# Generation of Functional Broadside Tests for Transition Faults

Irith Pomeranz, *Fellow, IEEE*, and Sudhakar M. Reddy, *Life Fellow, IEEE*

***Abstract*—Scan design allows a circuit to be tested using states that the circuit cannot enter during functional operation. It was observed that nonfunctional operation during testing may cause excessive currents that can cause a good chip to fail the test because of voltage droops caused by the excessive current demand. A good chip may also fail due to the propagation of signal transitions along nonfunctional long paths, especially during at-speed testing. This problem is studied in this paper in the context of tests for transition faults. A method for determining transition faults that are untestable under functional operation-conditions is described. Two procedures for generating transition-fault tests that use only functional operation conditions are also described. The first procedure accepts as input a broadside test set for transition faults. The second procedure accepts as input a test sequence for the nonscan circuit. Although such a test sequence is more complex to generate and simulate, it results in higher numbers of faults detected under functional operation conditions.**

***Index Terms*—Broadside tests, overtesting, reachable states, transition faults.**

## I. Introduction

SCAN DESIGN [1], [2] allows a synchronous sequential circuit to be brought to states that the circuit cannot visit during functional operation. As a result, it allows the circuit to be tested using test vectors that are not applicable during functional operation. It was observed in [3] that tests using nonfunctional operation may cause excessive switching activity, leading to excessive peak-current demands. Excessive currents may cause supply-voltage droops that increase circuit delays, leading to good chips failing at-speed tests [3]. Considering delay faults, faults that can only be detected during nonfunctional operation do not affect circuit performance during normal operation. It was observed in [4] that tests, which detect delay faults under nonfunctional operation conditions, may fail defect-free chips if the tests propagate signal transitions along circuit paths that cannot be activated during functional operation, and whose delays are larger than the clock period. These observations lead to the need to test delay faults under functional operation conditions in order to avoid unnecessary yield loss.

Manuscript received March 16, 2005; revised July 13, 2005. The work of I. Pomeranz was supported in part by the Semiconductor Research Corporation (SRC) under Grant 2004-TJ-1244. The work of S. M. Reddy was supported in part by SRC under Grant 2004-TJ-1243. This work was presented in part at the Proceedings of Design Automation Conference, June 2004, pp. 928–933. This paper was recommended by Associate Editor S. Hellebrand.

I. Pomeranz is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA.

S. M. Reddy is with the Electrical and Computer Engineering Department, University of Iowa, Iowa City, IA 52242 USA.

Digital Object Identifier 10.1109/TCAD.2005.860959

In this paper, we consider the following issues related to testing for transition faults under functional operation conditions. In Section II, we define functional circuit operation as operation that occurs within the set of reachable states of the circuit. Different definitions of functional operation and methods to compute functional tests can be found in [5]–[7]. In Section III, we discuss a complete method to derive broadside tests for transition faults that use only reachable states. We refer to such tests as functional broadside tests. Based on this method, we describe in Section III a procedure for determining transition faults that cannot be detected by functional broadside tests. The procedure is based on earlier procedures presented in [8] and [9].

In Sections IV and V, we discuss two heuristic methods to derive functional broadside tests for transition faults. The first method accepts a broadside transition-fault test set $C$ that may include unreachable states. It uses a simulation-based process to determine reachable states that will replace unreachable states in $C$. The second method extracts functional broadside tests from test sequences for the nonscan sequential circuit. It can be applied using deterministic or random test sequences. Experimental results are included in Sections IV and V, demonstrating the advantages of both methods. While the first method does not require test generation or fault simulation for the nonscan circuit, the second method results in a higher number of detected faults. In Section VI, we discuss the fault coverage of the resulting test sets, and their use as part of a test set that detects all the transition faults that are detectable using scan.

## II. Functional Operation

Transition faults are used for modeling delay defects. In standard scan design, two methods to test for transition faults are used: skewed load [10] and broadside [11]. In both methods, the initial state for the test is scanned in. Since any arbitrary initial state may be scanned in, the circuit may be tested using states that cannot be visited during normal functional operation.

Functional operation typically starts after the circuit is brought to a known state by reset, by loading the initial state from an external source, or by applying a synchronizing sequence. For the purpose of the discussion in this paper, functional operation of the circuit is defined as operation starting from a reachable state. A reachable state is a state that can be reached from all the circuit states (or from the all-unspecified state). If the circuit is synchronizable, the synchronization state is a reachable state since it can be reached from every state by applying the synchronizing sequence. In addition, all the states

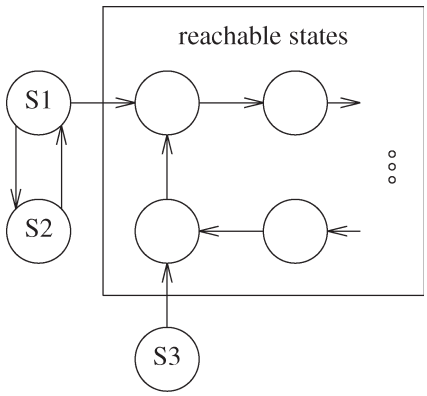



Fig. 1. Example state diagram.

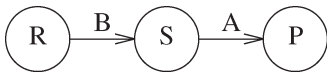



Fig. 2. Example of reachable states.

that the circuit can visit after it is synchronized are reachable states. If reset is used, or if the initial state is loaded from an external source, the initial state is a reachable state, and all the states that the circuit can visit after it is initialized are reachable states. Fig. 1 illustrates the partition of the set of states into reachable and unreachable states. States $S_1$ and $S_2$ can only be reached from each other, but not from any other state. Therefore, they are unreachable. Similarly, $S_3$ is an unreachable state. Functional operation is assumed to start after the circuit enters a reachable state.

We will use the following property. If $S$ is a reachable state, and there exists a primary input sequence $A$ that takes the circuit from state $S$ to another state $P$, then $P$ is also a reachable state. This is illustrated by Fig. 2. Consider an arbitrary state $R$. Since $S$ is a reachable state, there exists a primary input sequence $B$ that takes the circuit from $R$ to $S$. The sequence $B \cdot A$ takes the circuit from $R$ to $S$ and then to $P$. Therefore, $P$ is reachable from $R$. Since $R$ is an arbitrary state, $P$ is reachable from every circuit state, and it is therefore a reachable state. This property holds when $B$ takes the circuit into $S$ starting from the all-unspecified state.

In this paper, we will assume that a synchronizing sequence is used to bring the circuit from the all-unspecified state to a synchronization state, which is a reachable state, before functional operation starts. Reachability analysis can start from the all-unspecified state, or from the synchronization state. We will use the first option here. The second option can be used to simplify reachability analysis for large circuits.

## III. Background

The problem of unnecessary yield loss due to testing of transition faults under nonfunctional operation conditions was discussed in [4]. The solution proposed in [4] was to use tests that consist of a scan operation followed by a sequence of several primary input vectors (more than the two vectors necessary for testing transition faults). The primary input vectors are applied in the functional operation mode, as in the case of broadside transition-fault tests [11]. The last two input vectors of the sequence constitute a two-pattern broadside test for a targeted transition fault. This approach was based on the intuition that a longer sequence of primary input vectors applied after a scan operation is more likely to cause the circuit to enter its functional operation mode prior to the application of the last two input vectors of the test. We note that even arbitrarily long sequences of primary input vectors may not guarantee that a circuit will be brought to its functional operation mode. Fig. 1 demonstrates such a case through states $S_1$ and $S_2$. If a test starts by scanning in $S_1$ or $S_2$, the circuit may stay in these states and not enter a reachable state even under an arbitrarily long sequence of primary input vectors. In general, consider the case where the state diagram of the circuit has a cycle that includes unreachable states. Suppose that the cycle is reachable from the scan-in state. An arbitrarily long input sequence may take the circuit into the cycle, and testing using the last two input vectors of the sequence will occur under nonfunctional operation conditions. Moreover, if the state diagram has a path of length $n$ that includes unreachable states, a sequence of length $n$ following scan-in does not guarantee that testing using the last two input vectors of the sequence will occur under functional operation conditions.

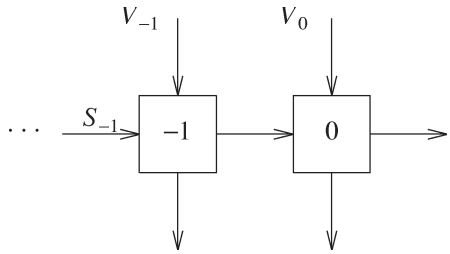



Fig. 3. ILA model.

The procedure from [12] attempts to generate broadside tests that use only functional operation conditions by first finding undetectable faults, identifying states necessary for detecting them, and then avoiding these states during test generation. However, this approach may not avoid generating tests using unreachable states of the circuit even when appropriate reachable states are easy to find. Specifically, it does not guarantee that detectable faults will not be detected using unreachable states even when appropriate reachable states can be found.

To generate functional broadside tests for synchronizable circuits, it is possible to use a sequential test generator as follows. Consider the two time frames labeled 0 and $-1$ of an iterative-logic-array (ILA) model of a sequential circuit given in Fig. 3. To generate a test for a slow-to-rise (slow-to-fall) transition fault on a line $r$ in the circuit, we assume that the next-state variables as well as the primary outputs of time-frame 0 are observable. Under this condition, the sequential test generator needs to generate a test sequence that will detect the stuck-at 0 (stuck-at 1) fault on line $r$ in time-frame 0, while simultaneously setting the value of line $r$ in time-frame $-1$ to 0 (1) (the stuck-at fault is referred to as the stuck-at fault corresponding to the transition fault [13]). Suppose that the test generator produces the test sequence $V_{-k} \cdots V_{-1} V_0$, and that

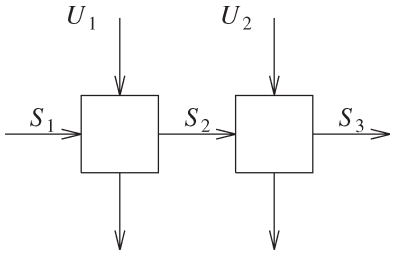



Fig. 4. Two-pattern test.

the sequence takes the circuit from the all-unspecified state at time frame $-k$ to state $S_{-1}$ at time frame $-1$. A two-pattern broadside test for the transition fault is obtained by scanning in $S_{-1}$ and then applying the two-pattern test $\langle V_{-1}V_0\rangle$. If $S_{-1}$ is fully specified, then the existence of a sequence $V_{-k}\cdots V_{-2}$ that takes the circuit from the all-unspecified state to $S_{-1}$ implies that $S_{-1}$ is a reachable state. We thus obtain a functional broadside test for the fault under consideration. If $S_{-1}$ is not fully specified, we simulate the sequence $V_{-k}\cdots V_{-2}$ starting from the synchronization state of the circuit. The fully specified state $\hat{S_{-1}}$ obtained at time frame $-1$ is a reachable state (since it is reachable from the synchronization state), and it can be used as the scan-in state for the functional broadside test.

While sequential test generation provides a complete solution to the problem of generating functional broadside tests, it is a computationally expensive solution. We discuss alternative solutions in Sections IV and V. We use the relationship to sequential test generation in order to identify transition faults that are untestable by functional broadside tests. The number of such faults will allow us to evaluate the effectiveness of the proposed procedures for generating functional broadside tests. We determine the untestable faults by adopting a method proposed earlier for determining sequentially untestable stuck-at faults [8]. The method is based on finding faults that are undetectable in an ILA of limited length $K$, assuming that the present state of the leftmost cell is fully controllable, and the next state of the rightmost cell is fully observable. Such a method was given for transition faults in [9, Procedure 1]. It can be shown that this procedure actually identifies transition faults that cannot be tested using functional broadside tests. We use the results of this procedure in the experiments reported in Section V.

We use the following notation. A test vector for a stuck-at fault is denoted by $U \cdot S$, where $U$ is a primary input vector and $S$ is a scan-in state. To apply the test vector, $S$ is first scanned in. The primary input vector $U$ is then applied in functional mode. The resulting state is then scanned out.

Transition faults require two-pattern tests. We denote a two-pattern test by $\langle U_1 \cdot S_1, U_2 \cdot S_2\rangle$. Under a broadside test, $S_2$ is the next state obtained when $U_1$ is applied to the primary inputs of the circuit and the present state is $S_1$. This is illustrated in Fig. 4. The test is applied as follows. The state $S_1$ is scanned in. The primary input vector $U_1$ is applied next. The next-state $S_2$ obtained under $U_1$ and $S_1$ is latched in the flip-flops. Next, $U_2$ is applied. The final state $S_3$ obtained under $U_2$ and $S_2$ is latched in the flip-flops. Finally, state $S_3$ is scanned out.

## IV. Generating Functional Broadside Tests From Arbitrary Broadside Tests

The solution we explore in this section starts from a broadside transition-fault test set $C$ generated without considering functional operation. The proposed procedure accepts the test set $C$ as input and produces a new test set $CR$ (where the $R$ stands for reachable). Using a simulation-based process, the procedure checks for every test $t \in C$ whether its scan-in state $S$ is a reachable state. During this process, the procedure also collects a set of fully specified reachable states $\Psi$ that are as close to $S$ as possible. If $S$ is reachable then $t$ is included without modification in the new test set $CR$. If $S$ is unreachable, then the states in $\Psi$ are used for defining new tests, which are included in $CR$ instead of $t$. The resulting test set $CR$ contains only tests with reachable states. This solution does not require any modifications to the test-generation procedure, and does not require sequential test generation or fault simulation.

In the following sections, we describe the details of this process. For ease of presentation, we first consider stuck-at faults with test vectors of the form $U \cdot S$. In Section IV-A, we describe a simulation-based procedure for finding reachable states. In Section IV-B, we describe the procedure for modifying a test set $C$ for stuck-at faults into a test set $CR$ with reachable states. Broadside test sets for transition faults are discussed in Section IV-C. Experimental results for transition faults are included in Section IV-D.

### A. Finding Reachable States

In this section, we describe the procedure we use for checking whether a state $S$, which is part of a test vector $U \cdot S$, is reachable in the sequential circuit. The same procedure is used for collecting a set of fully specified states $\Psi$, which are reachable in the sequential circuit and have the smallest possible Hamming distances from $S$. If $S$ is not reachable, the states in $\Psi$ will be used for replacing $S$ to form new test vectors that potentially detect the same faults but contain reachable states. The motivation for requiring that the states in $\Psi$ would be as close to $S$ as possible can be seen as follows. A test for a fault can typically be unspecified without losing the detection of the fault. This implies that some tests at a certain Hamming distance will also detect the fault. The smaller the distance, the more likely it is that the tests will detect the fault. States at the smallest possible distance from $S$, when used to replace $S$ in a test, are thus more likely to help detect the same faults as the test based on $S$.

Different methods of checking state reachability have been described in [14] and [15]. The method described here is based on simple heuristics, and it leads naturally to the generation of the sets $\Psi$ that are needed for defining functional broadside tests.

We demonstrate the procedure by considering International Symposium on Circuits and Systems (ISCAS)-89 benchmark circuit $s27$, which is shown in Fig. 5. A test set $C$ for $s27$ is shown in Table I. The circuit has four primary inputs 1, 2, 3, 4, and three state variables 5, 6, 7. Each test vector in $C$ has the form $U_i \cdot S_i$, where $U_i$ is the four-bit subvector applied to the

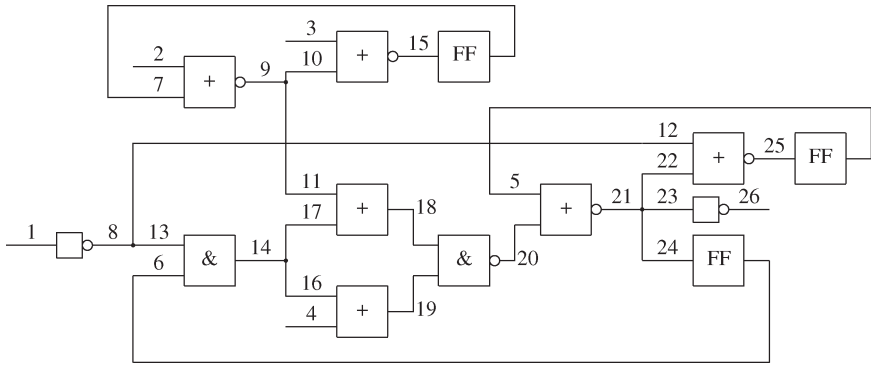



Fig. 5. ISCAS-89 benchmark circuit $s27$.

TABLE I
TEST SET FOR $s27$

| $i$ | $U_i$ | $S_i$ |
|---|---|---|
| 0 | 0000 | 011 |
| 1 | 1001 | 010 |
| 2 | 0100 | 110 |
| 3 | 0111 | 001 |
| 4 | 1101 | 011 |
| 5 | 1010 | 000 |

primary inputs, and $S_i$ is the three-bit subvector applied to the present-state variables.

Considering a test vector $U_i \cdot S_i$, we need to check whether $S_i$ is reachable. We use a simulation-based procedure to compute an input sequence that brings the circuit from the all-unspecified state $xx \cdots x$ to $S_i$, or to a state that is as close to $S_i$ as possible. We initially set $P = xx \cdots x$. Starting from $P$, we construct the input sequence by considering at most $L_{\max}$ consecutive time units, where $L_{\max}$ is a preselected constant. At every time unit $u$, we apply a preselected number $N_{\rm rand}$ of random input vectors $V_0 \cdots V_{N_{\rm rand}-1}$. We then select one of the vectors $V_r$ to be included in the input sequence. The selection is based on an analysis of next-states $Q_0 \cdots Q_{N_{\rm rand}-1}$ obtained under $V_0 \cdots V_{N_{\rm rand}-1}$, respectively. The selected next-state $Q_r$ becomes the present-state $P$ at time unit $u+1$.

To select $V_r$, we compute the following parameters for every next-state $Q_r$. The first parameter is the number of specified values in $Q_r$, denoted by n_spec($Q_r$). Since the test vectors in $C$ are fully specified, our first priority is to bring the circuit to a fully specified state. We achieve this goal by selecting a state that has the highest value of n_spec($Q_r$) [16]. The second parameter is denoted by new_state($Q_r$). We set new_state($Q_r$) $= 1$ if $Q_r$ was not reached before along the input sequence being computed. Otherwise, new_state($Q_r$) $= 0$. We prefer $Q_r$ to be a new state since this maximizes our ability to explore the space of reachable states. The third parameter we compute is the Hamming distance between $S_i$ and $Q_r$, denoted by dist($S_i, Q_r$). The Hamming distance is incremented by one for every bit where $S_i$ is different from $Q_r$. This includes bits where $Q_r$ is unspecified ($S_i$ is completely specified). We prefer to select a state for which dist($S_i, Q_r$) is minimum as a heuristic to guide us towards reaching $S_i$. In addition, if $S_i$ is not reached, it is advantageous to obtain states that are as close to it as possible as discussed earlier.

TABLE II
CONSTRUCTING INPUT SEQUENCE FOR $S_1$: (a) $u = 0$; (b) $u = 1$; (c) $u = 2$

(a)

| $r$ | $V_r$ | $Q_r$ | *n_spec* | *new_state* | *dist* |
|---|---|---|---|---|---|
| 0 | 0100 | 0x1 | 2 | 1 | 2 |
| 1 | 0101 | 0x1 | 2 | 1 | 2 |
| 2 | 1101 | 101 | 3 | 1 | 3 |
| 3 | 1100 | 101 | 3 | 1 | 3 |
| 4 | 0100 | 0x1 | 2 | 1 | 2 |

(b)

| $r$ | $V_r$ | $Q_r$ | *n_spec* | *new_state* | *dist* |
|---|---|---|---|---|---|
| 0 | 0110 | 000 | 3 | 1 | 1 |
| 1 | 0110 | 000 | 3 | 1 | 1 |
| 2 | 0000 | 001 | 3 | 1 | 2 |
| 3 | 0101 | 001 | 3 | 1 | 2 |
| 4 | 1010 | 100 | 3 | 1 | 2 |

(c)

| $r$ | $V_r$ | $Q_r$ | *n_spec* | *new_state* | *dist* |
|---|---|---|---|---|---|
| 0 | 0110 | 000 | 3 | 0 | 1 |
| 1 | 1111 | 100 | 3 | 1 | 2 |
| 2 | 1011 | 010 | 3 | 1 | 0 |
| 3 | 0001 | 010 | 3 | 1 | 0 |
| 4 | 0011 | 010 | 3 | 1 | 0 |

Considering state $S_1 = 010$ of $s27$ in Table I, and using $L_{\max} = 20$ and $N_{\rm rand} = 5$, we show in Table II the results of the procedure described above. We select $V_2 = 1101$ at time unit 0 based on the parameters obtained for $Q_2 = 101$ in Table II(a); we select $V_0 = 0110$ at time unit 1 based on the parameters obtained for $Q_0 = 000$ in Table II(b); and we select $V_2 = 1011$ at time unit 2 based on the parameters obtained for $Q_2 = 010$ in Table II(c). Since $Q_2 = S_1$, the procedure terminates with the indication that $S_1$ is reachable. The three fully specified present states reached during this process are included in the set $\Psi$. We have $\Psi = \{101, 000, 010\}$.

Applying the same process to the other test vectors in Table I, we find that $S_0$, $S_3$, $S_4$, and $S_5$ are reachable. The only state that cannot be reached is $S_2$. For $S_2$, we obtain an input sequence of length $L_{\max} = 20$ that visits the states shown in Table III (we only show in Table III fully specified new states when they are reached). Under column $u$, we show the time unit. Under column $V_u$ we show the input vector at time unit $u$. Under column $P_u$ we show the present state at time unit $u$. We then show the three parameters related to $P_u$. The states reached in Table III are included in the set $\Psi$. They will be used later to define new test vectors that will replace $U_2 \cdot S_2$.

TABLE III
INPUT SEQUENCE FOR $S_2 = 110$

| $u$ | $V_u$ | $P_u$ | *n_spec* | *new_state* | *dist* |
|---|---|---|---|---|---|
| 1 | 1101 | 100 | 3 | 1 | 1 |
| 2 | 0110 | 101 | 3 | 1 | 2 |
| 3 | 0011 | 000 | 3 | 1 | 2 |
| 4 | 0101 | 010 | 3 | 1 | 1 |
| 5 | 0010 | 011 | 3 | 1 | 2 |
| 10 | 1111 | 001 | 3 | 1 | 3 |

The procedure described above is given next as Procedure 1.

*Procedure 1—Finding reachable states closest to a state $S$:*

1) Let $P$ be the all-unspecified state. Set $\Psi = \emptyset$. Set $u = 0$.
2) Set $\Omega = \emptyset$. For $r = 0, 1, \ldots, N_{\rm rand} - 1$ :
   a) Let $V_r$ be a random input vector. Add $V_r$ to $\Omega$.
   b) Simulate the combinational logic of the circuit under present-state $P$ and primary input vector $V_r$. Let $Q_r$ be the next state.
   c) Find the number of specified bits n_spec$(Q_r)$ in $Q_r$.
   d) If $Q_r \in \Psi$, set new_state$(Q_r) = 0$; otherwise, set new_state$(Q_r) = 1$.
   e) Find the Hamming distance dist$(S, Q_r)$ between $S$ and $Q_r$.
3) Select a vector $V_r \in \Omega$ as follows:
   a) Let n_spec_max $= \max\{$n_spec$(Q_r) \ : \ V_r \in \Omega\}$. Remove from $\Omega$ every vector $V_r$, such that n_spec$(Q_r) <$ n_spec_max.
   b) Let new_state_max $= \max\{$new_state$(Q_r) \ : \ V_r \in \Omega\}$. Remove from $\Omega$ every vector $V_r$, such that new_state$(Q_r) <$ new_state_max.
   c) Let min_dist $= \min\{$dist$(S, Q_r) \ : \ V_r \in \Omega\}$. Remove from $\Omega$ every vector $V_r$, such that dist$(S, Q_r) >$ min_dist.
   d) Select the first vector in $\Omega$. Let this vector be $V_r$.
4) If $Q_r$ is fully specified, add $Q_r$ to $\Psi$. If $Q_r = S$, stop.
5) Set $P = Q_r$. Set $u = u + 1$. If $u < L_{\rm max}$ go to Step 2).

The complexity of Procedure 1 is determined by the need to simulate up to $N_{\rm rand} L_{\rm max}$ input vectors. Parallel pattern simulation can be used to speed up Procedure 1. It is important to note that only logic simulation of the fault-free circuit is used in Procedure 1.

### B. Obtaining Test Vectors with Reachable States

In this section, we describe the procedure that accepts a test set $C$ for stuck-at faults, where some of the states may be unreachable, and produces a test set $CR$ for stuck-at faults, where all the states are reachable. As before, a test vector is represented as $U \cdot S$.

The test set $CR$ is obtained as follows. For every test vector $U_i \cdot S_i \in C$, if $S_i$ is reachable, then $U_i \cdot S_i$ is copied to $CR$. If $S_i$ is not reachable, we add to $CR$ a set of test vectors $CR(U_i \cdot S_i)$, instead of $U_i \cdot S_i$. Every test vector in $CR(U_i \cdot S_i)$ has the form $U_i \cdot P_u$, i.e., we maintain the primary input subvector $U_i$ of $U_i \cdot S_i$ in every test vector included in $CR(U_i \cdot S_i)$. For $P_u$, we use the states included in $\Psi$ during the application of Procedure 1 to $S_i$. We consider the states $P_u$ in $\Psi$ by order of increasing distance dist$(S_i, P_u)$. In this way, we ensure that earlier test vectors are more likely to detect the faults detected by $U_i \cdot S_i$, while later test vectors may be dropped. We add several tests to replace $U_i \cdot S_i$ in $CR$ as this increases the likelihood that the faults detected by $U_i \cdot S_i$ will be detected by $CR$.

TABLE IV
TEST SET $CR$ FOR $s27$

| $i$ | $U_i$ | $S_i$ |
|---|---|---|
| 0 | 0000 | 011 |
| 1 | 1001 | 010 |
| 2 | 0111 | 001 |
| 3 | 1101 | 011 |
| 4 | 1010 | 000 |
| 5 | 0100 | 100 |
| 6 | 0100 | 010 |
| 7 | 0100 | 101 |
| 8 | 0100 | 000 |
| 9 | 0100 | 011 |
| 10 | 0100 | 001 |

The set $CR(U_i \cdot S_i)$ may contain test vectors that do not detect any new faults. As a result, the number of test vectors in $CR$ may be larger than necessary. We remove unnecessary test vectors from $CR$ by performing fault simulation with fault dropping. A test vector that does not detect any fault during this process is removed from $CR$. The upper bound on the number of test vectors included in $CR$ before dropping any test vectors is $|C|L_{\rm max}$ (for every original test vector in $C$, we may include in $CR$ up to $L_{\rm max}$ test vectors).

For the test set of $s27$ shown in Table I, we found earlier that the states $S_0$, $S_1$, $S_3$, $S_4$, and $S_5$ are reachable. The corresponding test vectors $U_i \cdot S_i$ for $i = 0, 1, 3, 4, 5$ are copied from $C$ to $CR$. For $U_2 \cdot S_2$, we define a set of test vectors $CR(U_2 \cdot S_2)$ based on the states shown in Table III. The resulting test set is shown in Table IV. Performing fault simulation with fault dropping for the test set $CR$ shown in Table IV, we find that test vectors $U_6 \cdot S_6, \ldots, U_{10} \cdot S_{10}$ do not detect any new faults. We end up with a test set $CR$ that includes the first six test vectors in Table IV.

The procedure for constructing $CR$ from $C$ is given next as Procedure 2.

*Procedure 2—Constructing the test set $CR$:*

1) Let $C = \{U_i \cdot S_i \ : \ 0 \leq i < n\}$ be the given test set. Set $CR = \emptyset$.
2) For $i = 0, 1, \ldots, n - 1$:
   a) Call Procedure 1 with $S_i$.
   b) If $S_i \in \Psi$, add $U_i \cdot S_i$ to $CR$. Else:
      i) Set min_dist $= \min\{$dist$(S_i, P_u) : P_u \in \Psi\}$.
      ii) For dist $=$ min_dist, min_dist $+ 1, \ldots, N_{\rm sv}$ (where $N_{\rm sv}$ is the number of state variables):
         A) For every $P_u \in \Psi$, if dist$(S_i, P_u) =$ dist, add $U_i \cdot P_u$ to $CR$.
3) Let $CR = \{U_i \cdot S_i : 0 \leq i < m\}$. Let $F$ be the set of faults detected by $C$. For $i = 0, 1, \ldots, m - 1$:
   a) Simulate $F$ under $U_i \cdot S_i$ with fault dropping.
   b) If $U_i \cdot S_i$ does not detect any fault in $F$, remove $U_i \cdot S_i$ from $CR$.

The complexity of Procedure 2 is determined by the fact that Procedure 1 is called $|C|$ times. Procedure 2 also performs fault simulation with fault dropping of $CR$.

TABLE V
INPUT SEQUENCE FOR $S_2 = 001$

| $u$ | $V_u$ | $P_u$ | *n_spec* | *new_state* | *dist* |
|---|---|---|---|---|---|
| 0 | 1101 | xxx | 0 | 0 | 3 |
| 1 | 0001 | 101 | 3 | 1 | 1 |
| 2 | 1011 | 001 | 3 | 1 | 0 |
| 3 | 0000 | 100 | 3 | 1 | 2 |
| 4 | 1011 | 000 | 3 | 1 | 1 |
| 5 | 1000 | 010 | 3 | 1 | 2 |
| 6 | 0011 | 100 | 3 | 0 | 2 |
| 7 | 1101 | 000 | 3 | 0 | 1 |
| 8 | 0100 | 101 | 3 | 0 | 1 |
| 9 | 1100 | 001 | 3 | 0 | 0 |
| 10 | | 101 | 3 | 0 | 1 |

### C. Application to Broadside Test Sets for Transition Faults

In this section, we describe the modifications required in Procedure 2 to accommodate broadside test sets for transition faults.

Given a broadside test set $C$ such that the state $S_1$ of a test $\langle U_1 \cdot S_1, U_2 \cdot S_2 \rangle \in C$ may be unreachable, we obtain a test set $CR$ where all the states are reachable as follows.

We consider every test $\langle U_1 \cdot S_1, U_2 \cdot S_2 \rangle \in C$. We first check whether $S_1$ is a reachable state by calling Procedure 1. If $S_1$ is a reachable state, $S_2$ is guaranteed to be a reachable state as well (every state that can be reached from a reachable state is also a reachable state). In this case, we copy the test into $CR$ and we do not consider it further. Otherwise, we consider the test in two phases. In the first phase, we focus on $S_1$, and we define new tests that have reachable states as part of their first pattern instead of $S_1$. In the second phase, we focus on $S_2$ (even if $S_2$ is already reachable). The need for the second phase will be clarified later.

In the first phase, where $S_1$ is considered, we follow the same procedure as for stuck-at faults. Procedure 1 finds a set of reachable states $\Psi$ that are as close as possible to $S_1$. We consider the states in $\Psi$ by order of increasing distance from $S_1$. When $P_u \in \Psi$ is considered, we add $\langle U_1 \cdot P_u, U_2 \cdot Q_2 \rangle$ to $CR$, where $Q_2$ is the state obtained by applying $U_1$ to the primary inputs when the circuit is in state $P_u$. We note that $Q_2$ may be different from $S_2$. Moreover, even if $P_u$ is close to $S_1$, there is no guarantee that $Q_2$ will be close to $S_2$. This is the motivation for focusing on $S_2$ in the second phase, which is described next.

In the second phase, we apply Procedure 1 to check whether $S_2$ is a reachable state. We extend Procedure 1 to store, for every state $P_u \in \Psi$, the state $P_{u-1}$ from which $P_u$ was reached, and the primary input vector $V_{u-1}$ under which $P_u$ was reached. We will later use $V_{u-1}$ and $P_{u-1}$ to define the first pattern of a test. Due to this use of $P_{u-1}$, a state $P_u$ is entered into $\Psi$ only if $P_{u-1}$ is fully specified. We terminate Procedure 1 with an input sequence of length $L_{\max}$ even if $S_2$ is reached during the construction of the sequence. This will provide more candidate tests to replace the test under consideration as explained later.

We show an example based on $s27$ with $L_{\max} = 10$ in Table V. The test under consideration is $\langle 0100 \cdot 110, 0000 \cdot 001 \rangle$, with $S_2 = 001$. The sequence shown in Table V is the one traced by Procedure 1. The set $\Psi$ in this case consists of the following entries. There is no entry corresponding to $P_0$ and no entry corresponding to $P_1$ since $P_0$ is not fully specified. There is an entry for every one of the other time units, consisting of $P_2 = 001$ with $V_1 = 0001$ and $P_1 = 101$; $P_3 = 100$ with $V_2 = 1011$ and $P_2 = 001$; and so on, up to $P_{10} = 101$ with $V_9 = 1100$ and $P_9 = 001$.

TABLE VI
BROADSIDE TESTS WITH REACHABLE STATES FOR $s27$

| (a) Distance 0 | (b) Distance 1 | (c) Distance 2 |
|---|---|---|
| <0001·101,0000·001> | <0000·100,0000·000> | <1011·001,0000·100> |
| <0100·101,0000·001> | <0011·100,0000·000> | <1011·000,0000·010> |
| | <1101·000,0000·101> | <1000·010,0000·100> |
| | <1100·001,0000·101> | |

We use $\Psi$ to define new tests with reachable states as follows. For $P_u \in \Psi$, which is associated with $P_{u-1}$ and $V_{u-1}$, we define the test $\langle V_{u-1} \cdot P_{u-1}, U_2 \cdot P_u \rangle$. In this test, $P_u$ is as close to $S_2$ as possible, and $V_{u-1} \cdot P_{u-1}$ guarantees that $P_u$ will be obtained as part of the second pattern of the test. In addition, $P_{u-1}$ is a reachable state. As in the case of stuck-at faults, we consider the states $P_u$ in $\Psi$ by order of increasing distance from $S_2$ when adding new tests to $CR$.

In the example of $s27$, based on the test $\langle 0100 \cdot 110, 0000 \cdot 001 \rangle$, we obtain the following tests. Using the states $P_u$ for $u = 2$ and 9, at distance 0 from $S_2$, we obtain the tests shown in Table VI(a). Using the states $P_u$ for $u = 4, 7, 8,$ and 10, at distance 1 from $S_2$, we obtain the tests shown in Table VI(b). Using the states $P_u$ for $u = 3, 5,$ and 6, at distance 2 from $S_2$, we obtain the tests shown in Table VI(c).

It is important to note that we obtain several tests with the same second pattern. However, the tests differ in their first pattern and therefore have the potential of detecting different faults. This is the motivation for including states in $\Psi$ even if they are not new states.

As in the case of stuck-at faults, $CR$ may contain tests that are not necessary for detecting any transition faults in the circuit. We remove unnecessary tests by performing fault simulation with fault dropping of $CR$.

### D. Experimental Results for Broadside Test Sets

We applied the procedure described in Section IV-C to broadside test sets for ISCAS-89 and International Test Conference (ITC)-99 benchmark circuits. We only consider circuits for which synchronizing sequences can be computed starting from the all-unspecified initial state using three-value logic. We use $N_{\text{rand}} = 100$ and $L_{\max} = 1000$.

We used two types of broadside test sets for transition faults. Each one of these test sets will be used as the test set $C$ to which the proposed procedure will be applied. Both test sets are fully specified. The use of incompletely specified test sets can simplify the identification of reachable states for each test, since the reachable states would only need to match the specified values of the test.

For several of the circuits, a deterministic broadside test set is available, which is produced by the deterministic-test-generation procedure from [17]. We compacted the test set by applying forward-looking reverse-order fault simulation [18]. We denote the resulting test set by $C_d$.

To obtain test sets for all the circuits considered, we use the following process that yields a test set denoted by $C_r$. We start from a stuck-at test set $C_0$ where the tests have the form $U_i \cdot S_i$.

TABLE VII
BROADSIDE-TEST SETS FOR ISCAS-89 BENCHMARKS
(NUMBERS OF FAULTS)

| | trans | detected | | | |
|---|---|---|---|---|---|
| | | orig(C) | | mod(CR) | |
| circuit | flts | det | f.c | det | e.f.c |
| s208 | 416 | 321 | 77.16 | 227 | 70.72 |
| s298.d | 596 | 487 | 81.71 | 433 | 88.91 |
| s298 | 596 | 487 | 81.71 | 433 | 88.91 |
| s344.d | 688 | 650 | 94.48 | 624 | 96.00 |
| s344 | 688 | 650 | 94.48 | 624 | 96.00 |
| s382.d | 764 | 599 | 78.40 | 587 | 98.00 |
| s382 | 764 | 599 | 78.40 | 587 | 98.00 |
| s400 | 800 | 617 | 77.12 | 603 | 97.73 |
| s420 | 840 | 618 | 73.57 | 241 | 39.00 |
| s526.d | 1052 | 680 | 64.64 | 658 | 96.76 |
| s526 | 1052 | 680 | 64.64 | 658 | 96.76 |
| s641.d | 1280 | 1230 | 96.09 | 1037 | 84.31 |
| s641 | 1280 | 1225 | 95.70 | 1037 | 84.65 |
| s1196.d | 2392 | 2389 | 99.87 | 2390 | 100.04 |
| s1196 | 2392 | 2389 | 99.87 | 2390 | 100.04 |
| s1423.d | 2846 | 2520 | 88.55 | 2459 | 97.58 |
| s1423 | 2846 | 2501 | 87.88 | 2456 | 98.20 |
| s5378.d | 10590 | 9749 | 92.06 | 7461 | 76.53 |
| s5378 | 10590 | 9595 | 90.60 | 7238 | 75.44 |
| s35932.d | 71864 | 62651 | 87.18 | 62673 | 100.04 |
| s35932 | 71864 | 62673 | 87.21 | 62673 | 100.00 |

TABLE VIII
BROADSIDE-TEST SETS FOR ISCAS-89 BENCHMARKS
(TEST-SET SIZES)

| | orig(C) | | | mod(CR) | | |
|---|---|---|---|---|---|---|
| circuit | tst | rch1 | rch2 | all | eff | flr |
| s208 | 33 | 0 | 24 | 19514 | 36 | 24 |
| s298.d | 36 | 3 | 10 | 4525 | 62 | 34 |
| s298 | 36 | 0 | 15 | 4521 | 64 | 29 |
| s344.d | 48 | 0 | 8 | 48571 | 104 | 48 |
| s344 | 44 | 1 | 5 | 37919 | 77 | 41 |
| s382.d | 40 | 0 | 14 | 41248 | 76 | 42 |
| s382 | 45 | 2 | 14 | 42032 | 86 | 44 |
| s400 | 46 | 1 | 10 | 41945 | 89 | 46 |
| s420 | 66 | 0 | 20 | 65641 | 44 | 23 |
| s526.d | 68 | 0 | 11 | 67640 | 120 | 79 |
| s526 | 70 | 1 | 15 | 55366 | 137 | 72 |
| s641.d | 103 | 0 | 51 | 121754 | 200 | 77 |
| s641 | 79 | 0 | 26 | 100749 | 167 | 66 |
| s1196.d | 231 | 2 | 227 | 457747 | 528 | 222 |
| s1196 | 224 | 0 | 211 | 447441 | 567 | 216 |
| s1423.d | 129 | 0 | 5 | 257355 | 393 | 143 |
| s1423 | 134 | 0 | 0 | 267328 | 364 | 136 |
| s5378.d | 396 | 0 | 0 | 769367 | 801 | 204 |
| s5378 | 365 | 0 | 0 | 708416 | 671 | 193 |
| s35932.d | 112 | 0 | 14 | 22288 | 493 | 168 |
| s35932 | 96 | 0 | 4 | 19104 | 500 | 182 |

We include in $C_r$ a test $\langle U_i \cdot S_i, U_j \cdot Q_j \rangle$ for every $U_i \cdot S_i$, $U_j \cdot S_j \in C_0$. We replace $S_j$ by $Q_j$, which is the next state obtained under $U_i$ and $S_i$. We add to $C_r$ a constant number $K$ of random two-pattern tests $\langle V_i \cdot Q_i, V_j \cdot Q_j \rangle$, where $V_i$, $V_j$ and $Q_i$ are random vectors, and $Q_j$ is the next state obtained under $V_i$ and $Q_i$. We simulate $C_r$ with fault dropping to remove unnecessary tests. We then apply forward-looking reverse-order fault simulation to drop additional tests from $C_r$.

The results of the proposed procedure when applied to $C_d$ and $C_r$ are reported in Tables VII–X. The test set $C_r$ is generated using $K = 100\,000$. The results related to numbers of (detected) faults are shown in Tables VII and IX. The results related to test-set sizes are shown in Tables VIII and X. When $C_d$ is used, we append $d$ to the circuit name.

TABLE IX
BROADSIDE-TEST SETS FOR ITC-99 BENCHMARKS
(NUMBERS OF FAULTS)

| | trans | detected | | | |
|---|---|---|---|---|---|
| | | orig(C) | | mod(CR) | |
| circuit | flts | det | f.c | det | e.f.c |
| b03 | 768 | 722 | 94.01 | 449 | 62.19 |
| b04 | 2284 | 2056 | 90.02 | 1922 | 93.48 |
| b05 | 2976 | 2359 | 79.27 | 1224 | 51.89 |
| b06 | 332 | 281 | 84.64 | 275 | 97.86 |
| b07 | 1936 | 1488 | 76.86 | 1066 | 71.64 |
| b08 | 844 | 701 | 83.06 | 691 | 98.57 |
| b09 | 678 | 573 | 84.51 | 518 | 90.40 |
| b10 | 870 | 740 | 85.06 | 710 | 95.95 |
| b11 | 1830 | 1571 | 85.85 | 1528 | 97.26 |

TABLE X
BROADSIDE-TEST SETS FOR ITC-99 BENCHMARKS
(TEST-SET SIZES)

| | orig(C) | | | mod(CR) | | |
|---|---|---|---|---|---|---|
| circuit | tst | rch1 | rch2 | all | eff | flr |
| b03 | 57 | 0 | 7 | 27728 | 85 | 42 |
| b04 | 97 | 0 | 8 | 193569 | 360 | 104 |
| b05 | 134 | 0 | 11 | 1003 | 83 | 63 |
| b06 | 26 | 0 | 22 | 252 | 33 | 23 |
| b07 | 87 | 0 | 7 | 855 | 70 | 44 |
| b08 | 68 | 0 | 5 | 44760 | 135 | 61 |
| b09 | 37 | 0 | 7 | 3079 | 66 | 38 |
| b10 | 69 | 0 | 8 | 45794 | 113 | 62 |
| b11 | 112 | 0 | 7 | 195483 | 263 | 108 |

In Tables VII and IX, after the circuit name, we show the number of transition faults. We then show the number of transition faults detected by the test set $C$, and the fault coverage achieved by $C$. Next, we show the number of transition faults detected by the modified test set $CR$, and the effective fault coverage achieved by $CR$. The effective fault coverage is the number of faults detected by $CR$ as a percentage of the number of faults detected by $C$.

In Tables VIII and X, under column orig$(C)$ we show the number of tests in $C$, the number of tests in $C$ that have a reachable state $S_1$ as part of their first pattern, and the number of tests in $C$ that have an unreachable state $S_1$ but a reachable state $S_2$ as part of their second pattern (when $S_1$ is reachable, $S_2$ is guaranteed to be reachable as well, and we do not count these cases separately). We then show the size of $CR$ before it is reduced, the size of $CR$ after it is reduced by fault simulation with fault dropping, and the size of $CR$ after it is reduced by forward-looking reverse-order fault simulation [18].

It should be noted that the test sets we use do not guarantee the detection of all the faults detectable by broadside tests. In addition, Procedure 1 is not guaranteed to identify that a given state is reachable, or produce the closest reachable states. As a result, there may be some variations in the numbers of detected faults depending on the test set. For example, for $s1196$, $C_d$ and $C_r$ detect 2389 faults while $CR$ based on $C_d$, and $CR$ based on $C_r$, detect 2390 faults. The following points can be seen from Tables VII–X.

1) The test set $C_r$ achieves a fault coverage that is typical of broadside tests. This can be seen by comparing the fault coverage to the fault coverage achieved by the deterministic test set $C_d$ when it is available. It is a result of using stuck-at-based tests in addition to random tests.

2) The first pattern of a two-pattern test in $C_r$ (or $C_d$ when it is available) rarely includes a reachable state $S_1$. The second pattern includes a reachable state $S_2$ more often. This is related to the fact that $S_2$ is obtained from $S_1$ through the functional operation of the circuit.
3) The number of tests in $CR$ after test compaction is similar to the number of tests in $C$ (which can be $C_d$ or $C_r$).
4) The test set $CR$ detects large percentages of the faults detected by $C$ using functional broadside tests. The test set $CR$ detects fewer transition faults than $C$ since some faults are undetectable using functional broadside tests. We provide information about the numbers of such faults in Section V-C.
5) The number of faults detected by $CR$ is essentially independent of whether $C_d$ or $C_r$ is used as the test set $C$.

## V. Generating Functional Broadside Tests From Test Sequences for Nonscan Circuits

Instead of computing sets of reachable states as in Section IV, the procedure proposed in this section is based on the following observations. An input sequence $T$ that does not use the scan chain (i.e., an input sequence $T$ applicable to the circuit during functional operation starting from the all-unspecified state) can provide effective functional broadside tests assuming that $T$ synchronizes the circuit and detects stuck-at faults. The relationship between transition-fault tests and stuck-at fault tests [13] leads to the effectiveness of tests extracted from $T$ in detecting transition faults. In addition, the fully specified states obtained after $T$ synchronizes the circuit are by definition reachable states. We demonstrate the extraction of a functional broadside test set from an input sequence $T$ in Section V-A. We extract additional broadside tests from a given input sequence $T$ by deriving additional input sequences from $T$, which detect the same or additional stuck-at faults. New input sequences are derived by static test compaction of $T$ and by modifying selected bits of $T$. These procedures are described in Section V-B. Experimental results are presented in Section V-C. We stress that, similar to the tests generated in Section IV, the functional broadside tests extracted in this section are also two-pattern tests applied using scan.

### A. Functional Broadside Tests Based on Input Sequences

In this section, we demonstrate that an input sequence $T$, which does not use the scan chain, provides functional broadside tests for transition faults. In our discussion, we assume that $T$ is a test sequence that detects most (or all) of the detectable stuck-at faults in the circuit.

A test sequence generated by deterministic test generation for ISCAS-89 benchmark circuit $s27$ is shown in Table XI(a). For every time unit $u$, we show the input vector $V_u$ applied at time-unit $u$, and the circuit-state $S_u$ at time-unit $u$. The circuit is synchronized at time-unit 2 into state $S_2 = 010$. The synchronization state is a reachable state, and starting at time-unit 2, the circuit operates in its functional mode. Consequently, we can use every two consecutive time units starting at time-unit 2 to define a functional broadside test for the circuit.

TABLE XI
Input Sequence for $s27$: (a) Initial; (b) After Omission; (c) Modified

| (a) | | | (b) | | | (c) | |
|---|---|---|---|---|---|---|---|
| $u$ | $V_u$ | $S_u$ | $u$ | $V_u$ | $S_u$ | $u$ | $V_u$ |
| 0 | 0111 | xxx | 0 | 0111 | xxx | 0 | 1000 |
| 1 | 1001 | 0x0 | 1 | 1001 | 0x0 | 1 | 0110 |
| 2 | 0111 | 010 | 2 | 1001 | 010 | 2 | 0000 |
| 3 | 1001 | 010 | 3 | 0100 | 010 | 3 | 0011 |
| 4 | 0100 | 010 | 4 | 1011 | 011 | 4 | 0101 |
| 5 | 1011 | 011 | 5 | 0000 | 100 | 5 | 1100 |
| 6 | 1001 | 100 | 6 | 1011 | 000 | 6 | 0100 |
| 7 | 0000 | 100 | | | | 7 | 1101 |
| 8 | 0000 | 000 | | | | 8 | 1111 |
| 9 | 1011 | 000 | | | | 9 | 0100 |

At time-unit 2, $V_2 = 0111$ and $S_2 = 010$ result in the next-state $S_3 = 010$. We define the functional broadside test $\langle V_2 \cdot S_2, V_3 \cdot S_3 \rangle = \langle 0111 \cdot 010, 1001 \cdot 010 \rangle$. If $V_3 \cdot S_3$ detects a stuck-at fault, the broadside test may detect the corresponding transition fault.

At time-unit 3, $V_3 = 1001$ and $S_3 = 010$. At time-unit 4, $V_4 = 0100$ and $S_4 = 010$. We define the functional broadside test $\langle V_3 \cdot S_3, V_4 \cdot S_4 \rangle = \langle 1001 \cdot 010, 0100 \cdot 010 \rangle$.

In general, given a test sequence $T = V_0 V_1 \cdots V_{L-1}$ that takes the circuit from the all-unspecified state $S_0$ through the sequence of states $S_0 S_1 \cdots S_{L-1}$, we define a functional broadside transition-fault test-set FBT as follows. Let $u_{\rm sync}$ be the first time unit where $S_u$ is fully specified. Let $A_{\rm tr}$ be the set of transition faults. Initially, FBT $= \emptyset$ and $U_{\rm tr} = A_{\rm tr}$. For $u = u_{\rm sync}, u_{\rm sync} + 1, \ldots, L - 2$, we extract the test $t = \langle V_u \cdot S_u, V_{u+1} \cdot S_{u+1} \rangle$ from $T$. We simulate $U_{\rm tr}$ under $t$ with fault dropping. If $t$ detects any fault in $U_{\rm tr}$, we add it to FBT. In the case of $s27$, FBT detects 43 of 52 transition faults.

### B. Extracting Additional Tests

Given an input sequence $T$, a functional broadside transition-fault test set FBT can be extracted as described in Section V-A. In this section, we describe procedures for deriving additional input sequences from which additional functional broadside tests can be extracted.

Suppose that $T$ detects a set of stuck-at faults $D_{\rm sa}$. Suppose in addition that FBT detects a subset $D_{\rm tr}$ of transition faults, leaving the subset $U_{\rm tr}$ of transition faults undetected. Consider a transition fault $f_{\rm tr} \in U_{\rm tr}$ and the corresponding stuck-at fault $f_{\rm sa}$ (if $f_{\rm tr}$ is the slow-to-rise fault on line $r$, the corresponding stuck-at fault $f_{\rm sa}$ is $r$ stuck-at 0; and if $f_{\rm tr}$ is the slow-to-fall fault on line $r$, the corresponding stuck-at fault $f_{\rm sa}$ is $r$ stuck-at 1 [13]). Suppose that $T$ detects $f_{\rm sa}$ at time unit $u > u_{\rm sync}$, where $u_{\rm sync}$ is the time unit where $T$ synchronizes the circuit. Since $f_{\rm tr} \in U_{\rm tr}$, we know that the test $\langle V_{u-1} \cdot S_{u-1}, V_u \cdot S_u \rangle$ does not detect $f_{\rm tr}$ even though $f_{\rm sa}$ is detected at time unit $u$. Our goal is to derive from $T$ a different test sequence $\hat{T}$ that detects $f_{\rm sa}$, with the expectation that extraction of transition-fault tests from $\hat{T}$ may yield a test that detects $f_{\rm tr}$. We describe two processes for deriving new test sequences from $T$ next.

*1) Vector Omission:* The first process derives a new test sequence $\hat{T}$ for a stuck-at fault $f_{\rm sa}$ using a vector-omission process [19]. Initially, $\hat{T} = T$. We omit test vectors from $\hat{T}$ one at a time. After the vector $V_u$ at time unit $u$ of $\hat{T}$ is omitted,

we check whether $\hat{T}$ still detects $f_{\rm sa}$. If not, we restore $V_u$ into $\hat{T}$. Otherwise, $V_u$ is eliminated from $\hat{T}$. We consider all the time units of $\hat{T}$ until a fraction $p_{\rm omit}$ of the vectors have been omitted. If this does not happen in the first pass over $\hat{T}$, we consider all the time units that remain in $\hat{T}$ a second time. This continues until $p_{\rm omit}$ of the vectors have been omitted, or no vector is omitted in a complete pass over $\hat{T}$.

In a given pass over $\hat{T}$, we consider the time units for omission in the following order. Let $u_{\rm sync}$ be the first time unit where the state of the fault-free circuit is fully specified under $\hat{T}$. Let the length of $\hat{T}$ be $\hat{L}$. We consider the time units according to the ordered list $U = \langle u_{\rm sync}, u_{\rm sync}+1, \ldots, \hat{L}-1\rangle$. When time unit $u$ is considered, with probability $p_{\rm try}$ we attempt to omit $V_u$, for a constant $p_{\rm try}$. This process is motivated by the following observations.

Omitting $V_u$ can affect the circuit state at time units $u+1$, $u+2, \ldots$. Thus, omitting vectors at earlier time units can potentially result in a larger number of new transition-fault tests. This implies that earlier time units should be considered for omission earlier in the process to increase the likelihood that they will be omitted. This is the reason for considering the time units in increasing order. However, randomizing the vector-omission process is important for ensuring that significantly different test sequences will be obtained when different faults are considered. We achieve this by attempting to omit a vector with probability $p_{\rm try}$, and skipping over a vector with probability $1-p_{\rm try}$.

After obtaining $\hat{T}$, we extract new transition-fault tests from $\hat{T}$ and perform the transition-fault simulation of these tests. If $f_{\rm tr}$ is detected by one of these tests, consideration of $f_{\rm sa}$ is terminated. Otherwise, if $f_{\rm tr}$ is not detected, we continue to omit test vectors from $\hat{T}$ based on $f_{\rm sa}$. The omission process is the same as the one described above except that $\hat{T}$ is used as a starting point instead of $T$. If any vectors are omitted from $\hat{T}$, we obtain a new test sequence $\hat{T}$ that can potentially provide new tests. We extract the tests and perform transition-fault simulation. We continue this process until $f_{\rm tr}$ is detected, or until no additional vectors can be omitted from $\hat{T}$.

To demonstrate this process, we consider $s27$ with the input sequence $T$ shown in Table XI(a). The transition-fault test set FBT based on $T$ leaves the fault $f_{\rm tr}^9$ undetected (the fault is the $1 \rightarrow 0$ transition fault on line 5). The corresponding stuck-at fault, $f_{\rm sa}^9$, is detected by $T$ (the corresponding stuck-at fault is line 5 stuck-at 1). Vector omission based on $f_{\rm sa}^9$ proceeds as follows. We use $p_{\rm omit} = 1/3$ for this example. This implies that omission will stop after three vectors are omitted successfully. We use $p_{\rm try} = 1/3$. This implies that, with probability 1/3, an attempt will be made to omit a vector. This is implemented by selecting a random value between 0 and 2 for every time unit $u$, and considering $u$ only if the value is 0. Initially, $U = \langle 2,3,4,5,6,7,8,9\rangle$. In the first pass over $U$, omission is not attempted for time units 2, 3, 4, 5, 6, and 7 due to the random values generated for these time units. Omission is attempted for time unit 8, and $V_8$ is omitted from $\hat{T}$ since $f_{\rm sa}^9$ is still detected after the omission. Omission is not attempted for time unit 9. After the first pass over $U$, one vector has been omitted and $U = \langle 2,3,4,5,6,7,9\rangle$. In the second pass over $U$, omission is attempted for time-unit 2, and $V_2$ is omitted from $\hat{T}$ since $f_{\rm sa}^9$ is still detected after the omission. Omission is not attempted for time-units 3, 4, and 5. Omission is attempted for time-unit 6, and $V_6$ is omitted from $\hat{T}$, since $f_{\rm sa}^9$ is still detected after the omission. At this point, the fraction of omitted vectors reaches $p_{\rm omit}$ and the omission process terminates. The resulting test sequence is shown in Table XI(b).

From the test sequence of Table XI(b), we obtain the functional broadside transition-fault tests $\langle 1001 \cdot 010, 0100 \cdot 010\rangle$, $\langle 0100 \cdot 010, 1011 \cdot 011\rangle$, $\langle 1011 \cdot 011, 0000 \cdot 100\rangle$, and $\langle 0000 \cdot 100, 1011 \cdot 000\rangle$. The fourth test detects $f_{\rm tr}^9$. The last two tests are different from the tests obtained from Table XI(a). In general, new tests are obtained due to two effects. 1) The omission of $V_u$ causes $V_{u-1}$ and $V_{u+1}$ to be applied consecutively under $\hat{T}$. This results in a transition-fault test based on $V_{u-1}$ and $V_{u+1}$, which is not available before omission. 2) The omission of $V_u$ causes the states at time units $u+1, u+2, \ldots$ to change. As a result, a test based on $V_w$ and $V_{w+1}$, where $w > u$, may contain different states from the corresponding test before omission.

The overall extraction process is given as Procedure 3 below. It is important to note that we insist on the detection of $f_{\rm sa}$ by $\hat{T}$ to ensure that a time unit is available for an effective second pattern of a test for $f_{\rm tr}$.

*Procedure 3—Extracting functional broadside transition-fault tests from an input sequence $T$:*

1) Let $A_{\rm tr}$ consist of all the transition faults of the circuit. Set $U_{\rm tr} = A_{\rm tr}$. Set FBT $= \emptyset$. Let $D_{\rm sa}$ consist of all the stuck-at faults detected by $T$. Let $u_{\rm sync}$ be the first time unit where the fault-free circuit state is fully specified under $T$.
2) Extract a set $\hat{\rm FBT}$ of functional broadside transition-fault tests from $T$. For every $t \in \hat{\rm FBT}$:
   a) Simulate $U_{\rm tr}$ under $t$. Let the subset of detected faults be $\hat{U}_{\rm tr}$. If $\hat{U}_{\rm tr} \neq \emptyset$, remove $\hat{U}_{\rm tr}$ from $U_{\rm tr}$ and add $t$ to FBT.
3) For every fault $f_{\rm tr} \in A_{\rm tr}$, if $f_{\rm tr} \in U_{\rm tr}$ and the corresponding stuck-at fault $f_{\rm sa}$ is in $D_{\rm sa}$:
   a) Set $\hat{T} = T$.
   b) Set $U = \langle u_{\rm sync}, u_{\rm sync}+1, \ldots, \hat{L}-1\rangle$, where $\hat{L}$ is the length of $\hat{T}$. Set $n_{\rm omit} = 0$.
   c) For every time-unit $u \in U$, as long as $n_{\rm omit} < p_{\rm omit}\hat{L}$:
      i) With probability $p_{\rm try}$, perform the following steps:
         A) If $f_{\rm sa}$ continues to be detected by $\hat{T}$ after omitting $V_u$, then omit $V_u$ from $\hat{T}$ and increment $n_{\rm omit}$ by one.
         B) Remove $u$ from $U$.
   d) If $n_{\rm omit} < p_{\rm omit}\hat{L}$ and $U \neq \emptyset$, go to Step 3c).
   e) If $n_{\rm omit} > 0$:
      i) Extract functional broadside transition-fault tests from $\hat{T}$ as in Step 2).
      ii) If $f_{\rm tr} \in U_{\rm tr}$, go to Step 3b).

*2) Sequence Modification:* After applying Procedure 3 to $T$, we modify $T$ to obtain a new input sequence $T_{\rm mod}$. During the modification, we ensure that $T_{\rm mod}$ would continue to detect every stuck-at fault in $D_{\rm sa}$ (i.e., every stuck-at fault detected by $T$), whose corresponding transition fault is in $U_{\rm tr}$. The transition faults in $U_{\rm tr}$ are the ones that were not detected by

TABLE XII
FUNCTIONAL BROADSIDE TEST SETS BASED ON DETERMINISTIC TEST SEQUENCES

| | | | extract | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | trans | | | | | n. | tst- |
| circuit | flts | Sec.4 | len | s.a. | i.det | | f.det | | tst | time | able |
| s298 | 596 | 433 | 117 | 526 | 419 | (8) | 433 | (10) | 52 | 1.24 | 433 |
| s344 | 688 | 624 | 57 | 660 | 594 | (10) | 620 | (10) | 58 | 144.67 | 624 |
| s382 | 764 | 587 | 516 | 723 | 569 | (17) | 587 | (19) | 67 | 103.56 | 597 |
| s400 | 800 | 603 | 611 | 746 | 580 | (15) | 603 | (19) | 67 | 51.54 | 615 |
| s526 | 1052 | 658 | 1006 | 898 | 629 | (16) | 660 | (16) | 109 | 49.38 | 677 |
| s641 | 1280 | 1037 | 101 | 1119 | 965 | (9) | 1037 | (9) | 116 | 243.04 | 1037 |
| s820 | 1640 | NA | 491 | 1579 | 1233 | (1) | 1343 | (1) | 224 | 2168.57 | 1343 |
| s1196 | 2392 | 2390 | 238 | 2389 | 2048 | (3) | 2386 | (3) | 389 | 89.64 | 2390 |
| s1423 | 2846 | 2459 | 1024 | 2661 | 2385 | (117) | 2484 | (123) | 271 | 12885.05 | 2520 |
| s1488 | 2976 | NA | 455 | 2932 | 2640 | (2) | 2728 | (2) | 255 | 4.97 | 2729 |
| s5378 | 10590 | 7461 | 646 | 8390 | 7908 | (230) | 8196 | (237) | 398 | 846.95 | 8287 |
| s35932 | 71864 | 62673 | 150 | 64492 | 62279 | (28) | 62673 | (28) | 230 | 1.64 | 62673 |
| b03 | 768 | 449 | 130 | 568 | 435 | (15) | 449 | (15) | 65 | 0.62 | 504 |
| b04 | 2284 | 1925 | 168 | 2022 | 1819 | (90) | 1930 | (93) | 164 | 298.44 | 1934 |
| b06 | 332 | 275 | 35 | 310 | 250 | (4) | 275 | (4) | 31 | 42.43 | 275 |
| b09 | 678 | 518 | 269 | 572 | 483 | (27) | 523 | (36) | 61 | 68.81 | 546 |
| b10 | 870 | 710 | 190 | 808 | 665 | (18) | 710 | (23) | 102 | 23.20 | 725 |
| b11 | 1830 | 1528 | 675 | 1690 | 1433 | (40) | 1528 | (55) | 179 | 76.77 | 1533 |

any test derived from $T$. To allow tests for these faults to be generated, we would like the corresponding stuck-at faults to be detected by $T_{\rm mod}$. We denote the subset of stuck-at faults in $D_{\rm sa}$ that correspond to transition faults in $U_{\rm tr}$ by $F_{\rm sa}$. We obtain $T_{\rm mod}$ from $T$ as follows.

We initially set $T_{\rm mod} = T$. For every time-unit $u$ of $T_{\rm mod}$ and every primary input $i$ separately, we complement the value of primary-input $i$ under $T_{\rm mod}$ at time-unit $u$. Denoting this value by $T_{\rm mod}[u, i]$, we set $T_{\rm mod}[u, i] = \overline{T_{\rm mod}[u, i]}$. We then simulate the faults in $F_{\rm sa}$ under $T_{\rm mod}$. If any fault remains undetected, we complement $T_{\rm mod}[u, i]$ again in order to restore its earlier value. Otherwise, we keep the complemented value of $T_{\rm mod}[u, i]$. We repeat this for every $u$ and $i$. At the end of this process, $T_{\rm mod}$ detects all the faults in $F_{\rm sa}$, and it is different from $T$ in as many bits as possible.

For illustration, we consider $s27$ with the test sequence shown in Table XI(a). After applying Procedure 3, the stuck-at faults corresponding to the undetected transition faults are $f_7$, $f_9$, $f_{10}$, $f_{11}$, $f_{21}$, $f_{31}$, $f_{34}$, and $f_{37}$. We find that it is possible to complement $T[0, 0]$ without losing the detection of any of these faults. In a similar way, we find that it is possible to complement $T[0, 1]$ (with $T[0, 0]$ already complemented), $T[0, 2]$ (with $T[0, 0]$, and $T[0, 1]$ already complemented), $T[0, 3]$, $T[1, 0]$, $T[1, 1]$, $T[1, 2]$, and $T[1, 3]$. When we try to complement $T[2, 0]$, we find that $f_{10}$ will remain undetected, and we therefore complement $T[2, 0]$ again. Continuing in the same manner, we find the modified test sequence shown in Table XI(c).

We first extract new transition-fault tests from $T_{\rm mod}$. We then apply Procedure 3 to $T_{\rm mod}$. If any one of these procedures yields a test for one of the faults in $U_{\rm tr}$, we set $T = T_{\rm mod}$ and repeat the modification and extraction procedures. This continues until no new faults are detected.

### C. Experimental Results

We applied the proposed procedure (Procedure 3 and the sequence-modification procedure of Section V-B) to benchmark circuits. We used $p_{\rm omit} = 1/10$ and $p_{\rm try} = 1/5$ in Procedure 3.

We used three types of test sequences for $T$. For all the circuits considered we used test sequences generated by sequential-test-generation procedures for stuck-at faults and compacted by static test compaction [20]. We denote such a test sequence by $T_{\rm cmp}$. To demonstrate that the proposed procedure is effective even if the input sequence $T$ does not detect all the detectable stuck-at faults, we also used random input sequences. We considered random input sequences of lengths 100, 200, ..., until the stuck-at fault coverage was high enough to allow the proposed procedure to extract transition-fault tests that detect the same number of faults detected based on $T_{\rm cmp}$. We denote a random input sequence by $T_{\rm rnd}$. For some of the circuits, which have large numbers of faults that cannot be detected by a random input sequence of reasonable length, we generated test sequences with limited stuck-at fault coverage. Test generation started with a random input sequence of length 200, denoted by $T_0$. During the test-generation process, after the $i$th stuck-at fault was targeted and detected, we stored the test sequence, denoted by $T_i$. We used $T_i$, for $i = 0, 1, \ldots$, as a test sequence $T$ with a limited stuck-at fault coverage. We report the results for $T_{i_0}$ that allowed us to detect the same number of transition faults as $T_{\rm cmp}$. We report the results obtained using $T_{\rm cmp}$ for all the circuits considered. For some of the circuits, we also report the results for $T_{\rm rnd}$ or $T_{i_0}$.

The results are given in Tables XII–XIV, as follows. In Table XII, we use $T_{\rm cmp}$, in Table XIII, we use $T_{\rm rnd}$, and in Table XIV, we use $T_{i_0}$ as the test sequence $T$. After the circuit name, we show the number of faults (the number of noncollapsed single stuck-at faults, which is also the number of transition faults). We then show the number of transition faults detected using functional broadside test sets obtained by the procedure described in Section IV. Under the column "extract" we show the length of the test sequence $T$, and the number of stuck-at faults it detects. We then show the number of transition faults detected by FBT based on the original

TABLE XIII
FUNCTIONAL BROADSIDE TEST SETS BASED ON RANDOM TEST SEQUENCES

| | | | extract | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | trans | | | | | | tst- |
| circuit | flts | Sec.4 | len | s.a. | i.det | | f.det | | tst | n. time | able |
| s298 | 596 | 433 | 700 | 368 | 297 | (22) | 433 | (120) | 54 | 172.40 | 433 |
| s344 | 688 | 624 | 100 | 593 | 524 | (12) | 620 | (65) | 69 | 270.27 | 624 |
| s641 | 1280 | 1037 | 200 | 984 | 870 | (37) | 1037 | (132) | 143 | 180.66 | 1037 |
| b03 | 768 | 449 | 200 | 346 | 358 | (131) | 449 | (191) | 71 | 119.84 | 504 |
| b04 | 2284 | 1925 | 400 | 1319 | 1490 | (416) | 1931 | (693) | 194 | 210.28 | 1934 |
| b06 | 332 | 275 | 100 | 279 | 254 | (14) | 275 | (25) | 33 | 1.11 | 275 |

TABLE XIV
FUNCTIONAL BROADSIDE TEST SETS BASED ON PARTIAL TEST SEQUENCES

| | | | extract | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | trans | | | | | | tst- |
| circuit | flts | Sec.4 | len | s.a. | i.det | | f.det | | tst | n. time | able |
| s382 | 764 | 587 | 443 | 658 | 527 | (41) | 587 | (66) | 73 | 757.26 | 597 |
| b09 | 678 | 518 | 357 | 407 | 288 | (38) | 524 | (177) | 104 | 31.69 | 546 |
| b10 | 870 | 710 | 296 | 767 | 633 | (29) | 711 | (56) | 111 | 49.40 | 725 |

sequence $T$ (subcolumn $i.det$), and the number of transition faults detected by FBT after it is augmented by the procedures of Section V-B (subcolumn $f.det$). In parentheses, we show the number of transition faults detected by FBT, for which the corresponding stuck-at faults are not detected by $T$. Under column "tst" we show the number of tests in FBT at the end of the proposed procedure. Under column "n.time" we show the run time of the proposed procedure when it reaches its final fault coverage, normalized to the time it takes to simulate the input sequence $T$ under the set of all the single stuck-at faults. In the last column, we give an upper bound on the number of transition faults that are detectable by functional broadside tests. The bound is computed as the number of all the transition faults, minus the number of faults found to be untestable by functional broadside tests using [9, Procedure 1]. Since this procedure may not find all the transition faults that are untestable by functional broadside tests, the number reported is an upper bound on the number of testable faults.

The following points can be seen from Tables XII–XIV.

1) It is not necessary for the test sequence $T$ to detect all the single stuck-at faults in the circuit. Partial coverage of stuck-at faults is sufficient for detecting the same number of transition faults. This is true as long as the test sequence detects a sufficient percentage of the stuck-at faults, or it is long enough to provide a sufficient number of candidate tests to compensate for the lower stuck-at fault coverage.
2) For many of the circuits considered, the proposed procedures found tests for essentially all the transition faults that are testable by functional broadside tests. If fault coverage loss is unacceptable, nonfunctional tests can be added for the remaining faults as discussed in Section VI.
3) The procedure that extracts broadside tests from input sequences detects more faults than the procedure of Section IV if test sequences with sufficient fault coverage are available.

## VI. CONCLUDING REMARKS

We described two procedures for generating functional broadside tests for transition faults. Such tests use only the reachable states of the circuit, and thus, detect faults under functional operation conditions. This alleviates the unnecessary yield loss that may result from detecting faults during nonfunctional operation. The first procedure accepted a test-set $C$ whose tests may include unreachable scan-in states, and produced a test-set $CR$ whose tests include only reachable states. The second procedure was based on the extraction of functional tests from an input sequence $T$ for the nonscan circuit. Experimental results demonstrated that the resulting test sets detect most of the transition faults detectable by functional broadside tests.

When the resulting test sets do not detect all the faults that are detectable using scan, and fault coverage loss is unacceptable, it is possible to add nonfunctional (broadside or other) tests for the remaining faults. Such an extended test set would detect as many faults as possible using functional tests, and the remaining faults using nonfunctional tests. It would thus minimize the unnecessary yield loss due to testing under nonfunctional operation conditions.

## REFERENCES

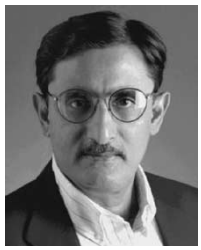
[1] M. J. Y. Williams and J. B. Angell, "Enhancing testability of large scale integrated circuits via test points and additional logic," *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 46–60, Jan. 1973.
[2] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," in *Proc. 14th Design Automation Conf.*, New Orleans, LA, Jun. 1977, pp. 462–468.
[3] J. Saxena, K. M. Butler, V. B. Jayaram, S. Kundu, N. V. Arvind, P. Sreeprakash, and M. Hachinger, "A case study of IR-drop in structured at-speed testing," in *Proc. Int. Test Conf.*, Charlotte, NC, 2003, pp. 1098–1104.
[4] J. Rearick, "Too much delay fault coverage is a bad thing," in *Proc. Int. Test Conf.*, Baltimore, MD, Oct. 2001, pp. 624–633.
[5] K.-T. Cheng and J.-Y. Jou, "Functional test generation for finite state machines," in *Proc. Int. Test Conf.*, Washington DC, Sep. 1990, pp. 162–168.
[6] F. Ferrandi, F. Fummi, L. Gerli, and D. Sciuto, "Symbolic functional

vector generation for VHDL specifications," in *Proc. Design, Automation Test Eur. Conf.*, Munich, Germany, Mar. 1999, pp. 442–446.
[7] R. S. Tupuri and J. A. Abraham, "A novel functional test generation method for processors using commercial ATPG," in *Proc. Int. Test Conf.*, Washington DC, Nov. 1997, pp. 743–752.
[8] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG theorems for identifying untestable faults in synchronous sequential circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 9, pp. 1155–1160, Sep. 1995.
[9] G. Chen, S. M. Reddy, and I. Pomeranz, "Procedures for identifying untestable and redundant transition faults in synchronous sequential circuits," in *Proc. Int. Conf. Computer Design*, San Jose, CA, 2003, pp. 36–41.
[10] J. Savir and S. Patil, "Scan-based transition test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 12, no. 8, pp. 1232–1241, Aug. 1993.
[11] ——, "Broad-side delay test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 13, no. 8, pp. 1057–1064, Aug. 1994.
[12] X. Liu and M. S. Hsiao, "Constrained ATPG for broadside transition testing," in *Proc. Int. Symp. Defect and Fault Tolerance VLSI Systems*, Boston, MA, 2003, pp. 175–182.
[13] J. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar, "Transition fault simulation," *IEEE Des. Test. Comput.*, vol. 4, no. 5, pp. 32–38, Apr. 1987.
[14] H. Cho, G. D. Hachtel, E. Macii, B. Plessier, and F. Somenzi, "Algorithms for approximate FSM traversal based on state space decomposition," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 15, no. 12, pp. 1465–1478, Dec. 1996.
[15] D. E. Long, M. A. Iyer, and M. Abramovici, "Identifying sequentially untestable faults using illegal states," in *Proc. Very Large Scale Integration (VLSI) Test Symp.*, Princeton, NJ, Apr. 1995, pp. 4–11.
[16] V. D. Agrawal, K. T. Cheng, and P. Agrawal, "A directed search method for test generation using concurrent simulator," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 2, pp. 131–138, Feb. 1989.
[17] Y. Shao, I. Pomeranz, and S. M. Reddy, "On generating high quality tests for transition faults," in *Proc. Asian Test Symp.*, Tamuning, Guam, Nov. 2002, pp. 1–8.
[18] I. Pomeranz and S. M. Reddy, "Forward-looking fault simulation for improved static compaction," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1262–1265, Oct. 2001.
[19] ——, "On static compaction of test sequences for synchronous sequential circuits," in *Proc. 33rd Design Automation Conf.*, Las Vegas, NV, Jun. 1996, pp. 215–220.
[20] ——, "Vector restoration based static compaction of test sequences for synchronous sequential circuits," in *Proc. Int. Conf. Computer Design*, Austin, TX, Oct. 1997, pp. 360–365.

**Irith Pomeranz** (M'89–SM'96–F'99) received the B.Sc. degree (*summa cum laude*) in computer engineering and the D.Sc. degree in electrical engineering from the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel, in 1985 and 1989, respectively.

From 1989 to 1990, she was a Lecturer at the Department of Computer Science, Technion. From 1990 to 2000, she was a Faculty Member in the Department of Electrical and Computer Engineering, University of Iowa, Iowa City. In 2000, she joined the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, where she is currently a Professor. Her research interests include testing of very large scale integration (VLSI) circuits, design for testability, synthesis, and design verification.

Dr. Pomeranz was a recipient of the National Science Foundation (NSF) Young Investigator Award in 1993, and of the University of Iowa Faculty Scholar Award in 1997. She served as the Associate Editor of the *ACM Transactions on Design Automation* and the IEEE TRANSACTIONS ON COMPUTERS. She served as the Guest Editor of the IEEE TRANSACTIONS ON COMPUTERS January 1998 Special Issue on Dependability of Computing Systems, and as the Program Co-Chair of the 1999 Fault-Tolerant Computing Symposium. She served as the Program Chair of the 2004 and the 2005 VLSI Test Symposium. She is serving as the General Chair of the 2006 VLSI Test Symposium.



**Sudhakar M. Reddy** (S'68–M'68–SM'84–F'87–LF'04) received the B.E. degree in electronics and communication engineering from Osmania University, Hyderabad, India, the M.E. degree in electronics and communication engineering from the Indian Institute of Science, India, and the Ph.D. degree in electrical engineering from the University of Iowa, Iowa City.

Since 1968, he has been a member of the faculty at the Department of Electrical and Computer Engineering, University of Iowa, where he is currently a Professor. In 1990, he was made a University of Iowa Foundation Distinguished Professor. Since 1972, he has been active in the areas of testable designs and test generation for logic circuits.

Dr. Reddy is a member of Tau Beta Pi, Eta Kappa Nu, and Sigma Xi. He has been an Associate Editor and twice a Guest Editor of the IEEE TRANSACTIONS ON COMPUTERS. He is the Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.