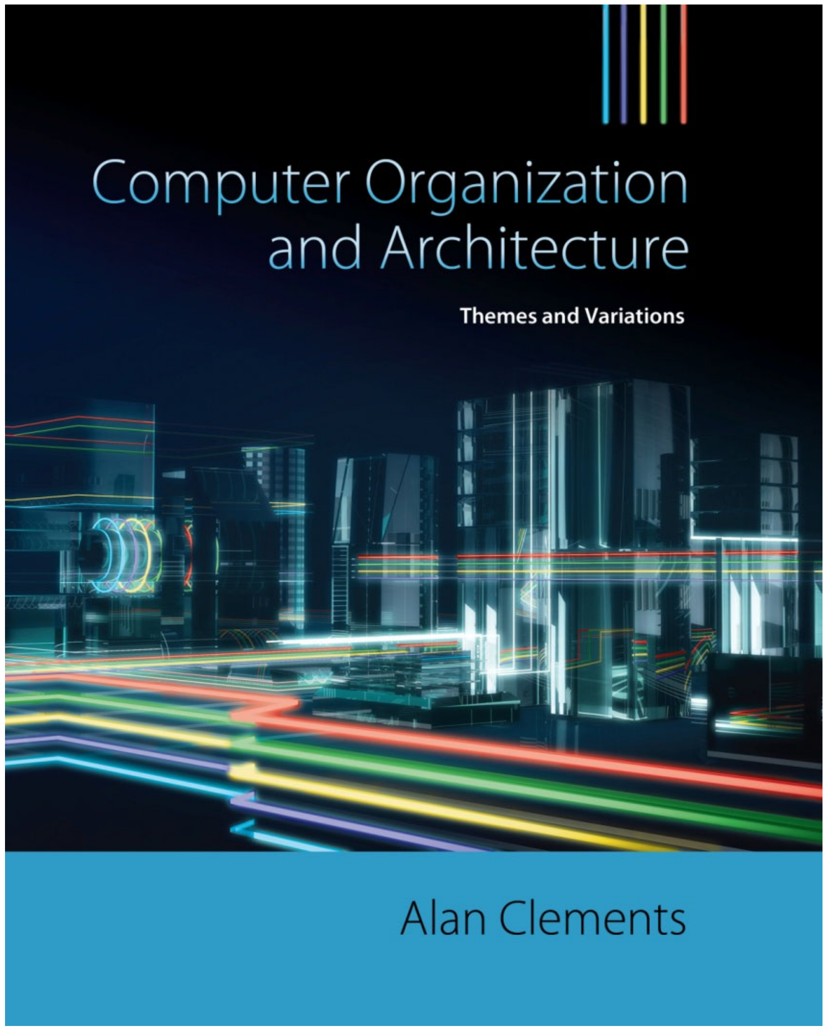
Part 4

Chapter 2

Computer Arithmetic and Digital Logic



CENGAGE Learning

Computer Logic

- Computers are constructed from two basic circuit elements — *gates* and *flip-flops*, known as *combinational* and *sequential* logic elements.

- A *combinational* logic element is a circuit whose output depends only on its current inputs,

  whereas

  A *sequential* logic element is a circuit whose output depends on its past history as well as its current inputs.

- A *sequential* element can *remember* its previous value (*memory element*).
- *Sequential* elements themselves can be made from simple combinational logic elements.
  - Hence, we can simply say that *computers can be constructed using just gates*

Logic Values

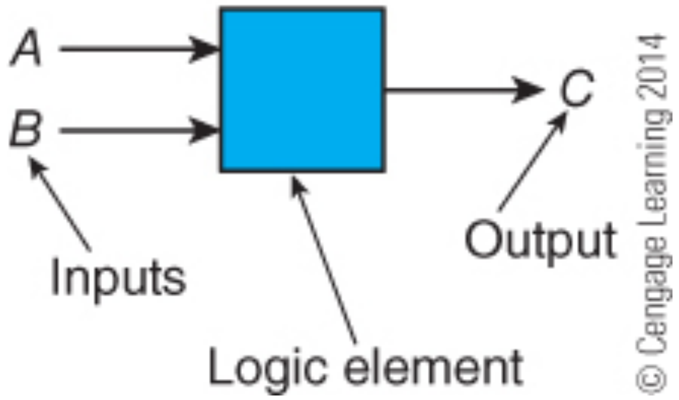
- A ***logic value*** can be either
  - the *logical 1* (also called the *true* or *high* state)
  - the *logical 0* (also called the *false* or *low* state)
- Each logic state has an ***inverse*** or a ***complement*** that is the opposite of its current state.
  - The complement of a *true* or *1* state is a *false* or *0* state
  - The complement of a *false* or *0* state is a *true* or *1* state

Gates

- Figure 2.14 shows a black box of a gate with two *input* terminals, *A* and *B*, and a single *output* terminal *C*.
  - This gate takes the *two logic values* at its input terminals and
  - produces *a logic output value* that ***depends only on***
    - the *states of the inputs* and
    - the *nature of the logic element*.

FIGURE 2.13

The logic element



- Examples of gates include
  - **AND**, **OR**, **NOT** , **NAND**, **NOR** , **Exclusive OR** gates.

Gates

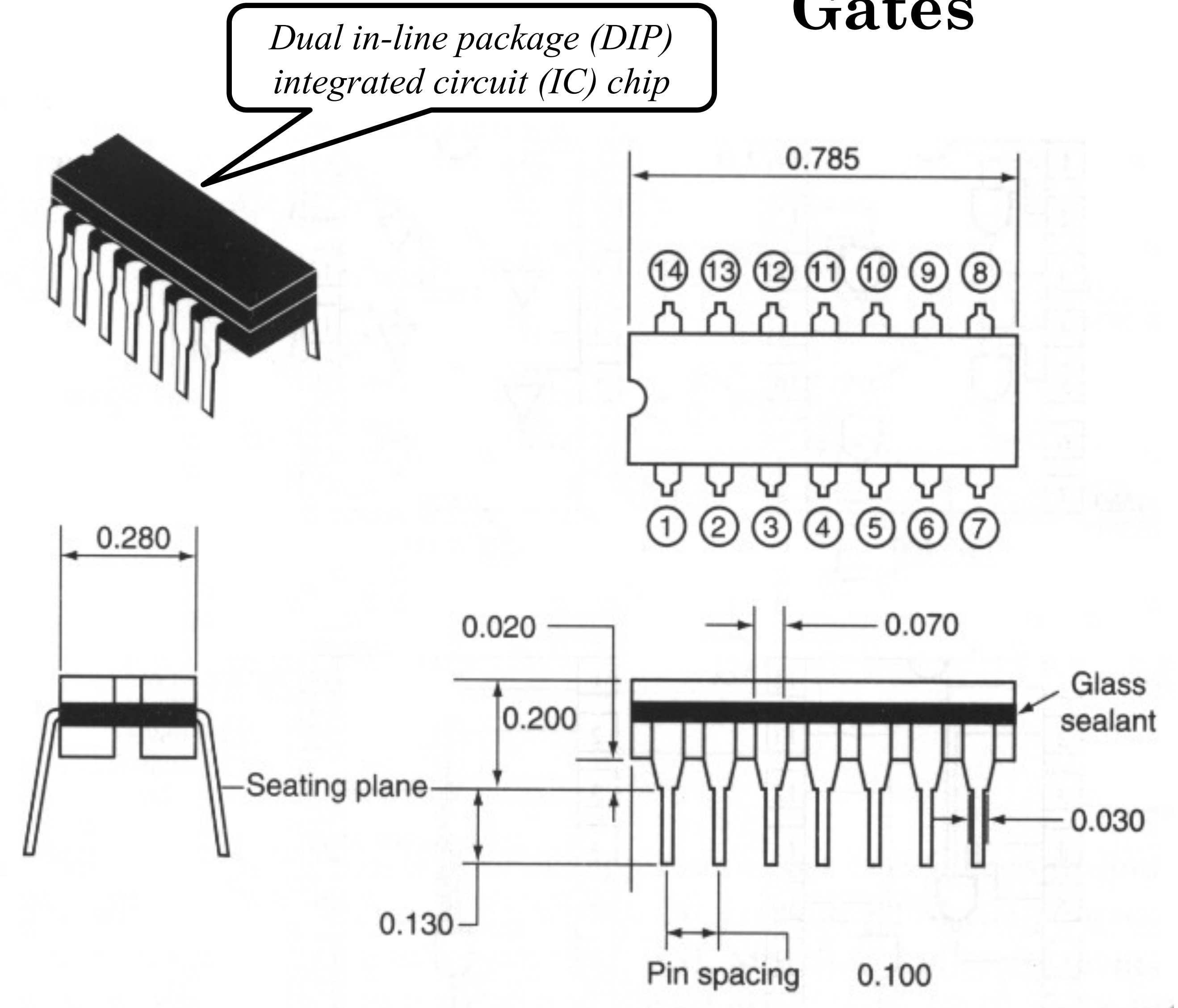
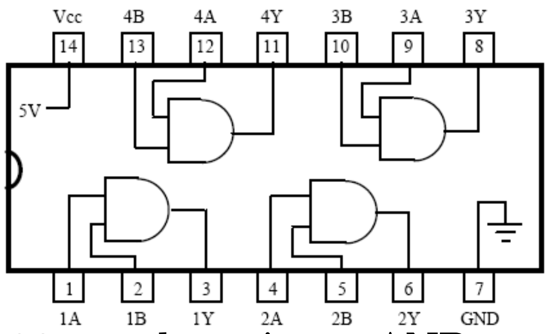
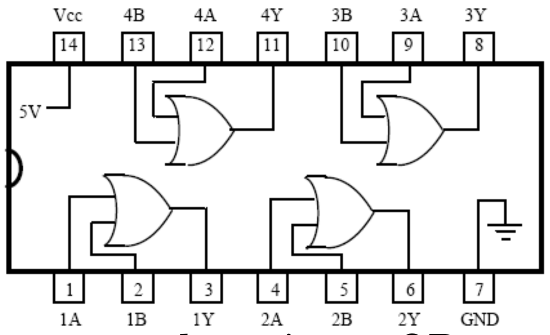


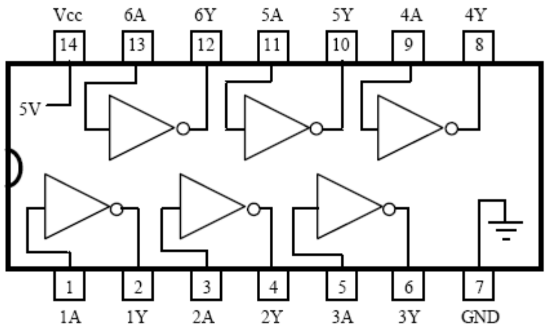
FIGURE 3.17
14-pin ceramic package (all linear dimensions are in inches).



7408: quad two input AND gates

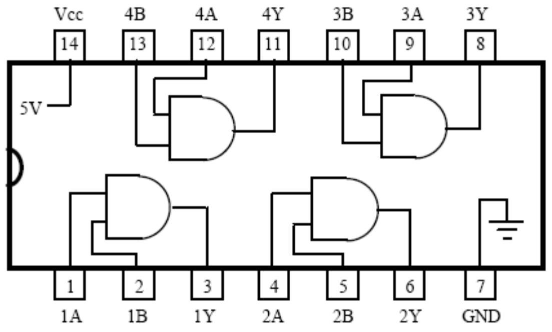


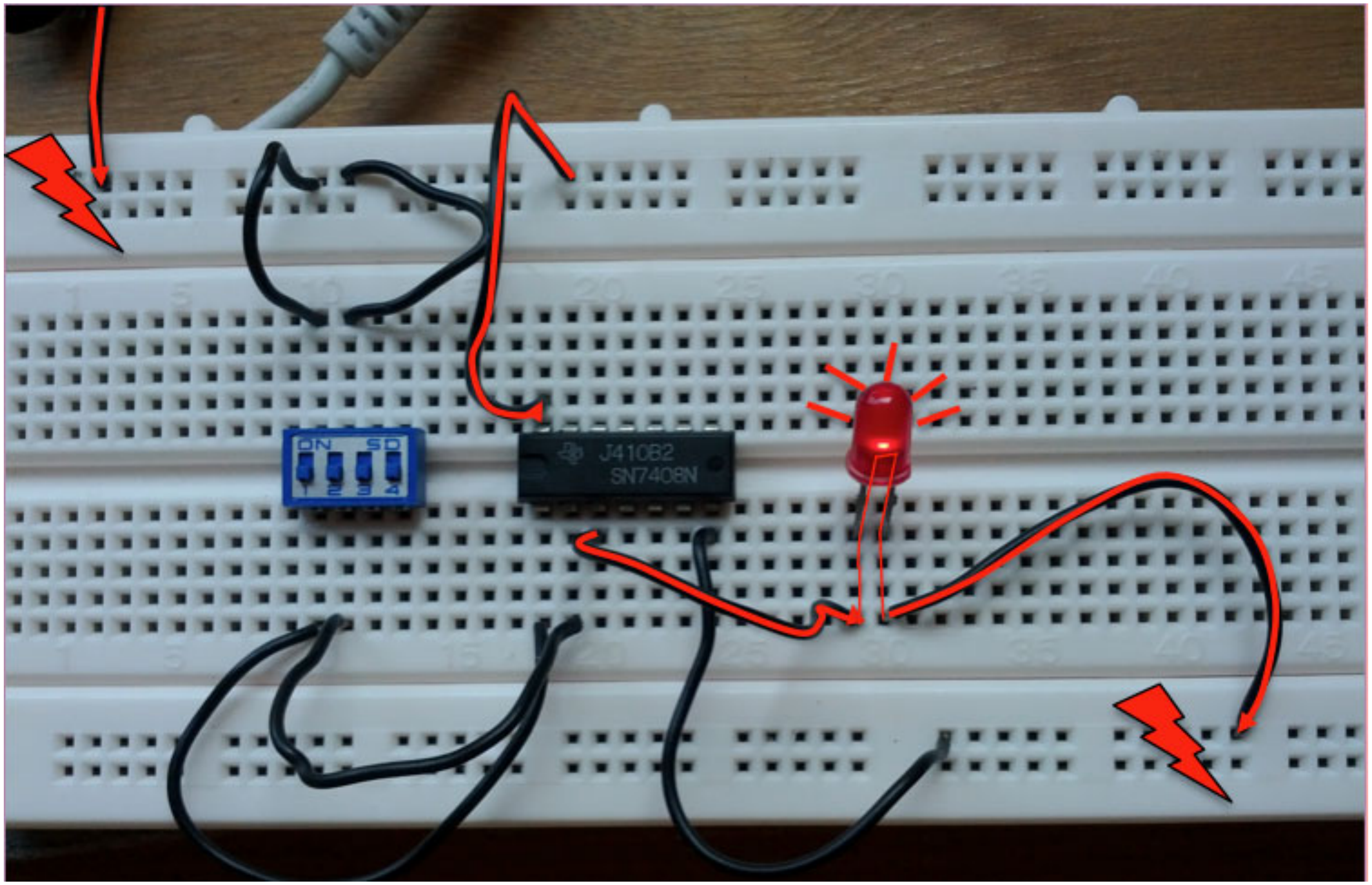
7432: quad two input OR gates



7404: hex inverter gates

Gates


Vcc
4B
4A
4Y
3B
3A
3Y
14
13
12
11
10
9
8
5V
1
2
3
4
5
6
7
1A
1B
1Y
2A
2B
2Y
GND


ON
SD
1 2 3 4
J410B2
SN7408N

The AND Gate

- The behavior of a gate is described by its *truth table* that *defines its output for each of the possible inputs*.
- Table 2.8a provides the truth table for the *two-input* **AND** gate.
  - If the two input are *A* and *B*, then the output C will be true (i.e., 1) if and only if both inputs *A* and *B* are true (i.e., 1) simultaneously.
- Table 2.8b gives the truth table for the *three input* **AND** gate.
  - If A, B, and C are the inputs, then the output D will be true (i.e., 1) if and only if all inputs are true (i.e., 1) simultaneously.
- The **AND** is represented by a "**.**"
  - the operation A **AND** B can be written as A **.** B

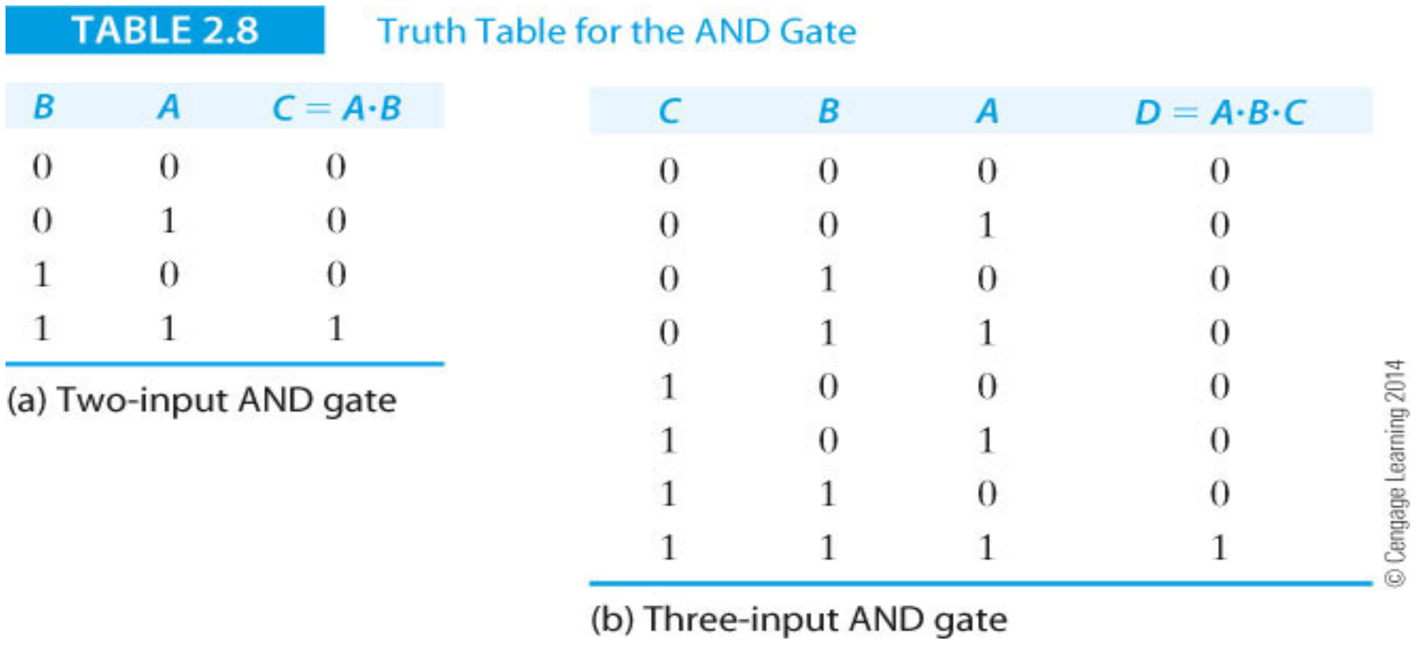


TABLE 2.8 Truth Table for the AND Gate

B	*A*	*C* = *A*·*B*
0	0	0
0	1	0
1	0	0
1	1	1

(a) Two-input AND gate

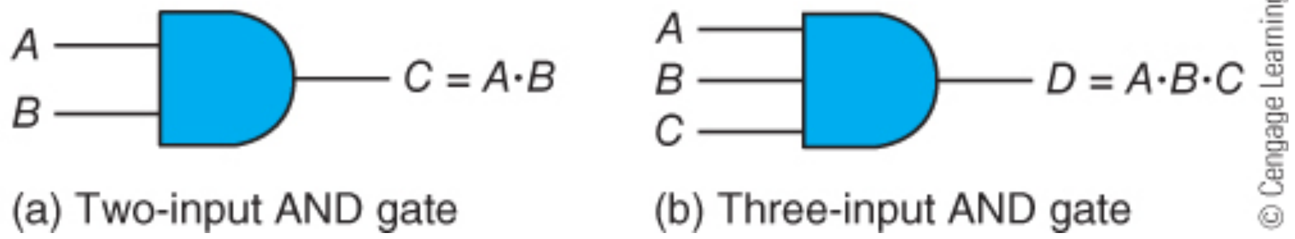
C	*B*	*A*	*D* = *A*·*B*·*C*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) Three-input AND gate

The AND Gate

- Figure 2.14 gives the symbols for 2-input and 3-input **AND** gates

FIGURE 2.14 The symbol for an AND gate



(a) Two-input AND gate (b) Three-input AND gate

The OR Gate

- The output of an **OR** gate is 1 if at least one of its inputs is 1.
- The only way to make the output of an **OR** gate go to a logical 0 is to set all its inputs to 0.
- The **OR** is represented by a "**+**"
  - the operation A **OR** B can be written as A + B

TABLE 2.9 Truth Table for the OR Gate

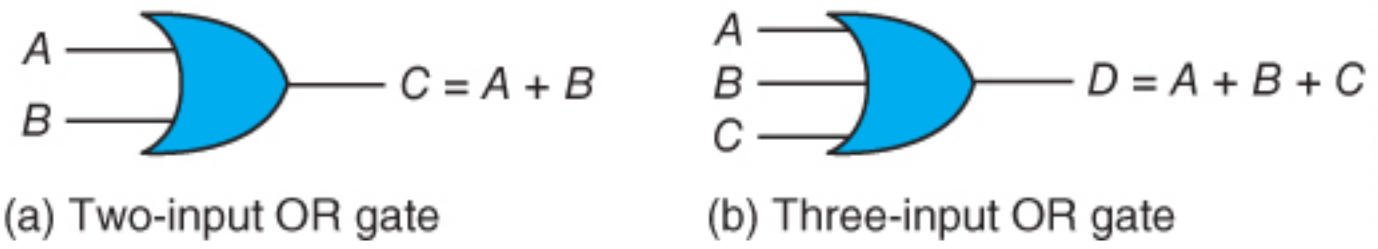
B	*A*	*C* = *A* + *B*
0	0	0
0	1	1
1	0	1
1	1	1

(a) Two-input OR gate

C	*B*	*A*	*D* = *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

(b) Three-input OR gate

FIGURE 2.15 The symbol for an OR gate



The NOT gate or inverter

- The **NOT** is represented by
  - an "***overbar***", e.g., the operation **NOT** A is written as $\overline{A}$

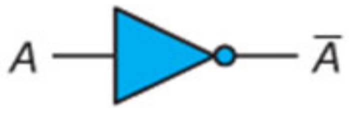
  or
  - a superscript c, e.g., the operation **NOT** A is written as A^c

  or
  - a tilde mark ~, e.g., the operation **NOT** A is written as ~ A

  or
  - a negation mark ¬, e.g., the operation **NOT** A is written as ¬ A

  or
  - an exclamation mark !, e.g., the operation **NOT** A is written as !A
- Note that, $\overline{\overline{A}} = A$

FIGURE 2.16 The symbol and truth table for an inverter



(a) Symbol for inverter

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

(b) Truth table of inverter

This bit is 1 not 0.
The book wrote it incorrectly as 0.

Comparing AND and OR Gates

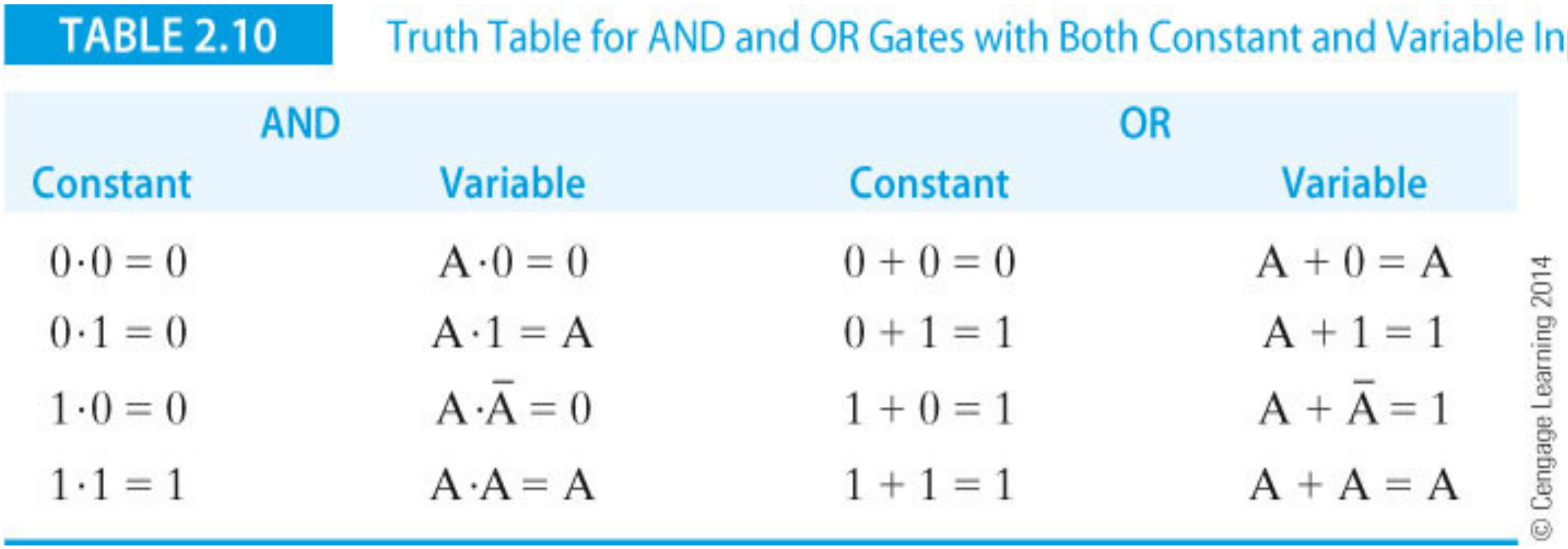


TABLE 2.10 Truth Table for AND and OR Gates with Both Constant and Variable Inputs

AND		OR	
Constant	Variable	Constant	Variable
$0 \cdot 0 = 0$	$A \cdot 0 = 0$	$0 + 0 = 0$	$A + 0 = A$
$0 \cdot 1 = 0$	$A \cdot 1 = A$	$0 + 1 = 1$	$A + 1 = 1$
$1 \cdot 0 = 0$	$A \cdot \overline{A} = 0$	$1 + 0 = 1$	$A + \overline{A} = 1$
$1 \cdot 1 = 1$	$A \cdot A = A$	$1 + 1 = 1$	$A + A = A$

Derived Gates NOR, NAND, Exclusive OR

- **NOR**, **NAND** and **XOR** are gates that can be derived from basic gates.
  - a **NOR** gate is an **OR** followed by an **inverter**.
  - A **NAND** gate is an **AND** followed by and **inverter** and
  - An **XOR** gate is an **OR** gate whose output is true *only if an odd number of its input is true.*

This is B not C

TABLE 2.11 Truth Table for the NOR Gate, NAND Gate, and Exclusive OR Gates

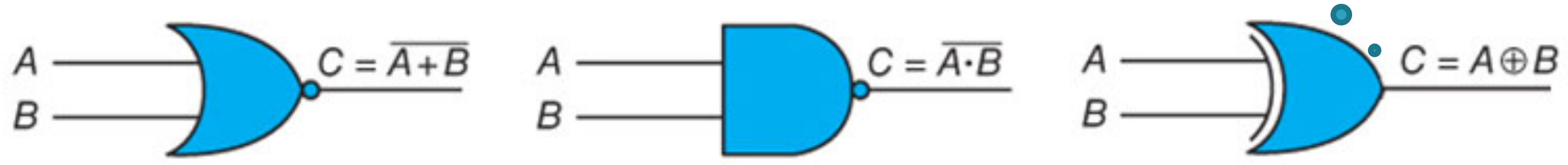
A	C	$C = \overline{A+B}$		A	B	$C = \overline{A \cdot B}$		A	B	$C = A \oplus B$
0	0	1		0	0	1		0	0	0
0	1	0		0	1	1		0	1	1
1	0	0		1	0	1		1	0	1
1	1	0		1	1	0		1	1	0

(a) The NOR gate (b) The NAND gate (c) The XOR gate

- These gates (**NOR**, **NAND**, and **XOR**) are used extensively in digital circuits and have their own symbols.

There is no bubble here.
The book added it incorrectly.

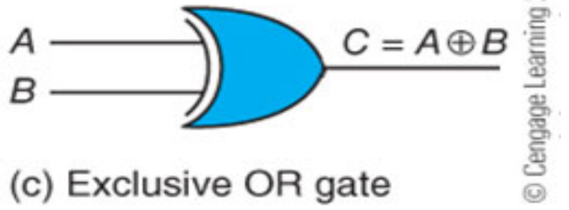
FIGURE 2.19 Three derived gates



(a) NOR gate (b) NAND gate (c) Exclusive OR gate

Exclusive OR

- The **Exclusive OR** function is written as **XOR** or **EOR**.
- The **Exclusive OR** is represented by $\oplus$ (e.g., $C = A \oplus B$).
- A two-input **XOR** gate can be constructed by *two* **inverters**, *two* **AND** gates and *one* **OR** gate, as shown in Figure 2.20. ($F = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$)



(c) Exclusive OR gate

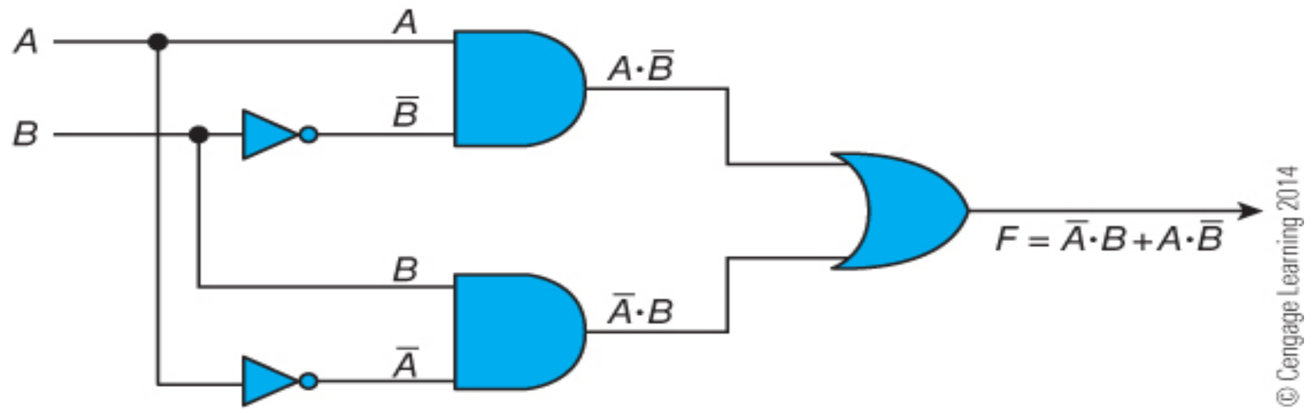
Exclusive OR Gates

A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

© Cengage Learning 2014

(c) The XOR gate

FIGURE 2.20 Constructing an XOR circuit from AND, OR, and NOT gates



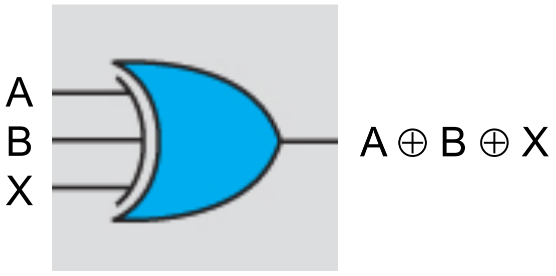
Three Input Exclusive OR

- A three-input **XOR** gate can be constructed with *two* **XOR** gates, each with two-inputs
- $C = A \oplus B = A . \overline{B} + \overline{A} . B$
- $A \oplus B \oplus X = C \oplus X = C . \overline{X} + \overline{C} . X$

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

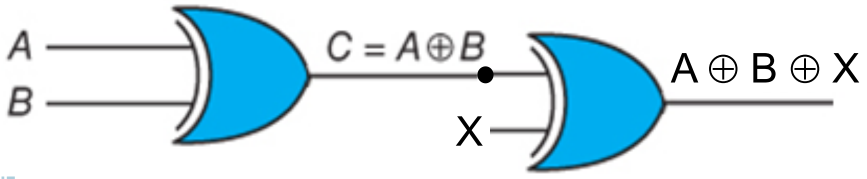
$$= (A . \overline{B} + \overline{A} . B) . \overline{X} + (\overline{A} . \overline{B} + A . B) . X$$

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



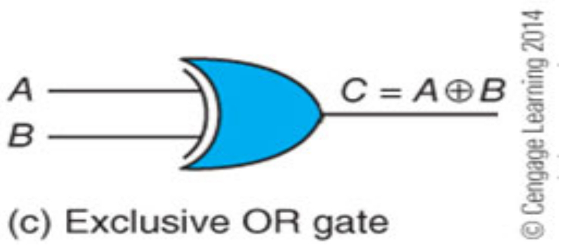
A	B	X	A ⊕ B ⊕ X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

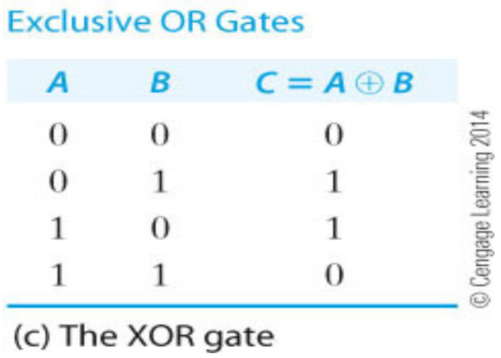


Inversion Bubbles

- By **convention**, the triangle in inverters are often omitted from circuit diagrams and the *bubble notation is used*.
- *A small bubble is placed at a gate's input to indicate inversion.*
- In the circuit below, the *two* **AND** gates form the product of (**NOT** A) **AND** B and A **AND** (**NOT** B), i.e., $\bar{A} . B + A . \bar{B}$
- This circuit implements **XOR**



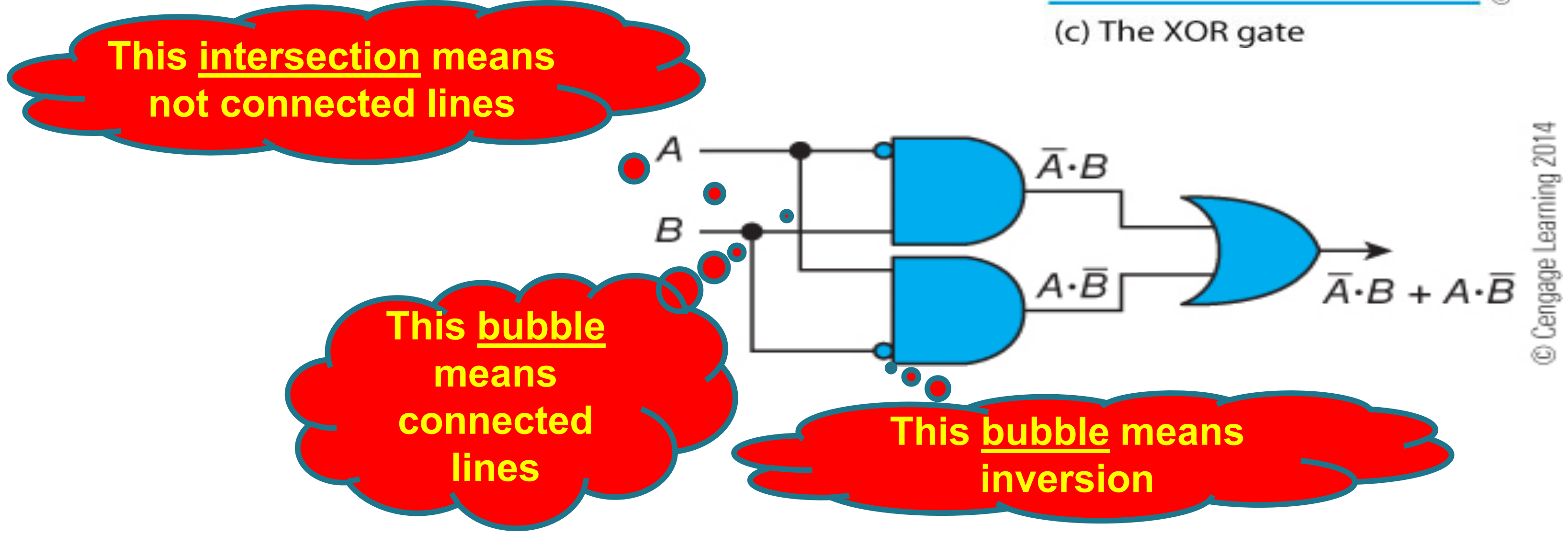
(c) Exclusive OR gate


Exclusive OR Gates

A	B	C = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

(c) The XOR gate

© Cengage Learning 2014


This intersection means not connected lines
A
B
$\bar{A} \cdot B$
$A \cdot \bar{B}$
$\bar{A} \cdot B + A \cdot \bar{B}$
© Cengage Learning 2014
This bubble means connected lines
This bubble means inversion

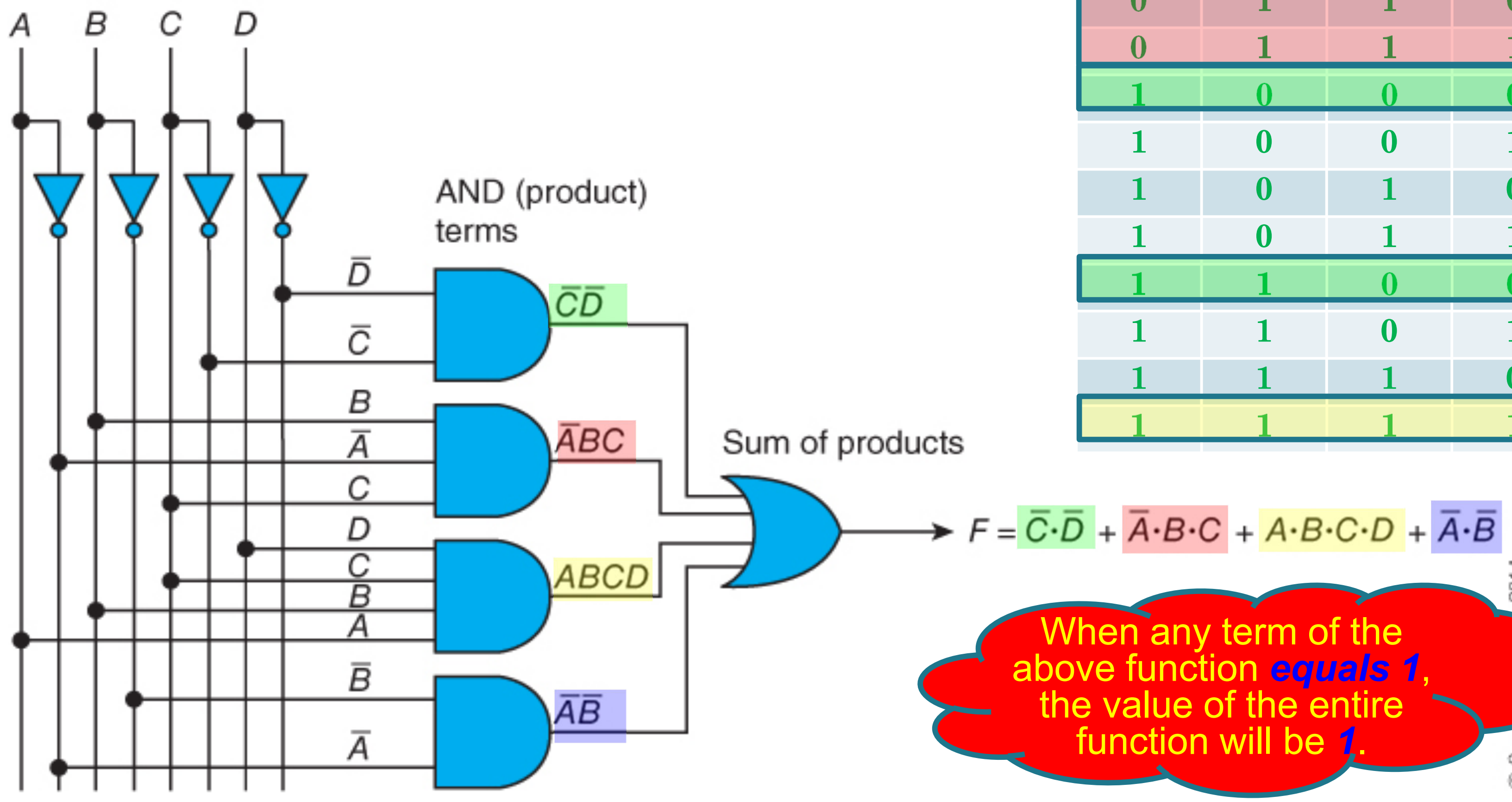
Example of a Digital Circuit

- This is called a ***sum of products*** circuit.
- The output is the **OR** of **AND** terms
- Lines that cross each other ***without a <u>black</u> dot*** at their intersection are ***not connected*** together
- lines that ***meet at a dot*** are **connected**.

The *sum of products* truth table is identified by its 1's as output

A	B	C	D	O/P
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

FIGURE 2.17 The generic AND-OR circuit



$$F = \overline{C}\cdot\overline{D} + \overline{A}\cdot B\cdot C + A\cdot B\cdot C\cdot D + \overline{A}\cdot\overline{B}$$

When any term of the above function ***equals 1***, the value of the entire function will be ***1***.

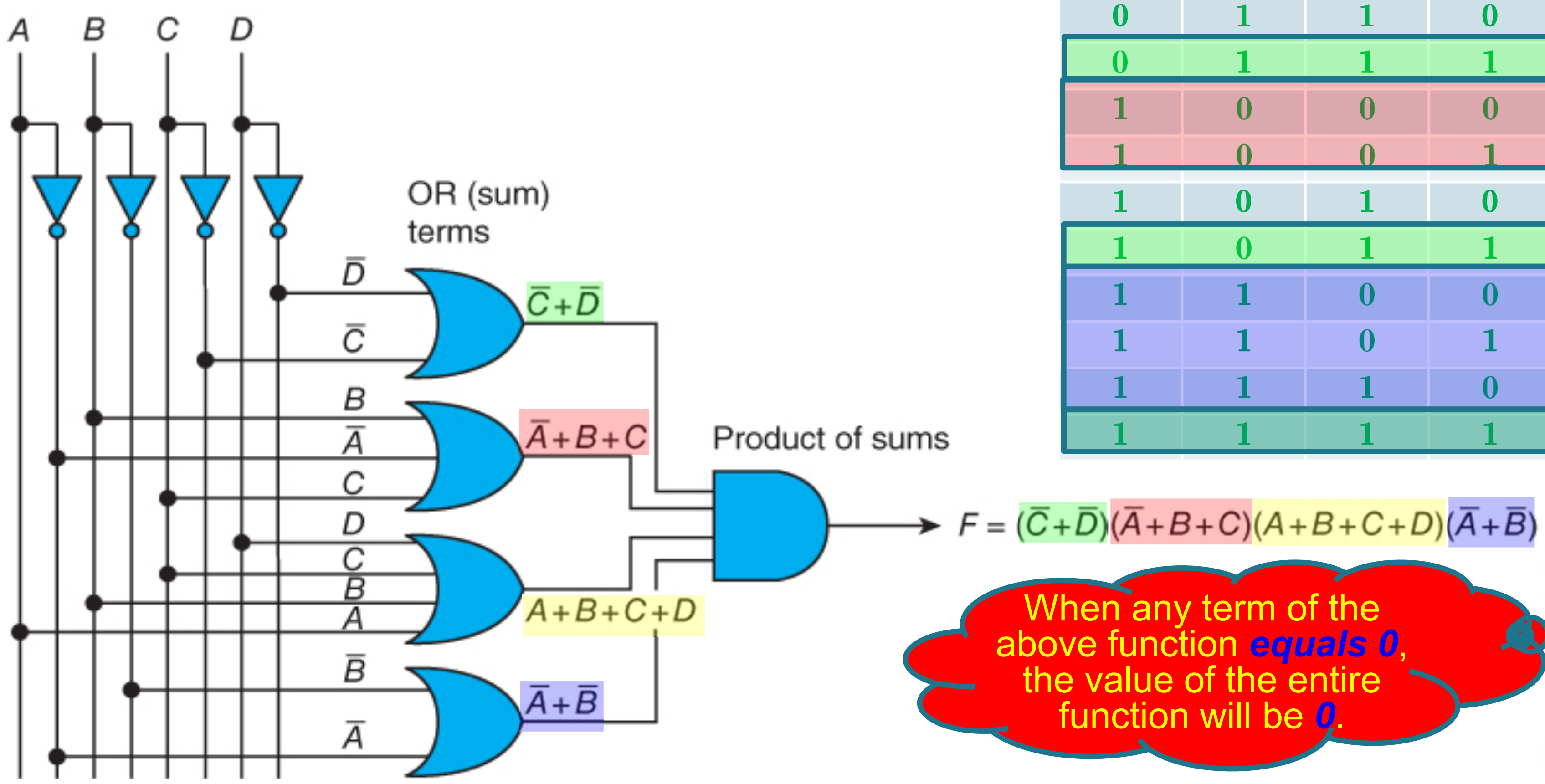
Example of a Digital Circuit

The *product of sums* truth table is identified by its 0's as output

- This is called a ***product of sums*** circuit.
- The output is the **AND** of **OR** terms

A	B	C	D	O/P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

FIGURE 2.18 The generic OR-AND circuit


A B C D
OR (sum) terms
$\overline{D}$
$\overline{C}$
$\overline{C}+\overline{D}$
B
$\overline{A}$
C
$\overline{A}+B+C$
D
C
B
A
A+B+C+D
$\overline{B}$
$\overline{A}$
$\overline{A}+\overline{B}$
Product of sums
$F=(\overline{C}+\overline{D})(\overline{A}+B+C)(A+B+C+D)(\overline{A}+\overline{B})$

When any term of the above function ***equals 0***, the value of the entire function will be **0**.

Boolean Algebra Follows Normal Algebraic Laws

- ❑ X + Y = Y + X (Commutative law)
- ❑ X . Y = Y . X (Commutative law)
- ❑ X + (Y + Z) = (X + Y) + Z (Associative law)
- ❑ X . (Y . Z) = (X . Y) . Z (Associative law)

$x \vee (Y \wedge z) = (x \vee Y) \wedge (x \vee z)$.

- ❑ X + Y . Z = (X + Y). (X + Z) (Distributive law)
- ❑ X . (Y + Z) = X . Y + X . Z (Distributive law)

$x \wedge (Y \vee z) = (x \wedge Y) \vee (x \wedge z)$.

- ❑ $\overline{X+Y} = \bar{X}.\bar{Y}$ (De Morgan's law)
- ❑ $\overline{X.Y} = \bar{X}+\bar{Y}$ (De Morgan's law)
- ❑ $X + \bar{X}.Y = X + Y$