Optimization of Digital Circuits using K Map

Lecture-11 & 12

By

Bhagyalaxmi Behera

Asst. Professor (Dept. of ECE)

INTRODUCTION

- *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.

- This lecture serves as a foundation for your understanding of that important topic and will enable you to execute a manual design of simple circuits, preparing you for skilled use of modern design tools.

Karnaugh map or K-map

- The complexity of the digital logic gates that implement a Boolean function is directly related to the complexity of the algebraic expression from which the function is implemented.
- Although the truth table representation of a function is unique, when it is expressed algebraically it can appear in many different, but equivalent, forms.
- Boolean expressions may be simplified by algebraic means but this procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process.
- The map method presented here provides a **simple, straightforward procedure** for minimizing Boolean functions.
- This method may be regarded as a pictorial form of a truth table. The map method is also known as the ***Karnaugh map or K-map*** .

Karnaugh map or K-map

- 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.

- 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.

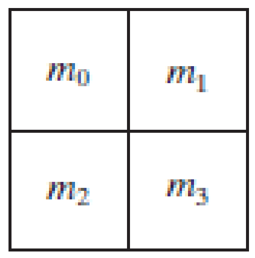
Karnaugh map or K-map

- The simplified expressions produced by the map are always in one of the two standard forms: **sum of products** or **product of sums.**

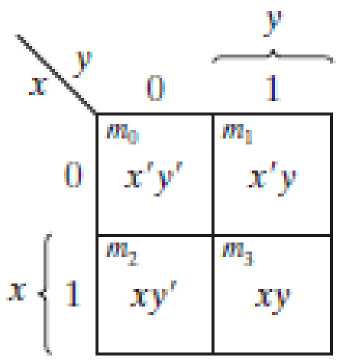
- It will be assumed that the simplest algebraic expression is an algebraic expression with a minimum number of terms and with the smallest possible number of literals in each term.

- This expression produces a circuit diagram with a **minimum number of gates** and the **minimum number of inputs to each gate.**

Two-Variable K-Map



(a)



(b)

- There are four minterms for two variables; hence, the map consists of four squares, one for each minterm.

- The map drawn in figure shows 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.

- Variable *x* appears primed in row 0 and unprimed in row 1. Similarly, *y appears primed in column* 0 and unprimed in column 1.

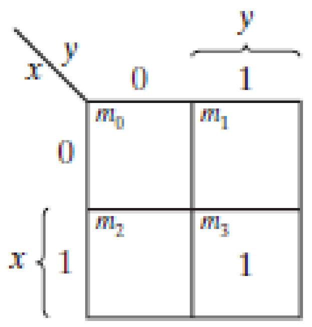
Representation of functions in the map

- If we mark the squares whose minterms belong to a given function, the two-variable map becomes another useful way to represent any one of the 16 Boolean functions of two variables.

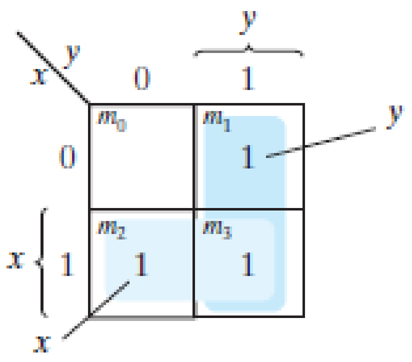
- As an example, the function *xy is shown in Fig. (a). Since xy is equal to m3, a 1 is placed inside the square that belongs to m3.*

- *Similarly, the function x + y is* represented in the map of Fig. (b) by three squares marked with 1's. These squares are found from the minterms of the function:

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



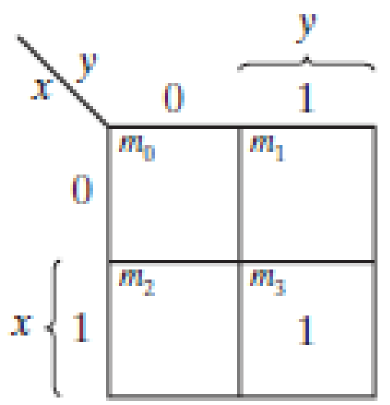
(a) xy



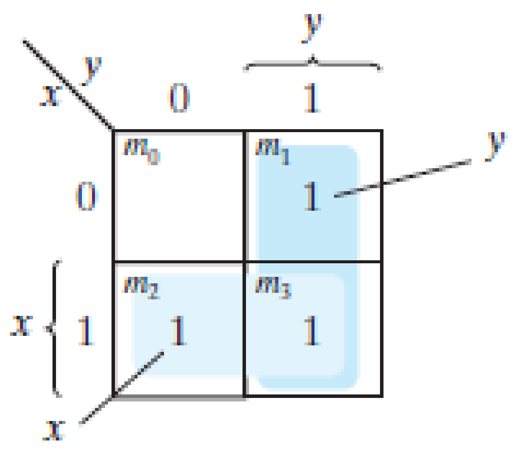
(b) $x + y$

Two-Variable K-Map

- F=X’Y+XY’+XY=Y(X’+X)+XY’=Y+XY’=(X+Y)(Y+Y’)= X+Y

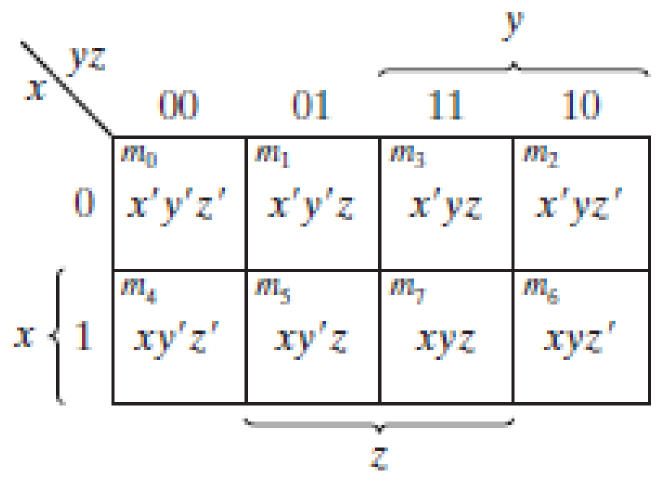


(a) xy



(b) $x + y$

Three-Variable K-Map



- There are eight minterms for three binary variables; therefore, the map consists of eight squares.
- Note that the minterms are arranged, not in a binary sequence, but in a sequence similar to the Gray code.
- The characteristic of this sequence is that **only one bit changes in value from one adjacent column to the next.**
- The map drawn is marked with numbers in each row and each column to show the relationship between the squares and the three variables.

Three-Variable K-Map

- For example, the square assigned to *m5 corresponds to row 1 and column 01.*

- *When these two* numbers are concatenated, they give the binary number 101, whose decimal equivalent is 5. Each cell of the map corresponds to a unique minterm, so another way of looking at square *m_5 = xy'z is to consider it to be in the row marked x and the column belonging* to *y'z (column 01).*

- *Note that there are four squares in which each variable is equal to 1* and four in which each is equal to 0.

- The variable appears unprimed in the former four squares and primed in the latter. For convenience, we write the variable with its letter symbol under the four squares in which it is unprimed.

Three-Variable K-Map

- The number of adjacent squares that may be combined must always represent a number that is a power of two, such as 1, 2, 4, and 8. As more adjacent squares are combined, we obtain a product term with fewer literals.

- **One square represents one minterm, giving a term with three literals.**

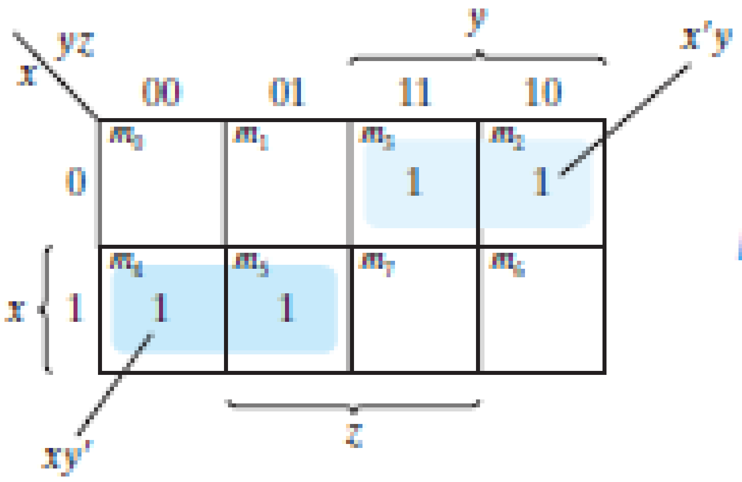
- **Two adjacent squares represent a term with two literals.**

- **Four adjacent squares represent a term with one literal.**

- **Eight adjacent squares encompass the entire map and produce a function that is always equal to 1.**

Simplify the Boolean function

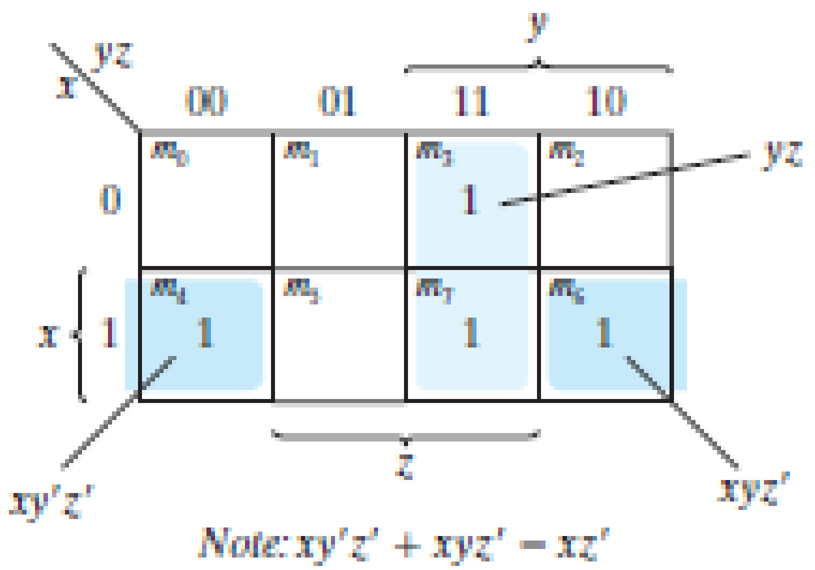
$$F(x, y, z) = \Sigma(2, 3, 4, 5)$$



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

Simplify the Boolean function

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

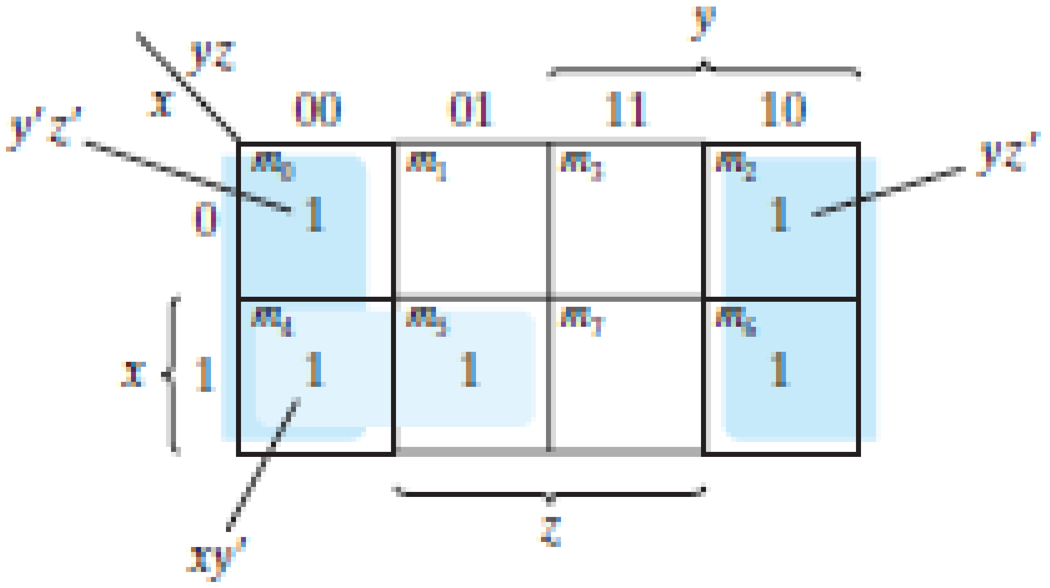


$$F = yz + xz'$$

Simplify the Boolean function

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

$$F = z' + xy'$$



Note: $y'z' + yz' = z'$

For the Boolean function

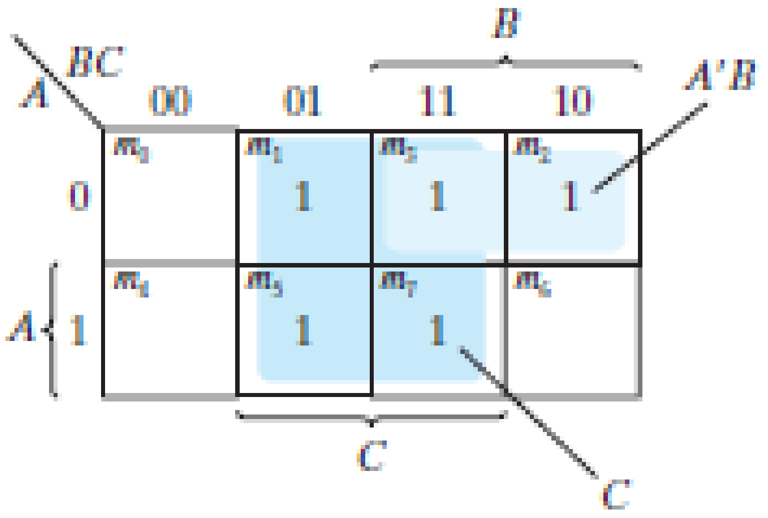
$$F = A'C + A'B + AB'C + BC$$

(a) Express this function as a sum of minterms.
(b) Find the minimal sum-of-products expression.

F=A’C(B+B’)+A’B(C+C’)+AB’C+BC(A+A’)=A’BC+A’B’C+A’BC+A’BC’+AB’C+ABC+A’BC

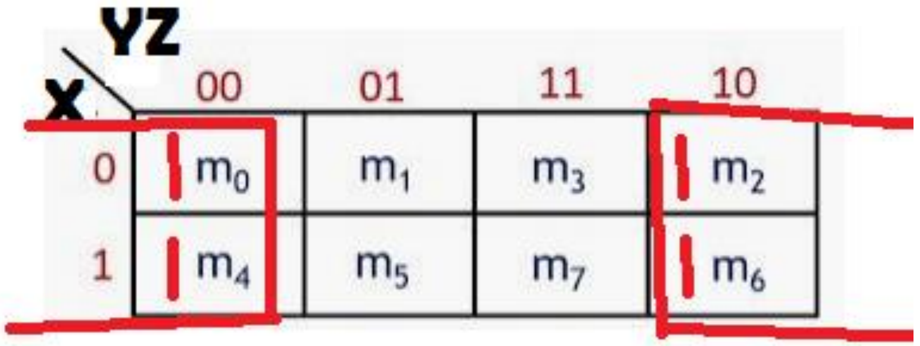
$$F(A, B, C) = \Sigma(1, 2, 3, 5, 7)$$

$$F = C + A'B$$

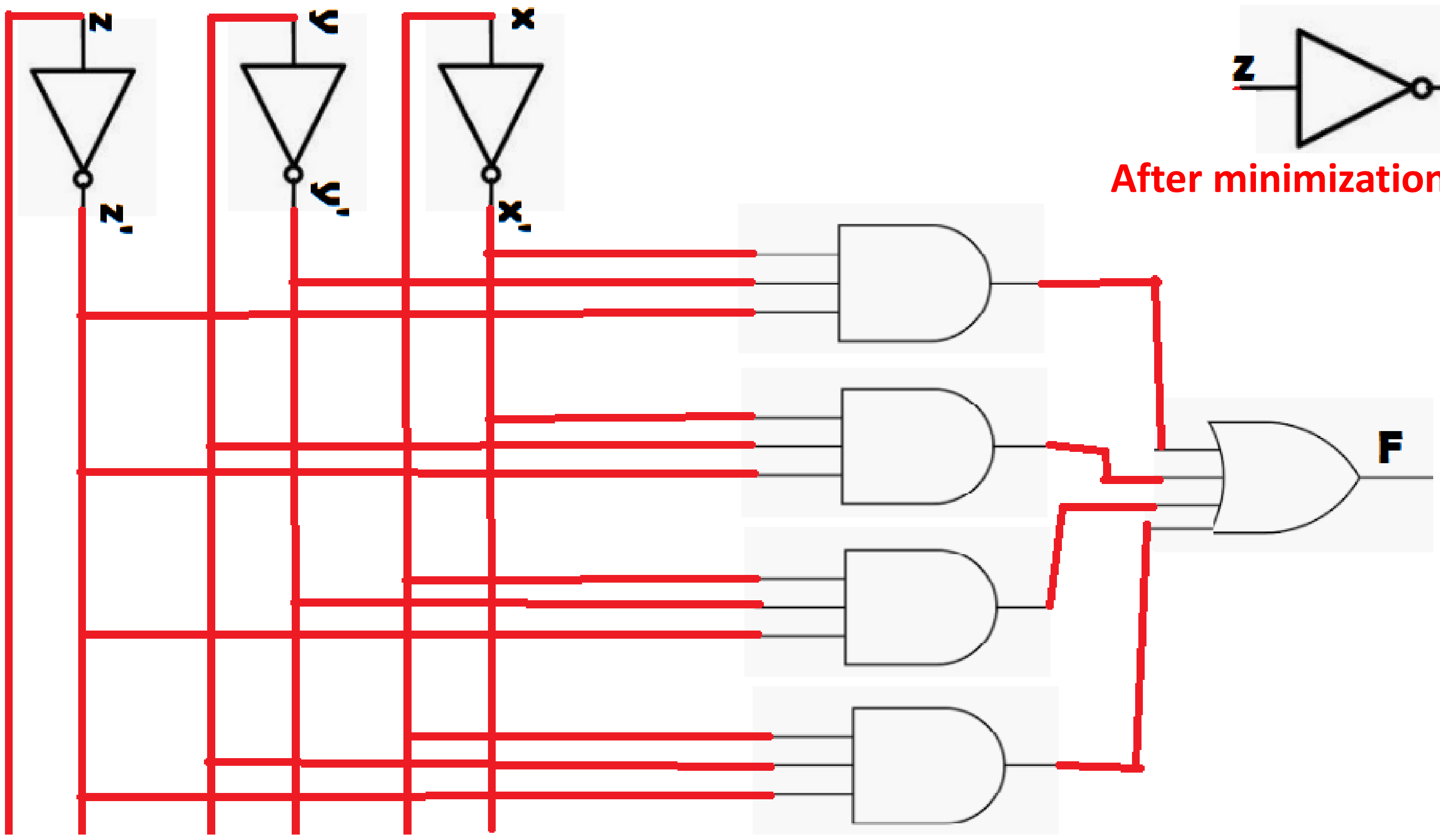


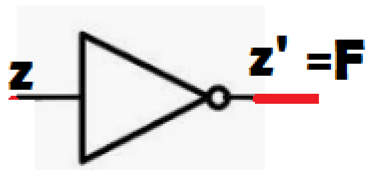
Example: Minimize the logical sum of the four adjacent minterms 0, 2, 4, and 6

$$
\begin{aligned}
m_0 + m_2 + m_4 + m_6 &= x'y'z' + x'yz' + xy'z' + xyz' \\
&= x'z'(y' + y) + xz'(y' + y) \\
&= x'z' + xz' = z'(x' + x) = z'
\end{aligned}
$$



CIRCUIT OPTIMIZATION





After minimization of function

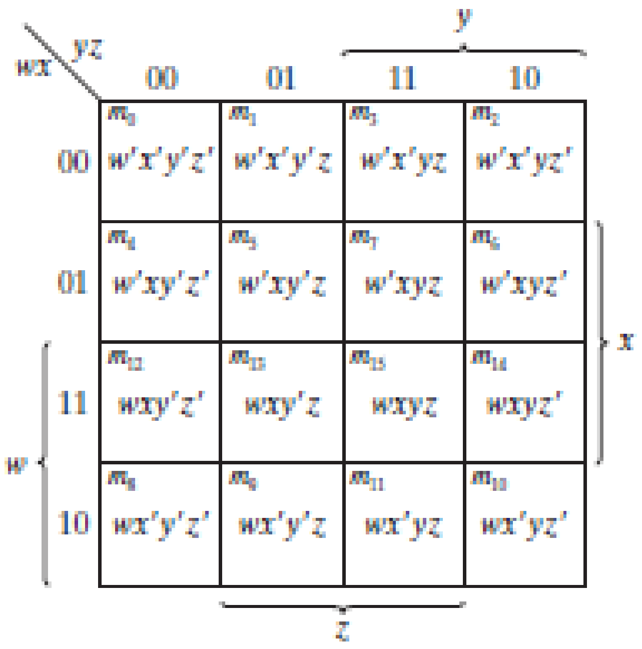
Before minimization of function

HDL Description of circuit

F = X'Y'Z' + X'YZ' + XY'Z' + XYZ'

```
// Verilog model: Circuit with Boolean expressions
module Circuit_Boolean (F, X, Y, Z);
Output  F;
Input  X, Y, Z ;
assign F = ((!X) && (!Y) && (!Z) ) || ((!X) && Y && (!Z) ) || (X  &&
    (!Y) && (!Z) ) || (X && Y && (!Z) );
endmodule
```

FOUR-VARIABLE K-MAP



- The map for Boolean functions of four binary variables (*w, x, y, z) is shown in Figure.* The map is drawn to show the relationship between the squares and the four variables.

- The rows and columns are numbered in a Gray code sequence, with only one digit changing value between two adjacent rows or columns.

- The minterm corresponding to each square can be obtained from the concatenation of the row number with the column number. For example, the numbers of the third row (11) and the second column (01), when concatenated, give the binary number 1101, the binary equivalent of decimal 13. Thus, the square in the third row and second column represents minterm *m13.*

FOUR-VARIABLE K-MAP

- The map minimization of four-variable Boolean functions is similar to the method used to minimize three-variable functions.

- Adjacent squares are defined to be squares next to each other. In addition, the map is considered to lie on a surface with the top and bottom edges, as well as the right and left edges, touching each other to form adjacent squares. For example, *m0 and m2 form adjacent squares, as do m3 and m11.*

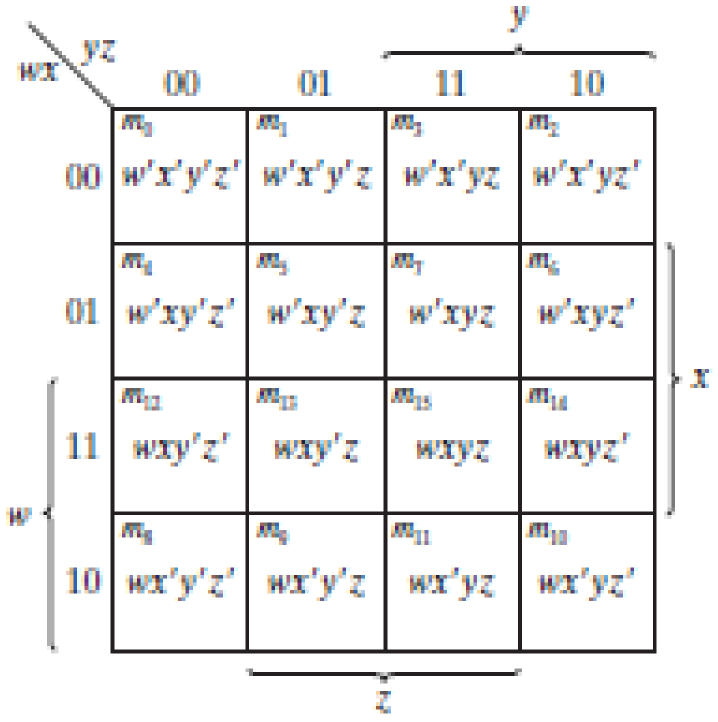
- *The* combination of adjacent squares that is useful during the simplification process is easily determined from inspection of the four-variable map:

FOUR-VARIABLE K-MAP

- One square represents one minterm, giving a term with four literals.
- Two adjacent squares represent a term with three literals.
- Four adjacent squares represent a term with two literals.
- Eight adjacent squares represent a term with one literal.
- Sixteen adjacent squares produce a function that is always equal to 1.

FOUR-VARIABLE K-MAP

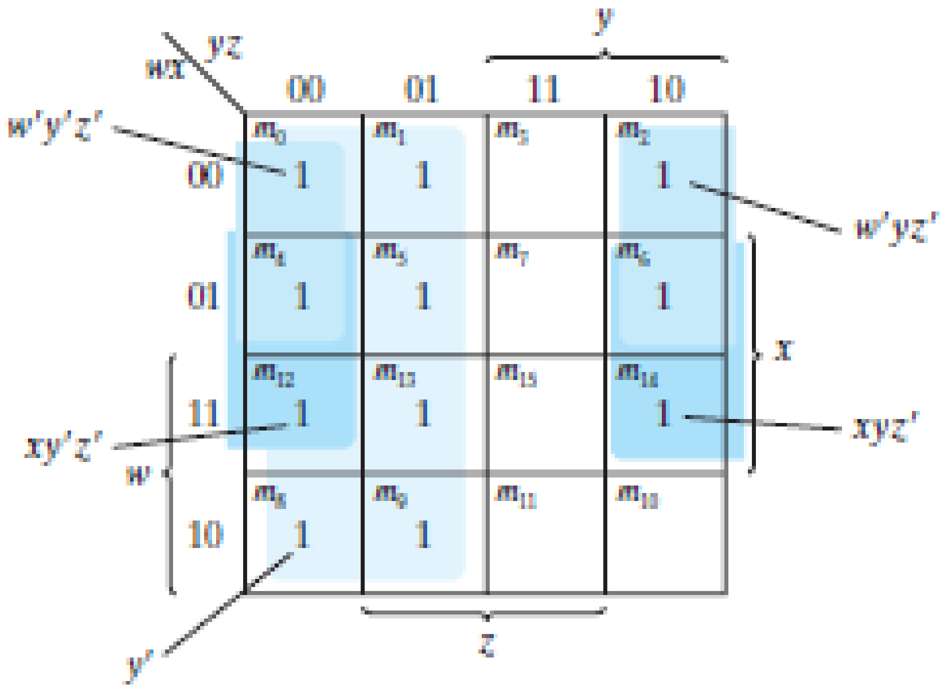
w	x	y	z	
0	0	0	0	m0
0	0	0	1	m1
0	0	1	0	m2
0	0	1	1	m3
0	1	0	0	m4
0	1	0	1	m5
0	1	1	0	m6
0	1	1	1	m7
1	0	0	0	m8
1	0	0	1	m9
1	0	1	0	m10
1	0	1	1	m11
1	1	0	0	m12
1	1	0	1	m13
1	1	1	0	m14
1	1	1	1	m15



Simplify the Boolean function

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

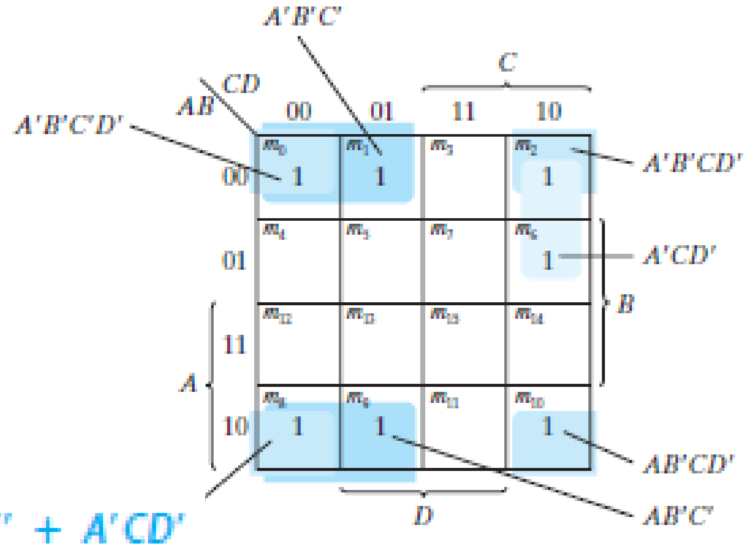
$$F = y' + w'z' + xz'$$



Note: $w'y'z' + w'yz' = w'z'$
$xy'z' + xyz' = xz'$

Simplify the Boolean function

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

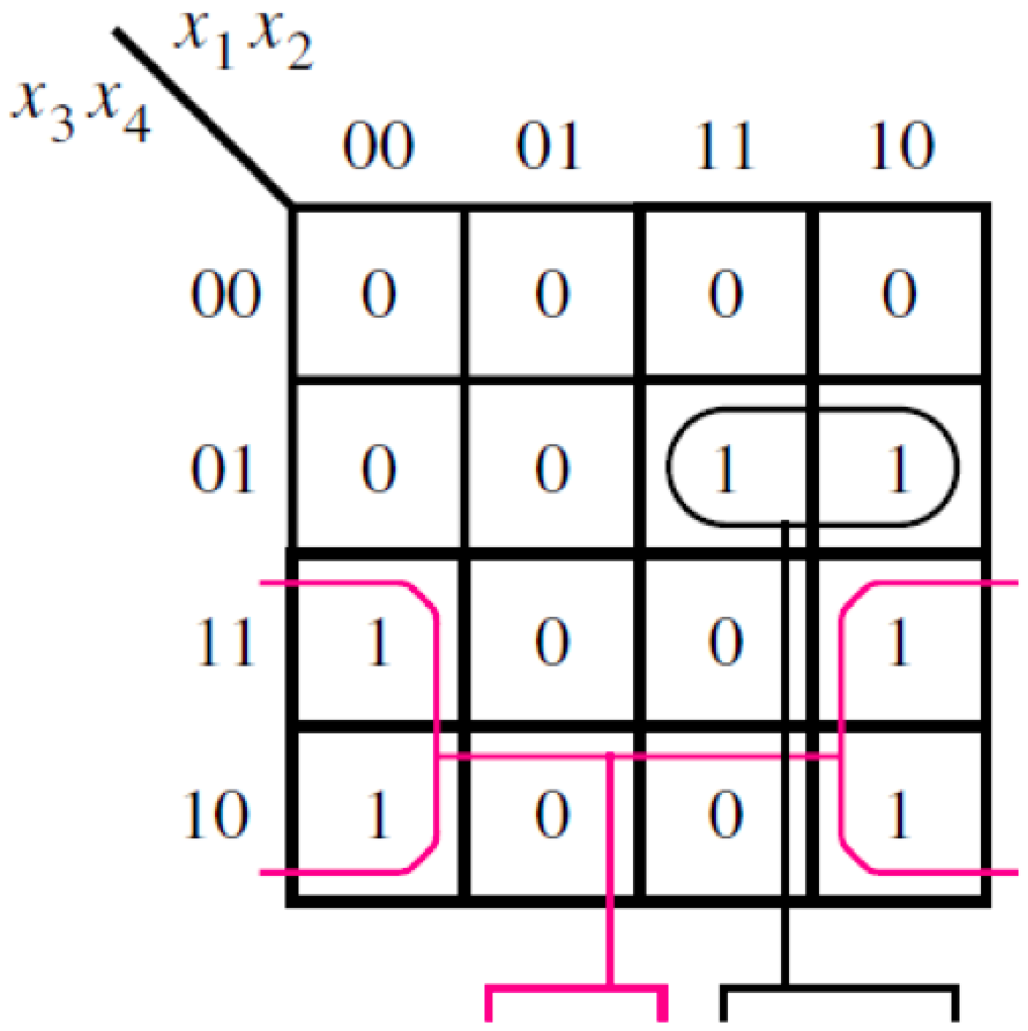


$$A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$$

Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$
$A'B'C' + AB'C' = B'C'$

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

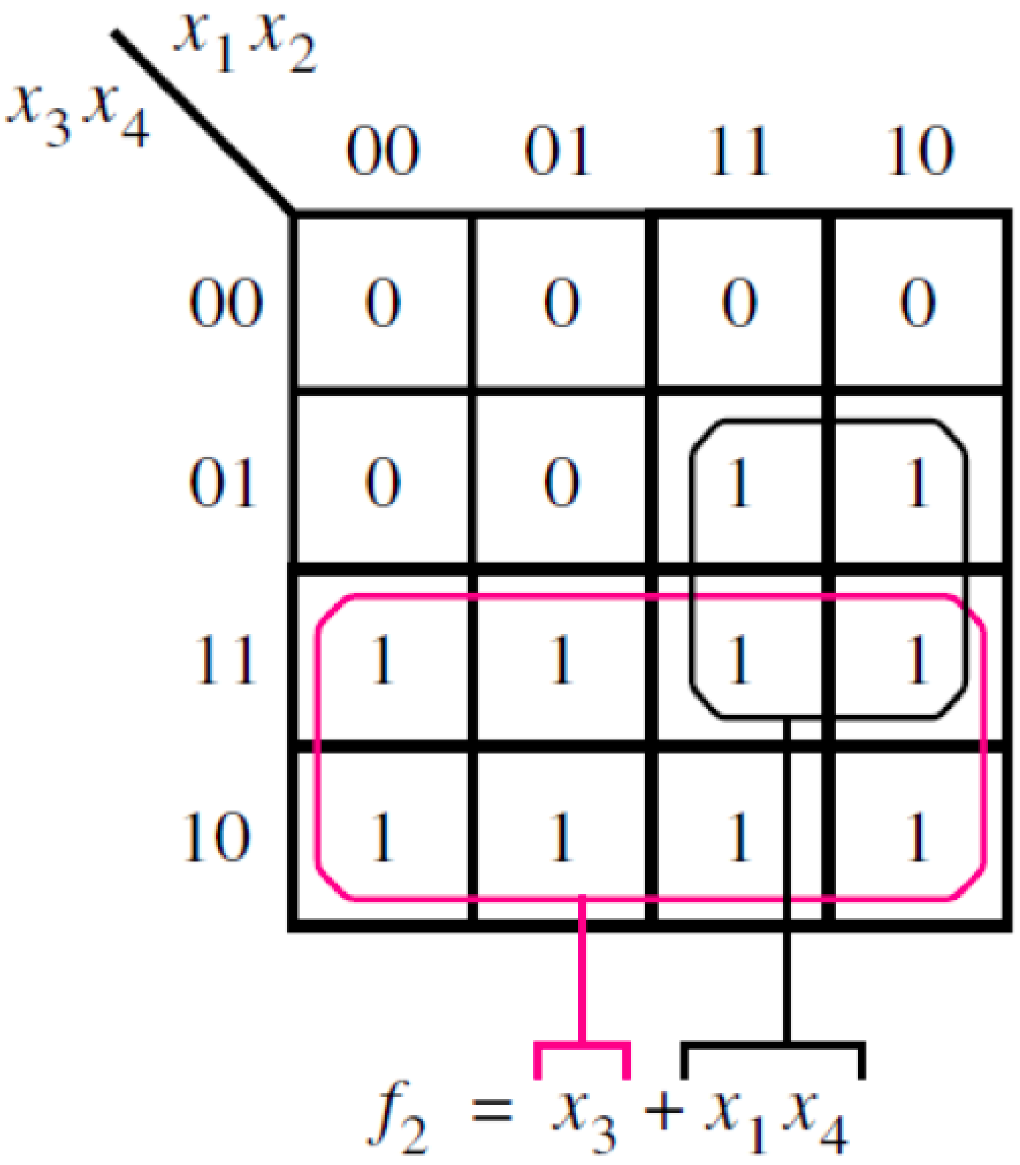
Example of a four-variable Karnaugh map



$$f_1 = \bar{x}_2 x_3 + x_1 \bar{x}_3 x_4$$

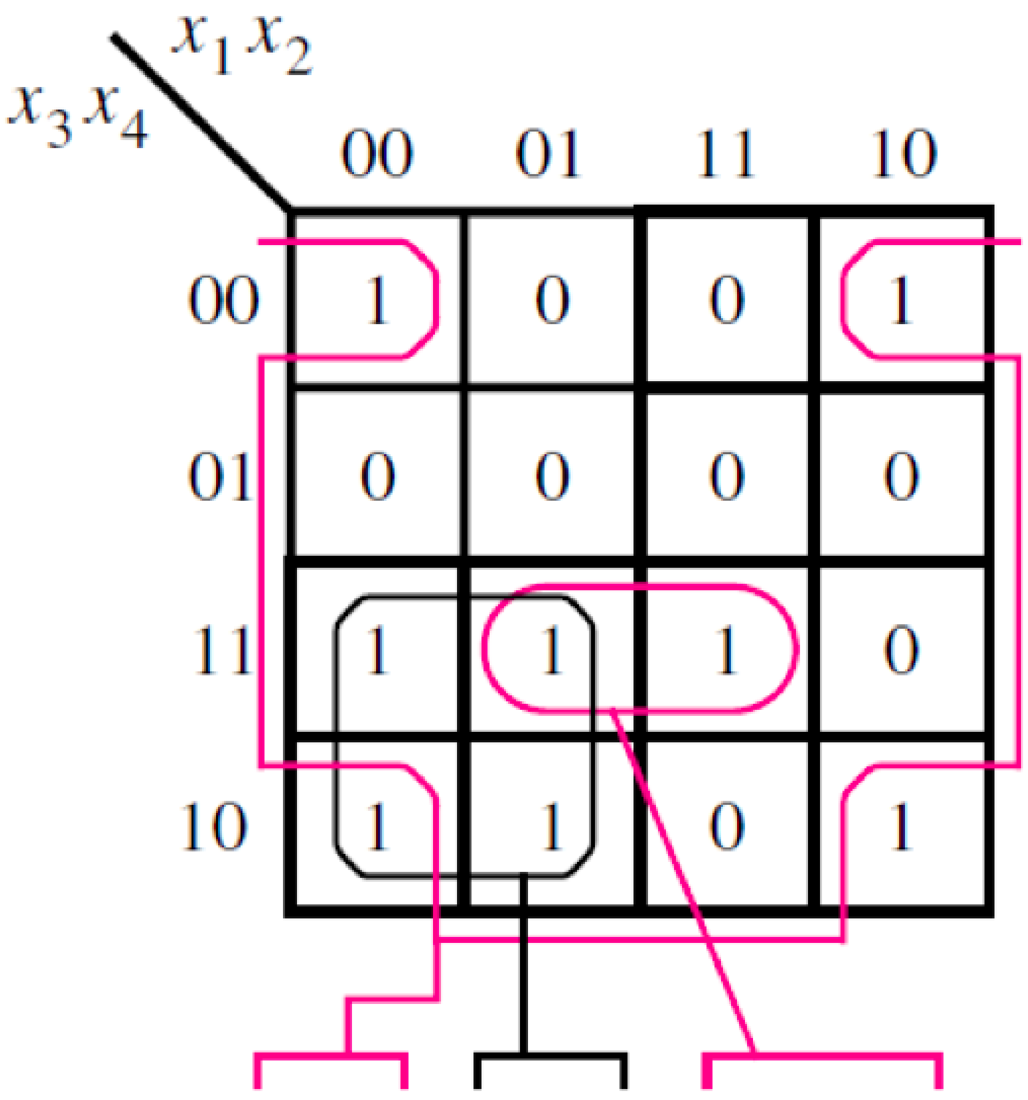
[Figure 2.54 from the textbook]

Example of a four-variable Karnaugh map



[Figure 2.54 from the textbook]

Example of a four-variable Karnaugh map

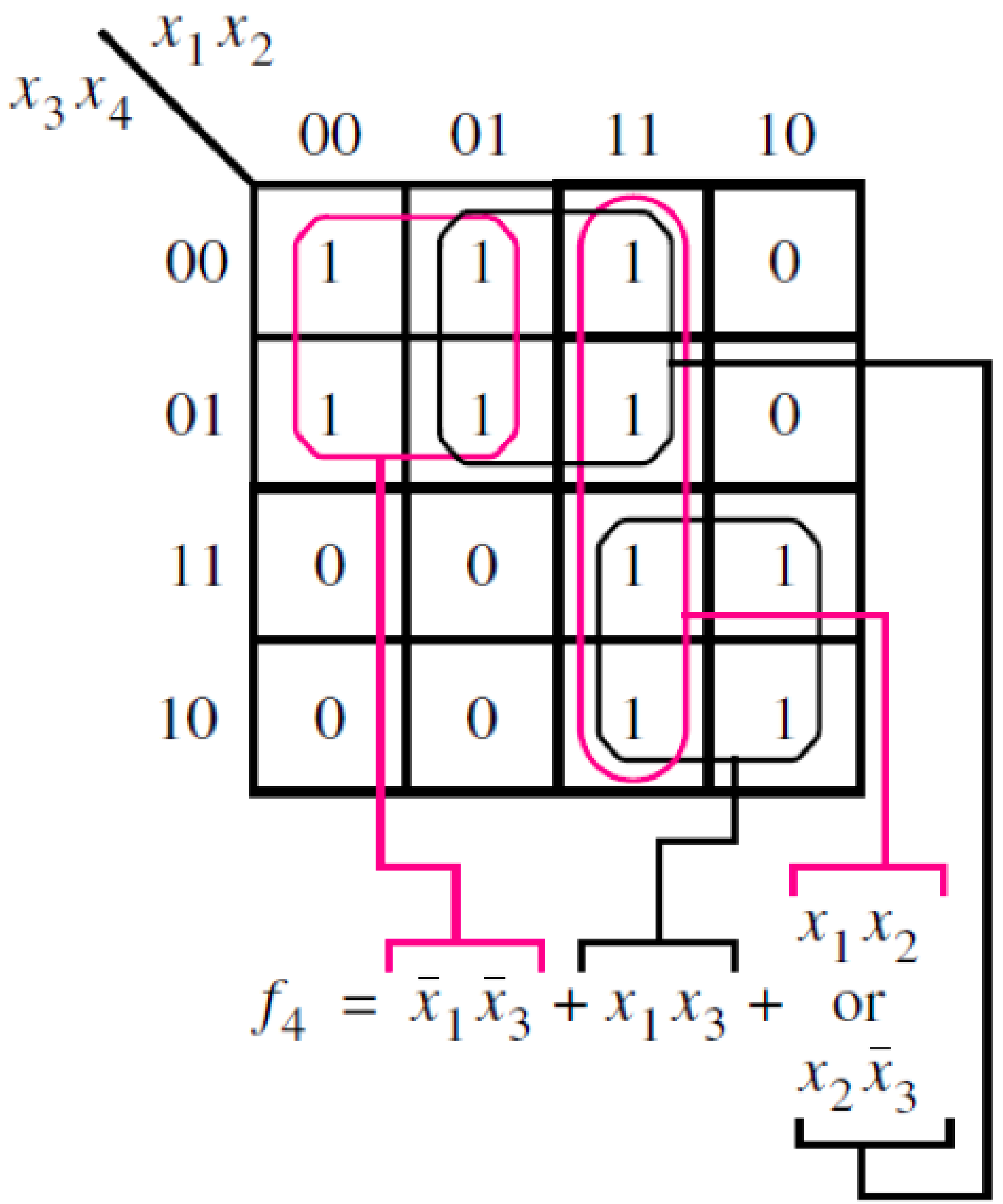


x_3x_4 \ x_1x_2	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	1	1	1	0
10	1	1	0	1

$$f_3 = \bar{x}_2\bar{x}_4 + \bar{x}_1 x_3 + x_2 x_3 x_4$$

[Figure 2.54 from the textbook]

Example of a four-variable Karnaugh map



[Figure 2.54 from the textbook]

HDL Description of The simplified Function

$F = X_1'X_3' + X_1X_3 + X_1X_2$

```
// Verilog model: Circuit with Boolean expressions
module Circuit (F, X1, X2, X3);
Output  F;
Input  X1, X2, X3;
assign F = ((! X1) && (! X3) ) || (X1 && X3 ) || (X1  && X2 );
endmodule
```

HDL Description of The Simplified Function

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

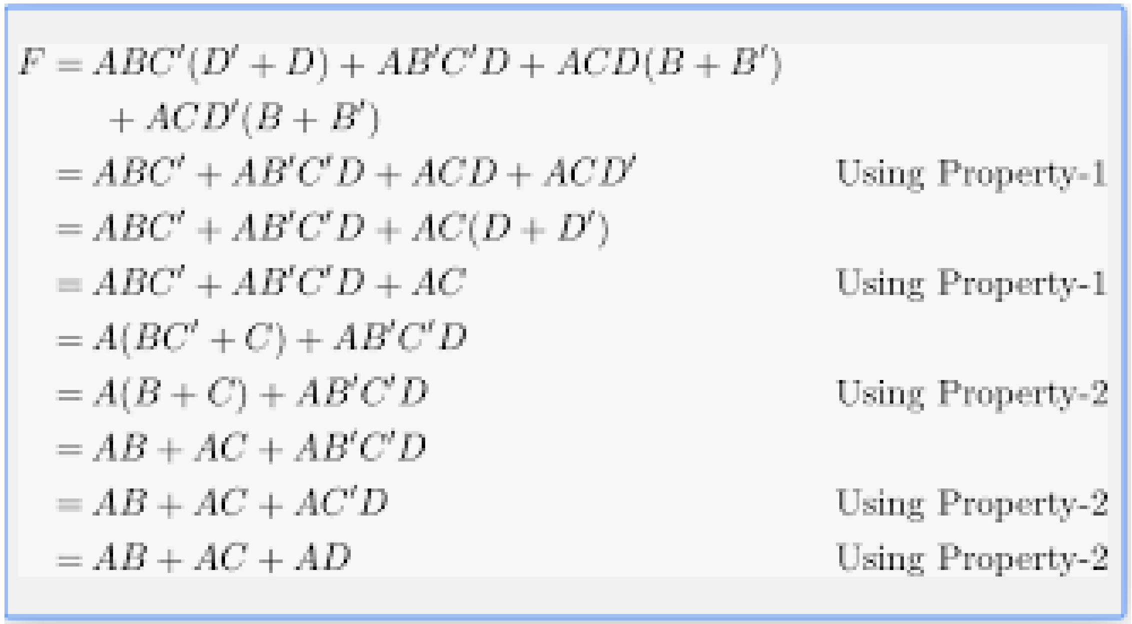
```
// Verilog model: Circuit with Boolean expressions
module Circuit (F, A,B,C,D);
Output  F;
Input  A, B, C, D;
assign F = ((! B) && (! D) ) || ((! B) && (! C) ) || ((! A) && C && (! D) );
endmodule
```

HDL Description of circuit

$F = X_1'X_3' + X_1X_3 + X_1X_2$

```
// Verilog model: Circuit with Boolean expressions
module Circuit (F, X1, X2, X3);
Output  F;
Input  X1, X2, X3;
assign F = ((! X1) && (! X3) ) || (X1&&X3 ) || (X1  && X2 );
endmodule
```

F=ABC’+AB’C’D+ACD+ACD’



$$
\begin{aligned}
F &= ABC'(D' + D) + AB'C'D + ACD(B + B') \\
&\quad + ACD'(B + B') \\
&= ABC' + AB'C'D + ACD + ACD' && \text{Using Property-1} \\
&= ABC' + AB'C'D + AC(D + D') \\
&= ABC' + AB'C'D + AC && \text{Using Property-1} \\
&= A(BC' + C) + AB'C'D \\
&= A(B + C) + AB'C'D && \text{Using Property-2} \\
&= AB + AC + AB'C'D \\
&= AB + AC + AC'D && \text{Using Property-2} \\
&= AB + AC + AD && \text{Using Property-2}
\end{aligned}
$$

Prime Implicants

- 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, and

(3) there are no redundant terms (i.e., minterms already covered by other terms).

- Sometimes there may be two or more expressions that satisfy the simplification criteria. The procedure for combining squares in the map may be made more systematic if we understand the meaning of two special types of terms.

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.*

- **The prime implicants of a function can be obtained from the map by combining all possible maximum numbers of squares.**

Prime Implicants

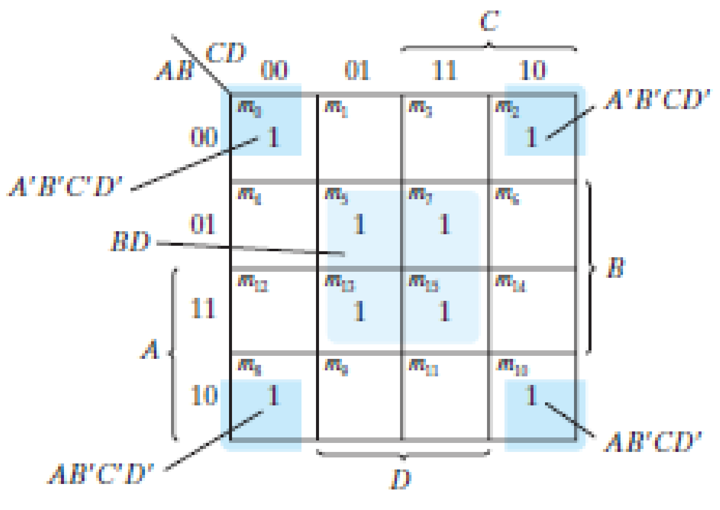
- This means that a single 1 on a map represents a prime implicant if it is not adjacent to any other 1’s. Two adjacent 1’s form a prime implicant, provided that they are not within a group of four adjacent squares.

- Four adjacent 1’s form a prime implicant if they are not within a group of eight adjacent squares, and so on.

- The essential prime implicants are found by looking at each square marked with a 1 and checking the number of prime implicants that cover it. The prime implicant is essential if it is the only prime implicant that covers the minterm.

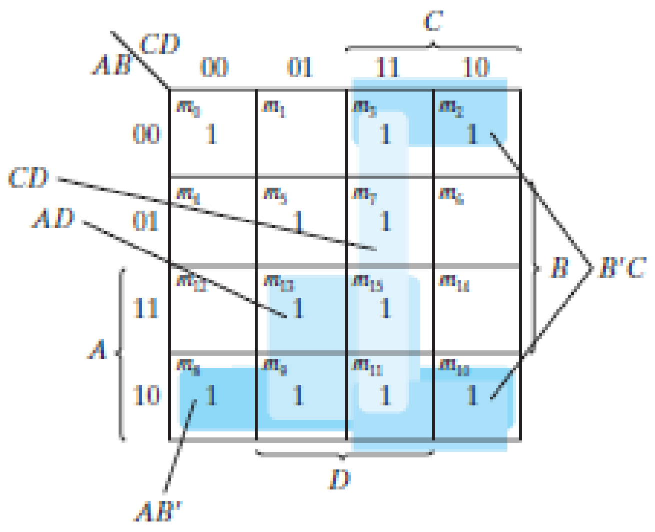
Consider the following four-variable Boolean function:

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



Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$

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



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

$$\begin{aligned} F &= BD + B'D' + CD + AD \\ &= BD + B'D' + CD + AB' \\ &= BD + B'D' + B'C + AD \\ &= BD + B'D' + B'C + AB' \end{aligned}$$

PRODUCT-OF-SUMS SIMPLIFICATION

- The procedure for obtaining a minimized function in product-of-sums form 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 standard sum-of-products form of a function denote the complement of the function.

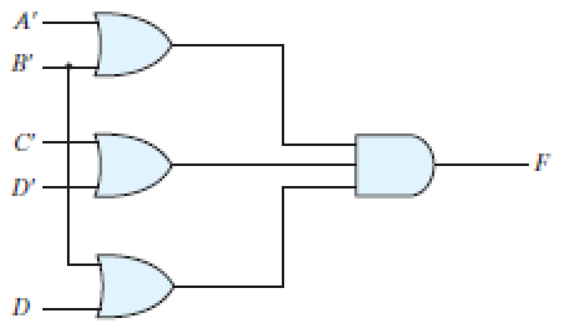
- From this observation, we see that the complement of a function is represented in the map by the squares not marked by 1's. If we mark the empty squares by 0's and combine them into valid adjacent squares, we obtain a simplified sum-of-products expression of the complement of the function (i.e., of *F).*

- *The complement of F gives us back the function F in product-of-sums form (a consequence of DeMorgan's theorem). Because of* the generalized DeMorgan's theorem, the function so obtained is automatically in product-of-sums form. The best way to show this is by example.

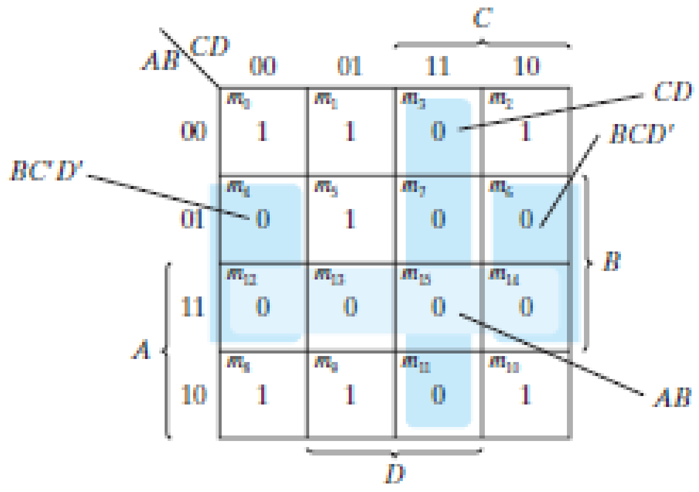
Simplify the following Boolean function into product-of-sums form

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

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



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



Note: $BC'D' + BCD' = BD'$

Practice Questions

1. Reduce the following equations

   (a) $Y = AB'C + AB'C' + A'BC'$

   (b) $Y = m(0,2,4,6,7)$

   (c) $F(A,B,C) = A'C' + A'B + AB'C$

   (d) $F(X,Y,Z) = m_0 + m_2 + m_5 + m_6 + m_7$

   (e) $Y = A'BC' + ABC + ABC' + A'BC + AB'C$

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

Practice Questions

Reduce the following equations

(a) $F(W,X,Y,Z) = \sum (0,1,2,3,5,6,11,13)$

(b) $F(P,Q,R,S)=\sum(0,2,5,7,8,10,13,15)$

(c) $Y(A,B,C,D) =\sum(2,3,12,13,14,15)$

(d) $F(A,B,C,D) = \sum(7,13,14,15)$

(e) $Y(A,B,C,D) = \sum(4,6,7,15)$