EN1013 Electronics I

Boolean Algebra

Chamira U. S. Edussooriya
B.Sc.Eng. (Moratuwa) M.A.Sc., Ph.D. (UVic)

Department of Electronic and Telecommunication Engineering
University of Moratuwa

Some of the tables and figures included in this presentation have been extracted from
Digital Design: With an Introduction to the Verilog HDL (M. M. Mano and M. D. Ciletti, Prentice Hall, 2012)

Introduction

- In order to reduce the cost of a digital system, *simpler* and *cheaper* (but *equivalent*) realizations of digital circuits are required to be derived.
- The mathematical methods employed to simplify digital circuits mostly rely on Boolean algebra.
- The algebraic system, now called Boolean algebra, is developed by George Boole in 1854.
- In 1938, Claude E. Shannon introduced a two-valued Boolean algebra called switching algebra in order to represent the properties of *bistable* electrical switching circuits.

Definition of Boolean Algebra

- Boolean algebra can be formally defined, by employing the postulates formulated by E. V. Huntington in 1904, as

  an algebraic structure defined by a set of elements, $\mathcal{B}$, together with two binary operators, $+$ and $\cdot$, provided that the following postulates are satisfied:

  1. (a) The structure is closed with respect to the operator $+$.
     (b) The structure is closed with respect to the operator $\cdot$.
  2. (a) The element 0 is an identity element with respect to $+$; that is, $x + 0 = 0 + x = x$.
     (b) The element 1 is an identity element with respect to $\cdot$; that is, $x \cdot 1 = 1 \cdot x = x$.
  3. (a) The structure is commutative with respect to $+$; that is, $x + y = y + x$.
     (b) The structure is commutative with respect to $\cdot$; that is, $x \cdot y = y \cdot x$.

4. (a) The operator $\cdot$ is distributive over $+$; that is, $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$.
   (b) The operator $+$ is distributive over $\cdot$; that is, $x + (y \cdot z) = (x + y) \cdot (x + z)$.
5. For every element $x \in \mathcal{B}$, there exists an element $\bar{x} \in \mathcal{B}$ (called the complement of x) such that (a) $x + \bar{x} = 1$ and (b) $x \cdot \bar{x} = 0$.
6. There exist at least two elements $x, y \in \mathcal{B}$ such that $x \neq y$.

- Notes:
  - A binary operator defined on a set $\mathcal{S}$ of elements is a *rule* that assigns, to each pair of elements from $\mathcal{S}$, a *unique* element from $\mathcal{S}$.
  - A set $\mathcal{S}$ is said to be closed with respect to a *binary operator* if, for every pair of elements of $\mathcal{S}$, the binary operator specifies a rule for obtaining a unique element of $\mathcal{S}$.
  - A set $\mathcal{S}$ is said to have an identity element with respect to a binary operation $*$ on $\mathcal{S}$ if there exists an element $e \in \mathcal{S}$ with the property that $e * x = x * e = x$ for every $x \in \mathcal{S}$.
  - The complement employed in Boolean algebra and the inverse employed in ordinary algebra are *not* the same.

Two-Valued Boolean Algebra

- A two-valued Boolean algebra (henceforth referred to as Boolean algebra for brevity) is defined on a set of two elements, $\mathcal{B} = \{0, 1\}$, with rules for the two binary operators $+$ and $\cdot$ as shown in the following operator tables:

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

- The rule for the complement operator is defined for verification of the postulate 5 as:

x	$\bar{x}$
0	1
1	0

- Duality principle: every algebraic expression deducible from the postulates of Boolean algebra remains valid if the *binary operators* and the *identity elements* are interchanged.
- In order to obtain the dual of an algebraic expression, we need to simply interchange $\cdot$ and $+$ binary operators and replace 1's by 0's and 0's by 1's.
- Four postulates and six basic theorems:

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Basic Theorems and Properties of Boolean Algebra *cont'd*

- Similar to postulates, a theorem has a dual.
- The postulates are the basic axioms of Boolean algebra and need no proof. The theorems must be proven from the postulates.
- Examples:
  - Prove the theorem 1(a) and 1(b).
  - Prove the theorem 2(a) and 2(b).
- The theorems involving two or three variables may be proven algebraically from the postulates and the theorems that have already been proven.
- The theorems of Boolean algebra can be proven by means of *truth tables*.
- Examples:
  - Prove the theorem 5(a) (DeMorgan's theorem) using truth tables.
  - Prove the theorem 4(b) (associative law) using truth tables.

Boolean Functions

- A Boolean function
  - expresses the logical relationship between binary variables
  - is evaluated by determining the binary value of the expression for all possible values of the variables.
- A Boolean function can be represented in a truth table.
- A Boolean function can be transformed from an *algebraic expression* into a *circuit diagram composed of logic gates* connected in a particular structure.
- Examples:
  - Draw the logic-circuit diagram for the Boolean function $F = x + \bar{y}z$.
  - Draw the logic-circuit diagram for the Boolean function $F = x\bar{y}z + x\bar{z}$.
- By manipulating a Boolean expression according to the rules of Boolean algebra, it is sometimes possible to obtain a *simpler* expression for the same function, which reduces
  - the number of gates in the circuit (because of the reduced number of terms)
  - the number of inputs to the gates (because of the reduced number of literals).

Boolean Functions *cont'd*

- Examples:
  - Simplify the Boolean functions F_1 and F_2 to a minimum number of literals; (a) $F_1 = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$ (b) $F_2 = xy + \bar{x}z + yz$.
- Karnaugh maps can be employed to manually simplify Boolean functions of up to *five* variables.
- Computer minimization programs that are capable of producing optimal circuits with millions of logic gates are used to simplify complex Boolean functions.
- The complement of a function F, denoted as $\bar{F}$, is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F.
- The complement of a function may be algebraically derived through DeMorgan's theorems, i.e.,

$$\overline{x_1 + x_2 + \cdots + x_n} = \bar{x}_1\bar{x}_2\cdots\bar{x}_n$$

$$\overline{x_1x_2\cdots x_n} = \bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_n.$$

Boolean Functions *cont'd*

- Examples:
  - Find the complements of the Boolean functions F_1 and F_2;
    (a) $F_1 = \bar{x}y\bar{z} + \bar{x}\bar{y}z$ (b) $F_2 = x(\bar{y}\bar{z} + yz)$.
- Note that the complement of a function can be derived by taking the dual of the function and complementing each literal.

Canonical and Standard Terms

- A minterm (or a standard product) having n literals is obtained from an AND term of n binary variables.
- The 2^n minterms can be obtained from the binary numbers between 0 and $2^n - 1$, with each binary variable being complemented if the corresponding bit of the binary number is 0 and uncomplemented if 1.
- A maxterm (or a standard sum) having n literals is obtained from an OR term of n binary variables.
- The 2^n maxterms can be obtained from the binary numbers between 0 and $2^n - 1$, with each binary variable being complemented if the corresponding bit of the binary number is 1 and uncomplemented if 0.
- Example:
  - Obtain the minterms and the maxterms for three binary variables.

Canonical and Standard Terms *cont'd*

Table 2.3
Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

- A Boolean function can be expressed algebraically from a given truth table by forming a minterm (maxterm) for each combination of the variables that produces a 1 (0) in the function and then taking the OR (AND) of all those terms.
- Boolean functions expressed as a sum of minterms or product of maxterms are said to be in canonical form.

- Example:
  - Consider the functions f_1 and f_2 shown in the table below. Express the function f_1 as a sum of minterms and the function f_2 as a product of maxterms.

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Canonical and Standard Terms *cont'd*

- Notation for a Boolean function $F(x, y, z)$ expressed in sum-of-minterms form:

$$F(x, y, z) = \underbrace{\Sigma}_{OR} \underbrace{(1, 4, 5, 6, 7)}_{\text{indices of minterms}}$$

- Notation for a Boolean function $F(x, y, z)$ expressed in product-of-maxterms form:

$$F(x, y, z) = \underbrace{\Pi}_{AND} \underbrace{(0, 2, 4, 5)}_{\text{indices of maxterms}}$$

- The conversion from one canonical form to another can be done by interchanging the symbols Σ and Π and listing those numbers missing from the original form.

Canonical and Standard Terms *cont'd*

- The two canonical forms very rarely express a Boolean function with the least number of literals.
- Another way to express Boolean functions is in standard form, where a term may contain one, two, or any number of literals.
- There are two types of standard forms: sum of products and products of sums.
- The logic diagram of a sum-of-products expression consists of a group of AND gates followed by a single OR gate.
- The logic diagram of a products-of-sums expression consists of a group of OR gates followed by a single AND gate.

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

for amplify the signals

Basic Digital Logic Gates *cont'd*

- Home work:
  - Read about positive and negative logic (pp. 63–65 in the textbook).