Iterative Test Generation for Gate-Exhaustive Faults to Cover the Sites of Undetectable Target Faults

Irith Pomeranz
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907, U.S.A.
E-mail: pomeranz@ecn.purdue.edu

***Abstract*—Gate-exhaustive faults address the fact that not all the defect mechanisms and behaviors are known in advance, and not all of them can be translated into fault models. Therefore, it is advantageous to ensure that a test set covers unexpected defects by exhaustive testing of gates or subcircuits. This paper observes that these properties make gate-exhaustive faults suitable for providing extra coverage for sites where coverage is missing because of undetectable target faults from other fault models. Undetectable faults result from logic redundancy, and leave circuit sites uncovered. To allow subcircuits to be considered as gates while avoiding the need to consider large numbers of faults, the gate-exhaustive approach is applied selectively. Instead of using all the input patterns of every gate, the iterative procedure described in this paper uses increasing numbers of input patterns of gates that include undetectable target faults in order to achieve a coverage goal for these faults. Experimental results demonstrate the extent to which it is possible to cover the sites of undetectable single stuck-at faults using tests for gate-exhaustive faults.**

I. Introduction

Defects that are encountered frequently in fabricated chips provide the basis for the definition of fault models. Such fault models are used by test generation procedures to produce test sets that detect commonly occurring defects. For example, bridge defects are addressed by bridging faults [1], and open defects are addressed by transistor and interconnect open faults [2]. New fault models are defined, or existing ones are enhanced to address new defect behaviors in new technologies [3]-[6]. An understanding of defect behaviors also drives the definition of cell-aware faults [7]-[9]. A cell-aware approach identifies input patterns that are likely to exhibit the presence of defects in a cell. A test satisfies two conditions: (1) it assigns the input pattern to the inputs of the cell, and (2) it propagates the output value of the cell to an observable output.

Gate-exhaustive approaches are different in that they do not attempt to relate faults with specific defect behaviors [10]-[14]. This is suitable for addressing the fact that not all the defect mechanisms and behaviors are known in advance, and not all of them can be translated into fault models. Therefore, it is advantageous to ensure that the test set covers unexpected defects by exhaustive testing of gates or subcircuits.

Similar to a cell-aware fault, a gate-exhaustive fault is defined by an input pattern of a gate, and has the same

This work was supported in part by NSF grant CCF-1714147.

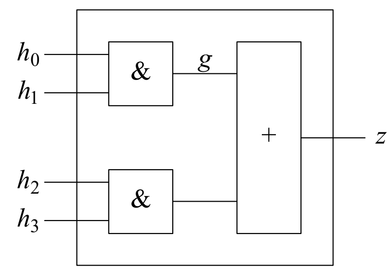

Fig. 1. Subcircuit

two requirements for a test. However, in a gate-exhaustive approach there is no attempt to select input patterns that are relevant to specific defects. Instead, all the input patterns of the gate are used for defining faults. A shortcoming of a gate-exhaustive approach is that it becomes infeasible when gates have large numbers of inputs.

The region-exhaustive approach from [14] is an extension of the gate-exhaustive approach where gates are replaced by regions, which are subcircuits that consist of several gates. An example of a region that consists of three gates is shown in Figure 1. The region-exhaustive fault model associates a fault with every input pattern of the region. The advantage of the region-exhaustive fault model over the gate-exhaustive fault model is that its test set exercises the region more thoroughly than a test set for gate-exhaustive faults. Regions are selected in [14] such that the total number of region-exhaustive faults does not exceed the number of gate-exhaustive faults.

Subcircuits are also considered in this paper for defining faults whose tests exercise the circuit more thoroughly than tests for gate-exhaustive faults. For simplicity of discussion, the faults are referred to as gate-exhaustive even though they are based on subcircuits. The key differences between the regions used in [14] and the subcircuits used in this paper are related to the motivation for using gate-exhaustive faults. In [14] the goal is to provide a better coverage for the entire circuit uniformly. In this paper, a gate-exhaustive approach is used selectively for providing alternate coverage for sites where coverage is missing because of the presence of undetectable target faults from other fault models. Undetectable faults exist because of logic redundancy. Instead of using all the input patterns of every gate, the approach described in this paper uses some of the input patterns of gates that include undetectable target faults in order to achieve a coverage goal for these faults based on the following rationale [15]-[16].

When a target fault is undetectable, because of logic redundancy, a test that would have covered its site is missing from the test set. As a result, the test set may not detect defects around this site, even if the defects are detectable. In [15], this issue is addressed by using fault models with orthogonal detection conditions. Let f_0 and f_1 be two faults from different fault models that are associated with the same site. Suppose that f_0 is undetectable. With the existing fault models, in many cases, f_0 being undetectable implies that f_1 is undetectable. With the approach suggested in [15], the detection conditions of f_1 are orthogonal to those of f_0, and f_1 can be detectable even if f_0 is not. Specifically in [15], a bridging fault model is defined whose detection conditions require two-cycle tests, and are orthogonal to those of transition faults. This allows the bridging faults to cover sites of undetectable transition faults. In [16], what are called optimistic unspecified transition faults are used for addressing the presence of undetectable standard single-cycle transition faults under multicycle tests.

The advantage of a gate-exhaustive approach in this context is that it is comprehensive in considering the alternate faults that may be used for covering sites of undetectable target faults. Specifically, if a target fault f_0 is undetectable, it is not necessary to search for a detectable fault f_1 from a different fault model that is associated with the same site. The gate-exhaustive approach will allow all the input patterns around the site of f_0 to be considered until sufficient coverage for the site of f_0 is achieved. The tests that are added to the test set to detect gate-exhaustive faults will not detect f_0, which is undetectable. However, they are expected to detect detectable defects around the site of f_0 in the same way as a test for f_0 is expected to detect such defects. These defects may not be otherwise detected if the site of f_0 remains uncovered.

Referring to Figure 1, if g is a line with an undetectable target fault, the subcircuit around it provides options for input patterns that can be used for covering the site of the fault. The procedure described in this paper defines overlapping subcircuits in order to increase the number of options further.

To address the fact that the number of gate-exhaustive faults can be excessive, the procedure described in this paper extends the set of gate-exhaustive faults that it considers iteratively until the coverage goal for the sites of undetectable target faults is reached. This allows subcircuits with arbitrary numbers of inputs to be used.

Only undetectable single stuck-at faults are considered in this paper. Other fault models can be considered in a similar way. To cover undetectable delay faults, two-cycle gate input patterns should be considered as gate-exhaustive faults. Undetectable single stuck-at faults exist because of logic redundancy.

The fault g stuck-at a is denoted by g/a. To simplify the discussion, a test that is added to the test set in order to cover the site of an undetectable single stuck-at fault is said to cover the fault (the fault is not detected since it is undetectable).

TABLE I
INPUT PATTERNS

h_0	h_1	h_2	h_3	g	z
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

The paper is organized as follows. Section II discusses the use of gate-exhaustive faults for covering undetectable single stuck-at faults. Section III describes the iterative test generation procedure. Section IV presents experimental results.

II. GATE-EXHAUSTIVE FAULTS

Figure 1 shows an example of a gate G with four inputs, h_0, h_1, h_2 and h_3, an output z, and an internal line g. All the 16 input patterns of the gate are shown in Table I. The corresponding values assigned to line g and gate output z are also shown. An input pattern corresponds to a gate-exhaustive fault whose detection requires that the input pattern be assigned to the inputs of the gate, and its output value would be propagated to an observable output.

Suppose that the target fault $g/0$ is undetectable because of logic redundancy. An input pattern of G (and the corresponding gate-exhaustive fault) is said to cover the site of $g/0$ if it assigns $g = 1$. This value is required for activating the fault. It ensures that the tests, which are used for covering $g/0$, are different from the tests that are used for covering or detecting $g/1$. This is important in case $g/1$ is detectable and does not require additional coverage. The input patterns that assign $g = 1$ are shown in the lower part of Table I. A test for a gate-exhaustive fault from the lower part of Table I is said to cover $g/0$.

Considering the target fault $g/1$, and assuming that it is undetectable because of logic redundancy, an input pattern of G is said to cover the fault if it assigns $g = 0$. The input patterns that assign $g = 0$ in Figure 1 are shown in the upper part of Table I. A test for one of the gate-exhaustive faults in the upper part of Table I is said to cover the target fault $g/1$.

In general, a gate G has inputs h_0, h_1, ..., h_{n-1} and output z. An input pattern of G is denoted by $v = v_0 v_1 ... v_{n-1}$, where v_i is the value of h_i, for $0 \leq i < n$.

The gate includes a set of lines that is also denoted by G. The inputs and output of the gate are included in G. Logic simulation of G under an input pattern v yields a value $v(g)$ for every line $g \in G$.

A target fault g/a is said to be included in G if $g \in G$. If g/a is undetectable, g/a is said to be covered by every input pattern v of G such that $v(g) = \overline{a}$. The requirement $v(g) = \overline{a}$ exists for a test that detects g/a, and it is made a requirement for covering g/a if it is undetectable. This ensures that different tests are used for g/a and $g/\overline{a}$. If g/a is covered by v, a test for the gate-exhaustive fault defined by v is said to cover the fault g/a.

For a constant N_C, the coverage goal for an undetectable target fault g/a is to detect N_C gate-exhaustive faults that cover g/a. During fault simulation or test generation, the actual number of detected gate-exhaustive faults that cover g/a is denoted by $n_c(g/a)$.

An undetectable target fault g/a cannot be covered if it is not possible to obtain the value $\overline{a}$ on g. In this case, $n_c(g/a) = 0$ will be obtained.

III. Test Generation Procedure

The iterative test generation procedure for gate-exhaustive faults is described in this section.

A. Overview

The set of target faults (single stuck-at faults in this paper) is denoted by F_{targ}. The procedure accepts a test set T for the set F_{targ}. Fault simulation with fault dropping of F_{targ} under T yields the set of undetected target faults U_{targ}. If the test generation procedure for F_{targ} is complete, and run to completion, U_{targ} contains only undetectable faults that exist because of logic redundancy. In addition, aborted faults, and other types of undetected faults, if any exist, are also included in U_{targ}, and treated in the same way as undetectable faults.

The procedure also accepts a partition of the circuit into subcircuits that are denoted by G_0, G_1, ..., G_{N-1}. These subcircuits are used for defining gate-exhaustive faults.

The procedure maintains a set of gate-exhaustive faults that is denoted by F_{gexh}. An entry of F_{gexh} is an input pattern $v_{j,k}$ of a gate G_j. Initially, $F_{gexh} = \emptyset$. The procedure proceeds iteratively as illustrated by Figure 2. The dashed boxes represent the following two options for applying test generation.

(1) The first option is to apply test generation in every iteration. In this case, every iteration adds faults to F_{gexh} and tests to T. In this form, gate-exhaustive faults are computed only as necessary for achieving coverage goals.

(2) The second option is to perform test generation only after the iterative part of the procedure defines the set F_{gexh} without performing test generation. In this form, the iterative part of the procedure evaluates the ability of the initial test set to cover undetectable single stuck-at faults before any tests are added to it. Test generation is then performed to improve this ability. This form of the procedure is used for presenting experimental results.

The details of the procedure are discussed next.

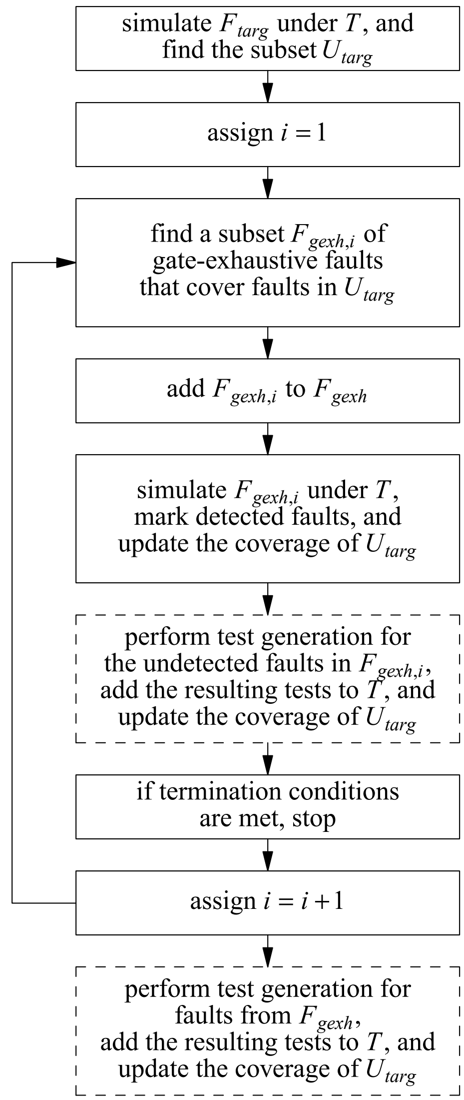

Fig. 2. Test generation procedure

B. Gate-Exhaustive Faults

In iteration $i \geq 1$, the procedure adds a subset of gate-exhaustive faults to F_{gexh}. The subset is denoted by $F_{gexh,i}$, and it is determined as follows.

For every gate G_j, where $0 \leq j < N$, and a constant N_A, the procedure attempts to add N_A input patterns of G_j to $F_{gexh,i}$. For this purpose, the procedure selects N_A input patterns of G_j randomly. Every input pattern $v_{j,k}$ that is selected for G_j is processed as follows.

If $v_{j,k}$ is already included in F_{gexh}, $v_{j,k}$ is discarded without being considered further.

Next, the procedure assigns the values from $v_{j,k}$ to the inputs of G_j, and finds the implications of these values. A conflict may occur if the inputs are not independent of each other. If a conflict occurs, $v_{j,k}$ is discarded.

If a conflict does not occur, the implications yield a value for every line of G_j, including its output, z_j. Let the value of

z_j be $v(z_j)$. If the fault $z_j/\overline{v(z_j)}$ is undetectable, the procedure will not be able to detect the gate-exhaustive fault associated with $v_{j,k}$. In this case, $v_{j,k}$ is discarded.

Next, the procedure considers every fault $g/a \in U_{targ}$ such that $n_c(g/a) < N_C$. The procedure checks whether $g \in G_j$, and $v(g) = \overline{a}$. These are the conditions under which a test for $v_{j,k}$ covers g/a. If the conditions are satisfied, the fault g/a is stored in a set that is denoted by $C_{j,k}$. If the gate-exhaustive fault $v_{j,k}$ is detected later on, the coverage of the faults in $C_{j,k}$ will increase.

If $C_{j,k} = \emptyset$, the detection of $v_{j,k}$ will not increase the coverage of any fault from U_{targ}. In this case, $v_{j,k}$ is discarded without being considered further.

If $v_{j,k}$ passes all the checks without being discarded, it is added to $F_{gexh,i}$.

After considering N_A input patterns for every gate, the procedure adds $F_{gexh,i}$ to F_{gexh}. It then performs fault simulation for the faults in $F_{gexh,i}$, and marks which faults are detected.

The use of $N_A > 1$ takes into consideration that gate-exhaustive faults may be discarded, or remain undetected after fault simulation and test generation. A large enough value of N_A ensures that significant numbers of faults are added to make the iteration useful in increasing the coverage of the sites of undetectable target faults.

C. Iterative Test Generation and Termination Conditions

When test generation is performed iteratively, the target faults for iteration $i \geq 1$ are the faults in $F_{gexh,i}$ that are not already detected by T.

The goal of test generation for a fault $v_{j,k} \in F_{gexh,i}$ is to assign the input values specified by $v_{j,k}$, and propagate a fault effect from the output of G_j to a primary output. The test generation procedure used for the implementation in this paper starts from a test that detects the output fault of G_j, and modifies the test to assign its input values. Other test generation procedures can be used instead.

If a test t is generated for a gate-exhaustive fault $v_{j,k} \in F_{gexh,i}$, fault simulation is carried out for the undetected faults in $F_{gexh,i}$ under t.

As gate-exhaustive faults from $F_{gexh,i}$ are detected, the coverage of single stuck-at faults from U_{targ} is updated as follows. For every fault $v_{j,k} \in F_{gexh,i}$ that is detected by T, and every fault $g/a \in C_{j,k}$, if $n_c(g/a) < N_C$, $n_c(g/a)$ is incremented by one.

The procedure terminates if $n_c(g/a) = N_C$ for every undetectable fault $g/a \in U_{targ}$. Otherwise, the procedure terminates after a constant number of consecutive iterations where the coverage does not increase for any fault in U_{targ}. The constant is denoted by N_I.

D. Evaluating the Initial Test Set

To evaluate the ability of the initial test set to cover the faults from U_{targ}, test generation is performed only after the iterative part of the procedure terminates without performing test generation. In this case, test generation considers the faults in F_{gexh} one by one. A fault $v_{j,k} \in F_{gexh}$ is considered as follows.

If $v_{j,k}$ is already detected, it is not considered further.

Next, the procedure checks the coverage of the faults in $C_{j,k}$. If every fault $g/a \in C_{j,k}$ has $n_c(g/a) = N_C$, $v_{j,k}$ is not considered further.

Test generation is carried out for $v_{j,k}$ only if it is not already detected, and there is at least one fault $g/a \in C_{j,k}$ with $n_c(g/a) < N_C$. If a test t is generated for $v_{j,k}$, fault simulation is carried out for F_{gexh} under t, and the coverage of undetectable single stuck-at faults is updated.

IV. Experimental Results

The procedure outlined in Figure 2 is applied to benchmark circuits with undetectable single stuck-at faults as follows.

Two-level subcircuits are obtained by tracing the circuit backward starting from every gate output. This results in overlapping subcircuits, with more opportunities to cover undetectable single stuck-at faults.

The coverage target for undetectable single stuck-at faults is $N_C = 8$. In every iteration, $N_A = 8$ gate-exhaustive faults are considered based on every gate. The two parameters match such that a single iteration will be sufficient if $N_A = 8$ gate-exhaustive faults are added to F_{gexh} based on every gate, and all the faults are detected. In effect, fewer faults are typically added, and fewer faults are detected in every iteration.

If the coverage goals are not reached for all the undetectable target faults, the procedure terminates after $N_I = 128$ consecutive iterations where the coverage of undetectable single stuck-at faults is not increased.

Test generation is applied after the iterative derivation and simulation of gate-exhaustive faults in order to evaluate the initial test set.

Two compact test sets for single stuck-at faults are used as initial test sets, a one-detection test set, and a ten-detection test set. Both test sets detect all the detectable faults. Undetectable faults exist because of logic redundancy. The ten-detection test set is interesting for the following reason.

In a ten-detection test set, every detectable single stuck-at fault is detected ten times, by ten different tests. The increased number of detections improves the ability of the test set to detect defects around the sites of detectable target faults [17]-[18]. If this is sufficient for covering the sites of undetectable target faults, the procedure from Figure 2 will not add any new tests to the test set. The expectation is that the use of a ten-detection test set will enhance the coverage of undetectable target faults, but the procedure from Figure 2 will be able to provide additional coverage.

To measure the quality of the test set T produced by the procedure from Figure 2, 16-detection fault simulation of T is carried out for single stuck-at faults. Let $n_d(g/a)$ be the number of tests in T that detect a detectable fault g/a. With 16-detection fault simulation, $n_d(g/a) \leq 16$ is obtained. The average value of $n_d(g/a)$ considering the detectable target faults is used for assessing the quality of the test set.

TABLE II
EXPERIMENTAL RESULTS (ONE-DETECTION TEST SET)

							s.a. faults			gate-exh faults			undet	
circuit	pi	ts	sim	tg	tests	incr	cov	16det	bridg	tot	inp	cov	cov	ntime
systemcaes	928	1	27	-	121	1.000	99.997	11.305	91.491	18	6	33.333	6.000	4.50
spi	274	1	3	-	406	1.000	99.992	10.559	82.222	15	6	60.000	8.000	1.12
pci_spoci_ctrl	83	1	141	-	146	1.000	99.968	9.847	79.236	135	9	5.926	8.000	40.82
s526	24	1	2	-	51	1.000	99.905	7.504	78.342	6	3	50.000	3.000	2.14
b04	78	1	14	-	44	1.000	99.869	9.441	84.623	45	7	13.333	5.667	7.12
b04	78	1	-	11	47	1.068	99.869	9.671	84.689	60	7	16.667	8.000	65.56
b07	53	1	4	-	52	1.000	99.845	8.933	80.995	17	4	47.059	2.667	2.46
b07	53	1	-	7	53	1.019	99.845	9.037	81.014	17	4	52.941	3.000	48.54
s38417	1664	1	117	-	103	1.000	99.680	12.429	93.698	5714	10	11.481	3.976	29.93
s38417	1664	1	-	3918	156	1.515	99.680	13.427	93.810	5762	10	13.763	4.494	159.73
tv80	372	1	971	-	489	1.000	99.380	11.064	83.734	12789	16	3.292	5.419	56.12
tv80	372	1	-	4842	751	1.536	99.529	11.889	84.116	13445	16	6.828	6.029	384.36
b11	38	1	58	-	59	1.000	99.290	8.009	76.108	354	9	9.887	2.692	20.54
b11	38	1	-	360	68	1.153	99.290	8.451	76.300	372	9	11.828	3.385	85.92
s1423	91	1	40	-	38	1.000	99.086	8.465	85.594	148	6	35.135	2.962	34.24
s1423	91	1	-	93	41	1.079	99.086	8.770	85.678	148	6	37.162	3.115	132.46
s13207	700	1	80	-	238	1.000	98.869	12.435	88.667	1793	9	29.002	3.752	10.39
s13207	700	1	-	1677	305	1.282	98.869	12.990	88.913	1799	9	34.186	4.104	124.21
s5378	214	1	26	-	111	1.000	98.867	11.785	90.099	466	7	36.695	2.958	6.93
s5378	214	1	-	440	117	1.054	98.867	11.974	90.193	467	7	37.901	3.008	48.67
b15	483	1	251	-	393	1.000	98.540	10.896	79.048	34575	16	4.746	3.731	55.69
b15	483	1	-	15184	418	1.064	98.540	11.005	79.120	43923	16	3.816	3.789	221.43
s15850	611	1	391	-	118	1.000	97.511	12.132	90.756	8680	13	13.687	2.537	48.52
s15850	611	1	-	5876	302	2.559	97.511	13.853	91.416	9014	13	16.130	2.946	437.98
b05	36	1	70	-	61	1.000	96.774	8.868	83.733	4257	12	4.933	3.552	29.05
b05	36	1	-	4452	83	1.361	96.774	10.182	84.544	6170	12	3.825	3.917	184.16
s38584	1464	1	782	-	142	1.000	95.567	11.144	85.048	11738	11	21.154	1.742	109.42
s38584	1464	1	-	7208	227	1.599	95.567	12.840	86.094	11776	11	22.113	1.803	260.10
b14	280	1	224	-	332	1.000	95.326	10.573	82.923	12922	10	9.975	2.239	25.71
b14	280	1	-	12767	656	1.976	95.832	11.395	83.533	12925	10	13.880	2.560	580.55
s9234	247	1	286	-	143	1.000	93.946	10.240	85.957	2340	10	22.778	1.554	36.63
s9234	247	1	-	2296	214	1.497	93.946	11.255	86.251	2341	10	26.826	1.703	165.65
s35932	1763	1	7	-	20	1.000	89.781	4.807	82.320	6143	3	34.983	0.434	13.86
s35932	1763	1	-	6123	79	3.950	89.781	12.872	82.394	6144	3	36.117	0.447	2268.70

A bridging fault coverage is used as an additional quality metric. Since bridging faults are not targeted by the procedure from Figure 2, they can be used for representing unmodeled faults as well as defects. A set B of four-way non-feedback bridging faults is selected such that, for every line g, and every value $a \in \{0, 1\}$, eight lines are selected randomly as the line h that dominates g when $h = a$ is assigned, and eight bridging faults are included in B.

The results are shown in Tables II and III as follows. In Table II, the initial test set is the one-detection test set, and in Table III, the ten-detection test set.

The first row for every test set shows the results of the iterative part of the procedure from Figure 2 without test generation. The results are shown after the last iteration that detects new gate-exhaustive faults by adding new faults and performing fault simulation. The procedure performs additional iterations where it adds and simulates gate-exhaustive faults but cannot detect them using existing tests. As a result, the fault coverage of gate-exhaustive faults decreases, but the same number of gate-exhaustive faults are detected.

The second row shows the results of the procedure from Figure 2 after test generation. The second row is omitted if test generation does not produce new tests.

After the circuit name, column pi shows the number of primary inputs. Column ts has a one when the initial test set is the one-detection test set, and a ten when the initial test set is the ten-detection test set. Column sim shows the index of the iteration of the procedure from Figure 2. Column tg shows the index of the last gate-exhaustive fault for which the procedure from Figure 2 generates a new test.

Column $tests$ shows the number of tests in T. With an initial test set T_{init}, column $incr$ shows the ratio $|T|/|T_{init}|$.

Column $s.a.\ faults$ subcolumn cov shows the fault coverage for single stuck-at faults. Subcolumn $16det$ shows the average number of times a detectable single stuck-at fault is detected when 16-detection fault simulation is carried out. Column $bridg$ shows the bridging fault coverage.

Column $gate - exh\ faults$ subcolumn tot shows the total number of gate-exhaustive faults in F_{gexh}. Subcolumn inp shows the maximum number of inputs for a gate that contributes gate-exhaustive faults to F_{gexh}. Subcolumn cov shows the fault coverage obtained for gate-exhaustive faults.

TABLE III
EXPERIMENTAL RESULTS (TEN-DETECTION TEST SET)

circuit	pi	ts	sim	tg	tests	incr	s.a. faults cov	s.a. faults 16det	bridg	gate-exh faults tot	gate-exh faults inp	gate-exh faults cov	undet cov	ntime
systemcaes	928	10	181	-	1115	1.000	99.819	15.666	96.662	8587	15	9.223	5.181	10.79
systemcaes	928	10	-	6914	1156	1.037	99.862	15.676	96.686	10788	15	8.083	5.000	90.29
spi	274	10	3	-	3695	1.000	99.992	15.165	90.128	15	6	66.667	8.000	1.04
pci_spoci_ctrl	83	10	22	-	1392	1.000	99.968	15.073	85.089	68	9	11.765	8.000	2.89
s526	24	10	2	-	492	1.000	99.905	15.199	87.702	6	3	50.000	3.000	1.31
b04	78	10	23	-	348	1.000	99.869	15.513	91.851	51	7	13.725	5.667	5.00
b04	78	10	-	14	352	1.011	99.869	15.520	91.879	60	7	21.667	8.000	30.17
b07	53	10	4	-	434	1.000	99.845	15.253	87.545	17	4	52.941	3.000	1.55
s38417	1664	10	41	-	784	1.000	99.680	15.691	99.449	3486	10	22.318	4.380	5.53
s38417	1664	10	-	3308	808	1.031	99.680	15.711	99.449	3607	10	23.371	4.624	61.90
tv80	372	10	424	-	4095	1.000	99.390	15.540	91.016	6513	16	8.030	6.159	10.70
tv80	372	10	-	2921	4232	1.033	99.536	15.541	91.078	7062	16	10.564	6.119	140.26
b11	38	10	58	-	572	1.000	99.290	14.878	82.375	346	9	10.405	2.769	7.26
b11	38	10	-	330	578	1.010	99.290	14.911	82.478	364	9	11.538	3.231	28.91
s1423	91	10	26	-	269	1.000	99.086	15.478	94.805	145	6	37.931	3.154	5.57
s13207	700	10	37	-	2341	1.000	98.869	15.319	97.375	1600	9	36.625	3.926	4.62
s13207	700	10	-	1355	2382	1.018	98.869	15.326	97.376	1622	9	38.903	4.104	66.34
s5378	214	10	23	-	992	1.000	98.867	15.556	97.560	433	7	42.956	3.008	3.20
s5378	214	10	-	431	997	1.005	98.867	15.566	97.563	435	7	43.908	3.142	21.99
s15850	611	10	126	-	983	1.000	97.511	15.559	97.728	6608	13	20.248	2.769	10.99
s15850	611	10	-	3617	1077	1.096	97.511	15.611	97.752	7116	13	20.756	2.963	147.79
b05	36	10	444	-	514	1.000	96.774	15.573	93.702	7501	12	3.053	3.802	120.86
b05	36	10	-	6895	536	1.043	96.774	15.586	93.754	7870	12	3.189	4.125	243.16
s38584	1464	10	782	-	1191	1.000	95.567	15.675	92.805	11142	11	23.604	1.794	58.42
s38584	1464	10	-	4789	1210	1.016	95.567	15.689	92.952	11180	11	23.739	1.808	105.88
b14	280	10	224	-	3058	1.000	95.326	15.469	87.916	12793	10	10.670	2.369	15.97
b14	280	10	-	12642	3351	1.096	95.827	15.471	88.059	12796	10	14.278	2.548	360.85
s9234	247	10	172	-	1132	1.000	93.946	15.352	95.677	2255	10	26.386	1.623	7.87
s9234	247	10	-	1987	1151	1.017	93.946	15.375	95.681	2256	10	27.793	1.677	39.71
s35932	1763	10	7	-	129	1.000	89.781	15.186	92.769	6143	3	36.122	0.447	2.94

Subcolumn $undet\ cov$ shows the average coverage of an undetectable single stuck-at fault. This is the average value of $n_c(g/a)$ for a fault $g/a \in U_{targ}$. With $N_C = 8$, $n_c(g/a) \leq 8$.

Column $ntime$ shows runtime information as follows. Let the runtime for single stuck-at fault simulation of the initial test set be ρ_{init}. Let the cumulative runtime for the procedure from Figure 2 be ρ_{Fig2}. The normalized runtime is computed as ρ_{Fig2}/ρ_{init}. The normalized runtime captures the increase in the runtime because of the need to compute gate-exhaustive faults and perform fault simulation and test generation for them.

There are variations in the results for the one-detection and ten-detection test sets because different gate-exhaustive faults are selected, and different tests are generated. Beyond these variations, the following points can be observed.

Among the circuits where the single stuck-at fault coverage exceeds 99.9%, there are cases where test generation for gate-exhaustive faults does not add any new tests to the one-detection test set. There are also cases where the one-detection test set is extended to improve its coverage of undetectable single stuck-at faults, but the ten-detection test set does not require additional tests. In these cases, gate-exhaustive faults can be used for evaluating how well the initial test set covers the sites of the undetectable single stuck-at faults.

With a lower single stuck-at fault coverage, the initial test set is typically increased to a larger extent in order to achieve an improved coverage of the sites of undetectable single stuck-at faults. While a higher increase in the number of tests occurs for the one-detection test set, the ten-detection test set also has to be extended in order to cover the sites of undetectable single stuck-at faults.

The tests that are added to the test set increase the coverage of undetectable single stuck-at faults significantly. They also increase the numbers of detections for detectable single stuck-at faults, and the bridging fault coverage. The increase occurs even when only a small number of tests are added to the test set.

The maximum number of gate inputs varies with the circuit. The iterative selection of input patterns allows subcircuits with large numbers of inputs to be considered without producing excessive numbers of gate-exhaustive faults.

V. CONCLUDING REMARKS

Gate-exhaustive faults address the fact that not all the defect mechanisms and behaviors are known in advance, and not all of them can be translated into fault models. This paper used these properties to obtain extra coverage for sites where coverage is missing because of undetectable single stuck-at faults that result from logic redundancy. To allow subcircuits to be considered as gates while avoiding the need to consider large numbers of faults, the gate-exhaustive approach was applied selectively, using increasing numbers of the input patterns of gates that include undetectable single stuck-at faults. Experimental results demonstrated the extent to which it is possible to cover the sites of undetectable single stuck-at faults using tests for gate-exhaustive faults.

REFERENCES

[1] S. T. Zachariah and S. Chakravarty, "A Scalable and Efficient Methodology to Extract Two Node Bridges from Large Industrial Circuits", in Proc. Intl. Test Conf., 2000, pp. 750-759.

[2] M. Renovell and G. M. Cambon, "Electrical Analysis of Floating-Gate Fault", IEEE Trans. on Computer-Aided Design, Nov. 1992, pp. 1450-1458.

[3] M. O. Simsir, A. Bhoj and N. K. Jha, "Fault Modeling for FinFET Circuits", in Proc. Intl. Symp. on Nanoscale Arch., 2010, pp. 41-46.

[4] 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.

[5] Y. Liu and Q. Xu, "On Modeling Faults in FinFET Logic Circuits", in Proc. Intl. Test Conf., 2012, pp. 1-9.

[6] 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.

[7] F. Hapke and J. Schloeffel, "Introduction to the Defect-oriented Cell-aware Test Methodology for Significant Reduction of DPPM Rates", in Proc. European Test Symp., 2012, pp. 1-6.

[8] F. Yang, S. Chakravarty, A. Gunda, N. Wu and J. Ning, "Silicon Evaluation of Cell-Aware ATPG Tests and Small Delay Tests", in Proc. Asian Test Symp., 2014, pp. 101-106.

[9] A. D. Singh, "Cell Aware and Stuck-open Tests", in Proc. European Test Symp., 2016, pp. 1-6.

[10] E. J. McCluskey, "Quality and Single-stuck Faults", in Proc. Intl. Test Conf., 1993, p. 597.

[11] R. D. Blanton and J. P. Hayes, "Properties of the Input Pattern Fault Model", in Proc. Intl. Conf. on Computer Design, 1997, pp. 372-380.

[12] K. Y. Cho, S. Mitra and E. J. McCluskey, "Gate Exhaustive Testing", in Proc. IEEE Intl. Test Conf., 2005, Paper 31.3, pp. 1-7.

[13] R. Guo, S. Mitra, E. Amyeen, J. Lee, S. Sivaraj and S. Venkataraman, "Evaluation of Test Metrics: Stuck-at, Bridge Coverage Estimate and Gate Exhaustive", in Proc. VLSI Test Symp., 2006, pp. 66-71.

[14] A. Jas, S. Natarajan and S. Patil, "The Region-Exhaustive Fault Model", in Proc. Asian Test Symp., 2007, pp. 13-18.

[15] I. Pomeranz, "A Bridging Fault Model for Line Coverage in the Presence of Undetected Transition Faults", in Proc. Design Automation and Test in Europe Conf., 2017.

[16] I. Pomeranz, "Covering Undetected Transition Fault Sites with Optimistic Unspecified Transition Faults under Multicycle Tests", in Proc. European Test Symp., 2018.

[17] B. Benware, C. Schuermyer, N. Tamarapalli, K.-H. Tsai, S. Ranganathan, R. Madge, J. Rajski and P. Krishnamurthy, "Impact of multiple-detect test patterns on product quality", in Proc. Intl. Test Conf., 2003, pp. 1031-1040.

[18] S. Venkataraman, S. Sivaraj, E. Amyeen, S. Lee, A Ojha and R. Guo, "An Experimental Study of n-Detect Scan ATPG Patterns on a Processor", in Proc. VLSI Test Symp., 2004, pp. 23-28.