แก้ไขครั้งล่าสุดเมื่อวันที่ 26 สิงหาคม 2568

Chapter 3

Gate-Level Minimization

Two-Variable K-Map

K-Map = Karnaugh map

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

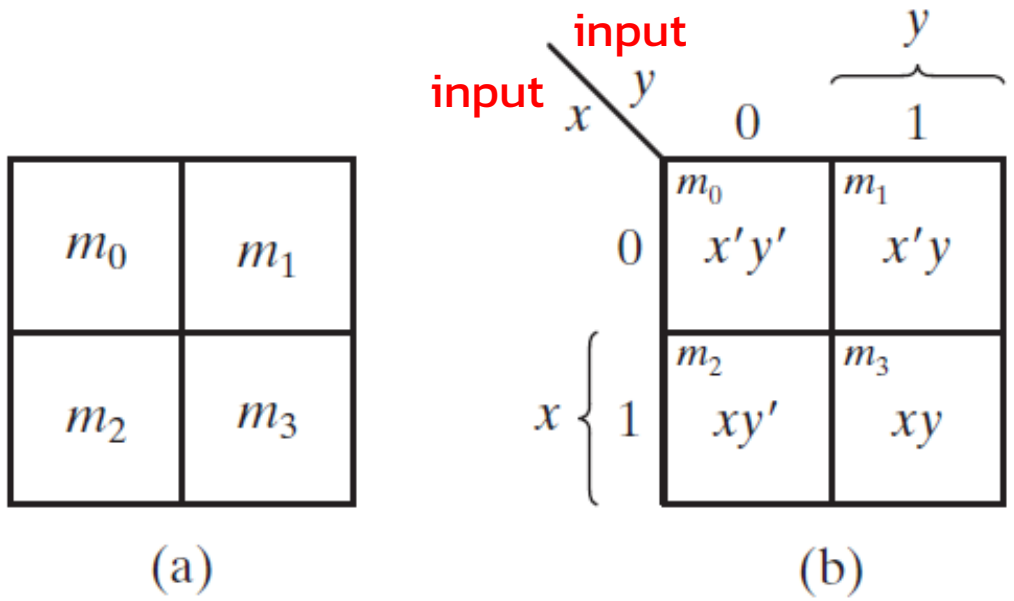


FIGURE 3.1
Two-variable K-map

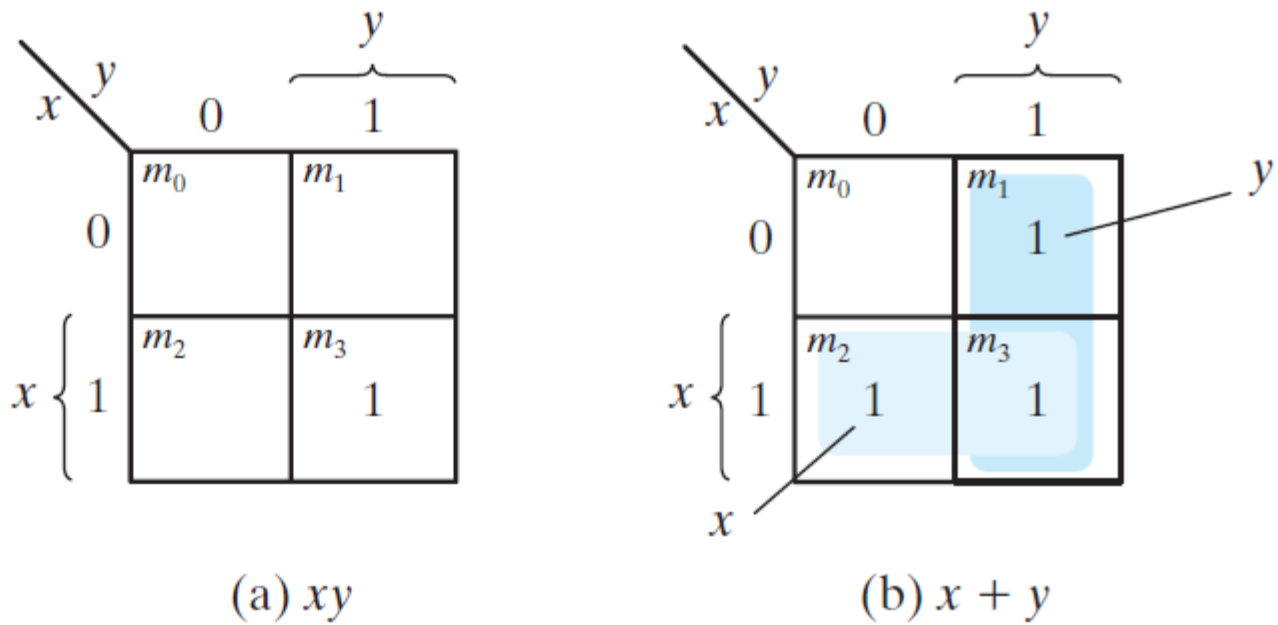
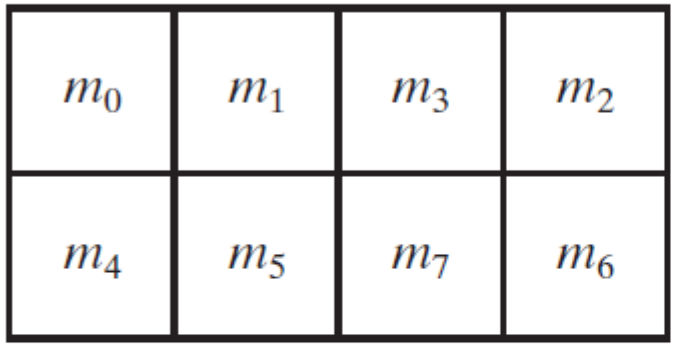
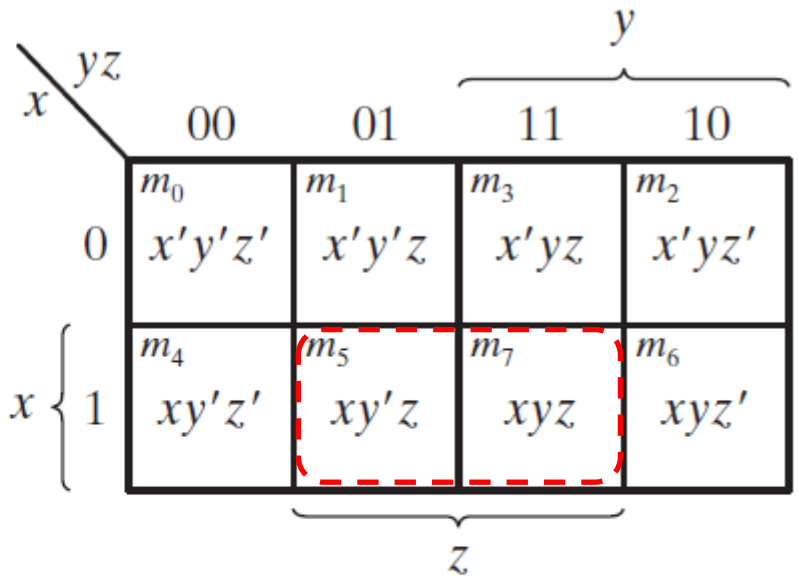


FIGURE 3.2
Representation of functions in the map

Three-Variable K-Map



(a)



(b)

FIGURE 3.3
Three-variable K-map

$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

EXAMPLE 3.1

Simplify the Boolean function

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

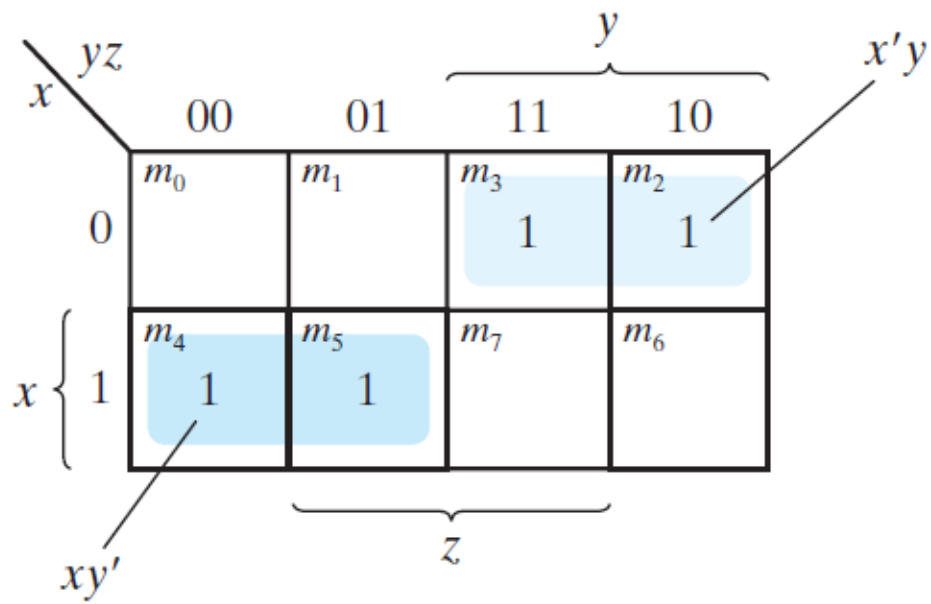
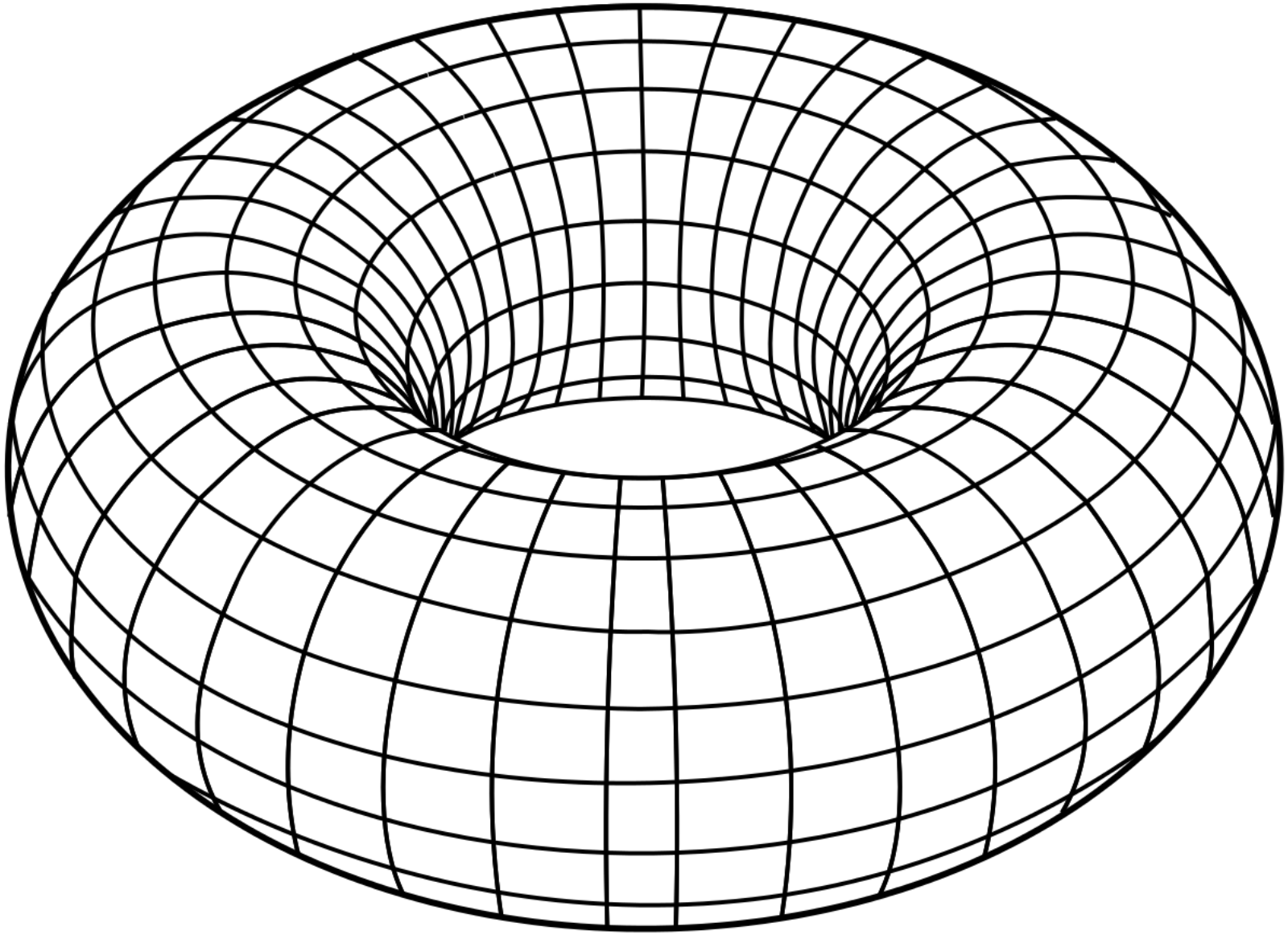


FIGURE 3.4
Map for Example 3.1, $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

K-Map เป็น torus (plural, tori)

วงให้เป็นสี่เหลี่ยม

$2^m \times 2^n$



EXAMPLE 3.2

Simplify the Boolean function

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

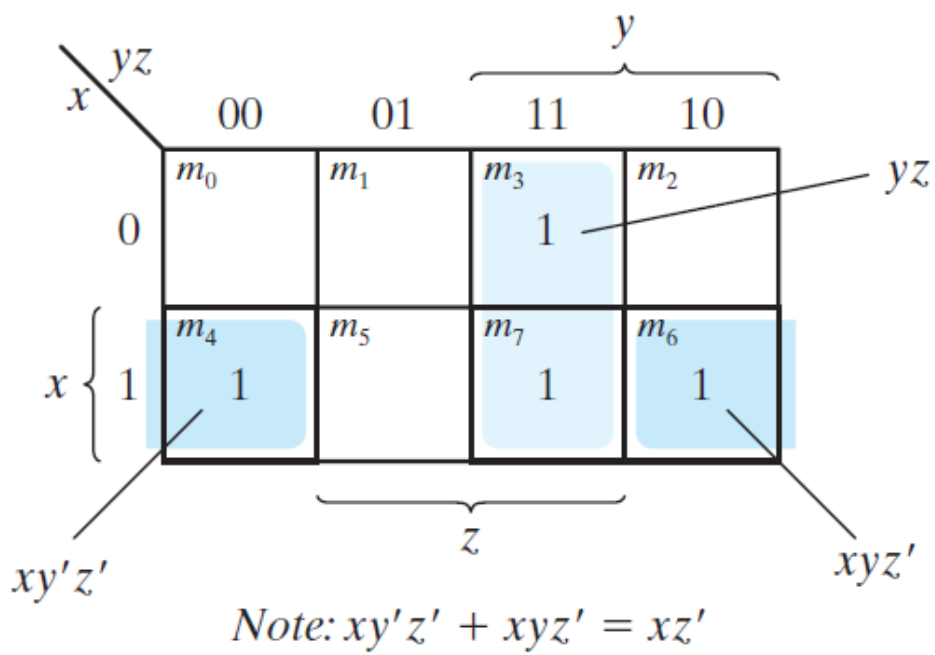


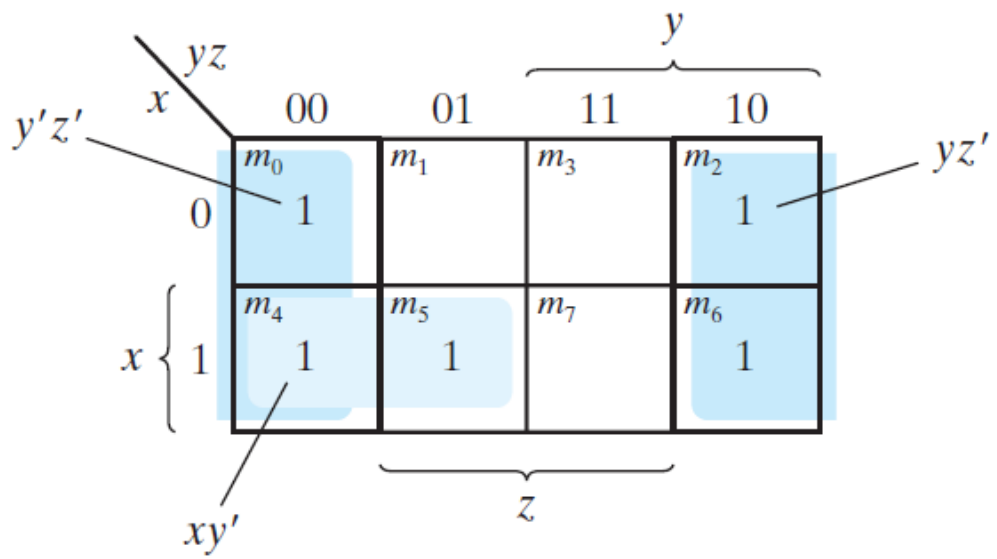
FIGURE 3.5
Map for Example 3.2, $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

EXAMPLE 3.3

Simplify the Boolean function

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

$$F = z' + xy'$$



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

FIGURE 3.6
Map for Example 3.3, $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

EXAMPLE 3.4

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.

Ans (a)

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

Ans (b)

$$F = C + A'B$$

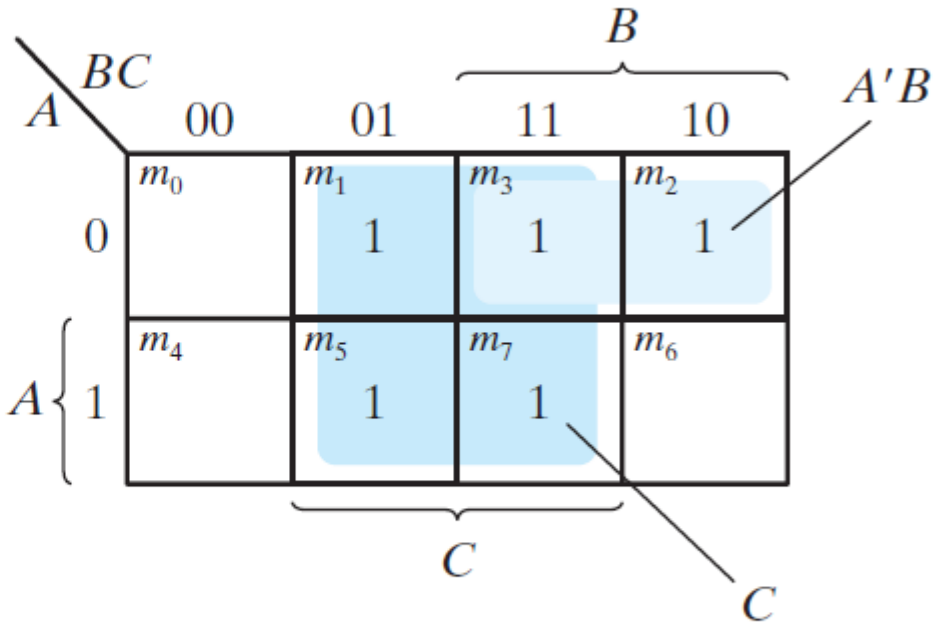
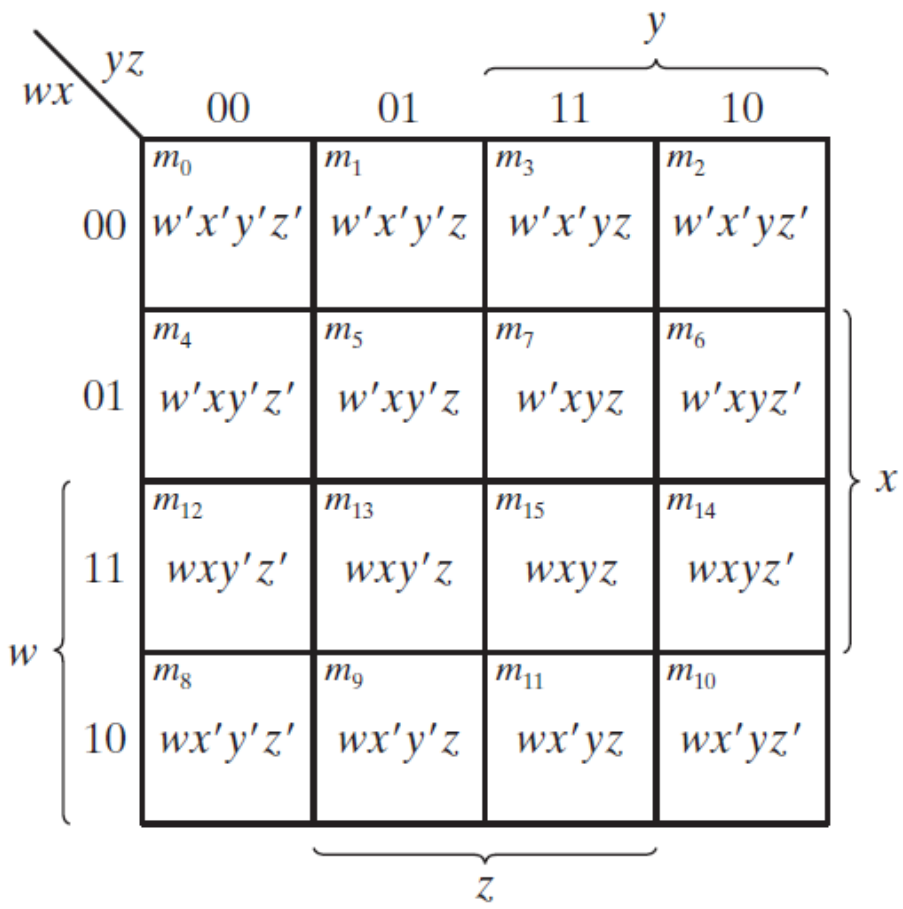


FIGURE 3.7
Map of Example 3.4, $A'C + A'B + AB'C + BC = C + A'B$

Four-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}

(a)



(b)

FIGURE 3.8
Four-variable map

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

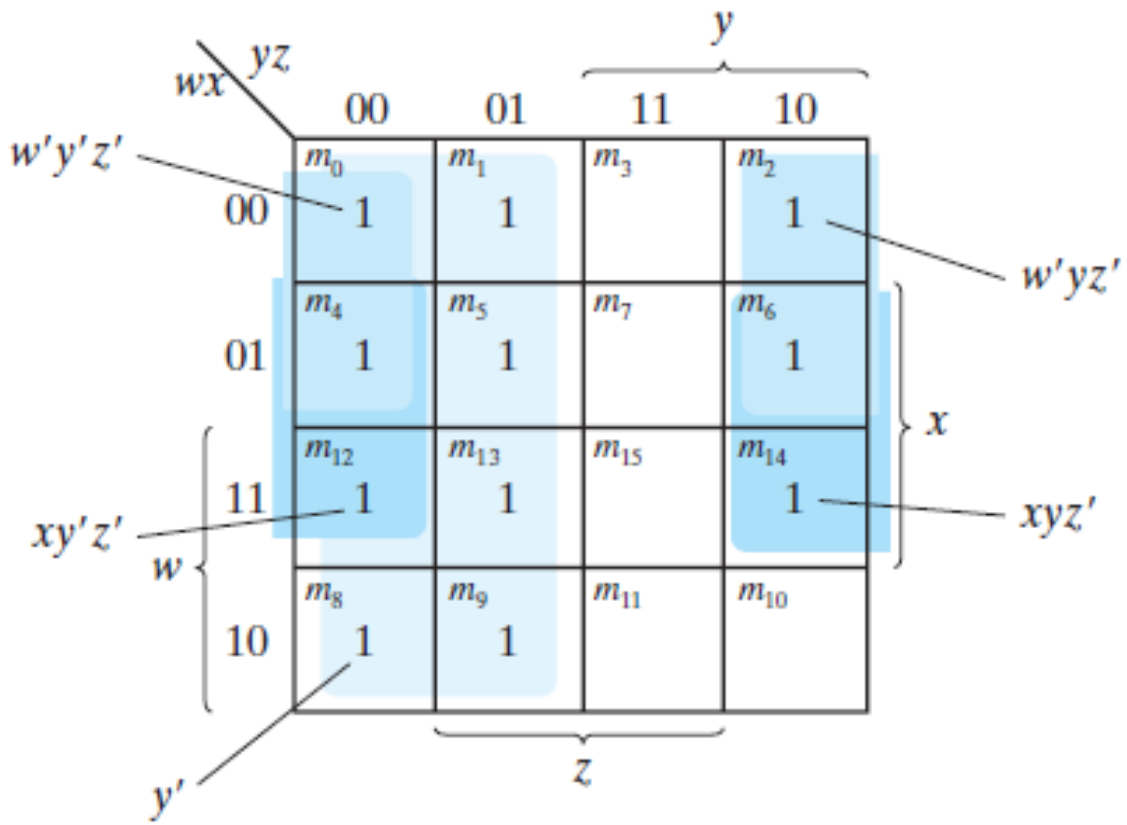
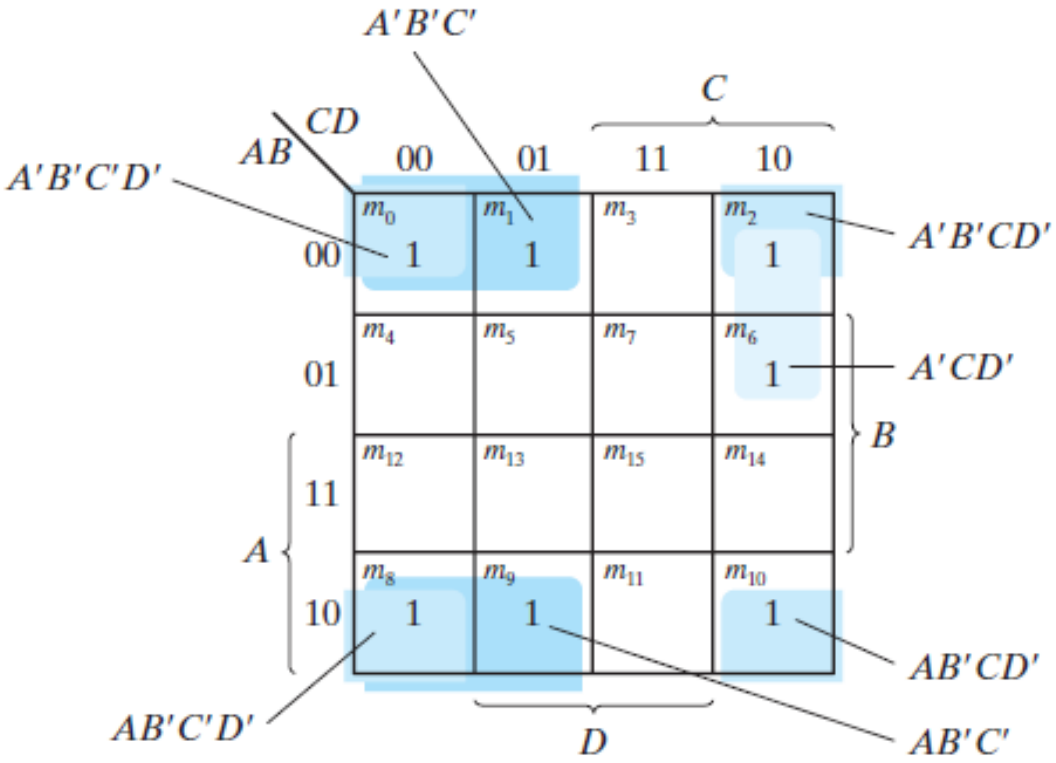


FIGURE 3.9
Map for Example 3.5, $F(w, x, y, z) = \Sigma(0,1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

EXAMPLE 3.6

Simplify the Boolean function

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



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

หลังจาก simplify แล้ว จะได้

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

FIGURE 3.10
Map for Example 3.6, $A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$

Prime Implicants

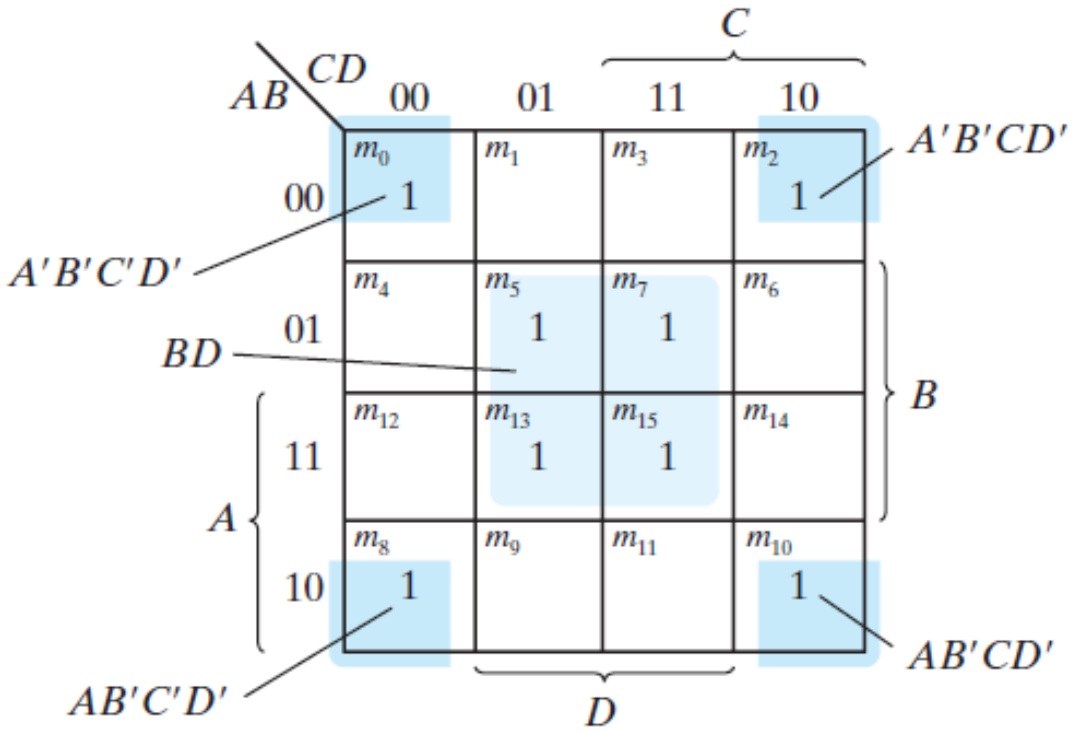
A *prime implicant* is a product term obtained by combining the maximum possible number of adjacent squares in the map.

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

ในการทำ gate-level minimization เราพิจารณาเฉพาะ prime implicants ว่าจะเลือกตัวใดมาใช้บ้าง แต่ไม่จำเป็นต้องเลือกมาทั้งหมด

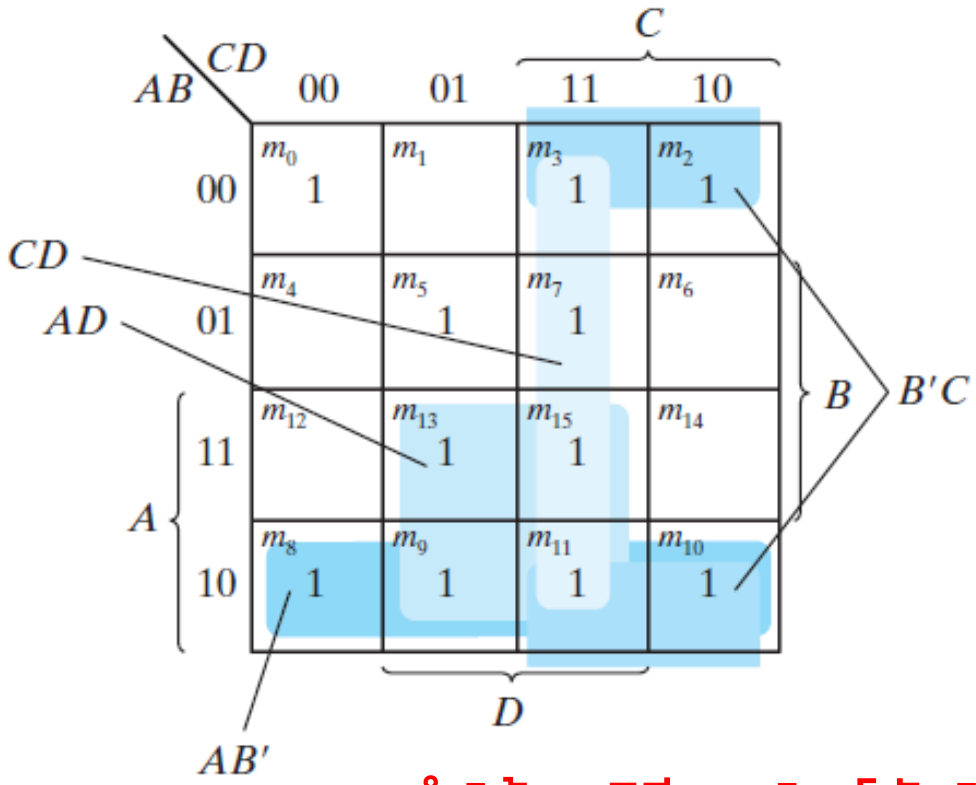
Consider the following four-variable Boolean function:

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



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'

ทำได้ 4 วิธี ดูสไลด์ถัดไป

FIGURE 3.11
Simplification using prime implicants

Figure 3.11(b) shows all possible ways that the three minterms can be covered with prime implicants. Minterm m_3 can be covered with either prime implicant CD or prime implicant $B'C$. Minterm m_9 can be covered with either AD or AB'. Minterm m_{11} is covered with any one of the four prime implicants.

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

Essential prime implicants

Five-Variable Map

Maps for more than four variables are not as simple to use as maps for four or fewer variables. A five-variable map needs 32 squares and a six-variable map needs 64 squares. When the number of variables becomes large, the number of squares becomes excessive and the geometry for combining adjacent squares becomes more involved.

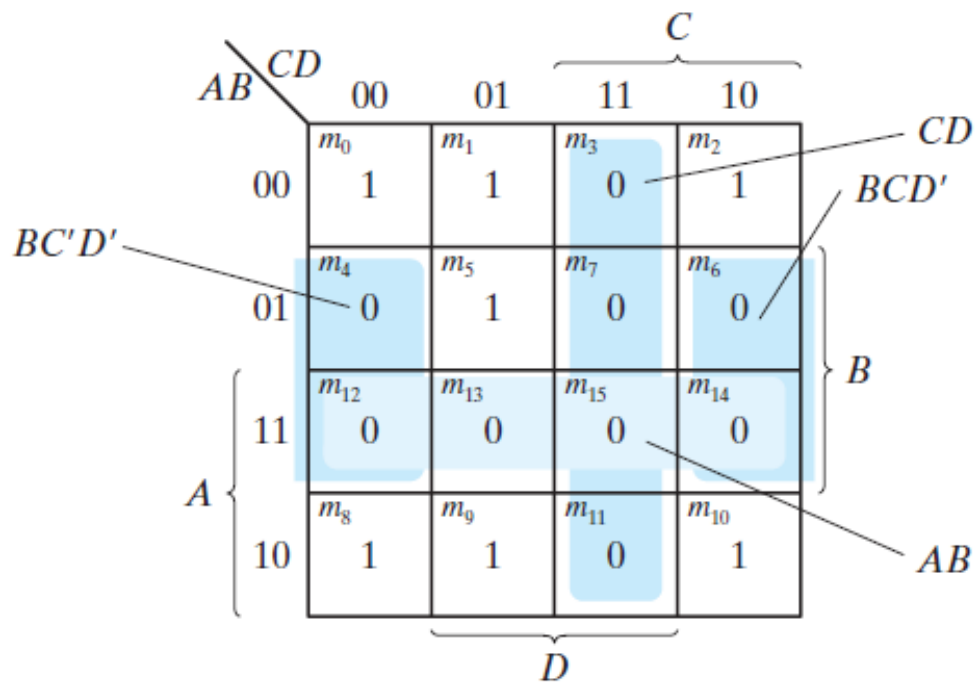
Maps for more than four variables are difficult to use and will not be considered here.

ทำด้วยมือ (hand calculation) ยาก ใช้วิธีเขียนโปรแกรม

EXAMPLE 3.7

Simplify the following Boolean function into (a) sum-of-products form and (b) product-of-sums form:

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



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

FIGURE 3.12
Map for Example 3.7, $F(A, B, C, D) = \Sigma(0,1, 2, 5, 8, 9,10) = B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$

จาก K-Map

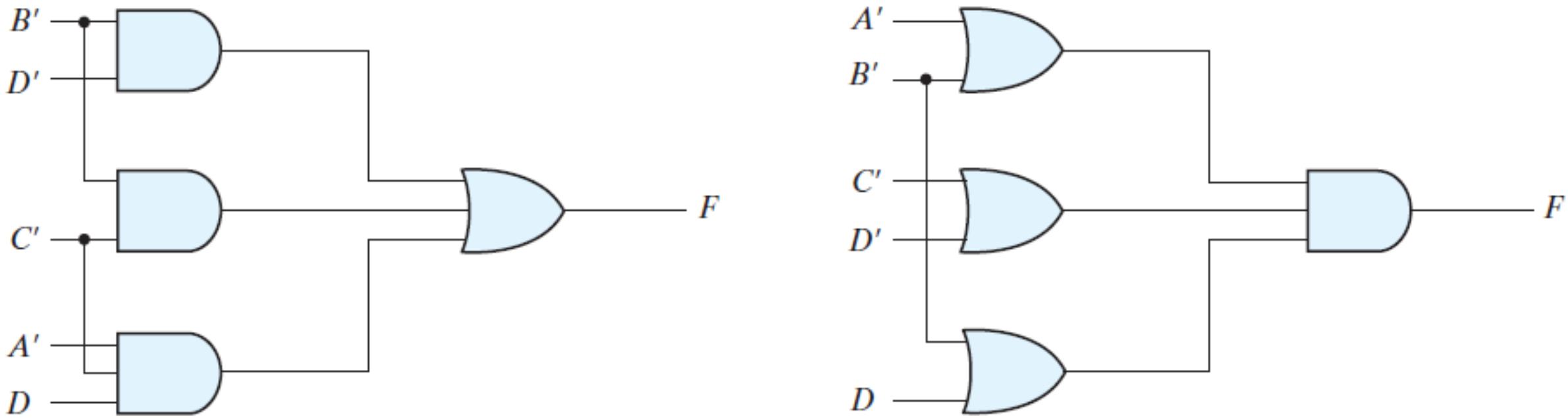
(a) $F = B'D' + B'C' + A'C'D$

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

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

Applying DeMorgan's theorem (by taking the dual and complementing each literal as described in Section 2.4), we obtain the simplified function in product-of-sums form:

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



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

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

FIGURE 3.13
Gate implementations of the function of Example 3.7

Don’t-care conditions

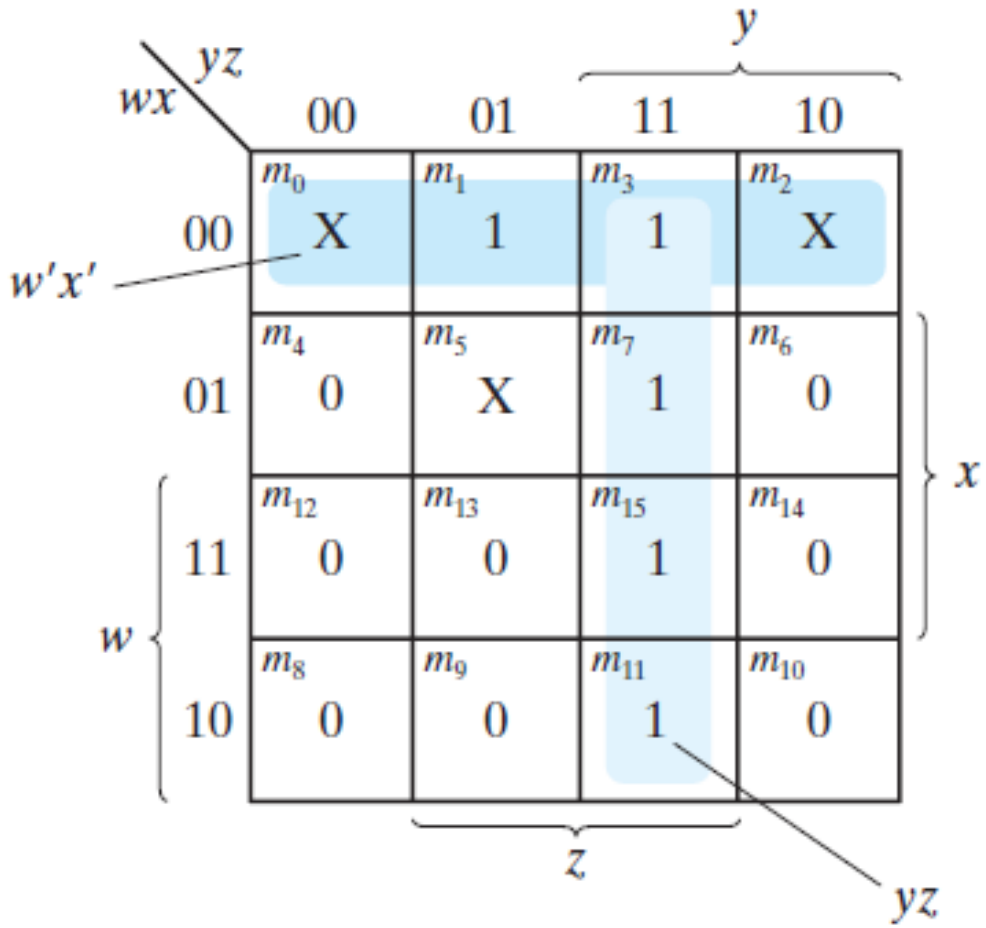
EXAMPLE 3.8

Simplify the Boolean function

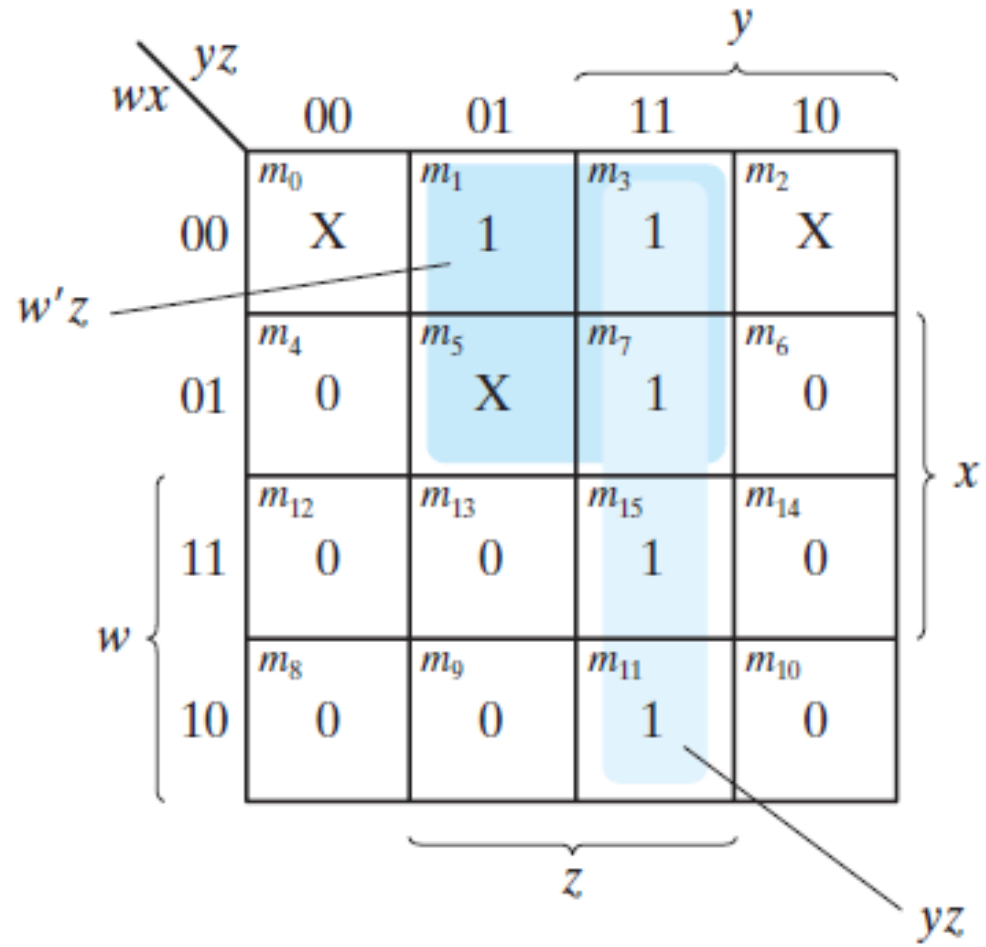
$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

which has the don’t-care conditions

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$



(a) $F = yz + w'x'$



(b) $F = yz + w'z$

FIGURE 3.15
Example with don't-care conditions

$$F(w, x, y, z) = yz + w'x' = \Sigma(0, 1, 2, 3, 7, 11, 15)$$
$$F(w, x, y, z) = yz + w'z = \Sigma(1, 3, 5, 7, 11, 15)$$

It is also possible to obtain a simplified product-of-sums expression for the function of Fig. 3.15. In this case, the only way to combine the 0's is to include don't-care minterms 0 and 2 with the 0's to give a simplified complemented function:

$$F' = z' + wy'$$

Taking the complement of F' gives the simplified expression in product-of-sums form:

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

In this case, we include minterms 0 and 2 with the 0's and minterm 5 with the 1's.

NAND Circuits

เหตุผลที่ใช้ NAND gates

- ใช้ transistor น้อยเพียง 4 ตัว
- จะได้ผลิตแต่ NAND gates

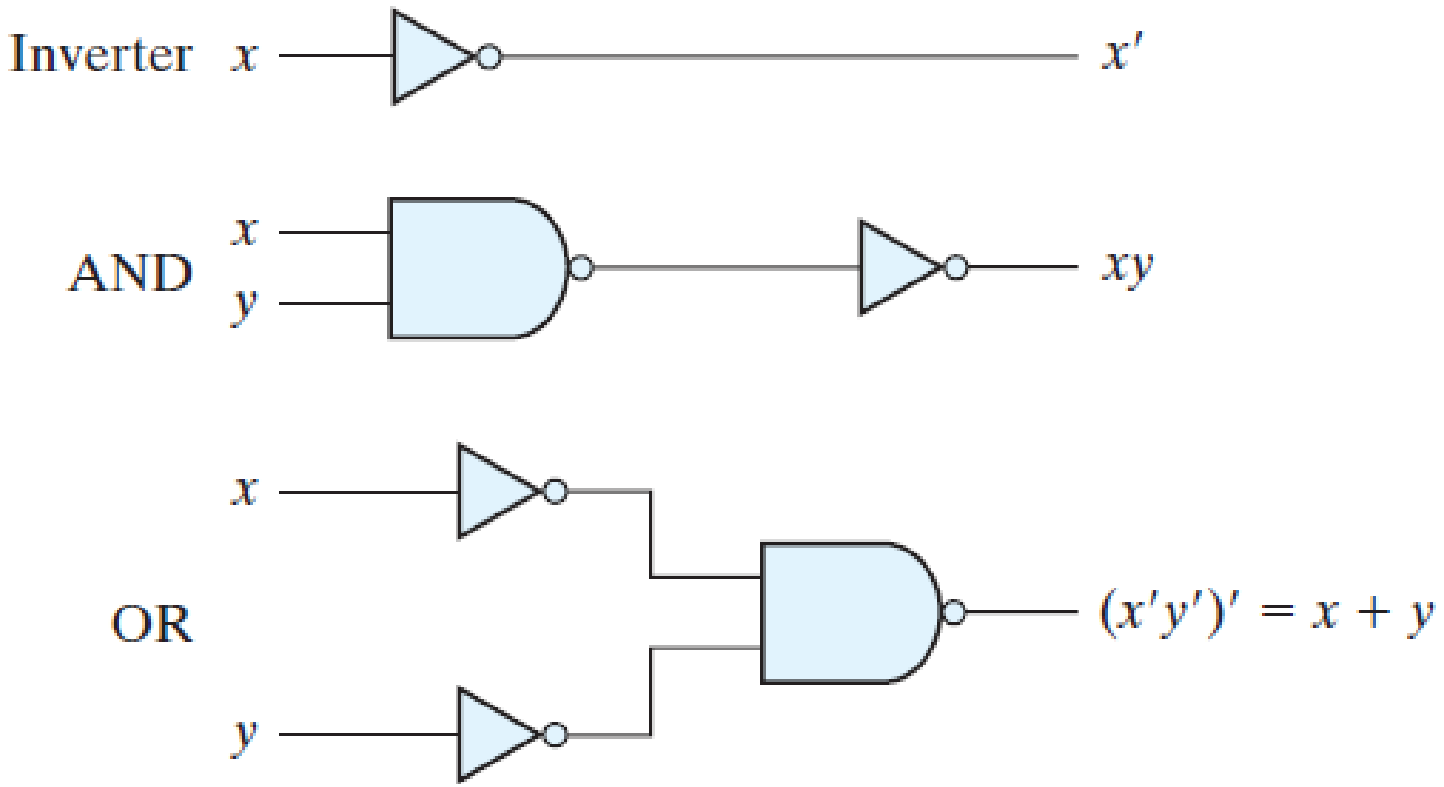


FIGURE 3.16
Logic operations with NAND gates

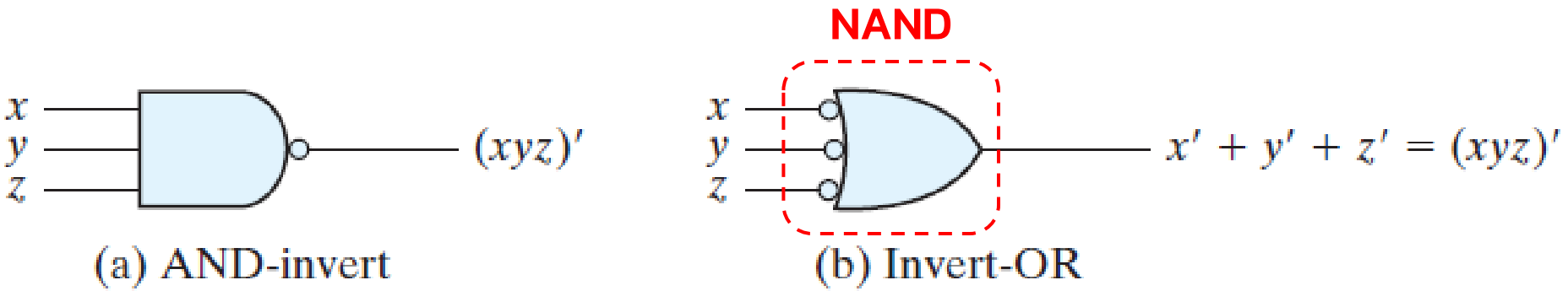
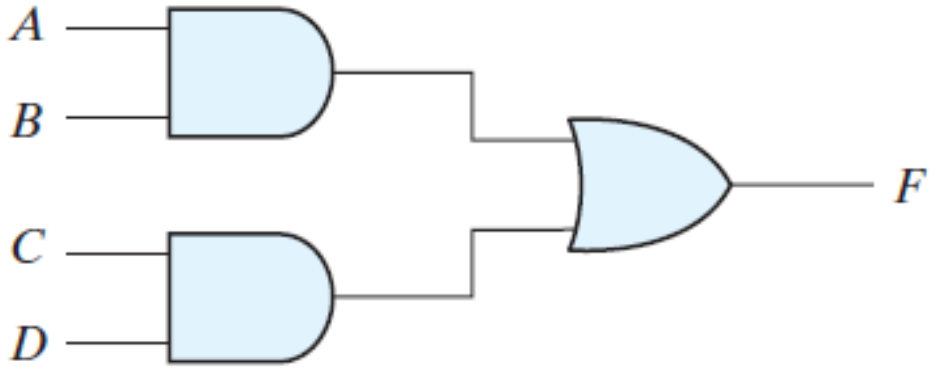
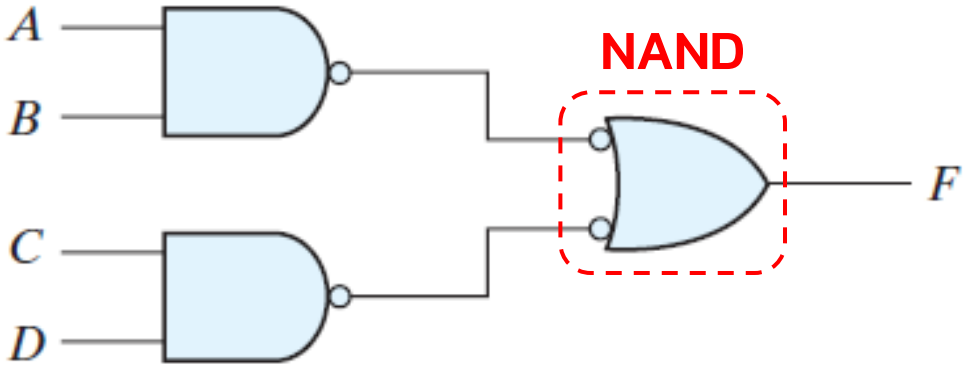


FIGURE 3.17
Two graphic symbols for a three-input NAND gate

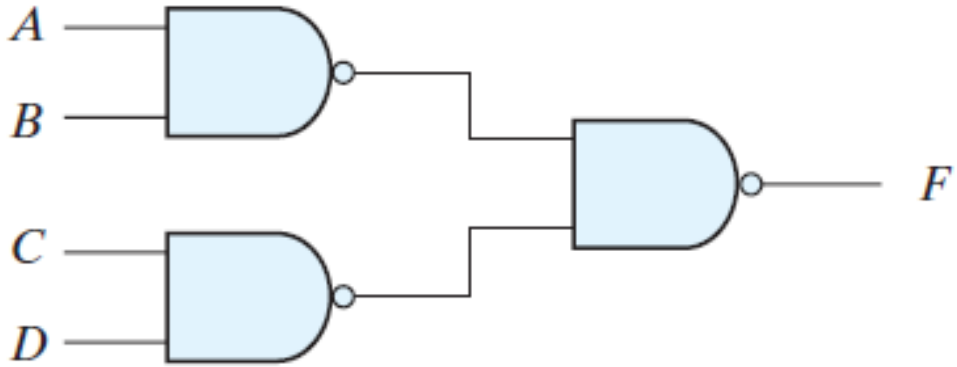


(a)

ใช้ NAND อย่างเดียวก็ได้



(b)



(c)

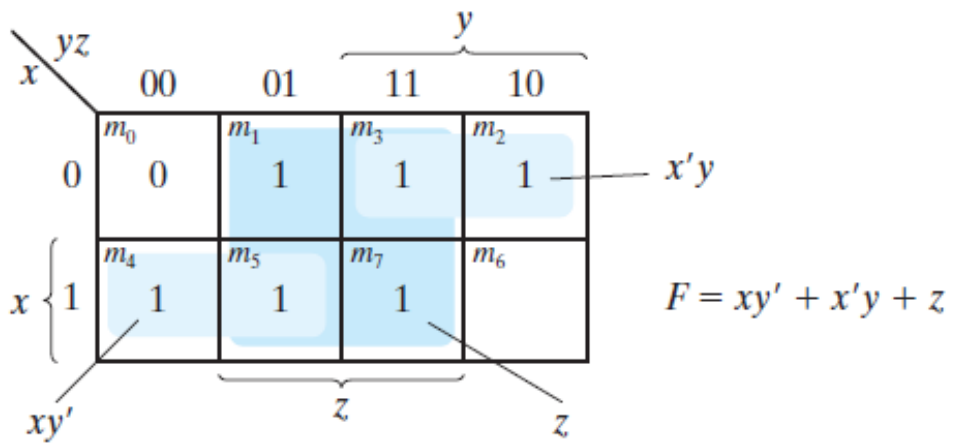
FIGURE 3.18
Three ways to implement $F = AB + CD$

$$F = ((AB)'(CD)')' = AB + CD$$

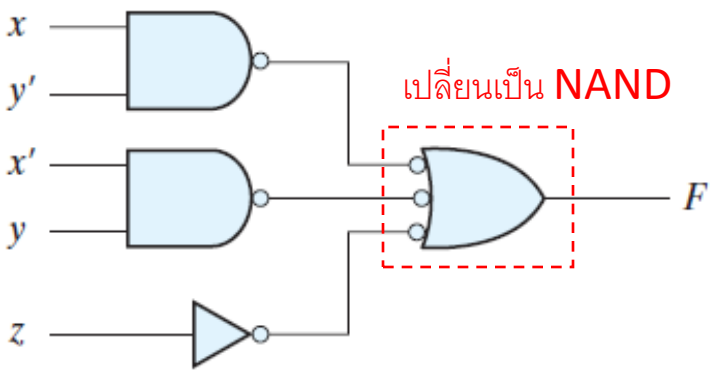
EXAMPLE 3.9

Implement the following Boolean function with NAND gates:

$$F(x, y, z) = (1, 2, 3, 4, 5, 7)$$

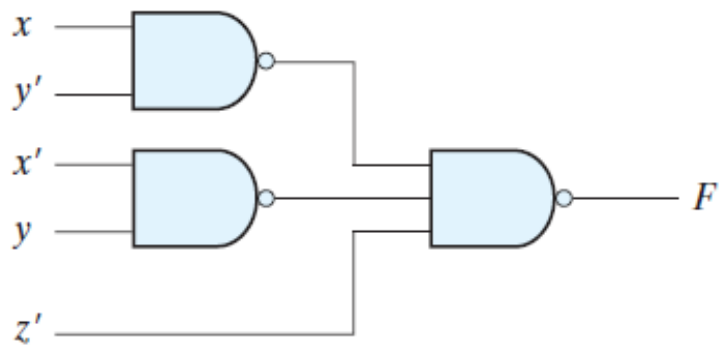


(a)



เปลี่ยนเป็น NAND

(b)

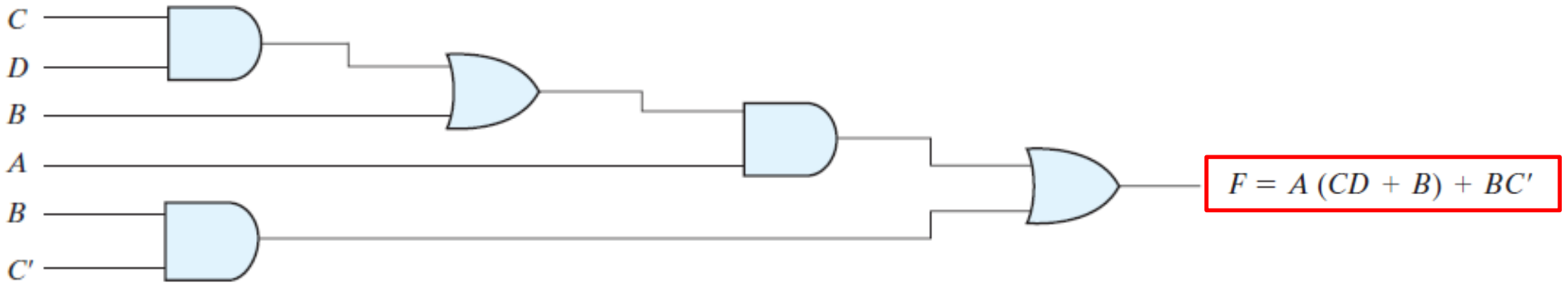


ยุบ NOT gate
รวมกับ z เป็น z'

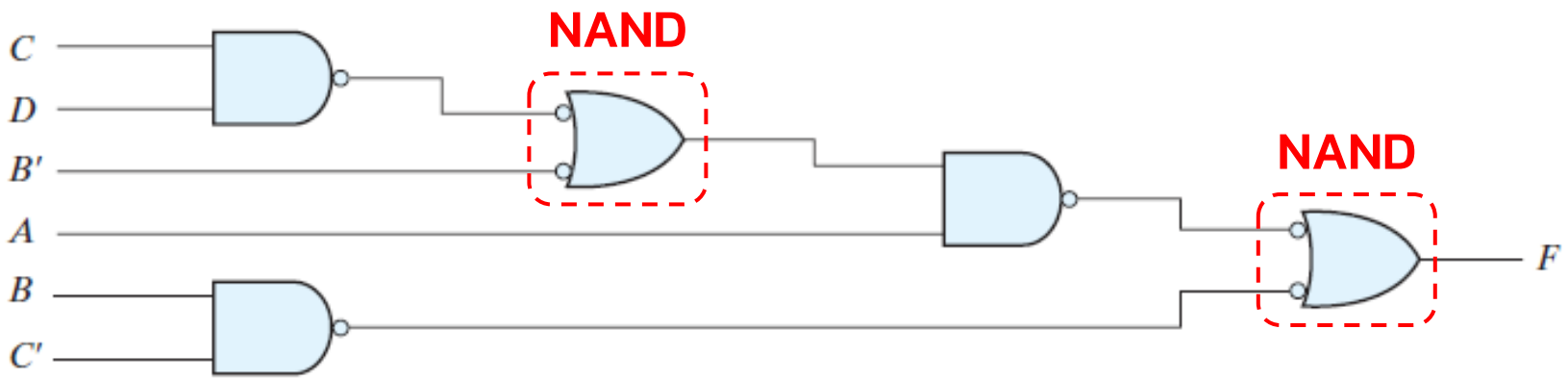
(c)

FIGURE 3.19
Solution to Example 3.9

Multilevel NAND Circuits

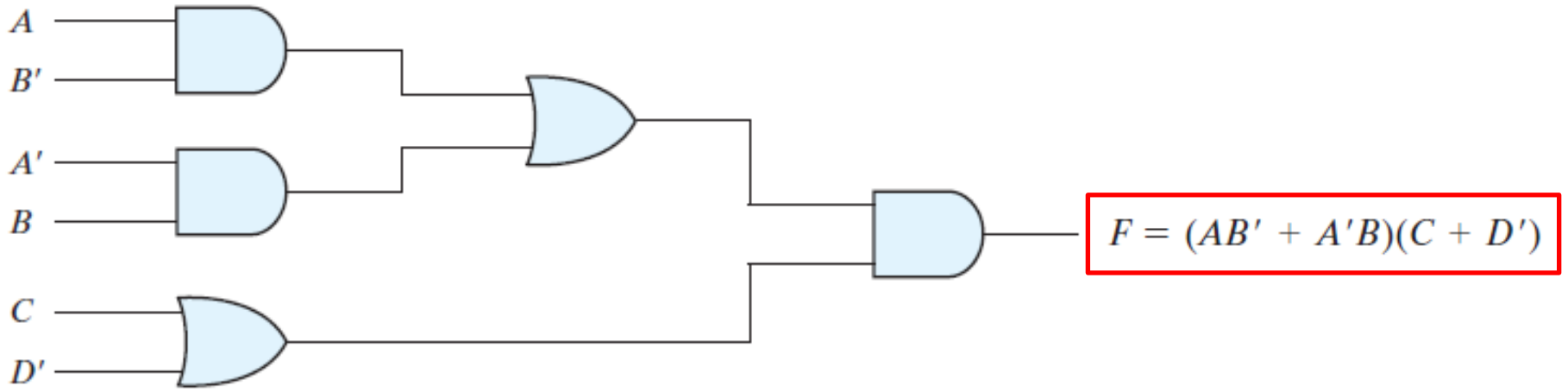


(a) AND–OR gates

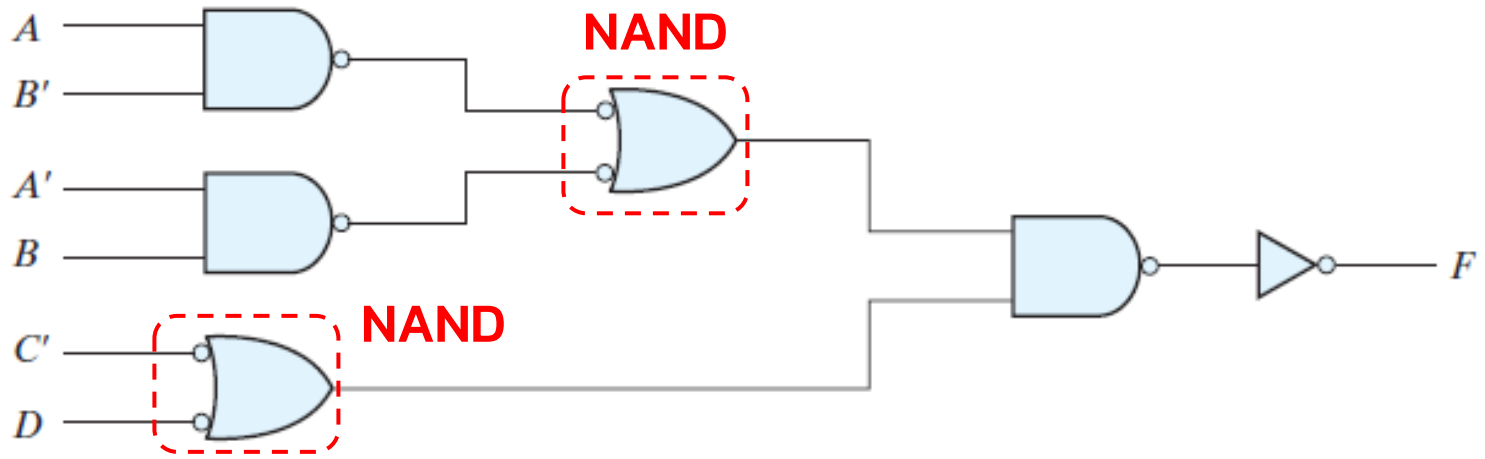


(b) NAND gates

FIGURE 3.20
Implementing $F = A(CD + B) + BC'$



(a) AND–OR gates



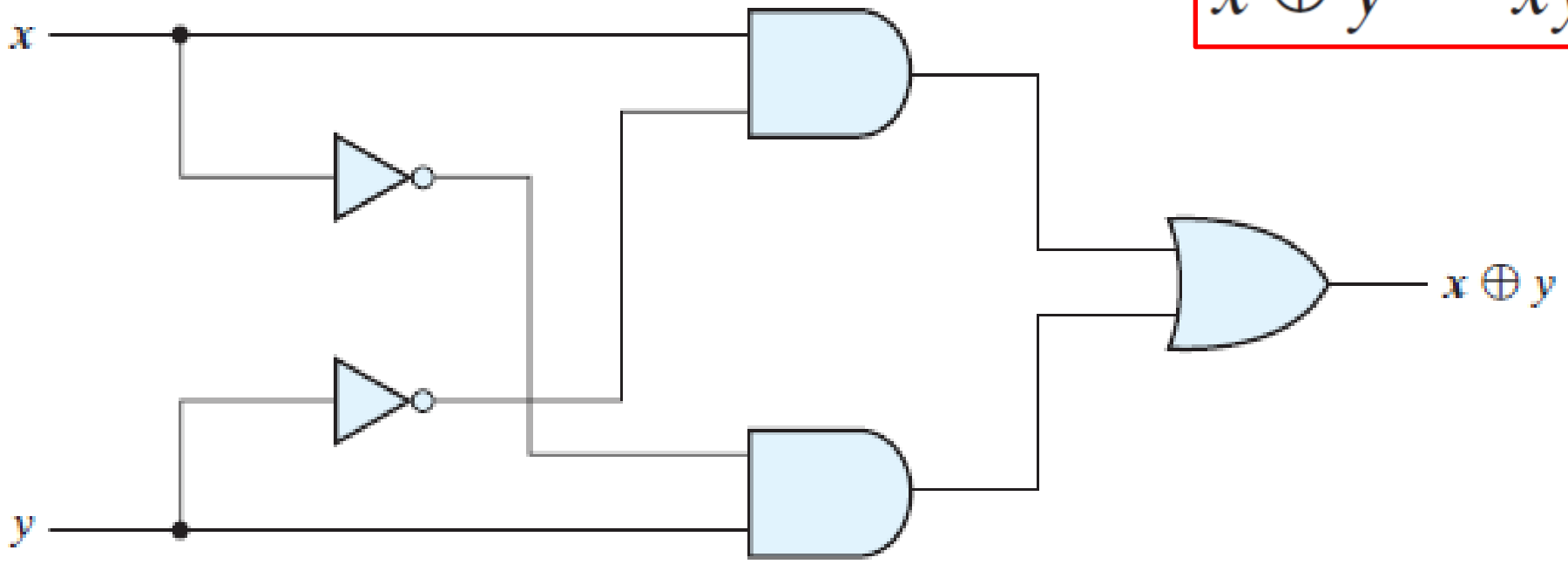
(b) NAND gates

FIGURE 3.21
Implementing $F = (AB' + A'B)(C + D')$

Exclusive-OR Function

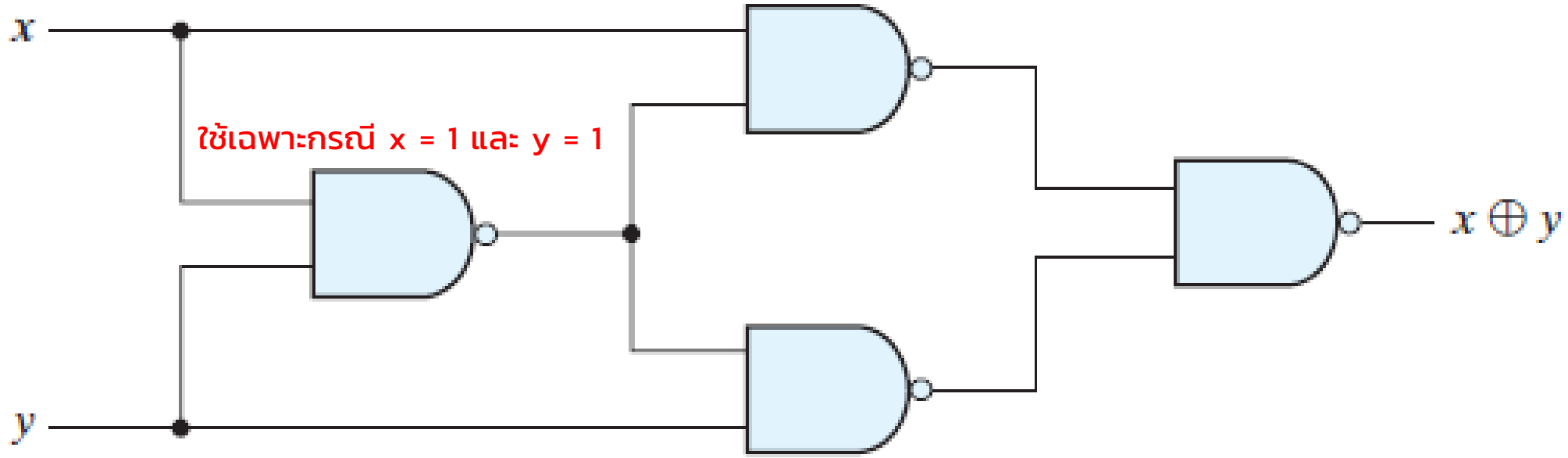
The exclusive-OR is equal to 1 if only x is equal to 1 or if only y is equal to 1 (i.e., x and y differ in value), but not when both are equal to 1 or when both are equal to 0.

$$x \oplus y = xy' + x'y$$



(a) Exclusive-OR with AND–OR–NOT gates

สร้าง XOR gate โดยใช้ NAND gate จำนวน 4 ตัว
ข้อนี้ยาก ให้ดูเฉลยเลย



(b) Exclusive-OR with NAND gates

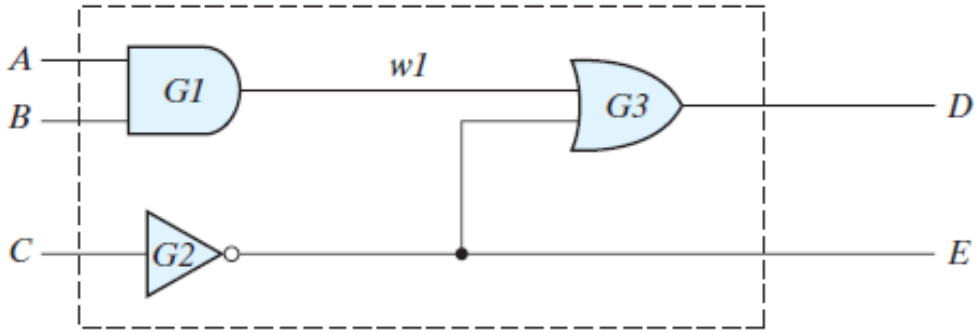
ประโยชน์ของ XOR gate คือ
ใช้เช็คจำนวนบิตที่เป็น 1 ในอินพุต ว่าเป็นเลขคี่ (odd) หรือ เลขคู่ (even)

(a) 3-input odd function

(b) 3-input even function

FIGURE 3.32
Logic diagram of odd and even functions

Hardware Description Language (HDL) แค่รู้เท่านั้น ไม่ต้องเขียนได้



HDL Example 3.1 (Combinational Logic Modeled with Primitives)

```
// Verilog model of circuit of Figure 3.35. IEEE 1364–1995 Syntax

module  Simple_Circuit (A, B, C, D, E);
  output        D, E;
  input         A, B, C;
  wire          w1;

  and           G1 (w1, A, B); // Optional gate instance name
  not           G2 (E, C);
  or            G3 (D, w1, E);
endmodule
```

Class — not Object

Gate Delays

HDL Example 3.2 (Gate-Level Model with Propagation Delays)

```
// Verilog model of simple circuit with propagation delay

module Simple_Circuit_prop_delay (A, B, C, D, E);
 output D, E;
 input  A, B, C;
 wire   w1;
                          Nanoseconds (default)
 and                  #(30) G1 (w1, A, B);
 not                  #(10) G2 (E, C);
 or                   #(20) G3 (D, w1, E);
endmodule
```

HDL Example 3.3 (Test Bench)

```
// Test bench for Simple_Circuit_prop_delay

module  t_Simple_Circuit_prop_delay;
 wire     D, E;
 reg      A, B, C;

Simple_Circuit_prop_delay M1 (A, B, C, D, E); // Instance name required

initial
  begin
    A = 1'b0; B = 1'b0; C = 1'b0;
    #100 A = 1'b1; B = 1'b1; C = 1'b1;
  end

 initial #200 $finish;
endmodule
```

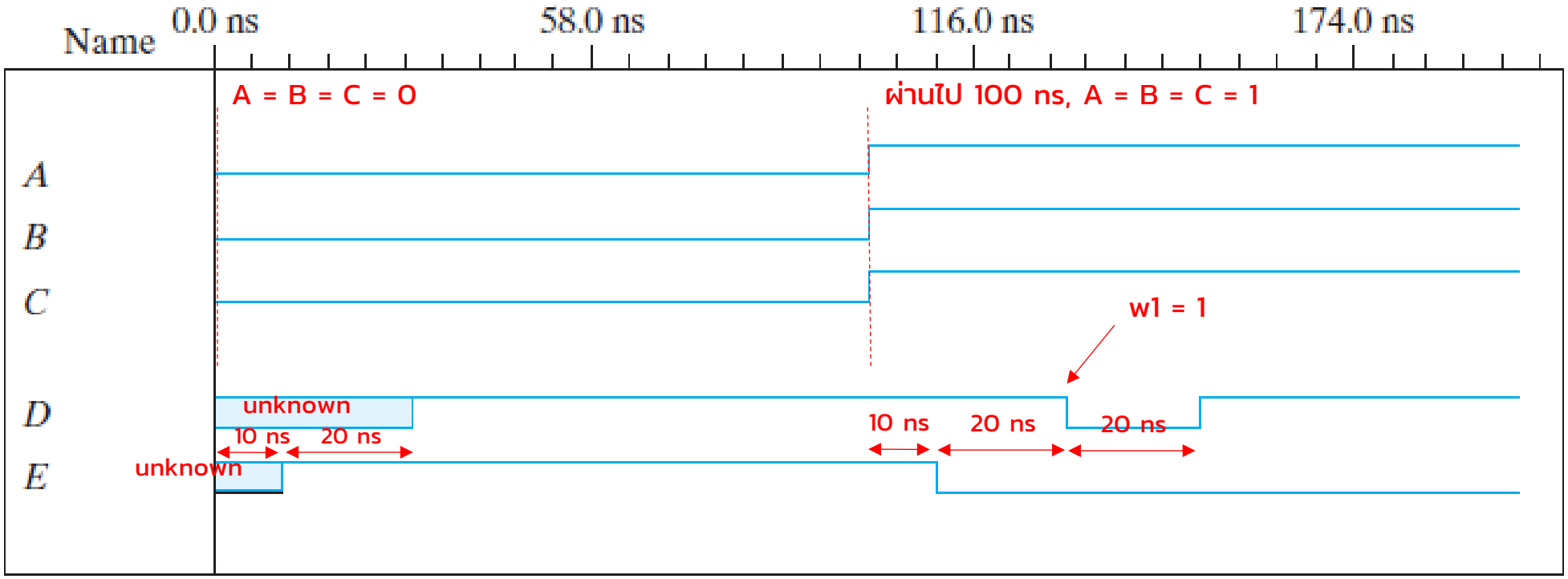
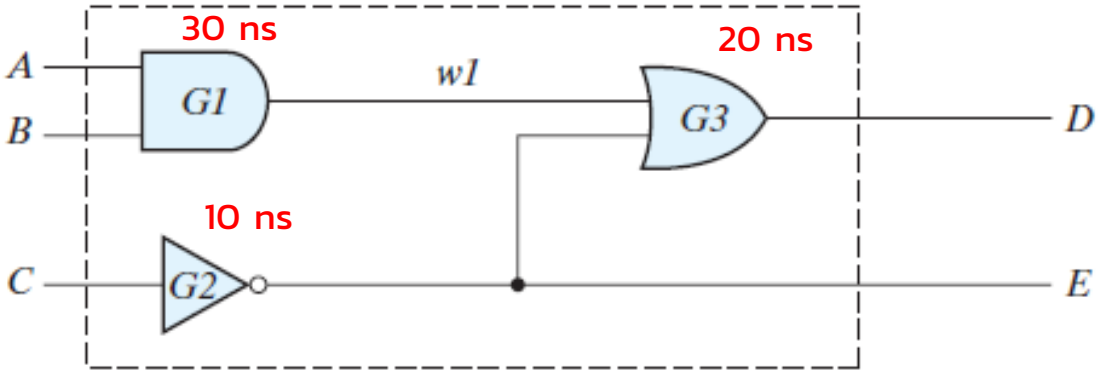


FIGURE 3.36
Simulation output of HDL Example 3.3



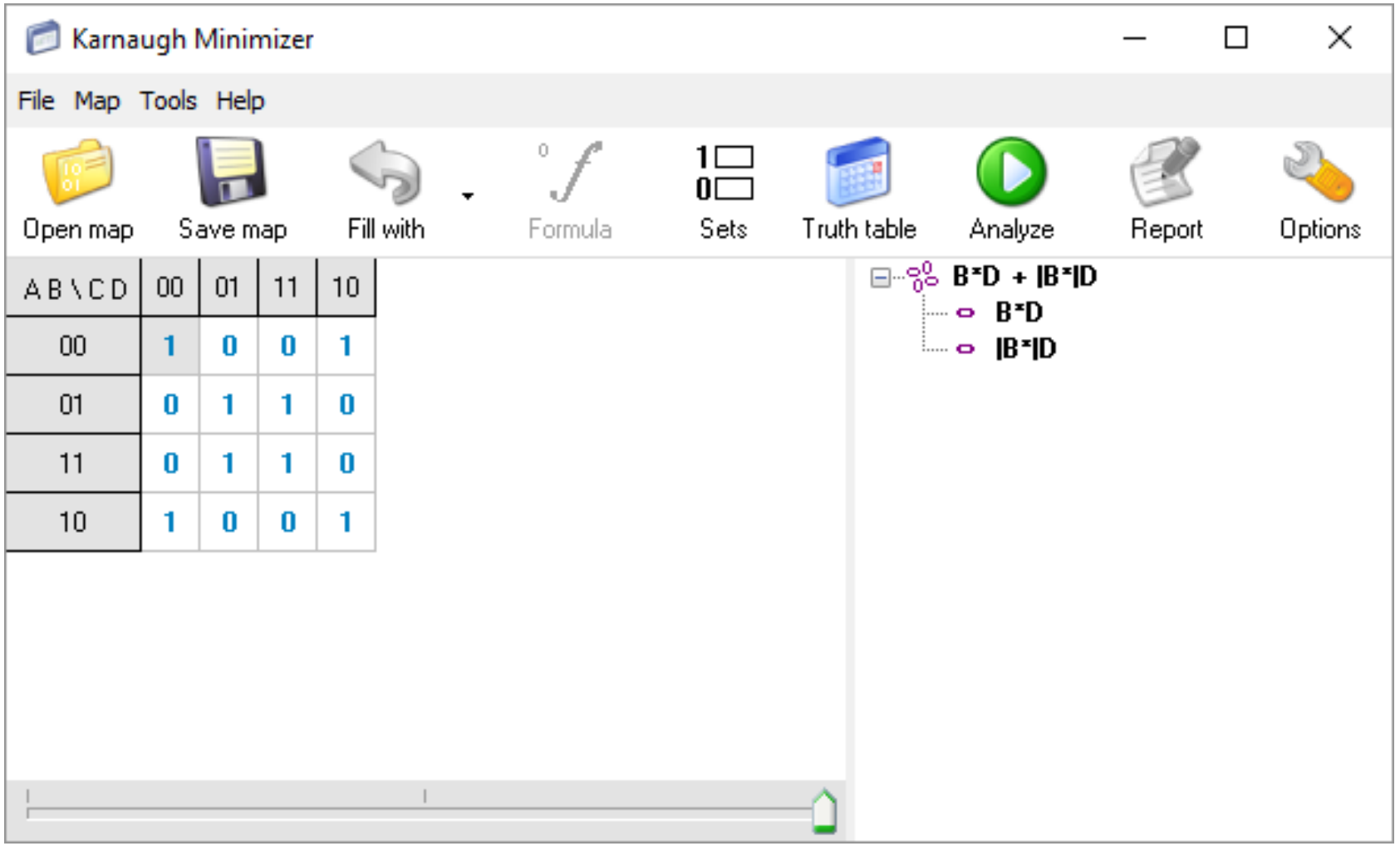
HDL Example 3.4 (Combinational Logic Modeled with Boolean Equations)

```
// Verilog model: Circuit with Boolean expressions

module Circuit_Boolean_CA (E, F, A, B, C, D);
  output      E, F;
  input       A, B, C, D;

  assign E = A || (B && C) || ((!B) && D);
  assign F = ((!B) && C) || (B && (!C) && (!D));
endmodule
```

คอมไพเลอร์สังเคราะห์ (synthesize) วงจรให้ตามเทคโนโลยีที่ใช้ (AND, OR, NAND, etc.)

Karnaugh Minimizer 2.0

https://karnaugh-minimizer.en.softonic.com (ติดตั้งซอฟต์แวร์จากอินเทอร์เน็ต ระวัง virus computer ด้วยนะครับ)



PROBLEMS

(Answers to problems marked with * appear at the end of the text.)

3.1* Simplify the following Boolean functions, using three-variable maps:

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

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

(c) $F(x, y, z) = \Sigma(0, 1, 2, 3, 5)$

(d) $F(x, y, z) = \Sigma(1, 2, 3, 7)$

3.2 Simplify the following Boolean functions, using three-variable maps:

(a)* $F(x, y, z) = \Sigma(0, 1, 5, 7)$

(b)* $F(x, y, z) = \Sigma(1, 2, 3, 6, 7)$

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

(d) $F(x, y, z) = \Sigma(1, 2, 3, 5, 6, 7)$

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

(f) $F(x, y, z) = \Sigma(3, 4, 5, 6, 7)$

3.3* Simplify the following Boolean expressions, using three-variable maps:

(a)* $xy + x'y'z' + x'yz'$

(b)* $x'y' + yz + x'yz'$

(c)* $F(x, y, z) = x'y + yz' + y'z'$

(d) $F(x, y, z) = x'yz + xy'z' + xy'z$