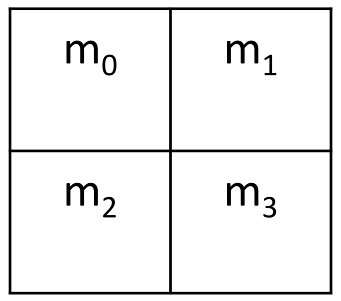
Gate-level minimization

- Gate-level minimization – optimal gate-level implementation of the Boolean functions describing a digital circuit.
- Simplification of Boolean expression by algebraic method lacks specific rules to predict each succeeding step during manipulation.
- The Map method provides simple, straightforward procedure for minimization of Boolean function.
- The Map method is pictorial form of truth table also known as *Karnaugh map* or *K-map*.
- K-map consist of squares, with each square representing one minterm of the function to be minimized.
- Since the Boolean function is expressed as sum of minterms, the area enclosed by those squares whose minterms are included in the function can be expressed as Boolean function.
- The Map shows visual diagram of all possible ways a function may be expressed in Standard form.
- By recognizing various patterns, alternative algebraic expression can be derived, from which simplest can be selected.

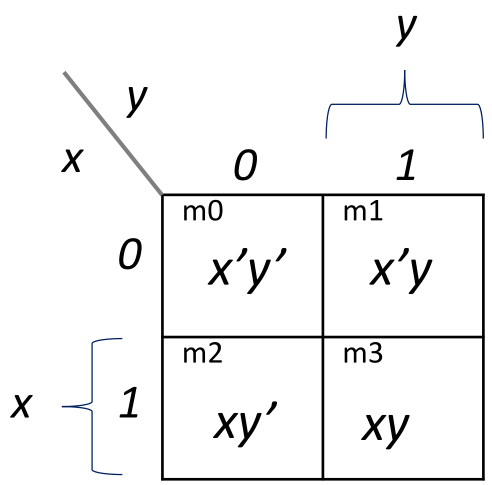
Gate-level minimization

- Simplified expression produced by the map will always be in one of the two Standard forms: Sum of products or Product of sum.
- Simplest algebraic expression is with minimum number of terms and with minimum number of literals in each term.
- Simplest expression is not unique i.e. it is possible to find two or more expressions satisfying minimization criteria. In such case, either case is satisfactory.

Two-variable map



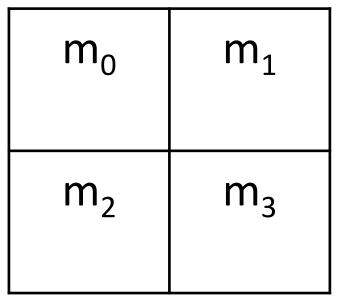
(a)



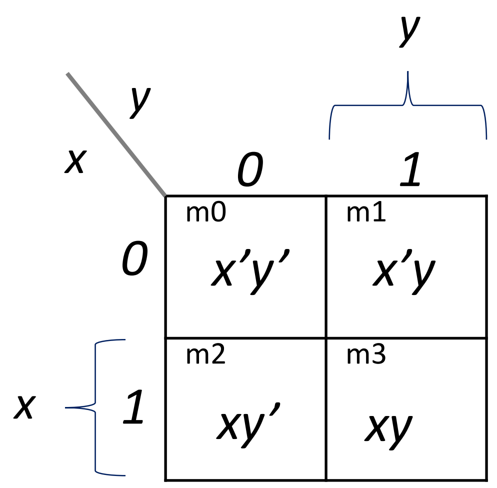
(b)

Gate-level minimization

Two-variable K-map



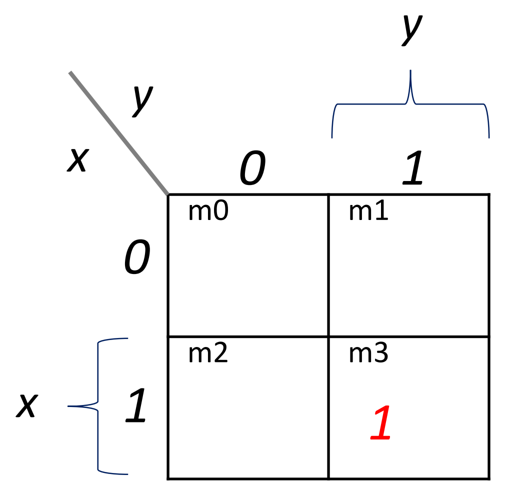
(a)



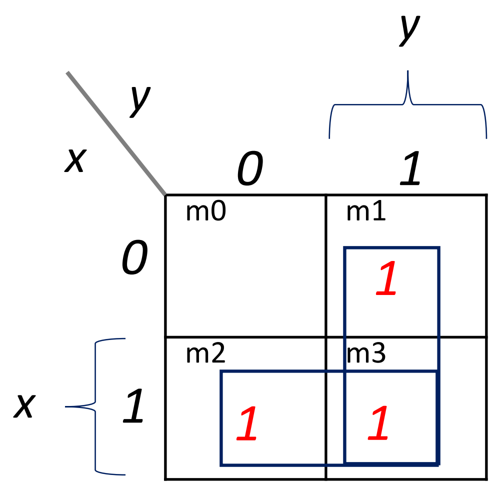
(b)

- For two variables there are four minterms, hence four squares in the map, one for each minterm. (Fig. a)
- Fig. (b) shows relationship between squares and two variables.
- 0 and 1 represent values of variables.
- x is primed in row 0 and unprimed in row 1.
- Similarly, y is primed in row 0 and unprimed in row 1.

Representation of Two-variable function in K-map



(a) xy



(b) x+ y

- A 1 is placed in any square indicates that the corresponding minterm is included in the output expression and 0 or no entry indicates that the minterm is not included in the output expression.
- In Fig. (a), since function xy is equal to m_3, 1 is placed inside the square that belongs to m_3.
- In Fig. (b), function x+y is represented by three squares marked with 1.

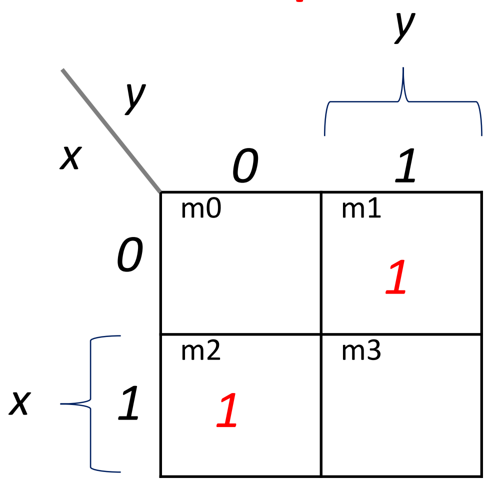
$$F = m_1 + m_2 + m_3 = x'y + xy' + xy = x+y$$

Representation of Two-variable function in K-map

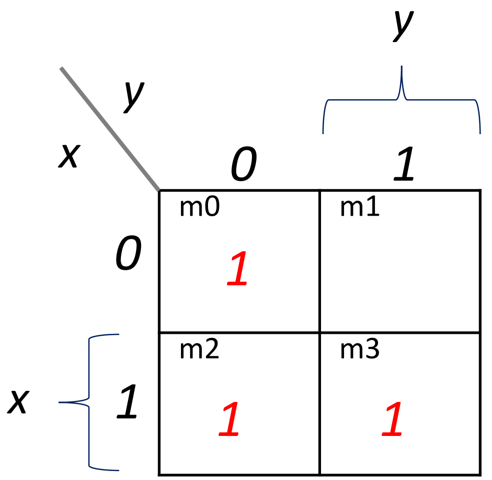
Ex.1 Map the expression F = $x'y + xy'$

The expression in minterm is $m_1 + m_2 = \Sigma\,(1,2)$

Therefore K-map is



Ex.2 Map the expression F = $\Sigma\,(0,2,3)$

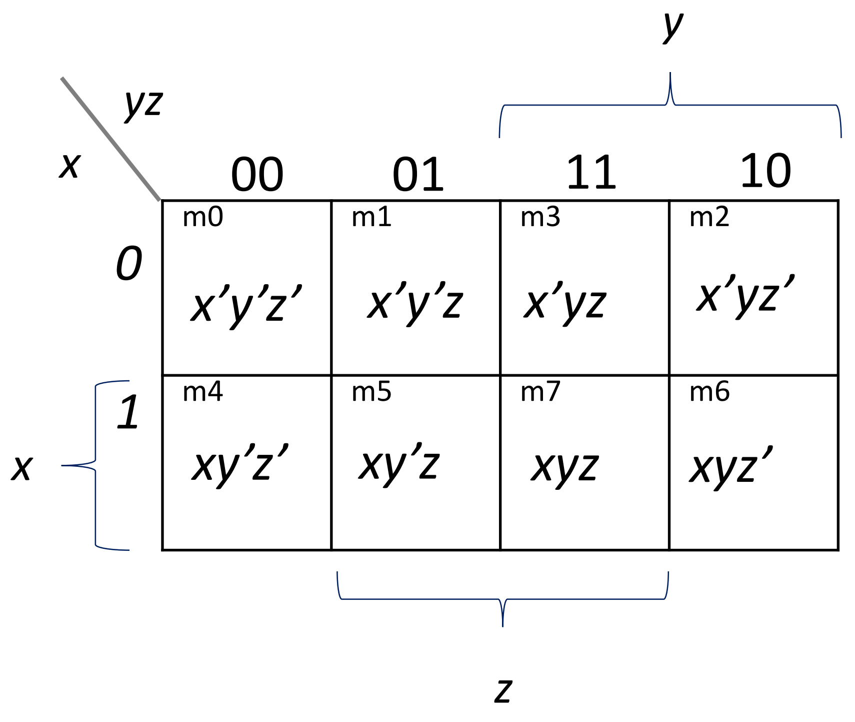


Three-variable K-map

- The Map consist of eight squares: There are eight minterms for three binary variables.
- Minterms are arranged in the sequence of Gray code.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

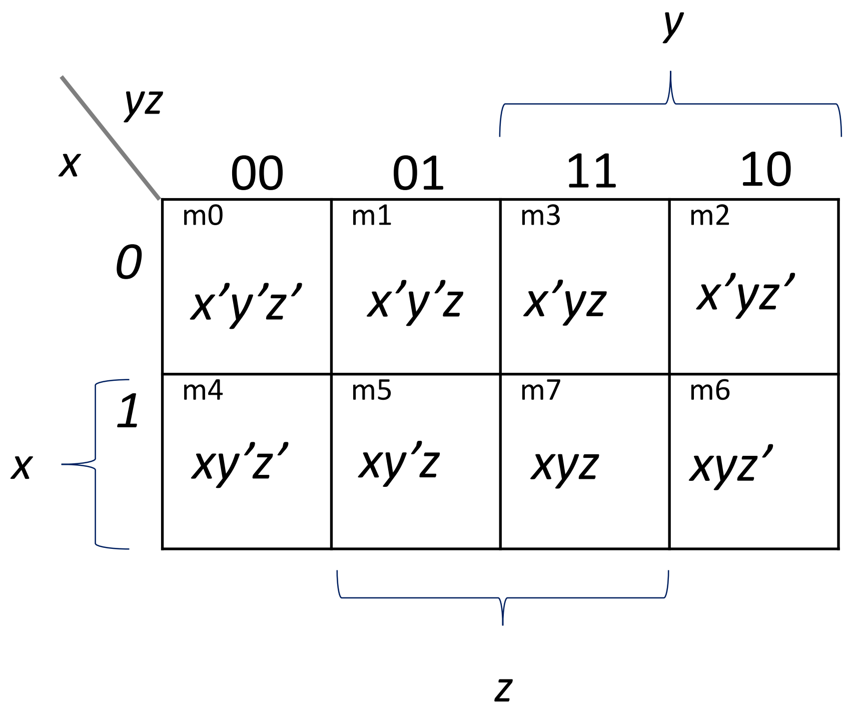
(a)



(b)

Three-variable K-map

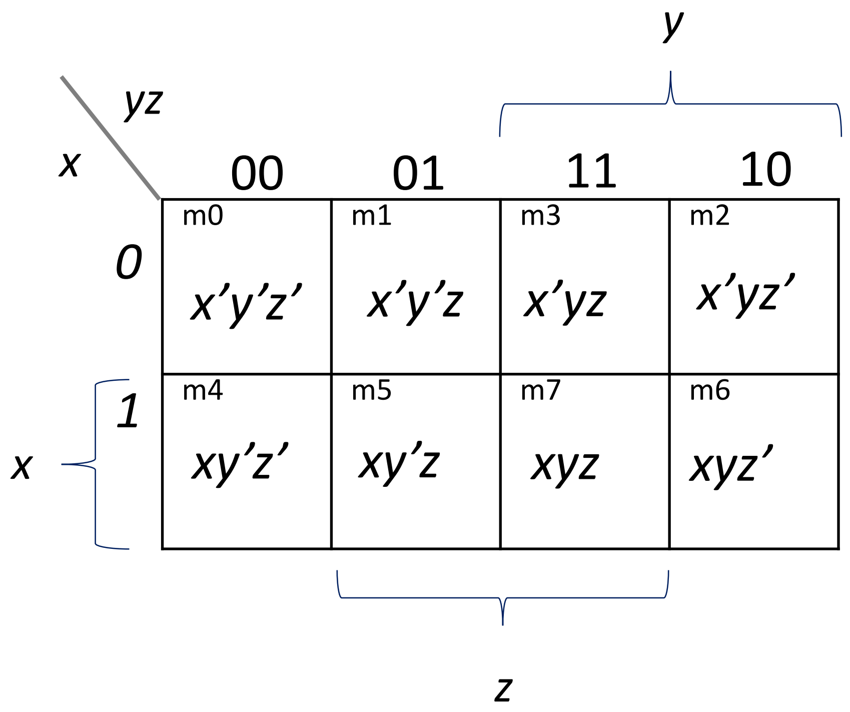
- The Map consist of eight squares: There are eight minterms for three binary variables.
- Minterms are arranged in the sequence of Gray code.
- Numbers in each row and column shows relationship between squares and three variables.
- Each cell of a map corresponds to a unique minterm.
- There are four squares in which each variable is equal to 1 therefore variable is unprimed and four in which each is equal to 0 and hence corresponding variable is primed.



Simplification of Boolean function using K-map

Three-variable

- Any two adjacent squares in the map differ by only one variable, which is primed in one square and unprimed in other.
- ex. m_5 and m_7
- Sum of two minterms in adjacent squares can be simplified to single AND term consisting two literals.
- $m_5 + m_7 = xy'z + xyz = xz(y'+y) = xz$
- Two squares differ by variable y, which can be removed when sum of two minterms is formed.



x \ yz	00	01	11	10
0	m0 $x'y'z'$	m1 $x'y'z$	m3 $x'yz$	m2 $x'yz'$
1	m4 $xy'z'$	m5 $xy'z$	m7 xyz	m6 xyz'

- *Any two minterms in adjacent square (vertically or horizontally not diagonally) that are OR ed together will cause removal of dissimilar variable.*

Simplification of Boolean function using K-map

Ex. 1. Simplify the Boolean function F(x,y,z) = Σ (2,3,4,5)

- 1 is marked in each minterm that represent the function.
- Find possible adjacent squares.
- Upper rectangle represents area enclosed by

  $m_3 + m_2 = x'yz + x'yz' = x'y$

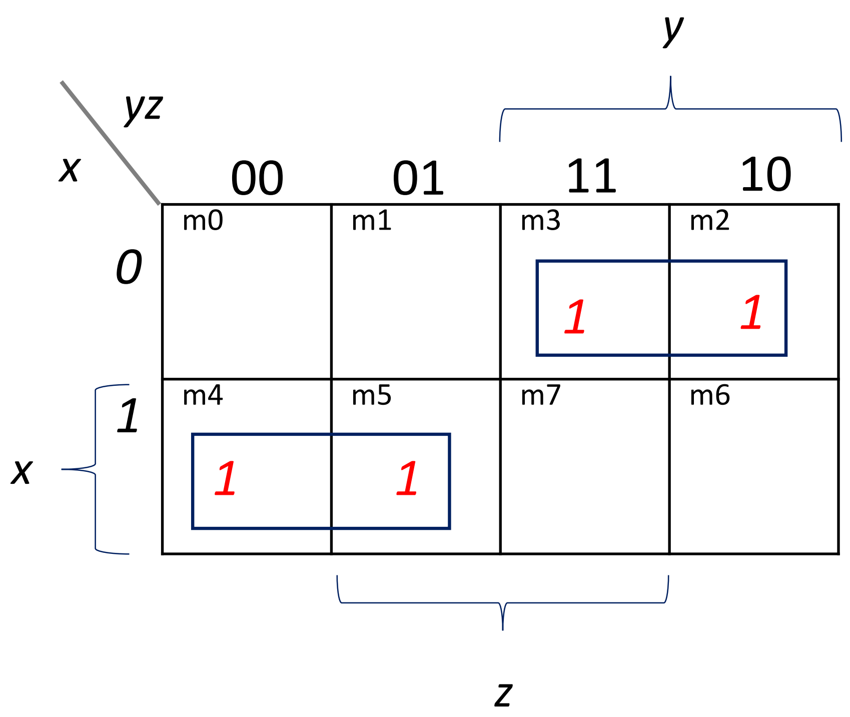
- lower rectangle represents area enclosed by

  $m_4 + m_5 = xy'z' + xy'z = xy'$

Logical sum of two product terms gives simplified expression

F = x'y + xy'

Three-variable



- *Any two minterms in adjacent square (vertically or horizontally not diagonally) that are OR ed together will cause removal of dissimilar variable.*

Simplification of Boolean function using K-map

Ex. 2 Simplify the Boolean function F(x,y,z) = Σ (3,4,6,7)

- There are four squares marked with 1's, one for each minterm of function.
- Two adjacent squares in third column are combined to five two-literal term.

$$m_3 + m_7 = x'yz + xyz = yz$$

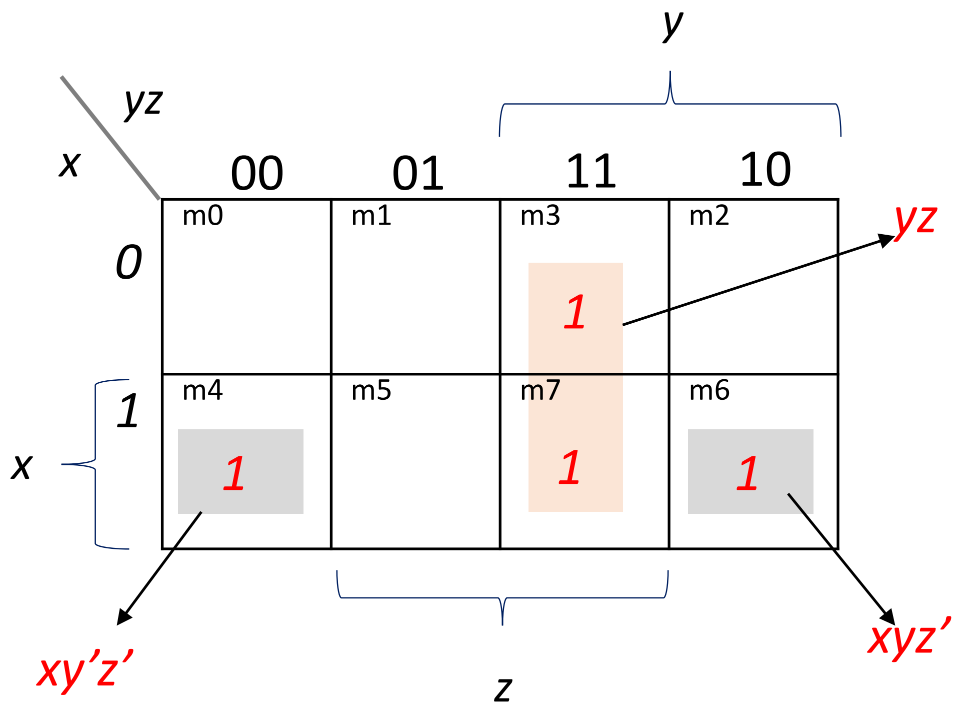
- Remaining two squares m_4 and m_6 with 1's are adjacent because these minterms differ only by one variable.

Therefore, $m_4 + m_6 = xy'z' + xyz' = xz'$

- The simplified function will be

$$F = y\,z + xz'$$

Three-variable



Simplification of Boolean function using K-map

- Consider, any combination of four adjacent squares in three variable map.
- Logical sum of any such four minterms results in an expression with only one literal.

ex. $m_0 + m_2 + m_4 + m_6 = x'y'z' + x'yz' + xy'z' + xyz'$

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

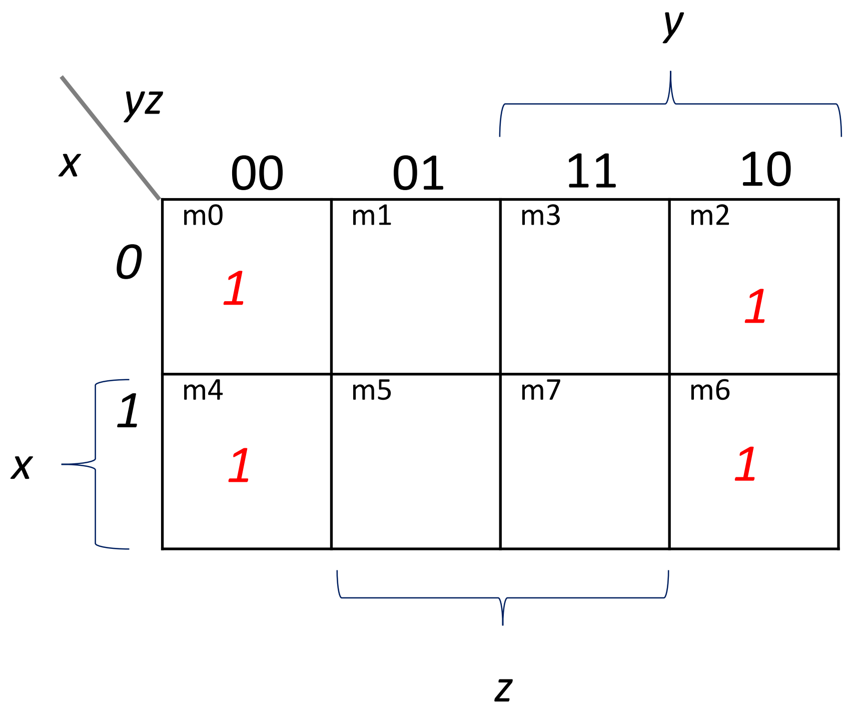
$$= x'z' + xz'$$

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

$$= z'$$

- Number of adjacent squares that may be combined must always represent a number that is power of two i.e. 1,2,4, and 8.
- Combination of more adjacent squares will result in product term with fewer literals.

Three-variable



- One square represents one minterm with three literals;
- Two adjacent squares represent a term with two literals.
- Four adjacent squares represents a term with one literal.
- *Eight adjacent squares of entire map produce a function that is always equal to 1.*

Ex.3 Simplify the Boolean function $F(x,y,z) = \Sigma\ (0,2,4,5,6)$

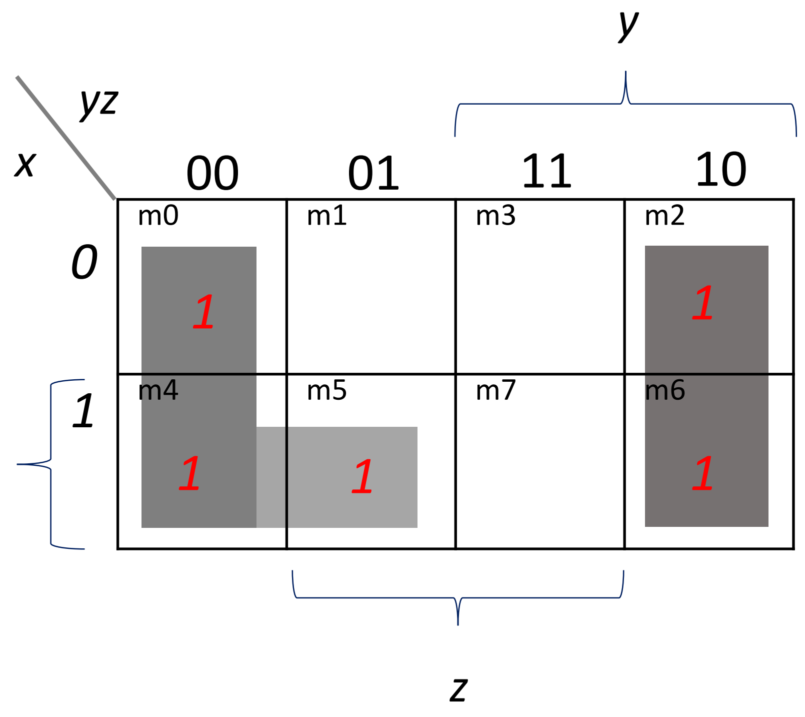
- First combine four adjacent squares: two in the first column and two in the last column, give term with single literal z'.

$m_0 + m_4 + m_2 + m_6 = x'y'z' + xy'z' + x'yz' + xyz'$

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

$= z'(y'+y)$

$= z'$



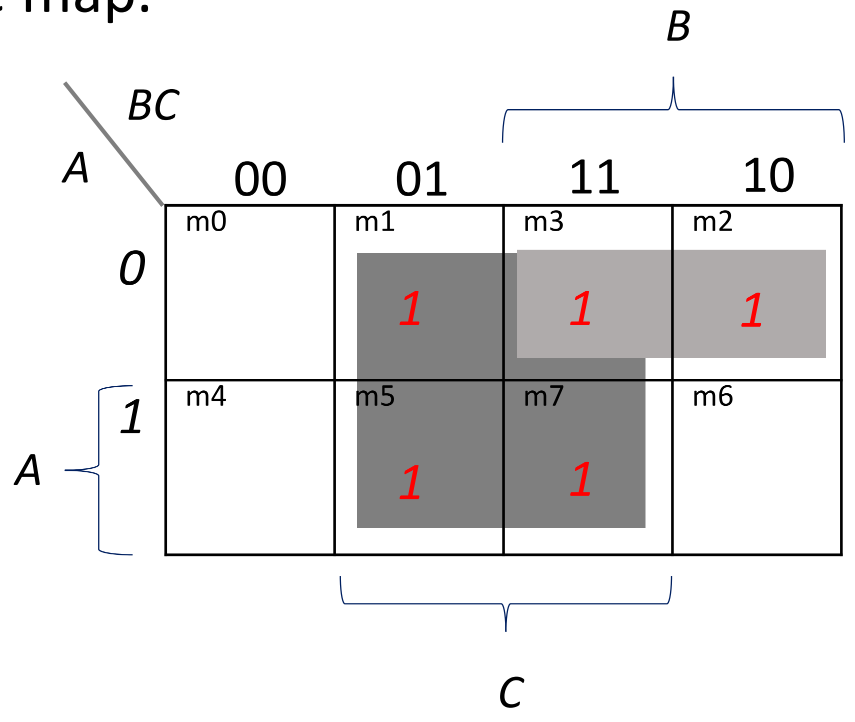
- Combine remaining single minterm m_5 with m_4 which has already been used once because these two squares give two-literal term.
- $m_4 + m_5 = xy'z' + xy'z = xy'(z'+z) = xy'$
- Simplified function is $F(x,y,z) = z'+xy'$

- If a function is not expressed in sum- of- minterms form, it is possible to use map to obtain minterms of a function and then simplify the function to an expression with minimum number of terms.
- The simplified expression of the function should be in sum-of-product form.
- Each term is plotted in the map in one, two or more squares.
- Minterms of the function are then read directly from the map.

Ex.4. F = A’C + A’B + AB’C + BC

(a) Express the function as sum of minterms.

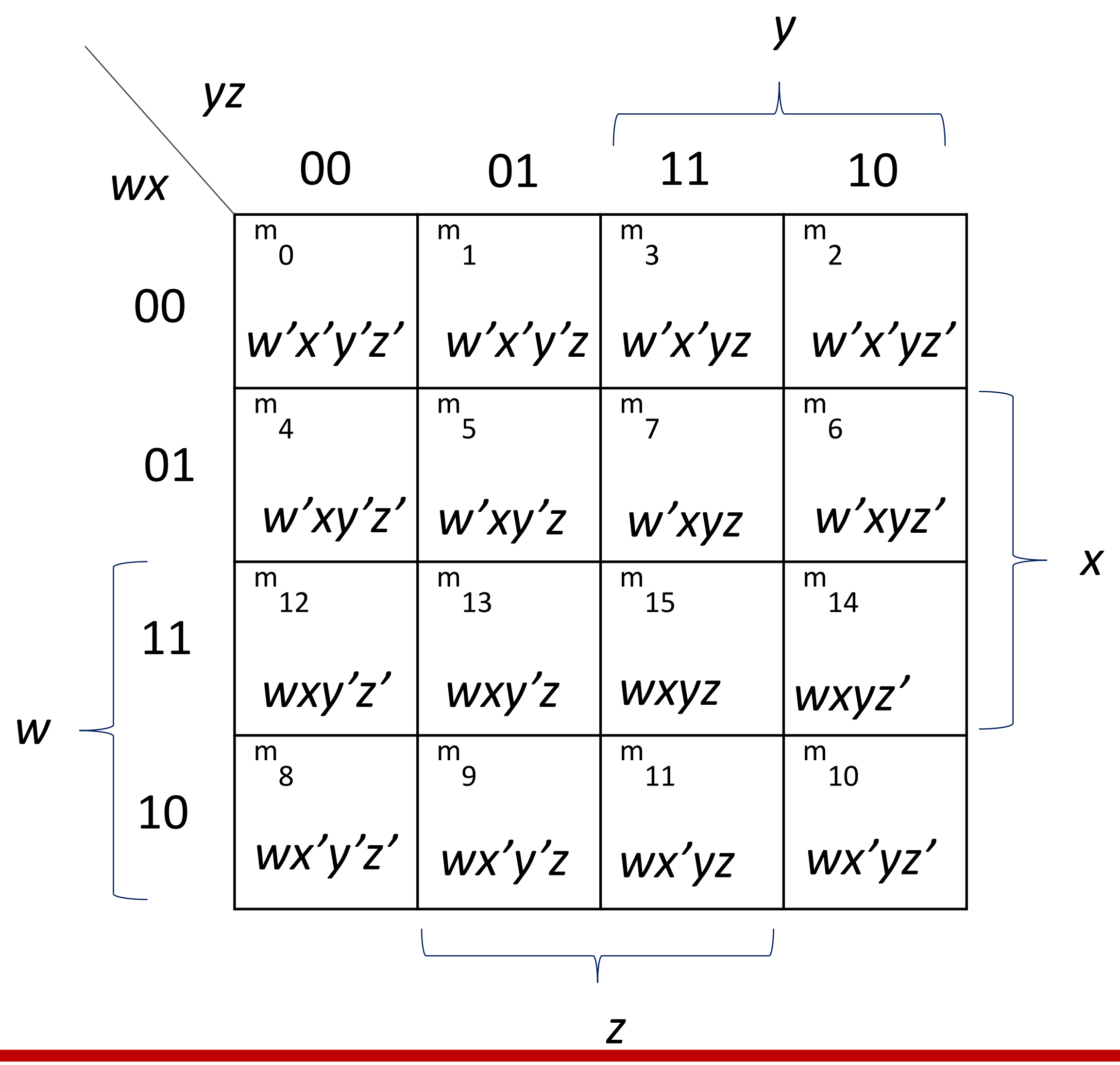
(b) Find the minimal sum-of products



Four-variable K-map

- For four variables there are 16 minterms hence 16 squares in the map.

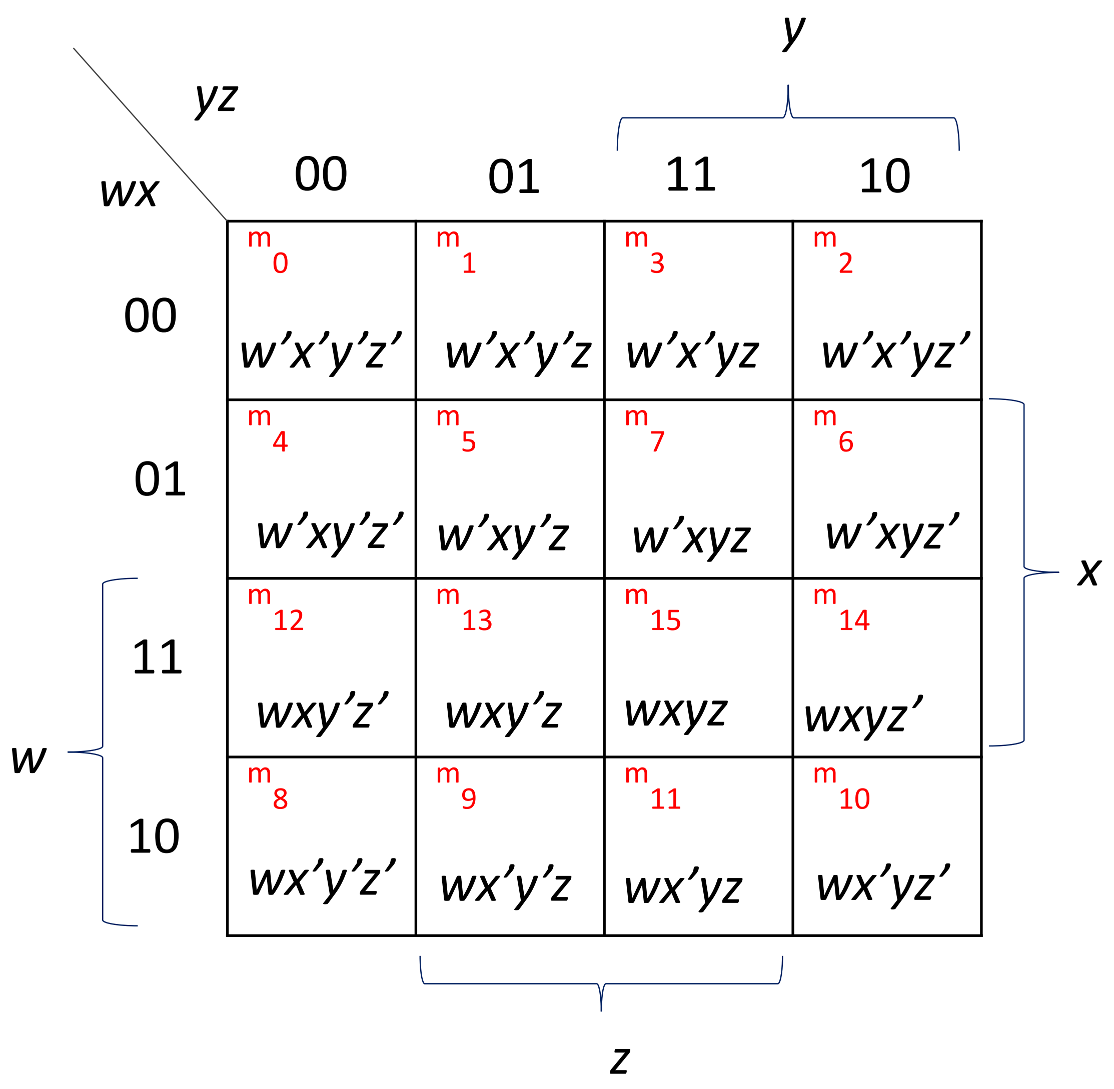
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}



wx \ yz	00	01	11	10
00	m_0 *w'x'y'z'*	m_1 *w'x'y'z*	m_3 *w'x'yz*	m_2 *w'x'yz'*
01	m_4 *w'xy'z'*	m_5 *w'xy'z*	m_7 *w'xyz*	m_6 *w'xyz'*
11	m_{12} *wxy'z'*	m_{13} *wxy'z*	m_{15} *wxyz*	m_{14} *wxyz'*
10	m_8 *wx'y'z'*	m_9 *wx'y'z*	m_{11} *wx'yz*	m_{10} *wx'yz'*

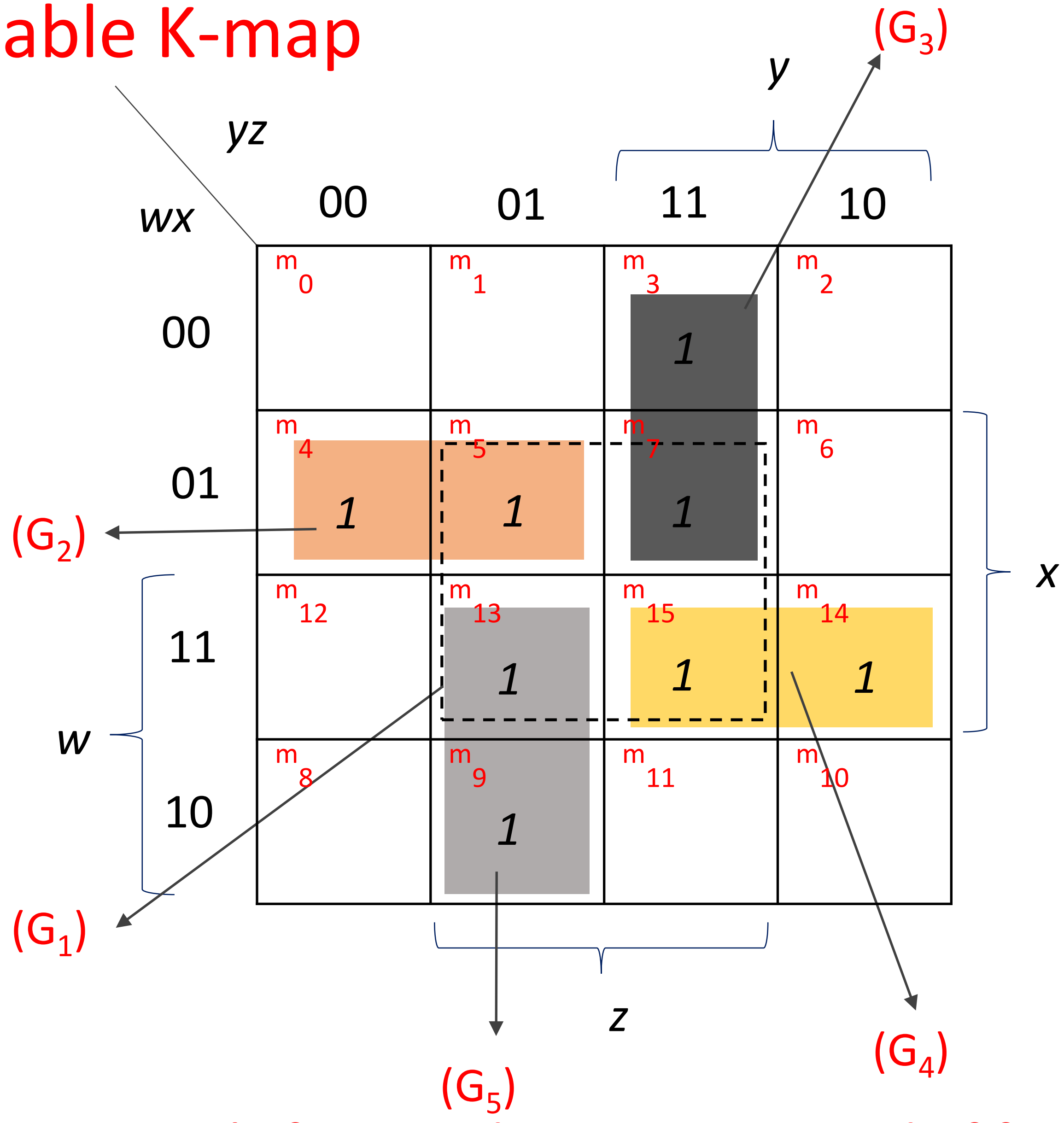
Four-variable K-map

- The combination of adjacent squares that is determined from four-variable map:
- One square represents one minterm, giving a term with four literals.
- Two adjacent squares represents a term with three literals.
- Four adjacent squares represent a term with two literals.
- Eight adjacent squares represent a term with one literals.
- Sixteen adjacent squares produce a function that is always equal to 1.



Four-variable K-map

- Pair: Group of two adjacent minterms. A *pair* eliminates one variable in output expression.
- Quad: Group of four adjacent minterms. A *quad* eliminates two variables in output expression.
- Octet: Group of eight adjacent minterms. A *octet* eliminates three variable in output expression.
- Redundant Group: A group in which all the elements are covered by some other group.



G_1 is redundant group and final out expression contains only four product terms instead of five terms.

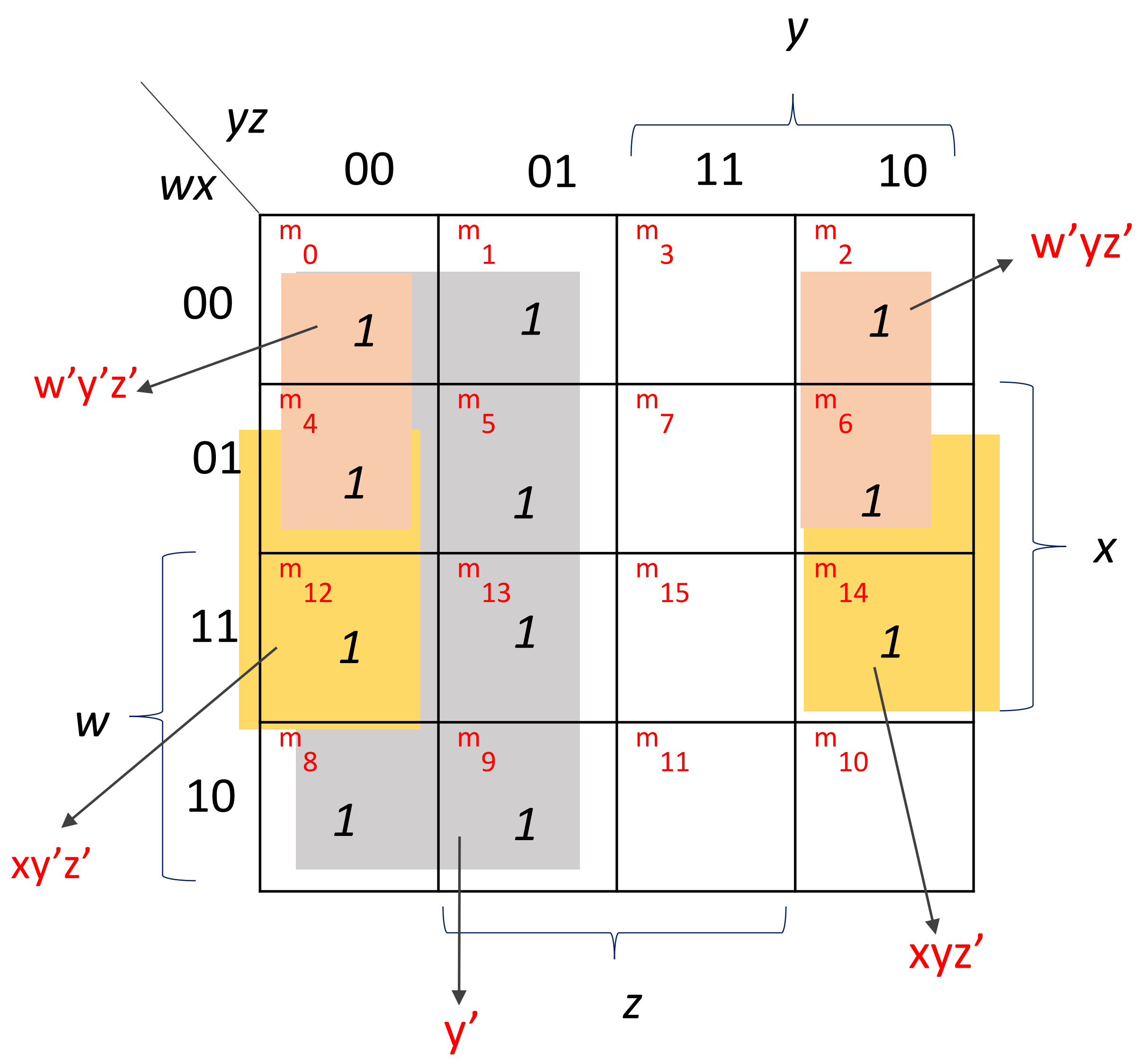
Four-variable K-map

Ex.1. Simplify the Boolean function

F(w,x,y,z) = Σ (0,1,2,4,5,6,8,9,12,13,14)

- Eight adjacent squares combined to form a term with on literal, y'.
- Remaining three minterms at right can not be combined to give simplified term.
- Top two minterms at right combined with top two at left to give the term w'z'.
- Combine squares made of two middle rows and two end columns, to give the term xz'.
- Simplified function is

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

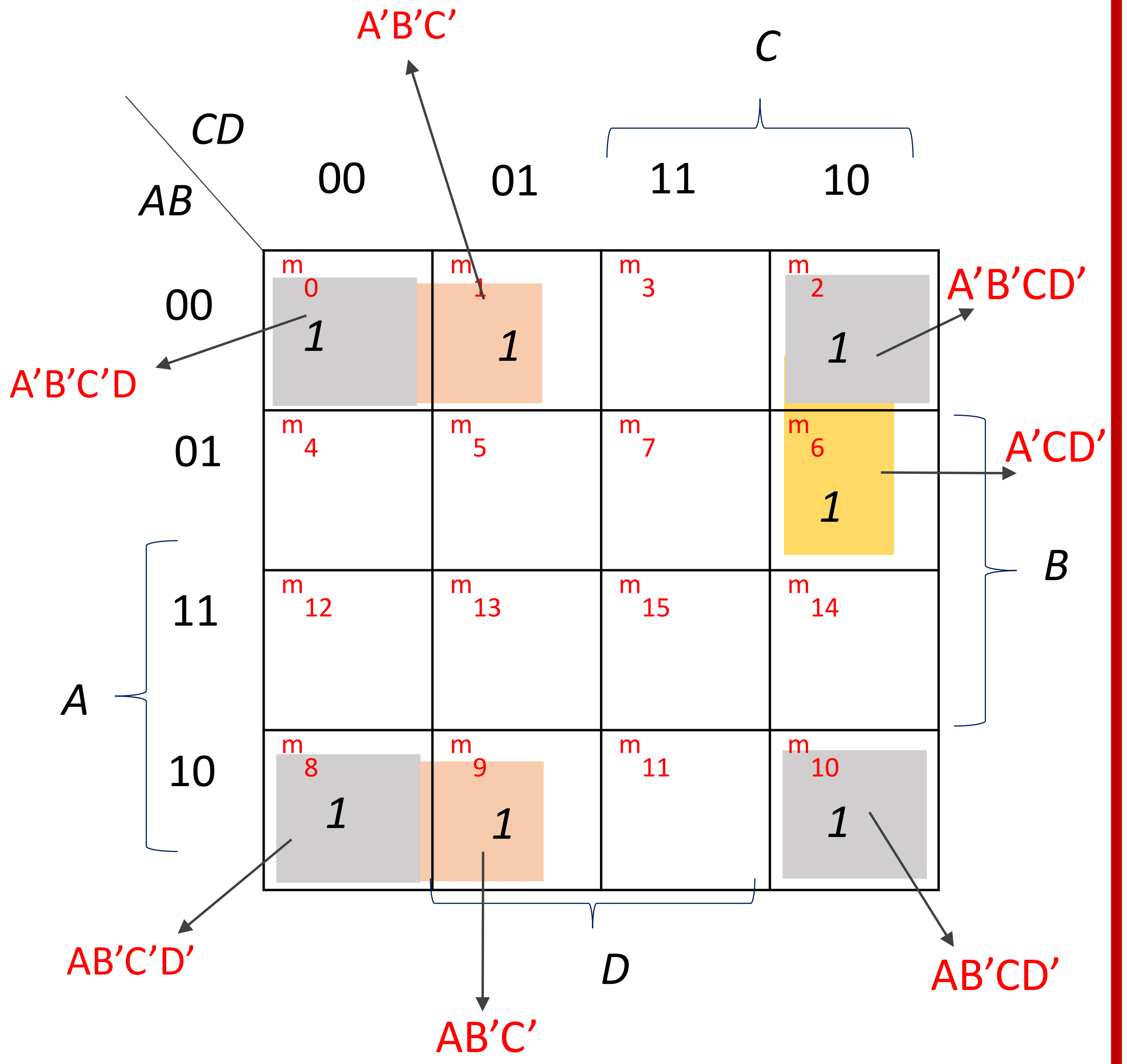


Four-variable K-map

Ex.1. Simplify the Boolean function

F = A'B'C' + B'CD' + A'BCD' + AB'C'

The simplified function F = B'D' + B'C' + A'CD'



Simplification of Boolean function in product-of-sums form

- The 1’s present in the map represent minterms of function.
- Minterms that are not included in the standard sum-of-product form of a function denote the complement of the function.
- Mark empty squares by 0’s and combine valid adjacent squares to obtain simplified expression of the complement of function i.e. *F’.*
- By taking complement of *F’* and using DeMorgans theorem, the simplified expression for function F can be obtained in product-of sums form.

Simplification of Boolean function in product-of-sums form

Ex. 1 Simplify the Boolean function *F* (A,B,C,D) = Σ (0,1,2,5,8,9,10) into

(a) sum-of-products (b) product-of-sums form.

- Squares marked with 0's represent minterms that are not present in function *F* therefore denote complement of F *i.e. F'*.
- Combining squares with 1's give simplified function in sum-of-products form

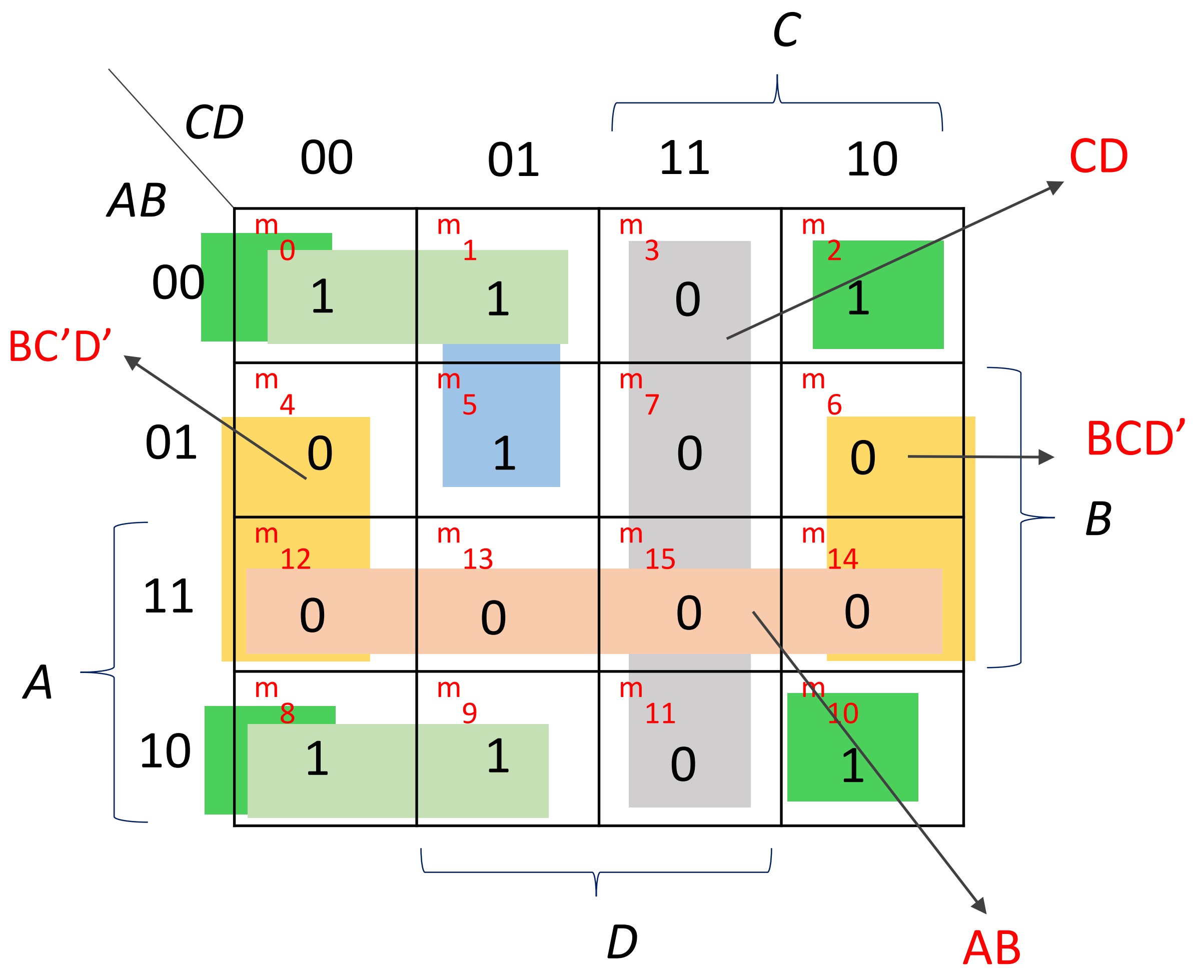
(a) F = B'D' +B'C' + A'C'D

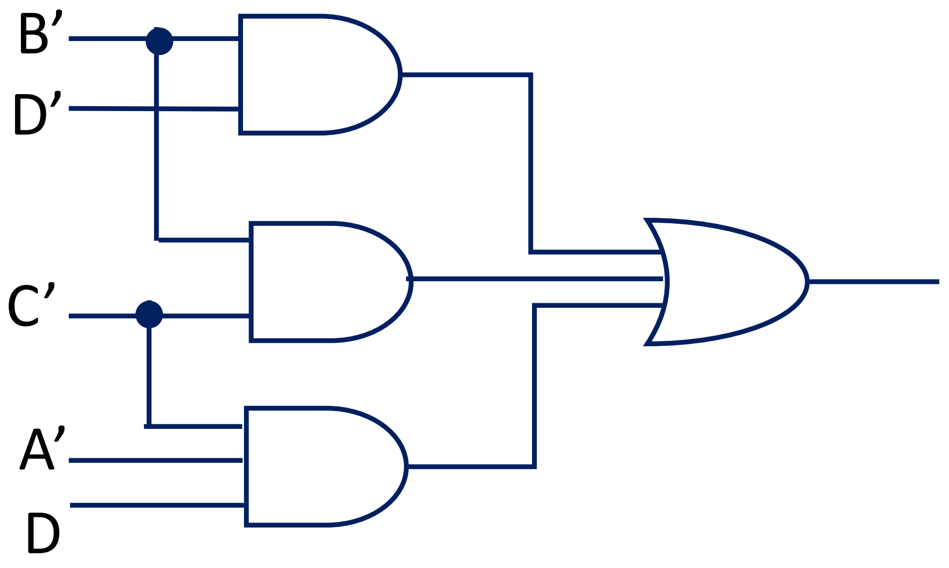
Combining squares marked with 0's give

F' = AB + CD + BD'

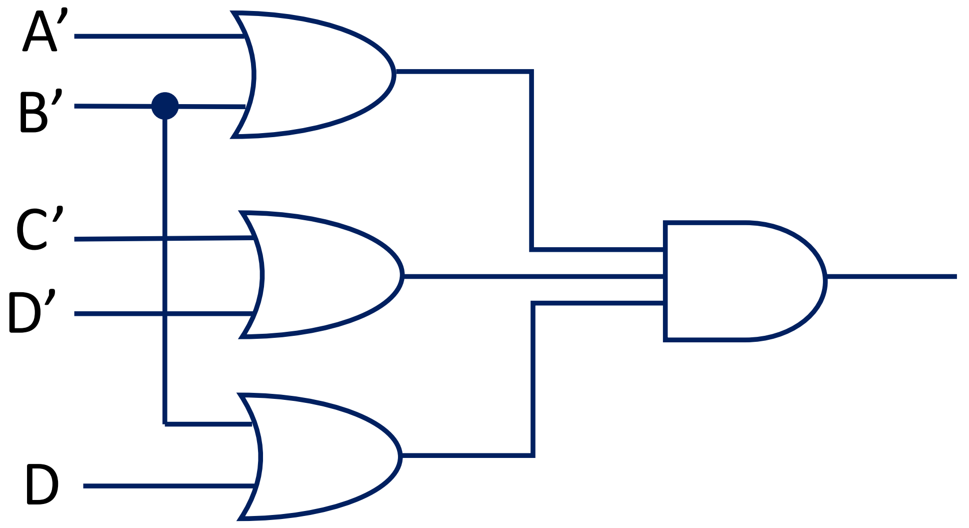
(F')' = (AB + CD + BD')' DeMorgans Theo.

(b) F = (A'+B') (C'+D') (B' + D) is simplified function in product-of-sums form.





F = B'D' +B'C' + A'C'D



F = (A'+B') (C'+D') (B' + D)

- Each configuration forms two levels of gates.
- Thus, implementation of a function in a standard form is said to be two-level implementation.

Simplification of Boolean function expressed in product-of-maxterms

Ex. 2. Simplify the function *F*, expressed in product-of-maxterms, from given truth table.

- Function *F* can be expressed in sum-of-minterms as

$F(x,y,z) = \Sigma\ (1,3,4,6)$

- In product-of-maxterm form

$F(x,y,z) = \Pi\ (0,2,5,7)$

- The 1's of the function represent *Minterms* and 0's of the function represents *Maxterms*.

- For sum-of-products, we combine 1's to obtain

$F = x'z + xz'$

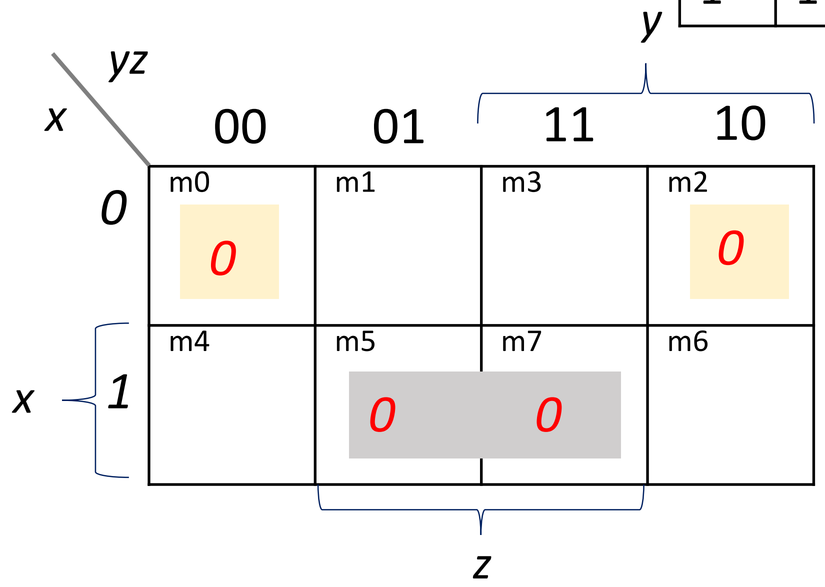
- For product-of-sums, we combine 0's to obtain

$F' = xz + x'z'$

$(F')' = (xz + x'z')'$

$F = (x'+z')\ (x + z)$

x	**y**	**z**	**F**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



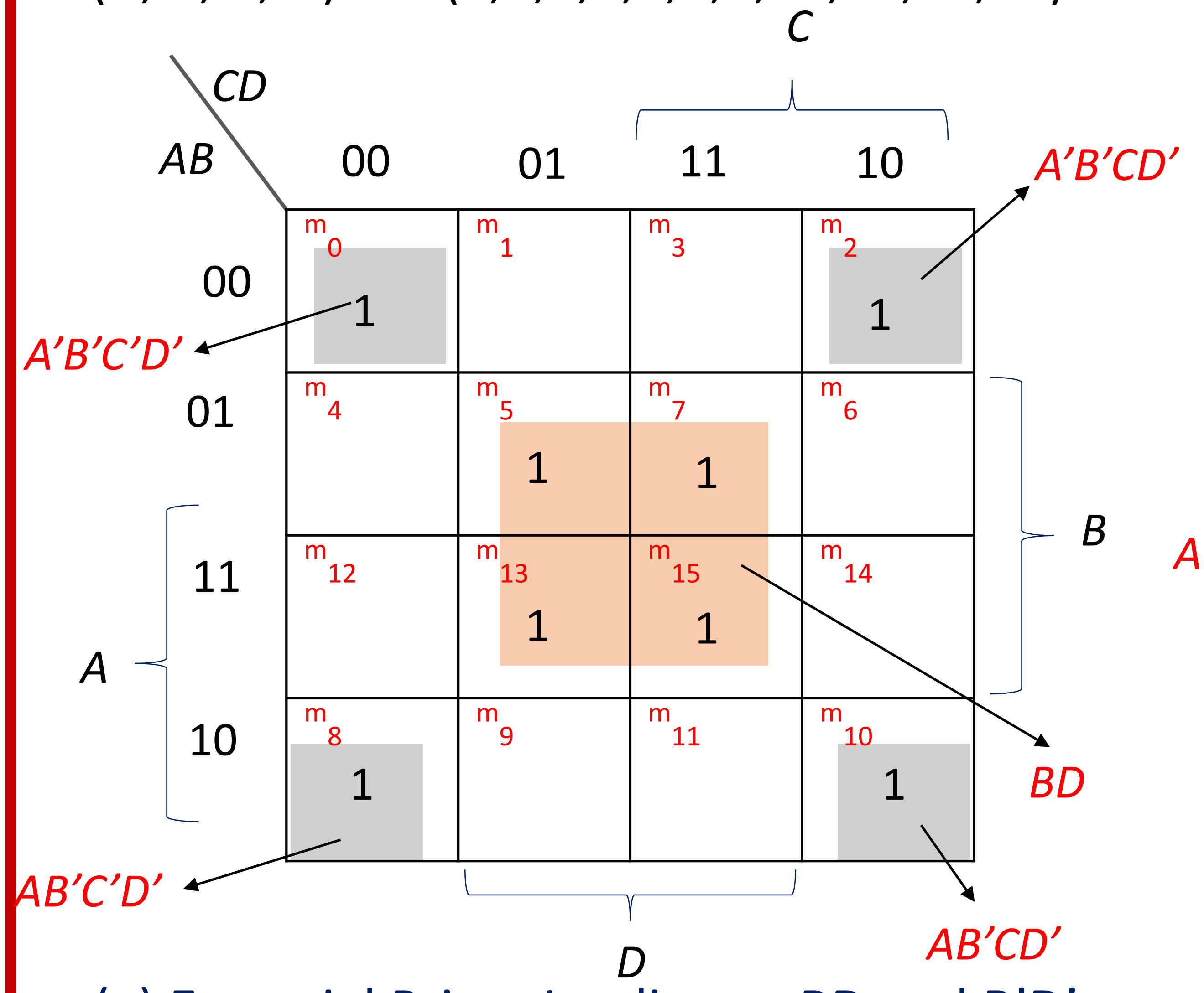
Prime Implicant

- Prime implicant: A product term obtained by combining maximum possible number of adjacent squares in the map.
    - Single 1’s on a map represents a prime implicant if it is not adjacent to any other 1’s.
    - Two adjacent 1’s form a prime implicant, if they are not within a group of four adjacent squares.
    - Four adjacent 1’s form a prime implicant if they are not within a group of eight adjacent squares.
- The prime implicant is essential if it is the only prime implicant that covers the minterm.
- If a minterm in a square is covered by only one prime implicant, then that prime implicant is called *essential*.

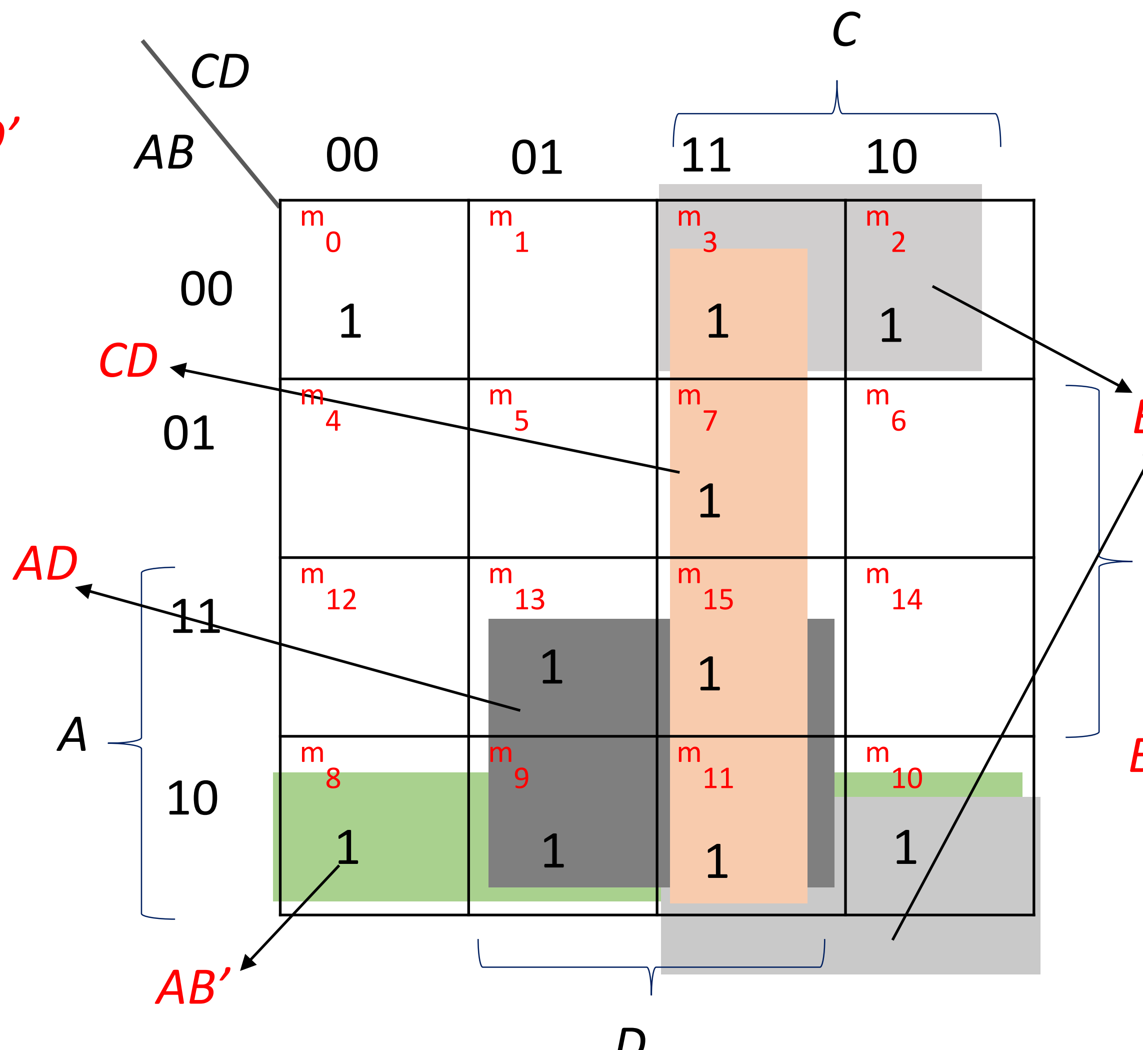
Simplified expression for the Boolean function is obtained from the logical sum of essential prime implicants, plus other prime implicants that may be needed to cover any remaining minterms not covered by essential prime implicants.

- Consider the following four-variable Boolean Function:

$F\ (A, B, C, D) = \Sigma\ (0,2,3,5,7,8,9,10,11,13,15)$



(a) Essential Prime Implicants BD and B'D'



(b) Prime Implicants CD,B'C,AD and AB'

- Simplified expression is obtained from the logical sum of two essential prime implicants and any two prime implicants that cover minterms m_3, m_9, and m_{11}.
- Therefore, four possible ways that the function can be expressed with four product term with two literals each:

$$F = BD + B'D' + CD + AD$$

$$F = BD + B'D' + CD + AB'$$

$$F = BD + B'D' + B'C + AD$$

$$F = BD + B'D' + B'C + AB'$$

Don’t care conditions

- Logical sum-of-minterms associated with Boolean function specify the condition when function is equal to 1.
- Function is equal to 0 for rest of the minterms.
- Both these conditions assumes that all the combinations of values of variables of function are valid.
- In practice, in some applications function is not specified for certain combinations of variables.

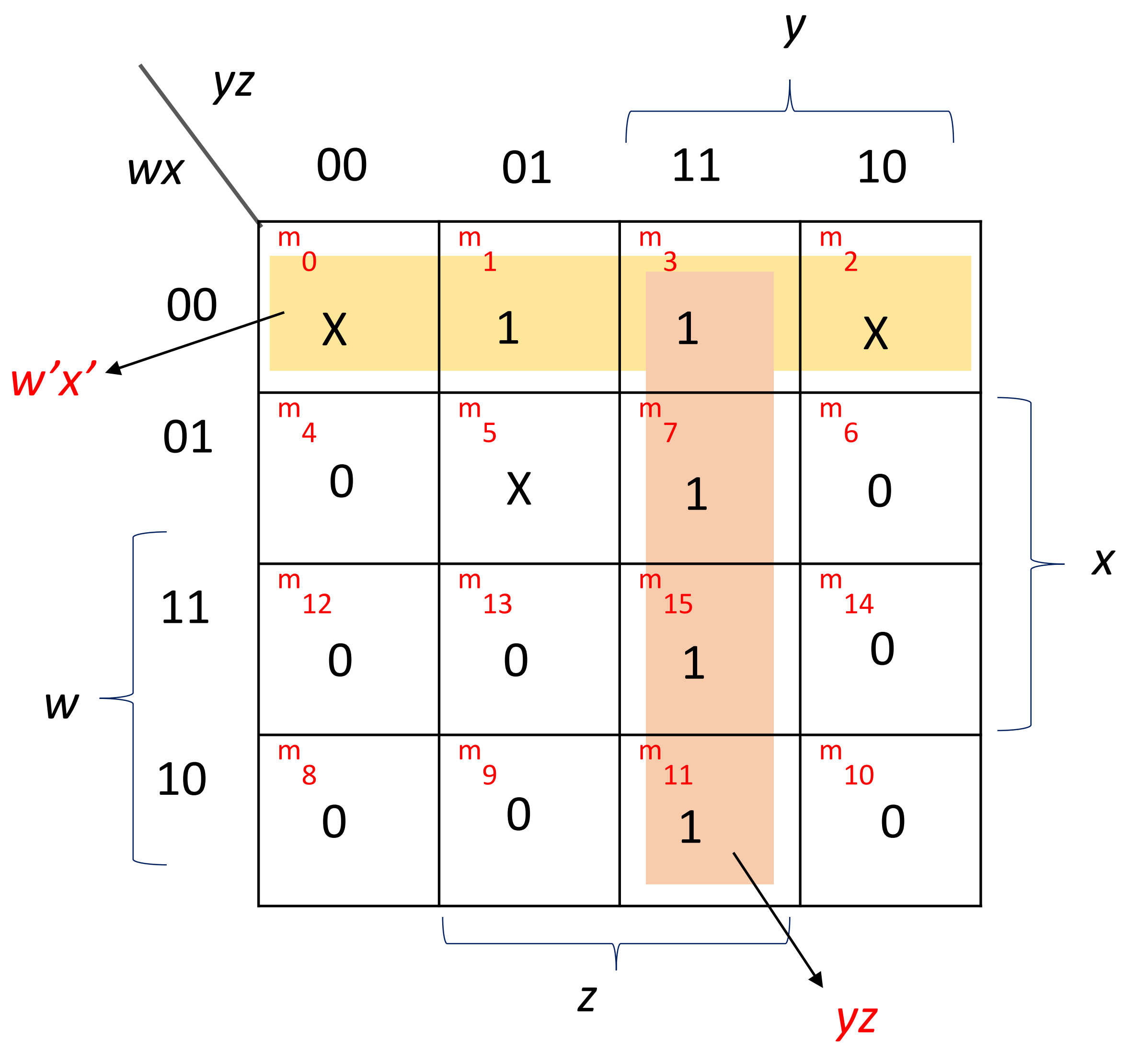
Ex. Four-bit binary code for the decimal digit has six combinations that are not valid and considered as unspecified.

- Functions that have unspecified outputs for some input combinations are called *incompletely specified functions*.
- The value assumed by the function for the unspecified minterm is not considered i.e. function can have any value for that unspecified minterm.
- Therefore, it is customary to call the unspecified minterms of a function as *don’t care conditions.*

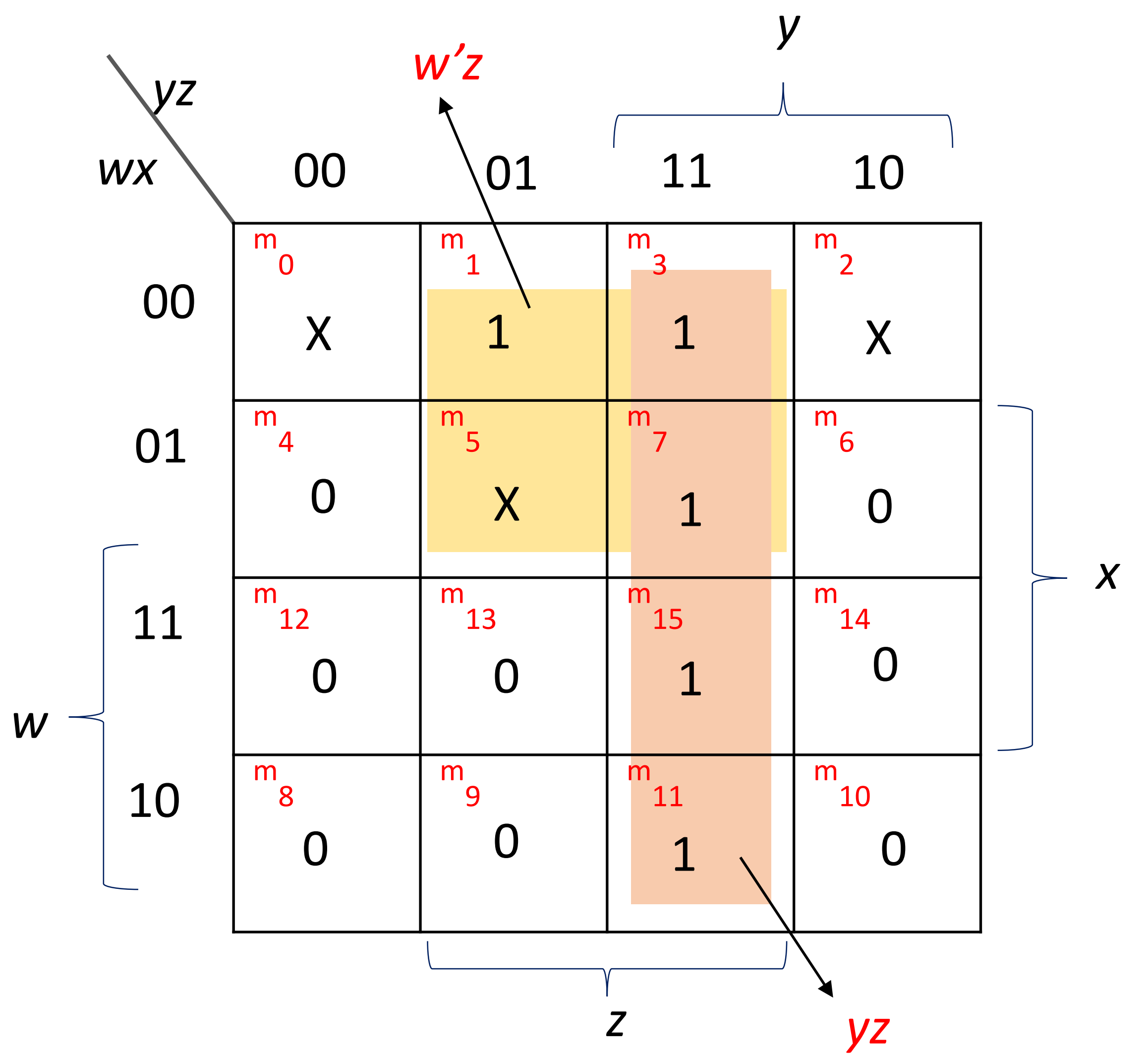
Don’t care conditions

- A don’t- care minterm is a combination of variables whose logical value is not specified.
- Such minterms can not be marked with 1 or 0.
- To distinguish don’t- care condition from 1’s and 0’s, a X is used.
- A ‘X’ inside square means we don’t- care if the value of function is 0 or 1 for that particular minterm.
- During simplification of the function, we can choose to include each don’t- care minterm with either 1’s or 0’s, depending on which combination gives the simplest expression.

- Ex. 1. Simplify the Boolean function *F* (w, x, y, z) = Σ(1,3,7,11,15) which has don't care conditions d (w, x, y, z) = Σ (0, 2, 5).



(a) F = yz + w'x'



(b) F = yz + w'z

- Don’t-care conditions are initially marked as ‘X’ in map and are considered as being either 0 or 1.
- The choice between 0 and 1 is made depending on the way the incompletely specified function is specified.
- Once the choice is made, the simplified function obtained will consist of a sum of minterms that includes those minterms, which are initially unspecified. And have been chosen to be included with 1’s.
- Therefore the simplified expressions for the F (w, x, y, z) = $\Sigma(1,3,7,11,15)$ with don’t care conditions d (w, x, y, z) = Σ (0, 2, 5) are

$$F\,(w, x, y, z) = yz + w'x' = \Sigma(0,1,2,3,7,11,15)$$

$$F\,(w, x, y, z) = yz + w'z = \Sigma(1,3,5,7,11,15)$$

- Both the expressions are acceptable because the only difference is in the value of F for don’t-care minterms.