Minimization of Exclusive Sum-of-Products Expressions for Multiple-Valued Input, Incompletely Specified Functions

Ning Song and Marek A. Perkowski, *Member, IEEE*

Abstract—**This paper presents a new operation (exorlink) and an algorithm to minimize Exclusive-OR Sum-of-Products expressions (ESOP's) for multiple valued input, two valued output, incompletely specified functions. Exorlink is a more powerful operation than any other existing one for this problem. Evaluation on benchmark functions is given and it proves the superiority of the program to those known from the literature.**

I. Introduction

IN RECENT years, the interest in the design of logic circuits which use Exclusive-OR (EXOR) gates has increased. Functions realized by such circuits can have fewer gates, fewer connections, and take up less area when realized in VLSI circuits. They can have also fewer cells when realized in Field Programmable Gate Arrays (FPGA's). Circuits with high EXOR components are also easily testable [6], [12]. Circuits of this type have applications in self-testing schemes, linear machines, arithmetic and communication circuits, encrypting schemes, coding schemes for error control and synchronization, sequence generation for process identification, system testing, etc. It was demonstrated in [17] that on average, the AND-EXOR PLA's require fewer product terms than the AND-OR PLA's. Both AND-OR PLA's and AND-EXOR PLA's can have input decoders, which lead to the application of multiple-valued logic as a mathematical technique for the minimization of such binary PLA's. The studies [15], [18] prove that both types of PLA's with input decoders require smaller area than the PLA's without input decoders. AND-EXOR PLA's realize Exclusive-OR Sum-of-Product expressions (ESOP's). AND-EXOR PLA's with n-bit $(n > 1)$ input decoders correspond to multiple-valued input ESOP's (MIESOP's). In this paper, we focus on the minimization of MIESOP's. Since a binary valued input is a special case of a multiple-valued input, MIESOP's are more general than binary valued input ESOP's. Minimization of ESOP's is a more difficult problem than that of Sum-of-Product expressions (SOP's) minimization. So far, exact solutions for ESOP's can be found only practically for functions with five, or sometimes a few more variables [9], [11], [19]. Therefore, the interest is mainly in approximate solutions. Two approaches to generate suboptimal solutions can be found in the literature. One approach is to minimize some canonical subfamilies of ESOP's (exact or approximate solutions). Another approach is to minimize ESOP's using heuristic algorithms. Efficient programs for subfamilies of ESOP's were given in [1], [4], [7], [21]. Heuristic ESOP minimization programs have been presented in [2], [5], [8], [10], [13], [16], [18], [20]. In these programs, two general methods have been used. One method is to base the minimization on the coefficients of canonical generalizations to Reed–Muller forms [1], [13], [24]. Another method is to perform a set of cube operations iteratively on ESOP's (starting from minterms, disjoint cubes, ESOP's, Reed–Muller forms, or other representations). Fleisher *et al.* [5] presented an algorithm which starts from positive Reed–Muller forms and performs three cube operations iteratively. Helliwell and Perkowski [8] introduced new cube operations, "*primary xlink*" and "*secondary xlink*," and presented an algorithm based on these operations. The algorithm from [8] was next improved in [10], and also extended for the case of logic with multiple-valued inputs. A new cube operation, "*unlink*," has also been added. The unlink operation was efficiently implemented in [20]. A few more cube operations were also included in an independent realization by Sasao [16], [18]. So far, this approach has achieved better results than other methods [18]. The literature clearly demonstrates that the more powerful the cube operations are, the better the results [18].

Some limitations exist in current programs. Hermes [20] is used for binary input functions only. EXMIN2 [18]] does not handle incompletely specified functions. In this paper, we present a new cube operation, *exorlink*, for the minimization of MIESOP's. This single operation contains all the cube operations presented in [5], [8], [10], [16], [18]. Based on our new cube operation, a new algorithm is also discussed in this paper. This new algorithm is more efficient than the existing ones [10], [18]. Our program based on the exorlink operation has the following advantages: it is applicable to both the binary input functions and the multiple-valued input functions; each input variable can have an arbitrary number of logic values; the function can be completely specified or incompletely specified; and the output can be single output or multiple output. To evaluate our program, we collected all the published benchmarks that we were able to find. Our experimental results show that for both binary input functions and multiple input functions, for both single output functions

Manuscript received May 3, 1993; revised March 3, 1994, November 15, 1994, and September 15, 1995. This paper was recommended by Associate Editor L. Trevillyan.

N. Song is with the Lattice Semiconductor Corporation, Milpitas, CA 95035 USA.

M. A. Perkowski is with the Department of Electrical Engineering, Portland State University, Portland, OR 97207 USA.

Publisher Item Identifier S 0278-0070(96)03478-1.

TABLE I
BINARY INPUT EXAMPLES

	inputs	outputs	EXMIN-2 C_T	EXMIN-2 C_L	EXMIN-2 Time (1)	EXORCISM-MV-2 C_T	EXORCISM-MV-2 C_L	EXORCISM-MV-2 Time (2)
9sym	9	1	53	433	25	51	425	39
t481	16	1	13	53	677	13	53	615
xor5	5	1				5	10	0.2
Adr4	8	5	31	162	13.5	31	150	15.5
Log8	8	8	99	690		89	675	261
Mlp4	8	8	63	395	42.9	61	389	89.9
Nrm4	8	5	71	536		67	503	157
Rdm8	8	8	32	161		31	157	15.4
Rot8	8	5	37	257		35	257	32.7
Sqr8	8	16	112	747		108	703	1468
Wgt8	8	4	59	330		55	307	122
5xp1	7	10	34	186	13	33	178	13.6
add6	12	7	127	872	430	127	883	239
b12	15	9	28	164	4	28	164	5.8
bw	5	28				22	297	5.9
clip	9	5	68	517	55	65	490	66.9
con1	7	2				9	37	0.4
f51m	8	8	32	161	10	31	155	10.8
inc	7	9				26	179	12.2
misex1	8	7				12	89	1.4
misex2	25	8				27	210	21.4
rd53	5	3	15	60	2	14	57	1.3
rd73	7	4	42	221	20	38	191	24.6
rd84	8	4	59	330	45	57	317	168.2
sao2	10	4	29	310	8	28	286	10.9
squar5	5	8				19	82	2.9
vg2	25	8	184	1992	163	184	1991	87.2
seq	41	35	259	5305	2797	249	4933	2471.8

(1) cpu seconds on SPARC Station 1+.
(2) cpu seconds on SPARC Station 1.

TABLE II
BINARY INPUT FUNCTIONS

	HEALEX C_T	ND C_T	HERMES C_T	EXORCISM-MV-2 C_T	EXORCISM-MV-2 C_L	EXORCISM-MV-2 Time (1)
9sym			95	51	425	39
ADR2	8			7	23	0.1
ADR4	31	31	34	31	150	15.5
MLP3	19		23	18	96	2.2
MLP4	68	61	92	61	389	89.9
SQR6	36	35	39	33	153	17.9
NRM4	73	73		67	503	157
RDM8	32	31		31	157	15.4
ROT8	35	35		35	257	32.7
WGT8	54	54		55	307	122

(1) cpu seconds on SPARC Station 1.

and multiple output functions, our program is superior to all the existing programs (see Tables I, II, and III for details). We also tested out programs extensively on incompletely specified functions and obtained very good results. However, there are not any published results with which we can compare our results.

Section II presents our new cube operation, exorlink. Contrary to the operations in EXORCISM [8], [10], Hermes [20], and EXMIN-2 [18], which can be applied under satisfaction of certain conditions only, exorlink can be applied to any two cubes with arbitrary distance, as it will be shown in Section II. The procedure of the exorlink operation is discussed, different examples are presented, and comparisons with operations of EXORCISM and EXMIN-2 are given.

TABLE III
MULTIPLE-VALUED INPUT FUNCTIONS

	inputs	outputs	EXMIN-2 C_T	EXMIN-2 C_L	EXORCISM-MV-2 C_T	EXORCISM-MV-2 C_L	EXORCISM-MV-2 Time (1)
Adr4	4	5	11	60	11	60	1.8
Log8	4	8	92	944	78	789	518
Mlp4	4	8	50	412	49	399	45.1
Nrm4	4	5	52	521	52	497	100.6
Rdm8	4	8	26	181	26	163	8.6
Rot8	4	5	26	242	26	245	14.2
Sqr8	4	16	108	1078	98	907	351
Wgt8	4	4	25	159	23	160	12.6

(1) cpu seconds on SPARC Station 1.

In Section III, our new algorithm used in EXORCISM-MV-2 is discussed. The major advantage of this algorithm is that it gives priority to those distance two operations which will directly reduce the number of cubes in the array. In this way, our program can achieve better results in shorter time as compared to the former algorithms. The new algorithms that handle multiple output functions and incompletely specified functions are also discussed in this section. They are incorporated into EXORCISM-MV-2. Section IV shows the experimental results. The conclusion is given in Section V.

II. THE MULTIPLE-VALUED EXORLINK OPERATION

In this section, we first give some definitions, and then the cost function is discussed. After introducing some basic properties of multiple-valued functions, our new cube operation, *exorlink* is presented. The remainder of the section discusses particular, special cases of this operation, and illustrates them with examples.

A. Definitions

Definition 1: A multiple-valued input, two-valued output, incompletely specified switching function f (also called *multiple-valued function,*) is a mapping $f(X_1, X_2, \cdots, X_n)$: $P_1 \times P_2 \times \cdots P_n \rightarrow B$, where X_i is a *multiple-valued variable*, $P_i = \{0, 1, \cdots, p_i - 1\}$ is a *set of admissible values* that this variable may assume, and $B = \{0, 1, x\}$ (x denotes a *don't care value*).

Definition 2: For any subset $S_i \subseteq P_i$, $X_i^{S_i}$ is a *literal* of X_i representing the function such that

$$X_i^{S_i} = \begin{cases} 1 & \text{if } X_i \in S_i \\ 0 & \text{if } X_i \notin S_i. \end{cases}$$

Definition 3: A product of literals, $X_1^{S_1} X_2^{S_2} \cdots X_n^{S_n}$, is referred to as a *product term* (also called *term* or *product*). A *minterm* is a product term that there exists only one value in each S_i for $i = 1, 2, \cdots, n$.

Definition 4: The EXOR of two products is assigned the value 1 if and only if the two products have different values. An EXOR of products is called an ESOP. It is also called a MIESOP if one wants to emphasize that the input variables are multiple valued.

Example 1: In 4-valued logic, given three terms $T_1 = X^{\{1,2\}} Y^{\{2,3\}}$, $T_2 = X^{\{2,3\}} Y^{\{1,2\}}$, and $T_3 = X^{\{0,1\}} Y^{\{1,3\}}$. $T_1 \oplus T_2 \oplus T_3$ is a MIESOP. We can also call it ESOP.

In cube notation [23], a term is represented by a cube, and each literal in the term is represented by a vector

$$c_1^0 c_1^1 \cdots c_1^{(p_1-1)} - c_2^0 c_2^1 \cdots c_2^{(p_2-1)} - \cdots - c_n^0 c_n^1 \cdots c_n^{(p_n-1)}$$

where

$$c_i^j = \begin{cases} 1 & \text{if } j \in S_i \\ 0 & \text{if } j \notin S_i. \end{cases}$$

For example, $X^{\{0\}}$ is denoted by $100 \cdots 000$, $X^{\{1\}}$ is denoted by $010 \cdots 000$, $X^{\{0,2\}}$ is denoted by $101 \cdots 000$, and $X^{\{0,1,\cdots,n\}}$ is denoted by $111 \cdots 111$, which represents the Boolean universe. A cube is a null cube if one or more variables contain all zeros.

Example 2: In Example 1, the ESOP can be written in cube notation as follows

$$[0110 - 0011] \oplus [0011 - 0110] \oplus [1100 - 0101].$$

Definition 5: The *distance of two terms* is the number of variables for which the corresponding literals have different sets of values.

Example 3: Given three terms, $T_1 = X^{\{0\}}Y^{\{1\}}$, $T_2 = X^{\{1\}}Y^{\{0,2\}}$, and $T_3 = X^{\{1\}}Y^{\{0,1\}}$, the distance of T_1 and T_2 is 2, because two literals have different sets of values

$$\text{for } X\text{: } \{0\} \neq \{1\},$$
$$\text{for } Y\text{: } \{1\} \neq \{0,2\}.$$

The distance of T_2 and T_3 is 1, because only one literal has different sets of values

$$\text{for } X\text{: } \{1\} = \{1\},$$
$$\text{for } Y\text{: } \{0,2\} \neq \{0,1\}.$$

We write *distance*$(T_i, T_j) = d$ to indicate that the distance of two terms T_i and T_j is d.

B. The Cost Function

The objective of logic minimization is to find a realization that reduces certain cost function. Our primary goal of ESOP synthesis is to *minimize the number of terms* in the ESOP expression. For the expression with the minimum number of terms our secondary goal is to minimize the total number of inputs to the AND and EXOR gates. The following *cost function* C is used in our algorithm

$$C = C_T + \frac{C_L}{C_{\text{Lin}}}$$

where

- C_T is the total number of terms in the solution;
- C_L is the total number of input wires to the AND and EXOR gates in the solution;
- C_{Lin} is the total number of input wires to the AND and EXOR gates in the initial function.

For instance, literal $X^{\{0,1,2\}}$ as an input to an AND gate requires a single wire for the 2-by-4 decoder realization of logic with 4-valued inputs [15]. Literal $X^{\{0,1\}}$ if realized as $X^{\{0,1,2\}}X^{\{0,1,3\}}$, requires two wires. Similarly $X^{\{0\}} = X^{\{0,1,2\}}X^{\{0,1,3\}}X^{\{0,2,3\}}$ requires three wires.

According to the cost function, if two solutions have different number of terms, the better solution is the one that has the smaller number of terms, because its C_T is smaller. If two solutions have the same number of terms, the better solution is the one that has the smaller number of inputs, because its $\frac{C_L}{C_{Lin}}$ is smaller.

Example 4: Consider an ESOP $X^{\{0,1\}}Y^{\{2\}} \oplus X^{\{2\}}Y^{\{2\}} \oplus X^{\{0\}}Y^{\{1\}}$ in 4-valued logic. It has 3 terms and 20 inputs (17 inputs to the AND gates and 3 inputs to the EXOR gate). The cost function is $3+(20/20) = 4$. The ESOP can be minimized to $X^{\{0,1,2\}}Y^{\{2\}} \oplus X^{\{0\}}Y^{\{1\}}$. It has 2 terms and 12 inputs (10 inputs to the AND gates and 2 inputs to the EXOR gate). The corresponding cost function is $2+(12/20) = 2.6$. The function can be further minimized to $X^{\{0,1,2\}}Y^{\{1,2\}} \oplus X^{\{1,2\}}Y^{\{1\}}$. The cost function is $2 + (10/20) = 2.5$.

C. The Formula

The following properties hold for multiple-valued input algebra.

1) $T \oplus T = 0$. Here T denotes a product term.
2) $X_i^{S_i} \oplus X_i^{R_i} = X_i^{S_i \oplus R_i}$.
3) $X_i^{S_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{R_j} = X_i^{S_i \oplus R_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{S_j \oplus R_j} = X_i^{S_i \oplus R_i}X_j^{R_j} \oplus X_i^{S_i}X_j^{S_j \oplus R_j}$.

The proofs for these properties are straightforward. Properties 1 and 2 directly result from Definition 4. Property 3 is true because

$$\begin{aligned} &X_i^{S_i \oplus R_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{S_j \oplus R_j} \\ &= (X_i^{S_i} \oplus X_i^{R_i})X_j^{S_j} \oplus X_i^{R_i}(X_j^{S_j} \oplus X_j^{R_j}) \\ &= X_i^{S_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{R_j} \\ &= X_i^{S_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{R_j}. \end{aligned}$$

Similarly it is proved that

$$X_i^{S_i \oplus R_i}X_j^{R_j} \oplus X_i^{S_i}X_j^{S_j \oplus R_j} = X_i^{S_i}X_j^{S_j} \oplus X_i^{R_i}X_j^{R_j}.$$

Extending the property 3 to two terms with n literals, we define a new cube operation as follows:
Let $T_S = X_1^{S_1} \cdots X_n^{S_n}$ and $T_R = X_1^{R_1} \cdots X_n^{R_n}$ be two terms. The exorlink of terms T_S and T_R is defined by the following formula:

$$T_S \otimes T_R = \bigoplus \{X_1^{S_1} \cdots X_{i-1}^{S_{i-1}} X_i^{(S_i \oplus R_i)} X_{i+1}^{R_{i+1}} \cdots X_n^{R_n} \mid \text{for such } i = 1, \cdots, n, \text{that } S_i \neq R_i\}.$$

Here $\otimes$ denotes the exorlink operation and $\oplus$ denotes the EXOR operation.

Definition 6: Given terms T_S and T_R, if the distance of two terms is d, then $T_S \otimes T_R$ is a *distance d exorlink*. It was proved in [22] that the exorlink can be applied to any two cubes in an array, without regard to their distance. According to the above formula, we can observe that distance d exorlink generates d resultant terms.

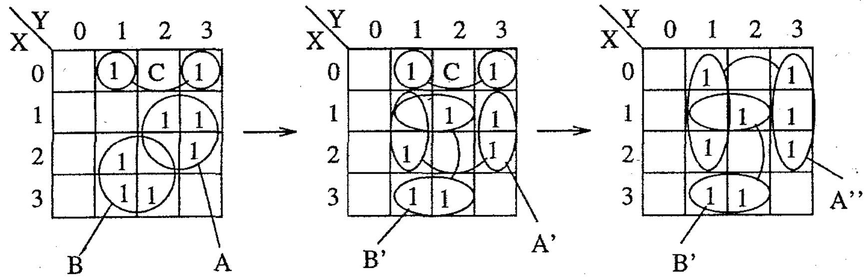


Fig. 1. ESOP minimization corresponding to Example 7.

In the remainder of this section, distance 0, distance 1, distance 2, and distance 3 exorlink operations will be discussed. These operations are used in our EXORCISM-MV-2 algorithm. The comparison of exorlink with the primary and secondary xlink operations [10] and the operations from EXMIN-2 [18] will also be presented in this section.

D. Distance 0 Exorlink

If the distance of two terms is 0, exorlink of the terms generates no resultant terms. These two terms are then removed from the ESOP description.

E. Distance 1 Exorlink

Given two terms T_S and T_R, let X^{S_i} and X^{R_i} be a pair of literals in terms T_S and T_R, respectively, such that $X^{S_i} \neq X^{R_i}$, and the other pairs of literals in the two terms are equal. Therefore, these two terms will be called "*distance 1 exorlinkable*." The distance 1 exorlink operation of the terms generates a single resultant term.

Example 5: Let $T_S = X^{\{1,2,3\}}Y^{\{2,3\}}$ and $T_R = X^{\{0,1\}}Y^{\{2,3\}}$

$$T_S \otimes T_R = X^{\{1,2,3\} \oplus \{0,1\}}Y^{\{2,3\}} = X^{\{0,2,3\}}Y^{\{2,3\}}.$$

F. Distance 2 Exorlink

Given two terms, if the distance of these two terms is 2 (assume $X^{S_i} \neq X^{R_i}$, and $Y^{S_j} \neq Y^{R_j}$), then distance 2 exorlink operation can be performed on them, and two resultant terms will be generated.

Please note that when distance ≥ 2, exorlink operation is not symmetric, which means $T_S \otimes T_R$ is different from $T_R \otimes T_S$.

Example 6: Given two terms $T_S = X^{\{0,1,3\}}Y^{\{1,3\}}$ and $T_R = X^{\{2,3\}}Y^{\{0,1\}}$.

$$\begin{aligned} T_S \otimes T_R &= X^{\{0,1,3\}}Y^{\{1,3\}} \otimes X^{\{2,3\}}Y^{\{0,1\}} \\ &= X^{\{0,1,2\}}Y^{\{0,1\}} \oplus X^{\{0,1,3\}}Y^{\{0,3\}} \\ T_R \otimes T_S &= X^{\{2,3\}}Y^{\{0,1\}} \otimes X^{\{0,1,3\}}Y^{\{1,3\}} \\ &= X^{\{0,1,2\}}Y^{\{1,3\}} \oplus X^{\{2,3\}}Y^{\{0,3\}}. \end{aligned}$$

Distance 2 operations do not directly reduce the number of terms in an ESOP. However, these operations reshape two terms to two different terms, thus provide opportunities for reducing the cost of ESOP's at some later stages. The nonsymmetry property of the distance 2 exorlink gives us two ways to reshape the two terms, which increases the opportunity for searching a better result. Our method on how to apply distance 2 and distance 3 exorlink in ESOP minimization is discussed in Section III.

Example 7: Given is an ESOP with three terms: $T_1 = X^{\{1,2\}}Y^{\{2,3\}}$, $T_2 = X^{\{2,3\}}Y^{\{1,2\}}$, and $T_3 = X^{\{0\}}Y^{\{1,3\}}$. In Fig. 1, the three terms T_1, T_2, and T_3 are represented by three cubes A, B, and C, respectively. $A \otimes B$ generates A' and B'; $A' \otimes C$ generates A''. The ESOP with three cubes is minimized to an ESOP with two cubes, B' corresponding to term $T_4 = X^{\{1,3\}}Y^{\{1,2\}}$, and A'' corresponding to term $T_5 = X^{\{0,1,2\}}Y^{\{1,3\}}$.

G. Distance 3 Exorlink

Distance 3 exorlink generates three resultant terms from two given terms. Distance 3 exorlink increases the number of terms in the ESOP. However, increasing the number of terms may help to reduce the number of terms at some later stage, and subsequently lead to better results.

Example 8: In binary logic, a given ESOP with four cubes is as follows:

$$000x,\ 0x11,\ x11x,\ 1010.$$

The distances between any pair of cubes from the above set is 3. So, there are no distance 1 or distance 2 operations that can be applied to this set of cubes. Performing distance 3 exorlink on the first two cubes leads to three cubes

$$000x \otimes 0x11 = 0111 \oplus 00x1 \oplus 0000.$$

Replacing the first two cubes by these three cubes, a new ESOP with five cubes is obtained

$$0111,\ 00x1,\ 0000,\ x11x,\ 1010.$$

Since the distance of two cubes: 0000 and 1010 is 2, a distance 2 exorlink can be performed on them

$$0000 \otimes 1010 = x010 \oplus 00x0.$$

After this operation, the ESOP contains five cubes

$$0111,\ 00x1,\ x010,\ x11x,\ 00x0.$$

Now, the distance of cubes $00x1$ and $00x0$ is 1, a distance 1 exorlink can be performed on them

$$00x1 \otimes 00x0 = 00xx.$$

The ESOP now contains four cubes

$$0111,\ 00xx,\ x010,\ x11x.$$

Performing distance 2 exorlink on cubes $x010$ and $x11x$, we obtain

$$x010 \otimes x11x = xx10 \oplus x111.$$

The ESOP is

$$0111,\ 00xx,\ xx10,\ x111.$$

Cubes 0111 and $x111$ can be combined into one cube

$$0111 \otimes x111 = 1111.$$

The final result is an ESOP with three cubes

$$00xx,\ xx10,\ 1111.$$

By using distance 3 exorlink, the number of cubes in the ESOP is temporarily increased from 4 to 5, but this increase helps to jump out of a local minimum of the cost function, and ultimately achieve a better result of 3 cubes.

H. Comparison with Other Cube Operations

In EXORCISM [10], as well as in Hermes [20], two operations are used to link the cubes, the primary xlink and the secondary xlink. Both operations can be applied under certain conditions [10], [20]. If two cubes are of the same dimension, then the primary xlink operation can be applied. If the distance of two cubes is 1, then the secondary xlink can be applied. In the above two cases, exorlink generates the same results as the primary xlink or secondary xlink operations. Since exorlink can be applied without any condition, both primary xlink and secondary xlink are special cases of exorlink. In EXMIN2 [18], a set of rules are used to link two cubes. Each rule can be applied under certain conditions. For instance, the rule RESHAPE can be applied on two terms X^AY^B and X^CY^D if $A \cap C = \phi$ and $B \supset D$ [18]. Rule 1 (X-MERGE) in EXMIN2 is equivalent to distance 1 exorlink operation. Similarly, Rule 2 to Rule 9 in EXMIN2 (RESHAPE, $\cdots$, X-REDUCE-3) are all special cases of distance 2 exorlink. Both the xlink and the rules in EXMIN2 do not cover all the cases for which two cubes are linkable. For instance, if $A \cap C \neq \phi$, $B \cap D \neq \phi$, $A \not\supset C$, $C \not\supset A$, $B \not\supset D$, and $D \not\supset B$, neither xlink nor the rules in EXMIN2 can be applied. Since exorlink can be applied unconditionally, it covers all the cases including those not covered by xlink and the rules in EXMIN2. The operation unlink is used in EXORCISM and the rule SPLIT is used in EXMIN2, for temporary increase of the number of cubes. The same functionality is achieved by distance 3 exorlink in EXORCISM-MV-2. Concluding, in our program, all the previous operations are combined into a single operation, described by one formula. Many particular operations can be obtained as special cases of this formula. The number of operations in EXMIN2 is larger than that in EXORCISM—this is one of the reasons why EXMIN2 generates better results than EXORCISM [18]. Similarly, exorlink is superior because it is a superset of all operations introduced in EXORCISM, Hermes, and EXMIN2.

III. The Algorithm of EXORCISM-MV-2

In this section, our algorithm to minimize MIESOP's is presented. For a completely specified function, the input is the array of MIESOP cubes of the function. Since the cubes from this array are ON-cubes, we call this array the ON-array. One has to keep in mind, however, that contrary to the SOP case, this array represents an EXOR of product terms (cubes), so it includes also OFF-minterms, since even numbers of overlapping ON-cubes produce OFF-minterms in their intersection. Thus, for completely specified functions, the ON-array specifies all ON-terms and some OFF-terms, and all nonspecified minterms are OFF-terms. In the case of an incompletely specified function, both the ON-array and the DC-array are used as the input to the program. Thus, the ON-array specifies the ON-terms and the OFF-terms, the DC-array specifies the DC-terms, and all nonspecified minterms are the OFF-terms. The MIESOP ON-array being the input to our program can represent one of following:

1) A nondisjoint MIESOP;
2) An array of disjoint cubes (a particular case of the MIESOP);
3) A set of minterms (a particular case of a set of disjoint cubes).

If both the ON-array and the DC-array are used, they are not necessarily in the same forms. The output of our program is in a MIESOP form. Using an option "unlink" from the program (which is a set of distance 2 exorlink operations) the output can be changed to an array of disjoint cubes. This way the output data can be either given back to the input of EXORCISM-MV-2 to be further minimized, or it becomes an input to other programs. The output MIESOP array can be also directly given back again to our program for further minimization. Since our algorithm is a heuristic one, the results may vary if a different starting point is used. If the initial function is in a SOP form, we use disjoint sharp option from ESPRESSO [14] to transform it into a disjoint form that is next read by our program.

A. Minimization of Completely Specified Functions

As we discussed in Section I-F, the main purpose of distance 2 operations is to provide opportunities for applications of distance 1 or distance 0 operations. Both EXORCISM and EXMIN-2 perform all the distance 2 operations. Our experiments show that such a method may not be efficient, and it may lead the program to falling into an infinite loop [16]. Therefore, in our new algorithm, instead of doing all possible distance 2 operations, only those distance 2 operations are performed which lead to distance 0 or distance 1 operations. More specifically, if the distance of two cubes, A and B, is 2, then $A \otimes B$ generates a pair of resultant cubes C_1 and C_2, while $B \otimes A$ generates a pair of resultant cubes D_1 and D_2. At this point, there are three choices.

1) Take C_1 and C_2;
2) Take D_1 and D_2;
3) Take A and B.

By calculating the distance of each of the cubes C_1, C_2, D_1, and D_2 with all the cubes in the ESOP array except cubes A

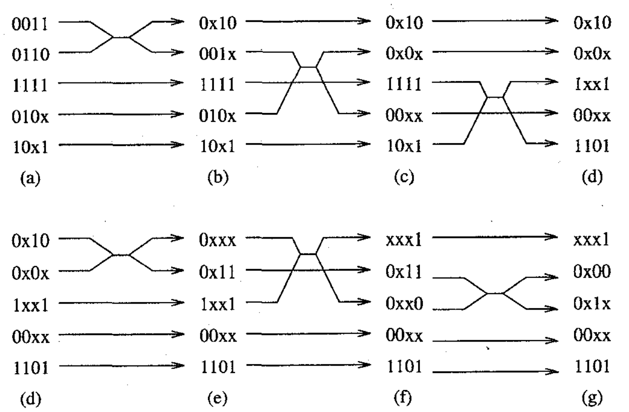


Fig. 2. Procedure of ESOP minimization. (a) The original ESOP. (b) The ESOP after one operation. (c) The ESOP after two operations. (d) The ESOP after three operations. ((d) is repeated to continue the flow) (e) The ESOP after four operations. (f) The ESOP after five operations. (g) The final ESOP.

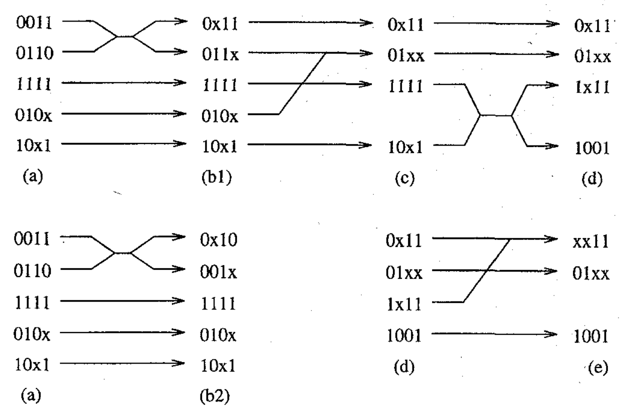


Fig. 3. New procedure of ESOP minimization. (a) The original ESOP. ((a) is duplicated to show two different flows). (b1) The ESOP after one operation. (b2) The ESOP after one operation in a different flow. (c) The ESOP after two operations. (d) The ESOP after three operations. ((d) is repeated to continue the flow) (e) The final ESOP.

and B, we know how many cubes in the array can be reduced if a pair of resultant cubes is selected. Note that a distance 0 operation reduces two cubes in the array and a distance 1 operation reduces one cube. We chose the pair of cubes which leads to a larger reduction. If the number of cubes will be reduced is the same for both pairs, we randomly select one pair. If the cube reduction is 0, cubes A and B are selected.

Example 9: Given an ESOP with 5 cubes

$$0011\ 0110\ 1111\ 010x\ 10x1.$$

If distance 2 operations were performed under this cube ordering, the diagram of applying operations would be as shown in Fig. 2.

After six operations, the function would be represented by another ESOP with 5 cubes, as shown in Fig. 2(g). The number of cubes has not been reduced so far.

The execution of our algorithm is illustrated in Fig. 3.

1) Perform all possible distance 0 and distance 1 exorlink operations. In this example, none of these operations are possible now [see Fig. 3(a)].
2) Check if the distance of two cubes is 2. For instance, the distance of cubes 0011 and 0110 is 2 as shown in Fig. 3(a).
3) Check the two pairs of resultant cubes with other cubes in the array to verify if one can find pairs of cubes whose distance is 0 or 1. In the example, cubes 0011 and 0110 generate two pairs of resultant cubes

$$0110 \otimes 0011 = 0x11 \oplus 011x$$

as shown in Fig. 3(b1), and

$$0011 \otimes 0110 = 0x10 \oplus 001x$$

as shown in Fig. 3(b2). Check these four resultant cubes with the remaining cubes in the array

$$1111\ 010x\ 10x1$$

and it can be found that the distance of cubes $011x$ and $010x$ is 1 as shown in Fig. 3(b1).
4) In step 3, if there are pairs of cubes whose distance is 0 or 1, the distance 2 exorlink is performed and followed by the distance 0 or distance 1 exorlink operations. For instance, in step 3 the distance of two cubes $011x$ and $010x$ is 1,

$$0110 \otimes 0011 = 0x11 \oplus 011x$$

is first performed and then followed by

$$011x \otimes 010x = 01xx$$

as shown in Fig. 3(c).
5) After performing distance 0 or distance 1 exorlink, go back to step 1. If there are no any pairs of cubes whose distance is 0 or 1 in step 3, do not perform distance 2 operation, and go back to step 2 to check other two cubes.

The sequence of steps 1–5 is continued as shown in Fig. 3(d) and 3(e). This procedure is performed iteratively as long as the reduction of component C_T of the cost function is possible. Comparing Fig. 2 with Fig. 3, one can appreciate that our new approach is more efficient.

B. Minimization of Multiple Output Functions

There are two ways to minimize a multiple output function.

1) Decompose the multiple output function to single output functions; minimize each single output function separately; and then minimize jointly the set of functions again;
2) Minimize the multiple output function directly.

Both the EXORCISM and the EXMIN use the first method. In our program, we let the user to select which method to use. The following procedure describes our approach.

1) If the function is a multiple output one and the option "decomposition to single output" is selected, then go to step 2; otherwise, go to step 5;
2) Decompose the function to a set of single output functions;

3) Minimize each single output function separately;
4) Combine the minimized single output functions to a multiple output function;
5) Minimize the multiple output function.

Example 10 illustrates the application of this procedure.

Example 10: Given is a 3-input 2-output binary function $(f_0, f_1) = F(x, y, z)$ which is represented by the following ON-array of cubes

001 10
010 11
101 10
111 11.

The first three symbols in each cube represent the input variables, the last two symbols represent the output variables. For instance, the first cube in the array, 001 10, means that when the input values combination is $x = 0, y = 0$, and $z = 1$, the output variables are $f_0 = 1$ and $f_1 = 0$, respectively. Since this is a two output function, it can be decomposed to two single output functions which are represented by the following array of 6 cubes

001 10
010 10
101 10
111 10
010 01
111 01.

By performing the above procedure, the first four cubes are minimized into the following two cubes

$01x$ 10
$xx1$ 10.

The last two cubes cannot be minimized at this stage. So these two cubes remain the same

010 01
111 01.

Now each single output function has been minimized separately. Next these two single output functions are combined to a multiple output function represented by the following four cubes

$01x$ 10
$xx1$ 10
010 01
111 01.

By minimizing these four cubes, the final result becomes an array of three cubes

$x11$ 01
$xx1$ 10
$01x$ 11.

The last cube is a product term that is common to both output f_0 and f_1.

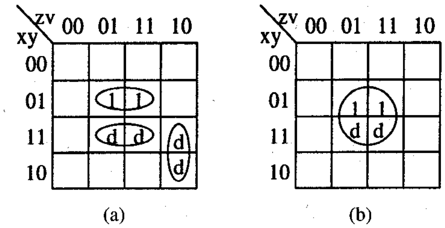


Fig. 4. An example of incompletely specified function. (a) The original ESOP. (b) The minimized ESOP.

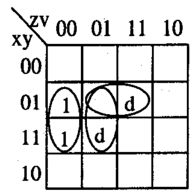


Fig. 5. The DC-array covers a OFF minterm.

C. Minimization of Incompletely Specified Functions

An incompletely specified function can be represented by an ON-array of cubes and a DC-array of cubes. A simple way to minimize an incompletely specified function is to assume that all the DC-cubes are OFF-cubes. However, linking the ON-array of cubes with the DC-array of cubes may generate better results.

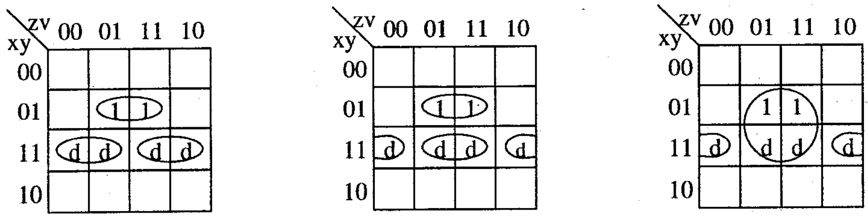
Example 11: Given is an ESOP with one ON-cube, $01x1$, and two DC-cubes, $11x1$ and $1x10$, as shown in Fig. 4(a). The function can be realized by the ON-cube only. By linking the ON-cube with one of the DC-cubes, we get the cube $x1x1$ as shown in Fig. 4(b), which is a better result than ON-cube $01x1$.

Saul [20] pointed out that minimization of incompletely specified functions in ESOP form is difficult, because of the following:

1) The DC-cubes may cover some OFF minterms, as shown in Fig. 5;
2) The DC-array may not contain a cube that can be directly linked with a cube in ON-array because of the positions or sizes of the DC-cubes, as shown in Figs. 6(a) and 7(a), respectively.

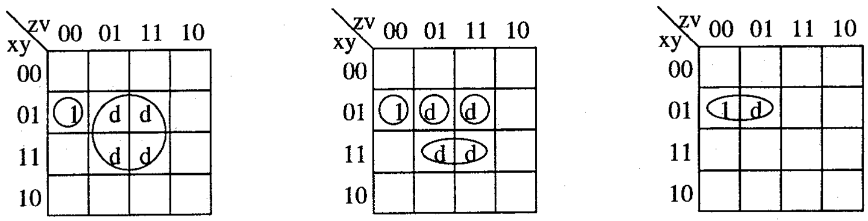
In Fig. 5, cubes $x101$ and $01x1$ are in the DC-array, and cube $x100$ is an ON-cube. The DC-cube $x101$ cannot be linked with the ON-cube $x100$, because the DC-cube $x101$ contains a minterm 0101, which is an OFF minterm. This problem can be solved by making the DC-array disjoint. In a disjoint DC-array, each DC-minterm is covered by a cube once, and an OFF minterm is not covered by any cube.

Saul [20] gave the algorithm to link the ON-cubes with the DC-cubes. His algorithm can reduce the number of connections, but cannot reduce the number of cubes, because only distance 1 link is performed between the ON-cubes and the DC cubes. Moreover, the distance 1 link may not be found by the program due to the position and the size of the DC-cubes. In Fig. 6(a), the ON-cube cannot be linked with any one of the DC-cubes. If the DC-cubes are in the right position, however,



(a) (b) (c)

Fig. 6. Position of DC-cubes. (a) No DC-cubes can be linked. (b) One DC-cube can be linked. (c) The final result.



(a) (b) (c)

Fig. 7. Size of DC-cubes. (a) The original ESOP. (b) The DC-cube is separated. (c) The final result.

they can be linked as shown in Fig. 6(b) and (c). In Fig. 7(a), the ON-cube cannot be linked with the DC-cube, because the size of the DC-cube is larger than the size of the ON-cube, which means the distance between the ON-cube and the DC-cube is not 1. If we can separate the DC-cube properly, as shown in Fig. 7(b), then the ON-cube can be linked with one of the DC-cubes, as shown in Fig. 7(c).

Our approach to minimize incompletely specified functions is described by the following procedure:

```
function don't care minimize (ON-array, DC-array)
for (each ON-cube) {
    R = ON-cube # DC-array
    if (R = φ) remove the ON-cube from ON-array
return ON-array.
}
```

Here R = *ON-cube # DC-array* is the sharp operation between the ON-cube and the DC-array. If R is an empty cube, this means that the ON-cube is covered by the DC-array. The ON-cube can then be removed.

If no more ON-cubes can be removed by this method, we can perform distance 2 exorlink on the ON-array in order to reshape the ON-cubes. Example 12 shows that reshape may help to reduce the ON-cubes.

Example 12: Given ON-array

$$110x$$
$$0x11$$
$$1110$$

and DC-array

$$0x10$$
$$10x1$$

as shown in Fig. 8(a). The minimization is carried out by the following steps:

1) Perform the don't care minimization procedure. Since no ON cubes are covered by the DC cubes, none of the ON-cubes can be removed;
2) Reshape the ON-array as shown in Fig. 8(b). Again, none of the ON-cubes can be removed;
3) Reshape the ON-array as shown in Fig. 8(c), the three ON-cubes are

$$11xx$$
$$xx11$$
$$1011.$$

Since the DC-cube $10x1$ contains the ON-cube 1011, the operation

$$1011\#10x1$$

generates an empty cube. So, the ON-cube 1011 can be removed. The final result is an array of two cubes: $11xx$ and $xx11$.

By performing sharp and distance 2 exorlink iteratively, the number of ON-cubes can be reduced, which serves our primary goal: minimizing the number of terms in the ESOP's. The next step is to achieve our secondary goal: minimizing the number of connections. This is done by trying all possible expanding operations on the ON-cubes.

The next section presents our whole algorithm.

D. The Algorithm of EXORCISM-MV-2

When our method is used for a completely specified function, it uses an *ON-array of cubes* (usually, but not necessarily, these are disjoint cubes). In the case of an incompletely specified function, the function is represented as the *ON-array of ON-cubes* and the *DC-array of DC-cubes (DC-cubes are cubes of don't cares).* The pairs of equal cubes are removed and distance 1 exorlink operations are performed iteratively. Then distance 2 exorlink operations are executed which may

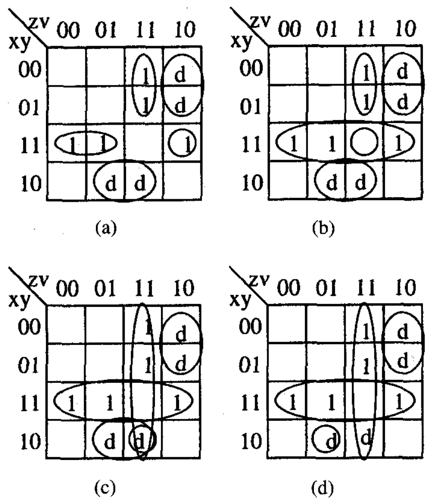


Fig. 8. Minimization of an incompletely specified function. (a) The original ESOP. (b) The ESOP after one operation. (c) The ESOP after two operations. (d) The final result.

provide opportunities for distance 0 or distance 1 exorlink. If distance 2 exorlink operations cannot further improve the cost function, distance 3 exorlink operations are performed. Again, only those distance 3 exorlink operations which lead to distance 0 or distance 1 exorlink operations are performed. After a number of loops, if there is no improvement in the number of terms, distance 2 exorlink operations are performed to minimize the number of connections.

In the case of a multiple-output function, by default, the function is first transformed from a multiple-output array to a set of single-output arrays. Each single-output array is minimized separately, and then the whole function is minimized by using the methods discussed in Section III-B.

For incompletely specified functions, the ON-array is minimized first. Then the sharp operation is executed between each cube in the ON-array and the cubes in the DC-array. If the sharp operation generates an empty cube, the ON-cube will be removed from the ON-array. If no more cubes in the ON-array can be sharped out, distance 2 exorlink operations are performed in order to provide further opportunities. Next, we try to expand each ON-cube into DC-cubes. Successful application of these operations decreases the number of connections.

Since our algorithm is a heuristic one, we do not know whether we have obtained the minimum solution or not. Therefore, some criteria to stop the program are necessary. The following methods can be used as the termination criteria.

1) *Cost functions*: A cost function can be used in the termination criterion. For instance, the program is terminated if the number of terms in the current solution meets a preset value.
2) *Number of iterations*: The program is terminated after a certain number of iterations. This is a simple method, but the quality of the results is not guaranteed.
3) *Execution time*: The program is terminated if time limit has been exceeded. This is the method used in EXORCISM.
4) *Improvements of the current solution*: By this method, the program is controlled by comparing the current result with the previous result. If there is no improvement for a certain number of iterations, the program goes to the next step. This is the method used in ESPRESSO and EXMIN. In our program, this is the method used by default. The user can also select other methods as options.

The whole algorithm is listed below.

Input: ON-array of cubes for a multiple-valued input, multiple-output function (In addition, a DC-array in the case of an incompletely specified function).

1) $F := \text{ON}$; $D := \text{DC}$.
2) SOLUTION := F, MIN_COST := COST(F). (COST(F) is calculated using the cost function shown in Section II-B. MIN_COST will be updated in the steps below to reflect always the lowest cost of solutions obtained until now. This solution is also stored).
3) If the option "do not decompose the function to single output functions" is selected, go to step 5; else go to step 4.
4) Decompose the function to a set of single output functions. For each single output function, perform the steps 5–8.
5) Perform all possible distance 0 operations.
6) Perform all possible distance 1 exorlink operations.
7) For each pair of cubes in F, check if a distance 2 exorlink is possible. If it is possible, further check if it makes a distance 0 operation or a distance 1 exorlink operation possible. If it is possible, perform the distance 2 exorlink operation and then perform distance 0 or distance 1 exorlink operation. Otherwise, do nothing.
8) Check the number of cubes (calculate C_T). If the number of cubes has not been reduced for certain number of iterations go to 9, else go to 5. (Currently the number of iterations is set to the value of 3.)
9) If the option "do not decompose the function to single output function" is selected, go to step 11;
else if the single functions have been combined to a multiple output function, go to step 11;
else if all the single output functions are minimized, go to step 10;
else go to step 5 to minimize the next single output function.
10) Combine the single output functions to a multiple output function, go to step 5.
11) For each pair of cubes in F, check if a distance 3 exorlink is possible. If it is possible, further check if it makes a distance 0 operation or a distance 1 exorlink operation possible. If it does, perform the distance 3 exorlink operation and then perform distance 0 or distance 1 exorlink operation. Otherwise, do nothing.
12) Check the number of cubes. If the number of cubes has not been reduced for certain number of iterations go to 13, else go to 11. (Currently the number of iterations is set to 3.)

13) Check all possible distance 2 exorlink operations. For each distance 2 exorlink operation, if it reduces the cost function COST (F), perform the distance 2 exorlink operation, otherwise, do nothing.
14) If the option "don't care" is not selected, go to 20, else go to 15.
15) If D is empty (no DC cubes) go to 20, else go to 16.
16) Perform a sharp operation between each cube in F and all cubes in D. If an empty cube is returned, remove the corresponding ON cube in F.
17) For each pair of cubes in F, check if a distance 2 exorlink is possible. If it is possible, further check if any resultant cubes from the distance 2 exorlink can be sharped out by D. If they can be sharped, perform the distance 2 exorlink operation and then perform the sharp operation. Remove the corresponding ON cubes in F. Otherwise, do not perform the distance 2 operation.
18) If the number of cubes has not been reduced for certain number of iterations, go to 19, else go to 15. (Currently the number of iterations is set to 3.)
19) Expand each cube in F into D if possible.
20) Print the output and stop the program.

IV. Evaluation of Results of EXORCISM-MV-2

In this section, our experimental results are compared with those published previously in the literature. In the following tables, *inputs* means the number of input variables, *outputs* means the number of output variables, *time* is The CPU time in second. C_T and C_L in the tables are the respective components of the cost function (Section II-B). To evaluate our program, we collected all the published benchmarks we were able to find, and we grouped them into two categories, binary input functions and multiple-valued input functions. Since no authors except ourselves have published results on incompletely specified functions so far, comparison in this category is impossible. Binary input examples are listed in Tables I and II. For multiple-valued input functions, Sasao is the only author besides ourselves who has published results. The comparison is shown in Table III. Table I compares EXORCISM-MV-2 with EXMIN-2 [SAS93b]. In Table I, the first three examples are single output functions, and the remaining are multiple output functions. Comparing with EXMIN-2, EXORCISM-MV-2 generates either the same or in most cases better results.

Table II compares our results with other algorithms. HEALEX [2] uses a different approach and generates good results on some examples. Comparing our results with those given in [2], our program in most cases generates better results. There is only one example in which our solution has one more term. ND is a nondeterministic algorithm given in [3]. This method generally generates good results but is very time consuming. Although the computation time of ND is about 50 times longer than EXMIN2 [18], there are still examples (SQR6 and NRM4) that EXORCISM-MV-2 generates better results. HERMES [20] uses the algorithm similar to our previous algorithm [10]. Table II shows that our new algorithm of EXORCISM-MV-2 gives better results.

TABLE IV
Functions with 3-b Decoders

	1-bit decoder		2-bit decoder		3-bit decoder	
	C_T	C_L	C_T	C_L	C_T	C_L
9sym	51	424	21	176	9	99
Adr4	31	150	11	60	11	96
Clip	65	490	61	577	43	597
Rd53	14	57	9	43	6	36
Rd73	38	191	19	121	13	91
Rd84	57	317	24	179	16	131
Sao2	28	286	27	366	24	496
Wgt8	55	307	23	160	15	137

TABLE V
Incompletely Specified Functions

	ON-cubes			ON- and DC- cubes		
	C_T	C_L	Time	C_T	C_L	Time
bench	27	197	4	24	178	15
fout	54	466	17	48	432	60
p1	60	592	40.5	53	562	107
p3	41	400	12	39	386	50

For multiple-valued functions, Table III shows that EXORCISM-MV-2 generates either the same or better results.

Table IV shows the results with 3-b decoders. It shows that for some of the examples, like Adr4, the solution with 3-b decoders requires more area than the solution with 2-b decoders. This is due to the structure of the problem. For these problems, it is better to use even bit decoders than to use odd bit decoders. While for other problems, odd bit decoders are useful. For instance, benchmark 9sym contains nine input variables. With 2-b decoders, it requires 18 lines to the AND plane, and 21 lines to the EXOR plane. By using three 3-b decoders, it requires 24 lines to the AND plane, and only 9 lines to the EXOR plane. In our program, odd bit decoders can be used together with even bit decoders. For instance, Rd53 contains five input variables. With 2-b decoders, the solution has nine terms. The area is 10 (lines to the AND plane) by 9 (number of terms) = 90. Using one 2-b decoder and one 3-b decoder, the solution has six terms. The area is 12 (lines to the AND plane) by 6 (number of terms) = 72.

Table V shows the experimental results for the minimization of incompletely specified functions. The results of minimizing ON-cubes only are compared with the results of minimizing ON- and DC-cubes. The results show that better results are achieved when DC-cubes are taken into account.

V. Conclusion

One approach to minimize ESOP's is to apply a set of cube operations iteratively on each pair of cubes in the array. Our new operation—exorlink is the most powerful operation in this approach, which can link any two cubes in an array of cubes of an arbitrary distance. All the cube operations introduced previously for this approach in the literature are special cases of the operation introduced by us. The superiority of the new cube operation ensures better results than the previous operations shown in the literature. Based on our new cube operation, a new minimization algorithm is also introduced. Our program EXORCISM-MV-2 was tested on many benchmark functions. The program in most cases gives the same or better solutions on binary and 4-valued completely specified functions. More importantly, it is able to minimize efficiently functions with arbitrary number of values for each variable, which allows to run it with input decoders having more than two binary inputs. Finally, EXORCISM-MV-2 is the only available program to minimize multiple-valued input, multiple-output, *incompletely specified* functions.

REFERENCES

[1] P. W. Besslich, "Efficient computer method for EXOR logic design," *IEE Proc.* Pt. E, vol. 130, 1983, pp. 203–206.

[2] P. W. Besslich and M. W. Riege, "An efficient program for logic synthesis of Mod-2 sum expressions," in *Proc. EUROASIC*, Paris, France, 1991, pp. 136–141.

[3] D. Brand and T. Sasao, "Minimization of AND-EXOR expressions using rewrite rules," *IEEE Trans. Comput.*, vol. 42, pp. 568–576, May 1993.

[4] L. Csanky, M. A. Perkowski, and I. Schäfer, "Canonical restricted mixed-polarity exclusive sums of products and the efficient algorithm for their minimization," in *Proc. IEEE Int. Symp. Circuits Syst.*, San Diego, CA, May 1992, pp. 17–20.

[5] H. Fleisher, M. Tavel, and J. Yeager, "A computer algorithm for minimizing Reed–Muller canonical forms," *IEEE Trans. Comput.*, vol. C-36, pp. 247–250, Feb. 1987.

[6] H. Fujiwara, "Logic testing and design for testability," in *Comput. Syst. Series.* Cambridge, MA: Mass. Inst. Tech., 1986.

[7] D. H. Green, "Reed–Muller canonical forms with mixed polarity and their manipulations," in *Proc. IEE*, Pt. E vol. 137, Jan. 1990, pp. 103–113.

[8] M. Helliwell and M. A. Perkowski, "A fast algorithm to minimize multi-output mixed-polarity generalized Reed–Muller forms," in *Proc. 25th ACM/IEEE Design Automation Conf.*, Anaheim, CA, June 1988, pp. 427–432.

[9] G. Papakonstantinou, "Minimization of modulo-2 sum of products," *IEEE Trans. Comput.*, vol. C-28, pp. 163–167, Feb. 1979.

[10] M. A. Perkowski, M. Helliwell, and P. Wu, "Minimization of multiple-valued input multi-output mixed-radix exclusive sums of products for incompletely specified boolean functions," in *Proc. 19th Int. Symp. Multiple-Valued Logic*, May 1989, pp. 256–263.

[11] M. A. Perkowski and M. Chrzanowska-Jeske, "An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified boolean functions," in *Proc. Int. Symp. Circuits Syst.*, New Orleans, LA, May 1990, pp. 1652–1655.

[12] D. K. Pradhan, *Fault-Tolerant Computing, Theory and Techniques*, vol. I. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[13] J. P. Robinson and C. L. Yeh, "A method for modulo-2 minimization," *IEEE Trans. Comput.*, vol. C-31, pp. 800–801, Aug. 1982.

[14] R. Rudell and A. Sangiovanni-Vincentelli, "ESPRESSO-MV: Algorithms for multiple-valued logic minimization," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1985, pp. 230–234.

[15] T. Sasao, "Multiple-valued decomposition of generalized boolean functions and the complexity of programmable logic arrays," *IEEE Trans. Comput.*, vol. C-30, pp. 635–643, Sept. 1981.

[16] ——, "EXMIN: A simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued input two-valued output functions," in *Proc. 20th Int. Symp. Multiple-Valued Logic*, May 1990, pp. 128–135.

[17] T. Sasao and P. W. Besslich, "On the complexity of mod-2 sum plas," *IEEE Trans. Comput.*, vol. 39, pp. 262–266, Feb. 1990.

[18] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued-input two-valued-output functions," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 621–632, May 1993.

[19] ——, "An exact minimization of AND-EXOR expressions using BDD's," in *Proc. IFIP WG 10.5 Workshop Appl. Reed–Muller Expansion Circuit Design*, Wilhelm Schickard-Institut fuer Informatik, 1993, pp. 91–98.

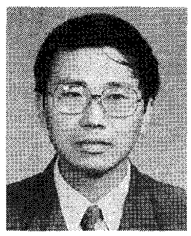
[20] J. M. Saul, "An improved algorithm for the minimization of mixed polarity Reed–Muller representations," in *Proc. Int. Conf. Comput. Design: VLSI Comput. Processors*, Sept. 1990, pp. 372–375.

[21] I. Schäfer and M. A. Perkowski, "Multiple valued generalized Reed–Muller forms," in *Proc. IEEE 21st Int. Symp. Multiple-Valued Logic*, May 1991, pp. 40–48.

[22] N. Song, "Minimization of exclusive sum of products expressions for multiple-valued input incompletely specified functions," Master's degree thesis, EE Dept., Portland State Univ., OR, 1992.

[23] S. Y. H Su and P. T. Cheng, "Computer minimization of multivalued switching functions," *IEEE Trans. Comput.*, vol. C-21, pp. 995–1003, Sept. 1972.

[24] X. Wu, X. Chen, and S. L. Hurst, "Mapping of Reed–Muller coefficients and the minimization of exclusive-OR switching functions," *IEEE Proc.* Pt. E, Jan. 1982, vol. 129, pp. 5–20.



Ning Song received the M.S. degree in management science and computer science from the Shanghai Jiaotong University, China, in 1983, and the M.S. degree in electrical engineering from Portland State University, OR, in 1992, where he is also pursuing the Ph.D degree.

He is currently with Lattice Semiconductor Corporation, Milpitas, CA. His research interests are in logic synthesis, technology mapping, and exclusive-or minimization.

Marek A. Perkowski (M'84) is a Professor with the Electrical Engineering Department of Portland State University, OR. He has published four books and more than 143 refereed papers in the areas of logic synthesis, formal method in hardware design, software and microprogram verification, fault-tolerant computing, design for testability, state machines, petri nets and control models, ovulation monitoring, field programmable gate arrays, robotics for handicapped, medical imaging, and knowledge based systems. In the summer of 1994, he was with Wright Laboratories of the Air Force on applying functional decomposition to machine learning. He has been the Program Chair for the 1995 IEEE International Symposium on Multiple-Valued Logic. His research interests include functional decomposition, field programmable analog arrays, fail-safe design and hardware verification of communicating state machines, abductive logic, and linearly independent transforms and logic.