Lecture 7 – Binary Logic

Chapter 2

Postulates of Boolean algebra – Duality principle

- The *duality principle* states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged, i.e., AND is changed to OR and vice versa
- In a two-valued Boolean algebra, the identity elements and the elements of the set *S* are the same: 1 and 0
- Thus, if the *dual* of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's

(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
(a)	$x + x = x$	(b)	$x \cdot x = x$
(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$

Theorems of Boolean algebra

- Theorem: $x + x = x$

$$
\begin{aligned}
x + x &= (x + x) \cdot 1 \\
&= (x + x) \cdot (x + x') \\
&= x + x \cdot x' \\
&= x + 0 \\
&= x
\end{aligned}
$$

- Its dual: $x \cdot x = x$

Theorems of Boolean algebra

- Theorem: $x + 1 = 1$

$$
\begin{aligned}
x + 1 &= 1 \cdot (x + 1) \\
&= (x + x') \cdot (x + 1) \\
&= x + x' \cdot 1 \\
&= x + x' \\
&= 1
\end{aligned}
$$

- Its dual: $x \cdot 0 = 0$

Theorems of Boolean algebra

- Theorem: $x + xy = x$ (absorption theorem)

$$\begin{aligned} x + xy &= x \cdot 1 + x.y \\ &= x \cdot (1 + y) \\ &= x \cdot 1 \\ &= x \end{aligned}$$

- Its dual: $x \cdot (x + y) = x$

Theorems of Boolean algebra

- Theorem: $(x + y)' = x' \cdot y'$ (DeMorgan's theorem)
- A simple way to prove the theorem is to prove that (x+y) and x'y' are complements of each other so that (x+y)' is the same as x'y'
- So, we need to prove that $(x + y) + x'y' = 1$ and $(x + y) \cdot x'y' = 0$.

  If they are both true, then $(x + y)' = x'y'$

$$
\begin{aligned}
(x + y) + x'y' &= x + y + x'y' \\
&= x + (y + x')(y + y') \\
&= x + (y + x') \cdot 1 \\
&= x + y + x' \\
&= y + (x + x') \\
&= y + 1 \\
&= 1
\end{aligned}
$$

Theorems of Boolean algebra

- We proved that $(x + y) + x'y' = 1$.
- Let's now show that: $(x + y) \cdot x'y' = 0$.
- If they are both true, then $(x + y)' = x'y'$ (DeMorgan's theorem)

$$
\begin{aligned}
(x + y) \cdot x'y' &= x \cdot x' \cdot y' + y \cdot x' \cdot y' \\
&= (x \cdot x') \cdot y' + x' \cdot (y \cdot y') \\
&= 0 \cdot y' + x' \cdot 0 \\
&= 0 + 0 \\
&= 0
\end{aligned}
$$

- Because both are true, the theorem is proved

Operator precedence

- The operator precedence for evaluating Boolean expressions is :
  - 1) parentheses, 2) NOT, 3) AND, 4) OR.
- In other words, expressions inside parentheses must be evaluated before all other operations
- The next operation that holds precedence is the complement, and then follows the AND and, finally, the OR
- As an example, consider one of DeMorgan's theorems
  - The left side of the expression is $(x + y)'$. Therefore, the expression inside the parentheses is evaluated first and the result then complemented
  - The right side of the expression is $x'y'$, so the complement of x and the complement of y are both evaluated first and the result is then ANDed

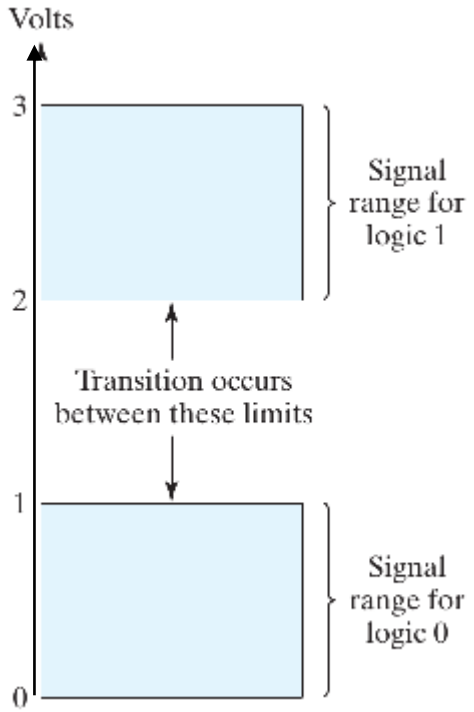
Logic gates!

- Logic gates are electronic circuits that operate on one or more input signals to produce an output signal

- Electrical signals such as voltages or currents exist as analog signals having values over a given continuous range, say, 0 to 3 V, but in a digital system these voltages are interpreted to be either of two recognizable values, 0 or 1

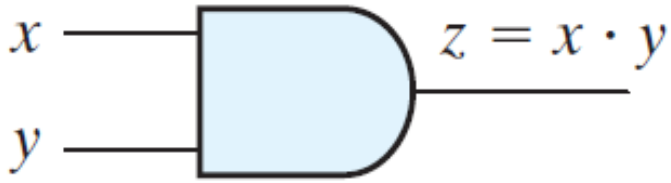
- Logic circuits respond to two separate voltage levels that represent a binary variable equal to logic 1 or logic 0

- For example, a particular digital system may define logic 0 as a signal equal to 0 V and logic 1 as a signal equal to 3 V

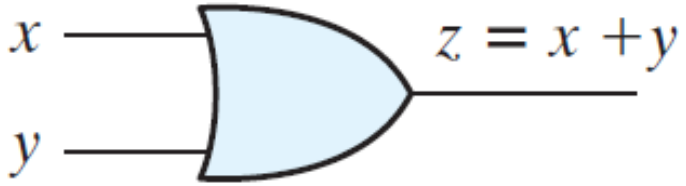


Logic gates

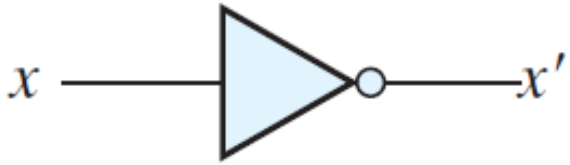
- We represent the logic gates for basic Boolean operations as shown
- Logic gates can have more than two inputs (except for NOT gate of course!)



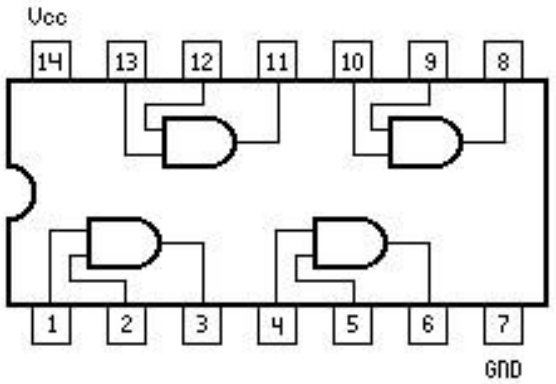
(a) Two-input AND gate



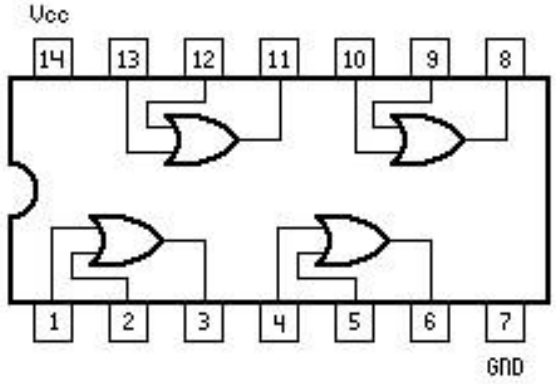
(b) Two-input OR gate



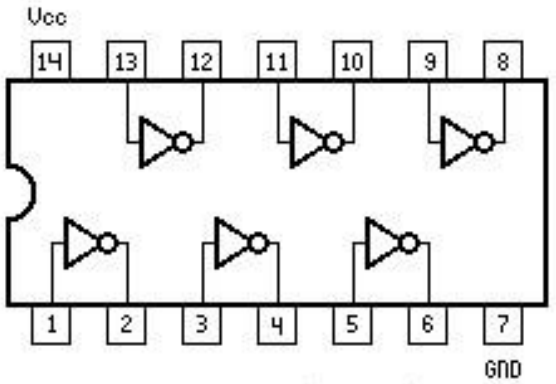
(c) NOT gate or inverter



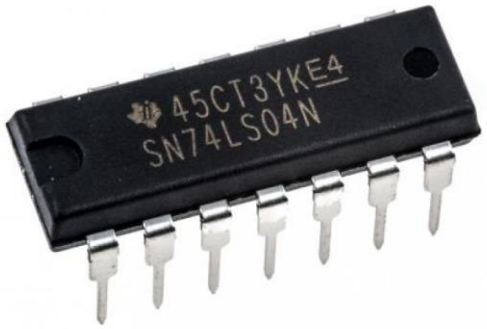
7408 (AND)



7432 (OR)

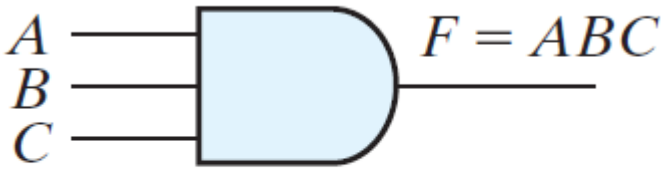


7404 (NOT)

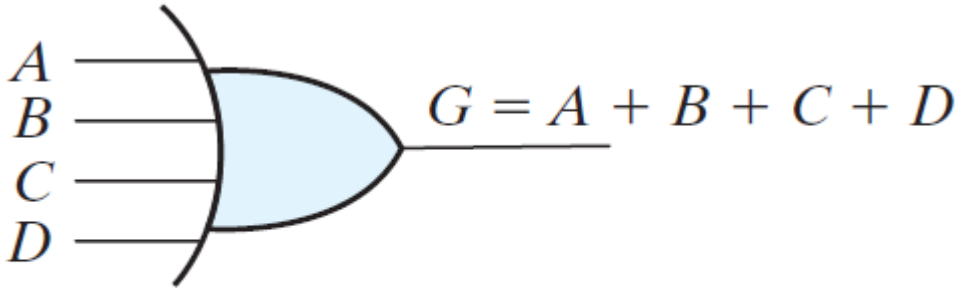


Logic gates

- AND and OR gates may have more than two inputs
- The three-input AND gate responds with logic 1 output if all three inputs are logic 1. The output produces logic 0 if any input is logic 0
- The four-input OR gate responds with logic 1 if any input is logic 1; its output becomes logic 0 only when all inputs are logic 0



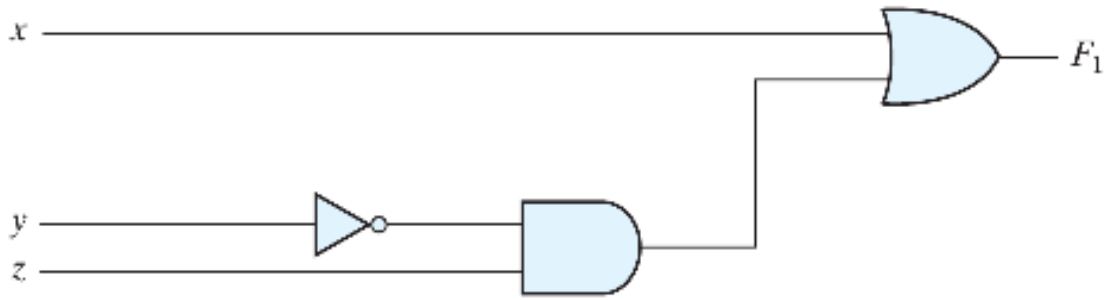
(a) Three-input AND gate



(b) Four-input OR gate

Boolean functions

- In normal algebra, we define functions on real number variables using addition, subtraction, etc.
- A Boolean function described by a Boolean expression consists of binary variables, the constants 0 and 1, and the logic operation symbols
- For a given value of the binary variables, the function can be equal to either 1 or 0 (closure on the Boolean set)
- As an example, consider the Boolean function: $F_1 = x + y'.z$



Boolean functions

- A Boolean function can be represented in a *truth table*

- The number of rows in the truth table is 2^n, where *n* is the number of variables in the function

- The binary combinations for the truth table are obtained from the binary numbers by counting from 0 through $(2^n - 1)$

- For example, there are eight possible binary combinations for assigning bits to the three variables *x*, *y*, and *z*

$$F_1 = x + y'.z$$

x	y	z	F_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

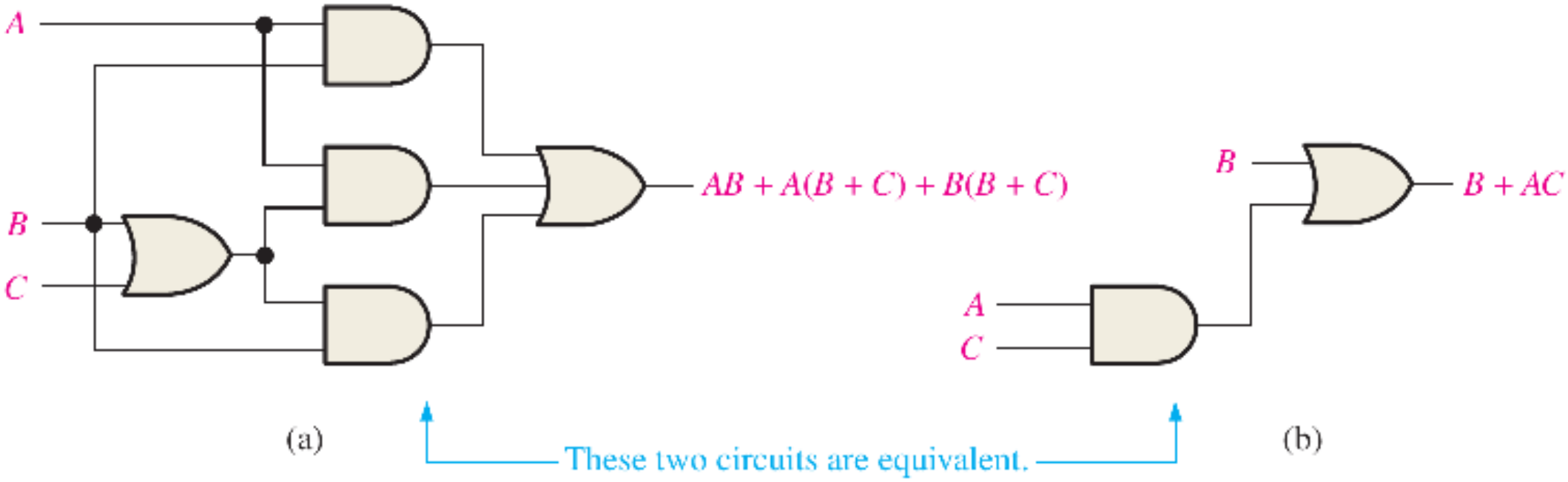
Boolean functions

- There is only one way that a Boolean function can be represented in a truth table
- *However, when the function is in algebraic form, it can be expressed in a variety of ways, all of which have equivalent logic*
- Here is a key fact that motivates our use of Boolean algebra: By manipulating a Boolean expression according to the rules of Boolean algebra, it is sometimes possible to obtain a simpler expression for the same logic function, and thus reduce the complexity of the circuit representing that logic
- Consider, for example, the following Boolean functions:

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

- Can we simply this?

Boolean functions



Boolean functions

- A simplified function also has a simplified logic gate implementation
- Therefore, the two circuits have the same outputs for all possible binary combinations of inputs (truth table)
- Thus, each circuit implements the same identical function, but the one with fewer gates and fewer inputs to gates is preferable because it requires fewer wires and components
- In general, there are many equivalent representations of a logic function
- Finding the most economic representation of the logic is an important design task

Boolean functions

- The manipulation of Boolean algebra consists mostly of reducing an expression for the purpose of obtaining a simpler circuit
- Functions of up to five variables can be simplified by the map method which will be discussed later
- For complex Boolean functions and many different outputs, designers of digital circuits use computer minimization programs that are capable of producing optimal circuits with millions of logic gates
- The manual method available is a procedure employing the basic relations and other manipulation techniques that become familiar with use, but remain, nevertheless, subject to human error
- Example: Simplify $F = xy + x'z + yz$