SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

College of Engineering and Technology

Department of Electronics and Communication Engineering

18ECC206J - VLSI Design

VI Semester, 2023-2024 (EVEN Semester)

Title of Mini Project : T- Shaped Traffic Light Control using Verilog

Date of Submission : 25/04/2024

Group Number :

Name & Reg Number : 1. Bhoovi Chauhan – RA2111053010005

2. Angad Singh Hoonjan – RA2111053010021

3. Sangeetha S Nair – RA2111053010026

Particulars	Max. Marks	Marks Obtained		
		Register No.	Register No.	Register No.
		RA2111053010005	RA2111053010021	RA2111053010026
Design Code	25			
Demo Verification & Viva	10			
Project Report	05			
Total	**40**			

REPORT VERIFICATION

Staff Name : Dr. Maria Jossy A

Signature :

T-SHAPED TRAFFIC LIGHT CONTROL USING VERILOG

Objective: To deal with a basic design of a Traffic Light Control for T-Shaped Road and test the output using Verilog HDL.

Software Detail:
For our project we are using the following software:

- Xilinx 7.1e ISE
- ModelSim-Altera

The project will be executed using Verilog HDL (Hardware Description Language). Some of the benefits of using Verilog HDL are:

- Easy to write
- Easy to understand as it is similar to C program.
- Easier to learn compared to VHSIC HDL.

Abstract:
Traffic control is a challenging problem in many cities. This is due to the large number of vehicles and the high dynamics of the traffic system. Poor traffic systems are the big reason for accidents, time losses. In this method of approach, it will reduce the waiting time of the vehicles at traffic signals. The hardware design has been developed using Verilog Hardware Description Language (HDL) programming.

Verilog designing is hardware descriptive language, the name itself suggest that it deals with the hardware designing and simulation. Basically, it becomes very difficult to mount the various electronic components on breadboard or PCB circuit. It also takes too much time for the simulation and sometimes many errors occur because of improper connection of components onto the circuit.

And thus, to overcome this factor hardware descriptive language comes into conclusion. we can code the process using Verilog and we can mount it on a circuit or just upload it to the circuit accordingly so that circuit will work as according to the code we have written.

HDL language is often used for sequential circuits like shift register, combinational logic circuit like adder, subtractor etc. Basically it describes the digital systems like microprocessor or a memory. Whatever design that is described in HDL are independent, it has its unique state of work, very much easy to simulate, designing and debugging, and very useful than schematics, especially for large circuits thus, to overcome difficulties or problems to design the circuits manually with breadboard and PCB, use of Verilog designing in this complex world is increasing a way better.

This project deals with a basic design of a T - Shaped Road for traffic light control. The output of system has been tested using Xilinx.

Problem Statement and Working Methodology:

We have considered a T - shaped road and heavy load of traffic is present over this road where we must control it using traffic signal. Six cases have been considered here and analyzed using state diagram and state table.

The directions, M1, MT, M2, S, that is been considered for analysis of our problem is shown in the figure 1.

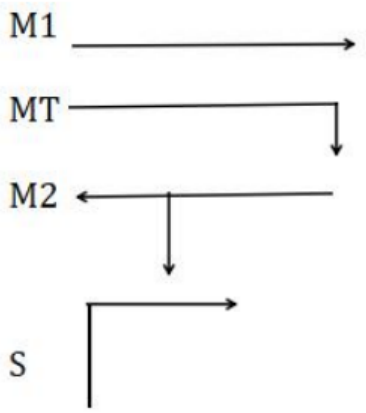

Figure 1 : Directions chart

- Green light indicates that there is no traffic and there is easy flow of vehicles in that route/direction.
- Red light indicates that there is a traffic jam and that route is blocked for the vehicles to move and,
- Yellow light indicates that the route has medium flow of vehicles

The problem statement is explained in the figure 2. Six states, S1, S2, S3, S4, S5, S6 are taken into consideration and state diagram (Figure 3), state table (Figure 1) is made using the following logic explained in the figure 2.

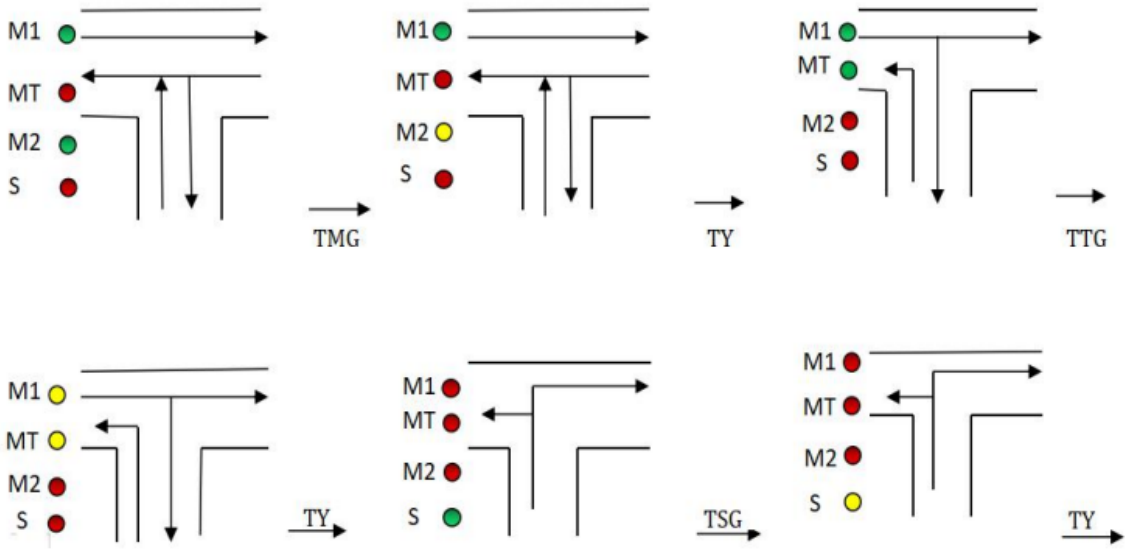

Figure 2 : Problem statement (Logic)

Time delays for changing from one state to another (shown in Figure 2) is considered as, TMG (from S1 to S2), TY (from S2 to S3), TTG (from S3 to S4), TY (from S4 to S5), TSG (from S5 to S6) and TY (from S6 to S1) and the cycle continues.
In Figure 3, the time delays are considered as follows:

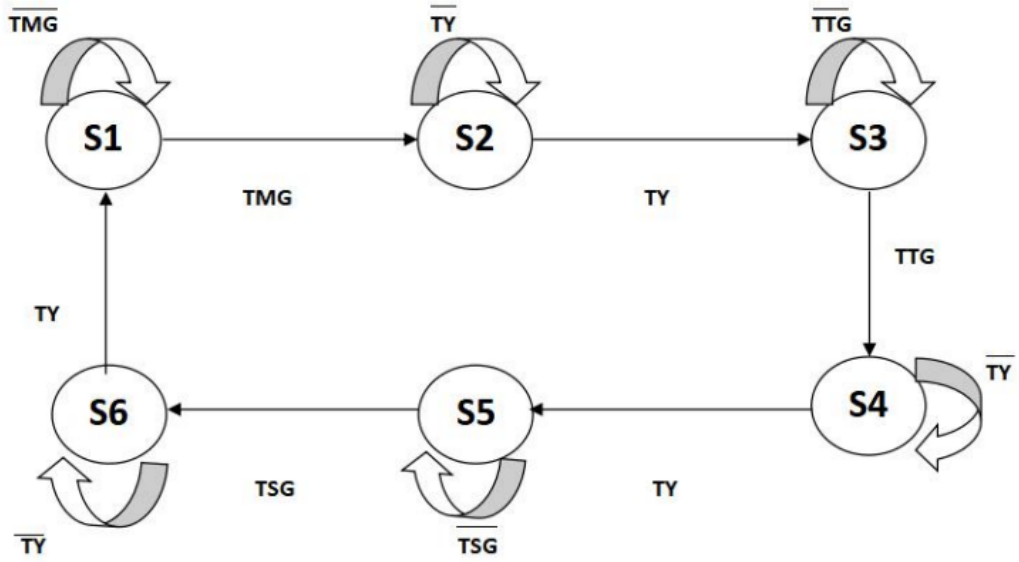

Figure 3 : State Diagram

- TMG = 7 seconds
- TY = 2 seconds
- TTG = 5 seconds
- TSG = 3 seconds

Until TMG seconds, the signal will remain in S1 state, and after TMG seconds, it will move to S2 state. Until TY seconds it will remain in S2 state and after TY seconds, it will move to S3 state, and so on. After TY seconds, in state S6, it will go back to S1 state and the cycle continues.
The state table for the problem statement is shown below in Table 1.

PRESENT STATE ABC	INPUT	NEXT STATE A*B*C*	S T	M1 RYG	M2 RYG	MT RYG	S RYG
000	-	001	1	000	000	000	000
001	$\overline{TMG}$ TMG	001 010	0 1	001	001	100	100
010	$\overline{TY}$ TY	010 011	0 1	001	010	100	100

011	$\overline{TTG}$ TTG	011 100	0 1	001	100	001	100
100	$\overline{TY}$ TY	100 101	0 1	010	100	010	100
101	$\overline{TSG}$ TSG	101 110	0 1	100	100	100	001
110	$\overline{TY}$ TY	110 001	0 1	100	100	100	010
111	-	000	0	000	000	000	000

Table 1 : State Table

In Table 1, R = Red, Y = Yellow and, G = Green.

ST = State Transition; A, B and C are considered as the present state.
The state table is made, considering the Logic diagram/problem statement given in Figure 2.
From the Figure 2, Figure 3, and Table 1, we can infer that:

In state S1(001); M1 = GREEN, implies, RYG value = 001,
MT = RED, implies, RYG value = 100,
M2 = GREEN, implies, RYG value = 001 and,
S = RED, implies, RYG value = 100.
After TMG seconds,
In state S2(010); M1 = GREEN, implies, RYG value = 001,
MT = RED, implies, RYG value = 100,
M2 = YELLOW, implies, RYG value = 010 and,
S = RED, implies, RYG value = 100.
After TY seconds,
In state S3(011); M1 = GREEN, implies, RYG value = 001,
MT = GREEN, implies, RYG value = 001,
M2 = RED, implies, RYG value = 100 and,
S = RED, implies, RYG value = 100.
After TTG seconds,
In state S4(100); M1 = YELLOW, implies, RYG value = 010,
MT = YELLOW, implies, RYG value = 010,
M2 = RED, implies, RYG value = 100 and,
S = RED, implies, RYG value = 100.

After TY seconds,
In state S5(101); M1 = RED, implies, RYG value = 100,
MT = RED, implies, RYG value = 100,
M2 = RED, implies, RYG value = 100 and,
S = GREEN, implies, RYG value = 001.
After TSG seconds,
In state S6(110); M1 = RED, implies, RYG value = 100,
MT = RED, implies, RYG value = 100,
M2 = RED, implies, RYG value = 100 and,
S = YELLOW, implies, RYG value = 010.
And after S6 state, the cycle repeats and goes to S1 state.

RTL Code:

```
`timescale 1ns / 1ps
module Traffic_Light_Controller(
	input clk,rst,
	output reg [2:0]light_M1,
	output reg [2:0]light_S,
	output reg [2:0]light_MT,
	output reg [2:0]light_M2,
	output reg [3:0]count,
	output reg [2:0]ps
	);

	parameter S1=1, S2=2, S3=3, S4=4, S5=5, S6=6;
	parameter tmg=7, ty=2, ttg=5, tsg=3;

	always @ (posedge clk or posedge rst)
	begin
		if(rst==1)
			begin
			ps<=S1;
			count<=0;
			end
		else
			case(ps)
				S1:if(count < tmg)
						begin
						ps<=S1;
						count<=count+1;
						end
					else
						begin
						ps<=S2;
```

```
                count<=0;
                end

S2:if(count < ty)
                begin
                ps<=S2;
                count<=count+1;
                end
        else
                begin
                ps<=S3;
                count<=0;
                end

S3:if(count < ttg)
                begin
                ps<=S3;
                count<=count+1;
                end
        else
                begin
                ps<=S4;
                count<=0;
                end

S4:if(count < ty)
                begin
                ps<=S4;
                count<=count+1;
                end
        else
                begin
                ps<=S5;
                count<=0;
                end

S5:if(count < tsg)
                begin
                ps<=S5;
                count<=count+1;
                end
        else
                begin
                ps<=S6;
```

```
                        count<=0;
                        end

                S6:if(count < ty)
                        begin
                        ps<=S6;
                        count<=count+1;
                        end
                    else
                        begin
                        ps<=S1;
                        count<=0;
                        end

                default:ps<=S1;

        endcase
end

always @ (ps)
begin
    case(ps)
        S1:
        begin
            light_M1<=3'b001;
            light_M2<=3'b001;
            light_MT<=3'b100;
            light_S<=3'b100;
        end

        S2:
        begin
            light_M1<=3'b001;
            light_M2<=3'b010;
            light_MT<=3'b100;
            light_S<=3'b100;
        end

        S3:
        begin
            light_M1<=3'b001;
            light_M2<=3'b100;
            light_MT<=3'b001;
            light_S<=3'b100;
```

```
                end

                S4:
                begin
                        light_M1<=3'b010;
                        light_M2<=3'b100;
                        light_MT<=3'b010;
                        light_S<=3'b100;
                end

                S5:
                begin
                        light_M1<=3'b100;
                        light_M2<=3'b100;
                        light_MT<=3'b100;
                        light_S<=3'b001;
                end

                S6:
                begin
                        light_M1<=3'b100;
                        light_M2<=3'b100;
                        light_MT<=3'b100;
                        light_S<=3'b010;
                end

                default:
                begin
                        light_M1<=3'b000;
                        light_M2<=3'b000;
                        light_MT<=3'b000;
                        light_S<=3'b000;
                end
        endcase
    end
endmodule
```

Testbench:

```
module Traffic_Light_Controller_TB_v;
	// Inputs
	reg clk, rst;

	// Outputs
	wire [2:0] light_M1;
	wire [2:0] light_S;
	wire [2:0] light_MT;
	wire [2:0] light_M2;
	wire [3:0] count;
	wire [2:0] ps;

	Traffic_Light_Controller dut (
		.clk(clk),
		.rst(rst),
		.light_M1(light_M1),
		.light_S(light_S),
		.light_MT(light_MT),
		.light_M2(light_M2),
		.count(count),
		.ps(ps)
	);

	initial begin
		rst=0; clk=0;
		#750;
		rst=1;
	end
	always #5 clk=~clk;
endmodule
```

Results:

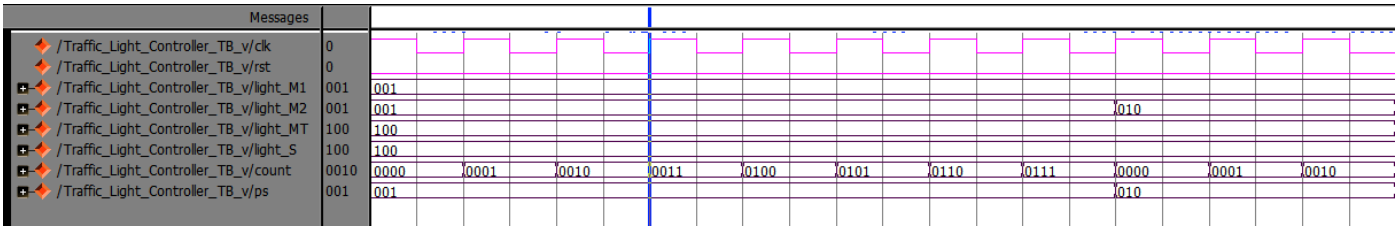

Figure: Simulation waveform for State 1 and 2

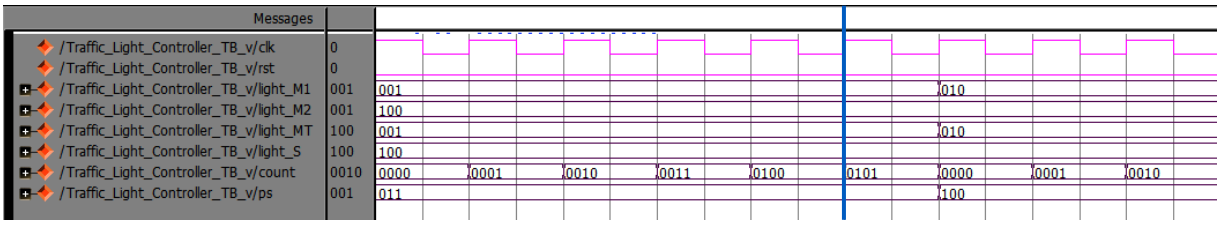

Figure: Simulation waveform for State 3 and 4

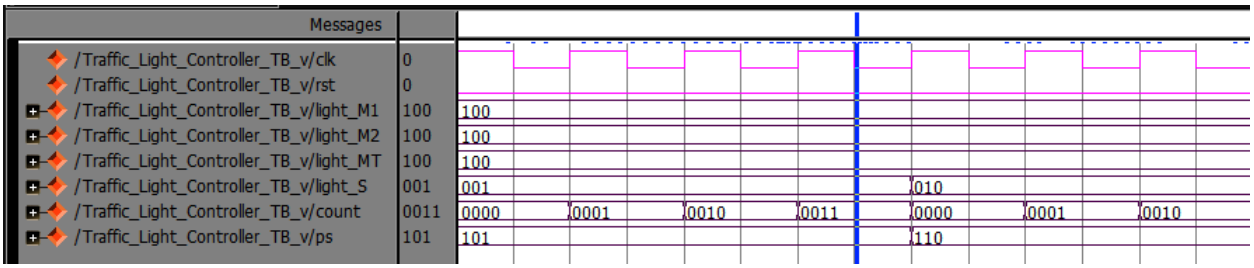

Figure: Simulation waveform for State 5 and 6

Conclusion:

In conclusion, the project presents a solution to the challenges of traffic control through Verilog Hardware Description Language (HDL) programming. By employing HDL, the design process becomes more efficient, eliminating the complexities and errors associated with manual circuit assembly. The implementation of a T-Shaped Road traffic light control system demonstrates the effectiveness of this approach. HDL enables easy simulation, debugging, and scalability, offering a superior alternative to traditional circuit design methods. The successful testing of the system using Xilinx underscores the practicality and potential impact of HDL in optimizing traffic management systems.

Group Photo: