

# **ECEN 248 - Lab Report**

**Lab Number: 11**

**Lab Title: The Traffic Light Controller Lab**

**Section Number: 513**

**Student's Name: Abhishek Bhattacharyya**

**Student's UIN: 731002289**

**Date: 4/25/2023**

**TA: Hesam Mazaheri**

## **Objectives:**

The main objective of this lab is to create an FSM to control the traffic lights at a highway intersection. The FSM was created in the prelab, and during the lab, a controller for the traffic lights is implemented using this FSM. The second part of this lab explores using a sensor input to control the FSM's outputs.

## **Design:**

To complete the design for this lab, first, the code for the FSM was written during the prelab. Then, the code for the traffic light controller was written using the FSM code, synchronizer module, and a counter built into the TLC controller code. All code for part 1 is included below.

For part two, the code was modified to include the farm sensor and modified FSM. All code for part 2 is included below as well.

```

1 `timescale 1ns/1ps
2 `default_nettype none
3
4 `define one_sec 50000000
5 `define three_sec 150000000
6 `define fifteen_sec 750000000
7 `define thirty_sec 1500000000
8
9 module tlc_fsm(state, RstCount, highwaySignal, farmSignal, Count, Clk, Rst);
10     output reg[2:0] state; // output for debugging
11     output reg RstCount; // use an always block
12     // another always block for these as well
13     output reg[1:0] highwaySignal, farmSignal;
14     input wire [30:0] Count; // use n computed earlier
15     input wire Clk, Rst; // clock and reset
16
17     parameter S0 = 3'b000,
18           S1 = 3'b001,
19           S2 = 3'b010,
20           S3 = 3'b011,
21           S4 = 3'b100,
22           S5 = 3'b101,
23           green = 2'b11,
24           yellow = 2'b10,
25           red = 2'b01;
26
27     reg[2:0] nextState;
28
29     always@(*)
30     case(state)
31     S0: begin
32         highwaySignal = red;
33         farmSignal = red;
34         if (Count >= `one_sec)
35             begin
36                 RstCount = 1;
37                 nextState = S1;
38             end
39         else
40             begin
41                 RstCount = 0;
42                 nextState = S0;
43             end
44         end
45     S1: begin
46         highwaySignal = green;
47         farmSignal = red;
48         if (Count >= `thirty_sec)
49             begin
50                 RstCount = 1;
51                 nextState = S2;
52             end
53         else
54             begin
55                 RstCount = 0;
56                 nextState = S1;
57             end
58         end
59     S2: begin
60         highwaySignal = yellow;
61         farmSignal = red;
62         if (Count >= `three_sec)
63             begin
64                 RstCount = 1;
65                 nextState = S3;
66             end
67         else
68             begin
69                 RstCount = 0;
70                 nextState = S2;
71             end
72         end
73     S3: begin
74         highwaySignal = red;
75         farmSignal = red;
76         if (Count >= `one_sec)
77             begin
78                 RstCount = 1;
79                 nextState = S4;
80             end
81         else
82             begin
83                 RstCount = 0;
84                 nextState = S3;
85             end
86         end
87     S4: begin
88         highwaySignal = red;
89         farmSignal = green;
90         if (Count >= `fifteen_sec)
91             begin
92                 RstCount = 1;
93                 nextState = S5;
94             end
95         else
96             begin
97                 RstCount = 0;
98                 nextState = S4;
99             end
100        end
101    S5: begin
102        highwaySignal = red;
103        farmSignal = yellow;
104        if (Count >= `three_sec)
105            begin
106                RstCount = 1;
107                nextState = S0;
108            end
109        else
110            begin
111                RstCount = 0;
112                nextState = S5;
113            end
114        end
115    endcase
116
117    // update loop
118    always@(posedge Clk)
119        if(Rst)
120            state <= S0;
121        else
122            state <= nextState;
123
124    endmodule
125

```

Figure 1: Source Code for the traffic light FSM

```

1 `timescale 1ns/1ps
2 `default_nettype none
3
4 `define one_sec 50000000
5 `define three_sec 150000000
6 `define fifteen_sec 750000000
7 `define thirty_sec 1500000000
8
9 module tlc_fsm(state, RstCount, highwaySignal, farmSignal, Count, Clk, Rst, farmSensor, farmSync);
10     output reg[2:0] state; // output for debugging
11     output reg RstCount; // use an always block
12     // another always block for these as well
13     output reg[1:0] highwaySignal, farmSignal;
14     input wire [30:0] Count; // use n computed earlier
15     input wire Clk, Rst, farmSensor, farmSync; // clock and reset
16
17     parameter S0 = 3'b000,
18           S1 = 3'b001,
19           S2 = 3'b010,
20           S3 = 3'b011,
21           S4 = 3'b100,
22           S5 = 3'b101,
23           S6 = 3'b110,
24           green = 2'b11,
25           yellow = 2'b10,
26           red = 2'b01;
27
28     reg[2:0] nextState;
29
30     always@(*)
31     case(state)
32     S0: begin
33         highwaySignal = red;
34         farmSignal = red;
35         if (Count >= `one_sec)
36             begin
37                 RstCount = 1;
38                 nextState = S1;
39             end
40         else if (farmSensor == 1)
41             begin
42                 RstCount = 1;
43                 nextState = S4;
44             end
45         else
46             begin
47                 RstCount = 0;
48                 nextState = S0;
49             end
50         end
51     S1: begin
52         highwaySignal = green;
53         farmSignal = red;
54         if (farmSensor == 1 && Count >= `thirty_sec)
55             begin
56                 RstCount = 1;
57                 nextState = S2;
58             end
59         else
60             begin
61                 RstCount = 0;
62                 nextState = S1;
63             end
64         end
65     S2: begin
66         highwaySignal = yellow;
67         farmSignal = red;
68         if (Count >= `three_sec)
69             begin
70                 RstCount = 1;
71                 nextState = S3;
72             end
73         else
74             begin
75                 RstCount = 0;
76                 nextState = S2;
77             end
78         end
79     S3: begin
80         highwaySignal = red;
81         farmSignal = red;
82         if (Count >= `one_sec)
83             begin
84                 RstCount = 1;
85                 nextState = S4;
86             end
87         else
88             begin
89                 RstCount = 0;
90                 nextState = S3;
91             end
92         end
93     S4: begin
94         highwaySignal = red;
95         farmSignal = green;
96         if (Count >= (`fifteen_sec + `three_sec) || (farmSensor == 0 && Count >= `three_sec))
97             begin
98                 RstCount = 1;
99                 nextState = S5;
100             end
101         else
102             begin
103                 RstCount = 0;
104                 nextState = S4;
105             end
106         end
107     S5: begin
108         highwaySignal = red;
109         farmSignal = yellow;
110         if (Count >= `three_sec && farmSensor == 1)
111             begin
112                 RstCount = 1;
113                 nextState = S6;
114             end
115         else if (Count >= `three_sec)
116             begin
117                 RstCount = 1;
118                 nextState = S0;
119             end
120         else
121             begin
122                 RstCount = 0;
123                 nextState = S5;
124             end
125         end
126     S6: begin
127         highwaySignal = red;
128         farmSignal = red;
129         if (Count >= `one_sec)
130             begin
131                 RstCount = 1;
132                 nextState = S1;
133             end
134         else
135             begin
136                 RstCount = 0;
137                 nextState = S6;
138             end
139         end
140     endcase
141
142    // update loop
143    always@(posedge Clk)
144        if(Rst)
145            state <= S0;
146        else
147            state <= nextState;
148
149    endmodule
150

```

Figure 3: Source Code for the traffic light FSM with farm sensor

```

1 `default_nettype none
2
3 // This module describes the top level traffic light controller module discussed in lab 11
4
5 module tlc_controller(
6     output wire [1:0] highwaySignal, farmSignal, //connected to LEDs
7     output wire [3:0] JB,
8     input wire Clk,
9
10    // the buttons provide input to our top-level circuit
11    input wire Rst, //use as reset
12    //input wire farmSensor
13);
14
15 // intermediate nets
16 wire RstSync;
17 wire RstCount;
18 reg [30:0] Count;
19
20 assign JB[3] = RstCount;
21
22 // synchronize button inputs
23 synchronizer syncRst(RstSync, Rst, Clk);
24 //synchronizer syncFS(farmSync, farmSensor, Clk);
25
26 // instantiate FSM
27
28 tlc_fsm FSM(
29     .state(JB[2:0]), //wire state up to J for debug
30     .RstCount(RstCount),
31     .highwaySignal(highwaySignal[1:0]),
32     .farmSignal(farmSignal[1:0]),
33     .Count{Count[30:0]},
34     .Clk(Clk),
35     .Rst(RstSync),
36     //farmSensor(farmSensor)
37 );
38
39 // counter w/ synchronous reset
40 always@(posedge Clk)
41     if(RstCount)
42         Count <= 0;
43     else
44         Count <= Count + 1;
45
46 endmodule
47

```

Figure 2: Source Code for the traffic light controller (v1)

```

1 `default_nettype none
2
3 // This module describes the top level traffic light controller module discussed in lab 11
4
5 module tlc_controller(
6     output wire [1:0] highwaySignal, farmSignal, //connected to LEDs
7     output wire [3:0] JB,
8     input wire Clk,
9
10    // the buttons provide input to our top-level circuit
11    input wire Rst, //use as reset
12    input wire farmSensor
13);
14
15 // intermediate nets
16 wire RstSync;
17 wire RstCount;
18 wire farmSync;
19 reg [30:0] Count;
20
21 assign JB[3] = RstCount;
22
23 // synchronize button inputs
24 synchronizer syncRst(RstSync, Rst, Clk);
25 synchronizer syncFS(farmSync, farmSensor, Clk);
26
27 // instantiate FSM
28
29 tlc_fsm FSM(
30     .state(JB[2:0]), //wire state up to J for debug
31     .RstCount(RstCount),
32     .highwaySignal(highwaySignal[1:0]),
33     .farmSignal(farmSignal[1:0]),
34     .Count{Count[30:0]},
35     .Clk(Clk),
36     .Rst(RstSync),
37     .farmSensor(farmSensor),
38     .farmSync(farmSync)
39 );
40
41 // counter w/ synchronous reset
42 always@(posedge Clk)
43     if(RstCount)
44         Count <= 0;
45     else
46         Count <= Count + 1;
47
48 endmodule
49

```

Figure 4: Source Code for the traffic light controller with farm sensor

## Results:

All results are included below in the **Lab 11 Demo** section.

## Lab 11 Demo

Below is a demo of the successfully completed part 1 circuit on the Zybo board.

<https://drive.google.com/file/d/1gsYRvBqwxibZcgCvK1LRSfOWhQswKixZ/view?usp=sharing>

As can be seen in the video, after the reset button is no longer pressed, the FSM successfully starts at initial state S0 (red, red), waits for 1 second, steps to S1 (green, red), waits for 30 seconds, steps to S2 (yellow, red), waits for 3 seconds, steps to S3 (red, red), waits for 1 second, steps to S4 (red, green), waits 15 seconds, steps to S5 (red, yellow), waits 3 seconds, and finally steps to S0 to restart the loop.

It can also be seen when I hit the reset button mid loop, the circuit returns to S0 and resets as expected.

Below is a demo of the successfully completed part 2 circuit on the Zybo board.

[https://drive.google.com/file/d/1aM2XqB\\_tARw\\_03Oj-S0Tjj9dUdnC6r3l/view?usp=share\\_link](https://drive.google.com/file/d/1aM2XqB_tARw_03Oj-S0Tjj9dUdnC6r3l/view?usp=share_link)

As can be seen in the video, the farmSensor button is working exactly as intended. When the farmSensor is pressed and released while the FSM is in S0, the steps directly to S4 (red, green) and stays in S4 for a total of three seconds before stepping forward to S5 (red, yellow). If the farmSensor button is pressed and held, and then released after the FSM has been in S4 for more than 3 seconds, the FSM immediately steps to S5 (red, yellow). If the farmSensor button is pressed and held for a long time, the FSM stays in S4 for up to a total of 18 seconds (3 seconds minimum + 15 additional seconds max time, as defined in the lab doc) before stepping to S5 (red, yellow) for 3 seconds, S0 (red, red) for 1 second, and then stepping to S1 (green, red) and staying in this state for at least 30 seconds. The FSM will stay in this state indefinitely until the farmSensor is pressed; when it is pressed, the FSM will step to S2 (yellow, red), S3 (red, red), and then S4 (red, green) to let farm road traffic through. If the farmSensor button is held permanently (2-3 minutes or longer), the FSM will first initialize at S0, step to S4 (red, green), and stay in S4 for 18 seconds. Then the FSM will step to S5, S6 (S6 is a duplicate of S0 which CANNOT step to S4 to avoid the FSM getting stuck with the farmroad light green), and then to S1, where the highway light will glow green for 15 seconds. After 15 seconds in S1, the FSM will step through S2, S3, and then S4 to light the farm light green for 18 seconds, and the cycle repeats.

## **Conclusion:**

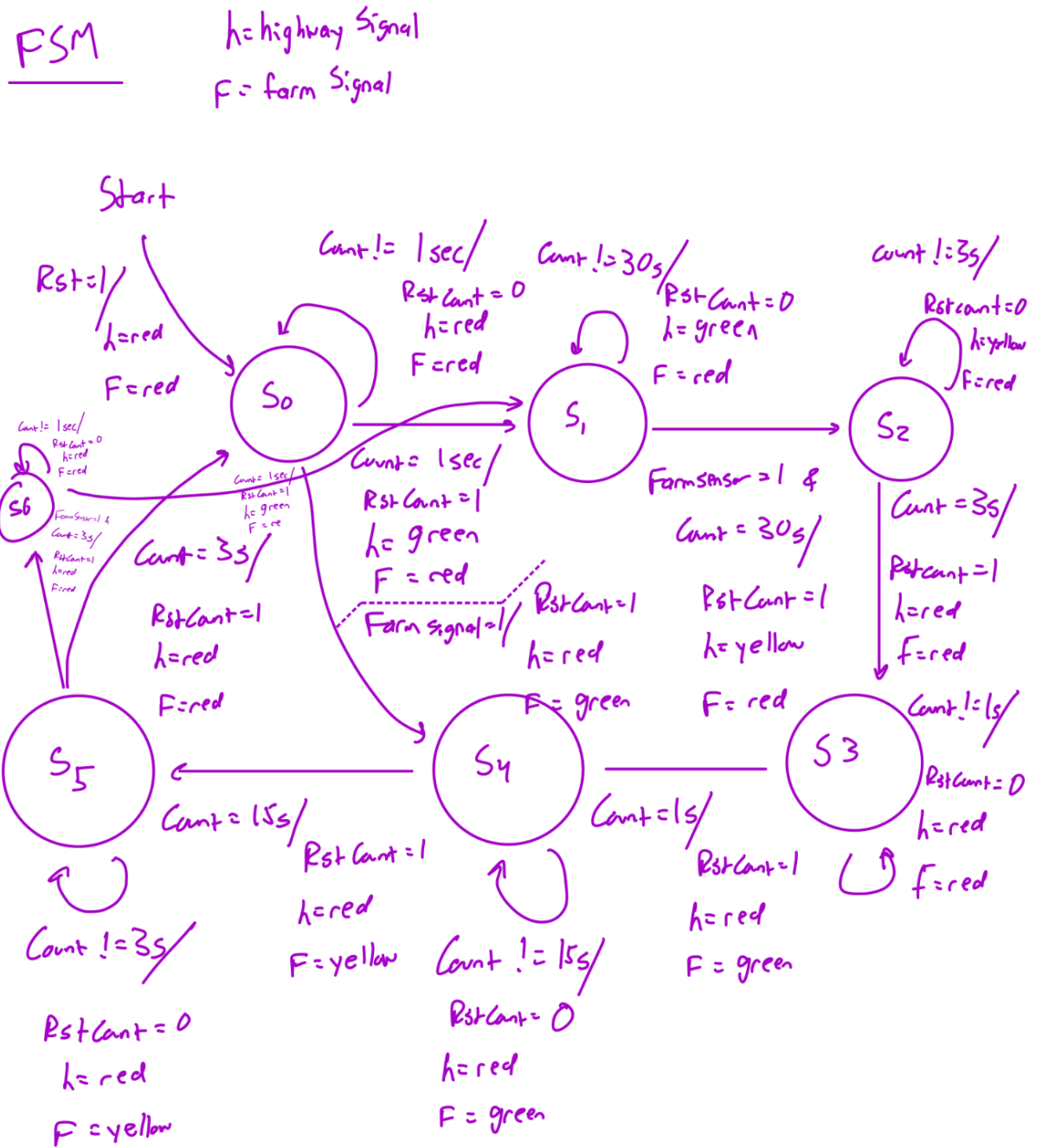
To complete this lab, I was able to use my knowledge of how to run and set up Vivado to code a traffic light FSM and controller code. Once this step was completed, the code was loaded onto the Zybo board and verified for functionality, and then a video taken for the lab demo. Next, part 2 was completed with the farm sensor, implemented on the Zybo board, and a video taken of the functioning circuit for the lab demo. Overall, this lab was a great way to end the semester, wrapping up our learning with an excellent and informative project.

## **Post-Lab Deliverables:**

1. Include the source code with comments for all modules that you wrote or modified in lab. You do not need to include code that was provided! Remember that code without comments will not be accepted!

**All code included in the results section.**

2. Include the state diagram for the modified traffic light controller FSM.



# Important Student Feedback

1. What did you like most about the lab assignment and why? What did you like least about it and why?

What I liked most about this lab assignment was that I was able to keep practicing with Verilog, as I appreciate gaining more much-needed experience. It was also very fulfilling seeing my Zybo board light up and work correctly after getting the code working. What I most disliked was just the general tedium of having to troubleshoot my code.

2. Were there any sections of the lab manual that were unclear? If so, what was unclear? Do you have any suggestions for improving the clarity?

The lab manual was clear and concise to understand.

3. What suggestions do you have to improve the overall lab assignment?

I do not have any suggestions to improve the lab assignment.