The logic design methods described in the preceding chapters are based on ANDs, ORs, and NOTs. However, for arithmetic circuits, error correcting circuits, and telecommunication circuits, effective use of EXOR gates can reduce the complexity of networks. In this chapter, we will consider AND-EXOR two-level expressions and their simplifications. These methods will be a base for logic synthesis using EXOR gates. In addition, this chapter introduces Boolean difference and fault detection, as applications of EXOR operations.

13.1 CLASSIFICATION OF AND-EXOR EXPRESSIONS

In this section, we will define various classes of AND-EXOR expressions, and show relations among them.

Theorem 13.1 (Expansion theorem) An arbitrary logic function $f(x_1, x_2, \ldots, x_n)$ can be expanded as

$$f(x_1, x_2, \ldots, x_n) = f_0 \oplus x_1 f_2$$  \hspace{1cm} (13.1)
$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 f_2 \oplus f_1$$  \hspace{1cm} (13.2)
$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 f_0 \oplus x_1 f_1,$$  \hspace{1cm} (13.3)

where $f_0 = f(0, x_2, \ldots, x_n)$, $f_1 = f(1, x_2, \ldots, x_n)$, and $f_2 = f_0 \oplus f_1$.

Equations (13.1)–(13.3) are the positive Davio expansion, the negative Davio expansion, and the Shannon expansion, respectively. The names of
the expansions (13.1) and (13.2) come from Prof. M. Davio who did pioneering work on AND-EXOR logical expressions.

**Definition 13.1** By expanding the function $f$ using (13.1) recursively, we have a logical expression with only un-complemented literals:

$$a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \cdots \oplus a_{n-1,n} x_{n-1} x_n \oplus \cdots \oplus a_{12 \cdots n} x_1 x_2 \cdots x_n.$$ (13.4)

This is a positive polarity Reed-Muller expression (PPRM).

In Section 3.5, we called it a Reed-Muller canonical expression. For a logic function, the PPRM is unique and is a canonical expression. Thus, no minimization problem exists. The average number of products in PPRMs for $n$-variable functions is $2^{n-1}$.

**Example 13.1** Let us represent the function $f = x_1 x_2 x_3$ by a PPRM.

By substituting $x_1 = x_1 + 1$, $x_2 = x_2 + 1$, $x_3 = x_3 + 1$, we have

$$f = (x_1 + 1)(x_2 + 1)(x_3 + 1) = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1 x_2 \oplus x_2 x_3 \oplus x_1 x_3 \oplus x_1 x_2 x_3.$$ Note that this expression uses un-complemented literals only.

In general, $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n$ requires $2^n$ products in a PPRM.

**Definition 13.2** By applying the positive Davio expansion or the negative Davio expansion to the given function $f$, we have a logical expression which has similar form as a PPRM. In this case, assume that we can use either un-complemented literals or complemented literals but not both for each variable. Such logical expression is a fixed polarity Reed-Muller expression (FPRM).

For an $n$-variable function, there are at most $2^n$ different FPRMs. The minimization problem is to find one with the minimum products among $2^n$ possible FPRMs. Among the minimization algorithms of FPRMs, there is one that requires $O(3^n)$ memory storage and computation time.

**Example 13.2** Let us represent the function $f = x_1 x_2 x_3 x_4 \lor \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$ by an FPRM.

From the Lemma 3.2, $f$ can be represented as $f = x_1 x_2 x_3 x_4 \oplus \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$. By applying the positive Davio expansion to $x_1$ and $x_2$, and the negative Davio
expansion to \( x_3 \) and \( x_4 \), we have the expression where \( x_1 \) and \( x_2 \) appear as uncomplemented literals, and \( x_3 \) and \( x_4 \) appear as complemented literals. Thus, by substituting \( \bar{x}_1 = x_1 + 1 \), \( \bar{x}_2 = x_2 + 1 \), \( x_3 = \bar{x}_3 + 1 \), and \( x_4 = \bar{x}_4 + 1 \) to \( f \), we have

\[
\begin{align*}
 f &= x_1 x_2 (\bar{x}_3 + 1)(\bar{x}_4 + 1) + (x_1 + 1)(x_2 + 1)\bar{x}_3 \bar{x}_4 \\
 &= x_1 x_2 (1 \oplus \bar{x}_3 \oplus \bar{x}_4 \oplus \bar{x}_3 \bar{x}_4) + (1 \oplus x_1 \oplus x_2 \oplus x_1 x_2) \bar{x}_3 \bar{x}_4 \\
 &= x_1 x_2 + x_1 x_2 \bar{x}_3 + x_1 x_2 \bar{x}_4 + x_1 x_2 \bar{x}_4 + x_1 x_3 \bar{x}_4 + x_2 \bar{x}_3 \bar{x}_4.
\end{align*}
\]

Note that the last expression is an FPRM. Fig. 13.1 shows the expansion tree. In the tree, the nodes with pD denote the positive Davio expansions, and the nodes with nD denote the negative Davio expansions. In each path from the root node to the constant 1, the logical products of the labels in the path corresponds to a product term in an FPRM. Note that each variable uses the same type of expansion in an FPRM.

In general, \( x_1 x_2 \cdots x_n \lor \bar{x}_1 \bar{x}_2 \cdots \bar{x}_n \) requires \( 2^{r+1} - 2 \) products in an FPRM.

**Definition 13.3** When the given function \( f \) is expanded by either the positive Davio expansion, the negative Davio expansion, or the Shannon expansion, we have a logical expression which is a generalization of FPRMs. Such an expression is a Kronecker expression (KRO).

For an \( n \)-variable function, there are at most \( 3^n \) different KROs.
Example 13.3 Let us represent the function \( f = x_1x_2x_3x_4 \lor \bar{x}_1\bar{x}_2x_3\bar{x}_4 \) by a KRO. By applying the Shannon expansions for all the variables, \( f \) is represented as \( f = x_1x_2x_3x_4 \lor \bar{x}_1\bar{x}_2x_3\bar{x}_4 \). Fig. 13.2 is the expansion tree for this function. In this tree, the nodes with S denote the Shannon expansion. There are two paths from the root node to the constant 1 nodes. They correspond to \( x_1x_2x_3x_4 \) and \( x_1x_2x_3\bar{x}_4 \). Note that all the variables use the same type of expansions.

Definition 13.4 For a given function \( f \), if \( f \) is expanded by the positive Davio expansion or the negative Davio expansion, then we have two sub-functions. For each of these sub-functions, if we expand it by the positive Davio expansion or the negative Davio expansion, where we may use different expansion method for each sub-function. Then, we have a logical expression which is a generalization of an FPRM. Such an expression is a pseudo Reed-Muller expression (PSDRM).

In a PSDRM, both the complemented and un-complemented literals may appear at the same time. For a given order of variables for expansion, an \( n \)-variable function have at most \( 2^{2^n-1} \) different PSDRMs.

Example 13.4 Let us represent the function \( f = x_1x_2x_3x_4 \lor \bar{x}_1\bar{x}_2x_3\bar{x}_4 \) by a PSDRM. First, if we use the positive Davio expansion with respect to \( x_1 \), we have

\[
f = 1 \cdot \bar{x}_2\bar{x}_3\bar{x}_4 \lor x_1(x_2x_3x_4 \lor \bar{x}_2\bar{x}_3\bar{x}_4).
\]
\[ f_0 = \overline{x}_2 \overline{x}_3 \overline{x}_4 \] is a PSDRM, where all the variables are expanded with the negative Davio expansion. Next, consider the expansion of \( f_2 = x_2 x_3 x_4 \oplus \overline{x}_2 \overline{x}_3 \overline{x}_4 \). Since this expression has a similar form to the original one, we can expand it as
\[
1 \cdot (x_3 x_4) = 1 \cdot (1 \cdot \overline{x}_3 \overline{x}_4).
\]
Clearly, \( f_2 = x_3 x_4 \) is a PSDRM. Then, we can expand \( f_21 = x_3 x_4 \oplus \overline{x}_3 \overline{x}_4 = 1 \cdot \overline{x}_4 \oplus x_3 \cdot 1 \). Therefore, the expansion is
\[
f = 1 \cdot \overline{x}_2 \overline{x}_3 \overline{x}_4 \oplus x_1 (1 \cdot \overline{x}_3 \overline{x}_4 \oplus x_2 (1 \cdot \overline{x}_4 \oplus x_3 \cdot 1))
\]
\[
= \overline{x}_2 \overline{x}_3 \overline{x}_4 \oplus x_1 \overline{x}_3 \overline{x}_4 \oplus x_1 x_2 \overline{x}_4 \oplus x_1 x_2 x_3.
\]

Fig. 13.3 is the expansion tree for this function. Note that \( x_2 \) and \( x_3 \) use both pD and nD expansions.

In general, \( x_1 x_2 \cdots x_n \lor \overline{x}_1 \overline{x}_2 \cdots \overline{x}_n \) requires \( n \) products in a PSDRM.

**Definition 13.5** For a given function \( f \), if \( f \) is expanded by the positive Davio expansion, the negative Davio expansion or the Shannon expansion, we have two sub-functions. For each of these sub-functions, suppose that we can use any one of the three expansions, then we have a KRO. In this case, if we can choose the expansion method independently for each sub-function, we have an expression which is a generalization of KRO. Such an expression is a pseudo Kronecker expression (PSDKRO).
In a PSDKRO, both complemented and un-complemented literals of the same variable may appear. If we fix the order of the expansion of the variables, there are at most $3^{2^n} - 1$ different PSDKROs for an $n$-variable function. In general, the number of the products in the minimum PSDKRO depends on the order of the expansion.

**Example 13.5** Let us represent a two-variable function $f(x_1, x_2)$ by a PSDKRO.

For example, if we use the Shannon expansion with respect to $x_1$, we have

$$f(x_1, x_2) = \bar{x}_1 f_0(x_2) \oplus x_1 f_1(x_2).$$

For $f_0$, if we use the positive Davio expansion, we have

$$f_0(x_2) = 1 \cdot f_{00} \oplus x_2 f_{02}.$$

For $f_1$, if we use the negative Davio expansion, we have

$$f_1(x_2) = 1 \cdot f_{11} \oplus \bar{x}_2 f_{12}.$$

From these, we have the expansion for $f$ as follows:

$$f(x_1, x_2) = \bar{x}_1 (1 \cdot f_{00} \oplus x_2 f_{02}) \oplus x_1 (1 \cdot f_{11} \oplus \bar{x}_2 f_{12})$$

$$= f_{00} \bar{x}_1 + f_{02} \bar{x}_1 x_2 + f_{11} x_1 + f_{12} x_1 \bar{x}_2,$$

where $f_{ij}$ is a constant 0 or 1. Fig. 13.4 is the expansion tree for the function. In a PSDKRO, three different types of expansions $S$, $pD$, and $nD$ may appear at the same time for the same variables.

![Expansion tree of a two-variable function by a PSDKRO.](image)
Definition 13.6 In an expansion (13.4) for a PPRM, suppose that the polarity of each literal at each product may be chosen arbitrary. In this case, we have the logical expression which is generalization of a PPRM. Such an expression is a generalized Reed-Muller expression (GRM).

There are at most \(2^{n2^{n-1}}\) different GRMs for an \(n\)-variable function. We do not know an efficient minimization algorithm for GRMs. Note that some researchers use GRMs to mean different expressions, so we should be careful not to confuse them.

Example 13.6 Let us represent the two-variable function \(f(x_1, x_2)\) by a GRM. The PPRM for \(f\) is as follows:

\[
  f = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_{12} x_1 x_2.
\]

For example, by complementing the polarities of the variables in \(x_1 x_2\), we have

\[
  f = b_0 \oplus b_1 x_1 \oplus b_2 x_2 \oplus b_{12} \bar{x}_1 \bar{x}_2.
\]

This is a GRM. However, it is not a PSDKRO. Because, if we set \(b_0 = 0\) and \(b_1 = b_2 = b_{12} = 1\), we have

\[
  f = x_1 \oplus x_2 \oplus \bar{x}_1 \bar{x}_2.
\]

It is not a PSDKRO. Since, we cannot represent it by combining the positive Davio expansion, the negative Davio expansion, and the Shannon expansion.

Definition 13.7 A logical expression that combines arbitrary product terms by EXORs is an exclusive-OR sum-of-products expression (ESOP).

An ESOP is the most general AND-EXOR logical expression. For an \(n\)-variable function, there are at most \(3^n\) different ESOPs with \(t\) products. No efficient minimization algorithm is known. So, heuristic simplification algorithms are developed.

Example 13.7 There are 9 ESOPs for two-variable functions with one product: \(\bar{x}\bar{y}, \bar{x} y, \bar{x} \cdot 1, x\bar{y}, x y, x \cdot 1, 1 \cdot \bar{y}, 1 \cdot y, 1\).

Theorem 13.2 An arbitrary \(n\)-variable symmetric function \((n = 2r)\) is represented by an ESOP with at most \(2 \cdot 3^{r-1}\) products.
(Proof) We use the mathematical induction.

When $r = 1$. There are 8 different two-variable symmetric function: $0$, $1$, $x_1 x_2$, $ar{x}_1 \bar{x}_2$, $x_1 \oplus x_2$, $x_1 \oplus \bar{x}_2$, $x_1 \lor x_2 = 1 \lor \bar{x}_1 \bar{x}_2$, $\bar{x}_1 \lor \bar{x}_2 = 1 \lor x_1 x_2$. Each of them can be represented by an ESOP with at most two products.

When $r \geq 2$. Suppose that an arbitrary $(n - 2)$-variable symmetric function $(n = 2r)$ is represented by an ESOP with at most $2 \cdot 3^{r-2}$ products. An arbitrary $n$-variable symmetric function can be represented by

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 \bar{x}_2 f_{00} + \bar{x}_1 x_2 f_{01} + x_1 \bar{x}_2 f_{10} + x_1 x_2 f_{11},$$

where

$$f_{00} = f(0, 0, x_3, \ldots, x_n),$$
$$f_{01} = f(0, 1, x_3, \ldots, x_n),$$
$$f_{10} = f(1, 0, x_3, \ldots, x_n),$$
$$f_{11} = f(1, 1, x_3, \ldots, x_n)$$

are $(n - 2)$-variable symmetric functions. Since $f$ is symmetric, we have $f_{01} = f_{10}$. Thus, we have

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 \bar{x}_2 f_{00} + (x_1 \oplus x_2) f_{01} + x_1 x_2 f_{11}.$$ 

By assigning $x_1 \oplus x_2 = 1 \oplus x_1 x_2 \oplus \bar{x}_1 \bar{x}_2$ to the above expression, we have

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 \bar{x}_2 (f_{00} \oplus f_{01}) + f_{01} \oplus x_1 x_2 (f_{01} \oplus f_{11}).$$

$f_{00}$, $f_{01}$, $f_{11}$ are $(n - 2)$-variable symmetric functions, and $f_{00} \oplus f_{01}$ and $f_{01} \oplus f_{11}$ are also symmetric functions (Theorem 5.11). Note that from the hypothesis of the mathematical induction, each sub-function can be represented by an ESOP with at most $2 \cdot 3^{r-2}$ products. Thus, the number of products in the left-hand side of the above equation is at most

$$3 \cdot (2 \cdot 3^{r-2}) = 2 \cdot 3^{r-1}.$$ 

Form the above discussions, we have the theorem. 

\[ \square \]

\textbf{Relations among Various Classes}

\textbf{Theorem 13.3} Suppose that $\text{PPRM}$, $\text{FPRM}$, $\text{PSDRM}$, $\text{PSDKRO}$, $\text{GRM}$, and $\text{ESOP}$ represent the set of all PPRMs, FPRMs, PSDRMs, PSDKROs,
GRMs, and ESOPs, respectively. Then, we have the following relations:

1. \( \text{PPRM} \subset \text{FPRM} \)
2. \( \text{FPRM} \subset \text{PSDRM} \)
3. \( \text{FPRM} \subset \text{KRO} \)
4. \( \text{KRO} \subset \text{PSDKRO} \)
5. \( \text{PSDRM} \subset \text{PSDKRO} \)
6. \( \text{PSDRM} \subset \text{GRM} \)

(Proof) By definitions, (1)--(5) clearly hold. If we consider an expansion of a PSDRM, then it is also a GRM. Thus, (6) holds.

Example 13.8

- \( xy \oplus yz \oplus zx \): PPRM. (All the literals are positive.)
- \( xy \oplus yz \oplus zx \): FPRM. It is not a PPRM.
  \((x \text{ and } z \text{ have positive literals, } y \text{ has negative literals.})\)
- \( xy \oplus yz \oplus zx \): PSDRM. It is not an FPRM.
  \((y \text{ and } z \text{ have positive and negative literals.})\)
- \( x \oplus xy \oplus x \bar{y} \): PSDKRO. It is not a KRO.
  \((\text{It cannot be represented by a KRO.})\)
- \( x \oplus xy \oplus x \bar{y} \): PSDKRO. It is not a PSDRM.
  \((y \text{ has three different expansions.})\)
- \( x + y + x \bar{y} \): GRM. It is not a PSDKRO.
  \((\text{It cannot be represented by a PSDKRO.})\)
- \( xyz + x \bar{y} \bar{z} \): KRO. It is not a GRM.
  \((x, y, \text{ and } z \text{ have the positive and the negative literals,})\)
  \text{It contains two product terms consisting of the same set of variables.})
- \( x \oplus y \oplus xy \oplus x \bar{y} \): ESOP. It is not a GRM nor a PSDKRO.

Fig. 13.5 shows the relation of Theorem 13.3 and Example 13.8.

Complexity for Various Logic Functions

Arbitrary product terms combined with ORs is a sum-of-products expression (SOP). To represent \( x_1 \oplus x_2 \oplus \cdots \oplus x_n \), a PPRM requires \( n \) products, and an SOP requires \( 2^{n-1} \) products. To represent \( \bar{x}_1 \bar{x}_2 \cdots \bar{x}_n \), a PPRM requires \( 2^n \) products, but other expressions require only one product. To represent \( x_1 x_2 \cdots x_n \lor \bar{x}_1 \bar{x}_2 \cdots \bar{x}_n \quad (n = 2r) \), a PPRM requires \( 2^n - 1 \) products, an FPRM requires \( 2^{r+1} - 2 \) products, a PSDRM requires \( n \) products. Other expressions require only two products. To represent \( x_1 x_2 \lor x_3 x_4 \lor \cdots \lor x_{n-1} x_n \quad (n = 2r) \), an SOP
Figure 13.5 Relations among various classes of AND-EXOR expressions.

Table 13.1 Number of 4-variable functions that require \( t \) products.

<table>
<thead>
<tr>
<th>( t )</th>
<th>PPRM</th>
<th>FPRM</th>
<th>KRO</th>
<th>PSDRM</th>
<th>PSDRM</th>
<th>GRM</th>
<th>ESOP</th>
<th>SOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>836</td>
<td>2268</td>
<td>2212</td>
<td>2268</td>
<td>2212</td>
<td>2268</td>
<td>1804</td>
</tr>
<tr>
<td>3</td>
<td>560</td>
<td>3496</td>
<td>8424</td>
<td>19160</td>
<td>21384</td>
<td>20856</td>
<td>21744</td>
<td>13472</td>
</tr>
<tr>
<td>4</td>
<td>1820</td>
<td>8878</td>
<td>15174</td>
<td>35150</td>
<td>36918</td>
<td>37818</td>
<td>37530</td>
<td>28904</td>
</tr>
<tr>
<td>5</td>
<td>4368</td>
<td>17884</td>
<td>19260</td>
<td>7404</td>
<td>3564</td>
<td>4512</td>
<td>3888</td>
<td>17032</td>
</tr>
<tr>
<td>6</td>
<td>8008</td>
<td>20152</td>
<td>19440</td>
<td>1480</td>
<td>1296</td>
<td>56</td>
<td>24</td>
<td>3704</td>
</tr>
<tr>
<td>7</td>
<td>11440</td>
<td>11600</td>
<td>864</td>
<td>24</td>
<td>0</td>
<td></td>
<td></td>
<td>512</td>
</tr>
<tr>
<td>8</td>
<td>12870</td>
<td>2336</td>
<td>0</td>
<td>24</td>
<td>24</td>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>11440</td>
<td>240</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>8008</td>
<td>32</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4368</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1820</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

av: \( \text{average} \)

av: average
Logic Design Using EXORs

Table 13.2 Number of products for arithmetic functions.

| Data Name | \(|f|\) | PP | FP | PSD | PSD | GRM | ESOP | SOP |
|-----------|-------|----|----|-----|-----|-----|------|-----|
| ADR4      | 255   | 34 | 34 | 34  | 34  | 34  | 31   | 75  |
| LOG8      | 255   | 253| 193| 171 | 119 | 105 | 96   | 123 |
| MLP4      | 225   | 97 | 97 | 97  | 82  | 71  | 61   | 121 |
| NRM4      | 255   | 216| 185| 157 | 140 | 97  | 96   | 71  |
| RDM8      | 255   | 56 | 56 | 56  | 41  | 38  | 31   | 31  |
| ROT8      | 255   | 225| 118| 83  | 74  | 43  | 35   | 57  |
| SQR8      | 255   | 168| 168| 168 | 158 | 136 | 121  | 180 |
| WGT8      | 255   | 107| 107| 107 | 107 | 107 | 54   | 255 |
| SYM9      | 420   | 210| 173| 173 | 127 | 90  | 52   | 84  |

\(|f|\) denotes the number of input combinations that produce non-zero outputs.

Table 13.3 Sufficient numbers of products to realize \(n\)-variable functions \((n = 2^r)\).

<table>
<thead>
<tr>
<th>Function</th>
<th>PP</th>
<th>FP</th>
<th>PSD</th>
<th>PSD</th>
<th>GRM</th>
<th>ESOP</th>
<th>SOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1 \oplus x_2 \oplus \cdots \oplus x_n)</td>
<td>(n)</td>
<td>(n)</td>
<td>(n)</td>
<td>(n)</td>
<td>(n)</td>
<td>(n)</td>
<td>(2^n - 1)</td>
</tr>
<tr>
<td>(\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n)</td>
<td>(2^n)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(x_1 x_2 \cdots x_n)</td>
<td>(2^{n-1})</td>
<td>(2^{r+1-2})</td>
<td>(2)</td>
<td>(n)</td>
<td>(2)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(\lor x_1 \lor x_2 \cdots \lor x_n)</td>
<td>(2^{r-1})</td>
<td>(2^{r-1})</td>
<td>(2^{r-1})</td>
<td>(2^{r-1})</td>
<td>(2^{r-1})</td>
<td>(2^{r-1})</td>
<td>(r)</td>
</tr>
<tr>
<td>(n)-bit adder</td>
<td>(2^{n+1}+n-2)</td>
<td>(2^{n+1}+n-2)</td>
<td>(2^{n+1}+n-2)</td>
<td>(2^{n+1}+n-2)</td>
<td>(6 \cdot 2^n - 4n - 5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

requires \(r\) products, but an AND-EXOR expressions requires \(2^r - 1\) products.

To represent an \(n\)-bit adder, \(2^{n+1} - 1\) products are sufficient for an ESOP, \(2^{n+1} + n - 2\) products are sufficient for other AND-EXOR expressions, and \(6 \cdot 2^n - 4n - 5\) products are sufficient for an SOP. Tables 13.1–13.3 show the number of products needed to represent various functions by various expressions.
13.2 SIMPLIFICATION OF ESOPS

The minimization of an SOP is equivalent to the problem of covering all the 1-cells in the Karnaugh map at least once by using a minimum number of loops. On the other hand, the minimization of an ESOP is equivalent to the problem of covering all the 1-cells an odd number of times and all the 0-cells in even number of times by using minimum number of loops. For example, consider the function shown in Fig. 13.6. An SOP requires five products. On the other hand, the ESOP shown in Fig. 13.7 requires only four products. These expressions are realized by two-level logic networks shown in Fig. 13.8 and Fig. 13.9, respectively. Note that the AND-EXOR realization requires fewer gates and fewer connections.
We have no efficient minimization algorithm for ESOPs. In this section, we will show **EXMIN2**, a heuristic algorithm using simplification rules. In EXMIN2, the following rules are iteratively used to simplify ESOPs:

1) **X.MERGE**
   - $x \oplus x \rightarrow 0$, $x \oplus \bar{x} \rightarrow 1$,
   - $x \oplus 1 \rightarrow \bar{x}$, $\bar{x} \oplus 1 \rightarrow x$

2) **RESHAPE**
   - $xy \oplus \bar{y} \rightarrow \bar{x}\bar{y} + x$

3) **DUAL.COMPLEMENT**
   - $x \oplus y \rightarrow \bar{x} \oplus \bar{y}$

4) **X.EXPAND._1**
   - $xy \oplus \bar{x}y \rightarrow x \oplus y$

5) **X.EXPAND._2**
   - $xy \oplus \bar{y} \rightarrow 1 \oplus \bar{x}y$

6) **X.REDUCE._1**
   - $x \oplus y \rightarrow x \oplus \bar{x}y$

7) **X.REDUCE._2**
   - $1 \oplus \bar{x}y \rightarrow xy \oplus \bar{y}$

8) **SPLIT**
   - $1 \rightarrow x \oplus \bar{x}$

**Example 13.9** Let us simplify the ESOP for the 4-variable function shown in Fig. 13.6. First, as shown in Fig. 13.10(a), transform the expression into a **disjoint sum-of-products expression** (DSOP), so that the product terms have no common minterms. In the DSOP, the operators $\lor$ can be replaced with $\oplus$ without changing the function represented by the expression.

1) Since, X.MERGE cannot be applied in Fig. (a), we apply the RESHAPE operation to the pair of product terms ($\mathbf{1}$, $\mathbf{2}$) to obtain Fig. (b).

2) For the pair of products ($\mathbf{3}$, $\mathbf{4}$) in Fig. (b), we apply the X.EXPAND._2 operation to obtain Fig. (c).

3) For the pair of products ($\mathbf{5}$, $\mathbf{6}$) in Fig. (c), we apply the X.EXPAND._2 operation to obtain Fig. (d). Note that 0-cells $x\bar{y}\bar{z}\bar{w}$, $x\bar{y}\bar{z}w$ and $x\bar{y}\bar{z}w$ are covered twice by loops.

4) For the pair of products ($\mathbf{7}$, $\mathbf{8}$) in Fig. (d), we apply the X.EXPAND._2 operation to obtain Fig. (e). Note that the 1-cell $\bar{x}\bar{y}\bar{z}\bar{w}$ is covered by loops three times.

5) For the pair of products ($\mathbf{1}$, $\mathbf{2}$) in Fig. (e), we apply the X.EXPAND._2 operation to obtain Fig. (f). This is a very complicated map.

6) For the pair of products ($\mathbf{3}$, $\mathbf{4}$) in Fig. (f), we apply the X.MERGE operation. Since $\bar{x}\bar{z} \oplus \bar{z} = (x \oplus 1)\bar{z} = x\bar{z}$, we have Fig. (g). Note that the number of loops is reduced to four.

7) For the pair of products ($\mathbf{5}$, $\mathbf{6}$) in Fig. (g), we apply the X.EXPAND._1 operation to obtain Fig. (h).

8) For the pair of products ($\mathbf{7}$, $\mathbf{8}$) in Fig. (h), we apply the X.EXPAND._2 operation to obtain the ESOP shown in Fig. 13.7. Note that, we cannot reduce the number of products nor number of literals any more.
Figure 13.10 An example of simplification.
13.3 FAULT DETECTION AND BOOLEAN DIFFERENCE

Definition 13.8 Let \( f(x_1, x_2, \ldots, x_n) \) be the output function of a fault-free network, and let \( f_\alpha(x_1, x_2, \ldots, x_n) \) be the output function of the network with the fault \( \alpha \). Then, the fault difference function, which distinguishes the two functions, is

\[
h(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) \oplus f_\alpha(x_1, x_2, \ldots, x_n).
\]

For an input vector \( a = (a_1, a_2, \ldots, a_n) \) that makes \( h = 1 \), the values of \( f \) and \( f_\alpha \) are different. Such an input vector \( a \) is a test input for the fault \( \alpha \). A stuck-at fault is a fault that makes an input or the output terminal to be constant 0 or 1.

Example 13.10 The network in Fig. 13.11 realizes the function \( f = xyz \lor \bar{x}\bar{y}\bar{z} \). If the input terminal of the variable \( x \) of the AND gate has a stuck-at-1 fault, then the output function becomes \( f_\alpha = 1 \cdot yz \lor \bar{x}\bar{y}\bar{z} \). In this case, the fault difference function is

\[
h = (xyz \lor \bar{x}\bar{y}\bar{z}) \oplus (1 \cdot yz \lor \bar{x}\bar{y}\bar{z}).
\]

By simplifying this expression, we have

\[
h = (xyz \oplus \bar{x}\bar{y}\bar{z}) \oplus (yz \oplus \bar{x}\bar{y}\bar{z}) = xyz \oplus yz = \bar{x}yz.
\]

This shows that if \( (x, y, z) = (0, 1, 1) \), then the value of \( f \) is 0, and the value of \( f_\alpha \) is 1. Thus, \( (0, 1, 1) \) is a test input for \( x \) stuck-at-1.

When the output function of the faulty network is equal to that of the fault-free network, the fault difference function is constantly equal to 0. Such a fault is a redundant fault and cannot be detected from the input or output terminals.

Definition 13.9 Let \( f(x_1, x_2, \ldots, x_n) \) be an n-variable logic function. The Boolean difference of the function \( f \) with respect to the variable \( x_i \) is

\[
\frac{df}{dx_i} = f(x_1, x_2, \ldots, \bar{x}_i, \ldots, x_n) \oplus f(x_1, x_2, \ldots, x_i, \ldots, x_n).
\]  
(13.5)
From the Shannon’s expansion theorem, we have
\[
f(x_1, x_2, \ldots, x_i, \ldots, x_n) = x_i f_i(0) \oplus \bar{x}_i f_i(1), \quad \text{and} \\
f(x_1, x_2, \ldots, x_i, \ldots, x_n) = \bar{x}_i f_i(0) \oplus x_i f_i(1),
\]
where
\[
f_i(0) = f_i(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n), \quad \text{and} \\
f_i(1) = f_i(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n).
\]
Therefore, the Boolean difference in (13.5) can be represented as
\[
\frac{df}{dx_i} = (x_i f_i(0) \oplus \bar{x}_i f_i(1)) \oplus (\bar{x}_i f_i(0) \oplus x_i f_i(1)) \\
= f_i(0) \oplus f_i(1).
\]
If \( \frac{df}{dx_i} = 0 \), \( f \) does not depend on \( x_i \). In this case, \( f \) is degenerate.

In a logic network with the output function \( f(x_1, x_2, \ldots, x_n) \), consider a fault \( \alpha \) where the input line \( x_i \) is stuck-at-0. The fault function \( f_\alpha \) produced by the fault \( \alpha \) is
\[
f_\alpha = f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = f_i(0).
\]
Also, the fault difference function that detects the fault \( \alpha \) is \( f \oplus f_i(0) \). By modifying it, we have
\[
f \oplus f_i(0) = \bar{x}_i f_i(0) \oplus x_i f_i(1) \oplus f_i(0) \\
= \bar{x}_i f_i(0) \oplus x_i f_i(1) \oplus (\bar{x}_i \oplus x_i) f_i(0) \\
= x_i (f_i(0) \oplus f_i(1)) \\
= x_i \frac{df}{dx_i}.
\]
Therefore, all tests for the stuck-at-0 fault in the input line \( x_i \) is represented by
\[
\{ \alpha \mid x_i \frac{df}{dx_i} = 1 \}, \quad \alpha \in B^n.
\]
Figure 13.12 Derivation of test inputs.

Similarly, the tests for a stuck-at-1 fault in the input line $x_i$ is represented by

$$\{ a \mid \bar{x}_i \frac{df}{dx_i} = 1 \}, \ a \in B^n.$$

**Example 13.11** Let us obtain the test set for the stuck-at-0 fault in the input line $x$ in Fig. 13.12. First, the output function $f$ is represented as

$$f = (xy)(z \lor w).$$

Next, by differentiating $f$ with respect to $x$, we have

$$\frac{df}{dx} = f(x = 0) \oplus f(x = 1) = (z \lor w) \oplus \bar{y}(z \lor w)$$

$$= y(z \lor w).$$

The expression showing the test set of a stuck-at-0 fault in $x$ is

$$x \frac{df}{dx} = xy(z \lor w).$$

Thus, $(x, y, z, w) = (1, 1, 1, 0), (1, 1, 1, 1), \text{ or } (1, 1, 0, 1)$ are the test inputs for $x$ stuck-at-0.

The next example shows that Boolean difference is useful for the simplification of multi-level logic networks.

**Example 13.12** In Fig. 13.13, the network $A$ realizes the function $y = h(x_1, x_2, \ldots, x_n)$ and the network $B$ realizes the function $f = z(x_1, x_2, \ldots, x_n, y)$. In this case, the condition that the output of the network $B$ does not depend on the output of the network $A$ is represented by $dz/dy = 0$. In other words, the inputs such that $z(\bar{y}) \oplus z(\bar{y}) \oplus 1 = 1$ will not influence the value of the output of the network $B$. This is the Observability Don't Care stated in Section 11.7.2.
Bibliographical Notes

Detailed discussions on each topic can be found as follows: Classification of AND-EXORs [89, 149, 150, 266, 352, 358, 364]; PPRMs [213, 269, 323, 439]; FPRM minimization [25, 27, 107, 108, 232, 365, 415]; PSDKRO minimization [360]; KRO minimization [3, 233, 365]; GRM minimization [73, 78, 91, 368, 369]; ESOP heuristic minimization [35, 119, 123, 165, 163, 329, 357, 359, 336, 391, 403]; ESOP exact minimization [211, 266, 308, 312, 361]; other AND-EXOR expressions [203, 427]; EXOR multi-level logic networks [66, 375, 404]; AND-OR-EXOR networks [92, 93, 109, 362]; AND-EXOR test [317, 322, 369, 335, 378, 409]; textbooks on test [125, 131, 217, 318]; fault detection [6, 133, 242, 331, 332, 399, 432]; Boolean difference [1, 377, 400, 401]; adder test [67, 192]; multi-valued EXORs [110]; decision diagrams using EXORs [106, 203, 364, 371]; and spectral techniques [178, 197, 392, 414, 416].
Exercises

13.1 Prove that the PPRM for $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n$ requires $2^n$ products.

13.2 Consider the $n$-variable functions that have $t$ products in their PPRMs. Show that there are $\binom{2^n}{t}$ such functions.

13.3 Prove that the average number of products in the PPRMs for $n$-variable functions is $2^{n-1}$.

13.4 Prove that FPRM for $x_1 x_2 \cdots x_n \lor \bar{x}_1 \bar{x}_2 \cdots \bar{x}_n$ ($n = 2r$) requires $2^{r+1} - 2$ products.

13.5 (M) Show that $x_1 x_2 \lor x_3 x_4 \lor \cdots \lor x_{n-1} x_n$ ($n = 2r$) can be represented by a PPRM with $2^n - 1$ products.

13.6 Show that there are at most $2^{2n-1}$ different PSDRMs for an $n$-variable function.

13.7 Show that there are at most $3^{2n-1}$ different PSDKROs for an $n$-variable function.

13.8 Show that there are at most $2^{n2^n-1}$ different GRMs for an $n$-variable function.

13.9 Convert the following expressions into ESOPs, and then simplify:

\[
F = (1 \oplus xy)(1 \oplus zw) \oplus 1,
G = \bar{x}y\bar{z}w \oplus x\bar{z} \oplus \bar{x}w \oplus x\bar{y}zw \oplus z\bar{w}.
\]

13.10 By using the EXMIN2 rules, simplify the following ESOPs:

\[
F = \bar{x}\bar{y}\bar{z} \oplus \bar{x}y\bar{z} \oplus x\bar{y}\bar{z} \oplus xyz,
G = \bar{x}\bar{z}w \oplus xy\bar{z}w \oplus x\bar{y}zw \oplus xyzw \oplus x\bar{y}z.
\]

13.11 (M) The degree of an AND-EXOR logical expression is the maximum number of literals in the products. Show that the following holds:

(a) An $n$-variable function $f = x_1 x_2 \cdots x_n \lor \bar{x}_1 \bar{x}_2 \cdots \bar{x}_n$ can be represented by a PSDRM with $n$ products and has the degree $n - 1$. 
LOGIC DESIGN USING EXOR GATES

Figure 13.14 Detection of stack-at fault.

Figure 13.15 Detection of stack-at fault.

(b) Let the degree of an ESOP for a function \( f \) be \( k \). Then, the degree of the PPRM for \( f \) is at most \( k \).
(c) If the degree of a PPRM for the function \( f \) is \( k \), then the degree of any ESOP for \( f \) is at least \( k \).
(d) The number of \( n \)-variable functions that can be represented by ESOPs whose degree is at most \( k \) is \( \Phi(n, k) = 2\eta(n, k) \), where \( \eta(n, k) = nC_0 + nC_1 + \cdots + nC_k \).

13.12 (M) Consider the logic function \( f \) that has an odd number of true minterms. Show that any ESOP for \( f \) has at least one minterm. For example, the ESOPs for two-variable function \( f = x \lor y \) are \( x \lor \bar{x} y, x\bar{y} \lor y, 1 \lor \bar{x} \lor \bar{y}, \) etc. All the ESOPs for \( f \) contain minterms of two-variables.

13.13 (M) Let \( A, B, C, D, E, F \subseteq P \), and \( A \oplus B = (\overline{A \cap B}) \cup (A \cap \overline{B}) \). Show the following equations:

\[
X^A \cdot Y^B \cdot Z^C \cdot X^D \cdot Y^E \cdot Z^F = X^{(A \oplus D)} \cdot Y^{(B \oplus E)} \cdot Z^{(C \oplus F)}
\]

13.14 Let \( f \) and \( g \) be functions of \( x, y, z, \ldots \). Prove that the following rules for Boolean difference holds:

1. \( \frac{df}{dx} = \frac{df}{dx} \)
2. \( \frac{df}{dx} = \frac{df}{dx} \)
3. \( \frac{d(f \cdot g)}{dx} = f \cdot \frac{dg}{dx} \oplus g \cdot \frac{df}{dx} \oplus \frac{df}{dx} \cdot \frac{dg}{dx} \).
Exercises

4. \( \frac{d(f \oplus g)}{dx} = \frac{dg}{dx} \oplus \frac{df}{dx} \),

5. \( \frac{d(f \lor g)}{dx} = f \cdot \frac{dg}{dx} \oplus \bar{g} \cdot \frac{df}{dx} \oplus \frac{df}{dx} \cdot \frac{dg}{dx} \).

6. \( \frac{d}{dx} \left( \frac{df}{dy} \right) = \frac{d}{dy} \left( \frac{df}{dx} \right) \).

13.15 Let \( f(x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n \). Show the following:

\[ \frac{df}{dx_i} = 1. \]

Let \( M(x, y, z) = xy \lor yz \lor zx \). Show the following:

\[ \frac{dM}{dx} = y \oplus z. \]

13.16 In Fig. 13.14, are the stack-at faults in the input terminal a of the NAND gate 2 detectable? Discuss the cases for the stack-at-0 fault and the stack-at-1 fault.

13.17 In Fig. 13.15, are the stack-at faults in the terminal b of the NAND gate 1 detectable? Discuss the cases for the stack-at-0 fault and the stack-at-1 fault.

13.18 Let \( X = (x_3, x_2, x_1, x_0) \) be a binary representation of an integer, \( 0 \leq X \leq 15 \). Design an AND-EXOR two-level logic network that realizes \( Y = X+1 \), where \( Y = (y_4, y_3, y_2, y_1, y_0) \).

13.19 Let \( f = xy \lor \bar{x} \bar{z} \).
   1. Represent \( f \) by a PPRM.
   2. Represent \( f \) by a minimum FPRM.

13.20 Show that a function \( f(x_1, x_2, \ldots, x_n) \) is linear iff

\[ \frac{df}{dx_i} = a_i \quad (a_i \in \{0, 1\}) \]

for all \( i \).

13.21 Consider the symmetric function \( f = S^5_{\{2,3\}} \).
   1. Represent \( f \) by a PPRM.
2. Represent $f$ by a minimum ESOP.

13.22 Prove the following:

$$(x_1 \lor \bar{x}_2)(x_2 \lor \bar{x}_3)\cdots(x_n \lor \bar{x}_1) = (x_1 \oplus \bar{x}_2)(x_2 \oplus \bar{x}_3)\cdots(x_n \oplus \bar{x}_1).$$

13.23 (D) Let $SB(n, k)$ be the $n$-variable function represented by the EXOR sum of all the products consisting of $k$ positive literals:

\[
SB(n, 0) = 1, \\
SB(n, 1) = \sum x_i, \\
SB(n, 2) = \sum_{i<j} x_ix_j, \\
SB(n, 3) = \sum_{i<j<k} x_ix_jx_k, \\
\ldots \ldots \ldots \\
SB(n, n) = x_1x_2\cdots x_n.
\]

Let $T(n, k)$ be the number of products in the minimum ESOP for $SB(n, k)$. Prove the following [352, 354]:

\[
T(n, k) \leq \binom{n}{k}, \\
T(n, 0) = 1, T(n, 1) = n, T(n, n) = 1, \\
T(n, k) \leq T(n - 1, k) + T(n - 1, k - 1), \\
T(n, k) = T(n, n - k).
\]