Extraction, Simulation and Test Generation for Interconnect Open Defects Based on Enhanced Aggressor-Victim Model*

Stefan Hillebrecht* Ilia Polian* Piet Engelke* Bernd Becker*
Martin Keim** Wu-Tung Cheng**

*Computer Architecture Group
Albert-Ludwigs-University
Georges-Köhler-Allee 51
D-79110 Freiburg i. Br., Germany
{stspinne|polian|engelke|becker}@informatik.uni-freiburg.de

**Mentor Graphics Corp.
8005 SW Boeckman Road
Wilsonville, OR, 97070, USA
martin_keim@mentor.com
wu-tung_cheng@mentor.com

Abstract

We present a flow to extract, simulate and generate test patterns for interconnect open defects. In contrast to previous work, the accuracy of defect modeling is improved by taking the thresholds of logic gates as well as noise margins into account. Efficient fault simulation is enabled by employing an aggressive fault collapsing strategy and an optimized fault list ordering heuristic which allows to combine the advantages of event-driven simulation with bit parallelism. Test generation complexity is kept in check by generating patterns for technology-independent segment-stuck-at faults first, thus reducing (though not completely eliminating) the need for sophisticated technology-aware test generation. Moreover, a comprehensive untestability analysis identifies new classes of untestable faults. Experimental results demonstrate high efficiency of the new flow, outperforming earlier work by two orders of magnitude.

Keywords: Interconnect opens, Open-via defects, ATPG, fault simulation

1 Introduction

Nanoscale CMOS integrated circuits are vulnerable to interconnect open defects. Specific features of nanoscale manufacturing processes leading to interconnect opens include optical proximity correction (OPC) for lithography [1], vias with high aspect ratios in dual-damascene copper interconnect technology [2] and complex interactions with low-κ interlevel dielectric materials [3]. Despite the substantial literature on modeling interconnect opens [4, 5, 6, 7], only few automatic test pattern generation (ATPG) tools target these defects directly [8, 9].

One difficulty in developing an ATPG tool for interconnect opens is the non-trivial electrical modeling. The logic effects of an interconnect open defect depend on values of *aggressor lines* with which the affected *victim line* is capacitively coupled as well as electrical parameters such as the input voltage thresholds of gates driven by the affected interconnect. Another difficulty is the high computational complexity of test generation and in particular proof of untestability. To activate the fault[1], certain conditions must hold for values on the aggressors. Since there are typically many different ways to satisfy these conditions, proving that one of the ways is inconsistent does not imply that a fault is untestable.

In this paper, we present a flow to simulate interconnect open defects or generate test patterns for them. We employ two refined electrical models: the non-robust enhanced aggressor-victim (NREAV) model which takes input voltage thresholds of the gates driven by the faulty interconnect into account and the robust enhanced aggressor-victim (REAV) model which considers noise margins on top of that. Both NREAV and REAV models are more accurate than models used in previously published interconnect open ATPG tools [8, 9].

In [8], a subset of aggressors was selected and all aggressors in this subset were required to be set to the value opposite to the fault-free value on the victim line. While this

*Parts of this work are supported by the German Research Foundation under grant BE 1176/14-1.

[1] In this paper, we use the word 'defect' to denote a physical defect in a manufactured circuit. The word 'fault' refers to a model of a defect. This includes sophisticated modeling approaches based on realistic defect behavior such as the interconnect open fault models employed here as well as synthetic models such as the stuck-at fault model.

approach leads to a generation of a valid test vector if successful, inability to produce such a vector does not imply that the fault is untestable. Only one fault per interconnect was considered in [8]. In [9], as in [10], the threshold of all gates was implicitly assumed to be $V_{DD}/2$. The ATPG approach in [9] considered all possible faults on an interconnect and took all aggressors into account. However, the run time of the tool from [9] was relatively high.

The ATPG component of the proposed flow is, to the best of our knowledge, the first published test generator for interconnect open defects which takes input voltage thresholds of the gates and noise margins into account. This generally leads to more challenging ATPG instances in which many restrictive conditions must be satisfied simultaneously. We employ algorithmic techniques to increase the performance of our tool: As many of the faults as possible are classified without performing expensive explicit test generation.

This is largely achieved by two techniques. First, several novel structural untestability analysis steps are performed at various steps of the flow taking into account information gathered so far. Second, test generation is first done for a simpler surrogate fault model, the segment-stuck-at model. Only faults not detected by the generated tests and not proven untestable are targeted explicitly.

A number of further speed-up techniques have been implemented, including 32-bit parallel event-driven fault simulation, a fault list ordering to accelerate the fault simulation and an improved fault collapsing. This is the first reported application of fault collapsing in context of interconnect opens. The size of the resulting test set is controlled by a test compaction procedure called twice during the flow.

Extensive experimental data investigate the implications of the new enhanced fault models on the ATPG results. The NREAV fault model is found to be of similar complexity to the model from [10] and leads to similar pattern count. The REAV model results in high-quality test sets which are larger than NREAV test sets but are much smaller than n-detection test sets of inferior coverage. Overall, high fault efficacy is achieved (i.e., most faults are either detected or proven untestable) while a speed-up of two orders of magnitude compared to [9] is realized even though more complex models are used. Effects of process variations affecting both the coupling capacitances and the gate thresholds are studied.

The remainder of the paper is organized as follows. The employed fault model and possible test generation strategies for this model are introduced in Section 2. The optimized parameter extraction, fault simulation and automatic test pattern generation flow is described in Section 3. Experimental results are reported in Section 4. Section 5 concludes the paper and outlines future work.

2 Preliminaries

2.1 Enhanced aggressor-victim fault models

The interconnect open defect models used in this paper are enhancements of the aggressor-victim model from [10] by gate threshold information. As in [9], the defective interconnect is represented by an RC tree. The RC tree consists of RC elements. An RC element represents one (small) piece of interconnect called *segment*. An RC element consists of one resistance (R) and, possibly, a number of capacitances (C) between the output of the resistor and other objects, e.g., other interconnect lines. The topology of an RC tree corresponds to the fanout structure of the corresponding interconnect. The RC tree is generated by layout parameter extraction (PEX) tools.

All open defects on a given piece of the interconnect are mapped to an open fault at the output of the corresponding RC element. Hence, we assume that an interconnect open fault may be located at the output of any RC element in the RC tree representing the interconnect. In contrast, only the source of an interconnect was considered as a possible defect site in [8] and only the vias on the interconnect were considered in [10]. The defect locations considered in this work are a superset of defect locations from [8, 10], because the source of an interconnect as well as all vias have a corresponding RC element within the RC tree.

An open fault at the output of an RC element separates the RC tree into the *stable part* which is driven by the source gate and the *floating part* which is physically disconnected from the source gate. The voltage on the stable part is not affected by the fault. All gates driven by the stable part see the fault-free logical value. In contrast, the voltage on the floating part is determined by *aggressor* interconnects, and the gates driven by the floating part may interpret this voltage as either logic-0 or logic-1, depending on their *input voltage threshold*.

According to the model from [10], aggressors are interconnects (logic signal lines or power or ground rails) with non-zero parasitic coupling capacitance to at least one RC element of the floating part. The number C_0 represents the cumulative coupling capacitance of the floating part with all aggressors assuming the voltage which corresponds to the logical value of 0, i.e., signal lines having logic-0 under the current test pattern, and ground rails. Symmetrically, the number C_1 denotes the cumulative coupling capacitance with signal lines having logic-1 and power supply rails.

In [10], the voltage on the floating part of the line was assumed to be $V_{DD} \cdot C_1/(C_0 + C_1)$. If C_0 exceeded C_1, all gates driven by the floating part were assumed to interpret the logical value of 0, otherwise the logical value of 1. This corresponds to the input voltage threshold of all gates being $V_{DD}/2$. In this work, each logic gate G has two thresholds: $V_{thL}(G)$ and $V_{thH}(G)$ with $V_{thL}(G) \leq V_{thH}(G)$. All volt-

ages below $V_{thL}(G)$ are interpreted as logic-0; all voltages above $V_{thH}(G)$ are interpreted as logic-1. Thresholds are assumed to be identical for gates of the same type but may vary between gates of different types.

We consider two fault models: *robust enhanced aggressor-victim* (REAV) model and *non-robust enhanced aggressor-victim* (NREAV) model. REAV model describes the stable operation of a gate in presence of noise. Under this model, voltages between $V_{thL}(G)$ and $V_{thH}(G)$ may be interpreted as either logic-0 or logic-1. We make the pessimistic assumption that no fault detection is possible in this case, so the interpreted value corresponds to the (fault-free) value on the stable part. Under NREAV model, we assume that gate G has a single threshold equal to $(V_{thL}(G) + V_{thH}(G))/2$. All voltages above (below) this value are assumed to be interpreted as logic-1 (logic-0).

For the sake of simplicity, we use two threshold values for both REAV and NREAV models as well as for the model from [10]. For single-threshold models (NREAV model and model from [10]) we assume that $V_{thL}(G)$ and $V_{thH}(G)$ have identical values. The condition for logic-0 being interpreted by gate G driven by the floating part is

$$\frac{C_1}{C_0 + C_1} < \frac{V_{thL}(G)}{V_{DD}}. \quad (1)$$

The symmetric condition for logic-1 is

$$\frac{C_1}{C_0 + C_1} > \frac{V_{thH}(G)}{V_{DD}}. \quad (2)$$

This equations replace the conditions $C_1/(C_0 + C_1) < 1/2$ and $C_1/(C_0 + C_1) > 1/2$ used in [10]. Setting $V_{thL}(G) = V_{thH}(G) = V_{DD}/2$ corresponds to the original model. Note that the new model captures Byzantine defect behavior, i.e., different gates driven by the floating part interpreting the same voltage level as different logical values.

2.2 Test generation strategies

To generate a test for an interconnect open fault, two approaches exist: forcing logic-0 or logic-1 on the floating part [9]. To detect the fault by *forcing logic-0*, a test vector must satisfy the following requirements: First, it must justify logic-1 on the stable part of the interconnect. Second, the aggressors must be assigned such that at least one gate G driven by the floating part interprets the voltage on the floating part as logic-0 (as computed by Eq. (1)). Finally, there must be a sensitized path from at least one such gate to an output, in order to guarantee observability.

Forcing logic-1 is symmetric: the test vector must justify logic-0 on the stable part and there must exist a gate G which interprets the voltage on the floating part as logic-1 and has a sensitized path to an output. If there is no test vector to detect an open fault by either forcing logic-0 or logic-1, the fault is *untestable*.

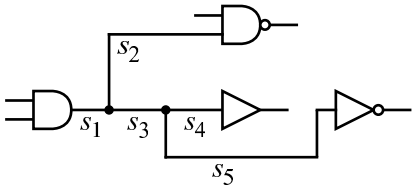

Figure 1: Segments s_1 through s_5 on an interconnect

Certain open faults may lead to oscillation [11, 12]. Whether an oscillating behavior should be counted as detection depends on the characteristics of the test equipment [13]. Hence, we attempt to generate a test vector which does not lead to oscillation whenever possible. We classify a fault which has only test vectors which lead to oscillation as 'detectable by oscillation only'.

2.3 Segment stuck-at faults

A *segment* is defined as the atomic element of an interconnect's topology on gate level. A segment may be located between the interconnect's source and a fanout (stem segment), between a fanout and a gate driven by the interconnect (branch segment), or between two fanouts (internal segment). Figure 1 depicts the partitioning of an interconnect into five segments. A *segment-stuck-at* fault is defined as a single-stuck-at fault which affects only the parts of the interconnects driven by the segment.

One segment corresponds to one or multiple RC elements in the RC tree. For instance, if an aggressor line is coupled with segment s_2 of the interconnect in Figure 1, a PEX tool will generally divide segment s_2 into two parts (one affected and one not affected by the coupling) and extract two RC elements each corresponding to one of these parts.

Segment-stuck-at faults are related to interconnect open faults. Segment-stuck-at faults are associated with segments while interconnect open faults are associated with RC elements. In the example above, the number of segment-stuck-at-fault locations on segment s_2 is 1 (consequently, there are two faults: segment-stuck-at-0 and segment-stuck-at-1). The number of interconnect open fault locations is 2 because segment s_2 corresponds to two RC elements.

Detection conditions for a segment-stuck-at fault are a subset of the detection conditions for an interconnect open fault on a corresponding RC elements. Let s be a segment and let r be any one of the RC elements corresponding to segment s. To detect the segment-stuck-at-1 (0) fault at segment s, all conditions to detect an interconnect open fault at r by forcing logic-0 (logic-1) except the constraints on the aggressors, i.e., Eqs. (1), (2), must be fulfilled.

1. Layout extraction
   (a) RC tree generation
   (b) Buffer insertion in gate-level net-list
   (c) Fault list generation
2. Preprocessing
   (a) Elimination of structurally untestable faults
   (b) Structural equivalent fault collapsing
   (c) Fault list ordering
3. Simulation of a given test set with fault dropping (if applicable)
4. Test generation for segment-stuck-at faults
   (a) Deterministic test generation with fault dropping
   (b) Test set compaction
5. Untestability analysis
   (a) Satisfiability check for Eqs. (1), (2)
   (b) Identification of a dominating aggressor in a local loop
   (c) Local implication analysis
6. Explicit test generation for interconnect open faults
7. Test set compaction
8. Untestability analysis of aborted faults

Figure 2: Extraction, simulation and test generation flow

3 Extraction, Simulation and ATPG

The flow to extract all required information from circuit layout, fault simulate a given set of test vectors (if present) and generate patterns for the faults not covered by that test set is summarized in Figure 2. Later, single steps of the flow are described in more detail.

3.1 Layout extraction

The layout extraction flow based on a parameter extraction (PEX) tool was described in [12]. The extraction procedure produces one RC tree for each interconnect in question. Furthermore, buffers are inserted in the gate-level net-list of the circuit to reflect the topology of the interconnect [12]. Finally, the fault list, which consists of all RC elements in all RC trees, is generated. It is possible to restrict the fault list to, e.g., only one fault per interconnect, or only RC elements corresponding to vias.

We optimized the flow in [12] to reduce the number of intermediate files in use. All required information is now generated directly from the Spice file produced by the PEX tool. Furthermore, more data is loaded prior to simulation or test generation and held in memory, to reduce time consumed by file I/O operations.

Table 1: Results of fault collapsing and untestable fault identification

Circuit	Untestable	Remaining faults		
	(no aggr)	uncollapsed	collapsed	reduction [%]
c0017	68	122	45	63.1
c0095	180	833	448	46.2
c0499	1098	4873	2498	48.7
c0880	1953	8042	4152	48.4
c1355	3460	11895	5641	52.6
c1908	4466	16814	8051	52.1
c2670	5586	29953	17786	40.6
c3540	7732	33408	17990	46.2
c6288	13842	48149	21956	54.4
c7552	17030	75340	41447	45.0

Table 2: Run time and number of simulation runs for ISCAS 85 circuit c7552

Number	Run time	Patterns	Good	Faulty circuit simulations		
patterns	[s]	per s	simulations	no opt.	opt.	skipped [%]
10	1.30	8	1	72.2k	19.9k	72.4
100	2.13	47	4	111k	38.0k	65.9
1k	4.74	211	32	335k	99.0k	70.5
10k	24.44	409	313	2.17M	611k	71.9
100k	186.30	537	3125	16.7M	4.90M	70.7
1M	1569.60	637	31250	139M	43.1M	69.1

3.2 Preprocessing

The fault list generated by the layout extraction procedure is collapsed using two techniques: first, faults with no aggressor are classified as untestable, because there is no option to detect them by forcing either logic-0 or logic-1. Second, faults corresponding to RC elements with no aggressors between them are merged (collapsed) because such faults are equivalent (have identical test set). The techniques are simple heuristics which do not identify all existing untestable or equivalent faults, yet they help to quickly reduce the size of the fault list.

Table 1 contains the number of faults identified as untestable by pre-processing (column 2) and the efficiency of collapsing. The number of faults before collapsing (including the untestable faults) is reported in column 3, the number of faults after collapsing is given in column 4, and the reduction in per cent can be found in column 5.

Finally, the fault list is re-ordered such that faults belonging to the same interconnect are sorted in topological order: If faults f_1 and f_2 are associated with RC elements r_1 and r_2, respectively, and RC element r_1 drives RC element r_2, fault f_1 must show up in the list before fault f_2. For instance, all faults associated with RC elements corresponding to segment s_1 in Figure 1 must appear before any fault associated with an RC element corresponding to segment s_3. This is required to facilitate event-driven fault simulation.

3.3 Fault simulation

If the flow is used in fault simulation mode, a test set to be simulated is given. In the test generation mode, a test set can also be provided. This test set is fault simulated, the detected

interconnect open faults are dropped and test generation is done for the remaining faults only. For instance, there may be a test set which is going to be applied anyway (e.g., a stuck-at test set), and only top-up patterns to improve the coverage of interconnect open faults may be desired. If no test set is provided, test generation is performed for all faults left after the preprocessing.

In contrast to [9], we implemented a high-performance 32-bit pattern-parallel simulation engine. Detected faults are dropped. We employ the following technique (related to event-driven simulation) to reduce the number of simulations. The faults on the same interconnect are sorted in topological order in the preprocessing stage. If the values interpreted by all gates driven by the interconnect do not differ from these values for the last simulated fault, the simulation can be skipped and the detection status of the last fault can be assigned to the current fault.

This technique is facilitated by the topological sorting of the faults because opens at RC elements located close to each other are likely to result in similar voltages on the floating part of the interconnect and thus identical interpretation by the driven gates. Note that due to pattern-parallel implementation the requirement of identical interpretation must hold for all 32 patterns in the packet under simulation.

To identify oscillation, a second simulation pass is performed if the fault is propagated through one of the aggressors. The value on the victim line is recalculated using Eqs. (1), (2), and if the new value does not match the old value, a dedicated logic value OSC is assigned to the victim line and propagated. The second pass is skipped if the logic values on the affected aggressors did not change.

Table 2 shows the simulation time and the efficiency of the introduced speed-up techniques for 10 through one million random patterns applied to 36137 collapsed faults not identified as untestable by preprocessing in ISCAS 85 circuit c7552. Columns 2 and 3 contain the absolute simulation time and the number of patterns simulated per second, respectively. The number of performed good-circuit simulations (for packets of 32 patterns) follows in column 4. Columns 5 and 6 report the number of faulty-circuit simulation to be performed in the absence of skipping techniques and the number of faulty-circuit simulations which had actually been performed. The final column contains the ratio of skipped simulations.

One can see that the simulator is able to handle large numbers of patterns. About two thirds of simulation runs can be skipped thanks to the implemented techniques.

3.4 Segment-stuck-at test generation

Instead of targeting the interconnect open faults left after preprocessing and fault simulation directly by an interconnect open test generation routine (such as done in [9]), we first generate test patterns for the technology-independent segment-stuck-at model introduced in Section 2.3.

Layout extraction (Section 3.1) modifies the gate-level net-list of the considered circuit by adding buffers to represent the actual interconnect topology. Segment-stuck-at faults in the original circuit thus correspond to single-stuck-at faults in the modified circuit. To generate the test set for the segment-stuck-at faults, stuck-at test generation with fault dropping is performed for the modified circuit using a commercial test pattern generator.

Since the conditions to detect a segment-stuck-at fault are a subset of their counterparts for an interconnect open fault, segment-stuck-at test sets are likely (though not guaranteed) to detect many interconnect open faults. Test generation for segment-stuck-at faults runs much faster than explicit test generation for interconnect open faults described in Section 3.6 which often requires multiple runs of the test generation tool for one fault. Hence, test generation (with fault dropping) is done for segment-stuck-at faults and all interconnect open faults detected are dropped.

While test generation for segment-stuck-at faults is technology-independent, fault dropping is performed using the simulator described in Section 3.3, which takes all modeling details into account. Because the resulting test sets may be inadequately large, an aggressive test compaction approach based on the set cover calculation is utilized.

The vector in the test set which detects most interconnect open faults is determined and selected. Then, the vector which detects most of the remaining faults is selected. This is iterated until all faults have been covered by at least one selected vector; all vectors which have not been selected are dropped. We did not encounter any scalability problems with this heuristic; for larger circuits other compaction methods [14, 15] or reverse-order simulation can be used instead.

3.5 Untestability analysis

At this stage, further fast untestability checks are run based on the information gathered so far. Detection of an interconnect open by forcing logic-0 is impossible if either the corresponding segment-stuck-at-1 fault is untestable, or Eq. (1) cannot be satisfied. The latter can occur because some of the aggressors with rather large parasitic cross-capacitance are power rails such that the value of C_0 cannot reach the required value. The conditions for untestability by forcing logic-1 are symmetric. If both options are ruled out, the interconnect open fault is proven untestable.

A further untestability check identifies specific situations when a *dominating* aggressor is driven by the gate which is the sink of the victim line as depicted in Figure 3 (*local loop*). An aggressor is called dominating if its value uniquely determines the value on the floating part of the interconnect (because there are no other aggressors or coupling capacitances

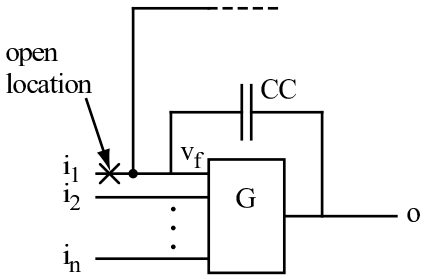

Figure 3: Untestability analysis based on a local loop

of other aggressors are too small).

Suppose that gate G in Figure 3 is an AND gate. To detect the open fault on i_1 by forcing logic-1, the fault-free value on i_1 has to be 0 and the fault-free values on all side inputs i_2 through i_n have to be 1. This implies logic-0 at the output o, thus preventing logic-1 from being forced on i_1 by aggressor o. To detect the fault by forcing logic-0, o must be set to 0. Since i_1 has to be set to 1 to activate the fault, one of the side-inputs must be 0. Hence, the fault effect cannot be propagated through G. The fault may still be detectable if there exists an alternative propagation path, i.e., if the floating part of i_1 is connected to another gate (has a fanout). Then, propagation is possible through that gate.

Based on this analysis, which can also be done for other gate types, a fraction of faults are identified as untestable. If G is an inverting gate, faults may be detectable by oscillation only (within the model assumptions). If alternative propagation paths exist, additional conditions to force propagation through these paths are generated. If these conditions cannot be fulfilled during subsequent test generation, the fault is again classified as untestable or detectable by oscillation only.

Finally, implications of the forced value on the victim are calculated. If these implications result in a value assignment to aggressors which prevents the forcing of the value, the fault is proven untestable. This step is run after identification of a dominating aggressor in a local loop because it is more time-consuming. The calculated implications are stored for subsequent use to prune search space during explicit test generation (Section 3.6).

Preprocessing, fault simulation, test generation for segment-stuck-at faults and untestability analysis have been added to the flow to reduce the number of interconnect open faults for which explicit test generation must be run. In previous work [9], only structural untestability analysis (part of the preprocessing step in this paper) and fault simulation were performed. Table 3 summarizes the numbers of interconnect open faults which must be targeted by explicit test generation (ETG) because they have not been classified by earlier stages of the flow (ETG-targeted faults, ETG-TFs).

Column 2 shows this number for the approach from [9].

Table 3: Number of faults which must be targeted by explicit test generation (ETG-TFs) in [9] and proposed flow

Circuit	ETG-TFs [9] (no initial test set)	Proposed flow			
		With s@ 1-det. test set		No initial set	
		Patterns	ETG-TFs	Patterns	ETG-TFs
c0017	51	6	0	3	0
c0095	647	12	3	12	3
c0499	3737	65	47	38	48
c0880	6053	67	48	32	66
c1355	8403	101	88	52	95
c1908	12341	155	226	91	241
c2670	24367	124	470	59	520
c3540	25670	182	614	110	694
c6288	34298	42	236	32	271
c7552	58304	205	971	122	1025

Columns 3 and 4 give the results for the proposed flow started with 1-detection single-stuck-at patterns as the initial test set. Column 3 contains the number of patterns in the initial test set together with patterns generated for segment-stuck-at faults, column 4 reports the number of faults not covered by that patterns and not classified as untestable. Columns 5 and 6 quote the same numbers if no initial test set is used. It is apparent that the described techniques significantly reduce the number of faults for which explicit test generation is needed.

3.6 Explicit test generation

The interconnect open faults not classified by methods described so far are subject to explicit test generation. The basic test generation procedure described in [9] has been modified with respect to the enhanced models. The aggressors are sorted in the decreasing order of their parasitic capacitances with the floating part. Then, the algorithm tries to generate a test vector which detects the fault by forcing logic-0 or logic-1. A branch-and-bound algorithm assigns logic values to the aggressors. A constrained stuck-at test generation instance is formulated based on the assigned values and test generation is performed using a commercial constrained stuck-at test generation tool. The details of the branch-and-bound algorithm are found in [9].

The obtained test vector is simulated and the validity of Eq. (1) (if the detection strategy was forcing logic-0) or Eq. (2) (if the detection strategy was forcing logic-1) is checked. If the required equation does not hold, additional constraints are derived and a new constrained stuck-at test generation is performed. Note that no gate threshold information was used in [9] and thus a simpler version of Eqs. (1) and (2) was employed. If no pattern could be found, a backtrack is initiated. A fault is untestable if there is no earlier decision which could be taken back.

Some earlier approaches [8] reduce the search complexity by excluding all aggressors with parasitic cross-capacitance

less than a certain limit from consideration. We do not use such search space pruning. If a fault has been classified as untestable by our flow, there is provably no test vector which leads to a detection of the fault according to the fault model used. A fault which could not be attributed to one of the classes 'detected', 'untestable', or 'detectable by oscillation only' is classified as 'aborted'. Aborted faults show up if the constrained-stuck-at test generation results in an abort or if a backtrack limit (15 in our experiments) has been reached.

After the explicit test generation, the combined test set of compacted segment-stuck-at patterns and patterns generated explicitly for the remaining interconnect open faults are again compacted using the same method as in Section 3.4. If an initial test set was provided, it is not modified by compaction because every pattern in it may be required for coverage of other defect types than interconnect opens.

3.7 Untestability analysis of aborted faults

For every fault aborted during explicit test generation, a further untestability check is performed. The check identifies a dominating aggressor a_{dom} which establishes an *inverting loop*, i.e., the fault-free value on a_{dom} is equal to the inverted value on the victim. This scenario obviously implies oscillation and rules out detectability of the considered fault. To find a_{dom}, all aggressors which lie on every path from the victim to an output are identified first. The dominance property is then checked for all these aggressors.

4 Application Results

ISCAS 85 and combinational cores of ISCAS 89 circuits were laid out using a commercial CMOS library and the parasitic coupling capacitances were extracted as outlined in Section 3.1. For each logic gate type, thresholds V_{thL} and V_{thH} were defined as input voltages for which the gate's output voltage equals $0.1 \cdot V_{DD}$ and $0.9 \cdot V_{DD}$, respectively (or vice versa if the gate is inverting). Although the values $0.1 \cdot V_{DD}$ and $0.9 \cdot V_{DD}$ may appear too conservative, they were chosen to demonstrate the performance of the test generation algorithm under tight constraints. The values V_{thL} and V_{thH} have been obtained from that values by analog simulations. For XOR gates, V_{thL} and V_{thH} have been determined for all possible input transitions (input 1 or input 2 switching, and logic-0 or logic-1 at the input which does not switch). The minimal value of V_{thL} and the maximal value of V_{thH} among all simulations were used as V_{thL} and V_{thH}.

Table 4 summarizes the results for all interconnect open faults in a circuit. Columns 2 through 12 report results for the ATPG tool run without an initial test set. Column 2 contains the number of segment-stuck-at patterns after compaction (generated by step 4 of our flow). Column 3 gives the number of patterns produced by the explicit test generation (step 6). Column 4 reports the number of patterns in the final test set after compaction (step 7). The total number of faults and their distribution to classes untestable, aborted and detectable is given next. Faults detectable by oscillation only count as untestable. The fault efficacy (the number of the detected faults divided by the number of faults not proven untestable) is shown in column 9. In column 10, fault efficacy is recalculated where each fault is weighted by the length of the piece of the interconnect corresponding to the fault's RC element. The CPU time of the tool and the real time (which includes the time consumed by the commercial stuck-at ATPG tool) are given in columns 11 and 12.

The proposed ATPG can generate test sets with high fault efficacy (typically over 99% according to both metrics used) in limited time. A large share of produced patterns have been generated for segment-stuck-at faults, thus eliminating many expensive constrained stuck-at ATPG calls. The generated test sets are reasonably compact.

Final five columns of Table 4 report results for the ATPG flow called with a 1-detection single-stuck-at test set as the initial test set. The numbers of single-stuck-at patterns, top-up patterns generated by our flow and the total number of patterns are shown in columns 13 through 15. Column 16 contains the fault efficacy (without weighting) and column 17 gives the real run time (including the invocations of constrained stuck-at ATPG during explicit test generation in Step 6).

The resulting sets are larger compared to those generated with no initial test set because not all patterns in a stuck-at test set are effective for interconnect open faults. It is interesting that the number of top-up patterns is significantly smaller than the number of stuck-at patterns for all but the largest circuits. The fault efficiency is comparable to that achieved with no initial test set (it is not identical due to slight variation in number of aborts). Test generation time is lower because some faults are dropped by simulating the initial test set and do not need to be targeted explicitly.

Table 5 contains the same data for fault lists restricted to open-via defects. The numbers of faults and thus the run times are significantly lower than for the complete open fault lists. The conclusions drawn for general open faults appear to also be valid for open-via faults.

Table 6 investigates the influence of the fault model on the ATPG results. It compares the pattern count and the fault efficacy of stuck-at test sets and interconnect open test sets generated under different assumptions on gate thresholds. We consider three interconnect open fault models: the model from [10], where the threshold of each gate equals $V_{DD}/2$ (this model has been used in [9]), the NREAV model (where, for each gate G, a single threshold equal to $(V_{thL}(G) + V_{thH}(G))/2$ is assumed) and the REAV model (where voltages between $V_{thL}(G)$ and $V_{thH}(G)$ are assumed to correspond to no definite logic value). Our main interest

Table 4: ATPG for complete interconnect open fault lists

Circuit	No initial test set											With initial test set				
	Patterns			Faults				Fault efficacy		Run time [s]		Patterns			Fault	Run time
	segm.	expl.	comp.	total	untest.	abort	det.	no weight	weighted	cpu	real	init.	top-up	total	efficacy	real [s]
c0499	41	2	42	4873	1373	57	3443	98.37	98.81	0.44	21	63	6	69	98.60	20
c0880	41	11	46	8042	2180	16	5846	99.73	99.76	0.57	6	64	12	76	99.62	6
c1355	65	13	70	11895	4250	65	7580	99.15	99.46	1.32	31	95	17	112	98.99	36
c1908	111	22	110	16814	5486	155	11173	98.63	98.81	3.58	39	148	23	171	98.82	39
c2670	78	46	89	29953	6858	165	22930	99.29	99.45	11.21	54	109	48	157	99.35	44
c3540	129	57	158	33408	9556	230	23622	99.04	99.27	18.36	80	166	68	234	99.14	69
c6288	37	13	37	48149	15613	124	32412	99.62	99.73	8.61	27	36	16	52	99.57	20
c7552	150	56	178	75340	20733	259	54348	99.53	99.53	49.32	144	184	66	250	99.55	104
cs00510	50	13	55	5747	1282	35	4430	99.22	97.64	0.48	7	69	14	83	99.89	10
cs00526	44	7	47	5739	1507	26	4206	99.39	99.84	0.39	16	63	4	67	99.46	9
cs00641	40	18	47	8093	2135	57	5901	99.04	99.77	0.74	36	43	16	59	99.18	35
cs00713	38	12	46	8430	2585	26	5819	99.56	99.82	0.61	14	49	15	64	99.52	13
cs00820	93	21	101	9485	1951	50	7484	99.34	99.78	2.55	22	124	18	142	99.34	21
cs00832	93	14	100	9868	2230	60	7578	99.21	98.32	2.48	14	126	15	141	99.24	11
cs00838	115	21	125	10023	2966	72	6985	98.98	98.62	1.38	24	162	21	183	98.97	23
cs00953	71	19	81	11449	2105	42	9302	99.55	99.16	1.70	27	102	22	124	99.59	24
cs01196	117	28	129	13059	2743	51	10265	99.51	99.79	4.01	29	174	33	207	99.53	29
cs01238	118	25	126	13373	2862	64	10447	99.39	99.76	4.65	42	178	30	208	99.40	40
cs01423	56	18	63	16315	3379	45	12891	99.65	99.63	2.05	9	64	20	84	99.70	8
cs01488	110	27	122	17468	3117	101	14250	99.30	99.26	10.09	52	134	34	168	99.34	48
cs01494	96	25	110	17008	2909	71	14028	99.50	99.70	9.13	52	138	28	166	99.56	48
cs09234	196	189	335	105952	26882	659	78411	99.17	99.53	209.75	420	219	212	431	99.22	309
cs13207	251	296	433	192065	29668	809	161588	99.50	99.55	953.69	1504	307	285	592	99.60	807
cs15850	188	255	374	205870	38527	999	166344	99.40	99.58	915.86	1576	197	269	466	99.43	1070
Average	97.0	50.3	126.0	36600.8	8037.4	176.6	28386.8	99.29	99.36	92.2	177.0	125.6	53.8	179.4	99.36	118.5

Table 5: ATPG for open-via fault lists

Circuit	No initial test set									With initial test set				
	Patterns			Faults				Fault	Real run	Patterns			Fault	Run time
	segm.	expl.	comp.	total	untest.	abort	detected	efficacy	time [s]	init.	top-up	total	efficacy	real [s]
c0499	36	3	35	730	103	9	618	98.56	18	63	3	66	98.56	17
c0880	35	9	35	1239	168	5	1066	99.53	4	64	4	68	99.44	5
c1355	50	6	54	1893	487	14	1392	99.00	21	95	7	102	98.80	8
c1908	88	11	91	2697	604	32	2061	98.47	24	148	9	157	98.57	23
c2670	61	26	71	4752	544	37	4171	99.12	31	109	26	135	99.22	29
c3540	113	37	129	5459	861	46	4552	99.00	54	166	42	208	99.07	51
c6288	30	7	34	7943	1611	24	6308	99.62	15	36	9	45	99.64	13
c7552	123	34	134	12224	1878	50	10296	99.52	81	184	35	219	99.56	71
cs00510	42	8	45	924	99	2	823	99.76	4	69	11	80	99.88	4
cs00526	38	6	40	817	119	6	692	99.14	6	63	2	65	99.14	6
cs00641	36	12	40	1208	201	13	994	98.71	33	43	12	55	98.81	33
cs00713	30	9	35	1267	241	6	1020	99.42	13	49	11	60	99.42	12
cs00820	76	14	83	1464	140	10	1314	99.24	9	124	13	137	99.24	9
cs00832	79	11	84	1511	181	5	1325	99.62	10	126	9	135	99.70	9
cs00838	99	14	106	1506	302	9	1195	99.25	13	162	13	175	99.34	13
cs00953	64	5	64	1832	183	7	1642	99.58	23	102	10	112	99.58	22
cs01196	97	14	108	2155	262	16	1877	99.15	17	174	18	192	99.15	16
cs01238	99	15	106	2205	257	17	1931	99.13	37	178	18	196	99.18	37
cs01423	47	14	50	2570	325	10	2235	99.55	5	64	14	78	99.60	10
cs01488	83	16	94	2930	264	17	2649	99.36	40	134	14	148	99.40	38
cs01494	82	16	87	2842	244	10	2588	99.62	42	138	14	152	99.62	39
cs09234	181	124	274	16288	2515	124	13649	99.10	178	219	133	352	99.07	142
cs13207	220	191	348	29260	2087	125	27048	99.54	374	307	183	490	99.57	246
cs15850	173	189	299	31663	3065	174	28424	99.39	478	197	166	363	99.37	351
cs38417	189	708	694	117182	2311	281	114590	99.76	6079	194	536	730	99.80	2684
cs38584	193	586	572	100662	4276	572	95814	99.41	4987	209	462	671	99.41	3198
Average	90.9	80.2	142.8	13662.4	897.2	62.4	12702.9	99.29	484.5	131.4	68.2	199.7	99.31	272.6

Table 6: Fault efficacy and pattern count of different test sets for complete open fault lists

Circuit	Faults	Untest.	Stuck-at 1-det.			Stuck-at 3-det.			Model from [10]			NREAV model			REAV model		
			pat.	det.	flt.eff.	pat.	det.	flt.eff.	pat.	det.	flt.eff.	pat.	det.	flt.eff.	pat.	det.	flt.eff.
c1355	11895	4250	95	7532	98.52	267	7563	98.93	63	7543	98.67	63	7535	98.56	70	7580	99.15
c1908	16814	5486	148	11104	98.02	406	11168	98.59	103	11039	97.45	105	11084	97.85	110	11173	98.63
c2670	29953	6858	109	22677	98.19	251	22838	98.89	78	22697	98.28	81	22770	98.59	89	22930	99.29
c3540	33408	9556	166	23142	97.02	378	23481	98.44	131	23164	97.12	134	23260	97.52	158	23622	99.04
c6288	48149	15613	36	32324	99.35	62	32404	99.59	31	32361	99.46	33	32354	99.44	37	32412	99.62
c7552	75340	20733	184	54051	98.98	444	54264	99.37	150	54075	99.03	158	54139	99.14	178	54348	99.53
cs01196	13059	2743	174	10080	97.71	455	10181	98.69	102	10076	97.67	107	10139	98.28	129	10265	99.51
cs01238	13373	2862	178	10287	97.87	464	10388	98.83	112	10353	98.50	117	10373	98.69	126	10447	99.39
cs01423	16315	3379	64	12767	98.69	149	12846	99.30	48	12728	98.39	50	12742	98.50	63	12891	99.65
cs01488	17468	3117	134	14137	98.51	358	14235	99.19	117	14171	98.75	113	14166	98.71	122	14250	99.30
cs01494	17008	2909	138	13948	98.93	358	14002	99.31	101	14003	99.32	101	14000	99.30	110	14028	99.50
cs09234	105952	26882	219	76093	96.23	532	77512	98.03	236	77269	97.72	237	77239	97.68	335	78411	99.17
cs13207	192065	29668	307	158000	97.29	811	160121	98.60	275	159030	97.93	296	159109	97.98	433	161588	99.50
cs15850	205870	38527	197	163447	97.67	497	164945	98.57	217	163943	97.97	218	163832	97.90	374	166344	99.40
Average	56905	12327	154	43542	98.07	388	43996	98.88	126	43747	98.30	130	43767	98.44	167	44306	99.33

is the performance of test sets generated under simplified assumptions on gate thresholds, compared to stuck-at test sets.

Column 2 of Table 6 contains the numbers of faults. Column 3 gives the number of faults untestable under the REAV model (which is the most accurate model). Columns 4 through 9 show the pattern count ('pat.'), the number of detected faults ('det.') and the fault efficacy ('flt.eff.') of 1-detection and 3-detection stuck-at test sets under the REAV model. Columns 10 through 12 contain these data for the test sets generated assuming the model from [10]. Columns 13 through 15 report these values for the NREAV model. The final three columns quote the performance of the test sets generated for the REAV model. The last row of the table contains average values. Note that all aborted faults were checked for untestability using the procedure from Section 3.7 based on the assumptions of the respective models.

Both single-threshold models yield quite similar results. It appears that the offset of the threshold to a realistic position in NREAV model does not incur additional costs. In contrast, considering two thresholds in REAV model appears to be more challenging, requiring more than 30% more patterns on average. We observed that the number of untestable faults is similar for both single-threshold models and some 10% larger for the REAV model. Also the number of aborts was significantly larger for the REAV model. On the other hand, the fault efficacy achieved by REAV test sets is the largest of all test sets.

Although 1-detection test sets are often larger than the test sets generated for interconnect opens, they typically cannot achieve their quality. 3-detection test sets tend to outperform simple models, yet their pattern count is excessive (more than twice the pattern count of REAV test sets). Assuming that the REAV model does reflect the realistic behavior of defects, it appears beneficial to consider this model when generating tests rather than use a simpler model.

Table 7 compares the run times of the proposed flow with [9] for open-via fault lists and complete fault lists, both in top-up pattern mode (with stuck-at patterns as an initial test set). The new flow outperforms the approach from [9] by two orders of magnitude despite more elaborate fault models being used.

We investigate the robustness of the generated test sets under process variations by a Monte-Carlo simulation. Table 8 summarizes the outcomes of 1,500 simulation runs for variations of coupling capacitances between the aggressors and the victim, the gate thresholds, and the combinations of both effects in circuit c7552. The variance σ of the parameters from their nominal value (NV) can be found in the table's first row. The average numbers of detected faults and the standard deviations are shown for five different test sets.

The relative detection capability of different models does not change under variations: REAV test sets cover most faults, followed by 3-detection stuck-at test sets, NREAV test sets, test sets generated assuming the model from [10] and stuck-at 1-detection test sets. The standard deviations are similar among different test sets, showing no clear trends towards one particular test set (although the test sets have different pattern count shown in Table 6). Variations of thresholds appear to have a larger effect on variations than variations of coupling capacitances. Interestingly, simultaneous

Table 7: Run time comparison with [9], in seconds

Circuit	Via top-up			Open top-up		
	[9]	prop.	speed up	[9]	prop.	speed up
c0880	356	4	89.00	514	5	102.80
c1355	1389	9	154.33	2076	10	207.60
c1908	1317	12	109.75	2119	13	163.00
c2670	2720	28	97.14	4852	37	131.14
c3540	4148	16	259.25	6776	25	271.04
c6288	5479	10	547.90	7732	13	594.77
c7552	14411	70	205.87	18801	93	202.16

Table 8: Effects of process variations on coupling capacitances and input voltage thresholds, circuit c7552

Test set	Capacitances only $\sigma = 5\%$ NV		Capacitances only $\sigma = 15\%$ NV		Thresholds only $\sigma = 5\%$ NV		Caps. and thresholds $\sigma = 5\%$ NV	
	Average detected	standard deviation	Average detected	standard deviation	Average detected	standard deviation	Average detected	standard deviation
Stuck-at 1-detection	54059.28	16.99	54052.07	31.54	54054.00	57.97	54060.47	55.95
Stuck-at 3-detection	54276.17	14.85	54273.96	29.37	54274.09	60.41	54281.34	57.64
Model from [10]	54088.75	17.87	54093.15	32.16	54094.49	60.82	54101.96	58.62
NREAV	54152.59	16.93	54139.69	32.66	54142.90	64.63	54148.91	63.06
REAV	54351.84	15.87	54316.16	30.49	54323.31	66.90	54327.20	64.39

variations of thresholds and coupling capacitances have a (slightly) weaker impact on standard deviation than variations of thresholds alone. This indicates that both variation mechanisms compensate each other's effects to some extent.

5 Conclusions and Future Work

We presented a flow for extraction, simulation and test generation for interconnect open defects based on improved representation of defect behavior by REAV and NREAV models. Numerous optimizations at all stages of the flow yielded an overall speed-up of two orders of magnitude compared to the previous work. While patterns generated under the REAV model are expected to keep their detection capability even in presence of noise, the model resulted in more challenging ATPG instances and larger test sets. In contrast, the complexity of test generation under the NREAV model turned out to be very similar to earlier modeling approaches.

One possible direction for the future research is the validation of the simulation results by more accurate electrical-level modeling taking factors such as gate leakage, trapped charge and dynamic effects into account, or a silicon experiment. Our simulation data suggest that, if the actual defects behave as described by the REAV model, the performance of NREAV patterns is limited (comparable to stuck-at test sets and below 3-detection test sets). A silicon experiment could clarify whether the restrictive detection conditions given by the REAV model are indeed required or whether NREAV model's weaker conditions are sufficient to achieve high defect coverage. A relevant question for circuits affected by statistical process variations is the appropriate selection of thresholds V_{thL} and V_{thH} in the REAV model.

Interconnect open defects may change the dynamic behavior of the circuit. Generating two-pattern tests (probably transition fault tests with additional constraints) is likely to allow the detection of faults which cannot be tested statically as well as resistive opens not covered in this work [16, 17]. Of special interest is the comparison with methods based on test application under non-nominal conditions such as Min-VDD [18]. Another open point is the generation of diagnostic patterns which are useful in distinguishing between different open defects.

6 References

[1] B. Koenemann. Test in the era of "What you see is NOT what you get". In *Int'l Test Conf.*, page 12, 2004. (Keynote address).

[2] S.P. Murarka, I.V. Verner, and R.J. Gutmann. *Copper—Fundamental Mechanisms for Microelectronis Applications*. John Wiley & Sons, 2000.

[3] R. Blish, T. Dellin, S. Huther, M. Johnson, J. Maiz, B. Likins, N. Lycoudes, J. McPherson, Y. Peng, C. Peridier, A. Preussger, G. Propkop, and L. Tullos. *Critical Reliability Challenges for the International Technology Roadmap for Semiconductors (ITRS)*, 2003.

[4] V.H. Champac, R. Rodriguez-Montanes, J .A. Segura, J. Figueras, and J.A. Rubio. Fault modelling of gate oxide short, floating gate and bridging failures in CMOS circuits. In *European Test Conf.*, pages 143–148, 1991.

[5] M. Renovell and G. Cambon. Electrical analysis and modeling of floating-gate fault. *IEEE Trans. on CAD*, 11(11):1450–1458, 1992.

[6] S. Rafiq, A. Ivanov, S. Tabatabaei, and M. Renovell. Testing for floating gates defects in CMOS circuits. In *Asian Test Symp.*, pages 228–236, 1998.

[7] D. Arumí, Rodríguez-Montañés, and J. Figueras. Experimental characterization of CMOS interconnect open defects. *IEEE Trans. on CAD*, 27(1):123–136, 2008.

[8] R. Gómez, A. Girón, and V. Champac. Test of interconnection opens considering coupling signals. In *IEEE Defect and Fault Tolerance Symp.*, pages 247–255, 2005.

[9] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng. Automatic test pattern generation for interconnect open defects. In *VLSI Test Symp.*, pages 181–186, 2008.

[10] Y. Sato, I. Yamazaki, H. Yamanaka, T. Ikeda, and M. Takakura. A persistent diagnostic technique for unstable defects. In *Int'l Test Conf.*, pages 242–249, 2002.

[11] H. Konuk and F. Joel Ferguson. Oscillation and sequential behavior caused by interconnect opens in digital CMOS circuits. In *Int'l Test Conf.*, pages 597–606, 1997.

[12] S. Spinner, J. Jiang, I. Polian, P. Engelke, and B. Becker. Simulating open-via defects. In *Asian Test Symp.*, pages 265–270, 2007.

[13] I. Polian, P. Engelke, M. Renovell, and B. Becker. Modeling feedback bridging faults with non-zero resistance. *Jour. of Electronic Testing: Theory and Applications*, 21(1):57–69, 2005.

[14] I. Hamzaoglu and J. Patel. Test set compaction algorithms for combinational circuits. *IEEE Trans. on CAD*, 19(8):957–963, 2000.

[15] E.M. Rudnick and J. Patel. Efficient techniques for dynamic test sequence compaction. *IEEE Trans. on Computers*, 48(3):323–330, 1999.

[16] W. Moore, G. Gronthoud, K. Baker, and M. Lousberg. Delay-fault testing and defects in deep sub-micron – does critical resistance really mean anything. In *Int'l Test Conf.*, pages 95–104, 2000.

[17] A. Czutro, N. Houarche, P. Engelke, I. Polian, M. Comte, M. Renovell, and B. Becker. A simulator of small-delay faults caused by resistive-open defects. In *European Test Symp.*, pages 113–118, 2008.

[18] B.R. Benware, R. Madge, C. Lu, and R. Daasch. Effectiveness comparisons of outlier screening methods for frequency dependent defects on complex ASICs. In *VLSI Test Symp.*, pages 39–46, 2003.