PIMPRI CHINCHWAD EDUCATION TRUST'S

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING SECTOR NO. 26, PRADHIKARAN, NIGDI, PUNE-44

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION S.Y. (E&TC) 2023-24

DIGITAL DESIGN USING VERILOG HDL

FORMATIVE ASSESSMENT - 2

Problem Statement

Test Specification:Write a test bench to simulate the traffic light controller, verifying correct light sequences and state durations. Test various scenarios, including a reset condition and normal operation.

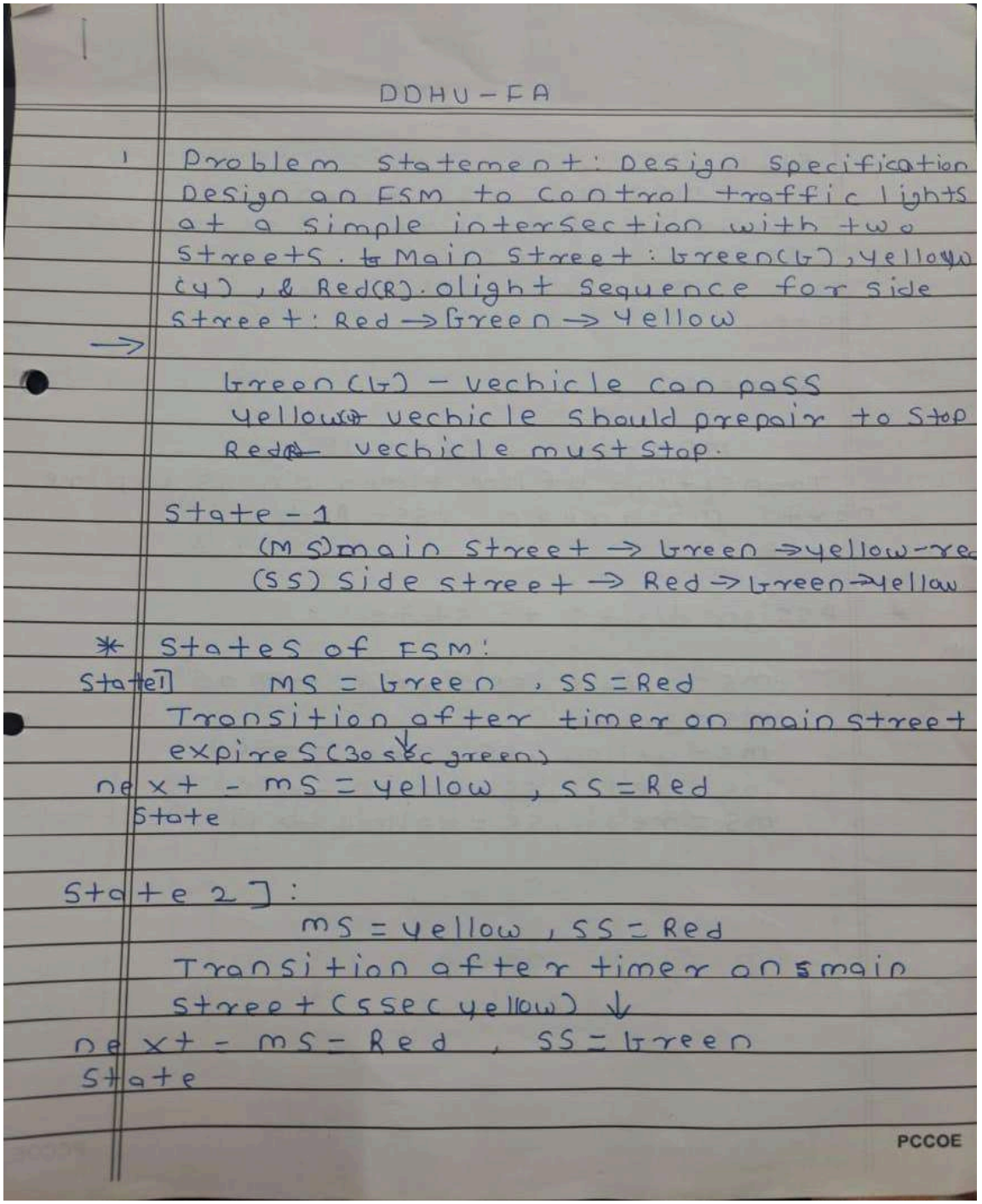
Under the Guidance of Mrs.Deepti Khurge mam

NAME:Suyash Ghadage
PRN NO:122B1E054 ENTC A

1. Introduction : A Finite State Machine (FSM) for Traffic Light Control in VHDL is a digital system designed to regulate the traffic signals at an intersection. The FSM operates by transitioning between different states that represent the lights: green, yellow, and red. Each state defines which light is active and for how long, ensuring smooth control of vehicle and pedestrian traffic.

2. Design Specification :

1.Design Specification :Design an FSM to control traffic lights at a simple intersection with two streets: Main Street (MS) and Side Street (SS). oStates for each street: Green (G), Yellow (Y), and Red (R). oLight sequence for Main Street: Green → Yellow → Red. oLight sequence for Side Street: Red → Green → Yellow.



State 3:

MS = Red , SS = Green

Transition after timer on ss expires (30 sec for green)

next state MS = Red , SS = yellow

State 4:

MS = red , SS = yellow

Transition after timer on ss expi

next state MS = Green , SS = Red.

* Assign digits to state:

MS = green , SS = red → 00

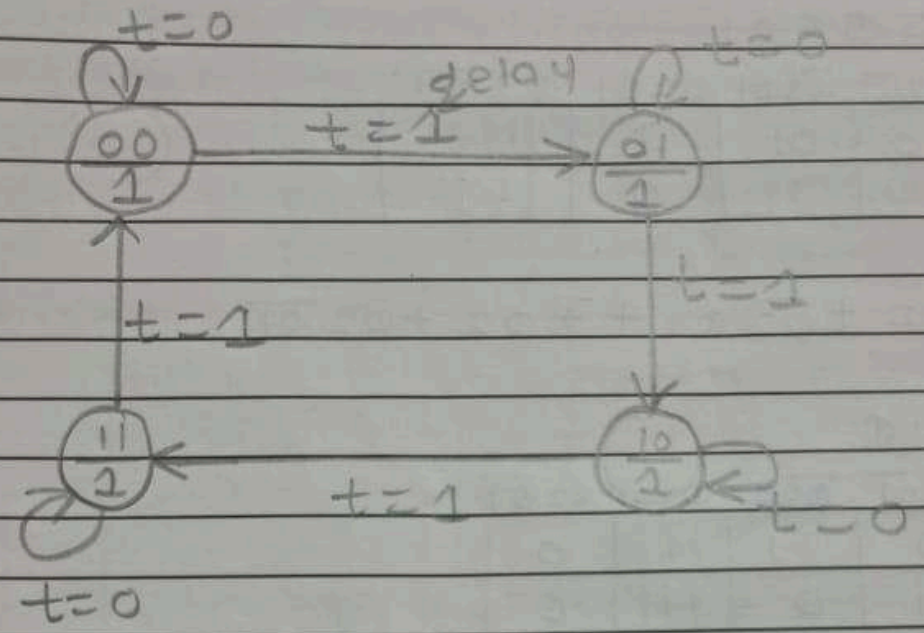
MS = green

MS = yellow , SS = red → 01

MS = red , SS = Green → 10

MS = red , SS = yellow → 11

* moore state diagram:



* Truth table:

t	Q_2	Q_1	Q_2^+	Q_1^+	D_2^+	D_1^+	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	1

* Kmap:

$D_2^+ = Q$

	$\bar{Q}_2\bar{Q}_1$	$\bar{Q}_2Q_1$	Q_2Q_1	$Q_2\bar{Q}_1$
$\bar{t}$	0	0	1	1
t	0	1	0	1

$D_2^+ = t\bar{Q}_2Q_1 + \bar{t}Q_2 + Q_2\bar{Q}_1$

$D_1^+ = Q$

	$\bar{Q}_2\bar{Q}_1$	$\bar{Q}_2Q_1$	Q_2Q_1	$Q_2\bar{Q}_1$
$\bar{t}$	0	1	1	0
t	1	0	1	0

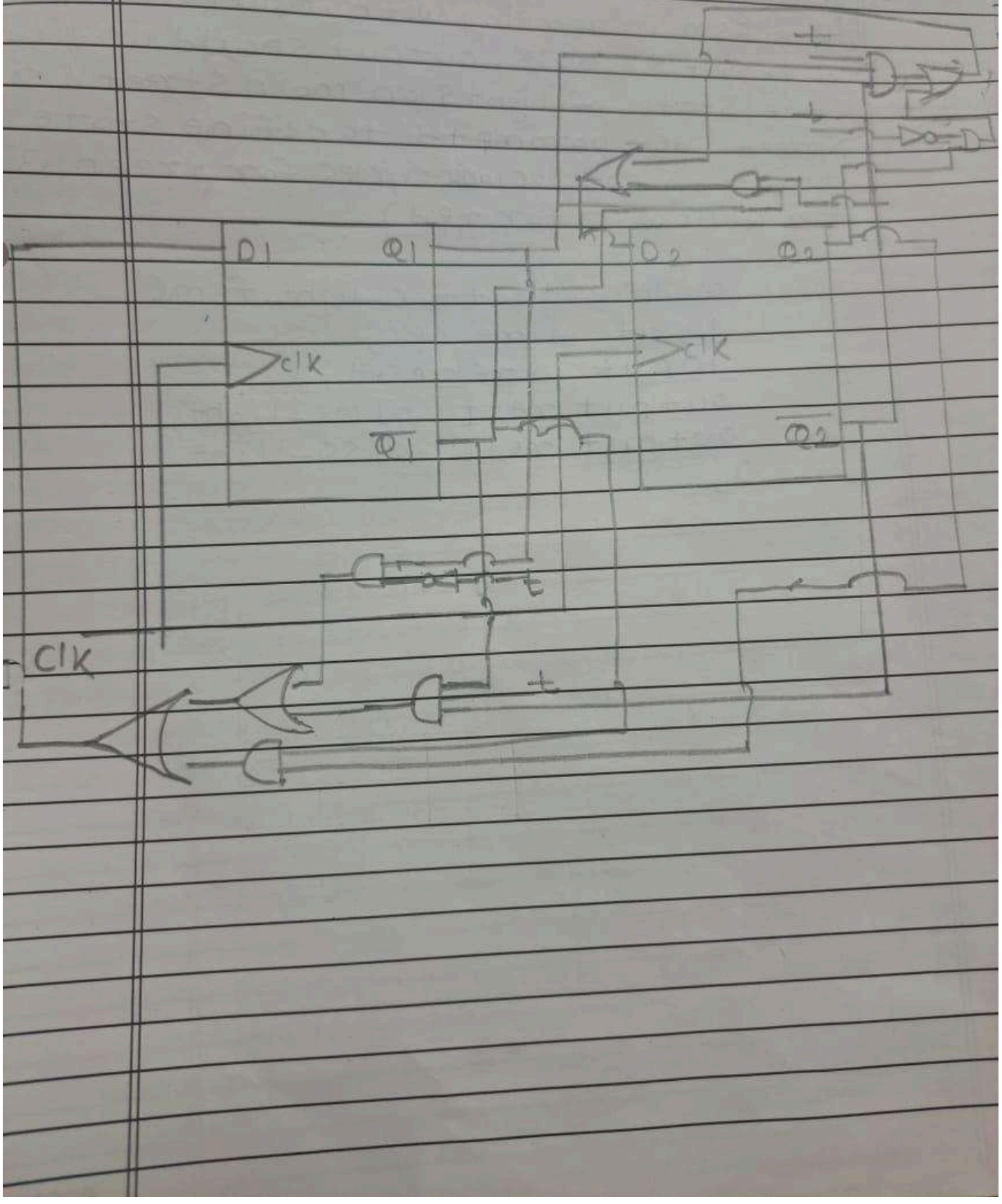
$D_1^+ = t\bar{Q}_2\bar{Q}_1 + \bar{t}Q_1 + Q_2Q_1$

$Y = Q$.

	$\bar{Q}_2\bar{Q}_1$	$\bar{Q}_2Q_1$	Q_2Q_1	$Q_2\bar{Q}_1$
$\bar{t}$	0	0	0	0
t	1	1	1	1

$Y = Q = t$.

* circuit diagram:



3. Test Bench:

We used a device from Artix-7 Family for this design.

Verilog Code Using Case Statement: Verilog Code Implementation:Implement the FSM in Verilog with inputs for the clock and reset signals.,Outputs should indicate the state of the lights on Main Street and Side Street.Use parameters to define state durations (e.g., 10 clock cycles for green, 3 for yellow, 7 for red).A test bench to simulate the traffic light controller, verifying correct light sequences and state durations. Test various scenarios, including a reset condition and normal operation

Verilog code :module tb_traffic_light_controller;

```
  // Testbench signals
  reg clk;
  reg reset;
  wire [1:0] main_street_light;
  wire [1:0] side_street_light;

  // Clock period
  localparam CLK_PERIOD = 10;

  // Instantiate the traffic light controller
  traffic_light_controller uut (
      .clk(clk),
      .reset(reset),
      .main_street_light(main_street_light),
      .side_street_light(side_street_light)
  );

  // Clock generation
  initial begin
      clk = 0;
      forever #(CLK_PERIOD / 2) clk = ~clk;
```

```
end

// Task to check light state and duration
task check_state(input [1:0] expected_main, input [1:0] expected_side, input integer duration);
    integer i;
    for (i = 0; i < duration; i = i + 1) begin
        @(posedge clk);
        if (main_street_light !== expected_main || side_street_light !== expected_side) begin
            $display("Error  at time %t: Expected MS=%b, SS=%b, Got MS=%b, SS=%b",
                    $time, expected_main, expected_side, main_street_light, side_street_light);
        end
    end
endtask

// Test procedure
initial begin
    // Apply reset
    $display("Starting simulation...");
    reset = 1;
    @(posedge clk);
    reset = 0;

    // Test MAIN_GREEN state
    $display("Testing MAIN_GREEN state (Main Street GREEN, Side Street RED)...");
    check_state(2'b01, 2'b00, uut.MAIN_GREEN_DURATION);

    // Test MAIN_YELLOW state
    $display("Testing MAIN_YELLOW state (Main Street YELLOW, Side Street RED)...");
    check_state(2'b10, 2'b00, uut.MAIN_YELLOW_DURATION);
```

```
        // Test SIDE_GREEN state
        $display("Testing SIDE_GREEN state (Main Street RED, Side Street GREEN)...");
        check_state(2'b00, 2'b01, uut.SIDE_GREEN_DURATION);

        // Test SIDE_YELLOW state
        $display("Testing SIDE_YELLOW state (Main Street RED, Side Street YELLOW)...");
        check_state(2'b00, 2'b10, uut.SIDE_YELLOW_DURATION);

        // Test reset functionality
        $display("Testing reset...");
        reset = 1;
        @(posedge clk);
        reset = 0;

        // Verify reset to MAIN_GREEN state
        check_state(2'b01, 2'b00, uut.MAIN_GREEN_DURATION);

        $display("Simulation complete.");
        $stop;
    end

endmodule
```