A New SAT-based ATPG for Generating Highly Compacted Test Sets

Stephan Eggersglüß*§ René Krenz-Bååth† Andreas Glowatz‡ Friedrich Hapke‡ Rolf Drechsler*§

§ Cyber-Physical Systems, DFKI Bremen
*University of Bremen,
28359 Bremen, Germany
{segg,drechsle}@informatik.uni-bremen.de

†Hochschule Hamm-Lippstadt
59063 Hamm, Germany
rene.krenz-baath@hshl.de

‡Mentor Graphics
21079 Hamburg, Germany
{andreas_glowatz,friedrich_hapke}@mentor.com

***Abstract*—The test set size is a highly important factor in the post-production test of circuits. A high pattern count in the test set leads to long test application time and exorbitant test costs. We propose a new test generation approach which has the ability to reduce the test set size significantly. In contrast to previous SAT-based ATPG techniques which were focused on dealing with hard single faults, the proposed approach employs the robustness of SAT-solvers to primarily push test compaction. Furthermore, a concept is introduced how the novel technique can be flexibly integrated into an existing industrial flow to reduce the pattern count. Experimental results on large industrial circuits show that the approach is able to reduce the pattern count of up to 63% compared to state-of-the-art dynamic compaction techniques.**

I. Introduction

The test set size is of high importance for the post-production test of circuits. Due to the limited storage capacities of automatic test equipment and long test application time, a high pattern count signifies high test costs. According to the *International Technology Roadmap for Semiconductors* (ITRS) [1], the test data volume will increase significantly in the next years. This development is caused by an increasing complexity of the applied faults models as well as a growing demand for higher test quality. Despite significant advances of test compression approaches [2], it is highly important to improve the compaction of traditional scan patterns.

In order to achieve highly compacted test sets, each test pattern has to detect as many faults as possible. Compaction methods can roughly be classified in two different categories: static compaction and dynamic compaction. Static compaction techniques are applied to already generated tests. Here, several patterns are merged into one test pattern by utilizing unspecified bits [3].

Dynamic compaction methods [3]–[5] are more effective but also computationally more intensive. These methods are integrated into the *Automatic Test Pattern Generation* (ATPG) process. The unspecified bits of a generated test cube are used to detect additional faults. Here, the previously generated test (cube) is given as constraint to the ATPG algorithm which then tries to generate a test for both faults. By this, the search space is incrementally restricted and the ATPG algorithm may not be able to generate a common test for the targeted set of faults although there exists one. In spite of this limitation, effective techniques and heuristics were developed based on this principle, e.g. [6]–[8]. Today dynamic compaction methods are well established in the industrial practice.

The approaches presented in [9], [10] introduced an additional compaction concept called *Multiple Target Test Generation* (MTTG). The main idea is to target multiple faults simultaneously to generate a test. This is done by multiple path sensitization using a structural ATPG algorithm, i.e. PODEM [11] or FAN [12]. The procedure will find one test which detects these faults if such a common test exists. However, the proposed methods are computationally highly intensive and not applicable to large circuits as reported in [5]. Therefore, the approach in [10] is only applied to replace existing tests in a post-generation phase while the method in [9] uses a simplified problem description to cope with the high complexity of MTTG.

In this paper, we present a new approach based on the multiple target principle. In contrast to the previous methods, the novel method is based on *Boolean Satisfiability* (SAT) and such can benefit from the recent advances in this area. Recently, SAT-based ATPG [13], [14] has been attracted much attention as a robust complement to classical structural ATPG providing a significantly increased fault coverage for industrial circuits [15]. In particular, SAT-based algorithms perform very well on hard problems due to their inherent learning features. This motivates the use of SAT for MTTG since it is a computationally very hard problem and previous approaches had problems to cope with the complexity of MTTG. In particular, the following contributions are introduced:

- New MTTG SAT formulation – It is shown how the MTTG problem can be formulated as a SAT problem. The new SAT formulation leverages the fact that parts of the circuit can be shared for multiple faults and learned information which boosts the search and can be used for all targeted faults.
- Integration into dynamic compaction flow – Besides the SAT formulation for MTTG, we present a method how to flexibly integrate SAT-based MTTG into an existing dynamic compaction flow that it can be easily used in industrial practice. As a result of the integration, the approach is able to produce test sets with a significantly reduced pattern count for stuck-at as well as transition

faults.

The rest of the paper is organized as follows. The next section discusses preliminaries. In particular, a short introduction to dynamic compaction and SAT-based ATPG is presented. Section III shows how MTTG is formulated as a SAT problem and Section IV describes how SAT-based MTTG can be effectively integrated into a dynamic compaction flow. Experimental results on large circuits which show the applicability in an industrial flow are presented in Section V. Section VI concludes this paper.

II. Preliminaries

Section II-A introduces the applied dynamic compaction flow. A brief introduction to SAT-based ATPG is given in Section II-B.

A. Dynamic Compaction

In contrast to static compaction techniques which operate on previously generated test patterns, dynamic compaction methods are integrated into the ATPG process. The main idea is that bits in the test which are left unspecified by the ATPG, i.e. assigned with the don't care value X, are used for detecting other faults as well.

Algorithm 1 Dynamic Compaction Flow

```
FaultList F_Prim = AllUndetectedFaults();
TestSet T = ∅;
while F_prim ≠ ∅ do
   Fault f_prim = PopFault(F_Prim);
   Test t = DoATPG(f_prim);
   if t == ∅ then
      continue;
   end if
   FaultList F_Add = AllUndetectedFaults();
   while F_Add ≠ ∅ do
      Fault f_add = PopFault(F_add);
      t = DoATPG(f_add, t);
   end while
   T = T ∪ t;
end while
return T;
```

Algorithm 1 shows the pseudo-code of a dynamic compaction procedure. First, some undetected primary target fault f_{prim} is chosen from the fault list and a test is generated for f_{prim}. If a test is found, then the procedure loops over all yet undetected faults and calls the ATPG with the previously computed test assignments as constraints. In other words, the ATPG can only assign the remaining unspecified inputs to generate a test for the additional target fault f_{add}.

This incremental limitation of the search space might prevent finding a test for f_{add}. On the other side, this is advantageous for traditional ATPG algorithms since they do not perform well on hard reasoning problems. In contrast, SAT-based ATPG algorithms behave very robust on such problems. Previous publications on SAT-based ATPG mainly focused on dealing with hard single-fault problems. In this paper, we firstly employ the robustness of SAT to push test pattern compaction. The following example clarifies the issue discussed above.

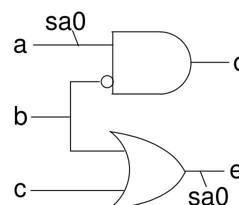

Fig. 1. Dynamic Compaction Example

Consider the example given in Figure 1, where the stuck-at-0 (sa0) fault at output e is assumed to be the primary target fault f_{prim} and an sa0 fault at the input a is an additional target fault f_{add}. If the ATPG computes a test for f_{prim}, where input b is assigned to '1', then there exists no additional assignment which would allow the detection of f_{add} since any test to detect f_{add} would require the assignment $b = 0$. The proposed SAT-based technique allows to concurrently generate a common test for f_{prim} and f_{add}. If such a test would not exist, then the SAT solver would indicate that the problem instance is unsatisfiable. In other words, if a common test for a set of faults exists, then it is guaranteed that the proposed algorithm will find it. In contrast to that traditional dynamic compaction approaches do not assure the successful compaction of these faults. The successful compaction of the same set of faults depends on several parameters such as ATPG-internal decision heuristics or the ordering of the faults list.

B. SAT-based ATPG

Classical ATPG algorithms work on a structural gate level netlist and apply efficient implication procedures to generate tests. In contrast to these structural ATPG algorithms, SAT-based ATPG algorithms work on a Boolean formula in *Conjunctive Normal Form* (CNF). A CNF Φ is a conjunction of clauses. A clause ω is a disjunction of literals, where a literal is a Boolean variable in positive (x) or negative ($\overline{x}$) form. A CNF Φ is satisfied if all clauses in Φ are satisfied and a clause ω is satisfied if at least one literal in ω is satisfied. A positive literal is satisfied if the corresponding variable is assigned with 1 while a negative literal is satisfied with an assignment 0.

Due to the homogeneity of the Boolean formula, very efficient implication strategies as well as effective conflict-based learning techniques can be applied and are incorporated in modern SAT solvers, e.g. [16]. In order to apply robust SAT solvers, the ATPG problem has to be formulated as a SAT instance in CNF [17].

Each signal s in the circuit $\mathcal{C}$ is assigned a Boolean variable x_s which represents the state of the signal. Then, each gate $g \in \mathcal{C}$ is transformed into a set of clauses by using the characteristic function of the gate type. Table I shows the CNF representation for the basic gate types. The CNF $\Phi_{\mathcal{C}}$ for the complete circuit is then composed by the conjunction of the clauses of all gates

TABLE I
CNF FOR BASIC GATES WITH OUTPUT c AND INPUTS a, b

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

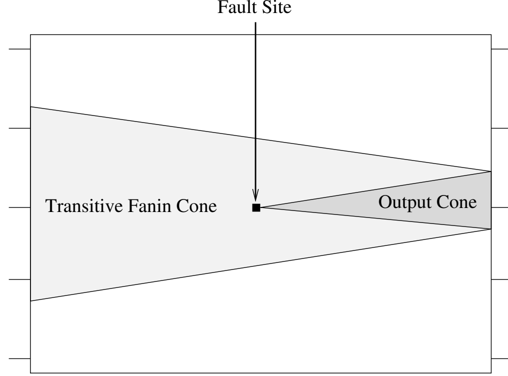

Fig. 2. Illustration of Fault f

$g_1, \ldots, g_n$:

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

In order to generate a test for fault f, the CNF of the circuit $\Phi_{\mathcal{C}}$ has to be augmented by fault-specific constraints Φ_f. These constraints include the fault site, a copy of the faulty output cone and fault propagation conditions, e.g. a D-chain encoding [17]. Note that not the complete circuit CNF has to be used but only those parts which are able to structurally influence the fault activation and propagation. This is illustrated in Figure 2 and in the following denoted by $\Phi_{\mathcal{C}}^f$.

Therefore, the complete CNF Φ_{test}^f for test generation for fault f is given by the following formula:

$$\Phi_{\text{test}}^f = \Phi_{\mathcal{C}}^f \cdot \Phi_f$$

The CNF Φ_{test}^f is then given to a SAT solver which solves the formula. If no solution exists, i.e. the formula is unsatisfiable, the fault is untestable. Otherwise, the solver returns a solution from which the test can be easily extracted.

III. SAT-BASED MULTIPLE TARGET TEST GENERATION

This section presents a new SAT formulation which guarantees to generate a test which detects multiple target faults if such a test exists.[1]

In order to detect one fault f, two different conditions have to be satisfied: fault activation and fault propagation (denoted above together with Φ_f). The fault activation condition affects only the correct copy of the circuit. Here, a difference between the correct and faulty circuit has to be invoked. This is simply done by setting the value of the fault site of the correct circuit to the opposite value of the fault's value, i.e. 0 (1) for detecting a stuck-at-1 (stuck-at-0) fault and denoted by Φ_{act}^f. The difference has then to be propagated to an observation point. This is done by duplicating the output cone of the fault site (Φ_{faulty}^f) and adding fault propagation conditions (Φ_{prop}^f). The propagation conditions involve variables from the correct part of the circuit as well as from the faulty part of the circuit since they are responsible to establish a difference between both parts.

When multiple faults $f_1, \ldots, f_n$ are targeted, each fault f_i has to be activated and propagated to an observation point. A conjunction of the SAT instances for all faults is not possible, since different tests for each fault would be generated. In order to ensure that one test is generated which detects $f_1, \ldots, f_n$, the correct copy of the circuit has to be shared for all faults. This is necessary because the test is extracted from the correct copy of the circuit. However, each fault has to be propagated separately since the single stuck-at (transition) fault model is assumed. Otherwise, the faults may possibly mask each other as assumed in the multiple stuck-at (transition) fault model.

The correct shared copy of the circuit $\mathcal{C}$ for faults $f_1, \ldots, f_n$ including the fault activation conditions is composed as follows:

$$\Phi_{\mathcal{C}}^{f_1,\ldots,f_n} = (\Phi_{\mathcal{C}}^{f_1} \cup \ldots \cup \Phi_{\mathcal{C}}^{f_n}) \cdot (\Phi_{\text{act}}^{f_1} \cup \ldots \cup \Phi_{\text{act}}^{f_n})$$

The faulty circuitry is represented by the following formula. Note that no parts of the CNF are shared between the faults to ensure separate detection:

$$\Phi_{\text{faulty}}^{f_1,\ldots,f_n} = \Phi_{\text{faulty}}^{f_1} \cdot \ldots \cdot \Phi_{\text{faulty}}^{f_n}$$

The fault propagation conditions are composed analogously as follows:

$$\Phi_{\text{prop}}^{f_1,\ldots,f_n} = \Phi_{\text{prop}}^{f_1} \cdot \ldots \cdot \Phi_{\text{prop}}^{f_n}$$

However, each clause set $\Phi_{\text{prop}}^{f_i}$ uses the variables from the corresponding faulty output cone from f_i and from the correct circuit $\mathcal{C}$. Finally, the complete SAT instance $\Phi_{\text{test}}^{f_1,\ldots,f_n}$ for MTTG for multiple faults $f_1, \ldots, f_n$ is as follows:

$$\Phi_{\text{test}}^{f_1,\ldots,f_n} = \Phi_{\mathcal{C}}^{f_1,\ldots,f_n} \cdot \Phi_{\text{faulty}}^{f_1,\ldots,f_n} \cdot \Phi_{\text{prop}}^{f_1,\ldots,f_n}$$

This is illustrated in Figure 3 with an example for three faults f_1, f_2, f_3. The shared correct circuit consists of the faulty output cones of each fault (marked through dashed triangles) and their transitive fanin cone. All faults have to be activated simultaneously. Each faulty output cone is duplicated and connected to the correct circuit at the border that the correct signals of the side inputs can be fed into the faulty output cone. Even when the output cones share parts of the correct circuits, no parts are shared in the faulty circuitry. Additionally, the propagation constraints are included.

The most important advantage of using SAT-based algorithms is the inherent application of its conflict-based learning abilities. The robustness and the ability to solve hard problem instances stem from this feature. Sharing the correct circuit which is often the largest part of the SAT instance means that all information learned in this part can be used for all

[1] Note that only the single stuck-at and single transition fault model is assumed, i.e. the test is guaranteed to detect these faults if they occur independently from each other.

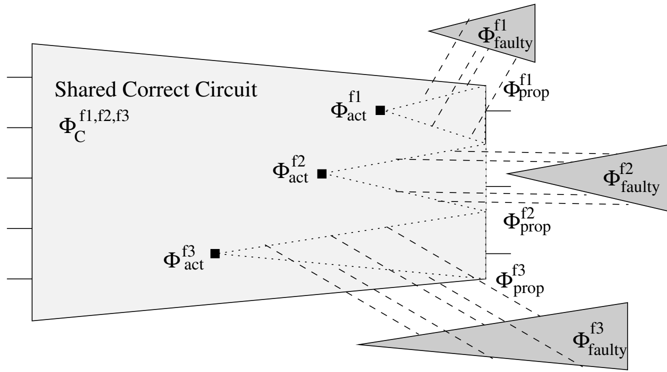

Fig. 3. SAT instance for Multiple Target Test Generation

faults and has not to be calculated for each fault separately as typically done by structural approaches using multiple path sensitization. By this, MTTG is significantly improved compared to the earlier approaches.

IV. Integration into Dynamic Compaction

This section describes how SAT-based MTTG can be effectively integrated into the dynamic compaction flow in industrial practice. Targeting each fault combination is far too time consuming to be used in practice. Furthermore, SAT-based MTTG is more complex than classical dynamic compaction. Often, a large number of faults have a good random testability. There is no need to use MTTG for these kind of faults, since it is likely that these faults can be efficiently compacted by the classical dynamic compaction flow.

Therefore, a combination of the classical dynamic compaction flow (as described in Algorithm 1) and SAT-based MTTG is proposed. The complete procedure is shown in Algorithm 2. At first, the classical dynamic compaction flow is started with the goal to prune a large number of faults which are easy to detect with only few test patterns. The number of additional targets detected by one test serves as a break criterion for the classical flow. When this number falls below a certain limit (denoted by N), the classical flow is aborted and the MTTG stage is started with the aim to compact the remaining faults more intensively. This procedure is based on the observation that classical dynamic compaction often needs long run times to generate tests for the remaining faults and a large part of the test set is used to detect only few faults.

In the MTTG stage, a set G of *NUM* faults is chosen (denoted by MTTG fault list in the following). The faults in G are treated in a similar way than the faults in the additional target loop in the classical dynamic compaction flow. At first, the primary target fault is taken out of G and added to the internal MTTG fault set H. If the fault is redundant, it is removed from the MTTG fault set. Otherwise, the fault is kept. Then, the subsequent fault in the MTTG fault list G is added to H and the MTTG is started for all faults in H. If all faults in H can be tested by one test, the test is temporary stored and the next fault is added to H. Note that the temporarily stored test is not used as constraint. If no test can be found for multiple faults, the fault which was added lastly is removed from H and it is proceeded with adding the subsequent fault.

Algorithm 2 MTTG Dynamic Compaction Flow

```
TestSet T = ∅;
{Do Classical Dynamic Compaction until break criterion N is reached.}
T = ClassicalDynamicCompaction(N);
FaultList F = AllUndetectedFaults(T);
while F ≠ ∅ do
   {Get a set of NUM not easy to detect faults which are likely to be combined.}
   FaultList G = GetPartialFaultList(F, NUM);
   FaultSet H = ∅;
   Test t1;
   while G ≠ ∅ do
      Fault f = PopFault(G);
      H = H ∪ f;
      {Do MTTG for all faults ∈ H and temporarily save the resulting test if exists}
      Test t2 = DoMTTG(H);
      if t2 == ∅ then
         H = H \ f;
      else
         t1 = t2;
      end if
   end while
   {Give the test calculated by MTTG as constraint to the classical additional target loop}
   t = DoAdditionalTargetLoop(F \ G, t1);
   T = T ∪ t1;
   F = AllUndetectedFaults(T);
end while
return T
```

By this, the procedure collects all faults in the MTTG fault list which can be detected by one test according to the given fault ordering. Eventually, the last generated test is given to the classical additional target loop as constraint. That means, the test is treated as a generated primary target fault test and the classical dynamic compaction procedure tries to leverage all remaining unspecified bits in order to detect as many faults as possible.

The effectiveness of this procedure depends strongly on the compilation of the MTTG fault list G, since these are the faults which are to be compacted intensively. In the current version, the list G is composed using a sophisticated structural heuristic, which has shown to be very effective.

The following example demonstrates the MTTG procedure:

Example 1: Given a list of undetected faults

$$F = f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}$$

At first, a set of *NUM* $= 5$ faults is chosen and ordered in the MTTG fault list G:

$$G = f_1, f_2, f_3, f_4, f_5$$

The first fault f_1 is added to the fault set H and test generation is started for f_1. Since f_1 is testable (Test t_1), the second fault f_2 is added to H and MTTG is used for both faults in H. The resulting test t_2 detecting both faults f_1, f_2 is temporarily stored. Then, f_3 is added to H and MTTG is started for f_1, f_2, f_3. It turned out that there is no test for all three faults. Therefore, f_3 is removed from H.

As a next step, f_4 is added to H and a test is generated for all faults included in H: f_1, f_2, f_4. The resulting test t_3 is stored. Fault f_5 is then added to H ($H = \{f_1, f_2, f_4, f_5\}$) and MTTG is started to generate a test for all faults in H. Since this failed, i.e. the SAT instance is unsatisfiable, and all faults in G were processed, test t_3 is used as the final MTTG test and given to the additional target loop. Assume that t_3 can be augmented to detect fault f_7 and f_8 using classical dynamic compaction. The remaining fault list F is as follows:

$$F = f_3, f_5, f_6, f_9, f_{10}$$

The fault list is then processed the same way as described above until all faults are detected.

V. Experimental Results

This section presents the experimental results of the proposed approach. At first, the experimental setup is described in Section V-A. Section V-B gives the concrete compaction results.

A. Experimental Setup

The proposed approach was implemented in C++ and integrated into an industrial test environment The underlying SAT solving engine is MiniSat v1.14 [16]. The experiments were conducted on an AMD Athlon 64 with 4096 MB RAM and a processor speed of 3000 MHz running GNU/Linux. The approach was evaluated on ITC'99 benchmarks as well as on large industrial circuits for the stuck-at as well as the transition fault model using launch-on-capture. The name of the industrial circuit roughly denotes the size of the circuit, i.e. l2787k contains about 2.7 million elements. The proposed approach is compared to a highly competitive dynamic compaction flow used in industrial practice. This flow was particularly optimized for producing very small test sets and uses additional techniques such as reverse order fault simulation to reduce the pattern count. Additionally, we provide experimental results for the ITC'99 benchmark set obtained by the well-known Atalanta ATPG-tool [18]. Note that the backtrack limit of the Atalanta-tool has been adapted to achieve similar fault coverages in comparison to the industrial flow.

As described above, the MTTG approach is integrated such that, at first, the classical flow is used to prune a large number of easy-to-detect faults. As a break condition for the classical flow, we chose a number of additional targets $N = 20$. That is, when the number of additional targets found by the ATPG falls below the limit of $N = 20$, the MTTG stage is started. As size of the MTTG fault list G, we chose the parameters, *NUM* $= 5$, *NUM* $= 10$ and *NUM* $= 50$.

B. Compaction Results

Table II shows the experimental results for the stuck-at fault model. The second and third column provide the results achieved by the Atalanta tool. The industrial benchmarks were not run on Atalanta since it does not support any tri-state elements. Column *DynComp* shows the compaction results, i.e. the number of tests, and the run times generated by the classical dynamic compaction flow only.

Column *MTTG stage* shows the overall results after the MTTG stage, as described in Section IV, in terms of run time and test set size. In the columns *NUM = X*, the pattern count of the corresponding configuration is shown while the corresponding run time is given in column *Time*. Generally, increasing the size of the MTTG fault list results also in a longer run time since more SAT instances have to be built and to be solved.

The results for *NUM* $= 5$ and for *NUM* $= 10$ show that the approach is able to significantly reduce the pattern count already with small MTTG fault lists, see for instance l49k and l2787k, where traditional dynamic compaction does not succeed to reduce the pattern count significantly. However, the pattern count increases slightly for some circuits, e.g. l57k and l565k. This is due to the heuristic nature of the fault selection for the MTTG. It can also be observed that the benchmark circuits behave differently than the industrial circuits. Here, the proposed techniques do not seem to be effective and are very time consuming.

Increasing the size of the MTTG fault list (*NUM* $= 50$) solves the limitation. Here, the pattern count for all circuits are reduced compared to the classical dynamic compaction flow. Column *%NUM = 50* shows the reduction of the test set size compared to the classical flow in percent. For some circuits, e.g. l57k and l565k, the reduction rate is rather small. For these circuits, a significant increase of the parameter *NUM* did not result in much better reduction rates. Therefore, we believe that the test set size for these circuits is already close to minimum.

The highest reduction rate can be achieved for the circuits l88k and l2787k with only 70% and even 36% of the original test set size of the industrial dynamic compaction. This shows the tremendous impact on the test set size of the proposed approach.

Table III further shows the impact on the test set size for the transition fault model where the problem of the large pattern count is much more serious. This can especially be seen by the larger total pattern counts for this fault model. The run time behavior for transition faults is generally similar to the behavior for stuck-at faults and therefore not reported here. The results show that the same high reduction rates can be achieved for transition faults. The highest reduction rates are achieved for the circuits l80k and l177k where the original test set size is reduced to only approx. 69% and 37%, respectively.

In summary, the proposed approach proved its effectiveness on large industrial circuits and is able to significantly reduce the test set size for the stuck-at fault model as well as for the transition fault model in comparison to an industrial ATPG

TABLE II
EXPERIMENTAL RESULTS - STUCK-AT

Circuit	Atalanta [18] Time	Atalanta [18] #Pat	DynComp Time	DynComp #Pat	MTTG stage Time	NUM = 5	Time	NUM = 10	Time	NUM = 50	% NUM = 50
b14	0:13m	1086	0:57m	781	2:00m	757	2:49m	742	10:04m	723	92.6
b15	3:06m	642	1:00m	467	2:40m	466	3:35m	466	11:32m	423	92.5
b17	24:36m	2514	6:53m	1104	13:26m	1105	15:26m	1120	28:58m	1079	97.7
b18	1:08h	9112	1:04h	1159	1:16h	1162	1:23h	1170	1:57h	1154	99.6
I49k	-	-	5:19h	353	14:05h	272	20:06h	274	49:33h	280	79.3
I57k	-	-	2:19m	457	3:31m	459	3:51m	452	7:18m	437	95.6
I80k	-	-	2:55m	261	3:26m	238	3:42m	245	5:12m	230	88.1
I88k	-	-	3:41m	767	4:42m	600	5:20m	579	10:19m	539	70.3
I177k	-	-	6:40m	554	19:28m	543	25:44m	524	1:03h	486	87.7
I456k	-	-	1:47h	1630	2:44h	1542	2:51h	1501	5:24h	1498	91.9
I462k	-	-	21:04m	750	27:04m	714	28:59m	701	50:28m	683	91.1
I565k	-	-	29:11m	519	33:00m	569	36:02m	526	1:14h	503	96.9
I2787k	-	-	12:07h	3104	12:20h	1590	14:14h	1405	25:19h	1130	36.4

TABLE III
EXPERIMENTAL RESULTS - TRANSITION

Circuit	DynComp	NUM = 5	NUM = 10	NUM = 50	% NUM = 50
b14	1383	1359	1346	1205	87.1
b15	1228	1174	1141	1000	81.4
b17	2431	2401	2346	2248	92.5
b18	2705	2576	2551	2445	90.4
I57k	1762	1663	1616	1501	85.2
I80k	544	471	433	378	69.5
I88k	7807	7904	7563	7158	92.0
I99k	3981	3611	3613	3570	89.7
I177k	2809	1291	1142	1059	37.7

flow which is tailored to generate highly compact test sets.

VI. CONCLUSIONS

The test set size for the post-production test is an important aspect for reducing test time and hence test costs in industrial practice. Classical dynamic compaction techniques are fast but use constrained test generation methods due to reasons of complexity. As a result, multiple faults which could be be tested with a single test may be missed which increases the pattern count of the test set.

In this paper, we proposed a new SAT formulation to generate tests for multiple faults simultaneously. The robustness and the learning features of SAT-based algorithms are leveraged in order to prevent unnecessary computations and to boost the efficiency of *Multiple Target Test Generation* (MTTG). Techniques have been presented how the resulting MTTG approach can be effectively integrated into a dynamic compaction flow as used in industrial practice. The experimental results on large industrial circuits have shown the feasibility of the approach and the tremendous impact on the test set size. The test set size can be reduced by up to 63% in comparison to sophisticated dynamic compaction techniques. Future work is the development of sophisticated heuristics for fault grouping to increase the effectiveness of the approach and the integration of incremental SAT techniques for run time reduction.

REFERENCES

[1] "International technology roadmap for semiconductors – test and test equipment," 2007, http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[2] J. Rajski, J. Tyszer, and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan," *IEEE Transactions on Computers*, vol. 47, no. 11, pp. 1188–1200, 1998.

[3] P. Goel and B. C. Rosales, "Test generation and dynamic compaction of tests," in *Proceedings of the International Test Conference*, 1979, pp. 189–192.

[4] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A method to generate compact test sets for combinational circuits," in *Proceedings of the International Test Conference*, 1991, pp. 194–203.

[5] 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.

[6] I. Hamzaoglu and J. H. Patel, "New techniques for deterministic test pattern generation," in *Proceedings of the VLSI Test Symposium*, 1998, pp. 446–452.

[7] S. Remersaro, J. Rajski, S. M. Reddy, and I. Pomeranz, "A scalable method for the generation of small test sets," in *Proceedings of Design, Automation and Test in Europe*, 2009, pp. 1136–1141.

[8] A. Czutro, I. Polian, P. Engelke, S. M. Reddy, and B. Becker, "Dynamic compaction in SAT-based ATPG," in *Proceedings of the IEEE Asian Test Symposium*, 2009, pp. 187–190.

[9] G.-J. Tromp, "Minimal test sets for combinational circuits," in *Proceedings of the International Test Conference*, 1991, pp. 204–209.

[10] 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.

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

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

[13] 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.

[14] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. M. Reddy, and B. Becker, "TIGUAN: Thread-parallel integrated test pattern generator utilizing satisfiability analysis," in *Proceedings of the International Conference on VLSI Design*, 2009, pp. 227–232.

[15] 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.

[16] N. Eén and N. Sörensson, "An extensible SAT solver," in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, vol. 2919, 2004, pp. 502–518.

[17] 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.

[18] 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.