Efficient SMT-based ATPG for Interconnect Open Defects

Dominik Erb, Karsten Scheibler, Matthias Sauer, Bernd Becker
University of Freiburg, Georges-Köhler-Allee 51, 79110 Freiburg, Germany

***Abstract*—Interconnect opens are known to be one of the predominant defects in nanoscale technologies. However, automatic test pattern generation for open faults is challenging, because of their rather unstable behaviour and the numerous electric parameters which need to be considered. Thus, most approaches try to avoid accurate modeling of all constraints and use simplified fault models in order to detect as many faults as possible or make assumptions which decrease both complexity and accuracy.**

This paper presents a new SMT-based approach which for the first time supports the Robust Enhanced Aggressor Victim model without restrictions and handles oscillations. It is combined with the first open fault simulator fully supporting the Robust Enhanced Aggressor Victim model and thereby accurately considering unknown values. Experimental results show the high efficiency of the new method outperforming previous approaches by up to two orders of magnitude.

***Index Terms*—Interconnect opens, test generation, ATPG, unknown values, SMT**

I. Introduction

Especially since the transition to copper-interconnect technology [1] interconnect opens are known to be one of the predominant defects within nanoscale technologies [2, 3]. Commonly, such defects are caused by missing conducting material within the interconnect and may occur between metal layers (Inter-layer-opens or VIA-opens) as well as within a particular layer (Intra-layer-opens). An interconnect affected by an open defect is divided into two parts: a stable part connected to the driver and a disconnected floating part whose value is hard to predict because of its dependency upon several electric parameters [4, 5]. Likewise, the non-trivial electric modeling of the floating part, which is dominated by coupling capacitances between neighboring interconnects (aggressors) [6], makes interconnect opens hard to handle for ATPG tools. This leads to the existence of several different fault models describing the behaviour of open defects on the basis of an underlying electric modeling. Only a few approaches target interconnect opens directly with the use of layout information [2, 7–10].

In [7] only a subset of aggressors is used with the requirement that all of them are set to the opposite fault free value. The approach presented in [8] utilizes a branch&bound method to generate test patterns for interconnect opens considering the fault model by Sato [2], while [9] uses constrained stuck-at fault tests in case no circuit parameters are available or voltage based tests which tolerate circuit parameter uncertainty. In [10] the branch&bound method of [8] was combined with a segment stuck-at test generation algorithm and an untestability analysis based on the Nonrobust and Robust Enhanced Aggressor Victim model with restricted propagation conditions decreasing accuracy.

In addition there are also approaches which do not consider layout information [11, 12]. In [11] interconnect diagnosis is done by constructing a superset of possible faulty behaviours utilizing a net-model and path tracing in order to determine all possible fault locations, while [12] proposed the usage of multiple stuck-at faults in order to generate tests for open defects in circuits with large fanouts.

978-3-9815370-2-4/DATE14/©2014 EDAA

In this paper we present to the best of our knowledge the first approach which

- supports the Robust Enhanced Aggressor Victim model without making any propagation restrictions and therefore considers unknown values at the inputs of an affected gate if necessary,
- allows to explicitly generate static test patterns, i.e. test patterns showing no oscillating behaviour and being robust against process variations,
- is able to model thousands of aggressors within a single fault instance,
- is scalable to larger circuits with over 500k of non equivalent faults,
- includes an accurate simulator (extending [13, 14]) for open faults to allow accurate consideration of unknown values.

Our method utilizes the SAT Modulo Theory (SMT) solver iSAT3 [15–17] based on Interval Constraint Propagation (ICP) for modeling the fault behaviour as arithmetic constraints. Therefore, we are able to represent coupling capacitances precisely by large integers as they are generated by layout parameter extraction (PEX) tools. In principle such constraints could also be translated into a pure propositional formula being examined by a SAT solver. However, this would require to map the integers to rather small values and therefore would lead to a loss of accuracy. Furthermore, a high number of aggressors leads to very large encodings of the constraints, rendering a SAT-based approach even less feasible.

Our experimental results show that most of the faults are statically detectable and compared to previous approaches [8, 10] our approach is up to two orders of magnitude faster while being more accurate.

The remainder of this paper is organized as follows. We introduce the terminology and the fault model used in Section II. The proposed framework and our approach is described in detail in Section III. Section IV discusses the experimental results and Section V concludes the paper.

II. Terminology and problem statement

An interconnect consists of one source, several segments and one or multiple sinks as shown in Figure 1. A segment (also called RC-element) is a combination of a resistor r and zero or more capacitances CC. The capacitances represent the influence of other interconnect lines on the output of a segment. According to [8], all open defects on a given piece of an interconnect are mapped to an open fault at the output of the corresponding RC-element and break the interconnect in two parts: the first part starting from the source is the stable part and fault free; the second part is disconnected from the source and affected by numerous electric parameters. This part is also called the floating part. Several models exist in order to describe open defects with the use of layout information. Sato et al. [2] introduced the Aggressor Victim model and stated, that the floating part is mainly influenced by aggressors. The value of the coupling capacitance (CC_i) determines the influence of the aggressor i on the floating part. V_{DD} and V_{SS} can occur as aggressors,

but in contrast to normal signals their logic value cannot be changed by a test pattern. Therefore, the induced parasitic coupling capacitance of these aggressors is constant. In [2] a fault is testable, if one or several aggressors induce a non zero parasitic coupling capacitance to at least one RC-element of the floating part. C_0 represents the cumulative coupling capacitance of all aggressors showing logic 0. C_1 represents the cumulative coupling capacitance of all aggressors showing logic 1.

The voltage of the floating part (V_f) is assumed to be $V_f = \frac{C_1}{C_0+C_1}V_{DD}$. If the cumulative coupling capacitance C_1 exceeds C_0 and therefore $V_f > 0.5V_{DD}$, all gates driven by the floating part interpret the value as logic 1 and logic 0 otherwise. Accordingly, Sato set the threshold of each gate to $V_{DD}/2$.

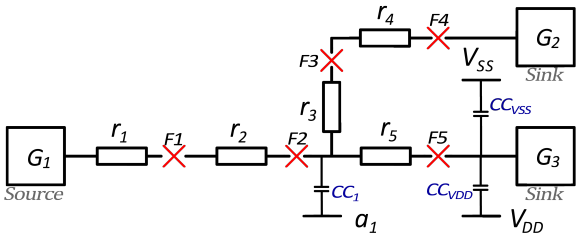

Fig. 1: Example of an interconnect represented as a tree of RC-elements with all possible open faults.

Figure 1 shows an interconnect with source G_1, two sinks G_2 and G_3 and five RC-elements. In total five open faults $F1, \ldots, F5$ are possible, which can be divided into two categories: Intra-layer opens only occur at VIAs, inter-layer opens can occur at each element of the RC-Tree. Depending on the fault location it may happen that not all gates connected by the interconnect are affected by the fault. Regarding Figure 1, Fault $F4$ only affects G_2 and has no influencing capacitances at all, while Fault $F5$ only affects G_3, but being victim of the aggressors V_{DD} and V_{SS} with the coupling capacitances CC_{VDD} and CC_{VSS}. Therefore, Fault $F4$ is untestable because there is no possibility to influence the floating part with any aggressor, while there may exist a valid test pattern for Fault $F5$.

A. Robust Enhanced Aggressor Victim model

In extension to [2], the authors of [10] tried to increase the accuracy of Sato's model by the use of two thresholds $V_{thL}(G)$ and $V_{thH}(G)$ for each gate G with $V_{thL}(G) \leq V_{thH}(G)$. Furthermore, different thresholds are allowed per gate type. As in [2], the voltage of the floating part is assumed to be $V_f = \frac{C_1}{C_0+C_1}V_{DD}$, however, two additional models are introduced: Robust Enhanced Aggressor Victim (REAV) and Nonrobust Enhanced Aggressor Victim (NREAV). Under the REAV model all voltages below $V_{thL}(G)$ are interpreted as logic 0 and all voltages above $V_{thG}(G)$ are interpreted as logic 1. A voltage between $V_{thL}(G)$ and $V_{thH}(G)$ may be interpreted as either logic 0 or logic 1 but the actually interpreted value is unknown.

To handle such behaviour [10] restricts the REAV model and assumes the interpreted value corresponds to the fault-free value if the voltage is between $V_{thL}(G)$ and $V_{thH}(G)$ i.e. it restricts the propagation conditions. We refer to this REAV with restricted conditions as *simplified REAV*. In contrast, we make no such restrictions and define these values as X-values. We refer to this new REAV model as *X-tolerant REAV*. Notice that a test pattern for simplified REAV might classify a fault as detectable, whereas a classification for X-tolerant REAV shows that the fault is actually not detected.

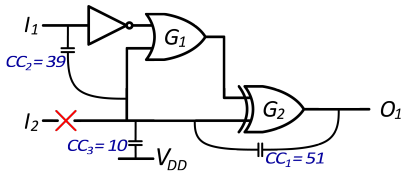

Fig. 2: Open fault at source of interconnect.

Figure 2 shows a circuit with a fault located at the source of the interconnect influenced by three aggressors I_1, V_{DD} and O_1. Let's assume the following thresholds are given for the inputs of the affected gates:

$$G_1 \text{ with } V_{thL}(G_1) = 0.42V_{DD},\ V_{thH}(G_1) = 0.47V_{DD}$$
$$G_2 \text{ with } V_{thL}(G_2) = 0.43V_{DD},\ V_{thH}(G_2) = 0.50V_{DD}$$

Considering simplified REAV, the test pattern $I_1 = 1, I_2 = 0$ would detect the fault – which is invalid regarding X-tolerant REAV. In detail, this pattern results in an induced voltage $V_f = \frac{CC_2+CC_3}{CC_2+CC_3+CC_1}V_{DD} = \frac{39+10}{39+10+51}V_{DD} = 0.49V_{DD}$ on the floating part. With the above thresholds, G_1 interprets the affected value to be logic 1 and G_2 assumes the fault-free value to be still present (i.e. logic 0) yielding an observable difference at output O_1. On the contrary, an approach for X-tolerant REAV interprets the value of G_2 as X-value resulting in no observable difference for all possible values of X.

Because O_1 itself depends on the fault site, an oscillating behaviour might be observed additionally [6][1]. A fault shows no oscillating behaviour, if the output values of all gates on the propagation path stay unchanged, i.e. the output values of all gates on a path starting at an affected gate and leading to at least one output used for fault detection do not change – irrespective of the aggressors being affected by a fault or not. If a test pattern is found, a fault is called *statically detected* in case no oscillation occurs on the propagation path – otherwise it is called *dynamically detected* as the values of all outputs used for fault detection may oscillate and therefore the fault effect might not be visible all the time. Hence, a fault marked as dynamically detected may lead to oscillation while a statically detected fault shows no oscillating behaviour on the propagation path. For the example stated above and the X-tolerant REAV model no test pattern which detects the fault statically exists. Instead, the pattern $I_1 = 0, I_2 = 1$ allows to mark the fault as dynamically detected[2].

For an input i of an affected gate G and the voltage of the floating part of a fault f computed by $V_f = \frac{C_1}{C_0+C_1}V_{DD}$, the interpreted value i depending on the voltage of the floating part V_f is summarized in Table I.

Voltage of floating part	interpreted value i
$V_f < V_{thL}(G)$	0
$V_{thL}(G) \leq V_f \leq V_{thH}(G)$	X
$V_f > V_{thH}(G)$	1

TABLE I: INTERPRETED VALUES ACCORDING TO REAV

Our approach explained in Section III mainly focuses on the X-tolerant REAV model. However, it is also applicable to the simplified REAV, NREAV or Sato model.

[1]For example O_1 may quickly toggle between logic 0 and 1 in the presence of an open fault, because if O_1 is 1 it induces a corresponding voltage to the fault site, thus the affected gate interprets the induced voltage differently leading to $O_1 = 0$; with O_1 set to 0 the induced voltage and the interpreted value would again change, leading to $O_1 = 1$.

[2]Because the induced voltage depends on the value of aggressor O_1 and different values lead to different interpretations of the voltage at gate G_2

III. Proposed ATPG framework

As seen in Figure 3, our proposed ATPG framework consists of three steps: (A) a preprocessing step identifies untestable and equivalent faults. (B) a fast accurate random pattern simulator could be used to reduce the number of faults which need to be considered by the next step. (C) the SMT-based explicit test pattern generation method either classifies all remaining faults as untestable (using three-valued logic) or returns a test pattern which is directly simulated to implement fault dropping.

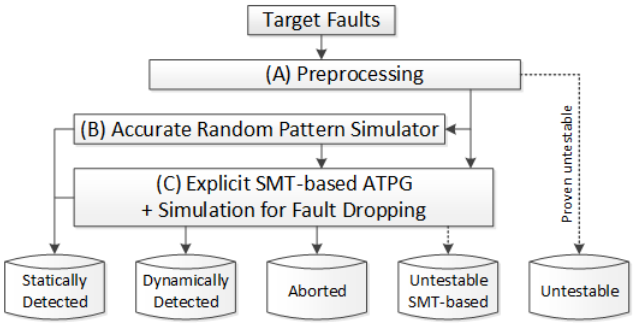

Fig. 3: Proposed ATPG Framework

A. Preprocessing

As stated in Section II, an interconnect open fault is located at the output of an RC-element. The preprocessing tries to identify both *equivalent* and *untestable* faults. A fault F is equivalent to another fault, if (1) the corresponding RC-element has no own capacitances and (2) there is at least one subsequent RC-element influenced by one or several aggressors between F's RC-element and the sink (e.g. Fault $F1$ in Figure 1 is equivalent to Fault $F2$). A fault is untestable, if up to the sink no aggressor lines influence at least one RC-element subsequent to the fault location (e.g. Fault $F3$ and $F4$ in Figure 1). A fault is also proven to be untestable if the voltage induced by all aggressors influencing the fault is interpreted as X-value by all affected gates, e.g. for all possible voltages induced by the aggressors, the voltage level of the floating part will be below $V_{thH}(G)$ and above $V_{thL}(G)$ (as for Fault $F5$ in Figure 1 if $V_{thH}(G_3) = 0.55V_{DD}, V_{thL}(G_3) = 0.45V_{DD}$ and $CC_{VDD} = CC_{VSS} = 0.5V_{DD}$).

B. Explicit test pattern generation using iSAT3

The proposed method exploits the expressiveness of SMT-formulas for interconnect open fault test pattern generation. SMT stands for SAT Modulo Theory and extends SAT by allowing arithmetic constraints. The constructed SMT-formula consists of two parts: (a) a Boolean encoding of the circuit and (b) arithmetic constraints describing the behaviour of the open fault. With this separation we are able to easily consider the different requirements of statically and dynamically detectable faults, because only the arithmetic constraints are changed and the Boolean part remains untouched. The SMT solver iSAT3 is used to determine the satisfiability of the constructed formula. If the formula is satisfiable, a test pattern is found. In case the formula is unsatisfiable, the fault is marked as *SMT-based untestable* according to Figure 3. As the problem of evaluating an SMT-based formula depends on the theory and is at least NP-complete (due to the underlying Boolean SAT problem), a time limit may be used for practical reasons. Instances not solved within the given time limit are marked as *aborted*.

1) The SMT solver iSAT3: SAT solvers calculate, whether a propositional formula is satisfiable or not. Usually the input formula is expected to be in conjunctive normal form (CNF). A CNF is a conjunction of clauses with each clause being a disjunction of literals. Most modern SAT solvers employ the conflict-driven clause learning framework (CDCL) which is an extension of the DPLL procedure introduced in [18]. While SAT expects Boolean variables or their negations as literals, SMT additionally allows arithmetic constraints. Besides variables and arithmetic operators, an arithmetic constraint contains a relational operator. During the SMT solving process all propositional variables and arithmetic constraints in the CNF are assigned to one of these values: `true`, `false` or `undefined`.

To satisfy an SMT-formula in CNF, every clause has to contain at least one literal or arithmetic constraint assigned to `true`. An additional check ensures, that arithmetic constraints assigned to `true` are indeed consistent. Therefore, traditional SMT solvers operate in two phases. First, a truth value is assigned to every literal or arithmetic constraint in order to satisfy the CNF. Second, it is checked if the arithmetic constraints are consistent – if not, a conflict clause is created and the search for a satisfying assignment is continued. The iSAT algorithm merges these two phases and tightly integrates ICP into the CDCL framework to reason about arithmetic constraints. In the context of this paper we concentrate on the problem specific encoding, for further details regarding the iSAT algorithm and iSAT3 refer to [15–17].

A very simple SMT-formula with Boolean variables a, b, c and integer variables x, y may look like this $((a \vee b) \wedge ((x + 3 \cdot y < 21) \vee c)$. The assignment $a =$`true`, $b =$`true`, $x = 3$, $y = 2$, $c =$`false` satisfies this formula. In fact the formulas created by our proposed method will look similar to the example: the clauses containing Boolean operations are introduced to encode the circuit; the arithmetic constraints (with multiplication, addition and large integer constants) represent the influence of the aggressors to an affected gate in relation to a threshold.

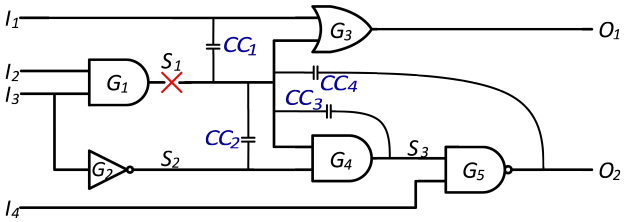

Fig. 4: Open fault at signal S_1 with multiple affected gates

2) Boolean encoding of the circuit: We describe the Boolean encoding with the help of an example. In Figure 4 S_1 is the output of G_1 as well as an input for G_3 and G_4. Assume the interconnect S_1 is broken. The stable part of the interconnect (in the following named $S_{1,1}$) is connected to G_1 and all gates not affected by the fault. The floating part consists of $S_{1,3}$ (connected to G_3) and $S_{1,4}$ (connected to G_4). The distinction between $S_{1,3}$ and $S_{1,4}$ is needed because the connected gates may interpret the induced voltage differently due to different gate specific thresholds. For example G_3 could interpret the induced voltage as logic 0, while G_4 assumes a X-value.

We now encode two different sub-circuits containing some or all gates and signals of the original circuit. The first sub-circuit $SCgood$ contains all signals and gates without modeling any fault. As stated above, $S_{1,1}$ is used for signal S_1 in the fault-free case. Therefore, we will use $S_{1,1}$ within $SCgood$. Regarding our

example, $SCgood$ contains $G_1, \ldots, G_5$ as well as the signals $I_1, I_2, I_3, I_4, S_{1,1}, S_2, S_3, O_1, O_2$.

The second sub-circuit $SCbad$ may reference gates or signals from the first sub-circuit, or it may contain copies of them. A gate G'_1 (or signal S'_2) stands for a copy of gate G_1 (or signal S_2). In particular this means G_1 and the copy G'_1 are of the same gate type, but S_2 and S'_2 may have different logic values.

In detail, $SCbad$ contains a copied version of all signals and gates having the floating part as input up to each reachable output. In our example, these are the gates G'_3, G'_4, G'_5 as well as the signals $S_{1,3}, S_{1,4}, S'_3, O'_1, O'_2$. As some gates in $SCbad$ are not driven exclusively by the floating part or by already included signals, all signals connected to the side inputs of these gates need to be included additionally referencing $SCgood$. Consequently, $SCbad$ contains the gates G'_3, G'_4, G'_5 and the signals $I_1, I_4, S_{1,3}, S_{1,4}, S_2, S'_3, O'_1, O'_2$.

It is possible that an affected gate interprets the induced voltage as an X-value. Hence, in the final formula $SCbad$ needs to be encoded three-valued while for $SCgood$ a two-valued encoding is sufficient. The generation of such a combined two- and three-valued CNF with the help of Tseitin transformations [19] has been proposed in [20]. Additionally, the search for a test pattern is sped up through D-chains which explicitly model the propagation paths of the fault effect [21]. Also, the number of gates and signals to be considered in $SCgood$ could be reduced by only encoding the union of the fault sites, all affected outputs' and the aggressors' input cones. The resulting SMT-formula consists of:

- CNF_{SCgood} encoded in two-valued logic (01),
- CNF_{SCbad} encoded in three-valued logic (01X),
- the D-chains,
- a constraint which enforces a difference between at least one output of $SCgood$ and its copy in $SCbad$,
- the arithmetic constraints we describe in the following.

3) Arithmetic constraints to ensure dynamic detectability: The arithmetic constraints contain the computation of the induced voltage V_f in relation to the thresholds of each affected gate. Because dynamically detectable faults are allowed to show oscillating behaviour it is sufficient to assume the aggressors are not influenced by the fault within the constraints. Hence, oscillations may or may not occur for a generated pattern. Therefore, oscillations could be ignored within the constraints and only the signals of the aggressors taken from $SCgood$ need to be considered within the computation of V_f.

According to Section II, the induced voltage is $V_f = \frac{C_1}{C_0+C_1} V_{DD}$ and C_1 is the sum of the coupling capacitances of all aggressors assigned logic 1. Assuming val_i represents the logical value of aggressor i in $SCgood$, the sum C_1 for a fault affected by n aggressors is expressed as $C_1 = \sum_{i=1}^{n} CC_i \cdot val_i$. This leads to

$$V_f = \frac{1}{C_0 + C_1} \cdot V_{DD} \cdot \sum_{i=1}^{n} (CC_i \cdot val_i)$$

$$\text{with the constants } V_{DD} \text{ and } C_0 + C_1 = \sum_{i=1}^{n} CC_i$$

The resulting constraints modeling an affected gate G along with its interpretation of the induced voltage and ensuring dynamic detectability are expressed as:

$V_f < V_{thL}(G)$	$\rightarrow 0$
$V_f > V_{thH}(G)$	$\rightarrow 1$
otherwise	$\rightarrow$ X

4) Arithmetic constraints to ensure static detectability: The arithmetic constraints ensuring static detectability are stricter compared to the dynamical ones. According to Section II, for static detection all affected gates used for fault propagation need to interpret the induced voltage similarly in a fault-free and in a fault affected circuit to ensure that no oscillating behaviour might be observed at any output used for fault detection. While in case of generating constraints to ensure dynamic detectability only the two-valued fault-free version (out of $SCgood$) of the aggressors is considered, we now have to additionally take the faulty three-valued version (out of $SCbad$) into account – if it exists. Regarding our example in Figure 4, S_3 is such an aggressor which is itself affected by the fault. Here it may also happen that the aggressor is set to an X-value in $SCbad$. According to Section II an X-value may be interpreted as either logic 1 or logic 0 but it is unknown which of both. For the constraints to ensure static detectability we need to guarantee the validity of the generated test patterns, independent of X-valued signals. Thus, we extend the arithmetic constraints to interpret an X-value as logic 0 if it is compared to the high threshold, and logic 1 if it is compared to the low threshold of a gate (i.e. the worst case). Consequently, in addition to V_f two supplementary voltages $V_f^{(X0)}$ and $V_f^{(X1)}$ are needed:

$$V_f^{(X0)} = \frac{1}{C_0+C_1} \cdot V_{DD} \cdot \sum_{i=1}^{n} CC_i \cdot val_i^{(X0)}$$
$$V_f^{(X1)} = \frac{1}{C_0+C_1} \cdot V_{DD} \cdot \sum_{i=1}^{n} CC_i \cdot val_i^{(X1)}$$

with

$$val_i^{(X0)} = \begin{cases} 0 \text{ if } val_i = 0 \\ 0 \text{ if } val_i = \text{X} \\ 1 \text{ if } val_i = 1 \end{cases} \qquad val_i^{(X1)} = \begin{cases} 0 \text{ if } val_i = 0 \\ 1 \text{ if } val_i = \text{X} \\ 1 \text{ if } val_i = 1 \end{cases}$$

These two supplementary voltages reference the signal within $SCbad$ if it is included, otherwise – as for V_f – the signal in $SCgood$ is referenced. The resulting constraints for an affected gate are:

$(V_f < V_{thL}(G)) \wedge (V_f^{(X1)} < V_{thL}(G))$	$\rightarrow 0$
$(V_f > V_{thH}(G)) \wedge (V_f^{(X0)} > V_{thH}(G))$	$\rightarrow 1$
otherwise	$\rightarrow$ X

C. X-aware interconnect open fault simulation

The event driven fault simulation engine used within the proposed ATPG is based on the simulator presented in [13, 14] extended for interconnect open faults. It uses a combination of numbered-X and two-valued simulation as well as a SAT-based approach in order to accurately consider all X-values which may be induced at affected gates. Hence, the simulator may classify faults as being detected by a test pattern which the SMT-based approach was not able to calculate a pattern for – as SAT- and hence SMT-based ATPG is not able to consider X-values accurately. The main differences between simulating stuck-at (as in [13]) and interconnect open faults are:

- the computation of the induced voltage and their effect on the floating part,
- the possibility that the fault effect is interpreted differently by affected gates,

- the ability to handle two different types of detection: dynamic detection and static detection.

In detail, prior to processing each fault, the simulator performs an accurate fault free simulation of the pattern to simulate. Afterwards, for each fault, first the induced voltage V_f is computed according to the values of the aggressors affecting it (as described in Section II), utilizing the results of the fault free simulation. According to the used model, V_f is subsequently compared to the thresholds $V_{thH}(G)$ and $V_{thL}(G)$ to determine the interpreted value. If the interpreted value causes a change in the output of an affected gate an event is created – otherwise not. Additionally, if the interpreted value is an X-value a new X-source is introduced at the affected gate. For further details on the event-driven simulation refer to [13]. All faults that are classified as detectable after this simulation are at least dynamically detected and need to be tested for static detection. All others stay undetected.

In order to test for static detection, the induced voltage is computed anew based on the results of the conducted simulation. As explained in Section III-B4 within this second test an aggressor may depend on the fault site and show an X-value. For the comparison with the high threshold (low threshold) of each affected gate, all aggressors assigned to an X-value are assumed to show a logic 0 (logic 1). Hence, we are assuming the worst case for each X-value. If no affected gate interprets the voltage differently than V_f, the fault is statically detected. Otherwise a second simulation is conducted searching for at least one still valid propagation path showing no oscillating behaviour. If such a path exists, the fault is also statically detectable, otherwise it stays dynamically detected.

IV. Evaluation

We evaluated the proposed approach using the combinational cores of ISCAS85 and ISCAS89 benchmarks. We use the same flow as in [10] for the circuit layout, the extraction of the parasitic coupling capacitances and the calculation of the gate thresholds $V_{thL}(G)$ and $V_{thH}(G)$. In contrast to [10] we always used the union of intra-layer (via-open) and inter-layer open faults and did not distinguish between them. Prior to each ATPG run, faults that are classified as equivalent by preprocessing (Section III-A) are merged.

A. Comparison to Previous Approaches

For a proper comparison with earlier approaches, we first measured the runtime of our approach utilizing the same machine as used in [8] and [10] (a single core of an AMD Opteron running at 2.3 GHz). As already mentioned [8] does explicit test pattern generation for the model proposed by Sato. To reduce the number of patterns to be handled explicitly [10] additionally uses random patterns and a method based on segment stuck-at faults for the simplified REAV model. In comparison our approach strictly focuses on the more accurate X-tolerant REAV model. Because of lack of space we only list circuits with a runtime larger than 5 seconds using our approach and results of earlier approaches – if available.

Table II compares the runtime of our results with those of [8, 10]. Additionally the number of aborts of [10] and our proposed method are shown. Apparently our approach is at least two orders of magnitude faster than [8] and on average 30.64 times faster than [10]. Furthermore, the number of aborts of our approach is significantly lower although we focus on X-tolerant REAV while [10] only considers simplified REAV. This difference in the number of aborts as well as in the runtime is most probably due to the fact that branch&bound based methods (as used in [8, 10]) generally explore many branches when there are many aggressors and affected gates. Therefore, methods based on branch&bound need multiple solver calls to determine one valid test pattern. On the contrary, our approach only needs one solver call per fault. We furthermore exploit the benefits of conflict learning in a CDCL SMT solver and are therefore more efficient – particularly for faults with a high number of aggressors or affected gates or with large cones of influence. In addition, these hard-to-detect faults are in most cases not found by random patterns or heuristic approaches using segment stuck-at faults.

TABLE II: Comparison to [8, 10]

Circuit	[8]	[10]		proposed		speedups	
	time	time	aborts	time	aborts	[8]	[10]
c6288	7732s	13s	124	40s	0	193	0.33
c7552	18801s	93s	259	33s	0	570	2.81
cs09234	-	420s	659	89s	0	-	4.71
cs13207	-	807s	809	259	10	-	3.11
cs15850	-	1 576s	999	185s	0	-	8.52
cs38584	-	47 530s	4 553	1 057s	2	-	44.97
$\sum$	-	50 961s	7 403	1 663s	12	-	30.64

B. Comparison of different ATPG modes

According to Section III-B, our approach supports two types of constraints which are now utilized in two different explicit test pattern generation modes. The first mode *DYNONLY* only considers detection requirements to ensure dynamic detectability. Within this mode the simulator may recognize that a fault is statically detected by a generated pattern, but there is no guarantee to find all statically detected faults. The second mode *COMB* combines the first with a more accurate investigation and handles oscillations to explicitly generate test patterns to ensure static detectability for all faults statically detectable according to Section II.

In order to compare these two modes, we conducted experiments on a single core of an Intel Xeon CPU running at 3.3 GHz. As solver backend the SMT solver iSAT3 is used with a timeout of 10 seconds. In both modes for each generated test pattern, the simulator proposed in Section III-C is used for fault dropping and classifies all faults detectable by that pattern as statically or dynamically detected.

Table III shows the results for the largest ISCAS85 and ISCAS89 benchmarks. The first three columns contain the circuit name along with the number of gates and the number of non-equivalent faults. The remaining columns contain the results of the two modes explained above. For both modes all faults were classified as statically or dynamically detected, untestable or aborted (i.e. iSAT3 hit the timeout).

In Columns 4 and 5 the number of statically and dynamically detected faults are listed for mode DYNONLY and in Columns 9 and 10 for mode COMB. To get a better overall picture we furthermore measured the number of detected faults for which an oscillating behaviour might be observed – but because of lack of space did not include the numbers in Table III. Among the 1 307 027 faults detected in both modes, 448 339 may show an oscillating behaviour. For all other faults, no aggressors depend on the fault site and therefore each generated test pattern detects the corresponding fault statically. Utilizing mode DYNONLY 383 912 faults which may oscillate are classified as statically

and 64 427 as dynamically detected. Utilizing mode COMB the number of faults only dynamically detected is reduced by a factor of 4.80 to only 13 422 dynamically detected faults on average. This means over 97% of all detectable faults which may show an oscillating behaviour are even though statically detected. In total utilizing mode COMB over 90% of the non-equivalent faults were marked statically detected. Summed up, out of 1.4 million faults (Column 3) only 72 were not classified in both modes (Columns 7, 12). Hence, over 99.99% of the non-equivalent faults were classified by our approach.

Although some circuits contain faults with up to 2 506 aggressors (cs38417) and up to 88 affected gates (cs38584) the runtime in both modes is quite low. The total runtime for mode DYNONLY was only ≈ 49 minutes and for mode COMB ≈ 85 minutes. Regarding scalability, our experiments show that the runtime is not only determined by the number of faults and gates, but rather by the number of hard-to-detect faults. Since only a few faults were not classified within the timeout of 10 seconds, our approach should also be able to classify most of the faults in circuits with more than 30k gates, which were not considered within this paper.

To summarize, our approach utilizing mode DYNONLY already allows to mark most of the faults as statically detectable if combined with an accurate fault simulation engine. However, additionally using ATPG to explicitly generate static test patterns, boosts the number of statically detectable faults further. Certainly, the much more complicated detection conditions lead to a moderately increased runtime. Nonetheless this does not hurt the scalability of our approach while providing more robust test patterns.

V. Conclusions

In this paper we presented a novel SMT-based approach for interconnect open faults able to accurately consider all aggressors and their influence on the interconnect open. To the best of our knowledge our method is the first which supports the X-tolerant READ model and allows to explicitly generate test patterns for statically detectable faults. The experimental results show that using our approach most of the faults could be marked as statically detectable even though over 30% of them may show an oscillating behaviour. Compared to previous approaches our method is more accurate and scales better for larger circuits.

In the future we want to extend our approach to generate test patterns for different types of defects and increase the scalability further.

Acknowledgments

The authors thank Professor Sudhakar Reddy from the University of Iowa and Linus Feiten, Stefan Hillebrecht from the University of Freiburg for supporting this work. This work was partially supported by the German Research Foundation (DFG) under grants BE 1176/14-2, SFB/TR14 AVACS and GRK1103.

References

[1] S. Murarka, I. Verner, and R. Gutmann, *Copper-fundamental mechanisms for microelectronic applications*. John Wiley, 2000.

[2] Y. Sato, L. Yamazaki *et al.*, "A persistent diagnostic technique for unstable defects," in *Test Conference, 2002. Proceedings. International*, 2002.

[3] G. Chen, S. Reddy *et al.*, "A unified fault model and test generation procedure for interconnect opens and bridges," in *Test Symposium, 2005.*

[4] S. Rafiq, A. Ivanov *et al.*, "Testing for floating gates defects in CMOS circuits," in *Test Symposium, 1998. ATS '98. Proceedings. Seventh Asian*, 1998, pp. 228–236.

[5] D. Arumí, R. Rodríguez-Montañés, and J. Figueras, "Experimental characterization of CMOS interconnect open defects," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 123–136, 2008.

[6] H. Konuk and F. Ferguson, "Oscillation and sequential behavior caused by interconnect opens in digital CMOS circuits," in *Test Conference, 1997. Proceedings., International*, 1997, pp. 597–606.

[7] R. Gomez, A. Giron, and V. Champac, "Test of interconnection opens considering coupling signals," in *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, 2005.

[8] S. Spinner, I. Polian *et al.*, "Automatic test pattern generation for interconnect open defects," in *26th IEEE VTS 2008.*, 2008.

[9] S. Reddy, I. Pomeranz, and C. Liu, "On tests to detect via opens in digital CMOS circuits," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, 2008, pp. 840–845.

[10] S. Hillebrecht, I. Polian *et al.*, "Extraction, simulation and test generation for interconnect open defects based on enhanced aggressor-victim model," in *Test Conference, 2008. ITC 2008. IEEE International*, 2008, pp. 1–10.

[11] S. Venkataraman and S. Drummonds, "A technique for logic fault diagnosis of interconnect open defects," in *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*, 2000, pp. 313–318.

[12] S. Reddy, I. Pomeranz *et al.*, "On testing of interconnect open defects in combinational logic circuits with stems of large fanout," in *Test Conference, 2002. Proceedings. International*, 2002, pp. 83–89.

[13] S. Hillebrecht, M. A. Kochte *et al.*, "Exact stuck-at fault classification in presence of unknowns," in *Proc. IEEE ETS*, 2012, pp. 1–6.

[14] M. A. Kochte and H.-J. Wunderlich, "SAT-based fault coverage evaluation in the presence of unknown values," in *Proc. Design, Automation and Test in Europe (DATE'11)*, 2011, pp. 1–6.

[15] M. Fränzle, C. Herde *et al.*, "Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 1, pp. 209–236, 2007.

[16] K. Scheibler, S. Kupferschmid, and B. Becker, "Recent improvements in the SMT solver iSAT," in *MBMV'13*, 2013, pp. 231–241.

[17] K. Scheibler and B. Becker, "Implication graph compression inside the SMT solver iSAT3," in *to appear in MBMV'14*, 2014.

[18] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem proving," *Communications of the ACM*, vol. 5, pp. 394–397, 1962.

[19] G. Tseitin, "On the complexity of derivation in propositional calculus," *Studies in constructive mathematics and mathematical logic*, vol. 2, 1968.

[20] D. Erb, M. A. Kochte *et al.*, "Accurate multi-cycle ATPG in presence of x-values," in *to be published in ATS13*, 2013.

[21] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. CAD*, vol. 11, no. 1, pp. 4–15, jan 1992.

TABLE III: Results of the proposed ATPG utilising different explicit test pattern generation modes.

circuit	gates	faults	DYNONLY: only generating dynamic test patterns					COMB: generating static and dynamic test patterns				
			statically detected	dynamically detected	untestable faults	aborted faults	runtime [s]	statically detected	dynamically detected	untestable faults	aborted faults	runtime [s]
c0499	202	2 498	1 859	189	450	0	1	1977	71	450	0	1
c6288	2 416	21 956	15 420	2 491	4 045	0	14	17 108	803	4 045	0	76
c7552	3 513	41 447	32 804	2 451	6 192	0	13	34 103	1 152	6 192	0	26
cs01238	509	7 712	6 122	404	1 186	0	2	6 445	81	1 186	0	2
cs01494	647	10 815	9 084	339	1 392	0	1	9 375	48	1 392	0	1
cs09234	5 597	61 915	48 203	4 017	9 695	0	34	50 521	1 699	9 695	0	54
cs13207	8 027	124 903	107 590	5 497	11 806	10	169	111 477	1 610	11 806	10	287
cs15850	9 786	127 906	106 760	6 857	14 289	0	60	111 393	2 224	14 289	0	95
cs38584	19 407	465 218	408 236	20 445	36 537	0	398	425 848	2 833	36 537	0	1 239
cs38417	22 397	558 845	506 522	21 737	30 524	62	2 270	525 358	2 901	30 524	62	3 304
$\sum$	72 501	1 423 215	1 242 600	64 427	116 116	72	2 962	1 293 605	13 422	116 116	72	5 085