

**Faculty of Engineering & Technology**

**Electrical & Computer Engineering Department**

**Advanced Digital System Design ENCS3310**

**Project Report**

**Prepared by:**

Adam Nassan 1202076

**Instructor:**

Dr. Abdallatif Abuissa

**Section:** 1

**Date:** 27/1/2023

المحتويات

[Brief introduction and background 5](#_Toc125754512)

[Design philosophy 7](#_Toc125754513)

[Defining design variables 7](#_Toc125754514)

[Usage of reset and go 7](#_Toc125754515)

[The states logic 8](#_Toc125754516)

[Design Test Bench 11](#_Toc125754517)

[Results 12](#_Toc125754518)

[Conclusion and future works 15](#_Toc125754519)

[Appendix 16](#_Toc125754520)

[Figure 1:Highway and Farm Roads 5](#_Toc125754439)

[Figure 2:Variables Define 7](#_Toc125754440)

[Figure 3: Go,Reset 8](#_Toc125754441)

[Figure 4:States 8](#_Toc125754442)

[Figure 5:Last State 9](#_Toc125754443)

[Figure 6:State-Color 10](#_Toc125754444)

[Figure 7:TestBench1 11](#_Toc125754445)

[Figure 8:TestBench2 11](#_Toc125754446)

[Figure 9 :wholesystem1 12](#_Toc125754447)

[Figure 10:WholeSystem2 12](#_Toc125754448)

[Figure 11:delay1 12](#_Toc125754449)

[Figure 12:delay2 13](#_Toc125754450)

[Figure 13::Go0\_del1 13](#_Toc125754451)

[Figure 14:Go0\_del2 13](#_Toc125754452)

[Figure 15:ResetTo1 14](#_Toc125754453)

[Figure 16:System done 14](#_Toc125754454)

[Table 1:All States of the traffic light 5](#_Toc125754463)

# Brief introduction and background

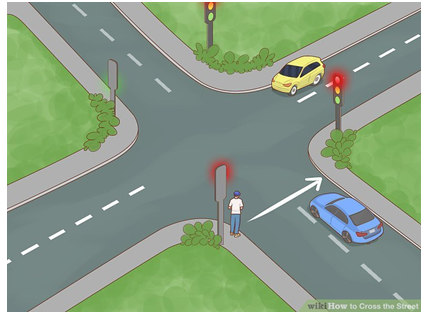


Figure :Highway and Farm Roads

In this project, we will design a traffic light controller for the highway and farm road intersection shown in Figure 1. The traffic light controller outputs four 2-bit signals, highway Signal 1 and highway signal 2 and farm Signal 1 and 2, for the highway road and the farm road, respectively. The following encoding is used for both signals.

Table :All 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 |

We worked in two stages in stage 1, we assigned every color to each state and in stage 2 using the assigned stages we worked in the whole design assigning the appropriate delays and the conditions as required in the design.

I used active-HDL student edition to write and simulate my codes.

# Design philosophy

## Defining design variables

Clock, Reset and Go as regular inputs and the highway and farm traffic lights as 2 bits outputs for the design.

I defined the counter as integer using it later for the delay, and all 17 states as a reg of an array of 17 elements every element represent 4 bit, finally defining the every color to its bit using parameters.

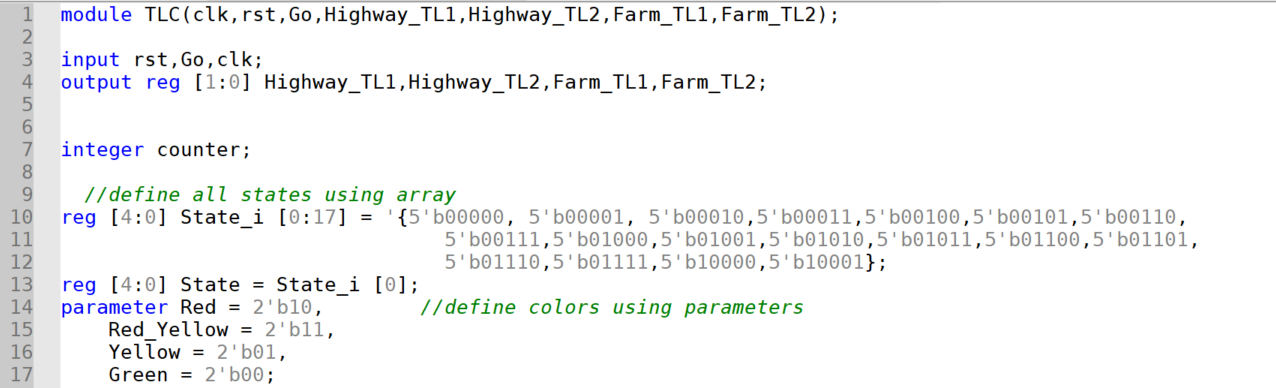


Figure :Variables Define

## Usage of reset and go

Here I used the reset as a condition it the reset = 1 then back to state 0 and reset the counter else check the second condition which is Go.

Here I used the Go as condition if the Go = 1 then start the traffic light states else freeze and do nothing.

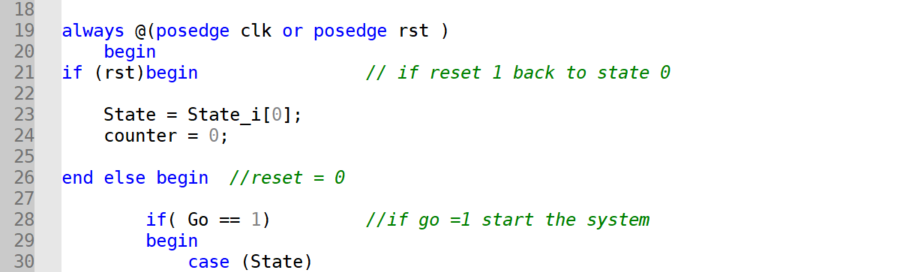


Figure : Go,Reset

## The states logic

Above we initially assigned the first state which is (Red, Red, Red, Red) to the 4 bit reg (State), then I treated counter as a delay when the counter = 1 which is the delay go to state 1 which is the next state, and so on for every state in the design.

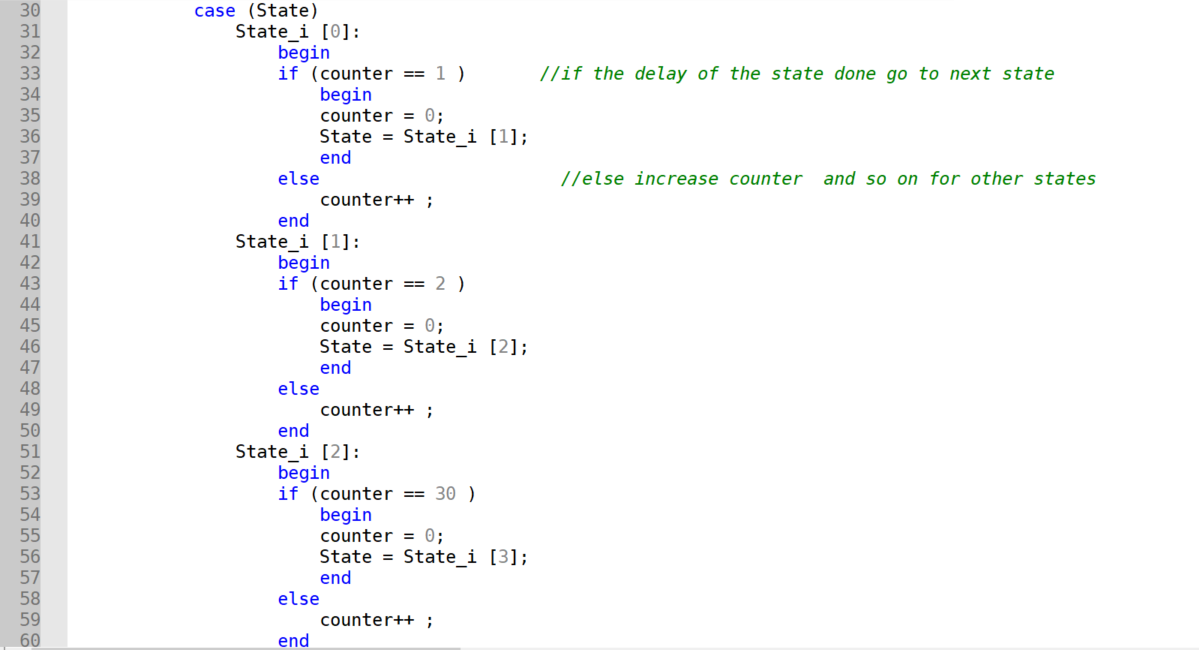


Figure :States

When it reaches the last state it must go back to state 0 again.

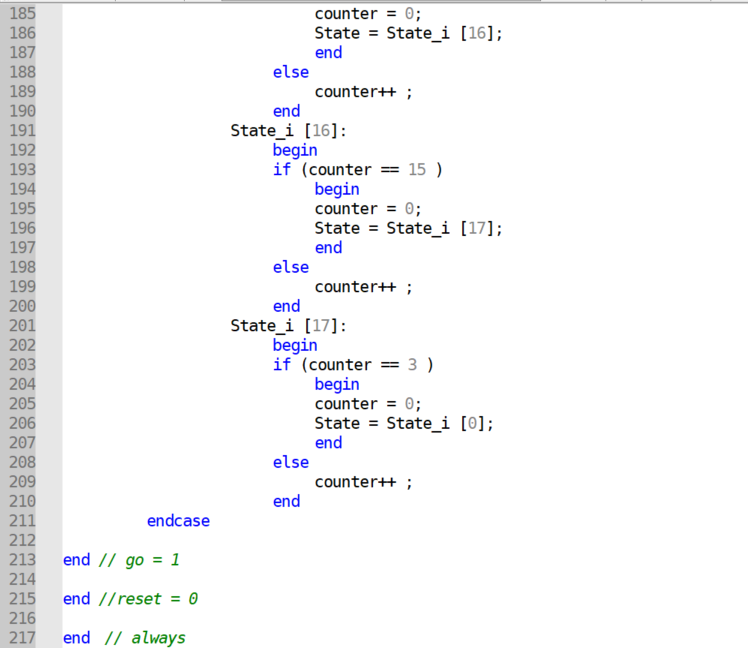


Figure :Last State

Here always for each positive edge clock to assign each traffic light color for its appropriate state as an example🡪state 0 to (Red, Red, Red, Red) as below:

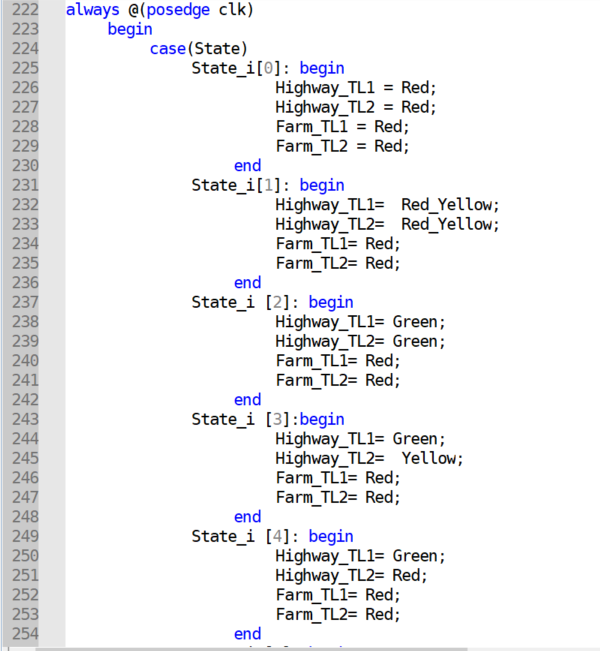


Figure :State-Color

## Design Test Bench

After this I built a test bench for this module to be sure that the system works perfectly, I initially tested (reset) by assigning clock to 0 and reset to 1 repeating 5 times every 5 pico seconds the clock changes, then tested (Go) reset to 0 Go to 0 repeat 10 times, then Go to 1 repeat 150 times to test some states the Go to zero to check the freeze then Go to 1 repeat 250 times to test all the states the reset to 1 to check it the system goes back to its initial state.

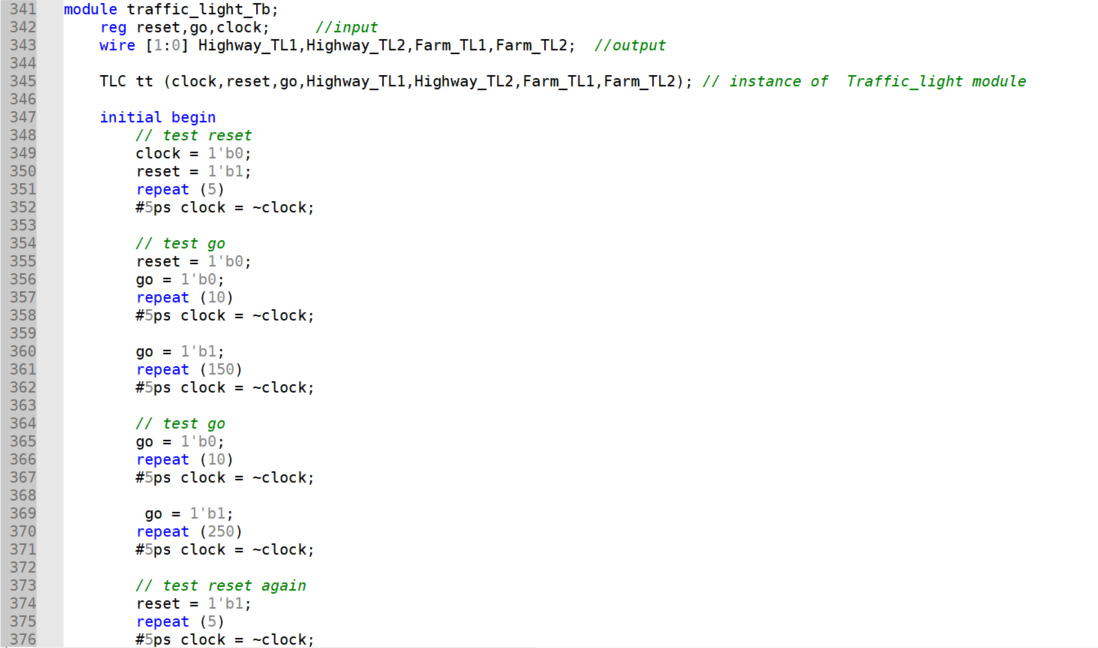


Figure :TestBench1

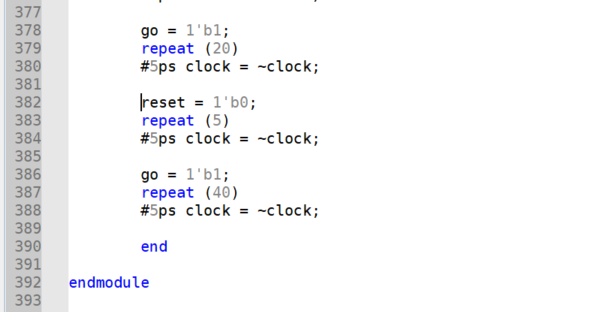


Figure :TestBench2

# Results

Here how the whole system goes between the states when reset 0 and go 1 considering the delays accurately.

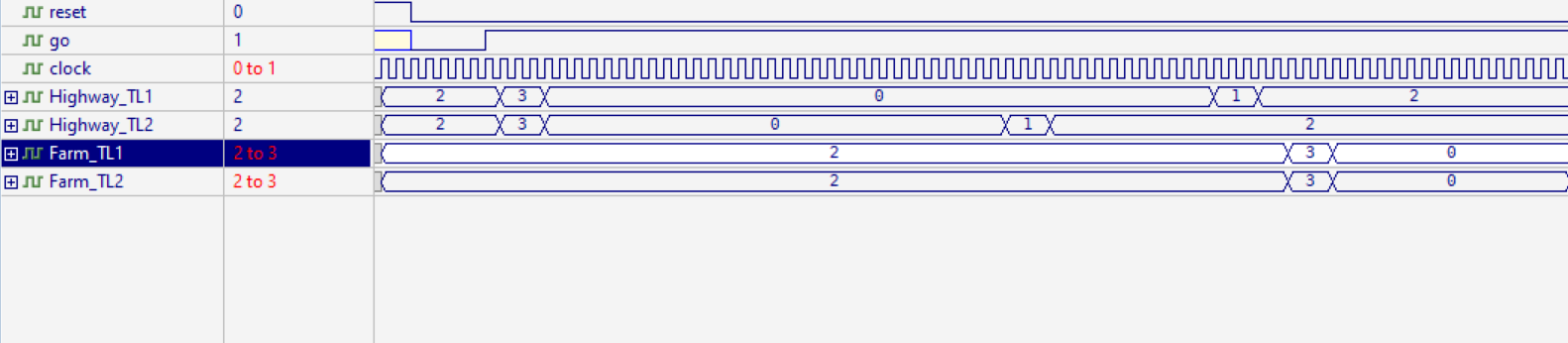


Figure :wholesystem1

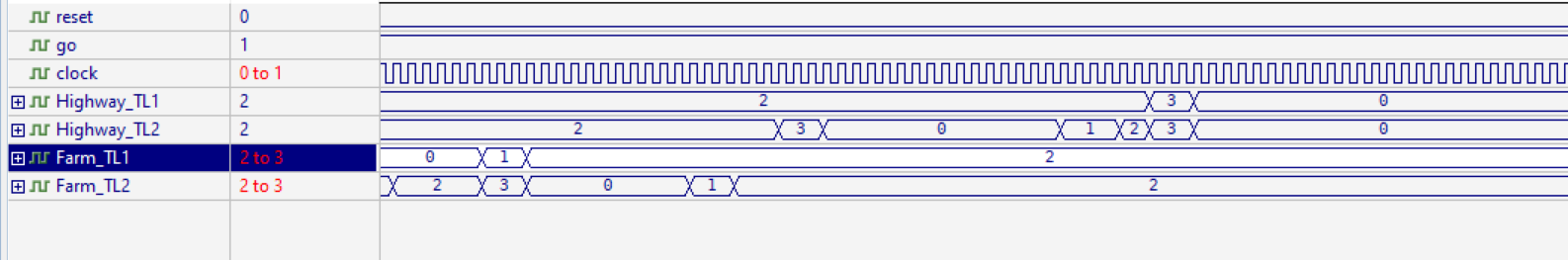


Figure :WholeSystem2

Here state 3 (Green, Yellow, Red, Red) which has delay 2 seconds its delay here is 455-425=30 pico seconds so every second in the table represented as 15 pico seconds in the design.

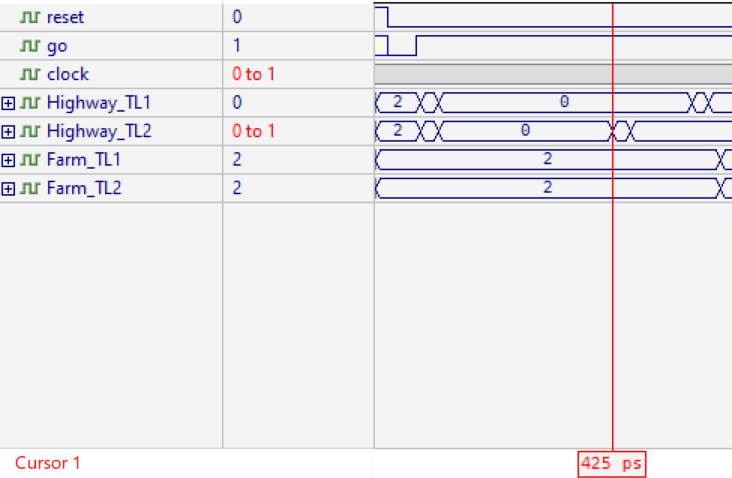


Figure :delay1

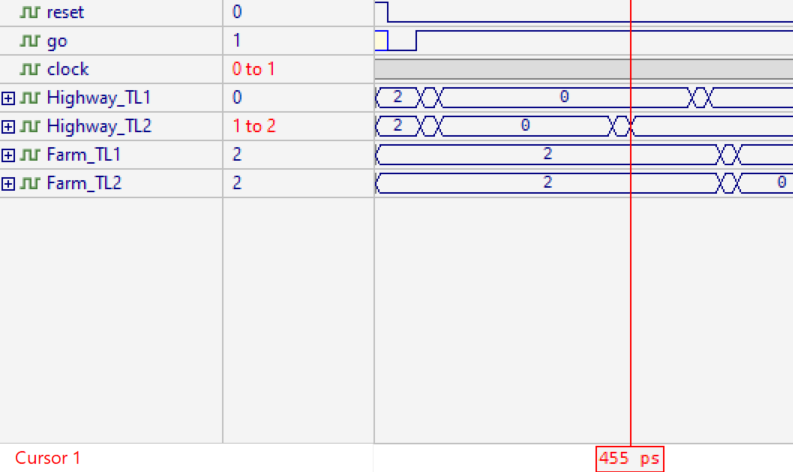


Figure :delay2

Here when Go = 0 we notice the delay in state 0 it must have 15 pico delay (1 second in the table) but it had 50 pico seconds (3.33 seconds in the table) freeze delay.

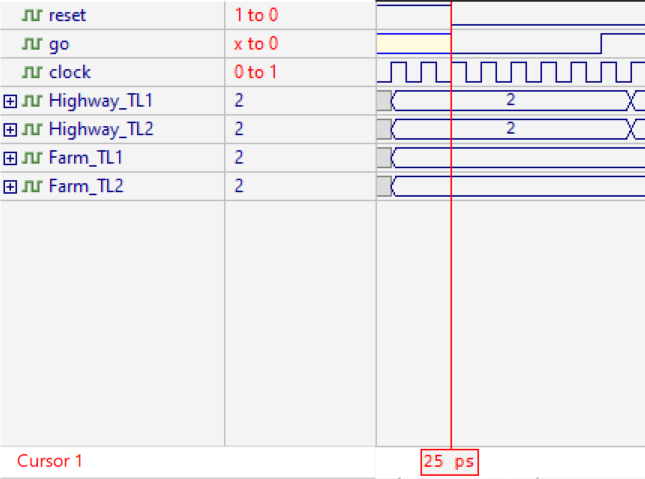


Figure ::Go0\_del1

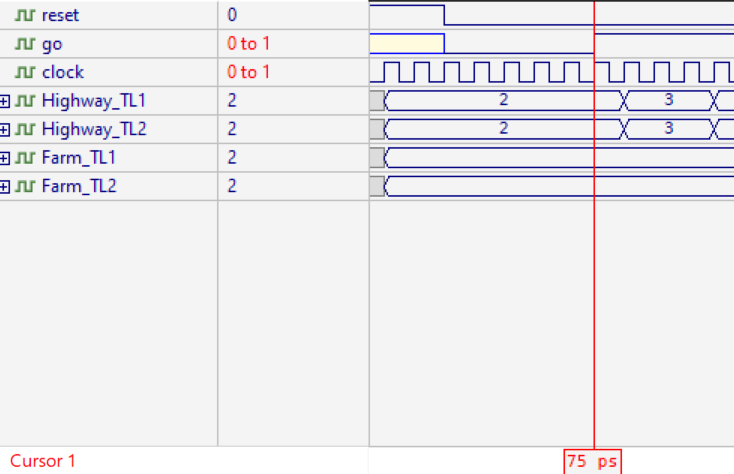


Figure :Go0\_del2

Here when reset goes to 1 during the design, the system goes back from state 8 to state 0.

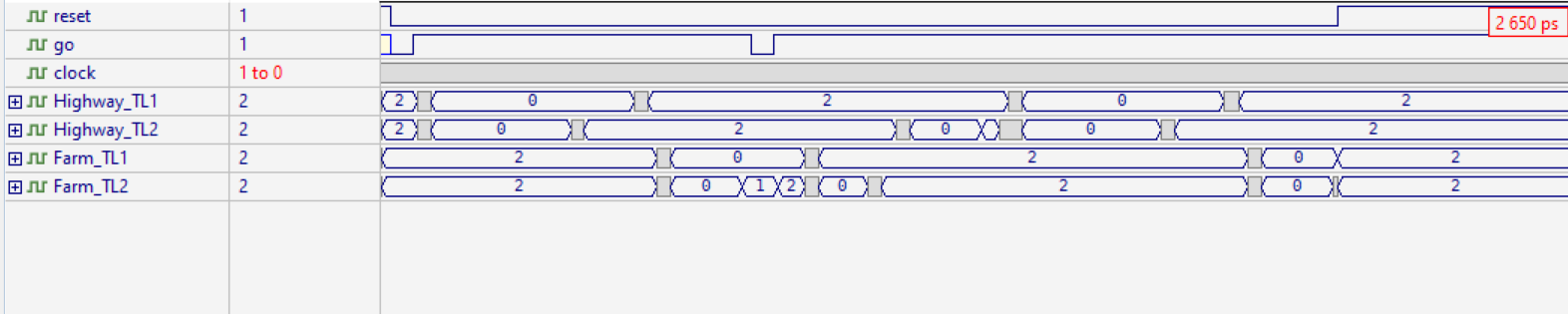


Figure :ResetTo1

Here when the system finish state 17 goes back to state 0 again and continue.

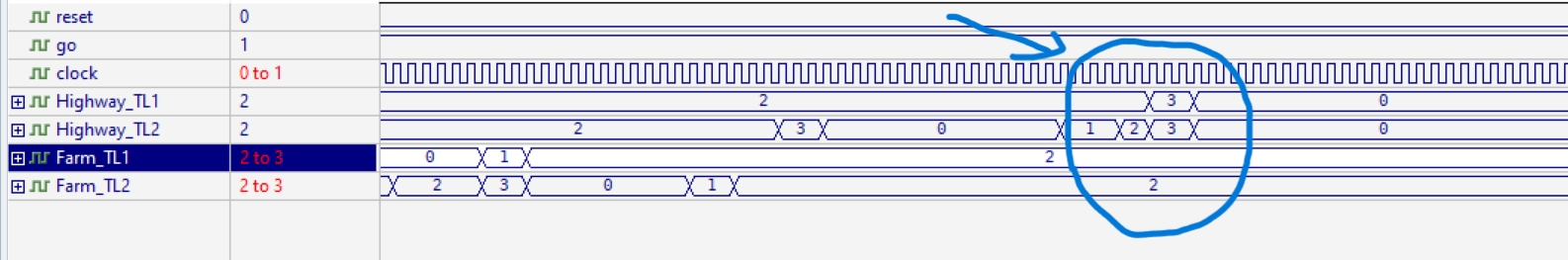


Figure :System done

# Conclusion and future works

After I designed the traffic light system in all its details, I checked all the cases and possibilities through the testbench. After searching for the possibility of errors and glitches, it turned out that the results of my code matched the theoretical results, and I suggest to do this design using function or tasks instead of my way because its less time consuming easier to find errors and easier to add or edit anything you want to change, and finally I became more familiar with the program and the Verilog itself.

# Appendix

module TLC(clk,rst,Go,Highway\_TL1,Highway\_TL2,Farm\_TL1,Farm\_TL2);

input rst,Go,clk;

output reg [1:0] Highway\_TL1,Highway\_TL2,Farm\_TL1,Farm\_TL2;

integer counter;

//define all states using array

reg [4:0] State\_i [0:17] = '{5'b00000, 5'b00001, 5'b00010,5'b00011,5'b00100,5'b00101,5'b00110,

5'b00111,5'b01000,5'b01001,5'b01010,5'b01011,5'b01100,5'b01101,

5'b01110,5'b01111,5'b10000,5'b10001};

reg [4:0] State = State\_i [0];

parameter Red = 2'b10, //define colors using parameters

Red\_Yellow = 2'b11,

Yellow = 2'b01,

Green = 2'b00;

always @(posedge clk or posedge rst )

begin

if (rst)begin // if reset 1 back to state 0

State = State\_i[0];

counter = 0;

end else begin //reset = 0

if( Go == 1) //if go =1 start the system

begin

case (State)

State\_i [0]:

begin

if (counter == 1 ) //if the delay of the state done go to next state

begin

counter = 0;

State = State\_i [1];

end

else //else increase counter and so on for other states

counter++ ;

end

State\_i [1]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [2];

end

else

counter++ ;

end

State\_i [2]:

begin

if (counter == 30 )

begin

counter = 0;

State = State\_i [3];

end

else

counter++ ;

end

State\_i [3]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [4];

end

else

counter++ ;

end

State\_i [4]:

begin

if (counter == 10 )

begin

counter = 0;

State = State\_i [5];

end

else

counter++ ;

end

State\_i [5]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [6];

end

else

counter++ ;

end

State\_i [6]:

begin

if (counter == 1 )

begin

counter = 0;

State = State\_i [7];

end

else

counter++ ;

end

State\_i [7]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [8];

end

else

counter++ ;

end

State\_i [8]:

begin

if (counter == 15 )

begin

counter = 0;

State = State\_i [9];

end

else

counter++ ;

end

State\_i [9]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [10];

end

else

counter++ ;

end

State\_i [10]:

begin

if (counter == 5 )

begin

counter = 0;

State = State\_i [11];

end

else

counter++ ;

end

State\_i [11]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [12];

end

else

counter++ ;

end

State\_i [12]:

begin

if (counter == 10 )

begin

counter = 0;

State = State\_i [13];

end

else

counter++ ;

end

State\_i [13]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [14];

end

else

counter++ ;

end

State\_i [14]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [15];

end

else

counter++ ;

end

State\_i [15]:

begin

if (counter == 2 )

begin

counter = 0;

State = State\_i [16];

end

else

counter++ ;

end

State\_i [16]:

begin

if (counter == 15 )

begin

counter = 0;

State = State\_i [17];

end

else

counter++ ;

end

State\_i [17]:

begin

if (counter == 3 )

begin

counter = 0;

State = State\_i [0];

end

else

counter++ ;

end

endcase

end // go = 1

end //reset = 0

end // always

always @(posedge clk)

begin

case(State)

State\_i[0]: begin

Highway\_TL1 = Red;

Highway\_TL2 = Red;

Farm\_TL1 = Red;

Farm\_TL2 = Red;

end

State\_i[1]: begin

Highway\_TL1= Red\_Yellow;

Highway\_TL2= Red\_Yellow;

Farm\_TL1= Red;

Farm\_TL2= Red;

end

State\_i [2]: begin

Highway\_TL1= Green;

Highway\_TL2= Green;

Farm\_TL1= Red;

Farm\_TL2= Red;

end

State\_i [3]:begin

Highway\_TL1= Green;

Highway\_TL2= Yellow;

Farm\_TL1= Red;

Farm\_TL2= Red;

end

State\_i [4]: begin

Highway\_TL1= Green;

Highway\_TL2= Red;

Farm\_TL1= Red;

Farm\_TL2= Red;

end

State\_i [5]: begin

Highway\_TL1= Yellow;

Highway\_TL2= Red;

Farm\_TL1= Red;

Farm\_TL2= Red;

end

State\_i [6]: begin

Highway\_TL1= Red;

Highway\_TL2=Red;

Farm\_TL1= Red ;

Farm\_TL2= Red;

end

State\_i [7]:begin

Highway\_TL1=Red;

Highway\_TL2=Red;

Farm\_TL1= Red\_Yellow ;

Farm\_TL2= Red\_Yellow ;

end

State\_i [8]: begin

Highway\_TL1=Red;

Highway\_TL2=Red;

Farm\_TL1= Green;

Farm\_TL2= Green;

end

State\_i [9]: begin

Highway\_TL1= Red;

Highway\_TL2= Red;

Farm\_TL1= Green;

Farm\_TL2= Yellow;

end

State\_i [10]: begin

Highway\_TL1=Red;

Highway\_TL2= Red;

Farm\_TL1= Green;

Farm\_TL2= Red;

end

State\_i [11]:begin

Highway\_TL1= Red;

Highway\_TL2 = Red;

Farm\_TL1 = Yellow;

Farm\_TL2 = Red\_Yellow;

end

State\_i [12]: begin

Highway\_TL1 = Red;

Highway\_TL2 = Red;

Farm\_TL1 = Red;

Farm\_TL2 = Green;

end

State\_i [13]: begin

Highway\_TL1 = Red;

Highway\_TL2 = Red;

Farm\_TL1 = Red;

Farm\_TL2 =Yellow;

end

State\_i [14]: begin

Highway\_TL1 = Red;

Highway\_TL2 = Red;

Farm\_TL1 = Red;

Farm\_TL2 = Red;

end

State\_i [15]:begin

Highway\_TL1 = Red;

Highway\_TL2 = Red\_Yellow;

Farm\_TL1 = Red;

Farm\_TL2 = Red;

end

State\_i [16]: begin

Highway\_TL1 = Red;

Highway\_TL2 = Green;

Farm\_TL1 = Red;

Farm\_TL2 = Red;

end

State\_i [17]: begin

Highway\_TL1 = Red;

Highway\_TL2 = Yellow;

Farm\_TL1 = Red;

Farm\_TL2 = Red;

end

endcase

end

endmodule

//1 sec represents by 15ps 2 sec represents by 30ps and so on

//testbench for traffic light system

module traffic\_light\_Tb;

reg reset,go,clock; //input

wire [1:0] Highway\_TL1,Highway\_TL2,Farm\_TL1,Farm\_TL2; //output

TLC tt (clock,reset,go,Highway\_TL1,Highway\_TL2,Farm\_TL1,Farm\_TL2); // instance of Traffic\_light module

initial begin

// test reset

clock = 1'b0;

reset = 1'b1;

repeat (5)

#5ps clock = ~clock;

// test go

reset = 1'b0;

go = 1'b0;

repeat (10)

#5ps clock = ~clock;

go = 1'b1;

repeat (150)

#5ps clock = ~clock;

// test go

go = 1'b0;

repeat (10)

#5ps clock = ~clock;

go = 1'b1;

repeat (250)

#5ps clock = ~clock;

// test reset again

reset = 1'b1;

repeat (5)

#5ps clock = ~clock;

go = 1'b1;

repeat (20)

#5ps clock = ~clock;

reset = 1'b0;

repeat (5)

#5ps clock = ~clock;

go = 1'b1;

repeat (40)

#5ps clock = ~clock;

end

endmodule