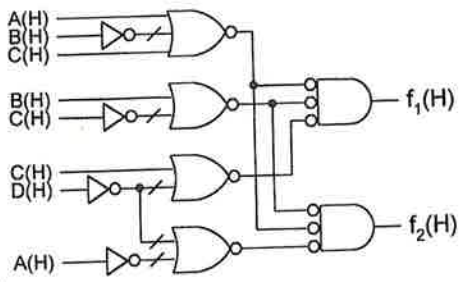

FIGURE 4.27
NOR/INV logic circuit for the optimized POS system of Fig. 4.25.

making use of all shared PIs in the table of Fig. 4.26a together with the required additional p-term cover yields a combined gate/input tally of 7/22.

Comparing the POS and SOP results with optimum system covers of cardinality 4 and 5, respectively, it is clear that the POS result is the more optimum (gate/input tally of 6/15 or 10/19 including inverters). Shown in Fig. 4.27 is the optimal NOR/INV logic implementation of the POS results given by Eqs. (4.37).

The simple search method used here to obtain optimum results becomes quite tedious when applied to multiple output systems more complicated than those just described. For example, a four-input/four-output SOP optimization problem would require at least 10 ANDed fourth-order K-maps, including one for each of six ANDed pairs. For systems this large and larger it is recommended that a computer optimization program (Appendix B) be used, particularly if a guaranteed optimum cover is sought. Optimum cover, as used here, means the least number of gates required for implementation of the multiple output system. Obviously, the number of inverters required and fan-in considerations must also be taken into account when appraising the total hardware cost.

4.6 ENTERED VARIABLE K-MAP MINIMIZATION

Conspicuously absent in the foregoing discussions on K-map function minimization is the treatment of function minimization in K-maps of lesser order than the number of variables of the function. An example of this would be the function reduction of five or more variables in a fourth-order K-map. In this section these problems are discussed by the subject of *entered variable* (EV) *mapping,* which is a "logical" and very useful extension of the conventional (1's and 0's) mapping methods developed previously.

Properly used, EV K-maps can significantly facilitate the function reduction process. But function reduction is not the only use to which EV K-maps can be put advantageously. Frequently, the specifications of a logic design problem lend themselves quite naturally to EV map representation from which useful information can be obtained directly. Many examples of this are provided in subsequent chapters. In fact, EV (entered variable) K-maps are the most common form of graphical representation used in this text.

If N is the number of variables in the function, then map entered variables originate when a conventional Nth-order K-map is compressed into a K-map of order $n < N$ with terms of $(N - n)$ variables entered into the appropriate cells of the nth-order K-map. Thus,

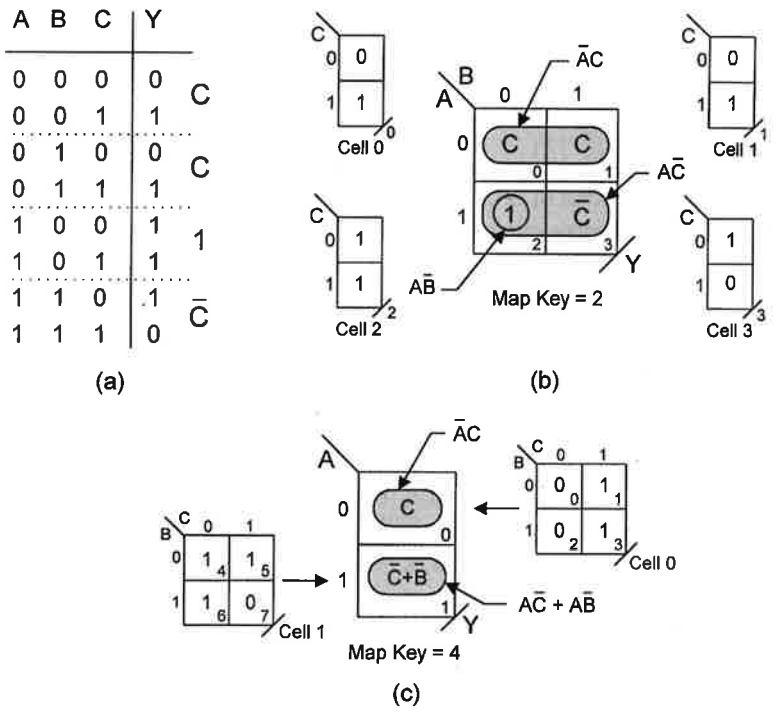

FIGURE 4.28
(a) Truth table for function Y in Eq. (4.39) showing subfunctions for a first-order map compression. (b), (c) Second and first-order EV K-maps showing submaps and minimum SOP cover extracted in minterm code.

each cell of the nth-order K-map becomes a submap of order $(N - n)$, hence K-maps within K-maps.

To illustrate, consider the three-variable function

$$Y(A, B, C) = \sum m(1, 3, 4, 5, 6), \tag{4.39}$$

which has been placed in a truth table and mapped into a second-order EV K-map, as shown in Figs. 4.28a and 4.28b. The subfunctions indicated to the right of the truth table are also represented as first-order submaps corresponding to the cells 0, 1, 2, and 3 in the EV K-map of Fig. 4.28b. The minimum cover is then obtained by looping out the cell entries, as shown by the shaded loops, giving the minimum result

$$Y_{SOP} = \bar{A}C + A\bar{C} + A\bar{B}. \tag{4.40}$$

Notice that the term $A\bar{C}$ covers only the $\bar{C}$ in the $1 = C + \bar{C}$ of cell 2. This requires that the C in the 1 be covered by one of the two OPIs, $A\bar{B}$ or $\bar{B}C$, and the former is chosen.

The same result can be obtained from a second-order compression if the expression of Eq. (4.39) is compressed into a first-order K-map. This is done in Fig. 4.28c, where B and C are now the EVs. The minimum cover is indicated by the shaded loops, yielding

the expression in Eq. (4.40). The OPI $\bar{B}C$ is not easily seen in the first-order EV K-map, but can be found by observing the 1's representing $\bar{B}C$ in the two submaps shown in Fig. 4.28c.

Map Key It has already been pointed out that each cell of the compressed nth-order K-map represents a submap of order $(N-n)$ for an $N > n$ variable function. Thus, each submap covers 2^{N-n} possible minterms or maxterms. This leads to the conclusion that any compressed nth-order K-map, representing a function of $N > n$ variables, has a Map Key defined by

$$\text{Map Key} = 2^{N-n} \quad N > n \tag{4.41}$$

The Map Key has the special property that when multiplied by a cell code number of the compressed nth-order K-map there results the code number of the first minterm or maxterm possible for that cell. Furthermore, the Map Key also gives the maximum number of minterms or maxterms that can be represented by a given cell of the compressed nth-order K-map. These facts may be summarized as follows:

Conventional K-map: Map Key = 1 (no EVs, 1's and 0's only)
First-order compression K-map: Map Key = 2 (one EV)
Second-order compression K-map: Map Key = 4 (two EVs)
Third-order compression K-map: Map Key = 8 (three EVs), etc.

As an example, the first-order compressed K-map in Fig. 4.28b has a Map Key of $2^{3-2} = 2$. So each of its cells represents two possible minterms (first-order submaps) beginning with minterm code number equal to (Map Key = 2) × (Cell Number). This is evident from an inspection of the truth table in Fig. 4.28a. Similarly, the second-order compression in Fig. 4.28c has a Map Key of $2^{3-1} = 4$. Therefore, each cell represents four possible minterms represented by the conventional second-order submaps shown to the sides of Fig. 4.28c.

The compressed K-maps in Fig. 4.28 can also be read in maxterm code as indicated by the shaded loops in Fig. 4.29. In this case the logic 1 in cell 2 must be excluded. The result for either the first-order or second-order compressed K-maps is

$$Y_{POS} = (\bar{A} + \bar{B} + \bar{C})(A + C). \tag{4.42}$$

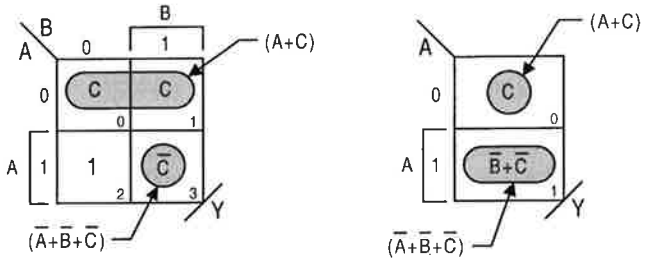

FIGURE 4.29
Second- and first-order EV K-maps showing minimum POS cover for function Y extracted in maxterm code.

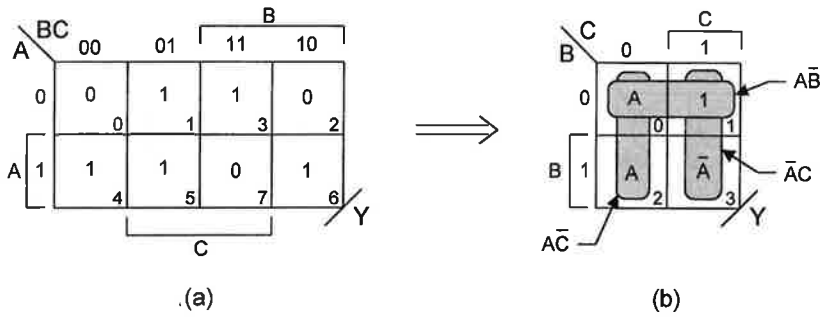

FIGURE 4.30
(a) Conventional K-map for function Y of Eq. (4.39). (b) Second-order EV K-map with entered variable A showing minimum cover for Y as extracted in minterm code.

That Y_{POS} in Eq. (4.40) and Y_{SOP} in Eq. (4.42) are algebraically equal is made evident by carrying out the following Boolean manipulation:

$$(\bar{A} + \bar{B} + \bar{C})(A + C) = \bar{A}C + A\bar{C} + [A\bar{B} + \bar{B}C],$$

where the two p-terms in brackets are OPIs, thereby rendering one to be redundant.

In the second-order K-maps of Figs. 4.28 and 4.29, C is taken to be the EV. However, any of the three variables could have been chosen as the EV in the first-order compression K-maps. As an example, variable A is the EV in Fig. 4.30, where the columns in the conventional K-map of (a) form the submaps of the cells in the compressed K-map of Fig. 4.30b. Minimum cover extracted in minterm code then yields the same result as Eq. (4.40). Or, if extracted in maxterm code, Eq. (4.42) would result. Thus, one concludes that the choice of EVs in a compressed K-map does not affect the extracted minimum result.

Reduced but nonminimum functions can be easily compressed into EV K-maps. This is demonstrated by mapping the four-variable function

$$X = B\bar{C}D + \bar{A}\bar{B} + A\bar{C}\bar{D} + \bar{A}BCD + A\bar{B}\bar{C} \tag{4.43}$$

into the third-order EV K-maps shown in Fig. 4.31, where the Map Key is 2. Here, D is the EV and $1 = (D + \bar{D})$. Figure 4.31a shows the p-terms (loops) exactly as presented in Eq. (4.43). However, regrouping of the logic adjacencies permits minimum SOP and POS cover to be extracted. This is done in Figs. 4.31b and 4.31c, yielding

$$\begin{aligned} X_{SOP} &= \bar{A}D + A\bar{C} + \bar{A}\bar{B} \\ X_{POS} &= (A + \bar{B} + D)(\bar{A} + \bar{C}), \end{aligned} \tag{4.44}$$

where the expressions for X_{SOP} and X_{POS} represent gate/input tallies of 4/9 and 3/7, respectively, excluding possible inverters.

The four-variable function X in Eq. (4.43) can also be minimized in a second-order EV K-map. Shown in Fig. 4.32 is the second-order compression and minimum SOP and POS cover for this function, giving the same results as in Eqs. (4.44). Notice that after covering the D in cell 1 of Fig. 4.32a, it is necessary to cover all that remains in cell 0 by looping out the 1 as an island to give $\bar{A}\bar{B}$. In this case the 1 has the value $1 = C + \bar{C} = D + \bar{D}$. Clearly, the 1 in cell 0 cannot be used in extracting minimum cover in maxterm code.

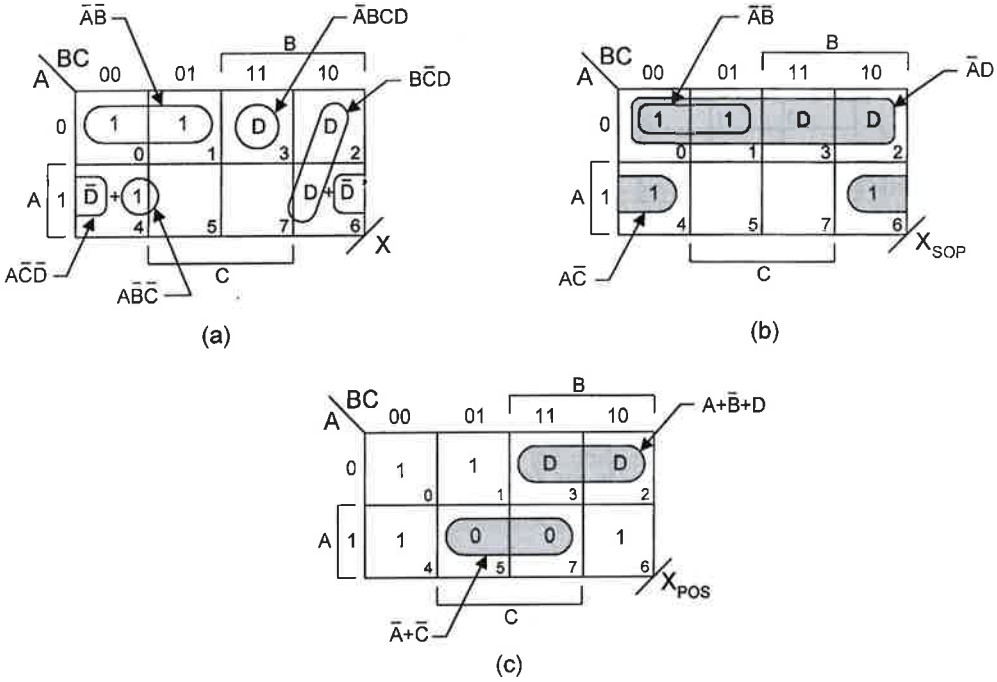

FIGURE 4.31
(a) First-order compression plot of the function X in Eq. (4.43) showing original p-terms. (b) Minimum SOP cover. (c) Minimum POS cover.

4.6.1 Incompletely Specified Functions

The EV mapping method is further illustrated by compressing the incompletely specified function

$$f(A, B, C, D) = \sum m(3, 6, 9, 10, 11) + \phi(0, 1, 4, 7, 8) \tag{4.45}$$

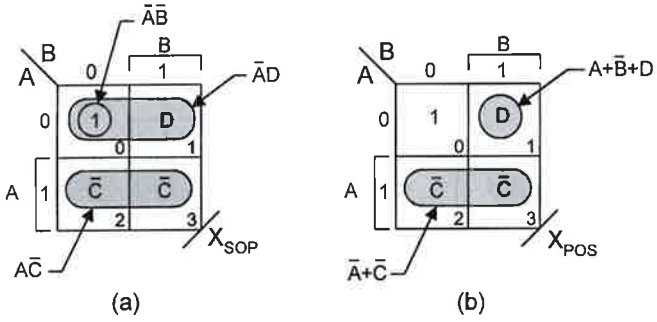

FIGURE 4.32
Second-order compressions of the function X showing (a) minimum SOP cover and (b) minimum POS cover.

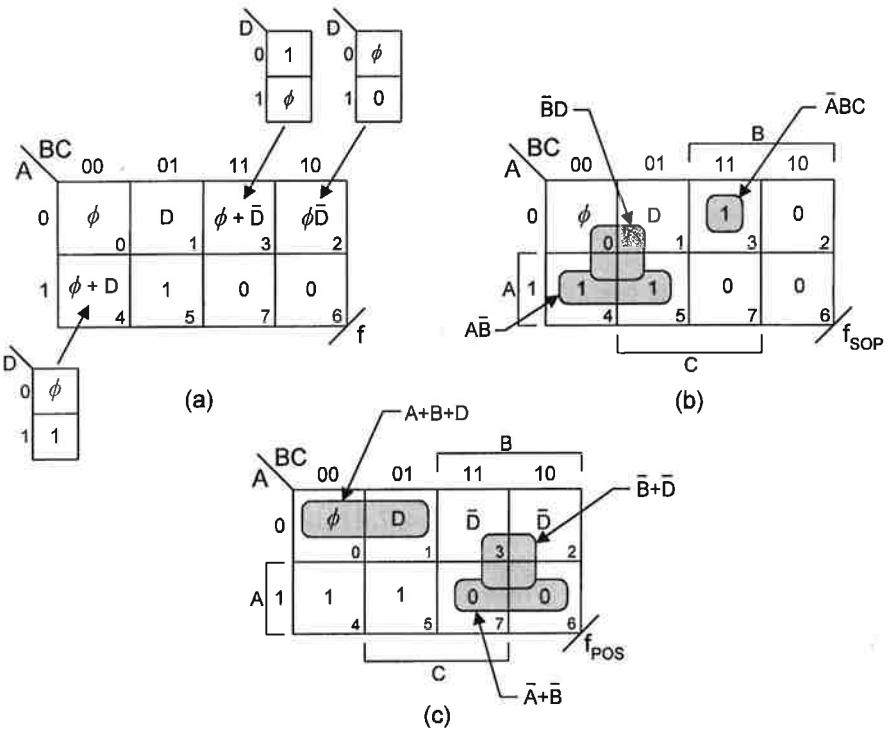

FIGURE 4.33
(a) First-order compression plot and submaps for the function f in Eq. (4.45). (b) Minimum SOP cover and (c) minimum POS cover.

into the third-order K-map in Fig. 4.33a, a first-order compression with a Map Key of 2. Here, the subfunctions are presented in their simplest form yet preserving all canonical information. In Figs. 4.33b and 4.33c are shown the minimum SOP and POS covers for this function, which produce the expressions

$$\begin{aligned} f_{SOP} &= \bar{B}D + \bar{A}BC + A\bar{B} \\ f_{POS} &= (A + B + D)(\bar{B} + \bar{D})(\bar{A} + \bar{B}), \end{aligned} \tag{4.46}$$

both of which have a gate/input tally of 4/10. In extracting the minimum expressions of Eqs. (4.46), the loop-out protocol is first applied to the entered variable D and then applied to the 1's or 0's.

Some observations are necessary with regard to Fig. 4.33 and Eqs. (4.46). First, these expressions are logically equivalent but are *not* algebraically equal. The reason is that the don't cares ϕ_4 and ϕ_7 in cells 2 and 3 are used differently for the f_{SOP} and f_{POS}. For example, $(\phi_7 + \bar{D})_{SOP} = 1$ for $\phi_7 = 1$ but $(\phi_7 + \bar{D})_{POS} = \bar{D}$, since, in this case, $\phi_7 = 0$. Second, the extraction process involved some techniques in dealing with ϕ's that have not been discussed heretofore. These techniques are set off for reference purposes by the following:

Remember:

- Treat the don't care (ϕ) as an *entered variable* — which it is.
- In simplifying incompletely specified subfunctions, apply the absorptive laws:

$$X + \phi\bar{X} = X + \phi$$
$$X \cdot (\phi + \bar{X}) = \phi X.$$

- Subfunctions of the type $(\phi + X)$ have an essential SOP component but no essential POS component. (Proved by substituting the set {0, 1} for ϕ.)
- Subfunctions of the type ϕX have an essential POS component but no essential SOP component. (Proved by substituting the set {0, 1} for ϕ.)

Concluding this section is the function

$$Z(A, B, C, D) = \prod M(2, 4, 7, 11, 12, 14, 15)$$
$$= \sum m(0, 1, 3, 5, 6, 8, 9, 10, 13), \qquad (4.47)$$

which is represented by the second-order EV K-maps in Fig. 4.34, where C and D are the EVs and the Map Key is 4. This example is interesting because of the XOR function in cell 1, which must be represented by both the SOP and POS defining relations, given in Eqs. (3.4), so as to extract minimum SOP and POS cover. To assist the reader in identifying the subfunctions, second-order conventional submaps in C and D axes are shown for each cell. Thus, the subfunction for cell 0 is $\sum m(0, 1, 3) = \bar{C} + D$, while that for cell 1 is

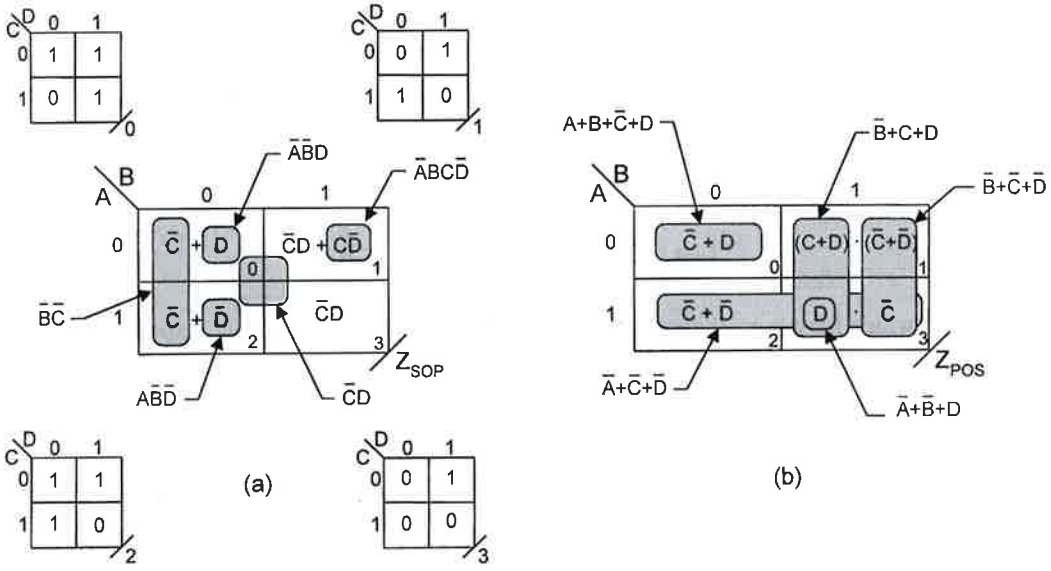

FIGURE 4.34
Second-order EV K-maps and submaps for Eqs. (4.47) showing (a) minimum SOP cover and (b) minimum POS cover.

$\sum m(5, 6) = C \oplus D = C\bar{D} + \bar{C}D = (C + D)(\bar{C} + \bar{D})$. The minimum SOP and POS results are given by

$$\begin{aligned} Z_{SOP} &= \bar{A}BC\bar{D} + \bar{A}\bar{B}D + A\bar{B}\bar{D} + \bar{C}D + \bar{B}\bar{C} \\ Z_{POS} &= (A + B + \bar{C} + D)(\bar{B} + \bar{C} + \bar{D})(\bar{B} + C + D)(\bar{A} + \bar{B} + D)(\bar{A} + \bar{C} + \bar{D}). \end{aligned} \tag{4.48}$$

From the results depicted in Fig. 4.34, certain conclusions are worth remembering and are set off by the following:

- In minterm code, subfunctions of the type XY are subsets of forms of the type $X + Y$.
- In maxterm code, subfunctions of the type $X + Y$ are subsets of forms of the type XY.

What this means is that subfunctions of the type XY can be looped out from terms of the type $X + Y$ to produce reduced SOP cover. For reduced POS cover, subfunctions of the type $X + Y$ can be looped out from terms of the type XY (there are more 0's in XY than in $X + Y$). For example, in Fig. 4.34 $\bar{C}D$ is looped out of both $\bar{C} + D$ and $\bar{C} + \bar{D}$ to contribute to minimum SOP cover. However, in Fig. 4.34b both $C + D$ and $\bar{C} + \bar{D}$ are looped out of $\bar{C}D$, leaving $\bar{C} + D$ to be covered by $\bar{A} + \bar{B} + D$.

4.7 FUNCTION REDUCTION OF FIVE OR MORE VARIABLES

Perhaps the most powerful application of the EV mapping method is the minimization or reduction of functions having five or more variables. However, beyond eight variables the EV method could become too tedious to be of value, given the computer methods available. The subject of computer-aided minimization tools is covered in Appendix B.

Consider the function

$$F(A, B, C, D, E) = \sum m(3, 11, 12, 19, 24, 25, 26, 27, 28, 30), \tag{4.49}$$

which is to be compressed into a fourth-order K-map. Shown in Fig. 4.35 is the first-order compression (Map Key $= 2$) and minimum SOP and POS cover for the five variable function in Eqs. (4.49). The minimized results are

$$\begin{aligned} F_{SOP} &= BC\bar{D}\bar{E} + \bar{C}DE + AB\bar{E} + AB\bar{C} \\ F_{POS} &= (A + \bar{D} + E)(\bar{C} + \bar{E})(B + E)(A + C + D)(B + D), \end{aligned} \tag{4.50}$$

which have gate input tallies of 5/17 and 6/17, respectively. Thus, the SOP result is the simpler of the two. Also, since there are no don't cares involved, the two expressions are algebraically equal. Thus, one expression can be derived from the other by Boolean manipulation.

A more complex example is presented in Fig. 4.36, where the six-variable function

$$\begin{aligned} &Z(A, B, C, D, E, F) \\ &= \sum m(0, 2, 4, 6, 8, 10, 12, 14, 16, 20, 23, 32, 34, 36, 38, 40, \\ &\qquad 42, 44, 45, 46, 49, 51, 53, 54, 55, 57, 59, 60, 61, 62, 63) \end{aligned} \tag{4.51}$$