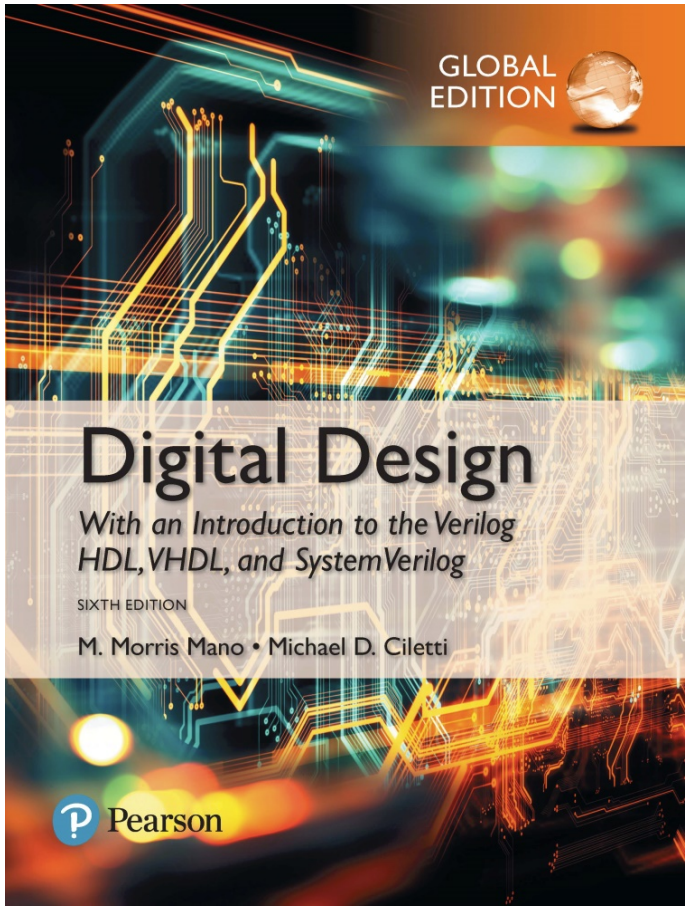
Digital Design (디지털 디자인)

With an Introduction to the Verilog HDL, VHDL, and SystemVerilog

5th or 6th Edition, Global Edition



Chapter 02

Boolean Algebra and Logic Gates

부울 대수와 논리 게이트

전자공학과 김동한 교수

Pearson

2.1 개요 & 2.2 기본 정의

- 컴퓨터는 2진 논리 사용 → 동일한 기능을 하면서 좀 더 간단(저렴)하게

→**부울대수 (Boolean algebra)** 이용

- 수학시스템: **요소**(element, 0과1), **연산자**(operator, AND와 OR와 NOT), **공리**(axiom: 증명 없이 바르다고 하는 명제)와 **공준**(postulate: 누구나 의심없이 인정할 수 있는 이치. 공리가 공준보다 더 일반적이고 당연하게 받아들여지는 사실)과 **정리**(theorem: 공리와 정의(定義)로부터 증명(證明)에 의해 유도)
- 대수(algebra)를 구성하기 위한 보편적인 공준
  - 폐쇄(closure): 요소의 모든 쌍에 대해 연산자가 요소로 대응
  - 결합법칙(associative law): (x*y)*z = x*(y*z)
  - 교환법칙(communicative law): x*y = y*x
  - 단위원(identity element): e*x = x*e = x, 역원 (inverse): x*y = e
  - 분배법칙(distributive law): x*(y·z) = (x*y)·(x*z)

2.3 부울 대수의 공리적 정리

- 1854년 George Boole이 개발. 1904년 E. V. Huntington에 의해 공식화
- 부울대수(헌팅턴의 식)

1. (a) 연산자 + 에 폐쇄적 (b) 연산자 · 에 폐쇄적
2. (a) 0은 + 에 관한 단위원 (b) 1은 · 에 관한 단위원
3. (a) + 에 관해서 교환적 (b) · 에 관해서 교환적
4. (a) · 는 + 에 관해서 분배적 (b) + 는 · 에 관해서 분배적
5. 보수(complement)가 존재 (a) $x + x' = 1$ (b) $x \cdot x' = 0$
6. 서로 다른 요소가 적어도 2개 존재

2.3 부울 대수의 공리적 정리

- 일반적인 대수와의 차이

1. 헌팅턴의 식은 결합법칙을 포함하지 않음. 하지만 유도 가능
2. · 는 + 에 관해서 분배법칙은 일반적인 대수에서는 성립 안됨
3. 부울 대수는 덧셈, 곱셈에 관한 역이 없어서 뺄셈, 나눗셈이 없음
4. 보수(complement)는 일반적인 대수에서는 없음
5. 일반적인 대수에서는 요소의 무한집합, 부울대수는 요소가 0, 1 뿐임

2.4 부울 대수의 기본 정리와 성질

- 쌍대 원리(duality principle): 헌팅턴의 식은 (a)와 (b)가 짝으로 표현

Table 2.1
Postulates(공준) and Theorems(정리) 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		$(x')' = x$			수용
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$	교환
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$	결합
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$	분배
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$	드모르간
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$	흡수

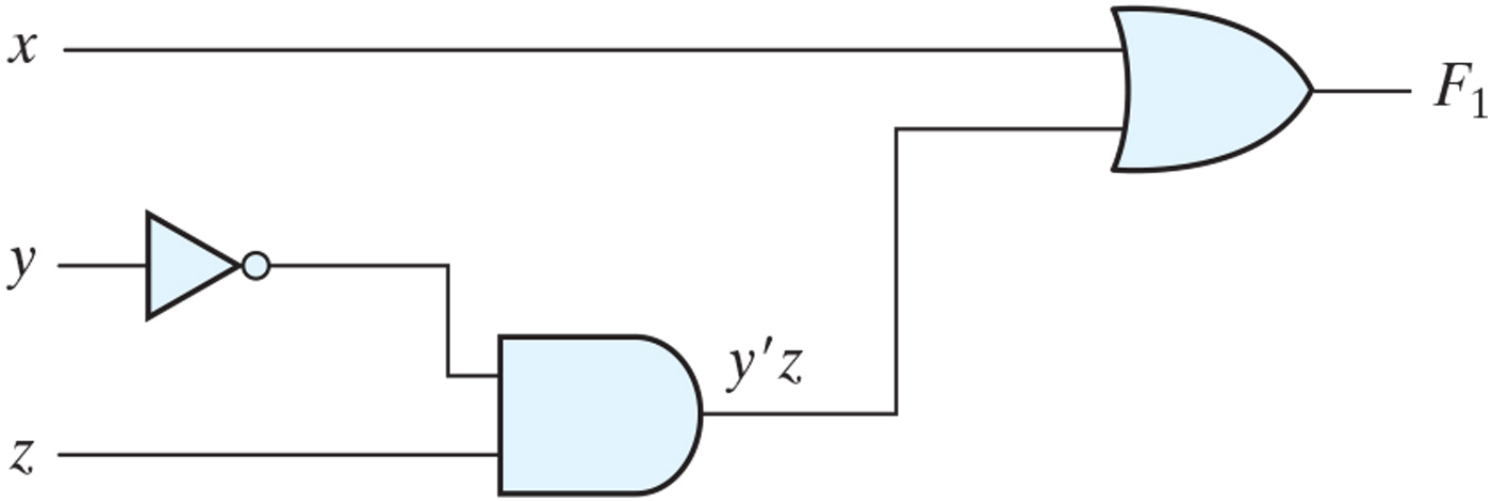
2.5 부울 함수

- 2진 변수(x, y, z)와 논리연산(AND, OR, NOT)을 다룸
- 예) $F_1 = x + y'z$
  - 함수 F_1는 x = 1 또는 y' = z = 1이면 1이고, 그렇지 않으면 0
- 2진 변수가 가질 수 있는 수는 0과 1 뿐이므로 변수들의 모든 조합을 표현하는 진리표를 이용하면 쉽게 결과를 알 수 있음

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

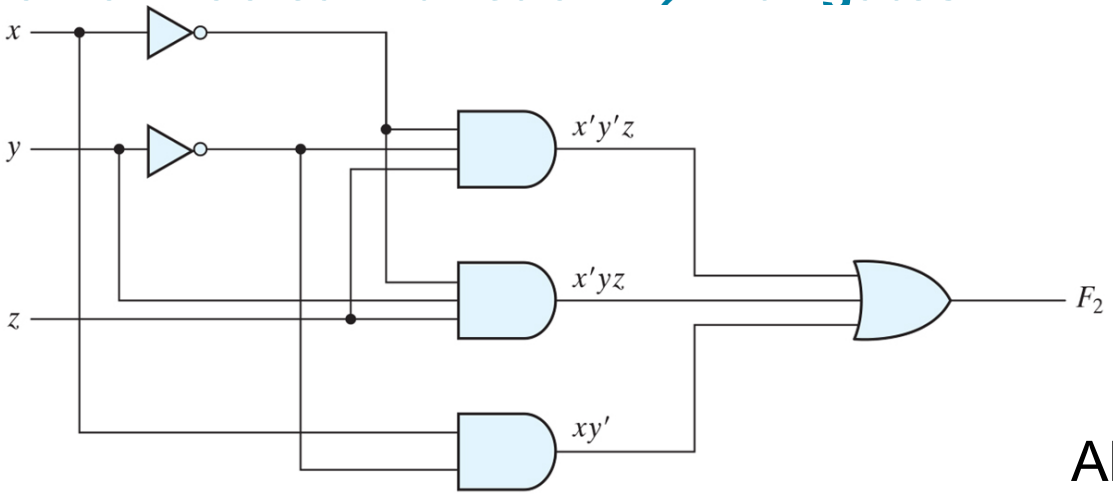
Figure 2.1
Logic diagram for the Boolean function $F_1 = x + y'z$.



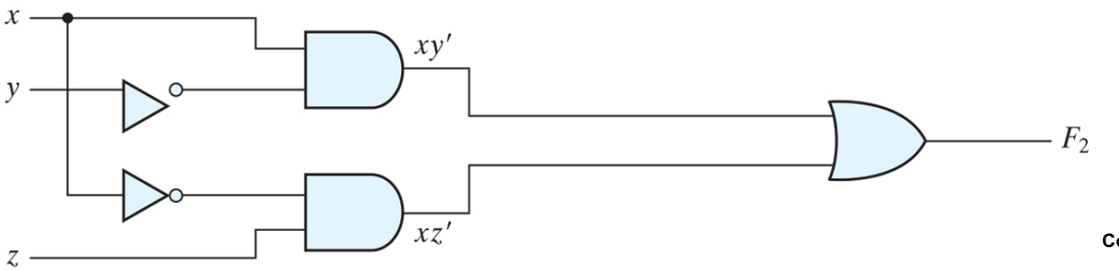
2.5 부울 함수

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

Figure 2.2
Implementation of Boolean function F_2 with gates.



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



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

AND 게이트 3→2
리터럴 8→4
항 3→2

2.5 부울 함수

• 예제2.1)

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. $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$ → 4번의 쌍대

2.5 부울 함수

- 드모르간의 법칙: 함수의 보수가 AND와 OR 연산자를 서로 바꾸고 각 리터럴을 보수화해서 얻음
  - (A + B + … + F)' = A'B'…F'
  - (AB..F)' = A' + B' + … F'
- 예제 2.2)
  - F_1 = x'yz' + x'y'z F_2 = x(y'z' + yz)의 보수?
  - 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') = x' + yz' + y'z

2.5 부울 함수

- 예제 2.3) 예제 2.2를 쌍대 원리로 구하라.
  - $F_1 = x'yz' + x'y'z$ $F_2 = x(y'z' + yz)$의 보수?
  - F_1의 쌍대 $= (x + y' + z)(x + y + z')$
  - 위의 결과에서 각 리터럴을 보수화하면, $F_1' = (x' + y + z')(x' + y' + z)$

  - F_2의 쌍대 $= x + (y' + z')(y + z)$
  - 위의 결과에서 각 리터럴을 보수화하면, $F_2' = x' + (y + z)(y' + z')$

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 2진 변수의 각 조합(AND)에서 대응되는 비트가 0이면 프라임(')을 붙이고 1이면 붙이지 않는 항을 최소항(minterm) 또는 표준곱(standard product)
- 2진 변수의 각 조합(OR)에서 대응되는 비트가 1이면 프라임(')을 붙이고 0이면 붙이지 않는 항을 최대항(maxterm) 또는 표준합(standard sum)

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

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 예) 부울 함수는 최소항의 합으로 표현 가능

$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$

$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$

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

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 예) 부울 함수의 보수를 보자. 아래 표에서 0을 만드는 조합에 대해서 최소항을 만들고 합(OR)을 구하면

$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$ → 이후 보수를 취하면

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

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

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

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 최소항의 합 또는 최대항의 곱으로 표시된 부울 함수는 정준형식임
- 최소항의 합
  - 임의의 부울 함수를 최소항의 합으로 표시하는 방법
    1. 부울 대수의 성질 이용: 예제 2.4
    2. 진리표를 만들고 최소항을 읽음: 표 2.5
- 예제 2.4) F = A + B'C를 최소항의 합으로 표시하라
  - 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 = ABC + ABC' + AB'C + AB'C' + A'B'C
    = $m_1 + m_4 + m_5 + m_6 + m_7 = \Sigma(1, 4, 5, 6, 7)$

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

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 최대항의 곱
  - 분배법칙(예를 들면, x + yz = (x + y)(x + z)) 적용 후 각 OR 항에서 임의의 누락된 변수 x를 xx'로 OR 처리해서 얻음
  - 진리표에서도 얻을 수 있음
- 예제 2.5) F = xy + x'z를 최대항의 곱으로 표시하라
  - F = xy + x'z = (xy + x')(xy + z) = (x + x')(y + x')(x + z)(y + z)
    = (x' + y)(x + z)(y + z)
  - 이 함수는 x, y, z를 갖고 있으므로 각 OR 항과 한 변수가 누락됨
    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') = $\prod(0, 2, 4, 5)$

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 정준 형식 사이의 변환
  - 최소항의 합으로 표시된 함수의 보수는 원래 함수로부터 빠진 최소항의 합과 같음

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

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

    드로르간의 정리를 쓰면

    $F(A, B, C) = (m_0 + m_2 + m_3)' = m_0'm_2'm_3' = M_0M_2M_3$

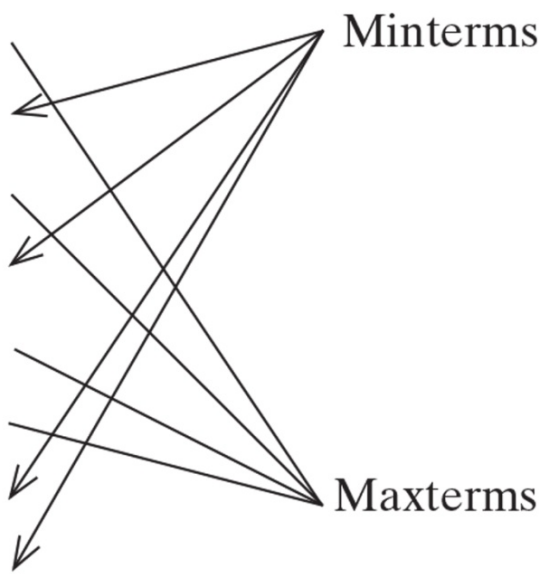
    $= \prod(0, 2, 3)$

    $m_j' = M_j$ 최대항은 최소항의 보수

  - 진리표를 이용하여 최대항의 곱을 얻는 예제는 표 2.6

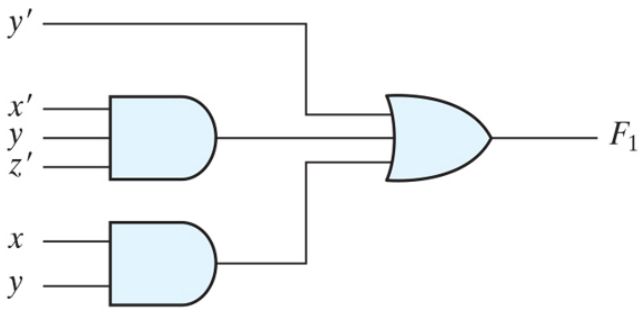
Table 2.6
Truth Table for F = xy + x'z.

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

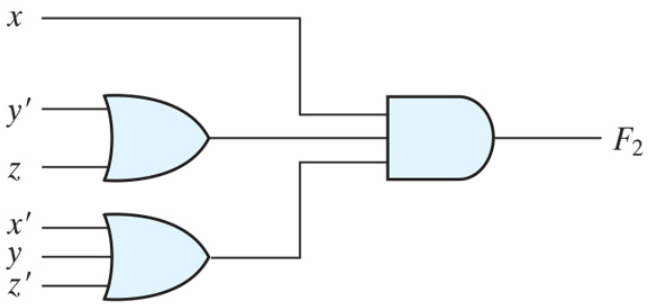


2.6 정준 형식 및 표준 형식(canonical & standard form)

- 부울 함수를 표시하는 또 다른 방법: 표준 형식 1. 곱의 합 2. 합의 곱
- 정준 형식과 표준 형식의 차이
  - 최소항: 모든 변수를 포함하는 곱의 항
  - 정준 형식인 최소항의 합 = 모든 변수를 포함하는 곱의 합
  - 최대항: 모든 변수를 포함하는 합의 항
  - 정준 형식인 최대항의 곱 = 모든 변수를 포함하는 합의 곱
- 예) $F_1 = y' + xy + x'yz'$ $F_2 = x(y' + z)(x' + y + z')$



(a) Sum of Products

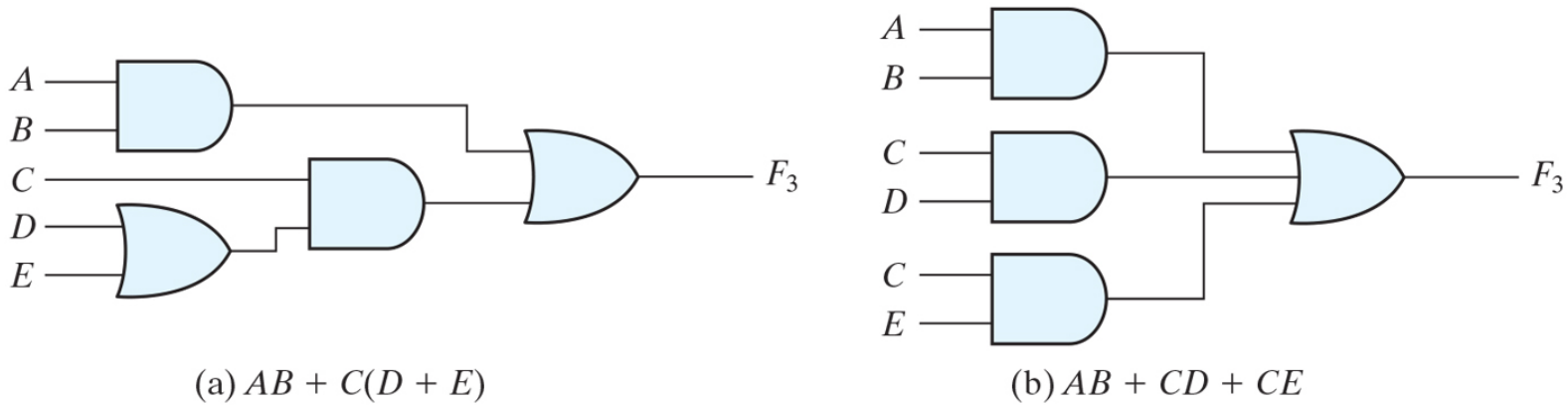


(b) Product of Sums

2.6 정준 형식 및 표준 형식(canonical & standard form)

- 표준 형식이 아닌 형태도 있음
  - 예) $F_3 = AB + C(D + E)$
    $= AB + CD + CE$

Figure 2.4
Three- and two-level implementation.



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

지연시간? 입력 개수?

2.7 기타의 논리 연산

Table 2.7
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

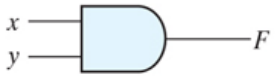
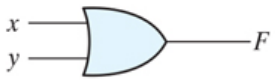
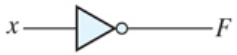
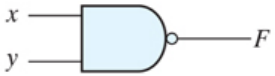
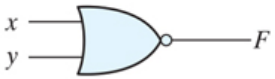
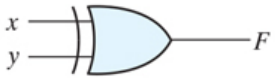
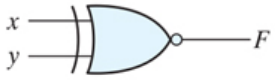
Table 2.8
Boolean Expressions for the 16 Functions of Two Variables.

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	금지
$F_5 = y$		Transfer	y	전이
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both	배타적 OR
$F_7 = x + y$	$x + y$	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	등가
$F_{10} = y'$	y'	Complement	Not y	보수
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x	함의
$F_{12} = x'$	x'	Complement	Not x	보수
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y	함의
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND	
$F_{15} = 1$		Identity	Binary constant 1	항등

2.8 디지털 논리 게이트

- 부울 함수는 AND, OR, NOT 연산의 항으로 표시되어 있으므로 게이트로 구현하기 용이함
- 앞 장의 16개 함수 중에 8개를 표준 논리 게이트로 사용함
- AND와 OR 게이트보다 NAND, NOR 게이트가 더 쉽게 구현되기 때문에 더 보편적임

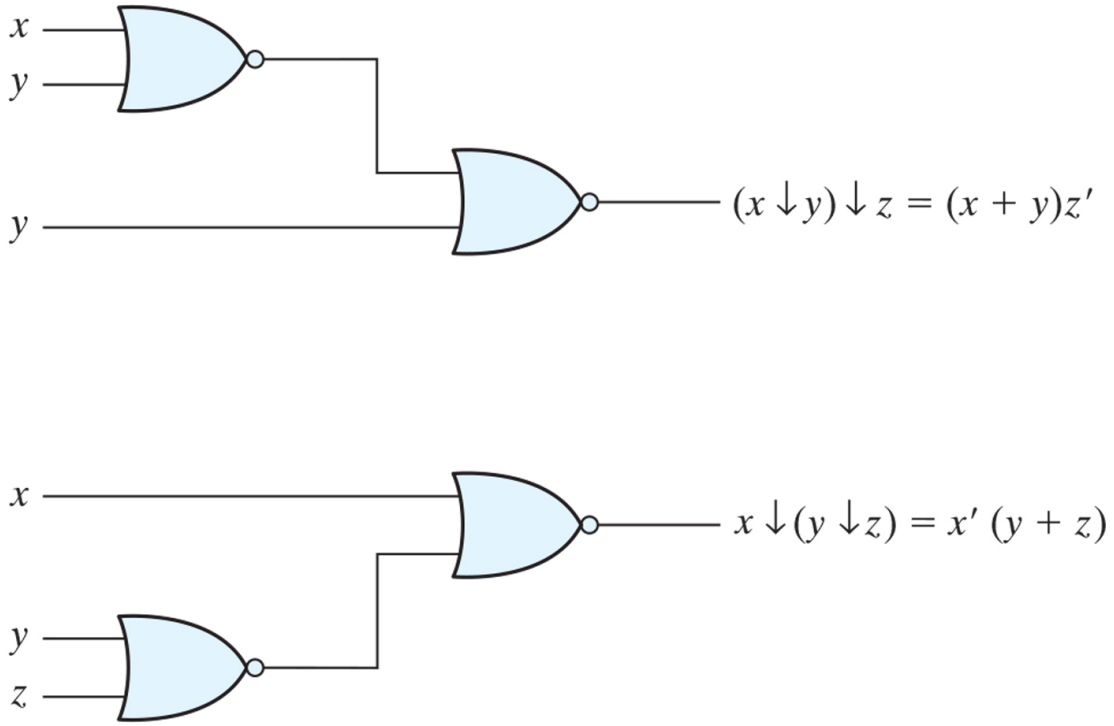
Figure 2.5
Digital logic gates.

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

2.8 디지털 논리 게이트

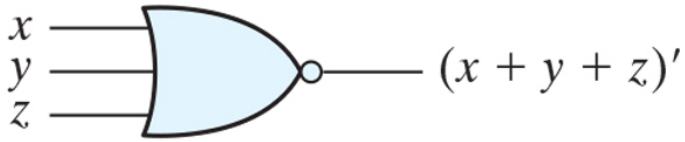
- NOR와 NAND 게이트는 결합법칙이 성립되지 않음 (그림 2.6), 3입력 NOR, NAND 게이트를 새로 정의해야 함 (그림 2.7)

Figure 2.6
Demonstrating the nonassociativity(비결합성) of the NOR operator: (x ↓ y) ↓ z ≠ x ↓(y ↓ z).

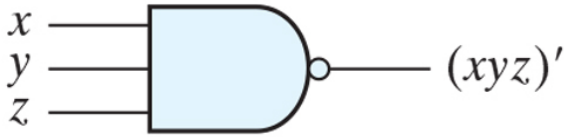


Pearson

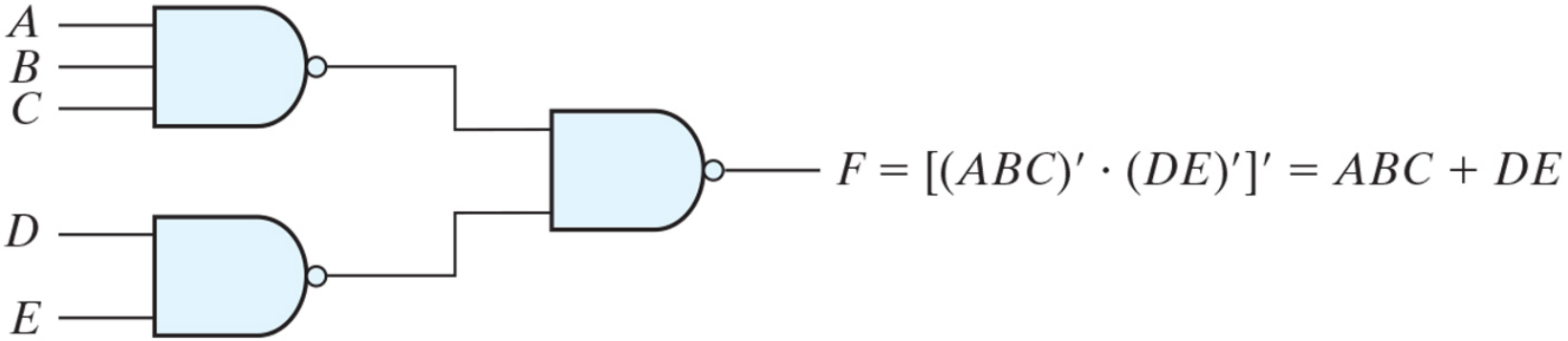
Figure 2.7
Multiple-input and cascaded NOR and NAND gates.



(a) 3-input NOR gate



(b) 3-input NAND gate

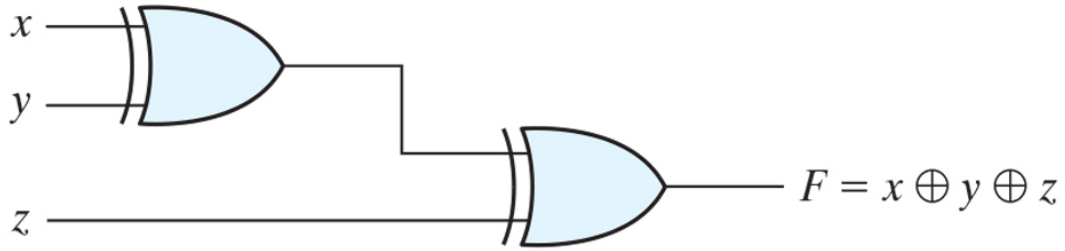


(c) Cascaded NAND gates

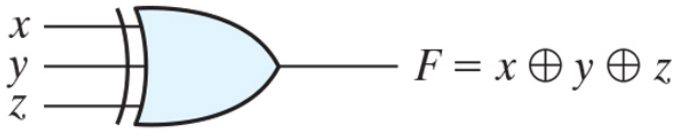
2.8 디지털 논리 게이트

- 배타적 OR 게이트는 기함수(odd function) (그림 2.8)이며, 다중 입력으로 구현하는 것이 비효율적 일 수 있음

Figure 2.8
Three-input exclusive-OR gate.



(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

2.8 디지털 논리 게이트

- 정-부 논리
  - 정논리: 논리 1을 나타내기 위해 고준위(고전압) H를 취하는 것
  - 정논리: 논리 1을 나타내기 위해 저준위(저전압) L을 취하는 것

Figure 2.9
Signal assignment and logic polarity.

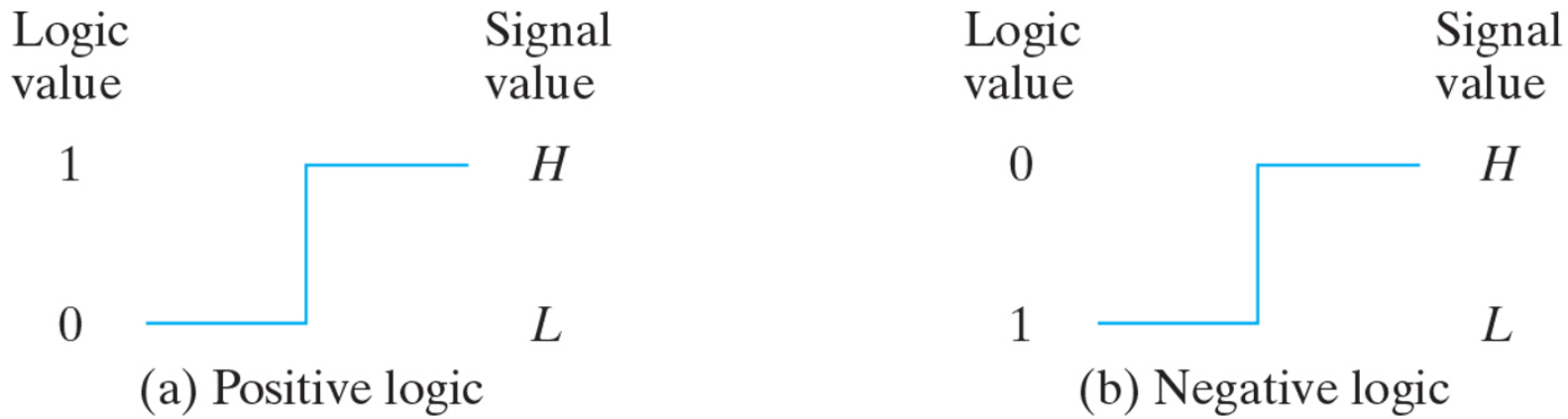
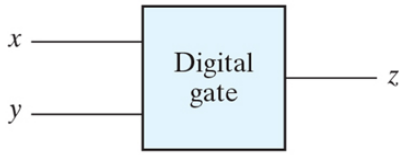


Figure 2.10
Demonstration of positive and negative logic.

x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

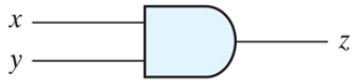
(a) Truth table with H and L



(b) Gate block diagram

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

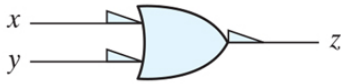
(c) Truth table for positive logic



(d) Positive logic AND gate

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

(e) Truth table for negative logic



(f) Negative logic OR gate

2.9 집적회로

- 집적의 규모
  - MSI, LSI, VLSI
- 디지털 논리군
  - TTL, ECL, MOS, CMOS
- 논리군의 파라미터
  - 팬아웃(fan-out)
  - 팬인(fan-in)
  - 전력 소산(power dissipation)
  - 전달 지연(propagation delay)
  - 잡음 여유도(noise margin)
- 알아두면 좋은 용어: ASIC, FPGA, CAD, PLD, HDL