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 it complement
- ✓ Literal cost – the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram
- ✓ Example, which solution is best?
  - $F = BD + A\underline{\overline{B}}C + A\underline{\overline{C}}\,\underline{\overline{D}}$ $L = 8$
  - $F = BD + ABC + ABD + \underline{A}B\underline{\overline{C}}$ $L = 11$
  - $F = (A + B)(A + D)(B + C + D)(\overline{B} + \overline{C} + D)$ $L = 10$

Gate Input Cost

- ✓ Gate input costs - 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, (G) and
  - optionally, the number of distinct complemented single literals (GN).
- ✓ Example, which solution is best?
  - $F = BD + A\bar{B}C + A\bar{C}\bar{D}$ G =11, GN = 14
  - $F = BD + A\bar{B}C + A\bar{B}\bar{D} + AB\bar{C}$ G =15, GN = 18
  - $F = (A + \bar{B})(A + D)(B + C + \bar{D})(\bar{B} + \bar{C} + D)$ G =14, GN = 17

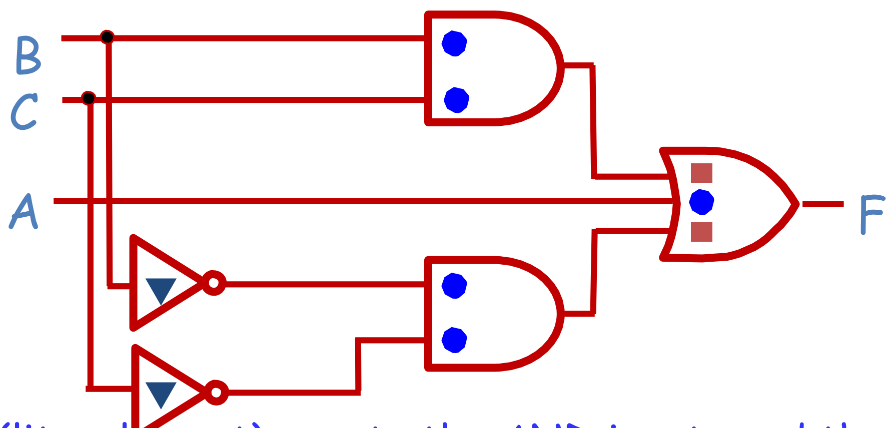
Cost Criteria (continued)

- ✓ Example 1:
- ✓ $F = A + B\,C + \bar{B}\,\bar{C}$

$GN = G + 2 = 9$

$L = 5$

$G = L + 2 = 7$



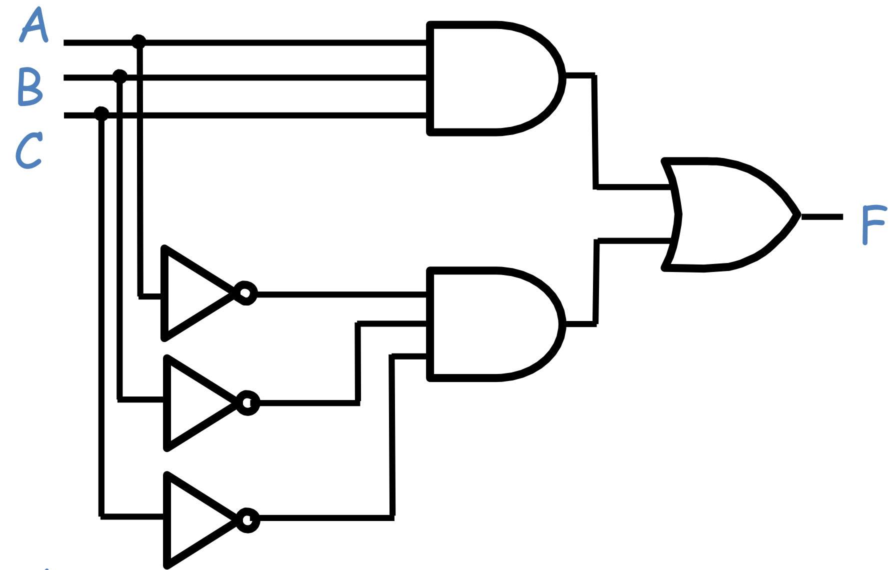
L (literal count) counts the AND inputs and the single literal OR input.

G (gate input count) adds the remaining OR gate inputs

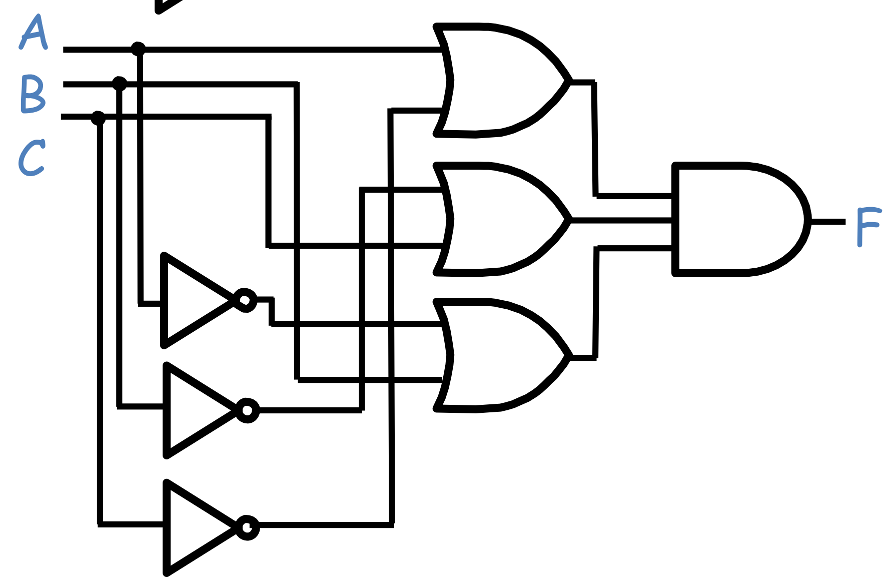
GN(gate input count with NOTs) adds the inverter inputs

Cost Criteria (continued)

- $F = A\,B\,C + \overline{A}\,\overline{B}\,\overline{C}$
- $L = 6 \quad G = 8 \quad GN = 11$



- $F = (A + \overline{C})(\overline{B} + C)(\overline{A} + B)$
- $L = 6 \quad G = 9 \quad GN = 12$



✓ <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

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 first.
- ✓ Introduce a graphical technique using Karnaugh maps (K-maps, for short)

Karnaugh Maps (K-map)

- A K-map is a collection of squares
  - Each square represents a minterm
  - 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 (corresponding to cubes)
- The K-map can be viewed as

A reorganized version of the truth table or a particular cube representation

Some Uses of K-Maps

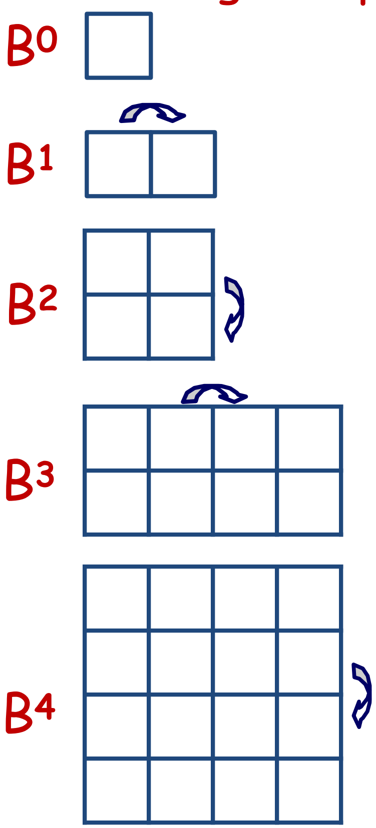
- Provide a means for:
  - Finding optimum
    - SOP and POS standard forms, and
    - two-level AND/OR and OR/AND circuit implementations

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

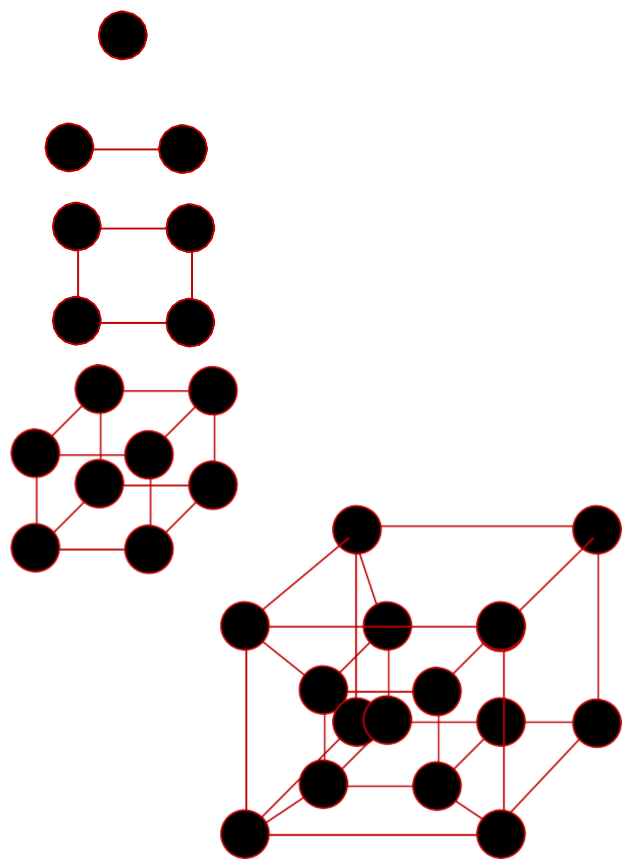
The Boolean Space B^n

- ✓ $B = \{0,1\}$
- ✓ $B^2 = \{0,1\} \times \{0,1\} = \{00, 01, 10, 11\}$

Karnaugh Maps:



Boolean Cubes:



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

x \ y	y = 0	y = 1
x = 0	m_0 $\overline{x\,y}$	m_1 $\bar{x}\,y$
x = 1	m_2 $x\,\bar{y}$	m_3 $x\,y$

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).

Function Table

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

K-Map

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

K-Map Function Representation

- ✓ Example: $F(x,y) = x$

$F = x$	$y = 0$	$y = 1$
$x = 0$	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:

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

K-Map Function Representation

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

G=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:

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

Duplicate xy

Three Variable Maps

- A three-variable K-map:

yz / x	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

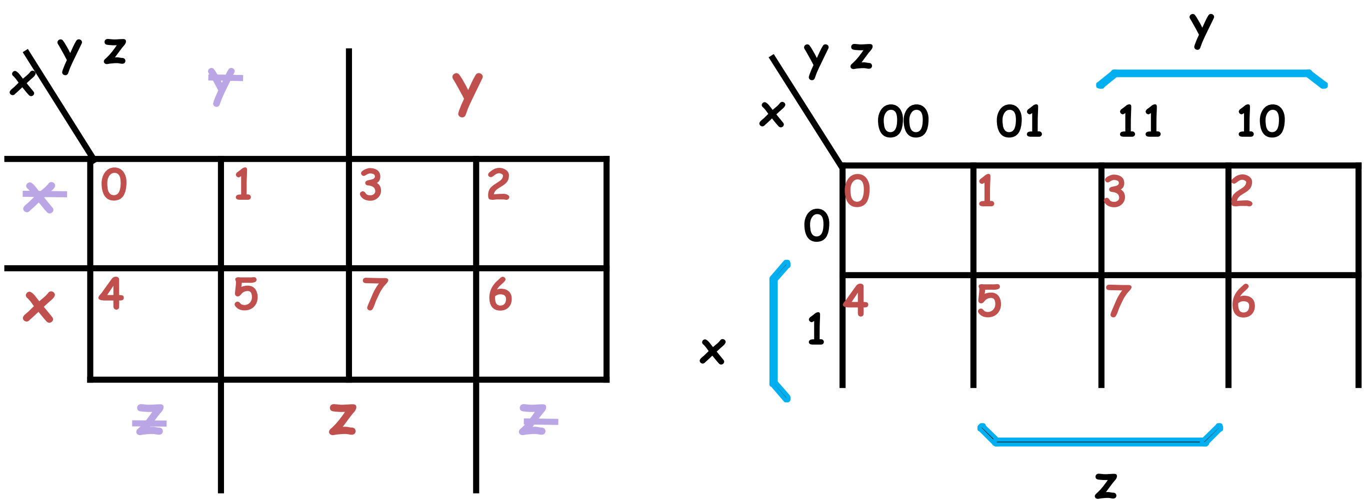
- Where each minterm corresponds to the product terms:

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

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

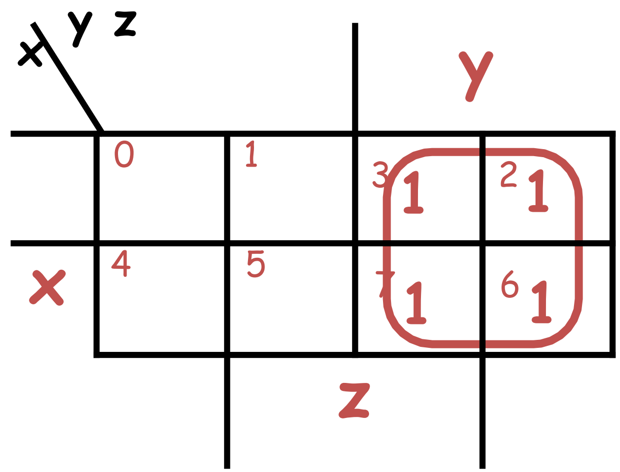
Alternative Map Labeling

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



Example: Combining Squares

- ✓ Example: Let
- ✓ $F(x, y, z) = \sum_m (2, 3, 6, 7)$



- ✓ Applying the Minimization Theorem three times:

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

- ✓ Thus the four terms that form a 2 × 2 square correspond to the term "y".

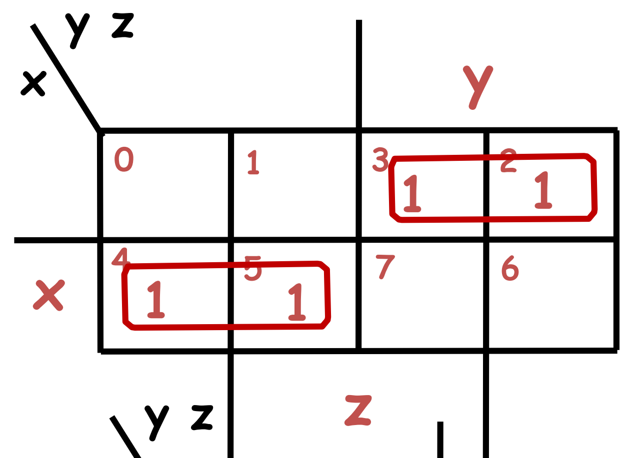
Combining Squares

- 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 3-variable K-Map:
  - One square represents a minterm with three variables
  - Two adjacent squares represent a cube that is product term with two variables
  - Four "adjacent" terms represent a cube that is product term with one variable
  - Eight "adjacent" terms is the function of all ones (no variables) is a tautology $f^1=1$.

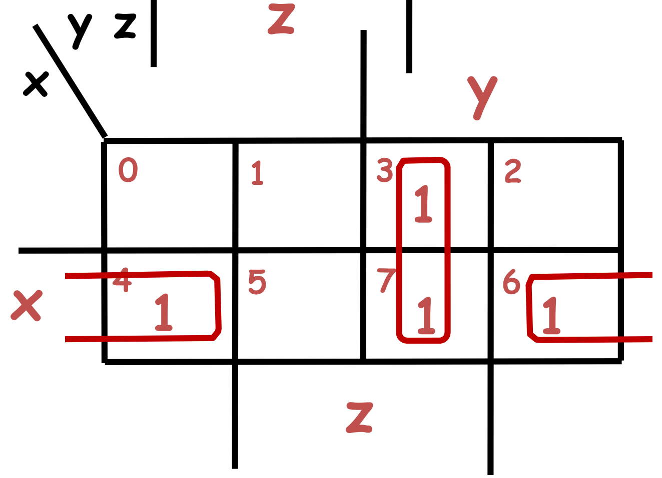
Example Functions

- By convention, we represent the minterms of F by a "1" in the map and leave the minterms of $\overline{F}$ blank
- Example:

$$F(x, y, z) = \sum_m (2, 3, 4, 5)$$
$$F(x, y, z) = \bar{x}\, y + x\, \bar{y}$$



- Example:
- $F(x, y, z) = \sum_m (3, 4, 6, 7)$
- $F(x, y, z) = y\, z + x\, \bar{z}$
- Learn the locations of the 8 indices based on the variable order shown (x, most significant and z, least significant) on the map boundaries

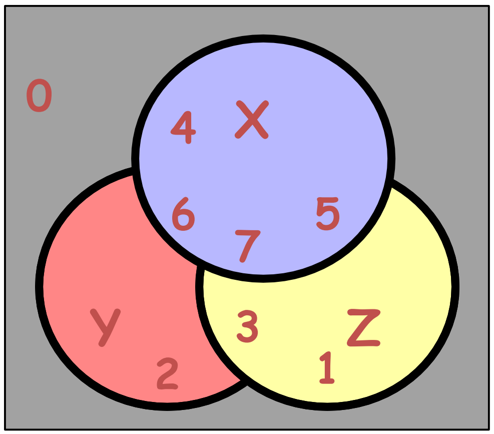


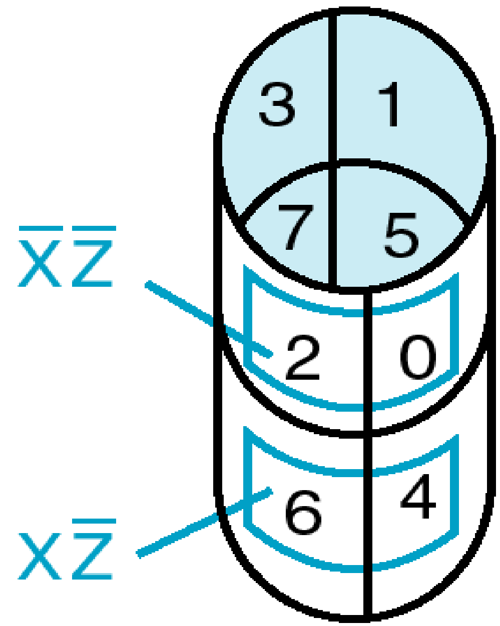
Three-Variable Maps

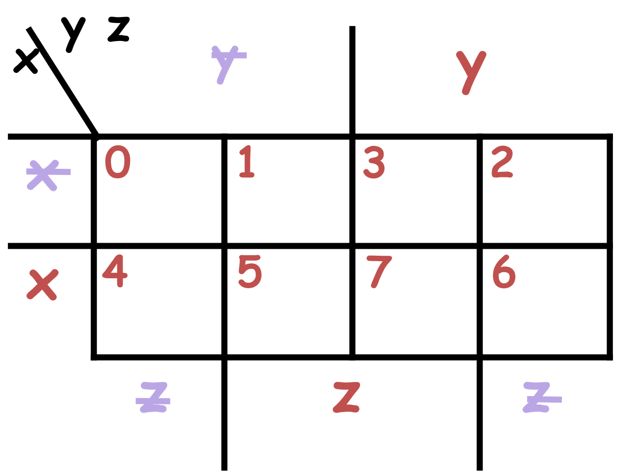
- ✓ Reduced literal product terms for SOP standard forms correspond to cubes i.e. to rectangles on the K-maps containing cell counts that are powers of 2.
- ✓ Rectangles of 2 cells represent 2 adjacent minterms; of 4 cells represent 4 minterms that form a “pairwise adjacent” ring.

Three-Variable Maps

- Topological warps of 3-variable K-maps that show *all* adjacencies:
  - Venn Diagram
  - Cylinder

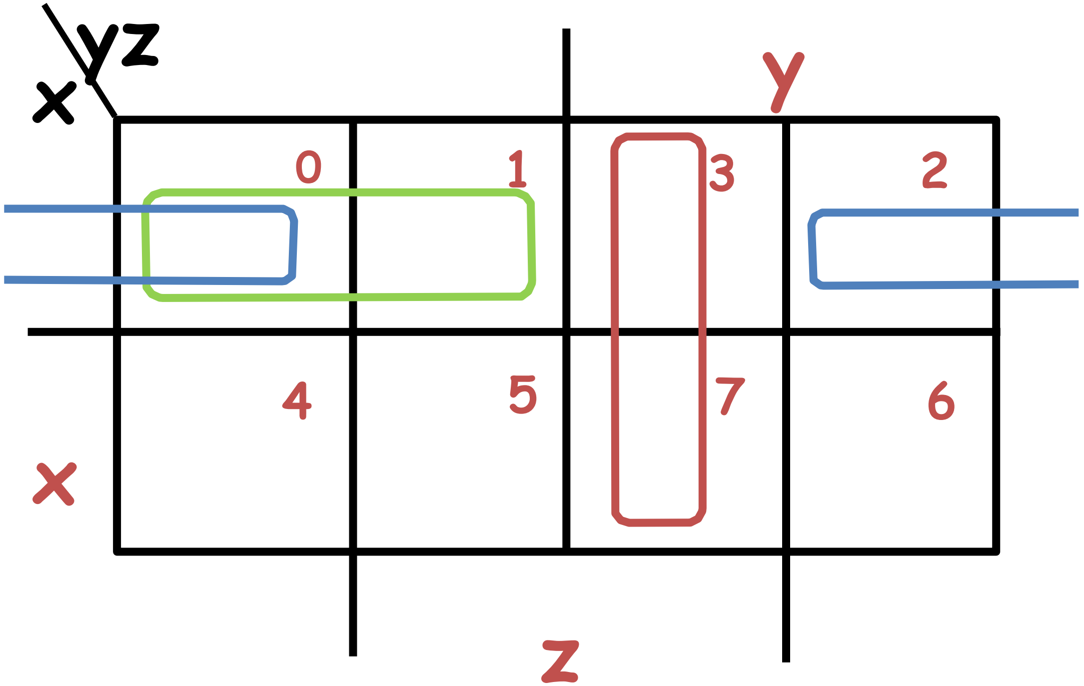






Three-Variable Maps

- ✓ Example Shapes of 2-cell Rectangles:

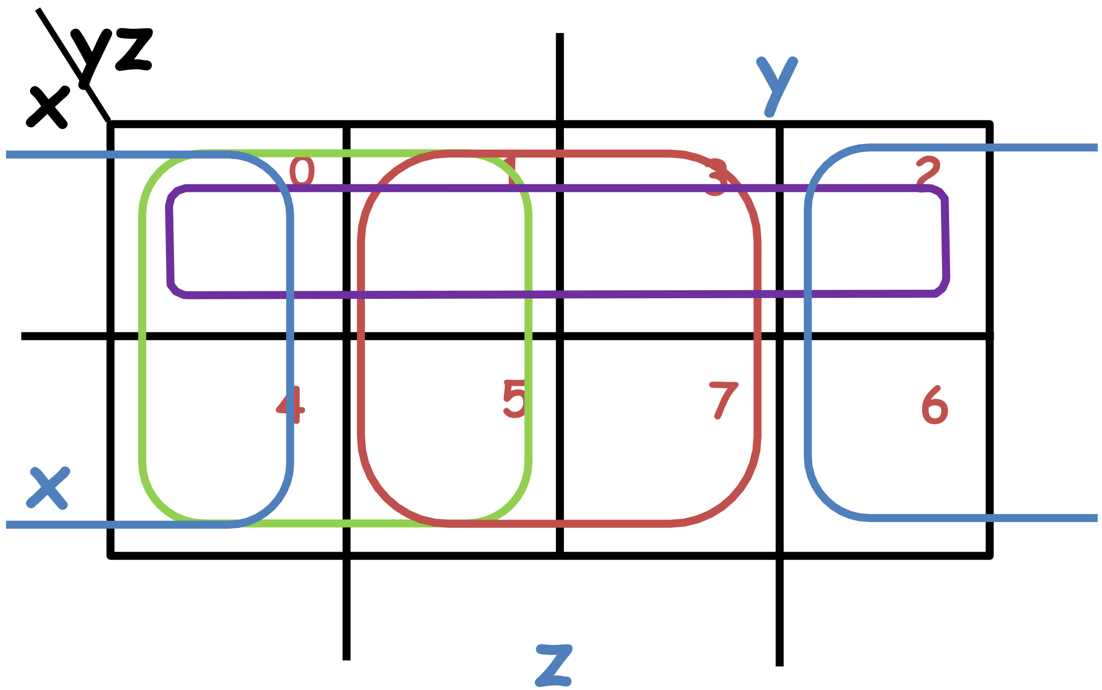


- ✓ Read off the product terms for the rectangles shown

$x'z'$ $x'y'$ yz

Three-Variable Maps

- ✓ Example Shapes of 4-cell Rectangles:

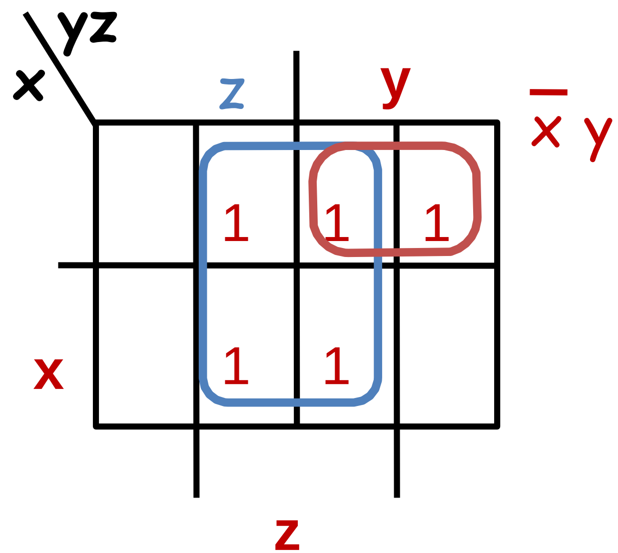


- ✓ Read off the product terms for the rectangles shown

z' z x' y'

Three Variable Maps

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

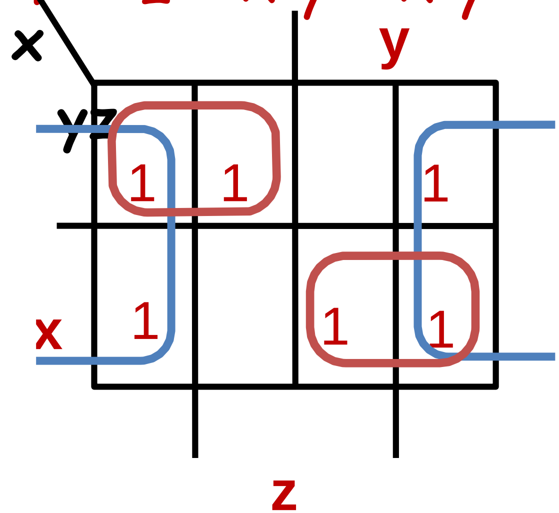


$F(x, y, z) = z + \bar{x}y$

Three-Variable Map Simplification

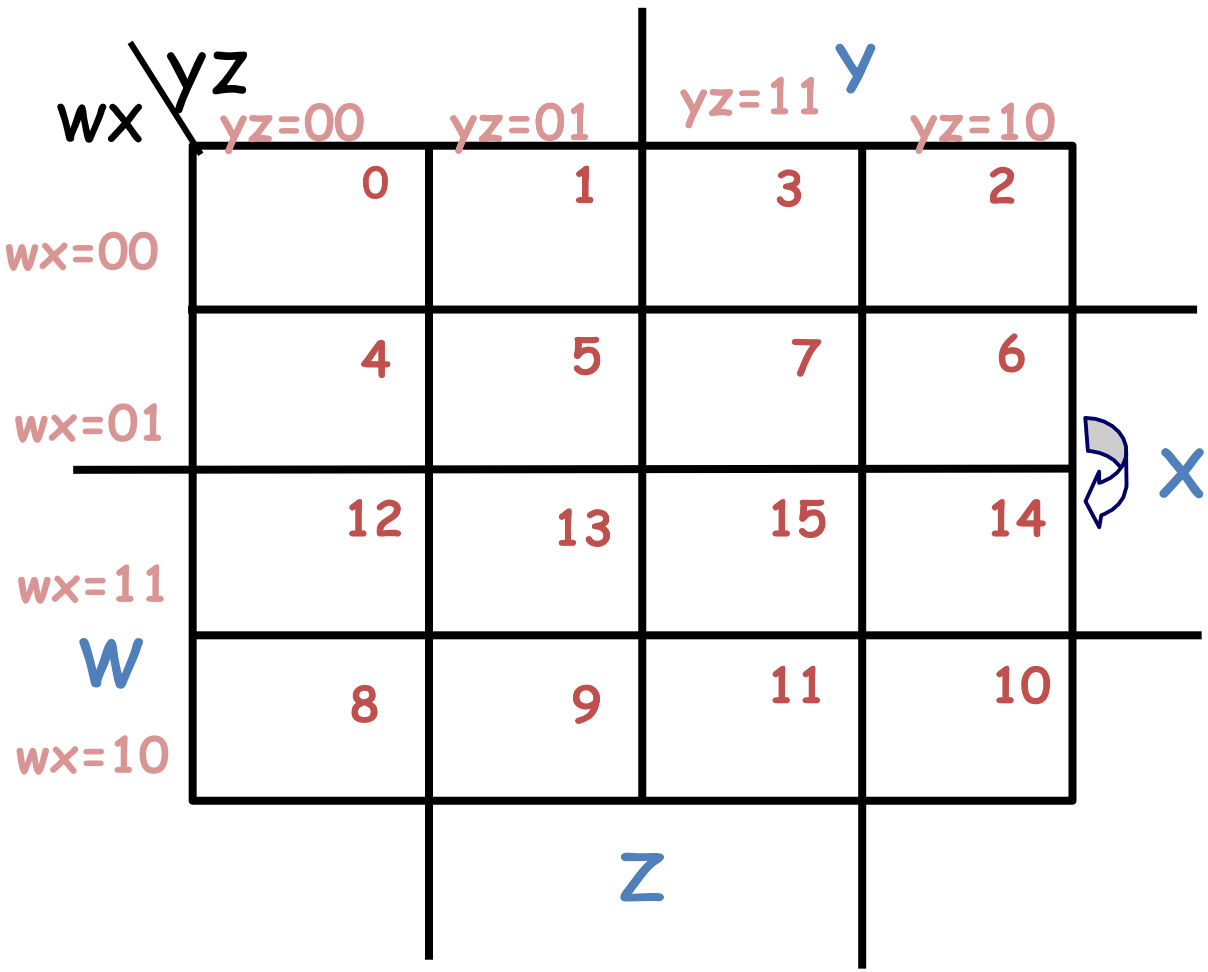
✓ Use a K-map to find an optimum SOP equation for $F(x, y, z) = \sum_m (0, 1, 2, 4, 6, 7)$

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



Four Variable Maps

✓ Map and location of minterms:



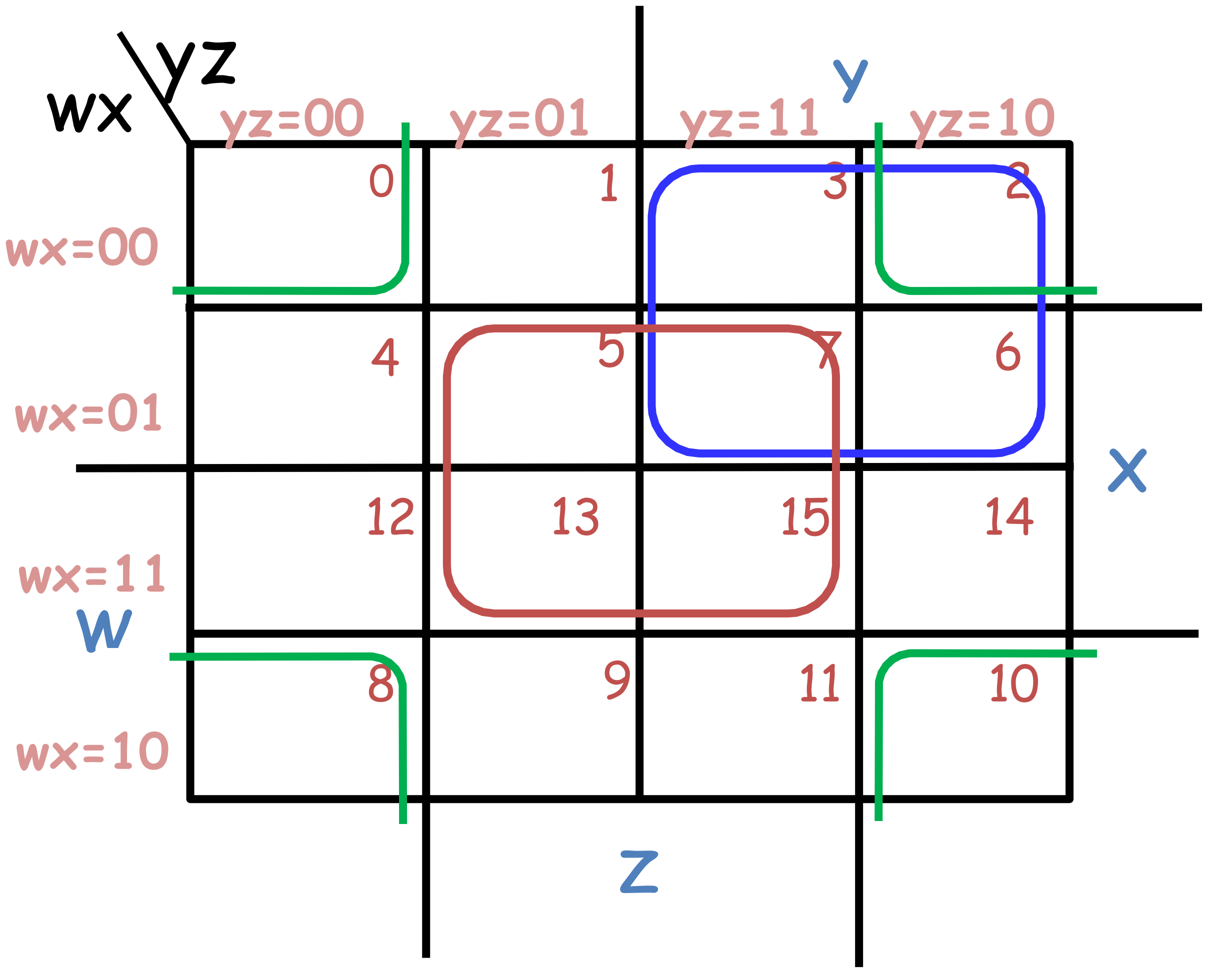
Four Variable Terms

✓ Four variable maps can have rectangles corresponding to:

- A single 1 = 4 variables, (i.e. Minterm)
- Two 1s = 3 variables,
- Four 1s = 2 variables
- Eight 1s = 1 variable,
- Sixteen 1s = zero variables (i.e. Constant "1")

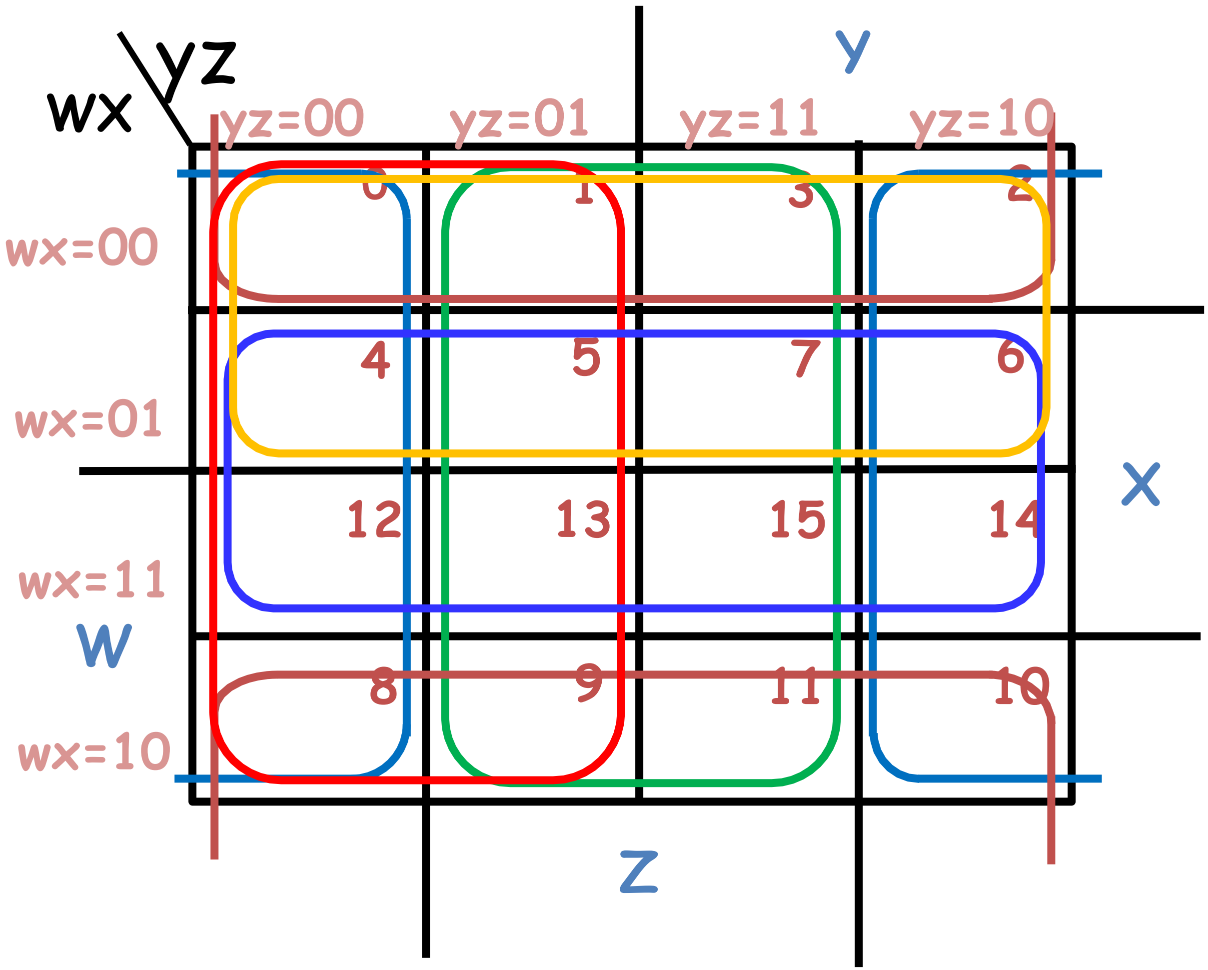
Four-Variable Maps

- ✓ Example Shapes of Rectangles:



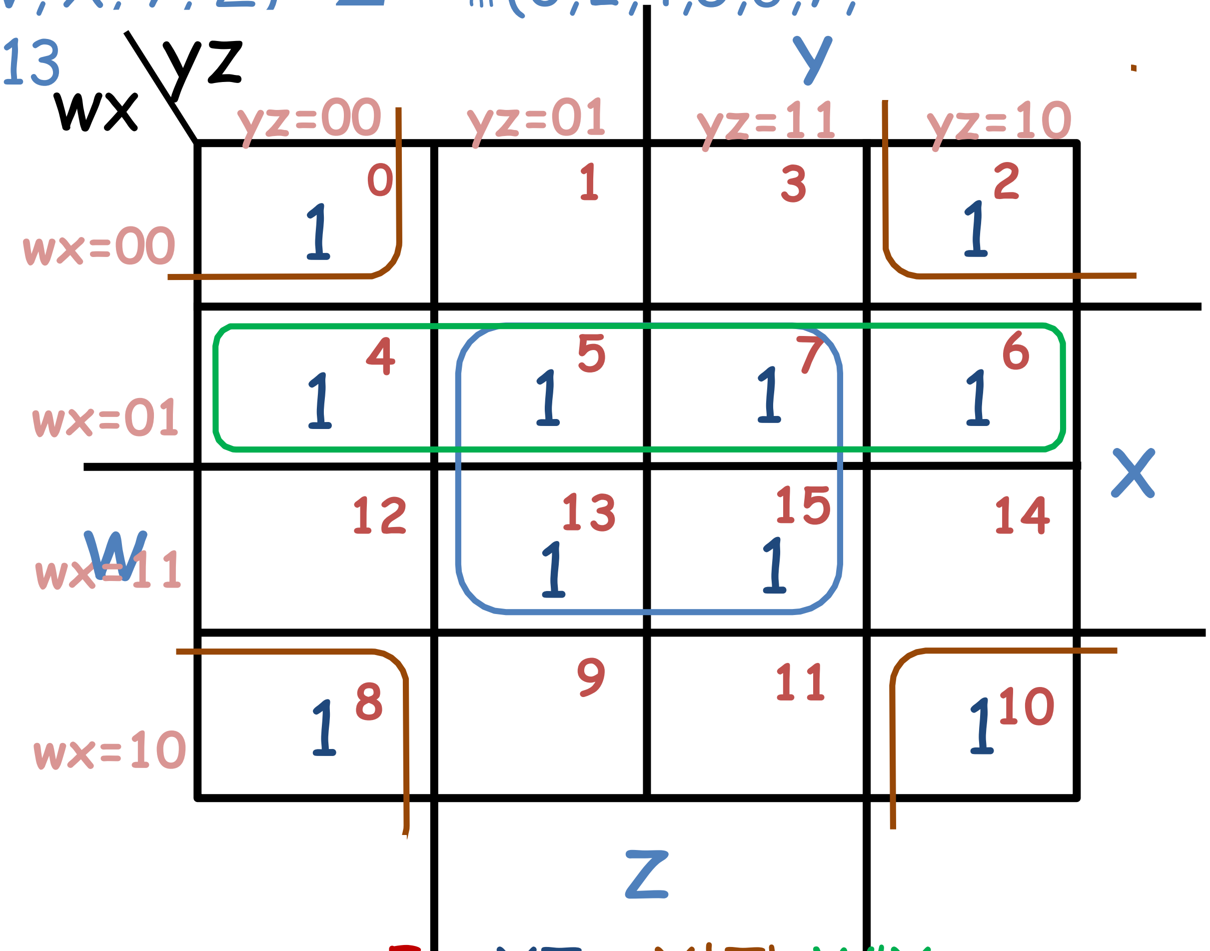
Four-Variable Maps

- ✓ Example Shapes of Rectangles:



Four-Variable Map Simplification

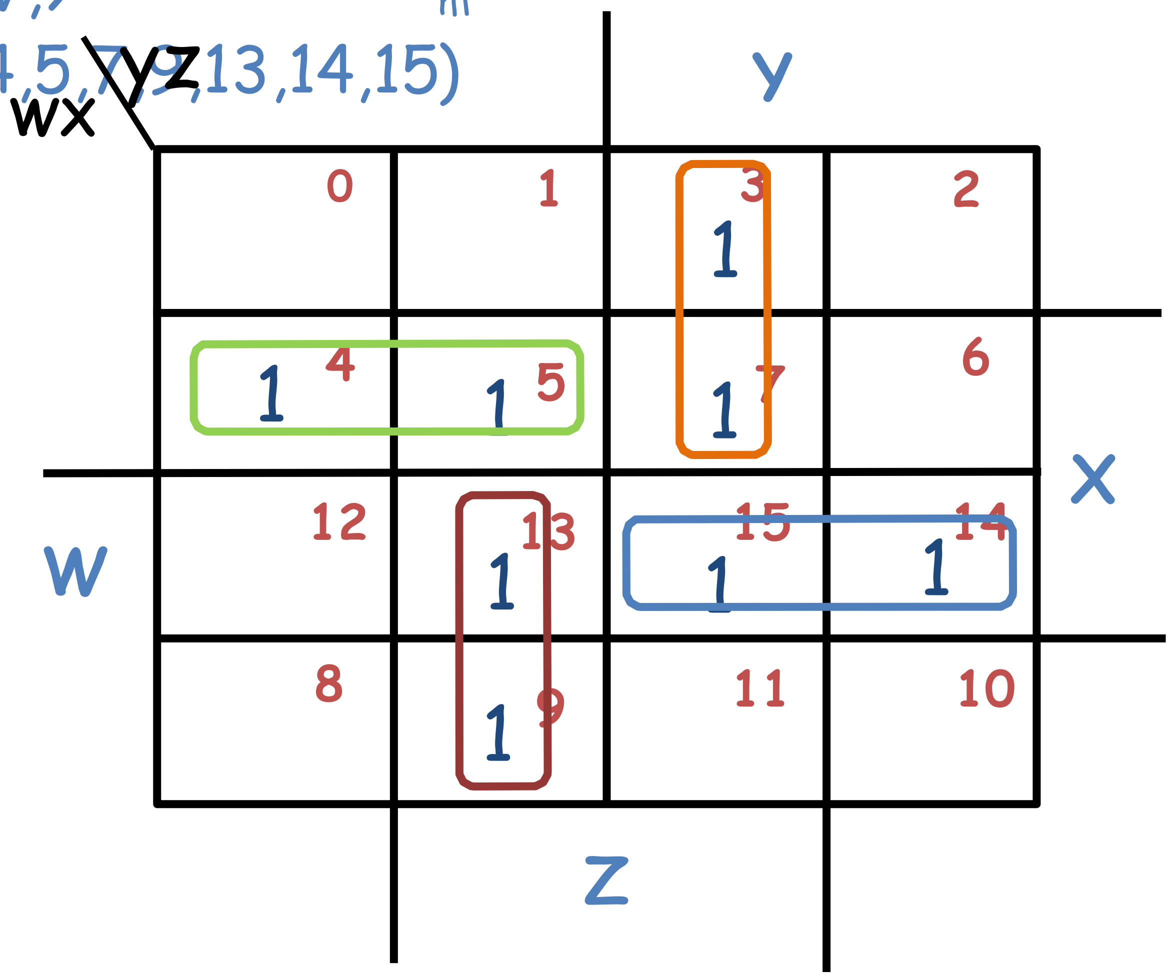
- $F(W,X,Y,Z) = \sum m(0,2,4,5,6,7,8,10,13$



$F = XZ + X'Z' + W'X$

Four-Variable Map Simplification

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



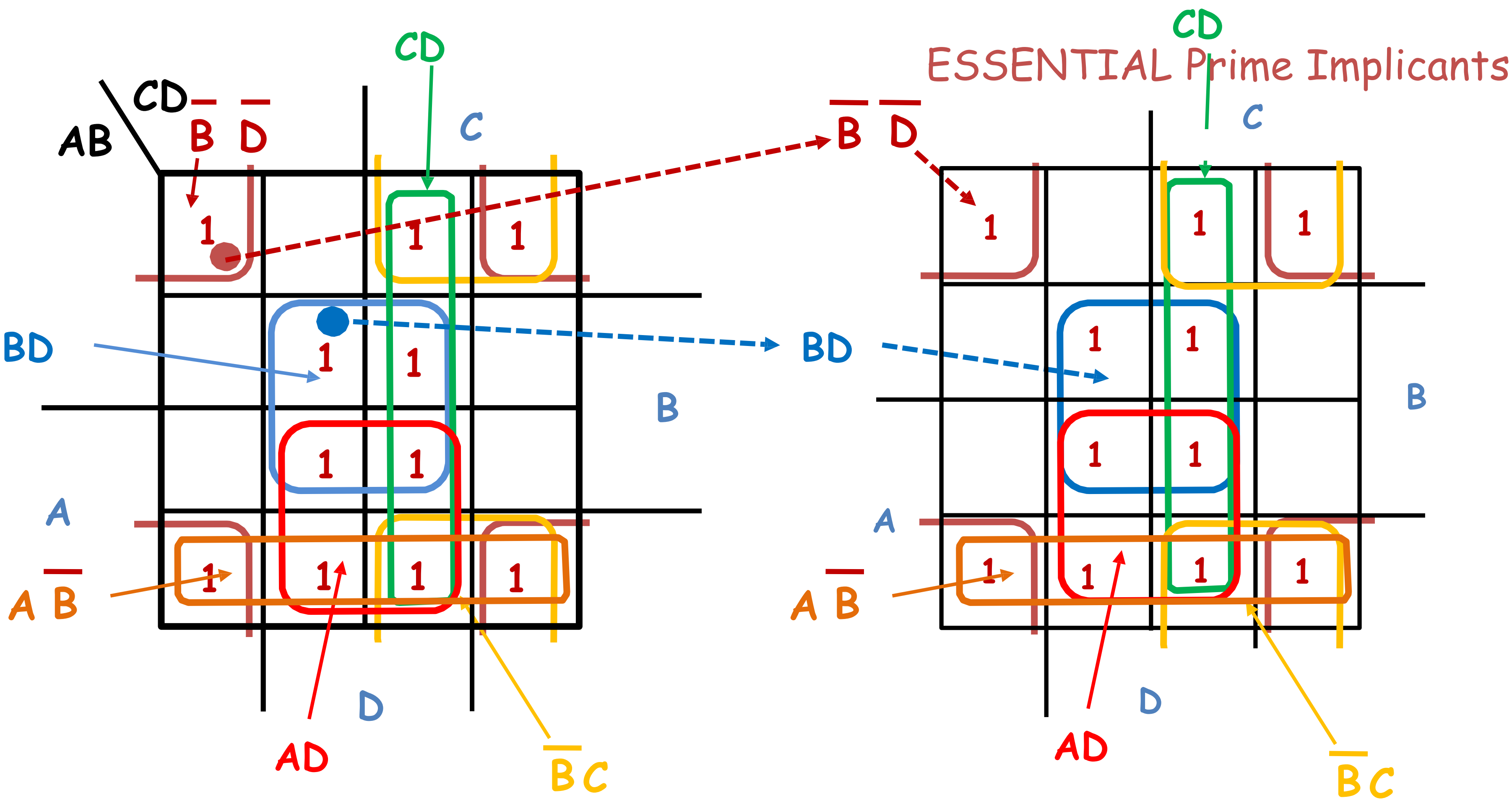
$F = W'XY' + W'YZ + WXY + WY'Z$

Systematic Simplification

- ✓A Prime Implicant is a cube i.e. 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

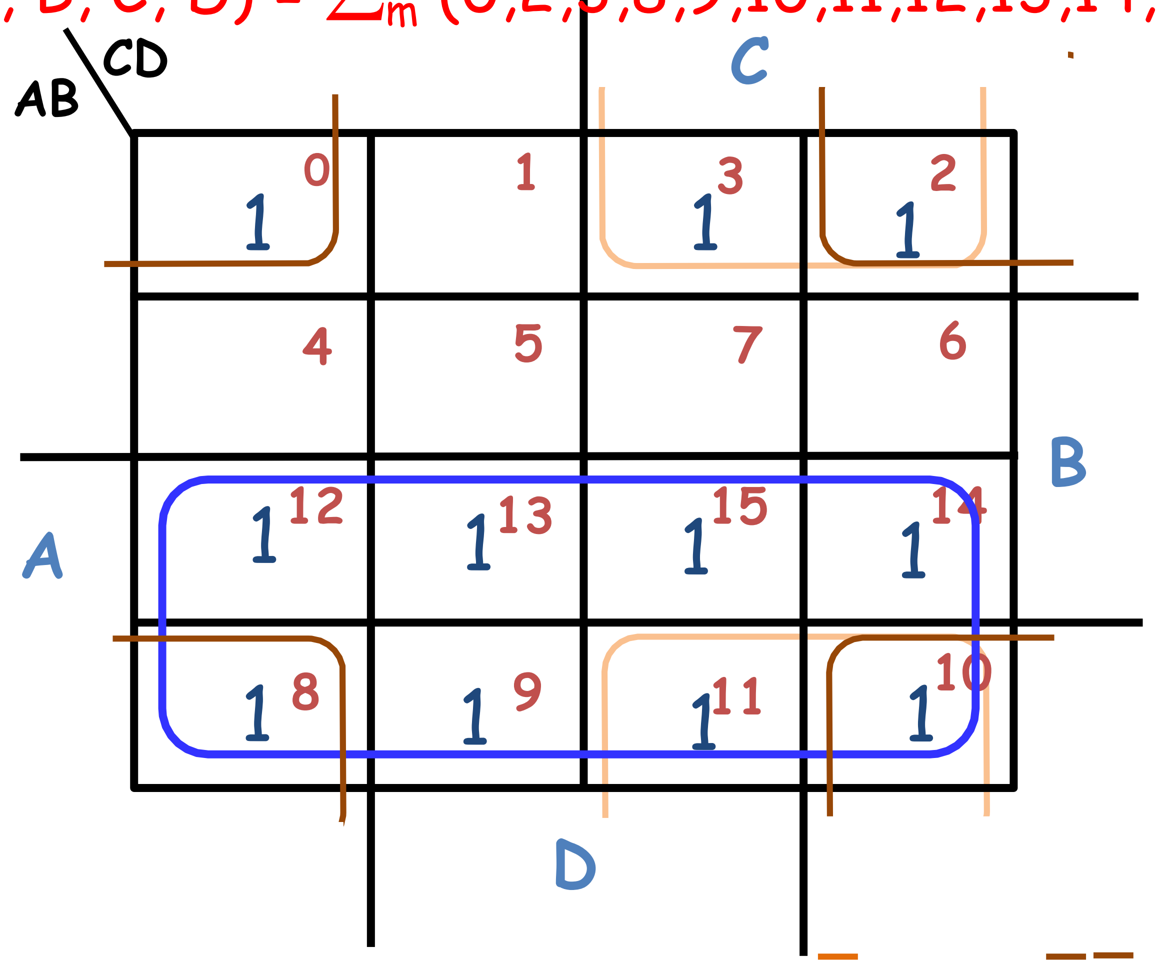
- ✓ Find ALL Prime Implicants



Minterms covered by single prime implicant

Prime Implicant Practice

- ✓ Find all prime implicants for:
- ✓ $F(A, B, C, D) = \sum_m (0,2,3,8,9,10,11,12,13,14,15)$

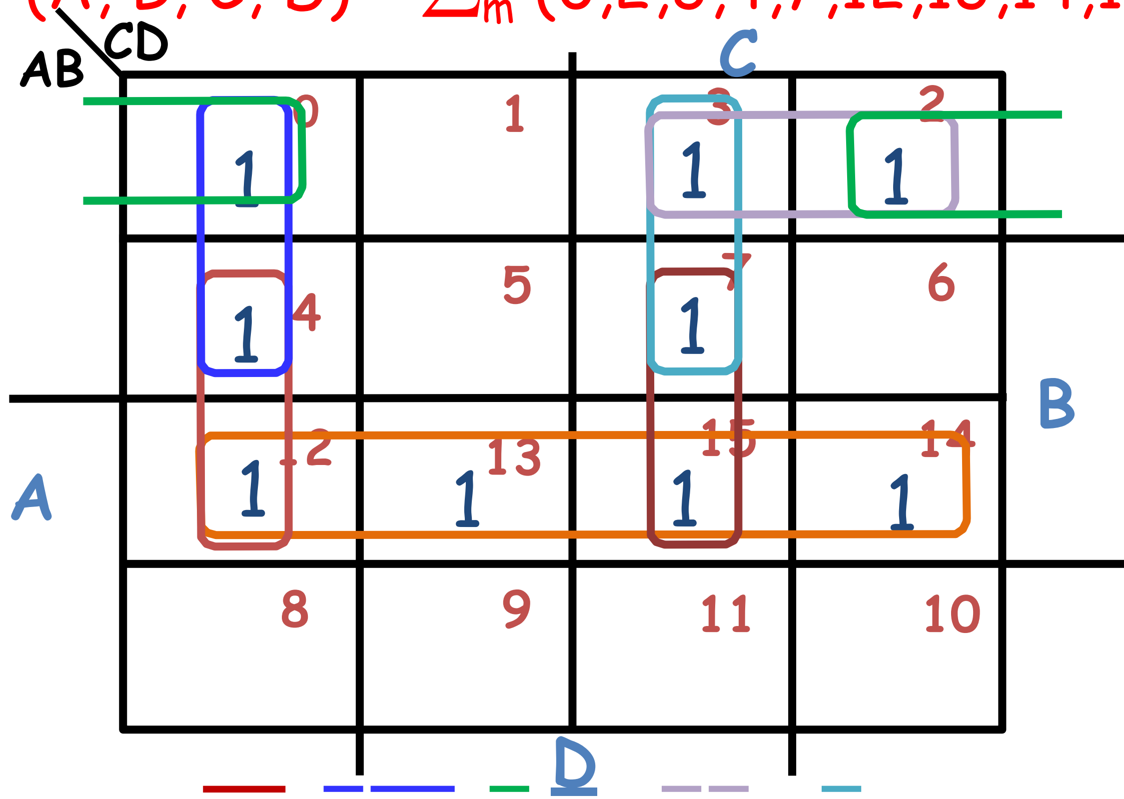


Prime implicants are: A, $\overline{B}C$, and $\overline{B}\,\overline{D}$

Another Example

✓ Find all prime implicants for:

$F(A, B, C, D) = \sum_m (0,2,3,4,7,12,13,14,15)$



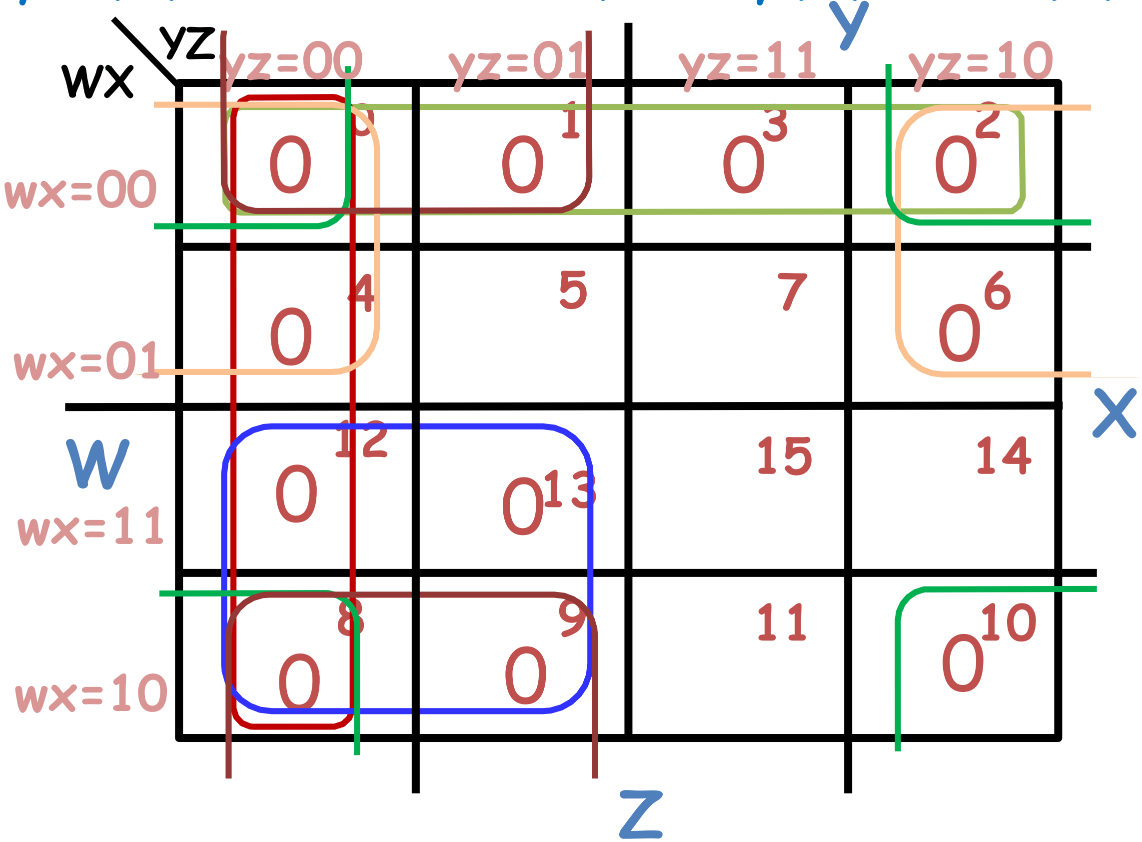
AB, $B\overline{CD}$, $\overline{A}\,\overline{CD}$, $\overline{A}BD$, $\overline{A}\,\overline{B}C$, $\overline{A}CD$, BCD

$\sum_m = AB + \overline{ACD} + \overline{A}CD + \overline{AB}C \quad \sum_c = \sum_m + \overline{A}\,\overline{BD} + BCD + B\overline{CD}$

K-Maps, implicates

- Find all prime implicates for:

$F=(w+y+z)\,(w'+x'+y)\,(x+y)\,(w+x+y')\,(w+x'+z)\,(w'+x+z)$



$\Pi_m = (w+x)(w'+y)(w+z)(x+z) \quad \Pi_c = \Pi_m (y+z)\,(y'+x')$

Don't Cares in K-Maps

- ✓ Sometimes a function table or map contains entries for which it is known that:
  - 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 need not be defined
- ✓ Instead, 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."

Incompletely Specified Functions

✓ $F = (f, d, r) : B^n \rightarrow \{0, 1, \mathbf{x}\}$
where **x** represents "don't care".

- f = onset function - $f(x)=1 \leftrightarrow F(x)=1$
- r = offset function - $r(x)=1 \leftrightarrow F(x)=0$
- d = don't care function - $d(x)=1 \leftrightarrow F(x)=*$

(f,d,r) forms a partition of B^n. i.e.

- $f + d + r = B^n$
- $fd = fr = dr = \emptyset$ (pairwise disjoint)