K-Map (Karnaugh Map)

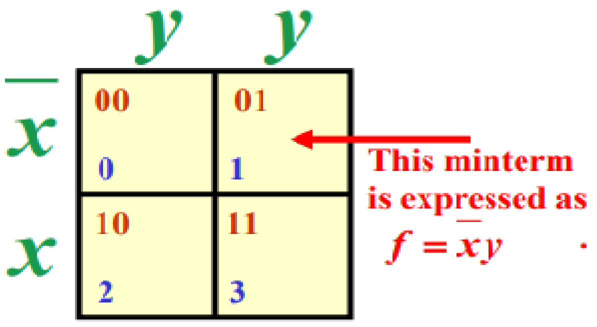
In many digital circuits and practical problems we need to find expression with minimum variables. We can minimize Boolean expressions of 3, 4 variables very easily using K-map without using any Boolean algebra theorems. K-map can take two forms Sum of Product (SOP) and Product of Sum (POS) according to the need of problem. K-map is table like representation but it gives more information than TRUTH TABLE. We fill grid of K-map with 0's and 1's then solve it by making groups.

- ➢ A K-map is a truth table graph, which aids in visually simplifying logic.
- ➢ It is useful for up to 5 or 6 variables, and is a good tool to help understand the process of logic simplification.
- ➢ The algebraic approach we have used previously is also used to analyze complex circuits in industry (computer analysis).

TWO VARIABLE K-MAP

• At the right is a 2-variable K-map.

• This very simple K-map demonstrates that an n-variable K-map contains all the combination of the n variables in the K- map space.



Two-Variable K-map, labeled for SOP terms. Note the four squares represent all the combinations of the two K-map variables, or minterms, in x & y (example above).

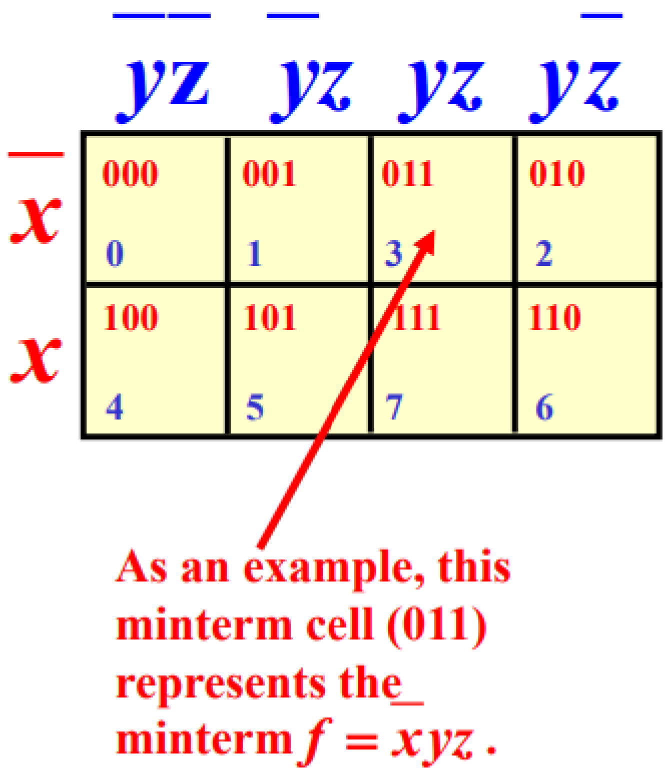
Three-Variable Karnaugh Map

• Each square represents a 3-variable minterm or maxterm.

• All of the 8 possible 3-variable terms are represented on the K-map.

• When moving horizontally or vertically, only 1 variable changes between adjacent squares, never 2. This property of the Kmap, is unique and accounts for its unusual numbering system.

• The K-map shown is one labeled for SOP terms. It could also be used for a POS problem, but we would have to re-label the variables



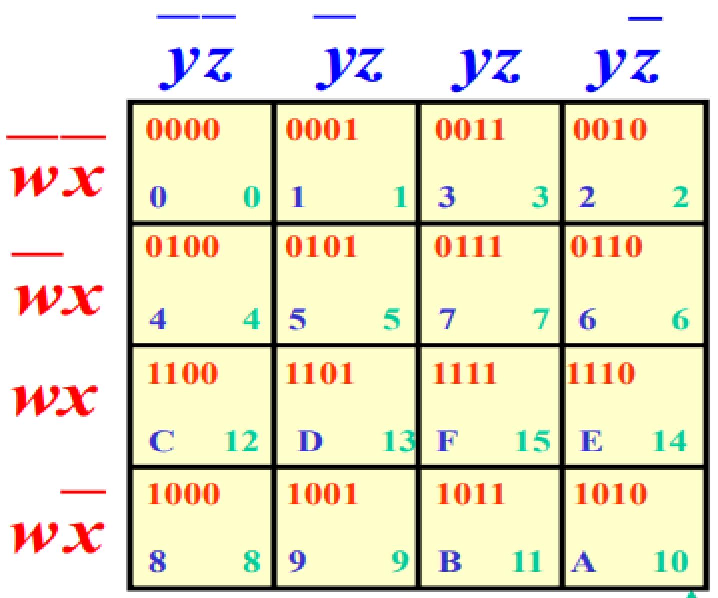
Four Variable Karnaugh Map

A 4-variable K-map can simplify problems of four Boolean variables.*

• The K-map has one square for each possible minterm (16 in this case).

• Migrating one square horizontally or vertically never results in more than one variable changing (square designations also shown in hex). Note that this is still an SOP K-map.

* Note that on all K-maps, the left and right edges are a common edge, while the top and bottom edges are also the same edge. Thus, the top and bottom rows are adjacent, as are the left and right columns.



	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{w}\bar{x}$	0000 0 0	0001 1 1	0011 3 3	0010 2 2
$\bar{w}x$	0100 4 4	0101 5 5	0111 7 7	0110 6 6
wx	1100 C 12	1101 D 13	1111 F 15	1110 E 14
$w\bar{x}$	1000 8 8	1001 9 9	1011 B 11	1010 A 10

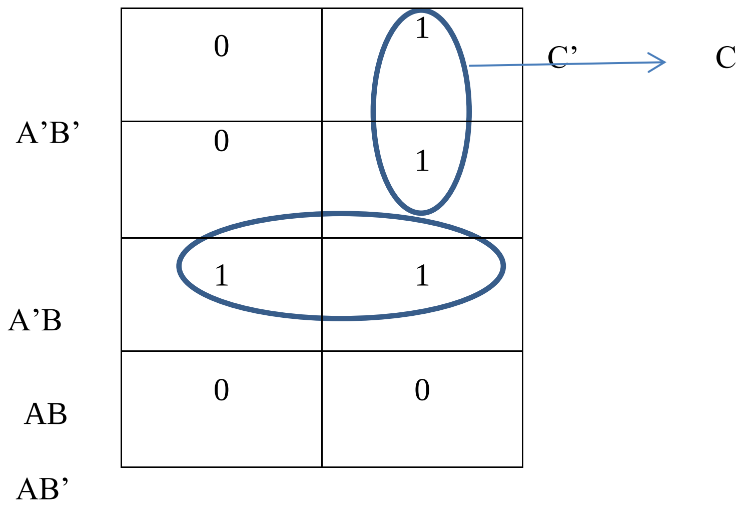
Note that this is still an SOP K-map.

<u>Steps to solve expression using K-map-</u>

1. Select K-map according to the number of variables.
2. Identify minterms or maxterms as given in problem.
3. For SOP put 1's in blocks of K-map respective to the minterms (0's elsewhere).
4. For POS put 0's in blocks of K-map respective to the maxterms(1's elsewhere).
5. Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.
6. From the groups made in step 5 find the product terms and sum them up for SOP form.

SOP FORM

1. **K-map of 3 variables-** Z= ∑A,B,C(1,3,6,7)

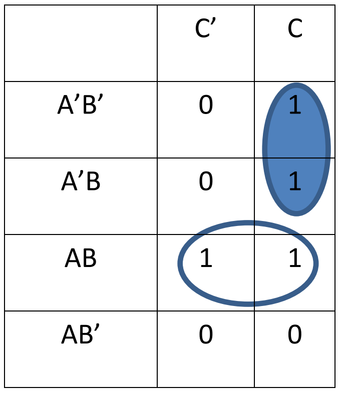


Final expression (A'C+AB)

From group we get product term— A'C and AB

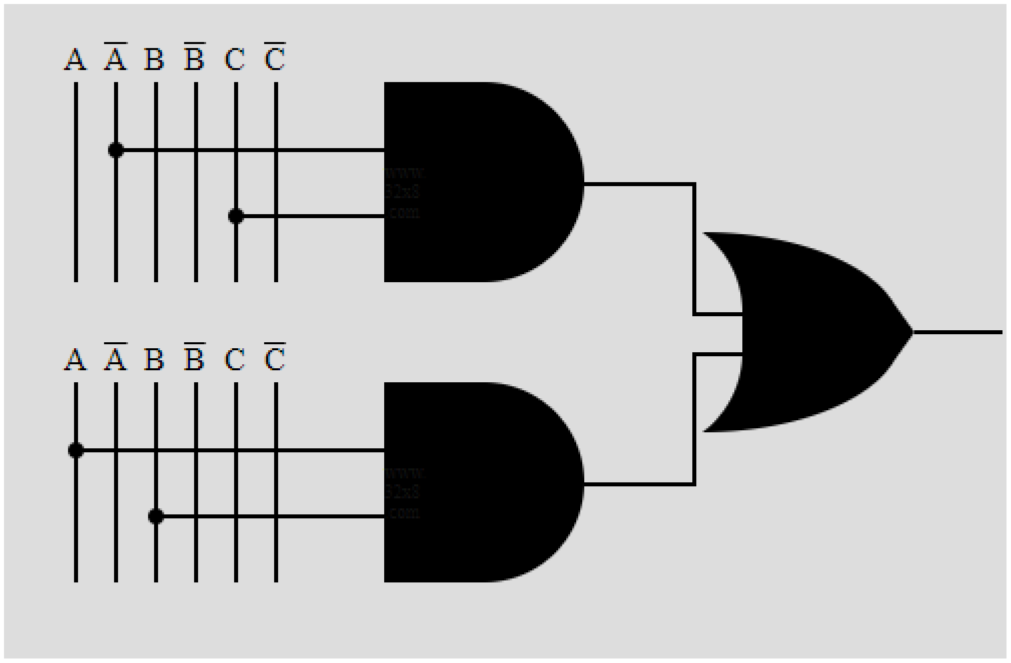
Summing these product terms we get- **Final expression (A'C+AB)**

SUM of PRODUCTS Map



	C'	C
A'B'	0	1
A'B	0	1
AB	1	1
AB'	0	0

y = A'C + AB



2. **K-map for 4 variables**

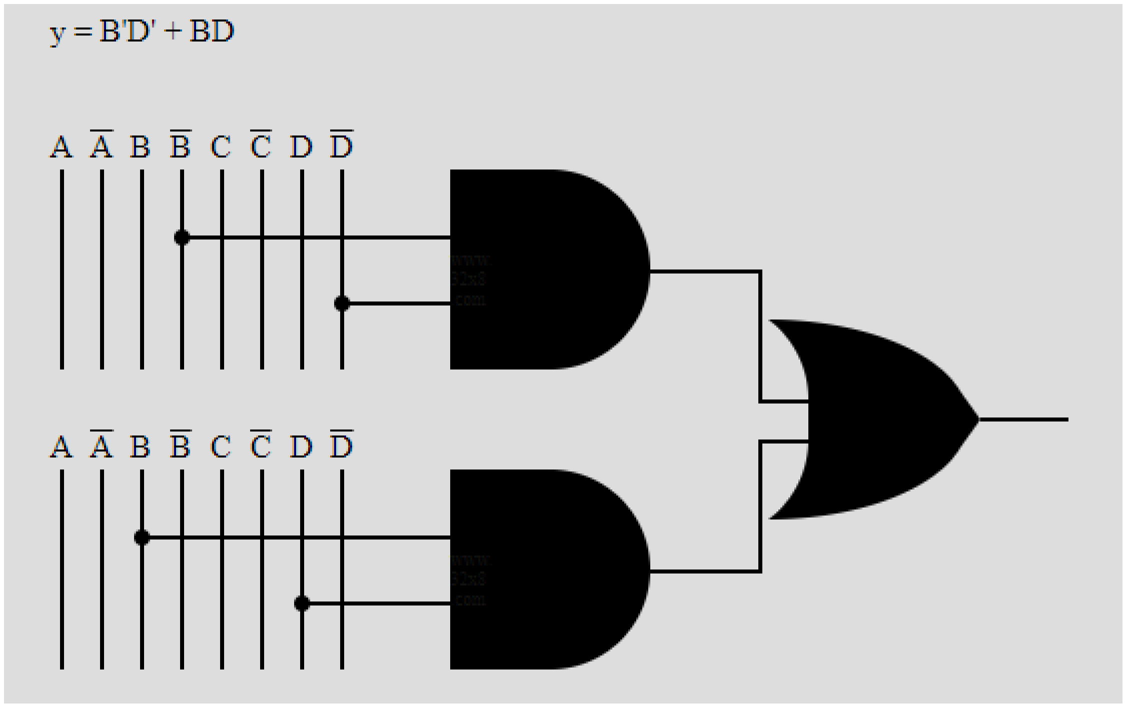
F(A,B,C,D)=∑(0,2,5,7,8,10,13,15)

Map

	C'D'	C'D	CD	CD'
A'B'	1	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	1	0	0	1

B'D'

BD


y = B'D' + BD
A $\overline{A}$ B $\overline{B}$ C $\overline{C}$ D $\overline{D}$
A $\overline{A}$ B $\overline{B}$ C $\overline{C}$ D $\overline{D}$

From **red** group we get product term— QS

From **green** group we get product term— Q’S’

Summing these product terms we get- **Final expression (QS+Q’S’)**

<u>POS FORM</u>

1. **<u>K-map of 3 variables-</u> F(A,B,C)=π(0,3,6,7)**

PRODUCT of SUMS

Map

	C’	C
A’B’	1	0
A’B	0	1
AB	1	1
AB’	0	0

Groups

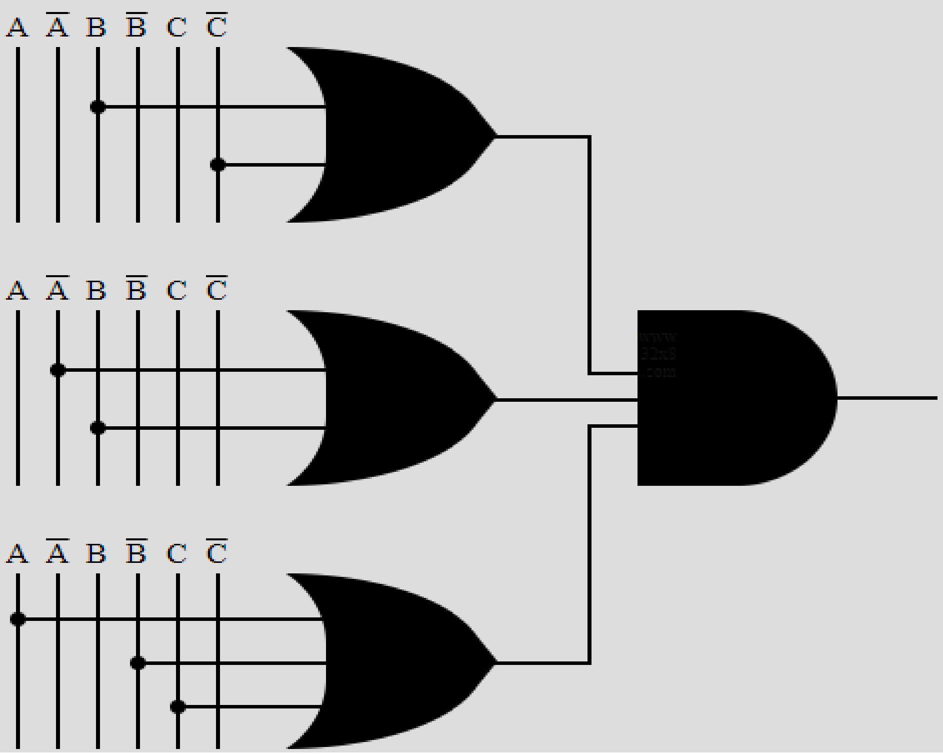
(1,5)	B’.C
(4,5)	A.B’
(2)	A’.B.C’

$$\overline{y} = \overline{B}.C + A.\overline{B} + \overline{A}.B.\overline{C}$$

$$\overline{\overline{y}} = \overline{\overline{B}.C + A.\overline{B} + \overline{A}.B.\overline{C}}$$

$$y = (B + C') (A' + B) (A + B' + C)$$

y = (B + C') (A' + B) (A + B' + C)



K-map of 4 variables- F(A,B,C,D)=π(3,5,7,8,10,11,12,13)

PRODUCT of SUMS

Map

	C’D’	C’D	CD	CD’
A’B’	0	0	1	0
A’B	0	1	1	0
AB	1	1	0	0
AB’	1	0	1	1

Map Layout

	C.D	C.D	C.D	C.D
A.B	0	1	3	2
A.B	4	5	7	6
A.B	12	13	15	14
A.B	8	9	11	10

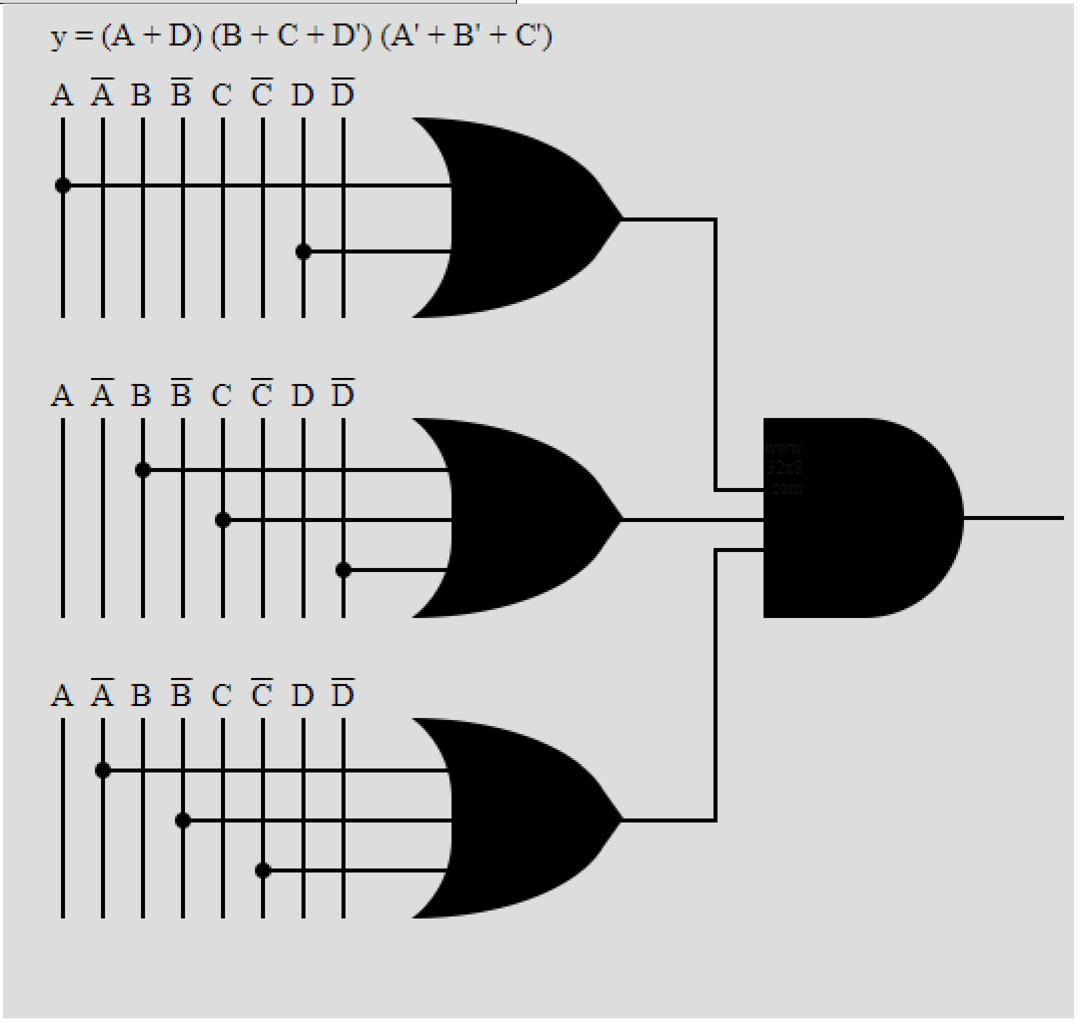
Groups

(0,2,4,6)	A.D
(1,9)	B.C.D
(14,15)	A.B.C

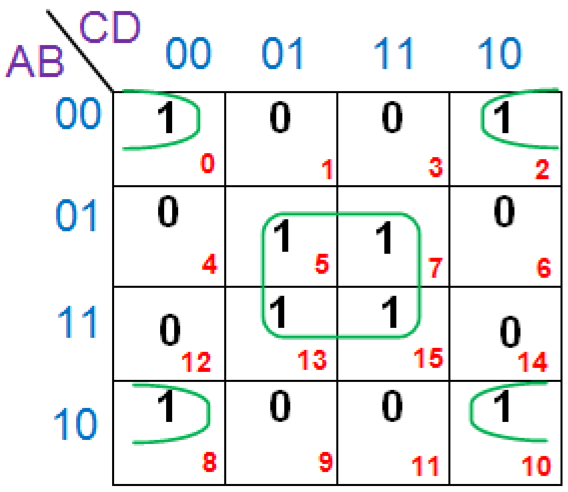
y = A’.D’ + B’.C’.D + A.B.C

$\overline{y} = \overline{A'.D' + B'.C'.D + A.B.C}$

$y = (A + D)\ (B + C + D')\ (A' + B' + C')$



$$Example\ 3: Y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\,\overline{D} + \overline{A}BC\,\overline{D} + \overline{A}BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C\overline{D} + ABC\overline{D} + ABCD$$



$$Simplified\ Expression: Y = BD + \overline{B}\,\overline{D}$$

Design a digital system whose output is defined as logically low if the 4-bit input binary number is a multiple of 3; otherwise, the output will be logically high. The output is defined if and only if the input binary number is greater than 2.

Step 1: Truth Table / Canonical Expression Leading to Min- or Max-Terms

The first step in designing any digital system is to have a clear idea of the variables involved in the process, along with their state-values. Further, depending on the problem statement, we have to arrive at the number of output variables and their values for each and every combination of the input literals, which can be conveniently represented in the form of a truth table.

In the given example:

Number of input variables = 4, which we will call A, B, C and D.

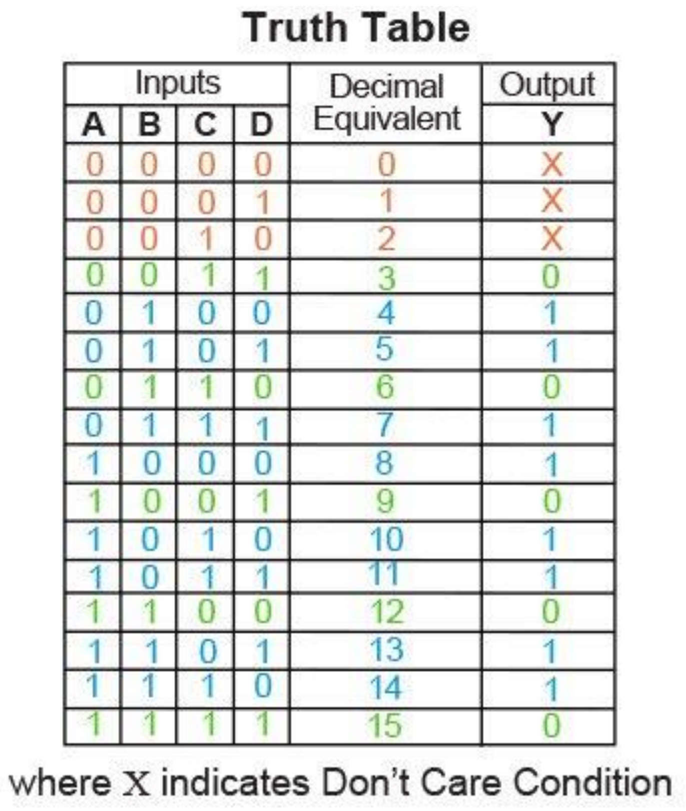
Number of output variables = 1, which we will call Y

where

Y = Don't Care, if the input number is less than 3 (orange entries in the truth table)

Y = 0, if the input number is an integral multiple of 3 (green entries in the truth table)

Y = 1, if the input number is not an integral multiple of 3 (blue entries in the truth table)



Truth Table

Inputs				Decimal Equivalent	Output
A	B	C	D		Y
0	0	0	0	0	X
0	0	0	1	1	X
0	0	1	0	2	X
0	0	1	1	3	0
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0	6	0
0	1	1	1	7	1
1	0	0	0	8	1
1	0	0	1	9	0
1	0	1	0	10	1
1	0	1	1	11	1
1	1	0	0	12	0
1	1	0	1	13	1
1	1	1	0	14	1
1	1	1	1	15	0

where X indicates Don't Care Condition

Table 1

Note that, in addition to the input and output columns, the truth table also has a column which gives the decimal equivalent of the input binary combination, which makes it easy for us to arrive at the minterm or maxterm expansion for the given problem. Thus for the given example,

Minterm expansion will be $\sum m(4,5,7,8,10,11,13,14) + \sum d\,(0,1,2)$

Maxterm expansion will be $\prod M(3,6,9,12,15) \cdot \prod D\,(0,1,2)$

However, sometimes the logical expression which is to be simplified might be directly given in terms of SOP or POS forms. In this case, the requirement for the truth table can be overlooked provided that we express the given expression in its canonical form, from which the corresponding minterms or maxterms can be obtained.

Step 2: Select and Populate K-Map

From Step 1, we know the number of input variables involved in the logical expression from which size of the K-map required will be decided. Further, we also know the number of such K-maps required to design the desired system as the number of output variables would also be known definitely. This means that, for the example considered, we require a single (due to one output variable) K-map with 16 cells (as there are four input variables).

Next, we have to fill the K-map cells with one for each minterm, zero for each maxterm, and X for Don't Care terms. The procedure is to be repeated for every single output variable. Hence for this example, we get the K-map as shown in Figure 2.

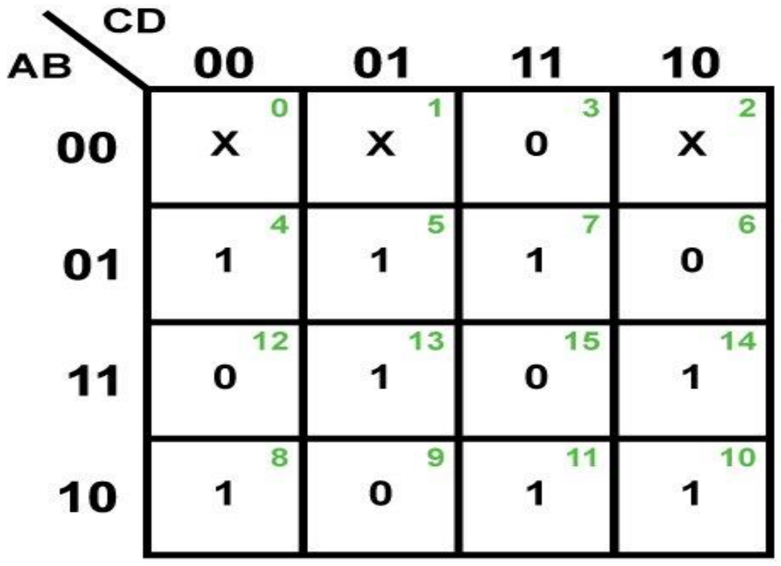


Figure 2: A completely filled 4-variable K-map

Step 3: Form the Groups

K-map simplification can also be referred to as the "simplification by grouping" technique as it solely relies on the formation of clusters. That is, the main aim of the entire process is to gather together as many ones (for SOP solution) or zeros (for POS solution) under one roof for each of the output variables in the problem stated. However, while doing so we have to strictly abide by certain rules and regulations:

- The process has to be initiated by grouping the bits which lie in adjacent cells such that the group formed contains the maximum number of selected bits. This means that for an *n*-variable K-map with 2^ncells, try to group for 2^n cells first, then for 2^{n-1} cells, next for 2^{n-2} cells, and so on until the “group” contains only 2^0 cells, i.e., isolated bits (if any). Note that the number of cells in the group must be equal to an integer power to 2, i.e., 1, 2, 4, 8. . . .
- The procedure must be applied for all adjacent cells of the K-map, even when they appear to be not adjacent—the top row is considered to be adjacent to the bottom row and the rightmost column is considered to be adjacent to the leftmost column, as if the K-map wraps around from top to bottom and right to left. For example, Group 1 of SOP form solution in Table 1.

- A bit appearing in one group can be repeated in another group provided that this leads to the increase in the resulting group-size. For example, cell 5 is repeated in both Group 3 as well as 4 in SOP form solution of Table 1 as it results in the formation of a group with two cells instead of a group with just one cell.
- Don’t Care conditions are to be considered for the grouping activity if and only if they help in obtaining a larger group. Otherwise, they are to be neglected. Here the Don't Care terms in the cells 0 and 1 are considered to create Group 2 of SOP solution form as it results in a group with four cells instead of just two.

	SOP Form Solution			**POS Form Solution**	
Number of groups having 16 cells	0			0	
Number of groups having 8 cells	0			0	
Number of groups having 4 cells (Blue Enclosures in Figure 3)	2	Group 1 (Cells 0,2,8,10)		1	Group 1 (Cells 0,1,2,3)
		Group 2 (Cells 0,1,4,5)			
Number of groups having 2 cells (Orange Enclosures in Figure 3)	4	Group 3 (Cells 5,7)	Group 4 (Cells 5,13)	2	Group 2 (Cells 1,9)
		Group 5 (Cells 10,11)	Group 6 (Cells 10,14)		Group 3 (Cells 2,6)
Number of groups having 1cell (Green Enclosures in Figure 3)	0			2	Group 4 (Cell 12)
					Group 5 (Cell 15)

Table 1

Hence for the example considered, the K-map showing the groups can be obtained as given by Figure 3 whose information is summarized in Table 1.

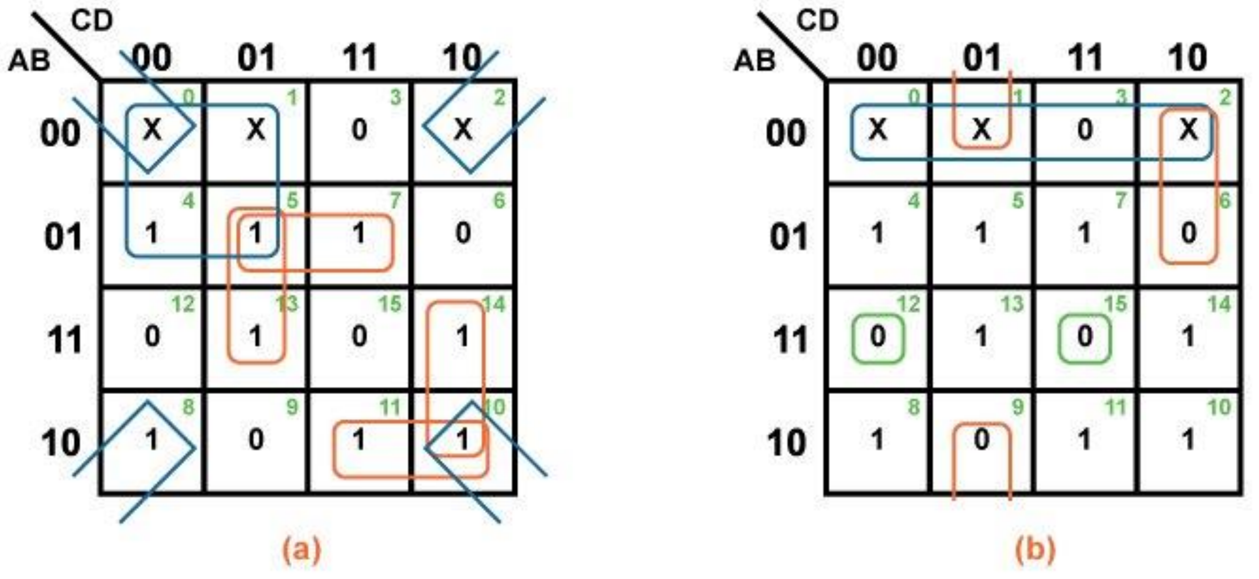


Figure 3: K-maps grouped for (a) SOP solution and (b) POS solution

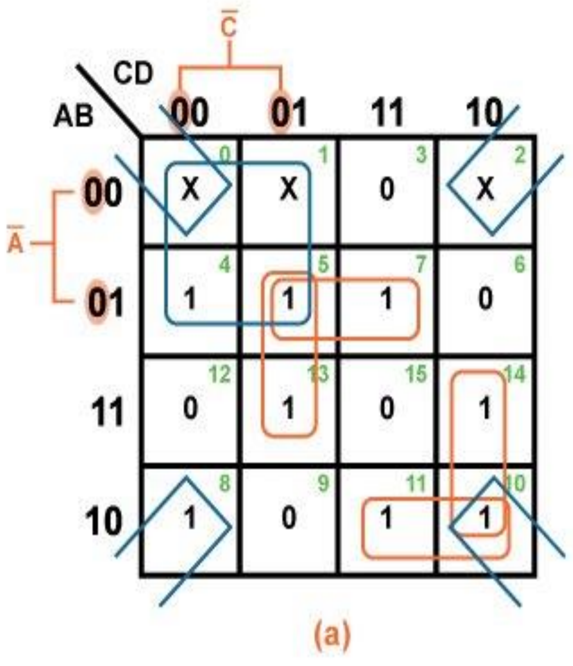
Step 4: Simplified Logical Expression

For each of the resulting groups, we have to obtain the corresponding logical expression in terms of the input-variables. This can be done by expressing the bits which are common amongst the Gray code-words which represent the cells contained within the considered group. Another way to describe the process of obtaining the simplified logical expression for a group is to eliminate the variable(s) for which the corresponding bits appear within the group as both 0 and 1.

Finally, all these group-wise logical expressions need to be combined appropriately to form the simplified Boolean equation for the output variable. The same procedure must be repeated for every output variable of the given problem.

For instance, in the example considered, the logical term for Group 2 of SOP form solution is obtained as $\overline{A}\overline{C}$. This is due to the fact that this group has 0 as the common Gray code-word bit both along its rows as well as its columns, highlighted in Figure 4(a). This gives us the literals $\overline{A}$ and $\overline{C}$.

Similarly, in the case of Group 1 of POS form solution, we can obtain the logical expression as A+B. This is because the group has the common Gray code-words as 0,0 along its row only (no code-word bits are common along its columns) which correspond to the input variables A and B.



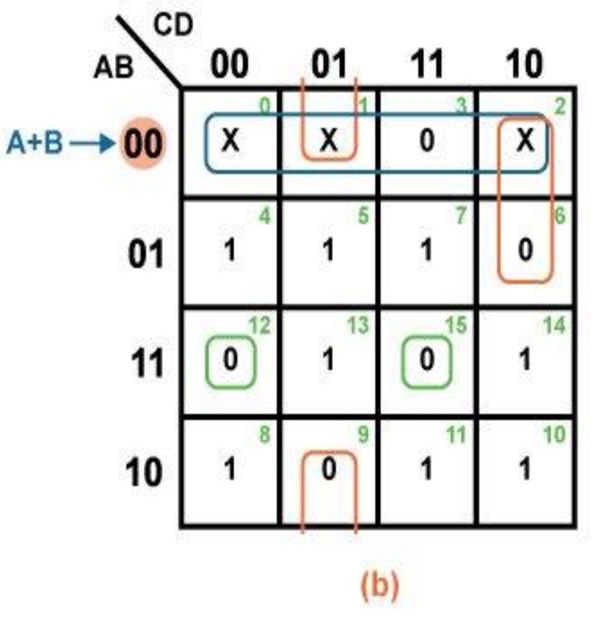


Figure 4: K-map simplification technique for (a) SOP solution and (b) POS solution

Following this same process, we can obtain the logical terms corresponding to each of the groups to finally form the logical expression for the particular output, as shown by Table 2.

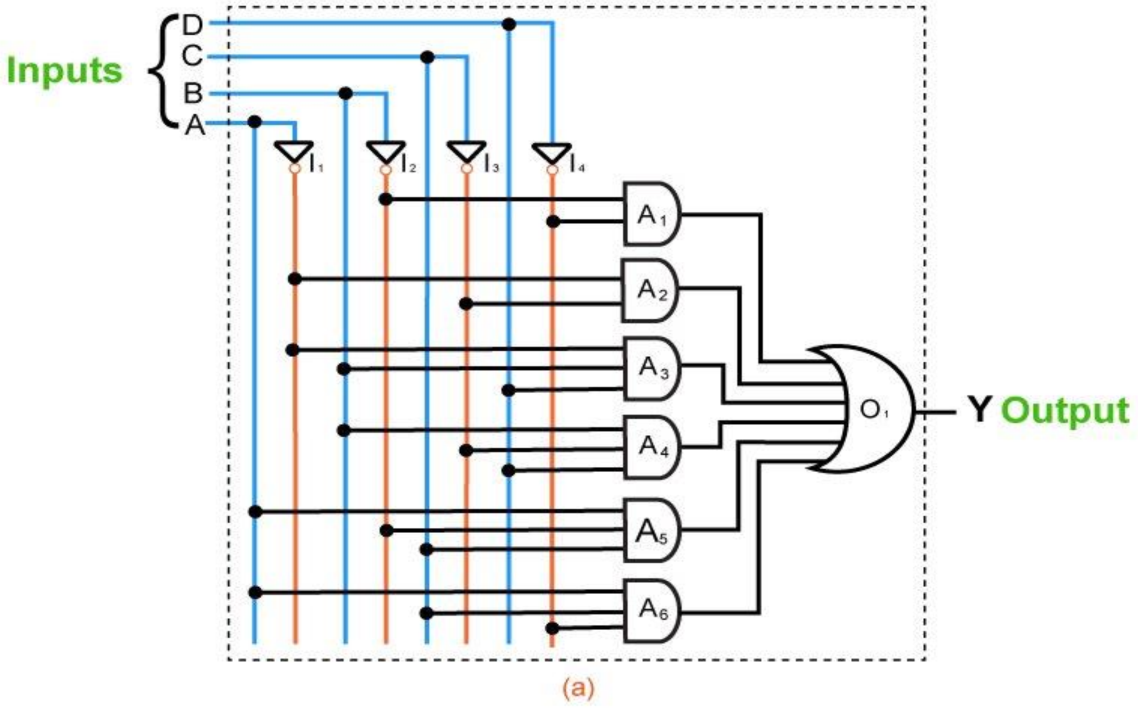
SOP Form Solution		POS Form Solution	
Groups	**Logical Expression**	**Groups**	**Logical Expression**
Group 1	$\bar{B}\bar{D}$	Group 1	A+B
Group 2	$\bar{A}\bar{C}$	Group 2	$B+C+\bar{D}$
Group 3	$\bar{A}BD$	Group 3	$A+\bar{C}+D$
Group 4	$B\bar{C}D$	Group 4	$\bar{A}+\bar{B}+C+D$
Group 5	$A\bar{B}C$	Group 5	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$
Group 6	$AC\bar{D}$		
Thus, $Y = \bar{B}\bar{D} + \bar{A}\bar{C} + \bar{A}BD + B\bar{C}D + A\bar{B}C + AC\bar{D}$		**Thus, $Y = (A+B)(B+C+\bar{D})(A+\bar{C}+D)(\bar{A}+\bar{B}+C+D)(\bar{A}+\bar{B}+\bar{C}+\bar{D})$**	

Table 2

Step 5: System Design

Having obtained the simplified logical expression, we can decide on the type and the number of gates required to realize the expected logic for every output bit, which further results in the complete design of the desired system.

Thus, the digital system corresponding to SOP and POS forms of solution for the given example can be designed using the basic gates like NOT, AND, and OR as shown by Figure 5(a) and 5(b).



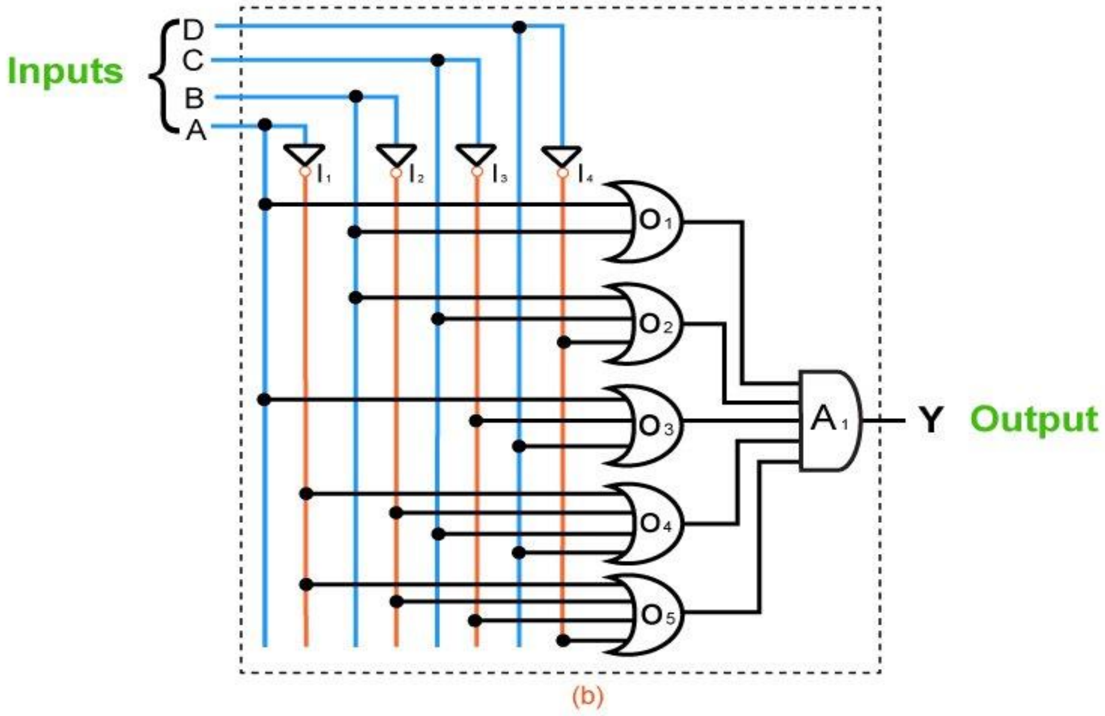
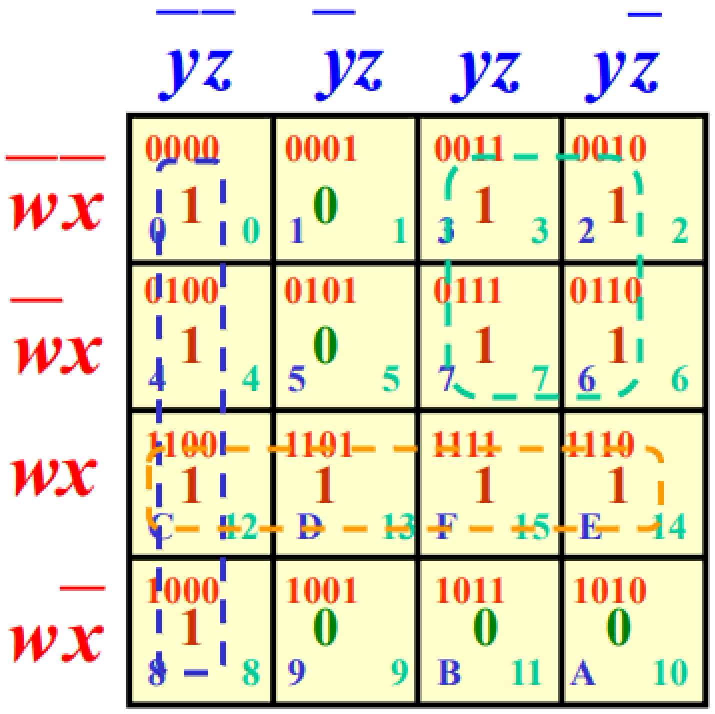


Figure 5: Digital system corresponding to (a) SOP form of solution and (b) POS form of solution

What is a prime implicant?

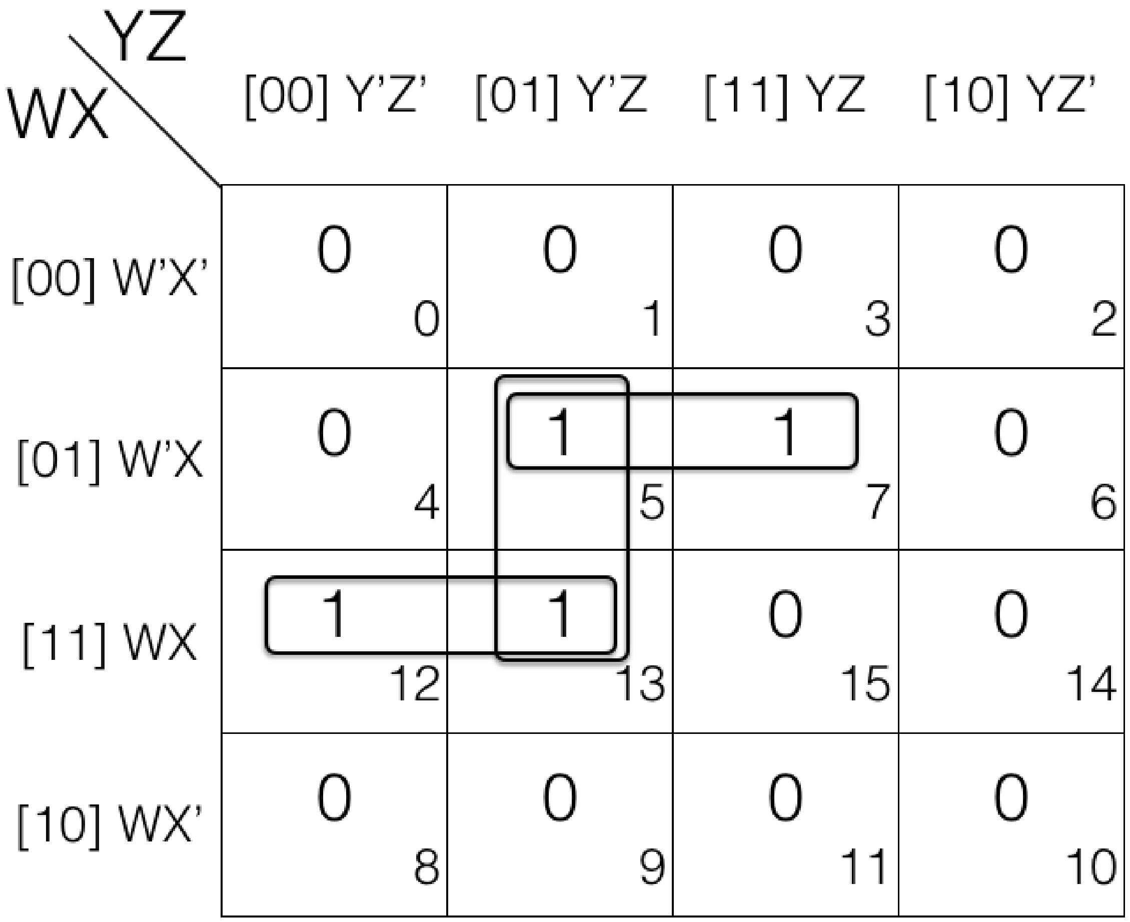
It is an implicant that covers as many 1 values (SOP K-map) or 0 values (POS K-map) as possible, yet still retains the identity of implicant (# of cells = power of 2, rectangular or square shape). Some SOP prime implicants are shown on the adjoining K-map.



Redundant Groups

After marking out the overlapping groups it is important to also check for redundant groups. If all the values of a group G (pair, quad or octet) is covered (overlapping) with other groups then that group G is redundant and ignored. Lets check an example.

Consider the following 4 variables K-map.



1st pair = W'XY'Z + W'XYZ = $m_5 + m_7$

2nd pair = WXY'Z' + WXY'Z = $m_{12} + m_{13}$

3rd pair = W'XY'Z + WXY'Z = $m_5 + m_{13}$

If we look at m_5 and m_{13} i.e. 3rd pair, it is a redundant group as m_5 and m_{13} is covered in 1st and 2nd **pair so we will remove the redundant group (3rd pair).**

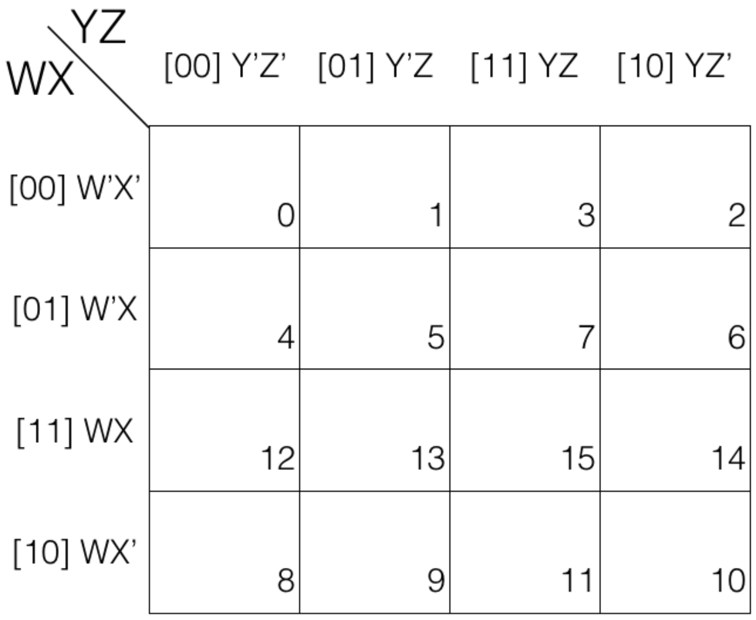
Updated pairs after reduction

1st pair = W'XY'Z + W'XYZ
= W'XZ

2nd pair = WXY'Z' + WXY'Z
= WXY'

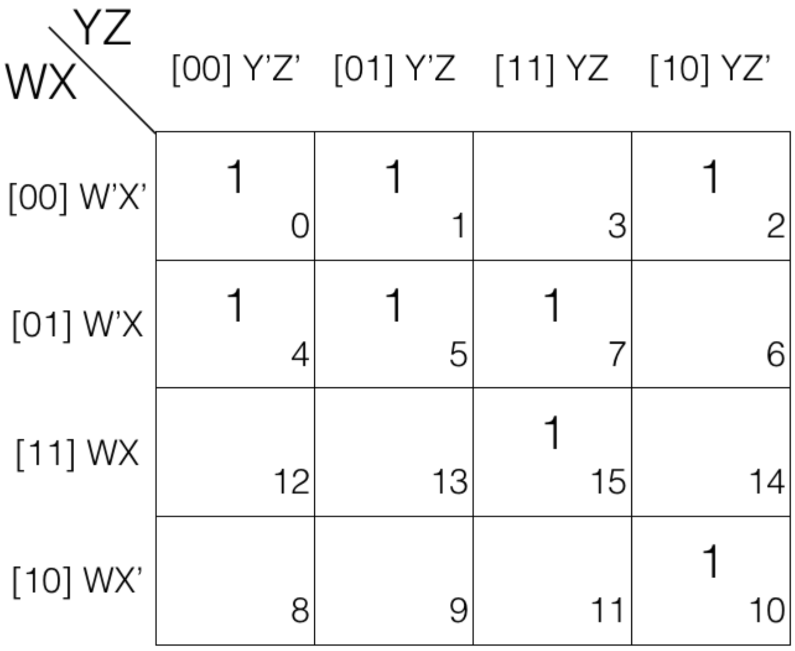
1. Reduce F(A,B,C,D) = ∑(0,1,2,4,5,7,10,15) using K-map

Since function F has 4 variables so we will create a 4 variable K-map having $2^4 = 16$ cells.



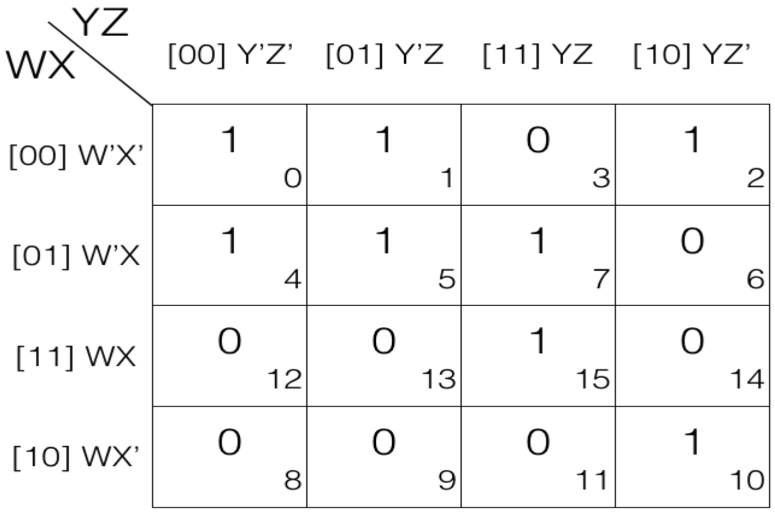
WX \ YZ	[00] Y'Z'	[01] Y'Z	[11] YZ	[10] YZ'
[00] W'X'	0	1	3	2
[01] W'X	4	5	7	6
[11] WX	12	13	15	14
[10] WX'	8	9	11	10

Now fill the cell marked with subscript 0,1,2,4,5,7,10 and 15 with value 1 as we are dealing with Sum of Products SOP.



WX \ YZ	[00] Y'Z'	[01] Y'Z	[11] YZ	[10] YZ'
[00] W'X'	1 $_{0}$	1 $_{1}$	$_{3}$	1 $_{2}$
[01] W'X	1 $_{4}$	1 $_{5}$	1 $_{7}$	$_{6}$
[11] WX	$_{12}$	$_{13}$	1 $_{15}$	$_{14}$
[10] WX'	$_{8}$	$_{9}$	$_{11}$	1 $_{10}$

And fill rest of the cells with value 0

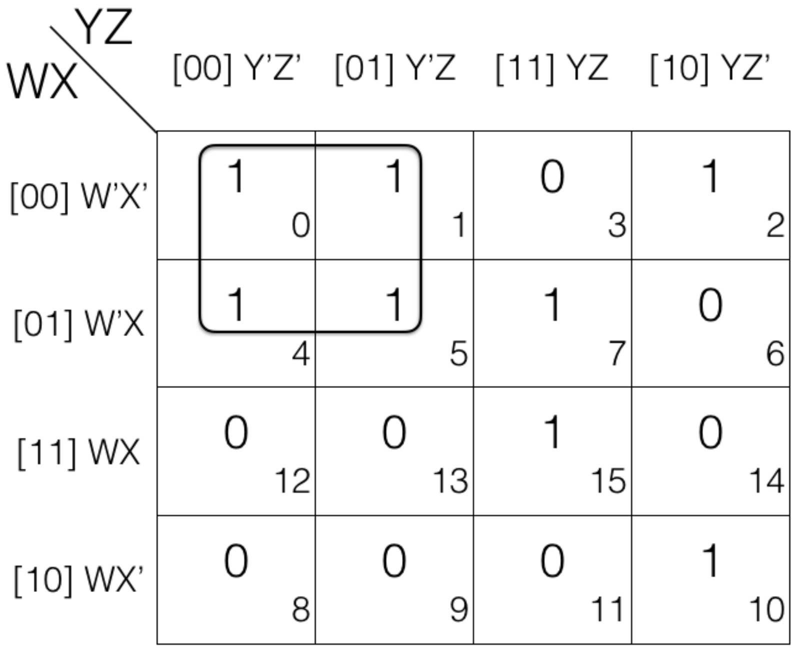


WX \ YZ	[00] Y'Z'	[01] Y'Z	[11] YZ	[10] YZ'
[00] W'X'	1 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
[01] W'X	1 $_{4}$	1 $_{5}$	1 $_{7}$	0 $_{6}$
[11] WX	0 $_{12}$	0 $_{13}$	1 $_{15}$	0 $_{14}$
[10] WX'	0 $_{8}$	0 $_{9}$	0 $_{11}$	1 $_{10}$

Now we will mark the octets, quads and pairs.

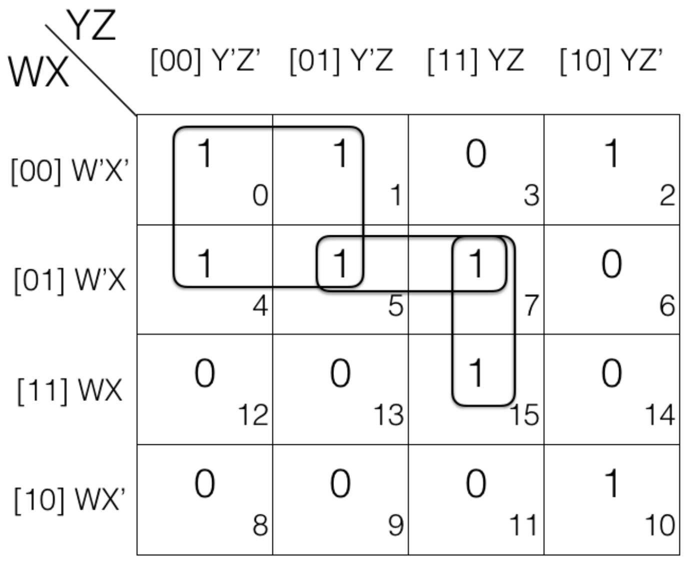
Looking at the K-map we can tell that there is no octets so we will look for quads.

The k-map has quad so we will mark it.



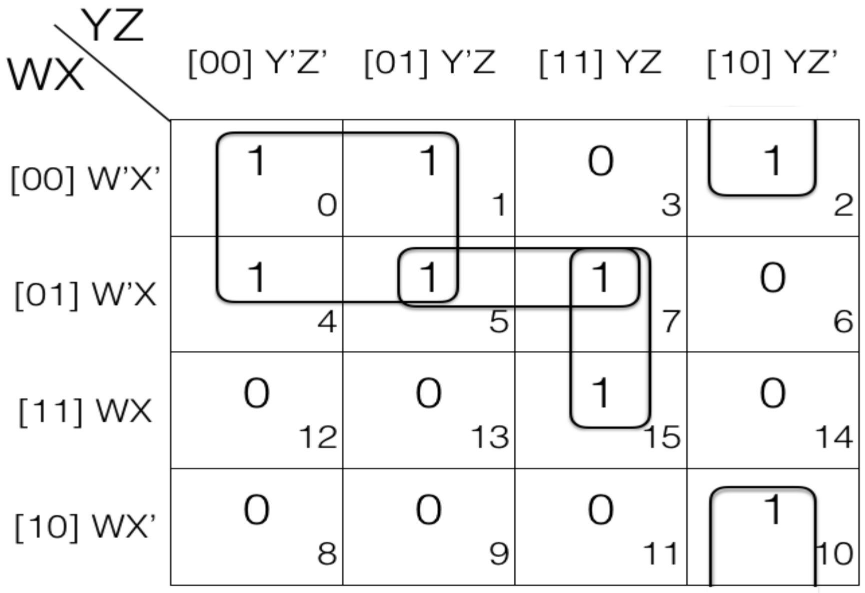
WX \ YZ	[00] Y'Z'	[01] Y'Z	[11] YZ	[10] YZ'
[00] W'X'	1 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
[01] W'X	1 $_{4}$	1 $_{5}$	1 $_{7}$	0 $_{6}$
[11] WX	0 $_{12}$	0 $_{13}$	1 $_{15}$	0 $_{14}$
[10] WX'	0 $_{8}$	0 $_{9}$	0 $_{11}$	1 $_{10}$

Now we will look for pairs. If we look close we can tell that the k-map has some pairs. So we will mark them.



Now we will map-roll and look for octets. If we look at the k-map we can tell that after map rolling we don't get any octet. So we will now check for quads.
We roll the k-map again and this time we are looking for quads. But giving a closer look we can tell that there is no quad after we do map rolling. So time for us to look for pairs.

This time we are lucky. Looking closer we can tell that after map rolling the k-map we have a pair. So we will mark it.



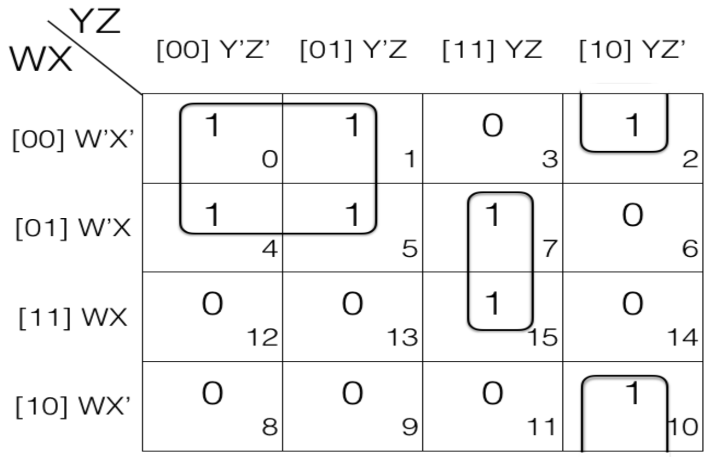
Now we will look for overlapping groups.

If we look at the K-map we will see that m_7 in pair (m_5, m_7) and pair (m_7, m_{15}) is shared in both the groups. So both the pairs are overlapping groups.

Similarly m_5 in pair (m_5, m_7) and quad (m_0, m_1, m_4, m_5) is shared in both the groups. So the pair and quad are overlapping groups.

And there is no other overlapping groups so, we will now check for redundant groups.

Looking at the K-map we can tell pair (m_5, m_7) is redundant as m_5 is covered in quad (m_0, m_1, m_4, m_5) and m_7 is covered in pair (m_7, m_{15}). So we will remove the pair (m_5, m_7).



Now we will write down the marked groups and find the reduced expression.

quad = (m_0, m_1, m_4, m_5) = (W'X'Y'Z' + W'X'Y'Z) + (W'XY'Z' + W'XY'Z)

= W'X'Y' + W'XY' [Z' changed, so removed]

= W'Y' [X' changed to X, so removed]

1st pair = (m_7, m_{15})

= W'XYZ + WXYZ

= XYZ [W' changed to W, so removed]

2nd pair = (m_2, m_{10})

= W'X'YZ' + WX'YZ'

= X'YZ' [W' changed to W, so removed]

Now we OR (+) the results to get the final reduced expression.

F = W'Y' + XYZ + X'YZ'

This is the required answer.

5 VARIABLE KMAP

Simplify the Boolean expression *f* (A,B,C,D,E) = ∑*m* (0,3,4,7,8,12,14,16,19,20,23,24,26,28)

Step 1:

Number of input variables = 5

Number of output variables = 1

Minterm expansion of the output is given as *f* (A,B,C,D,E) = ∑*m* (0,3,4,7,8,12,14,16,19,20,23,24,26,28)

Steps 2, 3, and 4:

Number of K-maps required = 1

Each K-map should have 32 cells in it.

Thus we get:

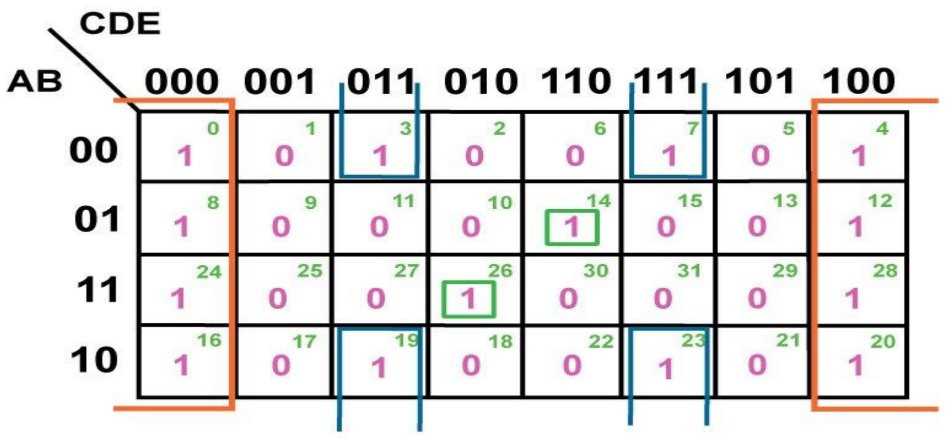


Figure 8: Grouped 32-cell K-map

Number of Groups with 32 cells	Nil		
Number of Groups with 16 cells	Nil		
Number of Groups with 8 cells (Orange Enclosures in Figure 8)	1	Group 1 (Cells 0,4,8,12,16,20,24,28)	$\bar{D}\bar{E}$
Number of Groups with 4 cells (Blue Enclosures in Figure 8)	1	Group 2 (Cells 3,7,19,23)	$\bar{B}DE$
Number of Groups with 2 cells	Nil		
Number of Groups with 1 cell (Green Enclosures in Figure 8)	2	Group 3 (Cell 14)	$\bar{A}BCD\bar{E}$
		Group 4 (Cell 26)	$AB\bar{C}D\bar{E}$
Thus, *f* (A,B,C,D,E) = $\bar{D}\bar{E} + \bar{B}DE + \bar{A}BCD\bar{E} + AB\bar{C}D\bar{E}$			

Table 6

Step 5:

The digital system corresponding to the function given is obtained as shown in Figure 9:

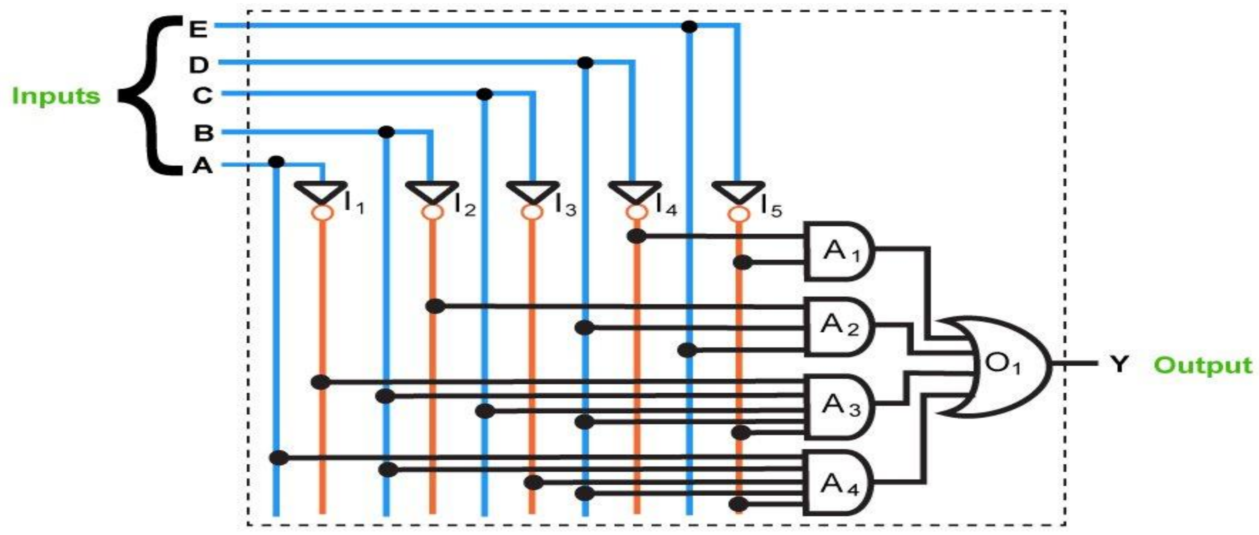


Figure 9: 5-variable K-map for the function shown in Example 3

Advantages of K-Maps

1. The K-map simplification technique is simpler and less error-prone compared to the method of solving the logical expressions using Boolean laws.
2. It prevents the need to remember each and every Boolean algebraic theorem.
3. It involves fewer steps than the algebraic minimization technique to arrive at a simplified expression.
4. K-map simplification technique always results in minimum expression if carried out properly.

Disadvantages of K-Maps

1. As the number of variables in the logical expression increases, the K-map simplification process becomes complicated.
2. The minimum logical expression arrived at by using the K-map simplification procedure may or may not be unique depending on the choices made while forming the groups. For example, for the output variable Y shown by the K-map in Figure 10, we can obtain two different, but accurate logical expressions. The variation in the solution obtained is observed in the third term, which may be either $\overline{B}\overline{C}$ or $A\overline{C}$ (highlighted in Figure 10). This difference depends on whether one chooses to group the cells (0,4) or (4,6) to form a two-celled group in order to cover the one found in the K-map cell numbered 4.