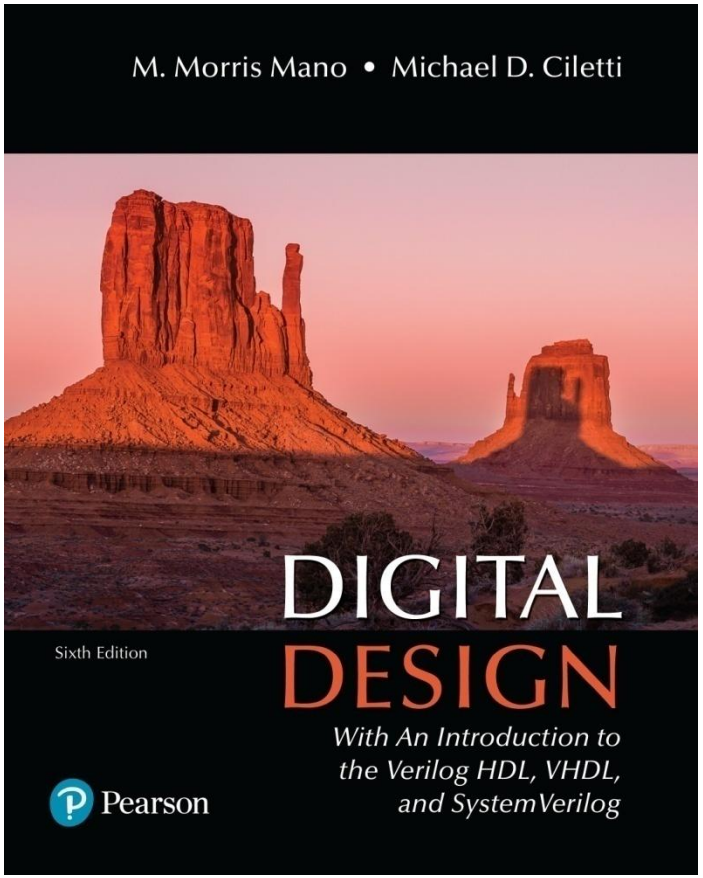
Digital Design

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

6th Edition



Chapter 03

Gate-Level Minimization

Pearson

Figure 3.1
Two-variable K-map.

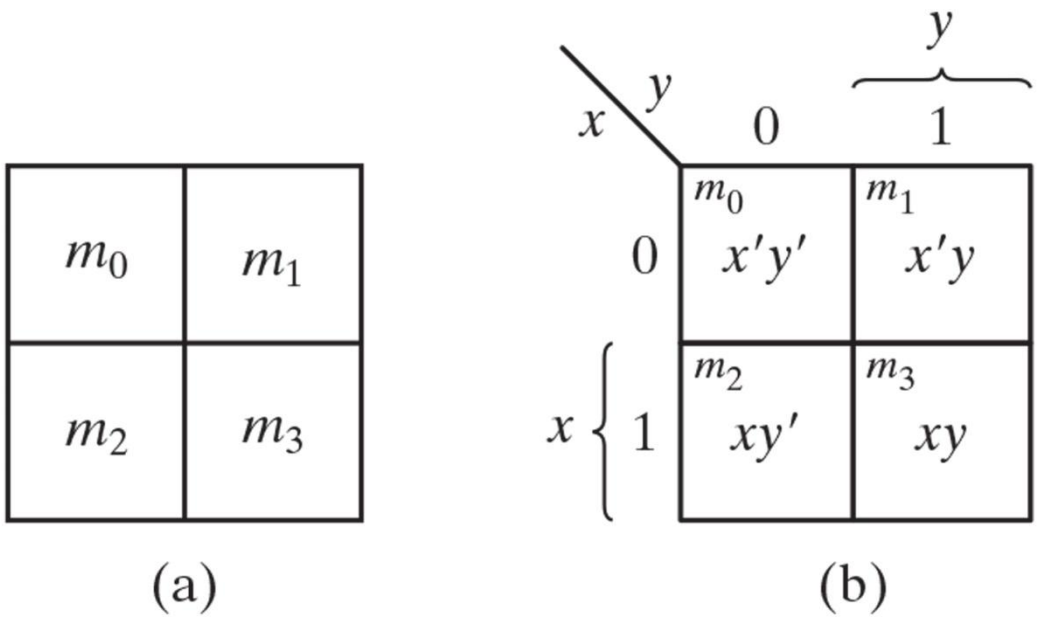


Figure 3.2
Representation of functions in the K-map.

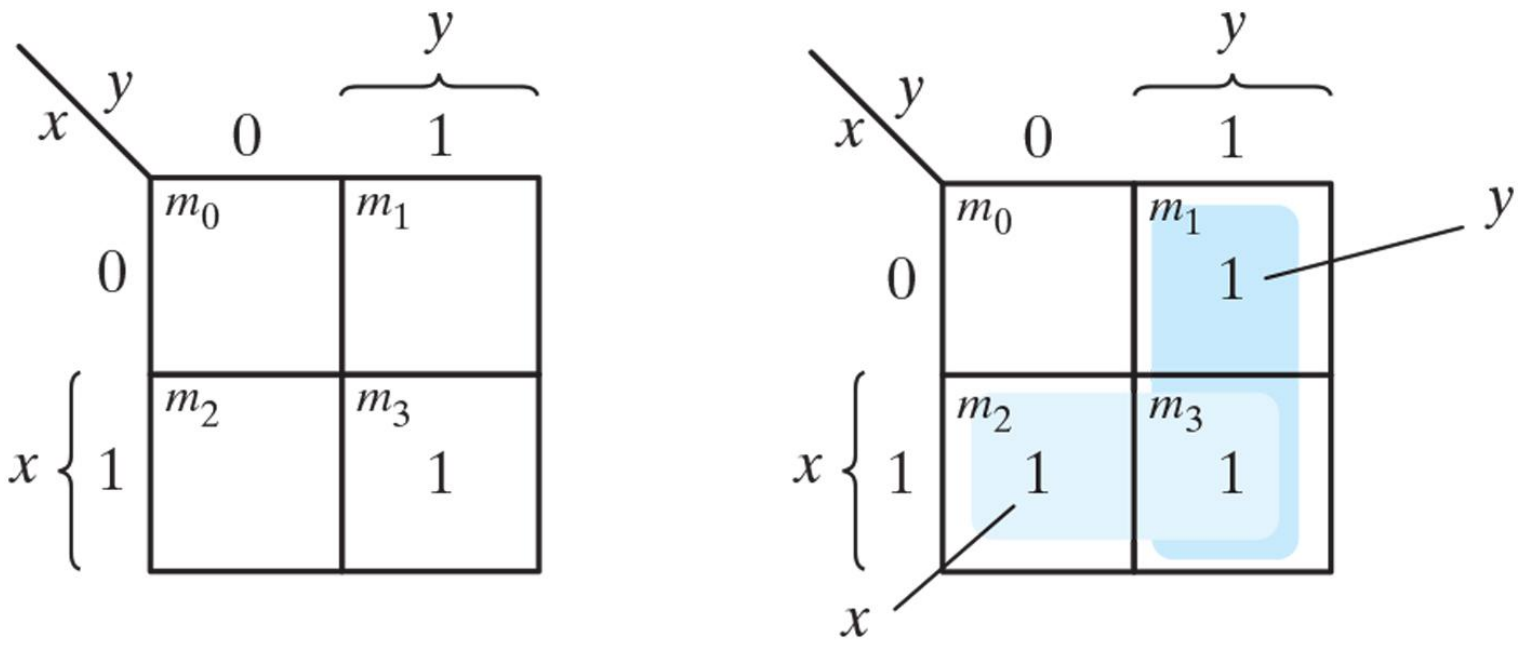
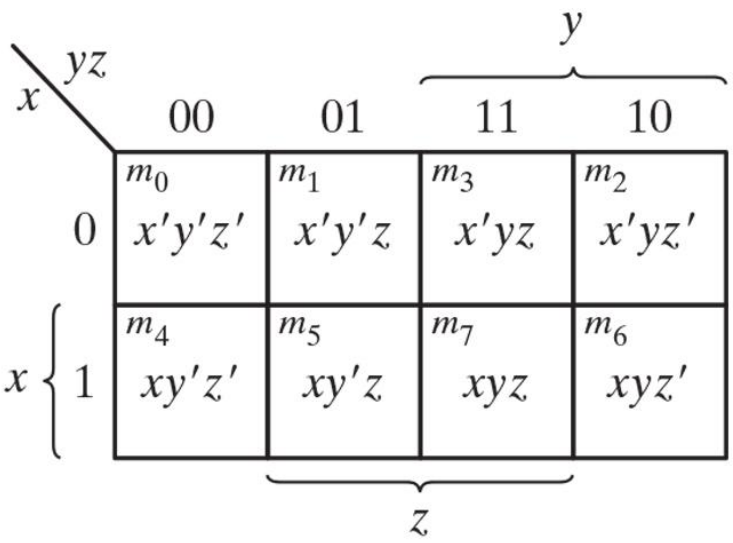


Figure 3.3
Three-variable K-map.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)



	00	01	11	10
0	m_0 $x'y'z'$	m_1 $x'y'z$	m_3 $x'yz$	m_2 $x'yz'$
1	m_4 $xy'z'$	m_5 $xy'z$	m_7 xyz	m_6 xyz'

(b)

Figure 3.4
Map for Example 3.1, *F(x, y, z)* = Σ (2, 3, 4, 5) = *x'y* + *xy'*.

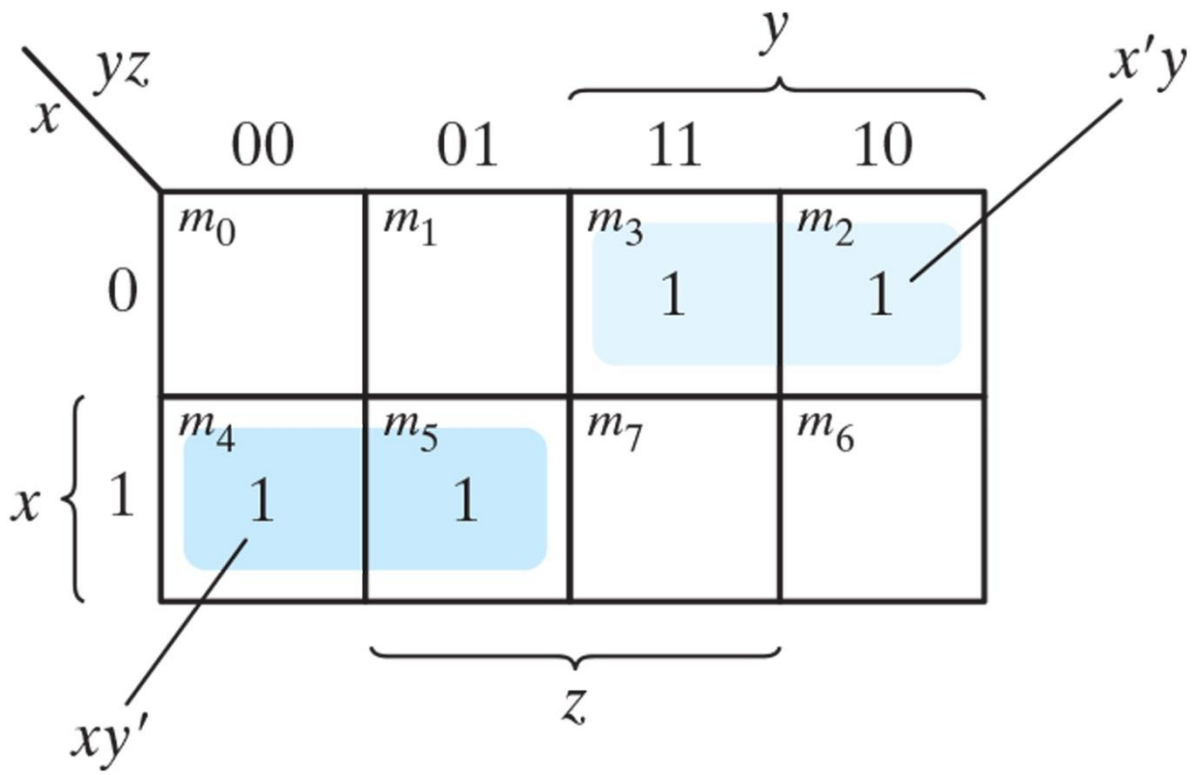


Figure 3.5
Map for Example 3.2, $F(x, y, z) = \Sigma\ (3, 4, 6, 7) = yz + xz'$.

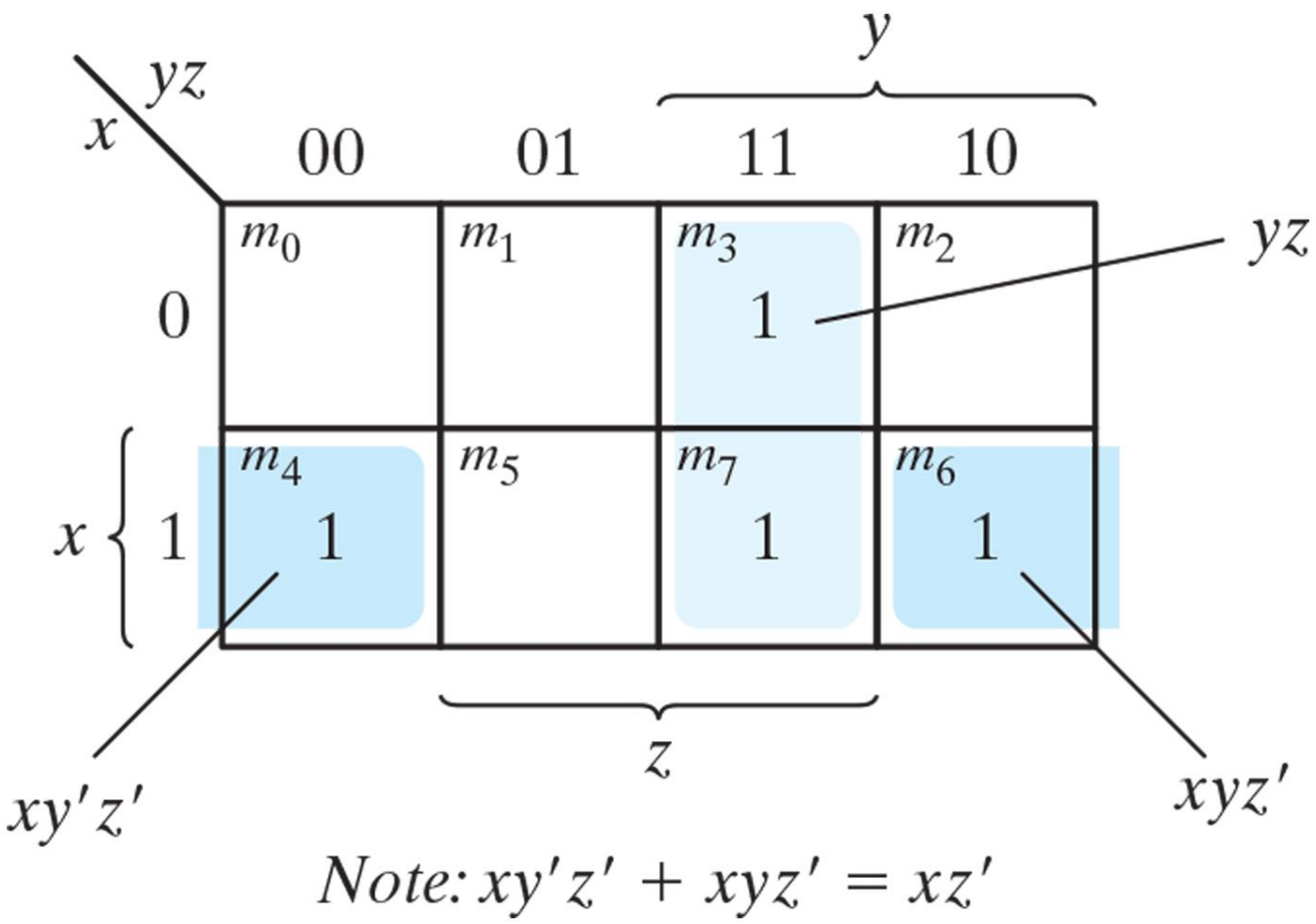
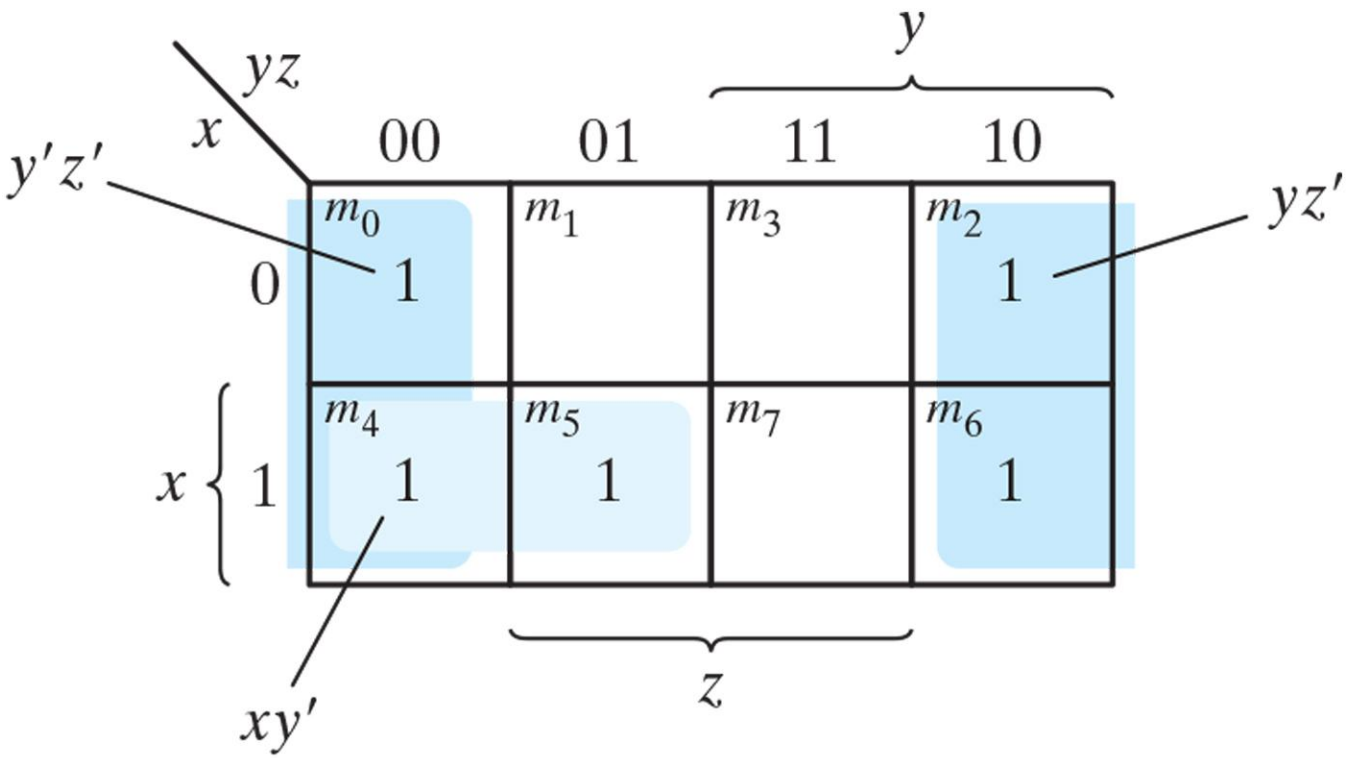


Figure 3.6
Map for Example 3.3, *F(x, y, z)* = Σ (0, 2, 4, 5, 6) = *z'* + *xy'*.



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

Figure 3.7
Map of Example 3.4, *A'C + A'B + AB'C + BC = C + A'B*.

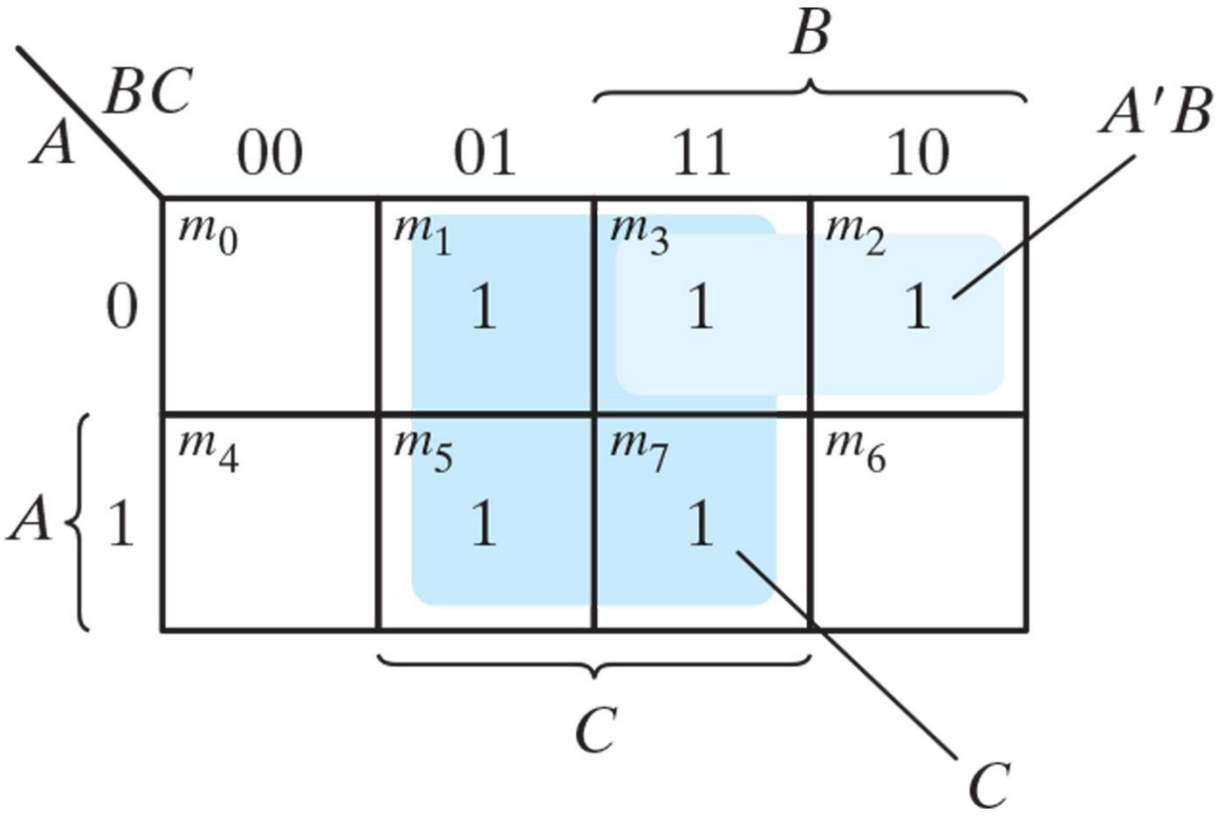
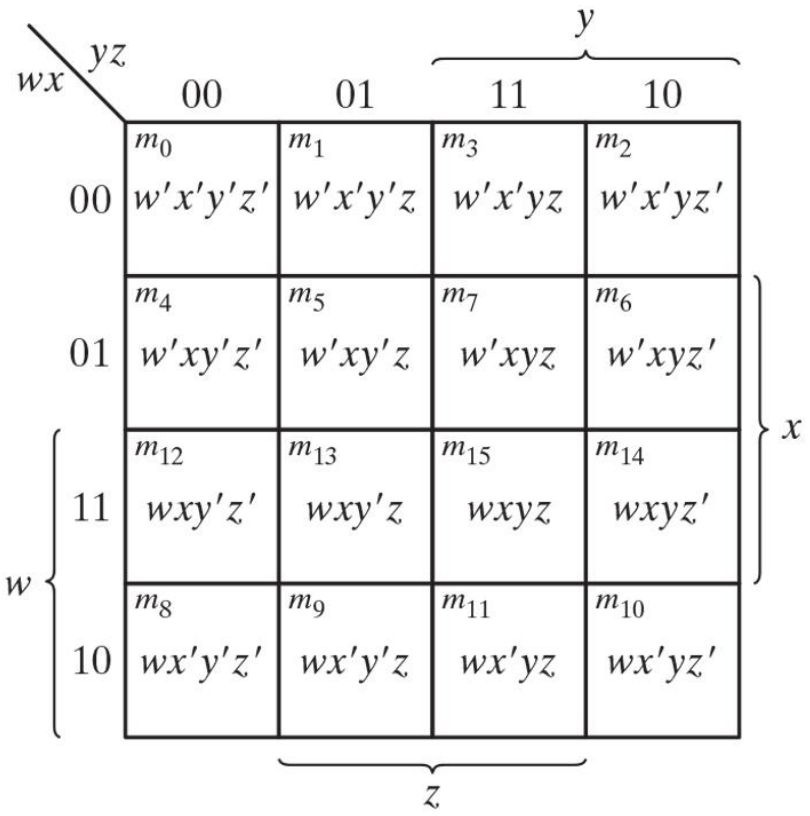


Figure 3.8
Four-variable map.

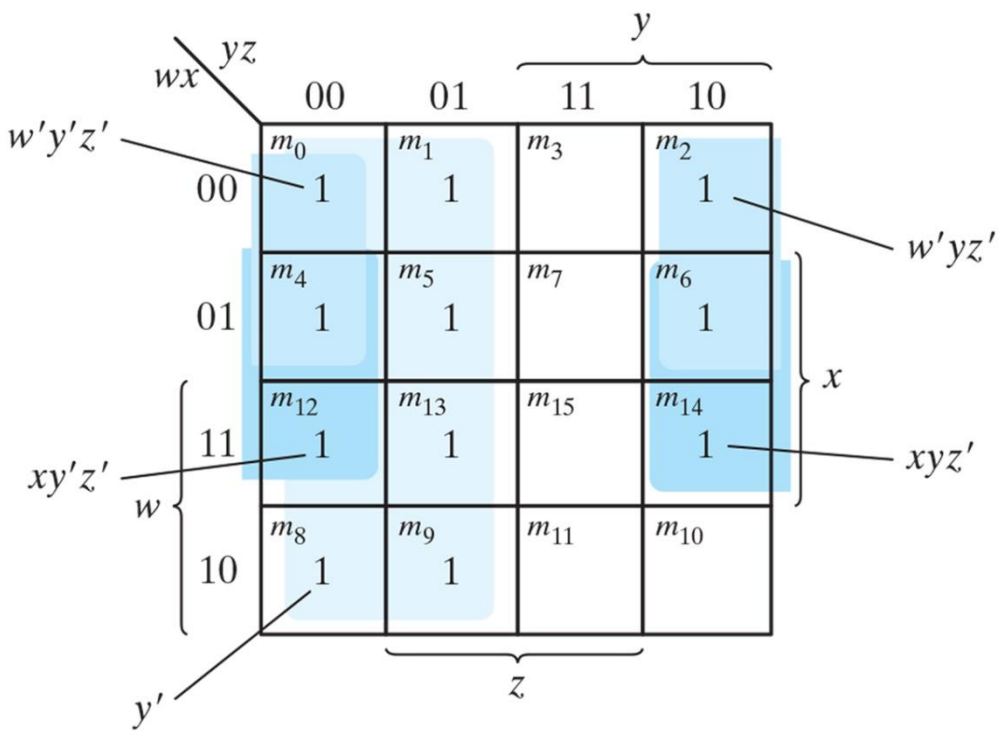
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)



(b)

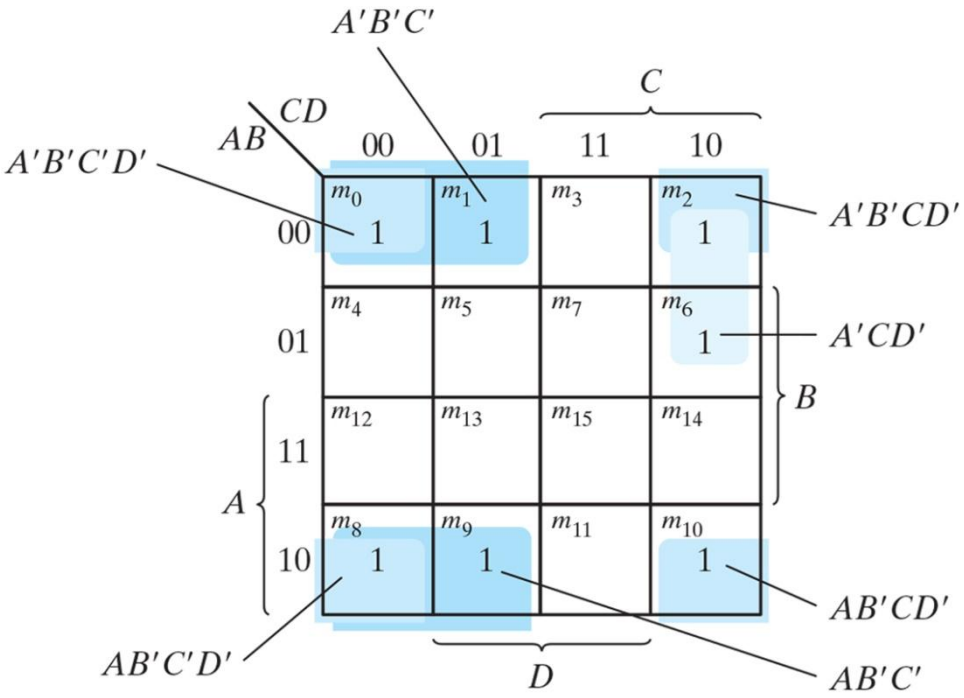
Figure 3.9
Map for Example 3.5, *F(w, x, y, z)* = Σ (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = *y'* + *w'z'* + *xz'*.



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

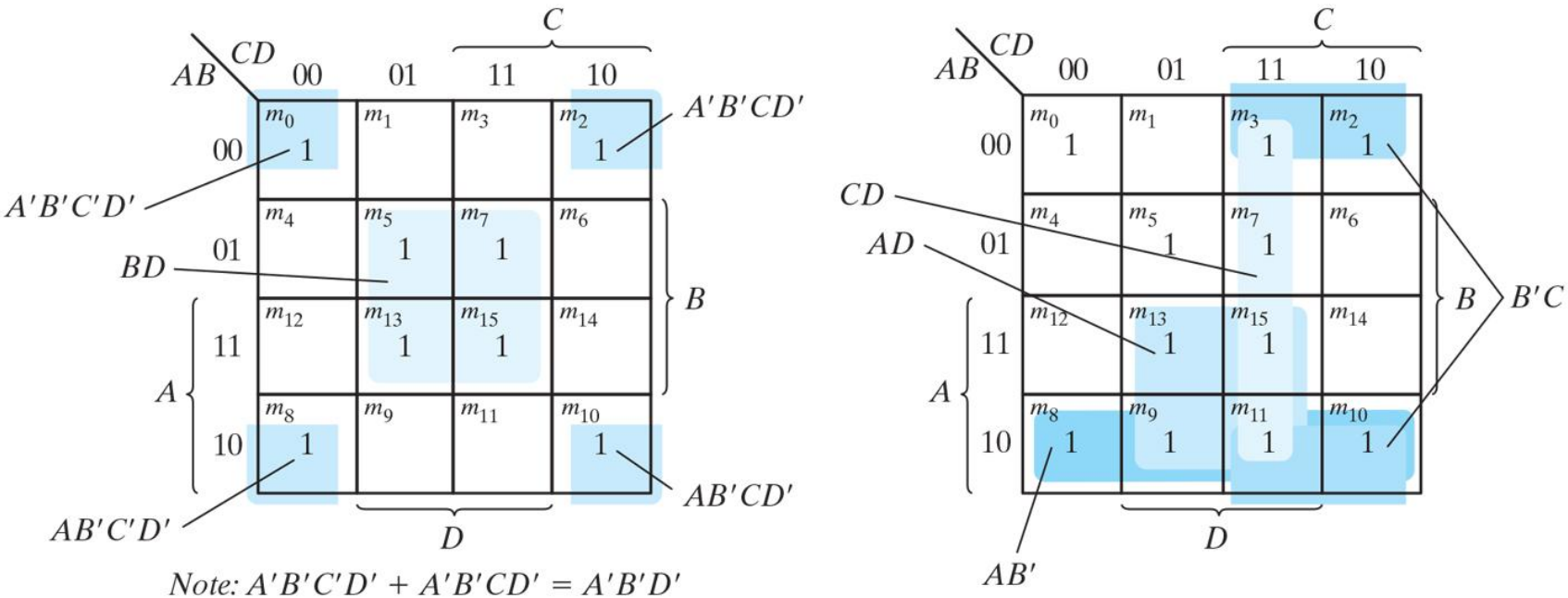
$xy'z' + xyz' = xz'$

Figure 3.10
Map for Example 3.6, *A'B'C' + B'CD' + A'BCD' + AB'C = B'D' + B'C' + A'CD'.*



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

Figure 3.11
Simplification using prime implicants.

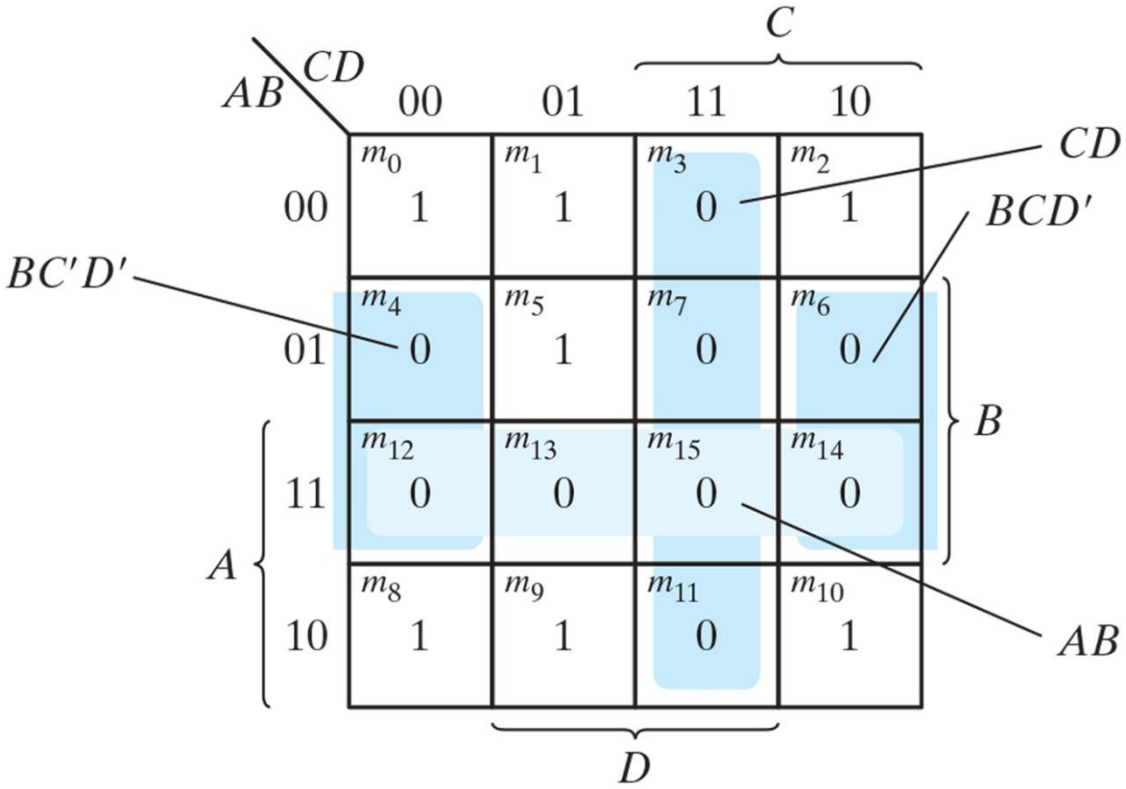


Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$

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

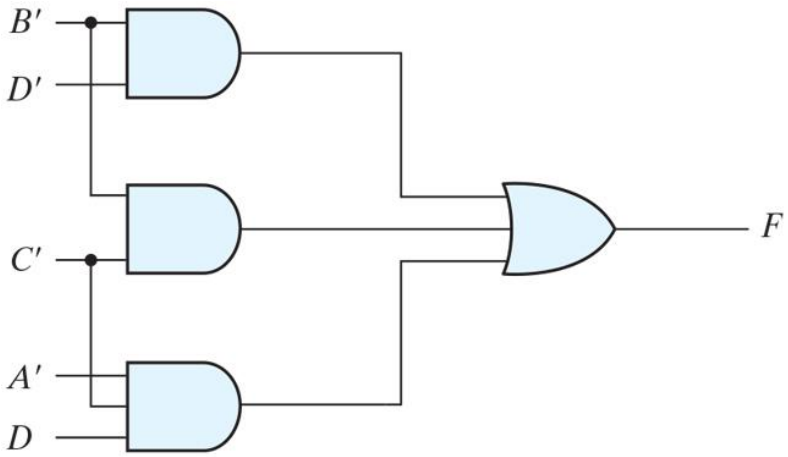
(b) Prime implicants CD, $B'C$, AD, and AB'

Figure 3.12
Map for Example 3.7, *F(A, B, C, D)* = Σ (0, 1, 2, 5, 8, 9, 10) = *BD + BC + ACD = (A' + B')(C' + D')(B' + D).*

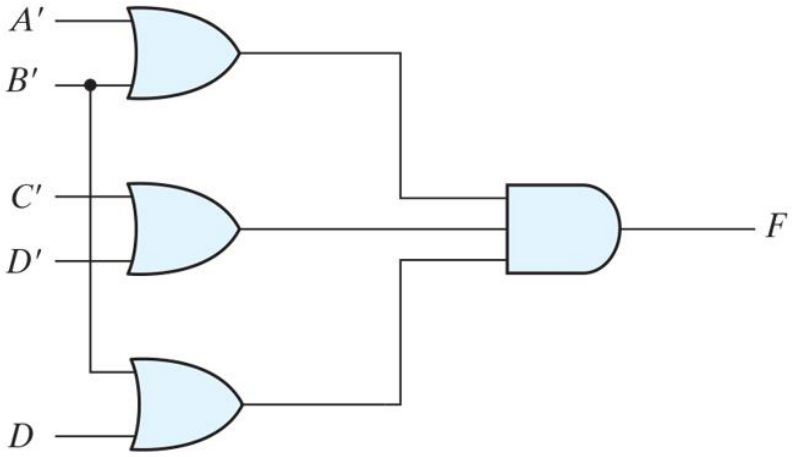


Note: $BC'D' + BCD' = BD'$

Figure 3.13
Gate implementations of the function of Example 3.7.



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

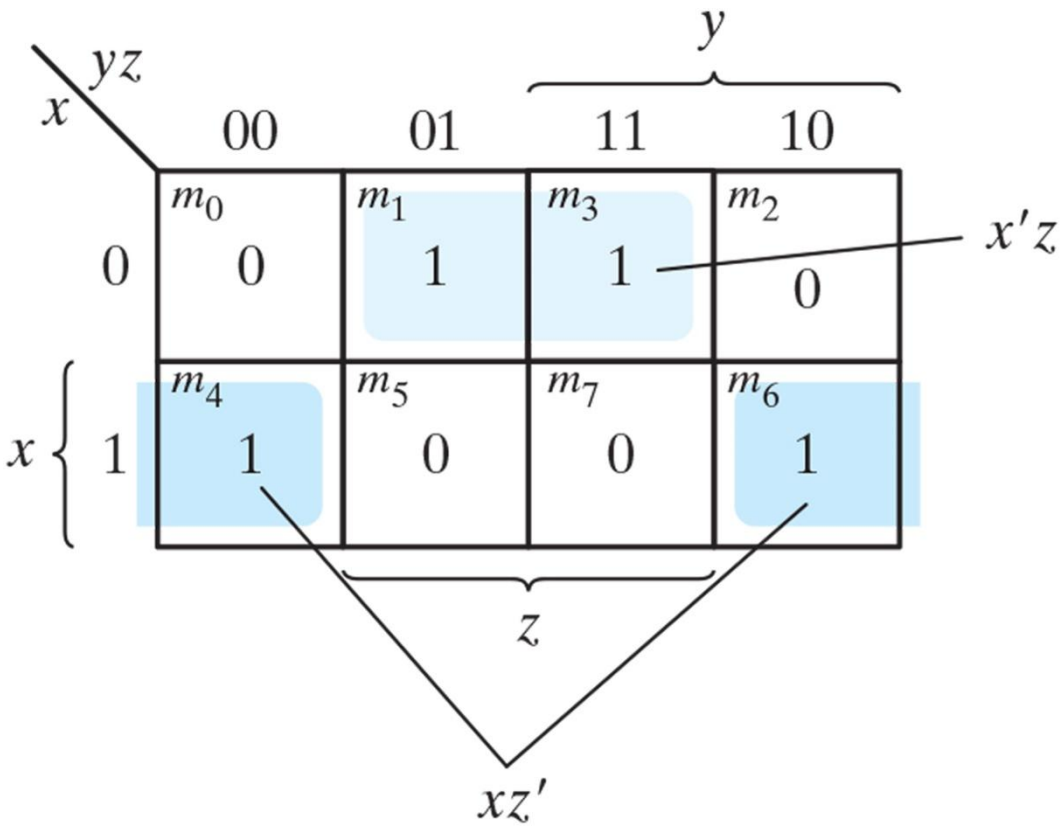


(b) $F = (A' + B')(C' + D')(B' + D)$

Table 3.1
Truth Table of Function F.

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

Figure 3.14
Map for the function of Table 3.1.



Practice Exercise 3.8

wxyz F	*wxyz F*
0000 1	1000 1
0001 0	1001 0
0010 1	1010 1
0011 0	1011 0
0100 0	1100 1
0101 0	1101 1
0110 0	1110 1
0111 0	1111 0

Figure 3.15
Example with don't-care conditions.

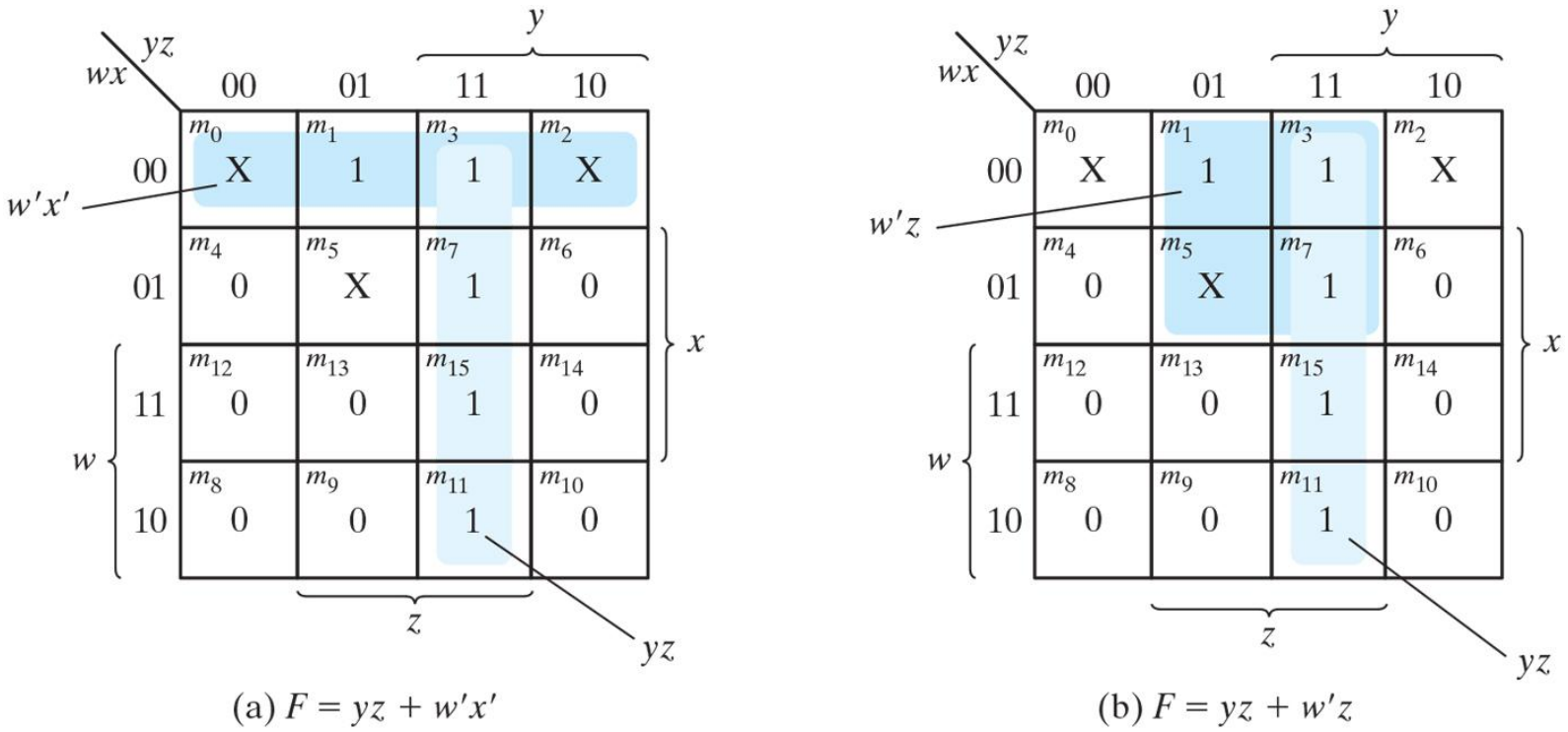


Figure 3.16
Logic operations with NAND gates.

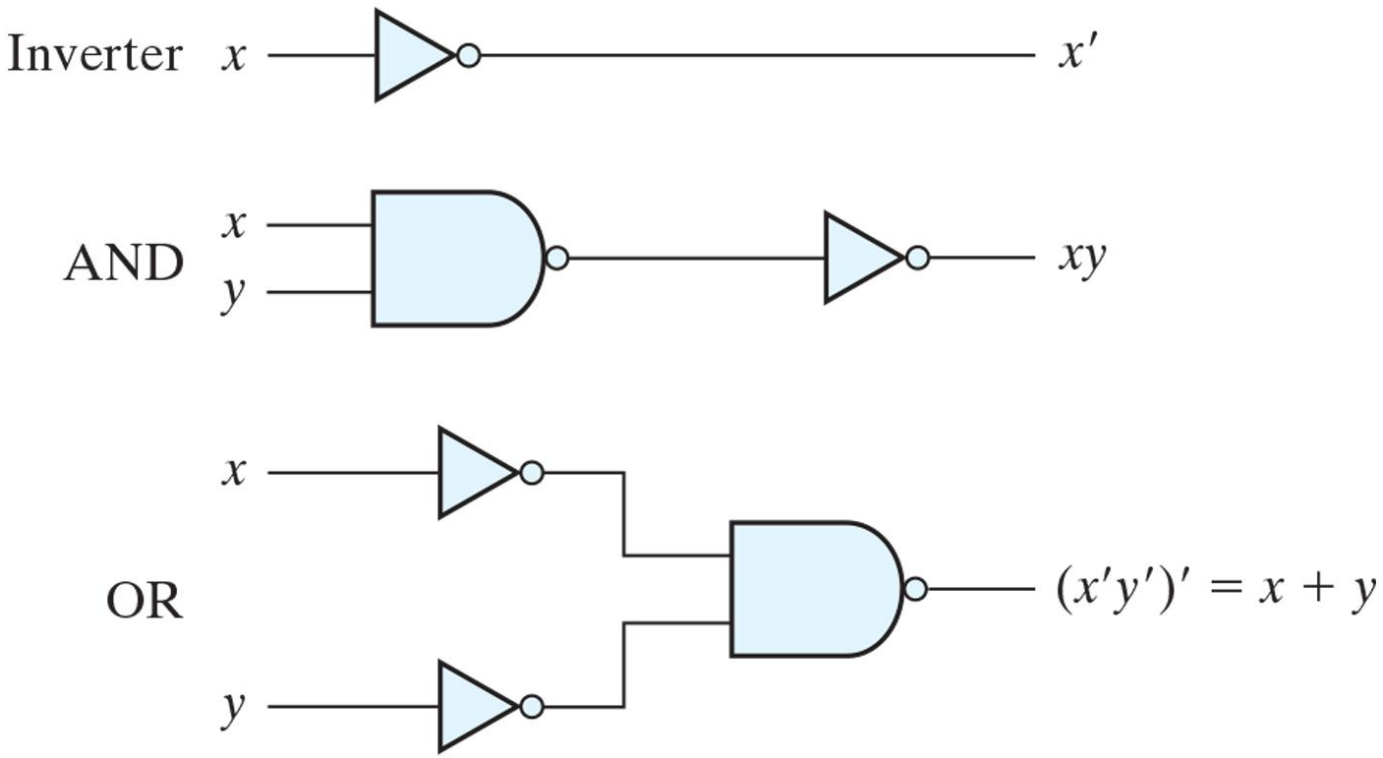
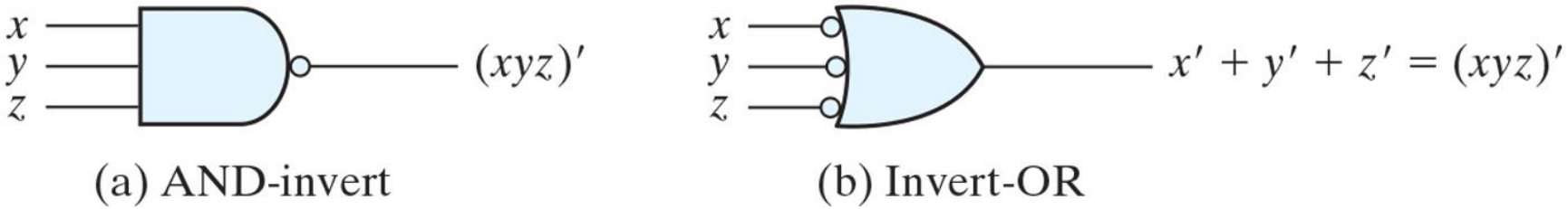


Figure 3.17
Two graphic symbols for a three-input NAND gate.



(a) AND-invert

(b) Invert-OR

Figure 3.18
Three ways to implement *F* = *AB* + *CD*.

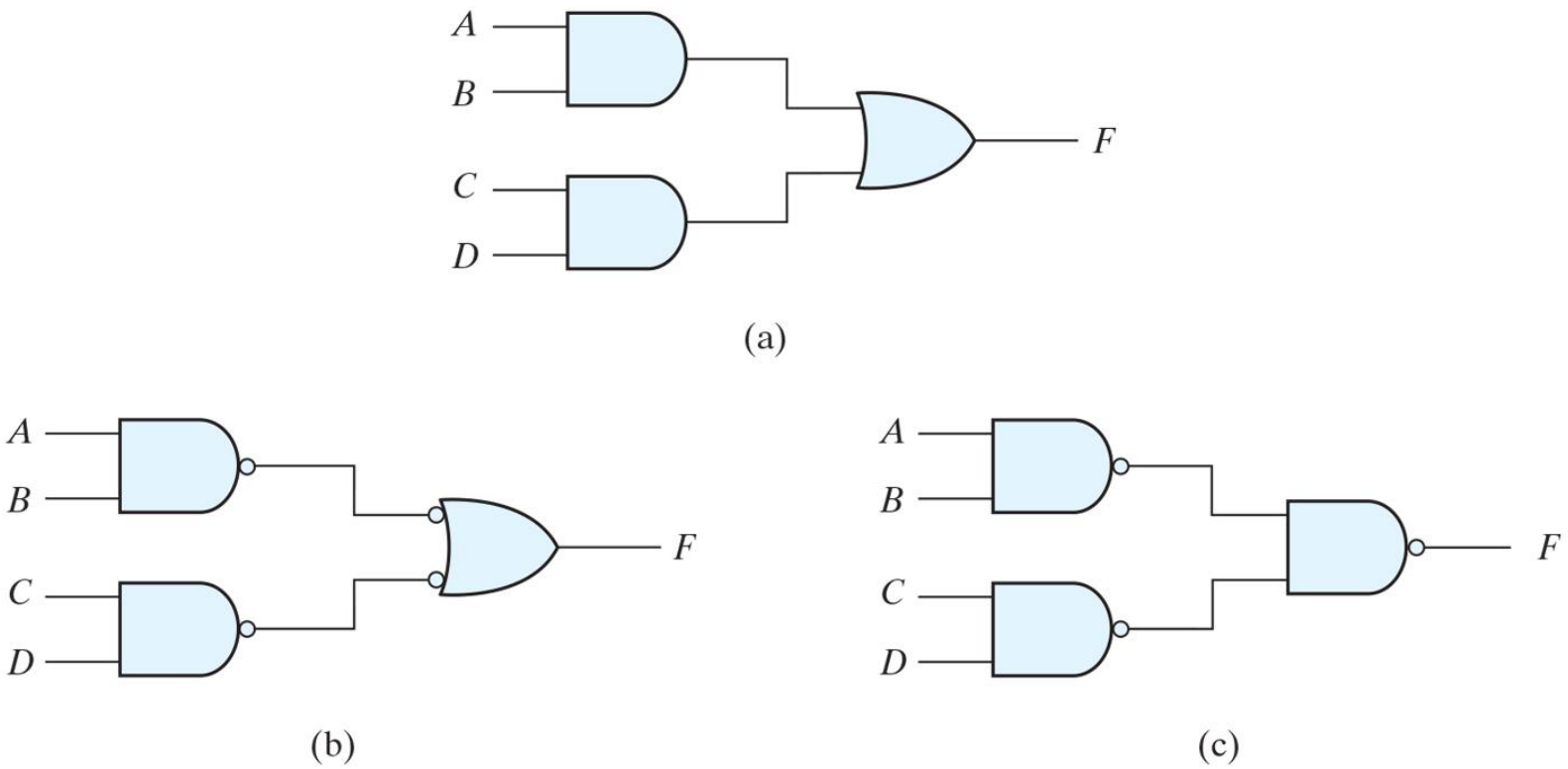
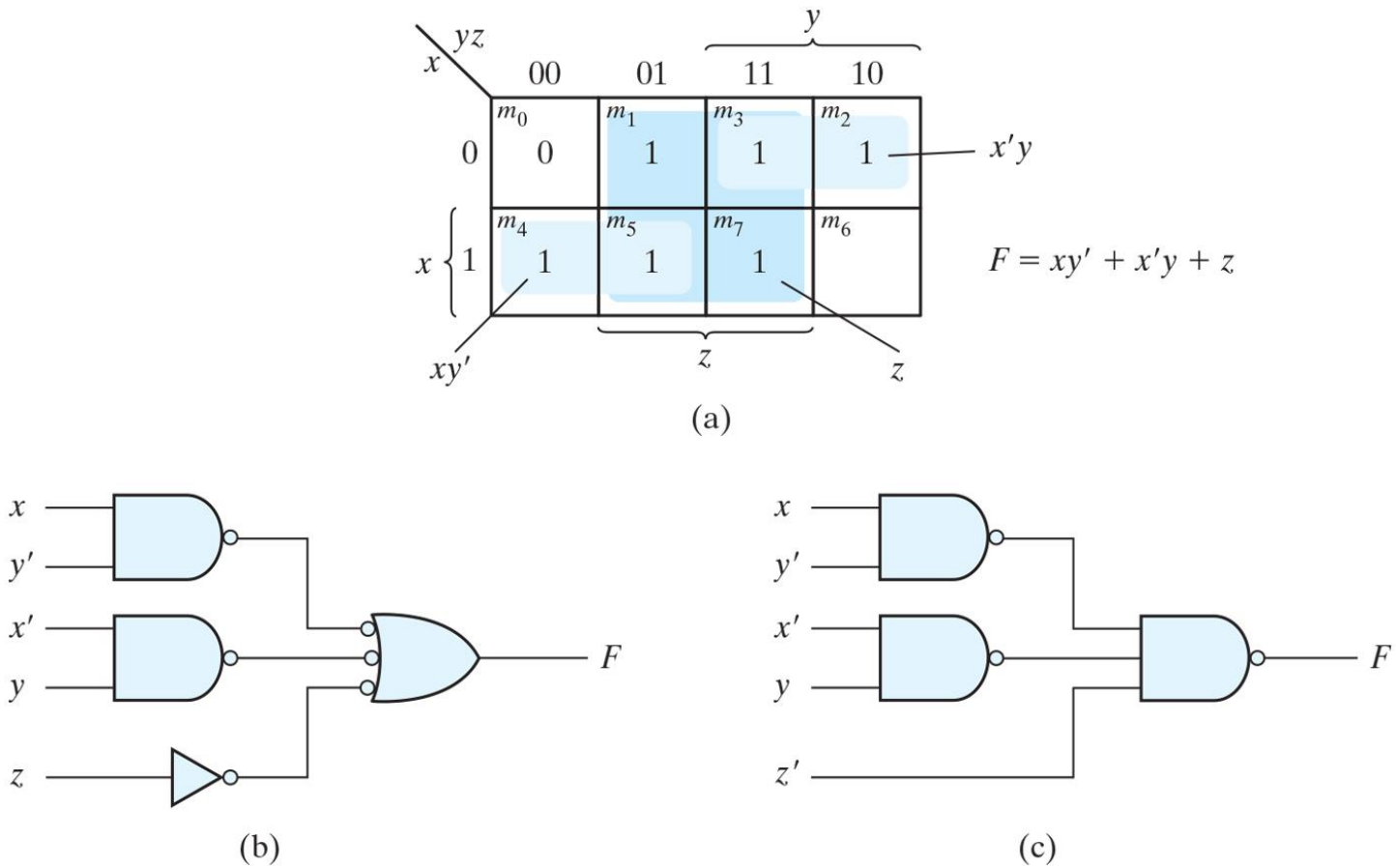


Figure 3.19
Solution to Example 3.9.



Practice Exercise 3.10

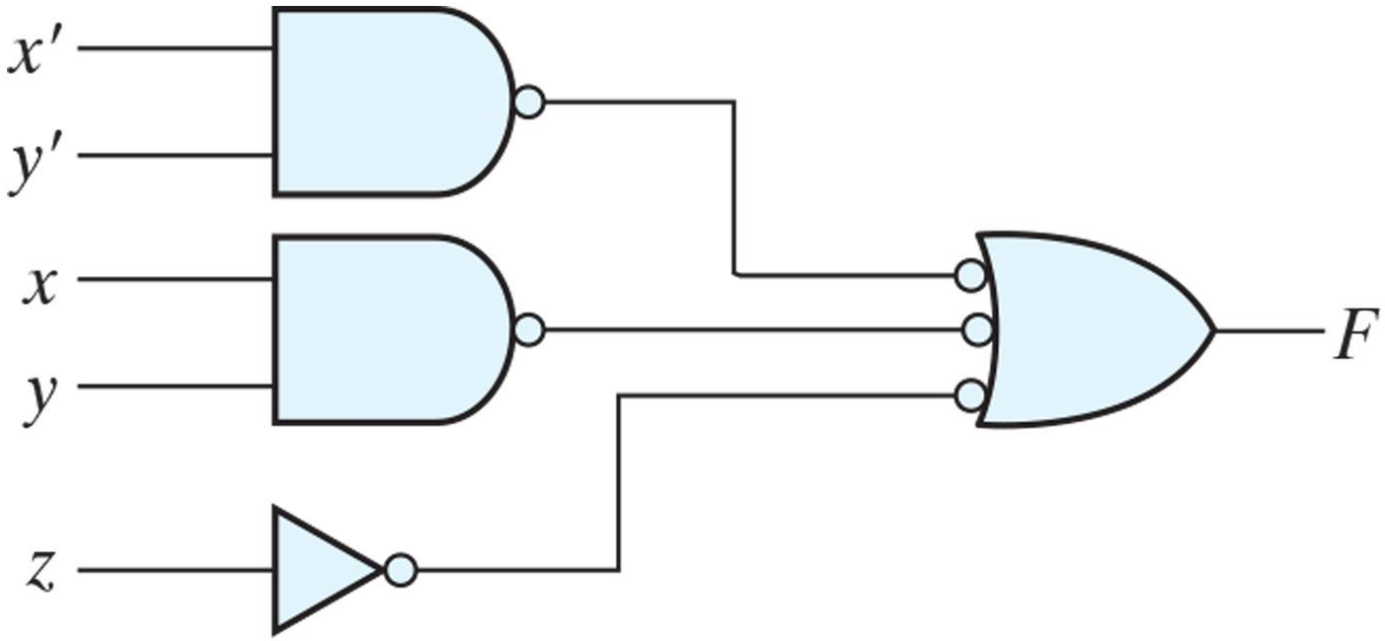
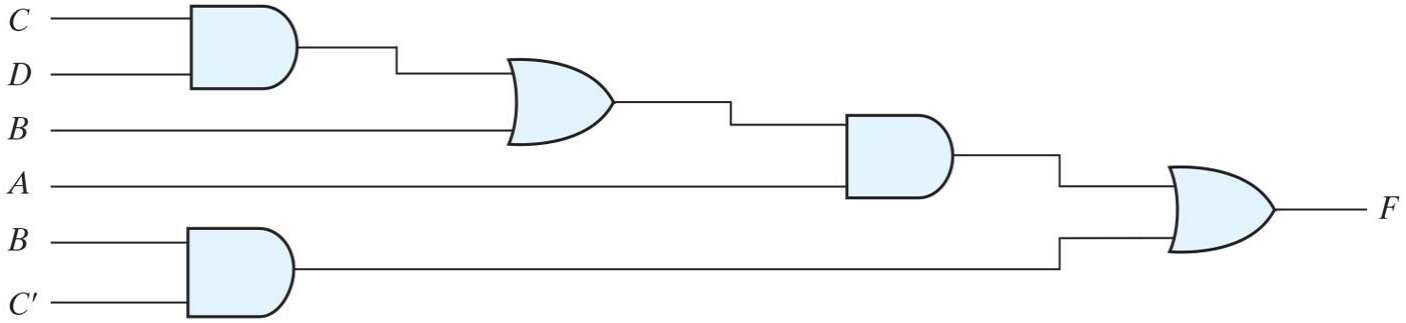
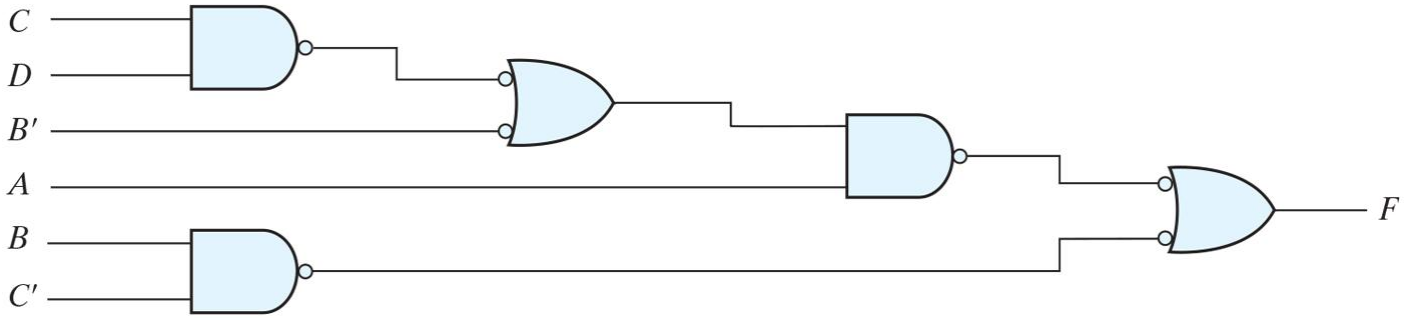


Figure 3.20
Implementing *F = A(CD + B) + BC'*.

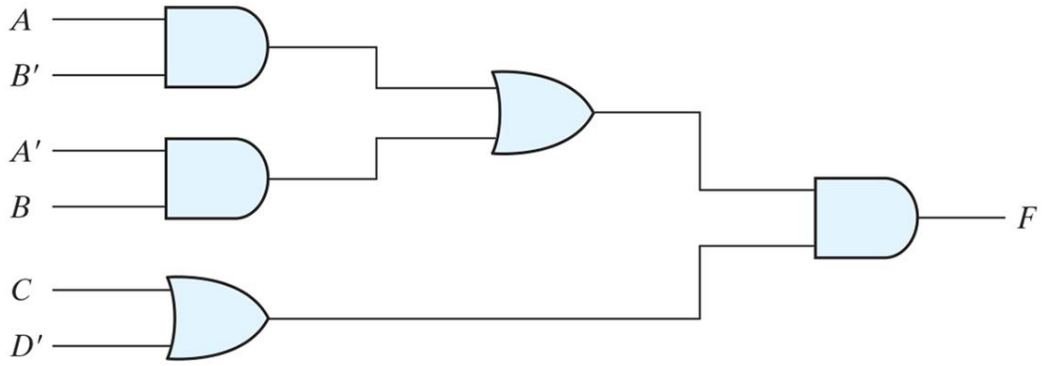


(a) AND–OR gates

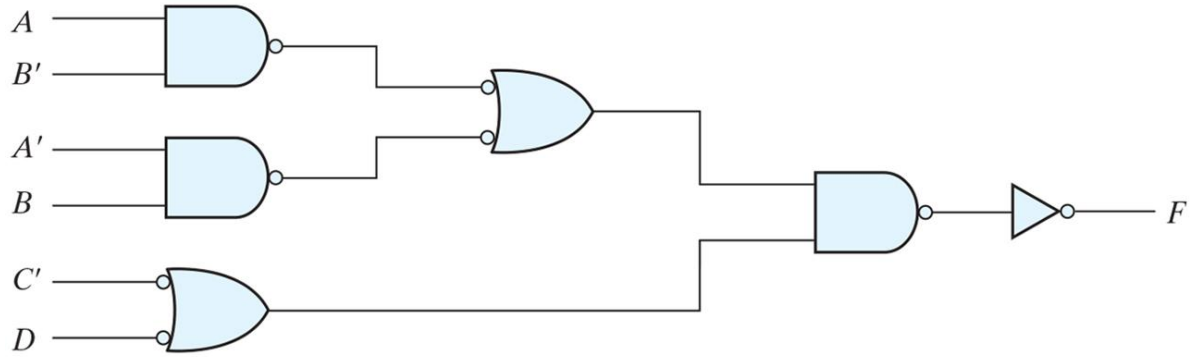


(b) NAND gates

Figure 3.21
Implementing *F = (AB' + A'B)(C + D')*.



(a) AND–OR gates



(b) NAND gates

Figure 3.22
Logic operations with NOR gates.

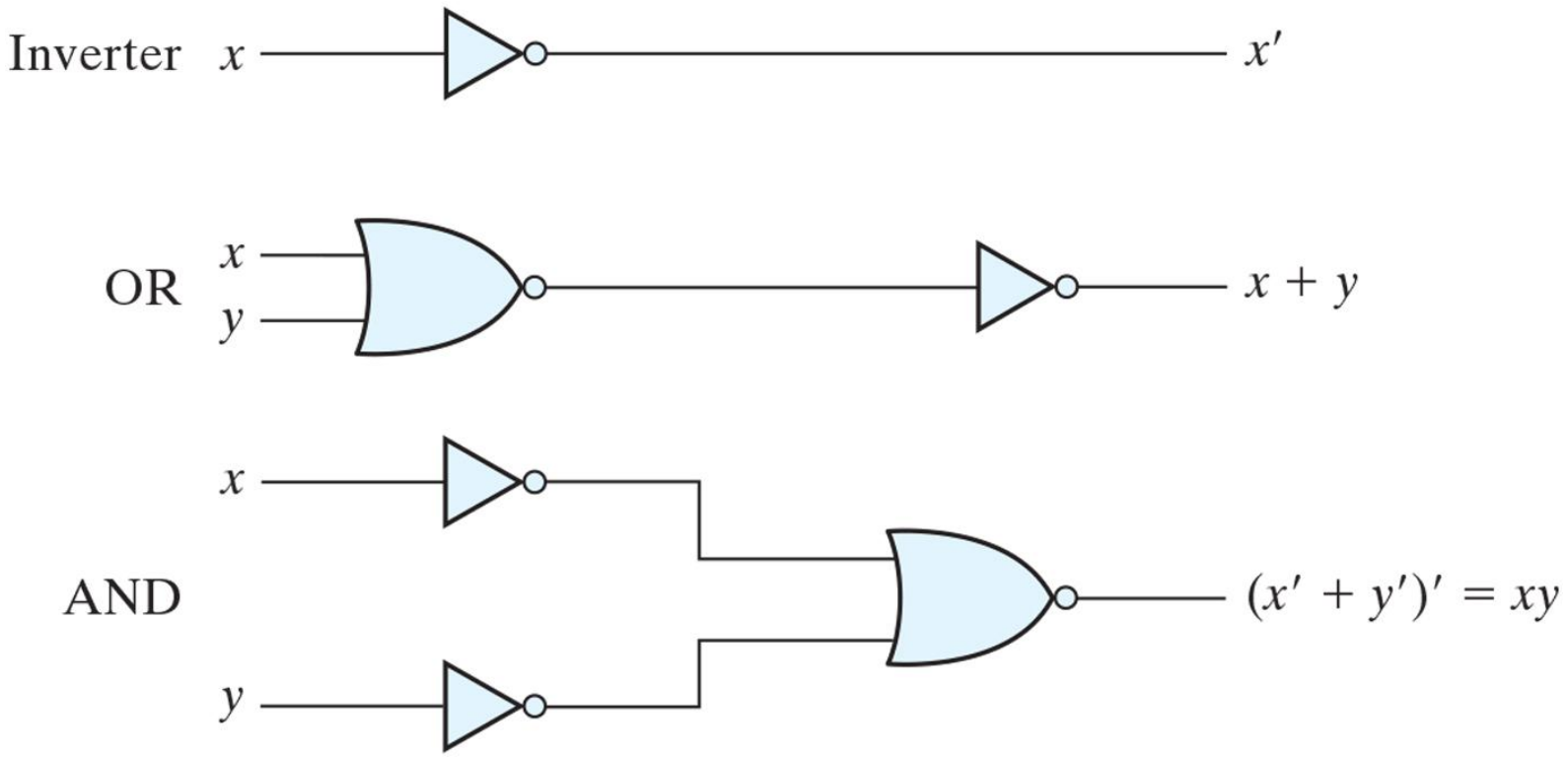
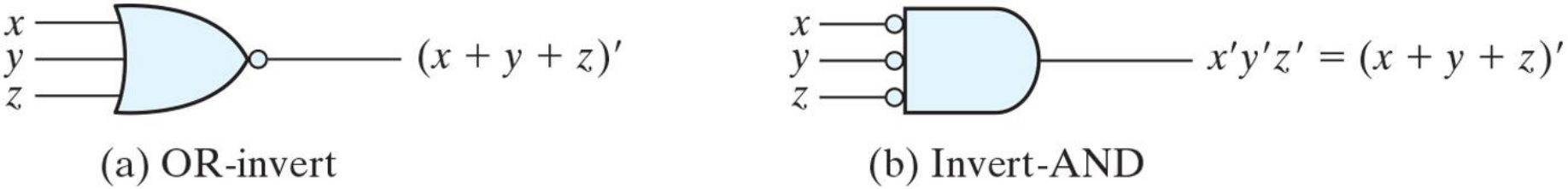


Figure 3.23
Two graphic symbols for the NOR gate.



(a) OR-invert

(b) Invert-AND

Figure 3.24
Implementing *F = (A + B)(C + D)E*.

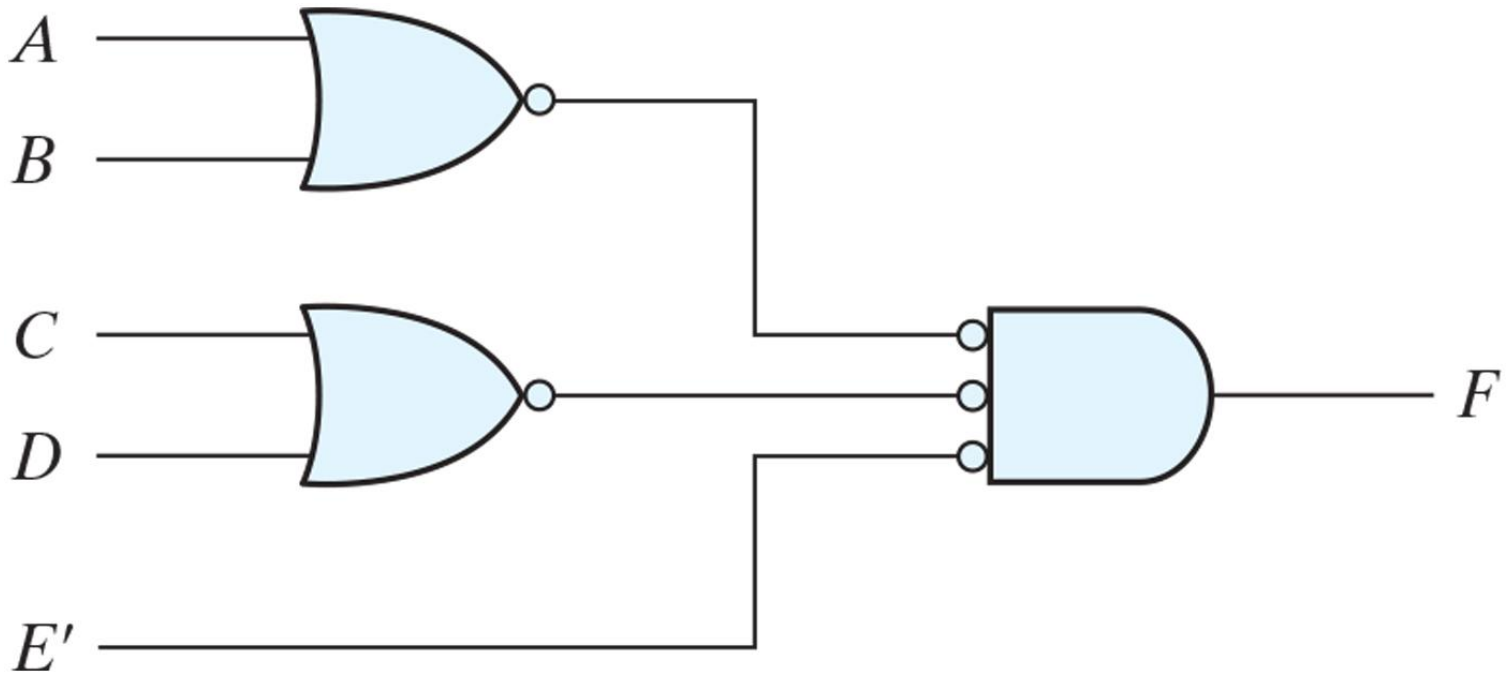
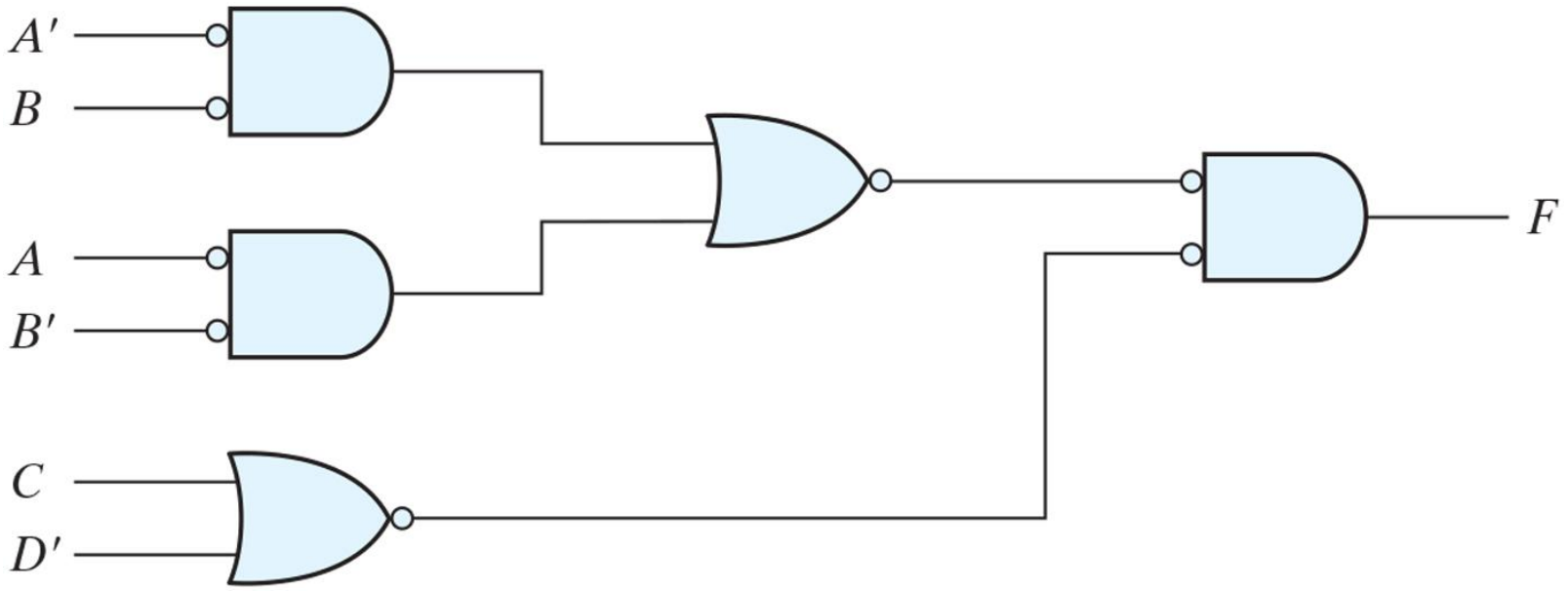


Figure 3.25
Implementing *F = (AB' + A'B)(C + D')* with NOR gates.



Practice Exercise 3.11

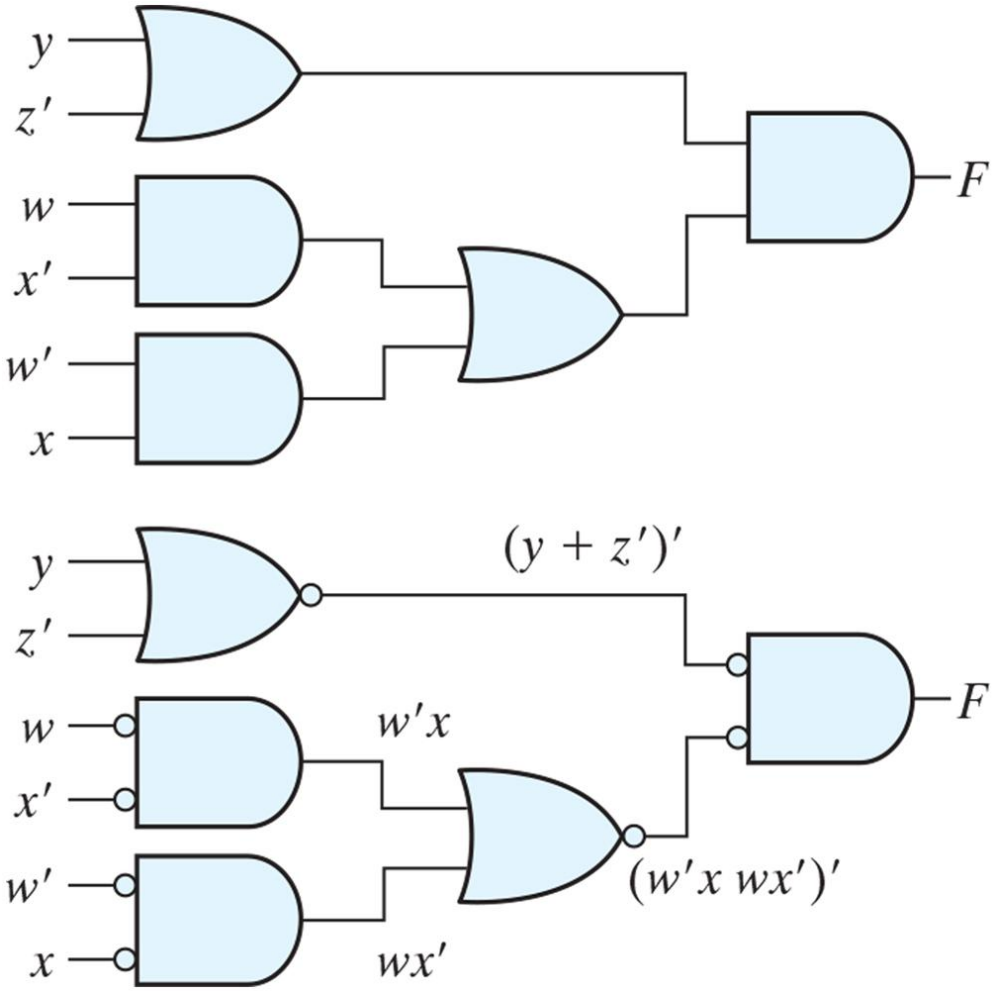
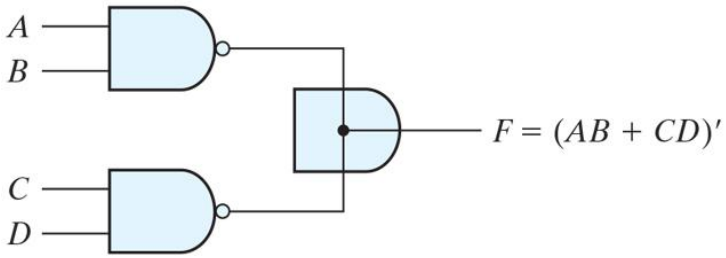
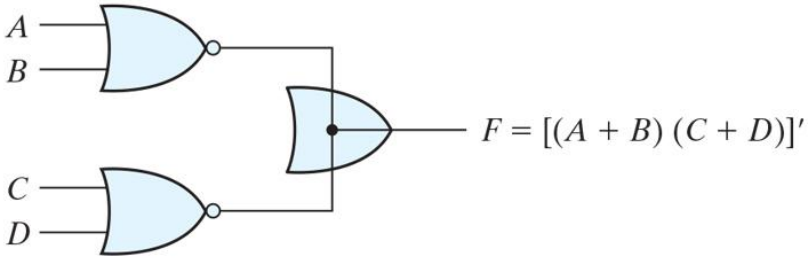


Figure 3.26
Wired logic
(a) Wired-AND logic with two NAND gates
(b) Wired-OR in emitter-coupled logic (ECL) gates.



(a) Wired-AND in open-collector TTL NAND gates.

(AND–OR–INVERT)



(b) Wired-OR in ECL gates

(OR–AND–INVERT)

Figure 3.27
AND–OR–INVERT circuits, *F = (AB + CD + E)'.*

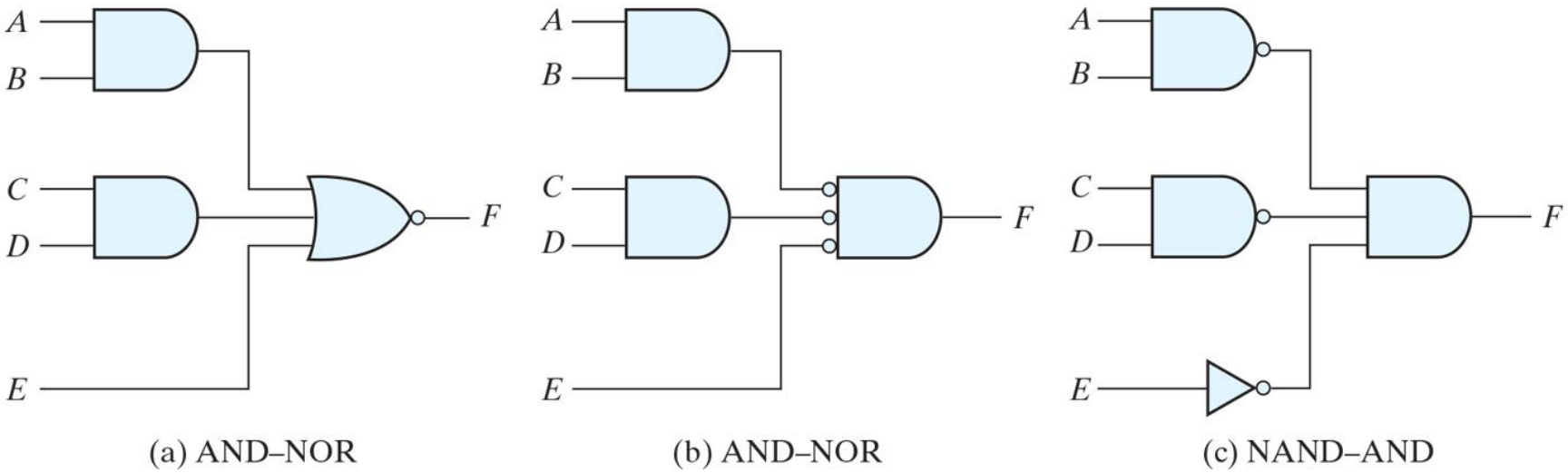


Figure 3.28
OR–AND–INVERT circuits, *F = [(A + B)(C + D)E]'.*

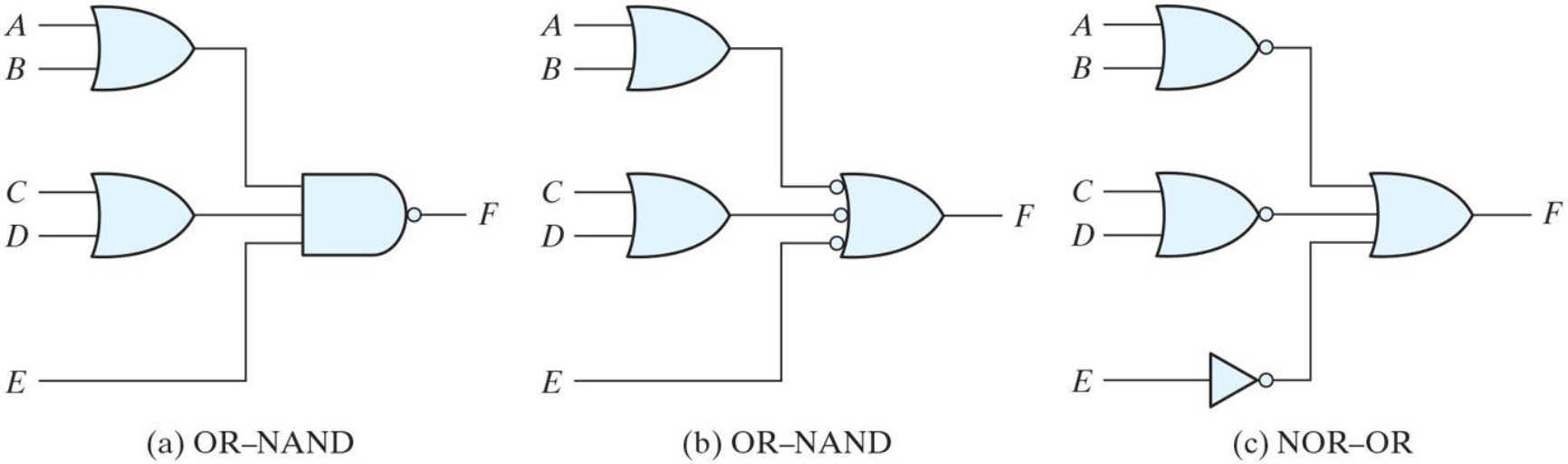
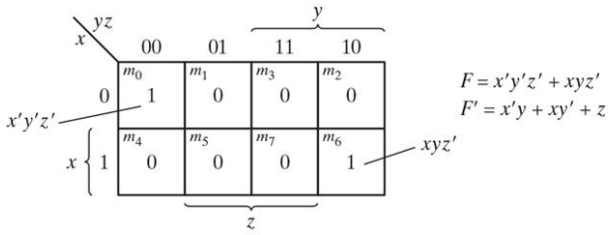


Table 3.2
Implementation with Other Two-Level Forms.

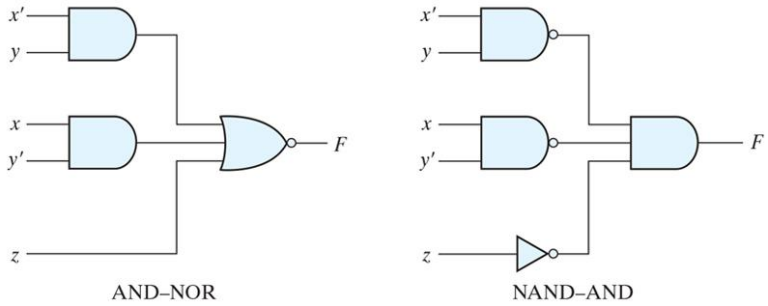
Equivalent Nondegenerate Implementation		Implements the Form	Simplify F' into	To Get an Output of
(a)	(b)*			
AND–NOR	NAND–AND	AND–OR–INVERT	Sum-of-products form by combining 0's in the map.	F
OR–NAND	NOR–OR	OR–AND–INVERT	Product-of-sums form by combining 1's in the map and then complementing.	F

*Form (b) requires an inverter for a single literal term.

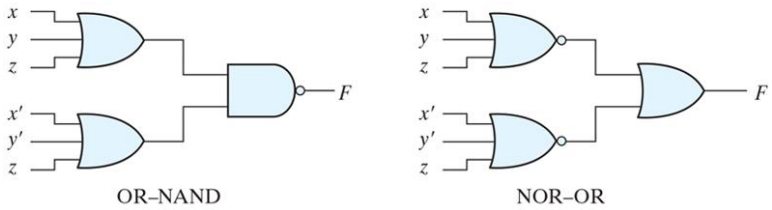
Figure 3.29
Other two-level implementations.



(a) Map simplification in sum of products

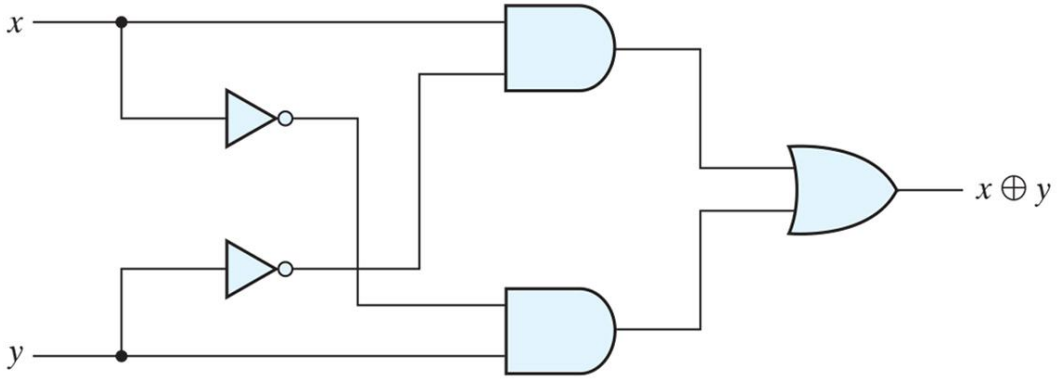


(b) $F = (x'y + xy' + z)'$

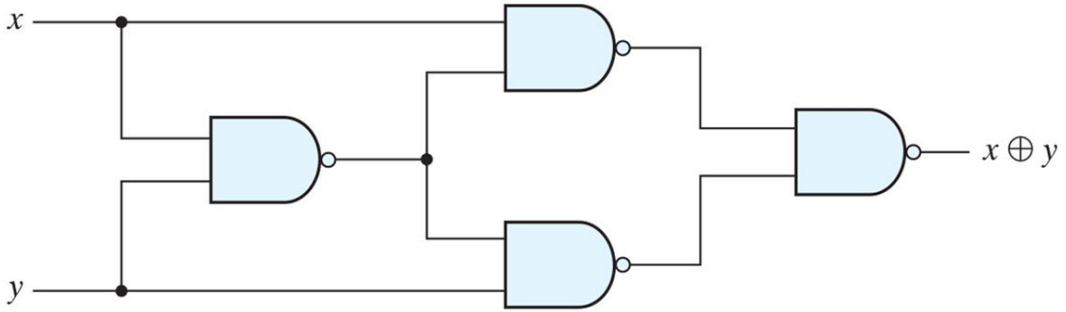


(c) $F = [(x + y + z)(x' + y' + z)]'$

Figure 3.30
Logic diagrams for exclusive-OR implementations.

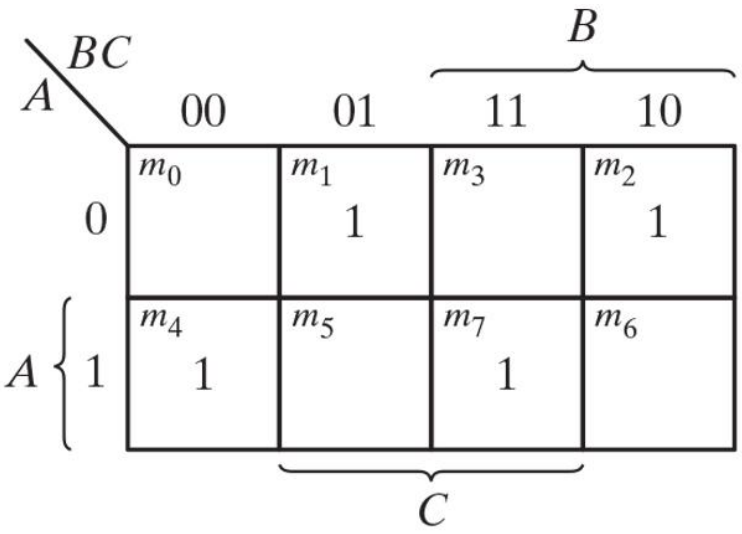


(a) Exclusive-OR with AND–OR–NOT gates

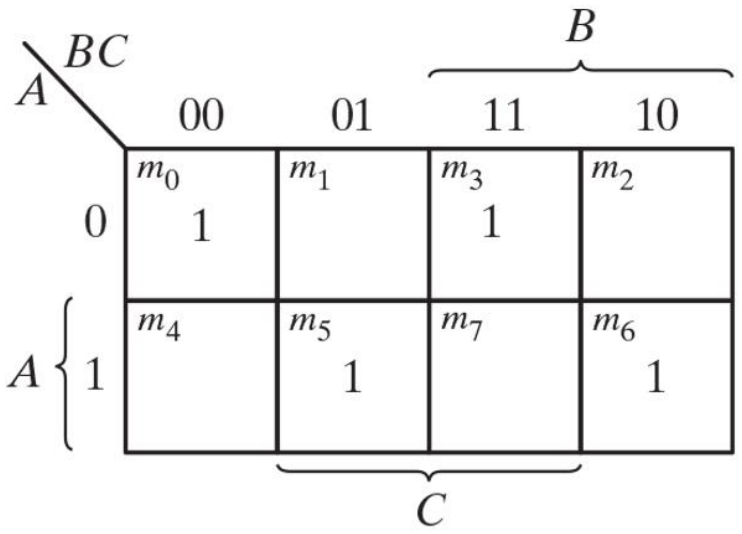


(b) Exclusive-OR with NAND gates

Figure 3.31
Map for a three-variable exclusive-OR function.

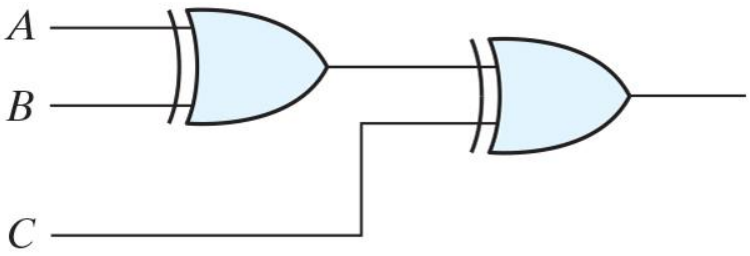


(a) Odd function $F = A \oplus B \oplus C$

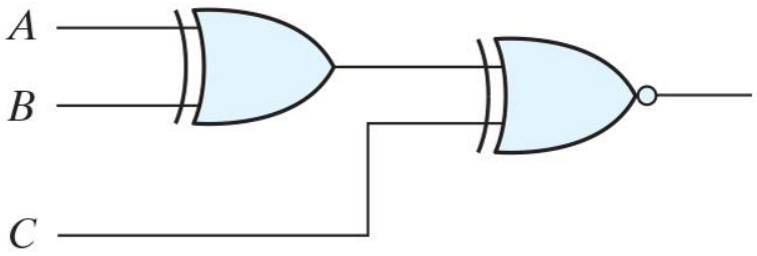


(b) Even function $F = (A \oplus B \oplus C)'$

Figure 3.32
Logic diagram of odd and even functions.

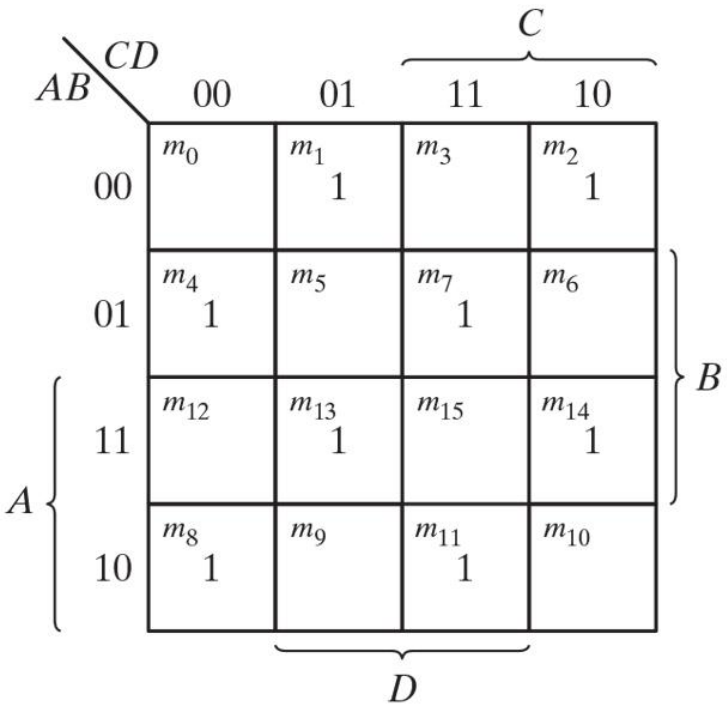


(a) 3-input odd function

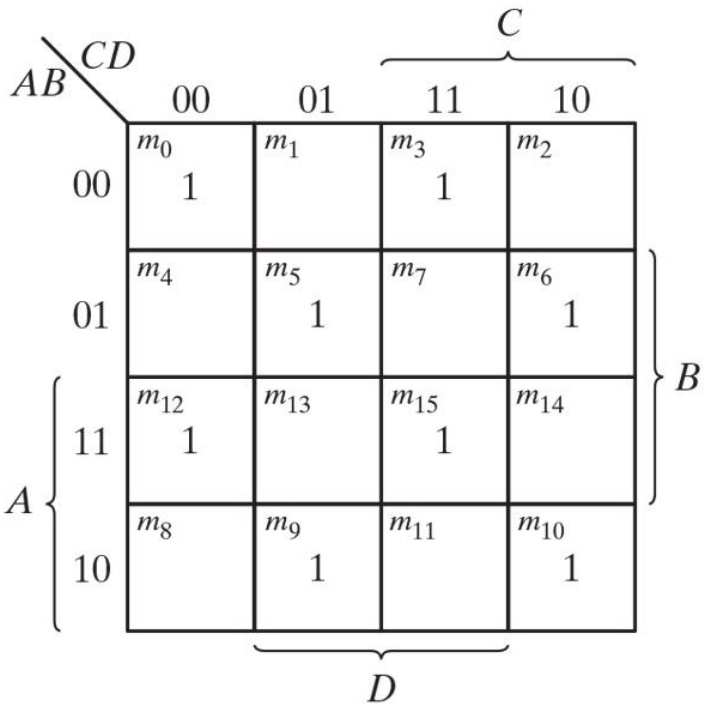


(b) 3-input even function

Figure 3.33
Map for a four-variable exclusive-OR function.



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

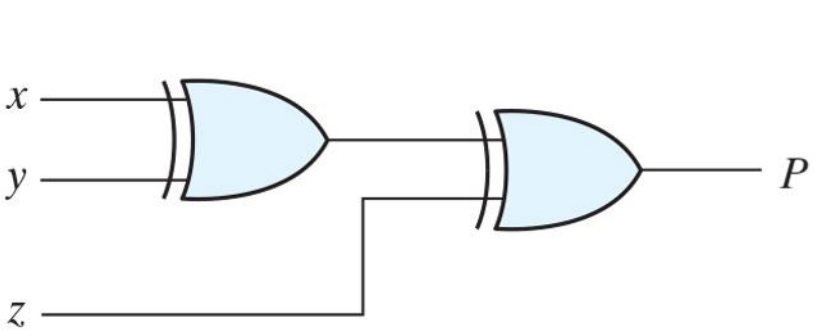


(b) Even function $F = (A \oplus B \oplus C \oplus D)'$

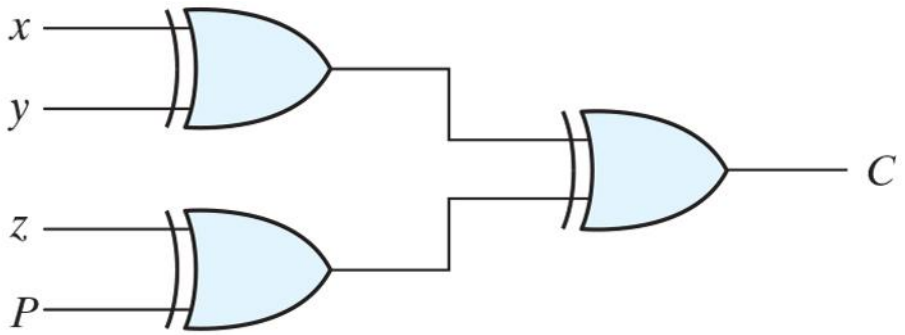
Table 3.3
Even-Parity-Generator Truth Table.

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

Figure 3.34
Logic diagram of a parity generator and checker.



(a) 3-bit even-parity generator



(b) 4-bit even-parity checker

Table 3.4
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

Figure 3.35
A logic diagram (schematic) for the Boolean equations *D* = *A* + *B* *E* = *CD*.

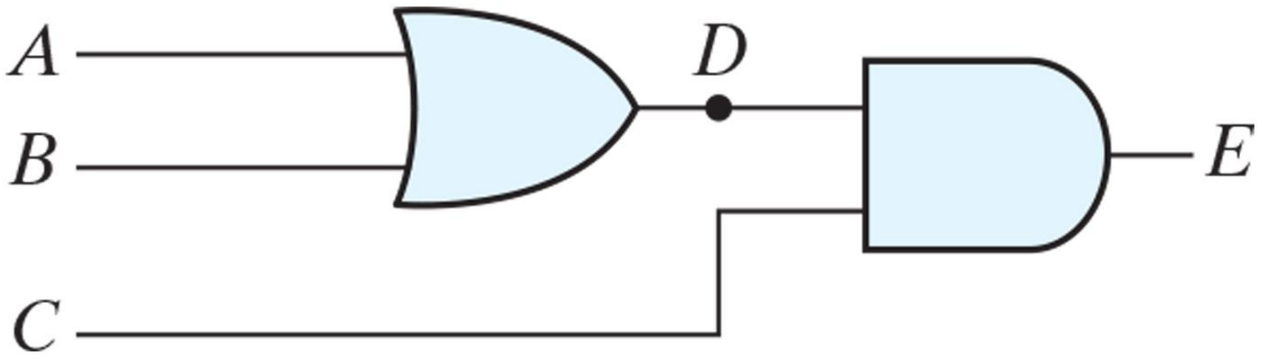


Figure PE3.12

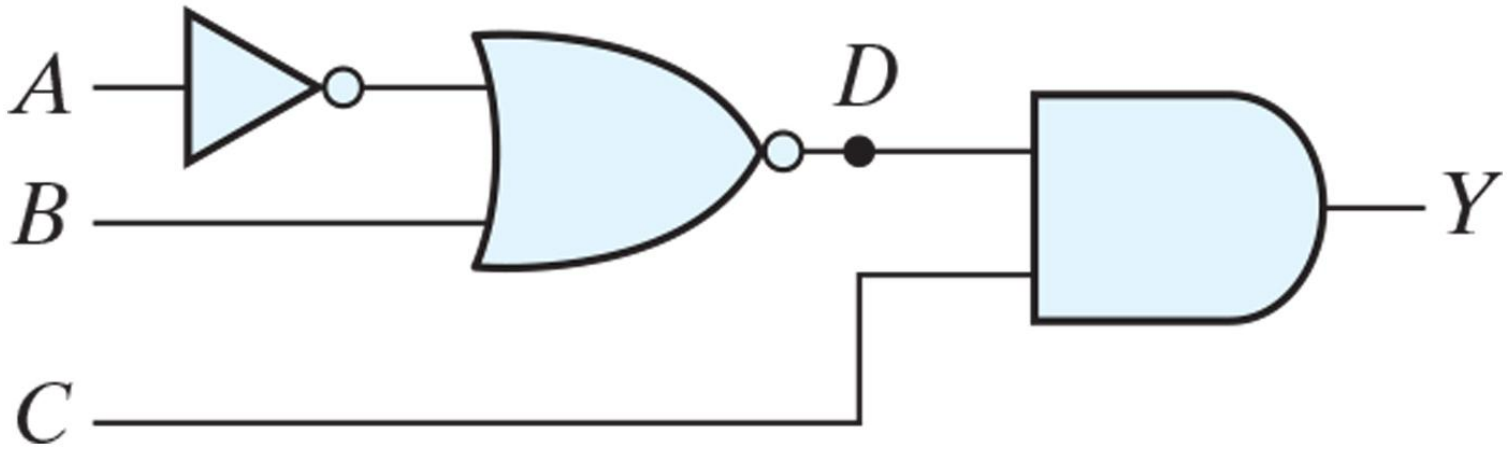


Figure 3.36
Entity-Architecture pair for *or_and_vhdl.*

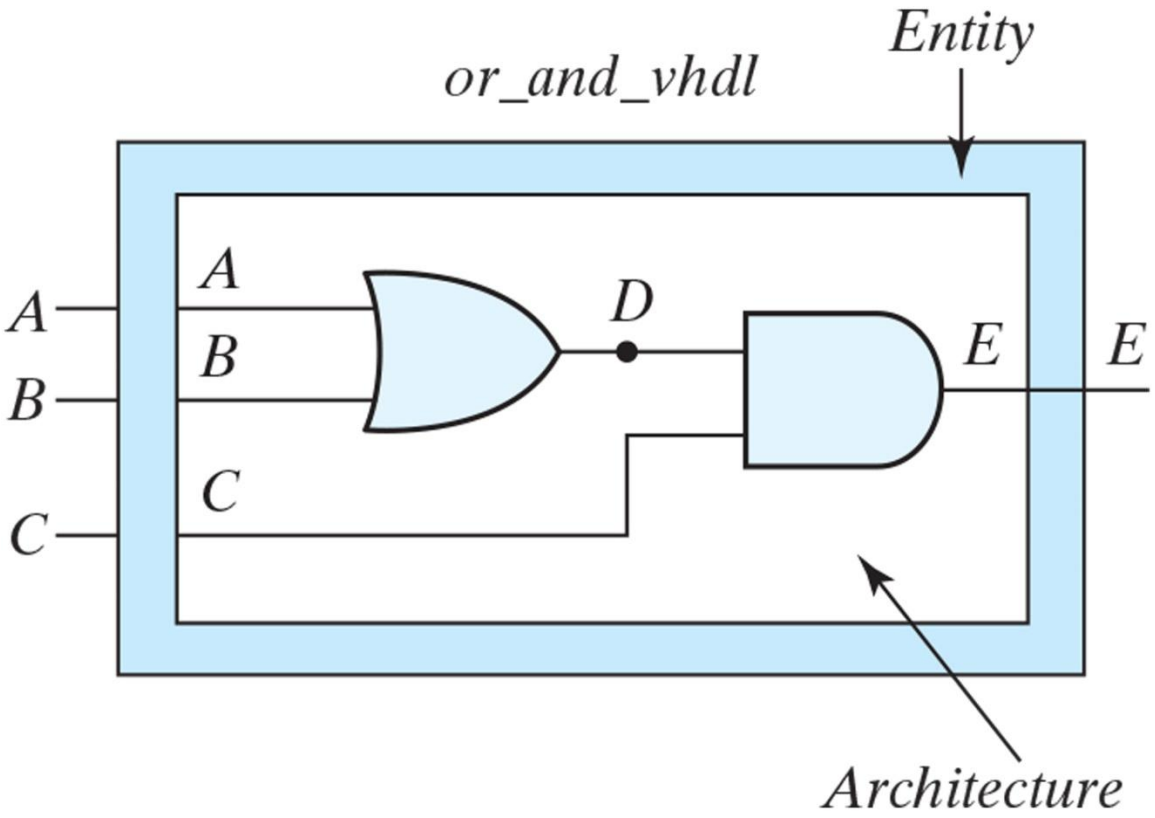


Figure PE3.13

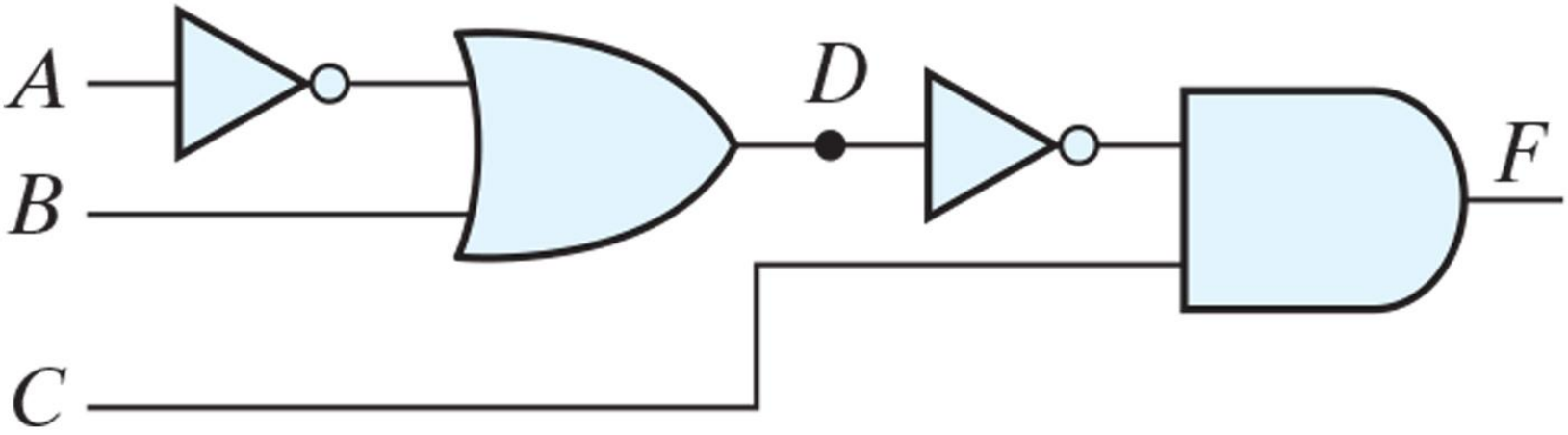


Figure 3.37
Schematic for *and_or_prop_delay.*

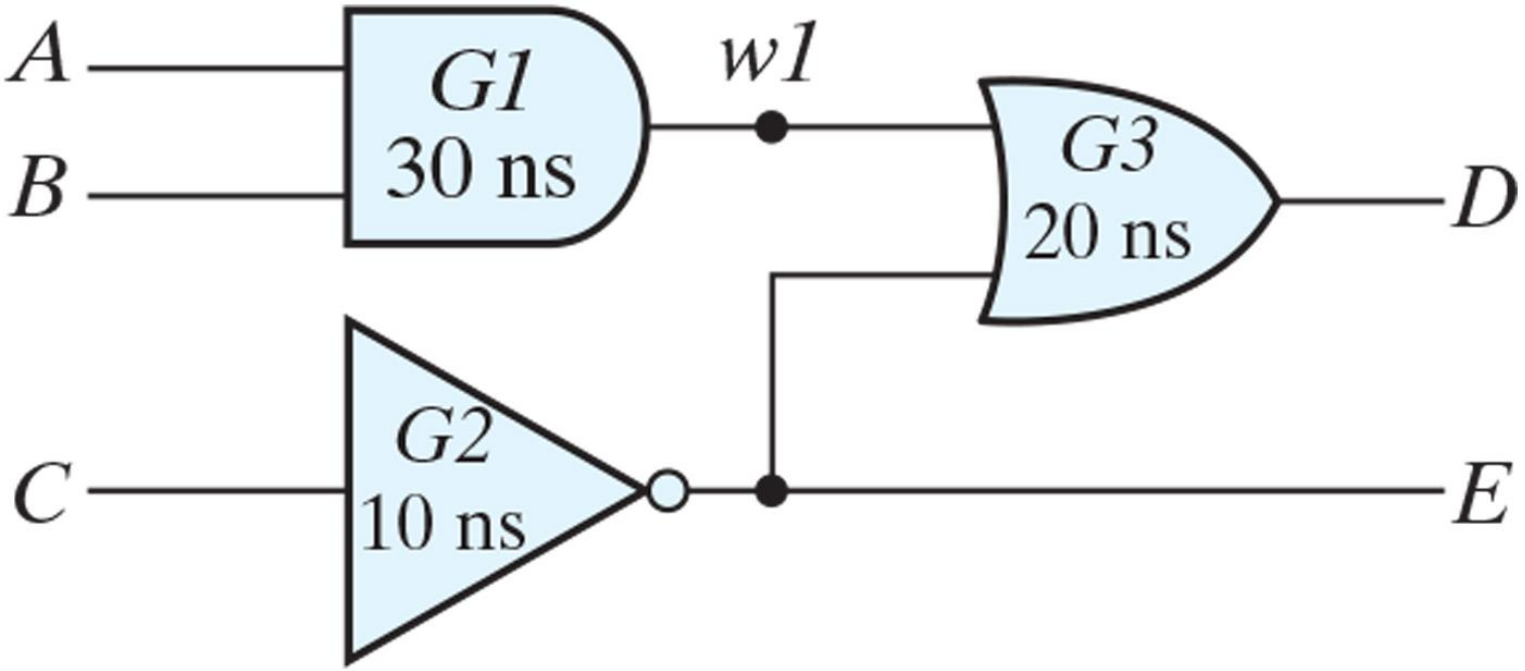


Table 3.5
Output of Gates after Delay.

	Time Units (ns)	Input	Output		
		A B C	*E*	*w1*	*D*
Initial	—	0 0 0	1	0	1
Change	—	1 1 1	1	0	1
	10	1 1 1	0	0	1
	20	1 1 1	0	0	1
	30	1 1 1	0	1	0
	40	1 1 1	0	1	0
	50	1 1 1	0	1	1

Figure 3.38
Simulation waveforms of *and_or_prop_delay*.

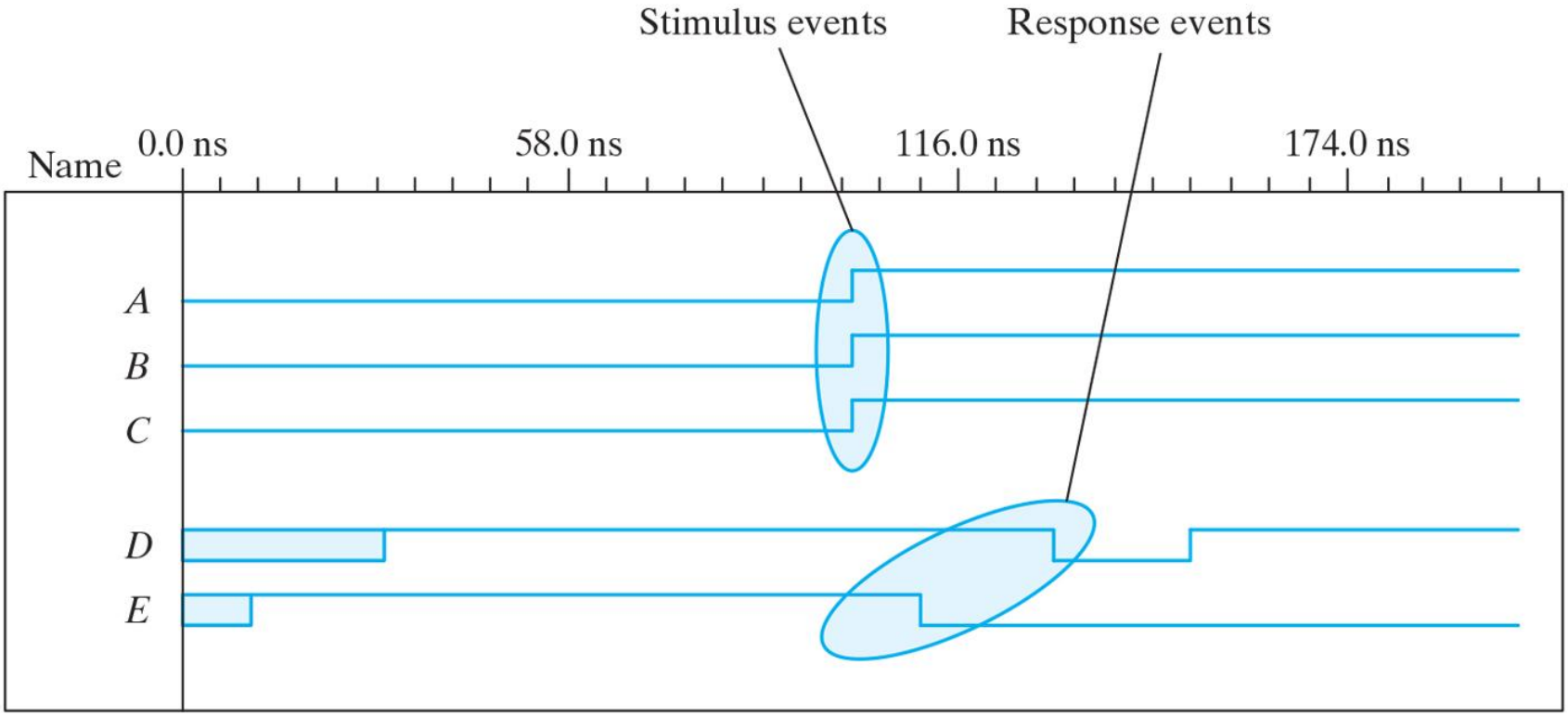


Figure 3.39
Entity-architecture for a structural model of *and_or_prop_delay_vhdl*.

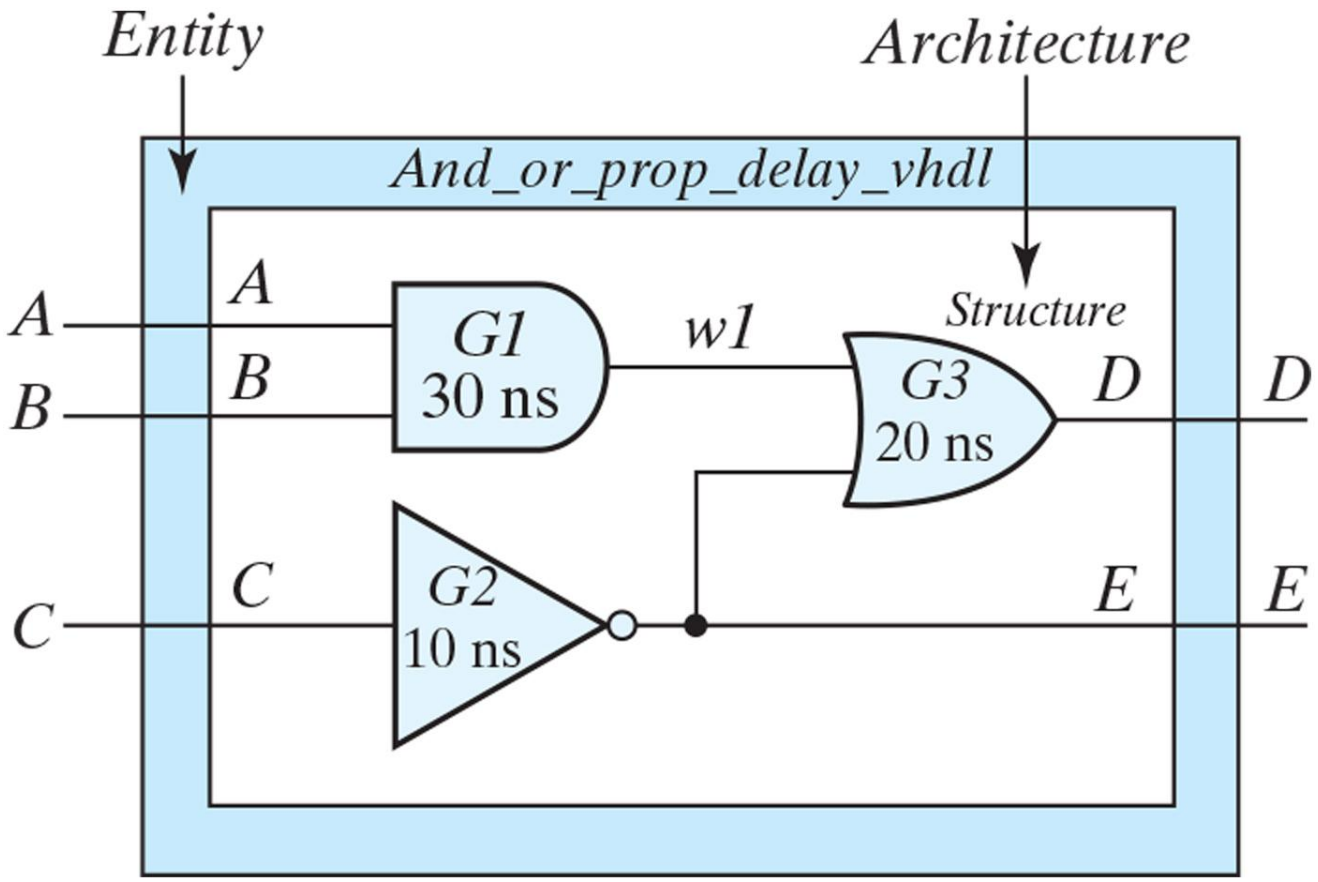


Table 3.6
Logic Symbols of the *IEEE_std_logic_1164* Package.

'U'	Uninitialized
'X'	Strong drive, unknown logic value
'0'	Strong drive, logic 0
'1'	Strong drive, logic 1
'Z'	High impedance
'W'	Weak drive, unknown logic value
'L'	Weak drive, logic 0
'H'	Weak drive, logic 1
'-'	Don't care

Figure 3.40
Schematic for *Circuit with_UDP_02467*.

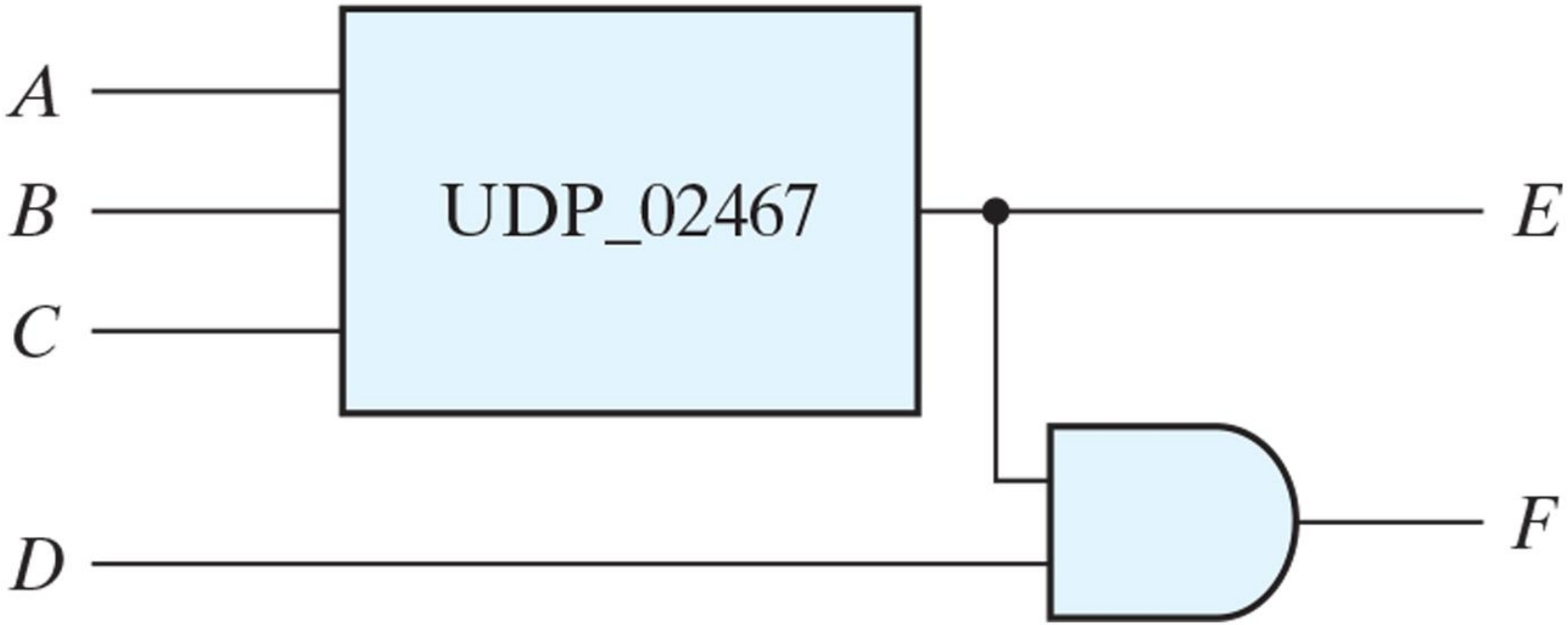


Figure P3.38