Ch.3 Gate-Level Minimization

MSOP ① minimal number of product terms
② minimal number of literals.

Karnaugh maps (K-maps)

- minimal sum of products(MSOP) form 만들기 위한 graphical technigue
  - minimal two-level implementation

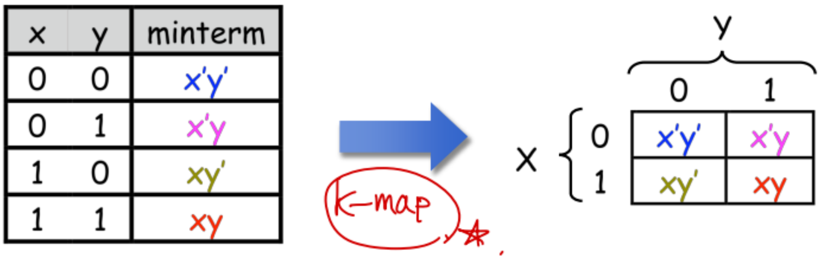
- K-maps are an alternative to algebra for simplifying expressions
  - don’t care conditions 조절하기 수월함
  - are only good for manual simplification of small expressions.

yz / x (x,y,z)

yz / wx (w,x,y,z)

- Remember
  - minterm 매기는 순서 기억할 것 (0, 1, 3, 2) , 세로줄 먼저
  - k-map의 모든 side는 wrap around 할 수 있다는 것.
  - 사각형 → 크기는 가장 크게, 개수는 가장 적게
  - solution은 여러 개일 수 있음 ⇒ MSP 제작 시 필수항 먼저 찾고 시작.

1. re-arranging the truth table



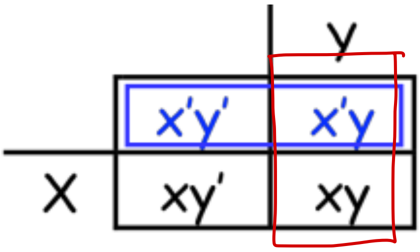
x	y	minterm
0	0	x'y'
0	1	x'y
1	0	xy'
1	1	xy

K-map ☆.

x \ y	0	1
0	x'y'	x'y
1	xy'	xy

2. 근처에 있는 값끼리 묶는다고 생각, 즉 ‘OR’

ex) x’y’ + x’y = x’(y’+y) = x’



→ □ 하나 = 하나의 "OR"

ex) x’y’ + x’y + xy = (x’y’ + x’y) + (x’y + xy) = x’ + y //

adjacent squares

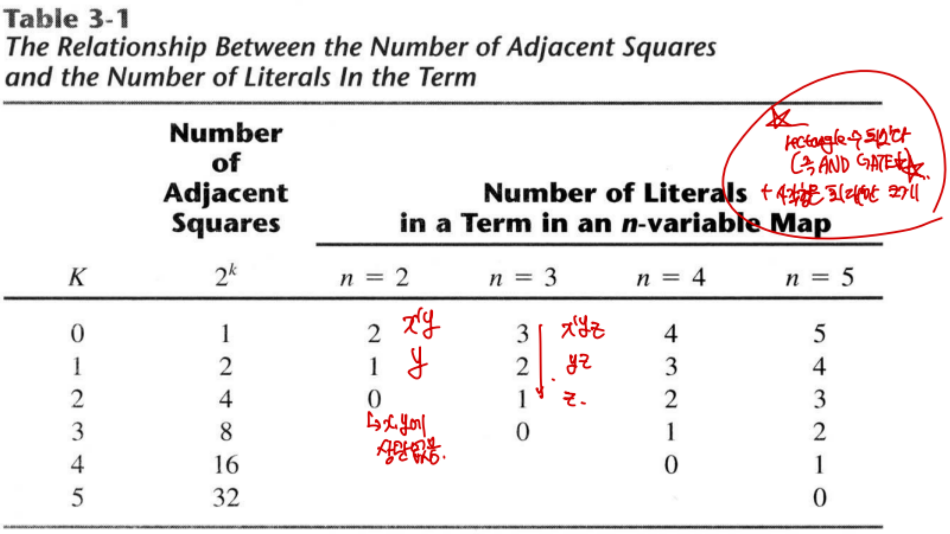
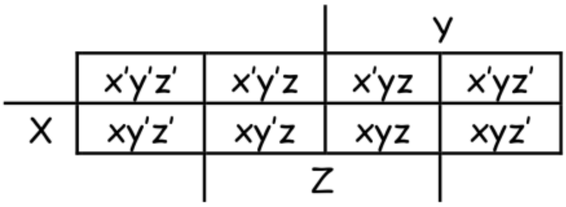


Table 3-1
The Relationship Between the Number of Adjacent Squares and the Number of Literals In the Term

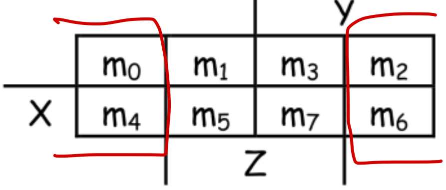
Number of Adjacent Squares		Number of Literals in a Term in an *n*-variable Map			
K	2^k	$n = 2$	$n = 3$	$n = 4$	$n = 5$
0	1	2 $x'y$	3 $x'yz$	4	5
1	2	1 y	2 yz	3	4
2	4	0 ↳ x, y에 상관없음.	1 z.	2	3
3	8		0	1	2
4	16			0	1
5	32				0

★ rectangle 수 되었다 (즉 AND GATE 수)★. + 사각형은 최대한 크게

a three-variable state



가능!



- 인접한 건 m0-m2, m4-m6도 가능

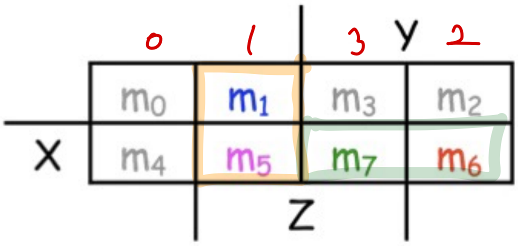
⇒ 최대한 간단하게 만들기 위해 사각형 하나에 많이 묶어줄 것!

- ex) f(x,y,z) = xy + y'z + xz

x	y	z	f(x,y,z)	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	0	3
1	0	0	0	4
1	0	1	1	5
1	1	0	1	6
1	1	1	1	7

↳① sum of minterm으로 바꿔주기

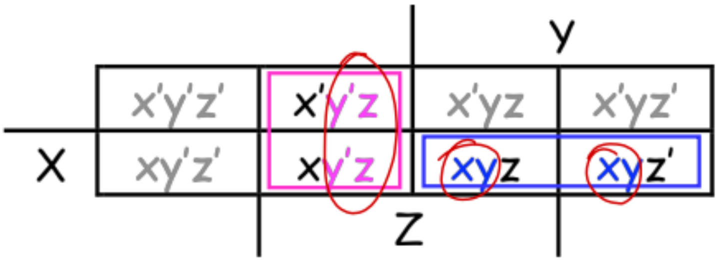
$$f(x,y,z) = m_1 + m_5 + m_6 + m_7$$



$$f(x,y,z)$$
$$= x'y'z + xy'z + xyz' + xyz$$
$$= m_1 + m_5 + m_6 + m_7$$

$= y'z + xy$

- groping the minterms together → 사각형 개수 최소화 하기
  - ① 사각형 하나 = and gate 하나
  - ② input 수 작게 할 것 = 사각형 크기 크게 할 것 (1, 2, 4, 8개씩 묶기)
  - ③ 중복되는 값이 많을수록 사각형 크기가 커질 것임



⇒ f(x, y, z) = y'z + xy

④ 해당하는 min term 네 개가 다 묶여야 하나로 표현 가능함.

⇒ 하나의 사각형은 하나의 product.

Example 3-4

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

- Express it in sum of products
- Find the minimal SOP

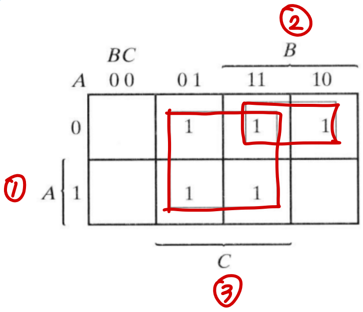
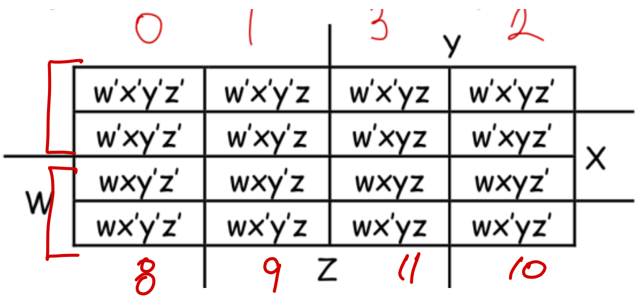


FIGURE 3-7
Map for Example 3-4; $A'C + A'B + AB'C + BC = C + A'B$

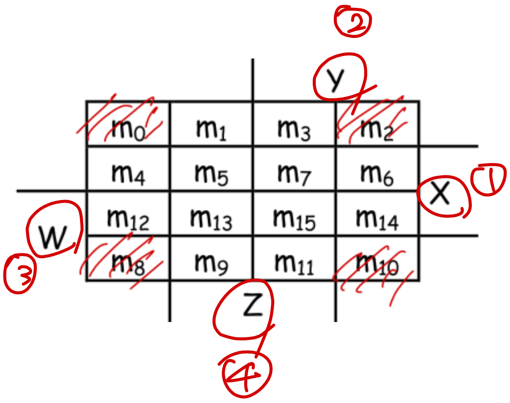
Four-variable K-maps

$f(\underbrace{w, x}_{세로}, \underbrace{y, z}_{가로})$

- **wrap around all four sides!!**



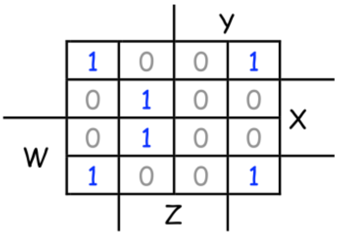
w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'
w'xy'z'	w'xy'z	w'xyz	w'xyz'
wxy'z'	wxy'z	wxyz	wxyz'
wx'y'z'	wx'y'z	wx'yz	wx'yz'



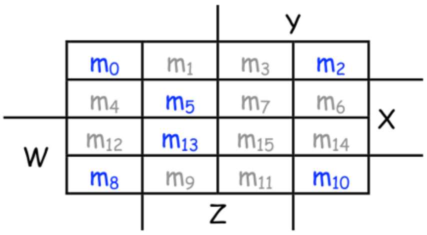
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

- ex) f = m0 + m2 + m5 + m8 + m10 + m13

- The expression is already a sum of minterms, so here's the K-map:

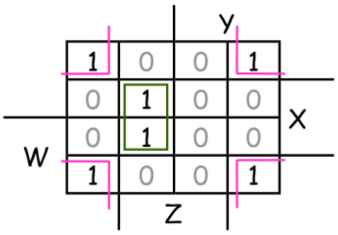


1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

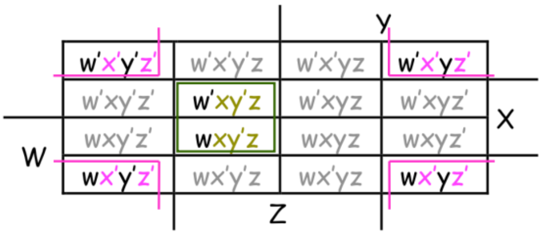


m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

- We can make the following groups, resulting in the MSP $x'z' + xy'z$.



1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1



$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

Prime Implications

Prime implicant

a product term obtained by combining the maximum possible number of adjacent squares.

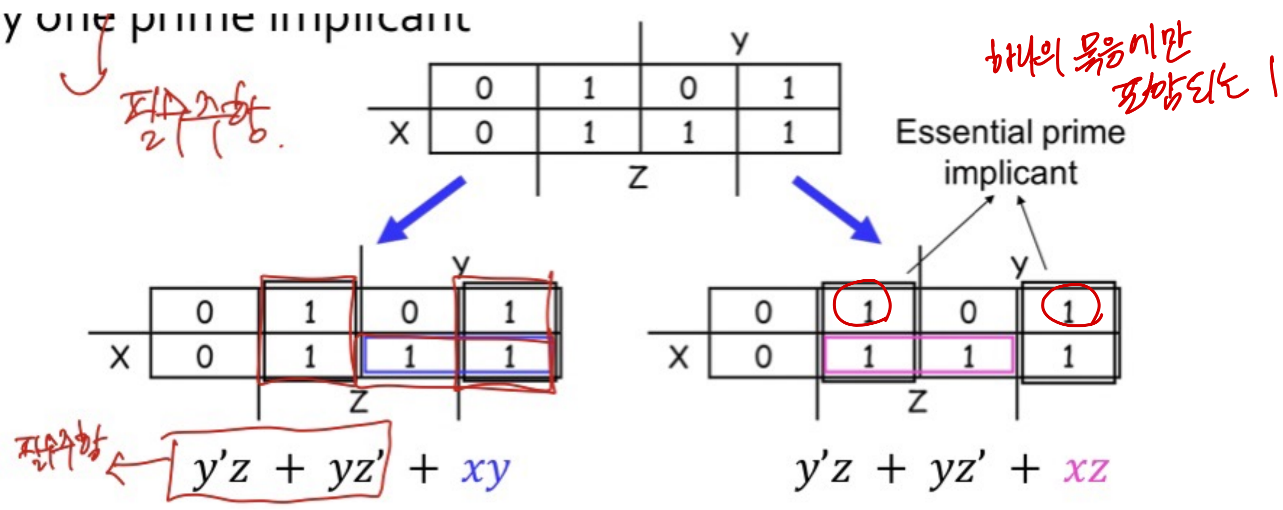
⇒ 주된 항(주항) , and gate 묶인 하나

⇒ 가능한 모든 사각형

essential prime implicant

a minterm in a square is covered by only one prime implicant

⇒ 필수주항 ⇒ 주항 중에 대체할 수 없는 것들



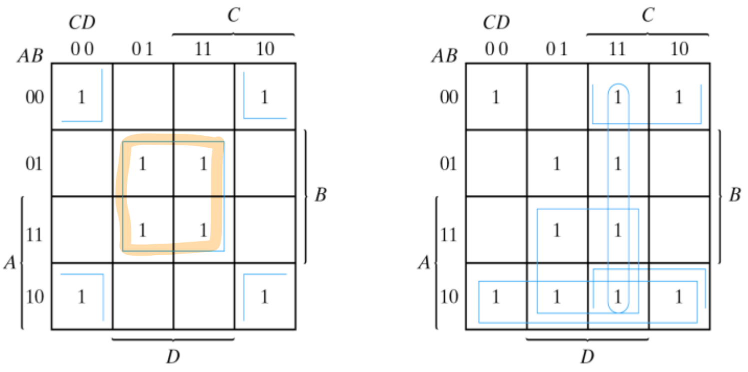
0	1	0	1
0	1	1	1

$\boxed{y'z + yz'} + xy$

$y'z + yz' + xz$

주항 : XY, Y'Z, YZ', XZ

- $F(A,B,C,D) = \Sigma(0,2,3,5,7,8,9,10,11,13,15)$ → 주항을 4개씩 묶음.



(a) Essential prime implicants BD and B′D′

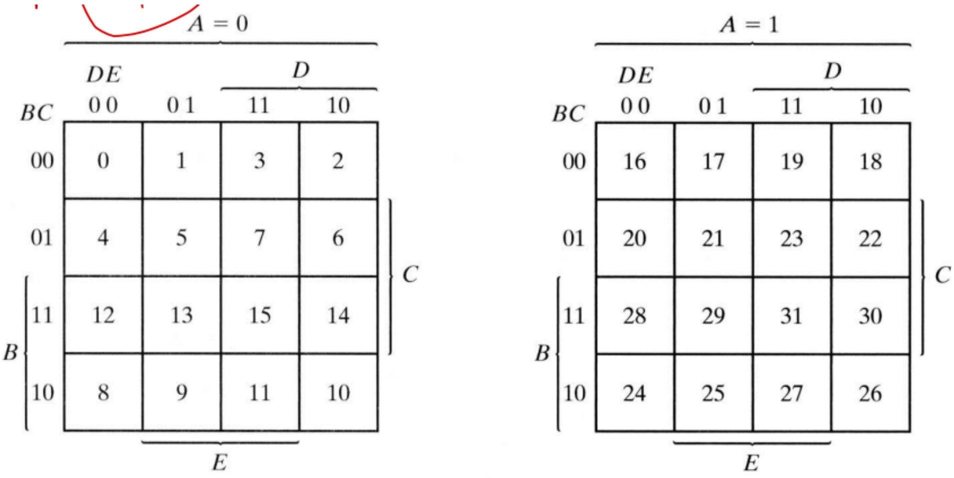
(b) Prime implicants CD, B′C AD, and AB′

$$\begin{aligned} F &= BD + B'D' + CD + AD \\ &= BD + B'D' + CD + AB' \\ &= BD + B'D' + B'C + AD \\ &= BD + B'D' + B'C + AB' \end{aligned}$$

Fig. 3-11 Simplification Using Prime Implicants

Five-variable map

- 6 이상은 비효율적 → 3차원이 필요함
- five : 2차원



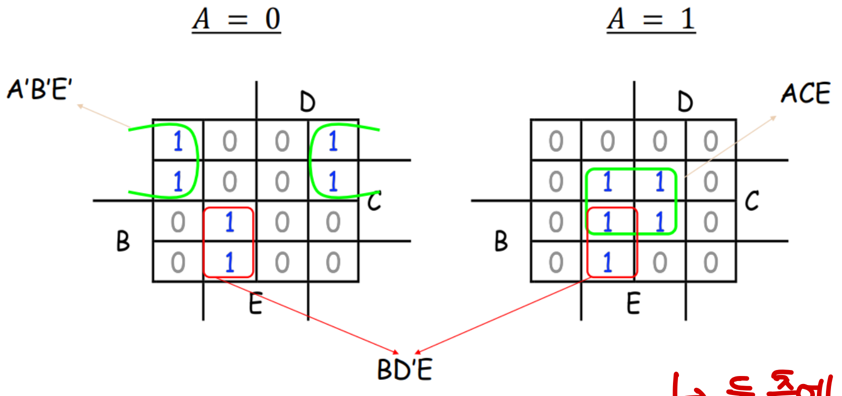
A = 0

BC \ DE	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A = 1

BC \ DE	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

- ex)



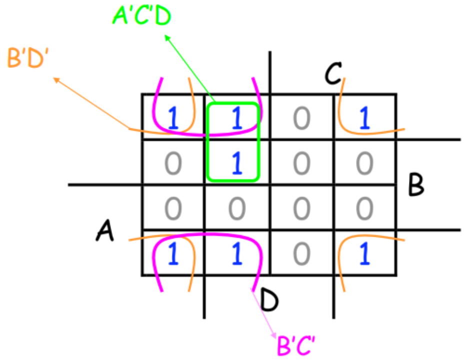
$$F = A'B'E' + BD'E + ACE$$

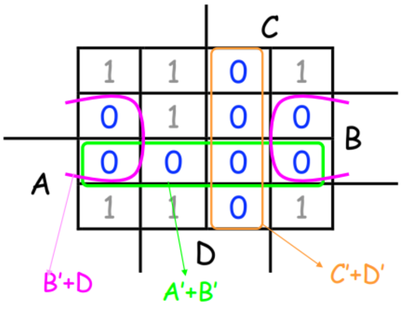
↳ 두 종에 똑같이
있는 것은 그 종 빼고 포함

POS

product of Sums Simplification

- considering the generalized De Morgan’s Theorem
    - 1-0 ⇒ 0으로 묶어주기
    - AND-OR ⇒ OR 말고 AND로 쓰기
    - Minterms-Maxterms
        - $F(A,B,C,D) = \sum(0,1,2,5,8,9,10)$
          $= \prod(3,4,6,7,11,12,13,14,15)$

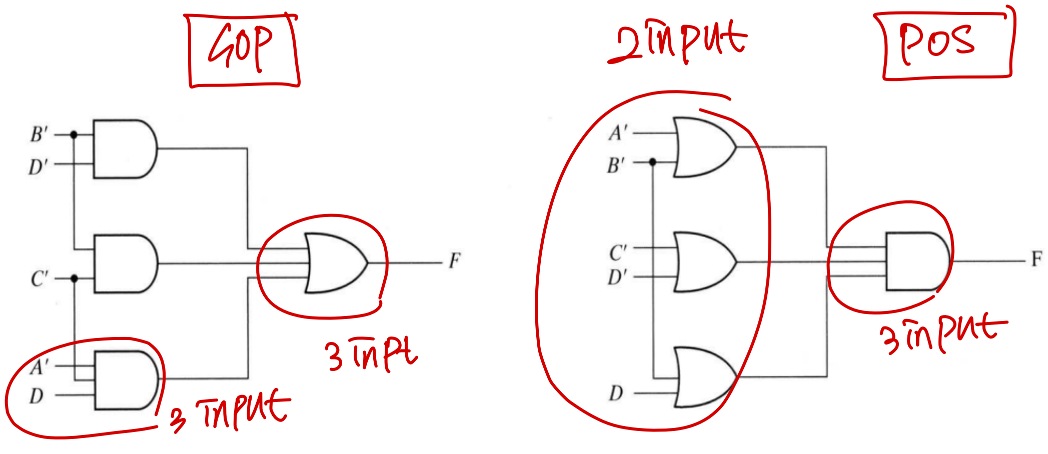




$$F = B'D' + B'C' + A'C'D \qquad F = (A' + B')(C' + D')(B' + D)$$

SOP → POS

- gate implementation



(a) $F = B'D' + B'C' + A'C'D$

(b) $F = (A' + B')(C' + D')(B' + D)$

간략화

Seven Segement Display

I don't care

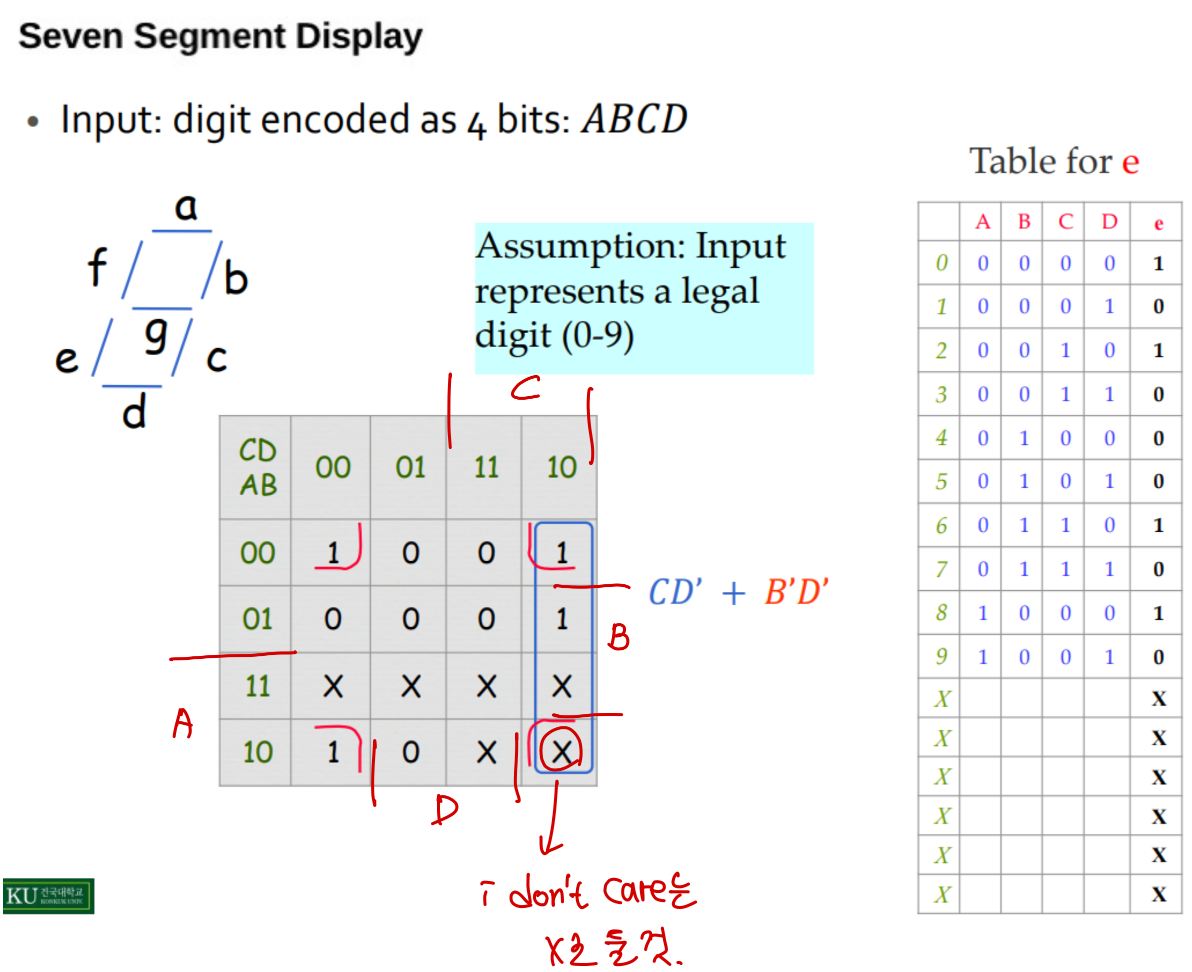
n개의 변수가 있는 function에서 모든 2^n개의 input 조합을 항상 필요로 하지 x

- 만약 특정한 input 조합이 절대 나타나지 않는다면 → guarantee 가능
- 몇몇 ouput이 회로에서 사용되지 않는다면 → 가능

- ex
  - X → i don't care
  - 0, 1 모두로 간주될 수 있음

⇒ 설정 X

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	X
1	1	1	1



Seven Segment Display

- Input: digit encoded as 4 bits: $ABCD$

Assumption: Input represents a legal digit (0-9)

CD AB	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

$CD' + B'D'$

i don't care는 X로 둘것.

Table for e

	A	B	C	D	e
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
X					X
X					X
X					X
X					X
X					X
X					X

NAND

why NAND and NOR gates?

AND, OR보다 더 자주 쓰임

- 제작하기 쉬움
- IC 에서의 기본 gate

NAND Circuits

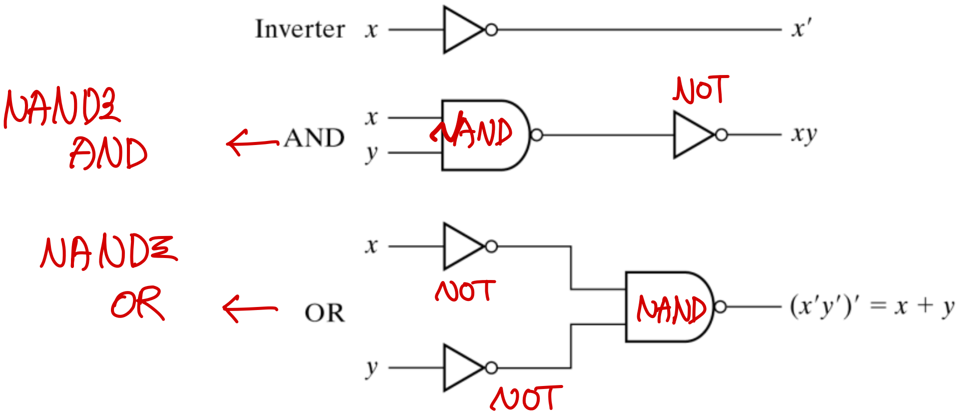
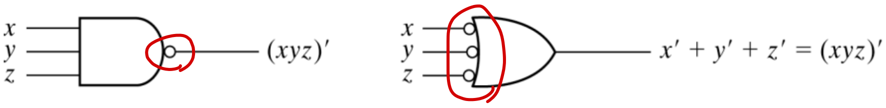


Fig. 3-18 Logic Operations with NAND Gates

AND-invert.

Invert-OR.

DeMorgan's Theorem



(a) AND–invert

⇒ NAND

(b) Invert–OR

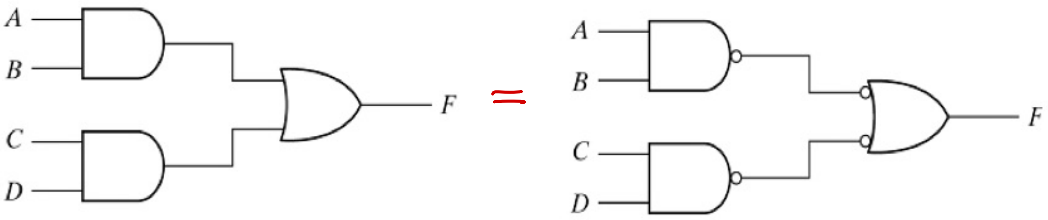
⇒ NAND

- **Conversion into NAND**
  1. covert all AND to NAND with AND-invert symbols
  2. convert all OR to NAND with invert-OR symbols
  3. check all the invert in the diagram
  4. if not compensated, insert an inverter

↳ 없는 inverter를 추가했기에
다시 한 번 점검할 것.

$$F = AB + CD$$

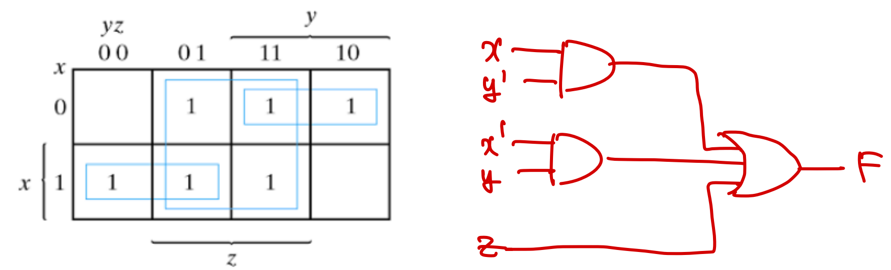
모든 입력 변수에
bubble 존재해야 함.



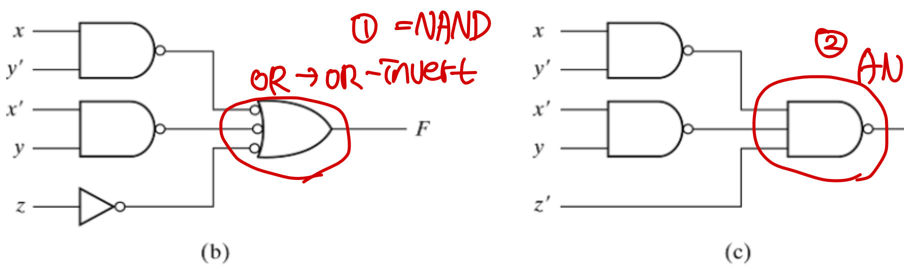
(a) AND-OR

(b) NAND

$$F(x,y,z) = \Sigma(1,2,3,4,5,7) = xy' + x'y + z$$



AND → AND-invert



① =NAND

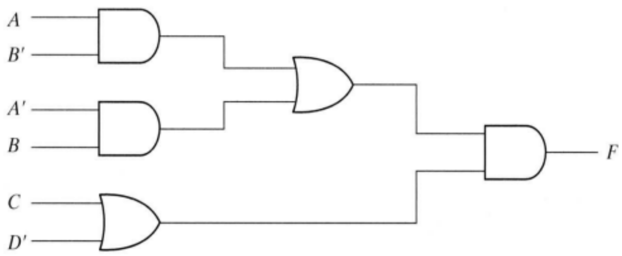
OR → OR-invert

② AND-invert로 가능

(b)

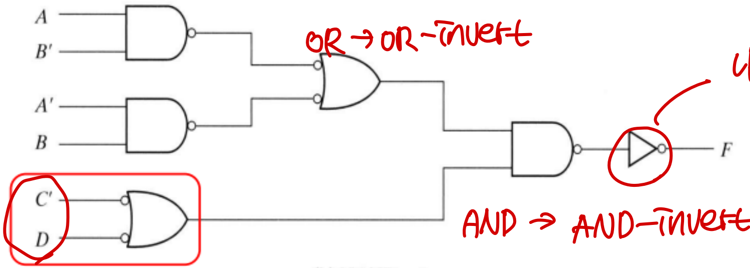
(c)

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



(a) AND-OR gates

AND → AND-invert



OR → OR-invert

나갈때 inverter

AND → AND-invert

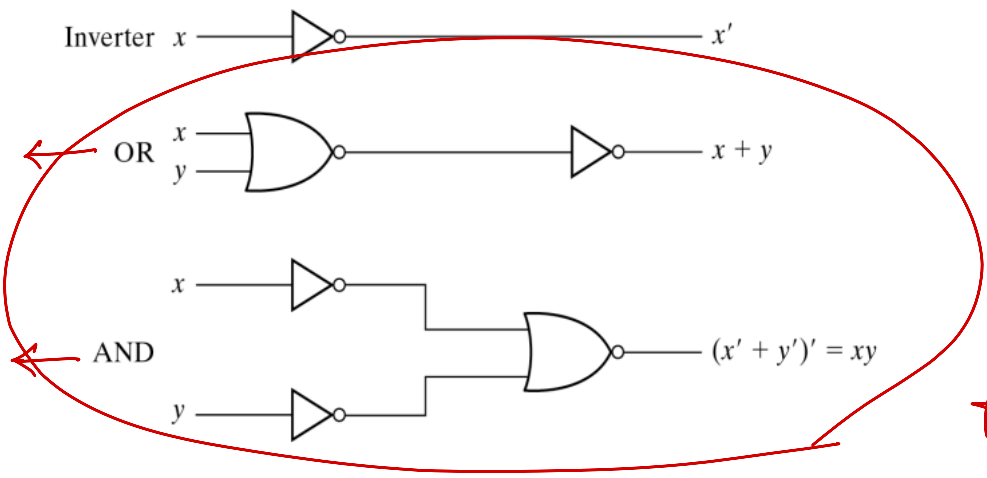
(b) NAND gates

OR → OR-invert

NOR

NOR Circuits

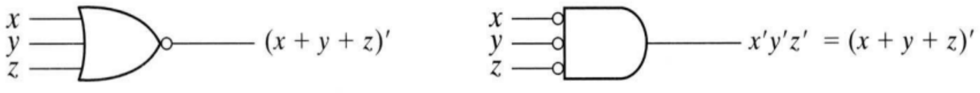
the dual of NANDS



OR을
NOR로

AND를
NOR로

DeMorgan's Theorem



(a) OR–invert

(a) Invert–AND

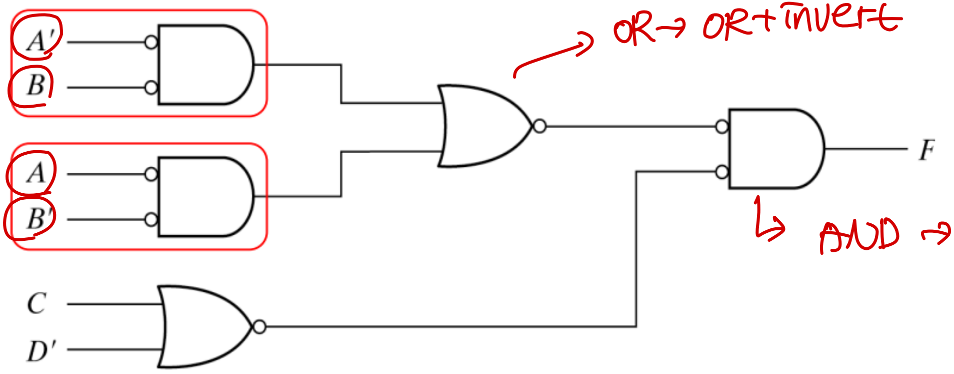
=
둘다 NOR

example

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

AND→ inverter+ AND

OR→ OR+ invert



↳ AND → invert+AND

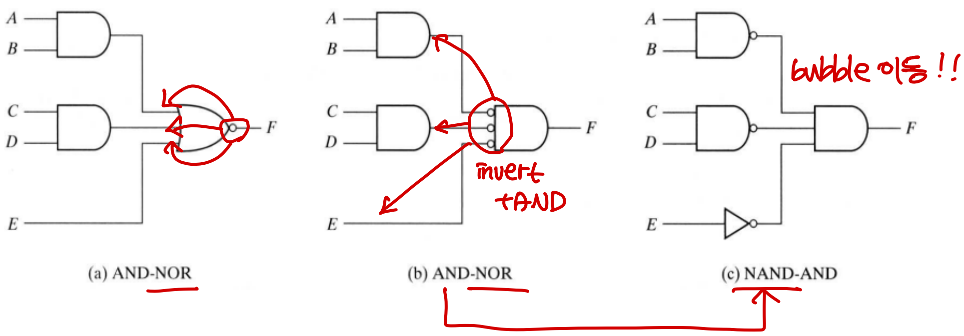
OR→ OR+ invert

AND-OR invert

the complement of SOP

- AND-NOR = NAND-AND

$$F = (AB + CD + E)'$$



bubble 이동!!

invert +AND

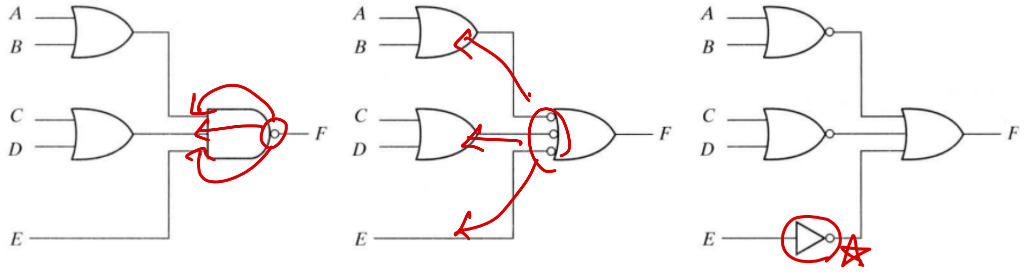
(a) AND-NOR (b) AND-NOR (c) NAND-AND

OR-AND Invert

the complement of POS

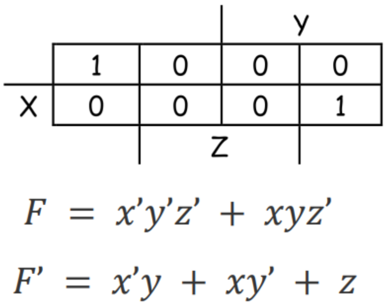
- OR-NAND = NOR-OR

$$F = [(A + B)(C + D)E]'$$



(a) OR-NAND (b) OR-NAND (c) NOR-OR

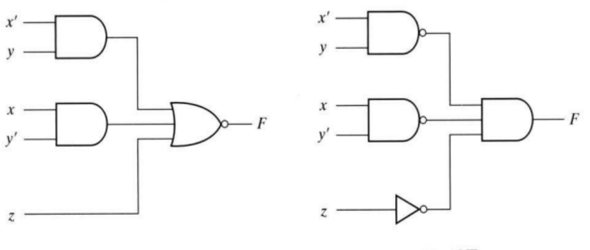
Example



			y	
	1	0	0	0
X	0	0	0	1
		Z		

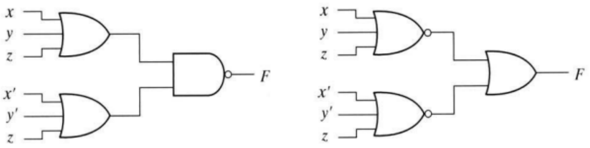
$F = x'y'z' + xyz'$

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



AND-NOR NAND-AND

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

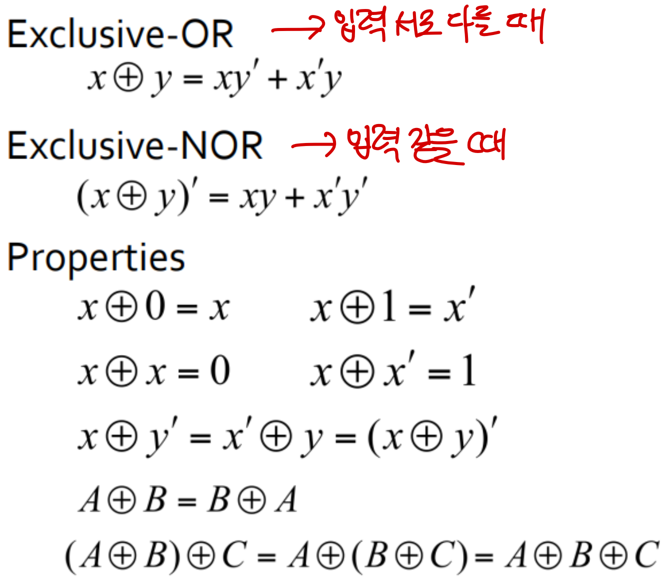


OR-NAND NOR-OR

(c) $F = [(x + y + z)(x' + y' + z)]'$

XOR Function

XOR → sum of minterm or input이 서로 다를 때



Exclusive-OR → 입력 서로 다를 때

$x \oplus y = xy' + x'y$

Exclusive-NOR → 입력 같을 때

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

Properties

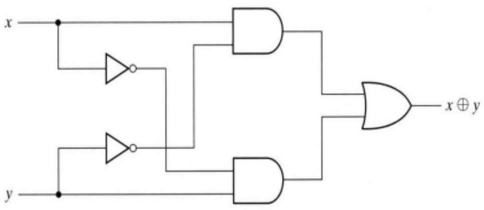
$x \oplus 0 = x \qquad x \oplus 1 = x'$

$x \oplus x = 0 \qquad x \oplus x' = 1$

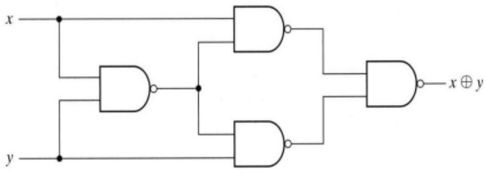
$x \oplus y' = x' \oplus y = (x \oplus y)'$

$A \oplus B = B \oplus A$

$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$



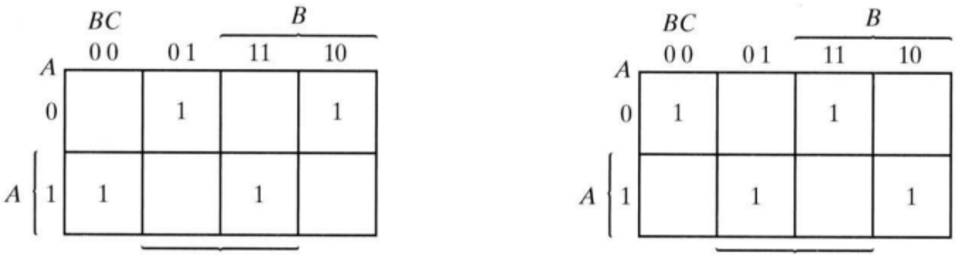
(a) With AND-OR-NOT gates



(b) With NAND gates

⇒ multiple input은 어려움

Odd Function



	00	01	11	10
0		1		1
1	1		1	

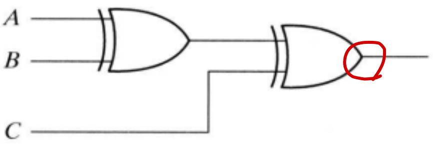
	00	01	11	10
0	1		1	
1		1		1

(a) Odd function
$F = A \oplus B \oplus C$

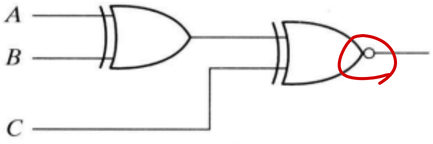
(a) Even function
$F = (A \oplus B \oplus C)'$

FIGURE 3-33
Map for a Three-variable Exclusive-OR Function

bubble 유무



(a) 3-input odd function



(b) 3-input even function

Parity Generation and checking

even-parity generator

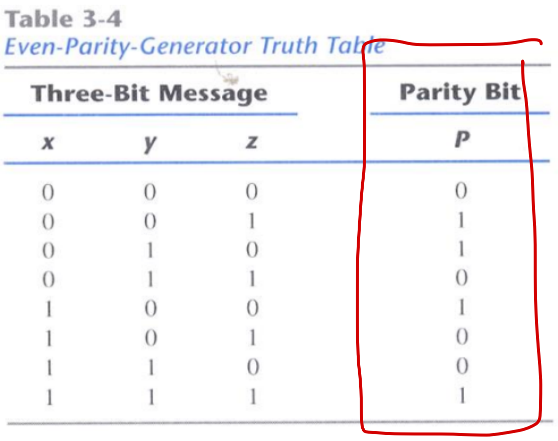
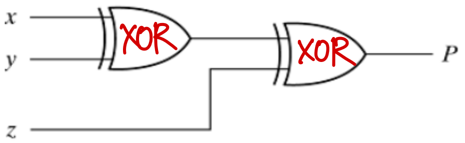


Table 3-4
Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	*y*	*z*	*P*
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

1의 개수가 짝수

Even-Parity Generator



(a) 3-bit even parity generator

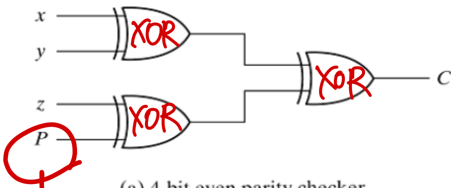
even-parity checker

Even-Parity Checker

Table 3-5
Even-Parity-Checker Truth Table

Four Bits Received				Parity Error Check
x	**y**	**z**	**P**	**C**
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

generator에서
온애



(a) 4-bit even parity checker

input 하나 추가

Reuse the parity generator!