A Project Report

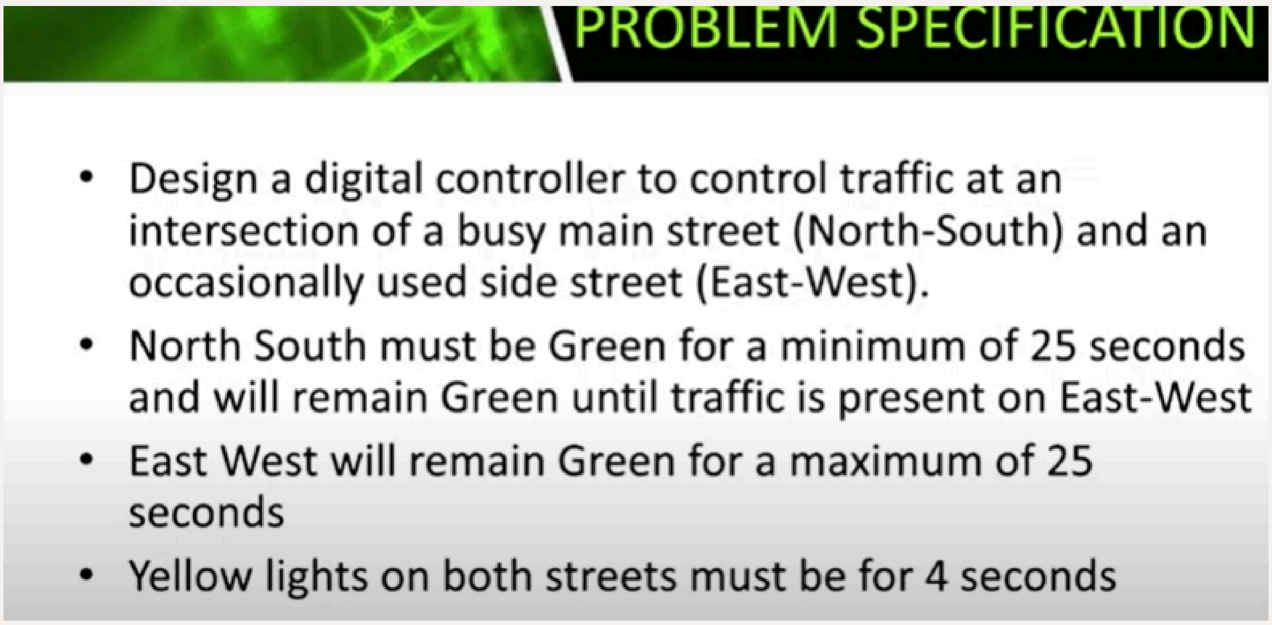
on

Traffic Light Controller

Created By -

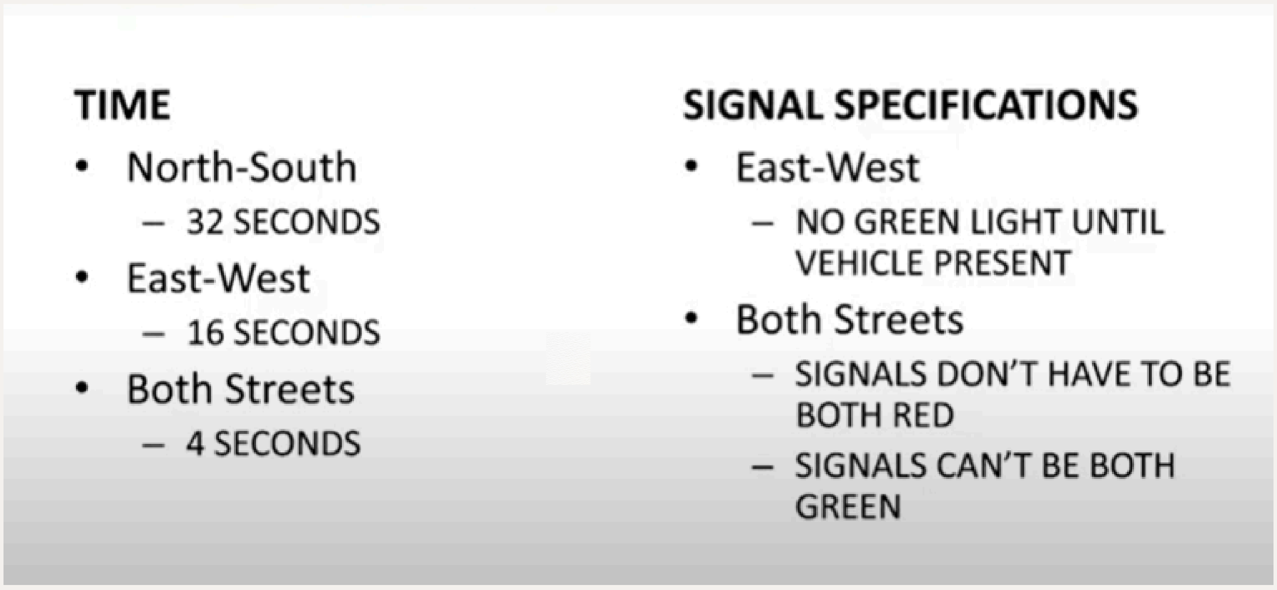
Anirban Nath
anirbannath3@gmail.com

Problem Statement



PROBLEM SPECIFICATION

- Design a digital controller to control traffic at an intersection of a busy main street (North-South) and an occasionally used side street (East-West).
- North South must be Green for a minimum of 25 seconds and will remain Green until traffic is present on East-West
- East West will remain Green for a maximum of 25 seconds
- Yellow lights on both streets must be for 4 seconds



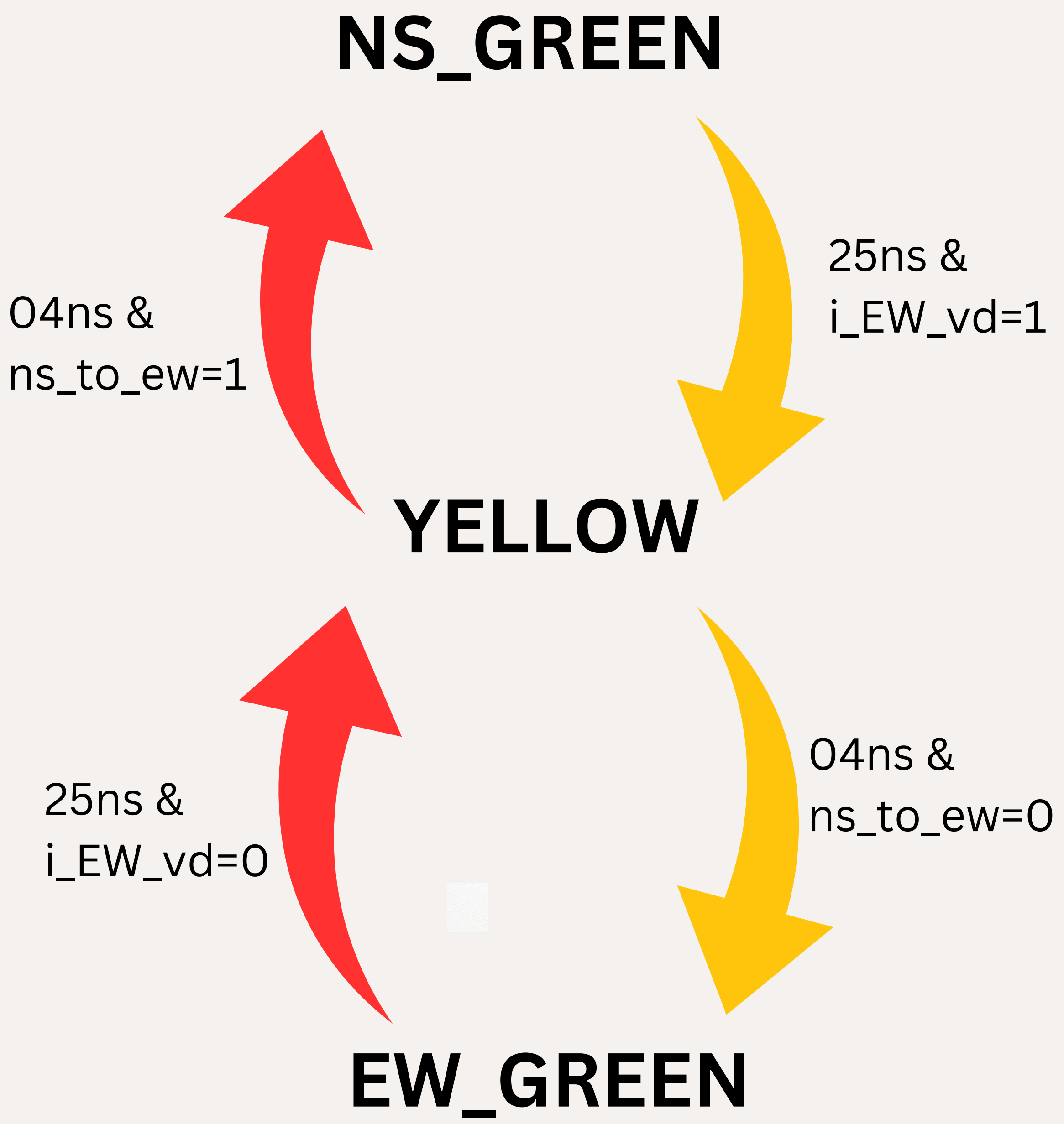
TIME

- North-South
  - 32 SECONDS
- East-West
  - 16 SECONDS
- Both Streets
  - 4 SECONDS

SIGNAL SPECIFICATIONS

- East-West
  - NO GREEN LIGHT UNTIL VEHICLE PRESENT
- Both Streets
  - SIGNALS DON'T HAVE TO BE BOTH RED
  - SIGNALS CAN'T BE BOTH GREEN

Logic


NS_GREEN
25ns &
i_EW_vd=1
04ns &
ns_to_ew=1
YELLOW
04ns &
ns_to_ew=0
25ns &
i_EW_vd=0
EW_GREEN

RTL Code

```
`timescale 1ns/1ps

module tlc_fsm(

  //inputs declarations
   input wire i_clk,
   input wire i_rst,
   input wire i_EW_vd,

   //outputs declarations
   output wire o_NS_red,
   output wire o_NS_yellow,
   output wire o_NS_green,
   output wire o_EW_red,
   output wire o_EW_yellow,
   output wire o_EW_green,
   output reg [5:0] count
);
   //State Declarations
   localparam [2:0]  NS_GREEN = 3'b001,
                     YELLOW   = 3'b010,
                     EW_GREEN = 3'b100;

   //internal register decalaration
   reg [2:0] state, next_state;
   reg r_NS_red, r_next_NS_red;
   reg r_NS_yellow, r_next_NS_yellow;
   reg r_NS_green, r_next_NS_green;
   reg r_EW_red, r_next_EW_red;
   reg r_EW_yellow, r_next_EW_yellow;
   reg r_EW_green, r_next_EW_green;
   reg ns_to_ew;
```

```
//Reset Conditions
always @(posedge i_clk) begin

      if (i_rst) begin
         r_NS_red    <= 1'b0;
         r_NS_yellow <= 1'b0;
         r_NS_green  <= 1'b1;
         r_EW_red    <= 1'b1;
         r_EW_yellow <= 1'b0;
         r_EW_green  <= 1'b0;
         state       <= NS_GREEN;
         ns_to_ew    <= 1'b0;
         count       <= 10'b0;
      end else begin

         r_NS_red    <= r_next_NS_red;
         r_NS_yellow <= r_next_NS_yellow;
         r_NS_green  <= r_next_NS_green;
         r_EW_red    <= r_next_EW_red;
         r_EW_yellow <= r_next_EW_yellow;
         r_EW_green  <= r_next_EW_green;
         state       <= next_state;

      end

      // Counter Logic
      if (i_rst || (count >= 25 && i_EW_vd && state == NS_GREEN) ||
         (count == 4 && state == YELLOW) ||
         (count >= 25 && state == EW_GREEN && ~i_EW_vd))
         count <= 0;
      else
         count <= count + 1;
   end
```

```
//Traffic Light Logic (state machine)
always @(*) begin

    r_next_NS_red    = r_NS_red;
    r_next_NS_yellow = r_NS_yellow;
    r_next_NS_green  = r_NS_green;
    r_next_EW_red    = r_EW_red;
    r_next_EW_yellow = r_EW_yellow;
    r_next_EW_green  = r_EW_green;
    next_state       = state;

    case (state)
      NS_GREEN: begin

        if (count >= 25 && i_EW_vd) begin
          next_state = YELLOW;
          ns_to_ew   = 1'b0;
        end else begin

          r_next_NS_red    = 1'b0;
          r_next_NS_yellow = 1'b0;
          r_next_NS_green  = 1'b1;
          r_next_EW_red    = 1'b1;
          r_next_EW_yellow = 1'b0;
          r_next_EW_green  = 1'b0;

        end
      end
```

```
YELLOW: begin

        if (count == 4) begin
          if (ns_to_ew)
            next_state = NS_GREEN;
          else
            next_state = EW_GREEN;
        end else begin

          r_next_NS_red    = 1'b0;
          r_next_NS_yellow = 1'b1;
          r_next_NS_green  = 1'b0;
          r_next_EW_red    = 1'b0;
          r_next_EW_yellow = 1'b1;
          r_next_EW_green  = 1'b0;

        end
      end

     EW_GREEN: begin

        if (count >= 25 && ~i_EW_vd) begin
          next_state = YELLOW;
          ns_to_ew   = 1'b1;
        end else begin

          r_next_NS_red    = 1'b1;
          r_next_NS_yellow = 1'b0;
          r_next_NS_green  = 1'b0;
          r_next_EW_red    = 1'b0;
          r_next_EW_yellow = 1'b0;
          r_next_EW_green  = 1'b1;

        end
      end
```

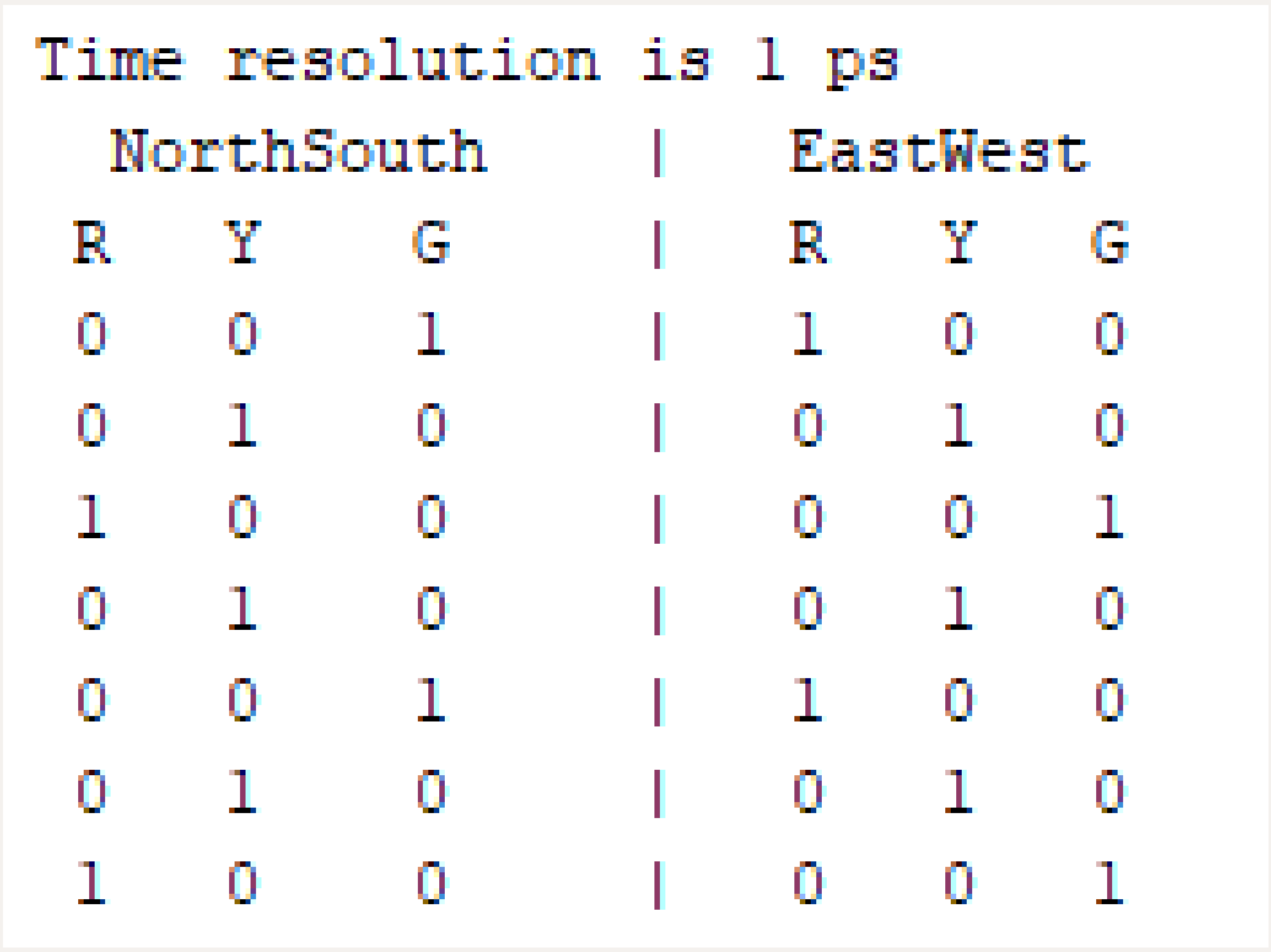
```
default: begin

            r_next_NS_red    = 1'b0;
            r_next_NS_yellow = 1'b0;
            r_next_NS_green  = 1'b1;
            r_next_EW_red    = 1'b1;
            r_next_EW_yellow = 1'b0;
            r_next_EW_green  = 1'b0;
            next_state      = NS_GREEN;
            ns_to_ew        = 1'b0;

         end
      endcase
   end


   //output assignments
   assign o_NS_red    = r_next_NS_red;
   assign o_NS_yellow = r_next_NS_yellow;
   assign o_NS_green  = r_next_NS_green;
   assign o_EW_red    = r_next_EW_red;
   assign o_EW_yellow = r_next_EW_yellow;
   assign o_EW_green  = r_next_EW_green;

endmodule
```

Testbench

```
`timescale 1ns / 1ps

module traffic_tb();

 // Inputs
 reg i_clk;
 reg i_rst;
 reg i_EW_vd;

 // Outputs
 wire o_NS_red;
 wire o_NS_yellow;
 wire o_NS_green;
 wire o_EW_red;
 wire o_EW_yellow;
 wire o_EW_green;
 wire [5:0] count;

 // Instantiate Traffic Light Controller
 tlc_fsm uut (
  .i_clk(i_clk),
  .i_rst(i_rst),
  .i_EW_vd(i_EW_vd),
  .o_NS_red(o_NS_red),
  .o_NS_yellow(o_NS_yellow),
  .o_NS_green(o_NS_green),
  .o_EW_red(o_EW_red),
  .o_EW_yellow(o_EW_yellow),
  .o_EW_green(o_EW_green),
  .count(count)
 );
```

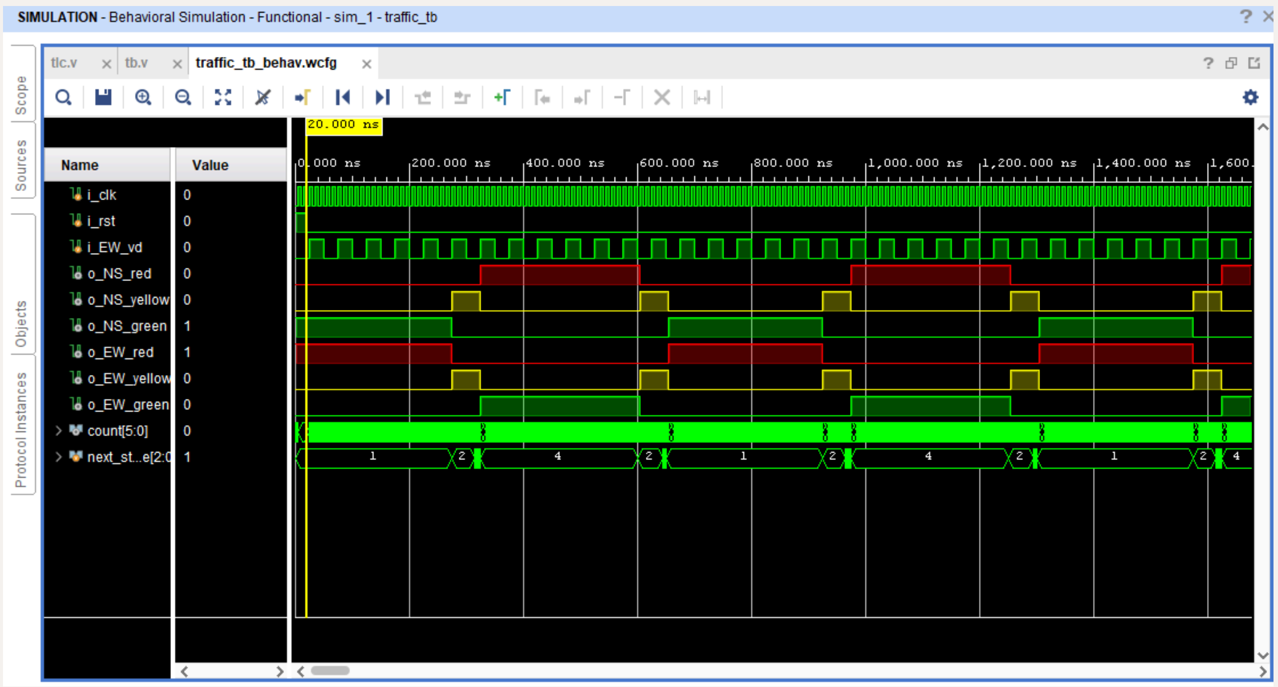
```
/// Clock generation
  always #5 i_clk = ~i_clk;
  always #25 i_EW_vd=~i_EW_vd;

  // Initial block for reset and input signal generation
  initial begin

    i_clk = 0;
    i_rst = 1;
    i_EW_vd = 0;
    #20 i_rst =0 ;

end

  // Monitor traffic light states
  initial begin
    $display("  NorthSouth    |   EastWest  ");
    $display(" R   Y    G     |   R   Y    G");
    $monitor(" %b  %b   %b    |   %b  %b   %b",
      o_NS_red, o_NS_yellow, o_NS_green,
      o_EW_red, o_EW_yellow, o_EW_green);

    // Simulate for a period of time
    #5000 $finish;
  end


endmodule
```

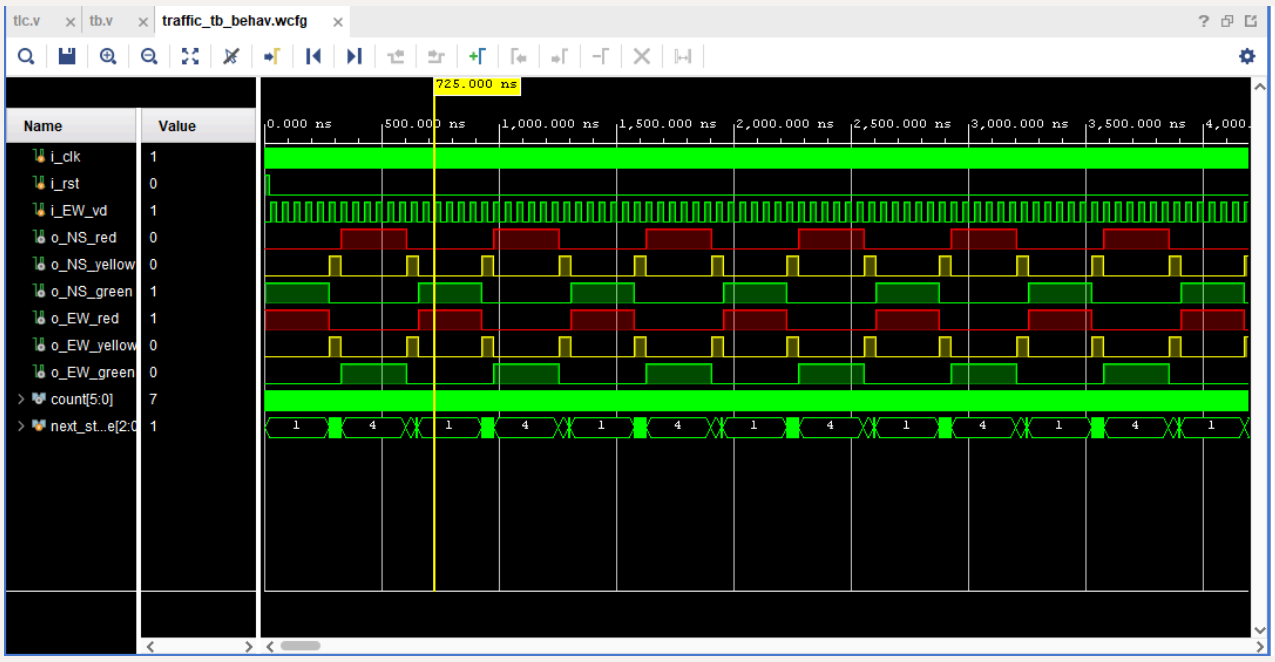
Output



```
Time resolution is 1 ps
  NorthSouth          |    EastWest
 R     Y      G       |   R     Y     G
 0     0      1       |   1     0     0
 0     1      0       |   0     1     0
 1     0      0       |   0     0     1
 0     1      0       |   0     1     0
 0     0      1       |   1     0     0
 0     1      0       |   0     1     0
 1     0      0       |   0     0     1
```

Waveforms


SIMULATION - Behavioral Simulation - Functional - sim_1 - traffic_tb
tlc.v
tb.v
traffic_tb_behav.wcfg
20.000 ns
Name
Value
i_clk 0
i_rst 0
i_EW_vd 0
o_NS_red 0
o_NS_yellow 0
o_NS_green 1
o_EW_red 1
o_EW_yellow 0
o_EW_green 0
count[5:0] 0
next_st...e[2:0] 1
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
1,200.000 ns
1,400.000 ns
Scope
Sources
Objects
Protocol Instances


tlc.v
tb.v
traffic_tb_behav.wcfg
725.000 ns
Name
Value
i_clk 1
i_rst 0
i_EW_vd 1
o_NS_red 0
o_NS_yellow 0
o_NS_green 1
o_EW_red 1
o_EW_yellow 0
o_EW_green 0
count[5:0] 7
next_st...e[2:0] 1
0.000 ns
500.000 ns
1,000.000 ns
1,500.000 ns
2,000.000 ns
2,500.000 ns
3,000.000 ns
3,500.000 ns

Thank You