Terminology

Literal

- Each appearance of a variable, either uncomplemented or complemented, in a logical term is called a *literal*.
- For example
  - Product term $x1x2x3$ has three literals
  - Sum term $(x1'+x3+x4'+x6)$ has four literals.

Implicant

A product term that indicates the input valuation(s) for which a given function is equal to 1 is called an *implicant* of the function. The most basic implicants are the minterms.

Prime Implicant

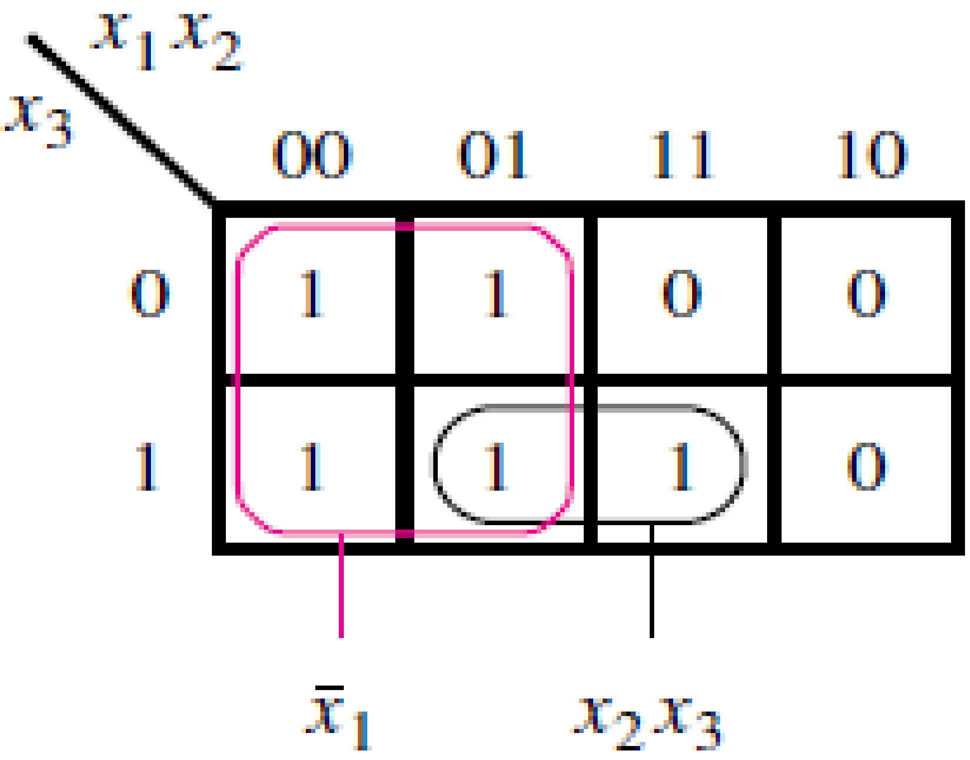
An implicant is called a *prime implicant* if it cannot be combined into another implicant that has fewer literals.

Example:

f (x1, x2, x3) = *∑m(0, 1, 2, 3, 7)*.

x3 \ x1x2	00	01	11	10
0	1	1	0	0
1	1	1	1	0

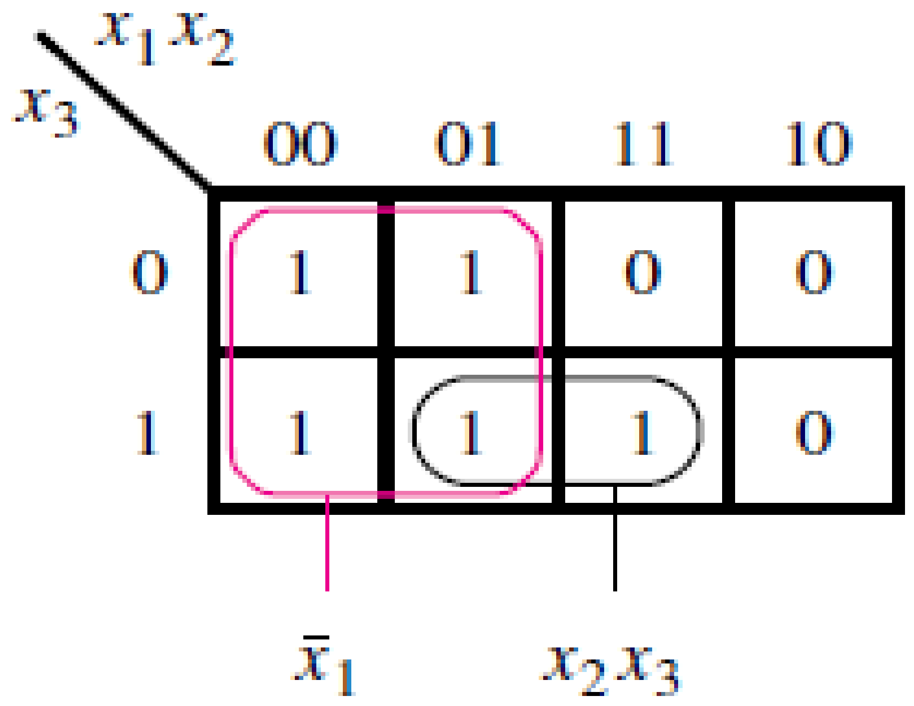
- There are 11 implicants for this function:
  - **five minterms**: x1'x2'x3', x1'x2'x3, x1'x2x3', x1'x2x3, and x1x2x3.
  - **five implicants** that correspond to all possible pairs of minterms: x1'x2' (m0 and m1), x1'x3' (m0 and m2), x1'x3 (m1 and m3), x1'x2 (m2 and m3), and x2x3 (m3 and m7).
  - **one implicant** that covers a group of four minterms: x1'.
- Here there are two prime implicants: *x*1' and *x*2*x*3.



$$f = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2x_3$$

$$f = \bar{x}_1\bar{x}_2 + \bar{x}_1x_2 + x_2x_3$$

$$f = \bar{x}_1 + x_2x_3$$



While all of these expressions represent the function *f* correctly, the cover consisting of prime implicants leads to the lowest-cost implementation.

Cost

- Cost of a logic circuit is the number of gates plus the total number of inputs to all gates in the circuit.

- If we assume that the input variables are available in both true and complemented forms, then the cost of the expression,

$$f = x_1\overline{x}_2 + x_3\overline{x}_4$$

is 9. Otherwise its cost is 13.

Minimization Procedure

The process of finding a minimum-cost circuit involves the following steps:

1. Generate all prime implicants for the given function f.
2. Find the set of essential prime implicants.
3. If the set of essential prime implicants covers all valuations for which $f = 1$,

   Then this set is the desired cover of f

   Else

   Determine the nonessential prime implicants that should be added to form a complete minimum-cost cover.

Minimization Procedure

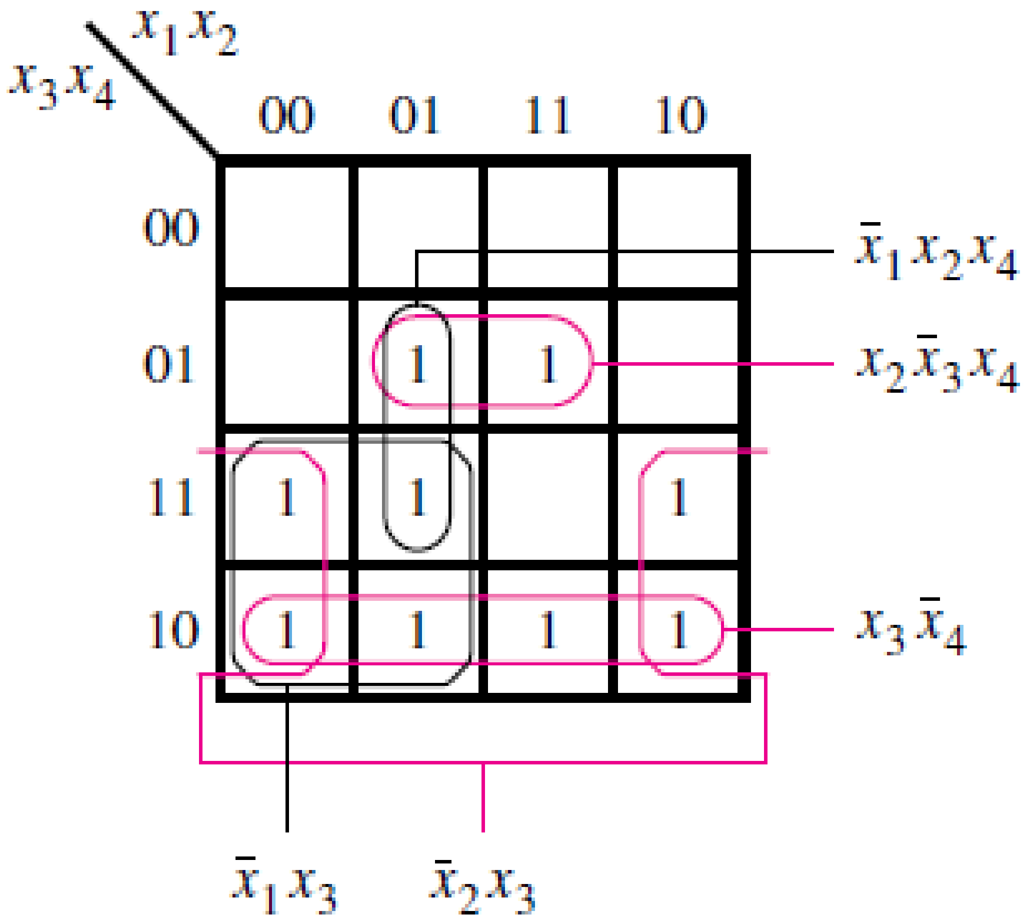
Consider the following example:

X3, x4 \ X1, x2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	0	1
10	1	1	1	1

Here there is a choice as to which prime implicants to include in the final cover

Minimization Procedure

- There are five prime implicants: x1’x3, x2’x3, x3x4’, x2x3’x4, x1’x2x4.
- The essential ones are x2’x3, (because of m11), x3x4’ (because of m14), x2x3’x4 (because of m13).
- They must be included in the cover. These three prime implicants cover all minterms for which $f = 1$ except $m7$. It is clear that $m7$ can be covered by either x1’x3 or x1’x2x4



Because $x1$’$x3$ has a lower cost, it is chosen for the cover.
Therefore, the minimum-cost realization is

$$f = \overline{x}_2x_3 + x_3\overline{x}_4 + x_2\overline{x}_3x_4 + \overline{x}_1x_3$$

Minimization Procedure

Consider the following example:

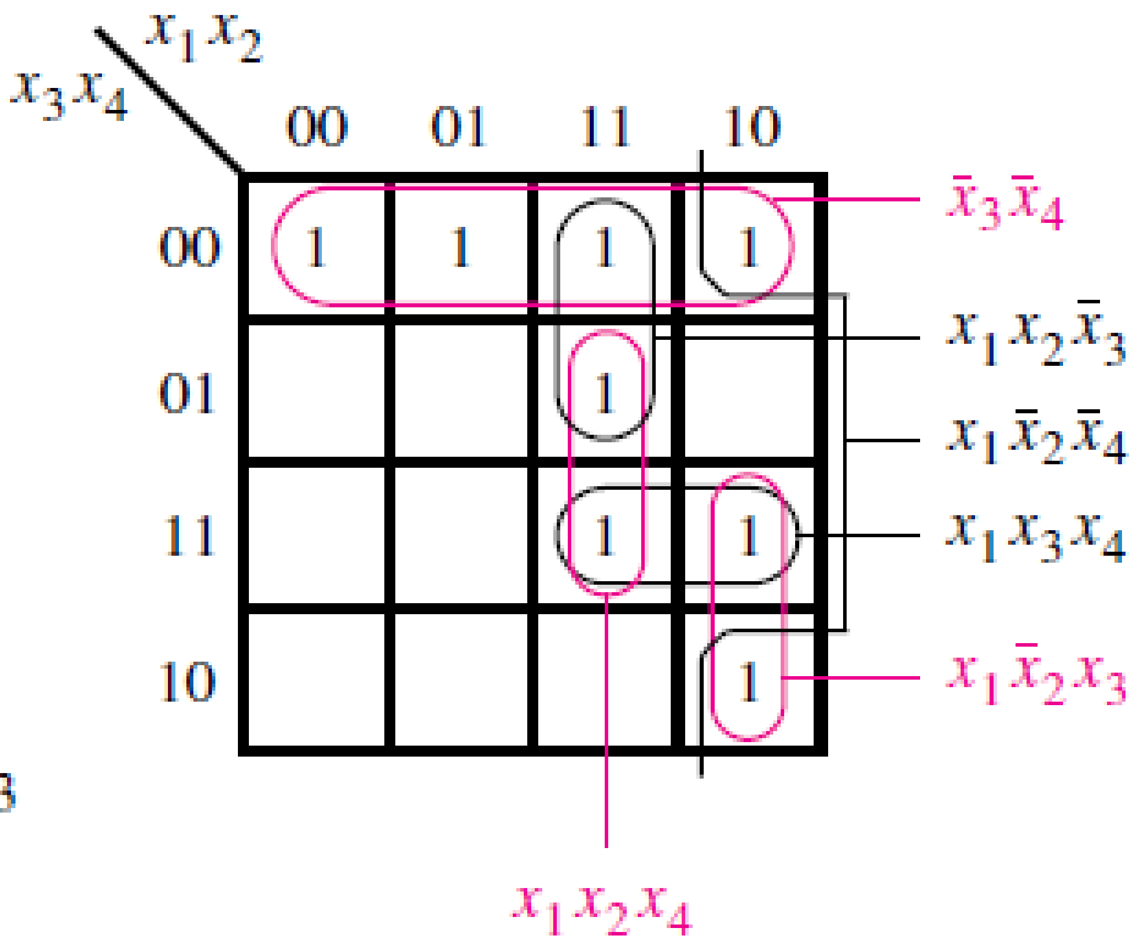
X3, x4 \ X1, x2	00	01	11	10
00	1	1	1	1
01	0	0	1	0
11	0	0	1	1
10	0	0	0	1

Consider the following example:

only $x3'x4'$ is essential.

Then the best choice to implement the minimum cost circuit is

$$f = \overline{x}_3\overline{x}_4 + x_1x_2x_4 + x_1\overline{x}_2x_3$$



Minimization Procedure

Consider the following example:

X3, x4 \ X1, x2	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	1	0	0	1

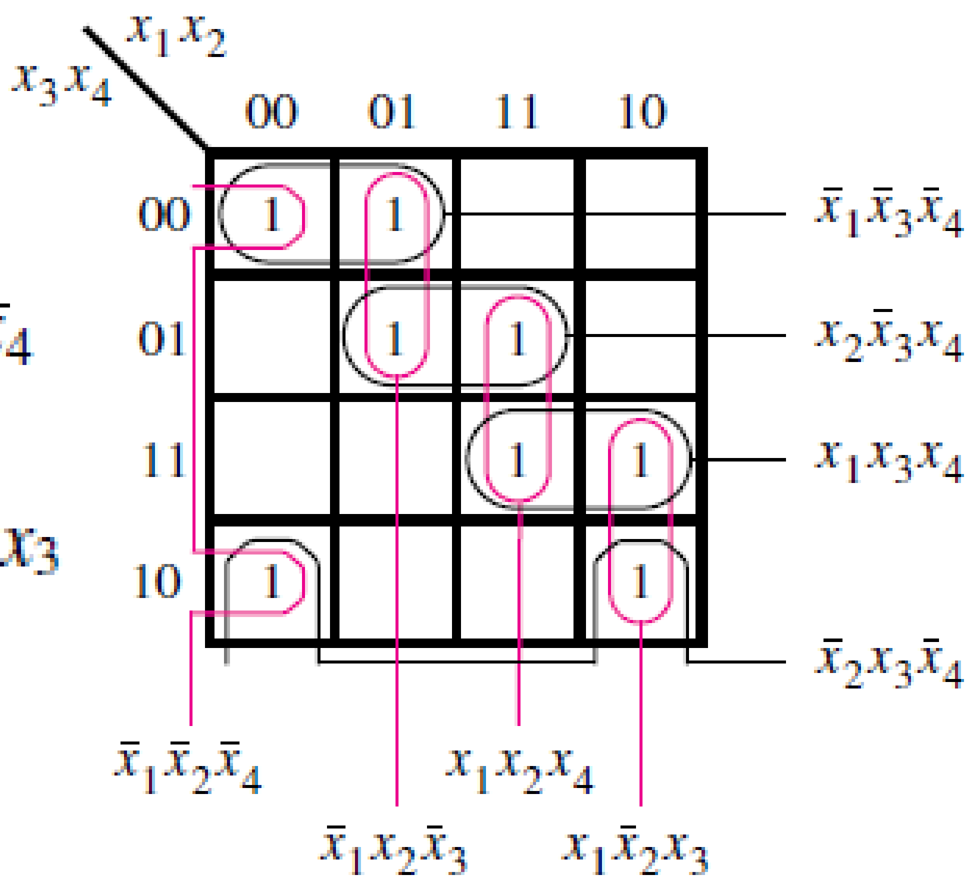
There are no essential prime implicants at all.

Sometimes there may not be any essential prime implicants at all. Example:

Two alternatives:

$$f = \bar{x}_1\bar{x}_3\bar{x}_4 + x_2\bar{x}_3x_4 + x_1x_3x_4 + \bar{x}_2x_3\bar{x}_4$$

$$f = \bar{x}_1\bar{x}_2\bar{x}_4 + \bar{x}_1x_2\bar{x}_3 + x_1x_2x_4 + x_1\bar{x}_2x_3$$



Minimization of Product-of-Sums Forms

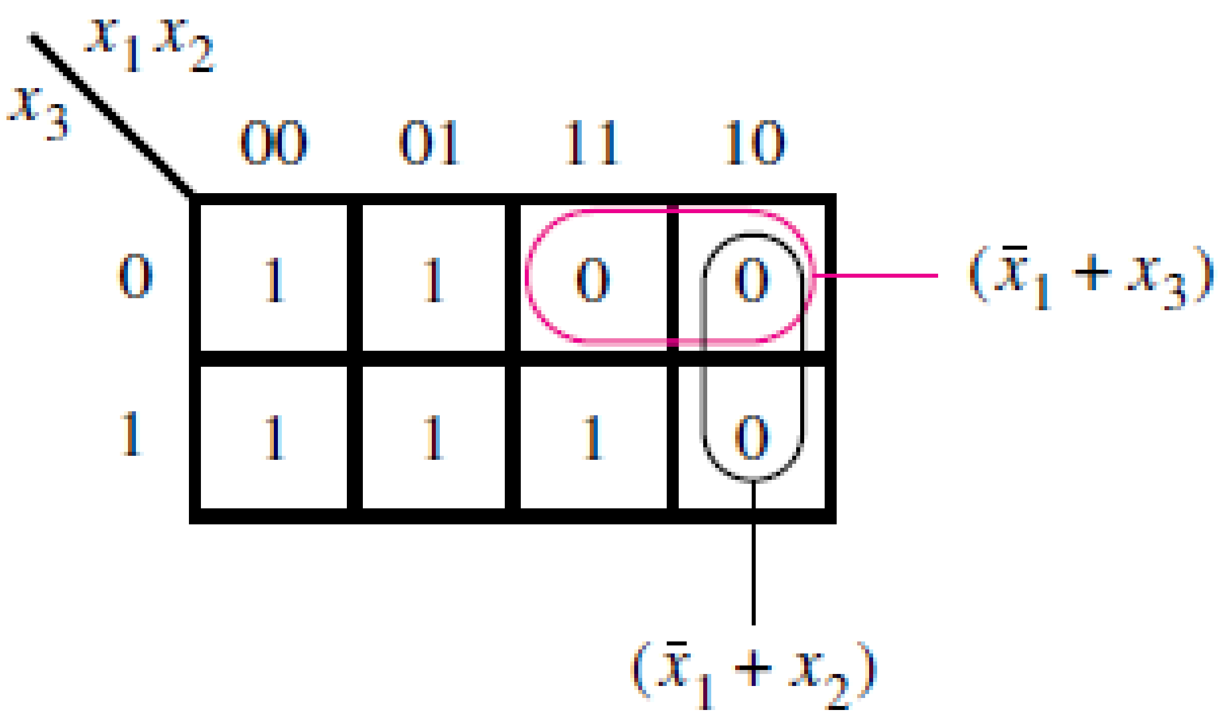
Example1:

x1x2 / x3	00	01	11	10
0	1	1	0	0
1	1	1	1	0

Minimization of Product-of-Sums Forms

Example1:

$$f = (\overline{x}_1 + x_2)(\overline{x}_1 + x_3)$$



Its cost is greater than the cost of the equivalent SOP implementation

$$f = \overline{x}_1 + x_2 x_3$$

SOP COST: 6 (assuming input variables are available in both true and complemented forms)

$$f = (\overline{x}_1 + x_2)(\overline{x}_1 + x_3)$$

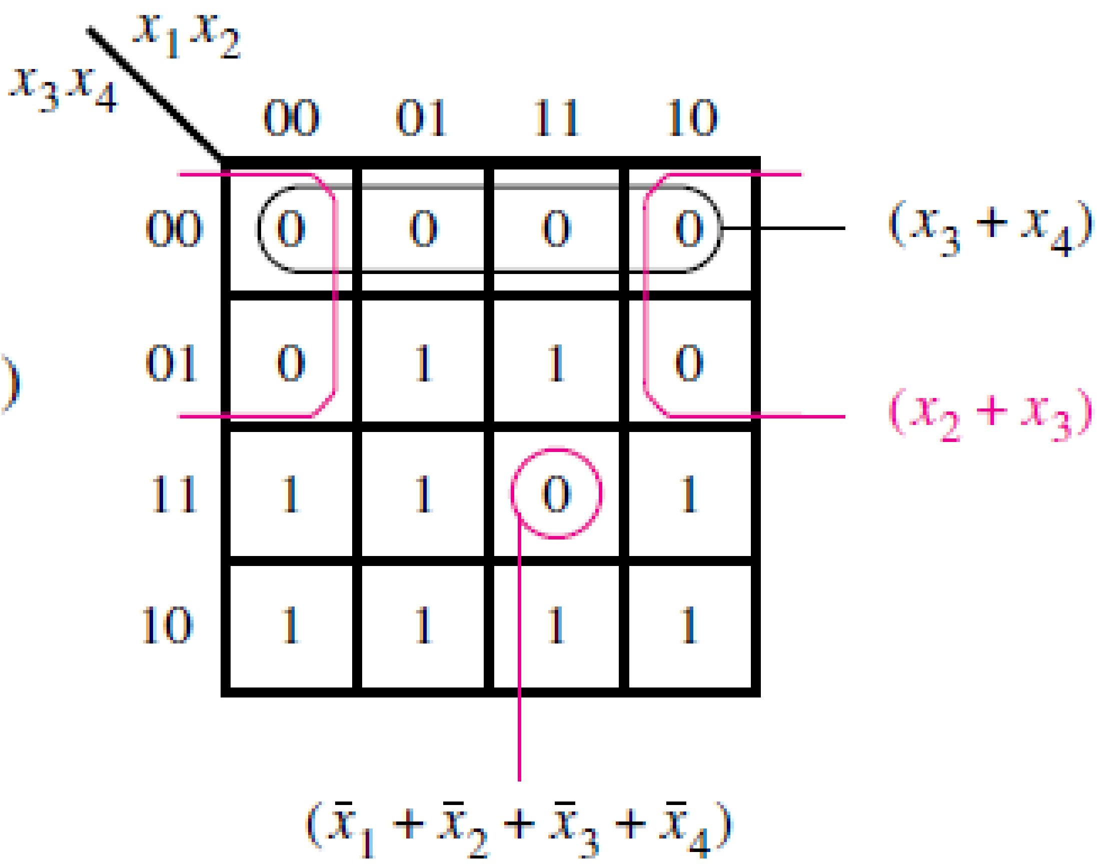
POS COST: 9

Example2:

X3, x4 \ X1, x2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	0	1
10	1	1	1	1

Example2:

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



Assuming that both the complemented and uncomplemented versions of the input variables $x1$ to $x4$ are available,

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

SOP COST: 18

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

POS COST: 15

SOP and POS implementations of a given function may or may not entail the same cost.

Incompletely Specified Functions

In digital systems it often happens that certain input conditions can never occur. For example, suppose that $x1$ and $x2$ control two interlocked switches such that both switches cannot be closed at the same time. Then the input valuations (x1, x2) = 11 is guaranteed not to occur. Then we say that (x1, x2) = 11 is a don’t-care condition, meaning that a circuit

with x1 and x2 as inputs can be designed by ignoring this condition. A function that has don’t-care condition(s) is said to be incompletely specified.

Incompletely Specified Functions

- A function that has don’t-care condition(s) is said to be incompletely specified.
- The don’t cares are denoted by the letter *d* in the map.
- A function with don’t care condition is specified as follows:

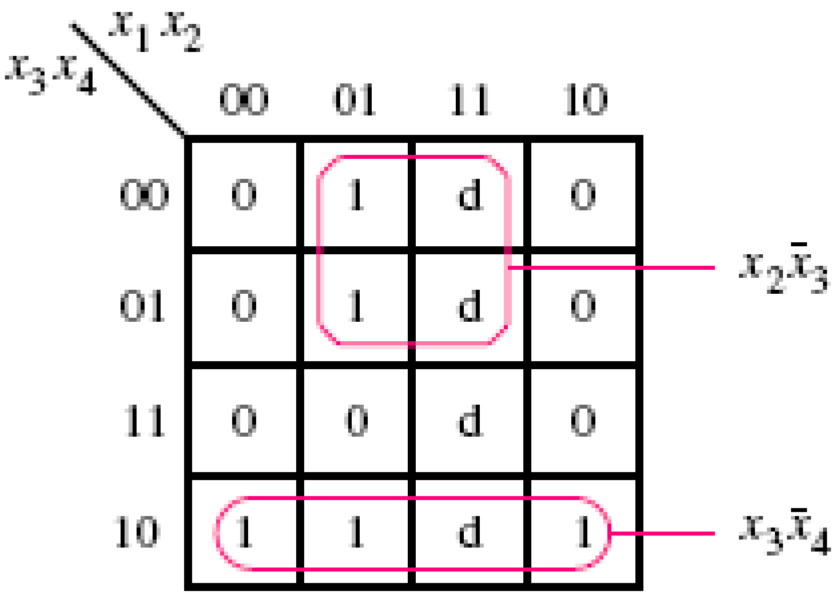
$$f(x_1, \ldots, x_4) = \sum m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$$

where *D* is the set of don’t-cares.

$$f(x_1, \ldots, x_4) = \sum m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$$

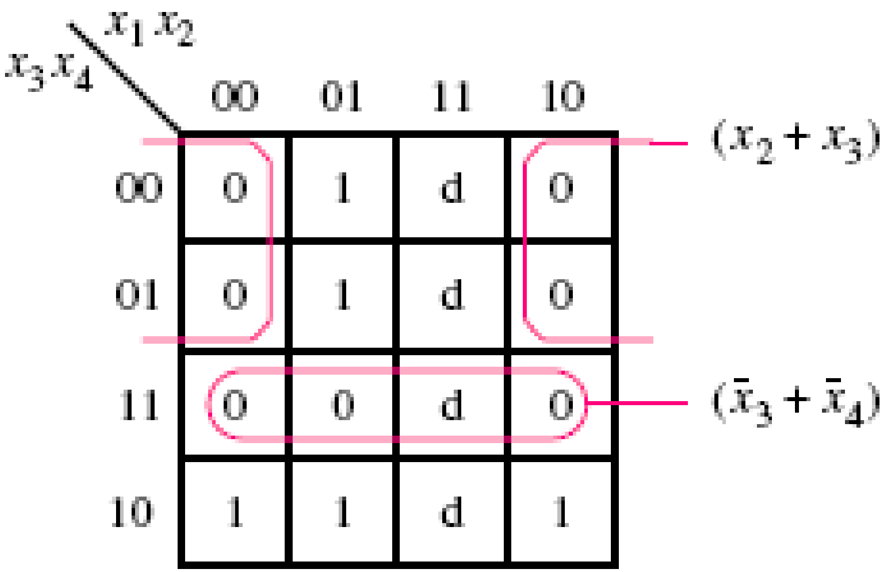
X3, x4 \ X1, x2	00	01	11	10
00	0	1	d	0
01	0	1	d	0
11	0	0	d	0
10	1	1	d	1

Incompletely Specified Functions



(a) SOP implementation

$$f = x_2\overline{x}_3 + x_3\overline{x}_4$$



(b) POS implementation

$$f = (x_2 + x_3)(\overline{x}_3 + \overline{x}_4)$$

Minimal SoP or PoS – two levels NAND and NOR Implementation

$$f = x_2 + x_1\overline{x}_3$$

$$f = (x_1 + x_2)(x_2 + \overline{x}_3)$$