Easily Testable Realizations for Generalized Reed-Muller Expressions

Tsutomu Sasao, *Fellow*, *IEEE*

Abstract—This paper presents a design method of easily testable AND-EXOR networks. It is an improvement of Reddy and Saluja-Reddy's methods, and has the following features: 1) The network uses generalized Reed-Muller expressions (GRMs) instead of Positive Polarity Reed-Muller expressions (PPRMs). The average number of products for GRMs is less than half of that for PPRMs, and is less than that of sum-of-products expressions (SOPs). 2) The network consists of a literal part, an AND part, an EXOR part, and a check part. 3) The EXOR part can be a tree instead of a cascade. Thus, the network is faster. 4) The test detects multiple stuck at faults under the assumption that the faults occur at most one part, either the literal part, the AND part, the EXOR part, or the check part.

Index Terms—AND-EXOR, testable design, Reed-Muller expression, circuit complexity, logic minimization, linear circuit.

1 Introduction

AND-EXOR based networks are easily testable [6]. Reddy showed that only four tests are required to test an EXOR cascade (Fig. 1) [12]. He combined this idea with the positive polarity Reed-Muller expressions (PPRMs), and showed that only $n + 4$ tests are required to test AND-EXOR realizations, where n is the number of the input variables (Fig. 2). Although these represent a small number of tests, networks based on his idea have the following problems: The first problem is that the network uses a cascade in the EXOR part. The propagation delay of the cascade tends to be large. In modern design, the speed of the network is vitally important, and cascades are unacceptable because of their slow speed. The second problem is an excessive amount of hardware. PPRMs usually require more products than other representations (see Tables 1 and 2). The third problem is that the tests cannot detect multiple-faults.

In this paper, we present an improved network that is easily testable. Its features are:

1) The network consists of a literal part, an AND part, an EXOR part, and a check part (Fig. 3).
2) The EXOR part can be a tree instead of a cascade. Thus, the network is faster.
3) The network uses generalized Reed-Muller expressions (GRMs) instead of Positive Polarity Reed-Muller expressions (PPRMs). The number of products for GRMs is, on the average, less than half of that for PPRMs, and is less than that of sum-of-products expressions (SOPs) (see Tables 1 and 2).
4) The test detects multiple stuck-at-faults under the assumption that the faults occur in at most one part, either the literal part, the AND part, the EXOR part, or the check part.

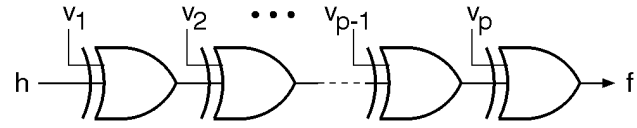

Fig. 1. Test for EXOR cascade.

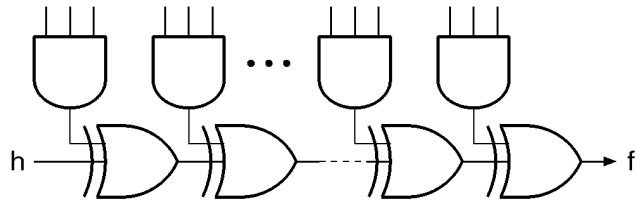

Fig. 2. Test for PPRM.

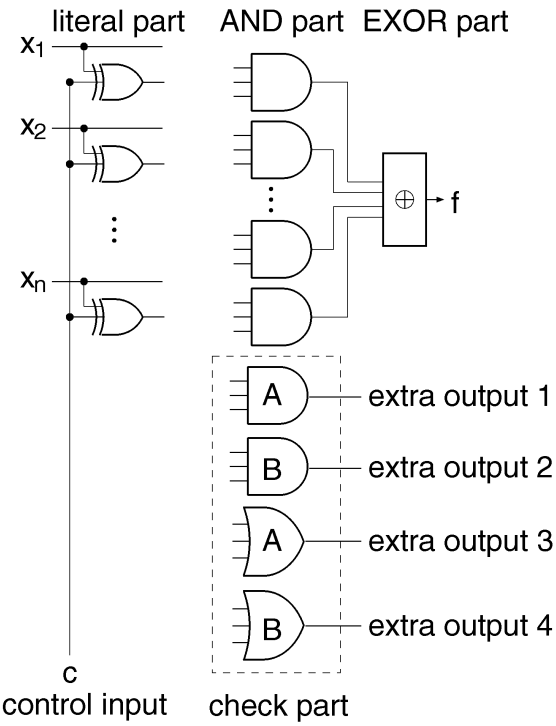

Fig. 3. Easily testable GRM realization.

2 Definitions and Basic Properties

2.1 PPRM, FPRM, GRM, and ESOP

In this part, we will define various classes of AND-EXOR expressions [17], [21].

DEFINITION 1. *x and $\overline{x}$ are literals of a variable x. A logical product which contains at most one literal for each variable is called a* **product term** *(or a* **product** *for short). Product terms combined with OR operators form a* **sum-of-products expression** *(SOP).*

DEFINITION 2. *A* **minterm** *is a logical product containing a literal for each variable. A minterm implying a function f is called a* **minterm of f**.

DEFINITION 3. *An expression for f is* **minimum** *for f if the number of the products is the minimum.*

The following lemma is the basis of the EXOR-based expansion.

• *The author is with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka 820, Japan.*
E-mail: sasao@cse.kyutech.ac.jp.

Manuscript received 19 Feb. 1996; revised 19 Nov. 1996.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number 104713.0.

TABLE 1
NUMBER OF PRODUCTS TO REPRESENT VARIOUS FUNCTIONS ($n = 2r$)

Function	PPRM	FPRM	GRM	ESOP	SOP
$x_1 \oplus x_2 \oplus \cdots \oplus x_n$	n	n	n	n	2^{n-1}
$\bar{x}_1\bar{x}_2\cdots\bar{x}_n$	2^n	1	1	1	1
$x_1x_2\cdots x_n \vee \bar{x}_1\bar{x}_2\cdots\bar{x}_n$	$2^n - 1$	$2^{r+1} - 2$	n	2	2
$x_1x_2 \vee x_3x_4 \vee \cdots \vee x_{n-1}x_n$	$2^r - 1$	$2^r - 1$	$2^r - 1$	$2^r - 1$	r
worst case ($n = 5$)	32	21	10	9	16

TABLE 2
NUMBER OF FOUR-VARIABLE FUNCTIONS REQUIRING t PRODUCTS

t	PPRM	FPRM	GRM	ESOP	SOP
0	1	1	1	1	1
1	16	81	81	81	81
2	120	836	2212	2268	1804
3	560	3496	20856	21744	13472
4	1820	8878	37818	37530	28904
5	4368	17884	4512	3888	17032
6	8008	20152	56	24	3704
7	11440	11600			512
8	12870	2336			26
9	11440	240			
10	8008	32			
11	4368				
12	1820				
13	560				
14	120				
15	16				
16	1				
av	8.00	5.50	3.68	3.66	4.13

LEMMA 1. *An arbitrary logic function $f(x_1, x_2, \ldots, x_n)$ can be expanded as follows:*

$$f = \bar{x}_1 f_0 \oplus x_1 f_1, \tag{1}$$

$$f = f_0 \oplus x_1 f_2, \tag{2}$$

and

$$f = f_1 \oplus \bar{x}_1 f_2, \tag{3}$$

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

2.1.1 PPRM (Positive Polarity Reed-Muller Expression)

Equations (1), (2), and (3) are called the Shannon expansion, the positive Davio expansion, and the negative Davio expansion, respectively. In particular, if we apply (2) recursively to a function f, then we can represent it as follows:

LEMMA 2. *An arbitrary n-variable function $f(x_1, x_2, \ldots, x_n)$ can be represented as*

$$\begin{aligned} f = {} & a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n \\ & \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus \cdots \oplus a_{n-1n}x_{n-1}x_n \oplus \\ & \cdots\cdots\cdots\cdots \\ & \oplus a_{12\cdots n}x_1x_2x_3 \cdots x_n . \end{aligned} \tag{4}$$

Equation (4) is called a positive polarity Reed-Muller expression (PPRM). For a given function f, the coefficients $a_0, a_1, a_2, \ldots, a_{12\cdots n}$ are unique. Thus, PPRM is a canonical representation. Note that all the literals are positive. Because PPRM is unique for a given function, we cannot simplify the expression.

2.1.2 FPRM (Fixed Polarity Reed-Muller Expression)

In (4), for each x_i ($i = 1, 2, \ldots, n$), if we use either a positive literal (x_i) or a negative literal ($\bar{x}_i$), then we have a fixed polarity Reed-Muller expression (FPRM). For each variable x_i, there are two ways of choosing the polarities: positive (x_i) or negative ($\bar{x}_i$). Thus, there are 2^n different sets of polarities for an n-variable function. For a given function and a given set of polarities, there exists a unique set of coefficients ($a_0, a_1, \ldots, a_{12\cdots n}$). Thus, FPRM is a canonical representation.

2.1.3 GRM (Generalized Reed-Muller Expression)

In (4), if we can freely choose the polarity for each literal, then we have a generalized Reed-Muller expression (GRM). A GRM is also called a canonical restricted mixed polarity form (CRMP) [1]. Note that some authors use GRMs to represent other classes of expressions [7]. There are $2^{n2^{n-1}}$ different sets of polarities for an n-variable function. This follows the fact that for each of the $n2^{n-1}$ literal in (4), there are two ways to choose the polarity. For a given set of polarities of literals, there is a unique set of coefficients ($a_0, a_1, \ldots, a_{12\cdots n}$). Thus, a GRM is a canonical representation for a logic function. Recently, we have developed an exact minimization algorithm [20] and a simplification algorithm for GRMs [4]. The first one can minimize any functions up to five variables and some functions with six variables, and the second one quickly simplifies the GRM for the functions with more variables. Other algorithms have also been developed [1].

2.1.4 ESOP (Exclusive or Sum-of-Products Expression)

Arbitrary product terms combined by EXORs is called an exclusive-or sum-of-products expression (ESOP). The ESOP is the most general AND-EXOR expression. EXMIN2 is a heuristic minimization algorithm for ESOPs, and obtains near minimal solutions in a reasonable computation time [18]. An exact minimization program is also available, but it is very time and memory consuming [19].

EXAMPLE 1.

1) $x_1x_2x_3 \oplus x_1x_2$ is a PPRM.
2) $x_1x_2\bar{x}_3 \oplus x_2\bar{x}_3$ is a FPRM, but not a PPRM (x_3 has negative literals).
3) $x_1 \oplus x_2 \oplus \bar{x}_1x_2$ is a GRM, but not a FPRM (x_1 has both positive and negative literals).
4) $x_1x_2x_3 \oplus \bar{x}_1\bar{x}_2\bar{x}_3$ is an ESOP, but not a GRM (it has two products with the form $x_1^*x_2^*x_3^*$, where x_i^* denotes either x_i or $\bar{x}_i$).

THEOREM 1. *Suppose that $\mathcal{PPRM}$, $\mathcal{FPRM}$, $\mathcal{GRM}$, and $\mathcal{ESOP}$ denote the set of PPRMs, FPRMs, GRMs, and ESOPs, respectively. Then, the following relation holds: $\mathcal{PPRM} \subset \mathcal{FPRM} \subset \mathcal{GRM} \subset \mathcal{ESOP}$* (Fig. 4).

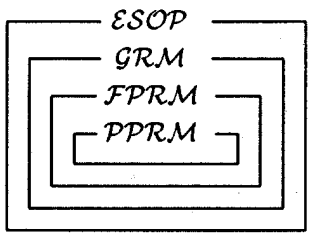

Fig. 4. Relations among various classes of AND-EXOR expressions.

2.2 Complexities of PPRMs, FPRMs, GRMs, and ESOPs

Table 2 compares the number of functions requiring a given number of products in the minimum expressions for $n = 4$. Note that PPRMs require, on the average, 8.00 products to realize an arbitrary function, while GRMs require only 3.68 products. This table also shows that, on the average, GRMs require fewer products than SOPs. Table 1 shows the number of products to represent various functions. Table 3 shows the number of products to represent arithmetic functions [4]. GRMs efficiently realize arithmetic functions. Except for sym9, GRMs require fewer products than SOPs. ESOPs usually require fewer products than GRMs.

TABLE 3
NUMBER OF PRODUCTS TO REPRESENT ARITHMETIC FUNCTIONS

Data	PPRM	FPRM	GRM	ESOP	SOP
adr4	34	34	34	31	75
inc8	16	16	15	15	37
log8	253	193	105	96	123
mlp4	97	97	71	61	121
nrm4	216	185	96	69	120
rdm8	56	56	31	31	76
rot8	225	118	51	35	57
sqr8	168	168	121	112	178
sym9	210	173	126	51	84
wgt8	107	107	107	58	255

DEFINITION 4. *The number of products in a minimum GRM for f is denoted by $\tau(GRM : f)$. The largest number of products to realize function of n variables by a minimum GRM is denoted by $\tau(GRM : n)$. The average number of products to realize n-variable functions by minimum GRMs is denoted by $\eta(GRM : n)$. Similar notations are used for other classes of expressions.*

LEMMA 3. *The number of n-variable functions requiring t products in PPRMs is* $\binom{2^n}{t}$.

PROOF. An arbitrary function f can be represented as (4). Because PPRM is a canonical representation, for each set of coefficients $(a_0, a_1, a_2, \ldots, a_{12\cdots n})$, there exists a unique function. There are 2^n as, and each can be either 0 or 1. Thus, the number of functions having t nonzero coefficients is $\binom{2^n}{t}$. Hence, we have the lemma. □

THEOREM 2. $\eta(PPRM : n) = 2^{n-1}$.

PROOF. An arbitrary function of n variables can be written as (4). The average is

$$\eta(PPRM : n) = \frac{1}{2^{2^n}} \sum_{t=0}^{2^n} t \cdot (\#\text{ of functions requiring } t \text{ products}) = \frac{1}{2^{2^n}} \sum_{t=0}^{2^n} t \cdot \binom{2^n}{t} = \frac{1}{2^{2^n}} 2^n 2^{2^n-1} = 2^{n-1}.$$

□

LEMMA 4. $\eta(GRM : n) \le 2 \cdot \eta(GRM : n-1)$.

PROOF. An arbitrary n-variable function can be expanded as $f = f_0 \oplus x_n f_2$, where f_0 and f_2 are functions of $(n-1)$ variables. Note that if F_0 and F_2 are GRMs, then $F_0 \oplus x_n \cdot F_2$ is also a GRM. Also $\tau(GRM : f) \le \tau(GRM : f_0) + \tau(GRM : f_2)$. Let $\mathcal{F}_n$ be the set of all the n-variable functions.

$$\begin{aligned}\eta(GRM : n) &= \frac{1}{2^{2^n}} \sum_{f \in \mathcal{F}_n} \tau(GRM : f) \\ &\le \frac{1}{2^{2^n}} \sum_{f \in \mathcal{F}_n} \{\tau(GRM : f_0) + \tau(GRM : f_2)\} \\ &= \frac{1}{2^{2^n}} \cdot 2^{2^n} \{\eta(GRM : n-1) + \eta(GRM : n-1)\} \\ &= 2 \cdot \eta(GRM : n-1).\end{aligned}$$

□

THEOREM 3. $\eta(GRM : n) \le (3.68) \cdot 2^{n-4}$, *when* $n \ge 4$.

PROOF. From Lemma 4, we have $\eta(GRM : n) \le 2^{n-4}\eta(GRM : 4)$. From Table 2, we have $\eta(GRM : 4) = 3.68$. Hence, the theorem. □

COROLLARY 1.

$$\frac{\eta(GRM : n)}{\eta(PPRM : n)} \le \frac{3.68}{8.00} = 0.46.$$

The above corollary shows that GRMs require, on the average, at most half of the products required for PPRMs.

3 EASILY TESTABLE REALIZATION FOR GRMS

3.1 Proposed Scheme

The proposed scheme consists of four parts: the literal part, the AND part, the EXOR part, and the check part, as shown in Fig. 3. The literal part has a control input c. During the normal operation, the control input c is set to one, and the literal part produces the positive (x_i) and the negative ($\bar{x}_i$) literals. During the test for the AND part, c is set to zero, and all the literal lines produce positive literals. The AND part consists of AND gates. In order to make the argument simple, we will use an AND gate even if it has only one input. Such an AND gate can be deleted without changing the function. The check part consists of two AND gates and two OR gates with extra observable outputs. This part is used to test the literal part. The EXOR part realizes a parity function. We can use EXOR tree for high speed operation. We assume that "only permanent stuck-at-0 (s-a-0) or stuck-at-1 (s-a-1) faults occur in at

most one part: either the literal part, the AND part, the EXOR part, or the check part. Multiple-faults may occur in each part."

3.2 Test for the EXOR Part

We will start with the test of the EXOR part, since this is the most important feature of the method. Although various test methods are known for the linear (EXOR only) networks [8], [22], most of them are inapplicable to our scheme. In this paper, we adopt Fujiwara's fault assumption [5] in the EXOR part: The faults change the function into a different linear function. This assumption is valid if the EXOR part is realized with EXOR gates, and only stuck-at-faults occur, in the inputs or the outputs.

THEOREM 4 [5]. *For an EXOR network realizing a parity function* $f = x_1 \oplus x_2 \oplus \cdots \oplus x_s$, $\{\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_s\}$ *is a test, where*

$$\begin{array}{rcl} & & (\ x_1,\ x_2,\ \ldots,\ x_s\ ) \\ \boldsymbol{a}_0 & = & (\ 0,\ 0,\ \ldots,\ 0\ ), \\ \boldsymbol{a}_1 & = & (\ 1,\ 0,\ \ldots,\ 0\ ), \\ \boldsymbol{a}_2 & = & (\ 0,\ 1,\ \ldots,\ 0\ ), \\ & & \ldots\ldots\ \ \ldots\ldots\ \ \ldots\ldots \\ \boldsymbol{a}_s & = & (\ 0,\ 0,\ \ldots,\ 1\ ). \end{array}$$

For example, for the network realizing $f = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, the test is {(0, 0, 0, 0), (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)}. When the EXOR part is realized with EXOR gates and only stuck-at-faults (s-a-faults) occur, all the multiple faults are detected by this test. Note that Reddy's method [12] cannot detect multiple faults.

Although Fujiwara's method is simple, it is not directly applicable to our scheme. For example, consider the network shown in Fig. 5. We cannot apply such inputs to the EXOR part, since AND gates exist between the input terminals and the EXOR part. From here, we will extend Fujiwara's result. Before going to the theorem, it is convenient to introduce the terminologies and properties of binary matrices by using simple examples.

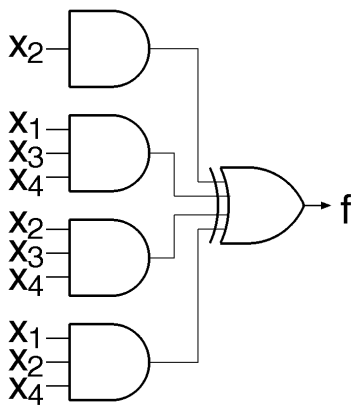

Fig. 5. Test for the EXOR part.

EXAMPLE 2. Consider a binary matrix:

$$M_1 = \begin{bmatrix} 110 \\ 011 \\ 101 \end{bmatrix}.$$

M_1 is **singular**, since mod-2 sum of the first and the second rows (columns) is equal to the third row (column).

$$M_2 = \begin{bmatrix} 101 \\ 110 \\ 010 \end{bmatrix}$$

is **nonsingular**, since no rows (columns) can be represented as a mod-2 sum of other rows (columns). In such a case, the row (column) vectors are **linearly independent**. Let,

$$M_2^{-1} = \begin{bmatrix} 011 \\ 001 \\ 111 \end{bmatrix}.$$

Note that

$$M_2 M_2^{-1} = \begin{bmatrix} 101 \\ 110 \\ 010 \end{bmatrix} \cdot \begin{bmatrix} 011 \\ 001 \\ 111 \end{bmatrix} = \begin{bmatrix} 100 \\ 010 \\ 001 \end{bmatrix} = I,$$

where I is a **unit matrix** and M_2^{-1} is called the **inverse** of M_2. Here, multiplication is AND and addition is mod-2. A matrix A is nonsingular iff the row (column) vectors are linearly independent. The **determinant** of a nonsingular matrix A is nonzero iff A has the inverse.

THEOREM 5. *For an EXOR network of s variables,* $\{\boldsymbol{a}_0, \boldsymbol{a}_1, a_2, \ldots, \boldsymbol{a}_s\}$ *is a test if A is nonsingular, where* $\boldsymbol{a}_0 = (0, 0, \ldots, 0)$,

$$A = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_s \end{bmatrix},$$

and $\boldsymbol{a}_i$ $(i = 0, 1, \ldots, s)$ *are binary vectors with s components.*

PROOF. By the fault assumption, the faulty network realizes a linear function of the form: $g = c_0 \oplus c_1 x_1 \oplus c_2 x_2 \oplus \cdots \oplus c_s x_s$. By applying $\boldsymbol{a}_0 = (0, 0, \ldots, 0)$, we can see the value of c_0. Let b_i be the outputs of the faulty network for the test $\boldsymbol{a}_i$ $(i = 1, 2, \ldots, s)$. Then, we have

$$A \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{bmatrix}.$$

Because A is nonsingular, A^{-1} exists, and we have

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{bmatrix} = A^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{bmatrix}.$$

This implies that we can identify all the coefficients $\{c_1, c_2, \ldots, c_s\}$ by the test $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_s\}$. □

Thus, the test for the EXOR part need not be the unit vectors. The set of s vectors that are linearly independent can be used for the test. The next problem is how to find such a set of vectors. Before showing the main theorem, we need to prove two lemmas.

DEFINITION 5. *Let* $\boldsymbol{a} = (a_1, a_2, \ldots, a_k)$ *and* $\boldsymbol{b} = (b_1, b_2, \ldots, b_k)$ *be binary vectors.* $\boldsymbol{a} \le \boldsymbol{b}$ *if* $a_i \le b_i$ *for* $i = 1, 2, \ldots, k$.

EXAMPLE 3. $(0, 1, 0) \le (1, 1, 0) \le (1, 1, 1)$.

LEMMA 5. *Let* $\boldsymbol{v}(k)$ *be an n-bit binary vector representing an integer k. Let* $\boldsymbol{L}(k)$ $(k = 1, 2, \ldots, 2^n - 1)$ *be binary vectors with* $(2^n - 1)$ *bits, where the* jth *element is one if* $\boldsymbol{v}(j) \le \boldsymbol{v}(k)$, *and zero otherwise. Then, the* $(2^n - 1) \times (2^n - 1)$ *matrix*

$$A = \begin{bmatrix} \boldsymbol{L}(1) \\ \boldsymbol{L}(2) \\ \vdots \\ \boldsymbol{L}\left(2^n - 1\right) \end{bmatrix}$$

is nonsingular.

PROOF. It is sufficient to show that the vectors $\boldsymbol{L}(1)$, $\boldsymbol{L}(2)$, …, and $\boldsymbol{L}(2^n - 1)$ are linearly independent. Suppose that these vectors are linearly dependent, then there exits a binary vector $(t_1, t_2, \ldots, t_{2^n - 1})$ that makes the equation

$$t_1\boldsymbol{L}(1) \oplus t_2\boldsymbol{L}(2) \oplus \cdots \oplus t_{2^n-1}\boldsymbol{L}\left(2^n - 1\right) = 0$$

true. Let t_k be the nonzero with maximum k. Thus,

$$\boldsymbol{L}(k) = t_1\boldsymbol{L}(1) \oplus \cdots \oplus t_{k-1}\boldsymbol{L}(k-1). \quad (5)$$

Since k is the maximum, the kth bit in $\boldsymbol{L}(k)$ is one and the corresponding bits in $\boldsymbol{L}(1)$, $\boldsymbol{L}(2)$, …, and $\boldsymbol{L}(k-1)$ are all zeros. However, this contradicts (5). Thus, we can conclude that $\boldsymbol{L}(1)$, $\boldsymbol{L}(2)$, …, and $\boldsymbol{L}(2^n - 1)$ are linearly independent. □

EXAMPLE 4. For $n = 3$, the vectors $\boldsymbol{L}(k)$ ($k = 1, 2, \ldots, 7$) defined in Lemma 5 are:

	001	010	011	100	101	110	111	
$\boldsymbol{L}(1)$ = (	1,	0,	0,	0,	0,	0,	0	),
$\boldsymbol{L}(2)$ = (	0,	1,	0,	0,	0,	0,	0	),
$\boldsymbol{L}(3)$ = (	1,	1,	1,	0,	0,	0,	0	),
$\boldsymbol{L}(4)$ = (	0,	0,	0,	1,	0,	0,	0	),
$\boldsymbol{L}(5)$ = (	1,	0,	0,	1,	1,	0,	0	),
$\boldsymbol{L}(6)$ = (	0,	1,	0,	1,	0,	1,	0	), and
$\boldsymbol{L}(7)$ = (	1,	1,	1,	1,	1,	1,	1	).

Thus, the matrix A is

$$A = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 1&1&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 1&0&0&1&1&0&0 \\ 0&1&0&1&0&1&0 \\ 1&1&1&1&1&1&1 \end{bmatrix}.$$

This is a triangular matrix whose diagonal elements are all ones. Thus, its determinant is 1, and A is nonsingular.

LEMMA 6. *The matrix that is obtained by recursively deleting the* kth *row and* kth *column from the matrix A defined in Lemma 5 is nonsingular.*

PROOF. As shown in Example 4, the matrix A defined in Lemma 5 is a triangular matrix. The matrix A_1 which can be obtained by deleting the kth row and kth column from A is also triangular and its diagonal elements are all ones. Thus, the determinant of A_1 is nonzero. Therefore, A_1 is nonsingular. □

THEOREM 6. *In an AND-EXOR network realizing a PPRM of n variables, for each product* p_i ($i = 1, 2, \ldots, s$), *consider a vector* $\boldsymbol{b}_i = (b_1, b_2, \ldots, b_n)$, *where* $b_j = 1$ *if the literal* x_j *appears in* p_i *and* $b_j = 0$ *otherwise* ($j = 1, 2, \ldots, n$). *Then,* $\{\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_s\}$ *is a test for the EXOR part, where* $\boldsymbol{b}_0 = (0, 0, \ldots, 0)$.

PROOF. For the input $\boldsymbol{b}_i$, let $\boldsymbol{a}_i = (a_1, a_2, \ldots, a_s)$ be the output vector of the AND part ($i = 1, 2, \ldots, s$). Note that the matrix

$$A = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_s \end{bmatrix}$$

is obtained by recursively deleting the kth row and kth column from the matrix defined in Lemma 5. Matrix A is nonsingular. By Theorem 5, we can see that $\{\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_s\}$ is a test for the EXOR part. □

EXAMPLE 5. Consider the PPRM realization in Fig. 5. The EXOR part can be tested by

	(	x_1,	x_2,	x_3,	x_4	)
$\boldsymbol{b}_0$	= (	0,	0,	0,	0	),
$\boldsymbol{b}_1$	= (	0,	1,	0,	0	),
$\boldsymbol{b}_2$	= (	1,	0,	1,	1	),
$\boldsymbol{b}_3$	= (	0,	1,	1,	1	),
$\boldsymbol{b}_4$	= (	1,	1,	0,	1	).

The output vectors of the AND part are

$\boldsymbol{a}_0$	= (0	0	0	0),	
$\boldsymbol{a}_1$	= (1	0	0	0),	
$\boldsymbol{a}_2$	= (0	1	0	0),	
$\boldsymbol{a}_3$	= (1	0	1	0),	*and*
$\boldsymbol{a}_4$	= (1	0	0	1).	

It is clear that $[\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \boldsymbol{a}_4]^t$ is nonsingular. Thus, $\{\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_4\}$ is a test for the EXOR part.

3.3 Test for the AND Part

We will use Saluja-Reddy's theorem to test the AND part [13].

DEFINITION 6. $\lfloor x \rfloor$ *denotes the integer part of x. The 0-weight of a vector is the number of zeros in the vector.*

THEOREM 7. *In a PPRM realization, suppose that the EXOR part and the literal part are fault free. Then, any t s-a-faults in the inputs or outputs of AND gates can be detected by applying all input vectors having 0-weight less than or equal to* $\lfloor \log_2 2t \rfloor$.

The proof can be found in [13].

EXAMPLE 6. Consider the PPRM realization of a four-variable function. Any two or fewer s-a-faults in the AND array can be detected by the test {(0, 0, 1, 1), (0, 1, 0, 1), (0, 1, 1, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 1, 0, 0), (0, 1, 1, 1), (1, 0, 1, 1), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1)}.

In order to test the AND part, the control input is set to zero. This will make the network realize a PPRM instead of the given GRM. When the literal part and the EXOR part are fault free, we can test the AND part by using Theorem 7.

3.4 Test for the Literal Part

To test the literal part, we use two pairs of extra AND and OR gates, and four extra observable outputs, as shown in Fig. 3. All the literal lines for x_i also connect to the AND gate A and to the OR gate A. All the literal lines for $x_i^{\bar{c}}$ also connect to the AND gate B and to the OR gate B. S-a-0 faults in the literal lines for x_i or $x_i^{\bar{c}}$ are detected as follows: Set $(c, x_1, x_2, \ldots, x_n) = (0, 1, 1, \ldots, 1)$. When the literal part is fault-free, all the literal lines become one. S-a-0 faults can be detected by the extra AND gates. S-a-1 faults in the literal lines x_i are detected as follows: Set $(c, x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$, and the OR gate A produces 1 if the line has an s-a-1 fault. S-a-1 faults in the literal lines $x_i^{\bar{c}}$ are detected as follows: Set $(c, x_1, x_2, \ldots, x_n) = (1, 1, \ldots, 1)$, and the OR gate B produces 1 if the line has an s-a-1 fault. The set $\{(0, 1, 1, \ldots, 1), (0, 0, 0, \ldots, 0), (1, 1, \ldots, 1)\}$ is also a test for the EXOR gates in the literal part.

3.5 Test for the Check Part

To test the s-a-1 fault in an input line of the extra AND gates, set it to 0, and set other lines to 1. To test the s-a-0 fault in the AND gates, set all the inputs to 1. To test the s-a-0 fault in an input of the OR gates, set it to 1, and set other inputs to 0. To test the s-a-1 fault in the OR gates, set all the inputs to 0.

3.6 Size of the Test

In this part, we will consider the total number of the tests for Fig. 3.

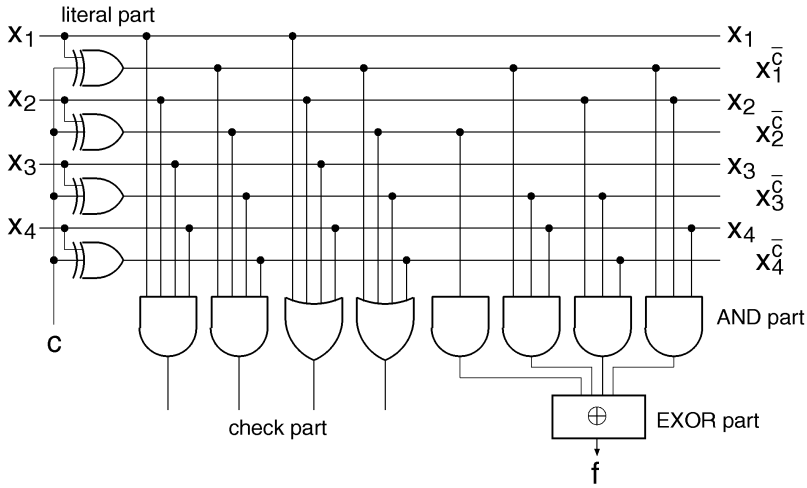

Fig. 6. An example of testable realizations.

TABLE 4
COMPARISON OF EASILY TESTABLE AND-EXOR NETWORKS

	Type	Fault Assumption	Linear Part	Test Length	Function Independence
Reddy IEEE TC 1972	PPRM	Single Stuck-at	Cascade	$n + 4$	Yes
Saluja-Reddy IEEE TC 1975	PPRM	Multiple Stuck-at	Cascade	$4 + \sum_{i=1}^{r} \binom{n}{i}$	Yes
Pradhan IEEE TC 1978	ESOP	Multiple Stuck-at	Cascade	$6 + 2n + \sum_{i=0}^{u} \binom{n}{i}$	Yes
T. Yamada IECE-Japan 1983	PPRM	Single Bridge	Cascade	$n + m + 5$	Yes
	ESOP	Single Bridge	Cascade	$2n + 2m + 7$	Yes
Sasao-Fujiwara IEICE-Japan 1987	ESOP	Single Stuck-at Cross point	Cascade	$2n + 4$	Yes
Sarabi-Perkowski DAC 1992	FPRM	Single Stuck-at	Cascade	$n + 4$	No
Perkowski-Csanky-Sarabi-Schäfer ICCD 1992	GRM	Single Stuck-at	Cascade	$n + 4$	No
Sasao This paper	GRM	Multiple Stuck-at	Tree	# of products + $n + 4 + \sum_{i=1}^{r} \binom{n}{i}$	No

$r = \lfloor \log_2 2t \rfloor$, t is the number of faults in the AND part.
u: maximum number of literals contained in any product term in the ESOP.
m: number of outputs.

1) Test for the EXOR part:
   $s + 1$, where s is the number of products in GRM.
2) Test for the AND part:

$$\sum_{i=0}^{r} \binom{n}{i},$$

   where $r = \lfloor log_2 2t \rfloor$, and t is the number of multiple faults to consider.
3) Test for the literal part: 3.
4) Test for the check part: $2n + 2$.

Because n identical tests appear both in 2 and 4, one identical test appears in 1, 3, and 4, also in 2 and 4, and some of the other tests may be identical, the total number of tests is at most

$$s + n + 4 + \sum_{i=1}^{r} \binom{n}{i}.$$

EXAMPLE 7. Consider the realization of GRM:

$$f = \bar{x}_2 \oplus \bar{x}_1\bar{x}_3x_4 \oplus x_2\bar{x}_3\bar{x}_4 \oplus \bar{x}_1x_2x_4$$

shown in Fig. 6.

1) Test for the EXOR part.

$$\begin{aligned}(c, x_1, x_2, x_3, x_4) = {} & (0, 0, 0, 0, 0),\\ & (0, 0, 1, 0, 0),\\ & (0, 1, 0, 1, 1),\\ & (0, 0, 1, 1, 1),\\ & (0, 1, 1, 0, 1).\end{aligned}$$

2) Test for the AND part for two or fewer s-a-faults.

$$(c, x_1, x_2, x_3, x_4) = (0, 1, 1, 1, 1), (0, 0, 1, 1, 1), (0, 1, 0, 1, 1), (0, 1, 1, 0, 1), (0, 1, 1, 1, 0), (0, 0, 0, 1, 1), (0, 0, 1, 0, 1), (0, 0, 1, 1, 0), (0, 1, 0, 0, 1), (0, 1, 0, 1, 0), (0, 1, 1, 0, 0).$$

3) Test for the literal part.

$$(c, x_1, x_2, x_3, x_4) = (0, 1, 1, 1, 1), (0, 0, 0, 0, 0), (1, 1, 1, 1, 1).$$

4) Test for the check part.

$$(c, x_1, x_2, x_3, x_4) = (0, 0, 0, 0, 0), (0, 1, 0, 0, 0), (0, 0, 1, 0, 0), (0, 0, 0, 1, 0), (0, 0, 0, 0, 1), (0, 1, 1, 1, 1), (0, 0, 1, 1, 1), (0, 1, 0, 1, 1), (0, 1, 1, 0, 1), (0, 1, 1, 1, 0).$$

Note that the first vector in 1 also appears in 3 and 4. The first vector in 2 also appears in 3 and 4. Four vectors with weight 3 appear both in 2 and 4. Thus, only 17 tests are necessary to test the network in Fig. 6.

4 Comparison with Other Methods

Table 4 summarizes the various testable AND-EXOR realizations. Note that all other methods use cascades in the EXOR part. Pradhan's method [10] uses ESOPs, which require fewer products than GRMs. However, the test length is excessive. When the number of the minterms of the function is odd, any ESOP for n-variable functions contains a product term of the form $x_1^* x_2^* \cdots x_n^*$ (minterm). This means the test is exhaustive. We can show that almost all functions require a test length near 2^n, when n is large. Yamada's method [23] also uses ESOPs. However, his method requires an additional EXOR cascade, five extra inputs, and one extra output. Also, his method cannot detect multiple faults. Sasao-Fujiwara's method [15] is for PLAs realizing ESOPs. Sarabi-Perkowski [14] extended Reddy's method to FPRMs. Perkowski-Csanky-Sarabi-Schäfer's method [9] uses GRMs. Their technique is to decompose a GRM into several FPRMs, and test the FPRMs by the method similar to Reddy's method.

The demerit of the present method is that the test depends on the functions realized. However, test generation is very easy when the GRMs are available. Also, the multiple faults in more than one part cannot be detected.

5 Conclusion and Comments

In this paper, we presented a design of easily testable AND-EXOR networks. The main features are:

1) The EXOR part can be tree, thus it is faster.
2) It uses GRMs instead of PPRMs, thus the number of the products is, on the average, less than half.
3) Multiple faults can be detected.

Although, it uses one extra input and four extra outputs, they can be accessed by the scan path technique [6].

ESOPs require fewer products than GRMs, but they cannot be used in this scheme. For example, consider the ESOP $x_1x_2x_3 \oplus \overline{x}_1\overline{x}_2\overline{x}_3$. This is not a GRM. Thus, it cannot be converted into a PPRM. So, we cannot test the EXOR part in the scheme. GRMs require a small number of products and have a good testable property similar to PPRMs.

After completion of this paper, Prof. Reddy informed the author that he developed a method to use a tree of EXOR gates instead of cascades [11]. His method requires $(m + 1)$ extra inputs and will have a fault detecting set independent of the function with cardinarity $2^m + n$, where m is an even number.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science, and Culture of Japan. Discussion with Prof. Fujiwara was very useful. Mr. Debatosh Debnath and Prof. J.T. Butler carefully reviewed the manuscript. Prof. Reddy sent me [11]. Mr. M. Matsuura prepared the LaTeX files. Preliminary version of this paper was presented at the IEEE Third Asian Test Symposium, November 15-17, 1994, Nara, Japan.

References

[1] L. Csanky, M.A. Perkowski, and I. Schäfer, "Canonical Restricted Mixed-Polarity Exclusive-OR Sums of Products and the Efficient Algorithm for Their Minimization," *IEE Proc.-E*, vol. 140, no. 1, pp. 69-77, Jan. 1993.

[2] T.R. Damarla and M. Karpovsky, "Fault Detection in Combinational Networks by Reed-Muller Transforms," *IEEE Trans. Computers*, vol. 38, no. 6, pp. 788-797, June 1989.

[3] M. Davio, J-P. Deschamps, and A. Thayse, *Discrete and Switching Functions*. McGraw-Hill Int'l, 1978.

[4] D. Debnath and T. Sasao "GRMIN: A Heuristic Simplification Algorithm for Generalized Reed-Muller Expressions," *IEE Proc. Computer Digital Technology*, vol. 143, no. 6, pp. 376-384, Nov. 1996.

[5] H. Fujiwara, "On Closedness and Test Complexity of Logic Circuits," *IEEE Trans. Computers*, vol. 30, no. 8, pp. 556-562, Aug. 1981.

[6] H. Fujiwara, *Logic Testing and Design for Testability*. MIT Press, 1985.

[7] D. Green, *Modern Logic Design*. Addison-Wesley, 1986.

[8] J.P. Hayes, "On Realizations of Boolean Functions Requiring a Minimal or Near Minimal Number of Test," *IEEE Trans. Computers*, vol. 20, no. 12, pp. 1,506-1,513, Dec. 1971.

[9] M.A. Perkowski, L. Csanky, A. Sarabi, and I. Schäfer, "Fast Minimization of Mixed-Polarity AND-XOR Canonical Networks," *Proc. ICCD-92*, pp. 33-36, Oct. 1992.

[10] D.K. Pradhan, "Universal Test Sets for Multiple Fault Detection in AND-EXOR Arrays," *IEEE Trans. Computers*, vol. 27, no. 2, pp. 181-187, Feb. 1978.

[11] S.M. Reddy, "Easily Testable Realization for Logic Functions," Technical Report no. 54, Univ. of Iowa, May 1972.

[12] S.M. Reddy, "Easily Testable Realizations for Logic Functions," *IEEE Trans. Computers*, vol. 21, no. 11, pp. 1,183-1,188, Nov. 1972.

[13] K.K. Saluja and S.M. Reddy, "Fault Detecting Test Sets for Reed-Muller Canonic Networks," *IEEE Trans. Computers*, vol. 24, no. 1, pp. 995-998, Oct. 1975.

[14] A. Sarabi and M.A. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed Polarity AND/XOR Canonical Networks," *Proc. Design Automation Conf. 1992*, pp. 20-35, June 1992.

[15] T. Sasao and H. Fujiwara, "A Design Method of AND-EXOR PLAs with Universal Tests," (in Japanese), Technical Report IECE Japan FTS86-25, Feb. 1987.

[16] T. Sasao and P. Besslich, "On the Complexity of MOD-2 Sum PLA's," *IEEE Trans. Computers*, vol. 39, no. 2, pp. 262-266, Feb. 1990.
[17] T. Sasao, "AND-EXOR Expressions and Their Optimization," *Logic Synthesis and Optimization*, T. Sasao, ed. 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. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 5, pp. 621-632, May 1993.
[19] T. Sasao, "An Exact Minimization of AND-EXOR Expressions Using BDDs," *Proc. IFIP 10.5 Workshop Application of the Reed-Muller Expansion in Circuit Design*, Sept. 1993.
[20] T. Sasao and D. Debnath, "Generalized Reed-Muller Expressions: Complexity and an Exact Minimization Algorithm," *IEICE Trans. Fundamentals*, vol. E78-A, no. 12, pp. 2,123-2,130, Dec. 1996.
[21] T. Sasao, "Representations of Logic Functions Using EXOR Operators," *Representation of Descrete Functions*, T. Sasao and M. Fujita, eds. Kluwer Academic Publishers, 1996.
[22] S.C. Seth and K.L. Kodandapani, "Diagnosis of Faults in Linear Tree Networks," *IEEE Trans. Computers*, vol. 26, no. 1, pp. 29-33, Jan. 1977.
[23] T. Yamada, "Easily Testable AND-XOR Combinational Logic Circuits," (in Japanese) *Trans. IECE*, vol. J.66-D, no. 1, pp. 105-110, Jan. 1983.