

Basic Boolean Algebra

– Lecture Series by Intel



© 2009 Intel India Pvt Limited. All rights reserved. The information provided in this presentation is intended for the sole use of the recipient and is for educational purposes only. No part of this presentation may be reproduced or transmitted in any form or by any means, including photocopying and recording, without written permission. Permission must also be obtained before any part of this presentation is stored in a retrieval system in any nature. No responsibility can be accepted by Intel India Pvt Limited, the Editorial Board or contributors for action taken as a result of information contained in this presentation. The views expressed in this presentation by the presenter are not necessarily those of the Editorial Board or Intel India Pvt Limited.



Lecture Series

Basic Boolean Algebra

Oct 28th

First hour

Advanced Boolean Algebra

Oct 28th

Second hour



Basic Boolean Algebra

What will we learn?

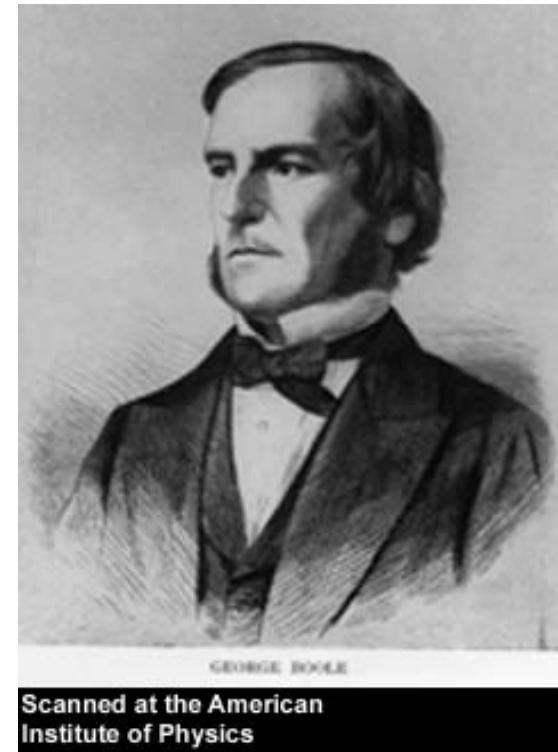
- ❑ Logic Gates
- ❑ Laws of Boolean Algebra
- ❑ Implementing Logic Functions
- ❑ Boolean Simplification



Historical Perspective

George Boole, 1815 - 1864

- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen's College in Ireland.
- Introduced binary variables
- Introduced the 3 fundamental logic operations: AND, OR, and NOT.



Boolean Constants and Variables

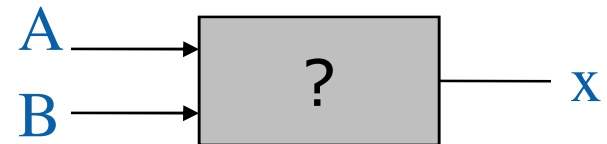
- ❑ Boolean 0 and 1 do not represent actual numbers but instead represent the state, or logic level.

Logic 0	Logic 1
False	True
Off	On
Low	High
No	Yes
Open switch	Closed switch

Truth Tables

- A truth table is a means for describing how a logic circuit's output depends on the logic levels present at the circuit's inputs.
- Any Boolean function can be represented in a truth table, where the number of rows is 2^n , and n is the number of binary variables in the function.

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	1
1	1	0



Logic Gates

□ Three basic logic operations

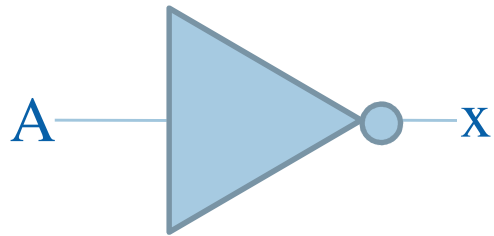
- Inverter
- AND
- OR

□ Other logic operations

- NAND
- NOR

Logic Gates

□ NOT gate

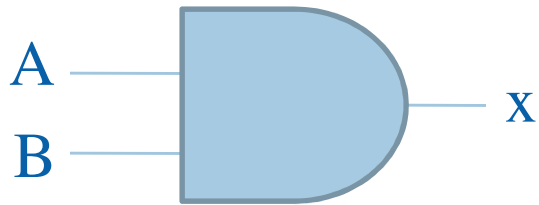


A	x
0	1
1	0

$$x = A'$$

Logic Gates

□ AND gate

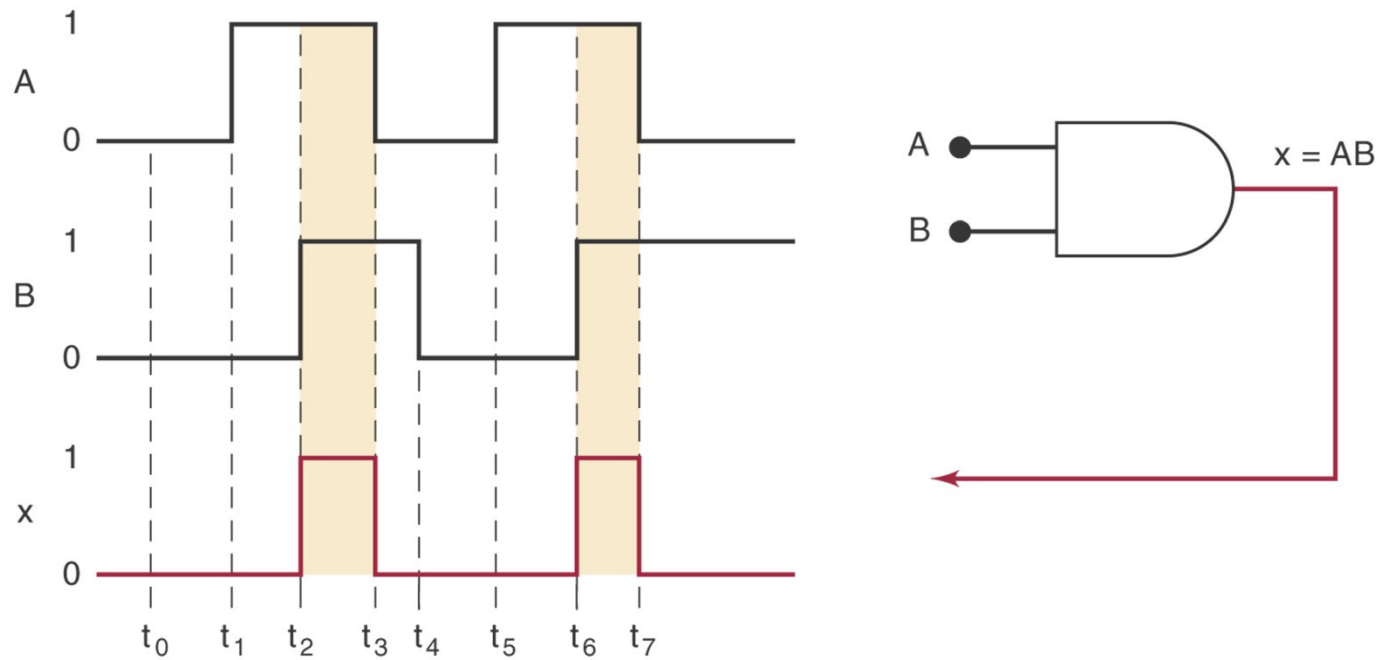


A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

$$X = A \cdot B$$

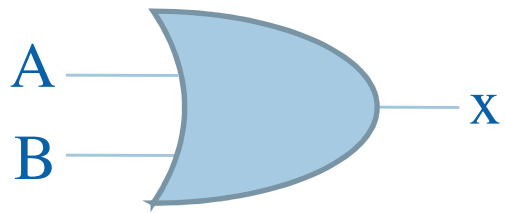
Logic Gates

AND gate - Timing Diagram



Logic Gates

□ OR gate

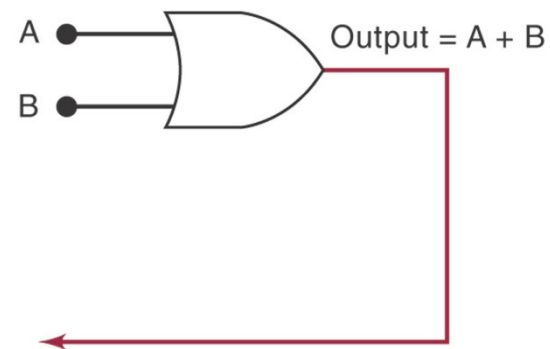
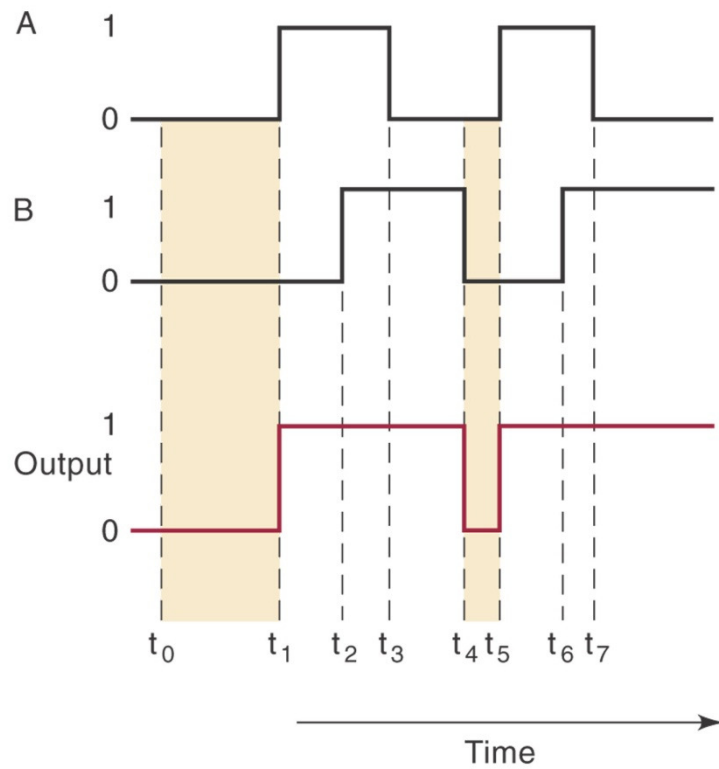


A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

$$x = A+B$$

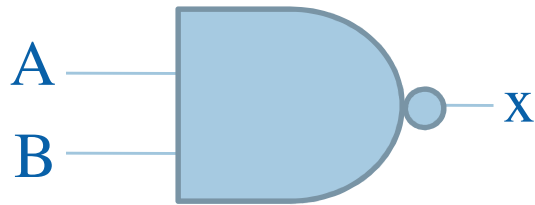
Logic Gates

OR gate - Timing Diagram



Logic Gates

□ NAND gate

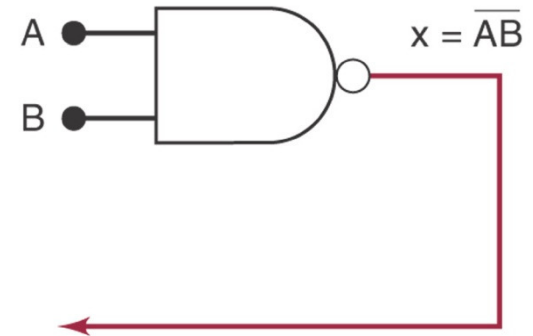
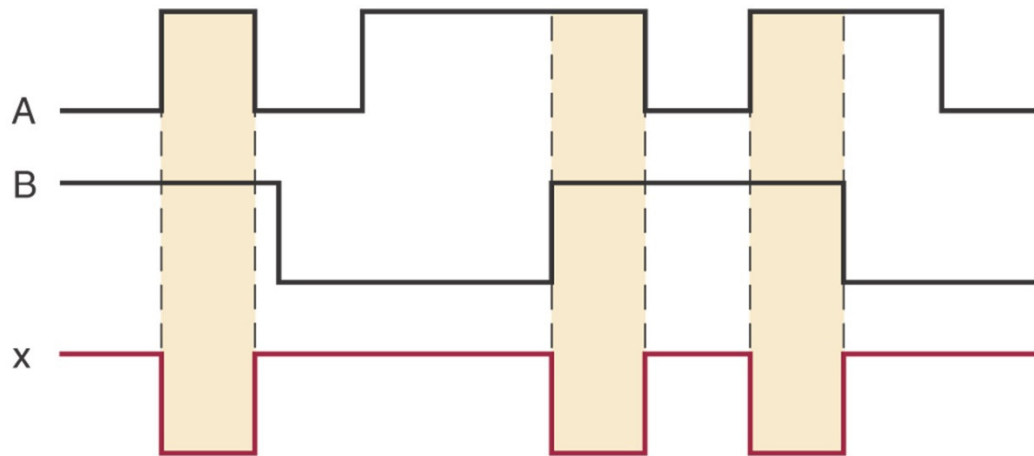


A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

$$X = \overline{A \cdot B}$$

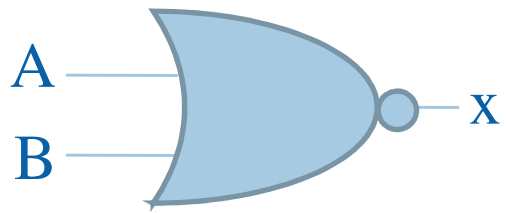
Logic Gates

□ NAND gate - Timing Diagram



Logic Gates

□ NOR gate

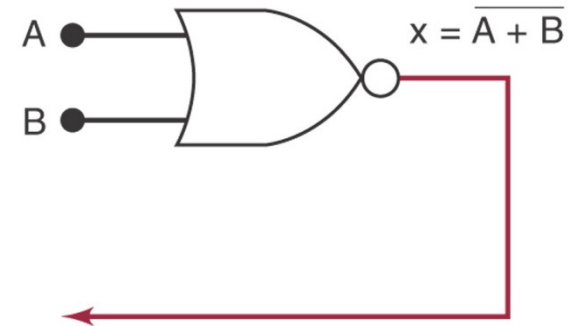
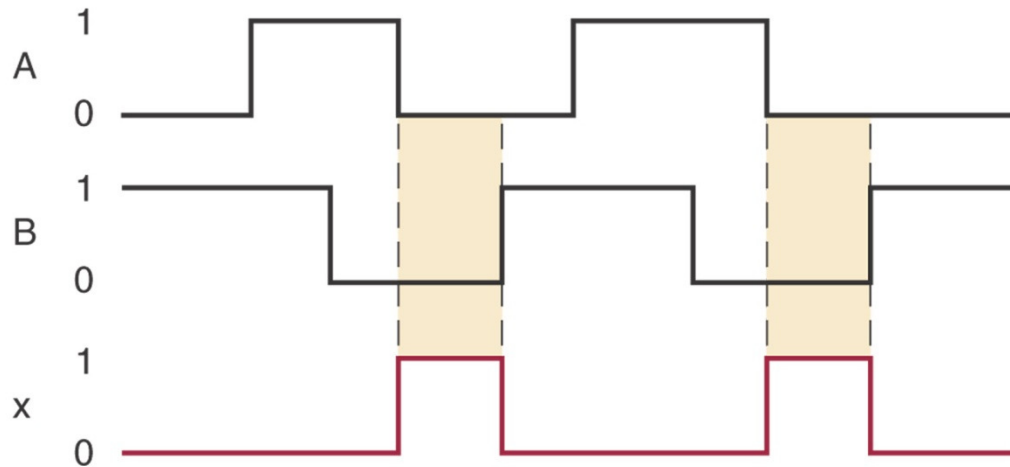


A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

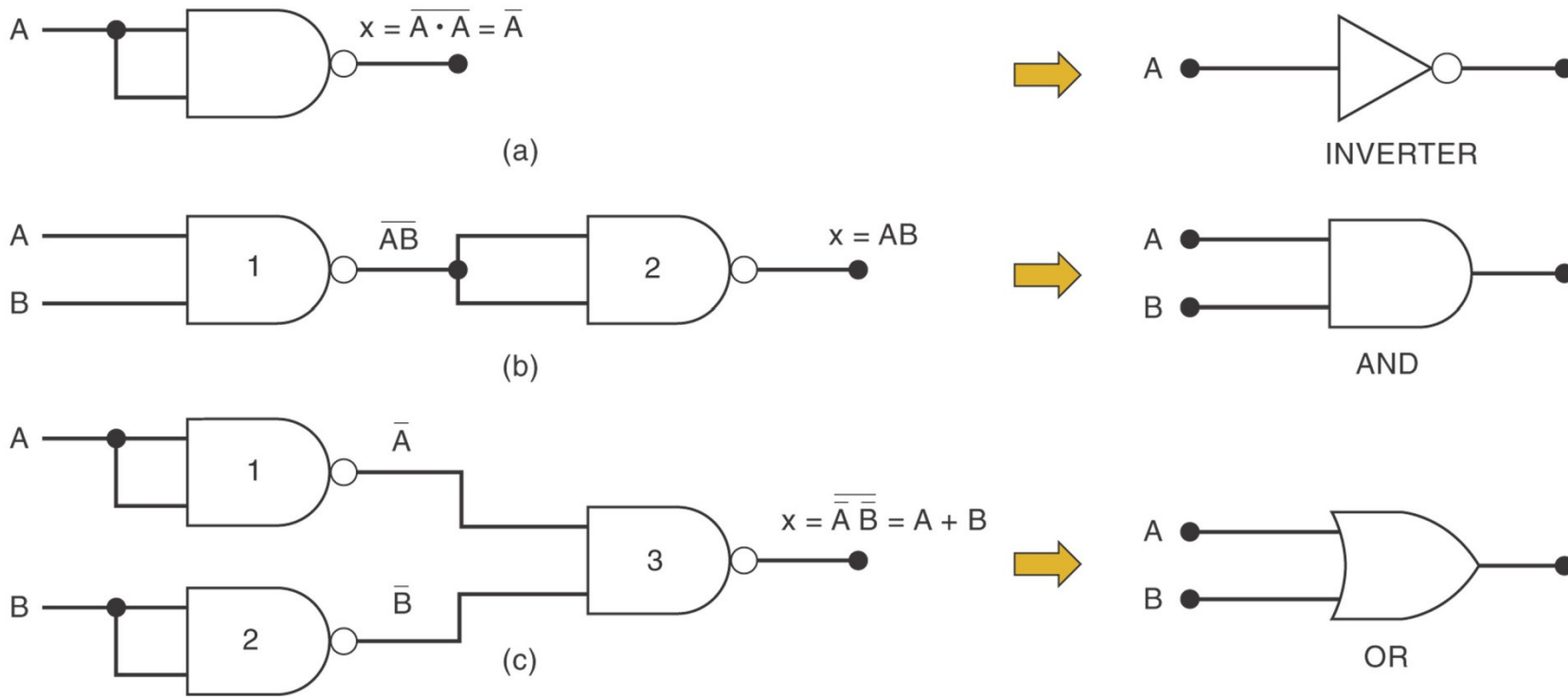
$$x = \overline{A+B}$$

Logic Gates

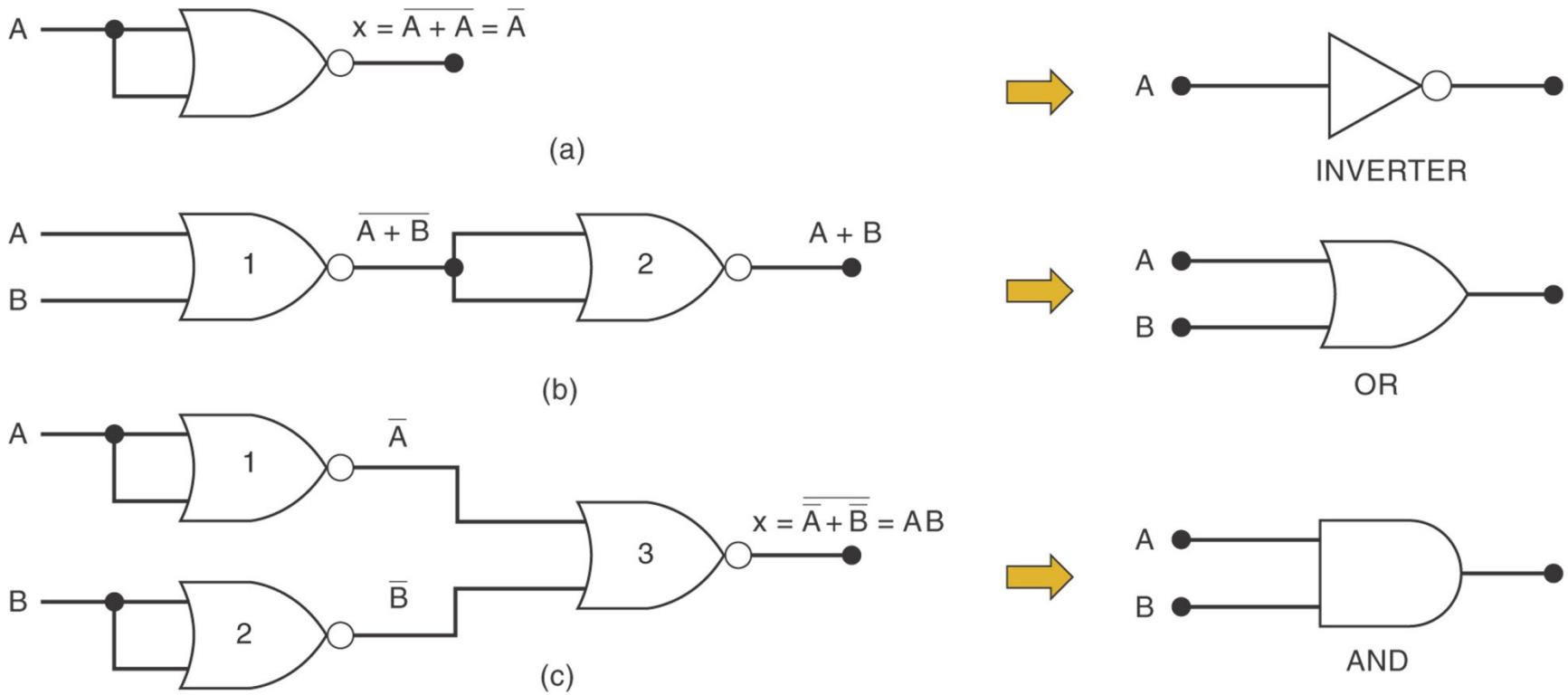
□ NOR gate - Timing Diagram



Universality of NAND Gates (NAND to Basic Gates Conversion)



Universality of NOR Gates (NOR to Basic Gates Conversion)

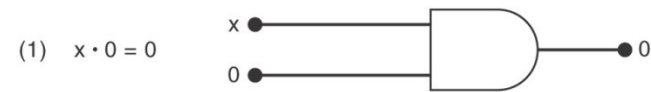


Boolean Function

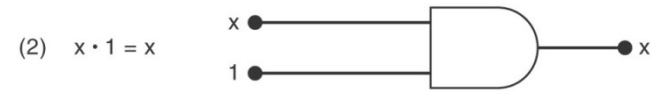
- ❑ A Boolean function is an expression formed with binary variables, the two binary operators OR and AND, the unary operator NOT, parenthesis, and equal sign.
- ❑ A binary variable can take the value of 0 or 1, and for a given value of the variables, the function can be either 0 or 1.

Boolean Equations (Single Variable)

1) $x \cdot 0 = 0$



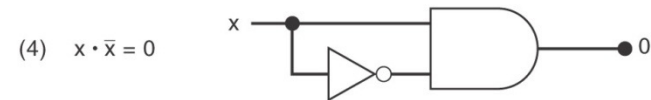
2) $x \cdot 1 = x$



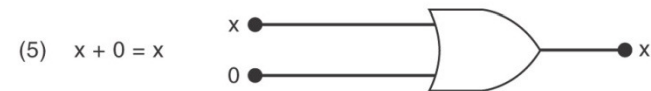
3) $x \cdot x = x$



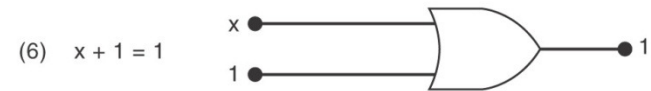
4) $x \cdot x' = 0$



5) $x + 0 = x$



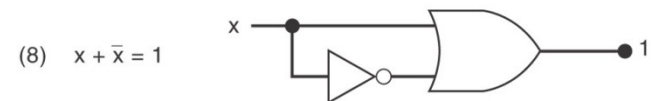
6) $x + 1 = 1$



7) $x + x = x$



8) $x + x' = 1$



Boolean Equations (Multiple Variables)

1) $x+y = y+x$

2) $x*y = y*x$

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

4) $x(yz)=(xy)z=xyz$

5) $x(y+z)=xy+xz$

6) $(w+x)(y+z)=wy+xy+wz+xz$

7) $x+xy=x$

8) $x+x'y=x+y$

9) $x'+xy=x'+y$

Duality Principle

- **Duality Principle** states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.

$x + 0 = x$	$x \cdot 1 = x$
$x + y = y + x$	$xy = yx$
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
$x + x' = 1$	$x \cdot x' = 0$

Boolean Manipulation

- ❑ *Literal*: A *literal* is a primed or unprimed variable.
- ❑ When a Boolean function is implemented with logic gates, *each literal* in the function designates an *input* to a gate, and *each term* is implemented with a *gate*.
- ❑ The *minimization* of the number of literals and the number of terms results in a circuit with *less* equipment.



Boolean Manipulation : Examples

- $x + x'y = (x + x')(x + y) = 1 \cdot (x + y) = x + y$
- $x(x' + y) = xx' + xy = 0 + xy = xy$
- $x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$
- $xy + x'z + yz = xy + x'z + yz(x + x')$
 $= xy + x'z + xyz + x'yz$
 $= xy(1 + z) + x'z(1 + y)$
 $= xy + x'z$
- $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$ *by duality.*

Complement of a Function

- The complement of a function F is F' and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F .
- Example:

$$(A + B + C)' = A'B'C'$$

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

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

Minterms

- ❑ A binary variable may appear either in its normal form (x) or in its complement form (x').
- ❑ Consider two binary variables x and y combined with an AND operation. Since each variable may appear in either the normal or complementary form, there are four possible combinations: xy , xy' , $x'y$, and $x'y'$.
- ❑ Each is called a *minterm* or *standard product*.
- ❑ Any Boolean function can be expressed as a sum of minterms (by *sum* is meant the ORing of terms).
- ❑ n variables can be combined to form 2^n minterms.

Minterms

- Minterms for 3 binary variables x, y, z

	Minterms	
$x y z$	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

Maxterms

- ❑ n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called *max terms* or *standard sums*.
- ❑ Any Boolean function can be expressed as a product of maxterms (by *product* is meant the *ANDing* of terms).
- ❑ Each maxterm is the complement of its corresponding minterm, and *vice versa*.
- ❑ Boolean functions expressed as a *sum* of minterms or *product* of maxterms are said to be in *canonical form*.

Maxterms

□ Maxterms for 3 binary variables x, y, z

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

Sum of Products (SOP)

- SOP is a Boolean expression containing AND terms, called *product terms*, of one or more literals each. The *sum* denotes ORing of these terms.
- Ex: $F_1 = y' + xy + x'yz'$

Product of Sums (POS)

- A *POS* is a Boolean expression containing OR terms, called *sum terms*. Each term may have any number of literals. The *product* denotes the ANDing of these terms.
- *Ex:* $F_2 = x(y' + z)(x' + y + z' + w)$

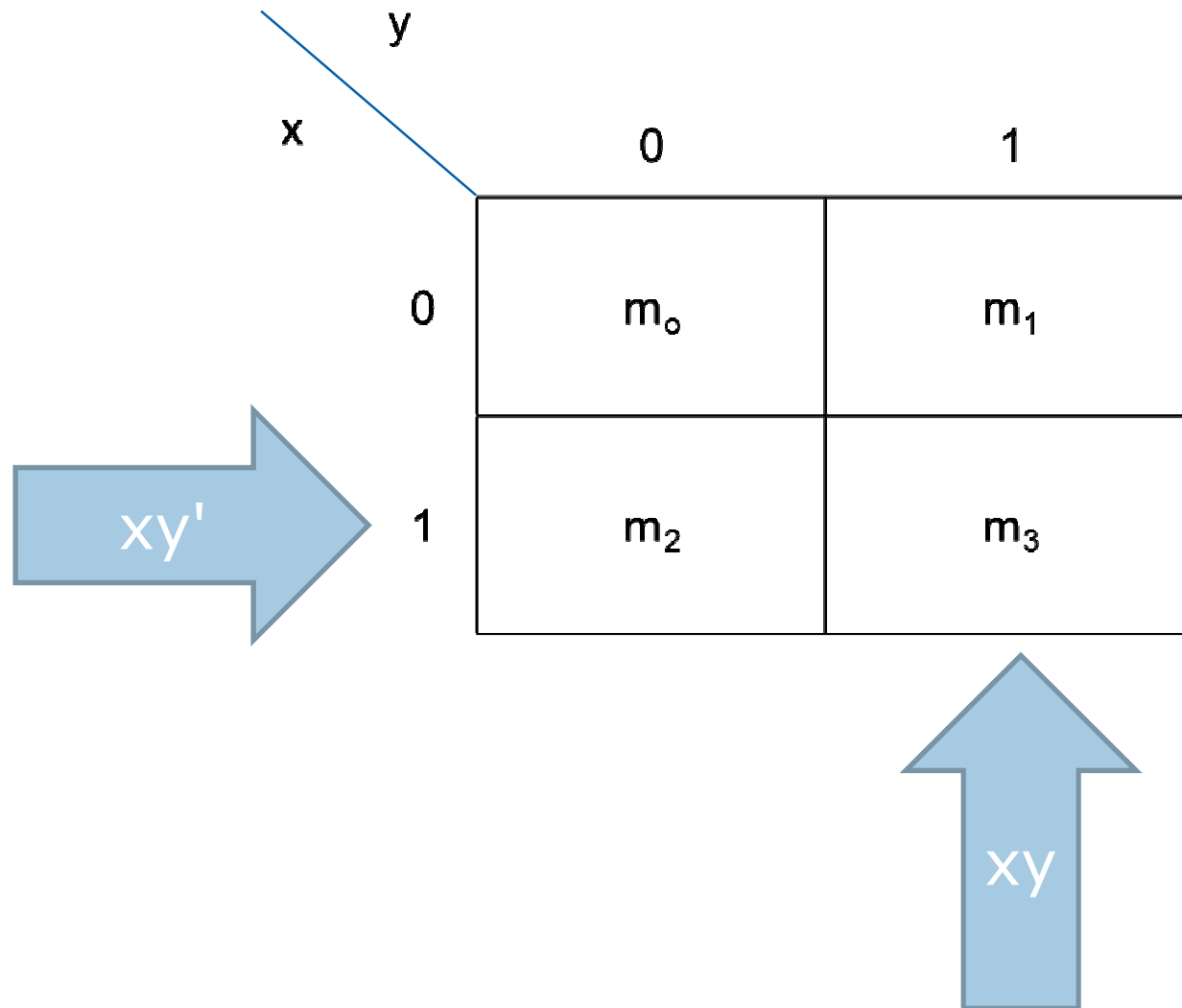
Boolean Simplification

- ❑ Using Boolean Equations: Boolean functions may be simplified algebraically, but the procedure of minimization lacks specific rules to predict each succeeding step in the manipulative process.
- ❑ Using K-Maps: A method that provides a simple straightforward procedure for minimizing Boolean functions is by using Karnaugh Map.

Boolean Simplification using K-Maps

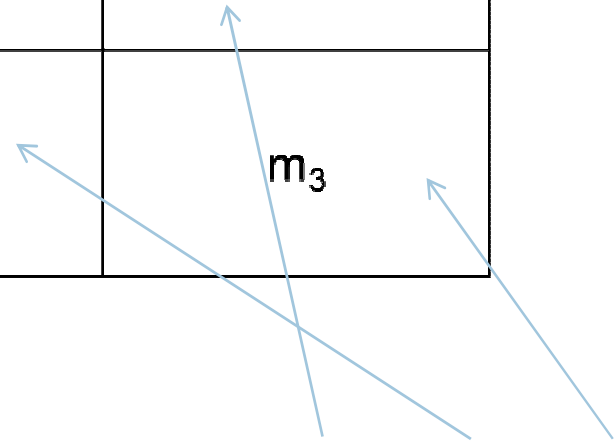
- ❑ Diagram made up of squares with each representing one minterm.
- ❑ A Boolean function is expressed in the map as the area enclosed by the squares whose minterms are included in the function.
- ❑ The map presents a visual diagram of all possible ways a function may be expressed in standard form.
- ❑ One may use the map to derive alternative algebraic expressions for the same function, from which the simplest one can be selected.

Two-Variable K-Map



Two-Variable K-Map

		y	
		0	1
x	0	m_0	m_1
	1	m_2	m_3



$$F(x,y) = x + y = x'y + xy' + xy = m_1 + m_2 + m_3$$

Two-Variable K-Map

		<i>y</i>	
	<i>x</i>		
		0	1
0			1
1		1	1

$$F(x,y) = x + y = x'y + xy' + xy = m_1 + m_2 + m_3$$

Three-Variable K-Map

		y z			
		00	01	11	10
x	0	$x'y'z'$ m_0	$x'y'z$ m_1	$x'yz$ m_3	$x'yz'$ m_2
	1	$xy'z'$ m_4	$xy'z$ m_5	xyz m_7	xyz' m_6

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}

Don't Cares in K-map

- ❑ The 1's and 0's in the map represent the minterms and make the function equals to 1 or 0.
- ❑ There are occasions when it does not matter if the function produces 0 or 1 for a given minterm, we say that we don't care what the function o/p is to be for this minterm.
- ❑ Minterms that may produce either 0 or 1 for the function are said to be don't-care conditions and are marked with 'X' in the map. These don't -care conditions can be used to provide further simplification of the algebraic expression.

Don't cares in K-Map

		y z			
		00	01	11	10
x	0	0	0	0	0
	1	0	1	x	x

$$f = xy'z$$

		y z			
		00	01	11	10
x	0	0	0	0	0
	1	0	1	x	x

$$f = xz$$

Basic Boolean Algebra

What **did** we learn?

- ❑ Logic Gates
- ❑ Laws of Boolean Algebra
- ❑ Implementing Logic Functions
- ❑ Boolean Simplification

