



University of  
Massachusetts  
Amherst

## Engin112 – Lectures 11 -13

### Karnaugh Maps Logic Minimization

Maciej Ciesielski  
Department of Electrical and Computer Engineering  
09/30 - 10/05/2011

## Recap from Chapter 2

- Canonical forms of Boolean functions
  - Sum of minterms, product of maxterms
- Non-canonical form
  - Sum of products
- DeMorgan's Theorem
- Boolean functions and gates
  - 16 binary Boolean functions with 2 variable
  
- Today's lecture – start Chapter 3
  - Minimization of Boolean functions
  - Karnaugh maps
  - Incompletely specified Boolean functions

# Minimization of Logic Functions

- We have chips with millions of gates
  - Why care about minimizing a function?
  - What do a few gates matter?
- Basic logic functions replicated thousands of times
  - Saving one gate for a memory cell pays off
- What is the criterion for “minimization”
  - Should we minimize
    - » Number of product terms?
    - » Number of logic operations?
    - » Number of variables (literals)?
    - » Number of wires?
    - » ...?
- For implementation: minimize number of gates

# How to Minimize Gate Count?

- Example:
  - $F = A'BC' + AB'C' + AB'C + ABC' = \Sigma(2,4,5,6)$
- How many gates do we need for implementation?
  - If AND gates have 3 inputs and OR gates have 4 inputs?
  - If all gates are binary (2 inputs)?
- Are there any tricks we can use?
  - Combine minterms:
    - $A'BC' + ABC' = BC'$
    - $AB'C' + AB'C = AB'$
  - $F = BC' + AB'$ 
    - » How many gates does  $F$  need now?
- Simplest expression:
  - Minimum number of terms and literals per term
- We need systematic approach to minimize expression
  - Answer: Karnaugh maps (K-maps)

# Karnaugh maps

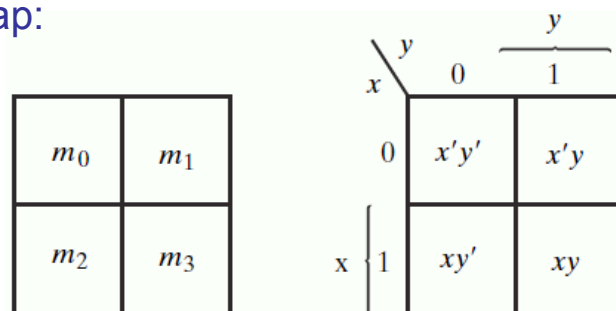
Maurice Karnaugh (1924 - )



- **Maurice Karnaugh**, an American physicist, worked in Bell Labs and IBM, invented Karnaugh maps.
- **Karnaugh map** (K-map) is a refinement of Edward Veitch's 1952 **Veitch diagram**, is a method to simplify Boolean algebra expressions.
- The Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability.

# Karnaugh Maps

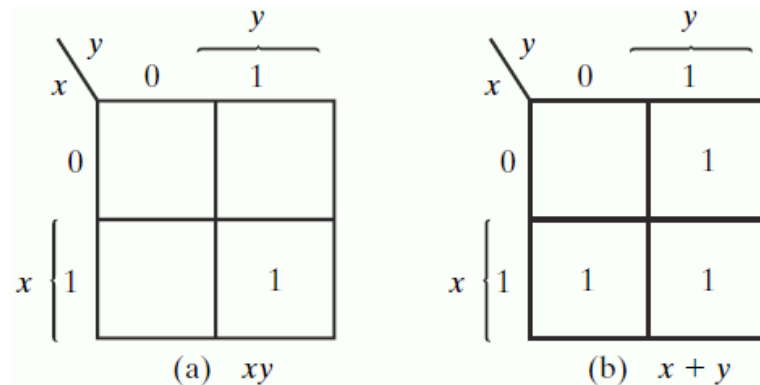
- **A Karnaugh map** is a graphical representation of a Boolean function
  - Minimization is performed by visually identifying “blocks” of logic
  - The larger the blocks, the fewer literals in the term
    - » Why ??
- **2-variable Karnaugh map:**



- Every minterm of function is present in the table

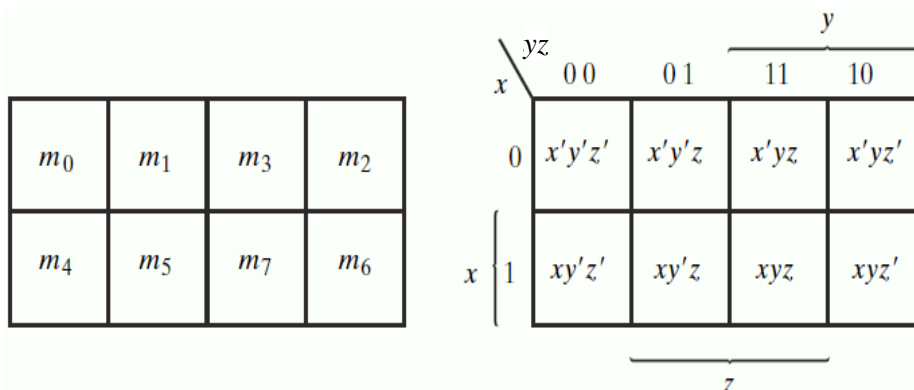
# Boolean Function in Karnaugh Map

- 1s and 0s represent function in Karnaugh map
  - 1 represent *On-set* ( $F=1$ ), 0 represents *Off-set* ( $F=0$ )
  - Similar to truth table
  - 0s are typically not shown



# 3-variable Karnaugh Map

- Karnaugh map with 3 variables:
  - Two variables on one side, one on the other
  - Note Gray code sequence (single variable change) facilitates grouping of 1-entries into logic blocks



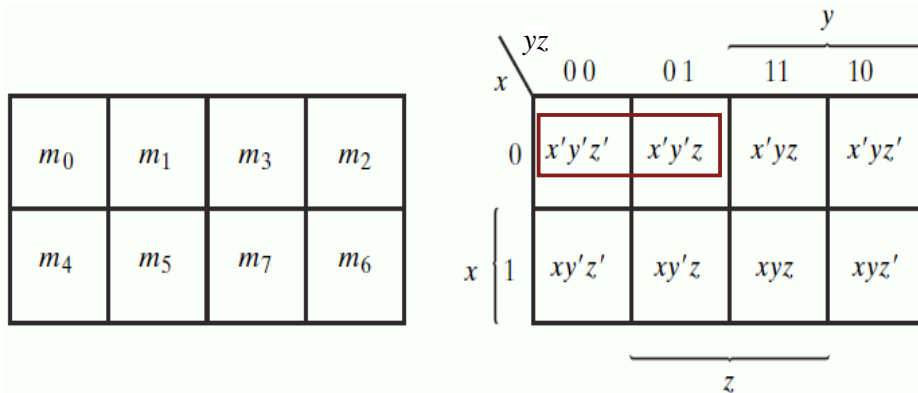
# Blocks in Karnaugh Maps

- Identifying blocks in Karnaugh maps

- Neighboring minterms can be combined:

$$x'y'z' + x'y'z = x'y'(z'+z) = x'y'$$

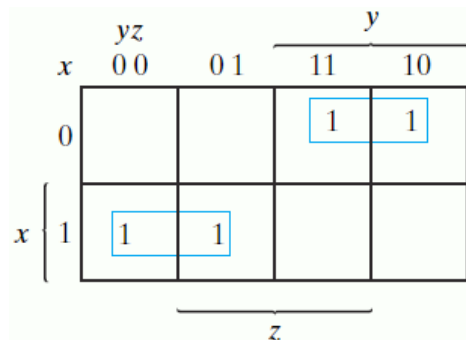
- Resulting expression uses fewer literals (z no longer present)



# Blocks in Karnaugh Maps

- Example:

- $F(x,y,z) = \Sigma(2,3,4,5) = x'yz' + x'yz + xy'z' + xy'z$



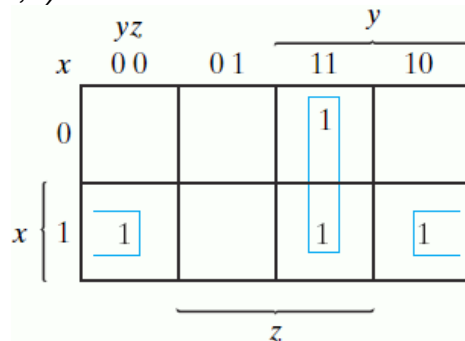
- Two blocks of size 2:

$$F = x'y + xy'$$

# Blocks in Karnaugh Maps

- What is the Karnaugh map for

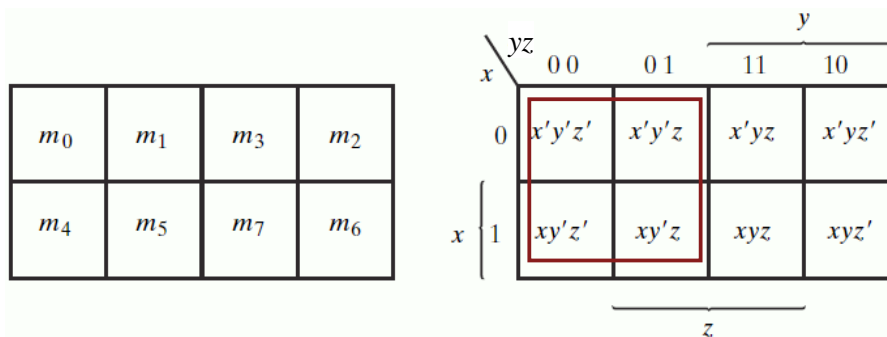
$$F(x,y,z) = \Sigma(3,4,6,7) ?$$



- Block can continue across “borders”, wrap around
  - Left to right
  - Top to bottom

# Blocks in Karnaugh Maps

- Can we combine more than two minterms?



- Yes:  $x'y'z' + x'y'z + xy'z' + xy'z = (x' + x)y'(z' + z) = y'$
- Any block that is “power of 2 size” can be reduced
  - Needs to be filled entirely with 1s
- Largest possible block yields simplest expression

# Overlapping Blocks

- Example:

$$F(A,B,C) = \Sigma(1,2,3,5,7)$$

		BC		B	
		00	01	11	10
A	0		1	1	1
	1		1	1	

C

- Blocks can overlap
  - Still find the largest possible power-2 blocks

# Converting Blocks into Expressions

- How to convert blocks into algebraic expressions?

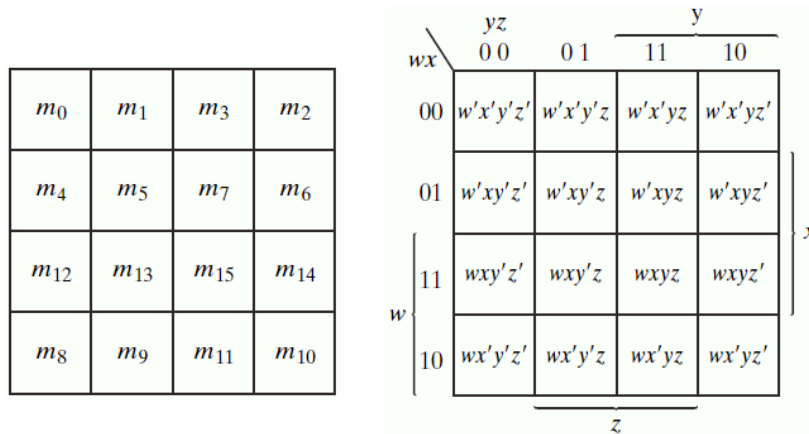
		BC		B	
		00	01	11	10
A	0		1	1	1
	1		1	1	

C

- Write down the variables that do not change
  - Example:  $F(A,B,C) = C + A'B$

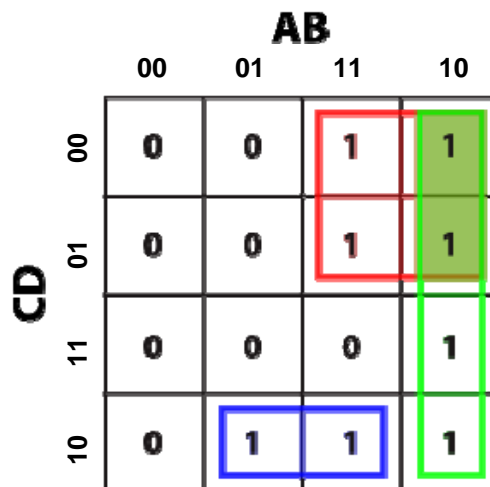
# 4-variable Karnaugh Map

- Karnaugh map can be extended to 4 variables:



# 4-variable Karnaugh Map

- K-map for 4 variable function:  $F = AC' + AB' + BCD'$



# 4-variable Karnaugh Map

- Example:  $F(w,x,y,z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$
$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$m_8$	$m_9$	$m_{11}$	$m_{10}$

		$y$			
	$yz$	00	01	11	10
$w$	$wx$				
	00				
	01				
	11				
10					
		$z$			

- Look for blocks wrapping around

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

# Choice of Blocks

- We can simplify function by using larger blocks
  - Do we really need all blocks?
  - Can we leave some out to further simplify expression?
- Function needs to contain special type of blocks
  - They are called Essential Prime Implicants
- Need to define new terms
  - Implicant
  - Prime implicant
  - Essential prime implicant

# Terminology

- **Implicant**
  - Any product term in the SOP form
  - A block of 1's in a K-map
- **Prime implicant**
  - Product term that cannot be further reduced
  - Block of 1's that cannot be further increased
- **Essential prime implicant**
  - Prime implicant that covers a 1 (minterm) that is not covered by any other prime implicant
- **Quine's Theorem** ([Willard Quine, 1908 – 2000, Harvard Univ.](#)):
  - Boolean function can be implemented with only prime implicants (but other solutions exist)
  - The number of such implicants is minimum

09/30 - 10/05/2011

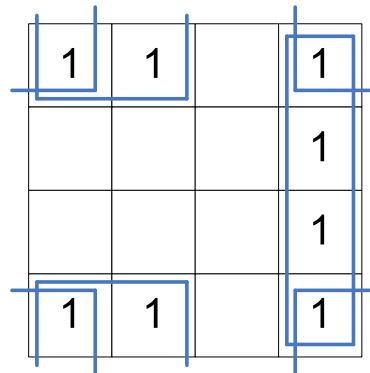
Engin 112 - Intro to ECE

19

# Essential Prime Implicants

- **Example:**  $F = \Sigma(0, 1, 2, 6, 8, 9, 10, 14)$ 
  - Show all prime implicants (how many?)
  - Which ones are essential and which not ?

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$
$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$m_8$	$m_9$	$m_{11}$	$m_{10}$



- **Essential implicants :**
  - *Implicant 1* =  $\Sigma(0, 1, 8, 9)$
  - *Implicant 2* =  $\Sigma(2, 6, 14, 10)$
- **Non-essential implicant:**  $\Sigma(0, 2, 8, 10)$

09/30 - 10/05/2011

Engin 112 - Intro to ECE

20

# Incompletely Specified Functions

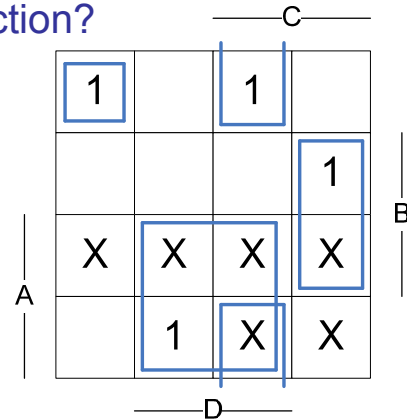
- Design a logic function that determines if a BCD digit is divisible by 3
  - Find minimum SOP logic implementation

- What are possible inputs for this function?

- Only  $(0000)_2 \rightarrow (1001)_2$  are used
- Other inputs are not valid

- What should we do with invalid inputs?

- “Don’t care”
- Output can take any value



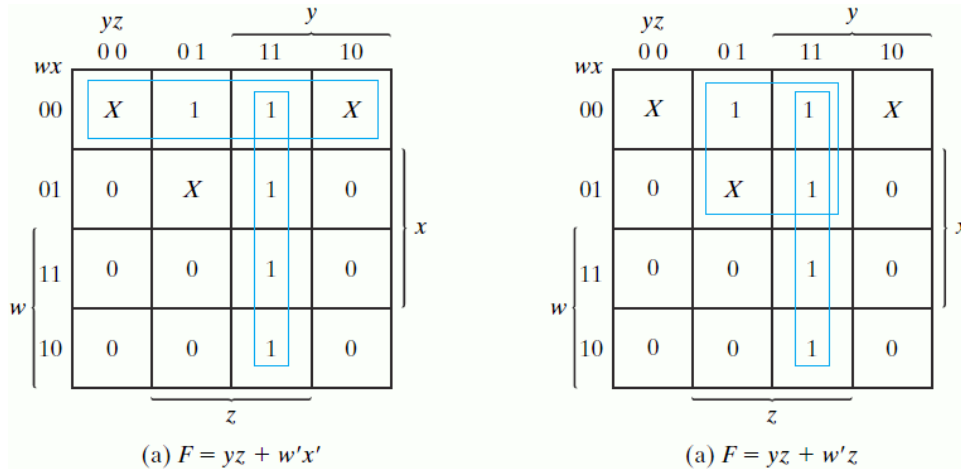
- $F(A,B,C,D) = AD + B'CD + BCD' + A'B'C'D'$

# Don't Cares: Incompletely Specified Functions

- Output of function irrelevant for some inputs
  - Input might never occur
  - Input is invalid
- Function is incompletely specified
  - Multiple completely specified functions can implement it
- Karnaugh map is marked with 'x' for don't cares
  - Output can be set to 1 or 0 (hence “don’t care”)
  - Choose most convenient output
  - Maximize block size
- Specification of don't care conditions:
  - Function:  $F(w,x,y,z) = \Sigma(1,3,7,11,15)$
  - Don't cares:  $d(w,x,y,z) = \Sigma(0,2,5)$

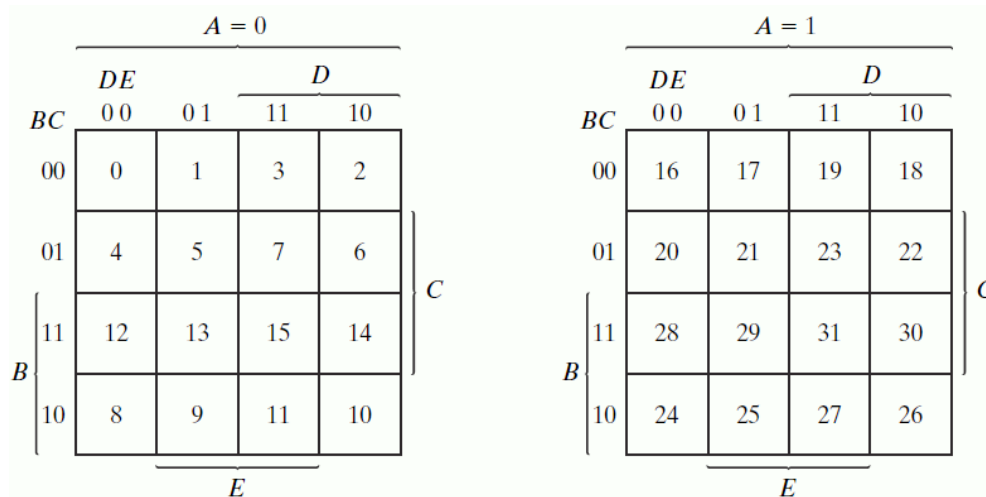
## Example with Don't Cares

- Minimization example:
  - $F(w,x,y,z) = \Sigma(1,3,7,11,15)$  and  $d(w,x,y,z) = \Sigma(0,2,5)$
- What are possible solutions?



## 5-variable Karnaugh Map

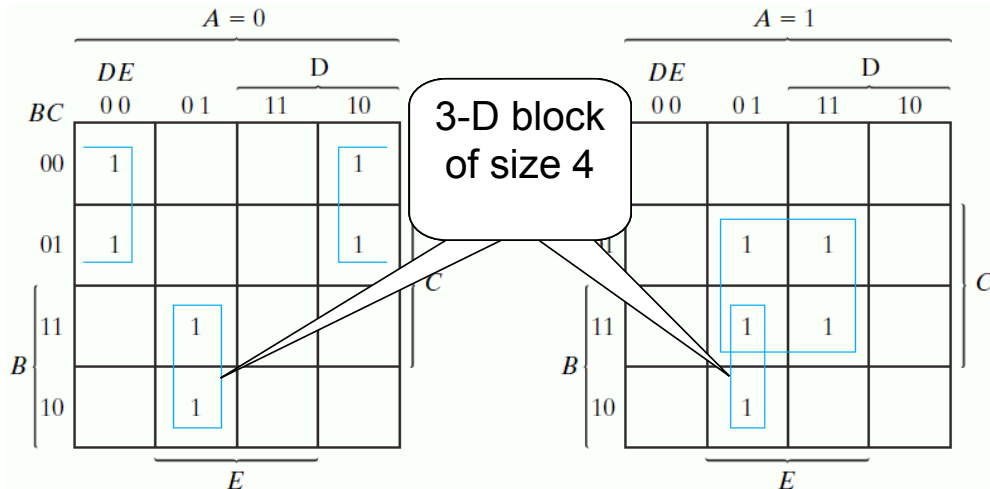
- Becomes a bit difficult to understand:



# 5-variable Karnaugh map

▪ Example:

$$F(A,B,C,D,E) = \Sigma(0,2,4,6,9,13,21,23,25,29,31)$$



$$F = A'B'E' + BD'E + ACE$$

# Product of Sums Minimization

▪ How to generate a product of sums from a Karnaugh map?

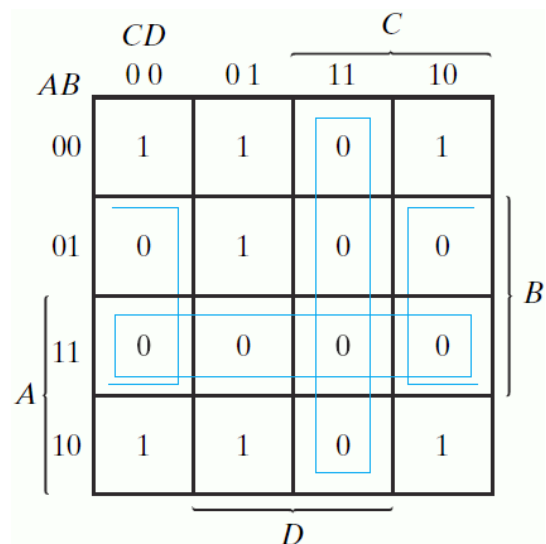
- Use duality of Boolean algebra (DeMorgan law)

▪ Look at 0s in map instead of 1s

- Generate blocks of 0's
- Gives inverse of function
- Use duality to generate product of sums

▪ Example:

- $F = \Sigma(0,1,2,5,8,9,10)$
- $F' = AB + CD + BD'$
- $F = (A'+B')(C'+D')(B'+D)$



# Midterm Exam

- Midterm Exam 1:
  - Tu, Oct. 18, 7:00 – 9:00 pm in Bartlett 65
  - Material: Chapters 1, 2, 3.