VLSI Design
1993, Vol. 1, No. 1, pp. 71–85
Reprints available directly from the publisher
Photocopying permitted by license only

Coverage of Node Shorts Using Internal Access and Equivalence Classes

WARREN H. DEBANY, JR.
Rome Laboratory (RL/ERDA), Griffiss AFB, New York, USA

A method is presented that determines the coverage of shorts (bridging failures) in digital logic circuits by internal access test techniques. These are test techniques that provide observability of circuit nodes, such as CMOS power supply current monitoring (including IDDQ), CrossCheck, and voltage contrast. Only fault-free circuit simulation is used to obtain node states. Two versions of the algorithm are presented: a simple algorithm that is suitable for use with two-state logic (0 and 1), and a more general algorithm for four-state logic (0, 1, *X*, and *Z*). The result is a set of sets of nodes, where a list of all potential shorts that could exist in the circuit yet be undetected after testing is obtained easily from the power sets of these sets; unlike other approaches the full universe of potential shorts is not generated. Experiments show that short, randomly generated sequences of test vectors detect essentially all detectable shorts of multiplicity 2 for both combinational and sequential circuits.

Key Words: *Digital logic testing; Shorts testing; Bridging failures; Power supply current monitoring; IDDQ; Fault coverage*

It is known that *shorts,* or *bridging failures,* are a common type of failure in VLSI circuits. A short is an unwanted conducting path between nodes of a circuit. A study that performed failure analysis of integrated circuits (MOS 4-bit microprocessors) found that 55% of the failures involved a short [1]. Ferguson and Shen studied the effects of CMOS integrated circuit defects such as pinholes, extra metal, and extra polysilicon. For one circuit, a 4 × 4 multiplier, they found that 48% of the defects caused shorts to occur [2].

In this paper, *logic lines* are the connections between logic gates through which signals flow. A set of (intentionally) connected logic lines constitutes a *logic node. Electrical nodes* include both logic nodes and connections made inside logic gates. The *multiplicity* of a short is defined to be the number of nodes that are shorted together. A *single* bridging failure is a *single* instance of shorted nodes; many nodes may be involved. The shorting of two nodes, *a* and *b*, is denoted by "*ab*," and the shorting of three nodes *a*, *b*, and *c*, is denoted by "*abc*."

The exact behavior of a short is difficult to predict. Williams and Angell [3], in the same paper that introduced scan design, proposed that shorts involving two nodes could be modeled as having either *wired-AND* behavior (where a node in the 0 state would dominate a node in the 1 state) or *wired-OR* behavior (where a node in the 1 state would dominate a node in the 0 state). This is a valid assumption for low-resistance shorts in some circuit technologies and most published work so far is based on the "wired" behavior model for shorts. CMOS is currently the dominant digital circuit technology due to its low power, high speed, and high levels of integration, but the "wired" behavior model is not valid for CMOS bridging failures. Soden and Hawkins have shown that the variable number of active pullup and pulldown transistors cause the behavior of a CMOS bridging failure not to be predictable by the "wired" model [4]. Therefore, most of the bridging failure analyses published to date are not applicable to CMOS. The method of this paper, however, based as it is on "internal access" test techniques, is applicable to CMOS testing.

Because of the importance of shorts as a failure mode in integrated circuits a great deal of work has been done in determining how to detect shorts. Just as for the detection of stuck-at (i.e., stuck-at-zero and stuck-at-one) faults, detection of a short requires

two conditions to occur: *activation* and *propagation*. A bridging failure is *activated* when at least one of the shorted nodes is caused to carry an incorrect logic value due to contention. The effects of the incorrect logic value must then be *propagated* to an observable output of the circuit.

Kodandapani and Pradhan [5] determined conditions under which test sets designed to detect stuck-at faults also detect shorts, and when shorts are undetectable. Using the "wired" model, they showed that it is possible for an undetectable bridging failure to mask an otherwise detectable stuck-at fault. They constrained their problem to the case where no feedback is introduced by a bridging failure in a combinational circuit.

Abramovici and Menon [6] derived relations between stuck-at faults and bridging failures, based on the "wired" model, such that the results of stuck-at fault simulation could be used to determine the detection of bridging failures without explicit simulation of the bridging failures (see the section on the Necessity to Consider Shorts of Multiplicities Greater Than 2). They considered the case where a bridging failure introduces feedback in a combinational circuit and derived necessary and sufficient conditions for detection by a single test vector. An interesting finding was that a set of tests with high stuck-at fault coverage does not necessarily have high bridging failure coverage.

INTERNAL ACCESS VS. EXTERNAL ACCESS TEST TECHNIQUES

It is important to understand the costs associated with any fault detection problem. There are three broad categories of test-related costs involved in testing a microcircuit:

- Test generation: the development of the test stimuli and responses; a non-recurring cost.
- Test grading: the determination of the coverage or quality of the tests; a non-recurring cost.
- Test application: the process of testing an individual microcircuit; a recurring cost.

Test generation has been shown to be an NP-Hard problem [7]. While the problem of simply activating a fault (either a stuck-at fault or a short) is itself NP-Hard, in practice it is the *propagation* of the fault's effects to the primary outputs that is the more difficult task in test generation [8].

Similarly, the high computational costs of *grading the fault coverage of a test* (usually by means of fault simulation) are due in large part to the propagation of fault effects through the model of the logic circuit. It has been shown that the computational effort involved in fault simulation, under what amount to *favorable* circumstances, grows at most as the square of the size of the circuit [9]. Unfortunately, this bound is easily achievable when fault grading a circuit that uses test compaction (such as signature analysis) where a single signature is read only at the end of the test. On the other hand, fault simulation can execute at nearly the speed of good circuit simulation if there is good observability of internal logic signals and "fault dropping" is used.

When a large number of circuits are involved, *test application* cost is the most important from the point of view of life-cycle cost. For digital circuits, test application cost is frequently measured in terms of the number of test vectors or number of bits of information that must be stored and applied/measured. These quantities can then be related directly to time and data storage requirements.

Most work in logic circuit fault detection is based on the assumption that the "normal" primary outputs of the circuit are the only points where the effects of a fault can be observed. Test points (additional primary inputs and/or outputs) are often introduced into a design in order to improve controllability and observability. Scan design can be thought of as providing control and observation points at other than the usual primary inputs and outputs—specifically, the inputs and outputs of the combinational logic in a full-scan design can be considered to be "virtual" primary inputs and outputs for the purpose of testing. However, this simply reduces the complexity of testing a sequential logic circuit to that of testing a combinational circuit (plus some overhead), which is the *starting* point for most fault detection studies and proofs of computational complexity, particularly in the case of detection of shorts.

Techniques have been developed that provide visibility, either virtually or directly, to the logic node or electrical node level. In this paper, such techniques are referred to as *internal access*; "traditional" techniques that use only the ordinary primary outputs to observe fault effects are referred to as *external access*. Four internal access techniques are discussed here.

Levi [10] proposed an internal access method for detecting faults in CMOS circuits where the power supply current (I_{DD}) is monitored during test application. Levi discussed the classes of CMOS failures that are detectable by this approach (some of which are detectable by no other procedure). In particular, if a short is activated in a CMOS circuit, then there

is a low-resistance path from V_{DD} to V_{SS}, resulting in a large I_{DD}. A short may be detectable even if contention does not cause an incorrect logic value to occur. The power supply terminals can be considered to be additional primary outputs, where fault effects from every electrical node are "directly" observable. Measurement of I_{DD}, particularly the *quiescent* current, is becoming a standard procedure [11]; this method is often referred to as "IDDQ" testing. Some Automatic Test Equipment (ATE) vendors have introduced features that assist in applying this technique.

CrossCheck Technology, Inc. has developed an internal access method that consists of adding a test point to the output of every gate in a CMOS logic circuit [12]. It allows "random access" of any addressable node in a circuit. The voltage at a node is sensed, compared to a threshold, and reported. Reporting is done either on a node-by-node basis or by means of a compressed output response produced by a linear-feedback shift register. This approach provides "virtual" observability of every logic node through a four-wire IEEE 1149.1-like test bus interface.

A third internal access technique uses electron-beam voltage contrast. The CADET system [13], developed by Rockwell International for US Army LABCOM, performs direct node-by-node voltage sampling by means of beam positioning and comparison to waveforms generated by circuit-level simulation.

The three techniques mentioned to this point provide observability to either the *logic node* or *electrical node* level in a circuit; they are capable of detecting an activated bridging failure even when it does not exhibit "wired" behavior.

In this paper two measures of cost are used: number of *steps* and number of *node tests* required for bridging failure detection. These terms, as they are used in this paper, are defined as follows:

- A *step* is defined to be a test vector on which a measurement is made.
- A *node test,* or more briefly a *test,* is defined to be where a measurement is made of the state of a node.

A measurement is not necessarily made on each test vector (i.e., not every test vector is designated as a step). However, if the circuit is sequential it may not be possible to delete test vectors that are not steps because they may be required to effect state changes.

When detecting bridging failures by means of I_{DD} measurement the cost of fault detection is related to the number of steps. Nodes are not tested individually by I_{DD} measurement because the effects of *all* activated bridging failures are observable at a single point: a power supply line. When using the CrossCheck approach or CADET the cost is related to the number of individual node tests made.

The three internal access test techniques discussed so far apply to the problem of testing individual microcircuits. When testing a circuit board (such as a printed circuit or wire-wrap board) many interconnections between microcircuits are accessible. Techniques such as bed-of-nails and guided-probe testing constitute a fourth type of internal access testing and the methods presented here can be used to grade the coverage of bridging failures involving nodes accessible at the board level and higher levels of assembly.

In this paper it is assumed that the circuit is quiescent during the time that a measurement is made. It is also assumed that some form of "traditional" external access testing is performed, in addition to internal access testing, to verify at least the gross behavior of the circuit and to detect stuck-at faults. This paper does not consider the possible invalidation of a measurement due to the introduction of feedback causing oscillations or by reaching an incorrect quiescent "machine state."

ON THE NECESSITY TO CONSIDER SHORTS OF MULTIPLICITIES GREATER THAN 2

A short may involve more than two nodes, as when closely spaced bus lines are involved [14, 15], but most of the literature that addresses the detection of shorts (by means of traditional external access testing) generally explicitly states or implicitly assumes that it is sufficient to consider only bridging failures of multiplicity 2. This assumption would be valid only if the following conjecture were true:

Conjecture (It is sufficient to consider bridging failures if only multiplicity 2): If bridging failure f is detected, then bridging failure f' is also detected, where f is any single bridging failure of multiplicity 2 and f' is any single bridging failure of multiplicity greater than 2 such that both nodes shorted in the presence of f are also shorted in the presence of f'.

Counterexample: Consider the logic circuit shown in Figure 1. Table I lists the logic states present on each node of the logic circuit in the following three cases:

- No short exists.
- Nodes a and b are shorted (bridging failure f).

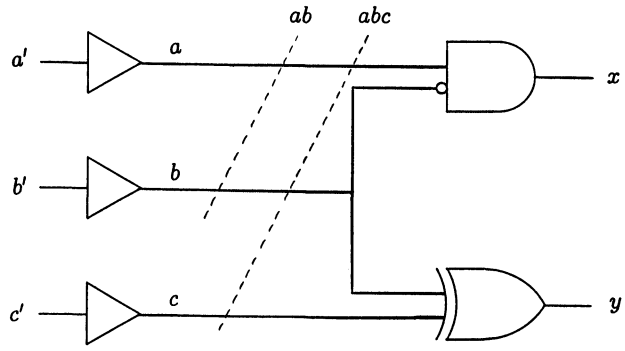

FIGURE 1 Sample circuit with bridging failures *ab* (multiplicity 2) and *abc* (multiplicity 3).

- Nodes a, b, and c are shorted (bridging failure f').

Assume that both bridging failures exhibit wired-AND behavior. Because neither *ab* nor *abc* introduces feedback and the good circuit is strictly combinational, we need consider only the eight possible vectors that can be applied to the primary inputs: a', b', and c'. Note that if fault detection is based only on errors observable at the primary outputs, x and y, then both *ab* and *abc* are detectable by at least one vector each.

If the conjecture were true, then any test vector that detects *ab* would detect *abc* as well. However, inspection of Table I reveals that vector 011 detects the multiplicity 2 bridging failure *ab* yet *does not detect* the multiplicity 3 bridging failure *abc*. Thus, at least for external access test techniques, this counterexample contradicts the general statement that it is sufficient to consider only multiplicity 2 bridging failures. □

On the other hand, for each of the three internal access test techniques considered in this paper it is easy to show (at least in the absence of feedback) that if any bridging failure f of multiplicity 2 is detected, then, *a fortiori,* any other bridging failure of multiplicity greater than 2 in which both nodes shorted in f also are shorted, would also be detected.

TABLE I
Node States for Circuit Shown in Figure 1, in the Presence of Bridging Failures *ab* and *abc*. Assuming Wired-AND Behavior, Test Vector 011 Detects *ab* Yet Does Not Detect *abc*

a'	b'	c'	a	b	c	x	y
0	0	0	0	0	0	0	0
0	0	1	0	0	1/1/0	1/1/0	0
0	1	0	0	1/0/0	0	1/0/0	0
0	1	1	0	1/0/0	1/1/0	0/1/0	0
1	0	0	1/0/0	0	0	0	1/0/0
1	0	1	1/0/0	0	1/1/0	1/1/0	1/0/0
1	1	0	1/1/0	1/1/0	0	1/1/0	0
1	1	1	0	0	0	0	0

"0" means that a state is 0 in the good circuit and in the presence of *ab* and *abc*
"1/1/0" means that the good state is 1, the state in the presence of *ab* also is 1, and the state in the presence of *abc* is 0

The number of potential bridging failures in a logic circuit is vastly larger than the number of stuck-at faults. For a model of a logic circuit, let L be the number of logic lines, N be the number of nodes, and m be the multiplicity of a single bridging failure. Then the number of single bridging failures of multiplicity m is given by $\binom{N}{m}$. For large N and small m, $\Sigma_{i=2}^{m} \binom{N}{i}$ as a function of N is $O(N^m)$. In contrast, the number of single stuck-at faults is only $2L$, without employing any fault collapsing. (Although $L \geq N$, usually L is not much greater than N.) Thus, even if bridging failures of only multiplicity 2 are considered the number of bridging failures is far larger than the number of stuck-at faults. Because it is not sufficient in general to consider only bridging failures of multiplicity 2, as proved by the counterexample above, it is impractical to use any technique that relies on explicit *listing,* much less *simulation,* of the universe of bridging failures.

Example 1: Consider the 54LS181 four-bit arithmetic logic unit (ALU). The 54LS181 logic model used in this example is composed of 112 logic gates. There are 126 distinct nodes and 261 logic lines in the 54LS181. Using the stuck-at fault model, the 261 logic lines correspond to 522 stuck-at faults that must be considered for fault grading or test generation. The number of bridging failures of multiplicity 2 is 7,875. Considering bridging failures of small multiplicities greater than or equal to 2, say 2–5, there are 2.55×10^8 potential bridging failures.

Sinha and Bhattacharya [15] have considered the problem of counting the number of *multiple* bridging failures, of given multiplicities ≥2, that can occur in a circuit. They defined the quantity "$N_{mg}(k)$" to be the number of multiple bridging failures of multiplicities 2, 3, . . . , k that can exist in a circuit with k nodes. Some values of $N_{mg}(k)$ are as follows: $N_{mg}(5) = 51$, $N_{mg}(10) = 115{,}974$, and $N_{mg}(15) = 1{,}382{,}958{,}544$.

These examples demonstrate that, considering only single bridging failures of multiplicity 2, and assuming that simple "wired" behavior is an accurate fault model, it is impractical to simulate the effects of every bridging failure in a logic circuit of "useful" size. Even if one assumes that shorts occur only between physically adjacent nodes, the number of potential bridging failures is still astronomical, and in any case such information is not known before layout

is complete (for example, [16]). It is futile, therefore, to attempt to grade a test vector sequence for coverage of shorts by simulating the effect of *every* potential short.

One alternative to full fault simulation is to use *fault sampling* to estimate a lower bound on bridging failure coverage, or to establish that the coverage is greater than some specified value [17]. Another alternative, proposed by Abramovici and Menon [6], uses fault simulation for stuck-at faults to determine bridging failure detection. For example, in the absence of feedback introduced by a short in a combinational circuit and assuming wired-AND behavior, bridging failure *ab* is detected by a test vector if and only if *a* has the state 0 and *b* stuck-at-zero is detected, or *b* has the state 0 and *a* stuck-at-zero is detected ([6, Theorem 1]). Using Abramovici and Menon's technique it is unnecessary to explicitly *simulate* the effect of each bridging failure, but it is still necessary to list every potential bridging failure and analyze each one by applying the requisite relations to the results of stuck-at fault simulation.

ALGORITHM FOR TWO-STATE LOGIC

A technique is presented in this section that determines the coverage of shorts by internal access methods. This technique requires only fault-free circuit simulation, instead of fault simulation, and requires *no explicit listing* of the universe of potential bridging failures. This technique reveals the sets of nodes that may be shorted together, in *any* multiplicity, that would be undetected by a test vector sequence.

The principle used is to establish *sets* of logic nodes that share some property. The notation used here is based on that of Preparata and Yeh [18].

Assume that nodes can have only two states: 0 and 1. For any two nodes *a* and *b* in a model of a logic circuit *C*, and a test sequence consisting of *n* or more vectors applied to *C*, define a relation ϵ_n such that

$$\text{“}a\epsilon_n b\text{”}$$

denotes the statement

“*a* has the *same* logic value as *b* for every test vector $1, 2, \ldots, n$”

It is easily shown that ϵ_n is reflexive, symmetric, and transitive, and so it is an *equivalence relation*. On any vector *n*, if all nodes of *C* are partitioned into *sets of nodes* such that ϵ_n is true for every pair of nodes within each set, and ϵ_n is false for every pair of nodes that are not in the same set, then each such set forms an *equivalence class*. (When nodes are *partitioned* into classes, each node appears in exactly one class only.)

Consider two nodes *a* and *b*. Define a *single difference* to be fault activation where $a = 0$ and $b = 1$, or $a = 1$ and $b = 0$, on some vector; a difference is needed to occur on only one vector. Define *both differences* to be fault activation where $a = 0$ and $b = 1$ on some vector, and $a = 1$ and $b = 0$ on another vector; both conditions must occur. In this paper, it is assumed that a “single difference” is necessary and sufficient for the detection of a short, if all nodes involved are tested on the appropriate step. The problem where “both differences” are necessary and sufficient has also been solved, but is not addressed in this paper.

Assuming internal access and “single difference,” this paper shows how sets of nodes are *refined* and *interpreted* on each vector. Lower and upper bounds are obtained for both the number of *steps* and the number of *tests* required, with respect to a given test vector sequence, and a coverage metric based on multiplicity 2 bridging failures is presented.

The equivalence classes have a direct interpretation in terms of bridging failure detection. An equivalence class containing exactly one node name, *a*, means that node *a* has been shown to have had a *different state from every other node at some point* during the application of the test vector sequence; therefore, *any* short involving *a* would have been activated and thus detected. An equivalence class containing more than one node, say three nodes *a*, *b*, and *c*, means that:

- Potential shorts that could exist without being detected include: *ab*, *bc*, *ac*, and *abc*, and
- Neither *a* nor *b* nor *c* can be shorted to any node in any *other* equivalence class without that condition being detected.

Thus, an equivalence class containing *N* nodes implies that there are $2^N - N - 1$ (i.e., the number of sets of cardinality 2 or greater in the power set of the equivalence class) potential shorts that would be undetected.

In this paper shorts are considered only between logic signals. All stuck-at-zero and stuck-at-one faults can be accounted for as well by adding two more nodes, V_{DD} and V_{SS}, to the set of nodes considered. However, this is not done in the algorithms or examples presented here.

```
procedure SHORT_GRADE2
/* S is a matrix of sets of node names, such
that S[i,j] is the j^th set of node names on the i^th test vector */
/* initialize: */
S[0,1] ← {the set of all node names} /* a priori,
    any node could be shorted to any other, so the initial equivalence
    class before the 1^st test vector is applied contains all node names */
s ← 1 /* initially, there is one equivalence class */
steps ← 0
tests ← 0
v ← {number of test vectors in the test vector sequence}
for n ← 1 to v do
    /* refine the equivalence classes, list nodes to be tested, and
       update counts of steps and tests */
    REFINE2(n, steps, tests, s)
repeat /* i */
{Print number of steps and tests}
/* s is the number of sets generated on the last test vector, v */
for j ← 1 to s do
    {Print the node names in set S[v,j]}
       /* potential undetected shorts can exist only between nodes
          in the same set, not between nodes in different sets */
repeat /* j */
end SHORT_GRADE2
```

FIGURE 2 Procedure SHORT_GRADE2.

Procedure SHORT_GRADE2 (Figure 2) and its subprocedure REFINE2 (Figure 3) grade a test sequence for bridging failure coverage. SHORT_GRADE2 generates the equivalence classes for two-state logic, with respect to ϵ_n, and assumes that "single difference" fault activation is sufficient for bridging failure detection. A more general algorithm that applies to four-state logic (where 0, 1, X, and Z are permitted) is introduced in the section on Algorithm for Four-State Logic. The notation used to document these algorithms is based on SPARKS [19]. In the version of the notation used here, operations that are better described in terms of English language descriptions, rather than in terms of SPARKS operations, are set off in braces({. . .}). Comments are set off using "/*" and "*/" as delimiters.

In this version of the algorithm all sets from previous steps are retained. A more efficient implementation would perform set operations *in situ* and eliminate the need to perform superfluous operations such as explicit set copies and deletions. Other data structures and operations may be more applicable, depending on the language of implementation, and on speed/storage tradeoffs.

It is now shown that SHORT_GRADE2 is an algorithm in that it produces a correct result in a finite number of steps.

Outline of proof: Both the number of nodes N and the number of test vectors are finite. Subprocedure REFINE2 is called once for each test vector. The number of sets manipulated by REFINE2 on any test vector is bounded above by N because each set contains no more than one instance of a node name and no two sets contain the same node name. Similarly, any single "set operation" involves manipulating no more than N node names. Therefore, the algorithm terminates in a finite number of steps.

Correctness of the refinement into equivalence

```
subprocedure REFINE2(n, steps, tests, s)
/* n : current test vector number
steps : total number of steps, so far, on which I_DD must be measured
tests : total number of nodes that must be tested, so far, using
    CrossCheck or CADET
s : entering, the number of sets generated on the previous step;
    exiting, the number of sets generated on the current step
T : the set of "nodes to be tested" on this test vector (if any)
did_split_set : TRUE if and only if at least one set was split
    on this test vector */
T ← ∅ /* initially, the set of "nodes to be tested" is the empty set */
did_split_set ← FALSE /* no split yet */
old_s ← s
s ← 0
for i ← 1 to old_s do
    /* logic states on test vector n are known for all nodes; apply the
       equivalence relation ε_n in order to refine the equivalence classes
       from those of the previous test vector */
    if {Some node in S[n-1,i] has the state 0 and another node
            has the state 1 on test vector n} then
        did_split_set ← TRUE
        S[n,s+1] ← {all nodes in S[n-1,i] that have the state 0}
        S[n,s+2] ← {all nodes in S[n-1,i] that have the state 1}
            /* no node can simultaneously have the state 0 and 1,
               so no two sets can contain the same node name
               on test vector n */
        s ← s + 2 /* two new sets were created */
        {Add all nodes that are in S[n,s+1] to set T}
        {Add all nodes that are in S[n,s+2] to set T}
    else /* i.e., all nodes in S[n-1,i] have the same state
        (all zeros or all ones) on test vector n */
        S[n,s+1] ← S[n-1,i] /* just copy */
        s ← s + 1 /* only one new set was created (which is merely
            a copy of a set from the previous test vector); no nodes are
            added to set T in this case */
    endif
repeat /* i */
```

FIGURE 3 Subprocedure REFINE2.

classes on each step can be shown by a trivial inductive proof based on the properties of the equivalence relation ϵ_n.

A step contributes to bridging failure detection if and only if at least one equivalence class is "split" on that step. Similarly, the only nodes that need to be tested on any step are those contained in an equivalence class that is "split" on that step. □

Example 2: Table II demonstrates a case where a logic circuit contains five nodes (a, b, c, d, e) and SHORT_GRADE2 is used to determine the bridging

```
if did_split_set = TRUE then
    steps ← steps + 1
    tests ← tests + {number of node names in T}
    {Print n} /* test vector n is a necessary step */
    {Print the names of the nodes in T} /* these are the only nodes
        that need to be tested on this step — the state (0 or 1) being
        tested is assumed to be known by the tester */
endif
return
end REFINE2
```

FIGURE 3 Subprocedure REFINE2 (continued).

failure coverage in response to some test vector sequence. The set refinements, counts of steps, and counts of tests are those obtained *after* the application of each test vector. Sets of nodes that were "split" are flagged by an asterisk (*); the union of these sets is the set of "nodes to be tested," T.

COVERAGE METRIC FOR TWO STATES

Consider only single bridging failures of multiplicity 2. For a logic circuit with N nodes there is a universe of $\binom{N}{2}$ bridging failures of multiplicity 2 that must be considered, and an equivalence class containing N' node names implies that there are $\binom{N'}{2}$ potential shorts that would not be detected. This observation leads to the following theorem:

Theorem 1: Following the application of a test vector sequence to a logic circuit with N nodes, denote by s the number of equivalence classes produced by SHORT_GRADE2. Denote by N_i the number of node names in the i^{th} equivalence class. The fraction of single bridging failures of multiplicity 2 that are detected by internal access by the test vector sequence is given by

$$1 - \frac{\sum_{i=1}^{s} \binom{N_i}{2}}{\binom{N}{2}}$$

where $\binom{x}{y} \triangleq 0$ for $x < y$. □

(It is convenient to define $\binom{x}{y} = 0$ for $x < y$, as is done for the boundary conditions in the construction of Pascal's Triangle.) Extension of Theorem 1 to the case of bridging failures of multiplicity greater than 2 is straightforward.

Example 3: Applying Theorem 1 to the data provided in Example 2, the coverage of bridging failures of multiplicity 2 on each test vector is shown in Table III.

BOUNDS ON STEPS AND TESTS FOR TWO STATES

SHORT_GRADE2 results in a minimum of both steps and tests when all sets are "split" into equal

TABLE II
Example of Application of SHORT_GRADE2. Sets of Nodes That Must be Tested are Flagged by "*"

Test Vector	a	b	c	d	e	Sets	Total Steps	Total Tests
"0"	—	—	—	—	—	$\{a, b, c, d, e\}$	0	0
1	0	0	0	1	1	$\{a, b, c\}^*, \{d, e\}^*$	1	5
2	0	0	1	0	0	$\{a, b\}^*, \{c\}^*, \{d, e\}$	2	8
3	1	1	1	0	0	$\{a, b\}, \{c\}, \{d, e\}$	2	8
4	0	1	1	0	1	$\{a\}^*, \{b\}^*, \{c\}, \{d\}^*, \{e\}^*$	3	12

TABLE III
Bridging Failure Coverage Calculated Using Theorem 1 for Sets Listed in Table II

Test Vector	Sizes of Sets	Calculation	Coverage
1	3, 2	$1 - [\binom{3}{2} + \binom{2}{2}]/\binom{5}{2}$	0.6
2	2, 1, 2	$1 - [2\binom{2}{2} + \binom{1}{2}]/\binom{5}{2}$	0.8
3	2, 1, 2	$1 - [2\binom{2}{2} + \binom{1}{2}]/\binom{5}{2}$	0.8
4	1, 1, 1, 1	$1 - [4\binom{1}{2}]/\binom{5}{2}$	1.0

halves on each test vector. (Note that this is not necessarily the only case where the minima occur, particularly when the number of node names in a set is not an integral power of 2.) SHORT_GRADE2 results in a maximum of both steps and tests when only one node is "split" from any equivalence class on any test vector. Based on these two cases, lower and upper bounds on steps and tests can be derived. Proof that these conditions result in the actual minima and maxima is rather involved and is omitted.

Theorem 2: Let N denote the number of nodes in a logic circuit. Define $M = \lceil \log_2 N \rceil$. ($\lceil x \rceil$ denotes the *ceiling* function: the smallest integer greater than or equal to x. M satisfies the inequalities $2^{M-1} < N \leq 2^M$.) The lower bounds (*LB*s) and upper bounds (*UB*s) on steps and tests that result from the application of SHORT_GRADE2 are:

LB on *steps*: M

LB on *tests*: $(M + 1)N - 2^M$

UB on *steps*: $N - 1$

UB on *tests*: $(N^2 + N - 2)/2$ □

These bounds are not necessarily achievable when SHORT_GRADE2 is applied to any given logic circuit, but they are achievable when the requisite node states are available.

SHORT_GRADE2 always takes the first "opportunity" to derive bridging failure detection information from the given test vector sequence. The number of steps and tests can be reduced in many cases by applying SHORT_GRADE2 to a permutation of the initial test vector sequence. That is to say, the test information contained in the node states is independent of the actual order of application of the test vectors, and some orderings result in fewer tests or steps being required to achieve the same result.

EXPERIMENTAL RESULTS FOR COMBINATIONAL CIRCUITS

In this section results of experiments on combinational circuits are presented. The circuits are taken from Advanced Micro Devices, Brglez and Fujiwara, and Texas Instruments [20–22]. It is irrelevant that many of these devices are implemented in bipolar ("TTL") circuit technology. The determination of bridging failure coverage based on the method of this paper relies only on the gate-level logic schematic and is independent of the circuit technology.

One hundred randomly generated test vector sequences were applied to the logic models of the circuits-under-test and evaluated for shorts coverage. Only two-state good circuit simulation was used in order to gather node states. For the ISCAS circuits the 100 test vector sequences were drawn from limited "pools" of 5,000 test vectors. For the non-ISCAS circuits, the sequences were chosen to be long enough to guarantee that each of the 100 sequences detected every detectable bridging failure; the value stated for the coverage is the *best* that any test vector sequence could achieve by measuring every node on every test vector. For the ISCAS circuits the coverage is the *minimum* obtained from 100 sequences of 200 test vectors each.

Table IV summarizes the results of the experiments. For each circuit the minimum, maximum, and

TABLE IV
Results of Applying SHORT_GRADE2 to Combinational Circuits. Averages are Rounded to Three Decimal Digits of Precision. Coverages are Rounded to Five Decimal Digits of Precision

Circuit	Nodes	Min Steps	Max Steps	Ave Steps	Ave as % of *LB*	Min Tests	Max Tests	Ave Tests	Ave as % of *LB*	Min Coverage (100 Exper)
54LS182	28	11	18	14.0	279%	140	195	158	116%	100.0%
54LS147	40	10	15	12.7	211%	206	267	227	105%	98.718%
54LS85	42	11	19	14.7	245%	238	302	263	114%	99.884%
54LS261	48	14	22	17.6	294%	281	346	308	113%	99.291%
25S05	113	16	25	20.4	291%	783	924	817	105%	99.747%
54LS181	126	17	28	22.2	317%	876	1124	933	106%	99.543%
25LS2517	154	21	35	27.6	345%	1235	1567	1370	121%	99.975%
C432	196	26	40	30.1	413%	1602	1819	1700	113%	99.948%
C499	243	26	41	33.4	418%	2257	2698	2430	126%	98.796%
C880	443	34	54	42.8	476%	4333	5384	4840	123%	99.790%
C1355	587	34	50	43.2	432%	7001	8180	7540	139%	99.033%
C1908	913	23	40	32.3	323%	8575	10030	9410	104%	99.094%
C2670	1502	43	57	50.9	463%	16707	18666	17500	110%	99.895%

average, for the 100 sequences, are reported for steps and tests. The average is also expressed as a percentage of the *LB* stated in Theorem 2. The "coverage" is for bridging failures of multiplicity 2 calculated on the basis of the sizes of the final equivalence classes in accordance with Theorem 1. Coverage is less than 100% in most cases because of logic constraints in the circuits—if two nodes always have the same logic state (such as is the case with the input and output nodes associated with a noninverting buffer), then it is not possible to detect a short between those nodes based on the fault activation assumption of this paper.

Observe that the *maximum* values for steps and tests are relatively close to the theoretical *lower bounds* stated in Theorem 2. The average values for steps range from 2.11 to 4.76 times the theoretical *LB*. For tests, the average values range from 1.05 to 1.39 times the theoretical *LB*.

For C2670 (the largest combinational circuit tested here) the averages are only 3.39% and 1.55% of the theoretical *upper bounds* for steps and tests, respectively.

The number of steps is always less than or equal to the number of test vectors, because not every vector necessarily contributes to bridging failure detection. The number of tests is always less than or equal to the product of the number of steps and the number of nodes, because not every node must be tested on every step. For the 54LS181, the first of the 100 test vector sequences graded by SHORT_GRADE2 resulted in 22 steps. The 22nd step occured on the 70th test vector. That is, no further bridging failure detection was achieved beyond test vector 70 (out of a possible 2^{14} = 16,384 vectors). Multiplying 22 steps by 126 nodes yields 2,142 possible node tests, but less than half that number of tests were required (only 899 tests).

ALGORITHM FOR FOUR-STATE LOGIC

In this section an extension is made to the two-state coverage algorithm that allows the use of four-state (0, 1, *X*, and *Z*) logic simulation. This permits the application of the principles of this paper to *sequential logic*.

For any two nodes *a* and *b* in a model of a logic circuit *C*, and a test vector sequence consisting of *n* or more vectors applied to *C*, define a relation γ_n such that

"$a\gamma_n b$"

denotes the statement

"*a* is not 0 when *b* is 1, and *a* is not 1 when *b* is 0, on any test vector 1, 2, . . . *n*"

It is easy to show that γ_n is reflexive and symmetric. However, γ_n is not transitive (that is, $a\gamma_n b$ and $b\gamma_n c$ do not imply $a\gamma_n c$). Therefore, γ_n is only a *compatibility relation* [18]. On any vector *n*, if all nodes of *C* are distributed into *sets of nodes* such that γ_n is true for every pair of nodes within each set, then each such set forms a *compatibility class*. Of course, if nodes take only the states 0 and 1, then γ_n and ϵ_n yield the same result, and the compatibility classes are thus also equivalence classes.

It is important to distinguish between *partitioning* and *distributing* nodes into classes. When nodes are partitioned each node appears in exactly one class; when nodes are distributed a node can appear in more than one class.

Procedure SHORT_GRADE4 generates the compatibility classes for four-state logic and assumes that "single difference" fault activation is sufficient for bridging failure detection. The text of the top-level procedure is identical to that of SHORT_GRADE2, except that subprocedure REFINE4 is called instead of REFINE2, so it is not reiterated here. The text of REFINE4 is shown in Figure 4.

It can be shown that SHORT_GRADE4 is an algorithm in the sense that it produces a correct result (based on the properties of the compatibility relation γ_n) and that it terminates in a finite number of steps, but the proof is omitted here.

Because the transitive property is lacking in this case, a node may appear in more than one compatibility class. Therefore, the interpretation of the compatibility classes that result from SHORT_GRADE4 is more complex than the interpretation of the equivalence classes that result from SHORT_GRADE2. Define the set *U* to be the union of all compatibility classes in which node name *a* appears. Then

- Node *a* may be shorted to any other node in *U* without that condition being detected, and
- Node *a* cannot be shorted to any node that is not in *U*, without that condition being detected.

Example 4: Table V demonstrates a case where a logic circuit contains five nodes (*a*, *b*, *c*, *d*, *e*) and SHORT_GRADE4 is used to determine the bridging failure coverage in response to some test vector sequence. The set refinements and counts of steps and tests shown are those that are obtained *after* the application of each test vector. Because some nodes have the state *X*, for convenience the set of "nodes

```
subprocedure REFINE4(n, steps, tests, s)
/* variables are identical to those used in REFINE2 */
T ← ∅ /* initially, the set of "nodes to be tested" is the empty set */
did_split_set ← FALSE
old_s ← s
s ← 0
for i ← 1 to old_s do
    /* logic states on test vector n are known for all nodes; apply the
       equivalence relation γn in order to refine the equivalence classes
       from those of the previous test vector */
    if {Some node in S[n-1,i] has the state 0, and another node
            has the state 1, on test vector n} then
        did_split_set ← TRUE
        S[n,s+1] ← {all nodes in S[n-1,i] that have the state 0, X, or Z}
        S[n,s+2] ← {all nodes in S[n-1,i] that have the state 1, X, or Z}
        /* nodes with state X or Z are placed in both S[n,s+1] and S[n,s+2] */
        s ← s + 2 /* two new sets were created */
        {Add all nodes that are in S[n,s+1], and have the state 0, to set T}
        {Add all nodes that are in S[n,s+2], and have the state 1, to set T}
    else /* i.e., all nodes in S[n-1,i] are 0, X, or Z, or are 1, X, or
        Z, on test vector n */
        S[n,s+1] ← S[n-1,i] /* just copy */
        s ← s + 1 /* only one new set was created (which is merely
            a copy of a set from the previous test vector);
            no nodes are added to set T in this case */
    endif
repeat /* i */
/* in S[n,1],S[n,2],...S[n,s] there may be duplicate sets and
    sets that are proper subsets of other sets */
{Retain only one copy of any duplicated sets in S[n,1],S[n,2],...S[n,s]}
{Delete any sets that are proper subsets of any other set in
    S[n,1],S[n,2],...S[n,s]}
s ← {the number of remaining sets}
if did_split_set = TRUE then
    steps ← steps + 1
    tests ← tests + {number of node names in T}
    {Print n} /* test vector n is a necessary step */
    {Print the names of the nodes in T} /* these are the only nodes
        that need to be tested on this step — the state (0 or 1) being
        tested is assumed to be known by the tester */
endif
return
end REFINE4
```

FIGURE 4 Subprocedure REFINE4.

TABLE V
Example of Application of SHORT_GRADE4. T is the Set of "Nodes to Be Tested" on Each Step

Test Vector	a	b	c	d	e	Sets	Total Steps	Total Tests	T
"0"	—	—	—	—	—	$\{a, b, c, d, e\}$	0	0	$\{\emptyset\}$
1	0	1	X	X	X	$\{a, c, d, e\}, \{b, c, d, e\}$	1	2	$\{a, b\}$
2	0	X	1	X	X	$\{a, d, e\}, \{b, c, d, e\}$	2	4	$\{a, c\}$
3	0	X	X	1	X	$\{a, e\}, \{b, c, d, e\}$	3	6	$\{a, d\}$
4	X	0	1	X	X	$\{a, e\}, \{b, d, e\}, \{c, d, e\}$	4	8	$\{b, c\}$
5	X	0	X	1	X	$\{a, e\}, \{b, e\}, \{c, d, e\}$	5	10	$\{b, d\}$
6	0	0	1	0	1	$\{a\}, \{b\}, \{c, e\}, \{d\}$	6	15	$\{a, b, c, d, e\}$
7	X	X	1	X	0	$\{a\}, \{b\}, \{c\}, \{d\}, \{e\}$	7	17	$\{c, e\}$

to be tested," T, is shown explicitly on each test vector.

COVERAGE METRIC AND BOUNDS FOR FOUR STATES

Because SHORT_GRADE4 does not necessarily *partition* nodes into classes, strictly speaking the simple coverage metric presented in Theorem 1 cannot be applied in the four-state case unless it happens that no node name appears in more than one compatibility class. Fortunately, as is demonstrated in the section on Experimental Results for Sequential Circuits, this appears to be the rule rather than the exception when SHORT_GRADE4 is applied to sequential circuits.

When a node name appears in more than one compatibility class, it is still possible to calculate a coverage metric but it is then necessary to *list* the remaining potential bridging failures in order to avoid counting any bridging failure more than once in enumerating the undetected bridging failures. However, even if it is necessary to calculate coverage by listing the potential bridging failures, in practice this set generally becomes fairly small after the application of only a few test vectors. This is best explained by means of an example.

Example 5: Referring to Table V, generated in Example 4, consider the compatibility classes that result on test vector 4: $\{a, e\}$, $\{b, d, e\}$, and $\{c, d, e\}$. Compatibility class $\{a, e\}$ implies that bridging failure ae would be undetected. Compatibility class $\{b, d, e\}$ implies that bridging failures bd, be, and de would be undetected. Compatibility class $\{c, d, e\}$ implies that bridging failures cd, ce, and de would be undetected. Bridging failure de appears twice, so there are only six unique potential bridging failures that would be undetected after applying test vectors 1–4 in Example 4. Thus, coverage of single bridging failures of multiplicity 2 in this case is $1 - 6/\binom{5}{2} = 1 - 6/10 = 0.4$.

An alternative is to simply avoid considering X and Z states. In many cases, it is possible to avoid the need to list the bridging failures by allowing SHORT_GRADE4 to use only test vectors for which all nodes have only 0 and 1 states.

A third alternative for determining the numerical coverage is to "blindly" apply Theorem 1 to the compatibility classes that result from SHORT_GRADE4. If any node names appear in more than one compatibility class, then the resulting value will at worst *underestimate* the actual coverage. One risk in relying on this method is that the coverage may be reported as being less than zero! However, this simple approach is the one used in the section on Experimental Results for Sequential Circuits and no problems were encountered in its application.

The *LB*s for both steps and tests that result from SHORT_GRADE4 coincide with those obtained from SHORT_GRADE2. The minima are achieved when all sets are "split" into equal halves on each test vector. For the *UB*s it appears that the worst case is where only two nodes are tested on each step. A five-node example is shown in Table VI. While this paradigm can be shown by exhaustion in small cases to provide the maxima for steps and tests, it has not actually been proven to provide the maxima for all cases. Therefore, although the following assertion is stated as a theorem, the expressions for the *UB*s are only *conjectures*:

Theorem 3: Let N denote the number of nodes in a logic circuit. Define $M = \lceil \log_2 N \rceil$. The lower bounds (*LB*s) and upper bounds (*UB*s) on steps and tests that result from the application of SHORT_GRADE4 are:

LB on *steps*: M
LB on *tests*: $(M + 1)N - 2^M$

TABLE VI
Example of Application of Upper Bound on Steps and Tests for SHORT_GRADE4. T is the Set of "Nodes to Be Tested" on Each Step

Test Vector	*a*	*b*	*c*	*d*	*e*	Sets	Total Steps	Total Tests	T
"0"	—	—	—	—	—	{*a*, *b*, *c*, *d*, *e*}	0	0	{∅}
1	0	1	X	X	X	{*a*, *c*, *d*, *e*}, {*b*, *c*, *d*, *e*}	1	2	{*a*, *b*}
2	0	X	1	X	X	{*a*, *d*, *e*}, {*b*, *c*, *d*, *e*}	2	4	{*a*, *c*}
3	0	X	X	1	X	{*a*, *e*}, {*b*, *c*, *d*, *e*}	3	6	{*a*, *d*}
4	0	X	X	X	1	{*a*}, {*b*, *c*, *d*, *e*}	4	8	{*a*, *e*}
5	X	0	1	X	X	{*a*}, {*b*, *d*, *e*}, {*c*, *d*, *e*}	5	10	{*b*, *c*}
6	X	0	X	1	X	{*a*}, {*b*, *e*}, {*c*, *d*, *e*}	6	12	{*b*, *d*}
7	X	0	X	X	1	{*a*}, {*b*}, {*c*, *d*, *e*}	7	14	{*b*, *e*}
8	X	X	0	1	X	{*a*}, {*b*}, {*c*, *e*}, {*d*, *e*}	8	16	{*c*, *d*}
9	X	X	0	X	1	{*a*}, {*b*}, {*c*}, {*d*, *e*}	9	18	{*c*, *e*}
10	X	X	X	0	1	{*a*}, {*b*}, {*c*}, {*d*}, {*e*}	10	20	{*d*, *e*}

UB on *steps*: $(N^2 - N)/2$

UB on *tests*: $N^2 - N$ □

EXPERIMENTAL RESULTS FOR SEQUENTIAL CIRCUITS

Table VII summarizes the results of applying SHORT_GRADE4 to sequential circuits. Circuit USFX contains structures used in a study of fault simulators [23]. PRSR is a linear-feedback shift register structure used to investigate how logic simulators propagate uninitialized logic signal values. The other test cases are ISCAS sequential circuits [24] where the *d*-flip-flops are implemented as 10-NAND gate-equivalent circuits.

For each circuit, a test vector sequence was applied and node states were obtained. The sequences applied to USFX and PRSR were quite short and exercised the circuits only minimally. The sequences applied to the ISCAS circuits consisted of 5,000 pairs of vectors, where each pair of vectors applied random data with the clock low and then high. Next, 100 sequences of 200 vector pairs each were selected at random and graded by SHORT_GRADE4. The "random ordering" is only with respect to the order in which SHORT_GRADE4 considers the test vectors, because the calculated node states would be invalid if the vectors were applied in a different order. The external access stuck-at fault coverages ranged from approximately 60% down to nearly 0. However, as is shown in Table VII, bridging failure coverage is extremely high in every case.

The types of statistics listed in Table VII are the same as those for the combinational circuit data presented in the section on Experimental Results for Combinational Circuits. Only in the case of PRSR did any node name appear in more than one final compatibility class. Coverage was calculated using Theorem 1 in every case, so the coverage stated for PRSR underestimates the actual coverage. The coverage is the *minimum* obtained from the 100 sequences considered for each circuit.

TABLE VII
Results of Applying SHORT_GRADE4 to Sequential Circuits. Averages are Rounded to Three Decimal Digits of Precision. Coverages are Rounded to Five Decimal Digits of Precision

Circuit	Nodes	Min Steps	Max Steps	Ave Steps	Ave as % of *LB*	Min Tests	Max Tests	Ave Tests	Ave as % of *LB*	Min Coverage (100 Exper)
USFX	111	13	19	15.7	225%	683	971	807	106%	97.281%
PRSR	113	15	43	18.4	263%	770	2338	847	109%	94.343%
S208	188	13	22	17.8	222%	1206	2003	1590	111%	83.610%
S298	263	16	30	22.5	250%	1980	3280	2590	122%	87.827%
S344	320	29	44	36.6	407%	2880	3811	3300	123%	99.453%
S386	227	21	36	27.4	343%	1790	2794	2250	126%	94.913%
S444	395	19	30	24.2	269%	3043	5864	4400	128%	83.082%
S641	605	39	60	50.8	508%	6807	9710	8360	148%	97.806%
S713	619	40	60	50.9	509%	7143	10278	8770	152%	97.366%
S1238	703	67	88	76.0	760%	9851	13735	11800	176%	99.081%

As was the case with the combinational circuits, the *maximum* values for steps and tests for the sequential circuits are fairly close to the theoretical *lower bounds* stated in Theorem 3. The average values for steps range from 2.22 to 7.60 times the theoretical LB. For tests, the average values range from 1.06 to 1.76 times the theoretical LB. For S1238 (the largest sequential circuit tested here) the averages are only 0.03% and 2.40% of the theoretical *upper bounds* for steps and tests, respectively.

CONCLUSION

This paper has presented a technique for determining the coverage of bridging failures by internal access test techniques. The technique involves partitioning nodes into equivalence classes (for two-state logic) or distributing nodes into compatibility classes (for four-state logic). In either case, the possibility of an undetected short exists only between nodes in the same equivalence class or compatibility class. For two-state logic, a simple multiplicity 2 bridging failure coverage metric is presented based on the sizes of the equivalence classes. For four-state logic, the coverage metric may require listing the set of potential bridging failures, but this set quickly becomes very small.

Experimental data are presented that demonstrate that the average number of steps or tests required for bridging failure detection, based on internal access, is not much greater than the theoretical minimum.

Experiments showed also that the coverage of bridging failures, even by short randomly generated test vector sequences, is extremely high for both combinational and sequential circuits. This suggests that internal access test techniques make explicit test vector generation for bridging failures unnecessary.

Acknowledgments

The author wishes to thank Bob Lipp for posing the question that prompted the investigations described here. Chuck Hawkins and Mark Levi reviewed earlier versions of this paper and made invaluable comments.

References

[1] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical Versus Logical Fault Models MOS LSI Circuits: Impact on Their Testability," *IEEE Transactions on Computers,* 527–531, June 1980.

[2] F.J. Ferguson and J.P. Shen, "Extraction and Simulation of Realistic CMOS Faults Using Inductive Fault Analysis," *Proceedings, International Test Conference,* 475–484, 1988.

[3] M.J.Y. Williams and J.D. Angell, "Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Transactions on Computers,* 46–60, January 1973.

[4] J.M. Soden and C.F. Hawkins, "Electrical Properties and Detection Methods for CMOS IC Defects," *Proceedings, 1st European Test Conference,* 159–167, 1989.

[5] K.L. Kodandapani and D.K. Pradhan, "Undetectability of Bridging Faults and Validity of Stuck-at Fault Test Sets," *IEEE Transactions on Computers,* 55–59, January 1980.

[6] M. Abramovici and P.R. Menon, "A Practical Approach to Fault Simulation and Test Generation for Bridging Faults," *IEEE Transactions on Computers,* 658–663, July 1985.

[7] O.H. Ibarra and S.K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Transactions on Computers,* 242–249, March 1975.

[8] H. Fujiwara, "Computational Complexity of Controllability/Observability Problems for Combinational Circuits," *IEEE Transactions on Computers,* 762–767, June 1990.

[9] P. Goel, "Test Generation Costs Analysis and Projections," *Proceedings 17th Design Automation Conference,* 77–84, 1980.

[10] M.W. Levi, "CMOS is Most Testable," *Proceedings, IEEE International Test Conference,* 217–220, 1981.

[11] R.R. Fritzemeier et al., "Increased CMOS IC Stuck-at Fault Coverage With Reduced I_{DDQ} Test Sets," *Proceedings, International Test Conference,* 427–434, 1990.

[12] M. Carroll, "Built-in Array Payoff: Better Fault Detection," *High Performance Systems,* 28–44, August 1989.

[13] E.B. Hakim and R.G. Sartore, "Microelectronic Design Validation and Fault Diagnostics Using Electron-Beam Testing," *Proceedings, GOMAC,* 385–388, 1988.

[14] C.R.P. Hartmann et al., "Fault Tolerant VLSI Design Using Error Correcting Codes," RADC-TR-88-321, February 1989.

[15] B.P. Sinha and B.B. Bhattacharya, "On the Numerical Complexity of Short-Circuit Faults in Logic Networks," *IEEE Transactions on Computers,* 186–190, February 1985.

[16] P. Nigh and W. Maly, "Layout-driven Test Generation," *Proceedings, International Conference on Computer-Aided Design (ICCAD),* 1989.

[17] W.H. Debany et al., "Fault Coverage Measurement for Digital Microcircuits," MIL-STD-883 Test Procedure 5012, Rome Laboratory (RL/ERDA), Griffiss AFB NY, 18 December 1989 (Notice 11) and 27 July 1990 (Notice 12).

[18] F.P. Preparata and R.T. Yeh, *Introduction to Discrete Structures for Computer Science and Engineering.* Reading, MA: Addison-Wesley, 1973.

[19] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms.* Rockville, MD: Computer Science Press, 1978.

[20] Advanced Micro Devices (AMD), *Bipolar Microprocessor Logic and Interface Data Book,* 1985.

[21] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN," *Proceedings, International Symposium on Circuits and Systems (ISCAS),* 1985.

[22] Texas Instruments (TI), *The TTL Data Book for Design Engineers,* Second Ed. 1976.

[23] S.A. Al-Arian et al., "Fault Simulator Evaluation," RADC-TR-89-230, November 1989.

[24] F. Brglez, D. Bryan, and K. Kozminski, "Combinational

Profiles of Sequential Benchmark Circuits," *Proceedings, International Symposium on Circuits and Systems* (*ISCAS*), 1989.

Biography

WARREN H. DEBANY, JR. is with the U.S. Air Force Rome Laboratory. He heads the Microcircuit Simulation and Testability Group in the Microelectronics Reliability Division. He received a B.S. in Electrical Engineering from the State University of New York at Buffalo and an M.S. in Computer Engineering and Ph.D. in Computer and Information Science from Syracuse University. His current research interests include digital logic modeling, test generation, the development and assessment of design-for-testability and built-in-test techniques, and testability measurement. He is a Senior Member of the IEEE, and is a licensed Professional Engineer in the state of New York.