Lecture 10 – K-maps

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**.

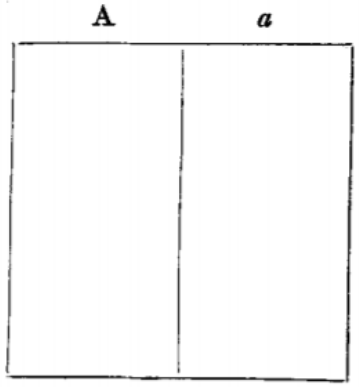
IN the Philosophical Magazine for July 1880 Mr. Venn has offered diagrams for the solution of logical problems involving three, four, and five terms. From the fact that he makes use of circles, ellipses, and other curvilinear figures, the construction of diagrams becomes more and more difficult as new 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.

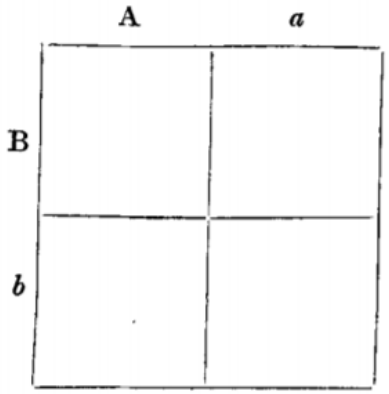
Gate level minimization

- 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*



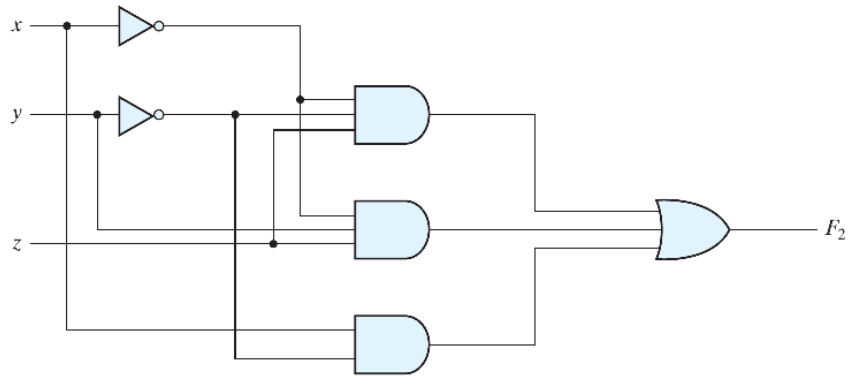
This represents a universe with its A and not-A "compartments." The quantitative relation of the compartments being insignificant, they may for convenience be represented as equal.

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 :—

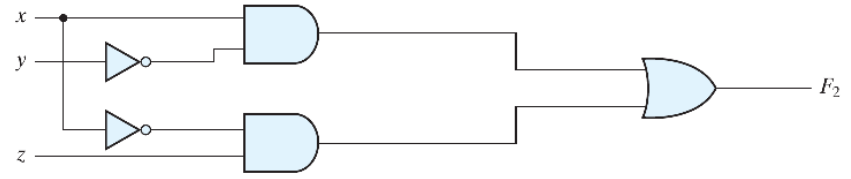


Gate level minimization

- *Gate-level minimization* is the task of finding an optimal gate-level implementation of the Boolean functions describing a digital circuit.
- The complexity of logic gate implementation of a Boolean function is directly related to the complexity of the algebraic expression from which the function is implemented.
- Algebraic simplification of Boolean expressions can be cumbersome
- Karnaugh map (K-Map) provides a systematic method to simplify Boolean expressions



(a) $F_2 = x'y'z + x'yz + xy'$

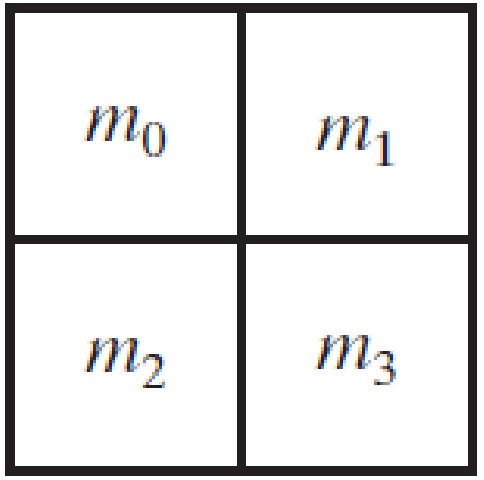


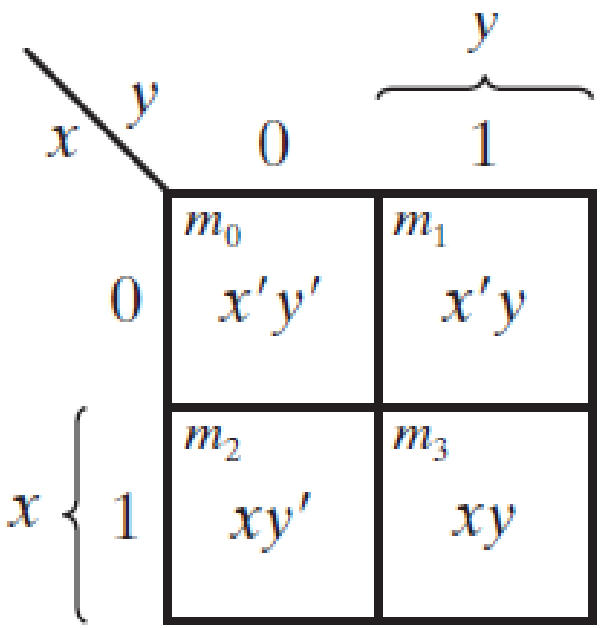
(b) $F_2 = xy' + x'z$

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$

2 variable K-map

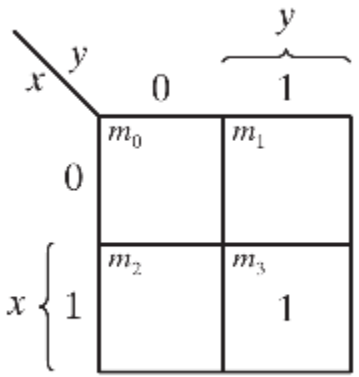
- There are four *minterms* for two variables – *xy, x'y, xy', x'y'*; hence, the map consists of four squares, one for each *minterm*
- The map can be drawn to show the relationship between the squares and the two variables *x* and *y*
- The 0 and 1 marked in each row and column designate the values 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



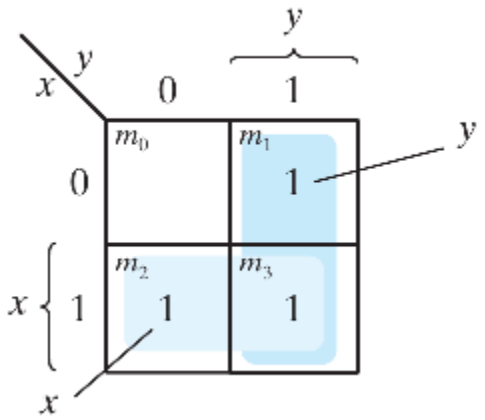


2 variable K-map

Example:



Map for F= *xy*

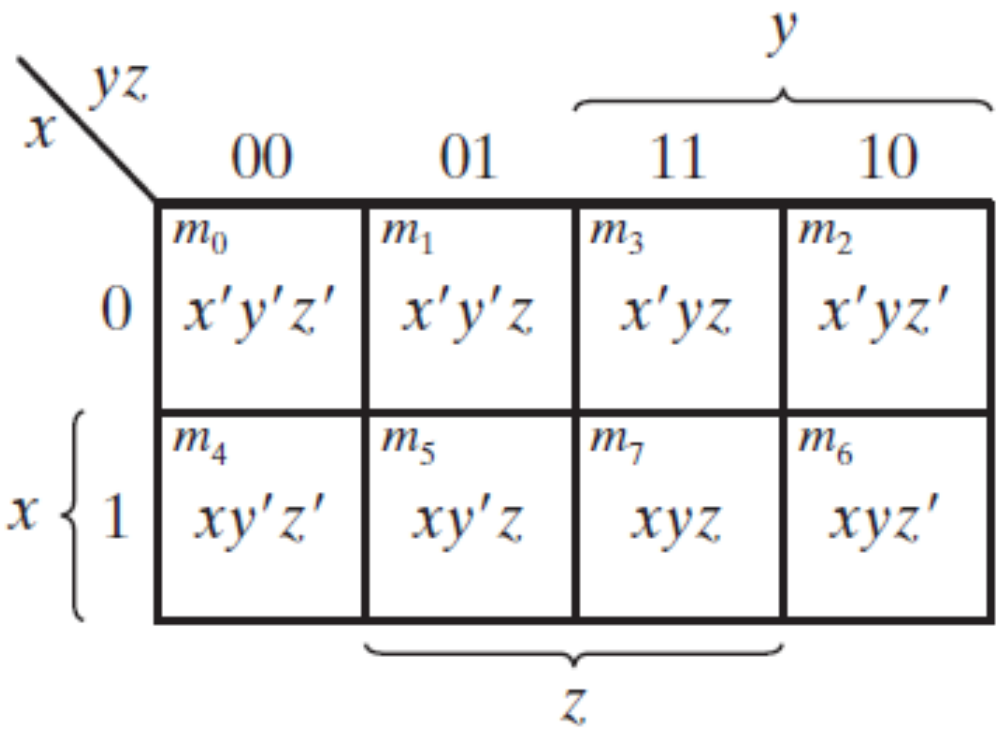


Map for F= *x*+*y*

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



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- Eg: xy'z + xyz = xz(y'+y)=xz

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) z (01, 11)

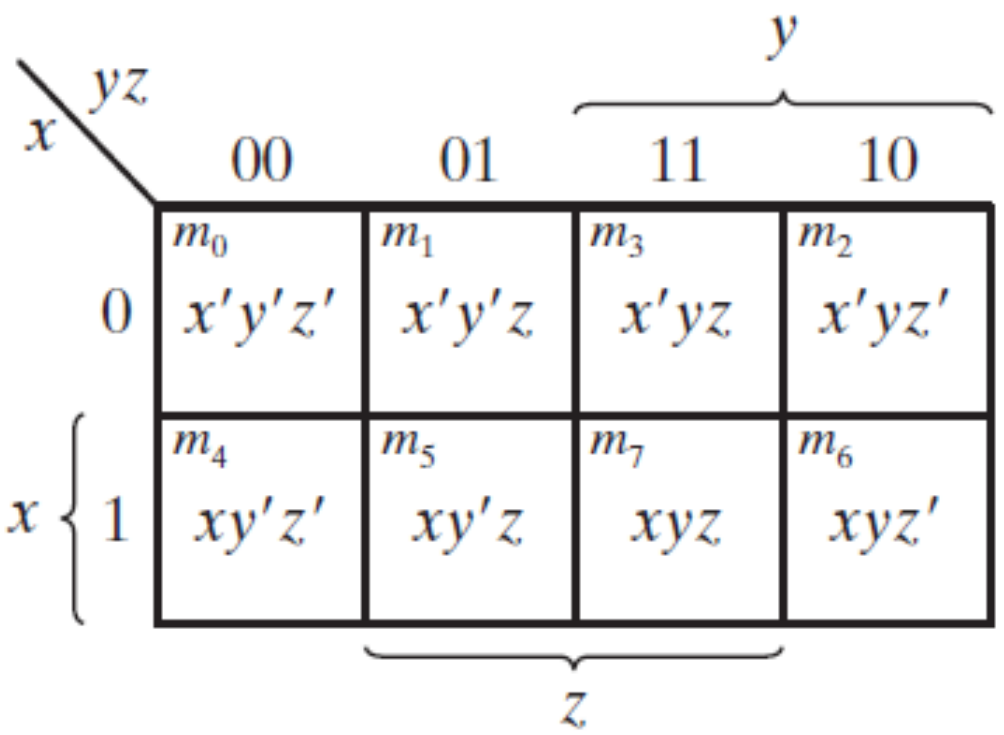
3 variable K-map

- 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. cluster of 8 squares (meaning the entire function is 1)
    2. A cluster of 4 squares (meaning one literal) – eg: xyz+xyz'+x'yz+x'yz' = xy + x'y = y
    3. A cluster of 2 squares (meaning two literals ANDed)
    4. single square with all three variables ANDed (the minterm)
2. We OR all the expressions related to the clusters

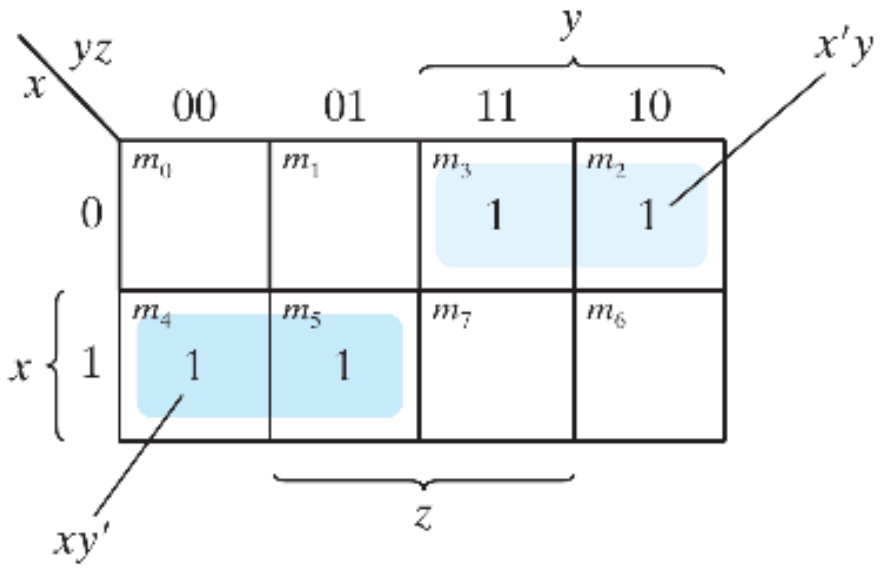
- Note that "adjacent" squares mean vertically or horizontally, not diagonally

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



3 variable K-map

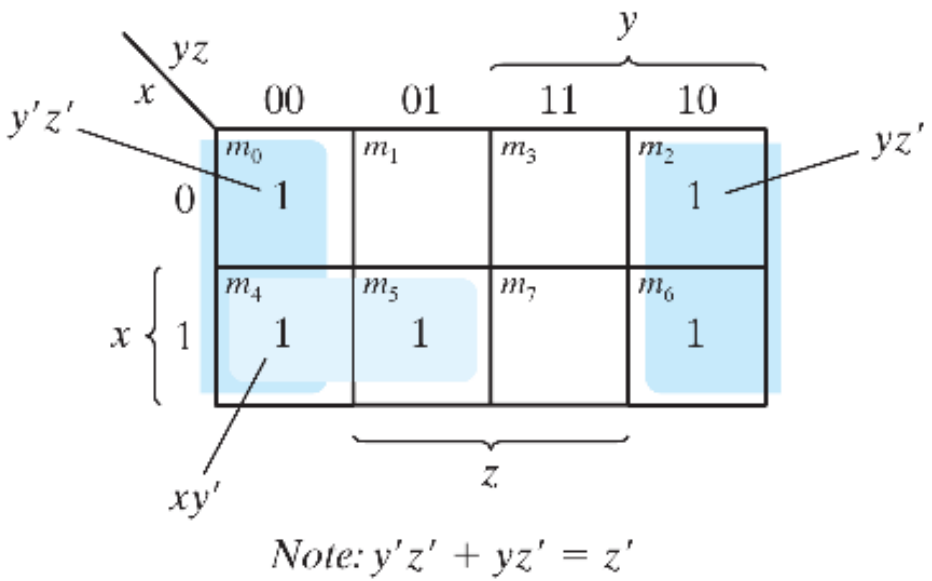
Simplify the Boolean function: $F(x, y, z) = \Sigma(2, 3, 4, 5)$



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

3 variable K-map

Simplify the Boolean function: $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$



m_2 is adjacent to m_0

m_6 is adjacent to m_4

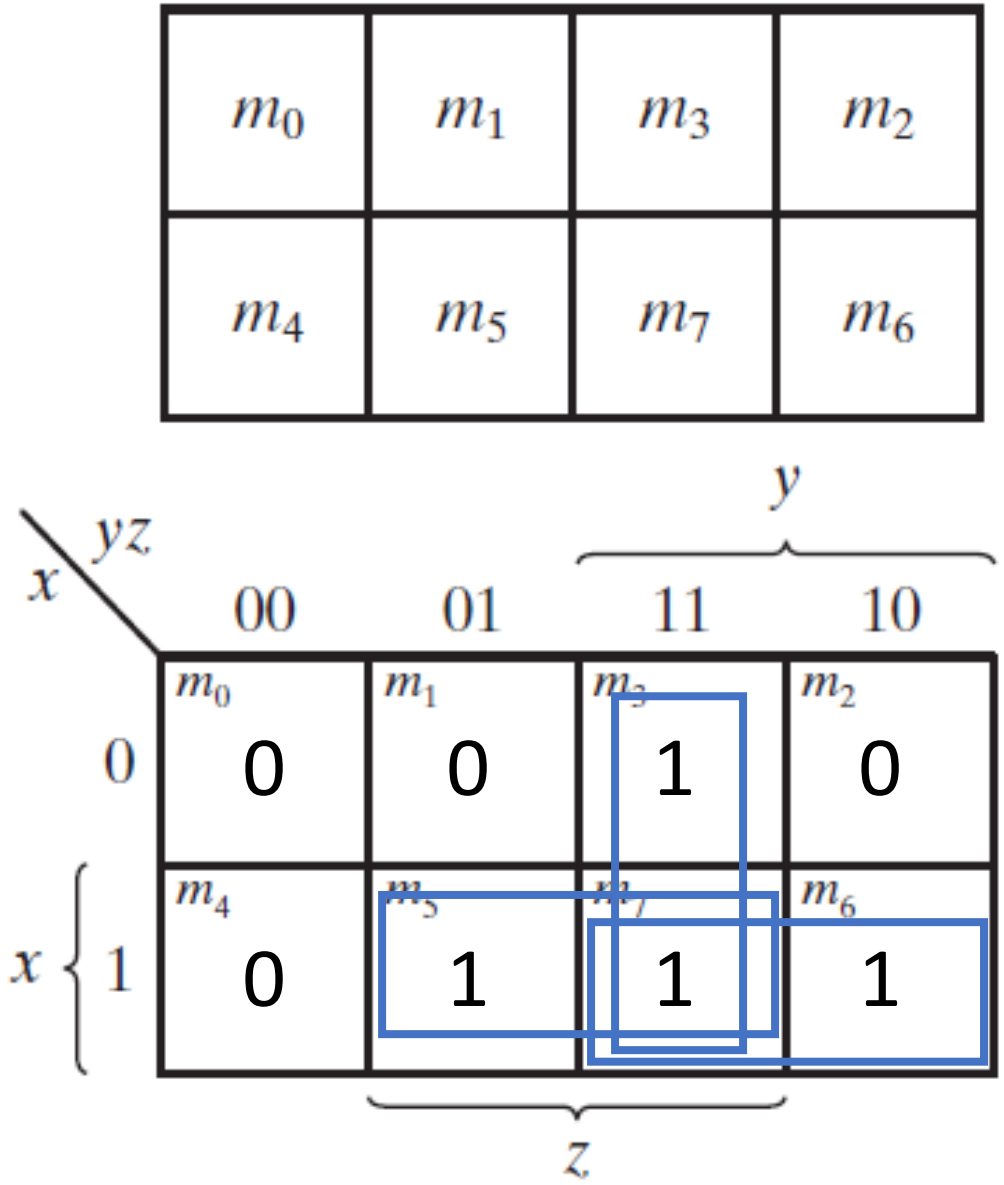
$F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

3 variable K-map

- Full adder:

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

- Let x = A , y = B, z = C_0
- Three clusters of 2 squares
- Thus, $C_1 = xy + yz + zx$
  $C_1 = AB + BC_0 + C_0A$



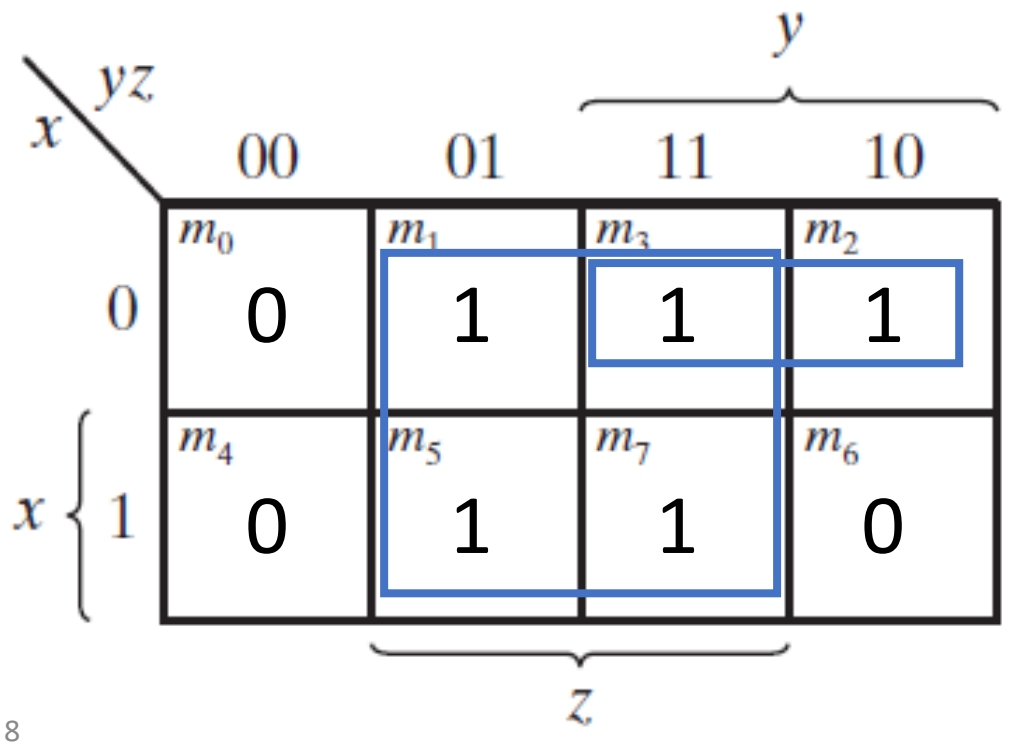
3 variable K-map

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

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

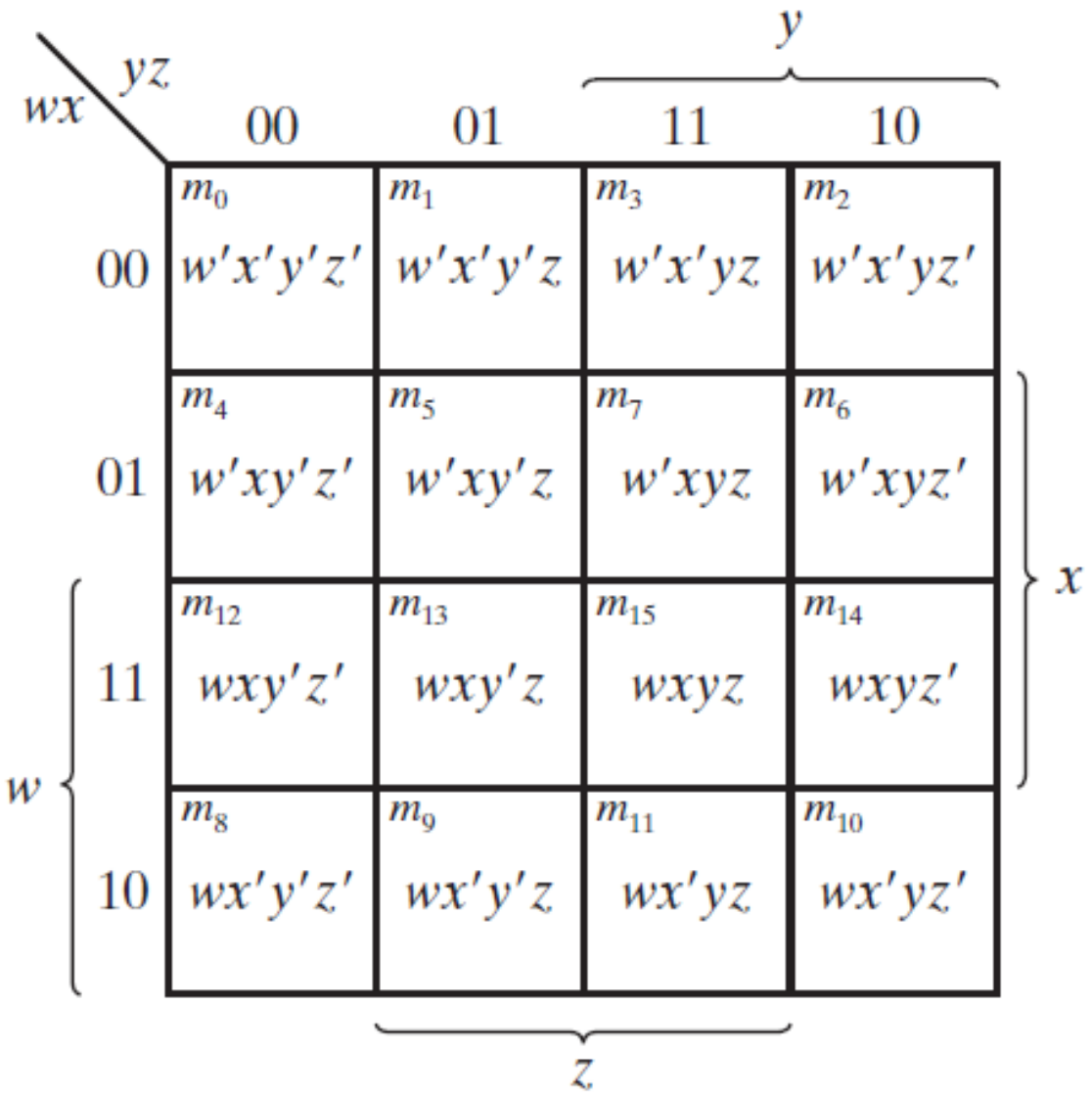
- Thus, $F = z + x'y$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



4 variable K-map

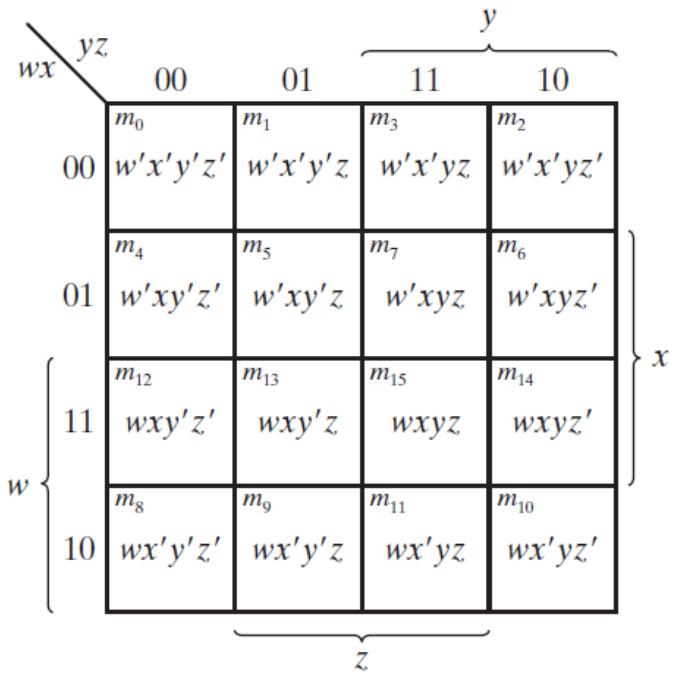
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}



4 variable K-map

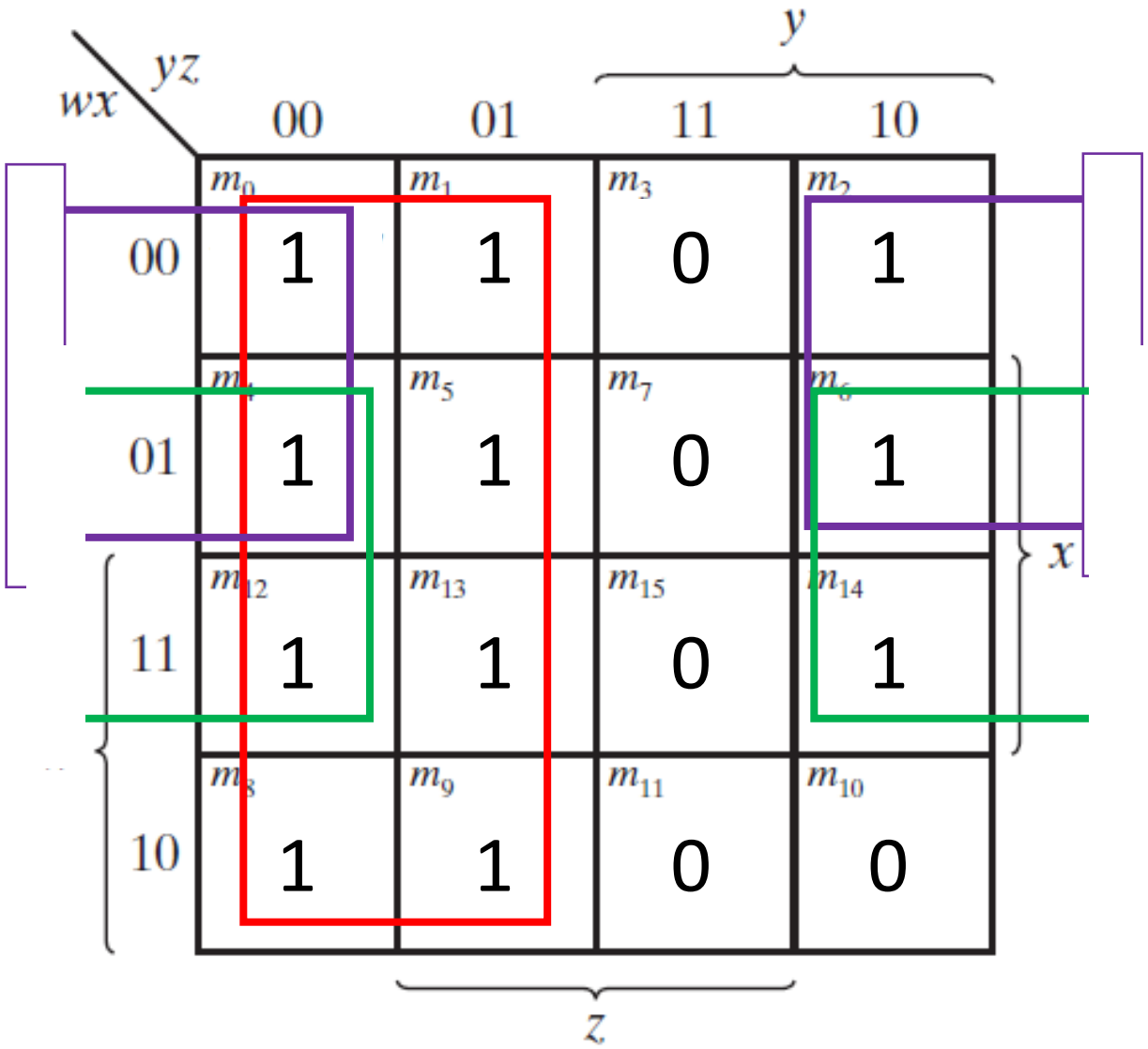
1. We look for a cluster of adjacent squares with the function value being 1 (or true):
   1. A cluster of 16 squares (meaning the entire function is 1)
   2. A cluster of 8 squares (meaning one literal)
   3. A cluster of 4 squares (meaning two literals ANDed)
   4. A cluster of 2 squares (meaning three 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
  $F\ (w, x, y, z) = y' + w'z' + xz'$



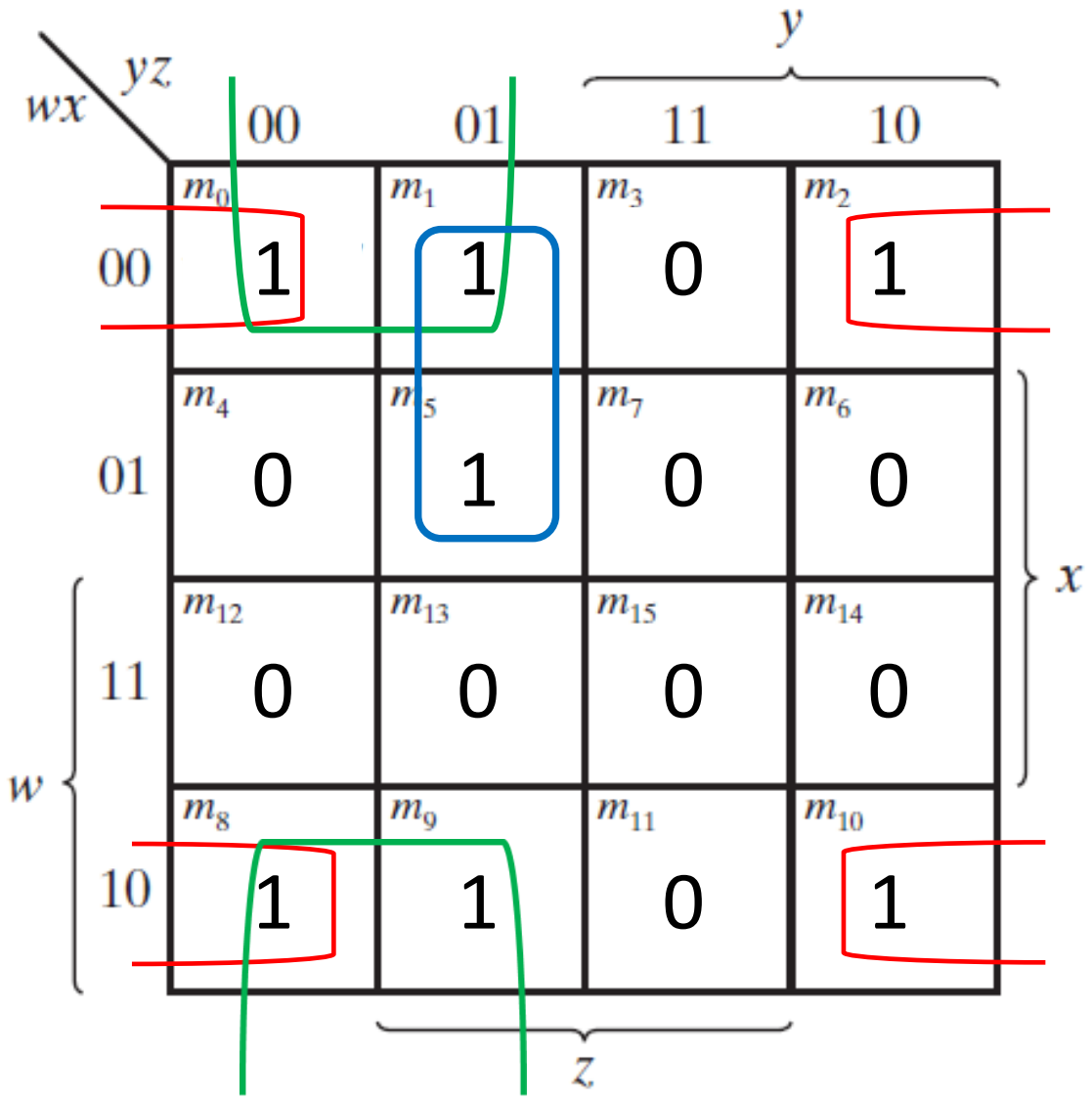
Product of Sums

- The minimized Boolean functions derived from the K-map were expressed in *sum-of-products* form
- With a minor modification, the product-of-sums form can be obtained
- The 1's placed in the squares of the map represent the *minterms* of the function
- From this observation, we see that the complement of a function is represented in the map by the squares not marked by 1's
- *If we mark the empty squares by 0's and combine them into valid adjacent squares, we obtain a simplified sum-of-products expression of the complement of the function (i.e., of F')*
- The complement of *F'* gives us back the function *F* in product-of-sums form (a consequence of DeMorgan's theorem)

Product of Sums

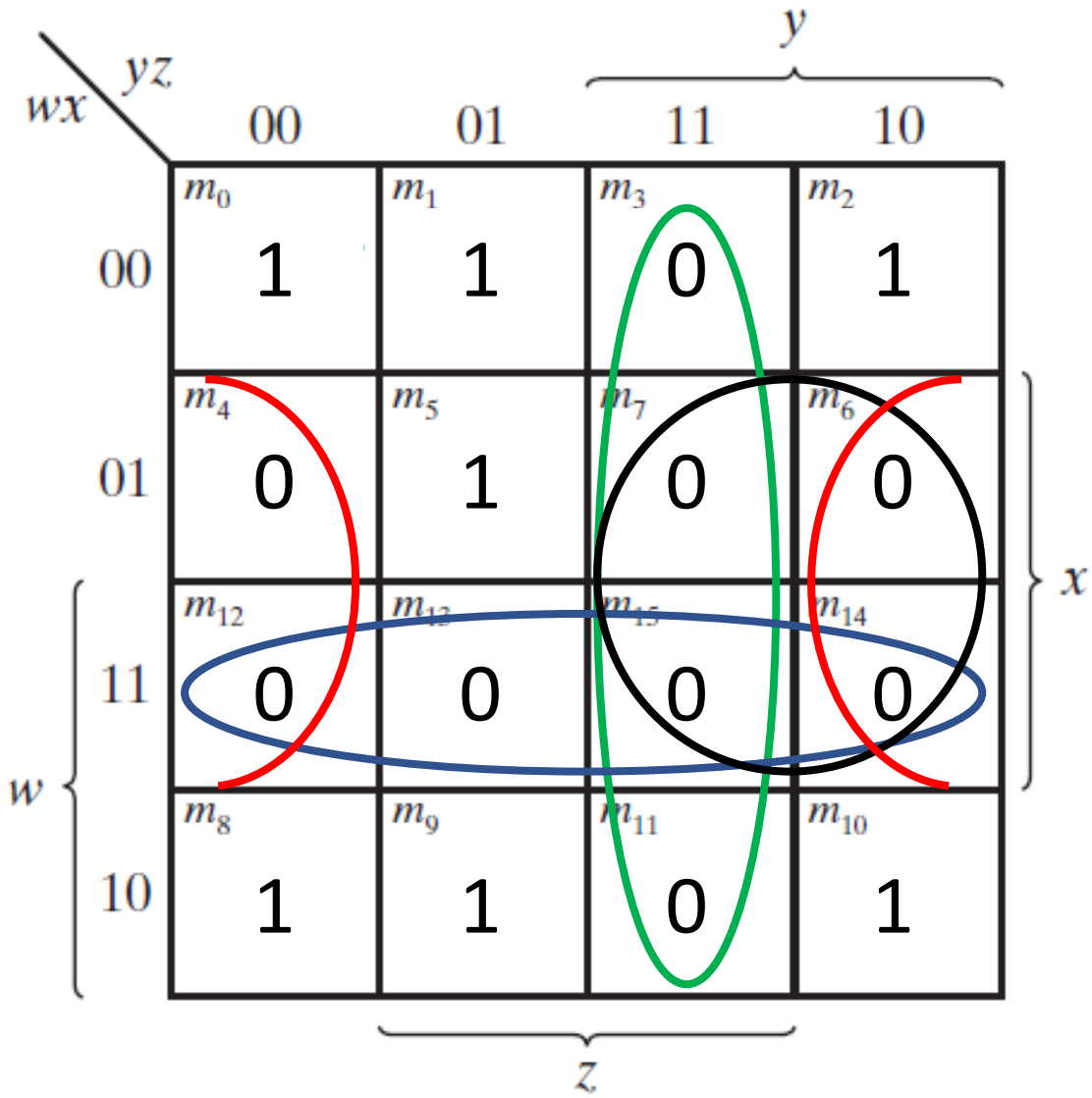
- Simplify the following Boolean function into (a) sum-of-products form and (b) product-of-sums form:
  $F\ (w, x, y, z)\ = \sum(0, 1, 2, 5, 8, 9, 10)$
- Two clusters of four squares: $x'z'$ and $x'y'$
- One cluster of two squares: $w'y'z$
- Thus, the function is:

$$F = x'z' + x'y' + w'y'z$$



Product of Sums

- The original function is:
$$F = x'z' + x'y' + w'y'z$$
- Now, let us see the complement of the function: F’
- This can be obtained from clustering all the 0s together
- We have four clusters of four (*prime implicants*)
- Of these, the *essential prime implicants* are: yz, wx, xz'
- Thus, $F' = wx + yz + xz'$



Product of Sums

- The original function is:
$$F = x'z' + x'y' + w'y'z$$
- Thus, the complement function is $F' = wx + yz + xz'$
- Now, we can obtain F back from F' using the DeMorgan's theorems
- Thus,
$$F = (w' + x')(y' + z')(x' + z)$$
- Hence, the actual simplest implementation of a function can be through its complement (product of sum)