Q1)

BCD stands for Binary Coded Decimal. In the BCD numbering system a decimal number is separated into four bits for each decimal digit within the number. Each decimal digit is represented by its weighted binary value performing a direct translation of the number. So a 4-bit group represents each displayed decimal digits from 0000 for a zero to 1001 for a nine. The inputs I0 to I9 represents decimal digits 0-9 respectively and the BCD corresponding to decimal digits are represented by bits Out[3], out[2] Out[1], Out[0].

Truth table :-

I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	Out[3]	Out[2]	Out[1]	Out[0]
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

The following boolean expressions are obtained from the above truth table :-

$Out[0] = I1 + I3 + I5 + I7 + I9$

$Out[1] = I2 + I3 + I6 + I7$

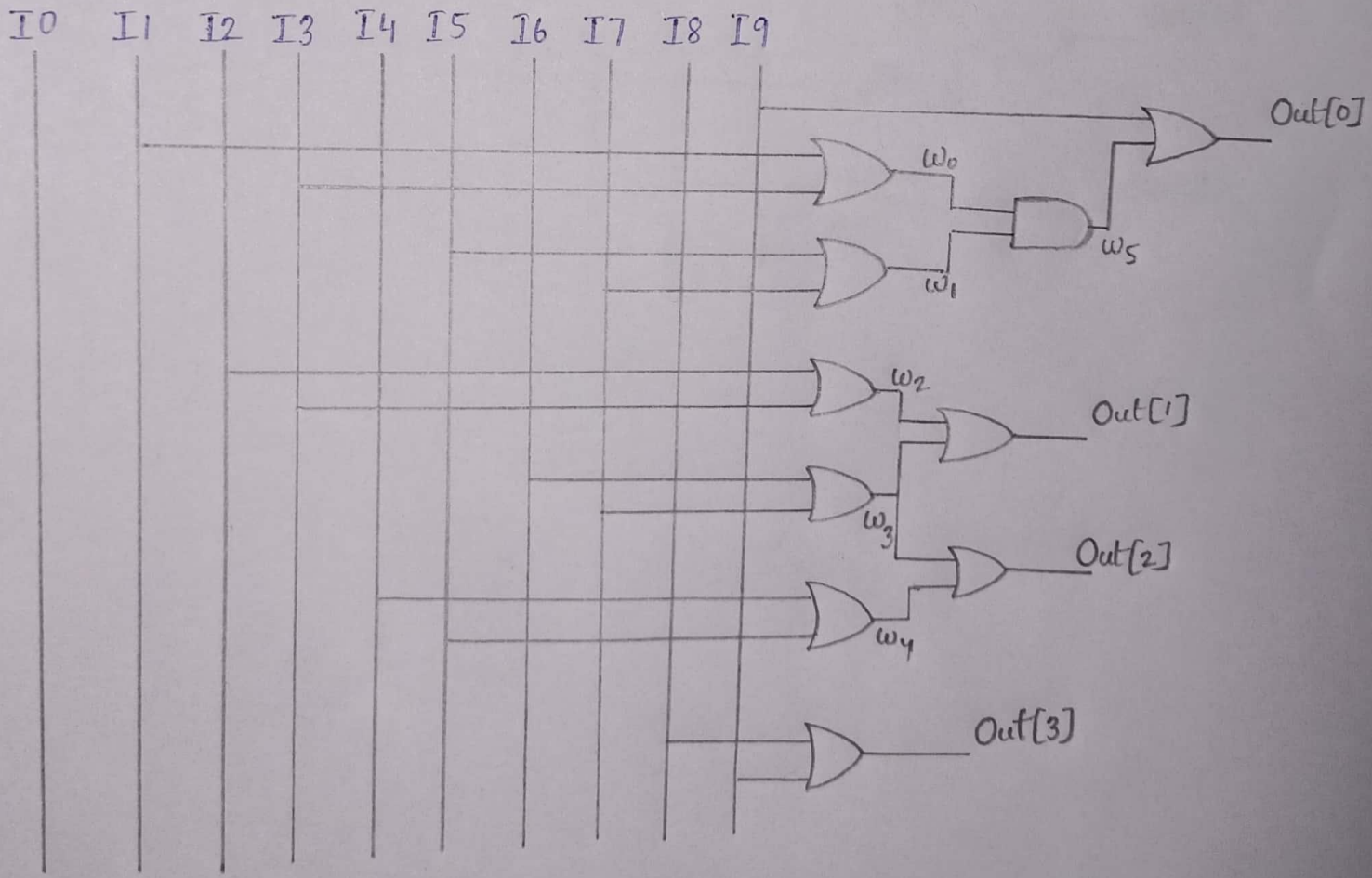
$Out[2] = I4 + I5 + I6 + I7$

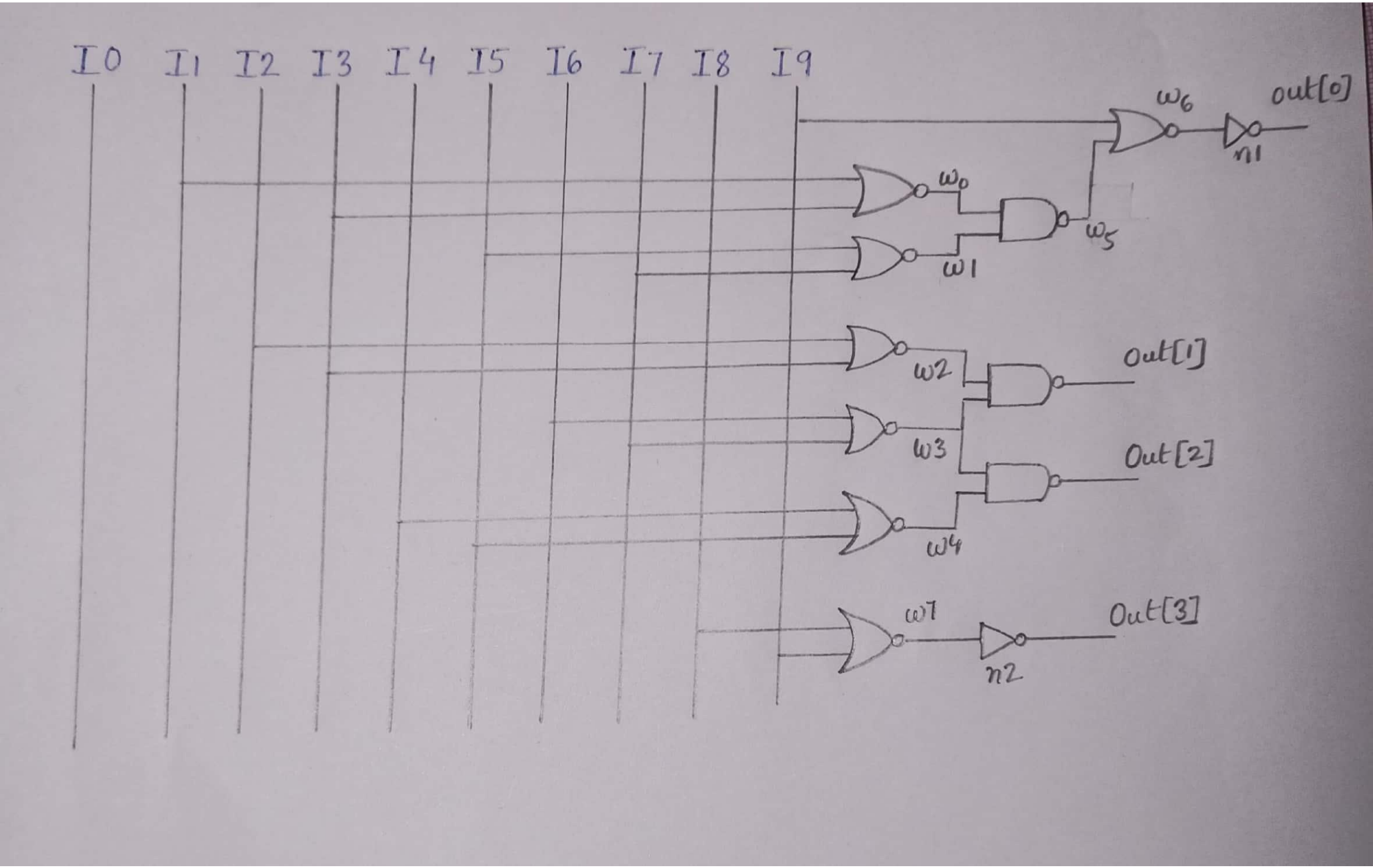
$Out[3] = I8 + I9$

the term $I6 + I7$ is common in $Out[1]$, $Out[2]$ so it is shared between $Out[1]$ and $Out[2]$

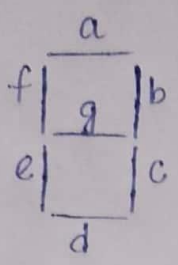
Now, we will implement the function using AND and OR gates. Later we will convert it into implementation using NAND, NOR, NOT gates.

Circuit Diagram:-





Q2)



The above figure shows the seven-segment display. If input to any segment is 1 then the segment glows. If input is 0 then the segment doesn't glows.

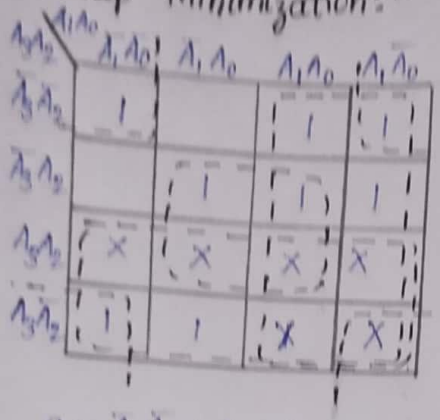
Truth Table :-

A3	A2	A1	A0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Now we will use four variable K-map to minimize the seven outputs separately with respect to four inputs.

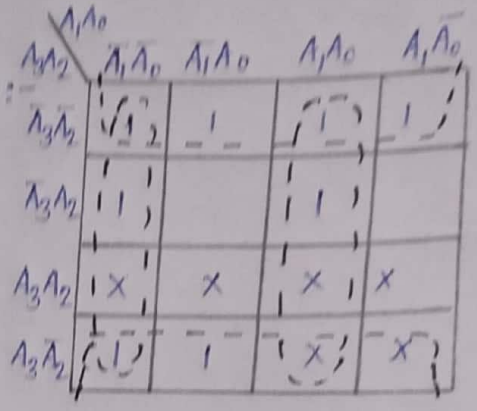
K-Map Minimization:-

a:-



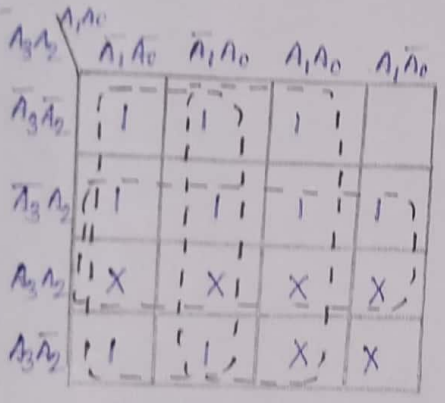
$a = A_2\bar{A}_0 + A_3 + A_1 + A_2A_0$

b:-



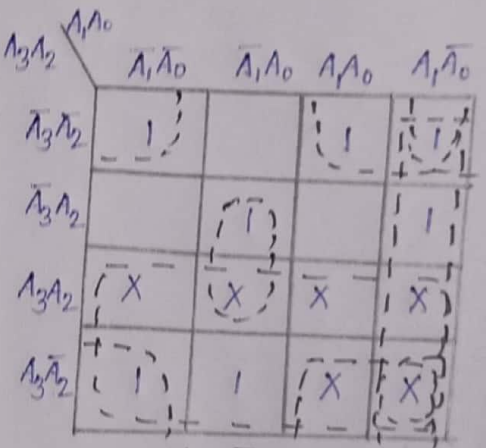
$b = A_2 + \bar{A}_1\bar{A}_0 + A_1A_0$

c:-



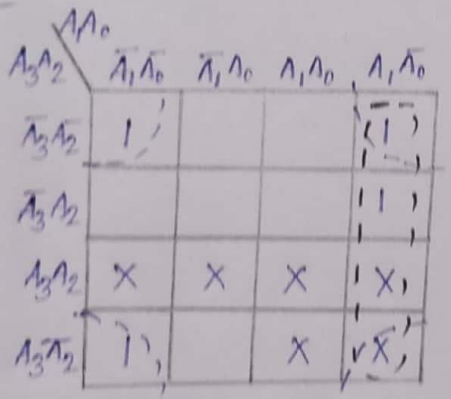
$c = \bar{A}_1 + A_0 + A_2$

d:-



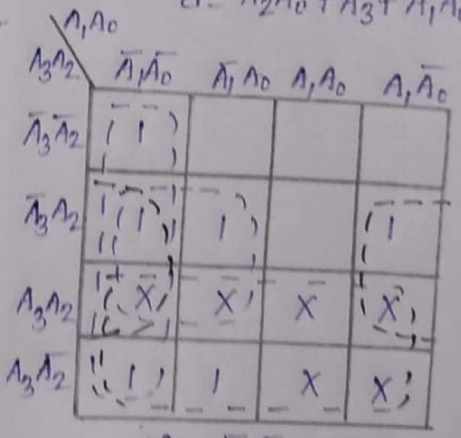
$d = \bar{A}_2\bar{A}_0 + A_3 + A_1\bar{A}_0 + A_1\bar{A}_2 + A_2\bar{A}_1$

e:-



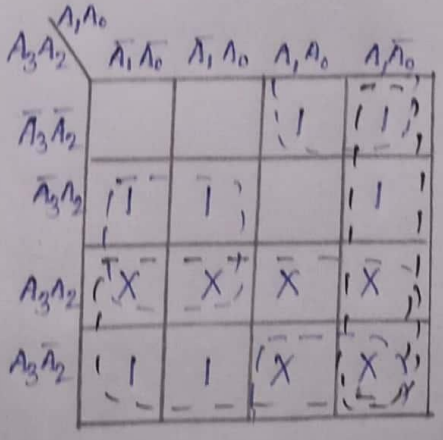
$e = \bar{A}_2\bar{A}_0 + A_1\bar{A}_0$

f:-



$f = \bar{A}_1\bar{A}_0 + A_3 + A_2\bar{A}_0 + A_2\bar{A}_1$

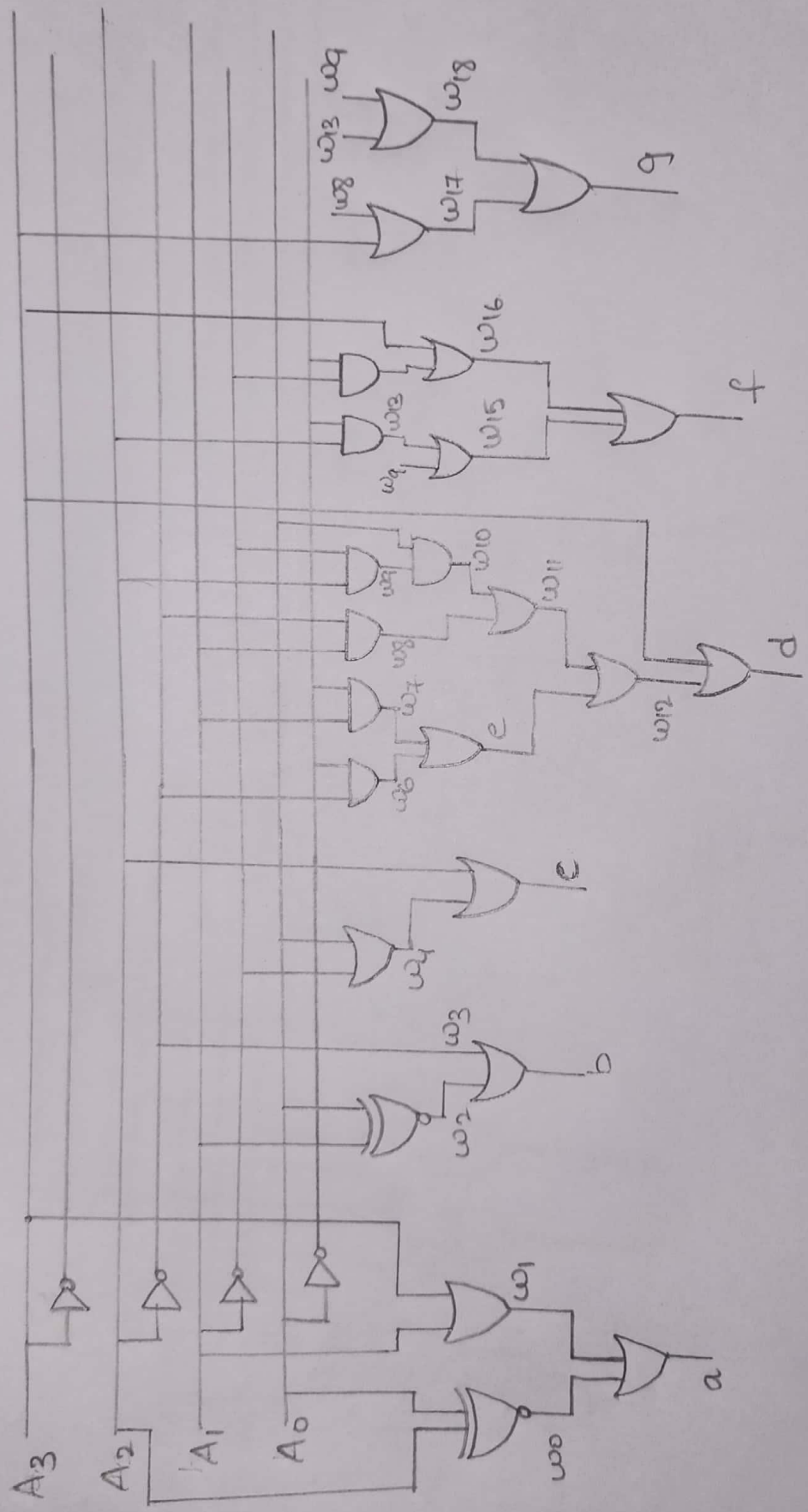
g:-



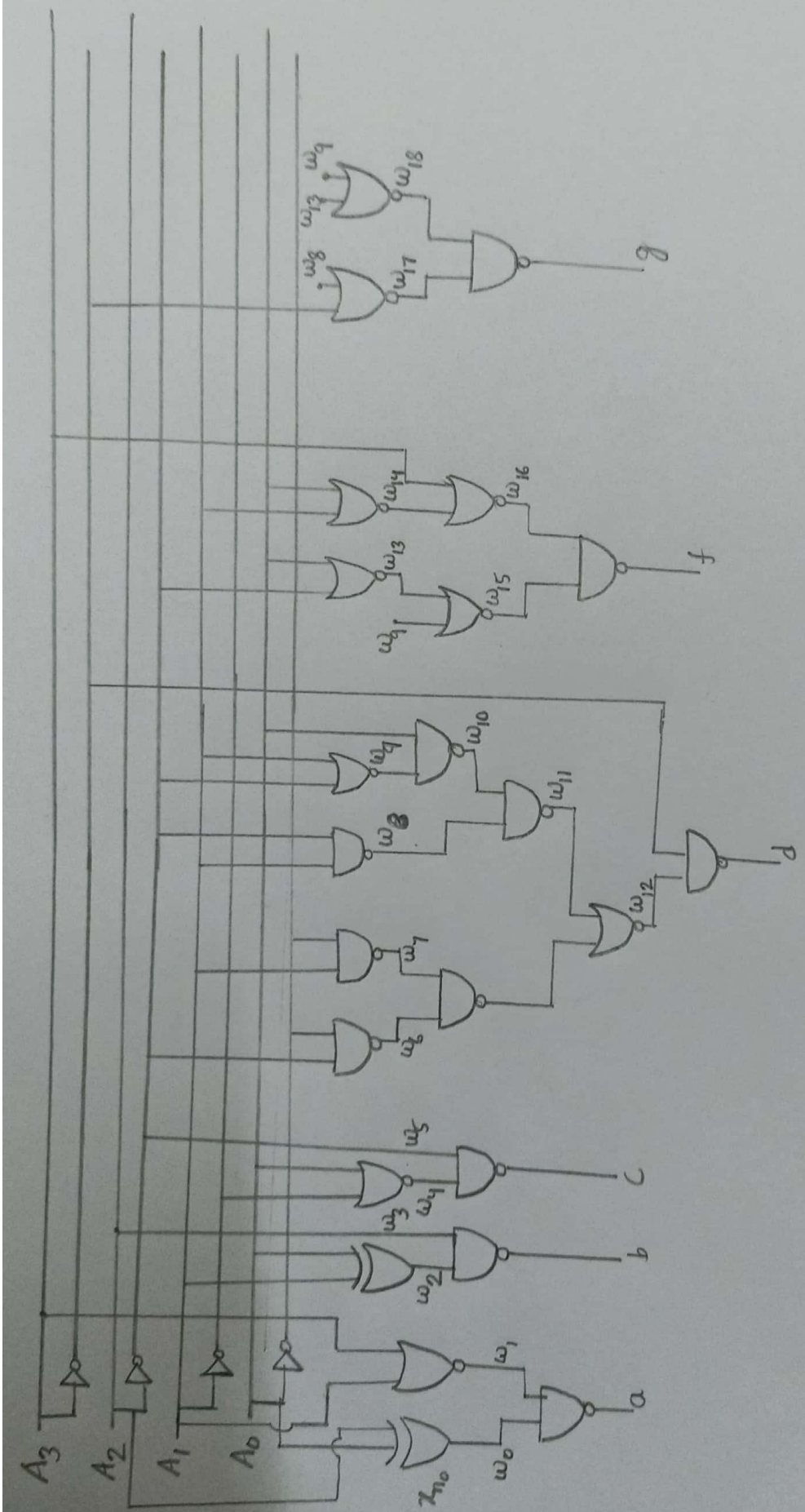
$g = A_3 + A_1\bar{A}_2 + A_2\bar{A}_0 + A_2\bar{A}_1$

Now we will implement using AND, OR gates. Later we will implement using gates (NAND, NOR, NOT, XOR, XNOR)

Circuit Diagram:-



Structural:-



3) Let the BCD representation of x and y be.

$$x = x_3 x_2 x_1 x_0$$
$$y = y_3 y_2 y_1 y_0$$

case(i) $x = y$.

Let f_0 be the function that represent $x = y$.

This case is possible when all bits in BCD representation of x and y are same. i.e.,

$x_3 = y_3$ and $x_2 = y_2$ and $x_1 = y_1$ and $x_0 = y_0$

This can be written as.

$$f_0 = (x_3 \odot y_3)(x_2 \odot y_2)(x_1 \odot y_1)(x_0 \odot y_0)$$
$$= \overline{(x_3 \odot y_3)(x_2 \odot y_2) + (x_1 \odot y_1)(x_0 \odot y_0)}$$

case (ii) $x < y$.

Let f_2 be function that represent $x < y$.

This is possible when.

$x_3 < y_3$ (or) $(x_3 = y_3)$ and $x_2 < y_2$ (or) $x_3 = y_3$ and $x_2 = y_2$ and $x_1 < y_1$ (or) $x_3 = y_3$ and $x_2 = y_2$ and $x_1 = y_1$ and $x_0 < y_0$

$$f_2 = \bar{x}_3 y_3 + (x_3 \odot y_3)\bar{x}_2 y_2 + (x_3 \odot y_3)(x_2 \odot y_2)\bar{x}_1 \bar{y}_1 + (x_3 \odot y_3)(x_2 \odot y_2)(x_1 \odot y_1)\bar{x}_0 y_0$$

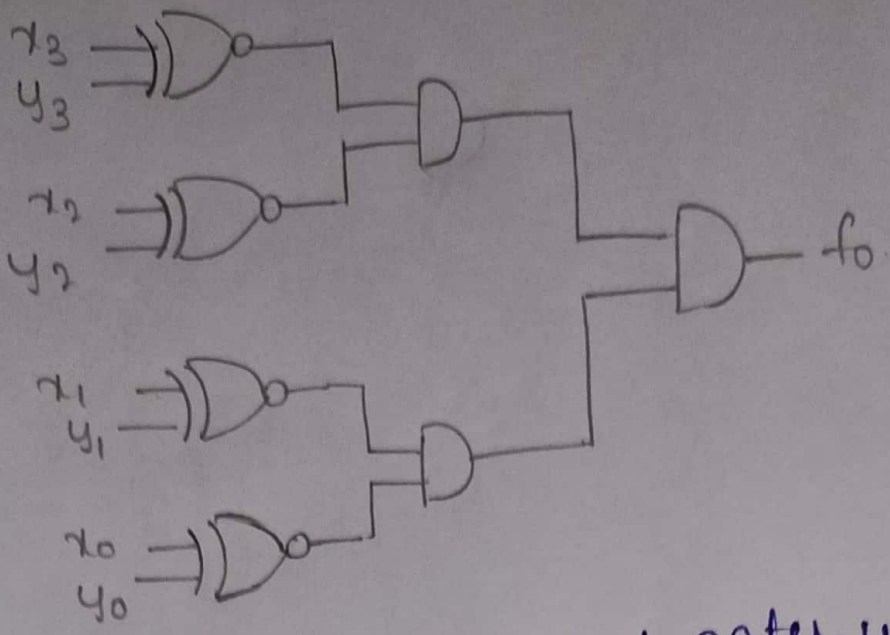
case (iii) $x > y$.

Let f_1 be function that represent $x > y$.

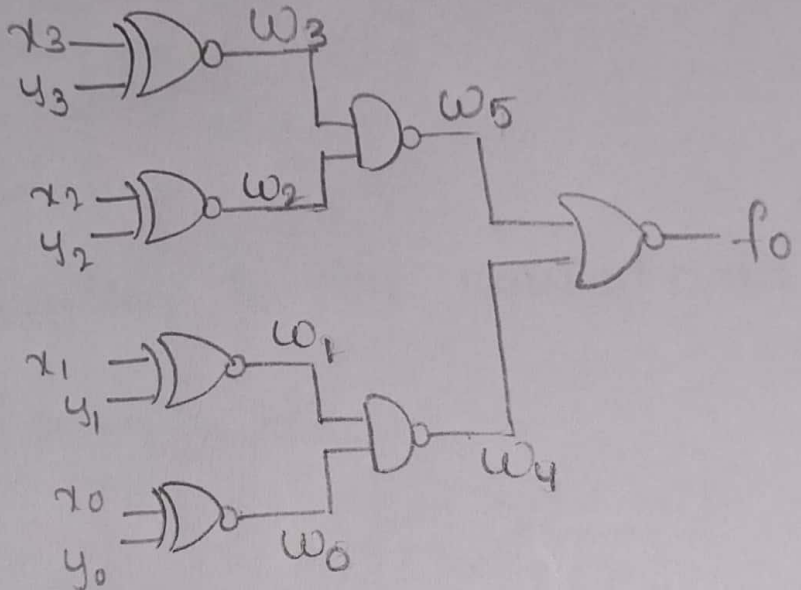
$$f_1 = x_3 \bar{y}_3 + (x_3 \odot y_3) x_2 \bar{y}_2 + (x_3 \odot y_3)(x_2 \odot y_2) x_1 \bar{y}_1 + (x_3 \odot y_3)(x_2 \odot y_2)(x_1 \odot y_1) x_0 \bar{y}_0$$

Circuit Diagram

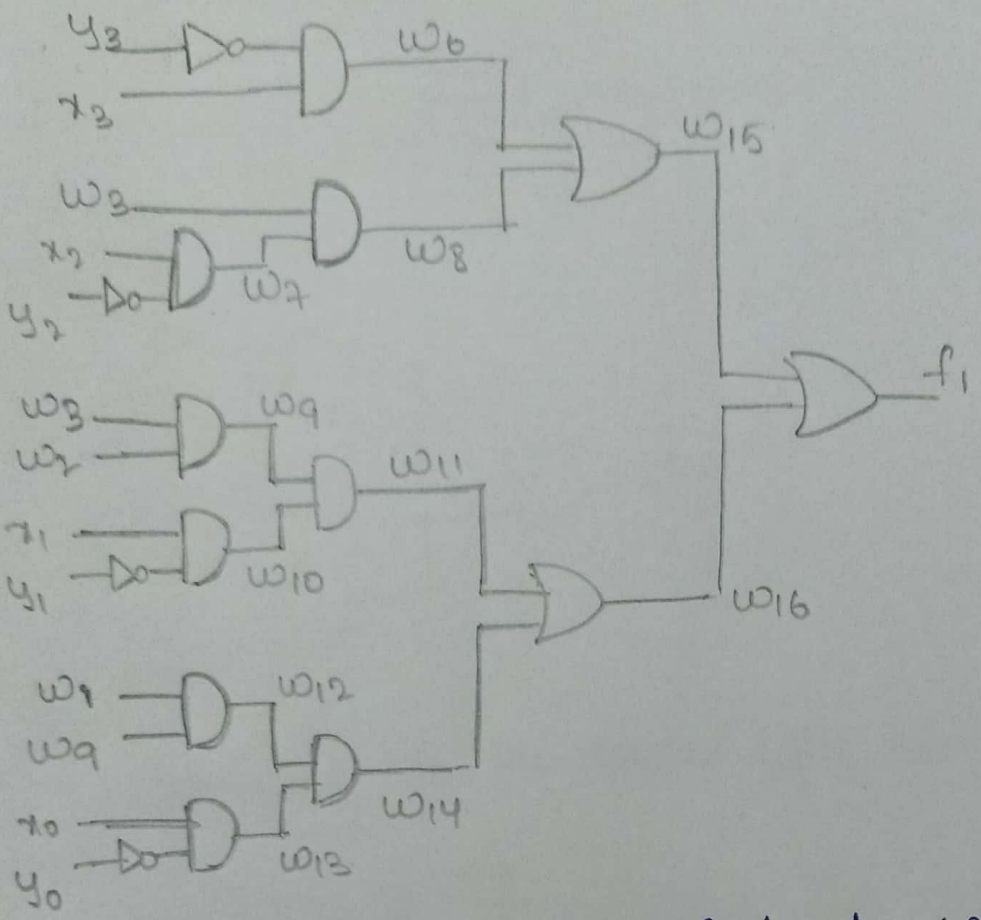
f_0



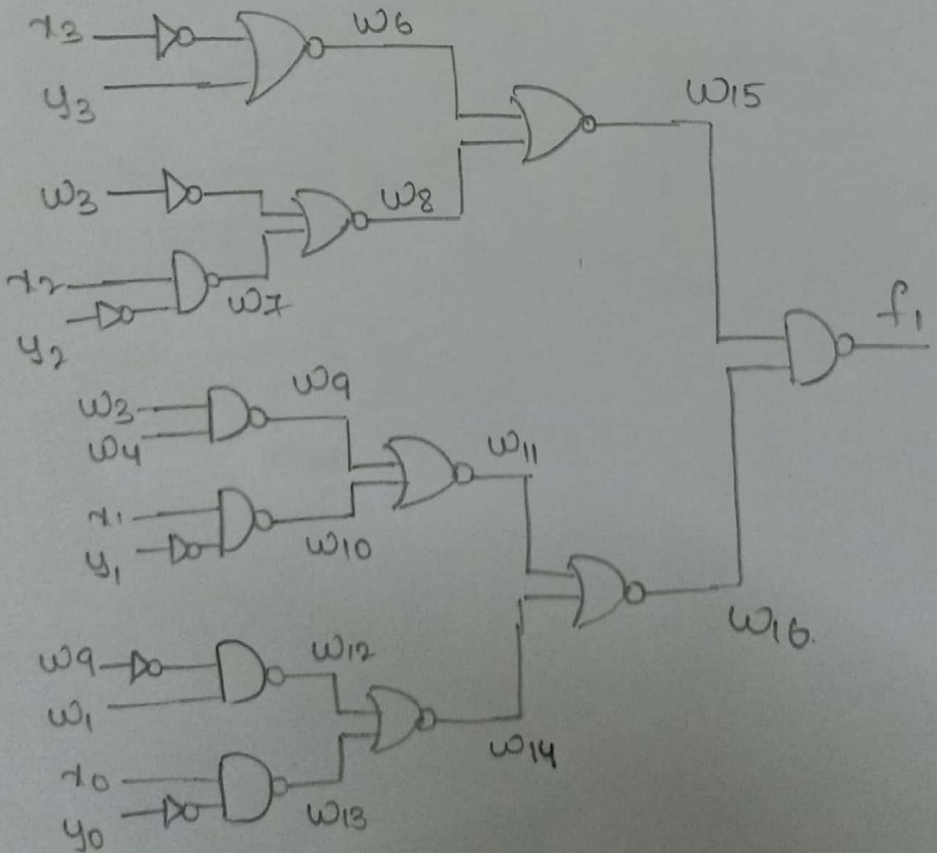
converting to the required gates, we get



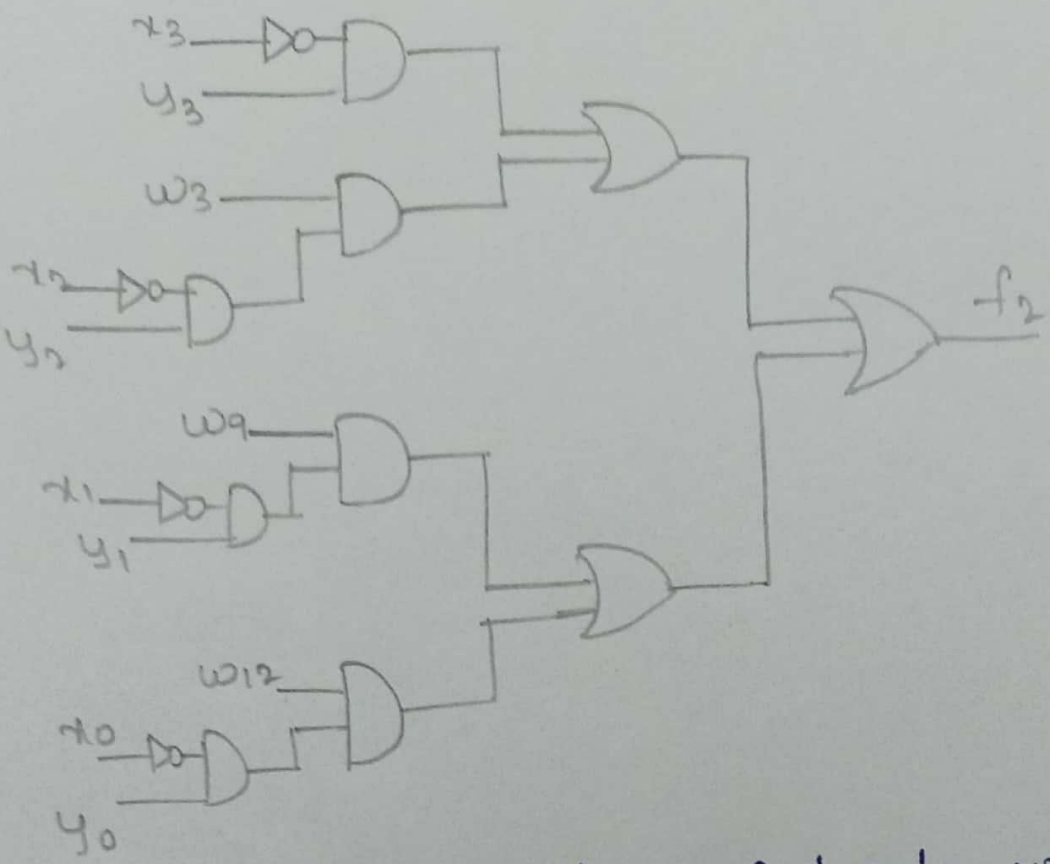
f_1



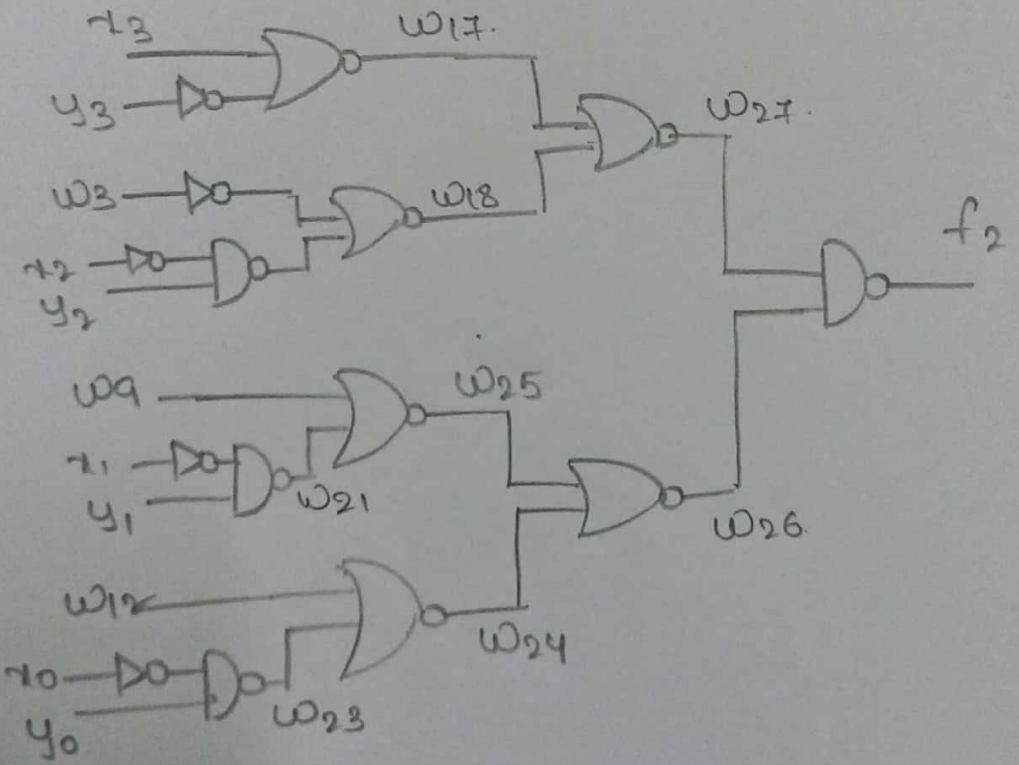
converting to the required gates, we get.

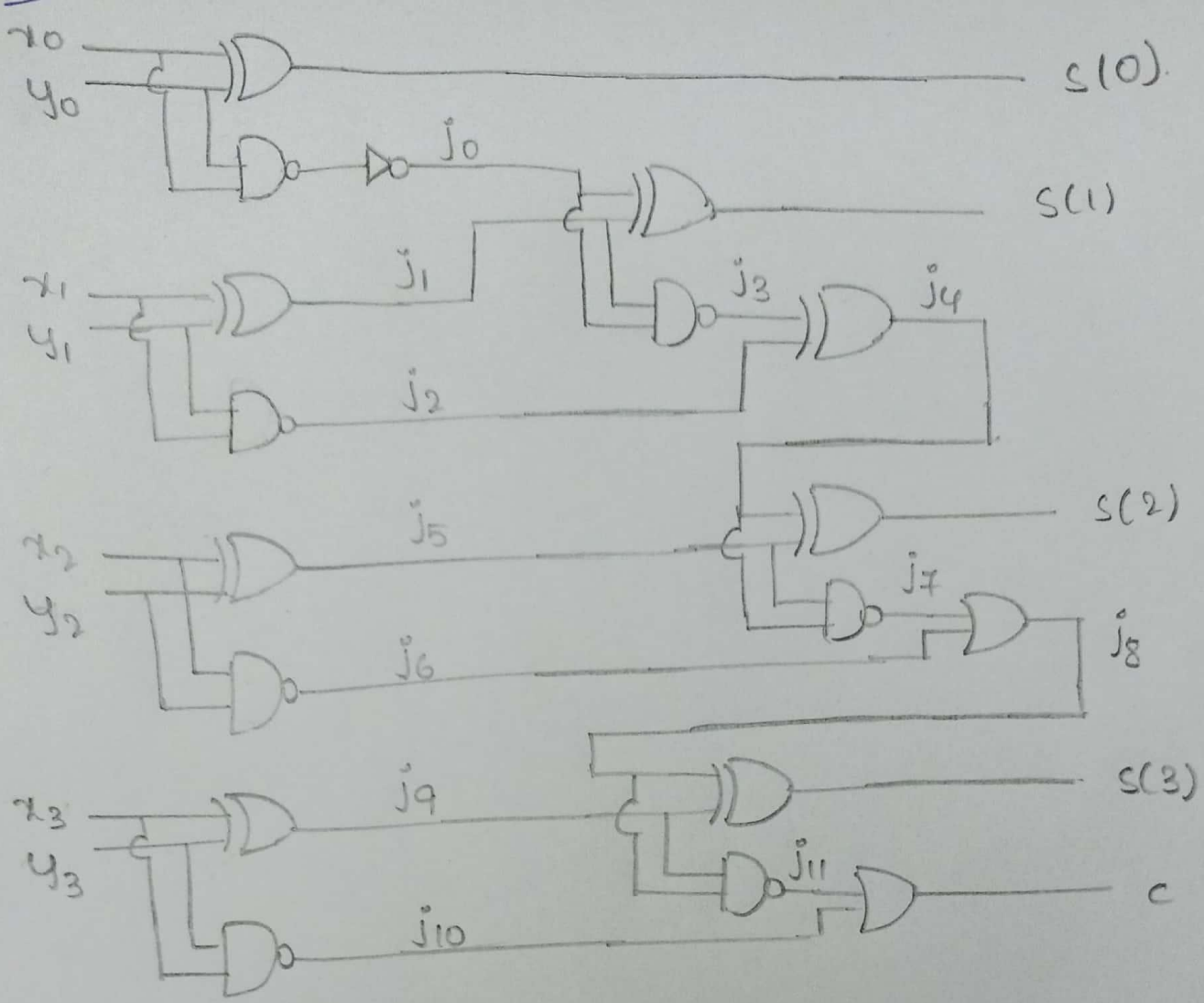


f_2.

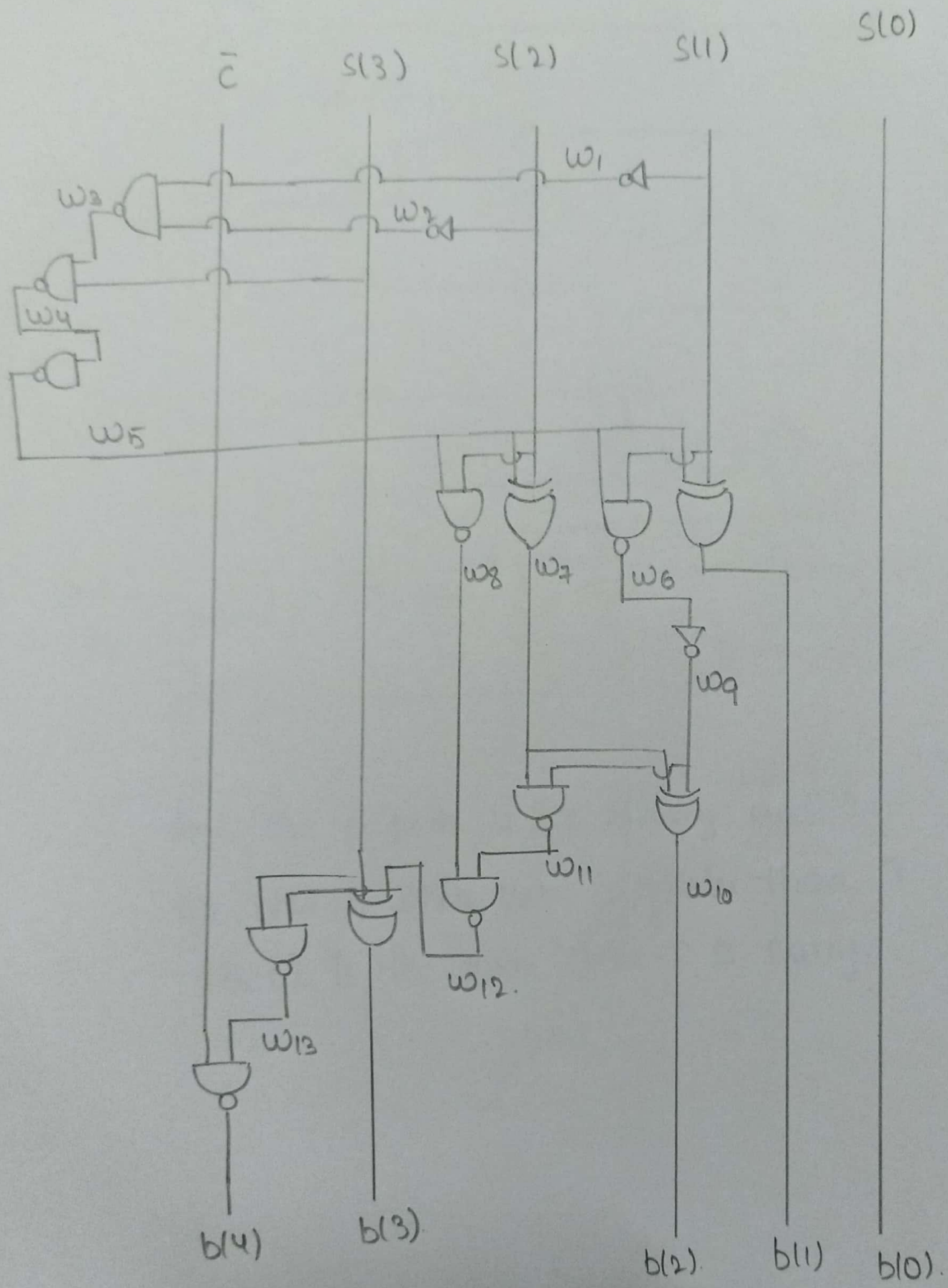


converting to the required gate, we get.





Here we perform 4 bit binary addition.
We add 6 when sum is greater than 9.
S(3:0) is the sum. and C is carry.



$\bar{c}$ can be obtained from previous circuit.

5).

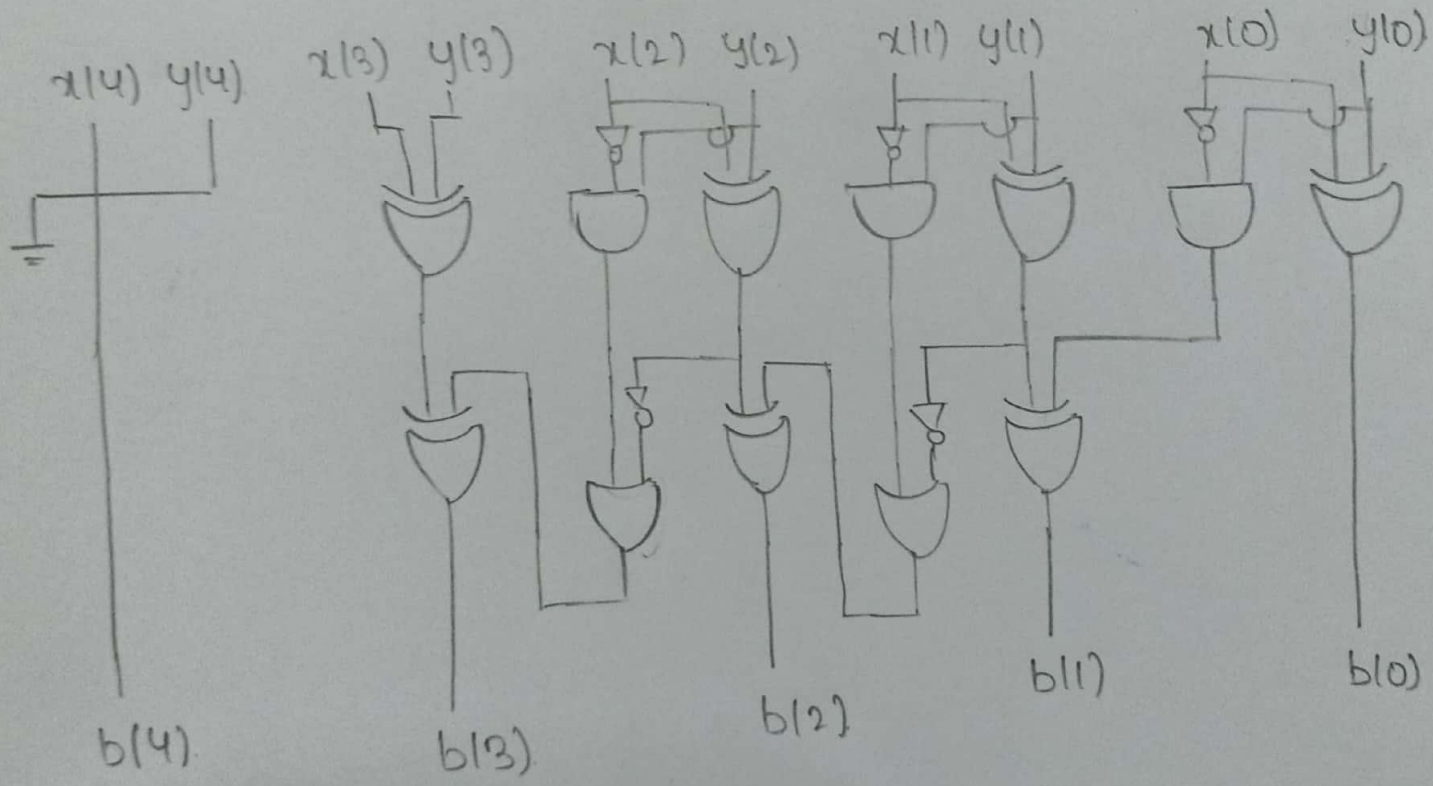
x and y are two single digit decimal integers such that $x \geq y \geq 0$.

x and y are five bit binary form.

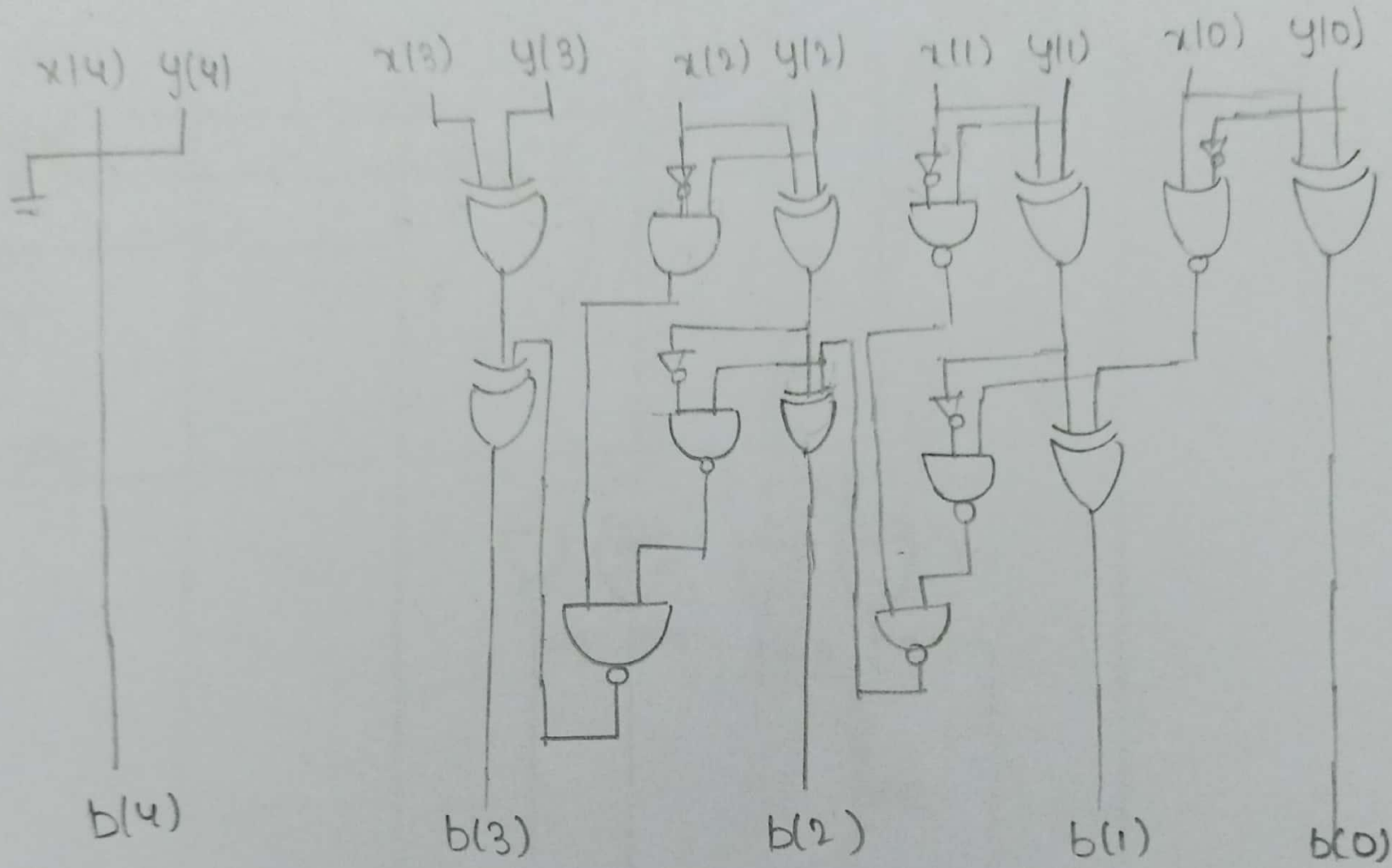
For single digit decimal integers BCD and binary subtraction will be same.

We can use half subtractor for least significant bit (y(0) and x(0))

Given $y \leq x$., there is no need of borrow circuit for MSB.



using required gates



For the 8,9 questions codes are presented below :

In the following, I have implemented the gate-level logic circuits using NOT, NAND, NOR, XOR and XNOR gates, and last four gates must be 2 i/p – 1 o/p gates. I re-used any of the results from the previous questions to a subsequent one just by representing the prior result by a Behavioral-level block diagram.

Problem 1) Suppose, you want to display a single-digit decimal non-negative integer (0 to 9) in a 7-segment LED display by selecting one of buttons numbered 0 to 9 in the input. Also suppose that, to perform addition and subtraction operations, you want to represent a single-digit decimal integer in binary coded decimal (BCD) form using four binary bits. So, you decided to first build a decimal to BCD encoder that will output the BCD in four-bit binary corresponding to one of the selected buttons. Assume that selecting a button corresponds to a logic 1. Build a gate-level logic circuit that will output a single-digit decimal integer in the BCD form depending on the selected number button.

Structural Verilog code :

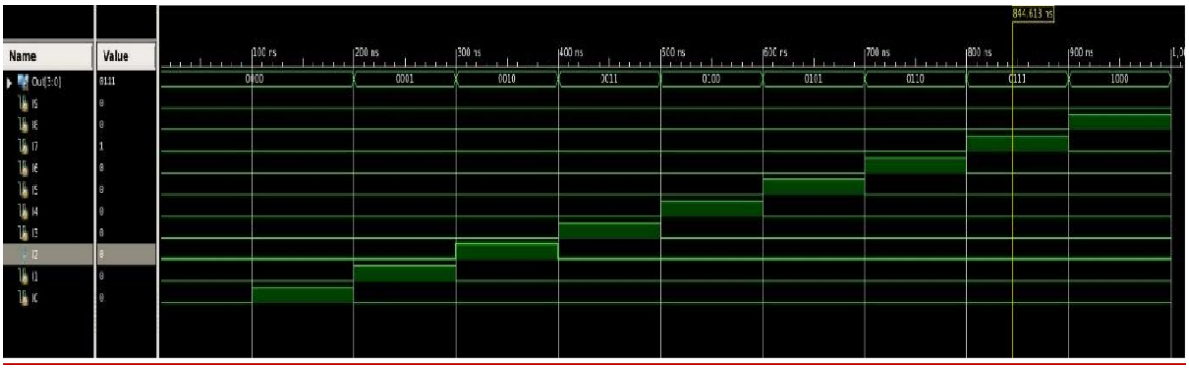
```
module question1(Out,I9,I8,I7,I6,I5,I4,I3,I2,I1,I0);

input I9,I8,I7,I6,I5,I4,I3,I2,I1,I0;

output [3:0]Out;

wire [7:0]x;


//minimized code instead using multiple times primitive gates instantiation

nor g0(x[0],I1,I3),g1(x[1],I5,I7),g3(x[3],I9,x[2]),g5(x[4],I2,I3),g6(x[5],I6,I7),g8(x[6],I4,I5),g10(x[7],I9,I8);

nand g2(x[2],x[0],x[1]),g7(Out[1],x[4],x[5]),g9(Out[2],x[6],x[5]);

not g4(Out[0],x[3]),g11(Out[3],x[7]);

endmodule
```

Test bench

```
module question1_tb;

// Inputs

        reg I9,I8,I7,I6,I5,I4,I3,I2,I1,I0;

// Outputs

        wire [3:0] Out;

// Instantiate the Unit Under Test (UUT)
```

```
        question1 uut (.Out(Out), .I9(I9), .I8(I8), .I7(I7), .I6(I6), .I5(I5), .I4(I4), .I3(I3), .I2(I2), .I1(I1), .I0(I0));

        initial begin

                // Initialize Inputs

        I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 1;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 1;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 1;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 1;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 1;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 1;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 0;I6 = 1;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 0;I7 = 1;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 0;I8 = 1;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 I9 = 1;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100  I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;

#100 $finish ;

        end

end module
```

Timing diagram :



Behavioral Verilog code :

```
module question1_beh(Out,I9,I8,I7,I6,I5,I4,I3,I2,I1,I0);
input I9,I8,I7,I6,I5,I4,I3,I2,I1,I0;
output [3:0]Out;
reg [3:0]Out;
always@(*)
        case({I9,I8,I7,I6,I5,I4,I3,I2,I1,I0})
        //    I-9876543210 bits order
                10'b1000000000 : Out = 4'b1001;
                10'b0100000000 : Out = 4'b1000;
                10'b0010000000 : Out = 4'b0111;
                10'b0001000000 : Out = 4'b0110;
                10'b0000100000 : Out = 4'b0101;
                10'b0000010000 : Out = 4'b0100;
                10'b0000001000 : Out = 4'b0011;
                10'b0000000100 : Out = 4'b0010;
                10'b0000000010 : Out = 4'b0001;
                10'b0000000001 : Out = 4'b0000;
        endcase
endmodule
```

Test bench

```
module question1_beh_tb;
// Inputs
        reg I9,I8,I7,I6,I5,I4,I3,I2,I1,I0;
// Outputs
        wire [3:0] Out;
// Instantiate the Unit Under Test (UUT)
        question1_beh uut (.Out(Out),
.I9(I9),.I8(I8),.I7(I7),.I6(I6),.I5(I5),.I4(I4),.I3(I3),.I2(I2),.I1(I1),.I0(I0));
```

```
        initial begin
                // Initialize Inputs
    I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 1;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 1;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 1;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 1;I2 = 0;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 1;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 0;I5 = 1;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 0;I6 = 1;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 0;I7 = 1;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#10 I9 = 0;I8 = 1;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#10  I9 = 1;I8 = 0;I7 = 0;I6 = 0;I5 = 0;I4 = 0;I3 = 0;I2 = 0;I1 = 0;I0 = 0;
#50 $finish;
        end
endmodule
```

Timing diagram :

Problem 2) Now, for the LED display, assume that the logic output 1 is needed for any of these seven segments in the display to light up the corresponding LED. Build a gate-level logic circuit that will display a single-digit decimal integer in the BCD form to symbolic form in 7-segment LED display.

Structural Verilog code :

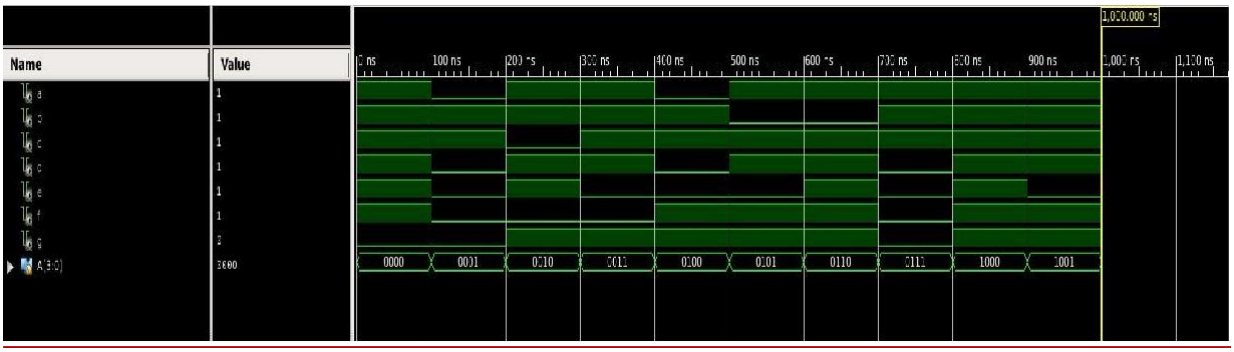
```
module question2(a,b,c,d,e,f,g,A);
 input [3:0]A;
 wire [3:0]Abar;
 output a,b,c,d,e,f,g;
 wire [18:0]x;
  not n[3:0](Abar[3:0],A[3:0]);
 not inv(x[3],x[8]);
xor g0(x[0],A[0],A[2]),g1(x[2],A[1],A[0]);
nor nr1(x[1],A[3],A[1]),nr2(x[4],Abar[1],A[0]),nr3(x[12],e,x[11]),nr4(x[9],Abar[2],A[1]),
nr5(x[15],x[9],x[13]),nr6(x[16],x[14],A[3]),nr7(x[14],A[1],A[0]),nr8(x[13],Abar[2],A[0]),nr9(x[17],x[3],
A[3]),nr10(x[18],x[13],x[9]);

nand
nd1(a,x[0],x[1]),nd2(b,x[2],A[2]),nd3(c,x[4],Abar[2]),nd4(x[6],Abar[2],Abar[0]),nd5(x[7],A[1],Abar[0]),
nd6(e,x[6],x[7]),nd7(d,x[12],Abar[3]),nd8(x[11],x[8],x[10]),nd9(x[8],A[1],Abar[2]),nd10(x[10],x[9],A[0
]),nd11(f,x[15],x[16]),nd12(g,x[17],x[18]);

endmodule
```

Test bench

```
module question2_tb;
// Inputs
        reg [3:0] A;
// Outputs
        wire a,b,c,d,e,f,g;
// Instantiate the Unit Under Test (UUT)
        question2 uut (.a(a),.b(b),.c(c),.d(d),.e(e),.f(f),.g(g),.A(A));

        initial begin
```

```
                    // Initialize Inputs
        A = 4'd0;
#100 A = 4'd1;
#100 A = 4'd2;
#100 A = 4'd3;
#100 A = 4'd4;
#100 A = 4'd5;
#100 A = 4'd6;
#100 A = 4'd7;
#100 A = 4'd8;
#100 A = 4'd9;
#100  A = 4'd0;
      end
endmodule
```

Timing diagram :



Behavioral Verilog code :

In this code we considered abcdefg as [6:0]Y

```
module question2_beh(Y,A);
 input [3:0]A;
 output [6:0]Y;
 reg[6:0]Y;
```

```
 always@(A)
begin
        case(A)
        4'b0000: Y =  7'b1111110;
        4'b0001: Y =  7'b0110000;
        4'b0011: Y =  7'b1111001;
        4'b0010: Y =  7'b1101101;
        4'b0110: Y =  7'b1011111;
        4'b0111: Y =  7'b1110000;
        4'b0101: Y =  7'b1011011;
        4'b0100: Y =  7'b0110011;
        4'b1000: Y =  7'b1111111;
        4'b1001: Y =  7'b1111011;
        4'b1101: Y =  7'b1011011;
        default Y=7'b0000000;
  endcase
end
endmodule
```

Test Bench :

```
module question2_beh_tb;
// Inputs
        reg [3:0] A;
// Outputs
        wire [6:0] Y;
// Instantiate the Unit Under Test (UUT)
question2_beh uut (.Y(Y),.A(A));
initial begin
                // Initialize Inputs
        #100 A = 4'd0;
#100 A = 4'd1;
```

#100 A = 4'd2;

#100 A = 4'd3;

#100 A = 4'd4;

#100 A = 4'd5;

#100 A = 4'd6;

#100 A = 4'd7;

#100 A = 4'd8;

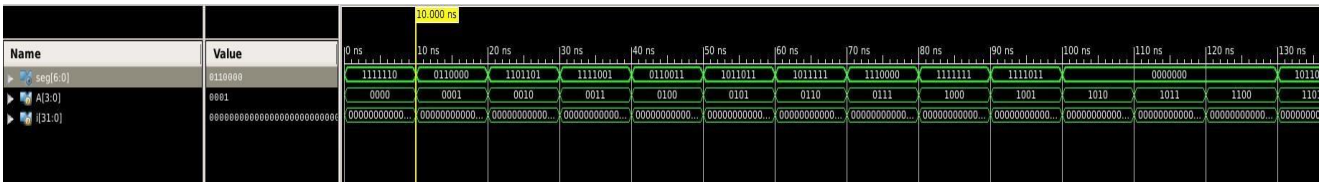
#100 A = 4'd9;

#100 A = 4'd0;

end

endmodule

Timing diagram:



Problem 3) Given two single-digit non-negative decimal integers x and y in the BCD form, build a gate-level logic circuit that will compare the two input integers. There will be 3 logic outputs, and one of these will be logic 1 and all others will be 0 depending on x > y, x < y, or x = y.

Structural Verilog code :

In this code we considered x,y as m,n

module question3(m,n,Y);

input [3:0]m,n;

output [2:0]Y;

wire [3:0]mbar,nbar;

wire [27:0]w;

not n0[3:0](mbar,m),n1[3:0](nbar,n);

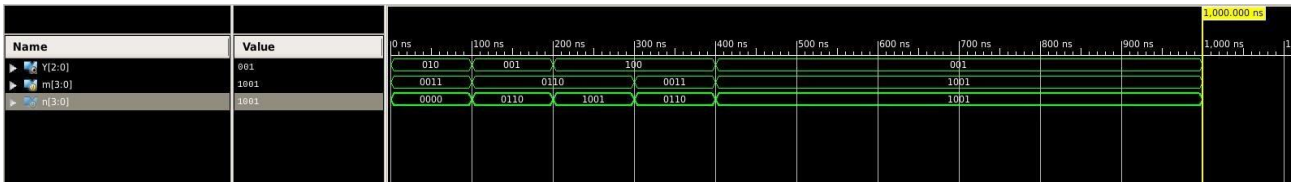
xnor xn[3:0](w[3:0],m,n);

```
nand
nd1(w[5],w[2],w[3]),nd2(w[4],w[1],w[0]),nd3(w[7],m[2],nbar[2]),nd4(w[12],w[1],w[29]),nd5(w[13],
m[0],nbar[0]),nd6(w[9],w[2],w[3]),nd7(w[10],m[1],nbar[1]),nd8(Y[1],w[15],w[16]),nd9(w[19],mbar[2
],n[2]),nd10(w[21],mbar[1],n[1]),nd11(w[23],mbar[0],n[0]),nd12(Y[2],w[26],w[27]);

nor
nr1(Y[0],w[5],w[4]),nr2(w[6],n[3],mbar[3]),nr3(w[8],w[28],w[7]),nr4(w[14],w[12],w[13]),nr5(w[11],w
[9],w[10]),nr6(w[24],w[12],w[23]),nr7(w[15],w[6],w[8]),nr8(w[16],w[11],w[14]),nr9(w[17],x[3],nbar[
3]),nr10(w[18],w[30],w[19]),nr11(w[25],w[9],w[21]),nr12(w[27],w[17],w[18]),nr13(w[26],w[24],w[2
5]);

endmodule
```

Test bench:

```
module question3_tb;

// Inputs

  reg [3:0] m;

  reg [3:0] n;

// Outputs

  wire [2:0] Y;

// Instantiate the Unit Under Test (UUT)

  question3 uut (.m(m),.n(n),.Y(Y));

  initial begin

// Initialize Inputs

        m = 4'd3;n = 4'd0;

#100 m = 4'd6;n = 4'd6;

#100 m = 4'd6;n = 4'd9;

#100 m = 4'd3;n = 4'd6;

#100  m = 4'd9;n = 4'd9;

 end

endmodule
```

Timing diagram:



Behavioral Verilog code:

In this code we considered x,y as m,n

```
module question3_beh(m,n,Y);
input [3:0]m,n;
output [2:0]Y;
reg [2:0]Y;

always@(m,n)
begin
 Y = 3'b0;
     if(m==n) Y[0] = 1;
     else if(m>n) Y[1] = 1;
     else Y[2] = 1;
end
endmodule
```

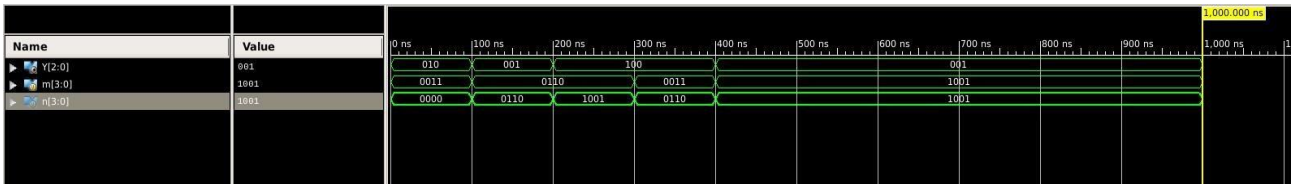
Test bench:

```
module question3_beh_tb;
// Inputs
  reg [3:0] m,n;
// Outputs
  wire [2:0] Y;
// Instantiate the Unit Under Test (UUT)
  question3_beh uut (.m(m),.n(n),.Y(Y));
initial begin
// Initialize Inputs
        m = 4'd3;n = 4'd0;
#100 m = 4'd6;n = 4'd6;
#100 m = 4'd6;n = 4'd9;
#100 m = 4'd3;n = 4'd6;
#100  m = 4'd9;n = 4'd9;
```

end

endmodule

Timing diagram:



Problem 4) You want to perform addition operation on two single-digit decimal integers x and y (both can be positive or zero) in the BCD form. First, build a gate-level logic circuit which will be able to perform addition on x and y (you can use bit-wise or two-bit wise or direct four-bit addition) and will produce the result in a five-bit form. Now, build a gate-level circuit that will convert the five-bit binary result to BCD form so that it can be displayed using two 7-segment LED displays.

Structural Verilog code:

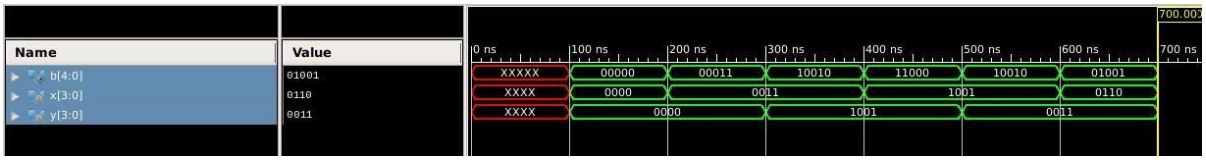
```
module bcdadders(b,x,y);

output [4:0]b;

input [3:0]x;

input [3:0]y;

wire [3:0]s;

wire c;

wire [12:0]j;

wire [13:0]w;


not n1(j[0],j[12]);

xor
g1(b[0],x[0],y[0]),g2(s[1],j[0],j[1]),g3(j[1],x[1],y[1]),g4(j[5],x[2],y[2]),g5(s[2],j[4],j[5]),g6(j[9],x[3],y[3]),

   g7(s[3],j[8],j[9]),l1(j[4],j[2],j[3]),l2(j[8],j[6],j[7]);

xnor l3(c,j[10],j[11]);

nand
k1(j[12],x[0],y[0]),k2(j[2],x[1],y[1]),k3(j[3],j[0],j[1]),k4(j[7],j[4],j[5]),k5(j[6],x[2],y[2]),k6(j[10],x[3],y[3]),

    k7(j[11],j[8],j[9]);

not n1(w[1],s[1]),n2(w[2],s[2]),n3(w[9],w[6]);
```

```
xor x1(b[1],s[1],w[5]),x2(w[7],w[5],s[2]),x3(b[2],w[7],w[9]),x4(b[3],s[3],w[12]);
nand nd1(w[3],w[1],w[2]),nd2(w[4],s[3],w[3]),nd3(w[5],w[4],c),nd4(w[6],w[5],s[1]),
    nd5(w[8],w[5],s[2]),nd6(w[11],w[7],w[9]),nd7(w[12],w[8],w[11]),nd8(w[13],s[3],w[12]),
    nd9(b[4],w[13],c);
endmodule
```

Test Bench:

```
module question4_str_tb;
// Inputs
reg [3:0] x;
reg [3:0] y;
// Outputs
wire [4:0] b;
// Instantiate the Unit Under Test (UUT)
question4_str uut (.b(b),.x(x),.y(y));
initial begin
// Initialize  Inputs
#100 x = 4'b0; y = 4'b0;
#100 x = 4'd3;
#100 y = 4'd9;
#100 x = 4'd9;
#100 y = 4'd3;
#100 x = 4'd6;
#100 $finish;
end
endmodule
```

Timing diagram:



Behavioral Verilog code:

```
module question4_beh(sum,x,y);
 input [4:0]x,y;
 output [4:0]sum;
 reg [4:0]sum,bcd;
 always@(x,y)
 begin
   sum= x+ y ;
 if(sum>9)
      bcd = sum + 5'b00110;
 else
      bcd = sum;
 end
 endmodule
```

Test Bench:

```
module question4_beh_tb;
//  Inputs
   reg [4:0] x;
   reg [4:0] y;
// Outputs
   wire [4:0] sum;
// Instantiate the Unit Under Test (UUT)
   question4_beh uut (.sum(sum),.x(x),.y(y));
initial begin
// Initialize Inputs
         x = 5'd1;y = 5'd3;
#100 x = 5'd3;y = 5'd6;
#100 x = 5'd3;y = 5'd6;
#100  x = 5'd9;y = 5'd6;
```

#100 x = 5'd6;y = 5'd6;

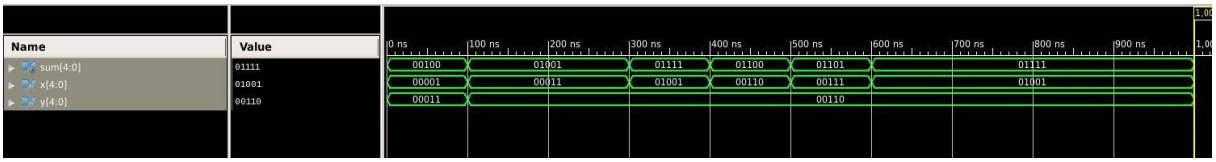
#100 x = 5'd7;y = 5'd6;

#100 x = 5'd9;y = 5'd6;

end

endmodule

Timing diagram:



Problem 5) You also want to perform subtraction operation on two single-digit decimal integers x and y ($x \geq y \geq 0$) represented in the BCD form. First, build a gate-level logic circuit which will be able to subtract a smaller non-negative number y from x, both of which are in five-bit binary form, and will produce a result in a five-bit binary form. Build a logic circuit, using the one you built or otherwise, to perform x–y with x, y and (x–y) in BCD form. (You can use any subtraction scheme.)

Structural Verilog code:

module question5(b,x,y);

input [4:0]x;

input [4:0]y;

output [4:0]b;

wire [15:1]w;

not n1(w[1],y[0]),n2(w[4],x[1]),n3(w[7],x[2]),n4(w[10],w[3]),n5(w[13],w[6]);

xor x1(b[0],x[0],y[0]),x2(w[3],x[1],y[1]),x3(w[6],x[2],y[2]),x4(w[9],x[3],y[3]),

x5(b[1],w[2],w[3]),x6(b[2],w[12],w[6]),x7(b[3],w[15],w[9]),x8(b[4],x[4],y[4]);

nand nd1(w[5],w[4],y[1]),nd2(w[8],w[7],y[2]),nd3(w[11],w[2],w[10]),

nd4(w[12],w[5],w[11]),nd5(w[14],w[12],w[13]),nd6(w[15],w[8],w[14]);

nor nr1(w[2],w[1],x[0]);

endmodule

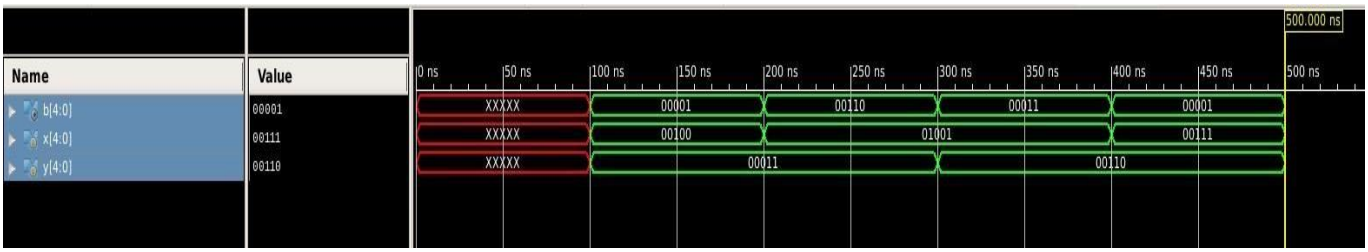
Test Bench:

module question5_tb;

// Inputs

```
reg [4:0] x;
reg [4:0] y;
// Outputs
wire [4:0] b;

// Instantiate the Unit Under Test (UUT)
question5 uut (.b(b),.x(x),.y(y));
initial begin
#100 x = 5'd4;y = 5'd3;
#100 x = 5'd9;
#100 y = 5'd6;
#100 x = 5'd7;
#100 $finish;
end
endmodule
```

Timing diagram:



Behavioral Verilog code:

```
module question5_beh(f,x,y);
  input [4:0]x;
  input [4:0]y;
  output reg [4:0]f;
  always@(*)
  begin
```

```
      f = x - y;
end
endmodule
```

Test Bench:

```
module question5_beh_tb;
// Inputs
  reg [4:0] x;
  reg [4:0] y;
// Outputs
  wire [4:0] f;
// Instantiate the Unit Under Test (UUT)
  question5_beh uut (.f(f),.x(x),.y(y));
  initial begin
// Initialize Inputs
  #100 x = 5'd4;y = 5'd3;
  #100 x = 5'd9;
  #100 y = 5'd6;
  #100 x = 5'd7;
  #100 $finish;
end
endmodule
```

Timing diagram: