Traffic Light Controller

Four Way Intersection Traffic Light System

Fall-2017

James Todd, Thierno Barry, Andrew Tamer, Gurashish Grewal
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: jdtodd@oakland.edu, thbarry@oakland.edu, tamer@oakland.edu, gsgrewal@oakland.edu

Abstract

The main purpose of building our traffic light control is to gain more understanding of the implementation of a modern digital system. We designed a four way traffic signal for a an intersection. We used VHDL to write the code that modeled the traffic system, and then used a finite state machine and accessory components to control the behavior of the system.

I. Introduction

For the final project we created a traffic light system utilizing with four different LED colors on the two tri-color LED's and a Seven Segment display both mounted on the Artix-7 board. This report covers our plan behind the design of our traffic light system, as well as how we completed and implemented the structure of the design. The idea behind this project was to create a traffic light with three different modes for a four-way intersection. The three different modes include a daytime mode, night mode, and emergency mode. Because the yellow light displayed by the tri-color LED wasn't aesthetically distinctive, we included a color changing switch that would change the yellow LED to a blue LED because of the contrast.

II. Methodology

A. Overview

The traffic light system is divided into four main sections, the top file, the clock divider, the tri-color LED file, and the Seven-Segment display file. In our simulation the traffic flowing east/west and north/south have identical signals and therefore only two tri-color LED's are needed to represent the intersection. The program includes three modes, the day mode, the night mode and the emergency mode. Daytime mode is the default mode. Switches will be used to control the other modes. The seven-segment display and two tri-color LEDs on FPGA will display the color of the signal depending on the state of the system, which is controlled by the FSM.

B. Top File

The top file is used only to connect the inputs of the device to the various components and then connect those components to the outputs of the board.

C. FSM

The FSM is the controller of the system. It uses the clock and switches as inputs and progresses through the state of the system as follows. The FSM operates in daytime mode normally. If emergency mode is activated and one of the lights is green, the systems will skip to yellow and progress to red as normal. Once both lights are red, the FSM will continue to display red lights in both directions until emergency mode is turned off. Emergency mode is supposed to represent a signal sent from an emergency vehicle such like an Ambulance or Fire Engine. This can be used so that an emergency vehicle can progress through the light with a lower chance of a traffic incident. Since there are only eight states in the FSM the outputs can be represented by three bits. These three bits are sent to both the LED file and SevSeg file. A detailed stated diagram is listed in the Appendix.

i) In daytime mode, the system will cycle through the states as shown in the figure below. The times indicated on the figure were chosen for display purposes. The time in each state could be easily edited by changing an integer value, which represents the number of seconds in each state.

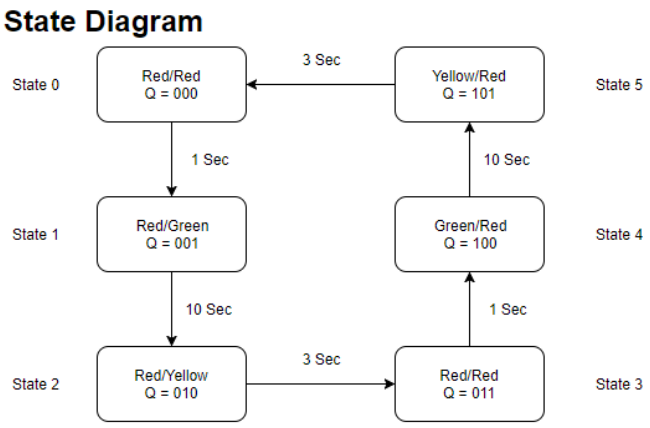


Figure 1: State diagram of the traffic signal.

ii) *Emergency Mode*- When the emergency switch is turned on the system will skip from green to yellow and progress normally to red/red. Once red/red is displayed by the FSM, the system will stay in that state until the Emergency mode switch is turned off. Since there are two states that represent red/red, the system has a small amount of memory and will fairly allow the next set of vehicles to progress through the intersection on their turn.

iii) *Night time Mode*- When the Night time mode switch is turned on the system will wait for a red/red signal, a yellow light will blink in the east/west direction, while a red light will blink in the north/south direction. These lights will alternate.

D. Tri-Color LED

The LED file receives a three-bit input unique to the present state controlled by the top file. Based upon the present state, the LEDs will display the appropriate color. One LED will represent the north/south signal, and the other LED will represent the east west signal. The LEDs change instantly depending on the state of the system which is controlled by the FSM. As the state changes, the color of the LEDs changes accordingly and will display the East/West signal on the left LED and the North/South signal on the right LED. Depending on which mode the program is in, the LEDS will turn on and off accordingly. For demonstration purposes, a blue LED acts as the yellow light of the traffic light. If the color change switch is flipped, the blue LED will change to yellow, and behave as it did before. *E. Seven Segment Display*

The seven-segment file receives input signal identical as the LED file, along with a slowed 1000 Hz clock signal produced by the clock divider. Instead of using the serializer provided, we decided to make our own display that uses two processes running at the same time. One controls the states used to move between the different displays on the seven-segment in order to display different letters on the Seven segment, whereas, the other process controls the signal outputting on the seven segment display. As the each state changes, the condition (S0-S6 from the Top file) is checked and as per the condition the process loops and displays the condition on Seven Segment accordingly. For example, if the condition is Red-Red, 000 will be selected and states will change at each clock input with a different anode and output displaying all the letters on different segments one at a time, but at such a rate that the human eye sees it as a single display.

F. Clock Divider

The clock divider uses the 100 MHz clock provided by the chip, and gives two outputs. Once the 100 MHz clock has given a high signal 100,000,000 times, the clock divider will output a "Z" signal for 10 ns, then will synchronously clear the counter and start the system back at 0. The FSM uses this as a 1 second signal to count the elapsed time in each state. The other signal is similar to the first, except it sends a 10kHz signal to the SevSeg file in to be used as the clock. The signal needed to be fast enough to cycle through the SevSeg displays and remain imperceptible to humans, yet if the signal was too fast, the SevSeg display wouldn't be able to display properly.

G. Alternate Approach

Instead of using the state of the system to control the outputs, the system could be controlled using the same 1 second counter and a second counter whose output would control the displays. This method would be harder to edit, as any change in timing would drastically change the circuit. A new design would be needed to change the length of time in any of the states. If the same counts were used a table and Karnaugh maps could be created. As an example, the following table and resulting Boolean equations could be created.

	A	B	C	D	E	North/South			East/West		
time (s)	q4	q3	q2	q1	q0	R	Y	G	R	Y	G
0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	1	0	0	1	1	0	0
2	0	0	0	1	0	0	0	1	1	0	0
3	0	0	0	1	1	0	0	1	1	0	0
4	0	0	1	0	0	0	0	1	1	0	0
5	0	0	1	0	1	0	0	1	1	0	0
6	0	0	1	1	0	0	0	1	1	0	0
7	0	0	1	1	1	0	0	1	1	0	0
8	0	1	0	0	0	0	0	1	1	0	0
9	0	1	0	0	1	0	0	1	1	0	0
10	0	1	0	1	0	0	0	1	1	0	0
11	0	1	0	1	1	0	1	0	1	0	0
12	0	1	1	0	0	0	1	0	1	0	0
13	0	1	1	0	1	0	1	0	1	0	0
14	0	1	1	1	0	1	0	0	1	0	0
15	0	1	1	1	1	1	0	0	0	0	1
16	1	0	0	0	0	1	0	0	0	0	1
17	1	0	0	0	1	1	0	0	0	0	1
18	1	0	0	1	0	1	0	0	0	0	1
19	1	0	0	1	1	1	0	0	0	0	1
20	1	0	1	0	0	1	0	0	0	0	1
21	1	0	1	0	1	1	0	0	0	0	1
22	1	0	1	1	0	1	0	0	0	0	1
23	1	0	1	1	1	1	0	0	0	0	1
24	1	1	0	0	0	1	0	0	0	0	1
25	1	1	0	0	1	1	0	0	0	1	0
26	1	1	0	1	0	1	0	0	0	1	0
27	1	1	0	1	1	1	0	0	0	1	0

Figure 2: Table created to map each second of a 28 second counter to the output of 6 LEDs.

From this table a Karnaugh map (K-map) could be created for each LED. A K-map and following Boolean equation was created to demonstrate the process of this approach.

ABC \ DE	00	01	11	10
000	1	0	0	0
001	0	0	0	0
011	0	0	1	1
010	0	0	0	0
100	1	1	1	1
101	1	1	1	1
111	x	x	x	x
110	1	1	1	1

Figure 3: Karnaugh map used to derive sum of products equation for the Red North/South LED

The following boolean equation can be derived using sum of products, and boolean algebra to reduce the equation to its simplest terms.

$$R1 = A + BCD + B'C'D'E'$$

This method was not chosen due to the lack of flexibility.

III. Experimental Setup

After completing the Top file with port mapping, a test bench was created with two inputs being passed onto the Top file. Our test bench was simple with "clock" and "resetn" being the two inputs. Xilinx Vivado version 2016.2 was used to simulate and implement the code. Testbench is a simple way to verify the working of the code but looking at the post timing simulation. The code used for the Testbench displayed below.

After completing the constraint file and writing the bitstream, the code was then uploaded to the NEXYS 4 DDR board. Our group did not use any outside hardware tools, except only the switches, seven segment display, LEDs and the buttons on the board. We expect the LEDs and seven segment display to behave according to which mode the traffic signal is operating. If the reset button is pressed the system should reset both LEDs to red, and the seven segment displays will say red-red.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_TrafficLight is

end tb_TrafficLight;

architecture Behavioral of tb_TrafficLight is

component TrafficLightTop
    Port ( clock : in STD_LOGIC;
           --LED : out STD_LOGIC_VECTOR (5 downto 0);
           --SevSeg : out STD_LOGIC_VECTOR (6 downto 0);
           resetn : in STD_LOGIC);
end component;

signal clock: std_logic;
signal resetn: std_logic;

begin

uut: TrafficLightTop port map(clock => clock, resetn => resetn);

f1: process
    begin
    wait for 10 ns;
    resetn <= '0'; wait for 20 ns;
    resetn <= '1';
    wait;
end process;

clk_begin: process
begin
    clock <= '0'; wait for 10 ns;
    clock <= '1'; wait for 10 ns;
end process;


end Behavioral;
```

IV. Results

After attempting the first approach using Karnaugh maps, the FSM approach proved to be the most logical and efficient way to control the outputs. The FSM code along with the other components of the project cycle through correctly, leading to the LED and seven segment display behaving as intended. Depending on which of the three switches are high, the traffic light system operates in the desired mode. The normal/day time mode works properly following the state diagram, the emergency mode works properly turning the lights red accordingly, and the night mode makes the lights blink red and yellow on each side respectively. Through different trials and testing, the traffic light does match our final intentions for the traffic light controller as intended during the time of design.

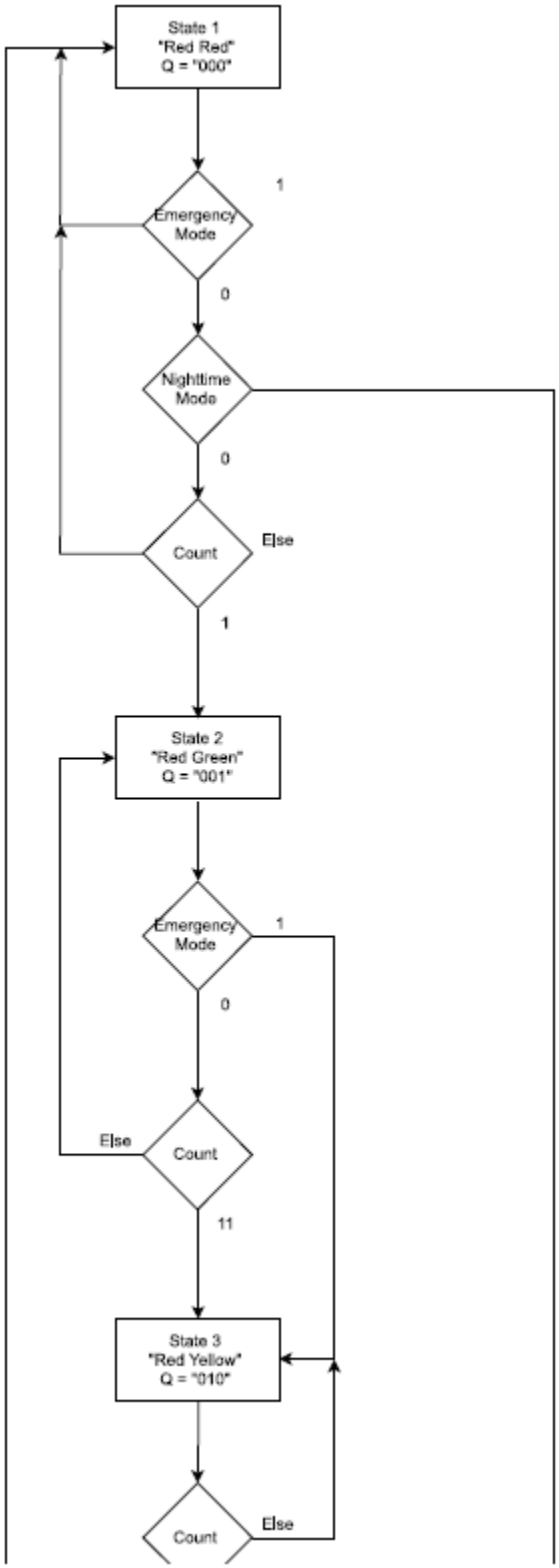
V. Conclusion

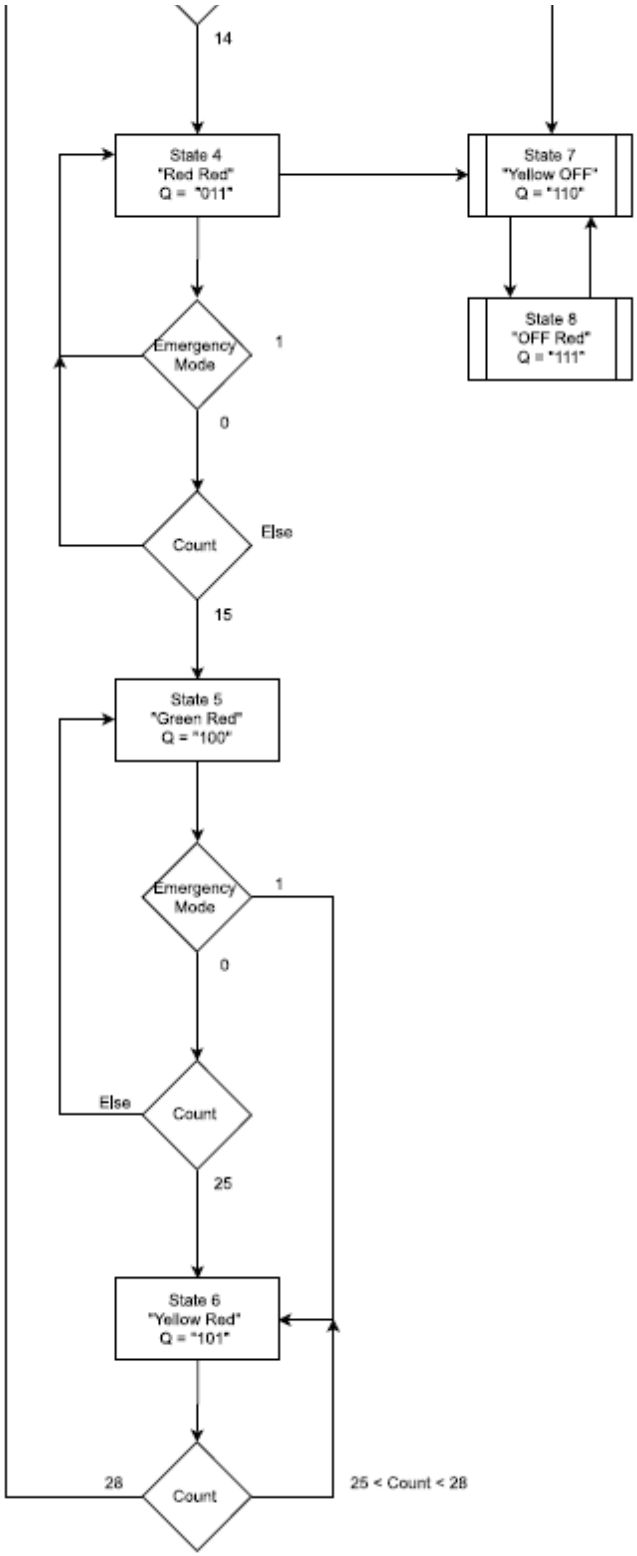
Through the use of a clock divider, a finite state machine, seven segment display, and LEDs, we were able to create a four way intersection traffic light system. At first, this project was very challenging as we tried to use K maps to come up with the equations that we were suppose to use. However, when we learned in class how to make state diagrams, the project became easier. Improvements that could be made to the design include adding sensors that could detect the number of cars that are on each line to be able to control the traffic better. Also, we could have use pulse width modulation to improve the color of the yellow light. When displayed, it looked too green.

References

1. Llamocca, Daniel . *VHDL Coding for FPGAs*, www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html

Appendix:


State 1
"Red Red"
Q = "000"
Emergency Mode
1
0
Nighttime Mode
0
Count
Else
1
State 2
"Red Green"
Q = "001"
Emergency Mode
1
0
Else
Count
11
State 3
"Red Yellow"
Q = "010"
Count
Else


14
State 4
"Red Red"
Q = "011"
State 7
"Yellow OFF"
Q = "110"
State 8
"OFF Red"
Q = "111"
Emergency Mode
1
0
Count
Else
15
State 5
"Green Red"
Q = "100"
Emergency Mode
1
0
Else
Count
25
State 6
"Yellow Red"
Q = "101"
28
Count
25 < Count < 28