Imperial College London

Lecture 4: Boolean Algebra

Professor Peter Cheung
Department of EEE, Imperial College London

(Floyd 4.1-4.4, 5.2-5.4)
(Tocci 3.8-3.14)

Imperial College London

Points Addressed in this Lecture

- Theorems & rules in Boolean algebra
- DeMorgan's Theorems
- Universality of NAND & NOR gates
- Active low & Active high
- Digital Integrated Circuits

Imperial College London

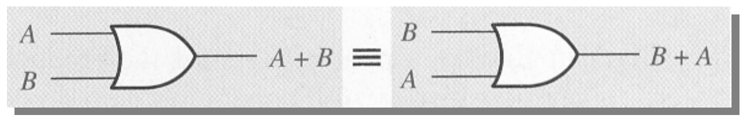
Laws of Boolean Algebra

- Commutative Laws
- Associative Laws
- Distributive Law

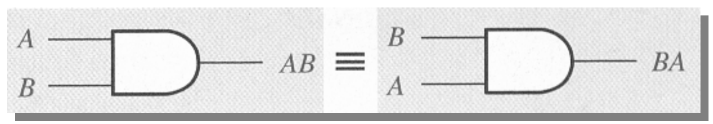
Imperial College London

Commutative Laws of Boolean Algebra

A + B = B + A



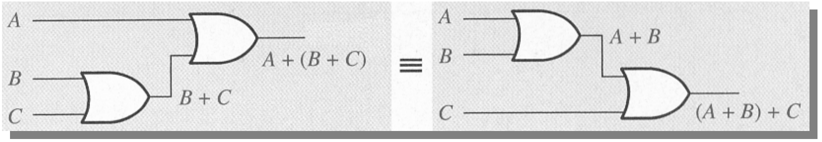
A • B = B • A



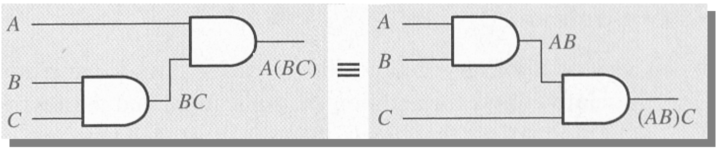
Imperial College London

Associative Laws of Boolean Algebra

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



A • (B • C) = (A • B) • C

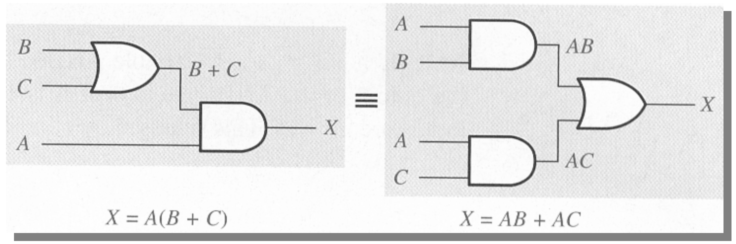


Imperial College London

Distributive Laws of Boolean Algebra

A • (B + C) = A • B + A • C

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



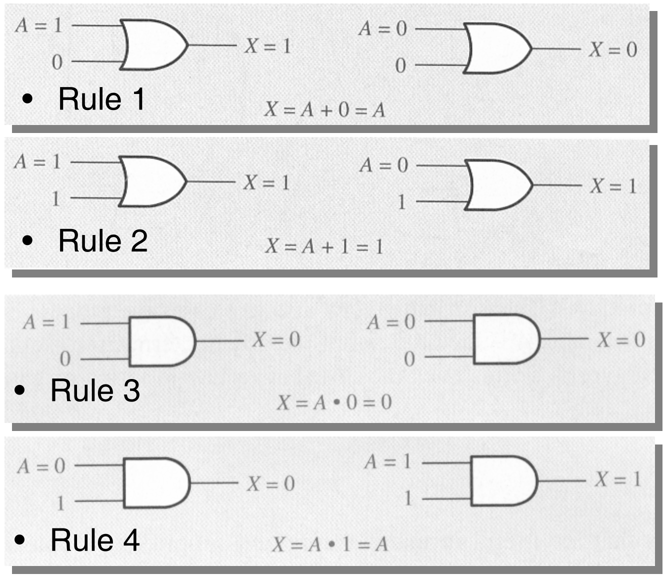
Imperial College London

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

Imperial College London

Rules of Boolean Algebra



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

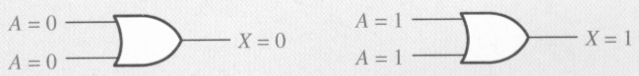
OR Truth Table

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

AND Truth Table

Imperial College London

Rules of Boolean Algebra



- Rule 5 $X = A + A = A$

- Rule 6 $X = A + \overline{A} = 1$

- Rule 7 $X = A \bullet A = A$

- Rule 8 $X = A \bullet \overline{A} = 0$

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

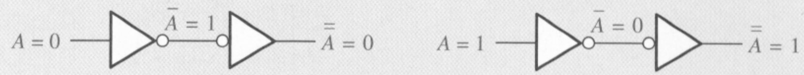
OR Truth Table

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

AND Truth Table

Imperial College London

Rules of Boolean Algebra



- Rule 9 $\overline{\overline{A}} = A$

- Rule 10: A + AB = A

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

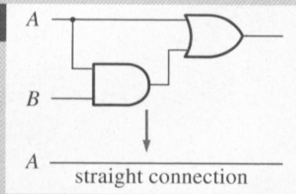
AND Truth Table

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 (A and A + AB)



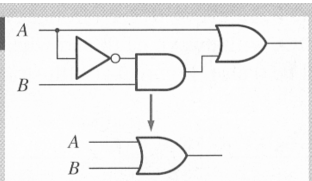
Imperial College London

Rules of Boolean Algebra

- Rule 11: $A + \overline{A}B = A + B$

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

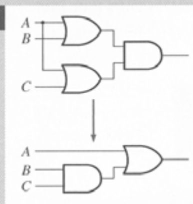
equal ($A + \overline{A}B$ and A + B)



- 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 ((A + B)(A + C) and A + BC)



Imperial College London

Examples

- Simplify the expression $y = A\overline{B}D + A\overline{B}\,\overline{D}$

$y = A\overline{B}$

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

$z = B$

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

$x = ACD + BCD$

Review Questions

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

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

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

  $\Rightarrow y = \overline{A}\,\overline{B}\,\overline{D}$

- Simplify $y = \overline{A}D + ABD$

  $\Rightarrow y = \overline{A}D + BD$

DeMorgan's Theorems

- Theorem 1

$$\overline{(x+y)} = \overline{x}\cdot\overline{y}$$

- Theorem 2

$$\overline{(x\cdot y)} = \overline{x}+\overline{y}$$

Remember:

"Break the bar, change the operator"

- DeMorgan's theorem is very useful in digital circuit design
- It allows **ANDs to be exchanged with ORs by using invertors**
- DeMorgan's Theorem can be extended to any number of variables.

Example of DeMorgan's Theorem

$F = \overline{X.Y} + \overline{P.Q}$ ← 2 NAND plus 1 OR

$= \overline{X} + \overline{Y} + \overline{P} + \overline{Q}$ ← 1 OR with some input invertors

Example

- Simplify the expression

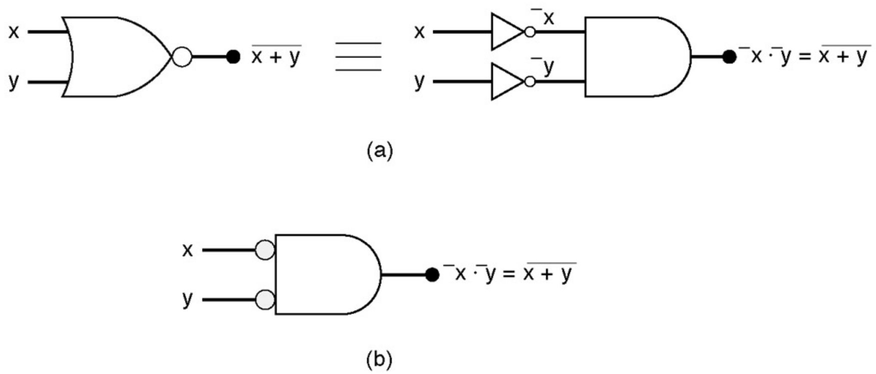
$$z = \overline{(\overline{A}+C)\cdot(B+\overline{D})}$$

to one having only single variables inverted.

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

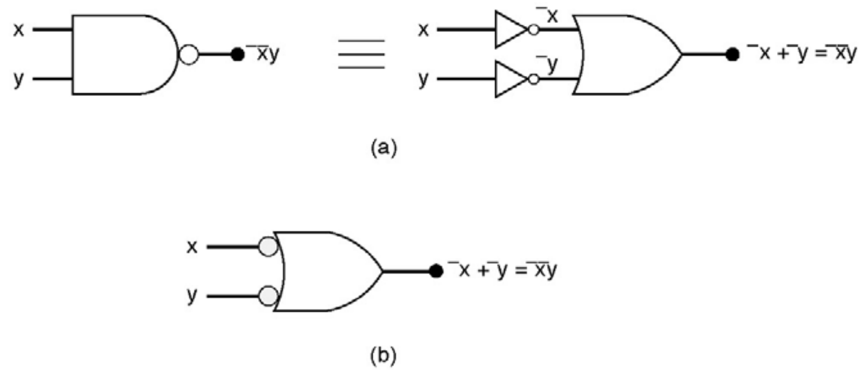
Imperial College London

Implications of DeMorgan's Theorems(I)



Imperial College London

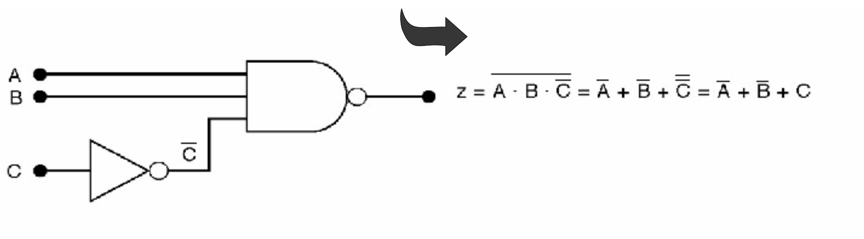
Implications of DeMorgan's Theorems(II)



Imperial College London

Example

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



Imperial College London

Review Questions

- Using DeMorgan's Theorems to convert the expressions to one that has only single-variable inversions.

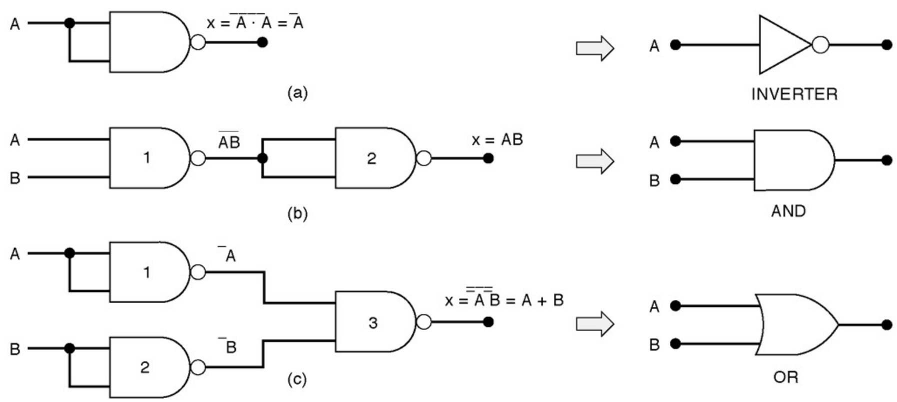
$$y=\overline{R\bar{S}T+\bar{\bar{Q}}} \rightarrow y=(\bar{R}+S+\bar{T})Q$$

$$z=\overline{(A+B)\cdot\bar{C}} \rightarrow z=\bar{A}\bar{B}+C$$

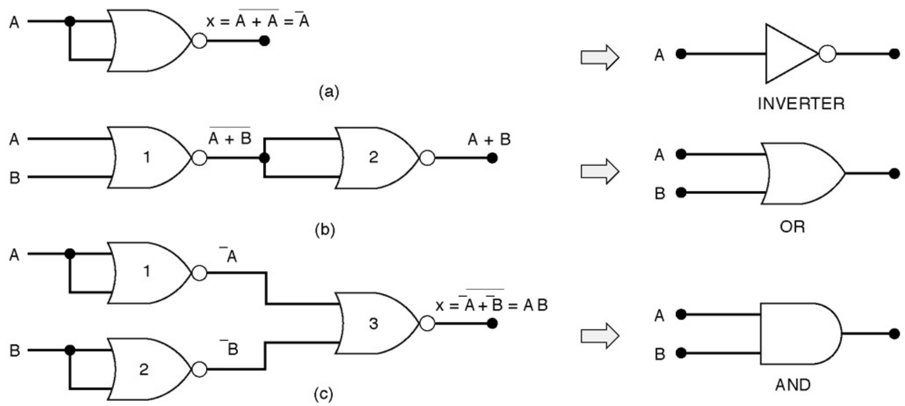
- Use DeMorgan's theorems to convert below expression to an expression containing only single-variable inversions.

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

Universality of NAND gates

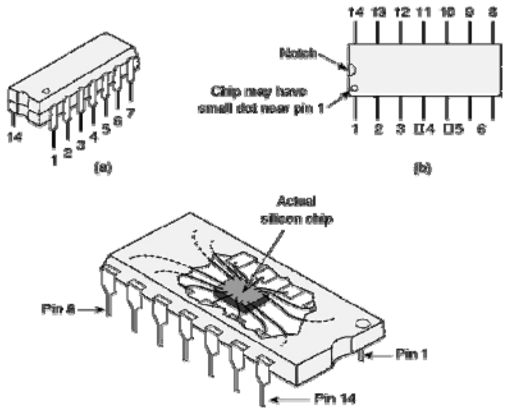


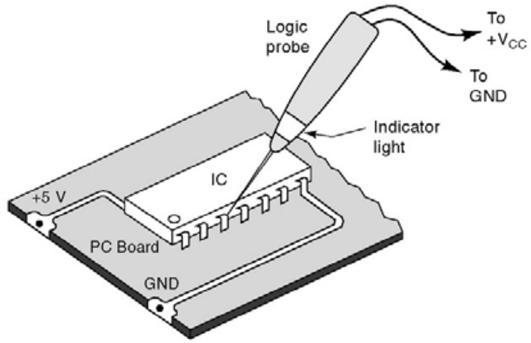
Universality of NOR gate



Basic Characteristics of Digital ICs

- Digital ICs (chips): a collection of resistors, diodes and transistors fabricated on a single piece of semiconductor materials called substrate.
- Dual-in-line package (DIP) is a common type of packages.



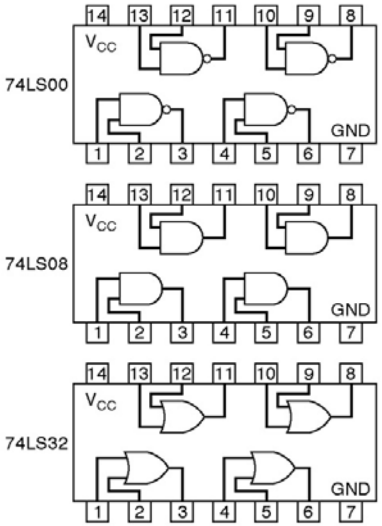


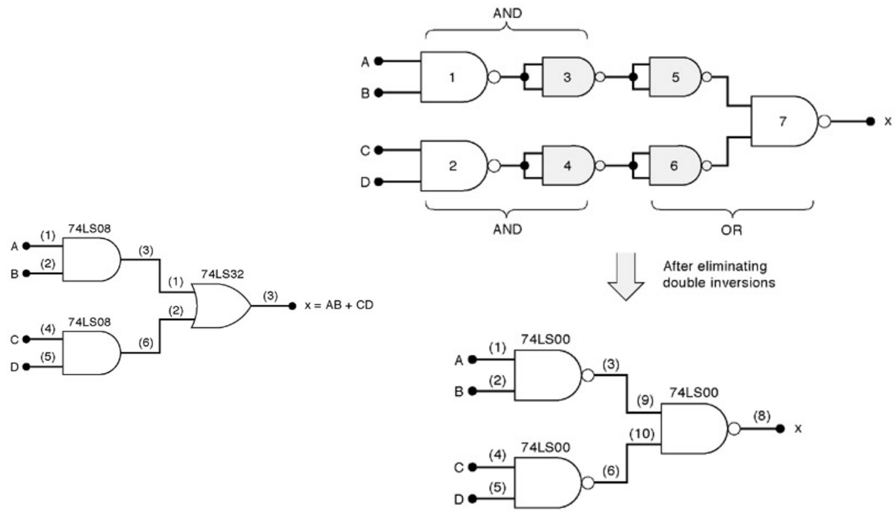
Indicator Light	Logic Level
OFF	LOW
ON (bright)	HIGH
DIM	INDETERMINATE*
FLASHING	PULSING

* Includes open or floating condition

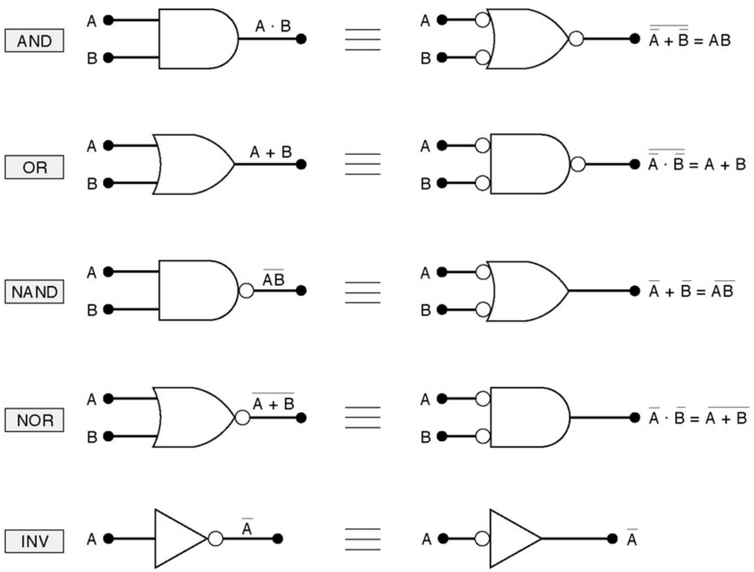
Logic probe is used to monitor the logic level activity at an IC pin or any other accessible point in a logic circuit

Example





Alternate Logic-Gate Representations



Obtain alternative symbol from standard ones

- Invert each input and output of the standard symbol. This is done by adding bubbles(small circles) on input and output lines that do not have bubbles and by removing bubbles that are already there.
- Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER, the operation symbol is not changed.)

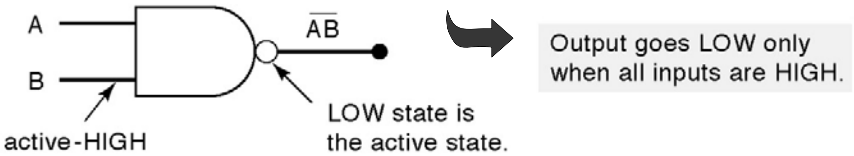
Several points

- The equivalences can be extended to gates with any number of inputs.
- None of the standard symbols have bubbles on their inputs, and all the alternate symbols do.
- The standard and alternate symbols for each gate represent the same physical circuit; there is no difference in the circuits represented by the two symbols.
- NAND and NOR gates are inverting gates, and so both the standard and the alternate symbols for each will have a bubble on either the input or the output, AND and OR gates are non-inverting gates, and so the alternate symbols for each will have bubbles on both inputs and output.

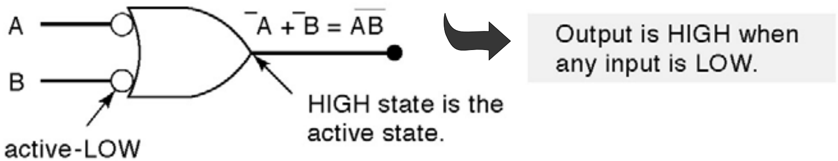
Logic-symbol interpretation

- Active high/low
  - When an input or output line on a logic circuit symbol has no bubble on it, that line is said to be **active-high**, otherwise it is **active-low**.

Interpretation of the two NAND gate symbols

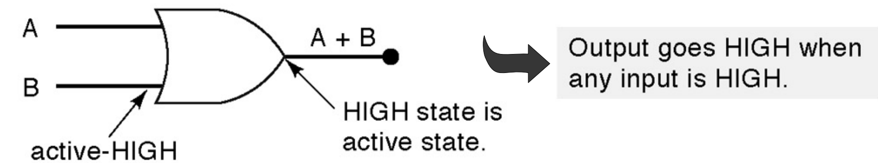


(a)

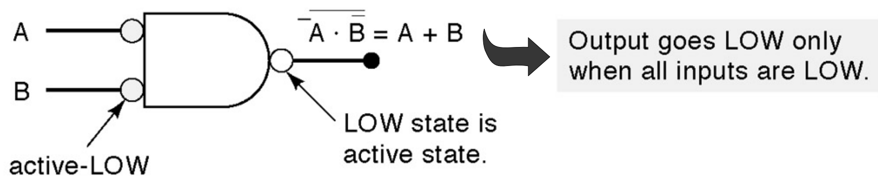


(b)

Interpretation of the two OR gate symbols

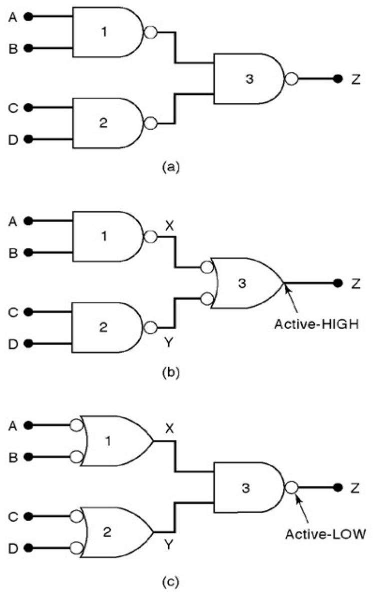


(a)



(b)

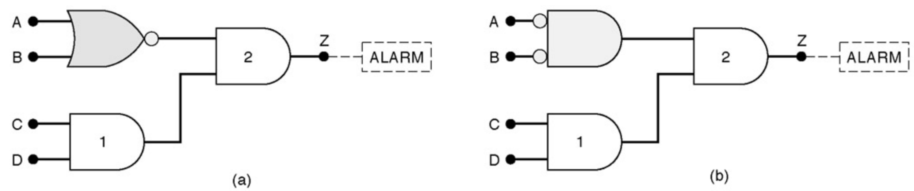
Equivalency of three circuits



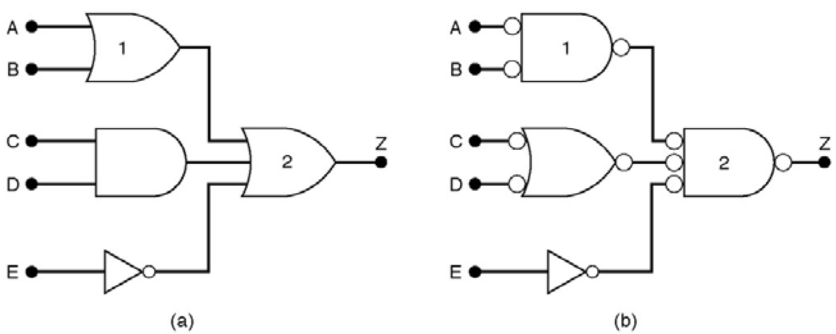
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- Which circuit diagram should be used?
- Bubble placement

  Whenever possible, choose gate symbols so that bubble outputs are connected to bubble inputs, and nonbubble outputs to nonbubble inputs



Active High & Active Low Representations



- When a logic signal is in its active state, it can be said to be **asserted <u>(=activated)</u>**. When a logic signal is not in its active state, it is said to be **<u>unasserted (=inactive)</u>**.

- The overbar is simply a way to emphasize that these are active-LOW signals.

Summary: How to represent the basic logic functions

- Logical statements in our own language
- Truth tables
- Traditional graphic logic symbols
- Boolean algebra expressions
- Timing diagrams

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