COE 202- Digital Logic

K-Map Simplification

Dr. Abdulaziz Y. Barnawi

COE Department

KFUPM

Outline

- Introduction
- Learn to minimize a function using K-Maps
  - 2-Variable K-Maps
  - 3-Variable K-Maps
  - 4-Variable K-Maps
- Prime and essential prime implicants
- SOP and POS simplification
- Simplification using don't care conditions
- 5-Variable K-Maps

Introduction

- Even though Boolean expressions can be simplified by algebraic manipulation, such an approach lacks clear regular rules for each succeeding step and it is difficult to determine whether the simplest expression has been achieved. Also, mistakes are easy to make
- In contrast, Karnaugh map (K-map) method provides a straightforward procedure for simplifying Boolean functions.
- K-maps of up to 4 variables are very common to use. Maps of 5 and 6 variables can be made as well, but are more cumbersome to use.

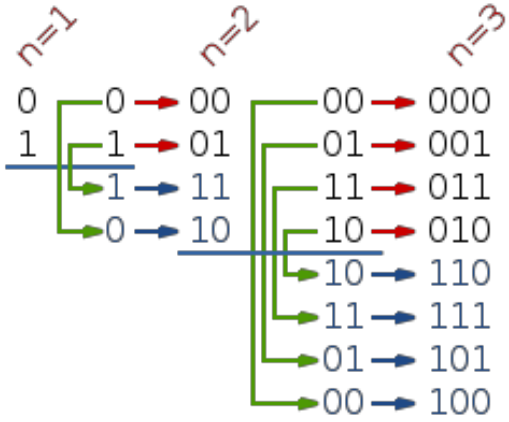
Introduction

- Simplified expressions produced by K-maps are always either in the SOP or the POS form.
- The map provides the same information contained in a Truth Table but in a different format.
- The objectives of this unit are to learn:
  - How to build a 2, 3, or 4 variables K-map.
  - How to obtain a minimized SOP or POS function using K-maps.

Gray Codes (review)

- The distance between two binary code-words is the number of bit positions in which the two code-words have different values
- For example, the distance between the code words 1001 and 0001 is 1 while the distance between the code-words 0011 and 0100 is 3
- This definition of code distance is commonly known as the Hamming distance between two codes
- In Gray Code, only one bit changes with each number increment

src: wikipedia.org



Truth Table Adjacencies

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

These minterms are adjacent in a gray code sense – they differ by only one bit.

We can apply the identity XY+XY'=X

F = A'B' + A'B = A'(B'+B) = A' (1)
= A'

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

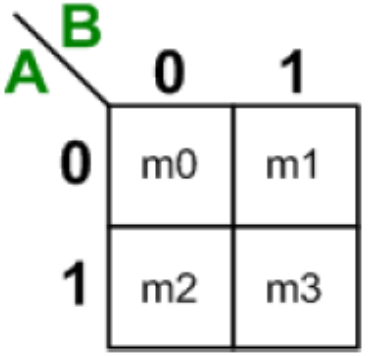
Same idea:

F = A'B + AB = B

Keep common literal only!

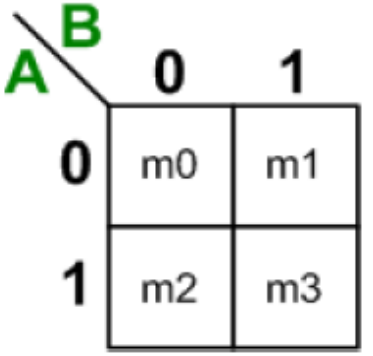
2-Variable K-Maps

- The 2-variable map is a table of 2 rows by 2 columns.
- The 2 rows represent the two values of the first input variable A, while the two columns represent the two values of the second input variable B.
- Thus, all entries (squares) in the first row correspond to input variable A=0, while entries (squares) of the second row correspond to A=1.
- Likewise, all entries of the first column correspond to input variable B = 0, while entries of the second column correspond to B=1.



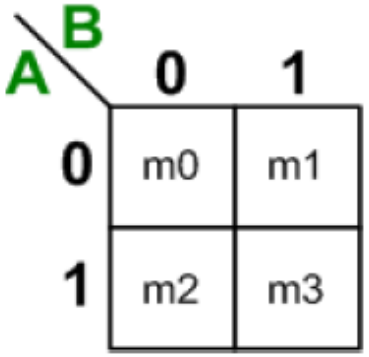
2-Variable K-Maps

- Thus, each map entry (or square) corresponds to a unique value for the input variables A and B.
- For example, the top left square corresponds to input combination AB=00. In other words, this square represents minterm m_0.
- Likewise, the top right square corresponds to input combination AB=01, or minterm m_1.
- In general, each map entry (or square) corresponds to a particular input combination (or minterm).



2-Variable K-Maps

- Two K-map squares are considered adjacent if the input codes they represent have a Hamming distance of 1.
- A K-map square with a function value of 1 will be referred to as a 1-Square.
- A K-map square with a function value of 0 will be referred to as a 0-Square.



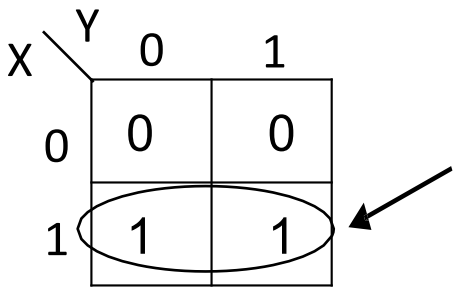
- The simplification procedure is summarized below:
  - Step 1: Draw the map according to the number of input variables of the function, i.e. 2^n squares, n = #of variables
  - Step 2: Fill "1's" in the squares for which the function output is "1"

2-Variable K-Maps

- Step 3: Form as big group of adjacent 1-squares as possible. There are some rules for this which you will learn with bigger maps.
- Step 4: Find the common literals for each group and write the simplified expression in SOP.

- Example: Consider the given truth table of two variable function. Obtain the simplified function using K-map.

X	Y	F
0	0	0
0	1	0
1	0	1
1	1	1

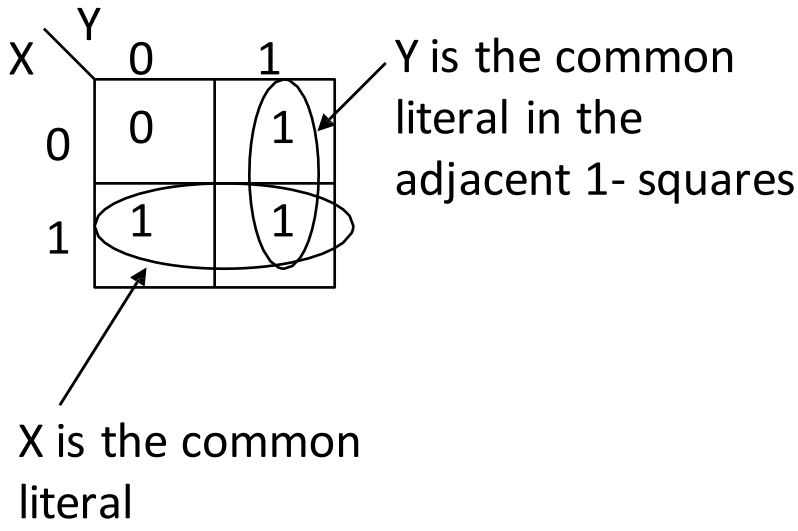


Combining all the 1's in only the adjacent squares

The final reduced expression is given by the common literals from the combination. So, the reduced function is: F(X,Y) = X

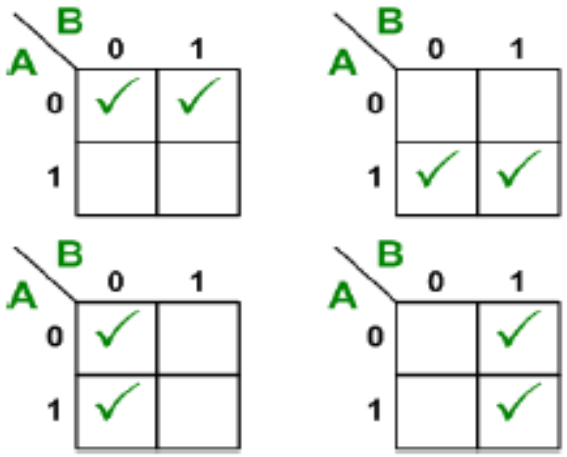
2-Variable K-Maps

- Question: Simplify the function F(X,Y) = ∑m(1,2,3)
- Solution: This function has 2 variables, and three 1-squares (three minterms where function is 1)
  - $F = m_1 + m_2 + m_3$
  - Note: The 1-squares can be combined more than once
  - The Minimized expression is **F = X + Y**

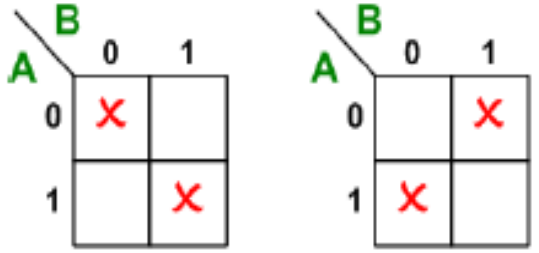


2 variable K-Maps (Adjacency)

- In an n-variable map each square is adjacent to “n” other squares, e.g., in a 2-variable map each square is adjacent to two other squares



- Examples of non-adjacent squares



3-Variable K-Maps

- There are eight minterms for a Boolean function with three-variables, e.g. *F(A, B, C)*
- Hence, a three-variable map consists of $2^3 = 8$ squares
- All entries (squares) in the first row correspond to input variable A=0, while entries (squares) of the second row correspond to A=1.

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

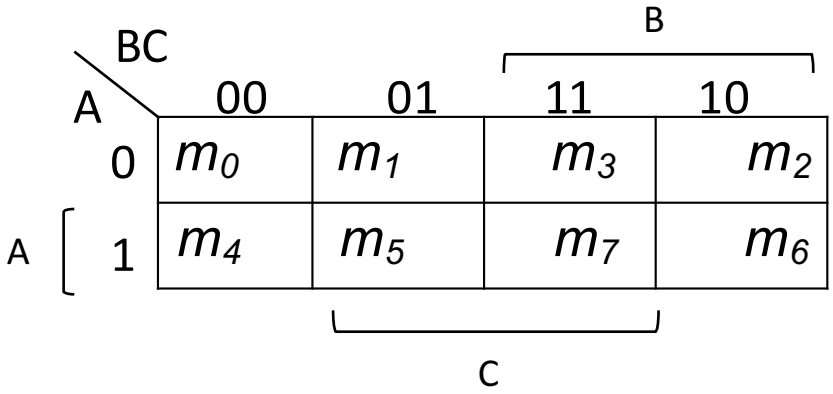
3-Variable K-Maps

- Likewise, all entries of the first column correspond to input variable B = 0, C = 0.
- All entries of the second column correspond to input variable B = 0, C = 1.
- All entries of the third column correspond to input variable B = 1, C = 1.
- All entries of the fourth column correspond to B=1, C = 0

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

3-Variable K-Maps

- To maintain adjacency neighbors don't have more than 1 different bit



- There are cases where two squares in the map are considered to be adjacent even though they do not physically touch each other.

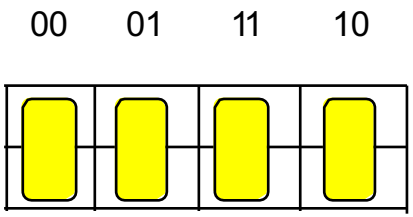
- m_0 is adjacent to m_2 and m_4 is adjacent to m_6 because the minterms differ by only one variable.

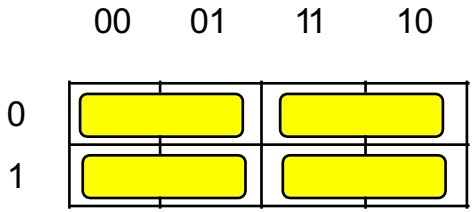
3-Variable K-Maps

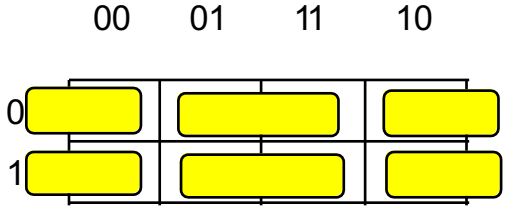
- How to minimize
  - Groups may only consist of 2, 4, 8, 16,… squares (always power of 2). For example, groups may not consist of 3, 6 or 12 squares.
  - Members of a group must have a closed loop adjacency, i.e., L-Shaped 4 squares do not form a valid group.
  - One square is represented by a minterm (i.e. a product term containing all 3 literals).
  - A group of 2 adjacent squares is represented by a product term containing only 2 literals, i.e., 1 literal is dropped.
  - A group of 4 adjacent squares is represented by a product term containing only 1 literal, i.e., 2 literals are dropped.

3 variable K-Maps (Adjacency)

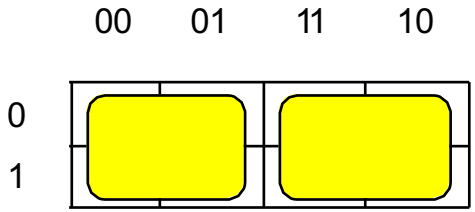
- A 3-variable map has 12 possible groups of 2 minterms. They become product terms with 2 literals

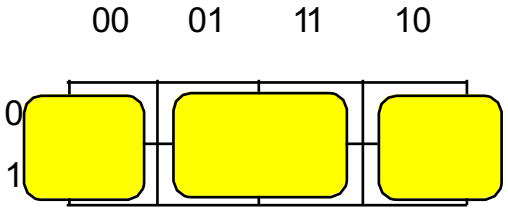


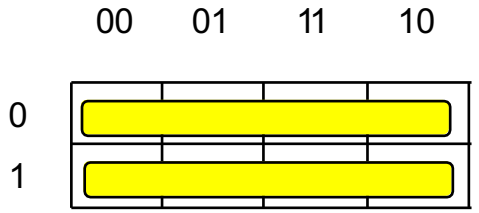




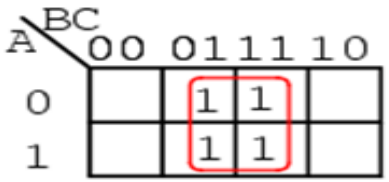
- A 3-variable map has 6 possible groups of 4 minterms. They become product terms with 1 literals



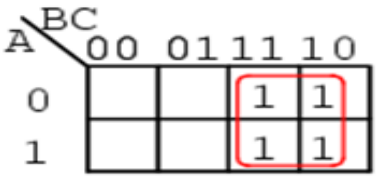




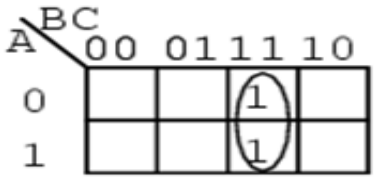
3-Variable K-Maps Examples



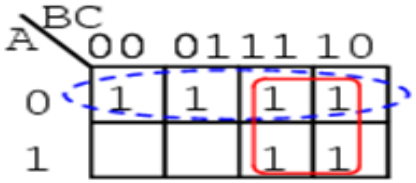
Out= C



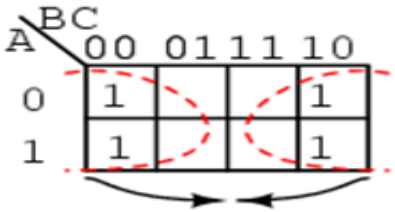
Out= B



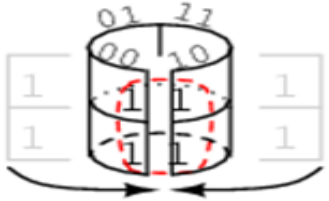
Out= BC



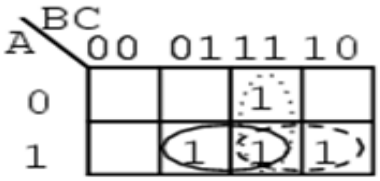
Out= $\overline{A}$ + B



Out= $\overline{C}$



Out= $\overline{A}$ + $\overline{C}$



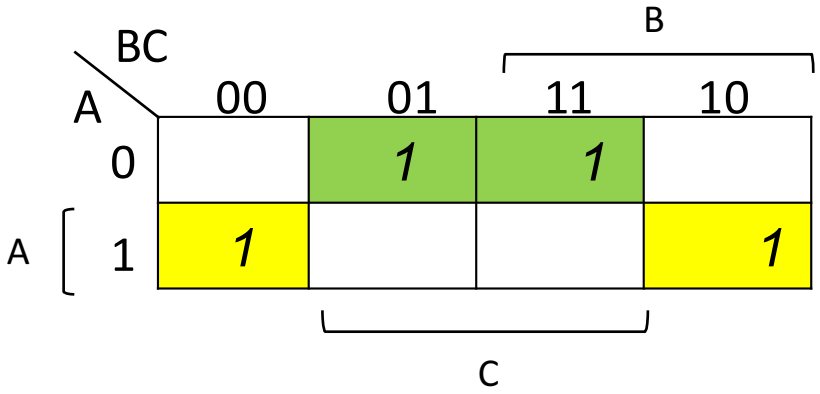
Output = AB + BC + AC

3-Variable K-Maps Examples

- Simplify F = $\sum$m(1, 3, 4, 6) using K-map

3-Variable K-Maps Examples

- Simplify F = $\sum$m(1, 3, 4, 6) using K-map



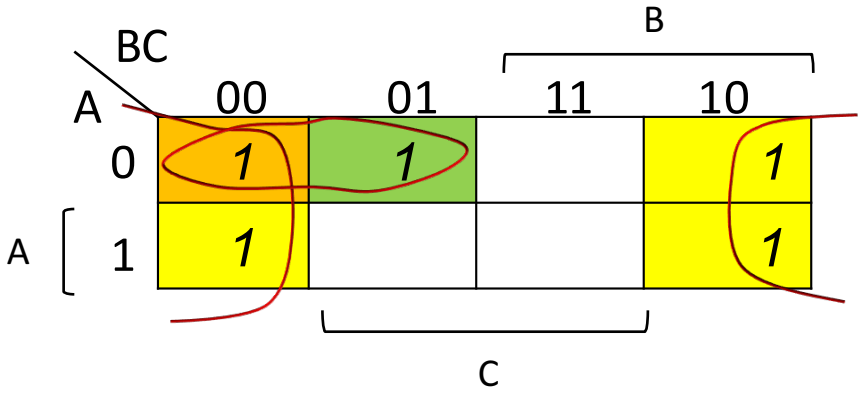
- Minimized F is ***F = A'C + AC'***

3-Variable K-Maps Examples

- Simplify F = $\sum$m(0,1, 2, 4, 6) using K-map

3-Variable K-Maps Examples

- Simplify F = ∑m(0,1, 2, 4, 6) using K-map



- Minimized F is ***F = A' B' + C'***

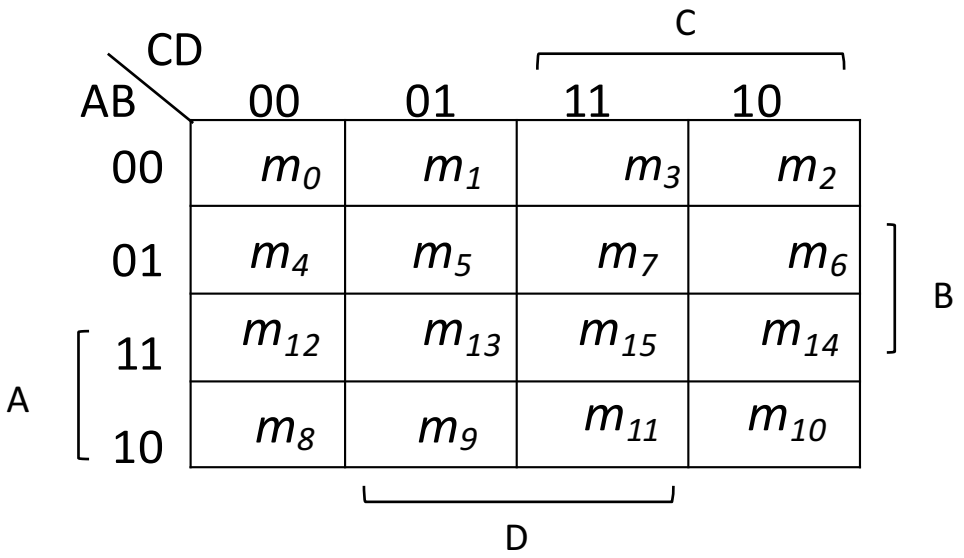
4-Variable K-Maps

- There are 16 minterms for a Boolean function with four-variables. Hence, four-variable map consists of 16 squares.
- Each square is adjacent to 4 other squares.
- A square by itself will represent a 4-literals output
- Combining 2 squares will generate a 3-literal output
- Combining 4 squares will generate a 2-literal output
- Combining 8 squares will generate a 1-literal output

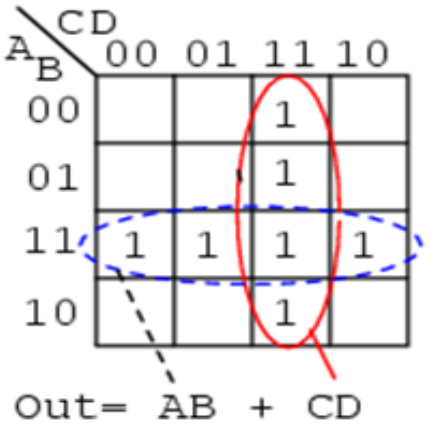
AB \ CD	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10

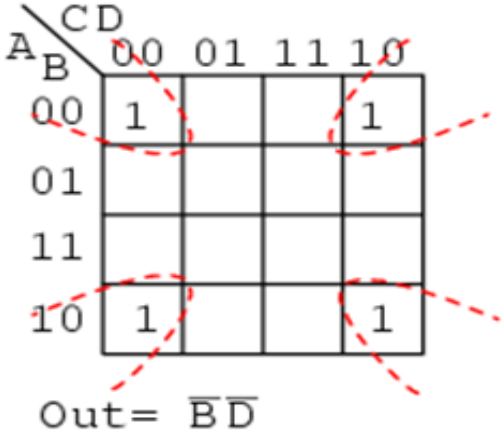
4-variable K-maps (Adjacency)

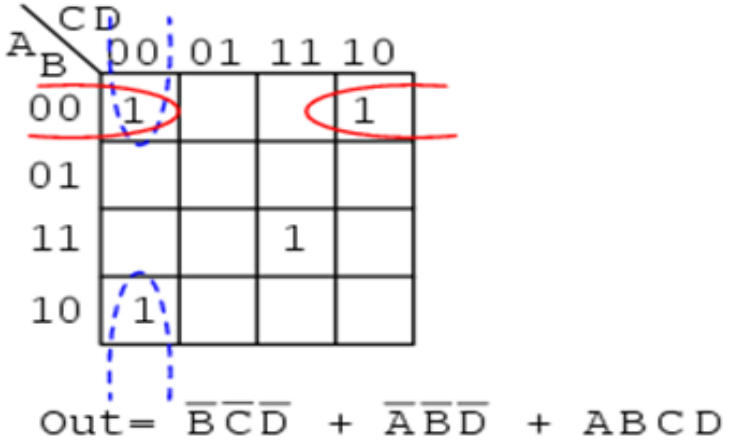
- You can only combine a power of 2 adjacent 1-squares. For e.g. 2, 4, 8, 16 squares. You cannot combine 3, 7 or 5 squares
- Right column and left column are adjacent; can be combined
- Top row and bottom column are adjacent; can be combined
- Many possible 2, 4, 8 groupings

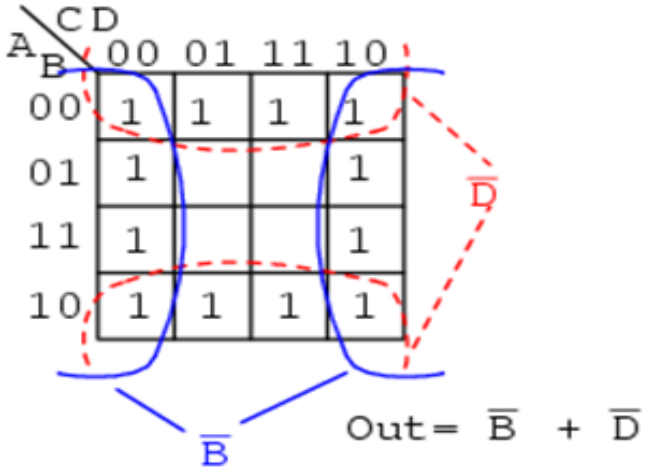


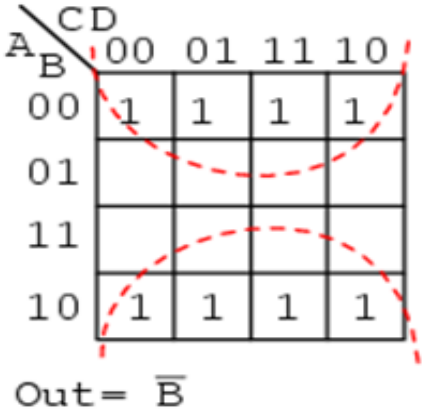
Four-Variable K-Map Examples

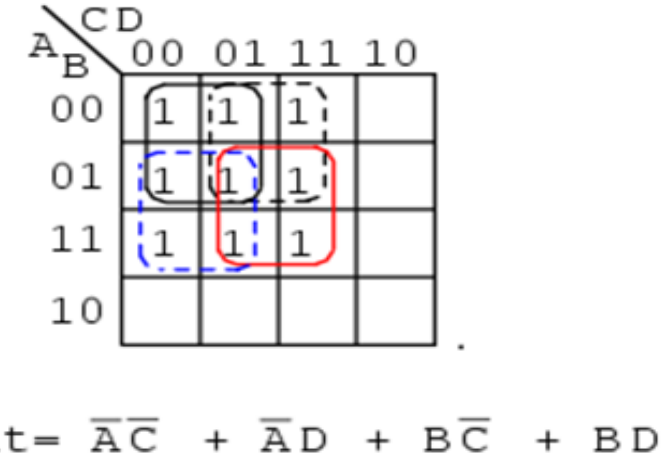










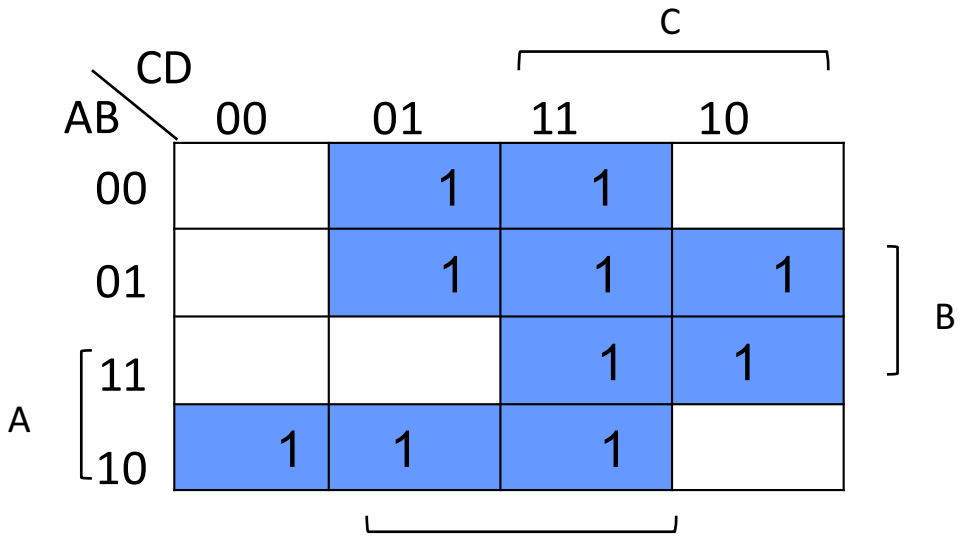


Four-Variable K-Map Examples

- Minimize the function $F(A,B,C,D)=\sum m(1,3,5,6,7,8,9,11,14,15)$

Four-Variable K-Map Examples

- Minimize the function F(A,B,C,D)=∑m(1,3,5,6,7,8,9,11,14,15)



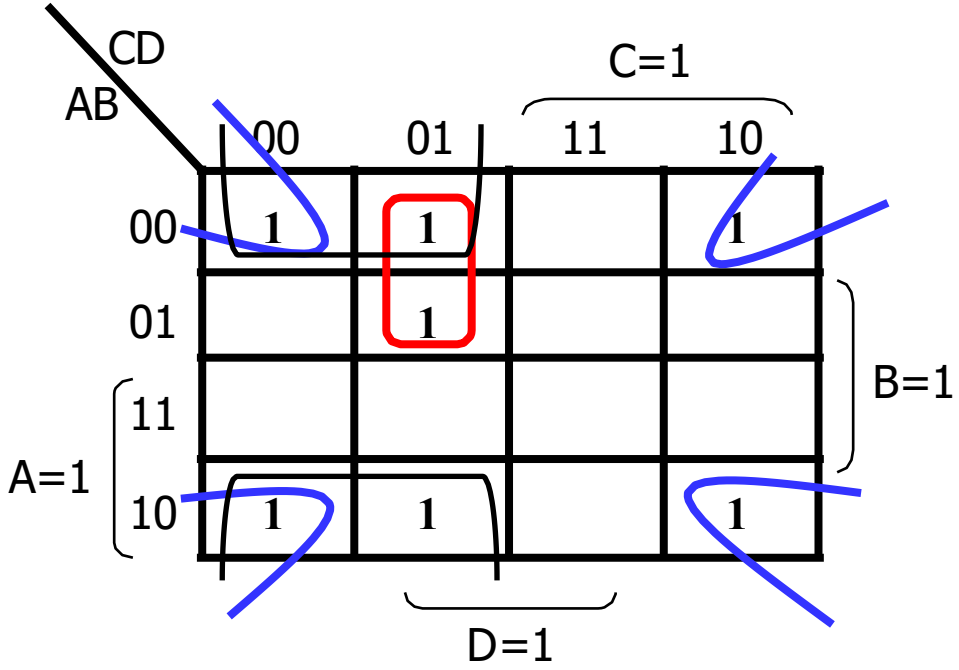
- Minimized F is ***F = CD + A'D + BC + AB'C'***

Four-Variable K-Map Examples

- $F(A,B,C,D) = \sum m(0,1,2,5,8,9,10)$

Four-Variable K-Map Examples

- F(A,B,C,D) = $\sum$m(0,1,2,5,8,9,10)

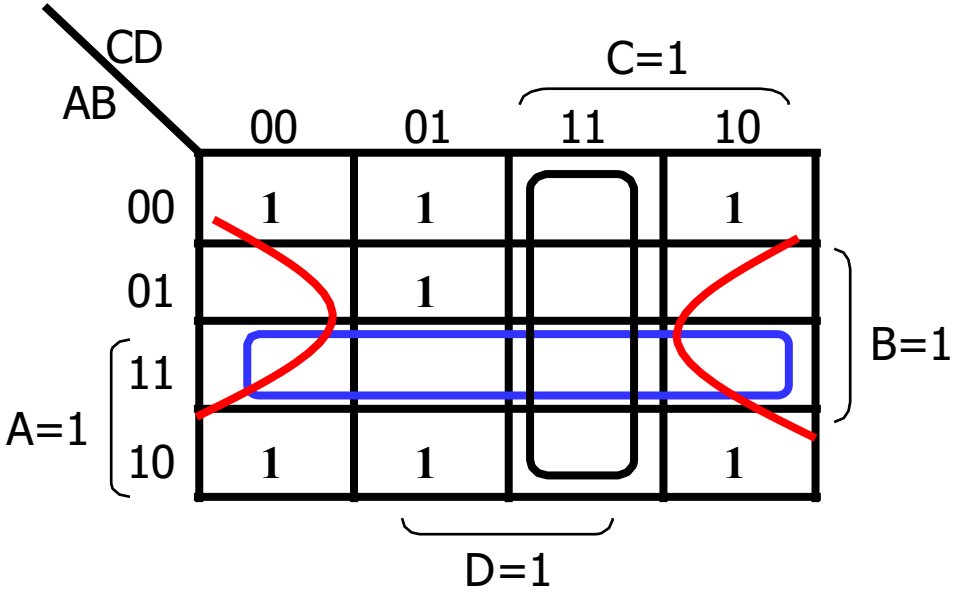


- Minimized F is ***F = B' D' + B' C' + A' C' D***

Four-Variable K-Map Examples

- Question: What if it is required to write the mininzed F in a product of sums (POS) form
  - Two methods:
    - Method 1: You already know one!
    - Method 2: Follow same rule as before but combine the ones of F', then find F.

- F(A,B,C,D)=∑m(0,1,2,5,8,9,10)
- Minimzed **F' = AB + CD + BD'**
- So, F'' = F = (A'+B')(C'+D')(B'+D)

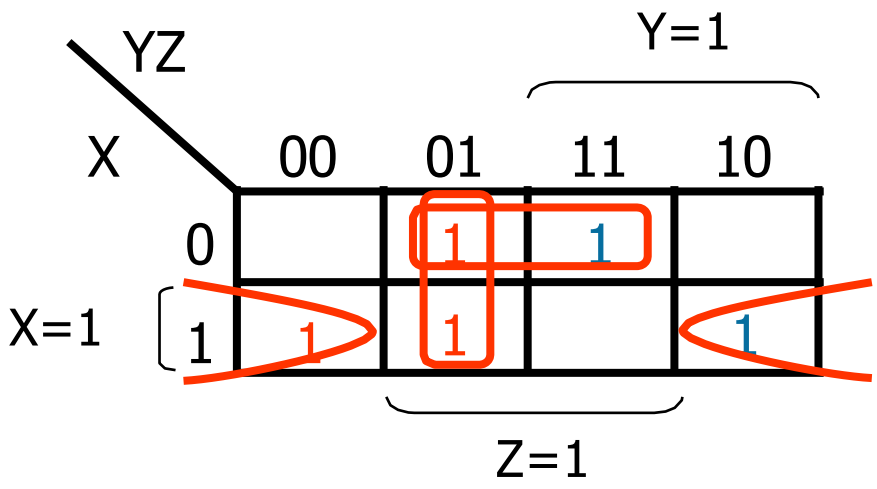


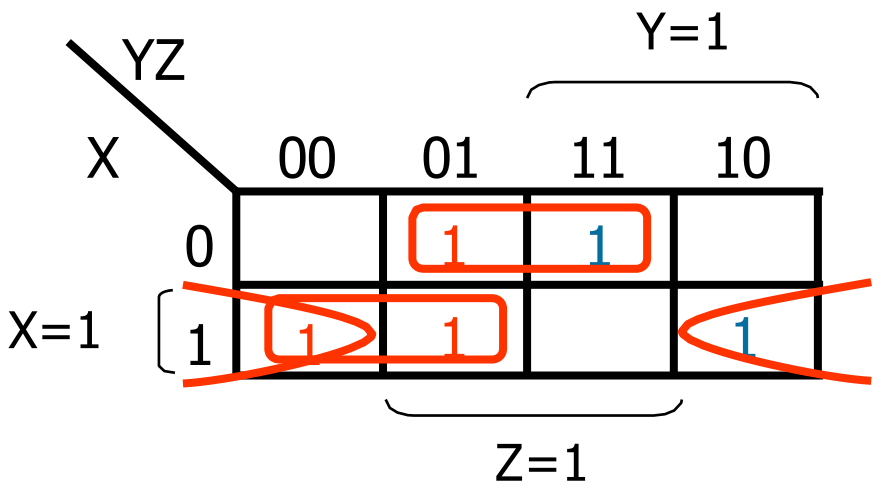
- A product term of a function is said to be an implicant.
- A Prime Implicant (PI) is a product term obtained by combining the maximum possible number of adjacent 1-squares in the map.
- A Prime Implicant is a product that we cannot remove any of its literals.
  - Largest groups of 1's
  - Not all prime implicants are needed!
- If a minterm is covered only by one prime implicant then this prime implicant is said to be an Essential Prime Implicant (EPI).

Examples of Types of Implicants

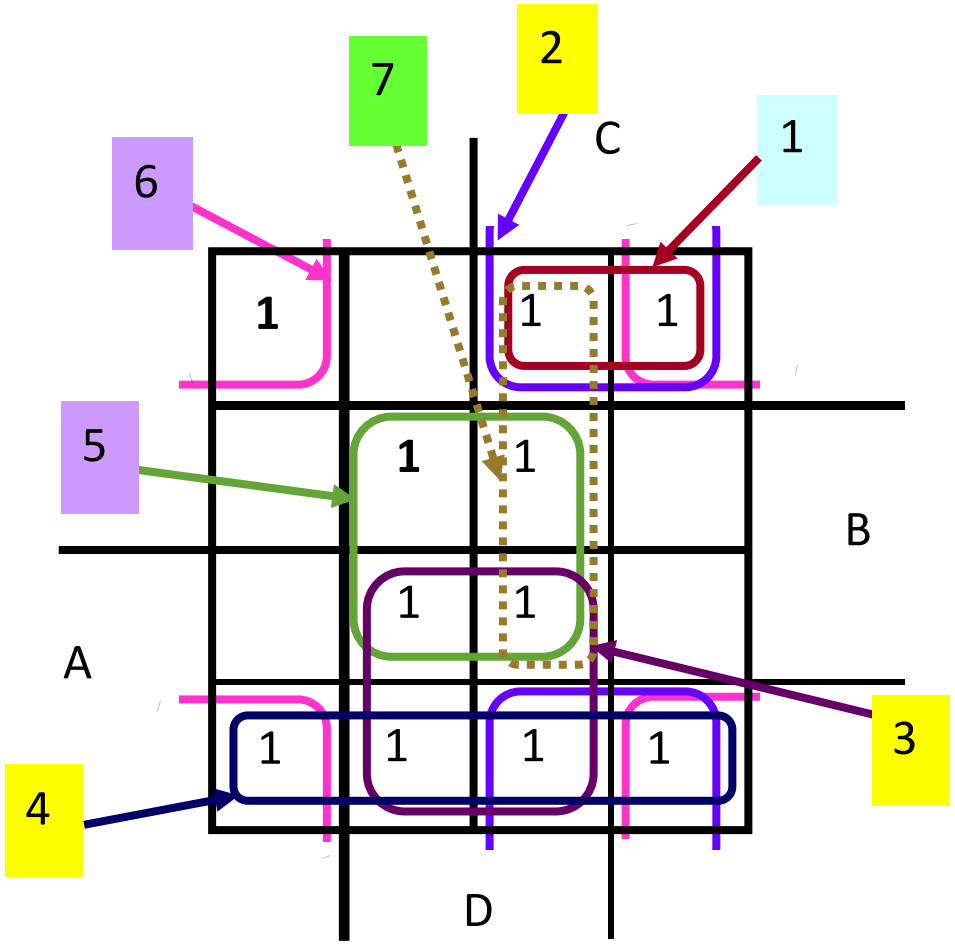
- Consider F(X,Y,Z) = (1,3,4,5,6), List all implicants, prime implicants and essential prime implicants

- Solution:
  - Implicants: XY'Z', XZ', XY', XY'Z, X'Y'Z, Y'Z, …
  - P.Is: XY', XZ', Y'Z, X'Z
  - E.P.Is: X'Z, XZ'





Examples of Types of Implicants



1 An implicant

2-4 Prime Implicants

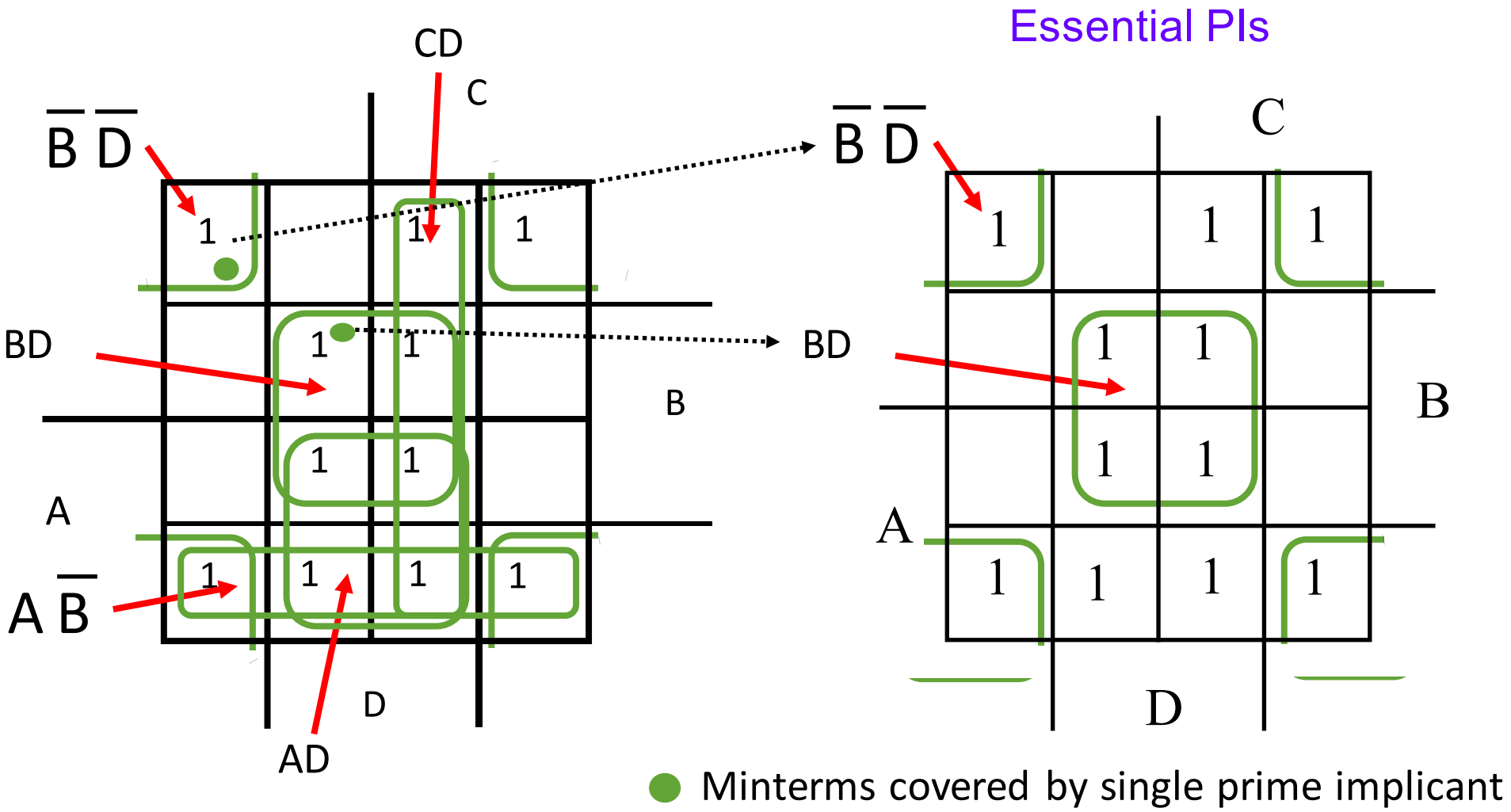
5-6 Essential Prime Implicants

Give a situation that makes implicant 1 a prime implicant

Does 7 represent a single product term? Is it an implicant? Why?

Minterm covered by only one prime implicant So minterm 5 is ……?

Examples of Types of Implicants

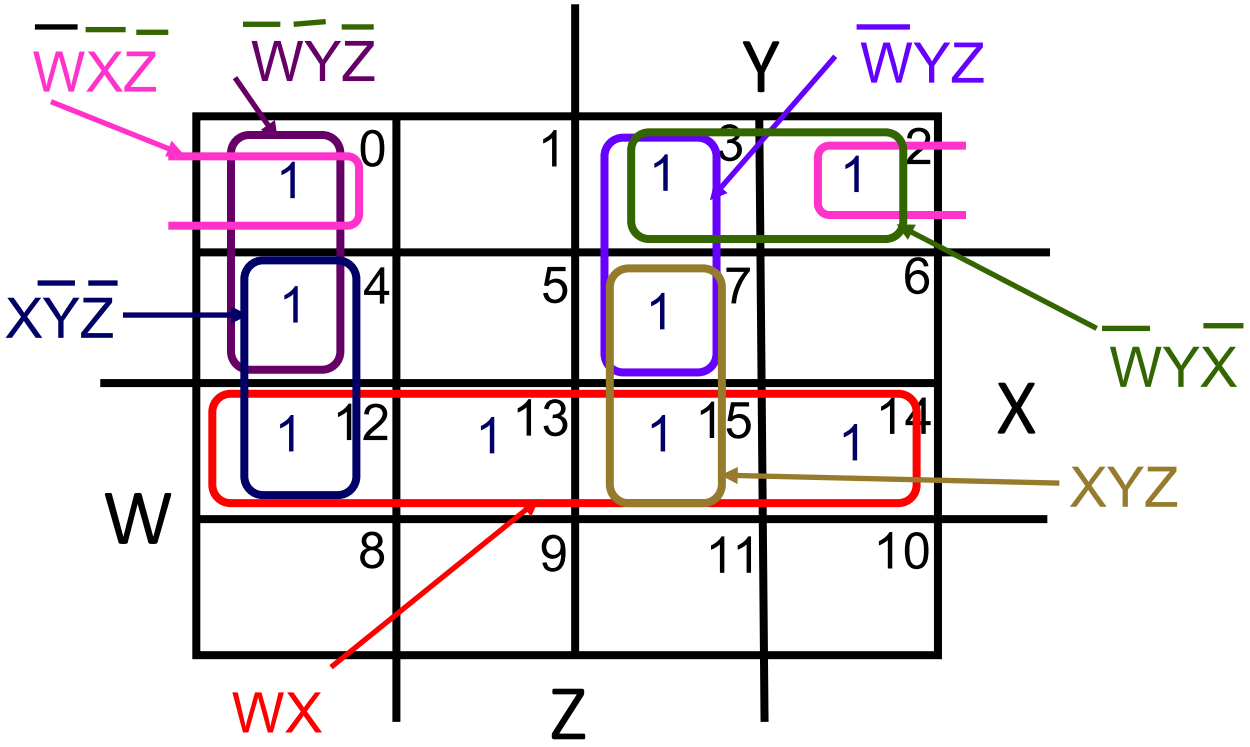


Examples of Types of Implicants

- Find all possible prime implicants for:

$$G(W,X,Y,Z) = \Sigma_m(0,2,3,4,7,12,13,14,15)$$

  - *Hint: There are seven prime implicants!*

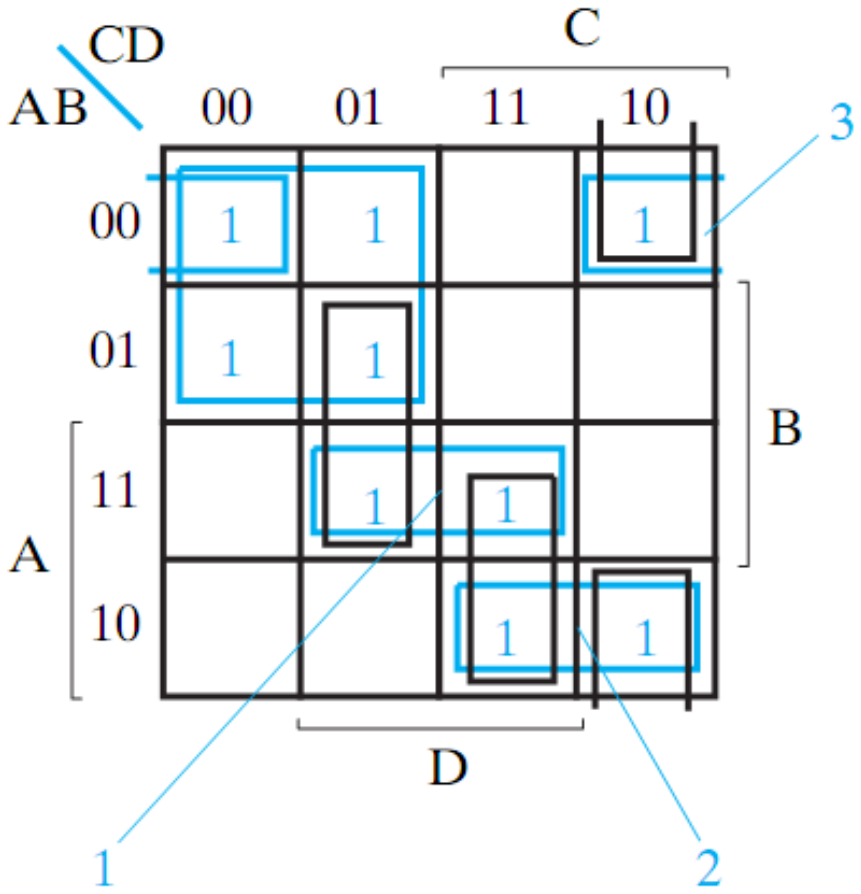


SOP Simplification procedure

- Identify all prime implicants covering 1's
  - Example: For a function of 3 variables, group all possible groups of 4, then groups of 2 that are not contained in groups of 4, then minterms that are not contained in a group of 4 or 2.
- Identify all essential prime implicants and select them.
- Check all minterms (1's) covered by essential prime implicants
- Repeat until all minterms (1's) are covered:
  - Select the prime implicant covering the largest uncovered minterms (1's)

SOP Simplification procedure

- Example: Simplify F(A, B, C, D) = ∑ m(0, 1, 2, 4, 5, 10,11,13, 15)
- Solution:
  - Only A'C' is E.P.I
  - For the remaining minterms:
  - Choose 1 and 2 (minimize overlap)
  - For m_2, choose either A'B'D' or B'CD'
  - *F = A'C' + ABD + AB'C + A'B'D'*



Don't Care Conditions

- In some cases, the function is not specified for certain combinations of input variables as 1 or 0.
- There are two cases in which it occurs:
  - The input combination never occurs.
  - The input combination occurs but we do not care what the outputs are in response to these inputs because the output will not be observed.
- In both cases, the outputs are called as unspecified and the functions having them are called as incompletely specified functions
- In most applications, we simply do not care what value is assumed by the function for unspecified minterms.

Don’t Care Conditions

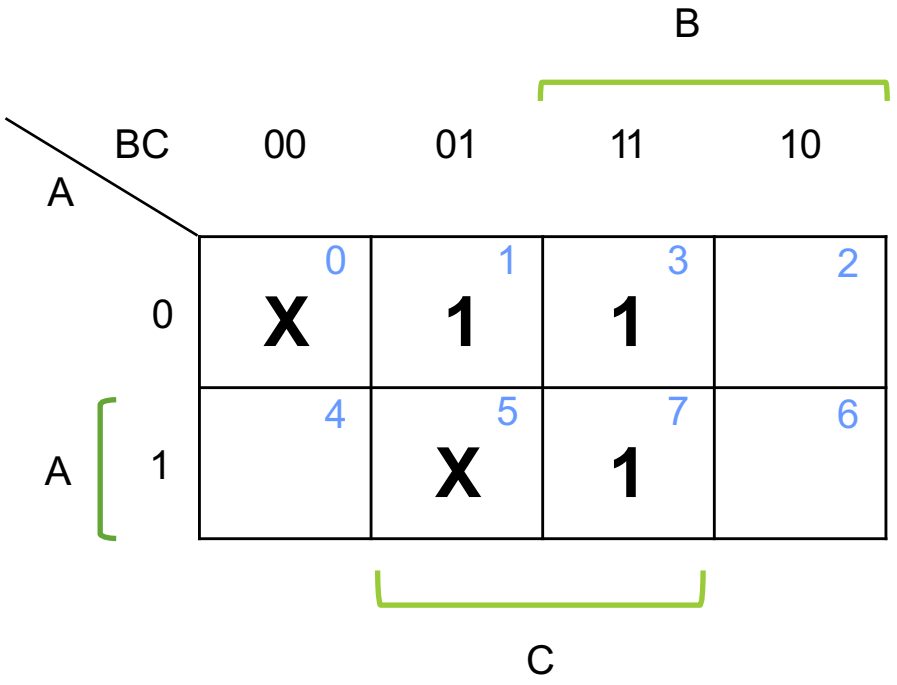
- Unspecified minterms of a function are called as don’t care conditions. They provide further simplification of the function, and they are denoted by X’s to distinguish them from 1’s and 0’s.
- In choosing adjacent squares to simplify the function in a map, the don’t care minterms can be assumed either 1 or 0, depending on which combination gives the simplest expression.
- A don’t care minterm need not be chosen at all if it does not contribute to produce a larger implicant.

SOP Simplification procedure using Don't Cares

- Identify all prime implicants covering 1's & X's
  - Each prime implicant must contain at least a single 1
- Identify all essential prime implicants and select them.
  - An essential prime implicant must be the only implicant covering at least a "1".
- Check all 1's covered by essential prime implicants
- Repeat until all 1's are covered:
  - Select the prime implicant covering the largest uncovered 1's.

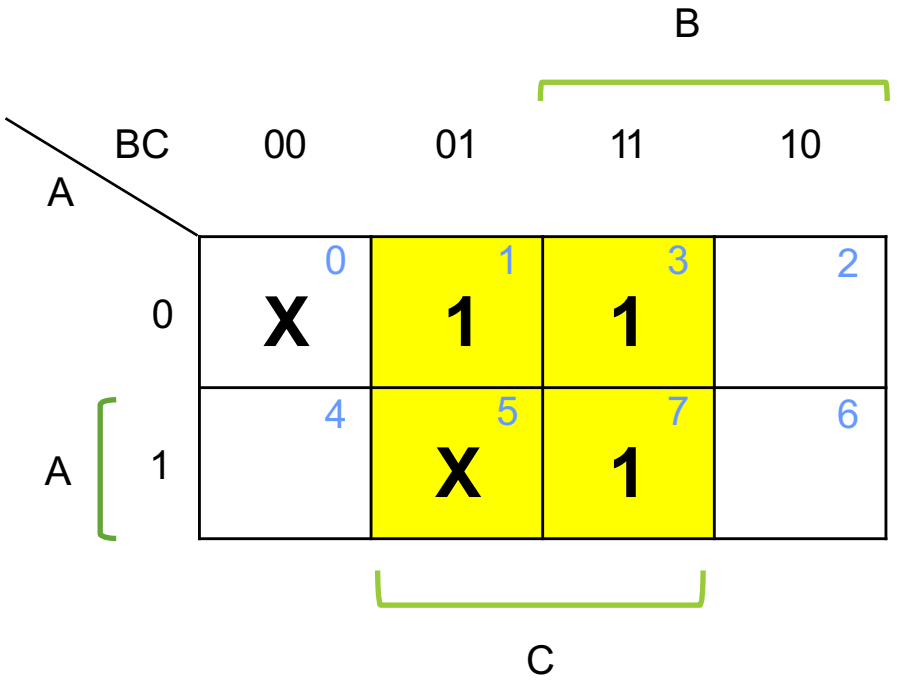
SOP Simplification procedure using Don't Cares

- Example: Minimize F = $\sum$m(1, 3, 7) + $\sum$d(0, 5)
- Circle the X's that help get bigger groups of 1's (or 0's if POS).
- Don't circle the X's that don't help.



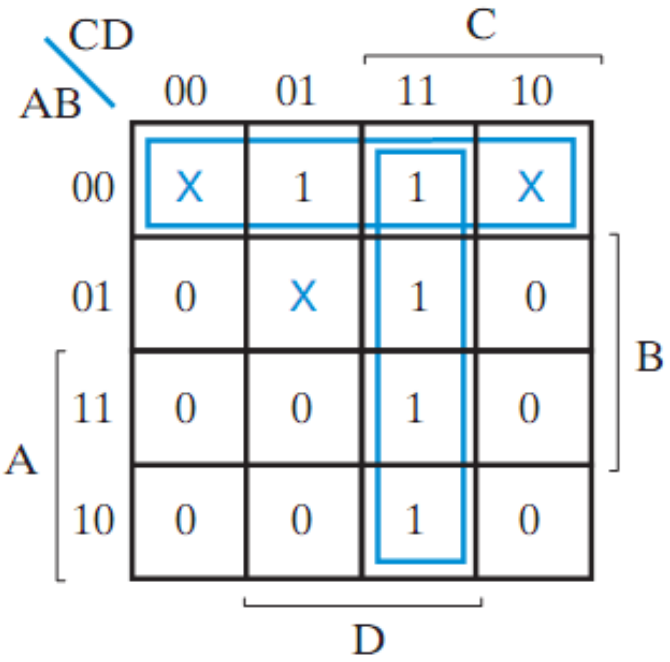
SOP Simplification procedure using Don't Cares

- Example: Minimize $F = \sum m(1, 3, 7) + \sum d(0, 5)$
- Circle the X's that help get bigger groups of 1's (or 0's if POS).
- Don't circle the X's that don't help.
- ***F = C***

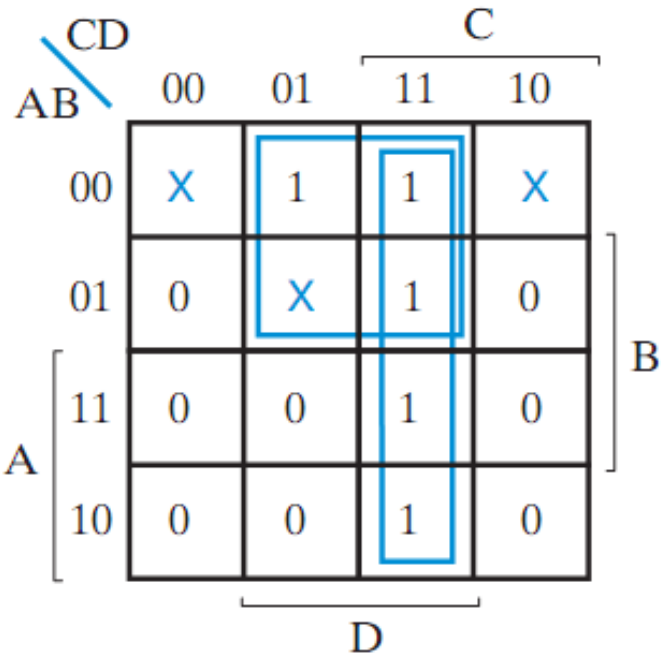


SOP Simplification procedure using Don't Cares

- Example: Minimize $F(A, B, C, D) = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 5)$
  - Two possible solutions!
  - Both acceptable
  - All 1's covered



(a) $F = CD + \overline{A}\,\overline{B}$



(b) $F = CD + \overline{A}D$

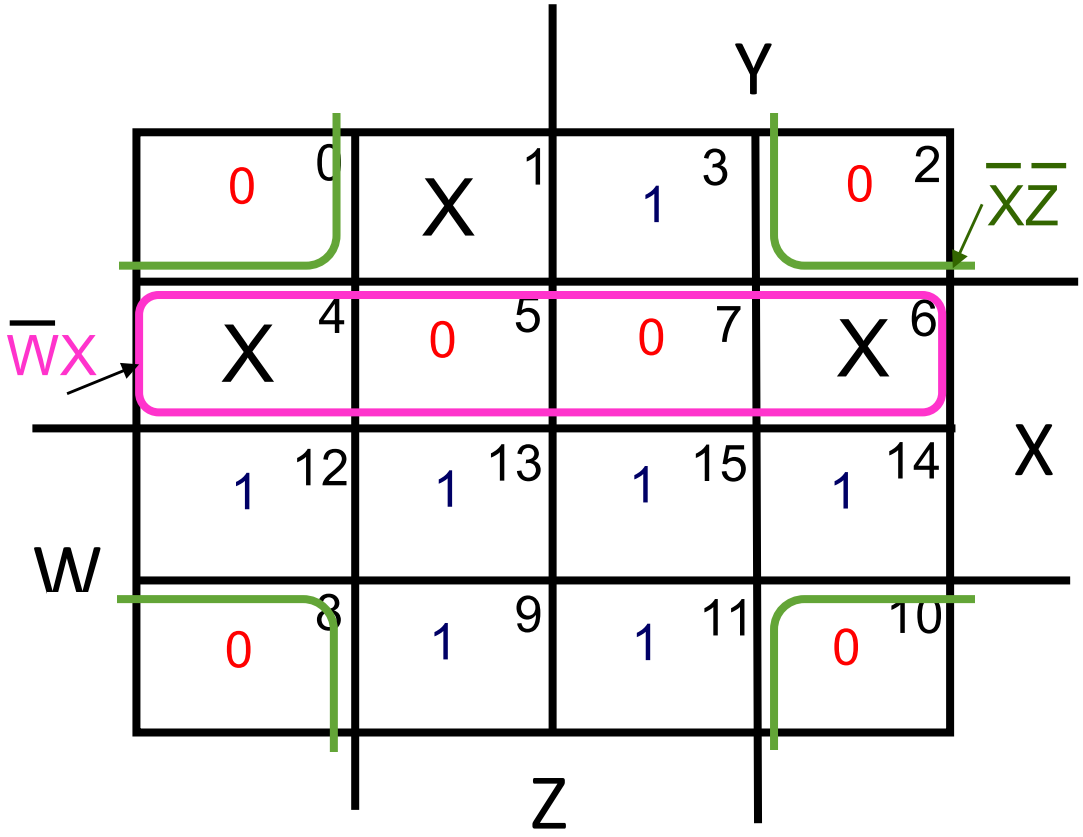
POS Simplification

- Until now we have derived simplified Boolean functions from the maps in SOP form. Procedure for deriving simplified Boolean functions POS is slightly different. Instead of making groups of 1's, make the groups of 0's.
- Since the simplified expression obtained by making group of 1's of the function (say F) is always in SOP form. Then the simplified function obtained by making group of 0's of the function will be the complement of the function (i.e., F') in SOP form.
- Applying DeMorgan's theorem to F' (in SOP) will give F in POS form.

POS Simplification

- Find the optimum POS solution for F, given:

$$F(A,B,C,D) = \Sigma_m(3,9,11,12,13,14,15) + \Sigma d\,(1,4,6)$$



5-Variable K-Maps

- 32 minterms require 32 squares in the K-map
- Minterms 0-15 belong to the squares with variable A=0, and minterms 16-32 belong to the squares with variable A=1
- Each square in A' is also adjacent to a square in A (one is above the other)
- Minterm 4 is adjacent to 20, and minterm 15 is to 31

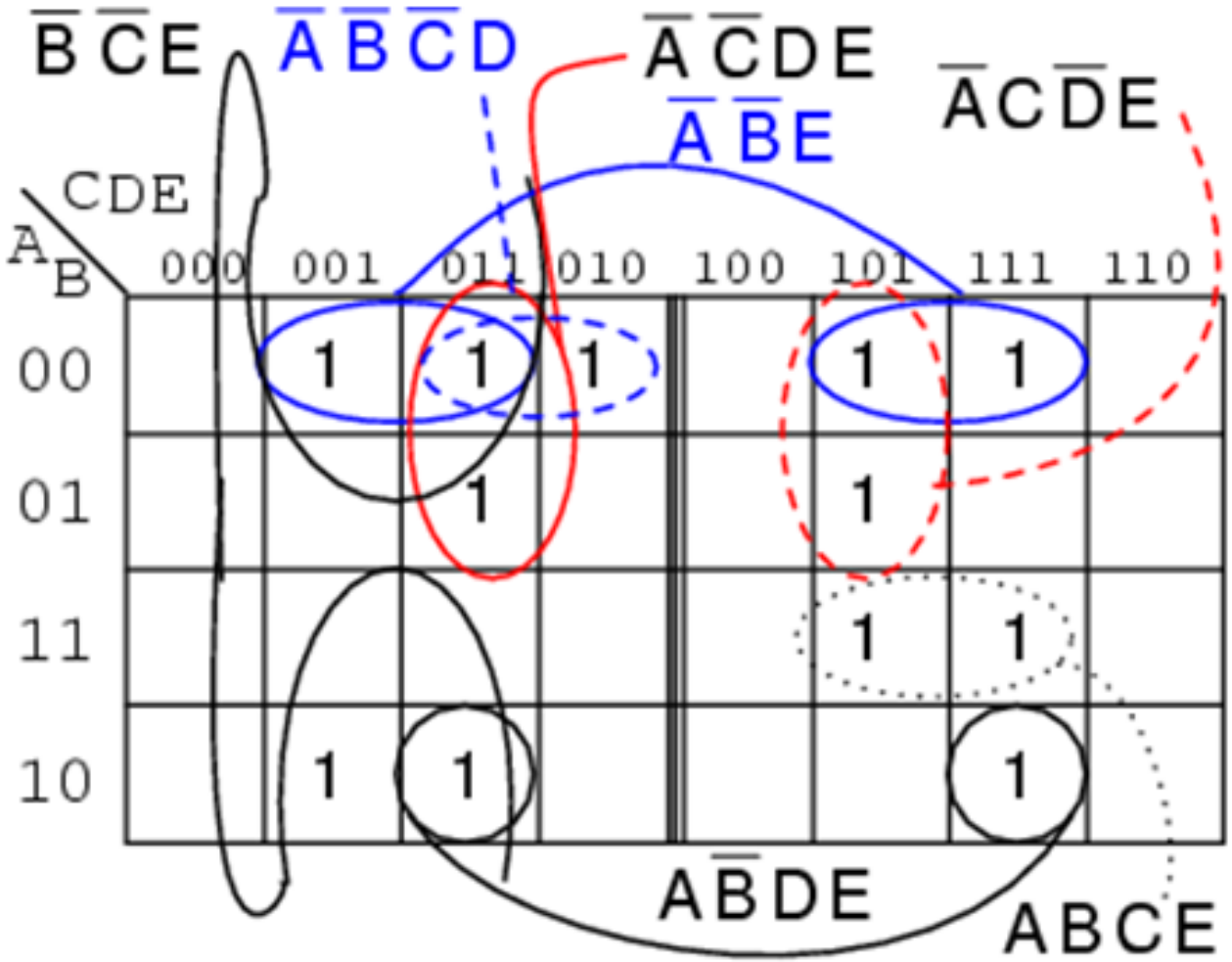
A=0

BC \ DE	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}

A=1

00	01	11	10
m_{16}	m_{17}	m_{19}	m_{18}
m_{20}	m_{21}	m_{23}	m_{22}
m_{28}	m_{29}	m_{31}	m_{30}
m_{24}	m_{25}	m_{27}	m_{26}

5-Variable K-Maps



Conclusion

- A K-Map is simply a folded truth table, where physical adjacency implies logical adjacency
- K-Maps are most commonly used hand method for logic minimization.