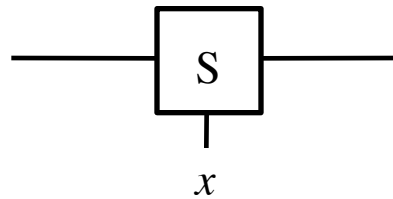


Chapter 2

Introduction to Logic Circuits

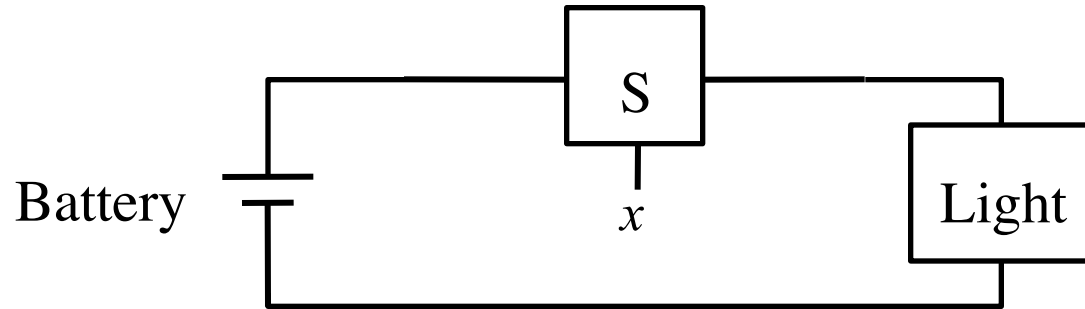


(a) Two states of a switch

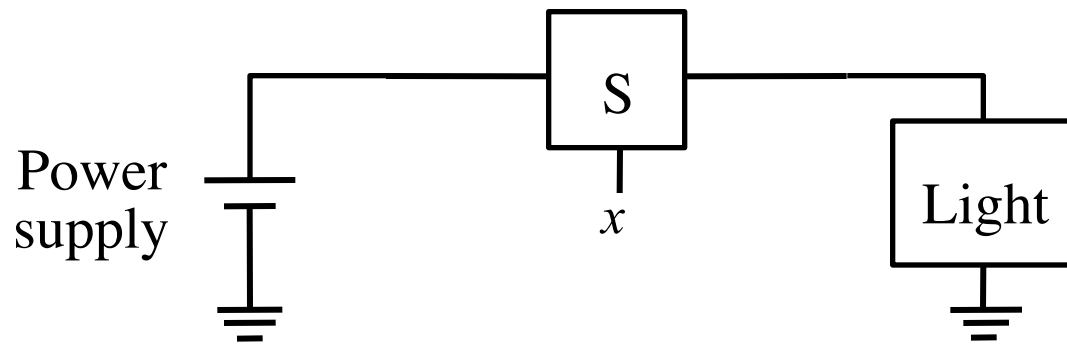


(b) Symbol for a switch

Figure 2.1. A binary switch.

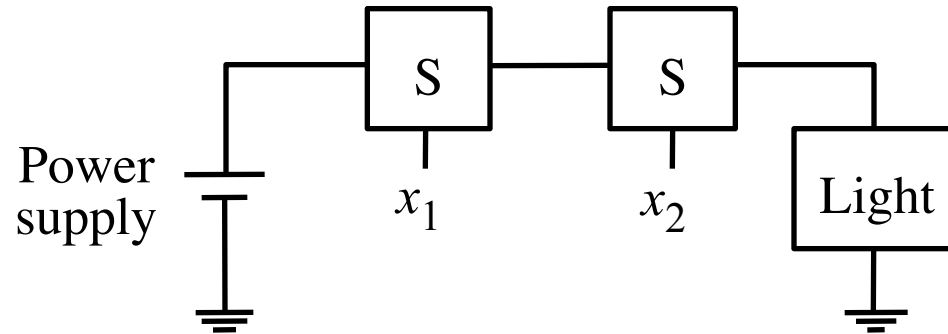


(a) Simple connection to a battery

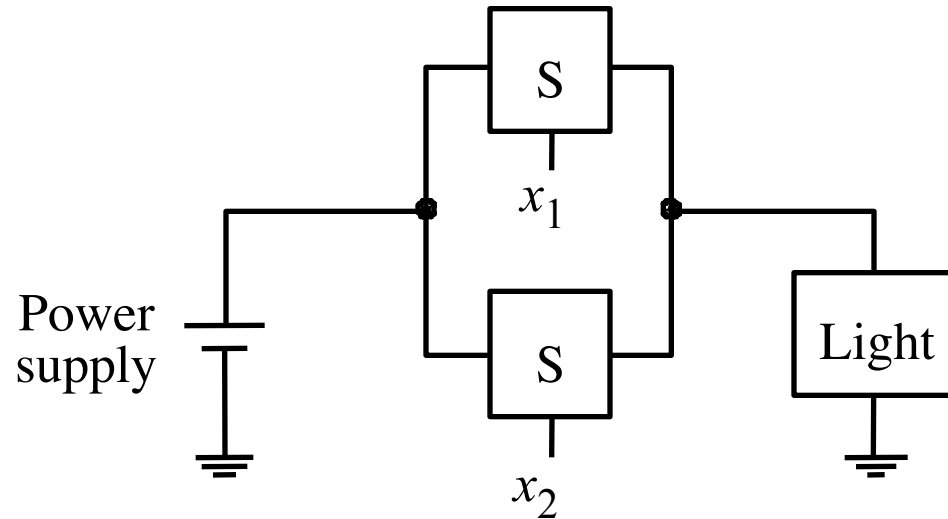


(b) Using a ground connection as the return path

Figure 2.2. A light controlled by a switch.



(a) The logical AND function (series connection)



(b) The logical OR function (parallel connection)

Figure 2.3. Two basic functions.

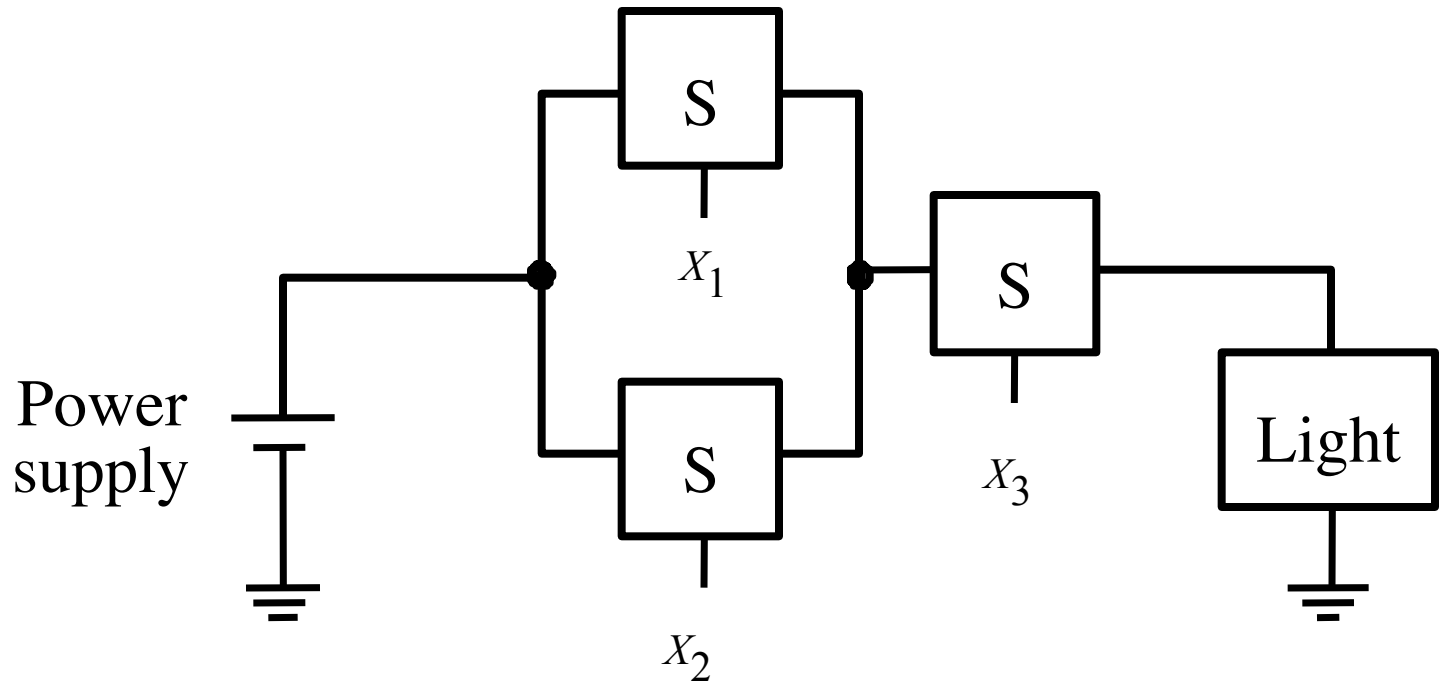


Figure 2.4. A series-parallel connection.

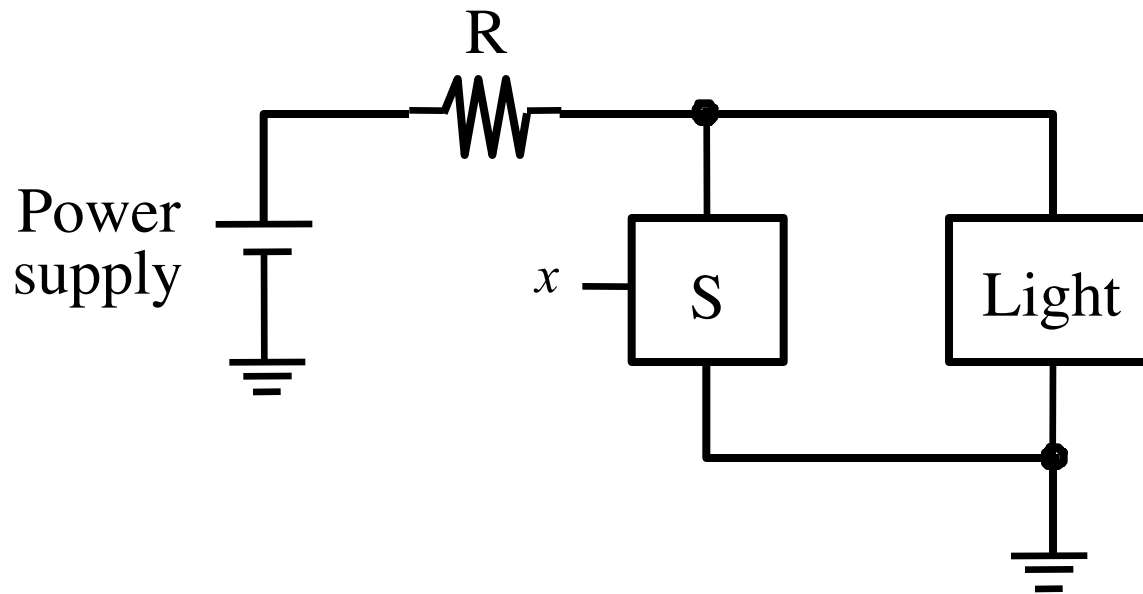


Figure 2.5. An inverting circuit.

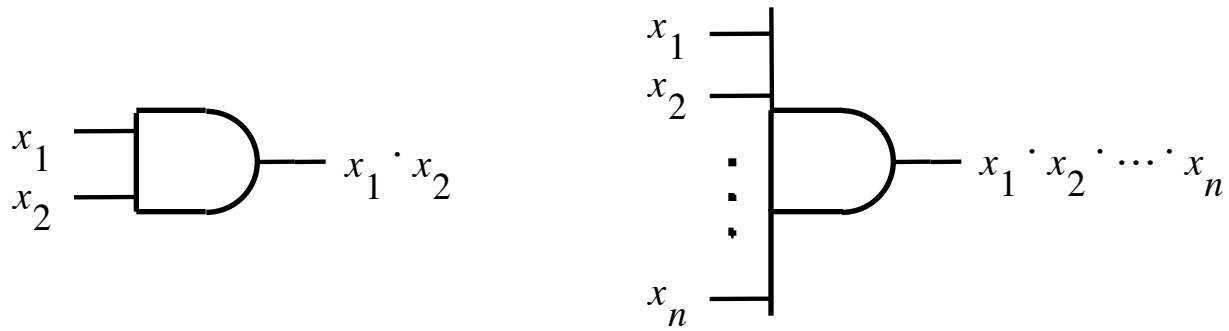
x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

AND
OR

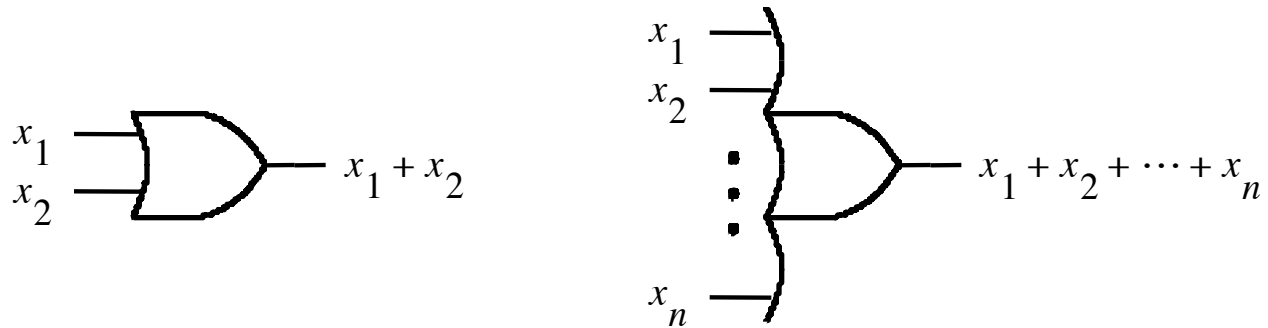
Figure 2.6. A truth table for the AND and OR operations.

A_1	A_2	A_3	$A_1 \cdot A_2 \cdot A_3$	$A_1 \vee A_2 \vee A_3$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

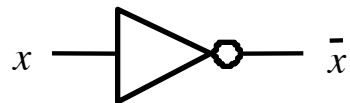
Figure 2.7. Three-input AND and OR operations.



(a) AND gates



(b) OR gates



(c) NOT gate

Figure 2.8. The basic gates.

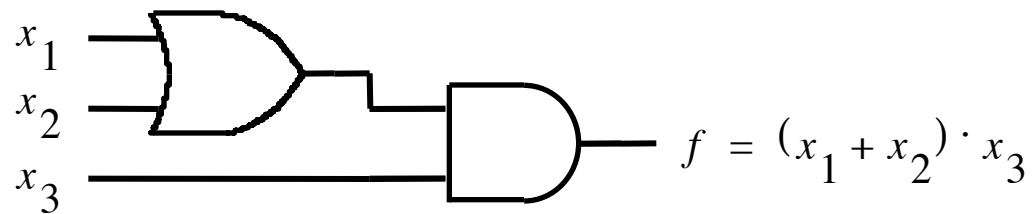
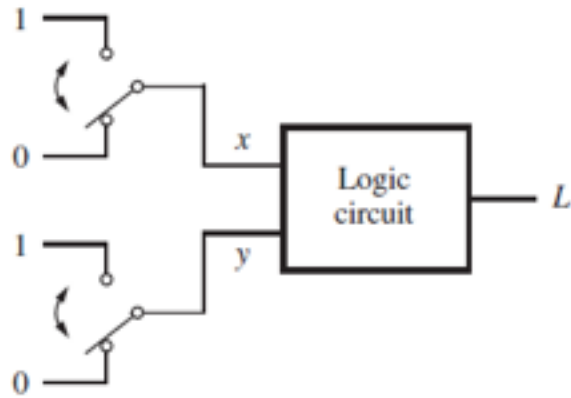


Figure 2.9. The function from Figure 2.4.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

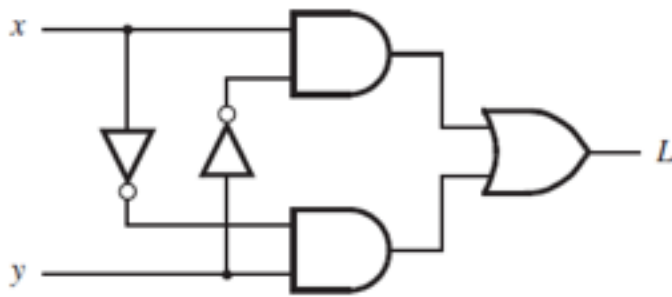
Figure 2.10. An example of logic networks.



(a) Two switches that control a light

x	y	L
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table



(c) Logic network



(d) XOR gate symbol

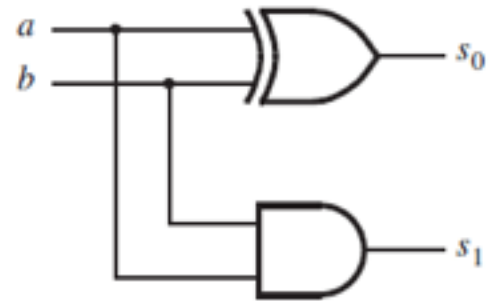
Figure 2.11. An example of a logic circuit.

$$\begin{array}{r}
 a \\
 +b \\
 \hline
 s_1 s_0
 \end{array}
 \qquad
 \begin{array}{cc}
 0 & 0 \\
 +0 & +1 \\
 \hline
 00 & 01
 \end{array}
 \qquad
 \begin{array}{cc}
 1 & 1 \\
 +0 & +1 \\
 \hline
 01 & 10
 \end{array}$$

(a) Evaluation of $S = a + b$

a	b	s_1	s_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0


(b) Truth table




(c) Logic network

Figure 2.12. Addition of binary numbers.

x	y	$x \cdot y$	$\overline{x \cdot y}$	\bar{x}	\bar{y}	$\bar{x} + \bar{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0



LHS



RHS

Figure 2.13. Proof of DeMorgan's theorem in 15a.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.14. The Venn diagram representation.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.15. Verification of the distributive property.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.16. Verification of $x \cdot y + \bar{x} \cdot z + y \cdot z = x \cdot y + \bar{x} \cdot z$.

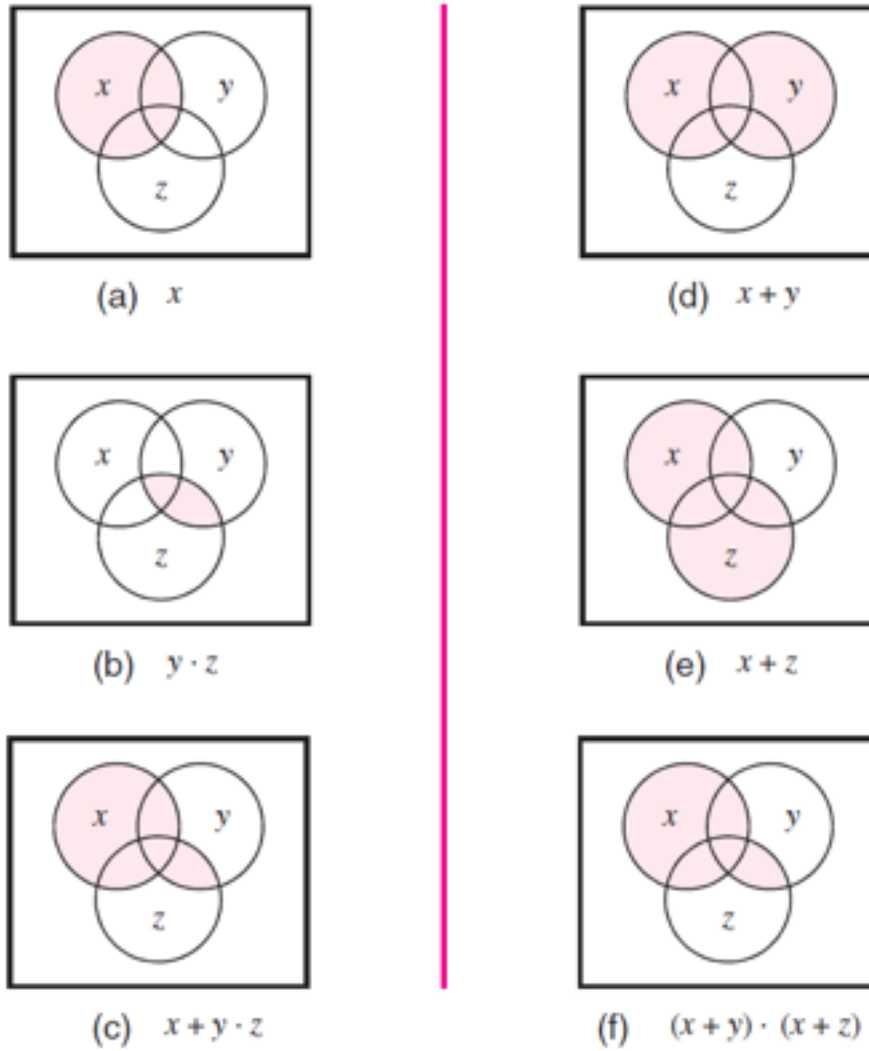


Figure 2.17. Proof of the distributive property 12b.

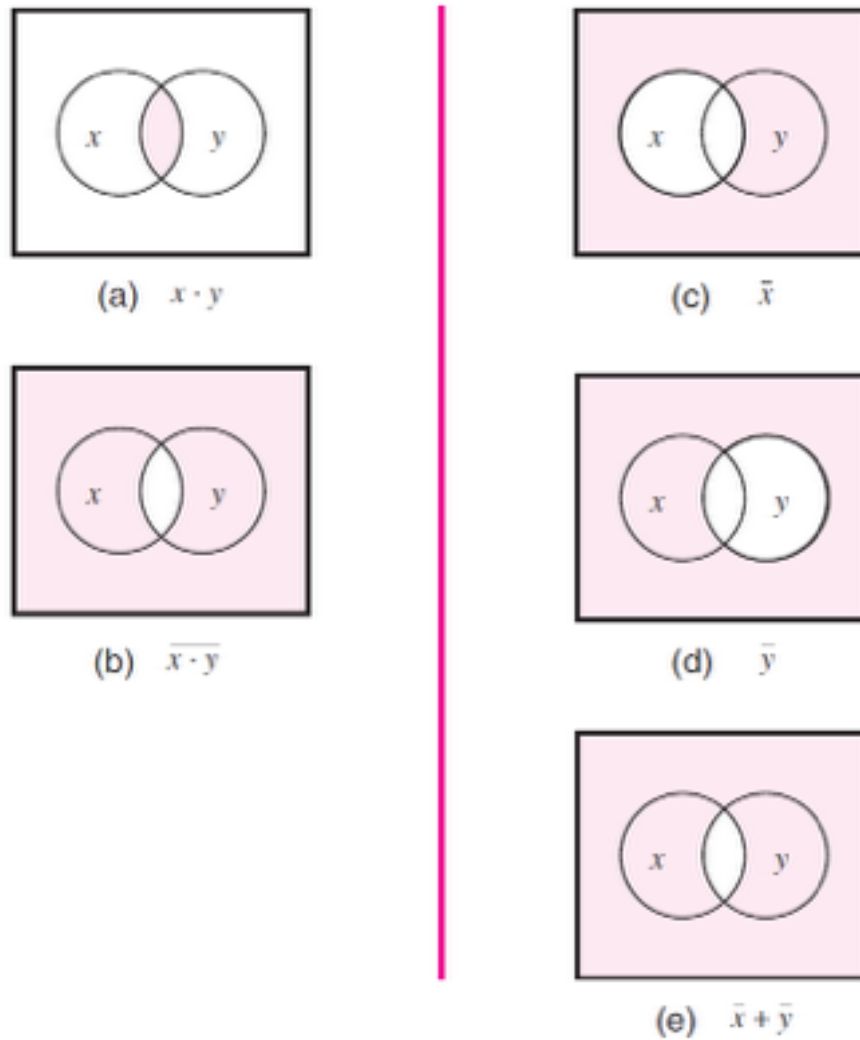
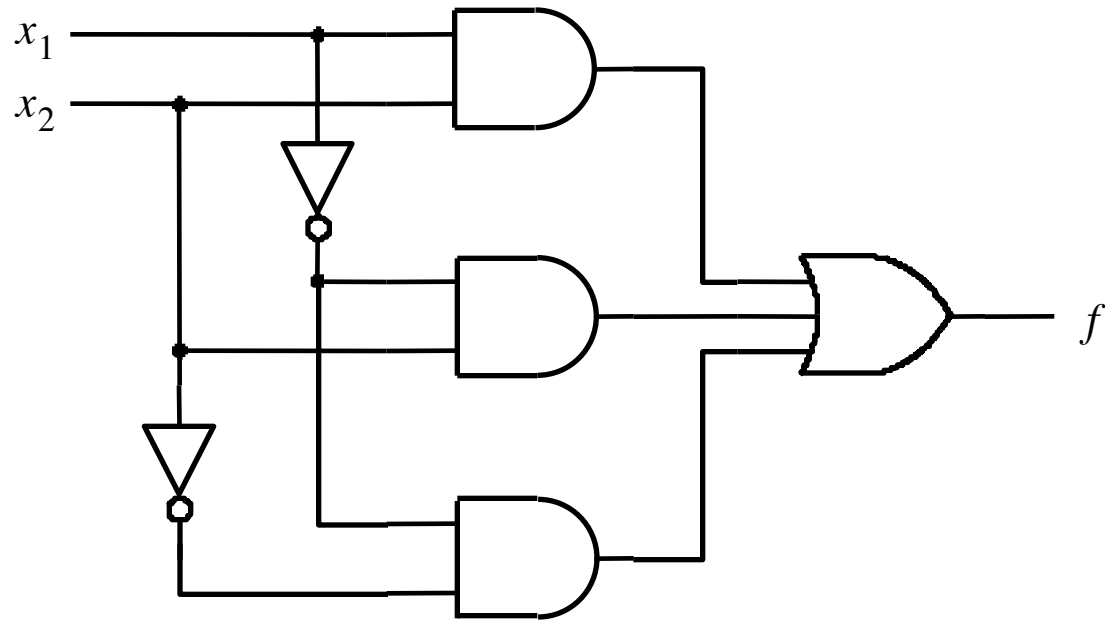


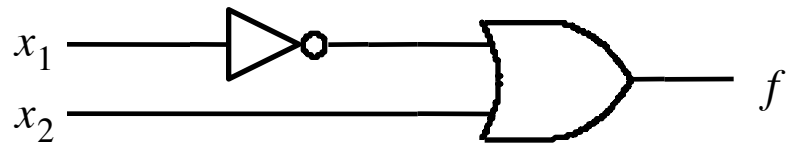
Figure 2.18. Proof of DeMorgan's theorem 15a.

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

Figure 2.19. A function to be synthesized.

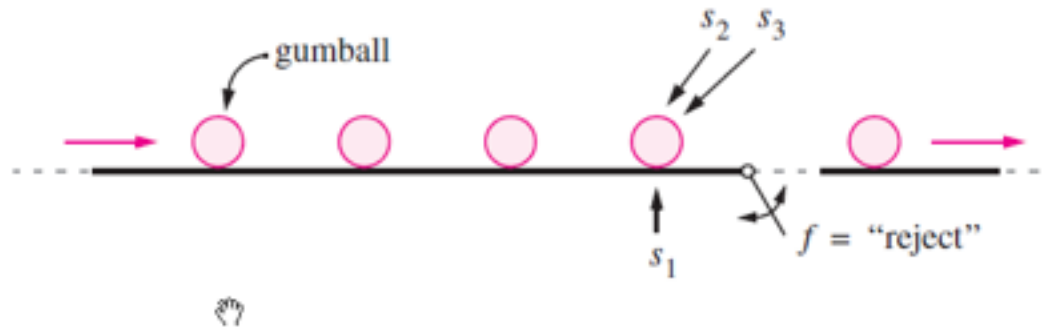


(a) Canonical sum-of-products



(b) Minimal-cost realization

Figure 2.20. Two implementations of the function in Figure 2.19.



(a) Conveyor and sensors

s_1	s_2	s_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b) Truth table

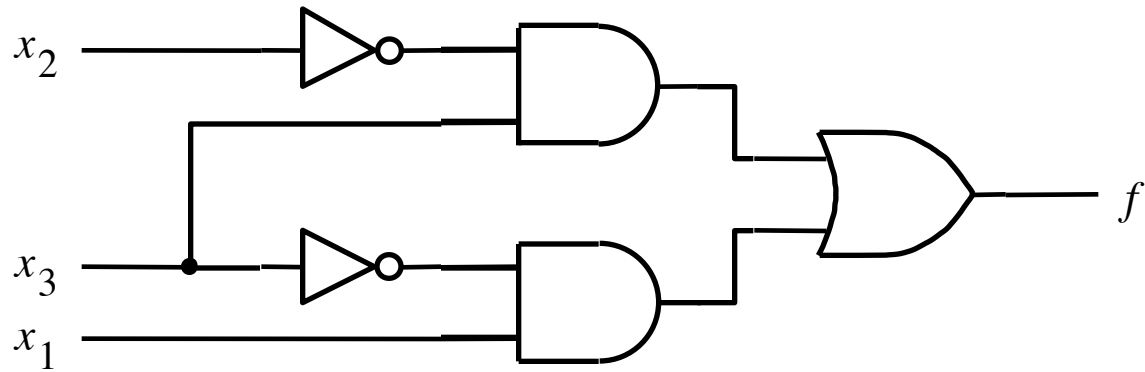
Figure 2.21. A bubble gumball factory.

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \bar{x}_1\bar{x}_2x_3$	$M_1 = x_1 + x_2 + \bar{x}_3$
2	0	1	0	$m_2 = \bar{x}_1x_2\bar{x}_3$	$M_2 = x_1 + \bar{x}_2 + x_3$
3	0	1	1	$m_3 = \bar{x}_1x_2x_3$	$M_3 = x_1 + \bar{x}_2 + \bar{x}_3$
4	1	0	0	$m_4 = x_1\bar{x}_2\bar{x}_3$	$M_4 = \bar{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\bar{x}_2x_3$	$M_5 = \bar{x}_1 + x_2 + \bar{x}_3$
6	1	1	0	$m_6 = x_1x_2\bar{x}_3$	$M_6 = \bar{x}_1 + \bar{x}_2 + x_3$
7	1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

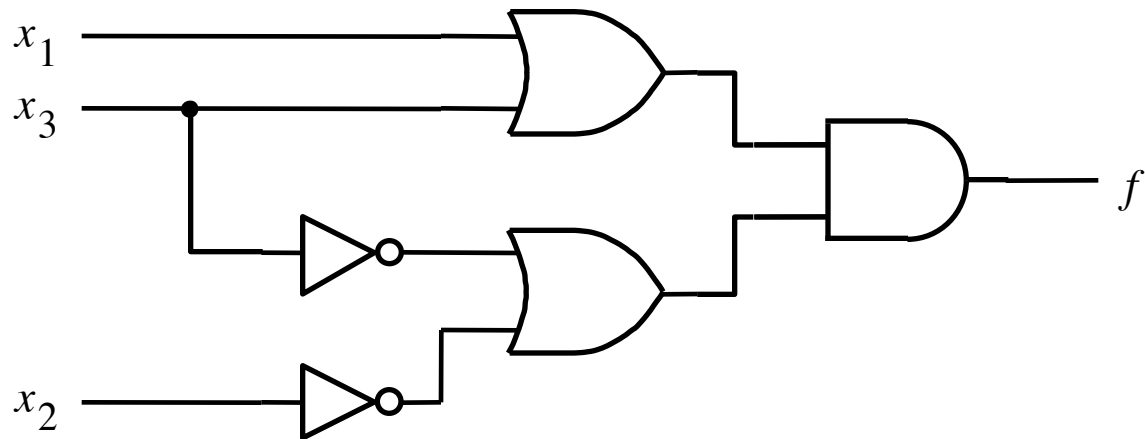
Figure 2.22 Three-variable minterms and maxterms.

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Figure 2.23. A three-variable function.

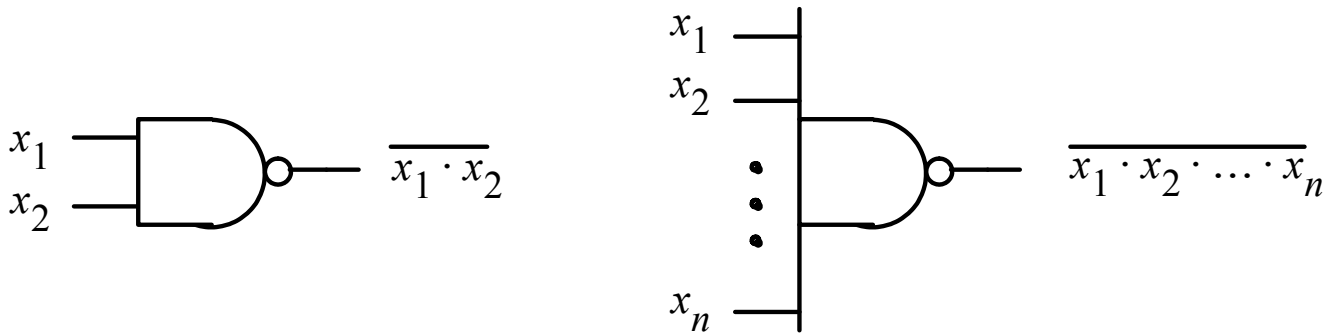


(a) A minimal sum-of-products realization

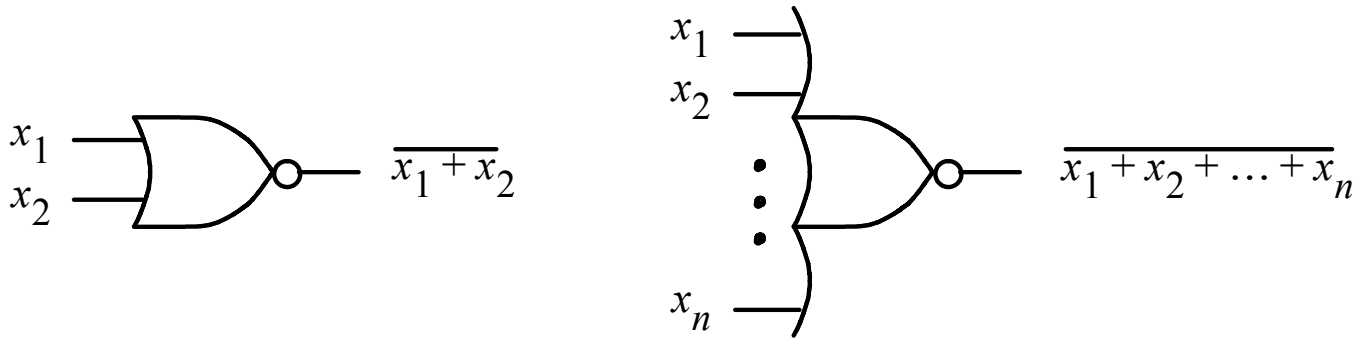


(b) A minimal product-of-sums realization

Figure 2.24. Two realizations of a function in Figure 2.23.

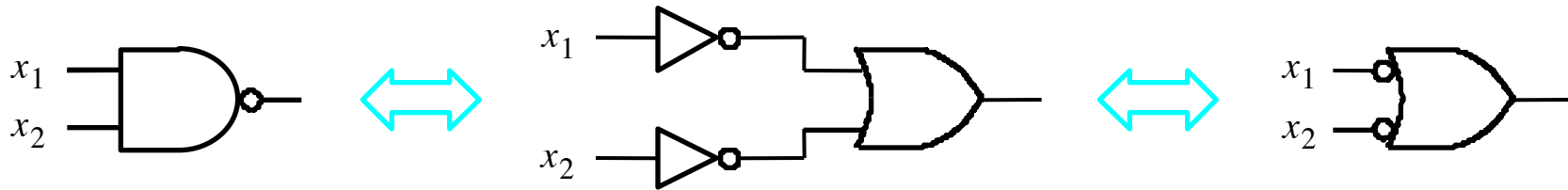


(a) NAND gates

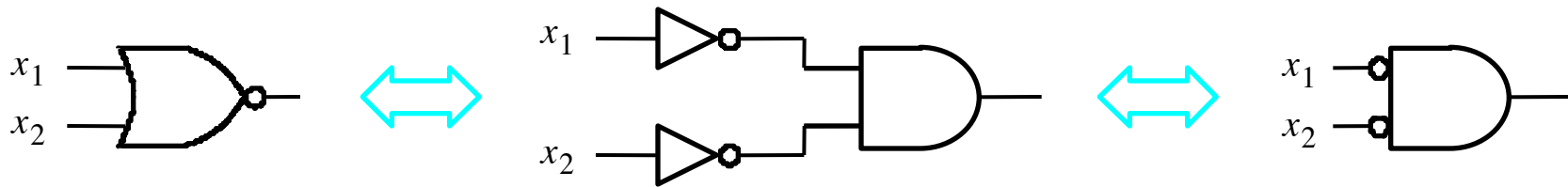


(b) NOR gates

Figure 2.25. NAND and NOR gates.



(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$



(b) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

Figure 2.26. DeMorgan's theorem in terms of logic gates.

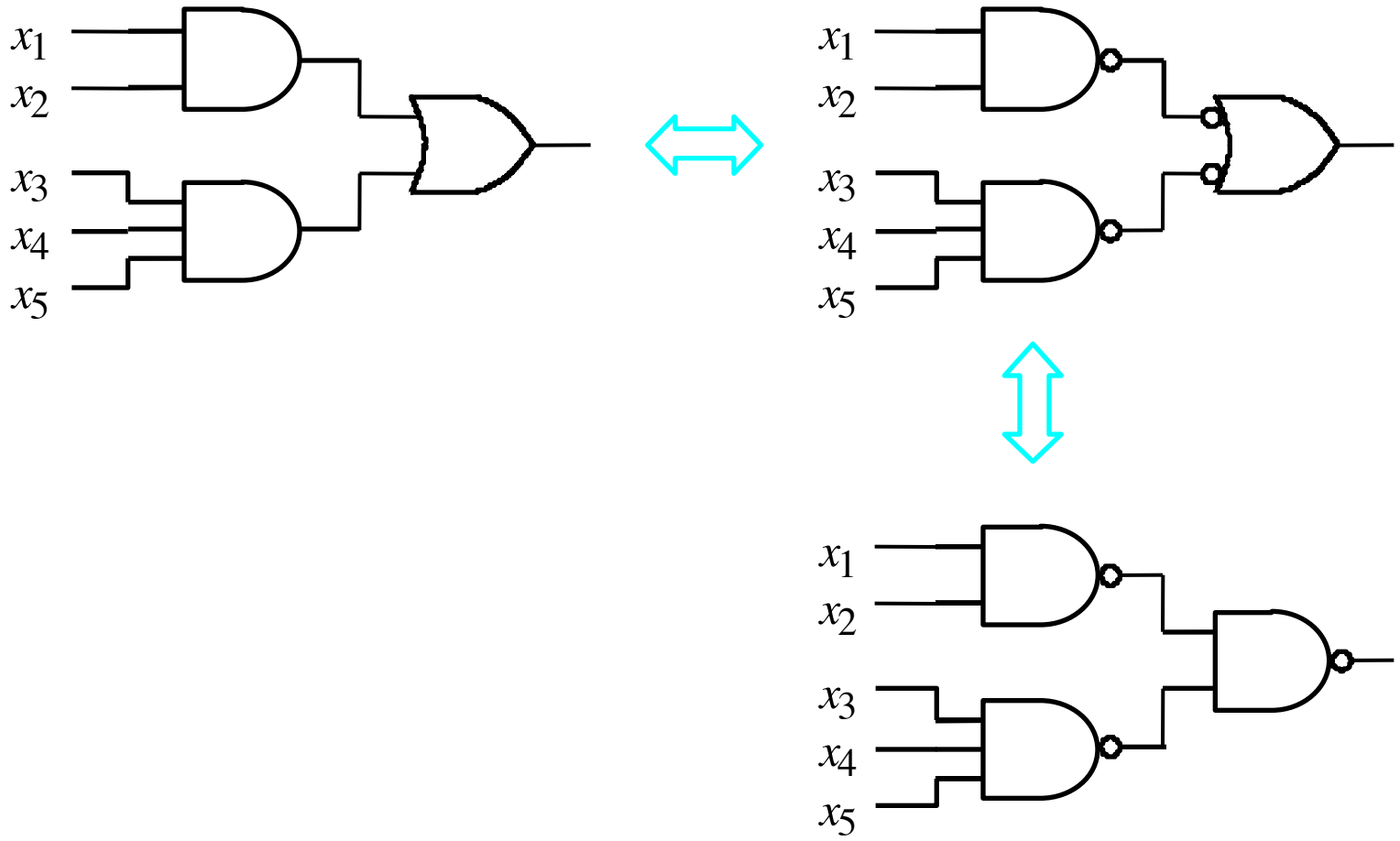


Figure 2.27. Using NAND gates to implement a sum-of-products.

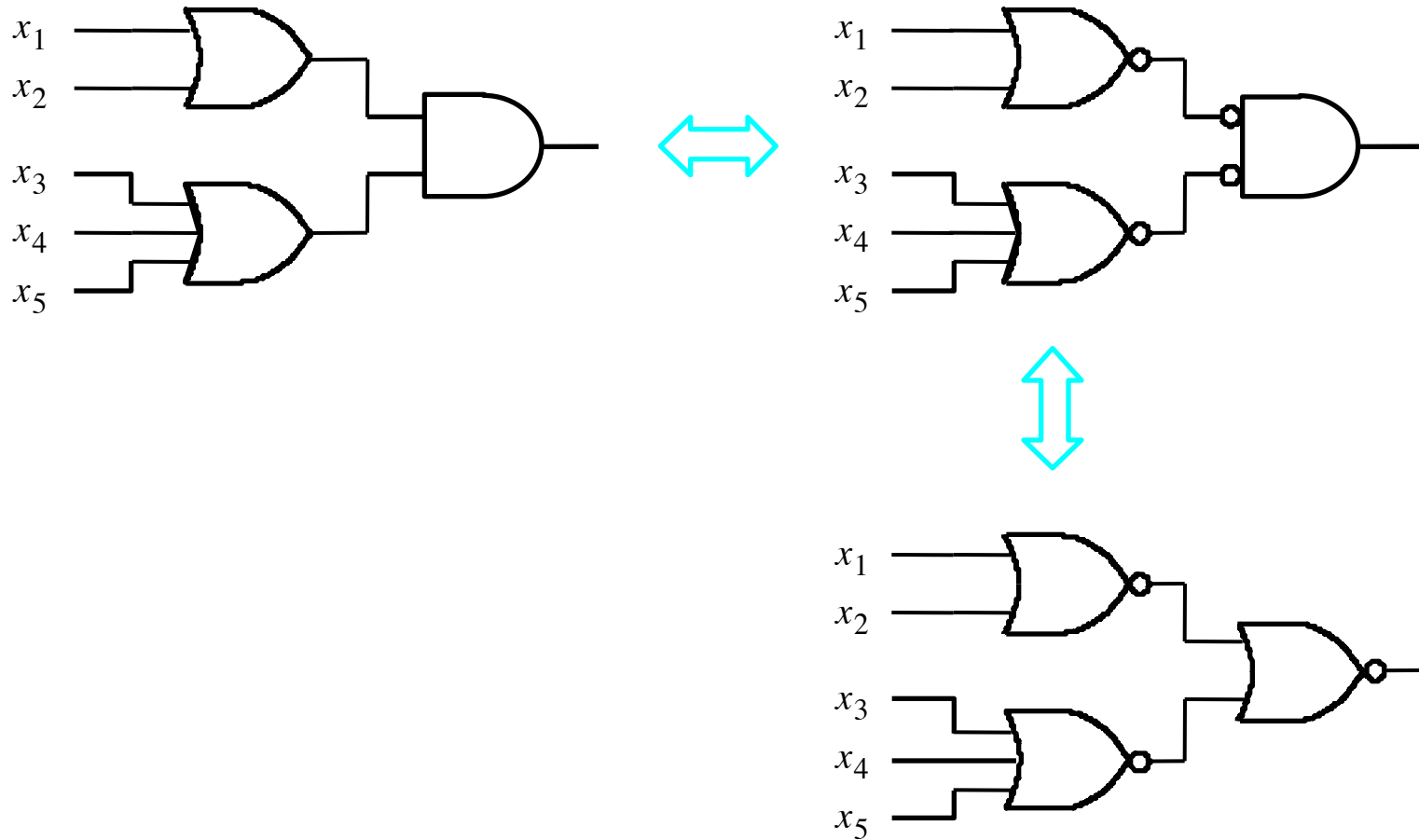
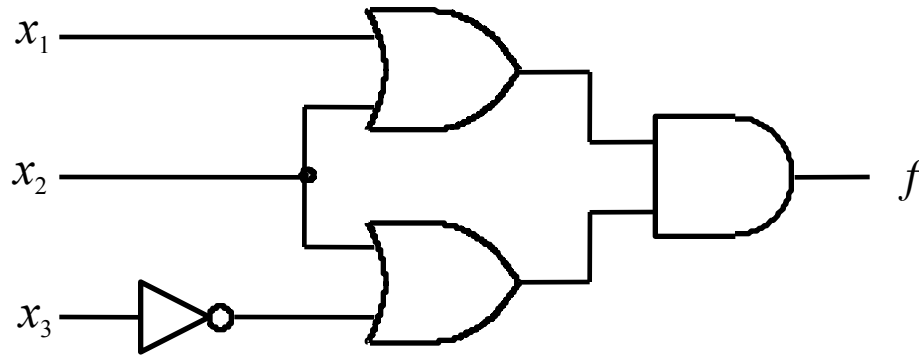
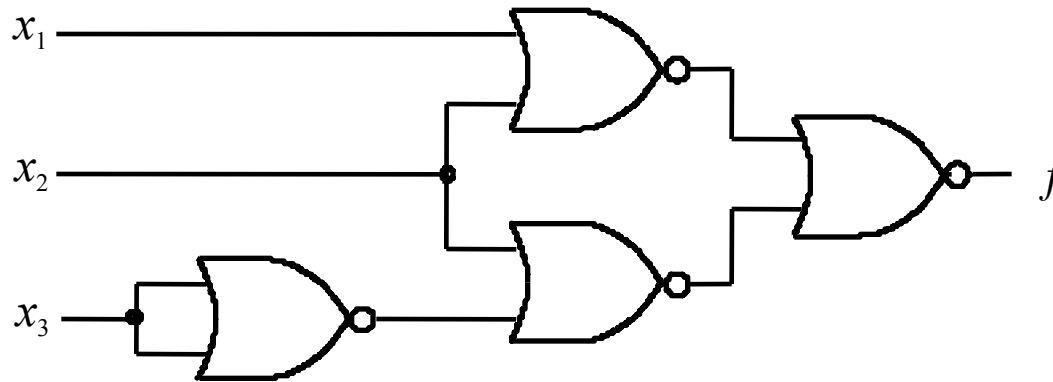


Figure 2.28. Using NOR gates to implement a product-of sums.

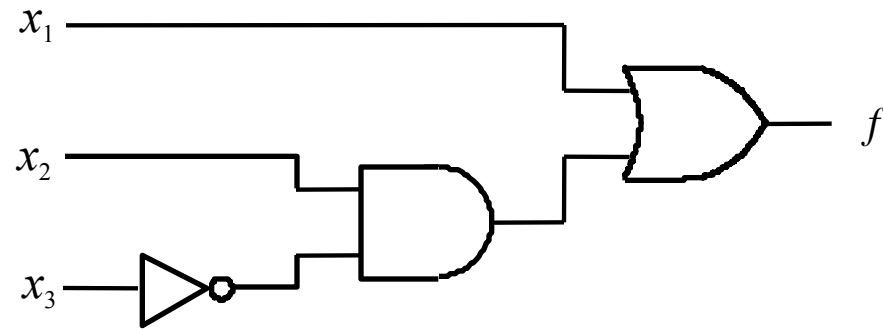


(a) POS implementation

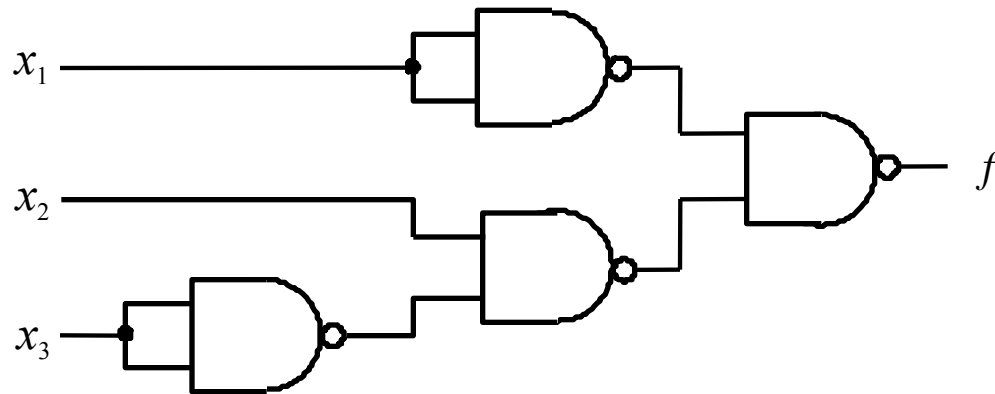


(b) NOR implementation

Figure 2.29 NOR-gate realization of the function in Example 2.11.



(a) SOP implementation



(b) NAND implementation

Figure 2.30. NAND-gate realization of the function in Example 2.10.

x_1	x_2	x_3	f
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

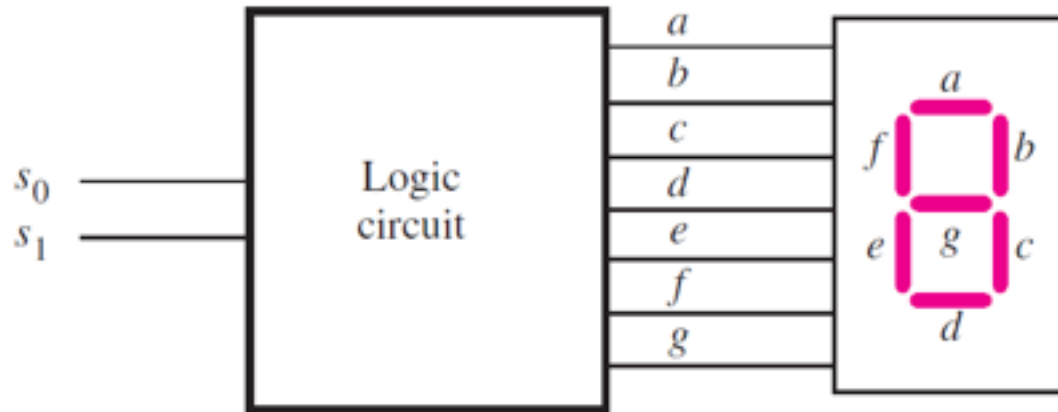
Figure 2.31. Truth table for a three-way light control.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.32. Implementation of the function in Figure 2.31.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.33. Implementation of a multiplexer.



(a) Logic circuit and 7-segment display

	s_1	s_0	a	b	c	d	e	f	g
0	0	0	1	1	1	1	1	1	0
1	0	1	0	1	1	0	0	0	0
2	1	0	1	1	0	1	1	0	1

(b) Truth table

Figure 2.34. Display of numbers.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.35. A typical CAD system.

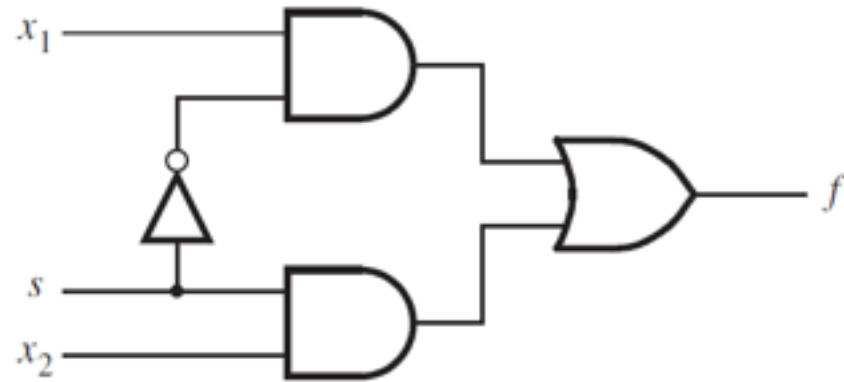


Figure 2.36. The logic circuit for a multiplexer.

```
module example1 (x1, x2, s, f);  
  input x1, x2, s;  
  output f;  
  
  not (k, s);  
  and (g, k, x1);  
  and (h, s, x2);  
  or (f, g, h);  
  
endmodule
```

Figure 2.37. Verilog code for the circuit in Figure 2.36.

```
module example2 (x1, x2, x3, x4, f, g, h);  
    input x1, x2, x3, x4;  
    output f, g, h;  
  
    and (z1, x1, x3);  
    and (z2, x2, x4);  
    or (g, z1, z2);  
    or (z3, x1, ~x3);  
    or (z4, ~x2, x4);  
    and (h, z3, z4);  
    or (f, g, h);  
  
endmodule
```

Figure 2.38. Verilog code for a four-input circuit.

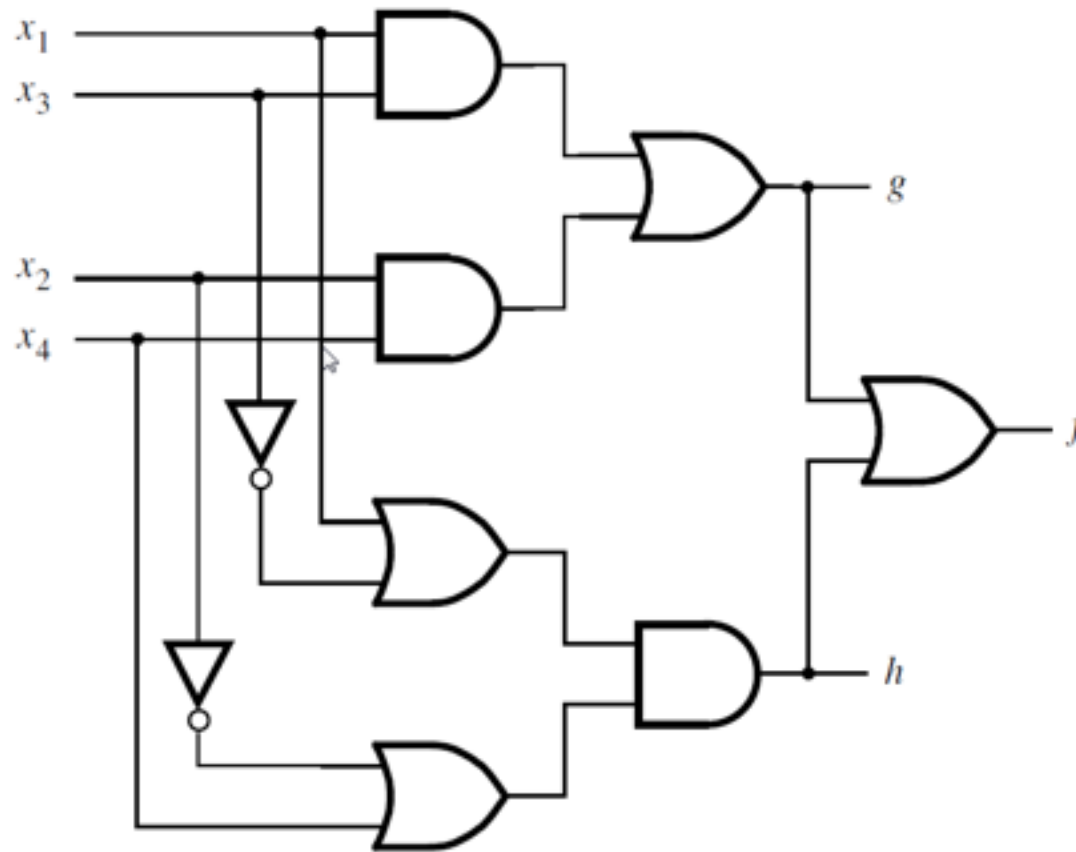


Figure 2.39. Logic circuit for the code in Figure 2.38.


```
module example3 (x1, x2, s, f);  
  input x1, x2, s;  
  output f;  
  
  assign f = (~s & x1) | (s & x2);  
  
endmodule
```

Figure 2.40. Using the continuous assignment to specify the circuit in Figure 2.36.

```
module example4 (x1, x2, x3, x4, f, g, h);  
    input x1, x2, x3, x4;  
    output f, g, h;  
  
    assign g = (x1 & x3) | (x2 & x4);  
    assign h = (x1 | ~x3) & (~x2 | x4);  
    assign f = g | h;  
  
endmodule
```

Figure 2.41. Using the continuous assignment to specify the circuit in Figure 2.39.

```
// Behavioral specification
module example5 (x1, x2, s, f);
    input x1, x2, s;
    output f; 1
    reg f;

    always @(x1 or x2 or s)
        if (s == 0)
            f = x1;
        else
            f = x2;

endmodule
```

Figure 2.42. Behavioral specification of the circuit in Figure 2.36.

```
// Behavioral specification
module example5 (input x1, x2, s, output reg f);

    always @(x1, x2, s)
        if (s == 0)
            f = x1;
        else
            f = x2;

endmodule
```

Figure 2.43. A more compact version of the code in Figure 2.42.

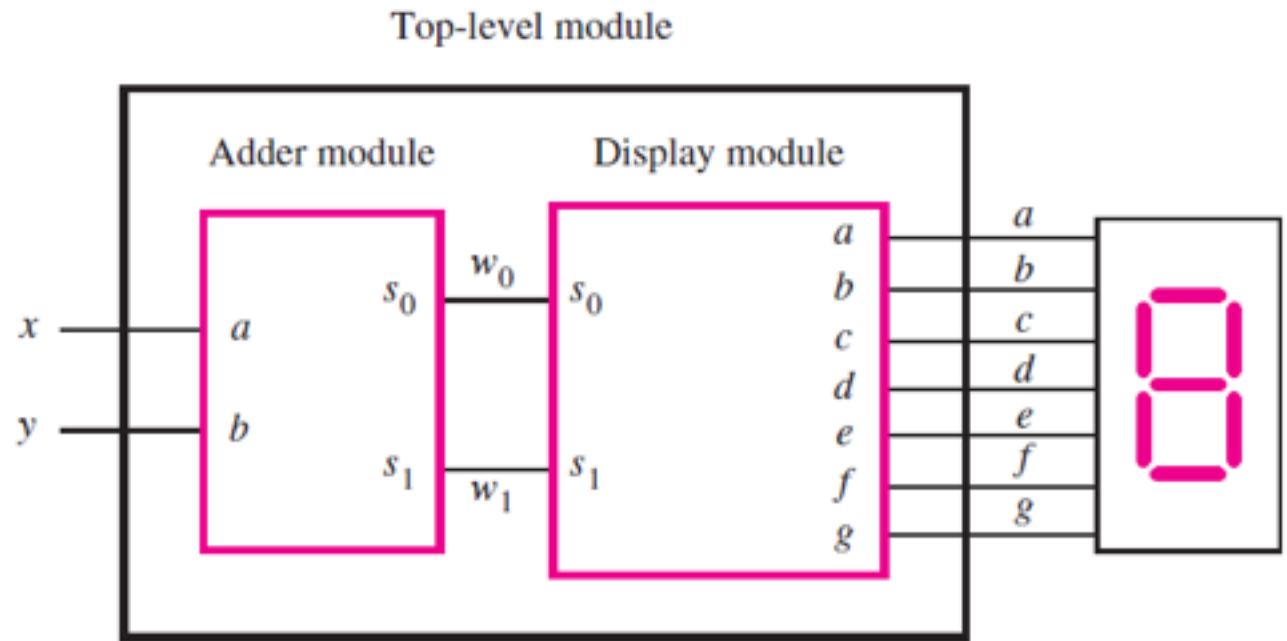


Figure 2.44. A logic circuit with two modules.

```
// An adder module
module adder (a, b, s1, s0);
    input a, b;
    output s1, s0;

    assign s1 = a & b;
    assign s0 = a ^ b;

endmodule
```

Figure 2.45. Verilog specification of the circuit in Figure 2.12.

```
// A module for driving a 7-segment display
module display (s1, s0, a, b, c, d, e, f, g);
    input s1, s0;
    output a, b, c, d, e, f, g;

    assign a = ~s0;
    assign b = 1;
    assign c = ~s1;
    assign d = ~s0;
    assign e = ~s0;
    assign f = ~s1 & ~s0;
    assign g = s1 & ~s0;

endmodule
```

Figure 2.46. Verilog specification of the circuit in Figure 2.34.

```
module adder_display (x, y, a, b, c, d, e, f, g);  
    input x, y;  
    output a, b, c, d, e, f, g;  
    wire w1, w0;  
  
    adder U1 (x, y, w1, w0);  
    display U2 (w1, w0, a, b, c, d, e, f, g);  
  
endmodule
```

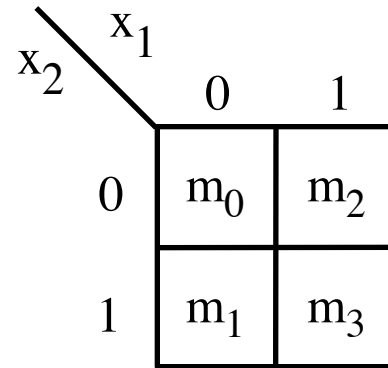
Figure 2.47. Hierarchical Verilog code for the circuit in Figure 2.44.

Row number	x_1	x_2	x_3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Figure 2.48 The function $f(x_1, x_2, x_3) = \sum m(0, 2, 4, 5, 6)$.

x_1	x_2	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

(a) Truth table



(b) Karnaugh map

Figure 2.49. Location of two-variable minterms.

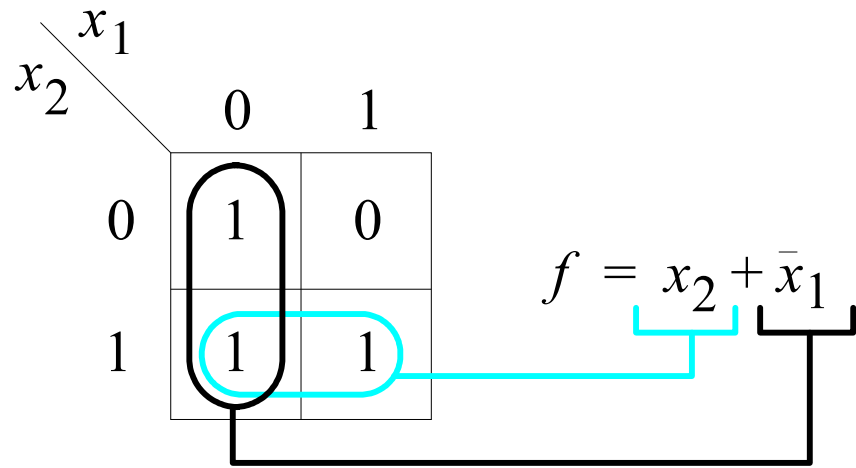


Figure 2.50. The function of Figure 2.19.

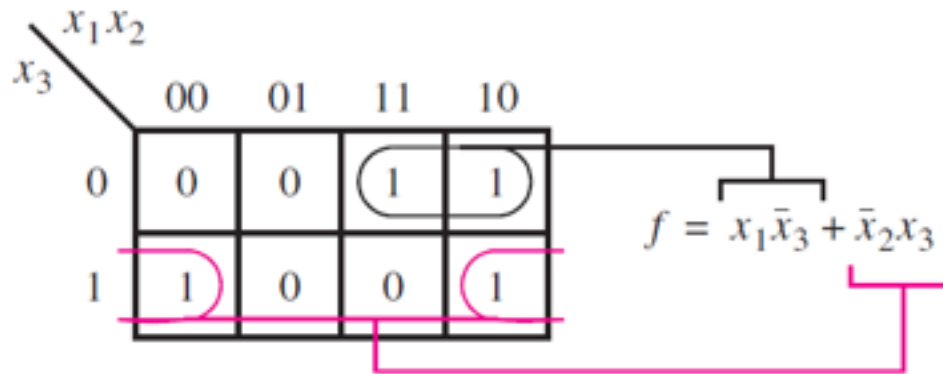
x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

(a) Truth table

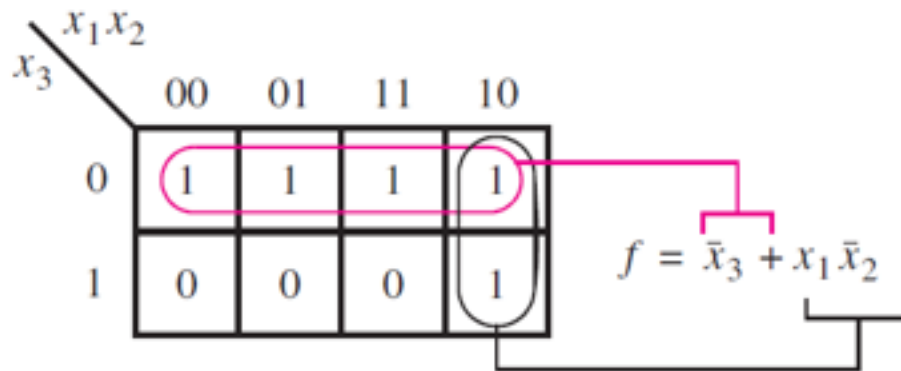
		x_1x_2			
		00	01	11	10
x_3	0	m_0	m_2	m_6	m_4
	1	m_1	m_3	m_7	m_5

(b) Karnaugh map

Figure 2.51. Location of three-variable minterms.



(a) The function of Figure 2.23



(b) The function of Figure 2.48

Figure 2.52. Examples of three-variable Karnaugh maps.

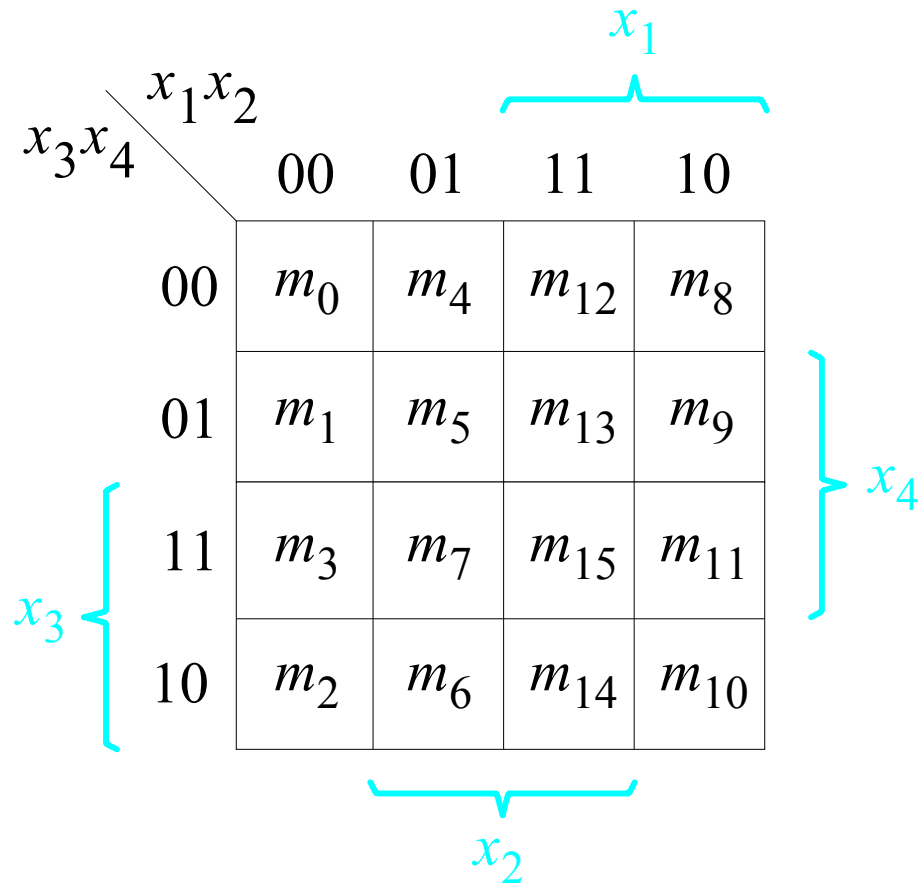


Figure 2.53. A four-variable Karnaugh map.

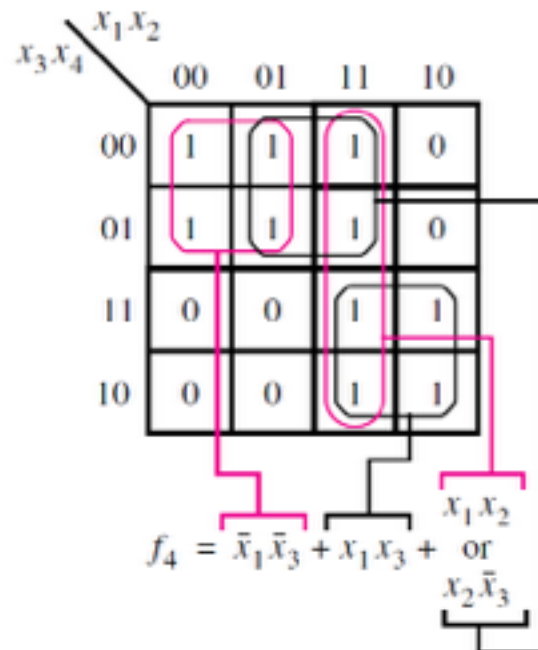
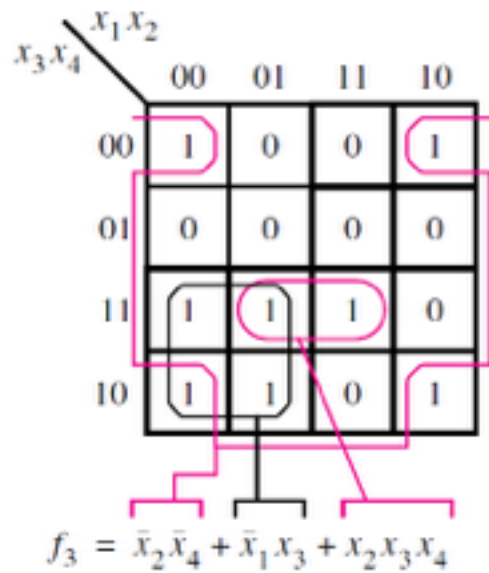
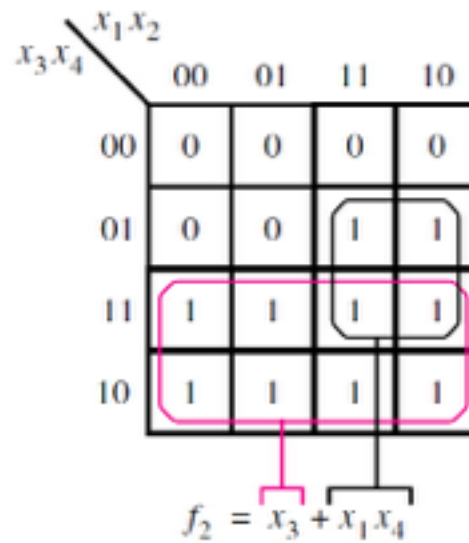
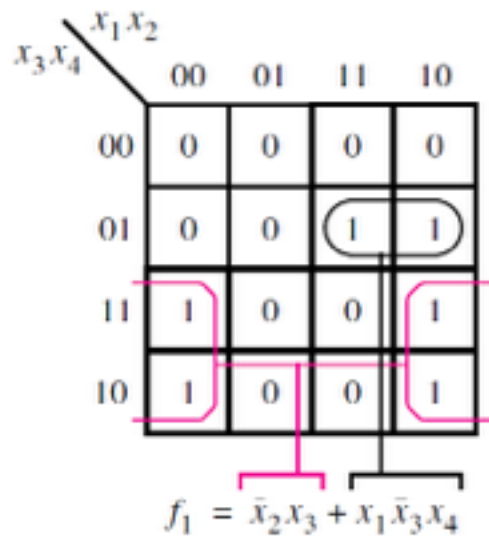


Figure 2.54. Examples of four-variable Karnaugh maps.

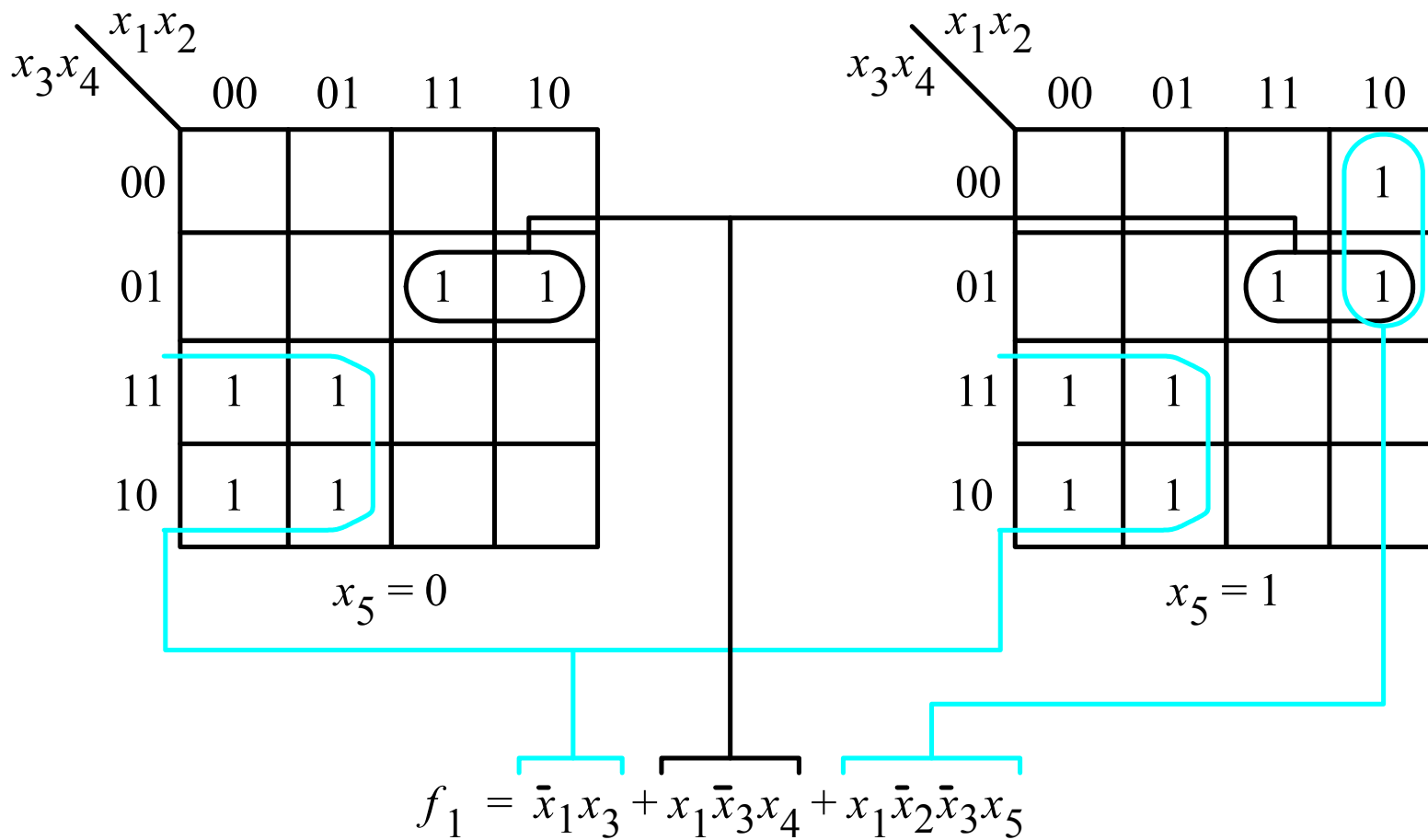


Figure 2.55. A five-variable Karnaugh map.

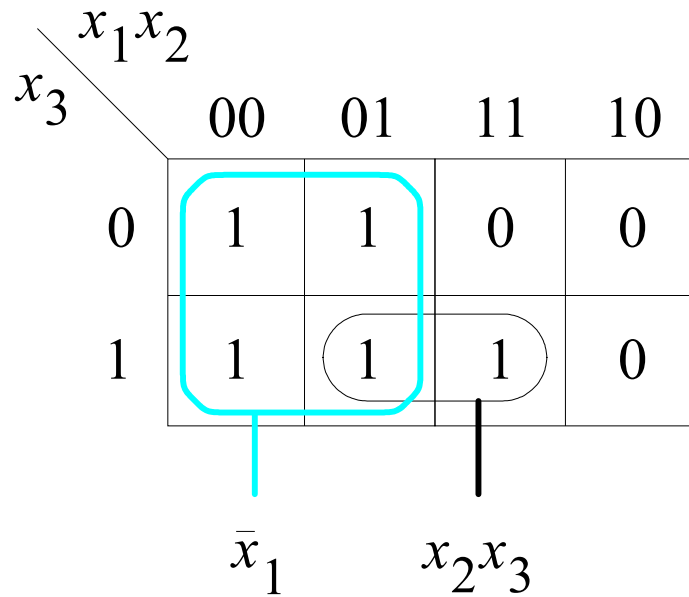


Figure 2.56. Three-variable function $f(x_1, x_2, x_3) = \Sigma m(0, 1, 2, 3, 7)$.

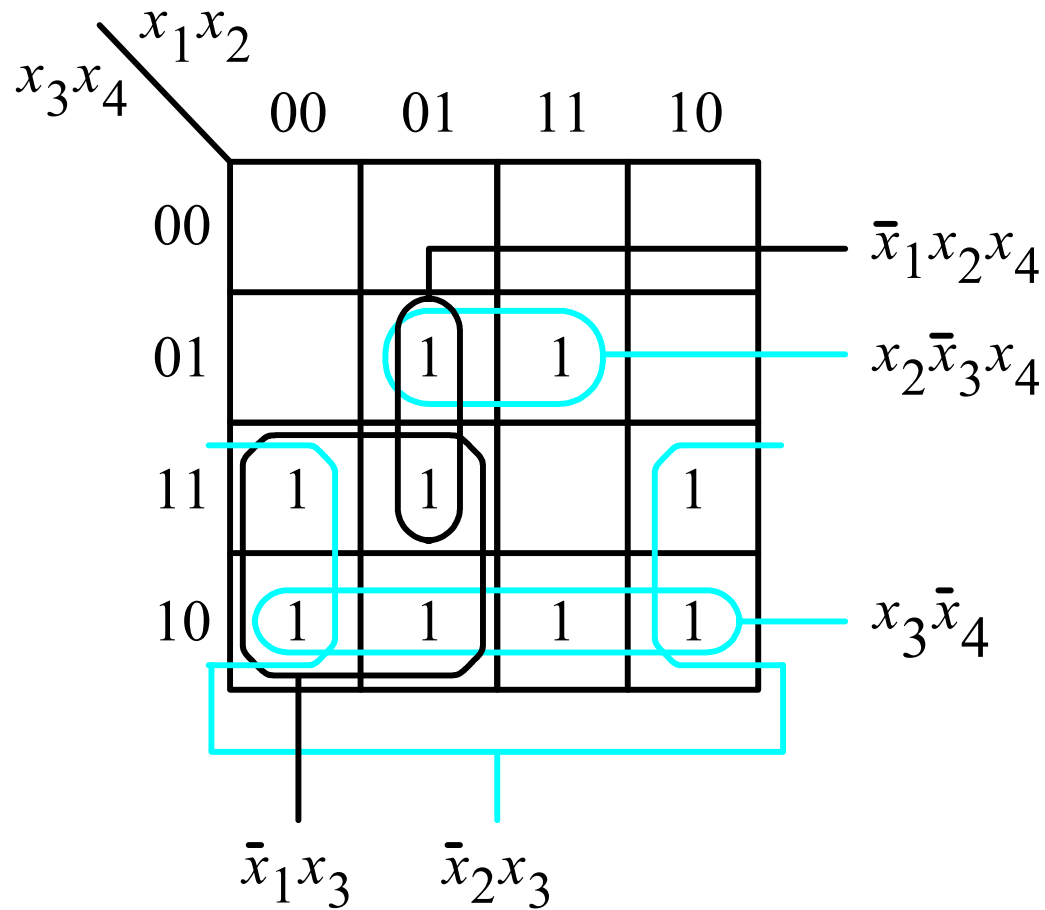


Figure 2.57. Four-variable function $f(x_1, \dots, x_4) = \Sigma m(2, 3, 5, 6, 7, 10, 11, 13, 14)$.

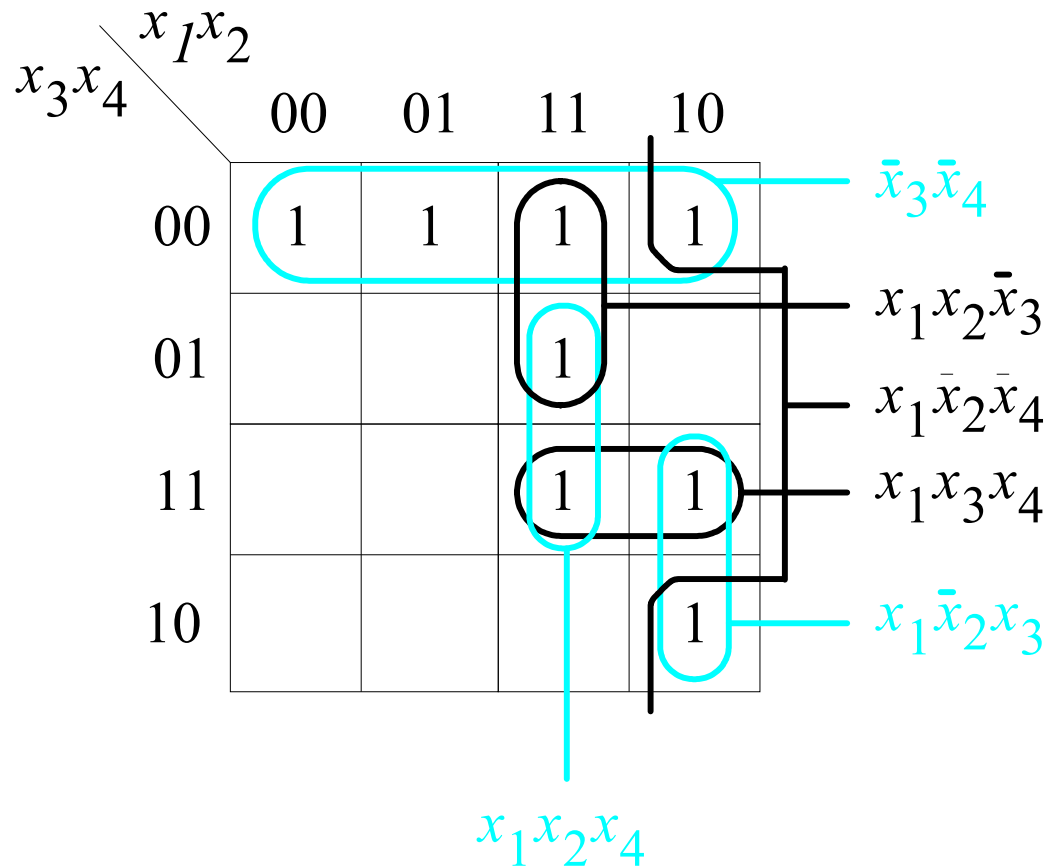


Figure 2.58. The function $f(x_1, \dots, x_4) = \Sigma m(0, 4, 8, 10, 11, 12, 13, 15)$.

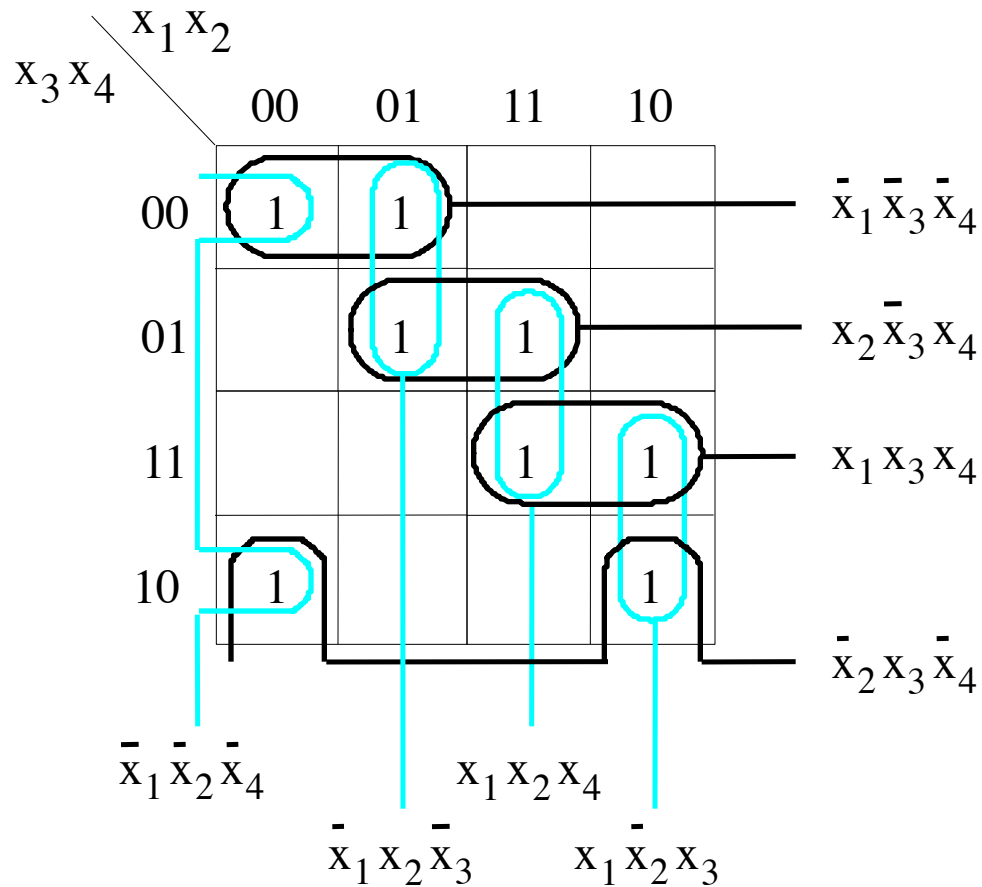


Figure 2.59. The function $f(x_1, \dots, x_4) = \Sigma m(0, 2, 4, 5, 10, 11, 13, 15)$.

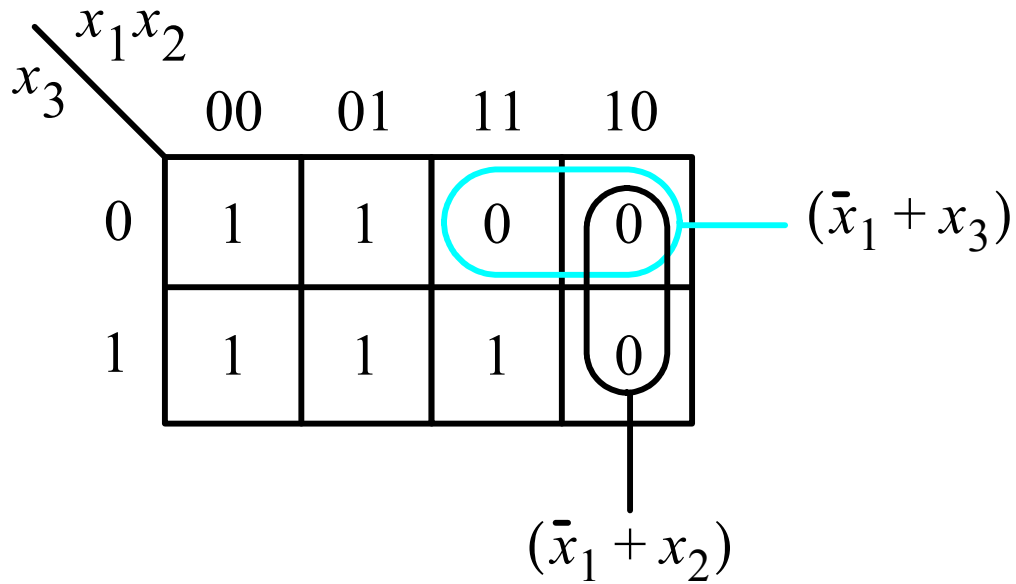


Figure 2.60. POS minimization of $f(x_1, x_2, x_3) = \Pi M(4, 5, 6)$.

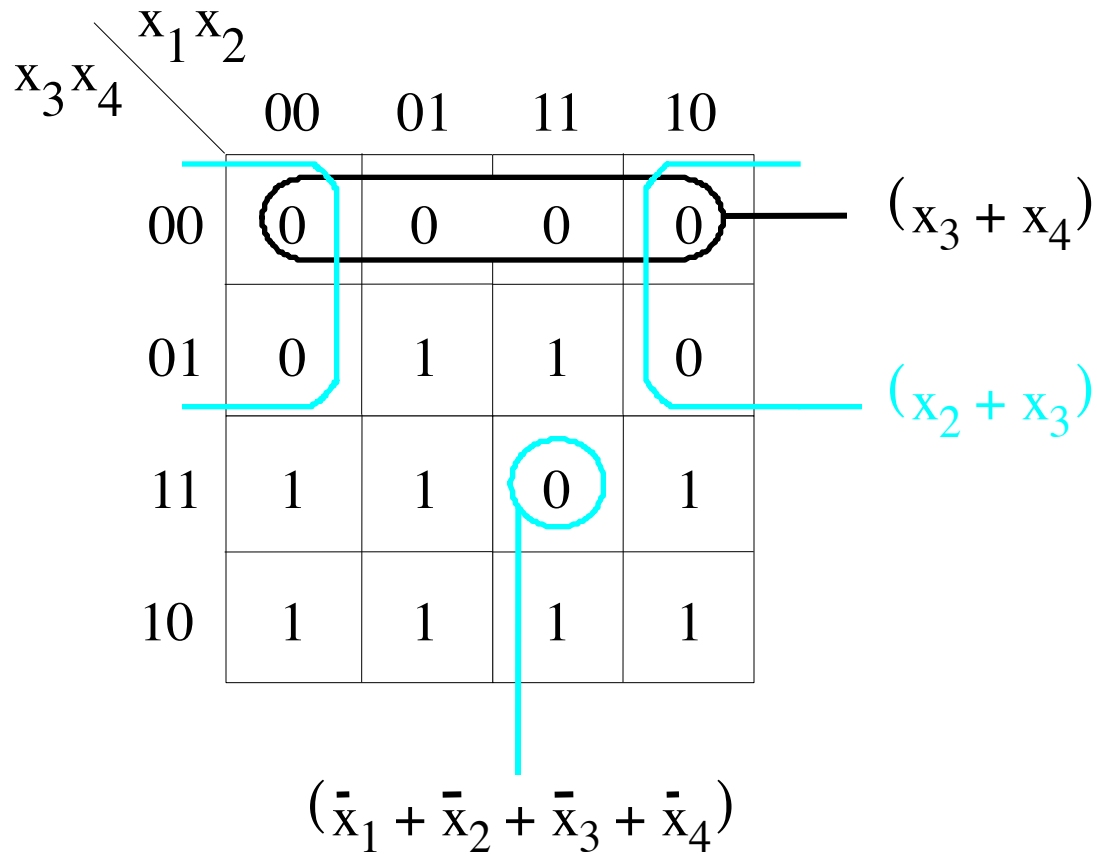


Figure 2.61. POS minimization of $f(x_1, \dots, x_4) = \prod M(0, 1, 4, 8, 9, 12, 15)$.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.62. Two implementations of the function $f(x_1, \dots, x_4) = \Sigma m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

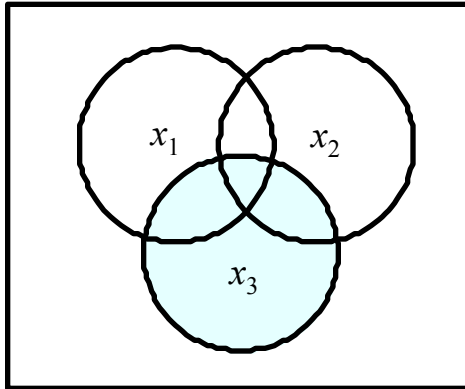
Figure 2.63. Using don't-care minterms when displaying BCD numbers.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

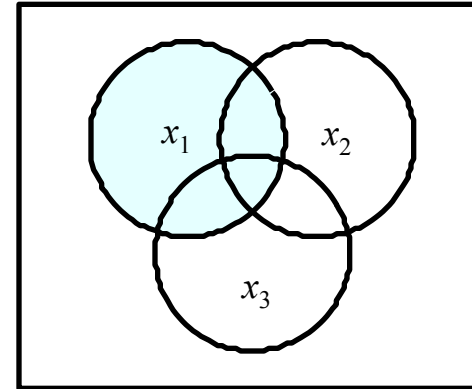
Figure 2.64. An example of multiple-output synthesis.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

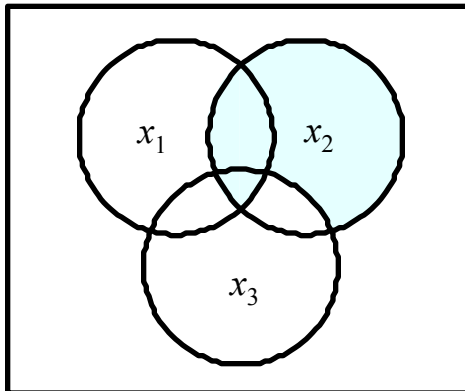
Figure 2.65. Another example of multiple-output synthesis.



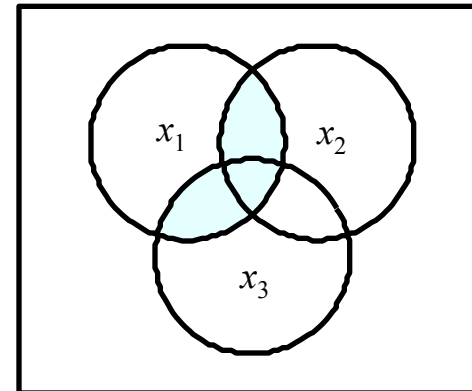
(a) Function A



(b) Function B



(c) Function C



(d) Function f

Figure 2.66. The Venn diagrams for Example 2.23.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.67. Karnaugh maps for Example 2.26.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure 2.68. Karnaugh maps for Example 2.27.

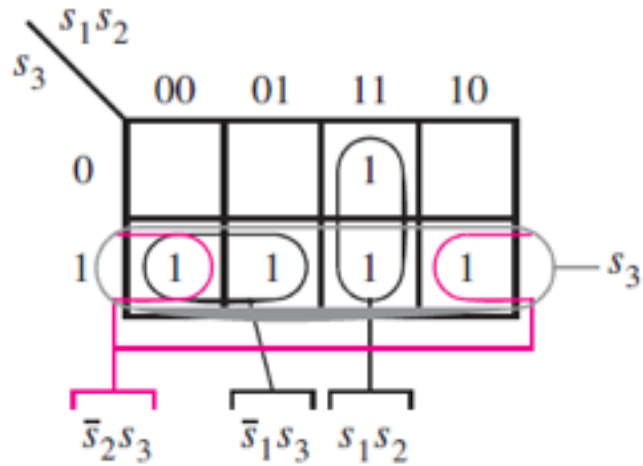


Figure 2.69. A K-map that represents the function in Example 2.28.

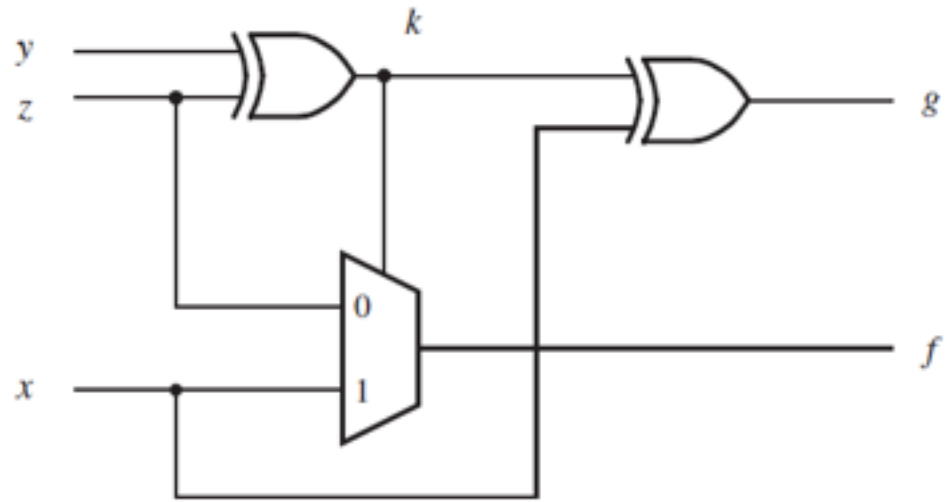


Figure 2.70. The logic circuit for Example 2.29.

```
module f_g (x, y, z, f, g);  
  input x, y, z;  
  output f, g;  
  wire k;  
  
  assign k = y ^ z;  
  assign g = k ^ x;  
  assign f = (~k & z) | (k & x);  
  
endmodule
```

Figure 2.70. Verilog code for Example 2.29.

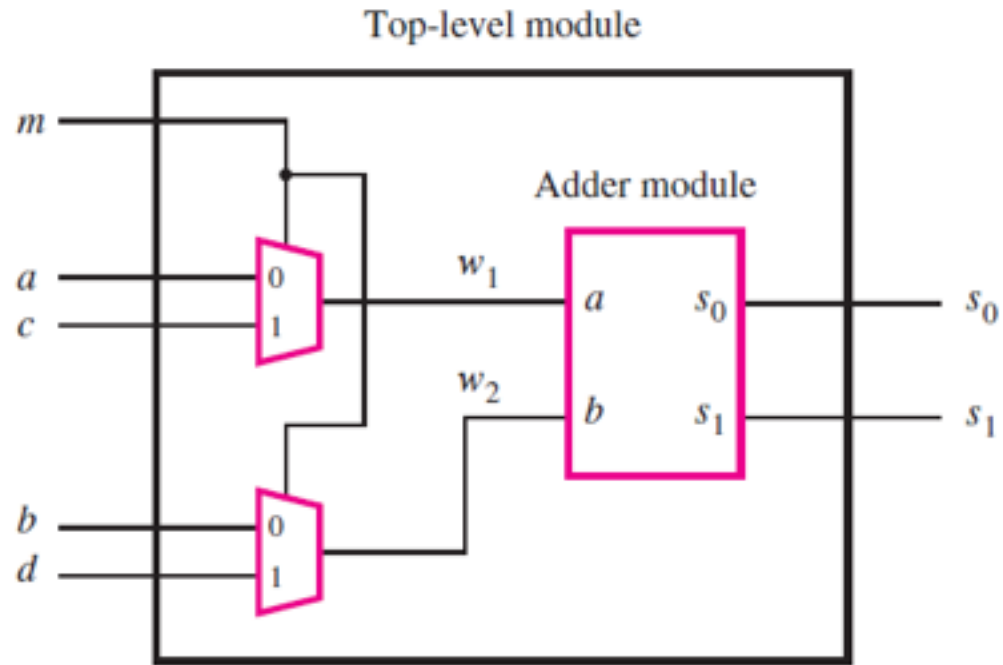


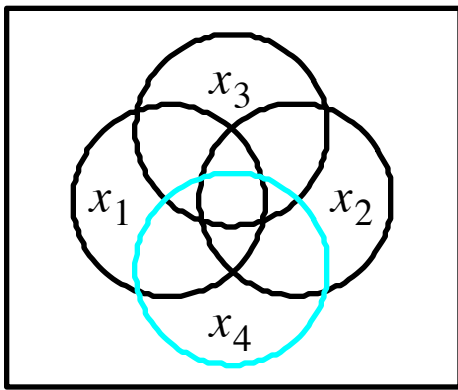
Figure 2.72. The circuit for Example 2.30.

```
module shared (a, b, c, d, m, s1, s0);  
  input a, b, c, d, m;  
  output s1, s0;  
  wire w1, w2;  
  mux2to1 U1 (a, c, m, w1);  
  mux2to1 U2 (b, d, m, w2);  
  adder U3 (w1, w2, s1, s0);  
endmodule
```

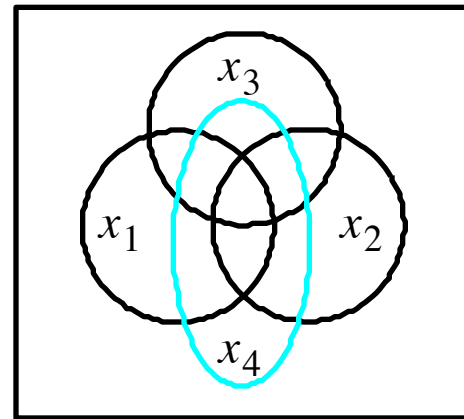
```
module mux2to1 (x1, x2, s, f);  
  input x1, x2, s;  
  output f;  
  assign f = (~s & x1) | (s & x2);  
endmodule
```

```
module adder (a, b, s1, s0);  
  input a, b;  
  output s1, s0;  
  assign s1 = a & b;  
  assign s0 = a ^ b;  
endmodule
```

Figure 2.73. Verilog code for Example 2.30.



(a)



(b)

Figure P2.1. Two attempts to draw a four-variable Venn diagram.

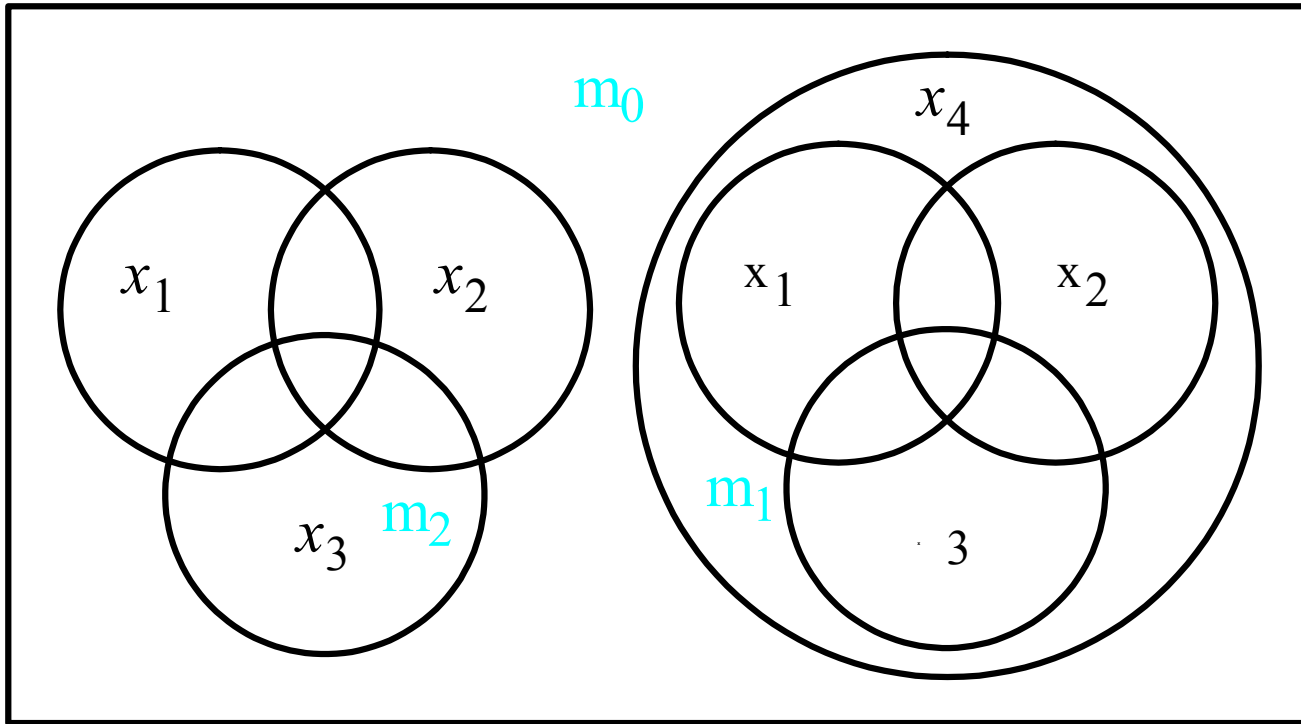


Figure P2.2. A four-variable Venn diagram.

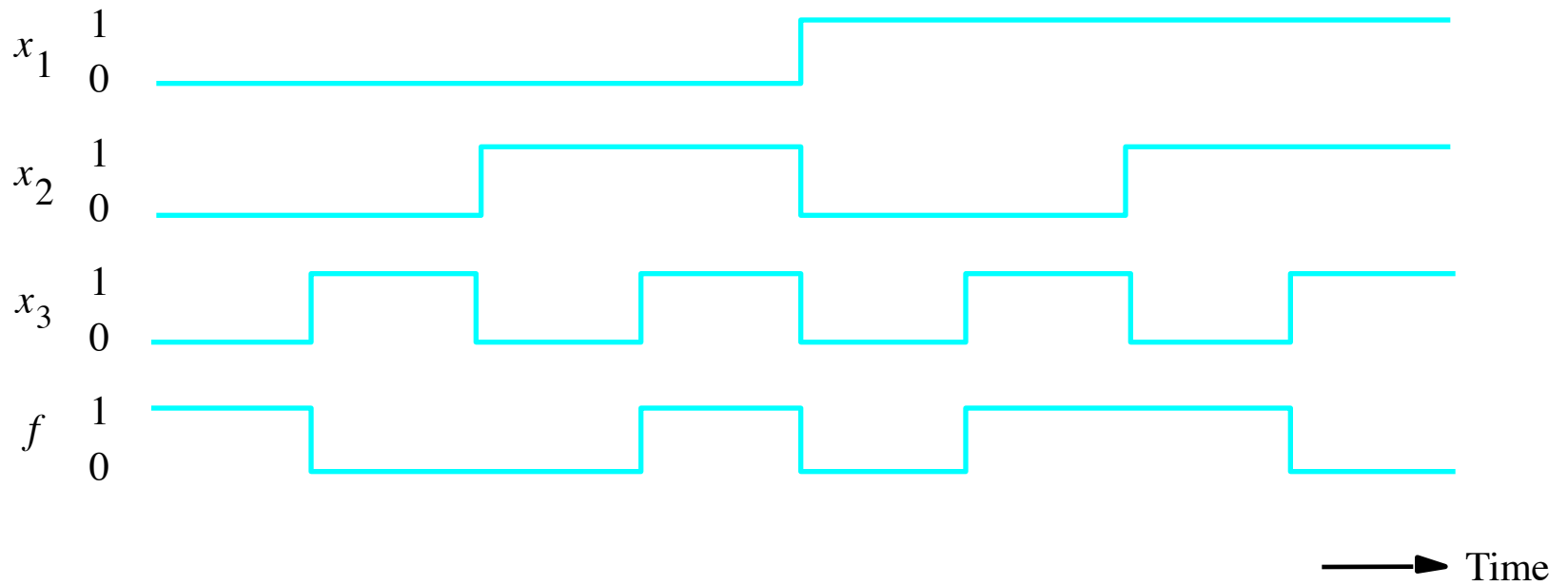


Figure P2.3. A timing diagram representing a logic function.

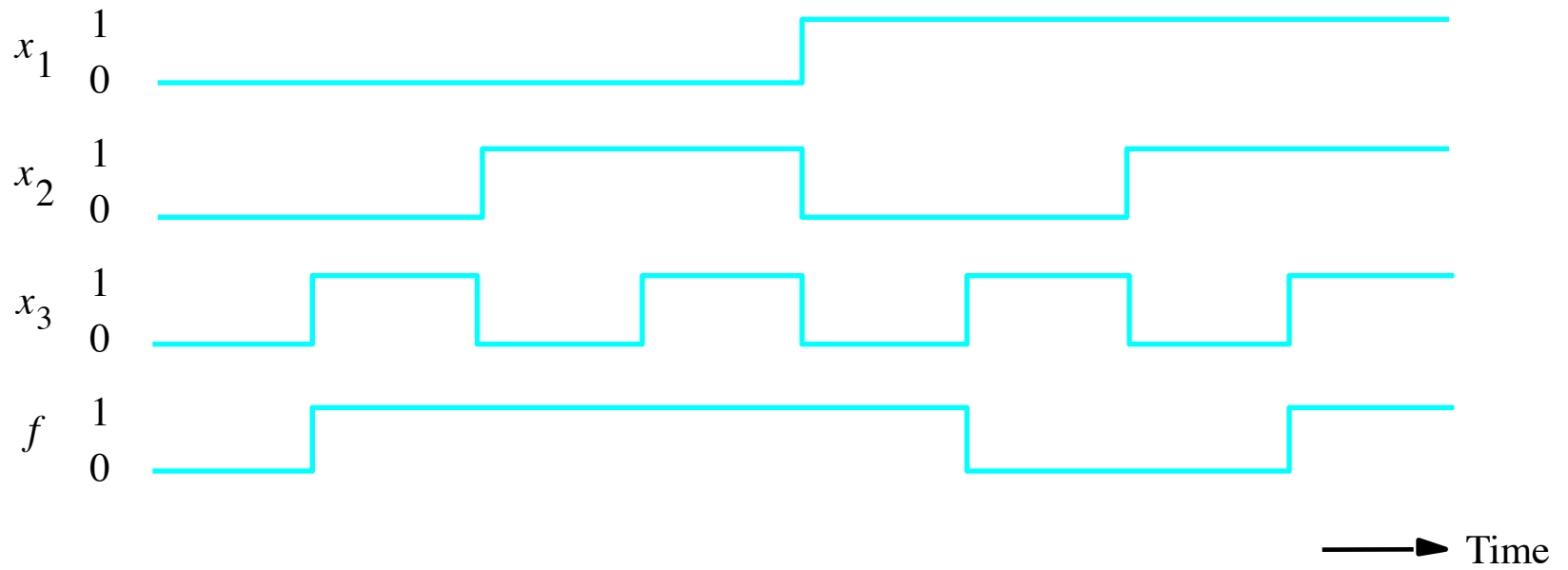


Figure P2.4. A timing diagram representing a logic function.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure P2.5. Circuit for problem 2.78.

Please see “**portrait orientation**” PowerPoint file for Chapter 2

Figure P2.6. Circuit for problem 2.79.