Lecture – 7 - Chapter 3

Outline

- Minimization
- Karnaugh maps (K-maps)

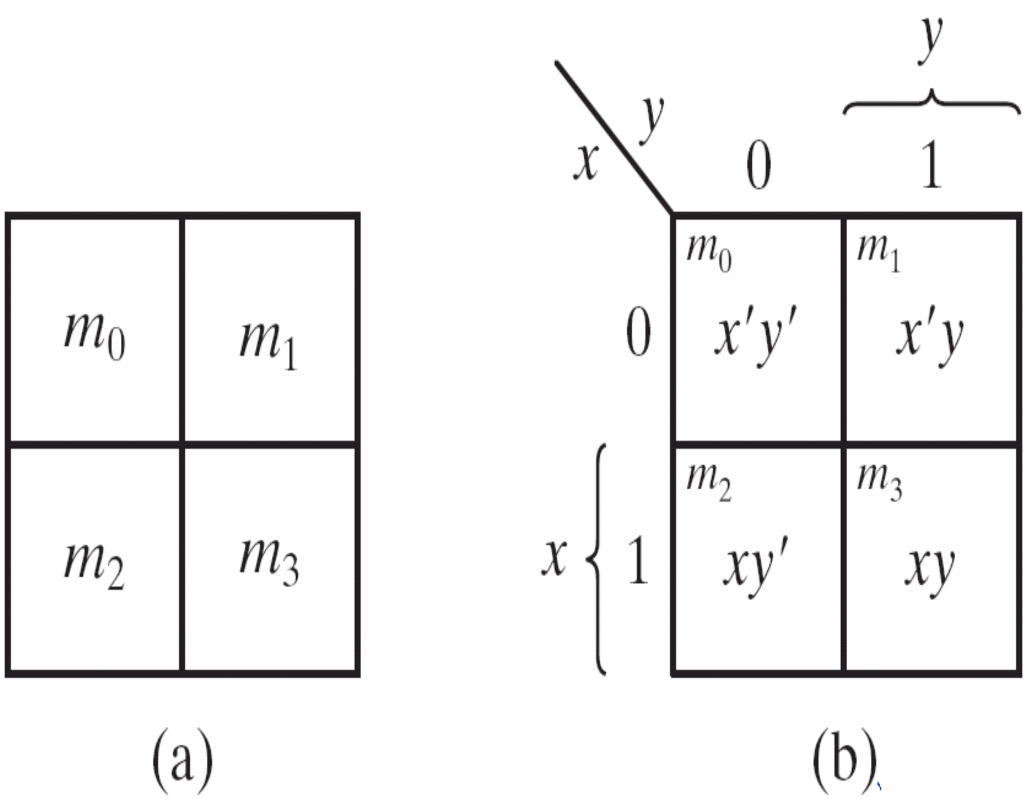
Karnaugh Maps (K-map)

- A K-map
  - Made up of squares
  - Each square represents a minterm
  - a graphical representation of a Boolean function
  - Alternative algebraic expressions for the same function are derived by recognizing patterns of squares
  - The simplified expressions produced by the map are always in the form of sum of products. (implemented in two-level circuits)

- The K-map can be viewed as
  - A reorganized version of the truth table

Two Variable Maps

- A 2-variable Karnaugh Map:



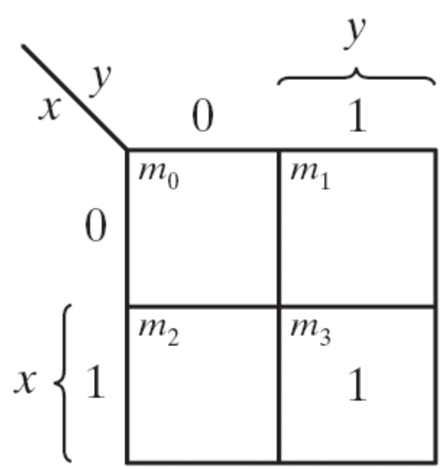
K-Map and Truth Tables

- The K-Map is just a different form of the truth table.
- Example – Two variable function:

F(x,y)=xy

Function Table

Input Values (x,y)	Function Value F(x,y)
0 0	0
0 1	0
1 0	0
1 1	1



(a) xy

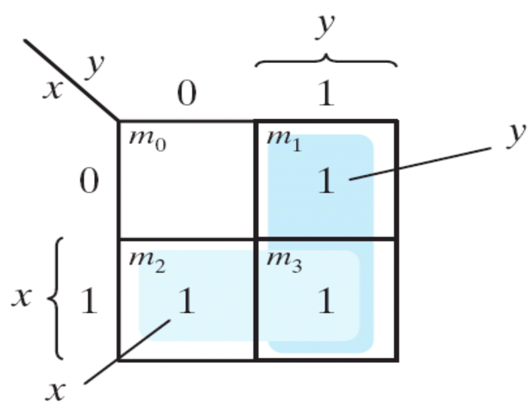
G(x,y) = m3

K-Map Function Representation

- Example: G(x,y) = x'y+xy'+xy

Function Table

Input Values (x,y)	Function Value F(x,y)
0 0	0
0 1	1
1 0	1
1 1	1



(b) $x + y$

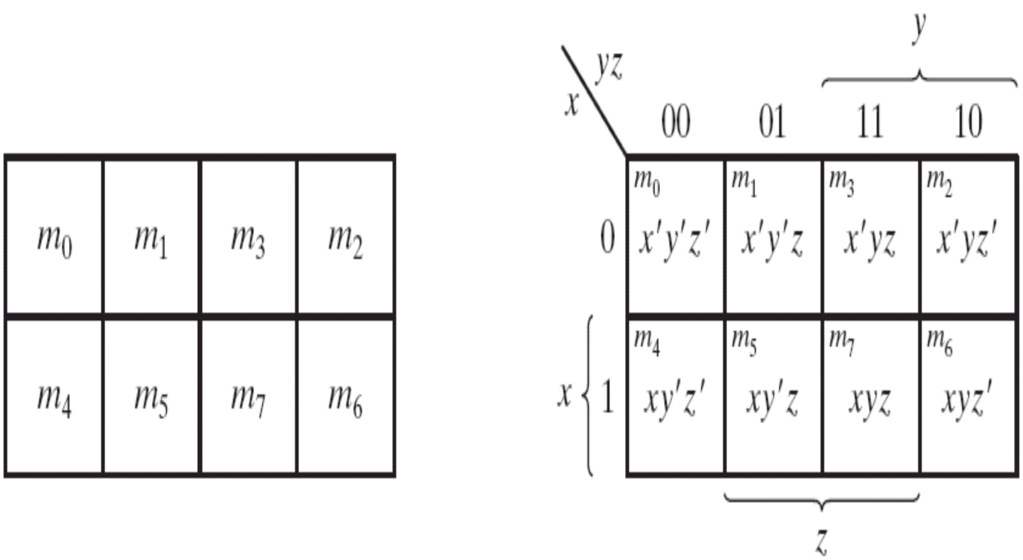
- G(x,y) = m1+m2+m3
- For G(x,y), two pairs of adjacent cells containing 1's can be combined using the Minimization Theorem:

$$G(x,y) = (\ x\,y' + x\,y) + (\ x\,y + x'\ y\ ) = x + y$$

Duplicate xy

Three Variable K-Map

- Row and Columns
- Any two adjacent squares in the map differ by only one variable
- two pairs of adjacent squares can be combined by re-moving the disimilar variable   m5 + m7

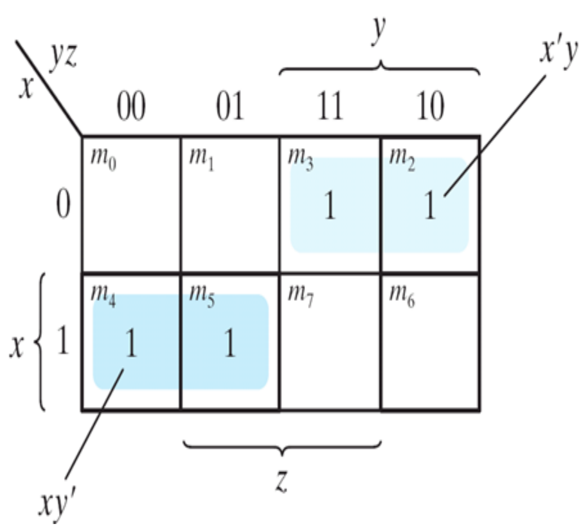


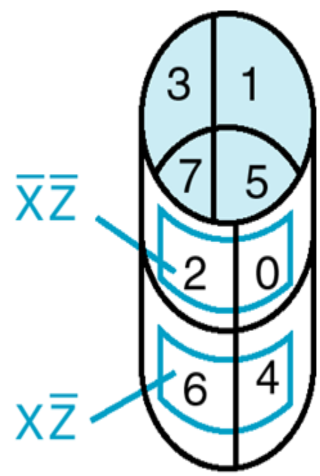
(a) (b)

Three Variable K-Map

- $F(x,y,z) = \sum(2,3,4,5)$

$m_3+m_2=?$ $m_0+m_2=?$
$m_4+m_5=?$ $m_4+m_6=?$



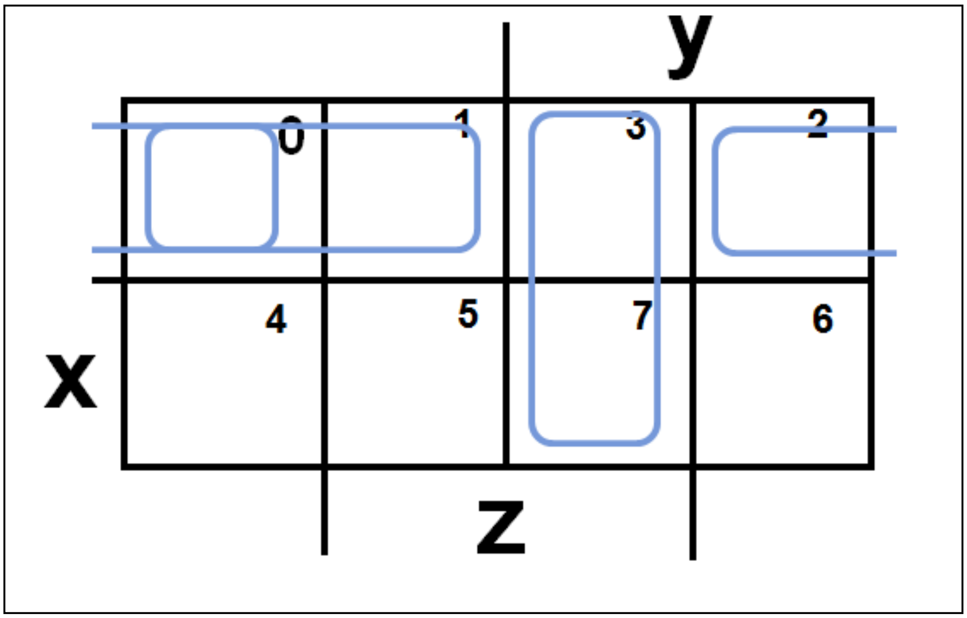


On a 3-variable K-Map:

- Two adjacent squares (2-cell Rectangles) represent a product term with two variables

Three-Variable Maps

- Example Shapes of 2-cell Rectangles:

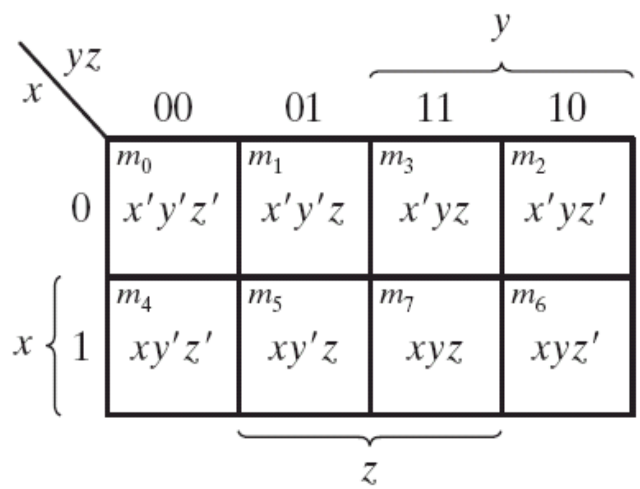


- Read off the product terms for the rectangles shown

THREE VARIABLE K-MAP

- More practice:

  F(x, y, z)= ∑ (2,3,6,7)
  F(x, y, z)= ∑ (1,3,5,7)
  F(x, y, z)= ∑ (0,2,4,6)

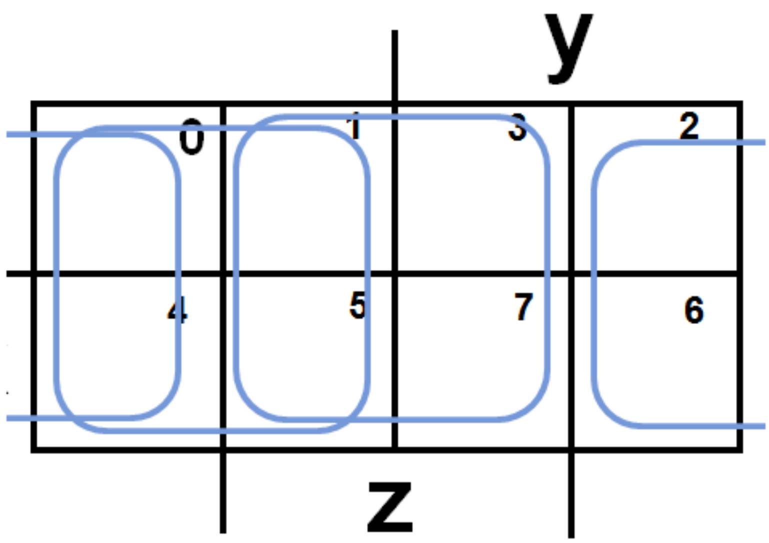


On a 3-variable K-Map:

- Four “adjacent” terms (Rectangles of 4 cells) represent a product term with one variable
- More adjacent squares are combined, obtain a product term with fewer literals

Three-Variable Maps

- Example Shapes of 4-cell Rectangles:

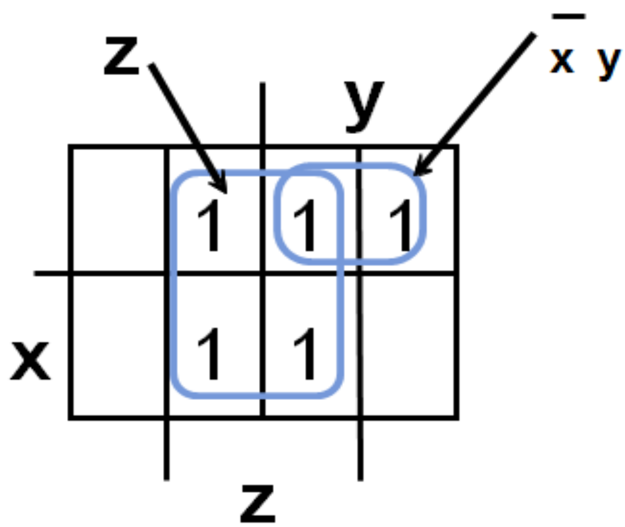


- Read off the product terms for the rectangles shown

Three Variable Maps

- K-Maps can be used to simplify Boolean functions by systematic methods. Terms are selected to cover the "1s"in the map.

- Example: Simplify $F(x, y, z) = \Sigma_m(1,2,3,5,7)$



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

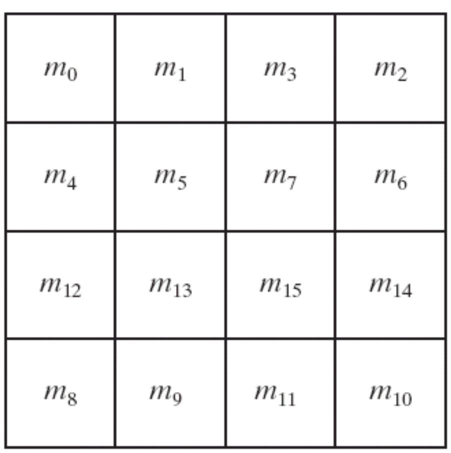
THREE VARIABLE MAPS

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

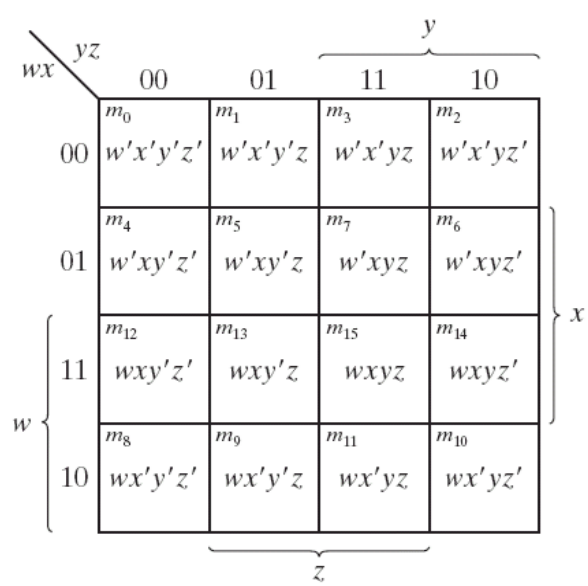
- Each produce term can be plotted in the map in one, two, or more squares

- The minterms of the function are then read directly from the map.

Four Variable Maps



(a)



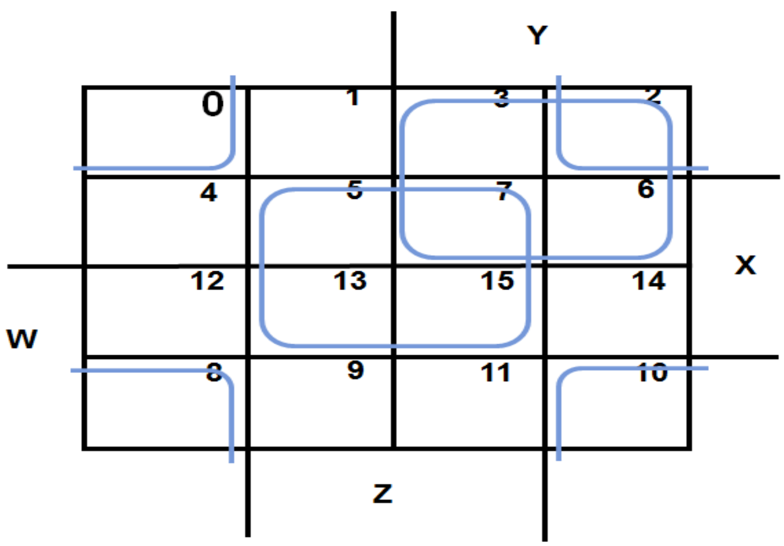
(b)

Four Variable Maps

- Four variable maps can have rectangles corresponding to:
  - Two adjacent squares = 3 variables,
  - Four adjacent squares = 2 variables
  - Eight adjacent squares = 1 variable,
  - Sixteen adjacent squares = zero variables (i.e. Constant "1")
- The larger the number of squares combined, the smaller is the number of variables

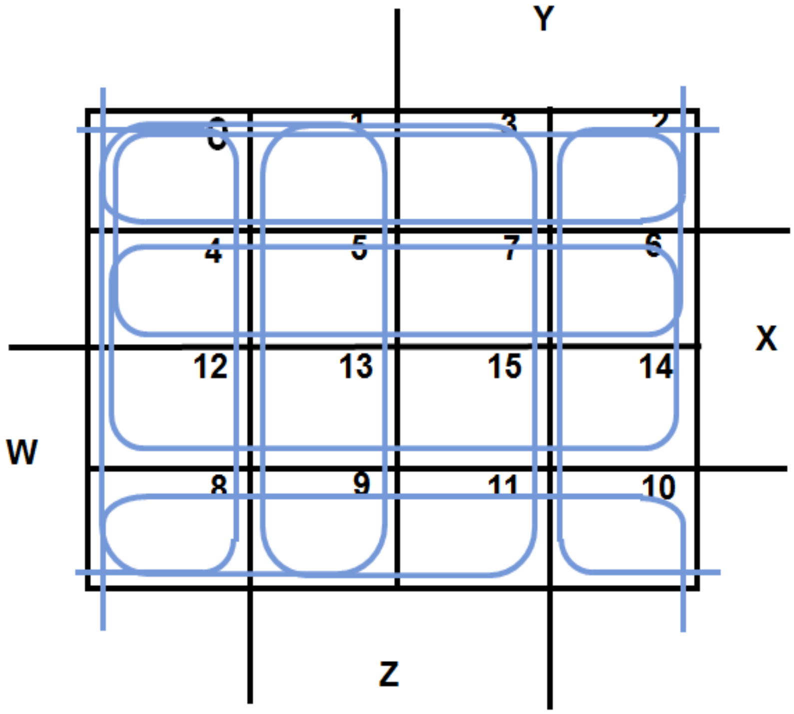
Four-Variable Maps

- Example Shapes of Rectangles:



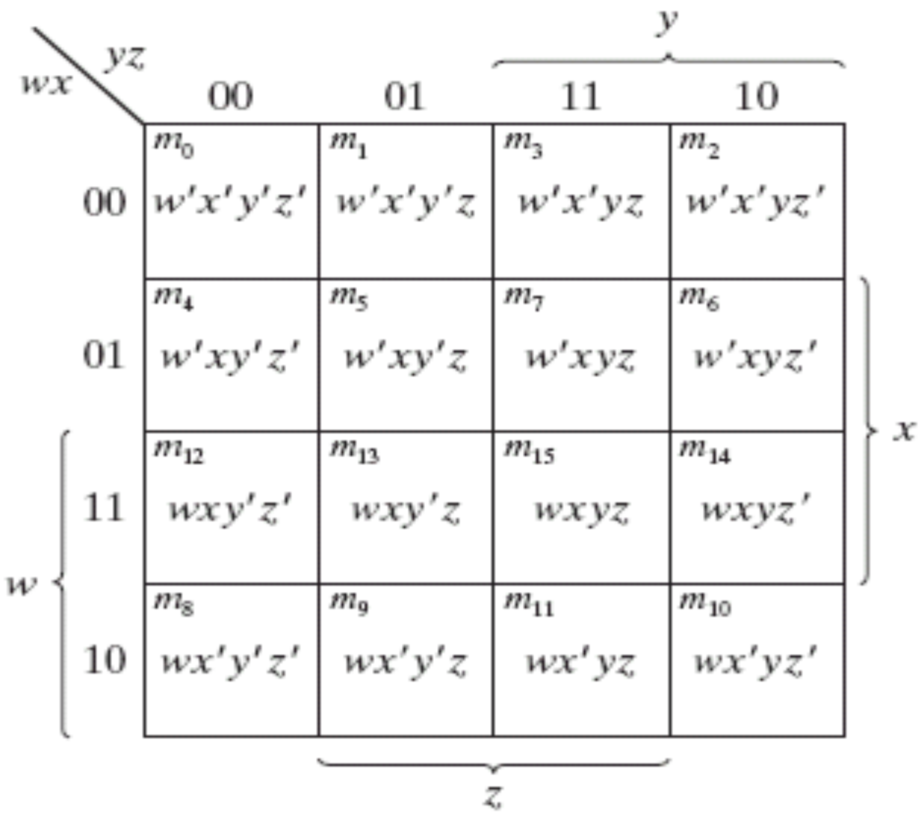
Four-Variable Maps

- Example Shapes of Rectangles:



Four-Variable Map Simplification

- F(w,x, y, z)= ∑ (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)



Four-Variable Map Simplification

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

Simplification Rules

- Objectives :

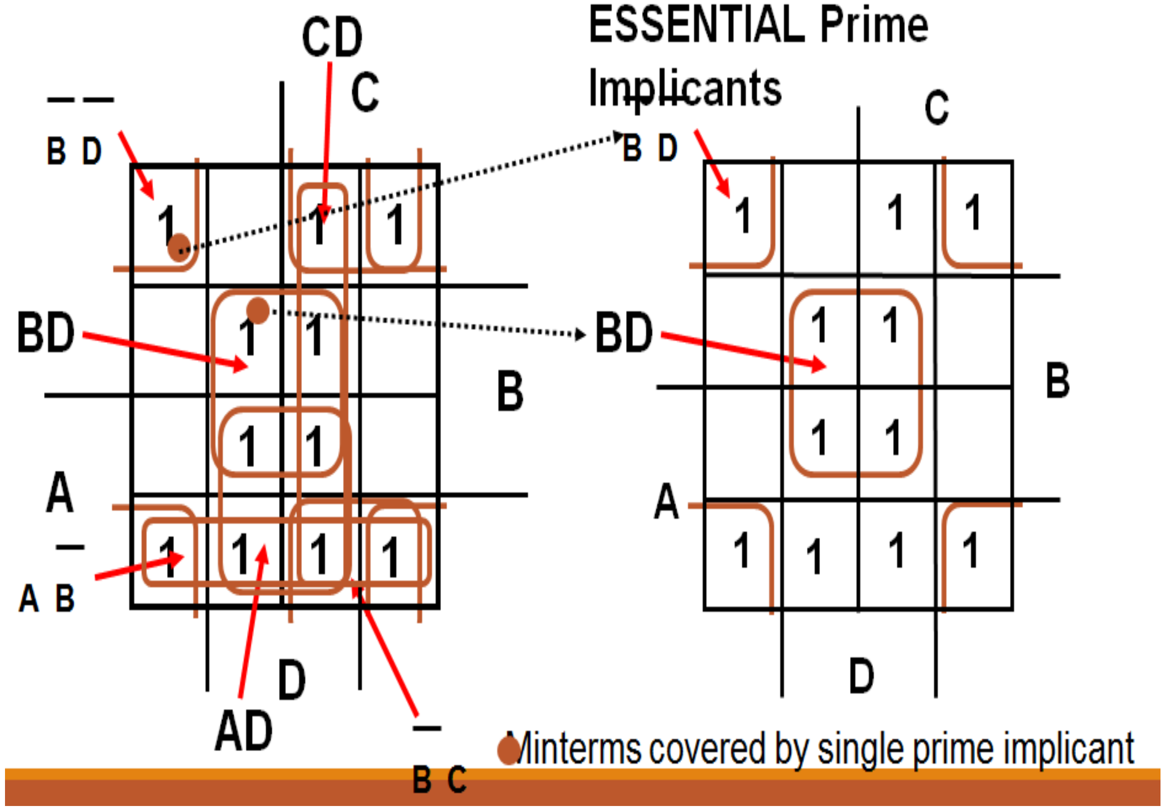
  - All the minterms of the functions are covered.
  - The number of terms in the expression is minimized
  - There are no redundant minterms.

- ***Prime Implicant:*** is a **product term** obtained by combining the **maximum possible number** of adjacent squares in the map into a rectangle.

- A prime implicant is called an ***Essential Prime Implicant*** if a minterm in a square is covered by only this prime implicant.

Example of Prime Implicants

- Find ALL Prime Implicants

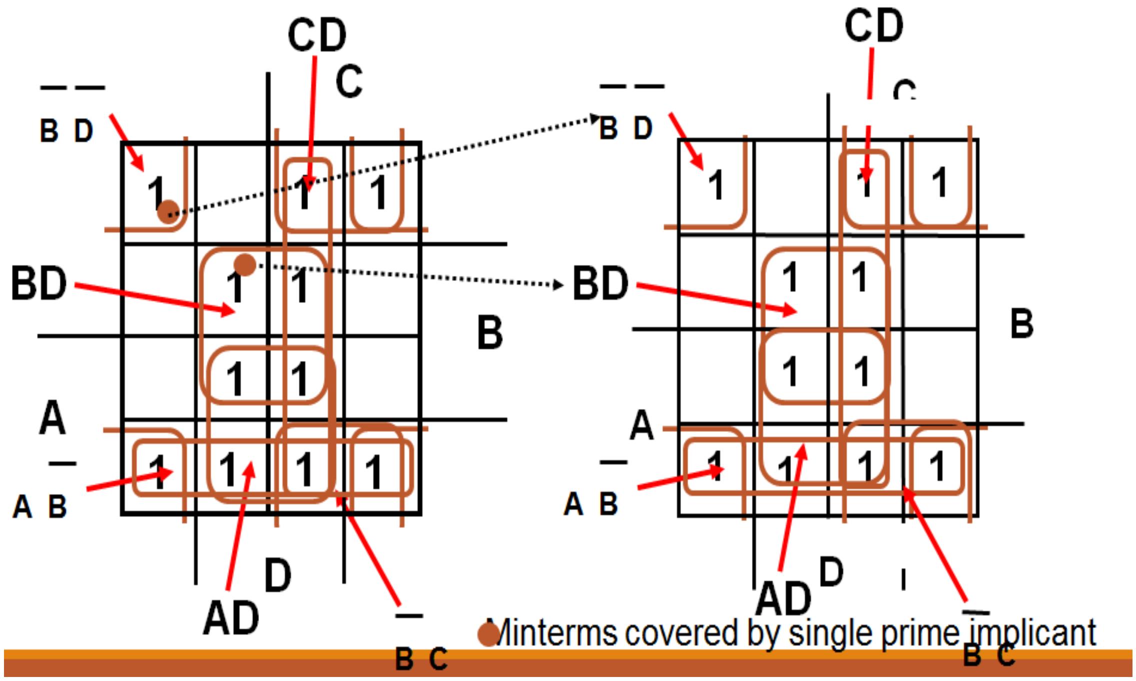


Optimization Algorithm

- Find all prime implicants.
- Include all essential prime implicants in the solution
- Select a minimum cost set of non-essential prime implicants to cover all minterms not yet covered:
    - Obtaining an optimum solution
      (There may be more than one way of combining squares)

Example of Prime Implicants

ESSENTIAL Prime Implicants



Optimization Algorithm

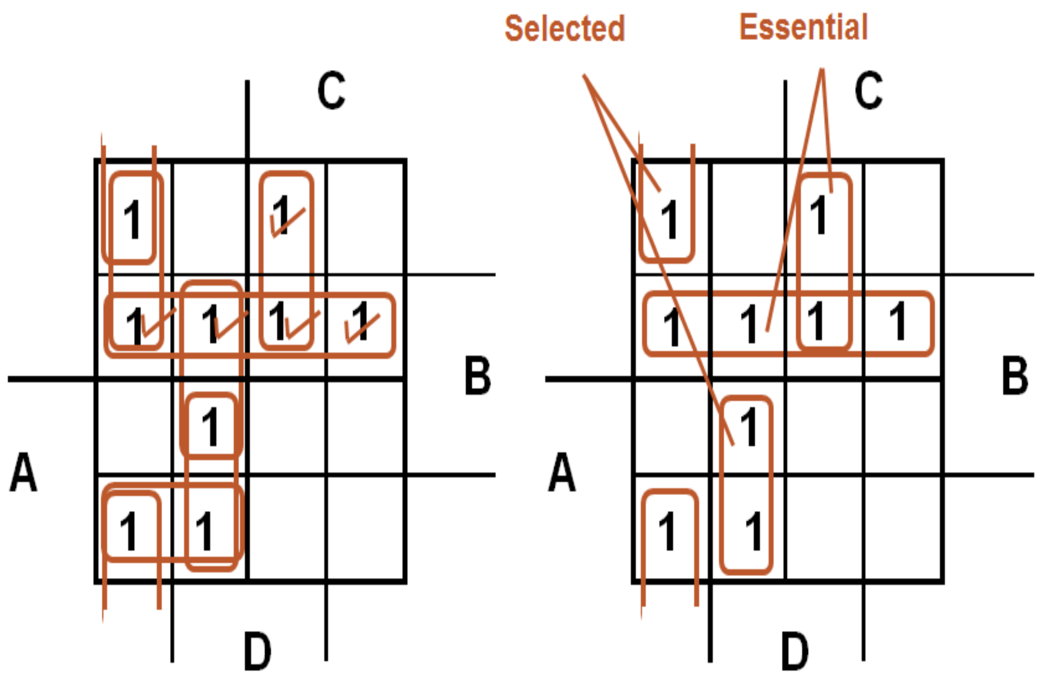
- Find all prime implicants.
- Include all essential prime implicants in the solution
- Select a minimum cost set of non-essential prime i mplicants to cover all minterms not yet covered:
    - Obtaining an optimum solution
    - There may be more than one way of combining squares
    - Obtaining a good simplified solution: Use the Selec- tion Rule

Prime Implicant Selection Rule

- Minimize the overlap among prime implicants as much as possible.
- Make sure that each prime implicant selected includes at least one minterm not included in any other prime implicant selected.

Selection Rule Example

- **Simplify F(A, B, C, D) given on the K-map.**



Minterms covered by essential prime implicants

Don't Cares in K-Maps

- Incompletely specified functions:
- function table or map contains entries
  - the input values for the minterm never occur, or
  - the output value for the minterm is not used
- In these cases, the output value need not be defined
- Instead, the output value is defined as a "don't care"

- Example 1: A logic function having the binary codes for the BCD digits as its inputs. Only the codes for 0 through 9 are used. The six codes, 1010 through 1111 never occur, so the output values for these codes are "x" to represent "don't cares."

- Example 2: A circuit that represents a very common situation that occurs in computer design
  - Input A, B, and C which take on all possible combinations, and
  - a single output Z =1 only for combinations A = 1 and B = 1 or C = 1, otherwise ignoring it.
  - Thus, Z is specified only for those combinations, and for all other combinations of A, B, and C, Z is a don’t care. Specifically, Z must be specified for AB + C = 1, and is a don’t care for :

    AB + C = 0

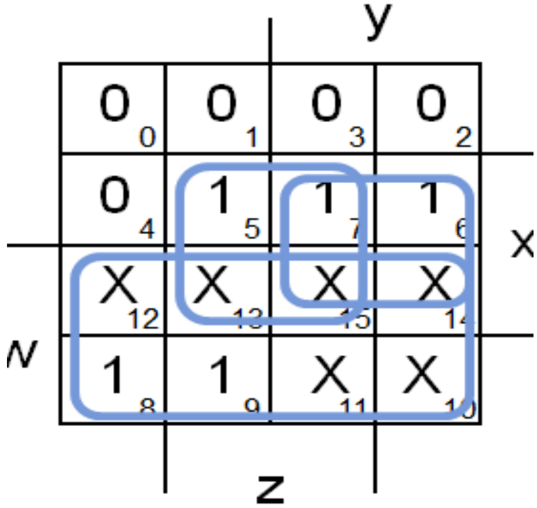
- By placing “don't cares” (an “x” entry) in the function table or map, the cost of the logic circuit may be lowered.
- Ultimately, each don’t care “x” entry may take on either a 0 or 1 value in resulting solutions

Don’t Care

- F(w,x, y, z)= ∑ (1, 3, 7, 11, 15)
- D(w,x,y,z)= ∑ (0, 2, 5)

EXAMPLE: BCD “5 OR MORE”

- The map below gives a function F1(w,x,y,z) which is defined as "5 or more" over BCD inputs. With the don't cares used for the 6 non-BCD combinations:



F1 (w,x,y,z) = w + x z + x y

This is much lower in cost than F2 where the “don't cares” were treated as "0s.“

$$F_2(w, x, y, z) = \bar{w}\, x\, z + \bar{w}\, x\, y + w\, \bar{x}\, \bar{y}$$

Example

- Find the optimum SOP solution:

$$\mathbf{F(A,B,C,D) = \Sigma_m(3,4,6,9,11) + \Sigma d(2,5,7,10,13)}$$

Selection Rule Example with Don't Cares

- Simplify F(A, B, C, D) given on the K-map.