TRAFFIC SIGNAL CONTROLLER

NAME: HANUMANTHRAO (PES2UG22EC059)

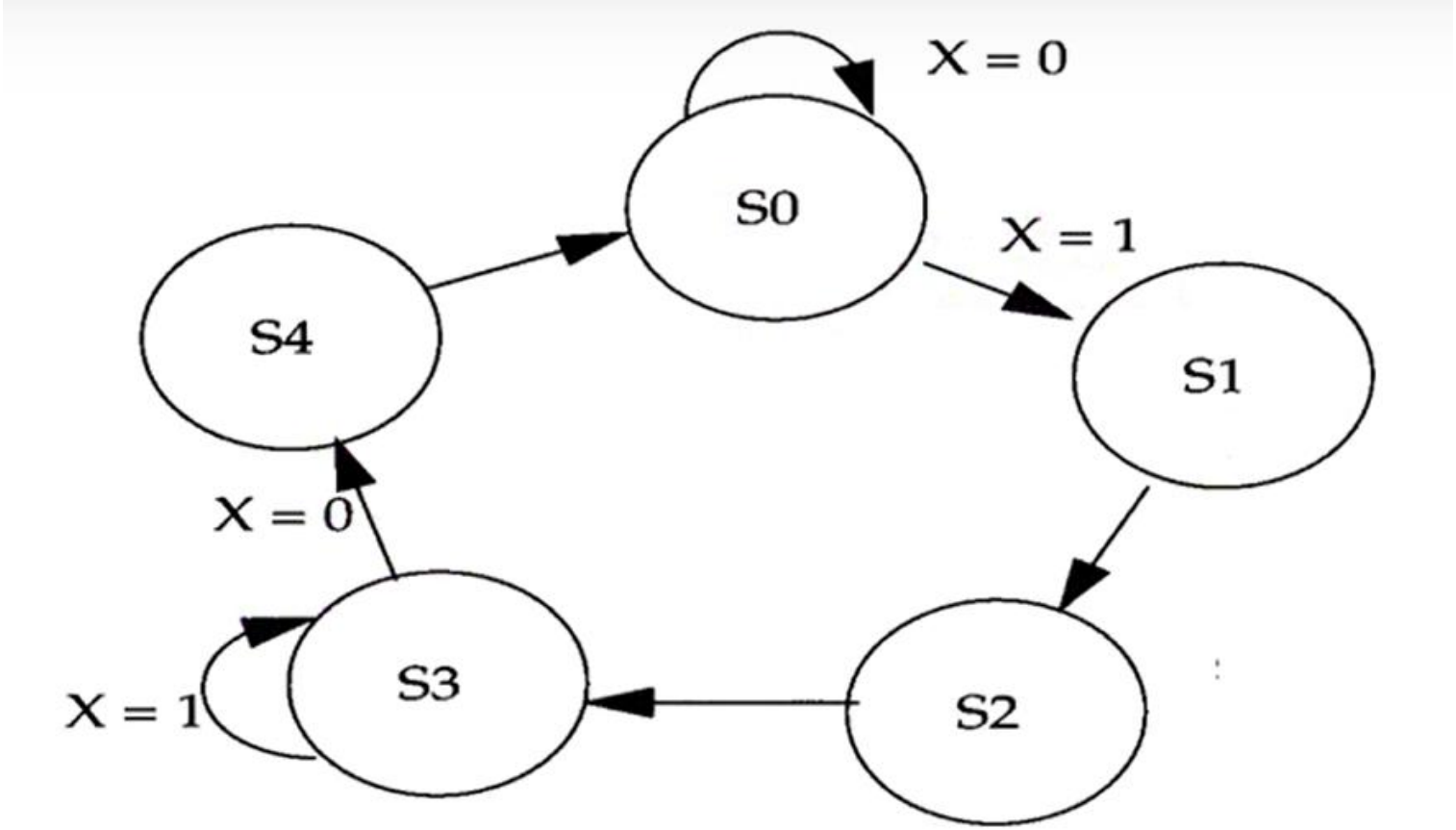
K.P RAGHAVENDRA (PES2UG22EC063)

GUIDE: Prof. Vinay Reddy

TA: Garima Bajpayi

TEAM NO :17

FINITE STATE MACHINE:



DUT CODE:

design.sv

```
module traffic(

  input logic clk,
  input logic reset,
  output logic [2:0] light
);

  typedef enum logic [1:0] {
    S_RED = 2'b00,
    S_GREEN = 2'b01,
    S_YELLOW = 2'b10
  } state_t;

  state_t state, next_state;
  logic [3:0] count;


  always_ff @(posedge clk or posedge reset) begin
    if (reset)
      state <= S_RED;
    else
      state <= next_state;
  end


  always_comb begin
    next_state = state;
    case (state)
      S_RED: if (count == 4'd10) next_state = S_GREEN;
      S_GREEN: if (count == 4'd10) next_state = S_YELLOW;
      S_YELLOW: if (count == 4'd5) next_state = S_RED;
      default: next_state = S_RED;
    endcase
  end

  always_ff @(posedge clk or posedge reset) begin
    if (reset) begin
      count <= 0;
      light <= 3'b100;
    end else begin
      count <= count + 1;
      case (state)
        S_RED: light <= 3'b100;
        S_GREEN: light <= 3'b001;
        S_YELLOW: light <= 3'b010;
      endcase
    end
  end

endmodule
```

PACKET CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv | traffic_monitor.sv | traffic_scoreboard.sv | tra

```
1  class traffic_packet;
2
3    // Input stimulus
4    rand bit reset;
5
6    // Expected output
7    rand bit [2:0] expected_light;
8
9    // Constraints (if needed)
10   constraint expected_light_c {
11     expected_light inside {3'b100, 3'b010, 3'b001};  // RED, YELLOW, GREEN
12   }
13
14   // Constructor
15   function new();
16     reset = 0;
17     expected_light = 3'b000;
18   endfunction
19
20   // Print the packet details
21   function void display();
22     $display("Traffic Packet :: reset = %b, expected_light = %03b", reset, expected_light);
23   endfunction
24
25 endclass
```

GENERATOR CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv | traffic_monitor.sv

```
1  class traffic_generator;
2
3    mailbox #(traffic_packet) packet_queue;
4
5    function new(mailbox #(traffic_packet) packet_queue);
6      this.packet_queue = packet_queue;
7    endfunction
8
9    task g1();
10     traffic_packet pkt;
11     repeat (10) begin  // Generate 10 packets
12       pkt = new();
13       pkt.randomize();  // Randomize packet fields
14       packet_queue.put(pkt);
15     end
16   endtask
17
18 endclass
```

DRIVER CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv

```
1  class traffic_driver;
2
3    virtual traffic_if.TB vif;
4
5    function new(virtual traffic_if.TB vif);
6      this.vif = vif;
7    endfunction
8
9    task drive(traffic_packet pkt);
10     vif.reset = pkt.reset;
11     @(posedge vif.clk);
12     vif.reset = 0;  // De-assert reset after one clock cycle
13   endtask
14
15 endclass
```

INTERFACE CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv

```
1  interface traffic_if(input logic clk);
2    logic reset;
3    logic [2:0] light;
4
5    // Modports
6    modport DUT (input reset, clk, output light);
7    modport TB (output reset, input clk, input light);
8
9  endinterface
```

MONITOR CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv | traffic_monitor.sv | traffic_

```
1  class traffic_monitor;
2
3    virtual traffic_if.TB vif;
4    mailbox #(traffic_packet) monitor_queue;
5
6    function new(virtual traffic_if.TB vif, mailbox #(traffic_packet) monitor_queue);
7      this.vif = vif;
8      this.monitor_queue = monitor_queue;
9    endfunction
10
11   task monitor();
12     traffic_packet pkt;
13     forever begin
14       @(posedge vif.clk);
15       pkt = new();
16       pkt.expected_light = vif.light;
17       monitor_queue.put(pkt);
18     end
19   endtask
20
21 endclass
```

SCOREBOARD CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv | traffic_monitor.sv | traffic_scoreboard.sv

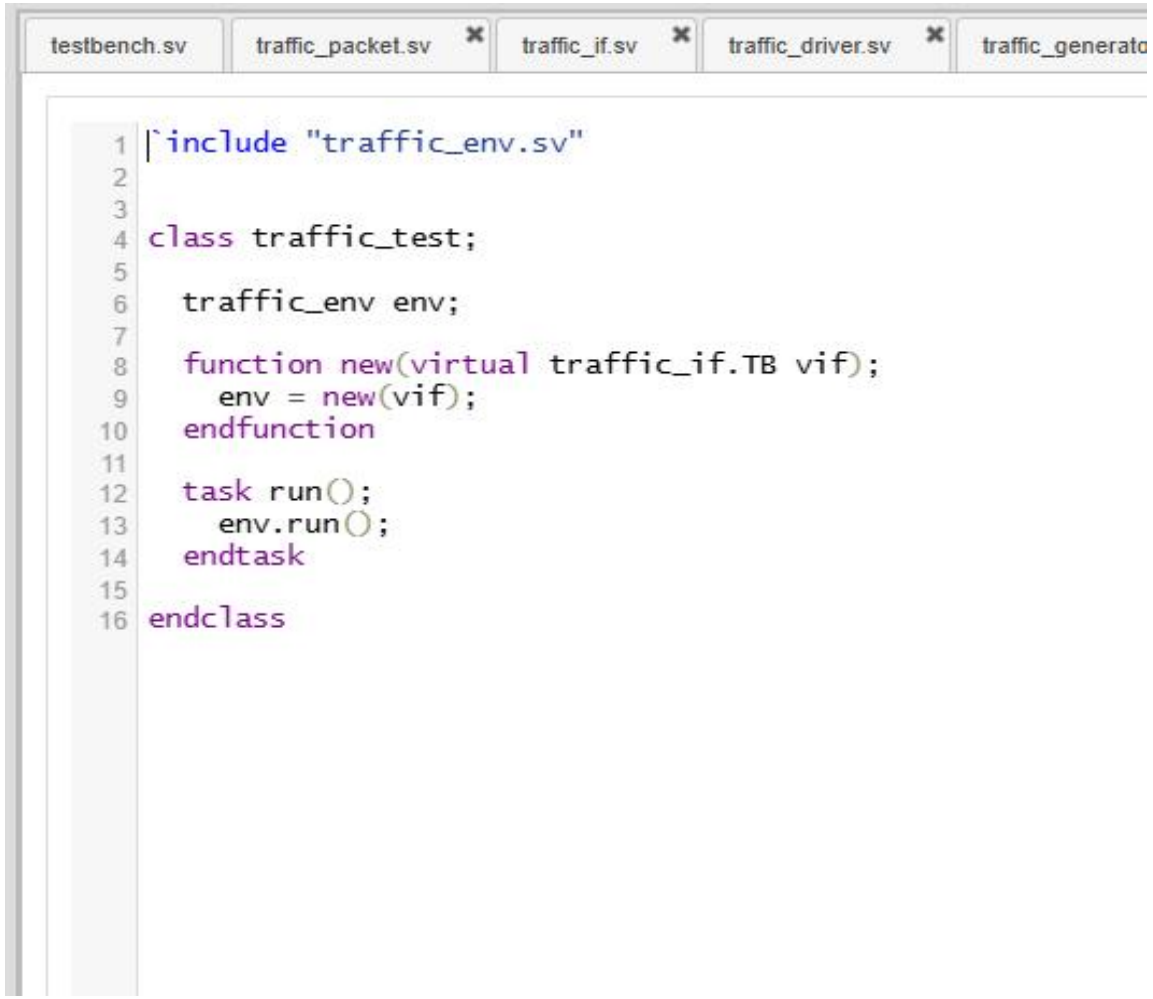
```
1  class traffic_scoreboard;
2
3    mailbox #(traffic_packet) monitor_queue;
4    int pass_count = 0, fail_count = 0;
5
6    function new(mailbox #(traffic_packet) monitor_queue);
7      this.monitor_queue = monitor_queue;
8    endfunction
9
10   task compare();
11     traffic_packet pkt;
12     forever begin
13       monitor_queue.get(pkt);  // Retrieve monitored packet
14       if (pkt.expected_light inside {3'b100, 3'b010, 3'b001}) begin
15         pass_count++;
16         $display("PASS: Expected light = %03b", pkt.expected_light);
17       end else begin
18         fail_count++;
19         $display("FAIL: Unexpected light = %03b", pkt.expected_light);
20       end
21     end
22   endtask
23
24 endclass
```

ENVIRONMENT CODE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv | traffic_monitor.sv | traffic_scoreboard.sv | traffic_env.sv

```
1  `include "traffic_packet.sv"
2  `include "traffic_generator.sv"
3  `include "traffic_driver.sv"
4  `include "traffic_monitor.sv"
5  `include "traffic_scoreboard.sv"
6
7
8  class traffic_env;
9
10   traffic_generator generator;
11   traffic_driver driver;
12   traffic_monitor monitor;
13   traffic_scoreboard scoreboard;
14
15   mailbox #(traffic_packet) packet_queue;
16   mailbox #(traffic_packet) monitor_queue;
17
18   virtual traffic_if.TB vif;
19
20   function new(virtual traffic_if.TB vif);
21     this.vif = vif;
22     packet_queue = new();
23     monitor_queue = new();
24     generator = new(packet_queue);
25     driver = new(vif);
26     monitor = new(vif, monitor_queue);
27     scoreboard = new(monitor_queue);
28   endfunction
29
30   task run();
31     fork
32       generator.g1();
33       monitor.monitor();
34       scoreboard.compare();
35       drive();
36     join
37   endtask
38
39   task drive();
40     traffic_packet pkt;
41     while (packet_queue.num() > 0) begin
42       packet_queue.get(pkt);
43       driver.drive(pkt);
44     end
45   endtask
46
47 endclass
```

TEST CODE:


testbench.sv
traffic_packet.sv
traffic_if.sv
traffic_driver.sv
traffic_generato
1 `include "traffic_env.sv"
2
3
4 class traffic_test;
5
6 traffic_env env;
7
8 function new(virtual traffic_if.TB vif);
9 env = new(vif);
10 endfunction
11
12 task run();
13 env.run();
14 endtask
15
16 endclass

TESTBENCH TOPLEVEL MODULE:

testbench.sv | traffic_packet.sv | traffic_if.sv | traffic_driver.sv | traffic_generator.sv | traffic_mo

```
1  `include "traffic_if.sv"
2  `include "test.sv"
3
4
5
6  module traffic_tb;
7
8    logic clk;
9    traffic_if tb_if(clk);
10
11   traffic dut(
12     .clk(tb_if.clk),
13     .reset(tb_if.reset),
14     .light(tb_if.light)
15   );
16
17   // Clock generation
18   initial begin
19     clk = 0;
20     forever #5 clk = ~clk;  // 10 ns clock period
21   end
22
23   // Run the test
24   initial begin
25     traffic_test test = new(tb_if.TB);
26     test.run();
27     #1000;  // Run simulation for 1000 time units
28     $finish;
29   end
30
31 endmodule
```

OUTPUT SNAPSHOTS:

```
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 100
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
PASS: Expected light = 001
```

PES UNIVERSITY
CELEBRATING 50 YEARS

THANK YOU