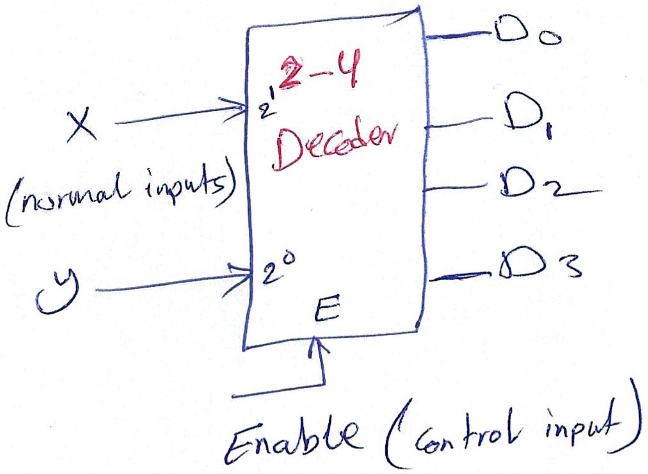
*Decoder with Enable(E) :-

E → high lever :- If Enable is zero then all outputs are zero regarless of the inputs (x,y)
If Enable is one the the outputs are the minterms

→ low level :- If Enable is one all the outputs are zero regarless of the inputs.



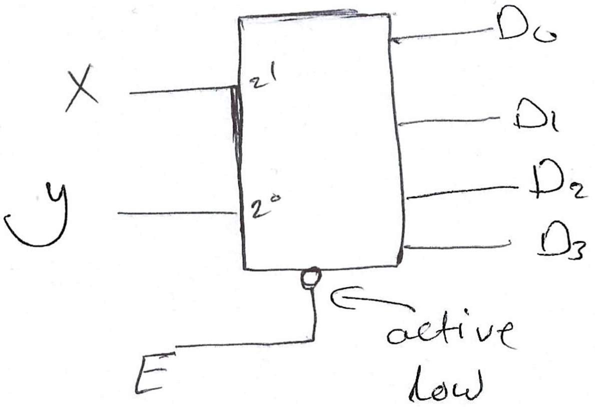
E	x	y	D_0	D_1	D_2	D_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

active high enable

⇓

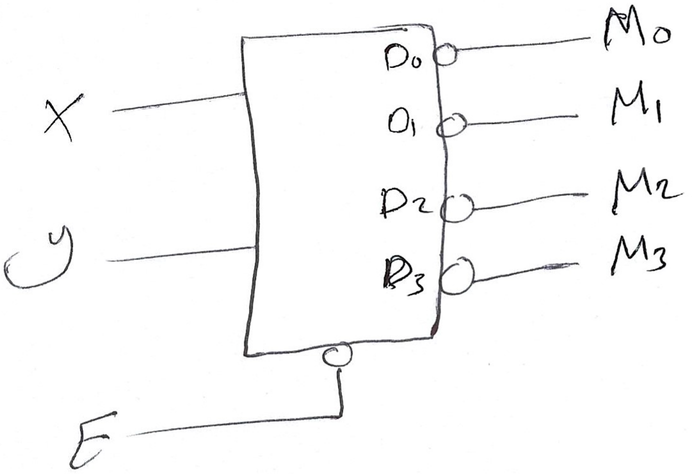
E	x	y	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Example :- Design 2X4 active high decoder with active low enable :-



E	x	y	D_0	D_1	D_2	D_3
1	X	X	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

Example :- Design a 2X4 active low decoder active low Enable



E	X	y	M_0	M_1	M_2	M_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

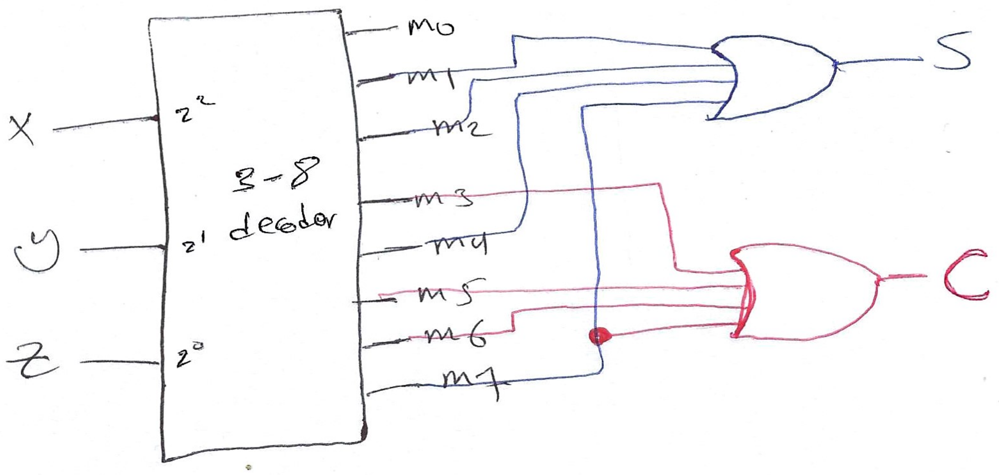
Example :- Implement the full adder using decoder

$S = \Sigma(1,2,4,7)$

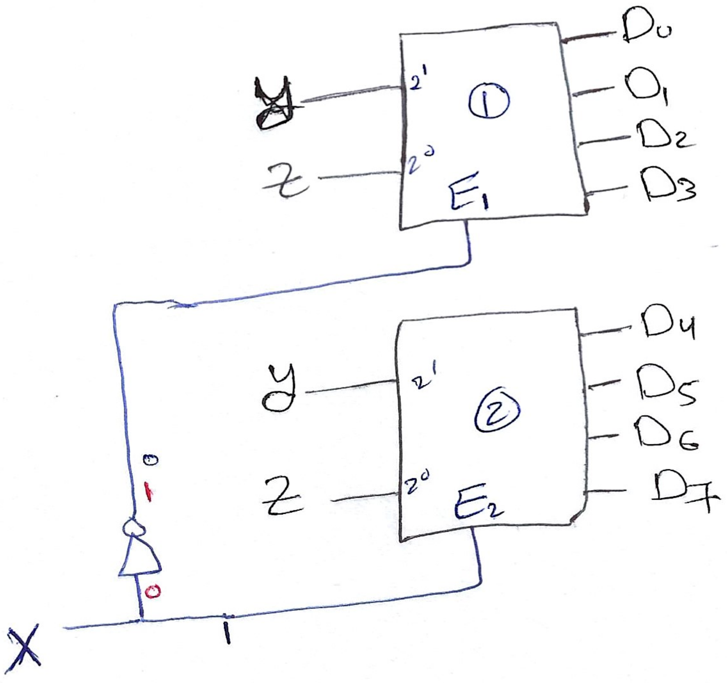
$C = \Sigma(3,5,6,7)$

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

we Can design the full adder using 3-8 decoder



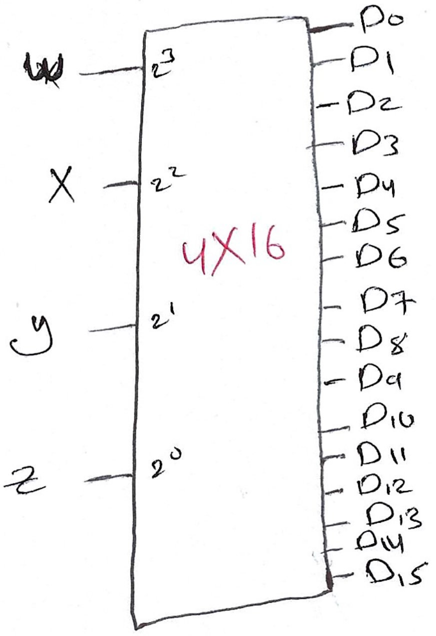
we Can implement the 3-8 decoder using 2-4 decoder using the enable input



X	y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

If X=0 then E_1 will be 1 and Decoder 1 will work and if X=1 then E_2 will be 1 and Decoder 2 works

Examples Design a 4X16 Decoder using 2x4 decoders



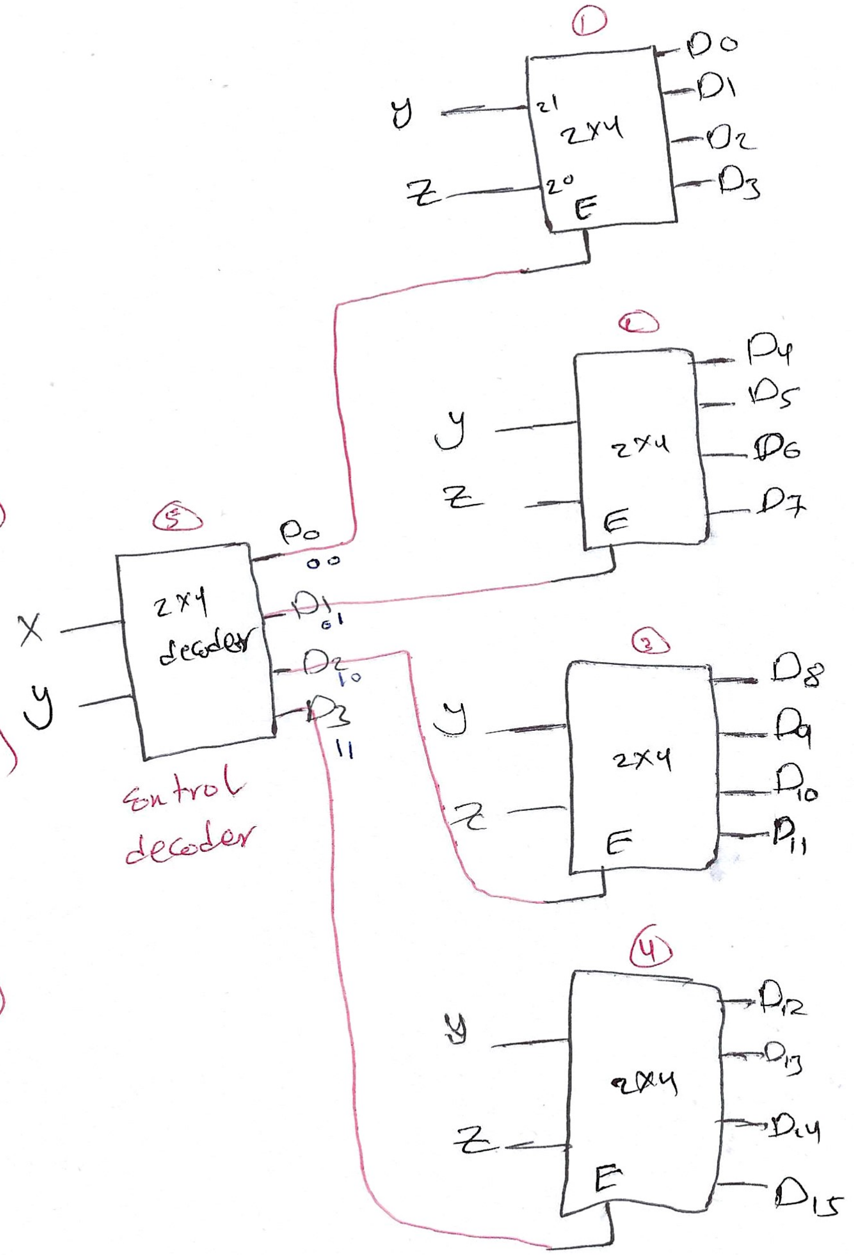
to implement 4 X 16 using 2 x4 decoders we need

$$\frac{16}{4} = 4 \text{ decoder } (2x4)$$

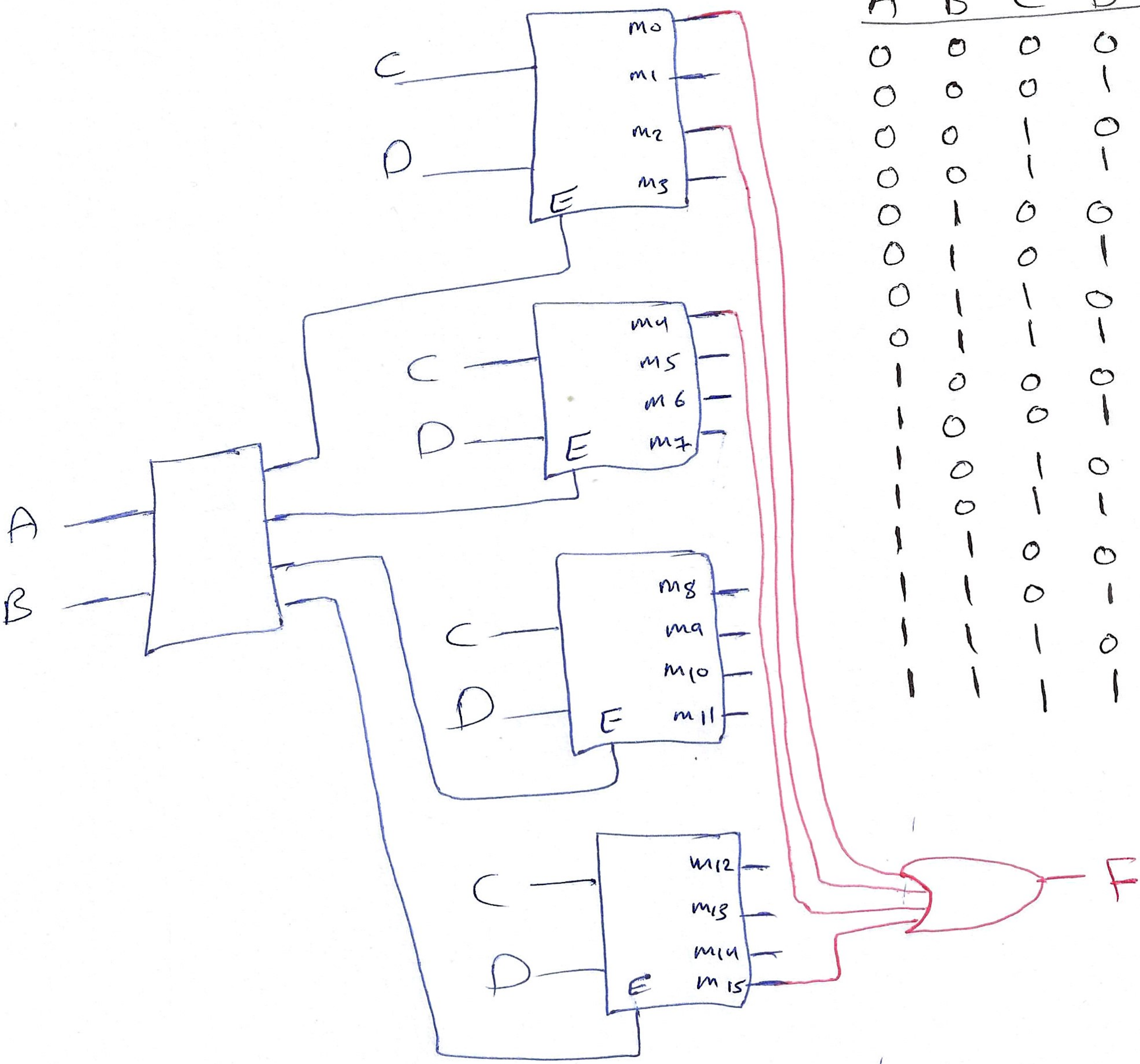
at least

we need 5

w	x	y	z	
0	0	0	0	Decoder(1)
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	Decoder(2)
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	Decoder(3)
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	Decoder(4)
1	1	0	1	
1	1	1	0	
1	1	1	1	

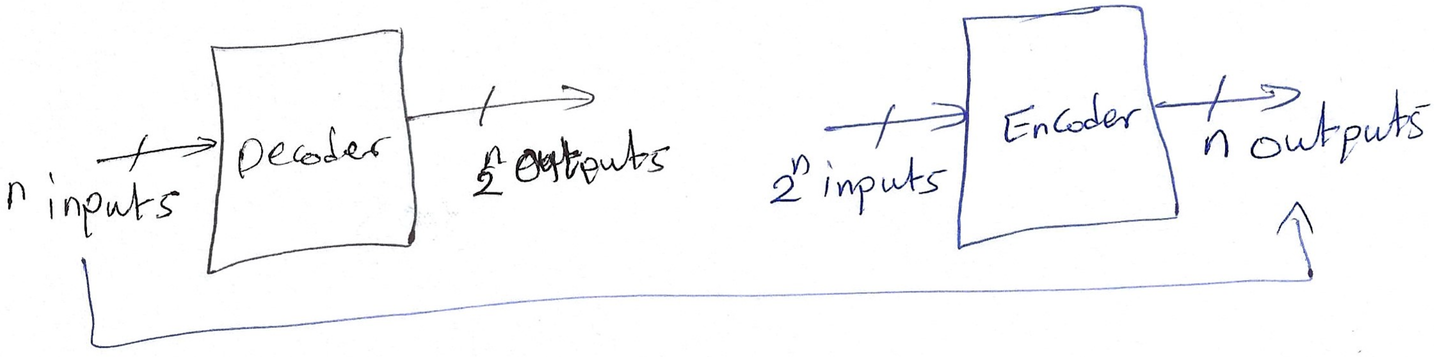


Example:- Implement the following fuction using 2x4 decoder. $F(A,B,C,D) = \sum(0,2,4,15)$

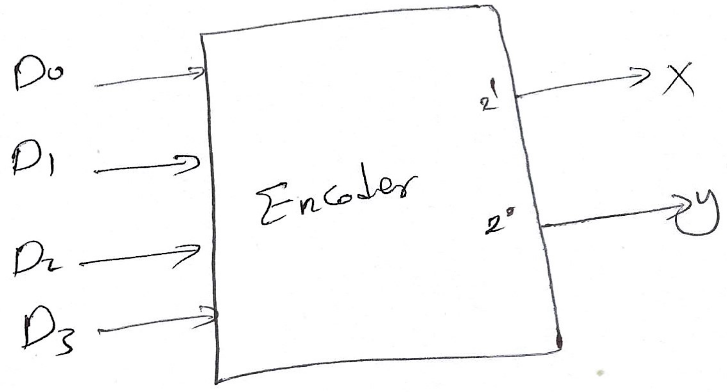


A	B	C	D	F	
0	0	0	0	1	m0
0	0	0	1	0	
0	0	1	0	1	m1
0	0	1	1	0	
0	1	0	0	1	m3
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	m15

* Encoder:- Inverse operation of decoder



$$\underbrace{4}_{\text{inputs}} \; X \underbrace{2}_{\text{outputs}} \quad \text{Encoder}$$



D0	D1	D2	D3	X	y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

from the truth table

$$X = D_2 + D_3$$
$$y = D_1 + D_3$$

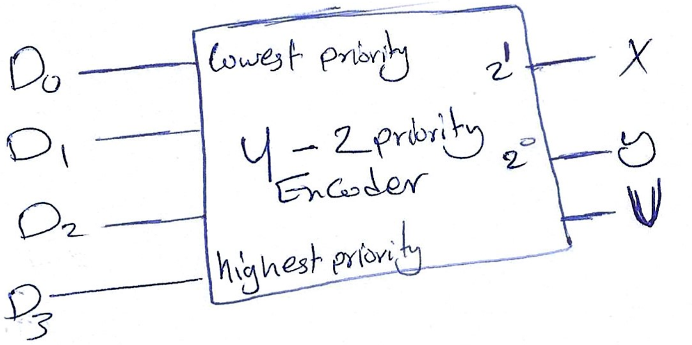
This type of Encoder has two problems(limitations)

① if all inputs($D_0 - D_3$) are Zero

② if more than one input is 1

Thus the term " priority encoder " is introduced with (V) valid output

* priority Encoder

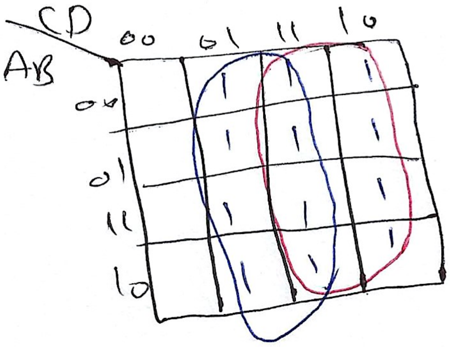


D0	D1	D2	D3	X	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
~~0~~	1	0	0	0	1	1
~~0~~	~~0~~	1	0	1	0	1
~~0~~	~~0~~	~~0~~	1	1	1	1

If $D_2 = 1$ the output is (1 0) regardless of the Values of D_1 and D_0.

If $D_3 = 1$ the output is (1 1) regardless of the Values of D_2, D_1 and D_0

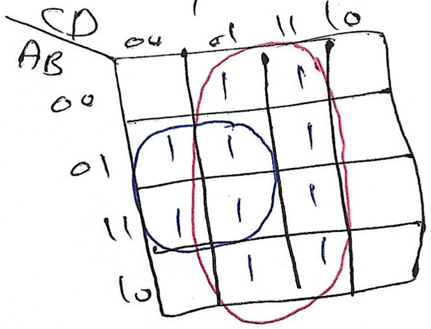
X



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

$X = D_2 + D_3$

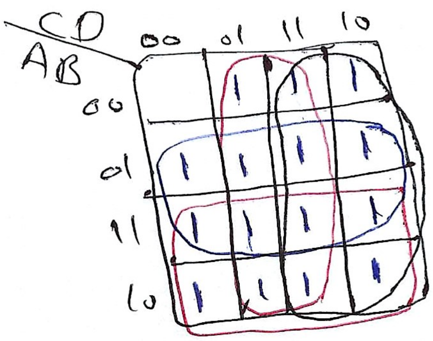
Y



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

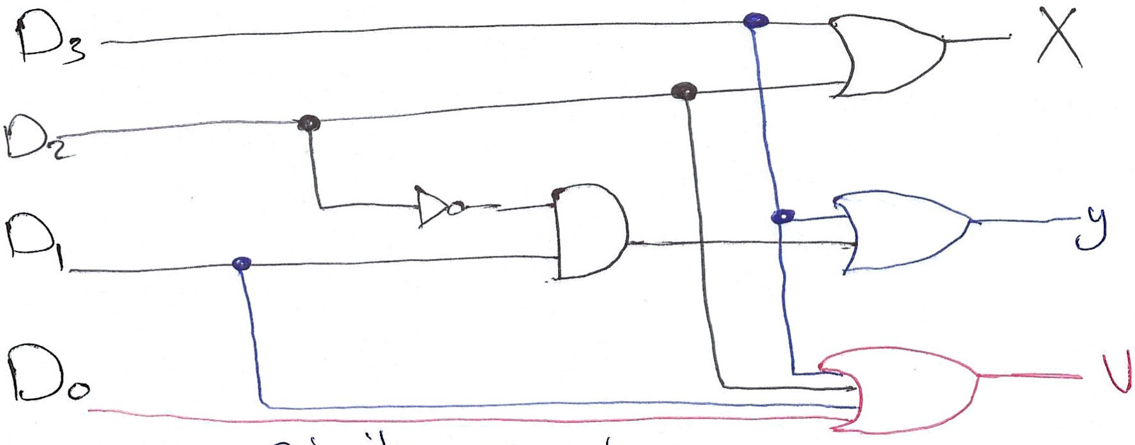
$y = D_3 + D_1 \bar{D}_2$

V



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

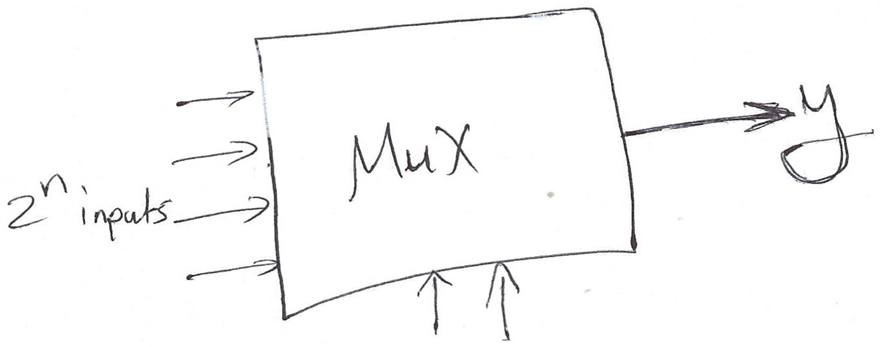
$V = D_0 + D_1 + D_2 + D_3$



Priority Encoder

* Multiplexer (Mux) :- A Combination Circuit that

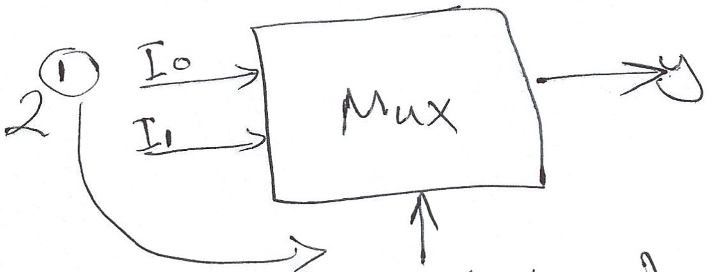
- Multiple data inputs (2^n) to select from
- An n-bit select inputs (S) used for Control
- one output (y)



selection (control inputs)

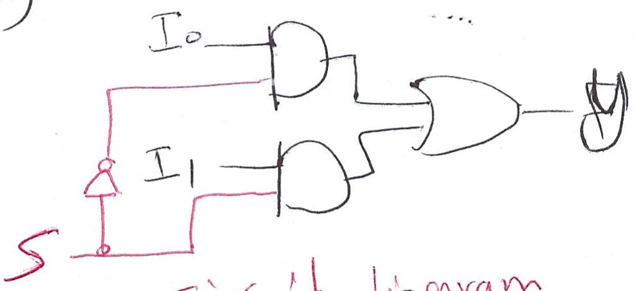
Just one of the data inputs directed to the output based on the Value of S

Example:- Design a 2 x 1 mux



S_0 (selection)

block diagram



Circuit diagram

if S == 0

F = I_0

else S == 1

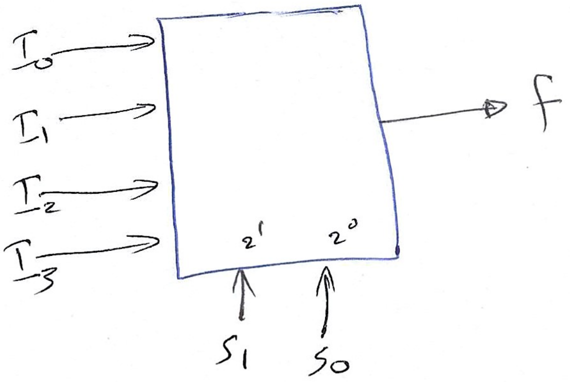
F = I_1

Truth table

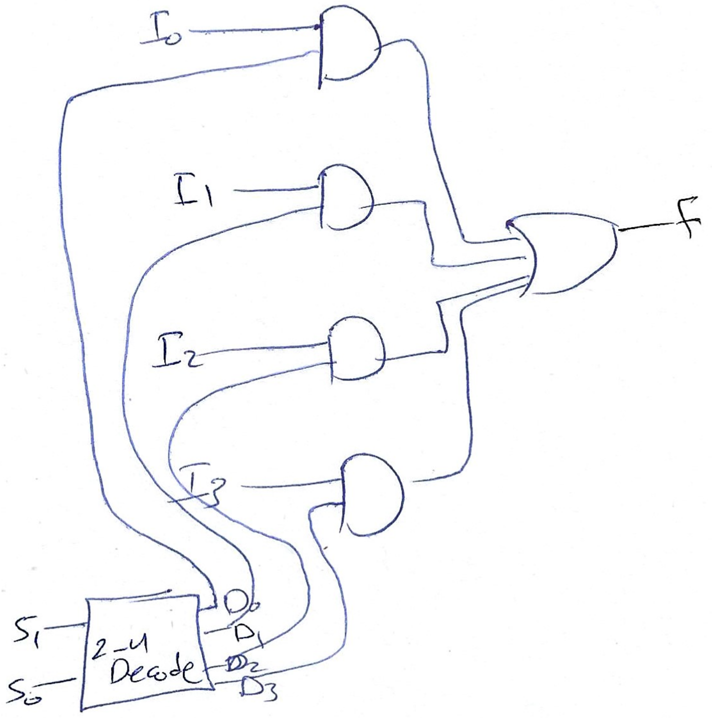
s	d0	d1	y
0	0	X	d0 = 0
0	1	X	d0 = 1
1	X	0	d1 = 0
1	X	1	d1 = 1

Example: 4 X 1 MuX

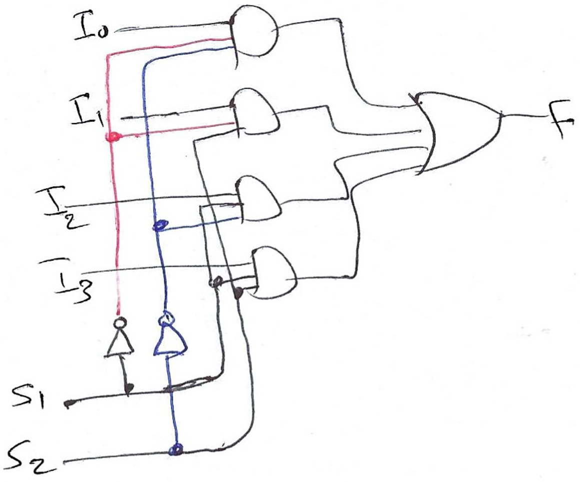
$2^{2} \rightarrow S$



```
If S1S0 == 00
    F = I0
else if S1S0 == 01
    F = I1
elseif S1S0 = 10
    F = I2
else F = I3
```

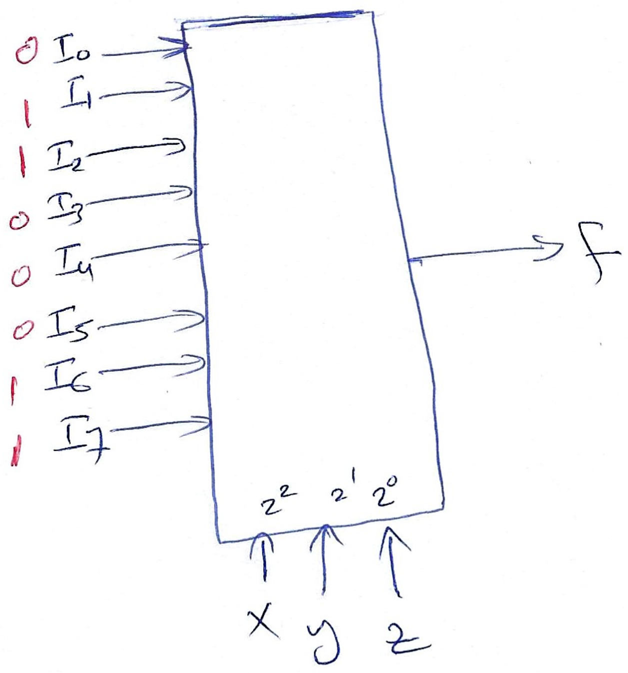


or



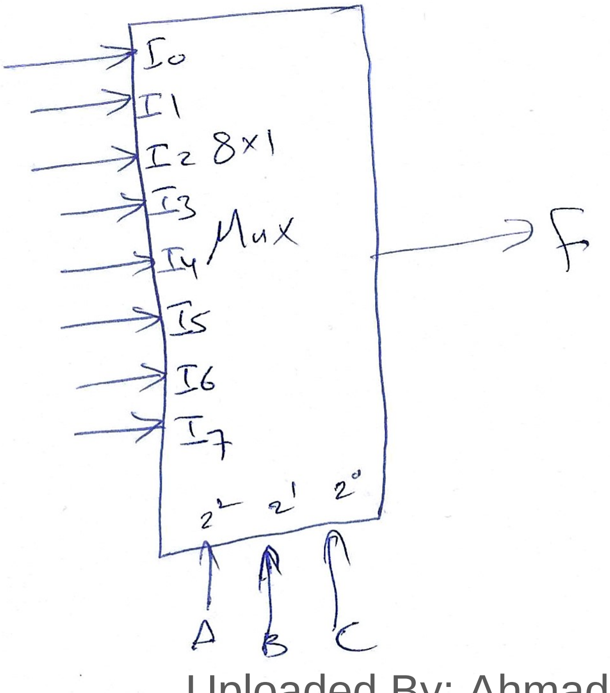
Example: implement the following Function using Mux. $F(x,y,z) = \sum(1,2,6,7)$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

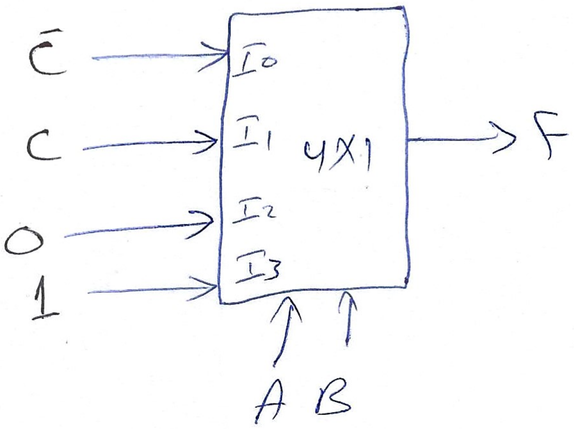


Example: implement the following function using 8x1 and 4x1 Mux. $F(A,B,C)=$

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

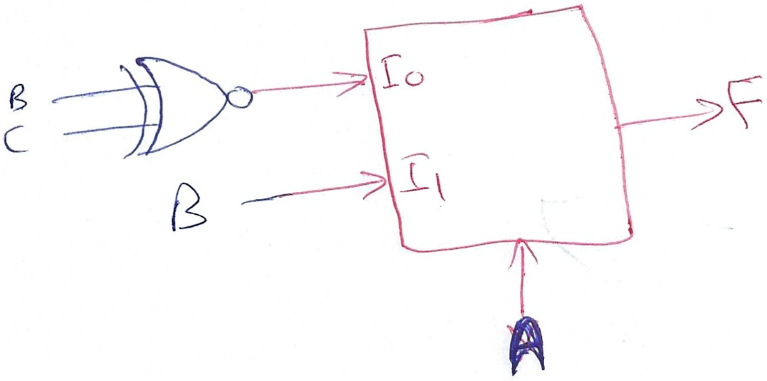


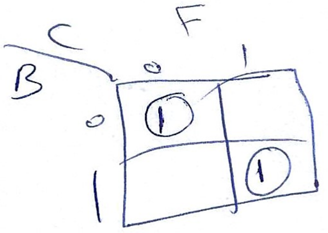
A	B	C		
0	0	0	1	$F=\bar{C}$
0	0	1	0	
0	1	0	0	$F=C$
0	1	1	1	
1	0	0	0	$F=0$
1	0	1	0	
1	1	0	1	$F=1$
1	1	1	1	



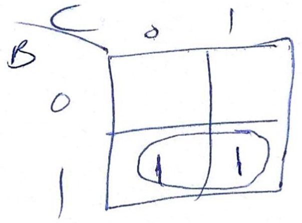
Example: Implement $F(A,B,C)=\sum(0,3,6,7)$ using 2X1 MUX.

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1





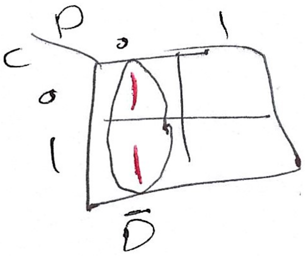
$F=\overline{(A\oplus B)}$

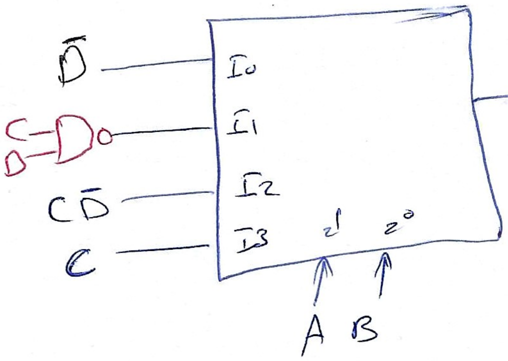


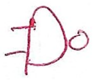
$F=B$

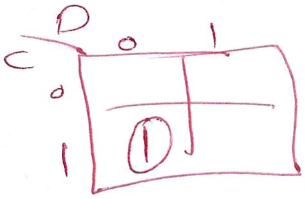
Example:- Implement the $F(A,B,C,D) = \Sigma(0,2,4,5,6,10,14,15)$ using 4X1 Mux.

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



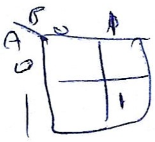






CD=11

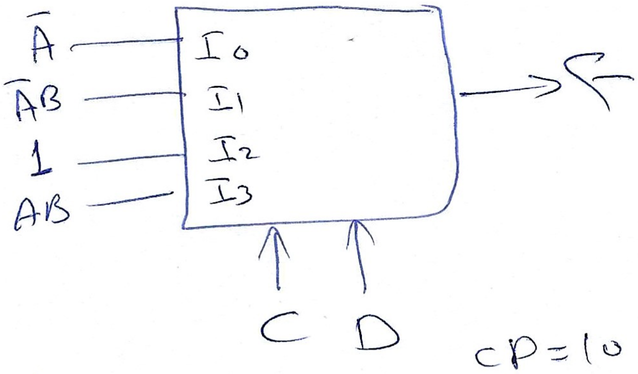
A B	F
0 0	0
0 1	0
1 0	0
1 1	1



CD=00

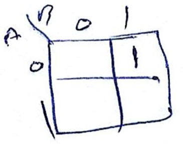
A B	F
0 0	1
0 1	1
1 0	0
1 1	0

$F = \bar{A}$



CD=01

A B	F
0 0	0
0 1	1
1 0	0
1 1	0



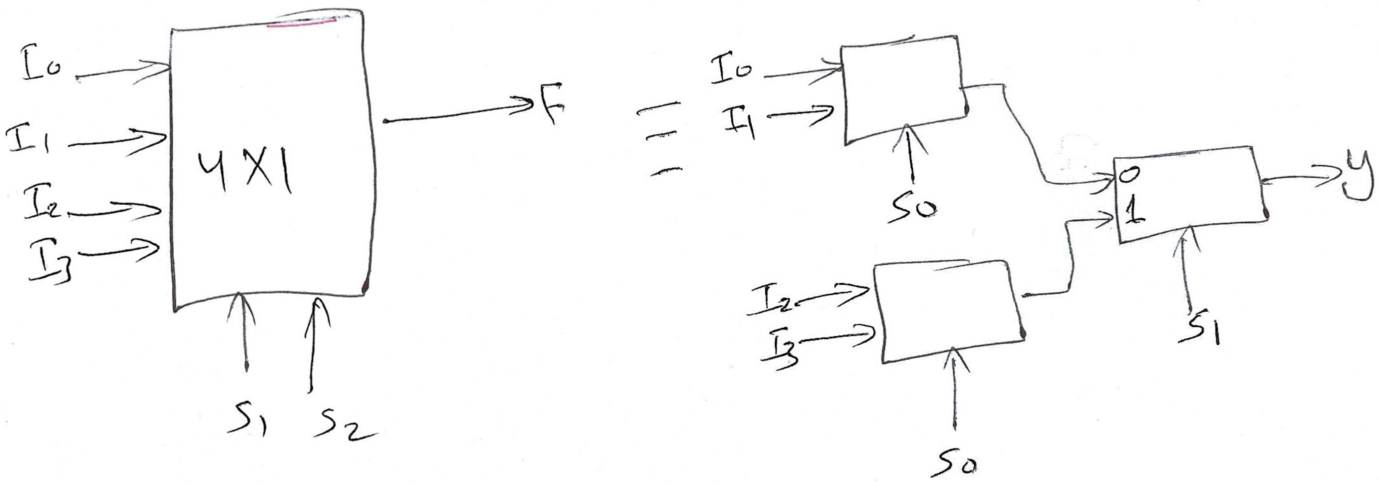
$F = \bar{A}B$

CD=10

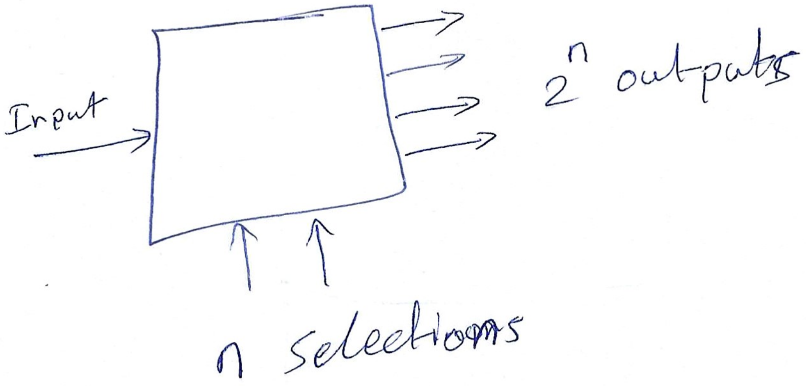
A B	F
0 0	1
0 1	1
1 0	1

F=1

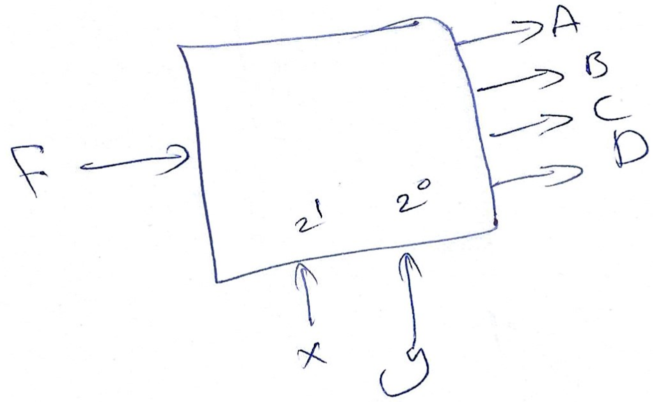
* Building larger Multiplexer



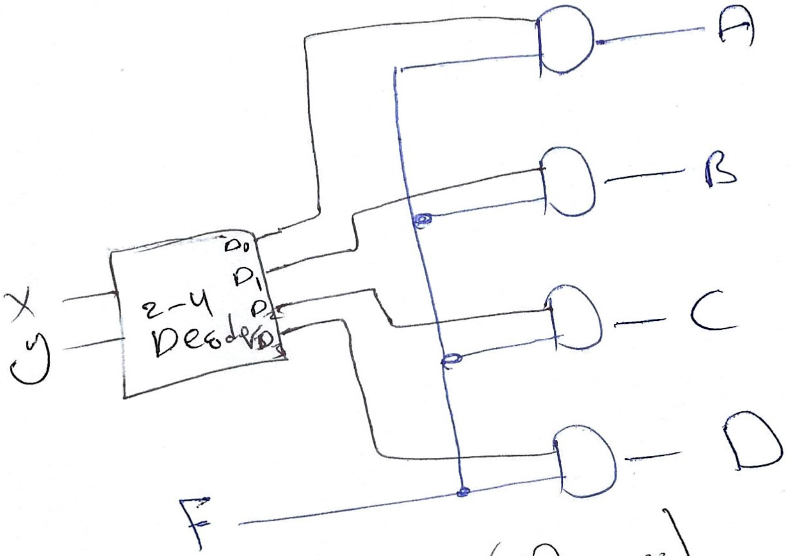
* Demultiplexer (Demux)



Demux 1 X 4

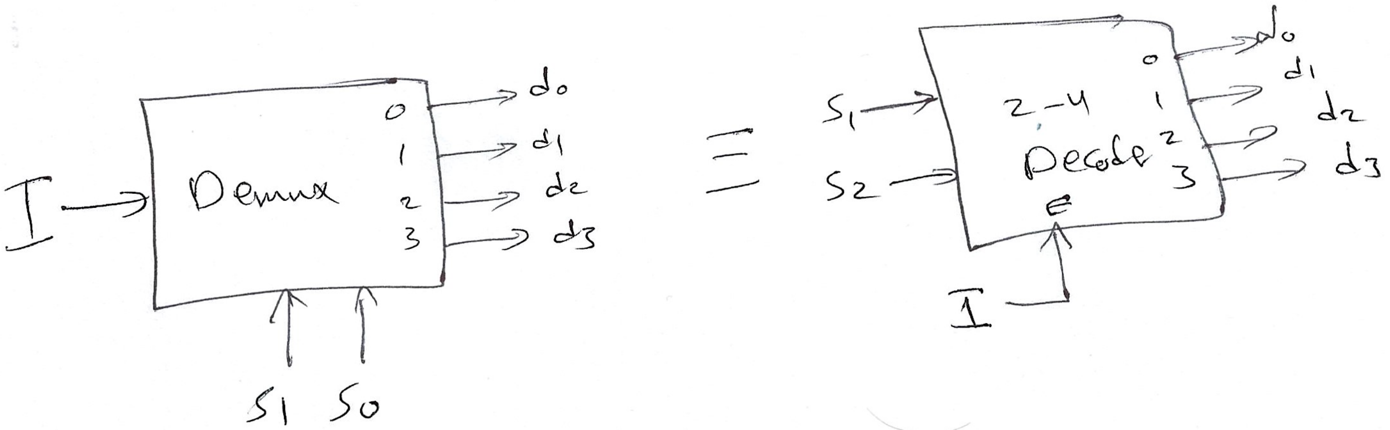


x	y	
0	0	A=F
0	1	B=F
1	0	C=F
1	1	D=F

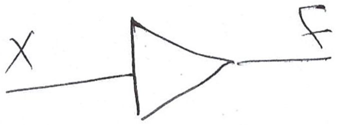


Circuit diagram (Demux)

* Demultiplexer = Decoder with Enable



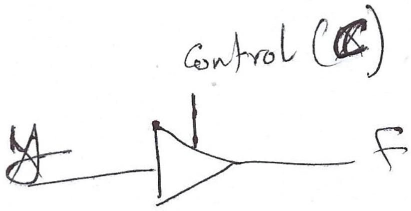
* Three (Tri) state buffer :-



buffer (normal)

F = X

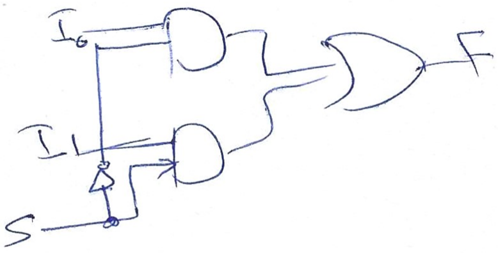
Normal buffer

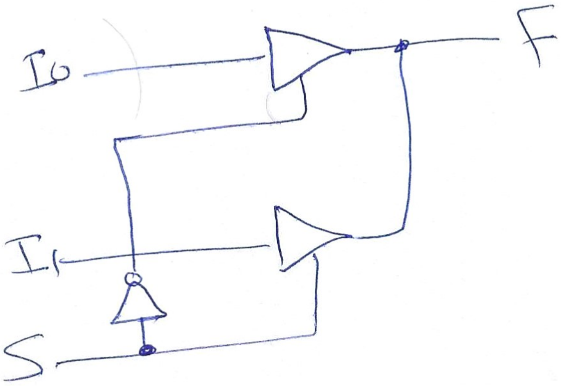


if C == 1 (short circuit)
F = y
else
F = z (open circuit)

Three state buffer

Example :- Design a 2X1 mux using buffer (Three state)

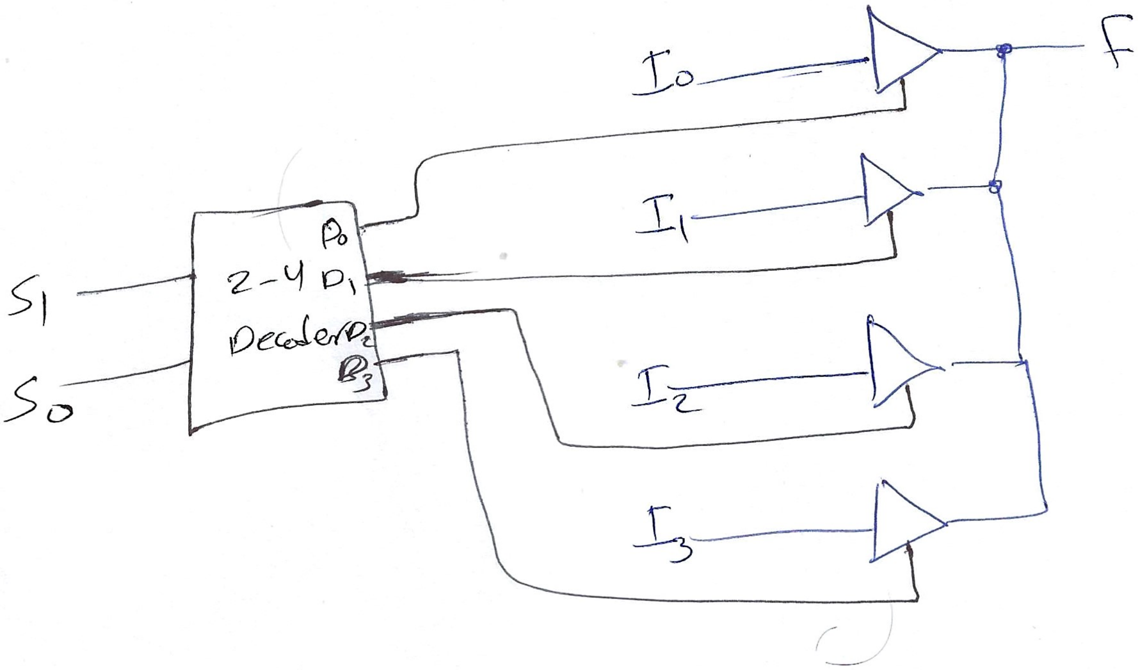




S = 0 F = I_0

S = 1 F = I_1

Example:- Design a 4-1 Mux using Three state buffer.



I_0
I_1
I_2
I_3
F
S_1
S_0
2-4
Decoder
D_0
D_1
D_2
D_3