the human eye. It required knowledge of the A7's physical construction to wire the outputs correctly to the breadboard where the LEDs were located.

The ability for pedestrians to safely navigate the intersection is an unsolved issue that may be addressed through implementation of pushbuttons to activate a walk signal that coordinates with the traffic light. This improvement would greatly increase safety for pedestrians and the project's versatility. In addition, this project requires manually switching between day and night mode. This could be automated through a clock or light sensor, changing modes based on time of day or daylight.

Other improvements could require taking road conditions, particularly speed limit, into account. A faster speed would require a longer stopping distance and therefore should have a longer yellow light interval.

## References

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

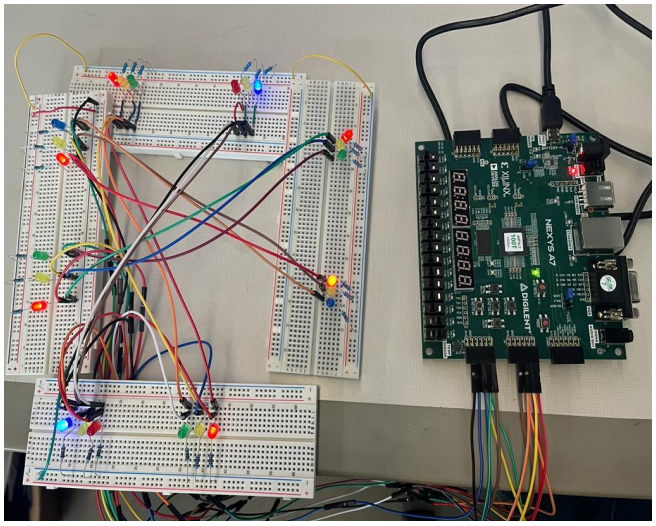

**Figure 1**: The breadboard/FPGA setup to demonstrate hardware function.

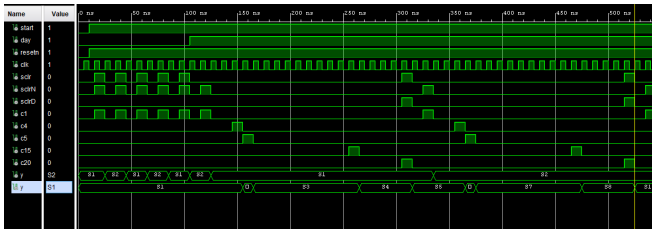

**Figure 2**: Timing diagram showing proper operation of the counter and FSMs.

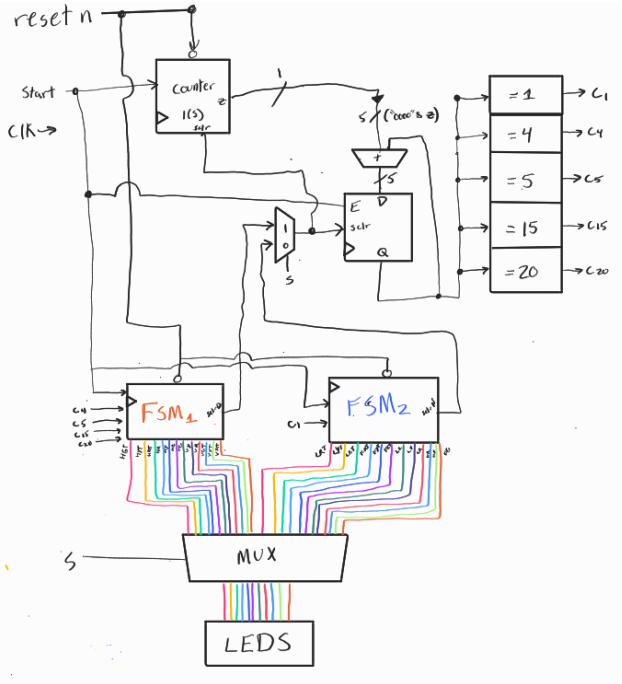


**Figure 3**: Block diagram of the overall system.

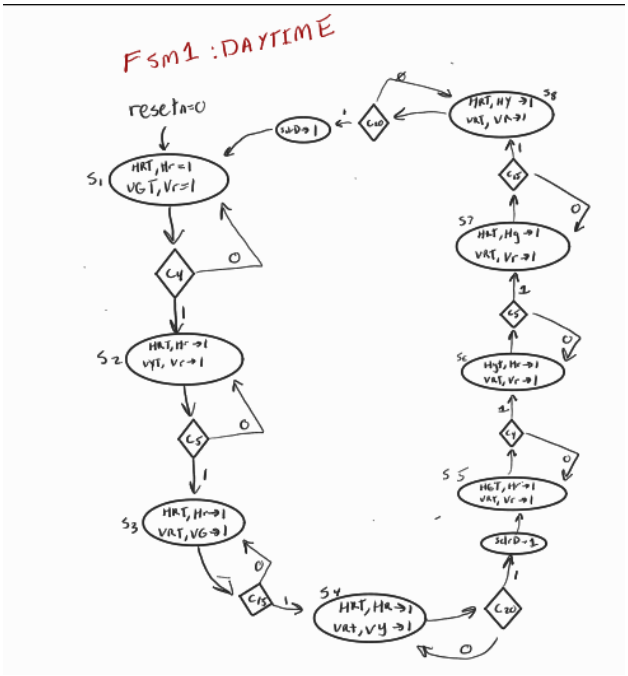


**Figure 4**: Selection diagram of the daytime finite state machine.

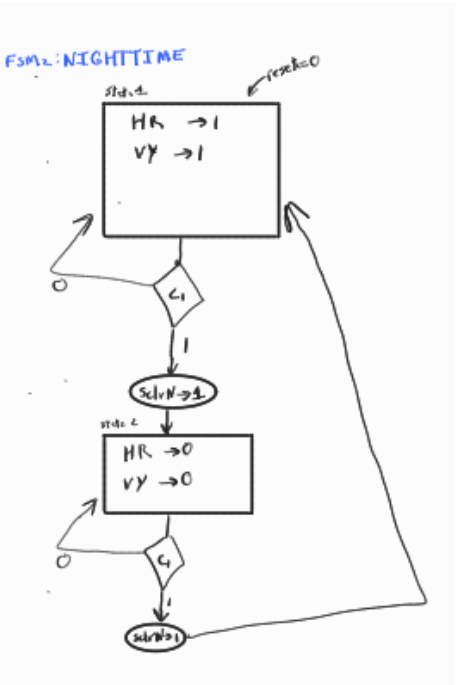


**Figure 5**: Selection diagram of the night finite state machine.

**Table 1:** State table for the Day mode

| State (Day) | East/West | | | | | | North/South | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HR | HY | HG | HRT | HYT | HGT | VR | VY | VG | VRT | VYT | VGT |
| S1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| S2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| S3 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| S4 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| S5 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| S6 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| S7 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| S8 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

**Table 2:** State table for the Night mode

| State | East/West | | | | | | North/South | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HR | HY | HG | HRT | HYT | HGT | VR | VY | VG | VRT | VYT | VGT |
| S1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| S2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |