Digital Systems

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

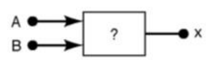
- A truth table describes the relationship between the input and output of a logic circuit.
- The number of entries corresponds to the number of inputs. For example a 2-input table would have $2^2 = 4$ entries. A 3-input table would have $2^3 = 8$ entries.

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

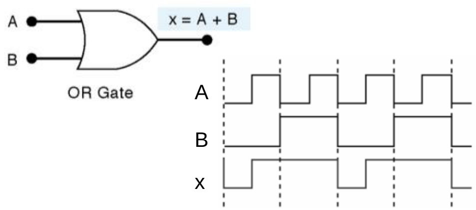
(c)

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

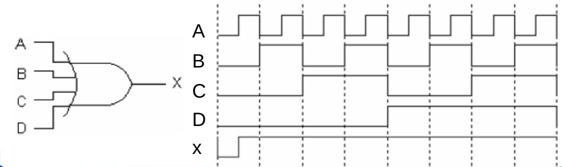
A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1



OR Operation With OR Gates

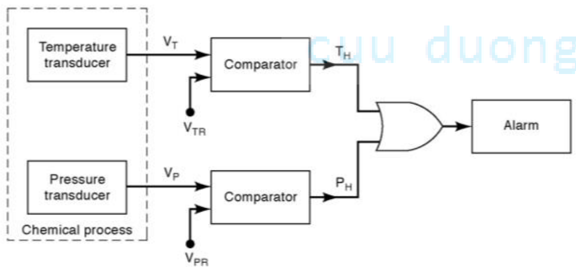
- The OR operation is similar to addition but when A = 1 and B = 1, the OR operation produces 1 + 1 = 1.
- In the Boolean expression
  x=1+1+1+1=1
  We could say that x is true (1) when A is true (1) OR B is true (1) OR C is true (1) OR D is true (1).
- In general, the output of an OR gate is HIGH whenever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



Review Questions

- What is the only set of input conditions that will produce a LOW output for any OR gate?
  - all inputs LOW
- Write the Boolean expression for a six-input OR gate
  - X=A+B+C+D+E+F
- If the A input in previous example is permanently kept at the 1 level, what will the resultant output waveform be?
  - constant HIGH

AND Operations with AND gates

- The Boolean expression for the AND operation is

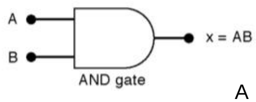
  X = A • B

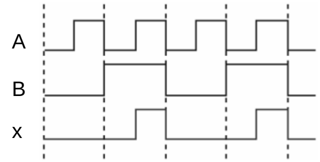
  – This is read as "x equals A and B."

  – 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



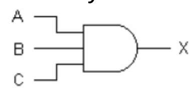


AND Operation With AND Gates

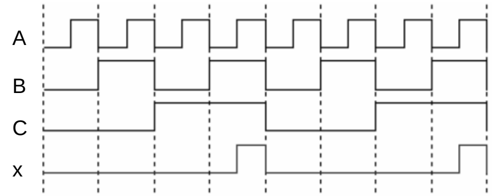
- The AND operation is similar to multiplication.
- In the Boolean expression

  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



Review Questions

- What is the only input combination that will produce a HIGH at the output of a five-input AND gate?
  – all 5 inputs = 1
- What logic level should be applied to the second input of a two-input AND gate if the logic signal at the first input is to be inhibited(prevented) from reaching the output?
  – A LOW input will keep the output LOW
- True or false: An AND gate output will always differ from an OR gate output for the same input conditions.
  – False

NOT Operation

- The Boolean expression for the NOT operation is

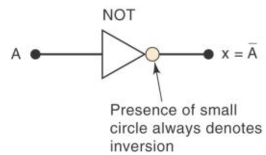
$$X = \overline{A} \qquad X = A'$$

- 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.

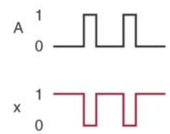
NOT

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

(a)



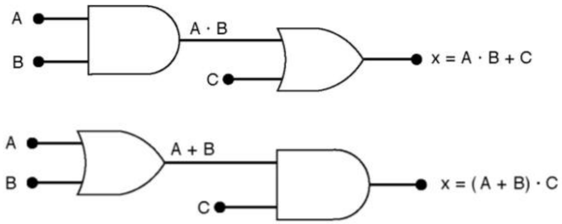
(b)



(c)

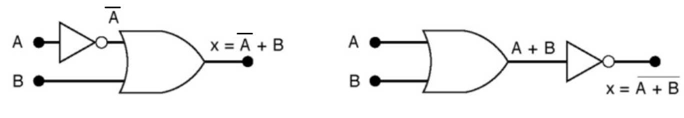
Describing Logic Circuits Algebraically

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



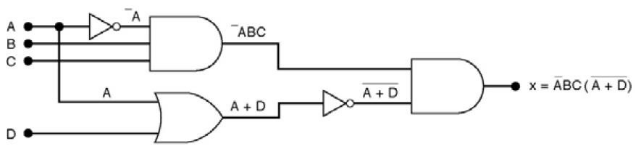
Describing Logic Circuits Algebraically

- 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.

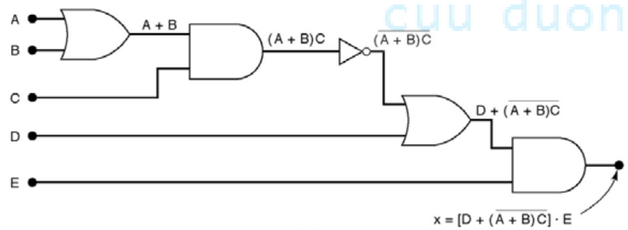


(a) (b)

More Examples



(a)



(b)

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:

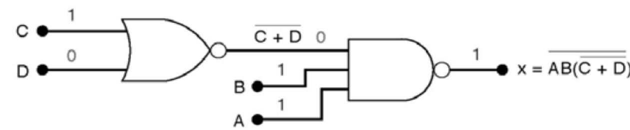
$$A = 0, B = 1, C = 1, \text{and } D = 1$$

$$\begin{aligned} x &= \overline{A}BC\overline{(A+D)} \\ &= \overline{0}\cdot 1\cdot 1\cdot \overline{(0+1)} \\ &= 1\cdot 1\cdot 1\cdot \overline{(0+1)} \\ &= 1\cdot 1\cdot 1\cdot \overline{(1)} \\ &= 1\cdot 1\cdot 1\cdot 0 \\ &= 0 \end{aligned}$$

17

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.



18

Implementing Circuits From Boolean Expressions

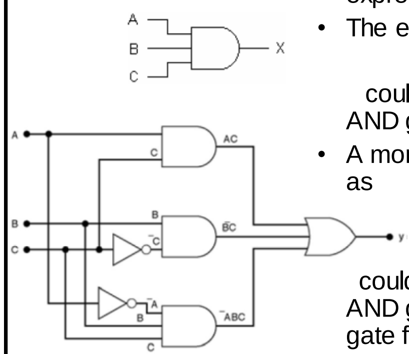
- It is important to be able to draw a logic circuit from a Boolean expression.
- The expression $x = A\cdot B\cdot C$

could be drawn as a three input AND gate.

- A more complex example such as

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

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.

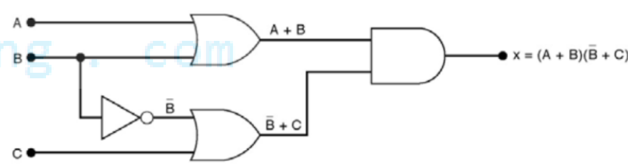


19

Example

- Draw the circuit diagram to implement the expression

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



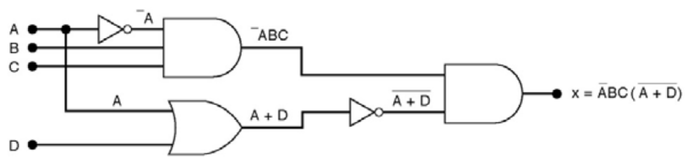
20

Review Question

- Draw the circuit diagram that implements the expression

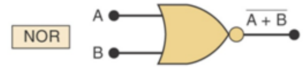
$$x = \overline{A}BC(\overline{A+D})$$

using gates having no more than three inputs.

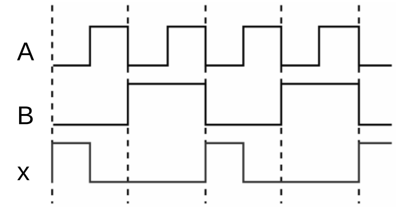


NOR Gates and NAND Gates

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

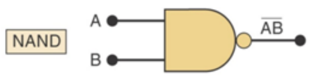


A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

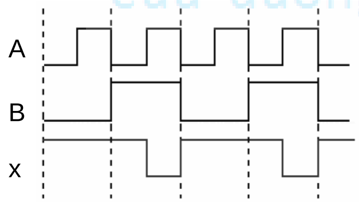


NOR Gates and NAND Gates

- 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

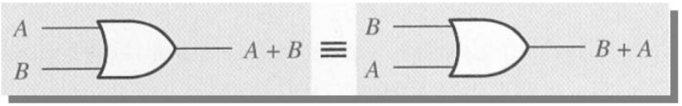


Laws of Boolean Algebra

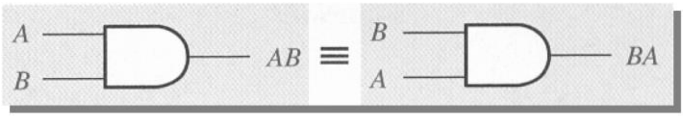
- Commutative Laws
- Associative Laws
- Distributive Law

Commutative Laws of Boolean Algebra

$$A + B = B + A$$



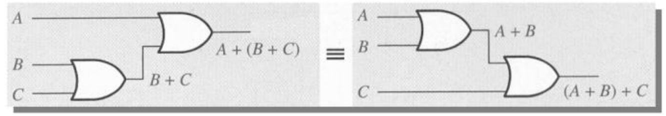
$$A \cdot B = B \cdot A$$



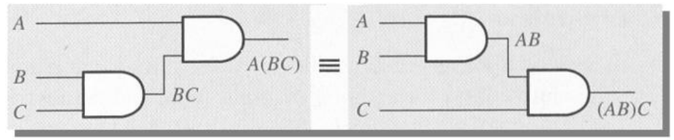
25

Associative Laws of Boolean Algebra

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



$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

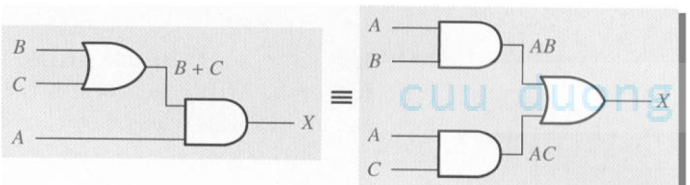


26

Distributive Laws of Boolean Algebra

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

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



$X = A(B + C)$ $\qquad$ $X = AB + AC$

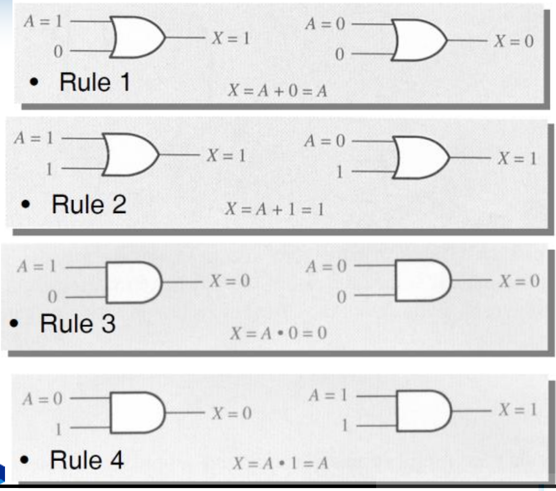
27

Rules of Boolean Algebra

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

28

Rules of Boolean Algebra



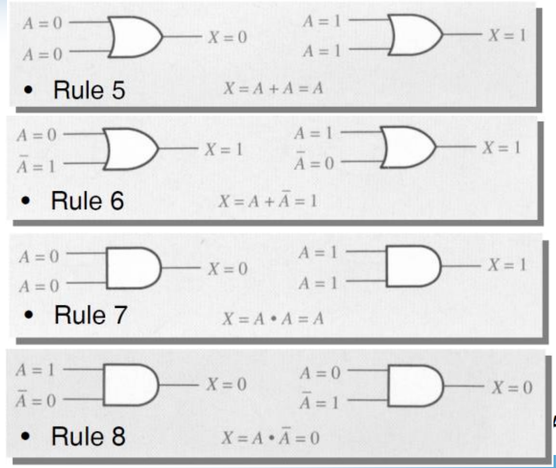
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

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



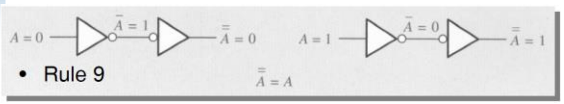
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

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

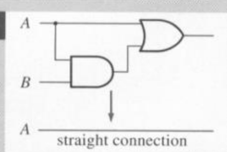
AND Truth Table

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

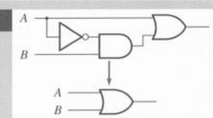


Rules of Boolean Algebra

- 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

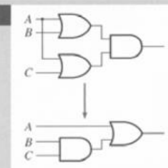
equal



- 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



Examples

- Simplify the expression

$$y = A\overline{B}D + A\overline{B}\overline{D}$$

$$y = A\overline{B}$$

$$z = (\overline{A} + B)(A + B)$$

$$z = B$$

$$x = ACD + \overline{A}BCD$$

$$x = ACD + BCD$$

$$y = A\overline{C} + AB\overline{C}$$

$$y = A\overline{C}$$

33

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

34

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:

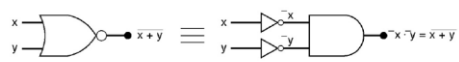
$\overline{ABC\ldots N} = \overline{A} + \overline{B} + \overline{C} + \ldots + \overline{N}$ and $\overline{A + B + C + \ldots + N} = \overline{A}\,\overline{B}\,\overline{C}\ldots\,\overline{N}$

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

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

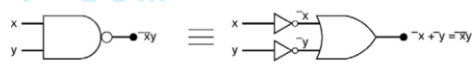
35

Implications of DeMorgan's Theorems

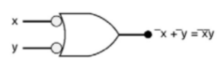


(a)

(b)



(a)

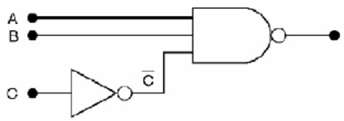


(b)

36

Implications of DeMorgan's Theorems

- Determine the output expression for the below circuit and simplify it using DeMorgan's Theorem



$$z = \overline{A \cdot B \cdot \overline{C}} = \overline{A} + \overline{B} + \overline{\overline{C}} = \overline{A} + \overline{B} + C$$

- Use DeMorgan's theorems to convert below expression to an expression containing only single-variable inversions

$$y = \overline{A + \overline{B} + \overline{C}D}$$

$$y = \overline{A}B(C + \overline{D})$$

Example of DeMorgan's Theorems

$$F = \overline{XY} + \overline{P.Q}$$

$$F = \overline{X} + \overline{Y} + \overline{P} + \overline{Q}$$

- Simplify the expression

$$z = \overline{(\overline{A} + C)(B + \overline{D})}$$

- to one having only single variables inverted.

$$z = A\overline{C} + \overline{B}D$$

Examples

- Simplify the expressions
  - z = (A' + B)(A+B)
- De Morgan's
  - z = ((a'+c) . (b+d'))'

Examples

- Simplify the expressions
  - z = (A' + B)(A+B)
    = A'A + A'B + AB + BB = 0 + (A'+A)B + B = B
- De Morgan's
  - z = ((a'+c) . (b+d'))'
    = (a'+c)' + (b+d')' = ac' + b'd

Exercises

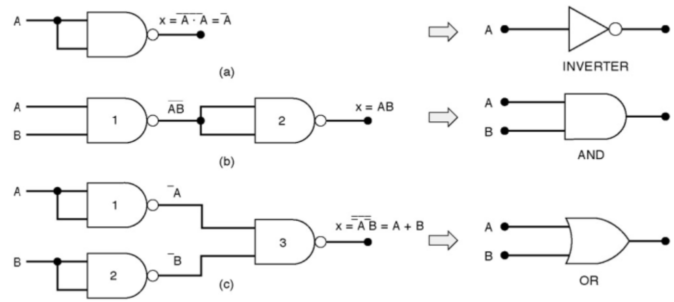
- Simplify the expressions
  - a) $x = (M + N)(\overline{M} + P)(\overline{N} + \overline{P})$
  - b) $z = \overline{A}B\overline{C} + AB\overline{C} + B\overline{C}D$

- De Morgan's

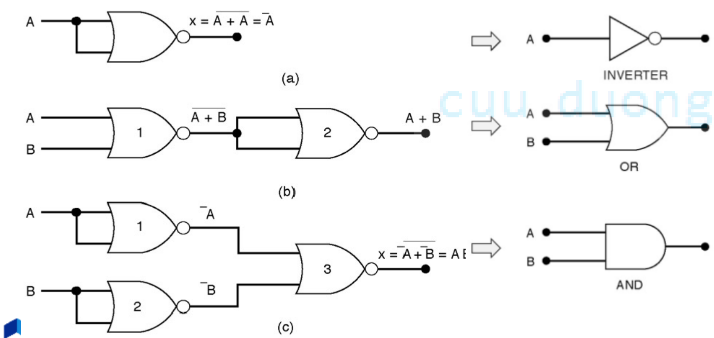
(a)* $\overline{\overline{A}B\overline{C}}$	(d) $\overline{A + \overline{B}}$	(g)* $\overline{A\overline{(B + \overline{C})}D}$
(b) $\overline{\overline{A} + \overline{B}C}$	(e)* $\overline{\overline{AB}}$	(h) $\overline{(M + \overline{N})(\overline{M} + N)}$
(c)* $\overline{AB\overline{CD}}$	(f) $\overline{\overline{A} + \overline{C} + \overline{D}}$	(i) $\overline{\overline{\overline{AB}C}D}$

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

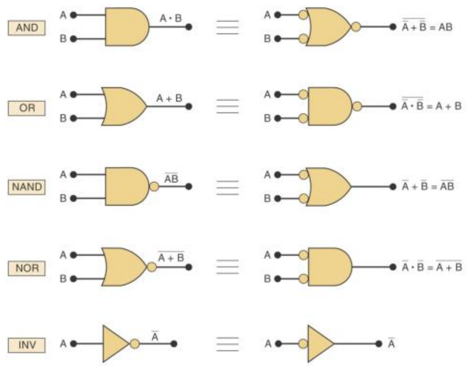


Alternate Logic-Gate Representations

- 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



45

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.

46

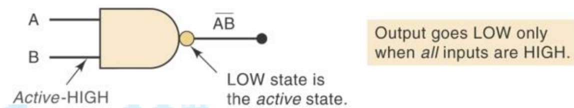
Alternate Logic-Gate Representations

- 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.

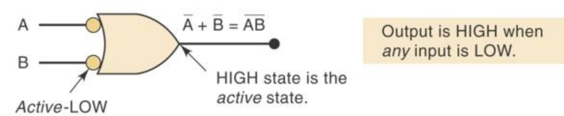
47

Alternate Logic-Gate Representations

- Interpretation of the two NAND gate symbols.



(a)

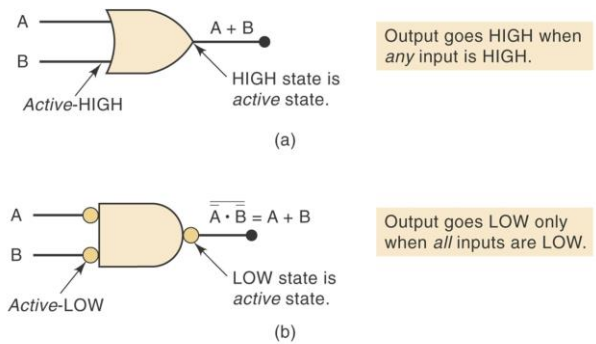


(b)

48

Alternate Logic-Gate Representations

- Interpretation of the two OR gate symbols.

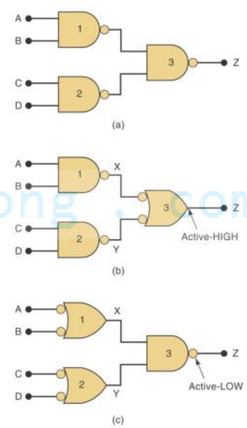


Which Gate Representation to Use

- 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.

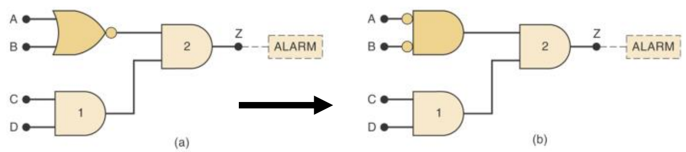


A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(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.

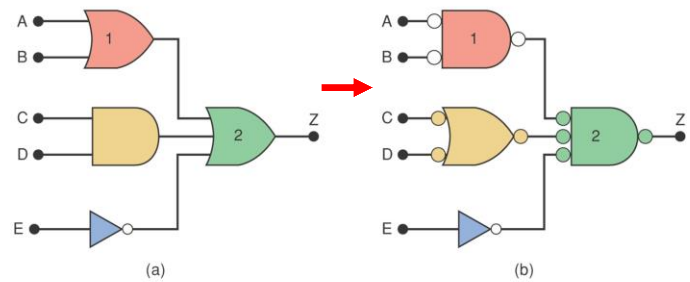
Example



Alarm is activated when Z goes high. Modify the circuit so that it represents the circuit operation more effectively.

Example

Z activates another circuit when it goes low. Convert Z to Active-Low



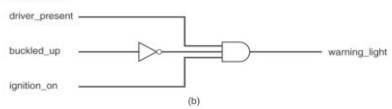
Methods of describing logic circuits

Boolean expression

warning_light = driver_present • $\overline{\text{buckled\_up}}$ • ignition_on

(a)

Schematic diagram



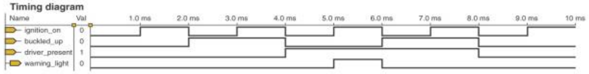
(b)

Truth table

driver_present	buckled_up	ignition_on	warning_light
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

(c)

Timing diagram

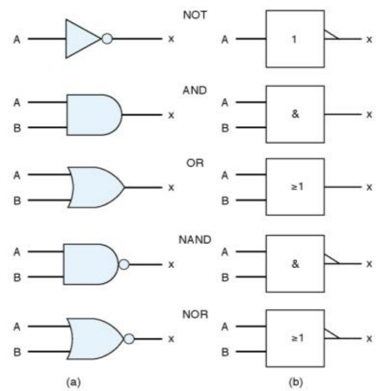


(d)

(a) Boolean expression; (b) schematic diagram; (c) truth table; (d) timing diagram.

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.



Summary of Methods to Describe Logic Circuits

- The three basic logic functions are AND, OR, and NOT.
- Logic functions allow us to represent a decision process.
  - If it is raining OR it looks like rain I will take an umbrella.
  - If I get paid AND I go to the bank I will have money to spend.

57

Summary

- Boolean Algebra: a mathematical tool used in the analysis and design of digital circuits
- OR, AND, NOT: basic Boolean operations
- OR: HIGH output when any input is HIGH
- AND: HIGH output only when all inputs are HIGH
- NOT: output is the opposite logic level as the input
- NOR: OR with its output connected to an INVERTER
- NAND: AND with its output connected to an INVERTER
- Boolean theorems and rules: to simplify the expression of a logic circuit and can lead to a simpler way of implementing the circuit
- NAND, NOR: can be used to implement any of the basic Boolean operations

58