DIGITAL LOGIC(H)

Chapter 3 part1: Gate-Level Minimization

2024 Fall

Today’s Agenda

- Recap
- Context
  - Gate level minimization using the Map Method
  - Product of sums simplification
  - Don’t Care Conditions
  - Multiple output circuits
- Reading: Textbook, Chapter 3.1-3.5

Recap

- Lecture2
  - Boolean Functions
    - binary variables
      - 0 and 1
    - Operators
      - AND, OR, NOT
    - 3 representations
      - Truth table
        - unique
      - Algebraic expression
      - Logic diagram
  - Basic Axiom and Theorems
    - One variable
      - Identity, Null Element, Idempotency, Involution, Complements
    - Multiple variables
      - Commutativity, Associativity, Distributivity, Absorption, Combining, Simplification, Consensus, Demorgan's
        - xy = yx
        - (xy)z = x(yz)
        - x(y + z) = xy + xz
        - (x + y)(x + z) = x + yz
        - (x + y)' = x'y'
  - Canonical and Standard forms
    - Canonical
      - Sum of Minterm (Product term with all variables)
      - Product of Maxterm (Sum term with all variables)
      - Mi = mi'
    - Standard
      - Sum of products (can have less variables)
      - Product of sums (can have less variables)

Boolean function simplification

- A function's truth-table representation is unique, while its algebraic expression is not unique.
- Complexity of digital circuit (gate count) ∝ complexity of algebraic expression (literal count)
  - $F=x'y'z+x'yz+xy'$ (3 AND term, 8 literals)
  - $F=x'z+xy'$ (2 AND terms, 4 literals)
- The simplest algebraic expression is one that has minimum number of terms with the smallest possible number of literals in each term
- Methods for gate-level minimization:
  - **Algebraic method**(逻辑代数): Boolean algebra (Last lecture)
  - **Karnaugh map**(卡诺图): the map method (This lecture)

Karnaugh Map (K-map)

- An array of squares each representing one minterm to be minimized
- Each K-map defines a unique Boolean function
  - A Boolean function can be represented by a truth table, a Boolean expression, or a map
- K-map is a visual diagram of all possible ways a function may be expressed
- Used for manual minimization of Boolean functions

Merging Minterms

- In function F, m1 and m3 in the truth table differ only in one position

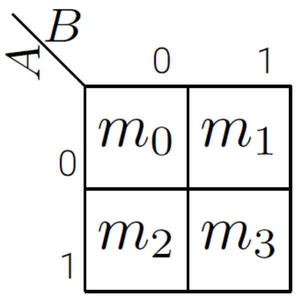
  001
  011 → 0?1

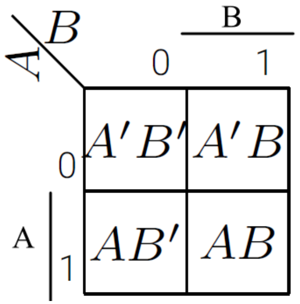
  - ? matches either 0 or 1
- The minterms in a function can be merged to form a simpler product term
  - 001 → x'y'z
  - 011 → x'yz
  - 0?1 → x'y'z+x'yz = x'z(y'+y) = x'z

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	0
1	1	1	0

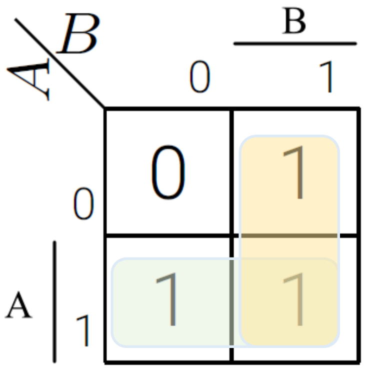
Two-Variable Map

- A two-variable map
  - is A truth table in square diagram
  - 4 minterms: A’B’, A’B, AB’, AB
  - row 0 stands for A’; row 1 stands for A
  - column 0 stands for B'; column 1 stands for B





	A	B	F
m_0	0	0	0
m_1	0	1	1
m_2	1	0	1
m_3	1	1	1

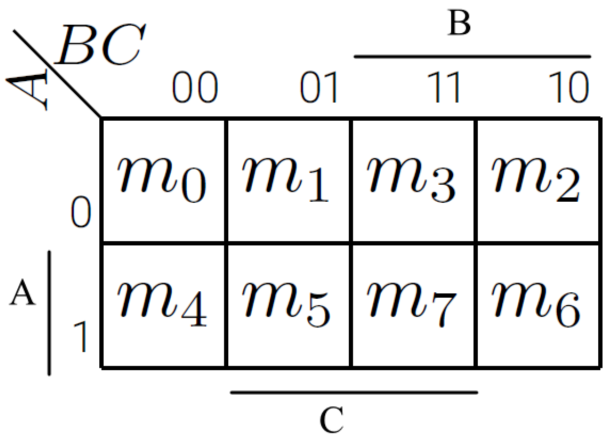


It’s ok for groups to overlap, if that makes them larger

$$m_1+m_2+m_3 = A'B+AB' +AB = A+B$$

Three-variable Map

- Minterms are arranged in the Gray-code sequence
- Any 2 (horizontally or vertically) adjacent squares differ by exactly 1 variable, which is complemented in one square and uncomplemented in the other.
- Any 2 minterms in adjacent squares that are ORed together will cause a removal of the different variable (adjacent applies not only the middle squares but also the boundary squares)

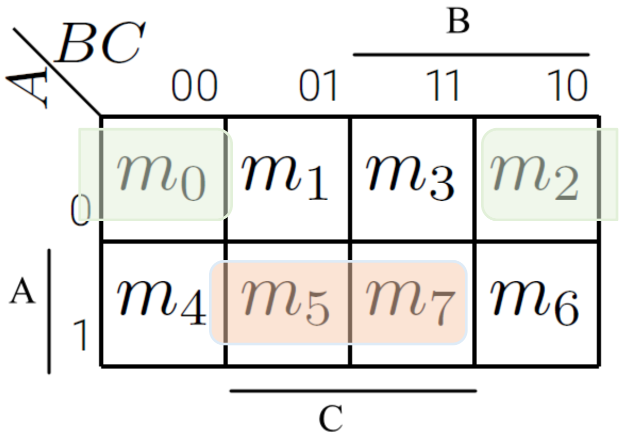


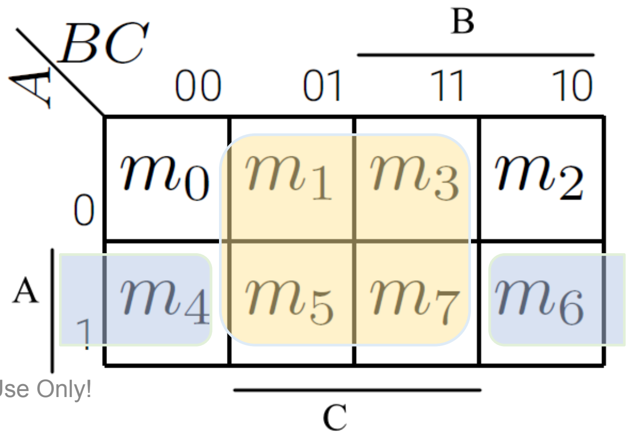
Three-variable Map

- Example (adjacent squares)
- m_5+m_7 = AB'C+ABC = AC(B+B') = AC
- m_0+m_2 = A'B'C'+A'BC' = A'C'(B+B') = A'C'
- m_4+m_6 = AB'C' + ABC' = AC' (B'+B) = AC'
- $m_1+m_3+m_5+m_7$

  = A'B'C+A'BC+AB'C+ABC=A'C(B+B')+AC(B+B')

  = A'C+AC=C





Example

- Simplify the following Boolean functions.

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

A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	0	0

Green circle: A'BC+A'BC' = A'B
Red circle: AB'C'+AB'C = AB'

F = A'BC+AB'C'+ABC'+ABC
= BC + AC'

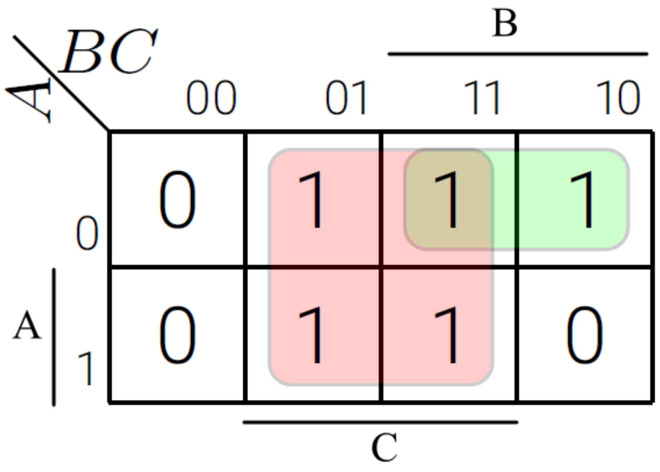
A \ BC	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Green circle: A'BC + ABC = BC
Red circle: AB'C' + ABC' = AC'

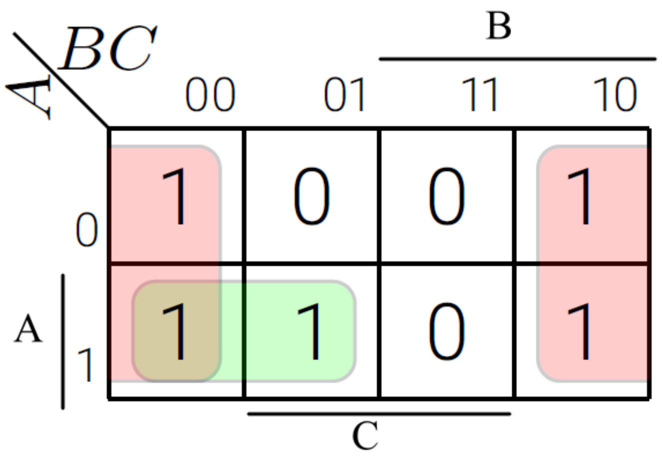
Example

- Simplify the following Boolean functions.

$F = \sum(1, 2, 3, 5, 7) = C + A'B$



$F = \sum(0, 2, 4, 5, 6) = C' + AB'$



It’s ok for groups to overlap,
if that makes them larger

Exercise

- Simplify the following Boolean function.
  - F = A'C + A'B + AB'C + BC

    = ?

$C + A'B$

A \ BC	00	01	11	10
0	0	1	1	1
1	0	1	1	0

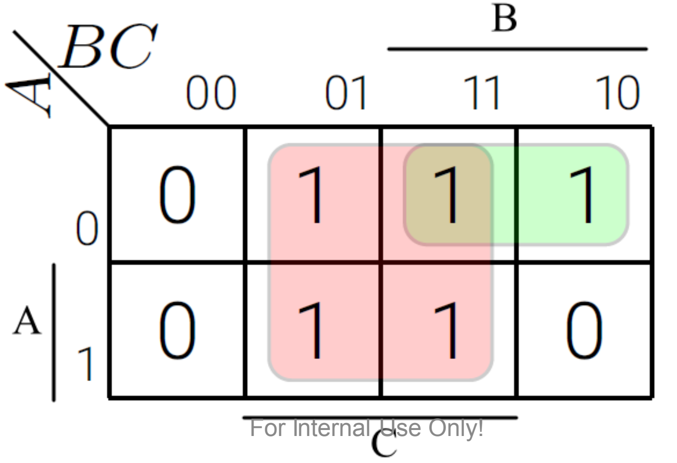
Exercise

- Simplify the following Boolean function.
  - F = A'C + A'B + AB'C + BC = ?
- Solution:
  - Express it in sum of minterms.
  - Find the minimal sum of products expression.
  - F = A'C + A'B + AB'C + BC

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

$$= A'BC + A'B'C + \cancel{A'BC} + A'BC' + AB'C + ABC + \cancel{A'BC}$$

$$= \sum(1, 2, 3, 5, 7) = C + A'B$$



Four-Variable Map

- The map
  - 16 minterms
  - Combinations of 2, 4, 8, and 16 adjacent squares

$AB \backslash CD$	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Implicants

- Implicant: any product term that implies the function
  - A product term that makes a function to be true

	minterm	implicant	
m_1	√	√	1-minterm
m_2	√	X	0-minterm
x'z	X	√	1 product term

x	y	z	F	
0	0	0	0	m_0
0	0	1	1	m_1
0	1	0	0	m_2
0	1	1	1	m_3
1	0	0	1	m_4
1	0	1	1	m_5
1	1	0	0	m_6
1	1	1	0	m_7

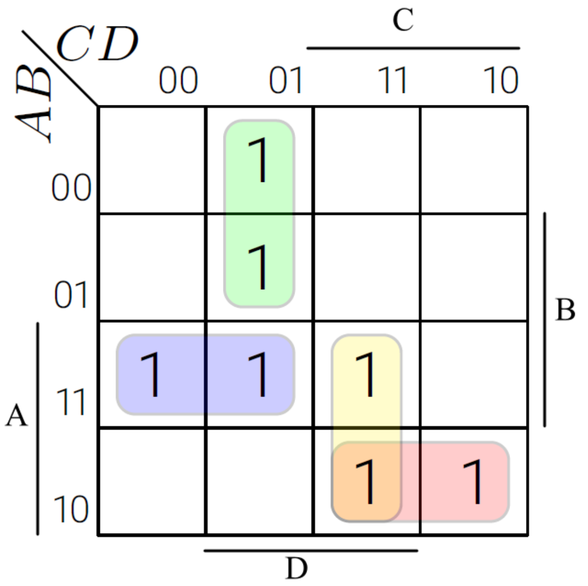
- Prime implicant (PI) (质蕴含)
  - A 1-product term obtained by combining the maximum possible number of adjacent squares in the map.
- Essential prime implicant (EPI) (基本质蕴含)
  - If a minterm in a square is covered by only one prime implicant

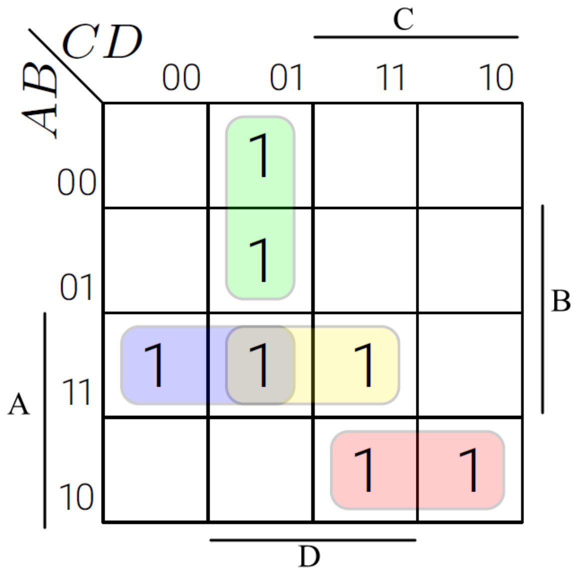
Tips for simplification

- Simplification Steps:
  - Determine all essential prime implicants.
  - Find other prime implicants that cover remaining minterms.
  - Logical sum all prime implicants.
- Tips:
  - Minimize the number of groups
  - Maximize the group size
  - It’s ok for groups to overlap, if that makes them larger

Example

- Simplify the following Boolean functions
- F = $m_1+m_5+m_{10}+m_{11}+m_{12}+m_{13}+m_{15}$
    - start with EPIs
    - Green circle: A′B′C′D + A′BC′D = A′C′D
    - Purple circle: ABC′D′ + ABC′D = ABC′
    - Red circle: …
    - …
  - F = A′C′D + ABC′ + ACD + AB′C
- This reduced expression is not a unique one
  - If pairs are formed in different ways, the simplified expression will be different.
  - F = A′C′D + ABC′ + ABD + AB′C

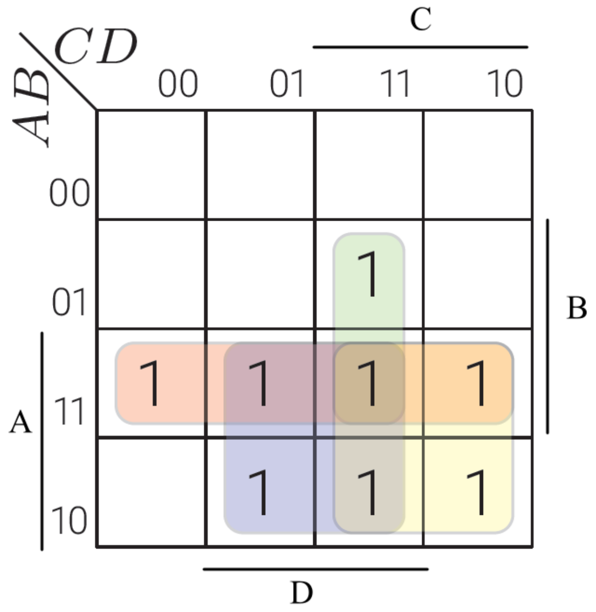




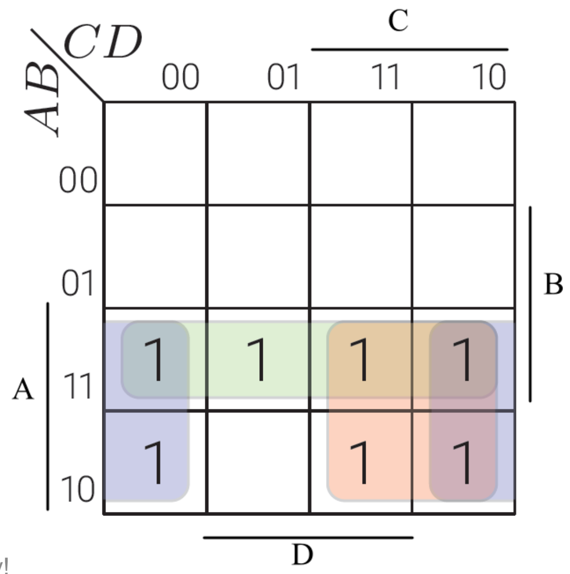
Example

- Simplify the following Boolean functions.

$F = \sum(7, 9, 10, 11, 12, 13, 14, 15)$
$= AB + AC + AD + BCD$



$F(A,B,C,D) = ABCD + AB'C'D' + AB'C + AB$
$= ABCD + AB'C'D' + AB'C(D + D')$
$+AB(C + C')(D + D')$
$= \ldots$
$= \sum(8, 10, 11, 12, 13, 14, 15)$
$= AB + AC + AD'$



Exercise

- Simplify the following Boolean functions.

F(A,B,C,D)
$= \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$
= ?

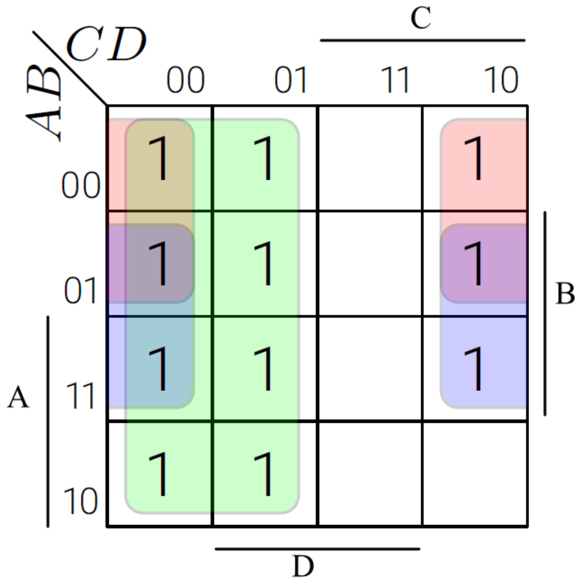
F(A,B,C,D)
$= A'B'C' + B'CD' + A'BCD' + AB'C'$
= ?

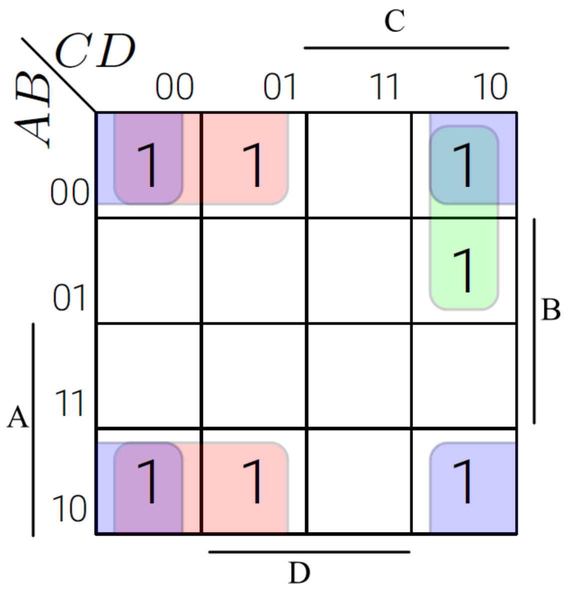
Exercise

- Simplify the following Boolean functions.

$F(A,B,C,D)$
$= \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$
$= C' + A'D' + BD'$

$F = A'B'C' + B'CD' + A'BCD' + AB'C'$
$= A'B'C'(D + D') + B'CD'(A + A') +$
$A'BCD' + AB'C'(D + D')$
$= A'B'C'D + A'B'C'D' + AB'CD' + A'B'CD' +$
$A'BCD' + AB'C'D + AB'C'D'$
$= \sum(0, 1, 2, 6, 8, 9, 10)$
$= B'C' + B'D' + A'CD'$





K-map Summary

- Any 2^k adjacent squares, k=0,1,...,n, in an n-variable map represent an area that gives a product term of n-k literals

K	# of adjacent squares	# of literals left in a term in an n-variable map		
		n=2	n=3	n=4
0	1	2	3	4
1	2	1	2	3
2	4	0	1	2
3	8		0	1
4	16			0

- Five-Variable Map
  - Map for more than four variables becomes complicated
  - Five-variable map: two four-variable map (one on the top of the other), contains 2^5 or 32 cells.

Product of Sums Simplification

- Previous Examples are Sum of Product Simplification
  - E.g. F = AB + A'D + AB'C (Product of sum form)
- How to find Product of Sum simplification
  - E.g. F = (A+B)(B+C') (Sum of Product form)
- POS simplification Steps
  - Simplified F' in the form of sum of products
    - Group adjacent 0-minterms squares together
  - Apply DeMorgan's theorem F = (F')'
  - F': sum of products → F: product of sums

Example

- Simplify the Boolean function into product of sums form:
  - F(A,B,C,D) = ∑(2, 3, 7, 10, 11, 15)
- Solution

因为零好圈

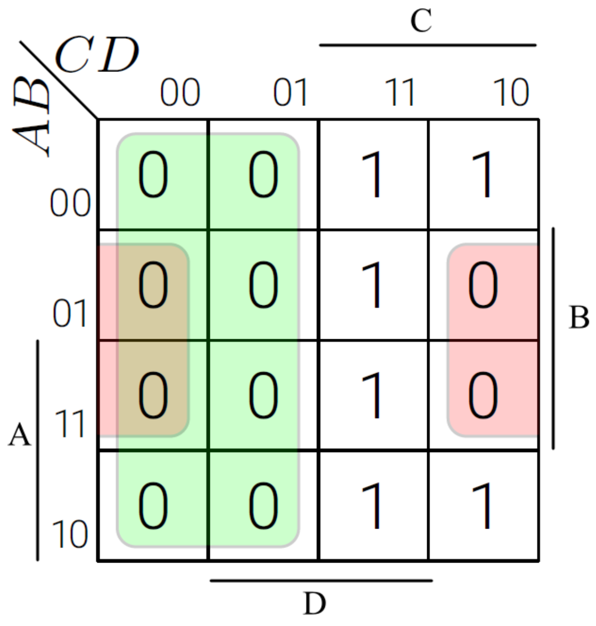
  - Step1: group the 0-minterms to find F complement

    F' = ∑(0, 1, 4, 5, 6, 8, 9, 12, 13, 14)

    = C' + BD'   (Group 0 minterms)
  - Step2: find the complement again to get original F

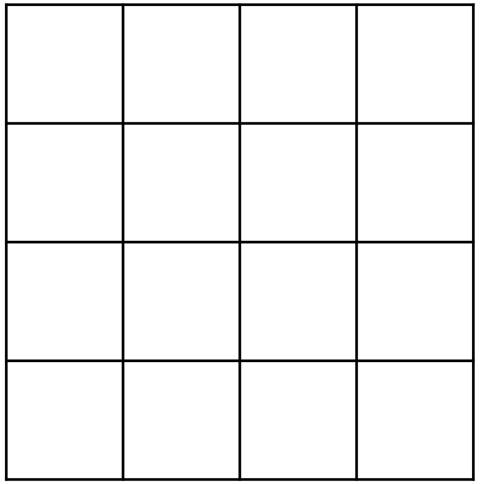
    F = (F')' = (C' + BD')'

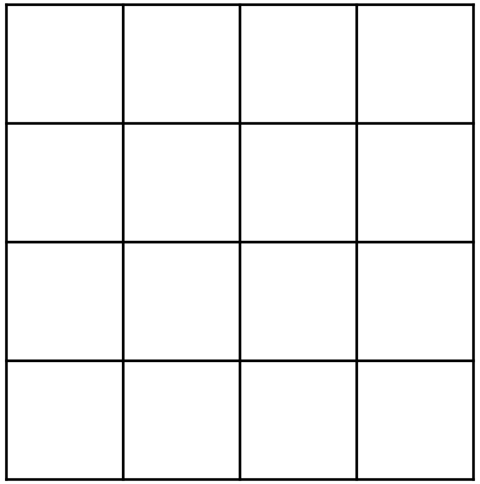
    = C(B'+D)   (DeMorgan's)



Exercise

- simplify $F(A, B, C, D) = \sum(0, 1, 2, 5, 8, 9, 10)$ into
  - sum-of-products form
    - F = ?
  - product-of-sums form
    - F = ?





Exercise

- simplify F(A, B, C, D) = ∑(0, 1, 2, 5, 8, 9, 10) into
  - sum-of-products form
    - F = B'D' + B'C' + A'C'D (Group 1-minterms)
  - product-of-sums form
    - F' = AB+CD+BD' (Group 0-minterms)
    - F = (A'+B')(C'+D')(B'+D) (DeMorgan's)

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

C B A D

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

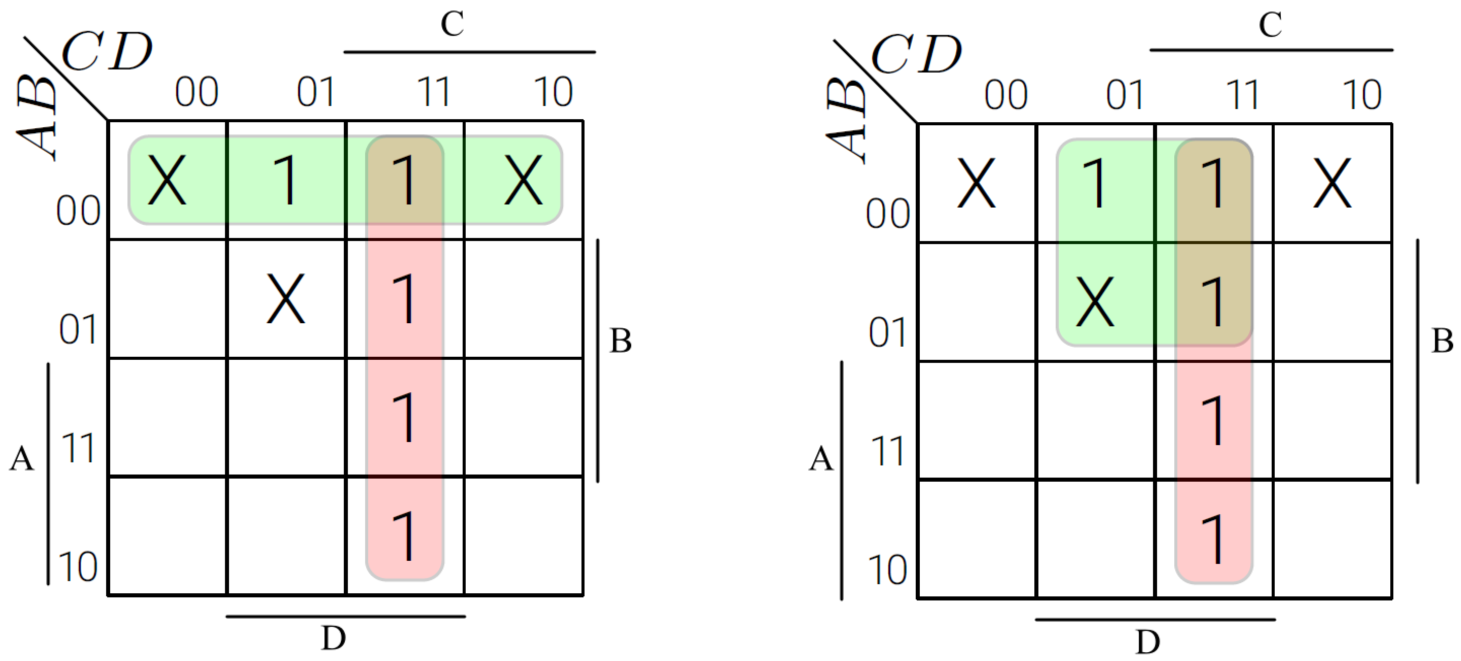
C B A D

Don’t care conditions

- Incompletely specified functions
  - Functions that have unspecified outputs for some input combinations
  - E.g. output are unspecified for 1010 to 1111 in 4-bit BCD code
- Don’t-care conditions
  - Unspecified minterms of a function, don’t-cares, Xs
  - Can be used on a map to provide further simplifications of the Boolean expression
  - Each X can be assigned an arbitrary value, 0 or 1, to help simplification procedure

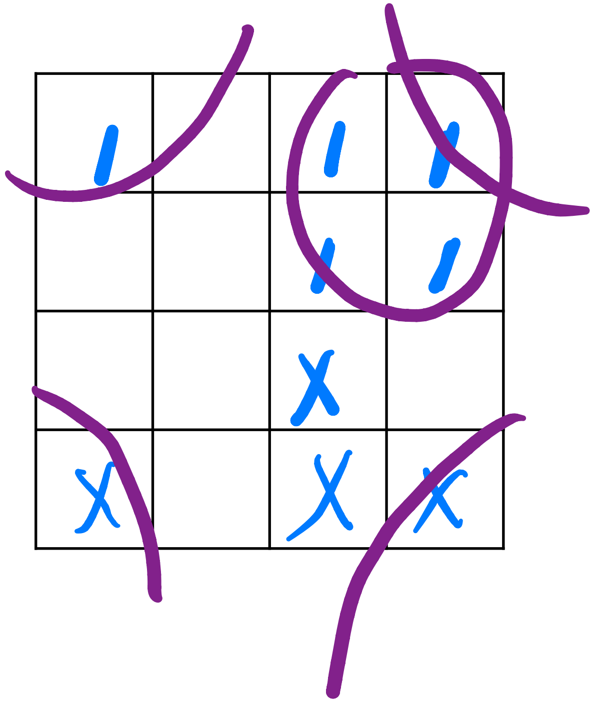
Example

- Simplify F(A, B, C, D) = ∑(1, 3, 7, 11, 15) with don't-care conditions d(A, B, C, D) = ∑(0, 2, 5).
  - F = A'B' + CD
  - or F = A'D + CD
  - Just make sure all 1 minterms are circled, thus both simplifications are acceptable



Exercise

- Using the Karnaugh map method obtain the minimal sum of the products expression for the function F(A,B,C,D) = Σ(0, 2, 3, 6, 7) + d(8, 10, 11, 15)



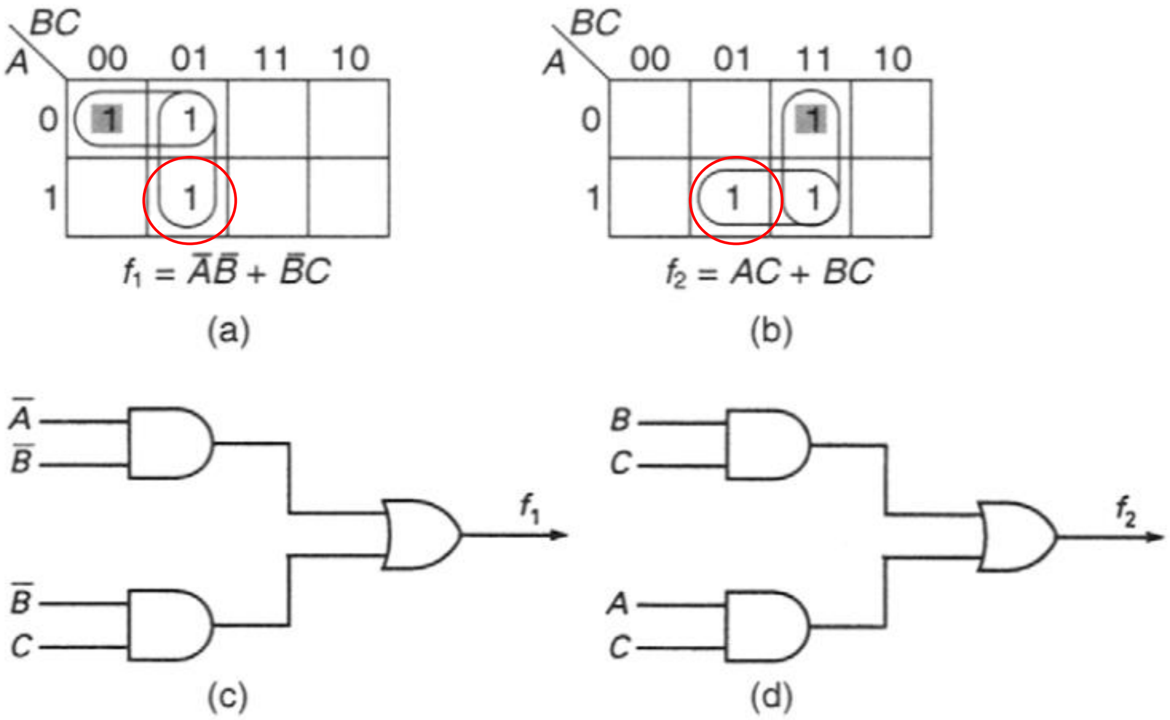
Exercise

- Using the Karnaugh map method obtain the minimal sum of the products expression for the function F(A,B,C,D) = Σ(0, 2, 3, 6, 7) + d(8, 10, 11, 15)
- F = A’C + B’D’

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

Multiple output circuits

- A circuit with two outputs
  - f1=A'B'+B'C
  - f2=AC+BC
  - 6 AND/OR gates in total
- We can optimize the circuit by sharing
  - Find the common term (which might not be the simplest but can be shared
  - e.g. AB'C (observe the common squares)



Multiple output circuits

- AB'C is the common minterm
- Thus The circuit can be described as
  - f1=A'B'+AB'C
  - f2=BC+AB'C
  - 5 AND/OR gates (less gates)