In Proc. European Test Symp., 2018

Interconnect-Aware Tests to Complement Gate-Exhaustive Tests

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

Srikanth Venkataraman
Intel Corporation
Hillsboro, OR 97124, U.S.A.
E-mail: srikanth.venkataraman@intel.com

***Abstract*—Gate-exhaustive and cell-aware tests are generated based on input patterns of cells in a design. While the tests provide thorough testing of the cells, the interconnects between them are tested only as input and output lines of cells. This paper defines cell-based faults that allow the interconnects to be tested more thoroughly within a uniform framework that only targets input patterns of cells. In contrast to a real cell that is part of the design, a dummy cell is used for defining interconnect-aware faults. Using a gate-level description of the circuit, a dummy cell contains an interconnect, an output gate of the real cell that drives it, and an input gate of the real cell that it drives. Experimental results for benchmark circuits show that many of the interconnect-aware faults are not detected accidentally by gate-exhaustive tests, and that the quality of the test set is improved by targeting interconnect-aware faults. Here, quality is measured by the numbers of detections of single stuck-at faults in a gate-level representation of the circuit.**

I. Introduction

Fault models are defined to capture the behaviors of commonly occurring defects, and provide targets for fault simulation and test generation procedures that ensure a high quality of testing. Examples of fault models include models for bridges [1]-[5], and models for transistor and interconnect opens [6]-[13]. New fault models are introduced to capture new behaviors as technologies evolve [14]-[17].

In cell-aware approaches, input patterns of cells that are effective in exhibiting the presence of defects in the cells are used directly [18]-[21]. A test for such an input pattern assigns the pattern to the inputs of the cell, and propagates the output value of the cell to an observable output. An analysis of the cell yields the input patterns that are effective for detecting defects in the cell. In a gate-exhaustive approach, all the input patterns of a gate or cell are used for testing it [22]-[23]. For ease of discussion, the input patterns for which tests are generated are considered as representing faults. Single-pattern gate-exhaustive faults are considered in this paper, but the discussion applies to cell-aware faults and two-pattern tests as well.

A test set that is generated by a gate-exhaustive (or cell-aware) approach detects faults on the inputs and outputs of the cells. Therefore, it detects faults on the interconnects between cells. For illustration, Figure 1 shows two cells, C_i and C_j. It is assumed that a gate-level description of the cells is available. Only one gate is shown within each cell, G_i in C_i and G_j in C_j. The cells may contain additional logic that is not shown in Figure 1, and the logic may consist of complex gates such that G_i and G_j may not appear as stand-alone gates in the layout. The two cells are connected by line $g_{i,j}$, which is the output of gate G_i in C_i, and an input of gate G_j in C_j. Faults on line $g_{i,j}$ are detected by tests for C_i and C_j.

However, tests for C_i and C_j may not provide a comprehensive coverage of the defects that may occur on $g_{i,j}$. Such defects may be activated only under certain conditions that are related to both C_i and C_j (as well as additional cells that are adjacent to $g_{i,j}$). The tests for C_i do not consider C_j, and the tests for C_j do not consider C_i. Therefore, none of these tests may create the conditions for detecting a defect with a complex dependence on the logic within both cells.

This issue is addressed in this paper by complementing the gate-exhaustive (or cell-aware) approach with an interconnect-aware approach that provides targets for more comprehensive testing of the interconnects. Interconnect-aware faults are defined such that they have the same format as gate-exhaustive faults. This is accomplished by defining a dummy cell for every interconnect, and defining gate-exhaustive faults for the dummy cells. To distinguish them from gate-exhaustive faults that are related to real cells, the faults are referred to as interconnect-aware faults.

Because of their identical format, tests for interconnect-aware faults can be generated together with gate-exhaustive tests by the same test generation procedure. This simplifies the test generation process that does not need to consider multiple fault models. It also contributes to test compaction since all the faults can be targeted simultaneously by dynamic or static test compaction procedures. Specifically, during dynamic test compaction, a single test can be extended to detect both types of faults uniformly.

Interconnect-aware faults are not intended to replace, and are not replaced by, fault models that target interconnects, such as bridges and opens. Interconnect-aware faults are defined based on a dummy cell that includes the interconnect. The dummy cell may not include the complete layout neighborhood of the interconnect, which is considered for the definition of bridges and opens. On the other hand, bridges and opens are limited to the corresponding defect behaviors, while the behavior of an interconnect-aware fault is not limited to a

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

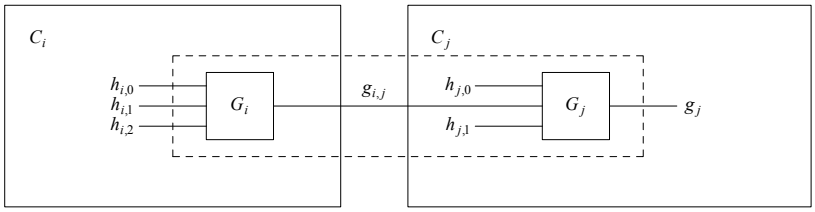

Fig. 1. Interconnect between cells

particular type of defect.

The use of interconnect-aware faults is expected to increase the quality of the test set. To measure this increase, test sets are generated for gate-exhaustive faults, and for gate-exhaustive and interconnect-aware faults. The test sets are compared based on their numbers of detections of single stuck-at faults. A higher number of detections for a single stuck-at fault implies that the test set creates more varied conditions around the site of the fault. This contributes to the detection of defects around the fault site [24]-[25].

The paper is organized as follows. Interconnect-aware faults are defined in Section II. Experimental results are presented in Section III.

II. Interconnect-aware Faults

For the discussion in this section, a cell is referred to as a real cell when it is part of the design. A dummy cell does not exist in the design. A dummy cell is defined around an interconnect, and it is used only for the purpose of defining interconnect-aware faults. Gate-exhaustive faults are defined for real cells. Interconnect-aware faults are defined for dummy cells. Both types of faults have the same format.

Dummy cells include parts of the real cells around an interconnect. For simplicity it is assumed that gate-level representations of real cells are available. The gate-level representation of a real cell is used for identifying gates within the real cell that are important for the definition of interconnect-aware faults. For example, gates that drive or are driven by an interconnect are important for the definition of faults that are related to the interconnect. The gate-level representation is assumed to be accurate in identifying these gates, and their input and output lines. The gates may not appear in the layout as stand-alone gates when the cells consist of complex gates.

Referring to Figure 1, the gate-level representation of the real cell C_i identifies G_i is an output gate that drives the interconnect $g_{i,j}$. The gate G_i has inputs $h_{i,0}$, $h_{i,1}$ and $h_{i,2}$, and output $g_{i,j}$. The gate-level representation of the real cell C_j identifies G_j as an input gate that is driven by the interconnect $g_{i,j}$. The gate G_j has inputs $h_{j,0}$, $h_{j,1}$ and $g_{i,j}$, and output g_j.

Considering C_i, suppose that it contains a complex gate for the function $g_{i,j} = (a_{i,0}a_{i,1} + a_{i,2}a_{i,3} + a_{i,4}a_{i,5})'$. In this case, G_i may be a NOR gate with inputs $h_{i,0} = a_{i,0}a_{i,1}$, $h_{i,1} = a_{i,2}a_{i,3}$ and $h_{i,2} = a_{i,4}a_{i,5}$. Even though $h_{i,0}$, $h_{i,1}$ and $h_{i,2}$ may not exist in the layout, generating tests that assign specific values to them in the gate-level description translates into tests that exercise $g_{i,j}$ in different ways, contributing to the detection of defects.

The following considerations are important when defining a dummy cell around an interconnect.

(1) Since interconnect-aware faults are used for complementing gate-exhaustive faults, it is important to ensure that the number of interconnect-aware faults is similar to the number of gate-exhaustive faults. If the number of interconnect-aware faults is significantly higher than the number of gate-exhaustive faults, they will dominate the test set, and if it is significantly lower, they will not have an impact on the quality of the test set.

(2) The cell C_i may have multiple outputs, and an output of C_i may fan out to multiple cells, or drive multiple input gates of C_j. It is possible to consider multiple interconnects together, or each one separately. Considering each interconnect separately results in smaller dummy cells. In this case, if the output of G_i fans out to several gates in C_j, each fanout line is considered separately.

(3) At a minimum, the dummy cell for a single interconnect $g_{i,j}$ between real cells C_i and C_j should include the gate G_i in C_i that drives $g_{i,j}$, and the gate G_j in C_j that $g_{i,j}$ drives. Without these gates, the dummy cell will not result in new faults. For example, if the dummy cell includes only G_i, input patterns of G_i will be used for defining interconnect-aware faults. However, these input patterns are covered by input patterns of C_i, and gate-exhaustive faults are already defined for G_i as part of C_i.

(4) In addition to driving gates in C_i and driven gates in C_j, it is possible to include in the dummy cell for $g_{i,j}$ other gates that are adjacent to $g_{i,j}$. These gates may be included in C_i, C_j or other real cells. As the number of gates in the dummy cell is increased, its number of inputs is increased as well, and the number of interconnect-aware faults that will be defined for it is increased.

Based on these considerations, and supported by experimental results for benchmark circuits, dummy cells are defined to include single interconnect lines, and the minimum number of gates that support the definition of new faults, as follows.

Let G_i be the output gate of C_i with output line $g_{i,j}$. Let G_j be the input gate of C_j with input line $g_{i,j}$. The dummy cell for $g_{i,j}$ includes G_i and G_j. The inputs to the cell are the inputs of G_i, and the inputs of G_j excluding $g_{i,j}$. The output of the cell is the output line g_j of G_j.

For $g_{i,j}$ in Figure 1, the dummy cell is shown by the dashed box, with inputs $h_{i,0}$, $h_{i,1}$, $h_{i,2}$, $h_{j,0}$ and $h_{j,1}$, and output g_j.

For every pair of real cells C_i and C_j such that C_i drives C_j through a line $g_{i,j}$, the procedure described in this section defines a dummy cell. The dummy cell is used for defining interconnect-aware faults that are related to $g_{i,j}$. All the input patterns of the dummy cell are used for defining faults as in the gate-exhaustive approach.

When using a cell-aware approach, if an accurate layout representation is available for the dummy cell, it can be analyzed similar to a real cell to identify input patterns that are important for the detection of defects.

For a circuit with r real cells, and assuming that every real cell has n inputs, the number of gate-exhaustive faults is $2^n r$. With d interconnects, and assuming that every dummy cell has m inputs, the number of interconnect-aware faults is $2^m d$. Assuming that the fanout of every real cell is q, we have that $d = qr$. The value of m is at least three if G_i and G_j are two-input gates with distinct inputs. The numbers of gate-exhaustive and interconnect-aware faults are similar if the fanout q is low, and the numbers of inputs are such that n is not significantly higher than m. This is the case for benchmark circuits for a large range of values of n.

Considering all the (gate-exhaustive or interconnect-aware) faults that can be defined for a (real or dummy) cell, many of the faults are undetectable. Prior to test generation, undetectable faults are identified in this paper as follows.

Let C_i be a cell with inputs $h_{i,0}$, $h_{i,1}$, ..., h_{i,k_i-1}. Suppose that the inputs are arranged in topological order such that, for $j_0 < j_1$, h_{i,j_0} may have a path to h_{i,j_1}, but h_{i,j_1} does not have a path to h_{i,j_0}. Let the fault under consideration be associated with the input pattern $(h_{i,0}, h_{i,1}, ..., h_{i,k_i-1}) = (v_{i,0}, v_{i,1}, ..., v_{i,k_i-1})$.

The first check for the detectability of the fault performs implications of the input values one after the other in topological order. For $0 < j < k_i$, before performing the implications of $h_{i,j} = v_{i,j}$, the procedure checks whether the implications performed earlier result in $h_{i,j} = \overline{v_{i,j}}$. If this is the case, it is not possible to assign the input pattern to the cell, and the fault is undetectable.

The second check for the detectability of the fault is performed after all the input values are implied successfully. In addition, the implications of a faulty value on the cell output are computed. The procedure then checks whether there is a path from the output of the cell to an observable output through which fault effects can be propagated. This requires a path where the values 0/0 and 1/1 do not appear. Only the values 0/1, 0/x, 1/0, 1/x, x/0, x/1 and x/x are allowed. If no such path exists, the fault is undetectable.

These checks can be performed efficiently for large numbers of faults to identify large numbers of undetectable faults.

III. Experimental Results

This section demonstrates the effects of adding interconnect-aware faults to gate-exhaustive faults for benchmark circuits.

For a parameter denoted by n, a circuit is partitioned into real cells with at most n inputs. Starting from the outputs of the combinational logic as cell outputs, every cell is extended as much as possible such that its number of inputs does not exceed n. The inputs of the cell that are not inputs of the combinational logic are then used as outputs of additional cells.

To cover a broad range of cases, several values of n are considered for every circuit, as follows. Let the maximum number of inputs for a gate in the circuit be g. In three separate experiments, the circuit is partitioned into cells with at most $n = g$, $2g$ and $3g$ inputs. Even with $n = g$, a cell may contain several gates with fewer than g inputs.

For the interconnect-aware approach, a dummy cell is defined for every pair of real cells C_i and C_j such that C_i drives C_j, and the gates G_i (at the output of C_i) and G_j (at the input of C_j) are multi-input gates.

For every input pattern of every (real or dummy) cell, a (gate-exhaustive or interconnect-aware) fault is defined as in a gate-exhaustive approach. Undetectable faults that can be identified without test generation are removed. The set of gate-exhaustive faults is denoted by F_{gate}. The set of interconnect-aware faults is denoted by F_{intr}.

Test generation with test compaction is carried out using a simulation-based procedure that was implemented for the purpose of this study. The procedure modifies tests for single stuck-at faults such that every additional test would detect as many target faults as possible.

Test generation proceeds as follows. First, test generation is carried out targeting only the faults in F_{gate}. Fault simulation is carried out for the faults in F_{intr} to determine which faults are detected accidentally. However, these faults are not targeted directly. The resulting test set is denoted by T_{gate}.

Next, test generation is carried out targeting all the faults in $F_{gate} \cup F_{intr}$. The resulting test set is denoted by T_{intr}.

Since test generation is carried out separately for the two sets of faults, T_{gate} and T_{intr} may not have any gate-exhaustive tests in common.

The run time for generating T_{gate} is denoted by rt_{gate}. The run time for generating T_{intr} is denoted by rt_{intr}. It is typically the case that $rt_{intr} > rt_{gate}$ because more faults need to be targeted by the test generation procedure. The ratio rt_{intr}/rt_{gate} captures the increase in the run time because of the increased number of faults.

To demonstrate that the quality of T_{intr} is improved relative to T_{gate}, both test sets are simulated using a ten-detection fault simulation procedure for single stuck-at faults in the gate-level description of the circuit. In this process, a single stuck-at fault is dropped from further simulation after it is detected by ten different tests.

The results are shown in Tables I and II as follows. There are up to three pairs of rows for every circuit, corresponding to the three values of n. All the three values are considered for several circuits to illustrate the range of results obtained. Additional results are shown using one or two values of n.

The first row of a pair for every circuit and value of n shows the results obtained when test generation targets only the faults in F_{gate}. The second row shows the results obtained when test generation targets all the faults in $F_{gate} \cup F_{intr}$.

TABLE I
EXPERIMENTAL RESULTS (ISCAS-89 AND ITC-99)

		faults			f.c.			10-det	
circuit	n	gate	intr	tests	gate	intr	rtime	ave	%10-det
s1423	4	1682	2650	85	98.454	80.340	1.00	8.028	63.894
s1423	4	1682	2650	142	98.454	91.057	2.64	8.812	76.766
s1423	8	2674	1953	322	83.059	87.609	1.00	9.227	85.017
s1423	8	2674	1953	321	83.059	91.551	1.32	9.324	85.677
s1423	12	3861	1849	690	80.653	90.590	1.00	9.578	90.363
s1423	12	3861	1849	700	80.653	92.104	1.14	9.631	91.089
s5378	4	7419	11275	226	79.579	40.089	1.00	8.587	75.429
s5378	4	7419	11275	293	79.579	44.133	0.72	8.804	79.187
s5378	8	17362	5696	479	54.187	43.504	1.00	9.148	84.597
s5378	8	17362	5696	506	54.187	45.681	0.67	9.168	85.097
s5378	12	41691	3436	1280	32.386	51.921	1.00	9.406	89.138
s5378	12	41691	3436	1290	32.386	54.075	0.50	9.448	90.028
s9234	4	9313	21595	296	81.456	43.936	1.00	7.588	57.067
s9234	4	9313	21595	443	81.456	48.155	0.86	8.073	63.159
s9234	8	21763	15441	953	48.794	49.414	1.00	8.808	72.138
s9234	8	21763	15441	1001	48.794	51.389	0.44	8.933	74.246
s9234	12	71967	14092	4094	23.140	52.725	1.00	9.367	82.157
s9234	12	71967	14092	4178	23.140	54.031	0.50	9.442	83.326
s13207	4	13404	34667	394	83.594	44.226	1.00	7.922	69.812
s13207	4	13404	34667	539	83.594	51.089	2.14	8.154	72.501
s13207	8	35720	21305	865	66.201	42.769	1.00	8.722	76.933
s13207	8	35720	21305	919	66.201	48.115	1.15	8.797	78.105
s13207	12	101528	18380	4203	39.978	46.306	1.00	9.104	86.775
s13207	12	101528	18380	4210	39.978	50.501	0.79	9.169	87.723
s15850	4	14101	26429	355	85.051	49.344	1.00	8.868	77.842
s15850	4	14101	26429	542	85.051	54.641	4.84	9.029	81.058
s15850	8	26566	18570	1647	65.968	54.599	1.00	9.155	83.258
s15850	8	26566	18570	1623	65.968	56.721	2.57	9.229	84.307
s15850	12	68008	17419	4727	51.537	55.790	1.00	9.419	87.633
s15850	12	68008	17419	4683	51.537	58.287	2.93	9.446	88.222
s35932	2	44859	65898	25	82.106	67.515	1.00	5.508	20.100
s35932	2	44859	65898	32	82.106	70.764	1.09	6.503	28.408
s35932	4	33465	32508	35	88.334	70.844	1.00	6.189	27.170
s35932	4	33465	32508	39	88.334	73.877	1.16	6.751	31.130
s35932	6	35193	30780	47	84.724	70.000	1.00	6.856	33.675
s35932	6	35193	30780	45	84.724	72.515	1.16	6.996	34.432
s38417	4	39325	49430	337	94.884	72.250	1.00	9.078	83.502
s38417	4	39325	49430	651	94.884	82.409	2.80	9.287	87.380
s38417	8	69523	28956	1476	84.048	73.588	1.00	9.641	92.976
s38417	8	69523	28956	1487	84.048	79.220	1.81	9.659	93.172
b04	4	2018	3882	118	89.445	61.128	1.00	8.454	70.134
b04	4	2018	3882	221	89.445	70.608	1.94	9.229	82.021
b04	8	4566	2127	359	61.980	79.972	1.00	9.577	87.593
b04	8	4566	2127	392	61.980	87.259	0.93	9.784	93.016
b04	12	18990	1742	1131	38.283	92.078	1.00	9.848	96.880
b04	12	18990	1742	1144	38.283	95.350	0.59	9.877	97.028
b07	5	1947	3886	177	90.550	62.146	1.00	8.285	69.062
b07	5	1947	3886	630	90.550	87.236	4.34	9.385	86.137
b07	10	3181	2774	583	85.822	76.820	1.00	9.337	85.799
b07	10	3181	2774	745	85.822	91.024	1.32	9.631	90.617
b07	15	3189	2750	647	87.394	77.636	1.00	9.451	88.673
b07	15	3189	2750	807	87.394	91.236	1.26	9.701	93.576
b14	5	20472	92808	1243	65.035	39.351	1.00	8.320	68.580
b14	5	20472	92808	4549	65.035	47.458	12.69	9.247	79.020
b14	10	114416	51995	6128	34.364	54.961	1.00	9.587	87.196
b14	10	114416	51995	7221	34.364	59.881	1.77	9.738	89.600
b15	5	48497	220477	944	60.868	37.073	1.00	8.707	76.928
b15	5	48497	220477	2374	60.868	39.885	5.67	9.262	85.370
b20	5	46995	186608	1683	63.047	41.220	1.00	9.022	79.016
b20	5	46995	186608	5966	63.047	48.529	11.13	9.639	87.816

Column n of Tables I and II shows the bound on the number of cell inputs. Column $faults$ shows the number of faults in F_{gate} followed by the number of faults in F_{intr}. Column $tests$ shows the number of tests in T_{gate} or T_{intr}. Column $f.c.$ shows the fault coverage obtained with respect to F_{gate}, followed by the fault coverage obtained with respect to F_{intr}. Column $rtime$ shows the increase in run time because of the use of interconnect-aware faults.

Column $10-det$ shows the results of ten-detection fault simulation, where a single stuck-at fault is dropped from consideration after it is detected ten times. Subcolumn ave shows the average number of detections of a single stuck-at fault. Subcolumn $\%10-det$ shows the percentage of faults that are detected ten times.

TABLE II
EXPERIMENTAL RESULTS (IWLS-05)

		faults			f.c.			10-det	
circuit	n	gate	intr	tests	gate	intr	rtime	ave	%10-det
aes_core	4	93109	328613	253	70.208	42.195	1.00	7.427	58.382
aes_core	4	93109	328613	313	70.208	43.126	3.98	7.750	62.378
aes_core	8	285775	216236	287	26.231	48.362	1.00	7.639	60.939
aes_core	8	285775	216236	320	26.231	49.076	2.22	7.792	63.100
des_area	4	19671	93812	126	67.480	36.056	1.00	6.887	50.529
des_area	4	19671	93812	210	67.480	38.454	3.37	7.939	57.948
des_area	8	73556	59488	324	32.740	45.204	1.00	8.801	74.247
des_area	8	73556	59488	335	32.740	46.016	1.55	8.901	74.439
des_area	12	478371	54533	419	5.321	46.555	1.00	8.963	77.821
des_area	12	478371	54533	441	5.321	47.371	0.98	9.141	78.697
i2c	4	3981	7182	130	81.261	60.930	1.00	8.243	68.507
i2c	4	3981	7182	330	81.261	71.568	2.63	8.973	82.542
i2c	8	7697	4582	323	55.229	72.741	1.00	9.090	84.211
i2c	8	7697	4582	399	55.229	79.376	1.08	9.193	85.837
i2c	12	41749	3498	1168	20.156	85.963	1.00	9.426	89.773
i2c	12	41749	3498	1173	20.156	87.822	0.39	9.500	90.886
pci_spoci_ctrl	4	2914	8090	280	80.817	53.857	1.00	7.303	59.115
pci_spoci_ctrl	4	2914	8090	559	80.817	64.586	3.08	8.013	67.734
pci_spoci_ctrl	8	6885	5400	723	51.184	62.870	1.00	8.231	70.938
pci_spoci_ctrl	8	6885	5400	887	51.184	71.333	1.59	8.456	73.267
pci_spoci_ctrl	12	23598	5118	1353	23.625	65.025	1.00	8.634	77.635
pci_spoci_ctrl	12	23598	5118	1457	23.625	71.551	0.74	8.796	79.499
sasc	4	2087	5243	41	84.859	56.456	1.00	7.189	48.695
sasc	4	2087	5243	77	84.859	64.658	2.46	8.354	68.505
sasc	8	6118	2687	127	44.819	86.491	1.00	9.345	83.096
sasc	8	6118	2687	133	44.819	88.537	0.99	9.426	84.875
sasc	12	24698	2647	504	24.192	86.777	1.00	9.794	96.145
sasc	12	24698	2647	504	24.192	89.120	0.52	9.846	97.212
simple_spi	4	2836	7531	80	82.969	54.110	1.00	7.783	61.019
simple_spi	4	2836	7531	137	82.969	60.643	1.80	8.533	71.585
simple_spi	8	8835	3944	325	44.867	78.195	1.00	9.386	86.816
simple_spi	8	8835	3944	333	44.867	80.502	0.81	9.435	88.244
simple_spi	12	60415	3696	906	18.027	81.710	1.00	9.865	96.763
simple_spi	12	60415	3696	891	18.027	82.468	0.22	9.876	97.144
spi	4	11519	34793	662	78.158	54.137	1.00	8.133	70.211
spi	4	11519	34793	979	78.158	58.765	1.92	8.626	76.812
spi	8	44932	18677	2264	37.437	71.612	1.00	9.316	84.875
spi	8	44932	18677	2266	37.437	73.433	0.75	9.356	85.752
spi	12	160586	16671	3600	17.002	74.339	1.00	9.595	90.737
spi	12	160586	16671	3595	17.002	75.520	0.41	9.618	90.814
systemcaes	4	33400	99060	147	71.051	47.131	1.00	8.324	65.341
systemcaes	4	33400	99060	256	71.051	50.729	3.68	9.390	85.366
systemcaes	8	108636	60991	460	29.919	57.438	1.00	9.513	87.398
systemcaes	8	108636	60991	479	29.919	59.315	1.24	9.629	90.075
systemcaes	12	317007	48957	421	11.642	60.843	1.00	9.469	85.735
systemcaes	12	317007	48957	462	11.642	63.204	0.90	9.683	92.540
systemcdes	4	11920	38092	94	70.973	50.037	1.00	8.090	62.860
systemcdes	4	11920	38092	143	70.973	51.961	4.10	8.806	75.386
systemcdes	8	29984	24011	124	33.575	61.139	1.00	8.600	72.175
systemcdes	8	29984	24011	151	33.575	63.142	1.35	8.909	78.188
systemcdes	12	96446	21369	509	16.927	63.026	1.00	9.832	94.880
systemcdes	12	96446	21369	505	16.927	63.676	0.58	9.834	94.911
tv80	4	28017	104618	939	74.915	47.549	1.00	8.363	71.537
tv80	4	28017	104618	2046	74.915	52.393	5.35	9.087	82.766
tv80	8	86682	73203	2270	36.657	54.629	1.00	9.201	84.819
tv80	8	86682	73203	2852	36.657	57.905	2.79	9.380	87.562
tv80	12	503580	62406	6973	12.345	58.683	1.00	9.537	91.322
tv80	12	503580	62406	7400	12.345	60.952	1.22	9.606	92.269
wb_dma	4	14531	29254	140	71.151	48.691	1.00	8.688	77.201
wb_dma	4	14531	29254	283	71.151	52.960	3.07	9.264	85.369
wb_dma	8	45638	10736	378	36.413	75.782	1.00	9.554	90.964
wb_dma	8	45638	10736	437	36.413	80.663	0.74	9.647	93.185
wb_dma	12	130260	8554	2119	19.190	84.463	1.00	9.773	96.449
wb_dma	12	130260	8554	2122	19.190	88.719	0.42	9.799	96.746

The following points can be seen from Tables I and II. The number of gate-exhaustive faults typically increases when n is increased. Although the number of real cells decreases, the cells have more input patterns. Overall, this increases the number of gate-exhaustive faults for $n \geq 4$.

The number of interconnect-aware faults decreases with n. This is a result of the fact that a smaller number of real cells results in a smaller number of interconnects, but the number of

inputs of a dummy cell remains similar. A smaller number of dummy cells with similar numbers of inputs yields a smaller number of interconnect-aware faults.

These trends are desirable since they imply that when gate-exhaustive faults cover more of the circuit, fewer interconnect-aware faults are defined to complement the test set.

Even with these trends, for all the values of n, the numbers of gate-exhaustive and interconnect-aware faults are such that none of them dominates the test generation process. This is the situation that motivated the use of minimum dummy cells.

Many interconnect-aware faults are not detected accidentally by tests for gate-exhaustive faults. For most of the circuits considered, the fault coverage of interconnect-aware faults increases significantly when these faults are targeted directly. This is true for all the values of n.

The number of tests increases with n because the number of target faults increases. Comparing the increase in the number of tests, $|T_{intr}|/|T_{gate}|$, to the increase in the number of faults, $|F_{gate} \cup F_{intr}|/|F_{gate}|$, the increase in the number of tests is lower. This is because of the use of test compaction that allows both types of faults to be targeted by the same tests. The increase in run time is also typically lower than the increase $|F_{gate} \cup F_{intr}|/|F_{gate}|$ in the number of faults. Both parameters are sometimes reduced when F_{intr} is targeted.

Based on the numbers of detections of single stuck-at faults, the quality of the test set increases in all the cases when interconnect-aware faults are targeted directly. The increase is typically smaller for the higher values of n. In these cases, the quality of the gate-exhaustive test set is higher, and the number of interconnect-aware faults is smaller relative to the number of gate-exhaustive faults. As a result, a smaller increase in quality is to be expected. Nevertheless, an improvement occurs in all the cases when interconnect-aware faults are targeted.

IV. Concluding Remarks

This paper defined cell-based faults that allow the interconnects between cells to be tested more thoroughly as part of a gate-exhaustive (or cell-aware) test generation process. The definition of an interconnect-aware fault is based on the definition of a dummy cell. Assuming that a gate-level description of the real cells is available, the dummy cell contains the interconnect, the output gate of the real cell that drives it, and the input gate of the real cell that it drives. This provides a uniform framework for targeting gate-exhaustive (or cell-aware) and interconnect-aware faults. Experimental results for benchmark circuits showed that many of the interconnect-aware faults are not detected accidentally by tests for gate-exhaustive faults. In addition, the quality of the test set is improved by targeting interconnect-aware faults based on the numbers of detections of single stuck-at 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] V. Krishnaswamy, A. B. Ma, P. Vishakantaiah, "A Study of Bridging Defect Probabilities on a Pentium (TM) 4 CPU", Intl. Test Conf., 2001, pp. 688-695.

[3] I. Polian, S. Kundu, J.-M. Galliere, P. Engelke, M. Renovell and B. Becker, "Resistive Bridge Fault Model Evolution from Conventional to Ultra Deep Submicron Technologies", in Proc. VLSI Test Symp., 2005, pp. 343-348.

[4] S. K. Goel, N. Devta-Prasanna and M. Ward, "Comparing the Effectiveness of Deterministic Bridge Fault and Multiple-Detect Stuck Fault Patterns for Physical Bridge Defects: A Simulation and Silicon Study", in Proc. Intl. Test Conf., 2009, pp. 1-10.

[5] C.-H. Wu, S. J. Lee and K.-J. Lee, "Test and Diagnosis Pattern Generation for Dynamic Bridging Faults and Transition Delay Faults", in Proc. Asia and South Pacific Design Automation Conf., 2016, pp. 755-760.

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

[7] H. Konuk and F. J. Ferguson, "An Unexpected Factor in Testing for CMOS Opens: The Die Surface", in Proc. VLSI Test Symp., 1996, pp. 422-429.

[8] V. H. Champac and A. Zenteno, "Detectability Conditions for Interconnect Open Defects", in Proc. VLSI Test Symp., 2005, pp. 305-311.

[9] D. Arumi, R. Rodriguez-Montanes and J. Figueras, "Defective Behaviours of Resistive Opens in Interconnect Lines", in Proc. European Test Symp., 2005, pp. 28-33.

[10] R. Gomez, A. Giron and V. Champac, "Test of Interconnect Opens Considering Coupling Signals" in Proc. Defect and Fault Tolerant VLSI Systems Symp., 2005, pp. 247-255.

[11] H. Takahashi, Y. Higami, S. Kadoyama, T. Aikyo, Y. Takamatsu, K. Yamazaki, H. Yotsuyanagi and M. Hashizume, "Clues for Modeling and Diagnosing Open Faults with Considering Adjacent Lines", in Proc. Asian Test Symp., 2007, pp. 39-44.

[12] 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.

[13] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim and W.-T. Cheng, "Automatic Test Pattern Generation for Interconnect Open Defects", in Proc. VLSI Test Symp., 2008, pp. 181-186.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] F. Hapke, W. Redemund, A. Glowatz, J. Rajski, M. Reese, M. Hustava, M. Keim, J. Schloeffel and A. Fast, "Cell-Aware Test", IEEE Trans. on Computer-Aided Design, Sept. 2014, Vol. 33, No. 9, pp. 1396-1409.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.