1. **Aim of the Experiment/Simulation (Task details):**

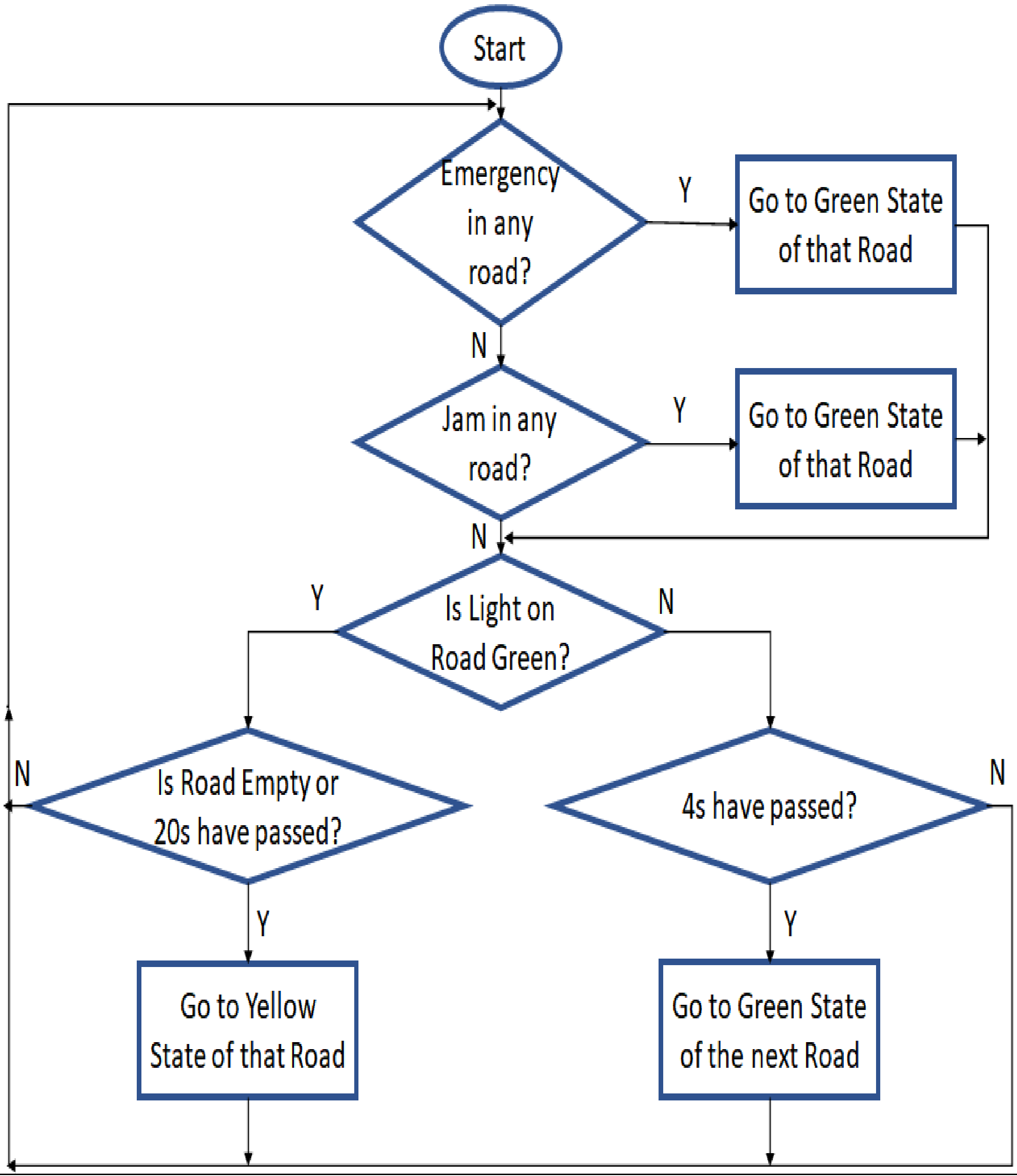
To :

- Design a Traffic Light Controller using FSM.
- Write the Verilog code and Verify with Test Bench

2. **Name of the Simulation Platform/Tools:** (If applicable)

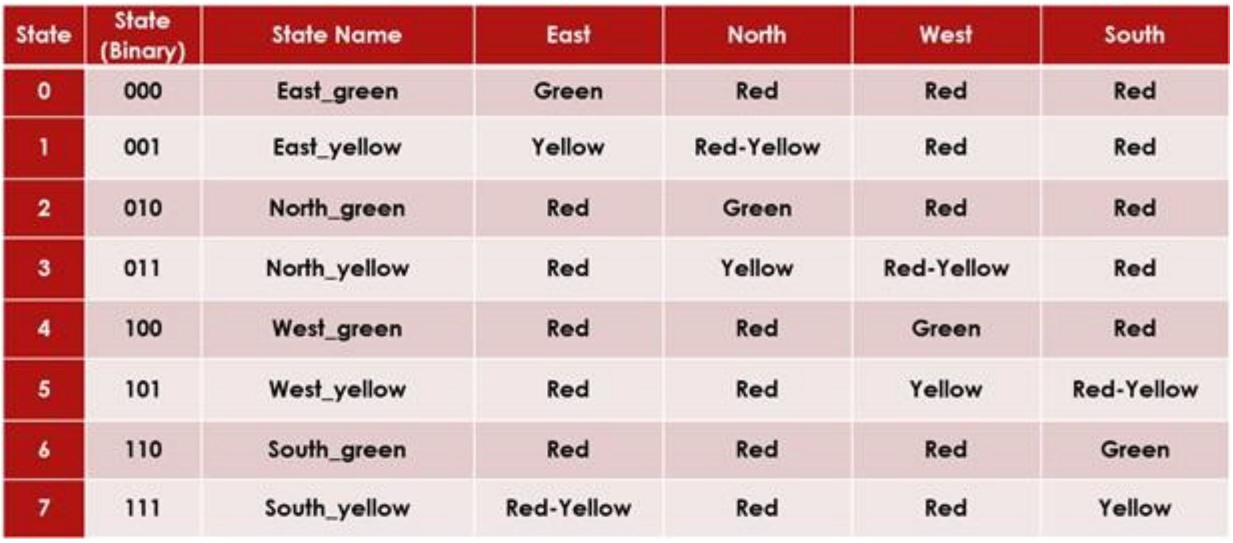
Model-Sim

3. **Theory:**



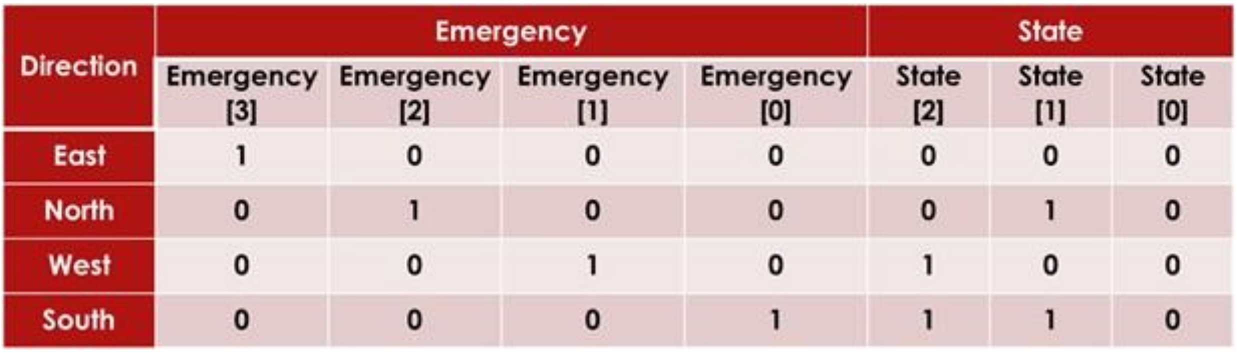
Start
Emergency in any road?
Y
Go to Green State of that Road
N
Jam in any road?
Y
Go to Green State of that Road
N
Is Light on Road Green?
Y
N
Is Road Empty or 20s have passed?
N
Y
Go to Yellow State of that Road
4s have passed?
N
Y
Go to Green State of the next Road

Description of states



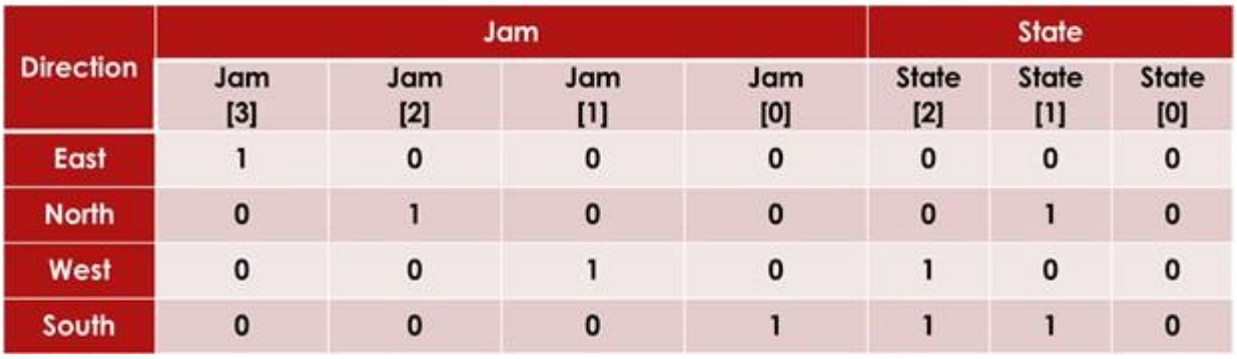
State	State (Binary)	State Name	East	North	West	South
0	000	East_green	Green	Red	Red	Red
1	001	East_yellow	Yellow	Red-Yellow	Red	Red
2	010	North_green	Red	Green	Red	Red
3	011	North_yellow	Red	Yellow	Red-Yellow	Red
4	100	West_green	Red	Red	Green	Red
5	101	West_yellow	Red	Red	Yellow	Red-Yellow
6	110	South_green	Red	Red	Red	Green
7	111	South_yellow	Red-Yellow	Red	Red	Yellow

Emergency condition :



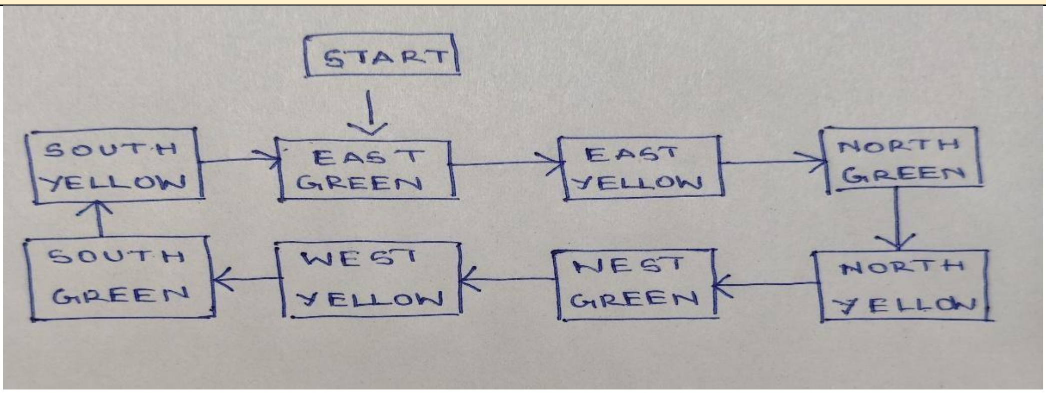
Direction	Emergency				State		
	Emergency [3]	Emergency [2]	Emergency [1]	Emergency [0]	State [2]	State [1]	State [0]
East	1	0	0	0	0	0	0
North	0	1	0	0	0	1	0
West	0	0	1	0	1	0	0
South	0	0	0	1	1	1	0

Jam Condition :



Direction	Jam				State		
	Jam [3]	Jam [2]	Jam [1]	Jam [0]	State [2]	State [1]	State [0]
East	1	0	0	0	0	0	0
North	0	1	0	0	0	1	0
West	0	0	1	0	1	0	0
South	0	0	0	1	1	1	0

4.	**State diagram:**



5.	**Verilog HDL Code and Outputs:**

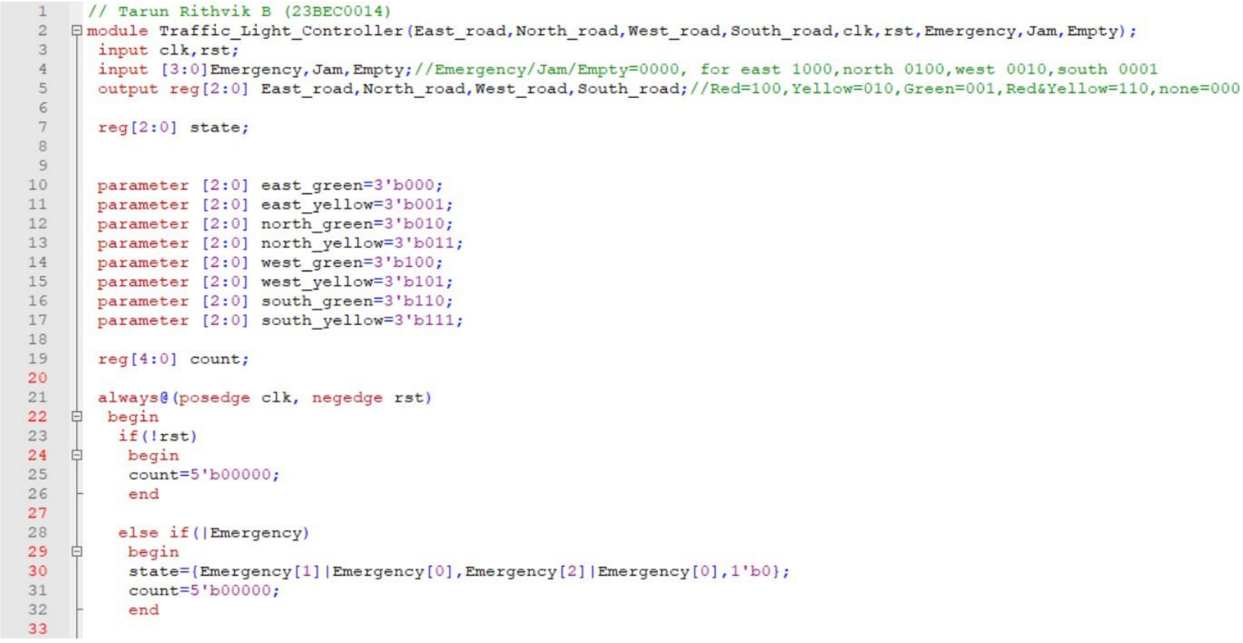
```
// Tarun Rithvik B (23BEC0014)
module
Traffic_Light_Controller(East_road,North_road,West_road,South_road,clk,rst,Emergency,Jam,E
mpty);
 input clk,rst;
 input [3:0]Emergency,Jam,Empty;//Emergency/Jam/Empty=0000, for east 1000,north 0100,west
0010,south 0001
 output reg[2:0]
East_road,North_road,West_road,South_road;//Red=100,Yellow=010,Green=001,Red&Yellow=
110,none=000

 reg[2:0] state;


 parameter [2:0] east_green=3'b000;
 parameter [2:0] east_yellow=3'b001;
 parameter [2:0] north_green=3'b010;
 parameter [2:0] north_yellow=3'b011;
 parameter [2:0] west_green=3'b100;
 parameter [2:0] west_yellow=3'b101;
 parameter [2:0] south_green=3'b110;
 parameter [2:0] south_yellow=3'b111;

 reg[4:0] count;

 always@(posedge clk, negedge rst)
  begin
   if(!rst)
    begin
    count=5'b00000;
    end

   else if(|Emergency)
    begin
    state={Emergency[1]|Emergency[0],Emergency[2]|Emergency[0],1'b0};
    count=5'b00000;
    end

   else if(|Jam)
    begin
    state={Jam[1]|Jam[0],Jam[2]|Jam[0],1'b0};
    count=5'b00000;
    end

   else
    begin
     case(state)

      east_green:
         begin
          if(count==5'b10011||Empty==4'b1000)
            begin
            count=5'b00000;
            state=east_yellow;
            end
          else
            begin
            count=count+5'b00001;
            state=east_green;
            end
          end

       east_yellow:
         begin
          if(count==5'b00011)
          begin
          count=5'b00000;
          state=north_green;
          end
          else
            begin
            count=count+5'b00001;
            state=east_yellow;
            end
          end

      north_green:
         begin
          if(count==5'b10011||Empty==4'b0100)
          begin
            count=5'b00000;
            state=north_yellow;
            end
          else
            begin
            count=count+5'b00001;
            state=north_green;
            end
          end

      north_yellow:
         begin
          if(count==5'b00011)
          begin
          count=5'b00000;
          state=west_green;
          end
          else
            begin
            count=count+5'b00001;
            state=north_yellow;
            end
          end
```

```
west_green:
   begin
    if(count==5'b10011||Empty==4'b0010)
    begin
      count=5'b00000;
      state=west_yellow;
      end
    else
      begin
      count=count+5'b000001;
      state=west_green;
      end
    end

west_yellow:
   begin
    if(count==5'b00011)
    begin
    count=5'b00000;
    state=south_green;
    end
    else
      begin
      count=count+5'b00001;
      state=west_yellow;
      end
    end

 south_green:
   begin
    if(count==5'b10011||Empty==4'b0001)
    begin
      count=5'b00000;
      state=south_yellow;
      end
    else
      begin
      count=count+5'b00001;
      state=south_green;
      end
    end

south_yellow:
   begin
    if(count==5'b00011)
      begin
      count=5'b00000;
      state=east_green;
      end
    else
      begin
```

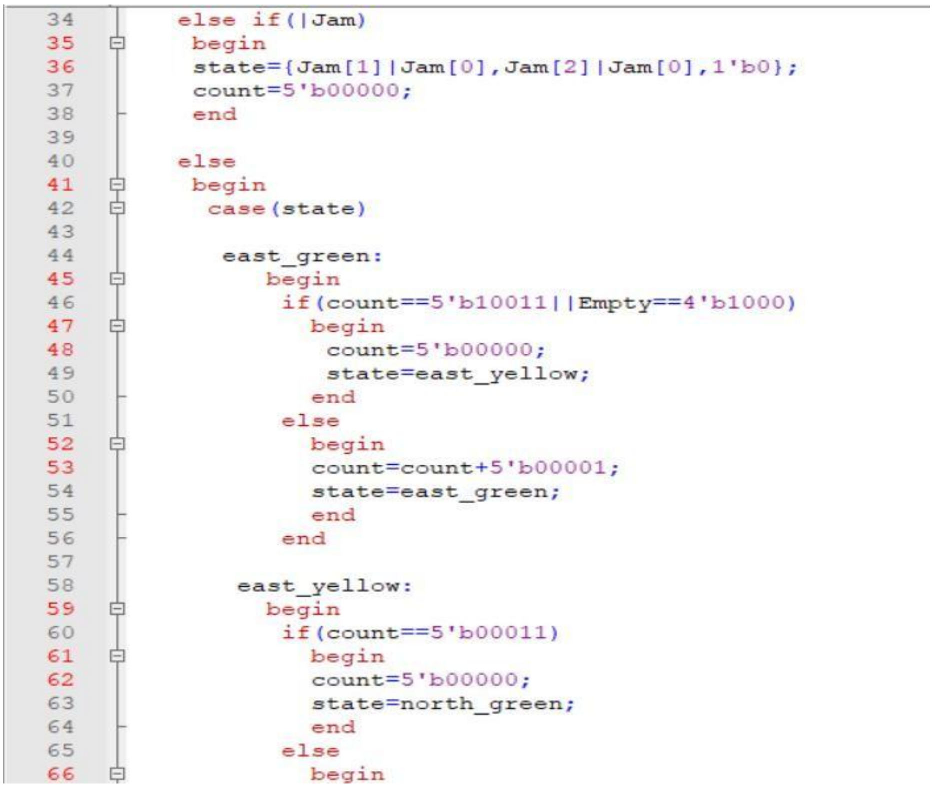
```
          count=count+5'b00001;
          state=south_yellow;
          end
        end

    default:
       begin
        count=5'b00000;
        state=east_green;

       end

   endcase
  end
 end

//Red=100,Yellow=010,Green=001,Red&Yellow=110,none=000
always@(state)
  begin
  if(!rst)
     begin
     East_road=3'b000;
     North_road=3'b000;
     West_road=3'b000;
     South_road=3'b000;
     end
  else
   begin
   case(state)

    east_green:
     begin
     East_road=3'b001;
     North_road=3'b100;
     West_road=3'b100;
     South_road=3'b100;
     end

    east_yellow:
     begin
      East_road=3'b010;
     North_road=3'b110;//
     West_road=3'b100;
     South_road=3'b100;
     end

    north_green:
     begin
      East_road=3'b100;
      North_road=3'b001;
      West_road=3'b100;
      South_road=3'b100;
```

```
       end

      north_yellow:
       begin
       East_road=3'b100;
       North_road=3'b010;
       West_road=3'b110;//
       South_road=3'b100;
       end

      west_green:
       begin
       East_road=3'b100;
       North_road=3'b100;
       West_road=3'b001;
       South_road=3'b100;
       end

      west_yellow:
       begin
       East_road=3'b100;
       North_road=3'b100;
       West_road=3'b010;
       South_road=3'b110; //
       end

      south_green:
       begin
       East_road=3'b100;
       North_road=3'b100;
       West_road=3'b100;
       South_road=3'b001;
       end

      south_yellow:
       begin
       East_road=3'b110;//
       North_road=3'b100;
       West_road=3'b100;
       South_road=3'b010;
       end

     endcase
    end
   end
endmodule

module tb_Traffic_Light_Controller;

  reg clk;
  reg rst;
  reg [3:0] Emergency;
reg [3:0] Jam;
reg [3:0] Empty;

wire [2:0] East_road;
wire [2:0] North_road;
wire [2:0] West_road;
wire [2:0] South_road;

Traffic_Light_Controller uut (
  .clk(clk),
  .rst(rst),
  .Emergency(Emergency),
  .Jam(Jam),
  .Empty(Empty),
  .East_road(East_road),
  .North_road(North_road),
  .West_road(West_road),
  .South_road(South_road)
);


always #5 clk = ~clk;
initial begin

  clk = 0;
  rst = 1;
  Emergency = 4'b0000;
  Jam = 4'b0000;
  Empty = 4'b0000;
  #10;
  rst = 0;
  #10;
  rst = 1;
  #20;
  Empty = 4'b0000;
  #200;
  Emergency = 4'b1000;
  #20;
  Emergency = 4'b0000;
  #20;
  Jam = 4'b0100;
  #20;
  Jam = 4'b0000;
  #20;
  Empty = 4'b1000;
  #40;
  Empty = 4'b0100;
  #40;
  Empty = 4'b0010;
  #40;
  Empty = 4'b0001;
  #40;
```

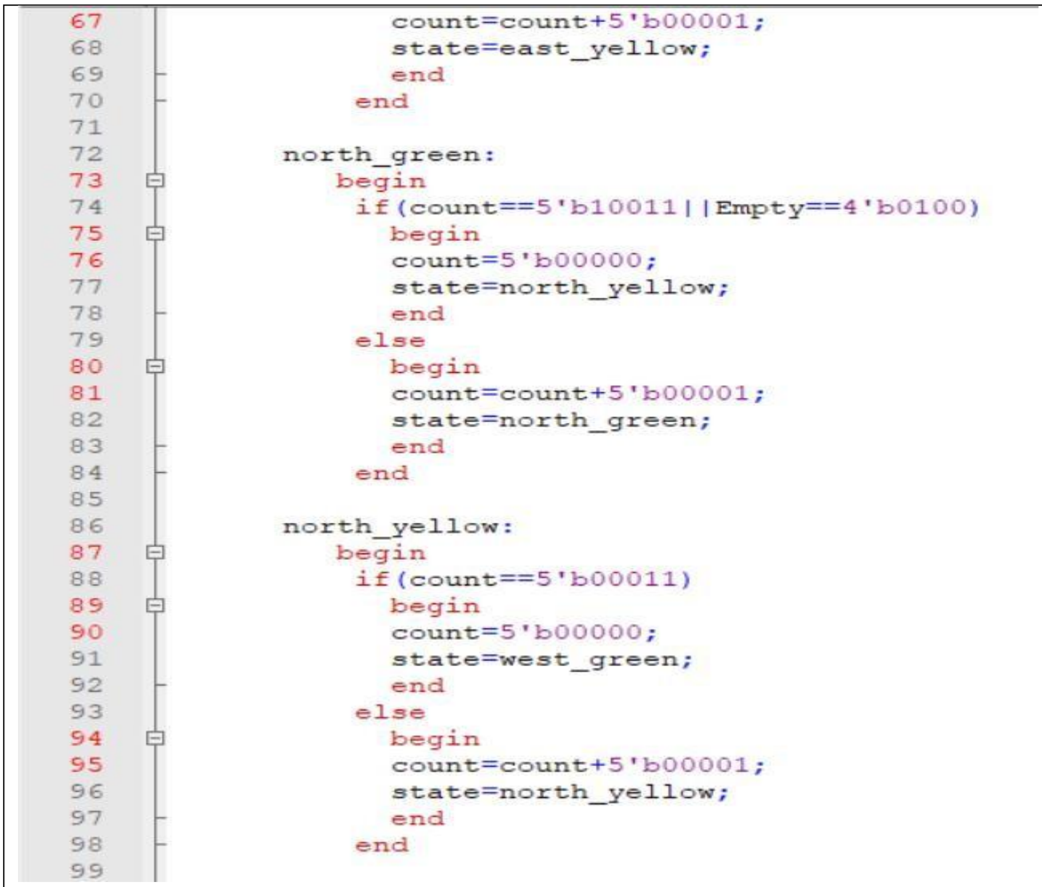
```
   Empty = 4'b0000;
   #200;
   $stop;
  end

  initial begin
   $monitor("Time=%0t | East=%b | North=%b | West=%b | South=%b | Emergency=%b |
Jam=%b | Empty=%b",
            $time, East_road, North_road, West_road, South_road, Emergency, Jam, Empty);
  end
endmodule
```



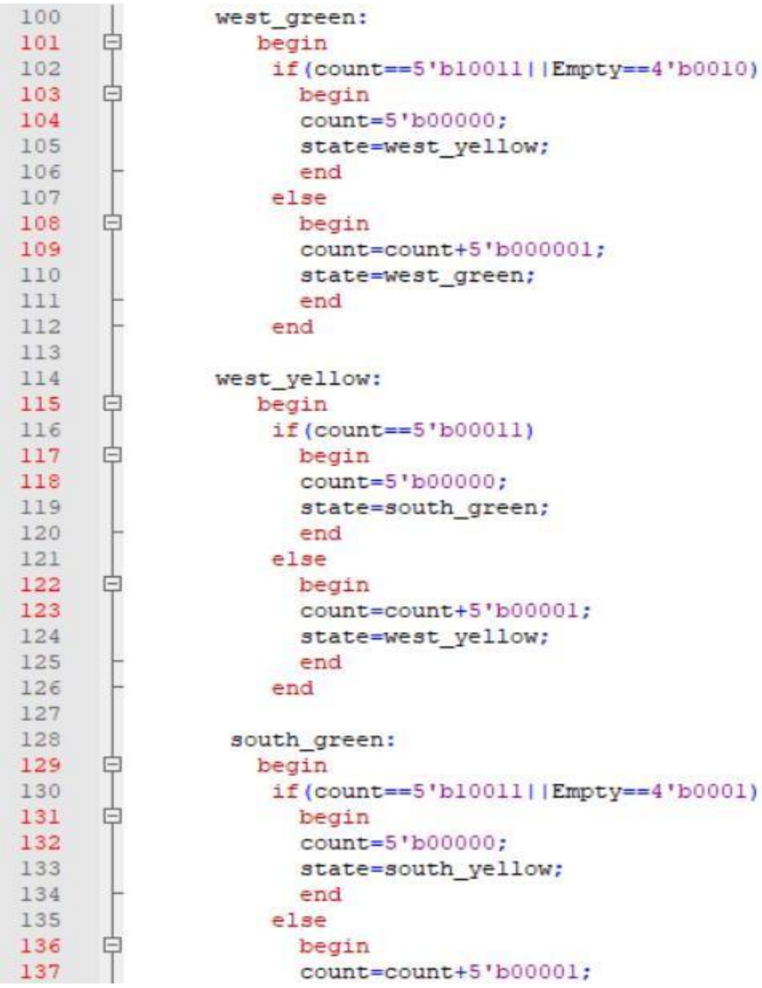
```
// Tarun Rithvik B (23BEC0014)
module Traffic_Light_Controller(East_road,North_road,West_road,South_road,clk,rst,Emergency,Jam,Empty);
 input clk,rst;
 input [3:0]Emergency,Jam,Empty;//Emergency/Jam/Empty=0000, for east 1000,north 0100,west 0010,south 0001
 output reg[2:0] East_road,North_road,West_road,South_road;//Red=100,Yellow=010,Green=001,Red&Yellow=110,none=000

 reg[2:0] state;


 parameter [2:0] east_green=3'b000;
 parameter [2:0] east_yellow=3'b001;
 parameter [2:0] north_green=3'b010;
 parameter [2:0] north_yellow=3'b011;
 parameter [2:0] west_green=3'b100;
 parameter [2:0] west_yellow=3'b101;
 parameter [2:0] south_green=3'b110;
 parameter [2:0] south_yellow=3'b111;

 reg[4:0] count;

 always@(posedge clk, negedge rst)
  begin
   if(!rst)
    begin
    count=5'b00000;
    end

   else if(|Emergency)
    begin
    state={Emergency[1]|Emergency[0],Emergency[2]|Emergency[0],1'b0};
    count=5'b00000;
    end

```



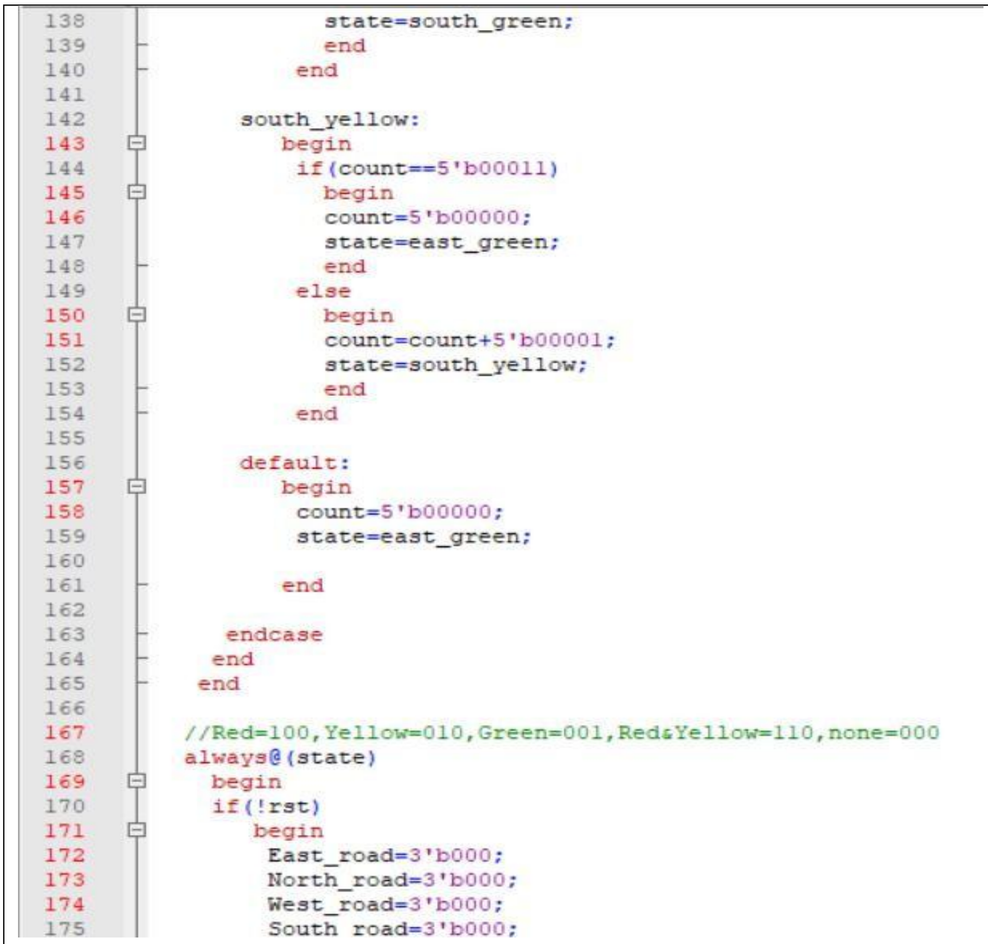
```
   else if(|Jam)
    begin
    state={Jam[1]|Jam[0],Jam[2]|Jam[0],1'b0};
    count=5'b00000;
    end

   else
    begin
     case(state)

      east_green:
         begin
          if(count==5'b10011||Empty==4'b1000)
            begin
             count=5'b00000;
             state=east_yellow;
            end
          else
            begin
            count=count+5'b00001;
            state=east_green;
            end
          end

       east_yellow:
         begin
          if(count==5'b00011)
            begin
            count=5'b00000;
            state=north_green;
            end
          else
            begin
```



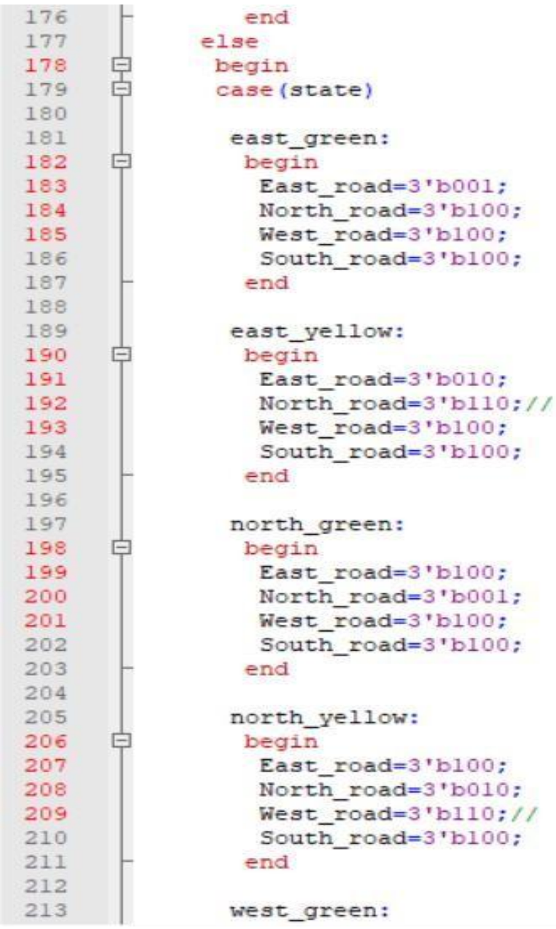
```
              count=count+5'b00001;
              state=east_yellow;
              end
            end

        north_green:
           begin
            if(count==5'b10011||Empty==4'b0100)
              begin
              count=5'b00000;
              state=north_yellow;
              end
            else
              begin
              count=count+5'b00001;
              state=north_green;
              end
            end

        north_yellow:
           begin
            if(count==5'b00011)
              begin
              count=5'b00000;
              state=west_green;
              end
            else
              begin
              count=count+5'b00001;
              state=north_yellow;
              end
            end

```



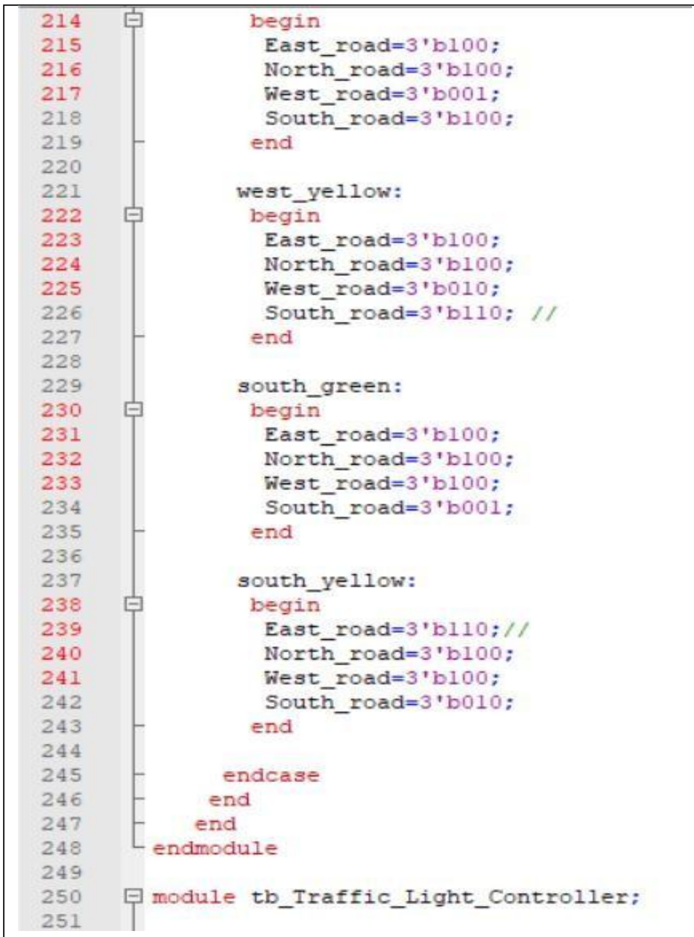
```
        west_green:
           begin
            if(count==5'b10011||Empty==4'b0010)
              begin
              count=5'b00000;
              state=west_yellow;
              end
            else
              begin
              count=count+5'b000001;
              state=west_green;
              end
            end

        west_yellow:
           begin
            if(count==5'b00011)
              begin
              count=5'b00000;
              state=south_green;
              end
            else
              begin
              count=count+5'b00001;
              state=west_yellow;
              end
            end

         south_green:
           begin
            if(count==5'b10011||Empty==4'b0001)
              begin
              count=5'b00000;
              state=south_yellow;
              end
            else
              begin
              count=count+5'b00001;
```



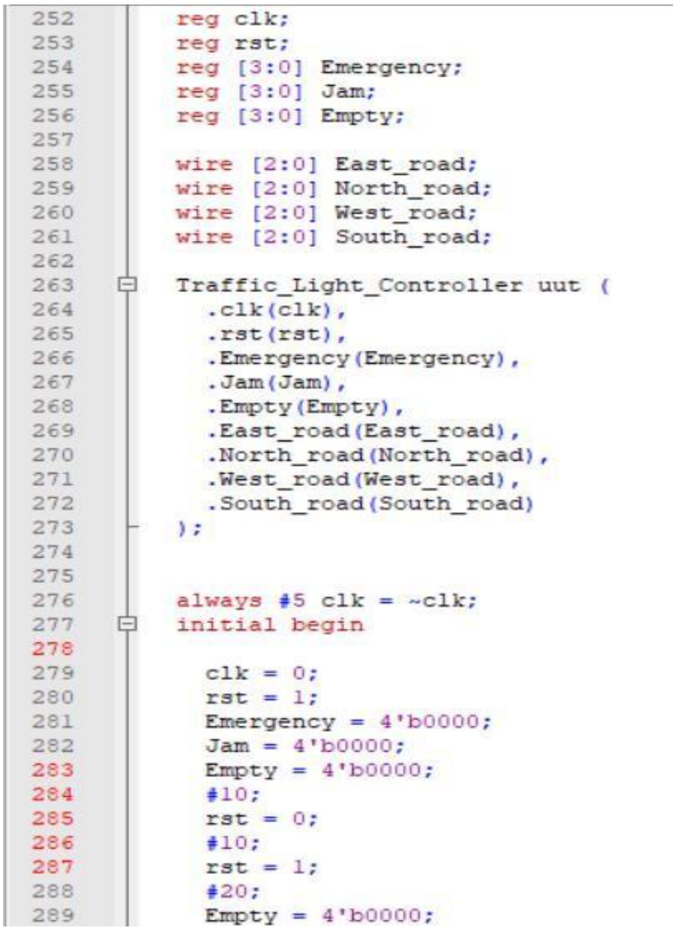
```
          state=south_green;
          end
        end

    south_yellow:
       begin
        if(count==5'b00011)
          begin
          count=5'b00000;
          state=east_green;
          end
        else
          begin
          count=count+5'b00001;
          state=south_yellow;
          end
        end

    default:
       begin
        count=5'b00000;
        state=east_green;

       end

   endcase
  end
 end

//Red=100,Yellow=010,Green=001,Red&Yellow=110,none=000
always@(state)
  begin
  if(!rst)
     begin
      East_road=3'b000;
      North_road=3'b000;
      West_road=3'b000;
      South road=3'b000;
```



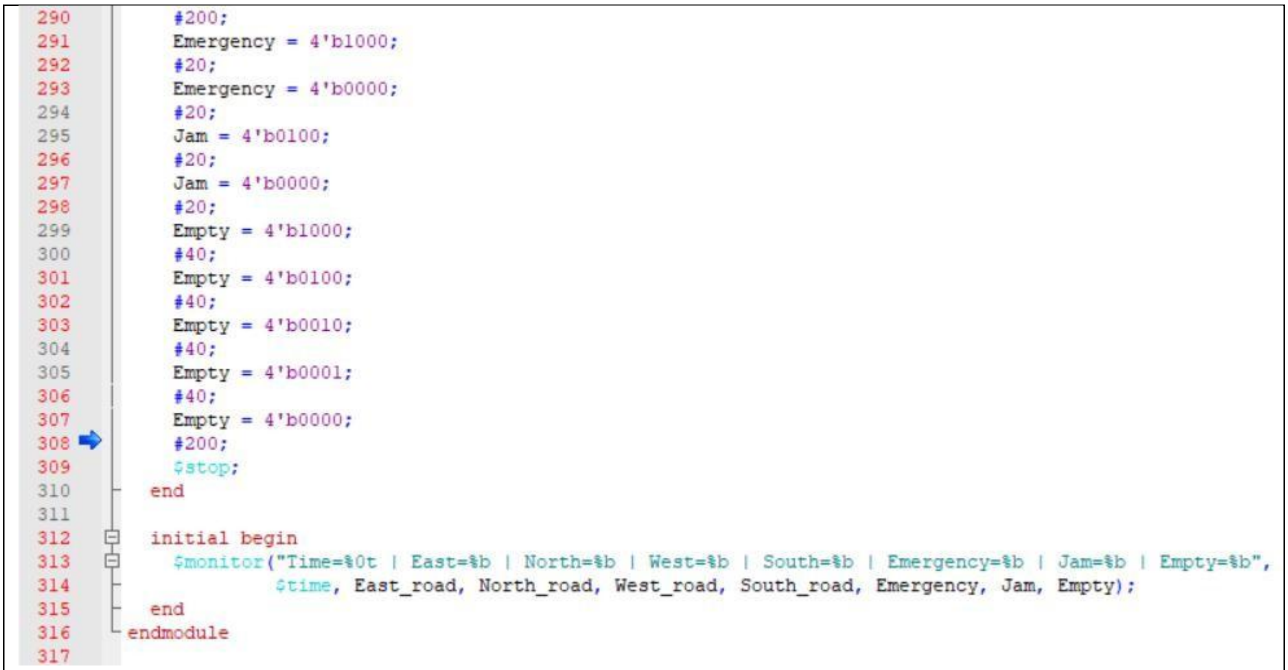
```
     end
  else
   begin
   case(state)

    east_green:
     begin
      East_road=3'b001;
      North_road=3'b100;
      West_road=3'b100;
      South_road=3'b100;
     end

    east_yellow:
     begin
      East_road=3'b010;
      North_road=3'b110;//
      West_road=3'b100;
      South_road=3'b100;
     end

    north_green:
     begin
      East_road=3'b100;
      North_road=3'b001;
      West_road=3'b100;
      South_road=3'b100;
     end

    north_yellow:
     begin
      East_road=3'b100;
      North_road=3'b010;
      West_road=3'b110;//
      South_road=3'b100;
     end

    west_green:
```



```
        begin
         East_road=3'b100;
         North_road=3'b100;
         West_road=3'b001;
         South_road=3'b100;
        end

       west_yellow:
        begin
         East_road=3'b100;
         North_road=3'b100;
         West_road=3'b010;
         South_road=3'b110; //
        end

       south_green:
        begin
         East_road=3'b100;
         North_road=3'b100;
         West_road=3'b100;
         South_road=3'b001;
        end

       south_yellow:
        begin
         East_road=3'b110;//
         North_road=3'b100;
         West_road=3'b100;
         South_road=3'b010;
        end

      endcase
     end
    end
endmodule

module tb_Traffic_Light_Controller;

```

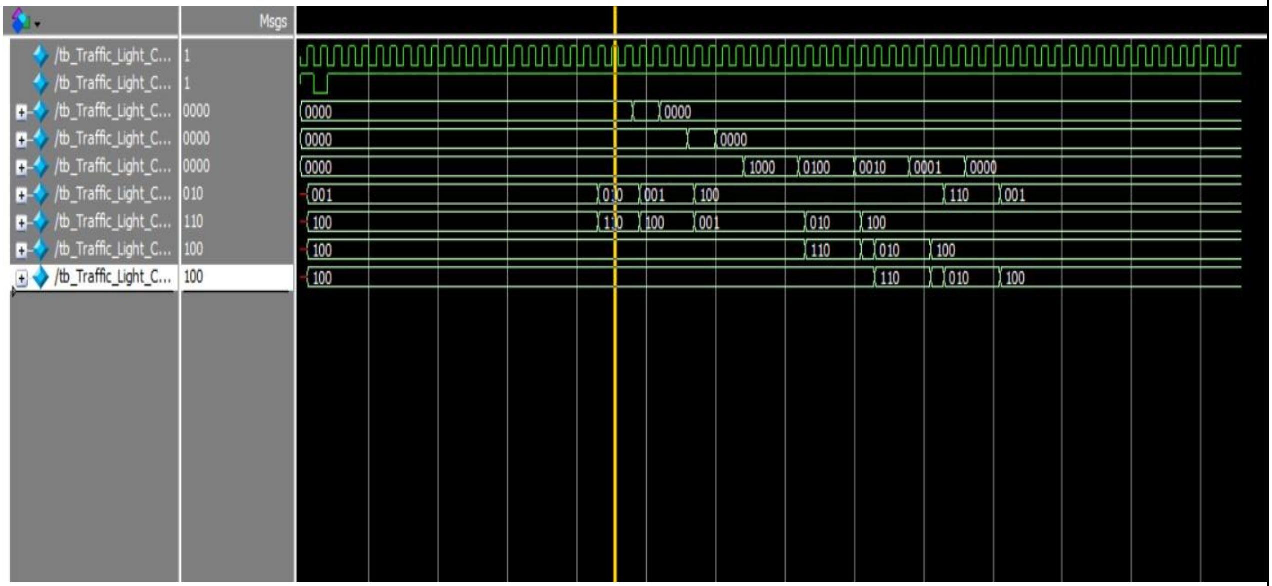


```
  reg clk;
  reg rst;
  reg [3:0] Emergency;
  reg [3:0] Jam;
  reg [3:0] Empty;

  wire [2:0] East_road;
  wire [2:0] North_road;
  wire [2:0] West_road;
  wire [2:0] South_road;

  Traffic_Light_Controller uut (
    .clk(clk),
    .rst(rst),
    .Emergency(Emergency),
    .Jam(Jam),
    .Empty(Empty),
    .East_road(East_road),
    .North_road(North_road),
    .West_road(West_road),
    .South_road(South_road)
  );


  always #5 clk = ~clk;
  initial begin

    clk = 0;
    rst = 1;
    Emergency = 4'b0000;
    Jam = 4'b0000;
    Empty = 4'b0000;
    #10;
    rst = 0;
    #10;
    rst = 1;
    #20;
    Empty = 4'b0000;
```



```
    #200;
    Emergency = 4'b1000;
    #20;
    Emergency = 4'b0000;
    #20;
    Jam = 4'b0100;
    #20;
    Jam = 4'b0000;
    #20;
    Empty = 4'b1000;
    #40;
    Empty = 4'b0100;
    #40;
    Empty = 4'b0010;
    #40;
    Empty = 4'b0001;
    #40;
    Empty = 4'b0000;
    #200;
    $stop;
  end

  initial begin
    $monitor("Time=%0t | East=%b | North=%b | West=%b | South=%b | Emergency=%b | Jam=%b | Empty=%b",
             $time, East_road, North_road, West_road, South_road, Emergency, Jam, Empty);
  end
endmodule
```

Output:


Msgs
/tb_Traffic_Light_C... 1
/tb_Traffic_Light_C... 1
/tb_Traffic_Light_C... 0000
/tb_Traffic_Light_C... 0000
/tb_Traffic_Light_C... 0000
/tb_Traffic_Light_C... 010
/tb_Traffic_Light_C... 110
/tb_Traffic_Light_C... 100
/tb_Traffic_Light_C... 100
0000 0000
0000 0000
0000 1000 0100 0010 0001 0000
001 010 001 100 110 001
100 110 100 001 010 100
100 110 010 100
100 110 010 100

6.	**Conclusions:**

Thus using the above logic we have designed a 4 way traffic control system.