96

Detection of stuck-at and bridging faults in Reed–Muller canonical (RMC) networks

Prof. T. Damarla
Prof. M. Karpovsky

Indexing terms: Boolean algebra, Fault detection

Abstract: Boolean function realisations by Reed–Muller networks have many desirable properties in terms of testability [11]. In the paper it is shown that there exists a single set of test patterns which would detect all single stuck-at and all single bridging (short-circuit) faults in Reed–Muller networks, and the number of test patterns is shown to be at most $3n + 5$, where n is the number of input variables in the function. In the case of networks with k outputs, where $k \leqslant 2^n$, the number of test patterns required to detect all single stuck-at and all single detectable bridging faults (both *AND* and *OR*) is also shown to be $3n + 5$.

1 Introduction

The problem of testing combinational networks is shown to be NP-complete [1] even for the simple but most widely used fault model stuck-at-fault (SAF) model. There are some standard algorithms, such as d-algorithm [2], PODEM [4] and FAN [3] for the generation of test patterns for combinational networks. Though these algorithms guarantee generation for a given fault if one exists, the time T required to generate a test pattern and for fault simulation is shown [4] to be $T = KG^3$, where G is the number of gates in a circuit and K is some constant. Even for circuits with modest sizes the time complexity is prohibitively large. If a fault model includes bridging faults (BFs), then test generation is even more complex. However, for some classes of networks it is possible to find reasonably efficient test generation procedures for fault models with single SAFs and bridging faults. In this paper fault detection (both single stuck-at and bridging faults) in Reed–Muller canonical (RMC) networks will be investigated.

There has been renewed interest in Reed–Muller networks [5–9, 14, 15] in the last few years. Reddy [11] has shown that RM canonical networks have simple tests for detection of SAFs and an upper bound on a number of test patterns for detection of all single SAFs is $3n + 3$, where n is the number of input variables, if we distinguish between the primary inputs and the fan-out lines of primary input lines which are connected to *AND* gates in Reed–Muller networks (see Fig. 1). In this paper, primary inputs and fan-out lines are not distinguished for the sake of bridging faults. When the primary inputs and their fan-outs are not distinguished, an upper bound on the number of patterns for detection of all single SAFs is shown [11] to be $2n + 3$. Others [12, 13] have considered fault detection in Reed–Muller networks but their fault models did not include bridging faults (BF). In this paper a test set which detects all SAFs and all detectable BFs (both *AND* and *OR* type) will be constructed. The size of the test set is shown to be at most $3n + 5$. Results obtained for single-output RM networks are generalised for multiple output RM networks.

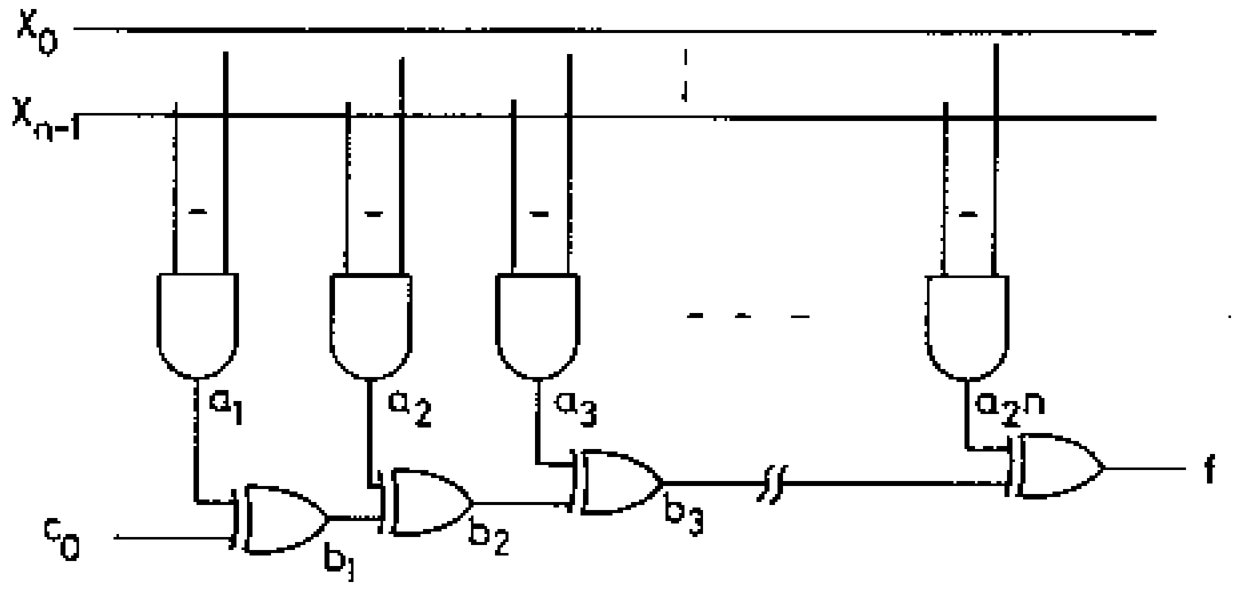

Fig. 1 *Reed–Muller canonical network*

Any Boolean function $f(x_0, x_1, \ldots, x_{n-1})$ has an unique Reed–Muller canonical representation given by

$$f(x_0, x_1, \ldots, x_{n-1}) = c_0 \oplus c_1 x_0 \oplus \cdots \oplus c_{2^n-1} x_0 x_1, \ldots, x_{n-1} \quad (1)$$

where $c_i \in \{0, 1\}$. The RM transform, mapping of $f = (f(0, \ldots, 0), f(0, \ldots, 1), \ldots, f(11, \ldots, 1))$ to $(c_0, c_1, \ldots, c_{2^n-1})$ and implementation of Boolean functions is shown in Reference 9. The general structure of the network based on the Reed–Muller canonical form is shown in Fig. 1.

Any line h_i in a network may be stuck-at-0 ($h_i/0$) or stuck-at-1 ($h_i/1$) or any two lines may be shorted resulting in either feedback [10] or a nonfeedback bridging fault (BF). The BF may be either *AND* ($(h_i h_j)_*$) or *OR* ($(h_i h_j)_+$) [10].

Let $N(x_i)(N(x_i x_j))$ denote the number of *AND* gates to which $x_i(x_i x_j)$ is an input in an RM canonical network, and $N(x_i \cup x_j)$ denote the number of *AND* gates to which x_i or x_j are inputs

$$N(x_i \cup x_j) = N(x_i) + N(x_j) - N(x_i x_j) \quad (2)$$

2 Test set generation

Consider some of the test patterns for detection of SAFs given in Reference 11, namely

Paper 6717E (C2, C3), first received 24th February 1988 and in revised form 23rd March 1989

Prof. Damarla is at the Department of Electrical Engineering, University of Kentucky, 453 Anderson Hall, Lexington, KY 40506, USA

Prof. Karpovsky is at the ECS Department, College of Engineering, Boston University, 110 Cummington St., Boston, MA 02215, USA

$$T_1 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccccc} c_0 & x_0 & x_1 & \cdots & x_{n-1} \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \end{array}\right] \end{array};$$

$$T_2 = \begin{array}{c} \begin{array}{ccccc} c_0 & x_0 & x_1 & \cdots & x_{n-1} \end{array} \\ \left[\begin{array}{ccccc} d & 0 & 1 & \cdots & 1 \\ d & 1 & 0 & \cdots & 1 \\ d & \cdot & \cdot & \cdots & \cdot \\ d & 1 & 1 & \cdots & 0 \end{array}\right] \end{array} \qquad (3)$$

where $d \in \{0, 1\}$. In addition to T_1 and T_2 test sets T_{AND} and T_{OR} which detect all $(x_i x_j)_*$ and $(x_i x_j)_+$ will be constructed. It will be shown that the test set $T = T_1 \cup T_2 \cup T_{AND} \cup T_{OR}$ is sufficient to detect all single SAFs and all BFs.

2.1 Generation of test set T_{AND} for detection of $(x_i x_j)_*$

Let A_i be the subset of input variables which are inputs to an *AND* gate whose output is a_i (see Fig. 1) and the cardinality of A_i (number of variables in A_i) is $|A_i|$. Consider an *AND* gate with minimal $|A_i|$ and its input variables $x_i \in A_i$. Generate a test pattern t_i as follows:

$$t_i = (x_0, x_1, \ldots, x_{n-1})$$

$$\text{such that } a_i(t_i) = 1 \qquad \text{and} \qquad x_j = 0 \quad \forall x_j \notin A_i$$

t_i detects any BF $(x_i x_j)_*$ where $x_i \in A_i$ and $x_j \notin A_i$, since the output changes from '1' to '0' when the fault occurs. Test pattern t_i divides the set of input variables $X = \{x_0, \ldots, x_{n-1}\}$ into two subsets A_i and $\bar{A}_i = X - A_i$ as shown in Fig. 2. Now only the BFs to be detected are $(x_r x_s)_*$ and $(x_u x_v)_*$, $x_r, x_s \in A_i$, for all $r \neq s$ and $x_u, x_v \in \bar{A}_i$, for all $u \neq v$. The process of test generation is carried out by considering minimal subset of variables $A_r \subset A_i$ such that $(x_r x_s)_*$ is detected for all $x_r \in A_r$ and for all $x_s \in \bar{A}_r = A_i - A_r$. This is accomplished by considering an *AND* gate with A_r as its subset of inputs (and if required some other variables from $\bar{A}_i$) such that the output of this *AND* gate $a_j(t_j) = 1$ for a test vector t_j and $a_j(t_j) = 0$ if there is a bridging fault $(x_r x_s)_*$ for all $x_r \in A_r$ and for all $x_s \in \bar{A}_r$. Every time a new test pattern is generated, the parent subset is further divided into two nonempty subsets of smaller size (one of the subsets possibly a single element subset), such that the bridgings between inputs from different subsets are detected. Every time a new test pattern is generated, the binary tree is expanded as in Fig. 2. The process of test generation is carried out until all the pendant vertices in Fig. 2 are input variables. Since every test pattern generates a new branch or branches, in the worst case every internal node of the tree corresponds to a test pattern. The number of internal nodes is $n - 1$ for a binary tree with n pendant vertices and we have for the number of test patterns required to detect all input BFs $(x_i x_j)_*$

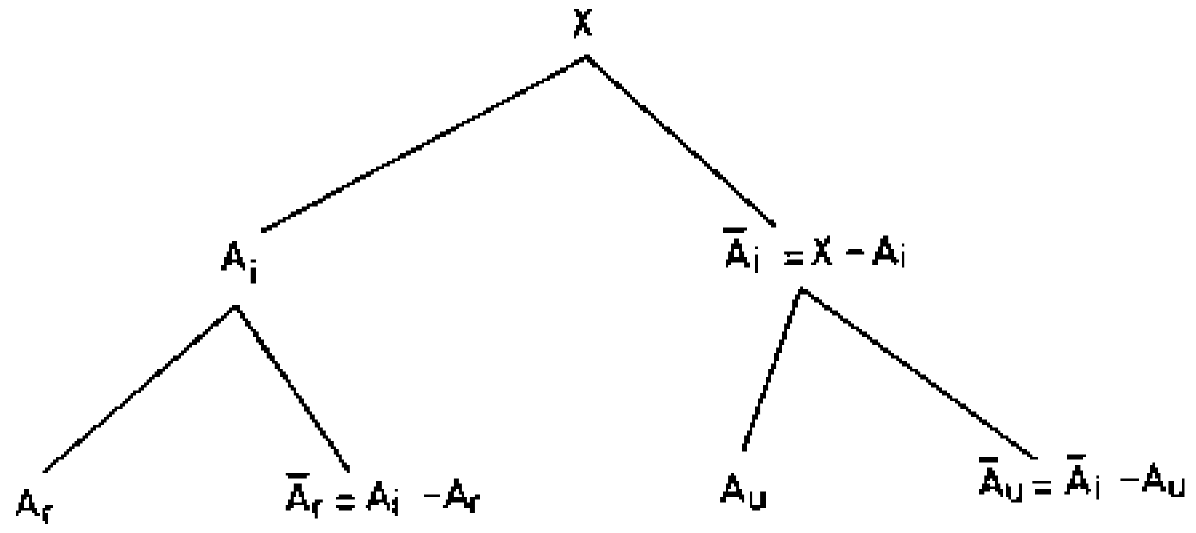

Fig. 2 *Binary tree*

$$|T_{AND}| \leqslant n - 1 \qquad (4)$$

The following example illustrates the construction of T_{AND}.

Example 1: Let

$$f = x_0 x_1 \oplus x_0 x_2 x_3 \oplus x_1 x_3 x_4 x_5 \oplus x_0 x_1 x_3 x_4 \oplus x_2 x_3 \oplus x_4 x_5 = a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6$$

Let us select A_1 as the minimal A_i among the possible minimal A_is, $i = (1, 5, 6)$. Then the test pattern $\tau_1 = 110000$ divides X into two subsets $\{x_0 x_1\}$ and $\{x_2 x_3 x_4 x_5\}$. To detect $(x_0 x_1)_*$ consider an *AND* gate a_2 with $x_0 \in A_2$ (note A_2 is not minimal for x_0) and the corresponding test pattern $\tau_2 = 101100$ detects $(x_0 x_1)_*$; similarly other test patterns are constructed. Test patterns $T_{AND} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ for detection of all input *AND* bridging faults are given below and the corresponding binary tree is given in Fig. 3. The test pattern which detects $(x_i x_j)_*$ is given as a matrix element m_{ij} in the matrix M.

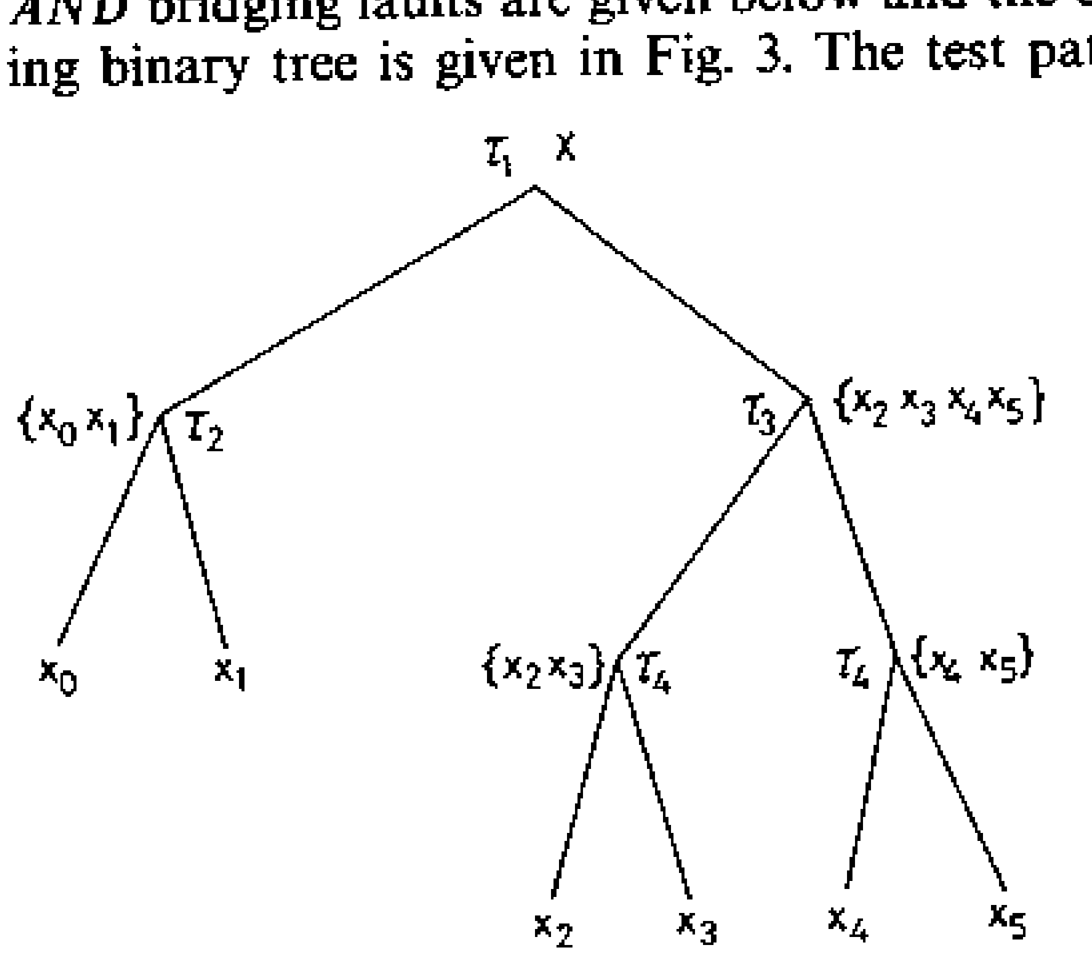

Fig. 3 *Binary tree generated by test patterns in example*

$$\begin{array}{ccccccc} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ \tau_1 = & 1 & 1 & 0 & 0 & 0 & 0 \\ \tau_2 = & 1 & 0 & 1 & 1 & 0 & 0 \\ \tau_3 = & 0 & 0 & 1 & 1 & 0 & 0 \\ \tau_4 = & 1 & 1 & 0 & 1 & 1 & 0 \end{array};$$

$$M = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \left[\begin{array}{ccccc} \tau_2 & \tau_1 & \tau_1 & \tau_1 & \tau_1 \\ 0 & \tau_1 & \tau_1 & \tau_1 & \tau_1 \\ \tau_1 & 0 & \tau_4 & \tau_3 & \tau_3 \\ \tau_1 & \tau_4 & 0 & \tau_3 & \tau_3 \\ \tau_1 & \tau_3 & \tau_3 & 0 & \tau_4 \end{array}\right] \end{array}$$

2.2 Generation of test set T_{OR} for detection of $(x_i x_j)_+$

To facilitate the generation of a test set for detection of *OR* input bridgings $(x_i x_j)_+$, an $[n \times n]$ parity matrix $P = (p_{ij})$ will be constructed where the ith row and the jth column correspond to the variables x_i and x_j, respectively, and p_{ij} is given by

$$p_{ij} = \begin{cases} 0 & \text{if } N(x_i x_j) = \text{even} \\ 1 & \text{if } N(x_i x_j) = \text{odd} \end{cases} \qquad (5)$$

where $N(x_i x_i) = N(x_i)$, (see eqn. 2). Notice that the P matrix gives the information on whether a variable x_i is going to an even or an odd number of *AND* gates depending on p_{ii} being '0' or '1'. Similarly, it also gives information on every pair of variables x_i and x_j, whether they jointly go to an even or an odd number of *AND* gates. To detect a bridging fault, the fault should affect an odd number of gates so that its effect will result in a change of an output. The generation of test patterns for $(x_i x_j)_+$ is described below depending on the various cases whether $p_{ii} = 0$ or 1 and $p_{ij} = 0$ or 1.

Case (a): $p_{ii} = 1$. Then, the test pattern

$$\begin{aligned} t &= (x_0, x_1, \ldots, x_{i-1}, \overset{i}{x_i}, x_{i+1}, \ldots, x_{n-1}) \\ &= (1\ 1\ \cdots\ 1\ \overset{i}{0}\ 1\ \cdots\ 1\ 1) \end{aligned} \tag{6}$$

detects $(x_i x_j)_+$, for all $j \neq i$. Input set X is partitioned into $\{x_i\}$ and $X - \{x_i\}$, and $t \in T_2$ where T_2 is defined by eqn. 3. Test patterns for all variables $x_i \in X$ for which $p_{ii} = 1$ should be generated first before going to *case (b)* or *case (c)*.

Case (b): $p_{ii} = 0$. If x_k is such that $p_{ik} = 1$, then test pattern

$$\begin{aligned} t &= (x_0, x_1, \ldots, x_i, \ldots, x_k, \ldots, x_{n-1}) \\ &= (1\ 1\ \cdots\ 1\ \overset{i}{0}\ 1\ \cdots\ 1\ \overset{k}{0}\ 1\ \cdots\ 1\ 1) \end{aligned} \tag{7}$$

detects $(x_i x_j)_+$ for all $j \neq i, k$ since the number of *AND* gates enabled after the bridging is equal to $N(x_i) - N(x_i x_k) =$ odd. If $p_{kk} = 0$, then the same test pattern (eqn. 7) detects $(x_k x_j)_+$ for all $j \neq i, k$. If $p_{kk} = 1$, then it comes under *case (a)*. Again, the test pattern (eqn. 7) generates a new branch of the binary tree in which x_i is an element. Test patterns for other variables are generated similarly (see *Example 2*).

If *case (a)* and *case (b)* are not satisfied for a single variable or a number of variables, the test patterns for them will be generated as described in *case (c)*.

Case (c): $P = [0]$. Find a subfunction $f_i = f(x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_{n-1})$ and the corresponding parity matrix P_1 for this function. If $P_1 = [0]$ for $f(X \mid x_i = 0)$ for all x_i then consider P_2 for $f(X \mid x_i = x_j = 0)$; if $P_2 \neq [0]$ for some x_i and x_j, then the test patterns are generated as in *case (a)* and *(b)*. If $P_2 = [0]$ for all $x_i = x_j = 0$, consider P_3, for the case when three variables set to 0, and so on (see *Example 2*).

As in the case of *AND* bridgings, test patterns generate a binary tree and the maximum number of test patterns required is $n - 1$. Hence

$$|T_{OR}| \leqslant n - 1 \tag{8}$$

Example 2: Let

$$\begin{aligned} f = {} & x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_0 x_1 \oplus x_0 x_1 x_2 \oplus x_0 x_4 \\ & \oplus x_1 x_5 \oplus x_2 x_3 \oplus x_2 x_4 \oplus x_0 x_1 x_3 \oplus x_0 x_1 x_2 x_3 x_4 \end{aligned}$$

Then

$$P = \begin{array}{c} \begin{array}{cccccc} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \end{array};$$

$$P_1 = \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

By eqn. 6 (*case (a)*), $p_{5,5} = 1$, $\tau_1 = (111110)$.

By eqn. 7 (*case (b)*), $p_{1,1} = 0$, $p_{1,4} = 1$, $\tau_2 = (101101)$; $p_{3,3} = 0$, $p_{3,4} = 1$, $\tau_3 = (111001)$.

Now (*case (c)*) set $x_0 = 0$ and obtain P_1. Then, from P_1 $p_{1,1} = 0$, $p_{1,5} = 1$, $\tau_4 = (001110)$. The binary tree generated by these test patterns is given by Fig. 4, and the BFs detected by these test patterns are represented by the following matrix:

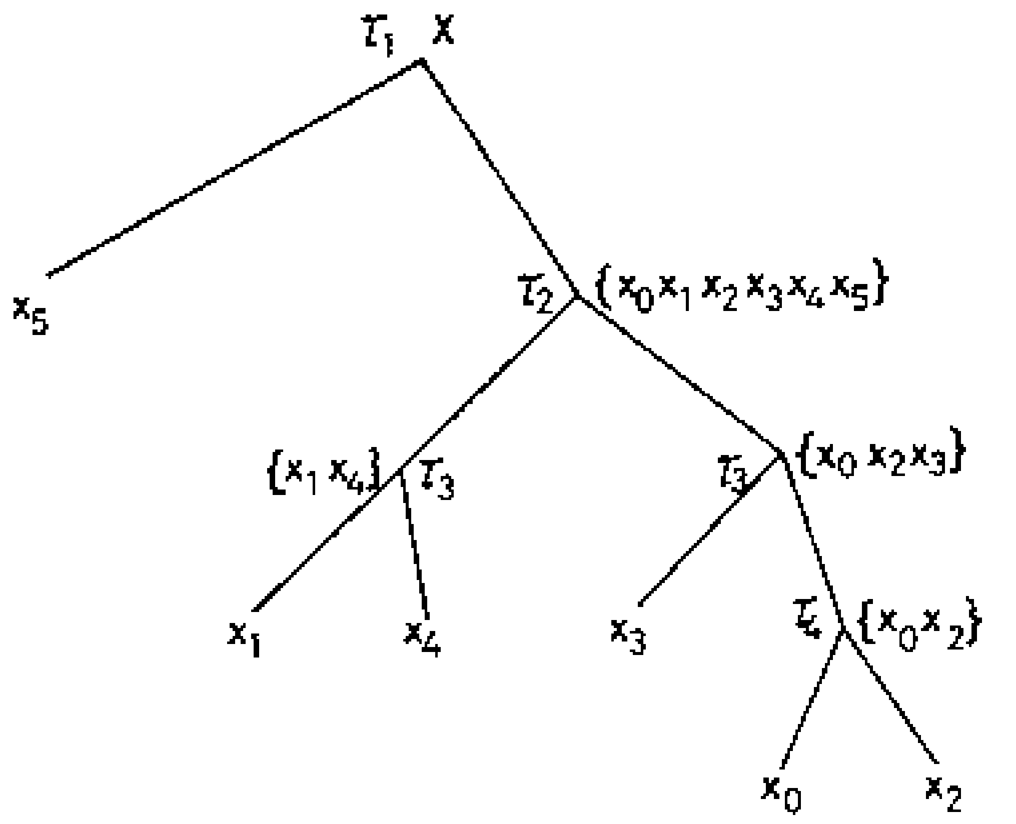

Fig. 4 *Binary tree generated by test patterns in example*

$$M = \begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline x_0 & \tau_2 & \tau_4 & \tau_3 & \tau_2 & \tau_1 \\ x_1 & 0 & \tau_2 & \tau_2 & \tau_3 & \tau_1 \\ x_2 & \tau_2 & 0 & \tau_3 & \tau_2 & \tau_1 \\ x_3 & \tau_2 & \tau_3 & 0 & \tau_2 & \tau_1 \\ x_4 & \tau_3 & \tau_2 & \tau_2 & 0 & \tau_1 \end{array}$$

Now we will show that test sets T_1, T_2, T_{AND} and T_{OR} are sufficient to detect all single SAFs and BFs in the network.

It is easy to see [11] that T_1 and T_2 detect all single SAFs except those at the primary inputs which are inputs to even number of *AND* gates; however there exists a test pattern with $x_i = 1$, $x_j = 0$ which detects $(x_i x_j)_*$ in T_{AND}. This test pattern detects $x_i/0$ faults. Similarly, there exists a test pattern in T_{OR} which assigns '0' to a subset of variables including x_i and '1' to the rest of the variables. This test pattern detects $x_i/1$ fault. Thus $T_{AND}(T_{OR})$ detects at least $n - 1$ faults $x_i/0(x_i/1)$, and two more test patterns are necessary and are constructed using the procedures described in T_{AND} and T_{OR}.

Now internal BFs will be considered. From Fig. 1 it is clear that only the following BFs can occur among the internal lines of an RMC network:

$(a_i b_i)$, $(x_i b_j)$, $(x_i a_j)$, $(b_i b_j)$ and $(a_i a_j)$

Now it will be shown that the test patterns already developed also detect the above-mentioned BFs. In the following paragraphs only test sets

$T_1 = \{t_1, t_2, t_3, t_4\}$, T_2, T_{AND} and T_{OR}

are used. Some of the BFs are detected using an asynchronous method [10], for example, to detect an *AND* bridging fault between an input x_i and output a_i of an *AND* gate, first a test pattern which results in output

$a_i = 0$ and then another test pattern which results in $a_i = 1$ should be applied. A faulty gate would give '0' as an output in both cases.

(i) $(a_i b_j)_*$, $(a_i b_j)_+$ and $(x_i b_j)_*$, $(x_i b_j)_+$: all these bridging faults are detected by t_2, t_3 and t_4.

(ii) $(x_i a_j)_*$ *AND* bridgings: if $x_i \notin A_j$, then $(x_i a_j)_*$ is detected by $t \in T_2$, where $x_i = 0$ and for all $j \neq i$, $x_j = 1$. If $x_i \in A_j$ and $N(x_i)$ is odd, then $(x_i a_j)_*$ is detected asynchronously [10] when (t_1, t_4) are applied in the given order. If $N(x_i)$ is even and $x_i \in A_j$, then test pattern $t_1 \in T_1$ which assigns $a_k = 0$ and one of the test patterns $t \in T_2$ which assigns $a_k = 1$ for some a_k, where x_i, $x_k \in A_k$, would detect $(x_i a_j)_*$ asynchronously, since the fault $(x_i a_j)_*$ causes $a_k = 0$ and changes the parity.

(iii) $(x_i a_j)_+$ *OR* bridgings: any test pattern $t \in T_2$, such that $a_j = 0$ and $x_i = 1$ detects $(x_i a_j)_+$.

(iv) $(b_i b_j)_*$, $(b_i b_j)_+$ and $(c_0 b_j)_*$: *AND* (*OR*) BFs are detected asynchronously by $t_1, t_3(t_3, t_1)$.

(v) $(a_i a_j)_*$, $(a_i a_j)_+$: test set T_2 is sufficient to detect both *AND* and *OR* BFs of this type.

For detection of some of the BFs (for example, $(x_i a_j)_*$ and $(b_i b_j)_{*,+}$) the order in which the test patterns applied is important. To detect these BFs the following sequence can be employed $(t_1, t_3, t_1, t_4, t_2)$ and hence t_1 is applied twice.

From the results presented in this Section one can see that any Reed–Muller canonical network requires at most $3n + 5$ test patterns to detect all single SAFs and all single detectable bridging faults (both *AND* and *OR* bridging faults).

3 Reed–Muller networks with multiple outputs

The general structure of a Reed–Muller network for a system of functions with k output functions and n input variables is shown in Fig. 5. We assume that $k \leqslant 2^n$. First,

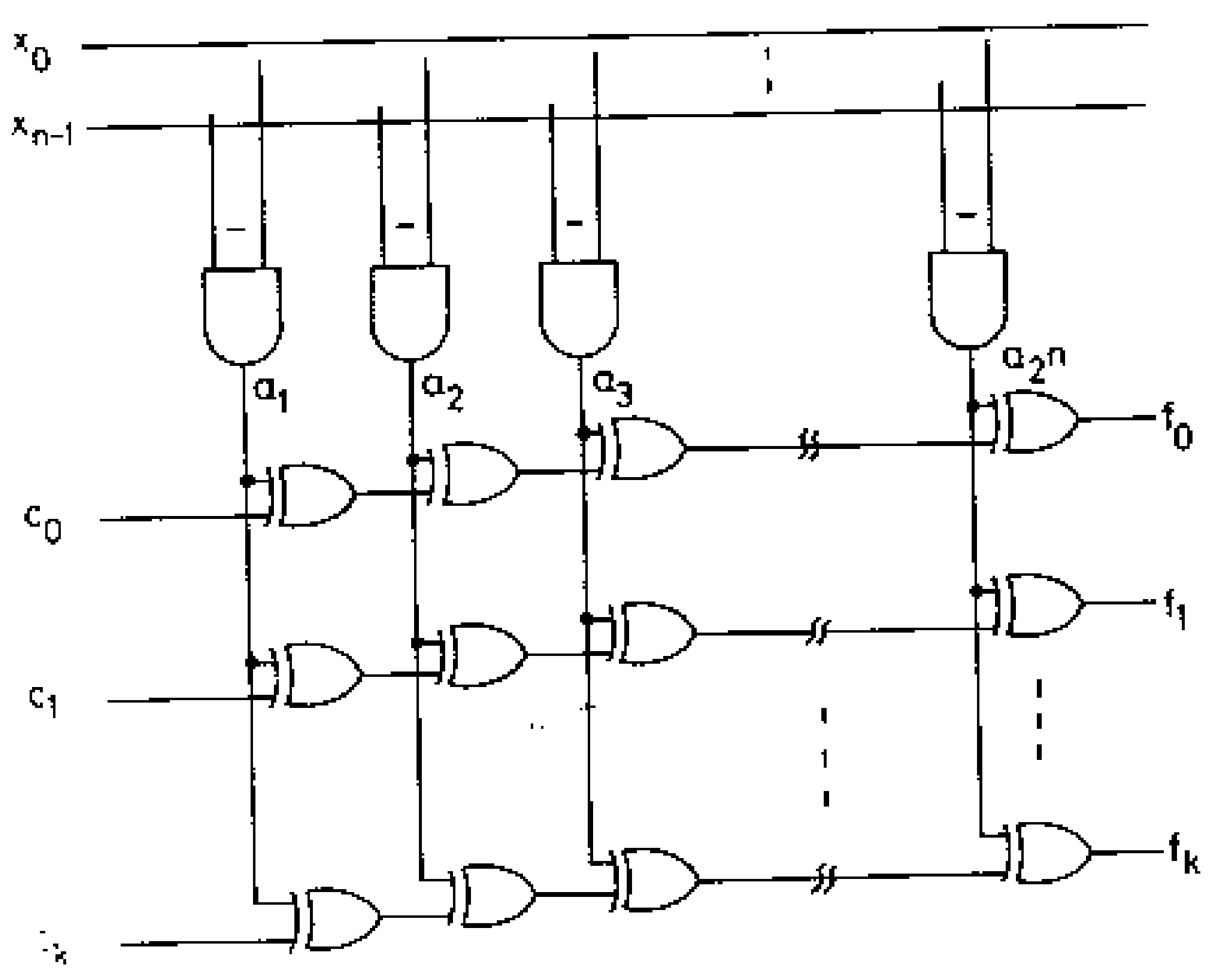

Fig. 5 *General structure for multiple output RMC networks*

using the technique described in Section 2 we construct a test for a subnetwork implementing f_0. (See Fig. 5. We assume that f_0 depends on all the input variables $x_0, \ldots, x_{n-1}$). For this test we set $c_0 = c_1 = \cdots = c_{k-1}$. All the SAFs and bridging faults are detected as described in Section 2. The only bridging faults that are to be detected are the ones between path c_i to f_i and another path c_j to f_j, for all $i \neq j \in \{0, 1, \ldots, k-1\}$. To detect these bridging faults it is sufficient to replace T_2 (see eqn. 3) by

$$T_2 = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{ccccccccccc} c_0 & c_1 & c_2 & c_3 & \cdots & c_{k-1} & x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ \end{array}$$

$$T_2 = \left[\begin{array}{ccccccccccc} 0 & 1 & 0 & 1 & \cdots & 1 & 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & 1 & \cdots & 1 & 1 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 0 & \cdots & 1 \\ & & & & \cdots & & & & & \cdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & & & & \cdots & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 0 \end{array}\right]$$

where the binary column under c_i is the binary representation of i.

It is evident that for any two columns c_i and c_j there exists a test vector which assigns a '0' to c_i and a '1' to c_j or vice versa, and hence detects both *AND* and *OR* bridging faults in paths c_i to f_i and c_j to f_j.

The above results are summarised in the following theorem:

Theorem: Any Reed–Muller network with k outputs and n inputs ($k \leqslant 2^n$) requires at most $3n + 5$ test patterns to detect all single SAFs and both *AND* and *OR* bridging faults which are detectable.

References

1 IBARRA, O.H., and SAHNI, S.K.: 'Polynomially complete fault detection problems', *IEEE Trans. Comput.*, 1975, **C-24**, pp. 242–249

2 ROTH, J.P.: 'Diagnosis of automata failures: a calculus and a method', *IBM J. Res. Dev.*, 1966, **10**, pp. 278–291

3 FUJIWARA, H., and SHIMONO, T.: 'On the acceleration of test generation algorithms', *IEEE Trans. Comput.*, 1983, **C-32**, pp. 1137–1144

4 GOEL, P.: 'An implicit enumeration algorithm to generate tests for combinational logic circuits', *IEEE Trans. Comput.*, 1981, **C-30**, pp. 215–222

5 BESSLICH, PH.W.: 'Spectral processing of switching functions using signal-flow transformations', *in* KARPOVSKY, M.G. (Ed.): 'Spectral techniques and fault detection' (Academic Press Inc., 1985)

6 BESSLICH, PH.W.: 'Efficient computer method for Ex-OR logic design., *IEE Proc. E., Comput. & Digital Tech.*, 1983, **130**, pp. 203–206

7 CHEN, X.: 'The mapping of spectral coefficients and its application', *Comput. & Elect. Eng.*, 1982, **9**, pp. 167–180

8 DAMARLA, T., and KARPOVSKY, M.: 'Reed–Muller spectral techniques for fault detection'. Second Int. Workshop on Spectral Techniques, Ecloe Polytechnique de Montreal, Montreal, Canada, October 1986

9 DAMARLA, T., and KARPOVSKY, M.: 'Fault detection in combinational networks by Reed–Muller transforms'. Accepted for publication in *IEEE Trans. Comput.*

10 KARPOVSKY, M., and SU, S.Y.H.: 'Detection and location of input and feedback bridging faults among input and output lines', *IEEE Trans. Comput.*, 1980, **C-29**, pp. 523–527

11 REDDY, S.M.: 'Easily testable realizations for logic functions', *IEEE Trans. Comput.*, 1972, **C-21**, pp. 1183–1188

12 SALUJA, K.K., and REDDY, S.M.: 'Fault detecting test sets for Reed–Muller canonic networks', *IEEE Trans. Comput.*, 1975, **C-24**, pp. 995–998

13 PRADHAN, D.K.: 'Universal test sets for multiple fault detection in AND-EXOR arrays', *IEEE Trans. Comput.*, 1978, **C-27**, (2), pp. 181–187

14 WU, X., CHEN, X., and HURST, S.L.: 'Mapping of Reed–Muller coefficients and the minimization of exclusive-OR switching functions', *IEE Proc. E, Comput. & Digital Tech.*, 1982, **129**, (1), pp. 15–20

15 ZHANG, Y.Z., and RAYNER, P.J.W.: 'Minimization of Reed–Muller polynomials with fixed polarity', *IEE Proc. E, Comput. & Digital Tech.*, 1984, **131**, (5), pp. 177–186