디지털논리회로
(Digital Logic Circuit)

- Chapter 2

앞으로 공부할 것

Chapter 1 Introduction
컴퓨터에서의 수 체계 (이진수 + alpha)

Chapter 2 Combinational Systems
진리표로 구현하는 combinational systems
BooI 대수로 표현되는 진리표
게이트로 표현되는 진리표
입력→Combinational systems→출력

Chapter 3 The Karnaugh Map (K-map)
진리표를 map으로 표시 → 단순화, 최소화 → 최적의 시스템 만들기

Chapter 5 Designing Combinational Systems
Combinational system(=진리표)로 만들 수 있는 *'그럴듯한'* 실제 시스템들
- adders, comparators, decoders, encoders, multiplexers..

Chapter 2에서 여러분은

Specification: 시스템 기능을 진리표로 작성할 수 있다

Description: 진리표를 스위칭 대수로 표현할 수 있다

Simplification: 스위칭 대수 특성 이용하여 최적화할 수 있다

Implementation: 게이트를 이용하여 설계할 수 있다

Chapter 2 Combinational Systems

2.1 조합회로 시스템 설계 과정

2.2 스위칭 대수 (Switching Algebra)

2.3 AND, OR, NOT 게이트에 의한 함수 구현

2.4 보수 (The Complement)

2.5 진리표로부터 대수적 표현

2.6 NAND, NOR, XOR 게이트

2.7 대수식의 간소화

2.8 대수 함수의 조작과 NAND게이트 구현

2.9 일반적인 부울 대수(Boolean Algebra)

Chapter 2 Combinational Systems

2.1 조합회로 시스템 설계 과정

2.2 스위칭 대수 (Switching Algebra)

2.3 AND, OR, NOT 게이트에 의한 함수 구현

2.4 보수 (The Complement)

2.5 진리표로부터 대수적 표현

2.6 NAND, NOR, XOR 게이트

2.7 대수식의 간소화

2.8 대수 함수의 조작과 NAND게이트 구현

2.9 일반적인 부울 대수(Boolean Algebra)

조합회로 시스템 설계 과정

1 단계: 각 **입력과 출력**을 2진으로 표현하라.

1.5 단계: 필요하면, 문제를 더 작은 부(sub) 문제로 나누어라.

2 단계: 설계 사양을 **진리표 혹은 대수 식**으로 형식화(formalize)해라.

3 단계: 서술을 **간단히** 하라.

4 단계: 설계 **목표와 제약에 근거**하여,

사용 가능한 부품으로 시스템을 **구현**한다.

두 입력 신호가
모두 1일 때만
출력 신호가 1이다.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Y = A + B

조합회로 시스템 설계 과정

입력과 출력 찾아보기

예문 1: 4개의 입력 A, B, C, D 와 출력 Z를 갖는 시스템은 입력 중 3개가 1 이면 Z = 1 이다

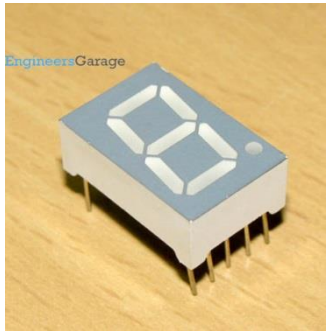
예문 2. (온/오프 되는) 전등은 3개의 스위치 중 어느 한 개로 제어할 수 있다. 스위치 한 개는 마스터 on/off 스위치이다. 만일 마스터 스위치가 off 되면 전등은 차단된다. 마스터 스위치가 on 이 될 때 다른 스위치들 중 한 개의 스위치가 (위에서 아래 혹은 아래에서 위로) 바뀌면 전등의 상태가 바뀐다.

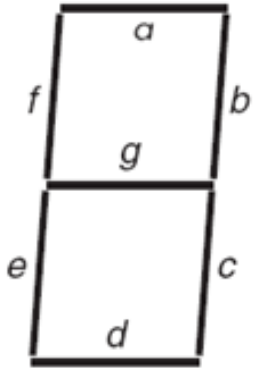
예문 3. 시스템은 1비트 2진 덧셈을 한다. 시스템은 3개의 입력 (더하려고 하는 2 비트와 다음 하위 비트로부터의 캐리)을 가지며 합 비트와 다음 상위 비트로 들어가는 캐리에 해당하는 2개의 출력을 제공한다.

조합회로 시스템 설계 과정

입력과 출력 찾아보기

예문 4. 시스템은 십진 숫자에 대한 코드를 입력으로 받고 대부분의 디지털 시계와 숫자 표시에 사용되는 7-세그먼트 디스플레이를 구동하는 신호를 출력으로 제공한다.





예문 5. 시스템은 2개의 4 비트 2진수와 캐리 입력을 나타내는 9개의 입력과, 합을 표현하는 5 비트 출력 1개를 갖는다. (각 입력 수는 0에서 15까지의 범위이고 출력은 0에서 31까지의 범위이다.)

Don’t care 조건

1 단계: 각 **입력과 출력**을 2진으로 표현하라.

어떨 때는 특정한 입력조합에 대해 출력은 정해져 있지 않음
Don’t care 조건: **X 로 표시**
0 또는 1로 해석

a	b	f
0	0	0
0	1	1
1	0	1
1	1	X

a	b	f_1	f_2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

Chapter 2에서 여러분은

Specification: 시스템 기능을 진리표로 작성할 수 있다

Description: 진리표를 스위칭 대수로 표현할 수 있다

Simplification: 스위칭 대수 특성 이용하여 최적화할 수 있다

Implementation: 게이트를 이용하여 설계할 수 있다

Truth Table(진리표) 작성하기

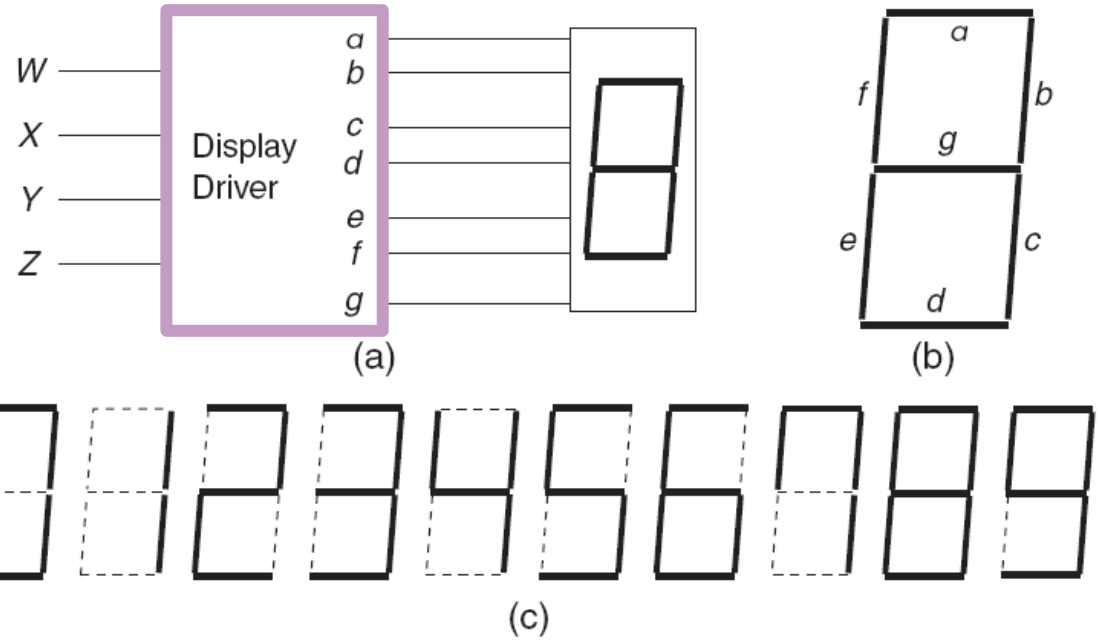
예문 1: 4개의 입력 A, B, C, D 와 출력 Z를 갖는 시스템은 입력 중 3개가 1이면 Z = 1이다

A	B	C	D	Z_1	Z_2	Z_3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	X

* 문제 설명에 대한 3가지 해석

Truth Table(진리표) 작성하기

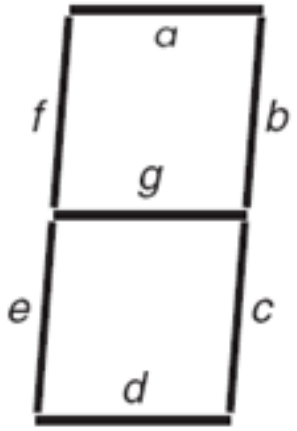
예문 4. 시스템은 십진 숫자에 대한 코드를 입력으로 받고 대부분의 디지털 시계와 숫자 표시에 사용되는 7-세그먼트 디스플레이를 구동하는 신호를 출력으로 제공한다.



***숫자 6, 7, 9: 두 가지 표현 방법이 있음**

1. 어떤 BCD(Binary Coded Decimal) code를 쓸건 지?
2. 스위치는 출력이 1이 되면 불이 들어오는지 0이 되면 들어오는지?
3. 6, 7, 9 처리는 어떻게 할건지?

Digit	W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	X	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	X	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	X	0	1	1
–	1	0	1	0	X	X	X	X	X	X	X
–	1	0	1	1	X	X	X	X	X	X	X
–	1	1	0	0	X	X	X	X	X	X	X
–	1	1	0	1	X	X	X	X	X	X	X
–	1	1	1	0	X	X	X	X	X	X	X
–	1	1	1	1	X	X	X	X	X	X	X



0-9이외의 입력은
안 들어온다고 가정한 상태

1. 어떤 BCD(Binary Coded Decimal) code를 쓸건 지? ➔ 8421
2. 스위치는 출력이 1이 되면 불이 들어오는지 0이 되면 들어오는지?
3. 6, 7, 9 처리는 어떻게 할건지? ➔ don't care로 처리

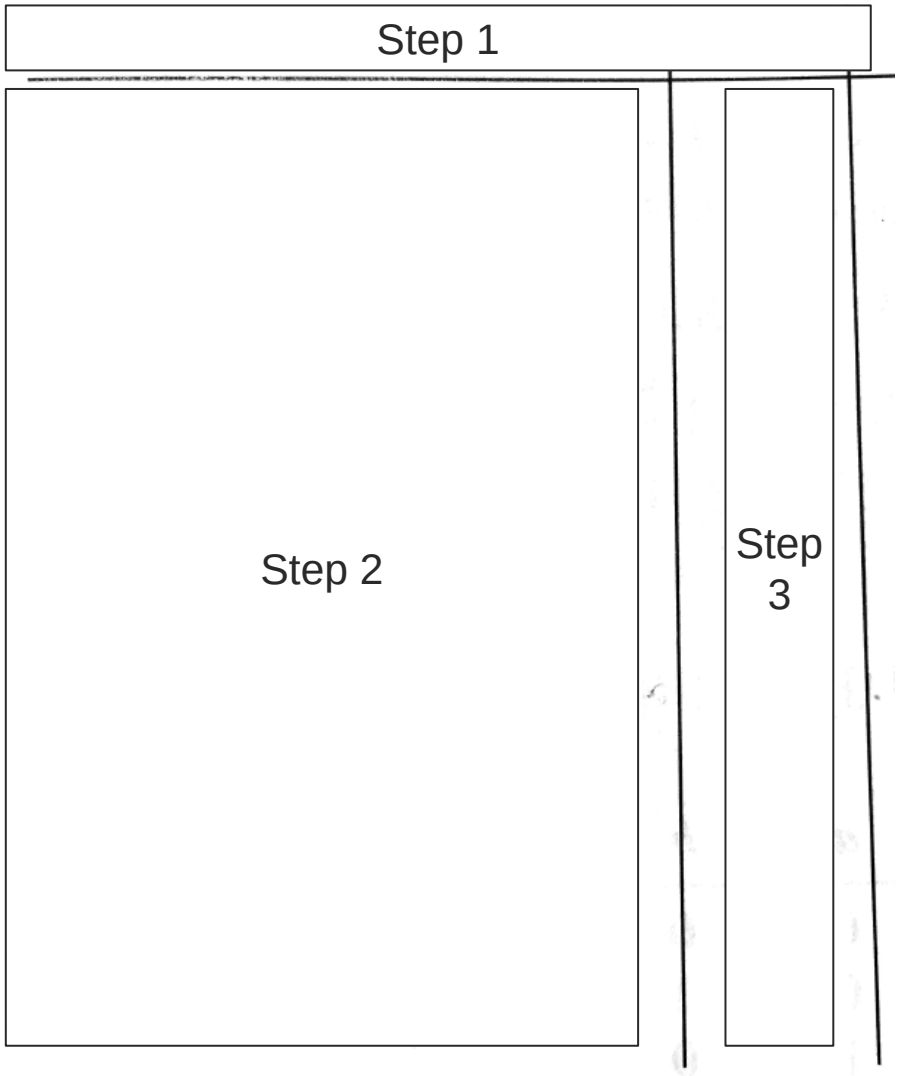
Q1 진리표 그리기

4개의 입력 A, B, C, D와 출력 W를 갖는 시스템이다.

입력은 4 bit unsigned binary number이다.

출력 W는 입력이 십진수로 2 혹은 3의 배수이지만, 동시에 2와 3의 배수는 아닐 때 1이 된다.

위의 시스템을 구현하기 위한 진리표를 작성하세요.


Step 1
Step 2
Step 3

Chapter 2 Combinational Systems

2.1 조합회로 시스템 설계 과정

2.2 스위칭 대수 (Switching Algebra)

2.3 AND, OR, NOT 게이트에 의한 함수 구현

2.4 보수 (The Complement)

2.5 진리표로부터 대수적 표현

2.6 NAND, NOR, XOR 게이트

2.7 대수식의 간소화

2.8 대수 함수의 조작과 NAND게이트 구현

2.9 일반적인 부울 대수(Boolean Algebra)

Chapter 2에서 여러분은

Specification: 시스템 기능을 진리표로 작성할 수 있다

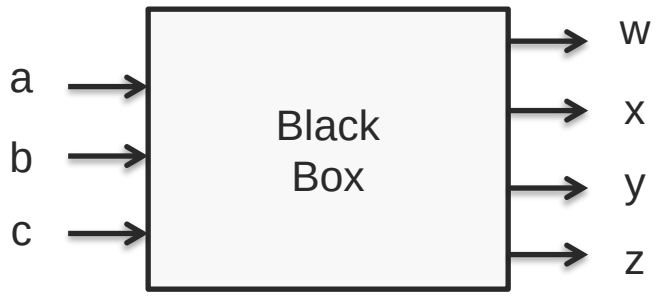
Description: 진리표를 스위칭 대수로 표현할 수 있다

Simplification: 스위칭 대수 특성 이용하여 최적화할 수 있다

Implementation: 게이트를 이용하여 설계할 수 있다

스위칭 대수 vs. 진리표

a	b	c	w	x	y	z
0	0	0	0	1	1	1
0	0	1	1	1	0	1
0	1	0	X	X	X	X
0	1	1	1	1	1	1
1	0	0	0	0	1	0
1	0	1	X	X	X	X
1	1	0	0	1	1	0
1	1	1	1	1	1	0



vs.

스위칭 대수의 필요성

AND, OR , NOT

- 회로의 각 게이트가 대수식으로 표현된다.
- 게이트로 이루어진 회로에 대한 입력과 출력의 관계를 함수로 나타낸다.
- 대수 식을 간소화하여 회로를 간단히 한다.
  - 대수학적 방법
  - 비대수학적(그래픽적) 방법
- 대수 식을 조작하여 조건에 맞는 회로를 구현할 수 있다.

스위칭 대수 정의

- 스위칭 

대수: 모든 변수는 0 또는 1

- 3가지 연산자의 정의
  - **OR (+)**
    a + b = 1 iff a = 1 or b = 1
  - **AND (•)**
    a • b = 1 iff a = 1 and b = 1
  - **NOT (′)**
    a′ is 1 iff a = 0
- 진리표로 나타내면..

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	ab
0	0	0
0	1	0
1	0	0
1	1	1

a	a'
0	1
1	0

스위칭 대수 정의

- 기본적인 불 대수식은 AND, OR, NOT을 이용하여 표현
- AND식은 곱셈의 형식으로 표현하고, OR 식은 덧셈의 형식으로 표현
- NOT식은 $\bar{A}$ 또는 A' 로 표현
- 완전한 논리식은 입력 항목들의 상태에 따른 출력을 결정하는 식

A=0 and B=1 **일 때 출력을 1로 만들려는 경우 출력 논리식**	$F = \overline{A}B$
A=0 or B=1 **일 때 출력을 1로 만들려는 경우 출력 논리식**	$F = \overline{A} + B$
(A=0 and B=1) or (A=1 and B=0) **일 때 출력을 1로 만들려는 경우 출력 논리식**	$F = \overline{A}B + A\overline{B}$

스위칭 대수 표현

- 1입력 논리식, 2입력 논리식, 3입력 논리식

1입력 논리식	2입력 논리식	3입력 논리식

입력	출력
A	F
0	$F = \overline{A}$
1	$F = A$

입력		출력
A	B	F
0	0	$F = \overline{A}\,\overline{B}$
0	1	$F = \overline{A}B$
1	0	$F = A\overline{B}$
1	1	$F = AB$

입력			출력
A	B	C	F
0	0	0	$F = \overline{A}\,\overline{B}\,\overline{C}$
0	0	1	$F = \overline{A}\,\overline{B}C$
0	1	0	$F = \overline{A}B\overline{C}$
0	1	1	$F = \overline{A}BC$
1	0	0	$F = A\overline{B}\,\overline{C}$
1	0	1	$F = A\overline{B}C$
1	1	0	$F = AB\overline{C}$
1	1	1	$F = ABC$

진리표 → 스위칭 대수

입력		출력
A	B	F
0	0	$F = \overline{A}\,\overline{B}$
0	1	$F = \overline{A}B$
1	0	$F = A\overline{B}$
1	1	$F = AB$

입력		출력
A	B	F
0	0	0
0	1	1
1	0	0
1	1	1

입력			출력
A	B	C	F
0	0	0	$F = \overline{A}\,\overline{B}\,\overline{C}$
0	0	1	$F = \overline{A}\,\overline{B}C$
0	1	0	$F = \overline{A}B\overline{C}$
0	1	1	$F = \overline{A}BC$
1	0	0	$F = A\overline{B}\,\overline{C}$
1	0	1	$F = A\overline{B}C$
1	1	0	$F = AB\overline{C}$
1	1	1	$F = ABC$

입력			출력
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

스위칭 대수 속성

- 계산 순서(order of precedence):
  - 괄호는 항상 먼저
  - NOT
  - AND
  - OR

  Ex) ab′+c′d=(a(b′))+((c′)d)

- 속성들은 항상 쌍대(dual)로 나타난다.

  Ex) P1a. a+b=b+a ⟺ P1b. ab=ba

쌍대성(duality) : 불 대수 공리나 기본 법칙에서 좌우 한 쌍에서 0과 1을 서로 바꾸고 동시에 '•' 과 '+' 를 서로 바꾸면 다른 한 쪽이 얻어지는 성질

스위칭 대수 속성

- **교환(c*ommutative)* :**
  P1a. a+b=b+a　　P1b. ab=ba

- **결합(*associative)* :**
  P2a. a+(b+c)=(a+b)+c　　P2b. a(bc)=(ab)c

  - 각 속성에서 한 변수는 대수 식도 포함한다.
    즉 xy′z + w′ = w′ + xy′z ← 교환법칙

쌍대성(duality) : 불 대수 공리나 기본 법칙에서 좌우 한 쌍에서 0과 1을 서로 바꾸고 동시에 '•' 과 '+' 를 서로 바꾸면 다른 한 쪽이 얻어지는 성질

스위칭 대수 속성

P3a. $a + 0 = a$
P4a. $a + 1 = 1$
P5a. $a + a' = 1$

P3b. $a \cdot 1 = a$
P4b. $a \cdot 0 = 0$
P5b. $a \cdot a' = 0$

P6a. $a + a = a$
P7. $(a')' = a$

P6b. $a \cdot a = a$

분배 법칙(*distributive* law):

P8a. $a(b+c)=ab+ac$

P9a. $ab + ab' = a$

P10a. $a + a'b = a+b$

← dual →

P8b. $a+bc=(a+b)(a+c)$

P9b. $(a + b)(a + b') = a$

P10b. $a(a' + b) = ab$

쌍대성(duality) : 불 대수 공리나 기본 법칙에서 좌우 한 쌍에서 0과 1을 서로 바꾸고 동시에 '•' 과 '+' 를 서로 바꾸면 다른 한 쪽이 얻어지는 성질

스위칭 대수 속성

- *DeMorgan's* theorem (드모르간의 정리)

P11a. $(a + b)' = a'\ b'$ P11b. $(ab)' = a' + b'$

a	b	$a + b$	$(a + b)'$	a'	b'	$a'b'$	ab	$(ab)'$	$a' + b'$
0	0	0	1	1	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	0
			11a			11a		11b	11b

P11aa. $(a + b + c\ldots)' = a'\,b'\,c'\ldots$

P11bb. $(abc\ldots)' = a' + b' + c' \ldots$

스위칭 대수 속성

- *DeMorgan's* theorem (드모르간의 정리) 예

  $f = wx'y + xy' + wxz$ 일때

  $f' = (wx'y + xy' + wxz)'$

  $= (wx'y)' (xy')' (wxz)'$

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

  법칙:

  1. 각 변수를 보수화한다.
  2. 0은 1로, 1은 0 으로 대체한다.
  3. AND는 OR, OR는AND로 대체한다.

     (연산의 순서를 유지하는 것을 확인한다. 때로는 부가적인 괄호가 필요)

Q2 이해 안되는 스위칭 대수의 속성을 확인해보려면?

P10a. a + a'b = a+b

진리표 그려보기

왼쪽: 입력 a, b 출력 f

오른쪽: 입력 a, b 출력 g

f와 g가 동일한 진리표 갖는지?

A B C	좌측식		우측식		
	$B \cdot C$	$A+B \cdot C$	$A+B$	$A+C$	$(A+B)(A+C)$
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

동일한 결과

스위칭 대수 용어 정리 (1)

- *문자(literal):*
  - 변수나 변수의 보수, 식의 복잡성을 결정하는데 측정기준
  - ab' + bc'd + a'd + e' : 8개의 문자
- *곱 항(product term)*
  - AND 연산자로 연결
  - 위의 예에서 4개의 곱 항 ab', bc'd, a'd, e' 가 있다.
  - 단일 문자도 곱 항

w'xyz' + wx'y'z' + wx'yz + wxyz	(product terms)
x + w'y + wxy'z	(product terms)
x' + y + z	(product terms)
wy'	(product term)
z	(product term)

스위칭 대수 용어 정리 (2)

- 표준 곱 항(standard product term):
  - 최소항(**minterm**)이라고도 한다.
  - 주어진 문제에서의 변수 모두를 포함하는 곱 항이다.(보수라도 OK)
  - 4변수 (w, x, y, z) 함수의 경우
    - w'xyz , wxyz: Standard product term
    - wy'z : just product term

스위칭 대수 용어 정리 (3)

- 곱의 합(sum of products, SOP) 식
  - Product term이 OR 연산자에 의해 연결
  - Ex) SOP
    - w'xyz' + wx'y'z' + wx'yz + wxyz (4 product terms)
    - x + w'y + wxy'z (3 product terms)
    - x' + y + z (3 product terms)
    - wy' (1 product term)
    - z (1 product term)

스위칭 대수 용어 정리 (4)

- 정규화 합(canonical sum)
  - 혹은 표준 곱 항의 합(sum of standard product terms)
  - SOP인데 모든 term이 standard product term
  - 4변수 (w, x, y, z) 함수의 경우: **어느 것이 canonical sum?**
    - w'xyz' + wx'y'z' + wx'yz + wxyz (4 product terms)
    - x + w'y + wxy'z (3 product terms)
    - x' + y + z (3 product terms)
    - wy' (1 product term)
    - z (1 product term)

스위칭 대수 용어 정리 (5)

- 최소 곱의 합(minimum SOP) 식
  - 주어진 함수에 대한 **가장 적은 수의 product term**을 갖는 SOP 식
  - 가장 적은 수의 항을 갖는 식들이 한 개 이상 있으면,
    - 최소는 **가장 적은 수의 문자(literal)를 갖는** 식들로 정의된다.
  - 주어진 문제에 대해 한 개 이상의 최소 곱의 합 식이 있을 수 있다.

(1) $x'yz' + x'yz + xy'z' + xy'z + xyz$	5 terms, 15 literals
(2) $x'y + xy' + xyz$	3 terms, 7 literals
(3) $x'y + xy' + xz$	3 terms, 6 literals
(4) $x'y + xy' + yz$	3 terms, 6 literals

스위칭 대수 용어 정리 (6)

- 합의 곱 식(product of sum, **POS**)
  - AND 연산자에 의해 한 개 이상의 sum term이 연결된 것이다.
  - 예) 4변수 (w, x, y, z) 함수의 경우
    - (w+x)(w+y) sum term
    - w(z+y) sum term
    - w sum term
    - w+x sum term
    - (w+x′+y′+z′)(w′+x+y+z′) sum term

- 정규화 곱(canonical product)
  - 혹은 표준 합 항의 곱(product of standard sum terms)
  - 모든 항이 표준 합 항(canonical SOP)으로 이루어진 합의 곱 식이다.

스위칭 대수 용어 정리 (7)

SOP: $x'y + xy' + xyz$

POS: $(x + y')(x' + y)(x + z')$

? $x' + y + z$ or xyz'

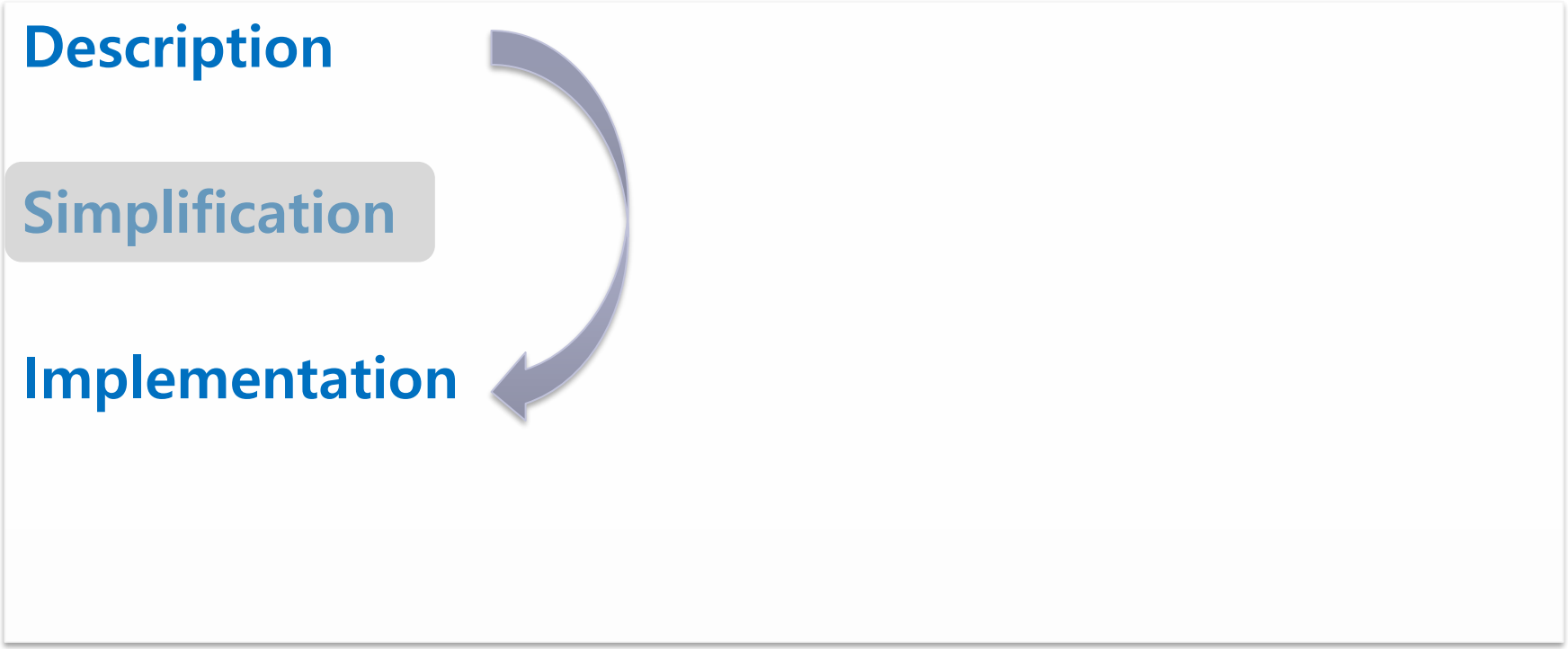
? $x(w' + yz)$ or $z' + wx'y + v(xz + w')$

Chapter 2 Combinational Systems

2.1 조합회로 시스템 설계 과정

2.2 스위칭 대수 (Switching Algebra)

2.3 AND, OR, NOT 게이트에 의한 함수 구현



Chapter 2에서 여러분은

Specification: 시스템 기능을 진리표로 작성할 수 있다

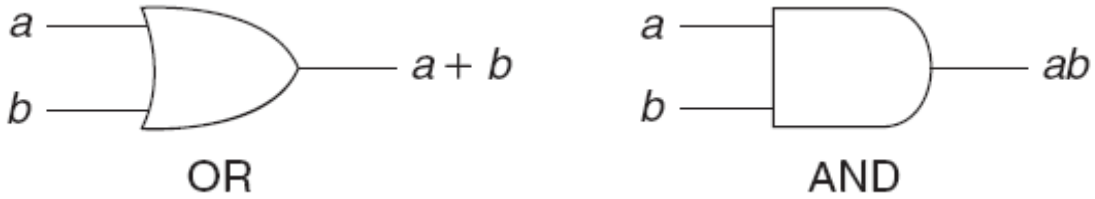
Description: 진리표를 스위칭 대수로 표현할 수 있다

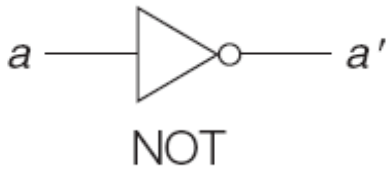
Simplification: 스위칭 대수 특성 이용하여 최적화할 수 있다

Implementation: 게이트를 이용하여 설계할 수 있다

스위칭 대수와 게이트

- 게이트
  - AND, OR, NOT와 같은 하나의 기본 함수를 구현하는 회로 소자
  - 다수 개의 입력 가능
  - 한 개의 출력



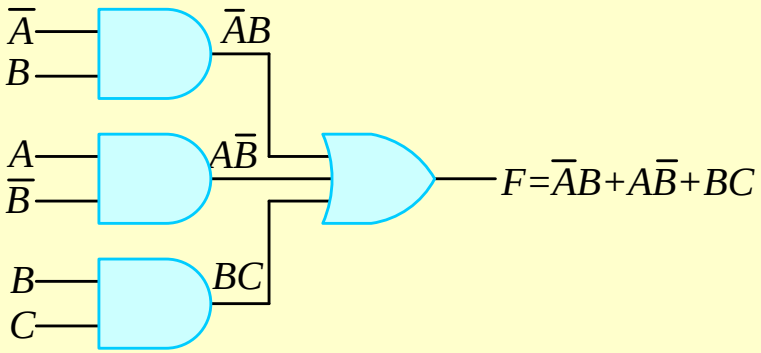


스위칭 대수 ➔ 게이트

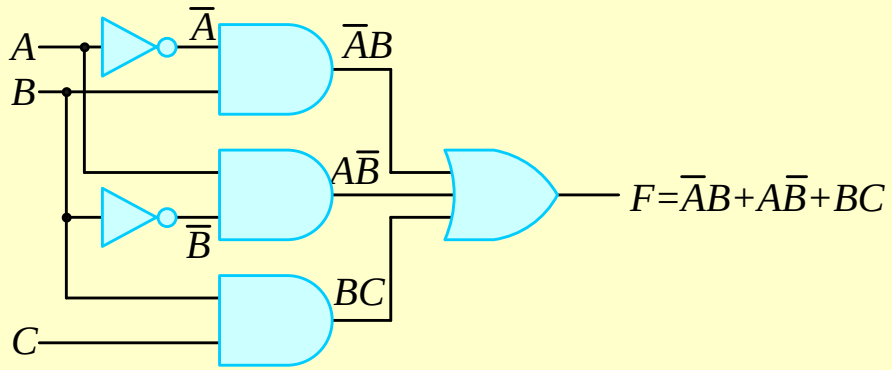
- AND, OR, NOT을 이용하여 논리식으로부터 회로를 구성 (AND-OR로 구성된 회로)

$$\bar{A}B + A\bar{B} + BC$$

보수입력 사용

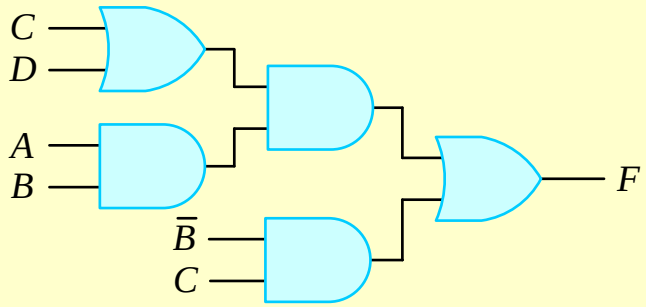
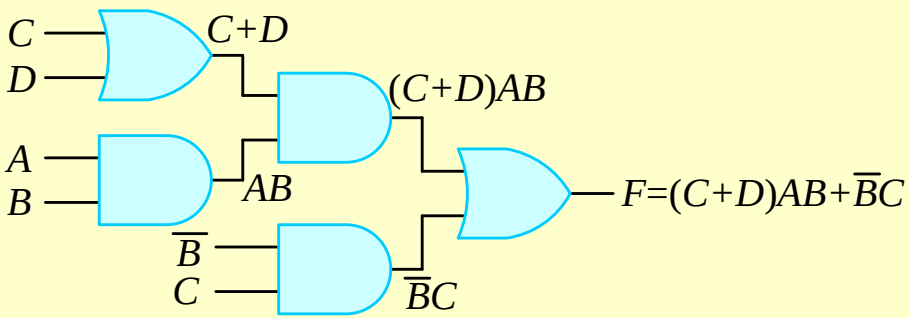


NOT 게이트 사용



게이트 ➔ 스위칭 대수

- 원래의 회로에 게이트를 거칠 때마다 게이트의 출력을 적어주면서 한 단계씩 출력 쪽으로 나아가면 된다.

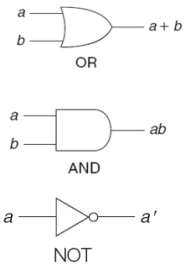
논리회로	논리식 유도 과정
	

AND, OR, NOT gate에 의한 함수 구현(1)

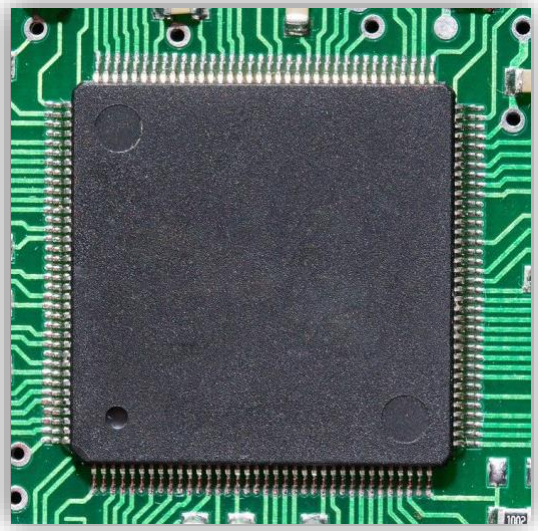
$$f = x'yz' + x'yz + xy'z' + xy'z + xyz$$

Description

- 3가지 연산자의 정의
  - **OR (+)**
    a + b = 1 iff a = 1 or b = 1
  - **AND (•)**
    a • b = 1 iff a = 1 and b = 1
  - **NOT (')**
    a' is 1 iff a = 0



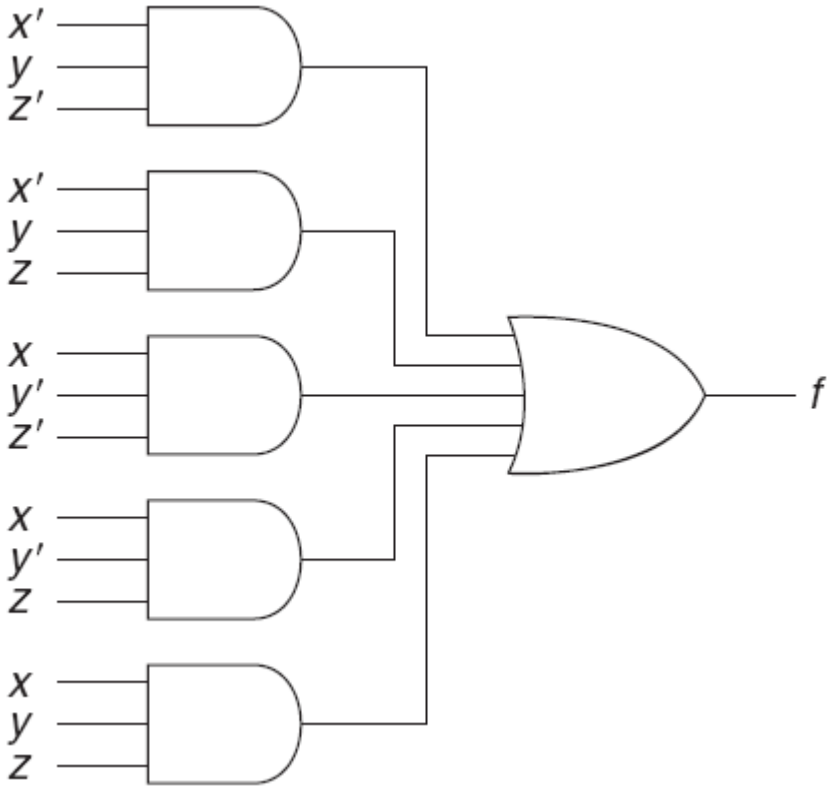
Implementation



AND, OR, NOT gate에 의한 함수 구현(2)

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

Description



Implementation

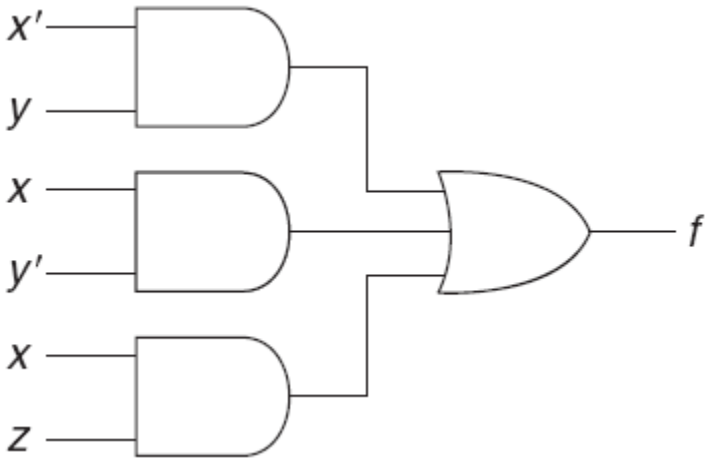
two-level circuit

AND, OR, NOT gate에 의한 함수 구현(3)

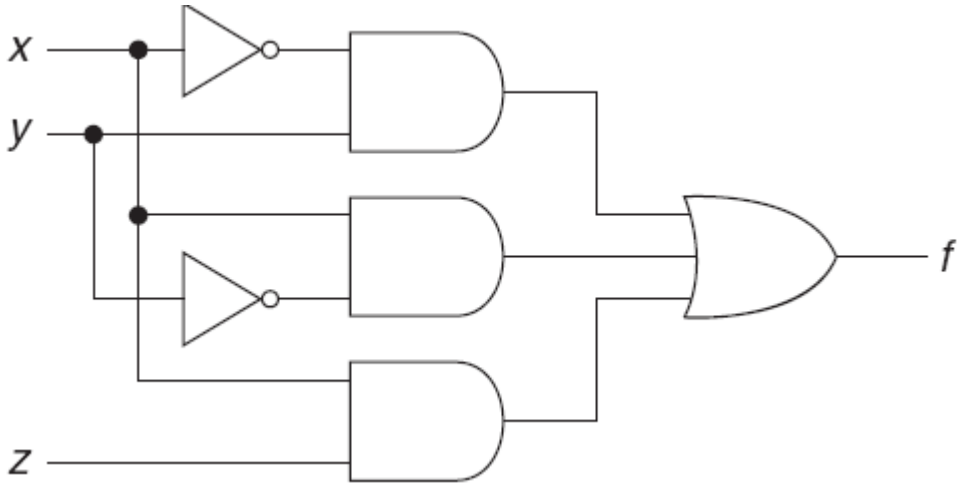
- 최소 곱의 합:

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

$$f = x\acute{y} + xy' + xz$$



Minimum sum of product implementation of *f*.



Circuit with only uncomplemented inputs.

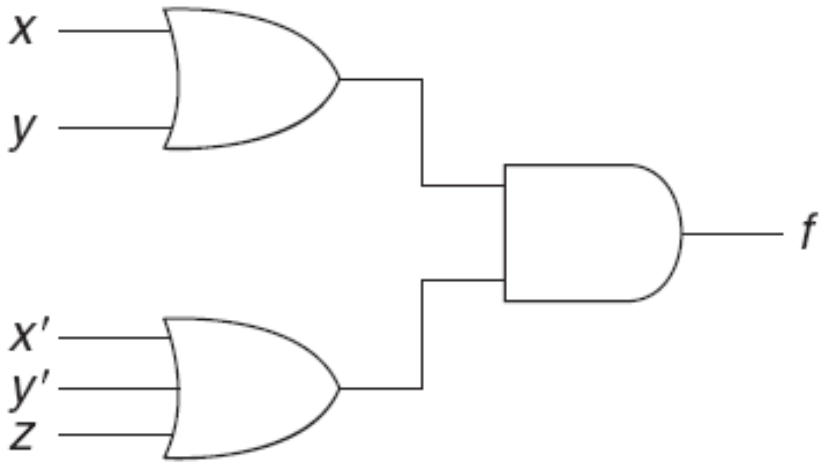
기능 구현 뿐 아니라 최적화까지 한 상태

three-level circuit

AND, OR, NOT gate에 의한 함수 구현(4)

- 합의 곱 식:

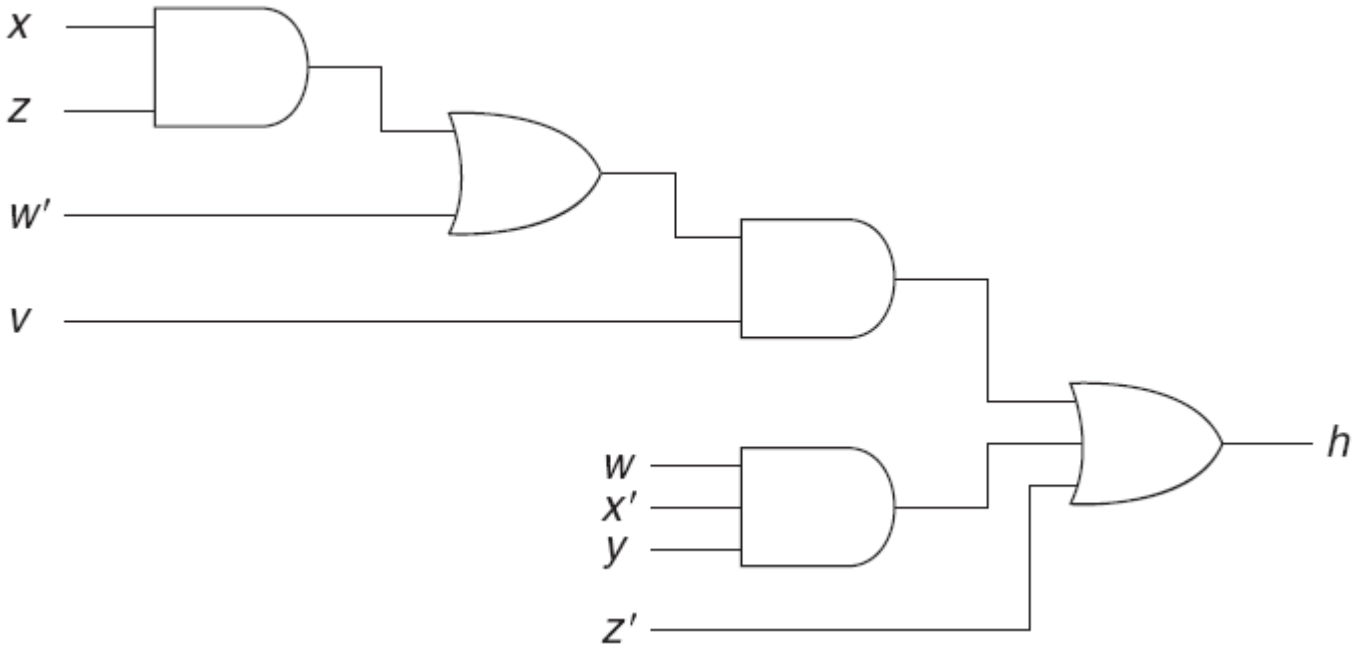
$$f = (x + y)(x' + y' + z)$$



AND, OR, NOT gate에 의한 함수 구현(5)

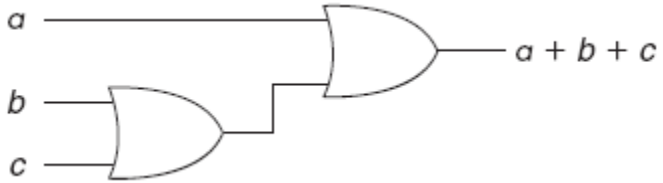
- 멀티레벨 회로:

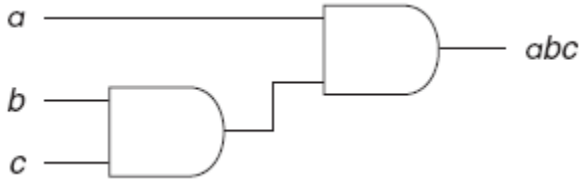
$$h = z' + wx'y + v(xz + w')$$



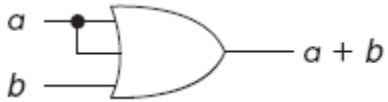
사용 가능한 소자들의 제약이 있을 때는

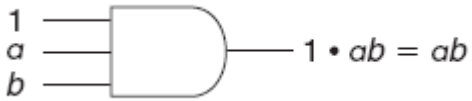
2-input OR/AND밖에 없을 때

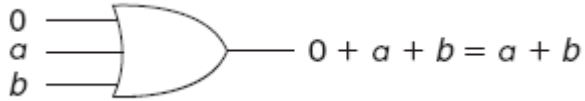


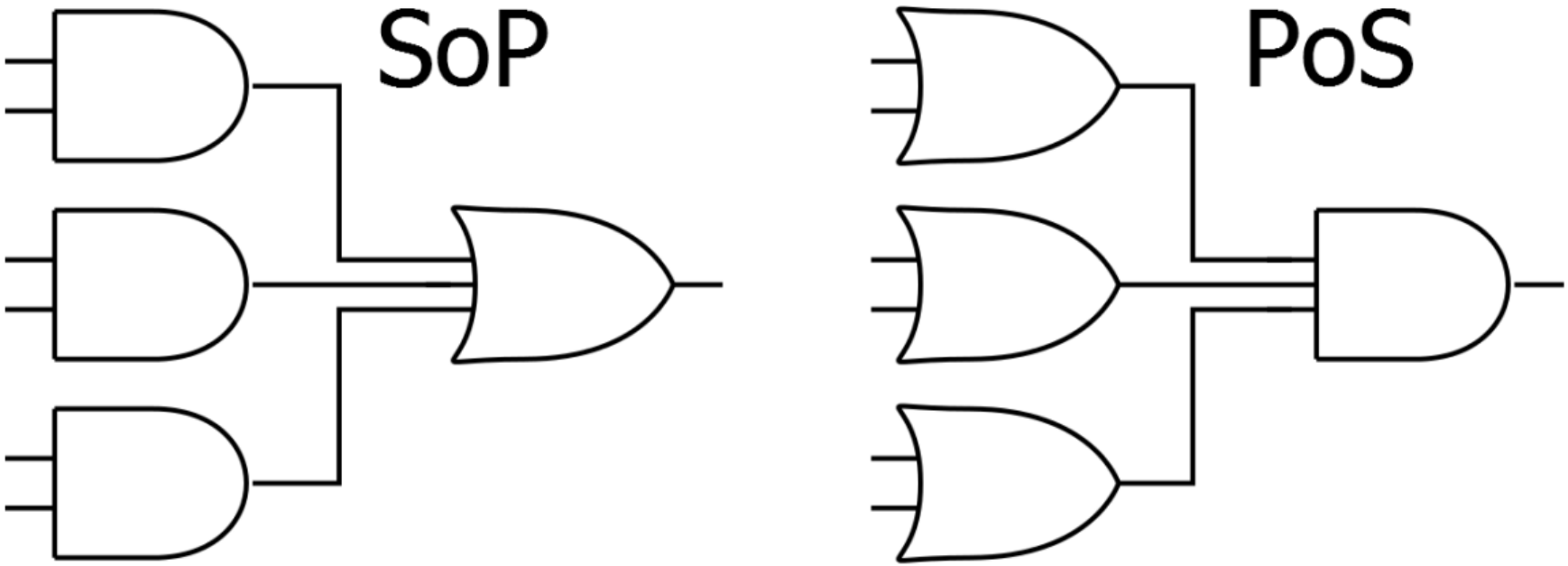


3-input OR/AND밖에 없을 때








SoP
PoS

Chapter 2 Combinational Systems

Chapter 2에서 여러분은

Specification: 시스템 기능을 진리표로 작성할 수 있다

Description: 진리표를 스위칭 대수로 표현할 수 있다

Simplification: 스위칭 대수 특성 이용하여 최적화할 수 있다

Implementation: 게이트를 이용하여 설계할 수 있다

진리표 → 대수적 표현

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

- f is 1 if $a = 0$ **AND** $b = 1$ **OR**
  if $a = 1$ **AND** $b = 0$ **OR**
  if $a = 1$ **AND** $b = 1$

입력		출력
A	B	F
0	0	$F = \overline{A}\,\overline{B}$
0	1	$F = \overline{A}B$
1	0	$F = A\overline{B}$
1	1	$F = AB$

대수적 표현을 minterm으로 나타내기

- Minterm?? = 표준 곱 항(standard product term):
  - 주어진 문제에서의 변수 모두를 포함하는 곱 항이다.(보수라도 OK)
  - 4변수 (w, x, y, z) 함수의 경우
    - w'xyz , wxyz: minterm
    - wy'z : just product term

ABC	Minterm	Number	
0 0 0	$A'B'C'$	0	m_0
0 0 1	$A'B'C$	1	m_1
0 1 0	$A'BC'$	2	m_2
0 1 1	$A'BC$	3	m_3
1 0 0	$AB'C'$	4	m_4
1 0 1	$AB'C$	5	m_5
1 1 0	ABC'	6	m_6
1 1 1	ABC	7	m_7

Minterm으로 표현 후 간소화

Specification

ABC	f	f ′
0 0 0	0	1
0 0 1	1	0
0 1 0	1	0
0 1 1	1	0
1 0 0	1	0
1 0 1	1	0
1 1 0	0	1
1 1 1	0	1

Description

$$f(A, B, C) = m_1 + m_2 + m_3 + m_4 + m_5$$
$$= \sum m(1, 2, 3, 4, 5)$$
$$= A'B'C + A'BC' + A'BC + AB'C' + AB'C$$

Simplification

$f = A'B'C + \underline{A'BC' + A'BC} + \underline{AB'C' + AB'C}$

$= A'B'C + A'B + AB'$ →P9a

$= A'C + A'B + AB'$ →P8a, P10a

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

Simplified SOP *f*

P1a. a+b=b+a
P2a. a+(b+c)=(a+b)+c
P3a. a + 0 = a
P4a. a + 1 = 1
P5a. a + a′ = 1

P6a. a + a = a
P7. (a′)′=a

P8a. a(b+c)=ab+ac
P9a. ab + ab′ = a
P10a. a + a′b = a+b

P1b. ab=ba
P2b. a(bc)=(ab)c
P3b. a · 1 = a
P4b. a · 0 = 0
P5b. a · a′ = 0

P6b. a · a = a

P8b. a+bc=(a+b)(a+c)
P9b. (a + b)(a + b′) = a
P10b. a(a′ + b) = ab

Minterm으로 표현 후 간소화

Specification

ABC	f	f ′
0 0 0	0	1
0 0 1	1	0
0 1 0	1	0
0 1 1	1	0
1 0 0	1	0
1 0 1	1	0
1 1 0	0	1
1 1 1	0	1

Description

$$f'(A, B, C) = \sum m(0, 6, 7)$$
$$= A'B'C' + ABC' + ABC$$

Simplification

$f' = A'B'C' + AB$ →P9a

$f = (A+B+C)(A'+B')$ →P11

Simplified POS *f*

우리는 SOP 만 간소화 할 수 있으므로
f'의 SOP를 간소화해서
드모르간 정리 사용하면
간소화된 POS 얻을 수 있음

P1a.	a+b=b+a	P1b. ab=ba
P2a.	a+(b+c)=(a+b)+c	P2b. a(bc)=(ab)c
P3a.	a + 0 = a	P3b. a · 1 = a
P4a.	a + 1 = 1	P4b. a · 0 = 0
P5a.	a + a′ = 1	P5b. a · a′ = 0
P6a.	a + a = a	P6b. a · a = a
P7.	(a′)′=a	
P8a.	a(b+c)=ab+ac	P8b. a+bc=(a+b)(a+c)
P9a.	ab + ab′ = a	P9b. (a + b)(a + b′) = a
P10a.	a + a′b = a+b	P10b. a(a′ + b) = ab

Chapter 2 Combinational Systems

2.1 조합회로 시스템 설계 과정

2.2 스위칭 대수 (Switching Algebra)

2.3 AND, OR, NOT 게이트에 의한 함수 구현

2.4 보수 (The Complement)

2.5 진리표로부터 대수적 표현

2.6 NAND, NOR, XOR 게이트

2.7 대수식의 간소화

2.8 대수 함수의 조작과 NAND게이트 구현

2.9 일반적인 부울 대수(Boolean Algebra)

간소화에 많이 쓰이는 식들..

♦ **Primary tools:**

P9a. $ab + ab' = a$ **P9b.** $(a + b)(a + b') = a$

P10a. $a + a'b = a+b$ **P10b.** $a(a' + b) = ab$

P6a. $a + a = a$ **P6b.** $a \cdot a = a$

P8a. $a(b+c)=ab+ac$ **P8b.** $a+bc=(a+b)(a+c)$

♦ **Useful two other properties**

P12a. $a + ab = a$ **P12b.** $a(a+b) = a$

P13a. $at_1 + a't_2 + t_1t_2 = at_1 + a't_2$

P13b. $(a + t_1)(a' + t_2)(t_1 + t_2)= (a + t_1)(a' + t_2)$

간소화에 많이 쓰이는 식들..

P13a. $at_1 + a't_2 + t_1t_2 = at_1 + a't_2$

$$
\begin{aligned}
at_1 + a't_2 &= (at_1 + at_1t_2) + (a't_2 + a't_1t_2) \\
&= at_1 + a't_2 + (at_1t_2 + a't_1t_2) \\
&= at_1 + a't_2 + t_1t_2
\end{aligned}
$$

표 2.15 Consensus.

a	t_1	t_2	at_1	$a't_2$	RHS	t_1t_2	LHS
0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	1	0	1
1	1	1	1	0	1	1	1

Q3 진리표, 스위칭 대수, 간소화 연습

- 다음 각 함수에 대하여

$f(x,y,z) = \sum m(1,3,6)$

$g(x,y,z) = \sum m(0,2,4,6)$

1. 진리표를 보여라
2. Minterm SOP 형태로 대수식을 나타내어라.
3. 최소 SOP 식을 보여라.

    f: 2term, 5 literal

    g: 1term, 1 literal

Chapter 2 Combinational Systems

2.1 조합회로 시스템 설계 과정

2.2 스위칭 대수 (Switching Algebra)

2.3 AND, OR, NOT 게이트에 의한 함수 구현

2.4 보수 (The Complement)

2.5 진리표로부터 대수적 표현

2.6 NAND, NOR, XOR 게이트

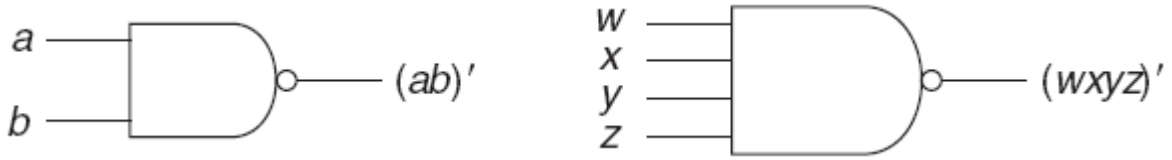
2.7 대수식의 간소화

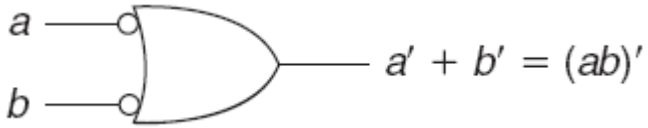
2.8 대수 함수의 조작과 NAND게이트 구현

2.9 일반적인 부울 대수(Boolean Algebra)

NAND 와 NOR

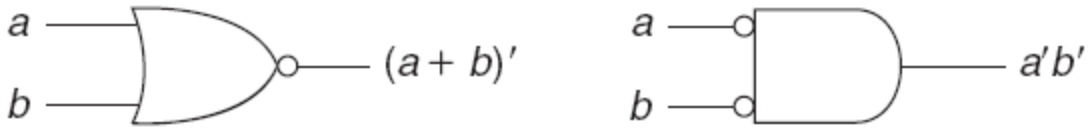
NAND gates.



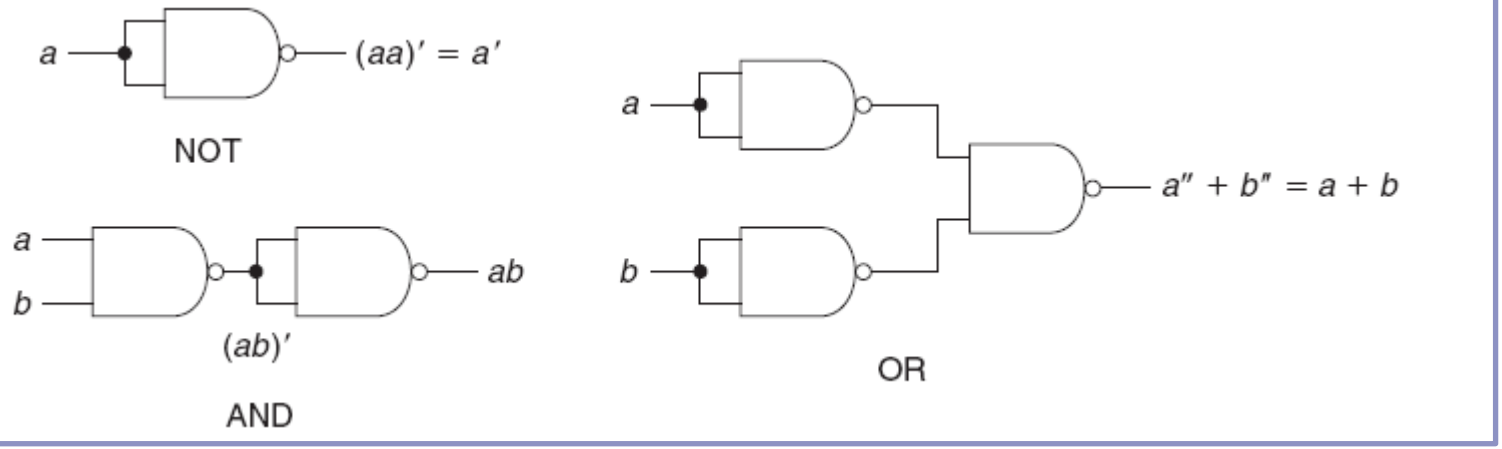


Alternative symbol
for NAND.

Symbols for NOR gate



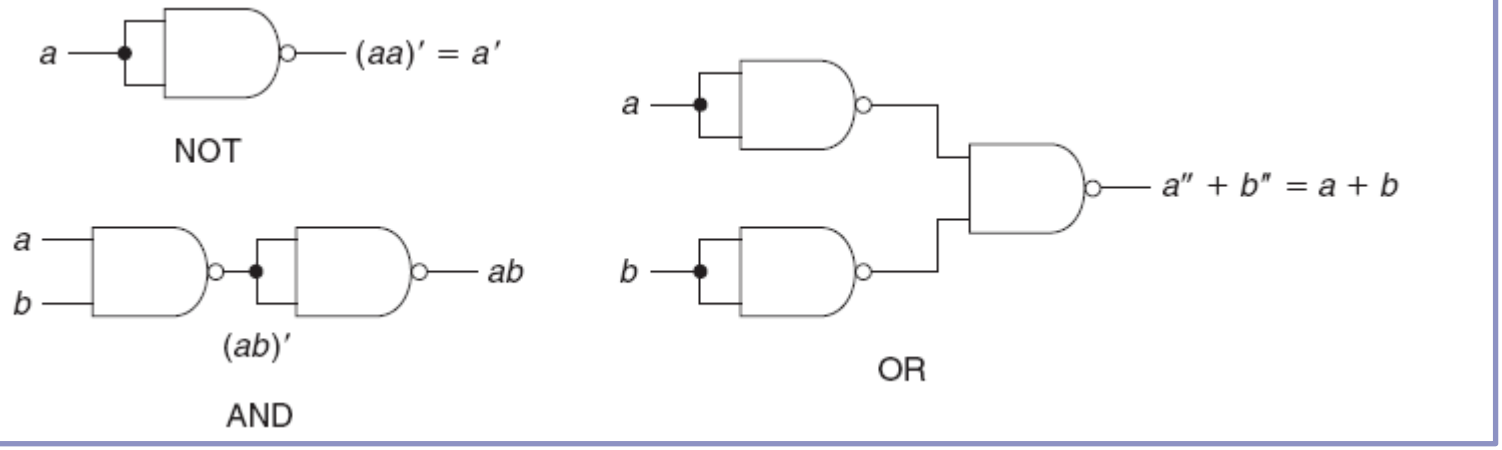
왜 NAND와 NOR?

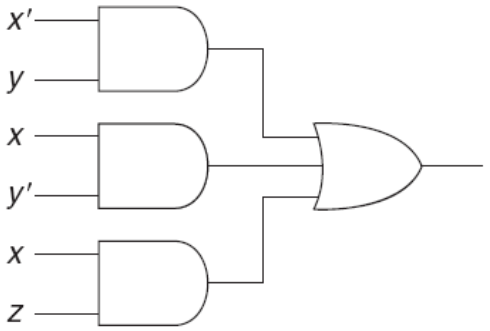


Functionally Complete

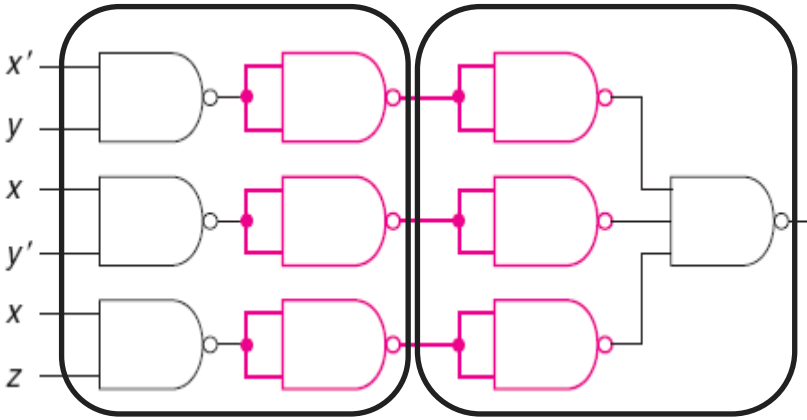
NAND/NOR만 있으면
NOT, AND, OR 모두 표현할 수 있다!

NAND의 완전성의 예

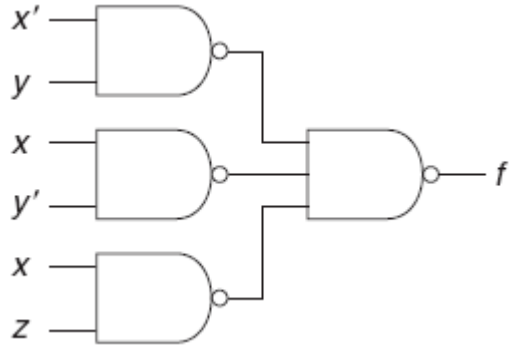




$f = x'y + xy' + xz$

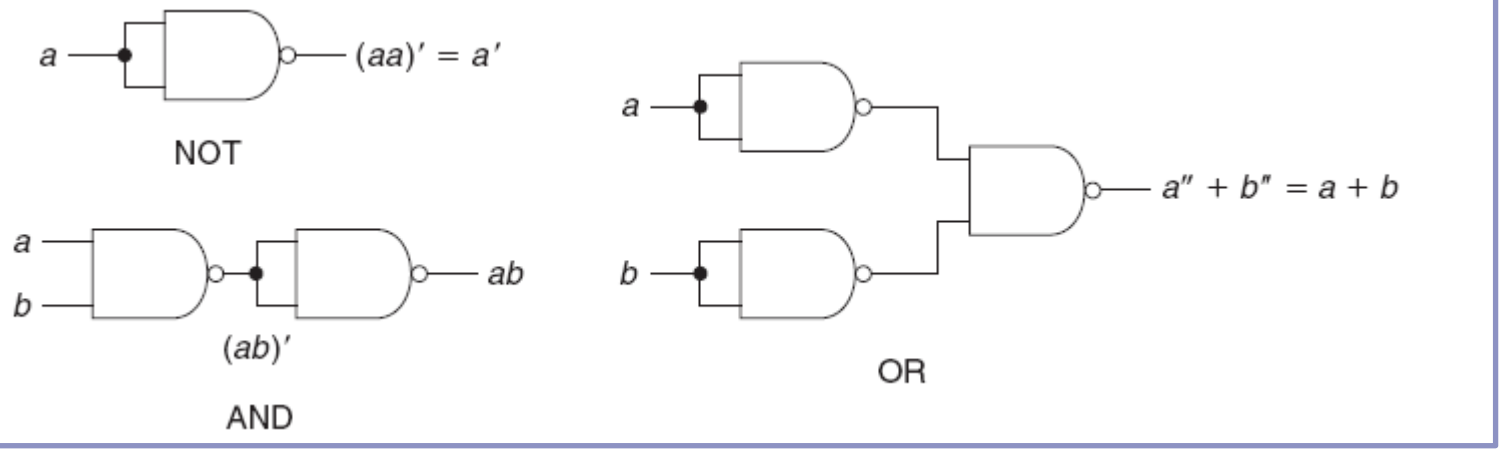


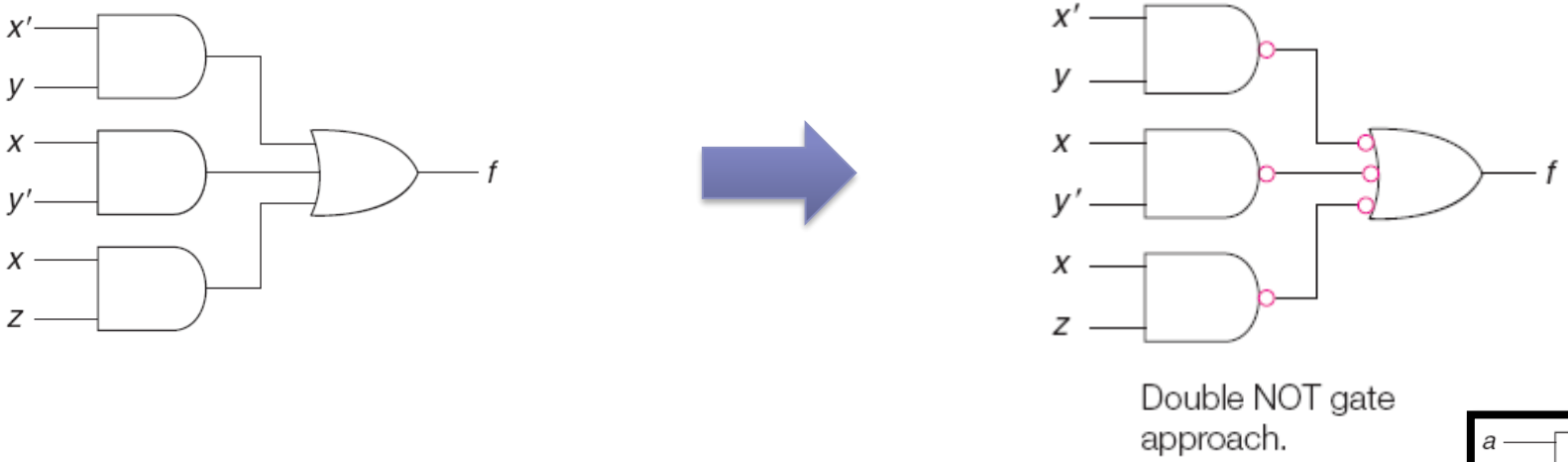
NAND gate implementation.



Better NAND gate implementation.

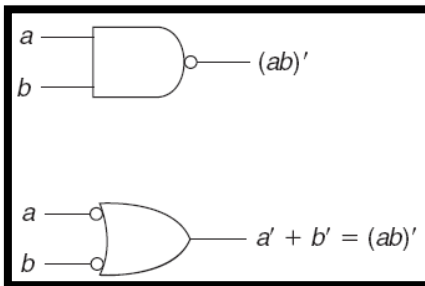
NAND의 완전성의 예





Double NOT gate approach.

이러한 접근 방법을 일반적으로 적용하면
아주 쉽게 변환 가능



NAND로의 변환 방법

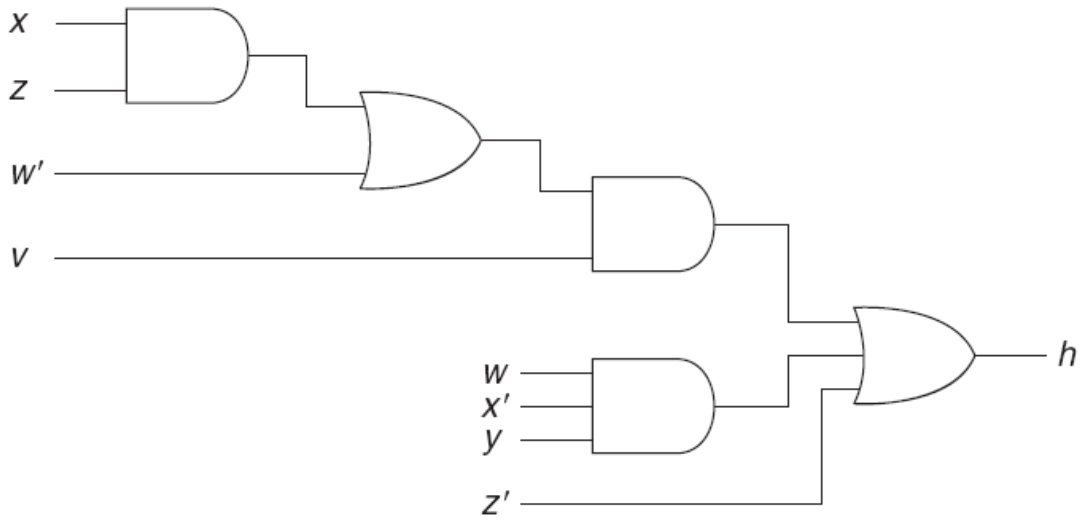
AND 와 OR 로 구성된 회로가 다음과 같은 경우에는 NAND로 변환하는 과정이 간단히 된다.

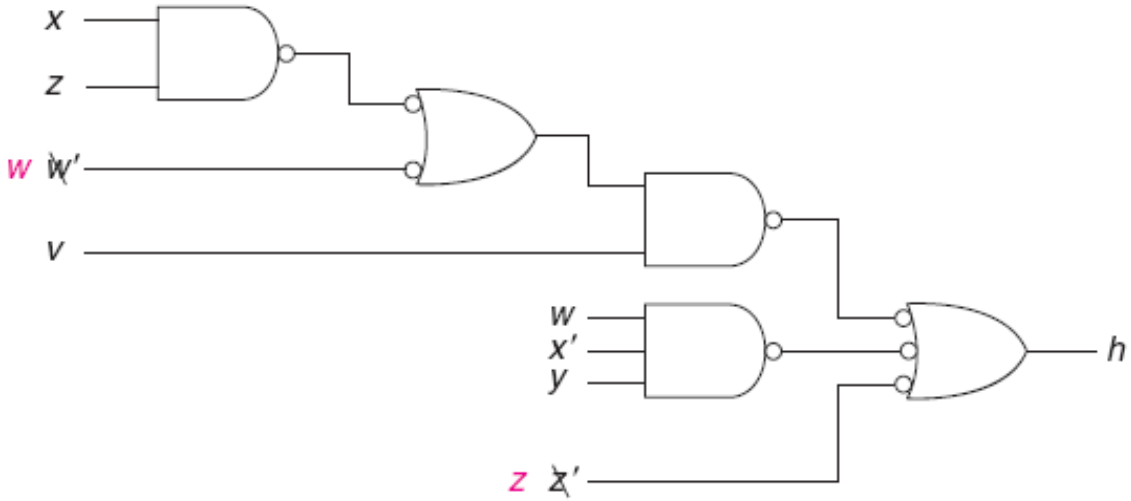
- 회로의 **출력이 OR 게이트에서** 나온다
- 모든 **OR 게이트 입력은 AND 출력**이거나 시스템 입력이다.
- 모든 **AND 게이트 입력은 OR 출력**이거나 시스템 입력이다.

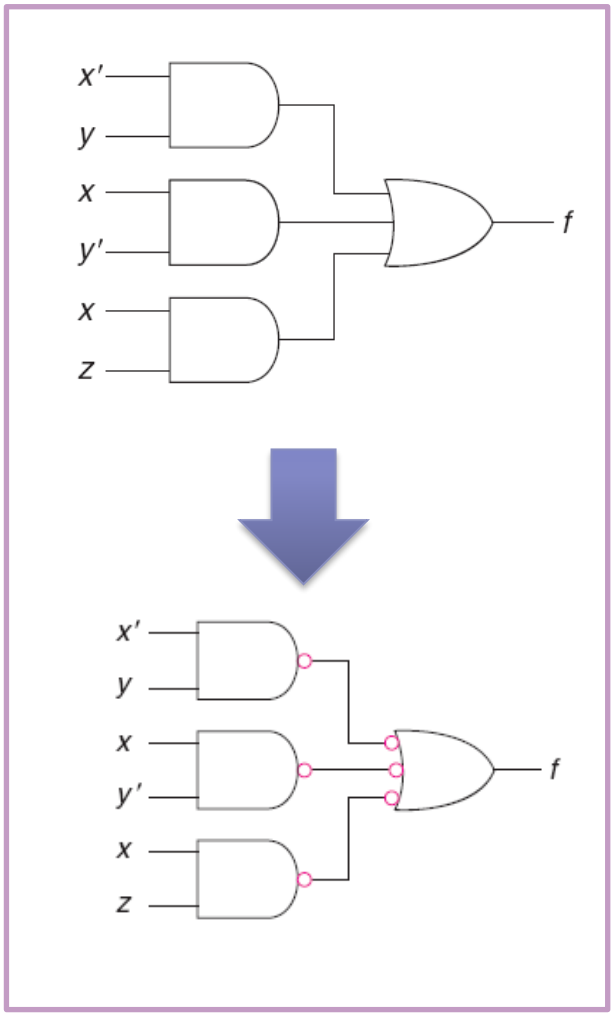
변환과정

1) 모든 게이트는 NAND 게이트로 대체
2) OR 게이트로 직접 들어오는 입력들은 보수화 한다.

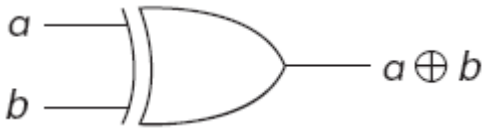
NAND로의 변환 방법



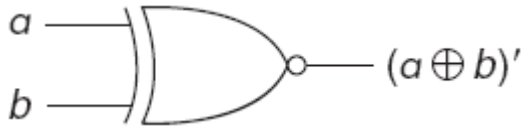




XOR (Exclusive-OR)



Exclusive-OR



Exclusive-NOR

X	Y	O
0	0	0
0	1	1
1	0	1
1	1	0

XOR

X	Y	O
0	0	1
0	1	0
1	0	0
1	1	1

XNOR

입력 값 **비교**에 아주 용이!

우리가 간소화를 잘 할 수 있을까?

- 어디서부터 어떻게 시작할지 모르겠는데..

  ➔ 시행착오와 경험

- 어떤 식을 어떤 순서로 사용해야 하는 거지?

  ➔ 알고리즘 같은 것 없음.. 이것 저것 시도해 보기

- 간소화가 됐는지 안됐는지, 더 해야 하는지 그만둬도 되는지 어떻게 할 수 있지?

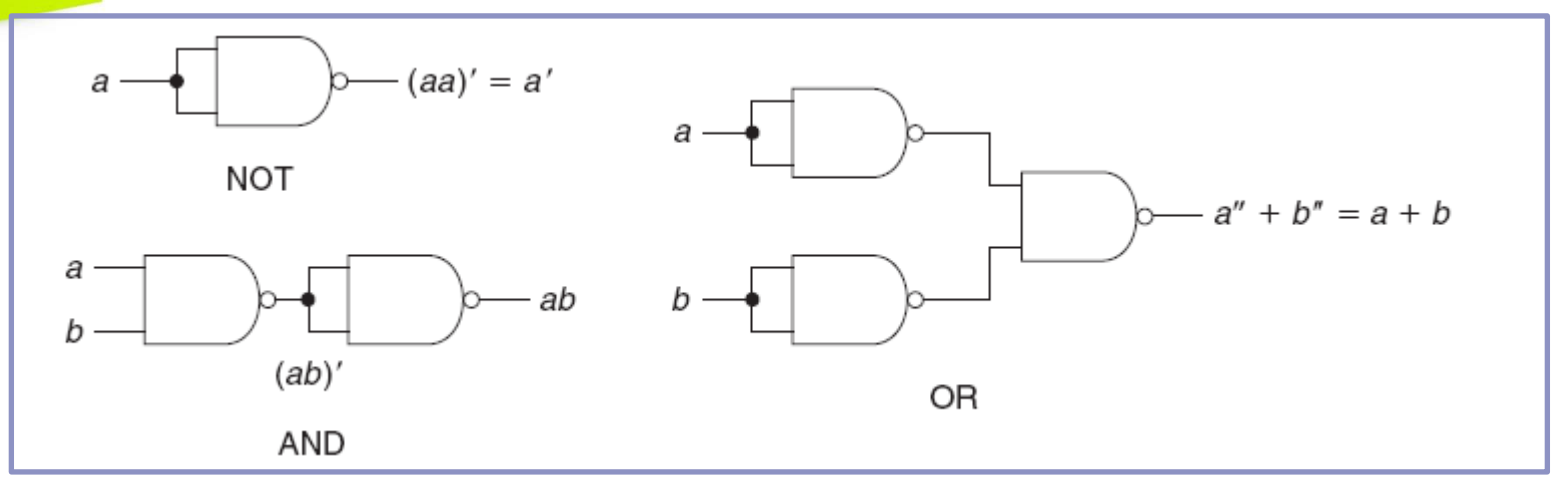
  ➔ 사실은 잘 모른다.

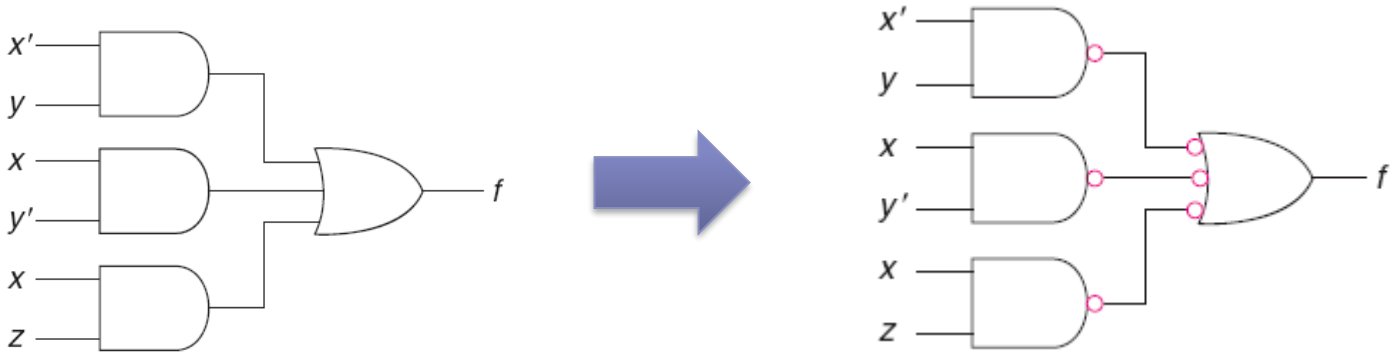
진리표 → 대수적 표현

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

- f is 1 if $a = 0$ **AND** $b = 1$ **OR**
  if $a = 1$ **AND** $b = 0$ **OR**
  if $a = 1$ **AND** $b = 1$
- f is 1 if $a' = 1$ **AND** $b = 1$ **OR**
  if $a = 1$ **AND** $b' = 1$ **OR**
  if $a = 1$ **AND** $b = 1$
- f is 1 if $a'b = 1$ **OR** if $ab' = 1$ **OR** if $ab = 1$
- $f = a'b + ab' + ab$

NAND/NOR





SOP를 NAND only 로 변환하는 과정

1)모든 게이트는 NAND 게이트로 대체

2)OR 게이트로 직접 들어오는 입력들은 보수화 한다.