1977

Diagnosis of Faults in Linear Tree Networks

Sharad C. Seth
University of Nebraska - Lincoln, seth@cse.unl.edu

K. L. Kodandapani
University of Regina

Follow this and additional works at: http://digitalcommons.unl.edu/csearticles
Part of the Computer Sciences Commons

http://digitalcommons.unl.edu/csearticles/25

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Journal Articles by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
Abstract—The problem of fault detection and location in tree networks of two input EXCLUSIVE-OR (EOR) gates is considered. The fault model assumes that an EOR gate can change to any other function of its two inputs except the equivalence function. An efficient procedure for single fault location is presented. In the worst case the number of tests necessary to locate single faults is bounded by a linear function of the number of input variables. Constructive upper bounds are obtained for the number of tests to detect multiple faults. Optimality of these bounds is argued and extension of results to other types of networks is considered.

Index Terms—Fault detection and location, linear tree networks, multiple faults, optimum test sets.

I. INTRODUCTION

LINEAR logic networks have been found useful in the encoding and decoding of digital information. As subnetworks they also appear in functional implementations based on Reed–Muller canonical (RMC) forms [1]. From a fault detection viewpoint two-level realizations of linear logic functions are the hardest to test since they require exhaustive testing. In this paper we show that both the problems of fault detection and fault location are greatly simplified for multilevel realizations of linear functions.

It is assumed that a linear network is built in the form of a binary tree using the two input EXCLUSIVE-OR (EOR) gate as the basic node element. A single fault in the tree corresponds to a failure of any EOR gate to any other function of its two inputs except the Equivalence function. A multiple fault corresponds to the simultaneous presence of one or more single faults. We will be concerned with the detection and location of single faults (Section II) and the detection of multiple faults (Section III). Extension of results to other types of networks is considered in Section IV.

Problems similar to the ones considered here have been discussed by Breuer [2] but his fault model permits only stuck type faults on the input and output lines of EOR gates.

II. DIAGNOSIS OF SINGLE FAULTS

A. Fault Detection

It is well known that to detect single faults in linear tree networks four tests are both necessary and sufficient [3]. A procedure for deriving these tests for an arbitrary linear tree network will be presented here. The basic idea is to label each line in the network uniquely with one of three binary vectors: 0011, 0110, or 0101. The labels on the input lines correspond to the required test inputs, while the other labels indicate the fault free response of the network to these test inputs.

Algorithm D: (To label a linear tree for single fault detection.)

Step D1: Label the output line arbitrarily with one of the three binary vectors: 0011, 0110, or 0101.

Step D2: For each EOR gate whose output line is labeled but whose input lines are not, repeat Step D3 until all lines in the network are labeled.

Step D3: Choosing from the same three vectors label the two input lines of the EOR gate differently from the output line and from each other.

The correctness of the algorithm follows from the following two observations.

1) Any one of the three binary vectors can be expressed as the component-wise EOR sum of the other two vectors;

2) Any pair of the three vectors applied as inputs to an EOR gate tests it exhaustively.

B. Fault Location

We would like to pinpoint the site of a single fault in a linear tree as accurately as possible; but in as much as a fault on a line connecting two EOR gates in a tree may be considered indistinguishable from certain faults in either of the two gates, it follows that the resolution of faults by external testing can at best be up to two adjacent gates.

Lemma 1: Let \( C \) be the correct response of an EOR tree network to a single fault detection test set obtained from Algorithm D. Then no single fault can change the tree response to \( C \).

Proof: If a single fault results in the response \( C \), it is possible only if the output of an EOR gate is itself complemented for all the test inputs. But, since the four tests
from Algorithm D test each EOR gate in the tree exhaustively, this would imply the gate function changing to the Equivalence function in contradiction of the fault model.

Lemma 1 can be strengthened to apply to any single fault detection test set because under the assumed fault model it can be easily shown that any test set to detect single faults must apply all four input combinations to each gate.

The algorithm to be developed for fault location makes use of the principle of binary search. Assume that the network $N$, viewed as a binary tree, has three components: a left subtree $N_L$, a right subtree $N_R$, and the root node $R$ which is the output gate of $N$. The algorithm will generally try to isolate a fault to one of three components. If $R$ was found to be faulty, the procedure could stop as the fault would be located to a single gate. Otherwise the procedure would be iterated on the faulty subtree.

We will use further notation for brevity. The input pattern of four tests obtained by applying Algorithm D to an EOR tree will be denoted by TIP; if an input pattern of all zeros is being applied it will be denoted by ZIP.

Let us consider the effect of applying TIP to $N_R$ and ZIP to $N_L$. Assume $B$ is the fault free response of $N$ to this pattern of four inputs. If no faults are present, $B$ would also be the output of $N_R$. We will now analyze the possible network responses in the presence of single faults.

If the fault lies in $N_L$, either it does not affect the output of $N_L$ or it affects it in the same way for all the four inputs since they remain unchanged. Thus, a fault in $N_L$ implies a response of $B$ or $\bar{B}$ for $N$.

If the output gate $R$ is faulty it is obvious that the response of $N$ can only be one of the four vectors; $0$ (all zeros), $1$ (all $1$'s), $B$, or $\bar{B}$.

A response other than $0$, $1$, $B$, or $\bar{B}$ must imply a fault in $N_R$. In addition a fault in $N_R$ could result in a constant output, but the response $B$ is ruled out because TIP is a single fault detection test set for $N_R$ and the response $\bar{B}$ is ruled out by Lemma 1.

The above analysis applies, mutatis mutandis, to the symmetric case in which TIP is applied to $N_L$ and ZIP is applied to $N_R$ and the fault free response of the network is assumed to be $C$. A summary of the fault isolation possible from the eight tests thus applied to $N$ is given in Table I; the “$x$” entries correspond to logically impossible situations. As indicated in the table, fault isolation is complete except in two cases. Since the two cases of ambiguity are symmetric it is enough to consider only one of the two, say, the one in which the response to the first four tests is $0$ or $1$ and the response to the second four tests is $C$ or $\bar{C}$. The conclusion is that either $R$ or $N_R$ is faulty. The only possible failure modes for $R$ consistent with these responses can be seen to be the functions: zero, one, $x$, and $\bar{x}$, where $x$ is the left input of $R$. Thus, if $R$ were faulty and we proceeded iterating the procedure on $N_R$, while holding inputs to $N_L$ at a constant value, the output of $R$ will be held at a constant value independent of the output of $N_R$.

In other words, recurrence of an ambiguous case rules out the possibility of a fault in $R$. In summary, any ambiguity in fault isolation at the current step is bound to be resolved in the next step.

The preceding discussion leads to the following algorithm for fault location. It is assumed that fault detection has preceded this algorithm.

Algorithm L: (To locate a single fault in a linear tree.)

Step L1: While $N$ is a nontrivial tree repeat Steps L2 and L3, otherwise indicate that the fault lies in $N$ and stop.

Step L2: Apply TIP to $N_R$ and ZIP to other inputs and record the output. Assume $B$ is the fault free output. If the recorded output is other than $0$, $1$, $B$, or $\bar{B}$ replace $N_R$ by $N_R$ and go back to Step L1. Otherwise continue on to Step L3.

Step L3: Apply TIP to $N_L$ and ZIP to other inputs and record the output. Assume $C$ is the fault free output. If the recorded output is other than $0$, $1$, $C$, or $\bar{C}$ replace $N_L$ by $N_L$ and go back to Step L1. Otherwise distinguish the following cases on the basis of the responses observed in Steps L2 and L3.

Case 1: (The responses are $[0,1]$ and $[0,1]$ or $[B,\bar{B}]$ and $[C,\bar{C}]$.) Indicate gate $R$ to be faulty and stop.

Case 2: (The response is $[0,1]$ and $[C,\bar{C}]$.) Replace $N$ by $N_R$ and go back to Step L1.

Case 3: (The response is $[B,\bar{B}]$ and $[0,1]$.) Replace $N$ by $N_L$ and go back to Step L1.

Analysis of Algorithm L: The worst case bound on the number of tests is easily obtained. At each iteration at most eight tests are applied and the number of iterations is bounded by $(d - 1)$ where $d$ is the depth of the tree. Thus, in the worst case $8(d - 1)$ tests will be necessary. For a complete binary tree $[4]$ with $m$ inputs the depth $d = \lceil \log_2 m \rceil$; therefore, in the worst case $8 \log_2 m$ tests will be required.

A bound on the average number of tests can be obtained by assuming that all responses to the four tests applied in Steps L2 or L3 are equally probable. For any iteration the average number of tests is given by

$$4p + 8(1 - p)$$

where $p$ is the probability that the recorded response in Step L3 is not $0$, $1$, $B$, or $\bar{B}$. Under the assumption of equal likelihood of all responses the above can be rewritten as

$$4 \cdot \frac{12}{16} + 8 \cdot \frac{4}{16} = 3 + 2 = 5.$$

Thus, the average number of tests will not exceed $5(d - 1)$. Again this bound has the smallest value $5 \lceil \log_2 m \rceil$ for an $m$ input complete binary tree.
III. DETECTION OF MULTIPLE FAULTS

The ease of detecting single faults in EOR tree networks extends also to multiple faults. In this section we will develop a procedure which allows the derivation of multiple fault detection test sets in an iterative fashion.

Lemma 2: Every multiple fault in a linear tree network is detectable.

Proof: We use strong induction on the number of levels in the network. The basic step for one level networks is trivial. Assume the result is true for all networks of levels up to \( n \). Consider a \((n + 1)\) level network \( N \) with components \( N_L, N_R, \) and \( R \) as defined in Section II. Let \( X_L \) and \( X_R \) be the (disjoint) sets of inputs applied, respectively, to \( N_L \) and \( N_R \). Two cases are possible: either a multiple fault includes a fault in \( R \) or it does not. In the first case \( R \) changes to some nonlinear function of its two inputs and hence a nonlinear function of \( X_L \) and \( X_R \). Such a fault is clearly detectable. In the second case a fault in \( N_L(N_R) \) is detectable by the inductive assumption at the output of \( N_L(N_R) \) and since every input is a sensitizing input for an EOR gate it would also be detectable at the output of \( N \).

Lemma 3: Assume the correct response of a linear tree network to a multiple fault detection test set is a binary vector \( C \). Then no multiple fault can change the response to \( \bar{C} \).

Proof: By contradiction assume the response is \( \bar{C} \) for some multiple fault. If an inverter is introduced at the output of the faulty network the response would be \( C \) thus indicating the absence of a fault. Now, because of the relation \((\bar{x} \oplus \bar{y}) = \bar{x} \oplus y\), the inverter can be “pushed back” through fault free EOR gates until it occurs at the output of a faulty gate. Here it can be subsumed to define a new component of the multiple fault (other fault components remain unchanged). It follows that the multiple fault, so defined, is not detectable by the test set in contradiction of Lemma 2.

Corollary 1: No multiple fault can complement the output function of a linear tree network.

Theorem 1: Assume the subnetworks \( N_L \) and \( N_R \) of a linear tree network \( N \) can be tested for multiple faults by \( n_l \) and \( n_r \) tests, respectively. Then \( N \) can be tested for multiple faults by \( n_l + n_r \) tests.

Proof: Let \( T(N_L) \) and \( T(N_R) \) be multiple fault detection test sets for \( N_L \) and \( N_R \) of sizes \( n_l \) and \( n_r \), respectively. Partition each test set into a set of false tests (those normally producing a 0) and a set of true tests (those normally producing a 1): 

\[
T(N_L) = \{a_1^L, a_2^L, \ldots, a_{n_l}^L \} \cup \{a_1^L, a_2^L, \ldots, a_{n_l}^L \}
\]

\[
T(N_R) = \{b_1^R, b_2^R, \ldots, b_{n_r}^R \} \cup \{b_1^R, b_2^R, \ldots, b_{n_r}^R \}
\]

where, without loss of generality, it can be assumed that 

\( 0 < j < n_l \) and \( 0 < k < n_r \).

Define a new set:

\[
T(N) = \{(a_j^L, y) : y \in T(N_L) \} \cup \{(a_k^R, x) : x \in T(N_R) \} \cup \{(a_j^L, b_k^R) : x \in T(N_L) \} \cup \{(a_k^R, b_j^L) : x \in T(N_R) \}
\]

Clearly, the size of \( T(N) \) is \( n_l + n_r \). We will now prove that \( T(N) \) detects all multiple faults in \( N \). The proof can be divided into two cases.

Case 1: The multiple fault includes a fault in the output EOR gate.

We can show by a method similar to that used in [3] that the four tests \((a_j^L, b_k^R), (a_j^L, b_k^R), (a_k^R, b_j^L), \) and \((a_k^R, b_j^L)\) would detect such a fault.

Case 2: The multiple fault does not include a fault in the output EOR gate.

Because of symmetry it is enough to consider that \( N_L \) is faulty (\( N_R \) may or may not be faulty). The proof goes by contradiction. Assume the fault is not detected by \( T(N) \), that is, the response of the faulty network is indistinguishable from that of the fault free network. In particular, this is true for the subset \( \{(x, b_j^L) : x \in T(N_L)\} \cup \{(x, b_j^L) : x \in T(N_R)\} \) which leads to only two possibilities for the response of \( N_L \) to \( T(N_L) \):

1) it is the same as the fault free response, or
2) it is the complement of the fault free response.

The first possibility contradicts Lemma 2 and the second contradicts Lemma 3.

Corollary 2: Any \( n \) input EOR tree can be tested by \( 2n \) tests.

Proof: Let \( F(n) \) be the number of tests required to detect multiple faults in an EOR tree with \( n \) inputs. Also let \( n_l \) and \( (n - n_l) \) be the number of inputs to test the two subtrees. Then from Theorem 1 we can write the following recursive relation:

\[
F(n) = F(n_l) + (n - n_l), \quad 1 < n_l < n;
\]

also

\[
F(1) = 2,
\]

since a line can be tested exhaustively by two tests.

It can be easily checked that \( F(n) = 2n \) is the unique solution to (1) and (2).

The result in Theorem 1 can be improved further if it is assumed that at least one of the subtrees is nondegenerate (i.e., does not consist of a direct connection from the input to the output).

Theorem 2: Assume that at least one of the subtrees \( N_L \) and \( N_R \) of \( N \) is nondegenerate and that \( N_L \) and \( N_R \) can be tested, respectively, by \( n_l \) and \( n_r \) tests. Then \( N \) can be tested by \( n_l + n_r - 1 \) tests.

Proof: Let \( T(N_L) \) and \( T(N_R) \) be as defined in the proof of Theorem 1. Assume, without loss of generality, that \( N_L \) is nondegenerate. Then \( T(N_L) \) must contain at least two false tests and two true tests (that is, \( j \geq 2 \) and \( n_l - j \geq 2 \)). Define a new set

\[
T(N) = \{(a_j^L, y) : y \in T(N_R) \} \cup \{(x, b_j^L) ; x \in T(N_L) \} \cup \{a_j^L \} \cup \{a_j^L \}
\]

Clearly, the size of \( T(N) \) is \( n_l + n_r - 1 \). It remains to be proved that \( T(N) \) detects all multiple faults. The proof can be divided into three mutually disjoint cases.
Case 1: The multiple fault includes a fault in the output gate.

The four tests \((a_1^0,b_1^0), (a_2^0,b_2^0), (a_3^0,b_3^0),\) and \((a_4^0,b_4^0)\) detect this fault. The argument is same as in Case 1 of Theorem 1.

Case 2: The multiple fault lies entirely within \(N_L\).

The subset \(\{x,b_1^0\}: x \in T(N_L)\) but \(x \neq a_1^0\) or \(\{x,b_1^0\}\) of \(T(N)\) applies \(T(N_L)\) to \(N_L\). Since \(N_R\) and the output gate are assumed to be fault free, and since any input to an EOR gate is a sensitizing input; the fault will be detected by these tests.

Case 3: The output gate is fault free but \(N_R\) is faulty (\(N_L\) may or may not be faulty).

The case is similar to Case 2 of Theorem 1, therefore, it follows that the subset \(\{a_1^0,y\}: y \in T(N_R)\) of \(T_N\) will detect such a fault.

Corollary 3: Any \(n\) input EOR tree can be tested for multiple faults be at most \(\lceil \frac{3n}{2} \rceil + 1\) tests.

Proof: Let the function \(F\) have the same meaning as in the proof of Corollary 1. Then the following equations for \(F\) can be written

\[
F(n) = F(n_1) + F(n - n_1) - 1, \quad 1 < n_1 < n \tag{4}
\]

\[
F(2) = 4 \tag{5}
\]

\[
F(1) = 2 \tag{6}
\]

It can be easily verified that the solution for \(F(n)\) satisfying (4)–(6) has a maximum value when the network represents a complete binary tree. In such a case if \(n\) is even, only (4) and (5) are applicable and it can be seen that \((3/2)n + 1\) is the unique solution. On the other hand, if \(n\) is odd, equation (6) is used for only one component in the decomposition of \(N\); for all other components, either (4) or (5) are applicable. In this case it can be seen that the solution for \(F(n)\) is \((3(n - 1)/2 + 1) + 1\). This is equal to \((3n - 1)/2 + 1\). In general, the solution can be expressed by the following equation:

\[
F(n) = \left\lceil \frac{3n}{2} \right\rceil + 1 \tag{7}
\]

which encompasses both cases.

The minimum value of \(F(n)\) satisfying (4)–(6) occurs when the tree is in the form of a cascade. In such a case (4) and (6) are repeatedly used, and (5) is used only once at the end of the decomposition process and when the node generates subtree reduces to a single EOR gate (see Fig. 1). Thus, each input adds an extra test except the last two inputs add four tests. The solution is clearly \(F(n) = n + 2\) which coincides with the value obtained by Hayes [3].

Example: Fig. 2 shows a linear tree network of seven variables. The dotted boxes show successive stages in the decomposition of the tree into left and right subtrees. For example, \(N_{L1}\) and \(N_{R1}\) are left and right subtrees of the network; \(N_{L2}\) and \(N_{R2}\) are the components of \(N_{L1}\) while \(N_{L3}\) and \(N_{R3}\) are components of \(N_{R1}\). Equation (3) is used to obtain a test set starting from the smallest components and iteratively moving up to bigger components. Thus, the first four tests \((t_1, \ldots, t_4)\) test \(N_{L2}\) and the next four, namely \(t_5, t_6, t_7\), and \(t_8\) test \(N_{R2}\), therefore, \(t_1\) through \(t_7\) constitute a test set for \(N_{L1}\). Similarly, \(t_1\) and \(t_8\) through \(t_{11}\) constitute a test set for \(N_{R1}\) as shown in the figure. Thus, 11 tests are sufficient to test the tree network.

### IV. Conclusion

#### A. Optimality of Multiple Fault Detection Test Sets

The following argument shows that the size of the test set obtained by repeated applications of (4)–(6) to a network is minimum.

Assume that the output gate of a network \(N\) is fault free; the multiple faults are thus restricted to subnets \(N_{L}\) and \(N_{R}\). Certainly a network with this restriction would require no more tests than the same network without the restriction.

Consider the detection of a multiple fault \(F\) whose two components \(F_L\) and \(F_R\) lie, respectively, in \(N_L\) and \(N_R\). Let \(t = (X_L,X_R)\) be a test for \(F\). If \(X_L\) detected \(F_L\) at the output of \(N_{L}\) and \(X_R\) detected \(F_R\) at the output of \(N_{R}\) then \(t\) cannot detect \(F\) at the output of \(N\) because both the inputs of the output gate will be complemented under the fault. Thus, if a fault in \(N_{L}\) is being sensitized by a test another fault in \(N_{R}\) cannot be sensitized by the same test. In other words, faults in \(N_{L}\) and \(N_{R}\) must be detected without an overlap. The minimum number of tests to achieve this is clearly \(n_{L} + n_{R} - 1\).

#### B. Extension of Results to Other Types of Networks

We will consider three types of networks: Linear trees with arbitrary fan-in: The basic modules in such networks realize parity function of two or more inputs. It is assumed that each module requires exhaustive testing for detection of faults within the module.

Algorithm D can be easily generalized to apply to these
networks; the number of tests required is equal to that required to test the gate with maximum fan-in.

Extension of Algorithm L to locate single faults is somewhat less obvious. Let the output gate have subnetworks $N_1, N_2, \ldots, N_m$ connected to its inputs. A single iterative step of the algorithm would consist of applying TIP (suitably redefined), in turn, to each $N_i$ while holding the remaining network inputs at the zero value. Arguments similar to those preceding Algorithm L can be given to show that complete fault resolution is possible in this way.

It is also possible to generalize Theorems 1 and 2 to the networks considered here. Take, for example, the extension of Theorem 2. If the output gate is assumed fault free the remaining network can be shown to be testable by $n_1 + n_2 + \cdots + n_m - 1$ tests, where $n_i$ is the size of the test set for the subnetwork $N_i$. On the other hand, the output gate with $m$ inputs is assumed to require $2^m$ tests which can be overlapped with the tests for $N_i$ by a technique similar to that used in the proof of Theorem 2 (as long as $(m - 1)$ components are nontrivial). Thus, the number of tests required will be max $[2^m, n_1 + n_2 + \cdots + n_m - 1]$.

Fanout-free NAND networks: For single fault location in such networks by an algorithm analogous to L, TIP becomes a single fault detection test set for the subnetwork under test and ZIP becomes a vector of constant input which sensitizes the path from the subnetwork under test to the output. However, the simplicity of Algorithm L is compromised in two ways by such an extension: 1) TIP is now dependent on the subnetwork under test and 2) a suitable alternative to ZIP must be recomputed for each iteration.

Methods of multiple fault detection in networks considered here, appear in [3] and [5].

---

Reed–Muller canonical networks: Linear cascade networks with a control input form subnetworks of RMC networks proposed by Reddy [1] to simplify the detection of single faults which do not occur on the primary input bus. RMC networks of $n$ inputs can be tested by $(n + 4)$ tests which are derivable independent of the network function. It is shown in [6] that RMC networks can be speeded up by replacing the cascade by a tree without affecting the size of the test set. However, this is accomplished at the cost of three control inputs instead of one.

ACKNOWLEDGMENT

The authors wish to thank Dr. S. M. Reddy for bringing to their attention his internal report [7] which also contains the extension discussed in [6].

REFERENCES


Sharad C. Seth (S’66–M’70) was born in Sagar, India on November 1, 1942. He received the B.E. degree in electronics and telecommunications from Jabalpur, India, the M.S. degree from the Indian Institute of Technology, Kanpur, India, in 1966, and the Ph.D. degree from University of Illinois, Urbana, in 1970.

Since then he has been on the faculty of the Department of Computer Science, University of Nebraska at Lincoln, currently holding an associate professor's position. His research interests are in the areas of fault-tolerant computing, switching theory, and software verification.

Dr. Seth is a member of the Association for Computing Machinery.

K. L. Kodandapani received the B.E. degree in electrical engineering from Mysore University, Mysore, India, the M.E. degree in applied electronics and servomechanisms, and the Ph.D. degree in electrical engineering from the Indian Institute of Science, Bangalore, India in 1971 and 1974, respectively.

From September 1974 to November 1975, he was a lecturer in the Computer Science Program, Indian Institute of Technology, Kanpur, India. Since December 1975, he has been a Post-Doctoral Fellow at the Department of Computer Science, University of Regina, Regina, Canada. His current research interests include fault-tolerant computing and parallel processing.