Half Adder or Binary Adder:

↳ is a combinational circuit that performs the addition of 2 bits. → (2 inputs & 2 outputs)

```
  a
+ b
-----
Cout S
```

(The Block diagram of half Adder (HA)

* The Truth table:

a	b	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

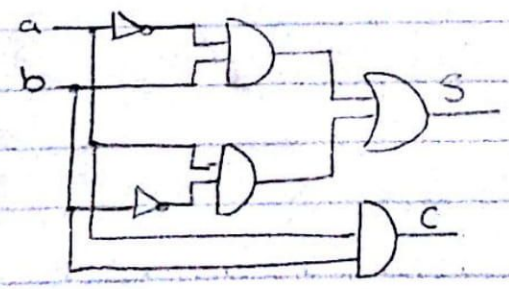
* The equations of S, Cout:

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

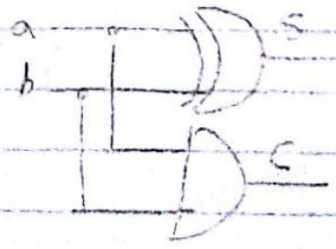
$= \boxed{a \oplus b}$

$\boxed{Cout = ab}$

* implementation by using AND, OR, NOT In SoP:



* Implementation by using XOR:



No
Date

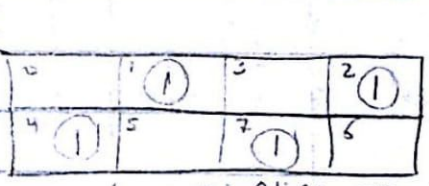
Full Adder :

(3 inputs & 2 outputs)

2 input + carry input — Sum — Carry output

Cin	a	b	C	S	Cout
a	0	0	0	0	0
+ b	0	0	1	1	0
	0	1	0	1	0
Cout S	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

$$S = \grave{a}\grave{b}c + \grave{a}b\grave{c} + a\grave{b}\grave{c} + abc$$



(No Simplification)

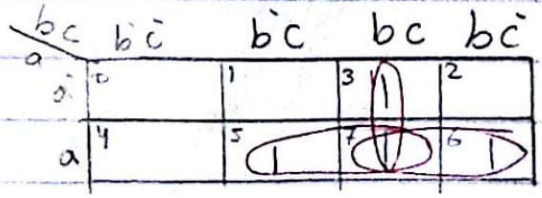
SOP $= \grave{a}(\grave{b}c + b\grave{c}) + a(\grave{b}\grave{c} + bc)$

$= \grave{a}(b \oplus c) + a(b \oplus c)'$

$= \boxed{a \oplus b \oplus c}$

by XOR

$$C_{out} = \grave{a}bc + a\grave{b}c + ab\grave{c} + abc$$



$$\boxed{C_{out} = bc + ac + ab}$$

SOP

$= ab(\grave{c} + c) + c(\grave{a}b + a\grave{b})$

by XOR $= \boxed{ab + c(a \oplus b)}$

* Implementation the Full Adder (FA) by using:
AND, OR, NOT:-

Where:-

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

$$C = bc + ac + ab$$

(From Kmap):

((You implement it))

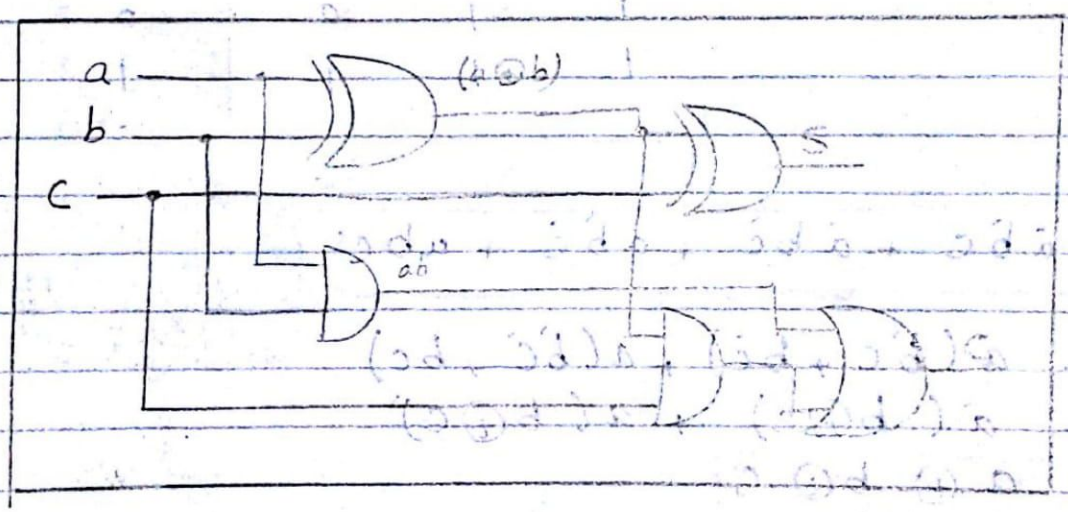
---

* How To implement A full Adder by using Two half Adder ?!

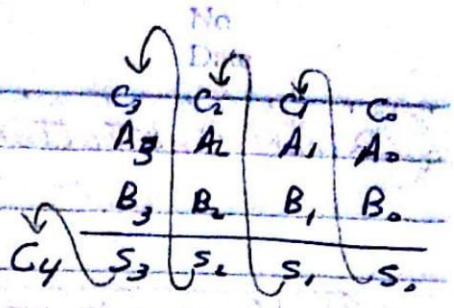
↳ From the equations of S, C :-

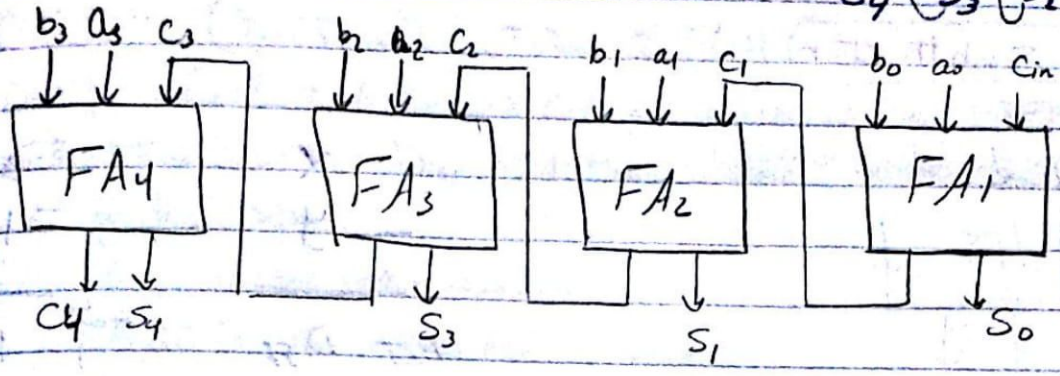
$$S = (a \oplus b) \oplus c$$

$$C = ab + c(a \oplus b)$$



* Cascade 4-bit Full Adder:-

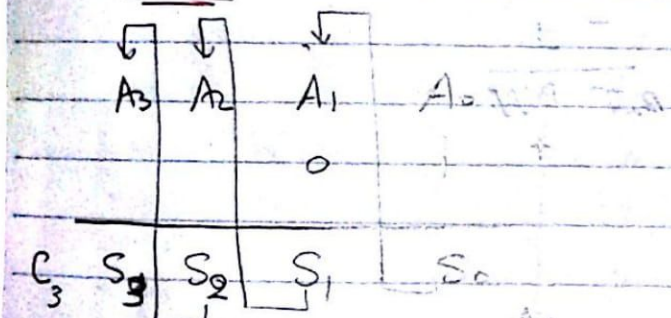




Q 4.11 (a)

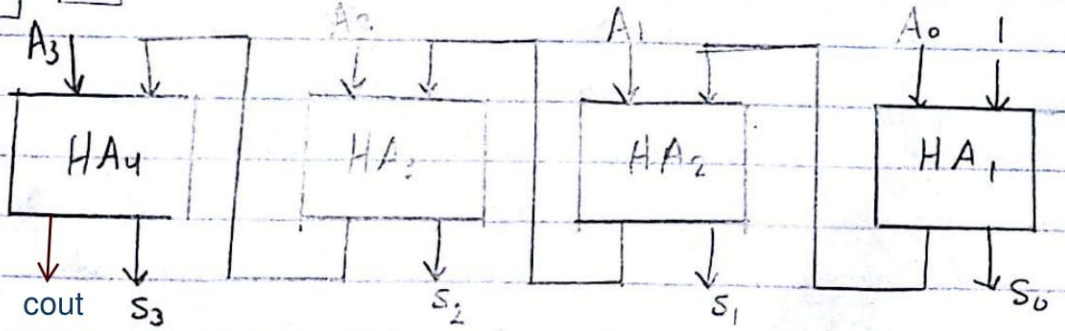
* Using Four half Adders, design a 4-bits Combinational Circuit incrementer (a circuit that adds 1 to a 4 bits binary number).

ANS.



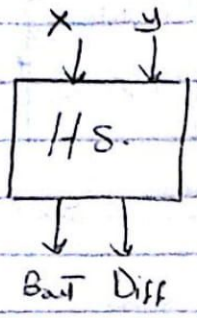
⇒ We will use (Half Adder) (HA)

Where → (2 inputs & 2 outputs)



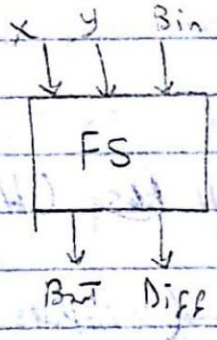
Binary Subtractor :-

* **Half Subtractor** (HS) (2 input, 2 output)



$$\begin{array}{r} x \\ -\ y \\ \hline \text{Bout}\ \ \text{Diff} \end{array} \qquad \begin{array}{r} \hookleftarrow \\ -1 \\ \hline \text{Bout} \\ 1 \quad\ 1 \end{array}$$

* **Full Subtractor** (FS)



$$\begin{array}{r} \text{Bin} \\ x \\ -\ y \\ \hline \text{Bout}\ \ \text{Diff} \end{array}$$

No
Date

* Binary subtractor:-

Q4.12 (b)

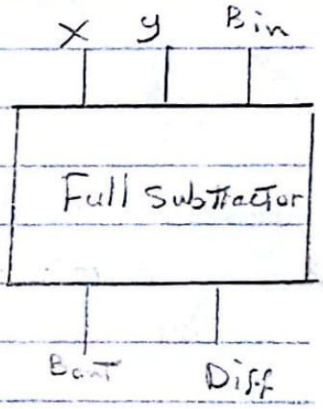
* Design a full subtractor circuit with three inputs, X, y, Bin and Two outputs Diff and Bout. The circuit subtracts X - y - Bin Where:

Bin is the input Borrow
Bout is the output Borrow and.
Diff is the difference result.

ANs:

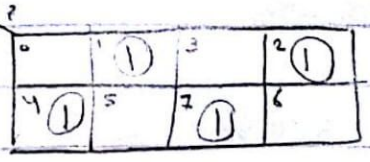
* The Truth table of the subtractor:-

X	y	Bin(Z)	Diff	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Let: Bin = Z

$Diff = \dot{x}\dot{y}z + \dot{x}y\dot{z} + x\dot{y}\dot{z} + xyz$



$= \dot{x}(\dot{y}z + y\dot{z}) + x(\dot{y}\dot{z} + yz)$

$= \dot{x}(y \oplus z) + x(y \oplus z)'$ No Simplification

$= x \oplus y \oplus z$

→

No
Date

$B_{out} = \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + xyz$

$= \boxed{\bar{x}y + \bar{x}z} + yz$

x \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	$y z$	$y\bar{z}$
$\bar{x}$	0	1 (1)	3 (1)	2 (1)
x	4	5	7 (1)	6

Adder / Subtractor

$A - B$

$A + (-B)$

↳ 2's Comp. = (1's Comp + 1)

$A + ((B \oplus 1) + 1)$

↳ 1's comp + 1 ⇒ 2's Comp. of B.

$[0 \oplus 1] \rightarrow 1$

$[1 \oplus 1] \rightarrow 0$ ⇒ XOR for 1's Comp.

⇒ **Mode input** M

* if $\boxed{M = 0}$

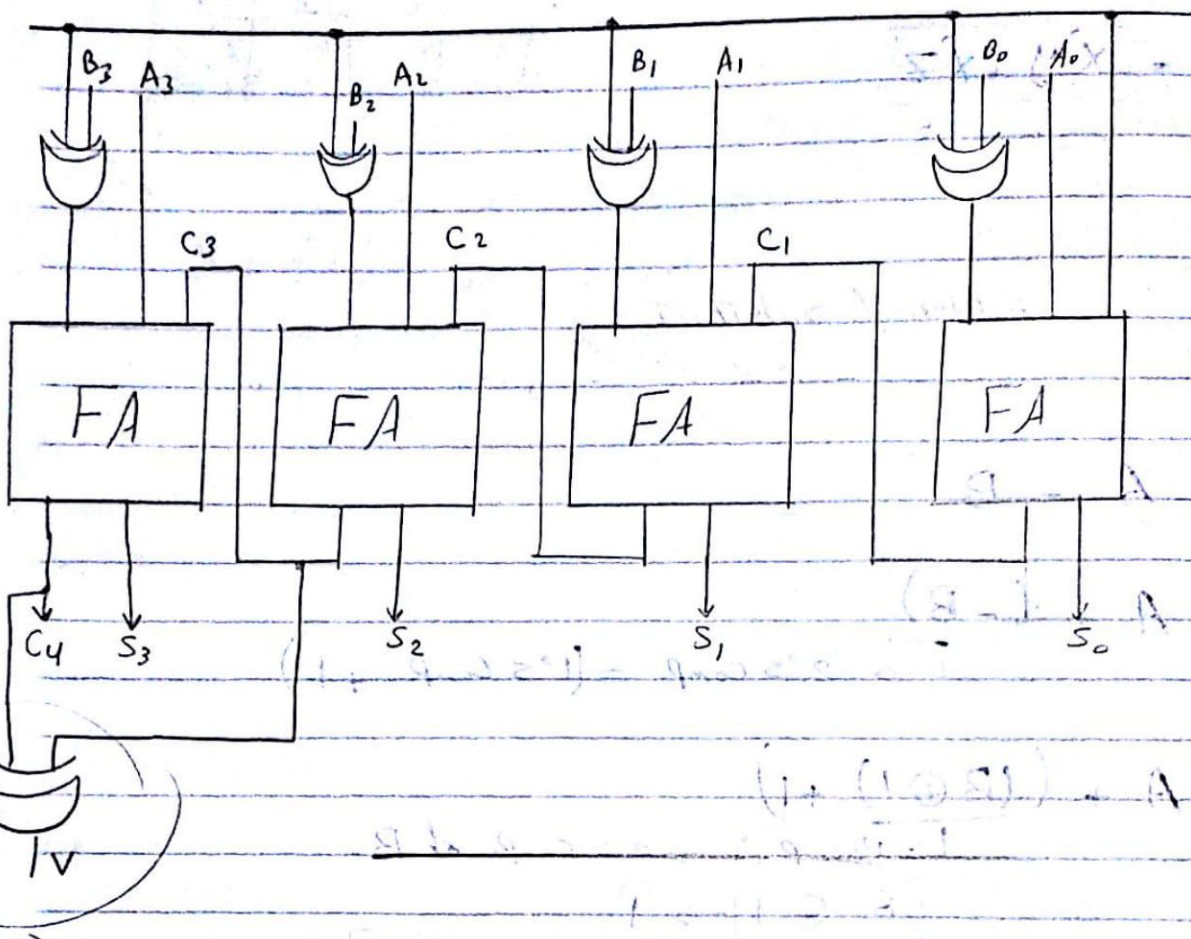
$A + [(B \oplus M) + M]$

$A + [(B \oplus 0) + 0]$ → for Adder when $M = 0$

* if $\boxed{M = 1}$

$A + [(B \oplus 1) + 1]$ → for subtractor when $M = 1$

→



To Detect if there is overflow or no: ($V = C_3 \oplus C_4$)

if $V = 1$ → Where (C_3 & C_4 are different)
Then There is an overflow

if $V = 0$ → Where (C_3 & C_4 are the same)
Then there isn't overflow.

No
Date

ex A B

if M = 1 , 1101 , 0101

```
  1 1
  1101
+ 1011   ← 2's comp.
-------
 (1)1000
```

→ -1000

Range of 4 bits = $[-2^{4-1}, +2^{4-1}-1]$ = [-8, +7]

$V = C_3 \oplus C_4 = 1 \oplus 1 = 0$ → No overflow.

No
Date

Q4.13

* The following Adder/Subtractor circuit has the following values for the mode input M and data inputs A & B.

The Block diagram of Adder/Subtractor

	M	A	B
a)	0	0111	0110
b)	0	1000	1001
c)	1	1100	1000
d)	1	0101	1010
e)	1	0000	0001

* In each of the previous cases, determine the values of the four Sum outputs, The Carry C, overflow V

ANS

a)

M = 0 → Adder

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1} \\ 0111 \\ +\ 0110 \\ \hline 1101 \end{array}$$

$C_4 = 0$ | Sum = 1101

$V = C_3 \oplus C_4$

$= 1 \oplus 0 = 1$

SUM	C	V
1101	0	1

→ G_H

No
Date

b)

$M = 0 \rightarrow$ Adder

C_{in} = 0

```
  1 0 0 0
  1 0 0 1
 ---------
1 0 0 0 1
```

C_4 | Sum

$\Rightarrow V = C_3 \oplus C_4$

$= 0 \oplus 1$

$= 1$

Sum	C	V
0001	1	1

c)

$M = 1 \rightarrow$ Subtractor

```
  1 1 0 0
  1 0 0 0   <- 2's Comp.
 ---------
1 0 1 0 0
```

C_4 | Sum

$V = C_3 \oplus C_4$

$= 0 \oplus 1 = 1$

Sum	C	V
0100	1	1

d)

$M = 1 \rightarrow$ Subtractor

```
 (Cin) 0 1 0 1
+      0 1 1 0   <- 2's Comp.
 ---------------
  0 1 0 1 1
```

C_4 | Sum

$\Rightarrow V = C_3 \oplus C_4$

$= 1 \oplus 0$

$= 1$

Sum	C	V
1011	0	1

No
Date

e)

M = 1 → SubTractor.

$C_{3=6}$

0 0 0 0

1 1 1 1 ← 2's Comp.

0 1 1 1 1

C_4 sum

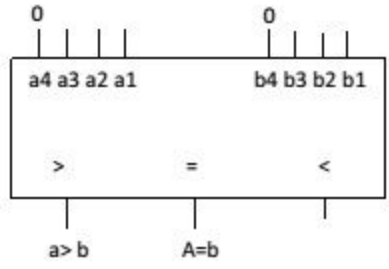
$V = C_3 \oplus C_4 =$

$= 0 \oplus 0 = 0$

Sum	C	V
1111	0	0

6. We need to determine whether a three-bit number, a_3, a_2, a_1, is equal to another number, b_3, b_2, b_1, or if it is greater than that number. (We do not need an output for less than.)

Implement this with AND and OR gates.



Here we will unwind the recursive definition of the A>B for n bits number stated as:

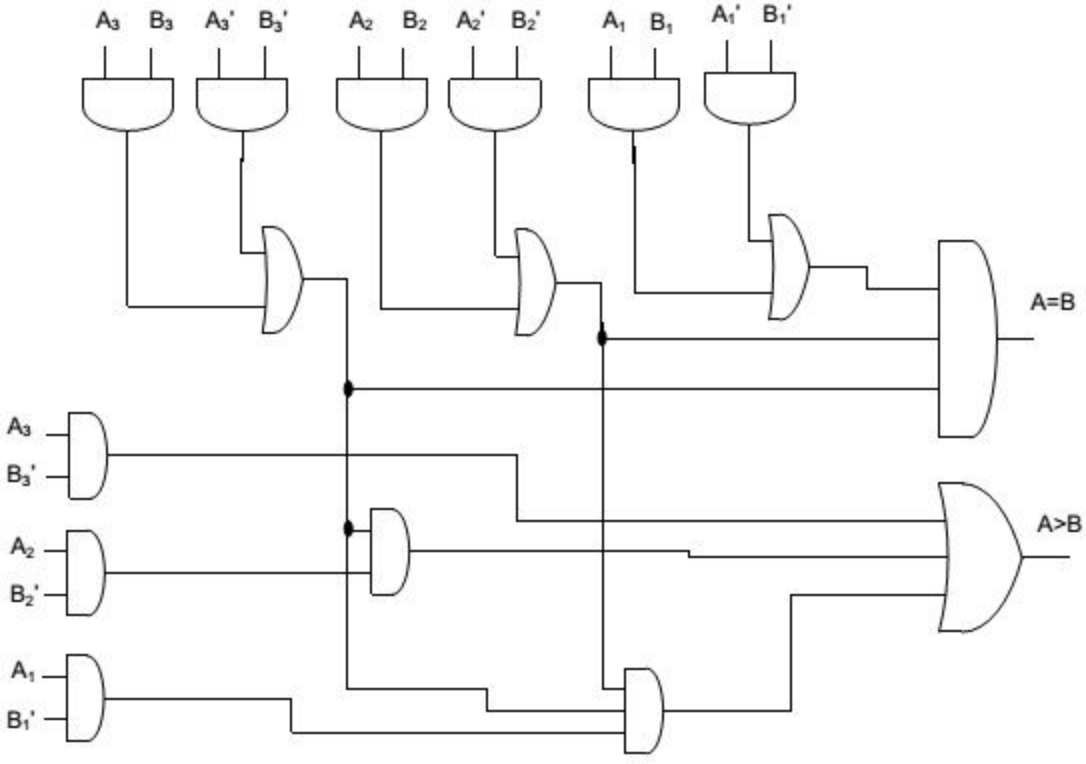
$A>B = (a_n > b_n)$ or $[(a_{n-1} > b_{n-1})$ and $(a_n > b_n)]$

For three bits number:

$A>B = (a_3 > b_3) + [(a_2 > b_2)$ and $(a_3 = b_3)] + [(a_1 > b_1)$ and$(a_2 = b_2)$ and $(a_3 = b_3)]$

F(A=B) =A xnor B = A B+A'B' = (A1 B1+ A1'B1')(A2 B2+A2' B2')(A3 B3+A3'B3')

Now build the circuit



4.11 Using four half-adders

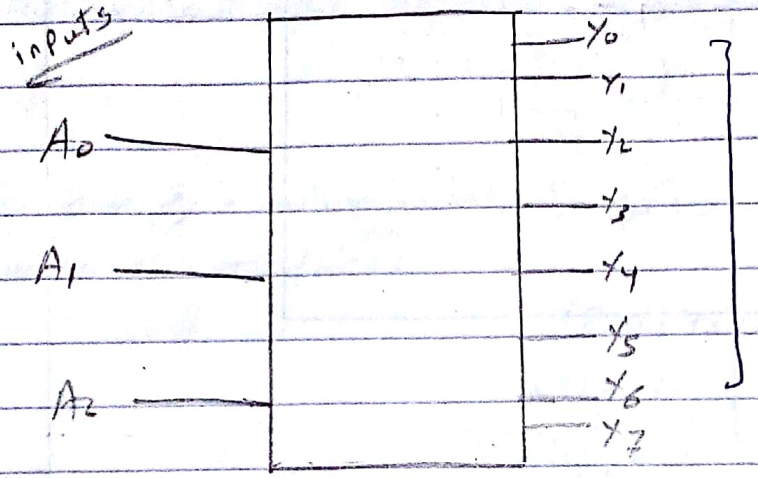
(a) Design a four-bit combinational circuit incrementer (a circuit that adds 1 to a four-bit binary number).

(b) Design a four-bit combinational circuit decrementer (a circuit that subtracts 1 from a four-bit binary number).

The solution >> in Assignment 5 file
on the e-learning site

Decoders :-

- Digital Circuits.
- Combinational gates.
- Converts input binary into single numeric output.



⇒ (n inputs) ⇒ (2^n outputs)

⇒ 3 inputs ⇒ $2^3 = 8$ outputs. from (0 → 7)

called

- 3 to 8 Decoder

or

- 1 of 8 ⇒ means that one of the outputs is Active.

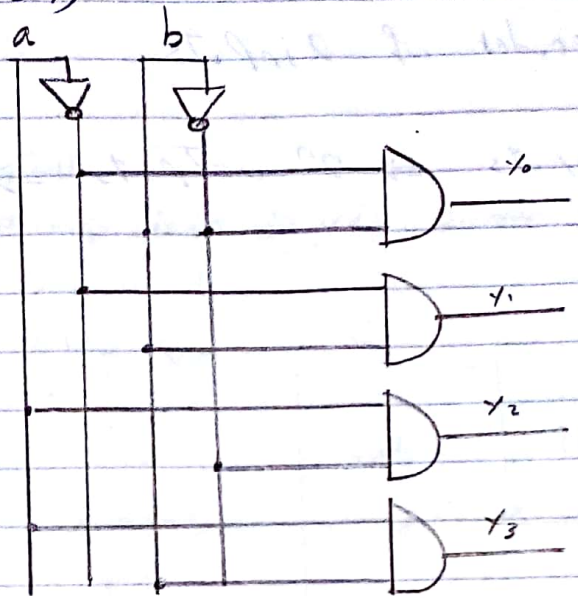
⇒ The Active Can be High or Low

High ↑ output [1]

Low ↑ outputs [0]

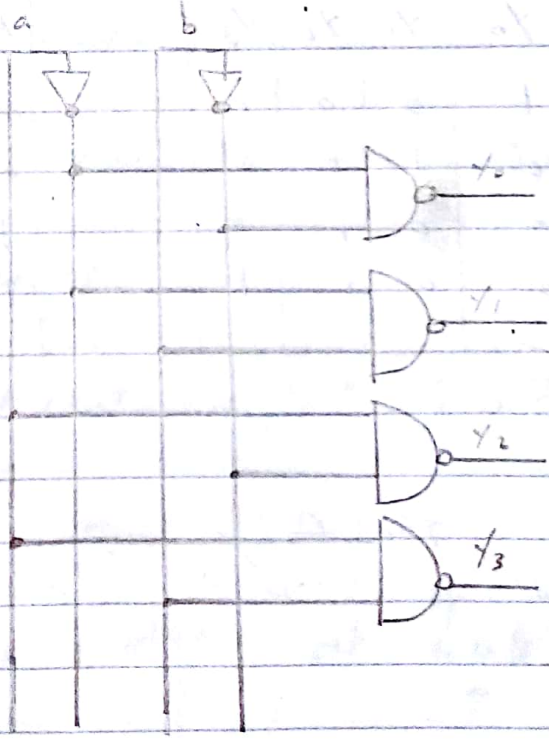
Active high Decoder (2-4)

$Y_0 = a'b'$
$Y_1 = a'b$
$Y_2 = ab'$
$Y_3 = ab$



Active low Decoder (2-4):

$Y_0 = (a'b')' = (a+b)$
$Y_1 = (a'b)' = (a+b')$
$Y_2 = (ab')' = (a'+b)$
$Y_3 = (ab)' = (a'+b')$



Decoder & Enable.

⇐ ال (enable) زي زرار للتحكم متى ال Decoder يكون Active ومتى يكون inactive

⇐ عندما ال (enable) يكون (Active) ال decoder هيكون Active (يشتغل)

⇐ وعندما ال (enable) يكون (inActive) كل ال outputs هتكون Inactive وبالتالي ال (decoder) هيكون inactive

⇐ ال (Enable) ممكن يكون Active high

↳ يعني أنه عندما يكون ① يكون Active
↳ يرمزله بـ (EN)

⇐ أو ممكن يكون Active low

↳ يعني أنه عندما يكون ⓪ يكون Active
↳ يرمز له بـ (EN`)

⇐ عادة عندما يحتوي ال (decoder) على (enable) واحد يكون هذا ال enable Active low
(EN`)

⇒

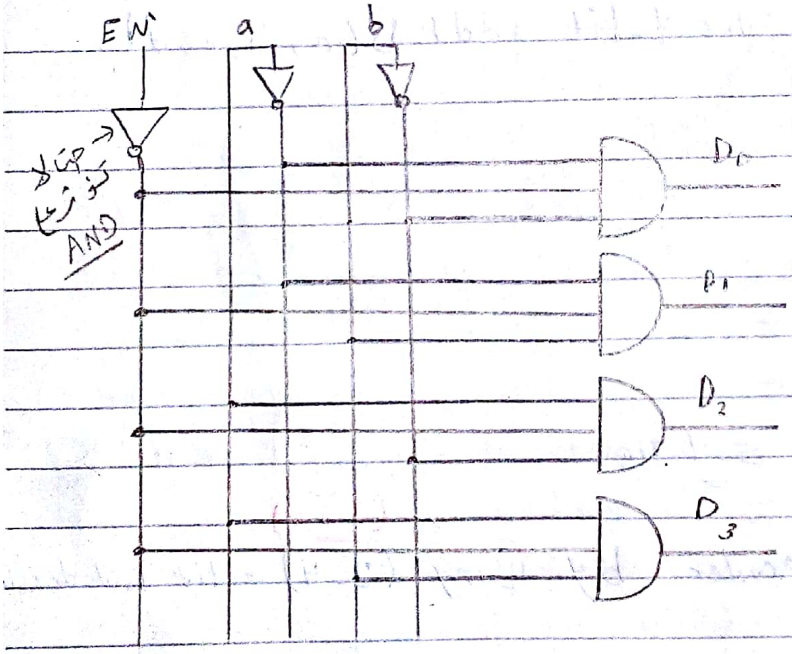
Truth Table

(Enable active low)

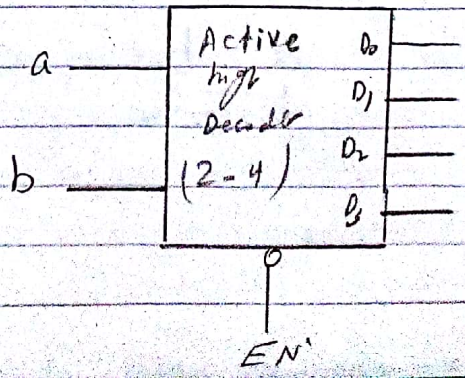
2 inputs — 4 outputs

EN'	a	b	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

$D_0 = a'b'$, $D_1 = a'b$, $D_2 = ab'$, $D_3 = ab$



Active high decoder with Active low enable.



Applications of Decoder

تطبيقات الـ (Decoder)

1) Implement logic functions:

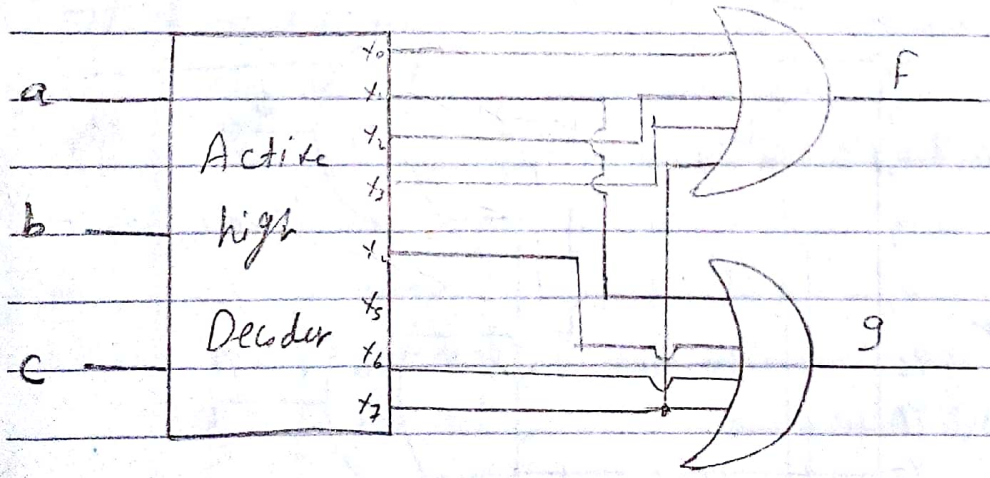
EX:

$$F(a,b,c) = \Sigma m(0,2,3,7)$$
$$g(a,b,c) = \Sigma m(1,4,6,7)$$

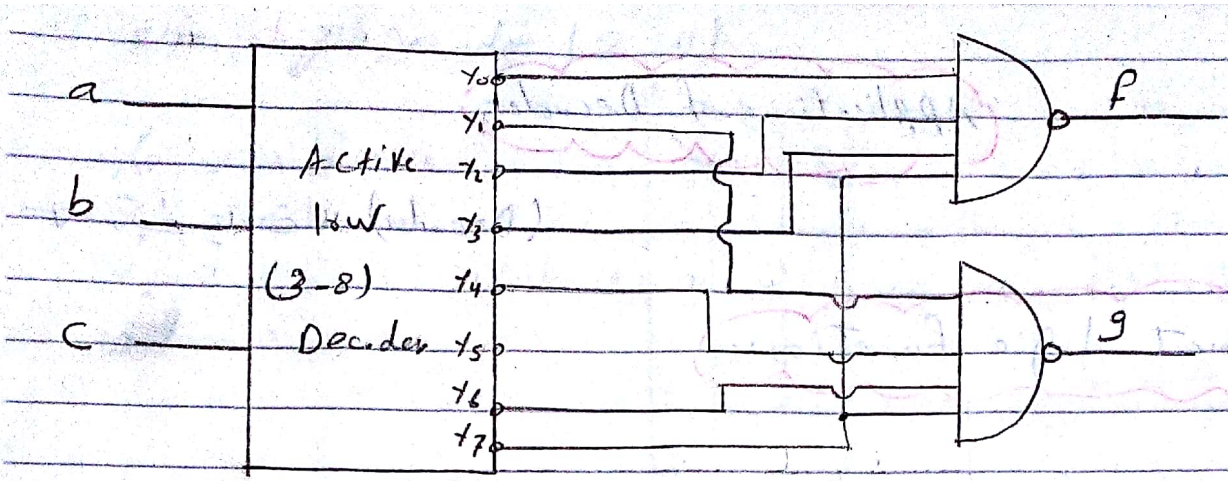
implement these functions by Decoder (Active high)

The Sol.

a, b, c ⇒ 3 inputs ⇒ 2^3 ⇒ 8 outputs, We will use (3 - 8) Decoder for each func



by Active low Decoder:



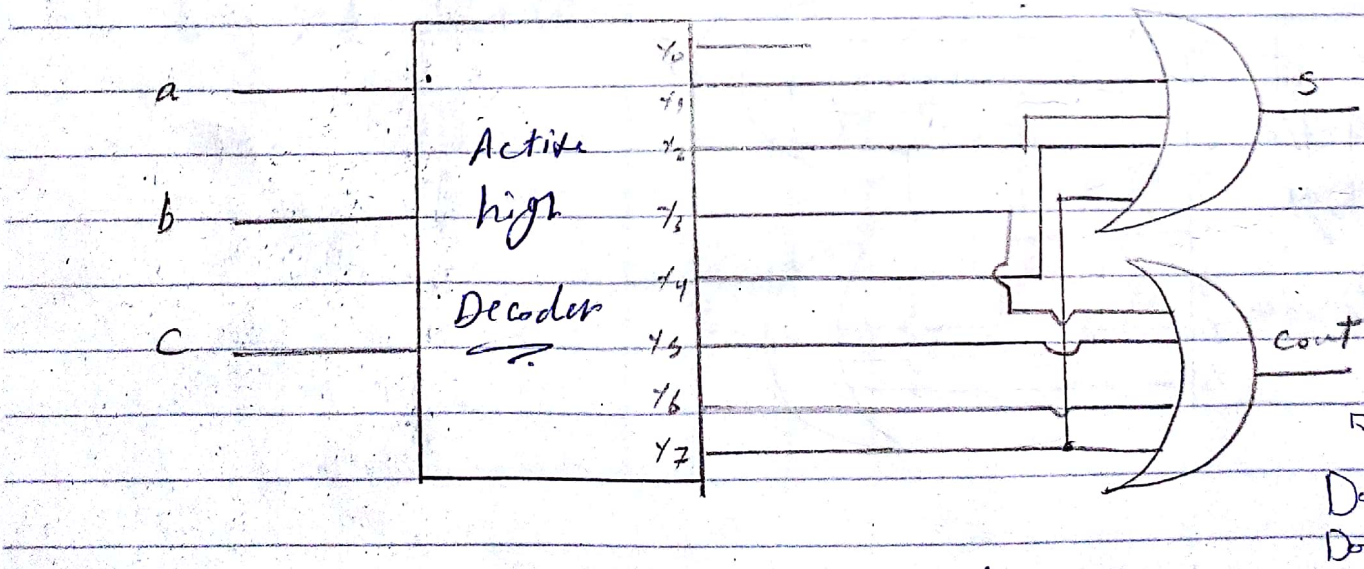
⇒ the Decoder can be used to implement any Combinational function circuits.

⇒ From the Truth table for full adder:

$S(a,b,c) = \Sigma m(1,2,4,7)$

$Cout(a,b,c) = \Sigma m(3,5,6,7)$

⇒ The implenatation by using Decoders is:

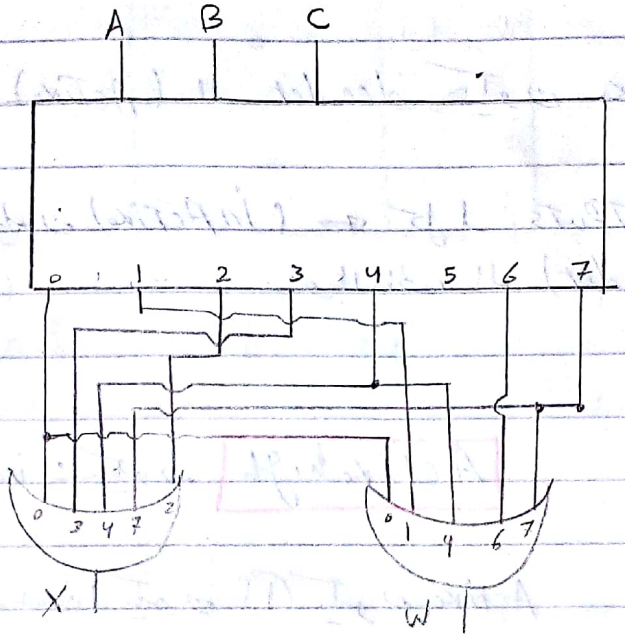


⇒ if we will use Active low Decoders, we use NAND

Ex:

(in quiz 2of1)

⇒ Consider the following circuit with an active high output decoder Draw a Truth table for X and W, then extract the switching formula for the design in terms of A, B, C.



The Solution

Truth table:

A	B	C		X	W
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	2	1	0
0	1	1	3	1	0
1	0	0	4	1	1
1	0	1	5	0	0
1	1	0	6	0	1
1	1	1	7	1	1

Switching formula ⟺ Algebraic expression

$X(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC$

$W(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC.$

⇒ Question #5 (Final 2009)

A 1-bit full subtractor that has three inputs, a borrow "Bin", , X and y, and produces two outputs (The difference D where D = "X-y-Bin" and the new borrow Bout).

a) Show the truth table for the full subtractor.
b) use the truth table to find the expressions for D and Bout
d) Minimize and implement the subtractor using 3-8 active Low decoder with one active Low enable, by using OR gates

The Solution:

a) The truth table for the full subtractor:

$2^3 = 8$

X	y	Bin	Bout	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

4

b)

$B_{out} = \bar{x}\bar{y}B_{in} + \bar{x}yB_{in}' + x\bar{y}B_{in}' + xyB_{in}$

$D = \bar{x}\bar{y}B_{in} + \bar{x}yB_{in}' + x\bar{y}B_{in}' + xyB_{in}$

c) We will minimize the functions by k map.

B_{out}

B_{in} \ xy	$\bar{x}\bar{y}$	$\bar{x}y$	xy	$x\bar{y}$
B_{in}'	0	2 (1)	6	4
B_{in}	1 (1)	3 (1)	7 (1)	5

$B_{out} = \bar{x}y + \bar{x}B_{in} + yB_{in}$

$B_{out} = \Sigma m(1,2,3,7)$

D

B_{in} \ xy	$\bar{x}\bar{y}$	$\bar{x}y$	xy	$x\bar{y}$
B_{in}'	0	2 (1)	6	4 (1)
B_{in}	1 (1)	3	7 (1)	5

$D = \bar{x}\bar{y}B_{in} + \bar{x}yB_{in}' + x\bar{y}B_{in}' + xyB_{in}$

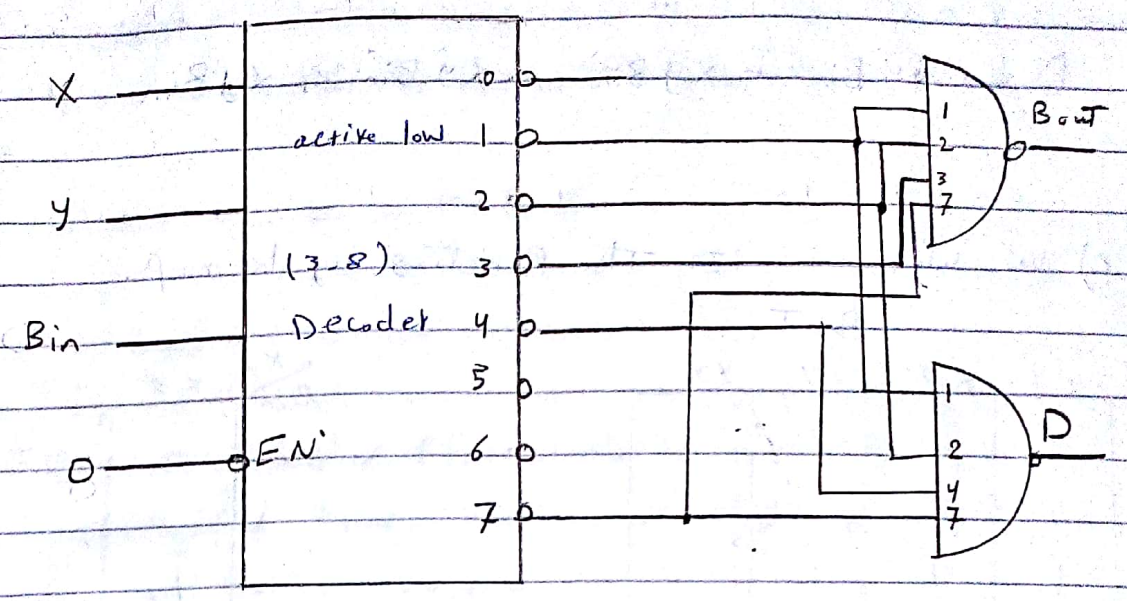
$= \Sigma m(1,2,4,7)$

⇒ We need to implemant these function by using (3-8) Active Low decoder with one active Low enable

& By using OR gate

⇒

↳ Active Low Decoder Use NAND gates

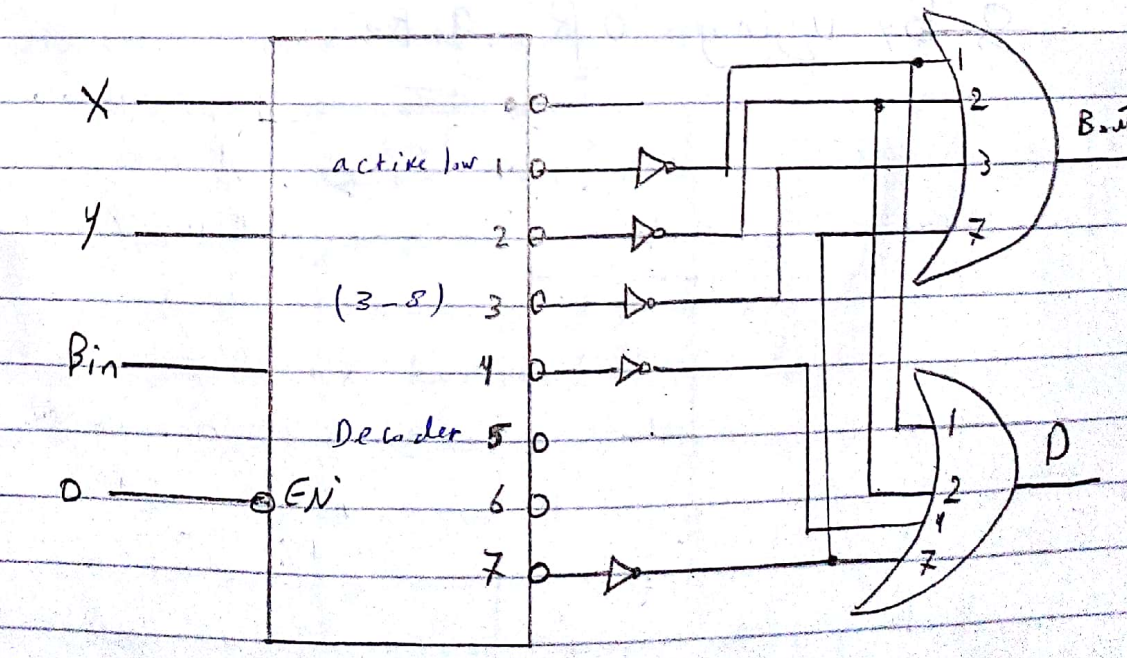


$B_{out} = \Sigma m(1,2,3,7)$ / $D = \Sigma m(1,2,4,7)$

⇐ لكنه يريد ان نستخدم (OR) وليس (NAND) وذلك لنفس الـ decoder

لذلك نستخدم (NOT) قبل (OR)

لأن الـ (decoder) Active low ⇐



Applications of Decoders

Application [2]:

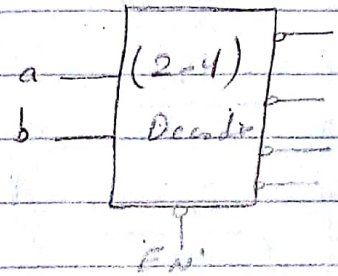
⇒ The Decoder helps to select one of many devices each of which has a unique address.

⇒ The address is the input to the decoder.

⇒ One output is active, to select one device that has this address.

Ex:

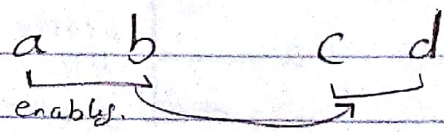
⇒ Use (2-4) active low decoders to select one of 16 devices, each has a unique 4-bit address (a, b, c, d)

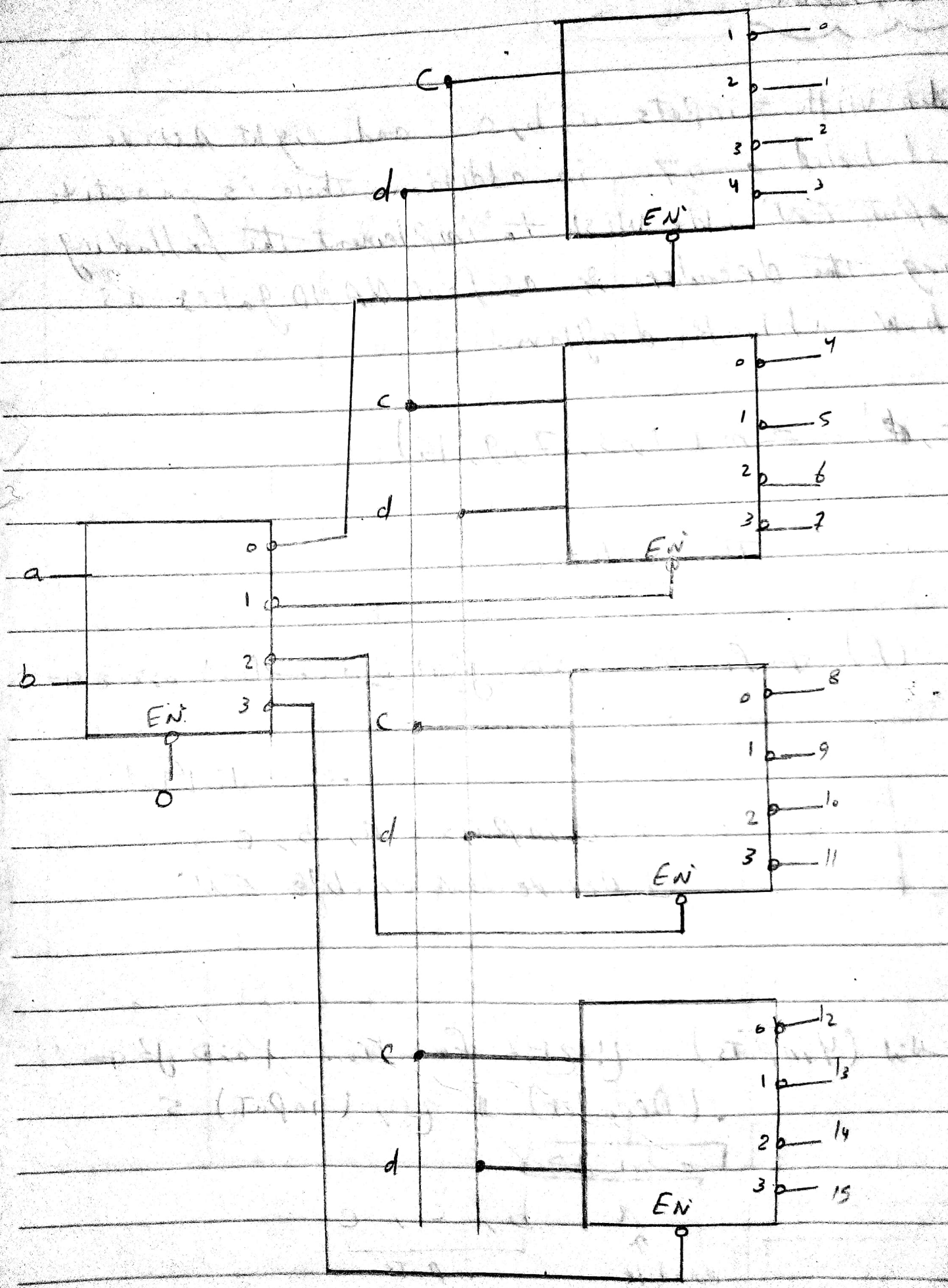


The Solution

⇒ to represent (4 to 16) decoder by using (3-8) (2-4) active low decoder

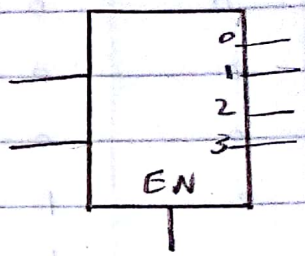
⇒ We need 16/4 ⇒ 4 decoder (2-4) active low Decoder.
and one decoder to choose among other 4 decoders.
based on 2 of the inputs.





⇒ We wish to build a 32-Way active high decoder using only four Way decoders shown below.

EN	a	b	0	1	2	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

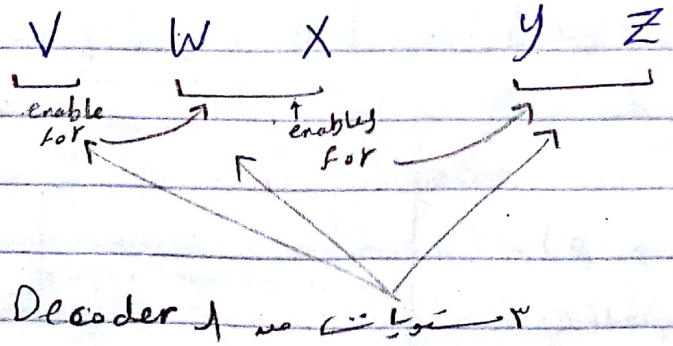


The solution:

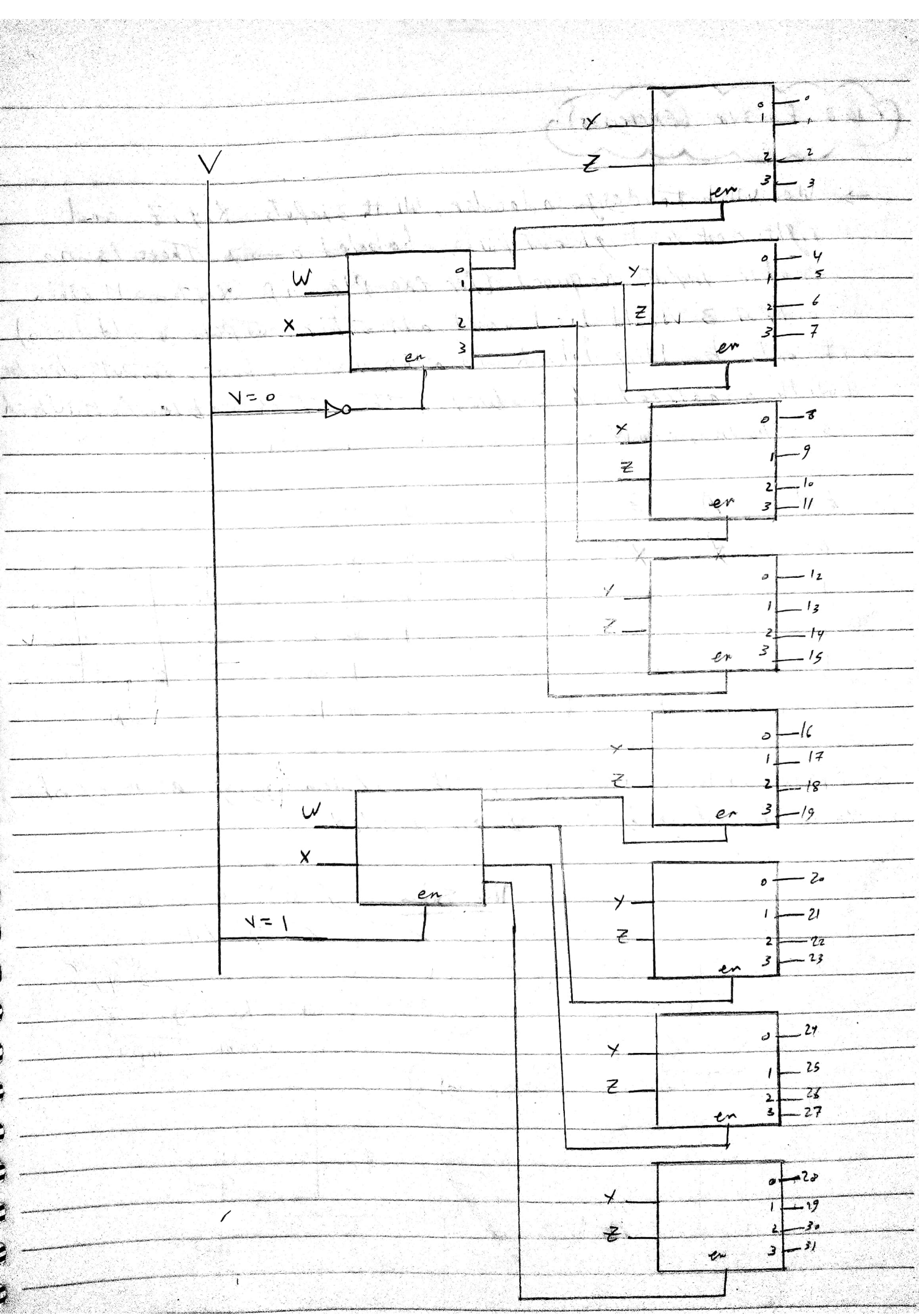
⇐ زي نقسم (Decoder) اللي عندنا (32 outputs) & (5 inputs) لـ (Decoder) اصغر بالشكل الذي أخذ (2input) فقط وخرج (4output) ولـ (1enable).

(32/4) Decoder output (5-32) Decoder output (2-4)

= 8 Decoder (2-4)



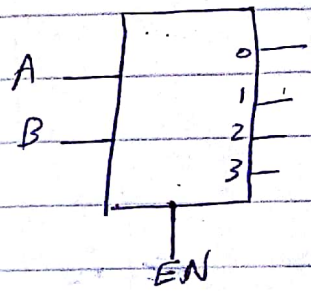
٣ متوباتس من



⇒ We wish to design a decoder, with 3 inputs X, Y, Z and eight active high outputs, labeled 0 → 7. There is no enable input required (for example if XYZ = 011, then output 3 would be 1 and all other outputs would be 0)

⇒ the only building block is a two-inputs, four-output decoder (with an active high enable), the Truth table for which is shown below.

EN	A	B	0	1	2	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



⇒ Draw a block diagram of the system using as many of these building blocks as are needed.

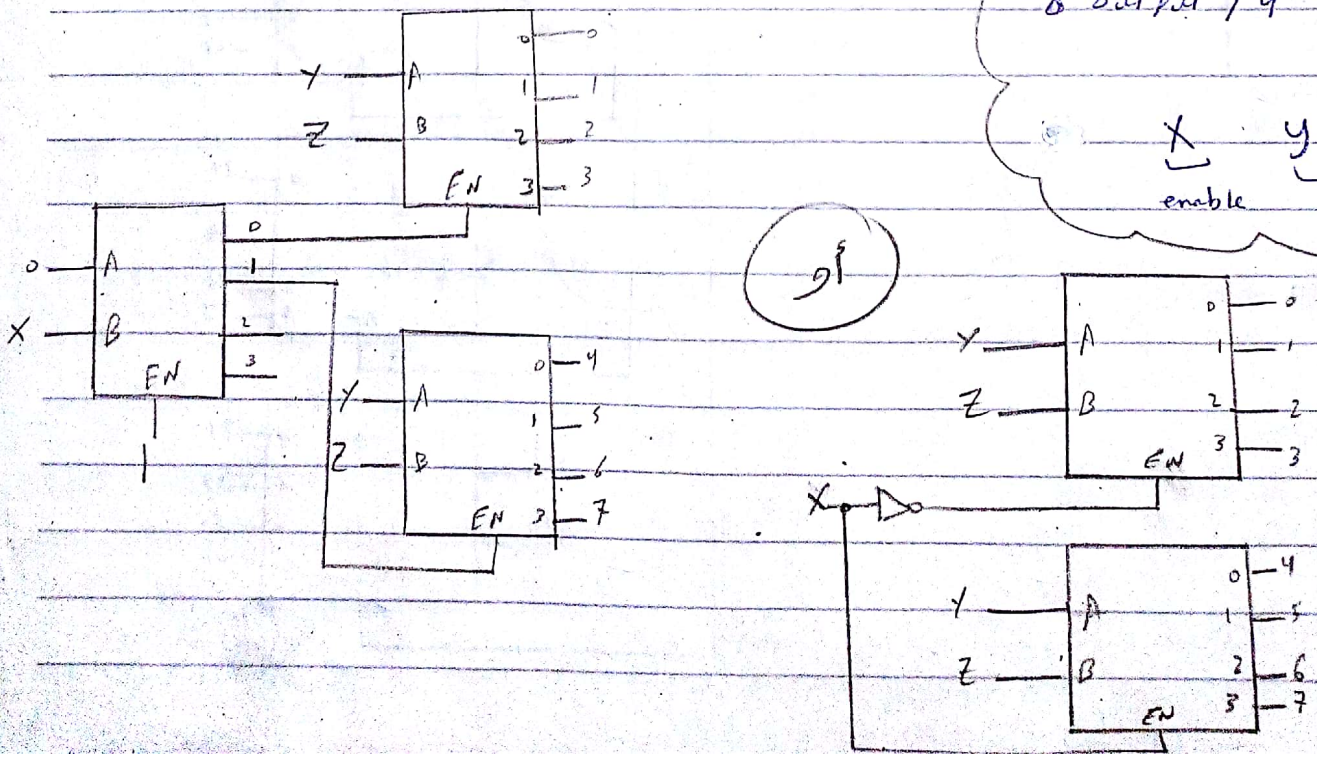
The Sol:

(4-8) Decoder (2-4) decoder

8 output / 4 = 2 decoder (2-4)

X Y Z

enable inputs.



EX:

⇒ Implement by using 3 inputs, 8 outputs decoders with 2 enables EN_1, EN_2':

$F = \sum m(0, 4, 5, 6, 7, 12, 15)$

$G = \sum m(1, 3, 12, 13, 14, 15)$

The solution.

W , x , y , z

W → enable ; x, y, z → 3 inputs

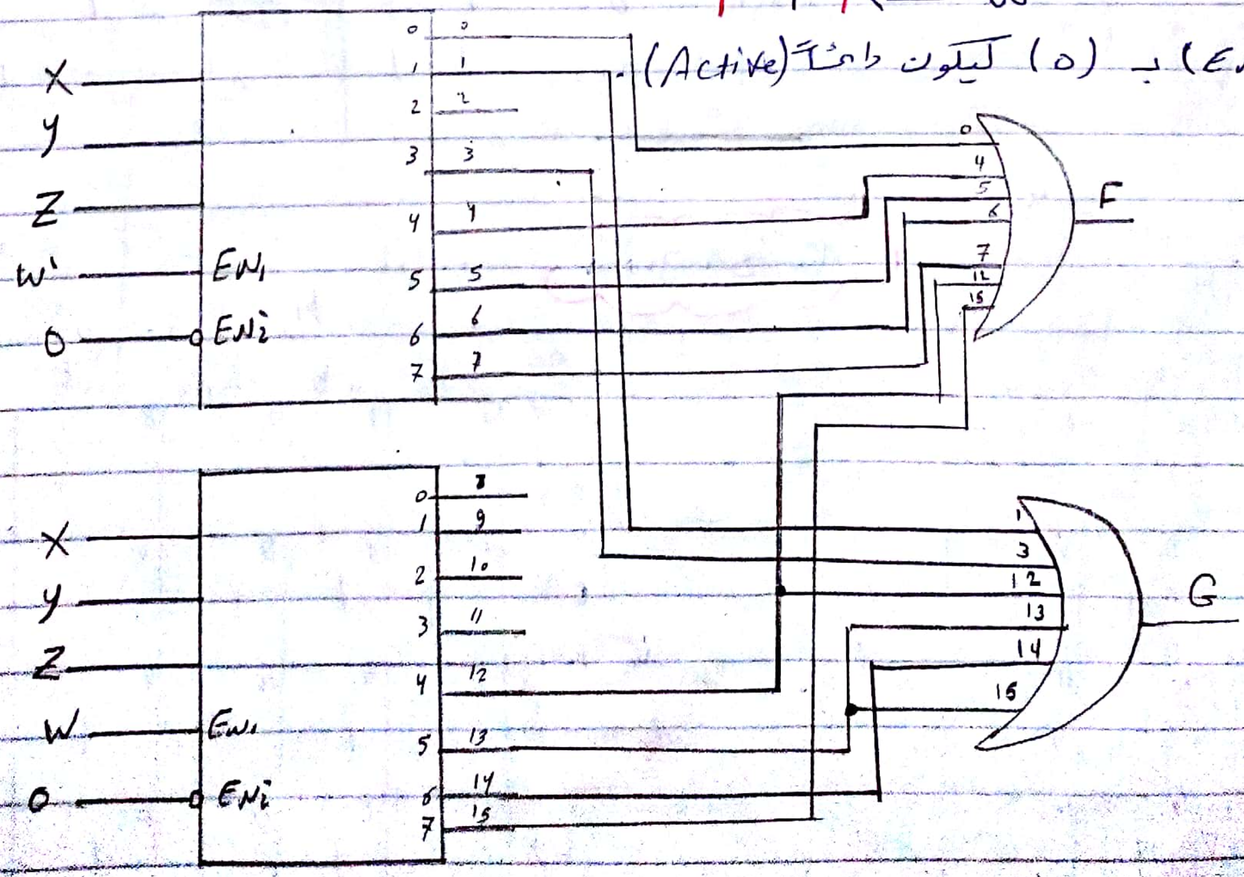
سنجعل ال (3 inputs) ⇐ x, y, z

وال (W) ⇐ هو (enable) سنوصله مع ال 2 enables

مثلاً (EN_1) ⇐ Active high

W	
0	← W'
1	← W

ونثبت (EN_2') بـ (0) ليكون دائماً (Active).

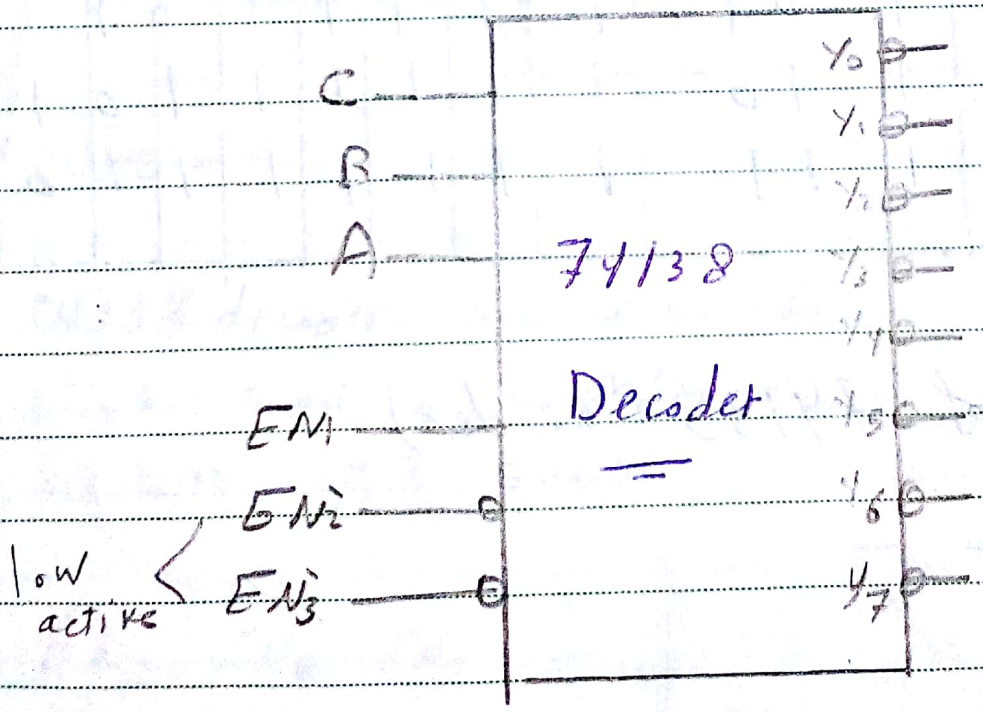


74138 Decoder

- is Active low outputs.
- has 3 Enable
  - 1 Active high: EN_1
  - 2 enable (Active low): $\overline{EN_2}$, $\overline{EN_3}$

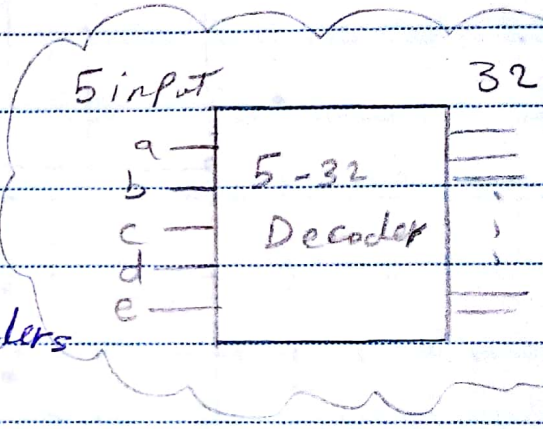
- (3 - 8) Decoder.

- A, B, C are inputs where C has "high order bit"



Truth table ⇒ P. 273.

Ex1: By using 74138 decoders, select one of 32 devices (each has a 5-bits address abcde)

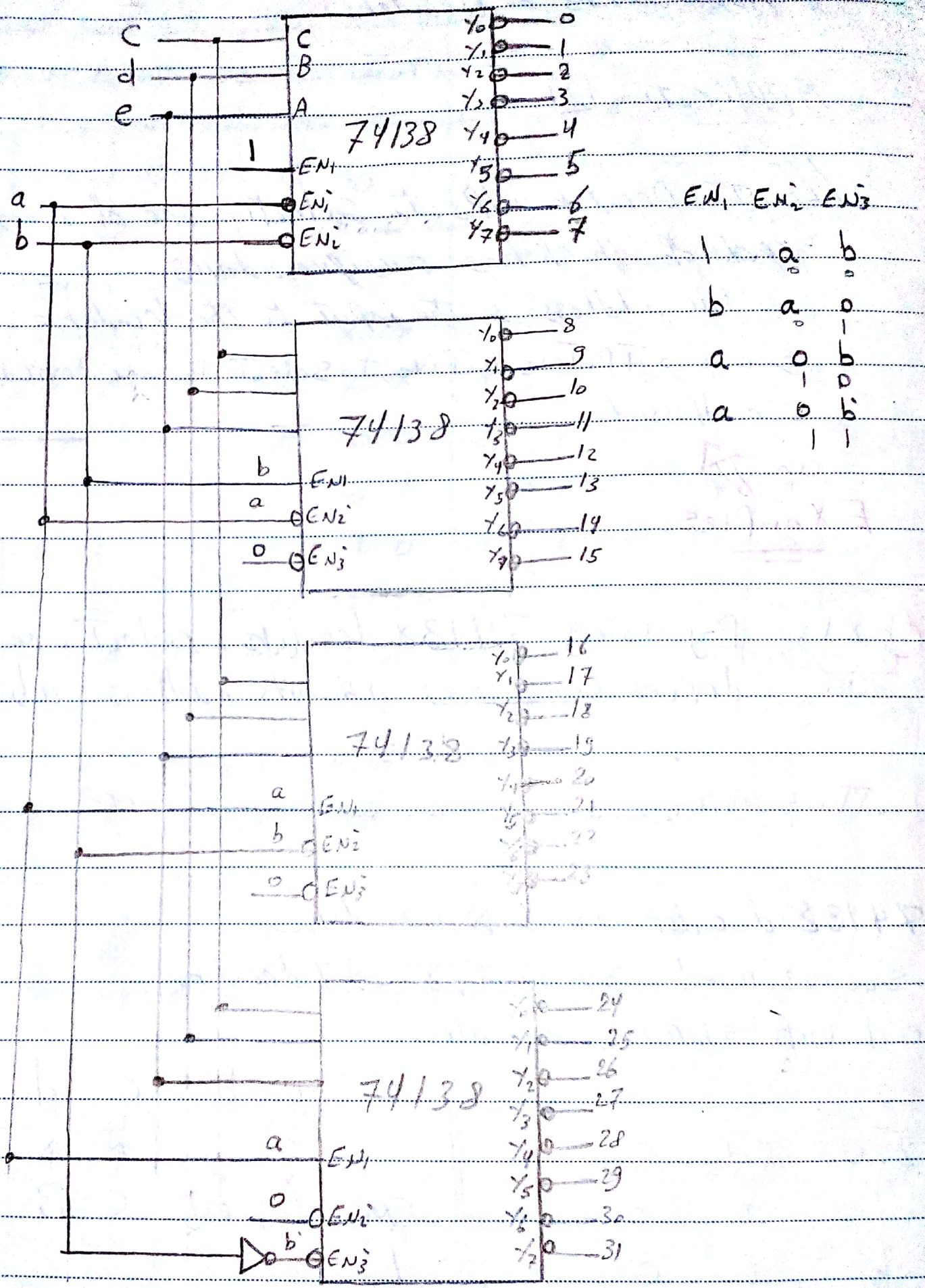


The solution

- 74138 decoder is 3-8 Decoder
- So, we need 32 output / 8 ⇒ 4 decoders
- each, with 3 inputs, 3 enables.

EN_1	$\overline{EN_2}$	$\overline{EN_3}$
1	a	b
b	a	0
a	0	b
a	b	0
OR b	0	a

a b	c d e	
Enables		
EN_1	$\overline{EN_2}$ (a)	$\overline{EN_3}$ (b)	C	B	A
1	0	0			
1	0	1			
1	1	0			
1	1	1			



* How to implement a Decoder to Convert Binary to octal ?!:

Ans :-

By using (3 x 8) Decoder:-

A	B	C	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

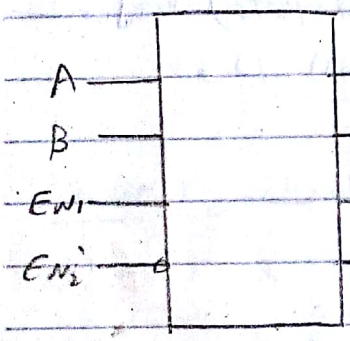
EX (midterm 2011) :-

⇒ Consider The following 2 functions F & H of the four variables a, b, c, and d, Whose minimum solutions are listed below :-

$$F(a,b,c,d) = b'cd + bc + abd$$

$$H(a,b,c,d) = ac + cd' + a'c'd + bd$$

⇒ Implement them using only decoders of the type shown (as many as needed) and two OR gates



EN_1	EN_2'	A	B	0	1	2	3
0	X	X	X	0	0	0	0
1	1	X	X	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	0
1	0	1	1	0	0	0	1

The solution:

F

cd \ ab	a'b'	a'b	ab	ab'
c'd'	0	4	12	8
c'd	1	5	13	9
cd	3	7	15	11
cd'	2	6	14	10

H

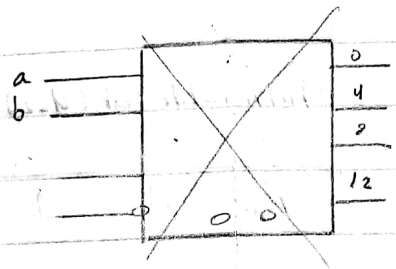
cd \ ab	a'b'	a'b	ab	ab'
c'd'	0	4	12	8
c'd	1	5	13	9
cd	3	7	15	11
cd'	2	6	14	10

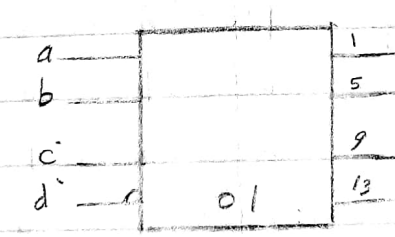
From K-map:

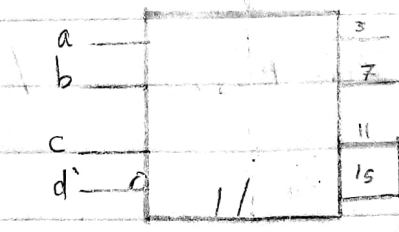
$F = \Sigma m(1, 6, 7, 9, 13, 14, 15)$

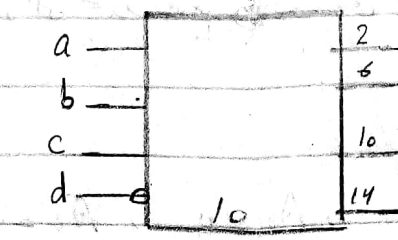
$H = \Sigma m(1, 2, 5, 6, 7, 10, 11, 13, 14, 15)$

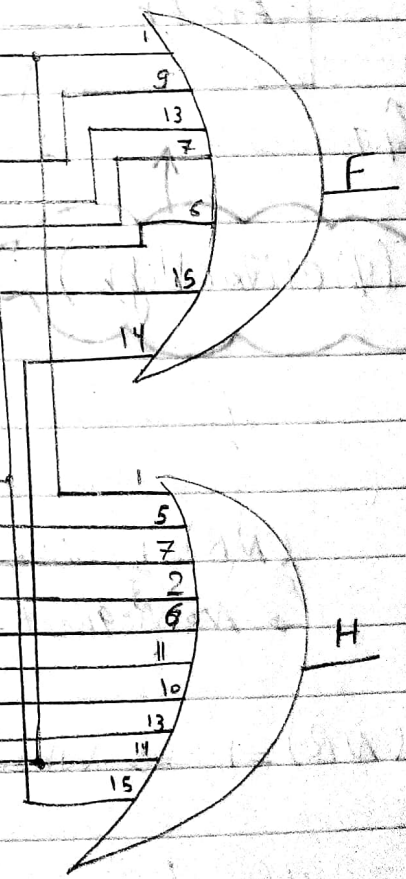
ملاحظة: من الـ (K-map) أن أول Row ← (00) ← $c'd'$ غير مقدم.
فممكننا جعل (c, d) (enables) وتلغا الـ (decoder) فعندما يكون (00) صار الـ inputs (A, b).











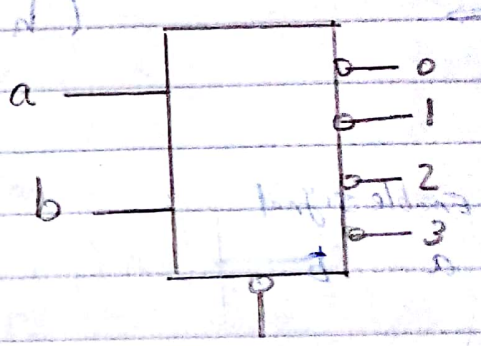
Active 1 ↓ Active 0 ↓

EN_1	EN_2
c	d
0	0 X
c` → 0	1 ← d`
c → 1	0 ← d
c → 1	1 ← d`

Implementing functions by using Decoders

Q: We have available as components three NAND gates (as many inputs as you need) and some active low input, active low enable decoders, as shown below (as many as you need).

EN	a	b	0	1	2	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



Show a diagram of a circuit to implement these functions: (for each group of functions)

⟹

a)

cd \ ab	a'b'	a'b	ab	ab'
c'd'	0	4	12	8
c'd	1	5	13	9
cd	3	7	15	11
cd'	2	6	14	10

cd \ ab	a'b'	a'b	ab	ab'
c'd'	1		1	
c'd			1	1
cd	1			1
cd'	1		1	

g

cd \ ab	a'b'	a'b	ab	ab'
c'd'				1
c'd	1		1	1
cd	1			
cd'	1		1	

h

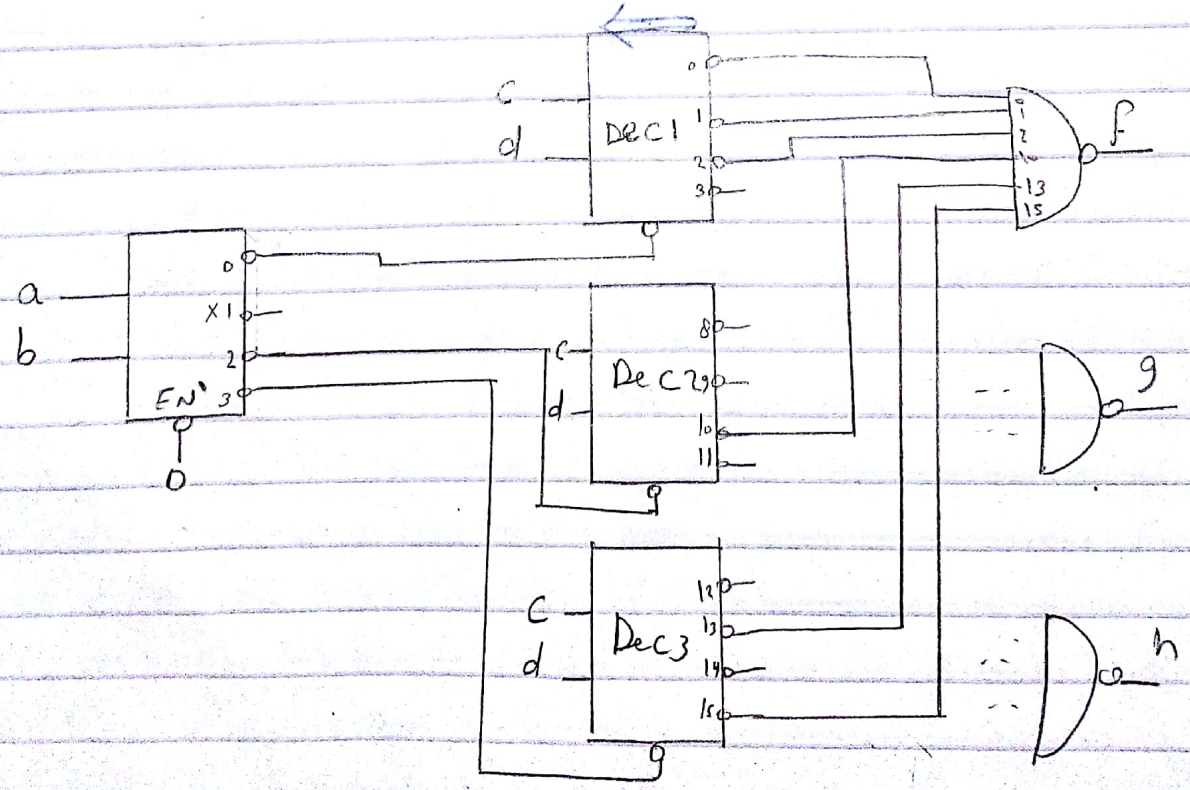
$f = \sum m(0,1,2,10,13,15)$

لاحظ أن (a'b) غير مستخدم في (h & g & f)

$\Rightarrow$ لذلك سنجعل (a,b) enables $\Leftarrow$

(c,d) inputs $\Leftarrow$

Enable signal a	b		inputs	outputs.
0	0	For Decoder 1	c d	0, 1, 2, 3
X	X	(Deleted)	c d	
1	0	For Decoder 2	c d	8, 9, 10, 11
1	1	For Decoder 3	c d	12, 13, 14, 15



b)

yz \ wx	$\bar{w}\bar{x}$	$\bar{w}x$	wx	$w\bar{x}$
$\bar{y}\bar{z}$	0	4	12	8
$\bar{y}z$	1	5	13	9
yz	3	7	15	11
$y\bar{z}$	2	6	14	10

f

$= \Sigma m(0,3,4,5,6,8,10,12)$

yz \ wx	$\bar{w}\bar{x}$	$\bar{w}x$	wx	$w\bar{x}$
$\bar{y}\bar{z}$	1	1	1	1
$\bar{y}z$	1	1		1
yz			1	
$y\bar{z}$			1	

g

$= \Sigma m(1,3,5,7,9,11,13,14)$

yz \ wx	$\bar{w}\bar{x}$	$\bar{w}x$	wx	$w\bar{x}$
$\bar{y}\bar{z}$	1	1	1	1
$\bar{y}z$		1	1	
yz		1	1	
$y\bar{z}$	1			1

h.

$= \Sigma m(0,2,4,5,7,8,10,12,13,15)$

⇐ في هذا المثال نجد أن ($\bar{y}\bar{z}$) مشتركة (كلهم واحد) في f & h وفارغة تمامًا في g

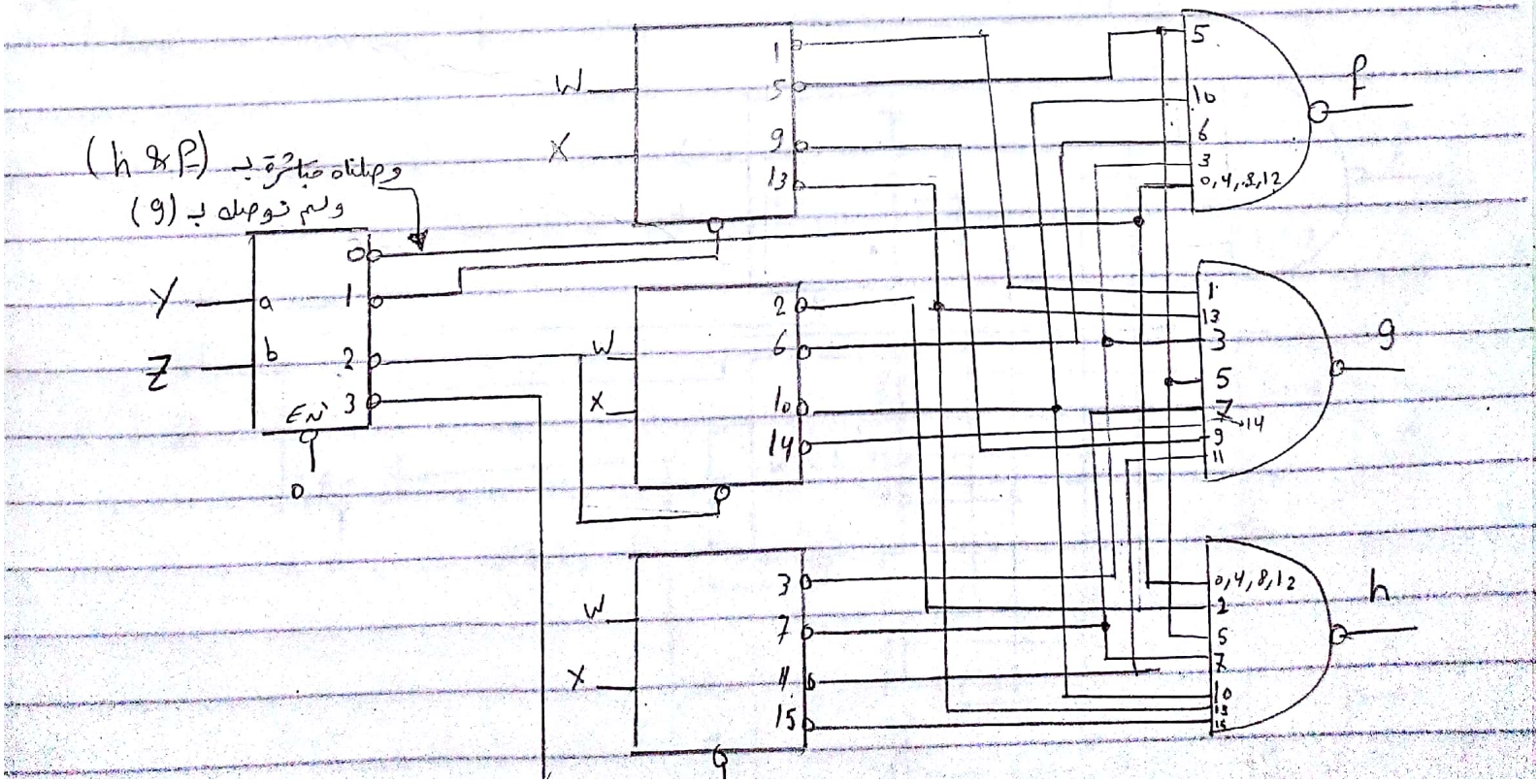
في هذه الحالة وظيفتنا نستخدم (Decoder) يعمل S (enable)

يمكننا حذف هذا الـ (Decoder).

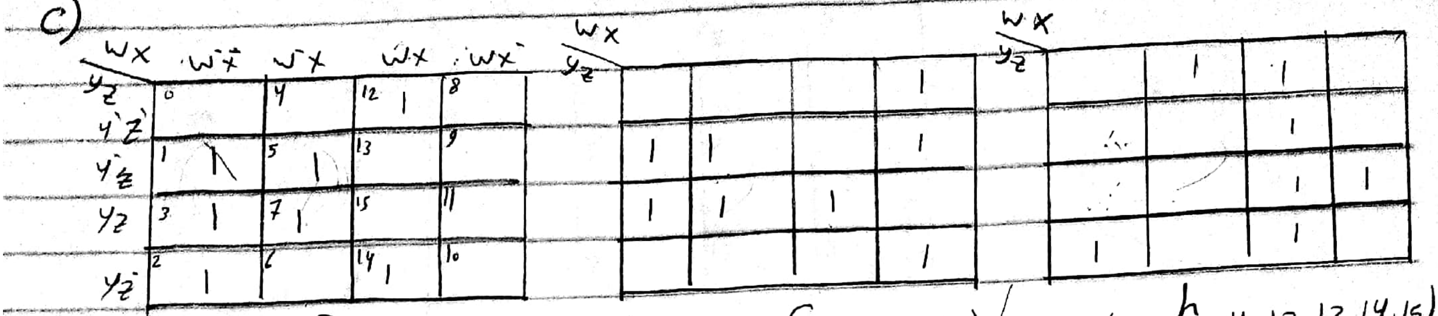
ونوصل الخرج مباشرة بالـ (enable) لأن كل مشتركة في f & g.

(NAND gate) نستخدم S (y, z) (enables) & S (w, x) (inputs)

Enable signal y	Enable signal z		inputs	output
0	0	(Deleted)	w x	0, 4, 8, 12 ← $\bar{y}\bar{z}$
0	1	Decoder1	w x	1, 5, 9, 13 ← $\bar{y}z$
1	0	Decoder2	w x	2, 6, 10, 14 ← $y\bar{z}$
1	1	Decoder3	w x	3, 7, 11, 15 ← yz



c)



f: $\Sigma m = (1,2,3,5,7,12,14)$

G: $= \Sigma m (1,3,5,7,8,9,10,15)$

h: $= \Sigma m (2,4,11,12,13,14,15)$

في هذا المثال نجد أن (1,3,5,7) موجودة تماماً في (f & g) وغير موجودة في (h)

لذلك (نتخلص) (1K.map) لتبسيط هذا الجزء (1,3,5,7) ← (W`Z)

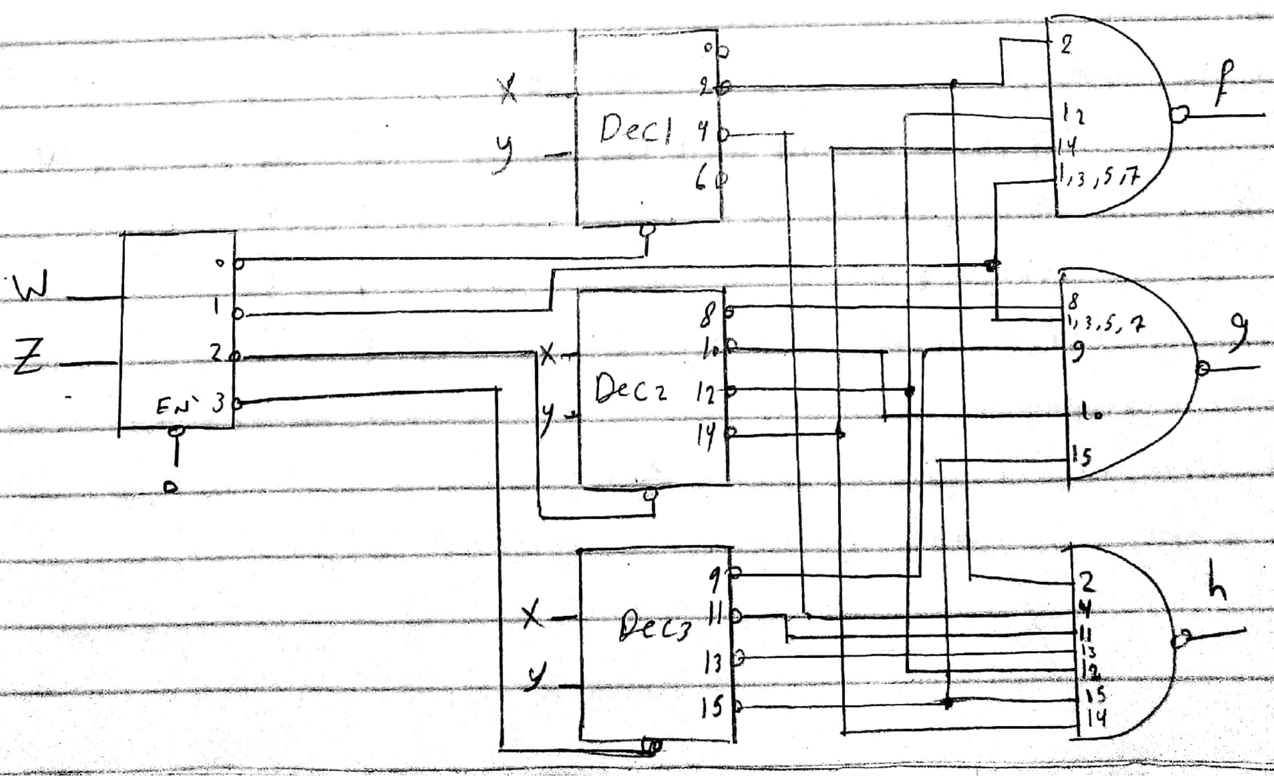
يعني هنستخدم (W,Z) كـ (enable) وعند (W`Z)

نحذف Decoder1

ونستخدم (X,Y) كـ (inputs)

Enable Signals W	Z		inputs	outputs	
0	0	Decoder 1	X Y	0,2,4,6	W`Z`
X 0	1	(Deleted)	~~XY~~	1,3,5,7	W`Z
1	0	Decoder 2	X Y	8,10,12,14	WZ`
1	1	Decoder3	X Y	9,11,13,15	WZ

توصيله بـ f&g



Scanned by CamScanner

YZ \ WX	W'X'	W'X	WX	WX'
Y'Z'	0	4	12	8
Y'Z	1	5	13	9
YZ	3	7	15	11
YZ'	2	6	14	10

⇐ هناك بعض الحالات حيث منظ بتخدم enables ك (W, X)

W, X → 00 The output will be (0, 1, 2, 3).

01 " " " (4, 5, 6, 7)

10 " " " (8, 9, 10, 11)

11 " " " (12, 13, 14, 15)

وعكس بتخدم enables 5 (Y, Z)

Y, Z → 00 The output will be (0, 4, 8, 12)

01 " " (1, 5, 9, 13)

10 " " (2, 6, 10, 14)

11 " " (3, 7, 11, 15)

وعكس بتخدم enables 5 (W, Y):

W, Y → W'_0 Y'_0 The output will be (0, 1, 4, 5)

W'_0 Y_1 " " (2, 3, 6, 7)

W_1 Y'_0 " " (8, 9, 12, 13)

W_1 Y_1 " " (10, 11, 14, 15)

⇐ أو نستخدم

(enables) S(W, Z)

W , Z ⇒ W̄ Z̄ 0 0 The output will be → (0, 2, 4, 6)

W̄ Z 0 1 " " → (1, 3, 5, 7)

W Z̄ 1 0 " " → (8, 10, 12, 14)

W Z 1 1 " " → (9, 11, 13, 15)

⇐ أو نستخدم

(enables) S (X , y)

X , y → X̄ ȳ 0 0 The output will be → (0, 1, 8, 9)

0 1 " " → (2, 3, 10, 11)

1 0 " " → (4, 5, 12, 13)

1 1 " " → (6, 7, 14, 15)

⇐ أو نستخدم

(enables) S (X , Z)

X , Z → 0 0 The output will be → (0, 2, 8, 10)

0 1 " " → (1, 3, 9, 11)

1 0 " " → (4, 6, 12, 14)

1 1 " " → (5, 7, 13, 15)

• نحصل على الـ (output) بأننا نحوس في الـ (K.map) فمثلاً بـ X , Z عندما تكون (0 0) يعني (X̄ Z̄) ونشوف الأرقام عند (X̄ Z̄) نجدها عند (0, 2, 8, 10)

على الـ K.map

• عند حل أي سؤال (تمثيل) (Function) على (Decoder) ننظر لو هناك أي جزء مشترك بين الـ (Functions) محذوف أو موجود بالكامل ومنه هنحدد إحنا هنستخدم أي (enables)

D)

yz \ wx	$w'x'$	$w'x$	wx	wx'
$y'z'$	0	4	12	8
$y'z$	1	5	13	9
yz	3	7	15	11
yz'	2	6	14	10

yz \ wx				
			1	1
		1	1	1
	1	1	1	
			1	

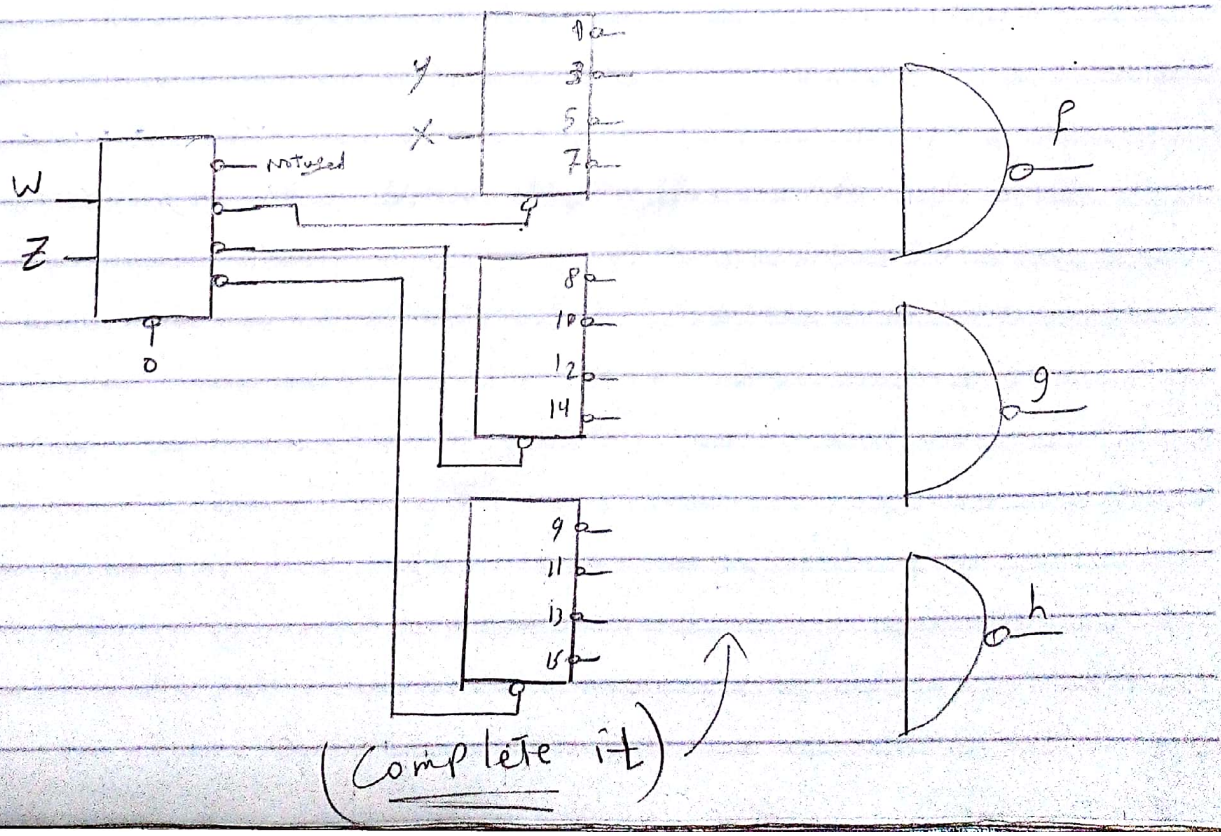
yz \ wx				
			1	
		1	1	
	1		1	1
			1	1

(function) كما نلاحظ عدم استخدام $(w'z')$ ⇐ والذي يمثل $(0, 2, 4, 6)$

معنى ذلك أننا سنستخدم (w,z) كـ (enables)

(y,x) كـ (inputs)

Enable signals w z		inputs	outputs
x 0 0	(Deleted)	y x	0, 2, 4, 6 ← $w'z'$
0 1	Decoder1	y x	1, 3, 5, 7 ← $w'z$
1 0	Decoder2	y x	8, 10, 12, 14 ← wz'
1 1	Decoder3	y x	9, 11, 13, 15 ← wz



e)

c'd' \ ab	a'b'	a'b	ab	ab'
c'd'	0	4 1	12 1	8
c'd	1	5 1	13 1	9
cd	3	7	14	11 1
cd'	2 1	6 1	15	10

	1	1	
	1	1	
			1
	1	1	

	1	1	1
	1	1	
	1		
1			1

⇐ بعدك (K-map) في (4, 5, 12, 13) أن عناصر خاصة في كل (Functions)
(b c') ⇐ والذي عبارة عن
(enables) S (b, c) ⇐ لذلك سنستخدم
inputs. S (a, d)

Enable Signals b	c		inputs	outputs	
0	0	Decoder1	a d	0, 1, 8, 9	↵ b'c'
0	1	Decoder2	a d	1, 3, 10, 11	↵ b'c
X 1	0	Deleted.	a d	4, 5, 12, 13	↵ bc'
1	1	Decoder3	a d	1, 7, 14, 15	↵ bc

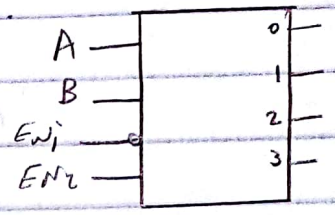
⇒ You Draw the Design.

EX: (Final)

We have found a minimum Sum of Products expressions for each of two functions F and G. For these functions, We have available as many of the decoders described below as are needed plus 2 eight-input OR gates. Show a block diagram for this implementation. With only these components.

⇒ All inputs are available both uncomplemented and complemented

⇒ 5 point Extra Credit: Show a diagram that uses only 2-eight-input OR gates and 3 of these decoders.



$F = A'B + BD + AB'C'D'$

$G = B'D' + BC + ABD'$

Solution

AB \ CD	A'B'	A'B	AB	AB'
C'D'	0	4 1	12	8 1
C'D	1	5 1	13 1	9
CD	3	7 1	15 1	11
CD'	2	6 1	14	10

	A'B'	A'B	AB	AB'
C'D'	1		1	1
C'D				
CD		1	1	
CD'	1	1	1	1

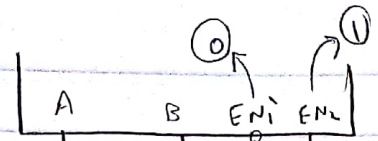
$F = \Sigma m(4,5,6,7,8,13,15)$

$G = \Sigma m(0,2,6,7,8,10,12,14,15)$

⇐ نلاحظ أن (1,3,9,11) ← فارغ تماماً في (F & G) والذي يمثل $B'D$

↩ لذلك سنستخدم (B, D) 5 (enables) و (A, C) 5 inputs.

⟹

Enable Signals B D		OUTPUTS
0 0 Dec1	A C B D`	(0, 2, 8, 10) ← B`D`
0 1	(Deleted)	
1 0 Dec2	A C B` D`	(4, 6, 12, 14) ← BD`
1 1 Dec3	A C B` D	(5, 7, 13, 15) ← B`D