2장 Boolean Algebra and Logic Gates

2.1 기본정의

2.2 부울대수의 공리적정의

2.3 부울대수의기본성질

2.4 부울함수

2.5 정형과 표준형

2.6 기타의 논리연산

2.7 디지탈 논리게이트

2.8 IC 디지탈 논리군

Boolean Algebra

- 1854 George Boole
- 부울대수의 가설과 정리

가설 2	$x+0 = x$	$x \cdot 1 = x$
가설 5	$x+x' = 1$	$x \cdot x' = 0$
정리 1	$x+x = x$	$x \cdot x = x$
정리 2	$x+1 = 1$	$x \cdot 0 = 0$
정리 3, 누승	$(x')' = x$	$(x')' = x$
가설 3, 교환	$x+y = y+x$	$xy = yx$
정리 4, 결합	$x+(y+z) = (x+y)+z$	$x(yz) = (xy)z$
가설 4, 분배	$x(y+z) = xy+xz$	$x+yz = (x+y)(x+z)$
정리 5, De Morgan	$(x+y)' = x'y'$	$(xy)' = x'+y'$
정리 6, 흡수	$x+xy = x$	$x(x+y) = x$

Duality Principle

- AND⇔OR, 0⇔1
- **정리의 증명에 유용**

  "어떠한 정리가 성립하면 그 쌍대도 역시 성립한다" (증명필요 없음)

  (ex) x+xy = x

  윗 식이 성립하면 그의 쌍대인 x(x+y) = x **도 당연히 성립함**

- **부울함수의 보수를 구하는데 유용**

  "쌍대를 취한후 각 변수를 부정"

  (ex) **부울함수** f = x(y′z′+yz) **의 보수** f′ **를 구하시오**

  dual of f = x+(y′+z′)(y+z)

  therefore, f′ = x′+(y+z)(y′+z′)

정리의 증명

1. 가설과 정리를 이용

   (ex) x+yz = (x+y)(x+z)

   우변 = xx+xz+xy+yz = x1+xz+xy+yz = x(1+z+y)+yz = x+yz

   = 좌변

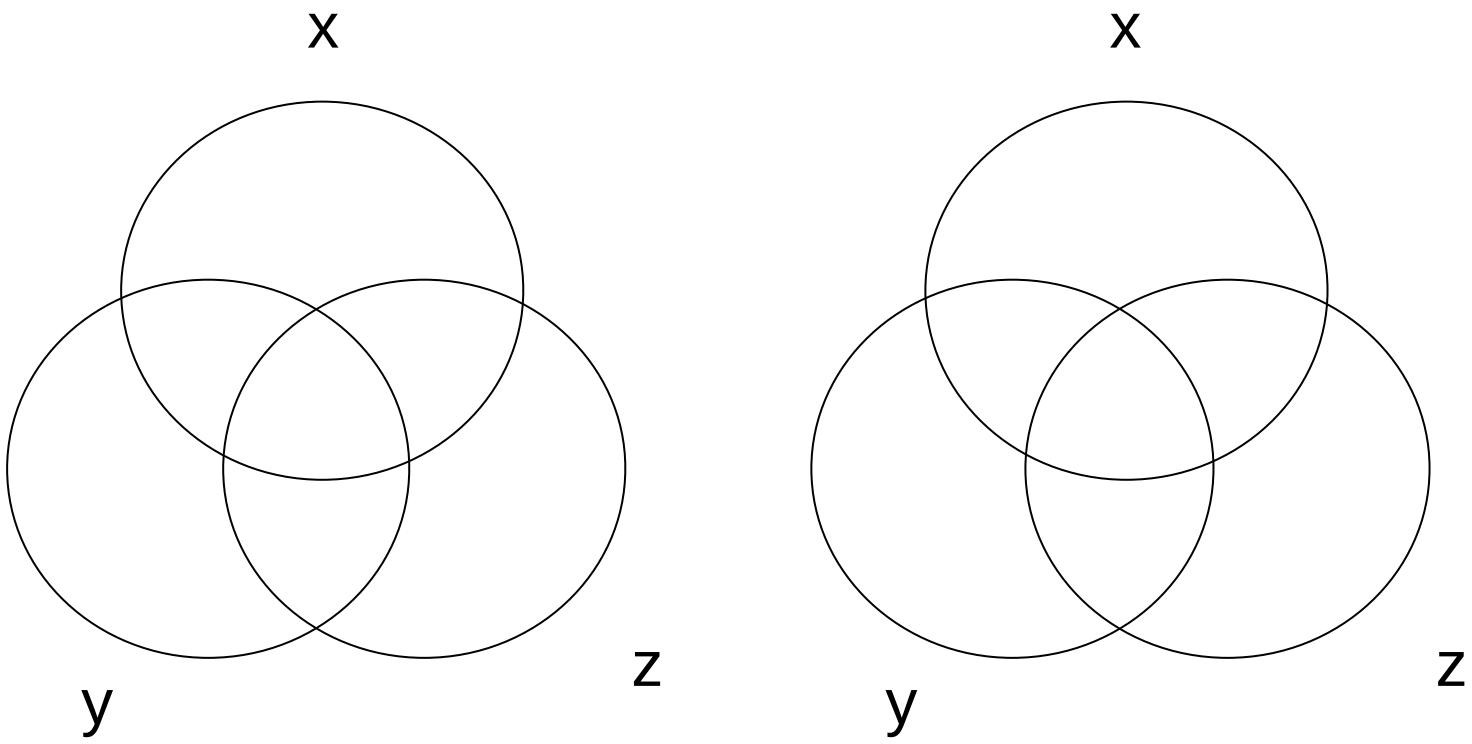
2. 진리표이용

x	y	z	yz	x+yz	x+y	x+z	(x+y)(x+z)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

정리의 증명 (계속)

3. Venn Diagram 이용

(ex) x+yz = (x+y)(x+z)



Boolean Function

- 연산자 우선순위 : ① 괄호 ② NOT ③ AND ④ OR
- 부울함수

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

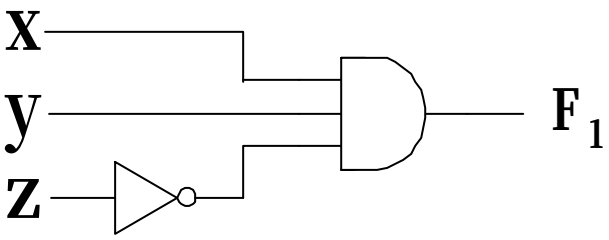
$F_1 = xyz'$

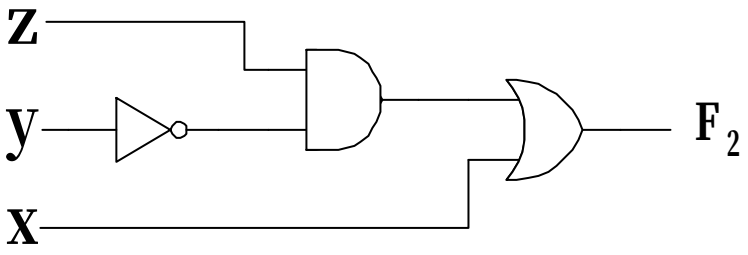
$F_2 = x'y'z + xy'z' + xy'z + xyz' + xyz$

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

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

$= x + y'z$





Boolean Function

x	y	z	F_3	F_4
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

$F_3 = x'y'z + x'yz + xy'z' + xy'z$

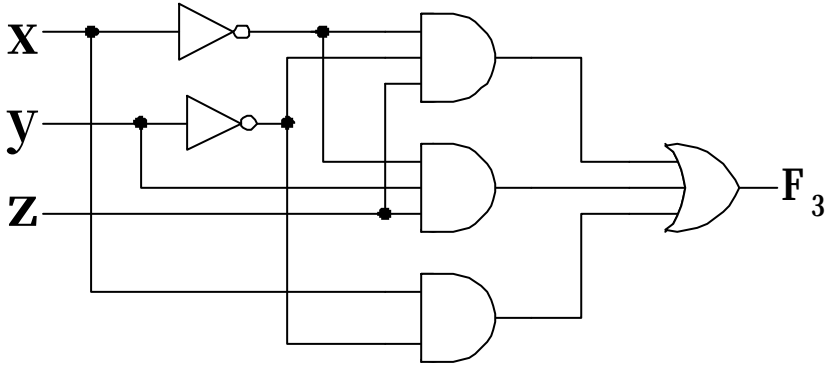
$= x'y'z + x'yz + xy'(z' + z)$

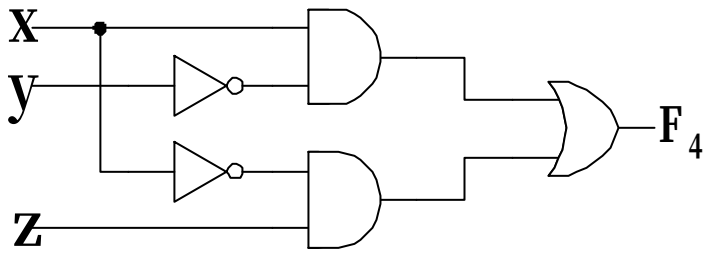
$= x'y'z + x'yz + xy'$

$F_4 = x'y'z + x'yz + xy'z' + xy'z = x'y'z + x'yz + xy'(z' + z)$

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

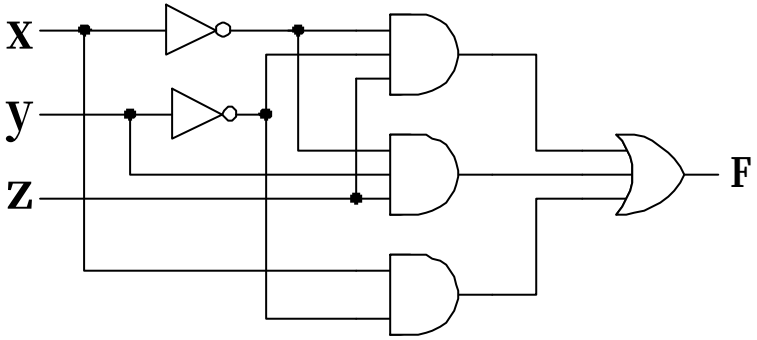
$= x'z + xy'$

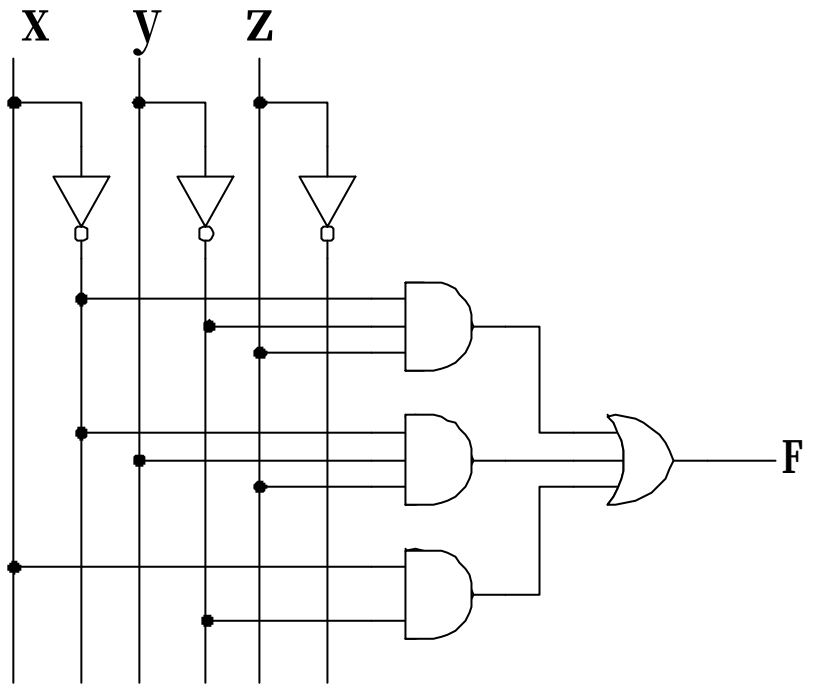




Gate Diagram

F = x' y' z+x' yz+xy'



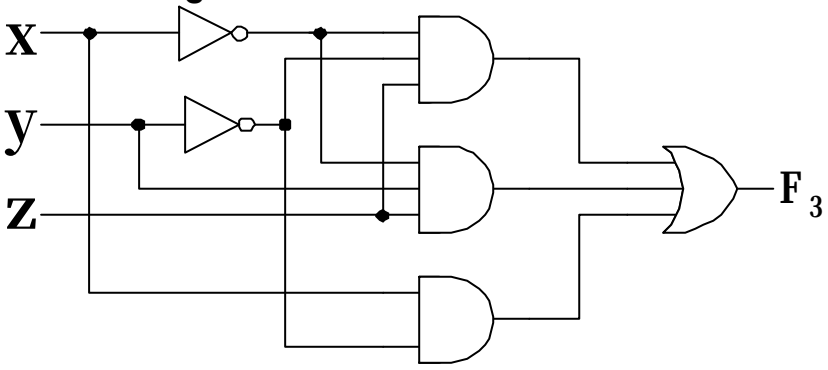


Minimization of Boolean Function

- 적은 갯수의 문자와 게이트를 갖는 부울함수식을 찾는 문제
- 함수식의 간소화 → **literal**수의 감소 → 입력의 수와 게이트 수의 감소
  → 설계비용의 절감, 회로의 면적감소 → 경제적

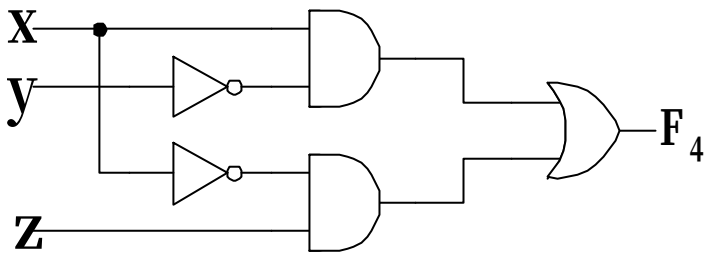
※ Literal : prime이 붙거나 안 붙은 변수

$F_3 = x'y'z + x'yz + xy'$



- 게이트 수 : 6개
  2-입력 AND : 1, 3-입력 AND : 2
  3-입력 OR : 1, NOT : 2
- Literal수 : 8개

$F_4 = x'z + xy'$



- 게이트 수 : 5개
  2-입력 AND : 2, 2-입력 OR : 1,
  NOT : 2
- Literal수 : 4개

Complement of Boolean Function

1. De Morgan의 정리이용

(ex) $f_1 = x'yz' + x'y'z$

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

$= (x'yz')'\,(x'y'z)'$

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

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

$f_2' = [x(y'z' + yz)]'$

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

$= x' + (y'z')'\,(yz)'$

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

2. Duality 이용

(ex) $f_1 = x'yz' + x'y'z$

dual of $f_1 = (x'+y+z')(x'+y'+z)$

therefore, $f_1' = (x+y'+z)(x+y+z')$

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

dual of $f_2 = x+(y'+z')(y+z)$

therefore, $f_1' = x' + (y+z)(y'+z')$

Canonical Form : sum of minterms, product of maxterms

- n bit로 2^n개의 조합을 표현할 때,
  - minterm : 각 변수를 AND로 결합하여 결과가 "1"이 되게 함
  - Maxterm : 각 변수를 OR로 결합하여 결과가 "0"이 되게 함
- 입력변수가 3개인 경우

	Minterm		Maxterm	
x y z	항	표시	항	표시
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

$m_i' = M_i$

$M_i' = m_i$

Minterm & Maxterm

			m_0	m_4		M_0	M_4	
x	y	z	$x'y'z'$	$xy'z'$	m_0+m_4	x+y+z	$x'+y+z$	M_0M_4
0	0	0	1	0	1	0	1	0
0	0	1	0	0	0	1	1	1
0	1	0	0	0	0	1	1	1
0	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	0	0
1	0	1	0	0	0	1	1	1
1	1	0	0	0	0	1	1	1
1	1	1	0	0	0	1	1	1

$m_k = \overline{M_k}$

$m_0+m_4 = \overline{M_0M_4}$

$m_0+m_4 = \text{å}\ (0,4) = \tilde{O}\ (1,2,3,5,6,7)$

0,4번째 "1" = 1,2,3,5,6,7번째 "0"

$M_0M_4 = \tilde{O}\ (0,4) = \text{å}\ (1,2,3,5,6,7)$

1,2,3,5,6,7번째 "1" = 0,4번째 "0"

Minterm and Maxterm

x	y	z	f_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7 = \Sigma(1,4,7)$

$f_1' = (m_1 + m_4 + m_7)' = m_1' m_4' m_7' = M_1 M_4 M_7 = \Pi(1,4,7)$

$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz' = m_0 + m_2 + m_3 + m_5 + m_6$

$= \Sigma(0,2,3,5,6)$

$f_1 = (f_1')' = (m_0 + m_2 + m_3 + m_5 + m_6)' = m_0' m_2' m_3' m_5' m_6'$

$= M_0 M_2 M_3 M_5 M_6 = \Pi(0,2,3,5,6)$

Therefore,

$f_1 = \Sigma(1,4,7) = \Pi(0,2,3,5,6)$

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

$f_1' = \Sigma(0,2,3,5,6) = \Pi(1,4,7)$

$= x'y'z' + x'yz' + x'yz + xy'z + xyz' = (x+y+z')(x'+y+z)(x'+y'+z')$

Minterm & Maxterm

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

g = å(0,4) = x' y' z' +xy' z'
∴ 0,4번째항이 "1"
∴ 1,2,3,5,6,7번째항이 "0"

g = å (0,4)
= Õ (1,2,3,5,6,7)

x	y	z	g'
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

g' = Õ (0,4) = (x+y+z)(x' +y+z) = y+xz+x' y+y+yz+x' z+yx+z
= y(x+x' +1+z+x)+z(x+x' +1)+xy = y+z+xy = y(1+x)+z
= y+z
∴ 0,4번째항이 "0"
∴ 1,2,3,5,6,7번째항이 "1"

g' = Õ (0,4)
= å (1,2,3,5,6,7)

Minterm and Maxterm

x	y	z	f_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

f_2 = å ( ) : sum of minterm

= Õ ( ) : product of maxterm

f_2' = å ( ) : sum of minterm

= Õ ( ) : product of maxterm

Sum of Minterm으로의 변환

$f_1(A,B,C) = A+B'C$

1. 식의 변환

$f_1 = A+B'C$

$= A(B+B')+B'C(A'+A)$

$= AB+AB'+A'B'C+AB'C$

$= AB(C+C')+AB'(C+C')+A'B'C+AB'C$

$= ABC+ABC'+AB'C+AB'C'+A'B'C+AB'C$

$= ABC+ABC'+AB'C+AB'C'+A'B'C$

$= m_7+m_6+m_5+m_4+m_1$

$= \Sigma(1,4,5,6,7)$

2. 진리표이용

A	B	C	f_1
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

$f_1 = m_1 + m_4+m_5+m_6+ m_7$

$= \Sigma(1,4,5,6,7)$

Product of Maxterm으로의 변환

$f_1(A,B,C) = A+B'C$

1. 식의 변환

$f_1 = A+B'C$

$= (A+B')(A+C)$

$= (A+B'+CC')(A+C+BB')$

$= (A+B'+C)(A+B'+C')(A+C+B)(A+C+B')$

$= (A+B'+C)(A+B'+C')(A+B+C)(A+B'+C)$

$= (A+B'+C)(A+B'+C')(A+B+C)$

$= M_2M_3M_0$

$= \tilde{O}(0,2,3)$

2. 진리표이용

A	B	C	f_1
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

$f_1 = m_1 + m_4+m_5 +m_6+ m_7$

$= \mathring{a} (1,4,5,6,7)$

$= \tilde{O} (0,2,3)$

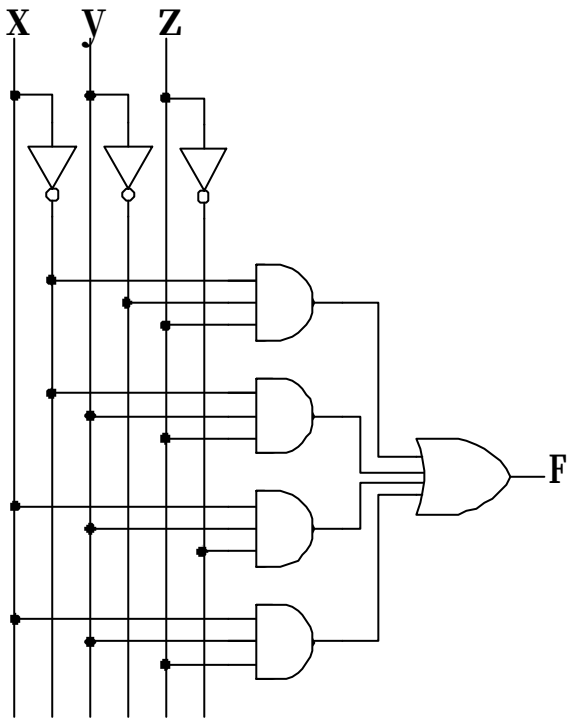
Sum of Minterm, Product of Maxterm으로의 변환

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

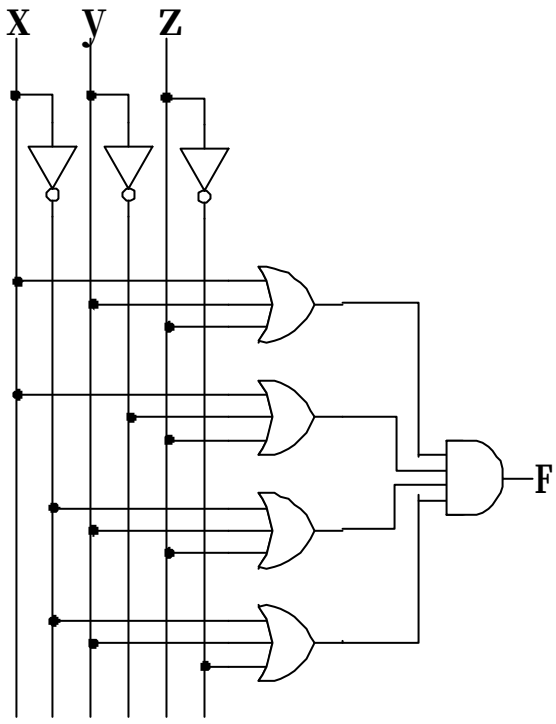
x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

f = å (1,3,6,7)
= Õ (0,2,4,5)

f = å (1,3,6,7) = x' y' z+ x' yz+ xyz' + xyz
f = Õ (0,2,4,5) = (x+y+z) (x+y' +z) (x' +y+z) (x' +y+z')



f = å (1,3,6,7)



f = Õ (0,2,4,5)

Standard Form : sum of products, product of sums

- **Sum of products : 변수들의 곱(AND)을 합(OR)한 형태**

  (ex) f = y' +xy+x' yz

- **Product of sums : 변수들의 합(OR) 을 곱(AND)한 형태**

  (ex) f = x(y' +z)(x+y' +z)

※ canonical form(sum of minterm, product of maxterm)도 standard form에 포함

- **비표준형 : 곱의 합형태와 합의 곱형태가 혼재**

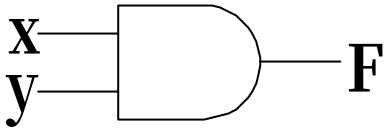
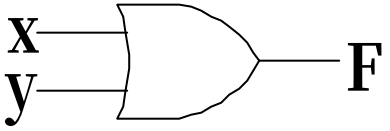
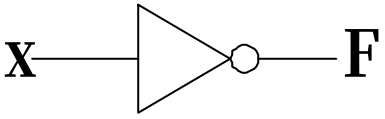
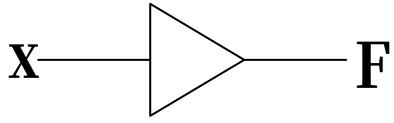
  → 표준형으로 변환 가능

(ex) f = (ab+cd)(a' b' +c' d')

= aba' b' +abc' d' +a' b' cd+cdc' d'

= abc' d' +a' b' cd

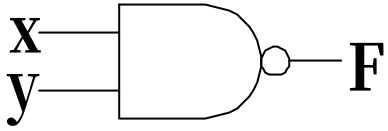
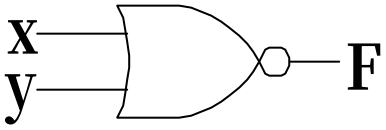
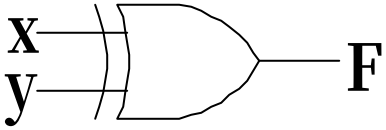
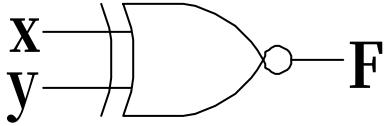
Digital Logic Gates

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
NOT (inverter)		F = x′	X | F 0 | 1 1 | 0
Buffer		F = x	X | F 0 | 0 1 | 1

모두 “1” → “1”

모두 “0” → “0”

Digital Logic Gates

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	모두 "1" → "0"
NOR		F = (x+y)'	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	모두 "0" → "1"
XOR		F = x' y+xy' = x ⊕y	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	다르면 → "1"
XNOR		F = xy+x' y' = x ⊙y	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1	같으면 → "1"

교환/결합법칙

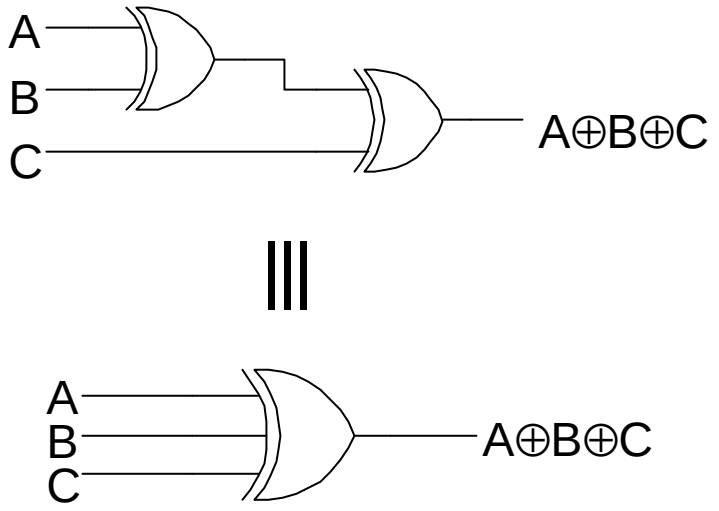
- AND, OR, XOR, XNOR : 교환/결합법칙 모두 성립
- NAND, NOR : 교환법칙은 성립하나 결합법칙은 성립하지 않음

	교환법칙	결합법칙
AND	$xy = yx$	$(xy)x = x(yz)$
OR	$x+y = y+x$	$(x+y)+z = x+(y+z)$
XOR	$x \oplus y = y \oplus x$	$(x \oplus y) \oplus z = x \oplus (y \oplus z)$
XNOR	$x \odot y = y \odot x$	$(x \odot y) \odot z = x \odot (y \odot z)$
NAND	$(xy)' = (yx)'$	$[(xy)'z]' \neq [x(yz)']'$
NOR	$(x+y)' = (y+x)'$	$[(x+y)'+z]' \neq [x+(y+z)']'$

XOR Gate

❑ **XOR 게이트는 기함수(odd function) : “1”의 갯수가 홀수일때 “1”출력**

A	B	C	A⊕B	A⊕B⊕C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1



IC Digital Logic Family

- RTL (Resistor Transistor Logic)
- DTL (Diode Transistor Logic)
- TTL (Transistor Transistor Logic) : 보편적으로 널리 쓰임
- ECL (Emitter Coupled Logic) :고속 동작에 적합
- MOS (Metal Oxide Semiconductor) : 고집적도
- CMOS (Complementary Metal Oxide Semiconductor) : 고집적도, 저전력
- I^2L (Integrated Injection Logic) : 고집적도

Positive Logic and Negative Logic

❑ **Positive logic (양논리)**

❑ **Negative logic (음논리)**

Positive logic AND

X Y	F
0 0	0
0 1	0
1 0	0
1 1	1

Positive logic OR

X Y	F
0 0	0
0 1	1
1 0	1
1 1	1

Negative logic AND

X Y	F
0 0	0
0 1	1
1 0	1
1 1	1

Negative logic OR

X Y	F
0 0	0
0 1	0
1 0	0
1 1	1

❑ **Positive logic AND = Negative logic OR**

❑ **Positive logic OR = Negative logic AND**

IC Logic Family의 특성평가

- **Fanout**
  정상동작에 영향을 주지 않고, 게이트의 출력에 걸어 줄 수 있는 부하의 갯수

- **전력소비(Power dissipation)**
  게이트를 동작시키기 위해 필요한 전력

- **지연시간(Propagation delay)**
  입력측의 신호가 출력측에 전달되기 까지 걸리는 시간

- **잡음여유(Noise margin)**
  입력신호에 포함될 수 있는 잡음의 최대 허용치