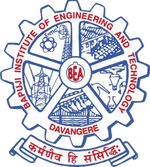
**Bapuji Educational Association ®**

**Bapuji Institute of Engineering and Technology**

**Davangere-577004**



Bachelor of Engineering and Technology, Fifth semester,

E&C E Course Project Report

On

**Verilog HDL Project Assignment**

**For**

**Verilog HDL (18EC56)**

**Course Project Associates of 5th Sem**

**4BD20ECO60 S Suhas**

**4BD20EC066 Shreyas P**

**4BD20EC090 Vyshnavi H K**

**Course Project Guide Head of the Department**

**Dr. Leela G.H Dr. G S Sunitha**

M Tech, PhD M Tech(DEAC),PhD, MIS, FIETE, FIE

**Department of Electronic & Communication Engineering**

**2022-2023**

**ACKNOWLEDGEMENT**

I take this opportunity to express my profound gratitude and deep

regards to my guide Dr. Leela G.H for their exemplary guidance, monitoring and constant encouragement throughout the course of this project.

The blessing, help and guidance given by her time to time shall carry me a long way in the journey of life on which I am about to embark.

My thanks and appreciations also go to my team members in developing the project and people who have willingly helped me out with their abilities.

***Three Way Traffic Controller***

**Objective:**

Our objective is to demonstrate the working process of Three-way traffic controller for a T-Intersection using Verilog HDL code, Where in T-Intersection roads we studied what are the possible ways a vehicle can move.

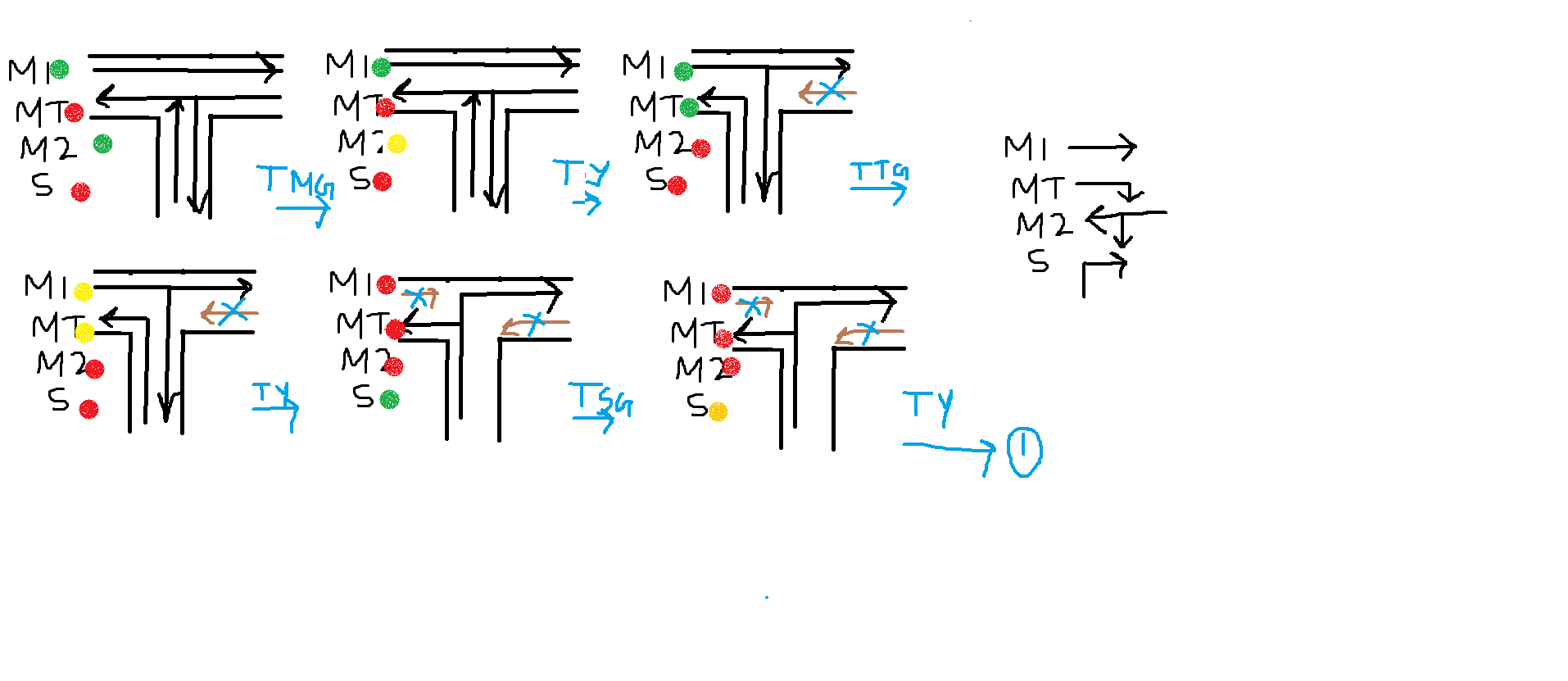
Based on the study, we observed that the vehicle which moves in straight path have no problem (path M1), And in Second path we considered a vehicle which has to make right turn (path MT) where it was bit difficult for the drives because they have to predict the opposite vehicle’s path before making turn, In third path The vehicle is coming towards us it can make left turn or can move straight (path M2) driver has to see if any vehicle is making turn form intersection road or the opposite moving vehicle has to make turn towards the intersection road, The last path is (path S) where the vehicle is coming from the intersection road and makes right turn in this situation the driver has to see the coming towards him and make turn.

Based on the above possible ways of paths, we figured out six cases in which vehicles can move towards there path in easier way by avoiding the accidents due to Intersection of roads.

**Principle of three-way traffic controller :**

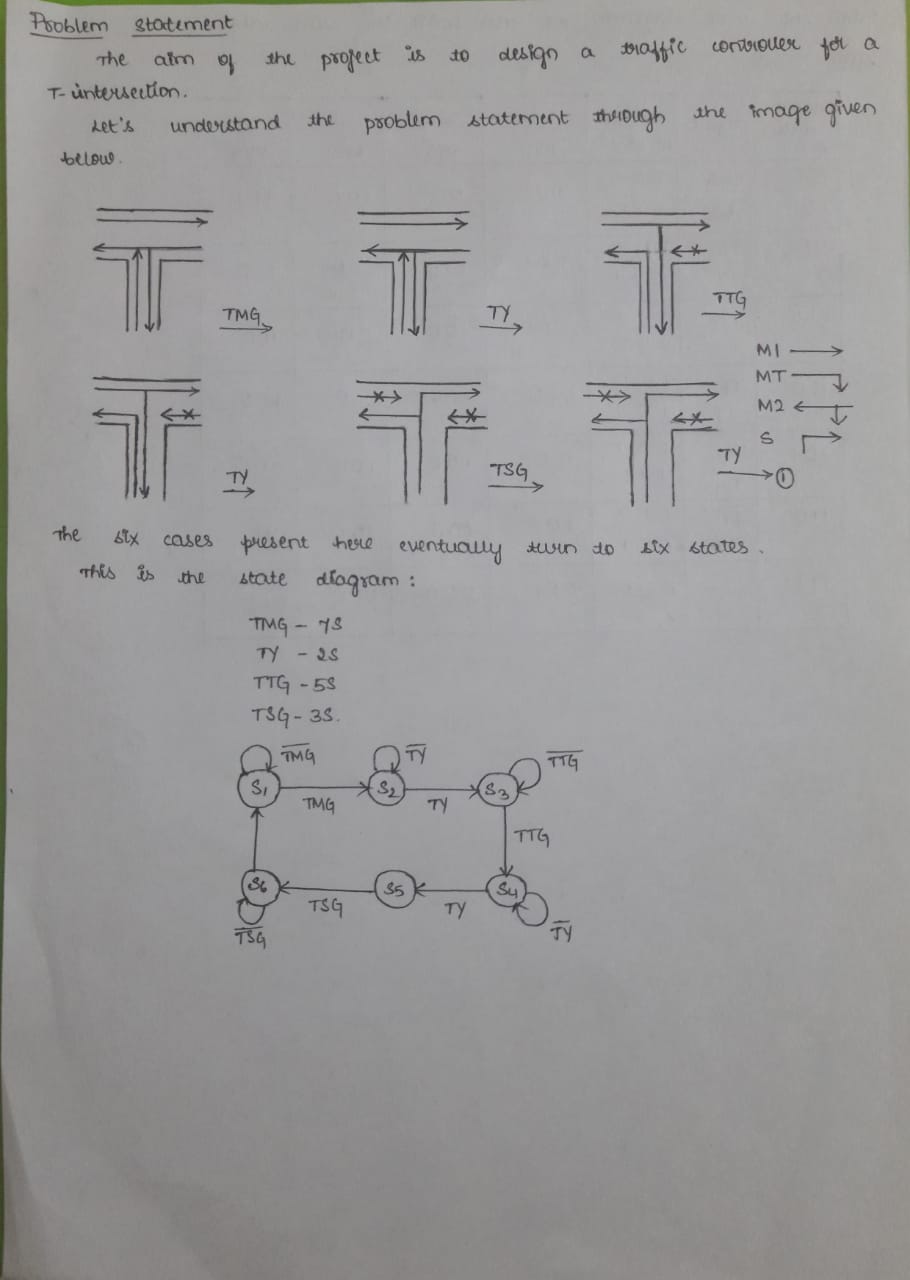
The aim of the project is to design a traffic controller for a T-intersection.

Let’s understand the problem statement through the image given below.

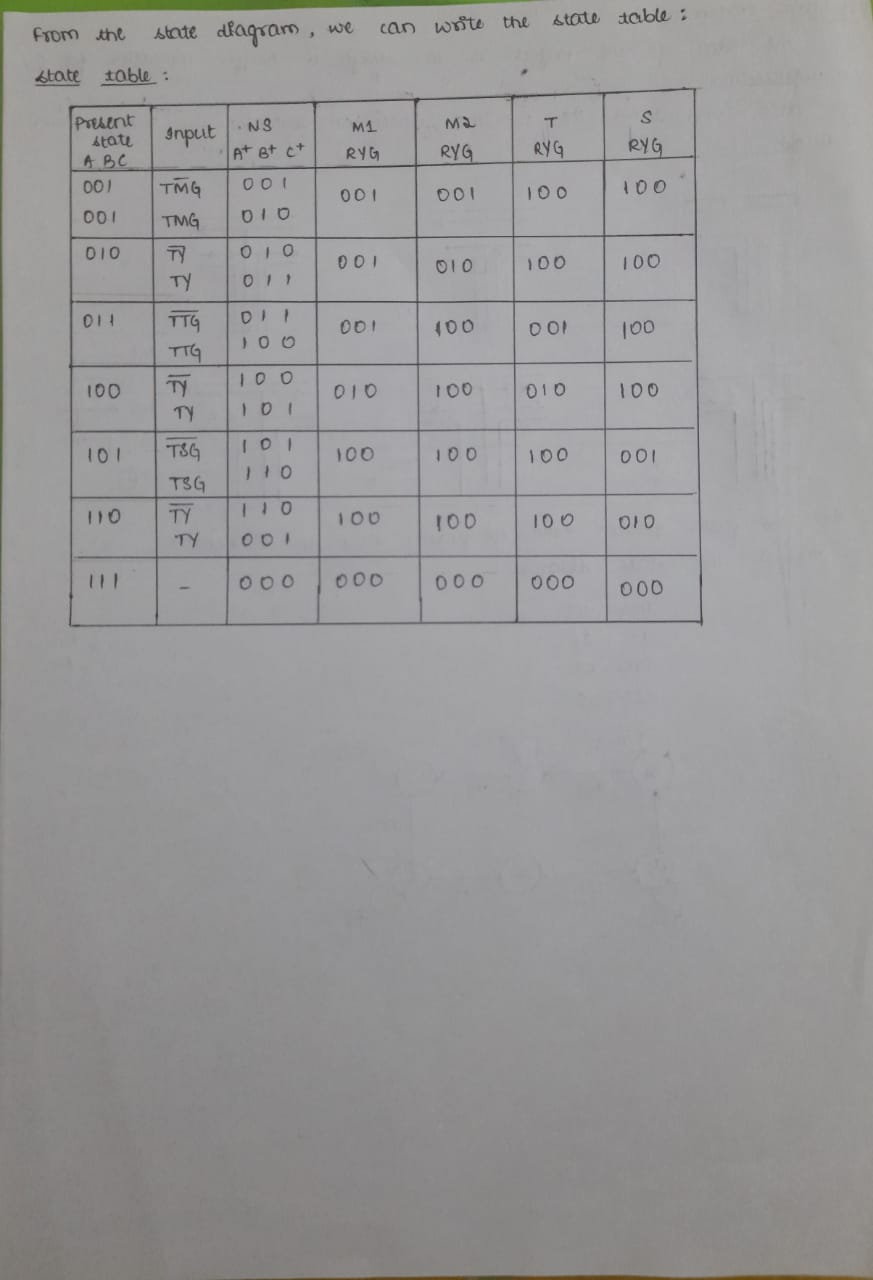


The six cases present here eventually turn to the six states.

This is the state diagram:



From the state diagram we for the state table:



**ABOUT LANGUAGE USED:**

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixedsignal circuits, as well as in the design of genetic circuits. In 2009, the Verilog standard (IEEE 1364-2005) was merged into the SystemVerilog standard, creating IEEE Standard 1800-2009. Since then, Verilog is officially part of the SystemVerilog language. The current version is IEEE standard 1800-2017.

Hardware description languages such as Verilog are similar to software programming languages because they include ways of describing the propagation time and signal strengths (sensitivity). There are two types of assignment operators; a blocking assignment (=), and a non-blocking (<=) assignment. The non-blocking assignment allows designers to describe a state-machine update without needing to declare and use temporary storage variables. Since these concepts are part of Verilog's language semantics, designers could quickly write descriptions of large circuits in a relatively compact and concise form. At the time of Verilog's introduction (1984), Verilog represented a tremendous productivity improvement for circuit designers who were already using graphical schematic capture software and specially written software programs to document and simulate electronic circuits.

The designers of Verilog wanted a language with syntax similar to the C programming language, which was already widely used in engineering software development. Like C, Verilog is case-sensitive and has a basic preprocessor (though less sophisticated than that of ANSI C/C++). Its control flow keywords (if/else, for, while, case, etc.) are equivalent, and its operator precedence is compatible with C. Syntactic differences include: required bit-widths for variable declarations, demarcation of procedural blocks (Verilog uses begin/end instead of curly braces {}), and many other minor differences. Verilog requires that variables be given a definite size. In C these sizes are inferred from the 'type' of the variable (for instance an integer type may be 8 bits).

A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output, and bidirectional ports. Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order within the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language.

Verilog's concept of 'wire' consists of both signal values (4-state: "1, 0, floating, undefined") and signal strengths (strong, weak, etc.). This system allows abstract modeling of shared signal lines, where multiple sources drive a common net. When a wire has multiple drivers, the wire's (readable) value is resolved by a function of the source drivers and their strengths.

A subset of statements in the Verilog language are synthesizable. Verilog modules that conform to a synthesizable coding style, known as RTL

(register-transfer level), can be physically realized by synthesis software. Synthesis software algorithmically transforms the (abstract) Verilog source into a netlist, a logically equivalent description consisting only of elementary logic primitives (AND, OR, NOT, flip-flops, etc.) that are available in a specific FPGA or VLSI technology. Further manipulations to the netlist ultimately lead to a circuit fabrication blueprint (such as a photo mask set for an ASIC or a bitstream file for an FPGA).

**Software Requirements:**

* xilinx ise 14.7

**VERILOG CODE:**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 09:21:34 01/08/2023

// Design Name:

// Module Name: Traffic\_light\_controller\_TB

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

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

);

parameter S1=0, S2=1, S3 =2, S4=3, S5=4,S6=5;

reg [3:0]count;

reg[2:0] ps;

parameter sec7=7,sec5=5,sec2=2,sec3=3;

always@(posedge clk or posedge rst)

begin

if(rst==1)

begin

ps<=S1;

count<=0;

end

else

case(ps)

S1: if(count<sec7)

begin

ps<=S1;

count<=count+1;

end

else

begin

ps<=S2;

count<=0;

end

S2: if(count<sec2)

begin

ps<=S2;

count<=count+1;

end

else

begin

ps<=S3;

count<=0;

end

S3: if(count<sec5)

begin

ps<=S3;

count<=count+1;

end

else

begin

ps<=S4;

count<=0;

end

S4:if(count<sec2)

begin

ps<=S4;

count<=count+1;

end

else

begin

ps<=S5;

count<=0;

end

S5:if(count<sec3)

begin

ps<=S5;

count<=count+1;

end

else

begin

ps<=S6;

count<=0;

end

S6:if(count<sec2)

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'b100;

end

default:

begin

light\_M1<=3'b000;

light\_M2<=3'b000;

light\_MT<=3'b000;

light\_S<=3'b000;

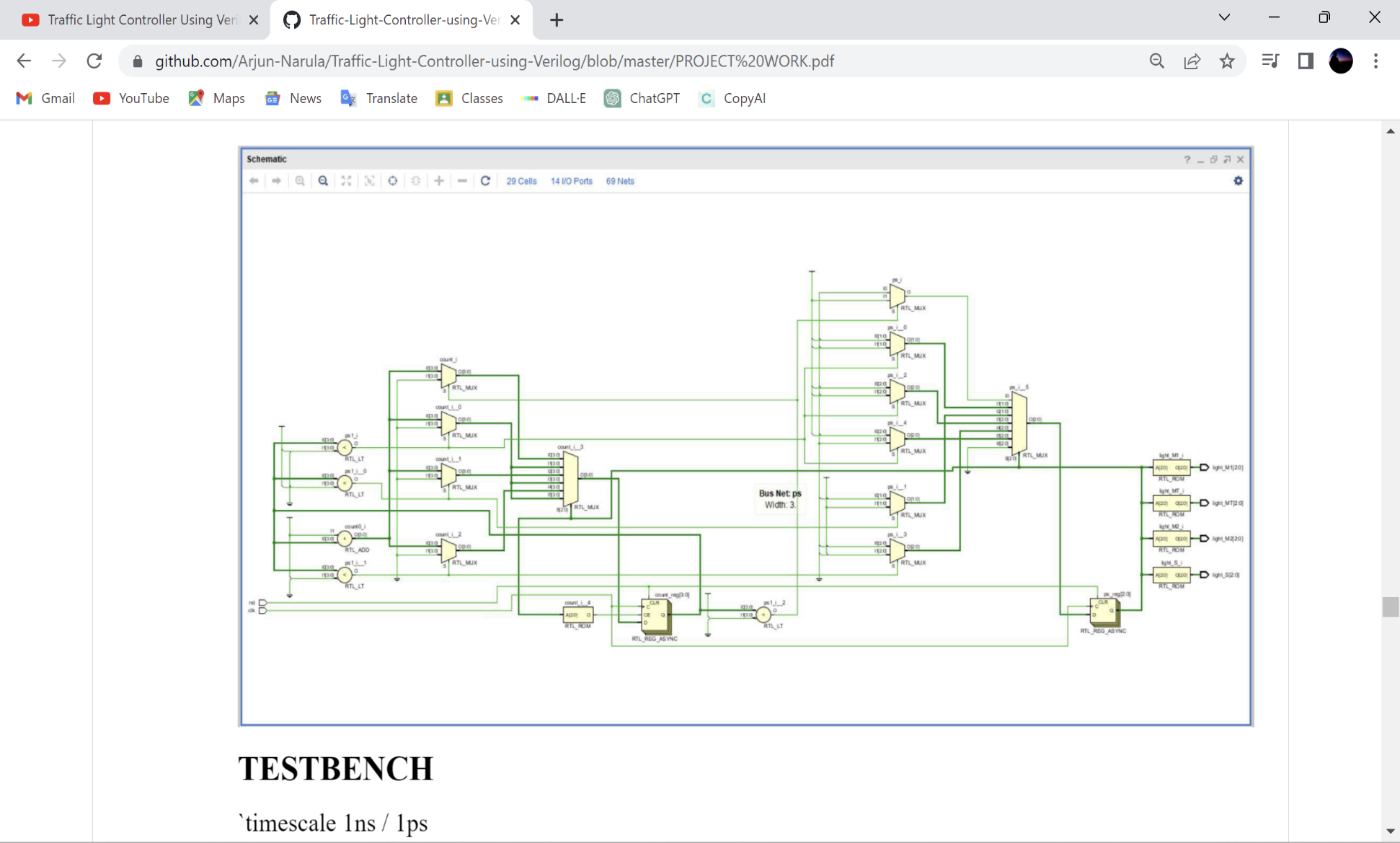
end

endcase

end

endmodule

**RTL-SCHEMATIC:**



**TESTBENCH:**

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 05:03:22 01/08/2023

// Design Name: Traffic\_Light\_Controller

// Module Name: /home/ise/Shared\_Vm/Projects/shreyas/Traffic\_light\_controller\_TB.v

// Project Name: shreyas

// Target Device:

// Tool versions:

// Description:

//

// Verilog Test Fixture created by ISE for module: Traffic\_Light\_Controller

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

////////////////////////////////////////////////////////////////////////////////

module Traffic\_light\_controller\_TB;

// Inputs

reg clk;

reg rst;

// Outputs

wire [2:0] light\_M1;

wire [2:0] light\_S;

wire [2:0] light\_MT;

wire [2:0] light\_M2;

// Instantiate the Unit Under Test (UUT)

Traffic\_Light\_Controller uut (

.clk(clk),

.rst(rst),

.light\_M1(light\_M1),

.light\_S(light\_S),

.light\_MT(light\_MT),

.light\_M2(light\_M2)

);

initial

begin

clk=1'b0;

forever #(10/2) clk=~clk;

end

// initial

// $stop;//to add ps

initial

begin

rst=0;

#1000;

rst=1;

#1000;

rst=0;

#(1000\*2);

$finish;

end

endmodule

**SIMULATED WAVEFORM:**

