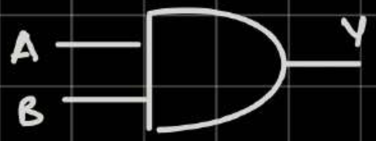
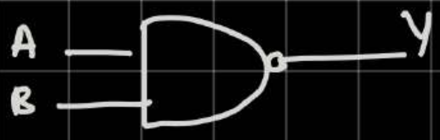
UNIT-1

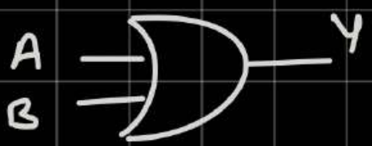
BASIC FUNCTION/GATES



AND

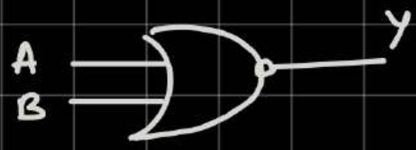
NAND





OR

NOR



XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

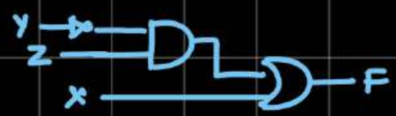
BOOLEAN FUNCTION: algebraic expression consists of binary variables and logic operations

Ex: $F = x + y'z$

Number of rows in truth table $= 2^n$

Binary number - counting from 0 through $2^n - 1$

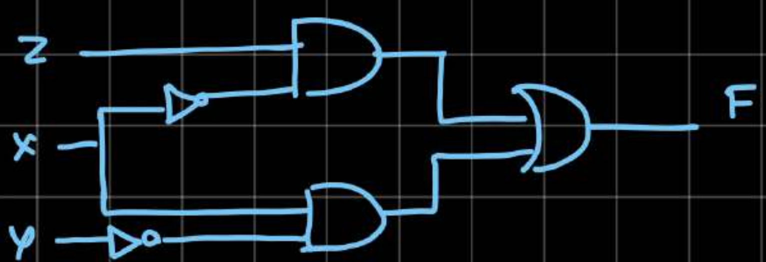
Q. $F = x + y'z$. Write the TT & gate implementation



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

Q. $F = x'y'z + x'yz + xy'$

$F = x'y'z + x'yz + xy' = x'z + xy'$



BOOLEAN IDENTITIES/LAWS

principle of duality: $+ \longleftrightarrow \cdot$ then $0 \longleftrightarrow 1$

	LAW	DUAL LAW
COMMUTATIVE	$a \cdot b = b \cdot a$	$a + b = b + a$
ASSOCIATIVE	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$	$(a+b)+c = a+(b+c)$
DISTRIBUTIVE	$a \cdot (b+c) = a \cdot b + a \cdot c$	$a + (b \cdot c) = (a+b) \cdot (a+c)$
DEMORGAN	$(a+b)' = a' \cdot b'$	$(a \cdot b)' = a' + b'$
IDEMPOTENCY	$a \cdot a = a$	$a + a = a$
IDENTITY	$a \cdot 1 = a$	$a + 0 = a$
BOUNDEDNESS	$a \cdot 0 = 0$	$a + 1 = 1$
COMPLEMENT	$a \cdot \bar{a} = 0$	$a + \bar{a} = 1$
ABSORPTION	$a + a \cdot b = a$	$a \cdot (a+b) = a$
INVOLUTION	$a = a$	$a = a$
USEFUL IDENTITY	$a + \bar{a} \cdot b = a + b$	$a \cdot (\bar{a} + b) = a \cdot b$

Q. Simplify the following Boolean expression

1. $x(x'+y) = x \cdot x' + xy = xy$
2. $x + x'y = (x+x')(x+y) = 1(x+y) = x+y$
3. $(x+y)(x+y') = x + xy' + xy + y \cdot y' = x(1+y'+y) = x$
4. $xy + x'z + yz = xy + x'z + yz(x+x') = xy + x'z + xyz + x'yz$
   $= xy(1+z) + x'z(1+y) = xy + x'z$
5. $(x+y)(x'+z)(y+z) = (x+y) \cdot (x'+z)$
   (Duality of the previous question)

Q. Find complement & duality

1. $F_1 = (x'yz' + x'y'z)$
   Duality: $(x'+y+z') \cdot (x'+y'+z)$
   Complement: $F_1' = (x+y'+z)(x+y+z')$

2. $F_2 = x(y'z' + yz)$
   Duality: $x + (y'+z') \cdot (y+z)$
   Complement: $F_2' = x' + (y+z)(y'+z')$

CANONICAL / STANDARD FORMS

			MINTERMS		MAXTERMS	
X	Y	Z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

$2^n \rightarrow$ distinct min/maxterms (SOP) (Σ) (POS) (Π)

No. of function = 2^{2^n}
as a product of max/minterms

Max $\rightarrow$ 0 = a
Min $\rightarrow$ 0 = a'

Q. Find minterms for $F = A + B'C$

$$F(A,B,C) = \Sigma(1,4,5,6,7)$$

Q. Express $F = xy + x'z$ as product of maxterms

$$F(x,y,z) = \Pi(0,2,4,5)$$

Q. Find min & maxterms for

① 00011011

SOP = $\Sigma(3,4,6,7)$
POS = $\Pi(0,1,2,5)$

② 01011010

SOP = $\Sigma(1,3,4,6)$
POS = $\Pi(0,2,5,7)$

GATE MINIMIZATION [K-MAPS]

K-map → Karnaugh map

IMPLICANT → composed of squares containing 1's
No. of squares in 2^n
double the area → one less literal

PRIME IMPLICANT → Largest no. of squares obeying rules

ESSENTIAL PRIME IMPLICANT → containing square in no other prime implicant

→ include all required prime implicants

→ Convert required implicants to Boolean formula.

3 prime implicant

2 essential prime implicant

required prime Implicant
↓
boolean formula

TWO VARIABLE K-MAP

$x \backslash y$	0	1
0	m_0	m_1
1	m_2	m_3

⇒

$x \backslash y$	0	1
0	$x'y'$	$x'y$
1 (x)	xy'	xy

$\underbrace{\quad}_{y}$

THREE VARIABLE K-MAP

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

$\Rightarrow$

x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

FOUR VARIABLE K-MAP

wx \ yz	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

wx \ yz	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

① Simplify the Boolean fn $F(x,y,z) = \Sigma(2,3,4,5)$

$x \backslash yz$	00	01	11	10
0	$_0$	$_1$	1_3	1_2
1	1_4	1_5	$_7$	$_6$

$\rightarrow x'y$

$\downarrow xy'$

$F = x'y + xy'$

② $F(x,y,z) = \Sigma(3,4,6,7)$

$x \backslash yz$	00	01	11	10
0			1	
1	1		1	1

$F = yz + xz'$

③ $F(x,y,z) = \Sigma(0,2,4,5,6)$

$x \backslash yz$	00	01	11	10
0	1			1
1	1	1		1

$F = xy' + z'$

④ $F(x,y,z) = \Sigma(0,1,6,7)$

$x \backslash yz$	00	01	11	10
0	1	1		
1			1	1

$F = x'y' + xy$

⑤ $F(x,y,z) = \Sigma(0,1,2,5)$

$x \backslash yz$	00	01	11	10
0	1	1		1
1		1		

$F = x'z' + y'z$

⑥ $F(x,y,z) = \Sigma(0,2,3,4,6)$

$x \backslash yz$	00	01	11	10
0	1		1	1
1	1			1

$F = z' + x'y$

⑦ $F(x,y,z) = \Sigma(1,2,3,5,7)$

$x \backslash yz$	00	01	11	10
0		1	1	1
1		1	1	

$F = z + x'y$

⑧ $F(A,B,C) = A'C + AB'C + BC + A'B$

(a) Express as a sum of minterms

(b) Find the minimal SOP

	00	01	11	10
0		1	1	1
1		1	1	

$F = C + \bar{A}B$

$$F(A,B,C) = A'C(B+B') + AB'C + (A+A')BC + A'B(C+C')$$
$$= A'BC + A'B'C + AB'C + ABC + A'BC + A'BC + A'BC'$$
$$= A'BC + A'B'C + AB'C + ABC + A'BC'$$
$$= m_3 + m_1 + m_5 + m_7 + m_2 = \Sigma(1,2,3,5,7)$$

⑨ $F(A,B,C) = \Sigma(1,2,3,5,6,7)$

A \ BC	00	01	11	10
0		1	1	1
1		1	1	1

$F = C + B$

⑩ $F(A,B,C) = \Sigma(0,1,2,4,5,6)$

A \ BC	00	01	11	10
0	1	1		1
1	1	1		1

$F = B' + C'$

⑪ $F(A,B,C) = \Sigma(0,2,3,4,6,7)$

A \ BC	00	01	11	10
0	1		1	1
1	1		1	1

$F = C' + B$

⑫ $F(A,B,C) = \Sigma(0,3,5)$

A \ BC	00	01	11	10
0	1		1	
1		1		

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

(13) $F(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$

wx \ yz	00	01	11	10
00	1_{0}	1_{1}	$_{3}$	1_{2}
01	1_{4}	1_{5}	$_{7}$	1_{6}
11	1_{12}	1_{13}	$_{15}$	1_{14}
10	1_{8}	1_{9}	$_{11}$	$_{10}$

$F = y' + w'z' + xz'$

(14) $F(w,x,y,z) = \Sigma(0,1,2,6,8,9,10)$

wx \ yz	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

$F = x'y' + w'yz' + wx'z'$

(15) $F(w,x,y,z) = \Sigma(0,2,5,7,8,10,13,15)$

wx \ yz	00	01	11	10
00	1			1
01		1	1	
11		1	1	
10	1			1

$F = w'y'z' + wxz + xyz + x'z'$

(16) $F(w,x,y,z) = \Sigma(2,3,5,7,8,9,10,11,13,15)$

wx \ yz	00	01	11	10
00			1	1
01		1	1	
11		1	1	
10	1	1	1	1

$F = xz + wx' + x'y$

(17) $F(A,B,C,D) = \Sigma(1,2,3,6,8,9,10,12,13,14)$

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

$F = CD' + A'B'D + AC'$

(18) $F(A,B,C,D) = \Sigma(3,4,5,7,9,13,14,15)$

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

$F = AC'D + A'BC' + A'CD + ABC$

* Quad not considered as lesser number of implicants preferred

(19) $F(A,B,C,D) = \Sigma(0,1,5,7,15,14,10)$

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

$F = A'B'C' + A'BD$
$+ ABC + ACD'$

(20) $F(w,x,y,z) = \Sigma(0,4,6,7,8,9,15)$

wx \ yz	00	01	11	10
00	1			
01	1		1	1
11			1	
10	1	1		

$F = w'y'z' + wx'y'$
$+ xyz + w'xy$

(21) $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

$F = A'B'C'D + A'B'C'D'$
$+ AB'CD' + A'B'CD'$
$+ A'BCD' + AB'C'D$
$+ AB'C'D'$

$= m_1 + m_0 + m_{10} + m_2$
$+ m_6 + m_9 + m_8$
$= \Sigma(0,1,2,6,8,9,10)$

$= B'C' + A'CD'$
$+ B'D'$

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

PRODUCT OF SUMS

→ Group all 0's to obtain F'
→ Apply complement to F' to obtain POS [Demorgan]
*→ Maxterms imply the 0's $[\Pi(\,)]$

① Express the following in POS form

$$F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$$

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

$F' = BD' + CD + AB$

$POS = F = (F')'$

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

② POS for $F(x,y,z) = \Pi(0,2,5,7)$

x \ yz	00	01	11	10
0	0			0
1		0	0	

$F' = x'y' + xz$

$POS = F = (x+y)\cdot(x'+z')$

③ Write the TT for $F(x,y,z) = \Sigma m(1,3,4,6)$ & solve for POS and SOP using k map

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

x \ yz	00	01	11	10
0	0	1	1	0
1	1	0	0	1

SOP: $F = x'z + xz'$

POS: $F' = x'z' + xz$

$F = (x+z)\cdot(x'+z')$

DON'T CARE CONDITIONS

under specific conditions function is eql to 1, otherwise 0.

Denoted by X

① Simply the Boolean fn $F(w,x,y,z) = \Sigma(1,3,7,11,15)$ with DCC $d(w,x,y,z) = \Sigma(0,2,5)$

wx \ yz	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

$F = w'x' + yz$

* Quad not necessary since all 1's are grouped

② $F(w,x,y,z) = \Sigma(1,3,7,11,15) + d(1,2,5)$

wx \ yz	00	01	11	10
00		1	1	X
01		X	1	
11			1	
10			1	

$F = yz + w'z$

③ $F(w,x,y,z) = \Sigma(4,5,6,7,12) + d(0,8,13)$

wx \ yz	00	01	11	10
00	X			
01	1	1	1	1
11	1	X		
10	X			

$F = w'x + y'z'$

④ $F(w,x,y,z) = \Sigma(0,1,2,4,5,7,10) + d(3,8,15)$

wx \ yz	00	01	11	10
00	1	1	X	1
01	1	1	1	
11			X	
10	X		1	

$F = w'y' + w'x' + yz$

⑤ $F(w,x,y,z) = \Sigma(2,4,5,13,15) + d(8,9,10,11)$

wx \ yz	00	01	11	10
00				1
01	1	1		
11		1	1	
10	X	X	X	X

$F = w'xy' + wz + x'yz'$

⑥ $F(x,y,z) = \Sigma(1,3) + d(5,6,7)$

x \ yz	00	01	11	10
0		1	1	
1		X	X	X

$F = z$

NAND & NOR IMPLEMENTATION

NAND:

AND-invert

invert-OR

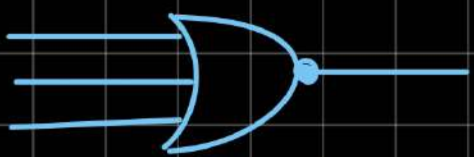
MULTILEVEL CIRCUIT FOR NAND

AND $\longrightarrow$ AND-invert

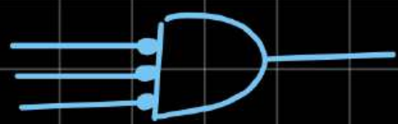
OR $\longrightarrow$ invert-OR

complement input literal or add a bubble when necessary

NOR



OR-invert



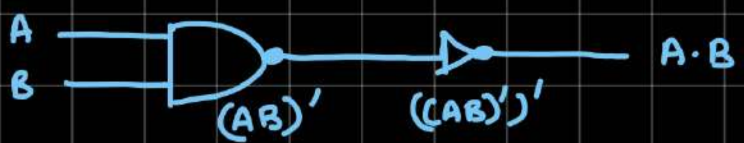
invert-AND

MULTILEVEL to NOR

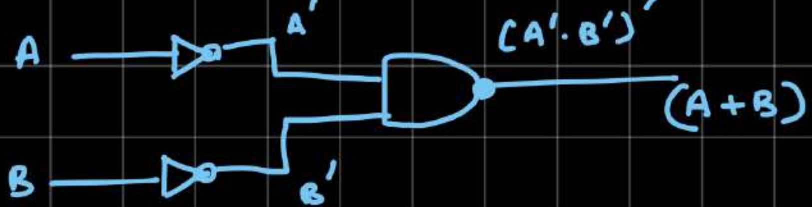
OR $\longrightarrow$ OR-invert

AND $\longrightarrow$ invert-AND

① Build an AND and OR gate using NAND gates



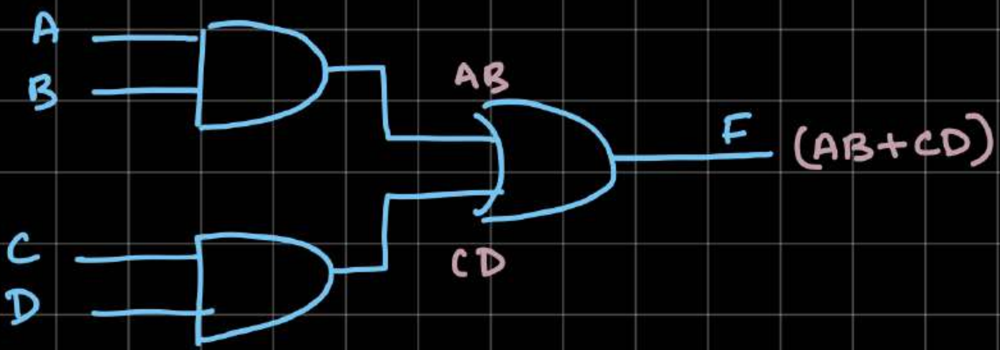
[AND]



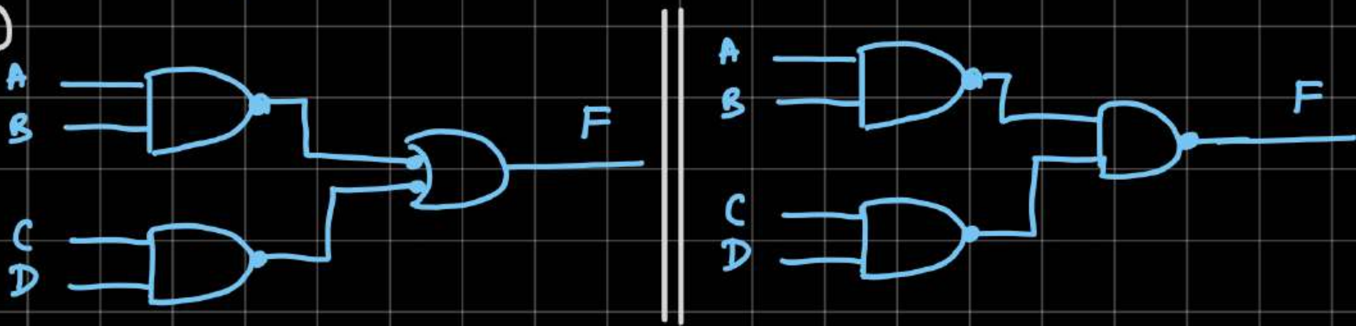
[OR]

② Implement $F = AB + CD$ (a) using AND and OR gate
(b) using NAND gate

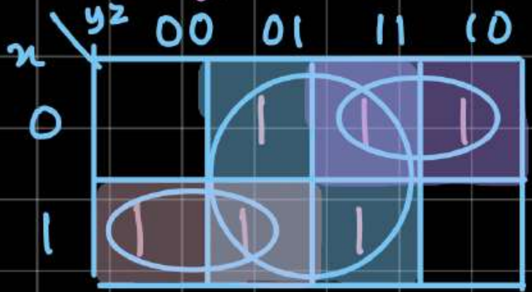
(a)



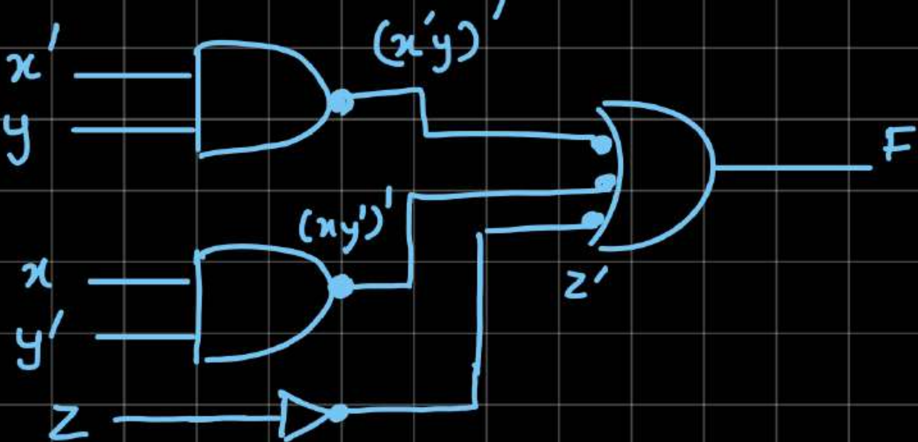
(b)



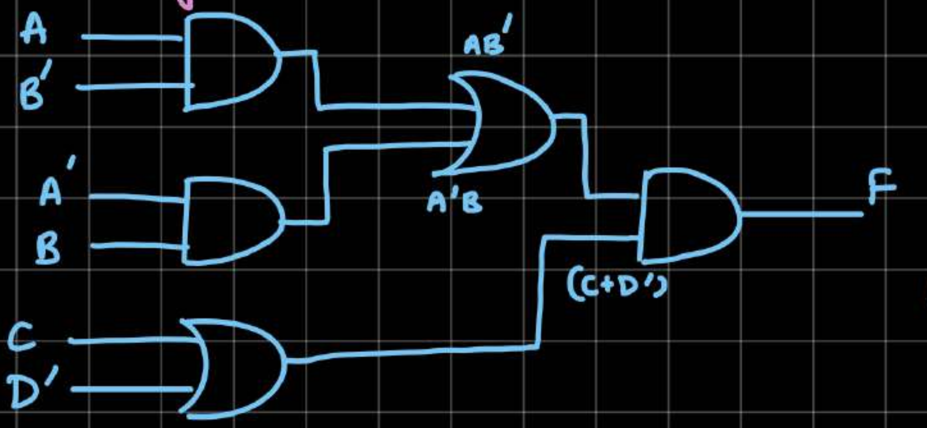
③ Implement the following Boolean function with NAND gates
$F(x,y,z) = (1,2,3,4,5,7)$ using AND-i, i-OR



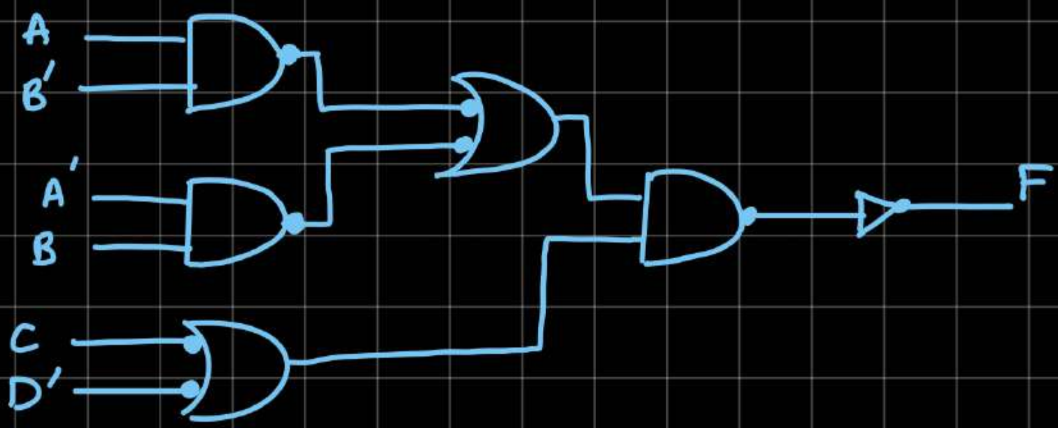
$F = z + x'y + xy'$



④ Implementing $F = (AB' + A'B)(C + D')$ using NAND



[AND-OR]

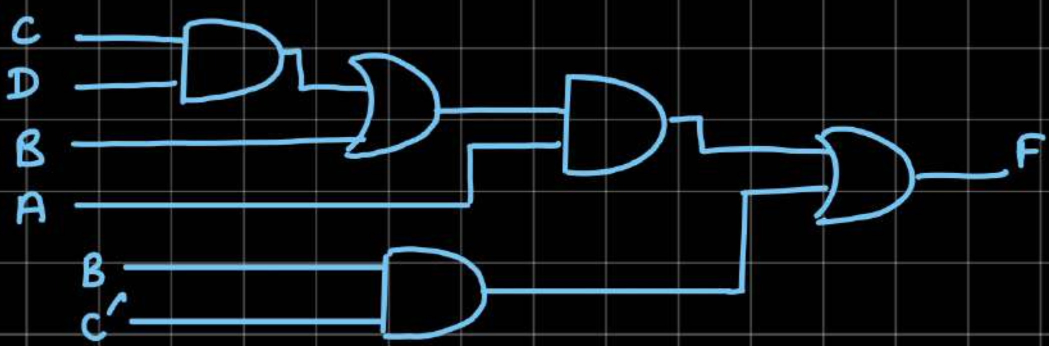


[NAND]

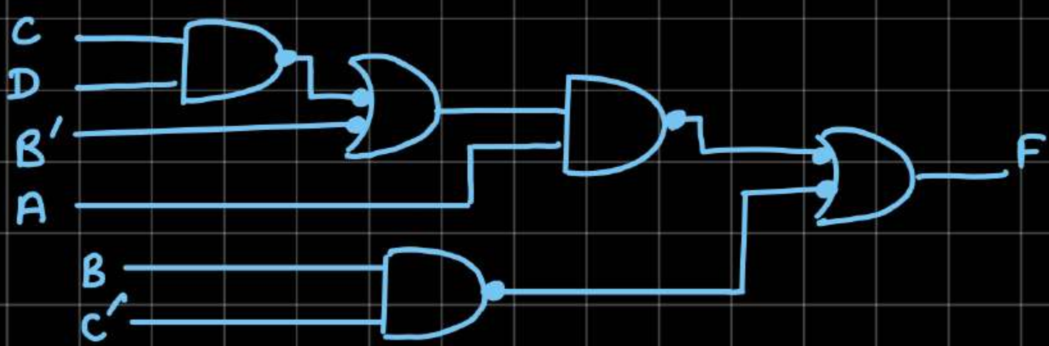
⑤ $F = A(CD + B) + BC'$ using NAND

$F = ACD + AB + BC'$

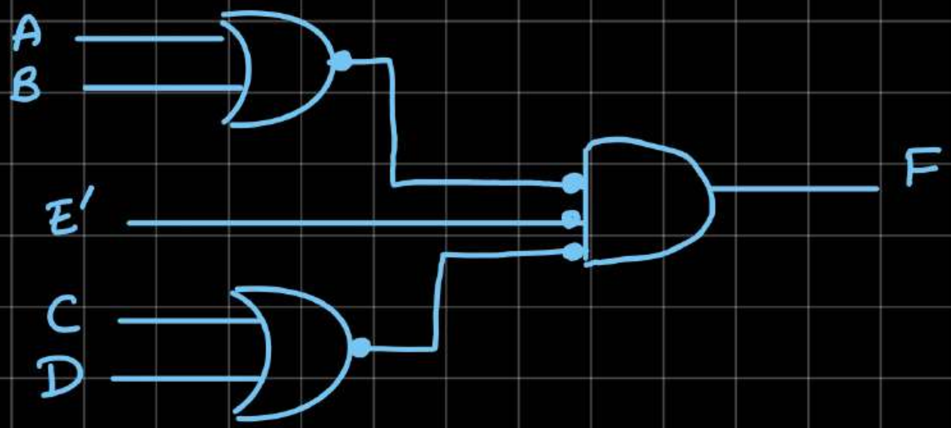
using AND-OR :



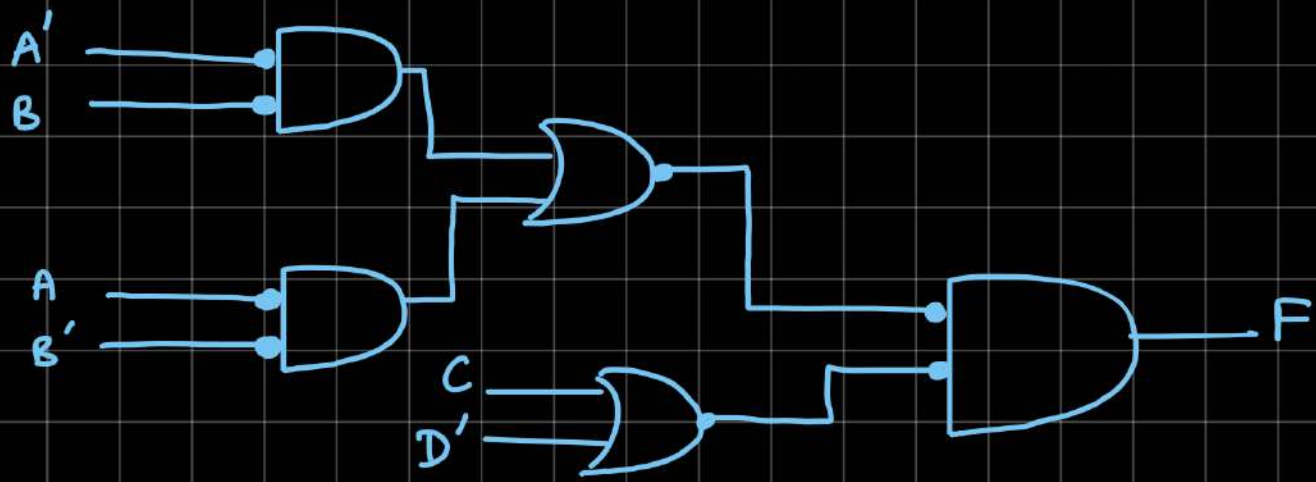
using NAND :



⑥ $F = (A+B)(C+D)E$ using NOR gates

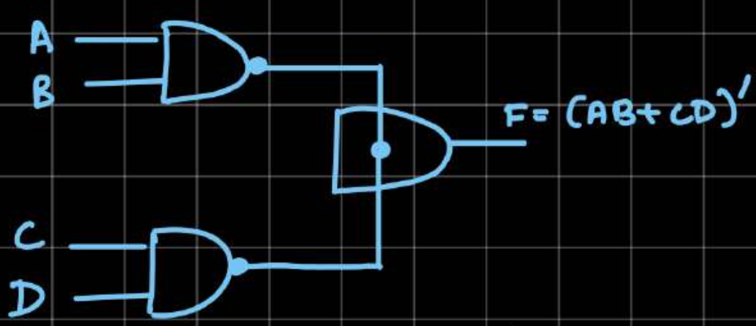


⑦ $F = (AB' + A'B)(C+D')$ using NOR gates



OTHER TWO-LEVEL IMPLEMENTATION

→ WIRED LOGIC

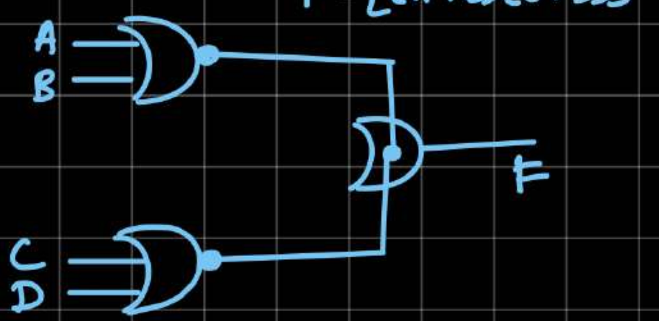


AND-OR-INVERT

Wired-AND logic with 2NAND

$$F = (AB)' \ldots (CD)' = (AB+CD)'$$
$$= (A'+B')(C'+D')$$

$F = [(A+B)(C+D)]'$



OR-AND-INVERT

Wired-OR in emitter-coupled logic gates

$$F = (A+B)' + (C+D)'$$
$$= [(A+B)(C+D)]'$$

NONDEGENERATE FORMS

one gate for first level and another for second level
16 possible combinations
- 8 are degenerate as they degenerate into a single operation
- 8 nondegenerate are used to produce SOP & POS

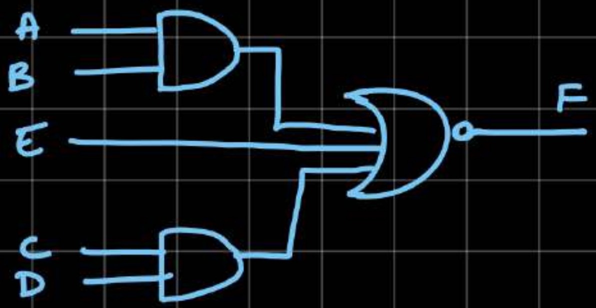
8 NONDEGENERATE FORMS:

AND-OR	OR-AND
NAND-NAND	NOR-NOR
NOR-OR	NAND-AND
OR-NAND	AND-NOR

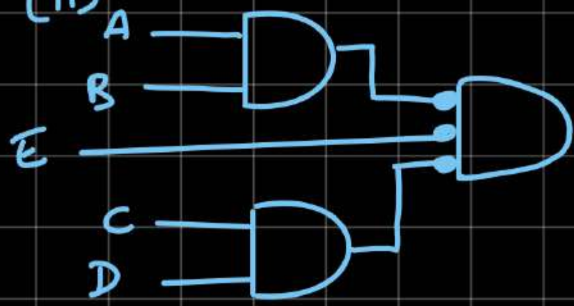
EQUIVALENT NONDEGENERATE (a)	(b)	Implements the fn	Simplify F'	To get Output
AND-NOR	NAND-AND	AND-OR-i	SOP by combining 0's	F
OR-NAND	NOR-OR	OR-AND-i	POS by combining 1's & complementing	F

① Implement the following function $F=(AB+CD+E)'$ using (i) AND-OR-invert (ii) AND-invert AND (iii) NAND-AND

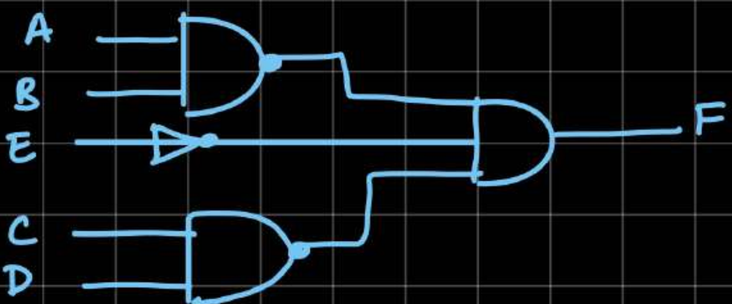
(i)



(ii)

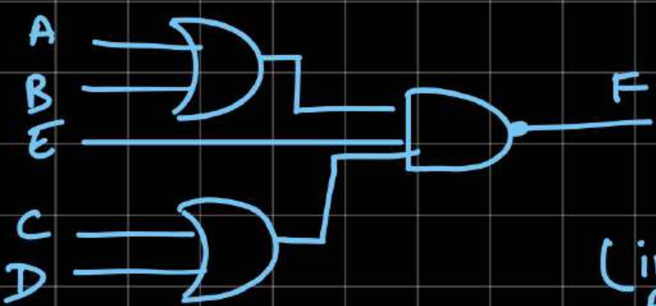


(iii)

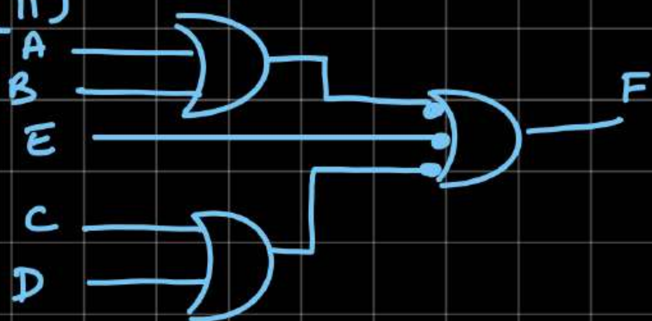


② Represent the following function $F=[(A+B)(C+D)E]'$ using (i) OR-AND-invert (ii) OR-invert-OR (iii) NOR-OR

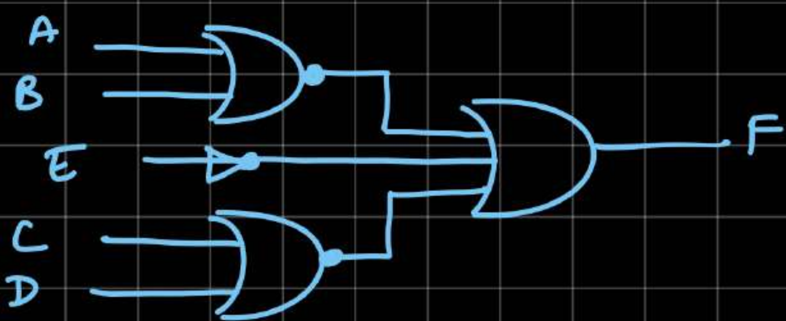
(i)



(ii)



(iii)



③ Implement $f(x, y, z) = M(0,6)$ with the four 2-level forms. Write SOP & F' form

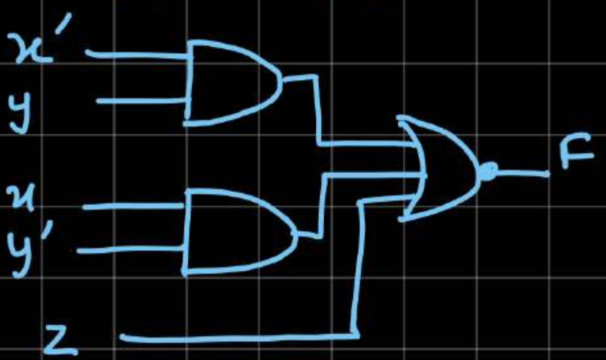
$x \backslash yz$	00	01	11	10
0	1	0	0	0
1	0	0	0	1

$F = x'y'z' + xyz' \longrightarrow$ SOP

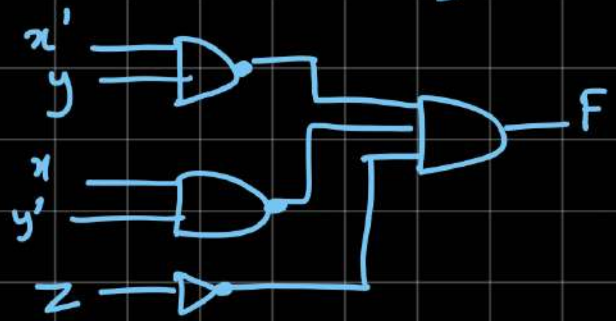
$F' = xy' + z + x'y$

POS: $F = (x'+y)(z')(x+y')$

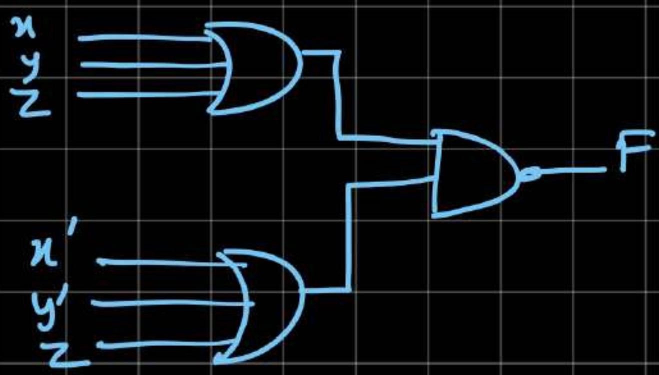
AND-NOR



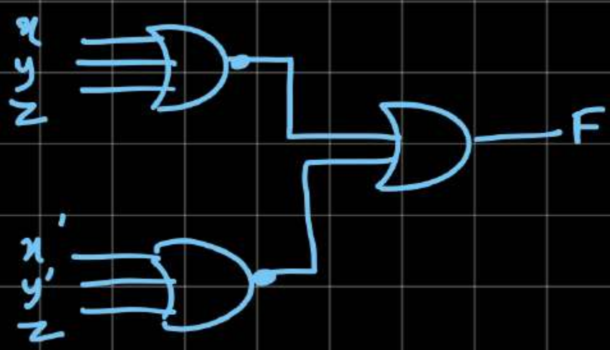
NAND-AND



OR-NAND



NOR-OR



EXCLUSIVE-OR FUNCTION (XOR)

$x \oplus y = xy' + x'y$

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

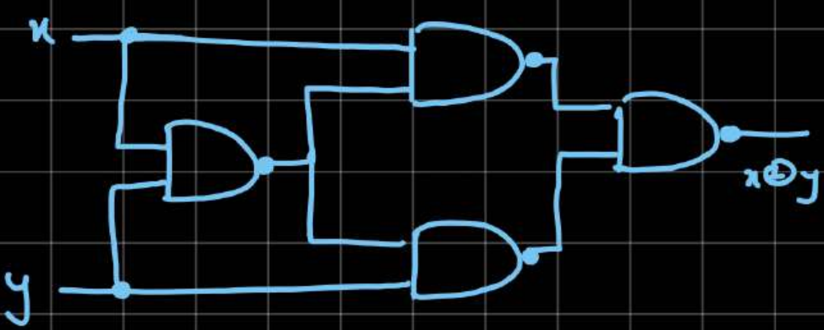
$x \oplus 0 = x$

$x \oplus 1 = x'$

$x \oplus x = 0$

$x \oplus x' = 1$

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



ODD FUNCTION
- ↳ odd parity
- ↳ odd no. of 1's

$A \oplus B \oplus C$

Even fn $\rightarrow (A \oplus B \oplus C)'$

Identify the minterms for odd & even functions for 3 variables

A \ BC	00	01	11	10
0		1		1
1	1		1	

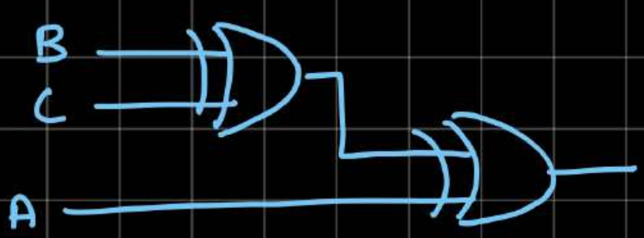
odd fn

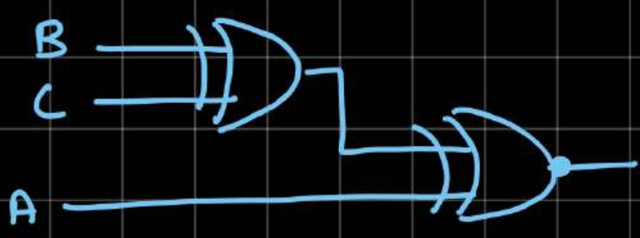
$F = A \oplus B \oplus C$
$= \Sigma(1, 2, 4, 7)$

A \ BC	00	01	11	10
0	1		1	
1		1		1

even fn

$F = (A \oplus B \oplus C)'$
$= \Sigma(0, 3, 5, 6)$





Identify the minterms for odd & even functions for 4 variables

ODD

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

$F = A \oplus B \oplus C \oplus D$
$= \Sigma(1, 2, 4, 7, 8, 11, 13, 14)$

EVEN

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

$F = (A \oplus B \oplus C \oplus D)'$
$= \Sigma(0, 3, 5, 6, 9, 10, 12, 15)$

PARITY GENERATION & CHECKING

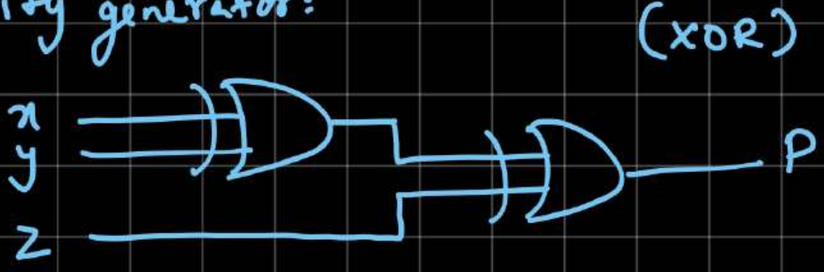
→ Exclusive OR for error detection & correction
→ parity bit → extra bit passed with message to make 1's even/odd.
Circuit that generates parity bit → parity generator
Circuit that checks the parity → parity checker

EVEN PARITY GENERATOR

THREE BIT MESSAGE			PARITY BIT
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Make the total no. of 1s bit even.

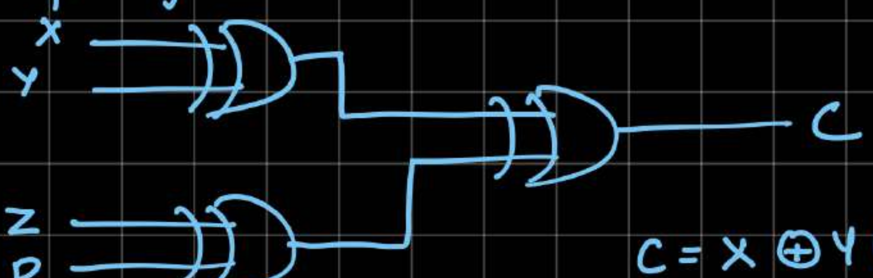
3-bit even-parity generator:



EVEN PARITY CHECKER

FOUR BITS RECEIVED				PARITY ERROR CHECKER
X	Y	Z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
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	1
1	1	1	1	0

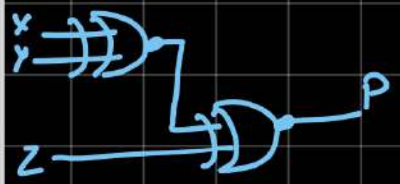
4-bit-even parity checker



$C = X \oplus Y \oplus Z \oplus P$

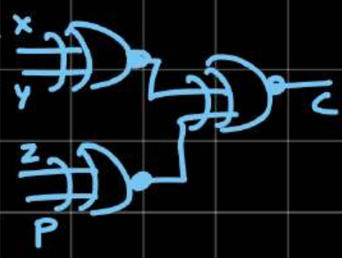
ODD PARITY GENERATOR $P = X \odot Y \odot Z$

THREE BIT MESSAGE			PARITY BIT
X	Y	Z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



ODD PARITY CHECKER $C = X \odot Y \odot Z \odot P$

FOUR BITS RECEIVED				PARITY ERROR CHECKER
X	Y	Z	P	C
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Logic circuits → Combination circuit
→ Sequential circuit — uses storage element

COMBINATIONAL CIRCUITS

↳ output depends on the inputs only

n input variable ⟶ 2^n input combination

n inputs & m outputs

→ ANALYSIS PROCEDURE

→ DESIGN PROCEDURE — CODE CONVERSION

Input → 8 4 2 1 format

BCD — Binary coded decimal [0-9] + Rest are don't care

Excess 3 — "$x+3$"

Gray code — XOR b/w consecutive bit

Build a circuit to convert BCD code to excess 3-code

Decimal	BCD A B C D	excess 3 code w x y z
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

w

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

$w = A + BD + BC$
$= A + B(C+D)$

x

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

$x = BC'D' + B'C + B'D$
$= B'(C+D) + B(C+D)'$

y

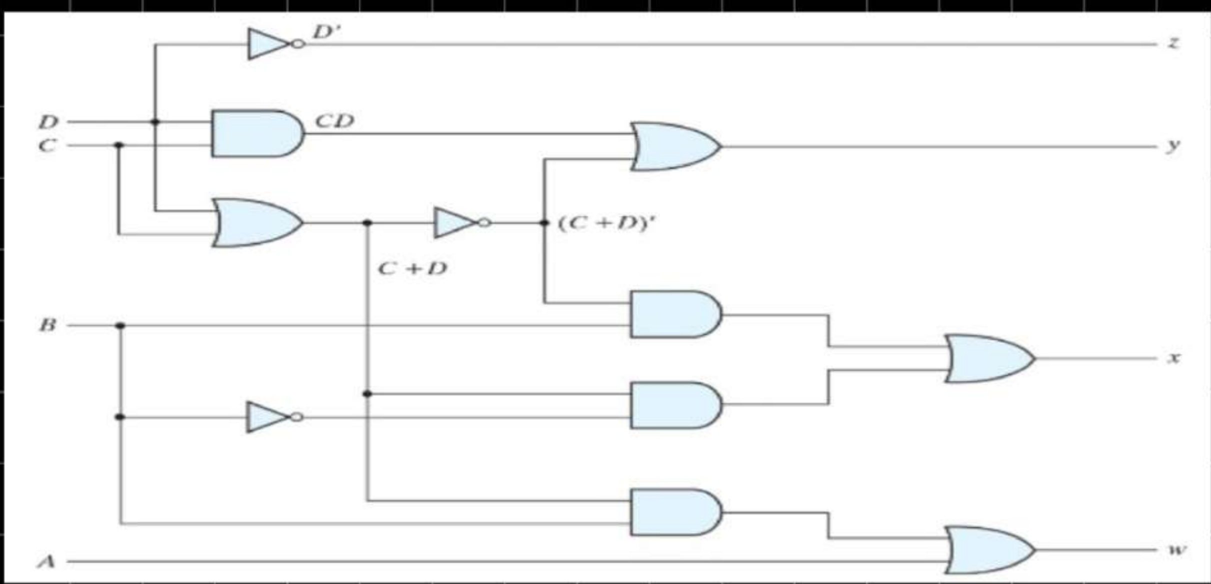
AB \ CD	00	01	11	10
00	1		1	
01	1		1	
11	X	X	X	X
10	1		X	X

$y = C'D' + CD$
$= CD + (C+D)'$

z

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

$z = D'$



Convert 8 4 -2 -1 to excess 3 code

Decimal	8 A	4 B	-2 C	-1 D	excess 3 code w	x	y	z
0	0	0	0	0	0	0	1	1
1	0	1	1	1	0	1	0	0
2	0	1	1	0	0	1	0	1
3	0	1	0	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	1	0	1	1	1	0	0	0
6	1	0	1	0	1	0	0	1
7	1	0	0	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	1	1	1	1	1	0	0

w:

AB \ CD	00	01	11	10
00	0	x	x	x
01	0	0	0	0
11	x	x	1	x
10	1	1	1	x

$w = A$

x:

AB \ CD	00	01	11	10
00	0	x	x	x
01	1	1	1	1
11	x	x	0	x
10	0	0	0	0

$x = \bar{A}B$

y:

AB \ CD	00	01	11	10
00	1	x	x	x
01	1	1	0	0
11	x	x	0	x
10	1	1	0	0

$y = \bar{C}$

z:

AB \ CD	00	01	11	10
00	1	x	x	x
01	1	0	0	1
11	x	x	0	x
10	1	0	0	1

$z = \bar{D}$

Convert BCD to Gray code

Decimal	BCD				GRAY CODE			
	A	B	C	D	w	x	y	z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

0010 BC
XOR
0011 Gray

w:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$w = A$

x:

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$x = \bar{A}B + A\bar{B}$$
$$= A \oplus B$$

y:

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

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

z:

AB \ CD	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$z = \bar{C}D + C\bar{D}$$
$$= C \oplus D$$

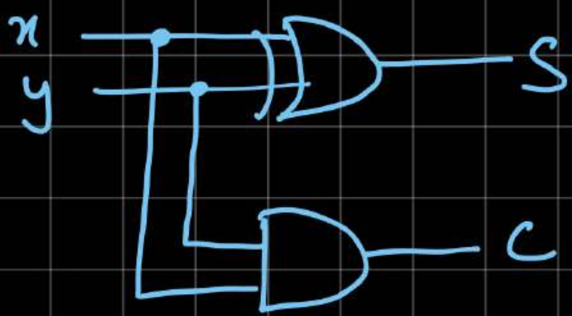
BINARY ADDERS

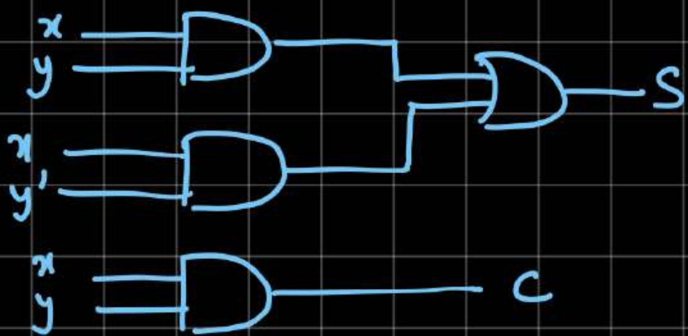
HALF ADDER

$S = x'y + xy' = x \oplus y$

$C = xy$

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





FULL ADDER

x	y	z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = x'y'z + x'yz' + xy'z' + xyz = x \oplus y \oplus z$$

$$C = xy + xz + yz = (x \oplus y)z + xy$$

S:

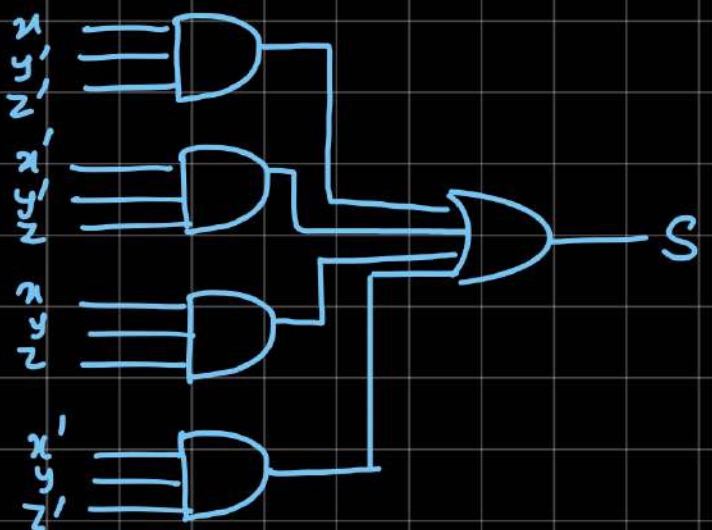
$x \backslash yz$	00	01	11	10
0		1		1
1	1		1	

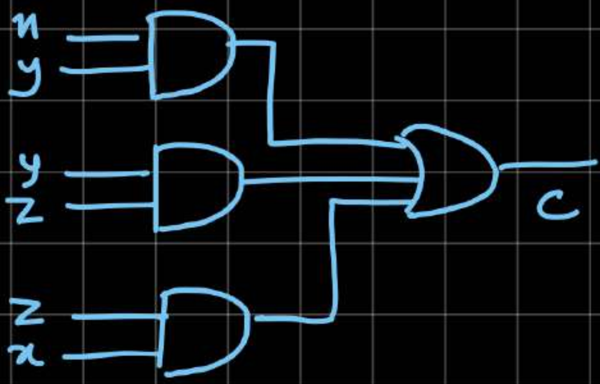
$$S = xy'z' + x'y'z + xyz + x'yz'$$

C:

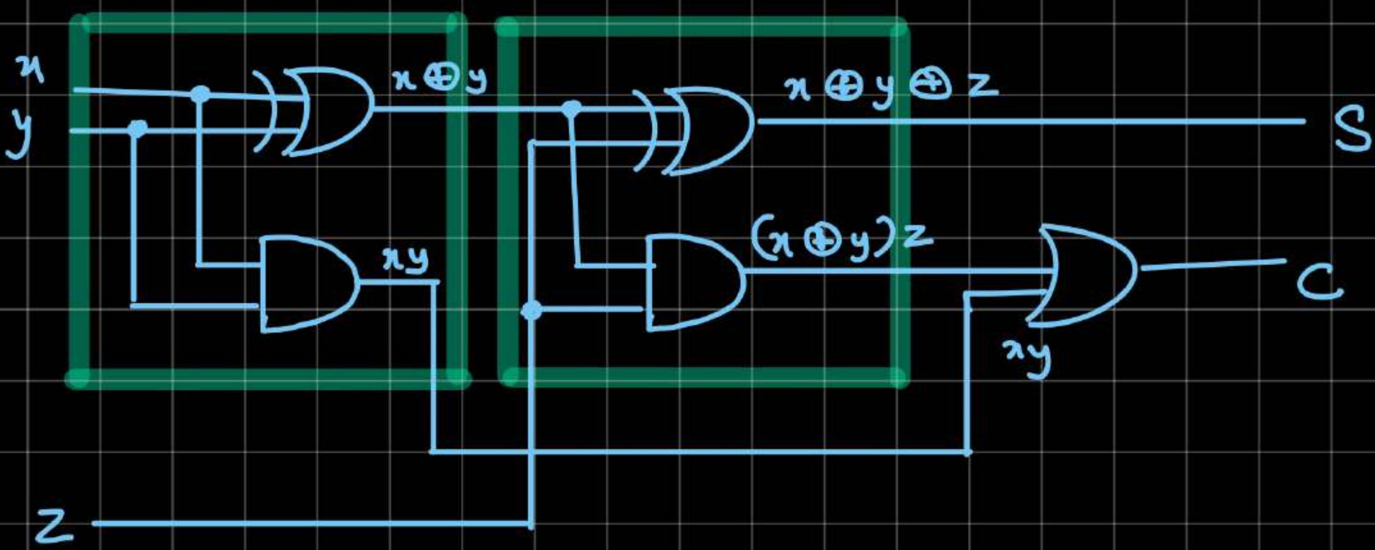
$x \backslash yz$	00	01	11	10
0			1	
1		1	1	1

$$C = xy + xz + yz$$



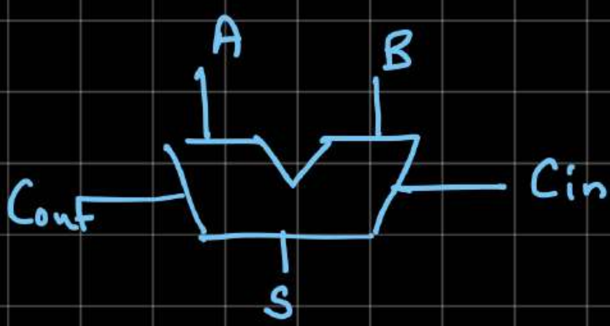


Implement FULL ADDER with two HALF ADDERS



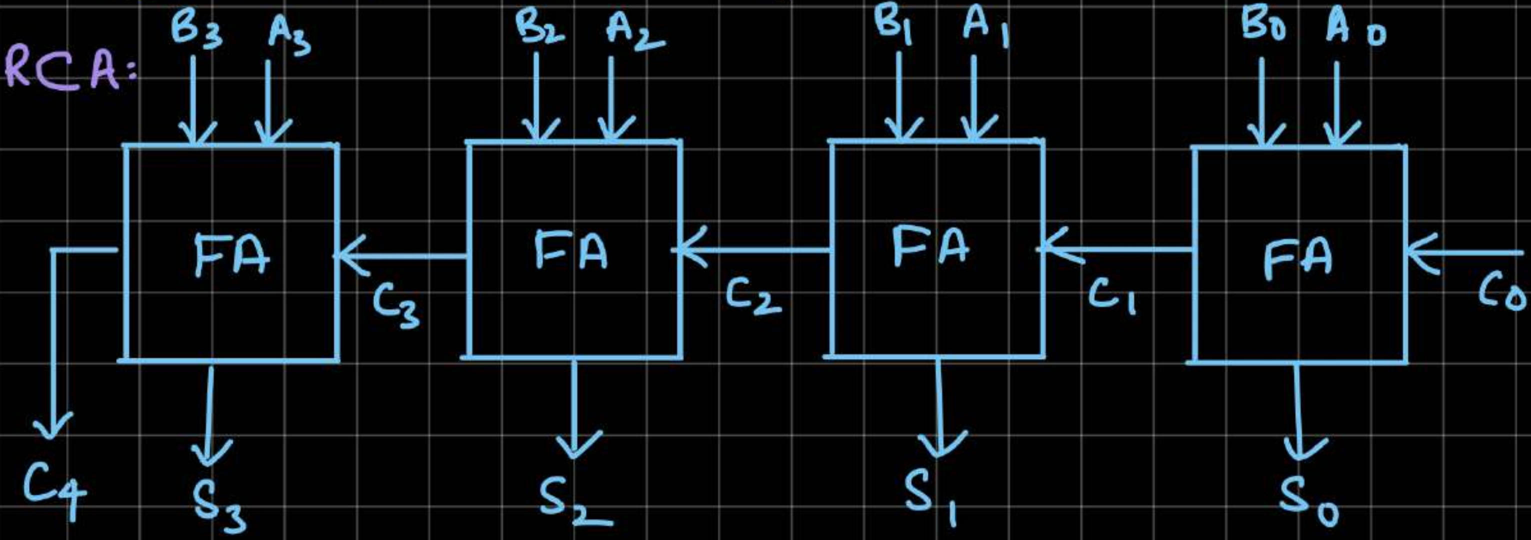
TYPES OF ADDERS

- RCA
- CLA
- Prefix adders



BINARY ADDER

RCA:

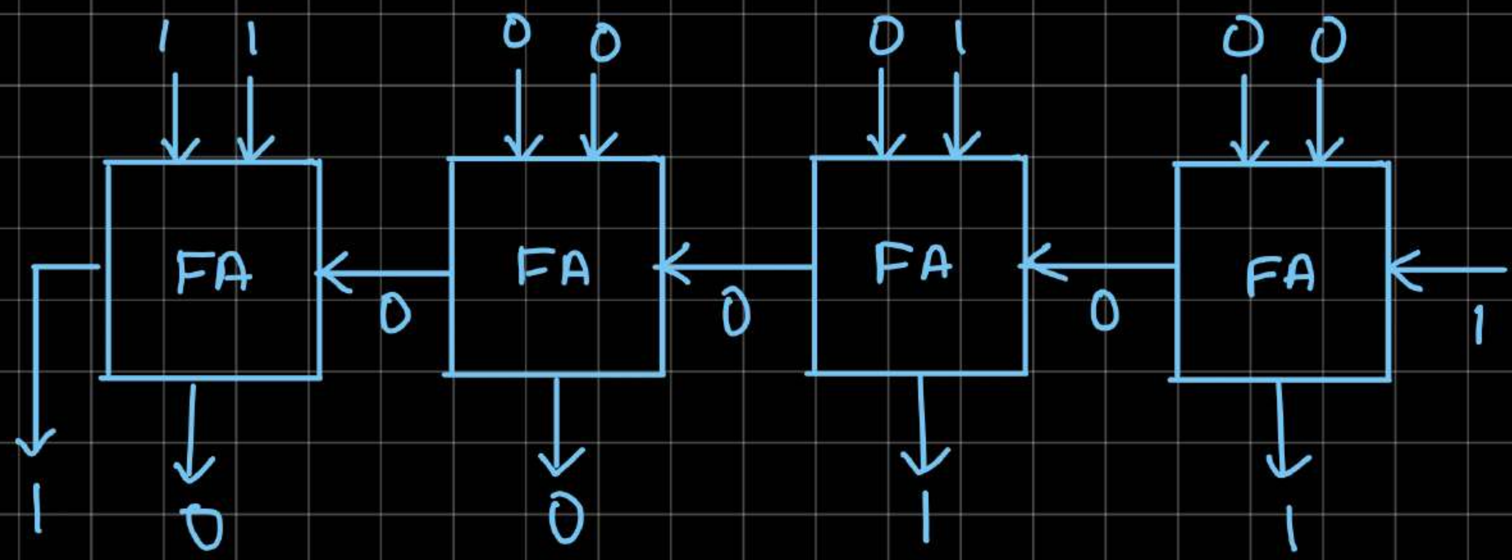


$A = 1011$ & $B = 0011$

$t_{ripple} = N t_{FA}$
RCA delay

Subscript	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

Q. Realise the following addition using Ripple Carry Adder for input $A = 1010$ & $B = 1000$. Consider $C_{in} = 1$



$S = 0011$ $\quad C_{out} = 1$

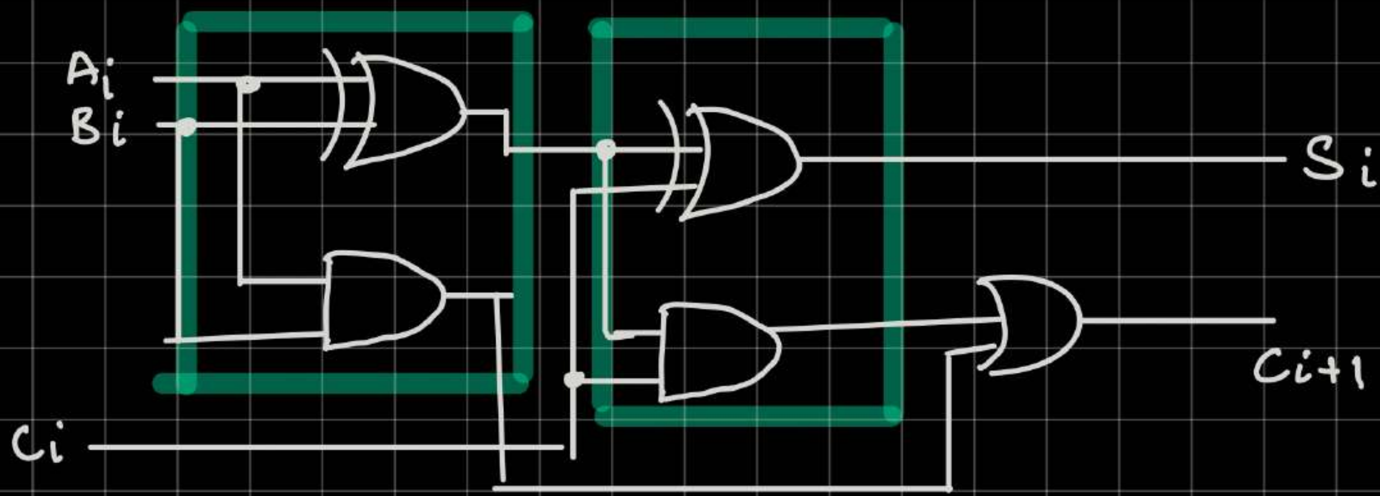
CARRY LOOK AHEAD LOGIC

RCA is slow and follows a sequential chain
To overcome this, CLA is used where they are divided into blocks & solved parallelly

$P_i \longrightarrow$ carry propagate – whether carry will propagate to $i+1$ stage
$G_i \longrightarrow$ Carry generate – Produces carry of 1 if $A_i=1$ & $B_i=1$

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

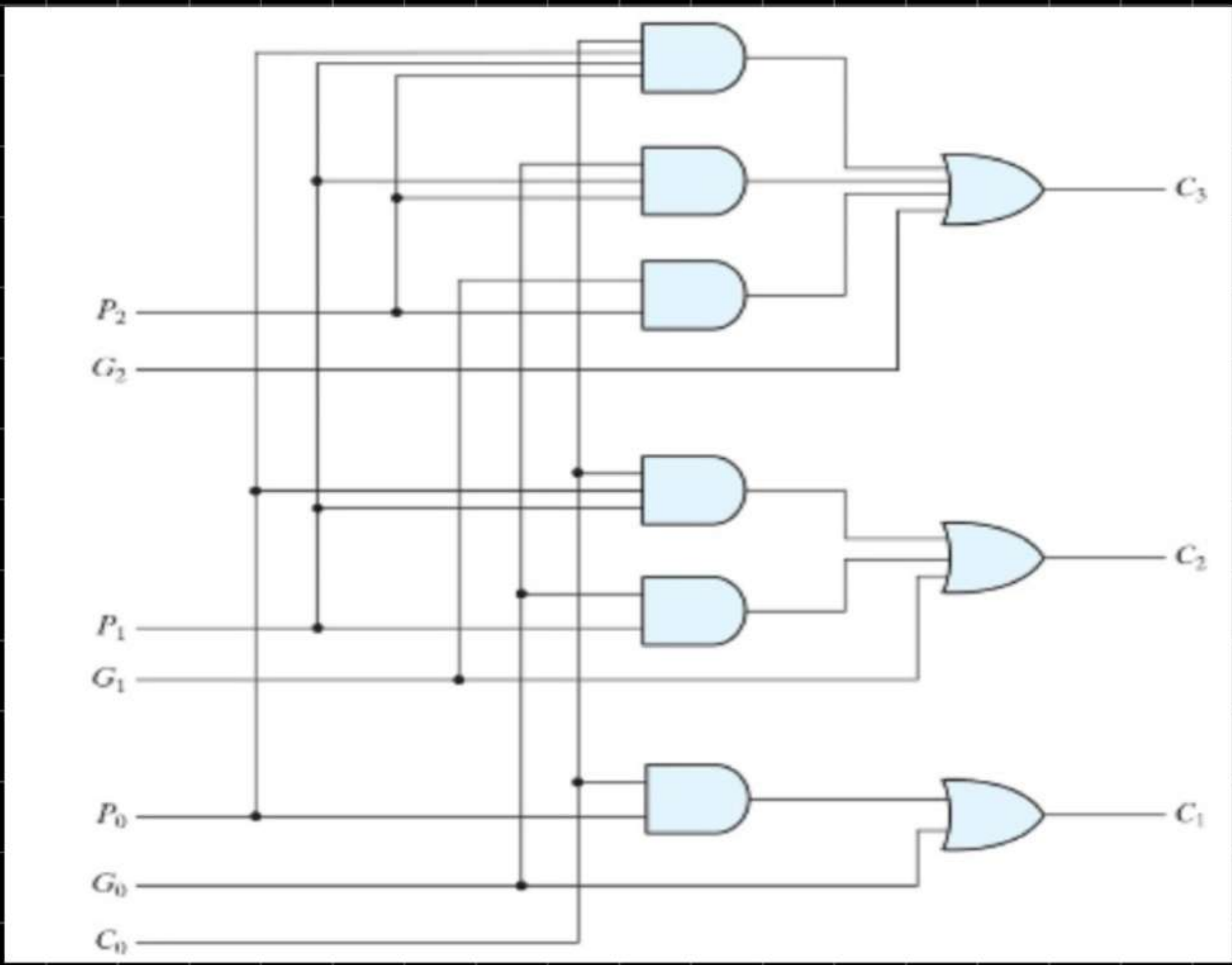
$\because$ $S_i = x_i \oplus y_i \oplus c_i$
$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$



Given the value of C_0, find the value of equation for C_3 and realise the given circuit

$$C_1 = G_0 + P_0 C_0$$
$$C_2 = G_1 + P_1 C_1$$
$$C_3 = G_2 + P_2 C_2$$
$$C_3 = G_2 + P_2 (G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_3 = G_2 + P_2G_1 + P_2P_1(G_0 + P_0C_0)$$
$$= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$



Implement addition of 2 binary numbers 1010 & 0001 show the calculation of S_0, S_1, S_2, S_3 & also calculate the carry at each stage. Find value of C_4 considering $C_0 = 1$

$$S_0 = P_0 \oplus C_0 = (0 \oplus 1) \oplus 1 = 0$$
$$C_1 = G_0 + P_0C_0 = (0 \cdot 1) + (1 \cdot 1) = 1$$
$$S_1 = P_1 \oplus C_1 = (1 \oplus 0) \oplus 1 = 0$$
$$C_2 = G_1 + P_1C_1 = (1 \cdot 0) + (1 \cdot 1) = 1$$
$$S_2 = P_2 \oplus C_2 = (0 \oplus 0) \oplus 1 = 1$$
$$C_3 = G_2 + P_2C_2 = (0 \cdot 0) + (0 \cdot 1) = 0$$
$$S_3 = P_3 \oplus C_3 = (1 \oplus 0) \oplus 0 = 1$$
$$C_4 = G_3 + P_3C_3 = (1 \cdot 0) + (1 \cdot 0) = 0$$

```
 011
 1010
 0001
    1
S = 1100
```

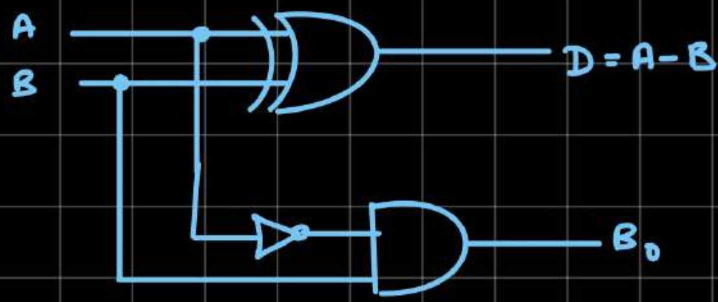
BINARY SUBTRACTOR

HALF SUBTRACTOR:

A	B	D	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

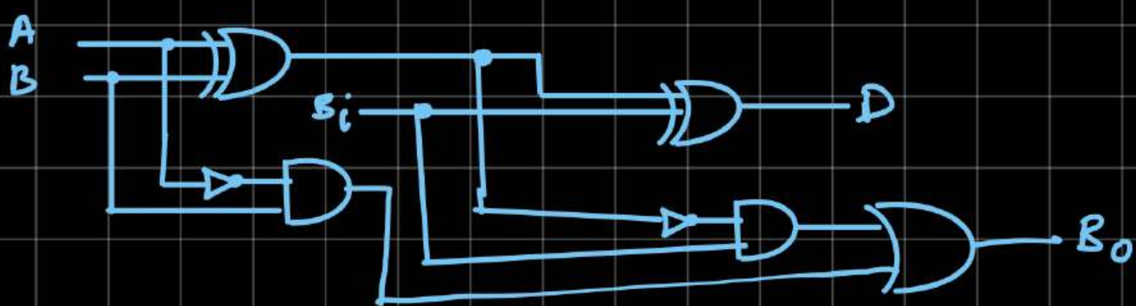
$$D = \bar{A}B + A\bar{B}$$
$$= A \oplus B$$

$$B_0 = \bar{A} \cdot B$$



FULL SUBTRACTOR

Input			output	
A	B	Bi	D	B_0
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



Build a 3-bit subtractor and represent in the form of circuit.

A	B	C	D	B
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

D:

C \ AB	00	01	11	10
0		1		1
1	1		1	

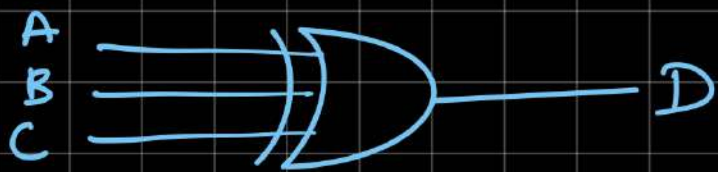
B_0:

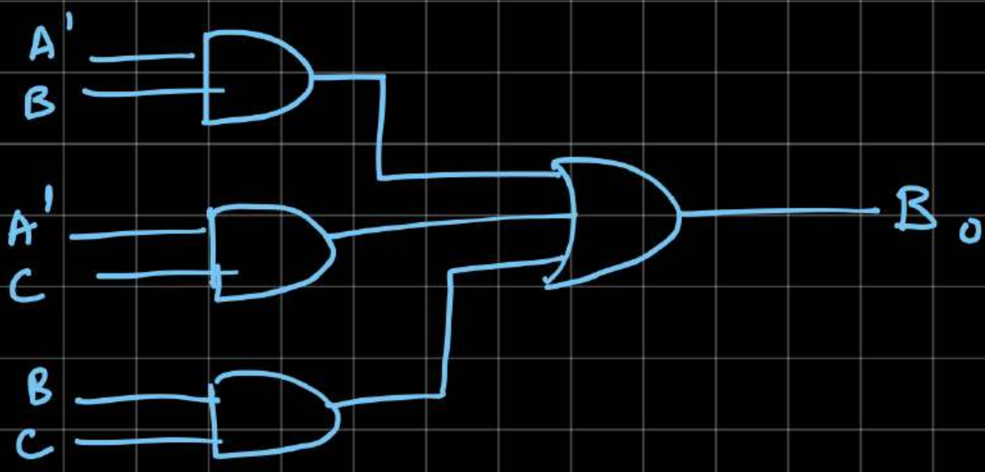
C \ AB	00	01	11	10
0		1		
1	1	1	1	

$$D = A'B'C + A'BC' + ABC + AB'C' = A \oplus B \oplus C$$

$$B_0 = A'B + A'C + BC$$

(Implement circuit)



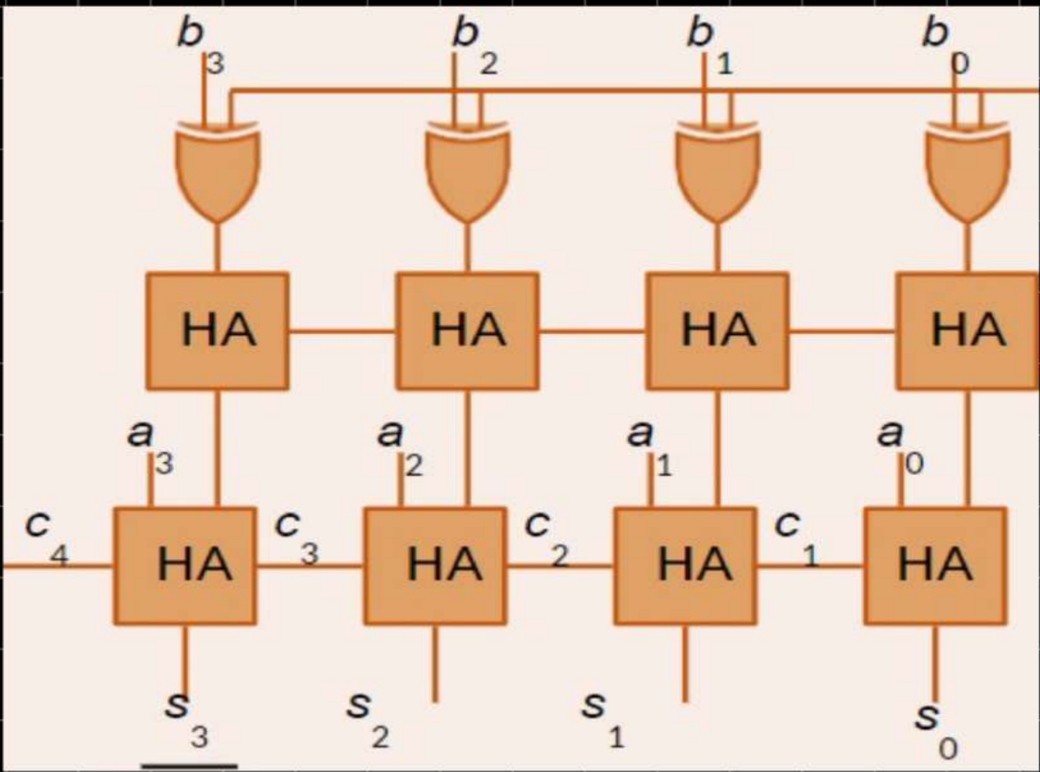


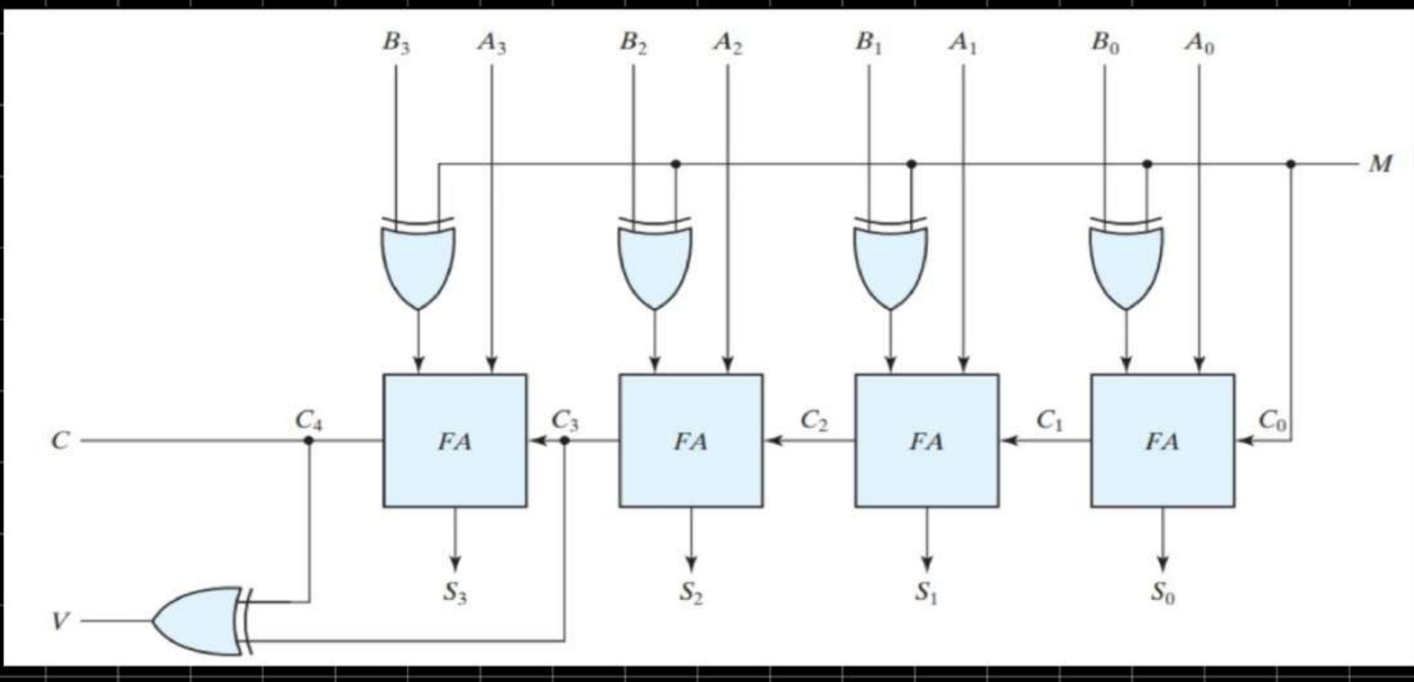
TWO'S COMPLEMENT

A − B = A + 2's complement of B
= A + 1's complement of B + 1

For unsigned, A ≥ B ⇒ A − B
A < B ⇒ 2's complement of (B − A)

For signed, A − B ⟶ no overflow





OVERFLOW CONDITION

opp sign → no OF
same sign → OF

when two numbers with n digits each are added and the sum is a number occupying n+1 digits, overflow is detected

① Represent −2 using 2's complement

(−2) ⇒ 1010 (+2) ⇒ 0010

$$\begin{array}{l|cccc} -2 & 1 & 0 & 1 & 0 \\ 1's & 0 & 1 & 0 & 1 \\ 2's & & & & 1 \\ \hline & 0 & 1 & 1 & 0 \end{array} = 6$$

$$\begin{array}{l|cccc} +2 & 0 & 0 & 1 & 0 \\ 1's & 1 & 1 & 0 & 1 \\ 2's & & & & 1 \\ \hline & 1 & 1 & 1 & 0 \end{array} = -6$$

② Find the decimal value of 2's complement representation of the complement 1001

$$\begin{array}{l|cccc} & 1 & 0 & 0 & 1 \\ 1's & 0 & 1 & 1 & 0 \\ 2's & & & & 1 \\ \hline & 0 & 1 & 1 & 1 \end{array} \Rightarrow 7$$

③ Find 5−3 using 2's complement

To find 2's complement of +3 ⇒

$$\begin{array}{l|cccc} & 0 & 0 & 1 & 1 \\ 1's & 1 & 1 & 0 & 0 \\ 2's & & & & 1 \\ \hline & 1 & 1 & 0 & 1 \end{array}$$

↓

2's complement of 3 replaces −3

$$5-3 \Rightarrow \begin{array}{ccccc} & 0 & 1 & 0 & 1 \\ & 1 & 1 & 0 & 1 \\ \hline \underline{1} & 0 & 0 & 1 & 0 \end{array}$$

0010 ⇒ 2

5 − 3 = 2

④ Subtract (i) −3 & 7 (iii) −7 & 5
(ii) 6 & 3 (iv) −7 & +1

(i) −3 & 7

2's complement of +3 ⇒

$$\begin{array}{lcccc} & 0 & 0 & 1 & 1 \\ \text{1's} & 1 & 1 & 0 & 0 \\ \text{2's} & & & & 1 \\ \hline & 1 & 1 & 0 & 1 \end{array}$$

2's complement of +7 ⇒

$$\begin{array}{lcccc} & 0 & 1 & 1 & 1 \\ \text{1's} & 1 & 0 & 0 & 0 \\ \text{2's} & & & & 1 \\ \hline & 1 & 0 & 0 & 1 \end{array}$$

−3 − 7 ⇒

$$\begin{array}{ccccc} & & & 1 & \\ & 1 & 1 & 0 & 1 \\ & 1 & 0 & 0 & 1 \\ \hline \underline{1} & 0 & 1 & 1 & 0 \end{array}$$

→ NA since out of range

(ii) 6 & 3

2's complement of +3 ⇒

$$\begin{array}{lcccc} & 0 & 0 & 1 & 1 \\ \text{1's} & 1 & 1 & 0 & 0 \\ \text{2's} & & & & 1 \\ \hline & 1 & 1 & 0 & 1 \end{array}$$

6
−3 ⇒

$$\begin{array}{ccccc} & 1 & & & \\ & 0 & 1 & 1 & 0 \\ & 1 & 1 & 0 & 1 \\ \hline \underline{1} & 0 & 0 & 1 & 1 \end{array}$$

⇒ 3

(iii) −7 & 5

2's complement of 5 ⇒

$$\begin{array}{lcccc} & 0 & 1 & 0 & 1 \\ \text{1's} & 1 & 0 & 1 & 0 \\ \text{2's} & & & & 1 \\ \hline & 1 & 0 & 1 & 1 \end{array}$$

2's complement of +7 ⇒

$$\begin{array}{lcccc} & 0 & 1 & 1 & 1 \\ \text{1's} & 1 & 0 & 0 & 0 \\ \text{2's} & & & & 1 \\ \hline & 1 & 0 & 0 & 1 \end{array}$$

−7 − 5 ⇒

$$\begin{array}{ccccc} & 1 & 1 & & \\ & 1 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

⇒ out of range

(iv) −7 & 1

2s complement of 1 ⇒

	0 0 0 1
1's	1 1 1 0
2's	1
	1 1 1 1

2's complement of +7 ⇒

	0 1 1 1
1's	1 0 0 0
2's	1
	1 0 0 1

⇒

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

1 → overflow

1 0 0 0 → 8

DECIMAL ADDER - BCD

Add 6 to convert the number from Binary to BCD (8421) if it exceeds 9

Max allowed : $9+9+1 = 19$ → cannot exceed this sum

45 + 28 ⇒

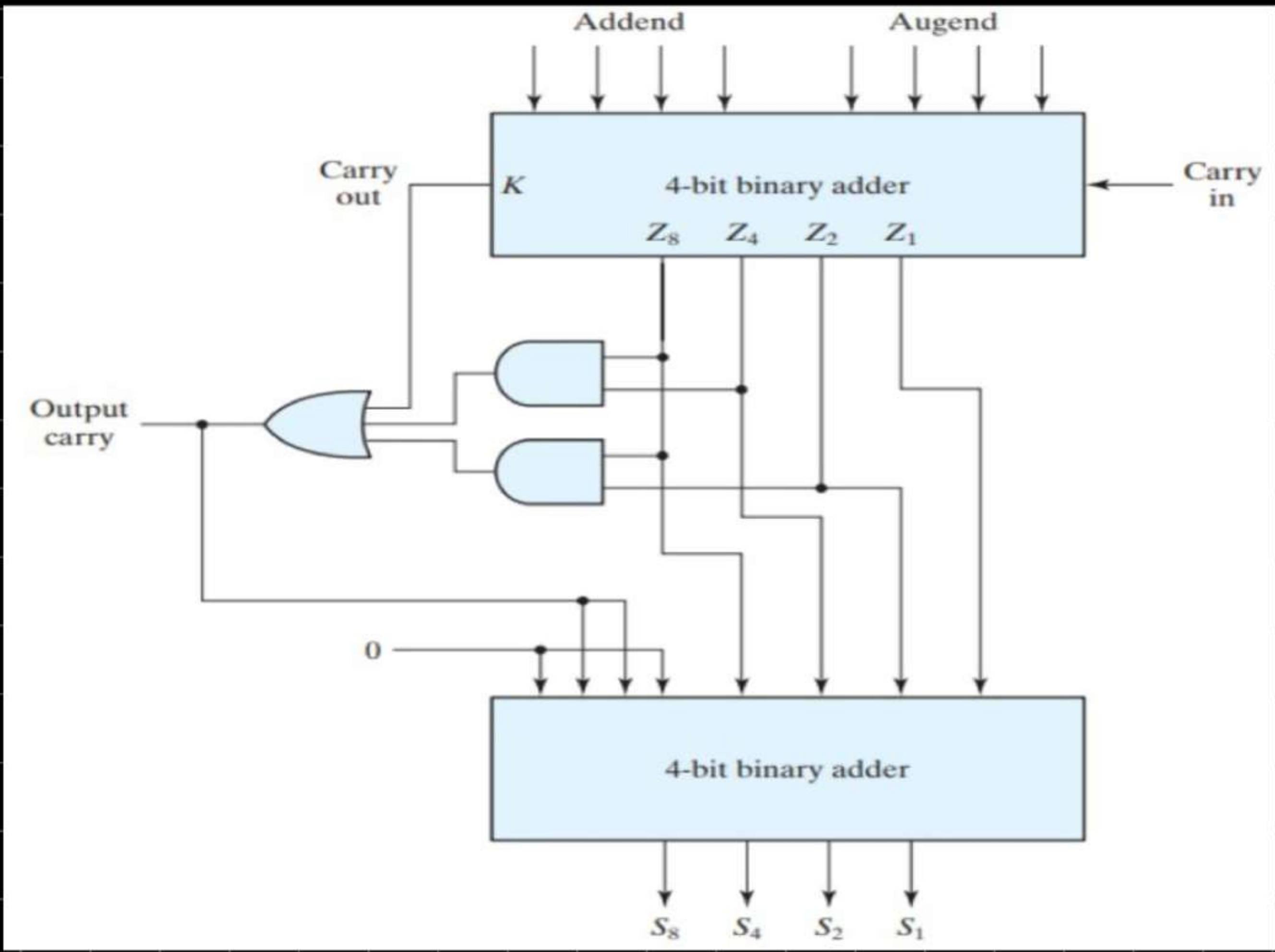
	1	
	0 1 0 0	0 1 0 1
	0 0 1 0	1 0 0 0
	0 1 1 1	1 1 0 1
+6		0 1 1 0
	0 1 1 1	0 0 1 1

→ BCD sum

6 ⇒ 0110

DECIMAL	BINARY SUM					BCD SUM				
	K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

$$C = K + Z_8 Z_4 + Z_8 Z_2$$



789 + 123 ⇒

0 1 1 1	1 0 0 0	1 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
1 0 0 1	1 0 1 1	1 1 0 0
	0 1 1 0	0 0 1 0
	0 0 0 1	1 1 1 0
9	1	2

BINARY MULTIPLIERS

each multiplication ⟶ partial product
sum of partial products ⟶ final product

$J_{bit} \times K_{bit}$

$J \times K$ AND gates

$(J-1)K$ bits adders

$(J+K)$ bits products

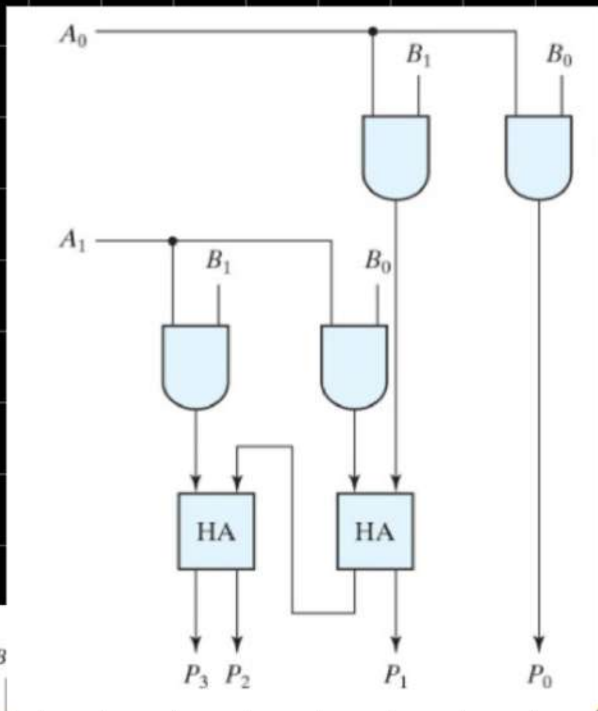
1010 x 1011

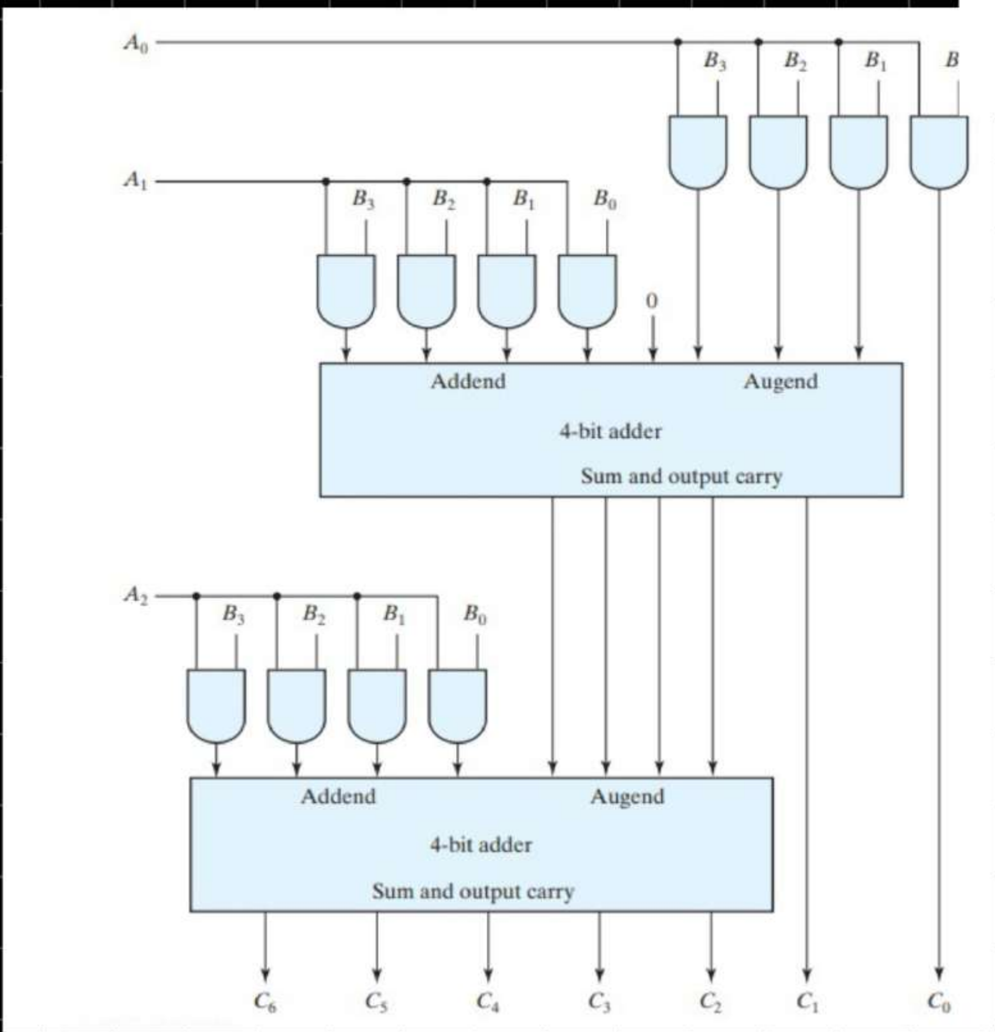
				1	0	1	0
			×	1	0	1	1
				1	0	1	0
			1	0	1	0	
		0	0	0	0		
	1	0	1	0			
	1	1	0	1	1	1	0

→ partial product 1
→ partial product 2
→ partial product 3
→ partial product 4

2 bit × 2 bit:

		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
P_3	P_2	P_1	P_0





4 bit by 3 bit

J = 4
K = 3
product → 7 bit
adders → 2 4bit
and gate → 12

MAGNITUDE COMPARATORS

Combinational circuit that compares two numbers A & B and determine their relative magnitude

A > B , A = B , A < B

Equality of each pair of bits can be expressed with XNOR

$x_i = A_i B_i + A_i' B_i'$ $\quad i = 0, 1, 2, 3$

1 BIT COMPARATOR

A	B	A = B	A > B	A < B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

output = $\bar{A}\bar{B} + AB$

1 hot encoding
→ 1 high

n-bit comparator

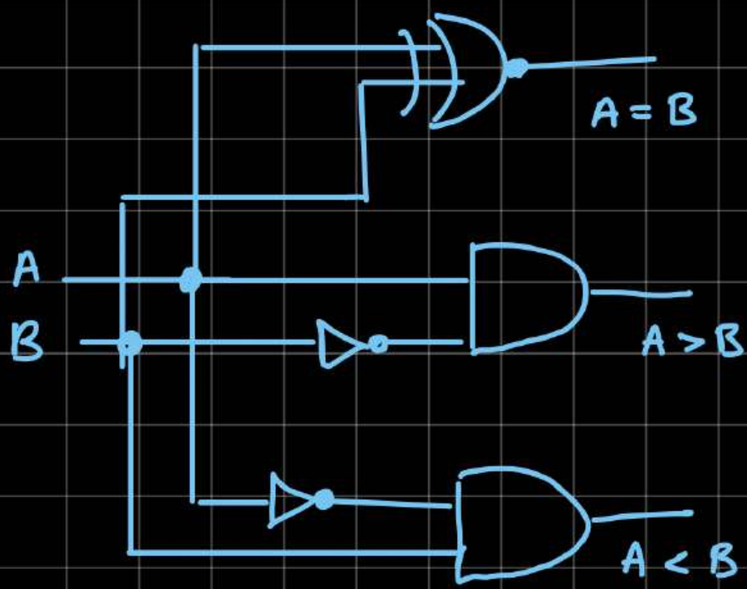
2n variables

2^{2n} rows

$A > B = A\bar{B}$

$A = B = A \odot B$

$A < B = \bar{A}B$



2 BIT COMPARATOR

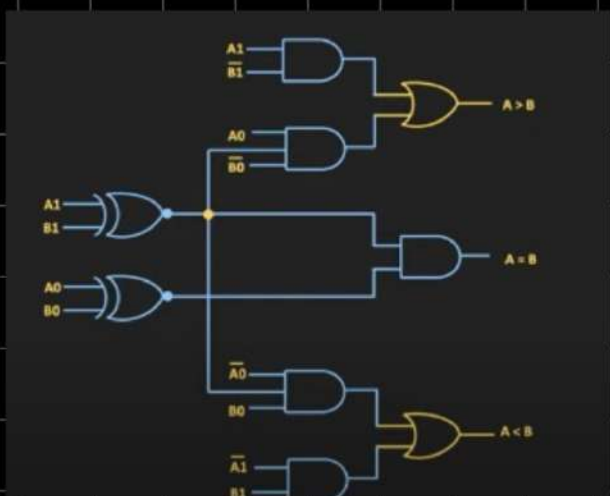
A_1	A_0	B_1	B_0	$A>B$	$A=B$	$A<B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

using K-Maps, we get,

$$A>B = A_1\overline{B_1} + A_0\overline{B_1}\,\overline{B_0} + A_0A_1\overline{B_0}$$
$$= A_1\overline{B_1} + (A_1 \odot B_1)A_0\overline{B_0}$$

$$A=B = (A_1 \odot B_1)(A_0 \odot B_0)$$

$$A<B = \overline{A_1}B_1 + B_0\overline{A_1}\,\overline{A_2} + \overline{A_0}B_0B_1$$
$$= \overline{A_1}B_1 + (A_1 \odot B_1)\overline{A_0}B_0$$



(A pattern can be noticed with the equations if looked at carefully)

4 BIT COMPARATOR

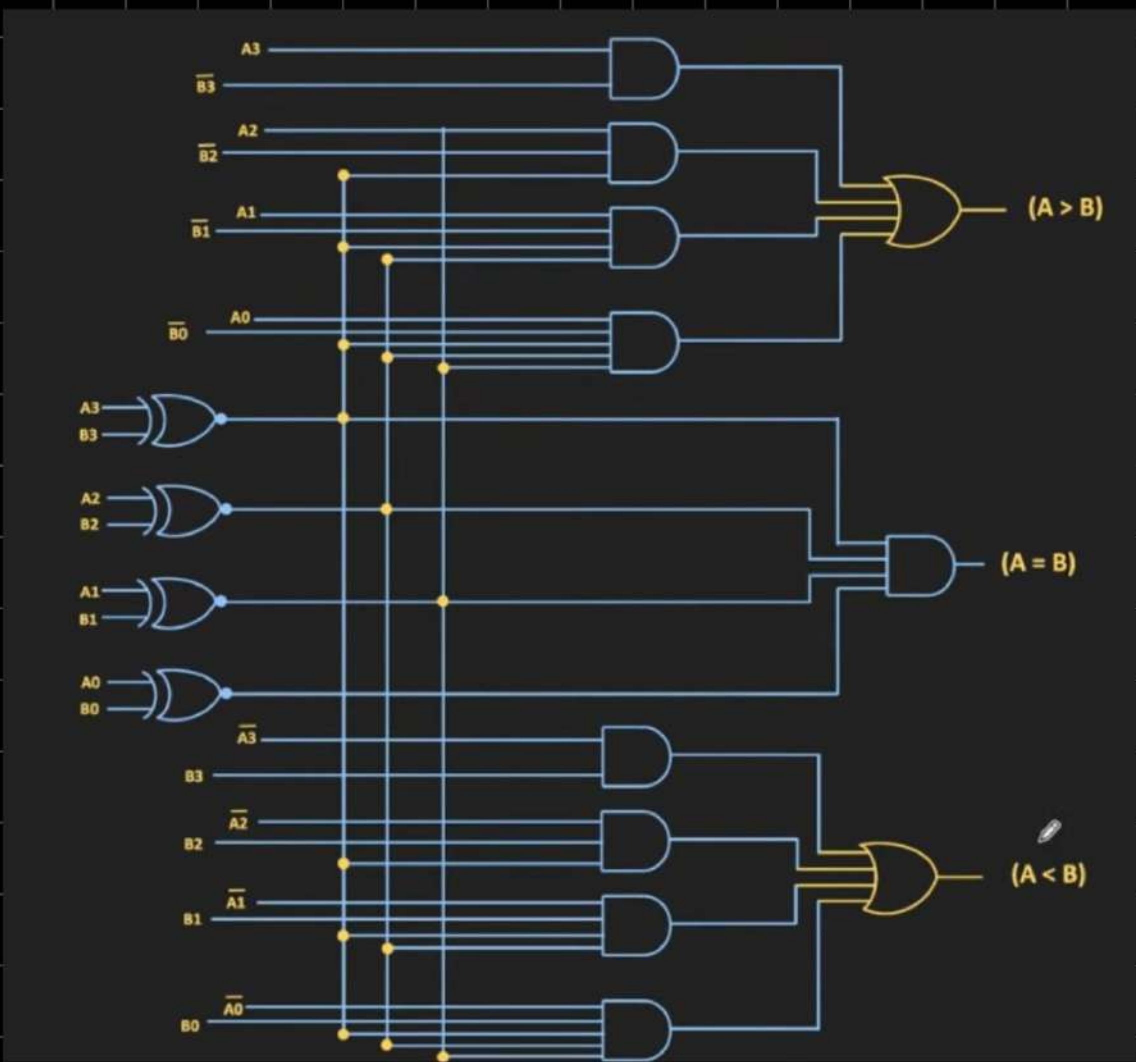
$A = A_3 A_2 A_1 A_0$ & $B = B_3 A_2 B_1 B_0$

$$A = B = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) . (A_0 \odot B_0)$$

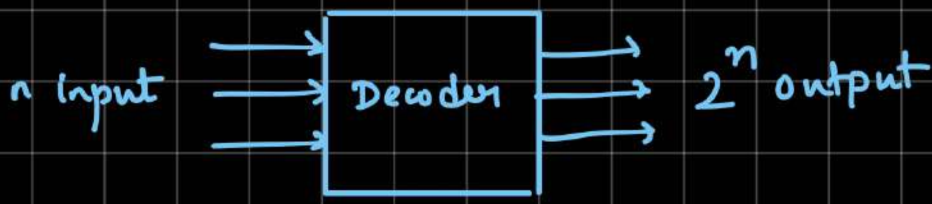
if
1. $A_3 > B_3$
2. $A_3 = B_3$ & $A_2 > B_2$
3. $A_3 = B_3$ & $A_2 = B_2$, $A_1 > B_1$
4. $A_3 = B_3$ & $A_2 = B_2$, $A_1 = B_1$ then $A_0 > B_0$

$$A > B = A_3 \overline{B_3} + (A_3 \odot B_3) A_2 \overline{B_2} + (A_3 \odot B_3)(A_2 \odot B_2) A_1 \overline{B_1} + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) A_0 \overline{B_0}$$

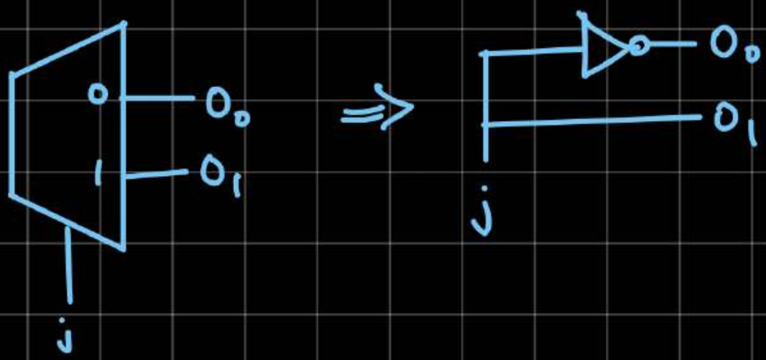
$$A < B = \overline{A_3} B_3 + (A_3 \odot B_3) \overline{A_2} B_2 + (A_3 \odot B_3)(A_2 \odot B_2) \overline{A_1} B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) \overline{A_0} B_0$$



DECODERS



1:2 Decoder



j	O_0	O_1
0	1	0
1	0	1

$O_0 = j'$

$O_1 = j$

2 : 4 Decoder

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

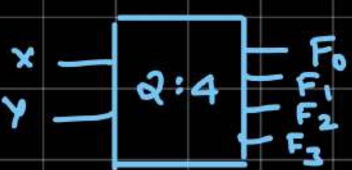
(Fill 1 diagonally)

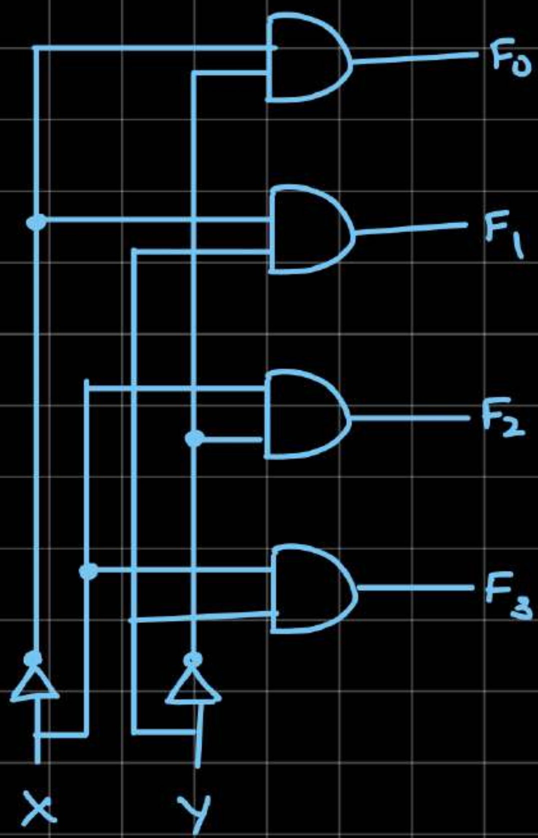
$F_0 = X'Y'$

$F_1 = X'Y$

$F_2 = XY'$

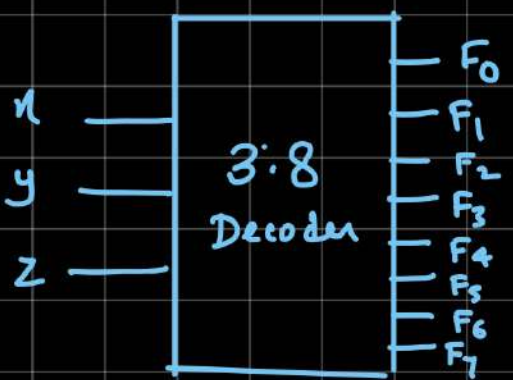
$F_3 = XY$

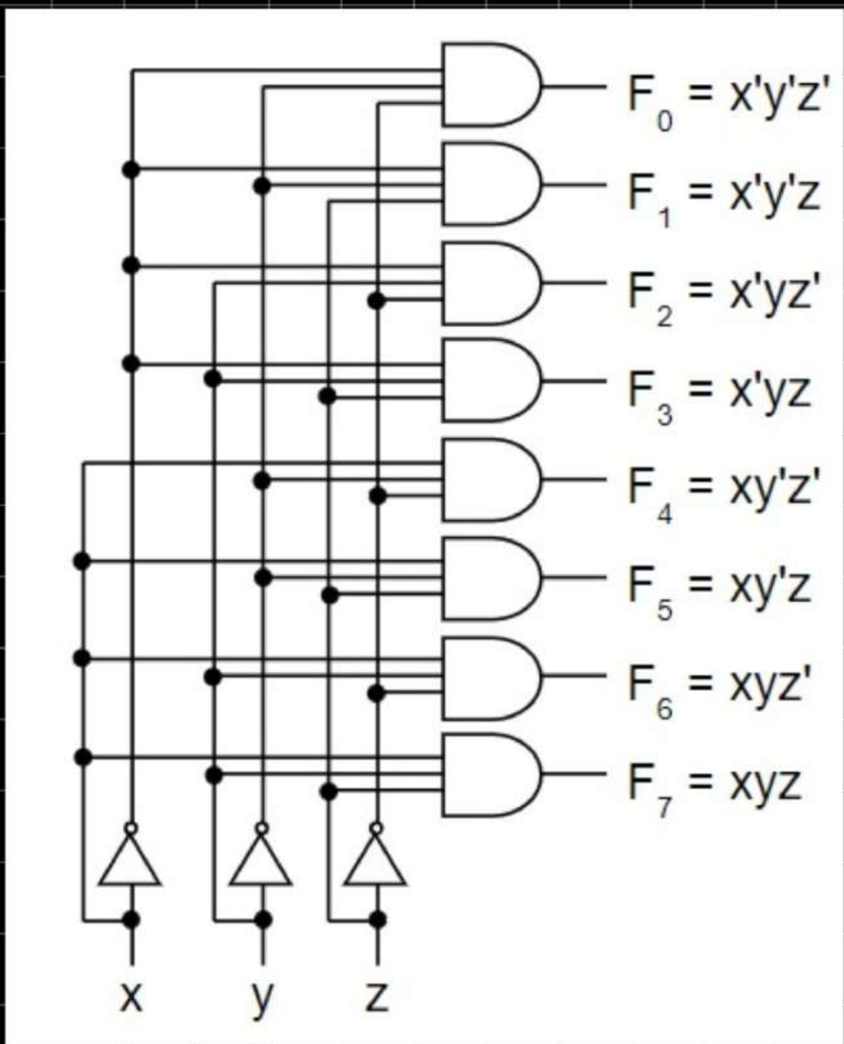




3 : 8 DECODER

X	Y	Z	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_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





$F_0 = x'y'z'$

$F_1 = x'y'z$

$F_2 = x'yz'$

$F_3 = x'yz$

$F_4 = xy'z'$

$F_5 = xy'z$

$F_6 = xyz'$

$F_7 = xyz$

DECODERS WITH NAND GATES

with enable $\Rightarrow$ use of NANDs only when E=0

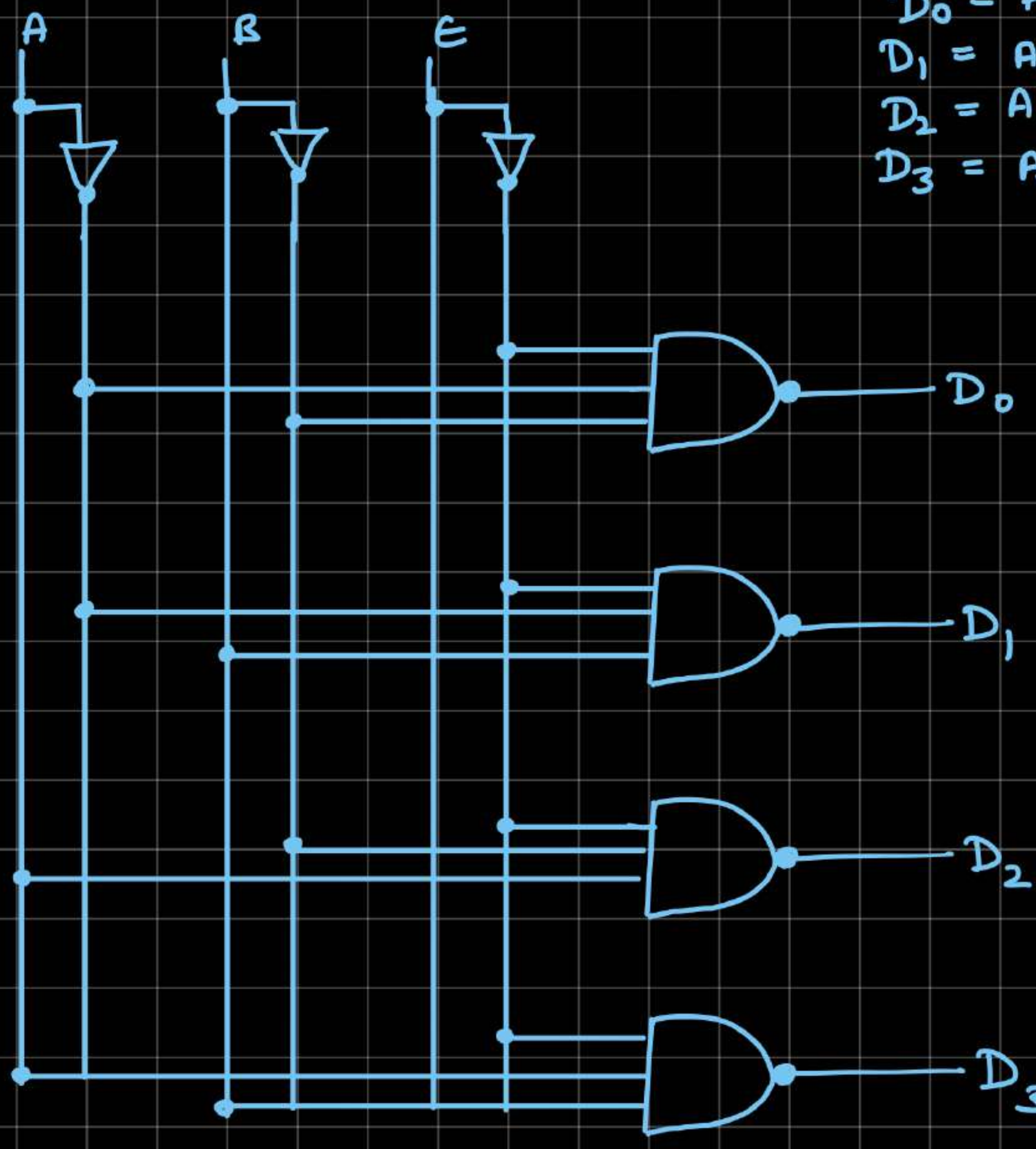
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

$D_0 = A'B'E'$

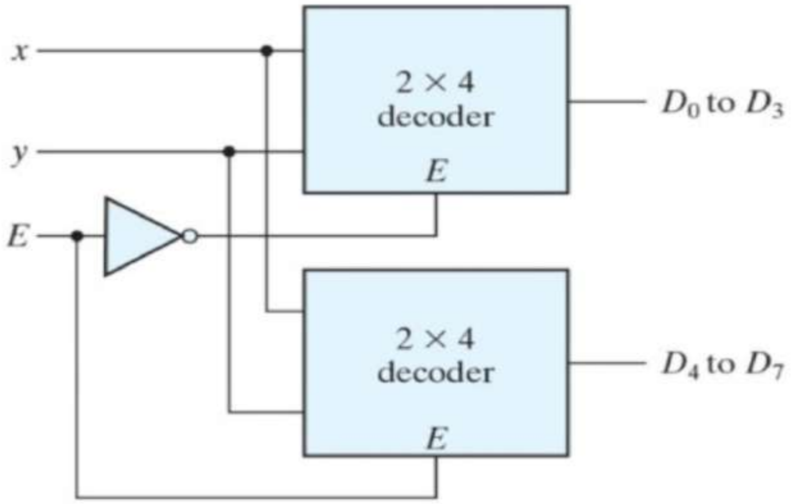
$D_1 = A'BE'$

$D_2 = AB'E'$

$D_3 = ABE'$

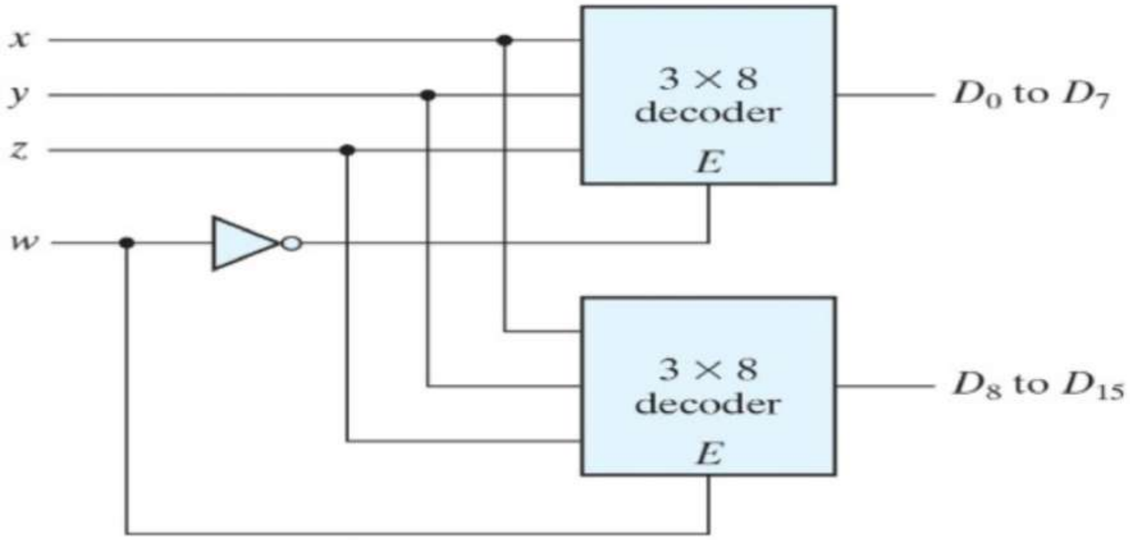


3 x 8 decoder constructed with two 2 x 4 decoders.



E	x	y	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

4 x 16 decoder constructed with two 3 x 8 decoders.



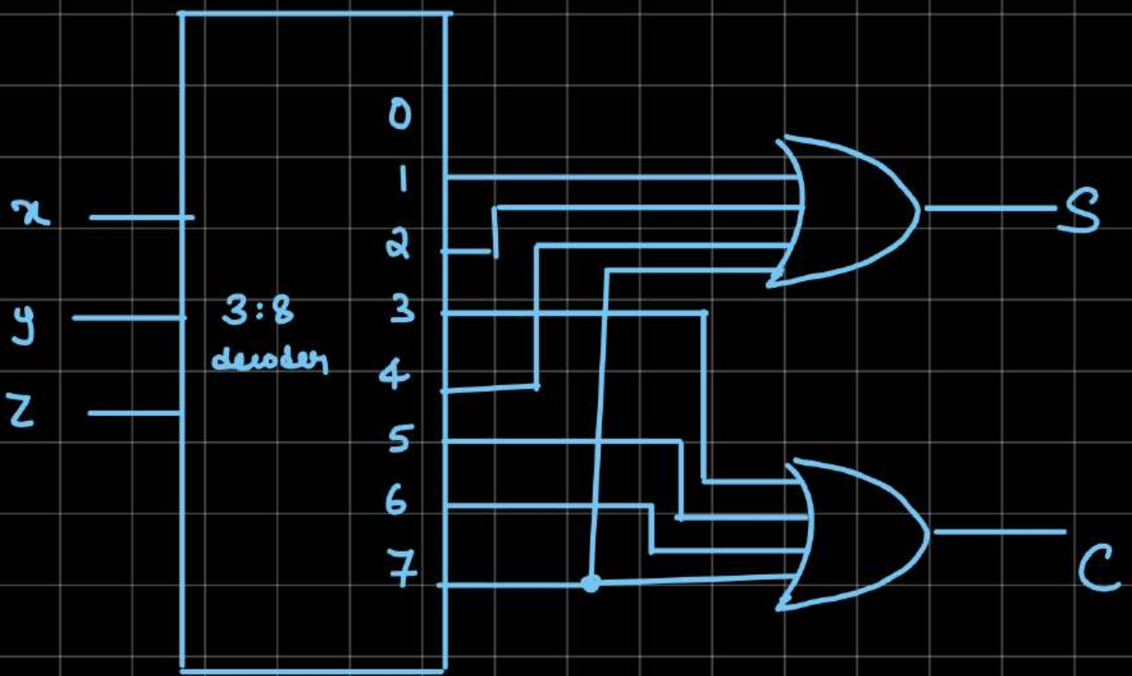
Enable can also be active high
In this example, only one decoder can be active at a time.
x, y, z effectively select output line for w

Implementation of full adder with a 3:8 decoder

X	Y	Z	S	C	
0	0	0	0	0	$\rightarrow F_0$
0	0	1	1	0	$\rightarrow F_1$
0	1	0	1	0	$\rightarrow F_2$
0	1	1	0	1	$\rightarrow F_3$
1	0	0	1	0	$\rightarrow F_4$
1	0	1	0	1	$\rightarrow F_5$
1	1	0	0	1	$\rightarrow F_6$
1	1	1	1	1	$\rightarrow F_7$

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

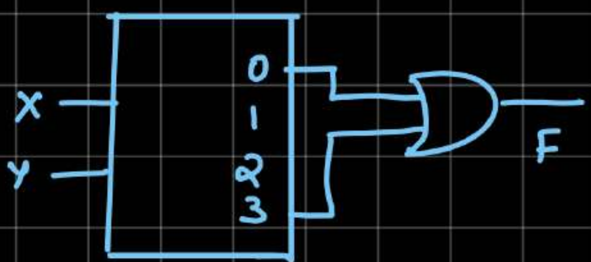
$C = \Sigma(3,5,6,7)$



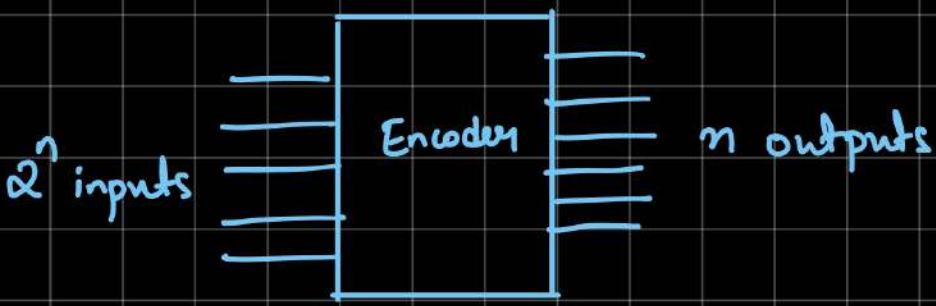
Build XNOR gate using 2:4 decoder

X	Y	$X \odot Y$	
0	0	1	$\rightarrow F_0$
0	1	0	$\rightarrow F_1$
1	0	0	$\rightarrow F_2$
1	1	1	$\rightarrow F_3$

$F = \Sigma(0,3)$



ENCODERS

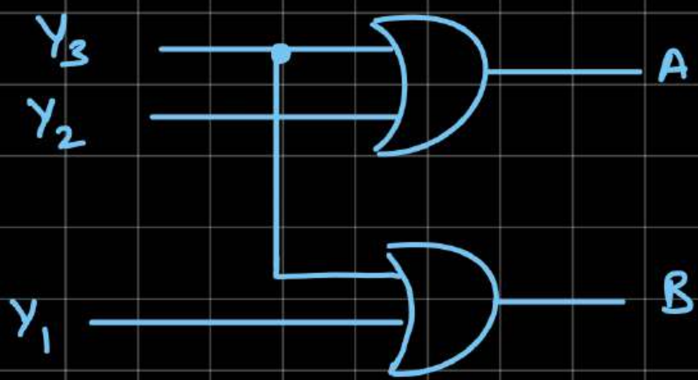


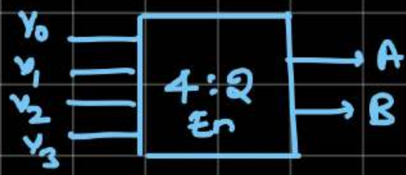
4 : 2 ENCODER

Input				Output	
Y_0	Y_1	Y_2	Y_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

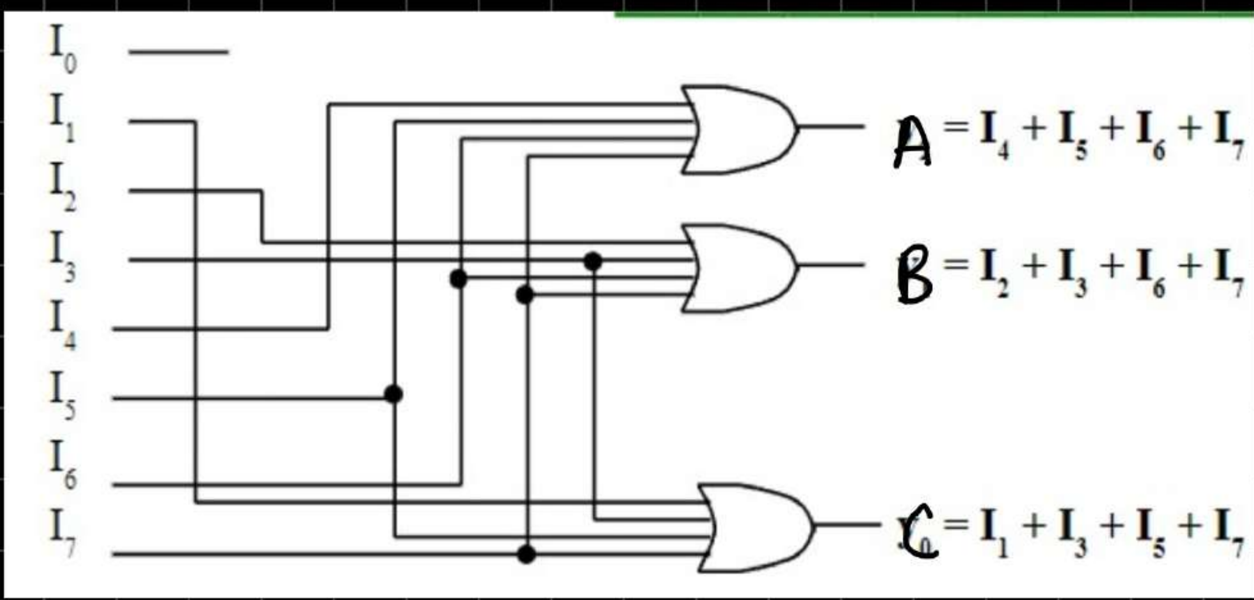
$A = Y_3 + Y_2$

$B = Y_3 + Y_1$





8 : 3 Encoder



8 : 3 ENCODER

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$A = I_4 + I_5 + I_6 + I_7$$
$$B = I_2 + I_3 + I_6 + I_7$$
$$C = I_1 + I_3 + I_5 + I_7$$

PRIORITY ENCODERS

⟶ encoder circuit that includes a priority function

⟶ if 2 or more inputs are equal to 1 at the same time then input having higher priority will take precedence

V ⟶ valid bit indicator ⟶ set to 1 if one or more input is 1

If V = 0, the don't care for output

4 : 2

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

For x :

$D_0D_1 \backslash D_2D_3$	00	01	11	10
00	X	1	1	1
01		1	1	1
11		1	1	1
10		1	1	1

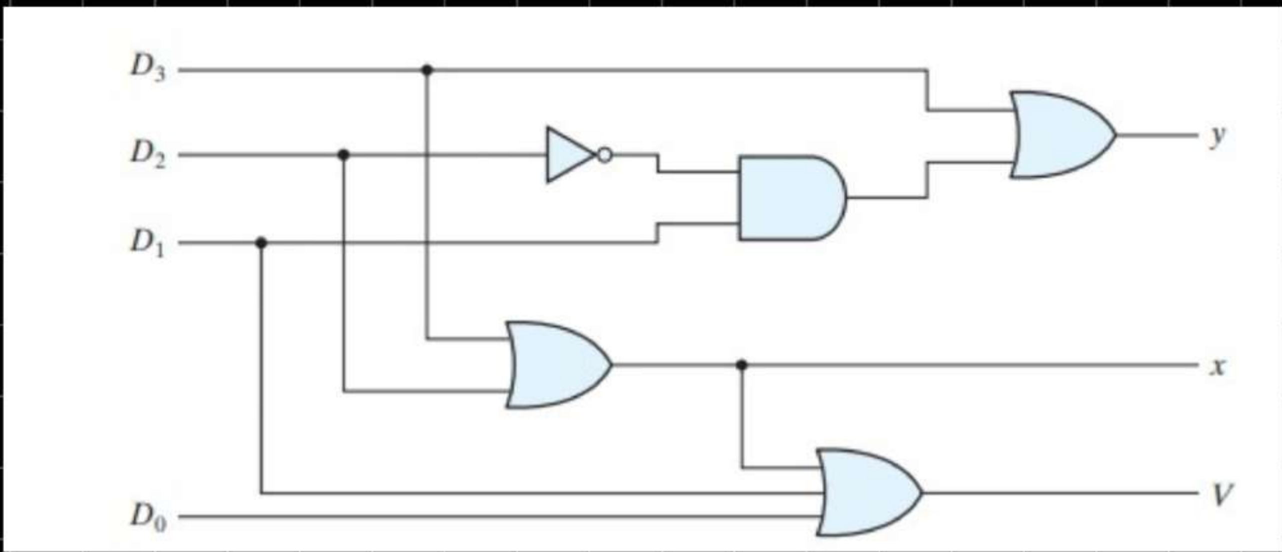
$x = D_3 + D_2$

For y :

$D_0D_1 \backslash D_2D_3$	00	01	11	10
00	X	1	1	
01	1	1	1	
11	1	1	1	
10		1	1	

$y = D_1 D_2' + D_3$

$V = D_0 + D_1 + D_2 + D_3$



8 : 3 ENCODER

Inputs								Outputs			
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	y_2	y_1	y_0	Idle
0	0	0	0	0	0	0	0	x	x	x	1
1	0	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	X	X	1	1	1	1	0

Priority Encoder :

$H7=I7$ (Highest Priority)

$H6=I6.I7'$

$H5=I5.I6'.I7'$

$H4=I4.I5'.I6'.I7'$

$H3=I3.I4'.I5'.I6'.I7'$

$H2=I2.I3'.I4'.I5'.I6'.I7'$

$H1=I1.I2'.I3'.I4'.I5'.I6'.I7'$

$H0=I0.I1'.I2'.I3'.I4'.I5'.I6'.I7'$

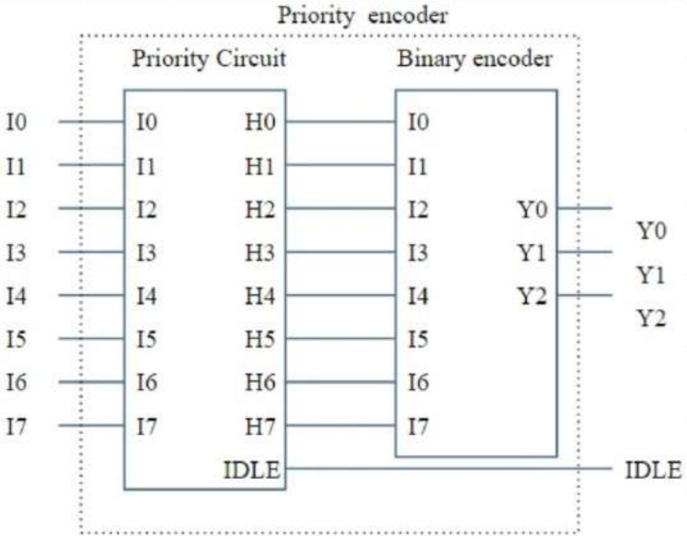
$IDLE=I0'.I1'.I2'.I3'.I4'.I5'.I6'.I7'$

Encoder

$Y0 = I1 + I3 + I5 + I7$

$Y1 = I2 + I3 + I6 + I7$

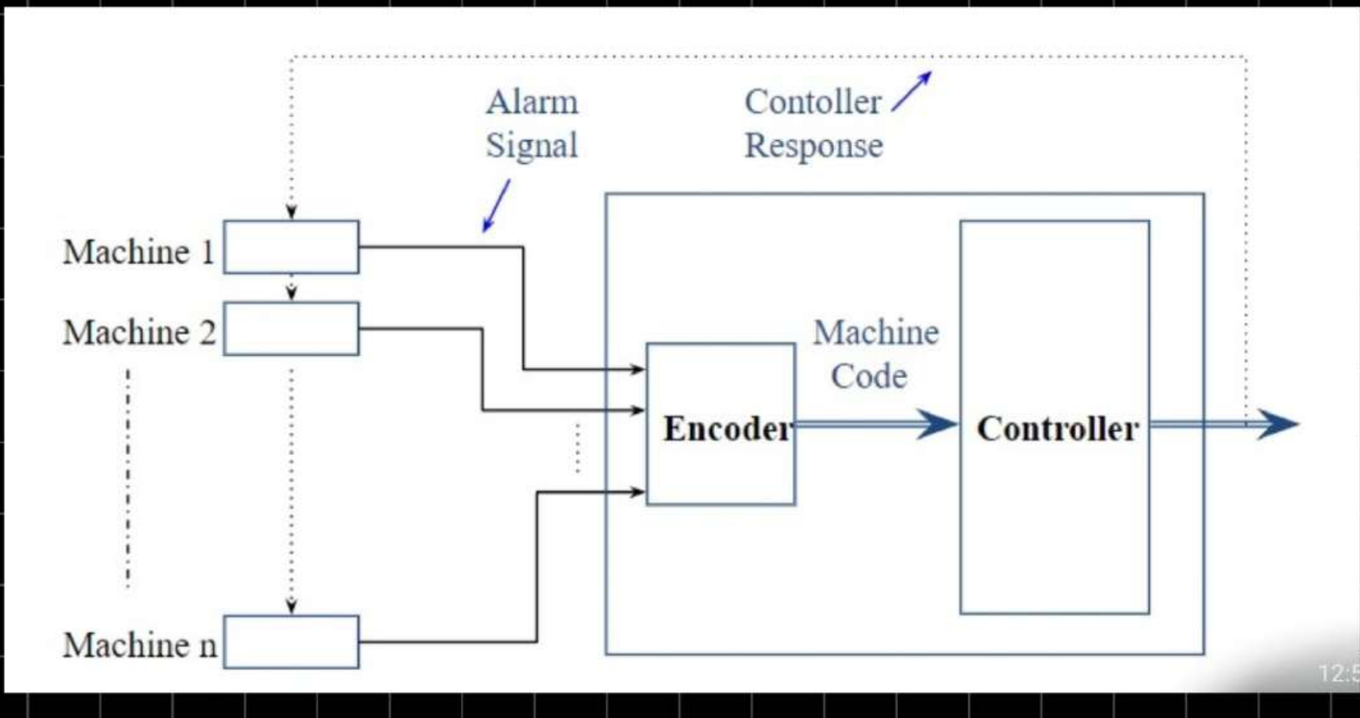
$Y2 = I4 + I5 + I6 + I7$



ENCODER APPLICATION

→ encoder identifies the requester & encodes the value

→ controller accepts digital inputs



MACHINES → different machines connected to the encoder

ENCODER → primary role to identify which machine is sending request & then encoding the value to machine code

also generates alarm signal if needed

CONTROLLER → Receives the encoded machine code from encoder

Based on the machine code, controller performs specific actions such as sending signals back to machine

ALARM SIGNAL → an alert generated by encoder if certain condition is met

→ Malfunction / Process completion / any other situation that needs immediate action

CONTROLLER RESPONSE →

A response sent back to the machine or trigger other actions that are needed

APPLICATION

Encoder → computer (keyboard)

Priority encoder → microprocessor

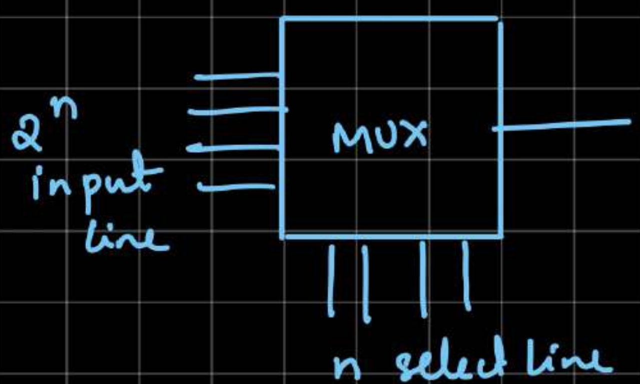
Decoders → memory systems

MULTIPLEXER // MUX

$n:1$ MUX → 2^n input

n select lines

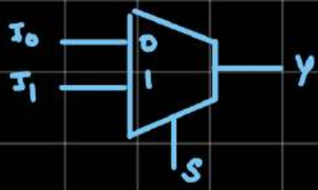
1 output



2:1 MUX

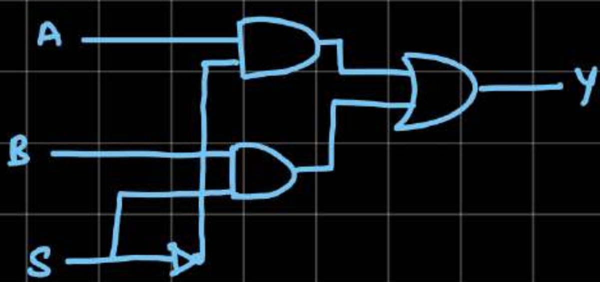
S	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

when $S=0$, $Y=A$
$S=1$, $Y=B$



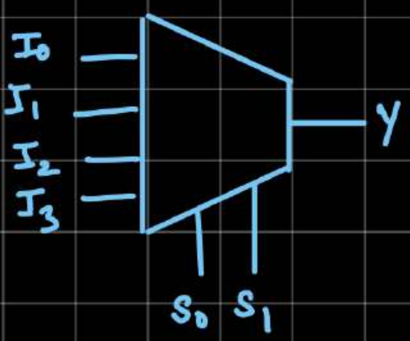
S \ AB	00	01	11	10
0			1	1
1		1	1	

$$Y = S'A + SB$$

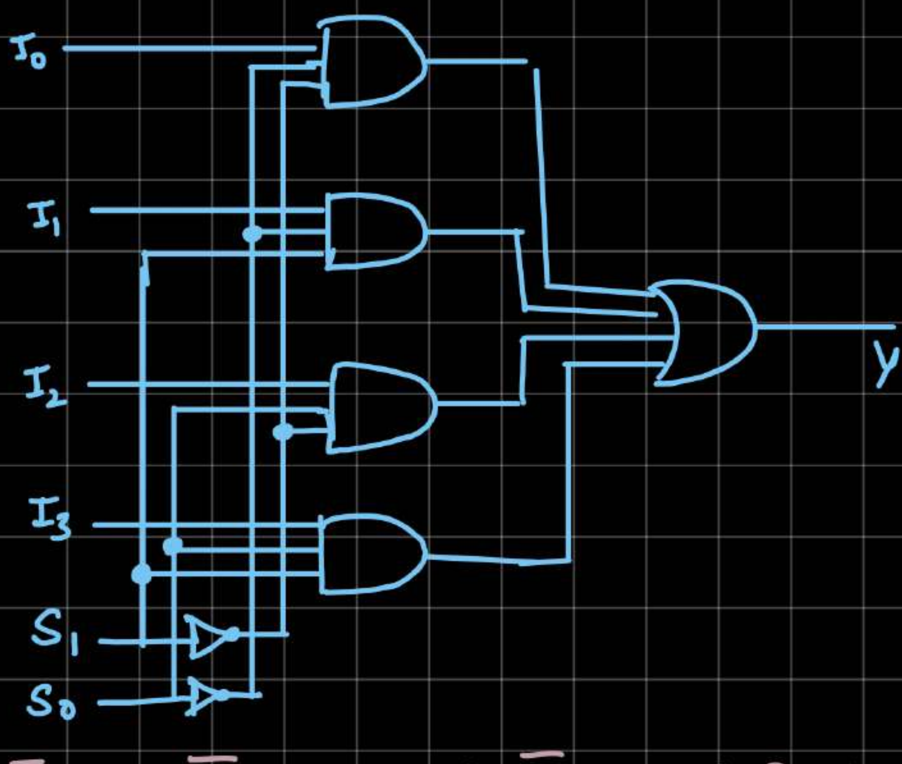


4:1 MUX

Implement a 4:1 MUX, realise the circuit & TT by considering the select lines

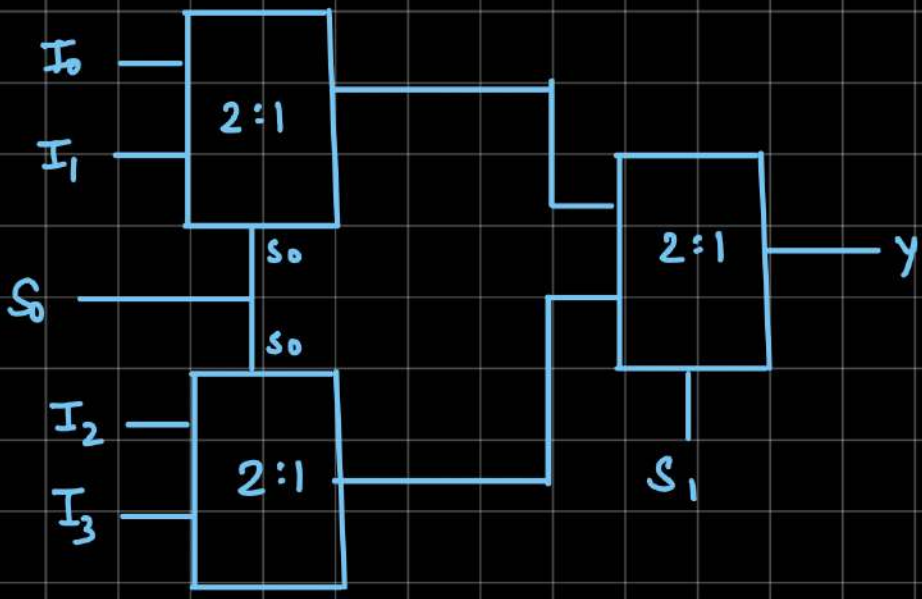


S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



$$Y = \bar{S_0}\bar{S_1}I_0 + \bar{S_0}S_1I_1 + S_0\bar{S_1}I_2 + S_0S_1I_3$$

Implement 4:1 MUX using 2:1 MUX



S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

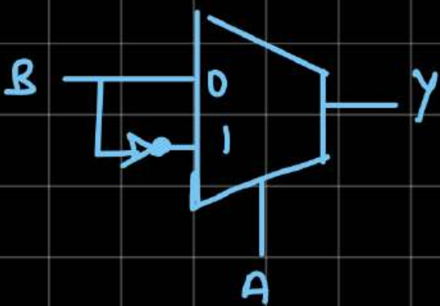
Design an XOR gate using 2:1 MUX

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Let A be select line

when $A=0$, $Y=B$

$A=1$, $Y=\overline{B}$



Implement the boolean function $F = \Sigma(1,2,6,7)$ using a MUX

A	B	C	Y
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

when

$A=0, B=0$

$Y=C$

$A=0, B=1$

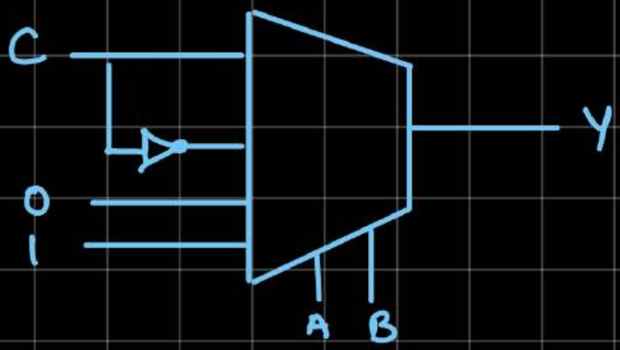
$Y=\overline{C}$

$A=1, B=0$

$Y=0$

$A=1, B=1$

$Y=1$



* follow the order when drawing MUX

Using 4:1 MUX design the following function

$$Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$$

$$Y = A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}BC$$

$$= \Sigma(0, 3, 4, 5)$$

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

when

$A = 0, B = 0$

$Y = \bar{C}$

$A = 0, B = 1$

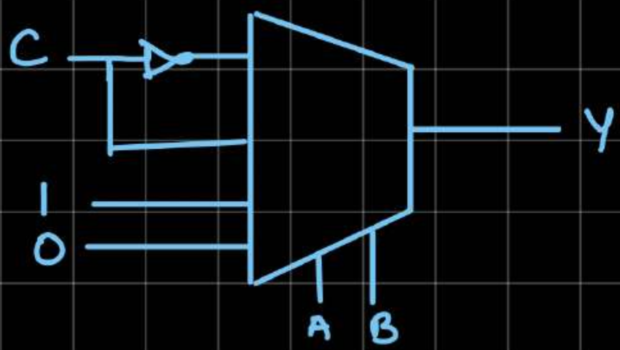
$Y = C$

$A = 1, B = 0$

$Y = 1$

$A = 1, B = 1$

$Y = 0$



Implement the boolean fn $F(A,B,C,D) = \Sigma(1,3,4,11,12,13,14,15)$ using a 8:1 MUX

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

when

$A=0, B=0, C=0$
$Y=D$

$A=0, B=0, C=1$
$Y=D$

$A=0, B=1, C=0$
$Y=\overline{D}$

$A=0, B=1, C=1$
$Y=0$

$A=1, B=0, C=0$
$Y=0$

$A=1, B=0, C=1$
$Y=D$

$A=1, B=1, C=0$
$Y=1$

$A=1, B=1, C=1$
$Y=1$