IEICE TRANS. INF. & SYST., VOL.E82–D, NO.9 SEPTEMBER 1999

PAPER

Easily Testable Realization Based on Single-Rail-Input OR-AND-EXOR Expressions

Takashi HIRAYAMA[†*], *Student Member*, **Goro KODA**[†], **Yasuaki NISHITANI**[†], *and* **Kensuke SHIMIZU**[†], *Members*

SUMMARY It is known that AND-EXOR two-level networks obtained by AND-EXOR expressions with positive literals are easily testable. They are based on the single-rail-input logic, and require $(n + 4)$ tests to detect their single stuck-at faults, where n is the number of the input variables. We present three-level networks obtained from single-rail-input OR-AND-EXOR expressions and propose a more easily testable realization than the AND-EXOR networks. The realization is an OR-AND-EXOR network which limits the fan-in of the AND and OR gates to n/r and r respectively, where r is a constant $(1 \leq r \leq n)$. We show that only $(r + n/r)$ tests are required to detect the single stuck-at faults by adding r extra variables to the network.

key words: *logic synthesis, exclusive-or, single stuck-at fault, easily testable realization*

1. Introduction

With the increasing complexity of VLSI circuitry, it is important to reduce the costs related to the testing process. Recently design techniques that are easily testable have been widely used to reduce the time for testing [1].

One of the classic realizations for testability is positive polarity Reed-Muller expression (PPRM) based networks. A PPRM is the exclusive-or (EXOR) of product terms with positive literals, and the network which realizes a PPRM results in a two-level AND-EXOR network. Reddy [6] showed that the single stuck-at faults in a PPRM network can be detected by $(n+4)$ tests that are independent of the functions realized, where n is the number of input variables. An extension to multiple stuck-at faults of PPRM networks was given in [7].

Afterward, a number of easily testable realizations with EXOR gates have been proposed. In Sarabi-Perkowski [8]'s method, single stuck-at faults can be detected by the tests similar to Reddy's. Yamada [12] presented a method to detect single stuck-at and bridging faults. Pradham [5] and Sasao [11] dealt with multiple stuck-at faults. These easily testable realizations require fewer gates than the PPRM realization by using both positive and negative literals. Every realization, however, requires $O(n)$ or more tests to detect the faults.

Manuscript received August 6, 1998.
Manuscript revised February 1, 1999.

[†]The authors are with the Faculty of Engineering, Gunma University, Kiryu-shi, 376–8515 Japan.
[*]Presently, with Ashikaga Institute of Technology.

We propose an easily testable realization based on *Single-rail-input Or-And-Exor expressions* (SOAEs), which consist of positive literals as well as PPRMs. Single-rail-input networks obtained from the realization can be easily implemented by MOS networks. In this paper, we first give an extension of PPRMs to the multiple-valued-input logic. Multiple-valued-input literals in the extension can be realized by sum terms with positive literals. Then, we present an expansion method to obtain SOAEs. SOAEs are realized by three-level OR-AND-EXOR networks with fan-in limitations such that every OR gate has r or less inputs and every AND gate has n/r or less inputs, where r is a constant $(1 \leq r \leq n)$. These three-level networks are called SOAE networks. Our goal is to reduce tests rather than to reduce gates. We finally show that all the single stuck-at faults in an SOAE network can be detected by $(r + n/r)$ tests by adding r extra variables to the network. If $n = r^2$ holds, the number of tests for this realization will be minimum, that is, $2\sqrt{n}$.

2. Preliminaries

An arbitrary logic function can be represented by the positive polarity Reed-Muller expression (PPRM). In this section, we give the definition of PPRMs and the testability of PPRM-based networks.

Definition 1: For a function f and a variable x, the subfunction of f restricted to $x = 0$ $(x = 1)$ is denoted by $f_{\bar{x}}$ (f_x). □

The following theorem is the basis of the expansion with EXORs [11].

Theorem 1 (expansion theorem): A logic function f can be expanded as follows.

$$\begin{array}{lcll} f & = & \bar{x}_1 f_{\bar{x}_1} \oplus x_1 f_{x_1} & \text{Shannon} \\ f & = & f_{\bar{x}_1} \oplus x_1 (f_{\bar{x}_1} \oplus f_{x_1}) & \text{positive Davio} \\ f & = & f_{x_1} \oplus \bar{x}_1 (f_{\bar{x}_1} \oplus f_{x_1}) & \text{negative Davio} \end{array}$$

□

The expression of a function f obtained by applying the positive Davio expansion recursively is called the *Positive Polarity Reed-Muller expression* (PPRM). (Similarly the expression obtained by applying the negative Davio expansion recursively is called the *Negative Polarity Reed-Muller expression* (NPRM).) A PPRM

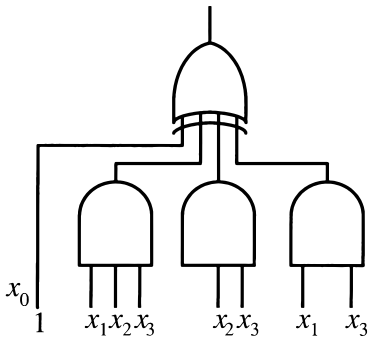


Fig. 1 PPRM network.

can be realized by the two-level AND-EXOR network, which is called the PPRM network.

Example 1: Figure 1 shows the PPRM network of $f = x_1x_2x_3 \oplus x_2x_3 \oplus x_1x_3 \oplus 1$. The constant input (CI) line is labeled x_0. □

This paper deals with the single stuck-at fault model because it covers a wide range of possible faults and has been most widely studied [1].

Definition 2: A *stuck-at fault* is the logical fault such that a signal line is stuck at a fixed logic value v ($v \in B, B = \{0,1\}$). Being stuck at v is denoted by s-a-v. We assume that at most one stuck-at fault occurs on the lines in a network. Input vectors to test whether the network is faulty are called *test vectors* (or *tests* shortly). □

It is well known that PPRM networks are easily testable. A PPRM network consists of AND gates and an EXOR gate; the input and output lines of AND gates are called AND part, and those of the EXOR gate are called EXOR part. In a PPRM network which realizes an n-variable function, s-a-1 faults in the AND part are detected by n tests, and s-a-0 faults in the AND part and stuck-at (s-a-0 and s-a-1) faults in the EXOR part are detected by 4 tests. The network thus requires only $(n+4)$ tests to detect all the single stuck-at faults [6]. The tests are independent of the function. In the following, we review the tests for a PPRM network. In [6], an EXOR gate that has more than 2 input lines is realized by the cascade of 2-input EXOR gates (Fig. 9) and the tests for the EXOR part are given as follows.

Test Set 1: (for s-a-0 faults in the AND part and stuck-at faults in the EXOR part of a PPRM network)

- $x_0 \leftarrow 0$, the other x_i's $\leftarrow 0$.
- $x_0 \leftarrow 1$, the other x_i's $\leftarrow 0$.
- $x_0 \leftarrow 0$, the other x_i's $\leftarrow 1$.
- $x_0 \leftarrow 1$, the other x_i's $\leftarrow 1$.

The third and fourth tests in Test Set 1 also detect s-a-0 faults in the AND part.

S-a-1 faults in the AND part are detected by the following Test Set 2.

Test Set 2: (for s-a-1 faults in the AND part of a PPRM network)

Generate the following tests for each k ($1 \le k \le n$), where d denotes a don't-care.

- $x_0 \leftarrow d$, $x_k \leftarrow 0$, the other x_i's $\leftarrow 1$.

Example 2: The tests for a PPRM network illustrated by Fig. 1 are given. Test Set 1 has 4 tests: $\{(x_0,x_1,x_2,x_3)\} = \{(0,0,0,0),(1,0,0,0),(0,1,1,1),(1,1,1,1)\}$. And Test Set 2 has n tests: $\{(x_0,x_1,x_2,x_3)\} = \{(d,0,1,1),(d,1,0,1),(d,1,1,0)\}$. From the assumption that only one line is faulty in a network, all the single stuck-at faults in Fig. 1 can be detected by the above $(n+4)$ tests. □

3. Extension of Testability to Multiple-Valued-Input Functions

In this section, the easily testable realization of PPRMs is extended to that of multiple-valued-input binary-output functions [9], [10]. This extension will be applied to the three-level networks in Sect. 4.

Definition 3: Let $P = \{0,1,\cdots,p-1\}$, $p \ge 2$, and $B = \{0,1\}$. $f : P^n \to B$ is called a *p-valued-input binary-output function*, or a *p-valued-input function*, shortly. □

Definition 4: For a nonempty subset S of P, X^S is called a *literal* of X. The value of X^S is defined as follows.

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

□

In this paper, binary variables are written in lowercase letters. Literals $x^{\{0\}}$, $x^{\{1\}}$, and $x^{\{0,1\}}$ of a binary variable x represent ordinary literals $\bar{x}$, x, and 1, respectively.

Example 3: 4-valued-input literals $X^{\{1\}}$, $X^{\{1,2,3\}}$, and $X^{\{0,2\}}$ are represented by the truth tables (a), (b), and (c) in Fig. 2, respectively. □

Definition 5: For a given set of p literals X^{S_0}, X^{S_1}, $\cdots$, $X^{S_{p-1}}$, matrix $\boldsymbol{M} \in B^p \times B^p$ such that

$$\begin{bmatrix} X^{S_0} \\ X^{S_1} \\ \vdots \\ X^{S_{p-1}} \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} X^{\{0\}} \\ X^{\{1\}} \\ \vdots \\ X^{\{p-1\}} \end{bmatrix}$$

is called the *characteristic matrix*, where the sum operation is EXOR. Each element m_{ij} of M is obtained as

X	$X^{\{1\}}$
0	0
1	1
2	0
3	0

(a)

X	$X^{\{1,2,3\}}$
0	0
1	1
2	1
3	1

(b)

X	$X^{\{0,2\}}$
0	1
1	0
2	1
3	0

(c)

Fig. 2 Examples of multiple-valued-input literals.

$$m_{ij} = \begin{cases} 0 & (j \notin S_i) \\ 1 & (j \in S_i) \end{cases} .$$

□

Lemma 1 and Theorem 2 are expansion methods for p-valued-input functions [10].

Lemma 1: A p-valued-input function $f(X_1, X_2, \cdots, X_n)$ can be expanded as

$$f = \bigoplus_{0 \le i \le p-1} X_1^{\{i\}} \cdot f_{X_1^{\{i\}}}, \tag{1}$$

where $f_{X_1^{\{i\}}}$ denotes $f(i, X_2, \cdots, X_n)$. □

Theorem 2 (expansion theorem [10]): A function $f : P^n \to B$ can be uniquely expanded in the form

$$f = \bigoplus_{0 \le i \le p-1} X^{S_i} \cdot h_i \tag{2}$$

if and only if the characteristic matrix $\boldsymbol{M}$ is non-singular. When $\boldsymbol{M}$ is non-singular, h_i is represented as

$$\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{p-1} \end{bmatrix} = {}^t(\boldsymbol{M}^{-1}) \begin{bmatrix} f_{X^{\{0\}}} \\ f_{X^{\{1\}}} \\ \vdots \\ f_{X^{\{p-1\}}} \end{bmatrix},$$

where ${}^t(\boldsymbol{M}^{-1})$ is the transpose of $\boldsymbol{M}^{-1}$. □

Equation (1) is the p-valued version of the Shannon expansion, and Eq. (2) is the general form of expansion. From Theorem 2, if S_0, S_1, $\cdots$, S_{p-1} and the non-singular matrix $\boldsymbol{M}$ are given, an expansion is determined. By applying the expansion recursively, the expression of f can be obtained.

Now, we restrict S_0, S_1, $\cdots$, S_{p-1} of Theorem 2 in order to obtain easily testable networks.

Definition 6: For a p-valued-input function, the expression obtained from the expansion defined by S_0, S_1, $\cdots$, S_{p-1} and the non-singular matrix $\boldsymbol{M}$ is called the (a,b)-*Restricted AND-EXOR expression* ((a,b)-RAE) if there exist two distinct constants a and b such that $a \notin S_i$ and $b \in S_i$ for any $S_i (\neq S_0)$. Let S_0 be $\{0, 1, \cdots, p-1\}$, i.e., literal X^{S_0} is the constant function 1. □

Example 4: An example of $p = 8$ is shown.

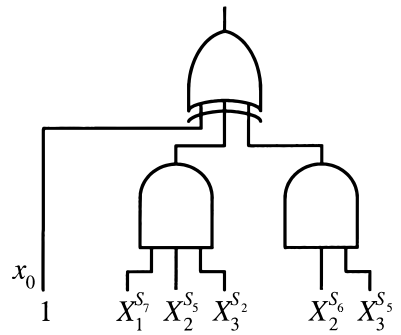


Fig. 3 (0,7)-RAE network of an 8-valued-input function with 3 variables.

$$\begin{cases} S_0 = \{0,1,2,3,4,5,6,7\} \\ S_1 = \{1,3,5,7\} \\ S_2 = \{2,3,6,7\} \\ S_3 = \{1,2,3,5,6,7\} \\ S_4 = \{4,5,6,7\} \\ S_5 = \{1,3,4,5,6,7\} \\ S_6 = \{2,3,4,5,6,7\} \\ S_7 = \{1,2,3,4,5,6,7\} \end{cases}$$

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\boldsymbol{M}$ is non-singular and each of $S_1, S_2, \cdots, S_7$ satisfies $0 \notin S_i$ and $7 \in S_i$. Given eight literals and the non-singular matrix $\boldsymbol{M}$, an arbitrary 8-valued-input function f can be expanded as Eq. (2). Therefore the $(0,7)$-RAE of f can be obtained.

$f = X_1^{S_7} X_2^{S_5} X_3^{S_2} \oplus X_1^{S_0} X_2^{S_6} X_3^{S_5} \oplus 1$ is an example of $(0,7)$-RAEs. The $(0,7)$-RAE network is realized as Fig. 3. The CI line is labeled x_0 because it is a binary-input line. Since the literal $X_i^{S_0}$ corresponds to the constant 1, lines leading from $X_i^{S_0}$ to AND gates can be omitted in (a,b)-RAE networks. □

As well as PPRM networks, (a,b)-RAE networks are easily testable. Since the value of every literal in (a,b)-RAEs is 0 for $X = a$ and 1 for $X = b$, the tests for (a,b)-RAE networks are obtained by substituting the value a for the value 0 and the value b for the value 1 of tests for PPRM networks (Test Set 1). Thus tests for (a,b)-RAE networks with n variables are written as follows.

Test Set 3: (for s-a-0 faults in the AND part and stuck-at faults in the EXOR part of an (a,b)-RAE network)

- $x_0 \leftarrow 0$, all X_i's $\leftarrow a$.
- $x_0 \leftarrow 1$, all X_i's $\leftarrow a$.

X_1	$X_1^{\{1\}}$
0	0
1	1
2	0
3	0

$=$

x_2	x_1	$x_1\bar{x}_2$
0	0	0
0	1	1
1	0	0
1	1	0

X_1	$X_1^{\{1,2,3\}}$
0	0
1	1
2	1
3	1

$=$

x_2	x_1	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 4 Realization of multiple-valued-input literals.

- $x_0 \leftarrow 0$, all X_i's $\leftarrow b$.
- $x_0 \leftarrow 1$, all X_i's $\leftarrow b$.

Test Set 4: (for s-a-1 faults in the AND part of an (a, b)-RAE network)

Generate the following tests for each k $(1 \leq k \leq n)$.

- $x_0 \leftarrow d$, $X_k \leftarrow a$, the other X_i's $\leftarrow b$.

From Test Set 3 and 4, an (a, b)-RAE network requires $(n + 4)$ tests.

4. Single-Rail-Input OR-AND-EXOR Expansion

By regarding 2^r-valued numbers as r-bit binary numbers, a 2^r-valued-input literal can be represented by a binary-input function with r variables. Then testability of (a, b)-RAEs can be also applied to easily testable realizations for binary-input functions.

Example 5: 4-valued-input literals $X_1^{\{1\}}$ and $X_1^{\{1,2,3\}}$ can be realized by $x_1\bar{x}_2$ and $x_1 \vee x_2$, respectively (Fig. 4), which are 2-variable functions. □

Let the binary representation of an integer i be $(i_r i_{r-1} \cdots i_1)_2$. A literal $X_1^{\{i\}}$ can be represented by the minterm $x_r^{\{i_r\}} x_{r-1}^{\{i_{r-1}\}} \cdots x_1^{\{i_1\}}$ because the value of $X_1^{\{i\}}$ is 1 only if $X_1 = i$. Hence,

$$\begin{bmatrix} X_1^{\{0\}} \\ X_1^{\{1\}} \\ \vdots \\ X_1^{\{2^r-1\}} \end{bmatrix} = \begin{bmatrix} \bar{x}_r \cdots \bar{x}_2 \bar{x}_1 \\ \bar{x}_r \cdots \bar{x}_2 x_1 \\ \vdots \\ x_r \cdots x_2 x_1 \end{bmatrix}$$

holds. From the above equation, Lemma 1 can be rewritten as the following corollary.

Corollary 1: A function f can be expanded with respect to variables $x_1, x_2, \cdots, x_r$ as follows.

$$f = {}^t\begin{bmatrix} \bar{x}_r \cdots \bar{x}_2 \bar{x}_1 \\ \bar{x}_r \cdots \bar{x}_2 x_1 \\ \vdots \\ x_r \cdots x_2 x_1 \end{bmatrix} \begin{bmatrix} f_{\bar{x}_r \cdots \bar{x}_2 \bar{x}_1} \\ f_{\bar{x}_r \cdots \bar{x}_2 x_1} \\ \vdots \\ f_{x_r \cdots x_2 x_1} \end{bmatrix}$$

□

Corollary 1 corresponds to an equation obtained by applying the Shannon expansion with respect to r variables recursively.

To realize easily testable networks, we present an expansion that generates OR-AND-EXOR expressions corresponding to $(0, 2^r - 1)$-RAEs. For r variables, there exist $2^r - 1$ sum terms consisting of positive literals. These sum terms are used as 2^r-valued-input literals $X_1^{S_1}, X_1^{S_2}, \cdots, X_1^{S_{2^r-1}}$, i.e.,

$$X_1^{S_i} = \bigvee_{k=1}^{r} i_k x_k, \tag{3}$$

where $(i_r i_{r-1} \cdots i_1)_2$ is the binary representation of i $(i \neq 0)$. $X_1^{S_0}$ is defined as the constant function 1. From Eq. (3), S_i is determined as the set of input values of the sum term whose output is 1. Then $X_1^{S_i} = 0$ for $X_1 = 0 = (00 \cdots 0)_2$ and $X_1^{S_i} = 1$ for $X_1 = 2^r - 1 = (11 \cdots 1)_2$ hold for all literals $X_1^{S_i}$ except $X_1^{S_0}$.

Example 6: If $r = 3$, 2^r literals are given as follows.

$$\begin{bmatrix} X_1^{S_0} \\ X_1^{S_1} \\ X_1^{S_2} \\ X_1^{S_3} \\ X_1^{S_4} \\ X_1^{S_5} \\ X_1^{S_6} \\ X_1^{S_7} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 \vee x_2 \\ x_3 \\ x_1 \vee x_3 \\ x_2 \vee x_3 \\ x_1 \vee x_2 \vee x_3 \end{bmatrix}$$

These literals realize as the same literals as Example 4. So, the matrix $\boldsymbol{M}$ for the above literals is written as Example 4. □

Since literals $X_1^{S_1}, X_1^{S_2}, \cdots, X_1^{S_{2^r-1}}$ are defined as sum terms, the characteristic matrix $\boldsymbol{M}$ is non-singular. Then we have the following expansion.

Theorem 3: Let $X_1^{S_0}$ be the constant function 1 and $X_1^{S_1}, \cdots, X_1^{S_{2^r-1}}$ be the sum terms defined by Eq. (3). An arbitrary function f can be expanded as

$$f = [X_1^{S_0}, X_1^{S_1}, \cdots, X_1^{S_{2^r-1}}] \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{2^r-1} \end{bmatrix}$$

$$= \bigoplus_{0 \leq i \leq 2^r-1} X_1^{S_i} \cdot h_i,$$

where h_i is represented with the characteristic matrix $\boldsymbol{M}$ as

$$\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{2^r-1} \end{bmatrix} = {}^t(\boldsymbol{M}^{-1}) \begin{bmatrix} f_{\bar{x}_r \cdots \bar{x}_2 \bar{x}_1} \\ f_{\bar{x}_r \cdots \bar{x}_2 x_1} \\ \vdots \\ f_{x_r \cdots x_2 x_1} \end{bmatrix}$$

□

We partition n variables $\{x_1, x_2, \cdots, x_n\}$ into n/r sets with r variables: $\{x_1, \cdots, x_r\}$, $\{x_{r+1}, \cdots, x_{2r}\}$, $\cdots$, $\{x_{n-r+1}, \cdots, x_n\}$, which can be regarded as 2^r-valued variables X_1, X_2, $\cdots$, $X_{n/r}$, respectively. For simplicity, we assume $n \bmod r = 0$ (or n is a multiple of r). Similar to Eq. (3), sum terms for these variable sets are generally represented by

$$X_{l+1}^{S_i} = \bigvee_{k=1}^{r} i_k x_{rl+k}, \tag{4}$$

where $0 \le l \le n/r - 1$. In Theorem 3, an n-variable function f is expanded into EXOR of $(n-r)$-variable functions h_i by using sum terms $X_1^{S_i}$ of $\{x_1, x_2, \cdots, x_r\}$. By applying the expansion to these subfunctions h_i with sum terms $X_2^{S_i}$ of $\{x_{r+1}, \cdots, x_{2r}\}$, EXOR of $(n-2r)$-variable functions can be obtained. In this way, by applying the expansion recursively until constant functions are reached, the *Single-rail-input Or-And-Exor expression* (SOAE) of f is finally obtained. This expression is canonical for the constant r because the subfunctions in Theorem 3 are uniquely defined. As well as the Davio expansion, the time complexity to obtain the SOAE of n-variable function f is $O(2^n)$ because the total number of recursions is at most 2^n.

There is another way to obtain the SOAE of f. It is obtained by applying de Morgan's theorem, $\bar{x}_{j_1}\bar{x}_{j_2}\cdots\bar{x}_{j_r} = (x_{j_1} \vee x_{j_2} \vee \cdots x_{j_r}) \oplus 1$, to every product term of the NPRM of f according to the variable partition.

Example 7: If $r = 2$, 4 variables $\{x_1, x_2, x_3, x_4\}$ are partitioned into $\{x_1, x_2\}$ and $\{x_3, x_4\}$. In this case, NPRM $f = \bar{x}_1\bar{x}_2 \cdot \bar{x}_3\bar{x}_4 \oplus \bar{x}_1 \cdot \bar{x}_3\bar{x}_4 \oplus \bar{x}_1\bar{x}_2 \oplus \bar{x}_2 \cdot \bar{x}_4 \oplus \bar{x}_1 \oplus \bar{x}_2 \oplus \bar{x}_4 \oplus 1$ is converted into the SOAE as follows.

$$\begin{aligned} f=&\{(x_1 \vee x_2) \oplus 1\}\{(x_3 \vee x_4) \oplus 1\} \\ &\oplus (x_1 \oplus 1)\{(x_3 \vee x_4) \oplus 1\} \oplus \{(x_1 \vee x_2) \oplus 1\} \\ &\oplus (x_2 \oplus 1)(x_4 \oplus 1) \oplus (x_1 \oplus 1) \oplus (x_2 \oplus 1) \\ &\oplus (x_4 \oplus 1) \oplus 1 \\ =&(x_1 \vee x_2)(x_3 \vee x_4) \oplus x_1(x_3 \vee x_4) \oplus x_2x_4 \end{aligned}$$

□

Networks realizing SOAEs are called SOAE networks, which form three-level OR-AND-EXOR networks with fan-in limitations such that every OR gate has r or less inputs and every AND gate has n/r or less inputs. This property is practical for logic synthesis because OR gates have different fan-in limitations from AND gates generally. The proper limitations can be set by r for the realization technology. In some special cases, SOAE networks may result in two-level; for example single-rail-input AND-EXOR networks (PPRM networks) are obtained if $r = 1$, and single-rail-input OR-EXOR networks are obtained if $r = n$.

Example 8: The 4-variable function $f = x_1x_2\bar{x}_3x_4 \vee x_1x_2x_3x_4 \vee \bar{x}_1x_2x_3\bar{x}_4$ is expanded as follows.

- If $r = 1$, $f = x_1x_2x_4 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3 \oplus x_2x_3x_4 \oplus x_2x_3$.
- If $r = 2$, $f = (x_1 \vee x_2)(x_3 \vee x_4) \oplus x_1(x_3 \vee x_4) \oplus x_2x_4$.
- If $r = 4$, $f = (x_1 \vee x_2 \vee x_3 \vee x_4) \oplus (x_1 \vee x_3 \vee x_4) \oplus (x_1 \vee x_2) \oplus (x_2 \vee x_4) \oplus x_1 \oplus x_2 \oplus x_4$. □

Since SOAE networks are based on the single-rail-input logic, they can be implemented by MOS gates easily. An EXOR gate can be also implemented by a cascade (see Sect. 6.3) of two-input MOS gates which realize $f = \overline{x \oplus y}$[4], and by assigning proper value to the CI line x_0. Although an SOAE network requires more gates than a PPRM network because of the fan-in limitations, the hardware cost between the two networks can be equal in the MOS gate implementation. Since 2^r subfunctions appear with respect to r variables in the expansion of Theorem 3, and also in the positive Davio expansion 2^r subfunctions appear with respect to r variables as well as Corollary 1, the number of AND gates in a PPRM network is equal to that of OR-AND subnetworks in an SOAE network on average. In MOS logic, an OR-AND-INVERT network with n inputs can be implemented by one MOS gate, whose cost is virtually equal to a NAND gate with same inputs. Therefore the hardware cost for an SOAE network is equal to a PPRM network on average.

5. Easily Testable Realization

Since SOAEs with n variables correspond to $(0, 2^r - 1)$-RAEs with n/r variables, all the single stuck-at faults in the AND and EXOR parts of SOAE networks are detected by the $(n/r + 4)$ tests of Test Set 3 and 4, where the value 0 and $(2^r - 1)$ in the tests are assigned to OR gates in binary numbers. However, not all the faults in the OR part are detected by these tests; every s-a-1 fault is detected while an s-a-0 fault may not be detected. To detect all the single s-a-0 faults in the OR part, additional tests are required.

Simply, all the single s-a-0 faults in the OR part of an SOAE network are detected by the following tests.

Test Set 5: (for s-a-0 faults in the OR part of a straightforward SOAE network)

Generate the following tests for each (l, k)
$(0 \le l \le n/r - 1,\ 1 \le k \le r)$

- $x_0 \leftarrow d$, $x_{rl+k} \leftarrow 1$, $x_{rl+j} \leftarrow 0$ $(1 \le j \le r, j \ne k)$, the other x_i's $\leftarrow 1$.

These tests detect s-a-0 faults on the input lines specified by k of the OR gate specified by l. Adding these n tests, the total number of tests is $(n + n/r + 4)$. It is slightly larger than the number of tests for PPRM networks, $(n + 4)$.

If r extra variables for the testing purpose are added to an SOAE network, the tests for s-a-0 faults in

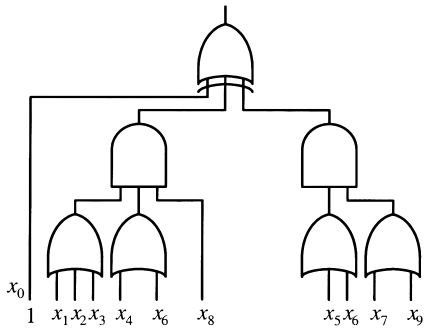


Fig. 5 Straightforward SOAE network.

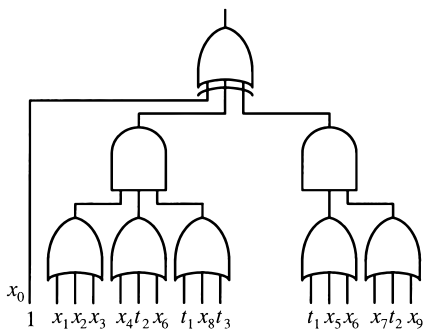


Fig. 6 Easily testable SOAE network.

the OR part can be reduced. The r variables are called *testing variables* and are denoted by $t_1, t_2, \cdots, t_r$. In the following, an easily testable realization for SOAEs with testing variables is described. The tests for the easily testable SOAE networks are given in Sect. 6.

From Eq. (4), sum terms in an SOAE are represented by $\bigvee_{k=1}^{r} i_k x_{rl+k} = (i_1 x_{rl+1} \vee i_2 x_{rl+2} \vee \cdots \vee i_r x_{rl+r})$ $(0 \leq l \leq n/r-1)$. To be easily testable, all the sum terms $\bigvee_{k=1}^{r} i_k x_{rl+k}$ in an SOAE are replaced with $\bigvee_{k=1}^{r}(i_k x_{rl+k} \vee \bar{i}_k t_k)$. This means that either x_{rl+k} or t_k is used as the input line of OR gates for each i_k, and that every OR gate has exactly r inputs.

Example 9: Figure 5 shows the straightforward SOAE network with $r = 3$ realizing $f = (x_1 \vee x_2 \vee x_3)(x_4 \vee x_6)x_8 \oplus (x_5 \vee x_6)(x_7 \vee x_9) \oplus 1$. The line x_0 represents the CI line. This example corresponds to the network of Fig. 3, whose literals are realized by OR gates in Fig. 5. On the other hand, Fig. 6 shows its easily testable realization. This network is obtained from the modified expression $f = (x_1 \vee x_2 \vee x_3)(x_4 \vee t_2 \vee x_6)(t_1 \vee x_8 \vee t_3) \oplus (t_1 \vee x_5 \vee x_6)(x_7 \vee t_2 \vee x_9) \oplus 1$. □

In normal operation, the easily testable networks obtained from the above modification are functionally equivalent to the straightforward networks if the value 0 is assigned to $t_1, t_2, \cdots, t_r$. The easily testable network require the CI line x_0 to test the primary output (PO) line. If an SOAE has the constant 1, the value 1 is assigned to the line x_0 in normal operation. If not, the value 0 is assigned.

6. Fault Detection for Easily Testable Networks

In this section, tests for the easily testable SOAE networks described in Sect. 5 are shown.

6.1 Tests for S-a-0 Faults in the AND and OR Parts

The following tests detect s-a-0 faults in the AND and OR parts. In the tests, testing variables $t_1, t_2, \cdots, t_r$ are fully exploited to detect the faults.

Test Set 6: (for s-a-0 faults in AND and OR parts of an easily testable SOAE network)

Generate the following tests for each k $(1 \leq k \leq r)$.

- $x_0 \leftarrow d$, $x_{rl+k} \leftarrow 1$ $(0 \leq l \leq n/r-1)$, the other x_i's $\leftarrow 0$, $t_k \leftarrow 1$, the other t_i's $\leftarrow 0$.

Every test in Test Set 6 assigns the value 1 to the AND part and detects all the single s-a-0 faults in the AND part as well as the third and fourth tests of Test Set 3. Also, every test sensitizes all the input lines of AND gates. For each k, the value 1 is assigned to exactly one input line of each OR gate, and the s-a-0 fault on the line is detected. By applying the test for all k, all the single s-a-0 faults in the OR part are also detected.

Example 10: S-a-0 faults in the AND and OR parts of the network given by Fig. 6 are detected by the tests $\{(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, t_1, t_2, t_3)\} = \{(d,1,0,0,1,0,0,1,0,0,1,0,0), (d,0,1,0,0,1,0,0,1,0,0,1,0), (d,0,0,1,0,0,1,0,0,1,0,0,1)\}$. Each of (a), (b), and (c) in Fig. 7 shows the lines to which the value 1 is assigned by each test, i.e., each test detects s-a-0 faults on these lines. □

To detect s-a-0 faults in the OR part, a straightforward SOAE network requires n tests of Test Set 5 while an easily testable SOAE network requires only r tests of Test Set 6 by using testing variables $t_1, t_2, \cdots, t_r$.

6.2 Tests for S-a-1 Faults in AND and OR Parts

The following tests detect s-a-1 faults in the AND and OR parts.

Test Set 7: (for s-a-1 faults in the AND and OR parts of an easily testable SOAE network)

Generate the following tests for each k $(0 \leq k \leq n/r-1)$.

- $x_0 \leftarrow d$, $x_{rk+l} \leftarrow 0$ $(1 \leq l \leq r)$, the other x_i's $\leftarrow 1$, all t_i's $\leftarrow 0$.

Except for assigning the value 0 to all t_i's, Test Set 7 is equivalent to Test Set 4, which is the tests for s-a-1 faults in the AND part of $(0, 2^r - 1)$-RAE

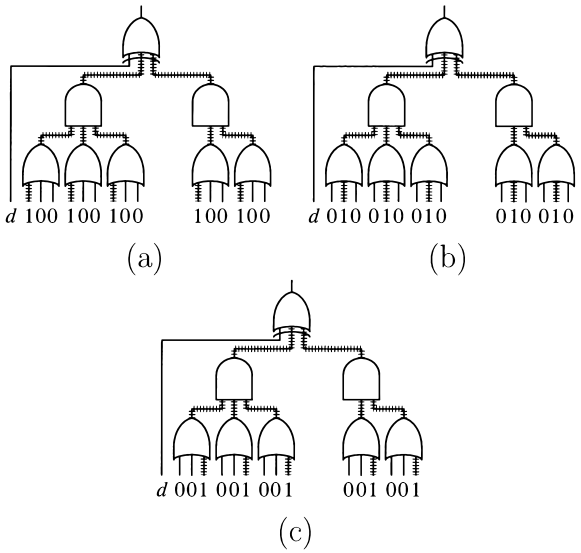


Fig. 7 Tests for s-a-0 faults in the AND and OR parts.

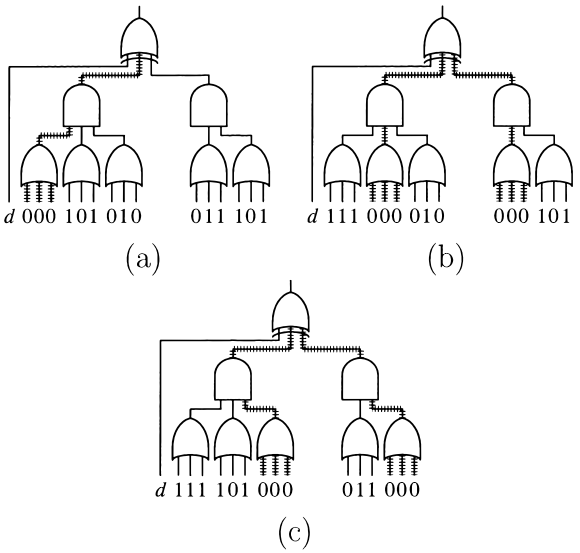


Fig. 8 Tests for s-a-1 faults in the AND and OR parts.

networks. Hence all the single s-a-1 faults in the AND part are detected. Note that assigning the value 0 to t_i's does not affect the outputs of OR gates since there are no OR gates consisting of only testing variables. In every AND gate, at most one input line specified by k is sensitized. In the OR gates connected to such lines, all input lines are assigned the value 0, and s-a-1 faults on those lines are detected. By applying the test for all k, all the single s-a-1 faults in the OR part are also detected.

Example 11: S-a-1 faults in the AND and OR parts of the network given by Fig. 6 are detected by the tests $\{(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, t_1, t_2, t_3)\} = \{(d, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0), (d, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0), (d, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)\}$. Each of (a), (b), and (c) in Fig. 8 shows the lines to which the value 0 is assigned by each test, i.e., each test detects s-a-1 faults on these lines. □

Test Set 6 and 7 are independent of the functions realized; they depend on the constant r.

6.3 Tests for Stuck-at Faults on the Primary Output and the Constant Input Lines

Stuck-at faults on the PO and the CI lines are not detected if the values assigned to these two lines are unchanged throughout Test Set 6 and 7. To avoid this, in Test Set 6, $x_0 \leftarrow 1$ is assigned in arbitrary i tests $(1 \leq i \leq r-1)$ and $x_0 \leftarrow 0$ is assigned in the rest of the tests. These tests change the values assigned to the PO and CI lines because every test in Test Set 6 assigns the value 1 to all the input lines of the EXOR gate except the CI line. Thus the above tests detect stuck-at faults on the PO and CI lines. Even if the EXOR gate is realized by the EXOR cascade [6](Fig. 9), the tests are

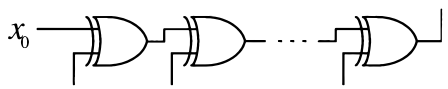


Fig. 9 EXOR cascade.

available, i.e., the tests detect stuck-at faults on the PO and CI lines and the interconnect lines between the 2-input EXOR gates.

6.4 Number of the Tests

From r tests of Test Set 6 and n/r tests of Test Set 7, we have the following.

Theorem 4: In the easily testable SOAE networks, all the single stuck-at faults are detected by $(r + n/r)$ tests. □

The number of tests $(r+n/r)$ is minimum when $r = \sqrt{n}$. Thus the following corollary holds.

Corollary 2: If $n = r^2$ holds, all the single stuck-at faults in the easily testable SOAE networks are detected by only $2\sqrt{n}$ tests. □

To detect all the single stuck-at faults, PPRM networks require $(n + 4)$ tests while the easily testable SOAE networks require smaller tests $2\sqrt{n}$.

6.5 Tests for the Primary Input Lines

In Test Set 6 and 7, the primary input (PI) lines are assumed to be fault-free. The case that the PI lines could be faulty is discussed in this section.

Tests in Test Set 6 and 7 may not detect a stuck-at fault on the PI lines because a faulty signal on the PI lines usually fans out to some input lines of the OR gates. A simple technique to detect such faults

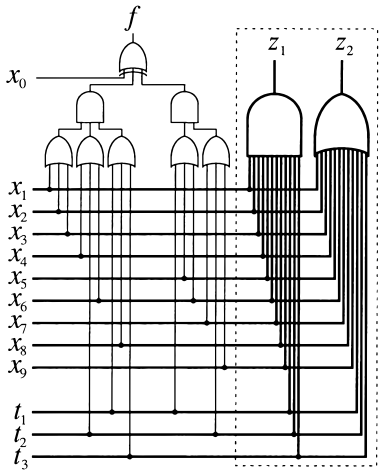


Fig. 10 Extra gates to detect faults on the PI lines (1).

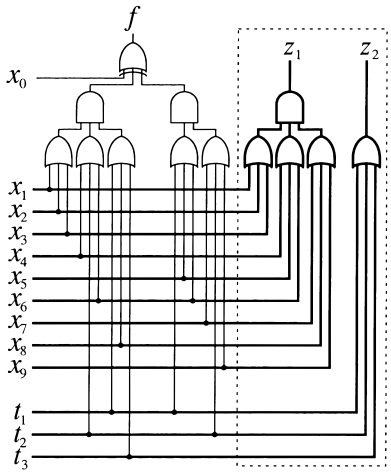


Fig. 11 Extra gates to detect faults on the PI lines (2).

is adding extra gates to the network [6], [11]. We show two methods below.

6.5.1 Extra Gates for Efficient Testing

In this method, an AND gate such that $x_1x_2\cdots x_n \cdot t_1t_2\cdots t_r$ and an OR gate such that $x_1 \vee x_2 \vee \cdots \vee x_n \vee t_1 \vee t_2 \vee \cdots \vee t_r$ are added (Fig. 10).

Then the following two tests are applied. Stuck-at faults on the PI lines are detected by observing the outputs z_1 and z_2.

Test Set 8:

- All x_i's and t_i's $\leftarrow 0$.
- All x_i's and t_i's $\leftarrow 1$.

This method has an advantage that the above tests also detect multiple stuck-at faults on the PI lines, whereas it has a disadvantage that these extra gates require larger fan-in, namely, $(n + r)$ inputs each. Although it is possible to realize an $(n + r)$-input gate by connecting small gates in order to satisfy the fan-in limitations, the following method requires smaller hardware costs under the fan-in limitations.

6.5.2 Extra Gates under the Fan-in Limitations

In this method, an OR gate such that $t_1 \vee t_2 \vee \cdots \vee t_r$ and an OR-AND subnetwork such that $(x_1 \vee x_2 \vee \cdots \vee x_r)(x_{r+1} \vee x_{r+2} \vee \cdots \vee x_{2r}) \cdots (x_{n-r+1} \vee x_{n-r+2} \vee \cdots \vee x_n)$ are added (Fig. 11).

In the method, stuck-at faults on the PI lines are detected through z_1 and z_2 by Test Set 6 and 7, i.e., this method requires no additional tests. Multiple stuck-at faults can not be detected.

7. Conclusions and Comments

In this paper, we presented an expansion method to obtain single-rail-input OR-AND-EXOR expressions (SOAEs). SOAEs are canonical for the given variable partition of expansion. An SOAE network forms the three-level OR-AND-EXOR network in which the fan-in of every OR gate is limited to r and the fan-in of every AND gate is limited to n/r, where r is a constant such that $1 \leq r \leq n$. Because of the single-rail input, those networks can be implemented by MOS networks easily. Although SOAE networks require more gates than PPRM networks, the hardware cost of the two kinds of networks could be equal in the MOS implementation. A straightforward SOAE network requires $(n + n/r + 4)$ tests to detect all the single stuck-at faults; the network, however, can be easily testable. We showed that an easily testable SOAE network requires only $(r + n/r)$ tests for the single stuck-at faults by adding r extra variables. This realization can be applied to an arbitrary function. The tests do not depend on the functions realized but on the constant r. If adding r extra primary inputs requires considerable hardware costs, the scan path technique [1] can be used instead.

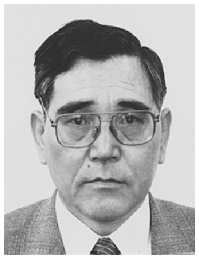
If SOAE networks are implemented by ordinary AND and OR gates, not MOS gates, the total number of OR gates is reduced by sharing the OR gates that realize the same sum terms [2]. In this case, By sharing the odd number of OR gates only, the testability for the single stuck-at faults is sustained because every OR gate has the odd number of fan-outs.

If $n = r^2$ holds, the most easily testable SOAE networks are obtained. In these networks, all the single stuck-at faults are detected by only $2\sqrt{n}$ tests. The former easily testable realizations require $O(n)$ or more tests to detect the faults. Although our realization requires $\sqrt{n}$ extra variables in this case, these are far fewer than the n main variables. The proposed realization has significantly reduced the number of tests by moderate additions of hardware.

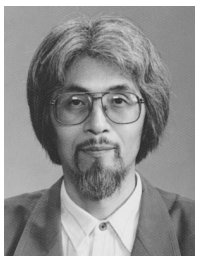
References

[1] M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital Systems Testing and Testable Design, revised printing, IEEE PRESS, New York, 1990.
[2] D. Debnath and T. Sasao, "Minimization of AND-OR-EXOR three-level networks with AND gate sharing," IEICE Trans. Inf. & Syst., vol.E80-D, no.10, pp.1001–1008, Oct. 1997.
[3] T. Hirayama, G. Koda, Y. Nishitani, and K. Shimizu, "Easily testable realization based on OR-AND-EXOR expansion with single rail inputs," Proc. IEEE APCCAS'98, pp.371–374, Nov. 1998.
[4] S. Muroga, VLSI System Design, John Wiley & Sons, Inc., 1982.
[5] D.K. Pradhan, "Universal test sets for multiple fault detection in AND-EXOR arrays," IEEE Trans. Comput., vol.C-27, no.2. pp.181–187, Feb. 1978.
[6] S.M. Reddy, "Easily testable realization for logic functions," IEEE Trans. Comput., vol.C-21, no.11. pp.1183–1188, Nov. 1972.
[7] K.K. Saluja and S.M. Reddy, "Fault detecting test sets for Reed-Muller canonic networks," IEEE Trans. Comput., vol.C-24, no.1. pp.995–998, Oct. 1975.
[8] A. Sarabi and M.A. Perkowski, "Fast exact and quasi-minimal minimization of highly testable fixed-polarity AND/XOR canonical networks," Proc. 29th ACM/IEEE DAC, pp.30–35, June 1992.
[9] T. Sasao, "Logic synthesis with EXOR gates," in Logic Synthesis and Optimization, ed. T. Sasao, pp.259–285, Kluwer Academic Publishers, 1993.
[10] T. Sasao, "Optimization of pseudo-Kronecker expressions using multiple-place decision diagrams," IEICE Trans. Inf. & Syst., vol.E76-A, no.3, pp.475–482, March 1994.
[11] T. Sasao, "Easily testable realizations for generalized Reed-Muller expressions," IEEE Trans. Comput., vol.46, no.6, pp.709–716, June 1997.
[12] T. Yamada, "Easily testable AND-EOR combinational logic circuits," IECE Trans., vol.J66-D, no.1, pp.105–110, Jan. 1983.

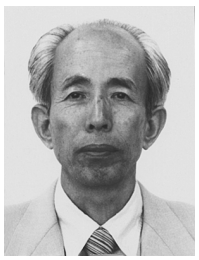
Takashi Hirayama received his BE, ME, and PhD degrees in computer science from Gunma University, Kiryu, Japan, in 1994, 1996, and 1999, respectively. He is currently a research assistant in the Department of Electrical and Electronics Engineering, Ashikaga Institute of Technology, Ashikaga, Japan. His research interests include high level and logic synthesis and design for testability.



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



Yasuaki Nishitani received his BE degree in electrical engineering, his ME and PhD degrees in computer science from Tohoku University, Sendai, Japan, in 1975, 1977, and 1984, respectively. In 1981 he joined the Software Product Engineering Laboratory at the NEC corporation. Since 1987 he has been an associate professor in the Department of Computer Science, Gunma University. His current research interests include the software engineering and distributed algorithms.



Kensuke Shimizu received his BE, ME, and PhD degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1962, 1964, and 1967, respectively. He was a lecturer from 1967 to 1968 and an associate professor from 1968 to 1976 in the Department of Electronic Engineering, Faculty of Engineering, Gunma University. Since 1976 he has been a professor in the Department of Computer Science, Faculty of Engineering, Gunma University, engaged in research and education in logic circuits, switching theory, and expert systems.