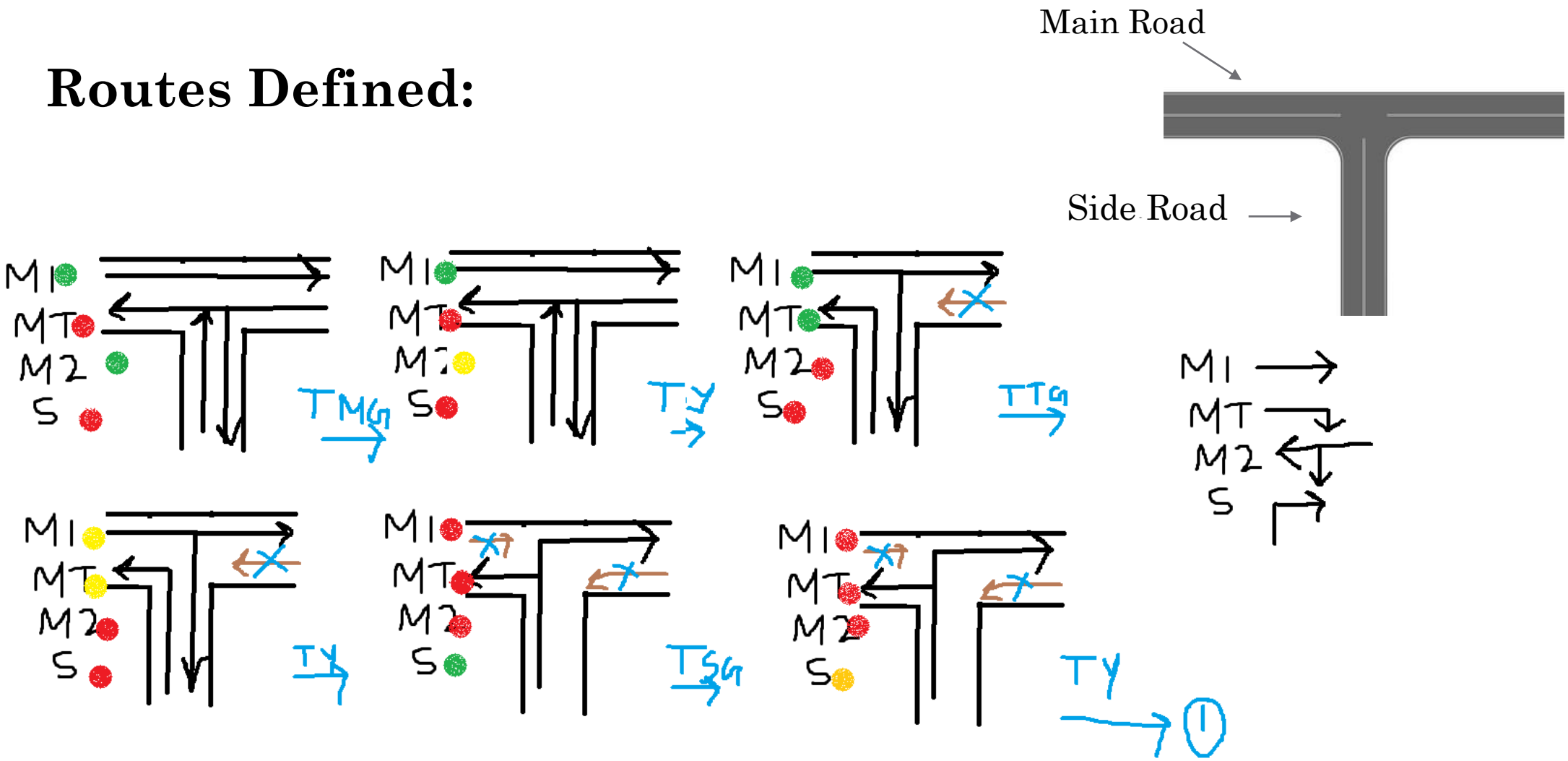
TRAFFIC LIGHT CONTOLLER

Name : Mor Vrushabh Bharatbhai
Roll No. : 220002055
Email ID : ee220002055@iiti.ac.in

Routes Defined:

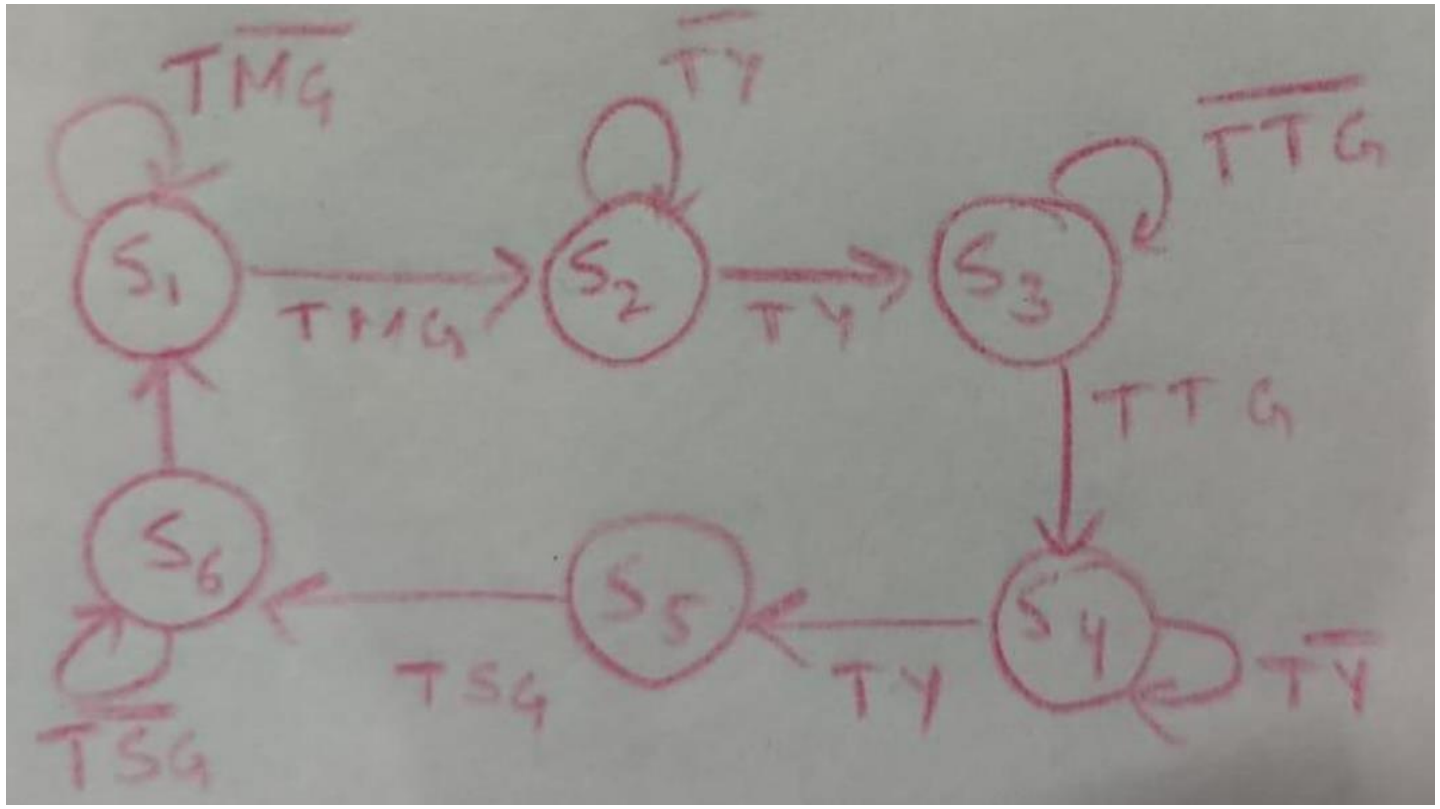


The six cases present here eventually turn to the six states.

State Diagram:

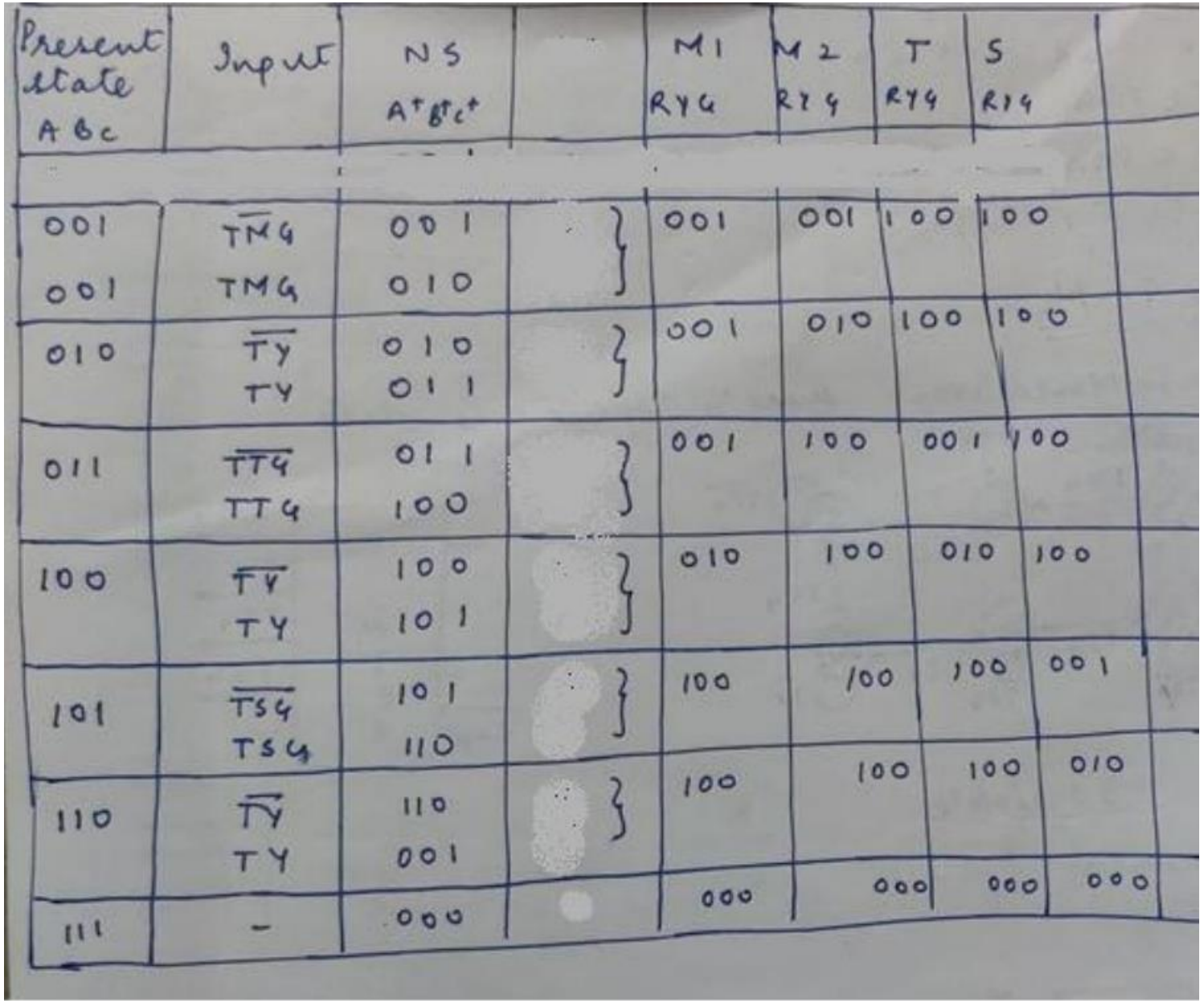
TMG : 7 sec
TY : 2 sec
TTG : 5 sec
TSG : 3 sec

- The Traffic Light Controller is basically acting as a Mealy Finite State Machine as the next state is dependent on the input as well as the previous states.



State Table:

- The state table is drawn from the state diagram.



Present state $A B c$	Input	NS $A^+ B^+ c^+$		M1 RYG	M2 RYG	T RYG	S RYG	
001	$\overline{TMG}$	001	}	001	001	100	100	
001	TMG	010						
010	$\overline{TY}$	010	}	001	010	100	100	
	TY	011						
011	$\overline{TTG}$	011	}	001	100	001	100	
	TTG	100						
100	$\overline{TY}$	100	}	010	100	010	100	
	TY	101						
101	$\overline{TSG}$	101	}	100	100	100	001	
	TSG	110						
110	$\overline{TY}$	110	}	100	100	100	010	
	TY	001						
111	-	000		000	000	000	000	

Verilog Code:

```
1  `timescale 1ns / 1ps
2  module Traffic_Light_Controller(
3      input clk,rst,
4      output reg [2:0]light_M1,
5      output reg [2:0]light_S,
6      output reg [2:0]light_MT,
7      output reg [2:0]light_M2
8      );
9      reg clock_out;
10     reg [28:0] cnt=29'd0;
11 parameter Divisor=29'd200000000;
12 always@(posedge clk)
13 begin
14 cnt<=cnt+29'd1;
15 if(cnt>=(Divisor-1))
16 cnt<=29'd0;
17 clock_out<=(cnt<Divisor/2)?1'b1:1'b0;
18 end
19
20     parameter  S1=0, S2=1, S3 =2, S4=3, S5=4,S6=5;
21     reg [3:0]count;
22     reg[2:0] ps;
23     parameter  sec7=7,sec5=5,sec2=2,sec3=3;
24
25
26
```

```
27     always@(posedge clock_out or posedge rst)
28         begin
29         if(rst==1)
30         begin
31         ps<=S1;
32         count<=0;
33         end
34         else
35             case(ps)
36                 S1: if(count<sec7)
37                         begin
38                         ps<=S1;
39                         count<=count+1;
40                         end
41                     else
42                         begin
43                         ps<=S2;
44                         count<=0;
45                         end
46                 S2: if(count<sec2)
47                         begin
48                         ps<=S2;
49                         count<=count+1;
50                         end
51
52                     else
```

Verilog Code:

```
52                  else
53                      begin
54                      ps<=S3;
55                      count<=0;
56                      end
57              S3: if(count<sec5)
58                      begin
59                      ps<=S3;
60                      count<=count+1;
61                      end
62
63                  else
64                      begin
65                      ps<=S4;
66                      count<=0;
67                      end
68              S4:if(count<sec2)
69                      begin
70                      ps<=S4;
71                      count<=count+1;
72                      end
73
74                  else
75                      begin
76                      ps<=S5;
77                      count<=0;
```

```
78                      end
79              S5:if(count<sec3)
80                      begin
81                      ps<=S5;
82                      count<=count+1;
83                      end
84
85                  else
86                      begin
87                      ps<=S6;
88                      count<=0;
89                      end
90
91              S6:if(count<sec2)
92                      begin
93                      ps<=S6;
94                      count<=count+1;
95                      end
96
97                  else
98                      begin
99                      ps<=S1;
100                     count<=0;
101                     end
102             default: ps<=S1;
103             endcase
```

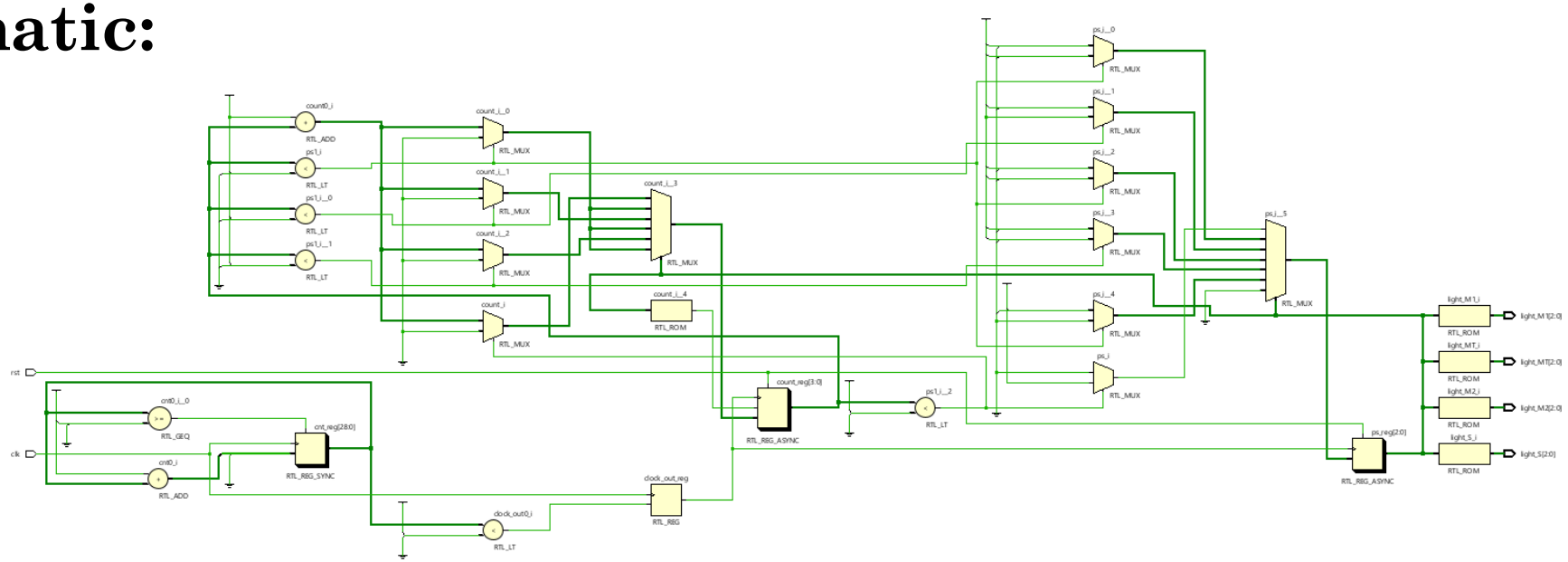
Verilog Code:

```
104             end
105
106             always@(ps)
107             begin
108
109                 case(ps)
110
111                     S1:
112                     begin
113                        light_M1<=3'b001;
114                        light_M2<=3'b001;
115                        light_MT<=3'b100;
116                        light_S<=3'b100;
117                     end
118                     S2:
119                     begin
120                        light_M1<=3'b001;
121                        light_M2<=3'b010;
122                        light_MT<=3'b100;
123                        light_S<=3'b100;
124                     end
125                     S3:
126                     begin
127                        light_M1<=3'b001;
128                        light_M2<=3'b100;
129                        light_MT<=3'b001;
130                        light_S<=3'b100;
131                     end
132                     S4:
133                     begin
134                        light_M1<=3'b010;
135                        light_M2<=3'b100;
136                        light_MT<=3'b010;
137                        light_S<=3'b100;
138                     end
139                     S5:
140                     begin
141                        light_M1<=3'b100;
142                        light_M2<=3'b100;
143                        light_MT<=3'b100;
144                        light_S<=3'b001;
145                     end
146                     S6:
147                     begin
148                        light_M1<=3'b100;
149                        light_M2<=3'b100;
150                        light_MT<=3'b100;
151                        light_S<=3'b010;
152                     end
153                     default:
154                     begin
155                        light_M1<=3'b000;
156                        light_M2<=3'b000;
157                        light_MT<=3'b000;
158                        light_S<=3'b000;
159                     end
160                 endcase
161             end
162
163
164 endmodule
```

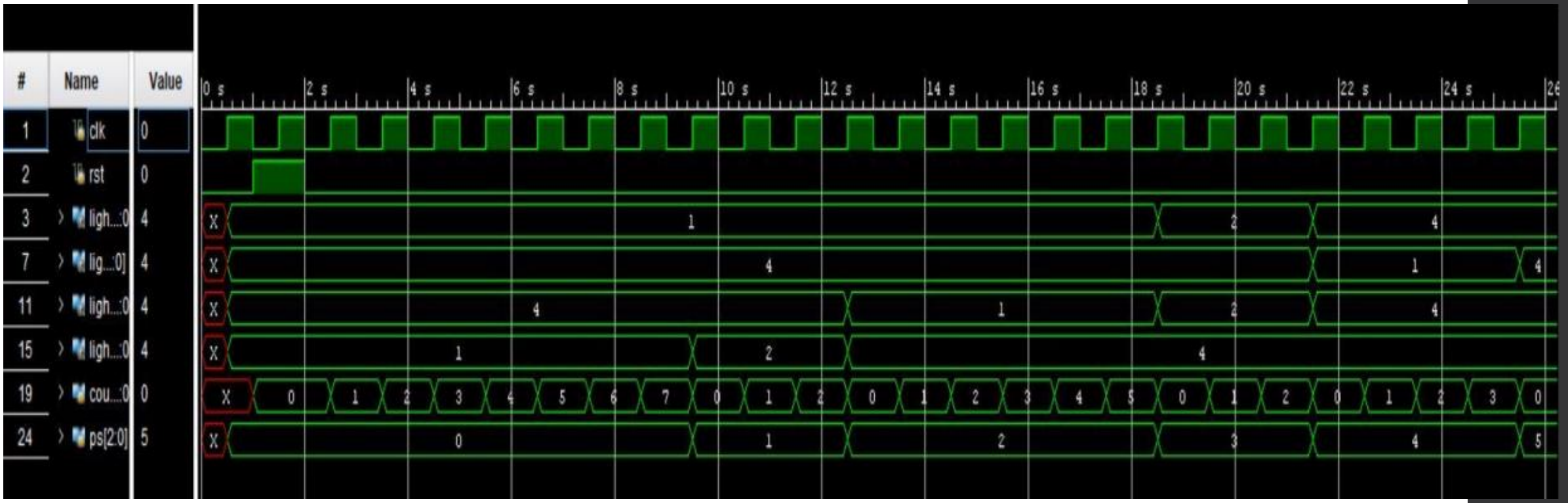
Testbench:

```
1  `timescale 1ns / 1ps
2  module Traffic_Light_Controller_TB;
3  reg clk,rst;
4  wire [2:0]light_M1;
5  wire [2:0]light_S;
6  wire [2:0]light_MT;
7  wire [2:0]light_M2;
8  Traffic_Light_Controller dut(.clk(clk) , .rst(rst) ,
9   .light_M1(light_M1) , .light_S(light_S)  ,
10  .light_M2(light_M2),.light_MT(light_MT)   );
11 initial begin
12     clk=1'b0;
13     forever #(1000000000/2) clk=~clk;
14 end
15 //    initial
16 //    $stop;//to add ps
17 initial begin
18     rst=0;
19     #1000000000;
20     rst=1;
21     #1000000000;
22     rst=0;
23     #(1000000000*200);
24     $finish;
25     end
26 endmodule
```

Schematic:



Output Waveform:




Thank you