Chapter 3

Boolean Algebra and Combinational Networks

Digital Logic

Definition:

digit, *n*. any of the Arabic figures of 0 through 9.

digital, *adj*. involving using digits to represent discretely all variables occurring in a given problem.

logic, *n*. a particular method of reasoning or argumentation.

Analog Systems

Definition:

analog, *adj*. that solve a given problem by using physical analogies, such as electric voltages or shaft rotations (which are quantitatively continuous), of the numerical variables occurring in the problem.

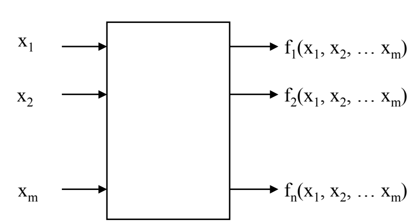
Digital vs. Analog

Digital solution to a problem is advantageous in that it allows us to build more reliable components and implement more sophisticated functions.

Two main types of logic circuit:

- Combinational (without memory) - chapters 2-6
- Sequential (with memory) - chapters 7 and 8

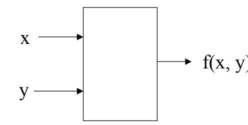
General form of a combinational circuit:



Truth table

- The behavior of a combinational circuit can always be specified by using a truth table.

For example, the behavior of the combinational circuit to the right can be described by the truth table shown below:



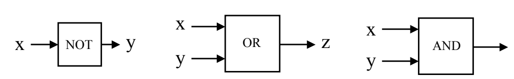
x	y	f(x, y)
0	0	f(0, 0)
0	1	f(0, 1)
1	0	f(1, 0)
1	1	f(1, 1)

A truth table exhaustively lists all possible input-output pairs.

Example of a truth table

x	y	f(x, y)
0	0	0
0	1	1
1	0	1
1	1	0

Three Basic Logic Devices

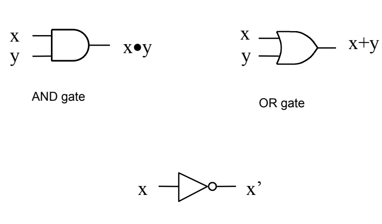


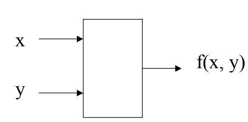
x	NOT x
0	1
1	0

OR		y = 0	y = 1
x	0	0	1
	1	1	1

AND		y = 0	y = 1
x	0	0	0
	1	0	1

Logic symbols for the basic gates



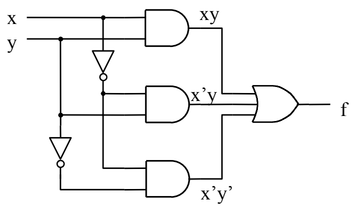


x	y	f(x, y)
0	0	1
0	1	1
1	0	0
1	1	1

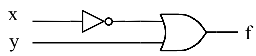
The intended function of this combinational circuit can be directly described by the truth table on the right, which can be represented by the Boolean function below:

$$f(x, y) = x'y' + x'y + xy$$

which can be realized by the logic circuit shown in the next slide.



Realization of $f(x, y) = x'y' + x'y + xy$



Minimal realization of $f(x, y)$

Algebra (Mathematical System)

An *algebra* (or a *mathematical system*) consists of

$S = <S, o_1, o_2, \ldots, o_k, r_1, r_2, \ldots, r_m, c_1, c_2, \ldots, c_n>$

where o_i's are operations defined on S, r_i's are relations defined among elements of S, and c_i's are constants, some elements of S having some special significance. Furthermore, it satisfies a set of axioms or postulates that governs its behavior.

Boolean Algebras

A ***Boolean algebra*** is a mathematical system:

$B = <B, +, \bullet, =, 0, 1>$

that satisfies the following postulates:

1a. Closure: for any $x, y \in B$, $x+y \in B$.
1b. Closure: for any $x, y \in B$, $x \bullet y \in B$.

Boolean Algebras (continued)

2a. Existence of identity: there exists an element in B, denoted by 0, such that $x+0=x$ for any $x \in B$.

2b. Existence of identity: there exists an element in B, denoted by 1, such that $x \bullet 1=x$ for any $x \in B$.

3a. Commutativity: for any $x, y \in B$, $x+y = y+x$.

3b. Commutativity: for any $x, y \in B$, $x \bullet y = y \bullet x$.

Boolean Algebras (continued)

4a. Distributive law: for any $x, y, z \in B$,

$x \bullet (y+z) = (x \bullet y)+(x \bullet z)$.

4b. Distributive law: for any $x, y, z \in B$,

$x+(y \bullet z) = (x+y) \bullet (x+z)$.

5. For any element $x \in B$ there exists an element $x' \in B$ such that (a) $x+x'=1$ and (b) $x \bullet x'=0$.

Boolean Algebras (continued)

6. There exists at least two elements in B

These are known as the Huntington postulates.

Huntington postulates

1. (a) $x+y \in B$ (b) $x \bullet y \in B$
2. (a) $x+0 = 0+x = x$ (b) $x \bullet 1 = 1 \bullet x = x$
3. (a) $x+y = y+x$ (b) $x \bullet y = y \bullet x$
4. (a) $x+(y \bullet z) = (x+y) \bullet (x+z)$
   (b) $x \bullet (y+z) = x \bullet y + x \bullet z$
5. (a) $x+x' = 1$ (b) $x \bullet x' = 0$
6. $|B| \geq 2$

Theorems

Thm 1: The element x' in a Boolean algebra is uniquely determined by x.

Thm 2:

(a) $x + 1 = 1$ (b) $x \bullet 0 = 0$

Thm 3: (involution)

(a) $0' = 1$ (b) $1' = 0$

Theorems (continued)

Thm 4: (both operations are idempotent)

(a) $x + x = x$ (b) $x \bullet x = x$

Thm 5: (involution)

$(x')' = x$

Thm 6: (absorption)

(a) $x + xy = x$ (b) $x \bullet (x+y) = x$

Theorems (continued)

Thm 7:

(a) $x + x'y = x + y$

(b) $x(x' + y) = xy$

Thm 8: (associativity)

(a) $(x + y) + z = x + (y + z)$

(b) $(x \bullet y) \bullet z = x \bullet (y \bullet z)$

Thm 9: (De Morgan's Law)

(a) $(x+y)'=x' \bullet y'$ (b) $(x \bullet y)'=x'+y'$

Theorems (continued)

Thm 10:

$B = \langle \{0, 1\}, +, \bullet, = \rangle$, where the two operations are defined as shown below is a Boolean algrbra.

x+y

	y	
+	0	1
x 0	0	1
1	1	1

$x \bullet y$

	y	
$\bullet$	0	1
x 0	0	0
1	0	1

Dual

Given an expression in Boolean algebra, its ***dual*** is obtained by replacing every "+" operator with "•" operator, and vice versa, and by replacing every 0 with 1, and vice versa.

Principle of duality

The dual of any true statement in Boolean algebra is also a true statement.

Perfect induction

- *Perfect induction* is proof by exhaustion in which all possibilities are considered.

The concept of a function

- It is a one-to-one or many-to-one mapping.
- The preceding truth table describes a mapping of a set of pairs of 0's and 1's into a set consisting of 0 and 1, viz.,

  $<0, 0> \rightarrow 0$ $\quad$ $<0, 1> \rightarrow 1$

  $<1, 0> \rightarrow 1$ $\quad$ $<1, 1> \rightarrow 0$
- That mapping is a function because it is many-to-one.

Truth table and function

- A truth table, therefore, describes a function.
- A truth table can be represented by using a Boolean expression, i.e., an expression written in terms of the language of Boolean algebra.

Truth table and Boolean function

- An output of a combinational circuit, which can be exhaustively listed on a column in the truth table, can also be described by a Boolean expression.
- The Boolean expression can be constructed as explained below.

Literals and product/sum terms

- A *literal* is defined as each occurrence of either a complemented or uncomplemented variable in a Boolean expression.
- A *product term* is a literal or a product (i.e., AND together, or conjunction) of literals.
- A *sum term* is a literal or a sum (i.e., OR together, or disjunction) of literals.

Minterms

- A minterm is a product term in which all variables occur exactly once, either complemented or uncomplemented.
- A minterm has the property that it is evaluated to 1 on one and only one row of the truth table.

Minterms of 2 variables

x	y	minterm that is evaluated to 1	m-notation
0	0	x'y'	m_0
0	1	x'y	m_1
1	0	xy'	m_2
1	1	xy	m_3

Minterms of 3 variables

x	y	z	minterm that is evaluated to 1	m-notation
0	0	0	x'y'z'	m_0
0	0	1	x'y'z	m_1
0	1	0	x'yz'	m_2
0	1	1	x'yz	m_3
1	0	0	xy'z'	m_4
1	0	1	xy'z	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

Truth table → Boolean expression

- The function described by a truth table can also be described by a sum (i.e., OR together, or disjunction) of minterms.
- The minterms involved are those, and only those, associated with the rows on which the minterm is evaluated to 1.

Thus the following function can be described as
f(x, y) = x'y' + xy'

x	y	f(x, y)
0	0	1
0	1	0
1	0	1
1	1	0

f(x, y, z) = x'yz' + xy'z + xyz

x	y	z	f(x, y z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Properties of minterms

- For a function of n variables, there are 2^n possible minterms.
- The sum of all possible minterms is equal to 1.
- The product of any two minterms is equal to 0 unless they are identical.

Row indices of a truth table

- It is convenient to refer to a row in a truth table by its index, which is equal to the binary equivalent of the values assigned to the variables on that row.
- Thus, for a truth table of a function of n variables, its rows are indexed by integers 0 through 2^n from top to bottom.

m-Notation

- A minterm can be denoted by m_i for short, where i is the index of the row on which that minterm is evaluated to 1.
- Thus f(x, y, z) = x'y'z' + x'yz + xyz' + xyz can be alternatively expressed as

$$f(x, y, z) = m_0 + m_3 + m_6 + m_7 = \Sigma m(0, 3, 6, 7)$$

Maxterms

- A maxterm is a sum term in which all variables occur exactly once, either complemented or uncomplemented.
- A maxterm has the property that it is evaluated to 0 on one and only one row of the truth table.

Maxterms of 2 variables

x	y	maxterm that is evaluated to 0	M-notation
0	0	x + y	M_0
0	1	x + y'	M_1
1	0	x' + y	M_2
1	1	x' + y'	M_3

Maxterms of 3 variables

x	y	z	maxterm that is evaluated to 0	M-notation
0	0	0	x + y +z	M_0
0	0	1	x + y + z'	M_1
0	1	0	x + y' + z	M_2
0	1	1	x + y' + z'	M_3
1	0	0	x' + y + z	M_4
1	0	1	x' + y + z'	M_5
1	1	0	x' + y' + z	M_6
1	1	1	x' + y' + z'	M_7

Truth table → Boolean expression

- The function described by a truth table can also be described by a product (i.e., AND together, or conjunction) of maxterms.
- The maxterms involved are those, and only those, associated with the rows on which the maxterm is evaluated to 0.

Thus the following function can be described as
$f(x, y) = (x + y')(x' + y')$

x	y	f(x, y)
0	0	1
0	1	0
1	0	1
1	1	0

$f(x, y, z) = (x + y + z)(x + y + z')(x + y' + z')(x' + y + z)(x' + y' + z)$

x	y	z	f(x, y z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Properties of maxterms

- For a function of n variables, there are 2^n possible maxterms.
- The product of all possible maxterms is equal to 0.
- The sum of any two maxterms is equal to 1 unless they are identical.

M-Notation

- A maxterm can be denoted by M_i for short, where i is the index of the row on which that maxterm is evaluated to 0.
- Thus $f(x, y, z) = (x+y+z)(x'+y+z)(x'+y'+z)$ can be alternatively expressed as

$$f(x, y, z) = M_0M_4M_6 = \Pi M(0, 4, 6)$$

Normal forms

- A Boolean function expressed in *sum-of-products form*, also known as *disjunctive normal form*
- A Boolean function expressed in *product-of-sums form*, also known as *conjunctive normal form*

Canonical forms

- Two types of expressions are directly obtainable from a truth table:

  minterm canonical formula

  (standard sum-of-products)

  maxterm canonical formula

  (standard product-of-sums)

Complementing a canonical formula

- The complement of a canonical formula is another canonical formula of the same type consisting of all missing terms.

Transformation of canonical formula

- A minterm canonical formula can be transformed into an equivalent maxterm canonical formula by
  1. complementing it by using the property of a canonical formula, and then
  2. complementing it again by applying the De Morgan's Law.

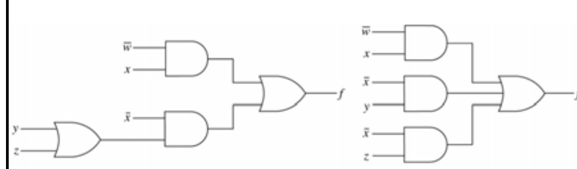
Double-rail logic

- In *double-rail logic*, it is assumed that all variables and their complements are always available as inputs.
- Otherwise, it is known as *single-rail logic*, and NOT gates have to be used to obtain the complements.
- Unless otherwise indicated, the use of double-rail logic is assumed.

Number of levels

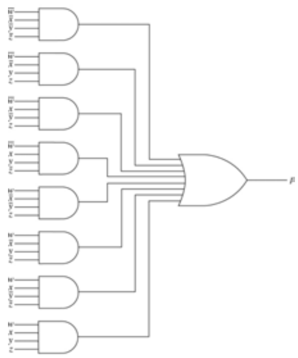
- The largest number of gates a signal must pass through from input to output is called the number of *level of logic*.

These two circuits implement the same function.

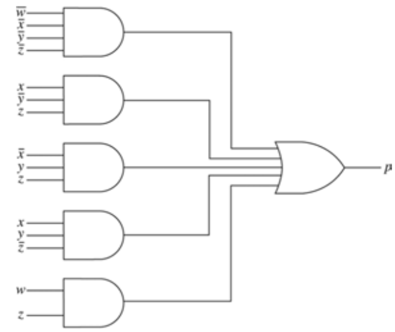


f(w, x, y, z) = w'x + x'(y + z) = w'x + x'y + x'z

2-level logic implementation of an odd-parity bit generator.



The logic diagram of a BCD odd-parity bit generator.



Universal set of operations

- A set of operations is said to be universal if any function can be described by using that set of operations alone.
- Examples:

  {AND, OR, NOT}

  {AND, NOT} {OR, NOT}

  {NAND} {NOR}

All 16 functions of 2 variables

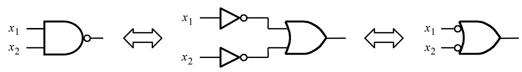
		$f_i(x, y) =$															
		i=0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	y	0	x•y (AND)	(x⊃y)'	x	(x⊂y)'	y	x⊕y (XOR)	x+y (OR)	x↓y (NOR)	x⊙y (XNOR)	y'	x⊂y	x'	x⊃y (IMPL.)	x↑y (NAND)	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

NAND and NOR functions

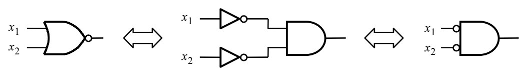
- By using current technology, the simplest logic gates are NAND and NOR gates.
- The system with operations of NAND and NOR does not constitute a Boolean algebra.
- These two operations are not associative.

NAND and OR realization

- A two-level NAND logic works exactly the same as a two-level AND-OR logic.
- A two-level NOR logic works exactly the same as a two-level OR-AND logic.
- A procedure to do NAND or NOR realization: Write the function in sum of products (or product of sums), and then use a two-level logic.

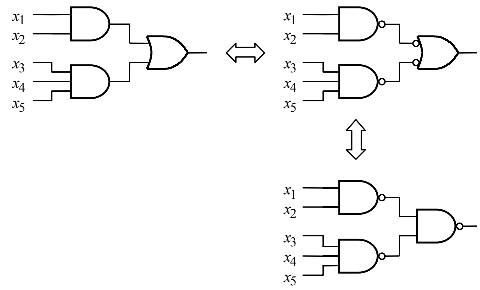


(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$

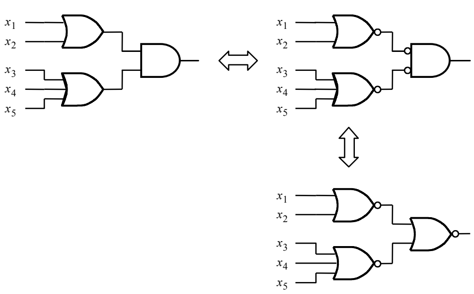


(b) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

DeMorgan's theorem in terms of logic gates

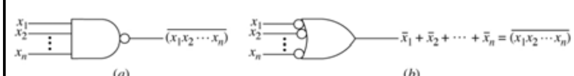


Using NAND gates to implement a sum-of-products

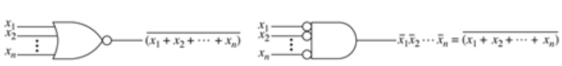


Using NOR gates to implement a product-of-sums

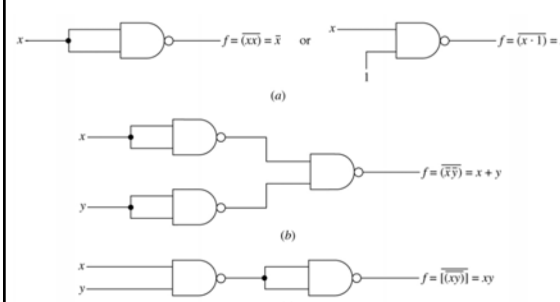
Nand-gate symbol in general



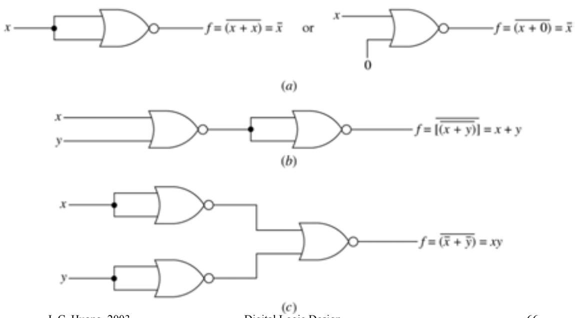
Nor-gate symbol in general



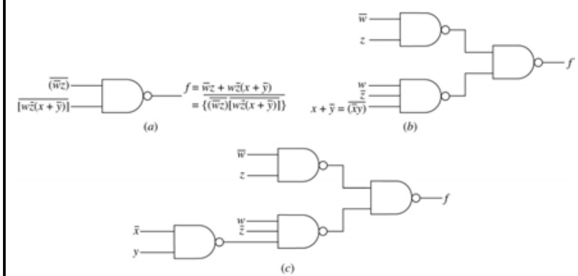
The universal property of nand-gates



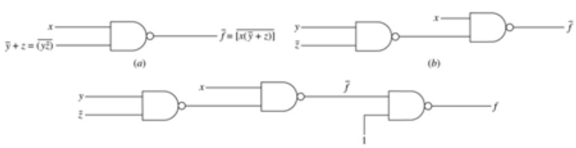
The universal property of nor-gates



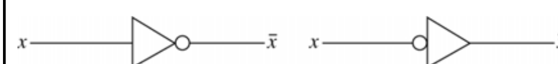
Steps involved in realizing f(w,x,y,z) = w'z + wz'(x+y')



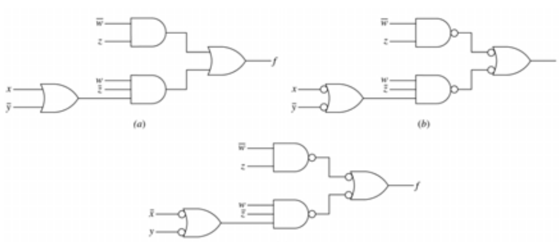
Steps involved in realizing $f(x,y,z)$ = x(y'+z)



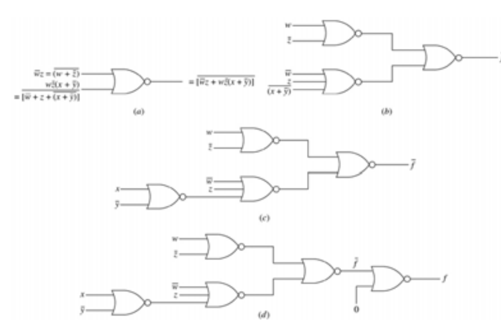
Two equivalent not-gate symbols



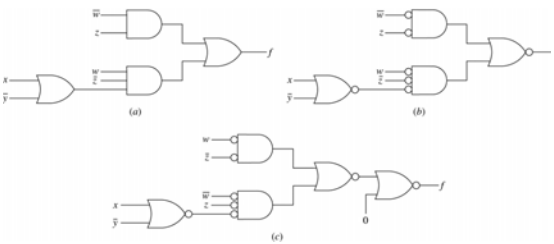
Steps involved in the realization of $f(w,x,y,z)$ = w'z + wz'(x+y')



Steps involved in realizing f(w,x,y,z) = w'z + wz'(x+y')



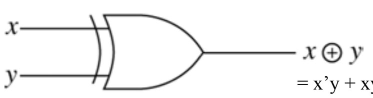
Steps involved in realizing $f(w,x,y,z) = w'z + wz'(x+y)$



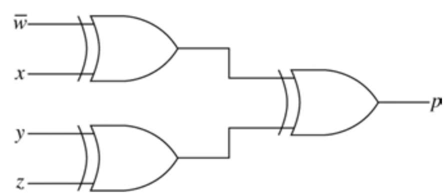
Exclusive-OR function

- It is commutative and associative
- $x \oplus 0 = x$ and $x \oplus 1 = x'$
- $x \oplus x = 0$ and $x \oplus x' = 1$
- It is also defined as "*odd* function" (meaning equal to one when the odd number of arguments are set to 1) for three or more variables.

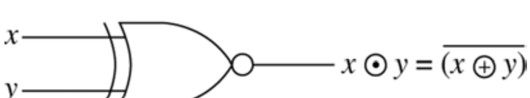
Exclusive-OR



$= x'y + xy'$



Exclusive NOR



$= x'y' + xy$
$= x \equiv y$

Gate Properties

- Voltage representation of 0 and 1
- Positive and negative logic
- Noise margins
- Fan-out
- Propagation delays
- Power dissipation

The working of a gate is commonly described by a table like the one shown below. What gate is described by this table? It depends on the type of logic used. If positive logic is used, i.e., 0 ← L and 1 ← H then it is an AND gate. If negative logic is used, i.e., 0 ← H and 1 ← H, then it is an OR gate.

input		output
x	y	
L	L	L
L	H	L
H	L	L
H	H	H

Power dissipation

- Low power dissipation and high switch speed are desired. The law of physics, however, requires high power to achieve fast switch.
- A common measure of gate performance is the product of the propagation delay and the power dissipation of the gate.