CSE 140 Discussion #2
Karnaugh Maps &
2-level Logic Minimization

Chengmo Yang

04/13/2009

Concepts for sum-of-product forms

- **Literal:** a variable or its compliment
- **Minterm:** a product involving all the inputs to the function with each variable appearing exactly ones
- **Implicant:** product (AND) of one or more literals
- **Prime Implicant:** an implicant not contained in any other bigger implicant with fewer literals
- **Essential Prime Implicant:** a prime implicant containing a minterm which is not contained in any other prime implicant

Concepts for product-of-sum forms

- **Literal:** a variable or its compliment
- **Maxterm:** a sum involving all the inputs to the function with each variable appearing exactly once
- **Implicate:** sum (OR) of one or more literals
- **Prime Implicate:** an implicate not contained in any other bigger implicate with fewer literals
- **Essential Prime Implicate:** a prime implicate containing a maxterm which is not contained in any other prime implicate

Minterm vs. maxterm representation

F = a xor b xor c
= (a'b+ab') xor c
= (a'b+ab')c' + (a'b'+ab)c
= a'bc' + ab'c' + a'b'c + abc
= Σ (1, 2, 4, 7)
= Π (0, 3, 5, 6)
= (a+b+c)(a+b'+c')(a'+b+c')(a'+b'+c)

F' = Σ (0, 3, 5, 6)
= a'b'c' + a'bc + ab'c + abc'
= Π (1, 2, 4, 7)
= (a+b+c')(a+b'+c)(a'+b+c)(a'+b'+'c)

Questions:

F = a *xor* b
G = a *xnor* b
F = G or F = G' ?

F = a *xor* b *xor* c
G = a *xnor* b *xnor* c
F = G or F = G' ?

K-Map example

Exercises 2.1(d) in text book

a	b	c	d	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

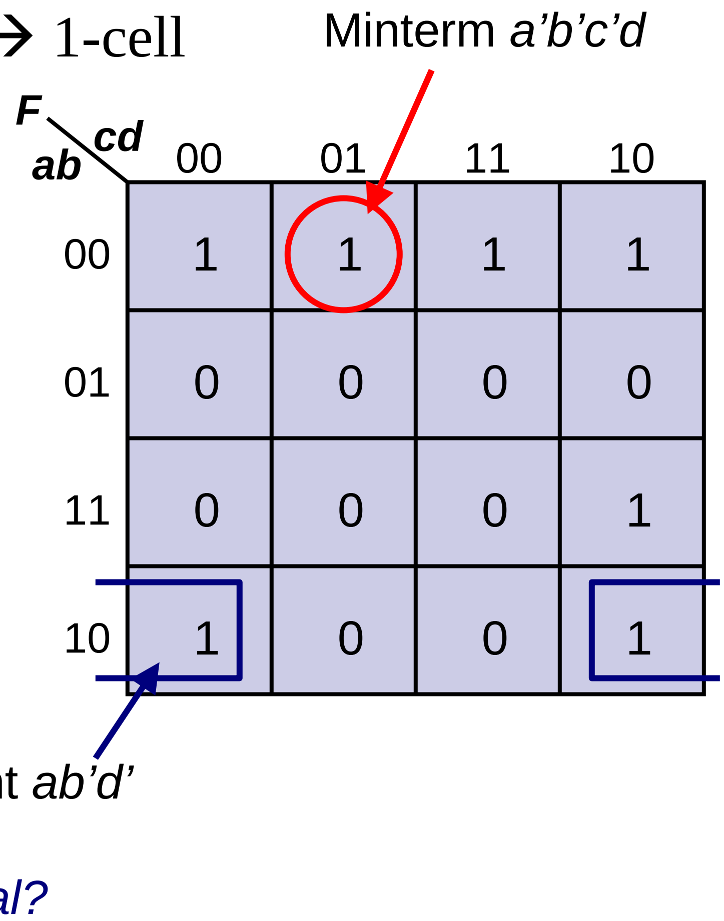
F ***ab*** \ ***cd***	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	1
10	1	0	0	1

K-Map example

Exercises 2.1(d) in text book

a	b	c	d	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Each minterm → 1-cell

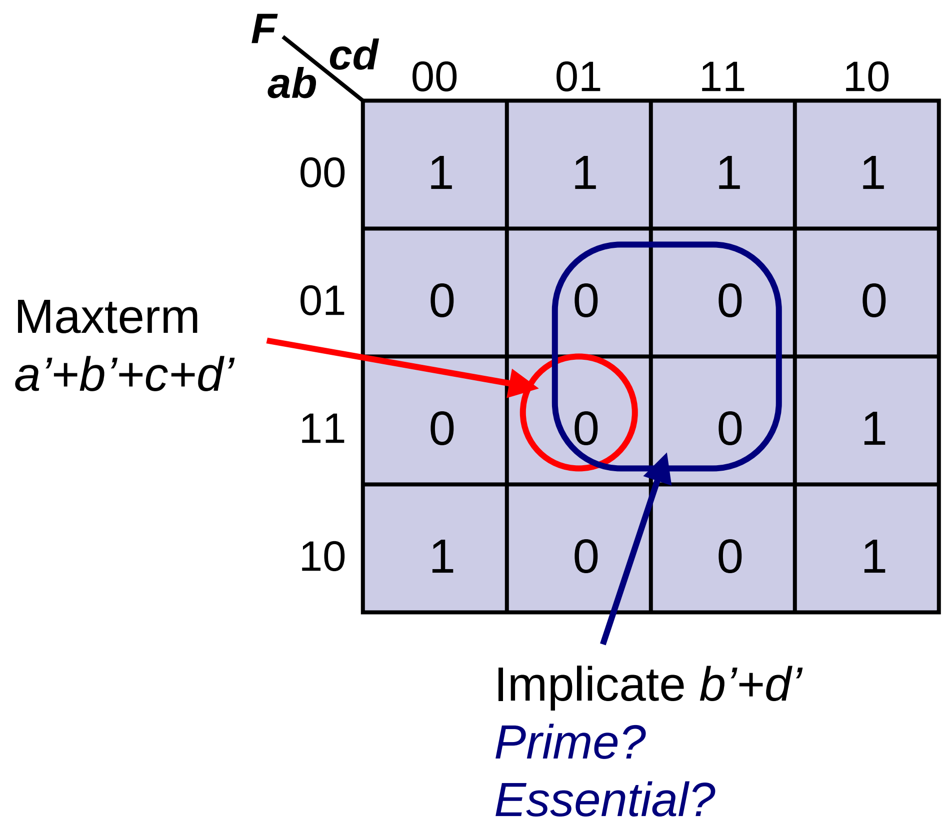


K-Map example

Exercises 2.1(d) in text book

a	b	c	d	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Each maxterm → 0-cell



2-level logic minimization

Sum-of-product form:

- Find all prime implicants (PIs)
- Find all essential prime implicants (EPIs)
- Select all EPIs
- Select a minimum number of PIs that cover the remaining minterms

Product-of-sum form:

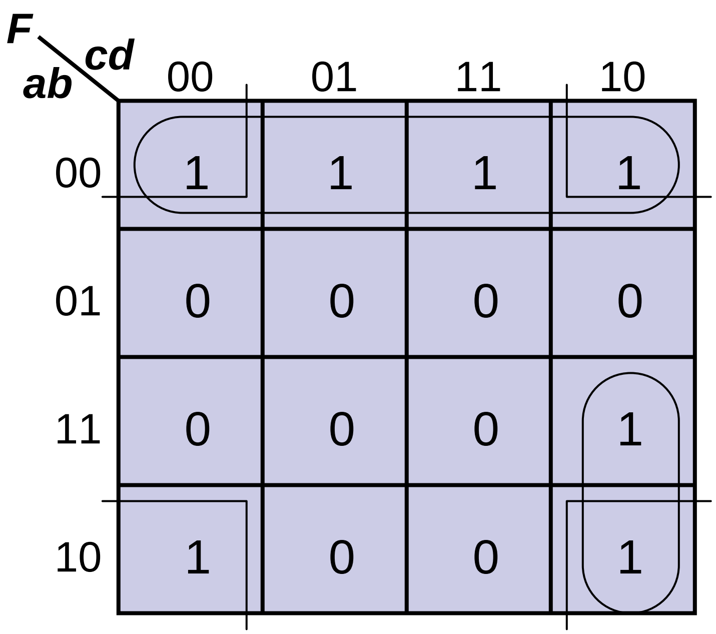
- Find implicates instead of implicants

K-Map example for SoP

PIs: a'b', b'd', acd'
EPIs: a'b', b'd', acd'

F = a'b' + b'd' + acd'

a	b	c	d	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

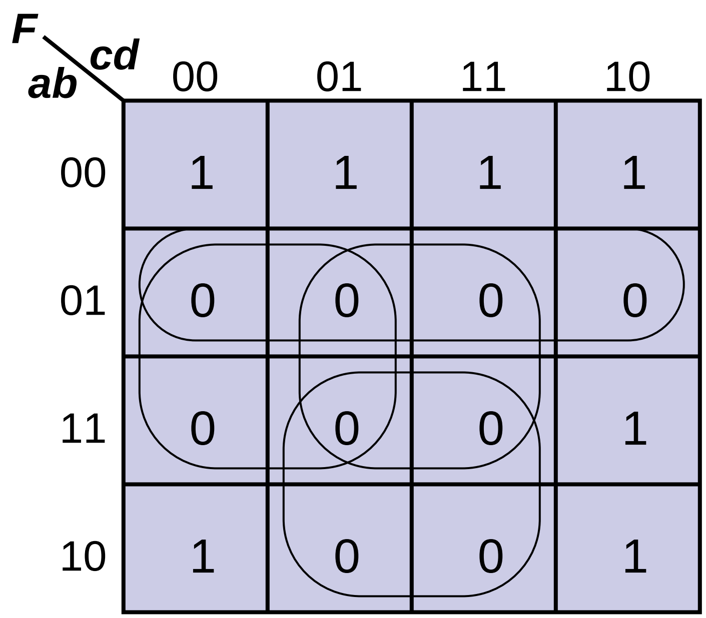


K-Map example for PoS

PIs: a+b’, b’+c, b’+d’, a’+d’
EPIs: a+b’, b’+c, a’+d’

F = (a+b’)(b’+c)(a’+d’)
= (a’b+bc’+ad)’

a	b	c	d	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0



2-level combinational circuit

- Sum-of-product form → 2-level AND-OR gates
- Product-of-sum form → 2-level OR-AND gates

F = a'b' + b'd' + acd'

Level1: two 2-AND, one 3-AND
Level2: one 3-OR

F = (a+b')(b'+c)(a'+d')

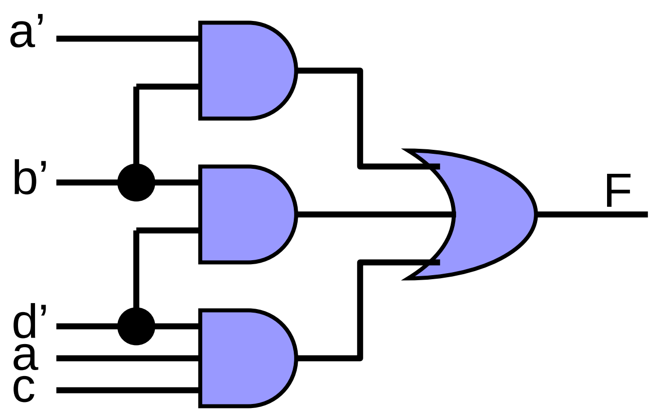
Level1: three 2-OR
Level2: one 3-AND

2-level combinational circuit

- Sum-of-product form → 2-level AND-OR gates
- Product-of-sum form → 2-level OR-AND gates

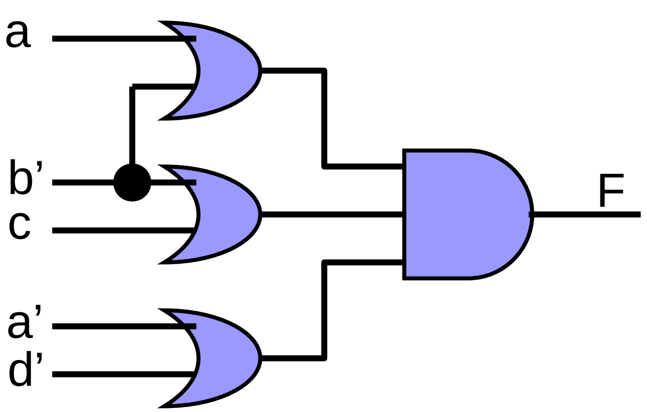
F = a'b' + b'd' + acd'

Level1: two 2-AND, one 3-AND
Level2: one 3-OR



F = (a+b')(b'+c)(a'+d')

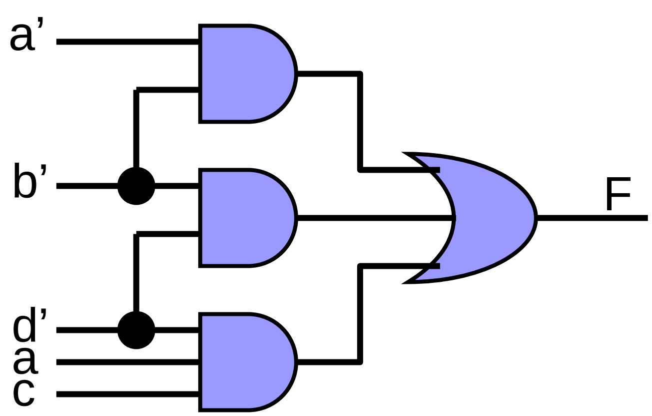
Level1: three 2-OR
Level2: one 3-AND



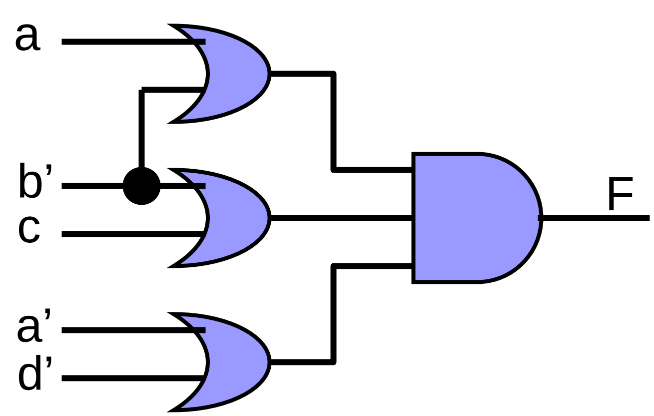
2-level logic transformation

- SoP form → 2-level AND-OR gates → 2-level NAND gates
- PoS form → 2-level OR-AND gates → 2-level NOR gates
- NAND and NOR: universal gates

F = a'b' + b'd' + acd'



F = (a+b')(b'+c)(a'+d')



2-level logic transformation

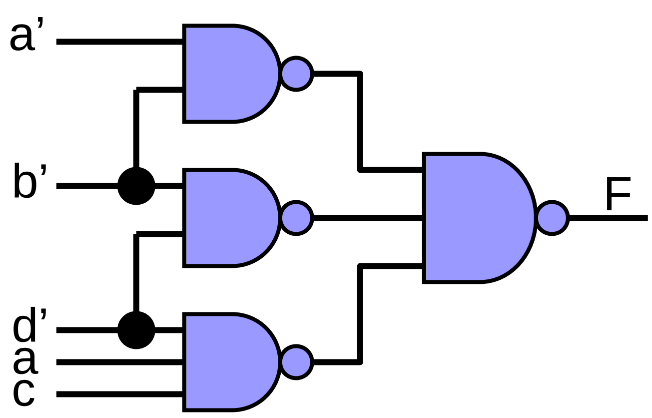
- SoP form → 2-level AND-OR gates → 2-level NAND gates
- PoS form → 2-level OR-AND gates → 2-level NOR gates
- NAND and NOR: universal gates

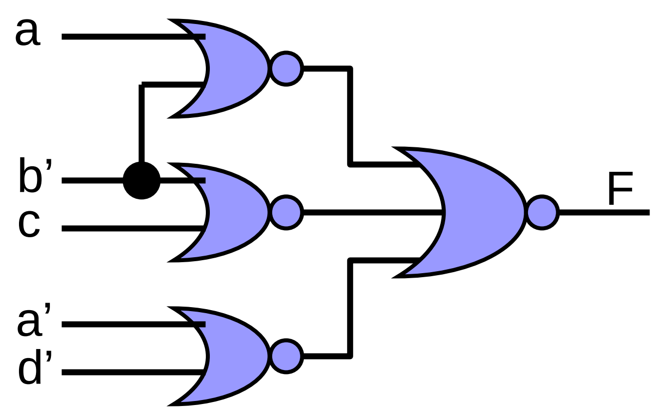
F = a'b' + b'd' + acd'

= ((a'b')' (b'd')' (acd')')'

F = (a+b')(b'+c)(a'+d')

= ((a+b')'+(b'+c)'+(a'+d')')'





Another example

F = abc + abd + abe + acd + ace + (a+d+e)’ + b’c’d + b’c’e + b’d’e’ + c’d’e’
= abc + abd + abe + acd + ace + a’d’e’ + b’c’d + b’c’e + b’d’e’ + c’d’e’

a = 0

F: bc \ de	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

a = 1

F: bc \ de	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

Another example

$$F = abc + abd + abe + acd + ace + (a+d+e)' + b'c'd + b'c'e + b'd'e' + c'd'e'$$
$$= abc + abd + abe + acd + ace + a'd'e' + \underline{b'c'd + b'c'e + b'd'e' + c'd'e'}$$

F (bc \ de)	00	01	11	10
00	1	1	1	1
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

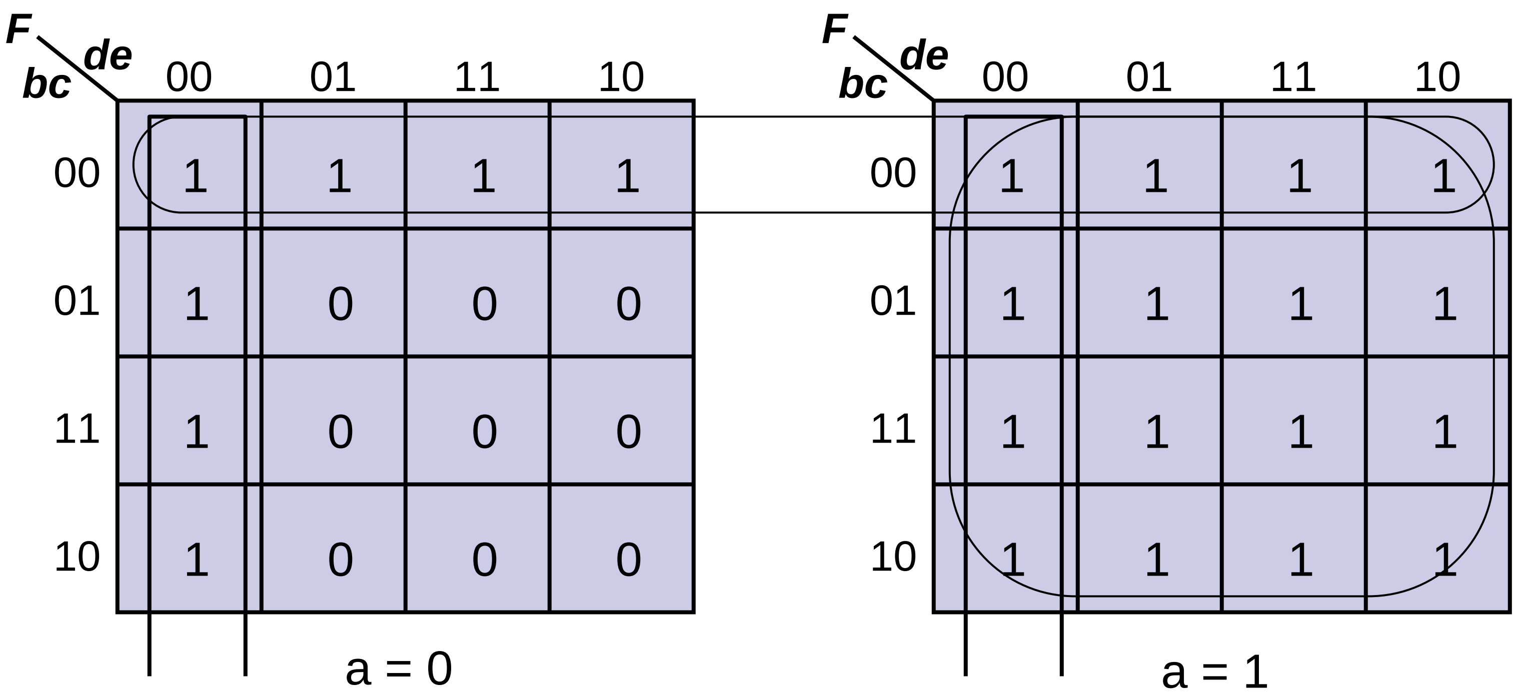
a = 0

F (bc \ de)	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

a = 1

Another example

F = abc + abd + abe + acd + ace + (a+d+e)’ + b’c’d + b’c’e + b’d’e’ + c’d’e’
  = abc + abd + abe + acd + ace + a’d’e’ + b’c’d + b’c’e + b’d’e’ + c’d’e’



F = a + d’e’ + b’c’

K-Map with don't cares

a	b	c	d	F	G
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	x	x
0	1	1	0	x	x
0	1	1	1	x	x
1	0	0	0	x	x
1	0	0	1	x	x
1	0	1	0	x	x
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

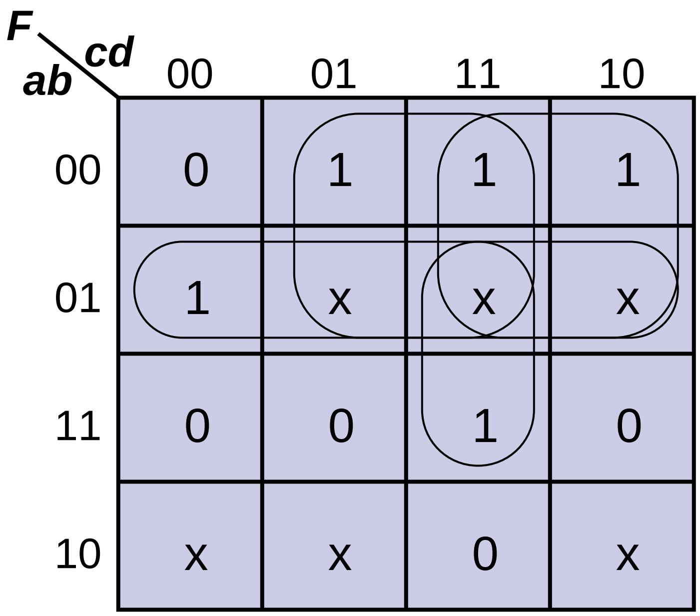
F ***ab*** \ ***cd***	00	01	11	10
00	0	1	1	1
01	1	x	x	x
11	0	0	1	0
10	x	x	0	x

K-Map with don’t cares

a	b	c	d	F	G
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	x	x
0	1	1	0	x	x
0	1	1	1	x	x
1	0	0	0	x	x
1	0	0	1	x	x
1	0	1	0	x	x
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

PIs: a’b, a’c, a’d, bcd
EPIs: a’b, a’c, a’d, bcd

F = a’b+ a’c + a’d+ bcd



Question: minimum PoS for G?