# Boolean Algebra

## BOOLEAN ALGEBRA

- **Formal logic**: In formal logic, a statement (proposition) is a declarative sentence that is either true(1) or false (0).
- It is easier to communicate with computers using formal logic.
- **Boolean variable:** Takes only two values – either true (1) or false (0).
- They are used as basic units of formal logic.

## Boolean function and logic diagram

- **Boolean function:** Mapping from Boolean variables to a Boolean value.
- **Truth table**:
  - Represents relationship between a Boolean function and its binary variables.
  - It enumerates all possible combinations of arguments and the corresponding function values.

## Boolean function and logic diagram

- **Boolean algebra**: Deals with binary variables and logic operations operating on those variables.
- **Logic diagram**: Composed of graphic symbols for logic gates.
  - A simple circuit sketch that represents inputs and outputs of Boolean functions.

## Boolean Algebra

- Boolean Algebra: a useful mathematical system for specifying and transforming logic functions.
- We study Boolean algebra as a foundation for designing and analyzing digital systems!

## Logical Operations

- The three basic logical operations are:
  - AND
  - OR
  - NOT
- AND is denoted by a dot (·).
- OR is denoted by a plus (+).
- NOT is denoted by an overbar ( ¯ ), a single quote mark (') after, or (~) before the variable.

## Operator Definitions

- Operations are defined on the values "0" and "1" for each operator:

| AND | OR | NOT |
|---|---|---|
| $0 \cdot 0 = 0$ | $0 + 0 = 0$ | $\overline{0} = 1$ |
| $0 \cdot 1 = 0$ | $0 + 1 = 1$ | $\overline{1} = 0$ |
| $1 \cdot 0 = 0$ | $1 + 0 = 1$ | |
| $1 \cdot 1 = 1$ | $1 + 1 = 1$ | |

## BASIC IDENTITIES OF BOOLEAN ALGEBRA

- A Boolean algebra is a closed algebraic system containing a set *K* of two or more elements and the two operators $\cdot$ and + which refer to logical AND and logical OR

## Basic Identities of Boolean Algebra (Existence of 1 and 0 element)

(1) $x + 0 = x$

(2) $x \cdot 0 = 0$

(3) $x + 1 = 1$

(4) $x \cdot 1 = 1$

## Basic Identities of Boolean Algebra (Existence of complement)

(5) $x + x = x$

(6) $x \cdot x = x$

(7) $x + x' = x$

(8) $x \cdot x' = 0$

## Basic Identities of Boolean Algebra (Commutativity):

(9) $x + y = y + x$

(10) $xy = yx$

## Basic Identities of Boolean Algebra (Associativity):

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

(12) $x (yz) = (xy) z$

## Basic Identities of Boolean Algebra (Distributivity):

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

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

## Basic Identities of Boolean Algebra (DeMorgan's Theorem)

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

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

## Basic Identities of Boolean Algebra (Involution)

(17) $(x')' = x$

## Some Properties of Boolean Algebra

- Boolean Algebra is defined in general by a set *B* that can have more than two values
- A two-valued Boolean algebra is also know as Switching Algebra. The Boolean set *B* is restricted to 0 and 1. Switching circuits can be represented by this algebra.

## Some Properties of Boolean Algebra

- The dual of an algebraic expression is obtained by interchanging + and · and interchanging 0's and 1's.
- The identities appear in dual pairs. When there is only one identity on a line the identity is self-dual, i. e., the dual expression = the original expression.
- Sometimes, the dot symbol '·' (AND operator) is not written when the meaning is clear

## Dual of a Boolean Expression

- Example: $F = (A + \overline{C}) \cdot B + 0$

  dual $F = (\overline{A} \cdot C + B) \cdot 1 = \overline{A} \cdot C + B$
- Example: $G = X \cdot Y + \overline{(W + Z)}$

  dual $G = (X+Y) \cdot \overline{(W \cdot Z)} = (X+Y) \cdot (\overline{W}+\overline{Z})$
- Example: $H = A \cdot B + A \cdot C + B \cdot C$

  dual $H = (A+B) \cdot (A+C) \cdot (B+C)$

## Boolean Operator Precedence

- The order of evaluation is:
  1. Parentheses
  2. NOT
  3. AND
  4. OR
- Consequence: Parentheses appear around OR expressions
- Example: F = A(B + C)(C + D) —

## Function Minimization using Boolean Algebra

- ***Examples***:
- (a) $a + ab = a(1+b)=a$
- (b) $a(a + b) = a.a + ab=a+ab=a(1+b)=a.$
- (c) $a + a'b = (a + a')(a + b)=1(a + b)$ $=a+b$
- (d) $a(a' + b) = a.a' + ab=0+ab=ab$

## Try

- F = abc + abc′ + a′c

## The other type of question

Show that;

1- $ab + ab' = a$

2- $(a + b)(a + b') = a$

1- $ab + ab' = a(b+b') = a.1=a$

2- $(a + b)(a + b') = a.a + a.b' + a.b+b.b'$

$= a + a.b' + a.b + 0$

$= a + a.(b' + b) + 0$

$= a + a.1 + 0$

$= a + a = a$

## More Examples

- Show that;

(a) $ab + ab'c = ab + ac$

(b) $(a + b)(a + b' + c) = a + bc$

(a) $ab + ab'c = a(b + b'c)$

$= a((b+b').(b+c))=a(b+c)=ab+ac$

(b) $(a + b)(a + b' + c)$

$= (a.a + a.b' + a.c + ab + b.b' + bc)$

= …

## DeMorgan's Theorem

(a) $(a + b)' = a'b'$
(b) $(ab)' = a' + b'$

Generalized DeMorgan's Theorem
(a) $(a + b + \ldots z)' = a'b' \ldots z'$
(b) $(a.b \ldots z)' = a' + b' + \ldots z'$

## DeMorgan's Theorem

- F = ab + c′d′
- F′ = ??

- F = ab + c′d′ + b′d
- F′ = ??

## ***DeMorgan's Theorem***

***Show that***: $(a + b.c)' = a'.b' + a'.c'$

## More *DeMorgan's* example

***Show that:*** $(a(b + z(x + a')))' = a' + b'(z' + x')$

$$
\begin{aligned}
(a(b + z(x + a')))' &= a' + (b + z(x + a'))' \\
&= a' + b'(z(x + a'))' \\
&= a' + b'(z' + (x + a')') \\
&= a' + b'(z' + x'(a')') \\
&= a' + b'(z' + x'a) \\
&= a' + b'z' + b'x'a \\
&= (a' + b'x'a) + b'z' \\
&= (a' + b'x')(a + a') + b'z' \\
&= a' + b'x' + b'z' \\
&= a' + b'(z' + x')
\end{aligned}
$$

## Truth Table

- **Logic diagram:** a graphical representation of a circuit
  - Each type of gate is represented by a specific graphical symbol
- **Truth table:** defines the function of a gate by listing all possible input combinations that the gate could encounter, and the corresponding output

## Truth Tables

- Example: Truth tables for the basic logic operations:

AND

| X | Y | Z = X·Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR

| X | Y | Z = X+Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

NOT

| X | $Z = \overline{X}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Truth Tables – Cont'd

- Used to evaluate any logic function
- Consider $F(X, Y, Z) = X\,Y + \overline{Y}\,Z$

| $X$ | $Y$ | $Z$ | $X\,Y$ | $\overline{Y}$ | $\overline{Y}\,Z$ | $F = X\,Y + \overline{Y}\,Z$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

# Logic Gates

## Logic Gates

- A logic gate is an elementary building block of a digital circuit .
- Most logic gates have two inputs and one output.
- At any given moment, every terminal is in one of the two binary conditions *low* (0) or *high* (1), represented by different voltage levels.

## 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 |

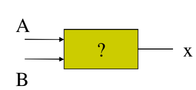
## Three Basic Logic Operations

- OR
- AND
- NOT
- Other derived logic operations
  - NOR
  - NAND
  - XOR
  - XNOR

## 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 inp

| Inputs | | Output |
|---|---|---|
| A | B | x |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



## OR Gate

- The *OR gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or."
- The output is "true" if either or both of the inputs are "true."
- If both inputs are "false," then the output is "false."

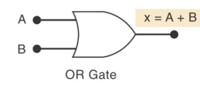
## OR Operation

- Boolean expression for the OR operation:
  x =A + B
- The above expression is read as "x equals A OR B"

OR

| A | B | x = A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(a)

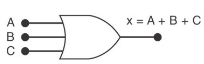


OR Gate

(b)

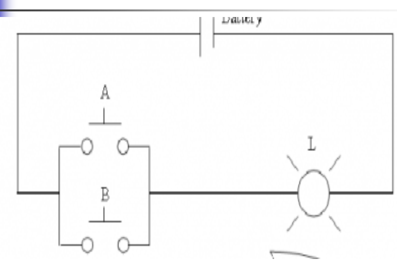
## OR Gate

- An OR gate is a gate that has two or more inputs and whose output is equal to the OR combination of the inputs.




| A | B | C | x = A + B + C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 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 |

## The switch equivalent of an OR gate




## AND Gate

- The *AND gate* is so named because, if 0 is called "false" and 1 is called "true,"
- the gate acts in the same way as the logical "and" operator.

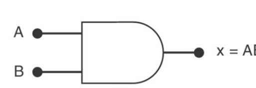
## AND Operation

- Boolean expression for the AND operation:
  x =A B
- The above expression is read as "x equals A AND B"

AND

| A | B | x = A · B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(a)



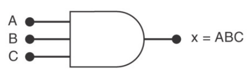
AND gate

(b)

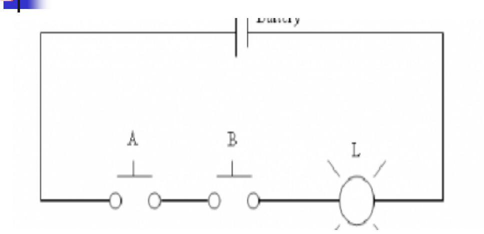
## AND Gate

- An AND gate is a gate that has two or more inputs and whose output is equal to the AND product of the inputs

| A | B | C | x = ABC |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |




## The switch equivalent of an AND gate




## NOT Operation

- A logical *inverter* , sometimes called a *NOT gate* to differentiate it from other types of electronic inverter devices,
- has only one input.
- It reverses the logic state.

## NOT Operation

Boolean expression for the NOT operation:

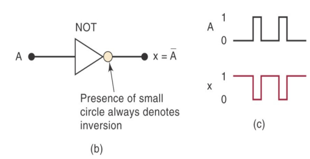
$x = \bar{A}$

- The above expression is read as "x equals the inverse of A"
- Also known as inversion or complementation.
- Can also be expressed as: A'

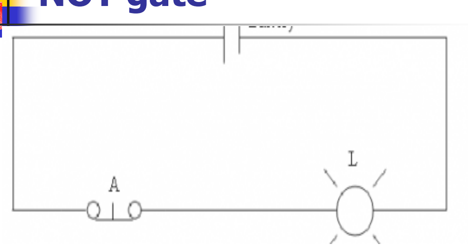
NOT

| A | $x = \bar{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

(a)




## The switch equivalent of a NOT gate



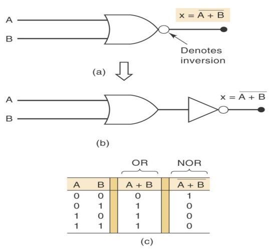

## *NOR gate*

- The *NOR gate* is a combination OR gate followed by an inverter.
- Its output is "true" if both inputs are "false."
- Otherwise, the output is "false."

## NOR Gate

- Boolean expression for the NOR operation:

x



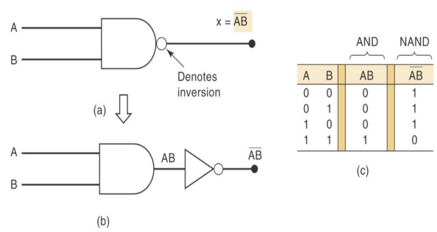

## The *NAND gate*

- The *NAND gate* operates as an AND gate followed by a NOT gate.
- It acts in the manner of the logical operation "and" followed by negation.
- The output is "false" if both inputs are "true."
- Otherwise, the output is "true."

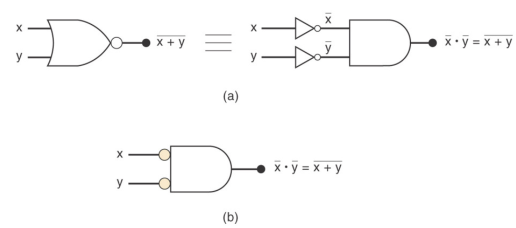
## NAND Gate

- Boolean expression for the NAND operation:

x




## alternative symbol for NOR function



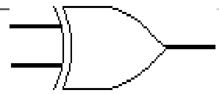

## *XOR ( exclusive-OR ) gate*

- The *XOR ( exclusive-OR ) gate* acts in the same way as the logical "either/or."
- The output is "true" if either, but not both, of the inputs are "true."
- The output is "false" if both inputs are "false" or if both inputs are "true."

## *XOR ( exclusive-OR ) gate*

- Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

## XOR ( exclusive-OR ) gate

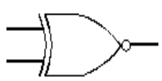


| Input 1 | Input 2 | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## XNOR (exclusive-NOR) gate

- The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter.
- Its output is "true" if the inputs are the same, and"false" if the inputs are different

## XNOR (exclusive-NOR) gate

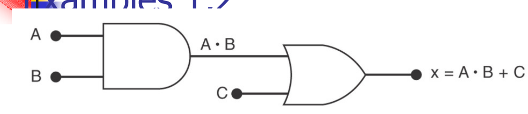


| Input 1 | Input 2 | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

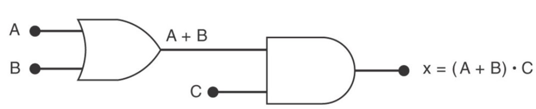
## Describing Logic Circuits Algebraically

- Any logic circuits can be built from the three basic building blocks: OR, AND, NOT
- Example 1: x = A B + C
- Example 2: x = (A+B)C
- Example 3: x = ($\overline{A+B}$)
- Example 4: x = ABC (A+D)

## Examples 1,2

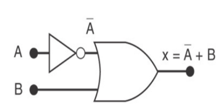



(a)

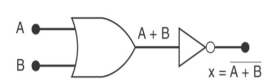



(b)

## Examples 3

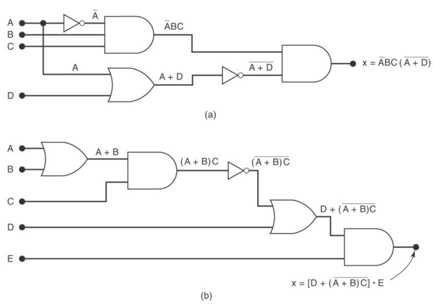



(a)




(b)

## Example 4

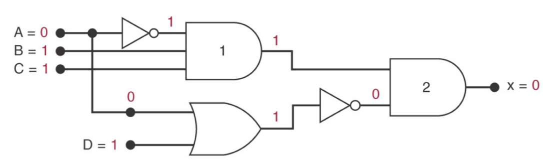



## Evaluating Logic Circuit Output

- Evaluation rule for Boolean expression
  1) Perform all inversions of single terms
  2) Perform all operations within parentheses
  3) Perform an AND operation before an OR operation
  4) If an expression has a bar over it, perform the expression first and then invert the result

## Evaluating Logic-Circuit Outputs

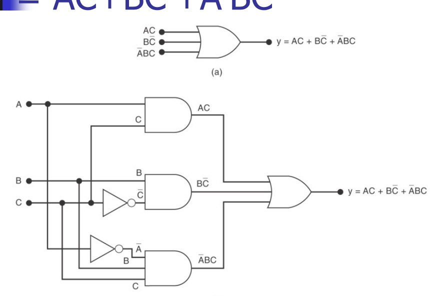
- $x = \bar{A}BC(\overline{A+D})$
- Determine the output x given A=0, B=1, C=1, D=1.
- Can also determine output level from a diagram



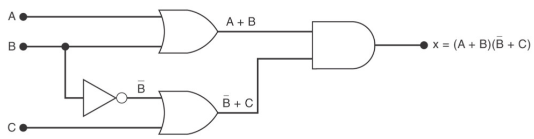

## Implementing Circuits from Boolean Expressions

- y = AC+BC′+A′BC
- x=(A+B)(B′+C)

## y = AC+BC′+A′BC



## x=(A+B)(B′+C)



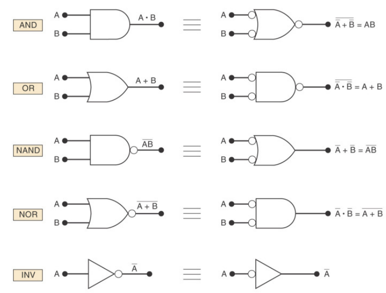

## Alternate Logic Symbols

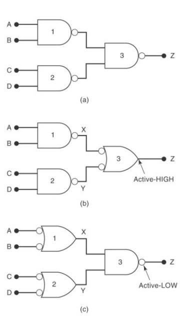
- Sometimes the logic of a circuit is more easily understood if an alternative gate shape is used.
- **DeMorgan=s Theorem** can be used to determine equivalent logic gates.

## Alternate Logic Symbols

- **Rules:**
- 1. Change the gate shape (AND ==> OR; OR ==> AND).
- 2. Change the bubbles (add where missing; remove if present).

## Alternate Logic-Gate Representation




| A | B | C | D | Z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 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 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(d)