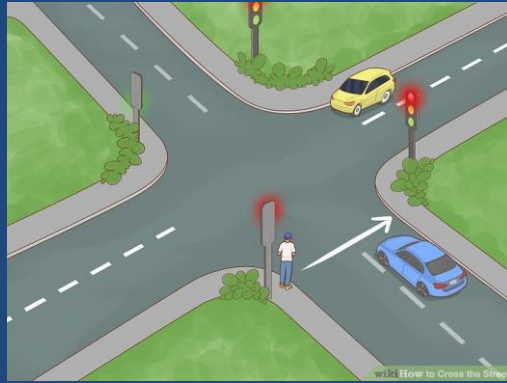


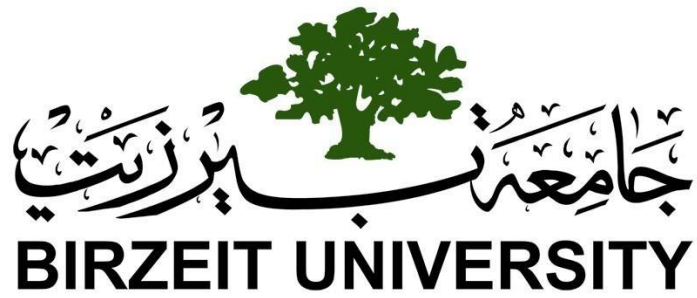
Highway and Farm Roads



Traffic Light Design

**ADVANCED DIGITAL DESIGN
ENCS3310**

Omar Masalmah



FACULTY OF ENGINEERING AND TECHNOLOGY
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
ADVANCED DIGITAL DESIGN ENCS3310
COURSE PROJECT

Prepared By:

Omar Masalmah "1200060"

Instructor :

Dr. Abdallatif Abuissa

Section : 1

Date : 27/1/2023

Table of Contents

1. Brief Introduction and Background.....	3
2. Design Philosophy.....	5
2.a. Traffic Module.....	5
3. Test Generator.....	10
4. System Analyzer.....	11
5. Verification.....	12
6. Results Waveforms.....	13
7. Conclusion.....	15
8. Appendix.....	16

Table of Figures

Table 1 : All the states of the traffic light.....	4
Figure 1: Highway and Farm Roads.....	3
Figure 2(a): Traffic module code.....	6
Figure 2(b): Traffic module code.....	9
Figure 3: 5-bit counter code.....	7
Figure 4: always for states of machine.....	8
Figure 5: always for values of signals.....	8
Figure 6: Test generator Code.....	10
Figure 7: System Analyzer code.....	11
Figure 8: Verification System code.....	12
Figure 9: System waveform.....	13
Figure 10: Reset waveform.....	13
Figure 11: Go waveform.....	13
Figure 12: Wrong value code.....	14
Figure 13: Wrong value message.....	14

BRIEF INTRODUCTION AND BACKGROUND

In this project we will build traffic light controller. The traffic light controller project is a digital system that controls the traffic lights at an intersection of a highway and a farm road. The system consists of four traffic lights, two for the highway and two for the farm road. Each traffic light has two states: red and green. The system also has a yellow state for each light, which is used when transitioning from green to red. The system also has a red-yellow state for each light, which is used when transitioning from red to green.

A state machine that analyzes the traffic light's current status and modifies it in accordance with a preset set of rules controls the system. A clock signal synchronizes the state machine, and it is also controlled by reset and go signals. The reset signal is used to reset the system, while the go signal is used to start or stop the state machine.

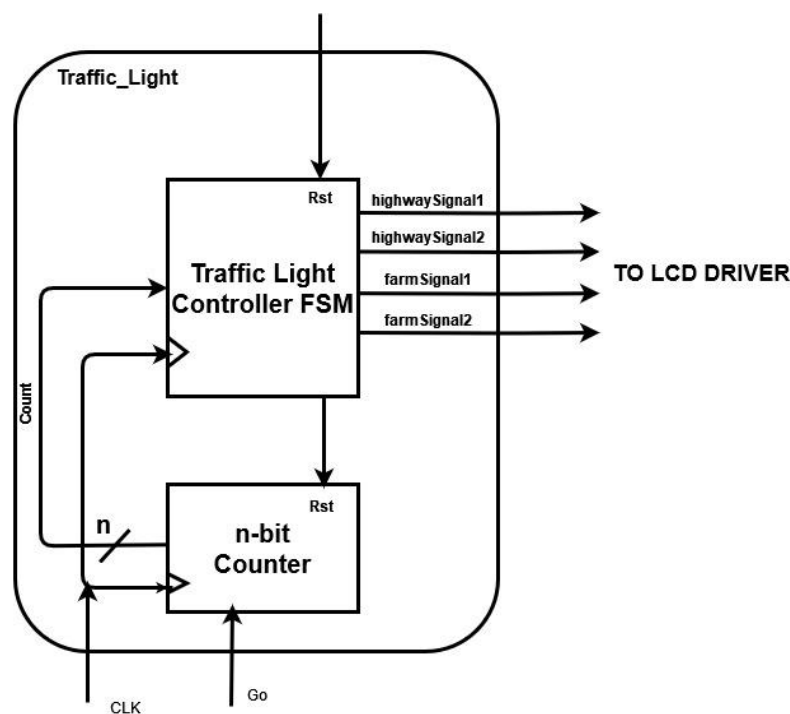


Figure 1: Highway and Farm Roads

The system has a counter as well, which is used to record the amount of time spent in each state. This is used to control the length of the various states and make sure the traffic lights are changed at the correct moment.

Table 1 : All the states of the traffic light

State	Highway TL1	Highway TL2	Farm TL1	Farm TL2	Delay [Sec]
S0	Red	Red	Red	Red	1
S1	Red-Yellow	Red-Yellow	Red	Red	2
S2	Green	Green	Red	Red	30
S3	Green	Yellow	Red	Red	2
S4	Green	Red	Red	Red	10
S5	Yellow	Red	Red	Red	2
S6	Red	Red	Red	Red	1
S7	Red	Red	Red-Yellow	Red-Yellow	2
S8	Red	Red	Green	Green	15
S9	Red	Red	Green	Yellow	2
S10	Red	Red	Green	Red	5
S11	Red	Red	Yellow	Red-Yellow	2
S12	Red	Red	Red	Green	10
S13	Red	Red	Red	Yellow	2
S14	Red	Red	Red	Red	1
S15	Red	Red-Yellow	Red	Red	2
S16	Red	Green	Red	Red	15
S17	Red	Yellow	Red	Red	3

DESIGN PHILOSOPHY

TRAFFIC MODULE

```
1 module traffic_light_controller (input clk, input Rst, input Go,
2                                output reg [4:0] state,
3                                output reg [1:0] highway_signal_1,
4                                output reg [1:0] highway_signal_2,
5                                output reg [1:0] farm_signal_1,
6                                output reg [1:0] farm_signal_2);
7
8 reg [4:0] counter ;
9
10
11 //declare a parameters
12 parameter red = 2'b10;
13 parameter green = 2'b00;
14 parameter yellow = 2'b01;
15 parameter red_yellow = 2'b11;
16
17 always @(posedge clk or negedge Rst ) begin
18
19     if (~Rst) begin
20         state = 0;
21         counter = 0;
22     end
23     else
24         begin
25             if (Go) begin
26                 //counter increase 1 each clock cycle
27                 counter = counter + 1;
28             end
29
30             if (Go) begin
31                 // case to determine time of each state then go to next state
32                 case (state)
33                     0: begin
34                         // if the counter reach time of delay it will go to next state
35                         if (counter == 2) begin
36                             if (counter == 2) begin
37                                 state = 1;
38                                 counter = 5'b0;
39                             end
40                         end
41                     1: begin
42                         if (counter == 2) begin
43                             state = 2;
44                             counter = 5'b0;
45                         end
46                     end
47                     2: begin
48                         if (counter == 30) begin
49                             state = 3;
50                             counter = 5'b0;
51                         end
52                     end
53                     3: begin
54                         if (counter == 2) begin
55                             state = 4;
56                             counter = 5'b0;
57                         end
58                     end
59                     4: begin
60                         if (counter == 10) begin
61                             state = 5;
62                             counter = 5'b0;
63                         end
64                     end
65                 endcase
66             end
67         end
68     end
```

```

69 5: begin
70
71     if (counter == 2) begin
72         state = 6;
73         counter = 5'b0;
74     end
75 end
76 6: begin
77
78     if (counter == 1) begin
79         state = 7;
80         counter = 5'b0;
81     end
82 end
83 7: begin
84
85     if (counter == 2) begin
86         state = 8;
87         counter = 5'b0;
88     end
89 end
90 8: begin
91
92     if (counter == 15) begin
93         state = 9;
94         counter = 5'b0;
95     end
96 end
97 9: begin
98
99     if (counter == 2) begin
100         state = 10;
101         counter = 5'b0;
102     end
103 end
104
105 10: begin
106
107     if (counter == 5) begin
108         state = 11;
109         counter = 5'b0;
110     end
111 end
112 11: begin
113
114     if (counter == 2) begin
115         state = 12;
116         counter = 5'b0;
117     end
118 end
119 12: begin
120
121     if (counter == 10) begin
122         state = 13;
123         counter = 5'b0;
124     end
125 end
126 13: begin
127
128     if (counter == 2) begin
129         state = 14;
130         counter = 5'b0;
131     end
132 end
133 14: begin
134
135     if (counter == 1) begin
136         state = 15;
137         counter = 5'b0;
138     end
139 end
140
141 15: begin
142
143     if (counter == 2) begin
144         state = 16;
145         counter = 5'b0;
146     end
147 end
148 16: begin
149
150     if (counter == 15) begin
151         state = 17;
152         counter = 5'b0;
153     end
154 end
155 17: begin
156
157     if (counter == 3) begin
158         state = 0;
159         counter = 1;
160     end
161 end
162 endcase
163 end
164

```

Figure 2(a): Traffic module code

I built a traffic light controller module that manages the traffic lights at the intersection of a country highway and a farm road. It has a number of inputs, including clk, Rst, and Go, and a number of outputs, including state, highway signal 1, highway signal 2, farm signal 1, and farm signal 2.

The inputs are:

clk is the clock signal that synchronizes the different parts of the design.

Rst is the reset signal that returns the design to its initial state.

Go is the signal that starts or stops the state machine.

The outputs are:

state is an output that represents the current state of the traffic lights

highway_signal_1 and **highway_signal_2** are the outputs that control the state of the first and second traffic lights on the highway.

farm_signal_1 and **farm_signal_2** are the outputs that control the state of the first and second traffic lights on the farm road.

```
reg [4:0] counter ;  
    if (Go) begin  
        //counter increase 1 each clock cycle  
        counter = counter + 1;  
    end
```

Figure 3: 5-bit counter code

The counter is implemented using a register that can store 5 bit values, which is incremented by one on each rising edge of the clock signal. It used to counts the number of clock cycles that have passed since it was last reset. It is used to keep track of the time spent in each state of the traffic lights, and to control the duration of each state. So it determines the time needed for each state.

The code includes two always blocks. The first always block is used to implement the state machine. To keep track of the status of the traffic lights, the block has a counter that keeps track of the number of clock cycles. The block additionally contains a case statement that establishes the next state in light of the current state and the counter's value.

```
always @(posedge clk or negedge Rst ) begin
    if (~Rst) begin
        state = 0;
        counter = 0;
    end
    else
    begin
        if (Go) begin
            //counter increase 1 each clock cycle
            counter = counter + 1;
        end

        if (Go) begin
            // case to determine time of each state then go to next state
            case (state)
            0: begin
                // if the counter reach time of delay it will go to next state
                if (counter == 2) begin
                    state = 1;
                    counter = 5'b0;
                end
            end
            1: begin
                if (counter == 2) begin
                    state = 2;
                    counter = 5'b0;
                end
            end
        end
    end
end
```

Figure 4: always for states of machine

As soon as the status changes, the second always block is activated. It has a case statement that determines how to drive the outputs based on how the traffic lights are currently working.

```
always @(state) begin
    // store the signals
    case (state)
    0: begin
        highway_signal_1 = red;
        highway_signal_2 = red;
        farm_signal_1 = red;
        farm_signal_2 = red;
    end
end
```

Figure 5: always for values of signals

```

165 always @(state) begin
166
167     // store the signals
168     case (state)
169     0: begin
170         highway_signal_1 = red;
171         highway_signal_2 = red;
172         farm_signal_1 = red;
173         farm_signal_2 = red;
174     end
175
176     1: begin
177         highway_signal_1 = red_yellow;
178         highway_signal_2 = red_yellow;
179         farm_signal_1 = red;
180         farm_signal_2 = red;
181     end
182
183     2: begin
184         highway_signal_1 = green;
185         highway_signal_2 = green;
186         farm_signal_1 = red;
187         farm_signal_2 = red;
188     end
189
190     3: begin
191         highway_signal_1 = green;
192         highway_signal_2 = yellow;
193         farm_signal_1 = red;
194         farm_signal_2 = red;
195     end
196
197     4: begin
232
233         highway_signal_1 = red;
234         highway_signal_2 = red;
235         farm_signal_1 = green;
236         farm_signal_2 = yellow;
237     end
238
239     10: begin
240         highway_signal_1 = red;
241         highway_signal_2 = red;
242         farm_signal_1 = green;
243         farm_signal_2 = red;
244     end
245
246     11: begin
247         highway_signal_1 = red;
248         highway_signal_2 = red;
249         farm_signal_1 = yellow;
250         farm_signal_2 = red_yellow;
251     end
252
253     12: begin
254         highway_signal_1 = red;
255         highway_signal_2 = red;
256         farm_signal_1 = red;
257         farm_signal_2 = green;
258     end
259
260     13: begin
261         highway_signal_1 = red;
262         highway_signal_2 = red;
263         farm_signal_1 = red;
264         farm_signal_2 = yellow;
265
266
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
4: begin
    highway_signal_1 = green;
    highway_signal_2 = red;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
5: begin
    highway_signal_1 = yellow;
    highway_signal_2 = red;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
6: begin
    highway_signal_1 = red;
    highway_signal_2 = red;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
7: begin
    highway_signal_1 = red;
    highway_signal_2 = red;
    farm_signal_1 = red_yellow;
    farm_signal_2 = red_yellow;
end
8: begin
    highway_signal_1 = red;
    highway_signal_2 = red;
    farm_signal_1 = green;
    farm_signal_2 = green;
end
14: begin
    highway_signal_1 = red;
    highway_signal_2 = red;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
15: begin
    highway_signal_1 = red;
    highway_signal_2 = red_yellow;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
16: begin
    highway_signal_1 = red;
    highway_signal_2 = green;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
17: begin
    highway_signal_1 = red;
    highway_signal_2 = yellow;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
default: begin
    highway_signal_1 = red;
    highway_signal_2 = red;
    farm_signal_1 = red;
    farm_signal_2 = red;
end
endcase
end
endmodule

```

Figure 2(b): Traffic module code

TEST GENERATOR

This is a code for a generate_test module which is used to generate test inputs for the traffic light controller design.

```
//generator module to set the inputs values
module generate_test(
    output reg clk,
    output reg Rst,
    output reg Go);

    initial begin
        clk=0;
        Rst=0;
        Go=0;
    end

    always #10 clk = ~clk;

    initial begin
        #20; Go = 1;
        #90; Rst = 1;
        #600; Rst = 0;
        #900; Rst = 1;
        #1000; Go = 0;
        #1300; Go = 1;
        #15000;
        $finish;
    end
endmodule
```

Figure 6: Test generator Code

SYSTEM ANALYZER

This is an analyzer module for the traffic light controller. It takes in the current state, as well as the current signal outputs from the traffic module (highway_signal_1, highway_signal_2, farm_signal_1, and farm_signal_2). It also defines the expected output signals for each state, stored in the variables highway_exp_1, highway_exp_2, farm_exp_1, and farm_exp_2.

Then check if the actual output signals match the expected output signals. If there is a mismatch, it will print a message indicating which signal is incorrect and in which state the mismatch occurred.

```
//analyzer module to check if the output signals are true
module analyze (
    input [4:0] state,
    input wire [1:0] highway_signal_1,
    input wire [1:0] highway_signal_2,
    input wire [1:0] farm_signal_1,
    input wire [1:0] farm_signal_2);

//declare a parameters
parameter red = 2'b10;
parameter green = 2'b00;
parameter yellow = 2'b01;
parameter red_yellow = 2'b11;

reg [4:0] counter;

//expected outputs
reg [1:0] highway_exp_1;
reg [1:0] highway_exp_2;
reg [1:0] farm_exp_1;
reg [1:0] farm_exp_2;

always @(state) begin
    case (state)
        1: begin
            highway_exp_1 = red;
            highway_exp_2 = red;
            farm_exp_1 = red;
            farm_exp_2 = red;

            if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 0");
            if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 0");
            if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state state: 0");
            if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state state: 0");
        end
    end
end
```

Figure 7: System Analyzer code

VERIFICATION

For my traffic light controller project, the top-level is **Testbench**. The **generate test**, **traffic light controller**, and **analyze modules** are all immediately called. The **clk**, **Rst**, and **Go** inputs (for the traffic light controller module) are generated by the generate test module. It is the job of the traffic light controller module to put the traffic light controller logic into execution. The traffic light controller module's expected and actual outputs must be compared in the analyze module, which decides whether the result is correct or not.

```
//Verification for the system
module Testbench;

    reg clk,Go,Rst;
    wire [4:0] state;
    wire [1:0] highway_1;
    wire [1:0] highway_2;
    wire [1:0] farm_1;
    wire [1:0] farm_2;

    generate_test gen(clk,Rst,Go);
    traffic_light_controller traffic(clk,Rst,Go,state,highway_1,highway_2,farm_1,farm_2);
    analyze anl(state,highway_1,highway_2,farm_1,farm_2);

endmodule
```

Figure 8: Verification System code

RESULTS WAVEFORMS

Here the waveform of the system that contain inputs and outputs.

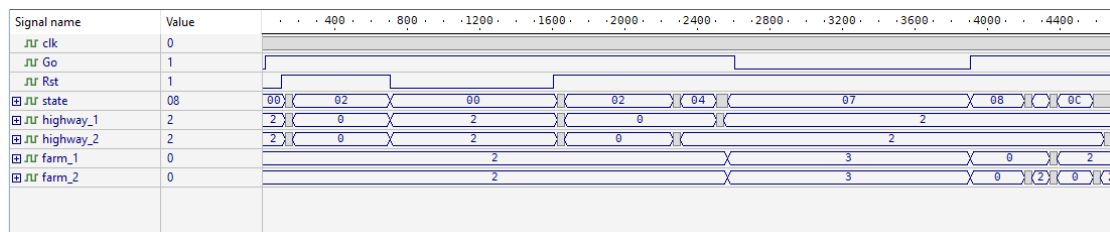


Figure 9: System waveform

When Rst is 0 the machine will reset to the state 0.

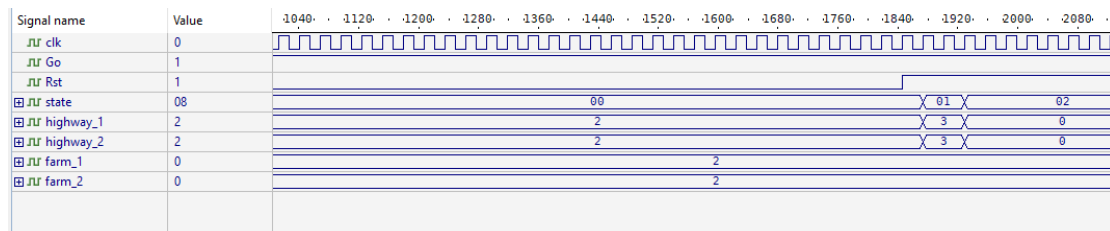


Figure 10: Reset waveform

In normal conditions the machine will move from state to state, but when Go are equal 0 the machine will set on current state until Go be 1.

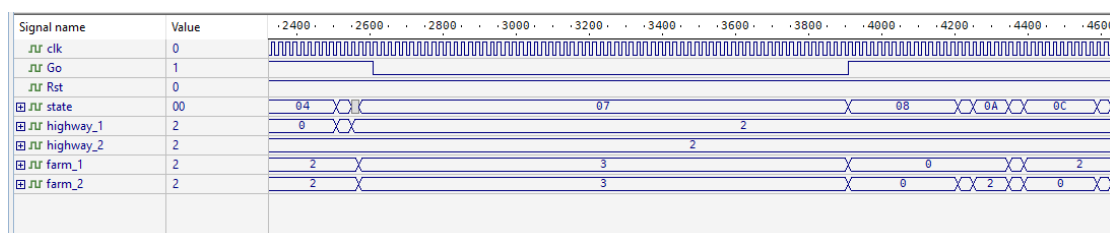
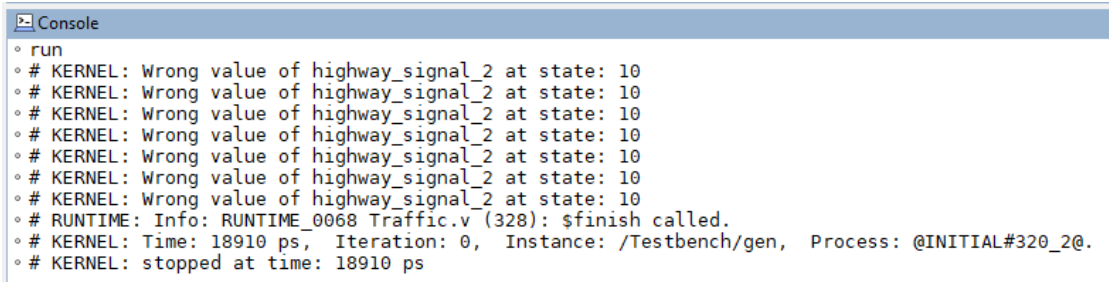


Figure 11: Go waveform

If I try to change some of signal value the analyzer will print wrong message.

```
10: begin
    highway_signal_1 = red;
    highway_signal_2 = green; //The real value is red
    farm_signal_1 = green;
    farm_signal_2 = red;
end
```

Figure 12: Wrong value code



The screenshot shows a console window titled "Console" with the following output:

```
run
# KERNEL: Wrong value of highway_signal_2 at state: 10
# KERNEL: Wrong value of highway_signal_2 at state: 10
# KERNEL: Wrong value of highway_signal_2 at state: 10
# KERNEL: Wrong value of highway_signal_2 at state: 10
# KERNEL: Wrong value of highway_signal_2 at state: 10
# KERNEL: Wrong value of highway_signal_2 at state: 10
# KERNEL: Wrong value of highway_signal_2 at state: 10
# RUNTIME: Info: RUNTIME_0068 Traffic.v (328): $finish called.
# KERNEL: Time: 18910 ps, Iteration: 0, Instance: /Testbench/gen, Process: @INITIAL#320_2@.
# KERNEL: stopped at time: 18910 ps
```

Figure 13: Wrong value message

NOTE: Wrong message appear more than one time because the system works periodically and repeats itself until finish time.

CONCLUSION

In this project, i have designed a traffic light controller for a highway and farm road intersection using Verilog. The traffic light controller is a finite state machine that uses a counter to keep track of time and change between different states.

Furthermore, i learned how to design and implement a traffic light controller system using Verilog. This includes both creating the individual system components, such as the counter and traffic light controller, as well as the connections that would link them all together. Additionally, we learnt how to use generate and analyze modules on test benches to test the system's functionality. We also learned about important ideas in digital design like state machines and timing limits.

Basically, this project has shown that a traffic light controller system can be designed and implemented using Verilog, and it may be improved upon to increase functionality and performance.

APPENDIX

```
module traffic_light_controller (input clk, input Rst, input Go,
```

```
    output reg [4:0] state,
```

```
    output reg [1:0] highway_signal_1,
```

```
    output reg [1:0] highway_signal_2,
```

```
    output reg [1:0] farm_signal_1,
```

```
    output reg [1:0] farm_signal_2);
```

```
reg [4:0] counter ;
```

```
//declare a parameters
```

```
parameter red = 2'b10;
```

```
parameter green = 2'b00;
```

```
parameter yellow = 2'b01;
```

```
parameter red_yellow = 2'b11;
```

```
always @(posedge clk or negedge Rst ) begin
```

```
    if (~Rst) begin
```

```
        state = 0;
```

```
        counter = 0;
```

```
    end
```

```
else
```

```
    begin
```

```
        if (Go) begin
```

```
            //counter increase 1 each clock cycle
```

```
            counter = counter + 1;
```

```
        end
```

```
if (Go) begin

    // case to determine time of each state then go to next state

    case (state)

0: begin

    // if the counter reach time of delay it will go to next state

    if (counter == 2) begin

        state = 1;

        counter = 5'b0;

    end

end

1: begin

    if (counter == 2) begin

        state = 2;

        counter = 5'b0;

    end

end

2: begin

    if (counter == 30) begin

        state = 3;

        counter = 5'b0;

    end

end

3: begin

    if (counter == 2) begin

        state = 4;

        counter = 5'b0;

    end

end
```

```
4: begin
    if (counter == 10) begin
        state = 5;
        counter = 5'b0;
    end
end

5: begin
    if (counter == 2) begin
        state = 6;
        counter = 5'b0;
    end
end

6: begin
    if (counter == 1) begin
        state = 7;
        counter = 5'b0;
    end
end

7: begin
    if (counter == 2) begin
        state = 8;
        counter = 5'b0;
    end
end

8: begin
    if (counter == 15) begin
        state = 9;
        counter = 5'b0;
    end
end
```

```
9: begin
    if (counter == 2) begin
state = 10;
counter = 5'b0;
    end
end

10: begin
    if (counter == 5) begin
state = 11;
counter = 5'b0;
    end
end

11: begin
    if (counter == 2) begin
state = 12;
counter = 5'b0;
    end
end

12: begin
    if (counter == 10) begin
state = 13;
counter = 5'b0;
    end
end

13: begin
    if (counter == 2) begin
state = 14;
counter = 5'b0;
    end
end
```

```
14: begin
    if (counter == 1) begin
        state = 15;
        counter = 5'b0;
    end
end

15: begin
    if (counter == 2) begin
        state = 16;
        counter = 5'b0;
    end
end

16: begin
    if (counter == 15) begin
        state = 17;
        counter = 5'b0;
    end
end

17: begin
    if (counter == 3) begin
        state = 0;
        counter = 1;
    end
end

endcase
end
end
```

```
always @(state) begin

    // store the signals

    case (state)

0: begin

    highway_signal_1 = red;

    highway_signal_2 = red;

    farm_signal_1 = red;

    farm_signal_2 = red;

end

1: begin

    highway_signal_1 = red_yellow;

    highway_signal_2 = red_yellow;

    farm_signal_1 = red;

    farm_signal_2 = red;

end

2: begin

    highway_signal_1 = green;

    highway_signal_2 = green;

    farm_signal_1 = red;

    farm_signal_2 = red;

end

3: begin

    highway_signal_1 = green;

    highway_signal_2 = yellow;

    farm_signal_1 = red;

    farm_signal_2 = red;
```

end

4: begin

highway_signal_1 = green;

highway_signal_2 = red;

farm_signal_1 = red;

farm_signal_2 = red;

end

5: begin

highway_signal_1 = yellow;

highway_signal_2 = red;

farm_signal_1 = red;

farm_signal_2 = red;

end

6: begin

highway_signal_1 = red;

highway_signal_2 = red;

farm_signal_1 = red;

farm_signal_2 = red;

end

7: begin

highway_signal_1 = red;

highway_signal_2 = red;

farm_signal_1 = red_yellow;

farm_signal_2 = red_yellow;

end

8: begin

highway_signal_1 = red;

highway_signal_2 = red;

farm_signal_1 = green;

farm_signal_2 = green;

end

9: begin

highway_signal_1 = red;

highway_signal_2 = red;

farm_signal_1 = green;

farm_signal_2 = yellow;

end

10: begin

highway_signal_1 = red;

highway_signal_2 = red;

farm_signal_1 = green;

farm_signal_2 = red;

end

11: begin

highway_signal_1 = red;

highway_signal_2 = red;

farm_signal_1 = yellow;

farm_signal_2 = red_yellow;

end

12: begin

 highway_signal_1 = red;

 highway_signal_2 = red;

 farm_signal_1 = red;

 farm_signal_2 = green;

end

13: begin

 highway_signal_1 = red;

 highway_signal_2 = red;

 farm_signal_1 = red;

 farm_signal_2 = yellow;

end

14: begin

 highway_signal_1 = red;

 highway_signal_2 = red;

 farm_signal_1 = red;

 farm_signal_2 = red;

end

15: begin

 highway_signal_1 = red;

 highway_signal_2 = red_yellow;

 farm_signal_1 = red;

 farm_signal_2 = red;

end

```
16: begin
```

```
    highway_signal_1 = red;
```

```
    highway_signal_2 = green;
```

```
    farm_signal_1 = red;
```

```
    farm_signal_2 = red;
```

```
end
```

```
17: begin
```

```
    highway_signal_1 = red;
```

```
    highway_signal_2 = yellow;
```

```
    farm_signal_1 = red;
```

```
    farm_signal_2 = red;
```

```
end
```

```
default: begin
```

```
    highway_signal_1 = red;
```

```
    highway_signal_2 = red;
```

```
    farm_signal_1 = red;
```

```
    farm_signal_2 = red;
```

```
        end
```

```
    endcase
```

```
end
```

```
endmodule
```

```
//generator module to set the inputs values
```

```
module generate_test(
```

```
    output reg clk,
```

```
    output reg Rst,
```

```
    output reg Go);
```

```

        initial begin
            clk=0;

            Rst=0;

            Go=0;

        end

        always #10 clk = ~clk;

        initial begin
            #20; Go = 1;

            #90; Rst = 1;

            #600; Rst = 0;

            #900; Rst = 1;

            #1000; Go = 0;

            #1300; Go = 1;

            #15000;

            $finish;

        end

    endmodule

//analyzer module to check if the output signals are true
module analyze (

    input [4:0] state,

    input wire [1:0] highway_signal_1,

    input wire [1:0] highway_signal_2,

    input wire [1:0] farm_signal_1,

    input wire [1:0] farm_signal_2);

```

```

//declare a parameters

parameter red = 2'b10;

parameter green = 2'b00;

parameter yellow = 2'b01;

parameter red_yellow = 2'b11;


reg [4:0] counter;


//expected outputs

reg [1:0] highway_exp_1;

reg [1:0] highway_exp_2;

reg [1:0] farm_exp_1;

reg [1:0] farm_exp_2;


always @(state) begin


case (state)

1: begin

    highway_exp_1 = red;

    highway_exp_2 = red;

    farm_exp_1 = red;

    farm_exp_2 = red;


if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 0");

if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 0");

if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state state: 0");

if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state state: 0");


end

```

2: begin

highway_exp_1 = red_yellow;

highway_exp_2 = red_yellow;

farm_exp_1 = red;

farm_exp_2 = red;

if (highway_signal_1 != highway_exp_1) \$display("Wrong value of highway_signal_1 at state: 1");

if (highway_signal_2 != highway_exp_2) \$display("Wrong value of highway_signal_2 at state: 1");

if (farm_signal_1 != farm_exp_1) \$display("Wrong value of farm_signal_1 at state: 1");

if (farm_signal_2 != farm_exp_2) \$display("Wrong value of farm_signal_2 at state: 1");

end

3: begin

highway_exp_1 = green;

highway_exp_2 = green;

farm_exp_1 = red;

farm_exp_2 = red;

if (highway_signal_1 != highway_exp_1) \$display("Wrong value of highway_signal_1 at state: 2");

if (highway_signal_2 != highway_exp_2) \$display("Wrong value of highway_signal_2 at state: 2");

if (farm_signal_1 != farm_exp_1) \$display("Wrong value of farm_signal_1 at state: 2");

if (farm_signal_2 != farm_exp_2) \$display("Wrong value of farm_signal_2 at state: 2");

end

4: begin

highway_exp_1 = green;

highway_exp_2 = yellow;

farm_exp_1 = red;

farm_exp_2 = red;

```

if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 3");

if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 3");

if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 3");

if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 3");

    end

5: begin

    highway_exp_1 = green;

    highway_exp_2 = red;

    farm_exp_1 = red;

    farm_exp_2 = red;

if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 4");

if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 4");

if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 4");

if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 4");

end

6: begin

    highway_exp_1 = yellow;

    highway_exp_2 = red;

    farm_exp_1 = red;

    farm_exp_2 = red;

if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 5");

if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 5");

if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 5");

if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 5");

end

```

7: begin

 highway_exp_1 = red;

 highway_exp_2 = red;

 farm_exp_1 = red;

 farm_exp_2 = red;

if (highway_signal_1 != highway_exp_1) \$display("Wrong value of highway_signal_1 at state: 6");

if (highway_signal_2 != highway_exp_2) \$display("Wrong value of highway_signal_2 at state: 6");

if (farm_signal_1 != farm_exp_1) \$display("Wrong value of farm_signal_1 at state: 6");

if (farm_signal_2 != farm_exp_2) \$display("Wrong value of farm_signal_2 at state: 6");

end

8: begin

 highway_exp_1 = red;

 highway_exp_2 = red;

 farm_exp_1 = red_yellow;

 farm_exp_2 = red_yellow;

if (highway_signal_1 != highway_exp_1) \$display("Wrong value of highway_signal_1 at state: 7");

if (highway_signal_2 != highway_exp_2) \$display("Wrong value of highway_signal_2 at state: 7");

if (farm_signal_1 != farm_exp_1) \$display("Wrong value of farm_signal_1 at state: 7");

if (farm_signal_2 != farm_exp_2) \$display("Wrong value of farm_signal_2 at state: 7");

end

9: begin

 highway_exp_1 = red;

 highway_exp_2 = red;

 farm_exp_1 = green;

```

    farm_exp_2 = green;

if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 8");
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 8");
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 8");
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 8");

end

10: begin
    highway_exp_1 = red;
    highway_exp_2 = red;
    farm_exp_1 = green;
    farm_exp_2 = yellow;

if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 9");
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 9");
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 9");
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 9");

end

11: begin
    highway_exp_1 = red;
    highway_exp_2 = red;
    farm_exp_1 = green;
    farm_exp_2 = red;

if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 10");
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 10");
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 10");
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 10");

end

```


12: begin

```
highway_exp_1 = red;  
highway_exp_2 = red;  
farm_exp_1 = yellow;  
farm_exp_2 = red_yellow;
```

```
if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 11");  
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 11");  
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 11");  
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 11");  
end
```

13: begin

```
highway_exp_1 = red;  
highway_exp_2 = red;  
farm_exp_1 = red;  
farm_exp_2 = green;
```

```
if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 12");  
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 12");  
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 12");  
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 12");
```

end

14: begin

```
highway_exp_1 = red;  
highway_exp_2 = red;  
farm_exp_1 = red;  
farm_exp_2 = yellow;
```

```
if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 13");  
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 13");  
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 13");  
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 13");  
end
```

15: begin

```
highway_exp_1 = red;  
highway_exp_2 = red;  
farm_exp_1 = red;  
farm_exp_2 = red;
```

```
if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 14");  
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 14");  
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 14");  
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 14");  
end
```

16: begin

```
highway_exp_1 = red;  
highway_exp_2 = red_yellow;  
farm_exp_1 = red;  
farm_exp_2 = red;
```

```
if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 15");  
if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 15");  
if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 15");  
if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 15");  
  
end
```

```

17: begin

    highway_exp_1 = red;

    highway_exp_2 = green;

    farm_exp_1 = red;

    farm_exp_2 = red;


if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 16");

if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 16");

if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 16");

if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 16");


end


18: begin

    highway_exp_1 = red;

    highway_exp_2 = yellow;

    farm_exp_1 = red;

    farm_exp_2 = red;


if (highway_signal_1 != highway_exp_1) $display("Wrong value of highway_signal_1 at state: 17");

if (highway_signal_2 != highway_exp_2) $display("Wrong value of highway_signal_2 at state: 17");

if (farm_signal_1 != farm_exp_1) $display("Wrong value of farm_signal_1 at state: 17");

if (farm_signal_2 != farm_exp_2) $display("Wrong value of farm_signal_2 at state: 17");

end

endcase

end

endmodule

```

```
//Verification for the system
```

```
module Testbench;
```

```
    reg clk,Go,Rst;
```

```
    wire [4:0] state;
```

```
    wire [1:0] highway_1;
```

```
    wire [1:0] highway_2;
```

```
    wire [1:0] farm_1;
```

```
    wire [1:0] farm_2;
```

```
    generate_test gen(clk,Rst,Go);
```

```
    traffic_light_controller traffic(clk,Rst,Go,state,highway_1,highway_2,farm_1,farm_2);
```

```
    analyze anl(state,highway_1,highway_2,farm_1,farm_2);
```

```
endmodule
```