**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Digital System Design**

**Lab#6**

**SUBMITTED TO:**

Dr. Muhammad Yasin

**SUBMITTED BY:**

AMINA QADEER

AMNA AHMED

FARHEEN FATIMAH

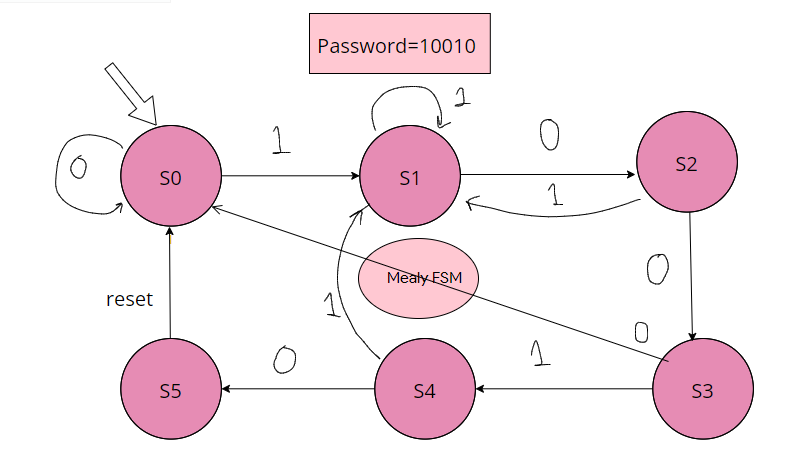
**DE-42 (C&SE)-A**

Electronic Combinational Lock

A diagram of a lock

Description automatically generated

Task 1:



Task 2:

Button Module:

module button(

input clk, in,

output out

);

reg r1, r2, r3;

initial begin

r1 <= 0;

r2 <= 0;

r3 <= 0;

end

always @ (posedge clk) begin

r1 <= in;

r2 <= r1;

r3 <= r2;

end

assign out = ~r3 & r2;

endmodule

Lock Module:

module lock(

input clk, rst, btn0, btn1,

output reg unlock,

output reg [7:0] segdisp

);

wire b0, b1;

reg [3:0] state;

button but0(

.clk(clk),

.in(btn0),

.out(b0)

);

button but1(

.clk(clk),

.in(btn1),

.out(b1)

);

parameter S0 = 0,

S1 = 1,

S2 = 2,

S3 = 3,

S4 = 4,

S5 = 5;

initial begin

state <= S0;

unlock <= 0;

segdisp <= 7'b1111111;

end

always @ (posedge clk or posedge rst) begin

if (rst) begin

state <= S0;

end

else begin

if (b0) begin

case (state)

S0: state <= S0;

S1: state <= S2;

S2: state <= S3;

S3: state <= S0;

S4: state <= S5;

endcase

end

if (b1) begin

case (state)

S0: state <= S1;

S1: state <= S1;

S2: state <= S1;

S3: state <= S4;

S4: state <= S1;

S5: state <= S5;

endcase

end

end

end

always @ (\*) begin

if (state == S5) begin

unlock <= 1;

end

else begin

unlock <= 0;

end

case (state)

4'h0: segdisp <= 7'b1000000; // 0

4'h1: segdisp <= 7'b1111001; // 1

4'h2: segdisp <= 7'b0100100; // 2

4'h3: segdisp <= 7'b0110000; // 3

4'h4: segdisp <= 7'b0011001; // 4

4'h5: segdisp <= 7'b0010010; // 5

4'h6: segdisp <= 7'b0000010; // 6

4'h7: segdisp <= 7'b1111000; // 7

4'h8: segdisp <= 7'b0000000; // 8

4'h9: segdisp <= 7'b0010000; // 9

default: segdisp <= 7'b1111111; // Off

endcase

end

endmodule

Constraints File:

Net "clk" LOC=V10 | IOSTANDARD=LVCMOS33;

Net "clk" TNM\_NET = sys\_clk\_pin;

TIMESPEC TS\_sys\_clk\_pin = PERIOD sys\_clk\_pin 100000 kHz;

Net "unlock" LOC = U16 | IOSTANDARD = LVCMOS33; #Bank = 2, pin name = IO\_L2P\_CMPCLK, Sch name = LD0

Net "rst" LOC = B8 | IOSTANDARD = LVCMOS33; #Bank = 0, pin name = IO\_L33P, Sch name = BTNS

Net "btn0" LOC = A8 | IOSTANDARD = LVCMOS33; #Bank = 0, pin name = IO\_L33N, Sch name = BTNU

Net "btn1" LOC = C4 | IOSTANDARD = LVCMOS33; #Bank = 0, pin name = IO\_L33N, Sch name = BTNU

Net "segdisp<0>" LOC = T17 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L51P\_M1DQ12, Sch name = CA

Net "segdisp<1>" LOC = T18 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L51N\_M1DQ13, Sch name = CB

Net "segdisp<2>" LOC = U17 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L52P\_M1DQ14, Sch name = CC

Net "segdisp<3>" LOC = U18 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L52N\_M1DQ15, Sch name = CD

Net "segdisp<4>" LOC = M14 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L53P, Sch name = CE

Net "segdisp<5>" LOC = N14 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L53N\_VREF, Sch name = CF

Net "segdisp<6>" LOC = L14 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L61P, Sch name = CG

Net "segdisp<7>" LOC = M13 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO\_L61N, Sch name = DP

Test Bench:

module lockTB;

// Inputs

reg clk;

reg rst;

reg btn0;

reg btn1;

// Outputs

wire unlock;

wire [7:0] segdisp;

// Instantiate the Unit Under Test (UUT)

lock uut (

.clk(clk),

.rst(rst),

.btn0(btn0),

.btn1(btn1),

.unlock(unlock),

.segdisp(segdisp)

);

always #1 clk = ~clk;

initial begin

// Initialize Inputs

clk = 0;

rst = 0;

btn0 = 0;

btn1 = 0;

// Wait 100 ns for global reset to finish

#100;

// correct combination

#10 btn1 = 1;

#10 btn1 = 0;

#10 btn0 = 1;

#10 btn0 = 0;

#10 btn0 = 1;

#10 btn0 = 0;

#10 btn1 = 1;

#10 btn1 = 0;

#10 btn0 = 1;

#10 btn0 = 0;

#50 rst = 1;

// incorrect combination

#10 btn0 = 1;

#10 btn0 = 0;

#10 btn1 = 1;

#10 btn1 = 0;

#10 btn0 = 1;

#10 btn0 = 0;

#10 btn1 = 1;

#10 btn1 = 0;

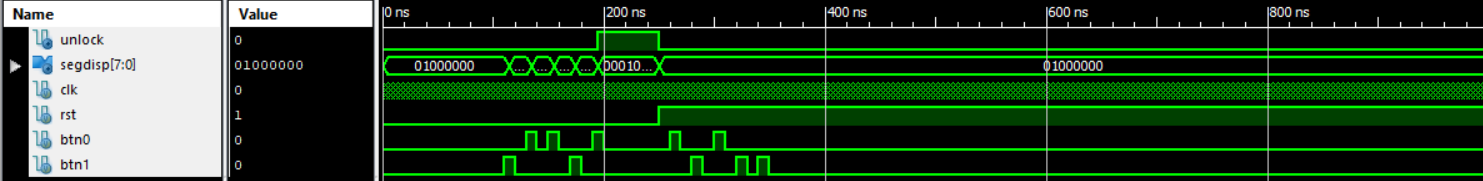
#10 btn1 = 1;

#10 btn1 = 0;

end

endmodule

Waveform:



Answers:

1. We have implemented Mealy FSM.
2. The state register requires 3 flip flops.
3. Two always blocks are used.