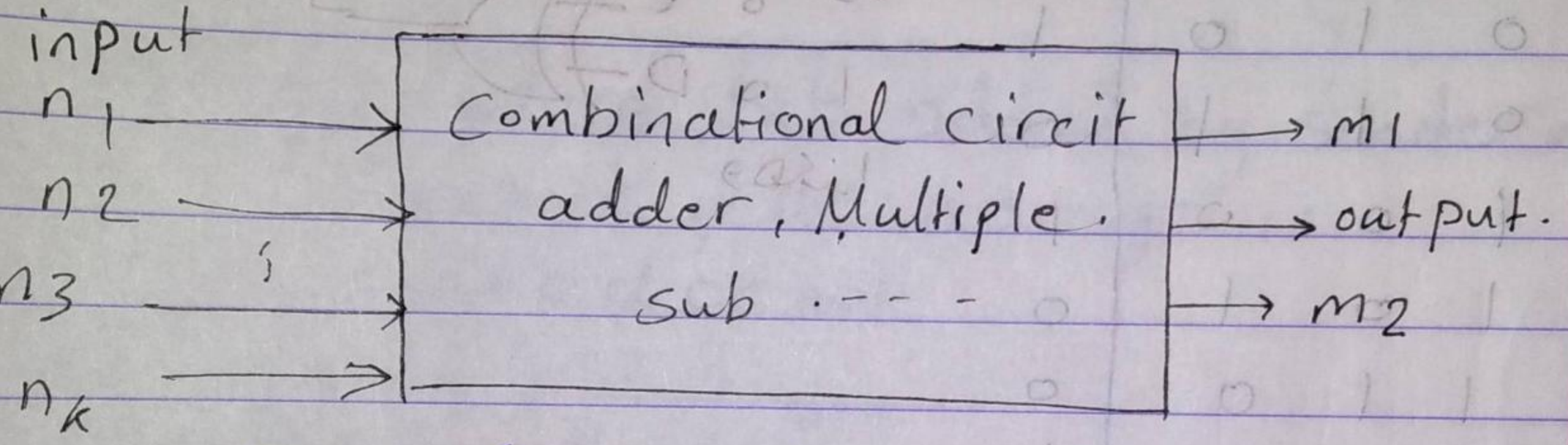
Chapter 4

Combinational logic circit (Design).

* There are two types of digital circit :-

① Combinational logic circit :- (ch4).

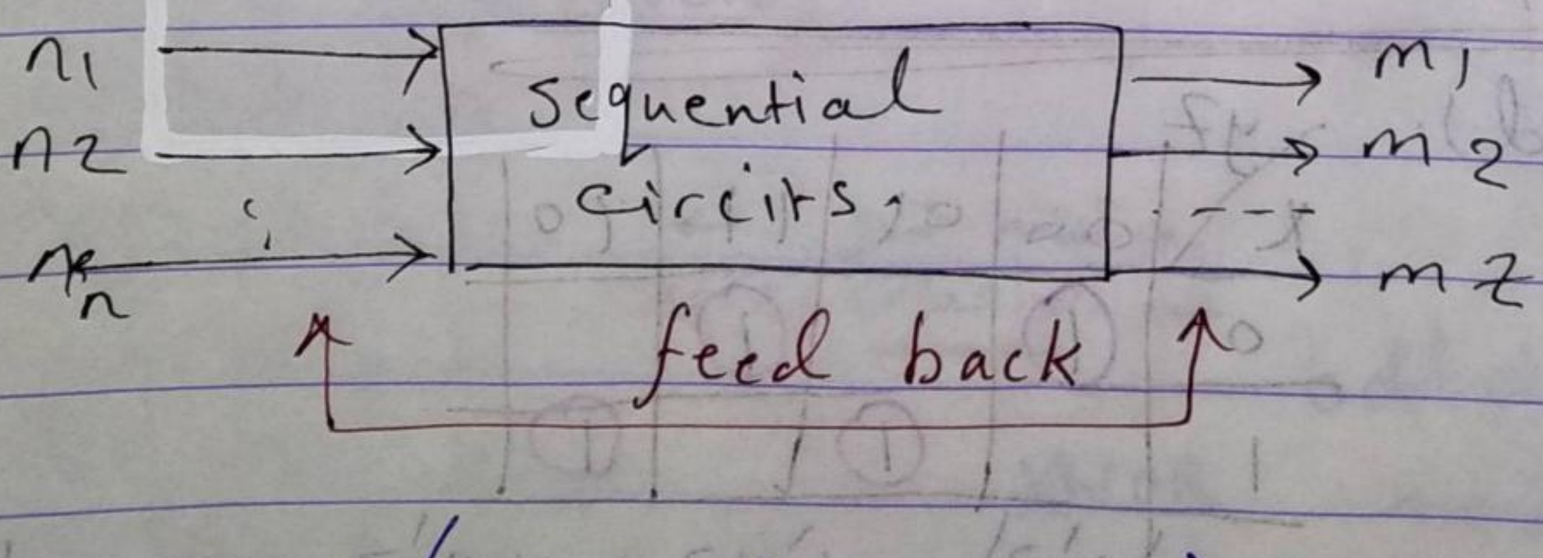
↳ output depends only in inputs.



$$\underbrace{m_1}_{\text{output}} = \underbrace{(n_1 \cdot n_2) + n_3}_{\text{inputs.}}$$

② Sequential circit (ch5, ch6).

output depend on input and previous output

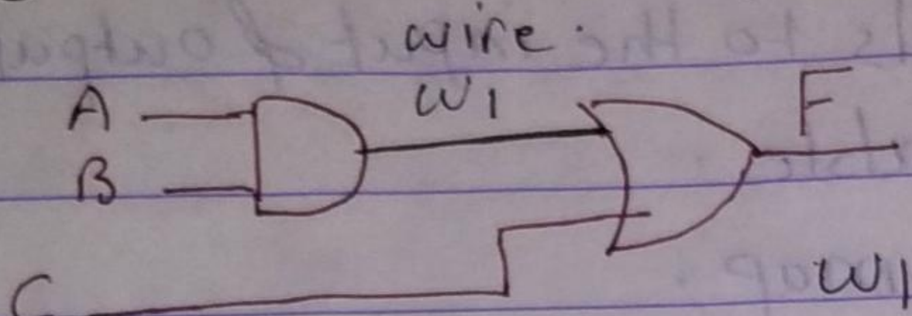


$$\underbrace{m_1}_{\text{output.}} = \underbrace{(n_1 + n_2 + m_3)}_{\text{input}} \cdot \underbrace{m_2}_{\text{output}}$$

Combinational Circits :-

Analysis ⟹ output ⟵ inputs

ex

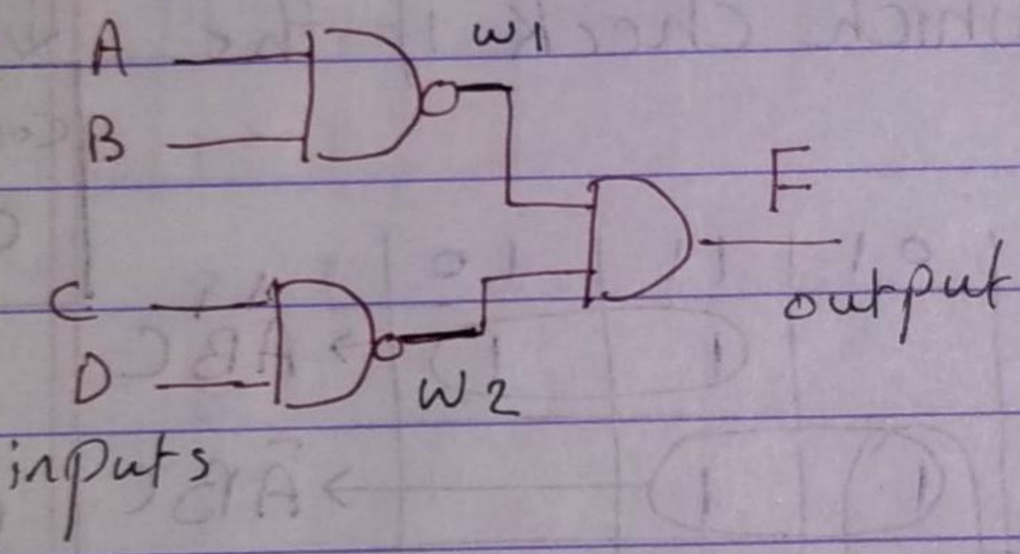


$w_1 = A \cdot B$.

$F = w_1 + C$.

$= \boxed{(A \cdot B) + C}$ → Analysis.

ex



$w_1 = (A \cdot B)'$

$= A' + B'$

$w_2 = (C \cdot D)'$

$= C' + D'$.

$F = w_1 \cdot w_2$

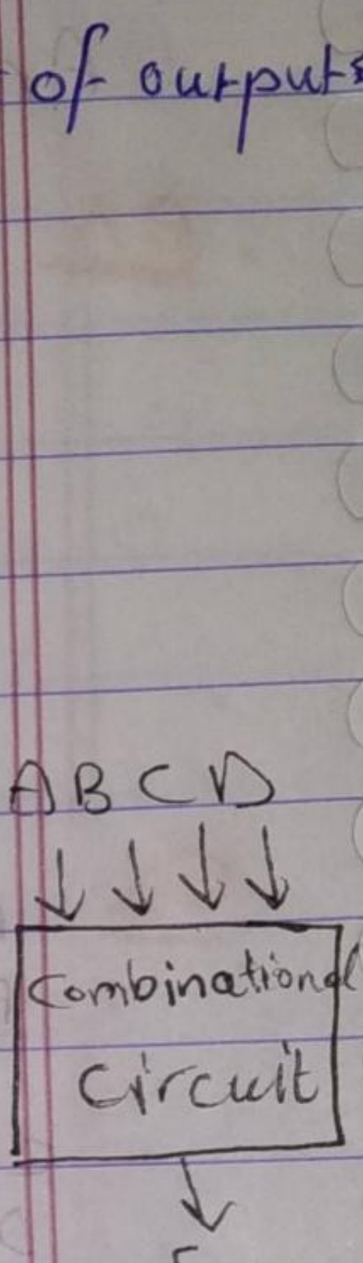
$F = (A' + B') \cdot (C' + D')$.

$F = \boxed{A'C' + A'D' + B'C' + B'D'}$. ← Analysis.

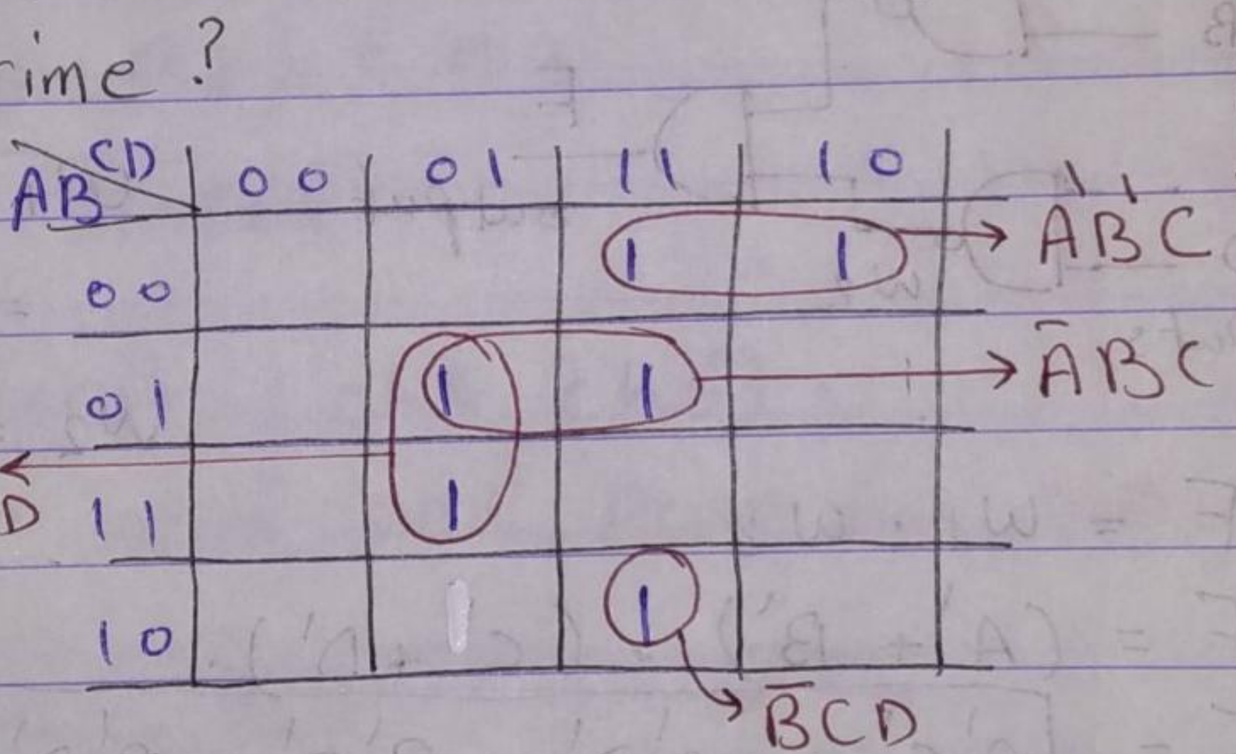
Design Procedure :-

① - Read the problem carefully.
② - Determine the number of inputs and number of outputs
③ - Assign letter symbols to the input & output.
④ - Derive the truth table.
⑤ - Simplify using K-map.
⑥ - Draw the circit.

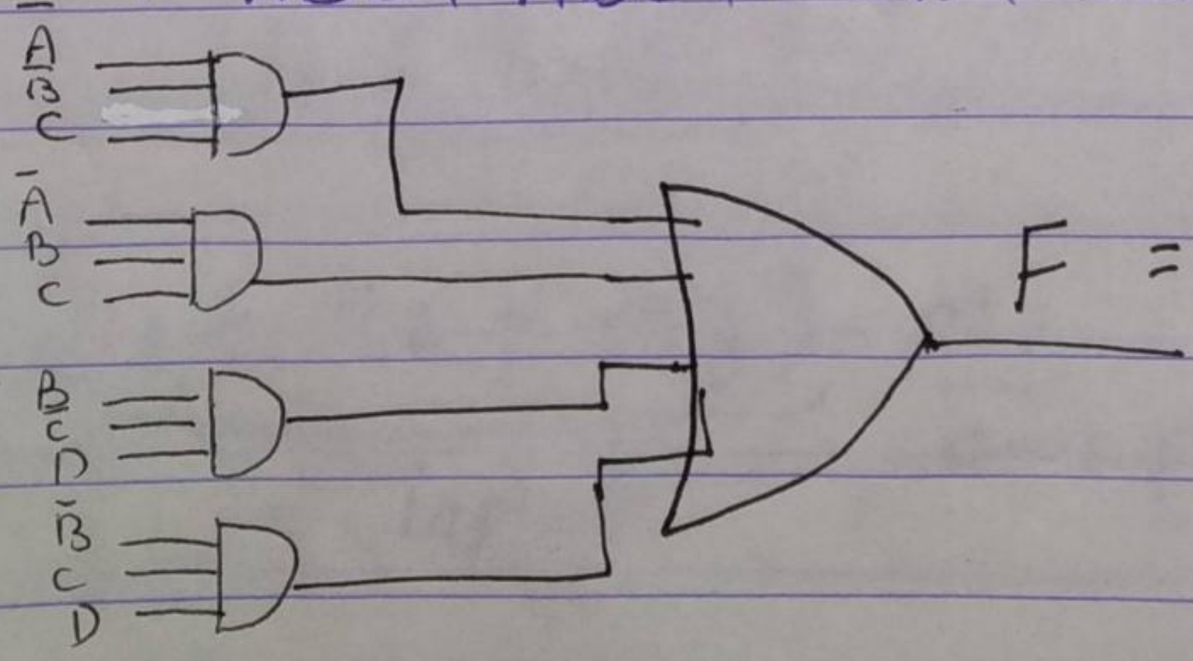
ex Design a combinational circit that takes 4 bit input number which check if the number is prime?



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

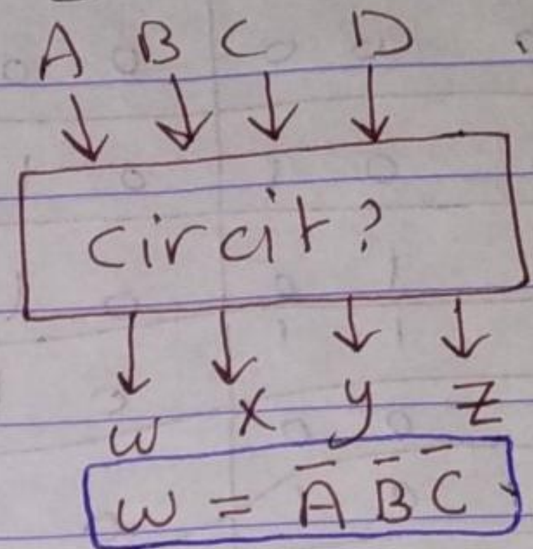


$$F = \bar{A}\bar{B}C + \bar{A}BC + B\bar{C}D + \bar{B}CD.$$



ex Design a combinational circit that gererate the 9's complement of a BCD digit?

A	B	C	D	W	X	y	z
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
x	x	x	x	x	x	x	x



$w = \bar{A}\bar{B}C$

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

$x = B\bar{C} + \bar{B}C$

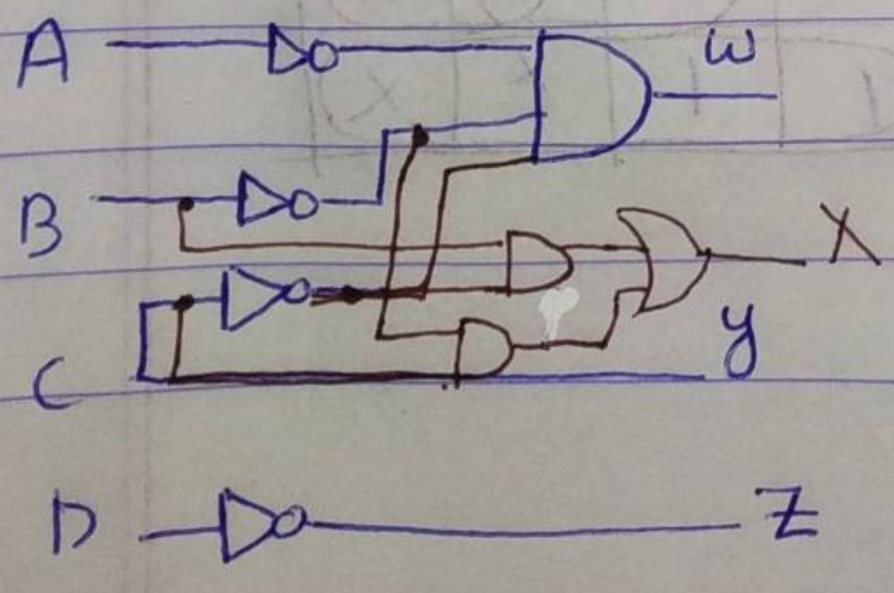
AB \ CD	00	01	11	10
00			1	1
01	1	1		
11	x	x	x	x
10			x	x

$y = C$

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

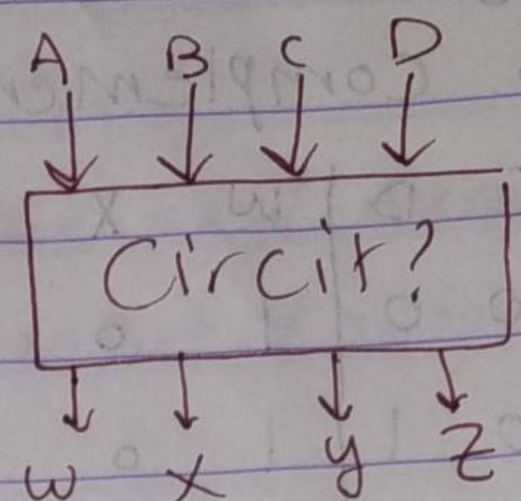
$z = \bar{D}$

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



ex Design a BCD to excess-3 Code Conversion?

A	B	C	D	W	X	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x



$Z = D'$

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

$y = C'D' + CD$

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

$X = B'C + B'D + BC'D'$

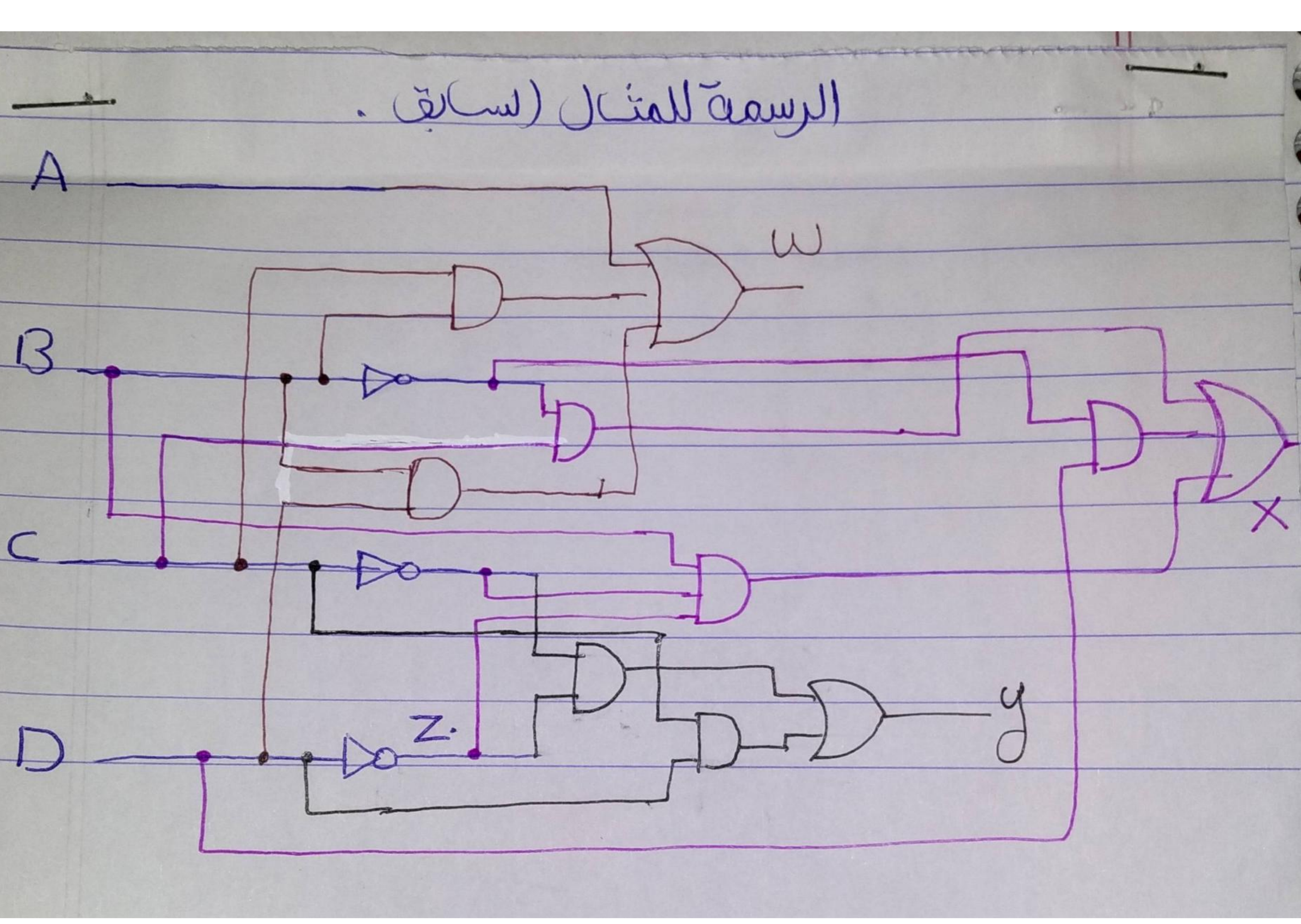
AB \ CD	00	01	11	10
00		1	1	1
01	1			
11	x	x	x	x
10	1	x	x	

$\bar{B}C$, $\bar{B}D$, $B\bar{C}\bar{D}$

$W = A + BC + BD$

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

الرسمة للمثال (سابق).



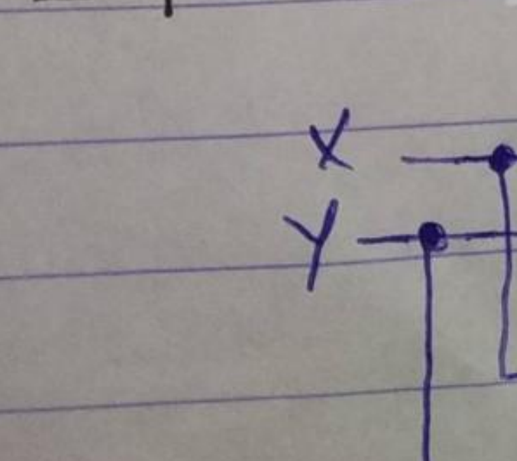
Binary Adder & Subtractor :-

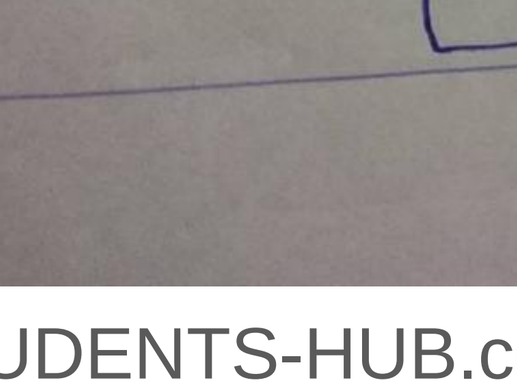
4 Possible operation for addition two Binary digit.

0 + 0	0 + 1	1 + 0	1 + 1
0 0	0 1	0 1	1 0
Carry sum	Carry sum	carry sum	Carry sum.

Half Adder (HA) :- Combinational curcuit that perform addition of two digits.

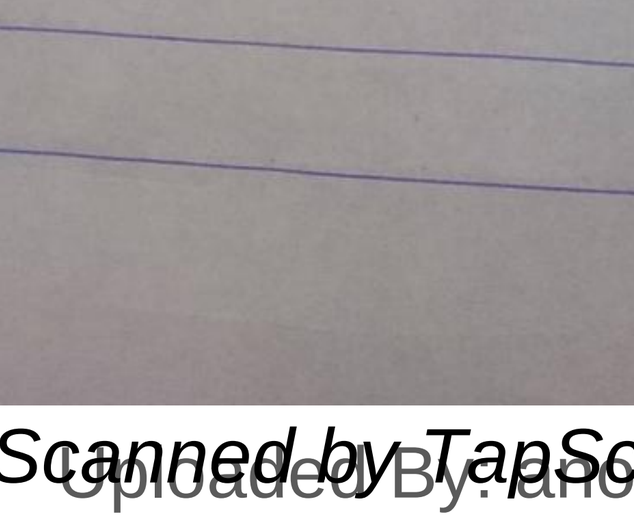
X	y	sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





$Sum = X'y + XY'$

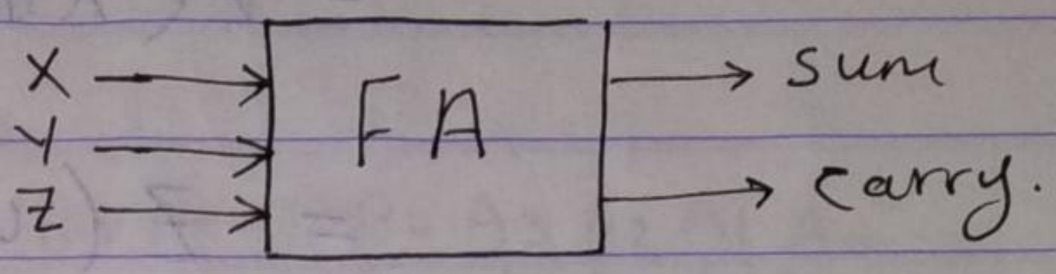
$Sum = X \oplus Y$



$Carry = X \cdot y$

Full Adder (FA) :- Combinational circuit that perform addition of three bits.

X	Y	Z	sum	Carry
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

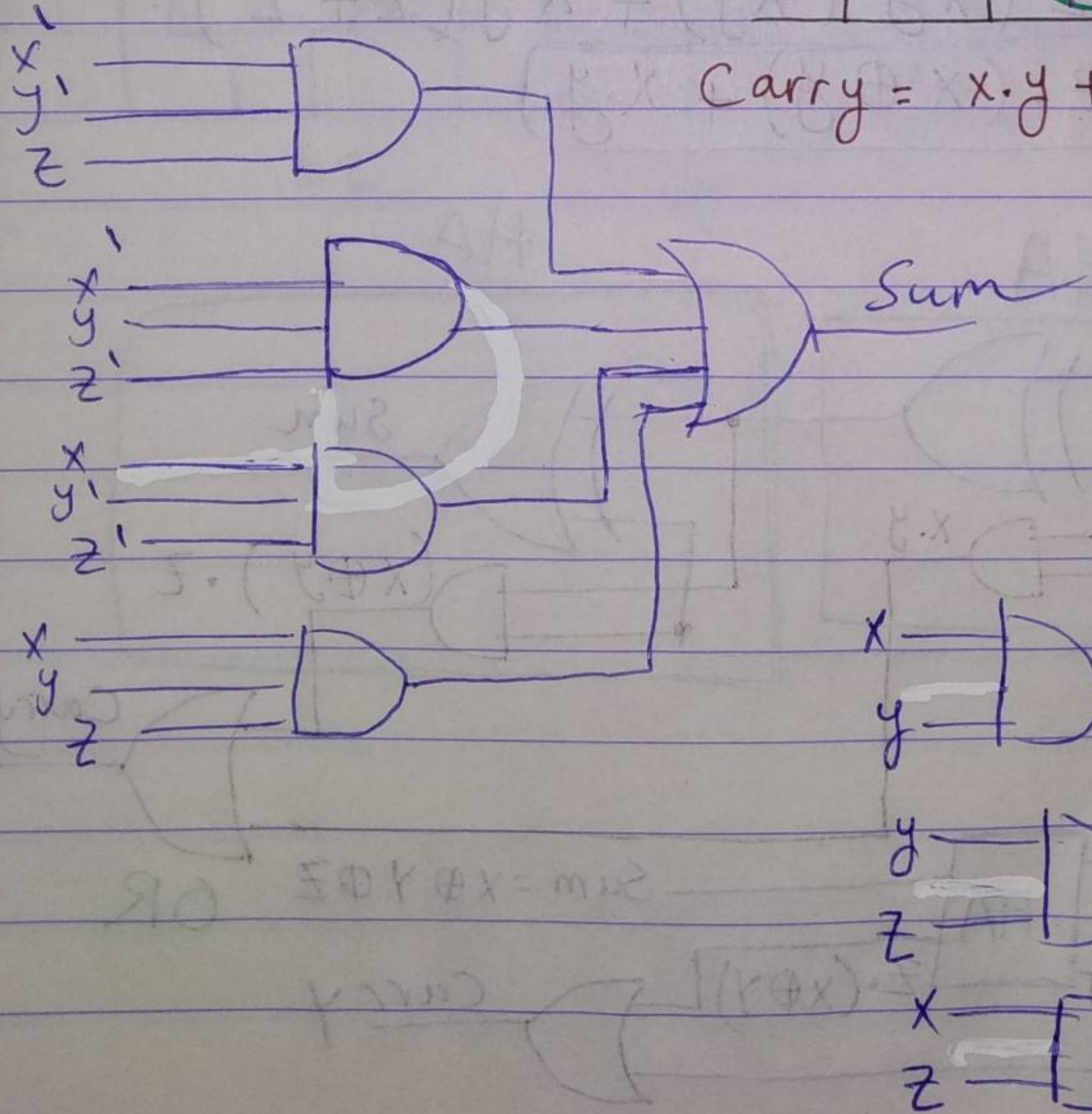


X \ YZ	00	01	11	10
0		(1)		(1)
1	(1)		(1)	

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

X \ YZ	00	01	11	10
0			1	
1		1	1	1

$Carry = x.y + y.z + X.z$



تكملة المثال

للتبسيط :-

$$Sum = z(x'y' + xy) + z'(x'y + xy')$$

$$= z(x \oplus y)' + z'(x \oplus y)$$

$$= z(w)' + z'(w)$$

$$= z \oplus w$$

$$\boxed{Sum} = z \oplus x \oplus y = \boxed{x \oplus y \oplus z}$$

Carry = بنوخذ كل رقم كالة

$$Carry = x'yz + xy'z + xyz + xyz'$$

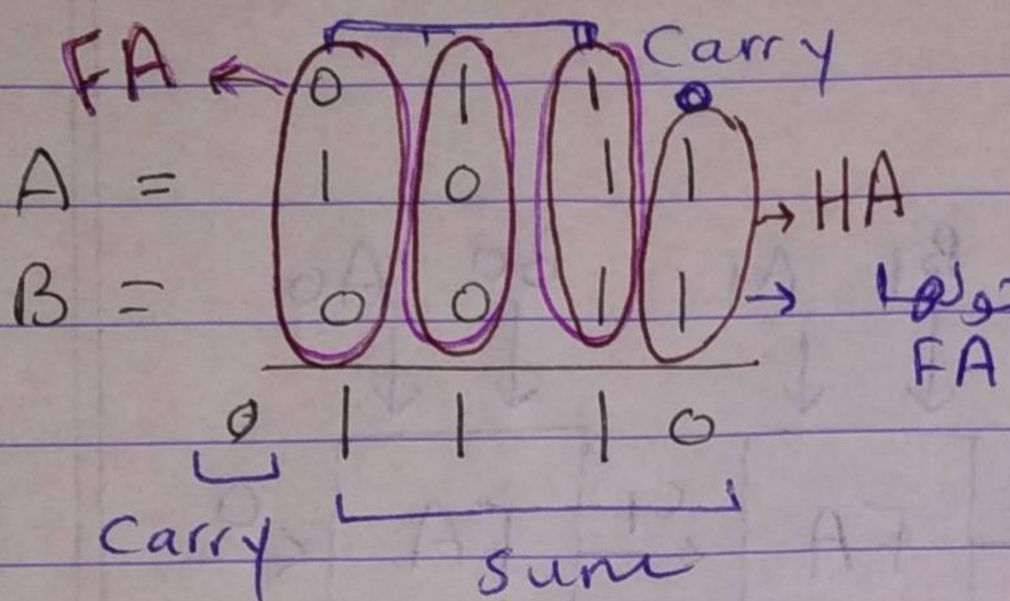
$$= z(x'y + xy') + x \cdot y(z + z')$$

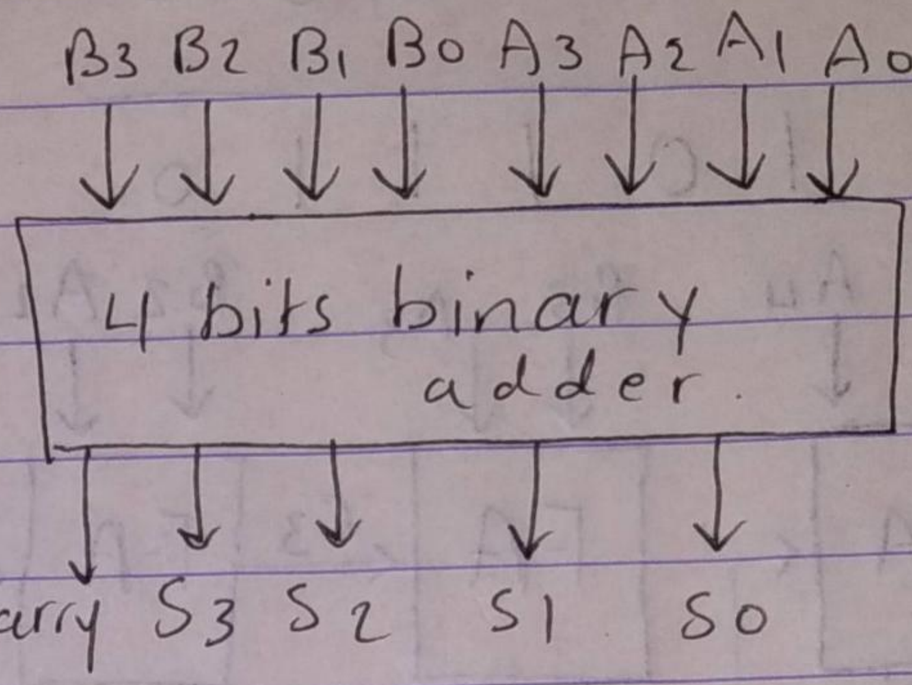
$$\boxed{Carry = z \cdot (x \oplus y) + x \cdot y}$$

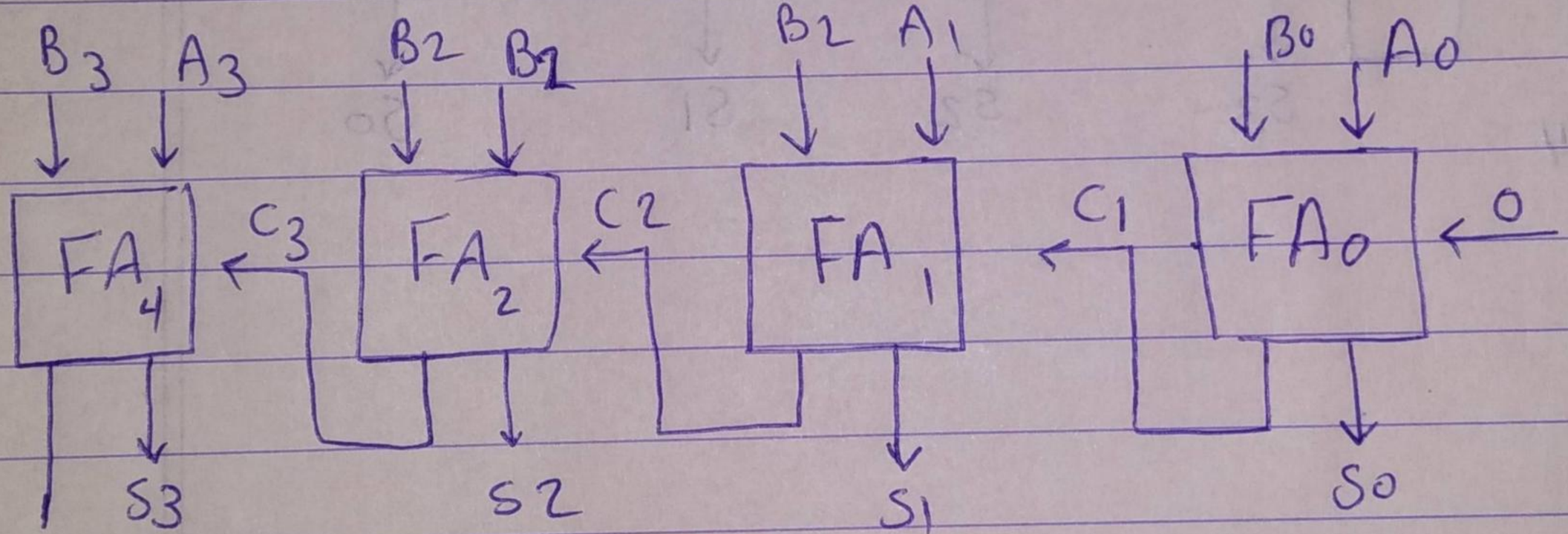
Binary Adder :- 4 bits binary adder.

ex $A = A_3\ A_2 A_1 A_0$.

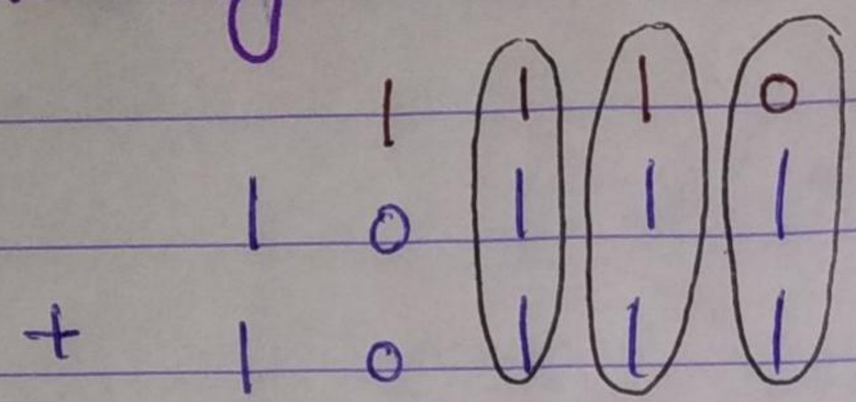
$B = B_3\ B_2\ B_1\ B_0$.

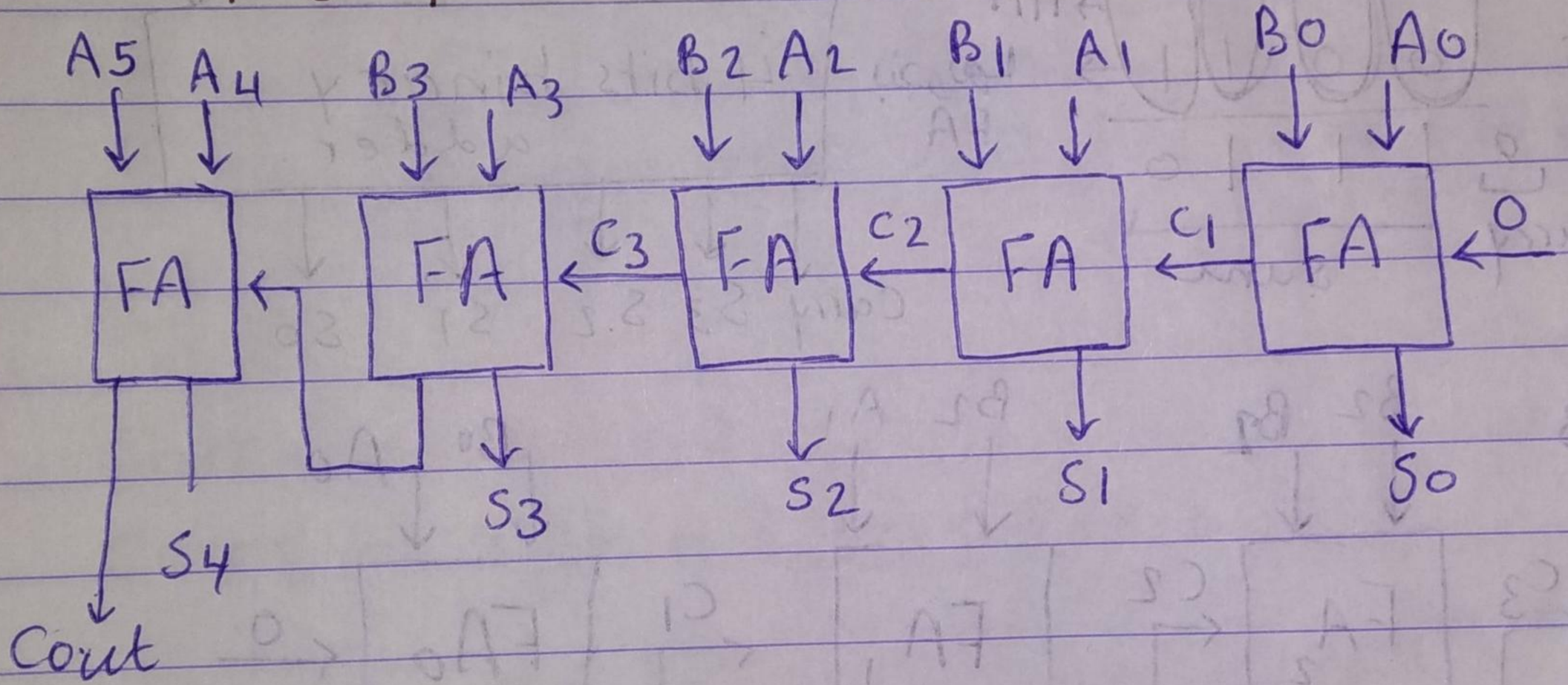






* Design 5 bits binary adder :-

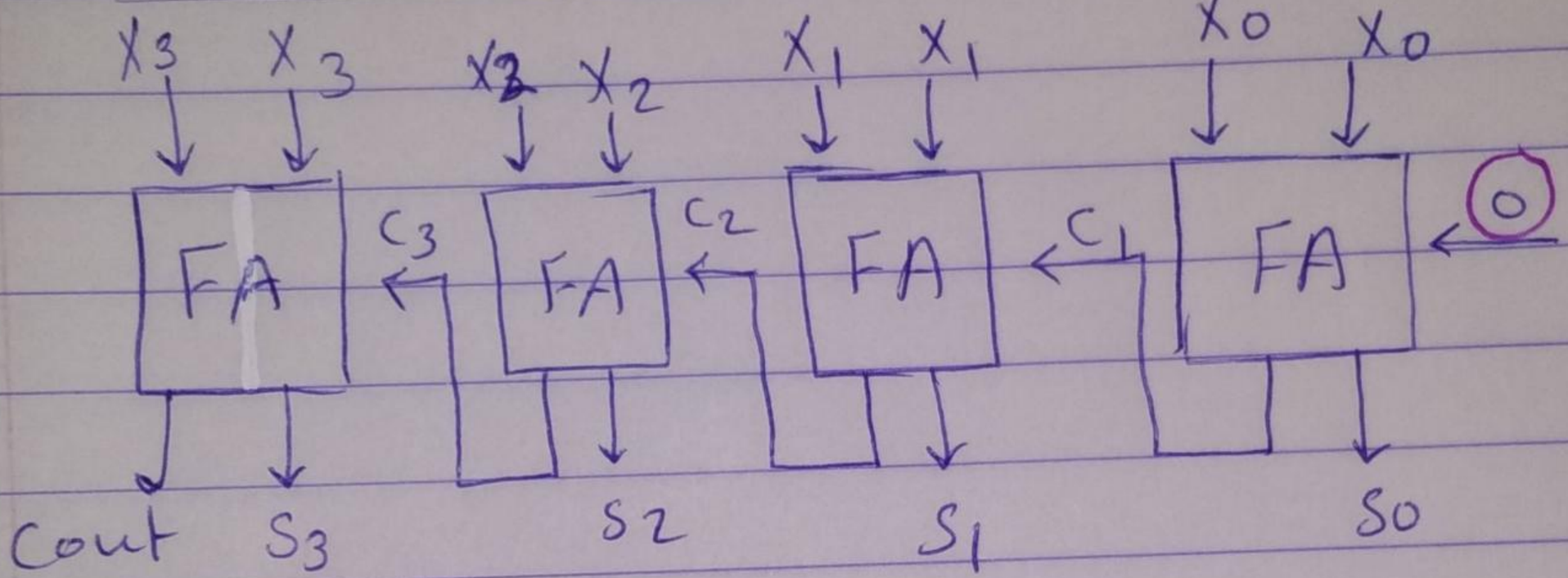




* Design Circuit that do the following operation $2X$ $X = 4$ bits.

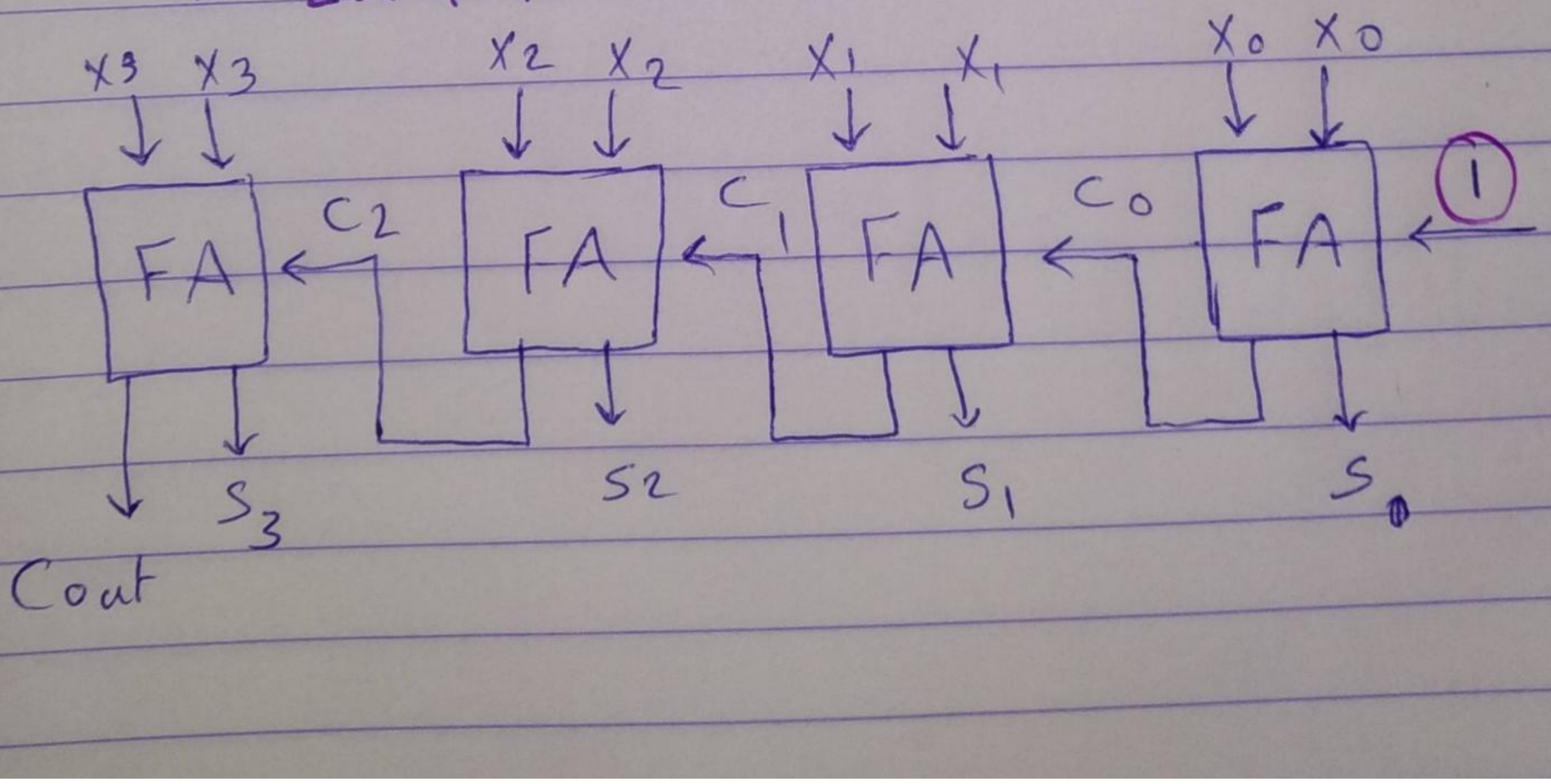
↳ $= X + X$.

$= [\ X_3\ X_2\ X_1\ X_0\ ]$.
$+\ X_3\ X_2\ X_1\ X_0$



ex Design acircit that do the following.

$2X + 1$ $X = 4$ bits.

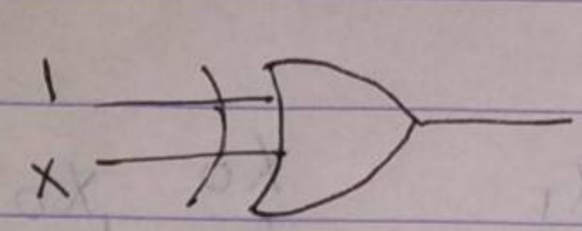


Binary Subtractor

$A - B = A + 2's\ compl.$

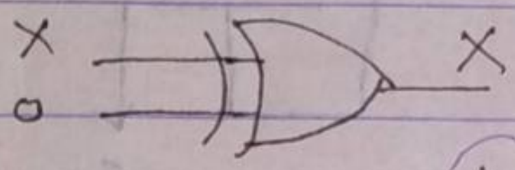
$= A + (1st\ compl + 1)$.

Note



$1 \oplus x \Rightarrow 1 \oplus 0 = 1$

$\hookrightarrow 1 \oplus 1 = 0$ $\Big\} x'$



$1 \oplus 0 = 1$

$0 \oplus 0 = 0$ $\Big\}$ x

ex Design 4 bit binary subtraktor

$A = 1011$ $\quad S = A - B = A + 2's\ compl.$

$B = 0011$ $\quad = A + 1^{st}\ Compl + 1$

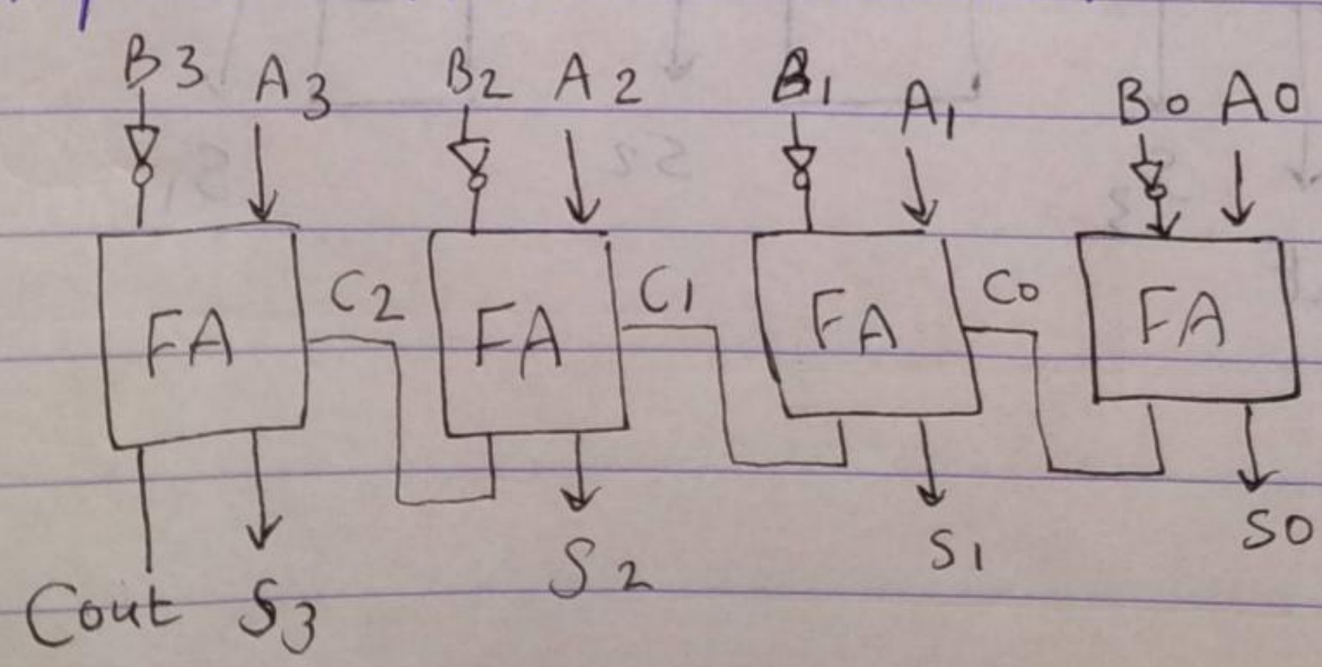
$2's\ compl\ B = 1100 + 1 = 1101$

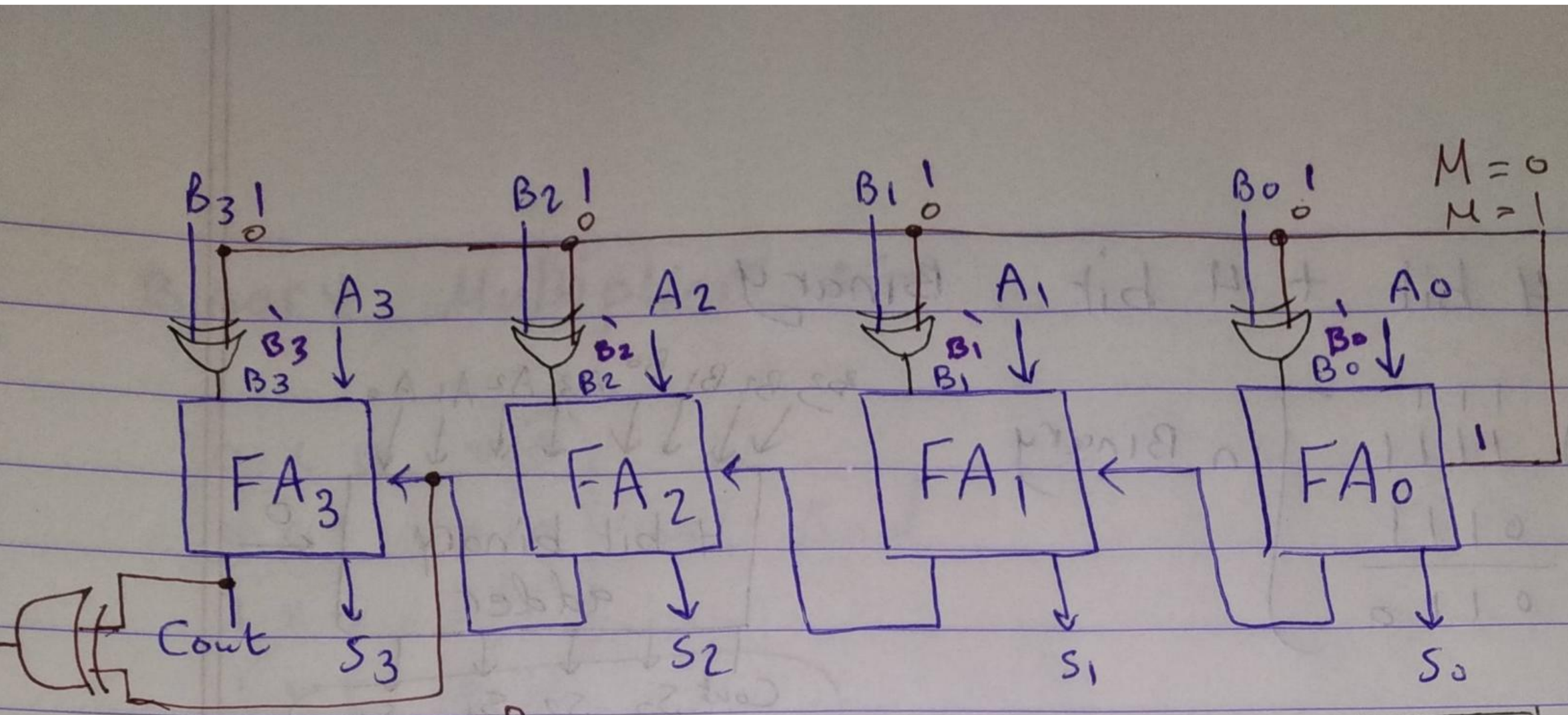
```
 1 1 1 0
  1011
+ 1101
------
 11000
```

$A + B' + 1$

$A + 2's\ compl$

$A - B'$





A+B

M = 0

Adder

$A_3\ A_2\ A_1\ A_0$

$B_3\ B_2\ B_1\ B_0$

Adder.

M = 1 **A − B**

$A_3\ A_2\ A_1\ A_0$

$B_3'\ B_2'\ B_1'\ B_0'$

Adder + Subtractor.

لبحط XOR بالآخر عشان نفحص اذا في overflow أو لاء.
(إذا كانوا متشابهات ما في overflow واذا مختلفات في).

Decimal Adder :— (BCD adder).

BCD → 0 - 9 valid 10 - 15 unvalid.

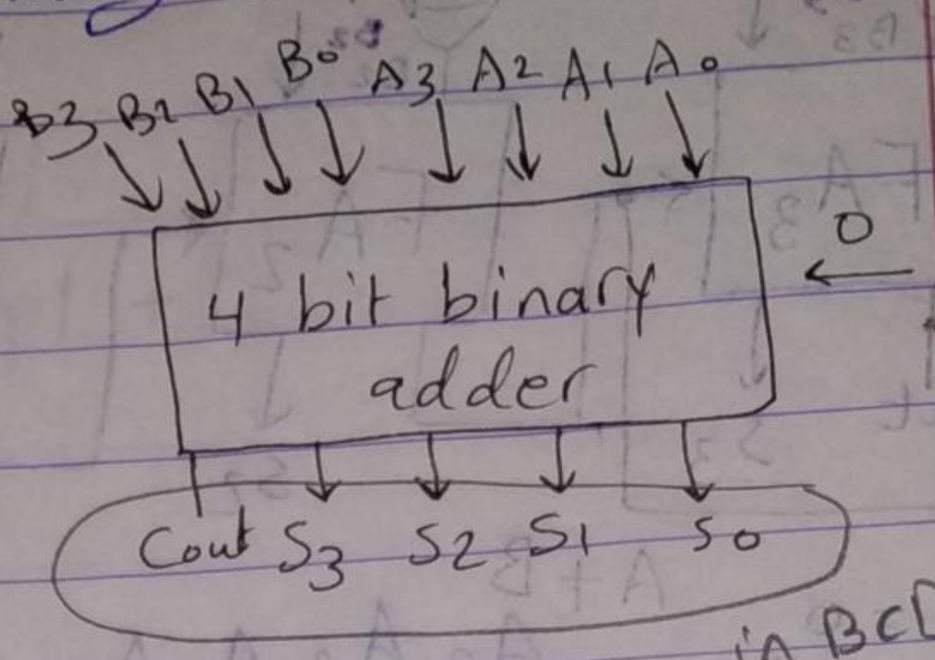
BCD + BCD Check sum < 9 OK.

BCD + BCD check sum > 9 + 6.

* 4 bit + 4 bit Binary

A 1111
B 0111
10110] in Binary

Cout



in BCD

≥ 9
$+6$

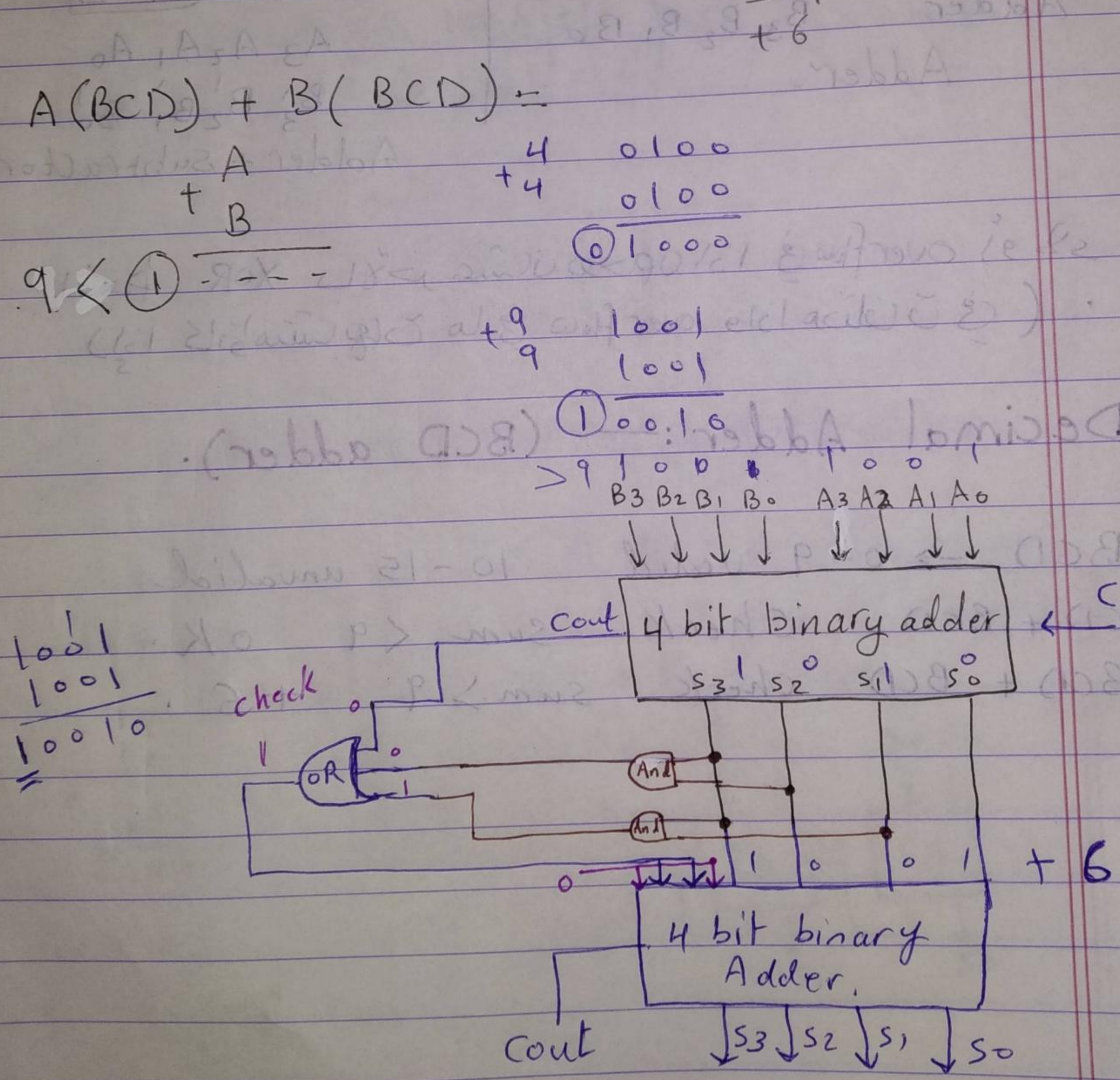
A(BCD) + B(BCD) =

A
+ B
---
9 < (1) - - - -

4 0100
+4 0100
---
(0)1000

9 1001
+9 1001
---
(1)0010

>9 1 0 0 1 0 0 1 0

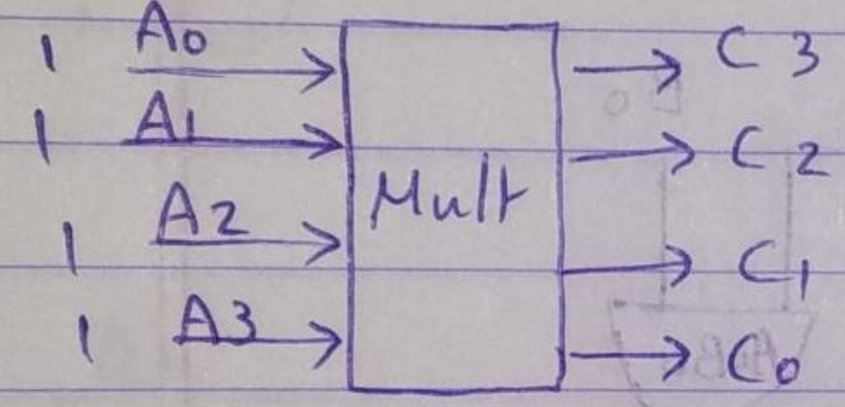


Binary Multiplier :-

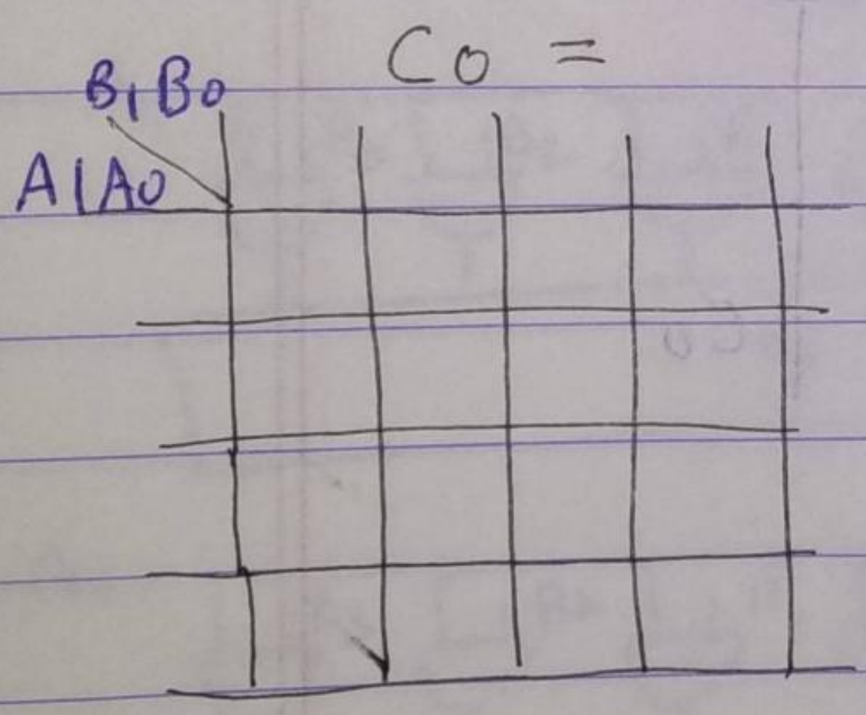
0 x 0	=	0
0 x 1	=	0
1 x 0	=	0
1 x 1	=	1

← And

ex Design 2 bit by 2 bit binary multiplier.



$C_0 =$



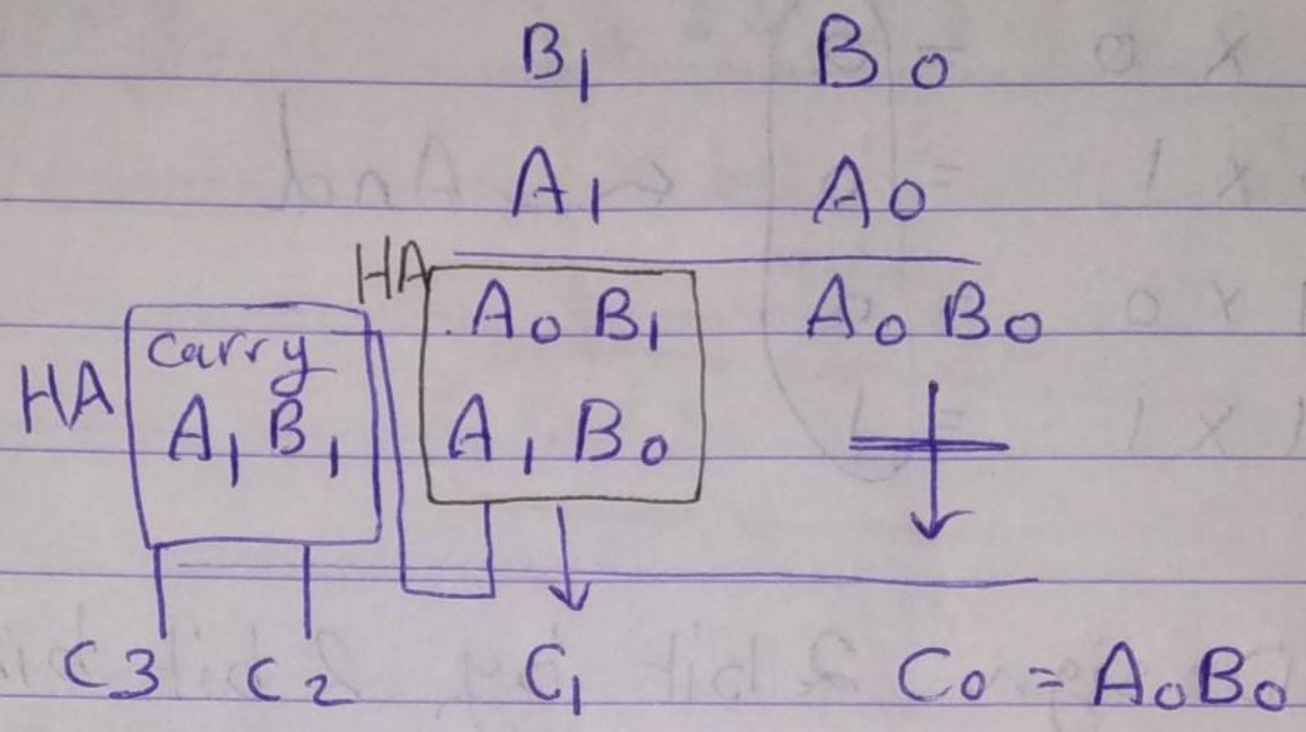
وبنكمل ل

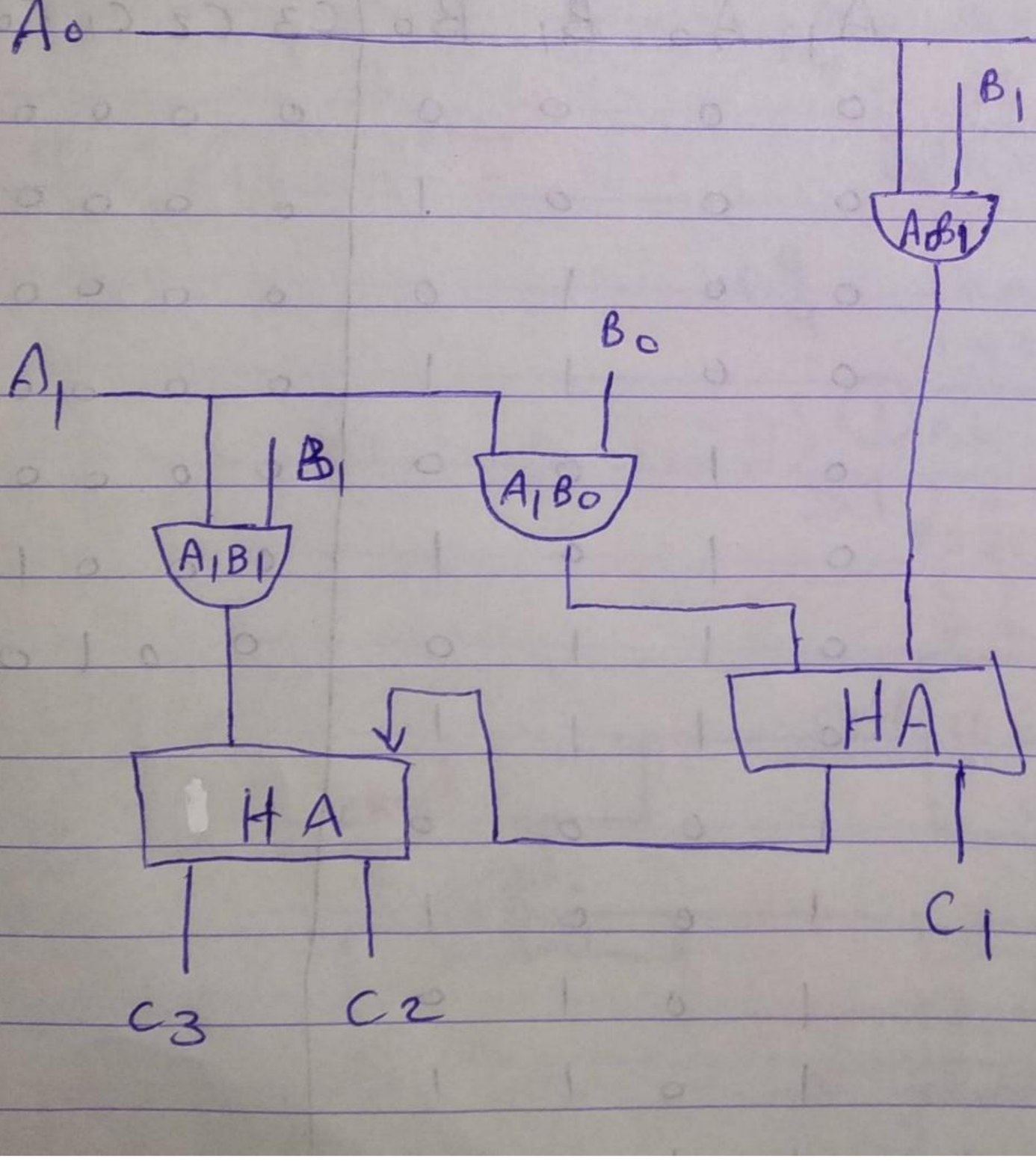
C_1, C_2, C_3

بس هاي الطريقة مش efficient (فعّالة).

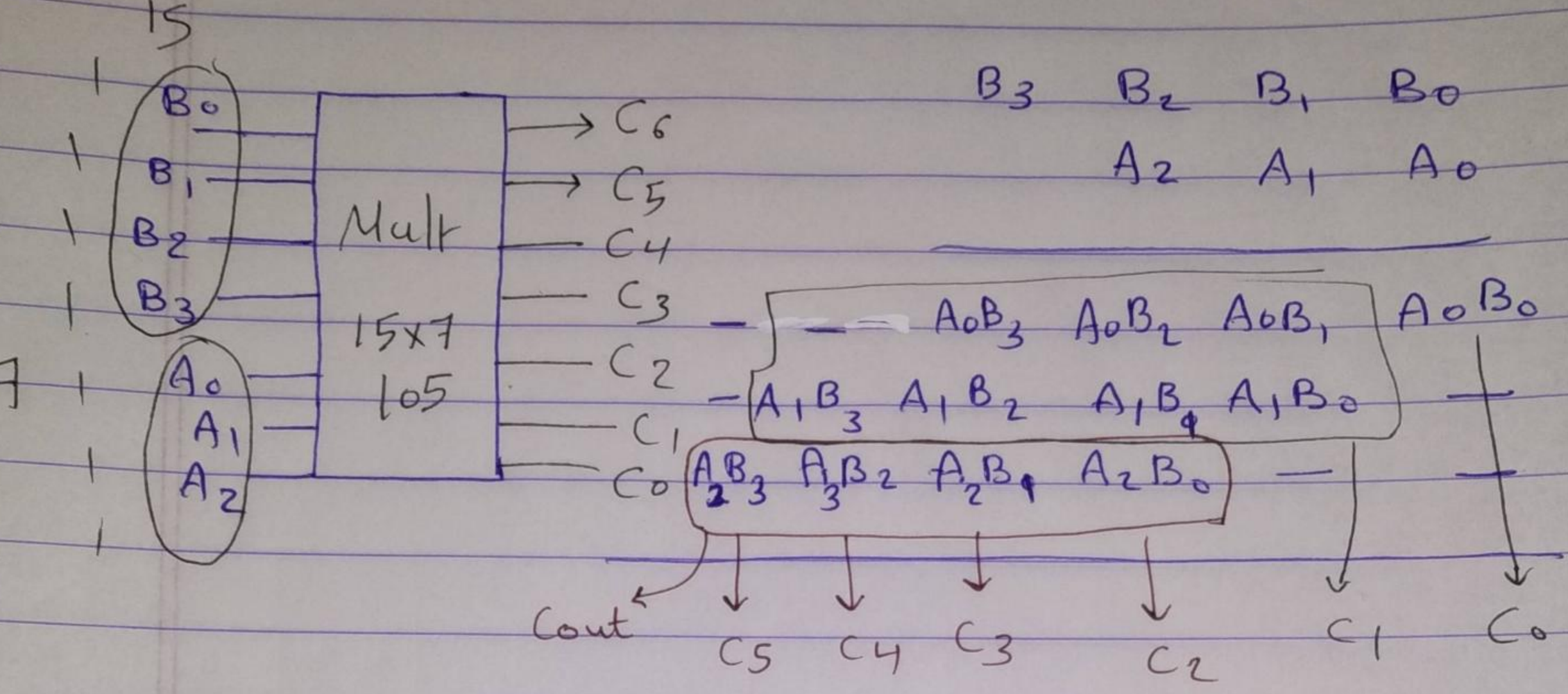
A_1	A_0	B_1	B_0	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1	1	0	0	1

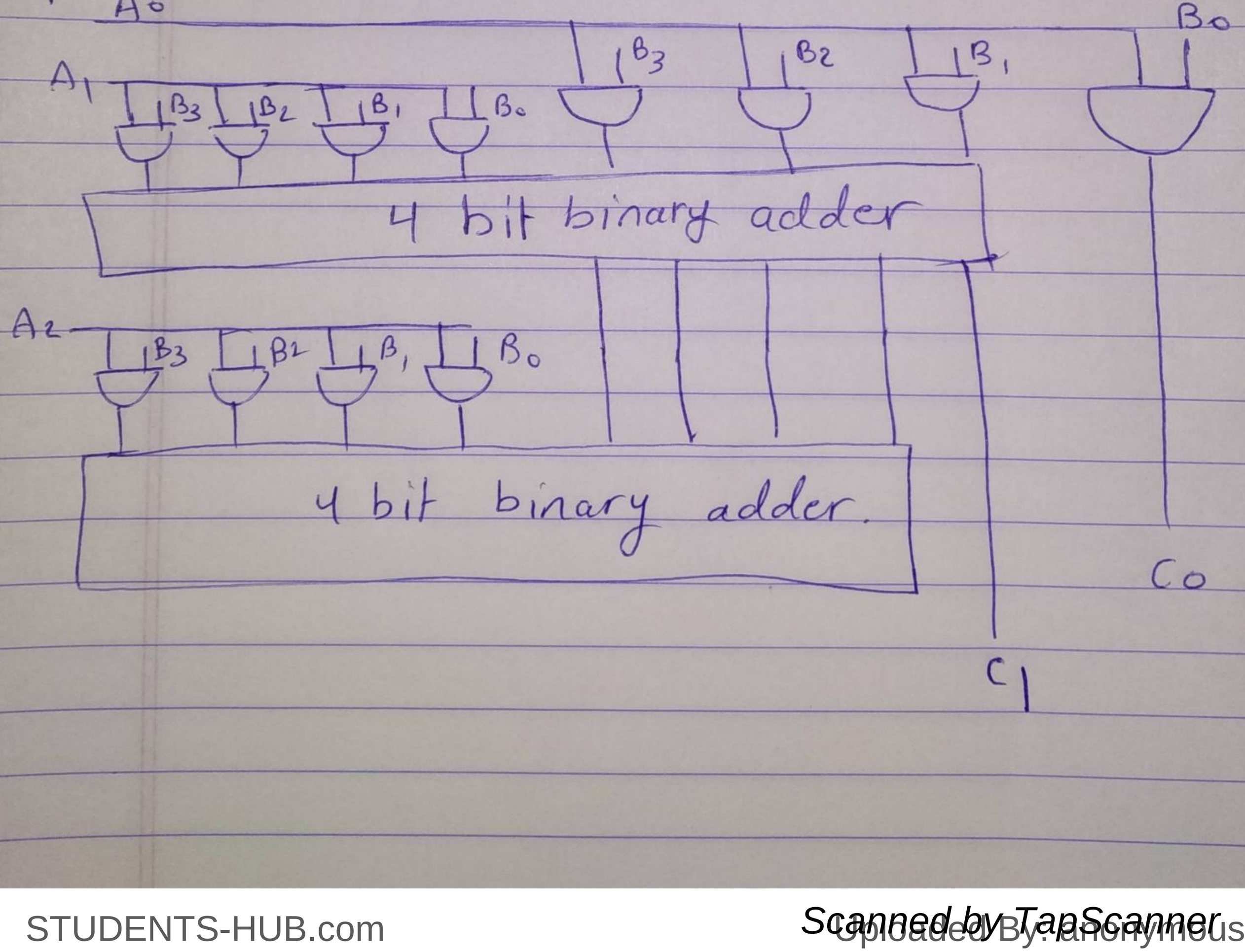
ex Design a 2 bit by 2 bit binary multiplier.



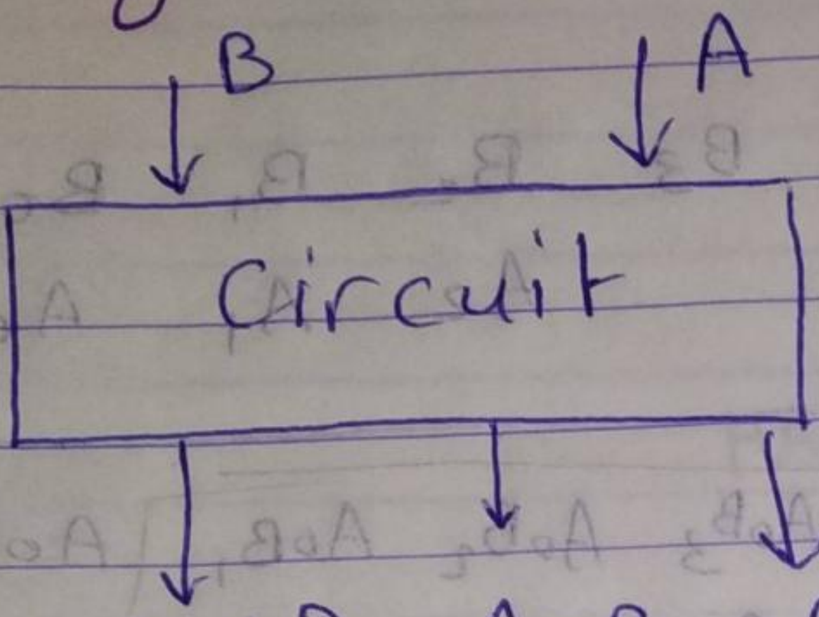


ex Design 4 bit x 3 bit multiplier.

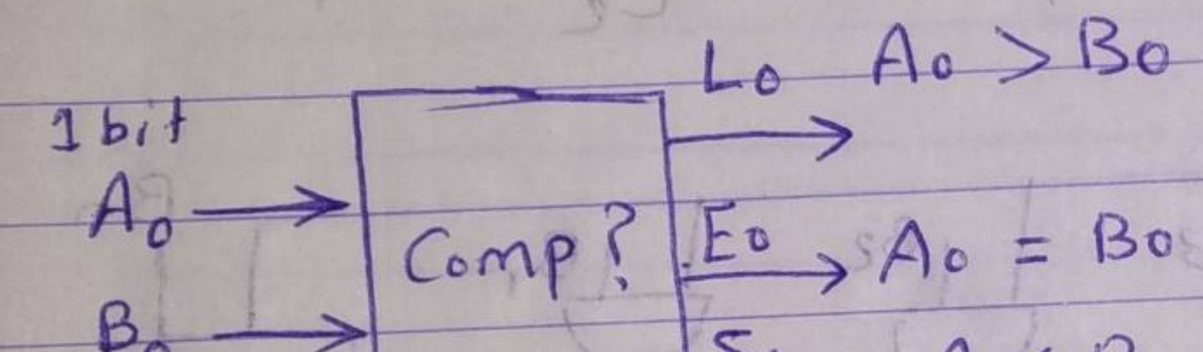




Magnitude Comparator :-



Design 1 bit magnitude Compulter.



A_0	B_0	L_0	S_0	E_0
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$L_0 = A_0 B_0'$

A_0 \ B_0	0	1
0		
1	1	

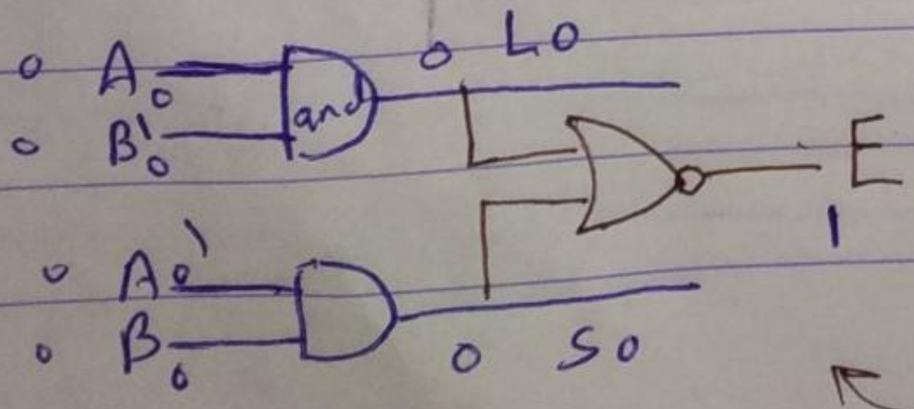
$S_0 = A_0' B_0$

$L_0 + S_0 = A_0 B_0' + A_0' B_0$

$E_0 = A_0' B_0' + A_0 B_0$

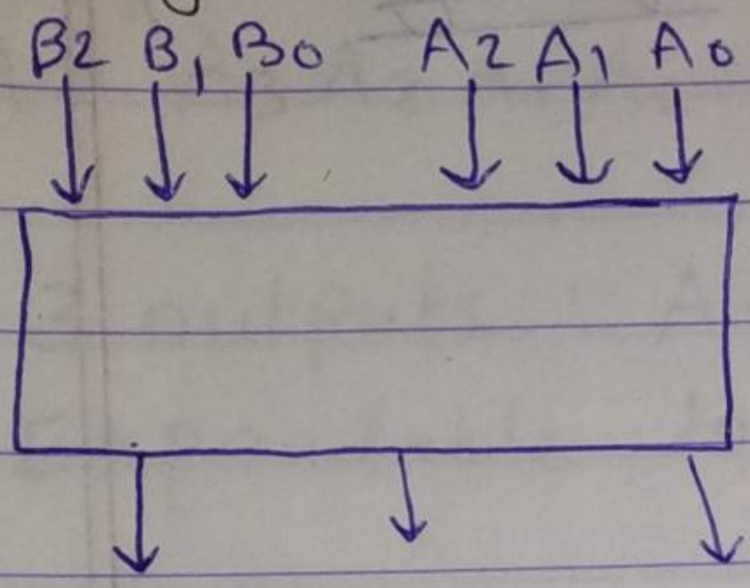
$E_0 = (L_0 + S_0)'$

$= (A_0 + B_0)'$.



← 1 bit magnitude Comparator.

Design 3 bit magnitude Comparator.

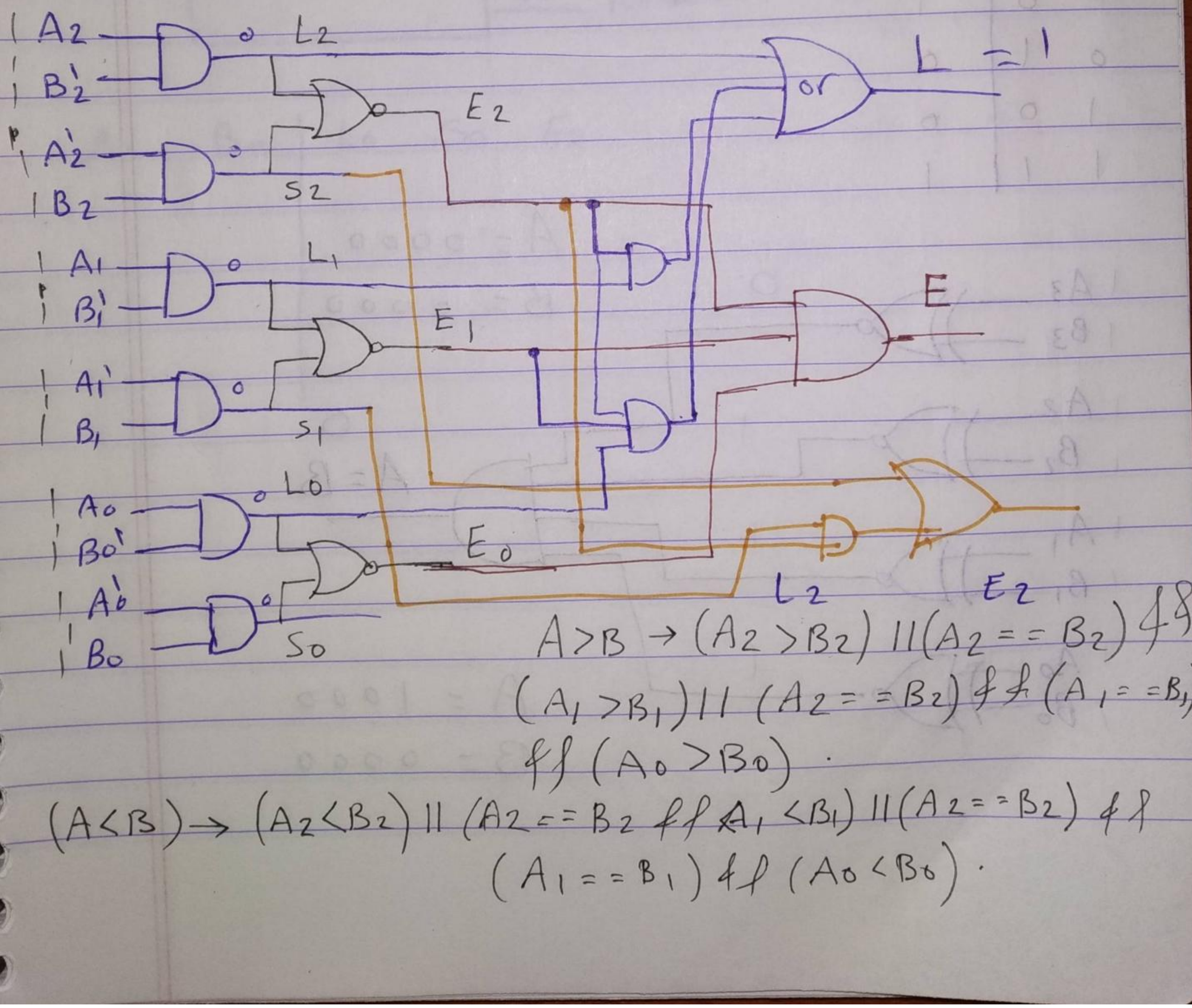


$A = A_2 A_1 A_0$.

$B = B_2 B_1 B_0$.

$L(A>B)$ $S(A<B)$ $E(A=B)$.

$A = B \rightarrow (A_2 == B_2) \,\&\, (A_1 == B_1) \,\&\, (A_0 == B_0)$.



$A > B \rightarrow (A_2 > B_2) \,||\, (A_2 == B_2) \,\&\&\, (A_1 > B_1) \,||\, (A_2 == B_2) \,\&\&\, (A_1 == B_1) \,\&\&\, (A_0 > B_0)$.

$(A < B) \rightarrow (A_2 < B_2) \,||\, (A_2 == B_2 \,\&\&\, A_1 < B_1) \,||\, (A_2 == B_2) \,\&\&\, (A_1 == B_1) \,\&\&\, (A_0 < B_0)$.

Design a 4 bit magnitude equality Comparator.

$A = A_3 A_2 A_1 A_0$

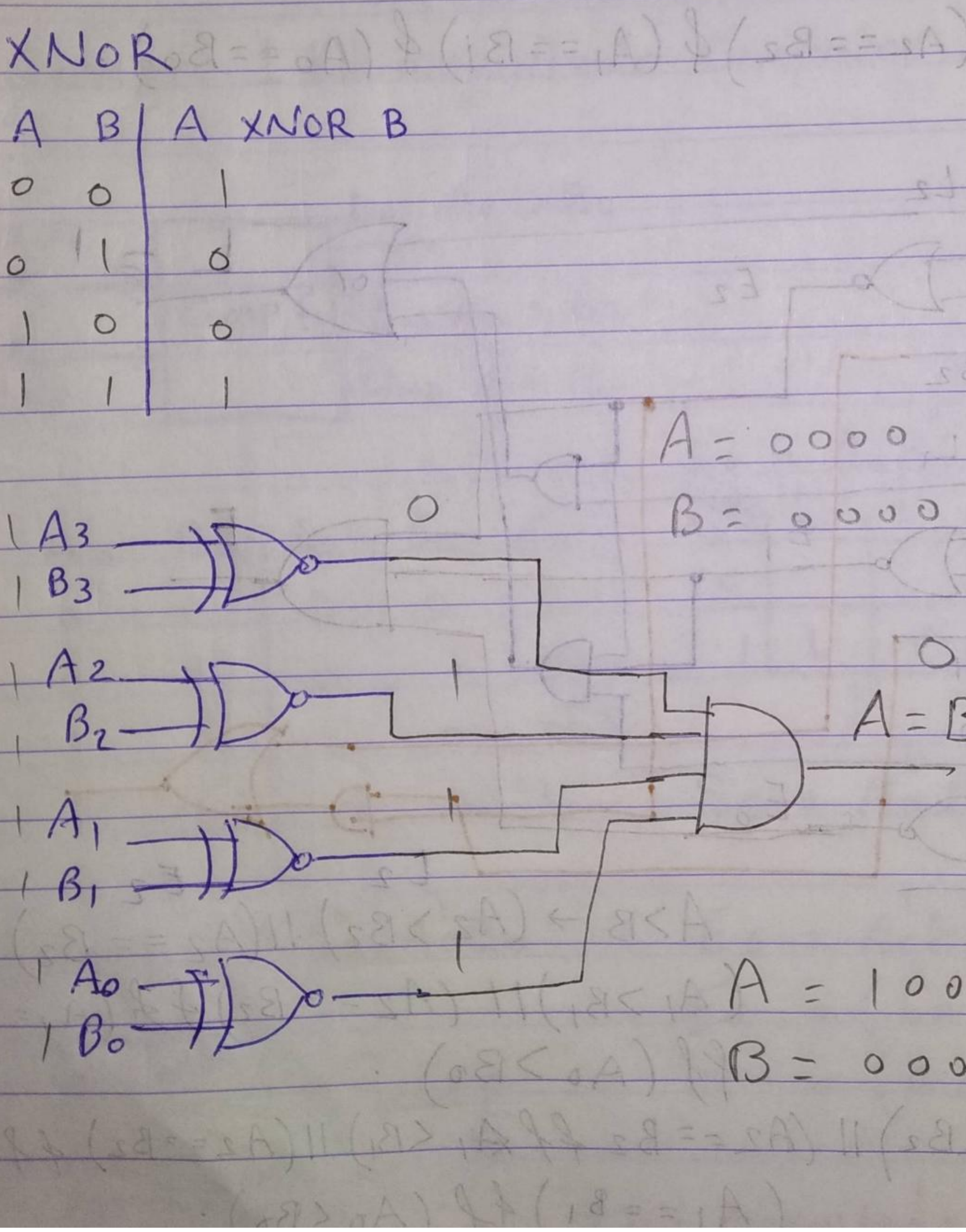
$B = B_3 B_2 B_1 B_0$

$A == B$

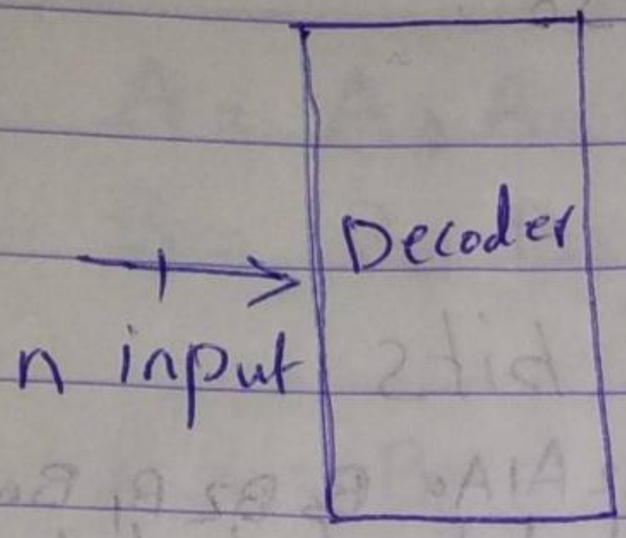
$(A_3 == B_3) \&\& (A_2 == B2) \&\& (A_1 == B_1) \&\& (A_0 == B_0)$

XNOR

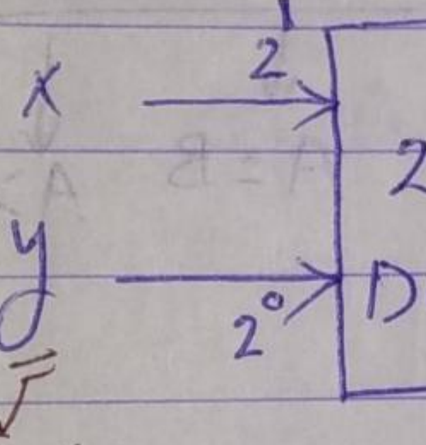
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1



Decoder : Combinational circit.



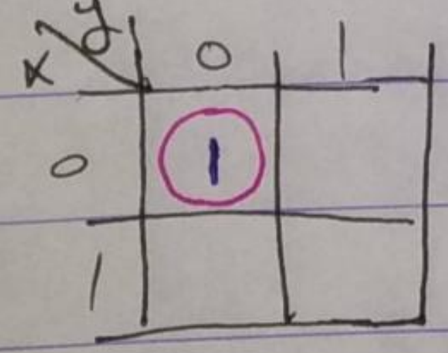
example :- Decoder 2×4.



Block digram.

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

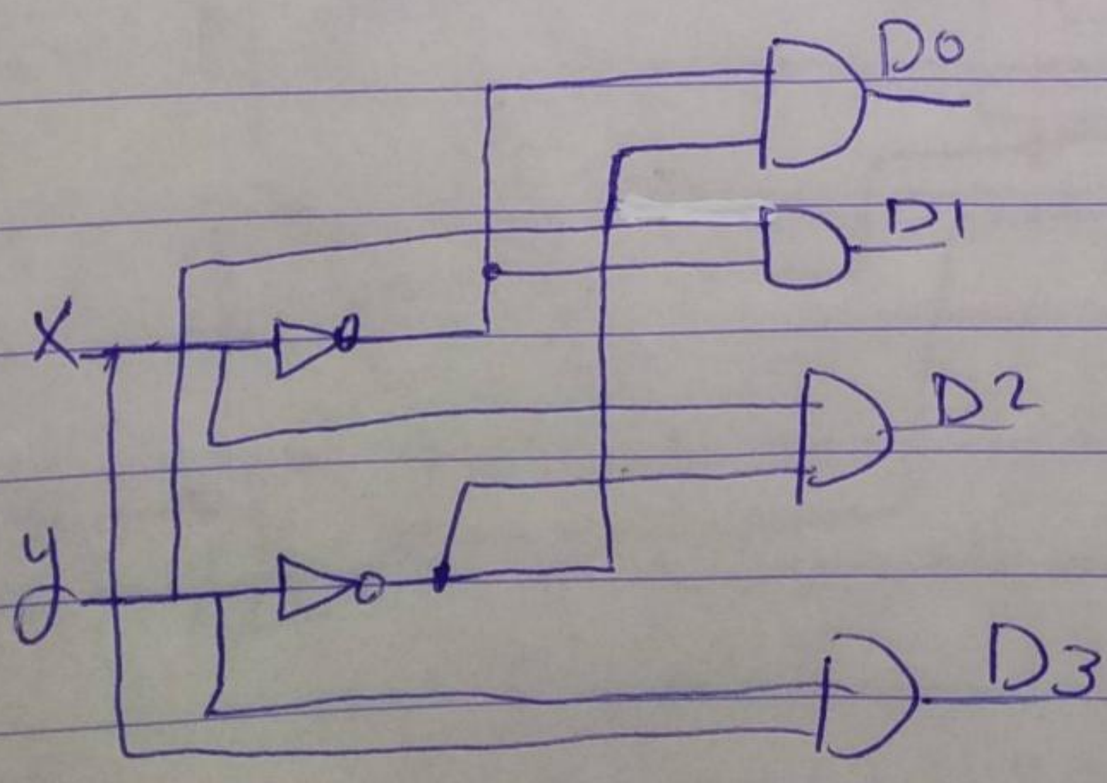
$D_0 = x'y' = m_0$.



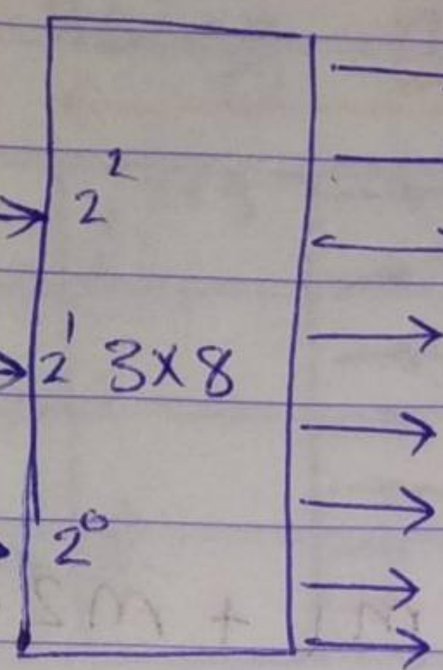
$D_1 = x'y = m_1$.

$D_2 = xy' = m_2$.

$D_3 = xy = m_3$.



Example 3 X 8 Line Decoder

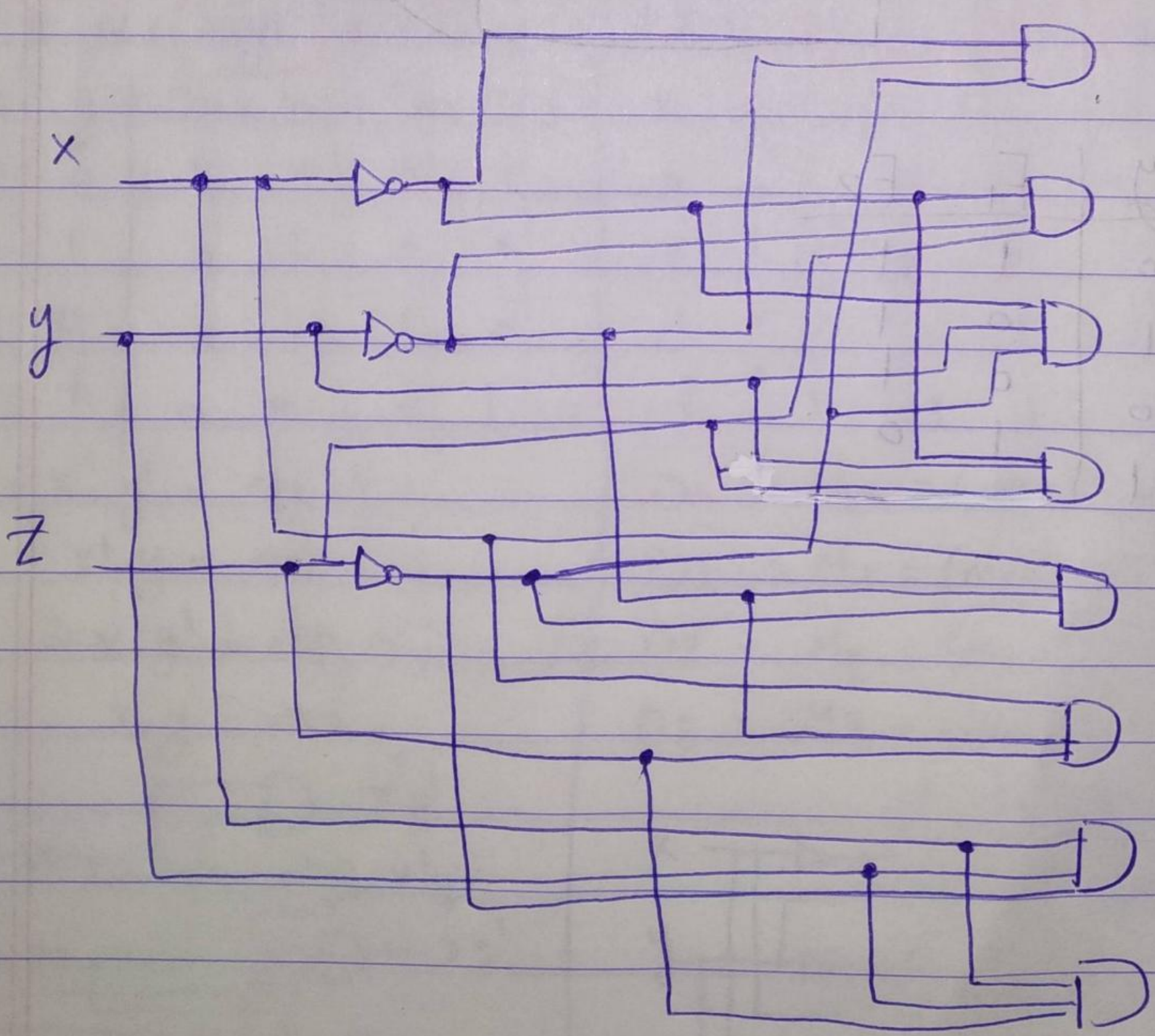


X	y	Z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = x'y'z'$ $D_1 = x'y'z$ $D_2 = x'yz'$

$D_3 = x'yz$ $D_4 = xy'z'$ $D_5 = xy'z$

$D_6 = xyz'$ $D_7 = xyz$.

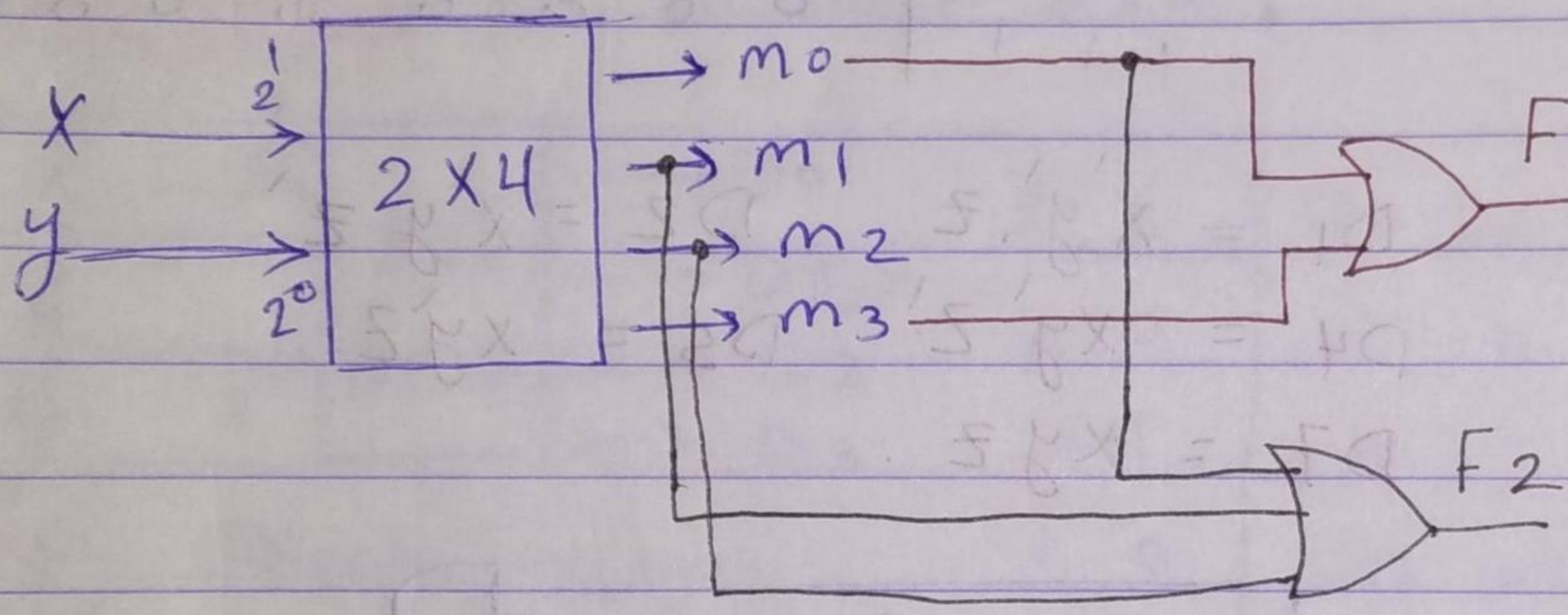


Example 1- Implement the following Function using Decoder ?.

$F_1(x, y) = \Sigma 0, 3$

$F_2(x, y) = \Sigma 0, 1, 2$

$F_1 = m0 + m3$. $F_2 = m0 + m1 + m2$.

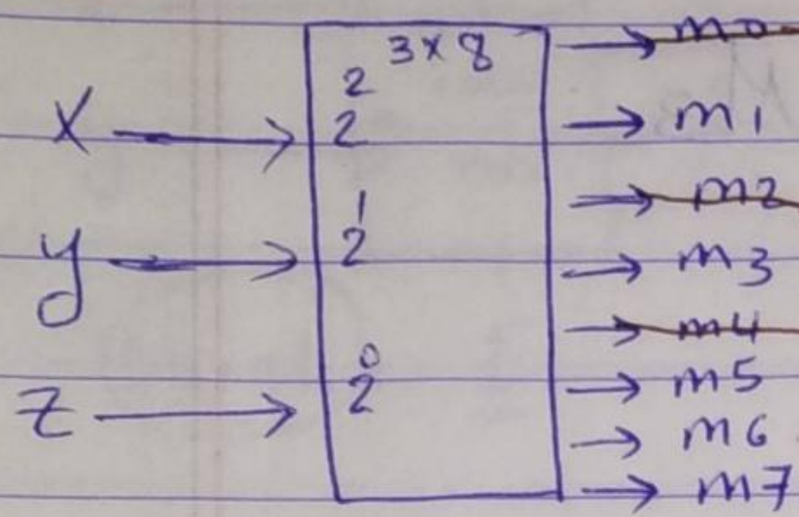


X	y	F_1	F_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	0

example :- Implement $F(x,y,z) = \sum 0,2,4$

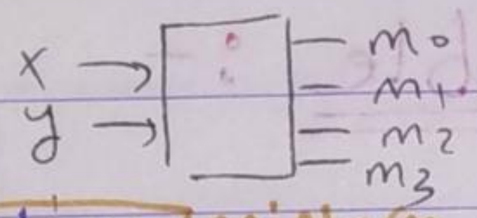
using active high decoder.

minterm ←



x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
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

example :- 2 x 4 active high decoder



minterm

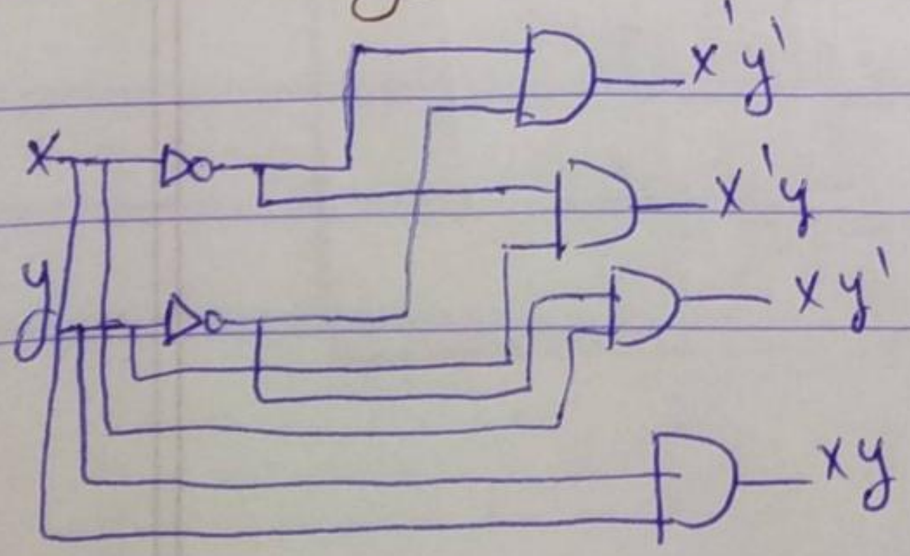
x	y	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$D_0 = x'.y' = m_0$

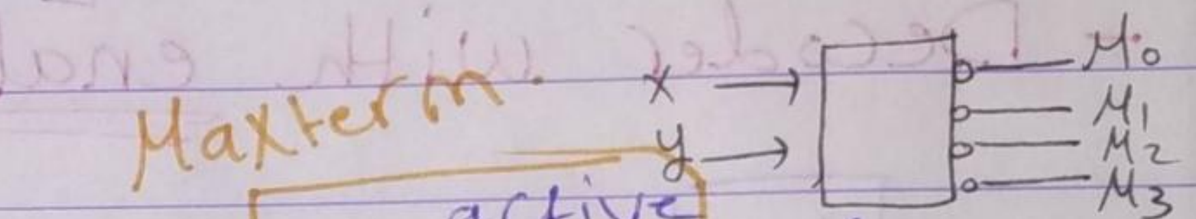
$D_1 = x'.y = m_1$

$D_2 = x.y' = m_2$

$D_3 = x.y = m_3$



Maxterm



2 x 4 active low decoder.

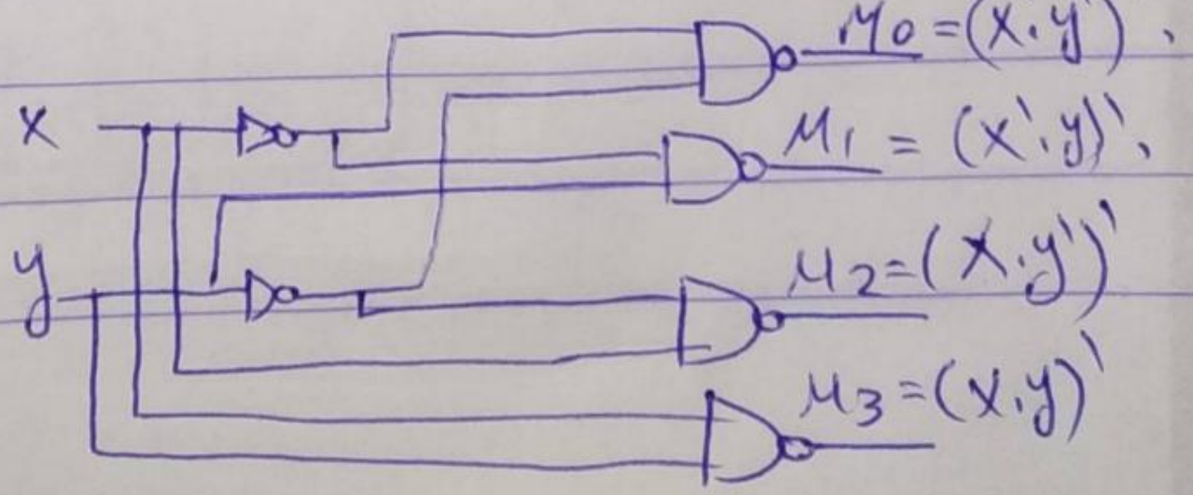
x	y	D0	D1	D2	D3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

$D_0 = M_0 = (m_0)' = (x'.y')' = x+y$

$D_1 = M_1 = (m_1)' = (x'.y)'$

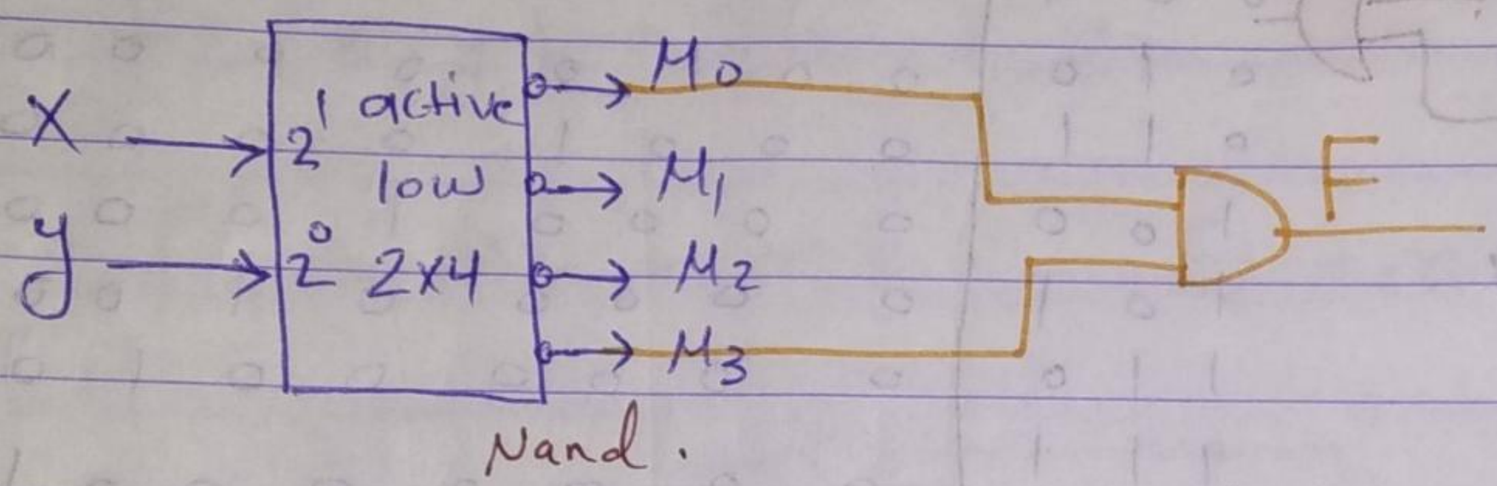
$D_2 = M_2 = (m_2)' = (x.y')'$

$D_3 = M_3 = (m_3)' = (x.y)'$

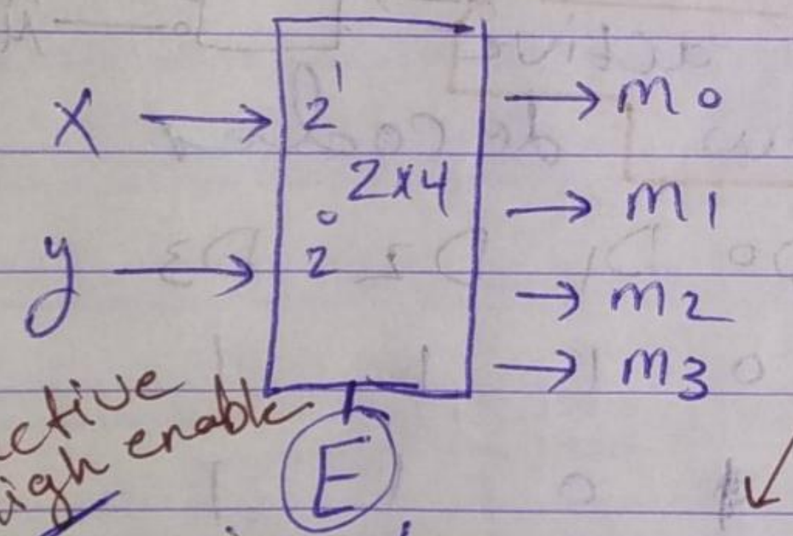


example :- Implement the following Function using decoder.

$$F(x,y) = \pi(0,3) = M_0 \cdot M_3$$



- Decoder with enable (E) :-

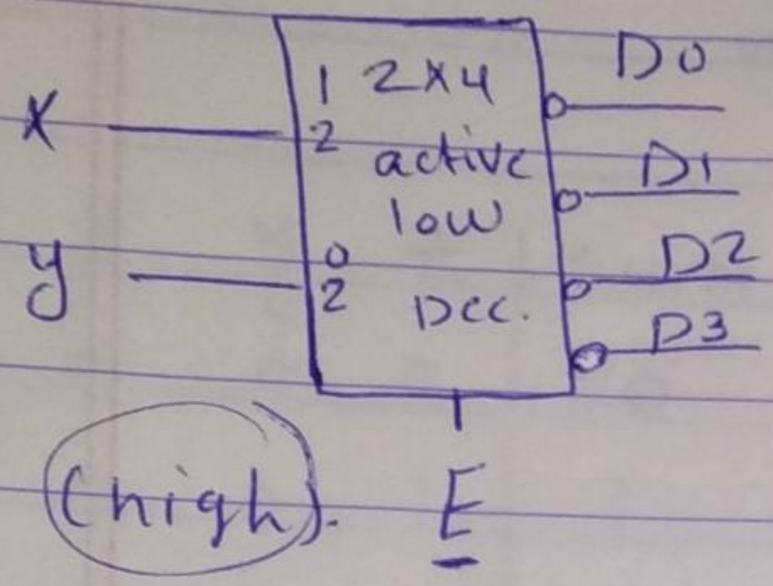


E	X	y	D0 m0	D1 m1	D2 m2	D3 m3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

active high enable.. active high decoder.

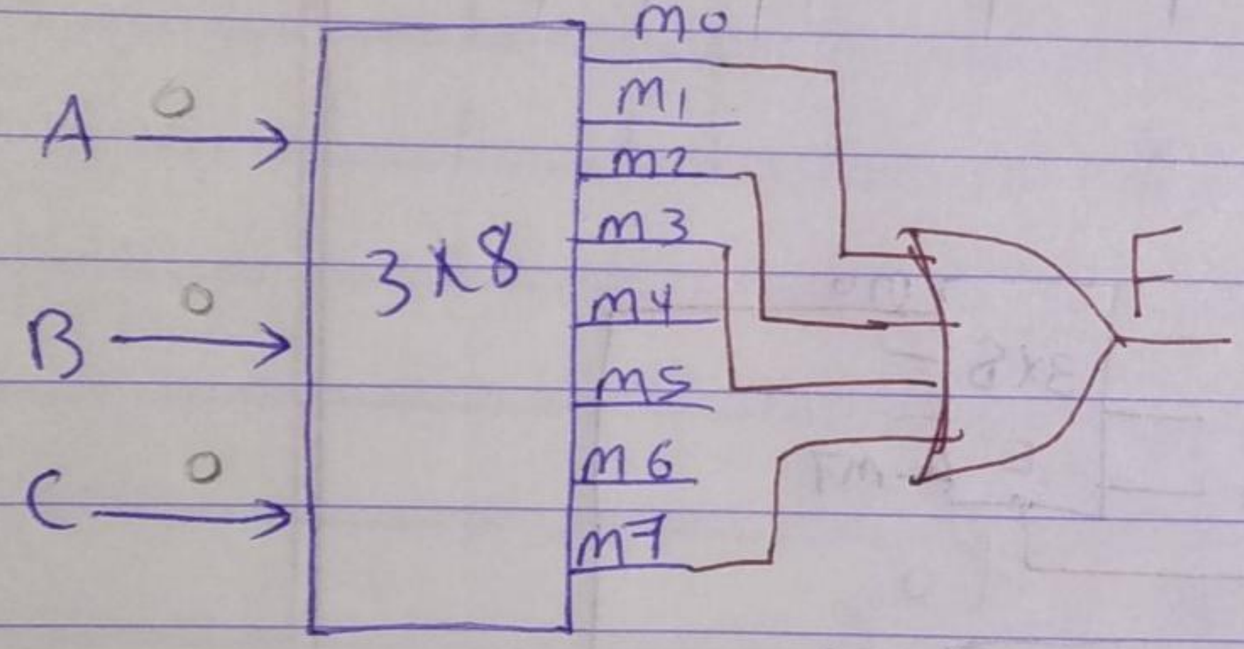
اذا كان صفر بغض النظر عن قيمة x,y فتخرج أصفار ولم يكون 1 بنشوف كيف كنا نعمل.

example :- 2 X 4 decoder.

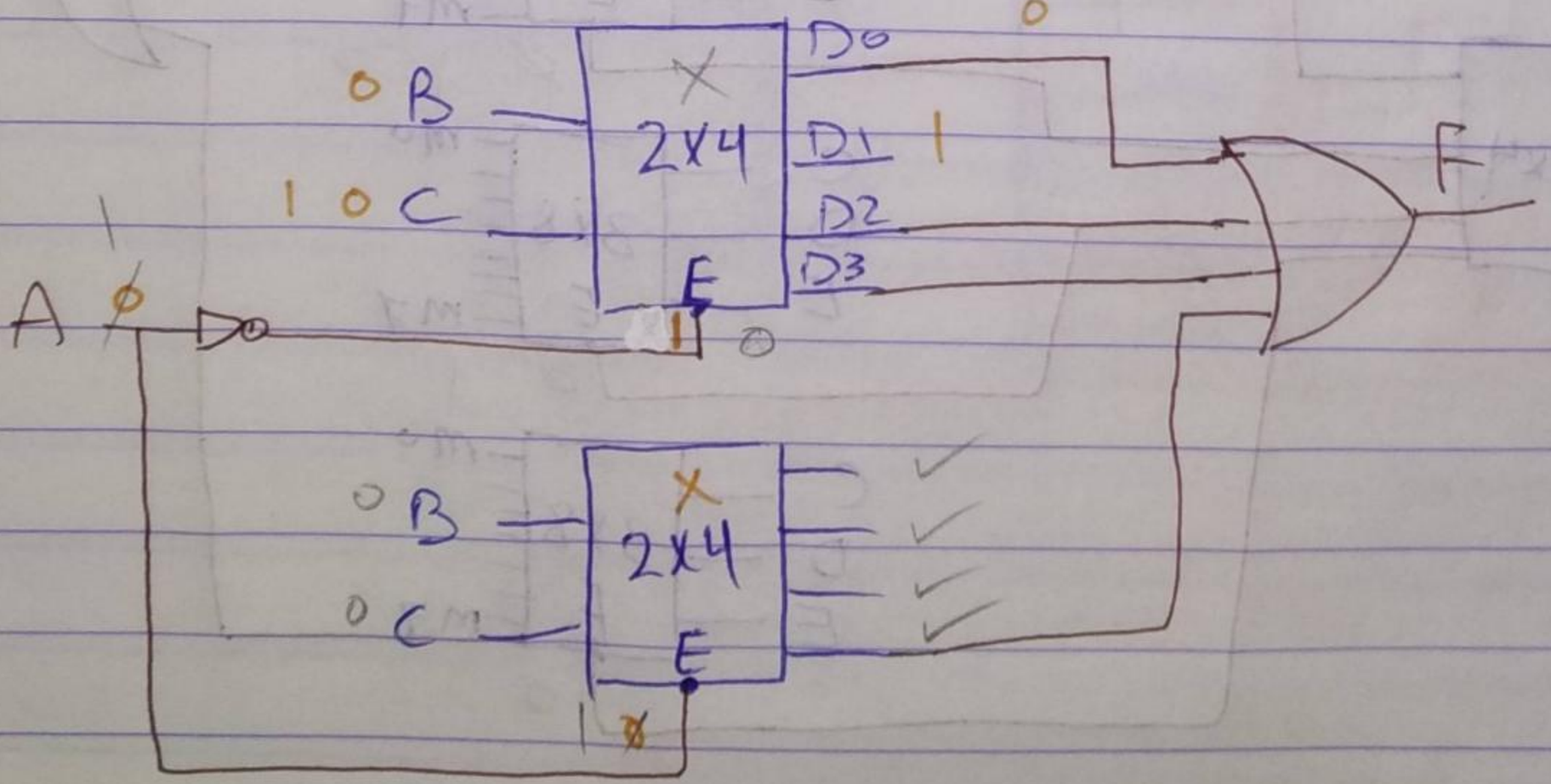


E	x	y	M_0 D0	M_1 D1	M_2 D2	M_3 D3
0	x	x	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

example :- $F(A,B,C) = \Sigma(0,2,3,7)$ Decoder.

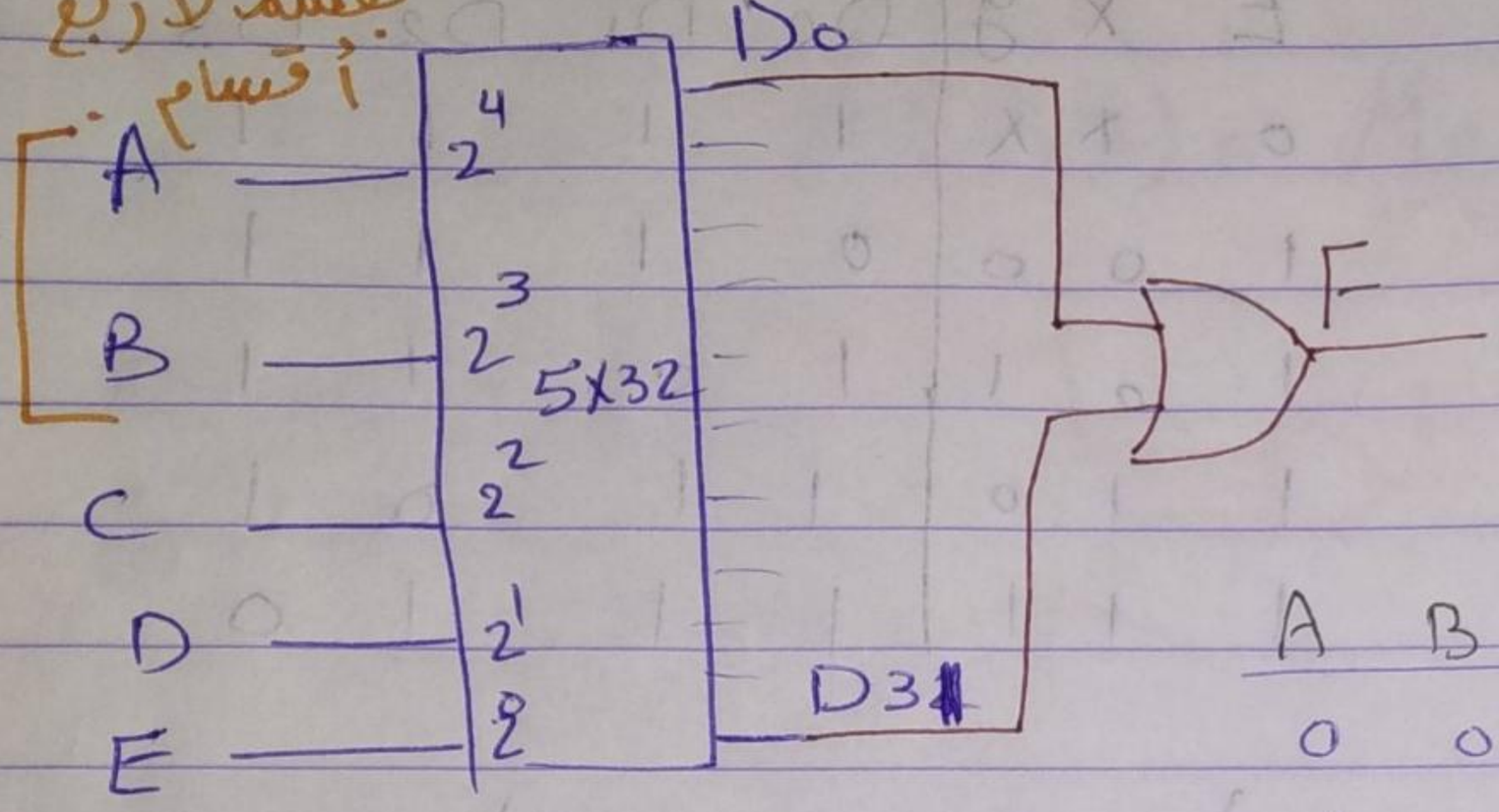


A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

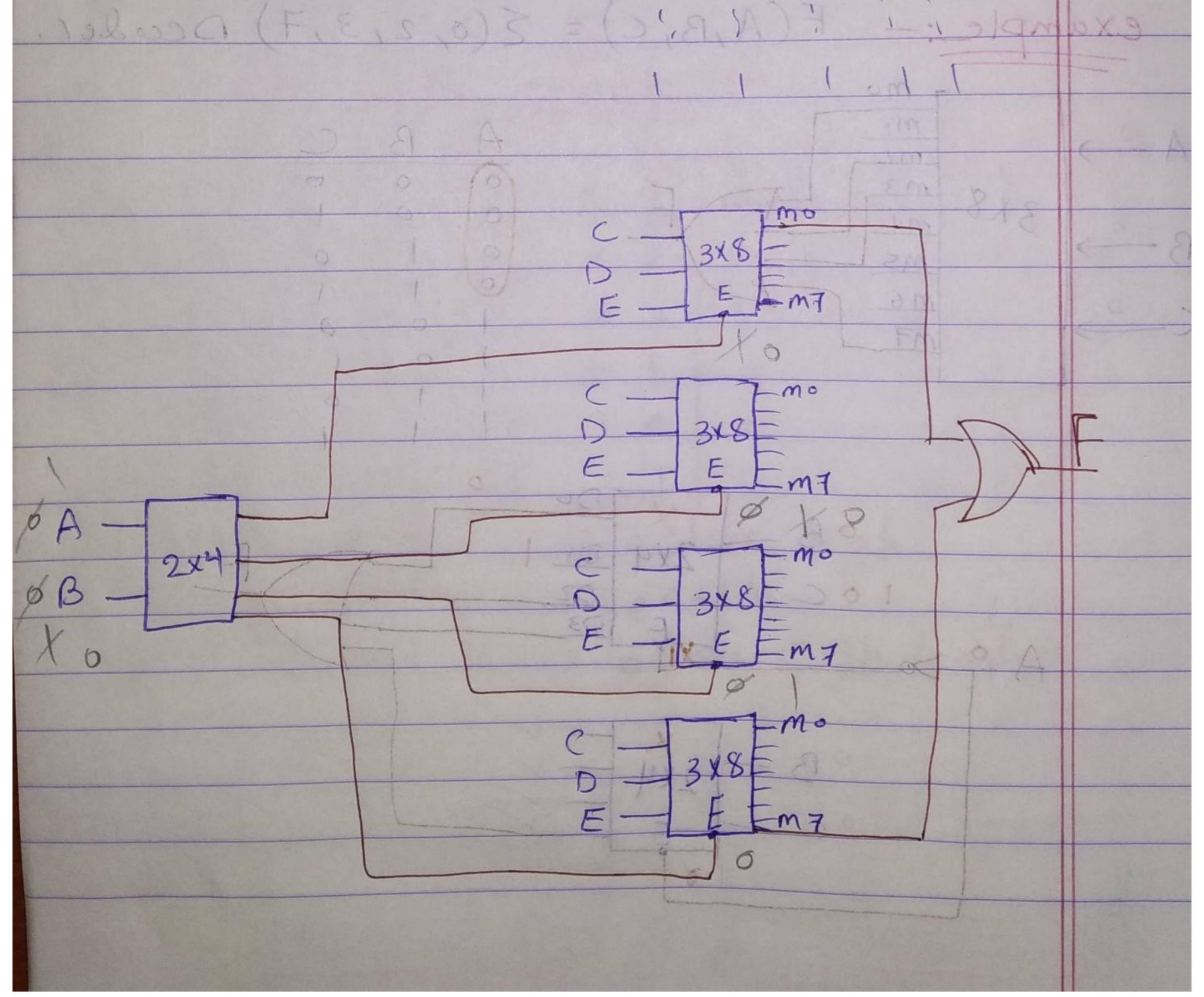


example :- $F(A,B,C,D,E) = \sum 0, 31$

بقسم لأربع أقسام



A	B	C	D	E
0	0	0	0	0
1	1	1	1	1

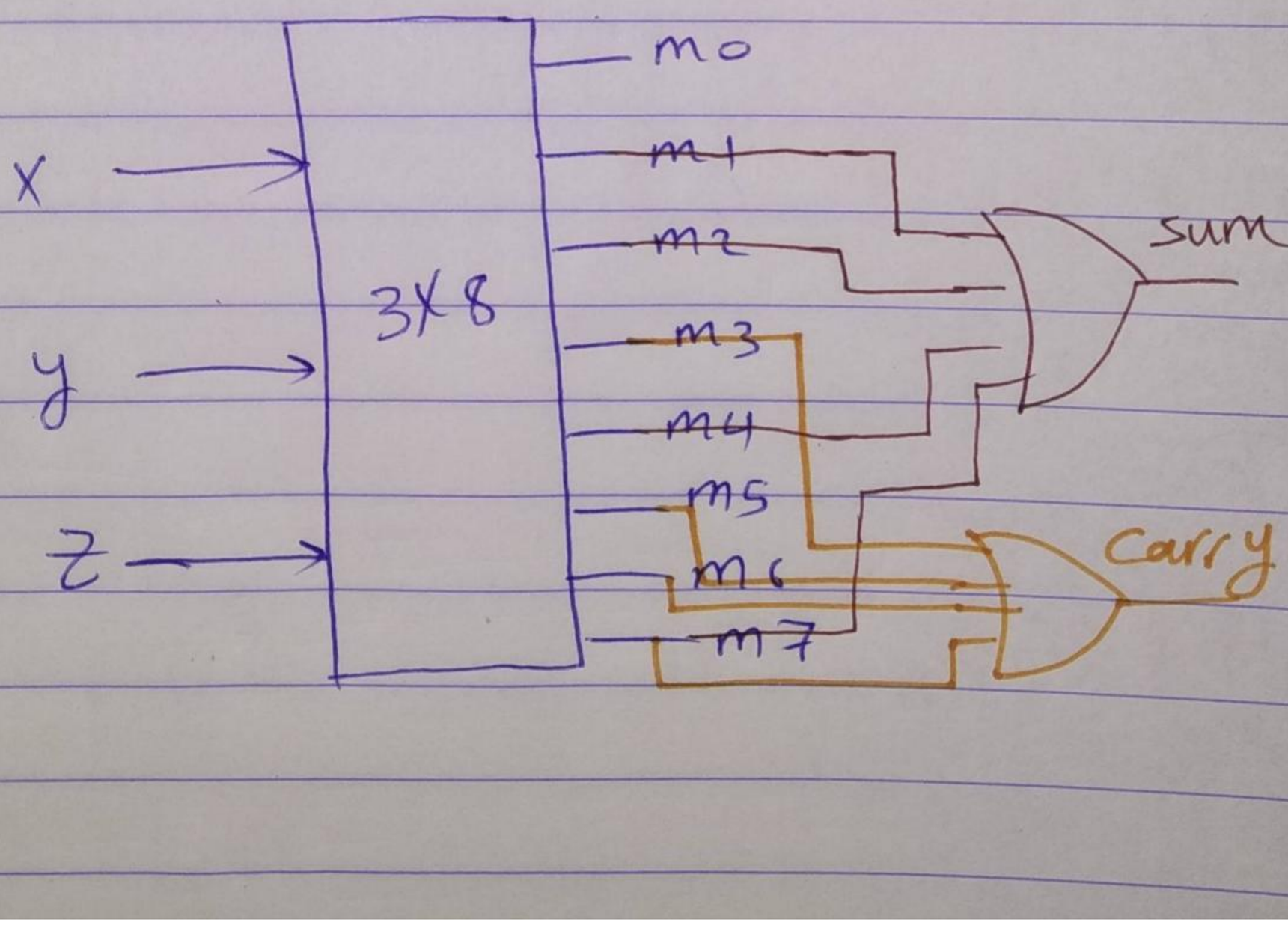


example :- Implement full adder using decoder.. (بوخذ الواحدات)

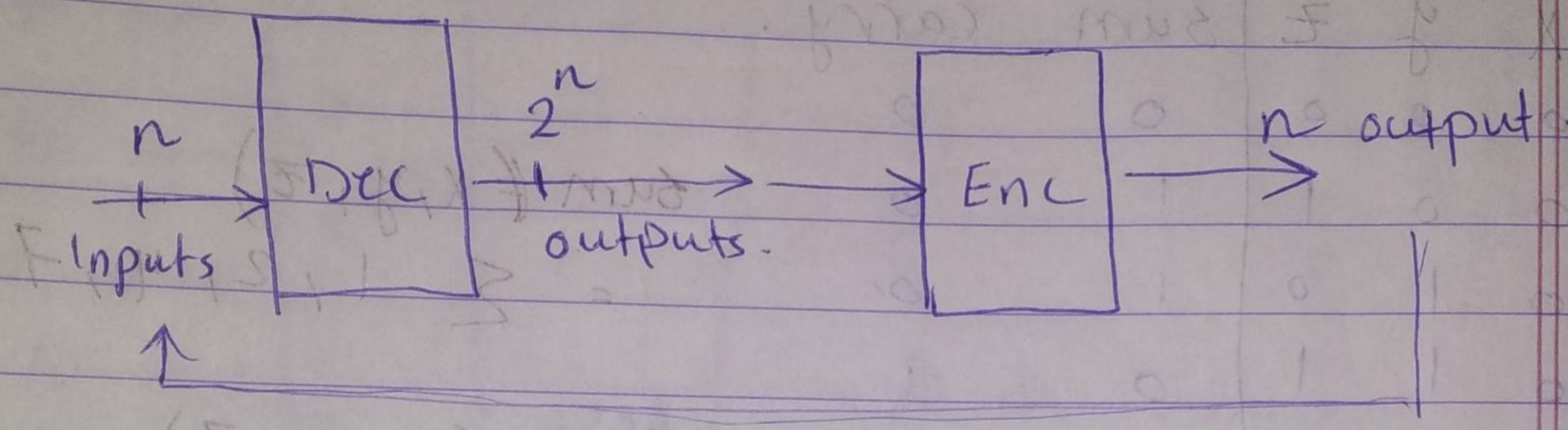
x	y	z	sum	carry.
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

sum (x, y, z)
$= \sum 1, 2, 4, 7.$

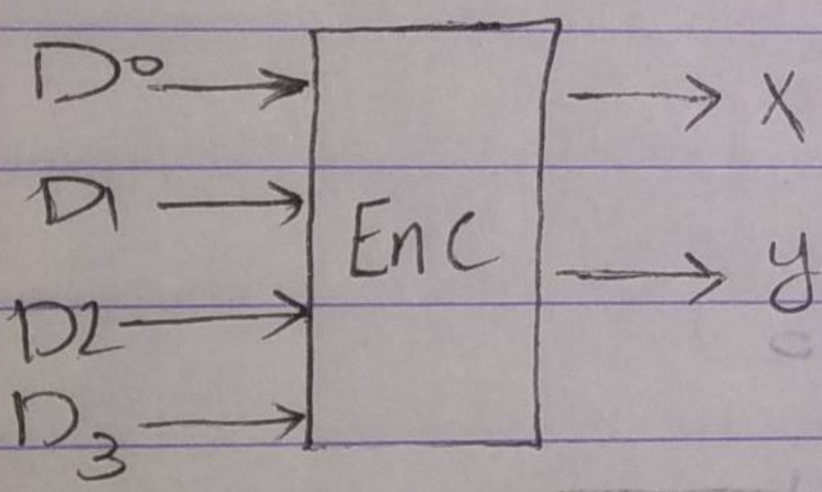
carry (x, y, z)
$= \sum 3, 5, 6, 7.$



Encoder :- Combinational circuit Inverse operation of decoder.



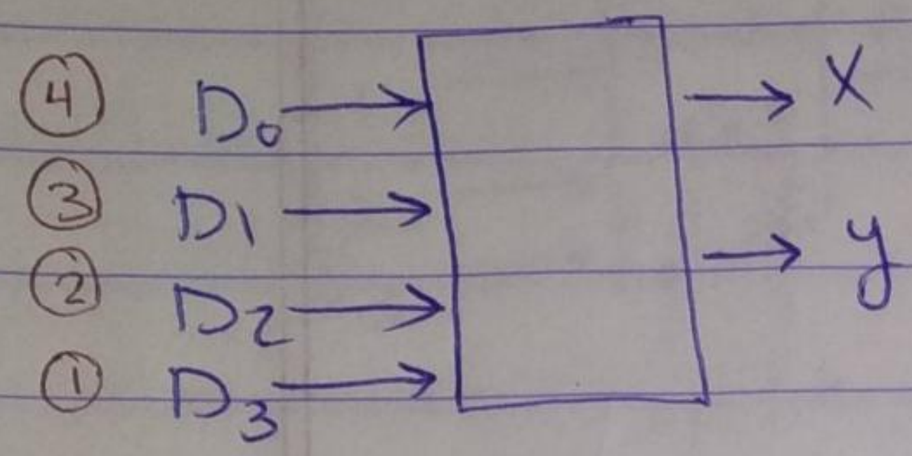
Example: 4 x 2 Encoder.



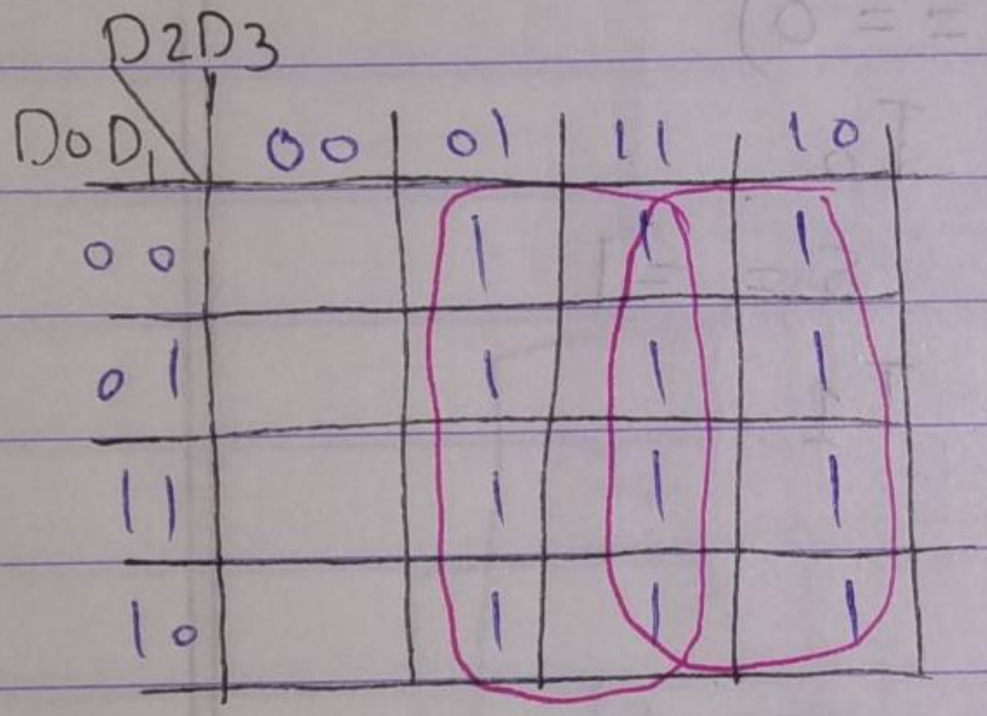
D_0	D_1	D_2	D_3	x	y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Priority Encoder :-

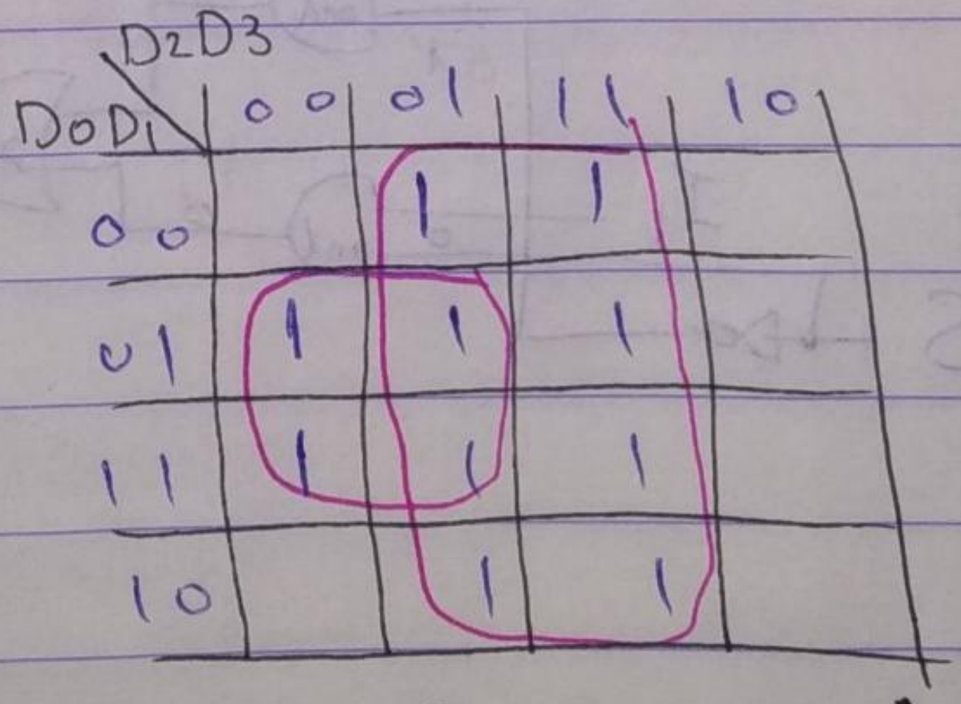
4 X 2 priority Encoder.



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



$X = D_2 + D_3$

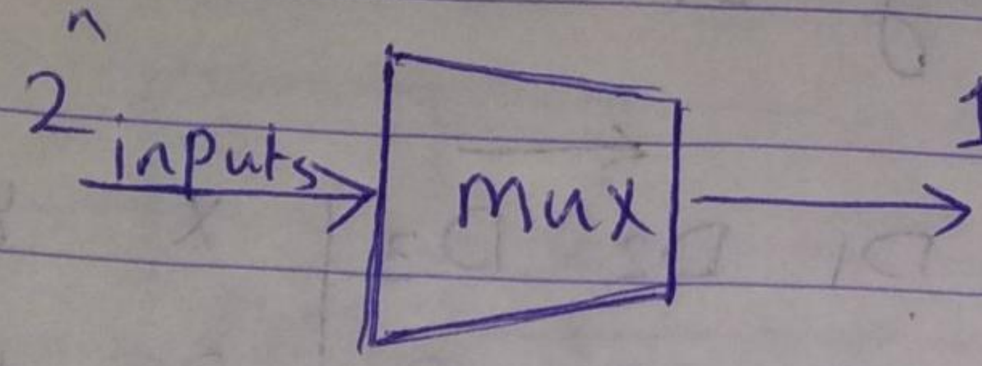


$y = D_3 + D_1D_2'$

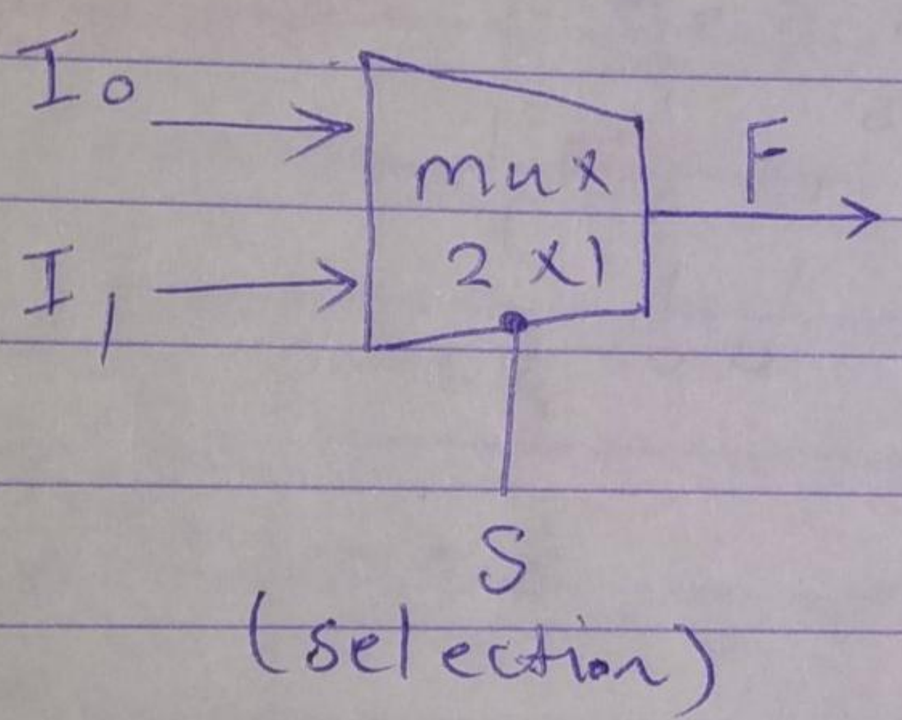
(كتبت ال inputs بدلالة ال outputs) .

اختيار من متعدد

Multiplexer (Mux) :- Combinational circit.



example :- Mux 2 x 1

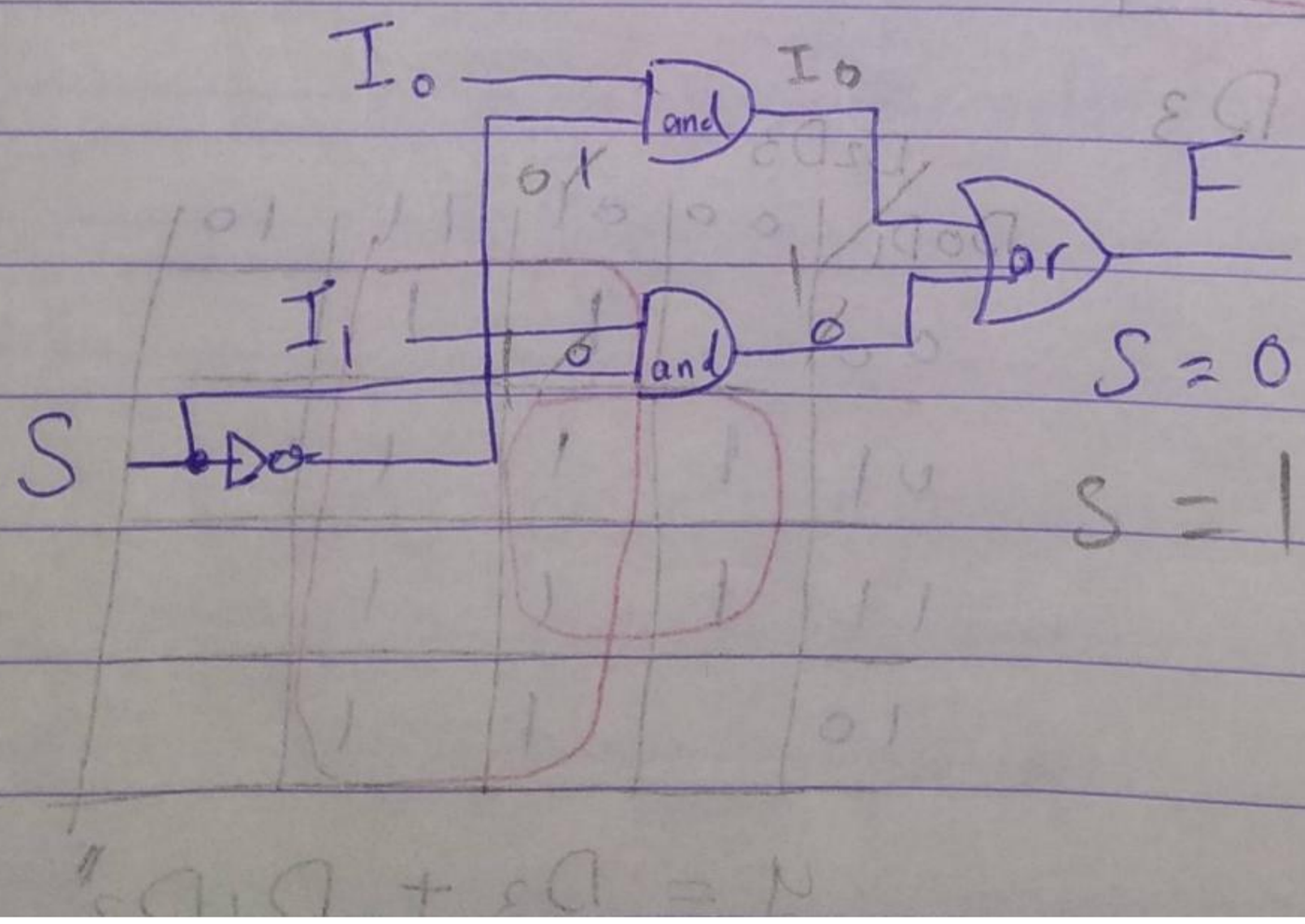


if (S == 0)

F = I0

else S == 1

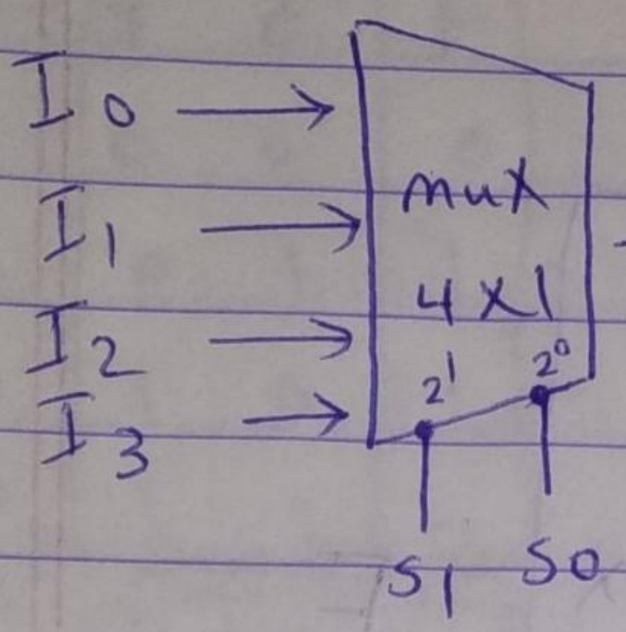
F = I1



S = 0 F = I0

S = 1 F = I1

example:- Mux 4 x 1

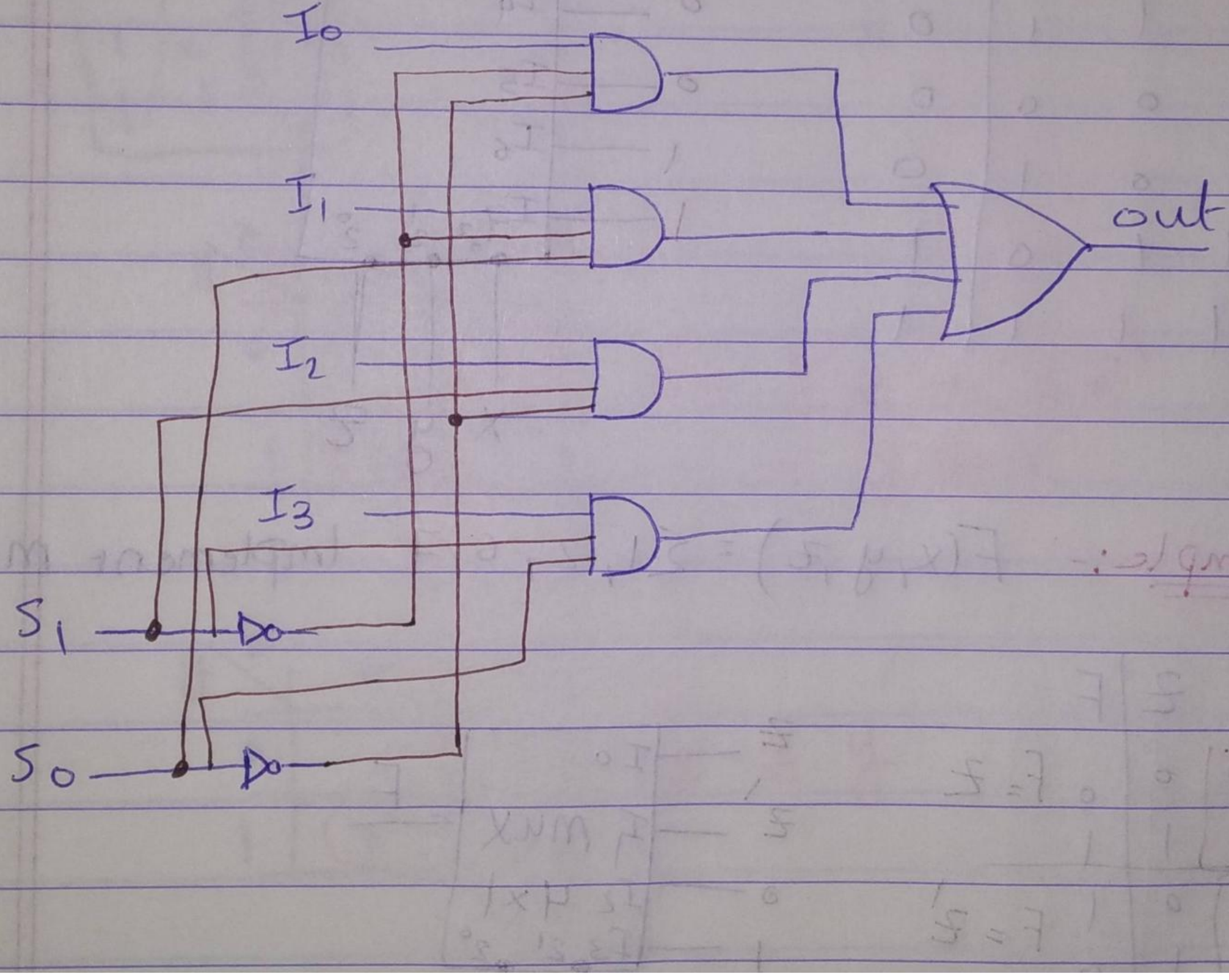


out = I_0 $S_1S_0 = 00$

out = I_1 $S_1S_0 = 01$

out = I_2 $S_1S_0 = 10$

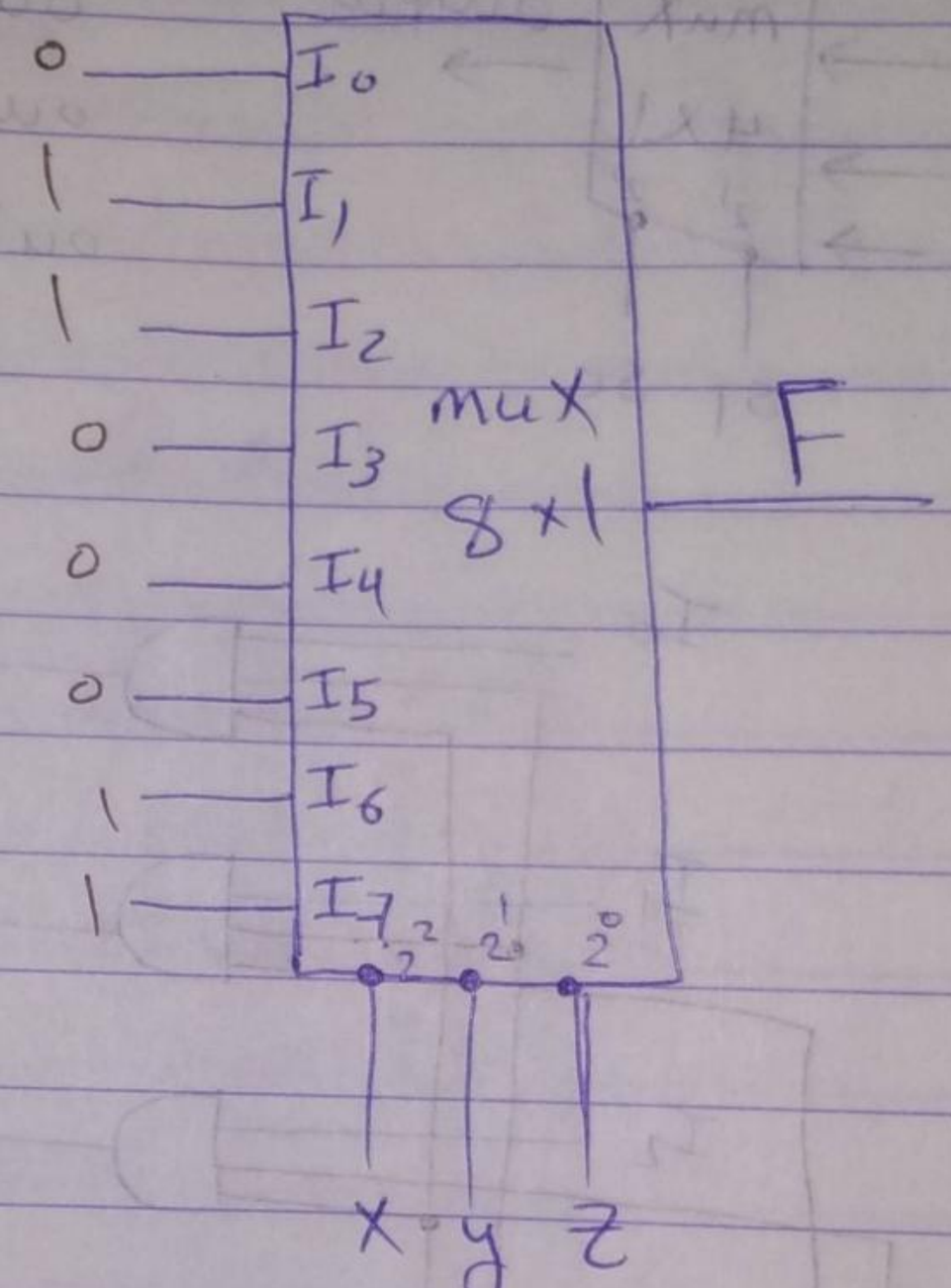
out = I_3 $S_1S_0 = 11$



example:- $F(x,y,z) = \sum 1,2,6,7$

Implement the Function using mux 8 x 1

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:- $F(x,y,z) = \sum 1,2,6,7$ Implement mux 4 x 1

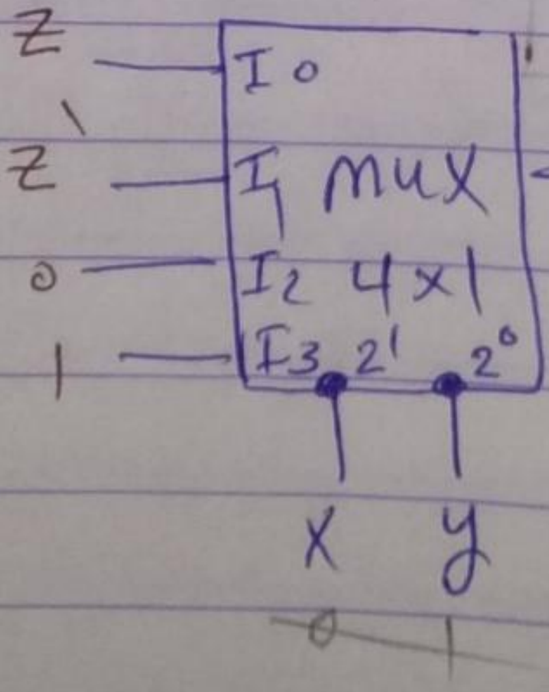
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$F = z$

$F = z'$

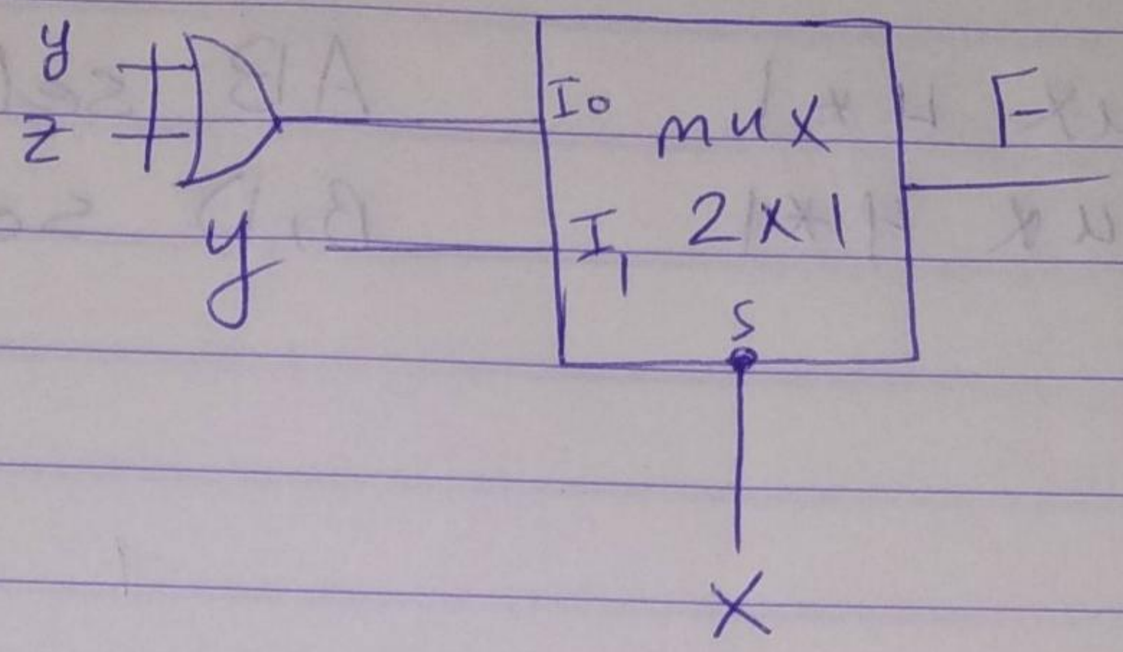
$F = 0$

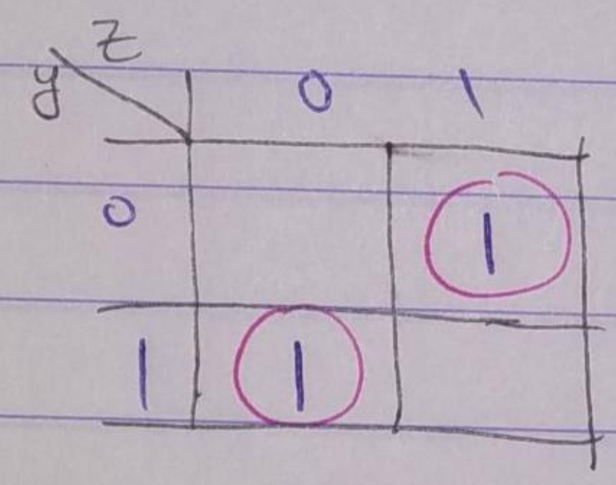
$F = 1$



example $f(x,y,z) = \sum 1,2,6,7$ Implement mux 2x1

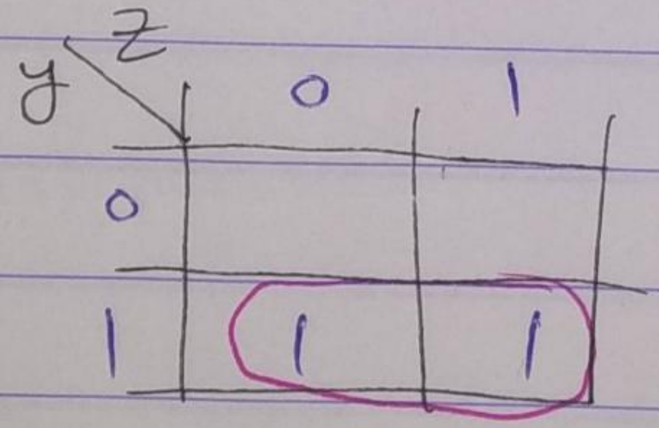
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1





$F = y'z + yz'$

$= y \oplus z$



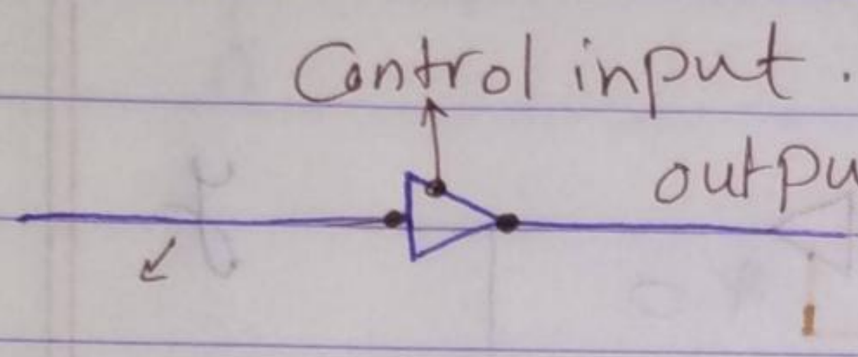
$F = y$

Three (Tri) State buffer

x → y Normal buffer.

$y = x$ (delay).

x	y
0	0
1	1



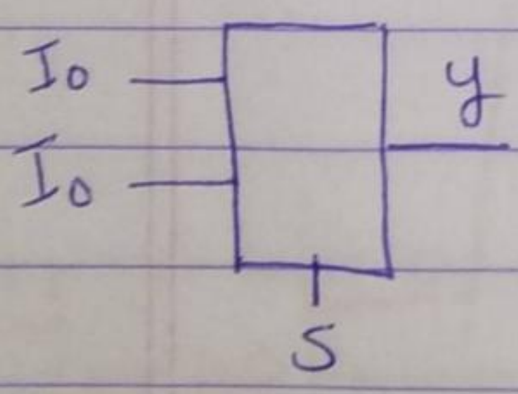
Normal input.

if control input (c) = 1

$y = \alpha$.

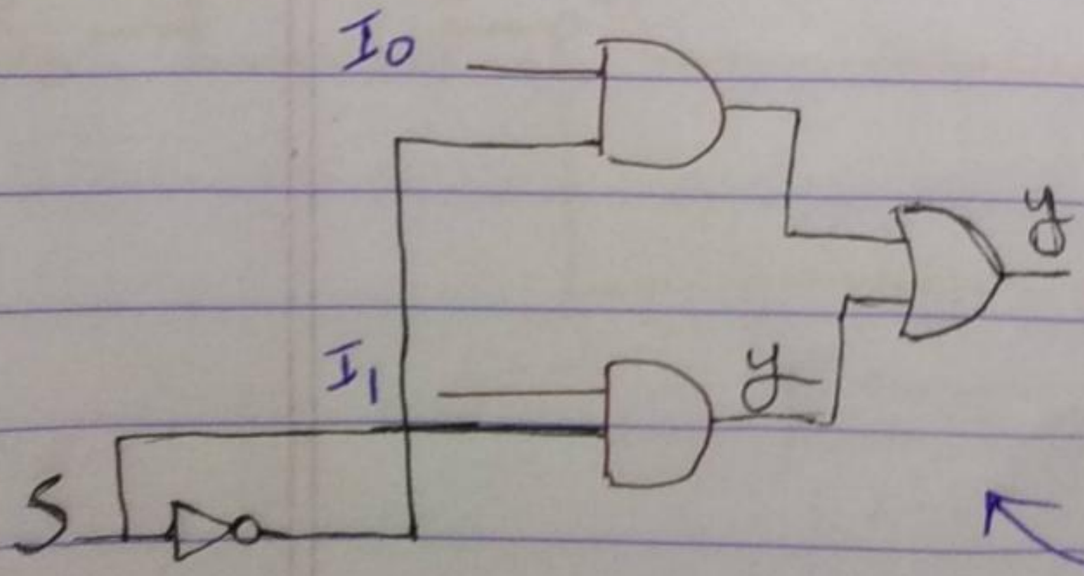
if c = = 0 (open circuit high imp

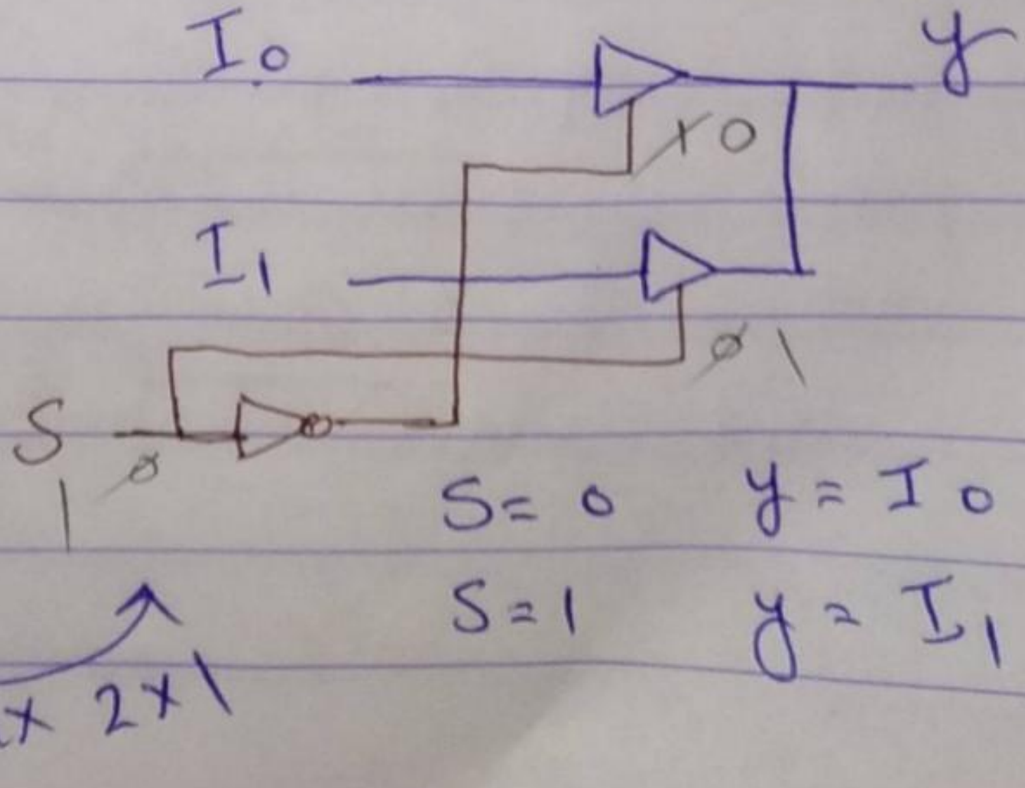
ex 2x1 mux.



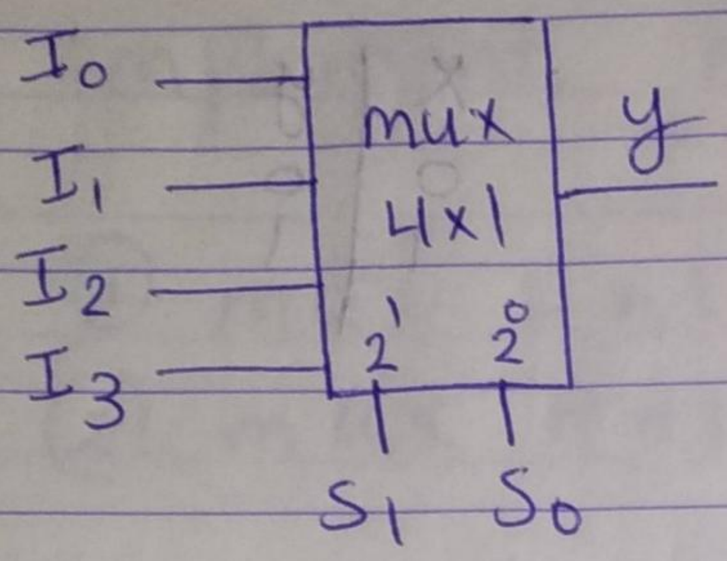
$y = I_0$, $S = 0$

$y = I_1$, $S = 1$





ex mux 4x1

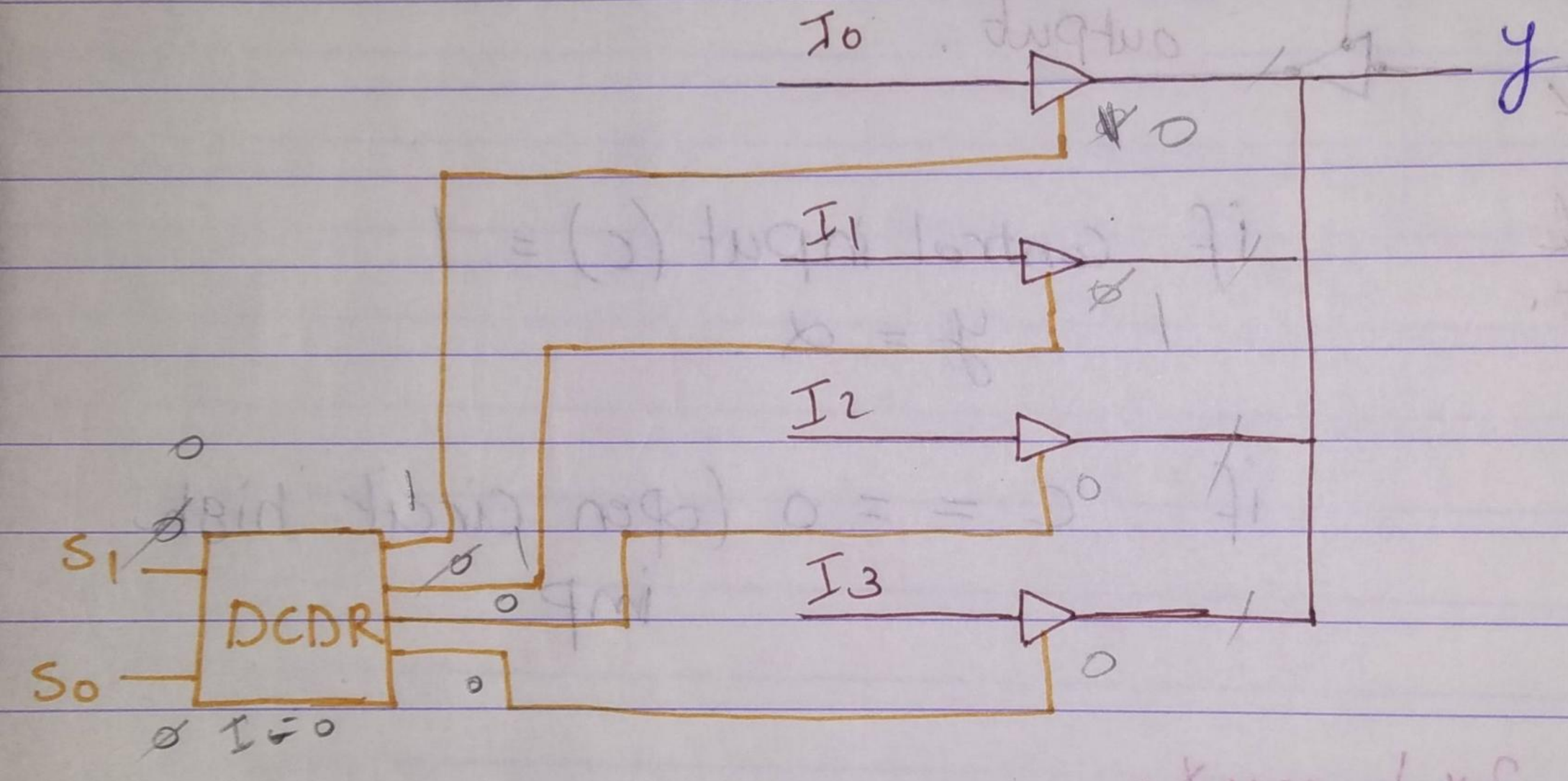


$y = I_0$, $S_1 S_0 = 0\,0$

$y = I_1$, $S_1 S_0 = 0\,1$

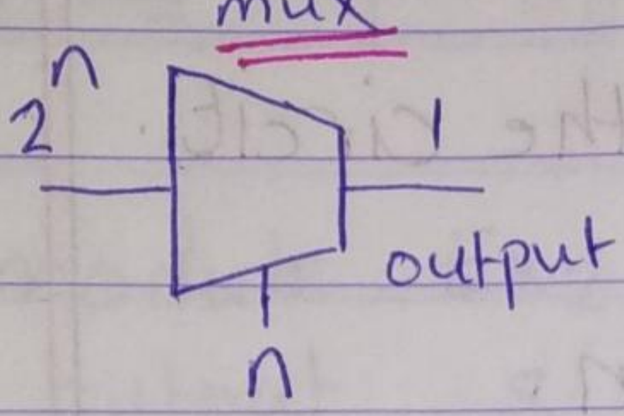
$y = I_2$, $S_1 S_0 = 1\,0$

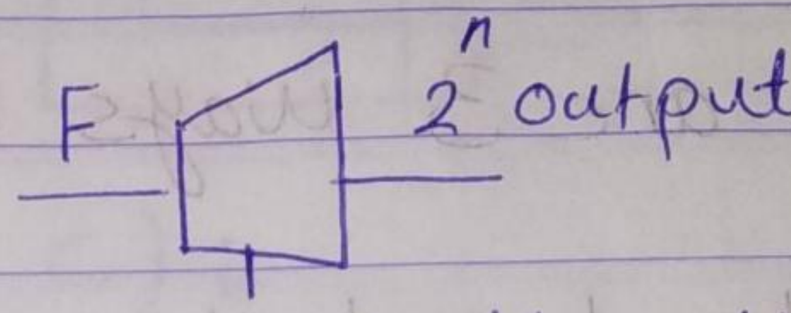
$y = I_3$, $S_1 S_0 = 1\,1$



Demultiplexer (DeMux) :-

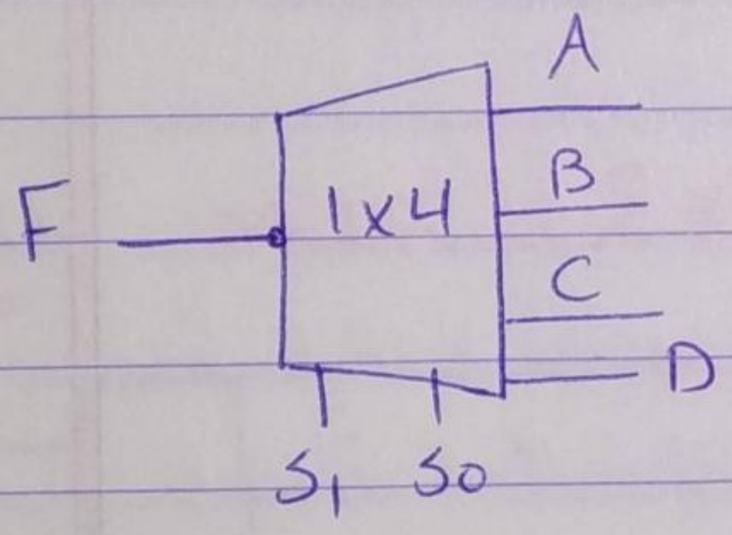
⇒ Inverse operation of multiplexer.





DeMux.

ex 1 x 4 DeMux.

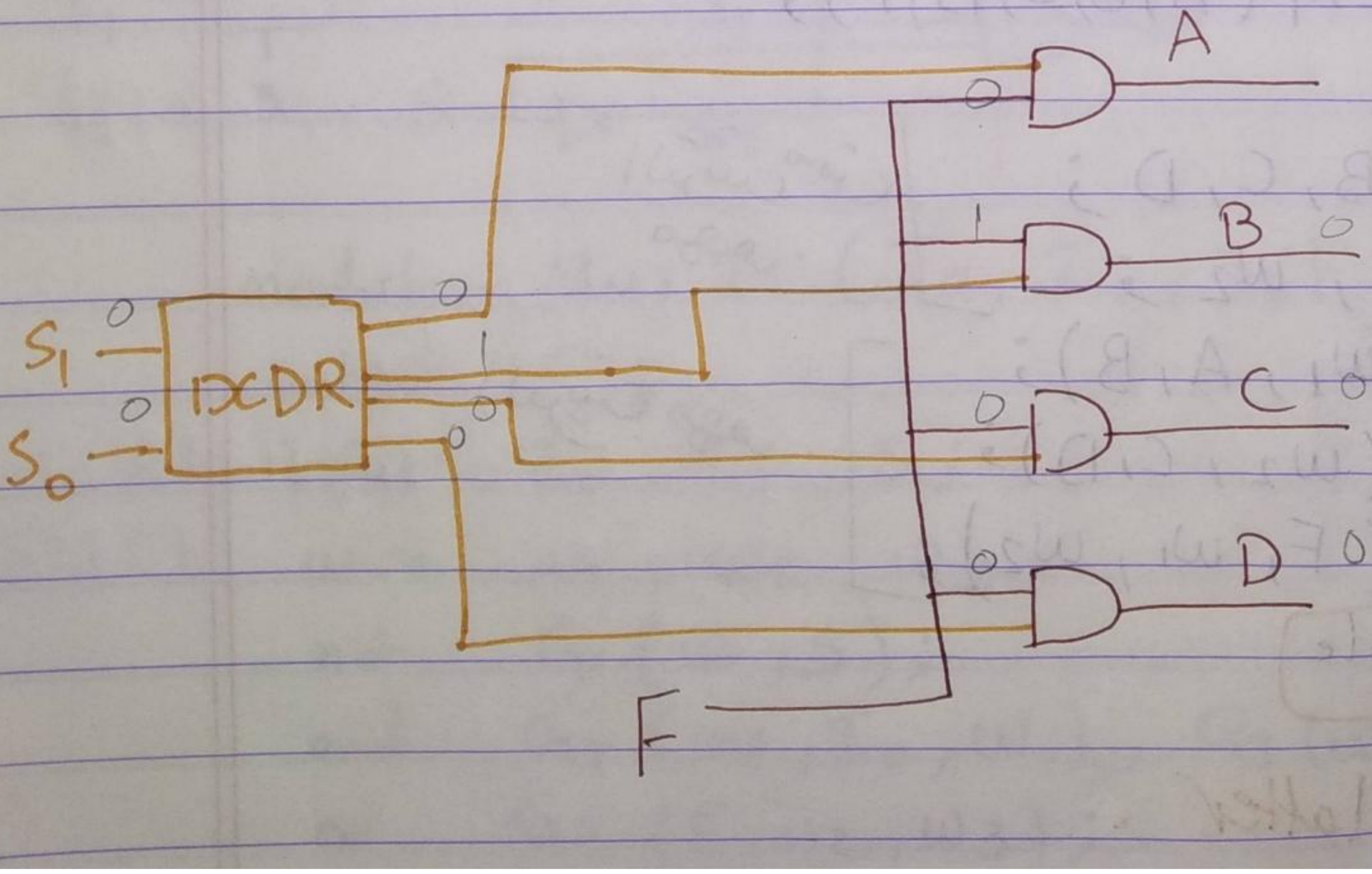


A = F , $S_1 S_0$ = 0 0

B = F , $S_1 S_0$ = 0 1

C = F , $S_1 S_0$ = 1 0

D = F , $S_1 S_0$ = 1 1



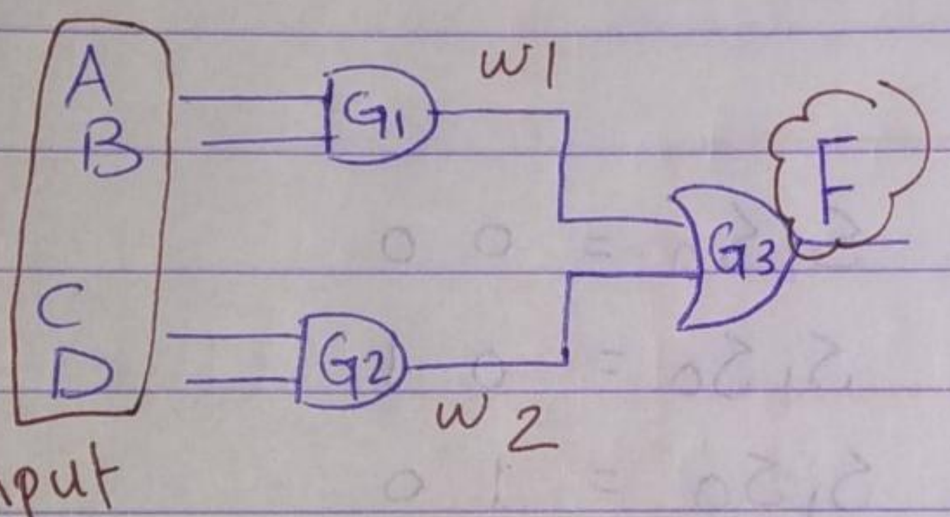
Programming for Combinational Circuit.

HDL (Hardware Description language).

There are 3 ways to build the circuit.

① Gate level Implementation.

ex (Hdl - gate level)



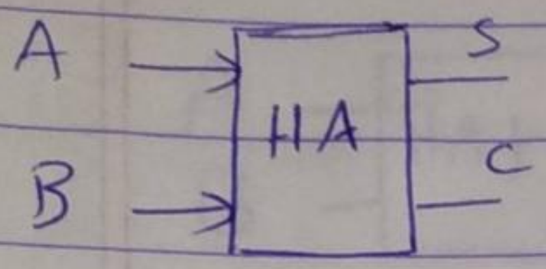

```
module Ex, (A,B,C,D,F);
output F;
input A,B,C,D;
wire w1, w2;
and G1(w1, A, B);
and G2(w2, C, D);
or G3(F, w1, w2);
endmodule
```

الترتيب مش مهم.

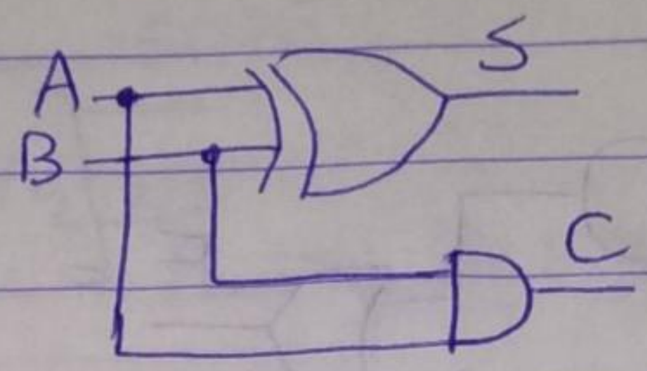
الترتيب مهم.

small latter.

ex Implementation HA (half-adder).

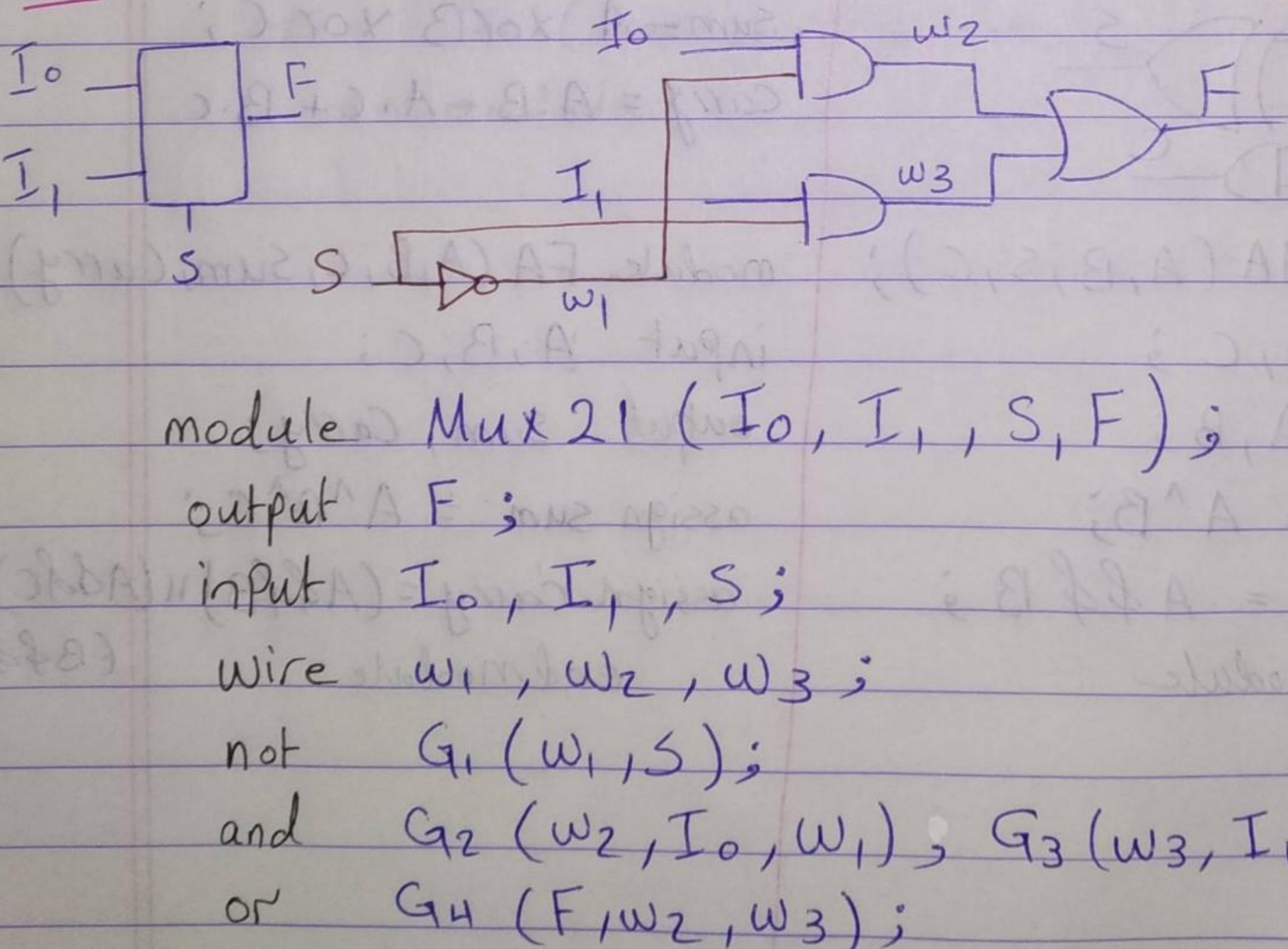


Gate level




```
module Ex2 (A, B, S, C);
 output  S, C;
 input   A, B;
  xor    G1 (S, A, B);
 and     G2 (C, A, B);
    endmodule.
```

ex mux 2x1 Gate-level:

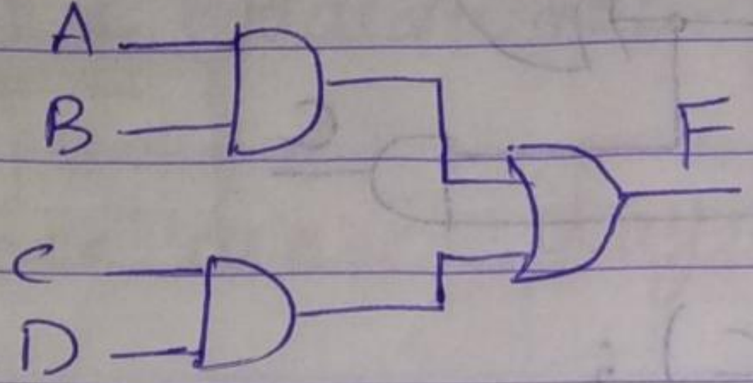



```
module Mux21 (I0, I1, , S, F);
 output F ;
 input  I0, I1, , S;
 wire   w1, w2, w3;
 not    G1 (w1, S);
 and    G2 (w2, I0, w1); G3 (w3, I1, , S);
 or     G4 (F, w2, w3);
    end module
```

(wire لغة)

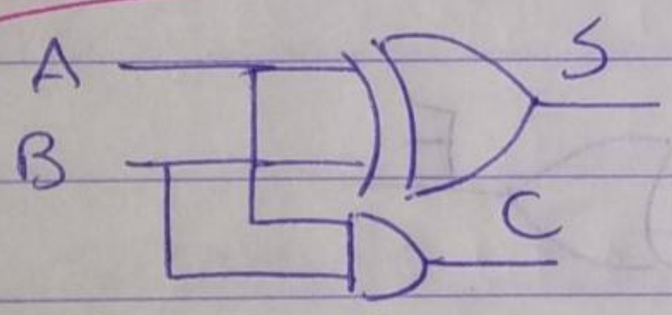
[2] Data Flow implementation (assign).

ex




```
module Ex (A, B, C, D, F);
input A, B, C, D;
output F;
assign F = (A && B) || (C && D);
    endmodule
```

ex HA




```
module HA (A, B, S, C);
output S, C;
input A, B;
assign S = A ^ B;
assign C = A && B;
    endmodule
```

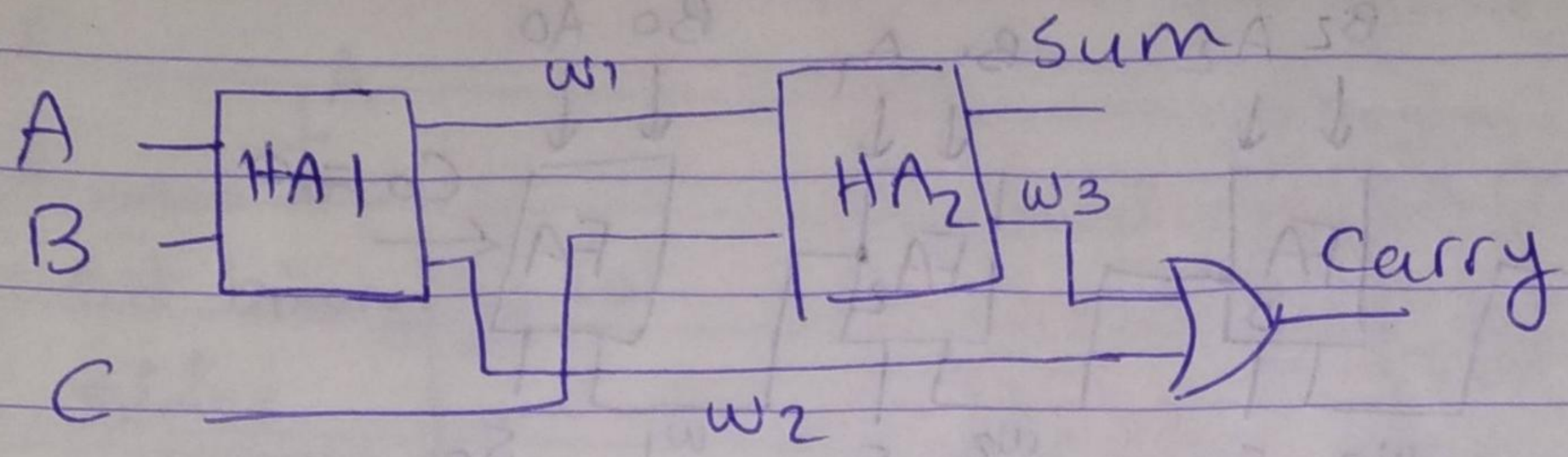
Full-adder

Sum = A xor B xor C;

carry = A.B + A.C + B.C

```
module FA (A, B, C, Sum, Carry);
input A, B, C;
output Sum, Carry;
assign sum = A ^ B ^ C;
assign carry = (A && B) || (A && C) ||
               (B && C);
    endmodule
```

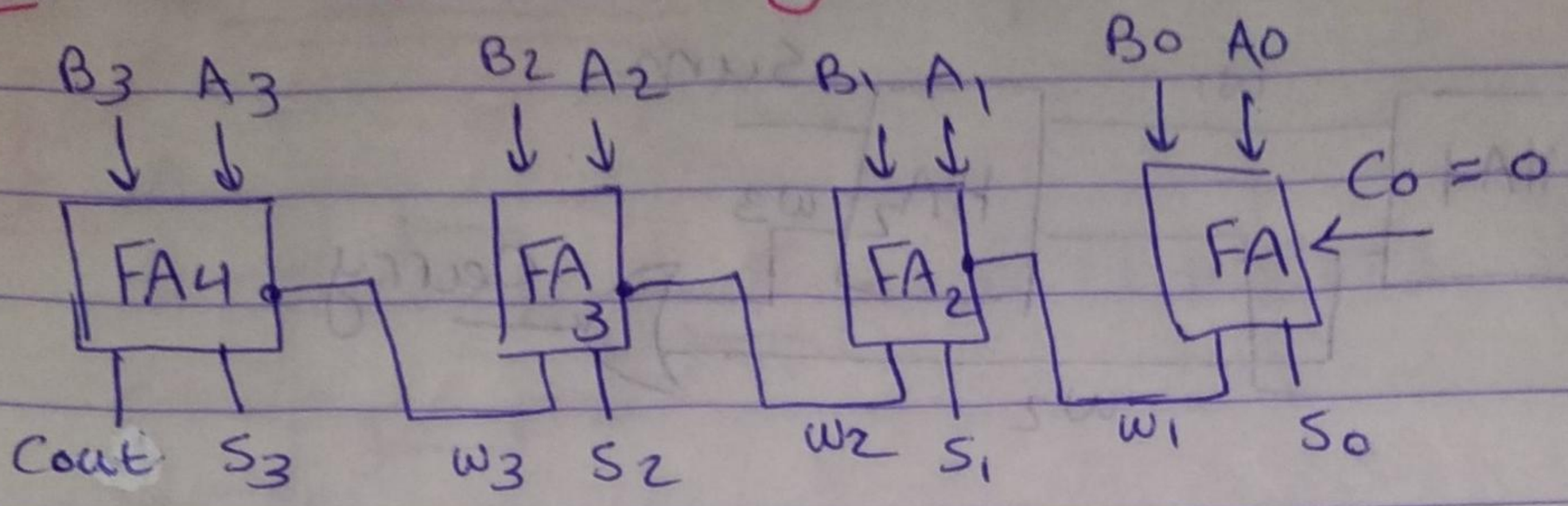
تكملة ⇐



Structure :-

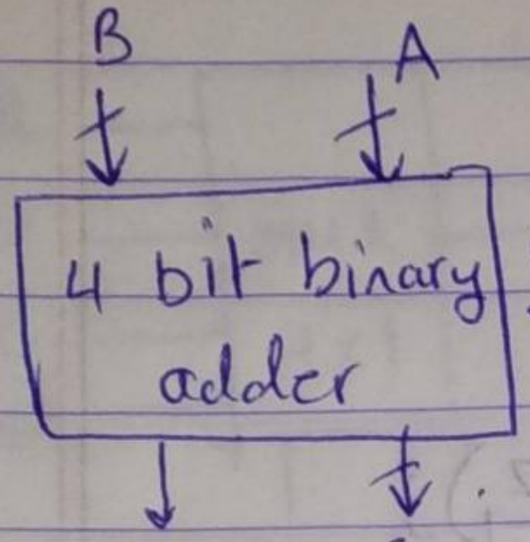
```
module FA (A, B, C, Sum, carry);
input   A, B, C;
output  Sum, carry;
wire    w1, w2, w3;
HA      HA1 (A, B, w1, w2);
HA      HA2 (w1, C, Sum, w2);
or      G1 (carry, w3, w2);
endmodule
```

ex 4 bit Binary adder:



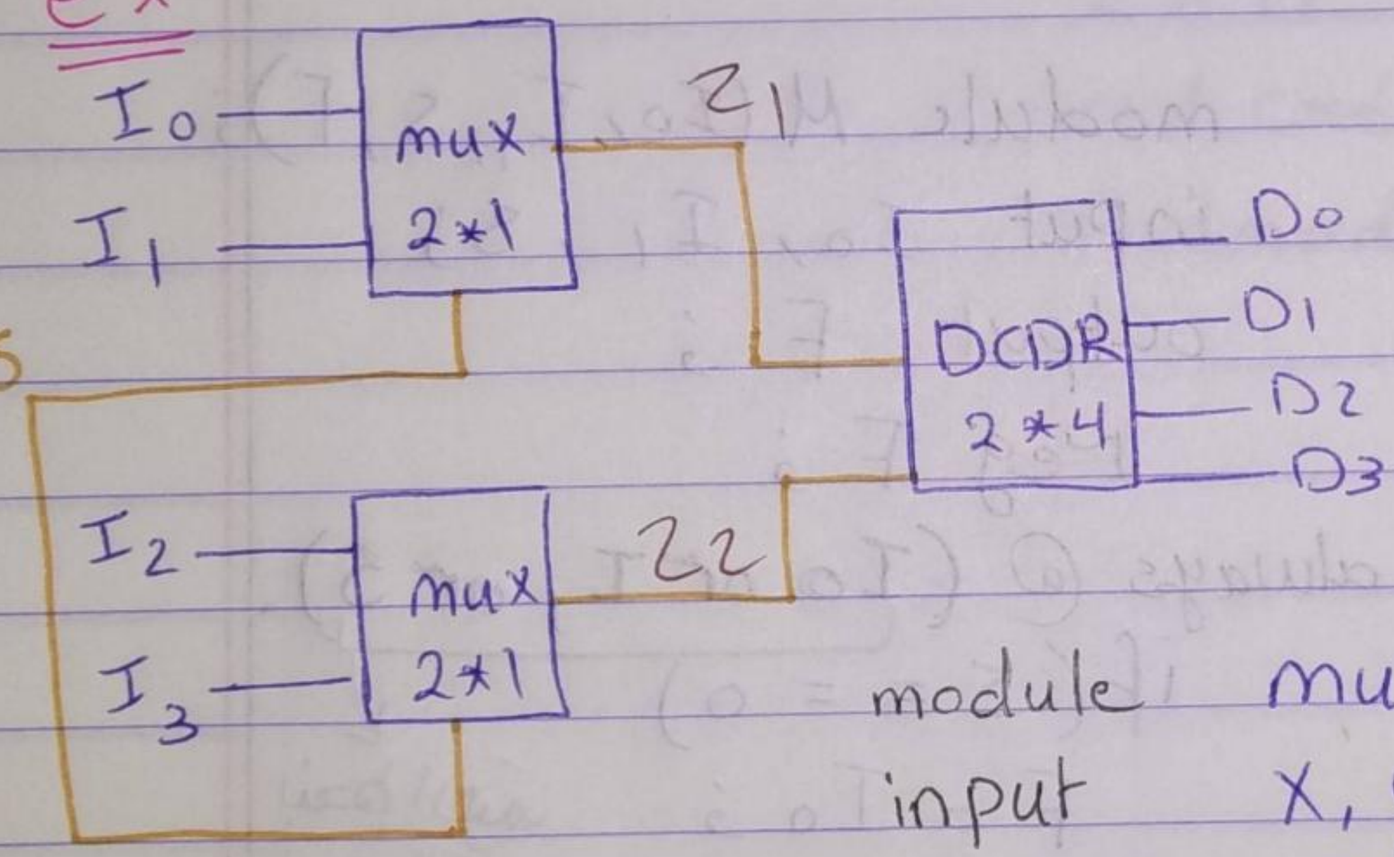

```
module 4BA(A, B, Co, S, Cout);
input Co;
input [3:0] A, B;
output [3:0] S;
output Cout;
wire w1, w2, w3;
FA FA1(A[0], B[0], Co, S[0], w1);
FA FA2(A[1], B[1], w1, S[1], w2);
FA FA3(A[2], B[2], w2, S[2], w3);
FA FA4(A[3], B[3], w3, S[3], Cout);
endmodule
```

* Data Flow (4 bit Binary adder).

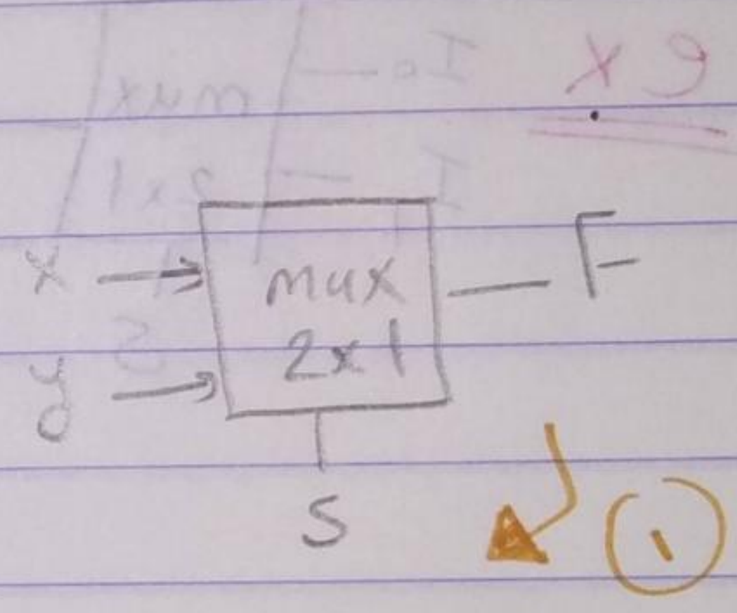



```
module BA(A,B,Co,Cout,S);
input [3:0] A,B;
input Co;
output [3:0] S;
output Cout;
assign {Cout,S} = A+B+Co;
endmodule
```

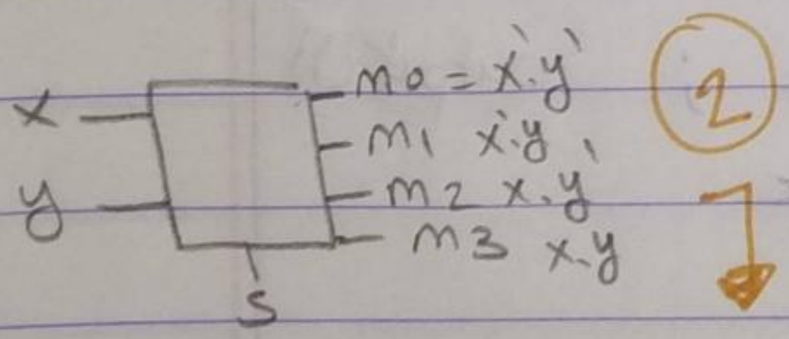
ex



ex




```
module muX21(X,y,S,F);
input X,y,S;
output F;
assign F = (X && !S) || (y && S);
endmodule
```



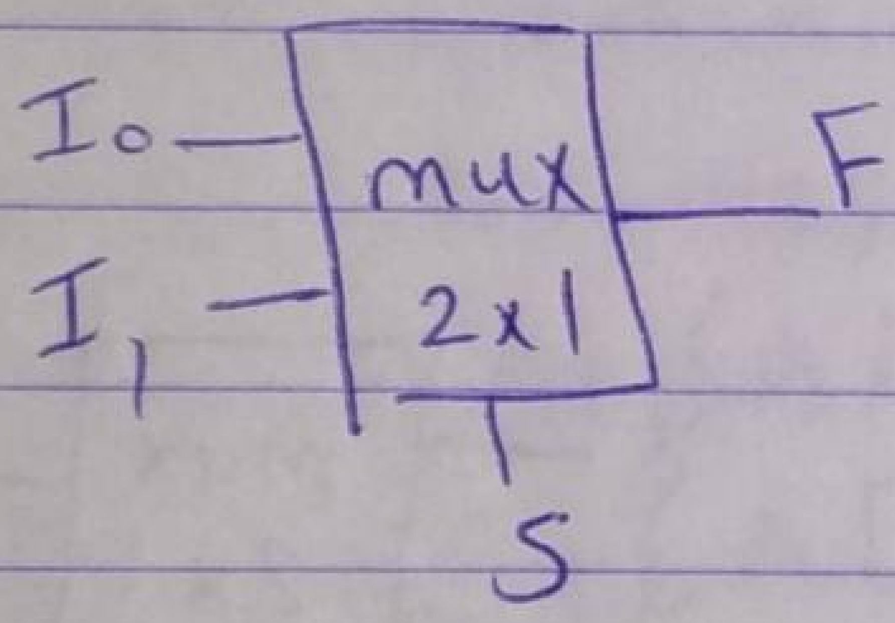

```
module DCDR(x,y,m);
input x,y;
output [0:3] m;
assign m[0] = (!x && !y);
       m[1] = (!x && y);
       m[2] = (x && !y);
       m[3] = (x && y); endmodule.
```

(3)

```
module system (I, D, S);
input [0:3] I;
input S;
output [0:3] D;
wire z1, z2;
mux21 M1 (I[0], I[1], S, Z1);
mux21 M2 (I[2], I[3], S, Z2);
endmodule
```

(3) Behaviral (always).

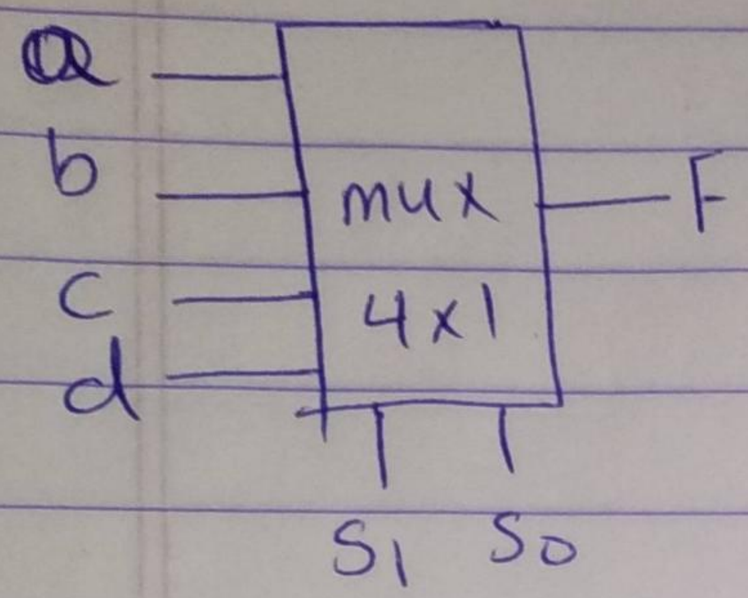
ex




```
module M (I0, I1, S, F);
input I0, I1, S;
output F;
reg F;
always @ (I0 or I1 or S).
if (S = = 0)
F = I0;
else
F = I1;
endmodule
```

بنحط القيم الي بتأثر على ال output

mux 4 x 1




```
module M1(a,b,c,d,S0,S1,F);
input  a,b,c,d,S0,S1;
output F;
reg    F;
always@(a or b or c or d or S0,S1)
  Case {S1,S0}
```

Binary ←

```
  2'b00 : F<=a;
  2'b01 : F<=b;
  2'b10 : F<=C;
  2'b11 : F<=d;
end case
endmodule
```