I) PRE-LAB →

For Obj. 1 → Design a 2 bit Comparator circuit.

a) Truth table →

Input				Output		
A_1	A_0	B_1	B_0	$A>B$	$A<B$	$A=B$
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

b) Minimised Boolean expression →

i) For A>B →

A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

ii) For A<B →

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

iii) For A=B →

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

$A>B = A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$

$A<B = A_1'B_1 + A_1'B_1B_0 + A_1'A_0'B_0$

$(A=B) = (A_0 \odot B_0)(A_1 \odot B_1)$

For A=B →

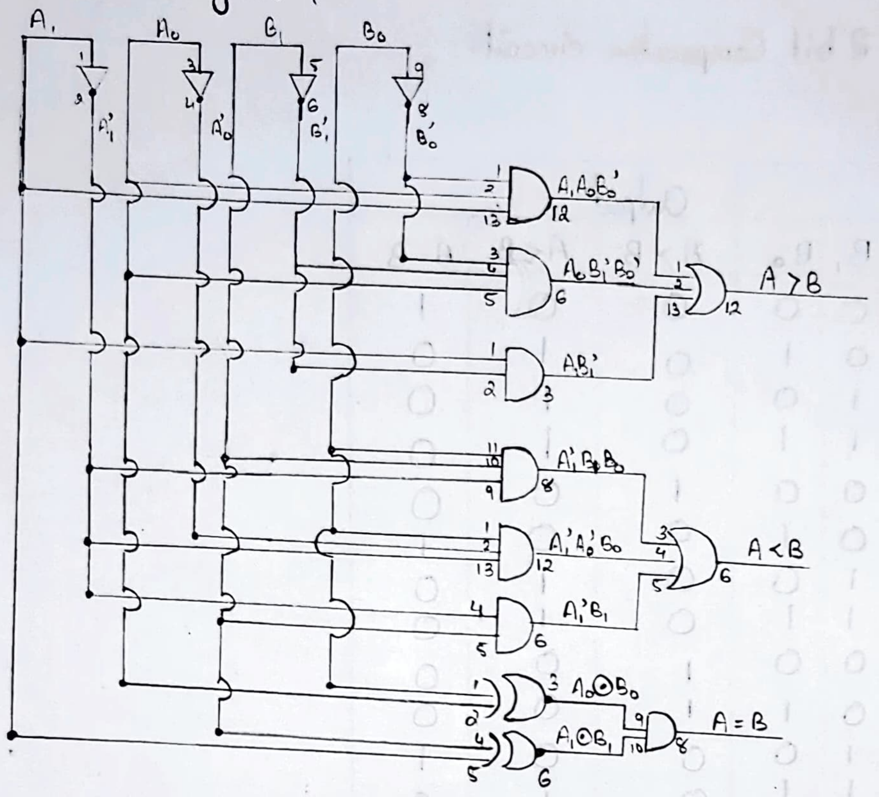
$$F = A_1'A_0'B_1'B_0' + A_1'A_0B_1'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1B_0'$$

$$= A_0'B_0'(A_1'B_1' + A_1B_1) + A_0B_0(A_1'B_1' + A_1B_1)$$

$$= (A_0B_0 + A_0'B_0')(A_1'B_1' + A_1B_1)$$

$$= (A_0 \odot B_0)(A_1 \odot B_0)$$

c) Circuit Diagram →



d) HDL code →

```
library ieee;
 use ieee.std_logic.1164.all;
 entity test1 is
     port (A1, A0, B1; B0 : in std_logic;
              F1, F2, F3 : out std_logic);
 end test 1;
 architecture boolean_eq of test 1 is
     signal A, B, C, D : std_logic.
     signal a, b, c, d, e, f, g, h : std_logic.
      A<=NOT A1;
      B<=NOT A0;
      C<=NOT B1;
      D<=NOT B0;
     a<=A1 AND A0 AND D;
     b<=A0AND C AND D;
    c<= A1 AND C;
    d<= A AND B1 AND B0;
    e<= A AND B AND B0;
    f<= A AND B1;
    g<= A0 EX-NOR B0;
    h<= A1 EX-NOR B0;
                        F1 = a OR b OR c;
                        F2 = d OR e OR f;
                        F3 = g AND K
                      end boolean_eq.
```

For Obj. 2 → Combinational circuit that converts 4 bit binary number to equivalent Gray code.

a) Truth Table →

Inputs				Outputs			
A	B	C	D	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

b) Minimised Boolean Expression →

i) For G_3 →

AB\CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$G_3 = A$

ii) For G_2 →

AB\CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_2 = A'B + AB'$
$= A \oplus B$

iii) For G_1 →

AB\CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

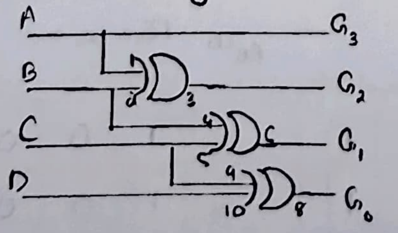
$G_1 = B'C' + B'C$
$= B \oplus C$

iv) For G_0 →

AB\CD	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$G_0 = C'D + C'D$
$= C \oplus D$

c) Circuit Diagram →



d) HDL code →

```
library ieee;
use ieee.std_logic.1164.all;
entity test2 is
        port (A, B, C, D : in std_logic;
              G3, G2, G1, G0 : out std_logic);
    end test2;
  architecture boolean_eq of test2 is
    begin
         G3 <= A;
         G2 <= A XOR B;
         G1 <= B XOR C;
         G0 <= C XOR D;
    end boolean_eq;
```

For Obj. 3 → Design a combinational circuit with four input lines that represents a decimal digit in BCD and four output lines that generates the 9's complement of the input digit.

a) Truth table →

Inputs				Outputs			
A	B	C	D	w	x	y	z
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0

b) Minimesed Boolean Expression →

For w →

AB \ CD	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

$w = A'B'C'$

For x →

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

$x = BC' + B'C = B \oplus C$

For y →

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	X	X	X	X
10	0	0	X	X

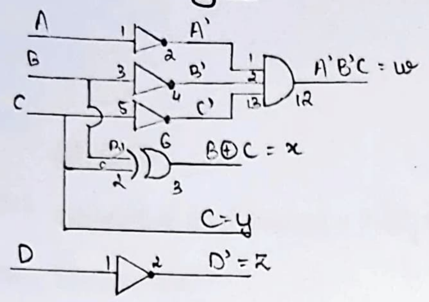
$y = C$

For z →

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$z = D'$

c) Circuit Diagram →



d) HDL code →

```
library ieee;
use ieee.std_logic.1164.all;
entity test3 is
        port CA, B, C, D : in std_logic;
                W, X, Y, Z : out std_logic;)
end test3;
 architecture boolean_eq of test3 is
       Signal  a, b, c : std_logic;
 Begin
        a<= NOT A;
        b<= NOT B;
        c<= NOT C;
       w <= a AND b AND c;
       X <= B XOR C;
       Y<= C;
      Z<= NOT D;
end  boolean_eq;
```

IV> POST LAB →

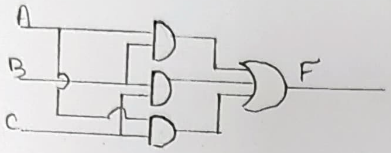
1> Design a 3 bit majority circuit.

Ans →

Inputs			Output
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A \ BC	00	01	11	10
0			1	
1		1	1	1

F = AB + BC + AC

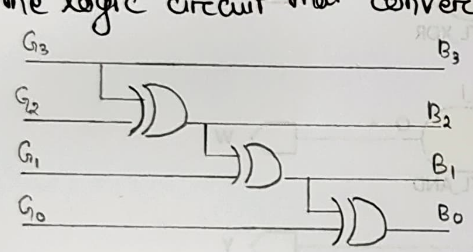


2> What is the advantage of Gray code?

Ans → The advantage of Gray code is that just one bit changes for each step. This will come in handy in circuits that are sensitive to glitches and other errors.

3> Draw the logic circuit that converts 4 bit Gray code to binary code.

Ans →



III> LAB →

Conclusion →

Obj 1 → It can be ~~sl~~ concluded that for a 2 bit Comparator circuit we need 7 and gates, 2 or gates and 2 ex-nor gates to get A<B, A>B & A=B.

Obj 2 → It can be concluded that ~~for~~ to convert a 4-bit binary to equivalent gray code we need 3 ex-or gates.

Obj 3 → It can be concluded that for a 9's complement of a BCD number we need 4 not gates, 1 ex-or gate and 1 and gate.