How to Test a Tree

Andrew B. Kahng,[1] Gabriel Robins,[2] Elizabeth A. Walkup[3]

[1] Department of Computer Science, UCLA, Los Angeles, California 90095-1596

[2] Department of Computer Science, University of Virginia, Charlottesville, Virginia 22903-2442

[3] Duet Technologies, Bellvue, WA 98006

Received 26 October 1992; revised 24 March, 1995; accepted 2 August 1997

Abstract: We address the problem of verifying that a tree is connected using *probe* operations which check mutual connectivity between two (or more) leaves of the tree. We present optimal algorithms for determining minimal probe sets that detect all possible edge and vertex faults in arbitrary trees. Our results are of particular interest for the testing of interconnection substrates in VLSI multichip module packaging technologies. © 1998 John Wiley & Sons, Inc. Networks 32: 189–197, 1998

Keywords: graph algorithms; connectivity verification; tree testing; VLSI testing; reliability

1. INTRODUCTION

This paper addresses the problem of verifying that all the nodes of an arbitrary given tree are mutually connected. We assume that the testing operations consist of *k-probes*, which query whether a given set of nodes in the tree are mutually connected. Naturally, we seek to minimize the necessary number of probes which still accomplish a complete testing of the given tree (it is not necessary that the location of any fault be identified by the tests).

Our motivating application is the testing of electrical interconnections in multichip module (MCM) packaging technologies for high-complexity VLSI systems. MCMs allow multiple ''bare'' chips to be directly mounted, very close to one another, on a substrate that contains all the required interchip wiring. This results in a smaller area, reduced power requirements, and increased system speed, since interconnections on an MCM are much shorter than on printed circuit boards. Thus, MCMs have become very attractive for the packaging of today's highly complex VLSI systems.

In manufacturing an MCM, all the substrate interconnections must be tested before chips are attached, in order to avoid having to later discard good chips along with a faulty substrate onto which the chips are mounted. These substrate interconnections are modeled as trees, and their connectivity is verified by *probes* applied to the tree ''leaves''—the terminals to which the chip inputs and outputs are attached.

Production MCM substrate testers simultaneously move k ''flying'' probe heads to various locations in the circuit and verify electrical connectivity by measuring resistance and capacitance values at these locations [5]. This enables the detection of faults along the $\binom{k}{2}$ unique paths in the tree between the various pairs of probe heads. Figure 1 shows the unique path tested by a 2-probe (this path connects nodes ''A'' and ''B'').

Earlier probe-based testing methods [4, 9, 10], which

Correspondence to: G. Robins.

Contract grant sponsor: NSF; contract grant numbers: MIP-9110696; MIP-9257982; MIP-9457412.

Contract grant sponsor: ARO; contract grant number: DAAK-70-92-K-0001; contract grant number: DAAL-03-92-G-0050.

Contract grant sponsor: Packard Foundation: Fellowship for Science and Engineering.

CCC 0028-3045/98/030189-09

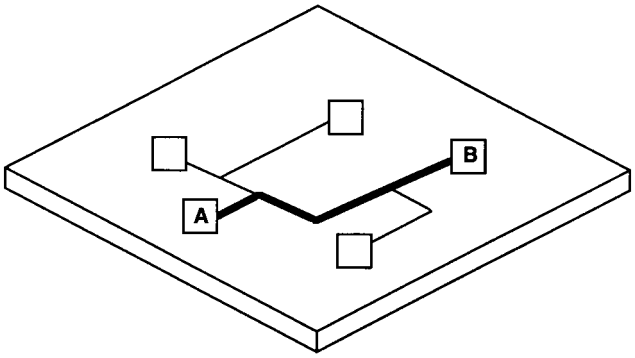

Fig. 1. Testing a tree network: the (A, B) probe tests the A–B path.

do not guarantee complete fault coverage, have been insufficient because the cost of an undetected fault (in terms of manufacturing time and wasted chips) is very high. Other methods for complete fault coverage are suboptimal with respect to the time required to generate the probe set [11]. In this paper, we give an algorithm that provides complete fault coverage, generates the optimal number of probes, and has optimal time complexity. Our results and discussion are for the case $k = 2$; however, extensions to $k > 2$ probe heads can easily be made. Empirical results from an implementation of our algorithms and benchmarks on actual MCM circuits are reported in [6–8]. Other methods for probe-based testing were independently investigated in [1–3, 12, 13].

2. DEFINITIONS

We begin by formally defining the trees that we will be testing, the operations that will be used to test them, and the two types of faults that we wish to discover.

Definition. A *tree* $T = (V, E)$ is an acyclic, undirected, connected graph consisting of a set of vertices V and a set of edges E. A given tree has l leaf vertices $L = \{p_1, p_2, \ldots, p_l\} \subseteq V$ and $|V| - l$ internal vertices.

Definition. A *rooted, oriented tree* has a distinguished root vertex, and the edges are all directed toward the root. We say that vertex v is the *child* of vertex u (and u is the *parent* of v) if there is a directed edge from v to u. Applying transitive reflexive closure to the parent and child relationships results in the *ancestor* and *descendant* relationships. In such a tree, the distance from a vertex v to any of its ancestors u is the number of edges along the directed path from v to u.

Definition. A *k-probe* consists of a set of k vertices whose mutual connectivity in T is checked. Thus, a 2-*probe*, (p_i, p_j) checks the unique path between p_i and p_j in T. A probe *succeeds* if the k vertices are mutually connected and *fails* otherwise.

Definition. Given a tree $T = (V, E)$, an *edge fault* at $e \in E$ removes e from T and thus separates T into the two subtrees, T_1 and T_2, incident to the endpoints of e.

Definition. Given a tree $T = (V, E)$, a *vertex fault* is a partition of an internal vertex $v \in V$ of degree d that separates T into a forest of $i \leq d$ connected subgraphs of T as follows:

- v is removed from T
- new vertices $\{v_1, \ldots, v_i\}$ are added to T
- each edge $e = \{u, v\}$ that was incident to v in T is replaced with $e' = \{u, v_j\}$ for some j, $1 \leq j \leq i$, and
- each member of $\{v_1, \ldots, v_i\}$ is incident to at least one such e'.

We say that a set of probes *detects all possible faults* of a given type when tree T contains a fault if and only if one or more of the probes fails. An edge fault that separates T into T_1 and T_2 can be detected by any probe (p_i, p_j) with $p_i \in T_1$ and $p_j \in T_2$. Detection of a vertex fault is more complicated since our definition embodies a physical model of vertex failure that is motivated by a *cracked via* [11] in a VLSI routing tree.* Figure 2 shows how a probe set that detects all possible edge faults may fail to detect a vertex fault; this phenomenon can occur at an internal vertex of degree 4 or higher.†

3. OPTIMAL EDGE FAULT TESTING BY 2-PROBES

In this section, we study the problem of edge fault detection:

The Edge Fault Detection Problem. Given an interconnection tree T with l leaves, determine a minimum-size probe set which detects all possible edge faults in T.

We observe that without loss of generality it suffices to probe only leaf vertices. Our algorithm for complete edge fault testing is quite simple: make an arbitrary in-order traversal of T starting from any internal vertex to order the leaves as $p_1, \ldots, p_l$ and then output the set of

* In integrated circuit routing, a *via* is an internal vertex of T that joins wires from two or more routing layers; see Figure 2.

† Observe that if faults at internal vertices do not have the ''physical'' traits of our model and are visible to all probe paths through the vertex then an algorithm which generates a probe set for edge-fault detection will automatically detect all possible vertex faults.

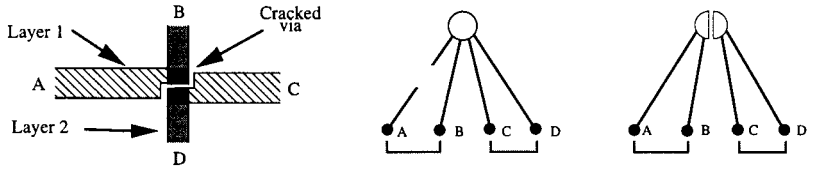

Fig. 2. A cross section of a via between two routing layers (left), an edge fault (middle), and a vertex fault (right) in a tree. The two probes (A, B) and (C, D) together detect any edge fault (middle), but cannot detect the vertex fault shown (right).

$\lfloor l/2 \rfloor$ probes $\{(p_i, p_{i+\lfloor l/2 \rfloor}) \mid 1 \leq i \leq \lfloor l/2 \rfloor\}$. If l is odd, also output the probe (p_1, p_l).

We call this algorithm PROBE1; Figure 3 illustrates its execution, while Figure 4 describes it formally. Clearly, PROBE1 runs in linear time, which is optimal. The following lemma and theorems show that PROBE1 produces a minimum size probe set for testing all edge faults.

Lemma 3.1. *A set of k-probes for detecting all edge faults in a tree T with l leaves has size at least* $\lceil l/k \rceil$.

Proof. Every edge which is incident to a leaf vertex must be tested, and any probe which tests it must have that leaf vertex as one of the probe locations. Since there are l leaves and at most k are tested by a single probe, at least $\lceil l/k \rceil$ probes are required for complete edge fault coverage. ■

Theorem 3.2. *For any edge* $e = \{v_i, v_{i'}\}$ *in T, PROBE1 outputs some probe which fails if there is an edge fault at e.*

Proof. Let the tree be rooted at an internal vertex, and let the leaves be labeled in in-order, exactly as done in algorithm PROBE1. Given an arbitrary edge $e = \{v_i, v_{i'}\}$ in the original tree T, where v_i is the parent of $v_{i'}$, let T' be the entire subtree of T rooted at $v_{i'}$. Note that since T and T' have distinct roots, T' is properly contained inside T (i.e., some leaves of T are not contained in T'). We claim that one of the probes generated by algorithm PROBE1 involves a leaf p_j in T' and another leaf p_k not in T', and we show that this probe (p_j, p_k) tests edge $e = \{v_i, v_{i'}\}$ for faults.

To see that this is true, assume toward contradiction that for every probe that involves a leaf of T', both of that probe's endpoints are leaves internal to T'. This implies that T' must contain the two middle leaves $v_{\lfloor l/2 \rfloor}$ and $v_{\lfloor l/2 \rfloor + 1}$, which, in turn, means that v_1 and v_l must also be in T'. Thus, T' coincides with T, contradicting the fact that T' is a proper subtree of T. Therefore, algorithm PROBE1 must have produced a probe involving a leaf p_j in T' and a leaf p_k not in T'. This probe forms the circuit $v_{i'}, \ldots, p_j, p_k, \ldots, v_m, \ldots, v_i, v_{i'}$, where v_m is the lowest common ancestor of both v_i and p_k; thus, the probe (p_j, p_k) tests the given arbitrary edge $e = \{v_i, v_{i'}\}$. ■

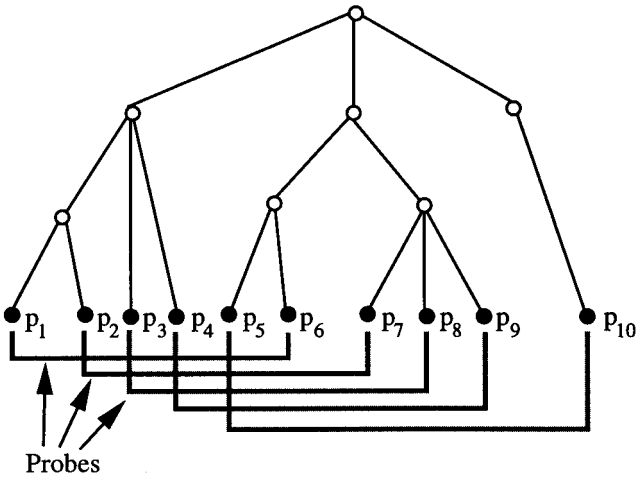

Fig. 3. A minimum probe set for detecting all possible edge faults.

Theorem 3.3. *Given a tree T with l leaves,* $\lceil l/2 \rceil$ *probes are necessary and sufficient for correct detection of all possible edge faults in T.*

Proof. Necessity follows from Lemma 3.1; correctness follows from Theorem 3.2, and the fact that no probe of T can fail if T is connected; and sufficiency follows from correctness and from the fact that a total of $\lceil l/2 \rceil$ probes are generated in lines 3 and 4 of PROBE1 in Fig. 4. ■

The extension of PROBE1 to handle k probes is straightforward, as follows. An in-order traversal is used to partition the leaves into k sets, each of size $\lfloor l/k \rfloor$ or $\lceil l/k \rceil$. Each probe consists of one leaf from each set (with one leaf from each of the size $\lfloor l/k \rfloor$ sets occurring a second time in the probe set and all others appearing once). The proof of the generalization of Theorem 3.2 for $k > 2$ still only requires us to show that one of the probes involving p_1 or p_l will fail.

4. OPTIMAL VERTEX AND EDGE FAULT TESTING BY 2-PROBES

We now study the problem of vertex (and edge) fault detection:

PROBE1: Computing a minimum probe set for edge fault detection	
Input: Tree $T = (V, E)$ with l leaves $p_1, p_2, \ldots, p_l \in V$	
Output: Minimum probe set which detects all possible edge faults	
1:	Root the tree arbitrarily at an internal vertex
2:	Label the leaves using an in-order traversal $p_1, \ldots, p_l$
3:	Output the set of probes $\{(p_i, p_{i+\lfloor \frac{l}{2} \rfloor}) \mid 1 \leq i \leq \lfloor \frac{l}{2} \rfloor\}$
4:	**If** l is odd **Then** output the probe (p_1, p_l)

Fig. 4. PROBE1: Generation of minimum probe set for edge-fault detection.

The Vertex and Edge Fault Detection Problem. Given an interconnection tree T with l leaves, determine a minimum-size probe set which detects all possible vertex and edge faults.

This section describes our PROBE2 algorithm, which produces in linear time the optimum number of 2-probes for complete detection of both vertex and edge faults. Recall from Lemma 3.1 that the number of leaves l in T induces a lower bound of $\lceil l/2 \rceil$ probes. We will see that PROBE2 very nearly achieves this lower bound: The algorithm generates at most $\max\{\lceil l/2 \rceil, d - 1\}$ probes, where d is the maximum degree of any vertex of T. Theorem 4.8 proves that this bound optimal.

As shown in Figure 2, testing paths through every edge incident to a vertex does not guarantee that all possible paths through the vertex are connected: probes (A, B) and (C, D) test all edges of the given vertex, but cannot tell us whether A and C are connected. When only edge faults are possible, as in Section 3, the faultless vertices become points through which one may assume connectivity to be transitive. Once we assume the vertex fault model, we must be certain that each of the $\binom{d}{2} = O(d^2)$ paths through a vertex of degree d is sound. Fortunately, Lemma 4.1 shows that only $O(d)$ probes are required to test the $O(d^2)$ paths.

Lemma 4.1. *Detection of a fault in a vertex v of degree d requires $1 + \lceil (d - k)/(k - 1) \rceil$ k-probes.*

Proof. We prove this lemma by induction on d.

Basis: For any d such that $1 \leq d \leq k$, exactly 1 probe is needed: We simply choose d leaves whose paths to each other must pass through the d edges incident to v, plus any $d - k$ additional leaves. Our formula holds since in this case $-1 < (d - k)/(k - 1) \leq 0$.

Induction: We assume that $d > k$ and that the induction hypothesis holds for all d' such that $1 \leq d' < d$. The first probe of v verifies that k edges incident to v are connected. The remaining $d - k$ edges must be shown to be mutually connected and, furthermore, must be shown to be connected to at least 1 of the first k edges probed. Thus, the remaining probes must show that $d - k + 1$ edges are connected. We then have that

$$1 + (1 + \lceil (d - k + 1 - k)/(k - 1) \rceil)$$
$$= 1 + (1 + \lceil (d - k - (k - 1))/(k - 1) \rceil)$$
$$= 1 + \lceil (d - k)/(k - 1) \rceil \text{ probes are required.} \quad \blacksquare$$

For 2-probes, Lemma 4.1 indicates that $d - 1$ probes are necessary to test a single vertex of degree d. For the graph consisting of d leaf vertices and a single internal vertex of degree d, the most straightforward way to perform this testing is to choose one special leaf vertex and issue the $d - 1$ probes which test its connectivity with each of the other $d - 1$ leaves.

This suggests that a single interior vertex v of degree d can be tested by dividing T into the d connected components, which result when v is removed from T, and then by issuing $d - 1$ probes which test a leaf of T in one of the components with a leaf of T from each of the other $d - 1$ components. A simple method based on this observation would call for separately creating probes of the $degree(v) - 1$ necessary paths through each vertex v resulting in a total of

$$\sum_{v \in V} degree(v) - 1 = -|V| + \sum_{v \in V} degree(v)$$
$$= 2*|E| - |V| = |V| - 2$$

probes. PROBE2 significantly improves upon this upper bound by taking advantage of two key ideas:

- probes may be generated which test paths through several vertices at once, and
- if two successful probes pass through the same *edge*, their leaves are mutually connected.

Both of these observations are employed in PROBE2 by ''passing messages'' containing each leaf name from the leaves of T up to a designated root. PROBE2 examines the local structure at internal vertices of T to decide how the leaf names listed in those messages should be either combined into probes or passed on toward the root to participate in subsequently-generated probes.

Algorithm PROBE2 (Fig. 5) first chooses an internal vertex R of maximum degree d and then roots the tree at R by orienting all edges toward R. Each leaf p_i is given the

PROBE2: Computing a minimum probe set for edge and vertex fault detection

Input: Tree $T = (V, E)$ with l leaves $p_1, p_2, \ldots, p_l \in V$
Output: Minimum probe set which detects all possible edge and vertex faults

1: $W \leftarrow V$
2: Find internal vertex $R \in W$ with maximum degree d
3: Root T by directing all edges towards R

/* Phase I: processing all internal vertices other than R */

4: **For** $i = 1$ to l, send message list $\{i\}$ from p_i to $parent(p_i)$
5: **While** $\exists v \in W$, $v \neq R$ having received message lists $M_1, \ldots, M_{degree(v)-1}$
6: **While** $\Sigma_{k=1}^{degree(v)-1} |M_k| > d$
/* note that this implies $\exists i$ such that $|M_i| \geq 2$ */
7: Choose arbitrary $x \in M_i$ for some M_i with $|M_i| \geq 2$
8: Choose arbitrary $y \in M_j$ for some $j \neq i$, $|M_j| \geq 1$
9: Output probe (p_x, p_y)
10: $M_i \leftarrow M_i - \{x\}$; $M_j \leftarrow M_j - \{y\}$
11: Concatenate message lists: $L \leftarrow M_1 \cup \ldots \cup M_{degree(v)-1}$
12: Send message list L to $parent(v)$
13: $W \leftarrow W - \{v\}$

/* Phase II: $W = \{R\}$; processing R which has received message lists $M_1, \ldots, M_d$ from its children */

14: **While** there are at least 2 nonempty message lists at R with one having size ≥ 2
15: Reorder $M_1, \ldots, M_d$ such that $|M_i| \geq |M_{i+1}|$ for all $1 \leq i < d$
16: Find maximum index $k \leq d$ such that $|M_k| > 0$ /* M_k is smallest non-empty list */
17: Choose arbitrary $x \in M_1$ with $y \in M_k$
18: Output probe (p_x, p_y)
19: $M_1 \leftarrow M_1 - \{x\}$; $M_k \leftarrow M_k - \{y\}$
20: Concatenate message lists: $L \leftarrow M_1 \cup \ldots \cup M_d$
/* Lemma 4.4 guarantees that at this point $0 \leq |M_k| \leq 1$ for all $1 \leq k \leq d$ */
21: **If** $|L| > 1$ **Then** output probes (p_{L_1}, p_{L_i}) $\forall$ $2 \leq i \leq |L|$ and terminate
/* note that L_k denotes the label at position k in the concatenated list L */
22: **Else** choose leaf vertex p_i such that i, L_1 were not both passed by the same child of R
23: Output probe (p_i, p_{L_1}) and terminate

Fig. 5. PROBE2: Optimal detection of all edge and vertex faults. See Fig. 6 for an execution example.

label i. The algorithm propagates *message lists* containing leaf names toward R, starting from the leaves in bottom-up order. Initially, each leaf p_i sends to its parent a message list containing only the label i. Phase I of the algorithm (lines 4–13 of Fig. 5) applies to internal vertices $v \neq R$. When such a vertex v has received message lists from all its children, it iteratively generates probes by pairing two labels from distinct incoming message lists as long as one of these lists is of size > 1; the two labels are then deleted from the respective lists. After the total number of labels in the incoming message lists has been reduced to d or less, all remaining labels are concatenated into a single message list that is passed up to v's parent. When only the root R remains unprocessed, Phase II (lines 14–23 of Fig. 5) applies a variation on the probe-matching method described above to the message lists at the root. Figure 6 traces the execution of PROBE2 on a small example.

We now prove a sequence of lemmas and theorems which lead to two main results, namely, that (1) algorithm PROBE2 outputs a probe set which detects all possible edge and vertex faults and (2) PROBE2 generates in optimal (i.e., linear) time the minimum possible number of probes. We begin with two lemmas which limit the size of the message lists passed up from each internal vertex and a third lemma which limits the number of labels in message lists present at line 20 of PROBE2.

Lemma 4.2. *No vertex passes a message list of size greater than d to its parent.*

Proof. By induction on the maximum distance to a vertex from any of its descendants.

Basis: If the distance is zero, then the vertex is a leaf and passes up a message list of size one.

Induction: We assume toward contradiction that vertex v passes up a message list containing more than d labels. By the induction hypothesis, each child of v passed up at most d labels, so the propagated labels must be from the message lists of two or more children. However, under

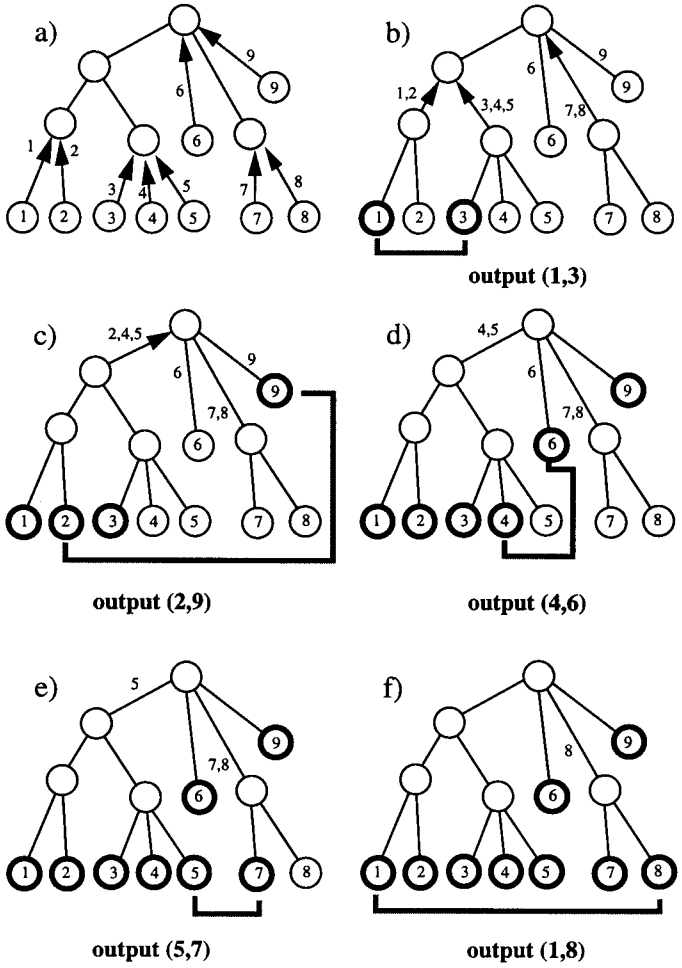

Fig. 6. Execution of PROBE2 on a tree containing nine leaves and five internal vertices; a total of five probes are generated (shaded arcs). Message lists passed up from nodes to their parents are displayed on the edges of T.

such conditions, the algorithm will continue to output probes that pair up labels from different message lists (lines 6–9 of Fig. 5) until the total size of the incoming message lists becomes d or less or until each child's message list has size one. In the first case, v sends up a list of size at most d. In the second case, v must have received message lists from at most $d - 1$ children and, hence, can pass up a message list of size at most $d - 1$. Either case provides our desired contradiction. ■

Lemma 4.3. *No internal vertex passes a message list of size less than $d - 1$ to its parent, unless this list contains the labels of all leaves that are descendants of that vertex.*

Proof. By induction on the maximum distance to a vertex from any of its descendants.

Basis: If the maximum distance is zero, then the vertex is a leaf and it passes up a message list of size one consisting of its only descendant, namely, itself.

Induction: Assume that vertex v passes up a message list of size i, where $i < d - 1$, and suppose toward contradiction that v has $> i$ descendants that are leaf vertices. The while statement at line 6 of Figure 5 can only reduce the total number of leaves present to $d - 1$ and, therefore, since $i < d - 1$, no probes could have been output at v. Hence, i is the sum of sizes of all message lists passed up by the children of v. Each of these children must have passed up a list of size no greater than i, and therefore by the induction hypothesis each child must have passed up a list containing the labels of all its leaf descendants. Thus, the list passed up from v is the union of the lists passed to v and contains the labels of all leaf vertex descendants of v. The size of the list passed up from node v is therefore equal to the number of its descendants, and we have our contradiction. ■

Lemma 4.4. *At line* 20 *of algorithm PROBE2, all M_i have size either* 0 *or* 1.

Proof. Suppose that when we reach line 20 of algorithm PROBE2 one of the message lists has size $|M_i| > 1$. There can be only one such M_i, else we would have

continued matching probes in the while loop at line 14. Consider the $d - 1$ probes generated at line 18 of algorithm PROBE2 which reduced the size of each of the other $d - 1$ message lists from 1 to 0. Each must have used one label from M_i, because M_i must always have been the largest message list, since no other list has size within one of $|M_i|$. Hence, M_i must initially have had size $\geq d + 1$, but this violates Lemma 4.2. ■

We are now ready to prove that PROBE2 generates a set of probes which test for all possible vertex and edge faults.

Lemma 4.5. *Let T be divided into connected components* $\{T_0, \ldots, T_j\}$ *by an arbitrary vertex fault at internal vertex* $v \neq R$. *Algorithm PROBE2 will issue at least one probe which involves leaves in different components* T_i, *thus detecting the vertex fault at* v.

Proof. We assume that the lemma is not true and show that this leads to a contradiction. We begin by noting that all leaves in any given message list are from the same component T_i. Without loss of generality, let T_0 be the component which contains T's root, R. Consider how the labels present at the children of v are either assembled into probes at v or passed on to v's parent. If any leaf is paired with a leaf in another component, then the fault must be discovered. If any leaf in some $T_i \neq T_0$ is passed up into v's message list to v's parent, then it will be paired with a leaf in T_0, and the fault will be discovered. Therefore, it must be the case that all leaves from any $T_i \neq T_0$ are paired only with leaves from the same T_i and none are passed up into v's list. Thus, all leaves initially present in T_i must be matched into probes at line 9 of algorithm PROBE2. In particular, since the message lists are not empty, there must be at least one match made, and, therefore, a *last* match made. However, this last match must leave at least one leaf in one of the message lists within T_i, since one of the lists involved in the match must have size at least 2. Thus, some labels from T_i are passed up to the next level, and this provides our contradiction. ■

Lemma 4.6. *Let T be divided into connected components* $\{T_0, \ldots, T_j\}$ *by an arbitrary vertex fault at the root vertex R. There will be at least one probe issued which probes leaves in different components* T_i, *thus detecting the vertex fault at* v.

Proof. Similar to the proof of Lemma 4.5 above, we note that every match made in line 18 of PROBE2 must be between labels from message lists within the same component and must leave at least one leaf present in the message list of each component T_i, since at least one of this lists considered for the match must have size 2 or more before the match. This means that at line 20 of algorithm PROBE2 L contains at least one label from each of the two or more components. Since the remaining probes all match to the same leaf, there will be at least one intercomponent match made. ■

Theorem 4.7. *Given an interconnection tree T, algorithm PROBE2 outputs a probe set which detects all possible edge and vertex faults.*

Proof. Lemmas 4.5 and 4.6 prove that all vertex faults at any internal vertex will be detected by the probes issued. To complete our proof that all vertex faults are covered, we need only note that all leaves appear in at least one probe, and any probe involving a given leaf vertex tests whether that vertex is connected to the rest of the tree. To show that all edge faults are covered, we note that every leaf name passed up through a given edge results in a probe that tests that edge, and all edges have at least one leaf name passed up through them. ■

Having shown that algorithm PROBE2 is correct, we now proceed to prove our claim that the number of probes generated by PROBE2 is optimal. We begin by formalizing our claim of a lower bound of $\max\{\lceil l/2 \rceil, d - 1\}$ on the number of probes necessary for complete fault testing.

Theorem 4.8. *It requires* $\max\{\lceil l/2 \rceil, d - 1\}$ 2*-probes to test a tree with l leaves and maximum vertex degree d for all possible edge and vertex faults.*

Proof. The lower bound of $\lceil l/2 \rceil$ follows directly from Theorem 3.3, and the lower bound of $d - 1$ from Lemma 4.1. ■

We now must prove that PROBE2 meets the bound of Theorem 4.8. Our only concern is that in line 21 of the code we generate $|L| - 1$ probes when the bound might allow us only $\lceil |L|/2 \rceil$ more. We therefore will show that when $|L| > 1$ it must be the case that $d - 1 > \lceil l/2 \rceil$ and that the total number of probes generated is $d - 1$. We begin with a series of lemmas relating the number of leaf labels arriving in message lists at the root to the number of leaves in the tree.

Lemma 4.9. *If* $2(i - 1)$ *or more labels are present among i nonempty message lists at the root R, and the difference in size between the two largest message lists is no more than* $i - 1$, *then either every label appears exactly once in the probe sequence or one label appears twice and all others appear once.*

Proof. By induction on i.

Basis: $i = 2$. There are two nonempty message lists at R, and either they are of the same size or one is one element larger than the other. If they both are of the same size s, their respective contents are matched by s probes (i.e., $s - 1$ probes output at line 18, plus one probe output

at line 21), and no label is repeated. If the sizes are s and $s + 1$, then some label will appear a second time (in a probe output at line 23) in order to match the single remaining label after s probes have been made at line 18.

Induction: It must be the case that either one of the initial nonempty lists has size 1 or that after some number of matches at line 18 the smallest of the lists has been reduced to size 1. At this point, there must still be at least $2(i - 1)$ labels distributed among the message lists. The next match leaves $i - 1$ nonempty message lists and a total number of leaves of at least $2((i - 1) - 1)$. The difference in sizes between the two largest message lists can never increase and, in this last match, must have decreased by one unless these sizes were already equal. Hence, this difference is now at most $(i - 1) - 1$. We may therefore invoke the induction hypothesis for $i - 1$. ■

Lemma 4.10. *If $2(i - 1) - j$ labels are present among i nonempty message lists at the root R for $j \geq 0$, then one of these labels will appear $j + 1$ times in the ensuing sequence of probes and all other labels will appear exactly once.*

Proof. By induction on i.

Basis: If $i = 2$ and $j = 0$, then we have two singleton messages lists which are matched at line 21, and all labels appear exactly once.

Induction: We know that there must be some list of size one, else we would have more than $2(i - 1)$ labels present. If there is a message list of size greater than one, we match one of its labels with a list of size one (line 18) and invoke the induction hypothesis for $2((i - 1) - 1) - j$ labels over $(i - 1)$ nonempty lists. If none of the nonempty lists has size greater than one, then all have size one and $2(i - 1) - j = i$, implying $i = j + 2$. We then must use one of the labels $i - 1 = j + 1$ times (i.e., $|L| = i$ in line 21) and all others only once. ■

Lemma 4.11. *If fewer than $2(d - 1)$ labels arrive in the messages lists at root R of degree d, then the vertices represented in those messages are the set of all leaves of T.*

Proof. Since the d children of the root must each pass up at least one leaf in its message list, the fact that there are strictly less than $2(d - 1)$ labels implies that no incoming message list at R can be of size $d - 1$ or greater. Applying Lemma 4.3 then gives the desired result. ■

Theorem 4.12. *Algorithm PROBE2 generates $\max\{\lceil l/2 \rceil, d - 1\}$ probes for a tree T with l leaves and maximum vertex degree d, which is the minimum number of probes necessary to detect all possible edge and vertex faults.*

Proof. As usual, we use l to denote the number of leaves in T. If $2(d - 1)$ or more labels arrive at the root R, then by Lemma 4.9 we have either l or $l + 1$ labels used in the sequence of what must be $\lceil l/2 \rceil$ probes. If $2(d - 1) - j$ labels arrive at the root for $j > 0$, then by Lemma 4.11 we have $l = 2(d - 1) - j$, and by Lemma 4.10, the sequence of probes contains $l + j = 2(d - 1)$ labels for a total of $d - 1$ probes. This is optimal by Theorem 4.8. ■

The time complexity of PROBE2 is also optimal: each vertex v passes no more than d labels to its parent, and thus each vertex will receive fewer than d^2 labels from its children. Assuming that d is a constant (dependent on the VLSI technology of our application area), the amount of processing at each vertex is a constant, and since each vertex is processed only once, the overall time complexity of algorithm PROBE2 is linear in the size of the input.

In considering the vertex and edge-fault detection problem for k-probes where $k > 2$, the following two observations are helpful. (1) At an internal vertex $v \neq R$, each k-probe need not test leaves from k different message lists—probing leaves from two different lists is enough to gain useful connectivity if we leave at least one leaf from one of the lists unmatched after the probe. (2) Rather than passing $d - 1$ leaf names up in its message list, each vertex should, if possible, pass on at least $k(1 + \lceil (d - k)/(k - 1) \rceil) - (d - 1)$ leaves to its parent. This ensures that the k leaves needed for each of the $1 + \lceil (d - k)/(k - 1) \rceil$ probes required to test the root will be available if all other message lists sent to the root have size 1. Benchmarks of this method on actual MCM circuits, along with issues related to efficient probe scheduling, are reported in [6–8].

5. CONCLUSION

We have studied the testing of an arbitrary interconnection tree topology using *probes* which verify the correctness of paths within the tree. Our formulation captures the increasingly important VLSI application of multichip module substrate testing and involves an interesting ''physical'' model of vertex failure in the connection topology. We have presented linear-time algorithms which yield minimum probe sets that detect all possible edge and vertex faults. Remaining areas of investigation include

- Characterizing the sets of probes that completely detect all edge/vertex faults in a given interconnection tree (the output of our algorithms are usually just one of many possible solutions);
- Obtaining a variant of PROBE2 which does not rely on rooting the tree at a vertex of maximum degree; and
- Extending the problem formulation to the testing of general graph topologies (e.g., with each probe opera-

tion verifying the *k*-connectivity between groups of graph vertices).

We thank Professor David Cantor for his insightful comments on an earlier draft of this paper. We also thank Professor C. K. Cheng for bringing the problem to our attention. We appreciate the help of Douglass Bateman, Sarah Friend, and Chris Helvig with proofreading. Additional related papers may be found at http://www.cs.virginia.edu/~robins/ and http://vLsicad.cs.ucla.edu/~abk/. A.B.K. was supported by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050. G.R. was supported by a Packard Foundation Fellowship and by NSF Young Investigator Award MIP-9457412. E.A.W. was supported by an NSF Graduate Fellowship.

REFERENCES

[1] R. Carragher, N. C. Chou, C. K. Cheng, and T. Russell, Distortion mapping for cofired ceramic substrate testing. *Proceedings of the International Symposium on Microelectronics* (Nov. 1993) 295–300.

[2] N. C. Chou, C. K. Cheng, and T. Russell, Dynamic probe scheduling optimization for MCM substrate test. *IEEE Trans. Comp. Hybrids Mfg. Tech.* (1994), 182–189.

[3] N. C. Chou, C. K. Cheng, and T. C. Russell, High-performance microelectronic substrate verification using probe testers. *Proceedings of the IEEE International ASIC Conference*, Rochester, NY (Sept. 1992) 230–233.

[4] J. C. Crowell, R. Keogh, and J. Conti, Moving probe bare board tester offers unlimited testing flexibility. *Ind. Elect. Equip. Design* (1984).

[5] C. Hilbert and C. Rathmell, Design and testing of high density interconnection substrates. *Proceedings NEPCON West* (1990).

[6] A. B. Kahng and G. Robins, On optimal interconnections for VLSI. Kluwer, Boston, MA (1995).

[7] A. B. Kahng, G. Robins, and E. A. Walkup, New results and algorithms for MCM substrate testing. *Proceedings of the IEEE International Symposium on Circuits and Systems*, San Diego, CA (May 1992) 1113–1116.

[8] A. B. Kahng, G. Robins, and E. A. Walkup, Optimal algorithms for substrate testing in multi-chip modules. *Int. J. High-Speed Electr. Syst.* **6** (1995) 595–612.

[9] W. H. Kautz, Testing for faults in wiring networks. *IEEE Trans. Comp.* **C-23** (1974) 358–363.

[10] B. McWilliams, private communication (invited talk at CANDE meeting), San Marcos, CA (April 1991).

[11] S. Z. Yao, N. C. Chou, C. K. Cheng, and T. C. Hu, A multi-chip module substrate testing algorithm. *Proceedings of the IEEE International ASIC Conference*, Rochester, NY (Sept. 1991) P9:4.1–P9:4.4.

[12] S. Z. Yao, N. C. Chou, C. K. Cheng, and T. C. Hu, An optimal probe testing algorithm for the connectivity verification of MCM substrates. *Proceedings of the IEEE International Conference on Computer-Aided Design*, Santa Clara, CA (Nov. 1992) 264–267.

[13] S. Z. Yao, N. C. Chou, C. K. Cheng, and T. C. Hu, A multi-probe approach for MCM substrate testing. *IEEE Trans. Computer-Aided Design* **13** (1994) 110–121.