Traffic Light Controller

Matt Budzynowski, Joshua Donahue, Trent Smith

Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI

mbudzynowski@oakland.edu, donahue2@oakland.edu, tsmith11@oakland.edu

I. Introduction

The scope of this project is to design and code a simple traffic light controller. This project can be implemented to control traffic flow through a four-way intersection. We will model a North-South-East-West intersection as a way to label each light (R_{NS} = red light in the North/South direction, G_{EW} = green light in the East/West direction, etc.). The main components that will be employed are three counters of different duration, and a Mealy finite state machine (FSM). The system will incorporate two modes: daytime (normal) operation and nighttime (flashing) operation.

II. Methodology

The main driving factor of this traffic light controller is the FSM. The operation of the system is comprised of nine different states, and the FSM drives the change of these states in order to operate the traffic lights as desired. Daytime operation mode consists of the following states:

- S1: G_{NS}, R_{EW}
- S2: Y_{NS}, R_{EW}
- S3: R_{NS}, R_{EW}
- S4: R_{NS}, G_{EW}
- S5: R_{NS}, Y_{EW}
- S6: R_{NS}, R_{EW}

S1 will be active for 10 seconds, S2 will be active for 3 seconds, and S3 will be active for 1 second. Following S3, states S4, S5 and S6 will follow the same timing pattern, so long as the system remains in daytime operation mode.

Nighttime operation mode gives a flashing yellow light on the North/South path and flashing red light on the East/West path. Since we want to make these lights alternate, this mode will incorporate two additional states:

- S7: Y_{NS}
- S8: R_{EW}

Finally, a system enable will be incorporated, turning the system off completely, adding the last state:

- S0: 0

Three separate counters are used as inputs to the state machine in order to drive the transitions between states. The counter implementation is as follows:

Clock: a clock will be used to drive the function of all counters in the system, as well as the FSM. The period of one clock cycle is 10ns (100MHz Frequency).

Counters: the traffic control system is based on counters that emit a logic-high pulse when they reach a predetermined count, which starts the next state of the process. Each state is associated with a different length counter because of the different duration of each state. The one-second counter emits a pulse after 10^8 clock cycles, the three-second counter emits a pulse after $3*10^8$ clock cycles, and the ten-second counter emits a pulse after 10^9 clock cycles.

Each of these counters employs an enable signal and a synchronous clear signal as inputs, to drive the behavior of the counter. These enables and sclr signals are driven by the FSM based on the current state and the inputs to the FSM. For example: in S1, the ten-second counter (c10) is active and the other two counters (c1 and c3) are inactive. At the end of the ten seconds, c10 emits a logic-high pulse, at which time the FSM looks for the position of the mode switch to determine the next state. If the next state is S2, the FSM will disable and clear c10, by emitting a logic-low signal for E10 and a logic-high

signal for sclr10, and enable c3 via a logic-high signal for E3.

In addition to controlling the counter enable and synchronous clear signals, the FSM also incorporates a 12-bit output (s); each bit of this output signal drives the behavior of one of the active-high traffic lights.

The provided state diagram (Fig.1) has been developed to formally show the description and behavior of each state, as well as the transitions to the next state, based on the system inputs and current state. Each input and internal signal abbreviation is specified (Fig.2), and the state assignment and state table are also provided (Fig.3 & 4). VHDL code was written directly from this state diagram, and the circuit interconnections are shown in the circuit diagram (Fig. 5).

Fig. 1: State Diagram

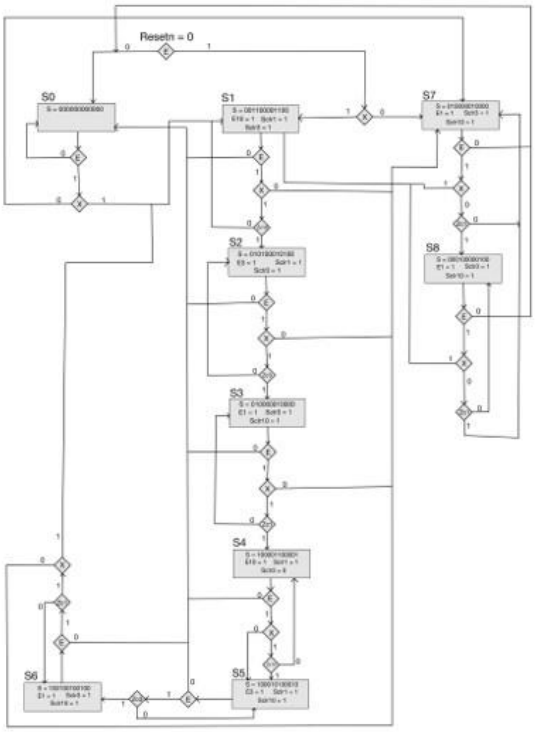


Fig. 2: Signal Abbreviations

Signal Abbreviations	
Signal Name	**Signal Assignment**
E	System Enable (0 = disable, 1 = enable)
X	Mode switch (0 = night, 1 = day)
zC1	1-sec counter "done" signal
zC3	3-sec counter "done" signal
zC10	10-sec counter "done" signal
E1	1-sec counter enable
E3	3-sec counter enable
E10	10-sec counter enable
sclr1	1-sec counter synchronous clear
sclr3	3-sec counter synchronous clear
sclr10	10-sec counter synchronous clear
**All "done", "enable" and "synchronous clear" signals are active-high	

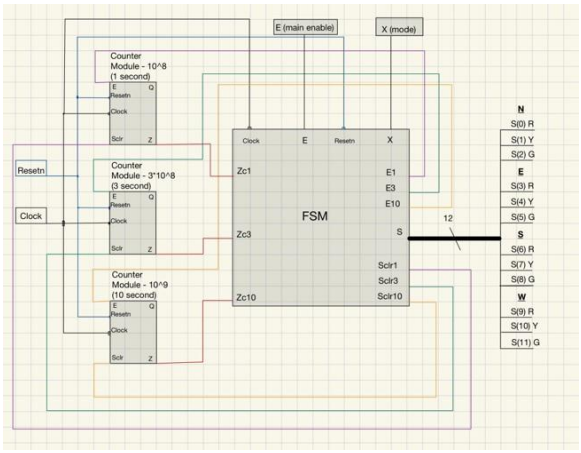
Fig. 3: State Assignments

State Assignments	
State	**Output (s = "$R_N Y_N G_N R_E Y_E G_E R_S Y_S G_S R_W Y_W G_W$")**
S0	"000000000000"
S1	"001100001100"
S2	"010100010100"
S3	"100100100100"
S4	"100001100001"
S5	"100010100010"
S6	"100100100100"
S7	"010000010000"
S8	"000100000100"

Fig. 4: State Table

State Table												
FSM Inputs (x = don't care)							FSM Outputs					
E	X	zC1	zC3	zC10	Current State	Next State	E1	E3	E10	sclr1	sclr3	sclr10
0	x	x	x	x	x	S0	0	0	0	0	0	0
1	0	x	x	x	S0	S7	1	0	0	0	1	1
1	0	x	x	x	S1	S7	1	0	0	0	1	1
1	0	x	x	x	S2	S7	1	0	0	0	1	1
1	0	x	x	x	S3	S7	1	0	0	0	1	1
1	0	x	x	x	S4	S5	0	1	0	1	0	1
1	x	x	0	x	S5	S5	0	1	0	1	0	1
1	x	0	x	x	S6	S6	1	0	0	0	1	1
1	0	0	x	x	S7	S7	1	0	0	0	1	1
1	0	0	x	x	S8	S8	1	0	0	0	1	1
1	x	x	1	x	S5	S6	1	0	0	0	1	1
1	0	1	x	x	S6	S7	1	0	0	0	1	1
1	0	1	x	x	S7	S8	1	0	0	0	1	1
1	0	1	x	x	S8	S7	1	0	0	0	1	1
1	1	x	x	x	S0	S1	0	0	1	1	1	0
1	1	x	x	0	S1	S1	0	0	1	1	1	0
1	1	x	0	x	S2	S2	0	1	0	1	0	1
1	1	0	x	x	S3	S3	1	0	0	0	1	1
1	1	x	x	0	S4	S4	0	0	1	1	1	0
1	1	x	x	x	S7	S1	0	0	1	1	1	0
1	1	x	x	x	S8	S1	0	0	1	1	1	0
1	1	x	x	1	S1	S2	0	1	0	1	0	1
1	1	x	1	x	S2	S3	1	0	0	0	1	1
1	1	1	x	x	S3	S4	0	0	1	1	1	0
1	1	x	x	1	S4	S5	0	1	0	1	0	1
1	1	1	x	x	S6	S1	0	0	1	1	1	0

Fig. 5: Circuit Diagram



III. Experimental Setup

Once the VHDL code was written, the bitstream was generated and then programmed onto an Artix-7 FPGA for experimental simulation. Since this circuit is relatively simple, no behavioral simulation was performed in Vivado. Instead, the board was programmed and then debugged through actual system operation. The only debugging step that was necessary was altering the code in the counters so that the synchronous clear signal reset the counters while the counter enable was "0". This ensured that each state lasts the intended duration, even with a mode change in the middle of a state.

Switch 1 and Switch 0 are assigned to the mode and system enable signals, respectively, and the CPU reset button will be assigned to the "resetn" signal. The "resetn" signal resets the FSM and all counters immediately when pressed (it is not a synchronous signal). LED's 13 down to 2 are assigned to R_N, Y_N, G_N, R_E, Y_E, G_E, R_S, Y_S, G_S, R_W, Y_W and G_W, in that order. With the board programmed in this manner, the traffic light controller operates exactly as planned.

IV. Results

We were able to employ the generic counter, as well as the principles of the finite state machine, to implement a properly functioning traffic light control system. See below link for a video of the system operating as desired, according to the system requirements and schematics.

https://youtube.com/shorts/qX00K5jK7VI

V. Conclusions

The Results of this project showed that the traffic light controller worked effectively and as planned. Challenges arose in the mapping of the objectives of safety in the real world to the state machine as different transitions. However, the final results ended up being almost the same as the traffic lights that are used on roads today. A challenge that would still need to be addressed is implementation of a large enough light system to control an intersection. The project was mapped to a Nexys board which allows the system's efficacy to show but is not practical for real world implementation.

VI. References

[1] Llamocca, Daniel. "VHDL Coding for FPGAs." *Reconfigurable Computing Research Laboratory*, Oakland University Electrical and Computer Engineering Department, www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html. Accessed 26 Nov. 2024.