Robust Algorithms for High Quality Test Pattern Generation Using Boolean Satisfiability

Stephan Eggersglüß Rolf Drechsler
Institute of Computer Science, University of Bremen,
28359 Bremen, Germany
{segg, drechsle}@informatik.uni-bremen.de

***Abstract*—Algorithms for *Automatic Test Pattern Generation* (ATPG) have to provide a high fault coverage in order to satisfy the quality demands of the chip industry. However, classical structural ATPG algorithms have problems to cope with the increased complexity of modern chip designs. The number of faults for which no test can be generated grows and the demands of the industry are compromised. New algorithms are necessary to retain the quality level. This results in a renewed interest in efficient ATPG algorithms which are fast and robust. ATPG algorithms based on *Boolean Satisfiability* (SAT) are a promising alternative to structural algorithms being very robust. However, SAT-based ATPG suffers from several limitations such as high run time or over-specified tests which prevent the use in industrial application.**

This paper proposes a SAT-based ATPG framework, which overcomes the limitations and allows for an efficient application in industrial practice. The framework is able to handle tri-state elements as well as unknown states. SAT formulations for the most prevalent fault models are proposed with special attention paid on the generation of high-quality tests. Novel techniques are introduced which boost the performance of the SAT-based ATPG process and reduce the number of unclassified faults to a minimum. Experimental results on large industrial circuits with multi-million elements show a significantly increased fault efficiency and very high fault coverage. The techniques presented make SAT-based ATPG suitable for the complexity of future designs and new complex fault models.

I. Introduction

The manufacturing process of today's chip designs is very susceptible to flaws. Therefore, each manufactured chip has to be subjected to a post-production test in order to filter out defective devices. A set of test patterns is applied in this test to ensure the correct behavior. Fault models are used to abstract from physical defects and the test set is generated by algorithms for *Automatic Test Pattern Generation* (ATPG).

Classical ATPG algorithms such as FAN [1] which are widely used in industry work on a structural netlist or on an implication graph [2]. These algorithms are very fast and many faults can typically be classified very quickly. Although being heavily improved in the last decades, e.g. by [3]–[7], these structural algorithms reach their limits and have problems to cope with hard-to-detect faults whose number is steadily increasing in today's complex designs. This leads to a growing proportion of faults which cannot be classified – especially for the *Transition Fault Model* (TFM) [8]. At the same time, the significance of delay testing and high-quality tests increases due the shrinking feature sizes. The growing proportion of unclassified faults compromises the demands of the industry for high fault coverage which is needed to guarantee high quality. This results in a renewed interest in efficient ATPG algorithms which are fast and robust, i.e. provide a high fault coverage in acceptable run time. Additionally, new fault models and the increased significance of effects like for instance crosstalk aggravate the need for new ATPG approaches which are able to cope with the increased complexity.

A promising solution to close the gap between test quality requirements and ATPG effectiveness is the application of solvers for *Boolean Satisfiability* (SAT). The SAT problem was the first problem proven to be NP-complete by Cook in 1971 [9]. SAT-based ATPG was proposed in the 1990s [10]–[12] and does not work on a structural netlist but on a Boolean formula in *Conjunctive Normal Form* (CNF). The problem is transformed into a Boolean formula Φ and a SAT solver, e.g. [13]–[15], is used to solve this formula.[1] However, the early approaches did not become widely accepted because of disadvantages such as overhead for circuit-to-CNF transformation, missing support of multiple-valued logics and over-specified solutions. Additionally, existing structural ATPG algorithms were fast enough to cope with designs of that time.

Since the 1990's, powerful SAT techniques have been developed and the efficiency of SAT solvers is still increasing. The homogeneity of the underlying CNF permits the application of efficient implication techniques and powerful conflict analysis strategies to solve SAT problems. Recently, the SAT-based ATPG tool PASSAT [16] has been proposed for stuck-at test pattern generation. PASSAT utilizes a multiple-valued logic and a Boolean encoding to cope with industrial characteristics such as input restrictions, unknown states and tri-state elements. The first results of PASSAT for industrial circuits were very promising. In particular, many hard-to-detect faults for which classical ATPG algorithms failed to generate a test could be solved by the SAT-based algorithm. However, SAT-based ATPG approaches has several shortcomings that prevent

This paper is an extended abstract of the doctoral thesis of Stephan Eggersglüß (with advisor Rolf Drechsler) which won the First Place at TTTC's E.J. McCluskey 2010 Best Doctoral Thesis Award Semi-Finals hosted at European Test Symposium (ETS).

[1] Other approaches, e.g. [7], are often considered as SAT-based ATPG as well. However, we use the term for approaches working with Boolean formulas in CNF.

978-1-4244-7207-9/10/$26.00 ©2010 IEEE

the efficient use in industrial practice. The main shortcomings are listed in the following:

- Run time – SAT-based ATPG shows promising results in classifying hard-to-detect faults. However, there is a significant overhead for easy-to-detect faults which typically represent the majority of faults. As a result, the overall run time is often not acceptable.
- Test pattern compactness – The ATPG process in an industrial environment is part of a greater flow. In order to obtain a small test set, test patterns are subsequently processed by techniques like test compaction or test compression. Typically, SAT-based ATPG generates over-specified test patterns which usually increases the size of the test set significantly. This is not acceptable since it directly results in higher test costs.
- Delay fault models – Test generation in an industrial environment is typically done for more than one fault model. The initial PASSAT approach does not consider other fault models than the stuck-at fault model, in particular not any delay fault model and the quality aspect.

The focus of the research work was to address the shortcomings listed above to make SAT-based ATPG applicable in industrial practice as well as to improve SAT-based ATPG in order to provide a basis for the growing complexity of future designs. The contributions presented in this paper can be summarized as follows:

- *New SAT formulations for fault models* – In order to enhance the general applicability of SAT-based ATPG, new efficient SAT formulations for the prevalent fault models suitable for industrial practice are provided.
- *Test quality improvements* – Test quality emerges as an important factor in the field of ATPG. Several techniques are presented which improve the quality of the generated tests.
- *Advances in SAT solving techniques* – Techniques are introduced which boost the performance of SAT-based ATPG as well as strengthen the robustness resulting in a very high fault efficiency.

Altogether, a SAT-based ATPG framework well suited for the use in an industrial test environment results. Figure 1 shows an illustration of this framework. The framework processes industrial circuits and can be applied for the fault models most widely used in industrial practice as shown in the upper part of the illustration. The core engine of the framework is a SAT solver which is enhanced with novel efficient SAT solving techniques in order to improve the performance and robustness. Additionally, certain options can be used to improve the quality or compactness of the test patterns. The framework is able to reduce the gap between the test quality requirements of the industry and the ATPG effectiveness significantly.

The proposed framework was implemented and integrated into the ATPG framework of NXP Semiconductors as a prototype in order to show the benefits which can be obtained using SAT-based ATPG in industrial practice. Experiments on large industrial circuits with up to 3.8 million elements show

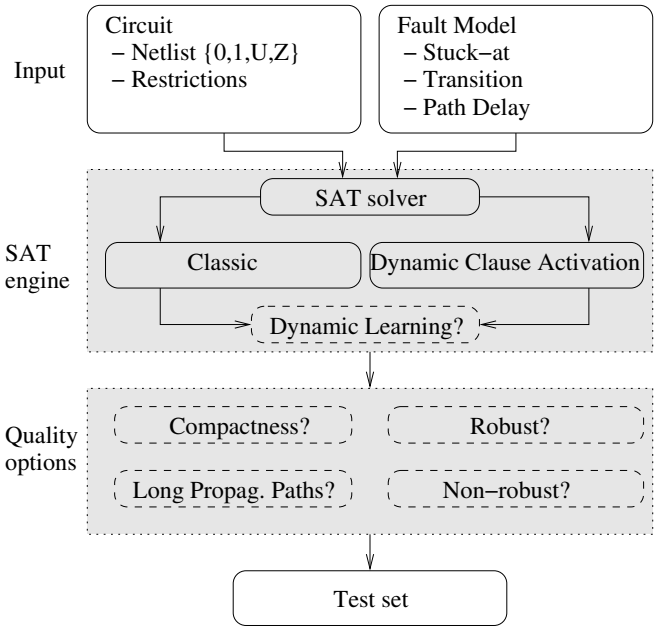


Fig. 1. Developed SAT-based ATPG framework for industrial application

the robustness and feasibility of the proposed techniques. The results also shows the high fault coverage and significantly increased fault efficiency which could be achieved by the proposed framework in industrial application.

The remainder of this paper is structured as follows. At first, Section II gives an introduction to SAT-based ATPG, especially for industrial circuits containing multiple-valued logic. Then, Section III deals with SAT formulations for the fault models most widely used in practice. Besides the pure SAT formulation of the ATPG problem, techniques which include structural information to speed up the search process and to increase the quality of the generated tests are presented. Section IV gives new solving techniques which are able to boost the performance and strengthen the robustness of the SAT-based ATPG process. Selected experimental results which show the robustness and feasibility of the proposed framework are presented in Section V. The conclusions are given in Section VI.

The developed techniques presented in this paper were published in several papers which are shown in the references section. These papers are cited differently (in alphanumerical style) within this paper to distinguish them from the normal references which are cited in numerical style.

II. Preliminaries

This section gives an introduction to the basics of SAT-based ATPG. Section II-A shows how SAT-based ATPG is conducted for Boolean circuits, while Section II-B deals with industrial circuits containing multiple-valued logic.

A. SAT-based ATPG

SAT-based ATPG works differently to classical structural ATPG. In order to make use of the efficient SAT techniques, the ATPG problem have to be represented as a Boolean

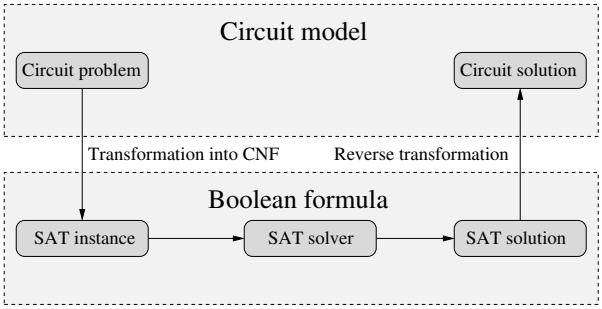


Fig. 2. SAT application flow

TABLE I
CNF FOR BASIC GATES WITH OUTPUT o AND INPUTS a, b (a FOR INV)

Gate	CNF
AND	$(o + \overline{a} + \overline{b}) \cdot (\overline{o} + a) \cdot (\overline{o} + b)$
NAND	$(\overline{o} + \overline{a} + \overline{b}) \cdot (o + a) \cdot (o + b)$
OR	$(\overline{o} + a + b) \cdot (o + \overline{a}) \cdot (o + \overline{b})$
NOR	$(o + a + b) \cdot (\overline{o} + \overline{a}) \cdot (\overline{o} + \overline{b})$
INV	$(o + a) \cdot (\overline{o} + \overline{a})$

formula in CNF. The general flow for applying a SAT solver to a circuit-oriented problem, e.g. ATPG, is shown in Figure 2. The original problem which is based on a circuit model must be transformed into a SAT instance in CNF. Then, the SAT solver is applied to the CNF to solve the formula. Finally, the obtained solution must be transformed from the SAT model to the original circuit model.

A CNF Φ in m Boolean variables is a conjunction of n clauses. Each clause is a disjunction of literals. A literal is a Boolean variable (x) or its complement ($\overline{x}$). The CNF Φ is *satisfied* if all clauses are satisfied. A clause is satisfied if at least one literal of the clause is satisfied. The CNF Φ is said to be *unsatisfiable* iff no solution can be found that satisfies Φ. The task of a SAT solver for a given Φ is to find a satisfying assignment or to prove that no such assignment exists. Due to the homogeneity of the CNF, the SAT solver is able to use very efficient implication procedures and learning schemes.

The SAT formulation for a single fault consists of two parts. First, those circuit parts which are to be transformed in CNF have to be identified. And second, the fault detection has to be modeled in CNF in order to find a test pattern. Consider Figure 3 for the description of the analysis for identifying the relevant circuit parts for one single fault. The starting point is the fault site. A depth-first search towards the outputs determines the *output cone* of the fault site. The output cone is the part of the circuit which could be influenced by the fault. Then, the *transitive fanin cone* of each output contained in the output cone is computed. Only those gates contained in this transitive fanin cone have to be considered when creating the SAT instance for generating a test pattern for the particular fault.

In the following, the circuit-to-CNF transformation is briefly described. More information can be found in [10]. A Boolean variable is assigned to each connection in circuit $\mathcal{C}$. The CNF Φ_g for each gate g in $\mathcal{C}$ is derived from the characteristic function which can be constructed using the truth table. Table I shows the CNF for the basic gates. The CNF $\Phi_\mathcal{C}$ representing the circuit's function is then constructed by the conjunction of the CNFs of all gates $g_1, \ldots, g_n \in \mathcal{C}$:

$$\Phi_\mathcal{C} = \prod_{i=1}^{n} \Phi_{g_i}$$

Note that $\Phi_\mathcal{C}^F$ describes the CNF of the circuit parts influenced by fault F as described above.

The CNF $\Phi_\mathcal{C}^F$ has to be extended by the fault-specific constraints Φ_F for generating a test for fault F. The fault-specific constraints include the fault site and a copy of the output cone describing the faulty circuitry. Additionally, structural information is used to encode the concept of D-chains [17] into CNF as proposed in [11].

More formally, a test for F is generated by evaluating the following formula:

$$\Phi_{test}^F = \Phi_\mathcal{C}^F \cdot \Phi_F$$

If Φ_{test}^F is unsatisfiable, the fault is untestable. A test can be easily derived from the satisfying assignment if Φ_{test}^F is satisfiable.

Example 1: Consider the example circuit shown in Figure 4. Figure 4(a) shows the good circuit and Figure 4(b) shows the same circuit containing a stuck-at-1 fault at line a. The Boolean difference is built in order to generate a test. This is shown in Figure 4(c). Note that structural information [11] is not included in this example for reason of simplicity. This

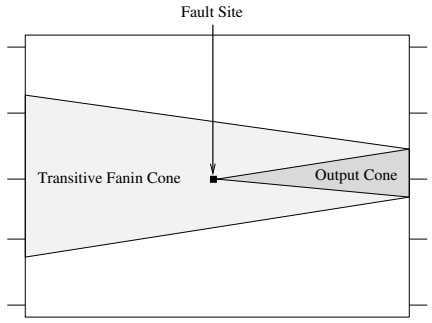
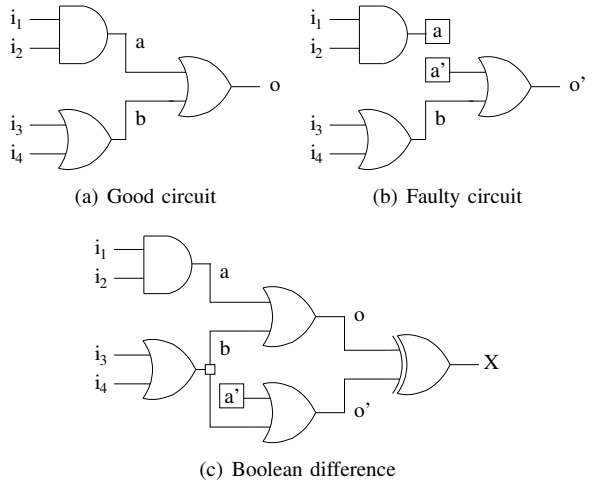


Fig. 3. Influenced circuit parts



(a) Good circuit (b) Faulty circuit

(c) Boolean difference

Fig. 4. Example for SAT formulation for the SAFM

TABLE II
CNF FOR BOOLEAN DIFFERENCE IN FIGURE 4(C)

$$\begin{array}{lcl} \Phi_C^F & = & (\bar{i}_1 + \bar{i}_2 + a) \cdot (i_1 + \overline{a}) \cdot (i_2 + \overline{a}) \cdot (i_3 + i_4 + \overline{b}) \\ & & \cdot (\bar{i}_3 + b) \cdot (\bar{i}_4 + b) \cdot (a + b + \overline{o}) \cdot (a + \overline{o}) \cdot (b + \overline{o}) \\ \Phi_F & = & (a' + b + \overline{o}') \cdot (a' + \overline{o}') \cdot (b + \overline{o}') \cdot (a') \cdot (\overline{o} + o' + X) \\ & & \cdot (o + \overline{o}' + X) \cdot (o + o' + \overline{X}) \cdot (\overline{o} + \overline{o}' + \overline{X}) \cdot (a') \cdot (\mathrm{X}) \end{array}$$

circuit is then transformed into CNF as shown in Table II. The SAT instance $\Phi_{test}^F = \Phi_C^F \cdot \Phi_F$ is given to a SAT solver. If the SAT solver determines UNSAT, the fault is untestable. If the SAT solver determines SAT, it automatically yields a solution which corresponds to a test pattern. Here, a satisfying solution of Φ_{test}^F is

$$i_1 = 0, i_2 = 1, i_3 = 0, i_4 = 0,$$

$$a = 0, a' = 1, b = 0, o = 0, o' = 1, X = 1$$

The test pattern T derives from the assignment of the input variables:

$$T = \{i_1 = 0, i_2 = 1, i_3 = 0, i_4 = 0\}$$

B. Multiple-Valued Logic

For practical purposes, considering only the Boolean values 0 and 1 during test generation as has been done in earlier SAT-based approaches is insufficient. A multiple-valued logic has to be used. There are two main reasons.

First, industrial circuits usually contain tri-state elements. These are used if a single signal is driven by multiple sources, e.g. bus structures. Besides the Boolean values 0 and 1, tri-state elements can assume another value, Z, modeling the state of high impedance.

Environment constraints that are applied to a circuit are another problem. The circuit can be embedded in a larger environment in industrial practice. As a result, some inputs of the circuit may not be controllable. The value of such a non-controllable input is assumed to be unknown, denoted by U. Unknown values have to be specially considered during ATPG.

Note that unknown values are not the same as don't care values. Don't care values are allowed to be assigned arbitrarily. Unknown values force a signal to be unassigned during the ATPG process. Considering unknown values and the state of high impedance, the following 4-valued logic is derived [16]:

$$\mathcal{L}_4 = \{0, 1, U, Z\}$$

Table III shows the truth table of an AND gate represented in $\mathcal{L}_4$. Other gate types are defined analogously. A Boolean encoding has to be used in order to apply a Boolean SAT solver to a circuit-oriented problem where the circuit is represented in a multiple-valued logic.

A Boolean encoding η is used to transform a multiple-valued problem in a Boolean problem. The value of each signal is represented by one Boolean variable in a purely Boolean circuit. In a multiple-valued circuit representation, one Boolean variable is insufficient. More Boolean variables have to be used to represent all values. Two Boolean variables c, c^* are used to encode all four values and represent the signal's value.

TABLE III
TRUTH TABLE FOR AN AND GATE IN $\mathcal{L}_4$

AND	0	1	U	Z
0	0	0	0	0
1	1	1	U	U
U	0	U	U	U
Z	0	U	U	U

TABLE IV
CNF REPRESENTATION OF AN AND GATE USING $\eta_{\mathcal{L}_4}$

$$\begin{array}{lll} (\overline{a} + \overline{b} + o) \cdot & (a^* + b^* + \overline{o}^*) \cdot & (\overline{a}^* + \overline{b}^* + o^*) \cdot \\ (a + a^* + \overline{o}) \cdot & (b + b^* + \overline{o}) \cdot & (\overline{a}^* + b + o^*) \cdot \\ (\overline{a} + \overline{b}^* + o) \cdot & (o + \overline{o}^*) & \end{array}$$

Table IV shows the CNF for an AND gate represented in $\mathcal{L}_4$ [18]. Clearly, the CNF representation is an overhead compared to the pure Boolean formulation. Typically, using $\mathcal{L}_4$ results in increased run time.

III. FAULT MODELS AND HIGH QUALITY

This section is structured with respect to the fault models used. The SAT formulations and improvements, respectively, are briefly described in the corresponding subsection.

A. Stuck-at Fault Model

The *Stuck-at Fault Model* (SAFM) [19] is the fault model most widespread in industrial practice. Early SAT-based ATPG approaches [11], [12] introduced SAT formulations for this fault model for Boolean logic. The recent SAT-based ATPG approach PASSAT [16] showed the SAT formulation for circuits containing multiple-valued logic which is also used in this work.

1) Hybrid Logic: The use of non-Boolean elements – such as tri-state elements – and unknown values necessitates the application of the four-valued logic $\mathcal{L}_4 = \{0, 1, U, Z\}$ as described in Section II-B. A Boolean encoding $\eta_{\mathcal{L}_4}$ is applied to transform the multiple-valued ATPG problem into a Boolean problem. As a result, efficient Boolean SAT-based algorithms can be applied. However, the size of the SAT instance increases significantly by applying a Boolean encoding. Transforming circuits with multiple-valued logic results in larger and often also more difficult to solve SAT instances than transforming circuits containing only Boolean logic.

Therefore, the use of a hybrid logic for SAT-based ATPG in industrial circuits is proposed [DEF+08][DEFT09]. 4-valued logic is used to model the circuit where necessary and Boolean logic where possible. The sources of Z and U values are identified and a structural classification is introduced to determine the logic used for each gate. The classification is applied as a pre-process and has to be done only once for each circuit. The run time overhead is negligible.

The experimental results clearly have shown that using hybrid logic instead of 4-valued logic only improves the performance of SAT-based ATPG significantly with respect to run time and unclassified faults and, thus, yields a more robust ATPG process.

2) Improved Compactness: A weakness of SAT-based ATPG is the large portion of specified bits in the computed test patterns, i.e. the test patterns are typically over-specified. While searching for a solution, state-of-the-art SAT solvers, e.g. zChaff [13] or MiniSat [14], prove either the unsatisfiability by showing that no solution for the given formula exists or the satisfiability by computing a satisfying assignment for the formula. The break condition of the latter case is the complete (non-conflicting) assignment of all variables. The large portion of specified bits is not acceptable in industrial practice. A large number of unspecified bits is required to apply essential techniques like test compaction and test compression effectively.

A post-processor is presented in [ED07][DEFT09] in order to overcome this limitation. Before the solution provided by the SAT solver is transformed into a test pattern, the solution is relaxed by a post-processor. Specified bits which are not necessary to detect the faults become unspecified by using structural information about the circuit.

In summary, the use of the proposed post-processor leads to a significant reduction of specified bits of test patterns generated by SAT-based ATPG as well as to a run time reduction depending on the padding strategy. Results from the application in industrial practice – which cannot be given here – confirmed that the generated test patterns are well suited for techniques such as test compaction and test compression.

B. Transition Fault Model

The *Transition Fault Model* (TFM) [8] takes the prevalent position among the delay fault models in the industrial production test. This fault model is widely used to ensure that a manufactured circuit is free of delay defects. One major reason behind the widespread use of the TFM is the similarity to the SAFM and its small fault population compared to other delay fault models. Nonetheless, test generation for transition faults is more complex than for stuck-at faults, the problem of the larger number of unclassified faults produced by classical structural ATPG algorithms is more serious for the TFM.

1) SAT Formulation: Test generation for the TFM has to incorporate the behavior of the circuits in at least two time frames. The work presented in [20] models the sequential behavior of the circuit as an *Iterative Logic Array* (ILA). The circuit's sequential behavior is mapped onto a combinational circuit by unrolling the combinational logic k times. For transition fault test generation, $k = 2$ holds.

The work presented in [ETF+07][DEF+09][DEFT09] shows how such an ILA using the launch-on-capture scheme [21] is represented in CNF and how stuck-at faults are injected in CNF to model transition faults.

Experimental results on large industrial circuits have shown that the proposed SAT formulation of the ATPG problem is well suited to cope with the increased complexity of the TFM. The large proportion of unclassified transition faults can be significantly reduced.

2) Long Propagation Paths: A *Small Delay Defect* (SDD) is a defect with defect size not large enough to cause a timing failure by its own. Due to the shrinking feature sizes and the increased speed of today's circuits, the likelihood of failures caused by SDDs increases and their detection has become an important issue in the production test [22]. Although being very small, SDDs might cause a timing violation when many of them are accumulated.

An SDD might escape during test application when a short path is sensitized since the accumulated delay of the distributed delay defect is not large enough to cause a timing violation. In contrast, the same SDD might be detected if a long path is sensitized [22], [23]. Unfortunately, common ATPG algorithms usually prefer short paths for transition fault propagation since the sensitization of these paths is typically more easier. Several techniques, e.g. [23]–[28], were proposed which enhance the quality of the tests by incorporating timing information into the search process or by utilizing test grading and selection.

Incorporating timing information into the search process is a large overhead resulting in a significantly longer ATPG process as reported in [28]. Therefore, a new SAT technique is proposed in [ETD08][TED10] which prioritizes long propagation paths during test generation. Instead of using timing information in the search process, a new incremental SAT instance generation scheme is used to incorporate timing information into SAT-based ATPG. Partial SAT instances are generated based on an output ordering. This ordering is generated based on the timing of the circuit. The sequence of partial SAT instances to solve forces the search process to choose a long propagation path.

By this, the efficient SAT solving techniques do not have to be modified and retain their robustness and efficiency. Therefore, the proposed method is very well scalable and produces only few run time overhead. Additionally, the increase in the number of unclassified faults is negligible. The experiments have shown that tests produced by this approach have significantly increased sensitized paths and, as a consequence, enhanced test quality.

C. Path Delay Fault Model

The most accurate delay fault model is the *Path Delay Fault Model* (PDFM) [29], [30]. This fault model captures small as well as large delay defects distributed along one path in the circuits. However, the number of paths in modern circuits is typically excessively large. For that reason, usually only tests for critical paths are generated to ensure the correct timing behavior of these paths. Furthermore, tests for the PDFM are used for diagnostic reasons if the timing behavior of particular paths should be verified. High-quality tests are required especially for diagnosis.

1) Robust Tests in Industrial Application: Concerning the quality, tests for PDFs can be roughly classified in two categories [31]: non-robust and robust. Both categories differ in the sensitization criteria of the path. Robust tests provide a higher quality and are therefore more desirable. However, static values have to be guaranteed for a robust test. This typically makes the robust test generation harder. Therefore, SAT-based algorithms are well suited for this kind of high-quality test generation. However, previous SAT-based approaches, e.g. [32], are not able to model the sequential behavior of the circuit

TABLE V
DERIVED LOGICS OF $\mathcal{L}_{19s}$

Logic	Value set
$\mathcal{L}_{11s}$	$\{S0, 00, 01, 10, 11, S1, 0U, 1U, U0, U1, UU\}$
$\mathcal{L}_{8s}$	$\{S0, 00, 01, 10, 11, S1, 0U, 1U\}$
$\mathcal{L}_{6s}$	$\{S0, 00, 01, 10, 11, S1\}$

adequately. A new SAT formulation for generating robust tests for the PDFM is proposed [EFD+07][EFD+07b][EFG+10].

An ILA representation as used for transition faults (and non-robust path delay test generation) is insufficient for the generation of robust path delay tests because two discrete points of time t_1, t_2 are modeled. However, no information about the transitions between t_1 and t_2 is given. Static values which guarantee the absence of glitches and hazards have to be modeled explicitly to guarantee a robust test. Therefore, the use of a 6-valued logic

$$\mathcal{L}_{6s} = \{S0, 00, 01, 10, 11, S1\}$$

for Boolean circuits and the 19-valued logic

$$\begin{aligned}\mathcal{L}_{19s} \quad = \quad & \{S0, 00, 01, 10, 11, S1, 0U, 1U, U0, U1, UU, \\ & 0Z, 1Z, Z0, Z1, UZ, ZU, ZZ, SZ\}\end{aligned}$$

is proposed. Each value represents the behavior of the signal in two time frames. For example. the value 01 denotes a rising edge, while 00 represents a zero which is not guaranteed to be static between t_1 and t_2. Static values are explicitly denoted by $S0, S1, SZ$. Since the use of the 19-valued logic is a large overhead for the CNF representation, a set of multiple-valued logics (presented in Table V) which is derived from $\mathcal{L}_{19s}$ is used. A structural analysis is applied to determine which values a gate can assume in which time frame. In contrast to the classification used for stuck-at faults, the sequential behavior of the circuit has to be modeled adequately and static values have to be included.

A Boolean encoding has to be used for each multiple-valued logic used. However, the Boolean encoding of a multiple-valued logic is not unique. Even for $\mathcal{L}_{11s}$, there are over one billion different possibilities. The work in [ED08] shows that the choosing the wrong Boolean encoding could result in a drastically increased run time which makes the application unacceptable. It is further shown how efficient encodings can be identified resulting in a fast and robust SAT-based ATPG process.

The application of the structural analysis and the combination of several multiple-valued logics and their Boolean encodings, respectively, is able to decrease the size of the SAT instance for test generation significantly. This enables the use of SAT-based ATPG for robust path delay test generation. Experimental results have shown the efficiency and robustness of the proposed SAT-based ATPG techniques even for large industrial circuits.

2) Incremental SAT Formulation: Generating robust tests for the PDFM is desirable. Unfortunately, typically only few paths in a circuit are robustly testable. For those paths which are not robustly testable, a non-robust test is generated (if one exists). So far, two SAT instances were generated and solved independently in this case. The fact that robust as well as non-robust test generation is executed sequentially can be leveraged by using incremental SAT. A new incremental SAT formulation is proposed for Boolean circuits [EFD08] as well as for industrial circuits [EFG+10].

Here, the SAT instance for non-robust test generation is enhanced with a direct encoding of static values in CNF which enables robust test generation. Consequently, robust test generation directly benefits from the previous non-robust test generation since a large part of the search space has been already traversed. Furthermore, all information learned by the SAT solver during non-robust test generation can be utilized for robust test generation. As a result, the run time needed for generating the test with highest possible quality can be significantly reduced and the fault coverage of robust tests can be increased. Only small run time overhead is produced compared to non-robust test generation only.

IV. SOLVING TECHNIQUES

The techniques and SAT formulations presented so far use the SAT solver as the solving engine as a black box. The problem is formulated as a SAT problem in CNF and the SAT solver is used to solve the problem. This causes some drawbacks for the application in the field of ATPG. Nonetheless, although being very robust in classifying many hard-to-detect faults, SAT-based ATPG algorithms suffer from the overhead for solving easy-to-detect faults which typically represent the majority of all faults. This leads to unacceptable high run times for some industrial circuits which makes the stand-alone application of SAT-based ATPG infeasible in industrial practice. In particular, the overhead is caused by the following drawbacks:

- Loss of structural knowledge – Classical ATPG algorithms benefit strongly from the structural knowledge about the ATPG problem. This knowledge is typically lost during the transformation into CNF.
- Transformation into CNF – Although the complexity of the CNF transformation is linear in the number of gates, the transformation time is not negligible. Especially in the ATPG domain where many instances based on the same circuit have to be solved, the transformation time is a significant overhead as reported in [11], [12].
- Completely specified solution – Due to reasons of efficiency, CNF-based SAT solvers compute a solution where all Boolean variables are specified, although many variables could be assigned with don't cares. This is disadvantageous for efficient test generation as well as for test compaction techniques.

A. Dynamic Clause Activation

The new SAT technique *Dynamic Clause Activation* (DCA) is proposed in [ETD09][ED10]. Using DCA, the SAT solver works on a subset of the original problem instance which is extended dynamically. This procedure has the following advantages:

- Exploitation of structural knowledge – Structural knowledge can be used during the search process due to retaining the gate connectivity information.

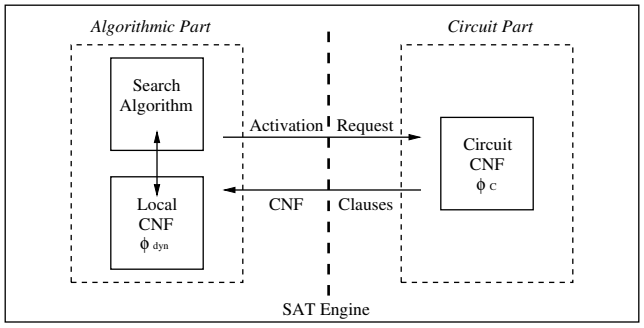


Fig. 5. Division of the proposed SAT engine

- Transformation into CNF – The CNF for the circuit is created only once. The required clauses are dynamically activated and by this the overhead of creating a complete SAT instance for each fault is significantly reduced.
- Implicit modeling of *Observability Don't Cares* (ODCs) – Due to the DCA technique, ODCs are modeled implicitly. By this, SAT core techniques, e.g. *Boolean Constraint Propagation* (BCP) and conflict analysis, do not have to be modified and retain their efficiency. Additionally, the generated tests contain an increased number of unspecified bits.

The modeling of ODCs or the appropriation of structural knowledge is not new [12], [33]–[38]. However, DCA is the first technique that permits the efficient combination and overcomes the problem of the high SAT instance generation time for ATPG. Core techniques such as fast BCP and conflict-driven learning which are very susceptible for changes are not altered. The proposed SAT engine which is named *Dynamic-SAT* is briefly sketched in the following.

In order to allow for a dynamic extension of the SAT instance, the proposed SAT engine using DCA is divided into two parts: a *circuit part* and an *algorithmic part*. This is illustrated in Figure 5. However, both parts are tightly integrated. The circuit part contains the complete CNF of the circuit (Φ_C) and serves as a database, whereas the algorithmic part contains the search algorithm working on a local CNF Φ_{dyn} representing the set of activated clauses. The search algorithm includes state-of-the-art SAT techniques such as conflict analysis and fast BCP and is only executed on Φ_{dyn}. In fact, each SAT solver can be taken as basis for the algorithmic part and extended to fit in a framework using DCA.

A set of rules is formulated based on structural information when activation requests has to be sent in order to activate the necessary parts of the CNF. The activation methodology is developed such that the efficient procedures of the SAT solver have not to be modified and retain their efficiency. Due to this technique, the time needed for SAT instance generation can be significantly reduced and ODCs can be modeled implicitly. This results in a significantly increased test compactness, i.e. portion of unspecified bits, especially if this methodology is combined with the post-processor described in Section III-A.

The experimental results on large industrial circuits for different fault models have shown that the proposed SAT engine are able to significantly reduce the run time of SAT-based ATPG by several factors on average and reduce or even close the run time gap between structural and SAT-based ATPG approaches. At the same time, the high level of robustness is retained.

B. Circuit-based Dynamic Learning

The proportion of unclassified faults produced by today's test generation algorithms grows due to the increased complexity of modern designs. However, a small percentage of unclassified faults, i.e. a high fault efficiency, is very important for the production test to keep a high fault coverage. A high fault coverage is needed to maintain a certain level of quality.

The application of SAT-based ATPG algorithms typically results in a significantly reduced number of unclassified faults compared to structural ATPG algorithms. SAT-based ATPG is very robust for hard-to-test faults due to the inherent learning strategies of modern SAT solvers. Although, many hard-to-test faults still remain unclassified in complex designs. A promising concept to strengthen the robustness of ATPG is the reuse of learned information.

SAT solvers learn in form of conflict clauses. Conflict clauses are recorded dynamically during the search process. Every time a conflict occurs during the search, the conflict is analyzed and a conflict clause is generated, i.e. learning is performed. In a circuit-oriented problem, a conflict clause corresponds to a conflicting value assignment of connections. The recorded conflict clauses can be used by a SAT solver to derive additional implications efficiently. Normally, all learned information is discarded after each run.

The tool TG-GRASP [12] introduced the plain concept of *pervasive* conflict clauses (or pervasive clauses in the following). These clauses depend on the circuit's function only and can be reused for each fault. However, no results concerning the benefit and information about the integration were given. An external database is used in [39] to store learned information. However, costly transformation steps are needed to utilize this concept and additional checks limit the amount of learned information.

A novel efficient circuit-based dynamic learning scheme is proposed in [ED09][ED10]. After solving the SAT instance for one fault, pervasive clauses are efficiently identified and stored in an internal database that the learned information can be reused to prune search space for subsequent faults. In particular, the internal database is combined with an efficient watch list strategy to avoid transformation steps and a checking procedure. The proposed procedure was designed to fit in an industrial test environment. The application results in a significantly decreased number of unclassified faults and very high fault efficiency. Additionally, the watch list strategy can be easily integrated into the activation methodology presented above. Both techniques – DCA and circuit-based dynamic learning – complement each other very well as shown in the next section.

V. Experimental Results

The presented SAT formulations, techniques and improvements were combined in a SAT-based ATPG framework and

TABLE VI
EXPERIMENTAL RESULTS – STUCK-AT FAULTS

	FAN		FAN long		PASSAT		DynamicSAT			DynamicSAT+		
Circ.	Ab.	Time	Ab.	Time	Ab.	Time	Ab.	Time	Imp.P	Ab.	Time	Imp.P
p44k	0	1:13m	0	1:13m	0	1:13h	0	5:10m	14.13x	0	5:20m	13.69x
p49k	2,719	4:26h	1,808	5:39h	6,611	7:10h	16,291	13:01h	0.55x	0	5:47h	1.24x
p57k	197	1:21m	138	6:55m	2	2:54m	3	0:58m	3.00x	3	1:00m	2.54x
p77k	0	0:07m	0	0:07m	0	0:08m	0	0:08m	1.00x	0	0:08m	1.00x
p80k	18	1:41m	12	2:40m	0	2:05m	1	1:26m	1.45x	0	1:25m	1.47x
p88k	56	1:36m	35	2:34m	0	1:53m	0	1:31m	1.24x	0	1:32m	1.23x
p99k	988	1:29m	455	6:05m	2	1:05m	9	1:16m	0.86x	8	1:23m	0.78x
p177k	99	2:13m	59	3:32m	0	1:28h	0	5:58m	14.75x	0	6:23m	13.79x
p456k	5,964	21:50m	3,524	2:14h	316	24:03m	1,282	59:09m	0.41x	204	1:32h	0.26x
p462k	1,009	9:17m	785	11:06m	72	40:14m	95	35:11m	1.14x	61	1:19h	0.51x
p565k	263	8:41m	175	11:21m	0	8:09m	0	18:00m	0.45x	0	17:59m	0.45x
p1330k	412	12:38m	282	13:33m	0	31:44m	38	59:03m	0.54x	33	57:56m	0.55x
p2787k	218,292	2:21h	165,699	11:53h	8,612	13:20h	45	6:45h	1.98x	36	6:51h	1.95x
p3327k	21,256	5:47h	15,990	11:22h	2,939	20:13h	2,435	18:11h	1.11x	440	20:18h	1.00x
p3852k	23,447	7:47h	17,247	11:44h	2,985	10:40h	11,796	15:07h	0.71x	224	12:51h	0.83x
Total	277,063		207,952		21,539		31,995			1,009		

TABLE VII
MONSOON – PDFM – ROBUST TEST GENERATION

	MONSOON		MONSOON+	
Circ.	Ab.	Time	Ab.	Time
p44k	0	2:02h	0	1:03h
p57k	898	1:40h	20	48:38m
p80k	901	1:01h	80	1:10h
p88k	0	8:30m	0	7:09m
p99k	1	6:12m	0	6:33m
p177k	3,042	10:17h	70	3:04h
p456k	164	1:32h	9	1:24h
p462k	0	39:49m	0	31:43m
p565k	78	27:53m	34	40:55m
p1330k	0	1:09h	0	58:58m
p2787k	55	3:13h	10	2:40h
p3327k	23	1:34h	1	2:21h
p3852k	72	4:14h	2	3:06h
Total	5,234		226	

integrated into the industrial test environment of NXP Semiconductors. The framework was extensively evaluated on large industrial circuits provided by NXP Semiconductors. This section provides some selected experimental results for each fault model in detail. Other results can be found in the referred papers.

A. Stuck-at Faults

Table VI shows the test generation results for the SAFM. The name of the circuit roughly denotes the size of the circuit, e.g. p3852k contains over 3.8 million elements. Fault dropping was enabled during the experiments. Column *FAN* gives the results for a highly optimized industrial FAN-based ATPG algorithm, while column *FAN long* shows results for the same algorithm with increased resources to reduce the number of unclassified faults. Results of the SAT-based ATPG approach PASSAT including all proposed techniques and SAT formulations which uses the SAT solver as a black box are given in column *PASSAT*. Column *DynamicSAT* shows the results for DynamicSAT using the DCA technique and column *DynamicSAT+* presents results for DynamicSAT using DCA and circuit-based dynamic learning. The number of unclassified faults are given in column *Ab.* Run time is given in either CPU minutes (m) or CPU hours (h) in column *Time*.[2]

The structural ATPG FAN is very fast but has a serious problem with the proportion of unclassified faults. Increasing the resources leads to a significant run time increase but results only in a slight decrease of the number of unclassified faults. On the other hand, the SAT-based ATPG approach PASSAT produces only few unclassified faults. However, the run time is very high in some cases compared to FAN. DynamicSAT+ is very fast and increases the performance of SAT-based ATPG by up to a factor of 14.75x compared to PASSAT (column *Imp.P*). In particular, DynamicSAT is very fast for those circuits for which PASSAT is unreasonably slow. At the same time, the number of unclassified faults is still very low compared to FAN. DynamicSAT+ is able to further reduce the number of unclassified faults to a minimum. Therefore, the combination of DCA and circuit-based dynamic learning overcomes the limitations of classical SAT-based ATPG and provides a fast and robust ATPG process.

B. Path Delay Faults

Table VII presents the results of MONSOON, the proposed SAT-based ATPG approach for robust path delay test generation. Unfortunately, to the best of our knowledge, no free path delay test generator that can generate robust tests and/or can handle static values and industrial constraints is available for comparison.

Column *MONSOON* gives the results of the proposed SAT-based ATPG approach as presented in Section III-C, while column *MONSOON+* is the same approach incorporating circuit-based dynamic learning. The 40,000 longest paths of each circuit were chosen for test generation and the launch-on-capture scan scheme is used. The experimental results show that MONSOON is very fast for robust test generation and produces only few unclassified faults for most circuits. MONSOON+ is able to accelerate the ATPG process and at the same time, diminish the number of unclassified faults significantly. Therefore, the proposed approach is well suited for high-quality test generation in industrial practice.

[2]Results concerning the test set size cannot be given.

C. Transition Faults

The proposed techniques have been evaluated for the TFM. However, the concrete results are not presented here. Instead, Table VIII shows the impact of the proposed SAT-based ATPG approach DynamicSAT+ on the fault coverage and fault efficiency for the TFM compared to the industrial FAN-based ATPG. Column *%FC* gives the fault coverage which could be achieved with the corresponding approach and column *%FE* presents the fault efficiency. The fault efficiency (or test coverage) is defined as the percentage of testable faults in faults not identified as untestable. Column *%FC Inc.* gives the fault coverage increase of DynamicSAT+ compared to FAN.

The fault efficiency of DynamicSAT+ is very high being either 100% or between 99.5% and 100%. This signifies a considerable increase compared to FAN and shows the robustness of the proposed approach. Furthermore, the application of DynamicSAT+ results in a significant fault coverage increase of up to 2% which is very important for the high quality demands of the industry.

TABLE VIII
IMPACT ON FAULT COVERAGE / FAULT EFFICIENCY – TFM

	FAN		DynamicSAT+		
Circ.	%FC	%FE	%FC	%FE	%FC Inc.
p44k	55.15	99.40	55.36	99.98	+0.21
p57k	96.36	98.71	97.23	99.99	+0.87
p77k	34.46	67.62	34.46	99.92	+0.00
p80k	94.86	98.58	96.06	100.00	+1.20
p88k	92.33	97.56	94.00	100.00	+1.67
p99k	89.91	95.95	90.91	99.99	+1.00
p177k	76.13	96.56	77.54	99.91	+1.41
p456k	84.17	94.43	86.18	99.50	+2.01
p462k	57.68	97.48	57.95	100.00	+0.27
p565k	94.81	99.44	95.02	100.00	+0.21
p1330k	90.44	99.54	90.57	100.00	+0.13

VI. CONCLUSIONS

Classical structural ATPG algorithms are usually very fast, but have problems to cope with hard-to-test faults which occur more and more frequently in today's complex designs. SAT-based ATPG algorithms are very robust for hard-to-test faults but suffer from the overhead for easy-to-test faults. Several limitations, i.e. loss of structural knowledge, circuit-to-CNF transformation time and completely specified solutions, prevented the application of SAT-based ATPG in industrial practice. Unacceptably high run times were needed for some circuits.

A SAT-based ATPG framework has been proposed in this paper which overcomes the limitations and enables an efficient application in industrial practice. The SAT-based ATPG framework is able to process industrial circuits containing tri-state elements and unknown values. SAT formulations for the most prevalent fault models, i.e. the stuck-at fault model, the transition fault model and the path delay fault model, have been developed. Special attention was paid on the generation of high-quality tests since the demands for these types of tests grows. Furthermore, the problem of the over-specified tests has been solved.

Efficient data structures and methodologies have been presented which boost the performance and strengthen the robustness of the SAT-based ATPG process. The new SAT technique of *Dynamic Clause Activation* (DCA) was proposed for speeding up SAT-based ATPG and, at the same time, retaining the high level of robustness. Efficient circuit-based dynamic learning techniques can be easily integrated into this technique leading to a significant improvement of the robustness of SAT-based ATPG.

Experimental results on large industrial circuits have shown that the proposed techniques are able to significantly reduce the run time of SAT-based ATPG and reduce or even close the run time gap between structural and SAT-based ATPG approaches. At the same time, the number of unclassified faults are reduced to a minimum leading to a very high fault efficiency and significantly increased fault coverage – in particular for the transition fault model. This is especially important to satisfy the high quality demands of the industry and forms a basis for the application of SAT-based ATPG for new complex fault models.

ACKNOWLEDGMENT

The authors would like to thank Görschwin Fey, University of Bremen, and Daniel Tille, Verified Systems International as well as Andreas Glowatz, Friedrich Hapke, René Krenz-Bååth and Juergen Schloeffel, Mentor Graphics Development Germany, for many helpful discussions and steady support.

Parts of this research work were supported by the German Federal Ministry of Education and Research (BMBF) in the Project MAYA under contract number 01M3172B, by the German Research Foundation (DFG) under contract number DR 287/15-1 and by the Central Research Promotion (ZF) of the University of Bremen under contract number 03/107/05.

REFERENCES

BOOK

[DEFT09] R. Drechsler, S. Eggersglüß, G. Fey, and D. Tille. *Test Pattern Generation using Boolean Proof Engines*. Springer, 2009.

JOURNALS

[DEF+08] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1329–1333, 2008.

[DEF+09] R. Drechsler, S. Eggersglüß, G. Fey, J. Schloeffel, and D. Tille. Effiziente Erfüllbarkeitsalgorithmen für die Generierung von Testmustern. *it - information technology*, 51(2):102–111, 2009.

[ED10] S. Eggersglüß and R. Drechsler. Efficient data structures and methodologies for SAT-based ATPG providing high fault coverage in industrial application. 2010. submitted to IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[EFG+10] 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. *Jour. of Electronic Testing: Theory and Applications*, 26(3):307–322, 2010.

[TED10] D. Tille, S. Eggersglüß, and R. Drechsler. Incremental solving techniques for SAT-based ATPG. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(7):1125–1130, 2010.

CONFERENCES

[ED07] S. Eggersglüß and R. Drechsler. Improving test pattern compactness in SAT-based ATPG. In *IEEE Asian Test Symp.*, pages 445–450, 2007.

[ED08] S. Eggersglüß and R. Drechsler. On the influence of Boolean encodings in SAT-based ATPG for path delay faults. In *Int'l Symp. on Multiple-Valued Logic*, pages 94–99, 2008.

[ED09] S. Eggersglüß and R. Drechsler. Increasing robustness of SAT-based delay test generation using efficient dynamic learning techniques. In *IEEE European Test Symp.*, pages 81–86, 2009.

[EFD07a] S. Eggersglüß, G. Fey, and R. Drechsler. SAT-based ATPG for path delay faults in sequential circuits. In *IEEE Int'l Symp. on Circuits and Systems*, pages 3671–3674, 2007.

[EFD+07b] S. Eggersglüß, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schloeffel. Combining multi-valued logics in SAT-based ATPG for path delay faults. In *ACM & IEEE Int'l Conf. on Formal Methods and Models for Codesign*, pages 181–187, 2007.

[ETD09] S. Eggersglüß, D. Tille, and R. Drechsler. Speeding up SAT-based ATPG using dynamic clause activation. In *IEEE Asian Test Symp.*, pages 177–182, 2009.

[ETF+07] S. Eggersglüß, D. Tille, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schloeffel. Experimental studies on SAT-based ATPG for gate delay faults. In *Int'l Symp. on Multiple-Valued Logic*, 2007.

INFORMAL PROCEEDINGS

[EFD+07] S. Eggersglüß, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schloeffel. SAT-based ATPG for path delay faults in industrial circuits. In *IEEE European Test Symposium, Informal Digest of Papers*, 2007.

[ETD08] S. Eggersglüß, D. Tille, and R. Drechsler. Robust tests for transition faults with long propagation paths using Boolean satisfiability. In *IEEE European Test Symposium, Informal Digest of Papers*, 2008.

OTHER REFERENCES

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

[2] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A transitive closure algorithm for test generation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 7, pp. 1015–1028, 1993.

[3] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, 1988.

[4] W. Kunz and D. K. Pradhan, "Accelerated dynamic learning for test pattern generation in combinational circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 5, pp. 684–694, 1993.

[5] J. P. Marques-Silva and K. A. Sakallah, "Dynamic search-space pruning techniques in path sensitization," in *Design Automation Conf.*, 1994, pp. 705–711.

[6] I. Hamzaoglu and J. H. Patel, "New techniques for deterministic test pattern generation," *Jour. of Electronic Testing: Theory and Applications*, vol. 15, no. 1–2, pp. 63–73, 1999.

[7] 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, 2002.

[8] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition fault simulation," *IEEE Design & Test of Computers*, vol. 4, no. 2, pp. 32–38, 1987.

[9] S. A. Cook, "The complexity of theorem proving procedures," in *3. ACM Symposium on Theory of Computing*, 1971, pp. 151–158.

[10] 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.

[11] 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.

[12] J. P. Marques-Silva and K. A. Sakallah, "Robust search algorithms for test pattern generation," in *Int'l Symp. on Fault-Tolerant Computing*, 1997, pp. 152–157.

[13] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Design Automation Conf.*, 2001, pp. 530–535.

[14] 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.

[15] A. Biere, "PicoSAT essentials," *Jour. of Satisfiability, Boolean Modeling and Computation*, vol. 4, no. 2–4, pp. 75–97, 2008.

[16] J. Shi, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schloeffel, "PASSAT: Efficient SAT-based test pattern generation," in *IEEE Annual Symp. on VLSI*, 2005, pp. 212–217.

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

[18] G. Fey, J. Shi, and R. Drechsler, "Efficiency of multi-valued encoding in SAT-based ATPG," in *Int'l Symp. on Multiple-Valued Logic*, 2006, pp. 25–30.

[19] R. D. Eldred, "Test routines based on symbolic logical statements," *Jour. of the ACM*, vol. 6, no. 1, pp. 33–36, 1959.

[20] H. Konuk and T. Larabee, "Explorations of sequential ATPG using boolean satisfiability," in *VLSI Test Symp.*, 1993, pp. 85–90.

[21] J. Savir and S. Patil, "Broad-side delay test," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 1057–1064, 1994.

[22] B. Kruseman, A. K. Majhi, G. Gronthoud, and S. Eichenberger, "On hazard-free patterns for fine-delay fault testing," in *Int'l Test Conf.*, 2004, pp. 213–222.

[23] P. Gupta and M. S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Int'l Test Conf.*, 2004, pp. 1053–1060.

[24] Y. Shao, I. Pomeranz, and S. M. Reddy, "On generating high quality tests for transitions faults," in *IEEE Asian Test Symp.*, 2002, pp. 1–8.

[25] R. Putman and R. Gawde, "Enhanced timing-based transition delay testing for small delay defects," in *VLSI Test Symp.*, 2006, pp. 336–342.

[26] N. Ahmed, M. Tehranipoor, and V. Jayram, "Timing-based delay test for screening small delay defects," in *Design Automation Conf.*, 2006, pp. 320–325.

[27] 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.

[28] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern grading and pattern selection for small-delay defects," in *VLSI Test Symp.*, 2008.

[29] G. L. Smith, "Model for delay faults based upon paths," in *Int'l Test Conf.*, 1985, pp. 342–349.

[30] 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.

[31] A. Krstić and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*. Kluwer Academic Publishers, Boston, MA, 1998.

[32] 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.

[33] A. Gupta, A. Gupta, Z. Yang, and P. Ashar, "Dynamic detection and removal of inactive clauses in SAT with application in image computation," in *Design Automation Conf.*, 2001, pp. 536–541.

[34] M. K. Ganai, L. Zhang, P. Ashar, A. Gupta, and S. Malik, "Combining strengths of circuit-based and CNF-based algorithms for a high-performance SAT solver," in *Design Automation Conf.*, 2002, pp. 747–750.

[35] M. K. Iyer, G. Parthasarathy, and K.-T. Cheng, "SATORI - a fast sequential SAT engine for circuits," in *Int'l Conf. on Computer-Aided Design*, 2003, pp. 320–325.

[36] M. N. Velev, "Encoding global unobservability for efficient translation to SAT," in *Int'l Conf. on Theory and Applications of Satisfiability Testing*, 2004, pp. 197–204.

[37] Z. Fu, Y. Yu, and S. Malik, "Considering circuit observability don't cares in CNF satisfiability," in *Design, Automation and Test in Europe*, 2005, pp. 1108–1113.

[38] S. Safarpour, A. Veneris, and R. Drechsler, "Improved SAT-based reachability analysis with observability don't cares," *Jour. of Satisfiability, Boolean Modeling and Computation*, vol. 5, pp. 1–25, 2008.

[39] G. Fey, T. Warode, and R. Drechsler, "Reusing learned information in SAT-based ATPG," in *Int'l Conf. on VLSI Design*, 2007, pp. 69–76.