No: Date:

NAME : SHAH RAZA
REG NO : 18PWCSE1658
SECTION : B
SUBJECT : DSD
ASSIGNMENT NO : 3

Signature : Raza

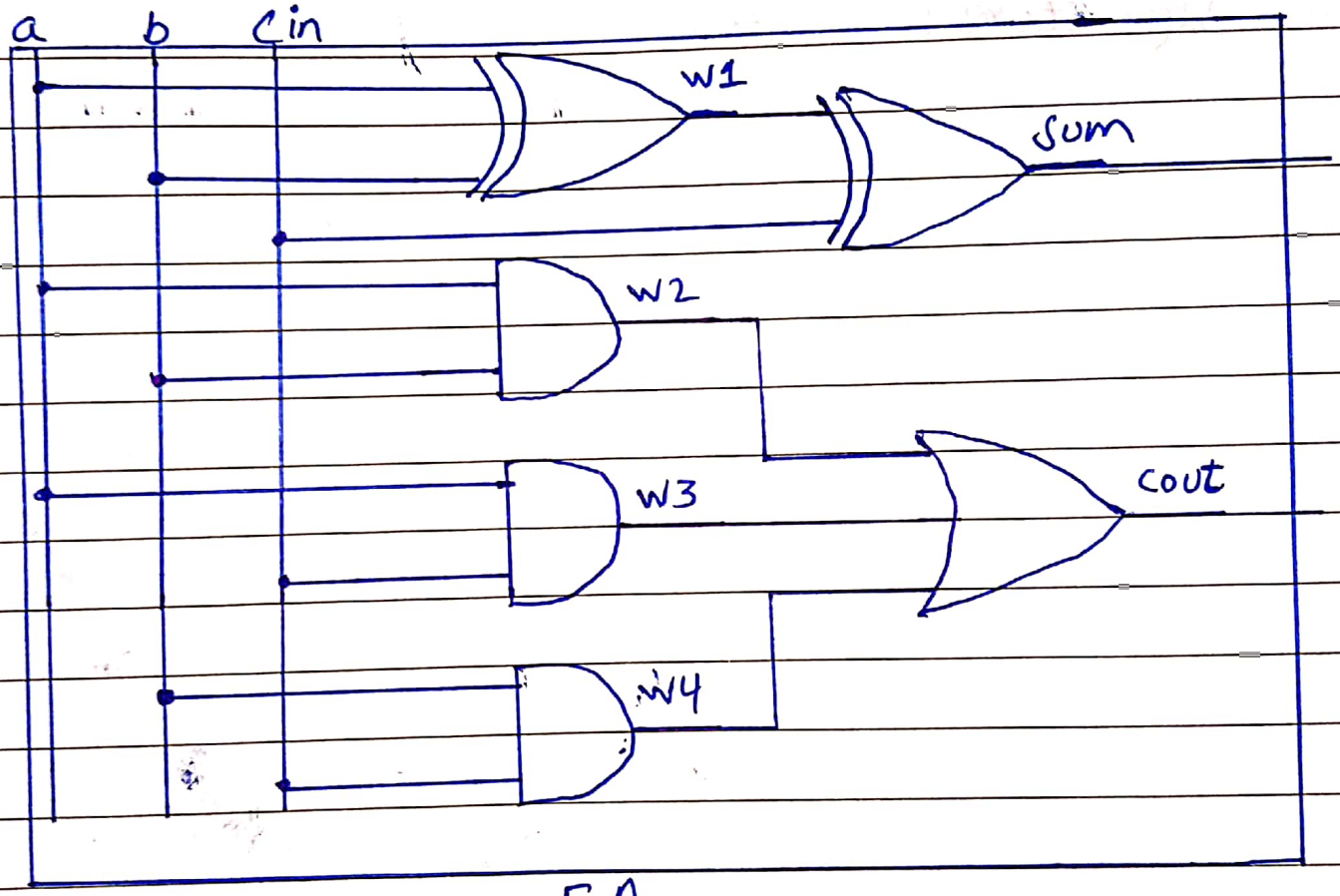
No: Date:

PROBLEM 1:

Draw the synthesized circuits described by the verilog codes.

a)

Solution:

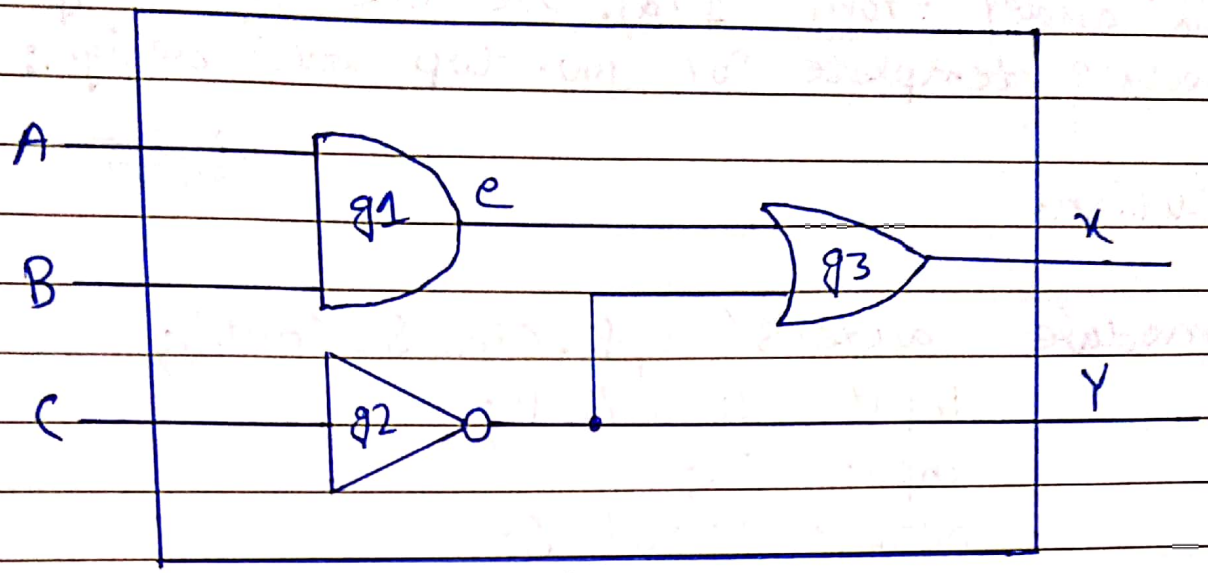


FA.

No: Date:

b)

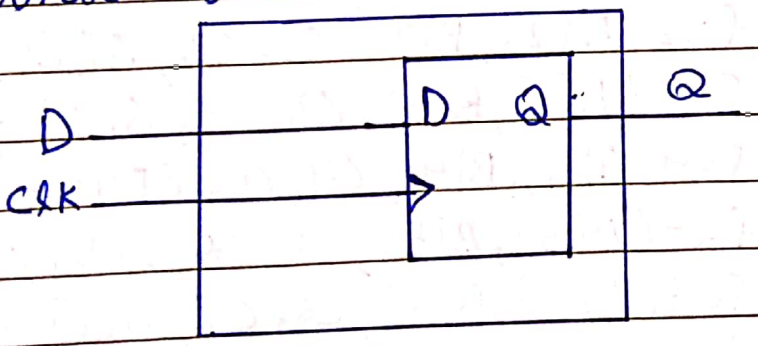
Solution:



smpl_circuit

c)

Solution:



D_latch

No: Date:

PROBLEM 2:

Implement an 8-bit ripple carry adder, using as building block the 1-bit full adder from 1(a). Use the following module templete for your top level design:

Solution:

```
module adder8(A, B, Cin, S, Cout);
    input [7:0] A, B;
    input Cin;
    output [7:0] S;
    output Cout;
    wire C1, C2, C3, C4, C5, C6, C7;

    FA fa0(A[0], B[0], Cin, C1, S[0]);
    FA fa1(A[1], B[1], C1, C2, S[1]);
    FA fa2(A[2], B[2], C2, C3, S[2]);
    FA fa3(A[3], B[3], C3, C4, S[3]);
    FA fa4(A[4], B[4], C4, C5, S[4]);
    FA fa5(A[5], B[5], C5, C6, S[5]);
    FA fa6(A[6], B[6], C6, C7, S[6]);
    FA fa7(A[7], B[7], C7, Cout, S[7]);
endmodule
```

No: Date:

PROBLEM 3:

Implement a 4-bit magnitude comparator using dataflow description (i.e using continuous assignment).

Solution:

```
module mag_comp4 (A, B, ALB, AGB, AEB);
    input [3:0] A, B;
    output ALB, AGB, AEB;

    assign ALB = (A<B);
    assign AGB = (A>B);
    assign AEB = (A==B);
endmodule
```

No: Date:

PROBLEM 4:

The following code is intended to implement a 1:2 de-MUX. The input is x and two outputs are y and z, and the switch control is s.

```
wire x, y;
reg z;
always @(x or y or z)
    begin
        if(!s) y=x;
        else z=x;
    end
```

a) Find out all the things wrong in this code.

Ans: The following are the things wrong in this code:

1. y should be declared as reg because it is an output.
2. Since y and z are outputs, they should not be in the sensitivity list of the always block.
3. s should be there in the sensitivity list of the always block.

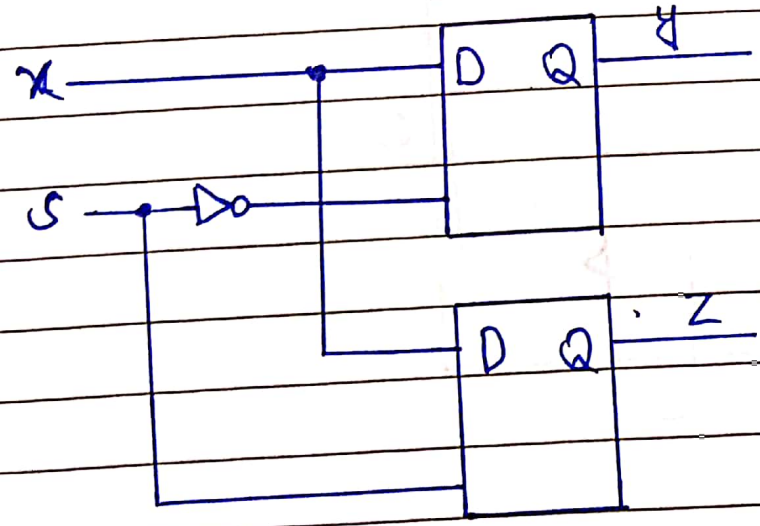
No: Date:

b) Does the code produce any latches? If no, state why. If yes, how many latches can be produced? And by which statement(s)?

Ans: Yes, this code produce 2 latches. One to store the previous value of z, when s=0 and y=x is executed. And the other to store the previous value of y, when s=1 and z=x is executed.

c) After all the problems you find in (a) are fixed, draw the logic diagram of the synthesized hardware from the corrected code.

Solution:



No: Date:

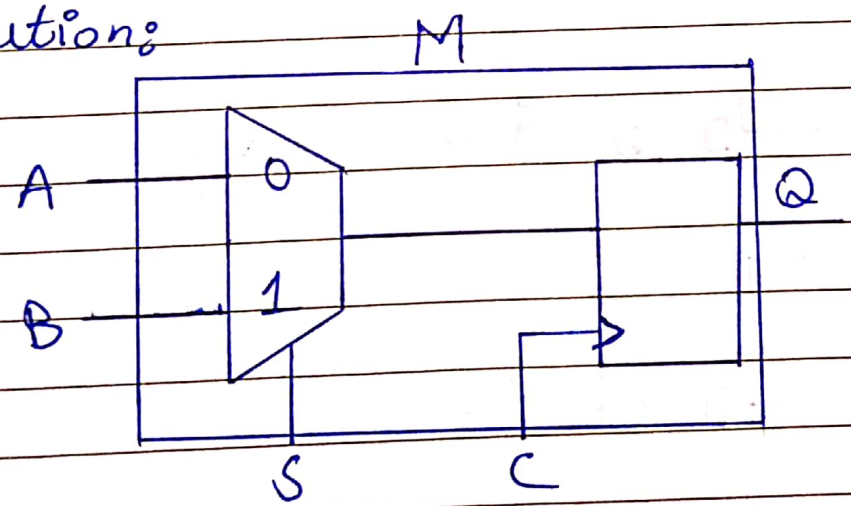
PROBLEM 5:

Below is a short snippet of verilog code of a digital functional block. Draw the logic circuit diagram that is described by the verilog code.

```
module M(Q, S, A, B, C);
    output Q;
    input S, A, B, C;
    reg Q;

    always @(posedge C)
        Q = (S) ? A : B;
endmodule
```

Solution:



No: Date:

PROBLEM 6:

Solution:

```
module Gray_3bit(C, R, OUT);
input C, R;
output [2:0] OUT;
reg [2:0] OUT;

parameter [2:0] A = 3'b000,
B = 3'b001, C = 3'b011, D = 3'b010,
E = 3'b110, F = 3'b111, G = 3'b101,
H = 3'100;

always @(posedge C)
    if (R)
        OUT = 3'b000;
    else
        case(OUT)
            A: OUT = B;
            B: OUT = C;
            C: OUT = D;
            D: OUT = E;
            E: OUT = F;
            F: OUT = G;
            G: OUT = H;
            H: OUT = A;
            default: OUT = A;
        endcase

endmodule
```

No: Date:

PROBLEM 7:

Raza

Solution:

```
module Mux8to1( S,I0,I1,I2,I3,I4,I5,I6,I1,O);
    inpat [2:0] S;
    input [7:0] I0,I1,I2,I3,I4,I5,I6,I7;
    output [7:0] O;
    reg [7:0] O;

    always @(*)
        case(S)
            3'b000: O = I0;
            3'b001: O = I1;
            3'b010: O = I2;
            3'b011: O = I3;
            3'b100: O = I4;
            3'b101: O = I5;
            3'b110: O = I6;
            3'b111: O = I7;
        endcase
endmodule
```

No: Date:

PROBLEM 8:

Solution:

```
module Design(C, R, D0, D1, D2, D3, D4, D5, D6, D7, OUT);
    input C, R;
    input [7:0] D0, D1, D2, D3, D4, D5, D6, D7;
    output [7:0] OUT;

    wire [3:0] GOUT;

    Gray_3bit gc1(C, R, GOUT);
    Mux8to1 m1(GOUT, D0, D1, D2, D3, D4, D5,
               D6, D7, OUT);

endmodule
```