Karnaugh Maps

CSIM601251

Instructor: Tim Dosen DDAK
Slide By : Erdefi Rakun
Fasilkom UI

Outline:

- **Circuit optimization intro**
- Intro to K-Maps
- Three variable maps
- Four variable maps
- Larger K-Maps
- Prime implicants and essential prime implicants
- Don't care in K-maps
- Selection rule

Note: Portion of this material are taken from Aaron Tan's slide and other portions of this material © 2008 by Pearson Education,Inc

Function Simplification

- Why simplify?
  - Simpler expression uses fewer logic gates.
  - Thus cheaper, uses less power, (sometimes) faster.
- Techniques
  - Algebraic
    - Using theorems
    - Open-ended; requires skills
  - Karnaugh Maps
    - Easy to use
    - Limited to no more than 6 variables
  - Quine-McCluskey
    - Suitable for automation
    - Can handle many variables (but computationally intensive)

Algebraic Simplification

- Example 1: Simplify $(x+y)\cdot(x+y')\cdot(x'+z)$

$(x+y)\cdot(x+y')\cdot(x'+z)$	
$= (x\cdot x + x\cdot y' + x\cdot y + y\cdot y')\cdot(x'+z)$	(associativity)
$= (x + x\cdot(y'+y) + 0)\cdot(x'+z)$	(idemp, assoc., complement)
$= (x + x\cdot 1)\cdot(x'+z)$	(complement, identity)
$= (x + x)\cdot(x'+z)$	(identity)
$= x\cdot(x'+z)$	(idempotency)
$= x\cdot x' + x\cdot z$	(associativity)
$= 0 + x\cdot z$	(complement)
$= x\cdot z$	(identity)

- Number of literals reduced from 6 to 2.

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
- Examples:
  - $F = BD + A B' C + A C' D'$ L = 8
  - $F = BD + A B' C + A B' D' + AB C'$ L =
  - $F = (A + B)(A + D)(B + C + D')(B' + C' + D)$ L =
  - Which solution is best?

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 excluding single literal terms,(G) and
  - optionally, the number of distinct complemented single literals (GN).
- Example:
  - $F = A\,B\,C\,D + A'\,B'\,C'\,D'$ G = 10, GN = 14
  - $F = (A'+B)(B'+C)(C'+D)(D'+A)$ G = , GN =
  - Which solution is best?

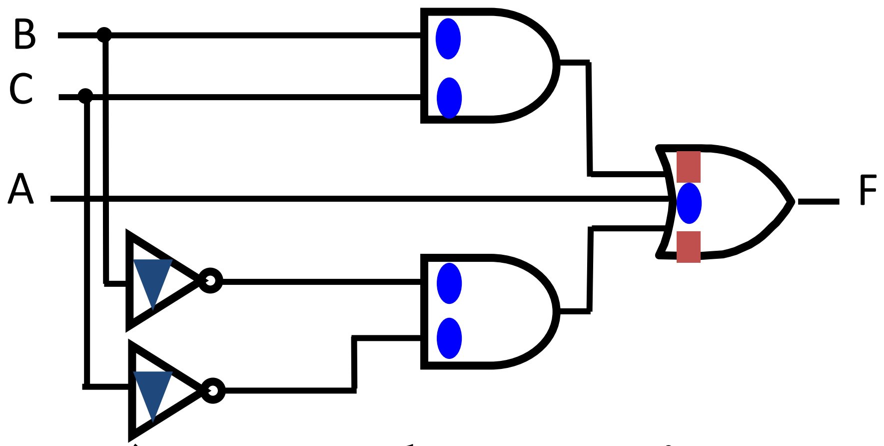
Cost Criteria (continued)

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

GN = G + 2 = 9

L = 5

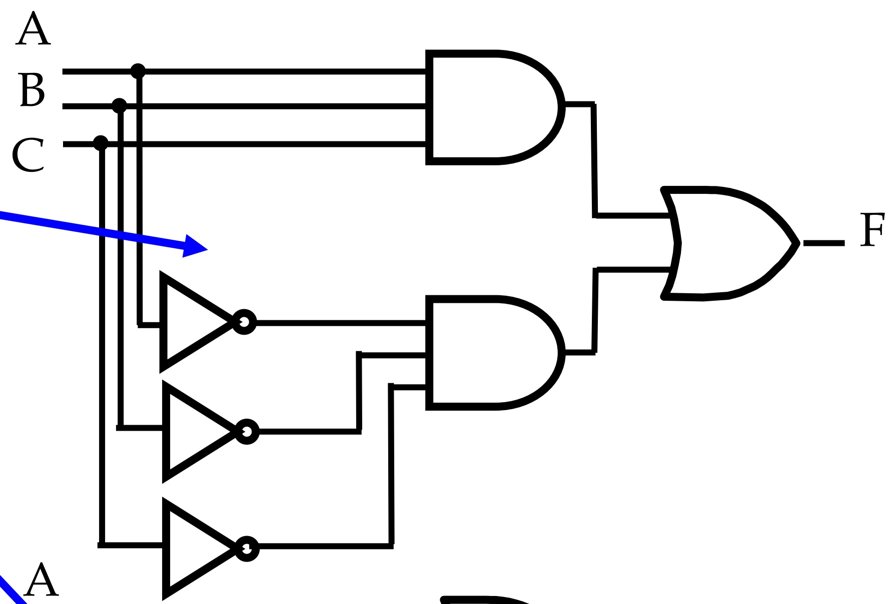
G = L + 2 = 7

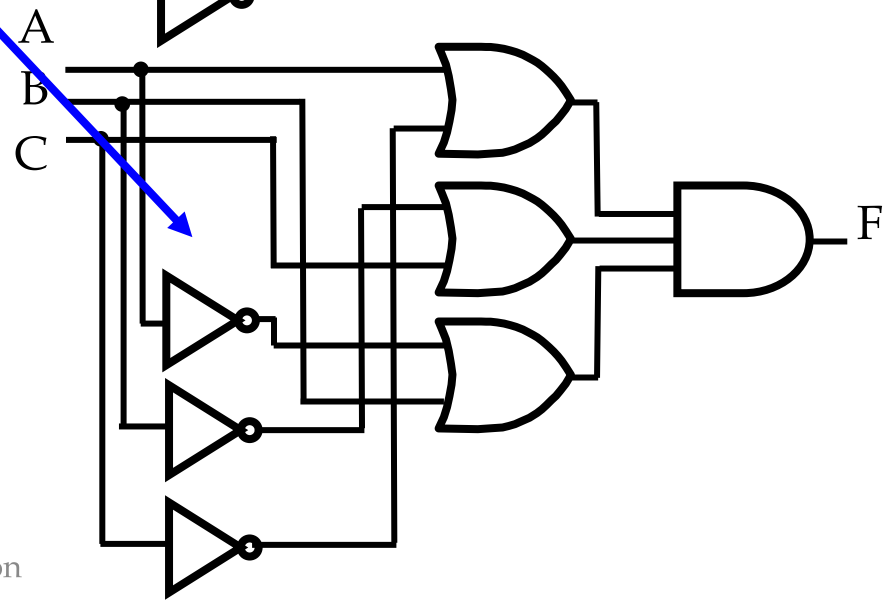


- L (literal count) counts the AND inputs and the single literal OR input.
- G (gate input count) adds the remaining OR gate inputs
- GN(gate input count with NOTs) adds the inverter inputs

Cost Criteria (continued)

- Example 2:
- F = A B C + A′ B′ C′
- L = 6 G = 8 GN = 11
- F = (A + C′)(B′+ C)(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 first.
- Introduce a graphical technique using Karnaugh maps (K-maps, for short)

Outline:

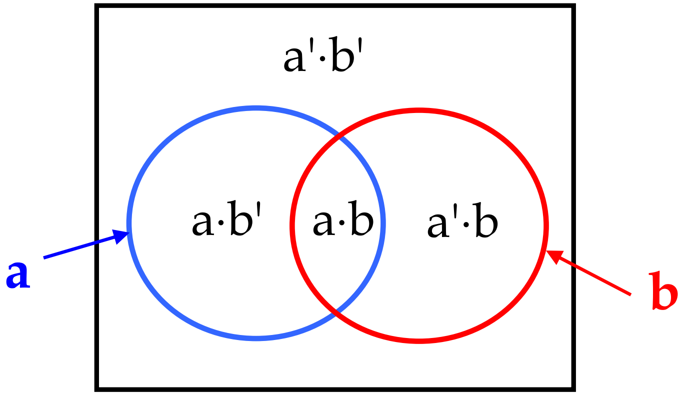
- Circuit optimization intro
- **Intro to K-Maps**
- Three variable maps
- Four variable maps
- Larger K-Maps
- Prime implicants and essential prime implicants
- Don’t care in K-maps
- Selection rule

Introduction to K-Maps

- Systematic method to obtain simplified (minimal) sum-of-products (SOP) expressions.
- Objective: *Fewest* possible product terms and literals.
- Diagrammatic technique based on a special form of *Venn diagram*.
- Advantage: Easy to use.
- Disadvantage: Limited to 5 or 6 variables.

Venn Diagram

- Example: 2 variables a and b represented by 2 circular regions. There are 4 minterms, each occupying their respective space.



- A set of minterms represents a certain Boolean function. Examples:

{ a·b, a·b' }	→ a·b + a·b' = a·(b+b') = a
{ a'·b, a·b }	→ a'·b + a·b = (a'+a)·b = b
{ a·b }	→ a·b
{ a·b, a·b', a'·b }	→ a·b + a·b' + a'·b = a + b
{ }	→ 0
{ a'·b', a·b, a·b', a'·b }	→ 1

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
- The K-map can be viewed as
  - A reorganized version of the truth table
  - A topologically-warped Venn diagram as used to visualize sets in algebra of sets

Some Uses of K-Maps

- Provide a means for:
  - 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 m0 and minterm m1 are "adjacent" and differ in the value of the variable y
  - Similarly, minterm m0 and minterm m2 differ in the x variable.
  - Also, m1 and m3 differ in the x variable as well.
  - Finally, m2 and m3 differ in the value of the variable y

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

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

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:

$$G(x,y) = \left(x\,\overline{y} + x\,y\right) + \left(x\,y + \overline{x}\,y\right) = x + y$$

Duplicate x y

Outline:

- Circuit optimization intro
- Intro to K-Maps
- **Three variable maps**
- Four variable maps
- Larger K-Maps
- Prime implicants and essential prime implicants
- Don't care in K-maps
- Selection rule

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

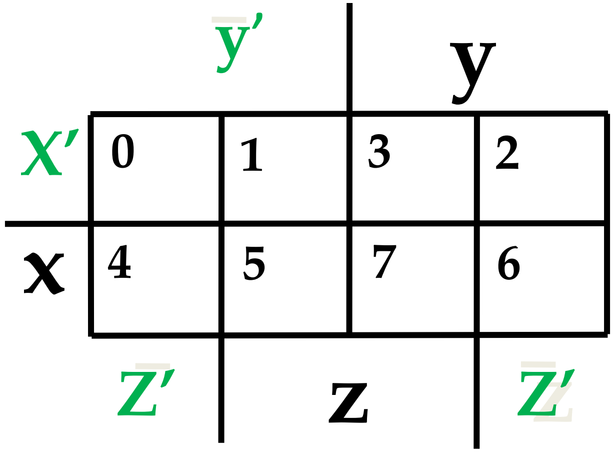
- Where each minterm corresponds to the product terms:

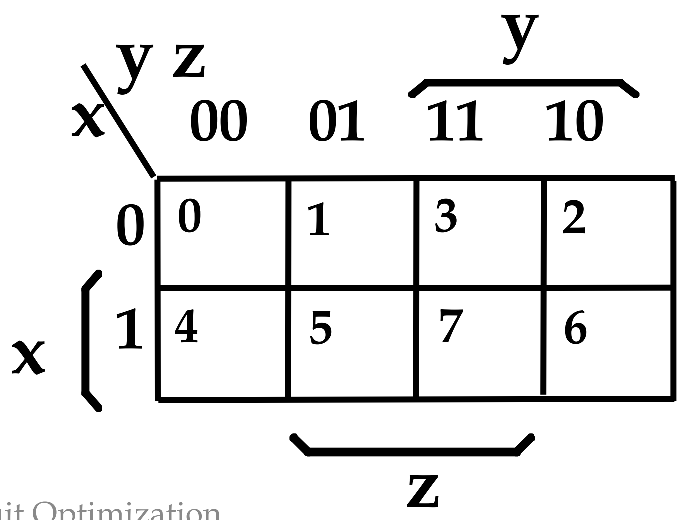
	yz=00	yz=01	yz=11	yz=10
x=0	$\bar{x}\,\bar{y}\,\bar{z}$	$\bar{x}\,\bar{y}\,z$	$\bar{x}\,y\,z$	$\bar{x}\,y\,\bar{z}$
x=1	$x\,\bar{y}\,\bar{z}$	$x\,\bar{y}\,z$	$x\,y\,z$	$x\,y\,\bar{z}$

- Note that if the binary value for an <u>index</u> 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 Functions

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

$$F(x, y, z) = \Sigma m(2,3,4,5)$$

			y	y
	0	1	3 1	2 1
x	4 1	5 1	7	6
		z	z	

- Example:

$$G(a, b, c) = \Sigma m(3,4,6,7)$$

			y	y
	0	1	3 1	2
x	4 1	5	7 1	6 1
		z	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

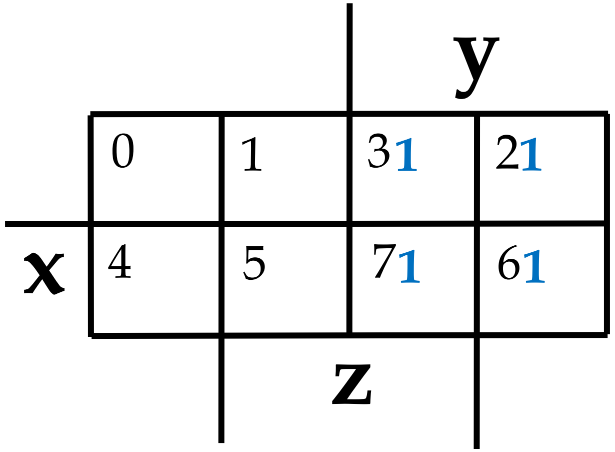
Combining Squares

- 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 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: Let

$$F = \Sigma m(2,3,6,7)$$



- Applying the Minimization Theorem three times:

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

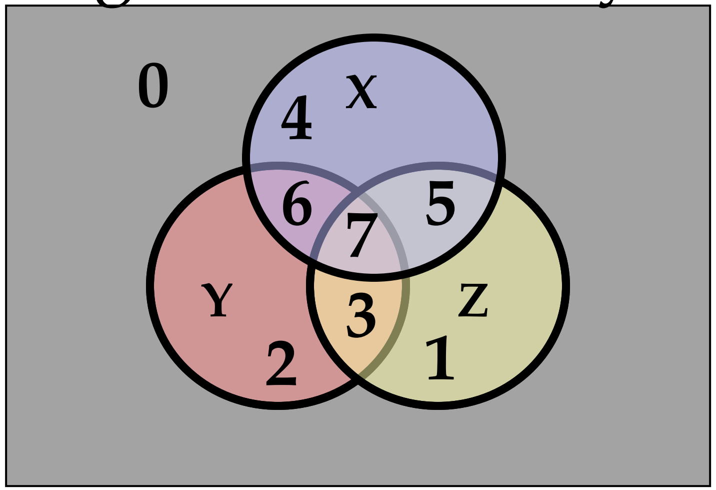
- Thus the four terms that form a 2 × 2 square correspond to the term "y".

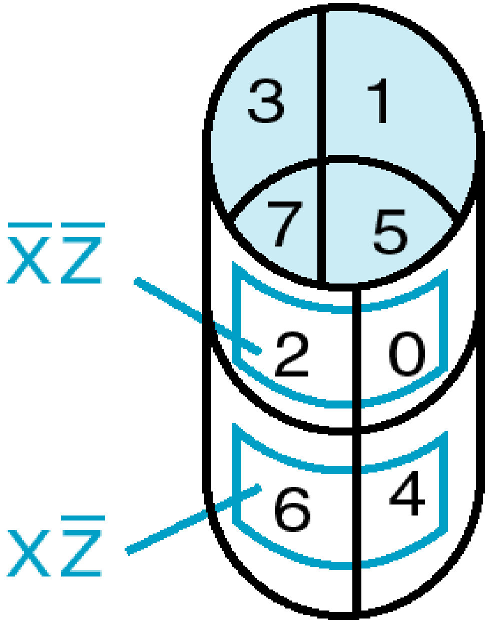
Three-Variable Maps

- 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; 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

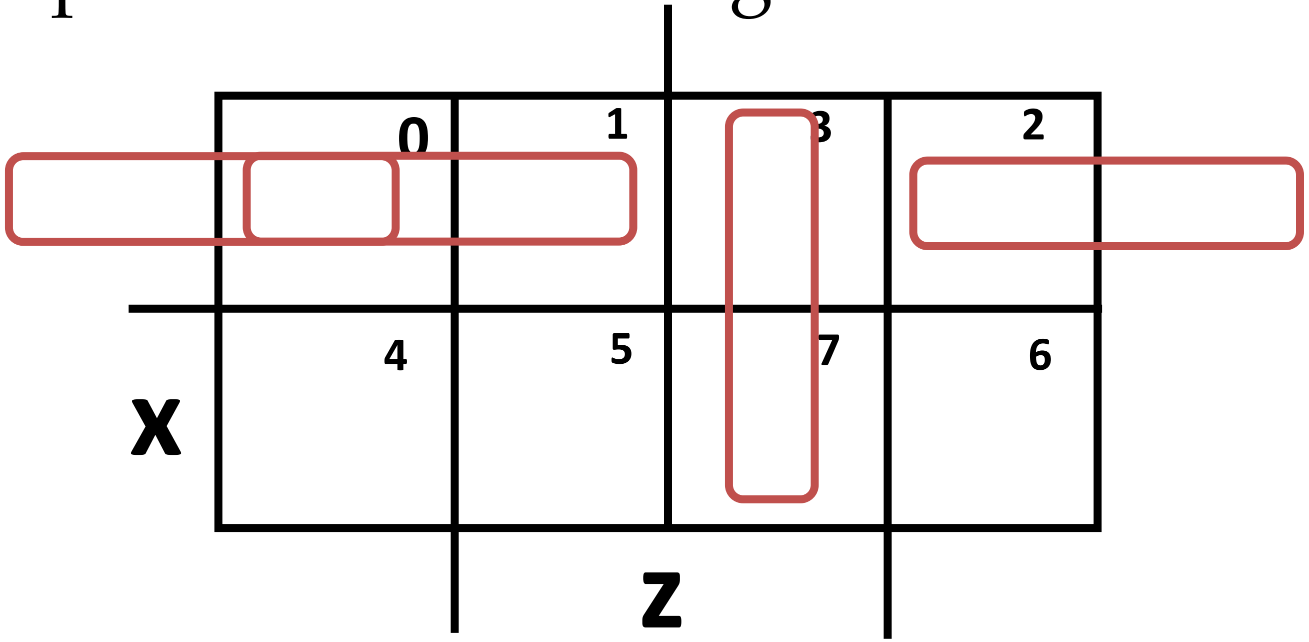
- Topological warps of 3-variable K-maps that show *all* adjacencies:
  - Venn Diagram
  - Cylinder





Three-Variable Maps

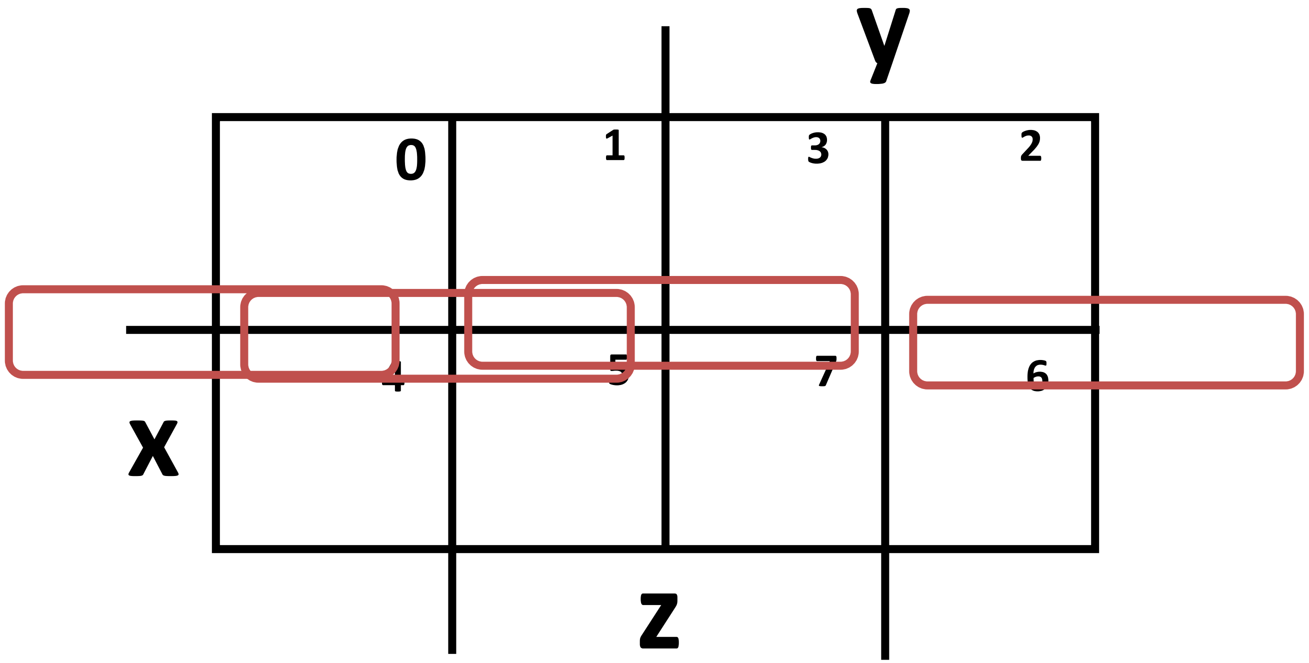
- Example Shapes of 2-cell Rectangles:



- Read off the product terms for the rectangles shown

Three-Variable Maps

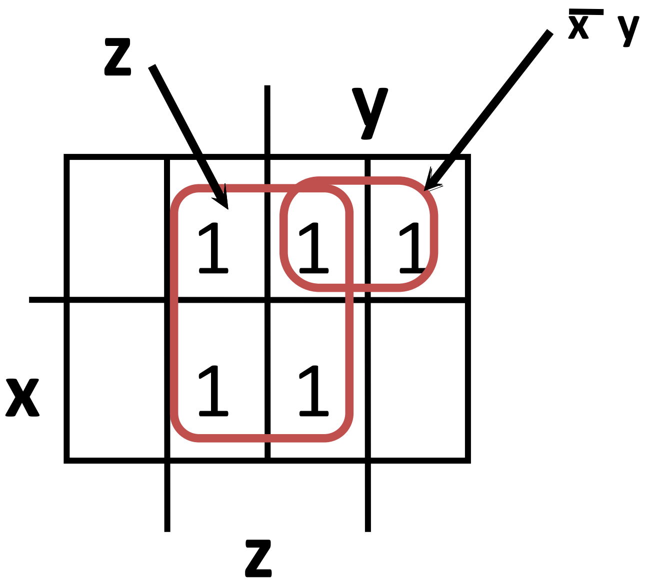
- Example Shapes of 4-cell Rectangles:



- 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 $\mathbf{F(x, y, z) = \Sigma_m(1,2,3,5,7)}$



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

Three-Variable Map Simplification

- Use a K-map to find an optimum SOP equation for

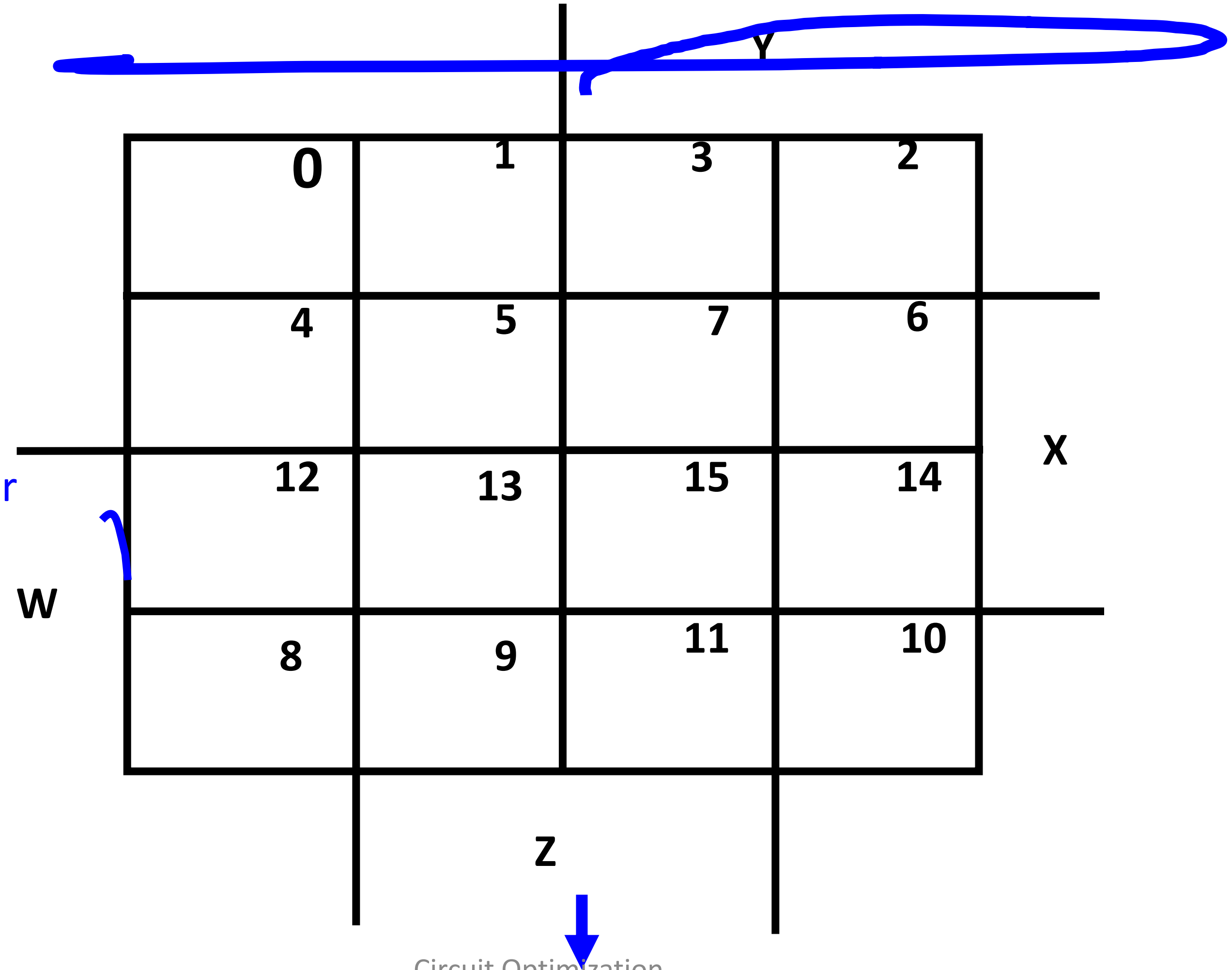
$$F(X, Y, Z) = \Sigma m(0,1,2,4,6,7)$$

Outline:

- Circuit optimization intro
- Intro to K-Maps
- Three variable maps
- **Four variable maps**
- Larger K-Maps
- Prime implicants and essential prime implicants
- Don’t care in K-maps
- Selection rule

Four Variable Maps

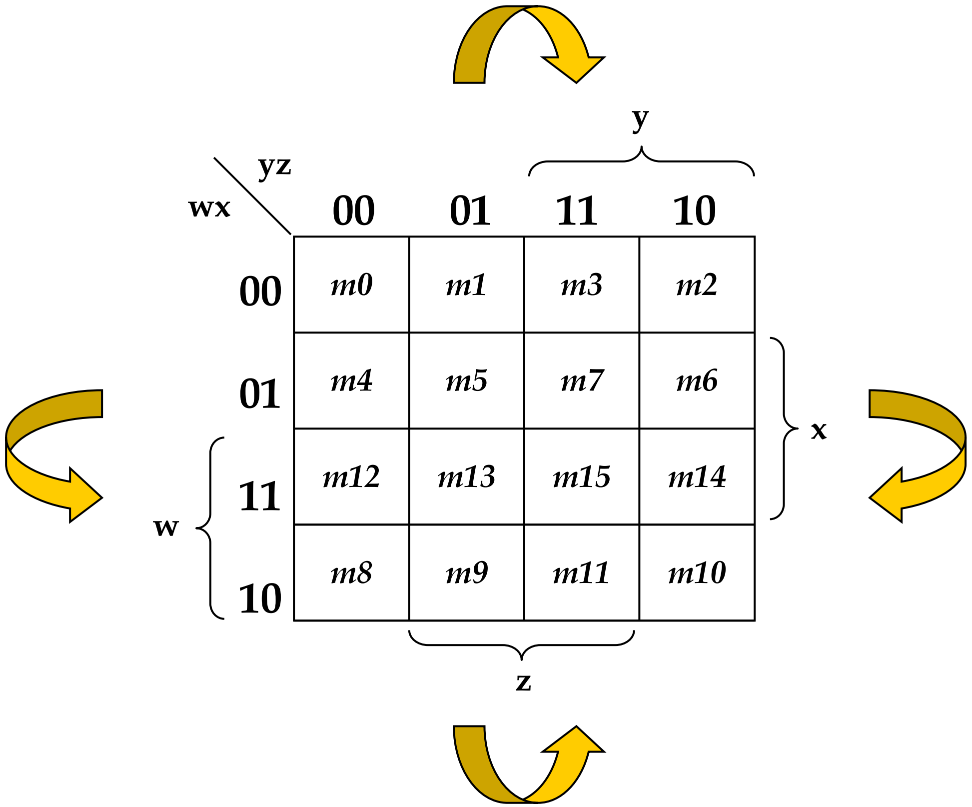
- Map and location of minterms:



Variable Order

4-Variable K-Maps (1/2)

- There are 16 square cells in a 4-variable K-map.
- Example: Let the variables be w, x, y, z.

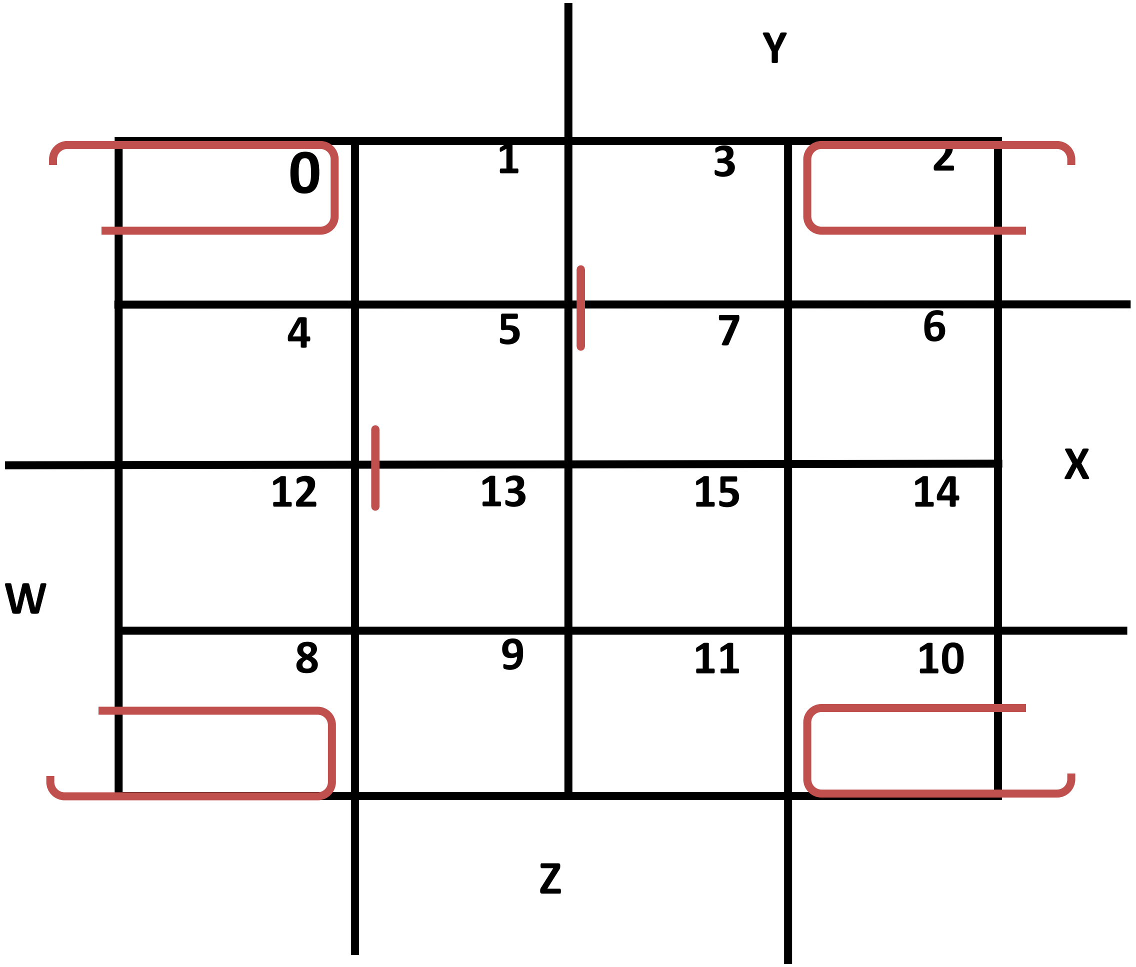


Four Variable Terms

- 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")

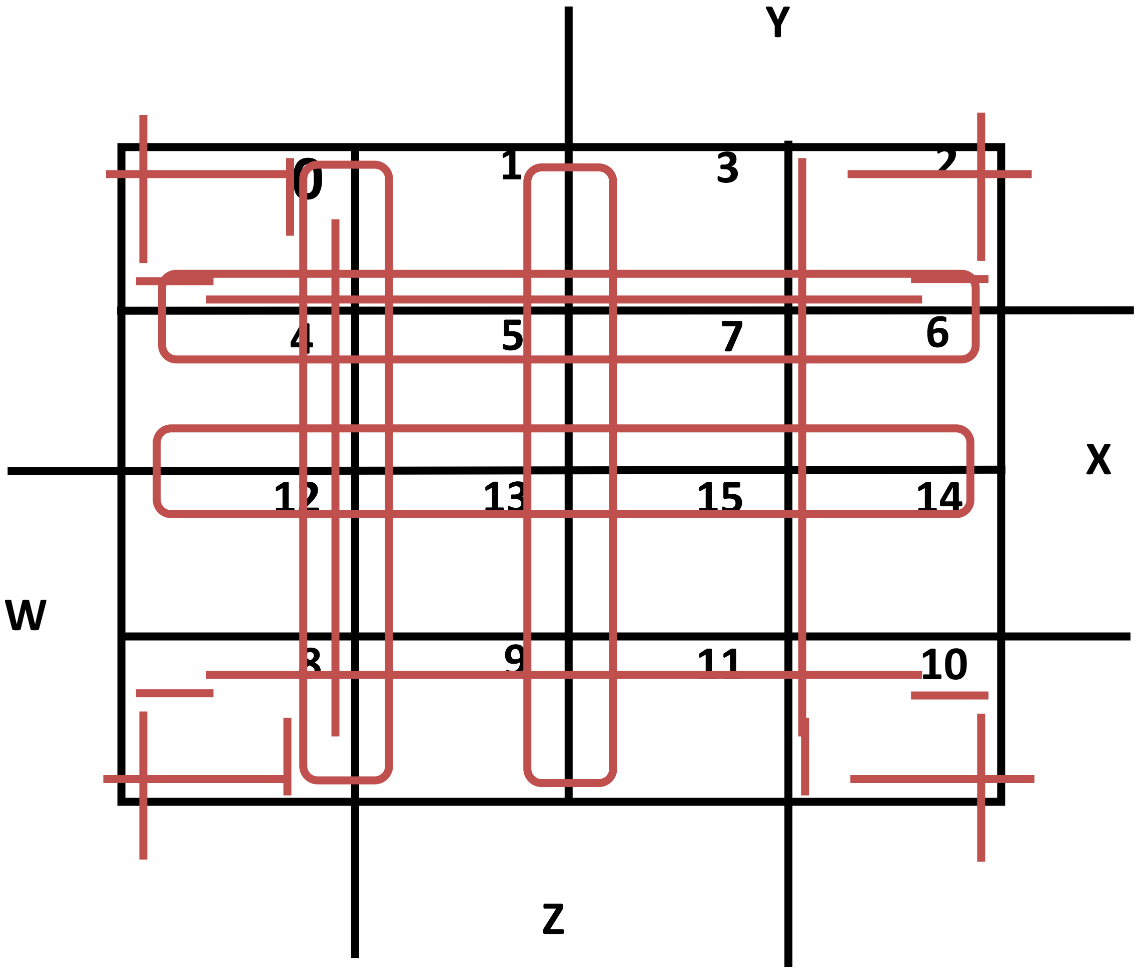
Four-Variable Maps

- Example Shapes of Rectangles:



Four-Variable Maps

- Example Shapes of Rectangles:



Four-Variable Map Simplification

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

Four-Variable Map Simplification

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

Outline:

- Circuit optimization intro
- Intro to K-Maps
- Three variable maps
- Four variable maps
- **Larger K-Maps**
- Prime implicants and essential prime implicants
- Don't care in K-maps
- Selection rule

Note: Portion of this material are taken from Aaron Tan's slide and other portions of this material © 2008 by Pearson Education,Inc

5-Variable K-Maps (1/2)

- Organised as two 4-variable K-maps. One for v' and the other for v.

v '

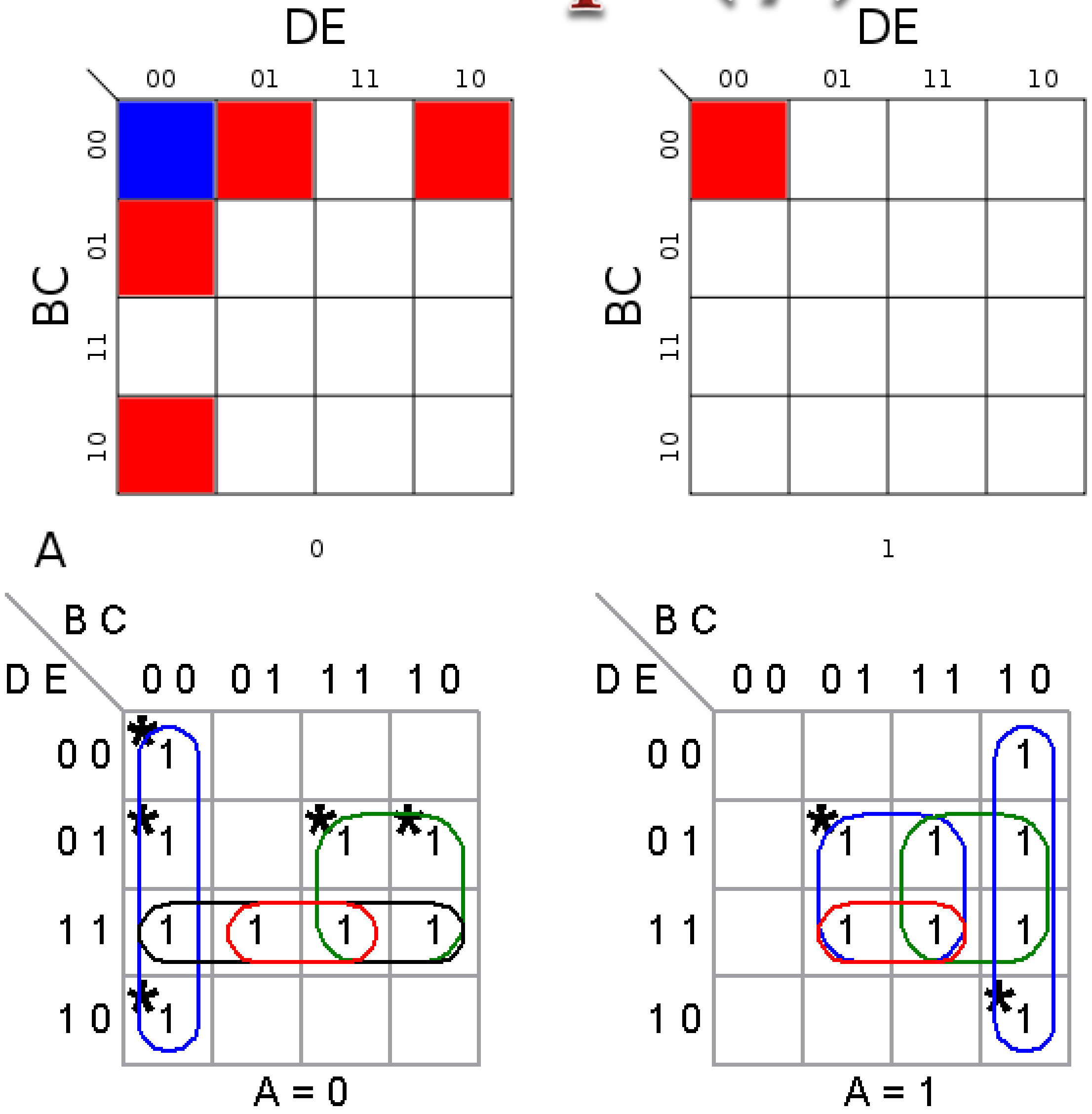
wx \ yz	00	01	11	10
00	*m0*	*m1*	*m3*	*m2*
01	*m4*	*m5*	*m7*	*m6*
11	*m12*	*m13*	*m15*	*m14*
10	*m8*	*m9*	*m11*	*m10*

v

wx \ yz	00	01	11	10
00	*m16*	*m17*	*m19*	*m18*
01	*m20*	*m21*	*m23*	*m22*
11	*m28*	*m29*	*m31*	*m30*
10	*m24*	*m25*	*m27*	*m26*

Corresponding squares of each map are adjacent.
Can visualise this as *one 4-variable K-map* being on TOP *of the other 4-variable K-map*.

5-Variable K-Maps (2/2)



Larger K-Maps (1/2)

- 6-variable K-map is pushing the limit of human’s “pattern-recognition” capability.
- K-maps larger than 6 variables are practically unheard of!
- Normally, a 6-variable K-map is organised as four 4-variable K-maps, mirrored along two axes.

Larger K-Maps (2/2)

b

a'·b'

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

a'·b

10	11	01	00	ef / cd
m18	*m19*	*m17*	*m16*	00
m22	*m23*	*m21*	*m20*	01
m30	*m31*	*m29*	*m28*	11
m26	*m27*	*m25*	*m24*	10

a

cd \ ef	00	01	11	10
10	*m40*	*m41*	*m43*	*m42*
11	*m44*	*m45*	*m47*	*m46*
01	*m36*	*m37*	*m39*	*m38*
00	*m32*	*m33*	*m35*	*m34*

a·b'

10	11	01	00	ef / cd
m58	*m59*	*m57*	*m56*	10
m62	*m63*	*m61*	*m60*	11
m54	*m55*	*m53*	*m52*	01
m50	*m51*	*m49*	*m48*	00

a·b

Try stretch your recognition capability by finding simplest sum-of-products expression for $\sum$ m(6,8,14,18,23,25,27,29,41,45,57,61).

Outline:

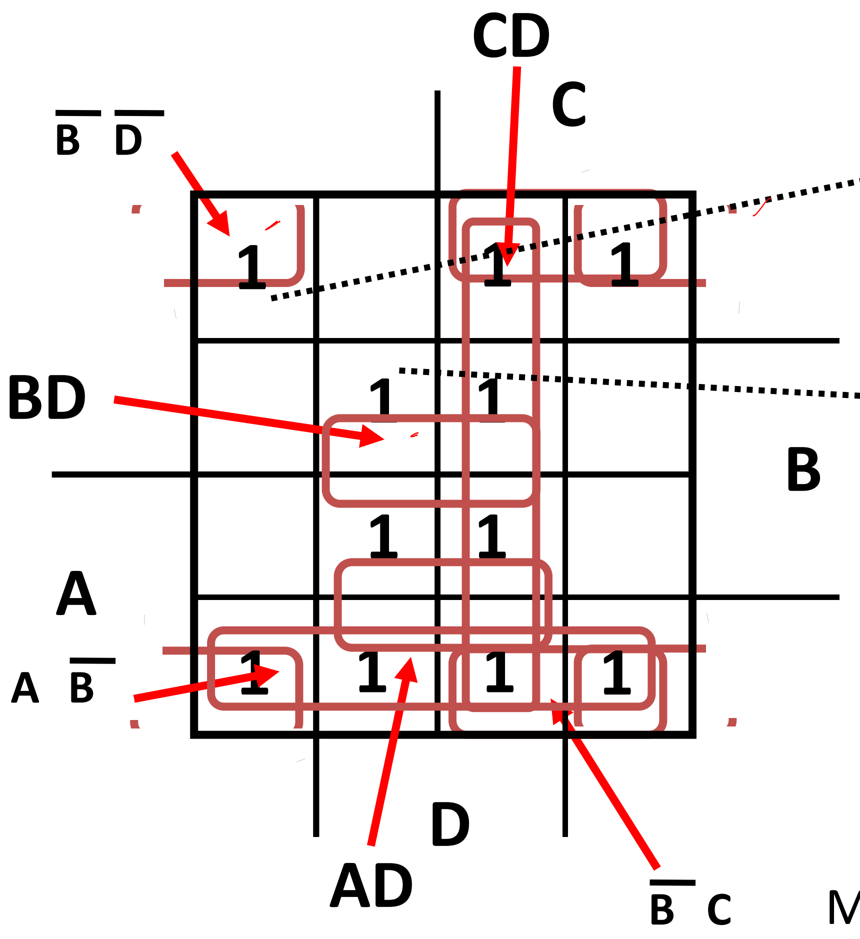
- Circuit optimization intro
- Intro to K-Maps
- Three variable maps
- Four variable maps
- Larger K-Maps
- **Prime implicants and essential prime implicants**
- Don't care in K-maps
- Selection rule

Systematic Simplification

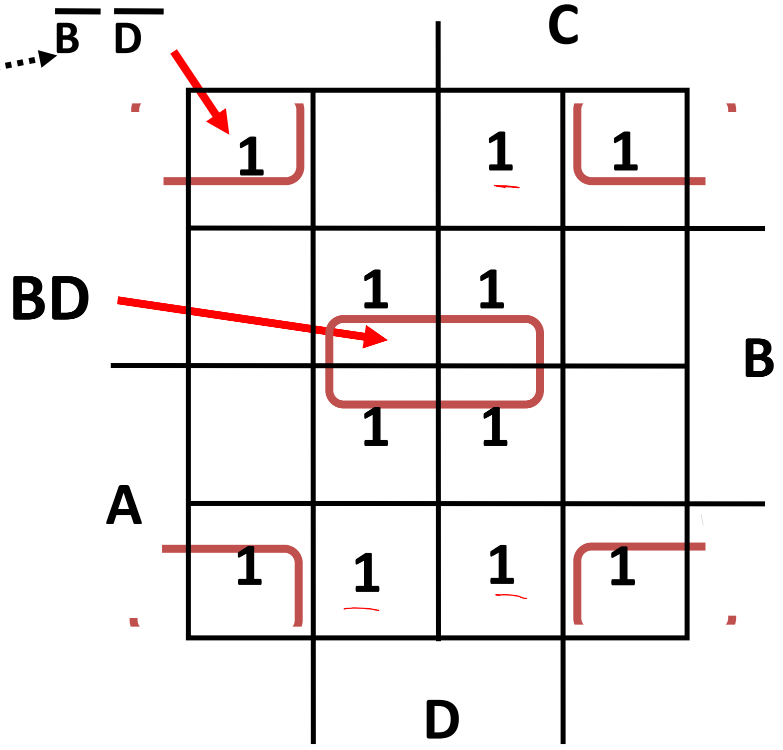
- A *Prime Implicant* is a product term obtained by combining the maximum possible number of adjacent squares in the map into a rectangle with the number of squares a power of 2.
- A prime implicant is called an *Essential Prime Implicant* if it is the only prime implicant that covers (includes) one 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



ESSENTIAL Prime Implicants

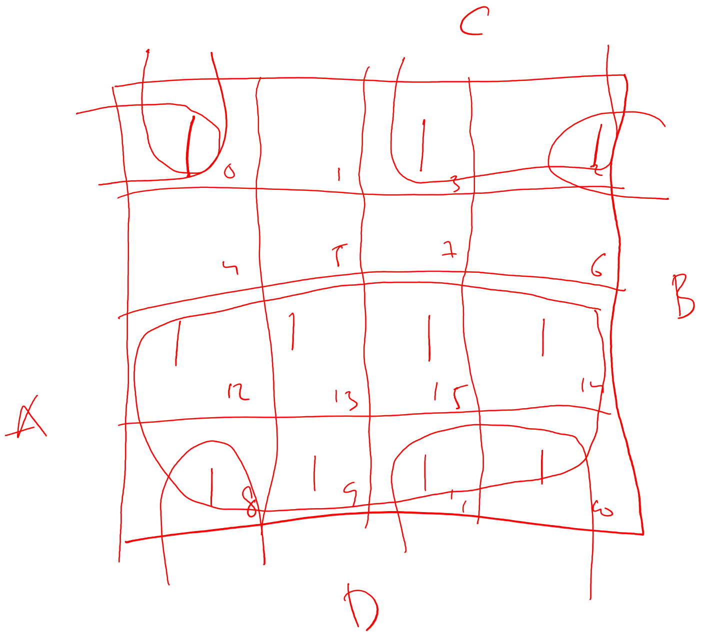


Minterms covered by single prime implicant

Prime Implicant Practice

- Find all prime implicants for:

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



A

$\bar{B}C$

$\bar{A}\bar{B}\bar{D}$

$\bar{B}\bar{C}\bar{D}$

Another Example

- Find all prime implicants for:

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

  – Hint: There are seven prime implicants!

Outline:

- Circuit optimization intro
- Intro to K-Maps
- Three variable maps
- Four variable maps
- Larger K-Maps
- Prime implicants and essential prime implicants
- **Don’t care in K-maps**
- Selection rule

Don't Cares in K-Maps

- 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 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 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 <u>never occur</u>, so the output values for these codes are "x" to represent "don't cares."

Don't Cares in K-Maps

- Example 2: A circuit that represents a very common situation that occurs in computer design has two distinct sets of input variables:
  - A, B, and C which take on all possible combinations, and
  - Y which takes on values 0 or 1.

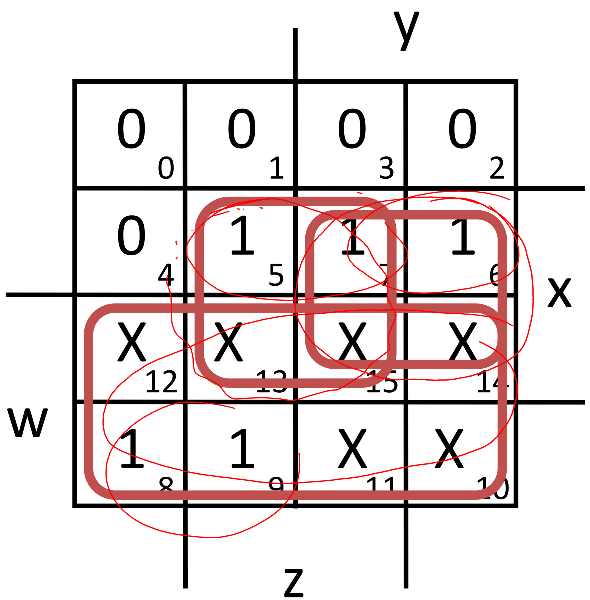
  and a single output Z. The circuit that receives the output Z observes it only for combinations of A, B, and C such A = 1 and B = 1 or C = 0, otherwise ignoring it. Thus, Z is specified only for those combinations, and for all other combinations of A, B, and C, Z is a don't care. Specifically, Z must be specified for $AB + \overline{C} = 1$, and is a don't care for :

$$\overline{AB + \overline{C}} = (\overline{A} + \overline{B})C = \overline{A}C + \overline{B}C = 1$$

- Ultimately, each don't care "x" entry may take on either a 0 or 1 value in resulting solutions
- For example, an "x" may take on value "0" in an SOP solution and value "1" in a POS solution, or vice-versa.
- Any minterm with value "x" need not be covered by a prime implicant.

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:



$F1\ (w,x,y,z) = w + x\,z + x\,y \quad G = 7$

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

$F_2(w, x, y, z) = \overline{w}\,x\,z + \overline{w}\,x\,y + w\,\overline{x}\,\overline{y} \quad G = 12$

» For this particular function, cost G for the POS solution for $F_1(w,x,y,z)$ is not changed by using the don't cares.

Product of Sums Example

- Find the optimum POS solution:

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

  - Hint: Use $\overline{F}$ and complement it to get the result.

Outline:

- Circuit optimization intro
- Intro to K-Maps
- Three variable maps
- Four variable maps
- Larger K-Maps
- Prime implicants and essential prime implicants
- Don't care in K-maps
- **Selection rule**

Optimization Algorithm

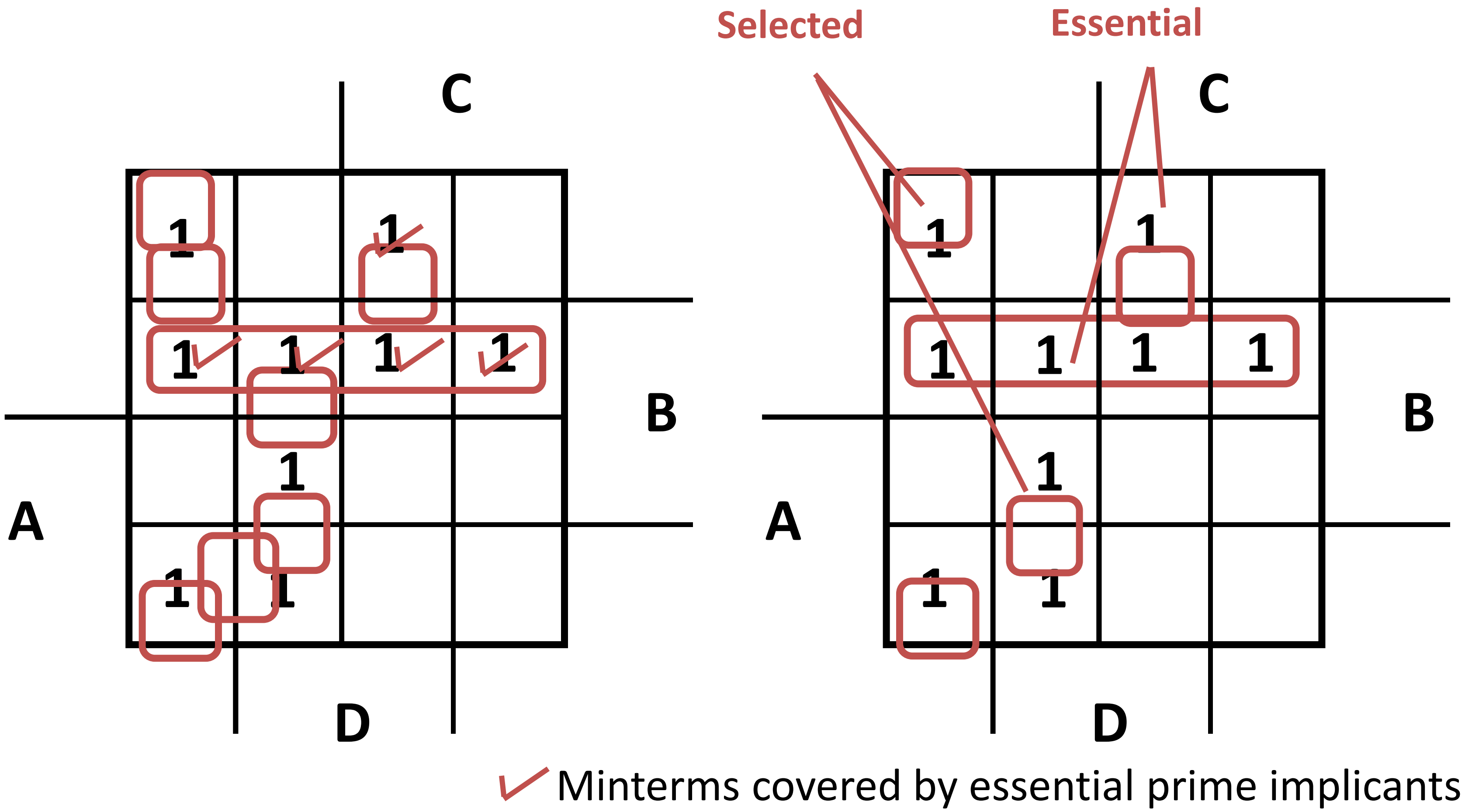
- Find all prime implicants.
- Include all essential prime implicants in the solution
- Select a minimum cost set of non-essential prime implicants to cover all minterms not yet covered:
  - Obtaining an optimum solution: See Reading Supplement - More on Optimization
  - Obtaining a good simplified solution: Use the Selection Rule

Prime Implicant 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.

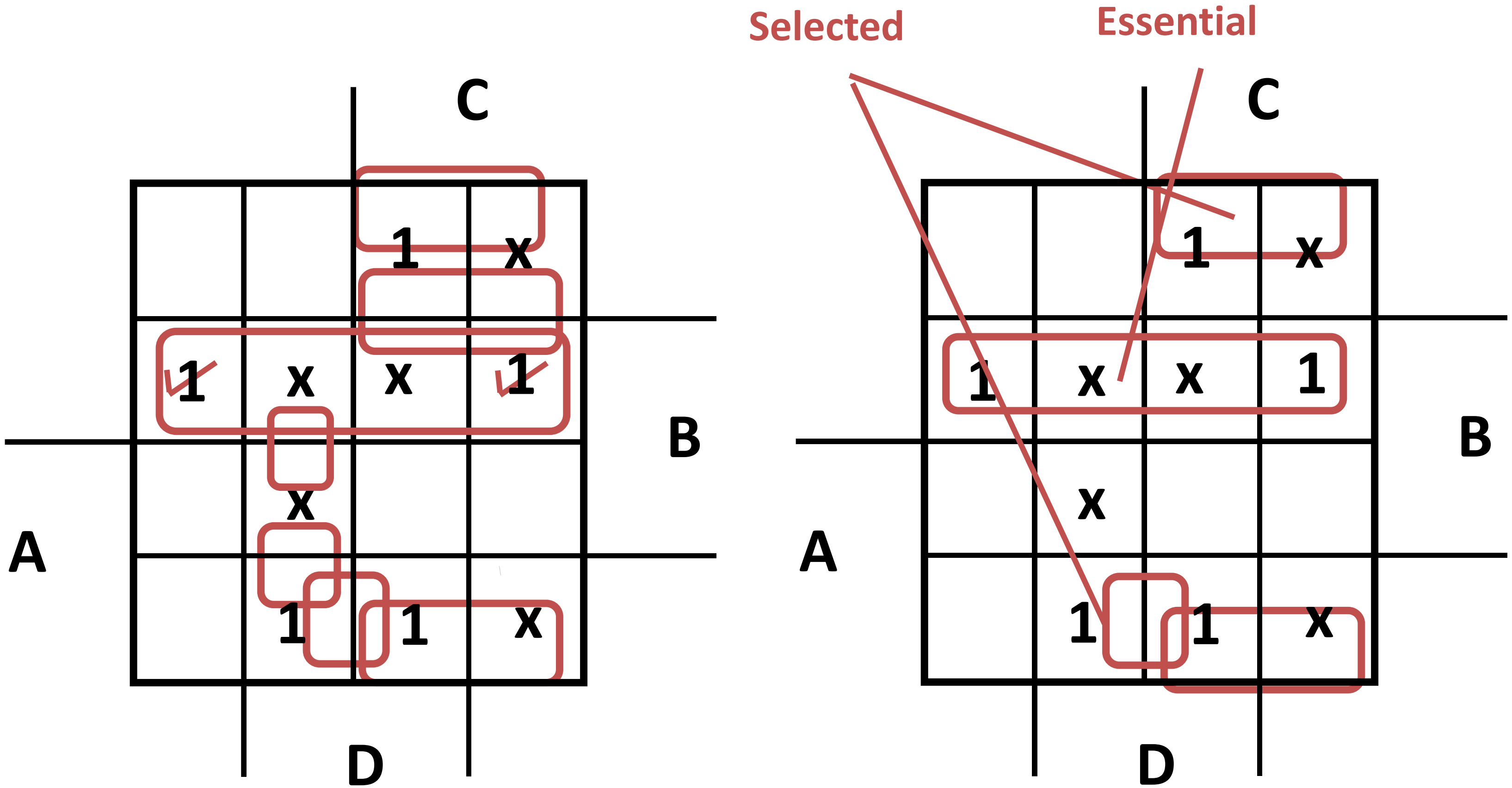
Selection Rule Example

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



Selection Rule Example with Don't Cares

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



Minterms covered by essential prime implicants