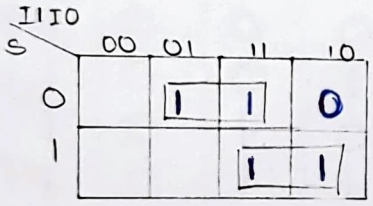
I) PRE-LAB →

For Obj. 1 → Design a 2×1 Multiplexer that will select the binary information from one of the two inputs lines and direct it to a single output line based on the value of a selection line.

a) Truth table →

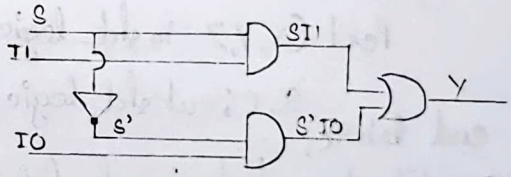
Inputs			Outputs
S	I1	I0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

b) Minimised Boolean Function →



$Y = S'I0 + SI1$

c) Circuit Diagram →



d) HDL Code →

```
library ieee;
use ieee.std_logic_1164.all;
entity test1 is
        port(I1, I0, S0 : in std_logic;
                Y: out std_logic);
end test1;
architecture behavioral of test1 is
        signal a,b, c : std_logic;
  begin
     a <= not S0;
     b <= S0 and I1;
     c <= a and I0;
     Y <= b or c;
  end behavioral;
```

For Obj 3→ Design a full adder using 3 to 8 line decoder and external OR gates.

a) Truth table →

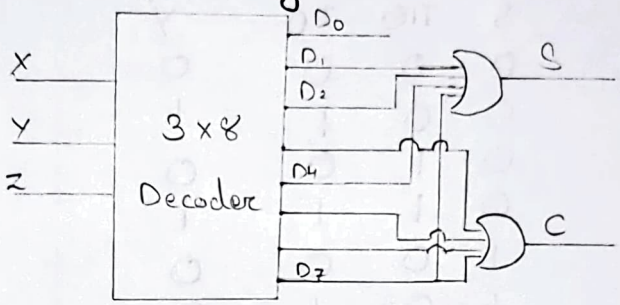
Inputs			Outputs	
X	Y	Z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

b) Minimised Boolean Function →

$S = X'Y'Z + X'YZ' + XY'Z' + XYZ$

$C = X'YZ + XY'Z + XYZ' + XYZ$

c) Circuit Diagram →



d) HDL code →

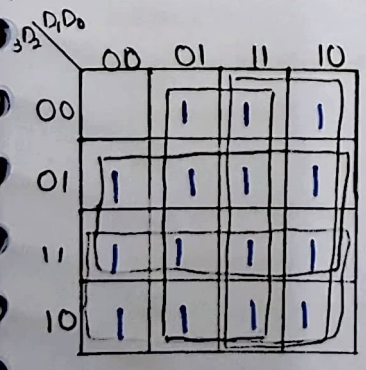
```
library ieee;
use ieee.std_logic_1164.all;
entity lab9c is
    Port (X, Y, Z : in std_logic;
          S, C : out std_logic);
end lab9c;
architecture behavioral of lab9c is
    signal a, b, d, e, f, g, h, i, j, k, l : std_logic;
begin
    a <= not X;
    b <= not Y;
    d <= not Z;
    e <= a and b and d;
    f <= a and b and Z;
    g <= a and Y and d;
    h <= a and X and Z;
    i <= X and b and d;
    j <= X and b and Z;
    k <= X and Y and d;
    l <= X and Y and Z;
    S <= f or g or i or l;
    C <= h or j or k or l;
end behavioral;
```

For Obj. 2 → Design a 4 bit priority encoder with inputs D3 (MSB), D2, D1, & D0 (LSB) and outputs X, Y, & V. The priority assigned to inputs is D3 > D2 > D1 > D0. The output V shows a value 1 when one or more inputs are equal to one. If all inputs are 0, V is equal to 0. When V=0, then other two outputs are not inspected and are specified as don't care conditions.

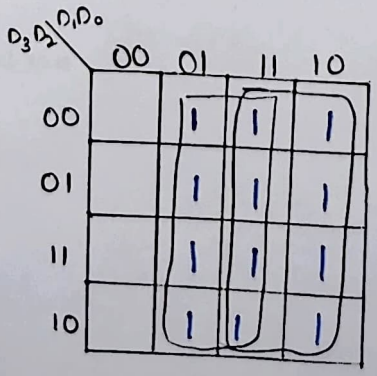
a) Truth Table →

Inputs				Outputs		
D_3	D_2	D_1	D_0	X	Y	V
0	0	0	0	X	X	0
0	0	0	1	1	1	1
0	0	1	0	1	0	1
0	0	1	1	1	1	1
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

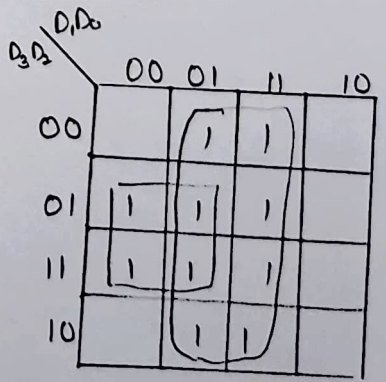
b) Minimised Boolean Expression →



$V = D0 + D1 + D2 + D3$

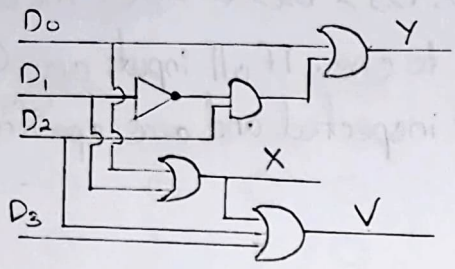


$X = D3 + D1$



$Y = D_2 D_1' + D_0$

c) Circuit Diagram →



d) HDL code →

```
library ieee;
use ieee.std_logic_1164.all;
entity lab9b is
    Port (D3, D2, D1, D0 : in std_logic;
          X, Y, V : out std_logic);
end lab9b;
architecture behavioral of lab9b is
    Signal a, b : std_logic;
begin
    a <= not D1;
    b <= a and D2;
    X <= D0 or D1;
    Y <= D0 or b;
    V <= D0 or D1 or D2 or D3;
end behavioral;
```

IV) POST LAB →

Q1) Why is a multiplexer known as data selector?

Ans → The selection of a particular input data line for the output is decided on the basis of selection lines. The multiplexer is often called as data selector since it selects only one of many data inputs.

Q2) Implement a full adder using two 4x1 multiplexers.

Ans →

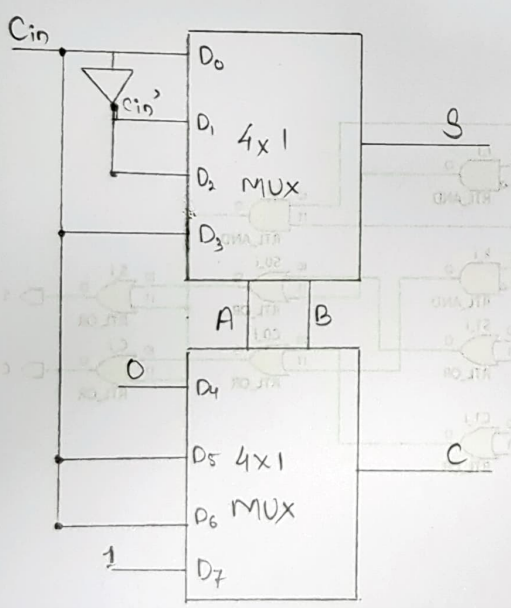
Inputs			Outputs	
A	B	C_{in}	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

→ S = C_{in}, C = 0

→ S = C_{in}', C = C_{in}

→ S = C_{in}', C = C_{in}

→ S = C_{in}, C = 1



III) LAB →

Obj 1 → It can be concluded that for 2x1 multiplexer we need 2 and gates, 1 not gate and 1 or gate.

Obj 2 → It can be concluded that for 4 bit encoder we need 1 not gate, 0 and gate & 3 or gates.

Obj 3 → It can be concluded that for a 3 to 8 line decoder we need one 3 to 8 decoder and 2 or gates.