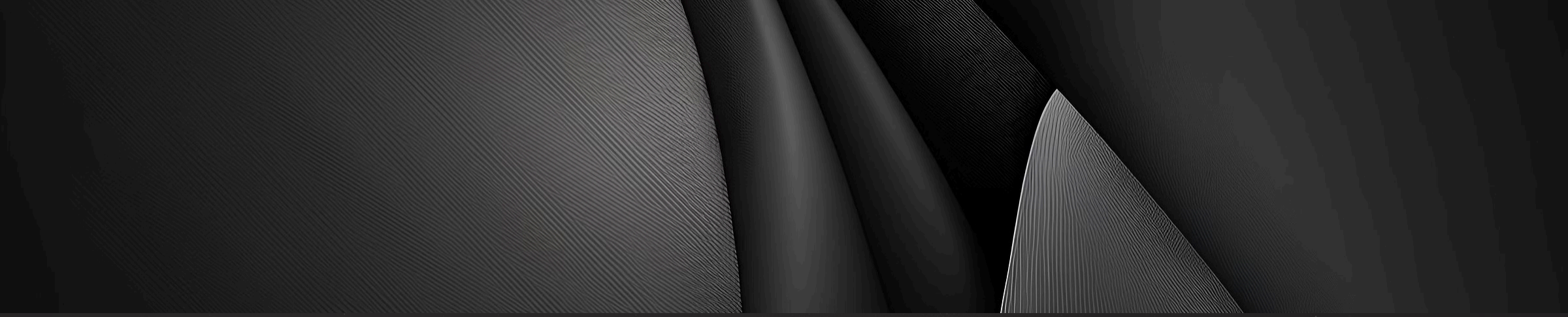
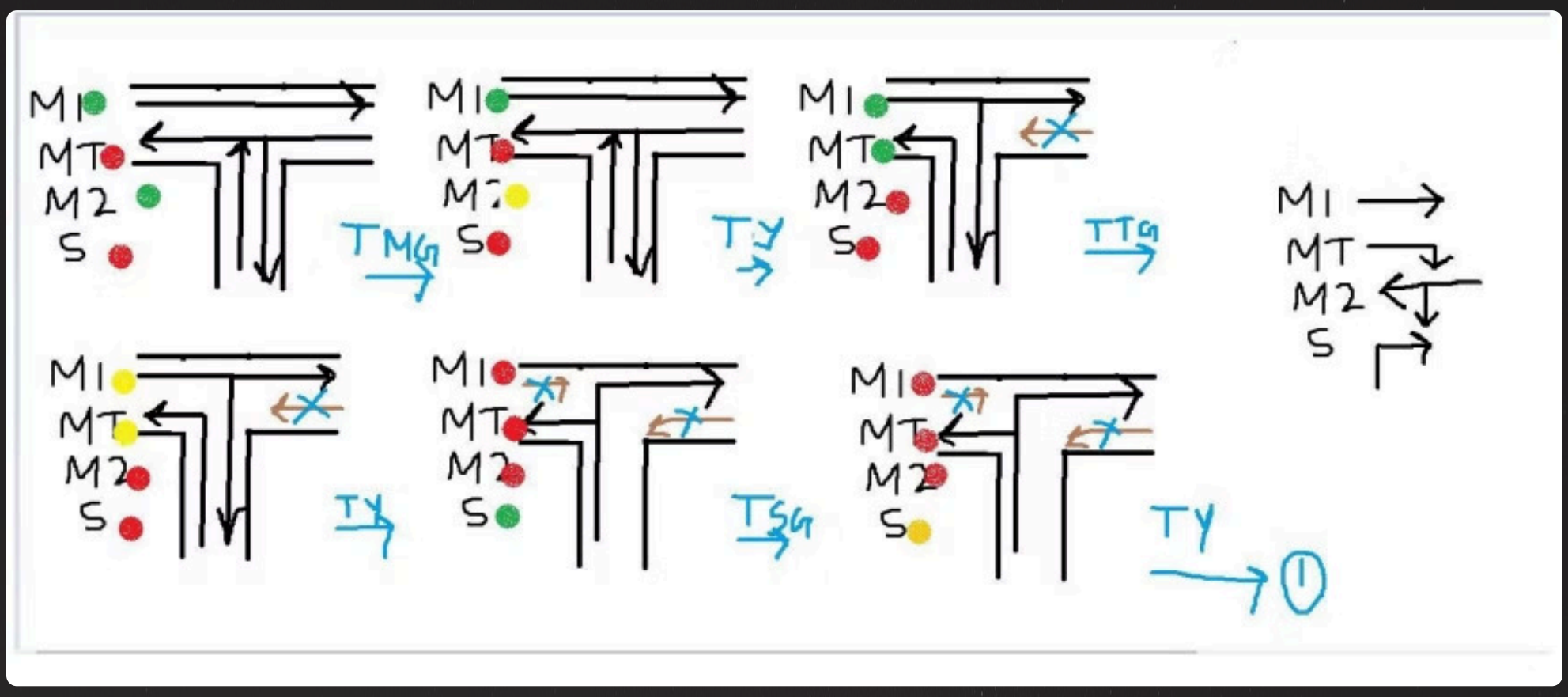


Topic: Traffic Light Controller Design using Verilog

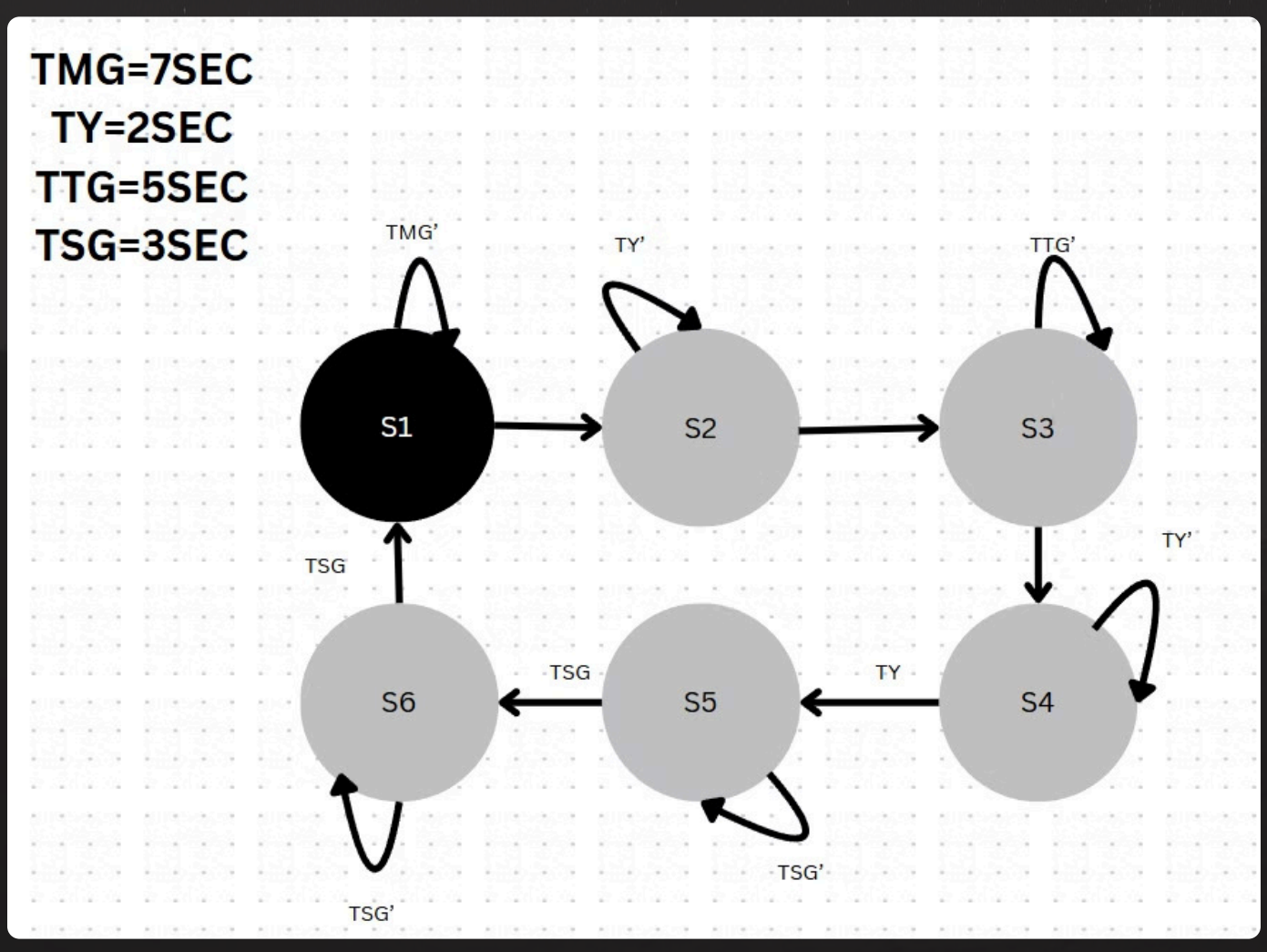
Submitted by: Swayam Swastik Sahu

PROBLEM STATEMENT

This project aims to design a traffic controller for a T-intersection. The image below illustrates the problem statement.



The six cases present here eventually turn to the six states . This is the state diagram:



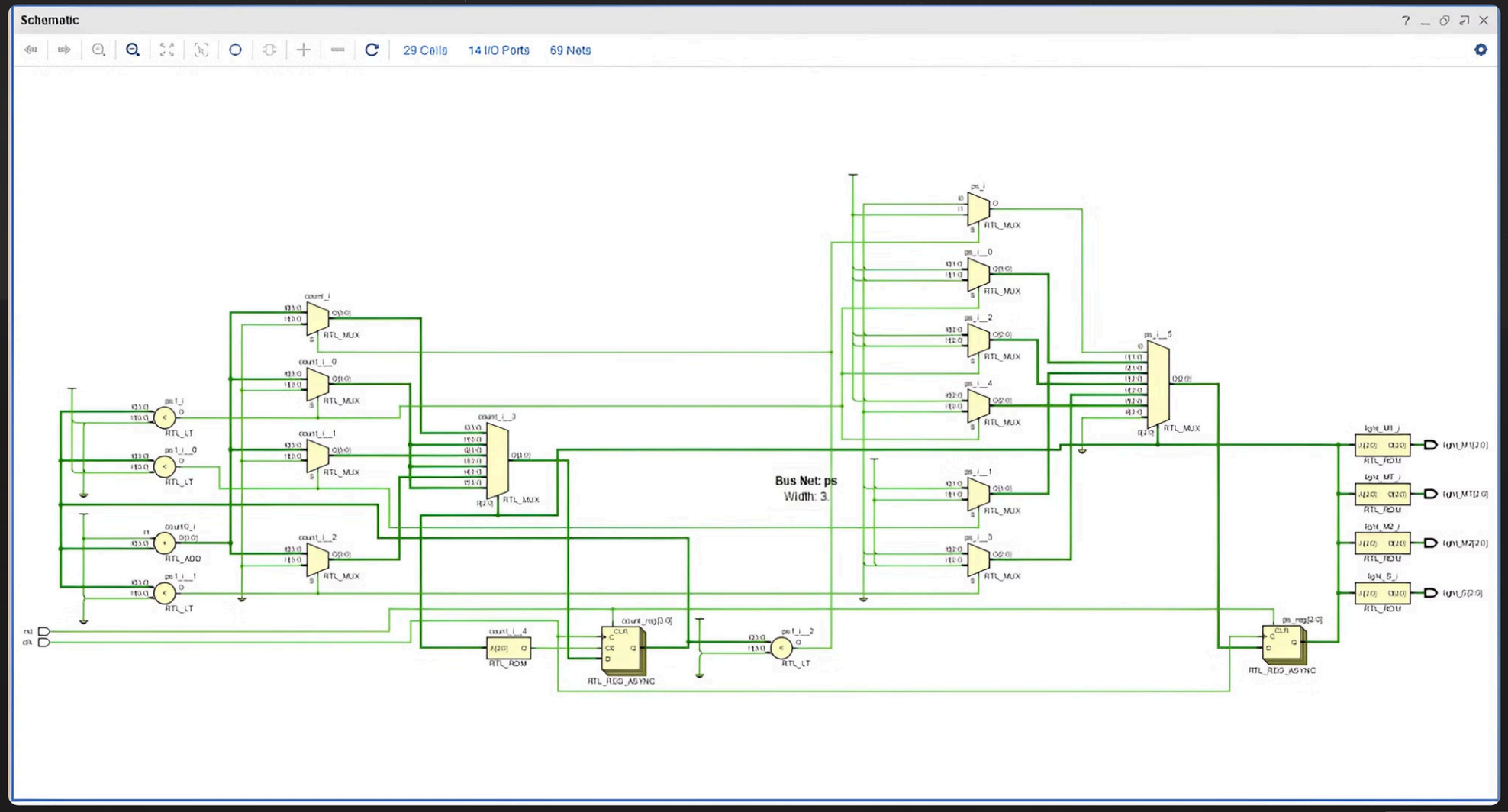
State Table

PRESENT STATE	INPUT	NEXT STATE	M1 R T G	M2 R Y G	MT R Y G	S R Y G
000	NA	001	001	001	100	100
000	NA	001				
001	TMG'(count<sec7)	001	001	001	100	100
001	TMG(count NOT <sec7)	010	001	010	100	100
010	TY'	010	001	010	100	100
010	TY	011	001	100	001	100
011	TTG'	011	001	100	001	100
011	TTG	100				
100	TY'	100				
100	TY	101				
101	TSG'	101				
101	TSG	110				
110	TY'	110				
110	TY	111				
111	COUNT<2	111	100	100	100	010
111	(COUNT<2)'	001	001	001	100	100

Verilog Code:

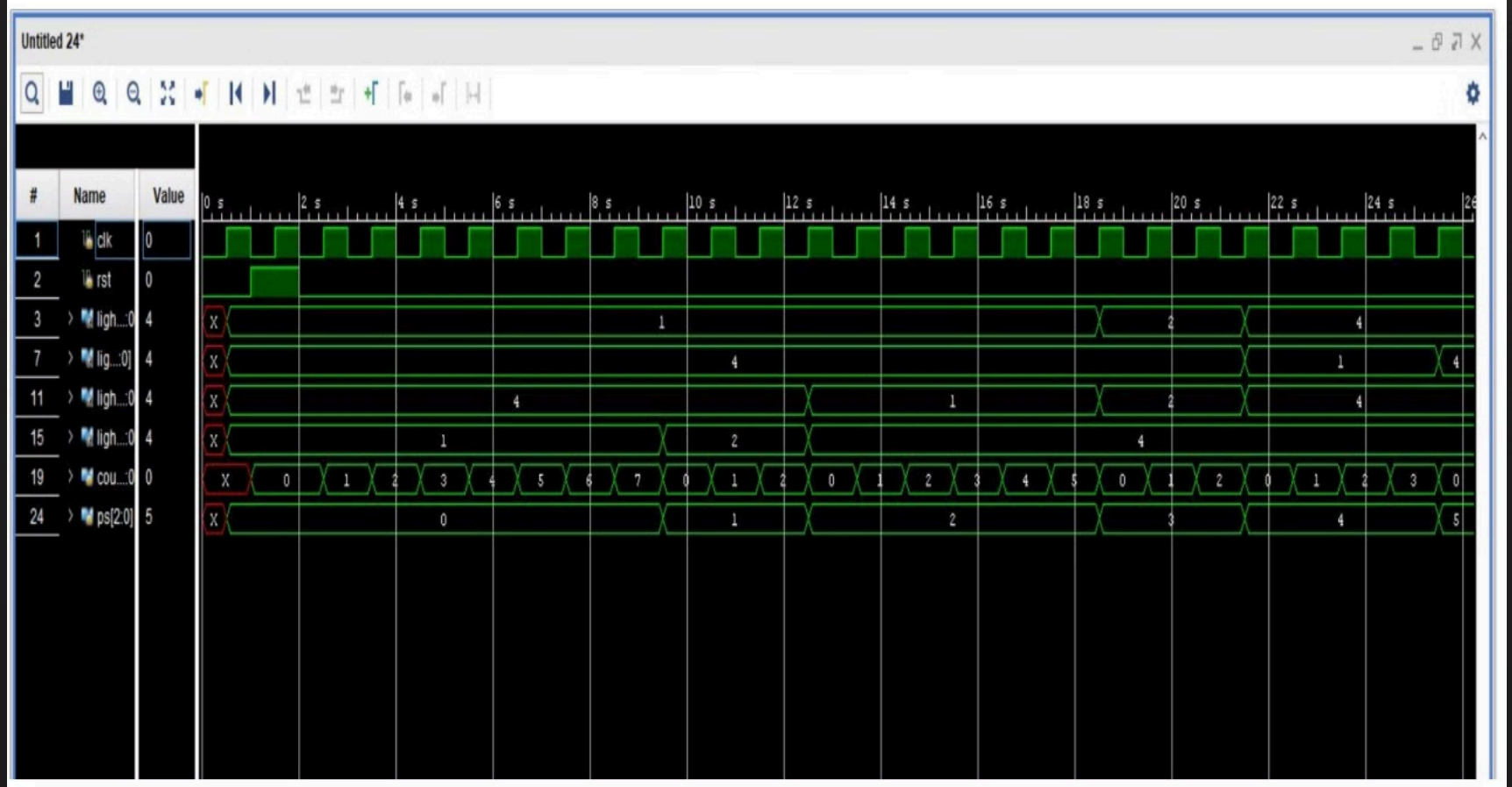
```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: // Engineer: // // Create Date: 16.07.2020 12:53:25 // Design Name: // Module Name: Traffic_Light_Controller // Project Name: // Target Devices: // Tool Versions: // Description: // // 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'b010;
         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



Simulation Wave & Timing Diagram

SIMULATED WAVEFORM



Upon analysing the waveform we can clearly see that the FSM works perfectly.