

# Improved SAT-based ATPG: More Constraints, Better Compaction

Stephan Eggersglüß\*<sup>†</sup>

Robert Wille\*<sup>†</sup>

Rolf Drechsler\*<sup>†</sup>

\*Institute of Computer Science  
University of Bremen, Germany  
{segg,rwille,drechsle}@informatik.uni-bremen.de

<sup>†</sup>Cyber-Physical Systems  
DFKI GmbH, Bremen, Germany

**Abstract**—Automatic Test Pattern Generation (ATPG) based on Boolean Satisfiability (SAT) is a robust alternative to classical structural ATPG. Due to the powerful reasoning engines of modern SAT solvers, SAT-based algorithms typically provide a high test coverage because of the ability to reliably classify hard-to-detect faults. However, a drawback of SAT-based ATPG is the test compaction ability. In this paper, we propose an enhanced dynamic test compaction approach which leverages the high implicative power of modern SAT solvers. Fault detection constraints are encoded into the SAT instance and a formal optimization procedure is applied to increase the detection ability of the generated tests. Experiments show that the proposed approach is able to achieve high compaction – for certain benchmarks even smaller test sets than the currently best known results are obtained.

## I. INTRODUCTION

The increasing complexity of digital logic in the development of modern circuits leads to significantly increasing test data volume for the post-production test resulting in high test costs. Hence, the industry undertakes immense efforts to reduce the amount of test data. Besides the development of test vector compression schemes, new test pattern reduction techniques are required to avoid the rising test costs [1].

The classical concept of dynamic test compaction [2] is used to increase test compaction for structural *Automatic Test Pattern Generation* (ATPG). A test cube for an initial fault is generated. Then, additional faults are targeted taking the pre-generated test cube or a sensitized path as constraint. By this, the unspecified values of the test cube are assigned in a way that other faults can be detected as well. However, each fault is independently targeted. Several structural techniques have been introduced in order to achieve high compaction. These techniques range from static compaction and ordering techniques [3]–[6] to fault grouping and heuristics [5], [7]–[11] as well as to post-processing techniques [12], [13].

ATPG based on *Boolean satisfiability* (SAT) [14]–[16] has been shown to provide a high fault or test coverage for large industrial circuits [17] since the powerful learning and implication techniques of modern SAT solvers are well suited to generate tests for hard-to-detect faults. Classical structural ATPG approaches typically have problems to cope with these kind of faults as shown in [18].

Recently, SAT-based algorithms have been shown to be well suited to generate high-quality tests targeting for example small delay defects [19]–[21] where usually a significantly increased search space has to be considered. However, a serious drawback of SAT-based ATPG is that the test compaction abilities are typically not as good as the test compaction abilities of structural ATPG algorithms, because SAT solvers act as a black box and provide over-specified solutions. This is very disadvantageous for test compaction. In fact, many specified bits can be substituted by don't cares using post-processing techniques (e.g. [22], [23]). However, this causes additional overhead.

Several techniques have been introduced to increase the test compaction abilities of SAT-based ATPG. The application

of SAT for test compression schemes is treated in [24]. The concept of dynamic compaction was transferred to the SAT-based ATPG domain in [25]. The SAT-based approach presented in [26] works in a different manner. A set of faults  $F$  is chosen and the ATPG generates a test detecting all faults in  $F$  if possible. If at least one fault cannot be detected with one other fault, no test is generated. Therefore, this approach is very run-time intensive and heavily relies on a heuristic which determines which faults could possibly be detected together. The approach presented in [27] proposes a dynamic post-processing technique which compacts a pre-generated testable path set using test relaxation techniques for detecting small delay defects. However, until today, SAT-based methods do not provide a special treatment which ensures the determination of a highly compact test set *during* the solving process.

In this paper, we propose a new basic test formulation following the ideas presented in [28]. Hereby, the test compaction abilities of SAT-based ATPG are enhanced during the actual test generation. In contrast to previous work, the aim of this formulation is to generate an initial test for one fault which is able to detect a large number of other faults without explicitly targeting them. In order to achieve this, the SAT-based ATPG formulation is combined with additional fault detection constraints influenced by fault simulation and path tracing techniques. These constraints determine if a fault can be detected and propagated locally. A *Pseudo-Boolean Optimization* (PBO) procedure is then applied to the ATPG formulation which leverages the powerful SAT solving engine and improves the local fault detection ability of the test.

By this, the generated test is typically able to detect a larger number of faults without targeting any of them explicitly. Experiments show that the proposed ATPG formulation is able to overcome the drawback, i.e. the poor compaction ability. The test set size generated by SAT-based ATPG can significantly be reduced by integrating the proposed formulation into a dynamic test compaction flow. For certain benchmarks, the proposed method even provides a smaller test set than the best known results available in literature – without special post-processing or fault grouping techniques.

The remainder of this paper is structured as follows: The next section briefly reviews the basics on SAT, PBO, and SAT-based ATPG. Section III motivates and sketches the idea of fault detection constraints proposed in this work. Afterwards, details on the proposed CNF formulation are provided in Section IV while its integration in the ATPG flow is described in Section V. Results obtained by an experimental evaluation are reported in Section VI. The paper is concluded in Section VII.

## II. PRELIMINARIES

In order to keep this work self-contained, this section briefly reviews the basics on the applied solving engines as well as on SAT-based ATPG.

### A. SAT and PBO

Solvers for *Boolean satisfiability* (SAT) and *Pseudo-Boolean Optimization* (PBO) are core technologies utilized in

this work for the purpose of ATPG. Both problems are defined as follows:

*Definition 1:* The *Boolean satisfiability problem* determines an assignment to the variables of a Boolean function  $\Phi : \{0, 1\}^n \rightarrow \{0, 1\}$  such that  $\Phi$  evaluates to 1 or proves that no such assignment exists. The function  $\Phi$  is thereby given in *Conjunctive Normal Form* (CNF). A CNF  $\Phi$  is a conjunction of clauses. A clause  $\omega$  is a disjunction of literals and a literal  $x$  is a Boolean variable in its positive ( $x$ ) or negative form ( $\bar{x}$ ).

*Definition 2:* The *pseudo-Boolean optimization problem* determines a satisfying solution for a pseudo-Boolean function  $\Psi : \{0, 1\}^n \rightarrow \{0, 1\}$  which – at the same time – minimizes an objective function  $\mathcal{F}$ . The *pseudo-Boolean function*  $\Psi$  is thereby a conjunction of constraints defined by  $\sum_{i=1}^n c_i x_i \geq c_n$ , where  $c_1, \dots, c_n \in \mathbb{Z}$  and  $x_i$  either is a positive or a negative literal. The *objective function*  $\mathcal{F}$  is defined by  $\mathcal{F}(x_1, \dots, x_n) = \sum_{i=1}^n m_i x_i$  with  $m_1, \dots, m_n \in \mathbb{Z}$ .

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

Accordingly, let  $\Psi = (2x_1 + 3x_2 + \bar{x}_3 \geq 3)(2x_1 + x_2 \geq 2)$  and  $\mathcal{F} = x_1 + x_2 + x_3$ . Then,  $x_1 = 1, x_2 = 0$ , and  $x_3 = 0$  is a solution to the PBO problem satisfying  $\Psi$  and, at the same time, minimizing  $\mathcal{F}$ .

Both, SAT and PBO, are well investigated problems. In the past, efficient solving algorithms (so called *SAT solvers* or *PBO solvers*, respectively) have been proposed (see e.g. [29], [30]). Instead of simply traversing the complete space of assignments, intelligent decision heuristics, powerful learning schemes, and efficient implication methods are thereby applied. In case of PBO, it is also common to translate the respective instance into a sequence of SAT instances in order to efficiently determine a solution [31]. In the following, we apply these techniques as black boxes delivering the solution for the proposed ATPG problem formulations.

### B. SAT-based ATPG

In contrast to classical structural ATPG which works directly on the gate-level netlist, SAT-based algorithms work on a Boolean formula in CNF as defined above. Due to the powerful implication and learning techniques, SAT solvers are well suited to solve hard problem instances. In order to leverage the powerful solving techniques for ATPG, the ATPG problem has to be formulated in CNF [14].

For this purpose, each connection  $x$  of a circuit is assigned a Boolean variable  $x$ . The functionality of each gate  $g$  is transformed into a set of clauses  $\Phi_g$ . The CNF  $\Phi_C$  of the circuit  $C$  is then constructed by a conjunction of the CNF of each single gate of  $C$ , i.e.

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

In order to generate a test for a fault  $f$ , i.e. stuck-at-0 ( $s$ -a-0) or stuck-at-1 ( $s$ -a-1), the circuit CNF  $\Phi_C$  is augmented by additional constraints  $\Phi_F^f$  for fault detection and fault propagation with respect to the specific fault  $f$ . That is, the following CNF formulation results:

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

The CNF  $\Phi_F^f$  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 [15]. The solution space of  $\Phi_{\text{Test}}^f$  includes all possible tests which detects  $f$ . The SAT solver provides one satisfying assignment which can be transformed into a test or proves that no such assignment exists.

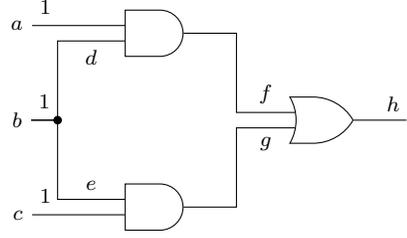


Fig. 1. Exemplary circuit

## III. MOTIVATION AND GENERAL IDEA

In this section, we sketch the general idea of enriching the CNF formulation with additional *Fault Detection Constraints* (FDC) in order to improve the compaction of the resulting test set. To this end, our idea is motivated first by a discussion of the current state-of-the-art.

### A. Motivation

In case a fault  $f$  is testable, the currently applied CNF formulation  $\Phi_{\text{Test}}^f$  for SAT-based ATPG always generates a test detecting  $f$ . Since this test might cover further faults  $f' \neq f$ , *fault simulation* is performed next, i.e. the pattern is simulated and further faults which are detected by it are removed from the fault list. By this, test compaction is improved. However, the amount of additional faults to be detected obviously strongly depends on the eventually determined test pattern.

*Example 2:* Consider the circuit shown in Fig. 1 and additionally assume that SAT-based ATPG is supposed to generate a test detecting the  $s$ -a-0 fault at connection  $h$ . If the ATPG returns the pattern  $a = 1, b = 1, c = 1$ , only one further fault is detected (namely the  $s$ -a-0 fault at connection  $b$ ). If instead SAT-based ATPG returns the test pattern  $a = 1, b = 1, c = 0$ , three more faults at connection  $d$  ( $s$ -a-0),  $f$  ( $s$ -a-0), and  $g$  ( $s$ -a-1) are additionally detected.

As described above, the solution space of  $\Phi_{\text{Test}}^f$  includes all possible test patterns. But so far, solving engines do not distinguish between these possible solutions. Eventually, a test pattern is determined rather randomly depending on heuristics and search strategies basically optimized for efficient SAT-solving rather than compaction. Structural ATPG techniques are able to group faults and to apply special heuristics based on the structure of the circuit to increase compaction. This is not directly possible in SAT-based ATPG, since SAT solvers act as black-boxes (which is part of the reason for their effectiveness for hard problems). Until today, SAT-based methods do not provide a special treatment which ensures the determination of a compact test set *during* the solving process.

In this work, we address this issue and propose an alternative extended CNF formulation which does not only lead to the determination of a test pattern detecting one particular fault, but also enables test pattern generation aiming to detect as many faults as possible. To this end, we are introducing fault detection constraints whose idea is outlined next.

### B. General Idea

Thus far, the CNF formulation for SAT-based ATPG  $\Phi_{\text{Test}}^f$  considers one fault  $f$  only. In order to improve compaction and inspired by some ideas from [32], a consideration of further faults  $f' \neq f$  is proposed. We thereby do not aim for an inclusion of *all* constraints for fault detection and fault propagation for each fault  $f'$  – this obviously would lead to a CNF formulation infeasible to solve. Instead, we propose

to focus on simple necessary conditions which can locally be considered.

More precisely, consider a fault  $f'$  at a connection  $x$ . In order to detect  $f'$

- the fault has to be *activated*, i.e. the value of the connection  $x$  has to be inverse to the fault value, i.e. 0 for s-a-1 and 1 for s-a-0,
- the succeeding *gate*  $h$  has to propagate the fault, i.e. all other inputs of  $h$  have to assume the non-controlling value, and
- there must be a *path* further propagating the fault.

*Example 3:* Consider again the circuit shown in Fig. 1 and the s-a-0 fault at connection  $g$ . In order to detect this fault,  $g$  has to be set to 1 (activating the fault) and connection  $f$  has to be set to 0 (i.e. the non-controlling value of an OR gate). Furthermore, the fault must be propagated further, i.e.  $h$  has to be s-a-0 testable as well.

All these conditions only argue locally and, hence, can easily be added to a CNF formulation. By additionally exploiting the objective function  $\mathcal{F}$  of PBO solvers, the instance can be re-formulated such that the solving engine shall not only determine a test pattern for the currently considered fault  $f$ , but is additionally supposed to satisfy the necessary conditions for as many further faults  $f' \neq f$  as possible.

Note that satisfying these conditions does not ensure that the respective faults  $f'$  indeed are covered by the corresponding test pattern – reconvergences and fault masking still might apply which is why fault simulation still needs to be performed. However, they provide simple and efficient additional objectives which may guide the solving engine in determining better test patterns by justifying important values and eliminating blocked paths. This leads to a significantly higher compaction as the results later in Section VI confirm.

#### IV. PROPOSED CNF FORMULATION

In order to realize the proposed idea, we are following the established SAT-based ATPG flow, i.e. for a given circuit  $C$  and a fault  $f$ , a CNF formulation

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

is created [15]. Besides that, further variables and constraints are added as follows:

*Definition 3:* For each connection  $x$  of a circuit  $C$ , two additional variables  $x_{f0}$  and  $x_{f1}$  are introduced. The variable  $x_{f0}$  is supposed to hold the value 0 if at least one necessary condition for a s-a-0 fault at connection  $x$  is *not* satisfied, while variable  $x_{f1}$  is supposed to hold the value 0 if this is the case for the respective s-a-1 fault. In the following, we call  $x_{f0}$  and  $x_{f1}$  *local fault detection variables*.

Then, constraints incorporating the conditions are added to the instance. Following the general idea, this leads to:

*Definition 4:* For each connection  $x$  of a circuit  $C$  which is the input of a gate with output  $z$ , second input  $y$ , and a controlling value  $cv$ , the following three constraints are created:

- The *activation constraints*  $\Phi_{\text{Activate}}$  check whether the value of  $x$  is the inverse of the fault value, i.e.

$$x = 1 \rightarrow x_{f1} = 0$$

$$x = 0 \rightarrow x_{f0} = 0.$$

$$\begin{aligned} \Phi_{\text{Activate}}: \quad & a = 1 \rightarrow a_{f1} = 0 \\ & a = 0 \rightarrow a_{f0} = 0 \\ & d = 1 \rightarrow d_{f1} = 0 \\ & d = 0 \rightarrow d_{f0} = 0 \end{aligned}$$

$$\begin{aligned} \Phi_{\text{Gate}}: \quad & \dots \\ & a = 0 \text{ (cv)} \rightarrow d_{f1} = 0 \\ & a = 0 \text{ (cv)} \rightarrow d_{f0} = 0 \\ & d = 0 \text{ (cv)} \rightarrow a_{f1} = 0 \\ & d = 0 \text{ (cv)} \rightarrow a_{f0} = 0 \end{aligned}$$

$$\begin{aligned} \Phi_{\text{Path}}: \quad & \dots \\ & f_{f0} = 0 \wedge f_{f1} = 0 \rightarrow a_{f1} = 0 \\ & f_{f0} = 0 \wedge f_{f1} = 0 \rightarrow a_{f0} = 0 \\ & f_{f0} = 0 \wedge f_{f1} = 0 \rightarrow d_{f1} = 0 \\ & f_{f0} = 0 \wedge f_{f1} = 0 \rightarrow d_{f0} = 0 \end{aligned}$$

$$\text{Opt. function: } \mathcal{F} = -a_{f1} - a_{f0} - b_{f1} - \dots$$

Fig. 2. Application of local fault detection variables

- The *gate constraints*  $\Phi_{\text{Gate}}$  check whether all other inputs of the succeeding gate assume the non-controlling value, i.e.

$$y = cv \rightarrow x_{f1} = 0$$

$$y = cv \rightarrow x_{f0} = 0.$$

- The *path constraints*  $\Phi_{\text{Path}}$  check whether a propagating path exists. This is done through the local fault detection variables  $z_{f0}$  and  $z_{f1}$  of the output connection  $z$  of the currently considered gate, i.e.

$$z_{f0} = 0 \wedge z_{f1} = 0 \rightarrow x_{f1} = 0$$

$$z_{f0} = 0 \wedge z_{f1} = 0 \rightarrow x_{f0} = 0$$

*Example 4:* The top of Fig. 2 sketches parts of the respective constraints for all connections of the circuit originally introduced in Fig. 1.

The implications for each connection are transformed into a CNF (resulting in the CNF  $\Phi_{\text{FDC}}$ ) and are added to  $\Phi_{\text{Test}}^f$  as local fault detection constraints.<sup>1</sup> Note that these constraints do not alter the solution space with respect to the tests since they are unidirectional and are only used to set the respective values of the local fault detection variables.

Then, these variables are used in order to guide the solving engine to determine a solution which maximizes the number of satisfied conditions. This is conducted by formulating an optimization function  $\mathcal{F}$ . This function is created including all fault detection variables whose corresponding faults have not yet been detected. Given a set of yet undetected faults  $F = f_1, \dots, f_m$  with  $x^i$  being the connection of fault  $f_i$  ( $1 \leq i \leq m$ ), the optimization function  $\mathcal{F}$  is formulated as follows:

$$\mathcal{F} = (-1) \cdot x_{f_1}^1 + \dots + (-1) \cdot x_{f_m}^m$$

By this, all constants which are associated with an activated fault detection variable are accumulated. Since PBO solvers typically minimize the result, each fault detection variable is associated with a negative variable. It would also be possible to prioritize certain regions by associating higher constant values to certain faults.

*Example 5:* The bottom of Fig. 2 sketches the respective optimization function  $\mathcal{F}$  for the circuit originally introduced in Fig. 1.

Applying a PBO solver to  $\Phi_{\text{Test}}^f \cdot \Phi_{\text{FDC}}$  and the given optimization function  $\mathcal{F}$ , the solver provides the test with the maximum of activated fault detection conditions. This test typically detects a larger number of additional faults as a test generated with the conventional SAT-based ATPG procedure.

<sup>1</sup>These constraints are only added to the circuit part relevant to the detection of  $f$ , i.e. the transitive fanin cone of all structural reachable observation points.

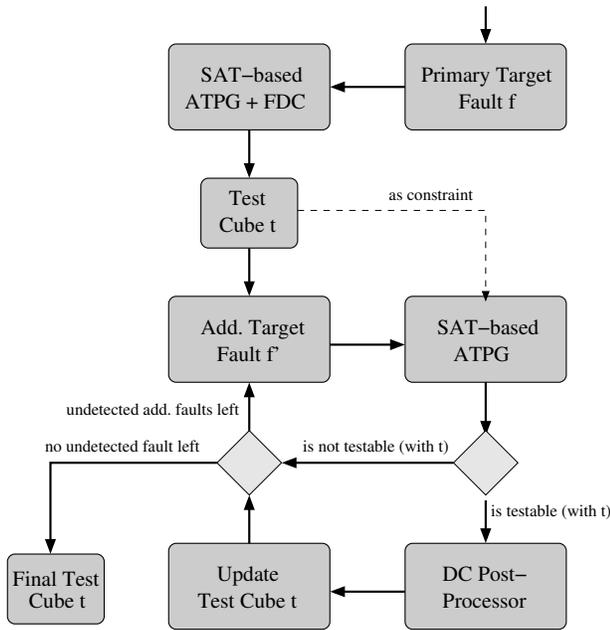


Fig. 3. Proposed SAT-based dynamic compaction flow

## V. INTEGRATION INTO A SAT-BASED ATPG FLOW

In order to show the superiority of the proposed CNF formulation compared to the conventional SAT-based ATPG with respect to the test compaction abilities, we integrated the formulation into a dynamic compaction flow. This section describes the details of the integration as well as the additionally applied optimizations

### A. Dynamic Compaction Flow

The applied dynamic compaction flow is illustrated in Fig. 3. First, a primary target fault  $f$  is chosen. The extended SAT-based ATPG formulation including fault detection constraints is used for test generation. The generated test cube is expected to detect many other yet undetected faults as well. Therefore, no test relaxation or other post-processing steps are performed. After the test cube  $t$  has been extracted from the solution, the additional target fault loop is entered.

In this loop, conventional SAT-based ATPG without fault detection constraints is performed for additional faults  $f'$  taking the pre-generated test cube  $t$  into account. If  $f'$  is testable (with  $t$ ), a post-processing technique is applied to increase the portion of don't cares in the solution. The test cube  $t$  is updated with the necessary values for detecting  $f'$ .

This procedure is repeated for all yet undetected additional faults. After all faults have been processed, the generated test cube  $t$  is added to the test set and the next primary target fault is selected until all faults are detected or proven untestable.

However, determining a solution for the proposed CNF formulation (representing an optimization problem rather than a decision problem) remains the bottleneck of the proposed approach. This is because one arbitrary solution of the solution space is sufficient as a result for the SAT problem. For our optimization problem, the solving engine has to select the best solution among all SAT solutions. Therefore, the run time is expected to grow by applying optimization solvers compared to conventional SAT-based ATPG. Hence, in addition to the integration into a main flow as shown in Fig. 3, two further optimizations are applied as described next.

### B. Leveraging Fault Dominance Relations

The solution space of the optimization problem spans over the variables used in the optimization function  $\mathcal{F}$ . Hence, the

efficiency of the search can be increased by pruning variables from  $\mathcal{F}$ . This is done by leveraging fault dominance relations.

A fault  $f_1$  is said to be *dominated* by a fault  $f_2$  if all tests detecting  $f_1$  also detect  $f_2$ . That means, if a test detects  $f_2$ , the fault  $f_1$  will also be detected. In a fanout-free region of the circuit, the fault dominance relations can be used to collapse the amount of faults to be tested. A test set that detects all faults on the inputs of that region also detects all faults inside that region. This is leveraged during the construction of the optimization function  $\mathcal{F}$ . Only faults on primary inputs or fanout branches are considered for the construction of  $\mathcal{F}$ , since these faults dominate the faults inside fanout-free regions.<sup>2</sup> However, the fault detection conditions for dominated faults still have to be added to the SAT instance to avoid gaps for the path constraints.

*Example 6:* Leveraging the fault dominance relations in the example circuit shown in Fig. 1 restricts the variables in the optimization function  $\mathcal{F}$  to the faults on the primary inputs  $a, b, c$  and the fanout branches  $d, e$ :

$$\mathcal{F} = -a_{f_1} - a_{f_0} - b_{f_1} - b_{f_0} - c_{f_1} - c_{f_0} - d_{f_1} - d_{f_0} - e_{f_1} - e_{f_0}$$

If we assume that the firstly generated test detects the faults  $b_{f_0}, d_{f_0}, f_{f_0}$  and  $g_{f_1}$ , these faults can be removed from  $\mathcal{F}$  in the next ATPG call:

$$\mathcal{F} = -a_{f_1} - a_{f_0} - b_{f_1} - c_{f_1} - c_{f_0} - d_{f_1} - e_{f_1} - e_{f_0}$$

By this, the optimization function will be reduced over time which potentially also reduces the run time.

### C. Two-Stage Flow

Additionally, a two-stage dynamic compaction flow based on the following observations is applied in order to achieve a further improvement:

- The first few test patterns already detect the majority of faults independently from using fault detection constraints or conventional SAT-based ATPG.
- Most of the run time is spent for generating test patterns detecting only few faults.
- If there are many undetected faults, the run time for the optimization solver is high because of the large optimization function.

Therefore, the following two-stage flow is proposed:

- 1) Use conventional SAT-based ATPG until a certain percentage  $P$  of faults is detected.
- 2) After the percentage of  $P$  detected faults has been reached, the proposed SAT-based ATPG formulation is used to detect the remaining faults.

Besides saving run time, it is expected that the two-stage flow is able to achieve a further reduction of tests since it ensures a certain diversity of assignments, because maximizing the local fault detection conditions might lead to the same assignments over and over again.

## VI. EXPERIMENTAL RESULTS

This section presents experimental results for the proposed approach. The SAT-based ATPG procedure as introduced above has been implemented in C++. The solver clasp [30] was used as PBO/SAT solver.<sup>3</sup> The ISCAS'89 and ITC'99 circuits were used as benchmarks for stuck-at test generation.

<sup>2</sup>Equivalent faults are removed earlier during the creation of the fault list.

<sup>3</sup>clasp was awarded *Best Single-Engine Solver in the Hard Combinatorial Track* in the SAT Challenge 2012 and won or was highly ranked in several categories of the PB Competition 2012.

Table I presents the experimental results. The upper part shows results for the ISCAS'89 benchmarks. Column *SAT + DC Post* shows the results of the conventional SAT-based dynamic compaction flow (similar to [25]) extended by a post-processor to substitute unnecessary care bits by don't cares to increase compaction. Column *SAT + FDC* gives the results for the proposed SAT-based ATPG formulation incorporating fault detection constraints (as described in Section V) and column *Two-Stage* presents the results for the proposed two-stage flow starting with conventional SAT-based ATPG and changing to SAT + FDC if the bound  $P$  is reached. Experiments have shown that a good choice for  $P$  is 50. Run time is given in CPU seconds in columns *Time* and the number of generated test patterns are shown in columns *Pat*. The reduction of the two-stage approach compared to the conventional SAT-based ATPG approach is given in column *%Red* and the average portion of care bits in the test patterns are shown in column *%Bits*. A certain amount of don't care bits are required for test compression techniques.

Additionally, the test set sizes of structural ATPG approaches are given for comparison. Column *MinPat* shows the best known results of generated test sets in the literature, i.e. taken from [9], [11], [13], and column *[11]* presents the test set sizes of the most recently proposed structural ATPG approach. These approaches use post-processing techniques such as redundant vector elimination, multiple target test generation, and grouping techniques.

Experimental results for ITC'99 benchmarks are given in the bottom of Table I. To the best of our knowledge, no compaction results for these circuits are reported in literature. Therefore, we compared the proposed approach also to the compaction results of the publicly available structural ATPG *Atalanta* in the newest version [33].

The results show that the proposed dynamic compaction approach is generally even faster than the conventional SAT-based ATPG approach. This is surprising at a first glance, since the optimization procedure typically takes more time than the SAT solving process. However, this run time reduction comes along with a significant reduction of the pattern count. If more faults can be detected with an ATPG call, less time-consuming further ATPG calls are needed. For larger circuits, e.g. b14, roughly half the run time is needed only. The run time can further be reduced for some circuits by the proposed two-stage approach. However, slightly more run time is needed for other circuits.

At the same time, the pattern count can be significantly reduced for all circuits by the FDC formulation. Here, the two-stage approach is generally more effective in producing smaller test sets. The pattern count is lower for most circuits. The highest reduction can be achieved for s15850 (54%).

The results also show that the compaction gap between SAT-based ATPG and structural ATPG can be significantly reduced without any additional techniques as used in structural ATPG. The proposed approach produces better results than the most recent approach for six of the ISCAS'89 benchmarks. SAT-based ATPG is even able to generate a test set for two circuits, namely i.e. s382 and s5378, that is smaller than any set generated by previous approaches. In case of s5378, the smallest known test set size can be reduced by 10%. Compared to *Atalanta*, the proposed approach produces constantly better results than the structural ATPG.

The experiments have shown that the proposed approach is able to increase the test compaction abilities of SAT-based ATPG significantly while, at the same time, reducing the run time. By this, a serious drawback of SAT-based ATPG is eliminated.

A disadvantage of SAT-based ATPG remains. The run time has been constantly higher than, for instance, the approach

in [11] or *Atalanta* (with the exception b15). However, it can also be seen that for larger circuits, i.e. b15, structural ATPG needs more run time to achieve 100% test coverage. This shows the robustness of SAT-based ATPG.

Most of the run time needed for SAT-based ATPG in this compaction flow is needed for SAT instance generation, since for each primary target fault and for each additional target fault a new SAT instance is created. However, these SAT instances are typically very similar to each other. Future work is therefore the development of an incremental SAT scheme, e.g. similar to the techniques presented in [34], which reuses parts of previously generated SAT instances and learned information and is suitable for dynamic compaction.

Furthermore, additional structural compaction techniques suitable for SAT-based ATPG should be identified and transferred to the SAT domain.

## VII. CONCLUSIONS

A new SAT-based ATPG approach has been presented which couples additional fault detection constraints with the conventional SAT-based ATPG formulation. An optimization solver is applied to maximize local fault detection conditions in order to generate tests detecting a larger number of faults without explicitly targeting them. Experiments show the effectiveness of the proposed SAT-based ATPG formulation and the impact on the test set size. The proposed SAT formulation is able to reduce the test set size by up to 54% compared to conventional SAT-based ATPG and, by this, eliminates a traditional drawback of SAT-based ATPG. For some benchmark circuits, the smallest known test set sizes could be further reduced by up to 10%. Additionally, the pattern count reduction goes along with a run time reduction. Future work is the development of suitable additional compaction and fault grouping techniques and further run time reduction by incremental SAT techniques.

## ACKNOWLEDGEMENTS

The authors would like to thank Melanie Diepenbeck for her support during the implementation of this work.

## REFERENCES

- [1] "International technology roadmap for semiconductors - test and test equipment," 2011, <http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Test.pdf>.
- [2] P. Goel and B. C. Rosales, "Test generation and dynamic compaction of tests," in *International Test Conference*, 1979, pp. 189-192.
- [3] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126-137, 1988.
- [4] L. N. Reddy, I. Pomeranz, and S. M. Reddy, "ROTCO: A reverse order test compaction technique," in *Euro ASIC Conference*, 1992, pp. 189-194.
- [5] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A method to generate compact test sets for combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 7, pp. 1040-1049, 1993.
- [6] X. Lin, J. Rajsiki, I. Pomeranz, and S. M. Reddy, "On static test compaction and test pattern ordering for scan designs," in *International Test Conference*, 2001, pp. 1088-1097.
- [7] S. B. Akers, C. Joseph, and B. Krishnamurthy, "On the role of independent fault sets in the generation of minimal test sets," in *International Test Conference*, 1987, pp. 1100-1107.
- [8] G.-J. Tromp, "Minimal test sets for combinational circuits," in *International Test Conference*, 1991, pp. 204-209.
- [9] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "Cost-effective generation of minimal test sets for stuck-at faults in combinational logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 12, pp. 1496-1504, 1995.
- [10] Z. Wang and D. M. H. Walker, "Dynamic compaction for high quality delay test," in *VLSI Test Symposium*, 2008, pp. 243-248.

TABLE I. EXPERIMENTAL RESULTS - TEST SET SIZE

Circ	Previous Approaches/Results				Proposed Approaches				%Red	%Bits
	MinPat	[11] Pat	SAT + DC Time	Post Pat	SAT + FDC Time	Pat	Two-Stage Time	Pat		
s208	27	32	0.18	44	0.12	31	0.11	30	32%	83%
s298	23	24	0.10	33	0.09	27	0.10	27	18%	60%
s344	13	15	0.14	28	0.10	20	0.11	19	32%	56%
s349	13	15	0.13	28	0.09	21	0.15	19	32%	58%
s382	25	25	0.16	30	0.18	24	0.19	26	13%	55%
s386	63	64	0.60	89	0.48	71	0.57	69	22%	80%
s400	24	25	0.20	31	0.14	26	0.16	27	13%	51%
s420	43	68	0.70	69	0.52	51	0.54	49	29%	83%
s444	24	24	0.19	29	0.15	26	0.16	26	10%	53%
s510	54	55	0.79	71	0.72	59	0.81	56	21%	65%
s526	49	52	0.40	59	0.35	56	0.30	52	12%	63%
s641	21	23	1.20	58	0.57	32	0.56	31	47%	54%
s713	21	22	1.43	62	0.79	27	0.86	31	50%	53%
s820	93	96	2.47	113	2.55	98	2.29	95	16%	64%
s832	94	97	2.48	111	2.46	96	2.40	94	15%	64%
s838	75	140	4.40	131	3.18	80	2.30	80	39%	38%
s953	76	78	3.51	107	3.69	83	2.94	83	22%	38%
s1196	113	121	10.05	159	8.41	125	8.38	125	21%	70%
s1238	121	128	12.38	174	10.95	130	9.48	132	24%	70%
s1423	20	26	5.13	60	6.35	32	4.86	31	48%	70%
s1488	101	102	4.68	140	4.28	113	4.13	114	19%	92%
s1494	100	102	4.83	139	4.39	116	4.59	116	17%	92%
s5378	97	101	21.29	118	18.92	87	15.68	88	25%	40%
s9234	105	126	208.79	258	160.91	160	137.30	153	41%	37%
s13207	233	236	366.80	452	224.48	261	199.94	260	42%	30%
s15850	95	98	442.27	250	251.86	116	242.95	115	54%	47%
s35932	10	10	40.93	30	44.85	25	37.83	23	23%	51%
s38417	68	78	314.26	129	229.73	88	221.03	89	31%	25%
s38584	110	118	244.51	233	169.31	142	197.37	141	39%	29%

Circ	Previous Approaches				Proposed Approaches				%Red	%Bits
	Atalanta Time	Pat	SAT + DC Time	Post Pat	SAT + FDC Time	Pat	Two-Stage Time	Pat		
b04	0.22	84	8.11	94	13.13	65	6.84	66	30%	61%
b05	0.10	80	14.28	98	9.84	48	8.78	49	50%	79%
b06	0.01	15	0.02	16	0.02	14	0.02	14	13%	38%
b07	0.01	50	1.84	46	1.87	38	1.55	35	22%	72%
b08	0.01	39	0.76	45	0.58	38	0.54	39	13%	52%
b09	0.01	34	0.47	39	0.57	29	0.34	28	28%	90%
b10	0.01	46	0.45	48	0.43	41	0.39	43	10%	56%
b11	0.03	92	10.53	96	10.41	79	9.64	82	15%	62%
b12	0.05	160	8.50	113	8.10	111	7.46	111	2%	51%
b13	0.01	37	0.34	36	0.26	28	0.31	29	19%	49%
b14	5316.57	943	15358.76	1390	7804.53	708	8656.74	728	47%	69%
b15	19552.98	656	11733.38	867	7355.20	515	7810.32	489	44%	54%

- [11] S. Remersaro, J. Rajski, S. M. Reddy, and I. Pomeranz, "A scalable method for the generation of small test sets," in *Design, Automation and Test in Europe*, 2009, pp. 1136–1141.
- [12] J.-S. Chang and C.-S. Lin, "Test set compaction for combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 11, pp. 1370–1378, 1995.
- [13] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," in *International Conference on Computer-Aided Design*, 1998, pp. 283–289.
- [14] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 1, pp. 4–15, 1992.
- [15] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1167–1176, 1996.
- [16] J. Shi, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schloeffel, "PASSAT: Efficient SAT-based test pattern generation," in *IEEE Annual Symposium on VLSI*, 2005, pp. 212–217.
- [17] S. Eggersglüß and R. Drechsler, "A highly fault-efficient SAT-based ATPG flow," *IEEE Design & Test of Computers*, vol. 29, no. 4, pp. 63–70, 2012.
- [18] 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*, vol. 27, no. 7, pp. 1329–1333, 2008.
- [19] M. Sauer, J. Jiang, A. Czutro, I. Polian, and B. Becker, "Efficient SAT-based search for longest sensitizable paths," in *IEEE Asian Test Symposium*, 2011, pp. 108–113.
- [20] M. Sauer, A. Czutro, I. Polian, and B. Becker, "Small-delay-fault ATPG with waveform accuracy," in *International Conference on Computer-Aided Design*, 2012, pp. 30–36.
- [21] S. Eggersglüß, M. Yilmaz, and K. Chakrabarty, "Robust timing-aware test generation using pseudo-boolean optimization," in *IEEE Asian Test Symposium*, 2012, pp. 290–295.
- [22] K. Miyase and S. Kajihara, "XID: Don't care identification of test patterns for combinational circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 2, pp. 321–326, 2004.
- [23] S. Eggersglüß and R. Drechsler, "Improving test pattern compactness in SAT-based ATPG," in *IEEE Asian Test Symposium*, 2007, pp. 445–450.
- [24] J. Balcerek, P. Fiser, and J. Schmidt, "Techniques for SAT-based constrained test pattern generation," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 37, no. 2, pp. 185–195, 2013.
- [25] A. Czutro, I. Polian, P. Engelke, S. M. Reddy, and B. Becker, "Dynamic compaction in SAT-based ATPG," in *IEEE Asian Test Symposium*, 2009, pp. 187–190.
- [26] S. Eggersglüß, R. Krenz-Bääth, A. Glowatz, F. Hapke, and R. Drechsler, "A new SAT-based ATPG for generating highly compacted test sets," in *IEEE Symposium on Design and Diagnosis of Electronic Circuits and Systems*, 2012, pp. 230–235.
- [27] M. Sauer, S. Reimer, T. Schubert, I. Polian, and B. Becker, "Efficient SAT-based dynamic compaction and relaxation for longest sensitizable paths," in *Design, Automation and Test in Europe*, 2013, pp. 448–453.
- [28] R. Drechsler, M. Diepenbeck, S. Eggersglüß, and R. Wille, "PASSAT 2.0: A multi-functional SAT-based testing framework," in *Latin American Test Workshop*, 2013.
- [29] N. Eén and N. Sörensson, "An extensible SAT solver," in *International Conference on Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, vol. 2919, 2004, pp. 502–518.
- [30] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-driven answer set solving," in *International Joint Conference on Artificial Intelligence*, 2007, pp. 386–392.
- [31] N. Eén and N. Sörensson, "Translating pseudo-Boolean constraints into SAT," *Journal of Satisfiability, Boolean Modeling and Computation*, vol. 2, no. 1–4, pp. 1–26, 2006.
- [32] R. Wille, H. Zhang, and R. Drechsler, "ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization," in *IEEE Annual Symposium on VLSI*, 2011, pp. 120–125.
- [33] H. K. Lee and D. S. Ha, "Atalanta: an efficient ATPG for combinational circuit," Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Tech. Rep., 1993.
- [34] S. Eggersglüß and R. Drechsler, "Efficient data structures and methodologies for SAT-based ATPG providing high fault coverage in industrial application," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 9, pp. 1411–1415, 2011.