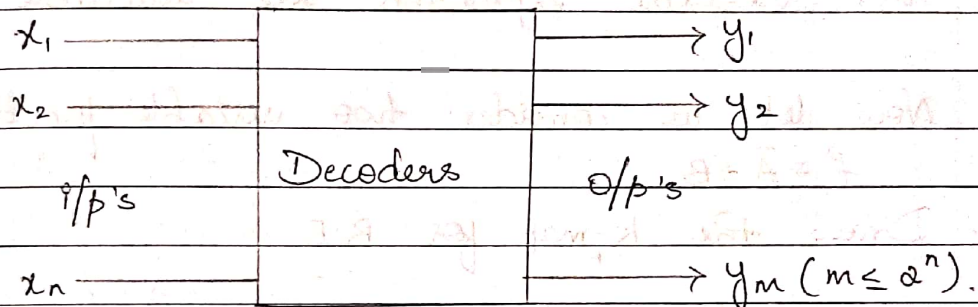


MODULE - 2

ANALYSIS AND DESIGN OF
COMBINATIONAL LOGIC CIRCUITSDecoders:

It is a device which after activated, it selects one of its possible 2^n outputs based on its 'n' bit digit.

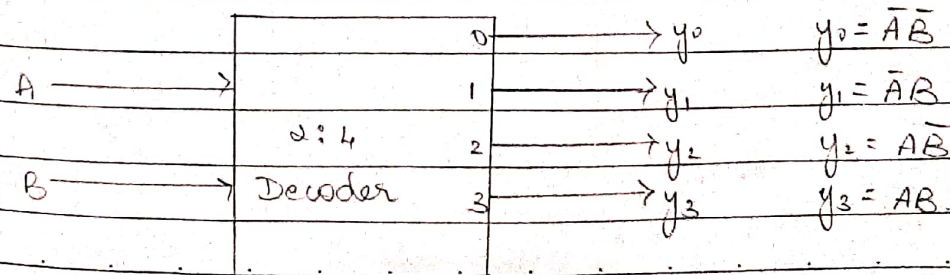
There are several types of decoders, the most common is the $n \rightarrow 2^n$ types.

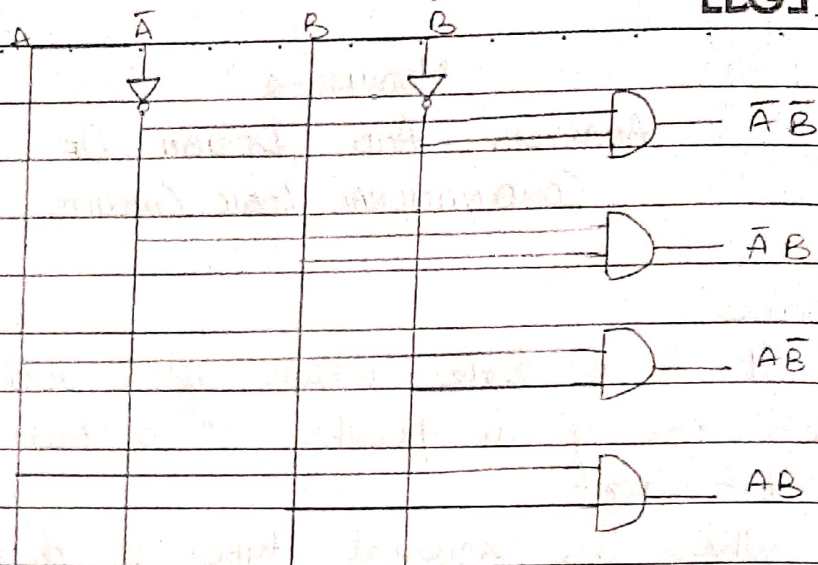


Now let us look at few typical commercial available decoders.

1) $2:4$ decoder:

Input		Output			
A	B	y_0	y_1	y_2	y_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

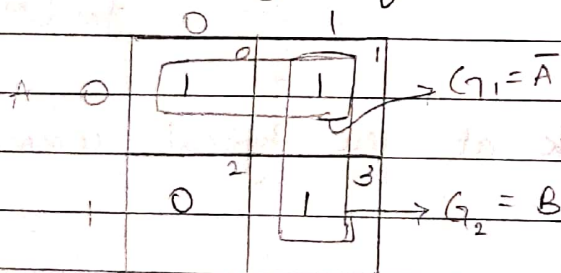




Observe that there are two variable expression are available.

Eg 1: Now let us consider two variable function.
 $f = \bar{A} + B$

⇒ Draw the k-map for B.E



$$y = \bar{A} + B$$

$$y_0 = 1 + 0 = 1$$

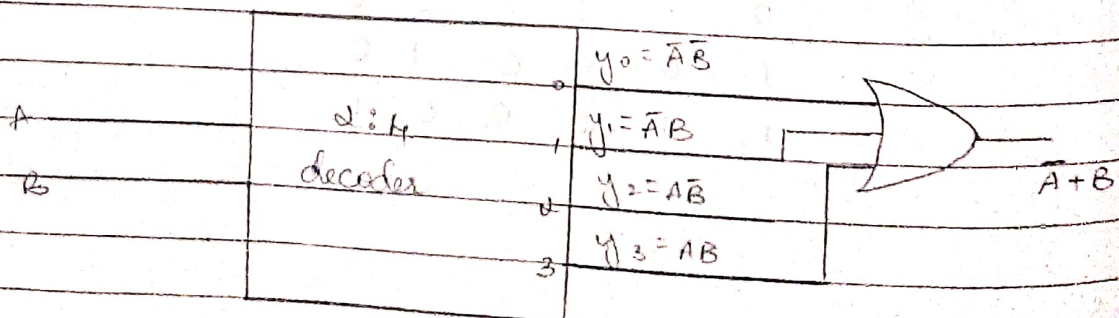
$$y_1 = 1 + 1 = 1$$

$$y_2 = 0 + 0 = 0$$

$$y_3 = 0 + 1 = 1$$

Note: Here grouping is done by considering 1's.
 $(G_1 + G_2)$

$$\therefore f(A, B) = \bar{A} + B$$

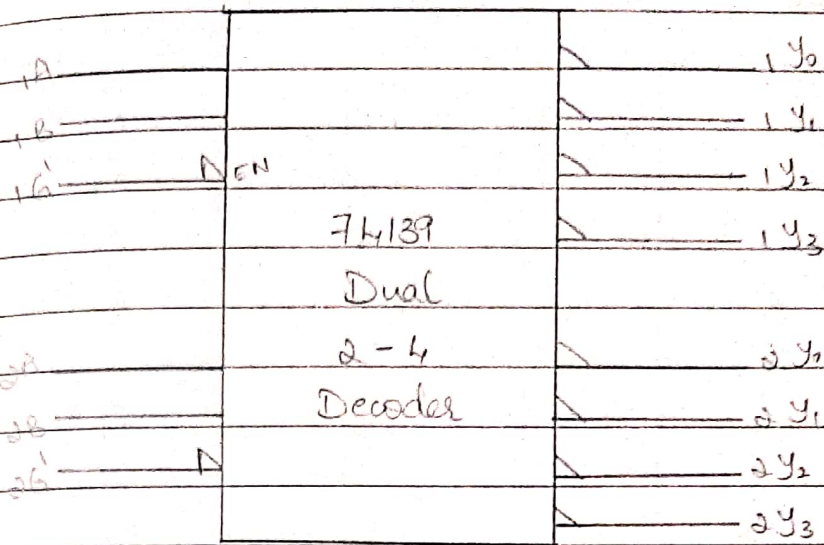


74139

to TTL family
Transistor-Transistor Logic

URBAN
EDGE

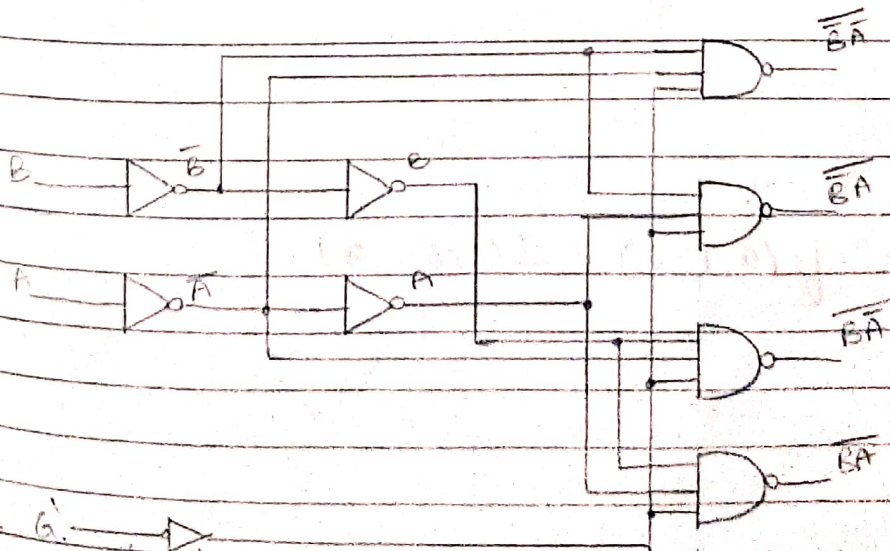
Dual 2:4 decoder using IC 74139



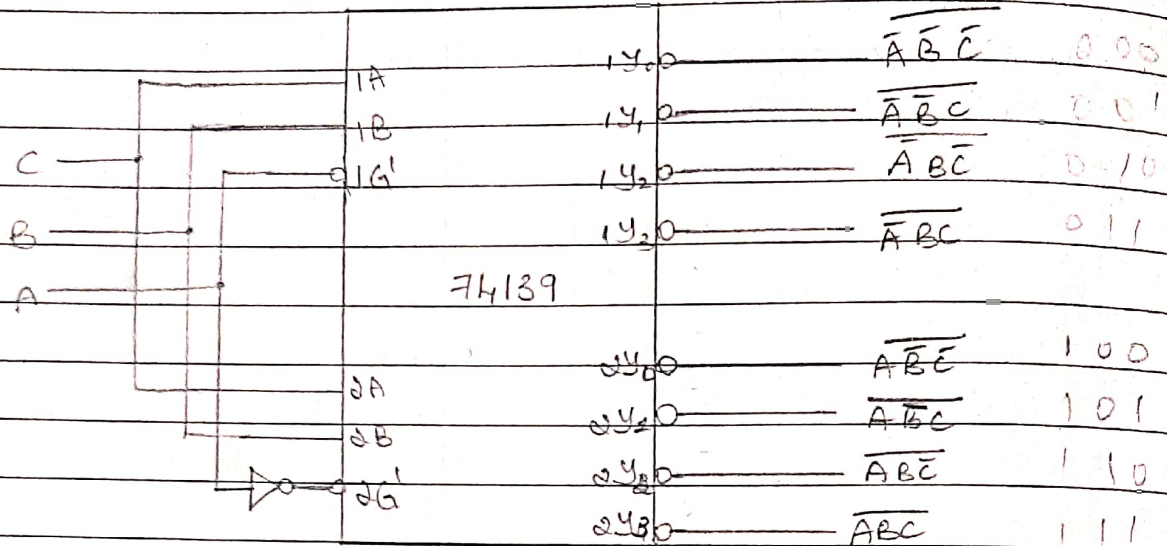
Dual 2-4 line decoder

Inputs			Outputs			
Enable	Select					
G'	B	A	Y_0	Y_1	Y_2	Y_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

functional table of 74139

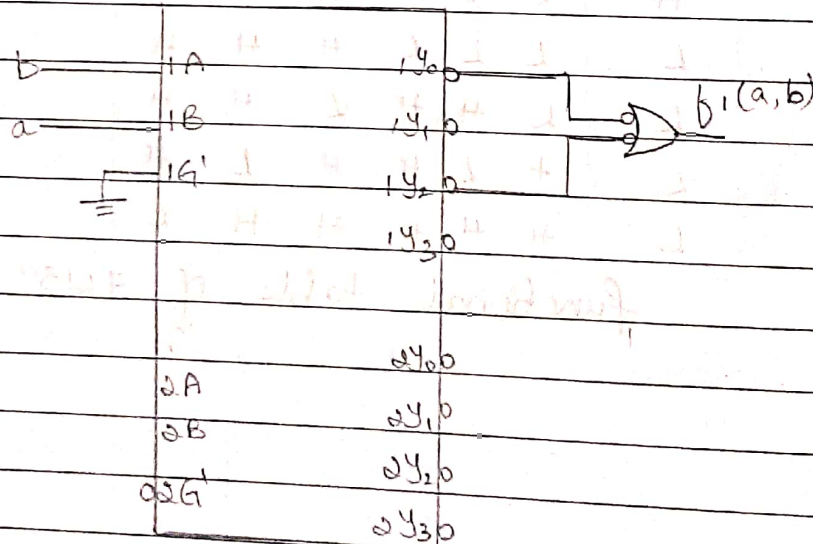


* Using dual 2-4 decoder we can construct a single 3-8 decoder



$\Sigma = \text{OR gate}$

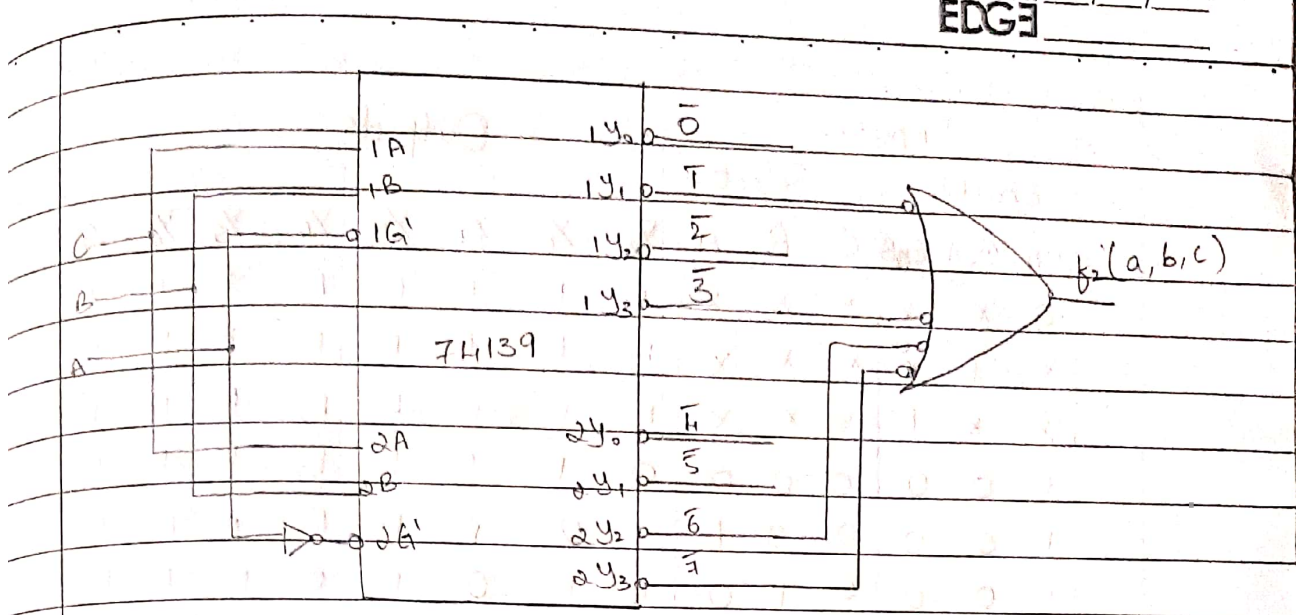
Rearrange the following B.F using 74139.
 $f_1(a, b) = \Sigma(0, 2)$



* Consider $f_2(a, b, c) = \Sigma(1, 3, 6, 7)$

IC = Integrated chip
chip $\Rightarrow$ silicon $\Rightarrow$ silica

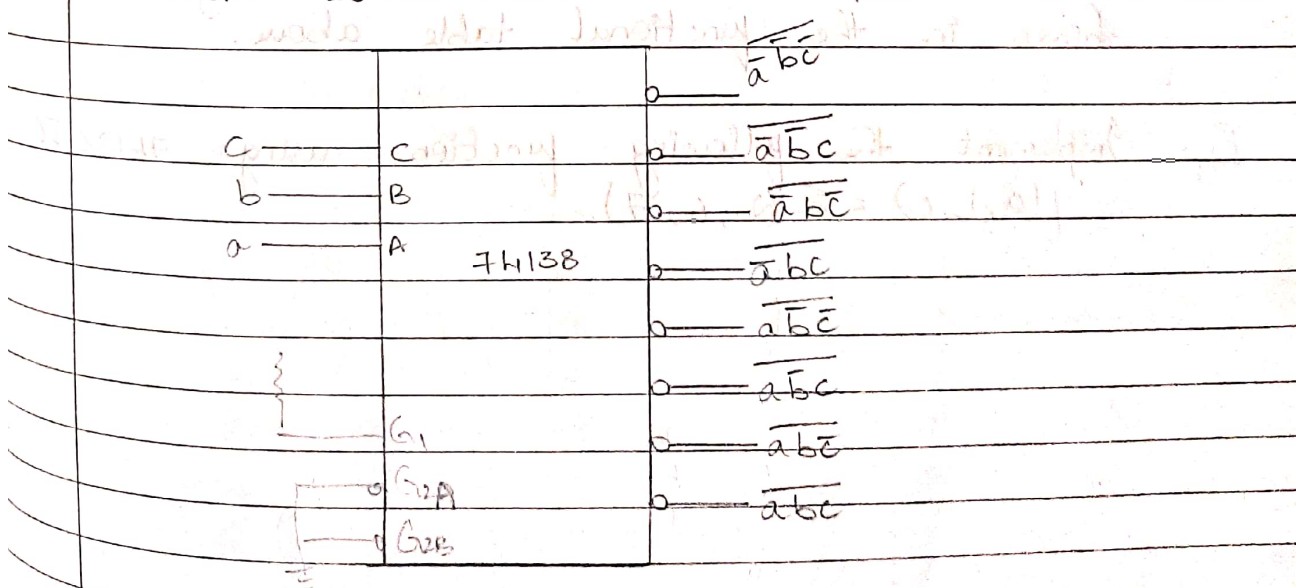
URBAN
EDGE



3-8 line decoders using IC 74138:

As in the case of 2-4 line decoders, 3-8 lines decoders are also available in IC named by 74138.

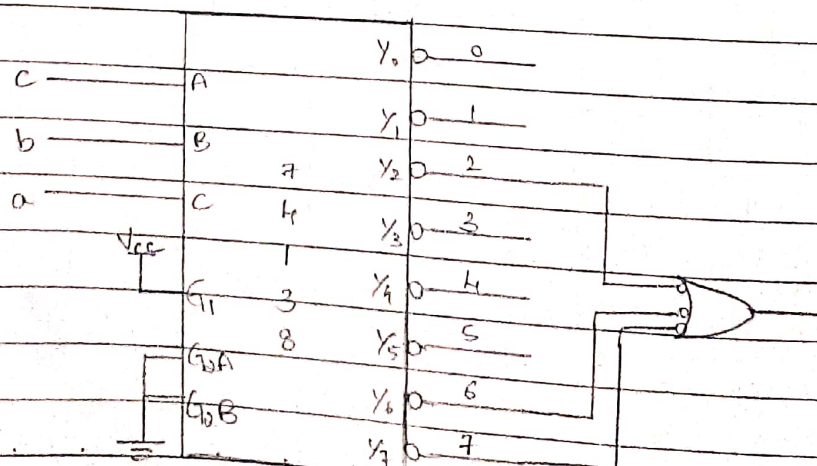
The device has 3 inputs A, B, C along with 3 enable functions i.e., G_1 , G_{2A} & G_{2B} . Out of 3 enable inputs G_{2A} & G_{2B} are bubbled version i.e., $G_{2A} = G_{2B} = 0$ & $G_1 = 1$.



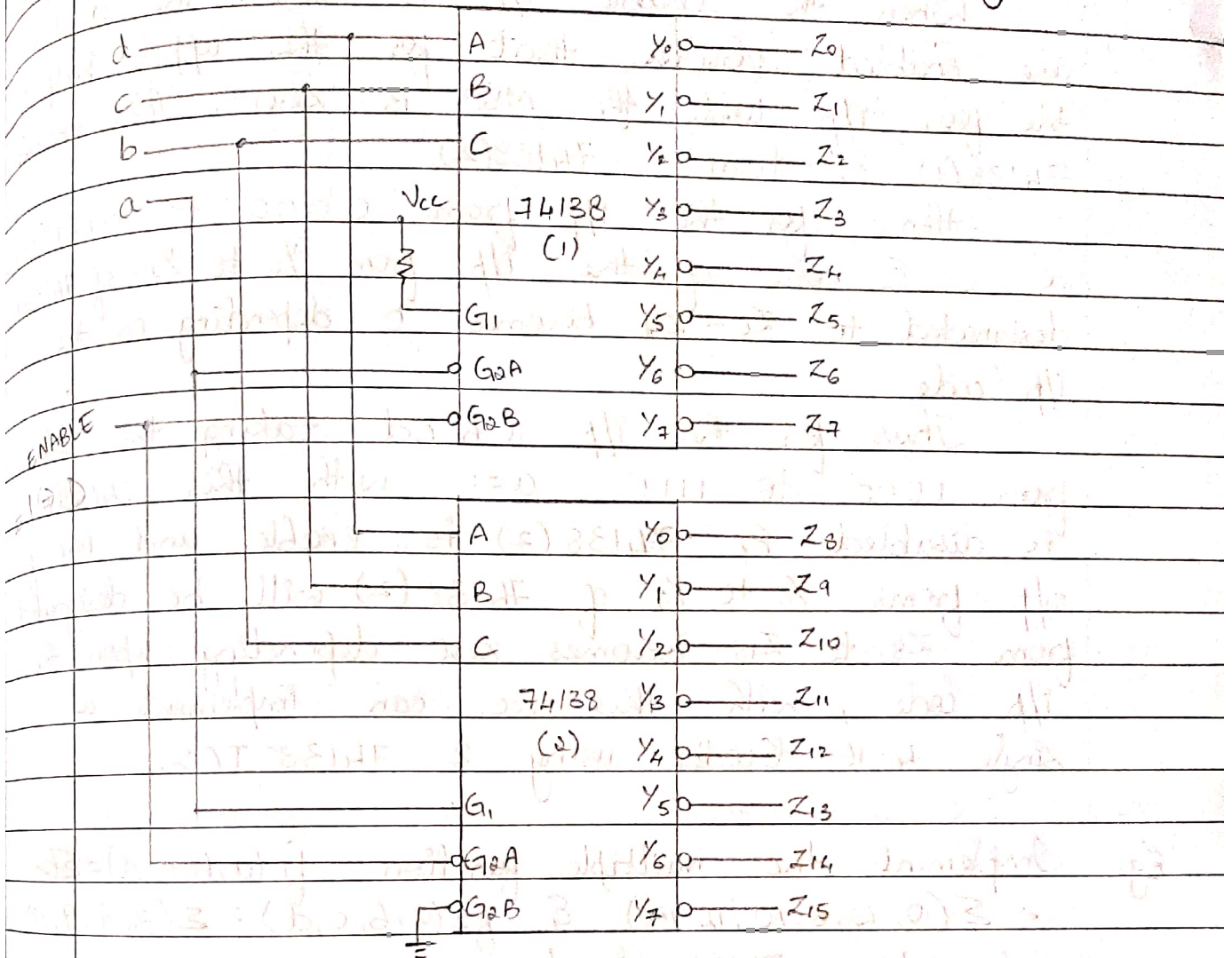
Inputs						Outputs							
Enable			Select										
G_1	G_2A	G_2B	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

The device functions as a decoder only when $G_1 = 1$ & $G_2A = G_2B = 0$ and under other conditions all the outputs will be high (1) as shown in the functional table above.

Eg: Implement the following function using 74138 IC.
 $f(a, b, c) = \Sigma(2, 6, 7)$.



Construct / Implement 4-to-16 line decoder using 74138 IC.



Solⁿ: In order to build a single 4-16 decoder we are supposed to use 2 sections of IC.

The most significant bit A is designated to zero for the first half of the function table & 1 for remaining rows is used in order to configure 4-16 decoder.

Let A, B, C, a, b, c, d be the 4 i/p with d being the LSB where G1 of 74138(1) & G2B of 74138(2) are connected to Vcc & ground.

G2B (74138(1)) & G2A (74138(2)) are used together as enable inputs.

When enable i/p is 1 both the devices

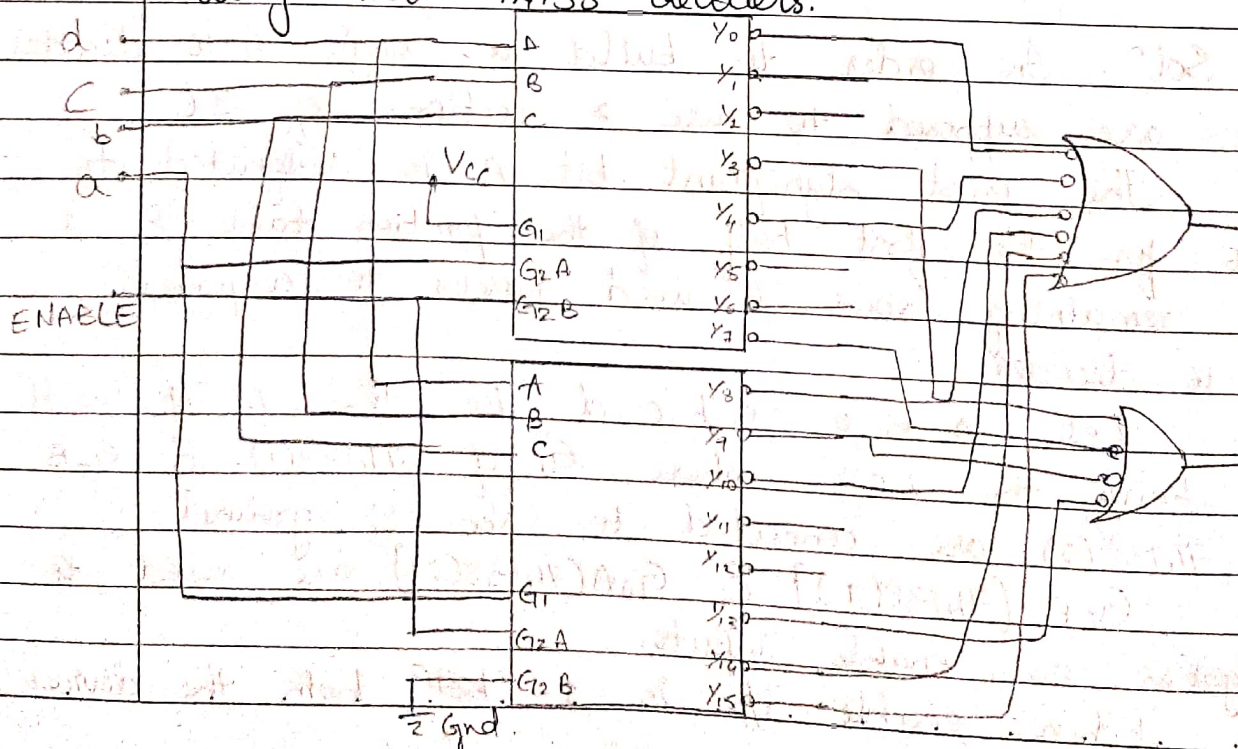
are disabled & the all the o/p are at 1.

When the enable i/p is zero, the devices are enabled observe that for the upper half of the four i/p table the MSP is zero. This enables 74138(1) & disables 74138(2).

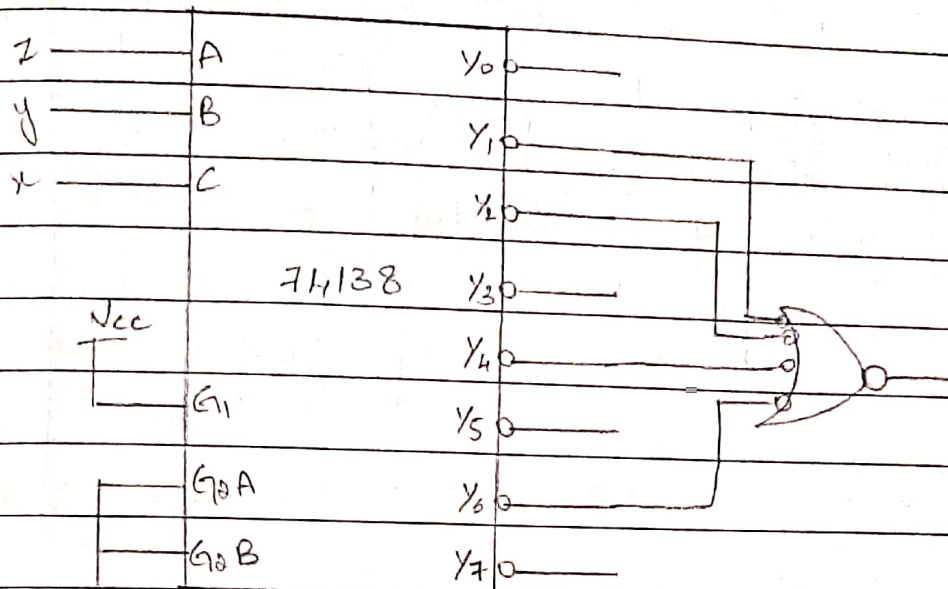
Hence for the i/p from 0000 to 0111 (0-7) $a = 0$ & one of the i/p from Y_0 to Y_7 of 74138(1) designated to $Z_0 - Z_7$ becomes 0 depending on the i/p code.

Thus for the i/p a, b, c, d taking the value from 1000 to 1111, $a = 1$ with this 74138(1) is disabled & 74138(2) is enable. and one of o/p from Y_0 to Y_7 of 74138(2) will be designated from Z_8 to Z_{15} becomes zero depending upon the i/p code, with this we can implement a single 4-16 decoder using 2 74138 IC's.

Eg: Implement the multiple function $f_1(a, b, c, d) = \Sigma(0, 4, 8, 10, 14, 15)$ & $f_2(a, b, c, d) = \Sigma(3, 7, 9, 13)$ using two 74138 decoders.



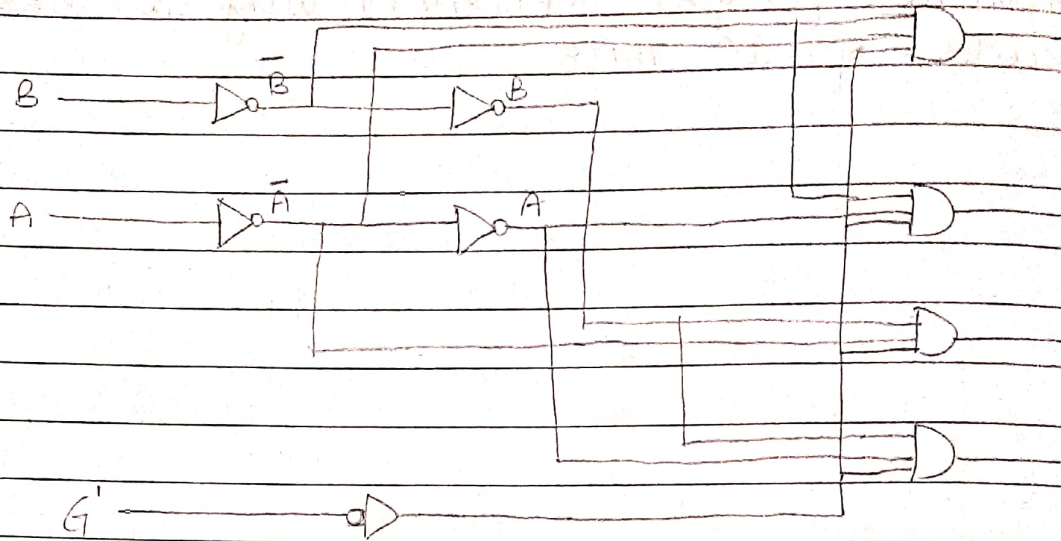
Implement $f(x, y, z) = \pi(1, 2, 4, 6)$ using a 3:8 decoder of IC 74138.



Now let us try to develop a 2:4 decoder with following specifications. (E, A, B, C) 3 = (Y0, Y1, Y2, Y3)

(i) Active-low enable input with active-high coded outputs.

i/p			o/p			
E	A	B	Y ₀	Y ₁	Y ₂	Y ₃
L	0	0	1	0	0	0
L	0	1	0	1	0	0
L	1	0	0	0	1	0
L	1	1	0	0	0	1



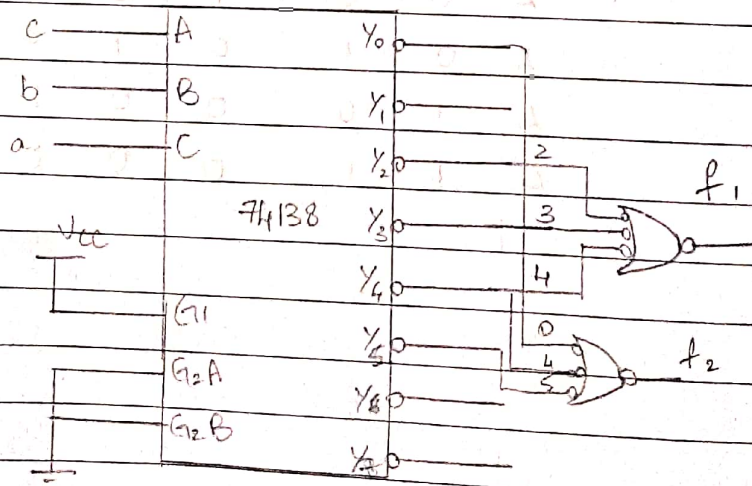
Implement the following function pairs using 74138 IC

1) $f_1(a, b, c) = \Sigma(0, 1, 5, 6, 7)$

$f_2(a, b, c) = \Sigma(1, 2, 3, 6, 7)$

Implementation should be done with minimum no. of gate inputs.

$\rightarrow \bar{f}_1 = \Sigma(2, 3, 4) \text{ \& } \bar{f}_2 = \Sigma(0, 4, 5)$

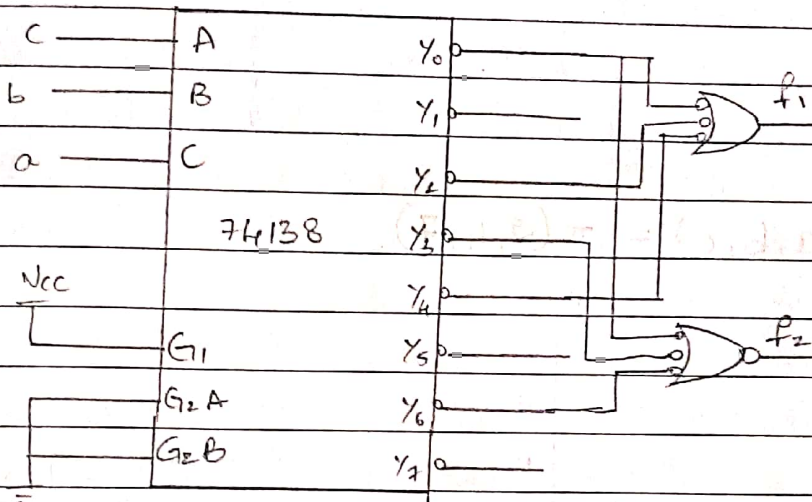


NOTE: In the above problem instead of taking the function pair f_1 & f_2 directly we have take the same in its complemented form.

The reason behind this scheme is to achieve the objective given in the question i.e., to implement using less no. of inputs.

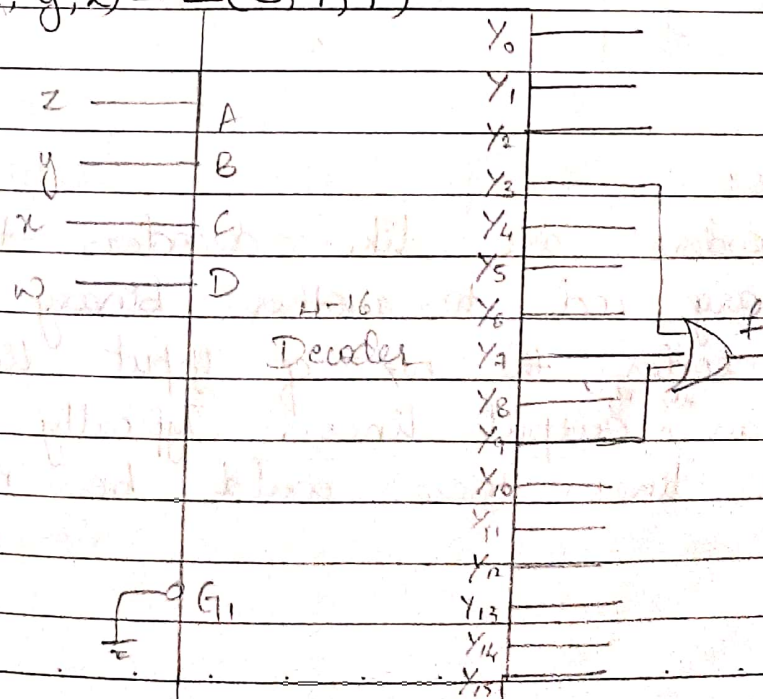
- 1) $f_1(a, b, c) = \Sigma(0, 2, 4)$ & $f_2(a, b, c) = \Sigma(1, 2, 4, 5, 7)$
same condition as above previous condition.

$f_1 = \Sigma(0, 2, 4)$ $3:8$
 $f_2 = \bar{f}_2 = \Sigma(0, 3, 6)$

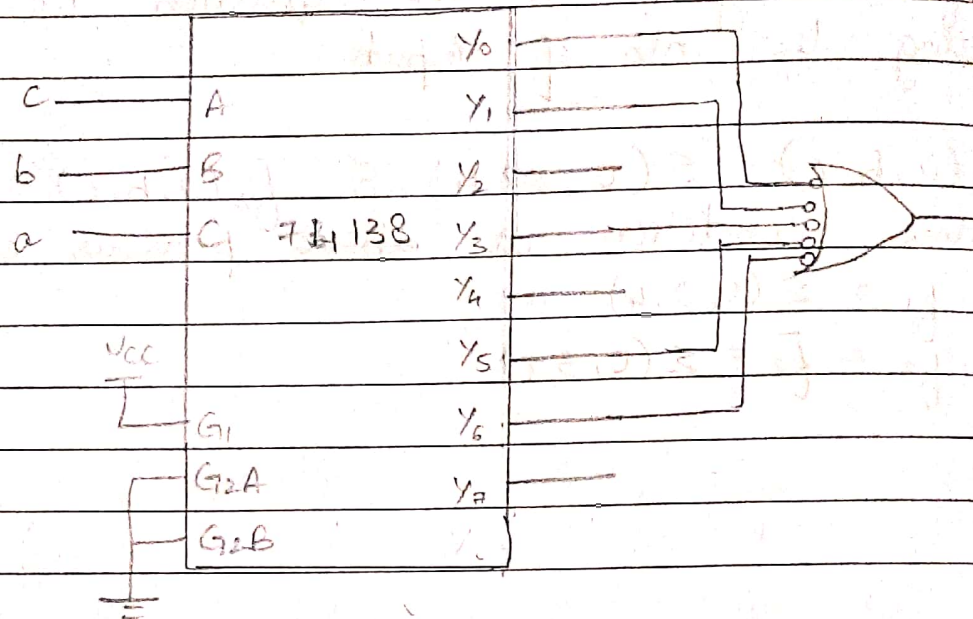


- 2) Implement the following in a suitable decoder in a active-low enable i/p & active-high o/p.

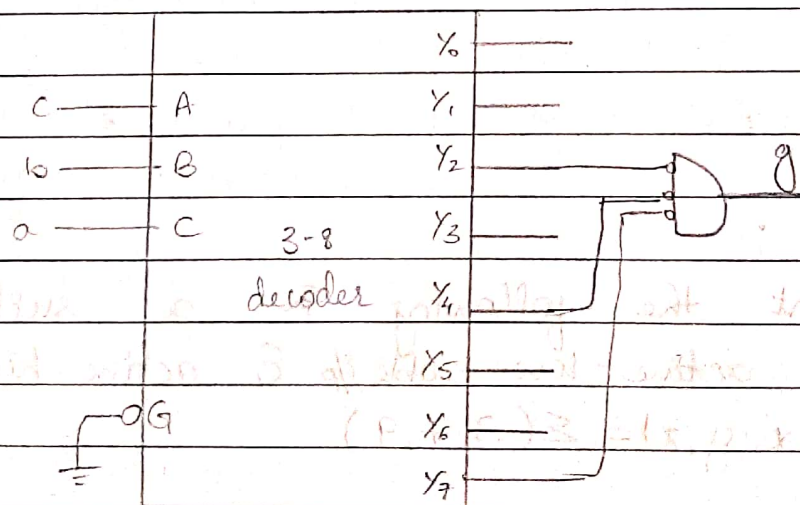
$f(w, x, y, z) = \Sigma(3, 7, 9)$



3) Implement $G(a, b, c) = \Sigma(0, 1, 3, 5, 6)$

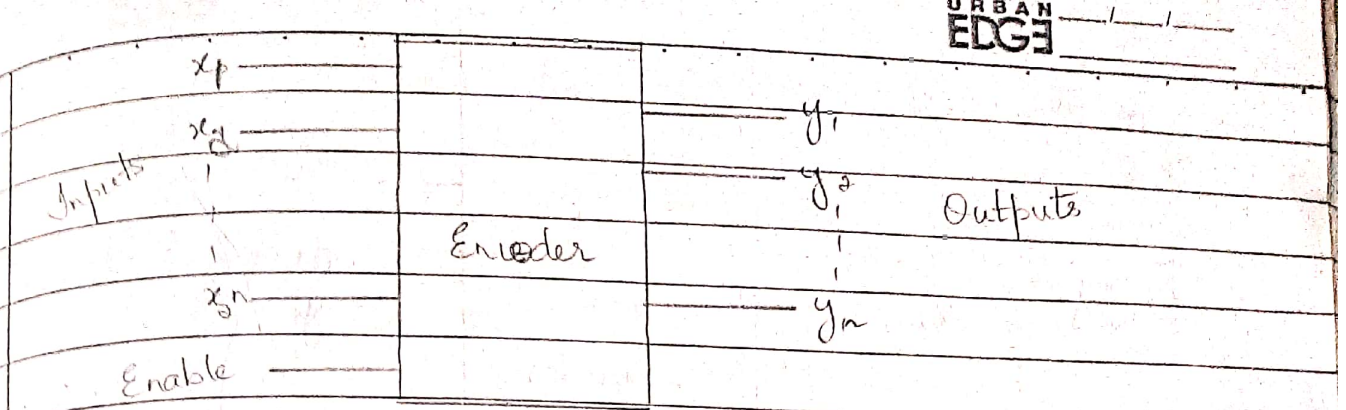


$$G(a, b, c) = \pi(2, 4, 7).$$



Encoders:

Encoders are like decoders that convert one binary code to another binary code. Generally in an encoder, the no. of input lines are much more than the output lines. Typically if there are 2^n input lines there could be n output lines.



A general encoder

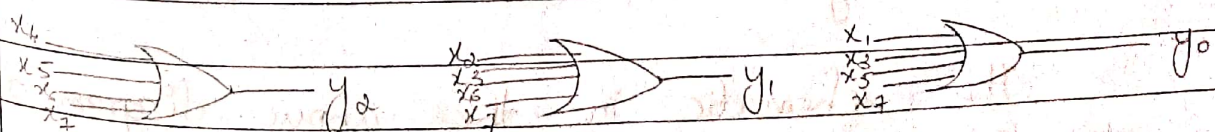
Inputs								Outputs			
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y_0	y_1	y_2	
1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	2
0	0	0	1	0	0	0	0	0	1	1	3
0	0	0	0	1	0	0	0	1	0	0	4
0	0	0	0	0	1	0	0	1	0	1	5
0	0	0	0	0	0	1	0	1	1	0	6
0	0	0	0	0	0	0	1	1	1	1	7

$$y_0 = x_4 + x_5 + x_6 + x_7$$

$$y_1 = x_2 + x_3 + x_6 + x_7$$

$$y_2 = x_1 + x_3 + x_5 + x_7$$

The above boolean expressions are the outputs which are written by looking into the truth table.



(a)

	x_0 ———	0		
	x_1 ———	1		
	x_2 ———	2		y_2
	x_3 ———	3		y_1
Inputs	x_4 ———	4	8-3	y_0
	x_5 ———	5	Encoder	
	x_6 ———	6		
	x_7 ———	7		

(b)

There are some problems associated with this implementation. It is possible that when more than one inputs are at logic one there may be an error in the output code.

for eg: If $x_3 = x_4 = 1$ then $y_2, y_1, y_0 = 1, 1, 1$

$$y_2 = x_4 + x_5 + x_6 + x_7$$

$$= 1 + 0 + 0 + 0$$

$$y_2 = 1$$

$$y_1 = x_2 + x_3 + x_6 + x_7$$

$$= 0 + 1 + 0 + 0$$

$$y_1 = 1$$

$$y_0 = x_1 + x_3 + x_5 + x_7$$

$$= 0 + 1 + 0 + 0$$

$$y_0 = 1$$

Whereas this output should have resulted only for $x_7 = 1$, the other problem is that the output is 000 when all the inputs are at logic zero, as well as when $x_0 = 1$.

The schematic in the above diagram does not distinguish b/w these two conditions of inputs.

These problems can be overcome from by priority encoder with a validity indicator. The truth table of a 8-3 priority encoder is as shown below.

Row	Inputs								Outputs			
no:	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y_2	y_1	y_0	Valid
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	1
2	x	1	0	0	0	0	0	0	0	0	1	1
3	x	x	1	0	0	0	0	0	0	1	0	1
4	x	x	x	1	0	0	0	0	0	1	1	1
5	x	x	x	x	1	0	0	0	1	0	0	1
6	x	x	x	x	x	1	0	0	1	0	1	1
7	x	x	x	x	x	x	1	0	1	1	0	1
8	x	x	x	x	x	x	x	1	1	1	1	1

Observe that in the table that the higher order bits have higher priority.

for eg: If x_3 is at logic 1 then irrespective of the logic values x_0 , x_1 & x_2 , the o/p code will be 0, 1, 1. The valid o/p being 1, indicates that the i/p code is valid [true]. With this the o/p's at row 0 & row 1 are now distinguished by the help of the logic level of the valid output.

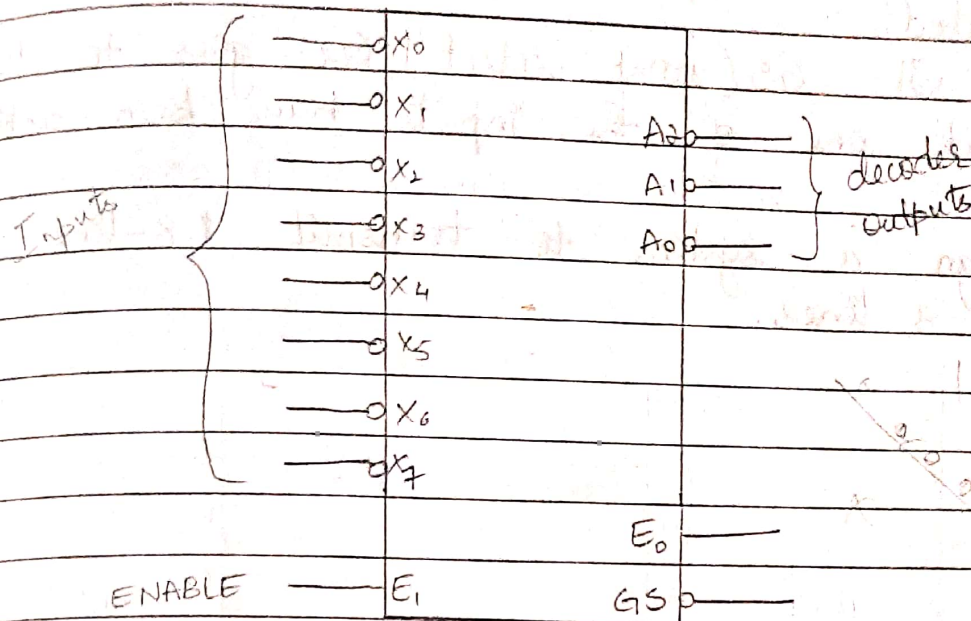
The truth table in the above is referred to as the condensed truth table as all don't care conditions of the i/p are not explicitly listed.

Problems:-

- 1) Write the condensed truth table for a 4 to 2 line priority encoder with a valid output where the highest priority is given to the highest bit position. All i/p with highest index E_i obtain the minimal sum expression for the output.

Cell no.	x_0	x_1	x_2	x_3	Y_1	Y_0	Valid
0	0	0	0	0	0	0	0
8	1	0	0	0	1	0	1
4, 12	x	1	0	0	x	1	1
2, 6, 10, 14	x	x	1	0	x	1	1
1, 3, 5, 7, 9	x	x	x	1	1	1	1
11, 13, 15	x	x	x	x	x	x	x

8 line - 3 line priority Encoder:



Inputs									Outputs				
E_1	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	A_2	A_1	A_0	GS	E_0
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1

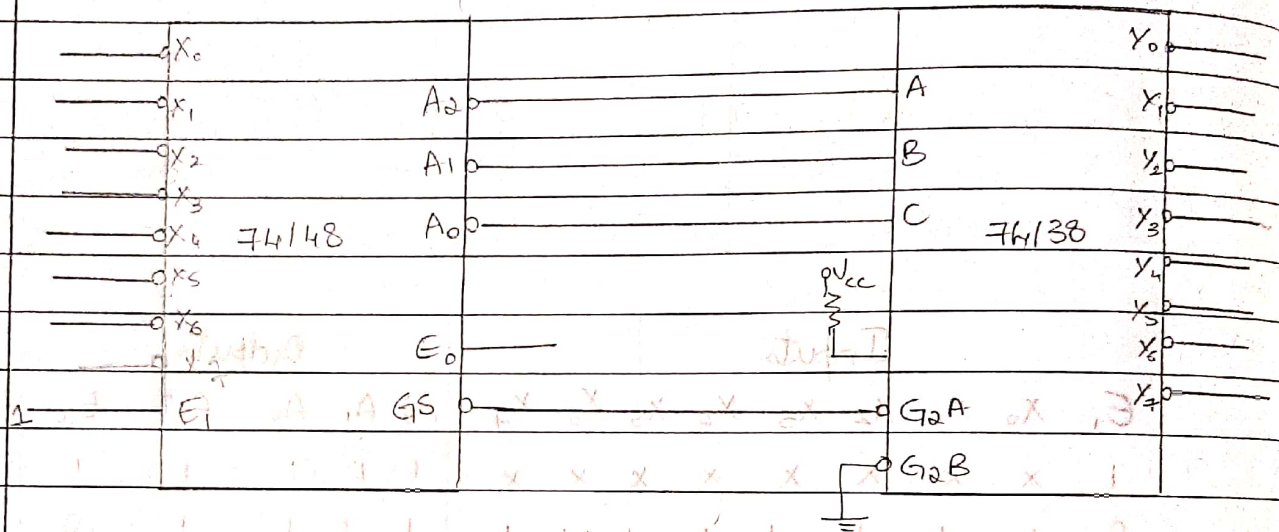
The 8:3 line priority encoder can be used to set priority 8 events E_1 it is available readily in the form of 74148.

Here when the enable encoder E_1 is active low[0], the o/p lines are high & the enable o/p E_0 is high. The enable output E_0 being low.

indicates none of the priority inputs have been activated.

The GS (group select) G_7 goes to low (0) to indicate one of the inputs have been activated.

- 1) Design a system¹ to transmit 8-line data over 3 lines.



- 2) Design a priority encoder for a system with 3 inputs with the middle bit with highest priority E_i it is encoded to 1, 0. The MSB with the next priority encode into 1, 1. while the LSB with least priority encode into 0, 1.

Cell no.	i/p			o/p	
	A	B	C	X	Z
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	1	0
3	0	1	1	1	1
4	1	0	0	1	0
5	1	0	1	1	1
6	1	1	0	1	0
7	1	1	1	1	1

URBAN
EDGE

k-map for Y:

		bc	00	01	11	10
0	0		0 ⁰	0 ¹	1 ³	1 ²
1	1		1 ⁴	1 ⁵	1 ⁷	1 ⁶

$$Y = G = a + b$$

k-map for Z:

		bc	00	01	11	10
0	0		0 ⁰	1 ¹	0 ³	0 ²
1	1		1 ⁴	1 ⁵	0 ⁷	0 ⁶

$$Z \Rightarrow G = A\bar{B} + \bar{B}C$$

for Y: $\bar{a}bc + abc + \bar{a}b\bar{c} + ab\bar{c}$

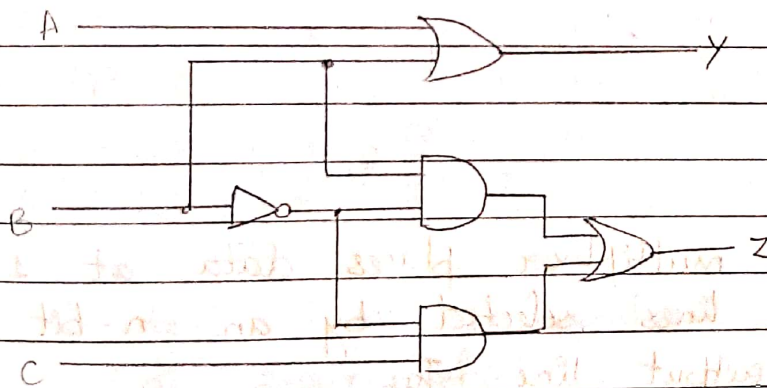
$$= bc + b\bar{c}$$

$$= b$$

$$= a\bar{b}\bar{c} + a\bar{b}c + abc + ab\bar{c}$$

$$= a\bar{b} + ab$$

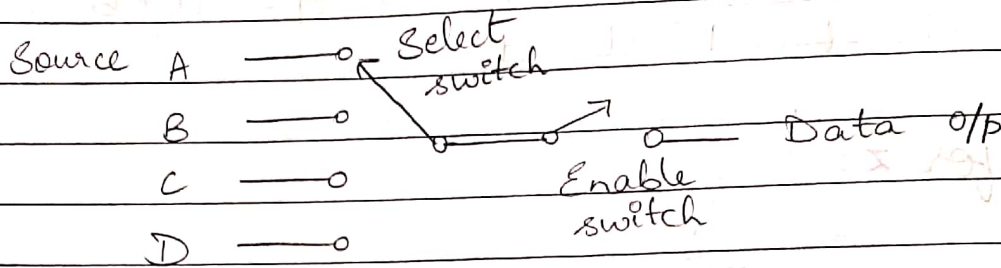
$$= a$$



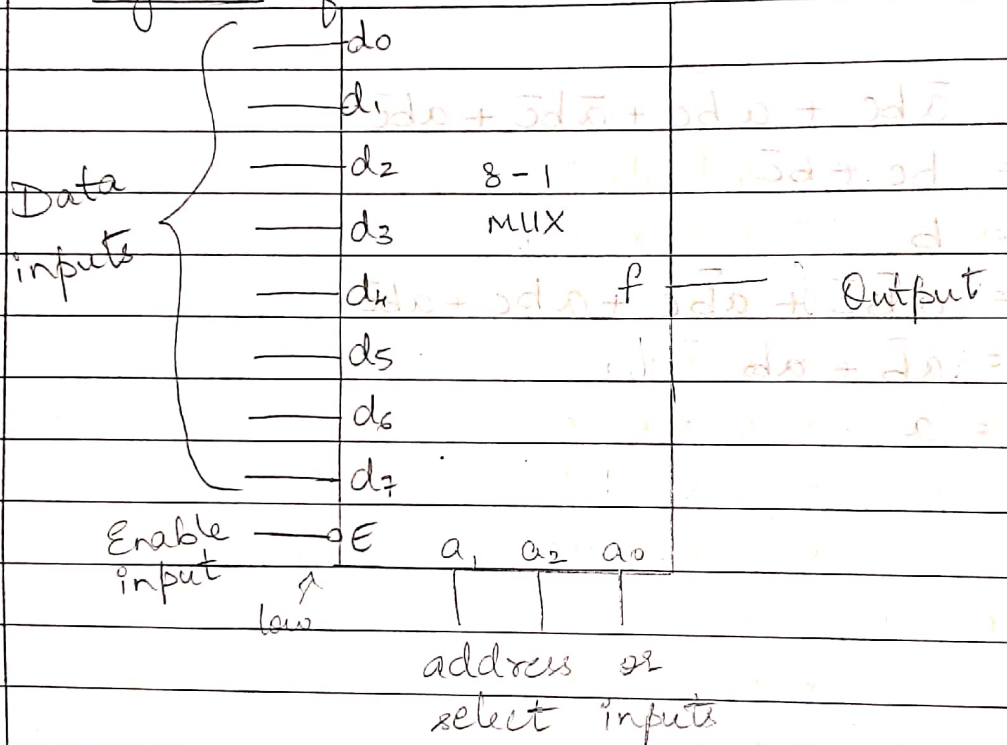
3) Design a 5 to 3 line priority encoder with following specification.

- (i) bit 3 with highest priority & it is encoded to 111.
- (ii) next highest priority to bit 1 is 100
- (iii) bit 4 $\rightarrow$ 001
- bit 0 $\rightarrow$ 011
- bit 2 $\rightarrow$ 101 (least priority)

Multiplexers:



Symbol of 8-1 MUX:

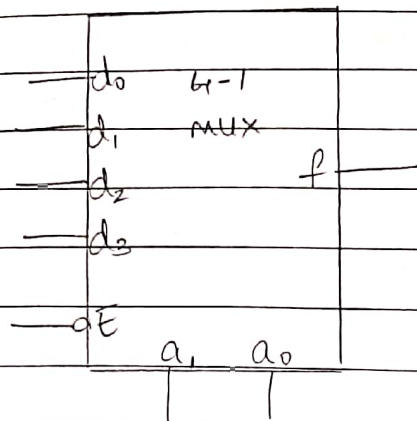


A multiplexer places data at 1 of its 2^n input lines selected by an n -bit address on its output line. Thus

Just as in case of decoders, multiplexers also have enable inputs as well as select line inputs.

The symbol of a 8-1 MUX is as shown above.

Eg: 4-1 Multiplexer:



Row no.	E	a ₁	a ₀	d ₀	d ₁	d ₂	d ₃	f
0	1	x	x	x	x	x	x	0
1	0	0	0	0	x	x	x	0
2	0	0	0	1	x	x	x	1
3	0	0	1	x	0	x	x	0
4	0	0	1	x	1	x	x	1
5	0	1	0	x	x	0	x	0
6	0	1	0	x	x	1	x	1
7	0	1	1	x	x	x	0	0
8	0	1	1	x	x	x	1	1

The truth table & symbol diagram of 4-1 MUX is as shown above.

We can see in 'row 0', when $E=1$ irrespective of other inputs f goes to zero. & when $E=0$, the data at the address or selected input appears at the output.

for eg: In row-3, the select inputs are 0, 1 which selects the data line d_1 & the data on d_1 appears at the output.

If $d_1=0$, $f=0$, if $d_1=1$ then $f=1$. [row-4]
This functional description can be represented in

a functional table shown below.

E	a_1	a_0	f
1	x	x	0
0	0	0	d_0
0	0	1	d_1
0	1	0	d_2
0	1	1	d_3

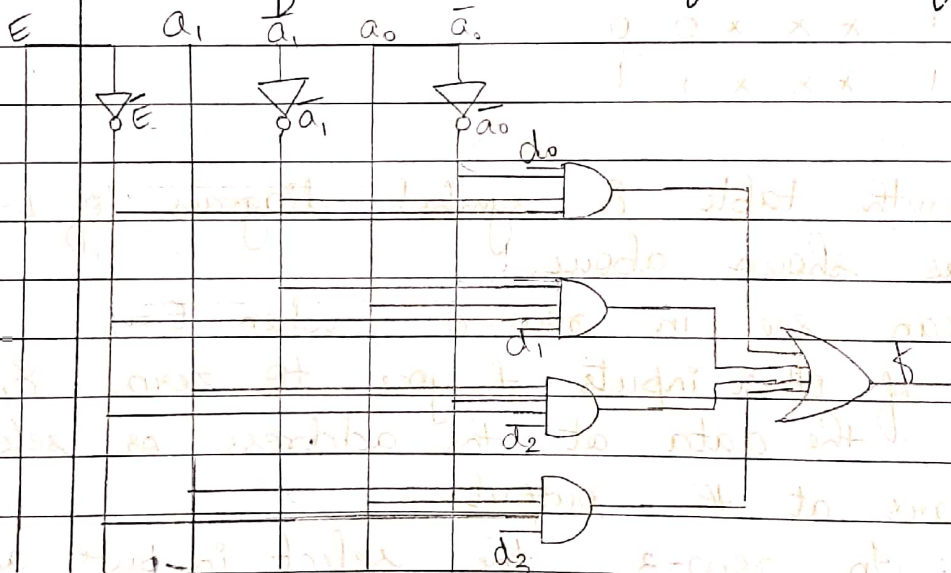
With this we can write an expressions for the output as follows:

$$f = (d_0 \bar{a}_1 \bar{a}_0 + d_1 \bar{a}_1 a_0 + d_2 a_1 \bar{a}_0 + d_3 a_1 a_0) \bar{E}$$

Suppose the MUX is enabled (when $E = 0$) & the select lines are a_1, a_0 & that is equal to 0, 1, then the above eqⁿ becomes.

$$f = d_1$$

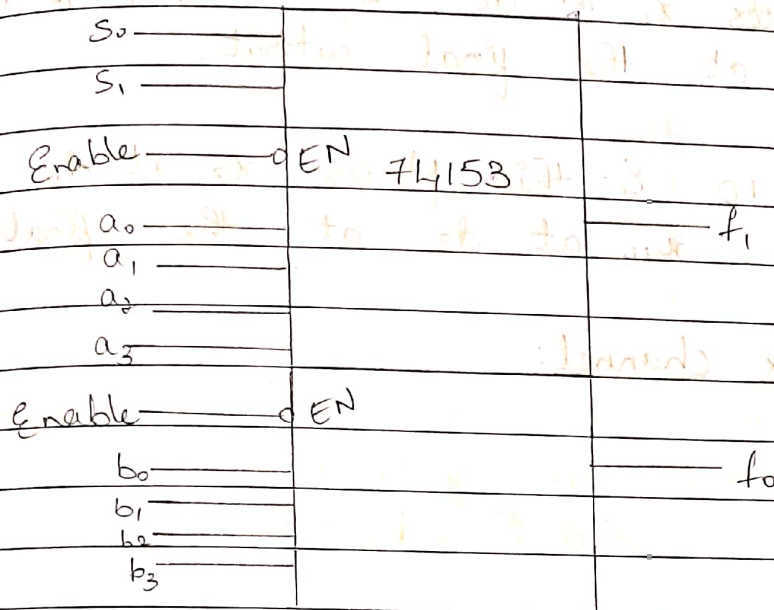
If $d_1 = 0$, then $f = 0$ & if $d_1 = 1$, then $f = 1$.



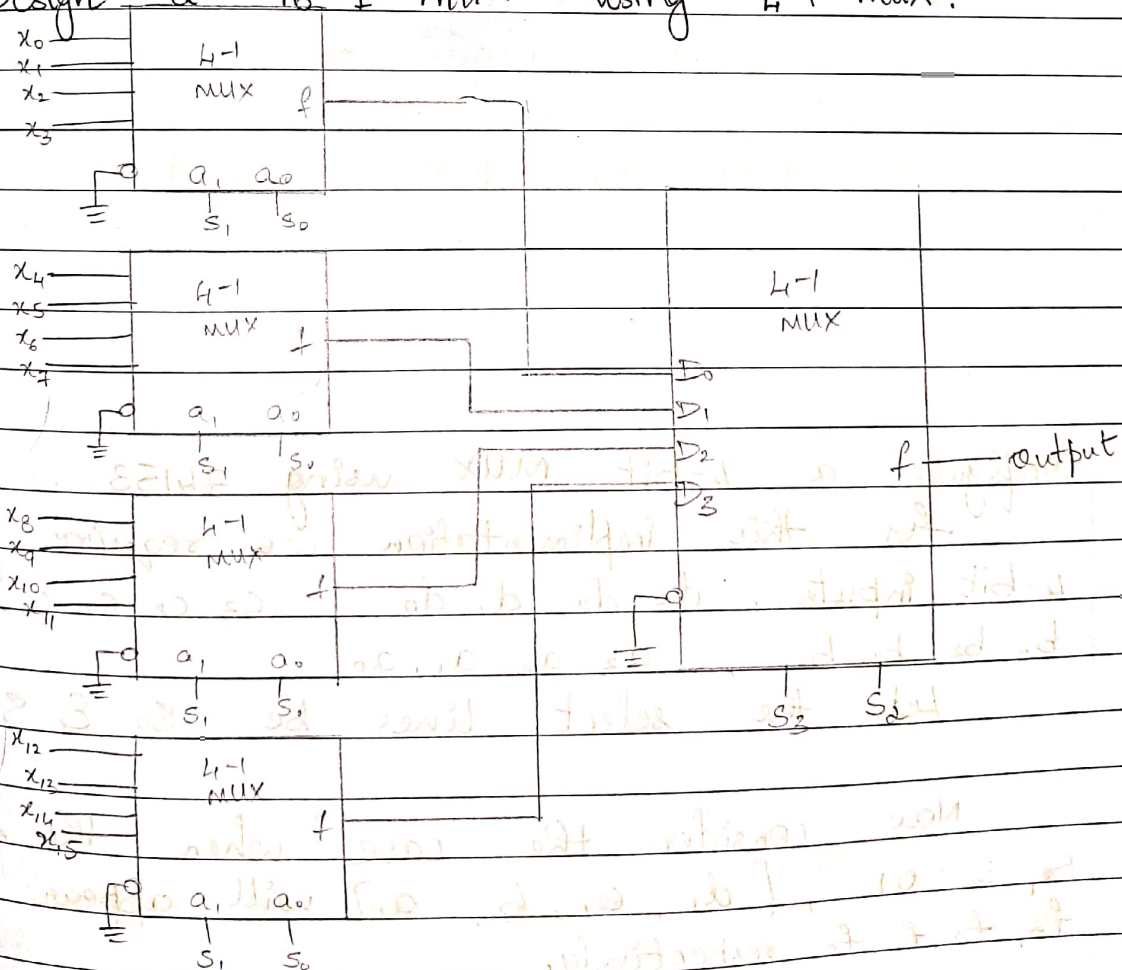
Several multiplexers can be combined to form a large logic circuit. for eg: A dual 4-1 MUX is employed to workout 8-1 MUX.

URBAN
EDGE

Such a dual 4-1 MUX is readily available in the form of an IC 74153.



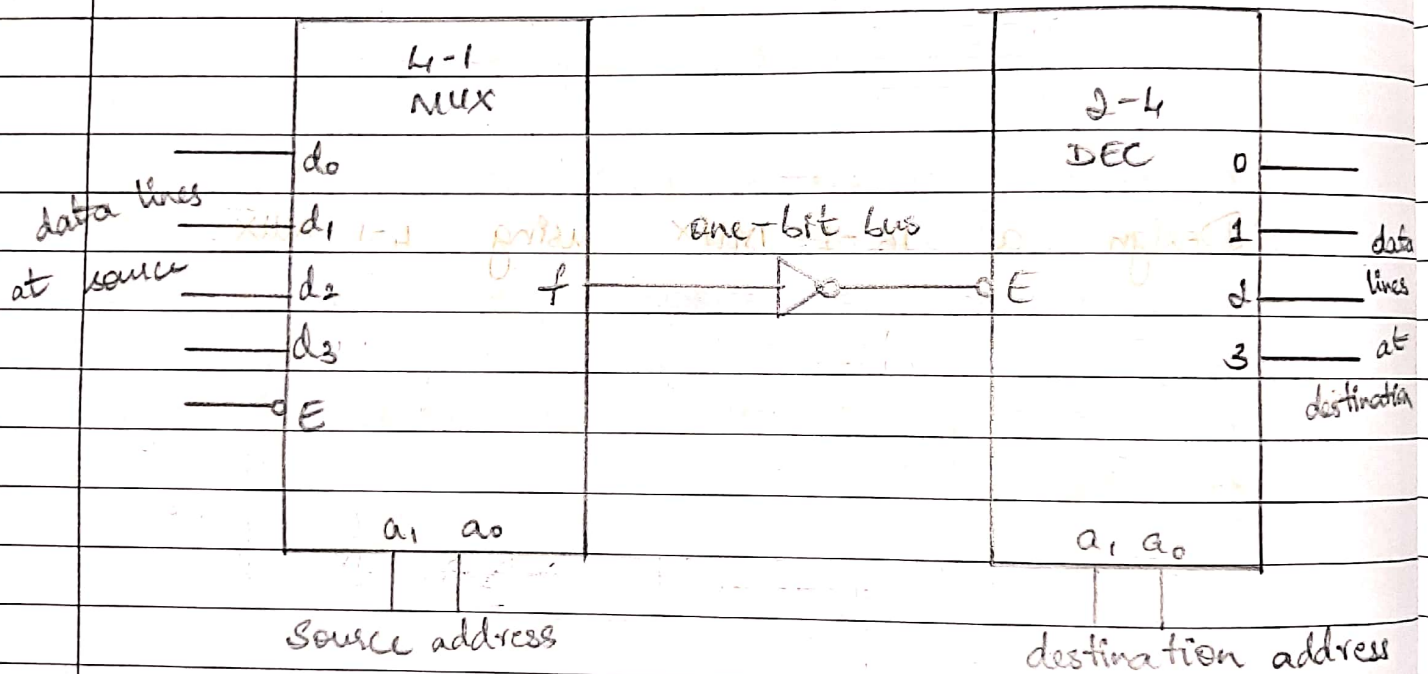
Design a 16-1 MUX using 4-1 MUX.



Let us say that the select line address $S_0, S_1, S_2, S_3 = 0011$ then this implies $S_1, S_0 = 01$. This places x_1 at d_0 , x_5 at d_1 , x_9 at d_2 , x_{13} at d_3 at the final output.

If $S_1, S_0 = 10$ this places x_2 at d_0 , x_6 at d_1 , x_{10} at d_2 , x_{14} at d_3 at the final output.

MUX - Demux channel:

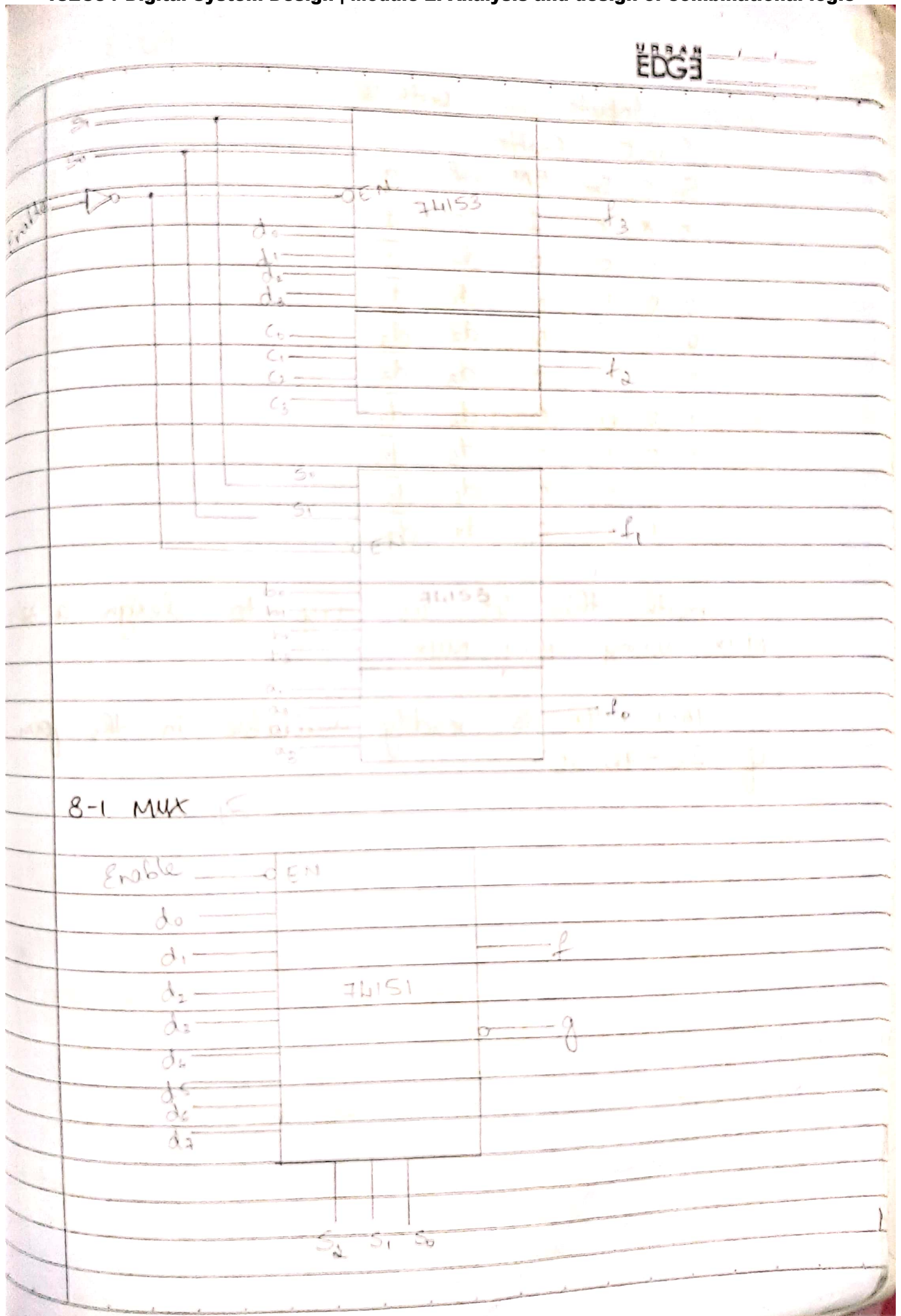


Configure a 4-bit MUX using 74153.

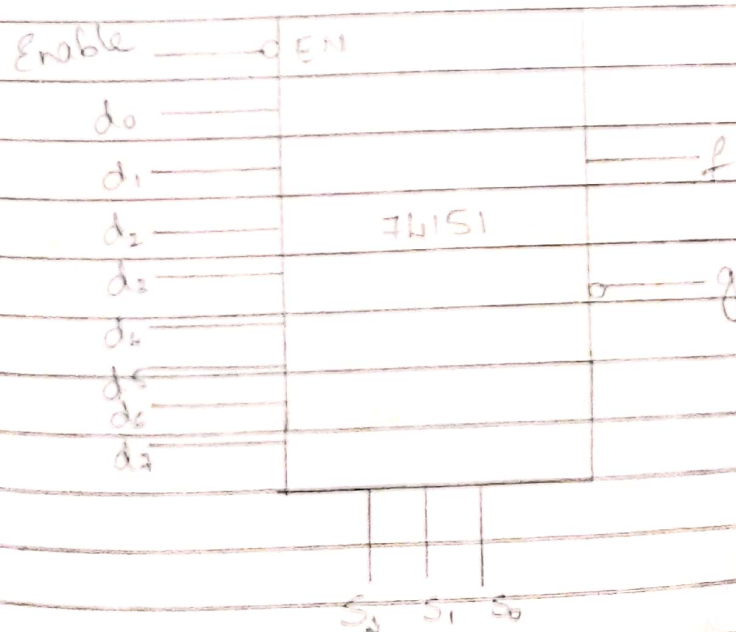
For this implementation we require four 4-bit inputs. (d_3, d_2, d_1, d_0 , c_3, c_2, c_1, c_0 , b_3, b_2, b_1, b_0 , a_3, a_2, a_1, a_0) - i/p's

Let the select lines be S_0 & S_1 .

Now consider the case when the select line $S_1, S_0 = 01$. [d_1, c_1, b_1, a_1] will appear on f_3, f_2, f_1, f_0 respectively.



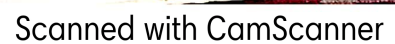
8-1 MUX



Inputs				Outputs	
Select			Enable	f	g
S_2	S_1	S_0	EN		
x	x	x	1	0	1
0	0	0	0	d_0	$\bar{d}_0$
0	0	1	0	d_1	$\bar{d}_1$
0	1	0	0	d_2	$\bar{d}_2$
0	1	1	0	d_3	$\bar{d}_3$
1	0	0	0	d_4	$\bar{d}_4$
1	0	1	0	d_5	$\bar{d}_5$
1	1	0	0	d_6	$\bar{d}_6$
1	1	1	0	d_7	$\bar{d}_7$

With this let us try to design a 32-1 MUX using 16-1 MUX.

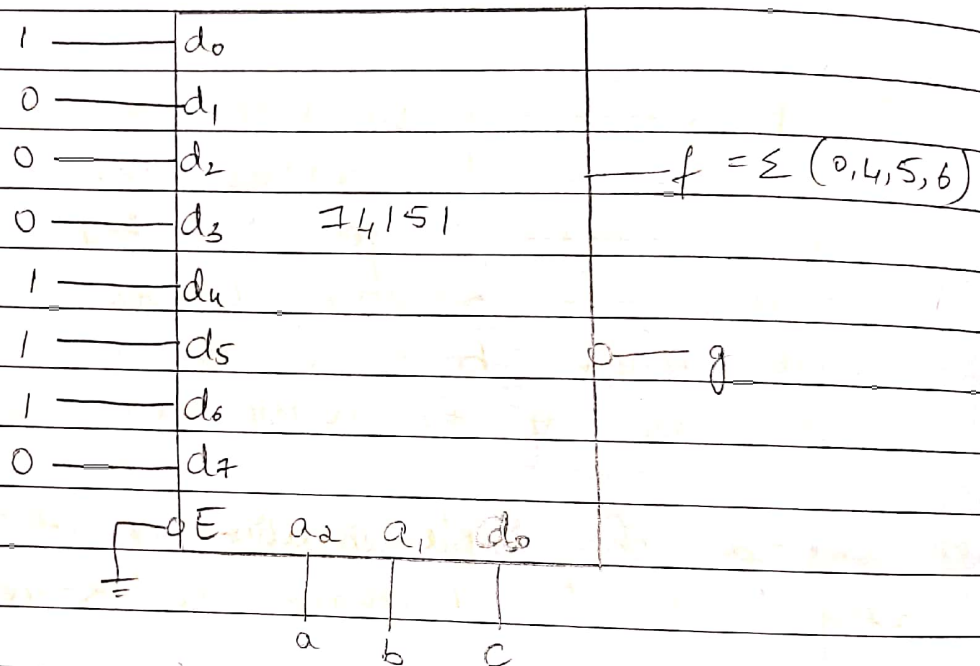
16-1 MUX is readily available in the form of IC-74150.



Logic design using MUX:

1) Implement the following using an 8-1 MUX.
 $f(a, b, c) = \Sigma(0, 4, 5, 6)$

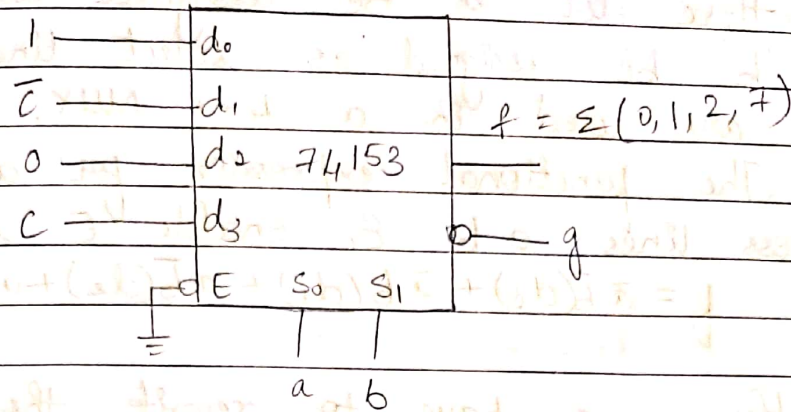
→ Let $d_0 = d_4 = d_5 = d_6 = 1$
 $d_1 = d_2 = d_3 = d_7 = 0$



2) Implement the following function using a 4-1 MUX
 $f(a, b, c) = \Sigma(0, 1, 2, 7)$

→ first let us write $f(a, b, c)$ in terms of algebraic function.

$$\begin{aligned}
 f(a, b, c) &= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + abc \\
 &= \bar{a}\bar{b}(c + \bar{c}) + \bar{a}b(\bar{c}) + ab(c) \\
 &= \bar{a}\bar{b}(1) + \bar{a}b(\bar{c}) + a\bar{b}(0) + ab(c) \\
 &= \bar{a}\bar{b}(d_0) + \bar{a}b(d_1) + a\bar{b}(d_2) + ab(d_3)
 \end{aligned}$$



The procedure followed in the previous example works fine if the address lines are assigned variables in order to form starting from MSB position.

Suppose the situation changes then we would like to assign b, E, c as the address lines or select lines of the multiplexers.

- 3) Consider the implementation of $f = \Sigma(1, 4, 5, 7)$ along with its truth table & k-map.

Cell no.	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

bc		00	01	11	10
a	0	0 ⁰	1 ¹	0 ³	0 ²
1	1 ⁴	1 ⁵	1 ⁷	0 ⁶	

Here let 'a' be assigned as select line 'a' & 'b' be assigned as select line 'a'. further they are used in a 4-1 MUX.

The functional expression for a 4-1 MUX with address lines a b & enable E set to zero then $f = \bar{a}\bar{b}(d_0) + \bar{a}b(d_1) + a\bar{b}(d_2) + ab(d_3)$.

Here we have to rewrite the k-map as shown below.

	00	01	11	10
a ↓ a	0	1	3	2
	d ₀		d ₁	
	4	5	7	6
	d ₂		d ₃	

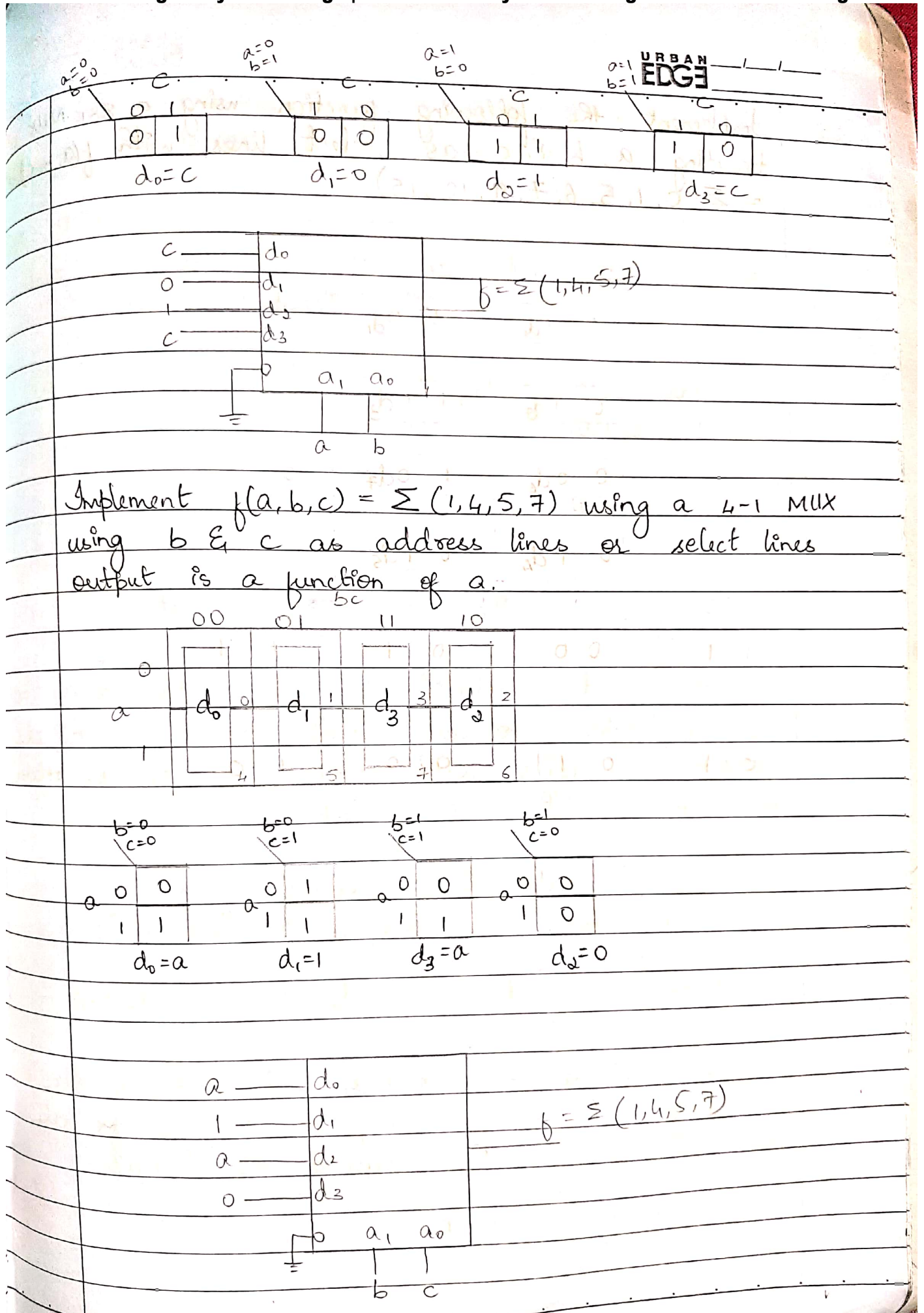
Here d₀ relates to cells with a=0, b=0 & d₁ relates to cells with a=0, b=1 & d₂ relates to cells with a=1, b=0 & d₃ relates to cells with a=1, b=1.

Address & data lines mapping:-

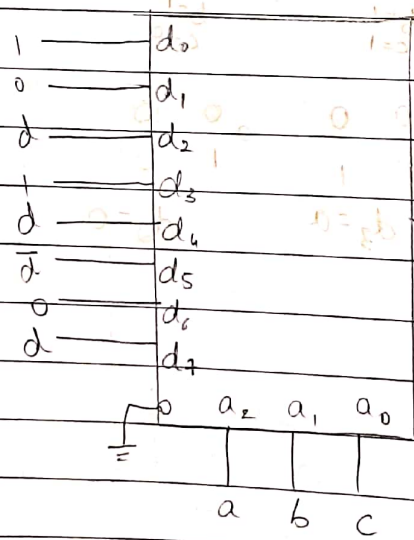
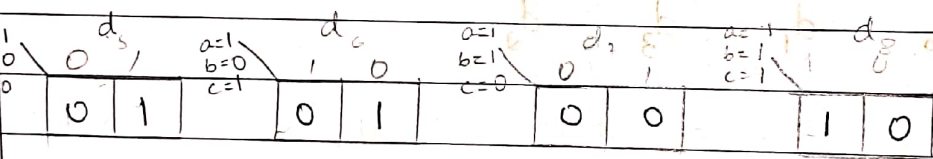
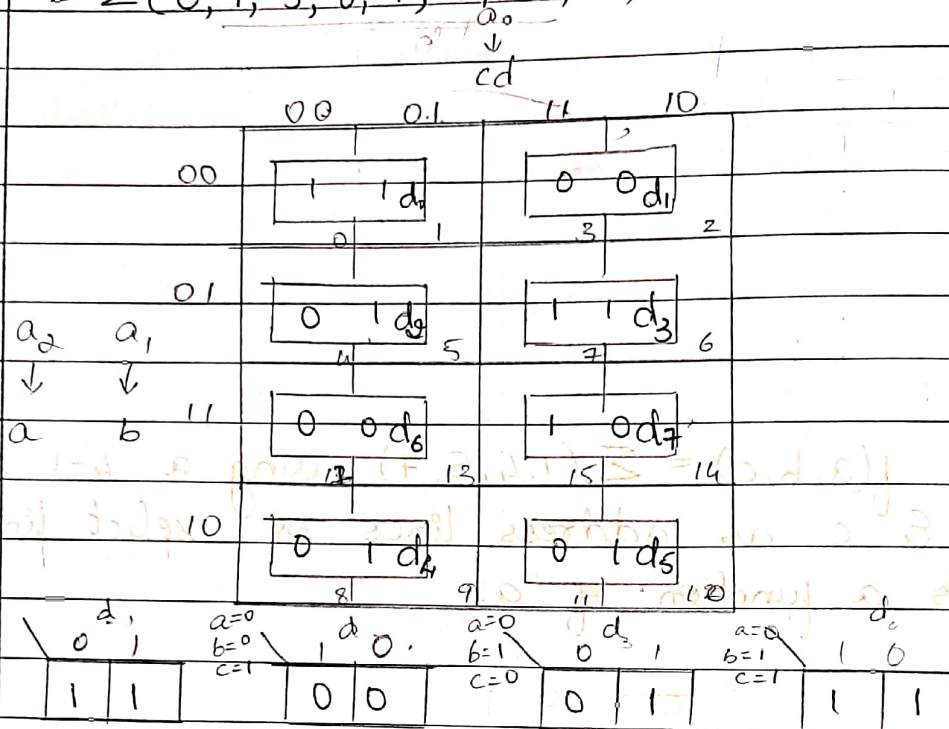
$a=0$ $b=0$	0	1	$a=0$ $b=1$	3	2	$a=1$ $b=0$	4	5	$a=1$ $b=1$	7	6
d ₀ -map		d ₁ -map		d ₂ -map		d ₃ -map					

By applying the k-map which is in 8-cells is being reduced to group of 4 cells for sake of simplification by referring the k-map.

	00	01	11	10
a	0	1	0	0
	1	1	1	0

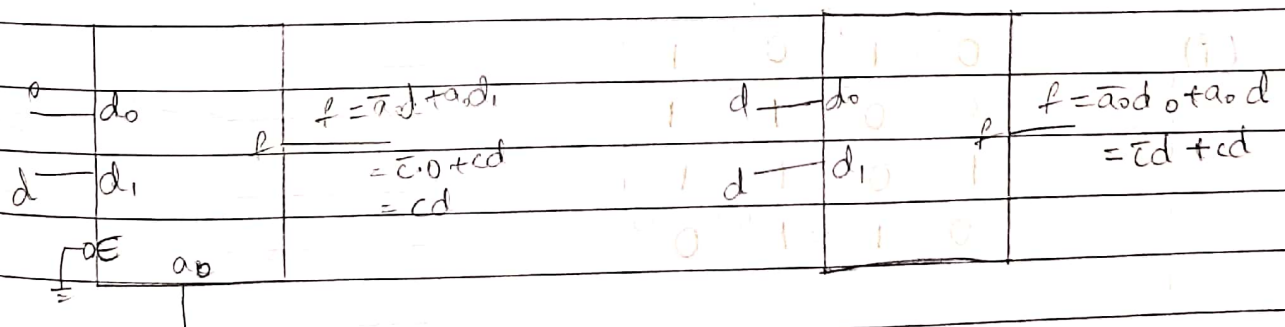
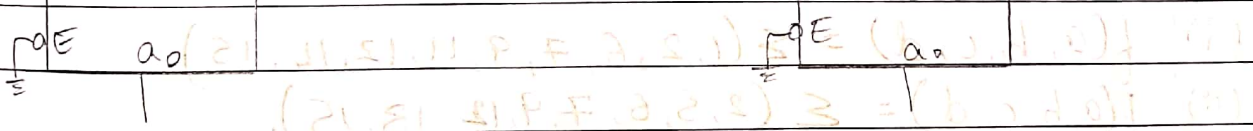
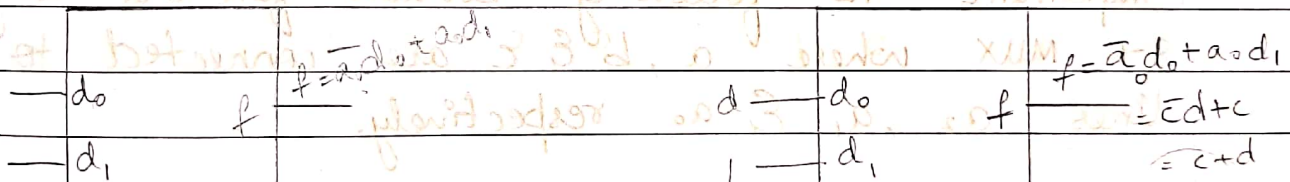
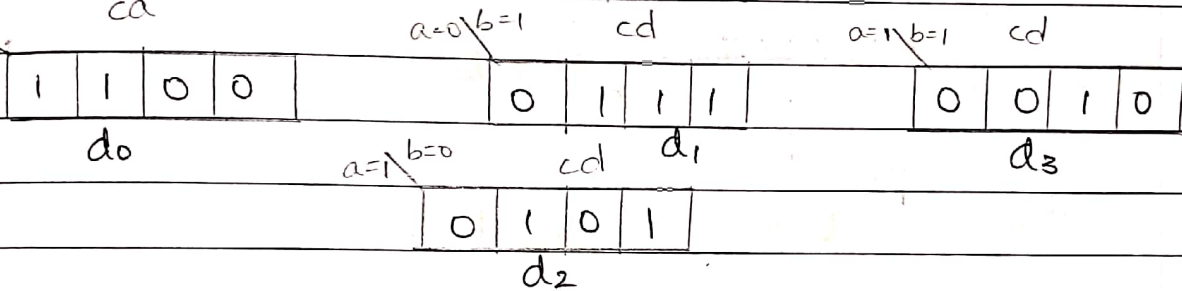
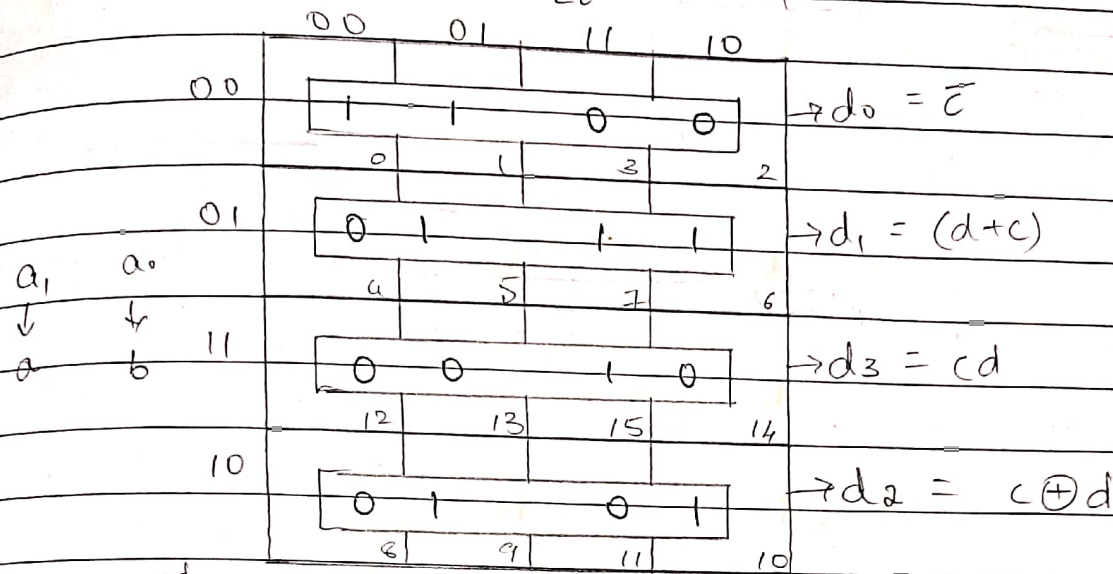


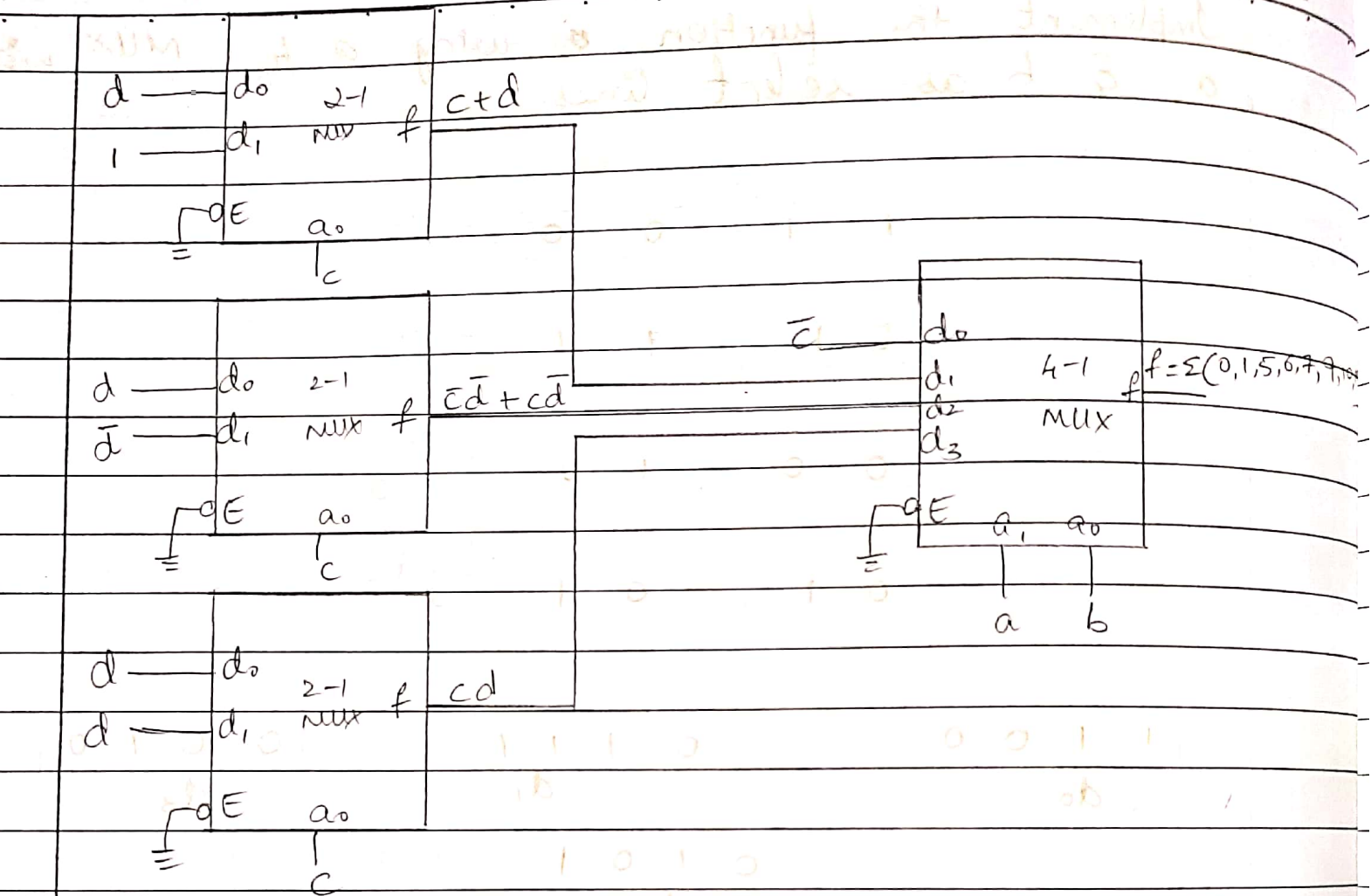
Implement the following function using a 8-1 Mux treating a, b & c as select lines with $f(a, b, c, d) = \Sigma(0, 1, 5, 6, 7, 9, 10, 15)$



$$f = \Sigma(0, 1, 5, 6, 7, 9, 10, 15)$$

Implement the function f using a 4:1 MUX with a & b as select lines. $f = \bar{c}d + cd$





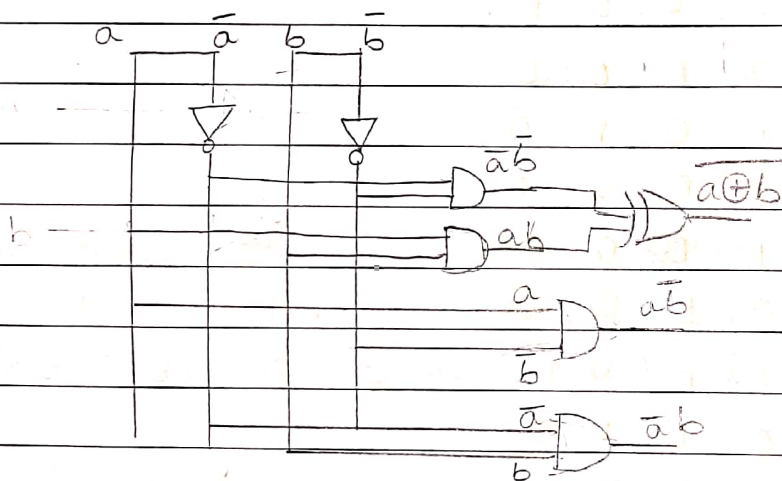
Implement the following boolean functions using an 8-1 MUX where a, b & c are connected to address lines a_2, a_1 & a_0 respectively.

- (i) $f(a, b, c, d) = \sum(1, 2, 6, 7, 9, 11, 12, 14, 15)$
- (ii) $f(a, b, c, d) = \sum_{cd} (2, 5, 6, 7, 9, 12, 13, 15)$

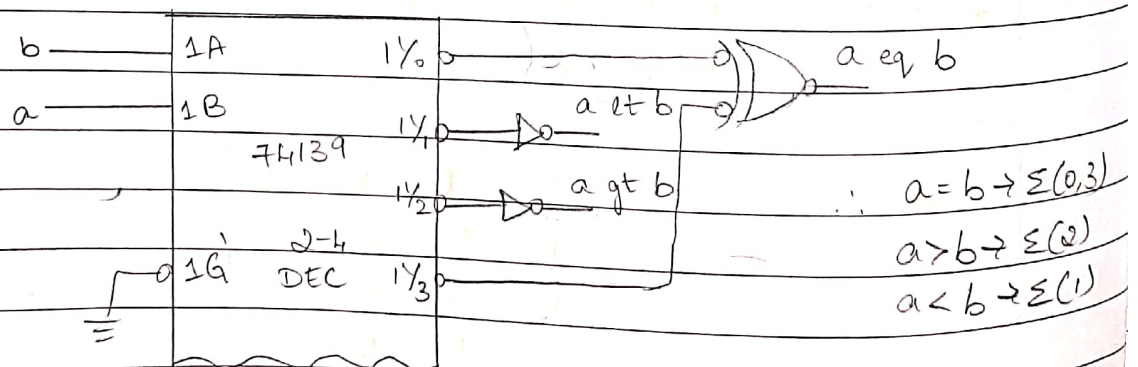
Comparators:-

Comparators are designed to compare the magnitude of 2 binary no. & to indicate whether 1 is greater than other no. or 1 is less than other no. (or) 1 is equal to the other no.

i/p's		o/p's			
a	b	a = b	a > b	a < b	
0	0	1	0	0	$a = b = \bar{a}\bar{b} + ab = \overline{a \oplus b}$
0	1	0	0	1	$a > b = a\bar{b}$
1	0	0	1	0	$a < b = \bar{a}b$
1	1	1	0	0	



Implement a 1-bit comparator using a 2-4 decoder, IC 74139 wst above problem.



Two bit comparator:

a_1 →

a_0 →

b_1 →

b_0 →

2-bit
Comparator

$z = a < b$

$y = a = b$

$x = a > b$

→ x

→ y

→ z

Cell no.	a_1	a_0	b_1	b_0	x	y	z
0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	1	0	0
5	0	1	0	1	0	1	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	1	0	0	1
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	1	0

$x =$

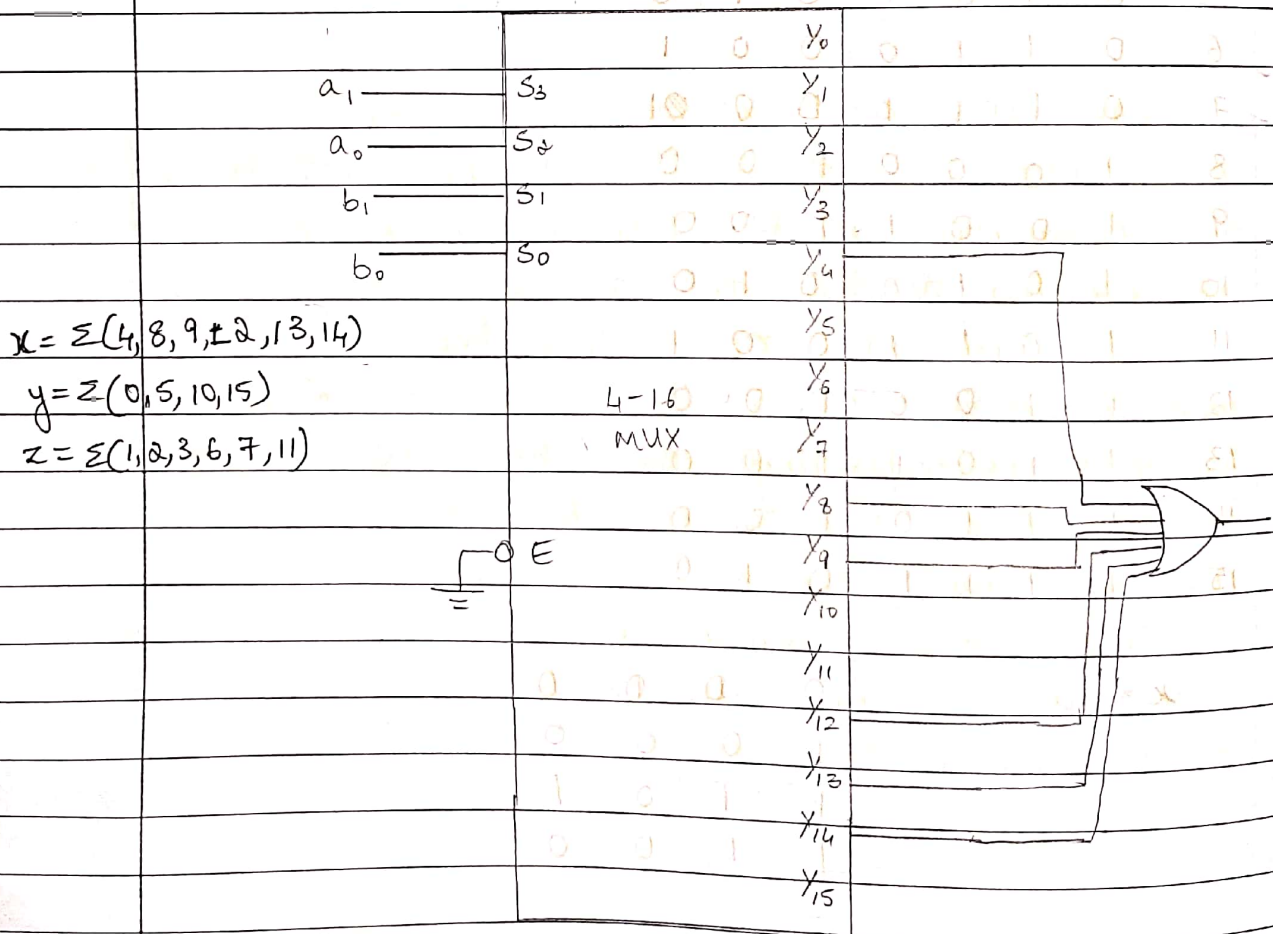
$y =$

	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$y =$		00	01	11	10
	00	1 ⁰	0 ¹	0 ³	0 ²
	01	0 ⁴	1 ⁵	0 ⁷	0 ⁶
	11	0 ¹²	0 ¹³	1 ¹⁵	0 ¹⁴
	10	0 ⁸	0 ⁹	0 ¹¹	1 ¹⁰

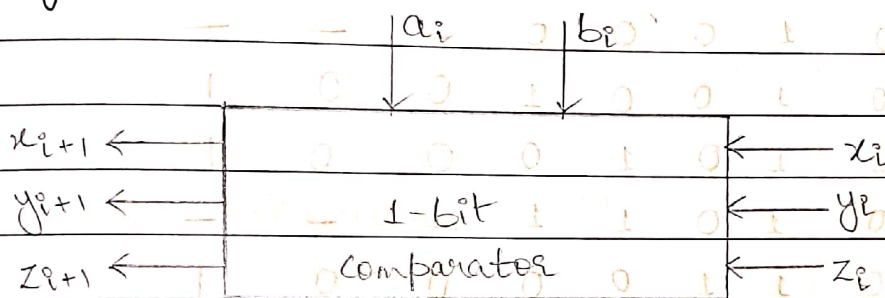
					$b_1 b_0$	
	$Z =$		00	01	11	10
		00	0 ⁰	1 ¹	1 ³	1 ²
a_1	01	0 ⁴	0 ⁵	1 ⁷	1 ⁶	
	11	0 ¹²	0 ¹³	0 ¹⁵	0 ¹⁴	
	10	0 ⁸	0 ⁹	1 ¹¹	0 ¹⁰	

Implement a 2-bit comparator using a 4-16 decoder for 2-bit comparator.



Iterative comparator:

Most often there is a requirement to compare two n bit, 8 bit or higher bit binary numbers. It is worthwhile to look at 1-bit comparator with inputs & outputs which can be used to cascade several of these to configure multibit comparators. The block diagram of a 1-bit comparator at the i^{th} stage which is cascaded & its truth table is:



Fig(a): Block diagram.

In row zero, $x_i = y_i = z_i = 0$, this will not occur coz one of them will be set to 1 by the comparator at the previous stage so the o/p corresponding to the row can be simply don't cares.

In row 1, $z_i = 1$ this implies that in the previous stage a_i was less than b_i so $z_{i+1} = 1$. In that corresponding row. That implies that in the previous stage a_{i-1} is less than b_{i-1} in other words, a_{i-1} must have been zero & b_{i-1} must have been 1.

Under these circumstances, in row 2, $y_i = 1$ this implies a_{i-1} & b_{i-1} must have been equal. In any row if $a_i > b_i$ then $x_{i+1} = 1$.

18EC34 Digital System Design | Module 2: Analysis and design of combinational logic

	cell no.	a_i	b_i	x_i	y_i	z_i	x_{i+1}	y_{i+1}	z_{i+1}
	0 { 0	0	0	0	0	0	—	—	—
	1 { 1	0	0	0	0	1	0	0	1
	2 { 2	0	0	0	1	0	0	1	0
	3 { 3	0	0	0	1	1	—	—	—
	4 { 4	0	0	1	0	0	1	0	0
	5 { 5	0	0	1	0	1	—	—	—
	6 { 6	0	0	1	1	0	—	—	—
	7 { 7	0	0	1	1	1	—	—	—
	8 { 8	0	1	0	0	0	—	—	—
	9 { 9	0	1	0	0	1	0	0	1
	10 { 10	0	1	0	1	0	0	0	1
	11 { 11	0	1	0	1	1	—	—	—
	12 { 12	0	1	1	0	0	0	0	1
	13 { 13	0	1	1	0	1	—	—	—
	14 { 14	0	1	1	1	0	—	—	—
	15 { 15	0	1	1	1	1	—	—	—
	16 { 16	1	0	0	0	0	—	—	—
	17 { 17	1	0	0	0	1	1	0	0
	18 { 18	1	0	0	1	0	1	0	0
	19 { 19	1	0	0	1	1	—	—	—
	20 { 20	1	0	1	0	0	1	0	0
	21 { 21	1	0	1	0	1	—	—	—
	22 { 22	1	0	1	1	0	—	—	—
	23 { 23	1	0	1	1	1	—	—	—
	24 { 24	1	1	0	0	0	—	—	—
	25 { 25	1	1	0	0	1	0	0	1
	26 { 26	1	1	0	1	0	0	1	0
	27 { 27	1	1	0	1	1	—	—	—
	28 { 28	1	1	1	0	0	1	0	0
	29 { 29	1	1	1	0	1	—	—	—
	30 { 30	1	1	1	1	0	—	—	—
	31 { 31	1	1	1	1	1	—	—	—

K-map for x_i

$a_i b_i$

	00	01	11	10
00	0 ⁰	0 ¹	— ³	1 ²
01	0 ⁴	0 ⁵	— ⁷	0 ⁶
11	0 ¹²	0 ¹³	— ¹⁵	1 ¹⁴
10	1 ⁸	1 ⁹	— ¹¹	1 ¹⁰

$x_i y_i$

K-map for y_i

$a_i b_i$

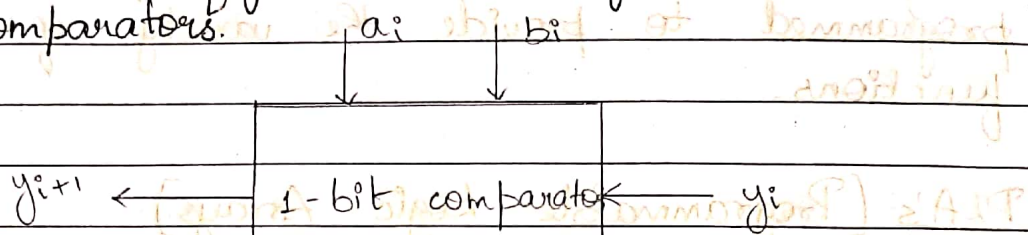
	00	01	11	10
00	0 ⁰	1 ¹	— ³	0 ²
01	0 ⁴	0 ⁵	— ⁷	0 ⁶
11	0 ¹²	1 ¹³	— ¹⁵	0 ¹⁴
10	0 ⁸	0 ⁹	— ¹¹	0 ¹⁰

K-map for z_i

$a_i b_i$

	00	01	11	10
00	0 ⁰	0 ¹ (0 ¹) z_i	— ³	0 ² (0 ²) z_i
01	1 ⁴	1 ⁵	— ⁷	1 ⁶
11	0 ¹²	0 ¹³ (0 ¹³) z_i	— ¹⁵	0 ¹⁴ (0 ¹⁴) z_i
10	0 ⁸	0 ⁹	— ¹¹	0 ¹⁰

- 1) Design a comparator to check if two n -bit noz. are equal. Configure this using cascaded stages of 1-bit comparators.



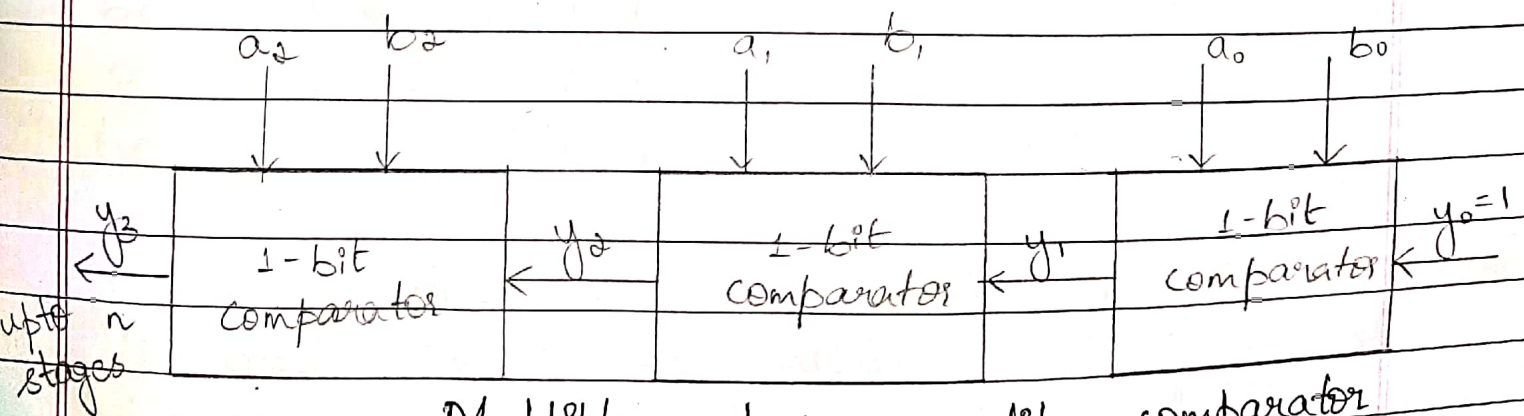
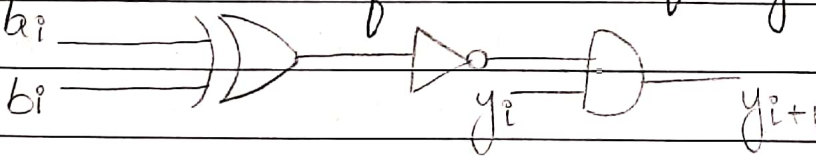
Block diagram

cell no.	a_i	b_i	y_i	y_{i+1}	$b_i y_i$
0	0	0	0	0	0 0 0 1 1 0
1	0	0	1	1	0 0 1 0 0 2
2	0	1	0	0	0 1 0 1 0 3
3	0	1	1	0	1 0 0 1 0 4
4	1	0	0	0	0 1 1 0 1 5
5	1	0	1	0	0 1 1 0 1 6
6	1	1	0	0	1 0 0 1 0 7
7	1	1	1	1	1 0 0 1 0 8

$$G = \bar{a}_i \bar{b}_i y_i + a_i b_i y_i$$

$$G = (a_i \oplus b_i) y_i$$

Implementation of 1-bit equality comparator



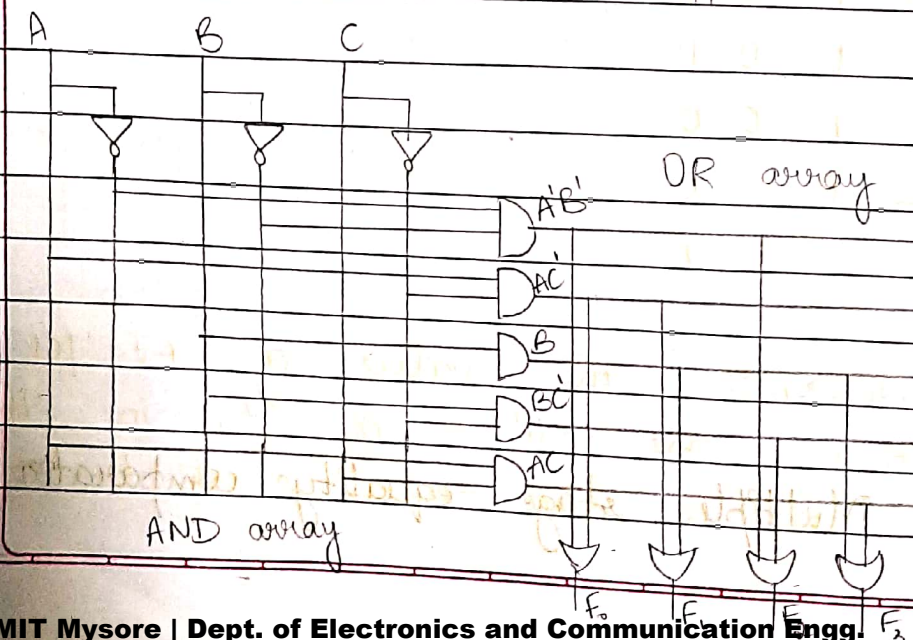
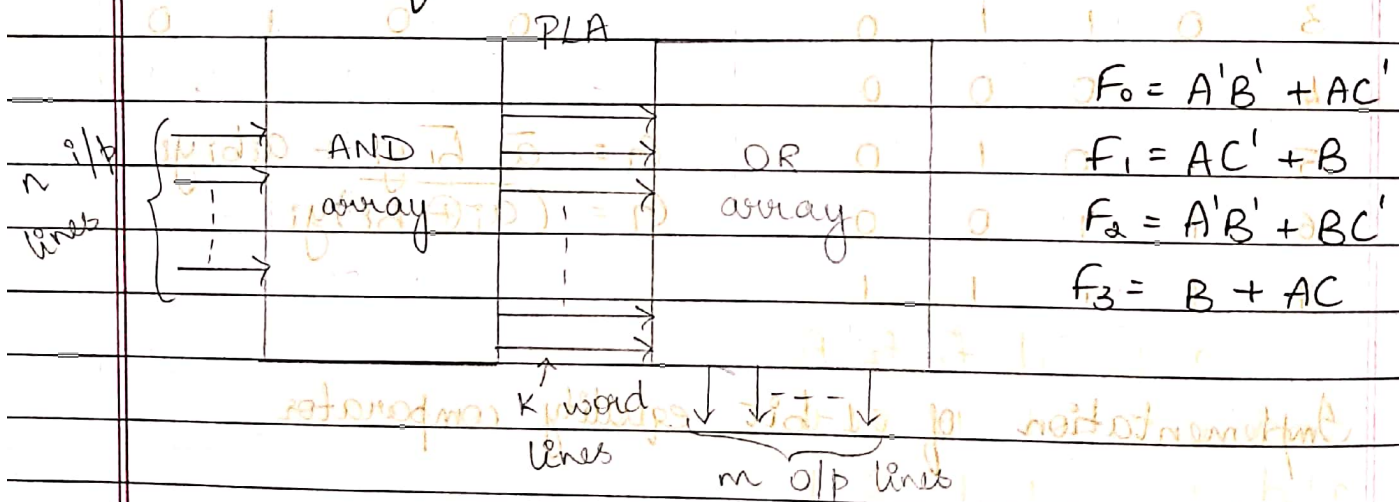
Multiple stage equality comparator

A PLD is a general name for a digital integrated circuits which is capable of being programmed to provide the variety of different logic functions.

PLA's (Programmable Logic Arrays):

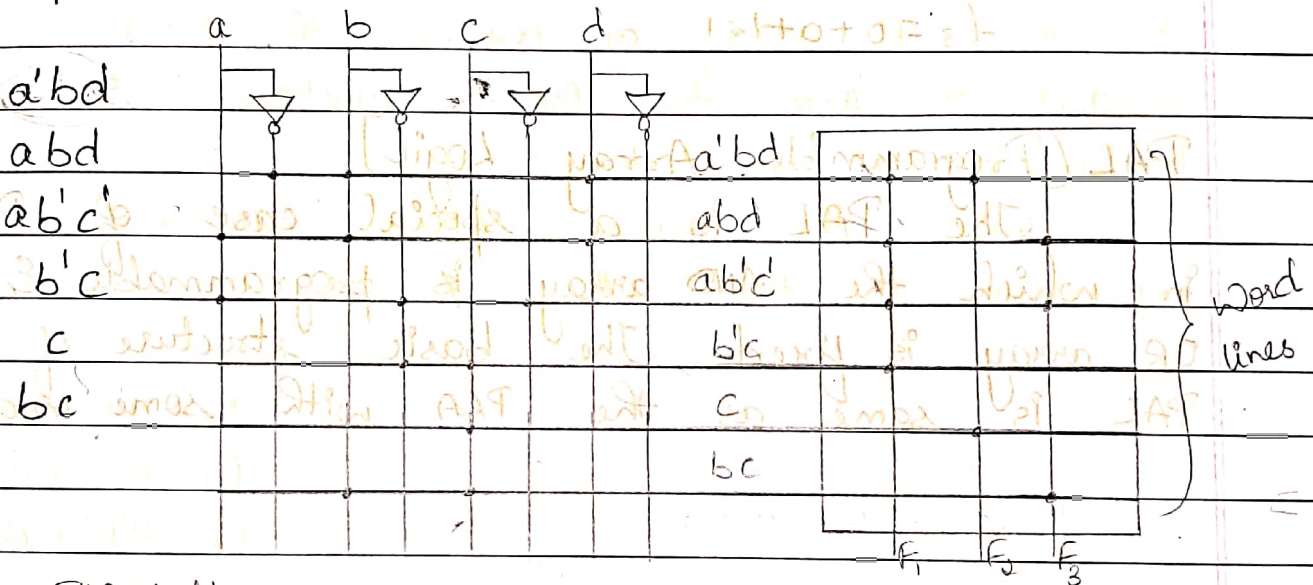
The PLA consist of AND array which realises selected product terms of the input variable E , the OR array or together the product terms needed to form the output functions. So a PLA implements a SOP terms.

Consider the general block diagram & an expression for the PLA.



Prod terms	A	B	C	F_0	F_1	F_2	F_3
$A'B'$	0	0	-	1	0	1	0
AC'	1	-	0	1	1	0	0
B	-	1	-	0	1	0	1
BC'	-	1	0	0	0	1	0
AC	1	-	1	0	0	0	1

Another case of PLA where AND & OR arrays are replaced by a dot symbol for the following expressions.



PLA table

	a	b	c	d	F_0	F_1	F_2	F_3
$a'bd$	0	1	-	1	1	0	1	0
abd	1	1	-	1	1	0	1	1
$ab'c'$	1	0	0	-	1	0	1	0
$b'c$	-	0	1	-	1	0	0	0
c	-	-	1	-	0	1	0	0
bc	-	1	1	-	0	0	1	1

To determine the value of F_i for given i/p combination, the values of F_i in the selected row of a PLA table must be OR together.

The following egs refers to the PLA table.

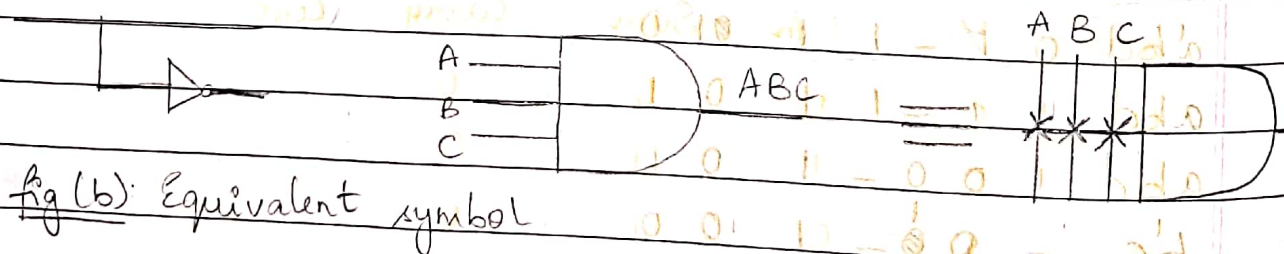
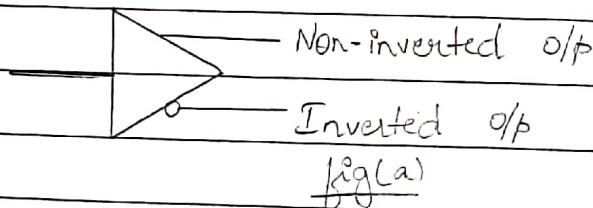
Case 1: If $abcd = 0001$, no rows are selected
 & all F 's are zero.

Case 2: If $abcd = 1001$, then 3rd row is selected
 & $f_1 = f_3 = 1$, $f_2 = 0$.

Case 3: If $abcd = 0111$, then 1st, 5th & 6th rows
 are selected & $f_1 = 1 + 0 + 0 = 1$, $f_2 = 1 + 1 + 0 = 1$
 $f_3 = 0 + 0 + 1 = 1$

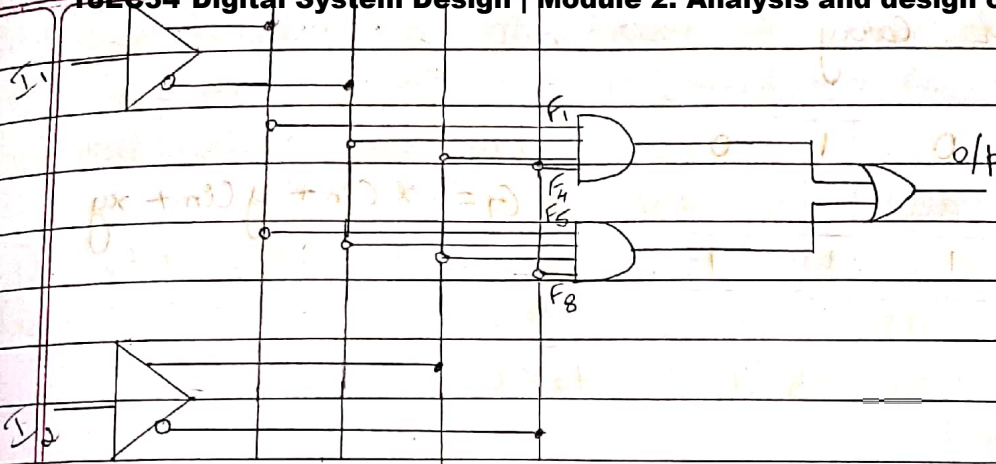
PAL (Programmable Array Logic).

The PAL is a special case of PLA in which the AND array is programmable & the OR array is fixed. The basic structure of the PAL is same as the PLA with some modification.

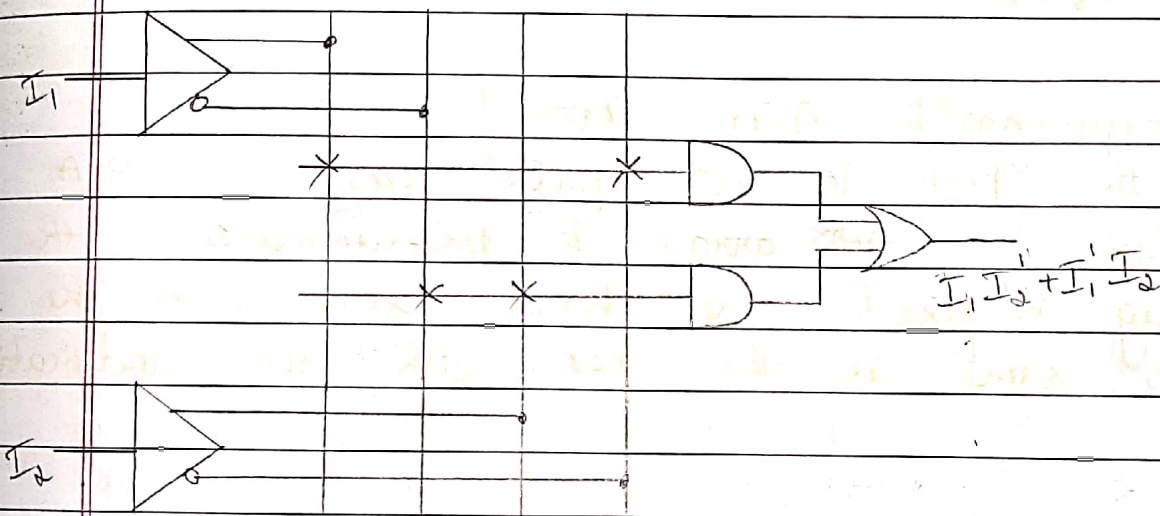


fig(b) Equivalent symbol

Eg: $f = I_1 \cdot I_2' + I_1' \cdot I_2$



(a) Unprogrammed



(b) Programmed

X	Y	Cin	Sum	Carry	Out
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	

K-map for Sum

00	01	11	10
0 ⁰	1 ¹	0 ³	1 ²
1 ⁴	0 ⁵	1 ⁷	0 ⁶

$$G = \bar{x}\bar{y}C_{in} + x\bar{y}\bar{C}_{in}$$

$$+ \bar{x}y\bar{C}_{in} + xyC_{in}$$

K-map for Carry

	00	01	11	10
0	0 ⁰	0 ¹	1 ³	0 ²
1	0 ⁴	1 ⁵	1 ⁷	1 ⁶

$$G = x C_{in} + y C_{in} + xy$$



CPLD's (Complex Programmable Logic Devices):

FPGA [Field Programmable gate array]:

An FPGA is an IC that contains an array of identical logic cells with programmable interconnections. The user can program the function realized by each logic cells & the connections b/w the cells with the help of LUT's.

DATE:

PAGE:

Consider a function $f(a, b, c, d) = a'f_0 + af_1$. [Based on Shannon's expansion theorem].

i) $f(a, b, c, d) = \Sigma(0, 1, 2, 3, 6, 7, 8, 12, 13, 15)$

		cd \ ab		00	01	11	10
ab \ cd	00	1	1	1	1	1	1
	01	0	0	1	1	1	1
	11	1	1	1	1	1	1
	10	1	0	0	0	0	0

$$G = \bar{c}\bar{d} + \bar{c}a + c\bar{a}\bar{b} + bcd$$

		cd \ ab		00	01	11	10
ab \ cd	00	1	1	1	1	1	1
	01	0	0	1	1	1	1
	11	1	1	1	1	1	1
	10	1	0	0	0	0	0

$$G = a'(c'd' + cd + b'c) + a(c' + bd)$$