IEICE TRANS. INF. & SYST., VOL.E87–D, NO.5 MAY 2004

PAPER

New Three-Level Boolean Expression Based on EXOR Gates

Ryoji ISHIKAWA[†a)], Takashi HIRAYAMA[††], Goro KODA[†], *and* Kensuke SHIMIZU[†], *Members*

SUMMARY The utilization of EXOR gates often decreases the number of gates needed for realizing practical logical networks, and enhances the testability of networks. Therefore, logic synthesis with EXOR gates has been studied. In this paper we propose a new logic representation: an ESPP (EXOR-Sum-of-Pseudoproducts) form based on pseudoproducts. This form provides a new three-level network with EXOR gates. Some functional classes in ESPP forms can be realized with shorter expressions than in conventional forms such as the Sum-of-Products. Since many practical functions have the properties of such classes, the ESPP form is useful for making a compact form. We propose a heuristic minimization algorithm for ESPP, and we demonstrate the compactness of ESPPs by showing our experimental results. We apply our technique to some logic function classes and MCNC benchmark networks. The experimental results show that most ESPP forms have fewer literals than conventional forms.

key words: *EXOR gates, three-level logic, pseudoproduct, compact design, testability*

1. Introduction

Logic synthesis with EXOR gates has been studied for years [12], [15], [17]–[19]. In many practical networks, the number of gates can be decreased by using EXOR gates. Furthermore, it is known that an EXOR-based network enhances the testability [5], [14], [16], [20]. Therefore, it is very important to develop the logic synthesis with EXOR gates.

The pseudoproduct proposed recently by Luccio et al. is a generalization of the conventional product, and has an EXOR-AND form [4], [10], [11]. They also proposed an EXOR-AND-OR three-level logical form called the Sum-of-Pseudoproducts (SPP) form, which is obtained by replacing the products in the standard Sum-of-Products (SOP) form with the pseudoproducts. SPP forms can represent Boolean functions with fewer literals than SOP forms. Furthermore, our investigations found that SPP forms are more testable than SOP forms [7]. It is known that the class of autosymmetric functions is suitable for efficient SPP representation [1]. The autosymmetric function differs from the usual symmetric function and can be transferred to itself by complementing some variables of the function. Although the ratio of autosymmetric functions to all functions is small, we found that many practical networks have autosymmetries. It is effective for the compactness and the testability of networks to detect networks including autosymmetric functions, and apply SPP forms to them.

When we realize autosymmetric functions in SPP forms, all pseudoproducts connected to the OR gate in the output are disjoint. Therefore, the function is not changed by replacing the OR gate with the EXOR gate. It is known that the utilization of the EXOR gate in the output enhances the testability of the network. If we use the EXOR gate instead of the OR gate in the output of SPP forms, we gain more compactness and testability. Accordingly, we propose a new pseudoproduct-based logical form and show the effectiveness in this paper. This form provides an EXOR-AND-EXOR three-level network. EXOR gates in the first and third level enhance the testability of the network [8]. The new three-level networks can be achieved with fewer gates than conventional two-level networks. We call this new form EXOR-Sum-of-Pseudoproducts (ESPP). ESPP forms are also suitable for parity functions, symmetric functions, and more. Functions belonging to such functional classes can be found in many practical networks. Therefore, it is important to realize ESPP forms in these networks.

We also propose an ESPP minimization algorithm which is heuristic like ESPRESSO-II [3] and EXMIN2 [18]. This algorithm is based on iterative improvements, and requires less time and memory costs than exact minimizations. In our experiments, we apply our algorithm, which is implemented using C language, to some logic functional classes and MCNC benchmark networks [13]. The experimental results show that ESPP forms have fewer literals than conventional forms. In other experiments, we also show that ESPP forms are especially suitable for autosymmetric functions.

This paper is organized as follows. In Sect. 2, we define a pseudoproduct and an ESPP form. In Sect. 3, we give the number of pseudoproducts used to realize some functional classes. Our technique to minimize an ESPP form is in Sect. 4. Experimental results and conclusions are in Sects. 5 and 6, respectively.

2. Definition of ESPP Form

In this section, we provide the definitions of pseudoproduct and ESPP form. We deal with a pseudoproduct proposed by Luccio and Pagli for realizing an EXOR-AND part of our ESPP form in this paper. The next definition is a recursive definition of pseudoproducts [10], [11].

Manuscript received September 1, 2003.
Manuscript revised December 22, 2003.

[†]The authors are with the Department of Computer Science, Gunma University, Kiryu-shi, 376–8515 Japan.
[††]The author is with the Department of Computer and Information Sciences, Iwate University, Morioka-shi, 020–8551 Japan.
a) E-mail: ishikawa@ja4.cs.gunma-u.ac.jp

[Definition 1]

1. Any single point in the Boolean n-space B^n ($B \in \{0, 1\}$) is regarded as a pseudoproduct of degree 0.
2. P with 2^m points is a pseudoproduct of degree m if P can be divided into two disjoint pseudoproducts P_1, P_2 of degree $m-1$, and there exists a subset α of variables such that P_2 can be derived from P_1 by complementing the variables in α.

[Example 1] We consider a pseudoproduct for realizing a function in Fig. 1. This function represents a pseudoproduct of degree 2 (it has 4=2^2 points). An initial solution of the function is an EXOR of minterms corresponding to the single points A, B, C, and D. These minterms are pseudoproducts of degree 0:

$$\bar{x}_0\bar{x}_1x_2x_3 \oplus \bar{x}_0x_1\bar{x}_2x_3 \oplus x_0\bar{x}_1x_2x_3 \oplus x_0x_1\bar{x}_2x_3 \qquad (1)$$

Next, we expand the minterms to new pseudoproducts. Since any pair of points is a pseudoproduct from Definition 1-2, both sets of the minterms {A, C} and {B, D} are pseudoproducts of degree 1. Then, we can represent the function by the following expression:

$$\bar{x}_1x_2x_3 \oplus x_1\bar{x}_2x_3 \qquad (2)$$

Factoring these two pseudoproducts, we find a pseudoproduct of degree 2:

$$(x_1 \oplus x_2)x_3 \qquad (3)$$

Note that the above pseudoproduct, which includes the minterms {A, B, C, D}, can be divided into two disjoint pseudoproducts {A, C} and {B, D}, and there exists a subset of variables $\alpha = \{x_1, x_2\}$ such that {B, D} can be derived from {A, C} by complementing the literals of the variables in α.

We now give the definition of an ESPP form.

[Definition 2] An EXOR-Sum-of-Pseudoproducts (ESPP) form is a logical form connected by the EXOR-sum of pseudoproducts.

[Property 1] An arbitrary n-variable function can be represented by an ESPP form.

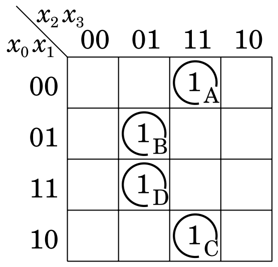


Fig. 1 Example of pseudoproduct.

This form generally provides an EXOR-AND-EXOR three-level network. It is known that, for almost all functions, three-level networks require fewer gates than two-level networks under the condition that the fan-in and fan-out of the gates are unlimited [21]. A pseudoproduct may be realized by an AND gate only in some networks. In this case, the ESPP form provides an AND-EXOR two-level network. Therefore, we may regard this form as a generalization of the EXOR-Sum-of-Products (ESOP) form. In this paper we assume the following conditions.

[Condition 1]

- For any i ($0 \le i \le n-1$, where n is the number of input variables), both variables x_i and $\bar{x}_i$ are available as inputs of the network.
- Each gate of an ESPP form has unlimited fan-in and fan-out.

A network realizing an ESPP form is called an ESPP network. EXOR gates in the first level of a network are called *EXOR factors*, and the outputs of these factors are connected to the inputs of AND gates in the second level. A pseudoproduct is a product of EXOR-factors. The pseudoproduct is connected to an EXOR gate in the third level. In some networks, EXOR-factors are directly connected to the EXOR gate in the third level.

As mentioned above, pseudoproducts are the generalization of conventional products. The conventional product corresponds to an AND gate and can be expanded only toward an adjacent product of the same degree on the Karnaugh map, while the pseudoproduct corresponds to an EXOR-AND or an AND form and can be expanded toward an pseudoproduct of the same degree from Definition 1. It is known that the utilization of EXOR gates often decreases the number of gates needed to realize practical logical networks. Furthermore, pseudoproduct-based logical forms are especially suitable to some functional classes, e.g., parity functions, symmetric functions, autosymmetric functions, and some well-known functional classes. Since most practical networks have the properties of these classes, ESPP forms are especially suitable for them.

In an ESPP form the minimum representation for the number of pseudoproducts does not always result in a network with the minimum number of gates. Here, we define the term *minimum* as follows.

[Definition 3] For a given function, an ESPP form is the minimum for the number of pseudoproducts (PP$_{min}$) if there exists no ESPP form representing the same function with fewer pseudoproducts. Similarly, an ESPP form is the minimum for literals (L$_{min}$) if there exists no ESPP form representing the same function with fewer literals.

There is no impartial method for comparing a two-level network with a three-level network. Therefore, we employ L$_{min}$ as the minimization goal in this paper.

[Example 2] We now consider an example which realizes a

Table 1 A function f.

x_0	x_1	x_2	x_3	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

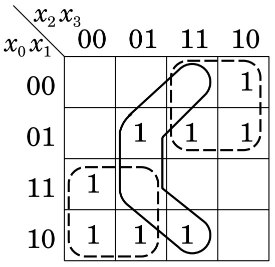


Fig. 2 Example of ESPP.

given function by an ESPP form. Table 1 shows a 4-variable function. We can express this function by an ESPP form:

$$f = (x_0 \oplus x_2) \oplus (x_1 \oplus x_2)x_3 \tag{4}$$

Figure 2 shows a map representing a minimum ESPP form on PP$_{min}$ and L$_{min}$ for the function. In this figure, the loop of the solid line and the loops of broken line correspond to expressions $(x_1 \oplus x_2)x_3$ and $(x_0 \oplus x_2)$, respectively. Each expression is a pseudoproduct of the function. Therefore, this form has 2 pseudoproducts and 5 literals. We can also realize this function by the conventional SOP (Eq. (5)) and SPP (Eq. (6)) forms:

$$\begin{aligned} f &= \bar{x}_0x_1x_3 + x_0\bar{x}_1x_3 + \bar{x}_0x_2\bar{x}_3 + x_0\bar{x}_2\bar{x}_3 & (5)\\ &= (x_0 \oplus x_1)x_3 + (x_0 \oplus x_2)\bar{x}_3 & (6)\end{aligned}$$

This SOP form has 4 products and 12 literals, and this SPP form has 2 pseudoproducts and 6 literals. The ESPP form has a smaller expression than the SOP and SPP forms.

The autosymmetric function [1], [2] is strongly related to pseudoproduct-based representations, and is defined as follows.

[Definition 4] A Boolean function f in $\{0,1\}^n$ is *closed under* $\boldsymbol{\alpha}$, with $\boldsymbol{\alpha} \in \{0,1\}^n$, if for each $\boldsymbol{w} \in \{0,1\}^n$, $\boldsymbol{w} \oplus \boldsymbol{\alpha} \in f$ if and only if $\boldsymbol{w} \in f$.

[Definition 5] A set $L_f = \{\boldsymbol{\alpha} : f \text{ is closed under } \boldsymbol{\alpha}\}$ is called a *linear space* of f. L_f has dimension $k = \log|L_f|$, where $|L_f|$ denotes the number of elements in the set L_f.

$x_0x_1 \backslash x_2x_3$	00	01	11	10
00	1	1		1
01	1		1	1
11	1	1		1
10	1		1	1

Fig. 3 Autosymmetric function.

[Definition 6] A function f is *k-autosymmetric*, or equivalently f has *autosymmetry degree k*, $0 \le k \le n$, if its linear space L_f has dimension k. If $k \ge 1$, f will be simply called *autosymmetric*.

[Example 3] We show a 4-variable autosymmetric function f in Fig. 3 as an example. Since the function f can be transferred to itself by complementing the variables x_0 and x_1, f is closed under 1100. Similarly, f is also closed under 0000, 1010 and 0110. Then, $L_f = \{0000, 0110, 1010, 1100\}$, and f is 2-autosymmetric.

3. Number of Pseudoproducts

In this section we discuss the number of pseudoproducts and the complexity of pseudoproducts for realizing some functional classes.

[Definition 7] Let M be the set of all minterms with n variables, and P and PP be the set of all possible products and pseudoproducts, respectively. In this paper the sets of products and pseudoproducts do not include the constant function 0; they do not include the constant function 1 (we define the number of literals as 0 in this case).

[Property 2] From Definition 7, the following properties hold: $M \subseteq P \subseteq PP$. The number of minterms M and pseudoproducts P is $|M| = 2^n$ and $|P| = 3^n$, respectively, where $|S|$ denotes the number of elements in the set S.

Since a pseudoproduct is the generalization of the conventional product, all of the products are included in the pseudoproducts. Therefore, the number of pseudoproducts is larger than the number of products. We derive the number of pseudoproducts of an n-variable functions ($2 \le n \le 6$). This result is in Table 2. Each row shows the number of pseudoproducts of degree m ($0 \le m \le 6$). The number of n-variable pseudoproducts of degree 1 (2 points) is $\binom{2^n}{2} = \frac{2^n(2^n-1)}{2} = 2^{n-1} \cdot (2^n - 1)$. In SOPs, the number of products of degree 1 is $2^{n-1} \cdot n$. The numbers of pseudoproducts with degrees greater than 1 are based on the combination of pseudoproducts of degree 1. Therefore, the number of pseudoproducts of an n-variable function grows exponentially with the number of inputs n.

Since the number of pseudoproducts is larger than the number of products, the pseudoproduct-based expressions may represent any n-variable functions with much smaller products than the product-based expressions.

We derive the number of pseudoproducts for n-variable functions, including parity functions and symmetric functions. Table 3 shows the number of products for realizing these functional classes by SOP, ESOP, SPP, and ESPP. In this table, n is the number of inputs, $2r = n$, and $3q = n$. The numbers of products in SOPs and ESOPs are shown in [21] and the number of products in SPPs are derived from the definition of SPPs and the papers [1], [2], [11]. Here, we prove the ESPPs' case:

(proof) The number of pseudoproducts of ESPPs in Table 3 is obtained as follows.

Parity function: since an EXOR factor is regarded as a single pseudoproduct, the number of the parity function is 1.

$x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n$:

$$\begin{aligned} & x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n \\ &= \overline{(1 \oplus x_1x_2)(1 \oplus x_3x_4)\cdots(1 \oplus x_{n-1}x_n)} \qquad (7) \\ &= \overline{1 \oplus x_1x_2 \oplus \cdots \oplus x_{n-1}x_n \oplus x_1x_2x_3x_4 \oplus \cdots} \\ &\quad \overline{\oplus x_{n-3}x_{n-2}x_{n-1}x_n \oplus \cdots \oplus x_1x_2\ldots x_{n-1}x_n} \\ &= x_1x_2 \oplus \cdots \oplus x_{n-1}x_n \oplus x_1x_2x_3x_4 \oplus \cdots \\ &\quad \oplus x_{n-3}x_{n-2}x_{n-1}x_n \oplus \cdots \oplus x_1x_2\ldots x_{n-1}x_n \end{aligned}$$

The number of the EXOR-sum terms $1 \oplus x_ix_{i+1}$ in Eq. (7) is r, where $r = n/2$ and $i = 1, 3, 5, \ldots, n-1$. In this proof the input number n is necessarily an even number. The number of products of the expansion of Eq. (7) is 2^r. Therefore, the number of pseudoproducts is $2^r - 1$.

$x_1x_2 \oplus x_3x_4 \oplus \cdots \oplus x_{n-1}x_n$: this function is an ESOP form. Each of the products is a pseudoproduct, and therefore the number of pseudoproducts is r ($r = n/2$).

Symmetric function: we find that the number of pseudoproducts for realizing a 3-variable symmetric function is 2 at most. Let us suppose that the number of pseudoproducts for realizing any $(n-3)$-variable symmetric function ($q = n/3$) is $2^{2(q-1)-1}$ at most when $q \geq 2$. Any n-variable symmetric function can be expanded as follows:

$$\begin{aligned} & f(x_1, x_2, \ldots, x_n) \\ &\quad = \bar{x}_1\bar{x}_2\bar{x}_3 f_{000} \oplus \bar{x}_1\bar{x}_2x_3 f_{001} \oplus \cdots \oplus x_1x_2x_3 f_{111} \end{aligned}$$

where $f_{a_1a_2a_3} = f(a_1, a_2, a_3, x_4, \ldots, x_n)$ ($a_i \in B$). Since the function f is symmetric, $f_{001} = f_{010} = f_{100}$ and $f_{011} = f_{101} = f_{110}$:

$$\begin{aligned} f &= \bar{x}_1\bar{x}_2\bar{x}_3 f_{000} \oplus (\bar{x}_1\bar{x}_2x_3 \oplus \bar{x}_1x_2\bar{x}_3 \oplus x_1\bar{x}_2\bar{x}_3) f_{001} \\ &\quad \oplus (\bar{x}_1x_2x_3 \oplus x_1\bar{x}_2x_3 \oplus x_1x_2\bar{x}_3) f_{011} \oplus x_1x_2x_3 f_{111} \\ &= \bar{x}_1\bar{x}_2\bar{x}_3(f_{000} \oplus f_{011}) \oplus (x_1 \oplus x_2 \oplus x_3) f_{001} \\ &\quad \oplus (\bar{x}_1 \oplus \bar{x}_2 \oplus \bar{x}_3) f_{011} \oplus x_1x_2x_3(f_{111} \oplus f_{001}) \end{aligned}$$

The subfunctions f_{000}, f_{001}, f_{011}, and f_{111} are composed of $2^{2(q-1)-1}$ pseudoproducts at most. The functions $f_{000} \oplus f_{011}$ and $f_{111} \oplus f_{001}$ are also symmetric functions [21]. Then, the number of pseudoproducts of the function f is $4 \cdot 2^{2(q-1)-1} = 2^{2q-1}$.

n-bit adder: Luccio et al. [10] show the proof of the number of pseudoproducts for realizing an n-bit adder in an EXOR-AND-OR three-level form. Since the pseudoproducts for realizing the n-bit adder are disjoint, we can also prove the number in an ESPP form by replacing the OR gate in the output with an EXOR gate. □

Most of the ESPP forms can represent any n-variable function with fewer products than can SOP and ESOP forms. Furthermore, practical networks can be efficiently realized by ESPP forms. ESOP and ESPP forms realizing a parity function have the same representation. Since parity function is regarded as a single pseudoproduct, the complexity of ESOP differs from ESPP.

4. Minimization of ESPPs

For the exact minimization of ESPP forms, we have no algorithm except for the exhaustive method. For near-minimum ESOP forms, most algorithms use iterative improvement methods. Because this method has shorter time cost and smaller memory space than the former, we use the iterative improvement method for the minimization of ESPP forms. Most iterative algorithms use the Reed-Muller Expansion or minterms of SOP for their initial solutions. However, these solutions require excessive memory space when the number of the inputs is large. Therefore, the initial solutions are disjoint SOP in our algorithm.

Table 2 Number of all n-variable pseudoproducts.

degree $m \setminus n$	2	3	4	5	6
0	4	8	16	32	64
1	6	28	120	496	2016
2	1	14	140	1240	10416
3	–	1	30	620	11160
4	–	–	1	62	2604
5	–	–	–	1	126
6	–	–	–	–	1

Table 3 Complexity of some functional classes.

forms	SOP	ESOP	SPP	**ESPP**
parity function	2^{n-1}	n	1	1
$x_1x_2 + \cdots + x_{n-1}x_n$	r	$2^r - 1$	r	$2^r - 1$
$x_1x_2 \oplus \cdots \oplus x_{n-1}x_n$	2^r	r	$2^r - 1$	r
symmetric function	2^{n-1}	$2 \cdot 3^{r-1}$	3^{r-1}	2^{2q-1}
n-bit adder	$6 \cdot 2^n - 4n - 5$	$2^{n+1} - 1$	$n^2/2 + n/2$	$n^2/2 + n/2$

$2r = n, 3q = n$

[Definition 8] An SOP in which each pair of products is disjoint is called a *disjoint Sum-of-Products expression*.

In a disjoint SOP, the OR operators can be replaced with EXOR operators without changing the function.

This algorithm uses the following rules for expanding, reducing and reconstructing pseudoproducts.

[Rule 1]

1. EXPAND_PP: If there exists a subset α of variables such that P_1 can be derived from P_2 by complementing the variables in α, $P_1 \oplus P_2$ is a set of minterms realizing a new pseudoproduct P (Definition 1-2).
2. EXPAND: $P_1 \cdot P_2 \oplus \bar{P}_2 \rightarrow 1 \oplus \bar{P}_1 \cdot P_2$.
3. MERGE: $P_1 \oplus P_1 \rightarrow 0$, $P_1 \oplus \bar{P}_1 \rightarrow 1$, $P_1 \oplus 1 \rightarrow \bar{P}_1$, $\bar{P}_1 \oplus 1 \rightarrow P_1$.
4. RESHAPE: $P_1 \cdot P_2 \oplus \bar{P}_2 \rightarrow \bar{P}_1 \cdot \bar{P}_2 \oplus P_1$.
5. DESCENDANT: A pseudoproduct P with degrees greater than 1 is decomposed into two pseudoproducts, P_1 and P_2, when one can be generated from the other by complementing the variables (the reverse operation of EXPAND_PP).
6. REDUCE: $1 \oplus \bar{P}_1 \cdot P_2 \rightarrow P_1 \cdot P_2 \oplus \bar{P}_2$.

Figure 4 shows examples of the above minimization rules. Among these rules, EXPAND_PP and MERGE reduce the number of pseudoproducts in an ESPP form. The other rules do not reduce the number of pseudoproducts, but modify the shape of the pseudoproducts. Note that the ESOP minimization algorithm relies only on MERGE to re-

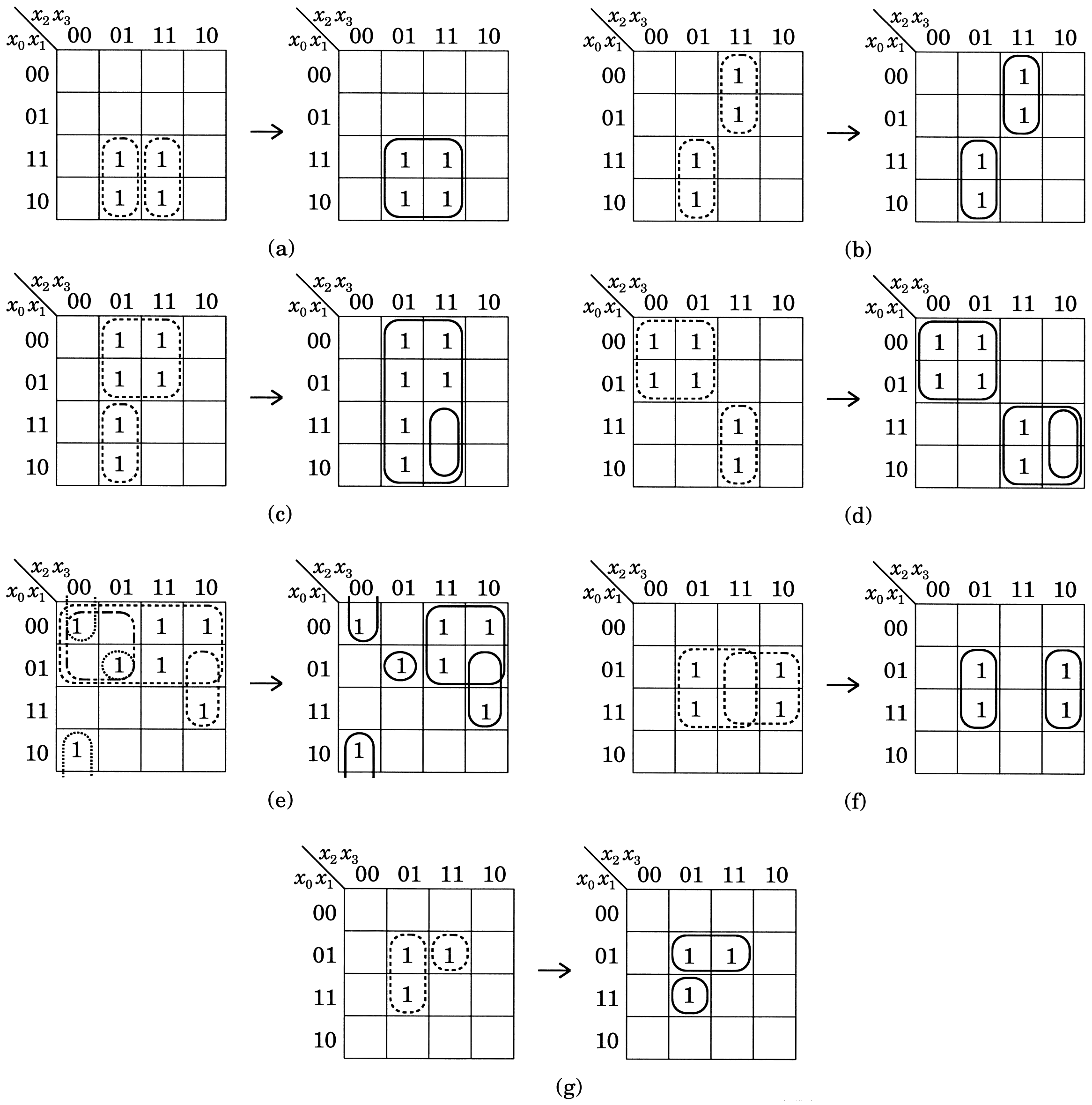


Fig. 4 Examples of minimization rules. (a) (b) EXPAND_PP, (c) (d) EXPAND, (e) (f) MERGE, (g) RESHAPE.

duce the number of the products. EXPAND_PP is the expansion operation defined by Definition 1, and provides more possibilities of expansion than conventional expansion operations. EXPAND_PP and EXPAND are effective for reducing the number of literals. RESHAPE is used to modify the shape of pseudoproducts. DESCENDANT and REDUCE are the reverse operation of EXPAND_PP and EXPAND, respectively. DESCENDANT makes disjoint pseudoproducts P_1 and P_2 of degree $m-1$ from a pseudoproduct P of degree m. The quality of the solution is sensitive to the order of the rules applied. The following algorithm uses a simple ordering of the rules. For a multiple-output function, we decompose it into single-output functions and treat each function independently.

[Algorithm 1]

```
1. For a given function $f$, convert it into a disjoint SOP
   form and make it the initial solution $F$. Let $F_{min}$ be the
   current minimum solution.
2. Choose the goal of the minimization between $PP_{min}$
   (the minimum for pseudoproducts) and $L_{min}$ (the minimum
   for literals). If $PP_{min}$ ($L_{min}$) is chosen, let $|F|$ be
   the number of pseudoproducts (literals) of the solution
   $F$. $F_{min} = F$.
3. (rearrange routine): For each pair of pseudoproducts of
   $F$, Rule 1 is applied in the following orders, according
   to which order is applicable:
     a. MERGE $\rightarrow$ EXPAND $\rightarrow$ MERGE
     b. EXPAND_PP $\rightarrow$ EXPAND $\rightarrow$ MERGE
     c. RESHAPE $\rightarrow$ EXPAND $\rightarrow$ MERGE
4. If $|F_{min}| > |F|$, then $F_{min} = F$.
5. If EXPAND or EXPAND_PP was applied in step 3, go
   to step 3 again.
6. Apply REDUCE and DESCENDANT in this order,
   and repeat step 3.
7. If $|F_{min}| > |F|$, then $F_{min} = F$, and go to step 3.
8. Return $F_{min}$.
```

In step 3 of Algorithm 1, "EXPAND → MERGE" is a basic process in order to decrease the numbers of literals and pseudoproducts. EXPAND can expand a pseudoproduct only into an adjacent pseudoproduct. Accordingly, EXPAND_PP is applied for expanding pseudoproducts more flexibly in step 3-b. Since applying the expanding rules only generates the biased local solutions, RESHAPE is applied in step 3-c.

5. Experimental Results

Our algorithm in the previous section is implemented using C on a PentiumIII 550 MHz (22.3 SPECint95) computer (OS: Solaris 8). We chose the minimum for literals L_{min} for minimization in this experiments. Symmetric functions can be found in many practical networks. Therefore, we obtained the average numbers of the products and literals of each representation (SOP, ESOP, SPP, and ESPP forms) for realizing the symmetric functions of two to six variables (Table 4). The SPP form is an EXOR-AND-OR three-level form proposed by Luccio et al. [10]. The SOP, ESOP, and ESPP forms in this paper were obtained by our algorithms. In Table 4, #P, #L and #PP denote the number of products, literals, and pseudoproducts, respectively. Since the concept of products differs from pseudoproducts, we can not compare them simply. However, the numbers of the literals of the ESPP forms are the smallest of the four, except in the case of $n = 2$. These results show that an ESPP form is effective for realizing functions.

Next, we give the number of products and literals on MCNC benchmark networks [13]. In this experiment, the multiple-output functions were decomposed into single-output functions and each function was minimized independently by Algorithm 1 (Table 5). In this table, "network" denotes the names of the benchmark networks and the numbers in parentheses are the output numbers. #I denotes the number of inputs of the networks. Most MCNC benchmark networks have autosymmetries, and all of the networks in this experiment have autosymmetric functions. This table shows that most of the ESPP forms have the smallest representations of four forms. Note that the SPP forms have better representation than the SOP and ESOP forms, and some of these have smaller literals than the ESPP forms. All the outputs of the networks rd53 and rd73 represent symmetric functions.

We also compared the CMOS and FPGA costs of our ESPP forms with the conventional SOP forms on MCNC benchmark networks (Table 6). In many technologies, EXOR and AND/OR gates have different costs. Bernasconi et al. [2] estimate that the cost of the 2-input EXOR gate is 4, while the cost of the 2-input OR and AND gates is 2. This is in proportion to also to the number of transistors used for the CMOS technology mapping (i.e., 4 transistors for AND/OR gates and 8 transistors for the EXOR gate). In this paper, we regard a k-input EXOR gate as the composition of $k-1$, 2-input EXOR gates for the associative property of the EXOR gate. Therefore, we can use the CMOS cost function μ_C, where a k-input EXOR gate costs $4(k-1)$, and a k-input AND/OR costs k. Another cost function is for the field programmable gate arrays (FPGAs). We also use the FPGA cost function μ_F, where k-input EXOR gates and k-input AND/OR gates have the same cost k. These measures of the costs are used for SPP forms in [2]. In Table 6, μ'_C is the CMOS cost in SOP networks, and its FPGA cost is $\mu'_F = \mu'_C$. $\#O$ denotes the number of outputs of the networks. In our experiment, the minimization of multiple output networks has been carried out individually for each output. This table demonstrates that ESPPs require lower costs than SOPs in many cases of both CMOS and FPGA technology mapping. Especially, networks which include many autosymmetric and/or symmetric functions tend to be smaller in ESPPs.

Autosymmetric functions are strongly related to SPP and ESPP forms. It is important for pseudoproduct-based representations to detect autosymmetric functions. We ob-

Table 4 Average number of products and literals for realizing symmetric functions.

n	SOP		ESOP		SPP		ESPP	
	#P	#L	#P	#L	#PP	#L	#PP	#L
2	1.71	2.57	1.37	3.12	1.12	1.62	1.16	2.00
3	2.38	4.23	2.25	4.12	1.62	3.68	1.54	3.62
4	3.83	10.86	4.03	9.31	2.40	7.84	2.37	7.07
5	10.45	42.29	6.57	20.37	3.59	15.60	3.13	14.38
6	20.41	101.27	10.85	40.76	5.41	29.93	5.02	27.98

Table 5 Number of products and literals on MCNC benchmark networks.

network	#I	SOP		ESOP		SPP		ESPP	
		#P	#L	#P	#L	#PP	#L	#PP	#L
5xp1 (1)	7	27	136	17	40	5	23	5	22
5xp1 (2)	7	26	141	30	63	5	30	6	34
5xp1 (3)	7	21	113	18	56	6	36	5	28
5xp1 (4)	7	31	162	9	42	5	30	4	26
con1 (1)	7	4	11	9	29	4	11	5	15
con1 (2)	7	5	12	9	24	4	22	4	14
misex1 (1)	8	2	7	8	18	1	5	1	5
misex1 (2)	8	5	18	9	30	4	15	3	11
misex1 (3)	8	5	21	14	47	5	21	4	20
rd53 (1)	5	12	57	10	20	3	12	3	12
rd53 (2)	5	11	44	5	5	1	5	1	5
rd53 (3)	5	5	20	5	20	3	14	3	14
rd73 (1)	7	35	140	18	42	13	76	9	24
rd73 (2)	7	43	258	33	154	10	82	11	88
rd73 (3)	7	64	448	7	7	1	7	1	7
squar5 (1)	5	9	21	3	11	2	6	3	11
squar5 (2)	5	11	25	5	15	3	9	2	8
squar5 (3)	5	11	25	5	16	4	13	4	15
squar5 (4)	5	14	40	7	20	4	16	3	12
sqrt8 (1)	8	26	113	23	90	10	61	9	57
sqrt8 (2)	8	10	33	11	34	6	25	7	28
sqrt8 (3)	8	3	6	5	9	3	6	3	6
sqrt8 (4)	8	2	2	2	2	1	2	1	2
xor5 (1)	5	16	80	5	5	1	5	1	5

Table 6 CMOS and FPGA costs on MCNC benchmark networks.

network	#I	#O	ESPP		SOP	ESPP/SOP	
			μ_C	μ_F	μ'_C	μ_C/μ'_C	μ_F/μ'_F
bw	5	28	380	239	436	0.87	0.55
clip	9	5	408	324	748	0.55	0.43
con1	7	2	57	38	32	1.78	1.18
inc	7	9	207	121	223	0.93	0.54
misex1	8	7	172	125	154	1.12	0.81
rd53	5	3	72	46	171	0.42	0.30
rd73	7	3	214	122	833	0.26	0.15
rd84	8	4	622	306	2029	0.31	0.15
sao2	10	4	297	219	431	0.69	0.51
Z5xp1	7	10	82	56	341	0.24	0.16
Z9sym	9	1	203	135	588	0.35	0.23
z4	7	4	87	57	311	0.28	0.18

Table 7 Average number of literals for realizing autosymmetric functions.

n	#func.	SOP	ESOP	SPP	ESPP
3 (autosymmetric)	72	3.819	3.377	2.479	2.330
3 (non-autosymmetric)	184	5.087	4.923	4.217	4.211
4 (autosymmetric)	3072	10.065	8.716	5.323	4.015
4 (non-autosymmetric)	62464	11.818	10.992	8.322	8.621

tained the number of autosymmetries in general functions by using our detection method [9]. The numbers of 3 and 4-variable autosymmetric functions was 72 and 3072, respectively. The ratio of autosymmetric functions to all functions is small, although we found that many practical networks have autosymmetries. Table 7 shows that the average number of literals for realizing autosymmetric functions and non-autosymmetric functions. The difference between the autosymmetric and non-autosymmetric functions in the ESPPs is larger than in the other forms, and the number of literals of the autosymmetric function in the ESPP form is the smallest of the four. Therefore, autosymmetric functions have a great influence on ESPP forms. ESPP forms are especially suitable for practical networks which have this property.

6. Conclusions

In this paper, we presented a new EXOR-based three-level representation, i.e., an ESPP form and a algorithm for ESPP minimization. The minimization algorithm is heuristic based on the iterative improvement. The algorithm contains a complex routine using some expansion and reconstructing rules, but the time and memory costs are small. In our experiments, ESPP forms of symmetric functions and MCNC benchmark networks were generated, and we compared the number of products and literals in the ESPP forms with those in the three other forms (SOP, ESOP, and SPP). In most functions and benchmarks, the number of literals in the ESPP forms are the smallest of the four. In addition, the number of literals on autosymmetric functions was obtained. These experimental results show that the ESPP form has a small representation on practical networks. Our future investigations will concentrate on faster and more effective procedures for minimizing ESPP forms, and a minimization algorithm for multiple-output networks.

References

[1] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Fast three-level logic minimization based on autosymmetry," DAC 2002 Proc., pp.425–430, June 2002.

[2] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Three-level logic minimization based on function regularities," IEEE Trans. Comput., vol.22, no.8, pp.1005–1016, Aug. 2003.

[3] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Kluwer, Norwell, MA, 1984.

[4] V. Ciriani, "Logic minimization using exclusive OR gates," DAC 2001 Proc., pp.115–120, June 2001.

[5] R. Drechsler, H. Hengster, H. Schäfer, J. Hertmann, and B. Becker, "Testability of 2-level AND/EXOR circuits," EDAC'97 Proc., pp.548–553, March 1997.

[6] S.L. Hurst, D.M. Miller, and J.C. Muzio, Spectral techniques in digital logic, Academic Press, 1985.

[7] R. Ishikawa, T. Igarashi, T. Hirayama, and K. Shimizu, "Logic representation with EXOR operations and its testability," ICFS 2000 Proc., pp.11-1–11-6, March 2002.

[8] R. Ishikawa, T. Igarashi, T. Hirayama, and K. Shimizu, "Pseudocube-based expressions to enhance testability," IEEE APCCAS 2002 Proc., pp.305–310, Dec. 2002.

[9] R. Ishikawa, G. Koda, and K. Shimizu, "Detection of autosymmetry in logic functions using spectrum technique," IEICE Trans. Inf. & Syst., vol.E86-D, no.12, pp.2691–2697, Dec. 2003.

[10] F. Luccio and L. Pagli, "Normal matrices, pseudo-cubes and pseudo-products," Congressus Numerantium, vol.127, pp.33–56, 1997.

[11] F. Luccio and L. Pagli, "On a new Boolean function with applications," IEEE Trans. Comput., vol.48, no.3, pp.296–310, 1999.

[12] P.K. Lui and J.C. Muzio, "Boolean matrix transforms for the minimization of modulo-2 canonical expansions," IEEE Trans. Comput., vol.41, no.3, pp.342–347, March 1992.

[13] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Distributed as part of the MCNC International Workshop on Logic Synthesis '93 Benchmark Distribution, May 1993.

[14] D.K. Pradhan, "Universal test sets for multiple fault detection in AND-EXOR arrays," IEEE Trans. Comput., vol.C-27, no.2, pp.181–187, Feb. 1978.

[15] S.M. Reddy, "Easily testable realization for logic functions," IEEE Trans. Comput., vol.C-21, no.11, pp.1183–1188, Nov. 1972.

[16] K.K. Saluja and S.M. Reddy, "Fault detecting test sets for Reed-Muller canonic networks," IEEE Trans. Comput., vol.C-24, no.1, pp.995–998, Oct. 1975.

[17] T. Sasao, "Logic synthesis with EXOR gates," in Logic Synthesis and Optimization, ed. T. Sasao, Kluwer Academic Publishers, 1993.

[18] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued input two-valued output functions," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.12, no.5, pp.621–632, May 1993.

[19] T. Sasao, "Representations of logic functions using EXOR operators," in Representations of Discrete Functions, ed. T. Sasao and M. Fujita, Kluwer Academic Publishers, 1996.

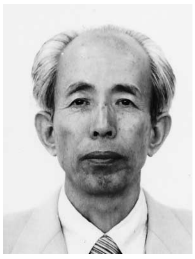
[20] T. Sasao, "Easily testable realizations for generalized Reed-Muller expansions," IEEE Trans. Comput., vol.46, no.6, pp.709–716, June 1997.

[21] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.

Ryoji Ishikawa received his B.E., and M.E. degrees in computer science from Gunma University, Kiryu, Japan, in 1998 and 2001, respectively. He is currently a student of doctor course in the Department of Computer Science, Gunma University. His research interests include logic synthesis and design for testability.

Takashi Hirayama received his B.E., M.E., and Ph.D. degrees in computer science from Gunma University, Kiryu, Japan, in 1994, 1996, and 1999, respectively. From 1999 to 2001 he was a research assistant in the Department of Electrical and Electronics Engineering, Ashikaga Institute of Technology. He is currently a lecturer in the Department of Computer and Information Sciences, Faculty of Engineering, Iwate University. His research interests include high level and logic synthesis and design for testability.

Goro Koda received his B.E., M.E. degrees in electronic engineering from Gunma University, Kiryu, Japan, in 1971, and 1973, respectively. In 1973 he joined the NEC corporation. Since 1975 he has been a research assistant at the Department of Computer Science, Gunma University. His research interests include CAD for logic networks.



Kensuke Shimizu received his B.E., M.E., and the Ph.D. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1962, 1964, and 1967, respectively. He was a Lecturer from 1967 to 1968 and an Associate Professor from 1968 to 1976 in the Department of Electronic Engineering, Faculty of Engineering, Gunma University. Since 1976 he has been a Professor in the Department of Computer Science, Faculty of Engineering, Gunma University, engaged in research and education in logic circuits, switch theory, and expert systems.