Digital Logic Design

Sung-Soo Lim

KESL
Kookmin Univ. Embedded System Lab

Today's Topic

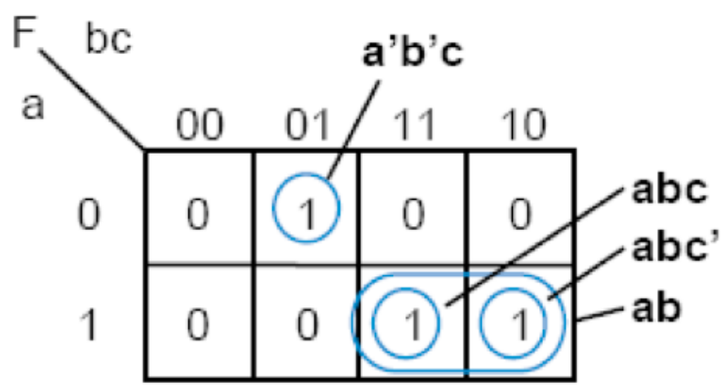
카르노맵 최적화를 체계적으로 수행하기

Terminology

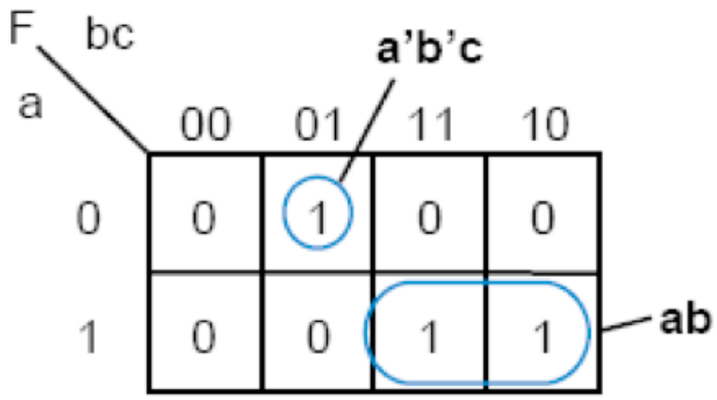
- Literal
  - 변수가 나타나는 형태
- Implicant
  - 리터럴의 곱의 형태
  - 카르노맵에서는 써클로 표현
- Prime implicant
  - 더 이상 확장할 수 없는 Implicant
  - 더 이상 확장할 수 없는 써클

F = a'b'c + abc + abc'

6 Literals
(a, a', b, b', c, c')



4 implicants

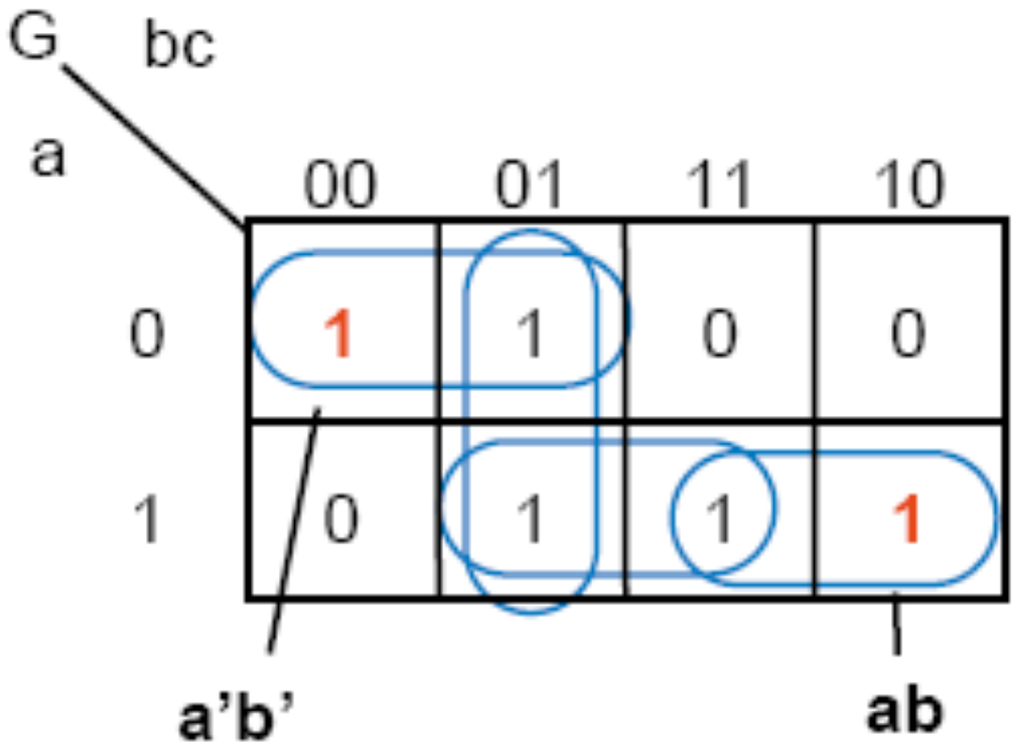


2 prime
Implicants

KESL
Kookmin Univ. Embedded Systems Lab

Terminology

- Essential prime implicant
  - 해당 함수를 만족시키기 위해 반드시 필요한 Prime Implicant
  - 모든 EPI (Essential prime implicant)는 꼭 포함해야 함
  - Non-EPI는 포함될 수도 있고 포함되지 않을 수도 있음

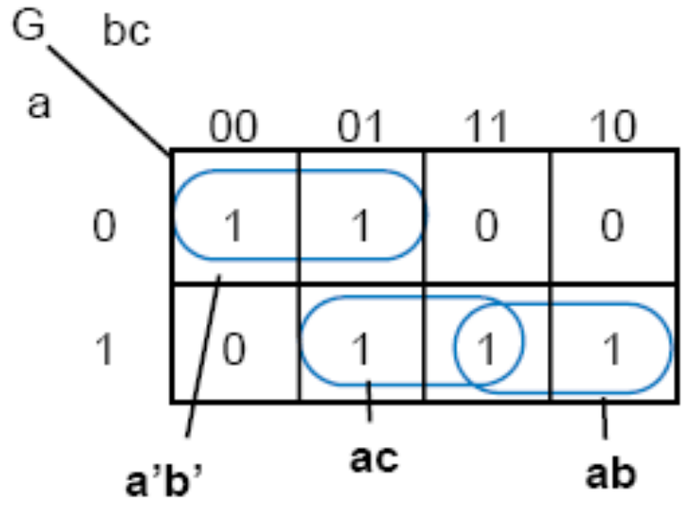


4 prime implicants

2 essential prime implicants

Terminology

- Cover
  - F = 1을 만족시키는 모든 써클의 집합
  - 해당 함수를 구현할 수 있는 방법
- Cost
  - Number of gates + number of inputs
  - Ignore complement (NOT)

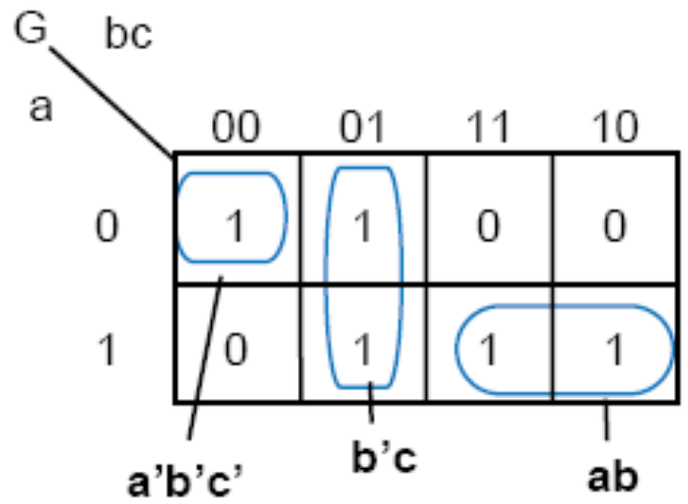


Cover 1: G = a'b' + ac + ab

G = a'b'c' + b'c + ab

1 3-input AND gates
2 2-input AND gates
1 3-input OR gates

cost = 4 + 10 = 14



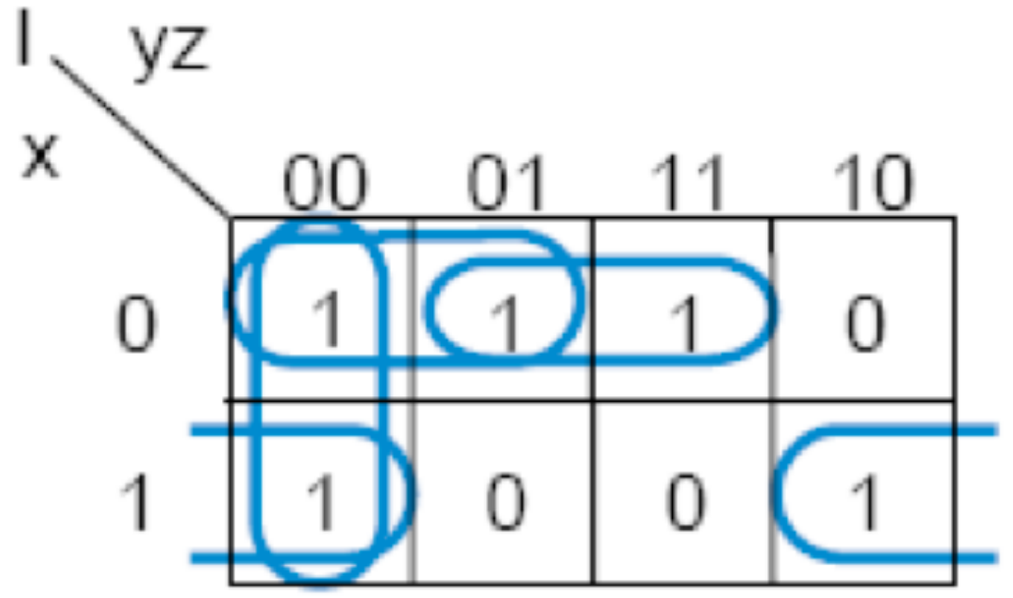
Cover 2: G = a'b'c' + b'c + ab

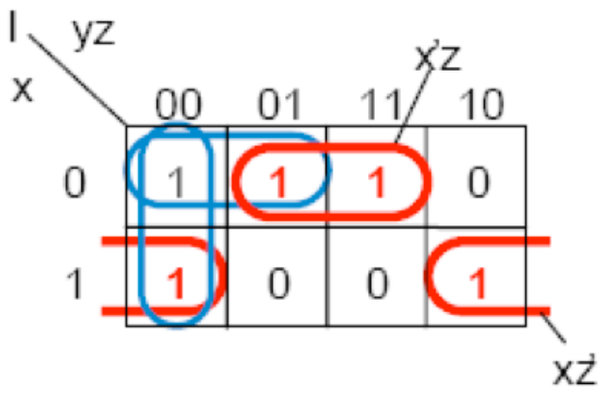
KESL
Kookmin Univ. Embedded Systems Lab

K-Map 최적화 절차

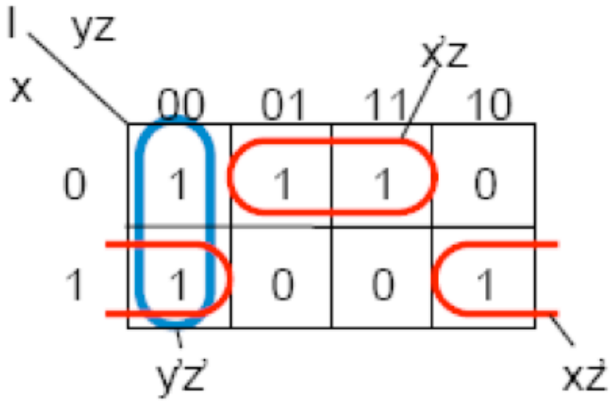
- (1) 모든 prime implicant를 생성
- (2) EPI를 모두 찾는다
- (3) EPI만으로 F=1을 만족?
  - Yes - were done!
  - No – Non-EPI 중에서 선택하여 Complete Cover를 만들 수 있는 PI들을 찾아냄

K-Map Optimization Example (1)



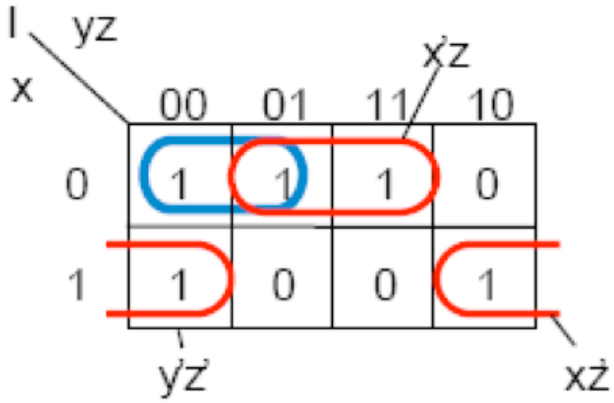


Cover:
e.p.i : x'z, xz'



Cover 1:
e.p.i : x'z, xz', y'z'

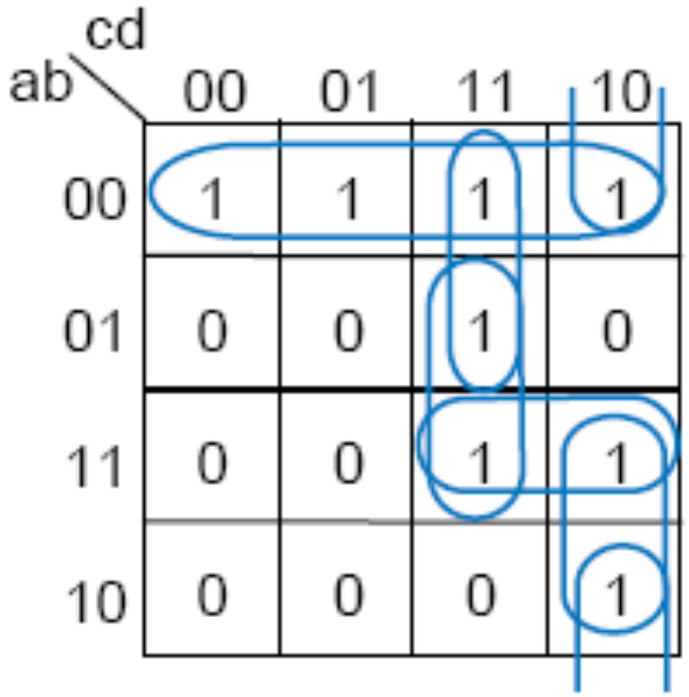
cost = 13

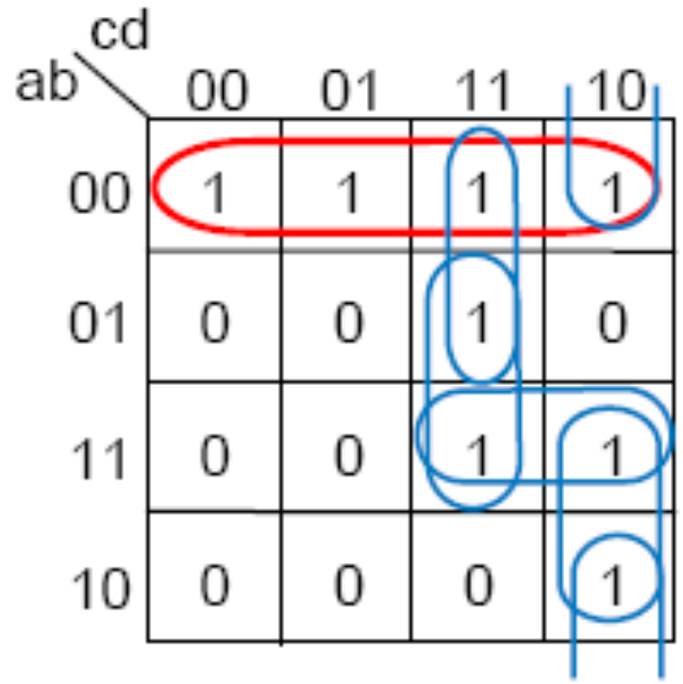


Cover 2:
e.p.i : x'z, xz', x'y'

cost = 13

K-Map Optimization Example (2)

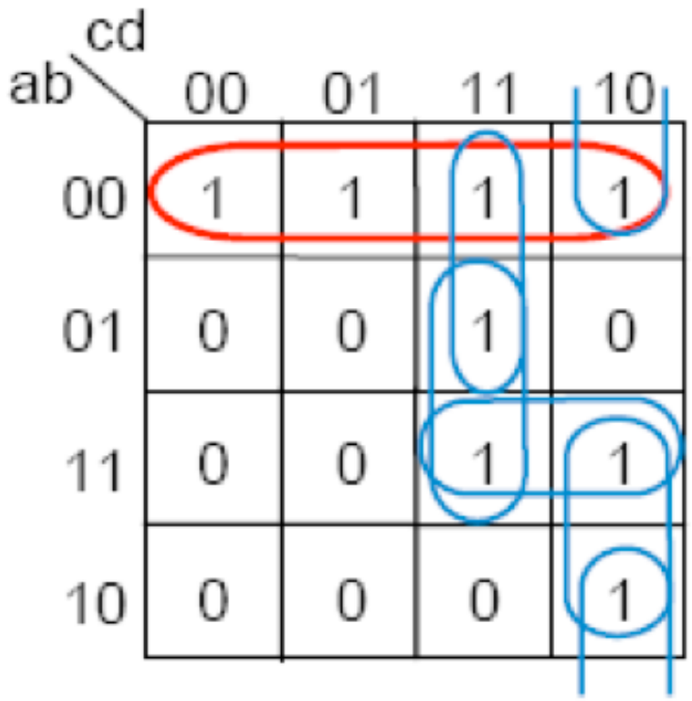




Cover:
e.p.i : a'b'

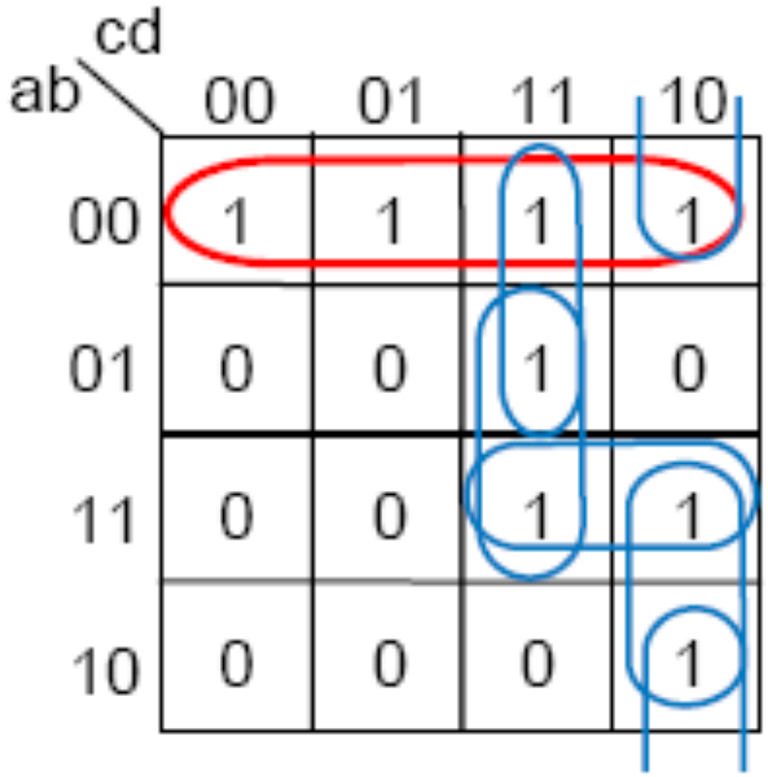
K-Map Optimization Example (2)

ab \ cd	00	01	11	10
00	1	1	1	1
01	0	0	1	0
11	0	0	1	1
10	0	0	0	1

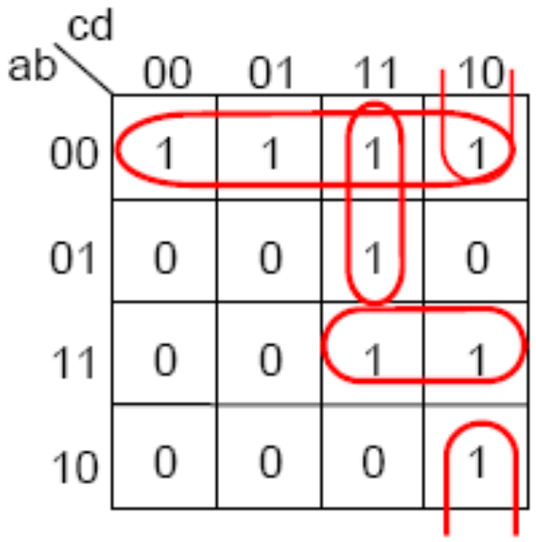


Cover:
e.p.i : a'b'

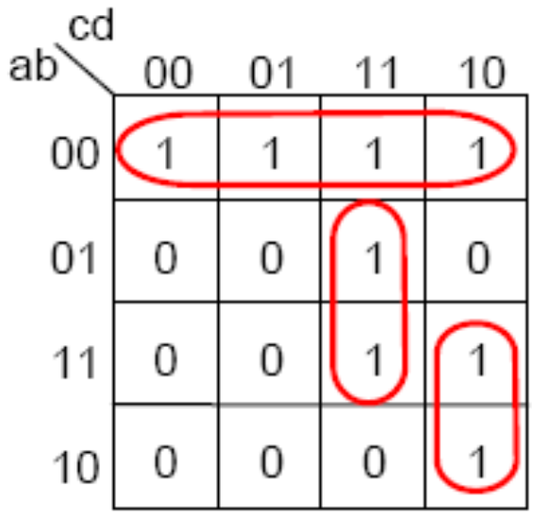
K-Map Optimization Example (2)



Cover:
e.p.i : a'b'



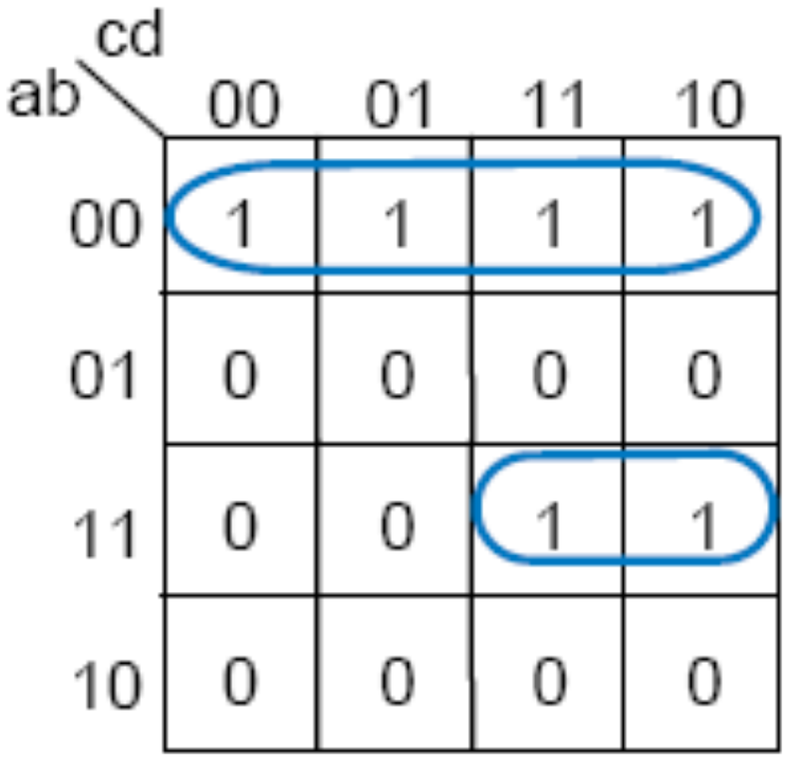
Cover 1 (cost = 20)
F = a'b' + a'cd + abc + b'cd'



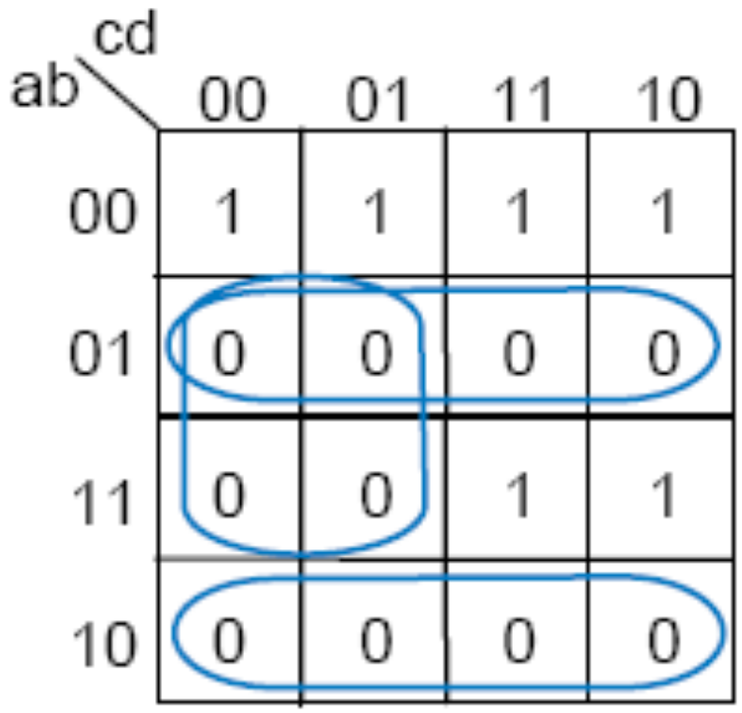
Cover 2 (cost = 15)
F = a'b' + bcd + acd'

K-Map for SOP vs. POS

- K-maps 표현 방법은 sum-of-products과 직접적으로 연관
- Product-of-sums 표현과 K-map의 관계는?



Sum-of-products
F = a'b' + abc



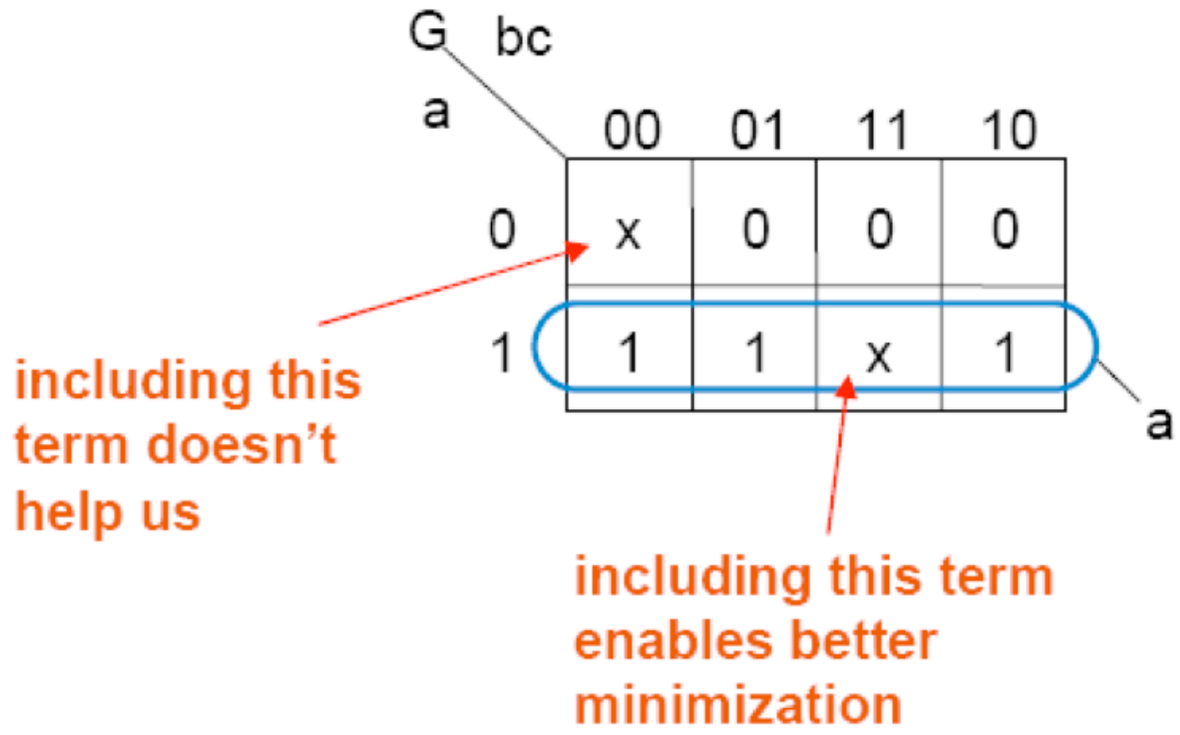
Product-of-sums
F = (a+b')(b'+c)(a'+b)

Don’t Care

- Don’t Care 입력
  - 해당 입력에 대한 출력은 시스템 구현에 전혀 영향이 없음
    - i.e. input condition can never occur
  - K-map에서 ‘X’로 표시
  - 최적화에 활용

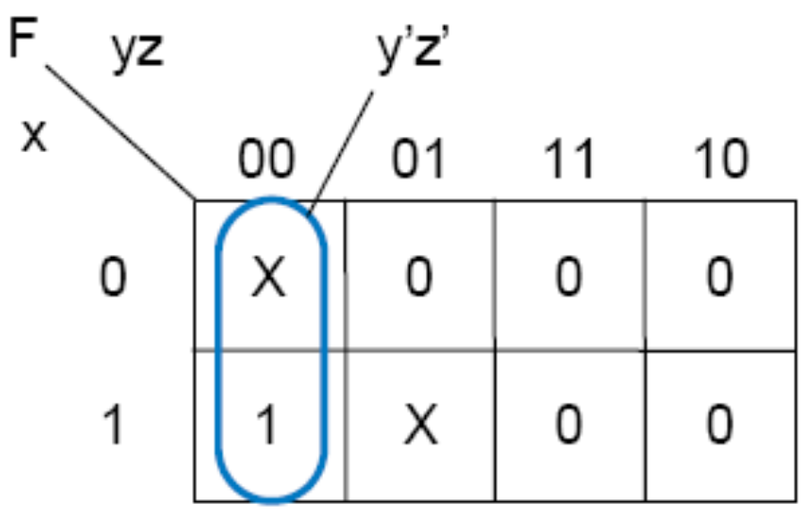
abc = 000 and abc = 111 are unused inputs

a	b	c	Z
0	0	0	x
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	x

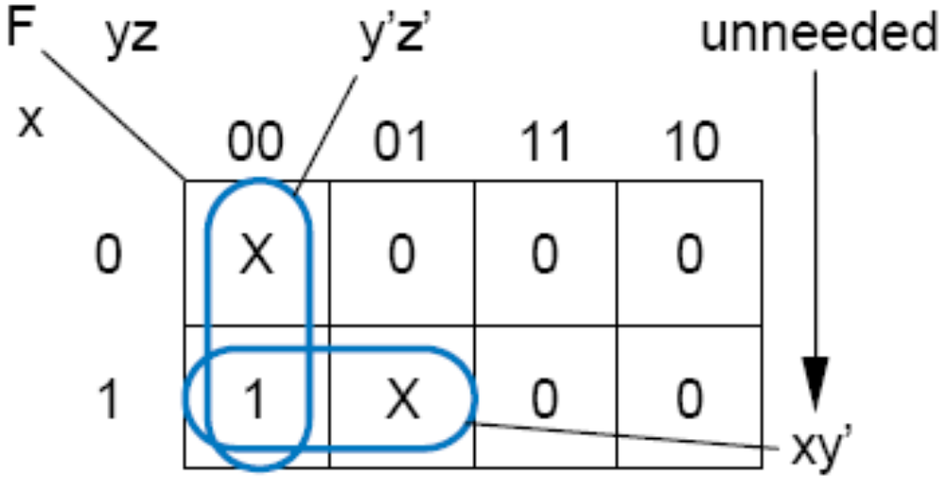


Don’t Care Input 활용하기

- Don’t care 출력은 최적화에 유리한 방향으로 ‘0’ 또는 ‘1’로 가정한다.
  - X 를 포함하는 경우는 최적화에 꼭 필요한 경우에만
  - 다른 X는 절대 포함하지 않는다



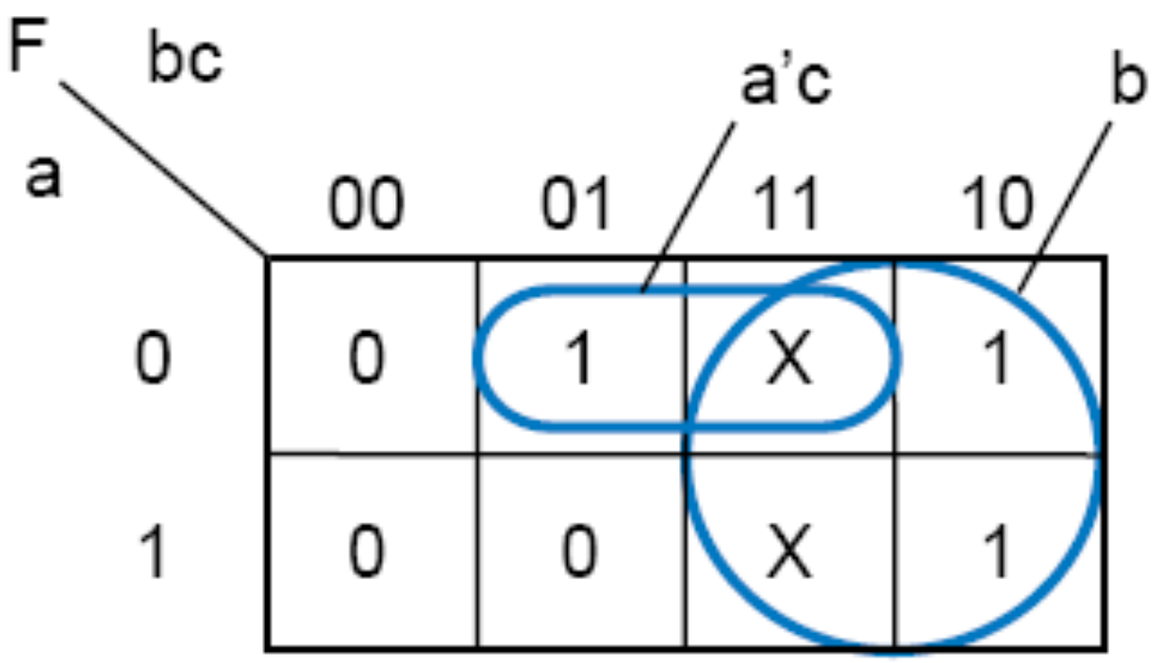
Good use of don’t cares



Unnecessary use of don’t cares; results in extra term

Example – Using Don't Care

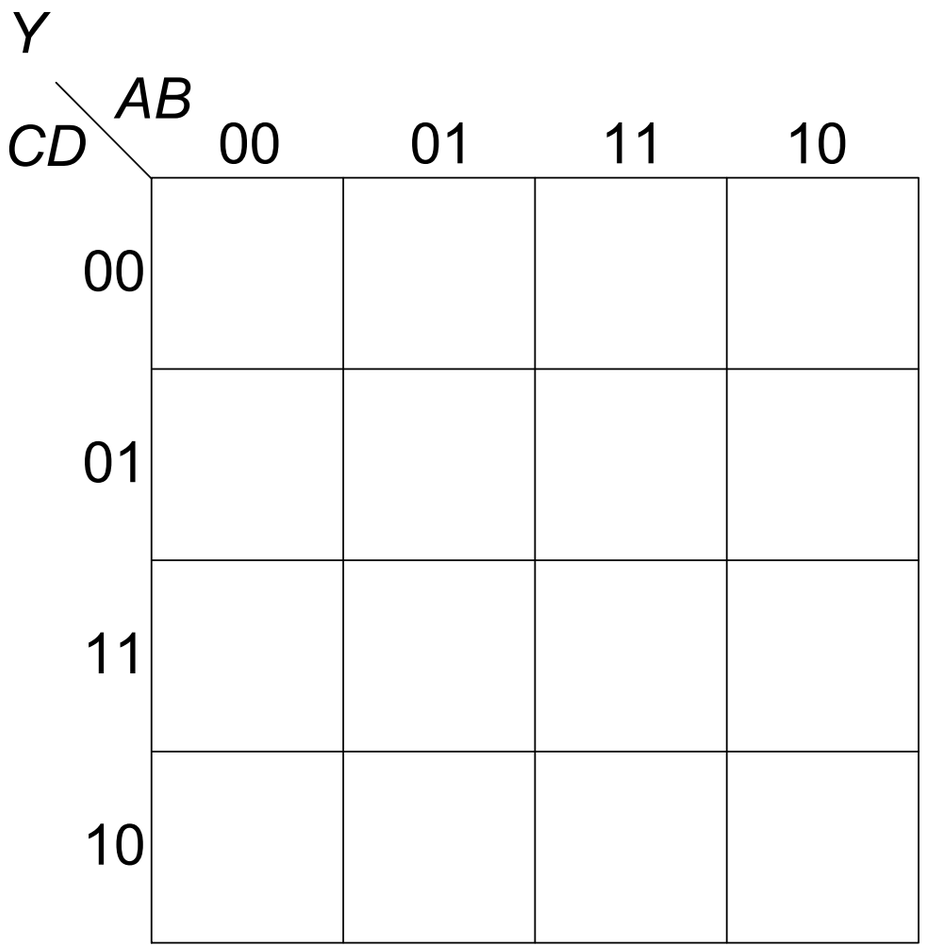
- Minimize:
  - F = a'bc' + abc' + a'b'c
  - Given don't cares: a'bc, abc



F = a'c + b

Using Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



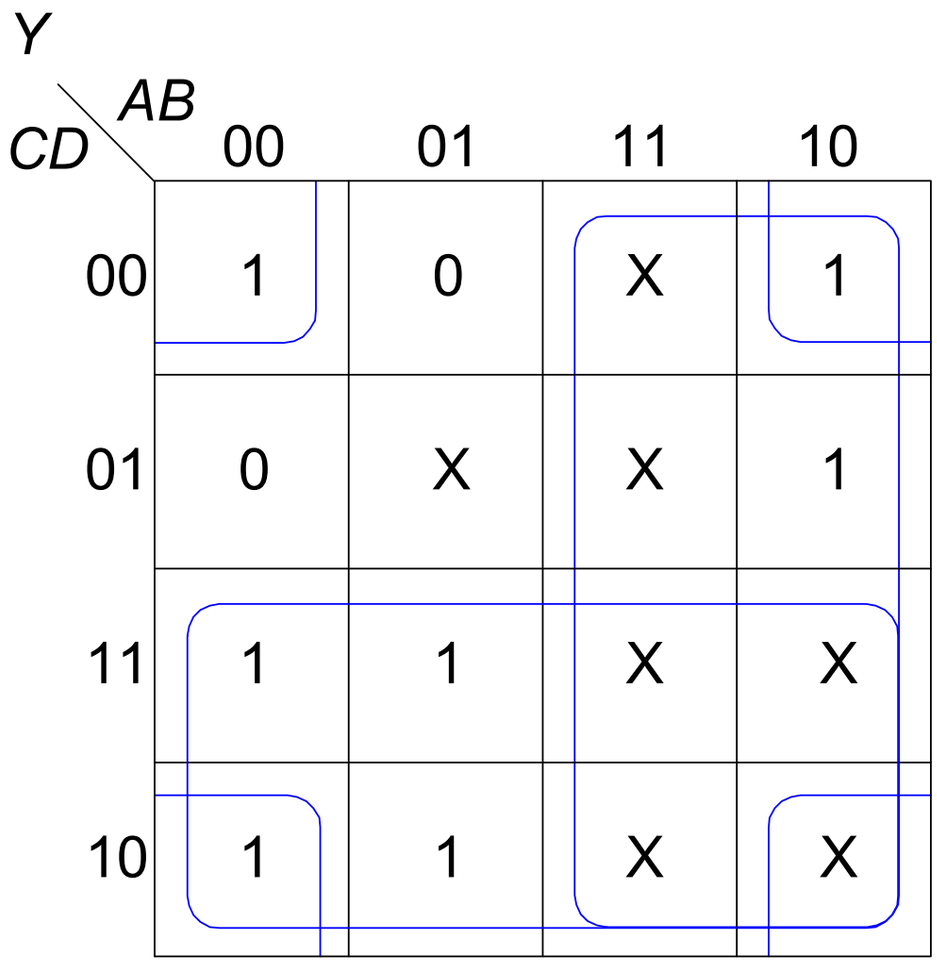
Using Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y / CD \ AB	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

Using Don't Cares

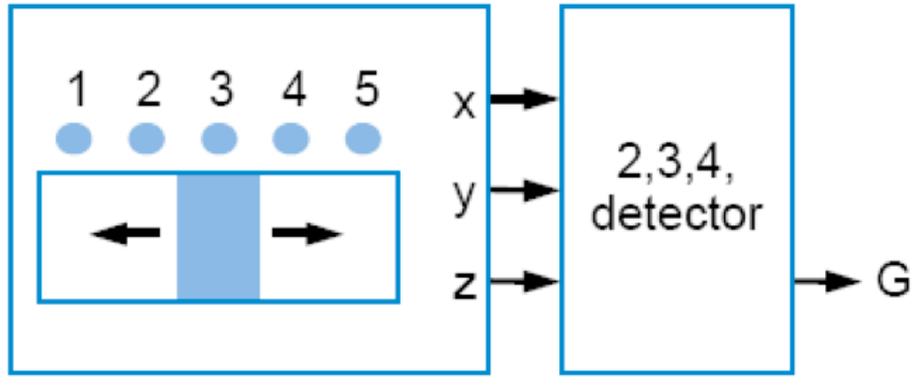
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

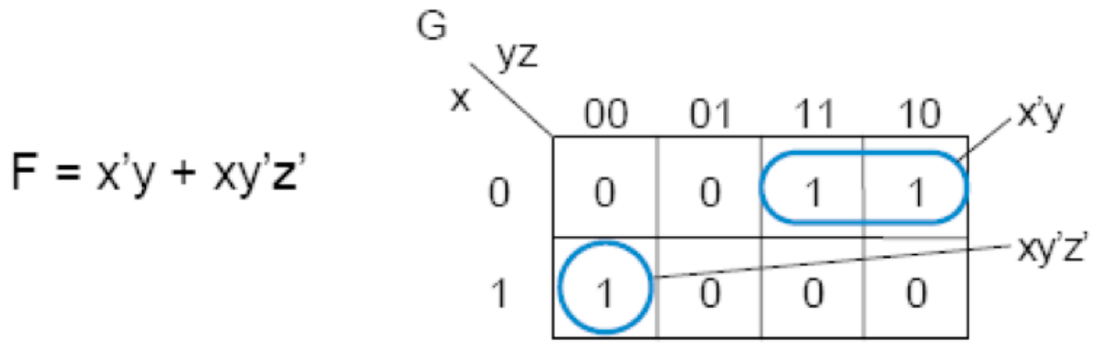


$$Y = A + \overline{B}\overline{D} + C$$

Example – Using Don't Care (2)

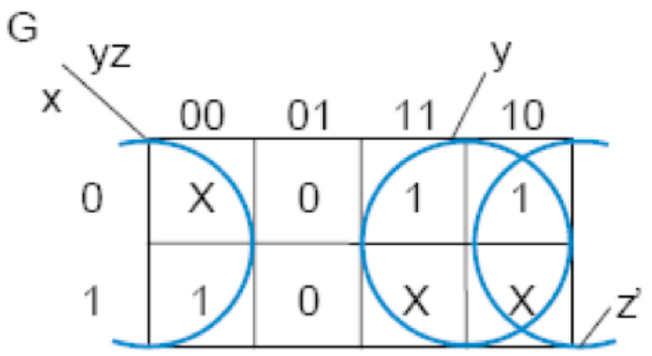
- Switch with 5 positions
  - position 1, xyz = 001
  - position 2, xyz = 010
  - position 3, xyz = 011
  - position 4, xyz = 100
  - position 5, xyz = 101
- Want circuit that
  - Outputs 1 when switch is in position 2, 3, or 4
  - Outputs 0 when switch is in position 1 or 5
  - Note that we never use some input combinations
    - 000, 110, 111





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

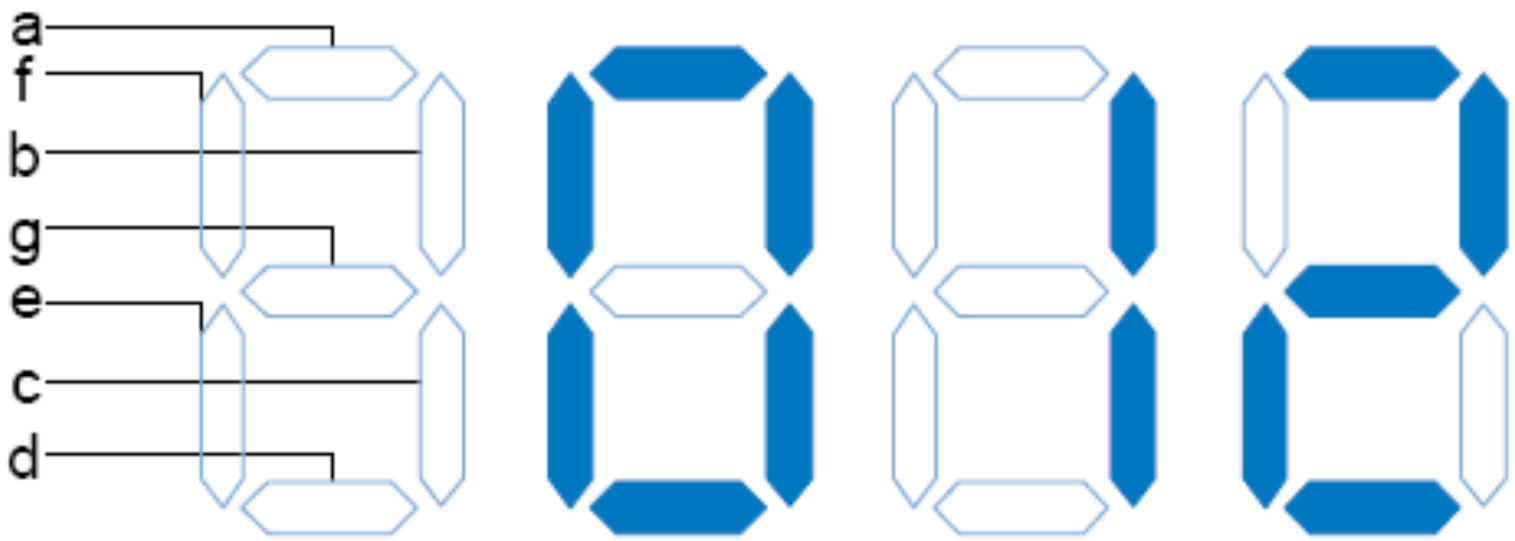
With don't cares:
$F = y + z'$



KESL
Kookmin Univ. Embedded Systems Lab

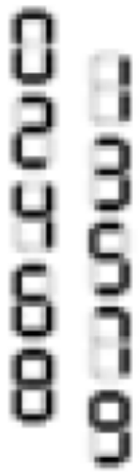
Multiple Output Circuits

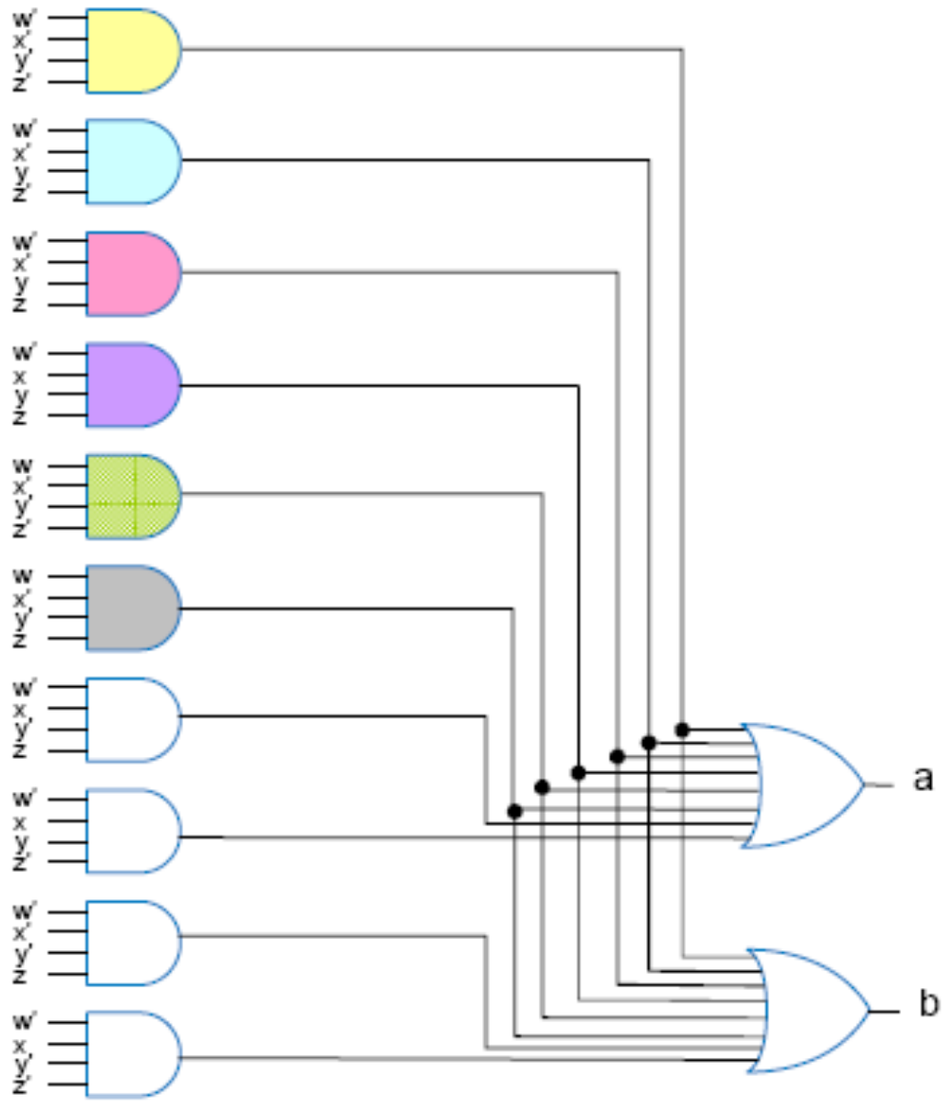
- 여러 출력이 필요한 회로 (사실상 대부분의 회로)
  - Seven-segment display
- MO (Multiple-Output) 회로를 최적화하는 기본 원리는?
  - 게이트 수를 줄여라!!



Seven Segment Display

w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



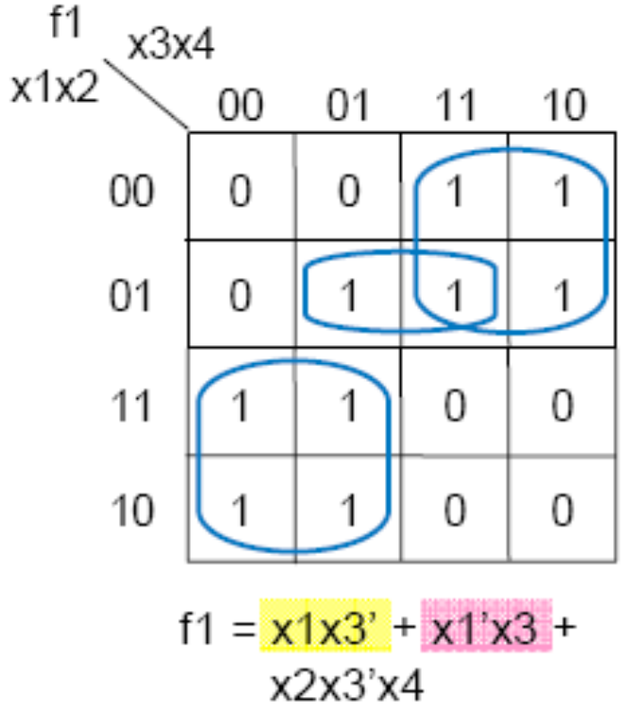


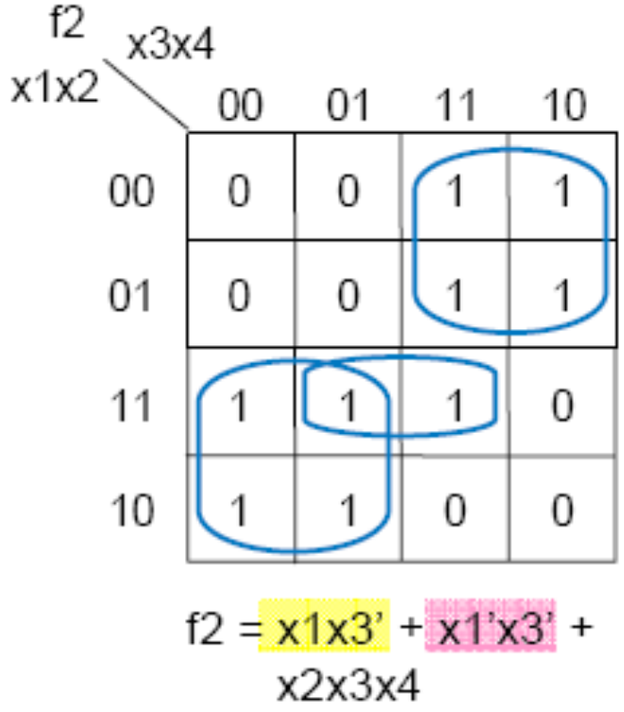
a = w'x'y'z' + w'x'yz' + w'x'yz + w'xy'z + w'xyz' + w'xyz + wx'y'z' + wx'y'z

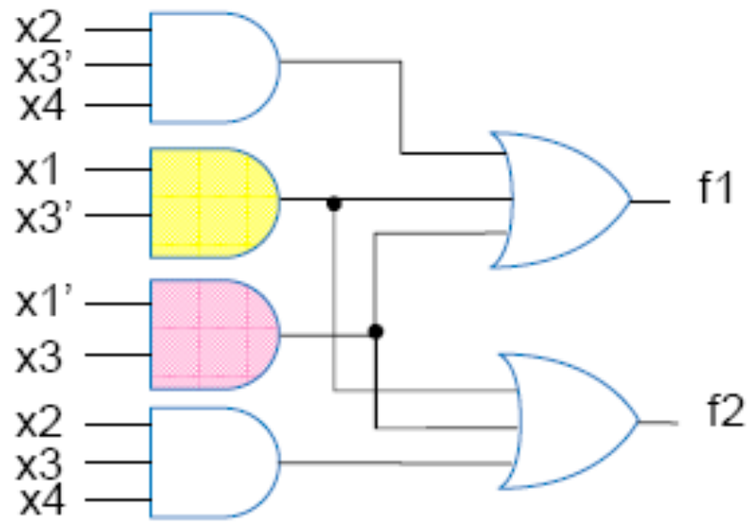
b = w'x'y'z' + w'x'y'z + w'x'yz' + w'x'yz + w'xy'z' + w'xyz + wx'y'z' + wx'y'z

Multiple Output Circuits 사례 (2)

- 다중 출력 K-maps 에서 비용 최적화
  - 로컬 최적화 vs. 글로벌 최적화
  - 공유 가능한 회로를 찾아라







Multiple Output Circuits 사례 (3)

f3

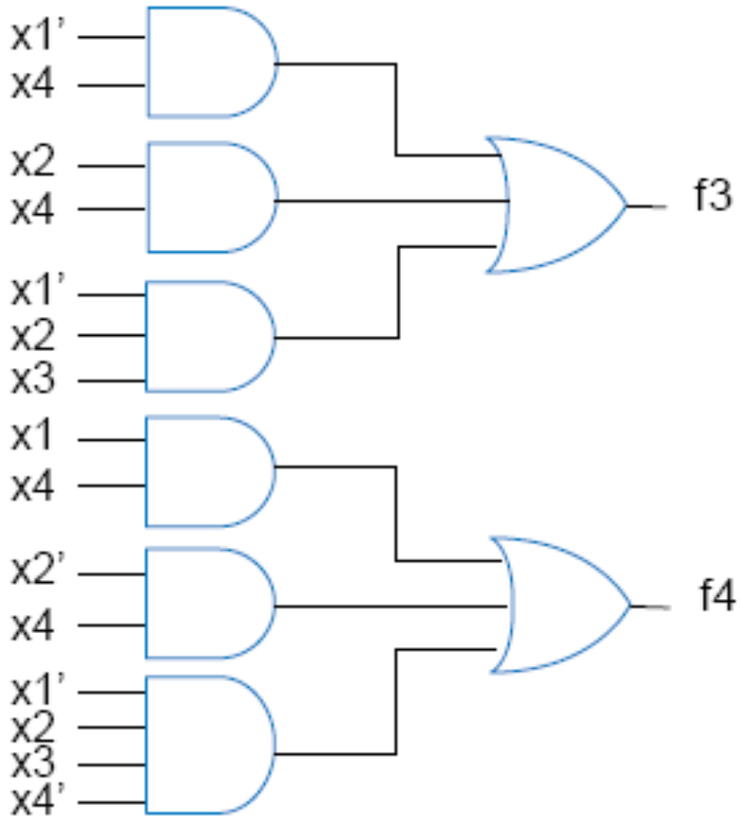
x1x2 \ x3x4	00	01	11	10
00	0	0	0	0
01	1	1	1	0
11	1	1	1	0
10	0	1	0	0

f3 = x1'x4 + x2x4 + x1'x2x3

f4

x1x2 \ x3x4	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	1	0	1	1
10	0	1	0	0

f4 = x1x4 + x2'x4 + x1'x2x3x4'



nothing to combine

Multiple Output Circuits 사례 (3)

- 어떻게든 공유 가능한 회로를 찾아내기

f3

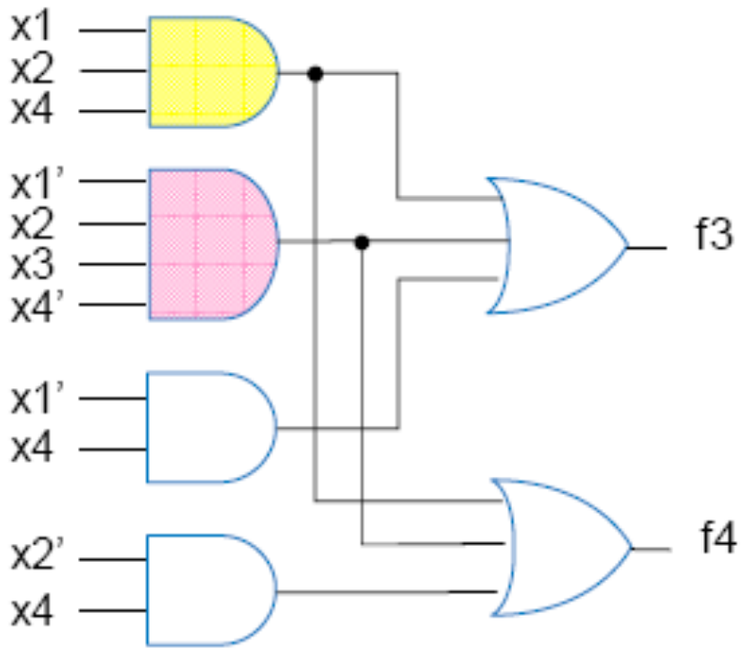
x1x2 \ x3x4	00	01	11	10
00	0	0	0	0
01	1	1	1	0
11	1	1	1	0
10	0	1	0	0

f3b = x1x2x4 + x1'x2x3x4' + x1'x4

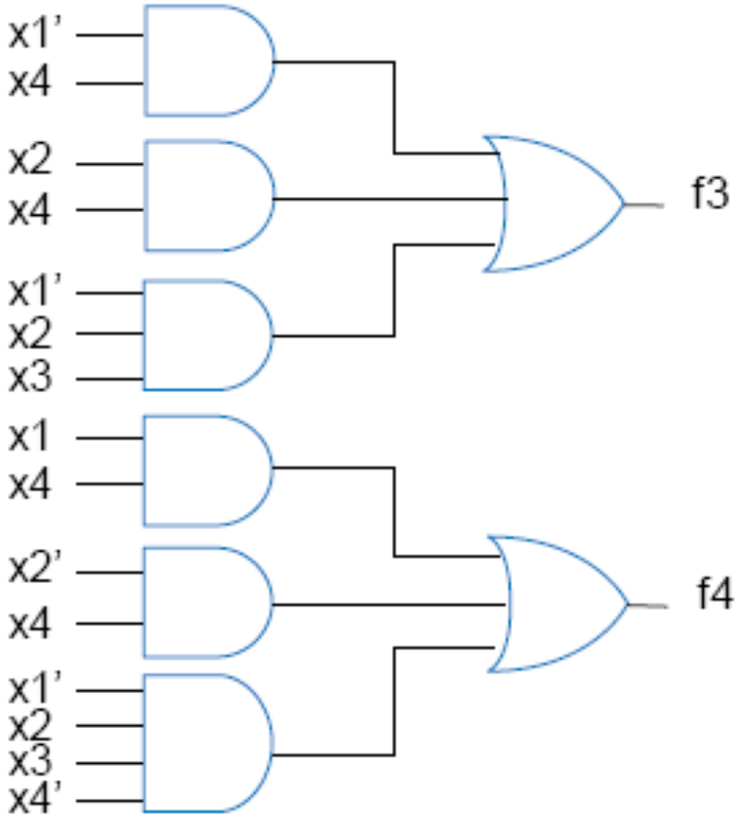
f4

x1x2 \ x3x4	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	1	0	1	1
10	0	1	0	0

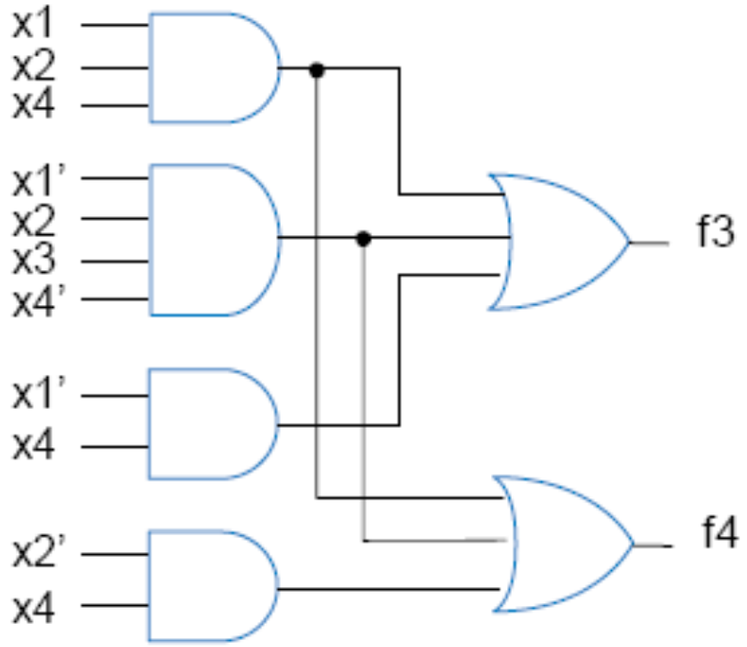
f4b = x1x2x4 + x1'x2x3x4' + x2'x4



Multiple Output Circuits Solutions 비교



cost = 8 + 21 = 29



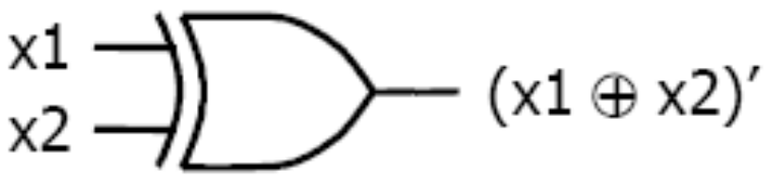
cost = 6 + 17 = 23

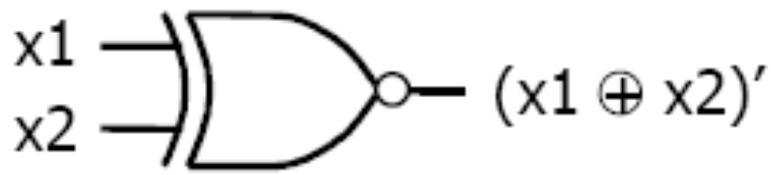
누가 이런 최적화를 수행해야 하나?

소프트웨어 도구!!

추가로 알아야 하는 논리 게이트

- XOR gate – exclusive OR
  - 두 입력이 다를 때 True
  - 입력이 3개 이상일 때에는?
- XNOR gate – complementing XOR gate
  - 입력이 3개 이상일 때에는?





XOR

a	b	F
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

a	b	F
0	0	1
0	1	0
1	0	0
1	1	1