Boolean Algebra and Logic Gates

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

Basic Definitions

- The most common postulates used to formulate various algebraic structures

1. Closure : A set S is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique elements of S.
2. Associative law : $(x*y)*z=x*(y*z)$ for all $x,y,z \in S$
3. Commutative law : $x*y=y*x$ for all $x,y \in S$
4. Identity elements: for all $x \in S$, $e*x=x*e=x$
   ex) set of integers I={..., -3, -2, -1, 0, 1, 2, 3, ...}, $x+0=0+x=x$
5. Inverse : A set S having the identity element e
   for all $x \in S$, exists $y \in S$, such that $x*y=e$
6. Distributive law : $x*(y \circ z)=(x*y) \circ (x*z)$
   $*$ is distributive over $\circ$

Axiomatic Definition of Boolean Algebra

1. (a) Closure with respect to the operator +
   (b) Closure with respect to the operator ⋆
2. (a) An identity element with respect to +, designed by 0: x+0=0+x=x
   (b) An identity element with respect to ⋆, designed by 1: x⋆1=1⋆x=x
3. (a) Commutative with respect to +: x+y=y+x
   (b) Commutative with respect to ⋆: x⋆y=y⋆x
4. (a) ⋆ is distributive over +: x⋆(y+z)=(x⋆y)+(x⋆z)
   (b) + is distributive over ⋆: x+(y⋆z)=(x+y)⋆(x+z)
5. For every element x∈B, there exists an element x'∈B (called the complement of x) such that (a) x+x'=1 and (b)x⋆x'=0.
6. There exists at least two elements x,y∈B such that x≠y.

Two-valued Boolean Algebra

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	0

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

x	x'
0	1
1	0

- Show 6 postulates
- Show *distributive* law $x \cdot (y+z) = (x \cdot y) + (x \cdot z)$

Basic Theorems and Properties of Boolean Algebra

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)	$x(x + y) = x$

- Duality
- Postulates need no proof

Prove Theorems

Theorem 1(a)

$$x + x = x$$

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

Theorem 1(b)

$$x \cdot x = x$$

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

DeMorgan's Theorem

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

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

Operator Procedure

1. Parentheses
2. NOT
3. AND
4. OR

Examples

$$(x+y)'$$

$$x'y'$$

Boolean Functions

- Boolean algebra is an algebra that deals with binary variables and logic operations.
- Boolean functions consists of binary variables, the constants 0 and 1, and the logic operation symbols.
- A Boolean function can be represented in a truth table.
- A Boolean function expresses the logical relationship between binary variables.
- A Boolean functions can be transformed from an algebraic expression into a circuit diagram composed of logic gates.

$$F_1 = x + y'z$$

– The function F1 is equal to 1 if x is equal to 1 or if both y' and z are equal to 1.

– Otherwise, F1 is equal to 0.

Boolean Functions

Table 2-2
Truth Tables for F_1 and F_2

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

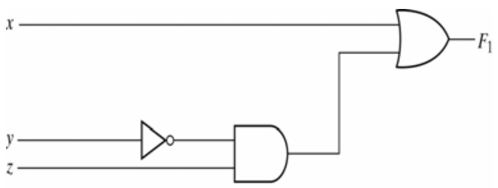
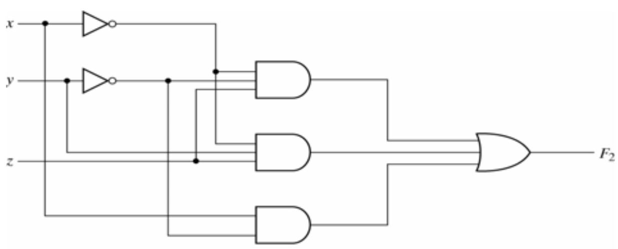


Fig. 2-1 Gate implementation of $F_1 = x + y'z$



(a) $F_2 = x'y'z + x'yz + xy'$

(b) $F_2 = xy' + x'z$

Fig. 2-2 Implementation of Boolean function F_2 with gates

- $F_1 = x + y'z$
- $F_2 = x'y'z + x'yz + xy'$
  $= x'z(y' + y) + xy'$
  $= x'z + xy'$
- Variety of algebraic form, but one truth table

Boolean Functions – Algebraic Manipulation

- Ex 2-1) Simplify the following Boolean functions to a minimum number of literals.

1. $x(x'+y) = xx' + xy = 0 + xy = xy.$
2. $x + x'y = (x+x')(x+y) = 1(x+y) = x + y.$
3. $(x+y)(x+y') = x + xy + xy' + yy' = x(1+y+y') = x.$
4. $xy + x'z + yz = xy + x'z + yz(x+x')$
   $= xy + x'z + xyz + x'yz$
   $= xy(1+z) + x'z(1+y)$
   $= xy + x'z$
5. $(x+y)(x'+z)(y+z) = (x+y)(x'+z)$: by duality from function 4.

Boolean Functions – Complement of a Function

- $(A + B + C)' = (A+x)'$ let $B+C=x$
  $= A'x'$ by theorem 5(a)(DeMorgan)
  $= A'(B+C)'$ substitute $B+C=x$
  $= A'(B'C')$ by theorem 5(a) (DeMorgan)
  $= A'B'C'$ by theorem 4(b)(associative)

=> $(A+B+C+D+...+F)' = A'B'C'D'...F'$
   $(ABCD...F)' = A' + B' + C' + D' + .. + F'$
   → Generalized form of Demorgan's theorem

Interchanging AND and OR, and complementing each literal

Examples – Complement of a Function

- Ex 2-2) Find the complement of the functions
  $F_1=x'yz'+x'y'z$, $F_2=x(y'z'+yz)$.
  $F_1' = (x'yz'+x'y'z)' = (x'yz')'(x'y'z)' = (x+y'+z)(x+y+z')$
  $F_2' = [x(y'z'+yz)]' = x'+(y'z'+yz)' = x'+(y'z')'(yz)' = x'+(y+z)(y'+z')$

- Ex 2-3) Find the complement of the functions F_1 And F_2 Ex 2-2 by taking their duals and complementing each literal.
  1. $F_1 = x'yz' + x'y'z$.
     The dual of F_1 is $(x'+y+z')(x'+y'+z)$
     Complement each literal : $(x+y'+z)(x+y+z')=F_1'$
  2. $F_2 = x(y'z'+yz)$.
     The dual of F_2 is $x+(y'+z')(y+z)$이다.
     Complement each literal : $x'+(y+z)(y'+z')=F_2'$

Examples

1. Simplify the following Boolean expressions to a minimum number of literals
   (a) $xy + xy'$ (b) $(x+y)(x+y')$
   (c) $xyz + x'y + xyz'$ (d) $(A+B)'(A'+B')'$
2. Reduce the following Boolean expressions to the indicated number of literals:
   (a) $A'C' +ABC + AC'$ to 3 literals
   (b) $(x'y'+z)'+z+xy+wz$ to 3 literals

Canonical Forms

- A binary variable x, may appear (x) or (x')
- Consider two binary variables x and y
- Combined with an AND operation
- Four possible combinations: x'y', x'y, xy', and xy

  -> is called a minterm, or a standard product
- n variables can be combined to form 2^n minterms
- In a similar fashion, n variables forming an OR term provide 2^n possible combinations, called maxterms, or standard sums.
- A Boolean function can be expressed algebraically from a given truth table by forming a minterm and then taking the OR of all those terms.

Canonical Forms

- Minterms and Maxterms

Complement

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

Canonical Forms

- Any Boolean function can be expressed as a sum of minterms (with “sum” meaning the ORing of terms).

Table 2-4
Functions of Three Variables

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

$$f_1 = x'y'z+xy'z'+xyz = m1+m4+m7$$

$$f_2 = x'yz+xy'z+xyz'+xyz = m3+m5+m6+m7$$

Canonical Forms

- Any Boolean function can be expressed as a product of maxterms (with “product” meaning the ANDing of terms).

$$f'_1 = x'y'z'+x'yz'+x'yz+xy'z+xyz'$$

$$f_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z) \quad = M_0M_2M_3M_5M_6$$

$$f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z) \quad = M_0M_1M_2M_4$$

- Boolean functions expressed as a sum of minterms or product of maxterms are said to be in canonical form.

Canonical Forms

- Ex 2-4) Express the Boolean function $F=A+B'C$ in a sum of minterms.

$$A = A(B+B') = AB + AB'$$
$$= AB(C+C') + AB'(C+C')$$
$$= ABC + ABC' + AB'C + AB'C'$$
$$B'C = B'C(A+A') = AB'C + A'B'C$$
$$F = A + B'C$$
$$= A'B'C + AB'C' + AB'C + ABC' + ABC$$
$$= m_1 + m_4 + m_5 + m_6 + m_7$$
$$= \sum(1, 4, 5, 6, 7)$$

Table 2-5
Truth Table for $F = A + B'C$

A	*B*	*C*	*F*
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

Canonical Forms

- Product of maxterms
- Ex 2-5) Express the Boolean function $F = xy + x'z$ in a product of maxterm form.

$$F = xy + x'z = (xy+x')(xy+z)$$
$$= (x+x')(y+x')(x+z)(y+z)$$
$$= (x'+y)(x+z)(y+z)$$
$$x' + y = x' + y + zz' = (x'+y+z)(x'+y+z')$$
$$x + z = x + z + yy' = (x+y+z)(x+y'+z)$$
$$y + z = y + z + xx' = (x+y+z)(x'+y+z)$$
$$F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$$
$$= M_0M_2M_4M_5$$
$$F(x, y, z) = \prod(0, 2, 4, 5)$$

Canonical Forms

- Conversion between Canonical Forms

  $F(A, B, C) = \sum(1, 4, 5, 6, 7)$

  $F'(A, B, C) = \sum(0, 2, 3) = m_0 + m_2 + m_3$

  $F = (m_0+m_2+m_3)' = m_0'm_2'm_3' = M_0M_2M_3 = \prod(0, 2, 3)$, $m_j' = M_j$

Ex) $F = xy + x'z$

$F(x, y, z) = \sum(1, 3, 6, 7)$

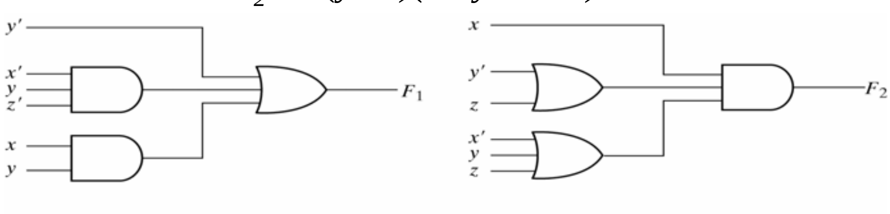
$F(x, y, z) = \prod(0, 2, 4, 5)$

Table 2-6
Truth Table for $F = xy + x'z$

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

Standard Forms

- Standard Forms: don't need to contain all the variables
  - Sum of product : $F_1 = y' + xy + x'yz'$
  - Product of sum : $F_2 = x(y'+z)(x'+y+z'+w)$



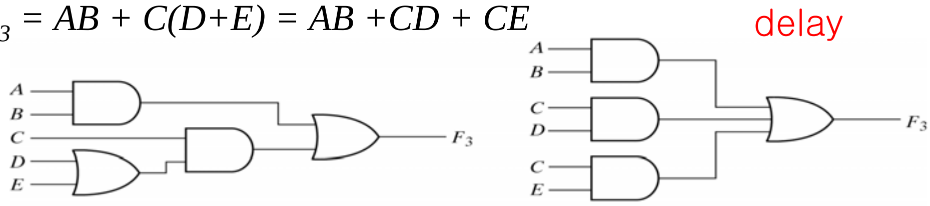
(a) Sum of Products (b) Product of Sums

Fig. 2-3 Two-level implementation

Preferred why ?

delay

- Ex) $F_3 = AB + C(D+E) = AB + CD + CE$



(a) $AB + C(D + E)$ (b) $AB + CD + CE$

Fig. 2-4 Three- and Two-Level implementation

Examples

1. Obtain the truth table of the following functions and express each function in sum of minterms and product of maxterms:

   (a) (xy + z)(y + xz)   *(b) (A' + B)(B' + C)*

   (c) y'z + wxy' + wxz' + w'x'z

2. Convert the following to the other canonical form:

$$(a)\quad F(x,y,z) = \sum(1,3,7)$$

$$(b)\quad F(A,B,C,D) = \prod(0,1,2,3,4,6,12)$$

Other Logic Operations

Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	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}
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

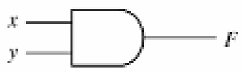
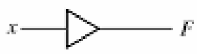
Number of possible Boolean function for n binary variables is 2^{2^n}

Other Logic Operations

Boolean Expressions for the 16 Functions of Two Variables

Boolean functions	Operator symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Digital Logic Gates

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	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

From 16 functions only eight are used as standard gates

Digital Logic Gate

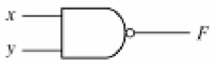
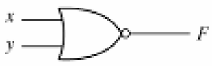
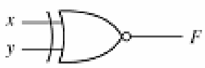
Name	Graphic symbol	Algebraic function	Truth table
NAND		$F = (xy)'$	x y F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$F = (x + y)'$	x y F 0 0 1 0 1 0 1 0 0 1 1 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y F 0 0 0 0 1 1 1 0 1 1 1 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y F 0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

Fig. 2-5 Digital logic gates

NAND and NOR are more popular than AND and OR since those can be easily constructed with TR

Digital Logic Gate–Extension to Multiple Inputs

- AND and OR are commutative and associative -> gate can be extended to multiple inputs
- But, NAND and NOR operators are not associative.

$(x\downarrow y)\downarrow z \neq x\downarrow (y\downarrow z)$

$(x\downarrow y)\downarrow z = [(x+y)'+z]'$

$= (x+y)z' = xz' + yz'$

$x\downarrow (y\downarrow z) = [x+(y+z)']'$

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

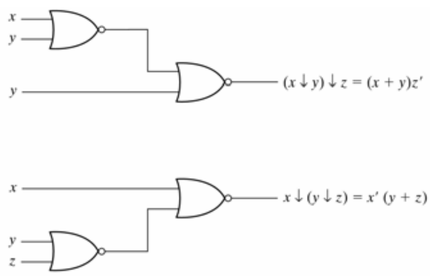


Fig. 2-6 Demonstrating the nonassociativity of the NOR operator; $(x\downarrow y)\downarrow z \neq x\downarrow (y\downarrow z)$

- To overcome, redefine

$x\downarrow y\downarrow z = (x+y+z)'$

$x\uparrow y\uparrow z = (xyz)'$

$F = [(ABC)'(DE)']' = ABC + DE$

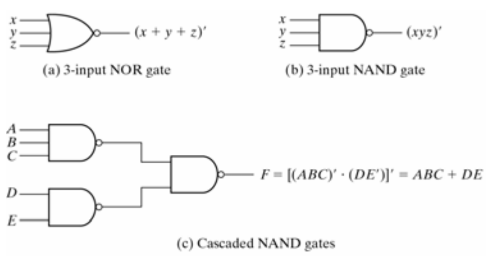
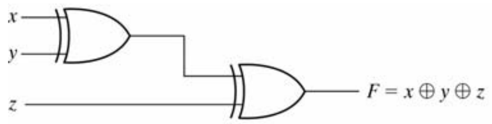


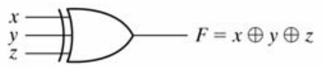
Fig. 2-7 Multiple-input and cascaded NOR and NAND gates

Digital Logic Gate

– exclusive-OR



(a) Using 2-input gates



(b) 3-input gate

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

(c) Truth table

Fig. 2-8 3-input exclusive-OR gate

Commutative,
associative
Odd function

- Positive and Negative Logic

(a) Positive logic (b) Negative logic

Fig. 2-9 signal assignment and logic polarity