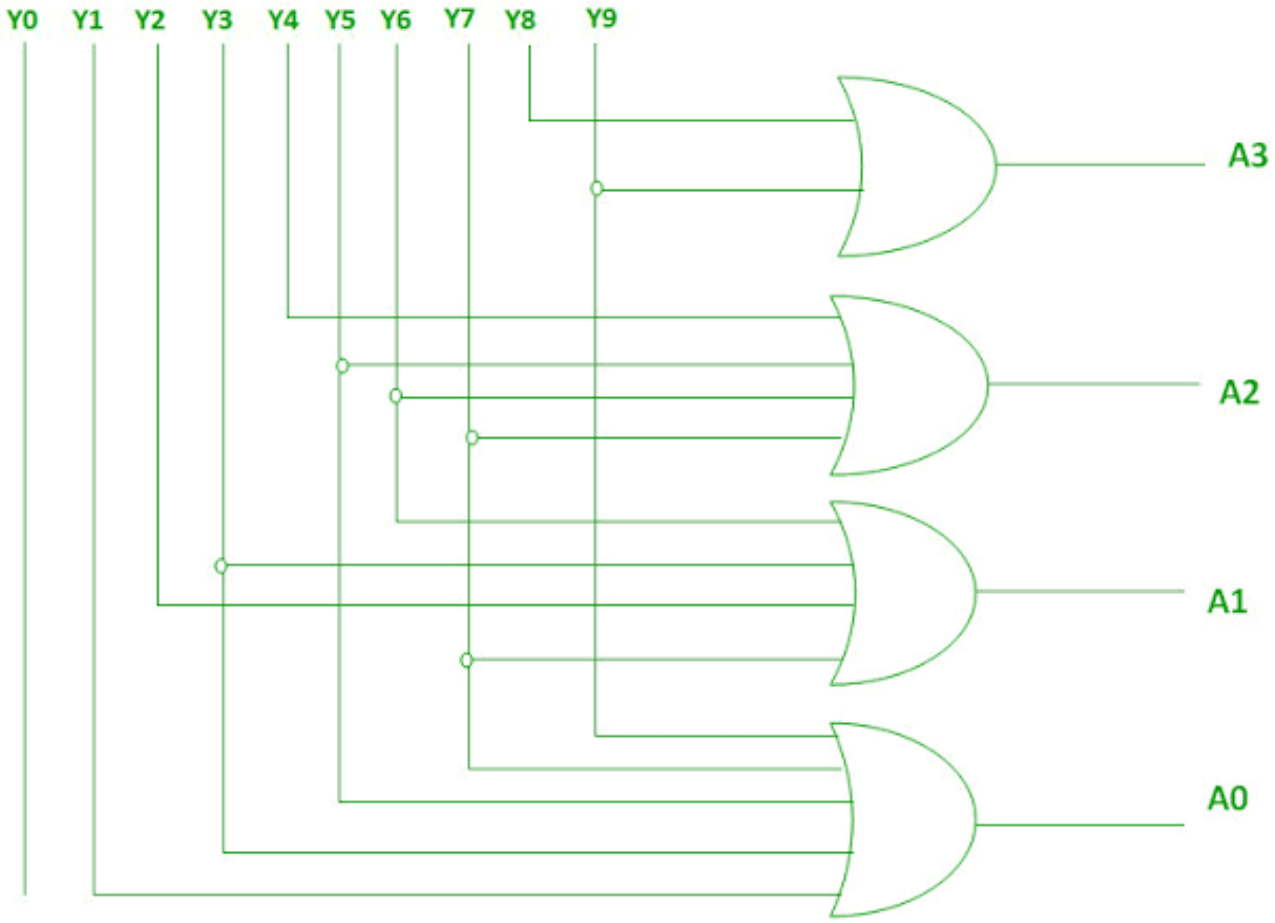
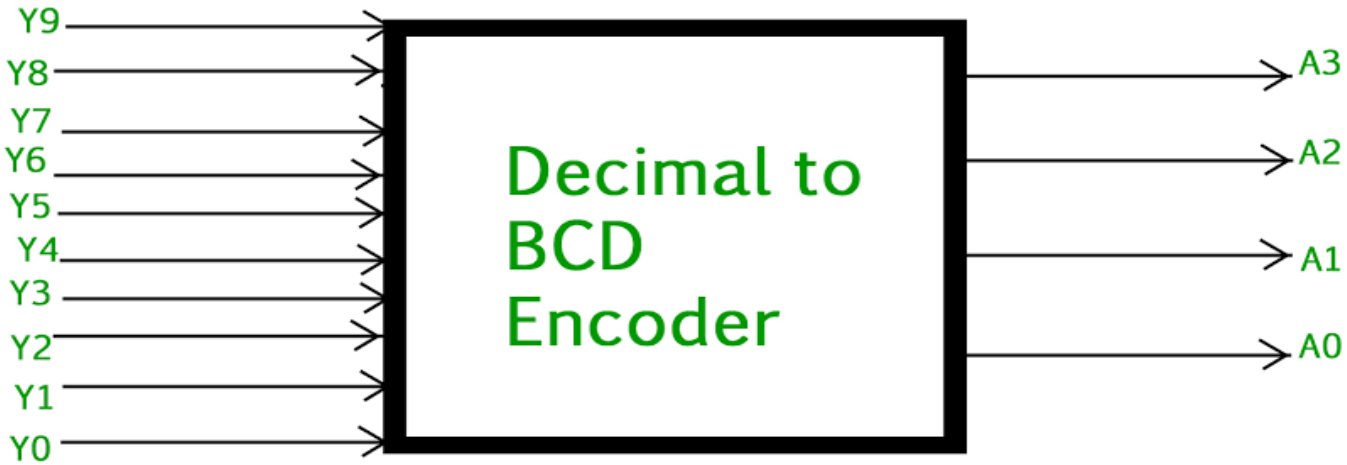
Q1. CONVERSION OF THE DECIMAL NUMBER FROM 0-9 INTO THE BCD FORMAT USING DECIMAL-BCD ENCODER.





INPUT OUTPUT TABLE

INPUTS										OUTPUTS			
Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_3	A_2	A_1	A_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

BEHAVIORAL LEVEL VERILOG COD

```
module dec_bcd1(decimal,bcd);
input [9:0]decimal;
output reg [3:0] bcd;
always@(decimal)
begin
   case(decimal)
   0:bcd=4'b0000;
   1:bcd=4'b0001;
   2:bcd=4'b0010;
   3:bcd=4'b0011;
   4:bcd=4'b0100;
   5:bcd=4'b0101;
   6:bcd=4'b0110;
   7:bcd=4'b0111;
   8:bcd=4'b1000;
   9:bcd=4'b1001;
   default : bcd=4'b0000;

   endcase
end
endmodule
```

Testbench for above code

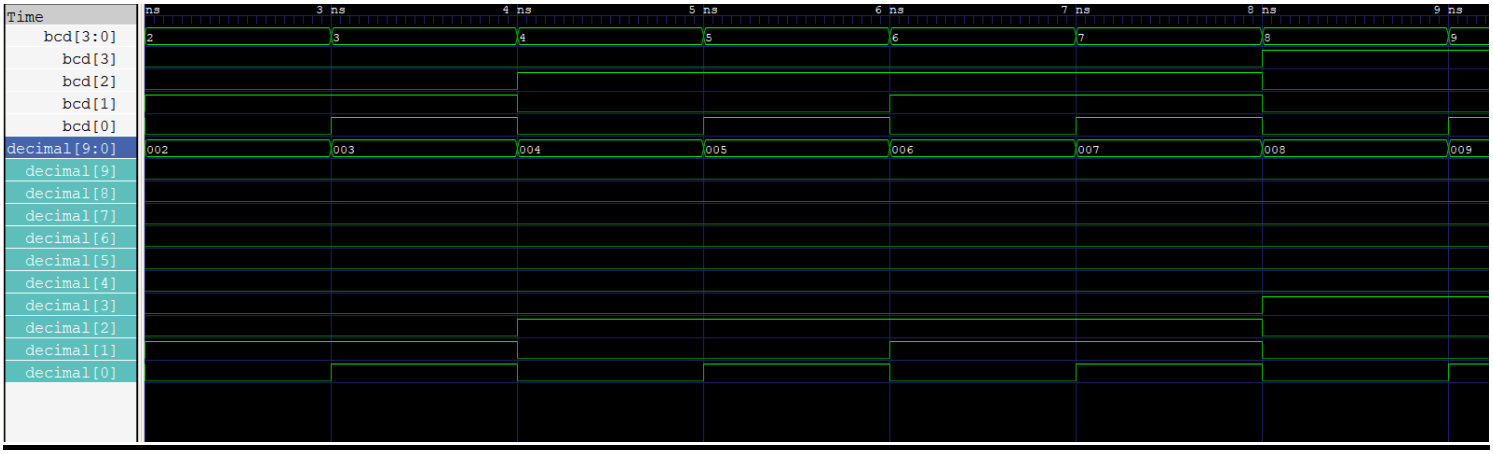
```
`timescale 1ns/1ns
`include "dec_bcd1.v"
module dec_bcd1_tb;
reg [9:0]decimal;
wire [3:0] bcd ;
dec_bcd1 uut (decimal,bcd);
initial begin
$dumpfile("dec_bcd1_tb.vcd");
$dumpvars(0,dec_bcd1_tb);
Decimal=16'd0;
```

```
#1
decimal= 16'd1;
#1
decimal= 16'd2;
#1
decimal= 16'd3;
#1
decimal= 16'd4;
#1
decimal= 16'd5;
#1
decimal= 16'd6;
#1
decimal= 16'd7;
#1
decimal= 16'd8;
#1
decimal= 16'd9;
#1
$finish;
end
initial
$monitor($time," decimal number =%d, bcd=%b",decimal,bcd );

Endmodule
```

OUTPUT ON TERMINAL

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

C:\iverilog\bin>vvp dec_bcd1_tb.vvp
VCD info: dumpfile dec_bcd1_tb.vcd opened for output.
                    0 decimal number =   0, bcd=0000
                    1 decimal number =   1, bcd=0001
                    2 decimal number =   2, bcd=0010
                    3 decimal number =   3, bcd=0011
                    4 decimal number =   4, bcd=0100
                    5 decimal number =   5, bcd=0101
                    6 decimal number =   6, bcd=0110
                    7 decimal number =   7, bcd=0111
                    8 decimal number =   8, bcd=1000
                    9 decimal number =   9, bcd=1001
dec_bcd1_tb.v:30: $finish called at 10 (1ns)
```

OUTPUT ON GTK WAVE



STRUCTURAL LEVEL VERILOG CODE

```
module dec_bcd(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,B0,B1,B2,B3);
wire t1,t2,t3,t4,t5,t6,t7,t8,t9,t10;
input D0;input D1;input D2;input D3;input D4;input D5;input D6;input D7;input D8;input
D9;output B0;output B1;output B2;output B3;

nor c1(t1,D1,D3);
nand c2(t2,1,t1);
nor c3(t4,t2,D5);
nor c4(t3,D7,D9);
```

```
nand c5(B0,t4,t3);

nor c6(t5,D2,D3);
nor c7(t6,D5,D6);
nand c8(B1,t5,t6);

nor c9(t7,D4,D5);
nor c10(t8,D6,D7);
nand c11(B2,t8,t7);

not c12(t9,D8);
not c13(t10,D9);
nand c14(B3,t9,t10);

endmodule
```

TESTBENCH FOR ABOVE CODE

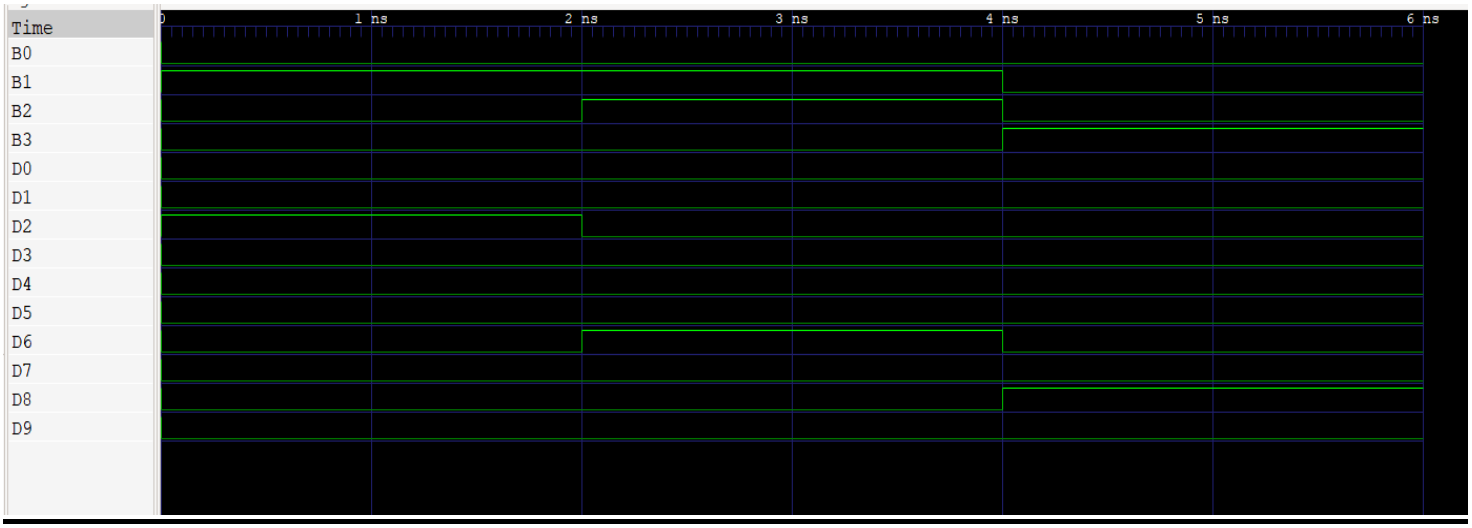
```
`timescale 1ns/1ns
`include"dec_bcd.v"

module tb_dec_bcd;
reg D0,D1,D2,D3,D4,D5,D6,D7,D8,D9;
wire B0,B1,B2,B3;

dec_bcd uut(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,B0,B1,B2,B3);
initial
begin
   $dumpfile("tb_dec_bcd.vcd");
   $dumpvars(0,tb_dec_bcd);
D0=1'b0; D1=1'b0;
 D2=1'b1;
 D3=1'b0;
 D4=1'b0;
 D5=1'b0;
 D6=1'b0;
```

```
 D7=1'b0;
 D8=1'b0;
 D9=1'b0;
 #2

 D0=1'b0;
 D1=1'b0;
 D2=1'b0;
 D3=1'b0;
 D4=1'b0;
 D5=1'b0;
 D6=1'b1;
 D7=1'b0;
 D8=1'b0;
 D9=1'b0;

 #2
 D0=1'b0;
 D1=1'b0;
 D2=1'b0;
 D3=1'b0;
 D4=1'b0;
 D5=1'b0;
 D6=1'b0;
 D7=1'b0;
 D8=1'b1;
 D9=1'b0;

 #2 $finish;



end
initial $monitor($time,"B0=%b,B1=%b,B2=%b,B3=%b",B0,B1,B2,B3);
endmodule
```

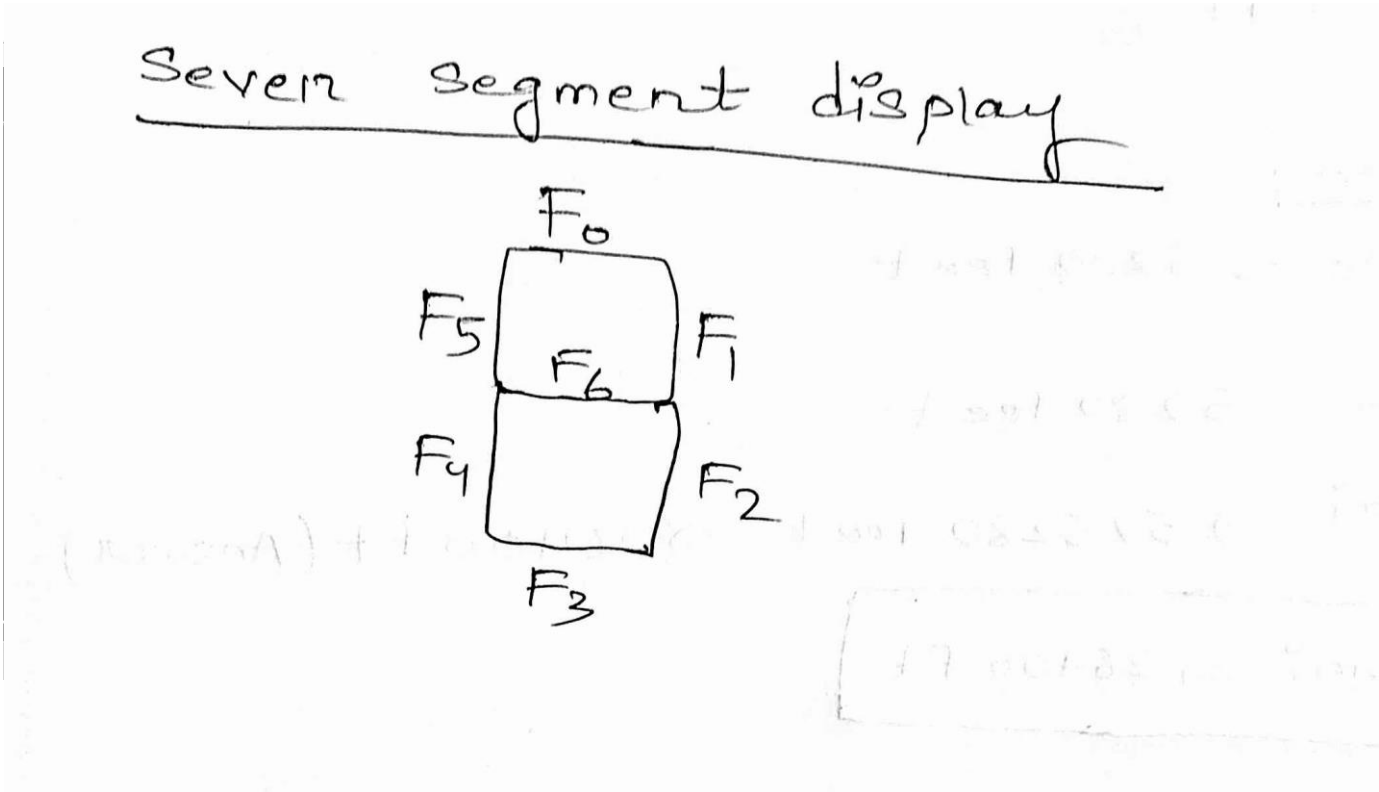
OUTPUT ON TERMINAL

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

                    9 decimal number =   9, bcd=1001
dec_bcd1_tb.v:30: $finish called at 10 (1ns)

C:\iverilog\bin>iverilog -o tb_dec_bcd.vvp tb_dec_bcd.v

C:\iverilog\bin>vvp tb_dec_bcd.vvp
VCD info: dumpfile tb_dec_bcd.vcd opened for output.
                    0B0=0,B1=1,B2=0,B3=0
                    2B0=0,B1=1,B2=1,B3=0
                    4B0=0,B1=0,B2=0,B3=1
tb_dec_bcd.v:48: $finish called at 6 (1ns)

C:\iverilog\bin>
```

OUTPUT ON GTKWAVE



QUESTION -2



LOGIC FRIDAY (MINIMISATION)

Logic Friday

File Operation Truthtable Equation Gates View Help

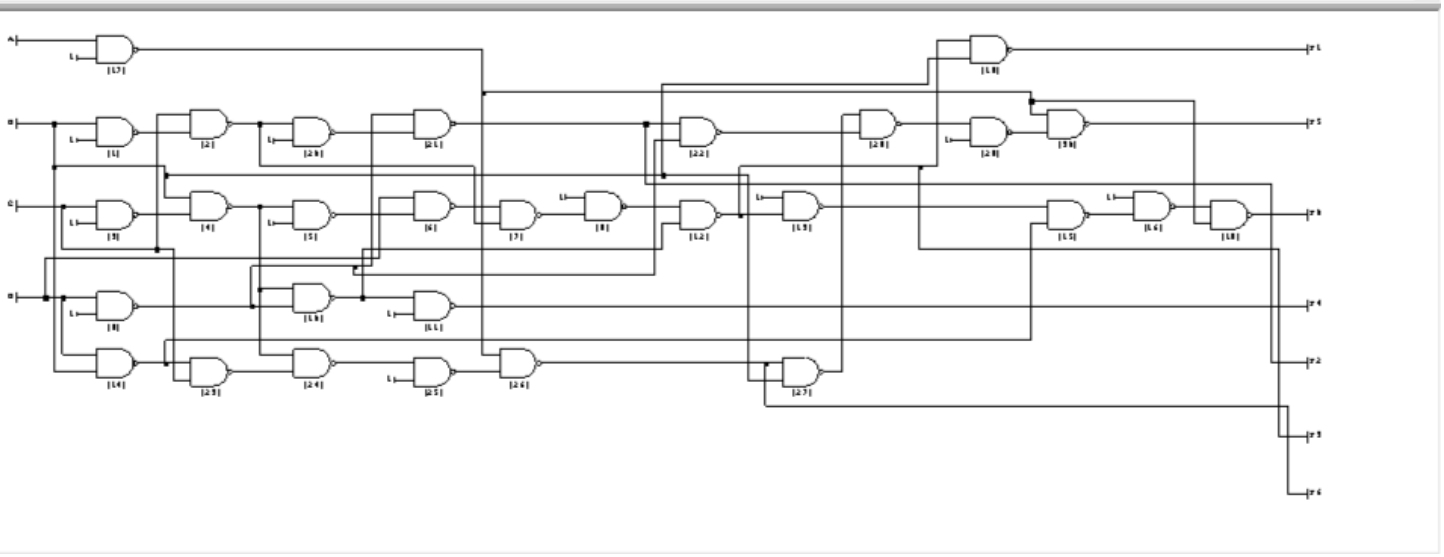
Functi... | Inputs | Outputs | True | False | DC | PI | Gates

<none>

Term	A	B	C	D	=>	F0	F1	F2	F3	F4	F5	F6
0	0	0	0	0		1	1	1	1	1	1	0
1	0	0	0	1		0	1	1	0	0	0	0
2	0	0	1	0		1	1	0	1	1	0	1
3	0	0	1	1		1	1	1	1	0	0	1
4	0	1	0	0		0	1	1	0	0	1	1
5	0	1	0	1		1	0	1	1	0	1	1
6	0	1	1	0		1	0	1	1	1	1	1
7	0	1	1	1		1	1	1	0	0	0	0
8	1	0	0	0		1	1	1	1	1	1	1
9	1	0	0	1		1	1	1	0	0	1	1
10	1	0	1	0		X	X	X	X	X	X	X
11	1	0	1	1		X	X	X	X	X	X	X
12	1	1	0	0		X	X	X	X	X	X	X
13	1	1	0	1		X	X	X	X	X	X	X
14	1	1	1	0		X	X	X	X	X	X	X
15	1	1	1	1		X	X	X	X	X	X	X

Double-click an output cell to change state, or select a range and use the menu. Press <Enter> when done.

```
Minimized:
F0 = B' D' + A  + C  + B D;
F1 = C D + B'  + C' D';
F2 = C'  + B  + D;
F3 = B' D' + B C' D + C D' + B' C ;
F4 = B' D' + C D';
F5 = B D' + A  + B C'  + C' D';
F6 = C D' + A  + B C'  + B' C ;
```



VERILOG CODES –

1.BEHAVIORAL VERILOG CODE-

```
module segment(bcd,seg);

input [3:0] bcd;

output reg [6:0] seg;

always@(bcd)

begin

   case(bcd)

   0:seg=7'b1111110;

   1:seg=7'b0110000;

   2:seg=7'b1101101;

   3:seg=7'b1111001;
```

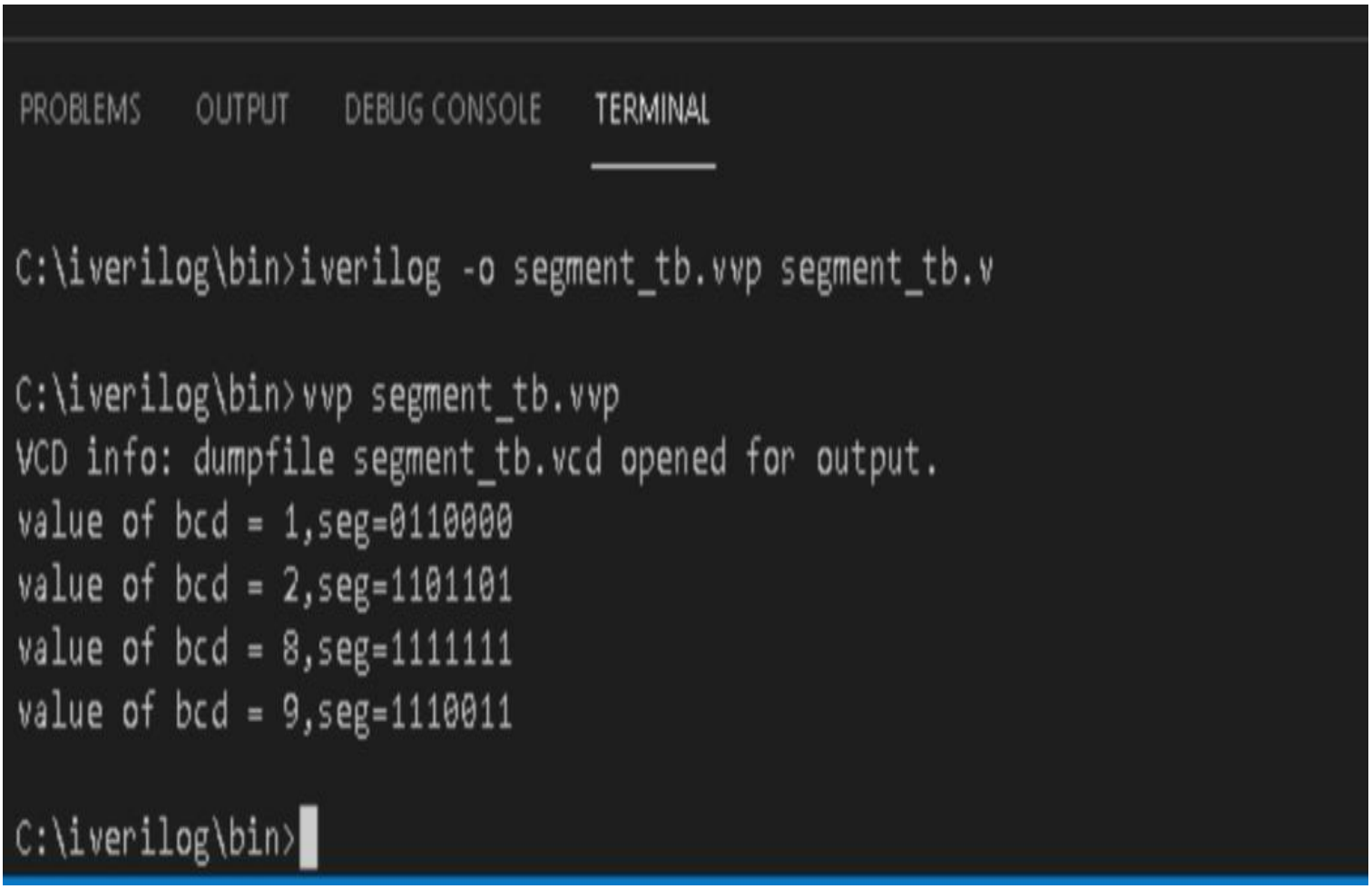
```
   4:seg=7'b0110011;
   5:seg=7'b1011011;
   6:seg=7'b1011111;
   7:seg=7'b1110000;
   8:seg=7'b1111111;
   9:seg=7'b1110011;
   default : seg=7'b0000000;
   endcase
end
endmodule
```

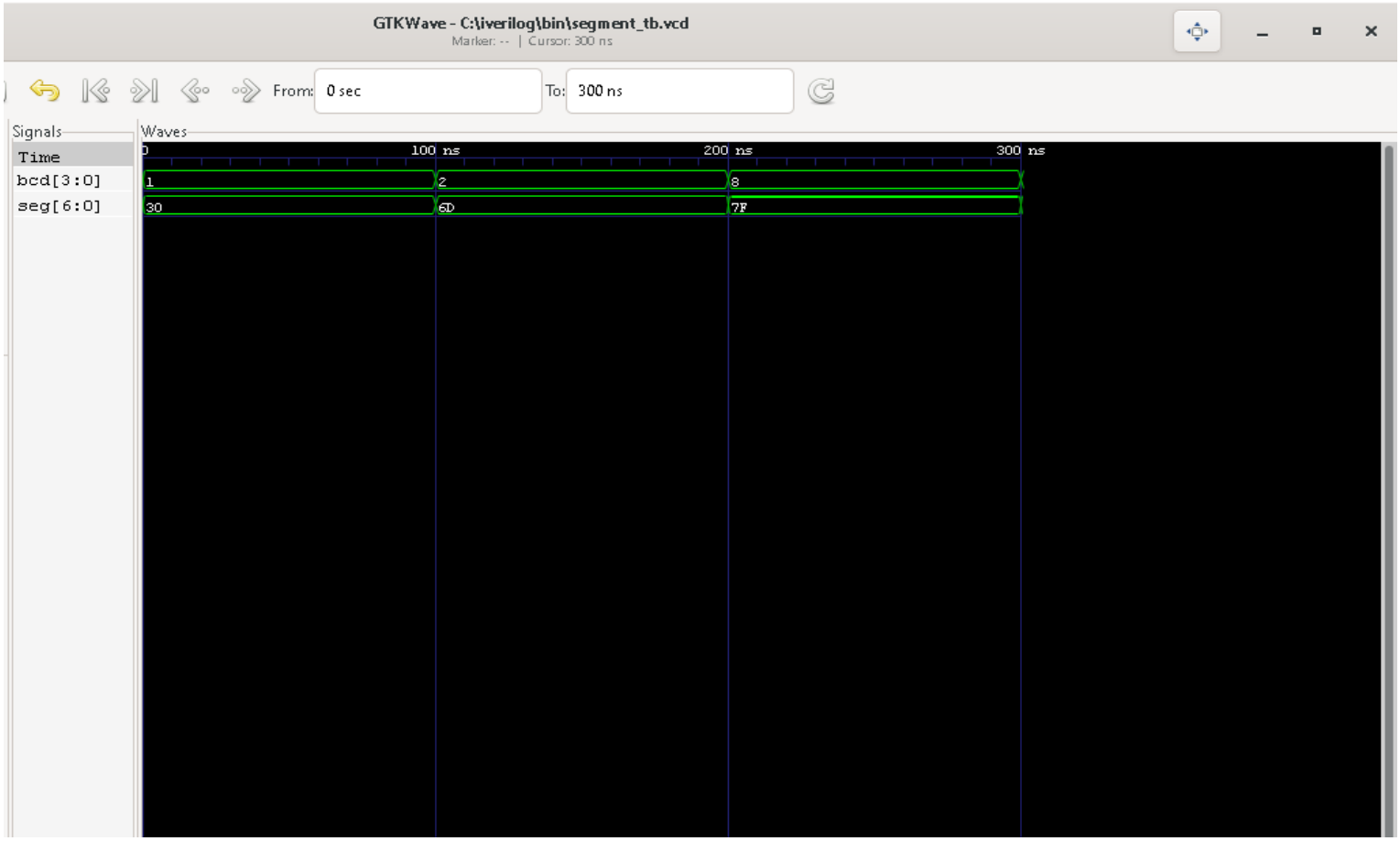
TESTBENCH CODE-

```
`timescale 1ns/1ns
`include "segment.v"
module segment_tb;
reg [3:0] bcd;
wire [6:0] seg;
segment uut(bcd,seg);
initial begin
$dumpfile("segment_tb.vcd");
$dumpvars(0,segment_tb);
$monitor("value of bcd =%d,seg=%b",bcd,seg);
bcd = 4'd1;
#100
```

```
bcd = 4'd2;
#100
bcd = 4'd8;
#100
bcd = 4'd9;
end
endmodule
```

OUTPUT-





STRUCTURAL VERILOG CODE

```
module display(a,b,c,d,f0,f1,f2,f3,f4,f5,f6);

wire t[0:39];

wire x,y;

input a;input b;input c;input d;output f0;output f1;output f2;output f3;output f4;output f5;output f6;

xnor (x,b,d);

not c1(t[1],x);

not c2(t[2],a);

not c3(t[3],c);
```

```
nand c4(t[4],t[1],t[2]);
nand c5(t[5],t[4]);
nand c6(f0,t[5],t[3]);
xnor c7(y,c,d);
not c8(t6,y);
nand c9(f1,b,t[6]);
not c10(t[7],b);
not c11(t[8],d);
nand c12(t[9],t[7],c);
not c13(t[10],t[9]);
nand c14(f2,t[10],t[8]);
not c15(t[11],b);
not c16(t[12],d);
not c17(t[13],c);
nand c18(t[14],t[12],c);
nand c19(t[15],t[13],d);
nand c20(t[16],t[15],t[11]);
not c21(t[17],t[15]);
not c22(t[18],t[17],b);
nand c23(t[19],t[18]);
nand c24(t[20],t[16],t[18]);
not c25(t[21],t[20]);
nand g1(f3,t[21],t[14]);
```

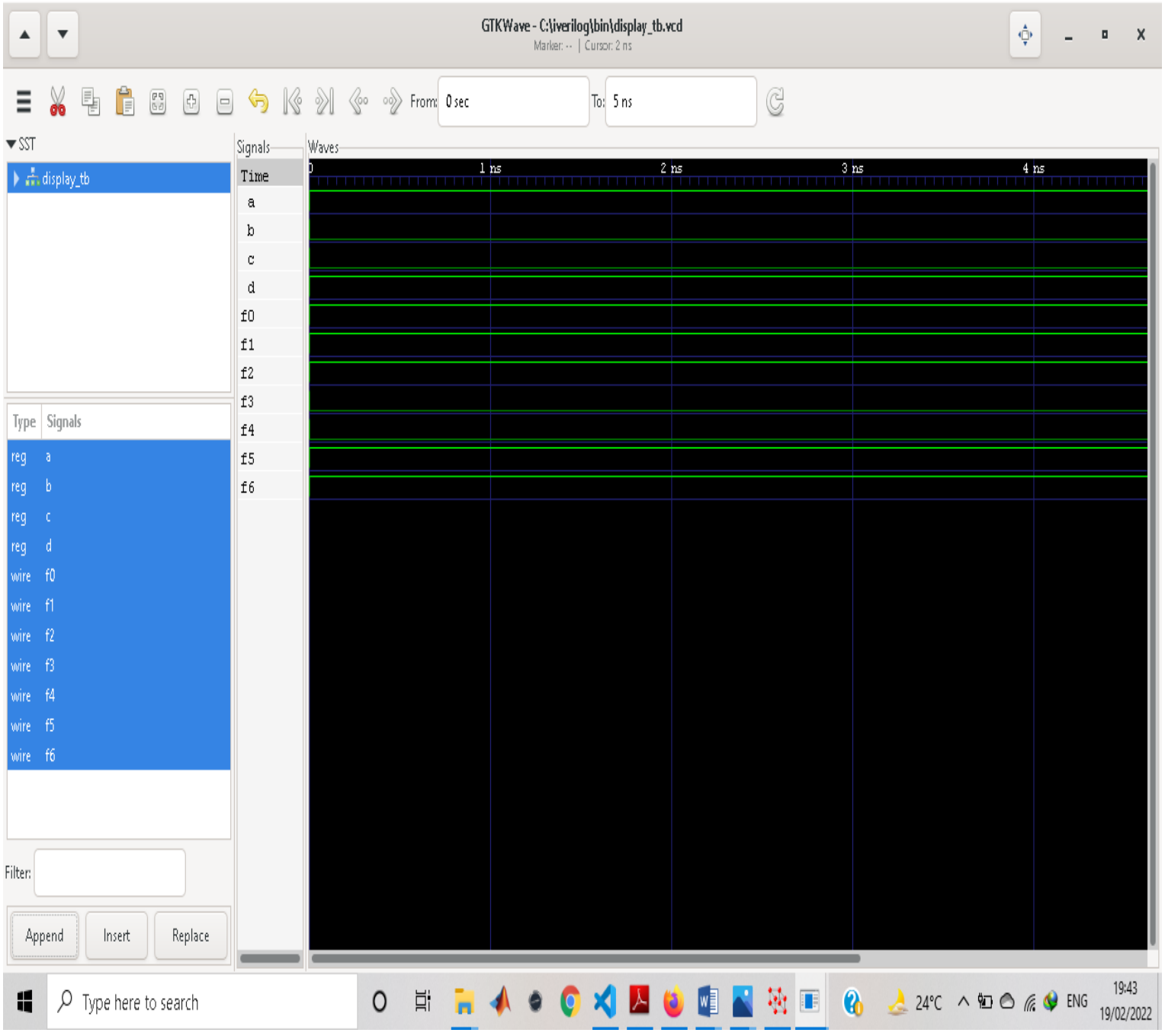
```
not c26(t[22],b);
nor c27(t[23],t[22],c);
nor c28(f4,t[23],d);
not c29(t[24],b);
not c30(t[25],d);
not c31(t[26],a);
not c32(t[27],c);
nand c33(t[28],t[24],d);
nand c34(t[29],t[25],b);
nand c35(t[30],t[28],t[27]);
nand c36(t[31],t[30],t[29]);
not c37(t[32],t[31]);
nand c38(f5,t[32],t[26]);
nand c39(t[33],c,d);
not c40(t[34],a);
not c41(t[35],b);
nand c42(t[36],t[33],b);
nand c43(t[37],t[35],c);
nand c44(t[38],t[37],t[36]);
not c45(t[39],t[38]);
nand c46(f6,t[39],t[34]);
endmodule
```

TESTBENCH CODE –

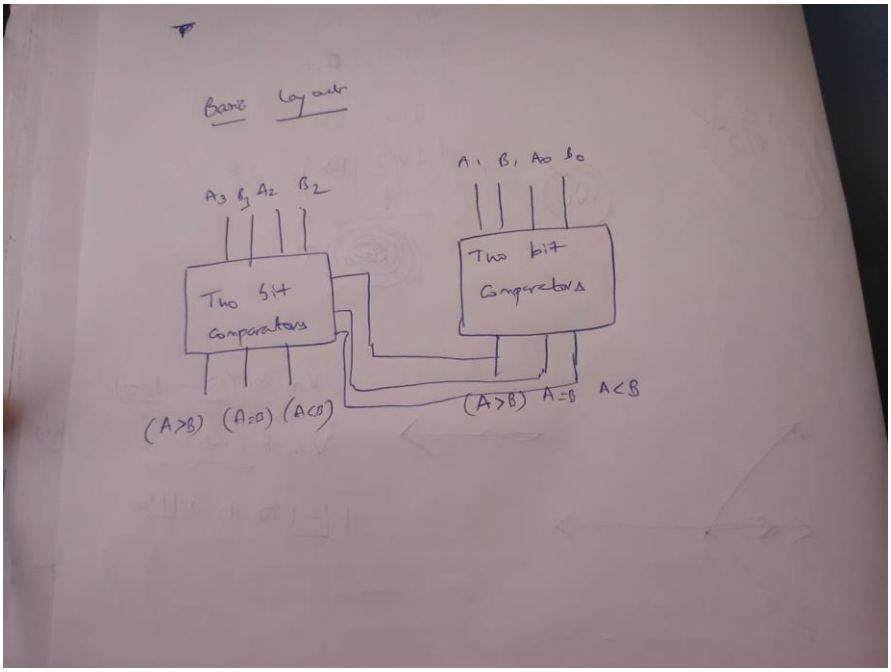
```
`timescale 1ns/1ns
`include"display.v"
module display_tb;
reg a,b,c,d;
wire f0,f1,f2,f3,f4,f5,f6;
display uut(a,b,c,d,f0,f1,f2,f3,f4,f5,f6);
initial
begin
   $dumpfile("display_tb.vcd");
   $dumpvars(0,display_tb);
   a=1'b1;
   b=1'b0;
   c=1'b0;
   d=1'b1;
#5;

end
initial
$monitor($time,"f0=%d,f1=%d,f2=%d,f3=%d,f4=%d,f5=%d,f6=%d",f0,f1,f2,f3,f4,f5,f6);
endmodule
```

OUTPUT-

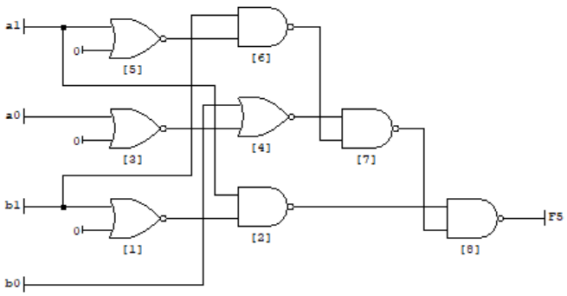
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\iverilog\bin>iverilog -o display_tb.vvp display_tb.v

C:\iverilog\bin>vvp display_tb.vvp
VCD info: dumpfile display_tb.vcd opened for output.
                  0f0=1,f1=1,f2=1,f3=0,f4=0,f5=1,f6=1
```



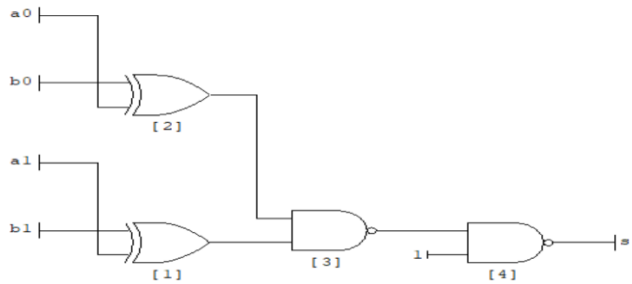
Q3. 4 bit comparator using two 2 bit comparators.



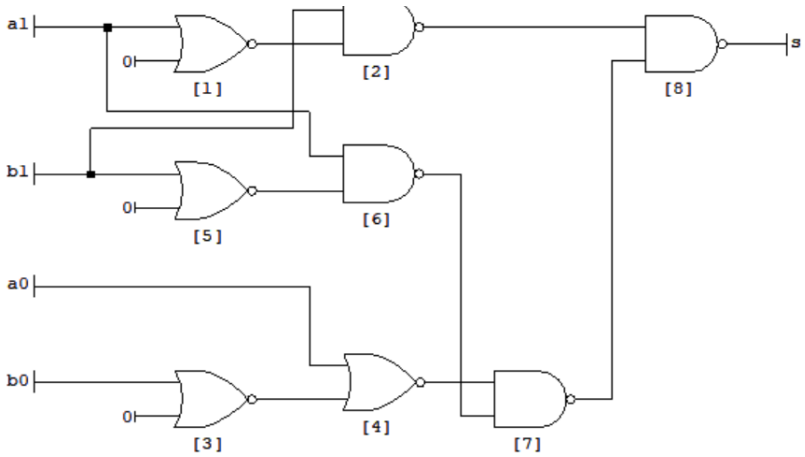
Two bit comparator circuit.
a>b:

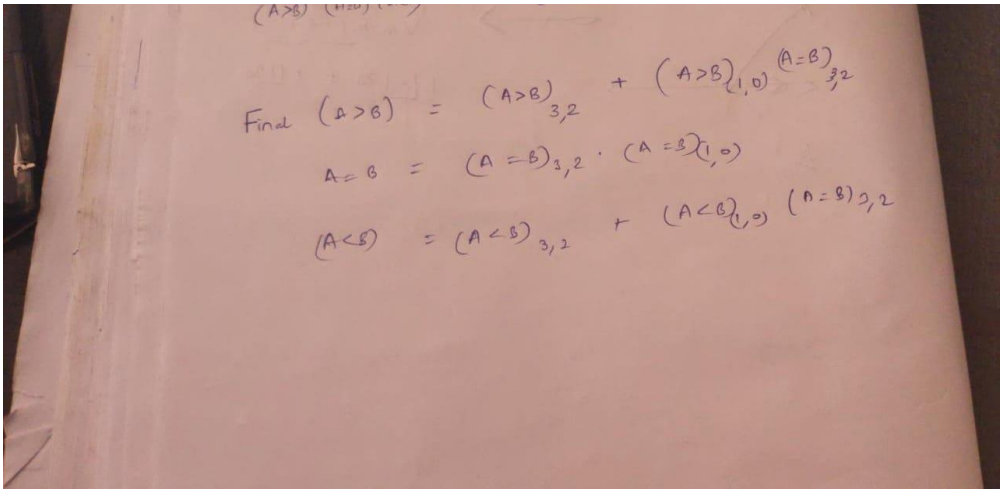


a=b



a<b





structural code:

```
 `include "twobit.v"

 module cst(greater,equal,less,a,b);
output greater;
output  equal;
output  less;
input [3:0]a;
input [3:0]b;
wire g1,g2;
wire l1,l2;
wire e1,e2;
wire t1,t2,t3;
```

```
twobit comp1(g1,l1,e1,a[1],a[0],b[1],b[0]);
twobit comp2(g2,l2,e2,a[3],a[2],b[3],b[2]);

nand c1(t1,g1,e2);
nand c2(t2,g2,1);
nand c3(greater,t1,t2);

nand c4(t3,e1,e2);
nand c5(equal,t3,1);

nand c6(t4,l1,e2);
not c7(t5,l2);
nand c8(less,t4,t5);

endmodule
```

Two bit comparator:

```
module twobit(G,L,E,a1,a0,b1,b0);
output G;output L;output E;
input a1;
input a0;
input b1;
input b0;
input g;
input l;
input e;
wire t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t16,t17;
//greater
nor c1(t1,a1,1'b0);
nand c2(t2,b1,t1);
nor c3(t3,b1,1'b0);
nand c4(t4,t3,a1);
nor c5(t5,a0,1'b0);
nor c6(t6,b0,t5);
nand c7(t7,t6,t2);
nand c8(G,t4,t7);
```

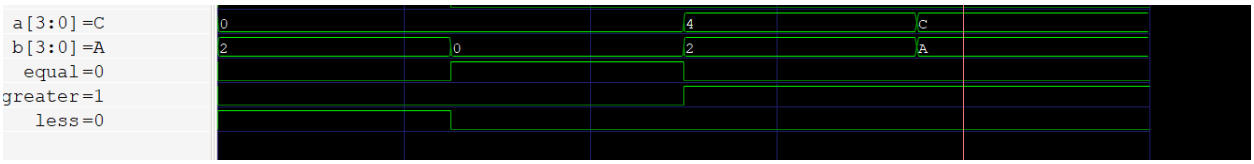
```
// equality check
xnor c9(t8,a0,b0);
xnor c10(t9,a1,b1);
nand c11(t10,t9,t8);
nand c12(E,t10,1);
// less than check

nor c13(t11,a1,1'b0);
nor c14(t13,b1,1'b0);
nand c15(t12,t11,b1);
nand c16(t14,t13,a1);
nor c17(t15,b0,1'b0);
nor c18(t16,a0,t15);
nand c19(t17,t16,t14);
nand c20(L,t12,t17);
endmodule
```

Test bench for comparator:

```
`timescale 1ns/1ns
`include "cst.v"

module cst_tb;
reg [3:0]a;
reg [3:0]b;
wire greater;
wire equal;
wire less;
wire g1,g2;
wire l1,l2;
wire e1,e2;

cst uut(greater,equal,less,a,b);
initial begin
    $dumpfile("cst1.vcd");
```

```
    $dumpvars(0,cst_tb);




      a=4'b0000;b=4'b0010;
      #5 a=4'b0000;b=4'b0000;

      #5 a=4'b0100;b=4'b0010;
      #5 a=4'b1100;b=4'b1010;

    #5 $display($time,"greater=%b,equal=%b,less=%b",greater,equal,less);



end


endmodule
```

Results:



Behavioural code:

```
module c(input[3:0] a,input[3:0] b,output greater,output equal,output
less);

reg greater;
reg equal;
reg less;
always @( a or b)
begin
```

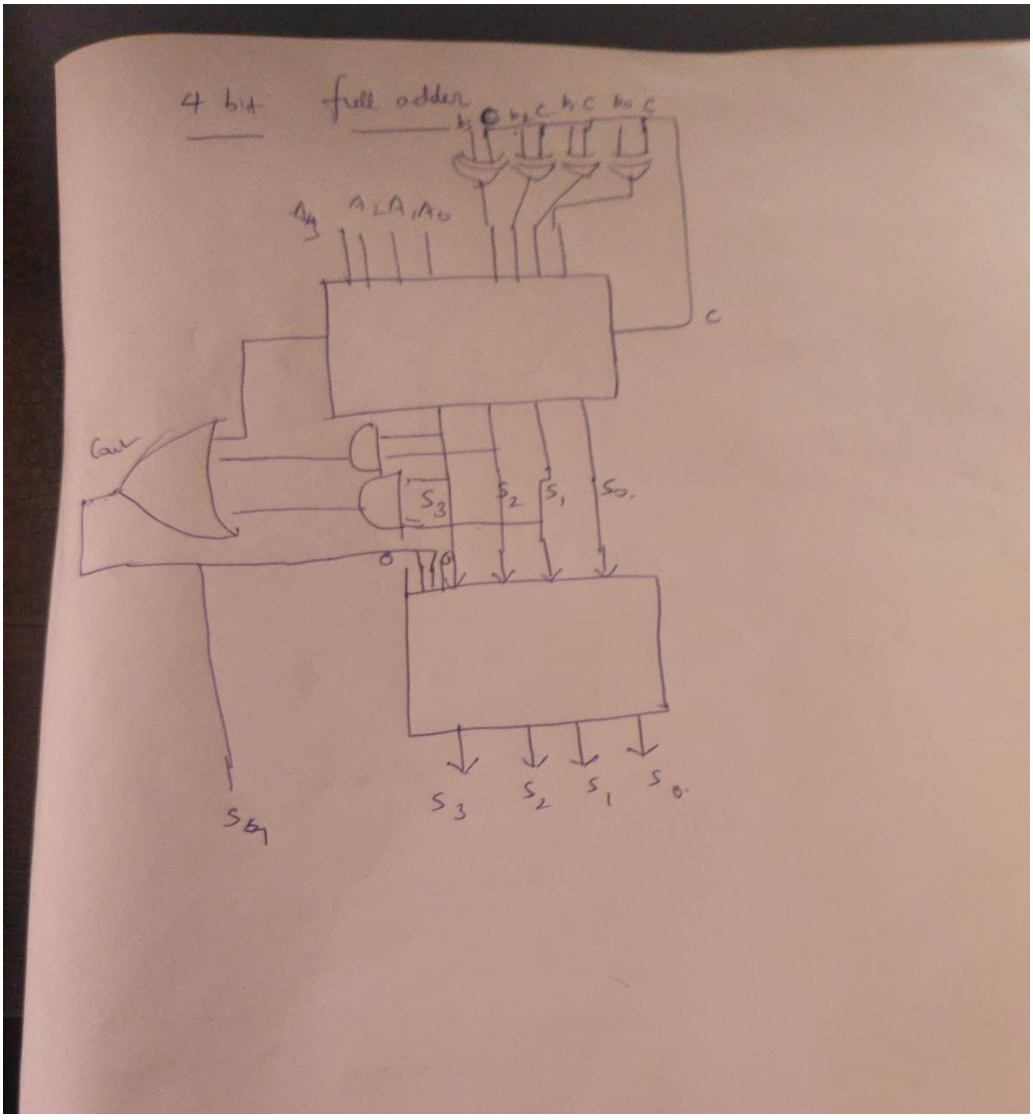
```
if(a>b)
begin
     greater=1;
     equal=0;
     less=0;
end
else if(a==b)
begin
     greater=0;
     equal=1;
     less=0;
end
else
begin
     greater=0;
     equal=0;
     less=1;
end
end
endmodule
```

Test bench:

```
`timescale 1ns/1ns
 `include "c.v"

 module comparator_tb;
 reg [3:0] a;
 reg [3:0] b;

 wire greater;
 wire equal;
 wire less;
 c uut(a,b,greater,equal,less);

 initial begin
     $dumpfile("comp.vcd");
     $dumpvars(0,comparator_tb);
```

```
       a=4'b0001 ; b=4'b0000;

       #5  a=4'b1110; b=4'b1110;

       #5  a=4'b0101; b=4'b0111;

      #5

      $display("test complete");


end

endmodule
```

Q4. 4 bit bcd full adder



We add six whenever then last carry bit of first full adder is one

Code:

```
module fulladder(cout,s,a,b,cin);

output cout,s;
input a,b,cin;
wire t1,t2,t3,t4,t5,t6;
```

```
//sum
xor c1(t6,a,b);
xor c2(s,t6,cin);
//carry
not c3(t1,a);
nand c4(t2,a,b);
not c5(t3,b);
nand c6(t4,t3,t1);
nand c7(t5,t4,cin);
nand c8(cout,t5,t2);

endmodule
```

```
`include"fulladder.v"
module bit(c,s,C,S,a,b,cin);
output [4:0]s;
output [2:0]c;
output [4:0]S;
output [3:0]C;
input  [3:0]a;
input [3:0]b;
input cin;
wire t1,t2,t3,t4,t5,t6;
wire[4:0] bcd;

fulladder c1(c[0],s[0],a[0],b[0],cin);
fulladder c2(c[1],s[1],a[1],b[1],c[0]);
fulladder c3(c[2],s[2],a[2],b[2],c[1]);
fulladder c4(s[4],s[3],a[3],b[3],c[2]);

//converting into bcd
nor c5(t1,s[1],s[2]);
nand c6(t2,1,t1);
nand c7(t3,s[3],t2);
nand c8(t4,1,s[4]);
nand c9(t5,t4,t3);
```

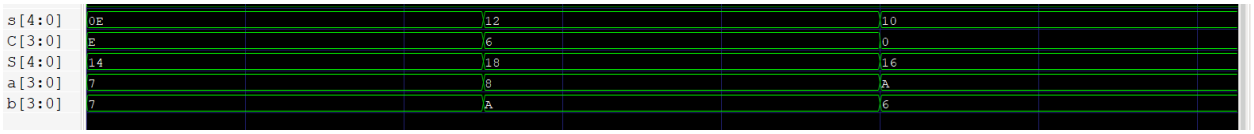
```
fulladder c10(C[0],S[0],s[0],1'b0,cin);
fulladder c11(C[1],S[1],s[1],t5,C[0]);
fulladder c12(C[2],S[2],s[2],t5,C[1]);
fulladder c13(C[3],S[3],s[3],1'b0,C[2]);
not c14(t6,t5);
not c15(S[4],t6);









endmodule
```

Test bench

```
`timescale 1ns/1ns
`include "bit.v"

module bit_tb;
reg [3:0]a;
reg [3:0]b;
wire [4:0]s;
wire [2:0]c;
wire [4:0]S;
wire [3:0]C;
wire [4:0]bcd;
reg cin;

bit uut(c,s,C,S,a,b,cin);
initial begin
    $dumpfile("bit.vcd");
    $dumpvars(0,bit_tb);
```

```
    a=4'b0111;b=4'b0111; cin=1'b0;
    #5 a=4'b1000;b=4'b1010;
    #5 a=4'b1010;b=4'b0110;

    #5


end
endmodule
```

Result:



Q5. TO PERFORM SUBTRCTION OPERATION ON TWO SINGLE DIGIT DECIMAL NUMBER WHERE A>B AND REPRESENTED IN BINARY FORM USING A SUBTRACTION SCHEME

STRUCTURAL LEVEL VERILOG CODE

```
module bcd_sub(S,C,A,B);
  output [4:0] S;
  output   C;
  input [4:0]    A;
  input [4:0]    B;

  wire  C0;
  wire  C1;
  wire  C2;
  wire  C3;

  wire  D0;
  wire  D1;
  wire  D2;
  wire  D3;
// the subtraction is done using 2's complement method

  xor(D0, B[0], B[4]);
  xor(D1, B[1], B[4]);
  xor(D2, B[2], B[4]);
  xor(D3, B[3], B[4]);
  xor(C, C3, B[4]);
   // here, if the MSB bit of B is 1 then it will perform the subtraction A-B, where A>B
   // here, if the MSB bit of B is 0, then it will perform the addition A+B

  full_adder fa0(S[0], C0, A[0], D0, B[4]);
  full_adder fa1(S[1], C1, A[1], D1, C0);
```

```
  full_adder fa2(S[2], C2, A[2], D2, C1);
  full_adder fa3(S[3], C3, A[3], D3, C2);
  assign S[4]=C; // since A-B is always positive

endmodule

module full_adder(S, Cout, A, B, Cin);
  output S;
  output Cout;
  input  A;
  input  B;
  input  Cin;

  wire   w1;
  wire   w2;
  wire   w3;


  xor(w1, A, B);
  xor(S, Cin, w1);
  nand(w2, A, B);
  nand(w3, w1, Cin);
  nand(Cout, w2, w3);
endmodule
```

TESTBENCH FOR THE ABOVE CODE

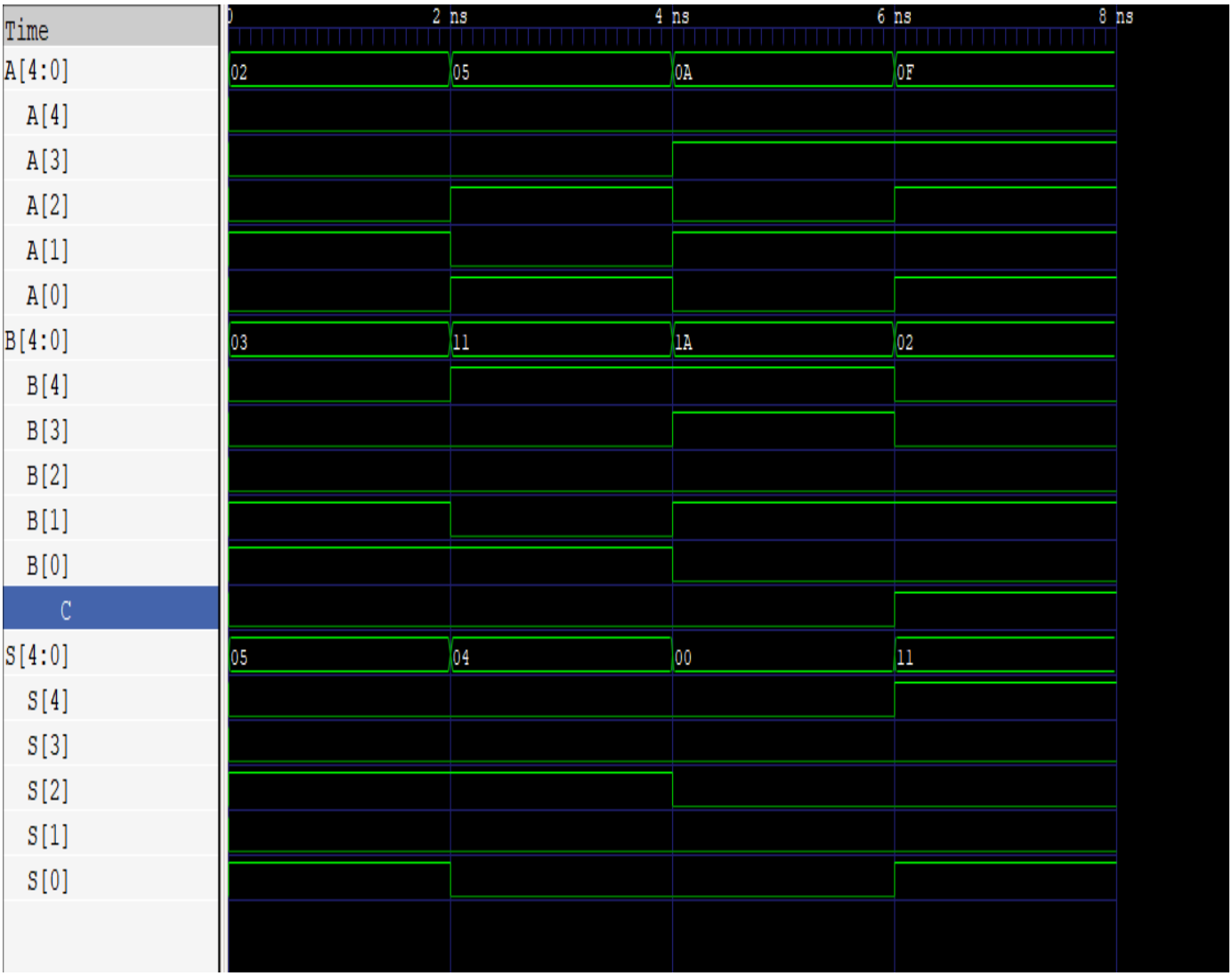
```
`timescale 1ns/1ns
`include"bcd_sub.v"

module tb_bcd_sub;
reg [4:0] A;
reg [4:0] B;
wire [4:0] S;
wire C;
```

```
bcd_sub uut(S,C,A,B);
initial
begin
   $dumpfile("tb_bcd_sub.vcd");
   $dumpvars(0,tb_bcd_sub);
   A=5'b00010;
   B=5'b00011;
    // A=2 B=3 MSB(B)=0 i.e A+B=5
   #2
   A=5'b00101;
   B=5'b10001;
   //A=5 B=1 MSB(B)=1 i.e A-B=4
   #2
   A=5'b01010;
   B=5'b11010;
    // A=10 B=10 MSB(B)=1 i.e A-B=0
   #2
   A=5'b01111;
   B=5'b00010;
    // A=15 B=2 MSB(B)=0 i.e A+B=17


   #2 $finish;
  end
 initial
$monitor($time,"S=%b,C=%b",S,C);
Endmodule
```

OUTPUT IN TERMINAL

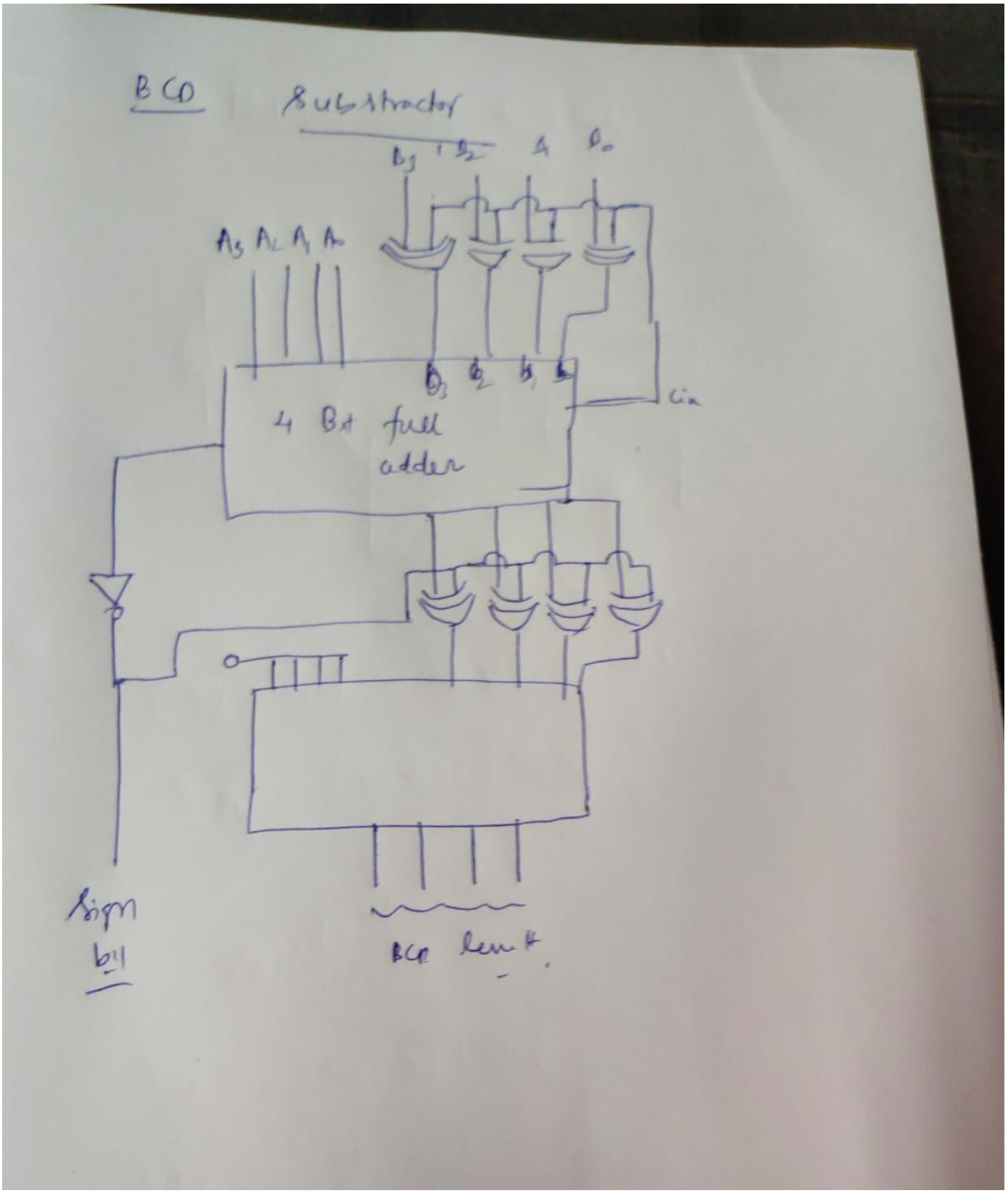
```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

tb_dec_bcd.v:48: $finish called at 6 (1ns)

C:\iverilog\bin>iverilog -o tb_bcd_sub.vvp tb_bcd_sub.v

C:\iverilog\bin>vvp tb_bcd_sub.vvp
VCD info: dumpfile tb_bcd_sub.vcd opened for output.
                   0S=00101,C=0
                   2S=00100,C=0
                   4S=00000,C=0
                   6S=10001,C=1
tb_bcd_sub.v:32: $finish called at 8 (1ns)

C:\iverilog\bin>
```

OUTPUT IN GTK WAVE



CASE -2

BCD SUBTRACTION



VERILOG CODE-

```
`include "dec_bcd.v"
`include "fulladder.v"

module bcd(Ca,Su,a,b,cin);
output [3:0]Ca;
output [4:0]Su;
input  [3:0]a;
input  [3:0]b;
input cin;

wire [3:0]s;
wire [3:0]c;
wire t0,t1,t2,t3,t4,t5;
wire [3:0]e;
xor g1(t0,b[0],cin);
xor g2(t1,b[1],cin);
xor g3(t2,b[2],cin);
xor g4(t3,b[3],cin);
fulladder c1(c[0],s[0],a[0],t0,cin);
fulladder c2(c[1],s[1],a[1],t1,c[0]);
fulladder c3(c[2],s[2],a[2],t2,c[1]);
fulladder c4(c[3],s[3],a[3],t3,c[2]);

not g5(t4,c[3]);
xor g6(e[0],s[0],t4);
xor g7(e[1],s[1],t4);
xor g8(e[2],s[2],t4);
xor g9(e[3],s[3],t4);

fulladder c5(Ca[0],Su[0],1'b0,e[0],t4);
fulladder c6(Ca[1],Su[1],1'b0,e[1],C[0]);
fulladder c7(Ca[2],Su[2],1'b0,e[2],C[1]);
fulladder c8(Ca[3],Su[3],1'b0,e[3],C[2]);
```

```
not g10(t5,t4);
not g11(S[4],t5);

endmodule
```

TESTBENCH CODE –

```
`timescale 1ns/1ns
`include "bcd.v"

module bcd_tb;

reg cin;

wire [3:0]Ca;
wire [4:0]Su;
reg [3:0]a;
reg [3:0]b;



bcd uut(Ca,Su,a,b,cin);
initial begin
   $dumpfile("bcd1.vcd");
   $dumpvars(0,bcd_tb);

   a=4'b1000;b=4'b0100;cin=1'b1;
   #5 a=4'b0001;b=4'b1000;



   #5 ;

end
endmodule
```

RESULT-