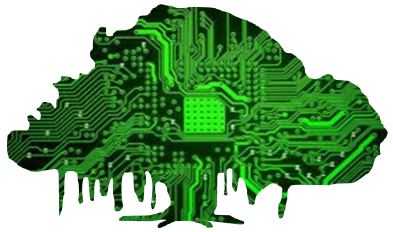


Lecture 6 – K-maps

Dr. Aftab M. Hussain,

Assistant Professor, PATRIoT Lab, CVEST

Chapter 3

Quiz 1

1. Which of the following is equivalent to the Boolean expression A+ AB + ABC + ABCD + ABCDE + ABCDEF?
   a. ABCDEF b. AB c. A+AD d. A + B + C + D + E + F

2. What is the most simplified form of this Boolean equation: $\overline{\overline{(\bar{A}BC\bar{D} + BCD)}(A \cdot \bar{B})}$
   a. $A + \bar{B}$ b. $\bar{A} + B$ c. $B \bullet C$ d. $B \bullet C \bullet D$

3. Which of the following equations is an example of the Associative Property?
   a. A•(B+C)=AB+AC b. A•1=A c. A+(B+C)=(A+B)+C d. A+B=B+A

4. Minterms are also called
   a. standard sum b. standard product c. standard deviation d. standard subtraction

5. In binary Boolean algebra, multiplicative inverse of a is:
   a. 0 b. 1 c. a' d. Not defined

6. In Boolean algebra 0 is a:
   a. commutative property b. additive identity
   c. associative identity d. multiplicative inverse

Quiz 1

7. Complement of a function expressed as a sum of minterms will be:
   a. sum of minterms
   b. product of minterms
   c. sum of maxterms
   d. product of maxterms

8. Which of these will always be a perfect square in any base b (> 10)?
   a. $(16)_b$
   b. $(64)_b$
   c. $(121)_b$
   d. $(1000)_b$

9. What is the hexadecimal equivalent of the unsigned binary number: $(10011101100011010101)_2$
   a. $(13B1D5)_{16}$
   b. $(9D8EA1)_{16}$
   c. $(9E7DA1)_{16}$
   d. $(9D8FA1)_{16}$

10. What is the hexadecimal equivalent of the octal number: $(763276327632)_8$
    a. $(B23B23B23)_{16}$
    b. $(68987BAE1)_{16}$
    c. $(F9AF9AF9A)_{16}$
    d. $(ED315134B)_{16}$

11. Solve: $(2D3B)_{16} - (1FED)_{16}$:
    a. $(13E)_{16}$
    b. $(D4E)_{16}$
    c. $(1372)_{16}$
    d. $(144E)_{16}$

12. What is the decimal value of the recurring binary number: $(0.111111\ldots..)_2$?
    a. $(1)_{10}$
    b. $(0.5)_{10}$
    c. $(0.5555\ldots.)_{10}$
    d. $(0.99)_{10}$

Quiz 1

13. In 4-bit signed 2's complement notation, the number $(100)_2$ is interpreted in decimal as:
    a. -4 b. +4 c. -8 d. 0

14. In signed magnitude BCD, the number $(100110011001)_2$ is interpreted in decimal as:
    a. +999 b. +99 c. -99 d. -999

15. For two input binary functions, what function does $xy + x'y'$ represent:
    a. XOR b. Implication c. Inhibition d. Equivalence

16. Which of the following signed representations has one representation for zero:
    a. 2's complement b. 1's complement c. signed-magnitude d. Both b. & c.

17. The AND of the two implication functions, (x implies y) AND (y implies x), gives:
    a. Always 1 b. XOR of x, y c. XNOR of x, y d. Always 0

18. Which of the following functions does not follow the commutative law:
    a. AND b. NAND c. Implication d. XOR

20. Which of the following gates can be used to make all other logic functions:
    a. NAND b. AND c. OR d. NOT

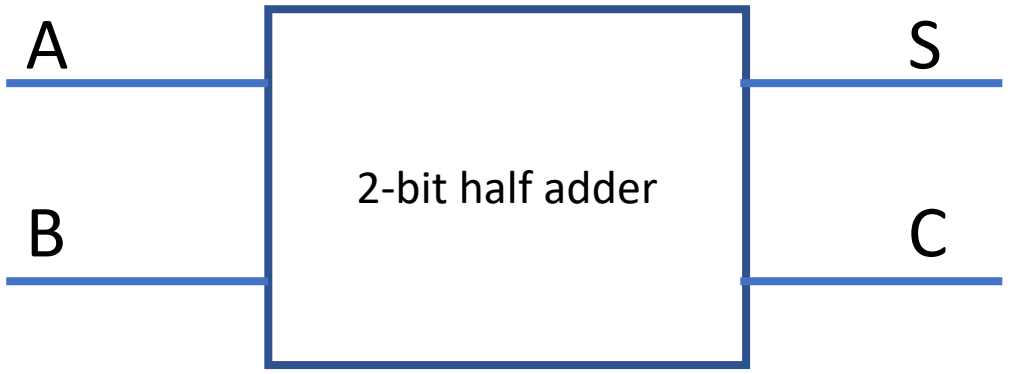
Course grade assignment

• Assignments	10%
• Lab Reports	10%
• Quizzes	10%
• Lab Exam	15%
• Midsem	20%
• Endsem	35%

Logic circuits

- Logic circuits are combinations of logic gates to make a particular logic function of our choice
- The logic function can be uniquely represented by a truth-table
- However, many algebraic expressions give the same truth-table and the best or most optimum way of making the circuit is to be found by designers
- Consider a simple 2-bit adder: A + B
- We go from having a logic function to getting the truth-table, then obtaining the circuit
- In this case, because of its simplicity, we know that

$$S = A \oplus \mathrm{B} \text{ and } C = A \cdot \mathrm{B}$$



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic circuits

- Let us say we wanted to continue adding, now we need to take care of the previous carry
- Thus, we need to add A, B and C to get a generic adding machine
- Still from observation, we can deduce that $S = A \oplus B \oplus C$
- But what is the algebraic expression for C_1?
- We know the sum of minterms, but is there an easier way?

A	B	C_0	S	C_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Gate level minimization

- *Gate-level minimization* is the design task of finding an optimal gate-level implementation of the Boolean functions describing a digital circuit

- This task is well understood, but is difficult to execute by manual methods when the logic has more than a few inputs

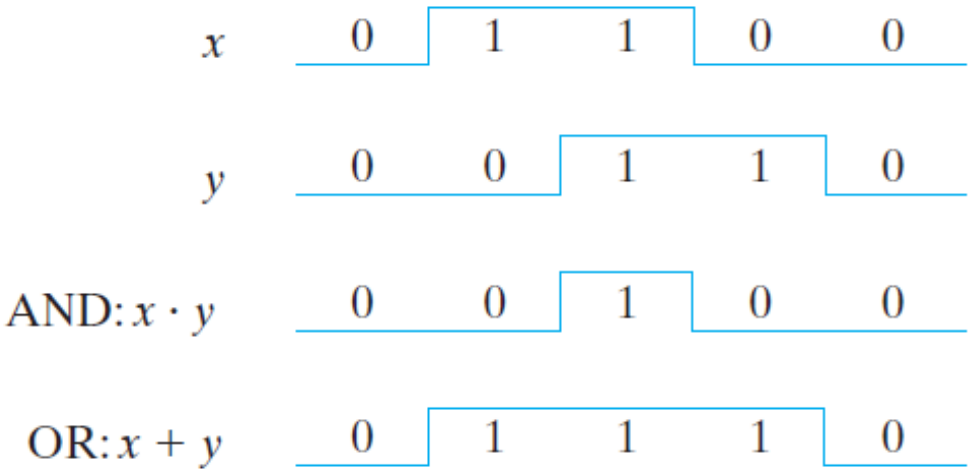
- Fortunately, computer-based logic synthesis tools can minimize a large set of Boolean equations efficiently and quickly

- Nevertheless, it is important that a designer understand the underlying mathematical description and solution of the problem

Gate level minimization

- We have seen two methods of visualizing digital circuit output: truth-tables and timing diagrams
- Now we will use another method to visualize the truth-table, called the map method or *Karnaugh maps*
- The map method provides a simple, straightforward procedure for minimizing Boolean functions
- This method may be regarded as a pictorial form of a truth table

AND			OR		
x	y	$x \cdot y$	x	y	$x + y$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1



Gate level minimization

- Some history: although we use the term Karnaugh map (named after Maurice Karnaugh, 1953) for the map method, it dates back further
- In 1881, Allan Marquand published a paper criticizing "Mr. Venn" for his approach to visualization of logic problems using curvilinear shapes and proposed a method with non-interesting squares
- Non-intersecting because he was concerned with two-state variables that are either *true* or *false*

XXXIII. *On Logical Diagrams for* n *terms.* *By* ALLAN MARQUAND, *Ph.D., late Fellow of the Johns Hopkins University**.

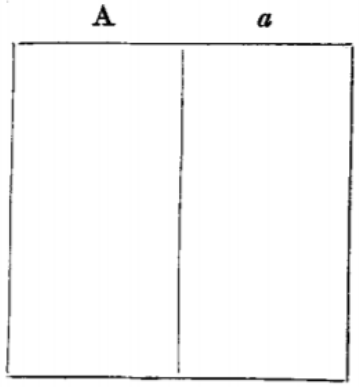
IN the Philosophical Magazine for July 1880 Mr. Venn has offered diagrams for the solution of logical problems involving three, four, and five terms. From the fact that he makes use of circles, ellipses, and other curvilinear figures, the construction of diagrams becomes more and more difficult as new terms are added. Mr. Venn stops with the five-term diagram, and suggests that for six terms "the best plan would be to take two five-term figures."

It is the object of this paper to suggest a mode of constructing logical diagrams, by which they may be indefinitely extended to any number of terms, without losing so rapidly their special function, viz. that of affording visual aid in the solution of problems.

Conceiving the logical universe as always more or less limited, it may be represented by any closed figure. For convenience we take a square. If then we drop a perpendicular from the middle point of the upper to the lower side of the square, the universe is prepared for a classification of its contents by means of a single logical term.

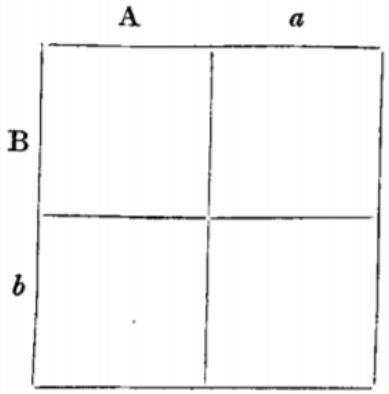
Gate level minimization

- Some history: although we use the term Karnaugh map (named after Maurice Karnaugh, 1953) for the map method, it dates back further
- In 1881, Allan Marquand published a paper criticizing “Mr. Venn” for his approach to visualization of logic problems using curvilinear shapes and proposed a method with non-interesting squares
- Non-intersecting because he was concerned with two-state variables that are either *true* or *false*



This represents a universe with its A and not-A “compartments.” The quantitative relation of the compartments being insignificant, they may for convenience be represented as equal.

The introduction of a second term divides each of the existing compartments. This may be done by a line drawn at right angles to our perpendicular and through its centre, thus :—

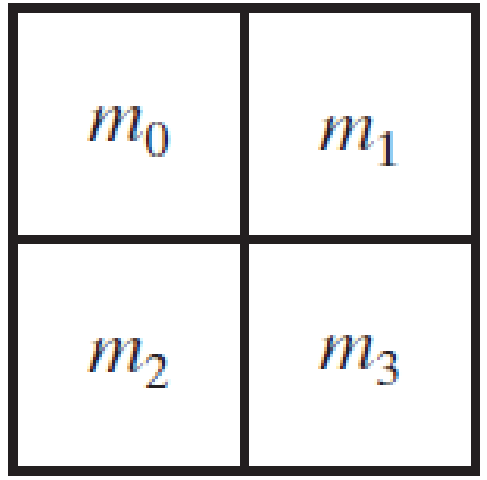


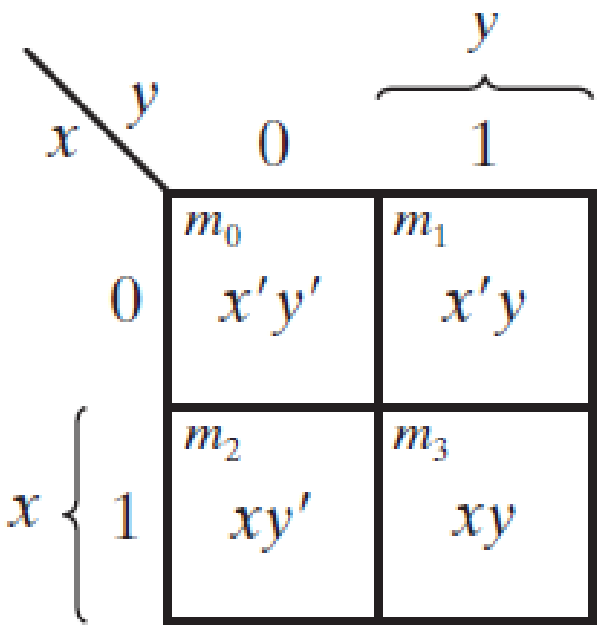
Gate level minimization

- A K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized
- Since any Boolean function can be expressed as a sum of minterms, it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function (i.e. are 1)
- In fact, the map presents a visual diagram of all possible ways a function may be expressed in standard form
- By recognizing various patterns, the user can derive alternative algebraic expressions for the same function, from which the simplest can be selected

2 variable K-map

- There are four minterms for two variables; hence, the map consists of four squares, one for each minterm
- The map can be redrawn to show the relationship between the squares and the two variables *x* and *y*
- The 0 and 1 marked in each row and column designate the values of variables
- In minterm form, variable *x* appears primed in row 0 and unprimed in row 1. Similarly, *y* appears primed in column 0 and unprimed in column 1





3 variable K-map

- In 3 variable problems, there are eight minterms for three binary variables; therefore, the map consists of eight squares
- The map is marked with numbers in each row and each column to show the relationship between the squares and the three variables
- For example, the square assigned to m_5 corresponds to row 1 and column 01
- Each cell of the map corresponds to a unique minterm, so another way of looking at the square is $m_5 = xy'z$
- Note that there are four squares in which each variable is equal to 1 and four in which each is equal to 0

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

x \ yz	00	01	11	10
0	m_0 $x'y'z'$	m_1 $x'y'z$	m_3 $x'yz$	m_2 $x'yz'$
1	m_4 $xy'z'$	m_5 $xy'z$	m_7 xyz	m_6 xyz'

y (11, 10)

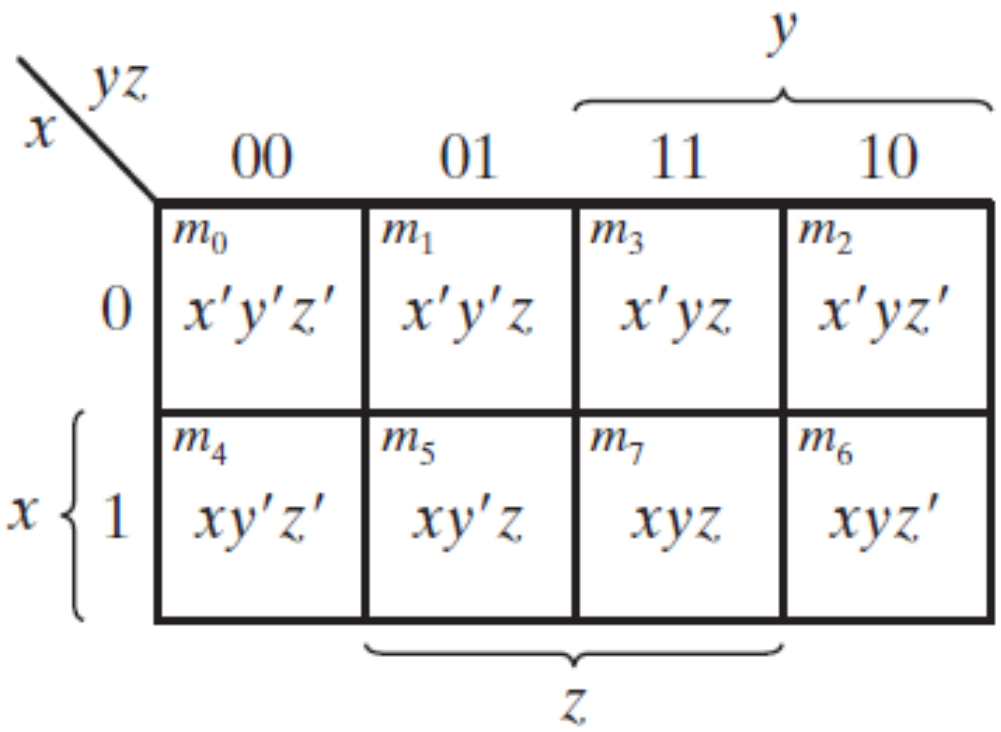
x (1)

z (01, 11)

3 variable K-map

- Here is the magic: To understand the usefulness of the map in simplifying Boolean functions, we must recognize the basic property possessed by adjacent squares: **Any two adjacent squares in the map differ by only one variable,** which is primed in one square and unprimed in the other
- For example, m_5 and m_7 lie in two adjacent squares
- Variable y is primed in m_5 and unprimed in m_7, whereas the other two variables are the same in both squares
- From the postulates of Boolean algebra, it follows that the sum of two minterms in adjacent squares can be simplified to a single product term consisting of only two literals

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



3 variable K-map

- Here is how we optimize the logic gate expression using maps:

1. We look for a cluster of adjacent squares with the function value being 1 (or true):
   1. A cluster of 8 squares (meaning the entire function is 1)
   2. A cluster of 4 squares (meaning one literal)
   3. A cluster of 2 squares (meaning two literals ANDed)
   4. A single square with all three variables ANDed (the minterm)
2. We OR all the expressions related to the clusters

- Note that “adjacent” squares mean vertically or horizontally, not diagonally

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

x \ yz	00	01	11	10
0	m_0 $x'y'z'$	m_1 $x'y'z$	m_3 $x'yz$	m_2 $x'yz'$
1	m_4 $xy'z'$	m_5 $xy'z$	m_7 xyz	m_6 xyz'

y (11, 10)

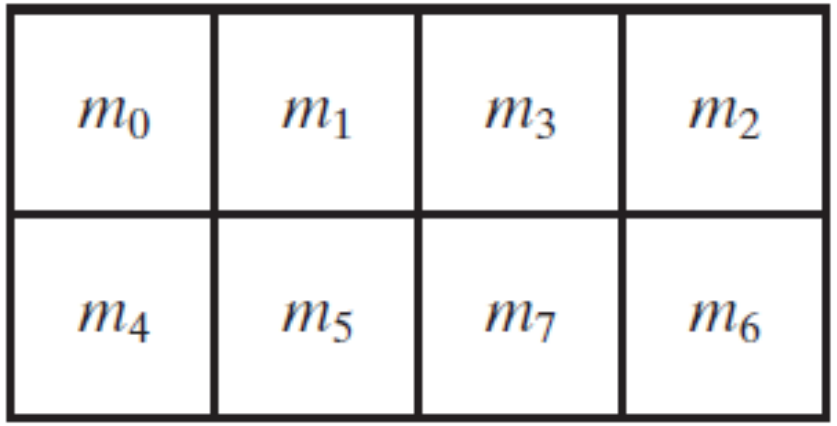
x (1)

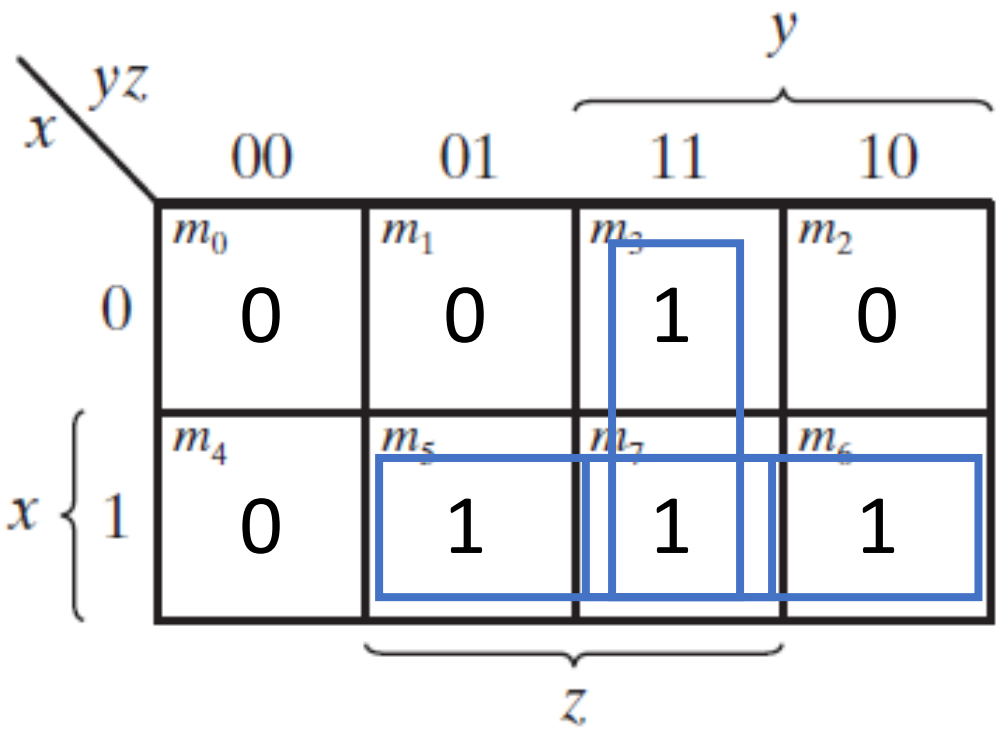
z (01, 11)

3 variable K-map

- Full adder (F):

A	B	C_0	S	C_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





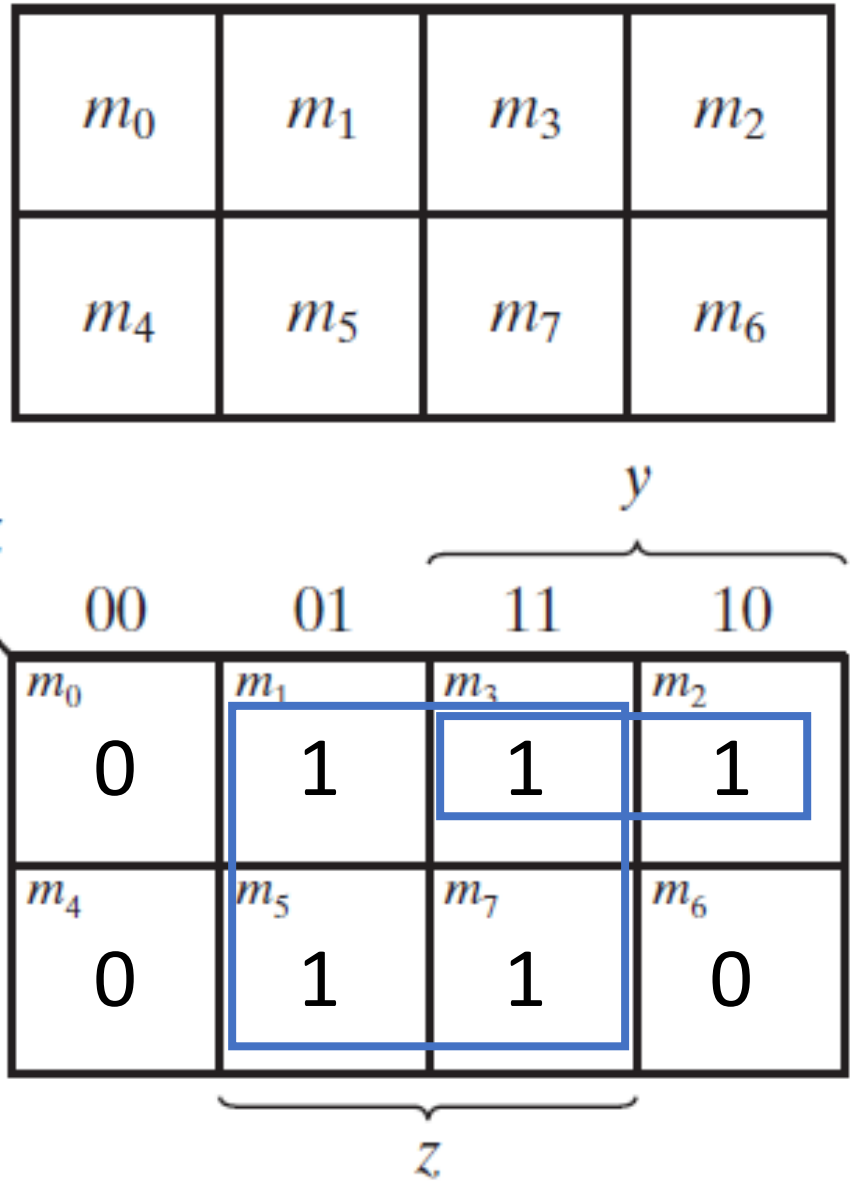
- Three clusters of 2 squares
- Thus, $F = xy + yz + zx$

3 variable K-map

- $F(x, y, z) = \sum(1,2,3,5,7)$

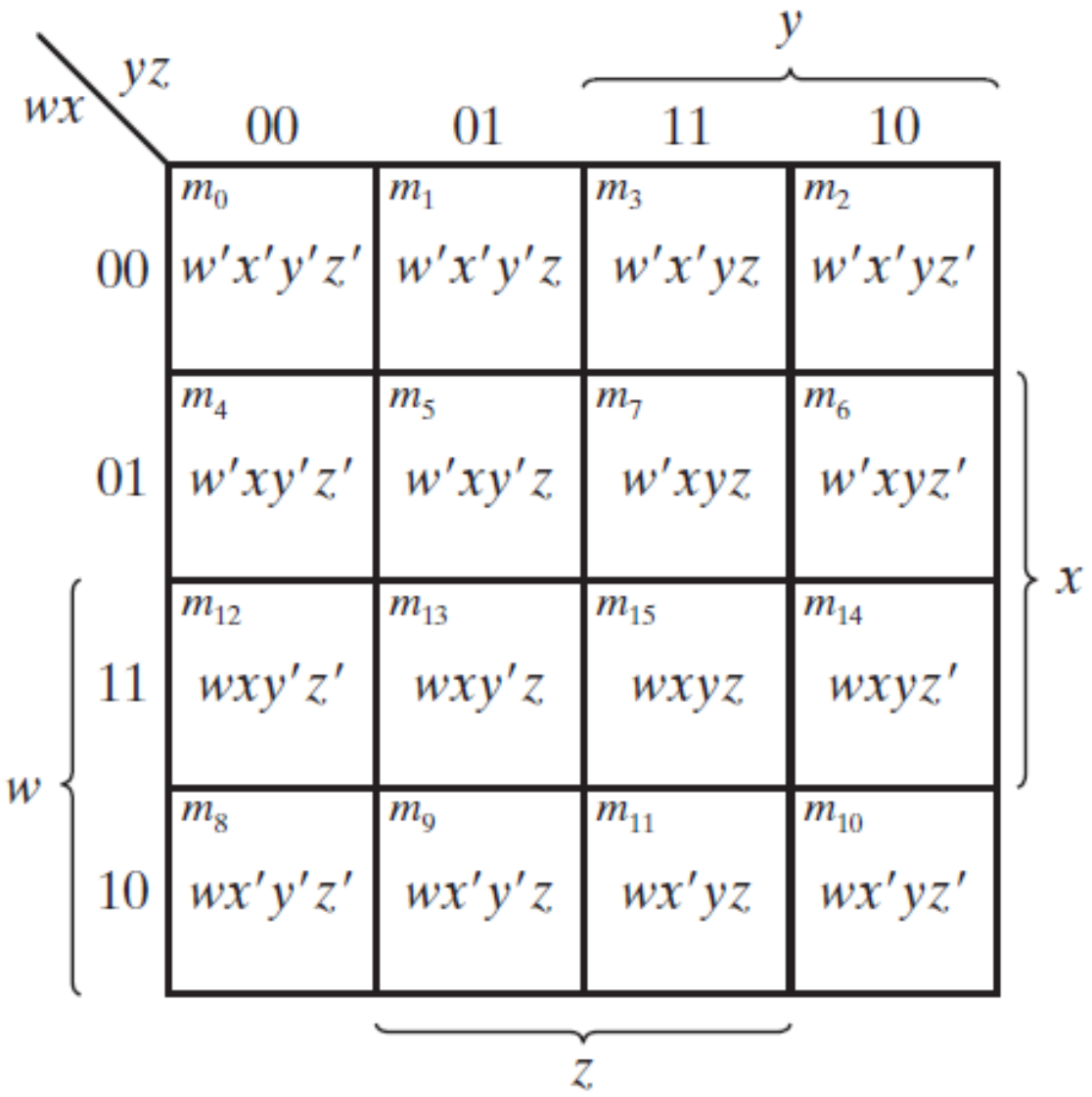
- We have one cluster of 4 squares
- This represents z
- One cluster of 2 squares
- This represents x'y

- Thus, $F = z + x'y$



4 variable K-map

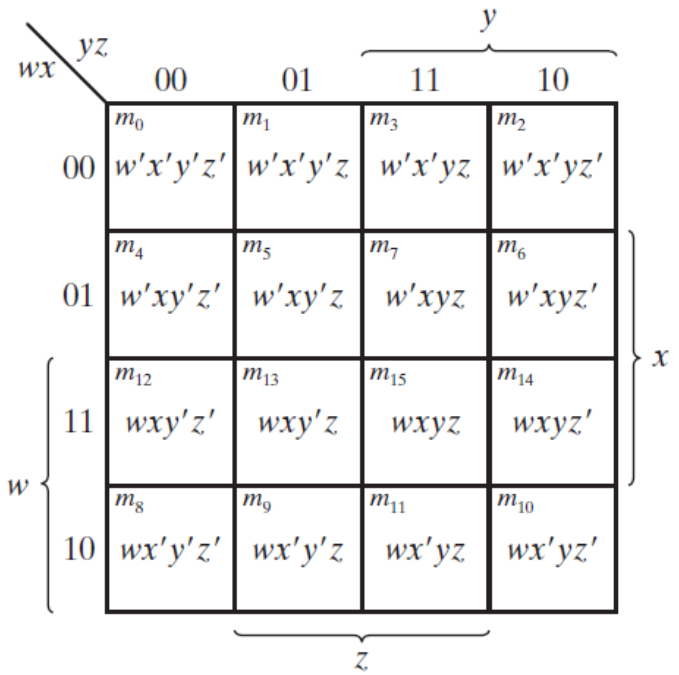
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}



4 variable K-map

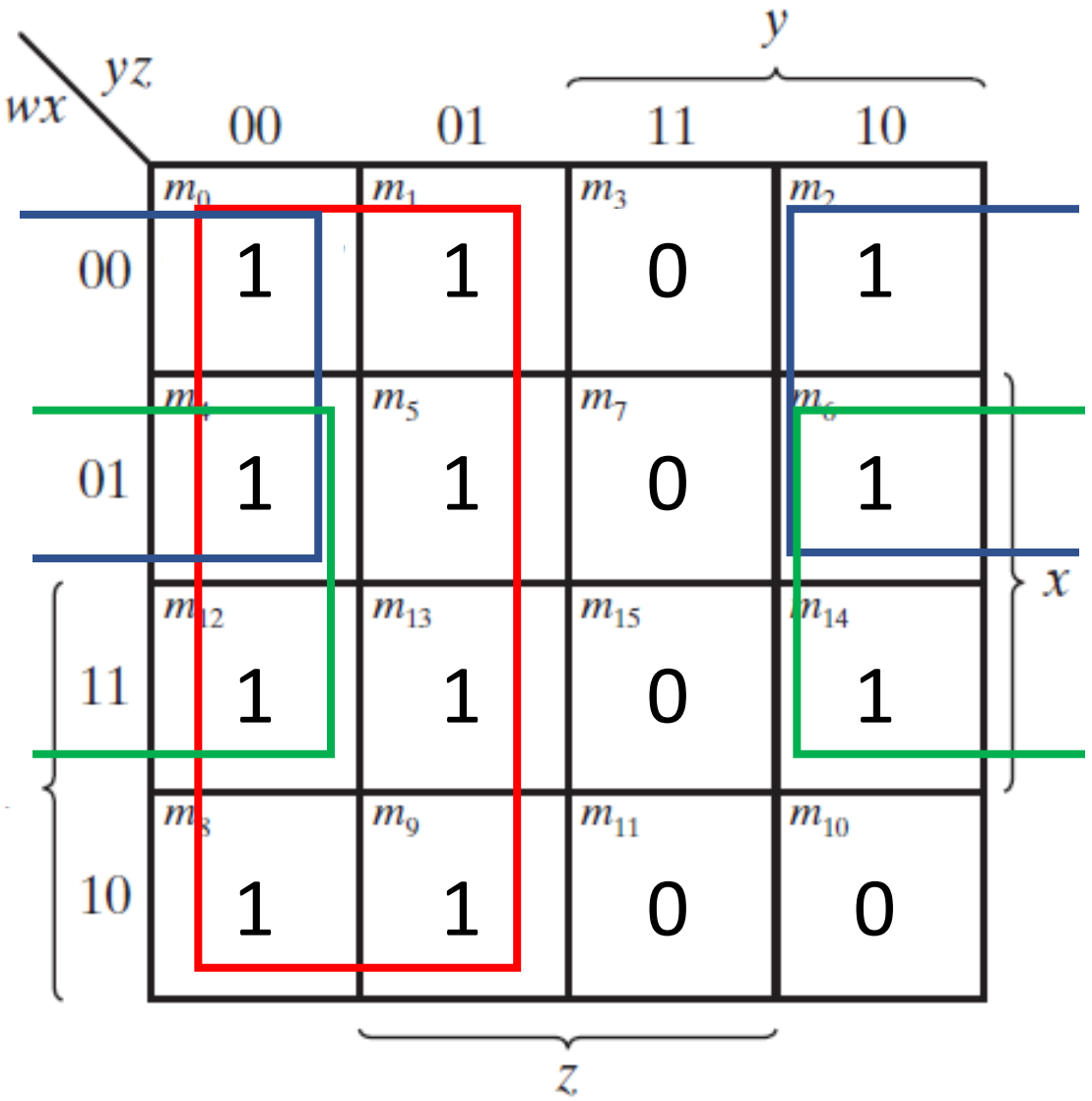
1. We look for a cluster of adjacent squares with the function value being 1 (or true):
    1. A cluster of 16 squares (meaning the entire function is 1)
    2. A cluster of 8 squares (meaning one literal)
    3. A cluster of 4 squares (meaning two literals ANDed)
    4. A cluster of 2 squares (meaning three literals ANDed)
    5. A single square with all the four variables ANDed (a minterm)
2. We OR all the expressions related to the clusters

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}



4 variable K-map

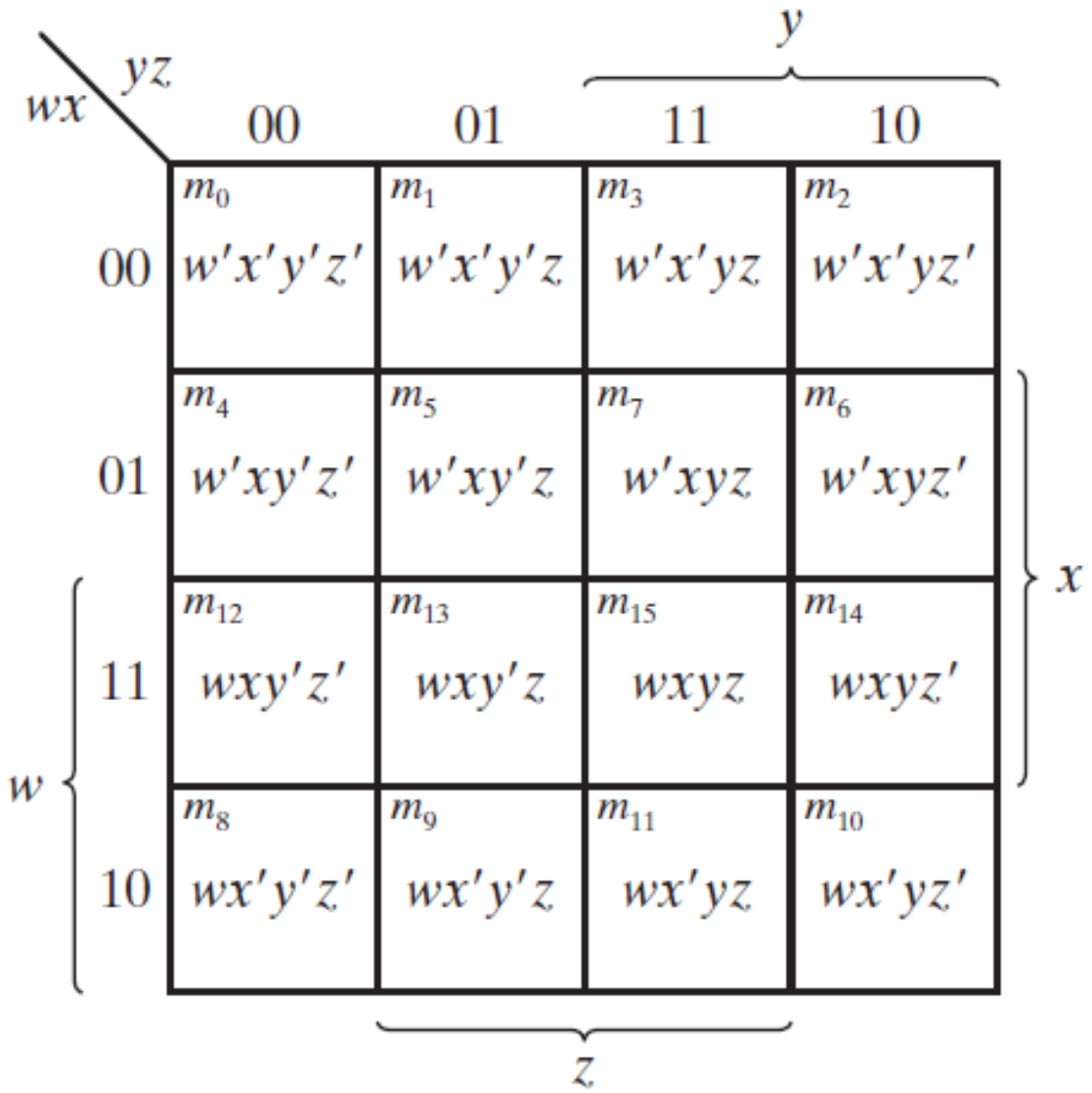
- Simplify the function
  $F\ (w, x, y, z)\ =$
  $\sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$
- Cluster of 16? No
- Cluster of 8?
- It represents y’
- Cluster of 4?
- This represents w’z’
- This represents xz’
- Thus, the function is
  $F\ (w, x, y, z) = y' + w'z' + xz'$



Always remember…

- In choosing adjacent squares in a map, we must ensure that

1. All the minterms of the function are covered when we combine the squares
2. The number of terms in the expression is minimized
3. There are no redundant terms (i.e., minterms already covered by other terms)



Prime implicants

- Sometimes there may be two or more expressions that satisfy the simplification criteria
- To make the process more standardized, we define a prime implicant.
- A *prime implicant* is a product term obtained by combining the maximum possible number of adjacent squares in the map
- If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be *essential*

$$F(x,y,z) = \sum(0,2,3,4,7)$$

$$F(x,y,z) = yz + y'z' + ?$$

$$F(x,y,z) = yz + y'z' + yx'$$

$$F(x,y,z) = yz + y'z' + z'x'$$