Covering Test Holes of Functional Broadside Tests

IRITH POMERANZ, Purdue University

Functional broadside tests were developed to avoid overtesting of delay faults. The tests achieve this goal by creating functional operation conditions during their functional capture cycles. To increase the achievable fault coverage, close-to-functional scan-based tests are allowed to deviate from functional operation conditions. This paper suggests that a more comprehensive functional broadside test set can be obtained by replacing target faults that cannot be detected with faults that have similar (but not identical) detection conditions. A more comprehensive functional broadside test set has the advantage that it still maintains functional operation conditions. It covers the test holes created when target faults cannot be detected by detecting similar faults. The paper considers the case where the target faults are transition faults. When a standard transition fault, with an extra delay of a single clock cycle, cannot be detected, an unspecified transition fault is used instead. An unspecified transition fault captures the behaviors of transition faults with different extra delays. When this fault cannot be detected, a stuck-at fault is used instead. A stuck-at fault has some of the detection conditions of a transition fault. Multicycle functional broadside tests are used to allow unspecified transition faults to be detected. As a by-product, test compaction also occurs. The structure of the test generation procedure accommodates the complexity of producing functional broadside tests by considering the target as well as replacement faults together. Experimental results for benchmark circuits demonstrate the fault coverage improvements achieved, and the effect on the number of tests.

CCS Concepts: •**Hardware →Test-pattern generation and fault simulation;**

Additional Key Words and Phrases: Functional broadside tests, multicycle tests, test compaction, test generation, transition faults.

ACM Reference format:
Irith Pomeranz. 2020. Covering Test Holes of Functional Broadside Tests. *ACM Trans. Des. Autom. Electron. Syst.* 11, 11, Article 11 (November 2020), 15 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Delay defects are prevalent in state-of-the-art technologies and require tests for delay faults to be applied to manufactured circuits [1]-[12]. Functional operation conditions during the application of scan-based tests are important for avoiding overtesting of delay faults [13]-[15]. Overtesting occurs when a scan-based test propagates transitions through a slow path that does not affect output values during functional operation [13]. It also occurs when a scan-based test creates excessive switching activity, resulting in voltage drops [14]-[15]. In both cases, the circuit appears to have a delay fault that slows it down even though it would operate correctly during functional operation.

Author's address: Irith Pomeranz, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA. E-mail: pomeranz@ecn.purdue.edu.
The work was supported in part by the National Science Foundation (NSF) under Grant No. CCF-1714147.

DOI: 0000001.0000001

Functional broadside tests address overtesting by maintaining functional operation conditions during their functional capture cycles [16]-[20]. The delay fault coverage achievable by functional broadside tests is lower than that achievable by scan-based tests that are not constrained to be functional. Scan-based tests that maintain close-to-functional operation conditions address the fault coverage gap by considering the tradeoff between the fault coverage and the deviation from functional operation conditions [21]-[27].

Before resorting to close-to-functional broadside tests to increase the fault coverage, this paper explores the possibility that a more comprehensive functional broadside test set can be obtained by targeting additional faults with similar (but not identical) detection conditions instead of the target faults that cannot be detected. A more comprehensive functional broadside test set has the advantage that it still maintains functional operation conditions. This approach addresses the test holes created when target faults cannot be detected. The test holes (or the presence of undetected target faults) are covered by targeting additional faults from different fault models with similar detection conditions. Such approaches for covering the test holes of a test set were suggested in [28]-[30] for different contexts not involving functional broadside tests. The unique challenges with functional broadside tests are the following.

(1) The fault coverage achievable by functional broadside tests is lower than with scan-based tests that are not constrained to be functional. Thus, more target faults need to be replaced to ensure that test holes are covered, and more replacement faults may also be undetectable.

(2) Test generation in [28]-[30] first considers the target faults, and then considers the test holes created by target faults that cannot be detected. This structure of the test generation procedure is inefficient for functional broadside tests for the following reason. A fault-oriented test generation procedure for functional broadside tests is computationally intensive. Instead, fault-independent simulation-based procedures provide a more cost-effective option. Instead of considering fault models one at a time and repeating the process of producing functional broadside tests for every fault model, a more efficient approach is to consider all the fault models together. This is the approach suggested in this paper. Under this approach, all the target as well as replacement faults are initially undetected. As long as a target fault is not detected, its replacements are considered to cover the potential test hole. When a target fault is detected, the replacements become unnecessary, and the test generation procedure eliminates them and their tests from consideration.

The paper considers the scenario where the target faults for functional broadside tests are transition faults. When a standard transition fault, with an extra delay of a single clock cycle, cannot be detected by a functional broadside test, an unspecified transition fault [31] is used instead. An unspecified transition fault captures, in a single fault, transition faults with different extra delays. Even if the standard transition fault cannot be detected, transition faults with different extra delays may be detected. These faults are targeted to cover the test hole, created when a standard transition fault cannot be detected, by targeting an unspecified transition fault. When the unspecified transition fault cannot be detected, a stuck-at fault is used instead. A stuck-at fault is suitable because it has some of the detection conditions of a transition fault.

The goal of covering test holes is not to detect all the unspecified transition faults and stuck-at faults, since this would result in a significantly larger test set, and the extra tests would cover similar faults. Instead, the goal is only to address the test holes left by standard transition faults that cannot be detected. Faults with similar (but not identical) detection conditions are suitable for this purpose since their tests are likely to detect the same defects. The use of two fault types as replacement faults increases the likelihood that a detectable replacement will be found. The structure of the test generation procedure ensures that the generation of functional broadside tests does not have to be repeated.

Considering the numbers of functional capture cycles in a test, stuck-at faults can be detected by single-cycle tests, whereas transition faults require tests with two or more clock cycles. To maintain a similarity to transition faults, tests with two or more clock cycles are also used for stuck-at faults.

A significant difference between the ability to detect a standard and an unspecified transition fault exists only under multicycle tests with more than two functional capture cycles. Multicycle tests are also effective for test compaction. Therefore, the test generation procedure considers multicycle tests.

The procedure is implemented by extracting functional broadside tests from functional test sequences [17]. Two versions of the test generation procedure are described. The first procedure will allow a detailed comparison of a two-cycle functional broadside test set for standard transition faults with a multicycle test set that covers test holes. After extracting an initial two-cycle functional broadside test set for standard transition faults, the procedure extracts two-cycle, three-cycle, four-cycle, ... tests to cover test holes.

To take better advantage of the ability of multicycle tests to provide test compaction, and avoid generating functional broadside tests sequentially for different fault models, the second procedure prefers to use multicycle tests with larger numbers of clock cycles, and considers all the fault types together.

The use of multicycle tests has at most a small effect on the fault coverage achievable for standard transition faults, whereas the fault coverage of unspecified transition faults increases significantly when multicycle tests are used. This observation was used in [32] to guide the selection of multicycle test sets that provide both test compaction and an increased defect coverage. The defect coverage in [32] is represented by unspecified transition faults. Accordingly, the differences between this work and [32] are the following.

(1) In [32] all the unspecified transition faults are considered as part of a test compaction procedure, whereas here, only unspecified transition faults that correspond to undetected standard transition faults are important to detect. The goal, which is not addressed in [32], is to cover the test holes of a functional broadside test set.

(2) The tests in [32] are not constrained to be functional broadside tests, requiring substantially different underlying test generation and test compaction processes.

(3) Stuck-at faults are not considered in [32].

The paper is organized as follows. Background related to functional broadside tests and transition faults is provided in Section 2. The test generation procedures for functional broadside tests are described in Section 3. Experimental results for benchmark circuits are given in Section 4.

2 BACKGROUND

This section provides background for the test generation procedure described in this paper.

2.1 Functional Broadside Tests

The procedure used in this paper for generating functional broadside tests extracts the tests from functional test sequences. Let $V_i = v_{i,0}v_{i,1}...v_{i,L-1}$ be a functional test sequence of length L. Logic simulation of the sequence is carried out starting from the initial state used for functional operation of the circuit. This results in a state $s_{i,u}$ for every clock cycle $0 \leq u \leq L$. Four clock cycles of the sequence, u, $u+1$, $u+2$ and $u+3$, are shown in Figure 1.

A two-cycle functional broadside test can be extracted from V_i by considering any two consecutive clock cycles. Considering clock cycles u and $u+1$ in Figure 1, a functional broadside test can be obtained that has a scan-in state $s_{i,u}$, and primary input vectors $v_{i,u}$ and $v_{i,u+1}$. After scanning in

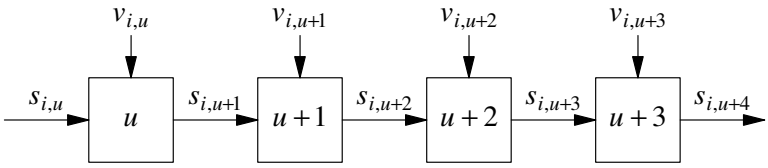

Fig. 1. Functional test sequence

$s_{i,u}$, the test applies $v_{i,u}$ and $v_{i,u+1}$ in two consecutive functional capture cycles. The final state is scanned out. The test is denoted by $t_{i,u,2} = \langle s_{i,u}, v_{i,u}, v_{i,u+1} \rangle$.

Assuming that V_i can occur during functional operation, the test $t_{i,u,2}$ takes the circuit through two state-transitions that can occur during functional operation. Therefore, it is a functional broadside test.

A two-cycle functional broadside test can be extracted from V_i for every $0 \leq u \leq L-2$. An l-cycle functional broadside test $t_{i,u,l} = \langle s_{i,u}, v_{i,u}, v_{i,u+1}, ..., v_{i,u+l-1} \rangle$ can be obtained from V_i for every $l \geq 2$ and $0 \leq u \leq L-l$. The test $t_{i,u,l}$ starts by scanning in $s_{i,u}$. The primary input vectors $v_{i,u}$, $v_{i,u+1}$, ..., $v_{i,u+l-1}$ are applied in l consecutive functional capture cycles. The final state is scanned out. Referring to Figure 1, $t_{i,u,3} = \langle s_{i,u}, v_{i,u}, v_{i,u+1}, v_{i,u+2} \rangle$, $t_{i,u,4} = \langle s_{i,u}, v_{i,u}, v_{i,u+1}, v_{i,u+2}, v_{i,u+3} \rangle$, and so on.

2.2 Transition Faults

A two-cycle scan-based test for the transition fault $f = g : a \rightarrow a'$ satisfies the following conditions. (1) It assigns $g = a$ under the first cycle. (2) It assigns $g = a'$ under the second cycle. (3) In the presence of the fault, $g = a$ is obtained under the second cycle. The test propagates the fault effect a'/a from g to an observable output under the second cycle. This implies that the second cycle is a test for the fault g stuck-at a.

Under a multicycle scan-based test, or under a functional test sequence, a transition fault may have a different effect on the circuit depending on the duration of the extra delay. In [33], the duration of the extra delay is measured in numbers of clock cycles, and transition faults are associated with different durations. Each duration defines a different transition fault, and each fault is simulated separately.

A standard transition fault is associated with a duration of a single clock cycle. Only standard transition faults are typically targeted by test generation procedures to avoid the increase in the number of faults when different durations are considered.

Transition faults with different durations than a standard transition fault are suitable for covering the test hole created when a standard transition fault cannot be detected. However, the number of target faults would increase significantly if faults of different durations are considered individually.

An unspecified transition fault captures all the durations of a transition fault in a single fault. Consequently, the number of unspecified transition faults is the same as the number of standard transition faults, but the faults capture a broader range of durations. This makes unspecified transition faults suitable for covering test holes created when standard transition faults cannot be detected.

This property of unspecified transition faults is achieved by introducing unspecified (x) values into the faulty circuit when fault effects may occur. The first activation of the fault requires a transition as in the case of a standard transition fault. The difference is that fault activation assigns an unspecified value to the fault site. The fault effect is then propagated using unspecified values. This is done to accommodate the fact that the duration of the fault is not specified. The fault may be

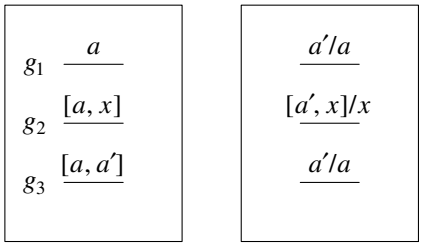

Fig. 2. Activation conditions

activated again using unspecified values as described later. An unspecified value propagated to an observable output is taken as an indication that the fault is detected. To increase the confidence that the fault is indeed detected, it is possible to require that unspecified values would be propagated to several observable outputs at one or more clock cycles.

Fault simulation of unspecified transition faults is carried out at the gate-level as for standard transition faults. The only difference is that, even under fully-specified tests, fault simulation of unspecified transition faults uses three values, $\{0, 1, x\}$. The activation conditions of the faults are illustrated by Figure 2 and considered in more detail next. For simplicity of discussion the tests are assumed to be fully-specified. Figure 2 shows two consecutive clock cycles of a test.

A standard transition fault $g : a \rightarrow a'$ is activated during two clock cycles of a test if $g = a$ in the first clock cycle, and $g = a'$ in the second clock cycle. This results in the value $g = a$ in the faulty circuit under the second cycle. A standard transition fault is illustrated by line g_1 in Figure 2.

Let x denote an unspecified value related to fault activation (if the test leaves unspecified values in the fault-free circuit they are given a different symbol). An unspecified transition fault $g : a \rightarrow a'$ is activated during two clock cycles of a test if $g = a$ or x in the first clock cycle, and $g = a'$ or x in the second clock cycle. This results in the value $g = x$ in the faulty circuit under the second cycle. An unspecified transition fault is illustrated by line g_2 in Figure 2.

The first time the fault is activated there are no unspecified values in the circuit that result from fault activation. Therefore, the first activation of the unspecified transition fault occurs as for a standard transition fault. During additional clock cycles, the fault may be activated again with $g = x$ in the first or second clock cycle. Similar to a standard transition fault, only two consecutive clock cycles are considered to determine whether the fault is activated.

Based on this discussion, the difference between standard and unspecified transition faults becomes more evident with more functional capture cycles between the scan operations of a test. In general, multicycle tests with more functional capture cycles allow more unspecified transition faults to be detected.

For completeness, the activation of the fault g stuck-at a is illustrated by line g_3 in Figure 2. The fault does not have any requirements for the first clock cycle, allowing either an a or an a' to be assigned in this clock cycle. In the second clock cycle, $g = a'$ activates the fault. This results in the value $g = a$ in the faulty circuit under the second cycle.

2.3 Target Faults

To explain how fault detection information is updated when the set of faults contains replacements to the target faults, let us consider a transition fault $f = g : a \rightarrow a'$. The fault is associated with three flags. The flag $str(f)$ is related to the standard transition fault associated with f. The flag

$xtr(f)$ is related to the unspecified transition fault associated with f. The flag $ssa(f)$ is related to the single stuck-at fault associated with f, which is g stuck-at a.

Initially, $str(f) = xtr(f) = ssa(f) = 0$ indicates that the fault is not detected. If a test is found for the standard transition fault, $str(f) = xtr(f) = ssa(f) = 1$ is assigned. Although the unspecified transition fault and stuck-at fault may not be detected, they do not require additional tests. The reason is that the tests are likely to be similar, and the goal is only to cover test holes.

If a test is found for the unspecified transition fault, $xtr(f) = ssa(f) = 1$ is assigned. In this case, the stuck-at fault does not require an additional test. If the standard transition fault is detected later, the assignment $str(f) = xtr(f) = ssa(f) = 1$ makes the test for the unspecified transition fault unnecessary.

If a test is found for the stuck-at fault, $ssa(f) = 1$ is assigned. If the standard or unspecified transition fault is detected later, the assignment $str(f) = xtr(f) = ssa(f) = 1$ or $xtr(f) = ssa(f) = 1$, respectively, makes the test for the stuck-at fault unnecessary.

3 TEST GENERATION PROCEDURES

The test generation procedure for functional broadside tests is described in this section. Two versions of the procedure are described. The first version, given by Procedures 1 and 2, uses multicycle tests only for eliminating test holes. These procedures will demonstrate the importance of multicycle tests for covering test holes. The second version, given by Procedure 3, uses multicycle tests for test compaction as well.

Procedure 3 is the one to be used for producing a compact comprehensive functional broadside test set that covers test holes created by standard transition faults that cannot be detected. Procedures 1 and 2 are included to demonstrate more clearly the effects of covering test holes.

The set of target faults is denoted by F. It consists of a transition fault $g : a \rightarrow a'$ for every line g and value $a \in \{0, 1\}$. A transition fault $f \in F$ is associated with three flags as defined earlier. Initially, $str(f) = xtr(f) = ssa(f) = 0$.

3.1 Initial Functional Broadside Test Set

In the first version of the procedure, a two-cycle functional broadside test set T_2^1 for standard transition faults is first obtained by applying Procedure 1 given below. The superscript 1 of T_2^1 indicates that only one type of faults is targeted. By targeting only standard transition faults, the test set provides a baseline for comparison. A more comprehensive test set will be obtained by targeting unspecified transition faults and stuck-at faults, and using multicycle tests, as described in Section 3.2.

Procedure 1 uses N functional test sequences, V_0, V_1, ..., V_{N-1}. It considers every two-cycle functional broadside test $t_{i,u,2}$, for $0 \leq i \leq N-1$ and $0 \leq u \leq L-2$. It simulates under $t_{i,u,2}$ every standard transition fault $f \in F$ such that $str(f) = 0$. The procedure updates which faults are detected, and adds the test to T_2^1 if any faults are detected.

Procedure 1: Initial Functional Broadside Test Set

(1) Assign $str(f) = xtr(f) = ssa(f) = 0$ for every $f \in F$.
(2) Assign $T_2^1 = \emptyset$.
(3) For $i = 0, 1, ..., N-1$:
    (a) For $u = 0, 1, ..., L-2$:
        (i) Simulate every fault $f \in F$ such that $str(f) = 0$ under $t_{i,u,2}$ as a standard transition fault. If the fault is detected, assign $str(f) = xtr(f) = ssa(f) = 1$.
        (ii) If any fault is detected, add $t_{i,u,2}$ to T_2^1.

(b) Apply to T_2^1 forward-looking reverse order fault simulation to remove unnecessary tests.

(4) For every test $t_{i,u,2} \in T_2^1$:
   (a) Simulate every fault $f \in F$ such that $xtr(f) = 0$ under $t_{i,u,2}$ as an unspecified transition fault. If the fault is detected, assign $xtr(f) = ssa(f) = 1$.
   (b) Simulate every fault $f \in F$ such that $ssa(f) = 0$ under $t_{i,u,2}$ as a stuck-at fault. If the fault is detected, assign $ssa(f) = 1$.

After every sequence is used for extracting functional broadside tests, the procedure applies forward-looking reverse order fault simulation to remove tests that become unnecessary after other tests are added to the test set.

At the end of Procedure 1, fault simulation is carried out for unspecified transition faults and stuck-at faults to determine which faults are detected accidentally. Fault simulation of unspecified transition faults is carried out for every fault $f \in F$ with $str(f) = 0$. If the unspecified transition fault is detected, $xtr(f) = ssa(f) = 1$ is assigned. Fault simulation of stuck-at faults is carried out for every fault $f \in F$ with $xtr(f) = 0$. If the stuck-at fault is detected, $ssa(f) = 1$ is assigned.

The computational effort of Procedure 1 is that of fault simulation with fault dropping of standard transition faults under $N(L - 1)$ two-cycle tests.

3.2 Comprehensive Functional Broadside Test Set

To cover the test holes of T_2^1, and obtain a more comprehensive functional broadside test set, the procedure considers unspecified transition faults and stuck-at faults under l-cycle tests, for $l = 2, 3, \ldots, l_{MAX}$, where l_{MAX} is a constant. The use of $l = 2$ is not expected to increase the fault coverage significantly. It is included to demonstrate that the difference between standard and unspecified transition faults is important for larger values of l.

The l-cycle test set is denoted by T_l^3. The superscript 3 indicates that three types of faults are targeted. Initially for $l = 2$, $T_2^3 = T_2^1$. For $l > 2$, $T_l^3 = T_{l-1}^3$ initially. Procedure 2 extends T_l^3 into a more comprehensive test set by considering N new functional test sequences. The sequences are $V_N, V_{N+1}, \ldots, V_{2N-1}$ for $l = 2$, $V_{2N}, V_{2N+1}, \ldots, V_{3N-1}$ for $l = 3$, and so on.

Procedure 2: Comprehensive l-Cycle Functional Broadside Test Set

(1) If $l = 2$, assign $T_2^3 = T_2^1$. Otherwise, assign $T_l^3 = T_{l-1}^3$.
(2) For $i = (l - 1)N, (l - 1)N + 1, \ldots, lN - 1$:
   (a) For $u = 0, 1, \ldots, L - l$:
      (i) Simulate every fault $f \in F$ such that $str(f) = 0$ under $t_{i,u,l}$ as a standard transition fault. If the fault is detected, assign $str(f) = xtr(f) = ssa(f) = 1$.
      (ii) Simulate every fault $f \in F$ such that $xtr(f) = 0$ under $t_{i,u,l}$ as an unspecified transition fault. If the fault is detected, assign $xtr(f) = ssa(f) = 1$.
      (iii) Simulate every fault $f \in F$ such that $ssa(f) = 0$ under $t_{i,u,l}$ as a stuck-at fault. If the fault is detected, assign $ssa(f) = 1$.
      (iv) If any fault is detected, add $t_{i,u,l}$ to T_l^3.
   (b) Apply to T_l^3 forward-looking reverse order fault simulation to remove unnecessary tests.

As discussed earlier, Procedure 2 uses an unspecified transition fault only when the standard transition fault is not detected. It uses a stuck-at fault only when the unspecified transition fault is not detected. Consequently, it is possible to obtain $xtr(f) = 1$ when the standard transition fault is detected without considering the unspecified transition fault. In addition, it is possible to obtain $ssa(f) = 1$ when the standard or unspecified transition fault is detected without considering the

stuck-at fault. This is consistent with the goal of using unspecified transition faults and single stuck-at faults only to cover test holes.

To avoid having to consider the same test more than once, Procedure 2 considers all the three types of faults under every test. As a result, it is possible that the procedure will first assign $ssa(f) = 1$ with $str(f) = xtr(f) = 0$. If the unspecified transition fault is detected by a test considered later, the procedure will result in $xtr(f) = ssa(f) = 1$ with $str(f) = 0$. If the standard transition fault is detected later, the procedure will result in $str(f) = xtr(f) = ssa(f) = 1$. The last test to detect the fault is considered by the forward-looking reverse order fault simulation procedure, and the procedure requires the same assignment to $str(f)$, $xtr(f)$ and $ssa(f)$ by the test used for detecting the fault. This ensures that the forward-looking reverse order fault simulation procedure will remove a test that becomes unnecessary after an unspecified or standard transition fault is detected.

3.3 Compact Comprehensive Functional Broadside Test Set

With Procedures 1 and 2, multicycle tests are used only for covering test holes. Procedure 3 given below uses multicycle tests for test compaction as well. In addition, it does not attempt to first exhibit the test holes, but considers all three fault types together. This is important for avoiding the generation of functional broadside tests repeatedly for different fault models.

The procedure extracts l-cycle tests, cycling through $l = l_{MAX}, l_{MAX} - 1, ..., 2$ as it considers additional functional test sequences. By starting with $l = l_{MAX}$ it gives a higher priority to multicycle tests with a higher number of clock cycles. This results in a reduced number of tests. The number of sequences for Procedure 3 is denoted by M.

Procedure 3: Compact Comprehensive Functional Broadside Test Set

(1) Assign $str(f) = xtr(f) = ssa(f) = 0$ for every $f \in F$.
(2) Assign $T = \emptyset$. Assign $l = l_{MAX}$.
(3) For $i = 0, 1, ..., M - 1$:
    (a) For $u = 0, 1, ..., L - l$:
        (i) Simulate every fault $f \in F$ such that $str(f) = 0$ under $t_{i,u,l}$ as a standard transition fault. If the fault is detected, assign $str(f) = xtr(f) = ssa(f) = 1$.
        (ii) Simulate every fault $f \in F$ such that $xtr(f) = 0$ under $t_{i,u,l}$ as an unspecified transition fault. If the fault is detected, assign $xtr(f) = ssa(f) = 1$.
        (iii) Simulate every fault $f \in F$ such that $ssa(f) = 0$ under $t_{i,u,l}$ as a stuck-at fault. If the fault is detected, assign $ssa(f) = 1$.
        (iv) If any fault is detected, add $t_{i,u,l}$ to T.
    (b) Apply to T forward-looking reverse order fault simulation to remove unnecessary tests.
    (c) Assign $l = l - 1$. If $l < 2$, assign $l = l_{MAX}$.

An example of Procedure 3 is shown in Table 1. The example is based on seven of the faults of benchmark circuit $s27$, f_0, f_1, ..., f_6, under nine functional broadside tests, t_0, t_1, ..., t_8. The first column of Table 1 shows the indices of the tests. The second column shows their numbers of functional capture cycles. Next, there is a column for every one of the faults. When a test t_i detects one of the faults associated with f_j, the table shows the flags $str(f_j)xtr(f_j)ssa(f_j)$.

In the case of f_0, the standard transition fault is detected by t_2. In the case of f_1, the unspecified transition fault is detected by t_0, and the standard transition fault is detected by t_3. The test holes correspond to f_4 and f_6. In the case of f_4, the unspecified transition fault is detected to cover the test hole. In the case of f_6, the stuck-at fault is detected to cover the test hole.

Table 1. Example of Procedure 3

		0	1	2	3	4	5	6
-	-	000	000	000	000	000	000	000
0	4	-	011	001	-	001	-	001
1	4	-	-	111	-	-	-	-
2	4	111	-	-	-	-	-	-
3	4	-	111	-	-	-	-	-
4	4	-	-	-	011	-	-	-
5	4	-	-	-	-	-	001	-
6	3	-	-	-	-	011	-	-
7	2	-	-	-	111	-	-	-
8	3	-	-	-	-	-	111	-

4 EXPERIMENTAL RESULTS

The results of Procedures 1, 2 and 3 for benchmark circuits are presented in this section.

4.1 Procedures 1 and 2

Procedures 1 and 2 are applied to several circuits with $N = 32$ functional test sequences for every value of l. The length of a functional test sequence is $L = 1024$. The sequences are generated by the procedure described in [20]. These parameters are sufficient for achieving the highest or close to the highest standard transition fault coverage achievable using functional broadside tests.

Multicycle functional broadside tests with 2, 3, ..., $l_{MAX} = 8$ clock cycles are extracted. This value of l_{MAX} is sufficient for demonstrating the effects of considering unspecified transition faults. Larger values may increase the fault coverage further for some circuits.

Three coverage metrics are computed for every test set T_l^n. The percentage of faults with $str(f) = 1$ is denoted by $\phi_{str}(T_l^n)$. The percentage of faults with $xtr(f) = 1$ is denoted by $\phi_{xtr}(T_l^n)$. The percentage of faults with $ssa(f) = 1$ is denoted by $\phi_{ssa}(T_l^n)$. Because of the way fault detection information is updated, we have that $\phi_{str}(T_l^n) \leq \phi_{xtr}(T_l^n) \leq \phi_{ssa}(T_l^n)$.

The results are shown in Tables 2 and 3 for the following test sets: (1) T_2^1; (2) T_2^3 if at least one of the three coverage metrics is increased; and (3) $T_{l_{max}}^3$ where $l_{max} \leq l_{MAX}$ is the largest value of l for which at least one of the three coverage metrics is increased. The circuits are ordered by increasing coverage metric for standard transition faults $\phi_{str}(T_{l_{max}}^3)$ achieved by the final test set $T_{l_{max}}^3$. A low coverage metric for standard transition faults is obtained for circuits with high levels of redundancy.

For every test set, after the circuit name, column *sv* shows the number of state variables, and column *pi* shows the number of primary inputs. Column *targ* shows how many fault types are targeted, where 1 stands for standard transition faults, and 3 stands for all three fault types. Column *tests* shows the number of tests in the test set. Column $func$ shows the maximum and average number of functional capture cycles in a test. Column *cov* shows the coverage metrics with respect to standard transition faults, unspecified transition faults, and stuck-at faults. Column *ntime* shows the cumulative runtime, divided by the runtime for computing T_2^1. This is referred to as the normalized runtime.

The following points can be seen from Tables 2 and 3. Comparing T_2^3 with T_2^1, all three coverage metrics typically do not increase, or increase to a small extent, by extracting additional two-cycle tests and considering all three fault types.

Table 2. Results of Procedures 1 and 2 ($\phi_{str}(T^3_{l_{max}}) < 75\%$)

					func		cov			
circuit	sv	pi	targ	tests	max	ave	str	xtr	ssa	ntime
steppermotordrive	25	3	1	23	2	2.00	27.646	27.646	56.085	1.00
steppermotordrive	25	3	3	24	2	2.00	27.646	27.646	56.217	3.62
steppermotordrive	25	3	3	19	5	2.63	27.646	31.085	56.217	19.00
b05	34	2	1	63	2	2.00	41.129	41.129	62.433	1.00
b05	34	2	3	44	8	4.43	41.129	51.848	62.433	47.64
b07	51	2	1	43	2	2.00	55.062	55.114	78.822	1.00
b07	51	2	3	44	2	2.00	55.062	55.114	78.977	3.35
b07	51	2	3	33	8	4.00	55.062	68.337	78.977	38.80
b03	30	5	1	34	2	2.00	58.464	58.464	74.870	1.00
b03	30	5	3	37	2	2.00	58.464	58.464	75.911	3.58
b03	30	5	3	35	5	3.00	58.464	65.365	75.911	18.37
usb_phy	98	14	1	104	2	2.00	58.354	58.354	75.186	1.00
usb_phy	98	14	3	106	2	2.00	58.685	58.685	75.434	3.24
usb_phy	98	14	3	103	8	3.17	58.933	63.896	75.517	37.44
s526	21	3	1	70	2	2.00	62.072	62.072	85.551	1.00
s526	21	3	3	72	2	2.00	62.072	62.072	86.027	3.22
s526	21	3	3	69	8	3.17	62.357	76.426	86.122	34.54
simple_spi	131	15	1	145	2	2.00	67.461	67.461	85.079	1.00
simple_spi	131	15	3	152	2	2.00	67.984	67.984	85.576	3.06
simple_spi	131	15	3	144	8	3.08	69.005	74.398	85.916	32.63
s5378	179	35	1	188	2	2.00	72.059	72.125	78.442	1.00
s5378	179	35	3	190	2	2.00	72.257	72.323	78.451	3.28
s5378	179	35	3	172	8	2.63	72.455	73.947	78.536	38.66
b14	247	33	1	424	2	2.00	73.568	73.597	82.066	1.00
b14	247	33	3	445	2	2.00	74.127	74.139	82.456	3.31
b14	247	33	3	434	8	2.79	74.540	80.861	82.712	25.04

Comparing $T^3_{l_{max}}$ with T^1_2, the coverage metric for standard transition faults, and the coverage metric for single stuck-at faults, typically do not increase, or increase to a small extent, by extracting multicycle tests and considering all three fault types. A significant increase typically occurs in the coverage metric for unspecified transition faults. Since the other coverage metrics increase at most slightly, the implication is that faults, which are detected only as single stuck-at faults by two-cycle tests, are detected as unspecified transition faults when multicycle tests are extracted. This is preferred since unspecified transition faults are closer to standard transition faults in that they represent delay defects with different durations.

Considering the final functional broadside test set, the coverage metric for unspecified transition faults is typically significantly higher than that of standard transition faults. In addition, the coverage metric for single stuck-at faults is typically significantly higher than that of unspecified transition faults. Thus, every additional fault type helps cover test holes left by the previous fault types. When Procedure 3 is applied, the coverage metrics will be used for demonstrating the presence of test holes that can be covered by considering unspecified transition faults and stuck-at faults.

Table 3. Results of Procedures 1 and 2 ($\phi_{str}(T^3_{l_{max}}) \geq 75\%$)

					func		cov			
circuit	sv	pi	targ	tests	max	ave	str	xtr	ssa	ntime
s382	21	3	1	40	2	2.00	76.047	76.047	95.550	1.00
s382	21	3	3	41	2	2.00	76.047	76.047	95.942	2.90
s382	21	3	3	35	6	2.80	76.047	83.770	95.942	17.87
b11	30	8	1	85	2	2.00	76.339	76.339	89.617	1.00
b11	30	8	3	88	2	2.00	76.339	76.339	89.781	3.15
b11	30	8	3	86	8	3.50	76.448	86.995	89.781	34.87
b09	28	2	1	36	2	2.00	76.106	76.106	91.888	1.00
b09	28	2	3	42	2	2.00	76.844	76.844	92.920	3.14
b09	28	2	3	36	8	2.92	76.844	85.103	92.920	32.24
b08	21	10	1	60	2	2.00	81.872	82.346	99.526	1.00
b08	21	10	3	61	2	2.00	81.872	82.346	99.882	2.68
b08	21	10	3	60	8	3.82	81.872	97.512	100.000	21.53
s1423	74	17	1	109	2	2.00	80.991	81.413	93.359	1.00
s1423	74	17	3	119	2	2.00	82.818	83.169	94.940	2.86
s1423	74	17	3	120	8	3.46	83.591	92.375	95.538	23.28
b04	66	12	1	92	2	2.00	84.501	84.501	92.557	1.00
b04	66	12	3	98	2	2.00	84.501	84.764	92.907	3.41
b04	66	12	3	93	7	2.67	84.545	91.156	92.907	29.69
sasc	117	15	1	116	2	2.00	84.465	84.465	98.858	1.00
sasc	117	15	3	120	2	2.00	85.183	85.183	99.151	2.48
sasc	117	15	3	144	8	3.29	86.488	93.799	99.347	19.01
i2c	128	17	1	177	2	2.00	81.748	81.748	94.033	1.00
i2c	128	17	3	192	2	2.00	83.986	83.986	95.291	2.51
i2c	128	17	3	179	8	3.75	86.597	90.769	96.573	18.96
spi	229	45	1	902	2	2.00	86.823	88.402	98.546	1.00
spi	229	45	3	996	2	2.00	89.154	90.809	99.056	2.97
spi	229	45	3	924	8	3.27	93.633	96.758	99.624	13.25
s953	29	16	1	138	2	2.00	93.966	93.966	99.318	1.00
s953	29	16	3	136	2	2.00	94.281	94.281	99.370	2.16
s953	29	16	3	129	7	2.37	94.334	99.370	99.370	11.06
systemcdes	190	130	1	212	2	2.00	99.660	99.660	99.975	1.00
systemcdes	190	130	3	201	5	2.08	99.660	99.901	99.975	5.19
des_area	128	239	1	310	2	2.00	100.000	100.000	100.000	1.00

In many cases, the use of multicycle tests allows the number of tests to be lower for a more comprehensive test set. It should be noted in this regard that not all the unspecified transition faults and single stuck-at faults are detected, since this would have required additional tests, and the tests would have been similar to the ones that detect standard transition faults. The goal for computing the more comprehensive test set is only to cover the test holes created when standard transition faults are not detected. Covering these test holes typically does not require a larger number of tests when multicycle tests are used.

Table 4. Results of Procedure 3

					func		cov			
circuit	sv	pi	seq	tests	max	ave	str	xtr	ssa	ntime
steppermotordrive	25	3	2	12	8	7.92	27.646	31.085	56.217	1.82
b05	34	2	5	32	8	7.34	41.129	51.848	62.433	3.51
b07	51	2	5	19	8	7.79	55.062	68.337	78.977	3.43
b03	30	5	3	20	8	7.90	58.464	65.365	75.911	2.57
usb_phy	98	14	100	87	8	6.61	58.850	63.896	75.517	46.43
s526	21	3	120	50	8	5.80	62.072	76.331	86.027	38.98
simple_spi	131	15	125	105	8	6.55	68.482	73.482	85.785	49.10
s5378	179	35	122	123	8	6.20	72.370	73.834	78.508	53.35
b14	247	33	125	342	8	5.73	74.412	80.710	82.672	48.25
s382	21	3	43	24	8	5.75	76.047	83.770	95.942	15.12
b11	30	8	120	64	8	6.48	76.393	86.995	89.781	43.32
b09	28	2	61	37	8	6.68	76.844	85.103	92.920	25.91
b08	21	10	23	48	8	7.44	81.872	97.512	100.000	7.04
s1423	74	17	122	97	8	6.04	83.345	92.129	95.221	36.76
b04	66	12	72	51	8	7.24	84.545	91.156	92.907	30.89
i2c	128	17	127	153	8	5.85	85.641	90.163	96.340	29.66
sasc	117	15	128	105	8	6.47	86.358	93.734	99.347	25.88
spi	229	45	128	708	8	5.18	91.695	95.856	99.457	33.14
s953	29	16	72	74	8	6.39	94.334	99.370	99.370	4.17
systemcdes	190	130	3	57	8	7.82	99.660	99.901	99.975	1.55
des_area	128	239	2	68	8	7.99	100.000	100.000	100.000	1.25
s9234	228	19	121	73	8	6.14	15.378	17.598	34.541	54.30
tv80	359	13	105	713	8	5.67	45.744	57.885	74.763	90.38
wb_dma	523	215	111	135	8	5.30	63.880	64.847	79.198	37.97
s38584	1452	12	45	755	8	5.51	65.137	69.735	76.674	36.65
b15	447	36	128	791	8	5.52	69.814	82.310	92.557	64.02
b22	709	33	16	722	8	5.21	74.479	81.198	85.276	8.82
systemcaes	670	258	106	242	8	6.51	79.977	88.358	96.072	13.95
b20	494	33	128	694	8	5.34	80.023	86.915	89.029	59.04
s35932	1728	35	5	45	8	7.80	87.211	89.468	89.781	3.61
aes_core	530	258	4	238	8	7.34	99.939	99.992	99.999	1.81

The normalized runtime is similar for circuits of different sizes, and does not increase with the size of the circuit. This indicates that the procedure scales similar to the fault simulation procedure that produces T_2^1. With the parameters used, this procedure simulates 32,736 two-cycle tests.

4.2 Procedure 3

Procedure 3 is applied with $M = 128$ sequences of length $L = 1024$. It extracts multicycle tests with $l \leq l_{MAX} = 8$.

The results of Procedure 3 are given in Table 4. The format is similar to Tables 2 and 3 with the following differences. Column *targ* in Tables 2 and 3 is replaced with column *seq* in Table 4. This column shows the number of functional test sequences used for extracting functional broadside

Table 5. c-Cycle Transition Faults

circuit	1	2	3	4	5	6	7	8
steppermotordrive	27.646	29.894	29.894	31.085	31.085	31.085	31.085	31.085
b05	41.129	44.926	47.581	49.328	50.168	50.571	51.075	51.075
b07	55.062	58.471	63.895	64.979	67.355	67.924	68.079	68.079
b03	58.464	62.630	63.802	64.974	65.234	65.365	65.365	65.365
usb_phy	58.850	60.091	61.208	63.234	63.606	63.606	63.896	63.896
s526	62.072	72.053	73.099	73.954	74.240	75.190	76.331	76.331
simple_spi	68.482	70.628	71.597	72.749	73.063	73.325	73.455	73.455
s5378	72.370	72.805	73.003	73.390	73.437	73.768	73.768	73.768
b14	74.412	79.651	79.942	80.454	80.541	80.652	80.710	80.710
s382	76.047	80.890	81.283	82.723	83.115	83.246	83.770	83.770
b11	76.393	79.563	82.896	84.317	86.066	86.831	86.995	86.995
b09	76.844	80.826	81.563	82.743	83.481	84.218	85.103	85.103
b08	81.872	86.611	90.877	92.773	94.550	95.735	97.275	97.275
s1423	83.345	86.894	88.370	90.232	90.864	91.286	91.673	91.673
b04	84.545	85.858	88.310	89.405	90.061	90.630	90.893	90.893
i2c	85.641	87.599	88.042	88.741	89.091	89.627	90.140	90.140
sasc	86.358	89.785	91.384	92.722	93.179	93.505	93.734	93.734
spi	91.695	93.123	94.042	94.243	94.686	94.820	95.003	95.003
s953	94.334	97.587	98.269	98.636	98.793	99.161	99.370	99.370
systemcdes	99.660	99.801	99.867	99.901	99.901	99.901	99.901	99.901
s9234	15.378	15.670	15.925	15.984	17.029	17.067	17.089	17.089
wb_dma	63.880	64.230	64.629	64.738	64.756	64.828	64.847	64.847
b15	69.814	73.695	76.042	77.274	78.610	81.109	82.138	82.138
systemcaes	79.977	82.694	83.824	84.924	87.651	87.938	88.355	88.355
b20	80.023	84.735	85.268	86.387	86.497	86.751	86.806	86.806
s35932	87.211	88.416	88.829	89.058	89.443	89.455	89.468	89.468
aes_core	99.939	99.962	99.970	99.990	99.991	99.991	99.992	99.992

tests. Only the final results are shown for every circuit. The circuits are arranged to match Tables 2 and 3.

For the circuits that appear in both tables, the number of tests in Table 4 is lower because of the more extensive use of multicycle tests.

As in Tables 2 and 3, the coverage metric for unspecified transition faults is typically significantly higher than that of standard transition faults, and the coverage metric for single stuck-at faults is typically significantly higher than that of unspecified transition faults. Thus, every additional fault type helps cover additional test holes.

The normalized runtime is similar for circuits of different sizes, and depends more strongly on the number of sequences used for extracting functional broadside tests.

As discussed earlier, an unspecified transition fault is used instead of several transition faults with different extra delays. These faults are referred to as c-cycle transition faults with $c \geq 1$. It is interesting to consider the fault coverages obtained when c-cycle transition faults are simulated under the test sets produced by Procedure 3. The results of fault simulation are given in Table 5 for several of the circuits from Table 4 using $1 \leq c \leq 8$. Table 5 demonstrates that the c-cycle transition

fault coverage increases with c, and reaches a fault coverage that is close to the coverage metric of unspecified transition faults from Table 4.

5 CONCLUDING REMARKS

This paper described a procedure for obtaining a functional broadside test set for transition faults that is more comprehensive than a two-cycle test set for standard transition faults. To achieve this goal, the procedure targets unspecified transition faults and single stuck-at faults that have similar (but not identical) detection conditions to standard transition faults. These additional faults are considered only when the standard transition faults cannot be detected. The goal is to address the test holes created by undetected standard transition faults. The procedure extracts functional broadside tests from functional test sequences. It uses multicycle tests to cover unspecified transition faults and achieve test compaction. It considers the three fault models together to accommodate the complexity of producing functional broadside tests. Experimental results for benchmark circuits demonstrated the fault coverage improvements achieved by this approach. In many cases, the use of multicycle tests allows test holes to be covered without increasing the number of tests.

REFERENCES

[1] A. Sreedhar, A. Sanyal and S. Kundu, "On Modeling and Testing of Lithography Related Open Faults in Nano-CMOS Circuits", in Proc. Design, Autom. and Test in Europe Conf., 2008, pp. 616-621.
[2] M. O. Simsir, A. Bhoj and N. K. Jha, "Fault Modeling for FinFET Circuits", in Proc. Intl. Symp. on Nanoscale Architectures, 2010, pp. 41-46.
[3] J. Zha, X. Cui and C. L. Lee, "Modeling and Testing of Interference Faults in the Nano NAND Flash Memory", in Proc. Design, Automation & Test in Europe Conf., 2012, pp. 527-531.
[4] D. Xiang, Z. Chen and L.-T. Wang, "Scan Flip-Flop Grouping to Compress Test Data and Compact Test Responses for Launch-on-Capture Delay Testing", ACM Trans. on Design Automation, Vol. 17, No. 2, April 2012, Article No. 18.
[5] S.-Y. Huang, Y.-H. Lin, K.-H. Tsai, W.-T. Cheng, S. Sunter, Y.-F Chou and D.-M. Kwai, "Small Delay Testing for TSVs in 3-D ICs", in Proc. Design Automation Conf., 2012, pp. 1031-1036.
[6] A. Mondal, P. P. Chakrabarti and P. Dasgupta, "Symbolic-Event-Propagation-Based Minimal Test Set Generation for Robust Path Delay Faults", ACM Trans. on Design Automation, Vol. 17 No. 4, Oct. 2012, Article No. 47.
[7] M. Sauer, A. Czutro, I. Polian and B. Becker, "Small-delay-fault ATPG with Waveform Accuracy", in Proc. Intl. Conf. on Computer-Aided Design, 2012, pp. 30-36.
[8] W. Zhao, J. Ma, M. Tehranipoor and S. Chakravarty, "Power-safe Application of TDF Patterns to Flip-chip Designs during Wafer Test", ACM Trans. on Design Automation, Vol. 18, No. 3, July 2013, Article No. 43.
[9] S. Di Carlo, G. Gambardella, P. Prinetto, D. Rolfo and P. Trotta, "SATTA: A Self-Adaptive Temperature-Based TDF Awareness Methodology for Dynamically Reconfigurable FPGAs", ACM Trans. on Reconfigurable Technology and Systems, Vol. 8, No. 1, Feb. 2015, Article No. 1.
[10] Y. Zhang, Z. Peng, J. Jiang, H. Li and M. Fujita, "Temperature-aware Software-based Self-testing for Delay Faults", in Proc. Design, Automation & Test in Europe Conf., 2015, pp. 423-428.
[11] H. G. Mohammadi, P.-E. Gaillardon and G. De Micheli, "Fault Modeling in Controllable Polarity Silicon Nanowire Circuits", in Proc. Design, Automation & Test in Europe Conf., 2015, pp. 453-458.
[12] A. S. Trinadh, S. Potluri, S. B. Ch., V. Kamakoti and S. G. Singh, "Optimal Don't Care Filling for Minimizing Peak Toggles During At-Speed Stuck-At Testing", ACM Trans. on Design Automation, Vol. 23, No. 1, Oct. 2017, Article No. 5.
[13] J. Rearick, "Too Much Delay Fault Coverage is a Bad Thing", in Proc. Intl. Test Conf., 2001, pp. 624-633.
[14] 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. Intl. Test Conf., 2003, pp. 1098-1104.
[15] S. Sde-Paz and E. Salomon, "Frequency and Power Correlation between At-Speed Scan and Functional Tests", in Proc. Intl. Test Conf., 2008, Paper 13.3, pp. 1-9.
[16] I. Pomeranz, "On the Generation of Scan-Based Test Sets with Reachable States for Testing under Functional Operation Conditions", in Proc. Design Autom. Conf., 2004, pp. 928-933.
[17] I. Pomeranz and S. M. Reddy, "Generation of Functional Broadside Tests for Transition Faults", IEEE Trans. on Computer-Aided Design, Oct. 2006, pp. 2207-2218.
[18] M. Valka, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, E. Sanchez, M. De Carvalho and M. Sonza Reorda "A Functional Power Evaluation Flow for Defining Test Power Limits during At-Speed Delay Testing", in Proc.

IEEE European Test Symp., 2011, pp. 153-158.
[19] A. Touati, A. Bosio, L. Dilillo, P. Girard, A. Virazel, P. Bernardi and M. Sonza Reorda "Exploring the Impact of Functional Test Programs Re-used for Power-aware Testing", in Proc. Design, Automation & Test in Europe Conf., 2015, pp. 1277-1280.
[20] I. Pomeranz, "A Static Test Compaction Procedure for Large Pools of Multicycle Functional Broadside Tests", IET Computers & Digital Techniques, 2018.
[21] Y.-C. Lin, F. Lu, K. Yang and K.-T. Cheng, "Constraint Extraction for Pseudo-Functional Scan-Based Delay Testing", in Proc. Asia and South Pacific Design Autom. Conf., 2005, pp. 166-171.
[22] Z. Zhang, S. M. Reddy and I. Pomeranz, "On Generating Pseudo-Functional Delay Fault Tests for Scan Designs", in Proc. Intl. Symp. on Defect and Fault Tolerance in VLSI Systems, 2005, pp. 398-405.
[23] I. Polian and F. Fujiwara, "Functional Constraints vs. Test Compression in Scan-Based Delay Testing", in Proc. Design, Autom. and Test in Europe Conf., 2006, pp. 1-6.
[24] I. Pomeranz and S. M. Reddy, "Definition and Generation of Partially-Functional Broadside Tests", IET Computers & Digital Techniques, Jan. 2009, pp. 1-13.
[25] E. K. Moghaddam, J. Rajski, S. M. Reddy and M. Kassab, "At-Speed Scan Test with Low Switching Activity", in Proc. VLSI Test Symp., 2010, pp. 177-182.
[26] T. Zhang and D. M. H. Walker, "Power Supply Noise Control in Pseudo Functional Test", in Proc. VLSI Test Symp., 2013, pp. 1-6.
[27] I. Pomeranz, "Piecewise-Functional Broadside Tests Based on Reachable States", IEEE Trans. on Computers, Aug. 2015, pp. 2415-2420.
[28] I. Pomeranz, "A Bridging Fault Model for Line Coverage in the Presence of Undetected Transition Faults", in Proc. Design Autom. & Test in Europe Conf., 2017.
[29] I. Pomeranz, "Covering Undetected Transition Fault Sites with Optimistic Unspecified Transition Faults under Multi-cycle Tests", in Proc. European Test Symp., 2018, pp. 1-2.
[30] I. Pomeranz, "Iterative Test Generation for Gate-Exhaustive Faults to Cover the Sites of Undetectable Target Faults", in Proc. Intl. Test Conf., 2019, Paper 1.3, pp. 1-7.
[31] I. Pomeranz and S. M. Reddy, "Unspecified Transition Faults: A Transition Fault Model for At-Speed Fault Simulation and Test Generation", IEEE Trans. on Computer-Aided Design, Jan. 2008, pp. 137-146.
[32] I. Pomeranz, "Target Faults for Test Compaction based on Multicycle Tests", ACM Trans. on Design Automation, 2020.
[33] K.-T. Cheng, "Transition Fault Testing for Sequential Circuits", IEEE Trans. on Computer-Aided Design, Dec. 1993, pp. 1971-1983.

Received August 2020