

# Basic Electronics Engineering (Spring 2024)

Resources of PPT:

- ❑ [www.google.com](http://www.google.com)
- ❑ Digital Design, 4<sup>th</sup> Edition  
M. Morris Mano and Michael D. Ciletti

# Syllabus



## Suggested Reading:

- **Mano and Ciletti, “Digital Design”, Pearson**
- **Sedra and Smith, “Microelectronic Circuits”, Oxford University Press**

# Binary Logic



**AND:**  $x \cdot y = z$  or  $xy = z$  is read “x AND y is equal to z.”

AND		
$x$	$y$	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

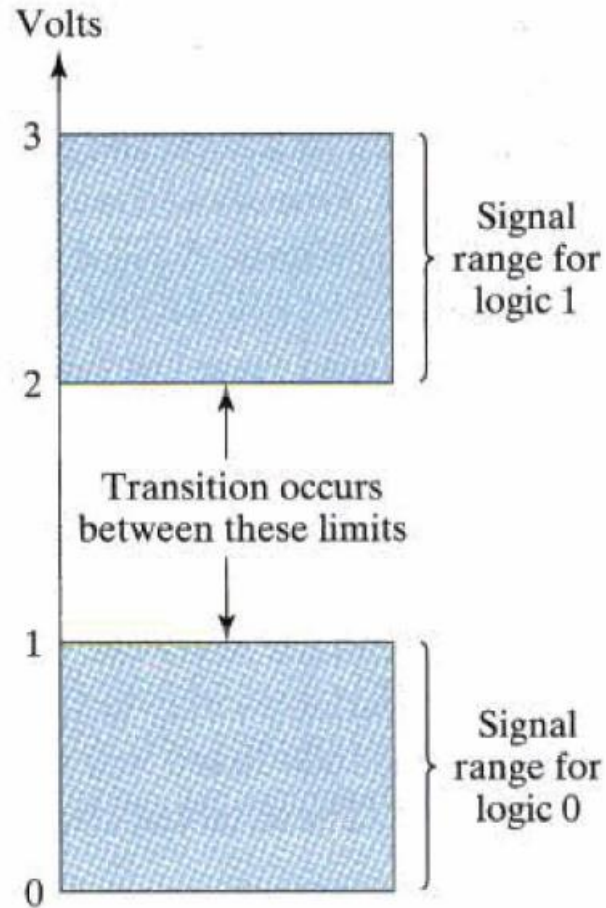
**OR:**  $x + y = z$  ; is read as “x OR y” is equal to z.

OR		
$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

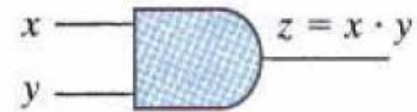
**NOT:**  $x' = z$  (or  $\bar{x} = z$ ) is read “not x is equal to z.”

NOT	
$x$	$x'$
0	1
1	0

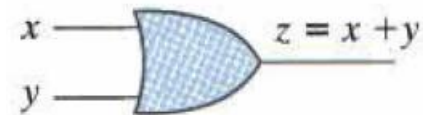
# Logic Gates



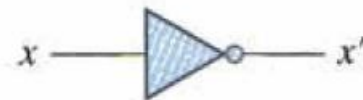
binary signals



(a) Two-input AND gate

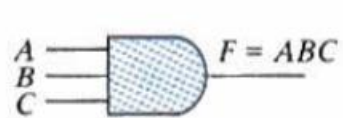
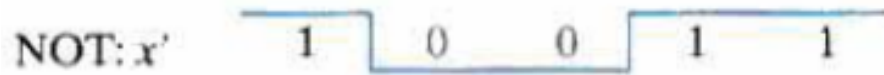
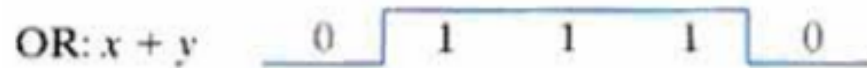


(b) Two-input OR gate

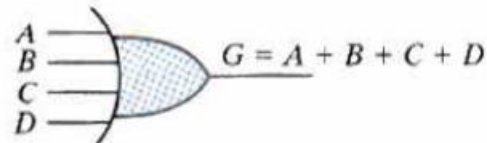


(c) NOT gate or inverter

# Logic Gates



(a) Three-input AND gate



(b) Four-input OR gate

*Closure.* A set  $S$  is closed with respect to a binary operator if, for every pair of elements of  $S$ , the binary operator specifies a rule for obtaining a unique element of  $S$ . For example, the set of natural numbers  $N = \{1, 2, 3, 4, \dots\}$  is closed with respect to the binary operator  $+$  by the rules of arithmetic addition, since, for any  $a, b \in N$ , there is a unique  $c \in N$  such that  $a + b = c$ . The set of natural numbers is *not* closed with respect to the binary operator  $-$  by the rules of arithmetic subtraction, because  $2 - 3 = -1$  and  $2, 3 \in N$ , but  $(-1) \notin N$ .

**Any Example of Non-closure operator?**

*Associative law.* A binary operator  $*$  on a set  $S$  is said to be associative whenever

$$(x * y) * z = x * (y * z) \text{ for all } x, y, z, \in S$$

*Commutative law.* A binary operator  $*$  on a set  $S$  is said to be commutative whenever

$$x * y = y * x \text{ for all } x, y \in S$$

# Boolean Algebra



*Identity element.* A set  $S$  is said to have an identity element with respect to a binary operation  $*$  on  $S$  if there exists an element  $e \in S$  with the property that

$$e * x = x * e = x \text{ for every } x \in S$$

*Example:* The element 0 is an identity element with respect to the binary operator  $+$  on the set of integers  $I = \{ \dots, -3, -2, -1, 0, 1, 2, 3, \dots \}$ , since

$$x + 0 = 0 + x = x \text{ for any } x \in I$$

The set of natural numbers,  $N$ , has no identity element, since 0 is excluded from the set.

*Inverse.* A set  $S$  having the identity element  $e$  with respect to a binary operator  $*$  is said to have an inverse whenever, for every  $x \in S$ , there exists an element  $y \in S$  such that

$$x * y = e$$

*Example:* In the set of integers,  $I$ , and the operator  $+$ , with  $e = 0$ , the inverse of an element  $a$  is  $(-a)$ , since  $a + (-a) = 0$ .

*Distributive law.* If  $*$  and  $\cdot$  are two binary operators on a set  $S$ ,  $*$  is said to be distributive over  $\cdot$  whenever

$$x * (y \cdot z) = (x * y) \cdot (x * z)$$

# Boolean Algebra



Boolean algebra is an algebraic structure defined by a set of elements,  $B$ , together with two binary operators  $+$  and  $\cdot$ .

1. (a) The structure is closed with respect to the operator  $+$ .  
(b) The structure is closed with respect to the operator  $\cdot$ .
2. (a) The element  $0$  is an identity element with respect to  $+$ ; that is,  $x + 0 = 0 + x = x$ .  
(b) The element  $1$  is an identity element with respect to  $\cdot$ ; that is,  $x \cdot 1 = 1 \cdot x = x$ .
3. (a) The structure is commutative with respect to  $+$ ; that is,  $x + y = y + x$ .  
(b) The structure is commutative with respect to  $\cdot$ ; that is,  $x \cdot y = y \cdot x$ .
4. (a) The operator  $\cdot$  is distributive over  $+$ ; that is,  $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ .  
(b) The operator  $+$  is distributive over  $\cdot$ ; that is,  $x + (y \cdot z) = (x + y) \cdot (x + z)$ .
5. For every element  $x \in B$ , there exists an element  $x' \in B$  (called the *complement* of  $x$ ) such that (a)  $x + x' = 1$  and (b)  $x \cdot x' = 0$ .
6. There exist at least two elements  $x, y \in B$  such that  $x \neq y$ .



# Boolean Algebra vs. Ordinary Algebra



1. Huntington postulates do not include the associative law. However, this law holds for Boolean algebra and can be derived (for both operators) from the other postulates.
2. The distributive law of  $+$  over  $\cdot$  (i.e.,  $x + (y \cdot z) = (x + y) \cdot (x + z)$ ), is valid for Boolean algebra, but not for ordinary algebra.
3. Boolean algebra does not have additive or multiplicative inverses; therefore, there are no subtraction or division operations.
4. Postulate 5 defines an operator called the *complement* that is not available in ordinary algebra.
5. Ordinary algebra deals with the real numbers, which constitute an infinite set of elements. Boolean algebra deals with the as yet undefined set of elements,  $B$ , but in the two-valued Boolean algebra defined next (and of interest in our subsequent use of that algebra),  $B$  is defined as a set with only two elements, 0 and 1.

# Boolean Algebra



$x$	$y$	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

$x$	$x'$
0	1
1	0

# Boolean Algebra



$x$	$y$	$z$
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$y + z$	$x \cdot (y + z)$
0	0
1	0
1	0
1	0
0	0
1	1
1	1
1	1

$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
1	0	1
1	1	1

# Boolean Algebra



## Postulates and Theorems of Boolean Algebra

Postulate 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulate 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Theorem 1	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Postulate 3, commutative	(a) $x + y = y + x$	(b) $xy = yx$
Theorem 4, associative	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
Postulate 4, distributive	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
Theorem 6, absorption	(a) $x + xy = x$	(b) $x(x + y) = x$

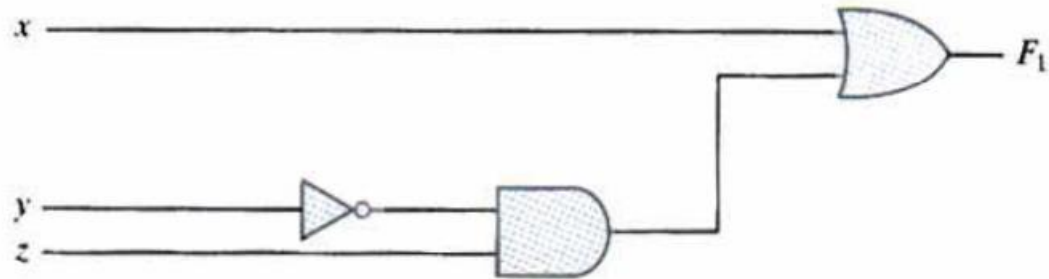
## How to Prove DeMorgan's Theorem? $(x + y)' = x'y'$

$x$	$y$	$x + y$	$(x + y)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$x'$	$y'$	$x'y'$
1	1	1
1	0	0
0	1	0
0	0	0

# Boolean Function

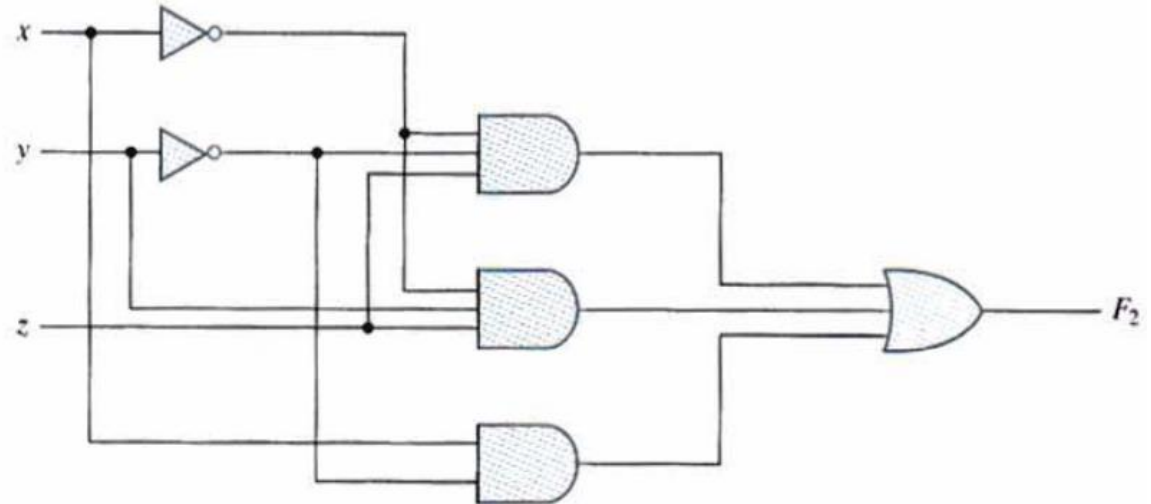
$$F_1 = x + y'z$$



<b><i>x</i></b>	<b><i>y</i></b>	<b><i>z</i></b>	<b><i>F</i><sub>1</sub></b>
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

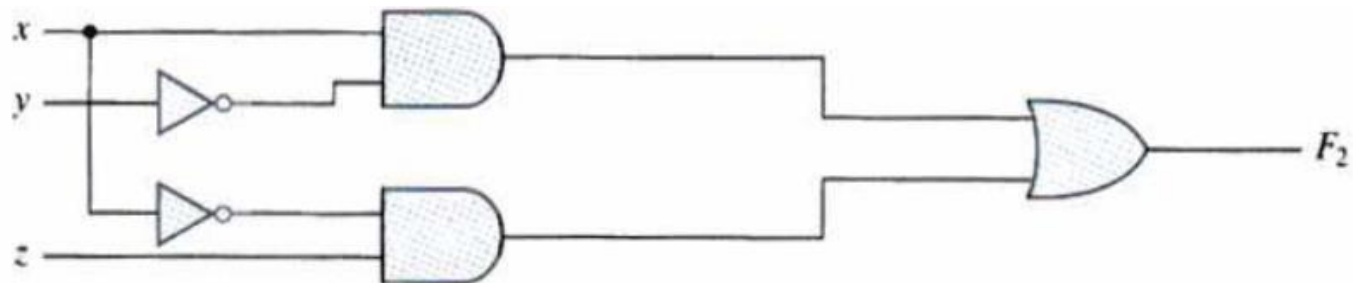
# Boolean Function

$x$	$y$	$z$	$F_2$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



$$F_2 = x'y'z + x'yz + xy'$$

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$



# Boolean Function Simplification



$$(a) \quad x + 0 = x$$

$$(a) \quad x + x' = 1$$

$$(a) \quad x + x = x$$

$$(a) \quad x + 1 = 1$$

$$(x')' = x$$

$$(a) \quad x + y = y + x$$

$$(a) \quad x + (y + z) = (x + y) + z$$

$$(a) \quad x(y + z) = xy + xz$$

$$(a) \quad (x + y)' = x'y'$$

$$(a) \quad x + xy = x$$

$$(b) \quad x \cdot 1 = x$$

$$(b) \quad x \cdot x' = 0$$

$$(b) \quad x \cdot x = x$$

$$(b) \quad x \cdot 0 = 0$$

$$(b) \quad xy = yx$$

$$(b) \quad x(yz) = (xy)z$$

$$(b) \quad x + yz = (x + y)(x + z)$$

$$(b) \quad (xy)' = x' + y'$$

$$(b) \quad x(x + y) = x$$

$$(x + y)(x + y') = x + xy + xy' + yy'$$

$$= x(1 + y + y')$$

$$= x.$$

# Boolean Function Simplification



$$xy + x'z + yz = xy + x'z + yz(x + x')$$

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

$$= xy(1 + z) + x'z(1 + y)$$

$$= xy + x'z.$$



# Minterms and Sum of Product (SoP)



Minterms									
<i>x</i>	<i>y</i>	<i>z</i>	Term	Designation					
0	0	0	$x'y'z'$	$m_0$					
0	0	1	$x'y'z$	$m_1$					
0	1	0	$x'yz'$	$m_2$					
0	1	1	$x'yz$	$m_3$					
1	0	0	$xy'z'$	$m_4$					
1	0	1	$xy'z$	$m_5$					
1	1	0	$xyz'$	$m_6$					
1	1	1	$xyz$	$m_7$					

<i>x</i>	<i>y</i>	<i>z</i>	Function $f_1$	Function $f_2$
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

Express  $f_1$  and  $f_2$  as a sum of product (SOP)/ minterms:

A Boolean function can be expressed algebraically from a truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

# Sum of Product (SoP)



Express the Boolean function  $F = A + B'C$  as a sum of minterms.

- Identify which variables are missing in each term.
- If the missing variable is  $x$ , multiply the corresponding term with  $(x + x')$

$$\begin{aligned} F &= A + B'C \\ &= A(B + B')(C + C') + B'C(A + A') \\ &= A(BC + BC' + B'C + B'C') + AB'C + A'B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \\ &= A'B'C + AB'C' + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

Example. Express the Boolean function  $F = x + yz$  as a sum of minterms.

Solution:

$$\begin{aligned} F &= x + yz = x + (yz) && \text{AND (multiply) has a higher precedence than OR (add)} \\ &= x(y+y')(z+z') + (x+x')yz && \text{expand 1}^{\text{st}} \text{ term by ANDing it with } (y+y')(z+z'), \text{ and 2}^{\text{nd}} \text{ term with } (x+x') \\ &= xyz + xyz' + xy'z + xy'z' + x'yz + x'yz && \\ &= m_7 + m_6 + m_5 + m_4 + m_3 \\ &= \Sigma(3, 4, 5, 6, 7) && \text{sum of 1-minterms} \end{aligned}$$

# Maxterms and Product of Sum (PoS)



## Maxterms

$x$	$y$	$z$	Term	Designation	$x$	$y$	$z$	Function $f_1$	Function $f_2$
0	0	0	$x + y + z$	$M_0$	0	0	0	0	0
0	0	1	$x + y + z'$	$M_1$	0	0	1	1	0
0	1	0	$x + y' + z$	$M_2$	0	1	0	0	0
0	1	1	$x + y' + z'$	$M_3$	0	1	1	0	1
1	0	0	$x' + y + z$	$M_4$	1	0	0	1	0
1	0	1	$x' + y + z'$	$M_5$	1	0	1	0	1
1	1	0	$x' + y' + z$	$M_6$	1	1	0	0	1
1	1	1	$x' + y' + z'$	$M_7$	1	1	1	1	1

Express  $f_1$  and  $f_2$  as a product of sum (POS)/ maxterms:

Consider the maxterm for each combination of the variables that produces a 0 in the function and then taking the AND of all those terms.

$$\begin{aligned}
 f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\
 &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6
 \end{aligned}$$

$$\begin{aligned}
 f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\
 &= M_0 M_1 M_2 M_4
 \end{aligned}$$

# Product of Sum (PoS)



- Use the distributive law:  $x + yz = (x + y)(x + z)$ .
- Use  $x + x' = 1$
- Use  $xx' = 0$ .

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

The function has three variables:  $x$ ,  $y$ , and  $z$ . Each OR term is missing one variable; therefore,

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

Combining all the terms and removing those which appear more than once, we finally obtain

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0 M_2 M_4 M_5 \\ &= \Pi(0, 2, 4, 5) \end{aligned}$$

# Product of Sum (PoS)



Example. Express the Boolean function  $F = x + yz$  as a product of maxterms.

Solution: First, we need to convert the function into the product-of-OR terms by using the distributive law as follows:

$$F = x + yz = x + (yz)$$

$$= (x + y)(x + z)$$

$$= (x + y + zz')(x + yy' + z)$$

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

$$= M_0 \bullet M_1 \bullet M_2$$

$$= \Pi(0, 1, 2)$$

AND (multiply) has a higher precedence than OR (add)

use distributive law to change to product of OR terms

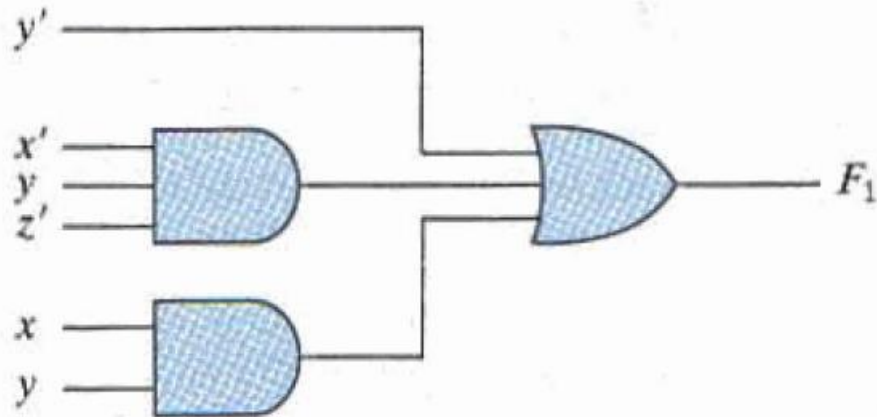
expand 1<sup>st</sup> term by ORing it with  $zz'$ , and 2<sup>nd</sup> term with  $yy'$

**product of 0-maxterms**

# SoP and PoS Logic Gate Implementation

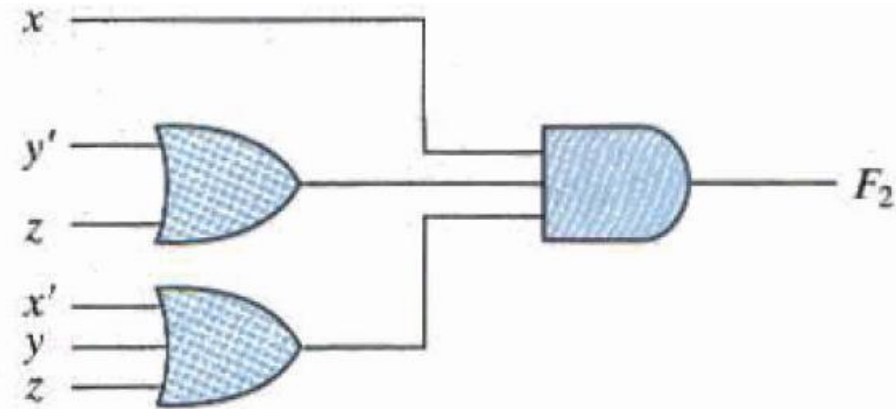


$$F_1 = y' + xy + x'yz'$$



(a) Sum of Products

$$F_2 = x(y' + z)(x' + y + z')$$



(b) Product of Sums

# Conversion from POS to SOP



<i>x</i>	<i>y</i>	<i>z</i>	Minterms		Maxterms	
			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	$xyz$	$m_7$	$x' + y' + z'$	$M_7$

$$F(x, y, z) = \Sigma(1, 4, 5, 6, 7)$$

This function has a complement that can be expressed as

$$F'(x, y, z) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

Now, if we take the complement of  $F'$  by DeMorgan's theorem, we obtain  $F$  in a different form:

$$F = (m_0 + m_2 + m_3)' = m'_0 \cdot m'_2 \cdot m'_3 = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

From the table, it is clear that the following relation holds:

$$m'_j = M_j$$



# Other Logic Gates

Name	Graphic symbol	Algebraic function	Truth table
------	----------------	--------------------	-------------

AND



$x$	$y$	$F$
0	0	0
0	1	0
1	0	0
1	1	1

OR



$x$	$y$	$F$
0	0	0
0	1	1
1	0	1
1	1	1

Inverter



$x$	$F$
0	1
1	0

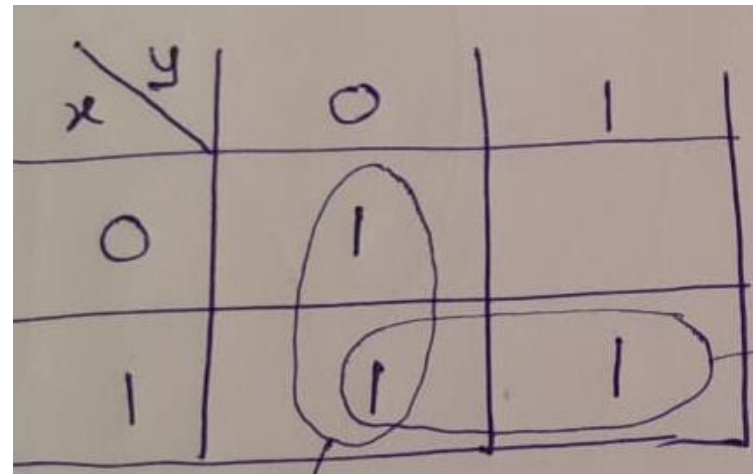


# K-MAP

1. Select a K-map according to the total number of variables.
2. Identify maxterms or minterms as given in the problem.
3. For SOP, put the 1's in the blocks of the K-map with respect to the minterms (elsewhere 0's).
4. For POS, putting 0's in the blocks of the K-map with respect to the maxterms (elsewhere 1's).
5. Making rectangular groups that contain the total terms in the power of two, such as 2,4,8 ..(except 1) and trying to cover as many numbers of elements as we can in a single group.
6. From the groups that have been created in step 5, find the product terms and then sum them up for the SOP form.

$$F = \sum (0, 2, 3)$$

0:	00
2:	10
3:	11



A hand-drawn K-map for a 2-variable function F(x, y). The map is a 2x2 grid with x and y as variables. The cells contain 0s and 1s. The 1s are at (0,1), (1,0), and (1,1). The 0s are at (0,0) and (1,1). The groups are: a vertical group of two 1s at (0,1) and (1,1), and a horizontal group of two 1s at (1,0) and (1,1).

x \ y	0	1
0	0	1
1	1	1

# K-MAP

$$F = \Sigma (0, 2, 3)$$

0:	0 0
1:	1 0
3:	1 1

$x \backslash y$	0	1
0	0	1
1	1	1

$$F_2 = x$$

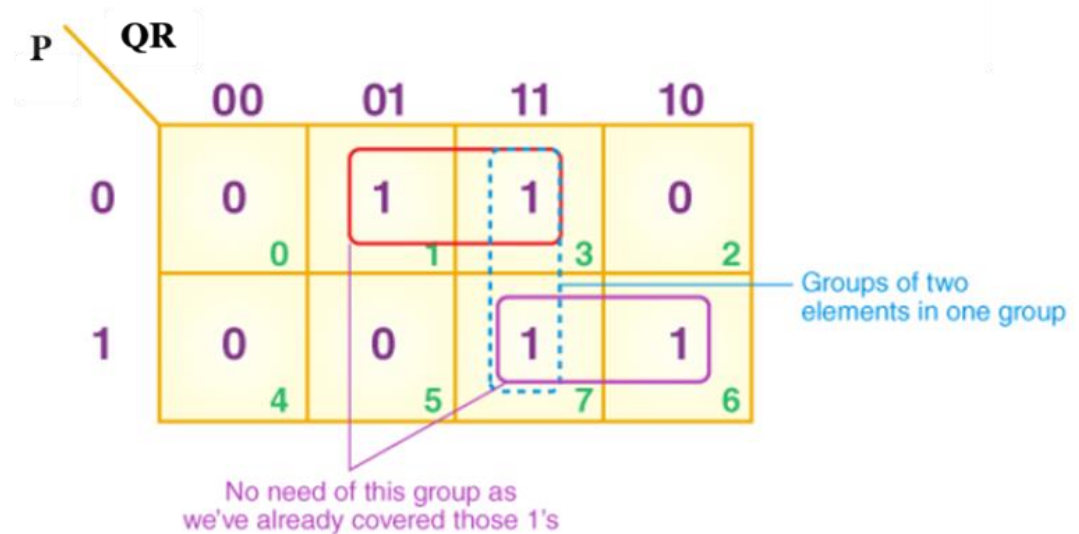
$$F_1 = y'$$

$$F = F_1 + F_2 = x + y'$$

# K-MAP

$$Z = \sum P, Q, R (1, 3, 6, 7)$$

1: 001  
3: 011  
6: 110  
7: 111



From the red group, the product term would be —

$P'R$

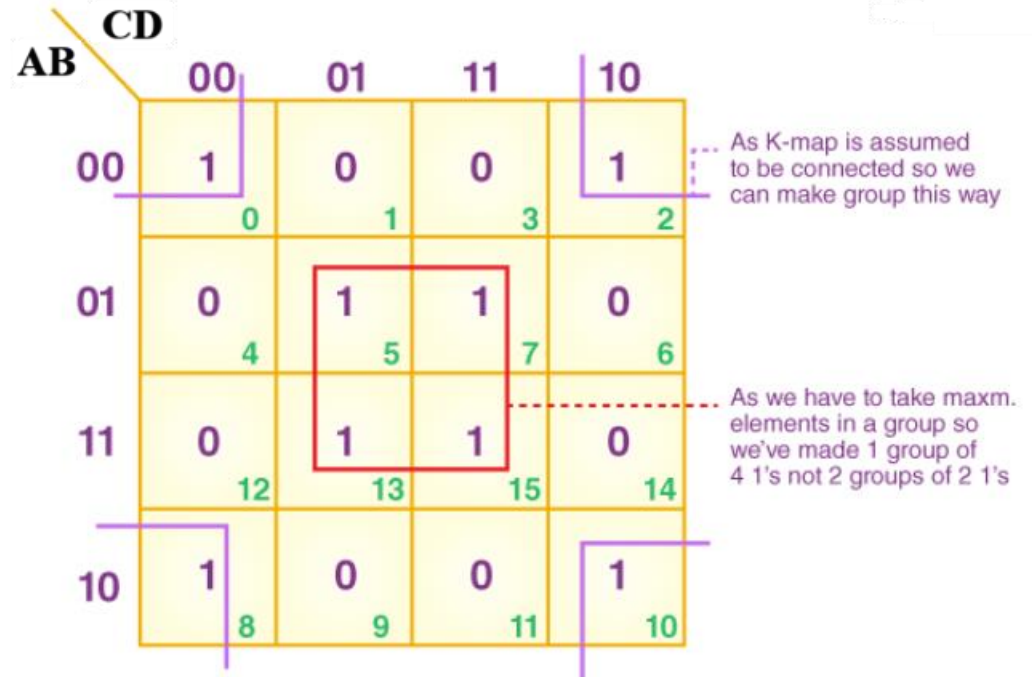
From the green group, the product term would be —

$PQ$

If we sum these product terms, then we will get this final expression ( $P'R + PQ$ )

# K-MAP

$$F(A, B, C, D) = \sum(0, 2, 5, 7, 8, 10, 13, 15)$$



From the red group, the product term would be —

$BD$

From the lilac group, the product term would be —

$B'D'$

If we sum these product terms, then we will get this final expression  $(BD + B'D')$

# Don't Care Condition



## Incompletely Specified Functions

Decimal Digit	8 4 2 1 BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

### Don't Care Combination

#### Un-Specified

1 0 1 0	→	
1 0 1 1	→	
1 1 0 0	→	0 or 1
1 1 0 1	→	
1 1 1 0	→	
1 1 1 1	→	

# Don't Care Condition



## Don't Care Combination

Minimal Expression in SOP form:

$$X \rightarrow 1$$

Minimal Expression in POS form:

$$X \rightarrow 0$$

# Don't Care Condition

$$F_1(A, B, C) = \sum (0, 1, 3, 5) + d(2, 4)$$

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

# Don't Care Condition

$$F_1(A, B, C) = \sum(0, 1, 3, 5) + d(2, 4)$$

without Don't cares

$$F_1(A, B, C) = \overline{A}\overline{B} + \overline{A}C + \overline{B}C$$

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

Annotations:

- Blue box around cells (0,0), (0,1), (0,2) labeled  $\overline{A}\overline{B}$
- Yellow box around cells (0,1), (0,2), (1,1) labeled  $\overline{A}C$
- Orange box around cells (0,1), (1,1) labeled  $\overline{B}C$





# Don't Care Condition

$$F_1(A, B, C) = \sum(0, 1, 3, 5) + d(2, 4)$$

with Don't cares

$$F_1(A, B, C) = \overline{A} + \overline{B}$$

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




Diagram illustrating the Karnaugh map for the function  $F_1(A, B, C)$  with don't care conditions. The map shows the function value for each combination of A, B, and C. The function is 1 for minterms 0, 1, 3, and 5, and don't care (X) for minterms 2 and 4. The simplified function is  $F_1(A, B, C) = \overline{A} + \overline{B}$ .

The map is grouped into two prime implicants:



- $\overline{A}$  (Grouped by a yellow rectangle covering minterms 0, 1, 3, and 2)
- $\overline{B}$  (Grouped by an orange rectangle covering minterms 0, 1, 4, and 5)

Source: <https://www.youtube.com/watch?v=ZayoUTi2tsA>

# Other Logic Gates

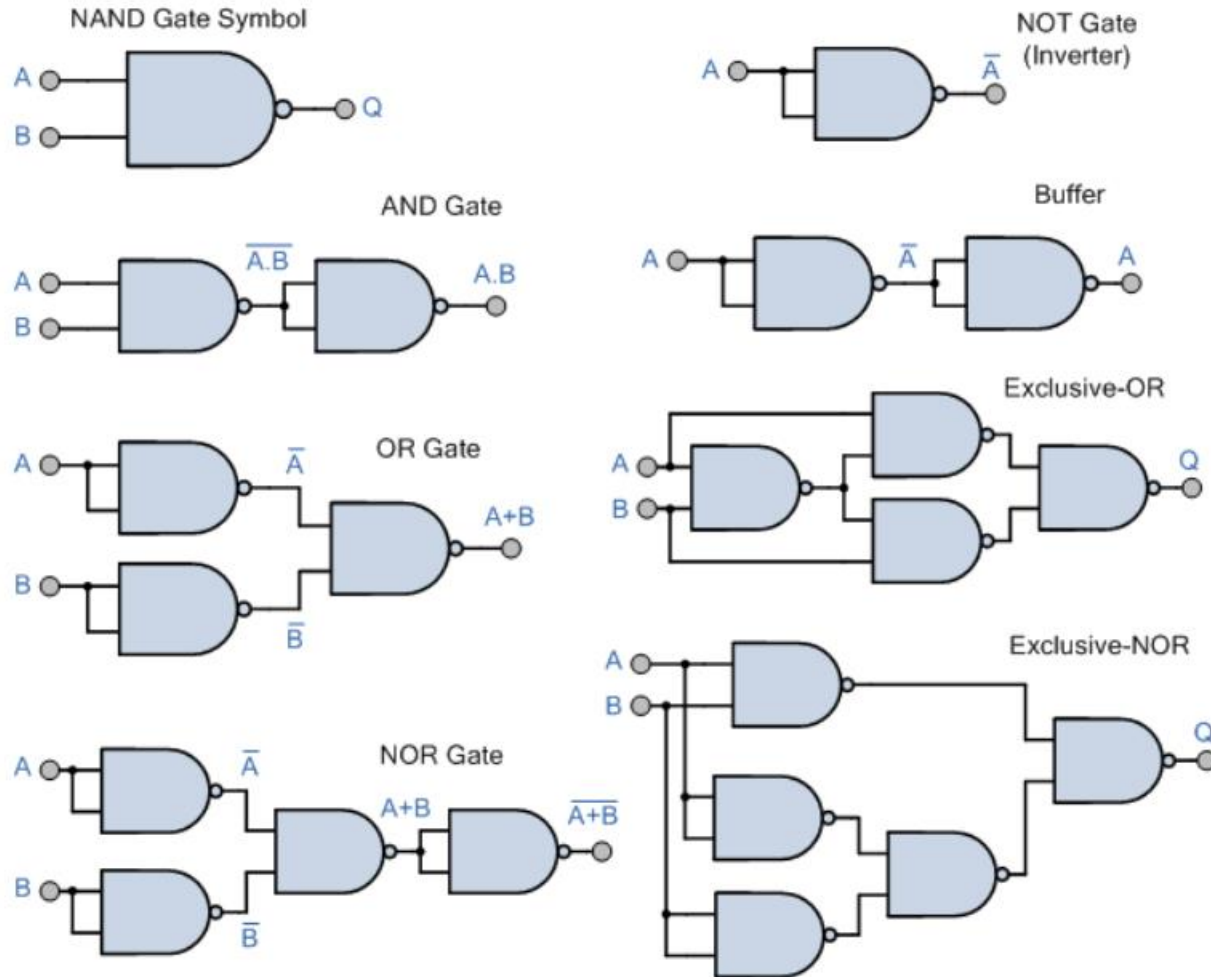
Name	Graphic symbol	Algebraic function	Truth table															
NAND		$F = (xy)'$	<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	$x$	$y$	$F$	0	0	1	0	1	1	1	0	1	1	1	0
$x$	$y$	$F$																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	$x$	$y$	$F$	0	0	1	0	1	0	1	0	0	1	1	0
$x$	$y$	$F$																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	$x$	$y$	$F$	0	0	0	0	1	1	1	0	1	1	1	0
$x$	$y$	$F$																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

# Other Logic Gates

Name	Graphic symbol	Algebraic function	Truth table															
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																
Buffer		$F = x$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	F	0	0	1	1									
x	F																	
0	0																	
1	1																	

Why **NAND** and **NOR** gates are called **universal gates**?

# All Logic Gates using only NAND Gate



# All Logic Gates using only NOR Gate

