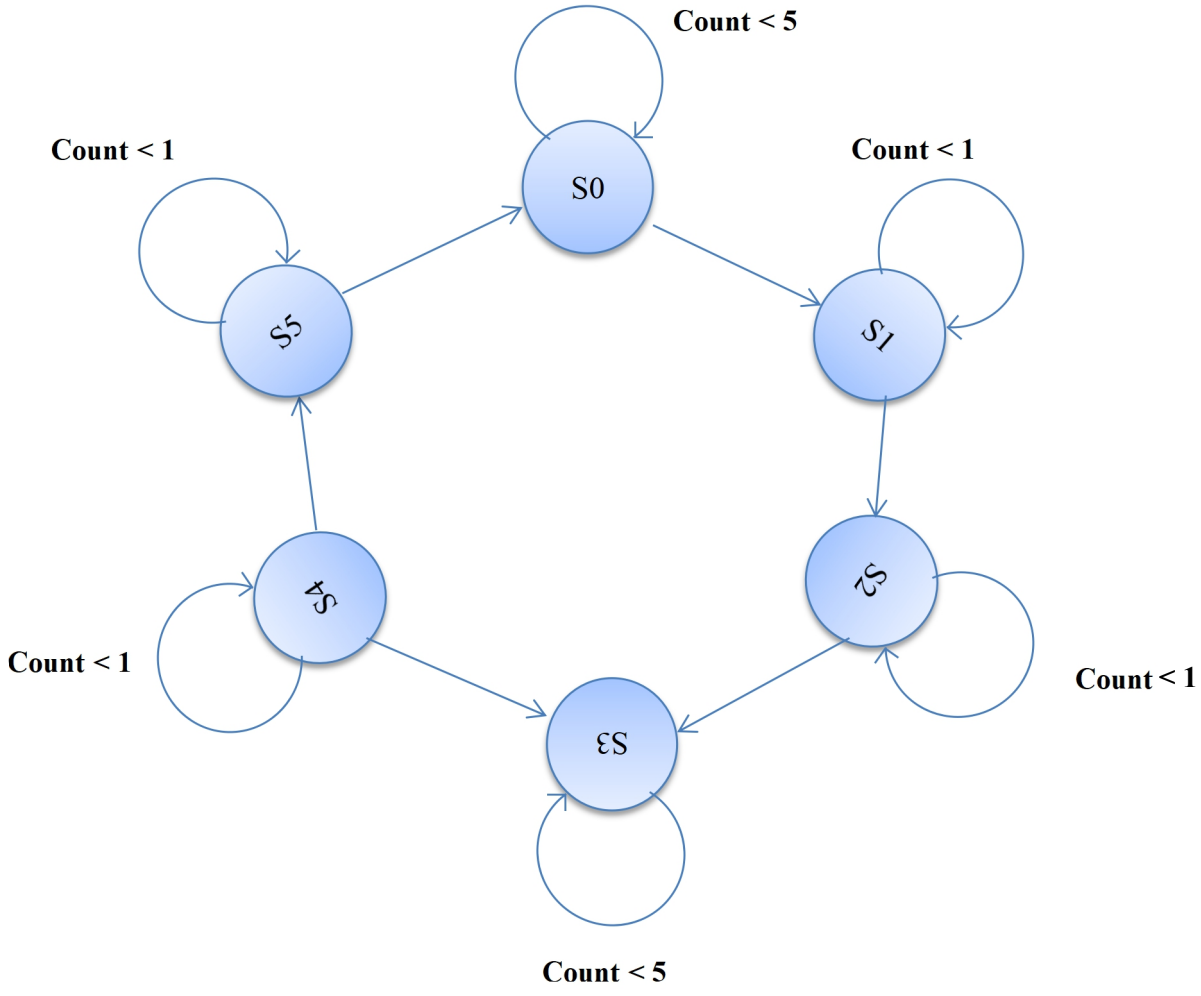
4 Way Traffic Control System
(Using FSM)

- **State Table :-**

State	EW-light	NS-light	Delay
S0	Red	Green	5 sec
S1	Red	Yellow	1 sec
S2	Red	Red	1 sec
S3	Green	Red	5 sec
S4	Yellow	Red	1 sec
S5	Red	Red	1 sec

- **State Diagram :-**

  - Here we are using the clock of **"1Hz"** as input so the time period of 1 cycle will be 1 second.
  - So after 1 cycle and 5 cycles of clock we automatically gets a delay of 1 second and 5seconds respectively.
  - The variable **"count"** will count the values of cycles of clock.



- **Code:-**

```
// 4-WAY Traffic control using FSM (Finite State Machine)

module tr_lights(clk,rst,EW_lights,NS_lights,state);
input wire clk;
input wire rst;
output reg [1:0] EW_lights;   // East - West Lights
output reg [1:0] NS_lights;   // North - South Lights
output reg [2:0] state;

reg [3:0] count;

// STATES
parameter S0 = 3'b000;
parameter S1 = 3'b001;
parameter S2 = 3'b010;
parameter S3 = 3'b011;
parameter S4 = 3'b100;
parameter S5 = 3'b101;

// DELAYS
parameter sec1 = 4'b0001; // 1 sec delay
parameter sec5 = 4'b0101; // 5 sec delay

// LIGHTS
parameter RED = 00;
parameter YELLOW = 01;
parameter GREEN = 10;

always @(posedge clk or rst)
begin
  if (!rst)
  begin
  case(state)
     S0: if(count < sec5)
                begin
                state <= S0;
                count <= count + 1;
                end
           else
                begin
                state <= S1;
```

```
        count <= 0;
        end

S1: if(count < sec1)
        begin
        state <= S1;
        count <= count + 1;
        end
    else
        begin
        state <= S2;
        count <= 0;
        end
S2: if(count < sec1)
        begin
        state <= S2;
        count <= count + 1;
        end
    else
        begin
        state <= S3;
        count <= 0;
        end
S3: if(count < sec5)
        begin
        state <= S3;
        count <= count + 1;
        end
    else
        begin
        state <= S4;
        count <= 0;
        end
S4: if(count < sec1)
        begin
        state <= S4;
        count <= count + 1;
        end
    else
        begin
        state <= S5;
        count <= 0;
        end
```

```
        S5: if(count < sec1)
                begin
                state <= S5;
                count <= count + 1;
                end
            else
                begin
                state <= S0;
                count <= 0;
                end

        default state <= S0;
        endcase
    end
  else
        begin
        state <= S0;
        count <= 0;
        end
end

 always @(*)
 begin
    case(state)
        S0: begin
                EW_lights = RED;
                NS_lights = GREEN;
            end

        S1: begin
                EW_lights = RED;
                NS_lights = YELLOW;
          end

        S2: begin
                EW_lights = RED;
                NS_lights = RED;
          end

        S3: begin
                EW_lights = GREEN;
                NS_lights = RED;
          end
```

```
        S4: begin
                EW_lights = YELLOW;
                NS_lights = RED;
             end

        S5: begin
                EW_lights = RED;
                NS_lights = RED;
             end

             default begin
                EW_lights = RED;
                NS_lights = GREEN;
               end
    endcase
 end

endmodule
```

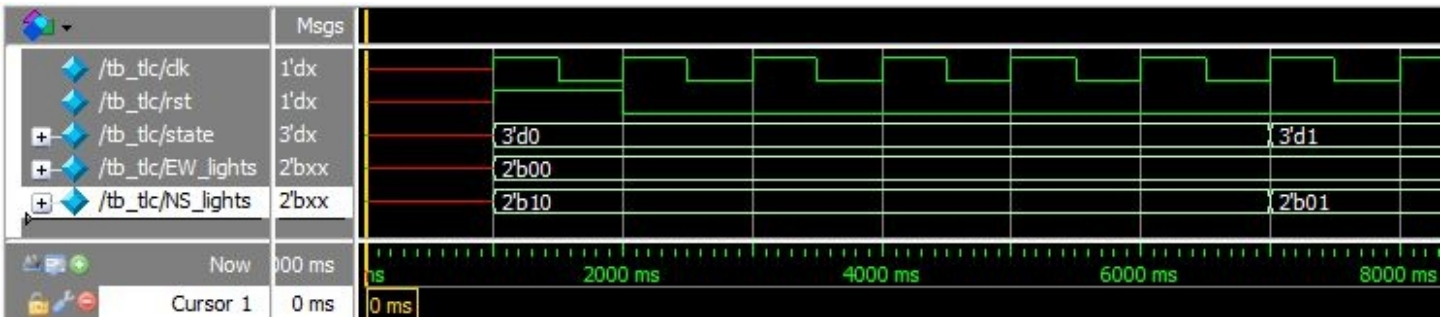
Test Bench:-

```
// Test Bench for Traffic Light Controler

`timescale 1ms/1ms
module tb_tlc;
reg clk;
reg rst;
wire [2:0] state;
wire [1:0] EW_lights;   // East - West Lights
wire [1:0] NS_lights;   // North - South Lights

tr_lights uut(.clk(clk),
              .rst(rst),
              .EW_lights(EW_lights),
              .NS_lights(NS_lights),
              .state(state)
);

always #500 clk = ~clk;
initial
```

```
 begin
  #1000;
  clk <= 1;
  rst <= 1; #1000;
  rst <= 0;
  end

endmodule
```

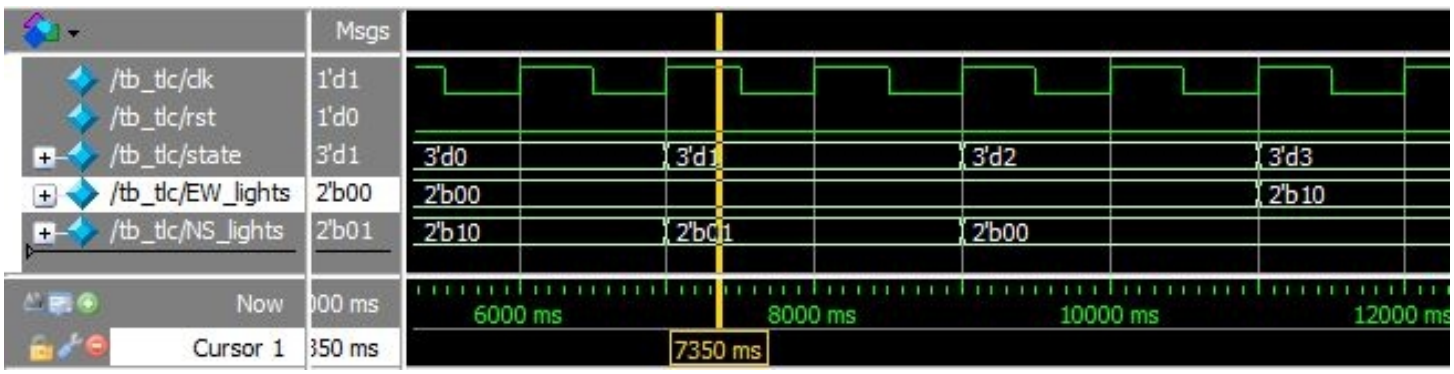
Waveforms:-

Initially for 1st second all values are unknown and then at the 2nd sec the clk is given and the rst signal is set high so the state becomes S0 and hence:
EW_lights = RED (00)
NS_lights = GREEN (10)



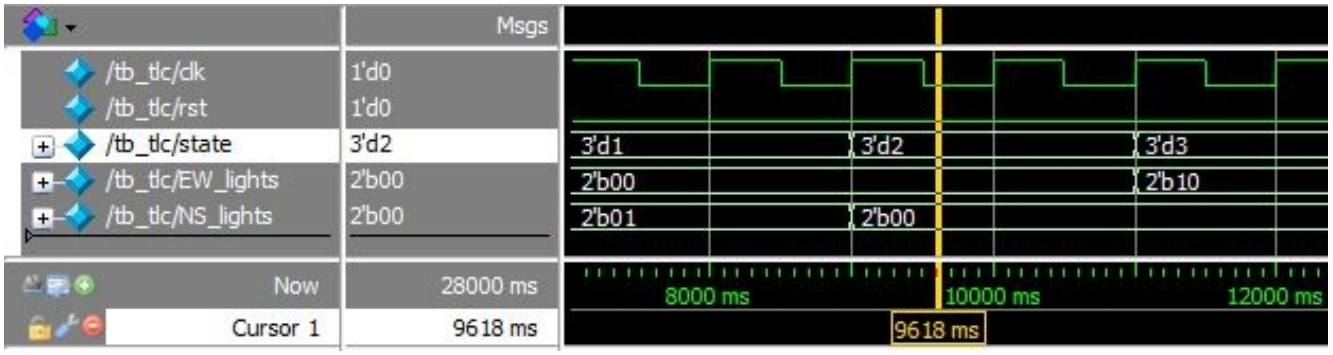
The system then remains in the S0 state for next 5 cycles (i.e. 5 sec) after that it goes in:

State: S1
EW_lights : RED (00)
NS_lights : YELLOW (01)

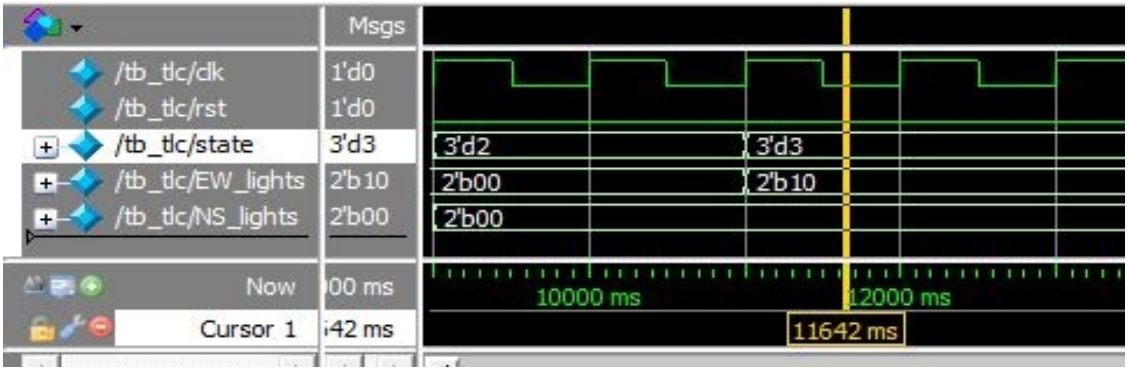


The system then remains in the S1 state for next 1 cycles (i.e. 1 sec) after that it goes in:

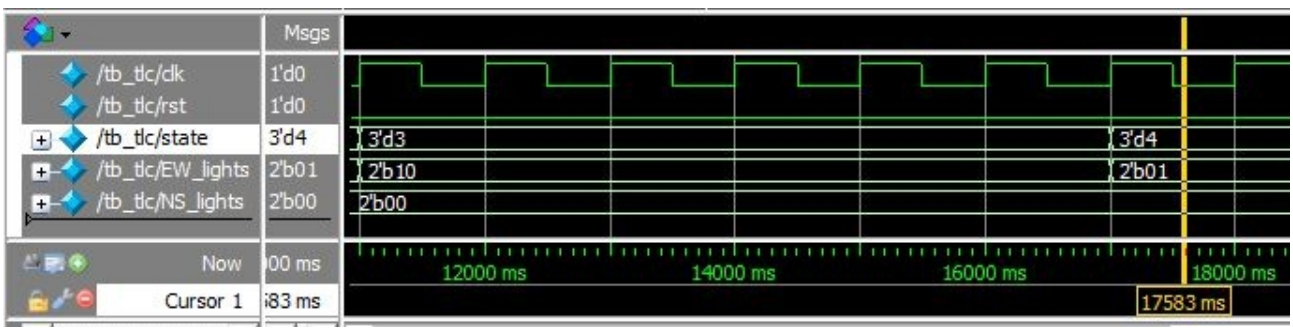
State: S2
EW_lights : RED (00)
NS_lights : RED (00)



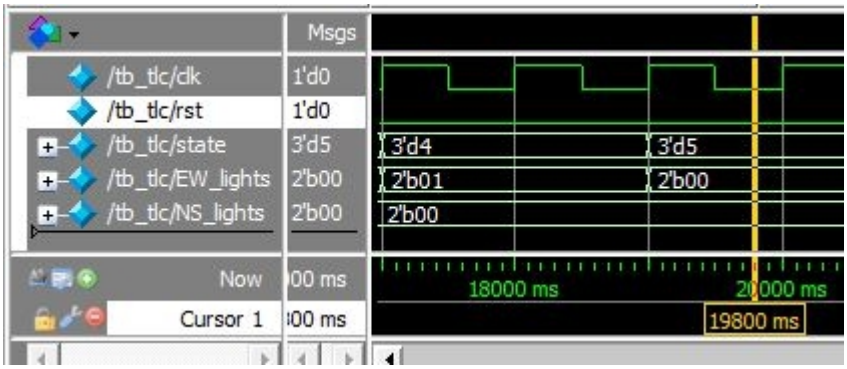
Then after 1 clock cycle (1 sec) we have:
State: S3
EW_lights : GREEN (10)
NS_lights : RED (00)



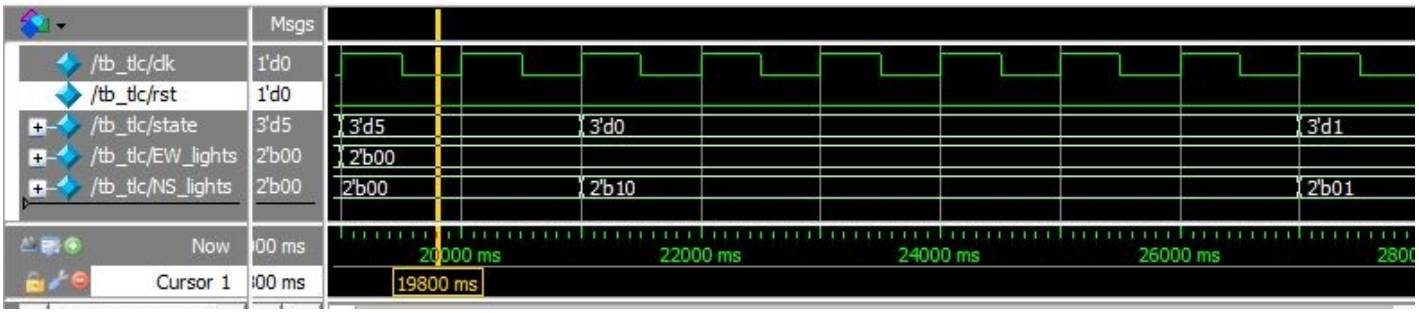
Then after 5 clock cycle (5 sec) we have:
State: S4
EW_lights : YELLOW (01)
NS_lights : RED (00)



Then after 5 clock cycle (5 sec) we have:
State: S4
EW_lights : RED(00)
NS_lights : RED (00)



Then after 1 clock cycle (1 sec) we have:
State: S5
EW_lights : RED (00)
NS_lights : GREEN (00)



And the process keeps on repeating itself as we can see in the waveform above.

THE END