Recent Advances in SAT-based ATPG: Non-Standard Fault Models, Multi Constraints and Optimization

Bernd Becker* Rolf Drechsler†‡ Stephan Eggersglüß†‡ Matthias Sauer*

*Department of Computer Science (IIF), Faculty of Engineering, University of Freiburg, Germany
{becker, sauerm}@informatik.uni-freiburg.de

†Group for Computer Architecture, Institute of Computer Science, University of Bremen, Germany
{segg,drechsle}@informatik.uni-bremen.de

‡DFKI GmbH, Cyber-Physical Systems, Bremen, Germany

***Abstract*—It is well-known that in principle automatic test pattern generation (ATPG) can be solved by transforming the circuit and the fault considered into a Boolean satisfiability (SAT) instance and then calling a so-called SAT solver to compute a test. More recently, the potential of SAT-based ATPG has been significantly extended. In this paper, we first provide introductory knowledge on SAT-based ATPG and then report on latest developments enabling applications far beyond classical ATPG.**

I. Introduction

Testing the correct functionality of nanoelectronic devices is an essential step in the production process as typical yield rates (i.e. the fraction of devices that perform properly) may reach numbers in the 50% range. Current processes have reached a point, where many classical pass/fail testing methods are enhanced by defect based fault models and/or grading-based techniques, assessing the quality of a circuit with regards to, e.g., circuit timing or power consumption.

An essential step for the whole test process is the generation of test patterns. They are generated by tools called *Automatic Test Pattern Generators* (ATPG).

Traditional ATPG algorithms, like the *D-Algorithm* [1] and successors [2], [3], work directly on the circuit structure, possibly in conjunction with additional data structures such as implication graphs [4] or advanced techniques to prune the solution space [5], [6]. The key idea of structural ATPG algorithms is to *propagate* a fault effect, using assignments to circuit lines, from the fault location to an observable circuit output. If this is possible without conflicting assignments, a *justification* phase tries to assign values to all supporting lines to find a *consistent* (i.e. conflict-free) assignment for each relevant line in the circuit. If a conflicting assignment has been identified, these assignments are reverted until either a consistent assignment has been found, or the complete search space has been covered. As we will see later, the advantage of structural methods, namely to directly work on the circuit structure, is also their weakness at least in the case of hard-to-detect or redundant faults. Nevertheless, state-of-the-art test generators are able to handle large industrial multi-million-gate designs.

It has long been known that an ATPG problem can be reduced to a *Boolean satisfiability* (SAT) instance and solved using a SAT solver [7]–[9]. However, this approach was not widely adopted as the structural approaches tended to exhibit better performance. More recently, significant improvements of the underlying SAT solvers in conjunction with extended solving capabilities specifically developed and tailored to ATPG changed this situation and led to an increased interest in such techniques.

SAT-based ATPG [10]–[13] has been shown to provide a high fault coverage for large industrial circuits. In particular, the powerful learning and implication techniques of modern SAT solvers are well suited to generate tests for hard-to-detect faults or determine the redundancy of faults. Classical structural ATPG approaches typically have problems to cope with these kind of faults as shown in [11].

The ability to handle redundant faults is becoming more important for two reasons. First, defects in nanoscale manufacturing technologies may not be described adequately by stuck-at faults [14]. Non-standard fault models such as resistive bridging faults [15], [16] or interconnect opens [17], [18] as well as robust delay test generation [19]–[21] may impose very specific conditions on the lines in the circuit, which are, in many cases, impossible to satisfy, so the fault is undetectable. Second, redundant structures are being increasingly used to enhance circuit reliability and yield [22], [23]. A significant fraction of faults in these structures are not detectable. To accurately estimate the defect coverage, the proof that the fault in question is undetectable (rather than aborted) is essential.

In this paper, we will first provide introductory knowledge on SAT-based ATPG with a particular emphasis on the techniques that led to SAT-based ATPG tools being competitive or even superior to classical structural ATPG solvers.

The main focus of the paper however will be on recent

advances and corresponding applications. In this context, the following observations are crucial: on the one hand, SAT solvers allow the convenient integration and flexible handling of multi constraints and their optimization; on the other hand, the capabilities of SAT solvers can be extended e.g., by using Pseudo Boolean SAT solvers or *SAT modulo Theory* (SMT) solvers. Exemplary applications will be sketched to provide insight and demonstrate the achievements.

The remainder of this paper is structured as follows: The next section briefly reviews basics on SAT-based ATPG. In Section III, we introduce complex static fault models and their modeling and utilization in SAT-based ATPG. Timing models and their analysis, e.g. sensitizable path computation and small delay fault detection is the topic of Section IV. In Section V, we demonstrate that optimization constraints can be integrated to allow the efficient computation of minimal test cubes and highly compacted test sets. We shortly review further extensions and current trends in Section VI and then conclude the paper.

II. ATPG Techniques

A. Boolean Satisfiability

This section briefly reviews the basics on the applied solving engines as well as on SAT-based ATPG. Solvers for *Boolean satisfiability* (SAT) and extensions thereof, like *Pseudo-Boolean Constraints* (PBC), MaxSAT and *Pseudo-Boolean Optimization* (PBO), are powerful formal proof engines which are frequently applied to solve complex problems in the field of circuit design. The underlying problems are defined as follows:

Definition 1 (SAT, PBC):

- Solvers for the SAT problem determine an assignment to the variables of a Boolean function $\Phi : \{0,1\}^n \rightarrow \{0,1\}$ such that Φ evaluates to 1 or prove that no such assignment exists. The function Φ is thereby given in *Conjunctive Normal Form* (CNF). A CNF Φ is a conjunction of clauses. A clause ω is a disjunction of literals and a literal x is a Boolean variable in its positive (x) or negative form ($\overline{x}$).
- Solvers for the *PBC problem* determine assignments to fulfill a conjunction of constraints defined by $\sum_{i=1}^{n} c_i \dot{x}_i \geq c_n$, where $c_1 \ldots, c_n \in \mathbb{Z}$ and $\dot{x}_i$ either is a positive or a negative literal, or they prove that that no such assignment exists.

MaxSAT and PBO define optimization versions of SAT and PBC, respectively. For the purpose of this paper, we restrict to the following definitions:

Definition 2 (MaxSAT, PBO):]

- In *MaxSAT*, a CNF formula is separated in soft and hard clauses. The optimization objective is to find an assignment that satisfies all hard clauses (as in SAT) and simultaneously maximizes the number of satisfied soft clauses.
- In *PBO* a PBC problem is extended by an objective function $\mathcal{F}$, which at the same time is to be minimized. The *objective function* $\mathcal{F}$ is defined by $\mathcal{F}(x_1, \ldots, x_n) = \sum_{i=1}^{n} m_i \dot{x}_i$ with $m_1, \ldots, m_n \in \mathbb{Z}$.

TABLE I
PBC and CNF representation for an AND gate $a \cdot b = c$

PBC	CNF
$(c + (1-a) + (1-b) \geq 1)\cdot$ $(a + (1-c) \geq 1)\cdot$ $(b + (1-c) \geq 1)$	$(c + \overline{a} + \overline{b})\cdot$ $(a + \overline{c})\cdot$ $(b + \overline{c})$

Example 1: Let $\Phi = (x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_3)(\overline{x}_2 + x_3)$. Then, $x_1 = 1, x_2 = 1$, and $x_3 = 1$ is a satisfying assignment solving the SAT problem.

Accordingly, let $\Psi = (2x_1 + 3x_2 + \overline{x}_3 \geq 3)(2x_1 + x_2 \geq 2)$ and $\mathcal{F} = x_1 + x_2 + x_3$. Then, $x_1 = 1, x_2 = 1$, and $x_3 = 1$ is a solution to the PBC problem given by the first two constraints, but obviously does not minimize $\mathcal{F}$. On the contrary, it follows easily that $x_1 = 1, x_2 = 0$, and $x_3 = 0$ is a solution to the PBC problem and, at the same time, minimizes $\mathcal{F}$.

SAT, PBC, MaxSAT and PBO are well investigated problems. In the past, efficient solving algorithms (so called *SAT solvers* or *PBC, MaxSAT, PBO solvers*, respectively) have been proposed (see e.g. [24]–[26]).

The effectiveness of these algorithms relies on several techniques. The most important techniques are efficient implication methods and powerful learning schemes. Conflict-based learning is performed during the solving process after a conflict, i.e. a conflicting assignment, occurred. Instead of simply backtracking as ATPG algorithms typically do, a conflict clause is recorded which prevents the solver to enter the same non-solution search space again. These conflict clauses enable the SAT solver to prune large parts of the search space as well as to understand complex relations between variables.

In the following, we apply these techniques as black boxes delivering the solution for the proposed problem formulations.

B. SAT-based ATPG

The application of the powerful SAT solving techniques to a circuit problem requires a transformation of the problem description into a Boolean formula in CNF as defined above. In order to create the SAT instance, the circuit $C = (S, G)$ with S as the set of signals and G the set of gates has to be transformed into CNF first.

For this purpose, each connection $s \in S$ of a circuit is assigned a Boolean variable x_s which represents the logical value of s, i.e. 0 or 1. Then, the functionality of each gate $g \in G$ is transformed into a set of clauses Φ_g using the Boolean variables associated with the input and output connections of g. The CNF can be easily derived for each gate type using truth tables or algebraic conversions. Table I shows an example formulation for an AND gate in PBC and CNF. The CNF Φ_C for the complete circuit C is then constructed by a conjunction of the CNF of each single gate of C, i.e.

$$\Phi_C = \Phi_{g_1} \cdot \ldots \cdot \Phi_{g_k}.$$

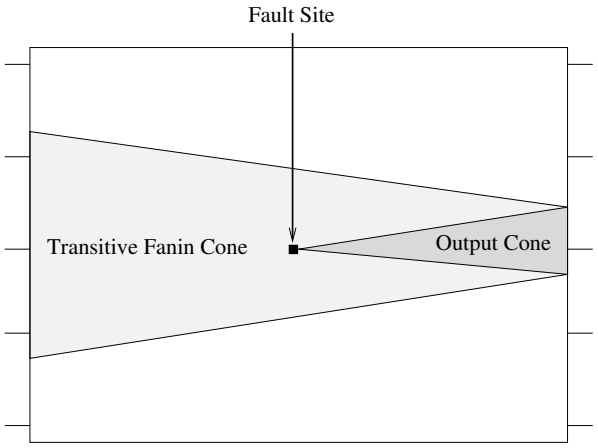

Fig. 1. Illustration of the SAT-based ATPG formulation (taken from [27]).

After the circuit CNF has been created, the SAT instance has to be augmented by the fault modeling. That is, given a fault f, e.g. a stuck-at-0 or stuck-at-1 fault, additional constraints Φ_F^f for fault excitation and fault propagation with respect to the specific fault f have to be formulated. The CNF Φ_F^f typically includes the complete output cone of the fault site, the faulty gate itself, and D-chain constraints to propagate the fault to an observation point [9]. Eventually, the following CNF formulation results as problem representation:

$$\Phi_{\text{Test}}^f = \Phi_C \cdot \Phi_F^f$$

This is illustrated in Figure 1. The solution space of Φ_{Test}^f includes all possible tests which detect f. Given to a SAT solver, the solver proves that Φ_{Test}^f is satisfiable by computing a satisfying assignment, which can be transformed into a test. If the fault f is untestable, the SAT solver proves that the solution space is empty, i.e. no satisfying assignment exists.

C. Multi Constraints and Optimization Techniques

Typically, the solution space for an ATPG problem for fault f includes more than one solution or even a large number of solutions. Each solution represents one test for f. As stated above, the importance of other issues for fault detection increases. Often, the detection of the pure fault is insufficient, e.g. sensitization constraints (see Sec. IV) have to be modeled for detecting small delay defects. However, the ATPG engine typically returns the first test found. This may not be the best test with respect to its ability to detect the fault. Therefore, guiding the ATPG towards finding not an arbitrary test but a test satisfying certain conditions is desirable.

As indicated above, this can be done by defining additional constraints (see also Sec. III), which sometimes have even to be handled by using an optimization solver, e.g. when computing test patterns with a minimal number of specified input bits, so-called minimal test cubes or when computing minimal test sets (see Sec. V).

III. Complex Static Fault Models

It is well-known, that complex defect mechanisms are not adequately covered by traditional fault models such as the stuck-at or transition delay fault model. As a consequence, *non-standard fault models* have been developed and specific ATPG-tools for individual fault models have been devised in the past.

As a more generic approach, in the following we present the *conditional multiple-stuck-at* fault model (CMS@) defined in [13], and an extension thereof the *enhanced conditional multiple-stuck-at* fault model (ECMS@) [28]. As a further application demonstrating the potential of SAT-based ATPG, we consider ATPG for interconnect open faults.

A. CMS@ and ECMS@ Fault Model

A CMS@ fault is given by r *condition lines* $a_1, \ldots, a_r$, $r \geq 0$, each one of them associated to a *condition* c_i; and by s *victim lines*, $s \geq 1$, each one of them associated to a logical value b_j. The following types of condition lines are supported: $\mathbf{0}$ – line is set to 0; $\mathbf{1}$ – line is set to 1; $\boldsymbol{F}$ (*fault-affected*) – fault effect must be propagated through that line; $\boldsymbol{NF}$ (*non-fault-affected*) – fault effect must not be propagated through that line. A circuit under a CMS@ fault exhibits faulty behaviour under any input vector that satisfies all conditions; in this case, every victim line v_j behaves as stuck-at-b_j.

It directly follows that a single-stuck-at-fault is represented by a CMS@ fault with an empty condition list and a victim list consisting of one entry. CMS@ allows the convenient integration of further fault models. As an example, we illustrate the mapping of the *resistive bridging fault model* on CMS@.

Bridging faults with non-zero bridge resistance may impact the behavior of a digital circuit in a non-trivial way [15], [29]. In general, a short defect with a non-zero resistance R_{sh} between interconnects a and b imposes intermediate voltages V_a and V_b between 0 and V_{DD} on the affected interconnects. These voltages are interpreted as logic values by the gates driven by a and b, depending on the logic thresholds of the gates. To detect a resistive short defect with a given resistance, specific values (detection conditions) on the gates driving the shorted interconnects may be required, and the fault effect may be visible on one or multiple gates driven by the shorted interconnects. These detection conditions may differ for short defects which involve the same pair of interconnects but have different resistances R_{sh}. It has been shown in [30], [31] that for every pair of interconnects a and b there is a finite number of representative resistances R_{sh} such that a test set, which detects all short defects with these resistances, covers all possible short defects between a and b. It is possible to formulate CMS@faults, in fact these are specific multiple stuck-at faults, which correspond to short defects with representative resistances. The details of the mapping are discussed in [31].

The CMS@ fault model can be further extended to allow the integration and optimization of further so-called *soft conditions*. We omit further details on the resulting fault model, called extended CMS@ (ECMS@) and its integration

in SAT-based ATPG. (For more details on optimization see also the following sections.) Rather we point out applications supported by ECMS@. As an example, a set of lines can be chosen and the number of 1s or 0s on these lines can be maximized or minimized, as well as the number of those lines that propagate a fault effect. In combination with time-frame expansion, this feature can also be used to generate test sequences needed for precisely controlling local switching activity during test-pair application, which is useful e.g. for noise-aware and low-power testing.

CMS@ and ECMS@ have been integrated in the SAT-based ATPG-tool TIGUAN. Besides testing of resistive bridges [13], applications include ATPG power-droop testing [32], minimization/maximization of fault affected outputs for stuck-at faults, and switching activity minimization for transition faults [28].

B. Interconnect Open Faults

CMS@ and ECMS@, are generic fault models working directly on the Boolean level. Depending on the defect class this may not be an adequate modeling level, even if static fault models are considered. Interconnect opens turn out to be such a case. Nevertheless, an extension to PBC solving allows efficient handling [33].

Interconnect opens are known to be one of the predominant defects within nanoscale technologies [34], [35]. 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 dominated by coupling capacitances between neighboring interconnects (aggressors) [36]. Different fault models exist describing the behaviour of open defects on the basis of an underlying electric modeling and layout information, among them the Robust Enhanced Aggressor Victim (REAV) model [37], an extension of Sato's Agressor Victim model [34].

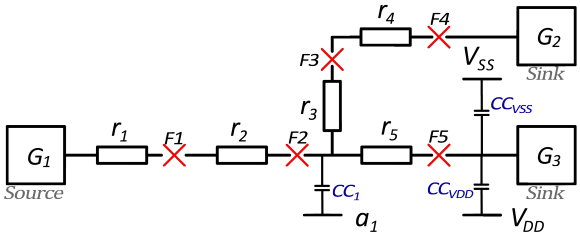

Fig. 2. Example of an interconnect represented as a tree of RC-elements with all possible open faults (taken from [33]).

An interconnect consists of one source, several segments and one or multiple sinks as shown in Figure 2. There are one source G_1, two sinks G_2 and G_3 and five so-called RC-elements. In total five open faults $F1, \ldots, F5$ are possible. Regarding Figure 2, 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}.

In general, 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.) C_0 (C_1) represents the cumulative coupling capacitance of all aggressors showing logic 0 (1). The voltage of the floating part (V_f) is assumed to be $V_f = \frac{C_1}{C_0+C_1} V_{DD}$.

In the REAV model for each gate type G, two thresholds $V_{thL}(G)$ and $V_{thH}(G)$ are given. The voltage V_f of the floating part is interpreted as logic 0 (1), iff $V_f < V_{thL}(G)$ ($V_f > V_{thH}(G)$. 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 X).

An ATPG method using the REAV model described above requires (a) a Boolean encoding of the circuit and (b) Pseudo Boolean Constraints describing the behaviour of the open fault. In [33] this is done utilizing the SAT Modulo Theory (SMT) solver iSAT3 [38], [39] based on Interval Constraint Propagation (ICP) for modeling the faulty behaviour. 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.

The resulting ATPG tool provides 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 for open faults to allow accurate consideration of unknown values.

IV. TIMING MODELS

The timing of a synchronous digital circuit is based on the notion of paths through which signal transitions travel from input to output. While so-called structural (longest) paths can be efficiently computed using graph algorithms, many tasks in particular with respect to timing analysis and delay test generation require knowledge on *sensitizable paths* and their length. Unfortunately, the identification of sensitizable paths turns out to be much harder than the determination of structural paths.

In contrast to previously known techniques that work on the circuit structure and separate path identification from sensitization checks [40], SAT-based computation of sensitizable paths offers the possibility to determine sensitizable paths of desired lengths (and other user-defined properties) together

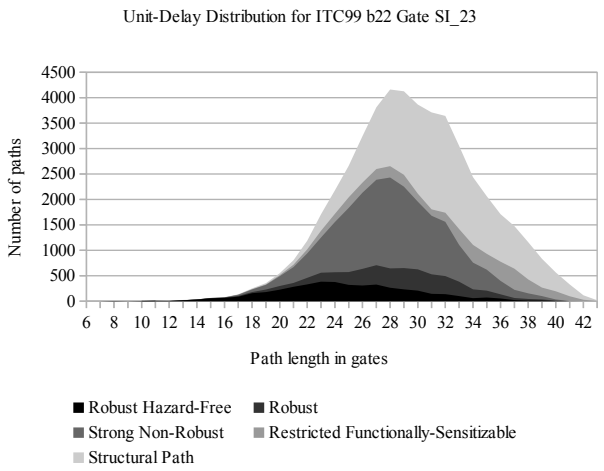

Fig. 3. Path delay histograms for input "SI_23" of circuit "b22" assuming different sensitization rules.

with primary input combinations executing the paths solving one monolithic encoding. We shortly present review three approaches going into different directions but all of them based on the above idea: PHAETON [41], an approach based on PBO [42], and WaveSAT [43]. realize this concept .

A. PHAETON

PHAETON[1] supports different delay models (unit delay, gate delay, pin-to-pin delay) and path-sensitization conditions (hazard-free robust, robust, strongly non-robust, restricted functional). The timing information is efficiently encoded directly into the SAT formula using an application-specific unary representation of integer numbers which supports operations required for controlling the length of found paths. Thanks to the monolithic formal encoding of the path generation instance as a SAT formula, PHAETON offers flexibility that by far exceeds that of structural procedures: a large number of requirements can be easily formulated, and application-specific functionality can be incorporated into the SAT formula generation.

Given the set of requirements, the PHAETON framework constructs a monolithic SAT instance Φ which is satisfiable if and only if at least one path which obeys the requirements exists. Among other optimizations, PHAETON maintains a *knowledge storage* which includes information learned during search; this storage is updated both when a solution has been found or not. It is possible to repeat the same request multiple times; in this case, PHAETON will search for different paths meeting the requirement by adding a *conflict clause* to prevent the previously found paths from being generated again.

We provide an example to demonstrate the potential of PHAETON: Some applications, including gate-criticality estimation [44], require the complete delay distribution of paths going through a specific gate. PHAETON is able to enumerate such paths and generate a histogram of path delays. Five such histograms are shown in Figure 3 for a specific circuit input and unit-delay model: one based on purely structural analysis and four assuming different path sensitization conditions.

PHAETON has been successfully employed for several tasks, from criticality estimation of circuit components [44], K longest path generation (KLPG) [45] to test generation for post-silicon validation and characterization [46], and variation-aware fault grading [47].

B. PBO-based Path Computation

In [42], a formal timing-aware ATPG approach relying on pseudo-Boolean encoding is presented. Using a higher level of encoding simplifies instance generation on the one hand, but has to be done carefully not to compromise scalability. The approach has been used to tackle *Small Delay Faults* (SDFs).

SDFs [48] extend the basic transition-delay fault by defining the fault as an increase in the delay of a specific gate g by a certain parametric value d. d is set to a rather small value, such that this additional delay may lead to a fault effect on some propagation path, but not on all. This results in a great increase in the complexity of the test generation as, unlike in the classical *Transition Delay Fault* (TDF), the detection of the fault depends on the delay of the propagation path. This fault model is related to the *path delay* fault model [49] where the additional delay is not allocated at a specific gate, but along a predefined path of the circuit. In order to accurately generate tests for SDFs, the delay of each gate needs to be known.

Typically, ATPG algorithms tend to sensitize short paths due to reasons of complexity. For a high-quality test, it is necessary that faults are detected via long or the longest sensitized paths. These paths have a smaller slack and are more likely to fail in the presence of SDFs. Timing-aware (structural) ATPG was proposed in [50]. Here, timing information is integrated into the search as a heuristic. As a result, the generated test typically sensitizes much longer paths than a test generated by regular ATPG. However, the run times as well as the pattern counts increase drastically as reported in [51].

SAT-based ATPG approaches to generate tests targeting longest paths were proposed in [42], [45]. The robustness of the underlying solving engines is leveraged to determine the provably longest paths through a fault site (dependent on the provided timing information).

An incremental search procedure is used to guide the test generation. The basic steps for TDF f are as follows:

1) Create a regular SAT-based ATPG instance Φ^f_{Test}. The solution space of this instance consists of all possible tests detecting f.
2) Create an objective function $\mathcal{F}_l$ which determines the length of a sensitized path of a test. This function includes a lower bound initially set to 0 or a low value.
3) Start the search procedure and generate an initial solution t_{init}. Evaluate $\mathcal{F}_l(t_{\text{init}})$ and use the result as a new lower bound of the objective. The initial solution does not necessarily sensitize a long path, although this is beneficially for the run time behavior. Save and exclude

[1] PatH Analysis and Enumeration on top of Test generatiON

the solution found from the solution space, e.g. by adding a conflict clause.

4) Continue the search while considering the new lower bound. If a new solution with a longer sensitized paths is found, update the lower bound and exclude the solution from the solution space as well.
5) If no better solution can be found, the last solution found, i.e. the optimal solution sensitizing the longest path, is returned.

The steps 3 – 5 are typically integrated into the solving engine and can be performed very efficiently.

The SAT-based timing-aware ATPG approach provides a powerful method for high-quality test generation. Therefore, the basic formulation is extended to target other hard problems. The approach in [42] integrates static signals using the formulation presented in [21]. This enables the generation of high-quality *robust tests*. In contrast to non-robust tests, a robust test guarantees the detection of a fault if other delay faults are present. By this, detecting the fault through the longest robustly testable paths is achieved.

C. WaveSAT

The detection of SDFs is traditionally performed by sensitizing a path of sufficient length from an input to an output of the circuit going through the fault site. While this approach allows efficient test generation algorithms, it may result in false positives and false negatives as well, i.e. undetected faults are classified as detected and vice versa, detectable faults are classified as undetectable. WaveSAT [43] is a SAT-based ATPG that overcomes these deficiencies by modeling waveforms on each relevant line of the circuit. The model incorporates individual delays for each gate and filtering of small glitches. It does not rely on the explicit notion of path sensitization. Instead, it directly considers the relationships between the possible waveforms on different lines of the circuit.

For a given SDF, a set of consistent waveforms on all circuit lines leading to detection is generated. The test pair is derived from the waveforms on the inputs (which are only allowed to switch once, at time 0). If no test pair is found, this constitutes the formal proof of untestability within the model assumptions. The waveforms are represented by series of Boolean variables which contain logical values at discrete points of time. Together with several optimizations in order to achieve scalability we succeed in systematically quantifying the inaccuracy obtained when using conventional path-oriented SDF ATPG for different sensitization conditions.

An example SAT encoding is given in Figure 4. It can be shown that no hazard-free-robustly sensitizable path through the fault location g_3 exists but the fault is nevertheless detectable.

WaveSAT can also be used for further applications, e.g. the detection of early-life failures [52].

V. Minimal Test Cubes and Compaction

A general shortcoming of SAT-based ATPG compared to structural ATPG is the over-specification of the computed test pattern. Typically, the SAT solver computes (in absence of structural information) a complete non-conflicting assignment of all Boolean variables in the SAT instance as satisfiability proof. The test is extracted from the assignment of the inputs. If done this way, the extracted values do not include any unspecified bits ("don't cares",X). This does not necessarily concern all inputs of the circuit, since only the relevant part of the circuit is represented in the SAT instance. However, the number of specified bits is typically higher than for structural ATPG, which is disadvantageous e.g. for test compaction or more sophisticated on-chip methods like "Embedded Deterministic Test" [53]. Moreover, underspecified tests offer great flexibility for controlling secondary test parameters like test power consumption [54]–[56] using refilling techniques. As the quality of these methods depends on the number of unspecified values, test patterns with as many unspecified inputs as possible are preferred.

Test relaxation techniques [57]–[59] tackle this problem. Thereby, static and dynamic methods have to be distinguished. Static methods require an initial test pattern and try to generalize input values by relaxation (replace specified by unspecified values without compromising the fault detection capabilities). Hence the quality of such methods depends on the initial pattern. Dynamic methods work with the requirements that the test imposes and try to find a test pattern with a high number of unspecified bits, a so-called *test cube* directly. In general, dynamic methods achieve better results, as they are free to choose the defined part of the input pattern in a way that improves the test cube. Only recently, the first method for computing provably minimal test cubes has been proposed. In the following subsection, we provide the key ideas of a SAT-based test cube minimization and then turn to test compaction.

A. Minimal Test Cubes

SAT-based methods to generate provably minimal test cubes are presented in [60].

The so-called *01X-logic* can be used to represent unspecified assignments in a circuit. $01X$-logic consists of three values $\{0, 1, X\}$, which represent three states: 0_{01X} (logic 0), 1_{01X} (logic 1) and X_{01X} (unknown, either logic 0 or logic 1). $01X$-logic can be expressed by propositional formulas. We encode the three-valued logic with two Boolean variables as follows: $0_{01X} = (1, 0),\ 1_{01X} = (0, 1),\ X_{01X} = (0, 0)$. The combination $(1, 1)$ is not allowed.

Figure 5 shows an example for the $01X$-encoding of a circuit where input x_1 is unspecified. Here, t_1 and t_2 indicate auxiliary variables which represent internal signals. In this example, the unspecified value is propagated to the output y_1, but not to y_0.

In Section II-B, we described how Φ^f_{Test}, the SAT instance corresponding to all tests for a given fault f is constructed when using Boolean logic. Analogously, we first create a SAT

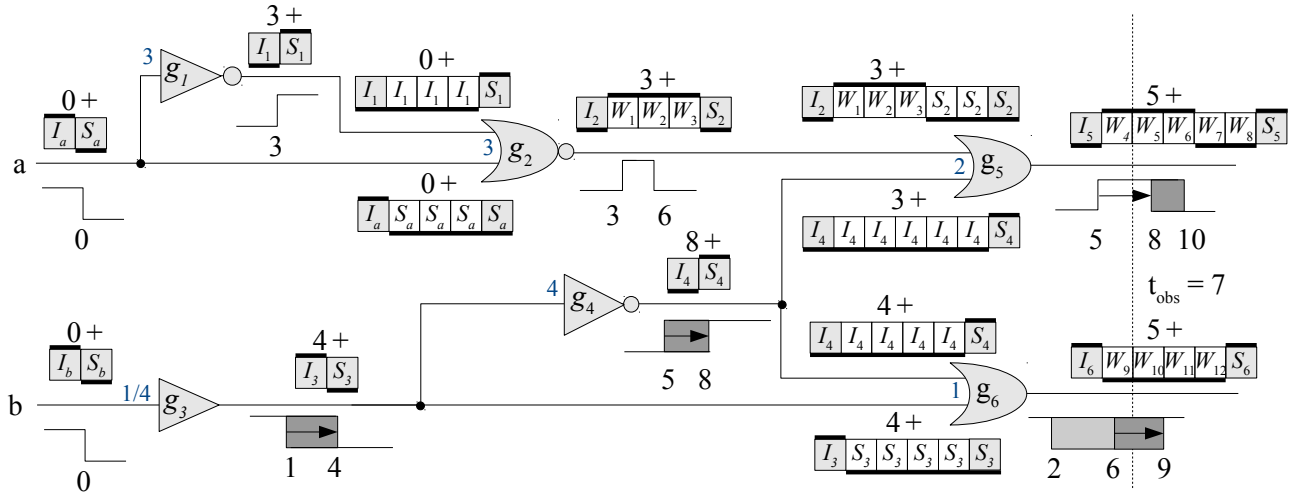

Fig. 4. Example SAT encoding of a $SDF(g_3, 3)$ (taken from [43]).

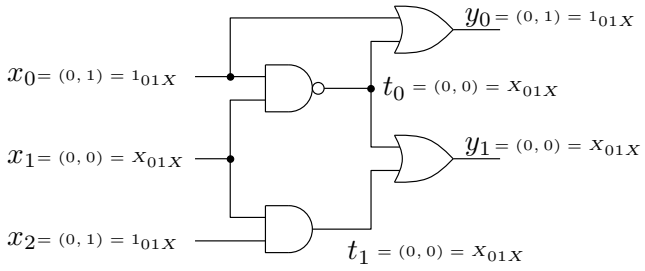

Fig. 5. $01X$ encoding for a certain input pattern (taken from [60]).

instance $\Phi^f_{\text{01x, Test}}$ modeling the set of all tests for fault f when using 01X-logic.

For all primary inputs, additional trigger variables t_i are introduced in addition. If $t_i = 1$, then the primary input I_i is assigned to $(0,0) = X_{01X}$, i.e. I_i is unspecified. Computing a minimal test cube can now be reduced to the following MaxSAT instance: We introduce a unit *soft* clause t_i and add the above implication as a hard clause to $\Phi^f_{\text{01x, Test}}$.

This method is $01X$-optimal, it calculates the minimal test cube for a $01X$-encoded circuit. Note, that this result may not be overall optimal in general, as is evidenced by the example in Fig. 5: $01X$-logic may incorrectly predict unspecified values on path revonvergences. As an example, consider again Figure 5, in particular y_1. With $01X$-encoding the assignment of y_1 is unspecified, although it will assume the value 1 regardless of the X-assignment of x_1: if $x_1 = 0$, the top input of the bottom OR is 1, therefore $y_1 = 1$ holds; if $x_1 = 1$ the bottom input of the bottom OR is 1, and therefore $y_1 = 1$ holds again.

As a second way to formulate and model unspecified values, an *exact formulation* with *Quantified Boolean Formula* (QBF) is introduced in [60]. Also the optimization problem considered above and the corresponding solution method can be extended to QBF and thus provably minimal test cubes can be obtained. For a detailed comparison and a discussion of the results obtainable, we refer to the original paper.

B. Local Fault Detection

However, not only the number of unspecified bits is important for an effective test compaction result, but also that the computed bits are compatible with other test cubes. It may happen that unfortunate bit assignments prevent other faults from being detected, e.g. blocked paths. The identification of "good" assignments is a difficult task if each fault is treated independently from other faults as structural ATPG approaches typically do.

In general, SAT-based algorithms are more powerful than structural ATPG algorithms. They are able to process more constraints more effectively. The work in [61] leverages this circumstance in order to improve the test compaction ability. Instead of targeting one fault only, the SAT instance is enhanced by local fault detection constraints. The main idea is that the reasoning engine should be guided not only to detect a primary fault f but also to satisfy as many local fault detection constraints as possible. By this, assignments which are potential useful for detecting other faults are made during the ATPG run without explicitly targeting other faults.

The local fault detection constraints consist of two additional variables x^s_{f0} and x^s_{f1} assigned to each connection $s \in S$ of a circuit C. These variables are associated with the corresponding stuck-at (transitions) faults f_0, f_1 of this line. The value of x^s_{f0} and x^s_{f1} represents whether all local fault detection conditions are satisfied. Additional constraints are added in order to imply the value of x^s_{f0} and x^s_{f1} from the circuit assignment. The following constraints are needed:

- The *activation constraints* Φ_{Activate} check whether the fault can be activated by the current circuit assignment, i.e. the value of x is the inverse of the fault value.
- The *gate constraints* Φ_{Gate} check whether all side inputs of the succeeding gate are non-blocking, i.e. assume the non-controlling value.
- The *path constraints* Φ_{Path} check whether a propagating path exists, i.e. whether the local fault detection con-

straints of the successor are satisfied.

In order to compute a test with enhanced detection ability, an objective function $\mathcal{F}$ is added. This function is created including all fault detection variables whose corresponding faults have not yet been detected. Given a set of yet undetected faults $F = f_1, \ldots, f_m$ with x^i being the connection of fault f_i ($1 \leq i \leq m$), the objective function $\mathcal{F}$ is formulated as follows:

$$\mathcal{F} = (-1) \cdot x^1_{f_1} + \ldots + (-1) \cdot x^m_{f_m}$$

The extended SAT instance $\Phi^f_{\text{Test}} \cdot \Phi_{\text{Activate}} \cdot \Phi_{\text{Gate}} \cdot \Phi_{\text{Path}}$ is given to a PBO solver along with $\mathcal{F}$. The solver yields the assignment which detects f and, at the same time, maximizes the satisfied local fault detection constraints. By this, the computationally power of modern solving engines is used to generate a test which potentially can be better compacted.

C. Multiple-Target Test Generation

The approach presented in [27] applies a different technique: Multiple-Target Test Generation (MTTG). Here, multiple faults are simultaneously considered. Giving a set of faults $f_1, \ldots, f_n$, the ATPG problem is formulated that a test t is generated which detects all faults $f_1, \ldots, f_n$ or prove that no such test exists. Obviously, the application of MTTG is advantageous for test compaction.

Early (structural) MTTG approaches [62], [63] were limited by the underlying ATPG technique. Only a very small of faults could be handled at the same time. The powerful modern SAT solving engines are able to handle a larger number of faults, since the conflict-driven learning techniques and heuristics are well-suited for such a restricted search space. In order to use SAT-based ATPG for MTTG for fault set $F = \{f_1, \ldots, f_n\}$, the SAT instance has to be extended as follows.

$$\Phi^F_{\text{Test}} = \Phi_C \cdot \Phi^{f_1}_F \cdot \ldots \cdot \Phi^{f_n}_F$$

The fault detection constraints $\Phi^{f_i}_F$ of each fault $f_i \in F$ have to be added to the SAT instance. This includes in particular the complete faulty output cone of the fault. The CNF Φ_C for the good circuit part can be shared for all faults. However, sharing the constraints of the faulty output cone is not possible since faults can be masked by this. A schematic illustration of this formulation is given in Figure 6.

The size of the SAT instance grows with the number of simultaneously considered faults. The MTTG formulation is a powerful technique to assemble a compact test set. However, a disadvantage is that a test can only be generated if all targeted faults can be detected by one test. If at least two faults conflict with each other, no test is generated. The search for a non-conflicting test set is a runtime-intensive task.

The technique presented in [64] resolves this issue. The proposed Optimization-based MTTG (OTG) technique wraps the problem by an objective function $\mathcal{F}$ which maximizes the number of detections. The optimization solver used generates a test with the maximal number of detected faults from a set of faults F. The SAT instance is modified such that faults can be dynamically deactivated if some faults conflict with each other.

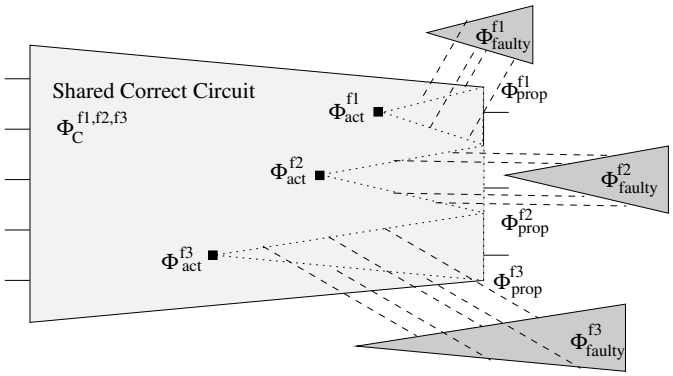

Fig. 6. Illustration of MTTG (taken from [27]).

TABLE II
TEST COMPACTION RESULTS

Circuit	Commercial	SAT
p35k	1523	624
p45k	2100	1780
p77k	552	549
p78k	82	78
p81k	379	556
p100k	2054	1690

The technique has been shown to produce a highly compacted test set. Additionally, an iterative application of this technique to a pre-generated compacted test set is able to further the size of the test set. Table II shows a comparison between a commercial (structural) ATPG tool and a SAT-based ATPG tool in terms of test set size.

In this section, we presented SAT techniques and extensions to be applied in the field of test compaction. Modern formal solving engines are used to increase the effectiveness of ATPG in terms of test compaction ability.

VI. Further Developments and Current Trends

There are several further developments enabled by SAT-based methods that complement the applications described until now. We just provide some pointers for further reading:

- The handling of unkowns (X) was shortly discussed in Section V. ATPG as well as fault simulation in the presence of unkowns is an interesting topic offering significant potential for SAT-based methods (see e.g. [65]–[67]).
- One of PHAETON's design goals has been to simplify the integration with applications making use of the versatile requirement system and hence enable sensitizable path-based concepts in novel areas beyond the classical scenario of small-delay fault testing, e.g., the functional detection of small delay faults through longest paths in sequential circuits [68], [69]. Thereby, optimization and compaction plays a role, as well as proving untestability by transfering methods known from the formal verification area [70] to the test area.
- A further step is to extend the above to a microprocessor environment and automatically generate functional mi-

croprocessor test sequences, e.g., for small-delay faults, based on Bounded Model Checking. A method is proposed for constraining the input space for generating functional test sequences (i.e., test programs) [71]. Also in this case, the close interaction between formal verification methods and ATPG knowhow is decisive for the success of the approach. Generating test programs more or less automatically for an on-line software-based self test of microprocessors will be a challenge in the future.

- Optimization with MaxSAT and the combination with so-called Bounded Model Checking (BMC) offers the possibility to have a fresh look at circuit initialization and even prove optimal results or disprove initializability [72].
- Power-aware test generation [73] is used to generate a test set which complies with the energy requirements of the chip. Current structural approaches struggle with the large number of additional constraints as well as with the problem of test inflation. SAT-based algorithms are a promising alternative in this field due to the presented advantages. A first SAT-based approach to reduce capture power was proposed in [74].

VII. Conclusions

We gave a brief description of basic SAT-based ATPG with a particular emphasis on the techniques that led to SAT-based ATPG tools being competitive or even superior to classical structural ATPG solvers.

The main focus was on recent advances and corresponding applications that go beyond classical ATPG. The successful and convenient integration of sophisticated static and dynamic fault models into SAT-based ATPG was demonstrated as well as the potential of SAT extensions, e.g., by using Pseudo Boolean SAT solvers or SAT modulo Theory (SMT) solvers or optimizing solvers. In particular, the advantages of SAT-based algorithms in the field of test cube minimization as well as dynamic compaction were shown. We finished the paper demonstrating the potential for future developments.

References

[1] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM Journal of Research and Development*, vol. 10, pp. 278–281, 1966.

[2] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic," *IEEE Trans. on Computers*, vol. 30, no. 3, pp. 215–222, 1981.

[3] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. on Computers*, vol. 32, no. 12, pp. 1137–1144, 1983.

[4] P. Tafertshofer and A. Ganz, "SAT based ATPG Using Fast Justification and Propagation in the Implication Graph," in *Int'l Conf. on Computer-Aided Design*, 1999, pp. 139–146.

[5] E. Gizdarski and H. Fujiwara, "SPIRIT: A Highly Robust Combinational test Generation Algorithm," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 12, pp. 1446–1458, December 2002.

[6] C. Wang, S. M. Reddy, I. Pomeranz, X. Lin, and J. Rajski, "Conflict driven techniques for improving deterministic test pattern generation," in *Int'l Conf. on Computer-Aided Design*, 2002.

[7] G. S. Tseitin, "On the Complexity of Derivations in Propositional Calculus," in *Studies in Constructive Mathematics and Mathematical Logics*, A. Slisenko, Ed., 1968.

[8] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 1, pp. 4–15, 1992.

[9] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1167–1176, 1996.

[10] J. Shi, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel, "PASSAT: Efficient SAT-Based Test Pattern Generation for Industrial Circuits," in *IEEE Int'l Symp. on VLSI*, 2005, pp. 212–217.

[11] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille, "On Acceleration of SAT-based ATPG for Industrial Designs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1329–1333, 2008.

[12] K. Yang, K.-T. Cheng, and L.-C. Wang, "TranGen: A SAT-Based ATPG for Path-Oriented Transition Faults," in *ASP Design Automation Conf.*, 2004, pp. 92–97.

[13] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. M. Reddy, and B. Becker, "Thread-Parallel Integrated Test Pattern Generator Utilizing Satisfiability Analysis," *International Journal of Parallel Programming*, vol. 38, no. 3-4, pp. 185–202, June 2010.

[14] R. C. Aitken, "New defect behavior at 130nm and beyond," in *IEEE European Test Symp.*, 2004, pp. 279–284.

[15] M. Renovell, F. Azaïs, and Y. Bertrand, "Detection of defects using fault model oriented test sequences," *Journal of Electronic Testing: Theory and Applications*, vol. 14, pp. 13–22, 1999.

[16] P. Engelke, I. Polian, M. Renovell, and B. Becker, "Automatic Test Pattern Generation for Resistive Bridging Faults," *Journal of Electronic Testing: Theory and Applications*, vol. 22, no. 1, pp. 61–69, February 2006.

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

[18] S. Hillebrecht, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng, "Extraction, Simulation and Test Generation for Interconnect Open Defects Based on Enhanced Aggressor-Victim Model," in *Int'l Test Conf.*, 2008, pp. 1–10.

[19] C.-J. Lin and S. M. Reddy, "On delay fault testing in logic circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 694–703, 1987.

[20] C. Chen and S. K. Gupta, "A satisfiability-based test generator for path delay faults in combinational circuits," in *Design Automation Conf.*, 1996, pp. 209–214.

[21] S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and R. Drechsler, "MONSOON: SAT-based ATPG for path delay faults using multiple-valued logics," *Journal of Electronic Testing: Theory and Applications*, vol. 26, no. 3, pp. 307–322, 2010.

[22] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems – Design and Evaluation*. Digital Press, 1992.

[23] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel, "Sequential element design with built-in soft error resilience," *IEEE Trans. on VLSI Systems*, vol. 14, no. 12, pp. 1368–1378, 2006.

[24] N. Eén and N. Sörensson, "An extensible SAT solver," in *Int'l Conf. on Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, vol. 2919, 2004, pp. 502–518.

[25] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-driven answer set solving," in *Int'l Joint Conf. on Artificial Intelligence*, 2007, pp. 386–392.

[26] T. Schubert, M. Lewis, and B. Becker, "antom — Solver Description," in *SAT Race*, 2010.

[27] S. Eggersglüß, R. Krenz-Bååth, A. Glowatz, F. Hapke, and R. Drechsler, "A new SAT-based ATPG for generating highly compacted test sets," in *IEEE Symp. on Design and Diagnosis of Electronic Circuits and Systems*, 2012, pp. 230–235.

[28] A. Czutro, M. Sauer, T. Schubert, I. Polian, and B. Becker, "SAT-ATPG Using Preferences for Improved Detection of Complex Defect Mechanisms," in *VLSI Test Symp.*, April 2012, pp. 170–175.

[29] P. Engelke, I. Polian, M. Renovell, and B. Becker, "Simulating resistive bridging and stuck-at faults," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2181–2192, October 2006.

[30] T. Shinogi, T. Kanbayashi, T. Yoshikawa, S. Tsuruoka, and T. Hayashi,

"Faulty resistance sectioning technique for resistive bridging fault ATPG systems," in *IEEE Asian Test Symp.*, 2001, pp. 76–81.

[31] P. Engelke, B. Becker, M. Renovell, J. Schlöffel, B. Braitling, and I. Polian, "SUPERB: Simulator Utilizing Parallel Evaluation of Resistive Bridges," *ACM Trans. on Design Automation of Electronic Circuits*, vol. 14, no. 4, pp. 56:1–56:21, August 2009.

[32] A. Czutro, M. Sauer, I. Polian, and B. Becker, "Multi-Conditional SAT-ATPG for Power-Droop Testing," in *IEEE European Test Symp.*, May 2012.

[33] D. Erb, K. Scheibler, M. Sauer, and B. Becker, "Efficient SMT-based ATPG for Interconnect Open Defects," in *Design, Automation and Test in Europe*, march 2014.

[34] Y. Sato, L. Yamazaki, H. Yamanaka, T. Ikeda, and M. Takakura, "A persistent diagnostic technique for unstable defects," in *Int'l Test Conf.*, 2002.

[35] G. Chen, S. Reddy, I. Pomeranz, J. Rajski, P. Engelke, and B. Becker, "A unified fault model and test generation procedure for interconnect opens and bridges," in *IEEE European Test Symp.*, 2005.

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

[37] S. Hillebrecht, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng, "Extraction, simulation and test generation for interconnect open defects based on enhanced aggressor-victim model," in *Int'l Test Conf.*, 2008, pp. 1–10.

[38] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert, "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.

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

[40] W. Qiu and D. M. H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit," in *Int'l Test Conf.*, 2003, pp. 592–601.

[41] M. Sauer, A. Czutro, T. Schubert, S. Hillebrecht, I. Polian, and B. Becker, "SAT-Based Analysis of Sensitisable Paths," in *IEEE Symp. on Design and Diagnosis of Electronic Circuits and Systems*, April 2011, pp. 93–98, Best Paper Award in the Test Category.

[42] S. Eggersglüß, M. Yilmaz, and K. Chakrabarty, "Robust timing-aware test generation using pseudo-boolean optimization," in *IEEE Asian Test Symp.*, 2012, pp. 290–295.

[43] M. Sauer, A. Czutro, I. Polian, and B. Becker, "Small-Delay-Fault ATPG with Waveform Accuracy," in *Int'l Conf. on Computer-Aided Design*, November 2012, pp. 30–36.

[44] ——, "Estimation of Component Criticality in Early Design Steps," in *IEEE Int'l On-Line Testing Symp.*, July 2011, pp. 104–110.

[45] M. Sauer, J. Jiang, A. Czutro, I. Polian, and B. Becker, "Efficient SAT-based search for longest sensitisable paths," in *IEEE Asian Test Symp.*, 2011, pp. 108–113.

[46] M. Sauer, A. Czutro, B. Becker, and I. Polian, "On the Quality of Test Vectors for Post-Silicon Characterization," in *IEEE European Test Symp.*, May 2012.

[47] A. Czutro, M. Imhof, J. Jiang, A. Mumtaz, M. Sauer, B. Becker, I. Polian, and H.-J. Wunderlich, "Variation-Aware Fault Grading," in *IEEE Asian Test Symp.*, November 2012, pp. 344–349.

[48] S. Goel and K. Chakrabarty, *Testing for Small-Delay Defects in Nanoscale CMOS Integrated Circuits*, ser. Devices, Circuits, and Systems. Taylor & Francis, 2013.

[49] G. L. Smith, "Model for Delay Faults Based upon Paths," in *Int'l Test Conf.*, 1985, pp. 342–349.

[50] X. Lin, K.-H. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware ATPG for high quality at-speed testing of small delay defects," in *IEEE Asian Test Symp.*, 2006, pp. 139–146.

[51] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test pattern selection for screening small-delay defects in very-deep submicron integrated circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 760–773, 2010.

[52] M. Sauer, Y. M. Kim, J. Seomun, H.-O. Kim, K.-T. Do, J. Y. Choi, K. S. Kim, S. Mitra, and B. Becker, "Early-Life-Failure Detection using SAT-based ATPG," in *Int'l Test Conf.*, 2013, pp. 1–10.

[53] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776 – 792, may 2004.

[54] R. Sankaralingam, R. Oruganti, and N. Touba, "Static compaction techniques to control scan vector power dissipation," in *VLSI Test Symp.*, 2000, pp. 35 –40.

[55] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, L.-T. Wang, K. Saluja, and K. Kinoshita, "Low-capture-power test generation for scan-based at-speed testing," in *Int'l Test Conf.*, nov. 2005, pp. 10 pp. –1028.

[56] K. Miyase, K. Noda, H. Ito, K. Hatayama, T. Aikyo, Y. Yamato, H. Furukawa, X. Wen, and S. Kajihara, "Effective IR-drop reduction in at-speed scan testing using distribution-controlling X-identification," in *Int'l Conf. on Computer-Aided Design*, nov. 2008, pp. 52 –58.

[57] I. Pomeranz, "Computing two-pattern test cubes for transition path delay faults," *IEEE Trans. on VLSI Systems*, vol. PP, no. 99, pp. 1 –11, 2012.

[58] N. Alawadhi and O. Sinanoglu, "Revival of partial scan: Test cube analysis driven conversion of flip-flops," in *VLSI Test Symp.*, may 2011, pp. 260 –265.

[59] S. Neophytou and M. Michael, "On the relaxation of n-detect test sets," in *VLSI Test Symp.*, 27 2008-may 1 2008, pp. 187 –192.

[60] M. Sauer, S. Reimer, I. Polian, T. Schubert, and B. Becker, "Provably Optimal Test Cube Generation Using Quantified Boolean Formula Solving," in *ASP Design Automation Conf.*, 2013, pp. 533–539, Best Paper Award Candidate.

[61] S. Eggersglüß, R. Wille, and R. Drechsler, "Improved SAT-based ATPG: More constraints, better compaction," in *Int'l Conf. on Computer-Aided Design*, 2013, pp. 85–90.

[62] G.-J. Tromp, "Minimal test sets for combinational circuits," in *Int'l Test Conf.*, 1991, pp. 204–209.

[63] J.-S. Chang and C.-S. Lin, "Test set compaction for combinational circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 11, pp. 1370–1378, 1995.

[64] S. Eggersglüß, K. Schmitz, R. Krenz-Bååth, and R. Drechsler, "Optimization-based multiple target test generation for highly compacted test sets," in *IEEE European Test Symp.*, 2014.

[65] S. Hillebrecht, M. A. Kochte, H.-J. Wunderlich, and B. Becker, "Exact stuck-at fault classification in presence of unknowns," in *IEEE European Test Symp.*, 2012, pp. 1–6.

[66] S. Hillebrecht, M. A. Kochte, D. Erb, H.-J. Wunderlich, and B. Becker, "Accurate QBF-based test pattern generation in presence of unknown values," in *Design, Automation and Test in Europe*, 2013, pp. 436–441.

[67] D. Erb, M. A. Kochte, M. Sauer, H.-J. Wunderlich, and B. Becker, "Accurate multi-cycle ATPG in presence of x-values," in *IEEE Asian Test Symp.*, 2013.

[68] M. Sauer, S. Kupferschmid, A. Czutro, S. M. Reddy, and B. Becker, "Analysis of Reachable Sensitisable Paths in Sequential Circuits with SAT and Craig Interpolation," in *Int'l Conf. on VLSI Design*, January 2012, pp. 382–387.

[69] M. Sauer, S. Kupferschmid, A. Czutro, I. Polian, S. M. Reddy, and B. Becker, "Functional Test of Small-Delay Faults using SAT and Craig Interpolation," in *Int'l Test Conf.*, November 2012, pp. 1–8.

[70] S. Kupferschmid, M. Lewis, T. Schubert, and B. Becker, "Incremental preprocessing methods for use in bmc," *Formal Methods in System Design*, vol. 39, pp. 185–204, 2011.

[71] A. Riefert, L. Ciganda, M. Sauer, P. Bernardi, M. Sonza Reorda, and B. Becker, "An Effective Approach to Automatic Functional Processor Test Generation for Small-Delay Faults," in *Design, Automation and Test in Europe*, march 2014.

[72] S. Reimer, M. Sauer, T. Schubert, and B. Becker, "Using MaxBMC for Pareto-Optimal Circuit Initialization," in *Design, Automation and Test in Europe*, march 2014.

[73] X. Wen, K. Enokimoto, K. Miyase, Y. Yamato, M. A. Kochte, S. Kajihara, P. Girard, and M. Tehranipoor, "Power-aware test generation with guaranteed launch safety for at-speed scan testing," in *VLSI Test Symp.*, 2011, pp. 166–171.

[74] S. Eggersglüß, "Peak capture power reduction for compact test sets using opt-justification-fill," in *IEEE Asian Test Symp.*, 2013, pp. 31–36.