Digital Logic Design 1

Tran Ngoc Thinh
HCMC University of Technology
http://www.cse.hcmut.edu.vn/~tnthinh

Boolean Constants and Variables

- Boolean algebra is an important tool in describing, analyzing, designing, and implementing digital circuits.
- Boolean algebra allows only two values; 0 and 1.
- Logic 0 can be: false, off, low, no, open switch.
- Logic 1 can be: true, on, high, yes, closed switch.
- Three basic logic operations: OR, AND, and NOT.

Truth Tables

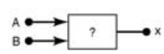
- A truth table describes the relationship between the input and output of a logic circuit.
- The number of entries corresponds to the number of inputs. For example a 2 input table would have $2^2 = 4$ entries. A 3 input table would have $2^3 = 8$ entries.

Truth Tables

- Examples of truth tables with 2, 3, and 4 inputs.

Inputs / Output

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



(a)

A	B	C	x
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	0
1	1	1	1

(b)

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
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	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

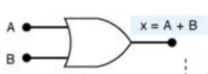
(c)

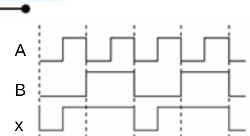
OR Operation With OR Gates

- The Boolean expression for the OR operation is
  X = A + B
  - This is read as "x equals A or B."
  - X = 1 when A = 1 or B = 1.
- Truth table, circuit symbol and timing diagram for a two input OR gate:

OR

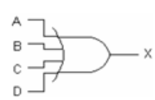
A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

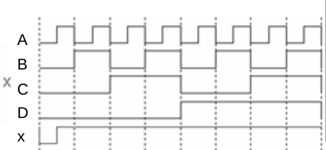




OR Operation With OR Gates

- The OR operation is similar to addition but when A = 1 and B = 1, the OR operation produces 1 + 1 = 1.
- In the Boolean expression
  x=1+1+1=1
  We could say that x is true (1) when A is true (1) OR B is true (1) OR C is true (1).
- In general, the output of an OR gate is HIGH whenever one or more inputs are HIGH

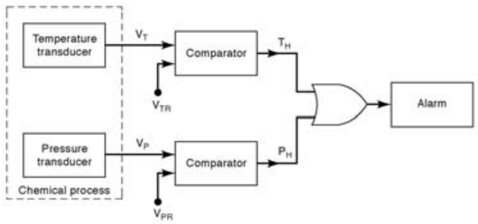




A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

OR Operation With OR Gates

- There are many examples of applications where an output function is desired when one of multiple inputs is activated.

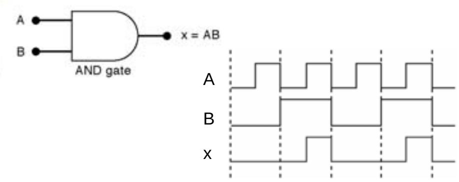


AND Operations with AND gates

- The Boolean expression for the AND operation is
  X = A • B
  - This is read as "x equals A and B."
  - x = 1 when A = 1 and B = 1.
- Truth table and circuit symbol for a two input AND gate are shown. Notice the difference between OR and AND gates.

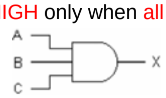
AND

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

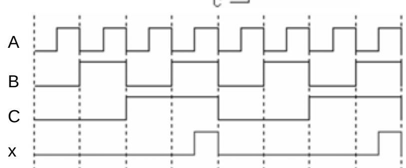


AND Operation With AND Gates

- The AND operation is similar to multiplication.
- In the Boolean expression
  X = A • B • C
  X = 1 only when A = 1, B = 1, and C = 1.
- The output of an AND gate is HIGH only when all inputs are HIGH.



A	B	C	X
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



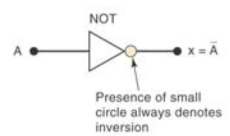
NOT Operation

- The Boolean expression for the NOT operation is
  $$X = \overline{A}$$
- This is read as:
  - x equals NOT A, or
  - x equals the inverse of A, or
  - x equals the complement of A
- Truth table, symbol, and sample waveform for the NOT circuit.

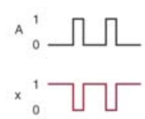
NOT

A	$x = \overline{A}$
0	1
1	0

(a)



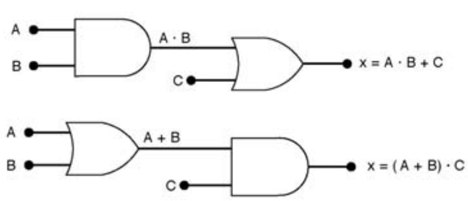
(b)



(c)

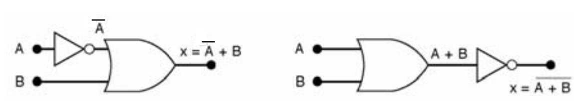
Describing Logic Circuits Algebraically

- The three basic Boolean operations (OR, AND, NOT) can describe any logic circuit.
- Examples of Boolean expressions for logic circuits:



Describing Logic Circuits Algebraically

- The output of an inverter is equivalent to the input with a bar over it. Input A through an inverter equals A.
- Examples using inverters.



Evaluating Logic Circuit Outputs

- Rules for evaluating a Boolean expression:
  - Perform all inversions of single terms.
  - Perform all operations within parenthesis.
  - Perform AND operation before an OR operation unless parenthesis indicate otherwise.
  - If an expression has a bar over it, perform the operations inside the expression and then invert the result.

Evaluating Logic Circuit Outputs

- Evaluate Boolean expressions by substituting values and performing the indicated operations:

$$
\begin{aligned}
&A = 0, B = 1, C = 1, \text{and } D = 1\\
x &= \bar{A}BC\overline{(A+D)}\\
&= \bar{0}\cdot 1\cdot 1\cdot \overline{(0+1)}\\
&= 1\cdot 1\cdot 1\cdot \overline{(0+1)}\\
&= 1\cdot 1\cdot 1\cdot \overline{(1)}\\
&= 1\cdot 1\cdot 1\cdot 0\\
&= 0
\end{aligned}
$$

Evaluating Logic Circuit Outputs

- Output logic levels can be determined directly from a circuit diagram.
- Technicians frequently use this method.
- The output of each gate is noted until a final output is found.

Implementing Circuits From Boolean Expressions

- It is important to be able to draw a logic circuit from a Boolean expression.
- The expression

$$x = A\cdot B\cdot C$$

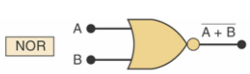
  could be drawn as a three input AND gate.
- A more complex example such as

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

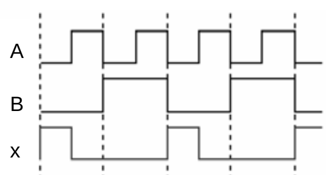
  could be drawn as two 2-input AND gates and one 3-input AND gate feeding into a 3-input OR gate. Two of the AND gates have inverted inputs.

NOR Gates and NAND Gates

- Combine basic AND, OR, and NOT operations.
- The NOR gate is an inverted OR gate. An inversion “bubble” is placed at the output of the OR gate.
- The Boolean expression is $x = \overline{A+B}$

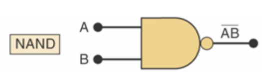


A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

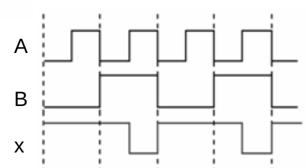


NOR Gates and NAND Gates

- The NAND gate is an inverted AND gate. An inversion “bubble” is placed at the output of the AND gate.
- The Boolean expression is $x = \overline{AB}$



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



Boolean Theorems

The theorems or laws below may represent an expression containing more than one variable.

$$x \cdot 0 = 0 \qquad x \cdot 1 = 1$$

$$x \cdot x = x \qquad x \cdot \overline{x} = 0$$

$$x + 1 = 1 \qquad x + x = x$$

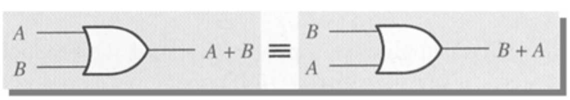
$$x + \overline{x} = 1$$

Laws of Boolean Algebra

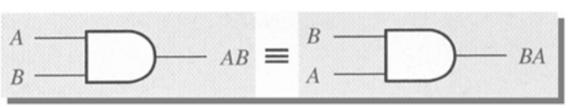
- Commutative Laws
- Associative Laws
- Distributive Law

Commutative Laws of Boolean Algebra

A + B = B + A

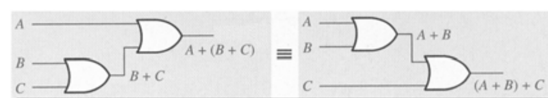


A • B = B • A

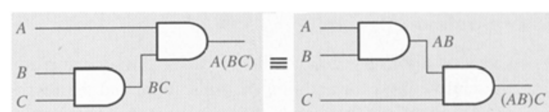


Associative Laws of Boolean Algebra

A + (B + C) = (A + B) + C



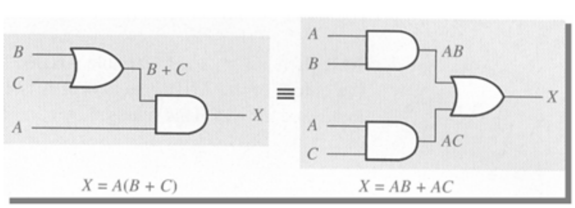
A • (B • C) = (A • B) • C



Distributive Laws of Boolean Algebra

A • (B + C) = A • B + A • C

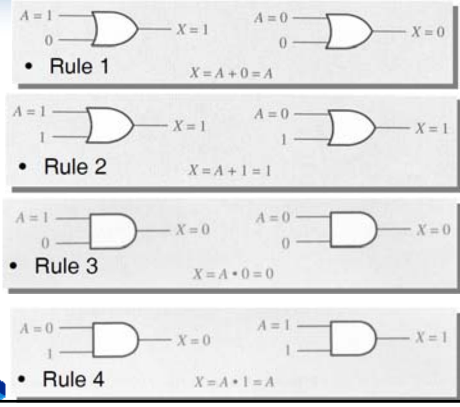
A (B + C) = A B + A C



Rules of Boolean Algebra

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

Rules of Boolean Algebra



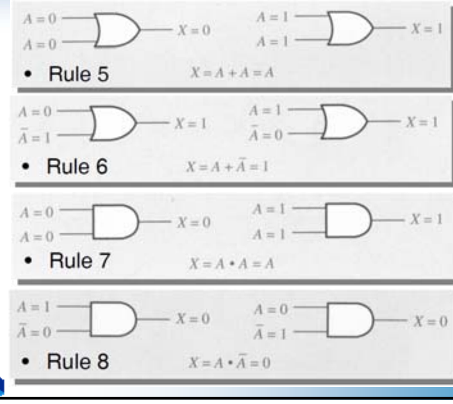
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

OR Truth Table

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

AND Truth Table

Rules of Boolean Algebra



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

OR Truth Table

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

AND Truth Table

Rules of Boolean Algebra

- Rule 10: A + AB = A

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

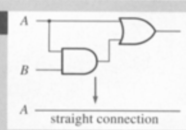
AND Truth Table

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

OR Truth Table

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

equal

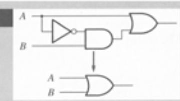


Rules of Boolean Algebra

- Rule 11: A + A’B = A +B

A	B	$\overline{A}B$	$A + \overline{A}B$	A + B
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

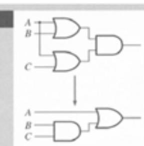
equal



- Rule 12: (A + B)(A + C) = A + BC

A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

equal



DeMorgan’s Theorems

- Theorem 1: When the OR sum of two variables is inverted, it is equivalent to inverting each variable individually and ANDing them.

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

- Theorem 2: When the AND product of two variables is inverted, it is equivalent to inverting each variable individually and ORing them.

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

DeMorgan’s Theorems

- A NOR gate is equivalent to an AND gate with inverted inputs.
- A NAND gate is equivalent to an OR gate with inverted inputs.

For N variables, DeMorgan’s theorem is expressed as:

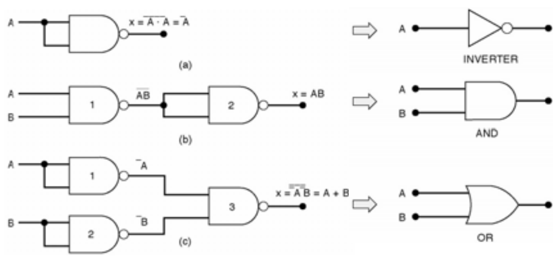
$\overline{ABC...N} = \overline{A} + \overline{B} + \overline{C} + ... + \overline{N}$ and $\overline{A + B + C + ... + N} = \overline{A}\,\overline{B}\,\overline{C}....\,\overline{N}$

Applying Demorgan’s theorem to the expression $\overline{ABC + DEF}$ we get:

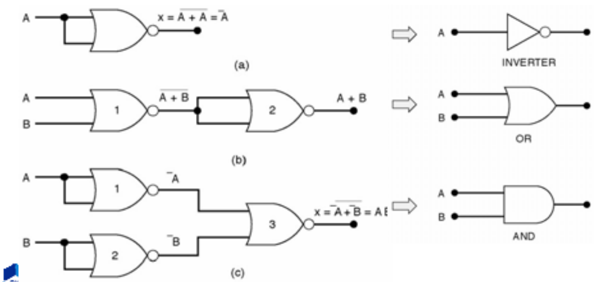
$$\overline{ABC + DEF} = (\overline{ABC})(\overline{DEF}) = (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$$

Universality of NAND and NOR Gates

- NAND or NOR gates can be used to create the three basic logic expressions (OR, AND, and INVERT)



Universality of NAND and NOR Gates

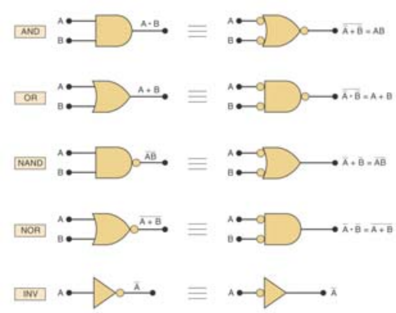


Alternate Logic-Gate Representations

- To convert a standard symbol to an alternate:
  - Invert each input and output (add an inversion bubble where there are none on the standard symbol, and remove bubbles where they exist on the standard symbol.
  - Change a standard OR gate to and AND gate, or an AND gate to an OR gate.

Alternate Logic-Gate Representations

- Standard and alternate symbols for various logic gates and inverter



Alternate Logic-Gate Representations

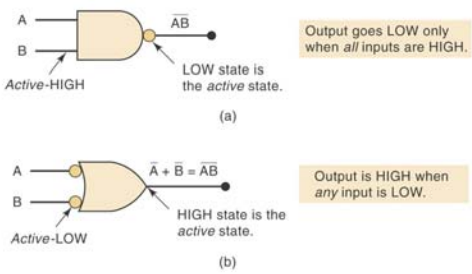
- The equivalence can be applied to gates with any number of inputs.
- No standard symbols have bubbles on their inputs. All of the alternate symbols do.
- The standard and alternate symbols represent the same physical circuitry.

Alternate Logic-Gate Representations

- Active high – an input or output has no inversion bubble.
- Active low – an input or output has an inversion bubble.
- An AND gate will produce an active output when all inputs are in their active states.
- An OR gate will produce an active output when any input is in an active state.

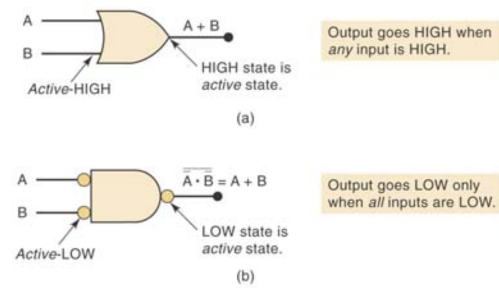
Alternate Logic-Gate Representations

- Interpretation of the two NAND gate symbols.



Alternate Logic-Gate Representations

- Interpretation of the two OR gate symbols.

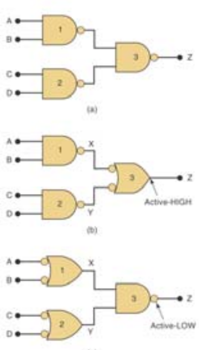


Which Gate Representation to Use

- Using alternate and standard logic gate symbols together can make circuit operation clearer.
- When possible choose gate symbols so that bubble outputs are connected to bubble input and nonbubble outputs are connected to nonbubble inputs.

Which Gate Representation to Use

- When a logic signal is in the active state (high or low) it is said to be asserted.
- When a logic signal is in the inactive state (high or low) it is said to be unasserted.
- A bar over a signal means asserted (active) low.
- The absence of a bar over a signal means asserted (active) high.



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)

(a) Original circuit using standard NAND symbols; (b) equivalent representation where output Z is active-HIGH; (c) equivalent representation where output Z is active-LOW; (d) truth table.

IEEE/ANSI Standard Logic Symbols

- Rectangular symbols represent logic gates and circuits.
- Dependency notation inside symbols show how output depends on inputs.
- A small triangle replaces the inversion bubble.