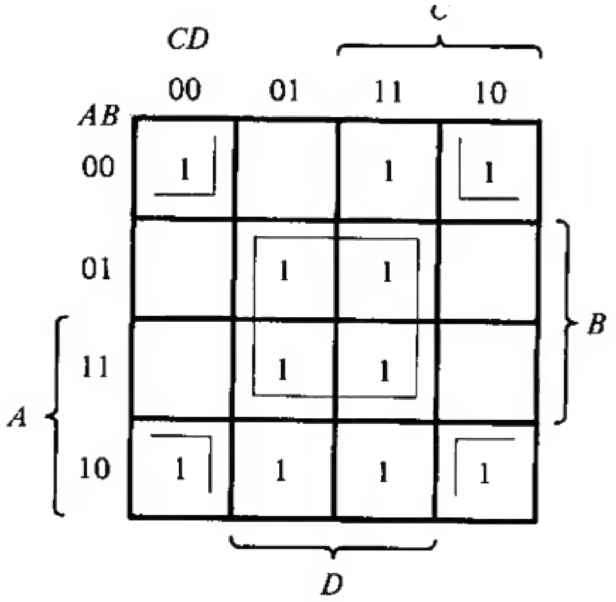
Digital Logic and Circuit
Paper Code: CS-102

Outline

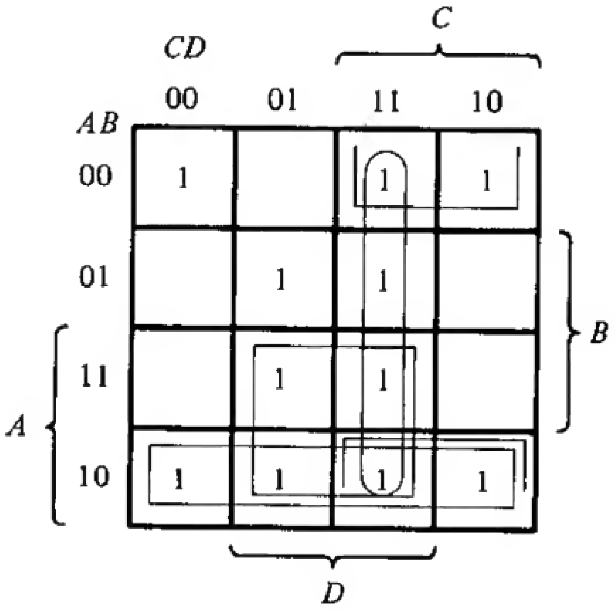
- Simplification of Boolean Functions using k-map
  - Prime and essential prime implicants
  - Sum of Minterm
  - Product of Maxterm

Prime implicants and essential prime implicants

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.

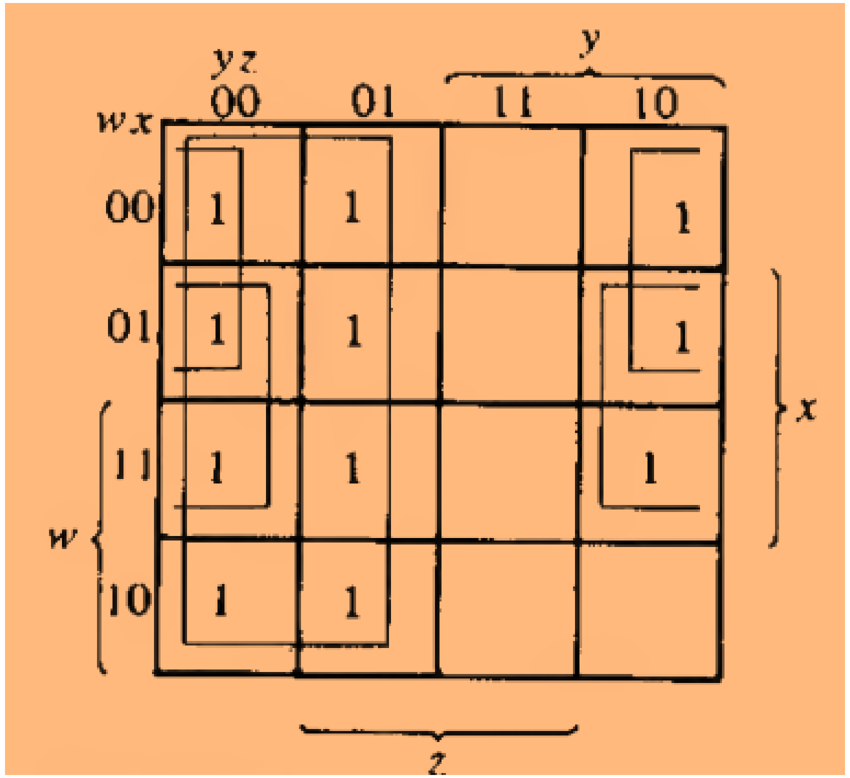


(a) Essential prime implicants BD and $B'D'$



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

Simplify the Boolean function $F(w, x, y, z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



The simplified function is
$F = y' + w'z' + xz'$

Simplify the Boolean function $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

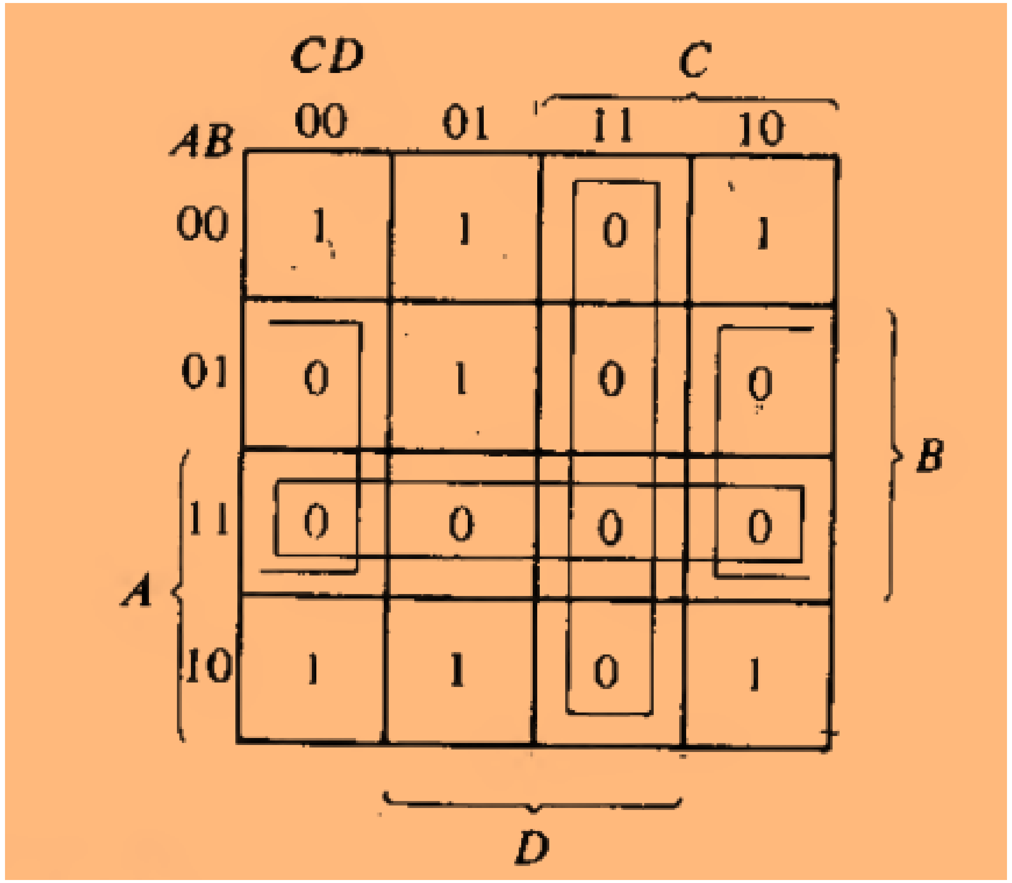
	Number of adjacent squares	Number of literals in a term in an n-variable map					
k	2^k	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$
0	1	2	3	4	5	6	7
1	2	1	2	3	4	5	6
2	4	0	1	2	3	4	5
3	8		0	1	2	3	4
4	16			0	1	2	3
5	32				0	1	2
6	64					0	1

PRODUCT OF SUMS SIMPLIFICATION

- The procedure for obtaining a minimized function in product of sums follows from the basic properties of Boolean functions.
- The 1 s placed in the squares of the map represent the minterms of the function.
- The minterms not included in the function denote the complement of the function. From this 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 expression of the complement of the function, i.e., of F ' .

Simplify the following Boolean function in (a) sum of products and (b) product of sum

$F(A, B, C, D) = \sum(0, 1, 2, 5, 8, 9, 10)$



Combining the squares with l's gives the simplified function in sum of products:

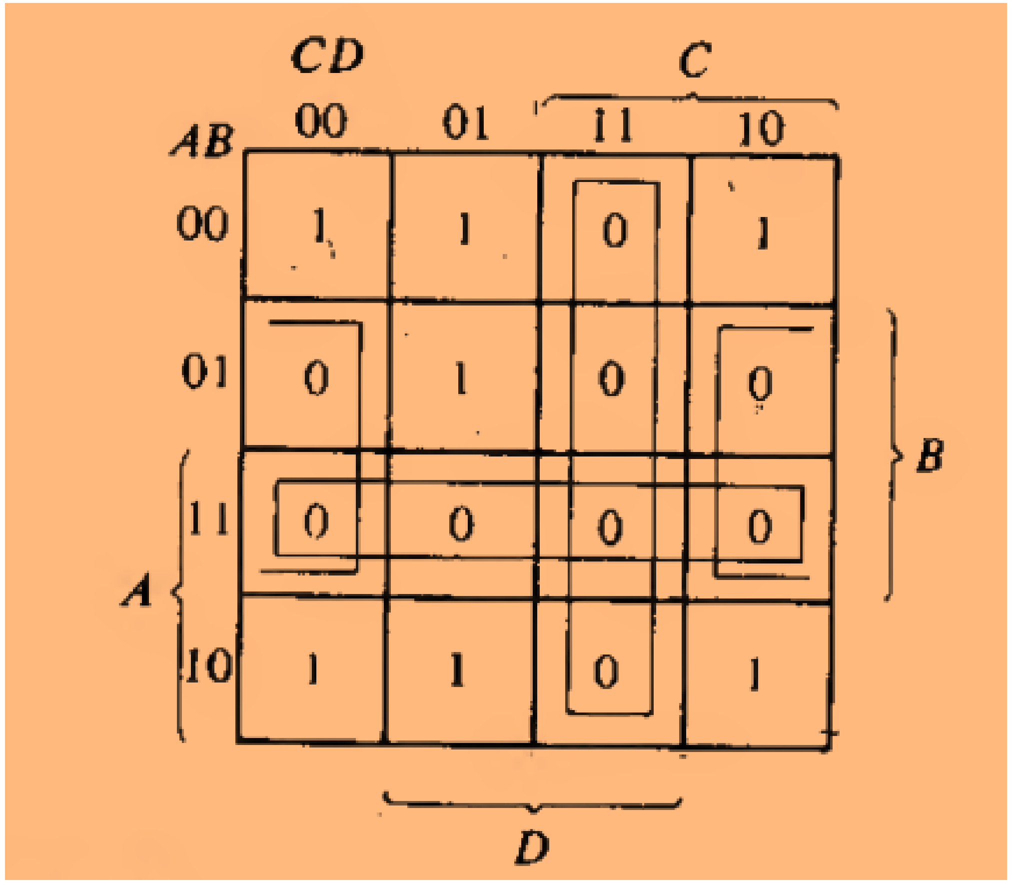
F = B'D' + B'C' + A'C'D

If the squares marked with 0's are combined, as shown in the diagram, we obtain the simplified complemented function:

F'=AB+CD+BD'

Simplify the following Boolean function in (a) sum of products and (b) product of sum

$F(A, B, C, D) = \sum(0, 1, 2, 5, 8, 9, 10)$



If the squares marked with 0's are combined, as shown in the diagram, we obtain the simplified complemented function:

$F' = AB + CD + BD'$

Applying DeMorgan's theorem, we obtain the simplified function in product of sums:

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

Don’t Care Conditions

- The logical sum of the minterms associated with a Boolean function specifies the conditions under which the function is equal to 1 .
- The function is equal to 0 for the rest of the minterms. This assumes that all the combinations of the values for the variables of the function are valid.
- In practice, there are some applications where the function is not specified for certain combinations of the variables.
- As an example, the four-bit binary code for the decimal digits has six combinations that are not used and consequently are considered as unspecified.
- In most applications, we simply don’t care what value is assumed by the function for the unspecified minterms.
- For this reason, it is customary to call the unspecified minterms of a function don t-care conditions. These don’t-care conditions can be used on a map to provide further simplification of the Boolean expression.

Don’t Care Conditions

Real circuits don’t always need to have an output defined for every possible input.

- For example, some calculator displays consist of 7-segment LEDs. These LEDs can display $2^7 - 1$ patterns, but only ten of them are useful.

If a circuit is designed so that a particular set of inputs can never happen, we call this set of inputs a *don’t care* condition.

They are very helpful to us in Kmap circuit simplification.

Example: Simplify the Boolean function $F(w, x, y, z) = \sum(1, 3, 7, 11, 15)$ that has the don't-care conditions $d(w, x, y, z) = \sum(0, 2, 5)$

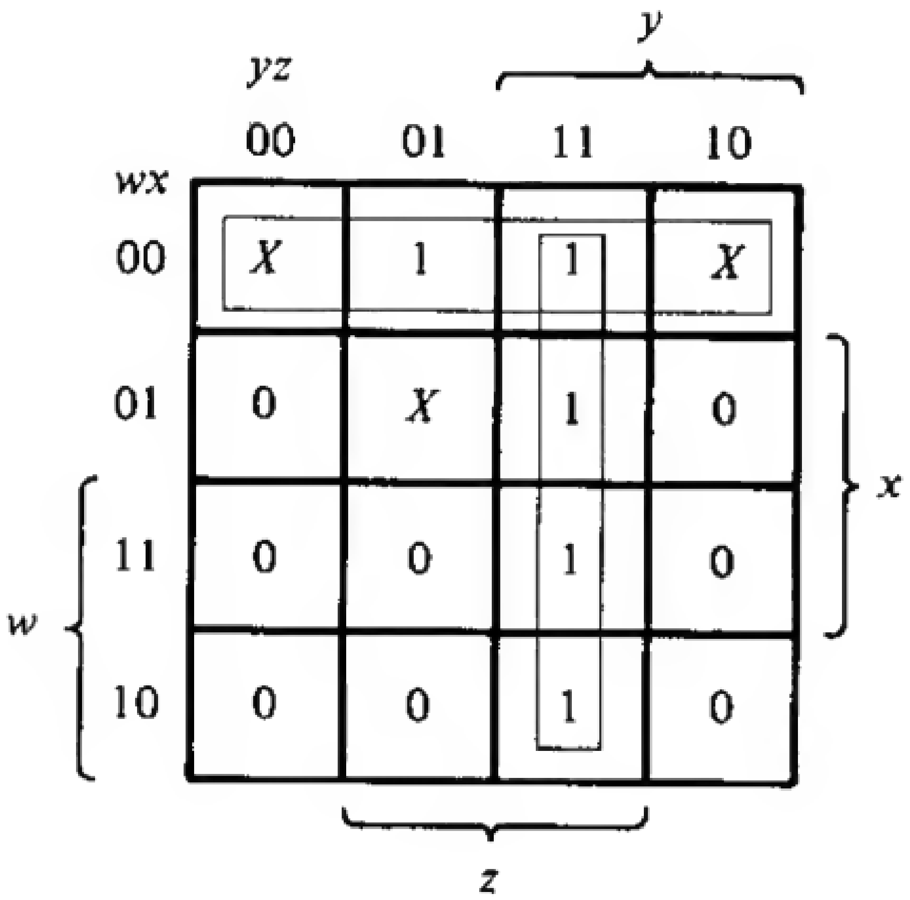
In a Kmap, a don't care condition is identified by an *X* in the cell of the minterm(s) for the don't care inputs, as shown below.

In performing the simplification, we are free to include or ignore the *X*'s when creating our groups.

wx \ yz	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

Don’t Care Conditions

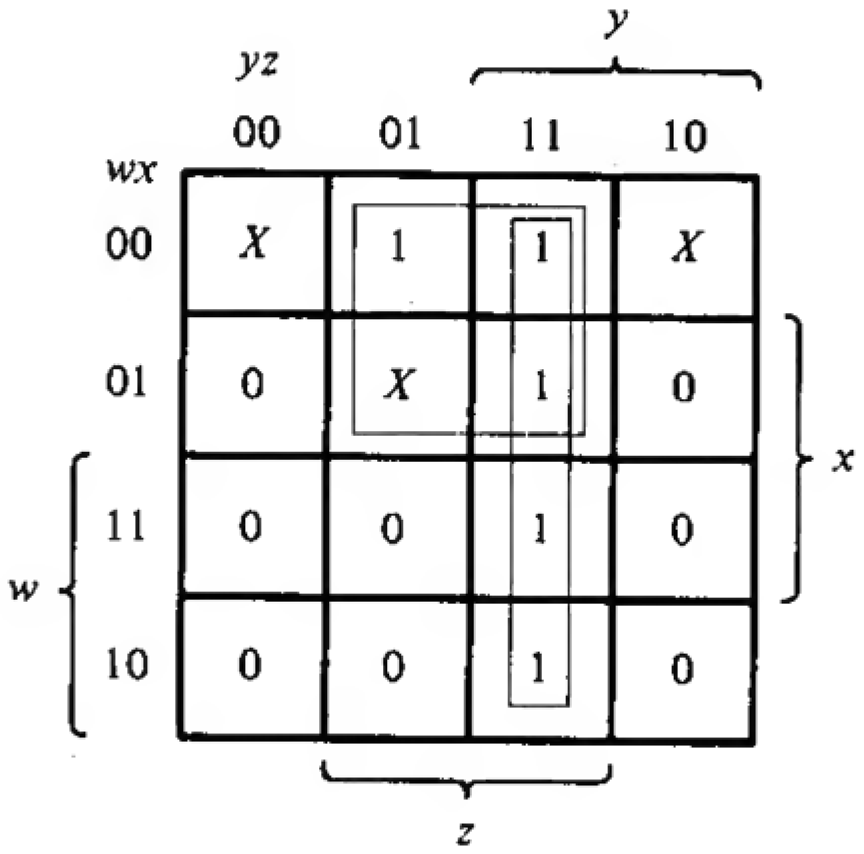
In one grouping in the Kmap below, we have the function:



$$F(W,X,Y,Z) = \overline{W}\,\overline{X} + YZ$$

Don’t Care Conditions

A different grouping gives us the function:



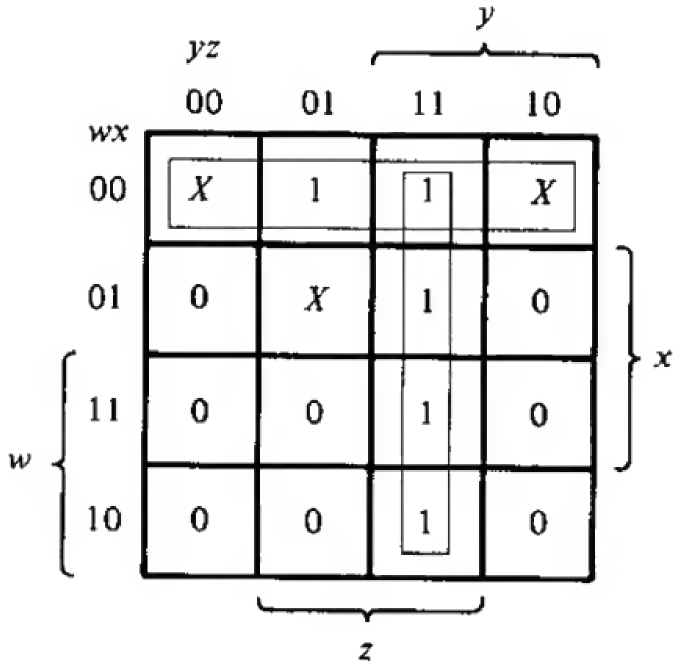
$$F(W,X,Y,Z) = \bar{W}Z + YZ$$

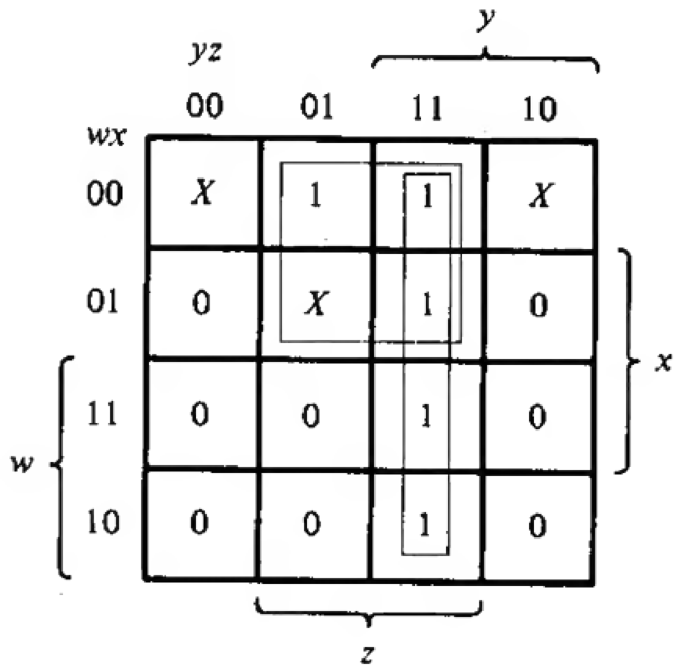
Don’t Care Conditions

The truth table of: $F(W,X,Y,Z) = \overline{W}\,\overline{X} + YZ$

differs from the truth table of: $F(W,X,Y,Z) = \overline{W}Z + YZ$

However, the values for which they differ, are the inputs for which we have don’t care conditions.





Suggested Reading

- M. Morris Mano, Digital Logic and Computer Design, PHI.

Thank you