Digital Logic Design

Lecture 4:

Chapter 3: Gate Level Minimization (K-Maps)

Mirvat Al-Qutt, Ph.D
Computer Systems Department , FCIS,
Ain Shams University

Lecture 3:

Chapter 2: Boolean Algebra and Logic Gates

Basic Definitions

- **Duality Principle (DeMorgan's Theorem)**
- Verify DeMorgan's Theorem

$$(x+y)' = x'y' \qquad x+y = (x'y')'$$

$$(x\ y)' = x'+y' \qquad x\ y = (x'+y')'$$

x	y	x'	y'	$x+y$	$(x+y)'$	$x'y'$	Xy	$x'+y'$	$(xy)'$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

Basic Definitions

- **Consensus Theorem**

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

$(x+y) \cdot (x'+z) \cdot (y+z) = (x+y) \cdot (x'+z)$

Proof:

$xy + x'z + yz$

$= xy + x'z + 1.yz$

$= xy + x'z + (x+x')yz$

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

$= (xy + xyz) + (x'z + x'zy)$

$= xy\,(1+z) + x'z\,(1+ y)$

$= xy + x'z$

Proof:

$(x+y)\bullet(x'+z)\bullet(y+z)$

$= (x+y)\bullet(x'+z)\bullet(0+y+z)$

$= (x+y)\bullet(x'+z)\bullet((xx')+y+z)$

$= (x+y)\bullet(x'+z)\bullet(x+y+z)\bullet(x'+y+z)$

$= ((x+y)+(0\bullet z))((x'+z)+(0\bullet y))$

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

Minterms and Maxterms

Challenge

- Convert from any form to the other

Table 2.4
Functions of Three Variables

x	*y*	*z*	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

F1= $\sum(m_1, m_4, m_7)$

F1=x'y'z + xy'z'+ xyz

F1=(x+y+z) (x+y'+z)
(x+y'+z')(x'+y+z') (x'+y'+z)

F1= $\prod(M_0 M_2 M_3 M_5 M_6)$

Sum of Minterms

- Sum of minterms: there are 2^n minterms and 2^{2n} combinations of functions with *n* Boolean variables.
- Example : express $F = A+B'C$ as a sum of **minterms**.
  - $F = A+B'C$
    $= A\,(B+B') + B'C$
    $= AB + AB' + B'C$
    $= AB(C+C') + AB'(C+C') + (A+A')B'C$
    $= ABC+ABC'+AB'C+AB'C'+A'B'C$
    $= A'B'C + AB'C' + AB'C + ABC' + ABC$
    $= m_1 + m_4 + m_5 + m_6 + m_7$
    $= \Sigma(1, 4, 5, 6, 7)$

1

2

or, built the truth table first

Table 2.5
Truth Table for $F = A + B'C$

A	*B*	*C*	*F*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Sum of Minterms

- Sum of minterms: there are 2^n minterms and 2^{2n} combinations of functions with *n* Boolean variables.
- Example: express $F = A+B'C$ as a product of **maxterms**

2

- $F = A+B'C$

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

$= (A+B'+CC')(A+C+BB')$

1

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

$= \prod(M_0 M_2 M_3)$

or, built the truth table first

Table 2.5
Truth Table for F = A + B′C

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Product of Maxterms

- Product of maxterms: using distributive law to expand.

- Example : express $F = xy + x'z$ as a product of **maxterms**.
  - $F = xy + x'z$

$$= (xy + x')(xy + z)$$

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

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

1

$$= (x'+y+zz')(x+z+yy')(y+z+xx')$$

$$=(x'+y+z)(x'+y+z')(x+z+y)\ (x+z+y')(y+\cancel{z+}x)$$

$$(y+\cancel{z+}x')$$

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

$$= M_0M_2M_4M_5$$

$$= \Pi(0, 2, 4, 5)$$

2

or, built the truth table first

Table 2.6
Truth Table for $F = xy + x'z$

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	1
1	1	1	1

Complement of a Function Expressed in Canonical Forms

- The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function.
  - $F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$
  - $F(A, B, C) = = \Pi(0, 2, 3)$

Thus,

- $F'(A, B, C) = \Sigma(0, 2, 3)$
- $F'(A, B, C) = \Pi(1, 4, 5, 6, 7)$

- By DeMorgan's theorem $m_j' = M_j$

x	y	z	F1	F1'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Conversion between Canonical Forms

- Example
  - $F = xy + x'z$
  - $F(x, y, z) = \Sigma(1, 3, 6, 7)$
  - $F(x, y, z) = \Pi\ (0, 2, 4, 6)$

  - **<u>Complement ????</u>**
  - $F'(x, y, z) = \Sigma(0, 2, 4, 6)$
  - $F'(x, y, z) = \Pi\ (1, 3, 6, 7)$

Truth Table for $F = xy + x'z$

x	y	z	F	F'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

Canonical Forms vs. Standard Forms

Canonical Forms

- Each minterm or maxterm must contain **all the variables** either complemented or uncomplemented,
- Sum of minterms (Product terms)
- OR Product of Maxterms (sum terms)

Standard forms

- the terms that form the function may obtain **one, two, or any number** of literals, .
- There are two types of standard forms:
  - Sum of products:

    $F_1 = y' + xy + x'yz'$
  - Product of sums:

    $F_2 = x(y'+z)(x'+y+z')$

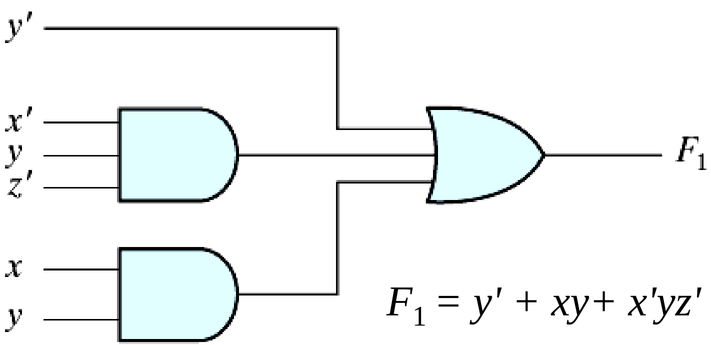
Standard Forms

- A Boolean function may be expressed in a **nonstandard** form
  - $F_3 = AB + C(B + A)$
- But it can be changed to a **standard** form by using The distributive law
  - $F_3 = AB + C(B + A) = AB + BC + AC$

- And it can be changed to a **canonical** form by using The distributive law after adding missing literal
  - $F_3 = AB + BC + AC = AB(C+C') + BC(A+A') + AC(B+B')$
  - $=ABC+ABC'+ABC+A'BC+ABC+AB'C$
  - $=ABC+ABC'+A'BC+AB'C$

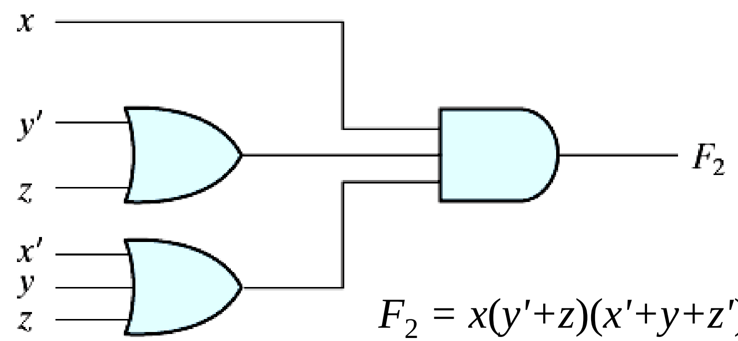
Implementation

- Two-level implementation



$F_1 = y' + xy + x'yz'$

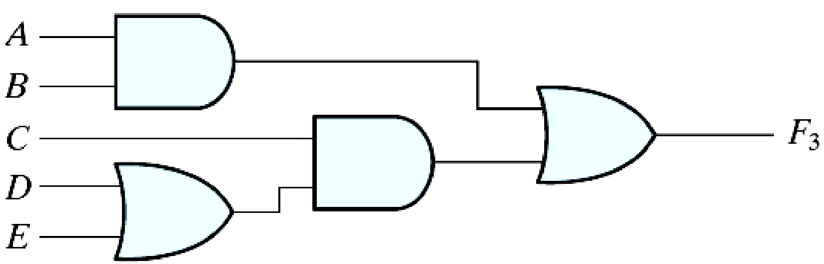
(a) Sum of Products



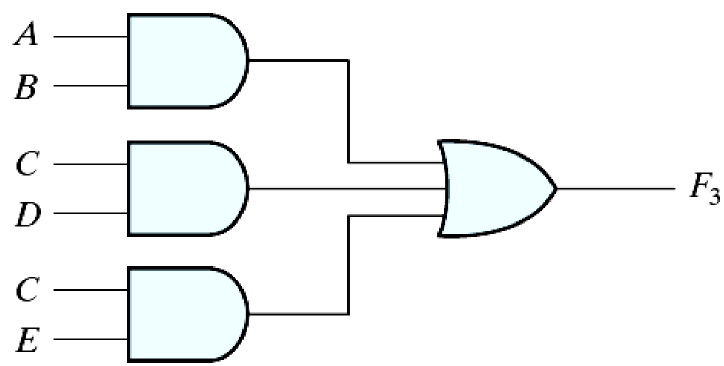
$F_2 = x(y'+z)(x'+y+z')$

(b) Product of Sums

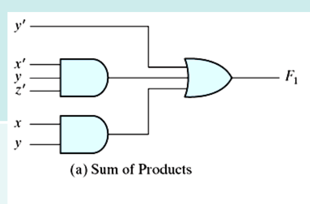
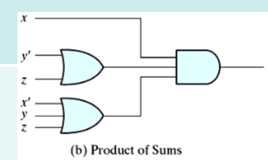
- Multi-level implementation



(a) $AB + C(D + E)$



(b) $AB + CD + CE$

SOP	POS
Sum of minterms $$F = \sum(m_0, m_2, \ldots .\, m_i)$$ Sum of terms that function gives 1	Product of Maxterms $$F = \prod(M_0 M_1 \ldots \ldots .\, M_i)$$ Product of terms that function gives 0
Minterms (Locate 1's) m_0 = x'y'z' = 000 m_1 = x'y'z = 001 …. m_7 = xyz = 111	Maxterms (Locate 0's) M_0 = x+y+z = 000 M_1 = x+y+z' = 001 …. M_7 =x'+y+'z' = 111
Convert Boolean function to SOP By multiplying each term by the missing variable Ored with its complement F = xy = xy(z+z') = xyz +xyz'	Convert Boolean function to POS By expanding using distributive law and then for each term add the missing variable ANDed with its complement F= x+y = x+y+zz' = (x+y+z)(x+y+z')
 (a) Sum of Products Logic Diagram: • 2 level implantation • Level of AND gates followed by one OR gate	 (b) Product of Sums Logic Diagram: • 2 level implantation • Level of OR gates followed by one AND gate

Other Logic Operations

- 2^n rows in the truth table of n binary variables.
- 2^{2^n} functions for n binary variables.
- 16 functions of two binary variables.

Table 2.7
Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- All the new symbols except for the exclusive-OR symbol are not in common use by digital designers.

Boolean Expressions

Table 2.8
Boolean Expressions for the 16 Functions of Two Variables

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Outline of Chapter 3

- 3.1 Introduction
- 3.2 The Map Method
- 3.3 Four-Variable Map
- 3.4 Five-Variable Map
- 3.5 Product-of-Sums Simplification
- 3.6 Don't-Care Conditions

Gate-level minimization

- Gate-level minimization refers to the design task of finding an ***optimal*** gate-level implementation of Boolean functions describing a digital circuit.

- Different representation of Boolean Function
  - Boolean Expression (Many)
  - Truth Table (Unique)
  - Logic Gates Diagram (Many)

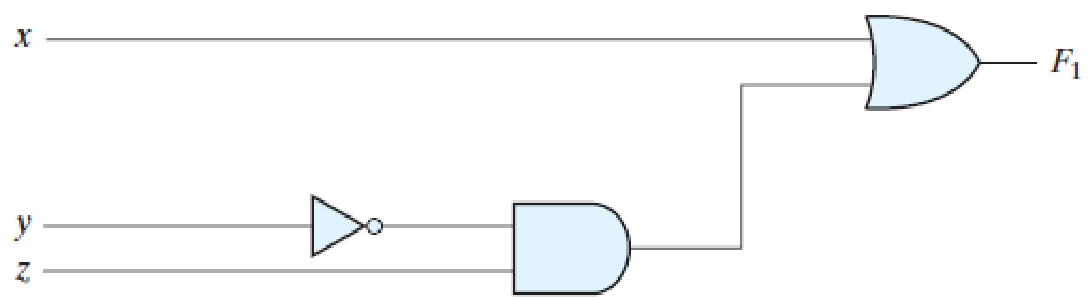


FIGURE 2.1
Gate implementation of $F_1 = x + y'z$

x	y	z	F_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The Map Method

- The complexity of the digital logic gates is directly related to the complexity of the algebraic expression

- **Logic minimization**

Algebraic approaches	The Karnaugh map
• lack specific rules • The simplified expression may not be unique	• A simple straight-forward procedure • A pictorial form of a truth table

The Map Method

x	*y*	*z*	*F1*	*F2*	*F3*
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	1

F1 = x **F2 = xy** **F3 = y**

K- Map :A pictorial form of a truth table

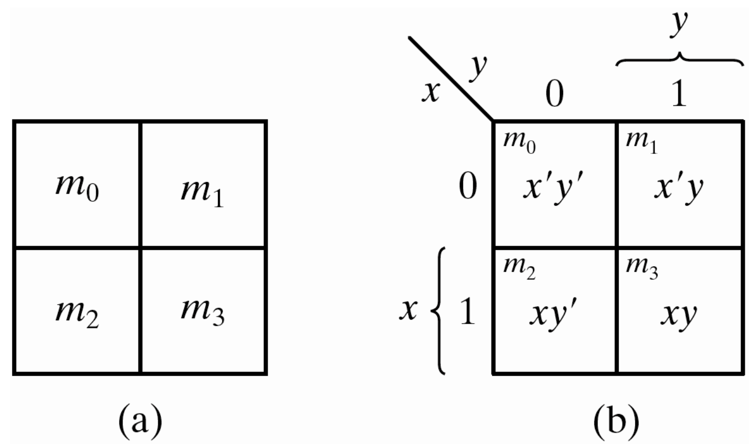
The Map Method

x	y	$F1$	$F2$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

K- Map :A pictorial form of a truth table

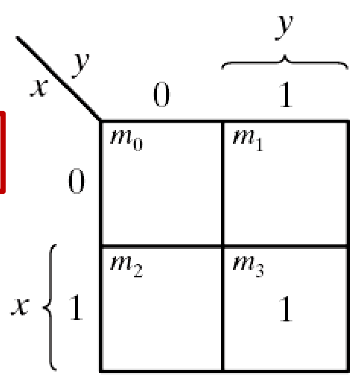
Two-Variable Map

- A two-variable map
  - Four minterms
  - *x*' = row 0; *x* = row 1
  - *y*' = column 0; *y* = column 1
  - A truth table in square diagram

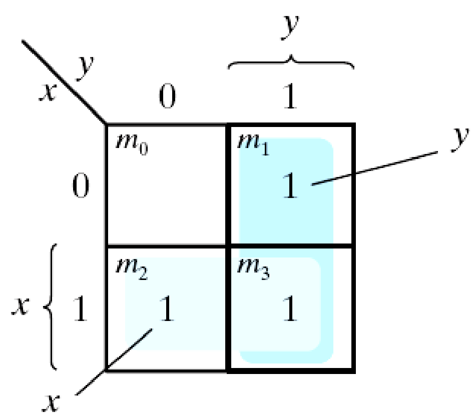


Two-variable Map

$m_3 = xy$



(a) *xy*



(b) *x* + *y*

$=m_1+m_2+m_3$
$=x'y+xy'+xy$
$=y(x'+x)+xy'$
$=y+xy'$
$=(y+x)(y+y')$
$=x+y$

A Three-variable Map

- A three-variable map: Eight minterms
  - The Gray code sequence
  - Any two adjacent squares in the map differ by only on variable
    - Primed in one square and unprimed in the other
    - $m_1 + m_3 = x'y'z + x'yz = x'z\,(y'+y) = x'z$

m_0	$\boxed{m_1}$	$\boxed{m_3}$	m_2
m_4	m_5	m_7	m_6

			y	y
$x \backslash xz$	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
x 1	$xy'z'$	$xy'z$	xyz	xyz'
		z	z	

A Three-variable Map

- A three-variable map: Eight minterms
  - The Gray code sequence
  - Any 4 adjacent squares in the map differ by two variable (have only 1 variable in common)
    - Primed in one square and unprimed in the other
    - $m_1 + m_3 + m_5 + m_6 = x'y'z + x'yz + xy'z + xyz = z(x'y'+x'y+xy'+xy)$
    - $= z(x'(y'+y)+x(y'+y))$
    - $= z(x'+x) = z$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

$x \backslash xz$	00	01	11 (y)	10 (y)
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
x 1	$xy'z'$	$xy'z$	xyz	xyz'
		z	z	

A Three-variable Map

- *Simplification of Adjacent Squares :*
  - $m_0 + m_2 = x'y'z' + x'yz' = x'z'(y'+y) = x'z'$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

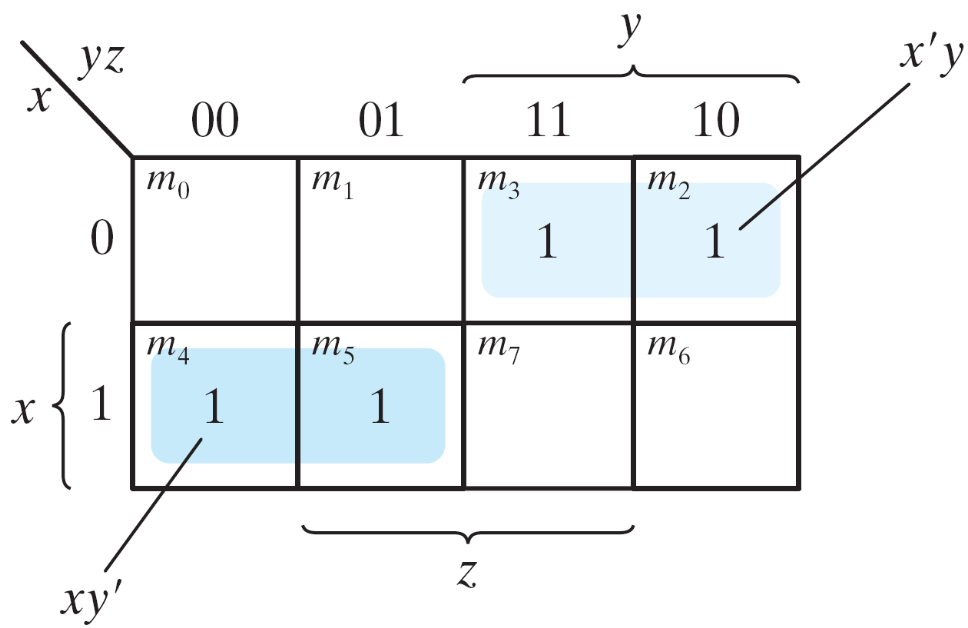
$x \backslash yz$	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

  - $m_4 + m_6 = xy'z' + xyz' = xz'(y'+y) = xz$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

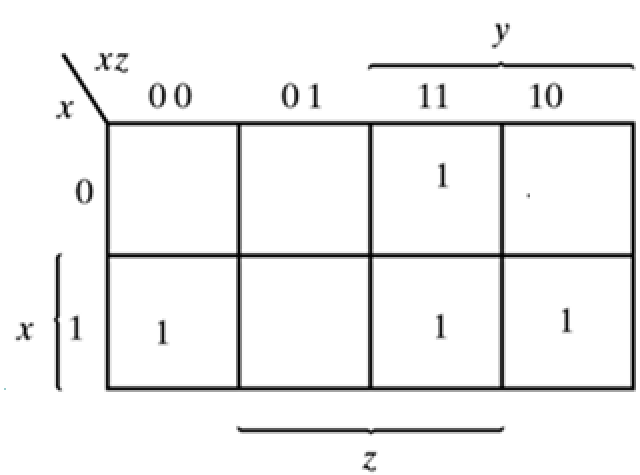
A Three-variable Map

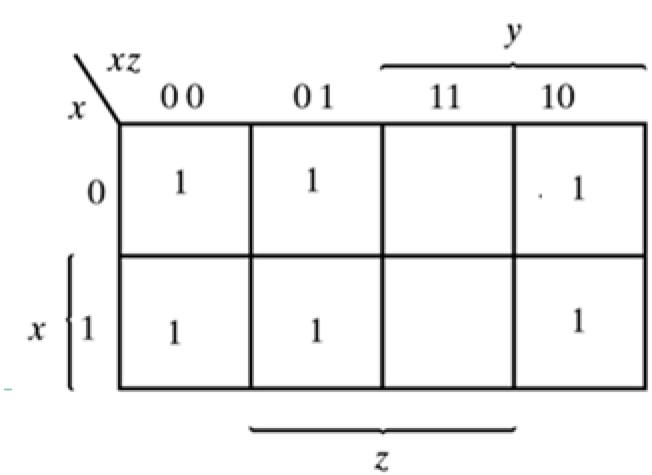
- Example: simplify the Boolean function F(x, y, z) = Σ(2, 3, 4, 5)
  - $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$



K-Map Rules

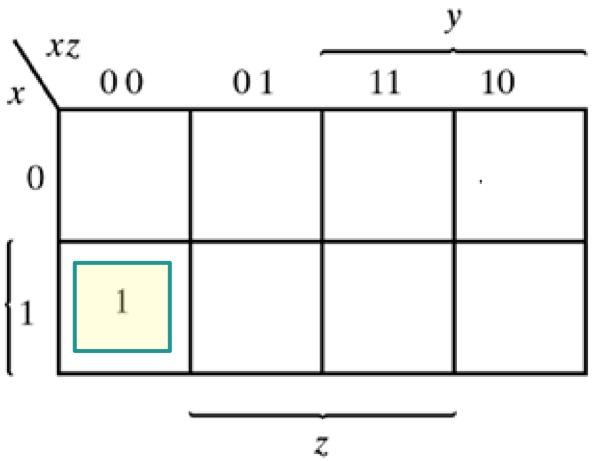
1. Group (2^n) adjacent 1's (2,4,8,16 ,…..)
2. Group possible maximum (2^n) adjacent 1's
3. Group overlapping is allowed as long as there are some 1's are not covered yet
4. Must cover all 1's, each one must be covered at least once, trying possible minimum number of coverage
5. Stop when all one's are covered at least once.
6. Each group expression is based on the shared area label.
7. Group adjacent (2^n) 1's on the same row, column , consider folded Maps



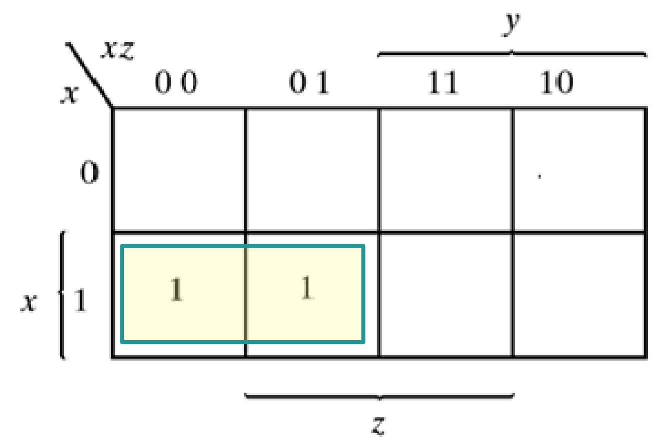


A Three-variable Map

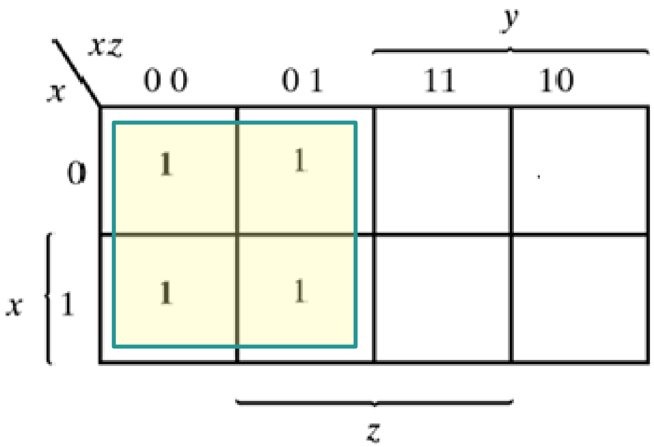
- Number of squares and Number of Variables



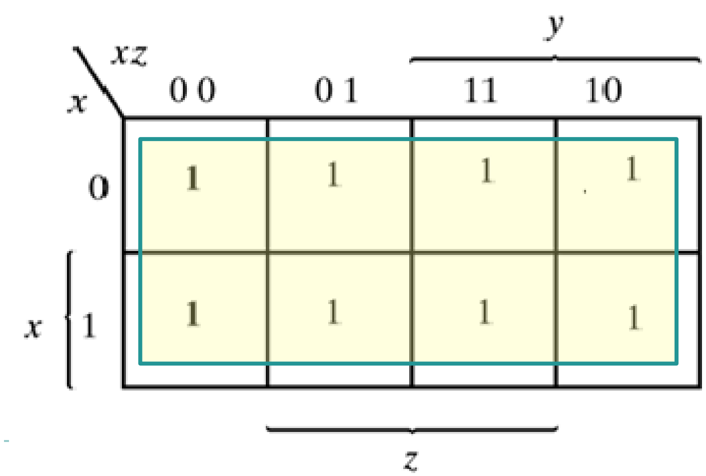
F=xy'z'



F=xy'



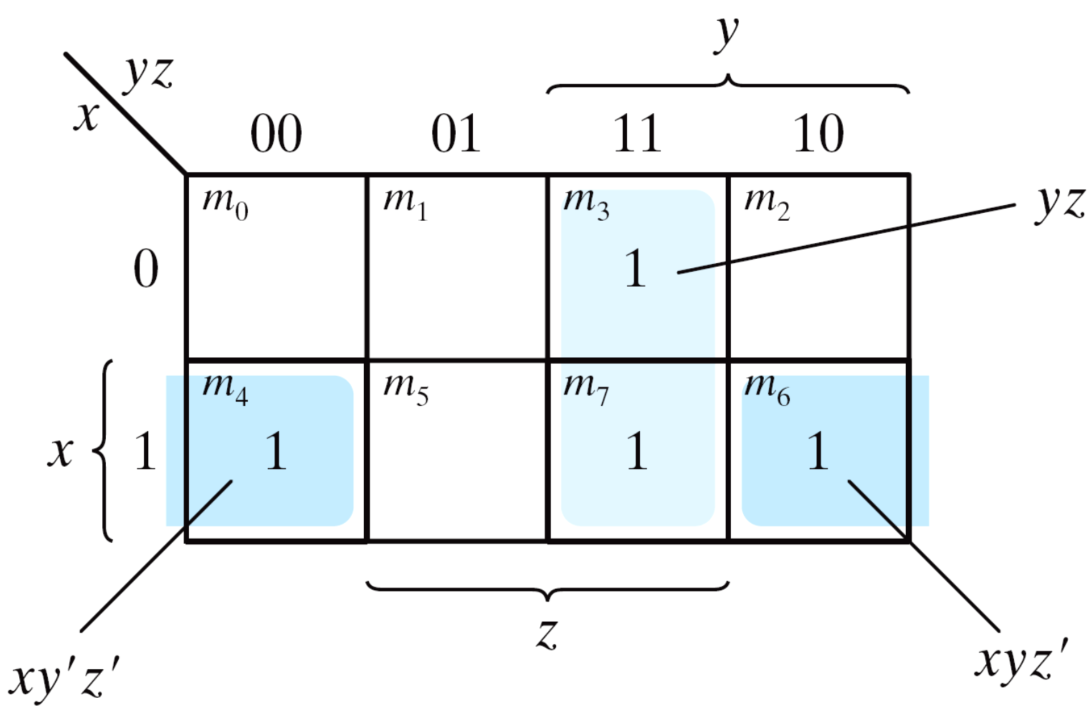
F= y'



F= 1

A Three-variable Map

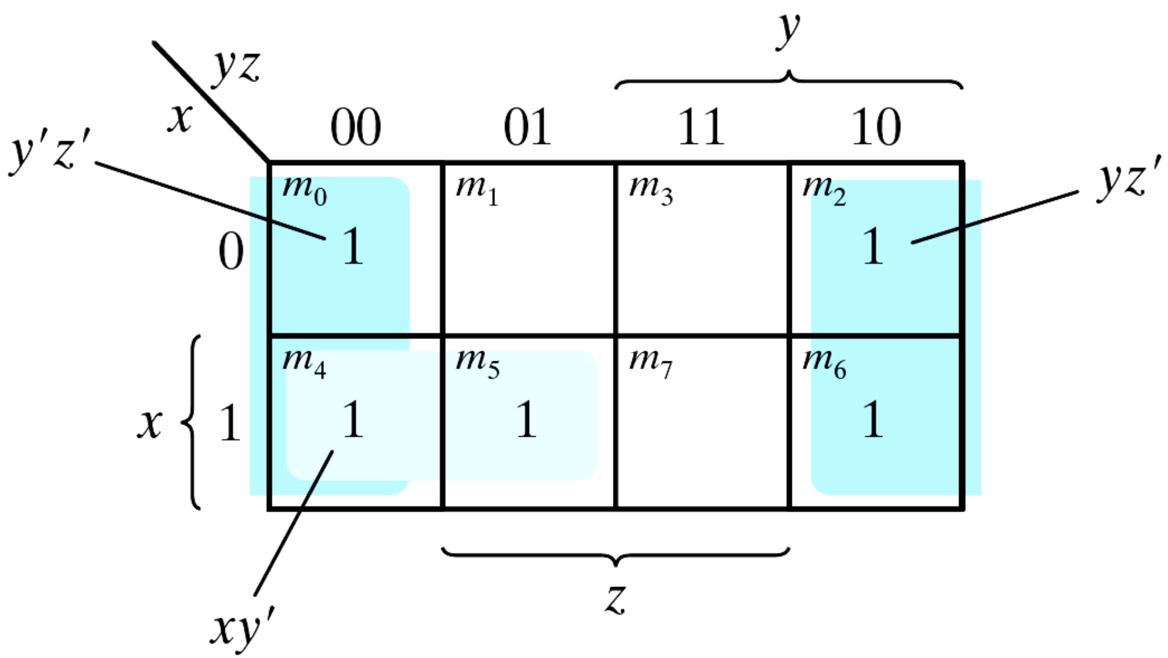
- Example simplify *F*(*x*, *y*, *z*) = S(3, 4, 6, 7)
- $F(x, y, z) = \Sigma\ (3, 4, 6, 7) = yz + xz'$



Note: $xy'z' + xyz' = xz'$

A Three-variable Map

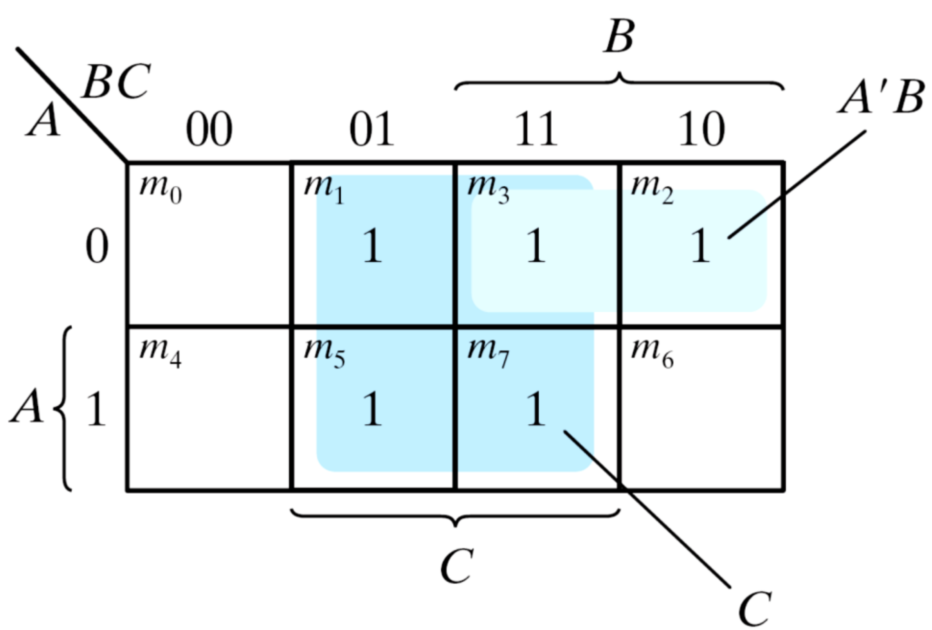
- Example: simplify $F(x, y, z) = \sum(0,2,4,5,6)$
- $F(x, y, z) = \Sigma\,(0, 2, 4, 5, 6) = z' + xy'$



Note: $y'z' + yz' = z'$

A Three-variable Map

- Example: let $F = A'C + A'B + AB'C + BC$
  a) Express it in sum of minterms.
  b) Find the minimal sum of products expression.
  $F(A, B, C) = \Sigma(1, 2, 3, 5, 7) = C + A'B$



Four-Variable Map

- The map (w,x,y,z)
  - 16 minterms
  - Combinations of 2, 4, 8, and 16 adjacent squares

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}

(a)

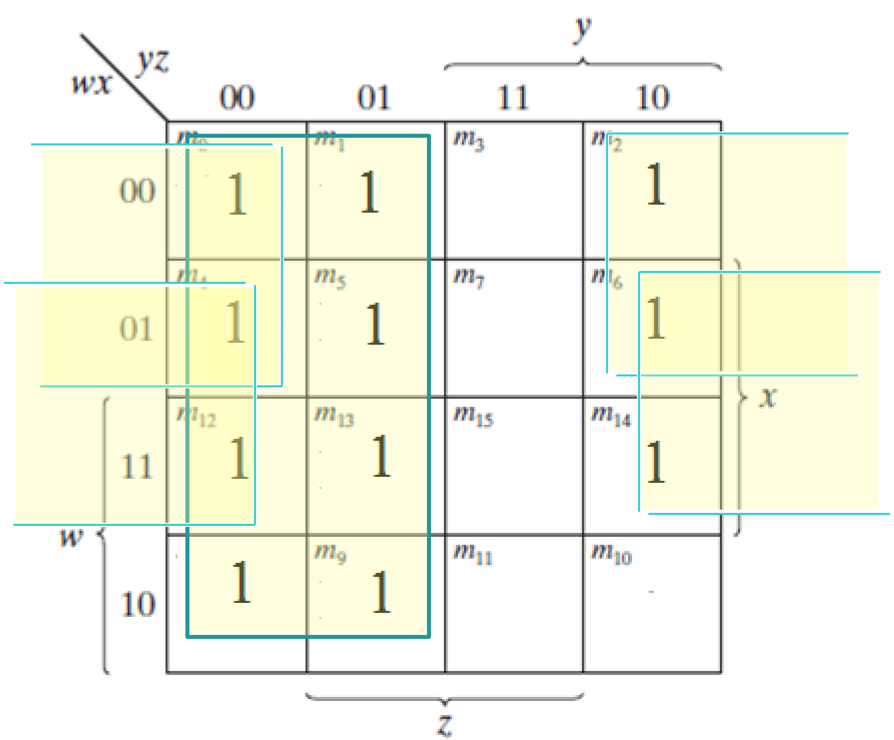
$wx \backslash yz$	00	01	11	10
00	m_0 $w'x'y'z'$	m_1 $w'x'y'z$	m_3 $w'x'yz$	m_2 $w'x'yz'$
01	m_4 $w'xy'z'$	m_5 $w'xy'z$	m_7 $w'xyz$	m_6 $w'xyz'$
11	m_{12} $wxy'z'$	m_{13} $wxy'z$	m_{15} $wxyz$	m_{14} $wxyz'$
10	m_8 $wx'y'z'$	m_9 $wx'y'z$	m_{11} $wx'yz$	m_{10} $wx'yz'$

y / x / z / w

(b)

Four-Variable Map

- Example: simplify $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



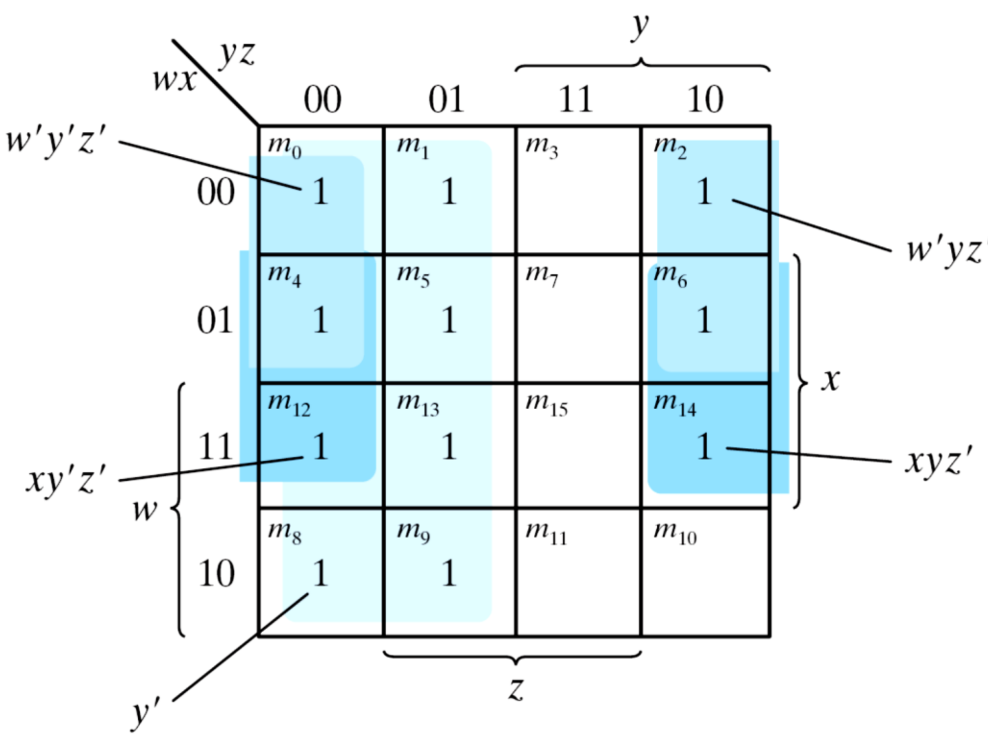
F = y'+

w'z'+

xz'

Four-Variable Map

- Example: simplify $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

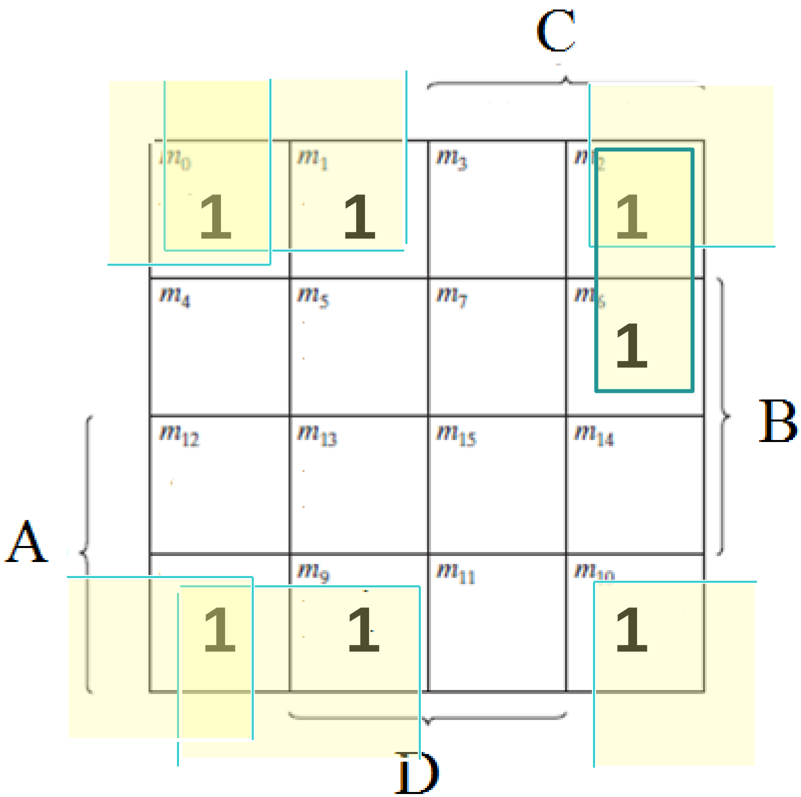


Note: $w'y'z' + w'yz' = w'z'$
$xy'z' + xyz' = xz'$

F = y'+w'z'+xz'

Four-Variable Map

- Example: simplify *F =A'B'C'+B'CD'+ A'BCD'+ AB'C'D+AB'D'*



B'C

B'D'

A'CD'

F = B'C'+B'D'+A'CD'

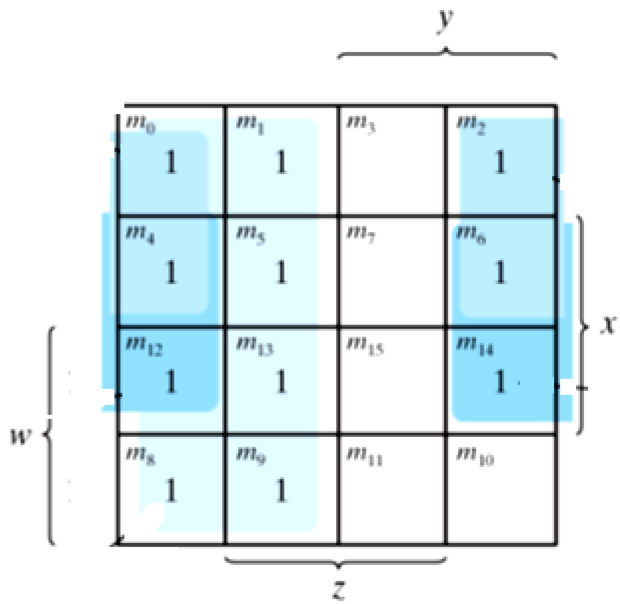
Number of adjacent squares and the number of literals

- 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		
2^k	$n = 2$	$n = 3$	$n = 4$
1	2	3	4
2	1	2	3
4	0	1	2
8		0	1
16			0

Product of Sums Simplification

- It is obvious that we can represent the function F in sum of product directly using K-Map



F = y'+w'z'+xz'

What if we want to represent F in Product of Sum Form !!!!!!!!!

Product of Sums Simplification

1. Get a Simplified *F'* in the form of sum of products (groupings zeros of F)
2. Get F with aid of DeMorgan's theorem $F = (F')'$

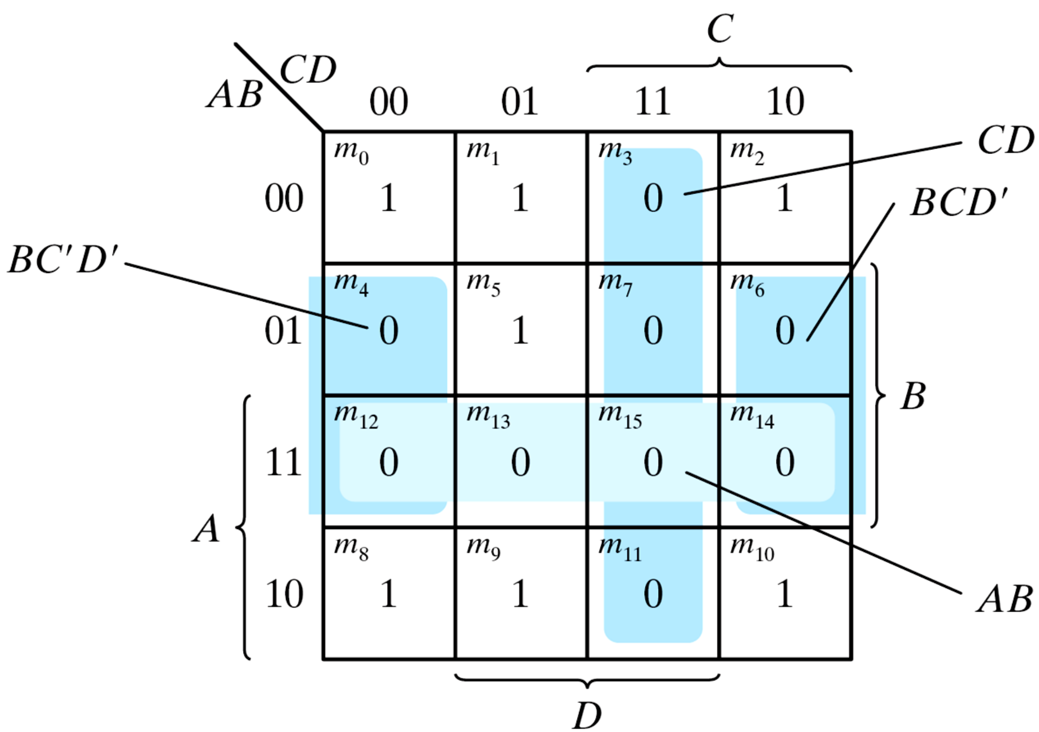
F': sum of products $\longrightarrow$ F: product of sums

Product of Sums Simplification

- Simplify F = ∑ (0, 1, 2, 5, 8, 9, 10) into

(a) sum-of-products form, and

(b) product-of-sums form:



Note: $BC'D' + BCD' = BD'$

F(A, B, C, D)=
B'D'+B'C'+A'C'D

F' = AB+CD+BD‘

then

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

Product of Sums Simplification

- Consider the function defined in Table 3.2.
  - In sum-of-minterm:

$$\boxed{F(x,y,z)=\sum(1,3,4,6)}$$

  - In sum-of-maxterm:

$$\boxed{F'(x,y,z)=\Pi(0,2,5,7)}$$

Table 3.2
Truth Table of Function F

x	*y*	*z*	*F*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Product of Sums Simplification

- Consider the function defined in Table 3.2.
  - Combine the 1's:

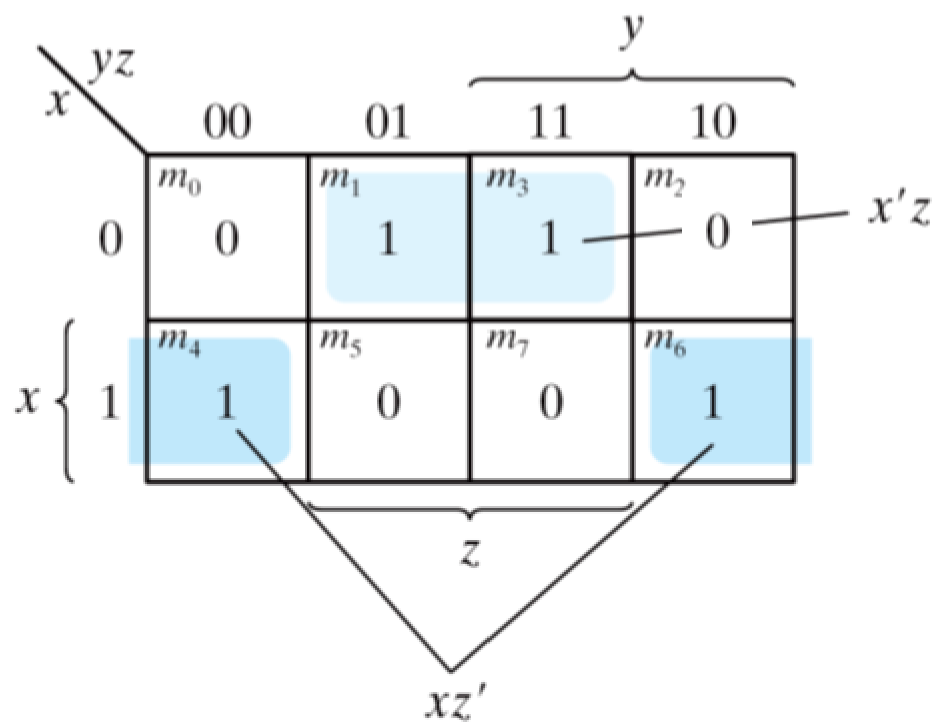
$$\boxed{F(x,y,z) = x'z + xz'}$$

  - Combine the 0's :

$$\boxed{F'(x,y,z) = xz + x'z'}$$

- Taking the complement of F′

$$\boxed{F(x,y,z) = (x'+z')(x+z)}$$



Don't-Care Conditions

- The value of a function is not specified for certain combinations of variables
- The don't-care conditions can be utilized in logic minimization
  - Can be implemented as 0 or 1

Don't-Care Conditions

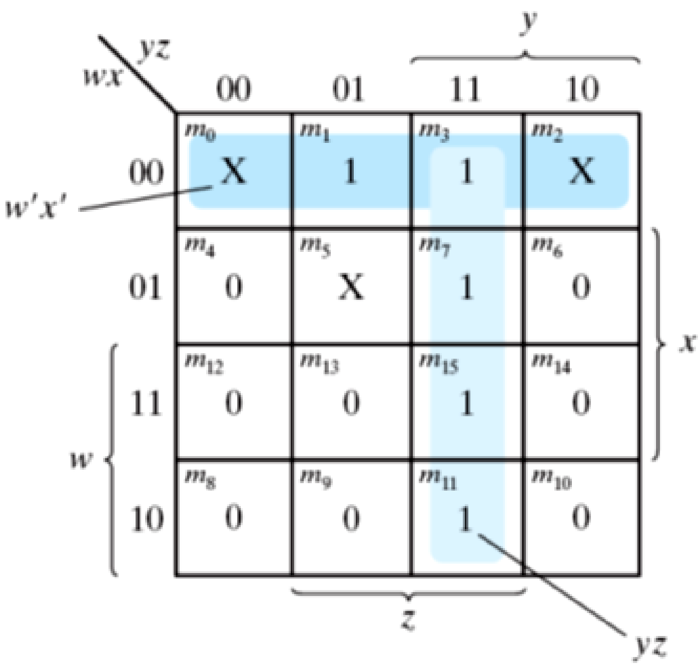
$$F = 1 \quad \text{if} \quad 0 < (wxyz)_{10} < 4$$
$$F = 0 \quad \text{if} \quad 4 \leq (wxyz)_{10} < 9$$

$wx \backslash yz$	00	01	11	10
00	x	1	1	1
01	0	0	0	0
11	x	x	x	x
10	0	x	x	x

w	x	y	z	F
0	0	0	0	x
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	0
1	0	0	1	x
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

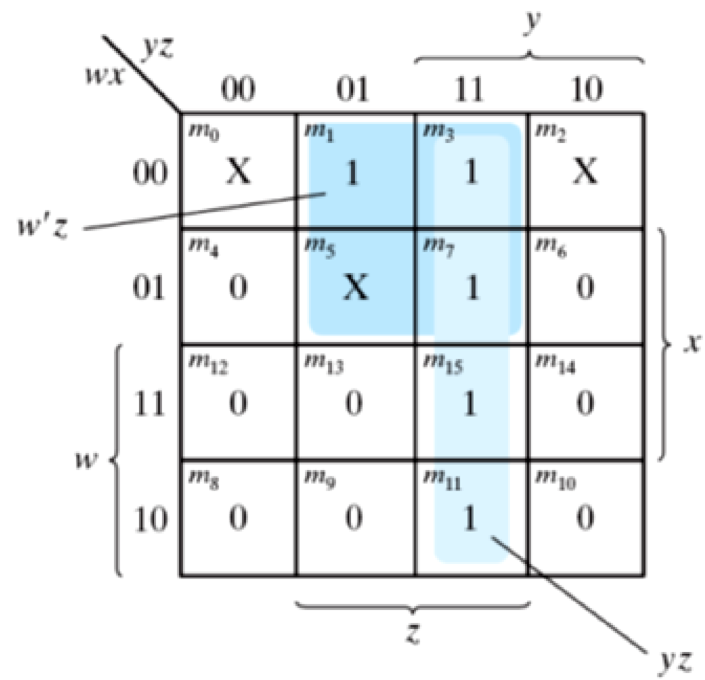
Don't-Care Conditions

- Example: simplify $F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$ which has the don't-care conditions $d(w, x, y, z) = \Sigma(0, 2, 5)$.



(a) $F = yz + w'x'$

$F = \Sigma(0, 1, 2, 3, 7, 11, 15)$



(b) $F = yz + w'z$

$F = \Sigma(1, 3, 5, 7, 11, 15)$


Thank You!