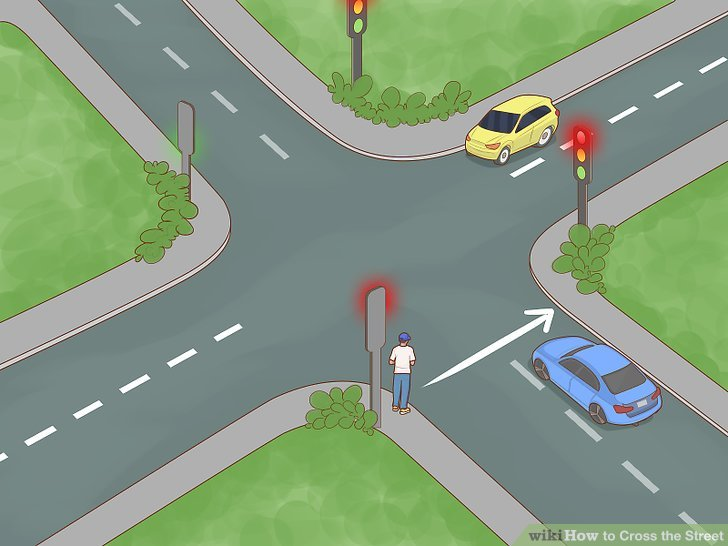
Highway and Farm Roads

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

1. Test Generator…………………………………………………………………………………. 10
2. System Analyzer……………………………………………………………………………….. 11
3. Verification………………………………………………………………………………………. 12
4. Results Waveforms……………………..……………………………………………………. 13
5. Conclusion……………………………………………………………………………………….. 15
6. 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

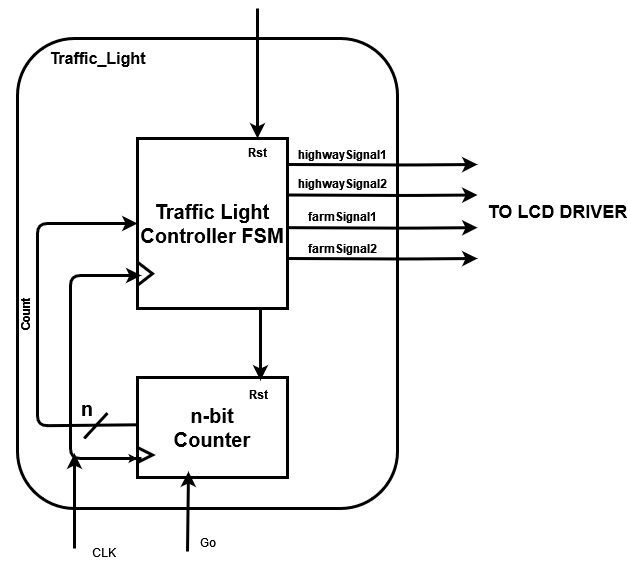
Figure 12: Wrong value code……………………………………………………………………………………………… 14

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*

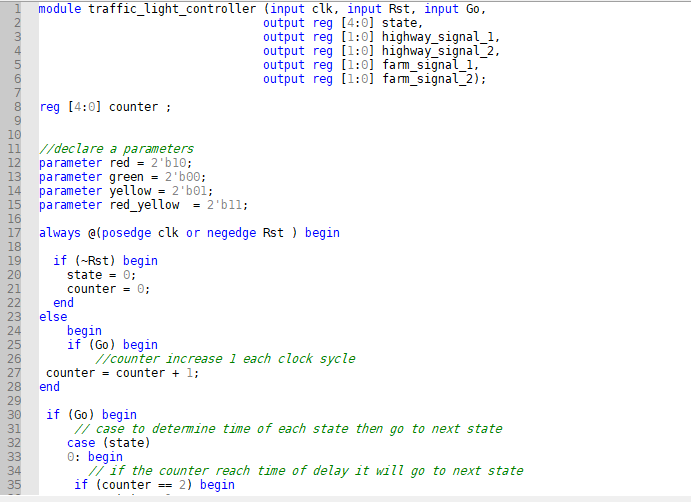
The system has a counter as well, which is used to record the amount of time spent in each state. This is used to control the length of the various states and make sure the traffic lights are changed at the correct moment.

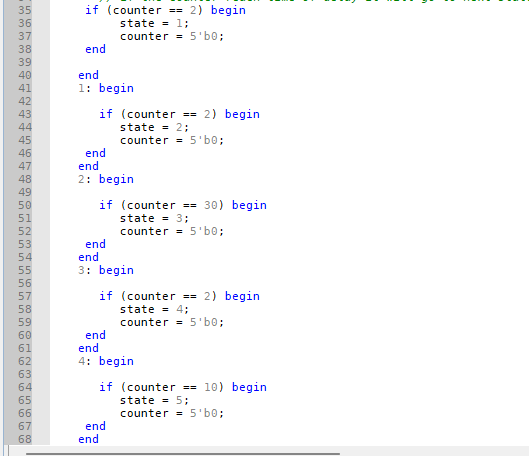
***Table 1 : All the states of the traffic light***

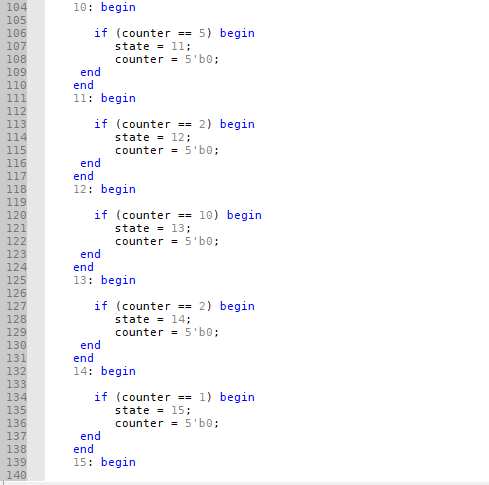
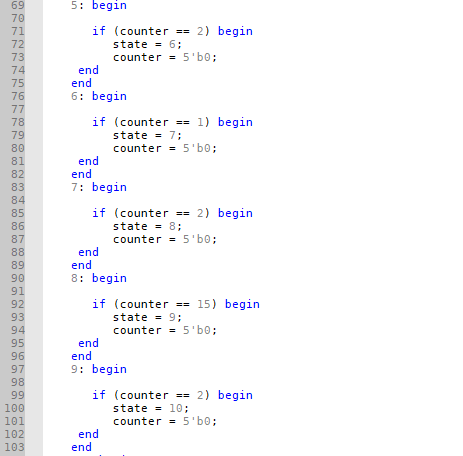
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| State | Highway TL1 | Highway TL2 | Farm TL1 | Farm TL2 | Delay [Sec] |
| S0 | Red | Red | Red | Red | 1 |
| S1 | Red-Yellow | Red-Yellow | Red | Red | 2 |
| S2 | Green | Green | Red | Red | 30 |
| S3 | Green | Yellow | Red | Red | 2 |
| S4 | 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 |

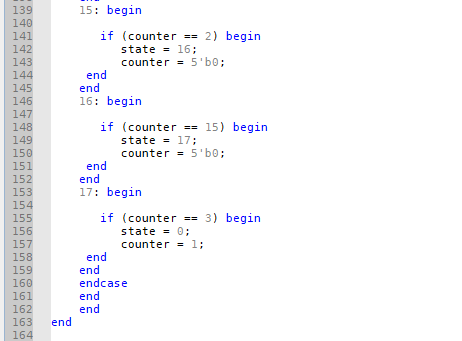
**Design Philosophy**

**Traffic Module**



****





*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.

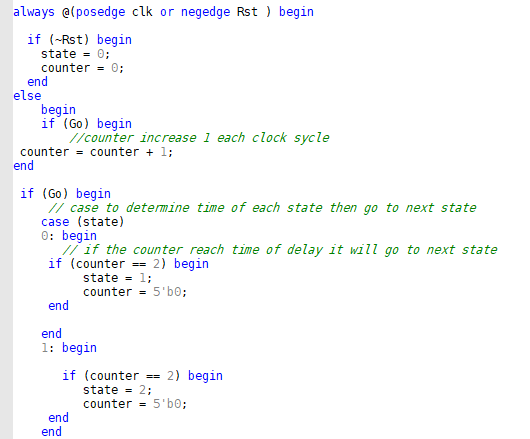
**farm\_signal\_1** and **farm\_signal\_2** are the outputs that control the state of the first and second traffic lights on the farm road.

*Figure 3: 5-bit counter code*

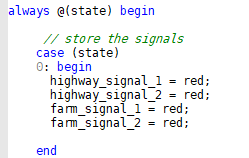
The counter is implemented using a register that can store 5 bit values, which is incremented by one on each rising edge of the clock signal. It used to counts the number of clock cycles that have passed since it was last reset. It is used to keep track of the time spent in each state of the traffic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

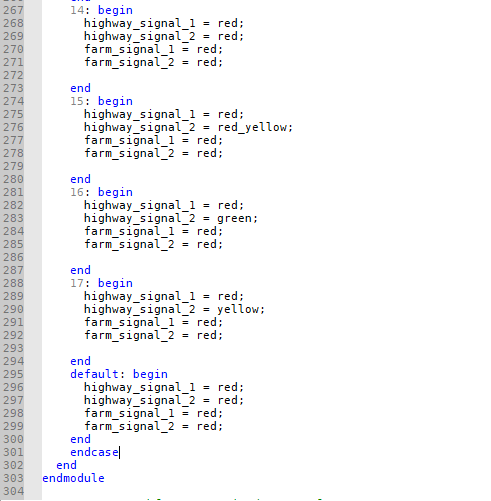
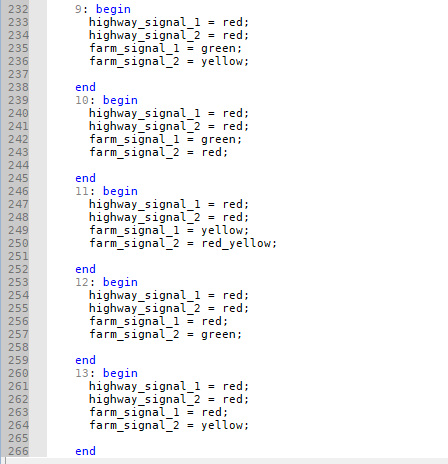
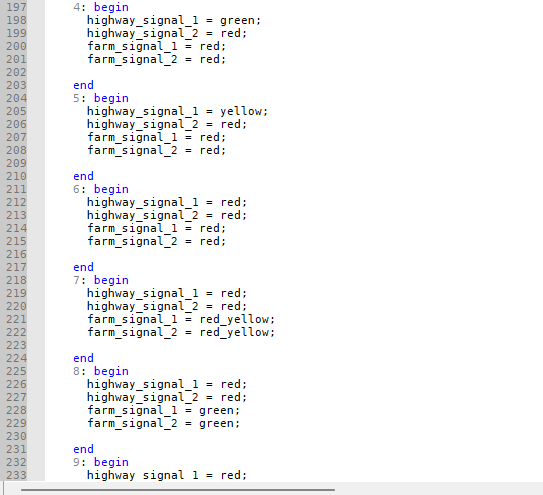
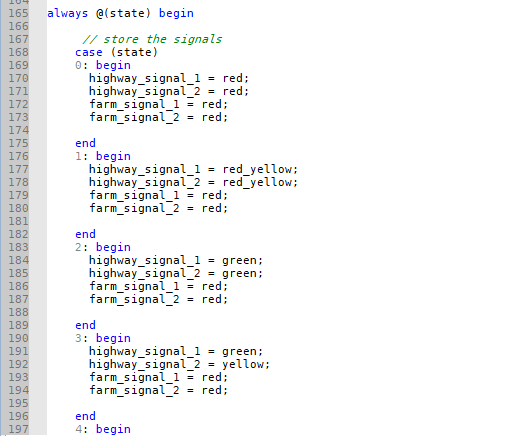


*Figure 4: always for states of machine*

**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.



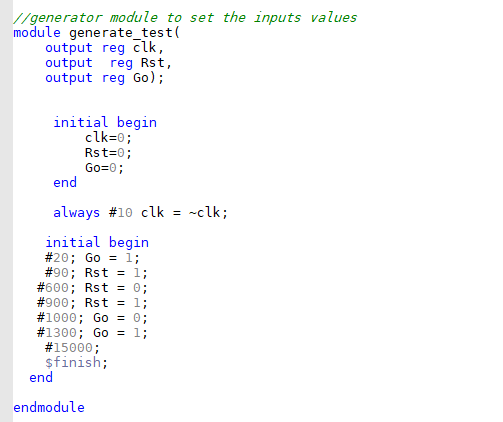
*Figure 5: always for values of signals*



*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.

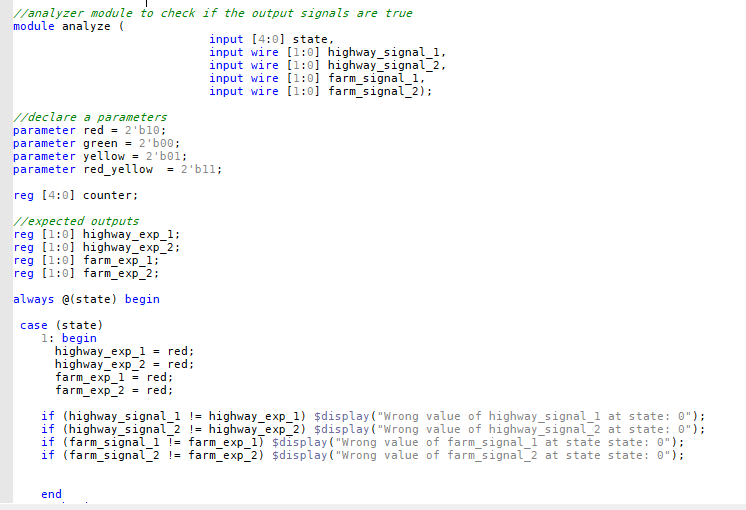


*Figure 6: Test generator Code*

**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.

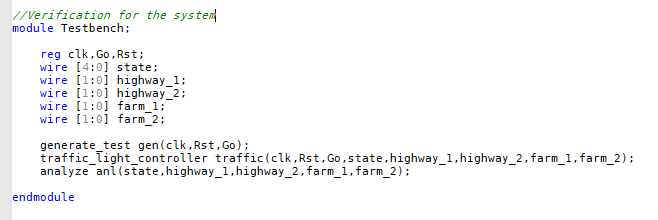
Then check if the actual output signals match the expected output signals. If there is a mismatch, it will print a message indicating which signal is incorrect and in which state the mismatch occurred.



*Figure 7: System Analyzer code*

**Verification**

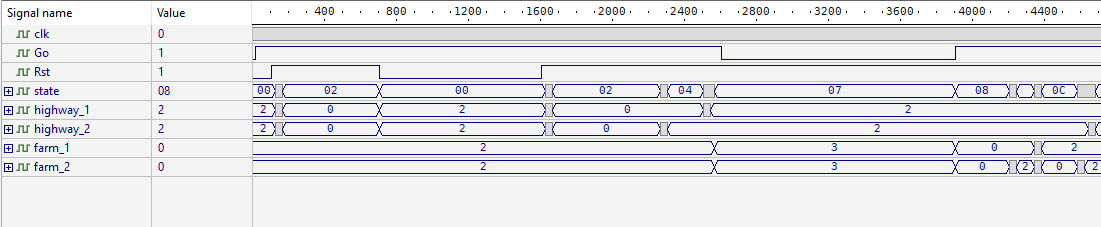
For my traffic light controller project, the top-level is **Testbench**. The **generate test, traffic light controller**, and **analyze modules** are all immediately called. The **clk, Rst**, and **Go** inputs (for the traffic light controller module) are generated by the generate test module. It is 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.

.

*Figure 8: Verification System code*

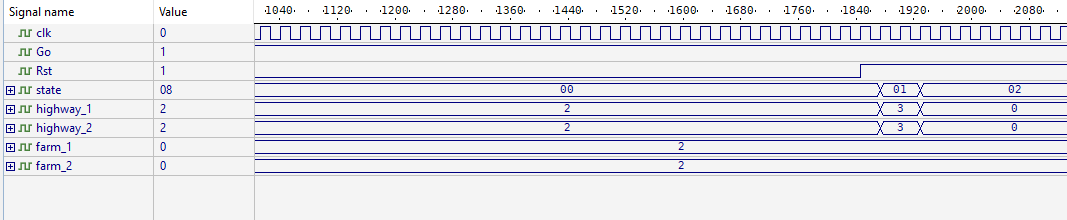
**Results Waveforms**

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

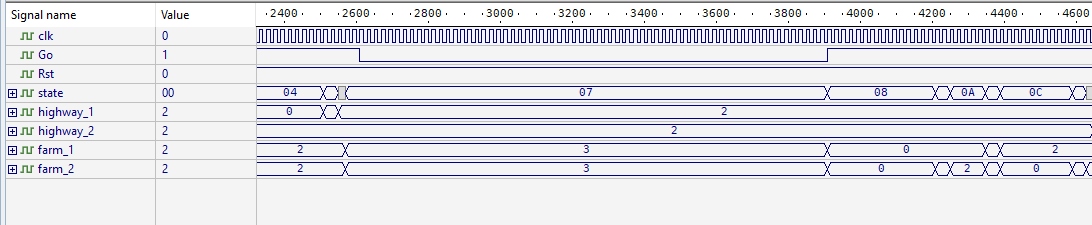


*Figure 9: System waveform*

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

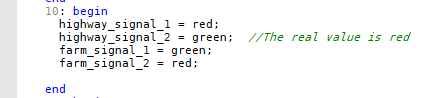
 *Figure 10: Reset waveform*

**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.**

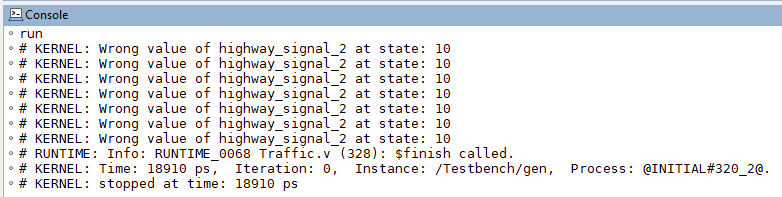
****

*Figure 11: Go waveform*

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

****

*Figure 12: Wrong value code*

****

*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 sycle

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

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