EXMIN: A Simplification Algorithm for Exclusive-OR-Sum-of-Products Expressions for Multiple-Valued Input Two-Valued Output Functions.

Tsutomu SASAO

Department of Computer Science and
Electronic Engineering,
Kyushu Institute of Technology
Iizuka 820, Japan

Abstract: Minimization of AND-EXOR PLA's with input decoders corresponds to minimization of the number of products in Exclusive-OR Sum-Of-Products expressions (ESOP's) for multiple-valued input two-valued output functions. This paper presents a simplification algorithm for ESOP's. The algorithm is based on an iterative improvement. Seven rules are used to replace a pair of products with another one. We simplified many AND-EXOR PLA's for arithmetic circuits. In most cases, AND-EXOR PLA's required fewer products than AND-OR PLA's.

I. Introduction

An ordinary programmable logic array (PLA) has an AND-OR structure shown in Fig.1. Because PLA's can be designed automatically, easily tested, and easily modified, they are extensively used in modern LSI's.

By replacing the OR array with the EXOR array in the PLA, we have an AND-EXOR PLA shown in Fig.2. AND-EXOR PLA's have several advantages over AND-OR PLA's. Firstly, AND-EXOR PLA's often require fewer products than AND-OR PLA's. Table 1 compares the number of products for various classes of functions [SAS90]. Secondly, AND-EXOR PLA's are easier to test than AND-OR PLA's. Similar to AND-OR PLA's [FUJ81], AND-EXOR PLA's can be made to be universal testable. However, AND-EXOR PLA's require a smaller amount of hardware and shorter test sequence [SAS87].

Although AND-EXOR PLA's have such merits, several problems must be solved before they are used in the practical designs. The first problem is that EXOR's are more expensive and slower than OR's. The second problem is that the design of AND-EXOR PLA's is more difficult than AND-OR PLA's.

In this paper, we consider the design problems of AND-EXOR PLA's. An AND-OR PLA is represented by a set of sum-of-products expressions (SOP's). Similarly, an AND-EXOR PLA is represented by a set of exclusive-or-sum-of-products expressions(ESOP's). In both cases, the number of products in a PLA is equal to the number of different products in the expressions. So, in order to minimize the size of PLA's, it is sufficient to minimize the number of different products in the expressions.

Minimization of SOP's have been studied for more than 30 years. Various algorithms have been developed to obtain minimum [MUR 79] and near minimum [HON74, BRA84, SAS84] PLA's. However, the minimization of ESOP's is much more difficult than that of SOP's. No efficient method is known to obtain a minimum ESOP for a given function. We have developed a simplification algorithms for both SOP's and ESOP's, and designed various PLA's [SAS90]. Our computer experiments show that ESOP's require fewer products than SOP's for most functions. So, AND-EXOR PLA's usually require fewer products than AND-OR PLA's.

It is well known [SAS81] that AND-OR PLA's with decoders shown in Fig.3 require fewer products than AND-OR PLA's without decoders (or AND-OR PLA with one-bit decoders) shown in Fig.1. By replacing the OR array with the EXOR array in the PLA in Fig.3, we have an AND-EXOR PLA with decoders shown in Fig.4. This PLA structure usually requires fewer products than AND-EXOR PLA's without decoders (or AND-EXOR PLA's with one-bit decoders). Similar to AND-OR PLA's with decoders, AND-EXOR PLA's with decoders are represented by ESOP's for multiple-valued input two-valued output functions [SAS 88].

Several simplification algorithms for ESOP's have been developed for two-valued input functions[EVE67, FLE87, HEL88, PAP79, ROB82, SAL79, SAS90, SWA72], and for multiple-valued input functions [PER89, SAS89a].

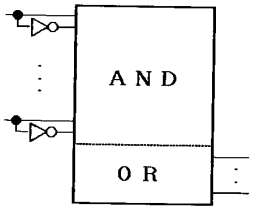

Fig.1 AND-OR PLA with 1-bit decoders

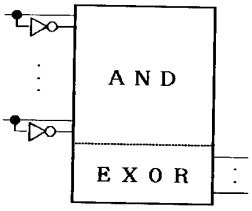

Fig.2 AND-EXOR PLA with 1-bit decoders

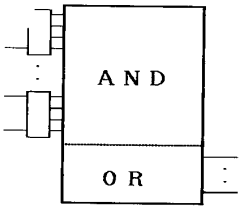

Fig.3 AND-OR PLA with 2-bit decoders

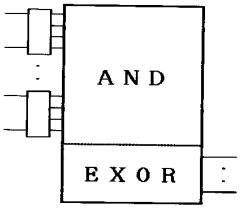

Fig.4 AND-EXOR PLA with 2-bit decoders

In particular, [HEL88] presents an algorithm and minimization results for various AND-EXOR PLA's for multiple-output functions. In [PER89], they extended the algorithm to treat AND-EXOR PLA's with input decoders.

This paper considers the same problem as [PER89], but uses a different approach. The preliminary version of this paper has been published as [SAS89a], and done independently of [PER89].

This paper is organized as follows:

In II, multiple-valued input two-valued functions and their ESOP's are introduced. Then, the design problem of multiple-output AND-EXOR PLA's with input decoders is formulated.

In III, a simplification algorithm for ESOP's is presented.

In IV, experimental results are shown and they are compared with the results of [HEL88].

II. ESOP's for Multiple-Valued Input Two-valued Output Functions

An AND-OR PLA with r-bit decoders realizes a SOP with 2^r -valued input two-valued output function [SAS88]. Similarly, an AND-EXOR PLA with r-bit decoder realizes an ESOP with 2^r-valued inputs [SAS89a, PER89].

Definition 2.1: A multiple-valued input two-valued output function (function for short) is a mapping
$f(X_1,X_2,\cdots,X_n)$: $P_1\times P_2\times\cdots\times P_n\rightarrow B$,
where X_i is a multiple-valued variable,
$P_i=\{0,1,\cdots,p_i-1\}$ is a set of values that this variable may assume, $B=\{0,1\}$, and $p_i\geqq 1$.

Definition 2.2: Let X be a variable which takes one of values in $P=\{0,1,\ldots,p-1\}$. For any subset $S\subseteq P$, X^S is a literal representing the function that

$$X^S = \begin{cases} 1 & \text{if } X\in S \\ 0 & \text{if } X\notin S \end{cases}$$

Definition 2.3: A product of literals $X_1^{S1}\cdot X_2^{S2}\cdots X_n^{Sn}$ is said to be a product term (also called term or product for short). A sum of product terms

$$\sum_{(S_1,S_2,\ldots,S_n)}\oplus X_1^{S1}\cdot X_2^{S2}\cdots X_n^{Sn} \qquad \text{------(2.1)}$$

is a Exclusive-or sum-of-products expression (ESOP).

When $S_i=P_i$, a literal $X_i^{S_i}$ denotes a constant 1, and so the literal is omitted from the products.

Theorem 2.1: An arbitrary multi-valued input two valued output function can be represented by an ESOP of the form (2.1).

(Proof) Let $(a_1,a_2,\cdots,a_n)$ be an element in B^n such that $f(a_1,a_2,\cdots,a_n)=1$. Then f can be represented by

$$\sum_{(a_1,a_2,\ldots,a_n)}\oplus X_1^{\{a_1\}}\cdot X_2^{\{a_2\}}\cdots X_n^{\{a_n\}}$$

This expression is an ESOP. (Q.E.D.)

For a given function, there exist many ESOP's.

Definition 2.4: For a given function, an ESOP is minimum if there exist no ESOP representing the same function with fewer products.

Example 2.1: Table 2 shows a function
$f(X_1,X_2,X_3)$: $P_1\times P_2\times P_3\rightarrow B$,
where $P_1=P_2=\{0,1\}$ and $P_3=\{0,1,2\}$. Fig.5 is a map representing this function. A minimum ESOP for f is $f=X_1^{\{1\}}\cdot X_2^{\{1\}}\oplus X_3^{\{2\}}$.

(End of Example)

Note that in the case of ESOP, every minterm of f must be covered by loop(s) odd times.

Table 2 Function f

X_1	X_2	X_3	f
0	0	0	0
0	0	1	0
0	0	2	1
0	1	0	0
0	1	1	0
0	1	2	1
1	0	0	0
1	0	1	0
1	0	2	1
1	1	0	1
1	1	1	1
1	1	2	0

2.2 Multiple-Output Function

When the circuit has more than two outputs, independent minimization does not always produces total minimization. In this part, we consider a minimization method for multiple-output AND-EXOR PLA's.

Table 1 Complexity of PLA's to realize various functions

	AND-OR PLA		AND-EXOR PLA	
	with 1-bit decoders	with 2-bit decoders	with 1-bit decoders	with 2-bit decoders
Arbitrary functions	2^{n-1}	$\frac{1}{2}\cdot 2^{n-1}$	$\frac{3}{4}\cdot 2^{n-1}$	$\frac{1}{2}\cdot 2^{n-1}$
Symmetric functions	2^{n-1}	$\frac{1}{3}\cdot 3^{n/2}$	$\frac{2}{3}\cdot 3^{n/2}$	$\frac{1}{3}\cdot 3^{n/2}$
Parity functions	2^{n-1}	$\frac{1}{2}\cdot 2^{n/2}$	n	$\frac{n}{2}$
n-bit adders	$6\cdot 2^n-4\cdot n-5$	n^2+1	$2^{n+1}-1$	$\frac{1}{2}\cdot(n^2+3n-2)$

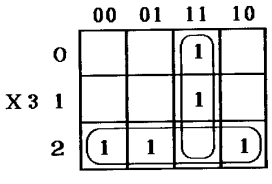

Fig.5 Map of Table 2

Definition 2.5: Let an m-output function be $f_i(X_1,X_2,\cdots,X_n)$, where (i=0,1,...,m-1).

A characteristic function for the multiple-output function is defined as follows [SAS84]:

$$F(X_1,X_2,\cdots,X_n,X_{n+1}) = \bigvee_{i=0}^{m-1} X_{n+1}^{\{i\}} \cdot f_i(X_1,X_2,\cdots,X_n) ,$$

where $X_{n+1}^{\{i\}}$ denotes the i-th output.

If the combinations of inputs and outputs are allowed in the original multiple-output function, then F=1, else F=0, in the characteristic function,

Theorem 2.2: A minimization of a PLA realizing a multiple-output function corresponds to a minimization of an ESOP for the characteristic function.

(Proof) Let F be a characteristic function of f_i (i=0,1,..,m-1).Let Φ_1 be a minimum ESOP for F, and t_1 be the number of products in the ESOP.

When we restrict the domain of the function F into $X_{n+1}=i$, we have the function f_i. Now, consider PLA1 where each product line corresponds to each product term of Φ_1 , and the variable X_{n+1}corresponds to the output part. Then PLA1 realizes the multiple-output functions $(f_0,f_1,\ldots,f_{m-1})$. Note that the number of product lines of PLA1 is t_1.

Next, consider a minimum PLA2 realizing a multiple-output function $(f_0,f_1,\ldots,f_{m-1})$. Let t_2 be the number of product lines in PLA2.

Let Φ_2 be the ESOP representing the function of PLA2. It is clear that Φ_2 represents a characteristic function F. Note that t_2 is equal to the number of products in Φ_2.

Because Φ_1 is a minimum ESOP for the characteristic function F, we have $t_1 \leqq t_2$. On the other hand, because PLA2 is a minimum PLA realizing the multiple-output function, we have $t_1 \geqq t_2$. Hence, $t_1 = t_2$. Therefore, we can minimize the PLA by minimizing an ESOP for the characteristic function. (Q.E.D.)

Example 2.2: Consider the 2-input 3-output function (f_1,f_2,f_3) shown in Table 3. Note that the characteristic function F is equal to f in Table 2. A minimum ESOP for F is

$$F = X_1^{\{1\}} \cdot X_2^{\{1\}} \oplus X_3^{\{2\}},$$

where X_3 represents the output part and may assume three values.

By restricting the above ESOP to X_3=0, X_3=1, and X_3= 2, we have minimum ESOP's for f_1, f_2, and f_3 :

$$f_0=f_1 = X_1^{\{1\}} \cdot X_2^{\{1\}} \quad , \quad \text{and}$$

$$f_2 = X_1^{\{1\}} \cdot X_2^{\{1\}} \oplus 1 .$$

Fig.6 shows the PLA realizing the above expressions. (End of Example)

Table 3 Function (f_0,f_1,f_2)

X_1 X_2	f_0 f_1 f_2
0 0	0 0 1
0 1	0 0 1
1 0	0 0 1
1 1	1 1 0

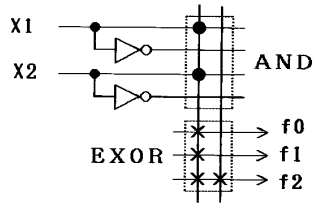

Fig.6 AND-EXOR PLA

III. Simplification Algorithm

3.1 Outline of the Algorithm

For absolute minimization of ESOP's, no efficient algorithm is known except for the exhaustive method [KOD89]. For near minimum ESOP's, most algorithms use iterative improvement methods [FLE87, EVE67, HEL88, PER89, BES83, SAS90, PAP79]. Some of the above algorithms use RME's and others use SOP's of minterms for their initial solutions. However, they require excessive memory space when the number of inputs is large.

The algorithm is called EXMIN, and has the following features:

(1) It simplifies ESOP's for multiple-valued input two-valued output functions.
(2) As initial solutions, disjoint SOP's are derived from simplified ESOP's. This produces an initial solution a moderate size.
(3) Seven rules are iteratively used to simplify ESOP's .
(4) As for multiple-output functions, total set of functions are simplified again after each function is simplified independently.

3.2 Initial Solution

In order to treat large scale PLA's, the initial solutions must be small enough to be stored in a memory of the computer. In this algorithm, the initial solutions are disjoint SOP's derived from simplified SOP's.

Definition 3.1: A SOP in which each pair of products is disjoint is called Disjoint Sum-Of-Products expression (DSOP).

In a DSOP, the OR operators can be replaced with the EXOR operators without changing the function represented by the expression. We have developed an algorithm which converts SOP's into DSOP's. Some heuristics are used to make the resulting DSOP's have as few products as possible.

3.3 Rules for Simplification

The algorithm uses the following seven rules:

(1) X-MERGE

$$X^a \oplus X^b = X^{(a \oplus b)}$$

(2) RESHAPE

$$X^a Y^b \oplus X^c Y^d = X^a Y^{(b \cap \bar{d})} \oplus X^{(a \cup c) Y^d}$$

if ($a \cap c = \phi$, $b \supset d$)

(3) DUAL-COMPLEMENT

$$X^a Y^b \oplus X^c Y^d = X^c Y^{(b \cap \bar{d})} \oplus X^{(\bar{a} \cap c) Y^b}$$

if ($a \subset c$, $b \supset d$)

(4) X-EXPAND-1

$$X^a Y^b \oplus X^c Y^d = X^a Y^{(b \cup d)} \oplus X^{(a \cup c) Y^d}$$
$$= X^{(a \cup c) Y^b} \oplus X^c Y^{(b \cup d)}$$

if ($a \cap c = \phi$, $b \cap d = \phi$)

(5) X-EXPAND-2

$$X^a Y^b \oplus X^c Y^d = X^{(a \cup c) Y^b} \oplus X^c Y^{(b \cap \bar{d})}$$

if ($a \cap c = \phi$, $b \supset d$)

(6) X-REDUCE-1

$$X^a Y^b \oplus X^c Y^d = X^{(a \cap \bar{c}) Y^b} \oplus X^c Y^{(d \cap \bar{b})}$$

if ($a \supset c$, $b \subset d$)

(7) X-REDUCE-2

$$X^a Y^b \oplus X^c Y^d = X^{(a \cap \bar{c}) Y^b} \oplus X^c Y^{(b \cap \bar{d})}$$
$$= X^a Y^{(b \cap \bar{d})} \oplus X^{(a \cap \bar{c}) Y^d}$$

if ($a \supset c$, $b \supset d$)

Fig.7 illustrates the above seven rules for 4-valued input functions. Among the seven rules, only X-MERGE reduces the number of the products in ESOP's. The other rules do not reduce the number of products, but modify the shape of products to make X-MERGE applicable. RESHAPE is also used to modify the shape of the products in SOP's [HON74]. Other rules are specific to ESOP's. For some classes of functions, such as parity functions, X-EXPAND-1 and X-EXPAND-2 are effective to reduce the number of products. X-EXPAND-1 produces two different results, and DUAL-COMPLEMENT interchange the two results. Also, note that if RESHAPE (or DUAL-COMPLEMENT) is applied twice to a pair of products, then the rule produces the original pair. X-REDUCE-1 and X-REDUCE-2 are reverse operations of X-EXPAND-1 and X-EXPAND-2, respectively. X-REDUCE-2 produces two different results, and RESHAPE interchange the two results.

3.4 Simplification Algorithm

The proposed algorithm relies on X-MERGE to reduce the number of products. The other rules are used when X-MERGE is unapplicable. The order of rules in the algorithm is quite sensitive to the quality of the solutions. However, the current version of the algorithm does not use any special heuristics on the order of the rules and products.

In the case of a two-valued input multiple-output function, first we decompose it into single output functions, and then simplify each function independently, and finally simplify the total set of function again.

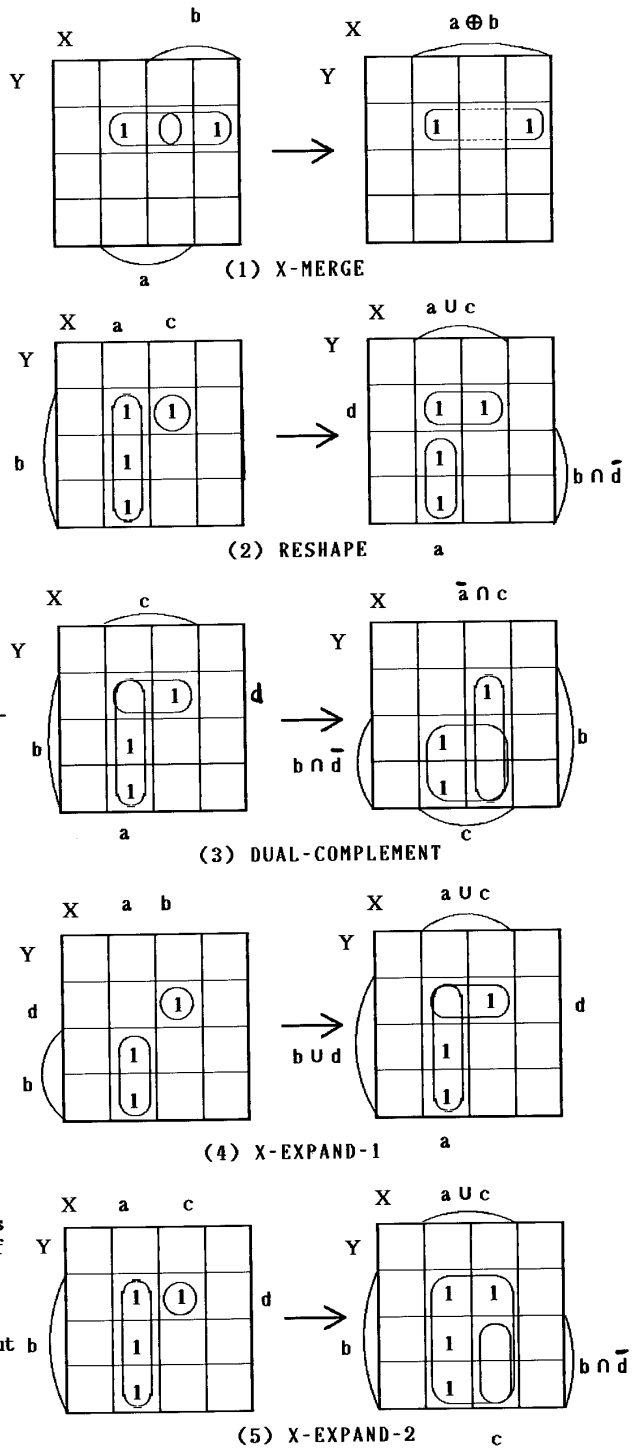

Fig.7 Example for simplification rules

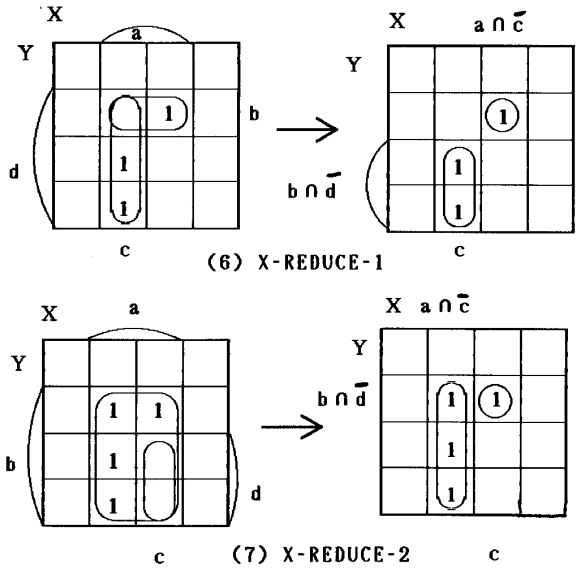

Fig.7 Example for simplification rules (continued)

In the case of a multi-valued input function, we have to be careful not to fall into a infinite loop in the program. That is, the successive application of X-EXPAND's and DUAL-COMPLEMENT can make the original cover, and so the program may not halt.

Example 3.1: Fig.8 illustrates that the successive application of X-EXPAND-2, DUAL-COMPLEMENT, X-EXPAND-2 and DUAL-COMPLEMENT will produce the original cover. (End of Example)

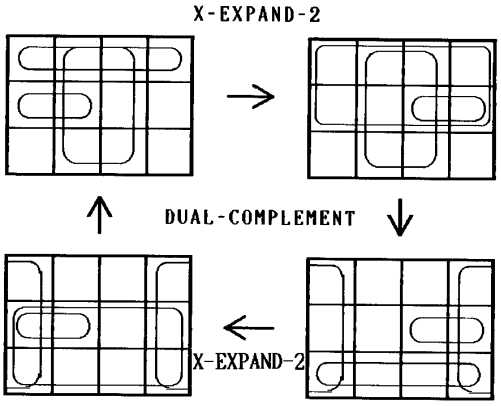

Fig.8 Example of infinite loop

In order to prevent the program falling into the infinite loop, we introduce the notion of a volume of an ESOP.

Definition 3.2: Let |S| denote the number of elements in S. The volume of a product $X_1^{S_1} \cdot X_2^{S_2} \cdots X_n^{S_n}$ is $|S_1| \cdot |S_2| \cdots |S_n|$. The volume of an ESOP is sum of volumes of the products in the ESOP.

Next lemmas consider the change of the volume of an ESOP in applying the rules.

Lemma 3.1: X-MERGE does not increase the volume.

(Proof) Let the original cubes be X^a and X^b. The volume of the original cubes is V0=|a|+|b|. The volume of the cube after X-MERGE is V1=|a|+|b|-|a∩b|. Hence, V1≦V0. (Q.E.D.)

Lemma 3.2: RESHAPE does not change the volume.

(Proof) Let the original cubes be X^aY^b and X^cY^d. The volume of the original cubes is V2=|a|·|b|+|c|·|d|. The volume of the cubes after reshape is V3=|a|·(|b|-|d|)+(|a|+|c|)·|d|. Hence, V2=V3. (Q.E.D.)

Lemma 3.3: DUAL-COMPLEMENT does not change the volume for two-valued input functions, but may decrease the volume for multi-valued input functions.

(Proof) The volume of the cubes after DUAL-COMPLEMENT is V4=|c|·(|b|-|d|)+(|c|-|a|)·|b|. So the difference of the volume is V4-V2=2{|b|·|c|-|c|·|d|-|a|·|b|}. In the case of 2-valued input functions, |b|=|c|=2, and |a|=|d|=1. So, V4=V2. In the case of multi-valued input functions, DUAL-COMPLEMENT may decreases the volume of the cubes as shown in Fig.8. (Q.E.D.)

Lemma 3.4: X-EXPANDs increase the volume.

(Proof) 1) Volumes of the cubes after X-EXPAND 1 are V5=|a|·(|b|+|d|)+(|a|+|c|)·|d|, and V6=(|a|+|c|)·|b|+|c|·(|b|+|d|), respectively. The differences of the volumes are V5-V2=2·|a|·|d|>0, and V6-V2=2·|b|·|c|>0.

2)The volume of the cubes after X-EXPAND 2 is V7=(|a|+|c|)·|b|+|c|·(|b|-|d|). The difference of the volume is V7-V2=2·|c|·{|b|-|d|}>0.

Hence, X-EXPANDs increase the volume of the cubes. (Q.E.D.)

Lemma 3.5: X-REDUCEs decrease the volume.

(Proof) X-REDUCEs are reverse operations of X-EXPANDs. So, X-REDUCEs decrease the volume of the cubes. (Q.E.D.)

Example 3.2: Consider the function in Fig.8. X-EXPAND-2 increases the volume, but DUAL-COMPLEMENT decrease the volume. This fact implies that the original ESOP is obtained after applying several rules. (End of Example)

As shown in the previous example, the application of DUAL-COMPLEMENT after X-EXPAND's may produce the original ESOP. In the case of two-valued input functions, RESHAPE and DUAL-COMPLEMENT do not decrease the volume, and X-EXPAND's increase the volume, so we can halt the algorithm when no other rule other than RESHAPE and DUAL-COMPLEMENT is applicable. On the other hand, in the case of multiple-valued input functions, DUAL-COMPLEMENT may decrease the volume. Therefore, we have to be very careful to make the program not to fall into an infinite loops.

Algorithm 3.1: (EXMIN)

(1) Convert a SOP into a DSOP. Let it be the initial solution.

(2) For a multi-output function, decompose it into single-output functions, and simplify each function independently by the method below, and then simplify the total set of functions again.

(2-1) For each pair of products in the ESOP, check if

X-MERGE is applicable. If so, merge them.

(2-2) For each pair of products in the ESOP, check if RESHAPE, volume non-decreasing DUAL-COMPLEMENT, X-EXPAND-1, and X-EXPAND-2 are applicable in this order. If so, apply the rule. After this, do the following:

(a) For the products modified by the above rules, check if X-MERGE is applicable. If so, merge them and go to (2-2).

(b) If X-EXPAND-1 or X-EXPAND-2 is applied in (2-2), then go to (2-2).

(3) Apply X-REDUCE-1 and X-REDUCE-2.

(4) Simplify total function again by (2-1) and (2-2).

(5) If the number of products is reduced in (4), then go to (3). Otherwise stop.

3.5 Examples of Simplification

In this section, we illustrate the simplification algorithm by using the two-valued input 4-variable function in Fig.9. Note that the SOP is already a DSOP. Also, note that X-MERGE is not applicable. So the only process to do in EXMIN is step (2-2). Example 3.3 will show the simplification by EXMIN, while Example 3.4 will show the simplification by the same algorithm except that the order of the rules are interchanged.

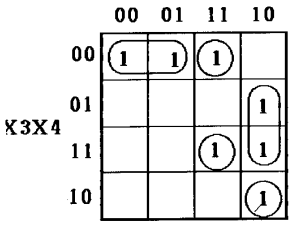

Fig.9 Map for the example function

Example 3.3 : In Fig.10(a), even if we apply RESHAPE, we cannot use X-MERGE. Because we cannot apply DUAL-COMPLEMENT, we try to apply X-EXPAND-1. We have two options to apply this rule: one is the pair of products (①,②), and the other is (②,③).If we apply X-EXPAND-1 to the pair (①,②), we obtain the ESOP shown in Fig.10(b). In this ESOP, we combine ④ and ⑤ by X-MERGE, and have the ESOP shown in Fig.10(c). In this ESOP, however, we cannot apply any rules. So the algorithm halts, and we have an ESOP with four products. (End of Example)

Example 3.4: In the ESOP shown in Fig.11(a), we cannot apply DUAL-COMPLEMENT. Now, we try to apply X-EXPAND-2 instead of X-EXPAND-1. We have three pairs of products to apply this rule: the first pair is (①,②), the second one is (③,④), and the third one is (③,⑤). If we apply X-EXPAND-2 to the pair (①,②), we have the ESOP shown in Fig.11(b). In this ESOP, we combine ⑦ and ④ by X-MERGE, and have the ESOP shown in Fig. 11(c). Now, we can combine ③ and ⑧ by X-MERGE again, and have the ESOP shown in Fig.11(d). Here, we cannot apply the rules anymore. So the algorithm halts, and we have an ESOP with three products. (End of Example)

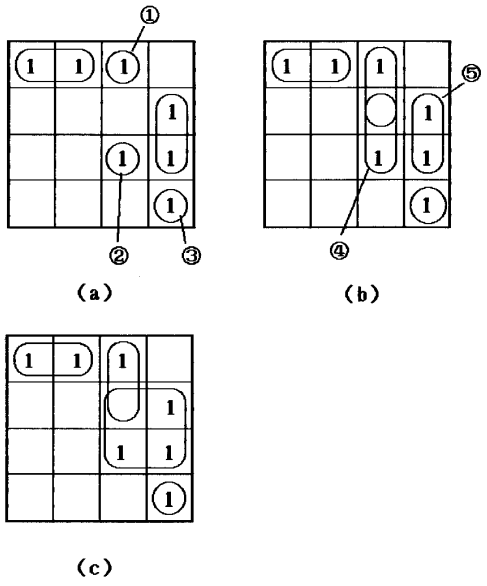

Fig.10 Example of simplification I

As shown in the above examples, the order of the rules in the algorithm is sensitive to the quality of the solution. In Example 3.4, if we apply X-EXPAND-2 to the pair (③,⑤), we get the same result as Example 3.3.

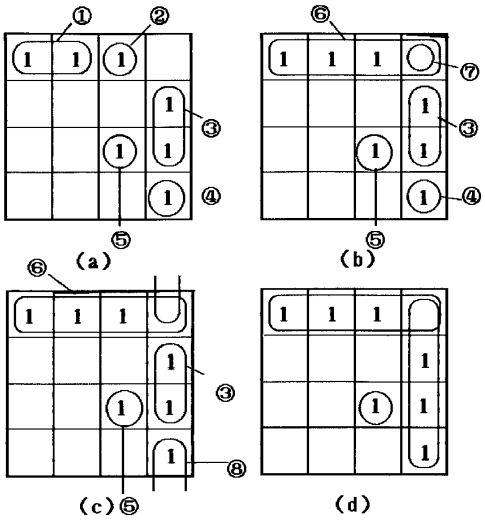

Fig.11 Example of simplification II

IV. Experimental Results

We coded EXMIN in FORTRAN and implemented it on a PC9801N (a personal computer utilizing a 10 MHz NEC V30 microprocessor). We simplified ESOP's for various arithmetic functions. Helliwell and Perkowski developed a simplification algorithm called EXORCISM, and reported its performance in detail [HEL 88]. They used 'primary xlinking' and 'secondary xlinking' rules to simplify ESOP's.

Table 4 compares the number of products and computation time for two programs. In most cases, EXMIN produced better solutions in shorter time. Note that EXORCISM used a 10 MHz IBM-PC AT, which is faster than PC9801N. However, EXMIN require shorter time than EXORCISM to obtain better solutions.The CPU time for EXMIN contains I/O of data.

[PER89] extended the simplification algorithm for ESOP's to treat multiple-valued input two-valued output functions. However, they did not show the experimental results. So, no statistical data have been published for the multiple-valued cases.

Table 5 compares the number of products of AND-OR PLA's and AND-EXOR PLA's with one-bit and two-bit decoders to realize various arithmetic functions [SAS88]. This table compares the complexity of AND-OR PLA's with AND-EXOR PLA's rather than showing the performance of EXMIN. We used QM [SAS84] to obtain absolute minimum solutions. For PLA's with two-bit decoders, we used an exhaustive method to find the optimum input assignments. Therefore, these data are all absolute minimum. On the other hand, for the AND-EXOR PLA's, we used EXMIN several times iteratively to obtain near minimum solutions. Table 5 lists the best solutions among several tries. Note that the solutions may not be minimum for AND-EXOR PLA's. From this table, we can see that PLA's with two-bit decoders require fewer products than PLA's with one-bit decoders for both AND-OR and AND-EXOR PLA's. Also, AND-EXOR PLA's require fewer products than AND-OR PLA's except for the case of LOG8. It might be possible to simplify these PLA's further, but currently we have no algorithm to obtain a better solution.

As for the number of products in these PLA's, we have the following results except for a few cases.

A > B > C > D, where
A=# of products in AND-OR PLA's with 1-bit decoders,
B=# of products in AND-EXOR PLA's with 1-bit decoders,
C=# of products in AND-OR PLA's with 2-bit decoders,
D=# of products in AND-EXOR PLA's with 2-bit decoders.

Table 4 Comparison with EXORCISM and EXMIN

Data	EXORCISM		EXMIN	
	term	time sec	term	time sec
ADR4	34	840	32	108
MLP3	19	48	18	11
MLP4	119	7800	66	452
SQR3	7	0	6	4
SQR6	40	180	39	54

Table 5 Number of Products of AND-OR PLA's and AND-EXOR PLA's

Data	AND-OR PLA decoders		AND-EXOR PLA decoders	
	1bit	2bit	1bit	2bit
ADR4	75	17	32	12
LOG8	123	93	105	105
MLP4	121	85	66	56
NRM4	120	70	76	62
RDM8	76	47	31	28
ROT8	57	37	37	32
SQR8	180	142	121	121
WGT8	255	54	66	26

V. Conclusion and Comments

In this paper, we presented a design method for AND-EXOR PLA's with input decoders, and formulated it as a minimization problem of ESOP's (Exclusive-or Sum-Of-Products expressions) for multiple-valued input two-valued output functions. We developed a simplification algorithm called EXMIN, which uses seven rules. We coded it in a FORTRAN program, simplified various functions, and showed that EXMIN produces better solutions in shorter computation time than EXORCISM developed by Helliwell and Perkowski. We also simplified AND-EXOR PLA's with two-bit decoders, and showed that in most cases they require fewer products than AND-OR PLA's with two-bit decoders.

VI. Acknowledgement

The author thanks Mr. M. Higashida who implemented the original version of EXMIN.
This work was supported in part by a research adjunct professorship at the Naval Postgraduate School, Monterey, California, U.S.A.

VII. References

[BES 83] Ph.W.Besslich,"Efficient computer method for EXOR logic design," IEE Proc.,vol.130, Part E, pp.203-206, 1983.
[BRA 84] R.K.Brayton, G.D.Hachtel, C.T.McMullen, and A.L.Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Boston, MA. Kluwer, 1984.
[DUE 86] G.Dueck and D.M.Miller,"A 4-valued PLA using the MOD SUM," Proc. of the 16 th International Sympo. on Multiple-valued Logic,pp.232-240, May 1986.
[EVE67] S.Even, I.Kohavi and A.Paz,"On minimal modulo-2 sums of products for switching functions,"IEEE Trans. Electronic Computers, Vol.EC-16, pp.671-674, Oct.1984.
[FLE87] H.Fleisher, M.Tavel and J.Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms," IEEE Trans.Comput.Vol.C-36,No.2,pp.247-250, Feb. 1987.
[FUJ81] H.Fujiwara and K.Kinoshita,"A Design of programmable logic arrays with universal tests," Joint

special issue on Design for Testability IEEE Trans. Comput., Vol.C-30, No.11,pp.823-828; also IEEE Trans. Circuit and Systems, Vol.CAS-28, No.11, pp.1027-1032, Nov. 1981.
[HEL88] M.Helliwell and M.Perkowski,"A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms,"25th DAC, pp.427-432, 1988.
[HON74] S.J.Hong, R.G.Cain and D.L.Ostapko, "MINI: A heuristic approach for logic minimization," IBM J.Res. & Develop. pp. 443-458, Sept. 1974.
[KOD 89] N.Koda and T.Sasao, "Four-variable AND-EXOR minimum expressions and their properties,"The 21 st Workshop on Fault Tolerant Computing, (in Japanese) July 4, 1989, also Technical Paper of IEICE Japan FTS-89-25, Oct. 24, 1989.
[MUK70] A. Mukhopadhyay and G.Schmitz,"Minimization of Exclusive OR and logical Equivalence of switching circuits," IEEE Trans. Comput., C-19,pp.132-140, 1970.
[MUL54] D.E.Muller,"Application of Boolean algebra to switching circuit design and to error detection,"IRE Trans. Electron. Comput., EC-3, pp.6-12, 1954.
[MUR79] S.Muroga, Logic design and Switching Theory, Wiley-Interscience Publication, 1979.
[PAP79] G.Papakonstantinou, "Minimization of modulo-2 sum of products,"IEEE Trans. Comput., C-28, pp.163-167, 1979.
[PER89]M.Perkowski, M.Helliwell and P.Wu,"Minimization of multiple-valued input multi-output mixed-radix exclusive sum of products for incompletely specified Boolean functions," Proc.of the 19 th International Symposium on Multiple-valued Logic, pp.256-263, May 1989.
[REE54] I.S.Reed,"A class of multiple-error-correcting codes and the decoding scheme," IRE Trans. Information Theory, PGIT-4. pp.38-49, 1954.
[ROB82] J.P.Robinson and Chia-Lung Yeh, "A method for modulo-2 minimization", IEEE Trans. Comput., C-31, pp. 800-801,1982.
[SAL79] K.K.Saluja and E.H.Ong, "Minimization of Reed-Muller canonic expansion," IEEE Trans. Comput., C-28, pp.535-537,1979.
[SAS81] T.Sasao, "Multiple-valued decomposition of generalized Boolean functions and the complexity of programmable logic arrays," IEEE Trans. on Comput., Vol.C-30, No.9, pp.635-643, Sept.1981.
[SAS84] T.Sasao,"Input variable assignment and output phase optimization of PLA's," IEEE Trans. Comput., Vol.C-33, No.10, pp.879-894, Oct. 1984.
[SAS87] T.Sasao and H.Fujiwara, "A design of AND-EXOR PLA's with universal tests," (in Japanese), The IECE Japan Technical paper, FTS86-25,Feb.23,1987.
[SAS88] T.Sasao, "Multiple-valued logic and optimization of programmable logic arrays, " IEEE Computer, Vol. 21, No.4, pp.71-80, April 1988.
[SAS89a]T.Sasao and M.Higashida,"On a design algorithm for AND-EXOR PLA's with input decoders,"(in Japanese), The 20 th Workshop on Fault Tolerant Computing, Jan. 24, 1989, also Technical Paper of IEICE Japan VLD-89-84, Dec. 15, 1989.
[SAS89b] T.Sasao, "On the optimal design of multiple-valued PLA's,"IEEE Trans. on Comput. Vol.38. No.4, pp. 582-592,April 1989.
[SAS90] T.Sasao and P.Besslich, "On the complexity of MOD-2 Sum PLA's,"IEEE Trans. on Comput. vol.32, No.2, pp262-266, Feb.1990.