جامعة بيرزيت

BIRZEIT UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

ADVANCED DIGITAL SYSTEMS DESIGN ENCS3310

Course Project

"traffic light design for two roads"

Name: **Musab Masalmah.**

ID: **1200078**

Dr. **Abdellatif Abu-Issa**

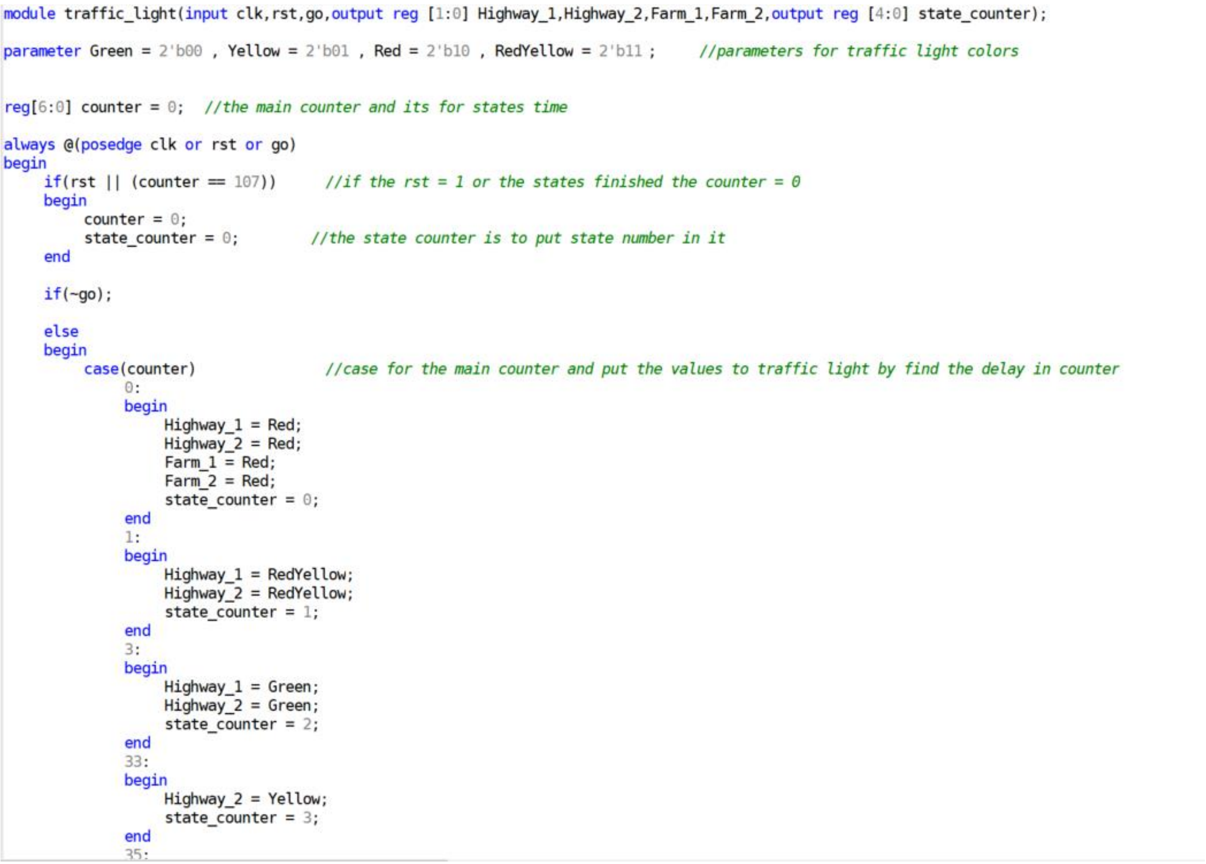
Contents

Introduction

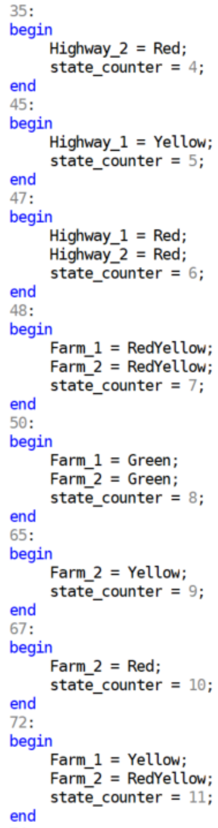
In this project, I will create a traffic light project that controls four streets, high way 1,high way 2, farm way1 and farm way 2, These four traffic lights will regulate the traffic of cars and people at this crossroads, allowing cars to pass through the streets without traffic accidents occurring between these cars.

I write several modules that work together in order to reach the final output of the traffic light. The Generator module sends values to the Traffic Light module, which performs the work and then sends the results to the Analyser , which checks Errors, and finally the test bench module, for which we make a simulation to show the final values.

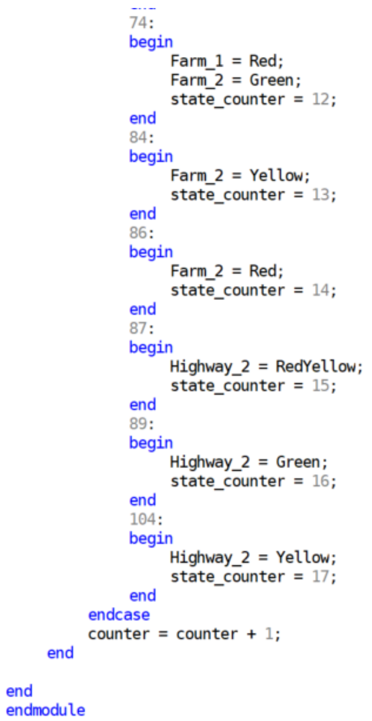
The Code



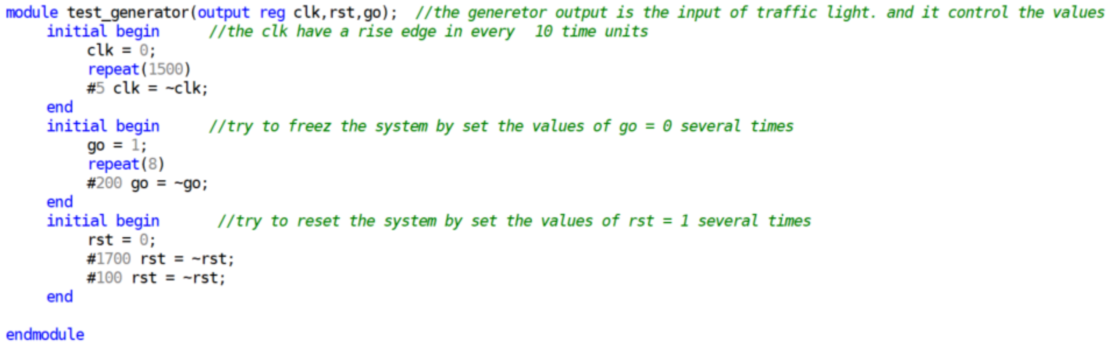
```
module traffic_light(input clk,rst,go,output reg [1:0] Highway_1,Highway_2,Farm_1,Farm_2,output reg [4:0] state_counter);

parameter Green = 2'b00 , Yellow = 2'b01 , Red = 2'b10 , RedYellow = 2'b11 ;      //parameters for traffic light colors


reg[6:0] counter = 0;  //the main counter and its for states time

always @(posedge clk or rst or go)
begin
    if(rst || (counter == 107))      //if the rst = 1 or the states finished the counter = 0
    begin
        counter = 0;
        state_counter = 0;         //the state counter is to put state number in it
    end

    if(~go);

    else
    begin
        case(counter)               //case for the main counter and put the values to traffic light by find the delay in counter
            0:
            begin
                Highway_1 = Red;
                Highway_2 = Red;
                Farm_1 = Red;
                Farm_2 = Red;
                state_counter = 0;
            end
            1:
            begin
                Highway_1 = RedYellow;
                Highway_2 = RedYellow;
                state_counter = 1;
            end
            3:
            begin
                Highway_1 = Green;
                Highway_2 = Green;
                state_counter = 2;
            end
            33:
            begin
                Highway_2 = Yellow;
                state_counter = 3;
            end
            35:
```



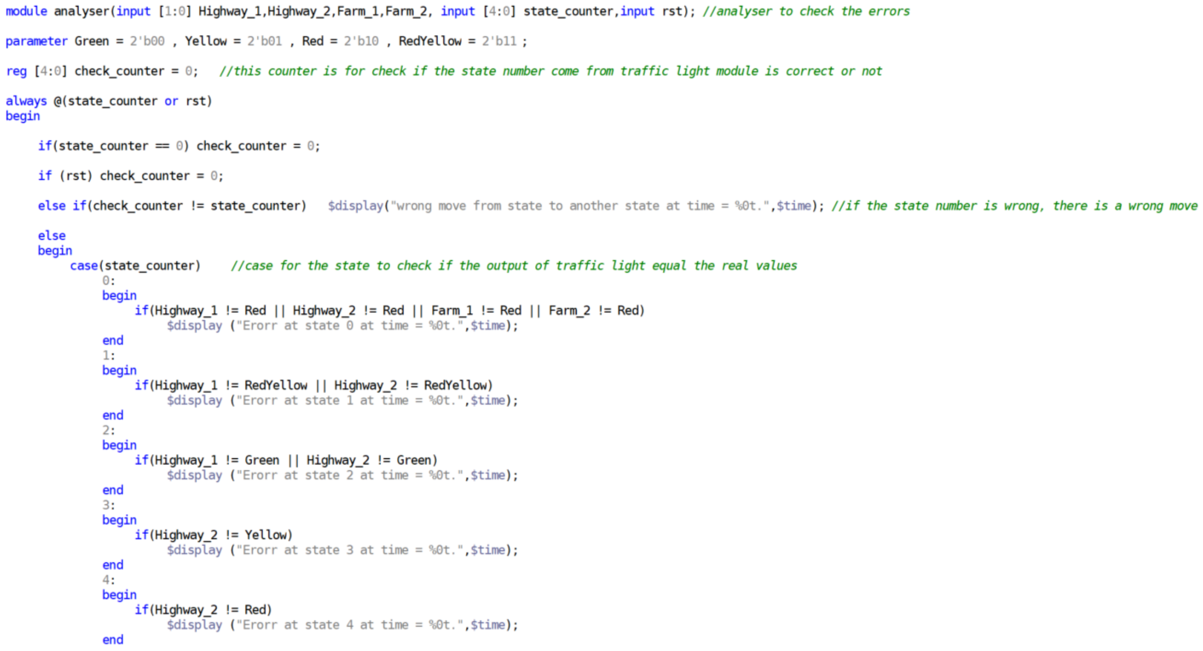
```
            35:
            begin
                Highway_2 = Red;
                state_counter = 4;
            end
            45:
            begin
                Highway_1 = Yellow;
                state_counter = 5;
            end
            47:
            begin
                Highway_1 = Red;
                Highway_2 = Red;
                state_counter = 6;
            end
            48:
            begin
                Farm_1 = RedYellow;
                Farm_2 = RedYellow;
                state_counter = 7;
            end
            50:
            begin
                Farm_1 = Green;
                Farm_2 = Green;
                state_counter = 8;
            end
            65:
            begin
                Farm_2 = Yellow;
                state_counter = 9;
            end
            67:
            begin
                Farm_2 = Red;
                state_counter = 10;
            end
            72:
            begin
                Farm_1 = Yellow;
                Farm_2 = RedYellow;
                state_counter = 11;
            end
```



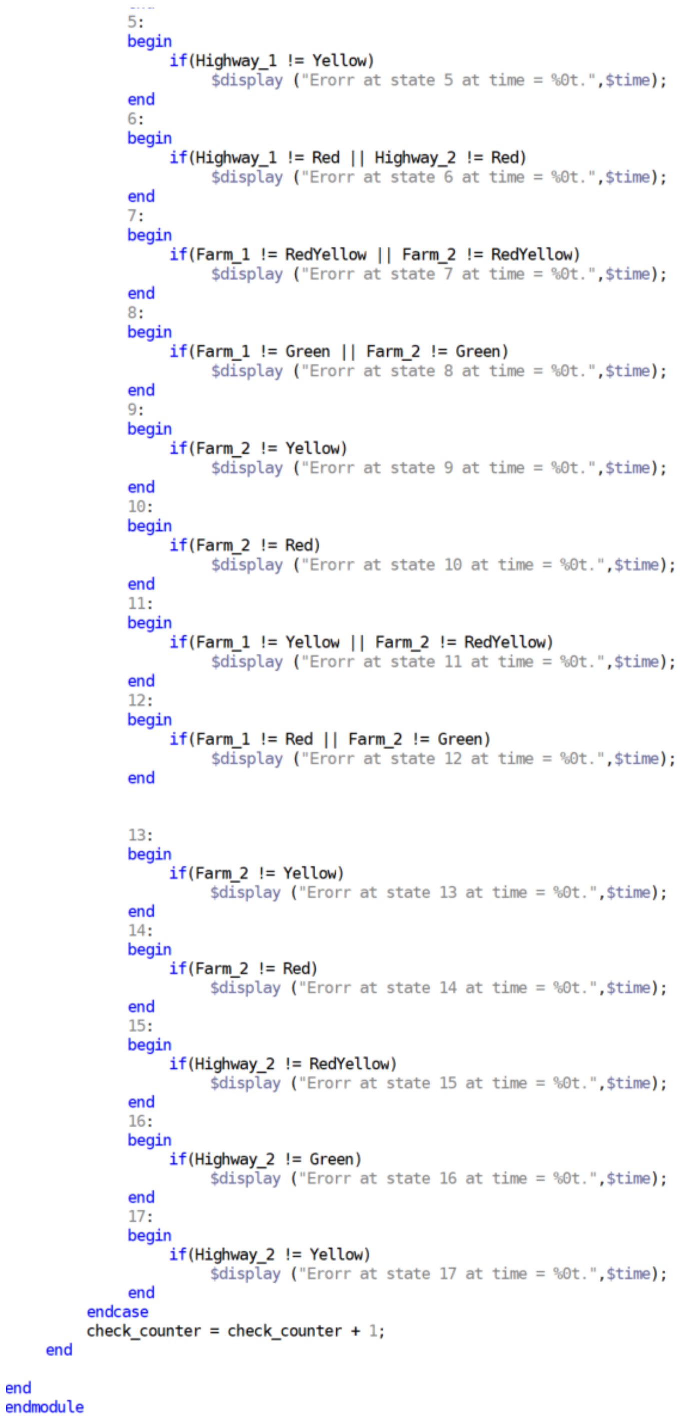
```
            end
            74:
            begin
                Farm_1 = Red;
                Farm_2 = Green;
                state_counter = 12;
            end
            84:
            begin
                Farm_2 = Yellow;
                state_counter = 13;
            end
            86:
            begin
                Farm_2 = Red;
                state_counter = 14;
            end
            87:
            begin
                Highway_2 = RedYellow;
                state_counter = 15;
            end
            89:
            begin
                Highway_2 = Green;
                state_counter = 16;
            end
            104:
            begin
                Highway_2 = Yellow;
                state_counter = 17;
            end
        endcase
        counter = counter + 1;
    end

end
endmodule
```



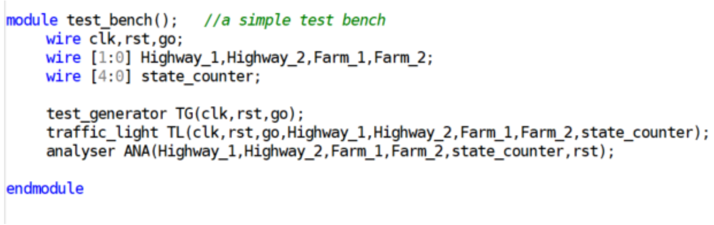
```
module test_generator(output reg clk,rst,go);  //the generetor output is the input of traffic light. and it control the values
    initial begin      //the clk have a rise edge in every  10 time units
        clk = 0;
        repeat(1500)
        #5 clk = ~clk;
    end
    initial begin      //try to freez the system by set the values of go = 0 several times
        go = 1;
        repeat(8)
        #200 go = ~go;
    end
    initial begin       //try to reset the system by set the values of rst = 1 several times
        rst = 0;
        #1700 rst = ~rst;
        #100 rst = ~rst;
    end

endmodule
```



```
module analyser(input [1:0] Highway_1,Highway_2,Farm_1,Farm_2, input [4:0] state_counter,input rst); //analyser to check the errors

parameter Green = 2'b00 , Yellow = 2'b01 , Red = 2'b10 , RedYellow = 2'b11 ;

reg [4:0] check_counter = 0;   //this counter is for check if the state number come from traffic light module is correct or not

always @(state_counter or rst)
begin

    if(state_counter == 0) check_counter = 0;

    if (rst) check_counter = 0;

    else if(check_counter != state_counter)   $display("wrong move from state to another state at time = %0t.",$time); //if the state number is wrong, there is a wrong move

    else
    begin
        case(state_counter)    //case for the state to check if the output of traffic light equal the real values
            0:
            begin
                if(Highway_1 != Red || Highway_2 != Red || Farm_1 != Red || Farm_2 != Red)
                    $display ("Erorr at state 0 at time = %0t.",$time);
            end
            1:
            begin
                if(Highway_1 != RedYellow || Highway_2 != RedYellow)
                    $display ("Erorr at state 1 at time = %0t.",$time);
            end
            2:
            begin
                if(Highway_1 != Green || Highway_2 != Green)
                    $display ("Erorr at state 2 at time = %0t.",$time);
            end
            3:
            begin
                if(Highway_2 != Yellow)
                    $display ("Erorr at state 3 at time = %0t.",$time);
            end
            4:
            begin
                if(Highway_2 != Red)
                    $display ("Erorr at state 4 at time = %0t.",$time);
            end
```



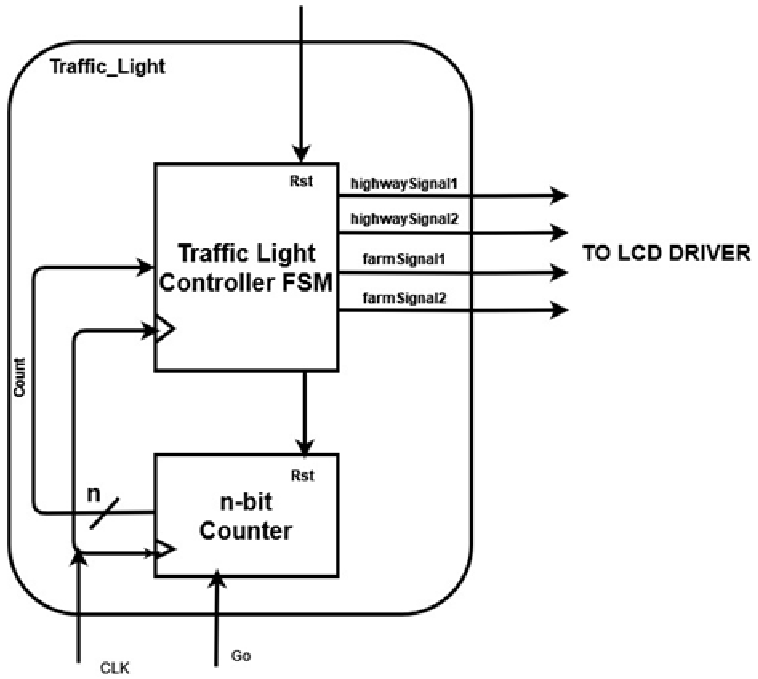
```
            5:
            begin
                if(Highway_1 != Yellow)
                    $display ("Erorr at state 5 at time = %0t.",$time);
            end
            6:
            begin
                if(Highway_1 != Red || Highway_2 != Red)
                    $display ("Erorr at state 6 at time = %0t.",$time);
            end
            7:
            begin
                if(Farm_1 != RedYellow || Farm_2 != RedYellow)
                    $display ("Erorr at state 7 at time = %0t.",$time);
            end
            8:
            begin
                if(Farm_1 != Green || Farm_2 != Green)
                    $display ("Erorr at state 8 at time = %0t.",$time);
            end
            9:
            begin
                if(Farm_2 != Yellow)
                    $display ("Erorr at state 9 at time = %0t.",$time);
            end
            10:
            begin
                if(Farm_2 != Red)
                    $display ("Erorr at state 10 at time = %0t.",$time);
            end
            11:
            begin
                if(Farm_1 != Yellow || Farm_2 != RedYellow)
                    $display ("Erorr at state 11 at time = %0t.",$time);
            end
            12:
            begin
                if(Farm_1 != Red || Farm_2 != Green)
                    $display ("Erorr at state 12 at time = %0t.",$time);
            end


            13:
            begin
                if(Farm_2 != Yellow)
                    $display ("Erorr at state 13 at time = %0t.",$time);
            end
            14:
            begin
                if(Farm_2 != Red)
                    $display ("Erorr at state 14 at time = %0t.",$time);
            end
            15:
            begin
                if(Highway_2 != RedYellow)
                    $display ("Erorr at state 15 at time = %0t.",$time);
            end
            16:
            begin
                if(Highway_2 != Green)
                    $display ("Erorr at state 16 at time = %0t.",$time);
            end
            17:
            begin
                if(Highway_2 != Yellow)
                    $display ("Erorr at state 17 at time = %0t.",$time);
            end
        endcase
        check_counter = check_counter + 1;
    end

end
endmodule
```



```
module test_bench();   //a simple test bench
    wire clk,rst,go;
    wire [1:0] Highway_1,Highway_2,Farm_1,Farm_2;
    wire [4:0] state_counter;

    test_generator TG(clk,rst,go);
    traffic_light TL(clk,rst,go,Highway_1,Highway_2,Farm_1,Farm_2,state_counter);
    analyser ANA(Highway_1,Highway_2,Farm_1,Farm_2,state_counter,rst);

endmodule
```

-Design philosophy.

Traffic Light Module:



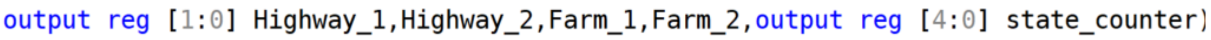
-Inputs:

In this module I put three input :

```
input clk,rst,go,
```

- Clk: the clock value change from 0 to 1 and reverse, when it change from 0 to 1 the system well work and change the other values.
- Rst: some time it will be 0 or 1, when it 1 the system will back to state zero ,else it will continue.
- Go: when go value = 1 the system will work normally, but when go = 0 the system values will freeze and not change until go = 1.

-Outputs:

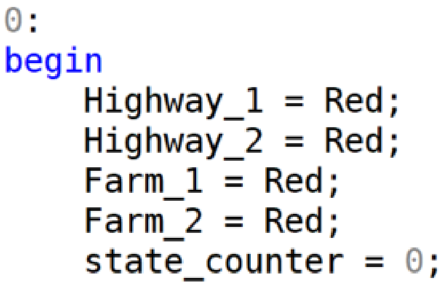


```
output reg [1:0] Highway_1,Highway_2,Farm_1,Farm_2,output reg [4:0] state_counter)
```

The first four outputs values when changes depends on the states values the this table:

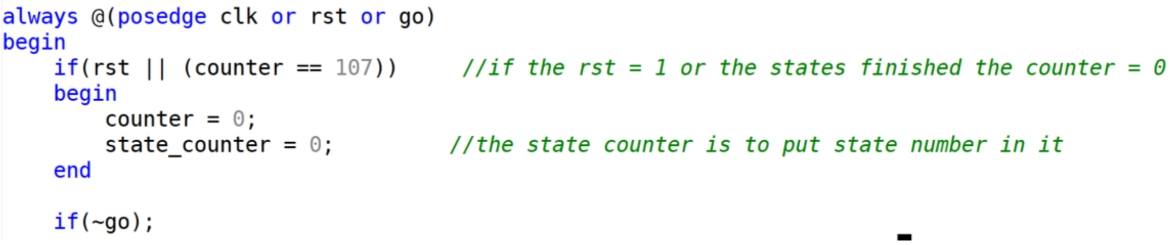
State	Highway TL1	Highway TL2	Farm TL1	Farm TL2	Delay [Sec]
S0	Red	Red	Red	Red	1
S1	Red-Yellow	Red-Yellow	Red	Red	2
S2	Green	Green	Red	Red	30

And the state counter value will be the number of state that the counter arrive:



```
0:
begin
    Highway_1 = Red;
    Highway_2 = Red;
    Farm_1 = Red;
    Farm_2 = Red;
    state_counter = 0;
```

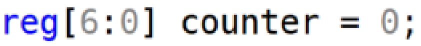
-The always and if statements:



```
always @(posedge clk or rst or go)
begin
    if(rst || (counter == 107))     //if the rst = 1 or the states finished the counter = 0
    begin
        counter = 0;
        state_counter = 0;         //the state counter is to put state number in it
    end

    if(~go);
```

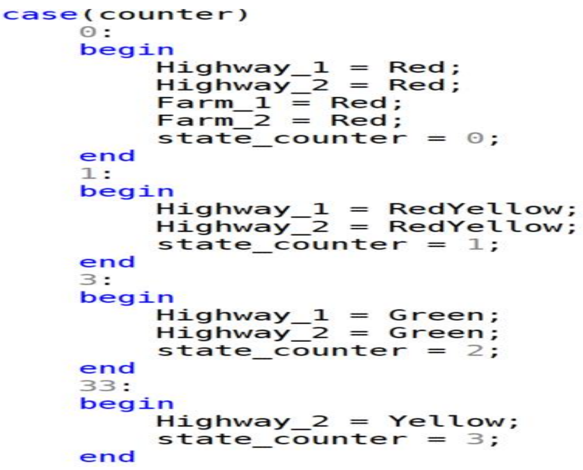
- The system values will be change if the positive edge of the clock come or if the values if go or rst changes.
- In the if statement if the rst = 1 or the states finished the counters will equal 0, and if the go = 1 the system will not do anything.

-The Main Counter:



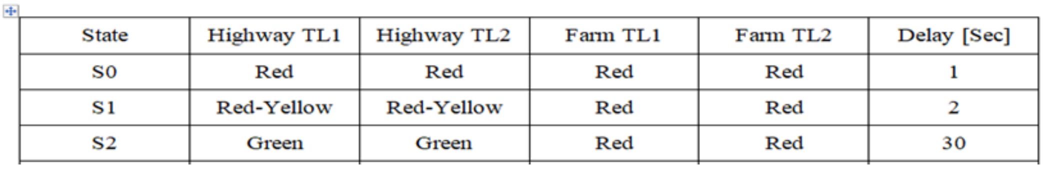
```
reg[6:0] counter = 0;
```

This counter counts the delay between the states, and when state delay time come ,the values of the traffic light will change depends on the state values.



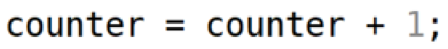
```
case(counter)
    0:
    begin
        Highway_1 = Red;
        Highway_2 = Red;
        Farm_1 = Red;
        Farm_2 = Red;
        state_counter = 0;
    end
    1:
    begin
        Highway_1 = RedYellow;
        Highway_2 = RedYellow;
        state_counter = 1;
    end
    3:
    begin
        Highway_1 = Green;
        Highway_2 = Green;
        state_counter = 2;
    end
    33:
    begin
        Highway_2 = Yellow;
        state_counter = 3;
    end
```

The counter value is increase one by one, and when it arrive a delay value in this table:



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

The values of traffic lights will changes., then in every positive edge clock the counter will be (counter + 1):



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

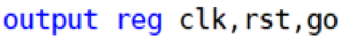
The generator Module:

This module generate the clk, rst and go values to the traffic light module.

-Inputs:

There is no input to the generator, because its generate the results.

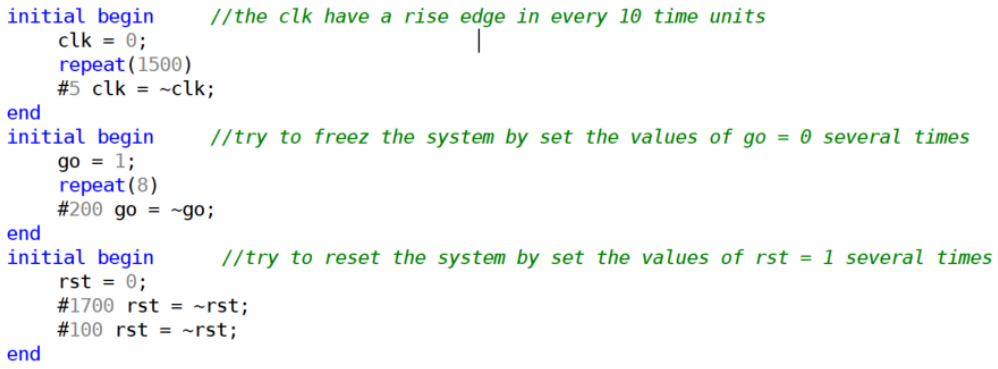
-Outputs:



```
output reg clk,rst,go
```

This three output value changes every certain period of time.

- The clk: it change every 5 ns and repeat the change for 1500 time.
- The go: it changes 8 time each 200 ns to check the system set the system freeze.
- The rst: it changes just one time at time 1700ns for 100ns to reset the system.

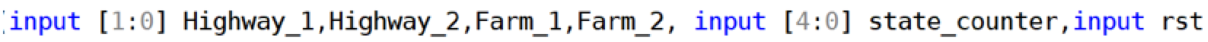


```
initial begin     //the clk have a rise edge in every 10 time units
    clk = 0;
    repeat(1500)
    #5 clk = ~clk;
end
initial begin     //try to freez the system by set the values of go = 0 several times
    go = 1;
    repeat(8)
    #200 go = ~go;
end
initial begin      //try to reset the system by set the values of rst = 1 several times
    rst = 0;
    #1700 rst = ~rst;
    #100 rst = ~rst;
end
```

The Analyzer Module:

This module analyze the results of the traffic light module, and check if it true or not and print error message if there is any errors.

-Inputs:



```
input [1:0] Highway_1,Highway_2,Farm_1,Farm_2, input [4:0] state_counter,input rst
```

The first 4 inputs: its inputs from the traffic light module and its results, and the analyzer check if these results true or false results.

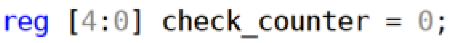
The state counter : is input from traffic light module, and the value of it is the state that the first 4 results come from it, and the analyzer check if there is no wrong move from state to state.

The rst : its input from traffic light module to reset the analyzer module when the traffic reset.

-Outputs:

The output of these module will be on the console, by display statements.

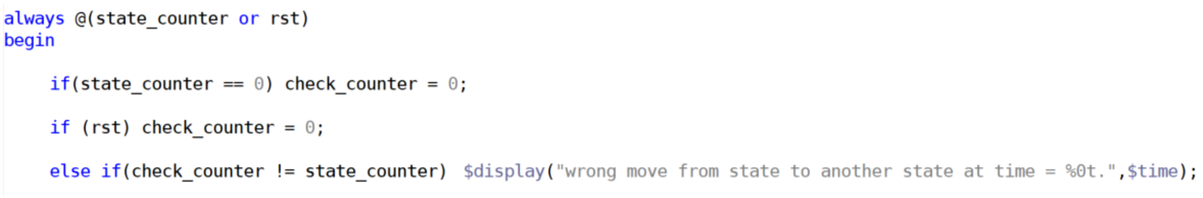
-The Check Counter:



```
reg [4:0] check_counter = 0;
```

It’s a counter to check if the state value that come from traffic is true or false, and the value of the check counter will be (check_counter + 1) every change in the state counter value.

-Always statement an if statements:

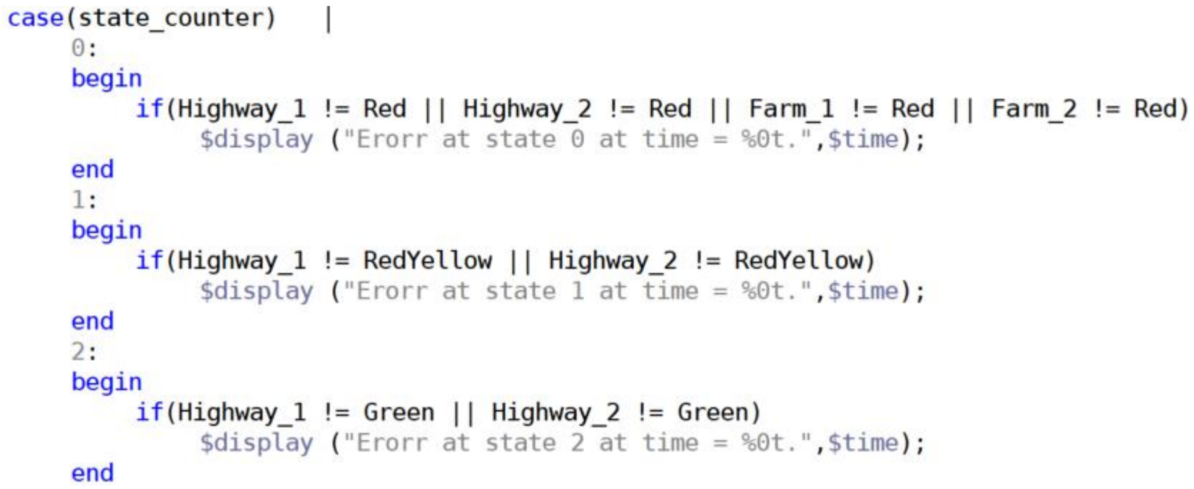


```
always @(state_counter or rst)
begin

    if(state_counter == 0) check_counter = 0;

    if (rst) check_counter = 0;

    else if(check_counter != state_counter)  $display("wrong move from state to another state at time = %0t.",$time);
```

The values inside the always statement changes when the values of the state counter or the reset change.

In the if statement we put the value of check counter to 0 when the state come from the traffic light is 0 and when the traffic light reset, and the check if the (state counter = check counter), if they does not equal the analyser will print the error message and else of that the system will start Comparing.

The Comparison:

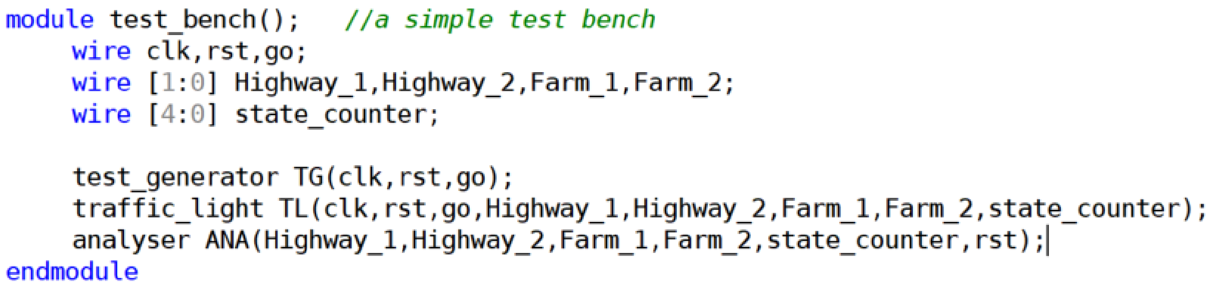


```
case(state_counter)
    0:
    begin
        if(Highway_1 != Red || Highway_2 != Red || Farm_1 != Red || Farm_2 != Red)
            $display ("Erorr at state 0 at time = %0t.",$time);
    end
    1:
    begin
        if(Highway_1 != RedYellow || Highway_2 != RedYellow)
            $display ("Erorr at state 1 at time = %0t.",$time);
    end
    2:
    begin
        if(Highway_1 != Green || Highway_2 != Green)
            $display ("Erorr at state 2 at time = %0t.",$time);
    end
```

The module start comparing between the results come from the traffic light and the true results and check if there is any wrong values then print the error message.

-The Test Bench Module:

You just called the modules and wired the inputs and outputs to the modules.

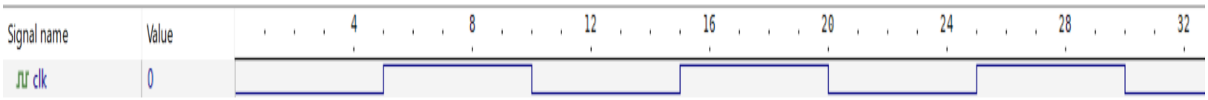


```
module test_bench();   //a simple test bench
    wire clk,rst,go;
    wire [1:0] Highway_1,Highway_2,Farm_1,Farm_2;
    wire [4:0] state_counter;

    test_generator TG(clk,rst,go);
    traffic_light TL(clk,rst,go,Highway_1,Highway_2,Farm_1,Farm_2,state_counter);
    analyser ANA(Highway_1,Highway_2,Farm_1,Farm_2,state_counter,rst);
endmodule
```

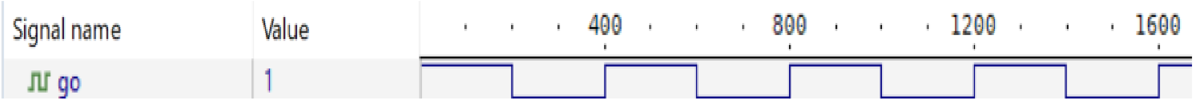
The Results

-For Generator:

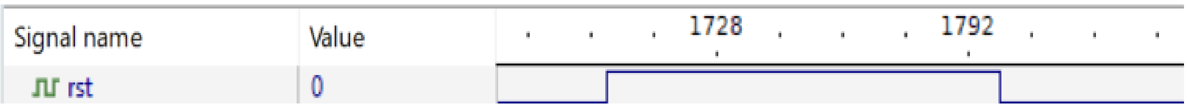
clock changes every 5ns from (1 to 0) or (0 to 1) , every 10 ns equals one unit time of the states counter.



go changes 8 time every 200ns to be 0 or 1.

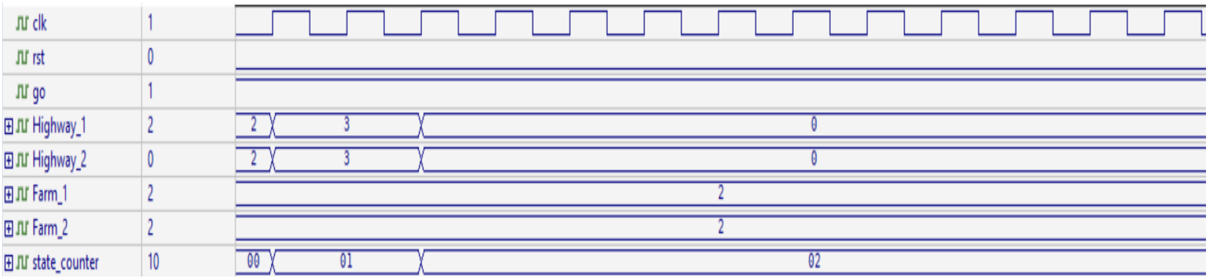


rst just change one time at time 1700 to be 1 and in time 1800 will be 0.



-For Traffic light:

1- In the normal case:

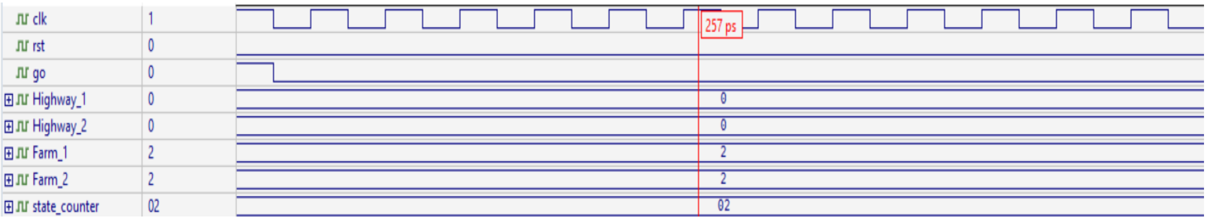


We see in the normal case that the that values of the high way in farm way are true an the same of the value in the table

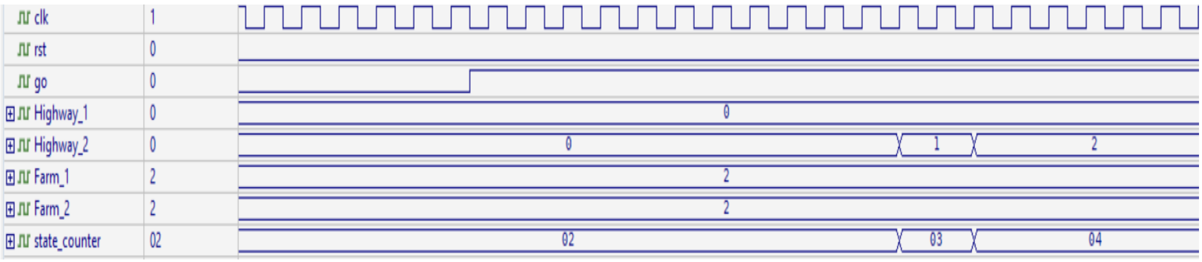
State	Highway TL1	Highway TL2	Farm TL1	Farm TL2	Delay [Sec]
S0	Red	Red	Red	Red	1
S1	Red-Yellow	Red-Yellow	Red	Red	2
S2	Green	Green	Red	Red	30

And the value of go = 1 ,the system will not freeze , rst = 0 the system will not reset.

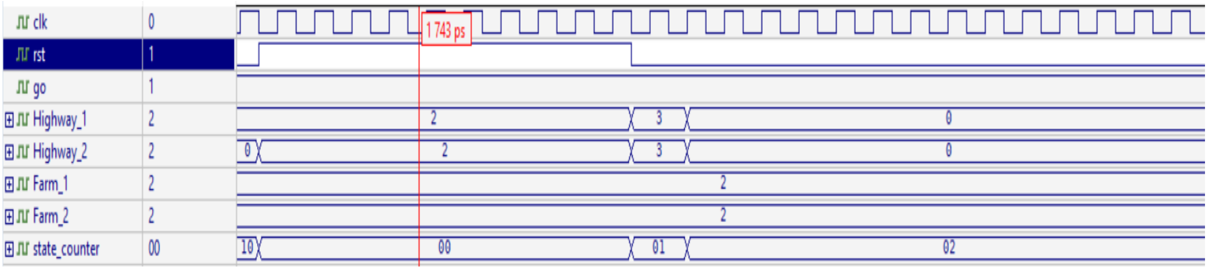
2- when (go = 0):



When the value of go = 0 , the values of the traffic light will freeze and don’t change until the value of go be 1 like this:



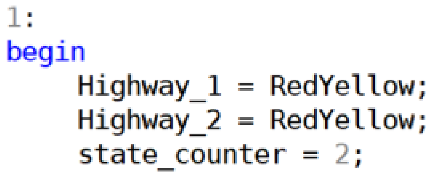
3- when (rst = 1):



When the value of rst = 1 , the values of traffic lights will be (red,2) for all, because it will back to the state zero , and when the rst back to zero the state will be state 1.

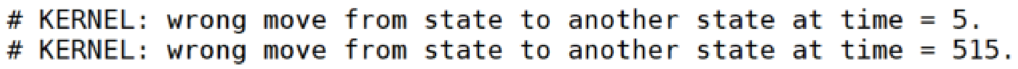
-For Analyzer:

wrong move from state to the next state, in this case the analyzer check counter will not equal the state counter then the analyzer will print error message, I try to make the state number of state 1 equals 2 to have an error.



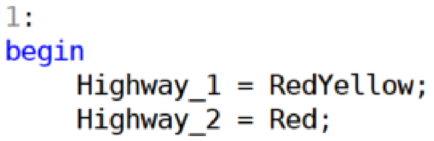
```
1:
begin
     Highway_1 = RedYellow;
     Highway_2 = RedYellow;
     state_counter = 2;
```

and this the output:



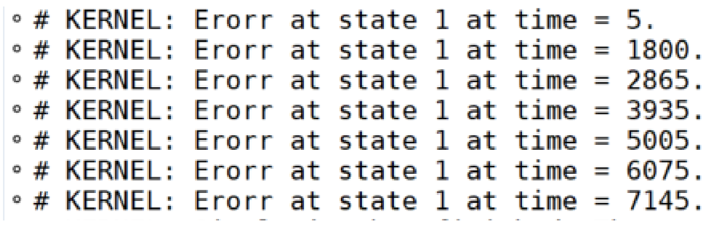
```
# KERNEL: wrong move from state to another state at time = 5.
# KERNEL: wrong move from state to another state at time = 515.
```

wrong value of the traffic light in some cases, in this case the analyzer check if the values of traffic light not equal the true values, and print an error message, I try to put a wrong value of high way 2 traffic light .



```
1:
begin
     Highway_1 = RedYellow;
     Highway_2 = Red;
```

And this is the output:



```
◦ # KERNEL: Erorr at state 1 at time = 5.
◦ # KERNEL: Erorr at state 1 at time = 1800.
◦ # KERNEL: Erorr at state 1 at time = 2865.
◦ # KERNEL: Erorr at state 1 at time = 3935.
◦ # KERNEL: Erorr at state 1 at time = 5005.
◦ # KERNEL: Erorr at state 1 at time = 6075.
◦ # KERNEL: Erorr at state 1 at time = 7145.
```

Conclusion and Future works

This project is a good project to represent the traffic lights in real life, by organizing the times between the four signals and organizing the daily between each of the cases that appear on the traffic lights and the other case, as well as the possibility of making a freeze for the system in some exceptional cases such as the passage of an ambulance and the possibility of making a restart system in the event of a system error.

I suggest in this system and with the development of science that sensors be added to each light signal so that it reduces the possibility of opening a light signal for a long time while not tamping cars waiting at this light as I would like there to be a timer to be with each light signal so that drivers can know how much time is left Until the signal becomes green, and a special button is added for pedestrians, so that it can be pressed by the pedestrian if he wants to cross, and there should also be a possibility for there to be two green lights to reduce waiting times.