Traffic Light Controller

Variable Cycle Time Design

Thomas Quinn, Brandon Londo, Alexander C. Vincent, Yezan Hussein

Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
tquinn@oakland.edu, vincent@oakland.edu, bjlondo@oakland.edu, yhussein@oakland.edu

***Abstract*— This project involves an FPGA based traffic light controller with user selected green light cycles. The state of the two lights outputs to two RGB LEDs, while the cycle times appear on the seven-segment display.**

I.) Introduction

The safety of all drivers on the road hinges upon a proper control of traffic flow. Allowing opposing directions of traffic, the right of way could result in fatal consequences. Our design simulates a traffic light controller that operates in both a safe and convenient fashion. It emulates the behavior of four traffic lights dictating automobile flow in opposing directions. The state of each light displays via the RGB LEDs on a Nexys4DDR board.

Sometimes, the flow of traffic in one direction is much heavier than in the opposing direction. It is appropriate to allow the heavier traffic to receive a green light for a longer time. To do this, the design features four cycle times which the user can select via switches. The current cycle time appears on the seven-segment display, allowing the user to confirm their selection.

II.) Methodology

All licensed drivers in the United States know the various states of a traffic light. The main task behind this project is manipulating the time between these state transitions. First, the team considered the size of the counters necessary for the design. The necessary counts are 1sec, 3sec, 15sec, 30sec, 45sec, and 60sec. One 15sec counter suffices to achieve the 15-60 second counts. This counter controls a Finite State Machine (FSM_Count) that outputs a signal at each of the 15 second increments. It does this by transitioning states each time the 15 second counter goes high. The FSM_Count outputs filter through two multiplexers, one for each direction of traffic (North/South and East/West).

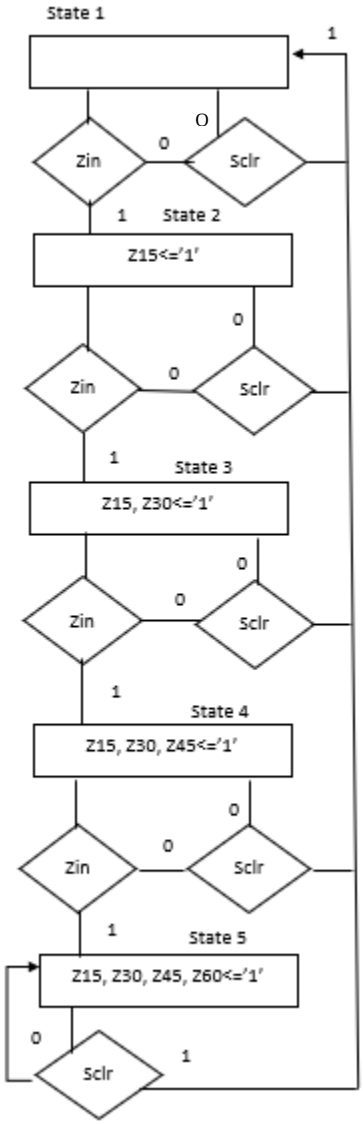

Figure 1: FSM Count

FSM1 is the hub of the program. It dictates the state of the lights and counters. Each state outputs the proper RGB code as well as enables and clears the necessary counters. The process in which this is done can be seen in the ASM below.

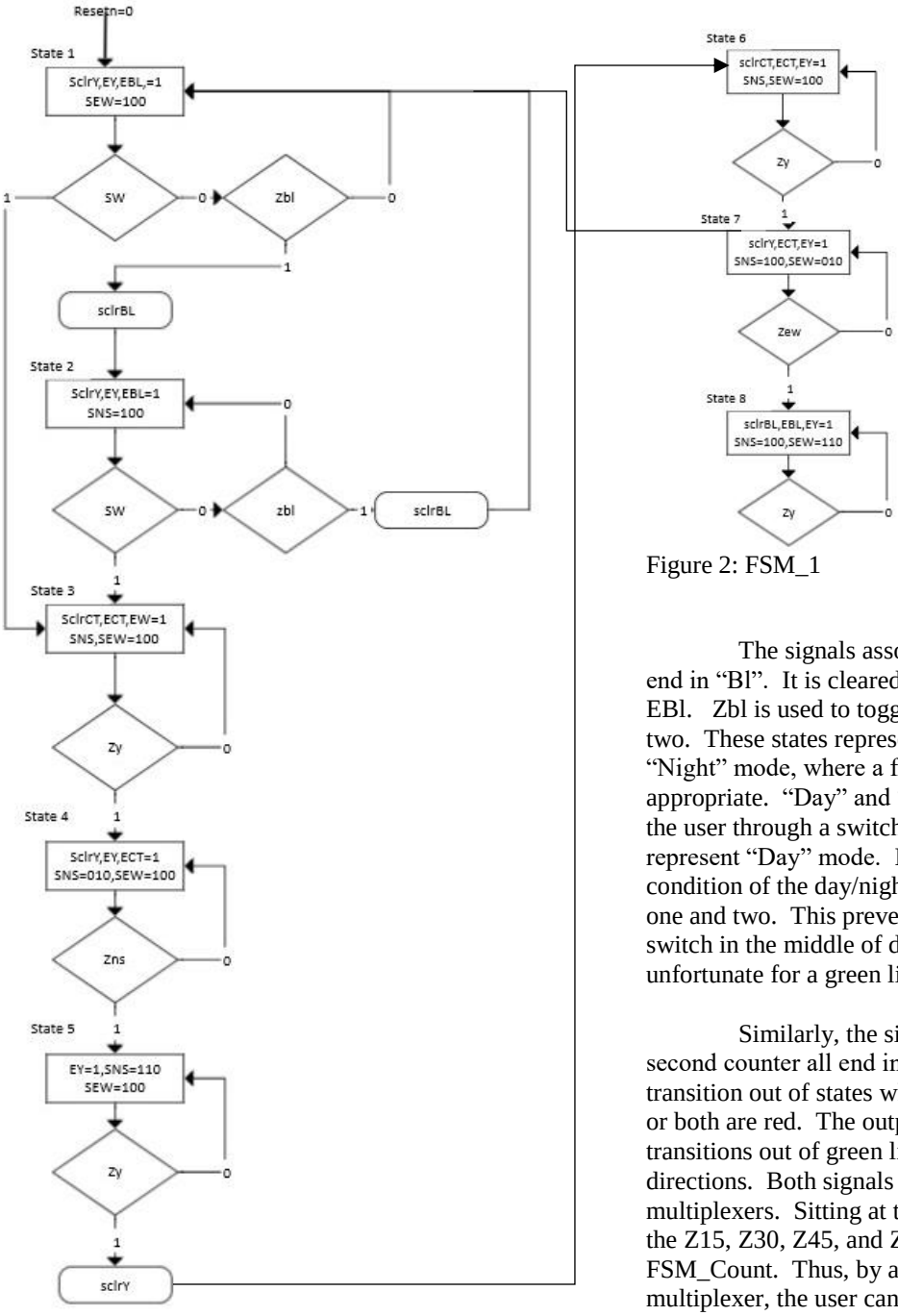

Figure 2: FSM_1

The signals associated with the 1 second counter end in "Bl". It is cleared with sclrBl and enabled with EBl. Zbl is used to toggle between states one and two. These states represent the blinking red lights of "Night" mode, where a four-way stop is appropriate. "Day" and "Night" modes are controlled by the user through a switch on the board. States 3-8 represent "Day" mode. It is important to note that the condition of the day/night switch is only checked in states one and two. This prevents a user from flipping the switch in the middle of day cycle operation, as it would be unfortunate for a green light to go directly to blinking red.

Similarly, the signals associated with the three second counter all end in "Y". The output Zy dictates the transition out of states where one of the lights is yellow, or both are red. The outputs ZEW and ZNS control the transitions out of green light states for their respective directions. Both signals are outputs from one of the two multiplexers. Sitting at the inputs of both multiplexers are the Z15, Z30, Z45, and Z60 signals coming from FSM_Count. Thus, by adjusting the select line of each multiplexer, the user can select which of these four signals will dictate the transition out of the respective green state.

The 7 segment displays output the green light cycle time for each direction. Since the cathodes of all the displays are tied together, only one pattern can appear at a time on any given display. Thus, to make different outputs seemingly appear on four displays at the same time, the anodes and outputs must be serialized. This involves turning each display individually at a rate which

appears constantly on to the human eye. Figure 3 pictures the FSM that achieves this operation.

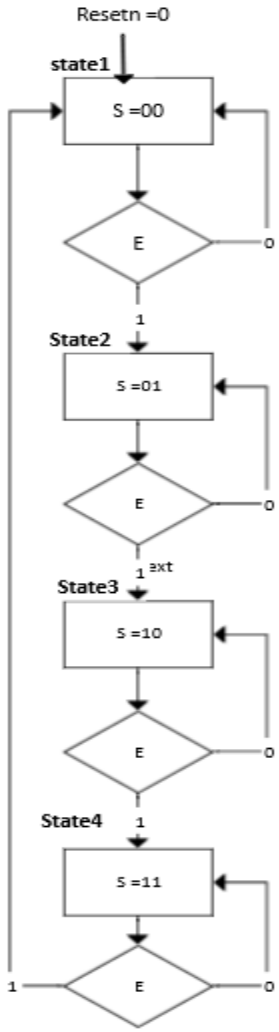

Figure 3: Serializer FSM

This FSM uses AdrNS and AdrEW as case statements to select the appropriate input to the BCD-7seg decoder. As shown below, the appropriate numbers for each display are hard coded to match the state of the select line switches.

```
with SW1 select
D <= "0001" when "00",
  "0011" when "01",
  "0100" when "10",
 "0110" when others;

with SW1 select
 C <= "0101" when "00",
   "0000" when "01",
   "0101" when "10",
  "0000" when others;

with SW2 select
  B <= "0001" when "00",
    "0011" when "01",
    "0100" when "10",
   "0110" when others;

 with SW2 select
   A <= "0101" when "00",
     "0000" when "01",
     "0101" when "10",
     "0000" when others;
```

By multiplexing the outputs and anodes of each display at a rate of 1ms, all four appear to be on simultaneously, With these operations the team was able to implement a functioning traffic light controller.

III.) Experimental Setup

The setup used to verify the functioning of the traffic light controller is as follows: Once the board (NEXYS 4 DDR) is programmed, the traffic light controller is tested by visual inspection. An external timepiece verifies that the displayed cycle times are accurate. For example if the user input was SW0=’1’, AdrEW=”10”, and AdrNS=”01” it is expected that seven segment display AN(0) is 5, AN(1) to be 4, AN(4) to be 0, and AN(5) to be 3. The cycle begins with both lights red, after three seconds the leftmost LED (led17) will transition to a green light. The green light will be on for 30 seconds. It then transitions yellow for 3 seconds. Then both LEDs are red for 3 seconds. Next the rightmost LED (led16) is green for 45 seconds. It then changes to yellow for three seconds. The cycle repeats while SW0=’1’. A test bench is also created to ensure each signal propagates when intended. The software used to implement this design is Xilinx Vivado 2017.2.

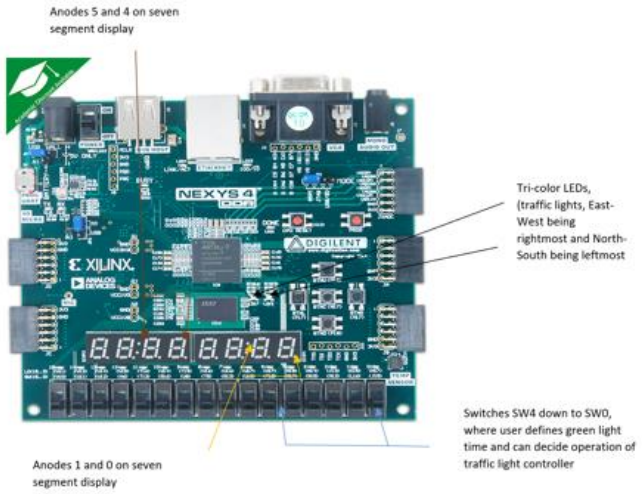

Figure 4: Nexus DDR Board

IV.) Results

The timepiece revealed an error in cycle times above 15 seconds. This error would propagate through the program the longer it ran. At first, the 30 second increment would take 37 seconds, the next time around, it would take 47. Stepping through a behavioral simulation lead to the solution. The FSM_Count simply needed to be synchronously cleared. After this addition, the cycle times worked perfectly.

Upon completion of the top-level code a top-level diagram was drawn.

V.) Conclusions

This program provides all the intended functions and features. The design process highlighted the usefulness of finite state machines in digital design. The switches and hex displays offer a nice level of user friendliness. Further versions of this project might include additional states for left-turn signals, and the ability to respond to crosswalk signals from pedestrians.

VI.) References

[1]. *Workshop: Digital Circuit Design with VHDL*. Web. 12 Nov. 2011. http://www.secs.oakland.edu/~llamocca/index.html

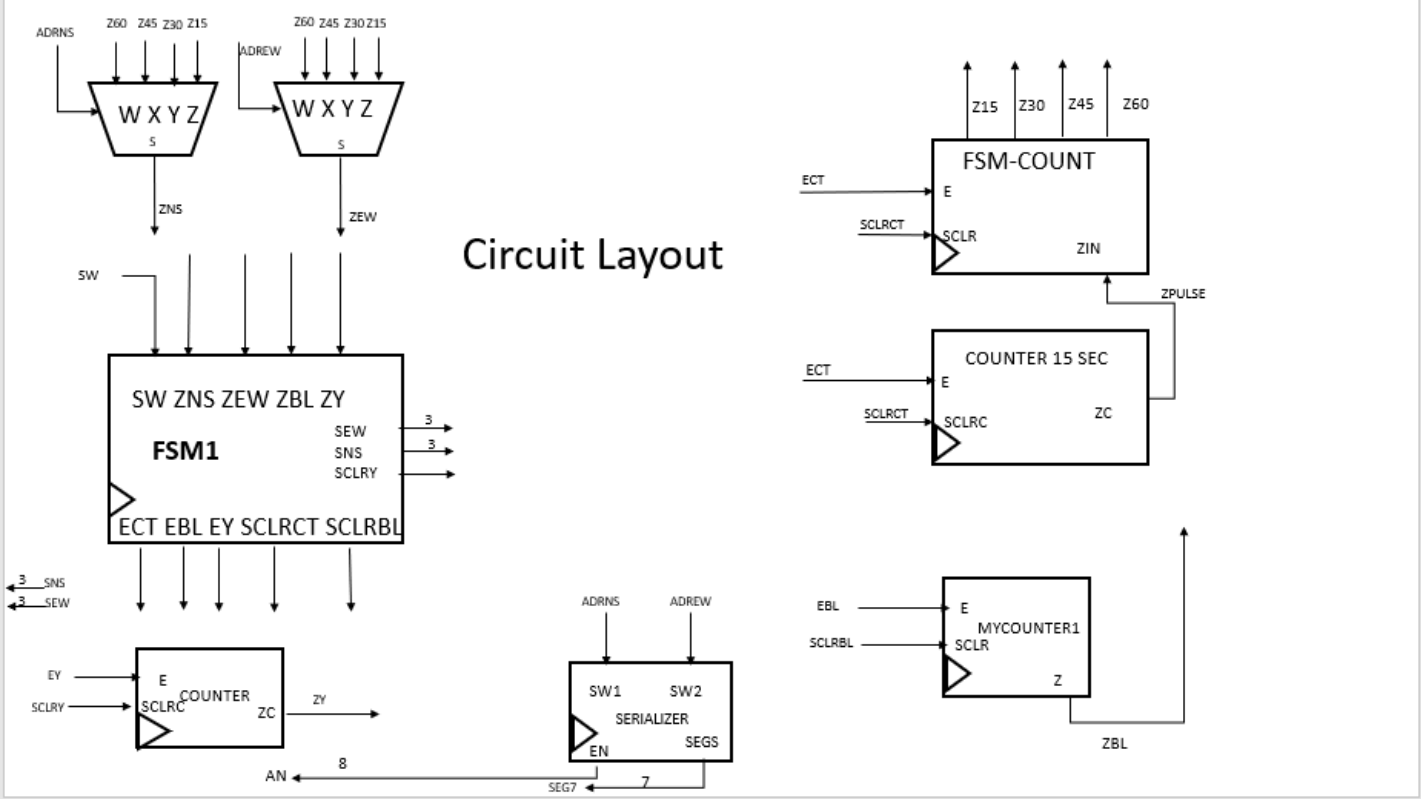

Figure 5: Top Level Design of Traffic Light Controller