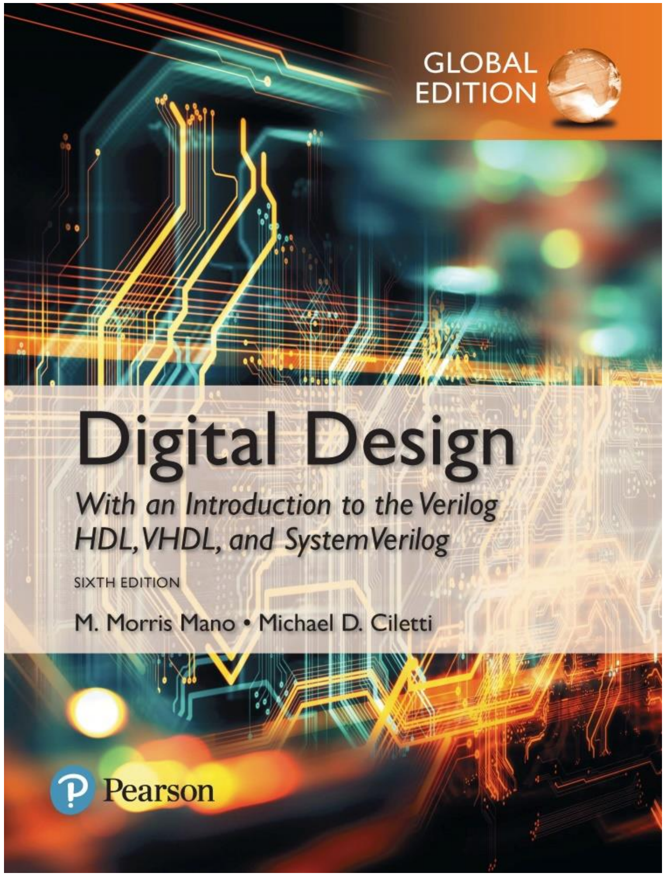
Digital Design

With an Introduction to the Verilog HDL, VHDL, and SystemVerilog

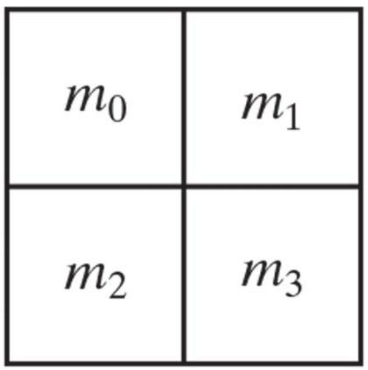
6^{th} Edition, Global Edition



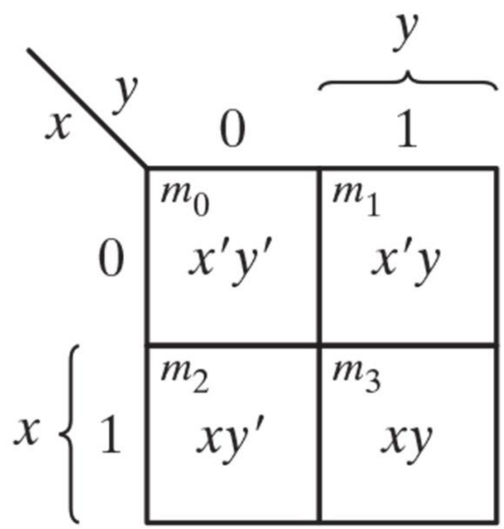
Chapter 03

Gate-Level Minimization

Two-variable K-map.

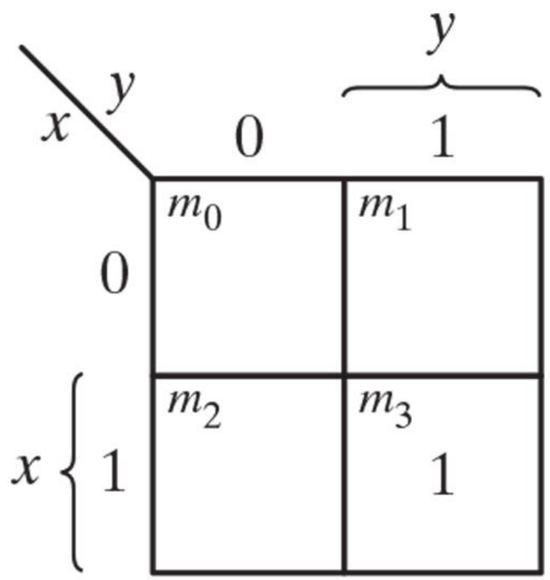


(a)

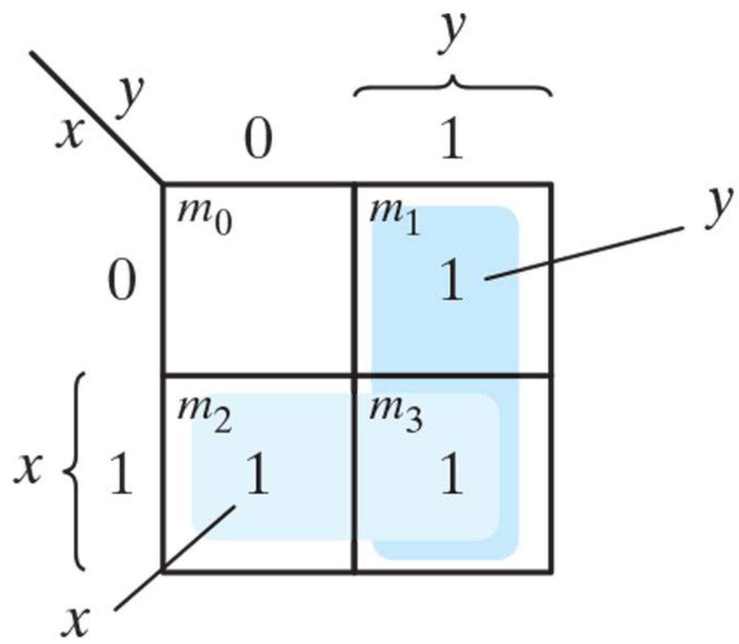


(b)

Representation of functions in the K-map.

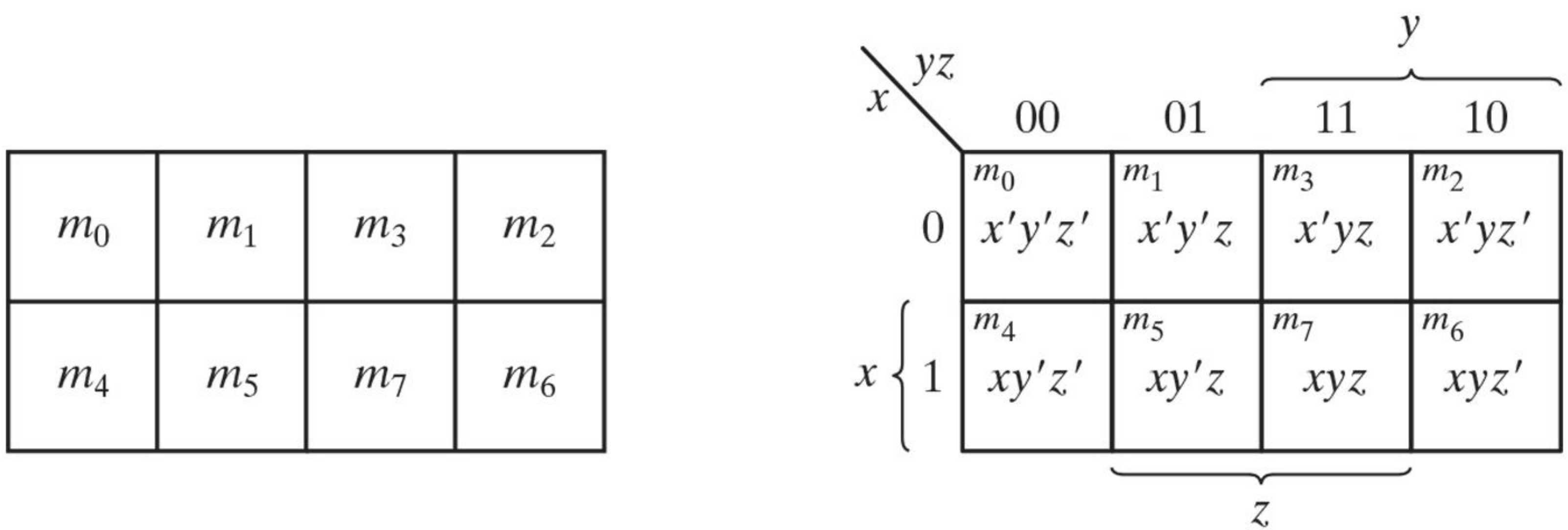


(a) xy



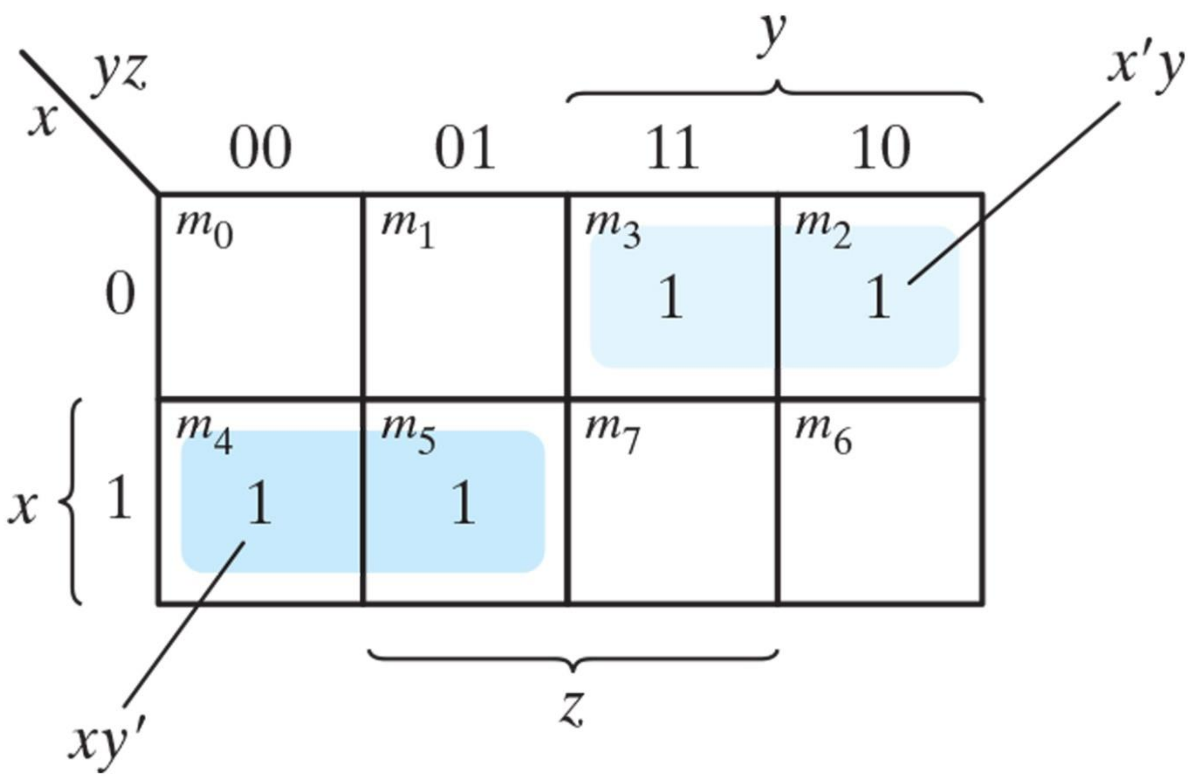
(b) $x + y$

Three-variable K-map.

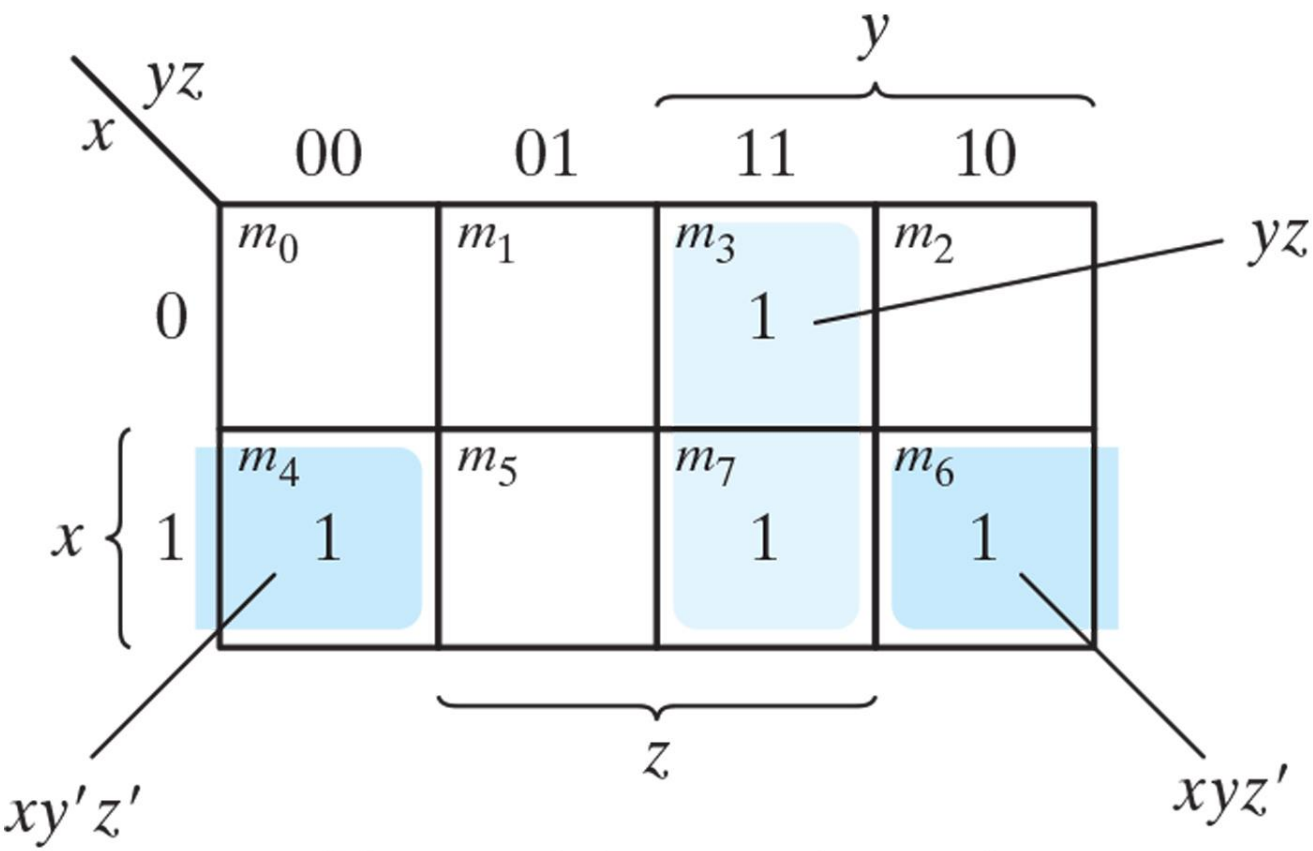


(a) (b)

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

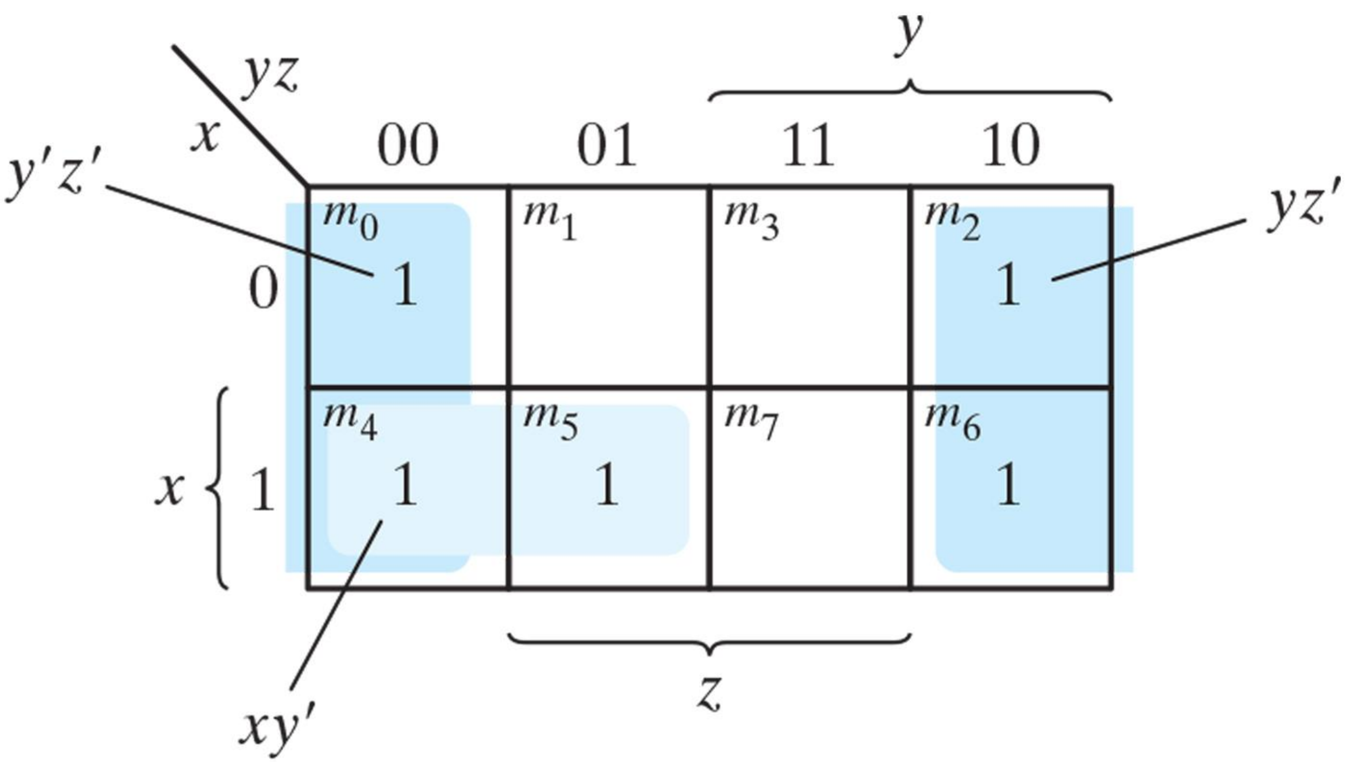


Map for Example 3.2, $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$.



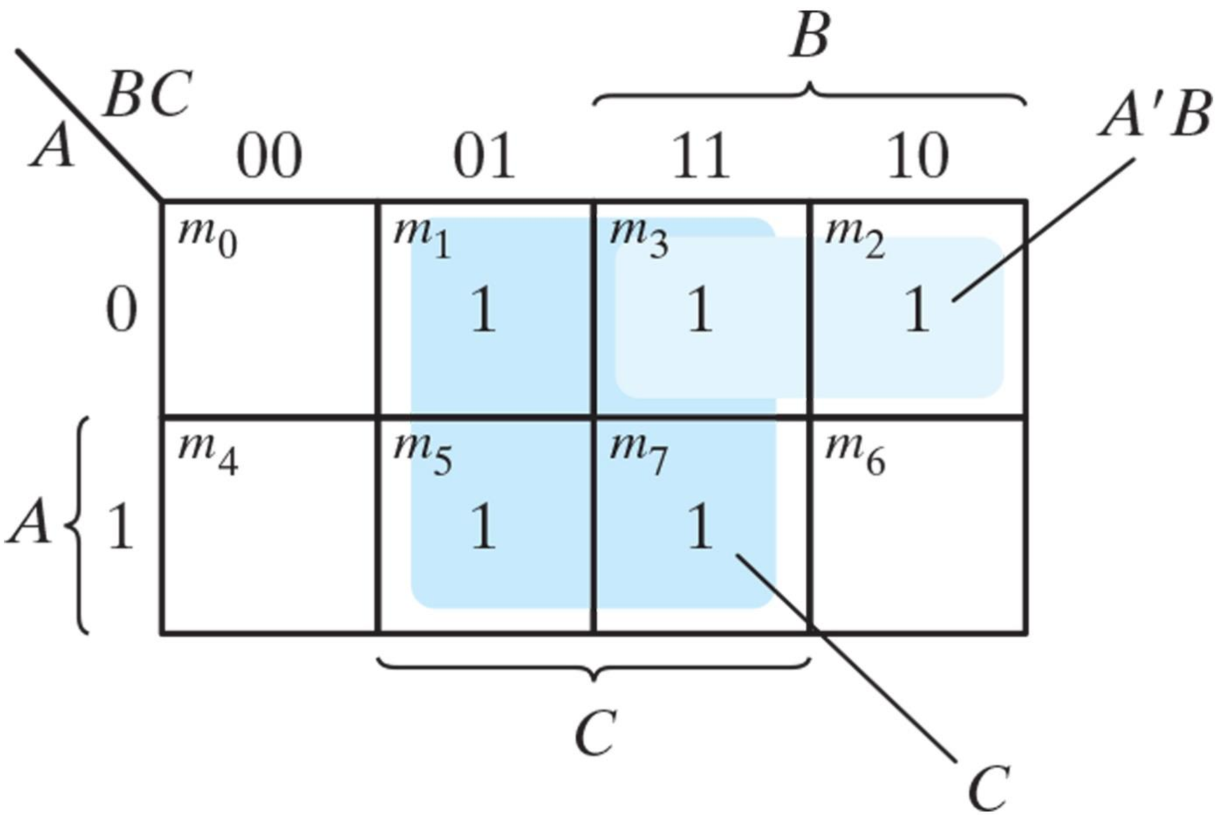
$Note: xy'z' + xyz' = xz'$

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



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

Map of Example 3.4, *A’C* + *A’B* + *AB’C* + *BC* = *C* + *A’B*.



Pearson

Four-variable map.

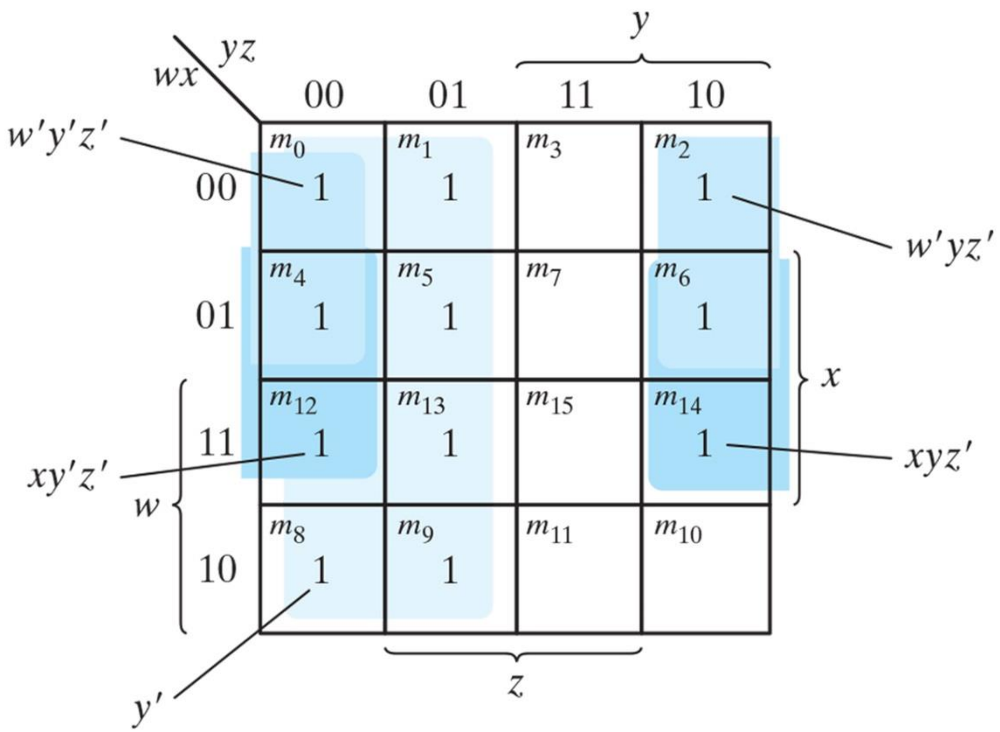
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)

				y		
	$wx \backslash yz$	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 $w'xyz'$	x
w	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} $wx'yz'$	
			z			

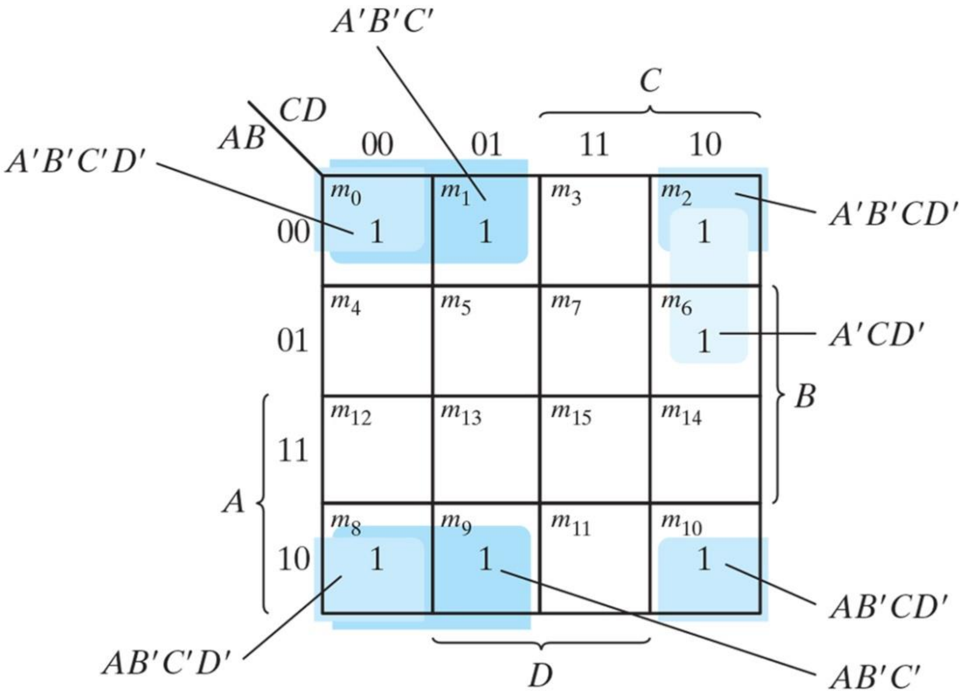
(b)

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



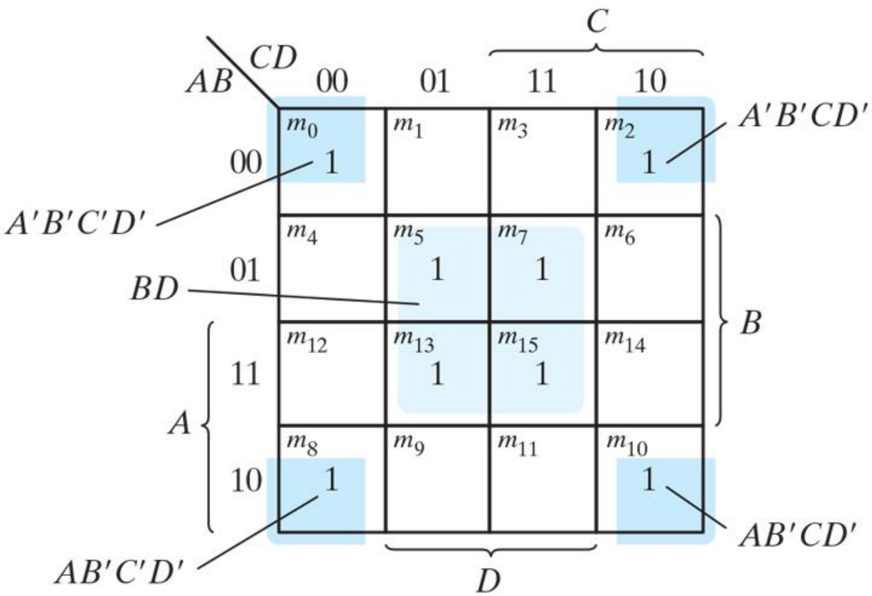
Note: $w'y'z' + w'yz' = w'z'$
$xy'z' + xyz' = xz'$

Map for Example 3.6, *A'B'C' + B'CD' + A'BCD' + AB'C = B'D' + B'C' + A'CD'.*



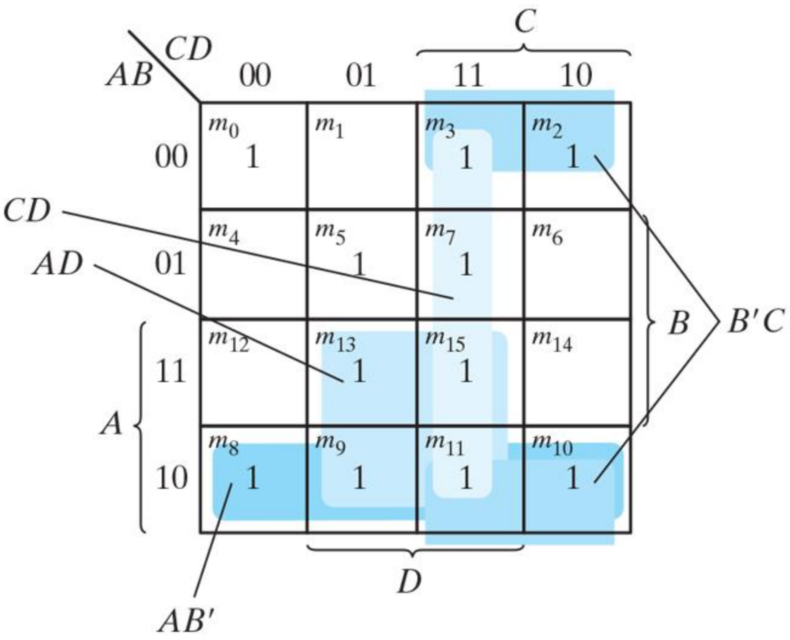
Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$
$A'B'C' + AB'C' = B'C'$

Simplification using prime implicants.



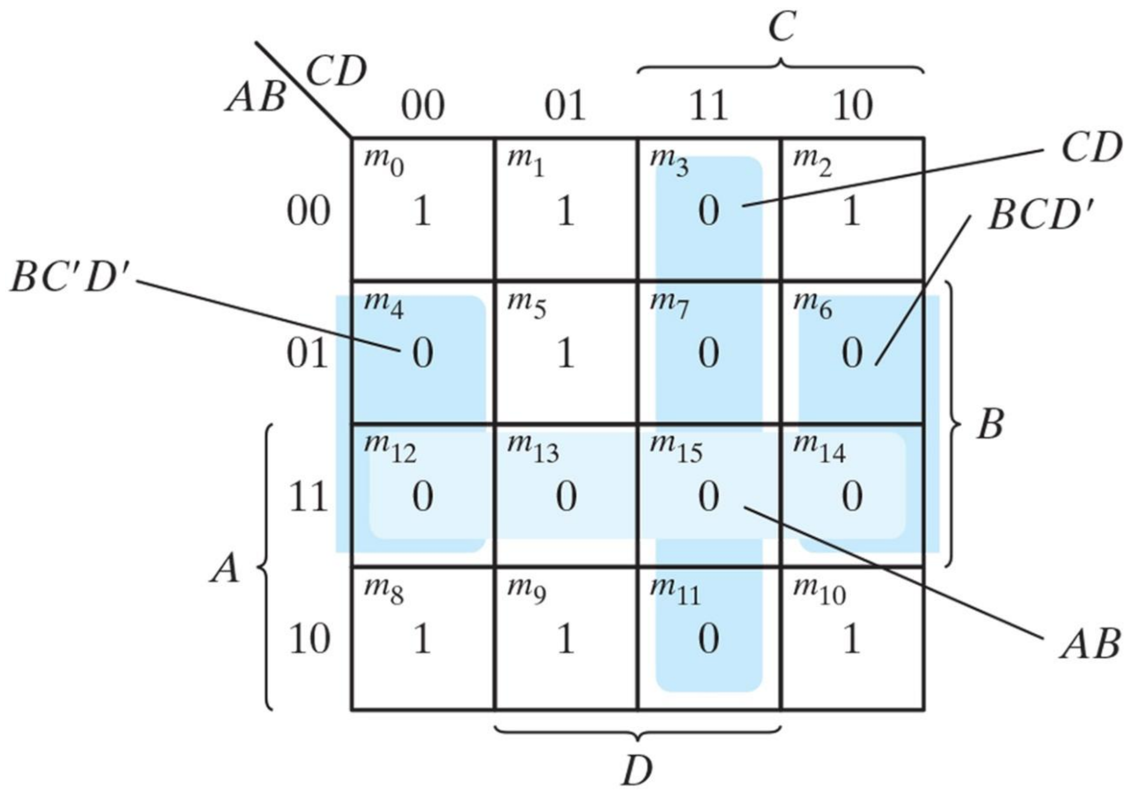
Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$

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



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

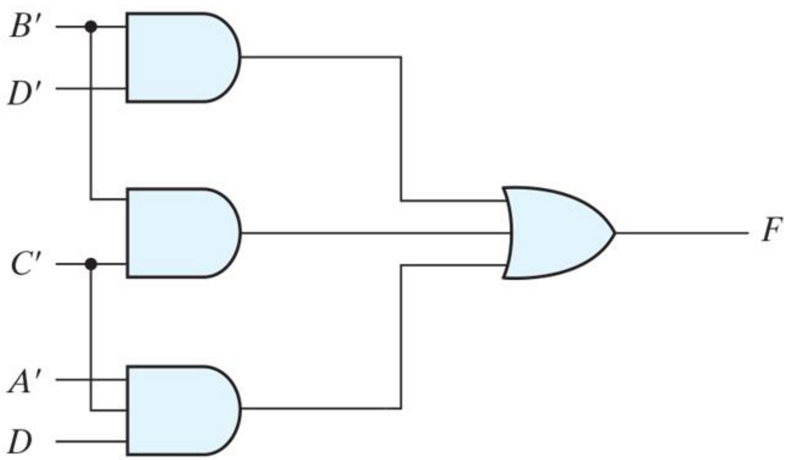
Map for Example 3.7, $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10) = BD + BC + ACD = (A' + B')(C' + D')(B' + D)$.



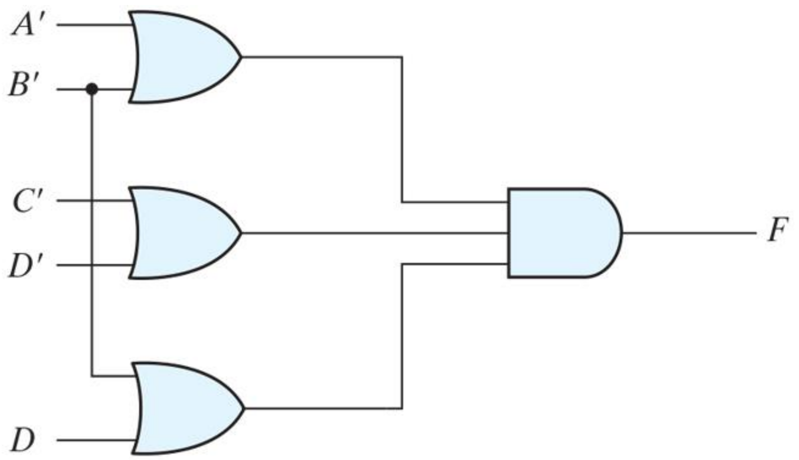
Note: $BC'D' + BCD' = BD'$

Pearson

Gate implementations of the function of Example 3.7.



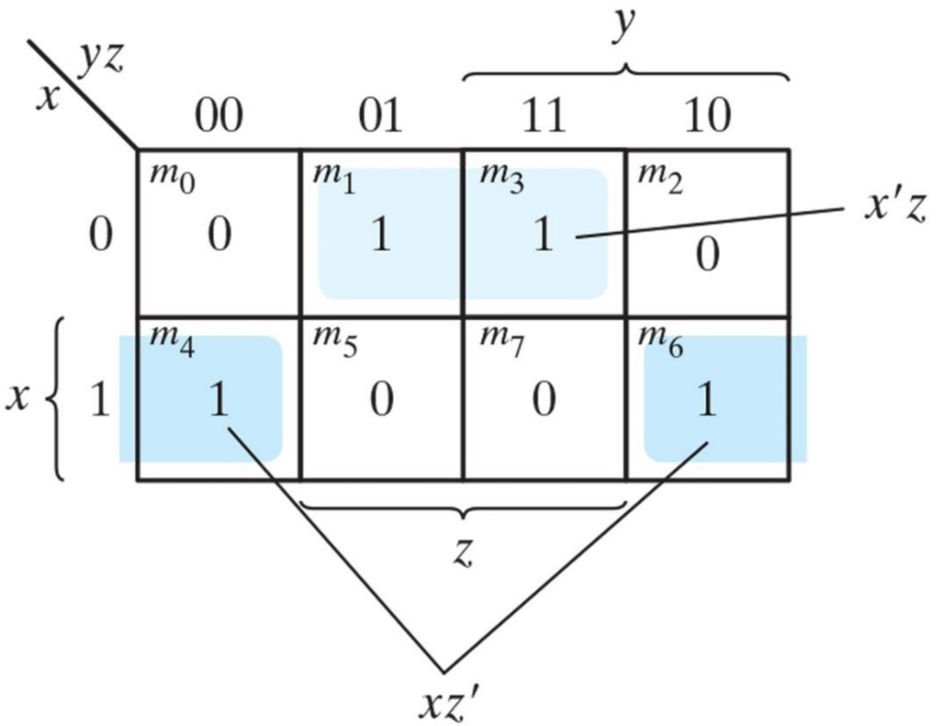
(a) $F = B'D' + B'C' + A'C'D$



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

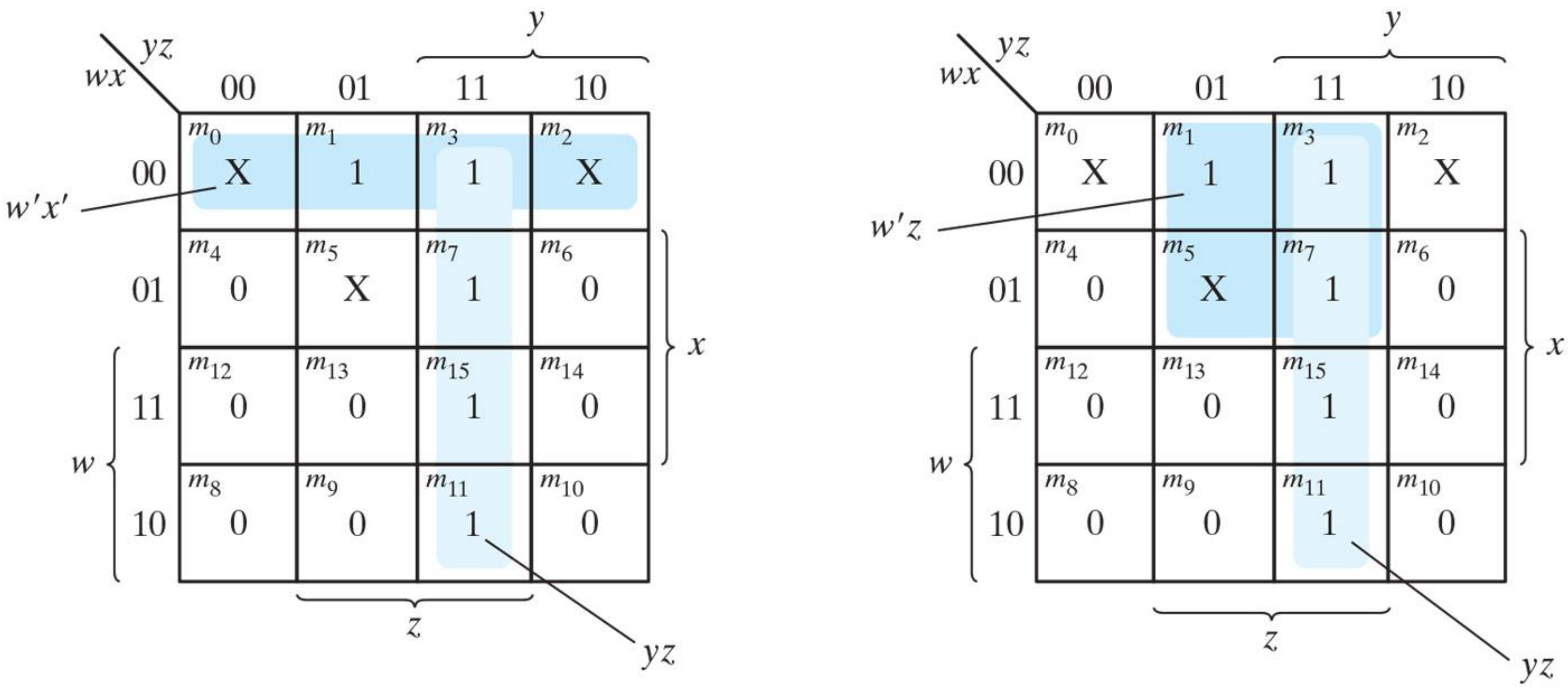
Map for the function of Table 3.1.

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



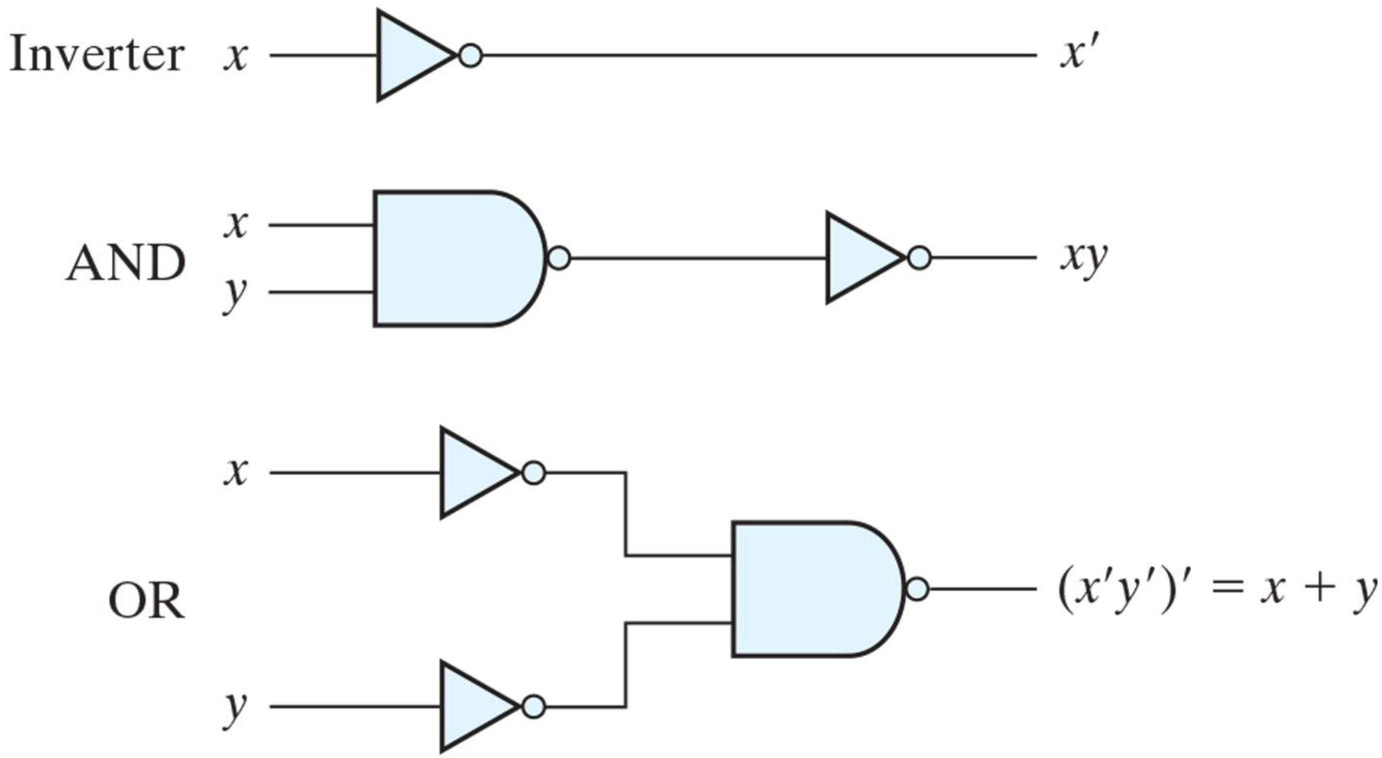
Example with don’t-care conditions.

Simplify the Boolean function, F(w, x, y, z) = Σ(1,3,7,11,15)
Don’t-care conditions, d(w, x, y, z) = Σ(0, 2, 5)

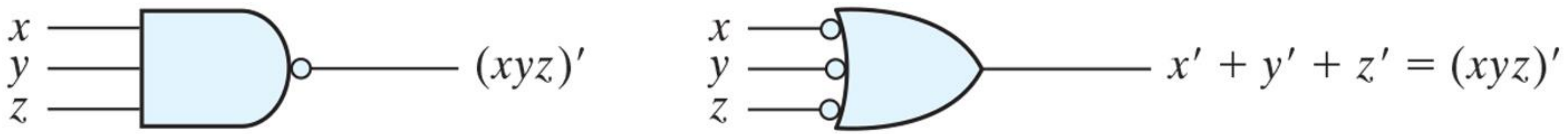


(a) $F = yz + w'x'$ (b) $F = yz + w'z$

Logic operations with NAND gates.



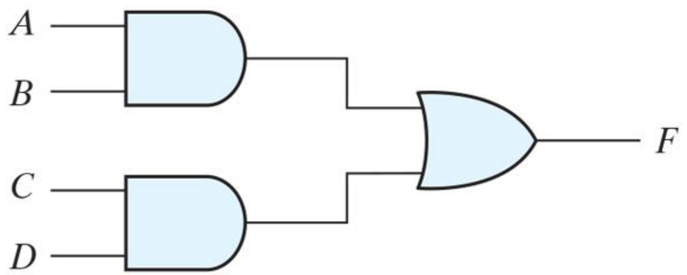
Two graphic symbols for a three-input NAND gate.



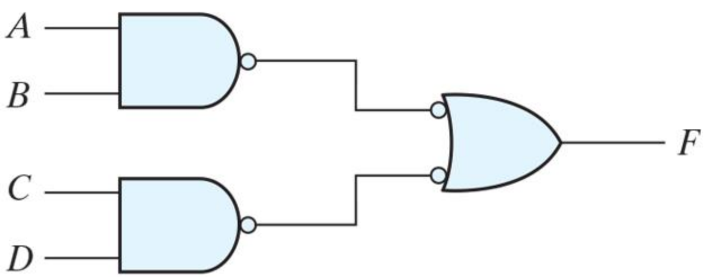
(a) AND-invert

(b) Invert-OR

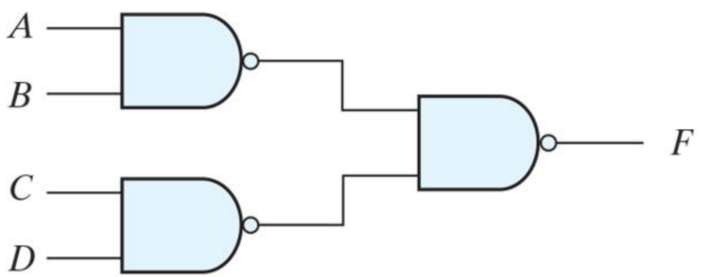
Three ways to implement *F* = *AB* + *CD*.



(a)

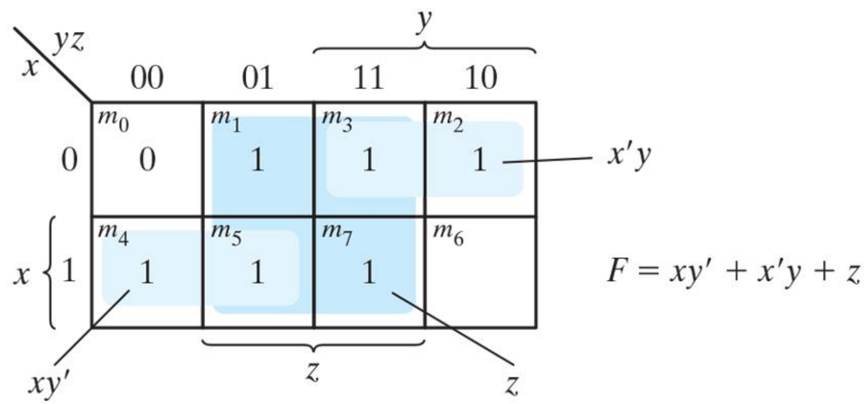


(b)

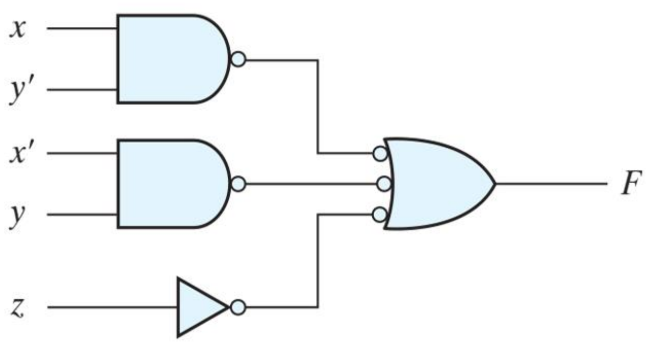


(c)

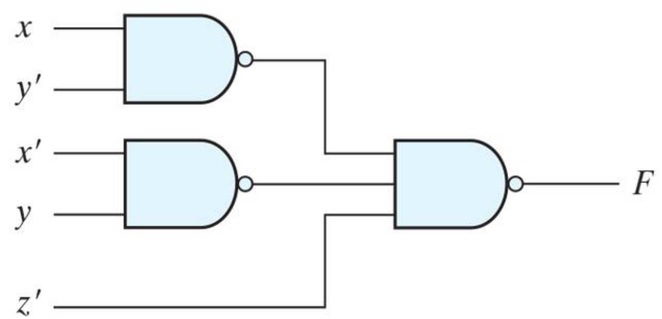
Solution to Example 3.9.



(a)

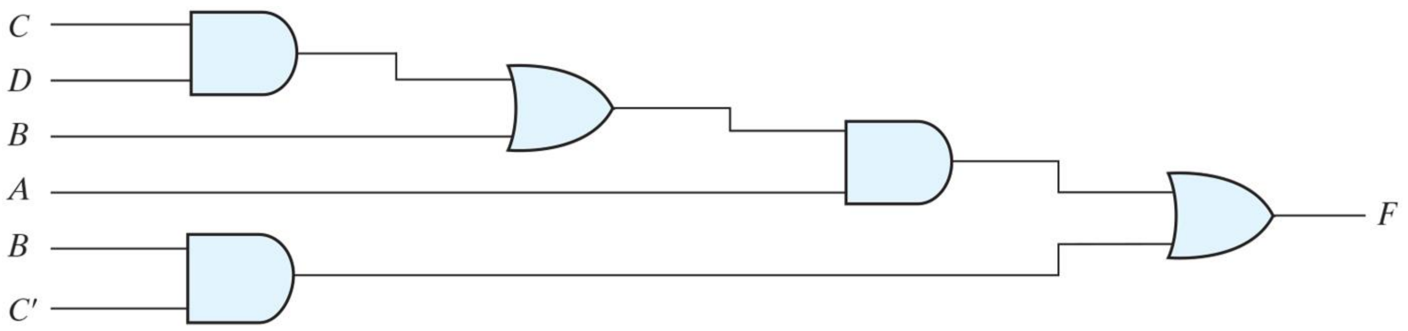


(b)

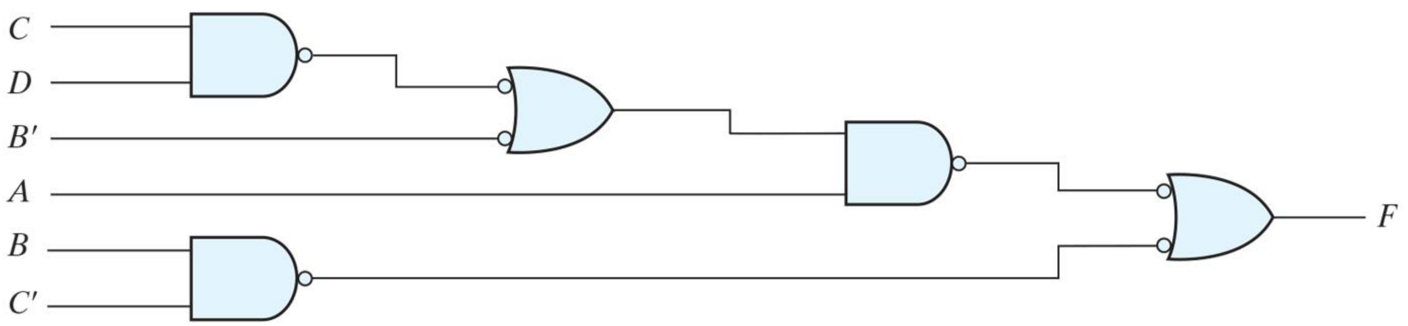


(c)

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

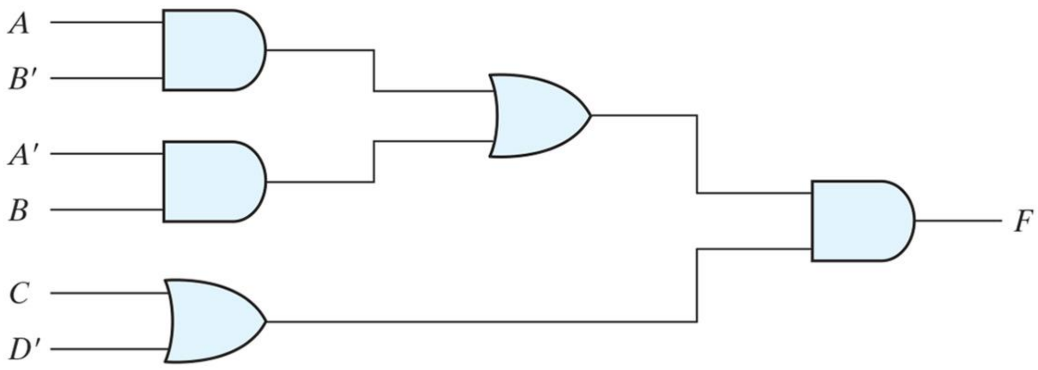


(a) AND–OR gates

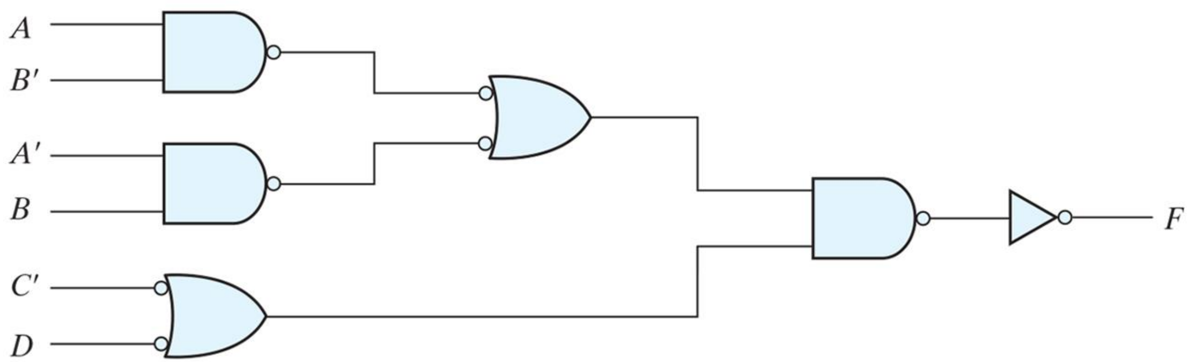


(b) NAND gates

Implementing *F* = (*AB'* + *A'B*)(*C* + *D'*).

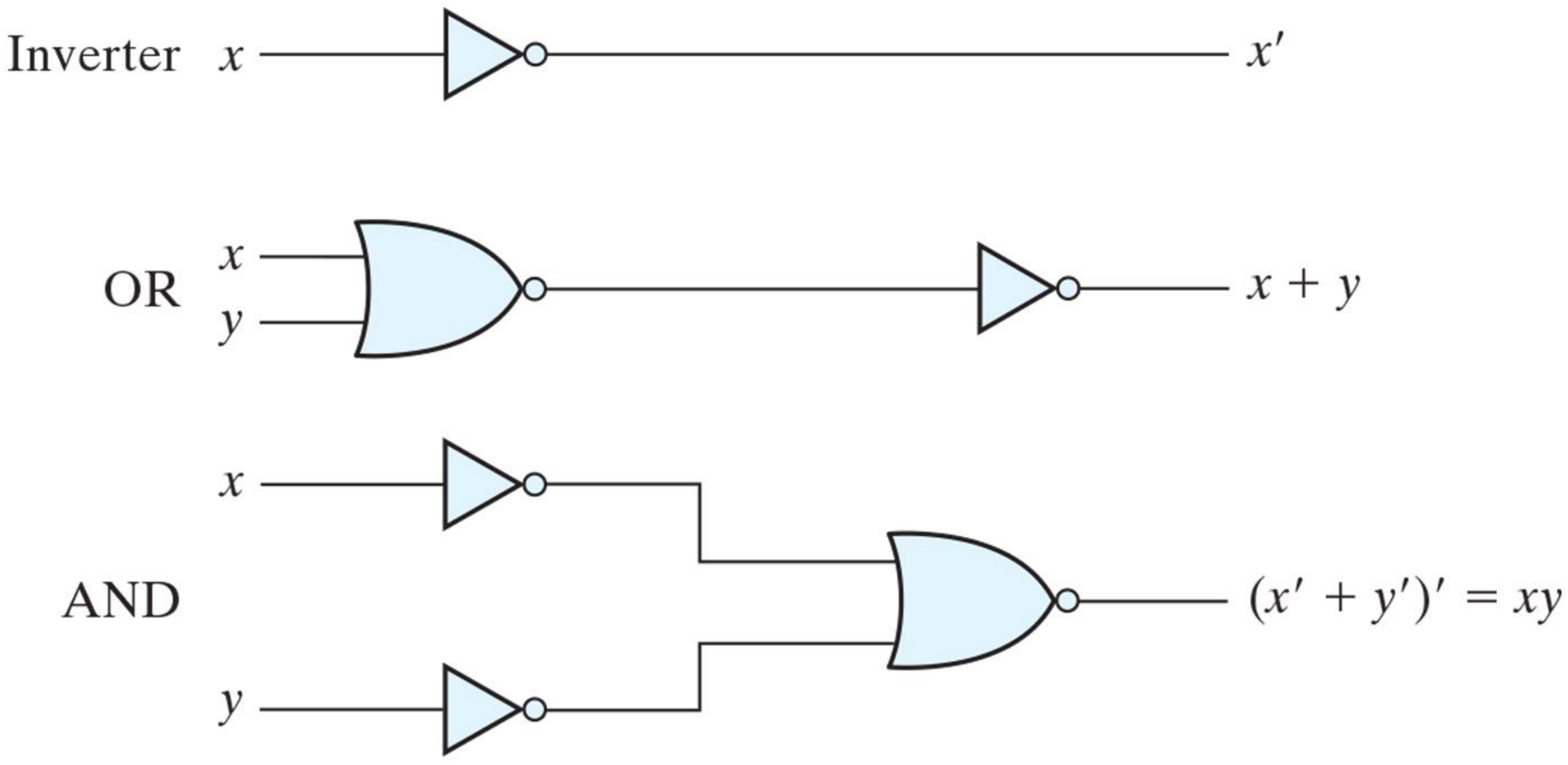


(a) AND–OR gates

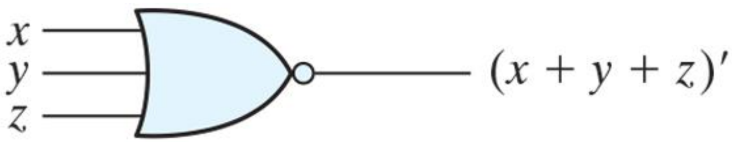


(b) NAND gates

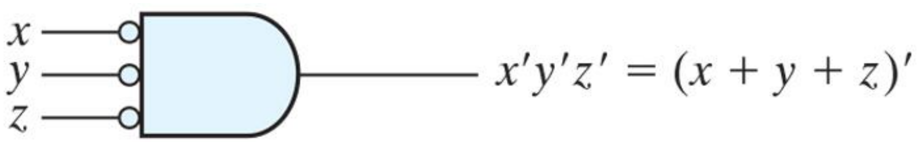
Logic operations with NOR gates.



Two graphic symbols for the NOR gate.

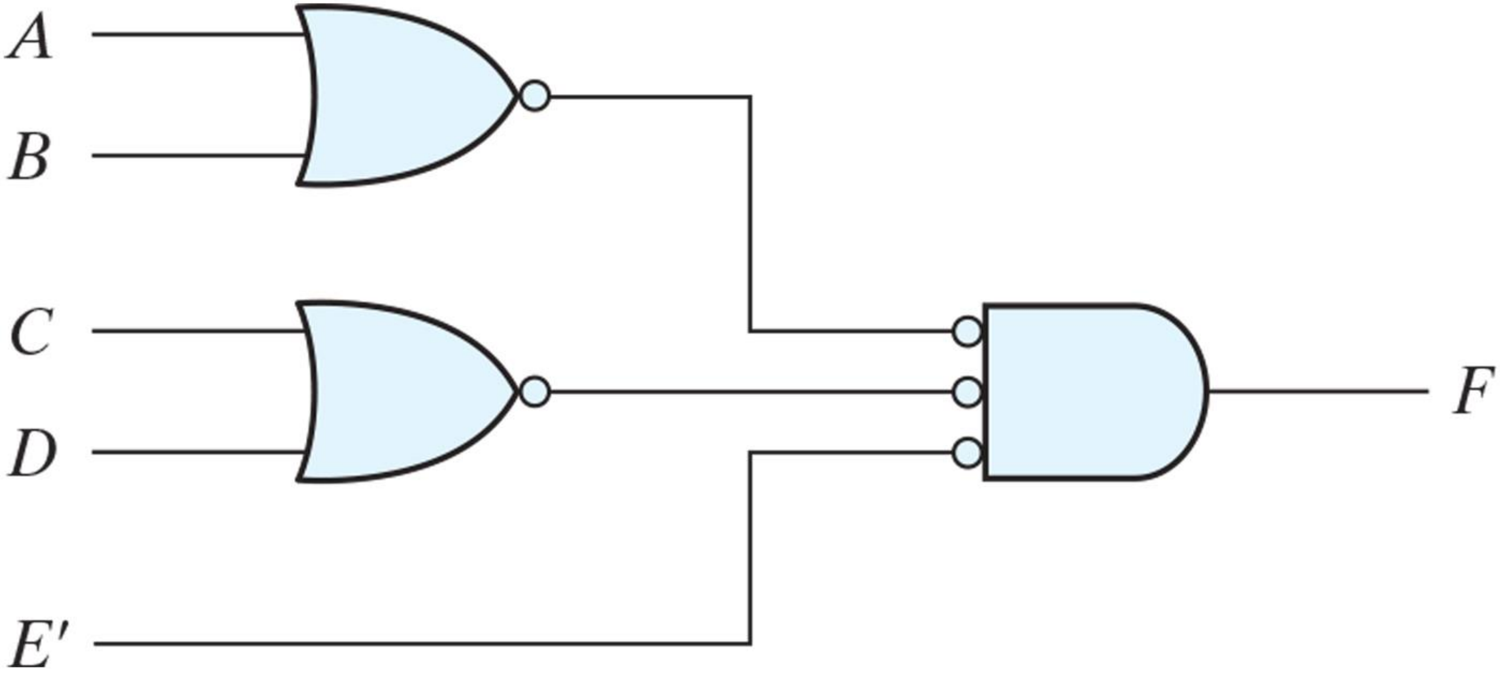


(a) OR-invert

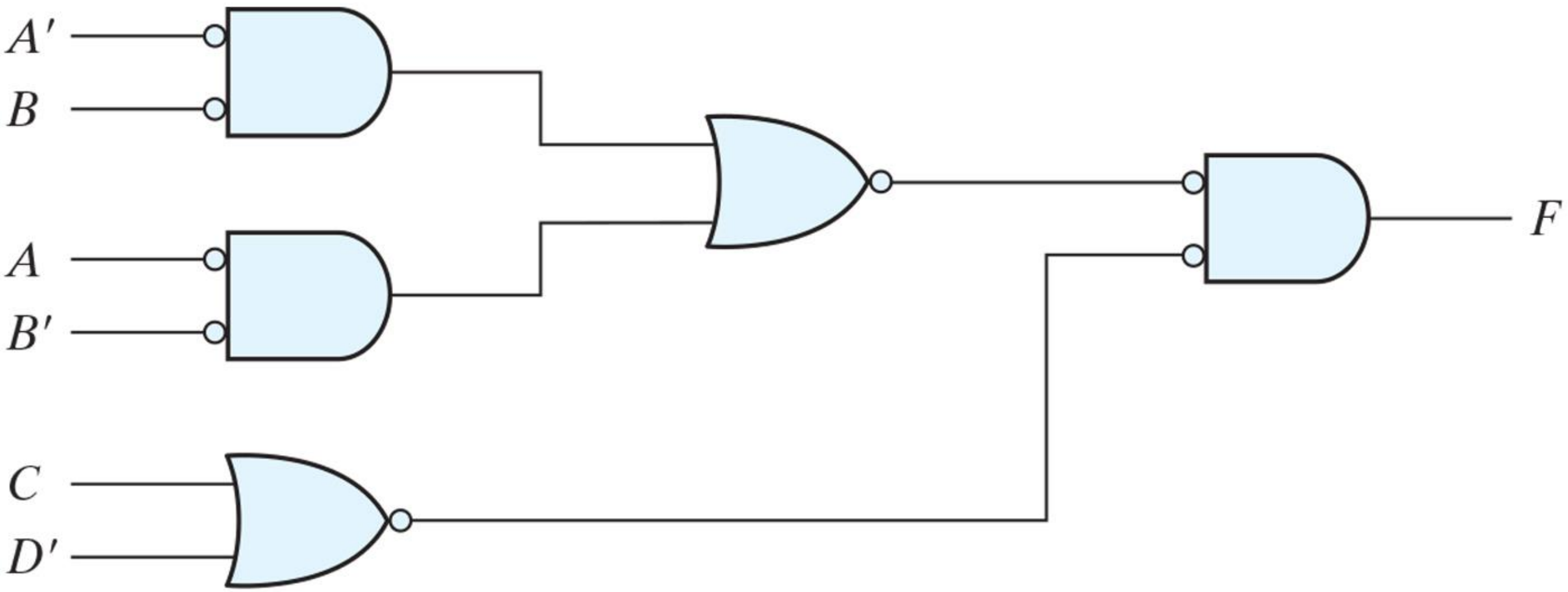


(b) Invert-AND

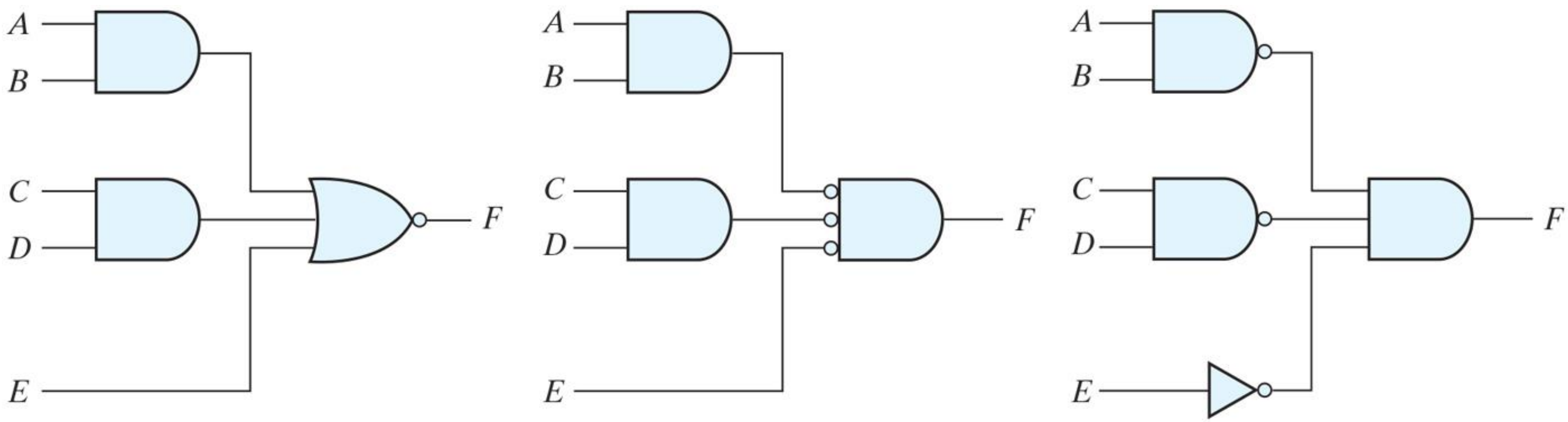
Implementing *F* = (*A* + *B*)(*C* + *D*)*E*.



Implementing *F = (AB' + A'B)(C + D')* with NOR gates.

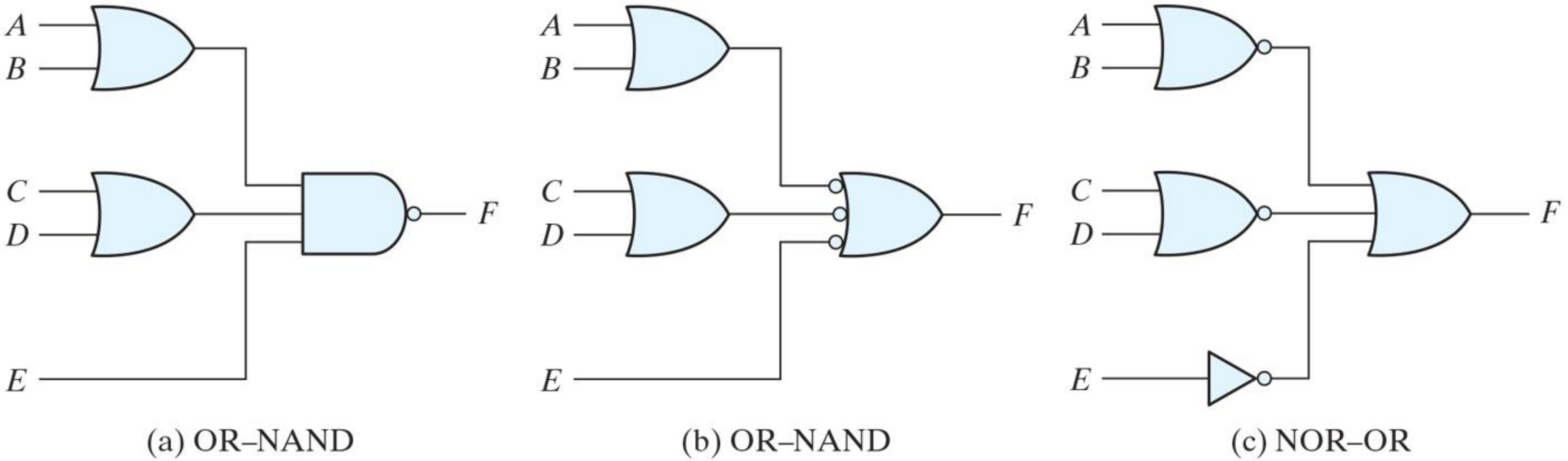


AND–OR–INVERT circuits, *F* = (*AB* + *CD* + *E*)’.



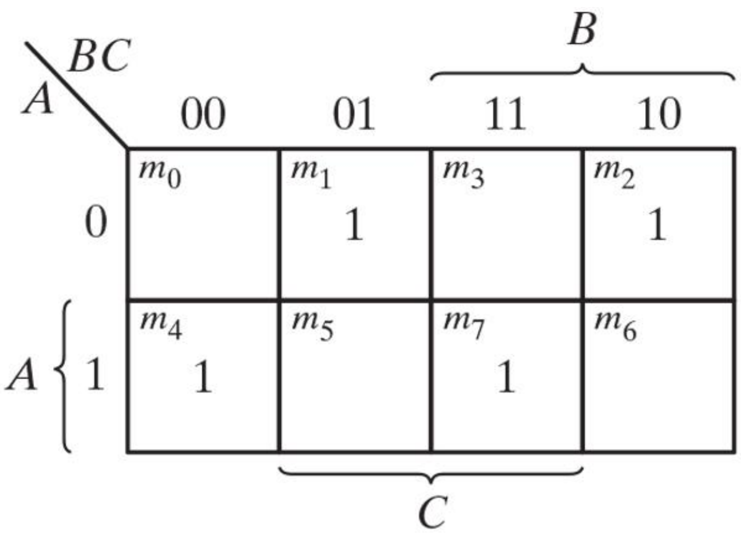
(a) AND–NOR (b) AND–NOR (c) NAND–AND

OR–AND–INVERT circuits, $F = [(A + B)(C + D)E]'$.

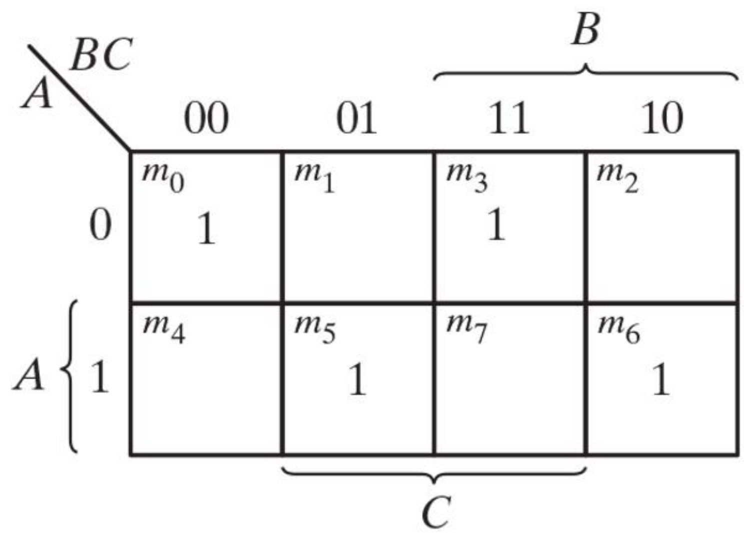


Map for a three-variable exclusive-OR function.

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



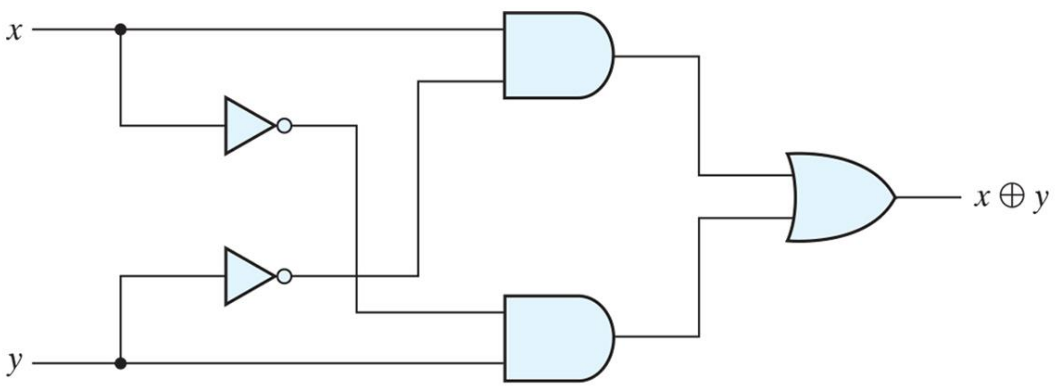
(a) Odd function $F = A \oplus B \oplus C$



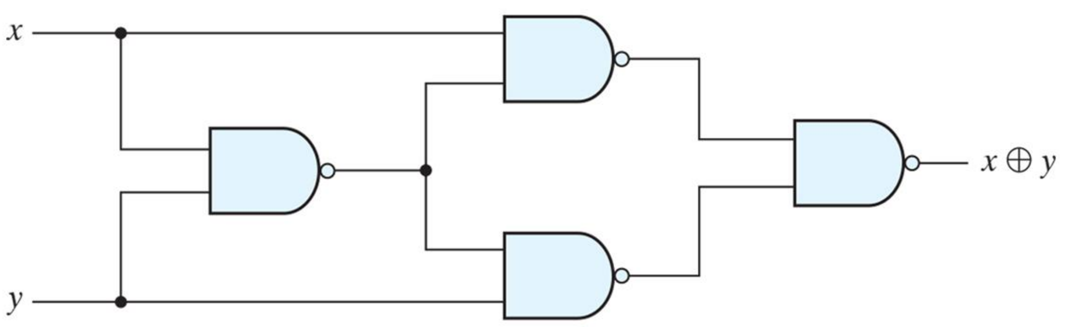
(b) Even function $F = (A \oplus B \oplus C)'$

Pearson

Logic diagrams for exclusive-OR implementations.



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

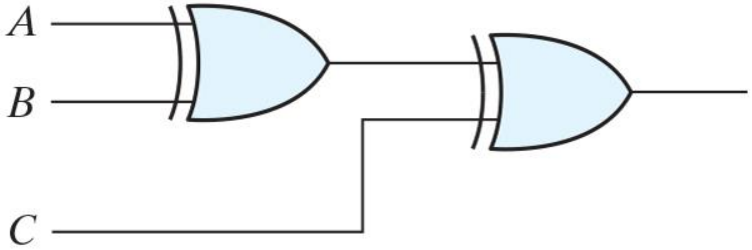


(b) Exclusive-OR with NAND gates

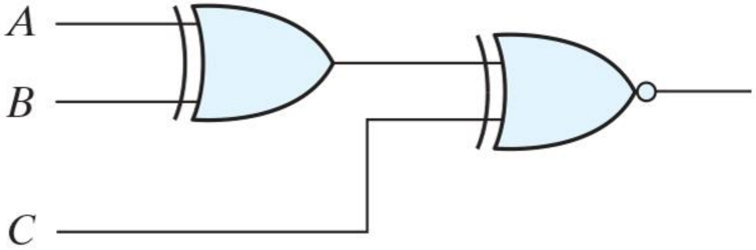
Even-Parity-Generator Truth Table.

Three-Bit Message			Parity Bit
x	*y*	*z*	*P*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Logic diagram of odd and even functions.



(a) 3-input odd function

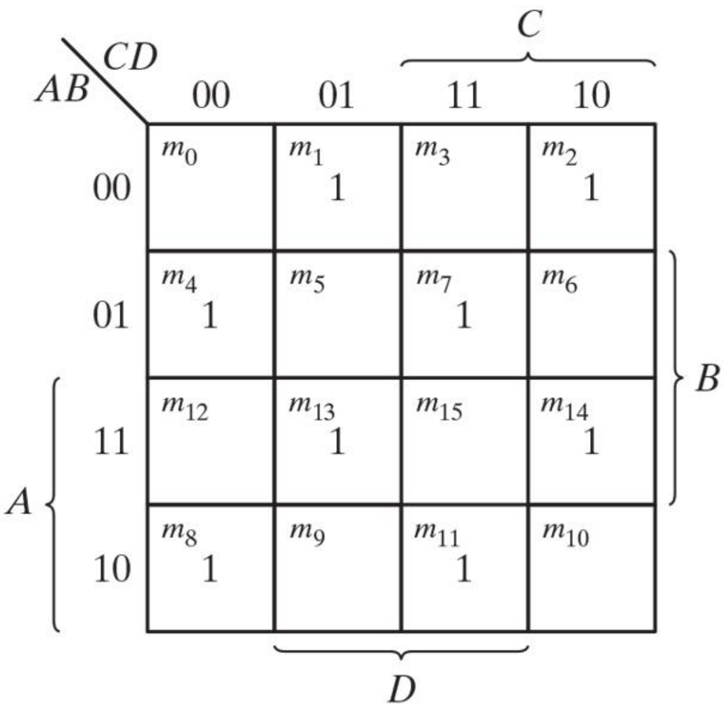


(b) 3-input even function

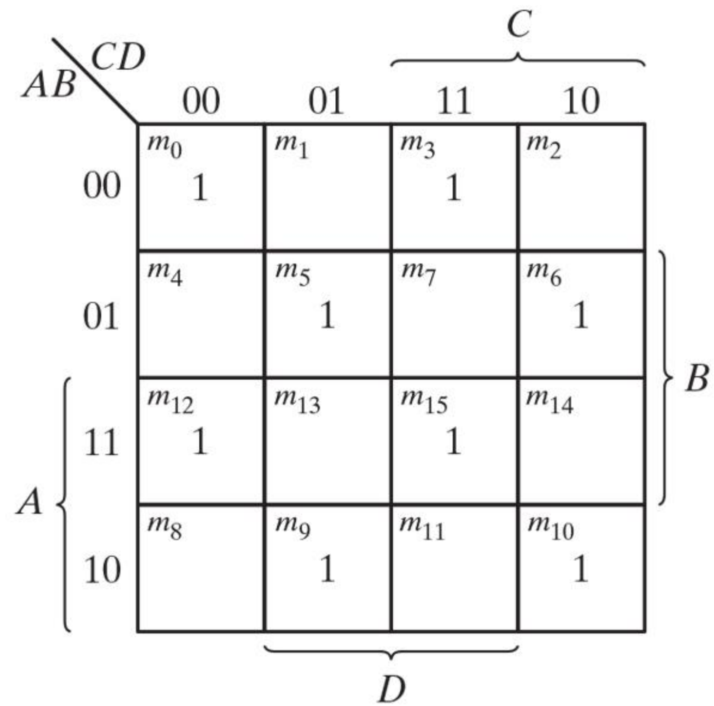
Even-Parity-Checker Truth Table.

Four Bits Received				Parity Error Check
x	*y*	*z*	*P*	*C*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Map for a four-variable exclusive-OR function.



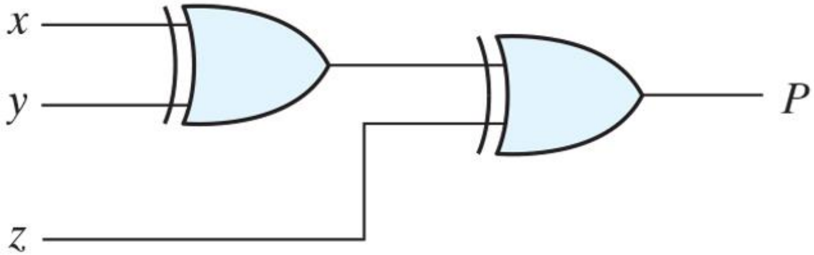
(a) Odd function $F = A \oplus B \oplus C \oplus D$



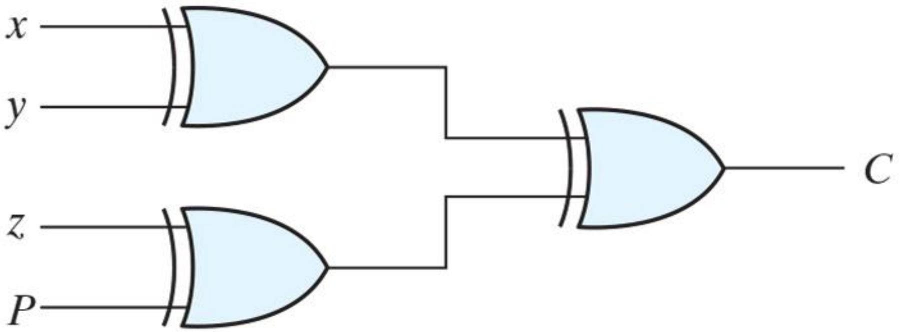
(b) Even function $F = (A \oplus B \oplus C \oplus D)'$

Pearson

Logic diagram of a parity generator and checker.



(a) 3-bit even-parity generator



(b) 4-bit even-parity checker

Schematic for *and_or_prop_delay.*