4-Way Traffic Intersection Control

Traffic Light Simulation

Joseph Asteefan, John Drabik, Christopher Thweatt, Madison Cornett
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: jasteefan@oakland.edu, jdrabik@oakland.edu, cgthweatt@oakland.edu, mcornett@oakland.edu

Abstract **- The purpose of this project is to program an Artix-A7 FPGA board using VHDL code to function as a traffic controller simulation using an input counter values and output LED markers. This was achieved using a finite state machine. The final code was generated to the FPGA board and demonstrated using external LEDs on a separate bread board.**

I. INTRODUCTION

The goal for this project is to simulate a functioning 4-way Traffic Light Controller using VHDL and a field programmable gate array (FPGA), designed with digital logic tools, including state machines and synchronous circuits. Using FPGAs as field hardware would provide traffic engineers the ability to quickly prototype and implement traffic signal updates to improve road and construction conditions. The reprogrammable nature of FPGAs enables exploration of states, flexible counters, and accumulators that will be essential for this traffic control simulation to function.

II. METHODOLOGY

Vivado was used to program an Artix-A7 FPGA board. Buttons, Pmod ports and LEDs were utilized to show the different functions of the 4-way traffic intersection scenario. A 4-way traffic intersection control light is demoed within the project with the use of a state machine. Each state represents a different light sequence for each of the signals, and cycles through the clock tick. An example of one state would be S0 = green for the road going north/south and red for the east/west road. A variable “count” is created and is added based on the clock tick. The state will change depending on the amount of time passed, and if the “count” value has reached a high enough count. This state machine produces a six-bit segment. The first three bits in the segment represent the light pattern for the north/south road direction and the next three bits represent the light pattern for the east/west road direction. An example of this case would be if the six-bit segment is 100001 the resulting light pattern would be 001 (red) for the east/west and 100 (green) for the north/south.

A. State Machine

A state diagram is used to show how the traffic flows. Starting at S0, the North/South lights stay green for 10 seconds, then turn yellow for 3 seconds, then turn red for 2 seconds. Starting at S3, the same happens for the East/West lights as the North/South lights stay red. A state table is provided to determine how each of the different LEDs will be active and which state the device will be in during these times. For the two directions, East/West and North/South, the state machine determines which state the light should be in to show the correct color LED at that time. For example, if the North/South direction has a green light, the opposite road at that intersection would then have to be a red light, as shown in S0. Figure 2 shows each of the possible variations of states that the traffic light may be in and the finite state machine determines when to switch to each state.

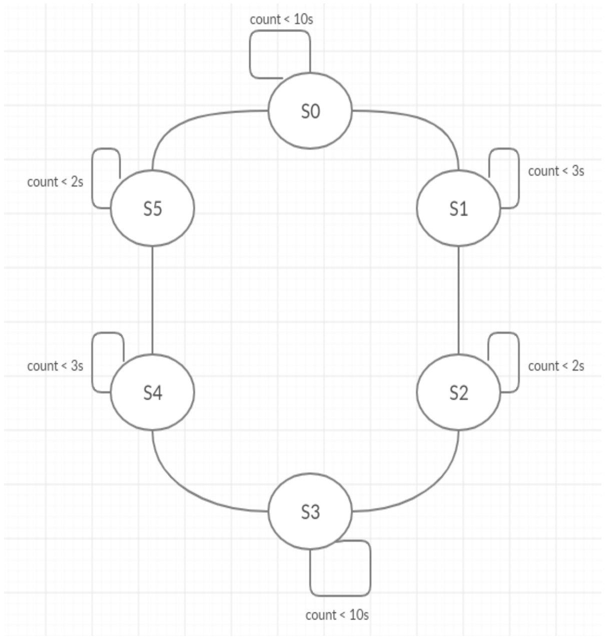
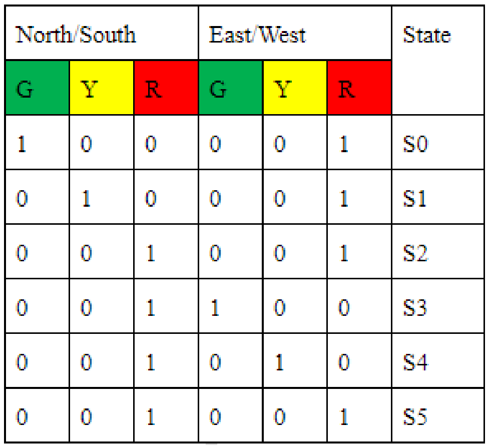


Figure 1: State Diagram



North/South			East/West			State
G	Y	R	G	Y	R	
1	0	0	0	0	1	S0
0	1	0	0	0	1	S1
0	0	1	0	0	1	S2
0	0	1	1	0	0	S3
0	0	1	0	1	0	S4
0	0	1	0	0	1	S5

Figure 2: State Table with Traffic Outputs

B. Counter

An integer counter is used to count the number of seconds. The Artix-A7 FPGA's built in clock runs very fast, but it was found that one second was equivalent to 100 million in integer. Since 30 seconds are needed for a traffic cycle, we needed a 3 billion integer range. However, the integer range for VHDL is between -2 and 2 billion roughly, so a -1.5 billion to 1.5 billion range is used. Starting at -1.5 billion, -1.4 billion is one second in, -1.3 billion is two seconds in, and so on. This allows the correct timing of the traffic intersection.

C. 7-Segment Serializer and Integer to BCD Decoder

A BCD (4-bit) decoder and serializer are used to add a timer, in decimal, that displays the length of traffic cycle and ensures the traffic cycle is 30 seconds. The BCD decoder is used to take the counters large integer values and convert them two 4-bit values, one for the tens digit (B) and the other for the ones digit (A). For example, if the count is less than -1.2 billion, which is equivalent to 3 seconds in, set A=0010 and B=0000. Suppose it is 15 seconds in, A=0101 and B=0001. These values are eventually used in the serializer.

The serializer is used to display both digits, the ones and tens spot, at the same time. The FPGA board can only display one 7-segment display at a time, so a serializer allows the two digits to take turns flashing every one millisecond, which gives the illusion that they're both on at the same time. The serializer is composed of a counter, finite state machine (FSM), 1-to-2 decoder, 2-to-1 multiplexer and BCD to 7-segment decoder [1].

The counter set the enable for the state machine high every millisecond. Setting the enable high switches the state in the FSM, as seen in Figure 4. At S0, the ones digit displays and at S1, the tens digit displays. The multiplexer and 1-to-2 decoder determine what number to display and on which display.

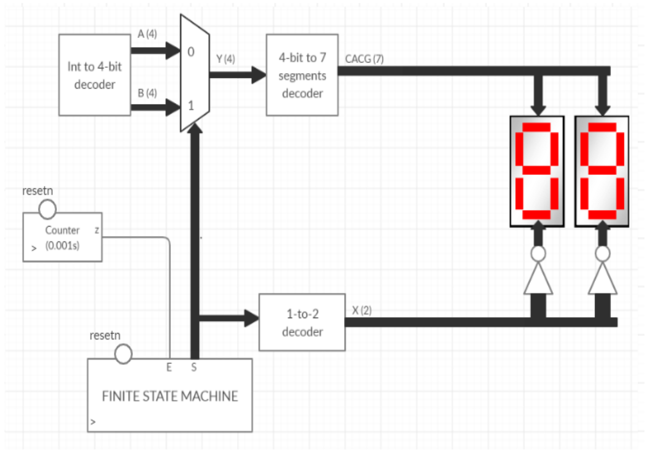


Figure 3: 7-Segment Serializer and Integer to BCD Decoder

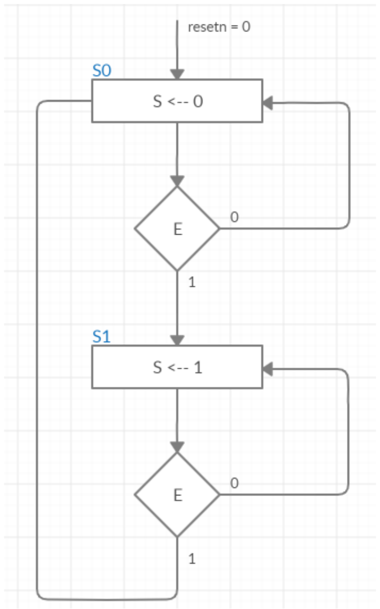


Figure 4: FSM in Serializer

D. Top File

The file consists of putting all these components together, as seen in Figure 5. The counter feeds its output to the traffic state machine and the integer to BCD decoder. The traffic state machine outputs the lights. The decoder feeds its output to the serializer which displays the timer.

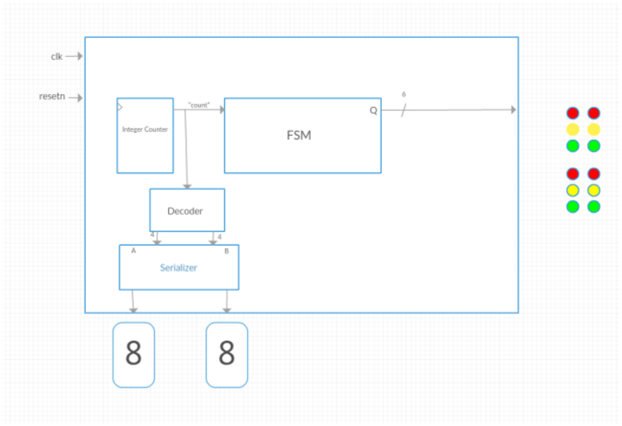
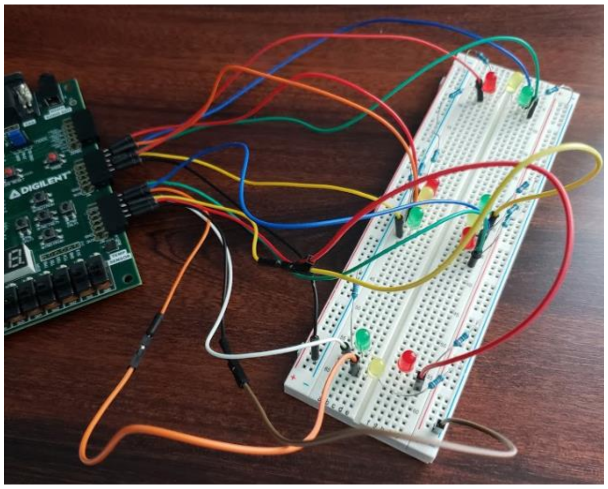


Figure 5. Top Level Design

III. EXPERIMENTAL SETUP

Vivado was used to program the Artix-A7 FPGA board. A reset button, built-in clock, breadboard, jumper wires external red, yellow, and green LEDs were utilized to show the 4-way traffic intersection. 220Ω resistors were used for the LEDs. The Pmod ports were used to output the six-bit light outputs on the breadboard.



IV. RESULTS

Once the top file was complete, and the bitstream was uploaded to the FPGA board, it was discovered that the count was too quick due to the built-in clock being fast. The count wasn't exactly 1 second like it was originally thought, and all the LEDs looked like they were all simultaneously on. To fix this issue, a larger integer value was created to reduce the rate of change and to allow for more time in between each cycle, and to show each of the different LEDs lighting at their own time. After fixing this, the LEDs displayed correctly, as well as the timer.

CONCLUSION

Utilizing a state machine structure and accumulative counter as a measured timer, the 4-way traffic light simulation was successfully adapted through the summation of varied VHDL code. The flexibility of FPGA experimentation in this situation allowed for learning and advancement of code through trial. Along with class taught architecture, the use of the Artix-7 Pmod ports were trialed to enable the use of external hardware that confirmed the function of the traffic control simulation. Future experimentation could include the implementation of FPGA hardware in the field, along with a user interface that would permit a technician's signal timing changes for appropriate road conditions.

REFERENCES

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