Lecture-7 (MSI Circuits)

MSI → Medium scale Integration

4 most common MSI circuits are,

(i) Decoder (n to 2^n)

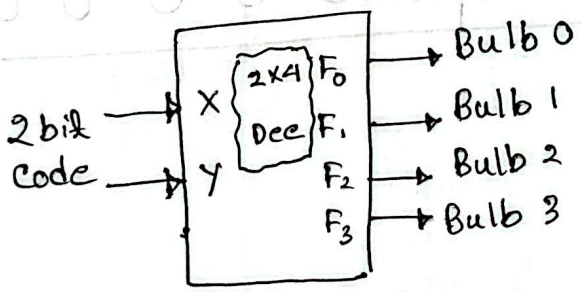
(ii) Demultiplexer (Num of selects, n to 2^n) (Output)

(iii) Encoder (Input 2^n, Output n)

(iv) Multiplexer (Input 2^n, Num of selects, N) (Output 1)

Decoder:

n inputs and 2^n outputs



X	Y	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

If codes 00, 01, 10, 11 are used to identify four light bulbs, we may use a 2-bit decoder.

$$F_0 = X'Y', \quad F_1 = X'Y, \quad F_2 = XY', \quad F_3 = XY$$

* Design a 3×8 decoder by Yourself:

3 input, 8 output

$n = 3 \;,\; 2^n = 2^3 = 8$

$F_0 = \bar{C}\bar{B}\bar{A}$, $F_1 = \bar{C}\bar{B}A$, $F_2 = \bar{C}B\bar{A}$

$F_3 = \bar{C}BA$, $F_4 = C\bar{B}\bar{A}$,

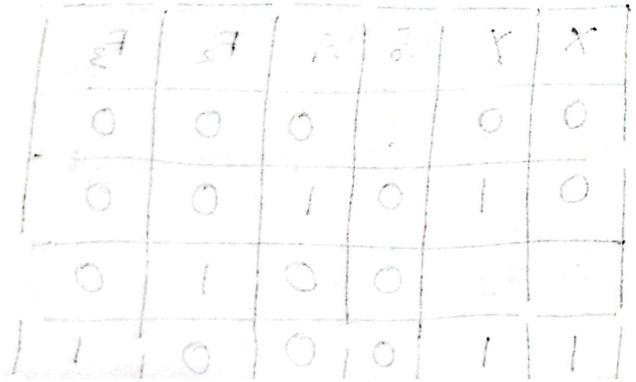
$F_5 = C\bar{B}A$

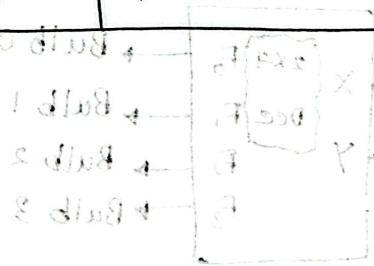
$F_6 = CB\bar{A}$

$F_7 = CBA$

Truth table:

C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0





Demultiplexer:

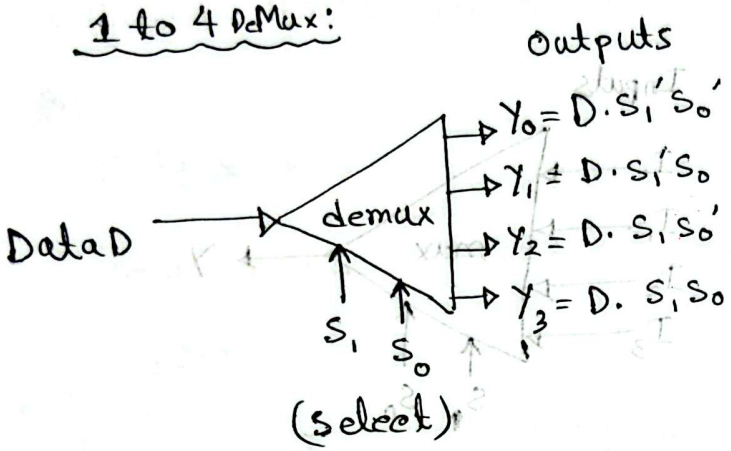
Output depends on select (S_1, S_0)

$= n$

If n num of selects

2^n output

1 to 4 DeMux:



S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

{Selectors kinda works like switch}

1 to 8 DeMux:

output 8

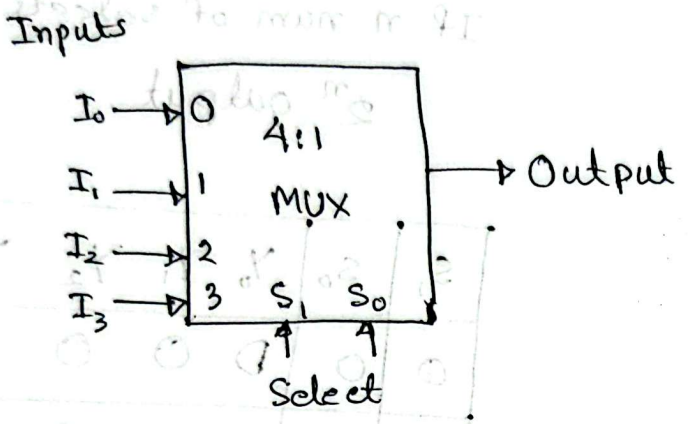
Input 1

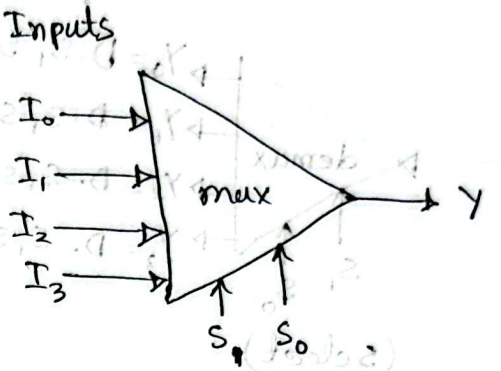
selector 3

* Always Input 1

Multiplexer:

4:1 MUX → * Output Always 1





$$\therefore Y = I_0(S_1'S_0') + I_1(S_1'S_0) + I_2(S_1S_0') + I_3(S_1S_0)$$

I_0	I_1	I_2	I_3	S_1	S_0	Y
d_0	d_1	d_2	d_3	0	0	d_0
d_0	d_1	d_2	d_3	0	1	d_1
d_0	d_1	d_2	d_3	1	0	d_2
d_0	d_1	d_2	d_3	1	1	d_3

* 8 to 1 MUX →

Input = 8

Output = 1

Selector = 3

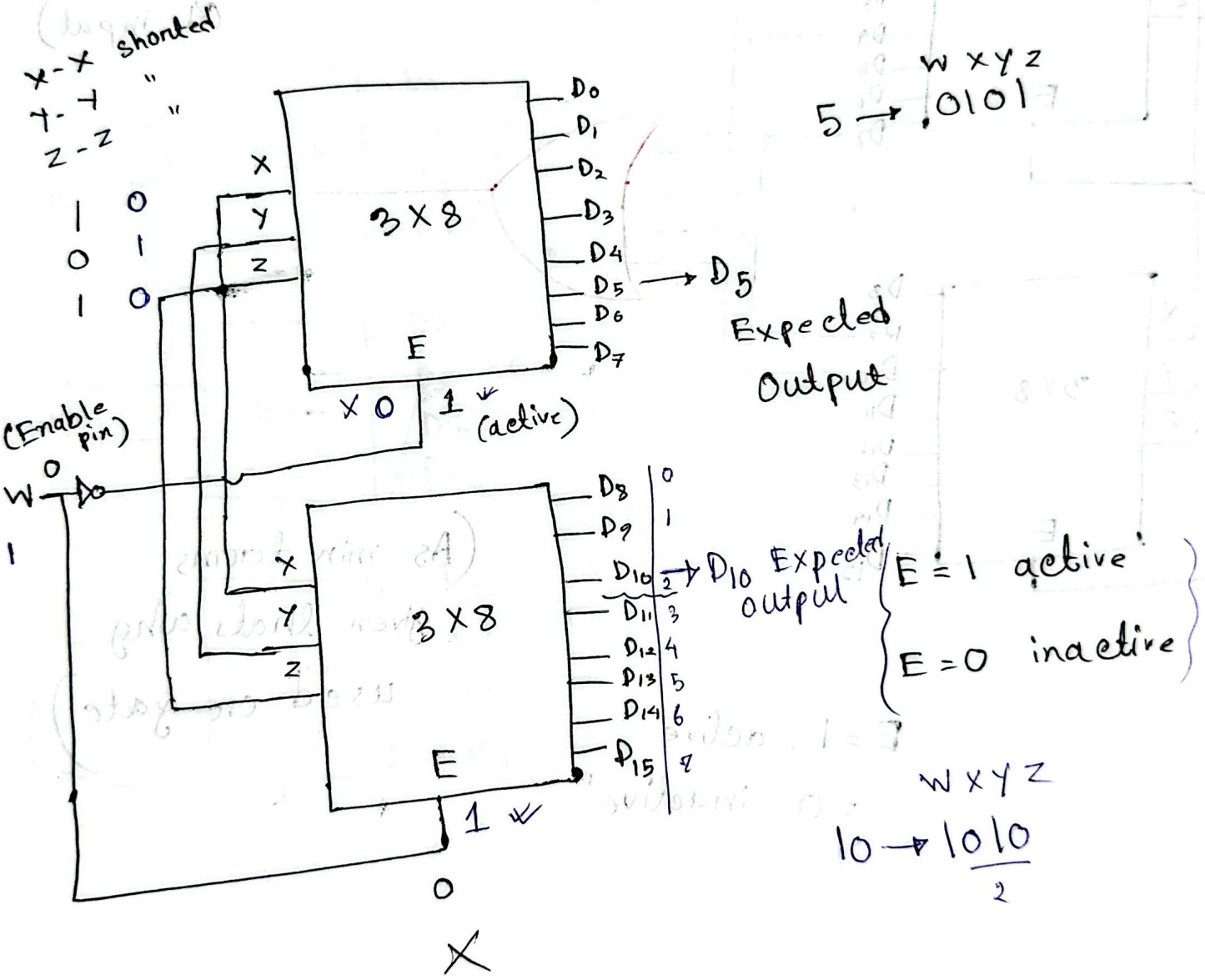
Building large decoders using small decoders.

* Build 4×16 Decoder using 3×8 Decoders.

Ans:

Output → 16

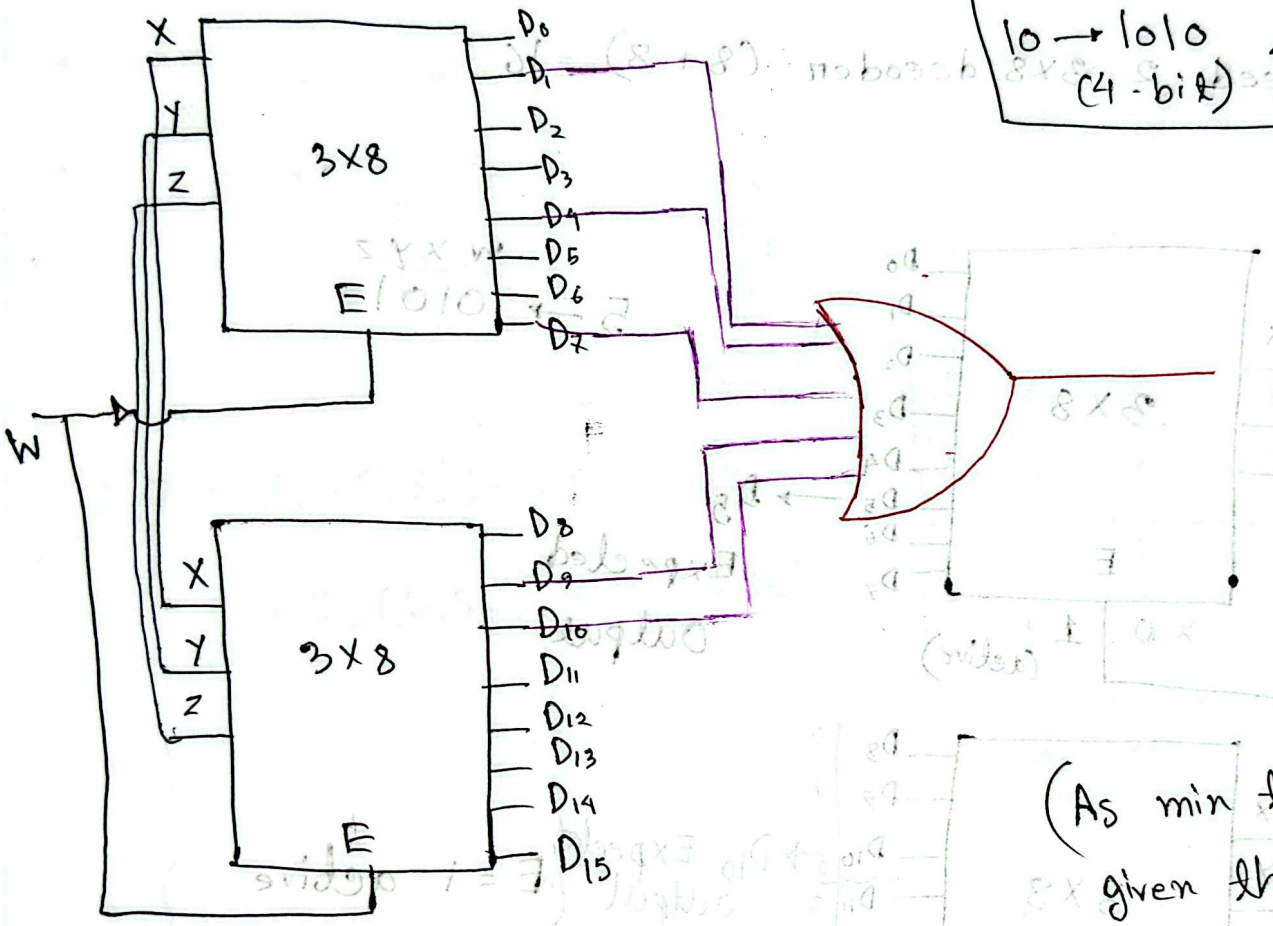
~~need~~ need 2 3×8 decoder (8 + 8) = 16



Function Implementation using Decoders.

* Implement $F(W, X, Y, Z) = \Sigma(1, 4, 7, 9, 10)$ using 3×8 Decoders only.

10 → 1010 (4-bit) 4 variable (4-input)

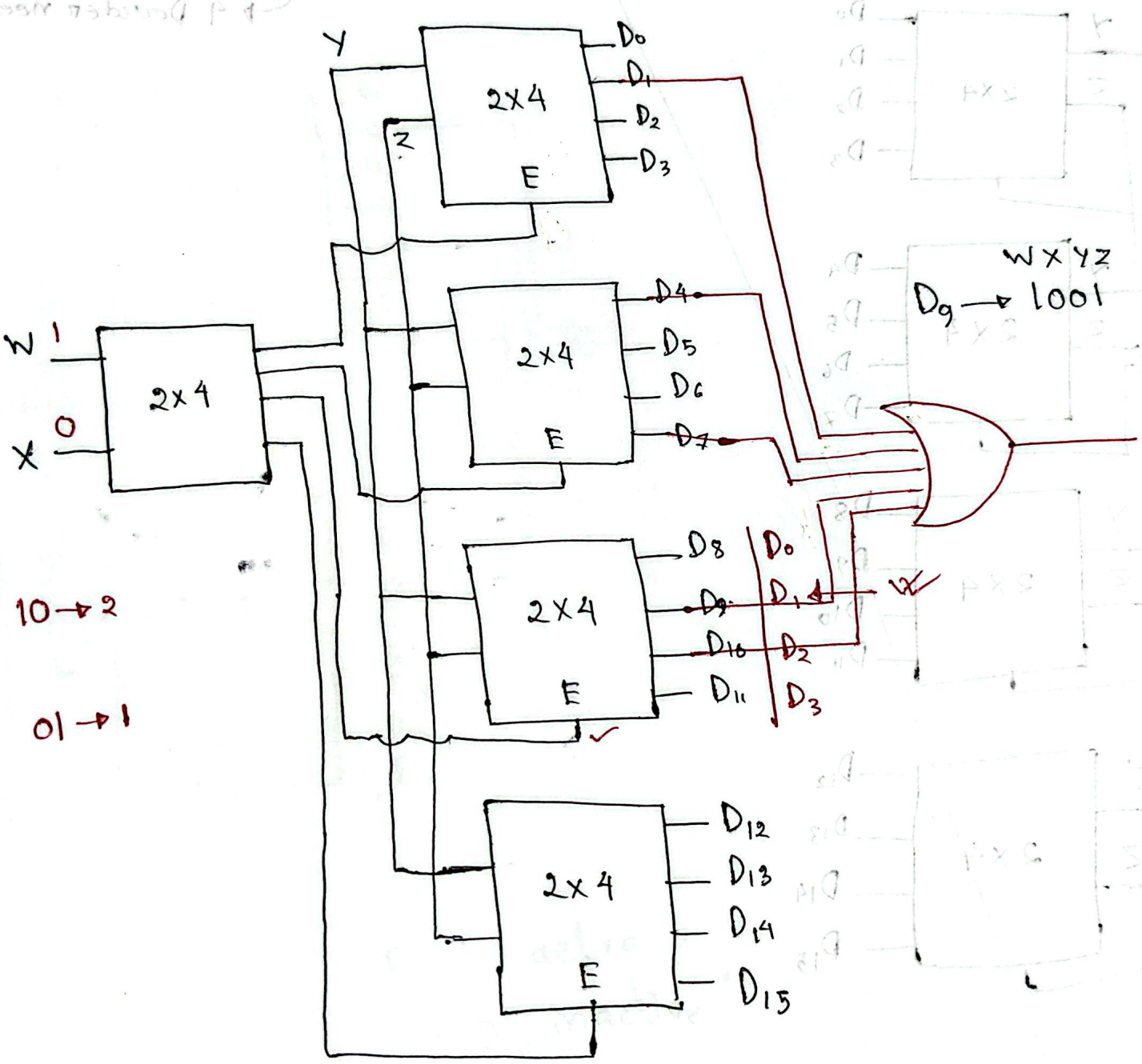


(As min terms given thats why used or gate)

E = 1, active

= 0, inactive

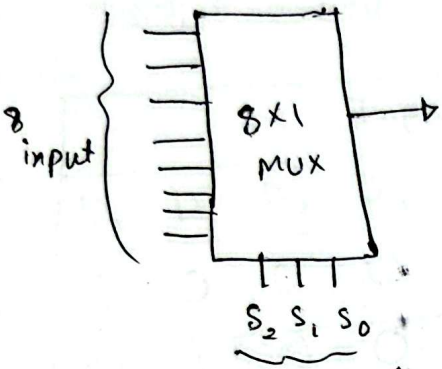
Implement $F(W, X, Y, Z) = \Sigma(1, 4, 7, 9, 10)$ using 2x4 Decoders only.



Building a large MUX using smaller MUX's

* Implement 8×1 MUX using 4×1 MUX.

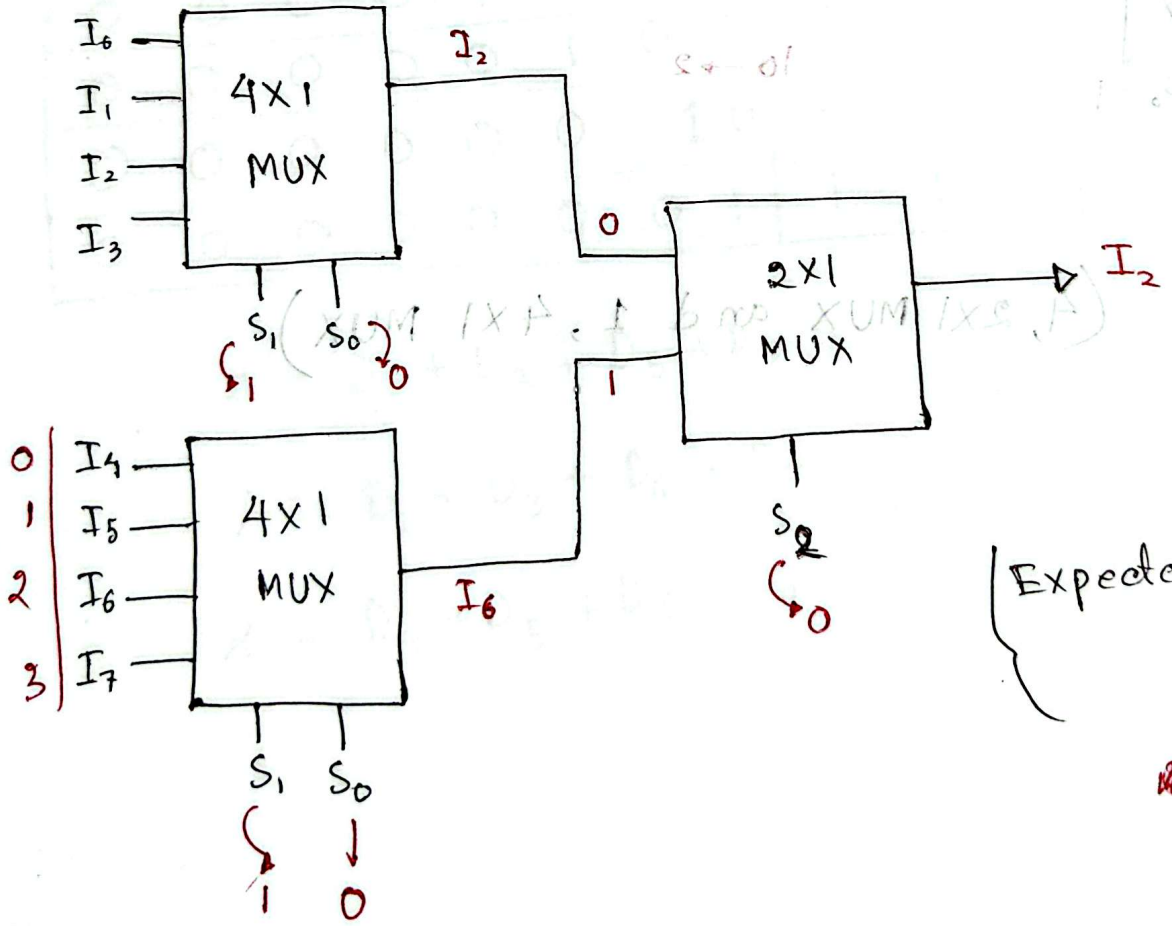
8×1 MUX



S_2, S_1, S_0
(MSB) (LSB)

(Method-1)

(S_1, S_0 in Both MUX's they are shorted)



Expected Output

I_2

010

$S_2 S_1 S_0$

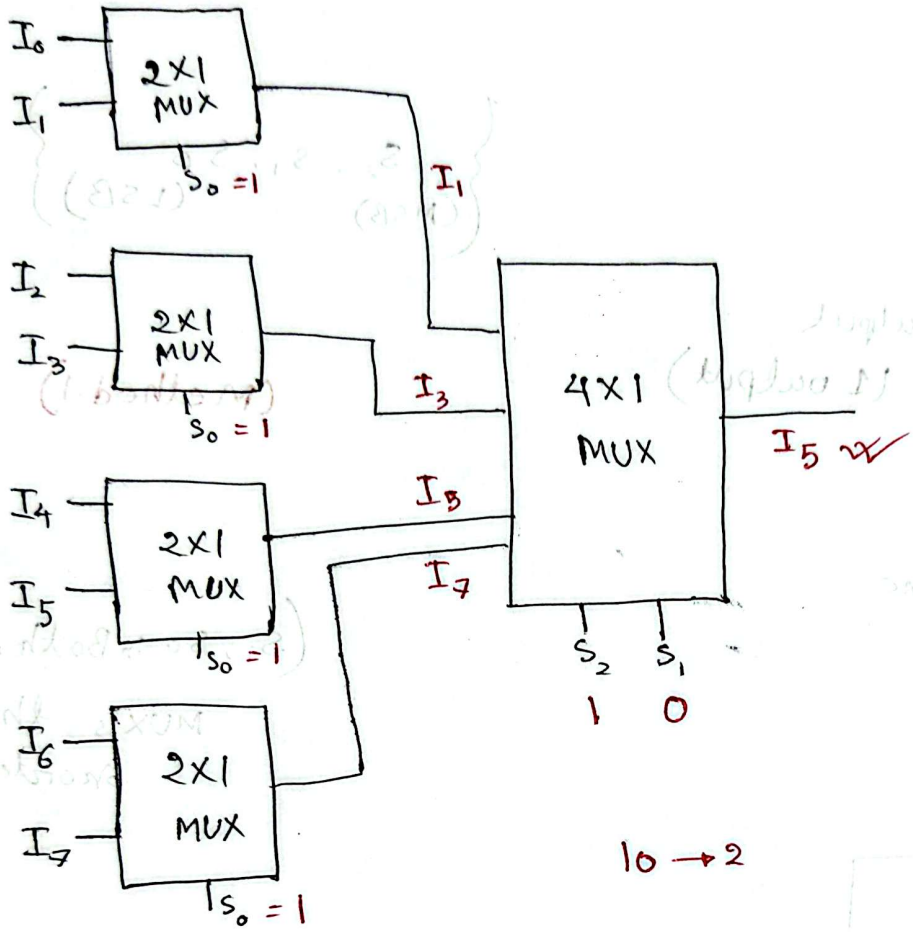
(Method-2)

MSB LSB

S_2 S_1 S_0

1 0 1 → 5

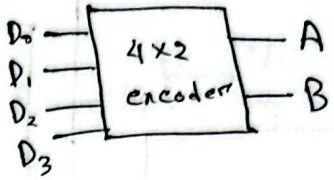
(I_5)



(4, 2X1 MUX and 1, 4X1 MUX)

Encoder

2^n input → n output



8×3 Encoder → 8-bit

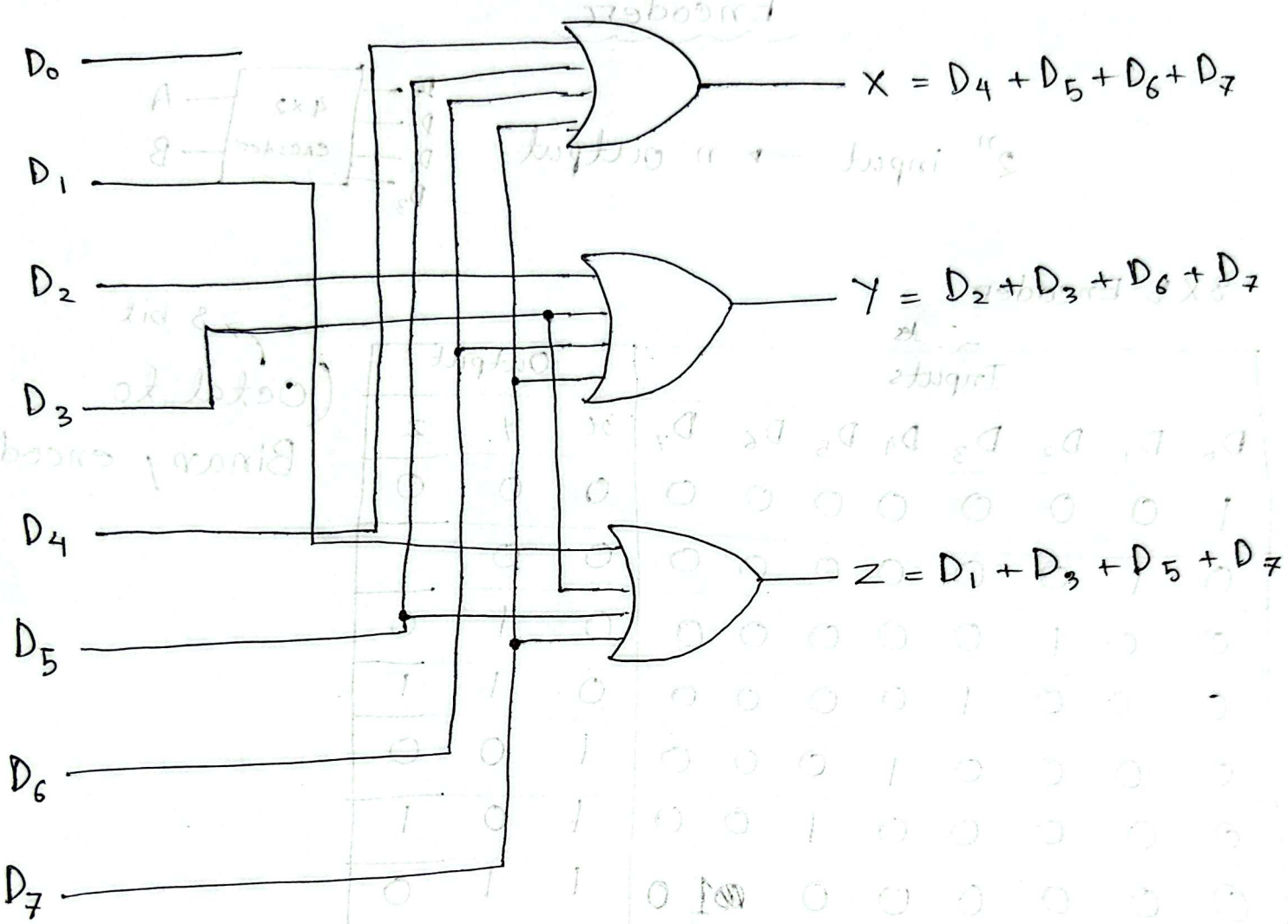
(Octal to Binary encoder)

Inputs								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

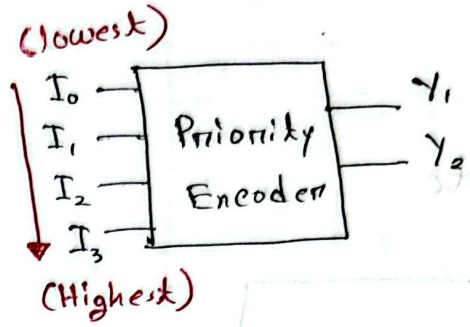
$$x = D_4 + D_5 + D_6 + D_7$$



{Circuit Diagram of 8×3 encoder}

Priority Encoder:

2^n input → n output



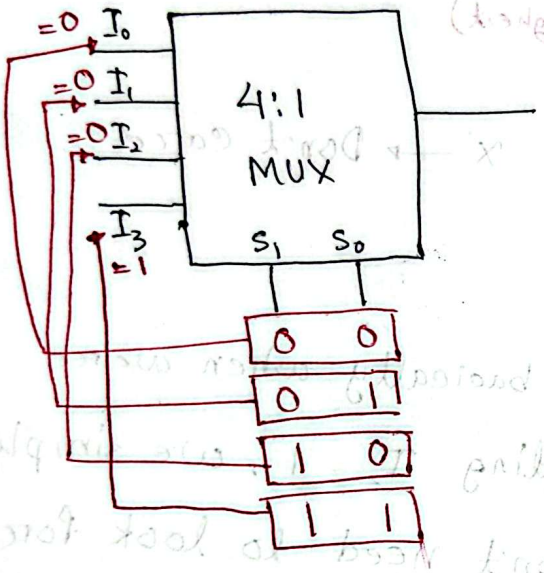
I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

x → Don't care

So bacically when we're getting $I_2 = 1$, we simply don't need to look for the values of I_1 and I_0. We can simply write the binary value of 2 in the Y_1 and Y_0.

Use 4:1 MUX to design AND, OR, NOT Gate

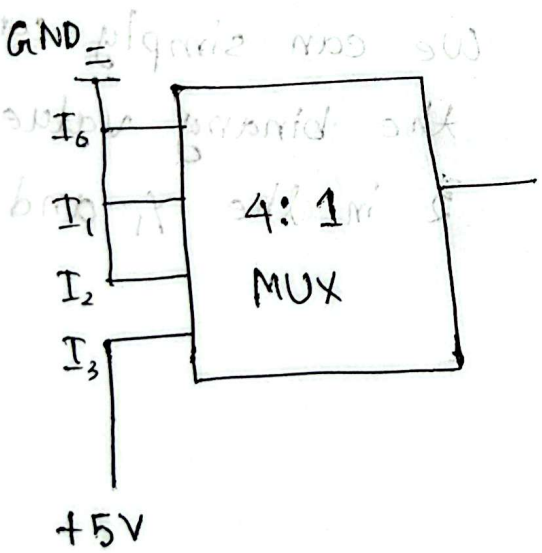
* AND Gate



AND

0	0	0
0	1	0
1	0	0
1	1	1

So what we can do is,

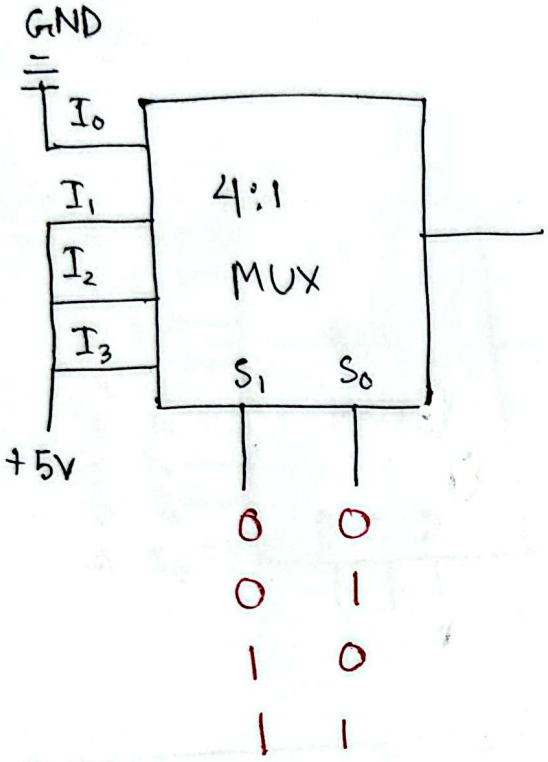


0 → GND

1 → +5V

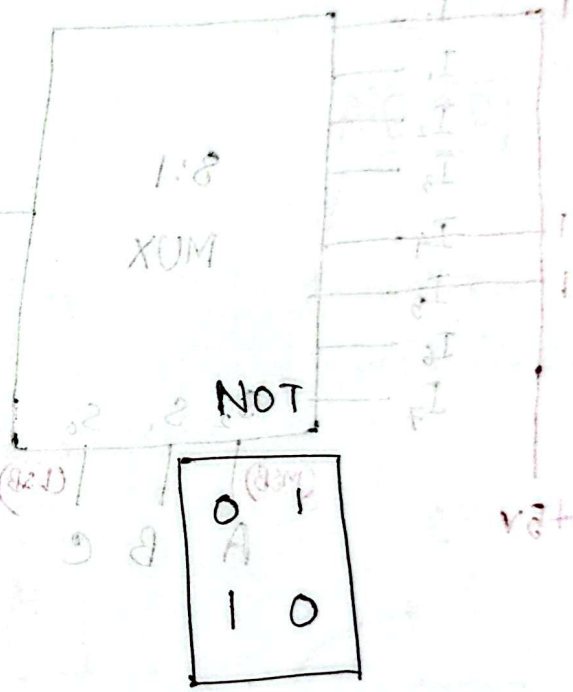
* OR Gate

OR		
0	0	0
0	1	1
1	0	1
1	1	1

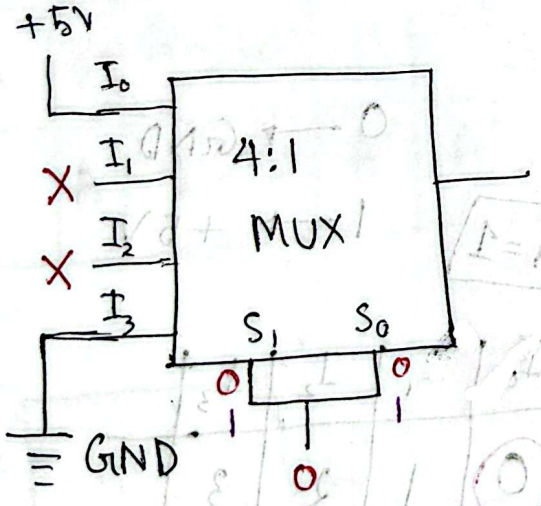


0 → GND

1 → +5V



* NOT Gate



→ 2 selectors in 4:1 MUX

But NOT gate takes input → 1 bit

0 → GND

1 → +5V

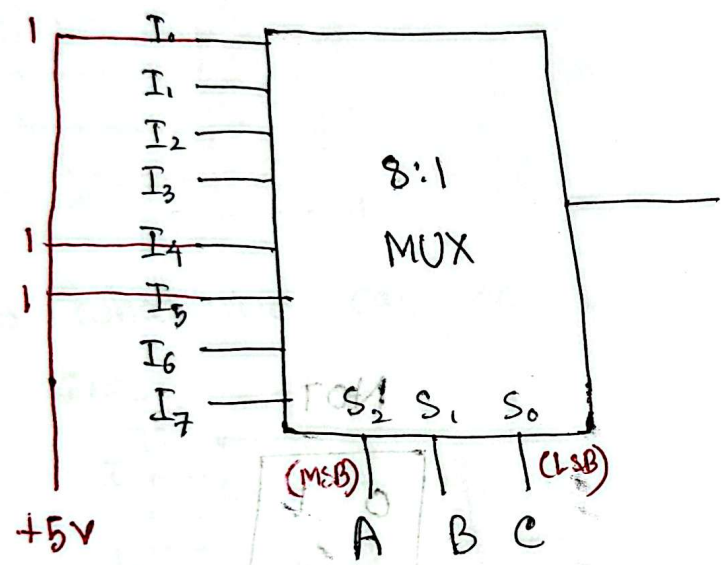
How to implement a boolean Function using 8 to 1 (8:1) MUX.

① Build $F = \Sigma(0,4,5)$ using

(i) 8:1 MUX

(ii) 4:1 MUX

3 bit (3 variable)

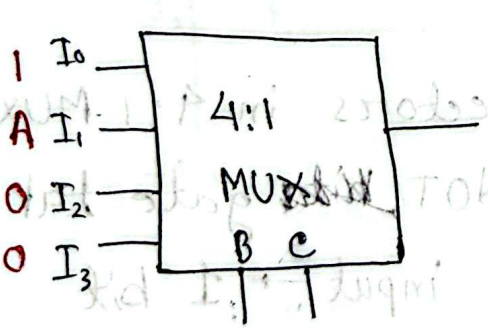
① 

0 → GND

1 → +5V

$A' + A = 1$

	I_0	I_1	I_2	I_3
A'	0	1	2	3
A	4	5	6	7
	1	A	0	0

(ii) 

(Larger function smaller MUX)

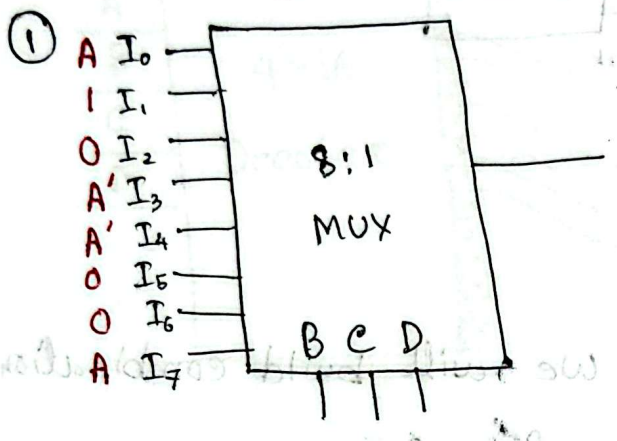
(11) Build $F(A,B,C,D) = \Sigma(1,3,4,8,9,15)$ using

① 8:1 MUX

⑪ 4:1 MUX

(larger function with smaller MUX)

① 8:1 MUX



	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	A	1	0	A'	A'	0	0	A

(start from the right)

A B C D

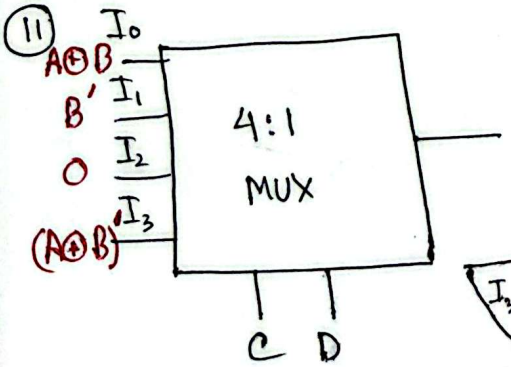
1 variable left (2 combination

00
01
10
11

A B | C D

left

⑪



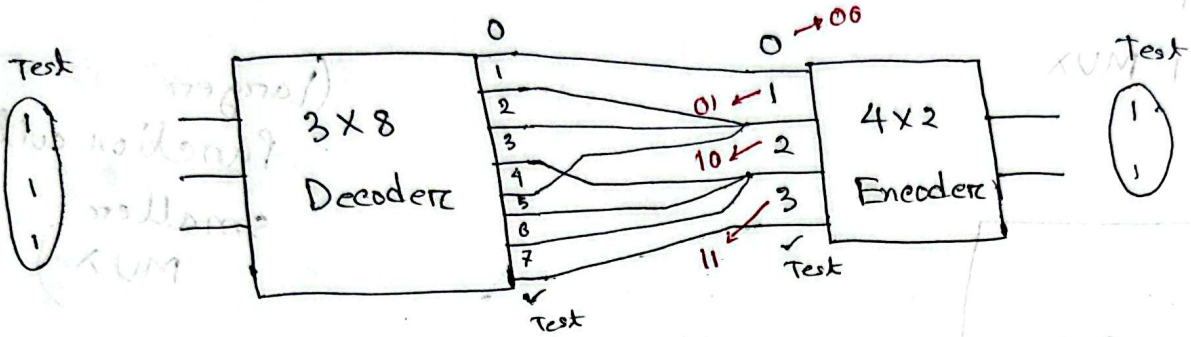
$I_3 = A'B' + AB = A \odot B = (A \oplus B)'$

I_0 $A'B + AB' = A \oplus B$

I_1 $A'B' + AB' = B'(A'+A) = B'$

	I_0	I_1	I_2	I_3
A'B'	0	1	2	3
A'B	4	5	6	7
AB'	8	9	10	11
AB	12	13	14	15
	A⊕B	B'	0	(A⊕B)'

Design Full Adder using 3x8 Decoder and 4x2 Encoder.



X	Y	Z	C	S	
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	0	5
1	1	0	1	0	6
1	1	1	1	1	7

Carry → C, Sum → S

We will build combinations for C S

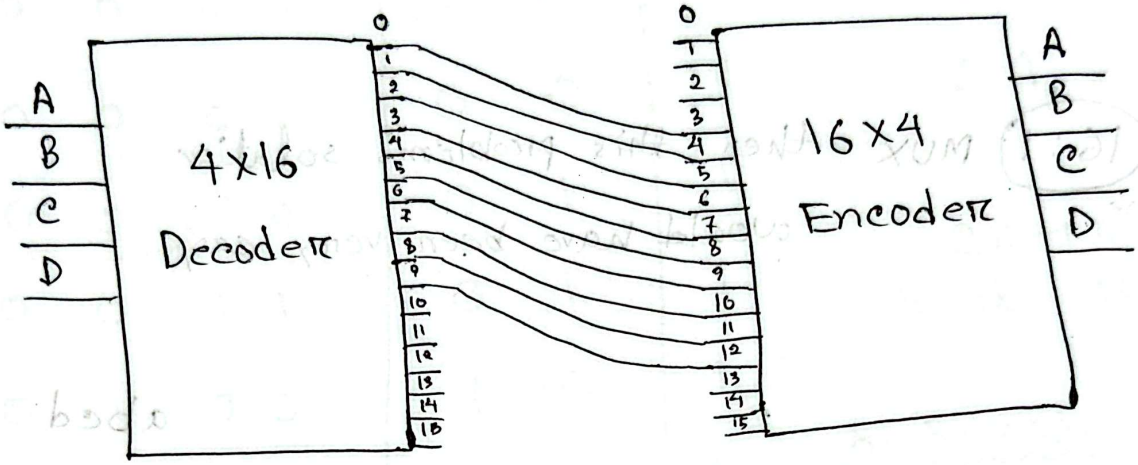
00, 01, 10, 11

0 → 1 → 3 → 7

1 → 2 → 4

3 → 5 → 6

Design BCD to Excess-3 code converter using 4X16 decoder and 16X4 encoder



{As BCD → (0-9) will be used}

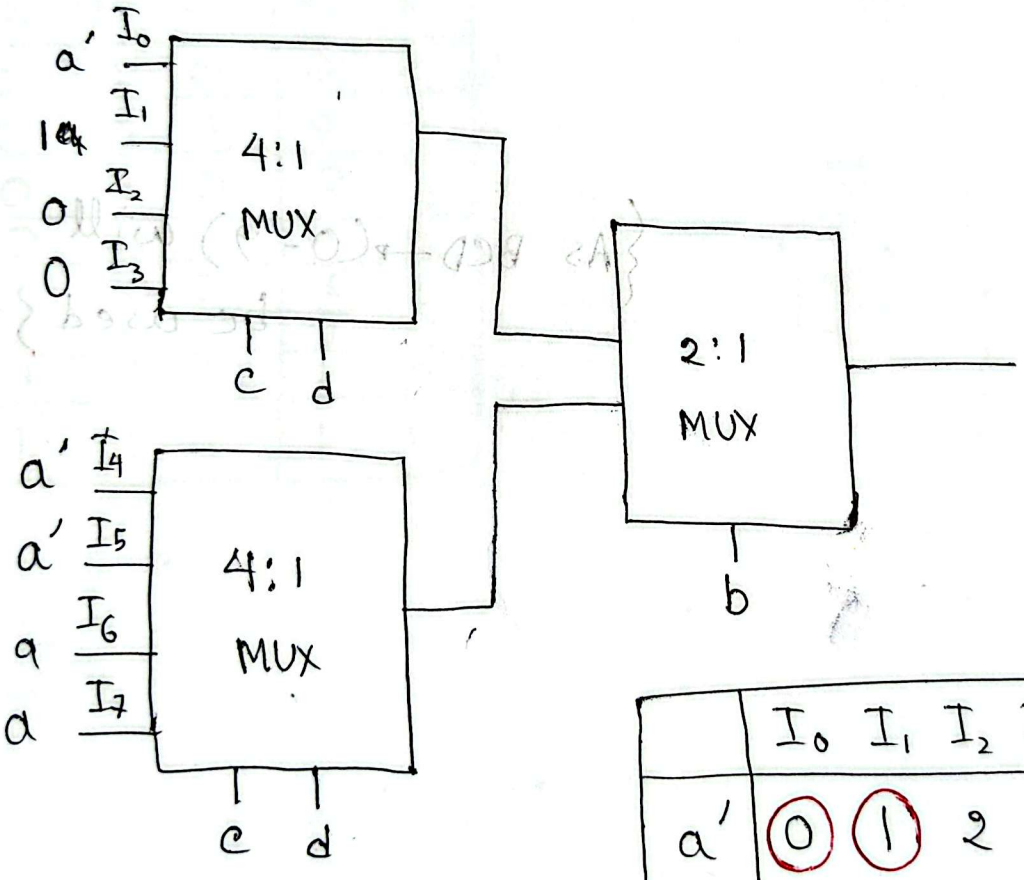
Implement the following Function using 4:1 and 2:1 MUX (both).

$F(a,b,c,d) = \Sigma(0,1,4,5,9,14,15)$

Ans:

if we could use (16:1) MUX then this problem's solution would have been very easy

abcd



a → a', a

$\{a' + a = 1\}$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
a'	0	1	2	3	4	5	6	7
a	8	9	10	11	12	13	14	15
	a'	1	0	0	a'	a'	a	a

(MSB) PRIORITY ENCODER (4-bit)

D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Here D_1's priority is greater than D_0 and $D_1 = 1$ so we wrote the binary of 1 → 01

Here, D_2's priority is greater so we wrote 2's binary = 10

D_3 is greater priority so 3 binary = 11

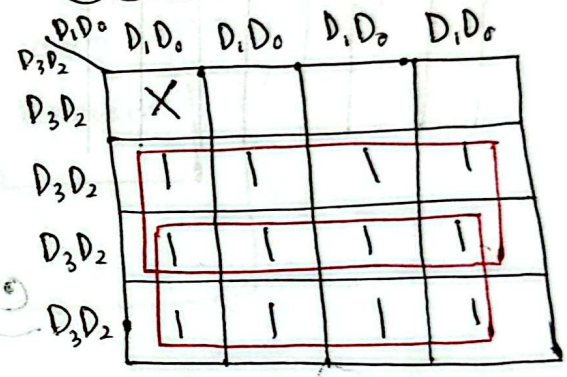
$\therefore A_1 = D_2 + D_3$

$\therefore A_0 = D_3 + D_2' D_1$

2n → n

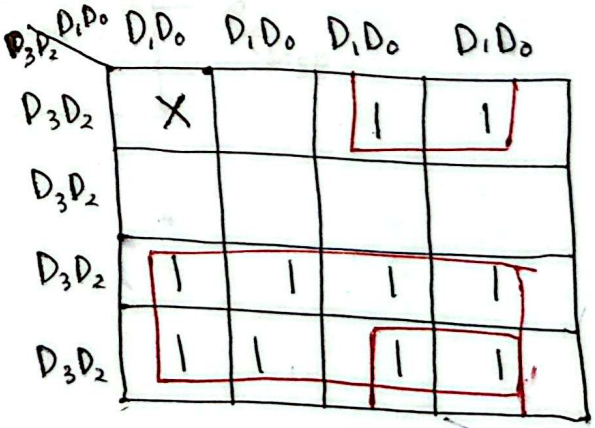
input output

A1 k-map:



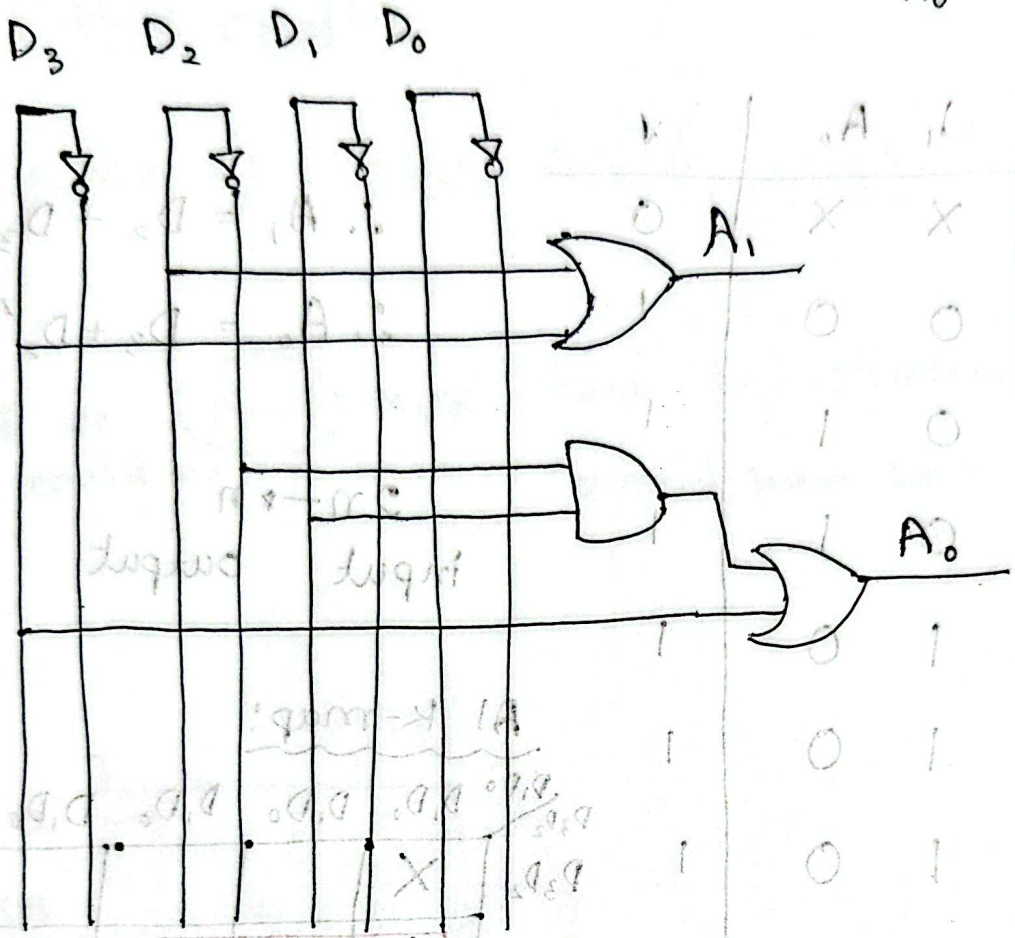
(We plugged 1 where A_1 val is 1)

A0 k-map:



$A_1 = D_2 + D_3$

$A_0 = D_3 + D_2' D_1$

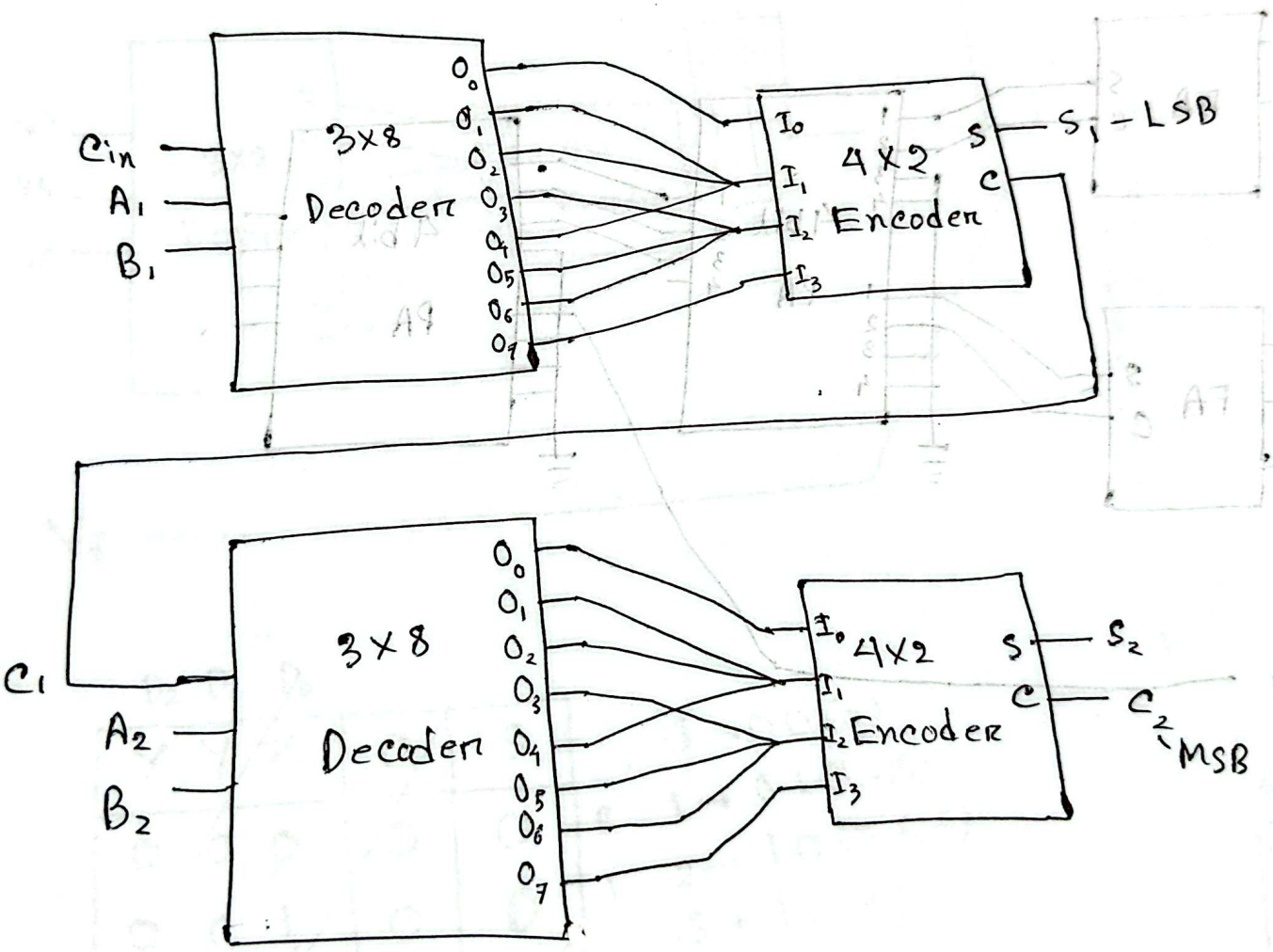


Circuit Diagram.

Exercise: Design a 2-bit PA where go have to use appropriate encoders and decoders. to build the internal circuit.

Ans: Generally, for 2-bit PA → 2 Full Adder

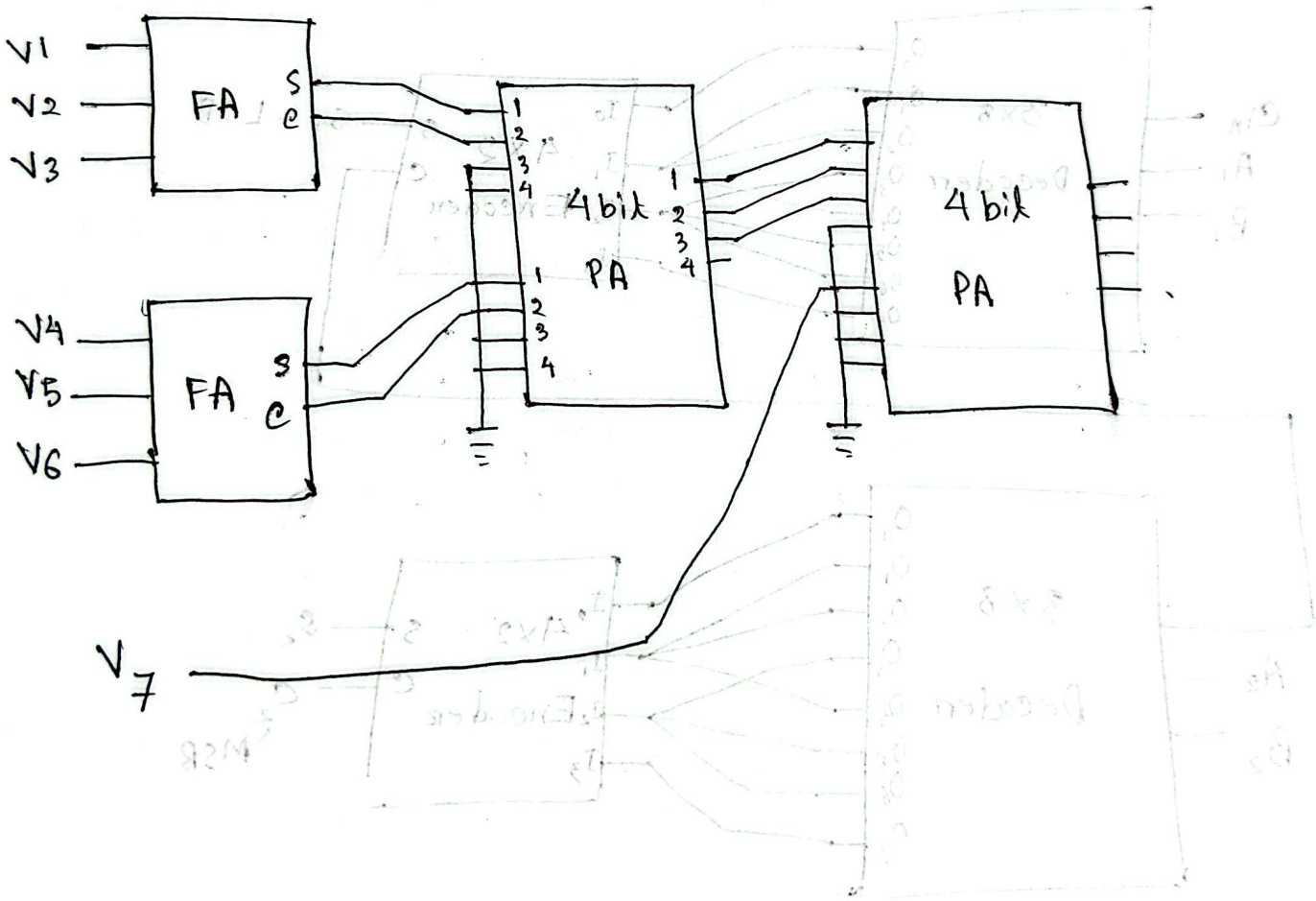
4-bit PA → 4 " "



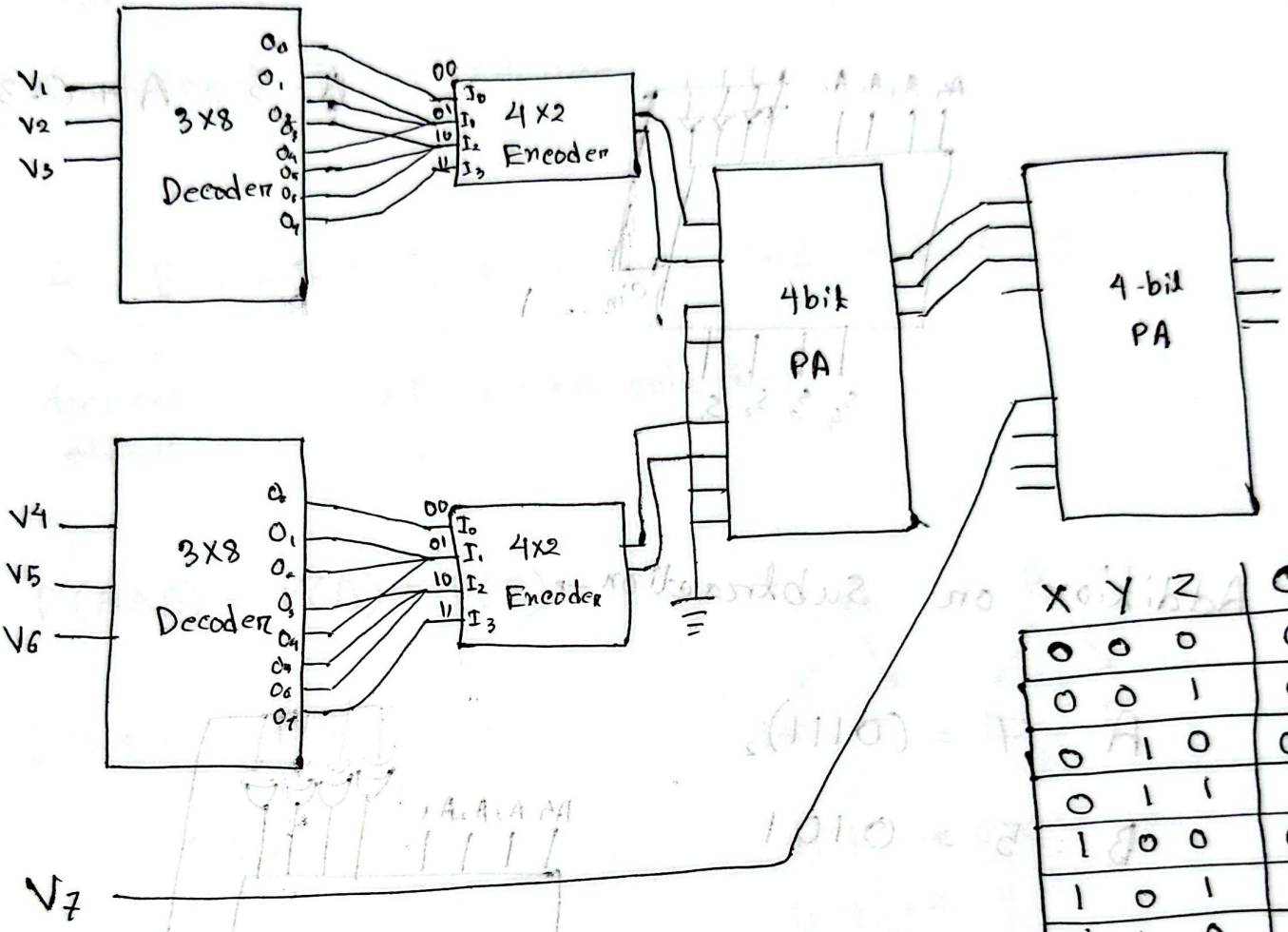
Design a 7-person Voting system using encoders and decoders and parallel adders. You can't use full adders. Meaning, for the inputs, you MUST use encoders and decoders as required.

Ans:

Normally with Full adder we used to draw →



But here,



X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$D_2\ D_1\ D_0$

X	Y	Z	C	S	
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	2
0	1	1	0	1	3
					4

0 → 00 (0)
1 → 01 (1,2,4)
2 → 10 (3,5,6)
3 → 11 (7)

$I_0\ I_1\ I_2\ I_3$

$D_2\ D_1\ D_0$

X Y Z	D_1	D_0
0 0 0	X	X
0 0 1	0	1
0 1 0	1	0