Chapter 4

4.1. SOP form: $f = \overline{x}_1 x_2 + \overline{x}_2 x_3$
POS form: $f = (\overline{x}_1 + \overline{x}_2)(x_2 + x_3)$

4.2. SOP form: $f = x_1\overline{x}_2 + x_1 x_3 + \overline{x}_2 x_3$
POS form: $f = (x_1 + x_3)(x_1 + \overline{x}_2)(\overline{x}_2 + x_3)$

4.3. SOP form: $f = \overline{x}_1 x_2 x_3 \overline{x}_4 + x_1 x_2 \overline{x}_3 x_4 + \overline{x}_2 x_3 x_4$
POS form: $f = (\overline{x}_1 + x_4)(x_2 + x_3)(\overline{x}_2 + \overline{x}_3 + \overline{x}_4)(x_2 + x_4)(x_1 + x_3)$

4.4. SOP form: $f = \overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_4 + x_2 x_3 x_4$
POS form: $f = (\overline{x}_2 + x_3)(x_2 + \overline{x}_3 + \overline{x}_4)(\overline{x}_2 + x_4)$

4.5. SOP form: $f = \overline{x}_3\overline{x}_5 + \overline{x}_3 x_4 + x_2 x_4 \overline{x}_5 + \overline{x}_1 x_3 \overline{x}_4 x_5 + x_1 x_2 \overline{x}_4 x_5$
POS form: $f = (\overline{x}_3 + x_4 + x_5)(\overline{x}_3 + \overline{x}_4 + \overline{x}_5)(x_2 + \overline{x}_3 + \overline{x}_4)(x_1 + x_3 + x_4 + \overline{x}_5)(\overline{x}_1 + x_2 + x_4 + \overline{x}_5)$

4.6. SOP form: $f = \overline{x}_2 x_3 + \overline{x}_1 x_5 + \overline{x}_1 x_3 + \overline{x}_3\overline{x}_4 + \overline{x}_2 x_5$
POS form: $f = (\overline{x}_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_4)(x_3 + \overline{x}_4 + x_5)$

4.7. SOP form: $f = x_3\overline{x}_4\overline{x}_5 + \overline{x}_3\overline{x}_4 x_5 + x_1 x_4 x_5 + x_1 x_2 x_4 + x_3 x_4 x_5 + \overline{x}_2 x_3 x_4 + x_2 \overline{x}_3 x_4 \overline{x}_5$
POS form: $f = (x_3 + x_4 + x_5)(\overline{x}_3 + x_4 + \overline{x}_5)(x_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4 + x_5)$

4.8. $f = \sum m(0, 7)$
$f = \sum m(1, 6)$
$f = \sum m(2, 5)$
$f = \sum m(0, 1, 6)$
$f = \sum m(0, 2, 5)$
etc.

4.9. $f = x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4$

4.10. SOP form: $f = x_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 x_4 + x_1 x_3 \overline{x}_4 + \overline{x}_1 x_2 x_3 + \overline{x}_1 x_3 x_4 + x_2 \overline{x}_3 x_4$
POS form: $f = (x_1 + x_2 + x_3)(x_1 + x_2 + x_4)(x_1 + x_3 + x_4)(x_2 + x_3 + x_4)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4)$
The POS form has lower cost.

4.11. The statement is false. As a counter example consider $f(x_1, x_2, x_3) = \sum m(0, 5, 7)$.
Then, the minimum-cost SOP form $f = x_1 x_3 + \overline{x}_1\overline{x}_2\overline{x}_3$ is unique.
But, there are two minimum-cost POS forms:
$f = (x_1 + \overline{x}_3)(\overline{x}_1 + x_3)(x_1 + \overline{x}_2)$ and
$f = (x_1 + \overline{x}_3)(\overline{x}_1 + x_3)(\overline{x}_2 + x_3)$

4.12. If each circuit is implemented separately:

$f = \overline{x}_1\overline{x}_4 + \overline{x}_1 x_2 x_3 + x_1\overline{x}_2 x_4$ Cost$=15$

$g = \overline{x}_1\overline{x}_3\overline{x}_4 + \overline{x}_2 x_3\overline{x}_4 + x_1\overline{x}_3 x_4 + x_1 x_2 x_4$ Cost $=21$

In a combined circuit:

$f = \overline{x}_2 x_3\overline{x}_4 + \overline{x}_1\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3 x_4 + \overline{x}_1 x_2 x_3$

$g = \overline{x}_2 x_3\overline{x}_4 + \overline{x}_1\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3 x_4 + x_1 x_2 x_4$

The first 3 product terms are shared, hence the total cost is 31.

4.13. If each circuit is implemented separately:

$f = \overline{x}_1 x_2 x_4 + x_2 x_4 x_5 + x_3\overline{x}_4\overline{x}_5 + \overline{x}_1\overline{x}_2\overline{x}_4 x_5$ Cost $=22$

$g = \overline{x}_3\overline{x}_5 + \overline{x}_4\overline{x}_5 + \overline{x}_1\overline{x}_2\overline{x}_4 + \overline{x}_1 x_2 x_4 + x_2 x_4 x_5$ Cost $=24$

In a combined circuit:

$f = \overline{x}_1 x_2 x_4 + x_2 x_4 x_5 + x_3\overline{x}_4\overline{x}_5 + \overline{x}_1\overline{x}_2\overline{x}_4 x_5$

$g = \overline{x}_1 x_2 x_4 + x_2 x_4 x_5 + x_3\overline{x}_4\overline{x}_5 + \overline{x}_1\overline{x}_2\overline{x}_4 x_5 + \overline{x}_3\overline{x}_5$

The first 4 product terms are shared, hence the total cost is 31. Note that in this implementation $f \subseteq g$, thus g can be realized as $g = f + \overline{x}_3\overline{x}_5$, in which case the total cost is lowered to 28.

4.14. $f = (x_3 \uparrow g) \uparrow ((g \uparrow g) \uparrow x_4)$ where $g = (x_1 \uparrow (x_2 \uparrow x_2)) \uparrow ((x_1 \uparrow x_1) \uparrow x_2)$

4.15. $\overline{f} = (((x_3 \downarrow x_3) \downarrow g) \downarrow ((g \downarrow g) \downarrow (x_4 \downarrow x_4))$, where $g = ((x_1 \downarrow x_1) \downarrow x_2) \downarrow (x_1 \downarrow (x_2 \downarrow x_2))$. Then, $f = \overline{f} \downarrow \overline{f}$.

4.16. $f = (g \uparrow k) \uparrow ((g \uparrow g) \uparrow (k \uparrow k))$, where $g = (x_1 \uparrow x_1) \uparrow (x_2 \uparrow x_2) \uparrow (x_5 \uparrow x_5)$ and $k = (x_3 \uparrow (x_4 \uparrow x_4)) \uparrow ((x_3 \uparrow x_3) \uparrow x_4)$

4.17. $\overline{f} = (g \downarrow k) \downarrow ((g \downarrow g) \downarrow (k \downarrow k))$, where $g = x_1 \downarrow x_2 \downarrow x_5$ and $k = ((x_3 \downarrow x_3) \downarrow x_4) \downarrow (x_3 \downarrow (x_4 \downarrow x_4))$. Then, $f = \overline{f} \downarrow \overline{f}$.

4.18. $f = \overline{x}_1(x_2 + x_3)(x_4 + x_5) + x_1(\overline{x}_2 + x_3)(\overline{x}_4 + x_5)$

4.19. $f = x_1\overline{x}_3\overline{x}_4 + x_2\overline{x}_3\overline{x}_4 + x_1 x_3 x_4 + x_2 x_3 x_4 = (x_1 + x_2)\overline{x}_3\overline{x}_4 + (x_1 + x_2)x_3 x_4$

This requires 2 OR and 2 AND gates.

4.20. $f = x_1 \cdot g + \overline{x}_1 \cdot \overline{g}$, where $g = \overline{x}_3 x_4 + x_3\overline{x}_4$

4.21 $f = g \cdot h + \overline{g} \cdot \overline{h}$, where $g = x_1 x_2$ and $h = x_3 + x_4$

4.22. Let $D(0, 20)$ be 0 and $D(15, 26)$ be 1. Then decomposition yields:

$g = x_5(\overline{x}_1 + x_2)$

$f = (\overline{x}_3\overline{x}_4 + x_3 x_4)g + \overline{x}_3 x_4\overline{g} = x_3 x_4 g + \overline{x}_3\overline{x}_4 g + \overline{x}_3 x_4\overline{g}$

Cost $= 9 + 18 = 27$

The optimal SOP form is:
$f = \overline{x}_3 x_4 \overline{x}_5 + \overline{x}_1 x_3 x_4 x_5 + x_1 \overline{x}_2 \overline{x}_3 x_4 + \overline{x}_1 \overline{x}_3 \overline{x}_4 x_5 + x_2 \overline{x}_3 \overline{x}_4 x_5 + x_2 x_3 x_4 x_5$
Cost $= 7 + 29 = 36$

4.23. The prime implicants are generated as follows:

List 1

0	0 0 0 0	√
2	0 0 1 0	√
4	0 1 0 0	√
8	1 0 0 0	√
5	0 1 0 1	√
9	1 0 0 1	√
7	0 1 1 1	√
15	1 1 1 1	√

List 2

0,2	0 0 x 0
0,4	0 x 0 0
0,8	x 0 0 0
4,5	0 1 0 x
8,9	1 0 0 x
5,7	0 1 x 1
7,15	x 1 1 1

The initial prime implicant table is

Prime implicant	Minterm 0	2	4	5	7	8	9	15
p_1 = 0 0 x 0	√	√						
p_2 = 0 x 0 0	√		√					
p_3 = x 0 0 0	√					√		
p_4 = 0 1 0 x			√	√				
p_5 = 1 0 0 x						√	√	
p_6 = 0 1 x 1				√	√			
p_7 = x 1 1 1					√			√

The prime implicants p_1, p_5 and p_7 are essential. Removing these prime implicants gives

Prime implicant	Minterm 4	5
p_2	√	
p_3		
p_4	√	√
p_6		√

Since p_4 covers both minterms, the final cover is

$$\begin{aligned} C &= \{p_1, p_4, p_5, p_7\} \\ &= \{00\text{x}0, 010\text{x}, 100\text{x}, \text{x}111\} \end{aligned}$$

and the function is implemented as

$$f = \overline{x}_1\overline{x}_2\overline{x}_4 + \overline{x}_1 x_2 \overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + x_2 x_3 x_4$$

4.24. The prime implicants are generated as follows:

List 1

0	0 0 0 0	✓
4	0 1 0 0	✓
8	1 0 0 0	✓
3	0 0 1 1	✓
6	0 1 1 0	✓
9	1 0 0 1	✓
7	0 1 1 1	✓
11	1 0 1 1	✓
13	1 1 0 1	✓
15	1 1 1 1	✓

List 2

0,4	0 x 0 0	
0,8	x 0 0 0	
4,6	0 1 x 0	
8,9	1 0 0 x	
3,7	0 x 1 1	✓
3,11	x 0 1 1	✓
6,7	0 1 1 x	
9,11	1 0 x 1	✓
9,13	1 x 0 1	✓
7,15	x 1 1 1	✓
11,15	1 x 1 1	✓
13,15	1 1 x 1	✓

List 3

3,7,11,15	x x 1 1
9,11,13,15	1 x x 1

The initial prime implicant table is

Prime implicant	Minterm 0	4	6	8	9	15
p_1 = 0 x 0 0	✓	✓				
p_2 = x 0 0 0	✓			✓		
p_3 = 0 1 x 0		✓	✓			
p_4 = 1 0 0 x				✓	✓	
p_5 = 0 1 1 x			✓			
p_6 = x x 1 1						✓
p_7 = 1 x x 1					✓	✓

There are no essential prime implicants. Prime implicant p_3 dominates p_5 and their costs are the same, so remove p_5. Similarly, p_7 dominates p_6, so remove p_6. This gives

Prime implicant	Minterm 0	4	6	8	9	15
p_1	✓	✓				
p_2	✓			✓		
p_3		✓	✓			
p_4				✓	✓	
p_7					✓	✓

Now, p_3 and p_7 are essential, which leaves

Prime implicant	Minterm 0	8
p_1	√	
p_2	√	√
p_4		√

Choosing p_2 results in the minimum cost cover

$$\begin{aligned} C &= \{p_2, p_3, p_7\} \\ &= \{\text{x000}, \text{01x0}, \text{1xx1}\} \end{aligned}$$

and the function is implemented as

$$f = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1 x_2 \overline{x}_4 + x_1 x_4$$

4.25. The prime implicants are generated as follows:

List 1

0	0 0 0 0	√
4	0 1 0 0	√
8	1 0 0 0	√
3	0 0 1 1	√
5	0 1 0 1	√
9	1 0 0 1	√
12	1 1 0 0	√
7	0 1 1 1	√
11	1 0 1 1	√
13	1 1 0 1	√
14	1 1 1 0	√

List 2

0,4	0 x 0 0	√
0,8	x 0 0 0	
4,5	0 1 0 x	√
4,12	x 1 0 0	√
8,9	1 0 0 x	√
8,12	1 x 0 0	√
3,7	0 x 1 1	
3,11	x 0 1 1	
5,7	0 1 x 1	
5,13	x 1 0 1	√
9,11	1 0 x 1	
9,13	1 x 0 1	√
12,13	1 1 0 x	√
12,14	1 1 x 0	

List 3

0,4,8,12	x x 0 0
4,5,12,13	x 1 0 x
8,9,12,13	1 x 0 x

The initial prime implicant table is

Prime implicant	Minterm 0	3	4	5	7	9	11
$p_1 = 0\ x\ 1\ 1$		√			√		
$p_2 = x\ 0\ 1\ 1$		√					√
$p_3 = 0\ 1\ x\ 1$				√	√		
$p_4 = 1\ 0\ x\ 1$						√	√
$p_5 = x\ x\ 0\ 0$	√		√				
$p_6 = x\ 1\ 0\ x$			√	√			
$p_7 = 1\ x\ 0\ x$						√	
$p_8 = 1\ 1\ x\ 0$							

Prime implicant p_5 is essential, so remove columns 0 and 4 to get

Prime implicant	Minterm 3	5	7	9	11
p_1	√		√		
p_2	√				√
p_3		√	√		
p_4				√	√
p_6		√			
p_7				√	

Here, p_3 dominates p_6, and p_4 dominates p_7; but costs of p_3 and p_4 are greater than the costs of p_6 and p_7, respectively. So, use branching. First choose p_3 to be in the final cover, which leads to

Prime implicant	Minterm 3	9	11
p_1	√		
p_2	√		√
p_4		√	√
p_6			
p_7		√	

Now, choose p_2 and p_7 (lower cost than p_4) to cover the remaining minterms. The resulting cover is

$$\begin{aligned} C &= \{p_2, p_3, p_5, p_7\} \\ &= \{\text{x011}, \text{01x1}, \text{xx00}, \text{1x0x}\} \end{aligned}$$

Next, assume that p_3 is not included in the final cover, which leads to

Prime implicant	Minterm 3	5	7	9	11
p_1	✓		✓		
p_2	✓				✓
p_4				✓	✓
p_6		✓			
p_7				✓	

Then p_6 is essential. Also, column 3 dominates 7, hence remove 3 giving

Prime implicant	Minterm 7	9	11
p_1	✓		
p_2			✓
p_4		✓	✓
p_7		✓	

Choose p_1 and p_4, which results in the cover

$$\begin{aligned} C &= \{p_1, p_4, p_5, p_6\} \\ &= \{0\text{x}11, 10\text{x}1, \text{xx}00, \text{x}10\text{x}\} \end{aligned}$$

Both covers have the same cost, so choosing the first cover the function can be implemented as

$$f = \overline{x}_2 x_3 x_4 + \overline{x}_1 x_2 x_4 + \overline{x}_3 \overline{x}_4 + x_1 \overline{x}_3$$

Observe that if we had not taken the cost of prime implicants (rows) into account and pursued the dominance of p_3 over p_6 and p_4 over p_7, then we would have removed p_6 and p_7 giving the following table

Prime implicant	Minterm 3	5	7	9	11
p_1	✓		✓		
p_2	✓				✓
p_3		✓	✓		
p_4				✓	✓

Now p_3 and p_4 are essential. Also choose p_1, so that

$$\begin{aligned} C &= \{p_1, p_3, p_4, p_5\} \\ &= \{0\text{x}11, 01\text{x}1, 10\text{x}1, \text{xx}00\} \end{aligned}$$

The cost of this cover is greater by one literal compared to both covers derived above.

4.26. Note that $X \# Y = X \cdot \overline{Y}$. Therefore,

$$
\begin{aligned}
(A \cdot B)\#C &= A \cdot B \cdot \overline{c} \\
(A\#C) \cdot (B\#C) &= A \cdot \overline{C} \cdot B \cdot \overline{C} \\
&= A \cdot B \cdot \overline{C}
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
(A + B)\#C &= (A + B) \cdot \overline{C} \\
&= A \cdot \overline{C} + B \cdot \overline{C} \\
(A\#C) + (B\#C) &= A \cdot \overline{C} + B \cdot \overline{C}
\end{aligned}
$$

4.27. The initial cover is $C^0 = \{0000, 0011, 0100, 0101, 0111, 1000, 1001, 1111\}$.
Using the $*$-product get the prime implicants
P = {00x0, 0x00, x000, 010x, 01x1, 100x, x111}.
The minimum cover is $C_{minimum}$ = {00x0, 010x, 100x, x111}, which corresponds to $f = \overline{x}_1\overline{x}_2\overline{x}_4 + \overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 \overline{x}_3 + x_2 x_3 x_4$.

4.28. The initial cover is C^0 = {0x0x0, 110xx, x1101, 1001x, 11110, 01x10, 0x011}.
Using the $*$-product get the prime implicants
P = {0x0x0, xx01x, x1x10, 110xx, x10x0, 11x01, x1101}.
The minimum cover is $C_{mimimum}$ = {0x0x0, xx01x, x1x10, 110xx, x1101}, which corresponds to $f = \overline{x}_1\overline{x}_3\overline{x}_5 + \overline{x}_3 x_4 + x_2 x_4 \overline{x}_5 + x_1 x_2 \overline{x}_3 + x_2 x_3 \overline{x}_4 x_5$.

4.29. The initial cover is C^0 = {00x0, 100x, x010, 1111, 00x1, 011x}.
Using the $*$-product get the prime implicants P = {00xx, 0x1x, x00x, x0x0, x111}.
The minimum-cost cover is $C_{minimum}$ = {x00x, x0x0, x111}, which corresponds to $f = \overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_4 + x_2 x_3 x_4$.

4.30. Expansion of $\overline{x}_1\overline{x}_2\overline{x}_3$ gives $\overline{x}_1$.
Expansion of $\overline{x}_1\overline{x}_2 x_3$ gives $\overline{x}_1$.
Expansion of $\overline{x}_1 x_2 \overline{x}_3$ gives $\overline{x}_1$.
Expansion of $x_1 x_2 x_3$ gives $x_2 x_3$.
The set of prime implicants comprises $\overline{x}_1$ and $x_2 x_3$.

4.31. Expansion of $\overline{x}_1 x_2 \overline{x}_3 x_4$ gives $x_2 \overline{x}_3 x_4$ and $\overline{x}_1 x_2 x_4$.
Expansion of $x_1 x_2 \overline{x}_3 x_4$ gives $x_2 \overline{x}_3 x_4$.
Expansion of $x_1 x_2 x_3 \overline{x}_4$ gives $x_3 \overline{x}_4$.
Expansion of $\overline{x}_1 x_2 x_3$ gives $\overline{x}_1 x_3$.
Expansion of $\overline{x}_2 x_3$ gives $\overline{x}_2 x_3$.
The set of prime implicants comprises $x_2\overline{x}_3 x_4$, $\overline{x}_1 x_2 x_4$, $x_3 \overline{x}_4$, $\overline{x}_1 x_3$, and $\overline{x}_2 x_3$.

4.32. Representing both functions in the form of Karnaugh map, it is easy to show that $f = g$. The minimum cost SOP expression is
$f = g = \overline{x}_2\overline{x}_3\overline{x}_5 + \overline{x}_2 x_3 \overline{x}_4 + x_1 x_3 x_4 + x_1 x_2 x_4 x_5$.

4.33. The cost of the circuit in Figure P4.2 is 11 gates and 30 inputs, for a total of 41. The functions implemented by the circuit can also be realized as

$$
\begin{aligned}
f &= \overline{x}_1\overline{x}_2\overline{x}_4 + x_2\overline{x}_3\overline{x}_4 + \overline{x}_1 x_3 x_4 + x_1 x_4 \\
g &= \overline{x}_1\overline{x}_2\overline{x}_4 + x_2\overline{x}_3\overline{x}_4 + \overline{x}_1 x_3 x_4 + \overline{x}_2 x_4 + x_3\overline{x}_4
\end{aligned}
$$

The first three product terms in f and g are the same; therefore, they can be shared. Then, the cost of implementing f and g is 8 gates and 24 inputs, for a total of 32.

4.34. The cost of the circuit in Figure P4.3 is 11 gates and 26 inputs, for a total of 37. The functions implemented by the circuit can also be realized as

$$
\begin{aligned}
f &= (\overline{x}_2 \uparrow x_4) \uparrow (\overline{x}_1 \uparrow x_2 \uparrow x_3) \uparrow (x_1 \uparrow \overline{x}_2 \uparrow x_3) \uparrow (\overline{x}_2 \uparrow \overline{x}_3) \\
g &= (\overline{x}_2 \uparrow x_4) \uparrow (\overline{x}_1 \uparrow x_2 \uparrow x_3) \uparrow (x_1 \uparrow \overline{x}_2 \uparrow x_3) \uparrow (\overline{x}_1 \uparrow \overline{x}_1)
\end{aligned}
$$

The first three NAND terms in f and g are the same; therefore, they can be shared. Then, the cost of implementing f and g is 7 gates and 20 inputs, for a total of 27.

4.35. Using gate level primitives, the circuit in Figure 4.25*b* can be implemented using the code

```
module prob4_35 (x1, x2, x3, x4, x5, f);
  input x1, x2, x3, x4, x5;
  output f;

  or (g, x1, x2, x5);
  not (notx3, x3);
  not (notx4, x4);
  and (a, x3, notx4);
  and (b, notx3, x4);
  or (k, a, b);
  and (c, g, k);
  not (notg, g);
  not (notk, k);
  and (d, notg, notk);
  or (f, c, d);

endmodule
```

4.36. Using continuous assignment, the circuit in Figure 4.25*b* can be implemented using the code

```
module prob4_36 (x1, x2, x3, x4, x5, f);
    input x1, x2, x3, x4, x5;
    output f;
    wire g, k;

    assign g = (x1 | x2 | x5);
    assign k = (x3 & ~x4) | (~x3 & x4);
    assign f = (g & k) | (~g & ~k);

endmodule
```

4.37. Using gate level primitives, the circuit in Figure 4.27*c* can be implemented using the code

```
module prob4_37 (x1, x2, x3, x4, x5, x6, x7, f);
    input x1, x2, x3, x4, x5, x6, x7;
    output f;

    nand (a, x1, x1);
    nand (b, x2, x3);
    nand (c, a, b);
    nand (d, x5, x5);
    nand (e, x6, x6);
    nand (g, d, e);
    nand (h, x4, g);
    nand (j, x7, x7);
    nand (k, h, j);
    nand (m, c, k);
    nand (f, m, m);

endmodule
```

4.38. Using continuous assignment, the circuit in Figure 4.27*c* can be implemented using the code

```
module prob4_38 (x1, x2, x3, x4, x5, x6, x7, f);
    input x1, x2, x3, x4, x5, x6, x7;
    output f;
    wire a, b;

    assign a = ~(~x1 & ~(x2 & x3));
    assign b = ~(~(x4 & ~(~x5 & ~x6)) & ~x7);
    assign f = ~(~(a & b));

endmodule
```

4.39. Using gate level primitives, the circuit in Figure 4.28b can be implemented using the code

```
module prob4_39 (x1, x2, x3, x4, x5, x6, x7, f);
    input x1, x2, x3, x4, x5, x6, x7;
    output f;

    nor (a, x2, x2);
    nor (b, x3, x3);
    nor (c, a, b);
    nor (d, x1, c);
    nor (e, x4, x4);
    nor (g, x5, x6);
    nor (h, e, g);
    nor (k, h, x7);
    nor (f, d, k);

endmodule
```

4.40. Using continuous assignment, the circuit in Figure 4.27c can be implemented using the code

```
module prob4_40 (x1, x2, x3, x4, x5, x6, x7, f);
    input x1, x2, x3, x4, x5, x6, x7;
    output f;
    wire a, b;

    assign a = ~(x1 | ~(~x2 | ~x3));
    assign b = ~(~(~x4 | ~(x5 | x6)) | x7);
    assign f = ~(a | b);

endmodule
```

4.41. Using the POS expression

$$f = (x_1 + x_2 + \overline{x}_3 + \overline{x}_4)(x_1 + \overline{x}_2 + \overline{x}_3 + x_4)(\overline{x}_1 + x_2 + \overline{x}_3 + x_4)(\overline{x}_1 + \overline{x}_2 + x_3 + \overline{x}_4)$$

the function can be implemented using the code

```
module prob4_41 (x1, x2, x3, x4, f);
    input x1, x2, x3, x4;
    output f;

    not (notx1, x1);
    not (notx2, x2);
    not (notx3, x3);
    not (notx4, x4);
    or (a, x1, x2, notx3, notx4);
    or (b, x1, notx2, notx3, x4);
    or (c, notx1, x2, notx3, x4);
    or (d, notx1, notx2, x3, notx4);
    and (f, a, b, c, d);

endmodule
```

4.42. Using the POS expression

$$f = (x_1 + x_2 + \overline{x}_3 + \overline{x}_4)(x_1 + \overline{x}_2 + \overline{x}_3 + x_4)(\overline{x}_1 + x_2 + \overline{x}_3 + x_4)(\overline{x}_1 + \overline{x}_2 + x_3 + \overline{x}_4)$$

the function can be implemented using the code

```
module prob4_42 (x1, x2, x3, x4, f);
    input x1, x2, x3, x4;
    output f;

    assign f = (x1 | x2 | ~x3 | ~x4) & (x1 | ~x2 | ~x3 | x4) &
               (~x1 | x2 | ~x3 | x4) & (~x1 | ~x2 | x3 | ~x4);

endmodule
```

4.43. The simplest expression is

$$f = \overline{x}_1\overline{x}_3 + x_2x_3(x_1 + x_4)$$

which can be implemented using the code

```
module prob4_43 (x1, x2, x3, x4, f);
    input x1, x2, x3, x4;
    output f;

    not (notx1, x1);
    not (notx3, x3);
    and (a, notx1, notx3);
    or (b, x1, x4);
    and (c, x2, x3, b);
    or (f, a, c);

endmodule
```

4.44. The simplest expression is

$$f = \overline{x}_1\overline{x}_3 + x_2x_3(x_1 + x_4)$$

which can be implemented using the code

```
module prob4_44 (x1, x2, x3, x4, f);
   input x1, x2, x3, x4;
   output f;

   assign f = (~x1 & ~x3) | (x2 & x3 & (x1 | x4));
endmodule
```

4.45. The simplest expression is

$$f = (\overline{x}_1 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3 + x_4)$$

which can be implemented using the code

```
module prob4_45 (x1, x2, x3, x4, f);
   input x1, x2, x3, x4;
   output f;

   not (notx1, x1);
   not (notx2, x2);
   not (notx3, x3);
   or (a, notx1, x3);
   or (b, x1, notx2, notx3, x4);
   and (f, a, b);

endmodule
```

4.46. The simplest expression is

$$f = (\overline{x}_1 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3 + x_4)$$

which can be implemented using the code

```
module prob4_46 (x1, x2, x3, x4, f);
   input x1, x2, x3, x4;
   output f;

   assign f = (~x1 | x3) & (x1 | ~x2 | ~x3 | x4);

endmodule
```

4.47. The simplest expression is

$$f = (x_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_3 + x_4)$$

which can be implemented using the code

```
module prob4_47 (x1, x2, x3, x4, f);
   input x1, x2, x3, x4;
   output f;

   not (notx1, x1);
   not (notx3, x3);
   or (a, x2, notx3);
   or (b, notx1, notx3, x4);
   and (f, a, b);

endmodule
```

4.48. The simplest expression is

$$f = (x_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_3 + x_4)$$

which can be implemented using the code

```
module prob4_47 (x1, x2, x3, x4, f);
   input x1, x2, x3, x4;
   output f;

   assign f = (x2 | ~x3) & (~x1 | ~x3 | x4);
endmodule
```