Exercise 7: State Machine

Revision: October 10, 2013

DIGILENT
www.digilentinc.com
1300 Henley Court | Pullman, WA 99163
(509) 334 6306 Voice and Fax

STUDENT

I am submitting my own work in this exercise, and I am aware of the penalties for cheating that will be assessed if I submit work for credit that is not my own.

KEY.
Print Name _____ WSU ID _____

Sign Name _____ Date _____

Estimated hours of work

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

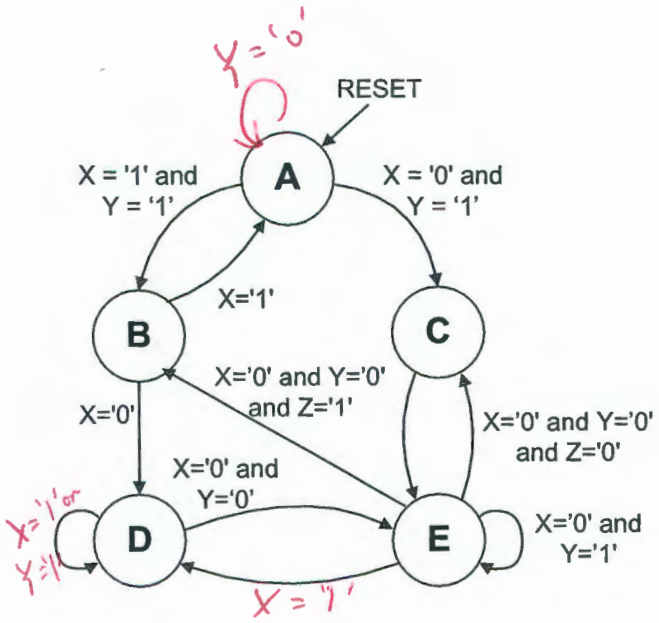
Weight

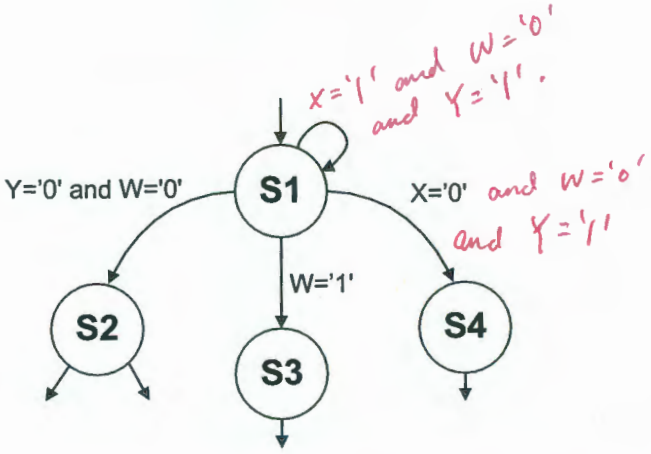
GRADER

#	Points	Score
1	14	
2	15	
3	15	
4	25	
5	25	
6	16	
T	110	

Total Score

Problem 1. (10 points) Modify the state diagram branching conditions in the diagrams below as needed to ensure the sum and exclusion rules are obeyed in each case. You can add a holding conditions or change branch codes as desired.





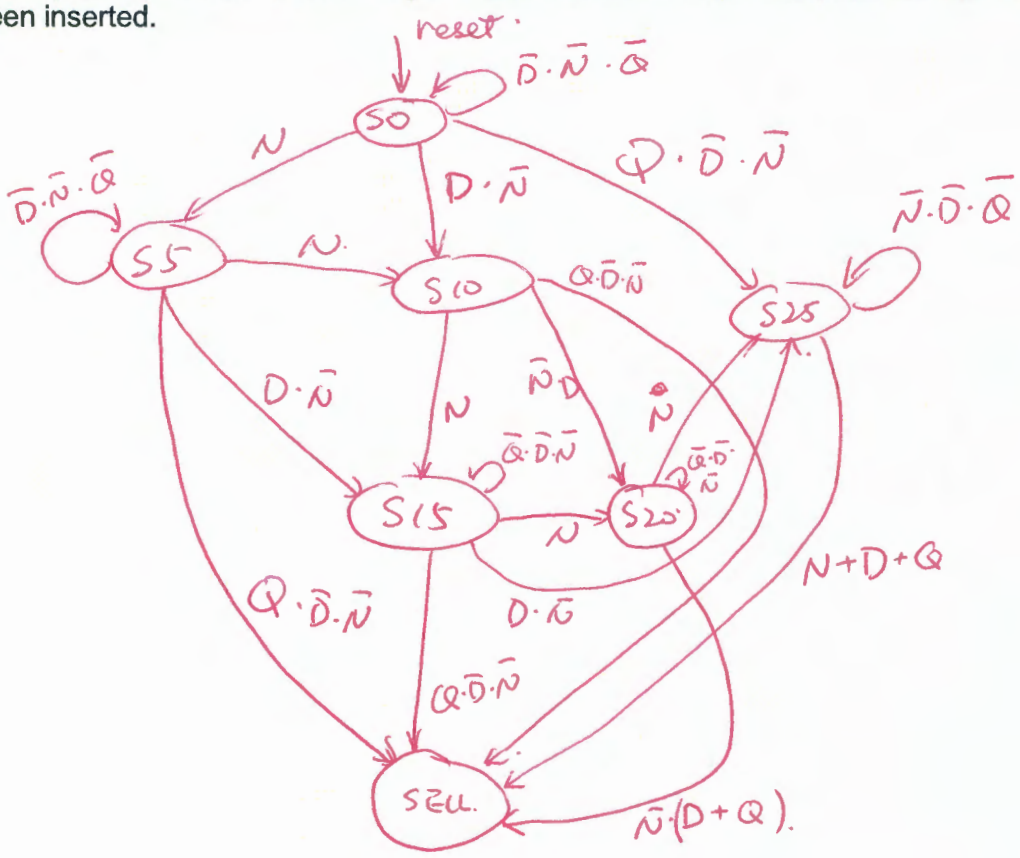
Modify the S4 branch and holding condition only

Problem 2. (15 points) A vending machine should SELL an item if 30 cents is input. The machine has a coin sensor that can detect nickels, dimes, and quarters, and reject everything else. No change is given (i.e, if two quarters are input, simply assert SELL and keep the fifty cents). Sketch a state diagram to assert SELL when adequate coinage has been inserted.

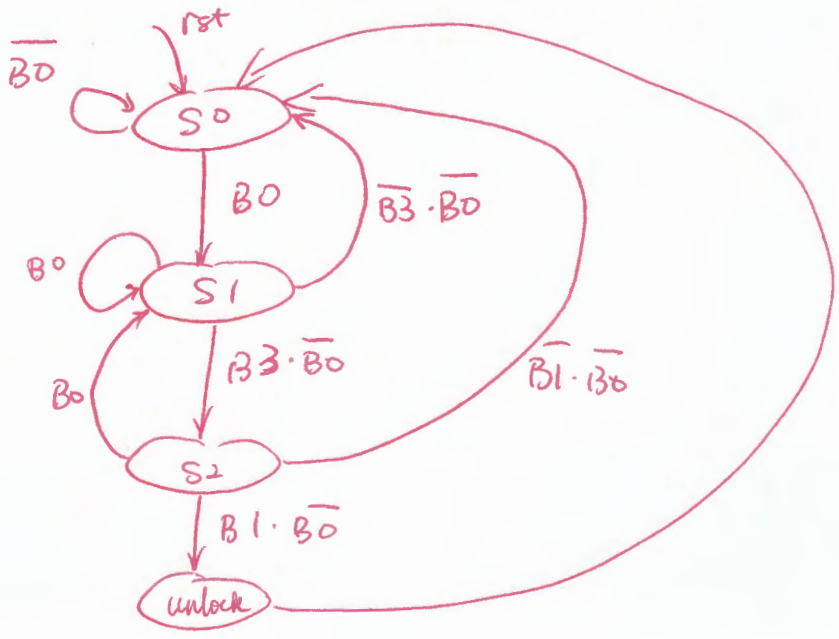
N - nickel.

D - dime.

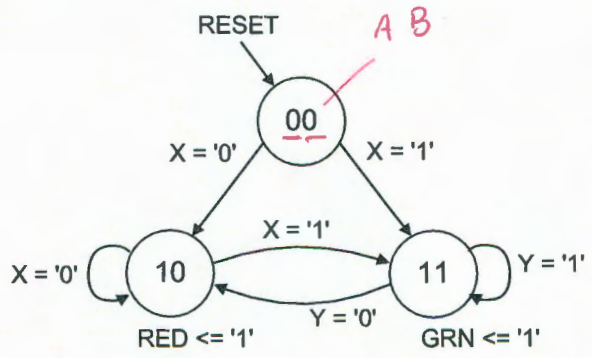
Q - quarter

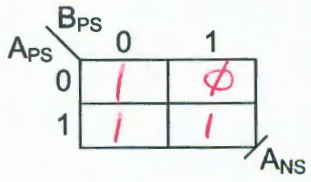


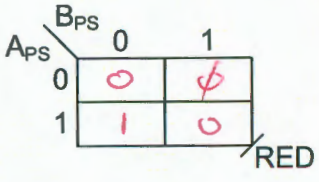
Problem 3. (15 points) Create a state diagram for a machine that can control a 4-button digital combination lock mechanism, unlocking only if the sequence B0-B3-B1 is detected.

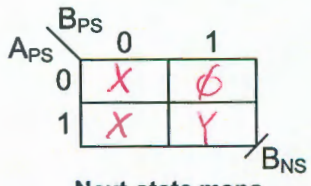


Problem 4. (25 points) Sketch a circuit for the state machines below.

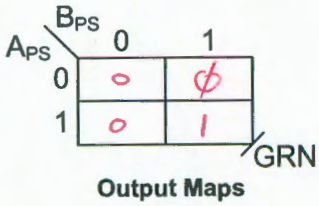








Next-state maps



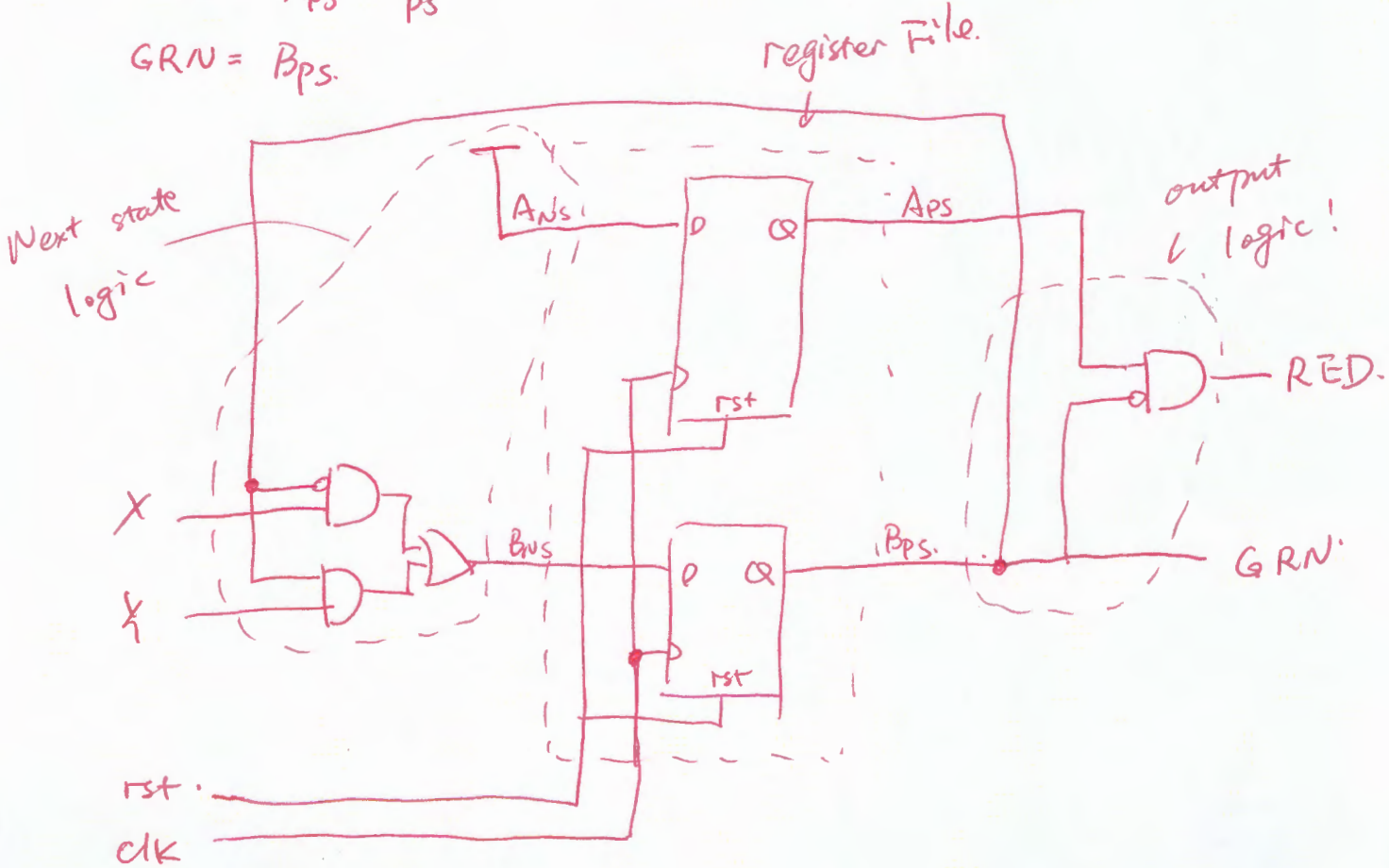
Output Maps

$\Rightarrow A_{NS} = 1.$

$B_{NS} = \overline{B_{PS}} \cdot X + B_{PS} \cdot Y.$

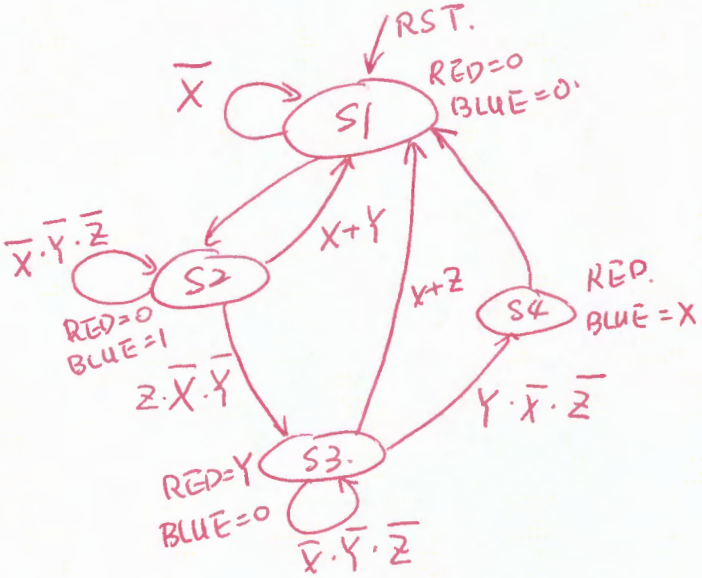
$RED = A_{PS} \cdot \overline{B_{PS}}$

$GRN = B_{PS}.$



Problem 5. (25 points) Sketch a state diagram based on the following Verilog Code

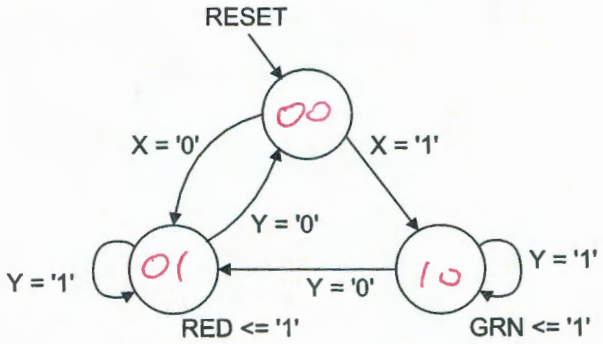
```
module fsm (
    CLK, RST, X, Y, Z, RED, BLUE);

input CLK, RST, X, Y, Z;
output reg RED, BLUE;

localparam S1 = 2'd0;
localparam S2 = 2'd1;
localparam S3 = 2'd2;
localparam S4 = 2'd3;

reg [1:0] ps, ns;

always @ (ps, x, y, z)
begin
    case (ps)
        S1: begin
            RED = 1'b0;
            BLUE = 1'b0;
            if (X == 1'b0) ns = S1;
            else ns = S2;
        end
        S2: begin
            RED = 1'b0;
            BLUE = 1'b1;
            if (X == 1'b0 && Y == 1'b0 && Z == 1'b0) ns = S2;
            else if (X == 1'b1 || Y == 1'b1) ns = S1;
            else if (Z == 1'b1 && X == 1'b0 && Y == 1'b0) ns = S3;
        end
        S3: begin
            RED = Y;
            BLUE = 1'b0;
            if (Y == 1'b1 && X == 1'b0 && Z == 1'b0) ns = S4;
            else if (X == 1'b0 && Y == 1'b0 && Z == 1'b0) ns = S3;
            else if (X == 1'b1 || Z == 1'b1) ns = S1;
        end
        S4: begin
            RED = 1'b1;
            BLUE = X;
            ns = S1;
        end
        default: begin
            RED = 1'b0;
            BLUE = 1'b0;
            ns = S1;
        end
    endcase
end

always @ (CLK, RST)
begin
    if (RST == 1'b1) ps <= S1;
    else ps <= ns;
end

endmodule
```



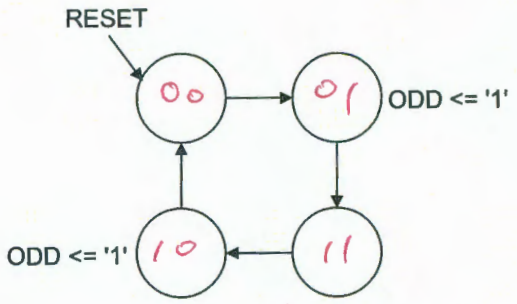
rst to state S1.

high active.

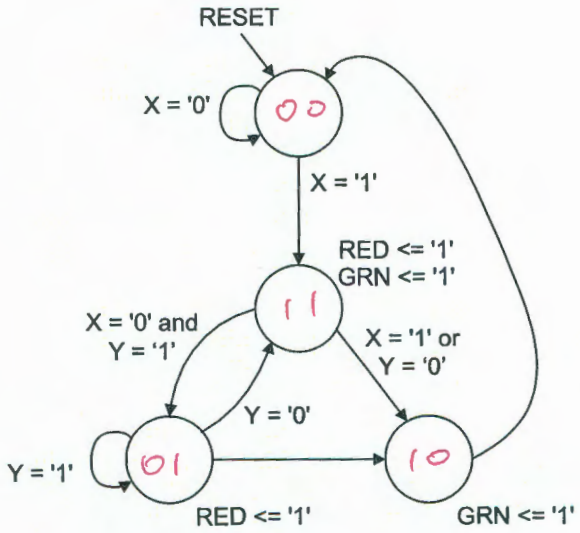
Problem 6. (16 points) Assign state codes to the state diagrams below, using unit-distance coding and/or matching state codes to outputs



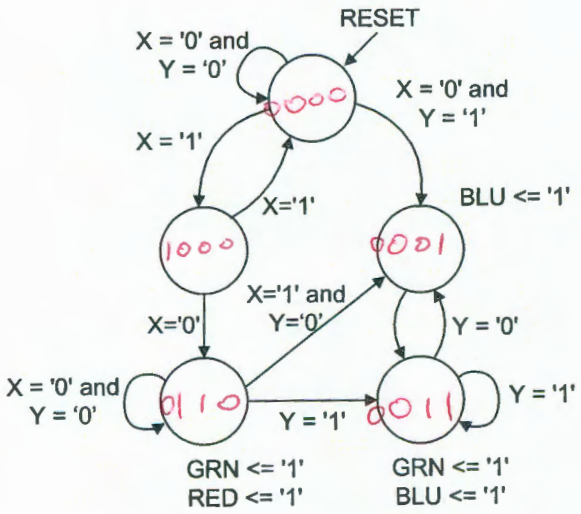
matching state codes to outputs



unit-distance and matching state codes to outputs.



matching state codes to outputs



matching state codes to outputs.