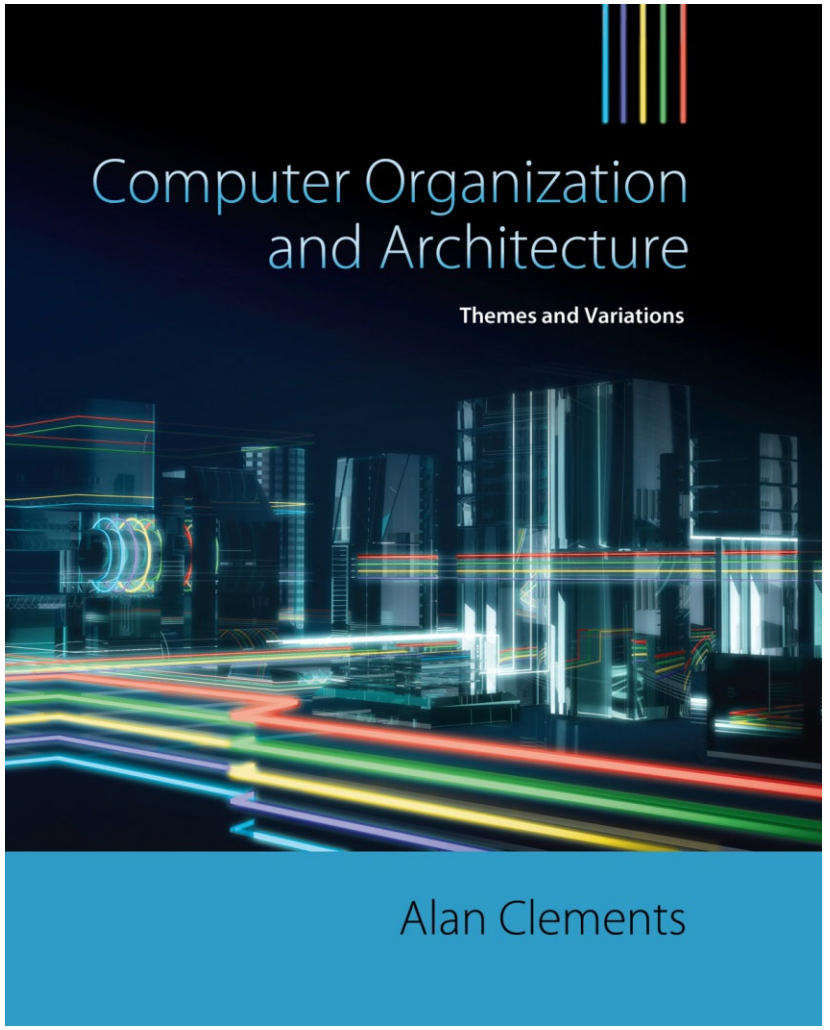
Part 4

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.

  ⇒ both take input to give output

- A *combinational* logic element is a circuit whose output depends only on its current inputs, → no history : no memory.

  whereas

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

  → has memory.

- 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*

flipflops can be built using 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)

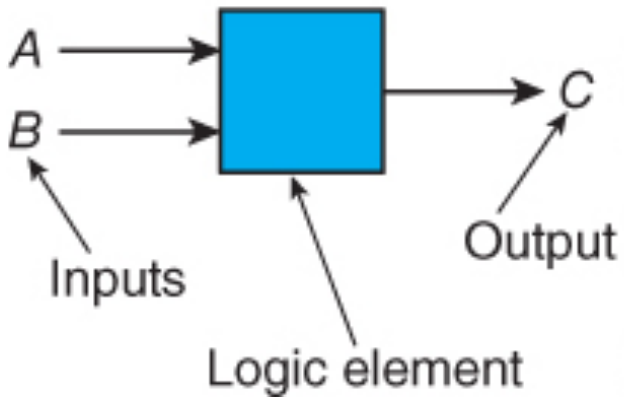
- Each logic state has an ***inverse*** or a ***complement*** that is the opposite of its current state.
  - The complement of a *true* or *1* state is a *false* or *0* state
  - The complement of a *false* or *0* state is a *true* or *1* state

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

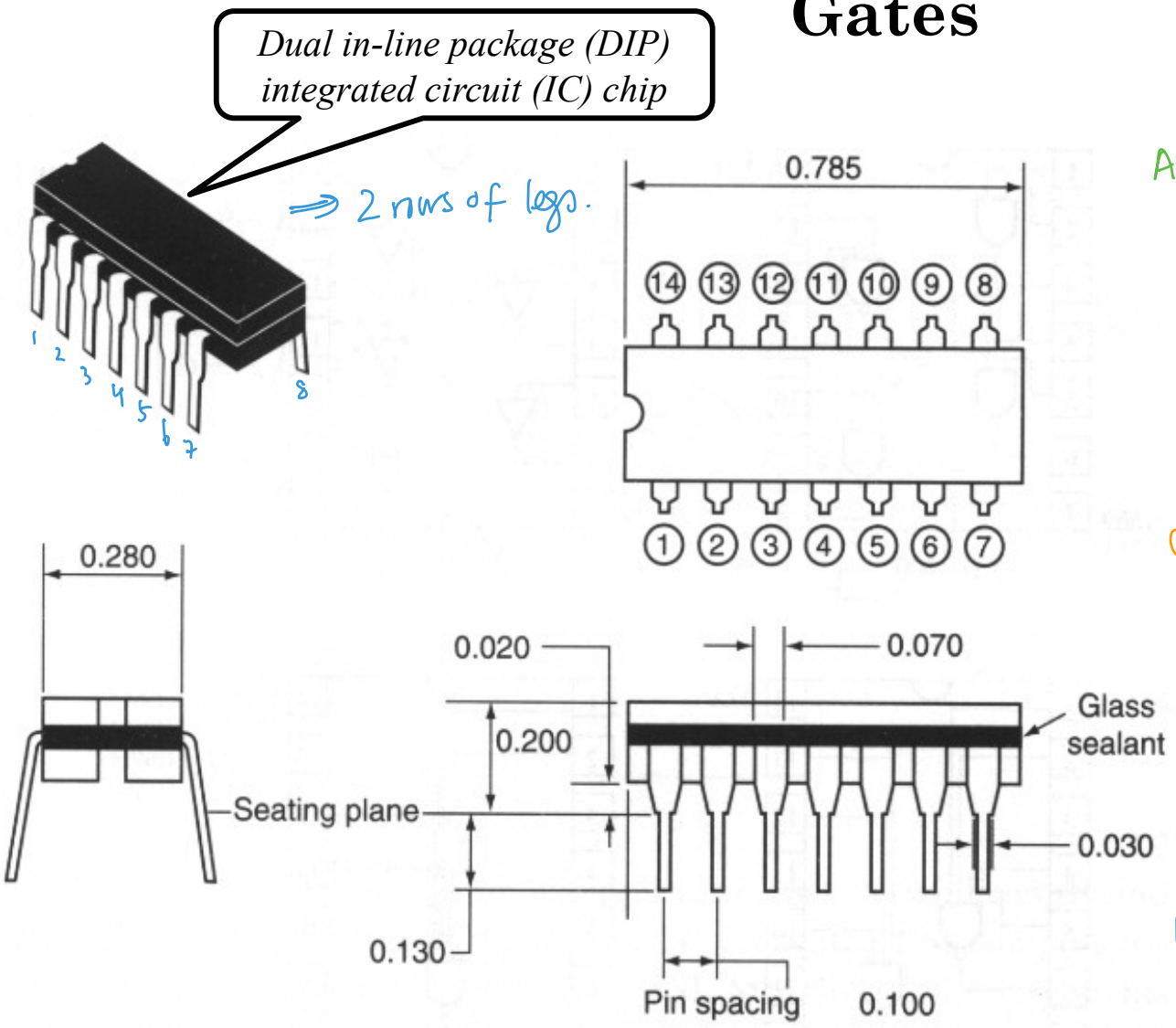
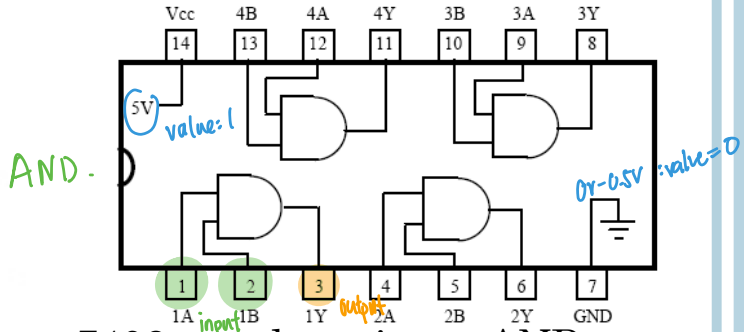
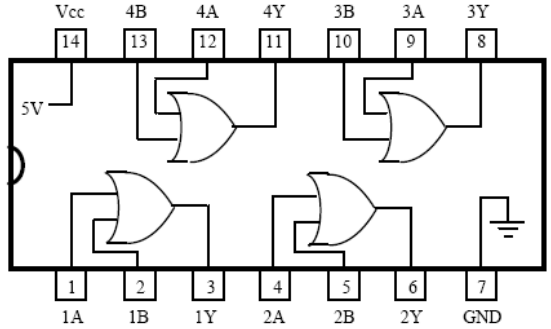


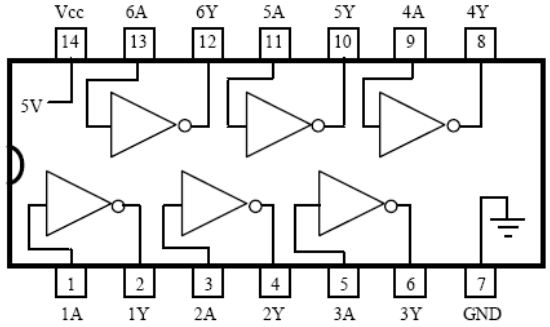
FIGURE 3.17
14-pin ceramic package (all linear dimensions are in inches).



7408: quad two input AND gates

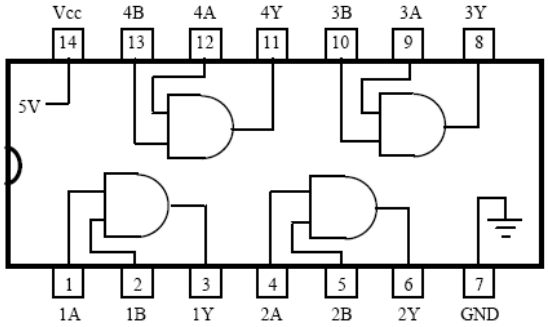


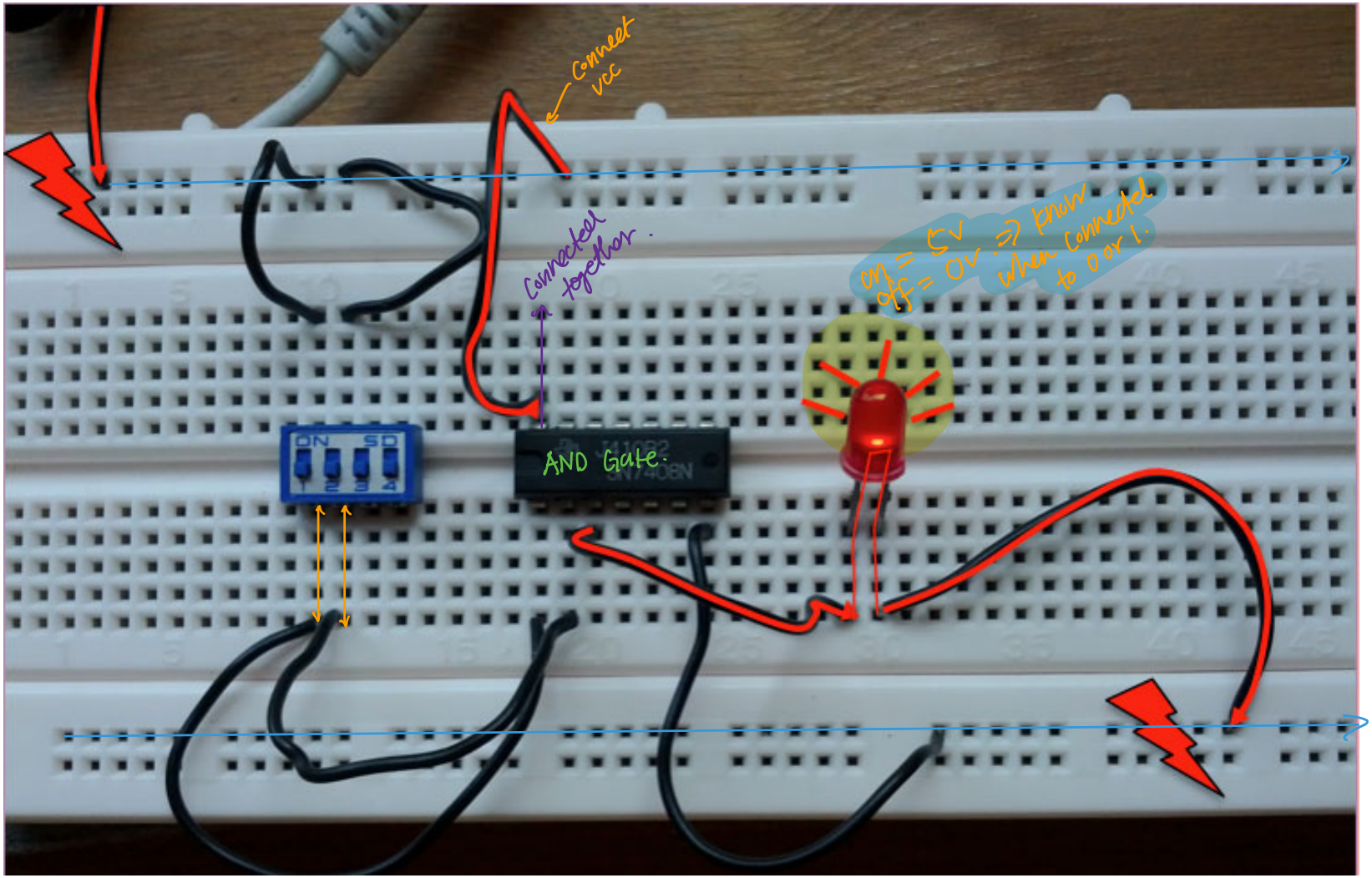
7432: quad two input OR gates



7404: hex inverter gates

Gates





output = 1 when all input = 1.

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

truth table: all possible cases.

B	A	C = A·B
0	0	0
0	1	0
1	0	0
1	1	1

(a) Two-input AND gate

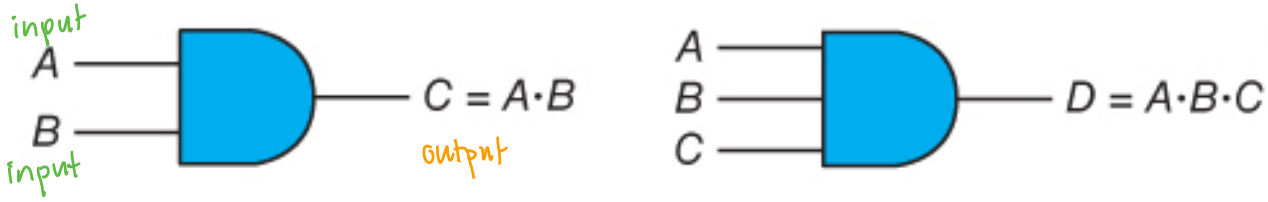
C	B	A	D = A·B·C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(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

produce 0 if all input = 0.
1 if at least one input = 1

- 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

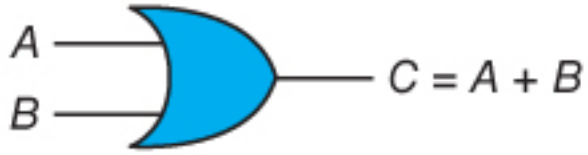
B	A	C = A + B
0	0	0
0	1	1
1	0	1
1	1	1

(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



(a) Two-input OR gate

(b) Three-input OR gate

The NOT gate or inverter

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

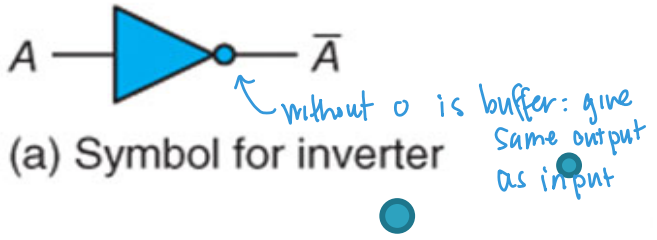
  or
  - a superscript c, e.g., the operation **NOT** A is written as A^c

  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.