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 past 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.

```
`timescale 1ns/1ps
`default_nettype none

`define one_sec 50000000
`define three_sec 150000000
`define fifteen_sec 750000000
`define thirty_sec 1500000000

module tlc_fsm(state, RstCount, highwaySignal,farmSignal, Count, Clk, Rst);
    output reg[2:0] state;      // output for debugging
    output reg RstCount;        // use an always block
    // another always block for these as well
    output reg[1:0] highwaySignal, farmSignal;
    input wire [30:0] Count;     // use n computed earlier
    input wire Clk, Rst;         // clock and reset

parameter S0 = 3'b000,
          S1 = 3'b001,
          S2 = 3'b010,
          S3 = 3'b011,
          S4 = 3'b100,
          S5 = 3'b101,
          green = 2'b11,
          yellow = 2'b10,
          red = 2'b01;

 reg[2:0] nextState;

 always@(*)
    case(state)
        S0: begin
            highwaySignal = red;
            farmSignal = red;
            if (Count >= `one_sec)
                begin
                RstCount = 1;
                nextState = S1;
                end
            else
                begin
                RstCount = 0;
                nextState = S0;
                end
            end
        S1: begin
            highwaySignal = green;
            farmSignal = red;
            if (Count >= `thirty_sec)
                begin
                RstCount = 1;
                nextState = S2;
                end
            else
                begin
                RstCount = 0;
                nextState = S1;
                end
            end
        S2: begin
            highwaySignal = yellow;
            farmSignal = red;

            if (Count>= `three_sec)
                begin
                RstCount = 1;
                nextState = S3;
                end
            else
                begin
                RstCount = 0;
                nextState = S2;
                end
            end
        S3: begin
            highwaySignal = red;
            farmSignal = red;
            if(Count >= `one_sec)
                begin
                RstCount = 1;
                nextState = S4;
                end
            else
                begin
                RstCount = 0;
                nextState = S3;
                end
            end
        S4: begin
            highwaySignal = red;
            farmSignal = green;
            if (Count >= `fifteen_sec)
                begin
                RstCount = 1;
                nextState = S5;
                end
            else
                begin
                RstCount = 0;
                nextState = S4;
                end
            end
        S5: begin
            highwaySignal = red;
            farmSignal = yellow;
            if(Count >= `three_sec)
                begin
                RstCount = 1;
                nextState = S0;
                end
            else
                begin
                RstCount = 0;
                nextState = S5;
                end
            end
        endcase

 // update loop
 always@(posedge Clk)
    if(Rst)
        state <= S0;
    else
        state <= nextState;

 endmodule
```

Figure 1: Source Code for the traffic light FSM

```
`timescale 1ns/1ps
`default_nettype none

`define one_sec 50000000
`define three_sec 150000000
`define fifteen_sec 750000000
`define thirty_sec 1500000000

module tlc_fsm(state, RstCount, highwaySignal,farmSignal, Count, Clk, Rst, farmSensor, farmSync);
    output reg[2:0] state;      // output for debugging
    output reg RstCount;        // use an always block
    // another always block for these as well
    output reg[1:0] highwaySignal, farmSignal;
    input wire [30:0] Count;     // use n computed earlier
    input wire Clk, Rst, farmSensor, farmSync;         // clock and reset

parameter S0 = 3'b000,
          S1 = 3'b001,
          S2 = 3'b010,
          S3 = 3'b011,
          S4 = 3'b100,
          S5 = 3'b101,
          S6 = 3'b110,
          green = 2'b11,
          yellow = 2'b10,
          red = 2'b01;

 reg[2:0] nextState;

 always@(*)
    case(state)
        S0: begin
            highwaySignal = red;
            farmSignal = red;
            if (Count >= `one_sec)
                begin
                RstCount = 1;
                nextState = S1;
                end
            else if (farmSensor == 1)
                begin
                RstCount = 1;
                nextState = S4;
                end
            else
                begin
                RstCount = 0;
                nextState = S0;
                end
            end
        S1: begin
            highwaySignal = green;
            farmSignal = red;
            if (farmSensor == 1 && Count >= `thirty_sec)
                begin
                RstCount = 1;
                nextState = S2;
                end
            else
                begin
                RstCount = 0;
                nextState = S1;
                end
            end
        S2: begin
            highwaySignal = yellow;
            farmSignal = red;

            if (Count>= `three_sec)
                begin
                RstCount = 1;
                nextState = S3;
                end
            else
                begin
                RstCount = 0;
                nextState = S2;
                end
            end
        S3: begin
            highwaySignal = red;
            farmSignal = red;
            if(Count >= `one_sec)
                begin
                RstCount = 1;
                nextState = S4;
                end
            else
                begin
                RstCount = 0;
                nextState = S3;
                end
            end
        S4: begin
            highwaySignal = red;
            farmSignal = green;
            if (Count >= (`fifteen_sec + `three_sec) || (farmSensor == 0 && Count >= `three_sec))
                begin
                RstCount = 1;
                nextState = S5;
                end
            else
                begin
                RstCount = 0;
                nextState = S4;
                end
            end
        S5: begin
            highwaySignal = red;
            farmSignal = yellow;
            if(Count >= `three_sec && farmSensor == 1)
                begin
                RstCount = 1;
                nextState = S6;
                end
            else if(Count >= `three_sec)
                begin
                RstCount = 1;
                nextState = S0;
                end
            else
                begin
                RstCount = 0;
                nextState = S5;
                end
            end
         S6: begin
            highwaySignal = red;
            farmSignal = red;
            if (Count >= `one_sec)
                begin
                RstCount = 1;
                nextState = S1;
                end
            else
                begin
                RstCount = 0;
                nextState = S6;
                end
            end
        endcase

 // update loop
 always@(posedge Clk)
    if(Rst)
        state <= S0;
    else
        state <= nextState;

 endmodule
```

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

```
`default_nettype none

// THis module describes the top level traffic light controller module discussed in lab 11

module tlc_controller(
    output wire [1:0] highwaySignal, farmSignal, //connected to LEDs
    output wire [3:0] JB,
    input wire Clk,

    // the buttons provide input to our top-level circuit
    input wire Rst  //use as reset
    //input wire farmSensor
);

    // intermediate nets
    wire RstSync;
    wire RstCount;
    reg [30:0] Count;

    assign JB[3] = RstCount;

    // syncronize button inputs
    synchronizer syncRst(RstSync, Rst, Clk);
    //synchronizer syncFS(farmSync, farmSensor, Clk);

    // instantiate FSM

  tlc_fsm FSM(
        .state(JB[2:0]), //wire state up to J for debug
        .RstCount(RstCount),
        .highwaySignal(highwaySignal[1:0]),
        .farmSignal(farmSignal[1:0]),
        .Count(Count[30:0]),
        .Clk(Clk),
        .Rst(RstSync)
        //.farmSensor(farmSensor)
  );

  // counter w/ sybcronous reset
  always@(posedge Clk)
        if(RstCount)
            Count <= 0;
        else
            Count <= Count + 1;

endmodule
```

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

```
`default_nettype none

// THis module describes the top level traffic light controller module discussed in lab 11

module tlc_controller(
    output wire [1:0] highwaySignal, farmSignal, //connected to LEDs
    output wire [3:0] JB,
    input wire Clk,

    // the buttons provide input to our top-level circuit
    input wire Rst,  //use as reset
    input wire farmSensor
);

    // intermediate nets
    wire RstSync;
    wire RstCount;
    wire farmSync;
    reg [30:0] Count;

    assign JB[3] = RstCount;

    // syncronize button inputs
    synchronizer syncRst(RstSync, Rst, Clk);
    synchronizer syncFS(farmSync, farmSensor, Clk);

    // instantiate FSM

  tlc_fsm FSM(
        .state(JB[2:0]), //wire state up to J for debug
        .RstCount(RstCount),
        .highwaySignal(highwaySignal[1:0]),
        .farmSignal(farmSignal[1:0]),
        .Count(Count[30:0]),
        .Clk(Clk),
        .Rst(RstSync),
        .farmSensor(farmSensor),
        .farmSync(farmSync)
  );

  // counter w/ sybcronous reset
  always@(posedge Clk)
        if(RstCount)
            Count <= 0;
        else
            Count <= Count + 1;

endmodule
```

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-S0Tjj9dUdnC6r3I/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:

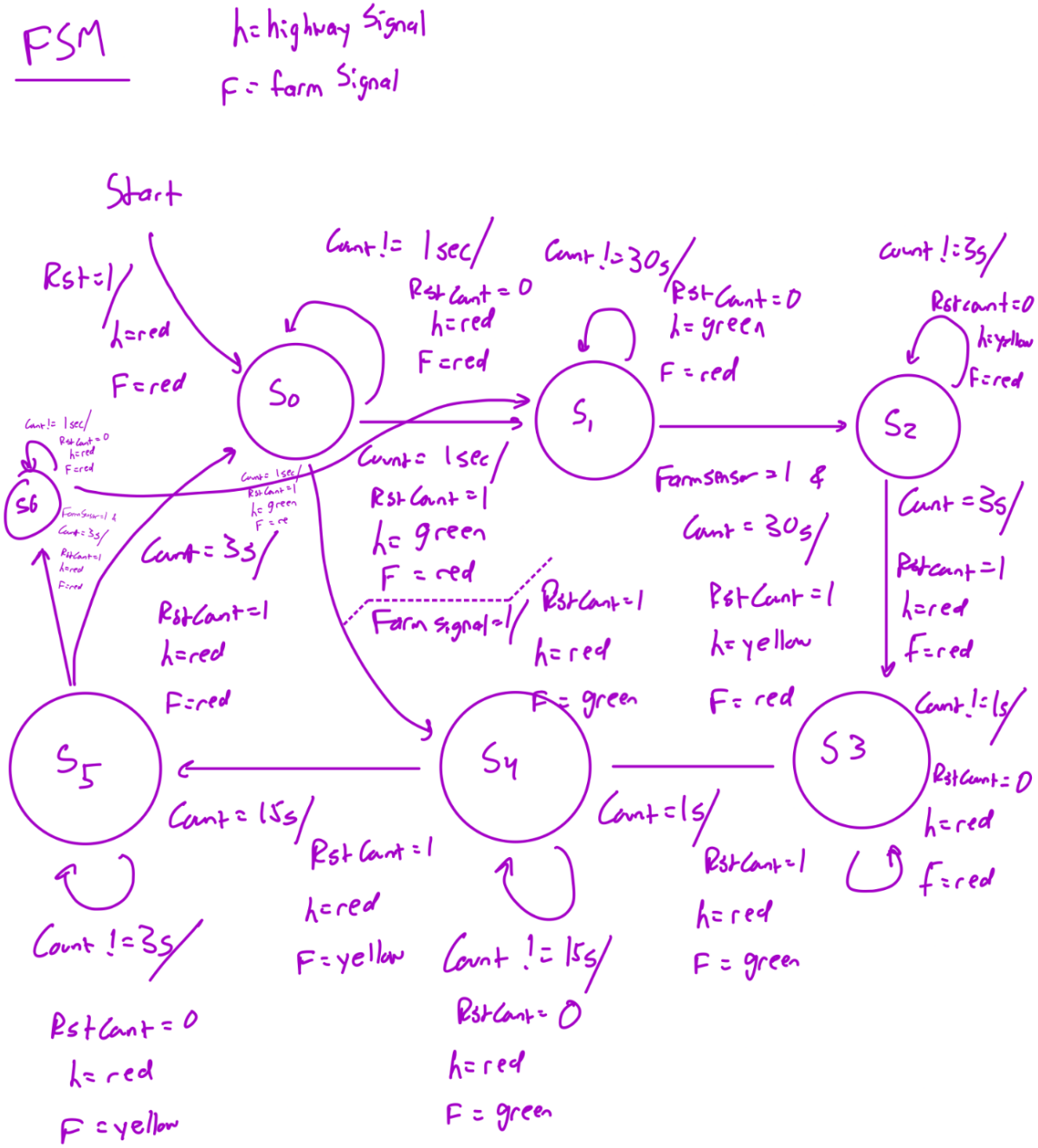
To complete this lab, I was able to use my knowledge of how to run and set up Vivado to code a traffic light FSM and controller code. Once this step was completed, the code was loaded onto the Zybo board and verified for functionality, and then a video taken for the lab demo. Next, part 2 was completed 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.