Logic and Computer Design Fundamentals

Chapter 2 – Combinational Logic Circuits

Part 2 – Circuit Optimization

Charles Kime & Thomas Kaminski

(Hyperlinks are active in View Show mode)

Updated by Dr. Waleed Dweik

Overview

- **Part 1 – Gate Circuits and Boolean Equations**
  - **Binary Logic and Gates**
  - **Boolean Algebra**
  - **Standard Forms**
- **Part 2 – Circuit Optimization**
  - **Two-Level Optimization**
  - **Map Manipulation**
- **Part 3 – Additional Gates and Circuits**
  - **Other Gate Types**
  - **Exclusive-OR Operator and Gates**
  - **High-Impedance Outputs**

Circuit Optimization

- Goal: To obtain the simplest implementation for a given function
- Optimization is a more formal approach to simplification that is performed using a specific procedure or algorithm
- Optimization requires a cost criterion to measure the simplicity of a circuit
- Distinct cost criteria we will use:
  - **Literal cost (L)**
  - **Gate input cost (G)**
  - **Gate input cost with NOTs (GN)**

Literal Cost

- ***Literal:*** a variable or its complement
- ***Literal cost (L):*** the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram
- Examples:
  - $F = BD + AB'C + AC'D'$
    - $L = 8$ (Minimum cost → Best solution)
  - $F = BD + AB'C + AB'D' + ABC'$
    - $L = 11$
  - $F = (A + B)(A + D)(B + C + D')(B' + C' + D)$
    - $L = 10$

Gate Input Cost

- ***Gate input cost (G):*** the number of inputs to the gates in the implementation corresponding exactly to the given equation or equations. (***G: inverters not counted, GN: inverters counted***)
- For SOP and POS equations, it can be found from the equation(s) by finding the sum of:
  - **All literal appearances**
  - **The number of terms excluding single literal terms,(G) and**
  - **optionally, the number of distinct complemented single literals (GN).**
- Examples:
  - $F = BD + AB'C + AC'D'$
    - $G = 11\,, GN = 14$ (Minimum cost → Best solution)
  - $F = BD + AB'C + AB'D' + ABC'$
    - $G = 15\,, GN = 18$
  - $F = (A + B)(A + D)(B + C + D')(B' + C' + D)$
    - $G = 14\,, GN = 17$

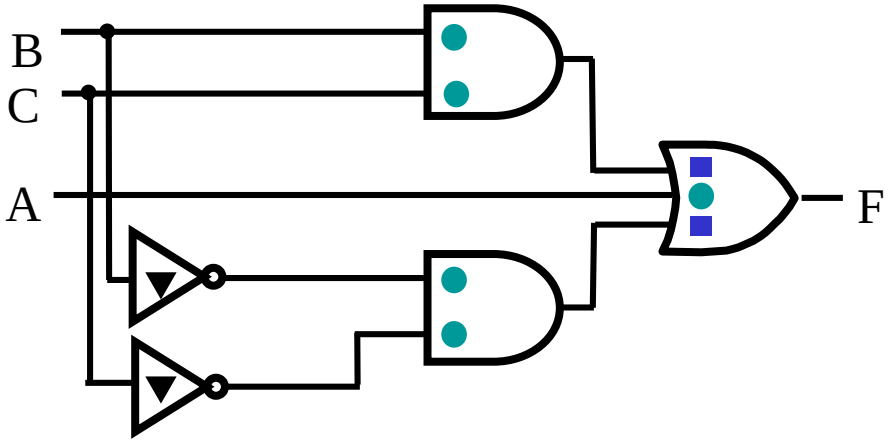
Cost Criteria (continued)

- **Example 1:**
- **$F = A + BC + \overline{B}\,\overline{C}$**

GN = G + 2 = 9

L = 5

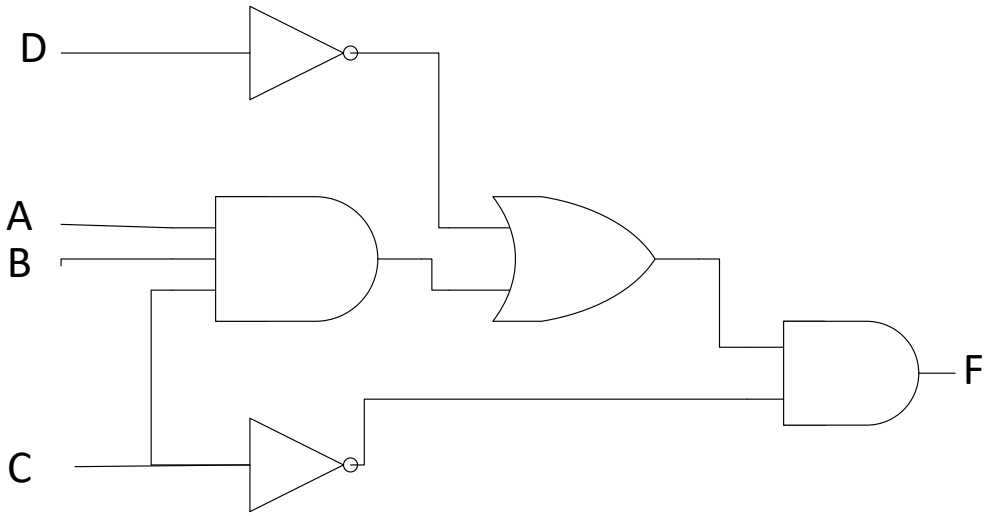
G = L + 2 = 7



- **L (literal count) counts the AND inputs and the single literal OR input.**
- **G (gate input count) adds the remaining OR gate inputs**
- **GN(gate input count with NOTs) adds the inverter inputs**

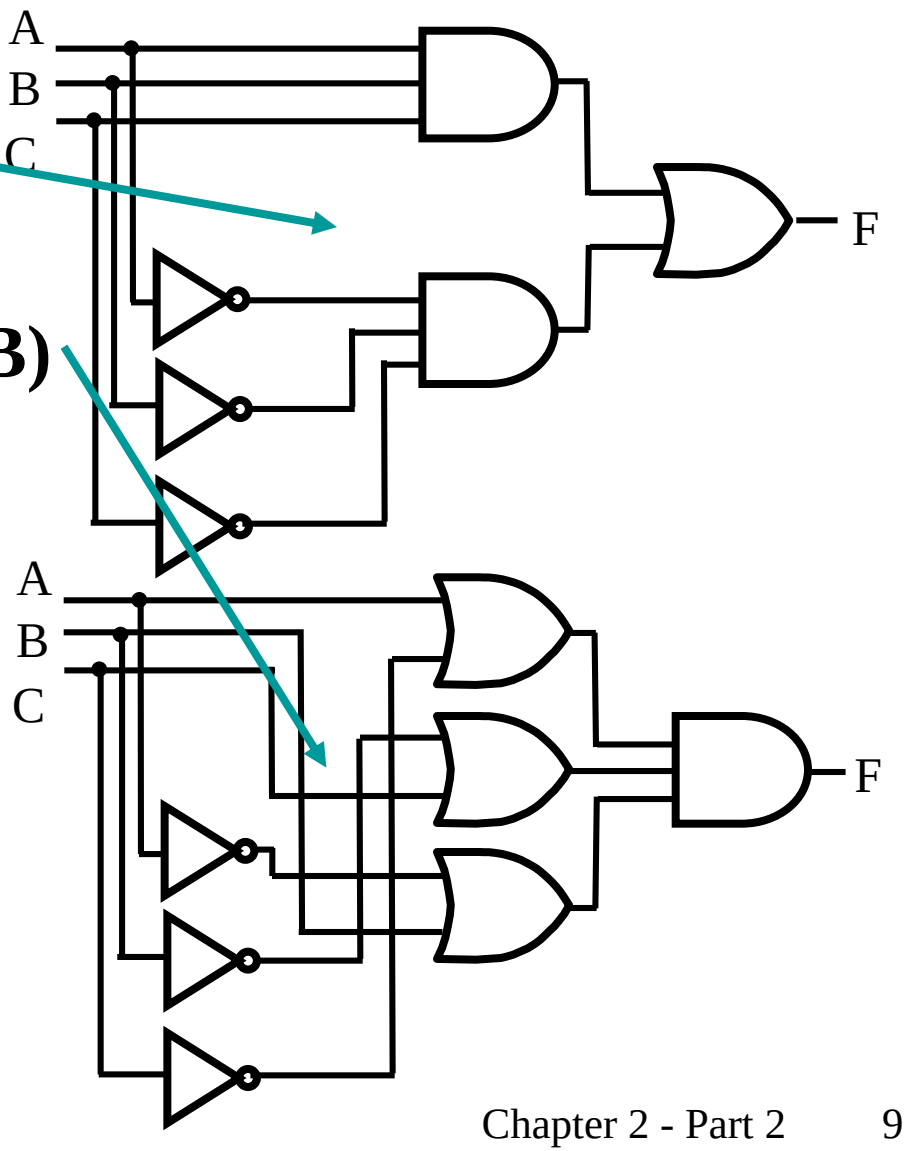
Cost Criteria (continued)

- **Example 2:**
- $F = (A, B, C, D) = (ABC + D').C'$
  - $L = 5$
  - $G = 5 + 2 = 7$
  - $GN = 7 + 2 = 9$



Cost Criteria (continued)

- **Example 3:**
- **$F = A\,B\,C + \overline{A}\,\overline{B}\,\overline{C}$**
- **L = 6, G = 8, GN = 11**
- **$F = (A + \overline{C})(\overline{B} + C)(\overline{A} + B)$**
- **L = 6 , G = 9, GN = 12**
- **<u>Same</u> function and <u>same</u> literal cost**
- **But first circuit has <u>better</u> gate input count and <u>better</u> gate input count with NOTs**
- **Select it!**



Boolean Function Optimization

- Minimizing the gate input (or literal) cost of a (a set of) Boolean equation(s) reduces circuit cost
- We choose gate input cost
- Boolean Algebra and graphical techniques are tools to minimize cost criteria values
- Some important questions:
  - When do we stop trying to reduce the cost?
  - Do we know when we have a minimum cost?
- Treat optimum or near-optimum cost functions for two-level (SOP and POS) circuits
- Introduce a graphical technique using Karnaugh maps (K-maps, for short)

Karnaugh Maps (K-map)

- A K-map is a collection of squares
  - Graphical representation of the truth table
  - Each square represents a minterm, or a maxterm, or a row in the truth table
  - For n-variable, there are 2^n squares
  - The collection of squares is a graphical representation of a Boolean function
  - Adjacent squares differ in the value of one variable
  - Alternative algebraic expressions for the same function are derived by recognizing patterns of squares

Some Uses of K-Maps

- Finding optimum or near optimum
  - SOP and POS standard forms, and
  - two-level AND/OR and OR/AND circuit implementations

  for functions with small numbers of variables
- Visualizing concepts related to manipulating Boolean expressions, and
- Demonstrating concepts used by computer-aided design programs to simplify large circuits

Two Variable Maps

- **A 2-variable Karnaugh Map:**
  - Note that minterm m_0 and minterm m_1 are "adjacent" and differ in the value of the variable y
  - Similarly, minterm m_0 and minterm m_2 differ in the x variable
  - Also, m_1 and m_3 differ in the x variable as well
  - Finally, m_2 and m_3 differ in the value of the variable y

	$\mathbf{y = 0}$	$\mathbf{y = 1}$
$\mathbf{x = 0}$	$m_0 = \bar{x}\bar{y}$	$m_1 = \bar{x}y$
$\mathbf{x = 1}$	$m_2 = x\bar{y}$	$m_3 = xy$

K-Map and Truth Tables

- The K-Map is just a different form of the truth table
- Example: Two variable function
  - We choose a,b,c and d from the set {0,1} to implement a particular function, $F(x, y)$

Input Values (x, y)	F(x, y)
0 0	a
0 1	b
1 0	c
1 1	d

Truth Table

	y = 0	y = 1
x = 0	*a*	*b*
x = 1	*c*	*d*

K-Map

K-Map Function Representation

- Example: $F(x, y) = x$

$F(x, y) = x$	y = 0	y = 1
x = 0	0	0
x = 1	1	1

- For function $F(x, y)$, the two adjacent cells containing 1's can be combined using the Minimization Theorem:

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

K-Map Function Representation

- Example: $G(x, y) = x + y$

$G(x, y) = x + y$	y = 0	y = 1
x = 0	0	1
x = 1	1	1

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

$$G(x, y) = (x\bar{y} + xy) + (\bar{x}y + xy)$$

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

Three Variable Maps

- **A three-variable K-map:**

	$yz = 00$	$yz = 01$	$yz = 11$	$yz = 10$
$x = 0$	m_0	m_1	m_3	m_2
$x = 1$	m_4	m_5	m_7	m_6

- **Where each minterm corresponds to the product terms:**

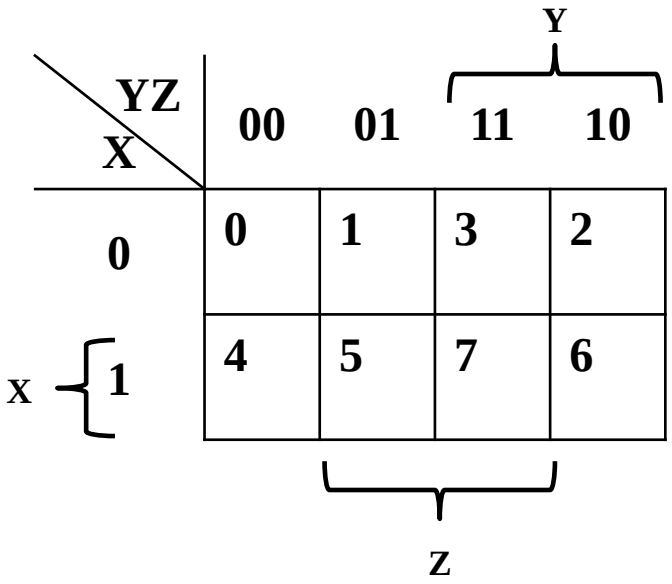
	$yz = 00$	$yz = 01$	$yz = 11$	$yz = 10$
$x = 0$	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
$x = 1$	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

- ***Note that if the binary value for an index differs in one bit position, the minterms are adjacent on the K-Map***

Alternative Map Labeling

- Map use largely involves:
  - Entering values into the map, and
  - Reading off product terms from the map
- Alternate labelings are useful:

	$\overline{Y}$		Y	
$\overline{X}$	0	1	3	2
X	4	5	7	6
	$\overline{Z}$	Z		$\overline{Z}$



Example Functions

- By convention, we represent the minterms of F by a "1" in the map and leave the minterms of $\bar{F}$ blank
- Example:
  - $F(x, y, z) = \sum_m(2,3,4,5)$

			Y	
	0	1	3 1	2 1
X	4 1	5 1	7	6
		Z		

- Example:
  - $G(a, b, c) = \sum_m(3,4,6,7)$

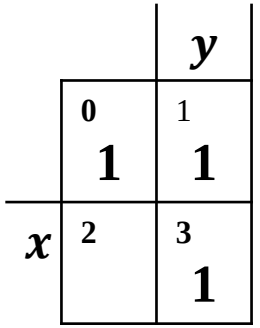
			b	
	0	1	3 1	2
a	4 1	5	7 1	6 1
		c		

- Learn the locations of the 8 indices based on the variable order shown (X, most significant and Z, least significant) on the map boundaries

Steps for using K-Maps to Simplify Boolean Functions

- Enter the function on the K-Map
  - Function can be given in truth table, shorthand notation, SOP,…etc
  - Example:
    - $F(x, y) = \bar{x} + xy$
    - $F(x, y) = \sum_m(0,1,3)$

x	y	F(x, y)
0	0	1
0	1	1
1	0	0
1	1	1



- Combining squares for simplification
  - Rectangles that include power of 2 squares {1, 2, 4, 8, …}
  - Goal: Fewest rectangles that cover all 1's → as large as possible
- Determine if any rectangle is not needed
- Read-off the SOP terms

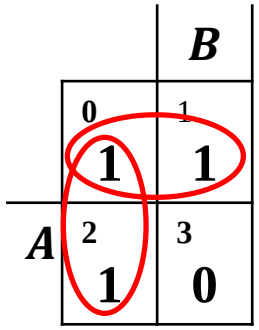
Combining Squares

- By combining squares, we reduce number of literals in a product term, reducing the literal cost, thereby reducing the other two cost criteria
- On a 2-variable K-Map:
  - One square represents a minterm with two variables
  - Two adjacent squares represent a product term with one variable
  - Four “adjacent” terms is the function of all ones (no variables) = 1.
- On a 3-variable K-Map:
  - One square represents a minterm with three variables
  - Two adjacent squares represent a product term with two variables
  - Four “adjacent” terms represent a product term with one variable
  - Eight “adjacent” terms is the function of all ones (no variables) = 1.

Example: Combining Squares

- Example: $F(A,B) = \sum_m(0,1,2)$

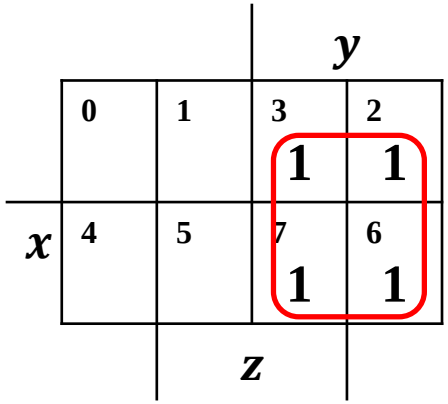
$$F(A,B) = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$$



- Using Distributive law
  - $F(A,B) = \bar{A} + A\bar{B}$
- Using simplification theorem
  - $F(A,B) = \bar{A} + \bar{B}$
- **Thus, every two adjacent terms that form a 2×1 rectangle correspond to a product term with one variable**

Example: Combining Squares

- Example: $F(x, y, z) = \sum_m(2,3,6,7)$
- $F(x, y, z) = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz$
- Using Distributive law
  - $F(x, y, z) = \bar{x}y + xy$
- Using Distributive law again
  - $F(x, y, z) = y$
- **Thus, the four adjacent terms that form a 2×2 square correspond to the term "y"**



Three-Variable Maps

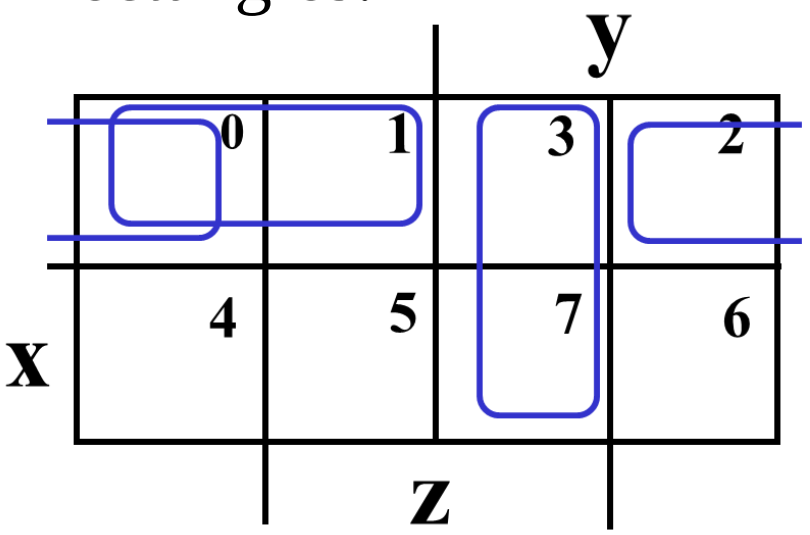
- Reduced literal product terms for SOP standard forms correspond to rectangles on K-maps containing cell counts that are powers of 2
- Rectangles of 2 cells represent 2 adjacent minterms
- Rectangles of 4 cells represent 4 minterms that form a “pairwise adjacent” ring
- Rectangles can contain non-adjacent cells as illustrated by the “pairwise adjacent” ring above

Three-Variable Maps

- Example shapes of 2-cell rectangles:

Three-Variable Maps

- Example shapes of 2-cell rectangles:



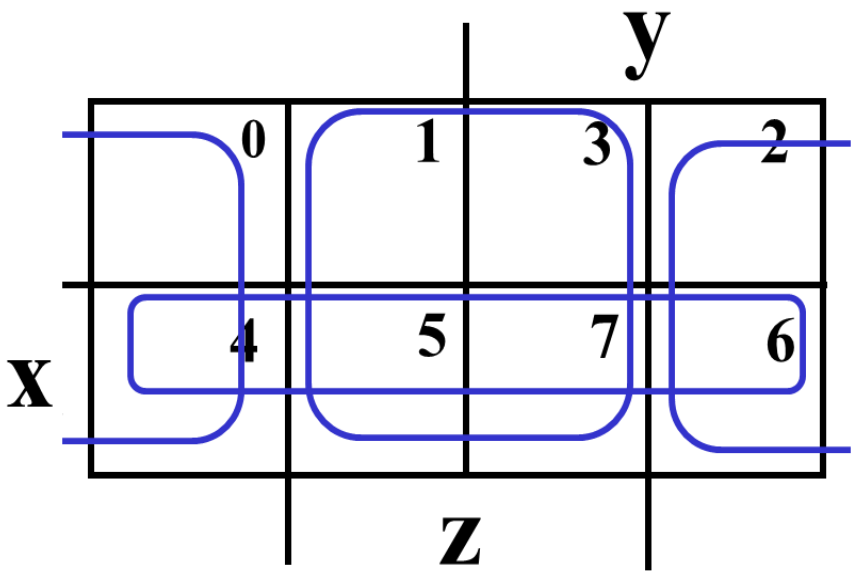
- Read-off the product terms for the rectangles shown:
  - $Rect(0,1) = \bar{X}\bar{Y}$
  - $Rect(0,2) = \bar{X}\bar{Z}$
  - $Rect(3,7) = YZ$

Three-Variable Maps

- Example shapes of 4-cell Rectangles:

Three-Variable Maps

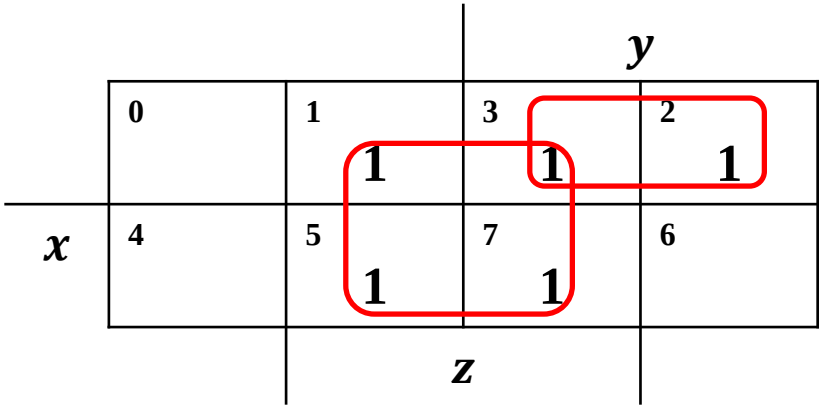
- Example shapes of 4-cell Rectangles:



- Read off the product terms for the rectangles shown:
  - $Rect(1,3,5,7) = Z$
  - $Rect(0,2,4,6) = \bar{Z}$
  - $Rect(4,5,6,7) = X$

Three Variable Maps

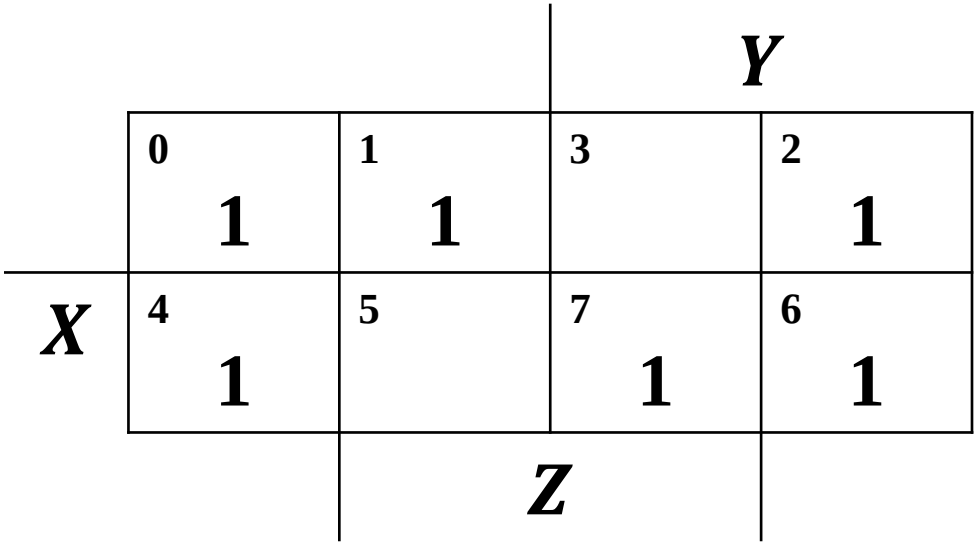
- K-maps can be used to simplify Boolean functions by systematic methods. Terms are selected to cover the "1s"in the map.
- Example: Simplify $F(x, y, z) = \sum_m(1,2,3,5,7)$



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

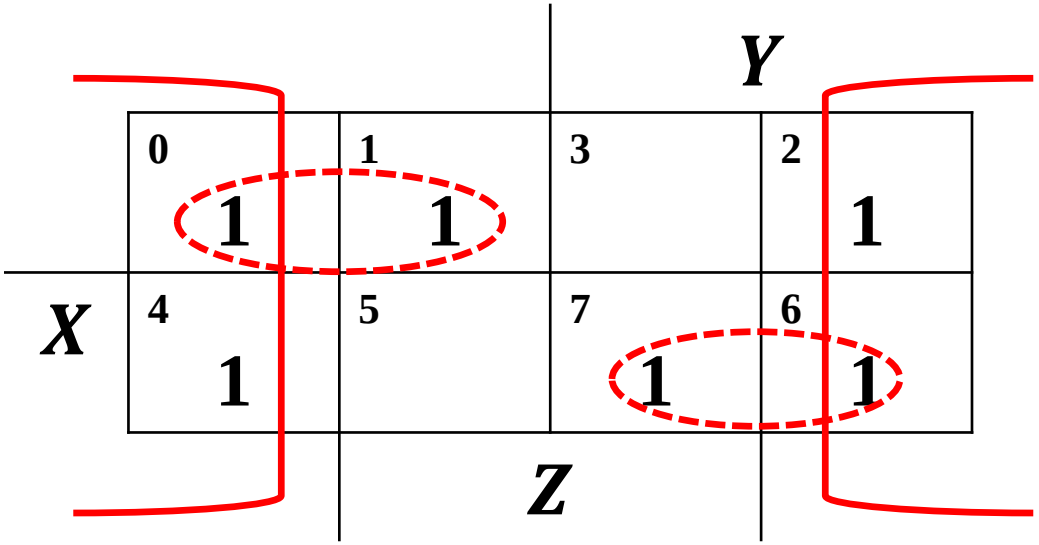
Three-Variable Map Simplification

- Use a K-map to find an optimum SOP equation for $F(X, Y, Z) = \sum_m(0,1,2,4,6,7)$



Three-Variable Map Simplification

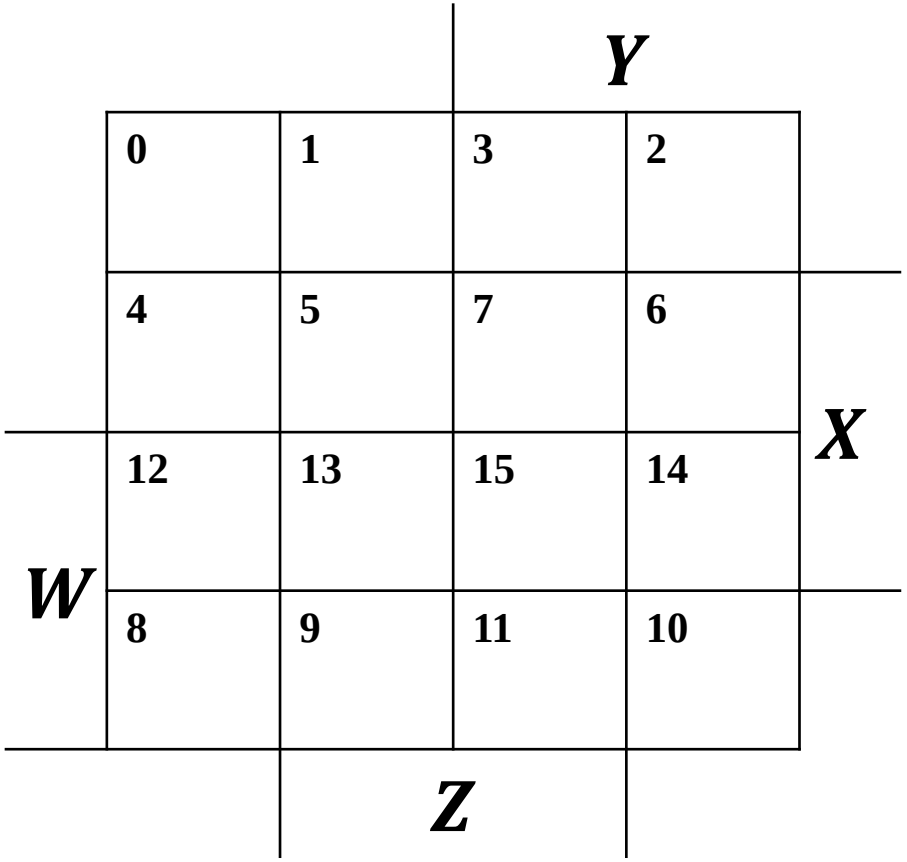
- Use a K-map to find an optimum SOP equation for $F(X,Y,Z) = \sum_m(0,1,2,4,6,7)$



$$F(X,Y,Z) = \bar{Z} + \bar{X}\bar{Y} + XY$$

Four Variable Maps

- Map and location of minterms $F(W, X, Y, Z)$:

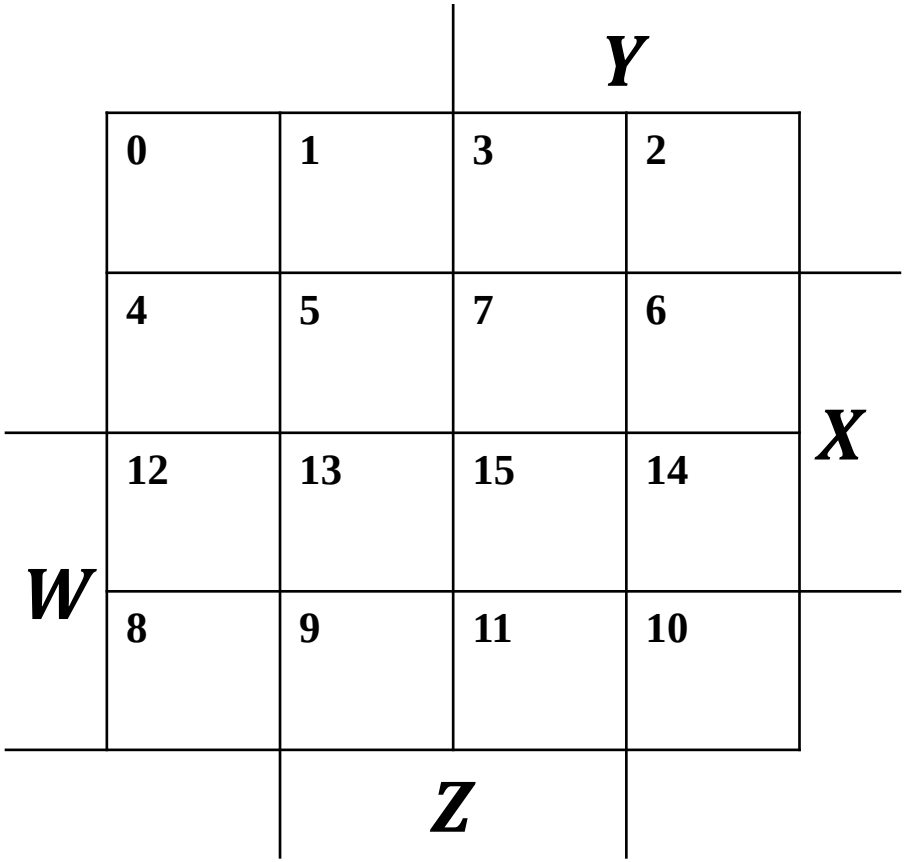


Four Variable Terms

- Four variable maps can have rectangles corresponding to:
  - A single 1: 4 variables (i.e. Minterm)
  - Two 1's: 3 variables
  - Four 1's: 2 variables
  - Eight 1's: 1 variable
  - Sixteen 1's: zero variables (function of all ones)

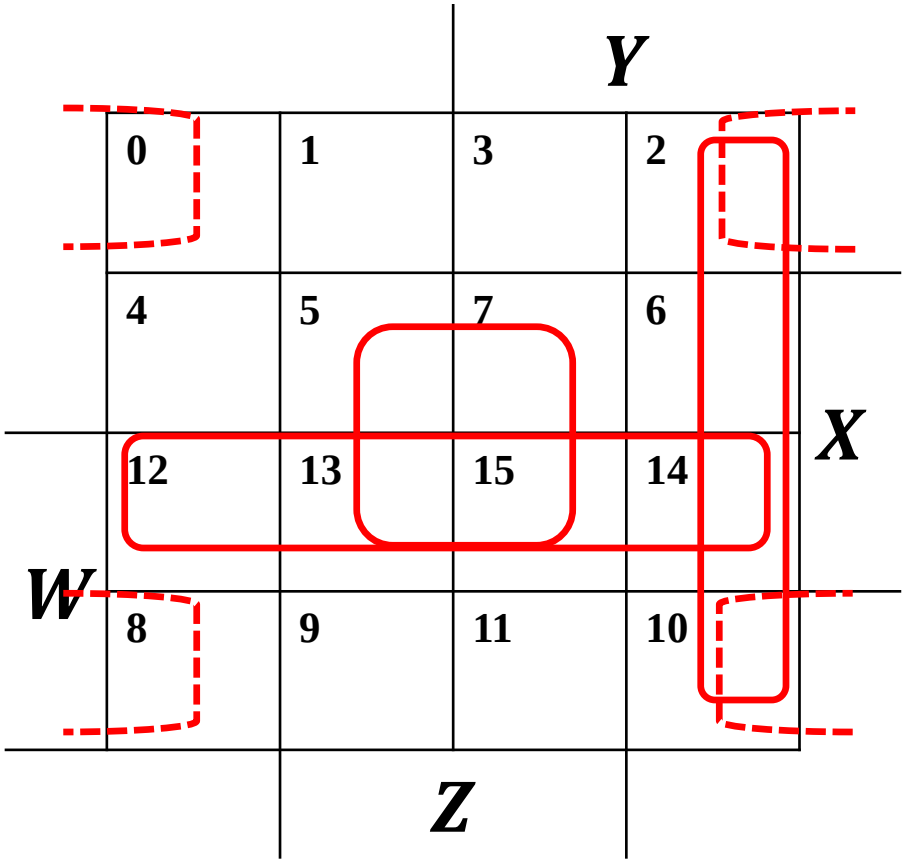
Four-Variable Maps

- Example shapes of 4-cell rectangles:



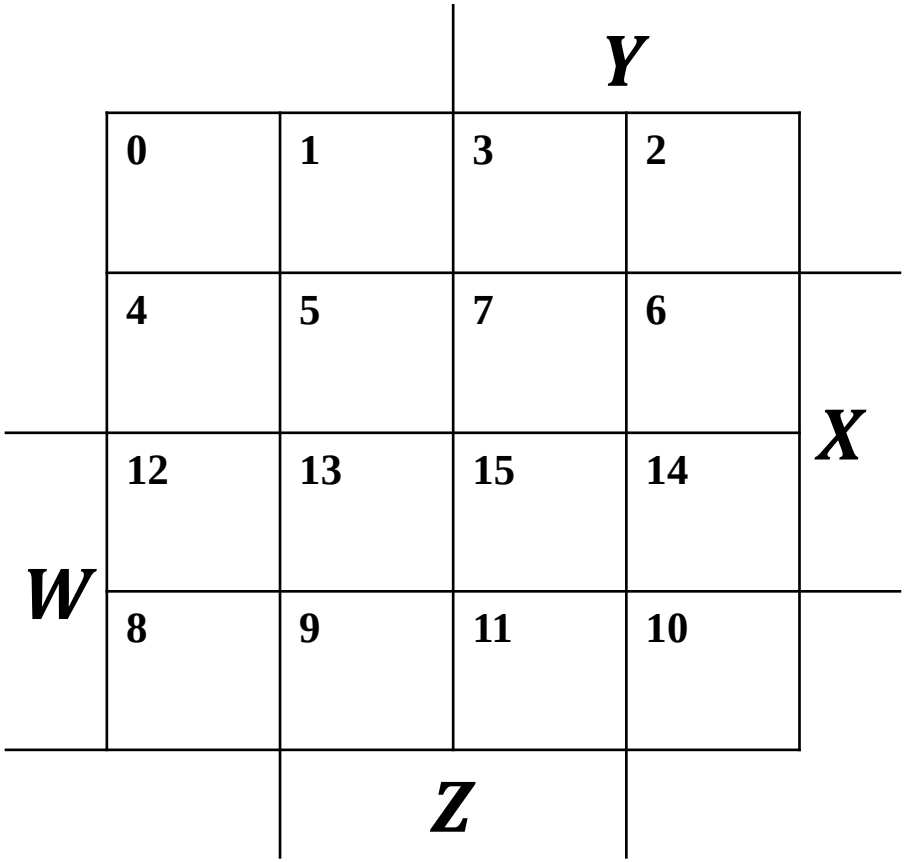
Four-Variable Maps

- Example shapes of 4-cell rectangles:



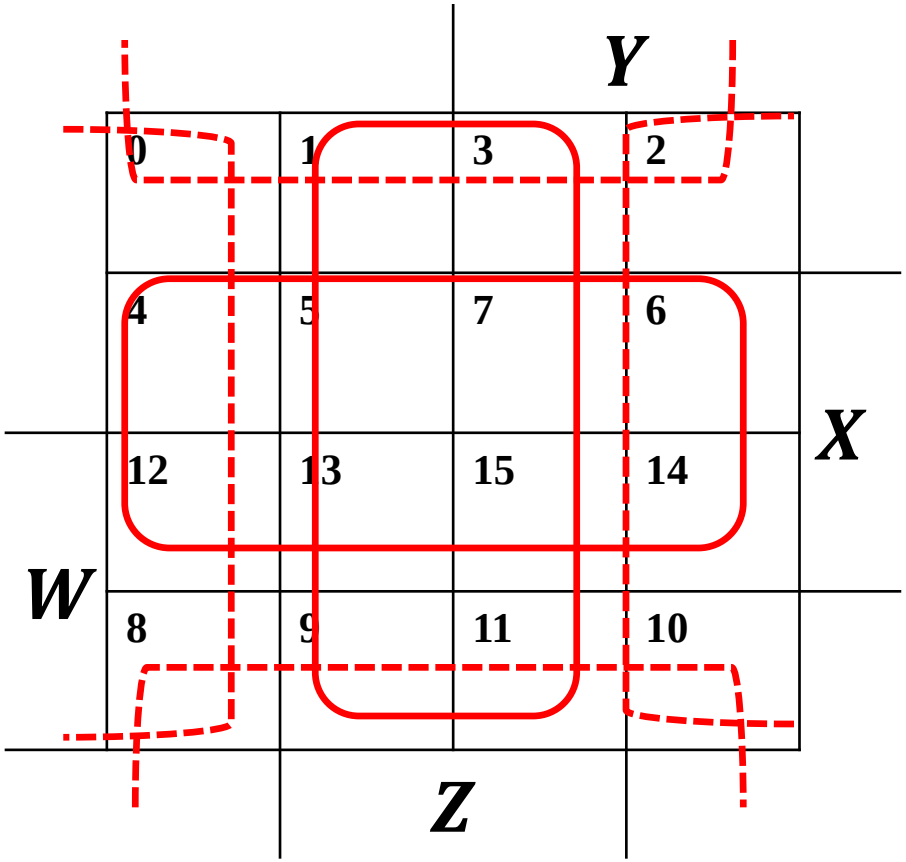
Four-Variable Maps

- Example shapes of 8-cell rectangles:



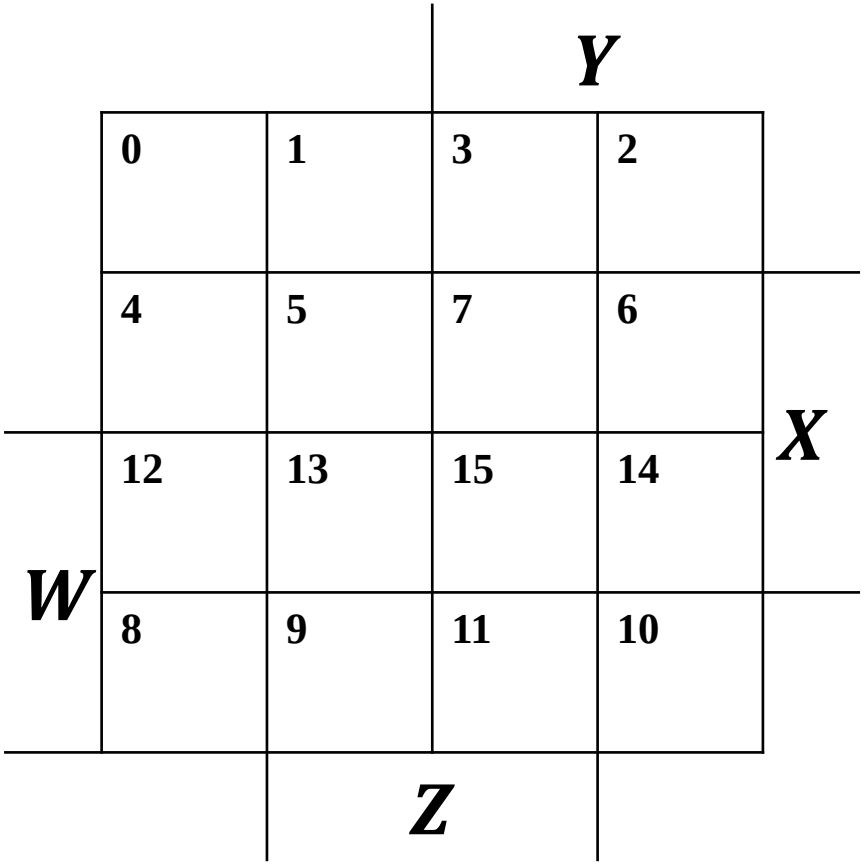
Four-Variable Maps

- Example shapes of 8-cell rectangles:



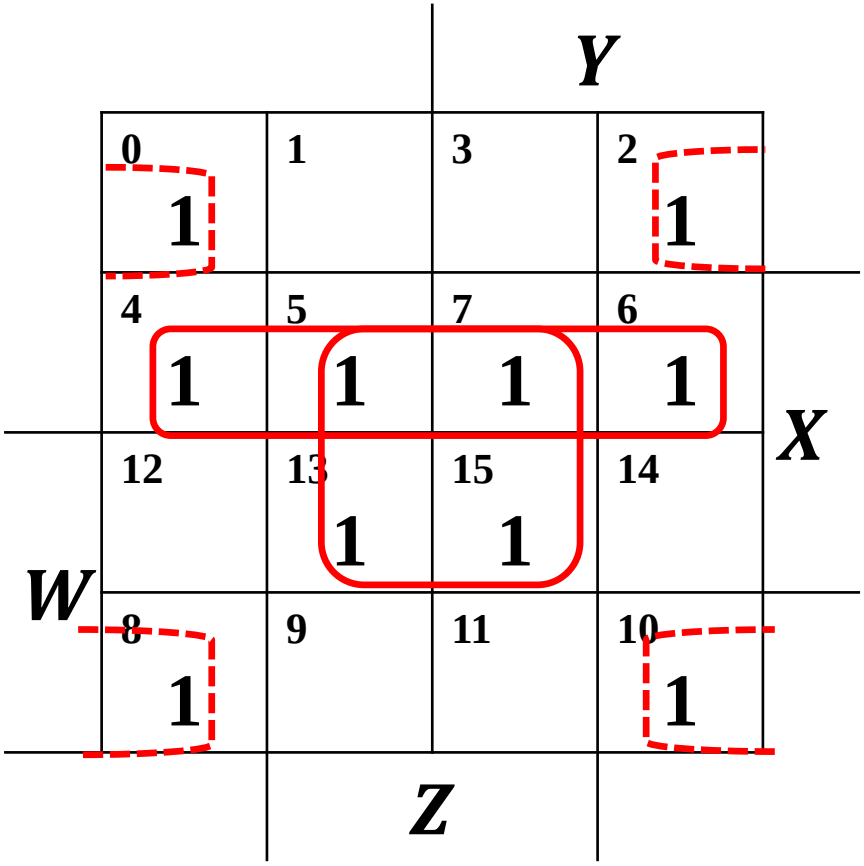
Four-Variable Map Simplification

- $F(W, X, Y, Z) = \sum_m(0,2,4,5,6,7,8,10,13,15)$



Four-Variable Map Simplification

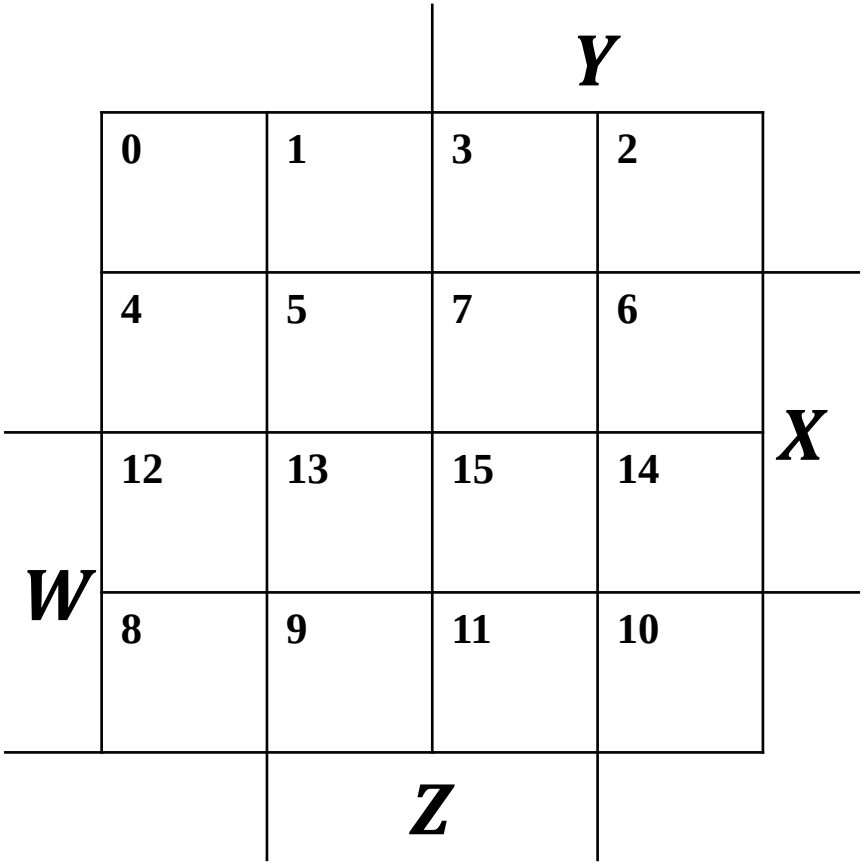
- $F(W,X,Y,Z) = \sum_m(0,2,4,5,6,7,8,10,13,15)$



$$F(W,X,Y,Z) = XZ + \bar{X}\bar{Z} + \bar{W}X$$

Four-Variable Map Simplification

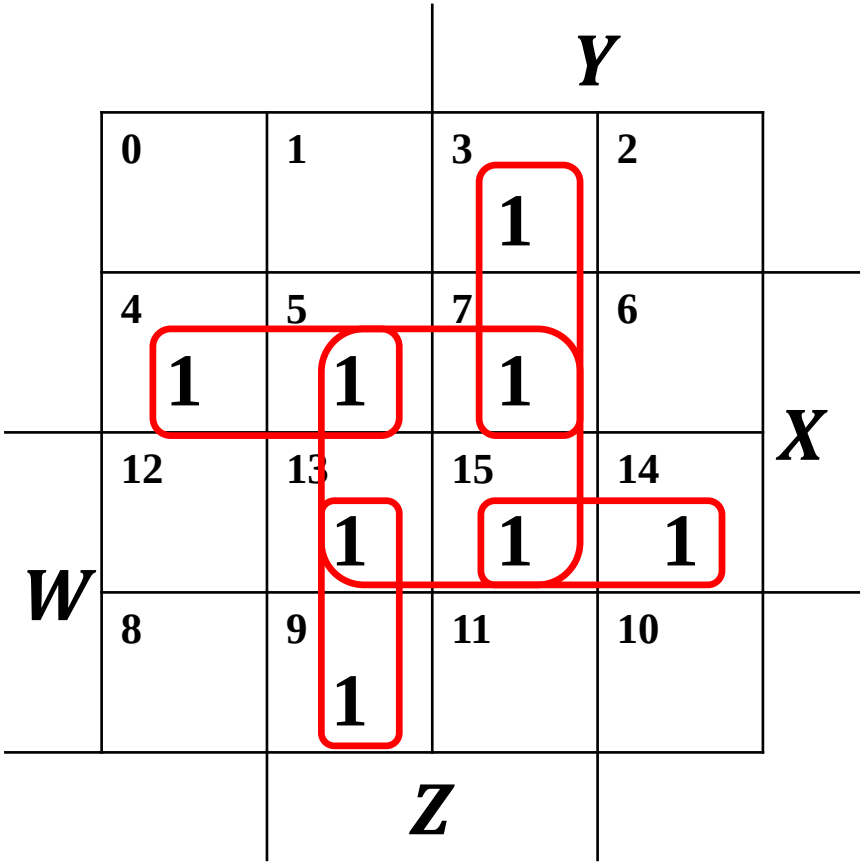
- $F(W,X,Y,Z) = \sum_m(3,4,5,7,9,13,14,15)$



$$F(W,X,Y,Z) = \bar{W}YZ + \bar{W}X\bar{Y} + WXY + W\bar{Y}Z$$

Four-Variable Map Simplification

- $F(W,X,Y,Z) = \sum_m(3,4,5,7,9,13,14,15)$



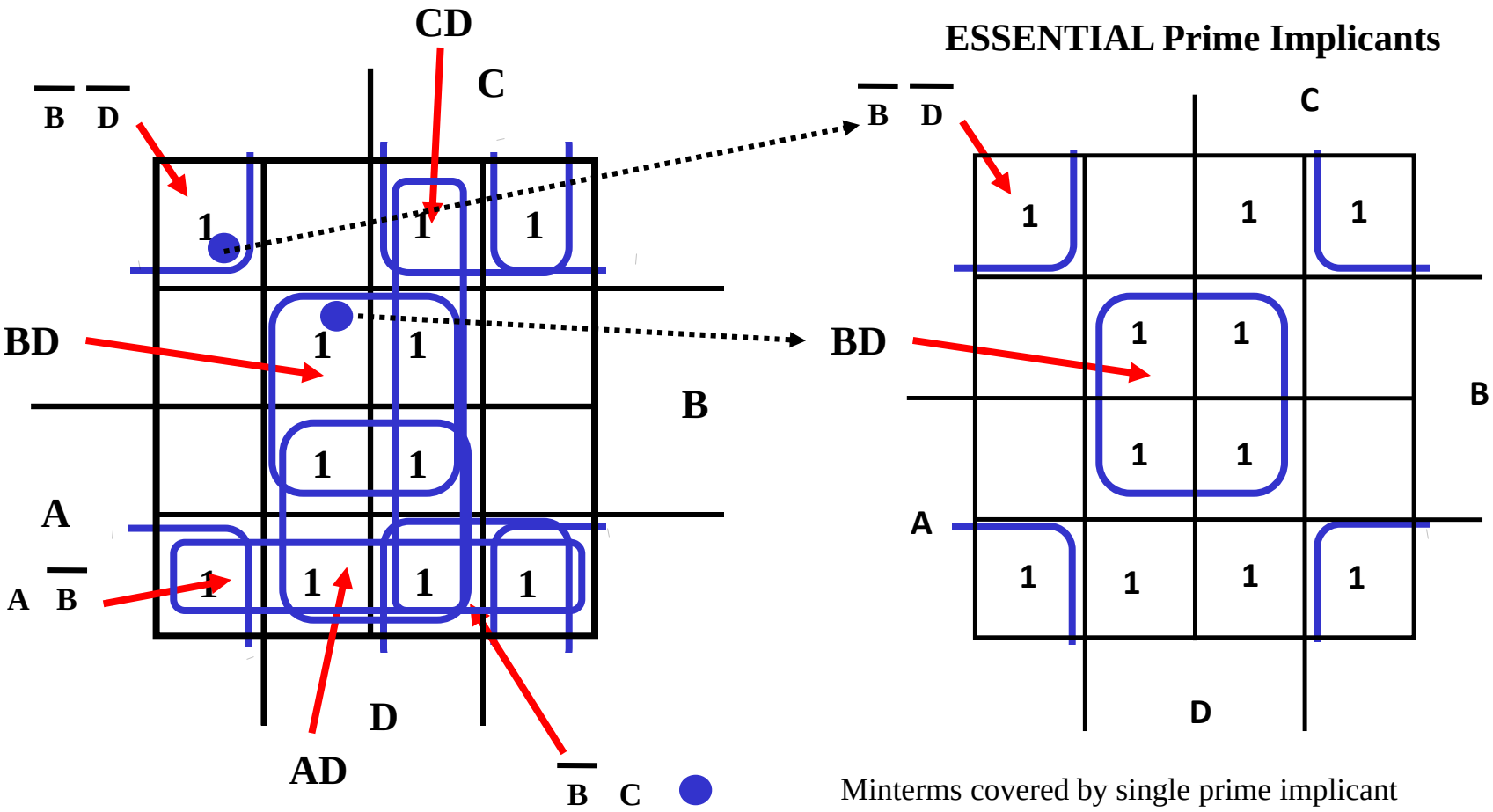
$$F(W,X,Y,Z) = \bar{W}YZ + \bar{W}X\bar{Y} + WXY + W\bar{Y}Z$$

Systematic Simplification

- ***Prime Implicant:*** is a product term obtained by combining the **maximum** possible number of adjacent squares in the map into a rectangle with the number of squares a power of 2
- A prime implicant is called an ***Essential Prime Implicant*** if it is the **only** prime implicant that covers (includes) one or more minterms
- Prime Implicants and Essential Prime Implicants can be determined by inspection of a K-Map
- A set of prime implicants *"covers all minterms"* if, for each minterm of the function, at least one prime implicant in the set of prime implicants includes the minterm

Example of Prime Implicants

- **Find ALL Prime Implicants**

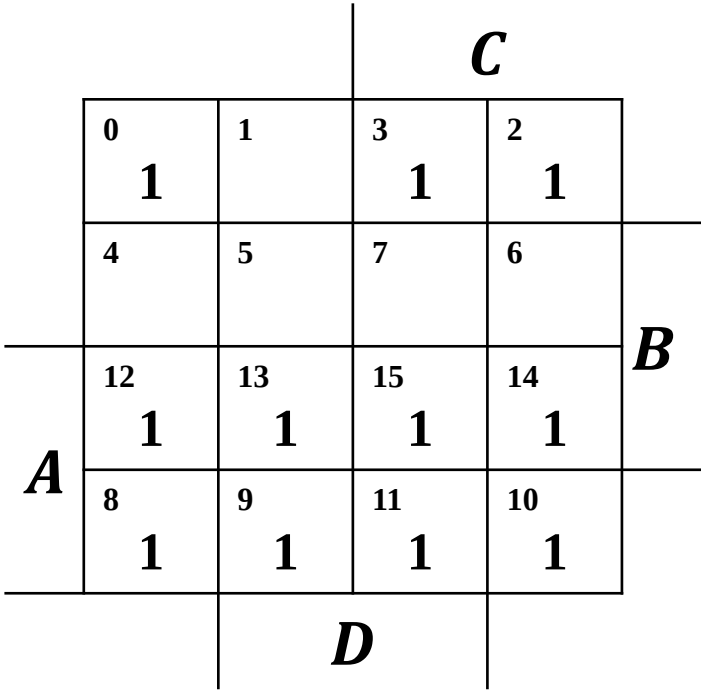


Prime Implicant Practice

- Find all prime implicants for:

$$F(A,B,C,D) = \sum_{m}(0,2,3,8,9,10,11,12,13,14,15)$$

- Prime Implicants:

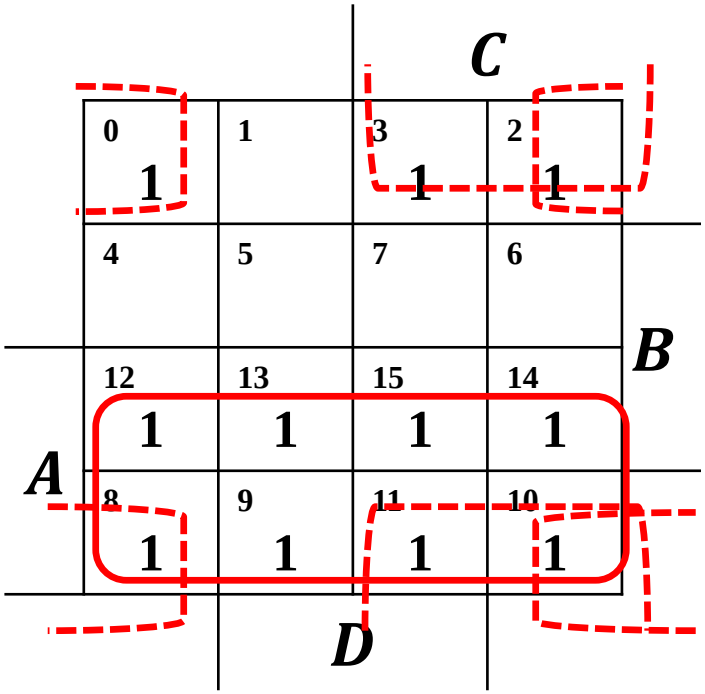


Prime Implicant Practice

- Find all prime implicants for:

$$F(A,B,C,D) = \sum_{m}(0,2,3,8,9,10,11,12,13,14,15)$$

- Prime Implicants:
  - A
  - $\bar{B}C$
  - $\bar{B}\bar{D}$

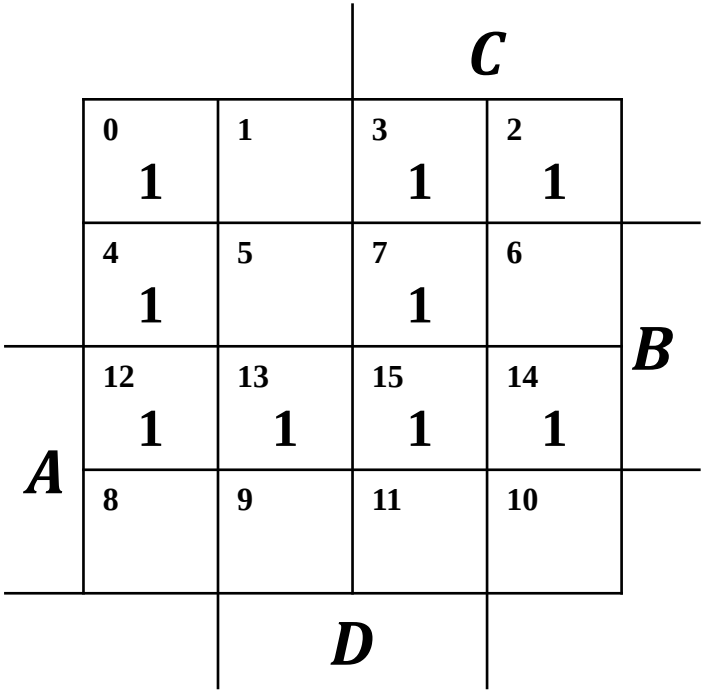


Another Example

- Find all prime implicants for:

$$G(A,B,C,D) = \sum_{m}(0,2,3,4,7,12,13,14,15)$$

- Hint: There are seven prime implicants!
- Prime Implicants:



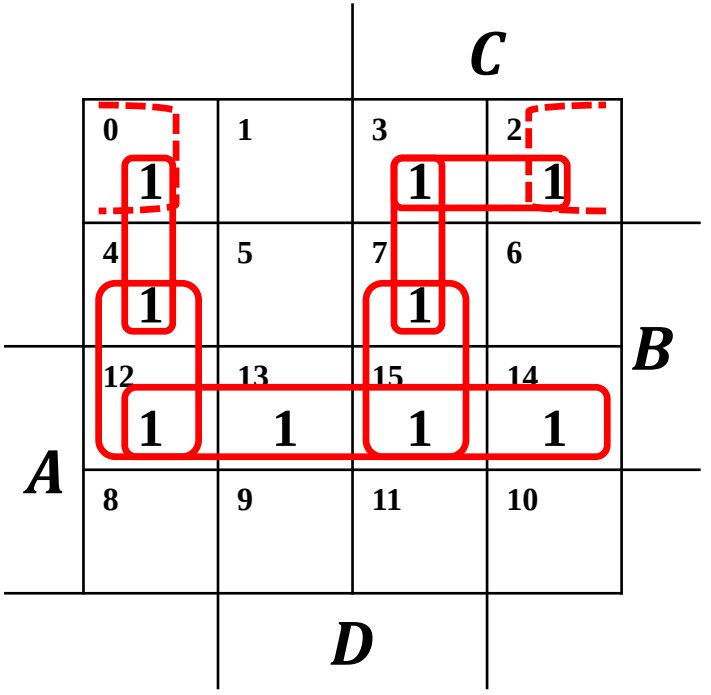
Another Example

- Find all prime implicants for:

$$G(A,B,C,D) = \sum_{m}(0,2,3,4,7,12,13,14,15)$$

- Hint: There are seven prime implicants!

- Prime Implicants:
  - AB
  - BCD
  - $B\bar{C}\bar{D}$
  - $\bar{A}CD$
  - $\bar{A}\bar{C}\bar{D}$
  - $\bar{A}\bar{B}C$
  - $\bar{A}\bar{B}\bar{D}$

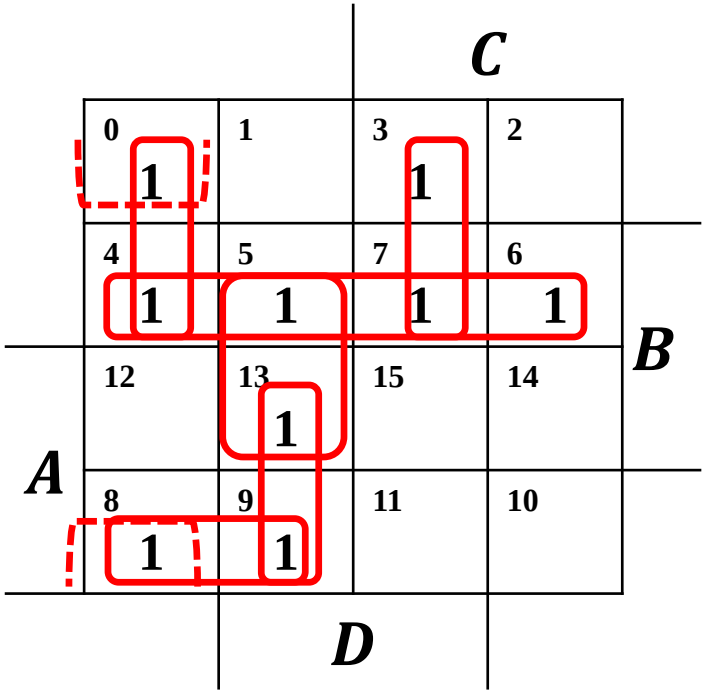


Optimization Algorithm

1. Find all prime implicants
2. Include all essential prime implicants in the solution
3. Select a ***minimum cost*** set of non-essential prime implicants to cover all minterms not yet covered
   - Selection Rule: Minimize the overlap among prime implicants as much as possible. In particular, in the final solution, make sure that each prime implicant selected includes at least one minterm not included in any other prime implicant selected

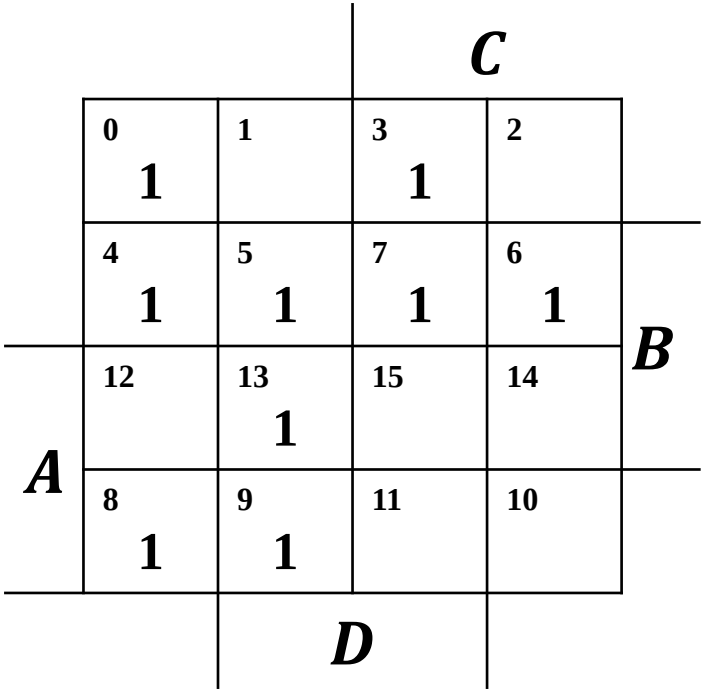
Selection Rule Example

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



Prime Implicants

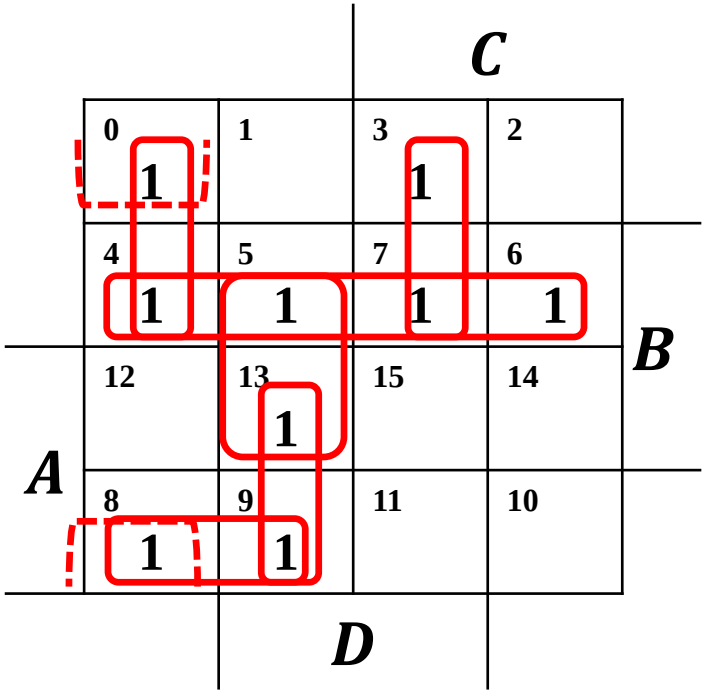
Selected Non-essential Prime Implicants



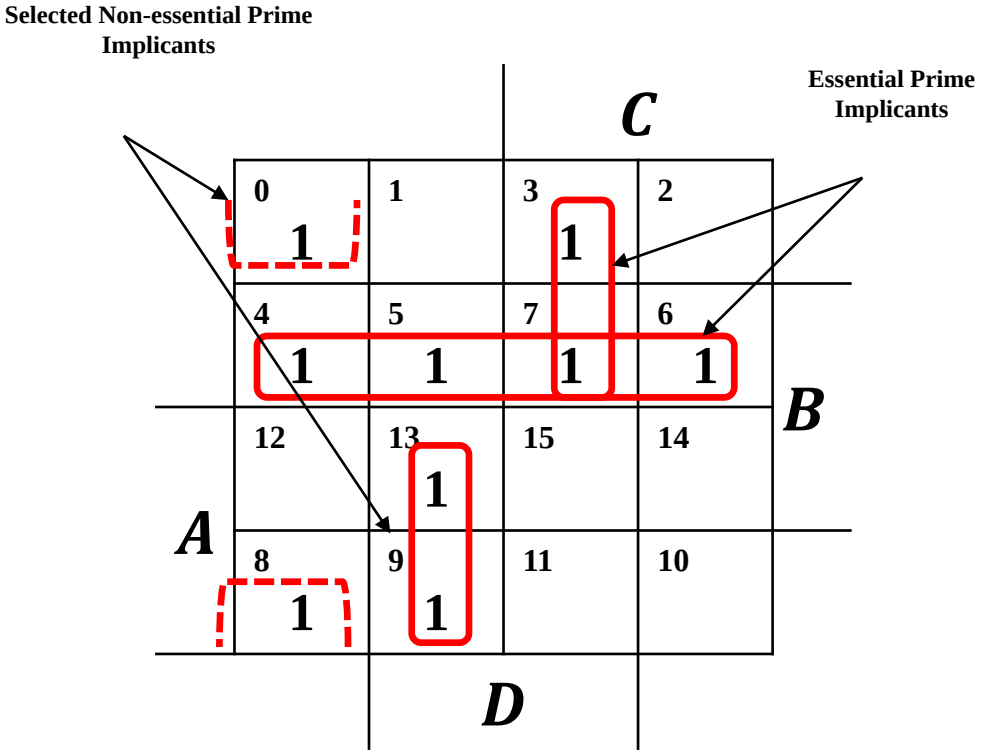
Essential and Selected Non-essential Prime Implicants

Selection Rule Example

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



Prime Implicants



Essential and Selected Non-essential Prime Implicants

Product of Sums Example

- Find the optimum POS solution for:

$$F(A,B,C,D) = \sum_m (1,3,9,11,12,13,14,15)$$

- Solution:
  - Find optimized SOP for $\bar{F}$ by combining 0's in K-Map of F
  - Complement $\bar{F}$ to obtain optimized POS for F

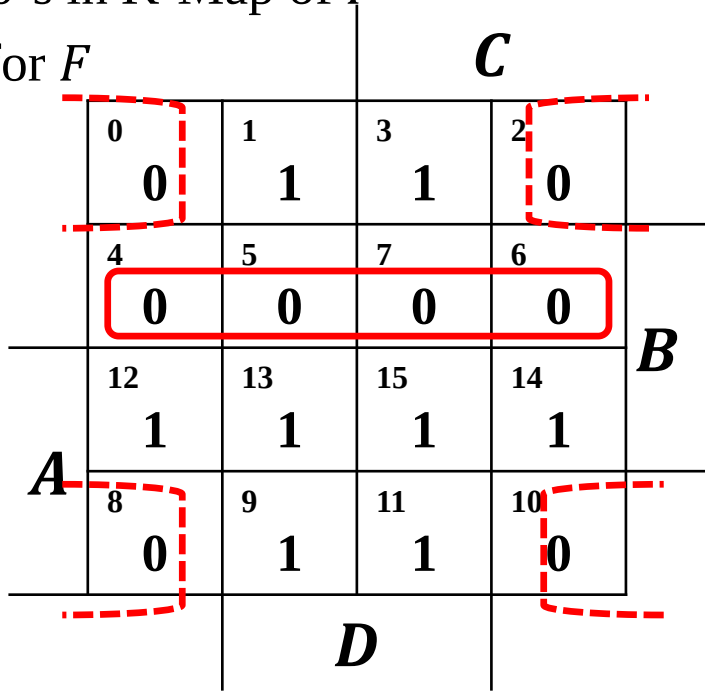
			C	C	
		0: **0**	1: **1**	3: **1**	2: **0**
		4: **0**	5: **0**	7: **0**	6: **0**	B
A		12: **1**	13: **1**	15: **1**	14: **1**	B
A		8: **0**	9: **1**	11: **1**	10: **0**
			D	D	

Product of Sums Example

- Find the optimum POS solution for:

$$F(A,B,C,D) = \sum_m (1,3,9,11,12,13,14,15)$$

- Solution:
  - Find optimized SOP for $\bar{F}$ by combining 0's in K-Map of F
  - Complement $\bar{F}$ to obtain optimized POS for F



- $\bar{F}(A,B,C,D) = \bar{A}B + \bar{B}\bar{D}$

- Using Demorgan's Law:

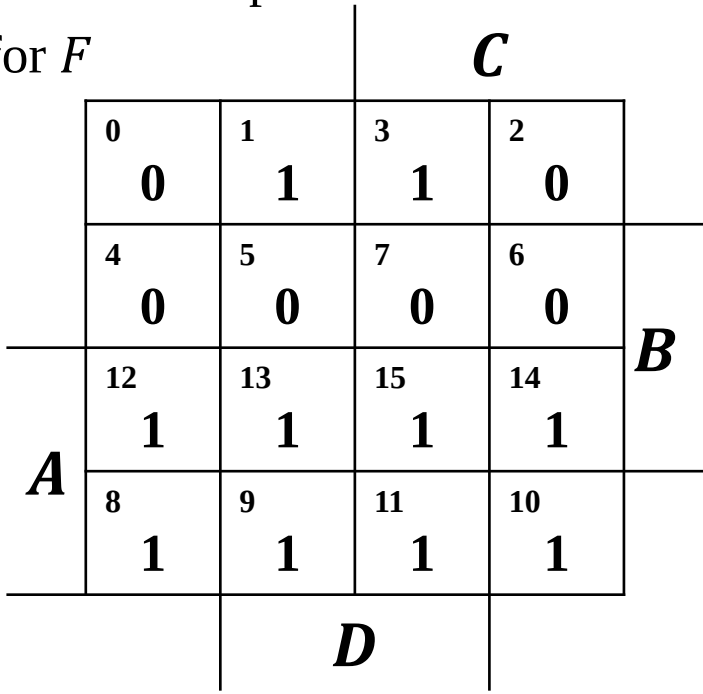
$$F(A,B,C,D) = (A + \bar{B})(B + D)$$

Example

- Find the optimum POS and SOP solution for:

$$F(A,B,C,D) = \prod_{M}(0,2,4,5,6,7)$$

- POS solution (Red):
  - Find optimized SOP for $\bar{F}$ by combining 0's in K-Map of F
  - Complement $\bar{F}$ to obtain optimized POS for F
- SOP solution (Blue):



Example

- Find the optimum POS and SOP solution for:

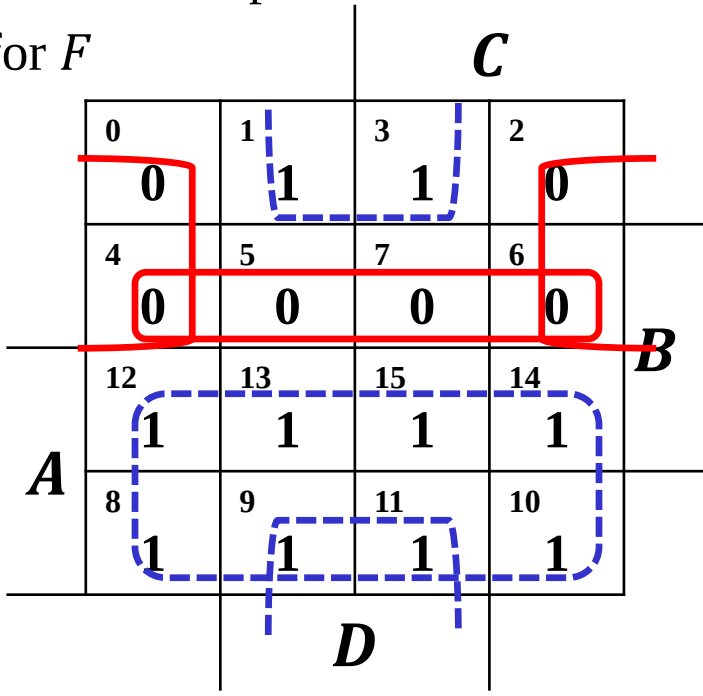
$$F(A,B,C,D) = \prod_{M}(0,2,4,5,6,7)$$

- POS solution (Red):
  - Find optimized SOP for $\bar{F}$ by combining 0's in K-Map of F
  - Complement $\bar{F}$ to obtain optimized POS for F

$$\bar{F}(A,B,C,D) = \bar{A}B + \bar{A}\bar{D}$$
$$F(A,B,C,D) = (A + \bar{B})(A + D)$$

- SOP solution (Blue):
  - Combining 1's in K-Map of F

$$F(A,B,C,D) = A + \bar{B}D$$

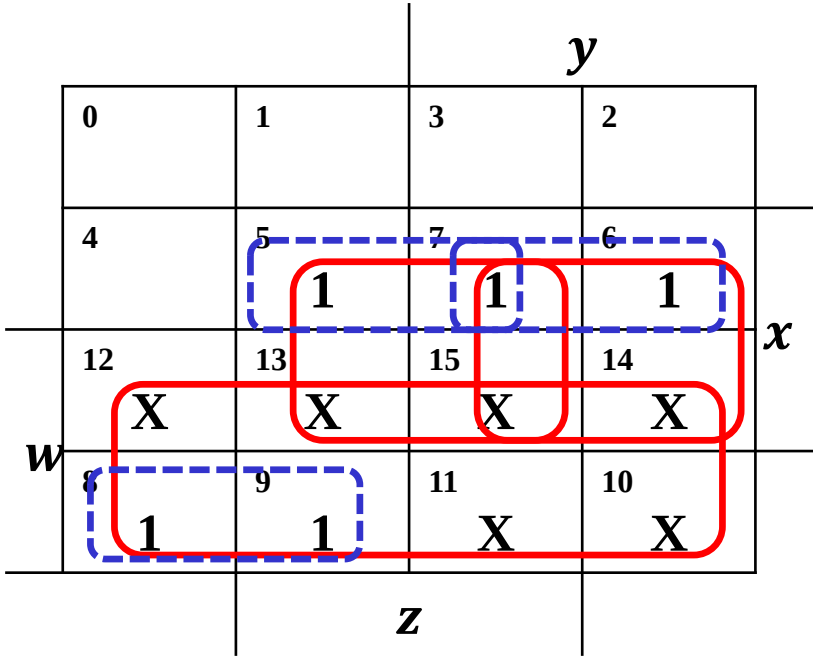


Don't Cares in K-Maps

- Incompletely specified functions: Sometimes a function table or map contains entries for which it is known:
  - the input values for the minterm will never occur, or
  - The output value for the minterm is not used
- In these cases, the output value is defined as a "don't care"
- By placing "don't cares" (an "x" entry) in the function table or map, the cost of the logic circuit may be lowered
- **Example:** 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"
- ***"Don't care" minterms cannot be replaced with 1's or 0's because that would require the function to be always 1 or 0 for the associated input combination***

Example: BCD “5 or More”

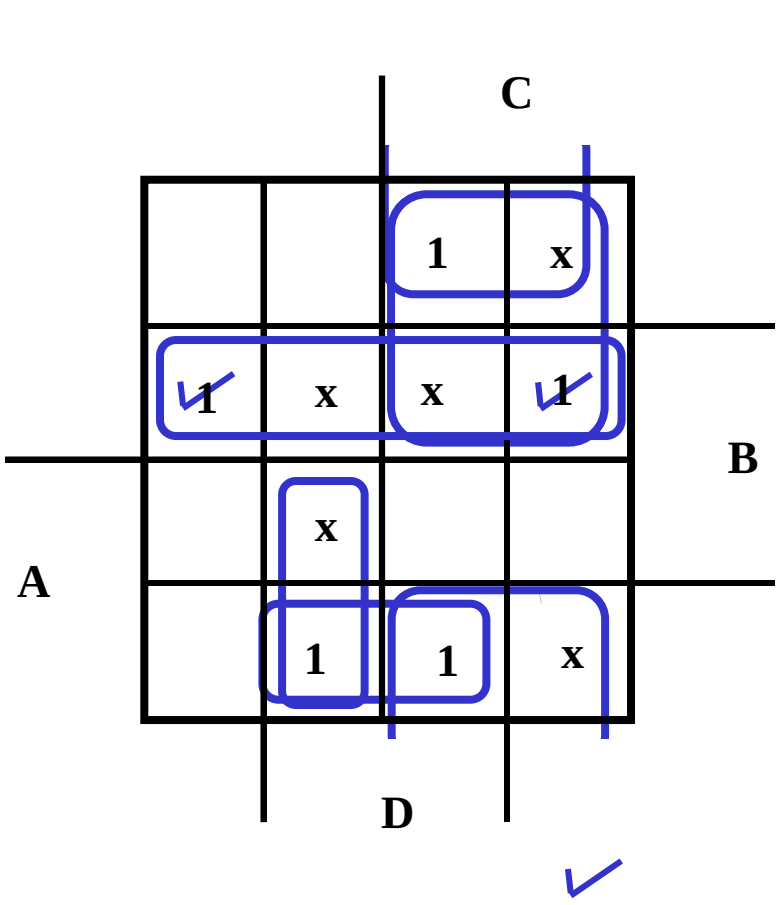
- The map below gives a function $F(w, x, y, z)$ which is defined as "5 or more" over BCD inputs. With the don't cares used for the 6 non-BCD combinations:
- If don’t cares are treated as 1’s (Red):
- $F_1(w, x, y, z) = w + xy + xz$
  - $G = 7$
- If don’t cares are treated as 0’s (Blue):
- $F_2(w, x, y, z) = \bar{w}xz + \bar{w}xy + w\bar{x}\bar{y}$
  - $G = 12$

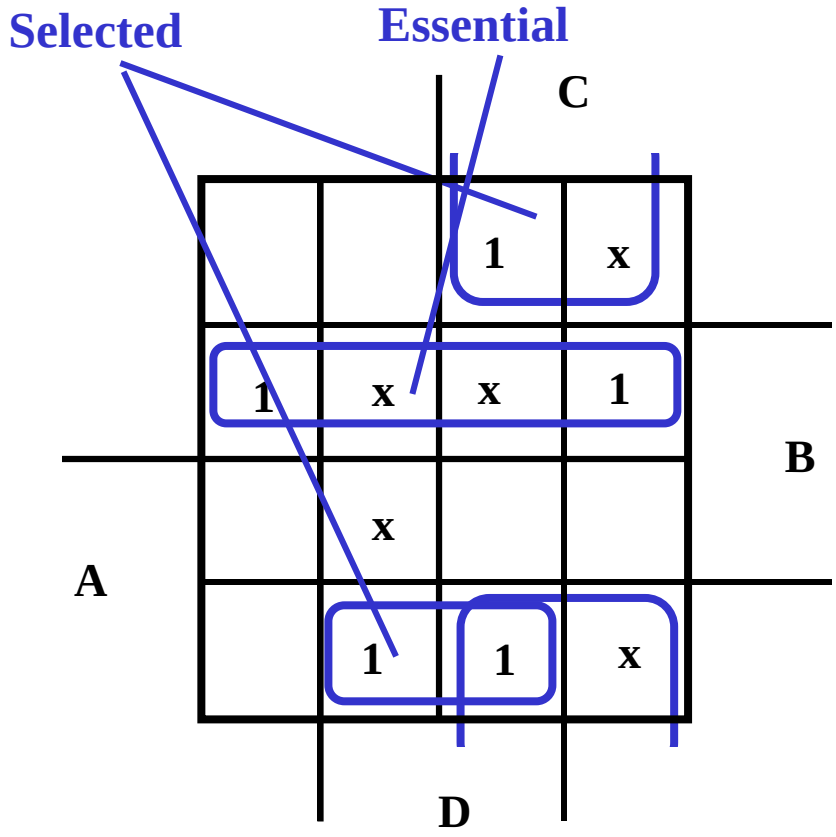


- ***For this particular function, cost G for the POS solution for $F(w, x, y, z)$ is not changed by using the don't cares***
  - ***Choose the one less inverters (i.e. less GN)***

Selection Rule Example with Don't Cares

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



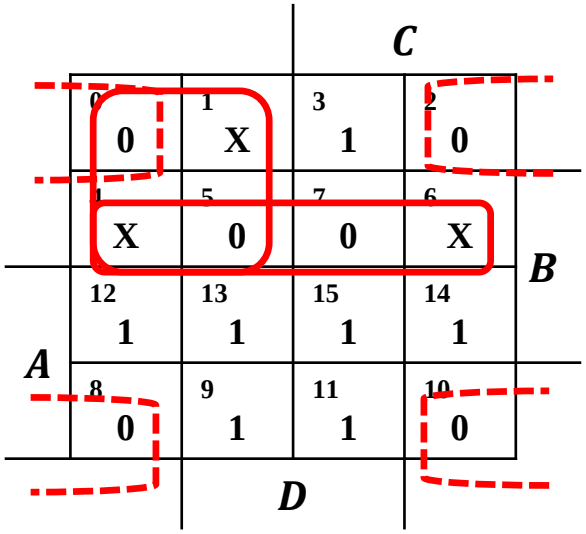


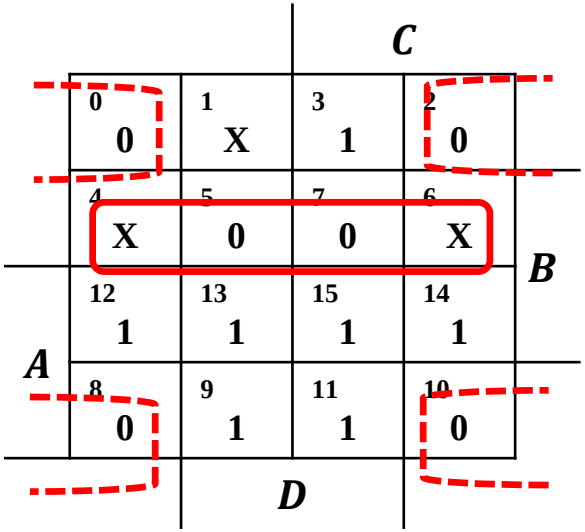
Minterms covered by essential prime implicants

Product of Sums with Don't Care Example

- Find the optimum **POS** solution for:

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



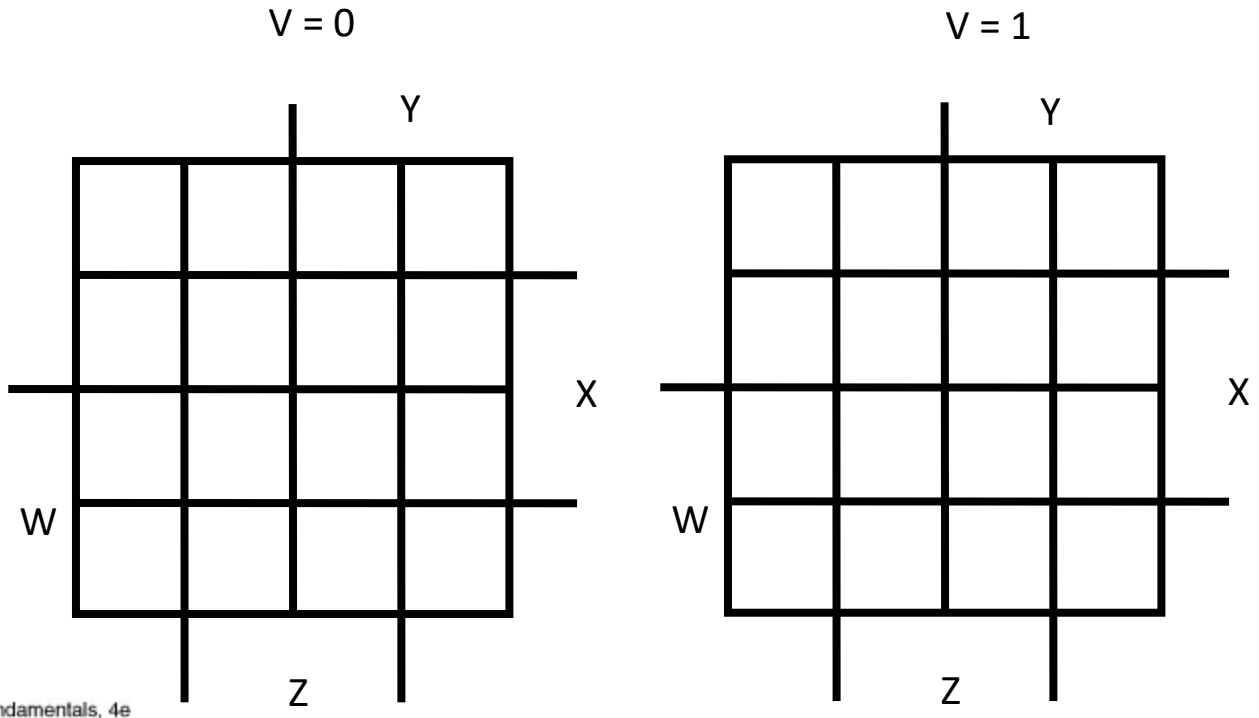


$$\bar{F}(A,B,C,D) = \bar{A}B + \bar{B}\bar{D}$$

$$F(A,B,C,D) = (A + \bar{B})(B + D)$$

Five Variable or More K-Maps

- For five variable problems, we use ***two adjacent*** *K-maps*. It becomes harder to visualize adjacent minterms for selecting PIs. You can extend the problem to six variables by using four K-Maps.



Terms of Use

- **All (or portions) of this material © 2008 by Pearson Education, Inc.**
- **Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.**
- **These materials or adaptations thereof are not to be sold or otherwise offered for consideration.**
- **This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.**