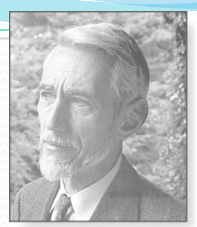
Boolean Algebra

Chapter 12

Chapter Summary



Claude Shannon
(1916 - 2001)

- Boolean Functions
- Representing Boolean Functions
- Logic Gates
- Minimization of Circuits (*not currently included in overheads*)

Boolean Functions

Section 12.1

Section Summary

- Introduction to Boolean Algebra
- Boolean Expressions and Boolean Functions
- Identities of Boolean Algebra
- Duality
- The Abstract Definition of a Boolean Algebra

Introduction to Boolean Algebra

- Boolean algebra has rules for working with elements from the set {0, 1} together with the operators + (Boolean sum), · (Boolean product), and $^-$ (complement).
- These operators are defined by:
  - *Boolean sum*: $1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0$
  - *Boolean product*: $1 \cdot 1 = 1, 1 \cdot 0 = 0, 0 \cdot 1 = 0, 0 \cdot 0 = 0$
  - *complement*: $\overline{0} = 1, \overline{1} = 0$

Example: Find the value of $1 \cdot 0 + \overline{(0 + 1)}$

Solution : $1 \cdot 0 + \overline{(0 + 1)} = 0 + \overline{1}$

$= 0 + 0$

$= 0$

Boolean Expressions and Boolean Functions

Definition: Let $B = \{0, 1\}$. Then $B^n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in B \text{ for } 1 \leq i \leq n\}$ is the set of all possible n-tuples of 0s and 1s. The variable x is called a *Boolean variable* if it assumes values only from B, that is, if its only possible values are 0 and 1. A function from B^n to B is called a *Boolean function of degree n*.

Example: The function $F(x, y) = x$ from the set of ordered pairs of Boolean variables to the set $\{0, 1\}$ is a Boolean function of degree 2.

TABLE 1

x	y	$F(x, y)$
1	1	1
1	0	0
0	1	0
0	0	1

Boolean Expressions and Boolean Functions (*continued*)

Example: Find the values of the Boolean function represented by $F(x, y, z) = xy + \bar{z}$.

Solution: We use a table with a row for each combination of values of x, y, and z to compute the values of $F(x,y,z)$.

TABLE 2

x	y	z	xy	$\bar{z}$	$F(x, y, z) = xy + \bar{z}$
1	1	1	1	0	1
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	1
0	1	1	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0
0	0	0	0	1	1

Boolean Expressions and Boolean Functions (*continued*)

Definition: Boolean functions F and G of n variables are equal if and only if $F(b_1, b_2, \ldots, b_n) = G(b_1, b_2, \ldots, b_n)$ whenever $b_1, b_2, \ldots, b_n$ belong to B. Two different Boolean expressions that represent the same function are *equivalent*.

Definition: The complement of the Boolean function F is the function $\bar{F}$, where $\bar{F}(x_1, x_2, \ldots, x_n) = \overline{F(x_1, x_2, \ldots, x_n)}$.

Definition: Let F and G be Boolean functions of degree n. The Boolean sum $F + G$ and the Boolean product FG are defined by

$(F + G)(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n) + G(x_1, x_2, \ldots, x_n)$

$(FG)(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n)G(x_1, x_2, \ldots, x_n)$

Boolean Functions

Example: How many different Boolean functions of degree n are there?

Solution: By the product rule for counting, there are 2^n different n-tuples of 0s and 1s. Because a Boolean function is an assignment of 0 or 1 to each of these different n-tuples, by the product rule there are 2^{2^n} different Boolean functions of degree n.

TABLE 4 The Number of Boolean Functions of Degree n.

Degree	*Number*
1	4
2	16
3	256
4	65,536
5	4,294,967,296
6	18,446,744,073,709,551,616

The example tells us that there are 16 different Boolean functions of degree two. We display these in Table 3.

TABLE 3 The 16 Boolean Functions of Degree Two.

x	y	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Identities of Boolean Algebra

TABLE 5 Boolean Identities.

Identity	*Name*
$\overline{\overline{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \overline{x} + \overline{y}$ $\overline{(x + y)} = \overline{x}\,\overline{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \overline{x} = 1$	Unit property
$x\overline{x} = 0$	Zero property

Each identity can be proved using a table.

All identities in Table 5, except for the first and the last two come in pairs. Each element of the pair is the *dual* of the other (obtained by switching Boolean sums and Boolean products and 0's and 1's.

The Boolean identities correspond to the identities of propositional logic (Section 1.3) and the set identities (Section 2.2).

Identities of Boolean Algebra

Example: Show that the distributive law $x(y + x) = xy + xz$ is valid.

Solution: We show that both sides of this identity always take the same value by constructing this table.

TABLE 6 Verifying One of the Distributive Laws.

x	y	z	$y+z$	xy	xz	$x(y+z)$	$xy+xz$
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

Formal Definition of a Boolean Algebra

Definition: A *Boolean algebra* is a set B with two binary operations $\vee$ and $\wedge$, elements 0 and 1, and a unary operation $^-$ such that for all x, y, and z in B:

$x \vee 0 = x$
$x \wedge 1 = x$

identity laws

$x \vee \bar{x} = 1$
$x \wedge \bar{x} = 0$

complement laws

$(x \vee y) \vee z = x \vee (y \vee z)$
$(x \wedge y) \wedge z = x \wedge (y \wedge z)$

associative laws

$x \vee y = y \vee x$
$x \wedge y = y \wedge x$

commutative laws

$x \vee (y \wedge z) = (x \vee y) \wedge (y \vee z)$
$x \wedge (y \vee z) = (x \wedge y) \vee (y \wedge z)$

distributive laws

The set of propositional variables with the operators $\wedge$ and $\vee$, elements **T** and **F**, and the negation operator $\neg$ is a Boolean algebra.

The set of subsets of a universal set with the operators $\cup$ and $\cap$, the empty set ($\varnothing$), universal set (U), and the set complementation operator ($^-$) is a Boolean algebra.

Representing Boolean Functions

Section 12.2

Section Summary

- Sum-of-Products Expansions
- Functional Completeness

Sum-of-Products Expansion

Example: Find Boolean expressions that represent the functions (i) $F(x, y, z)$ and (ii) $G(x, y, z)$ in Table 1.

Solution:

(i) To represent F we need the one term $x\bar{y}z$ because this expression has the value 1 when $x = z = 1$ and $y = 0$.

TABLE 1

x	y	z	F	G
1	1	1	0	0
1	1	0	0	1
1	0	1	1	0
1	0	0	0	0
0	1	1	0	0
0	1	0	0	1
0	0	1	0	0
0	0	0	0	0

(ii) To represent the function G, we use the sum $xy\bar{z} + \bar{x}y\bar{z}$ because this expression has the value 1 when $x = y = 1$ and $z = 0$, or $x = z = 0$ and $y = 1$.

The general principle is that each combination of values of the variables for which the function has the value 1 requires a term in the Boolean sum that is the Boolean product of the variables or their complements.

Sum-of-Products Expansion (*cont*)

Definition: A *literal* is a Boolean variable or its complement. A *minterm* of the Boolean variables $x_1, x_2, \ldots, x_n$ is a Boolean product $y_1 y_2 \cdots y_n$, where $y_i = x_i$ or $y_i = \overline{x}_i$. Hence, a minterm is a product of n literals, with one literal for each variable.

The minterm $y_1, y_2, \ldots, y_n$ has value has value 1 if and only if each x_i is 1.This occurs if and only if $x_i = 1$ when $y_i = x_i$ and $x_i = 0$ when $y_i = \overline{x}_i$.

Definition: The sum of minterms that represents the function is called the *sum-of-products expansion* or the *disjunctive normal form* of the Boolean function.

Sum-of-Products Expansion (*cont*)

Example: Find the sum-of-products expansion for the function $F(x,y,z) = (x + y)\,\bar{z}$.

Solution: We use two methods, first using a table and second using Boolean identities.

(*i*) Form the sum of the minterms corresponding to each row of the table that has the value 1.

TABLE 2

x	y	z	$x + y$	$\bar{z}$	$(x + y)\bar{z}$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	1	0	0
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	1	1
0	0	1	0	0	0
0	0	0	0	1	0

Including a tem for each row of the table for which $F(x,y,z) = 1$ gives us $F(x, y, z) = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z}$.

Sum-of-Products Expansion (*cont*)

(*ii*) We now use Boolean identities to find the disjunctive normal form of F(*x*,*y*,*z*):

$$
\begin{aligned}
F(x,y,z) &= (x + y)\,\bar{z} \\
&= x\bar{z} + y\bar{z} && \textit{distributive law} \\
&= x1\bar{z} + 1y\bar{z} && \textit{identity law} \\
&= x(y + \bar{y})\bar{z} + (x + \bar{x})y\bar{z} && \textit{unit property} \\
&= xy\bar{z} + x\bar{y}\bar{z} + xy\bar{z} + \bar{x}y\bar{z} && \textit{distributive law} \\
&= xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} && \textit{idempotent law}
\end{aligned}
$$

Functional Completeness

Definition: Because every Boolean function can be represented using the Boolean operators $\cdot$, $+$, and $^-$, we say that the set $\{\cdot, +, ^-\}$ is *functionally complete*.

- The set $\{\cdot, ^-\}$ is functionally complete since $x + y = \overline{\bar{x}\bar{y}}$.
- The set $\{+, ^-\}$ is functionally complete since $xy = \overline{\bar{x} + \bar{y}}$.
- The *nand* operator, denoted by $|$, is defined by $1|1 = 0$, and $1|0 = 0|1 = 0|0 = 1$. The set consisting of just the one operator nand $\{|\}$ is functionally complete. Note that $\bar{x} = x \mid x$ and $xy = (x|y)|(x|y)$.
- The *nor* operator, denoted by $\downarrow$, is defined by $0 \downarrow 0 = 1$, and $1 \downarrow 0 = 0 \downarrow 1 = 1 \downarrow 1 = 0$. The set consisting of just the one operator nor $\{\downarrow\}$ is functionally complete. (*see Exercises* 15 *and* 16)

Logic Gates

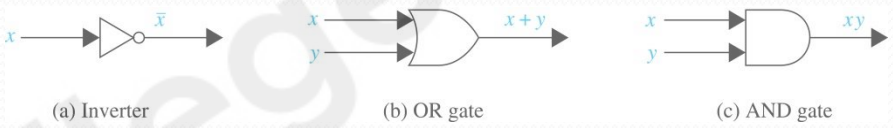
Section 12.3

Section Summary

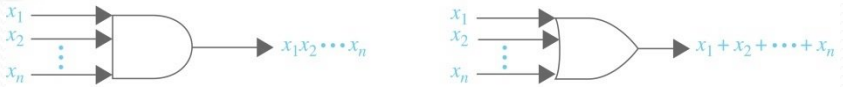
- Logic Gates
- Combinations of Gates
- Examples of Circuits

Logic Gates

- We construct circuits using *gates*, which take as input the values of two or more Boolean variables and produce one or more bits as output, and *inverters*, which take the value of a Boolean variable as input and produce the complement of this value as output.

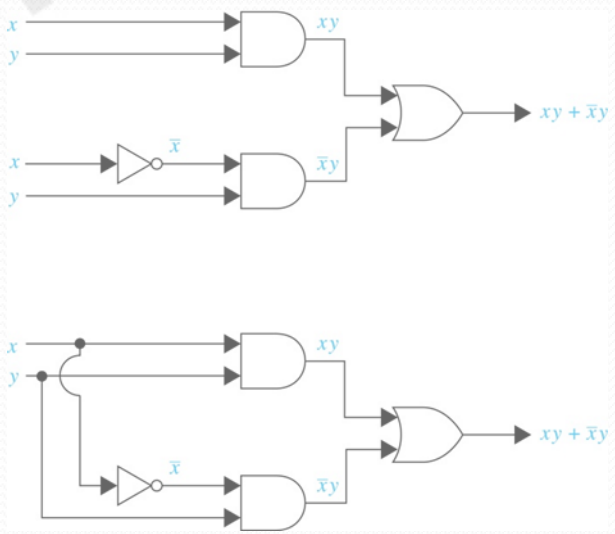


(a) Inverter (b) OR gate (c) AND gate



Combinations of Gates

- Combinatorial circuits can be constructed using a combination of inverters, OR gates, and AND gates. Gates may share input and the output of one or more gates may be input to another.
- We show two ways of constructing a circuit that produces the output $xy + \bar{x}y$.



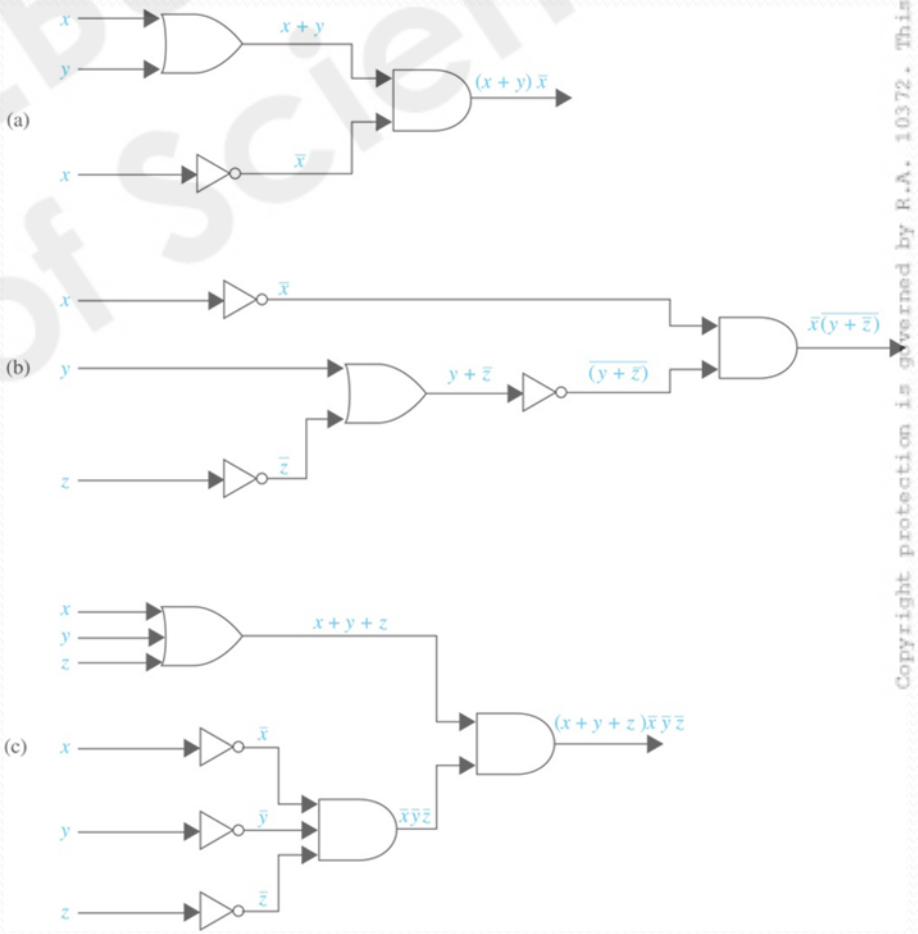
Combinations of Gates

Example: Construct circuits that produce these outputs

(a) $(x + y)\bar{x}$

(b) $\bar{x}\ \overline{(y + \bar{z})}$

(c) $(x + y + z)(\bar{x}\,\bar{y}\,\bar{z})$

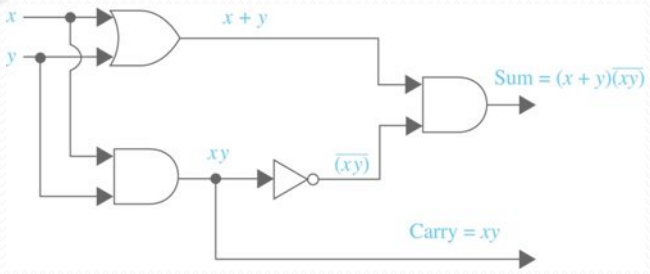


Adders

- Logic circuits can be used to add two positive integers from their binary expansions.
- The first step is to build a *half adder* that adds two bits, but which does not accept a carry from a previous addition.
- Since the circuit has more than one output, it is a *multiple output c*

TABLE 3
Input and Output for the Half Adder.

Input		*Output*	
x	*y*	*s*	*c*
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0

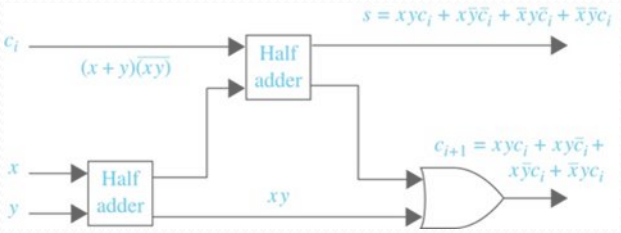


Adders (continued)

- A *full adder* is used to compute the sum bit and the carry bit when two bits and a carry are added.

TABLE 4
Input and Output for the Full Adder.

Input			*Output*	
x	y	c_i	s	c_{i+1}
1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	1	0
0	0	0	0	0



Adders (continued)

- A half adder and multiple full adders can be used to produce the sum of *n* bit integers.

  Example: Here is a circuit to compute the sum of two three-bit integers.