MEDIA SuUB Bremen
https://media.suub.uni-bremen.de

Titel/Title: On Acceleration of SAT-Based ATPG for Industrial Designs

Autor*innen/Author(s): Rolf Drechsler; Stephan Eggergluss; Gorschwin Fey; Andreas Glowatz; Friedrich Hapke; Juergen Schloeffel; Daniel Tille

Veröffentlichungsversion/Published version: Postprint

Publikationsform/Type of publication: Artikel/Aufsatz

Empfohlene Zitierung/Recommended citation:

R. Drechsler et al., "On Acceleration of SAT-Based ATPG for Industrial Designs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 7, pp. 1329-1333, July 2008, doi: 10.1109/TCAD.2008.923107.

Verfügbar unter/Available at:
(wenn vorhanden, bitte den DOI angeben/please provide the DOI if available)

10.1109/TCAD.2008.923107

Zusätzliche Informationen/Additional information:

© 2008 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

On Acceleration of SAT-Based ATPG for Industrial Designs

Rolf Drechsler, Stephan Eggersglüß, Görschwin Fey,
Andreas Glowatz, Friedrich Hapke,
Juergen Schloeffel, and Daniel Tille

***Abstract*—Due to the rapidly growing size of integrated circuits, there is a need for new algorithms for automatic test pattern generation (ATPG). While classical algorithms reach their limit, there have been recent advances in algorithms to solve Boolean Satisfiability (SAT). Because Boolean SAT solvers are working on conjunctive normal forms (CNFs), the problem has to be transformed. During transformation, relevant information about the problem might get lost and, therefore, is not available in the solving process. In this paper, we present a technique that applies structural knowledge about the circuit during the transformation. As a result, the size of the problem instances decreases, as well as the run time of the ATPG process. The technique was implemented, and experimental results are presented. The approach was combined with the ATPG framework of NXP Semiconductors. It is shown that the overall performance of an industrial framework can significantly be improved. Further experiments show the benefits with regard to the efficiency and robustness of the combined approach.**

***Index Terms*—Automatic test pattern generation (ATPG), Boolean satisfiability (SAT), formal methods, testing.**

I. Introduction

Guaranteeing that a manufactured chip correctly functions is an important task. Therefore, every chip has to pass a postproduction test during which a set of test patterns is applied to check for functional correctness. These test patterns are usually generated by algorithms for automatic test pattern generation (ATPG). Due to the ever-increasing size of integrated circuits, the size of the problem instances that have to be handled by ATPG algorithms also increases. This results in growing run times of classical (structural) ATPG algorithms such as PODEM [1], FAN [2], SOCRATES [3], and ATOM [4].

As one alternative, ATPG based on Boolean satisfiability (SAT) was first introduced in [5]. Based on the techniques from [6], the efficiency of SAT-based algorithms in the field of ATPG has also recently been shown for large industrial circuits [7], [8]. The efficiency of SAT-based ATPG is significantly driven by the development of powerful SAT engines [9]–[12] in the last ten years. While the early approaches of SAT-based ATPG algorithms could only deal with Boolean logic (see, e.g., [13]), the more recent techniques handle tristate elements and unknown values coming from the environment of a circuit based on four-valued logic (see, e.g., [7]).

Manuscript received November 16, 2007; revised February 5, 2008. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) in the Project MAYA under Contract 01M3172B, by the German Research Foundation (DFG) under Contract DR 287/15-1, and by the Central Research Promotion (ZF) of the University of Bremen under Contract 03/107/05. This paper was recommended by Associate Editor A. Ivanov.

R. Drechsler, S. Eggersglüß, G. Fey, and D. Tille are with the Computer Architecture Group, University of Bremen, 28359 Bremen, Germany (e-mail: drechsle@informatik.uni-bremen.de; segg@informatik.uni-bremen.de; fey@informatik.uni-bremen.de; tille@informatik.uni-bremen.de).

A. Glowatz, F. Hapke, and J. Schloeffel are with Mentor Graphics Development GmbH, 20097 Hamburg, Germany (e-mail: andreas_glowatz@mentor.com; friedrich_hapke@mentor.com; juergen_schloeffel@mentor.com).

Digital Object Identifier 10.1109/TCAD.2008.923107

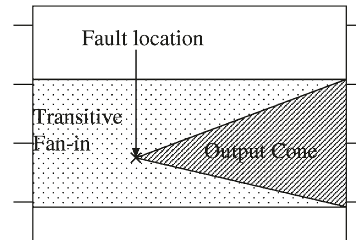

Fig. 1. Extraction of the influenced circuit parts.

SAT-based algorithms are not directly working on the circuit's structure but on an equation in a conjunctive normal form (CNF). Therefore, the problem must be transformed into a CNF. As a disadvantage, structural knowledge about the original problem, which is used by classical algorithms to speed up the search process, might get lost.[1] On the other hand, due to the homogeneity of the representation as a formula, powerful algorithms can be applied to solve the problem. As shown in [6] and [14], the inclusion of structural information leads to a faster solving process. In [15], a highly robust test generation SAT algorithm was proposed. However, in contrast to our approach, this algorithm works on an implication graph and, by this, does not make use of the recent efficient SAT techniques.

In this paper, we present techniques to improve SAT-based ATPG for the stuck-at fault model. The main contributions of this paper are as follows:

1) reduction of the CNF size by modeling the circuit in four-valued logic only where this is necessary;
2) combination of the ATPG engine with a classical approach, resulting in a hybrid proof engine.

All techniques have been implemented and integrated as a prototype in the industrial ATPG framework of NXP Semiconductors. Experimental results on publicly available benchmarks, as well as on several industrial circuits, show the efficiency and robustness of the approach.

II. Preliminaries

In Section II-A, the application of SAT for ATPG is explained (see [5]–[7] for more details). Also, the improvements due to problem-specific knowledge and SAT techniques are briefly reviewed. Then, the use of multiple-valued logic and its Boolean encoding are described in Section II-B.

A. SAT-Based ATPG

Modern Boolean SAT solvers work on the problem represented as a CNF. The classical transformation of a circuit into a CNF representation has been explained in, e.g., [5]. Based on this transformation, the creation of a CNF for a given ATPG problem is briefly explained in the following.

Consider the circuit in Fig. 1. First, the output cone of the fault site is marked by a depth-first traversal on the circuit. This determines all outputs that may be influenced by the fault. The transitive fan-in of these outputs influences the detection of the fault and must be contained in the SAT instance. This knowledge is used to create the SAT instance: A faulty version and a fault-free version of the circuit

[1] In preliminary studies, we also experimented with a circuit SAT solver but did not consistently observe improvements in run time or memory usage.

TABLE I
ENCODING OF THE FOUR-VALUED DOMAIN

X	Encode c_x	c_x^*	Interpretation
0	0	0	Signal X is 0
1	1	0	Signal X is 1
U	1	1	Signal X is unknown
Z	0	1	Signal X is at high impedance

are modeled. Different signal values may only occur in the fault site's output cone. Therefore, this cone is duplicated, whereas the remaining part is shared between both versions; that is, this part is not duplicated.

Finally, to ensure that the fault is propagated to an output, some constraints that encode the D-algorithm [16] are added. This approach was suggested in [6]. Three variables are used for each gate G.

1) G_f denotes the output value of G in the faulty circuit.
2) G_g denotes the output value of G in the correct circuit.
3) $G_d = 1$ iff G is on a D-chain.

A D-chain denotes a path from the fault site to a primary output, where, at each gate, a difference between the correct circuit and the faulty circuit occurs. This notation allows one to introduce additional implications that lead to a fast evaluation of the CNF. These implications are described in detail in [6].

B. Four-Valued Logic

As described in [5], for each signal in the circuit, one Boolean variable is needed to encode its Boolean value. However, in industrial circuits, it is insufficient to model only Boolean values. In these circuits, the two additional (non-Boolean) values U and Z may occur. Considering these two possible states of a signal results in the four-valued logic $L_4 = \{0, 1, U, Z\}$.

First, U describes an unknown value. This may occur, for instance, when ATPG is applied to a circuit C that is a subcircuit of a larger one. Then, it is possible that some of C's primary inputs (PIs) cannot be controlled. This is modeled by fixing those PIs to U. The value U has to explicitly be encoded in the SAT instance because otherwise, the SAT solver would assign Boolean values to such noncontrollable inputs. Second, most industrial circuits contain tristate elements, e.g., bus drivers. Their output can be at high impedance, which is denoted by Z. Note that in [17], an approach is introduced where U is encoded by an ordered pair of Boolean functions, i.e., an interval. However, Z cannot be represented using this method.

Because SAT is only defined on Boolean formulas, each of the four values has to be encoded.[2] The smallest possible encoding can be done by two Boolean variables. There exist 24 possible two-bit encodings of L_4. The chosen encoding determines which clauses are needed to model particular gates. This, in turn, influences the size of the resulting SAT instance and the efficiency of the SAT search.

Further details about choosing an efficient encoding are given in [18]. Table I shows the encoding used in the implementation. This encoding works well on circuits with a large portion of Boolean gates. The variable c_x^* indicates whether a value is Boolean or not.

Finally, to generate the CNF for each single gate type, the flow is similar to the two-valued approach: First, the CNF of a gate G is derived from the characteristic function, which can be constructed using the truth table. Then, it is minimized by ESPRESSO [19]. The minimization must be done only once for each gate type. The result is then stored as a template and used for each circuit. In the following, the function $\varphi_2(G)$ describes the CNF representation of G if G is modeled in Boolean logic, whereas $\varphi_4(G)$ is used if G is modeled in L_4.

[2]Alternatively, a multi-valued SAT solver could be applied. However, to easily incorporate new developments for Boolean SAT solvers, we apply a Boolean encoding.

TABLE II
CLAUSES FOR THE FOUR-VALUED AND GATE

$$(\overline{c}_a + \overline{c}_b + c_c) \cdot (c_a^* + c_b^* + \overline{c}_c^*) \cdot (c_c + \overline{c}_c^*) \cdot (c_a + c_a^* + \overline{c}_c) \cdot (c_b + c_b^* + \overline{c}_c) \cdot (\overline{c}_a^* + \overline{c}_b + c_c^*) \cdot (\overline{c}_a + \overline{c}_b^* + c_c^*) \cdot (\overline{c}_a^* + \overline{c}_b^* + c_c^*)$$

TABLE III
CNF SIZE FOR A TWO-INPUT AND GATE

	CNF description					
	2-valued			4-valued		
Constraint	Cls	Lit	∅len	Cls	Lit	∅len
$C_g \equiv (A_g \cdot B_g)$	3	7	2.3	8	23	2.9
$C_f \equiv (A_f \cdot B_f)$	3	7	2.3	8	23	2.9
$C_d \rightarrow (C_g \neq C_f)$	2	6	3	5	16	3.2
$C_d \rightarrow (D_d + E_d)$	1	3	3	1	3	3
Overhead	1	1	1	2.4	2.8	1.13

As an example, Table II shows the CNF for a four-valued AND gate. Table III denotes the number of clauses (column Cls), the number of literals (column Lit), and the average length of the clauses (column $\varnothing len$) of a two-input AND gate C with inputs A, B and successors D, E in the two- and four-valued models, respectively. The overhead of using L_4 is shown in the last row. Consequently, the use of L_4 results in a significantly larger CNF size, which typically leads to a longer run time. Further details about an efficient representation of gates with more than two inputs can be found in [20].

III. HYBRID LOGIC USAGE

This section presents a fast preprocessing step that achieves a reduced size of the CNF by partially modeling the circuit in Boolean logic instead of L_4. As described in Section II-B, circuits including multiple-valued logic cannot directly be modeled with Boolean logic. Therefore, a Boolean encoding is needed to apply SAT-based algorithms. By applying the encoding, the size of the SAT instance significantly increases.

Table III shows the number of clauses and literals needed to represent a two-valued AND gate in Boolean logic and L_4. Apparently, transforming circuits containing multiple-valued logic results in larger, and often more difficult to solve, SAT instances than transforming circuits containing only Boolean logic.

However, in most industrial circuits, the number of tristate elements is very small compared to the number of Boolean gates. The state of high impedance, i.e., the non-Boolean value Z, can only be assumed in those elements. In the case of propagating the Z-value to a Boolean gate, Z is interpreted as an unknown state, i.e., the non-Boolean value U. Additionally, the unknown state can be assumed by inputs when they are fixed to a non-Boolean value.

For that reason, only those elements that can assume non-Boolean values should be modeled in L_4. An element of the circuit can assume a non-Boolean value iff it satisfies one of the following conditions:

1) the element can handle Z-values, i.e., a tristate element;
2) the element is an input of the circuit and is fixed to a non-Boolean value;
3) the element is contained in the output cone of the aforementioned elements.

All other elements that can only assume Boolean values can be modeled in Boolean logic. Additionally, according to their function, tristate elements are usually located near the outputs. This results in a small output cone of the tristate elements. Furthermore, the percentage of inputs with unknown states is mostly very small. Therefore, only a small subset S of elements has to be modeled in L_4.

Determining the subset S is done in a preprocessing step by analyzing the structure of the circuit and classifying the circuit's elements.

```
1: list<gate> l;
2: set<gate> S;
3: l.add(all_non_Boolean_elements());
4: while !l.empty() do
5:    gate elem = l.first_element();
6:    S.add(elem);
7:    for all succ ∈ elem.all_successors() do
8:       if succ ∉ S & succ ∉ l then
9:          l.append(succ);
10:      end if
11:   end for
12:   for all pred ∈ elem.all_predecessors() do
13:      if pred ∉ S & pred ∉ l then
14:         mark_as_transition(pred);
15:      end if
16:   end for
17:   l.remove(elem);
18: end while
```

Fig. 2. Pseudocode of the structural classification.

The algorithm to determine the elements of the subset S of gates that can assume the non-Boolean values U or Z is presented in Fig. 2. A description of the algorithm follows. For structural classification, a modified depth-first search is applied to the circuit. First, all non-Boolean elements of the circuit (i.e., tristate elements and inputs fixed to U) are identified and stored in a list (line 3). Each element of the list is successively added to the set S (line 6). Every gate in the fan-out cone of those elements must be an element of S because a non-Boolean value can be propagated via this gate. For that reason, the successors of gates in S are added to the list (line 9), from which the current gate is deleted (line 17). When the list is empty, all gates of the fan-out cone of the non-Boolean elements are contained in S. Consequently, the subset S is determined.

Additionally, all direct predecessors p of a gate g in S with $p \notin S$ are marked as *transition* (line 14). At those gates, a transition between the different logics occurs; that is, the output and at least one input of a gate are modeled in different logics. Those transitions must particularly be handled to guarantee consistency. To avoid inconsistencies due to the different encoding, each gate marked as *transition* gets a second variable, which is fixed to 0 (see Table I).[3]

Due to marking of the predecessors, the complexity for the structural classification would be quadratic in the number of gates in the worst case. However, in practice, gates only have k predecessors with $k \ll n$, where n is the number of gates. For that reason, the complexity is given by $\mathcal{O}(n \cdot k)$. The structural classification must be done only once—prior to the ATPG process—and the extracted information can be used for each target. The following example demonstrates how the procedure works.

Example 1: Consider the part of a circuit shown in Fig. 3. The gates k and n are identified to be tristate elements and are therefore added to the set S. All successors of these gates can assume non-Boolean values. Therefore, p and q are also added to S.

Additionally, the gates h, i, l, m, and o are marked as *transition* because they are not in S but have a successor $s \in S$. In Fig. 3, the outgoing lines of those gates that are elements of S are marked bold and have an index of 4, whereas the outgoing lines of those gates that are marked as *transition* have an index of t.

Once all gates are classified, the additional information can be used while generating the CNF. A gate $g_s \in S$ is modeled in L_4, whereas for each gate $h \notin S$, Boolean logic is used. More formally, the CNF Φ_g for each gate g in the circuit can be determined by the following equation:

$$\Phi_g = \begin{cases} \varphi_4(g), & \text{if } g \in S \\ \varphi_2(g), & \text{if } g \notin S. \end{cases}$$

[3]Due to reconvergent paths, there is the possibility that gates that are contained in S are also marked as transition. These are ignored when fixing the second variable to zero.

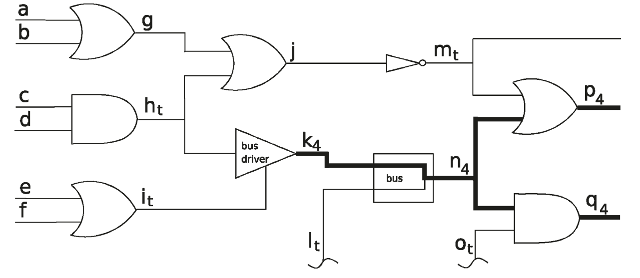

Fig. 3. Structural classification.

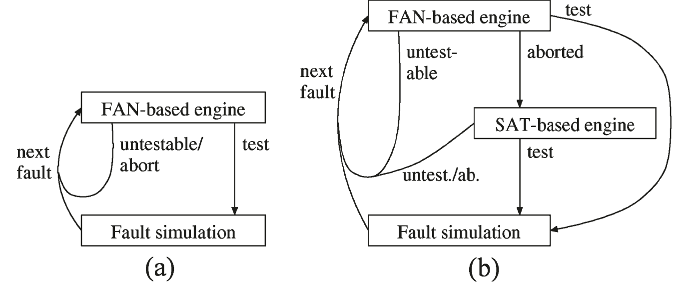

Fig. 4. Flow of the combined approach.

By using L_2 instead of L_4 where possible, the size of the CNF is decreased. The larger the portion of gates that only assume Boolean values, the larger is the reduction in size. In addition, there is nearly no overhead in run time because the preprocessing step has to be executed only once.

IV. Integration in an Industrial Framework

The hybrid approach was integrated as a prototype in the industrial ATPG framework of NXP Semiconductors. The primary goal of the integration is to improve the overall performance of the framework, i.e., decreasing the run time and increasing the number of classified faults. Therefore, the SAT-based approach was combined with the highly optimized classical ATPG tool that has been developed and used at NXP Semiconductors (formerly Philips) during the last 20 years. The tool relies on an enhanced FAN-like algorithm as the core engine. For simplicity, we refer to this as a FAN-based engine in contrast to the SAT-based approach.

In the following, first, a brief description of the ATPG flow is given. At the beginning, random test generation is started to efficiently exclude many easy-to-detect faults. Fig. 4(a) outlines the normal flow. For each fault in the fault list, the test generator is started. If the algorithm classifies the fault as *untestable*, i.e., no test pattern can be found, or as *aborted*, i.e., the time limit is exceeded, the next fault in the list is targeted. In the case of a testable fault, the generated test pattern is passed to a fault simulator, which calculates the additional faults detected by this pattern. These faults are removed from the fault list.

The combination with the SAT-based engine is based on four observations.

1) There exist faults that can easily be classified by the FAN-based engine when SAT needs a long run time, and vice versa.
2) The FAN-based engine directly works on the circuit structure, whereas the SAT-based engine needs additional run time to build the CNF.
3) The FAN-based engine classifies a large number of faults very efficiently.
4) The SAT-based engine performs well on untestable faults and faults that are hard to detect by the FAN-based engine.

Due to these observations, the procedure presented in Fig. 4(b) was chosen for the combination. In this procedure, the SAT-based engine

TABLE IV
CIRCUIT STATISTICS

Circuit	Targets	Elem	Elem 4v	% 4v
p44k	64,105	53,933	0	0
p49k	142,461	64,192	0	0
p80k	197,834	130,997	0	0
p88k	147,742	107,447	7,236	6.73
p99k	162,019	121,524	1,571	1.29
p177k	168,176	226,183	59,091	26.13
p462k	673,949	554,555	73,085	13.18
p565k	1,026,851	1,000,577	112,652	11.26
p1330k	1,516,144	1,352,286	109,690	8.11

is only started for those faults that cannot be classified by the FAN-based engine within a short time limit. This reduces the overhead for building the CNF for faults that are easy to detect by the FAN-based engine. Then, the SAT-based engine classifies additional faults that, in turn, might also help remove other faults from the fault list.

In industrial practice, it is crucial to have a minimal number of nonclassified faults. The integration reduces this number and combines the advantages of both engines. This leads to a faster and more robust ATPG system, as shown in Section V.

V. Experimental Results

The techniques presented in the previous sections have been implemented and integrated as a prototype in the ATPG framework of NXP Semiconductors. In this section, detailed experimental results on the ITC'99 benchmarks, as well as on industrial circuits, are reported and discussed. The industrial circuits have been found to be difficult cases for ATPG.[4] All of the following experiments were run on a Dual Dual-Core 64-bit Xeon (3 GHz, 32-GB RAM, GNU/Linux). MiniSat v1.14 [12] was used as a SAT solver.[5] For each circuit, a time limit of 40 h was set. Details about the decision heuristic used in the presented approach are given in [7].

Table IV shows some information about the industrial circuits. The first column reports the name of the circuit. The succeeding column gives the number of fault targets, i.e., the number of faults that have to be tested, whereas column $Elem$ provides the total number of elements of the circuit. Columns $Elem\ 4v$ and $\%\ 4v$ give the number of elements that can assume non-Boolean values and the percentage, respectively.

First, the use of hybrid logic is empirically evaluated. Information about the average CNF sizes of the experiments is given in Table V. The column named SAT_4v contains information about the four-valued approach (i.e., each signal is modeled using four-valued logic), whereas the column named SAT gives the results for the hybrid approach using two- and four-valued logic. Note that for the sake of comparison, the average sizes of the ITC'99 benchmarks are also given, although they could completely be modeled in Boolean logic.

Due to the typically very small number of gates that can assume non-Boolean values, the average sizes of the instances are significantly reduced. This applies to the number of clauses, as well as to the number of variables. Note that the size of the circuit does not correlate to the average size of the SAT instances. The largest SAT instances do not result from p1330k but from p49k. The results in terms of run time and number of aborted faults are shown in Table VI. Column SAT_4v gives the number of aborted faults ($ab.$) and the run time ($time$) of the four-valued approach. The number of aborted faults and the run time of the hybrid approach can be found in column SAT. The results show that the number of aborted faults of the hybrid approach is—in comparison to the SAT_4v approach—drastically reduced in almost all cases.

[4]This has also been confirmed for other commercial tools.

[5]Due to an observed performance loss in experiments with the newer version MiniSat 2, the older version was kept as the core engine.

TABLE V
AVERAGE INSTANCE SIZES—HYBRID LOGIC USAGE

	SAT_4v			SAT		
Circuit	Vars	Cls	∅len	Vars	Cls	∅len
b14	7,587	32,930	3.03	4,230	11,753	2.34
b15	9,801	45,964	3.25	5,588	15,917	2.38
b17	8,830	39,663	3.14	4,711	13,139	2.35
b18	8,617	37,316	3.08	4,569	12,575	2.33
b20	10,637	46,605	3.04	5,691	15,872	2.34
b21	10,727	46,888	3.03	5,907	16,502	2.34
b22	10,593	46,457	3.04	5,757	16,063	2.35
p44k	60,446	209,001	3.14	30,708	74,435	2.30
p49k	204,130	1,092,143	3.43	101,038	316,294	2.54
p80k	7,483	22,697	3.13	4,211	9,471	2.37
p88k	5,047	15,676	2.88	2,672	6,359	2.34
p99k	5,301	16,139	2.83	2,854	6,504	2.34
p177k	69,908	222,516	2.87	47,087	134,933	2.50
p462k	7,336	22,399	2.77	5,175	14,784	2.39
p565k	4,663	15,638	2.82	2,661	6,956	2.37
p1330k	20,580	62,929	2.81	18,163	55,475	2.54

Due to the heuristic nature of SAT solvers, it can also happen that the run times are longer, although the SAT instance is more compact (see p1330k). However, the experiments show that this rarely happens. Typically smaller instances also directly result in run time savings. In our experiments, improvements of up to a factor of 7 could be observed. Moreover, circuit p49k can be solved, whereas the previous SAT-based approach failed within the given run time limit. The experimental results clearly show that using hybrid logic significantly improves the performance of SAT-based ATPG with respect to run time and number of aborted faults.

The experimental results of the prototypical integration in the industrial ATPG framework are also given in Table VI. The results for the FAN-based engine exist in two different configurations: $FAN(de)$ describes the engine with the default parameters, whereas $FAN(long)$ uses increased backtrack and time limit.

Columns $FAN(de)$ and $FAN(long)$ present the results for the different configurations of the FAN-based engine, whereas columns $FAN(de) + SAT$ and $FAN(long) + SAT$ show the results of the combined approach with the corresponding configuration of the FAN-based engine. First, consider the stand-alone FAN-based engines and the hybrid SAT engine. With regard to the ITC'99 benchmarks, $FAN(de)$ has the smallest run time but the largest number of aborts, whereas SAT has no aborts and a slightly increased run time compared to $FAN(de)$. $FAN(long)$, however, has a significantly increased run time compared to $FAN(de)$ and SAT. Additionally, a large number of faults are aborted by $FAN(long)$. For the industrial circuits, the run times of the FAN-based engines and the hybrid SAT-based engine are rather balanced, with a slight advantage for the FAN-based engines. However, the hybrid SAT-based engine again aborts clearly less faults than the FAN-based engine. No circuit was completely classified by $FAN(de)$, whereas $FAN(long)$ could only completely classify p44k. In contrast, except for p49k, the hybrid SAT-based engine has zero or almost zero aborted faults.

The combined approach $FAN(de) + SAT$ has a drastically reduced number of aborted faults compared to $FAN(de)$. Compared to SAT, the run times are balanced, but $FAN(de) + SAT$ could also completely classify p462k and p1330k. Due to the increased resource limits, the run times of $FAN(long) + SAT$ are longer; in addition, circuit p49k was almost completed. Therefore, $FAN(de) + SAT$ would be the first choice if a good performance is desirable, whereas $FAN(long) + SAT$ is preferable when the number of aborts should primarily be decreased.

In summary, the combination of the SAT-based approach with a classical technique yields the best overall results. The combination of both engines results in significantly improved performance and robustness of the overall ATPG system.

TABLE VI
EXPERIMENTAL RESULTS

Circuit	SAT_4v		SAT		FAN(de)		FAN(long)		FAN(de)+SAT		FAN(long)+SAT	
	ab.	time	ab.	time	ab.	time	ab.	time	ab.	time	ab.	time
b14	0	1:00m	0	0:19m	107	0:11m	7	1:42m	0	0:12m	0	1:24m
b15	0	1:16m	0	0:24m	619	0:11m	318	26:25m	0	0:18m	0	23:02m
b17	0	4:36m	0	2:22m	1382	1:41m	622	56:54m	0	1:58m	0	50:55m
b18	0	27:33m	0	22:30m	740	19:16m	270	41:40m	0	20:34m	0	39:32m
b20	0	2:30m	0	0:56m	225	0:35m	42	7:46m	0	0:44m	0	6:46m
b21	0	2:41m	0	0:59m	198	0:39m	43	6:48m	0	0:43m	0	6:18m
b22	0	3:49m	0	1:35m	284	1:07m	52	9:34m	0	1:14m	0	9:58m
p44k	0	2:18h	0	26:01m	12	4:58m	0	4:59m	0	5:55m	0	8:03m
p49k	-	-	77	1:43h	3770	2:06h	162	2:38h	74	1:55h	2	2:49h
p80k	1	42:58m	0	9:43m	218	34:55m	21	39:13m	0	39:38m	0	57:20m
p88k	0	11:41m	0	9:33m	195	9:13m	38	12:40m	0	10:27m	0	18:56m
p99k	0	8:41m	0	6:50m	1398	6:02m	512	1:16h	0	7:25m	0	1:02h
p177k	941	10:28h	0	1:19h	270	16:06m	47	20:03m	0	19:03m	0	31:23m
p462k	129	3:31h	6	2:16h	1383	1:34h	423	2:07h	0	1:51h	0	3:07h
p565k	0	2:23h	0	2:23h	1391	2:21h	85	2:47h	0	2:47h	0	4:29h
p1330k	1	4:58h	1	5:05h	889	4:15h	144	4:28h	0	5:00h	0	7:30h

VI. CONCLUSION

Using structural information while transforming large industrial circuits into a CNF significantly reduces the size of the SAT instances for ATPG. As a consequence, the SAT solver needs less resources, which boosts the performance of the SAT-based ATPG approach. Furthermore, the integration of the SAT-based engine into the industrial ATPG framework of NXP Semiconductors improves the overall performance of the framework and leads to a fast and robust ATPG system.

REFERENCES

[1] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, no. 3, pp. 215–222, Mar. 1981.

[2] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1137–1144, Dec. 1983.

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

[4] I. Hamzaoglu and J. Patel, "New techniques for deterministic test pattern generation," *J. Electron. Test.—Theory and Applications*, vol. 15, no. 1/2, pp. 63–73, Aug.–Oct. 1999.

[5] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[6] P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 9, pp. 1167–1176, Sep. 1996.

[7] J. Shi, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel, "PASSAT: Efficient SAT-based test pattern generation for industrial circuits," in *Proc. IEEE Annu. Symp. VLSI*, 2005, pp. 212–217.

[8] J. Shi, G. Fey, R. Drechsler, A. Glowatz, J. Schlöffel, and F. Hapke, "Experimental studies on SAT-based test pattern generation for industrial circuits," in *Proc. Int. Conf. ASIC*, 2005, pp. 967–970.

[9] J. Marques-Silva and K. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Trans. Comput.*, vol. 48, no. 5, pp. 506–521, May 1999.

[10] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Proc. Des. Autom. Conf.*, 2001, pp. 530–535.

[11] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust SAT-solver," in *Proc.* Des. Autom. Test Eur., 2002, pp. 142–149.

[12] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proc. SAT*, 2003, vol. 2919, pp. 502–518.

[13] J. Marques-Silva and K. Sakallah, "Robust search algorithms for test pattern generation," Dept. Informatics, Tech. Univ. Lisbon, Lisbon, Portugal, Tech. Rep. RT/02/97, Jan. 1997.

[14] P. Tafertshofer, A. Ganz, and K. Antreich, "IGRAINE—An implication graph-based engine for fast implication, justification, and propagation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 8, pp. 907–927, Aug. 2000.

[15] E. Gizdarski and H. Fujiwara, "SPIRIT: A highly robust combinational test generation algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1446–1458, Dec. 2002.

[16] J. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. Develop.*, vol. 10, no. 4, pp. 278–291, Jul. 1966.

[17] R. T. Stanion, "Circuit synthesis verification method and apparatus," U.S. Patent 6 056 784, May 2, 2000.

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

[19] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Univ. California, Berkeley, CA, Tech. Rep. No. UCB/ERL M92/41, 1992.

[20] D. Tille, G. Fey, and R. Drechsler, "Instance generation for SAT-based ATPG," in *Proc. IEEE Workshop Des. Diagnostics Electron. Circuits Syst.*, 2007, pp. 153–156.