# Robust Tests for Transition Faults with Long Propagation Paths Using Boolean Satisfiability

Stephan Eggersglüß Daniel Tille Rolf Drechsler
Institute of Computer Science, University of Bremen
Bibliothekstr. 1, 28359 Bremen, Germany
{segg, tille, drechsle}@informatik.uni-bremen.de

***Abstract*—Increasing speed and decreasing gate sizes make it necessary to test the correct temporal behavior of a manufactured chip. In this paper, we present an efficient SAT formulation for generating robust test patterns for the transition fault model. For this, we apply a multiple-valued logic that is able to model static values and add structural information to model the robust sensitization criterion.**

**Furthermore, we introduce a SAT technique that prioritizes longer paths from the fault site to an output. As a result, the generated test patterns generally sensitize longer paths and consequently are more likely to detect errors caused by small delay defects. Experiments on ISCAS benchmarks and on industrial circuits show the feasiblity of our approach.**

## I. Introduction

Delay testing is widely used in practice to ensure the correct timing behavior of a manufactured chip. The prevalent fault models for testing delay faults are the *Path Delay Fault* (PDF) model [1], [2] and the *Transition Fault* (TF) model [3], [4]. The PDF model is the most appropriate model targeting cumulative delays along a logical path and can be classified broadly in two categories: non-robust and robust [5].

This model is well suited for detecting small as well as large delay defects. Achieving 100% robust PDF coverage would cover all small and large delay defects in the circuit. However, the number of paths in modern circuits is too large so that complete testing of all paths is not possible. Furthermore, only few paths are robust testable.

The TF model assumes a gross or large delay defect at one site in the circuit which is large enough that it can be observed at a primary output of the circuit. The advantage of this model is that the number of faults is linear in the number of connections in the circuit. Therefore, it is widely used in industry guaranteeing a good coverage. A test detects a TF, if it activates the transition at the fault site and sensitizes at least one single path from the fault site to an observation point, i.e. a (pseudo) primary output. By this, not only the large delay defect on the fault site can cause a wrong timing behavior, but also the distributed small delays along the sensitized path.

However, it is pointed out in [6] that for reasons of efficiency ATPG tools usually sensitize a short path to an output. This is disadvantageous, because a distributed delay defect is more likely to be detected on a longer path if the delay defect on the fault site is not large enough. A new TF model called *As Late As Possible Transition Fault* (ALAPTF) was proposed that tries to activate the fault as late as possible and sensitizes a long path to achieve a good detection rate of small delay defects. However due to the timing information, the ATPG method is computationally complex. In [7], a PODEM-based algorithm for detecting high quality tests for transition faults was proposed. But this algorithm focuses on the predetermination of path sensitization conditions for the longest paths.

In [8], an ATPG algorithm for detecting small delay defects that is based on path delay testing was introduced. The algorithm generates a set of multiple-detect test patterns and uses a pattern selection strategy to sensitize the long paths rather than the short paths. But no classification with respect to non-robust and robust tests is made. An approach for testing small delay defects based on the TF model is presented in [9]. There, standard TF testing is combinated with information gathered from static timing analysis. By grouping test patterns and adjusting their timing, the paths can be tested almost with no slack and by this obtain a higher coverage of small delay defects. In [10], a test pattern grading technique is proposed that selects test patterns from an n-detection pattern set according to their effect on small delay defect detection.

Due to the recent advances in techniques to solve the *Boolean Satisfiability* (SAT) problem [11]–[13], SAT-based algorithms have been shown to be efficient in the field of ATPG for delay faults. In [14], SAT-based ATPG for path-oriented TFs was performed, whereas in [15], PDFs are tested using a unified sensitization model. Both approaches have in common that they are not able to model static values what is necessary for robust test patterns. In [16], a SAT-based ATPG algorithm for PDFs was presented. In this approach, static values are modeled for generating robust test patterns using a multiple-valued logic.

The approach presented in this paper uses the multiple-valued logic from [16] for efficiently generating robust test patterns for TFs. For determining whether a path from the fault site is robustly sensitized, a SAT formulation for D-chains [17] is presented which is similar to the formulation for stuck-at faults introduced in [18]. Because it is advantageous for detecting small delay defects to propagate

the transition along a longer path, a SAT technique is presented that makes use of *Incremental SAT* [19], [20] (ISAT). This technique is similar to a technique presented in [21] which is used to speed up equivalence checking but not for increasing the quality.

All outputs at which the fault effect can potentially be observed are ordered according to their distance to the fault site and added incrementally to the SAT instance. By this, it can be ensured, that a long propagation path is chosen. Another advantage of this technique is that, beside the longer paths, the number of faults, which could not be classified, can be reduced. This is due to the circumstance that the considered part of the circuit is much smaller and therefore the SAT instances are less complex.

The remaining part of the paper is structured as follows. In the next section, SAT-based ATPG and the application of multiple-valued logic is introduced. In Section III, the SAT formulation for the D-chains to detect a transition fault is presented, whereas the ISAT technique for generating tests with long propagation paths is shown in Section IV. Experimental results for the proposed approach are presented in Section V and conclusions are drawn in Section VI.

## II. Previous Work

In this section, the usage of SAT in the field of ATPG is introduced. Therefore, Section II-A deals with the general transformation of a circuit problem, i.e. an ATPG problem, to a SAT problem, whereas in Section II-B, the usage of multiple-valued logic for generating robust test patterns for delay faults is explained.

### A. Transformation into SAT Problem

For applying a SAT solver to a circuit problem, the circuit problem has to be transformed into a Boolean formula in *Conjunctive Normal Form* (CNF)[1]. A CNF is a conjunction of clauses and each clause is a disjunction of literals, whereas each literal is a Boolean variable in its positive or negative form. To satisfy the CNF, each clause has to be satisfied. A clause is satisfied, if at least one literal in this clause is satisfied.

According to [22], for translating a circuit $C$ into a CNF $\Phi_C$, each gate $g$ of $C$ has to be transformed into a set of clauses $\Phi_g$. Then, $\Phi_C$ is composed by the conjunction of all clauses of each gate. More formally, the CNF of $C$ is given by the following formula, where $n$ denotes the number of gates in $C$:

$$\Phi_C = \prod_{i=1}^{n} \Phi_g$$

Each connection is represented by a Boolean variable and the CNF representation $\Phi_g$ of each gate $g$ can be obtained either by algebraic conversions or by the usage of a truthtable using the variable dedicated to the incoming or outcoming variables, respectively. In the following, the outgoing connection denotes the name of the gate.

[1]There also exist approaches using circuit-based SAT solvers, but preliminary studies have shown that CNF-based SAT solvers have significant advances regarding run time.

TABLE I
Off-Path Constraints for Robust and Non-Robust Tests

| gate type | robust | | non-robust |
|---|---|---|---|
| | falling | rising | |
| AND/NAND | S1 | X1 | X1 |
| OR/NOR | X0 | S0 | X0 |

The CNF $\Phi_C$ represents the functionality of the circuit, i.e. each satisfying assignment is valid according to the circuit's functionality. For generating test patterns for delay faults, two time frames $t_1, t_2$ have to be considered. For this, the circuit is duplicated such that two test vectors can be calculated. Therefore, $\Phi_C$ contains not a single time frame, but two time frames in the following. The derived CNF of the circuit must be extended by constraints, that model the considered fault. Although this paper deals with the TF model, this will be explained by the PDF model, because it is more general and the conditions concerning the propagation path hold for both.

A PDF is a fault on a path $p = g_1, \ldots, g_n$, where $g_1$ is an input and $g_n$ is an output. To detect the delay fault on $p$, a transition which is either *rising* or *falling* is applied to $g_1$ and propagated along $p$ to $g_n$. Therefore, there exist two different faults for each path in $C$. As stated in [5], there are two categories of tests for the PDF model: *robust* tests and *non-robust* tests. While robust tests guarantee the detection of the fault independently from other delay faults occuring at the same time, non-robust tests do not. By applying only non-robust tests, other delay faults can mask the target delay fault that it will not be observed.

Both types differ in the modeling of the off-path inputs of $p$. An off-path input is an input of a gate $g_i$ on path $p$ that is not $g_{i-1}$. In Table I, the constraints on the off-path inputs are shown. There, X1 (X0) signifies that the value in the final time frame $t_2$ has to be 1 (0). No restriction on $t_1$ is made. The value S1 (S0), however, means, that on both time frames the value has to be 1 (0) and no hazard occurs between them; the signal has to be static.

For modeling the fault, these constraints $\Phi_{fault}$ must be added to the SAT instance. However, static values which are necessary for robust test generation cannot be modeled by Boolean logic. Section II-B deals therefore with the usage of multiple-valued logic in robust test generation. In case of non-robust test generation, the constraints are added to the SAT instance in form of fixed assignments. If the SAT instance is satisfiable, the test pattern can be derived from the solution by extracting the assignments of the variables dedicated to the inputs.

In this paper, we consider the TF model. In this model, no path is given, but a fault site, i.e. a connection or a gate, and a transition which is either rising or falling. A test for the TF model must therefore sensitize at least one path from the fault site to an output. For determining the quality of the test, i.e. non-robust or robust, the same conditions for the off-path inputs as in the PDF model

can be applied with the difference that no path is specified. For detecting a fully sensitized path, so-called D-chains are used. This concept is explained in detail in Section III.

### B. Usage of Multiple-Valued Logic

In this section, the usage of multiple-valued logic for generating robust tests is briefly explained. Further details can be found in [16]. As described in Section II-A, for robust tests, it is necessary that some signals, i.e. off-path inputs, must be guaranteed to be static. This is not possible using Boolean logic. For example, consider an AND gate with incoming connections $a$ and $b$ and outgoing connection $c$. When $a$ has a rising transition and $b$ a falling one, the value on $c$ is in both time frames calculated as 0. However, $c$ is not static, but can have a glitch.

Therefore, a six-valued logic $L_6$ is applied which is able to model static values. Besides the Boolean values, $L_6$ contains also two more static values. The logic $L_6$ is defined as follows:

$$L_6 = \{0, \overline{0}, 01, 10, \overline{1}, 1\}$$

The name of each value determines the behavior of the connection in both time frames. The first position of the name denotes the value of the signal in $t_1$, whereas the value of $t_2$ is given by the second position. For instance, 01 means, that the value is 0 in $t_1$ and 1 in $t_2$. In case of only one position, the signal of $t_1$ and $t_2$ is equal. An overlined value means, that the signal cannot be guaranteed to be static.

For generating a CNF while modeling the circuit in $L_6$, a Boolean encoding is needed. Instead of two variables – each for one time frame – three variables are needed to encode the six values. The CNF of the circuit is then derived in a similar manner as described in the previous section using the CNF representation of each gate according to the chosen Boolean encoding.

## III. SAT Formulation: D-Chains

For the detection of a TF, at least one path from the fault site to an output must be sensitized according to the desired quality of the test, i.e. robust or non-robust. Generally, a TF can be modeled as a stuck-at fault, but using this model, no statement concerning the quality of the test can be made. Therefore, the concept of D-chains for generating robust tests for TFs is presented.

A potential D-chain is defined as a path starting at the fault site and ending at an output. A gate $g$ is on a potential D-chain, if it is located on such a path. For each gate $g$ on a potential D-chain, a variable $D_g$ is introduced[2]. The variable $D_g$ is 1, if and only if the targeted TF is propagated via $g$ to an output. For this, some implications must be added to the SAT instance that guarantee the correct behavior of the off-path inputs of $g$ according to sensitization criteria in Table I. In this section, the criteria concerning robust test generation are considered.

[2]Note that fanouts are handled as a gate and a D-variable is assigned to each outgoing branch.

For each incoming connection $j_i$ (if there are more than one) of $g$ that is located on a potential D-chain, it must be assured that $D_{j_i}$ cannot be 1, if the conditions on the off-path inputs, i.e. the other incoming connections, are not satisfied. The following cases must be considered:

- If the first values of the transition on connection $j_i$ is the non-controlling value $ncv$ and the second one is the controlling value $cv$ ($ncv \rightarrow cv$), e.g. the value 10 for an AND gate, all other incoming connections $j_1, ..., j_{n-1}$ must assume a static $ncv$ ($sncv$):

$$D_{j_i} \wedge (j_i = (ncv \rightarrow cv))$$
$$\rightarrow (\forall j_{k|0<k<n; j_k \neq j_i} : j_k = sncv)$$

- If the transition on $j_i$ is ($cv \rightarrow ncv$), the values of $j_1, ..., j_{n-1}$ are restricted to be non-controlling only in $t_2$, e.g. the values $01, \overline{1}, 1$ for an AND gate (given by $X \rightarrow ncv$):

$$D_{j_i} \wedge (j_i = (cv \rightarrow ncv))$$
$$\rightarrow (\forall j_{k|0<k<n; j_k \neq j_i} : j_k = X \rightarrow ncv)$$

- It must be ensured that at least one complete sensitized path, i.e. a D-chain, exists. Therefore, each gate $g$ for which $D_g = 1$ holds must have at least one successor $h$ for which $D_h = 1$ holds as well. Given gate $g$ on a potential D-chain with successors $h_1, \ldots, h_n$, the following implication must be added to the SAT instance:

$$(D_g = 1) \rightarrow \sum_{j=1}^{n} D_{h_j}$$

- The value of the fault site $f$ must be set to the targeted transition, i.e. 01 for a rising TF ($rTF$) and 10 for a falling TF ($fTF$). To propagate the delay defect, the D-variable $D_{fault}$ of the fault site must be set to 1, i.e. the following constraints are added:

$$(rTF \rightarrow f = 01) \oplus (fTF \rightarrow f = 10)$$
$$D_{fault} = 1$$

The following example demonstrates the methodology:

*Example 1:* Consider the circuit in Figure 1. The fault site for a rising TF is the outgoing connection of gate $f$. There are two potential D-chains: $f - f1 - g$ and $f - f2 - h$. Therefore, D-variables are assigned to all gates or connections, respectively, on potential D-chains, i.e. $f, f1, f2, g, h$. The value on $f$ is set to a rising transition 01 and $D_f$ is assigned to 1. One of the successors of $f$ must propagate the delay defect. When $D_{f2}$ is assumed to be 1, the off-path input $d$ must assume the static value 0. But this is not possible, because this would also lead to the static value 0 on $f$. By this, the fault cannot be justified. Consequently, $D_{f2}$ cannot be 1 targeting a rising TF on $f$. Therefore, the delay defect must be propagated via $f_1$. This is possible, because $D_{f1} = 1$ implies $e = 0$ which can be obtained by setting $a$ and $b$ to static 0.

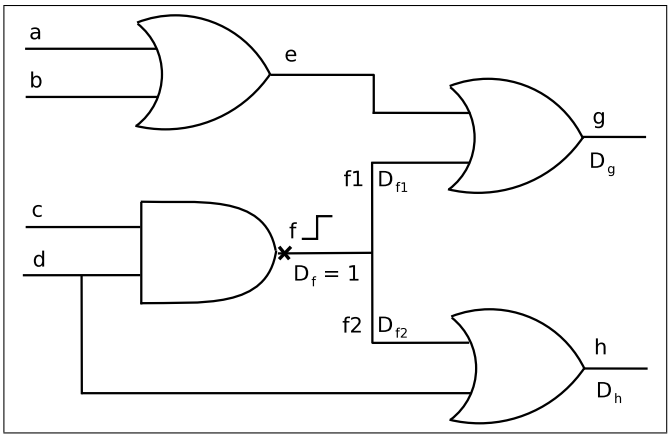


Fig. 1. Example circuit for D-chains

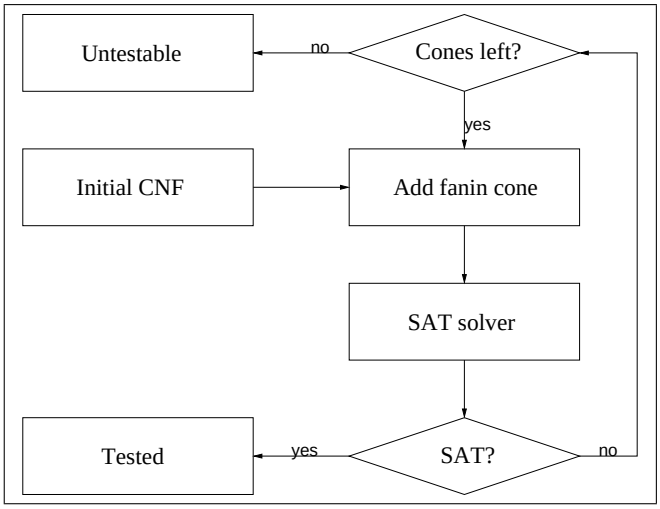


Fig. 2. Incremental Approach for Generating Tests with Long Propagation Paths

The shown implications and properties are transformed into CNF and added to the SAT instance. By this, the instance grows but also speeds up the search process as it is shown for stuck-at faults in [18]. Note that the SAT formulation for non-robust tests can be easily derived by relaxing the above described conditions.

## IV. ISAT: Long Propagation Paths

As mentioned above, generating long propagation paths is generelly preferred for detecting small delay defects. Unfortunately, after building the CNF, no circuit information can be used during the solving step. Therefore, the search cannot be guided directly to find long propagation paths. During the instance generation the entire influenced circuit part, i.e. all gates on potential D-chains and all gates which are needed to justify the fault, is transformed into the CNF. Each output on a potential D-chain is a candidate for the observation of the fault. As pointed out earlier, the likelihood of chosing a short propagation path is higher than chosing a long propagation path considering all outputs at one step.

We propose an incremental approach to guide the test algorithm preferring long propagation paths. A sketch of the algorithm is illustrated in Figure 2. After all outputs where a fault could be observed are determined by a depth-first-search, the list of outputs is sorted in terms of the distance to the fault site. In our approach, the distance is the number of gates on the potential D-chain, i.e. the length of the propagation path. The algorithm is easily extensible by technology-dependent parameters.

Afterwards, the fanin cone of the first output, i.e. the output with the longest path, is added to the initial SAT instance (consisting of the CNF that injects the fault). If this SAT instance is satisfiable a test pattern sensitizing the longest path is found. Otherwise no classification can be given, since a shorter path could exist.

Therefore, the initial SAT instance is augmented incrementally by the second output's fanin cone using ISAT. The information learned so far is kept. By this, the restarting search process could benefit such that already explored search space must not be traversed again. If there are no more fanin cones left, the fault is untestable.

More formally, let $len(o)$ be the function that denotes the length of the propagation path for output $o$. Then, the outputs $o_1, ..., o_n$ on a potential D-chain are ordered such that $len(o_i) \geq len(o_j) | i < j$. The CNF of the fanin cone of $o_i$ is given by $\Phi_{o_i}$, whereas $\Phi_{fault}$ represents the fault modeling. Then, the initial SAT instance is given by:

$$\Phi_1 = \Phi_{o_1} \cdot \Phi_{fault}$$

If $\Phi_1 = 0$, the next SAT instance is created as follows:

$$\Phi_2 = \Phi_1 \cdot \Phi_{o_2}$$

or more generally:

$$\Phi_i = \Phi_{i-1} \cdot \Phi_{o_i} : 0 < i \leq n$$

The procedure stops after solving $\Phi_n$ or if $\Phi_i = 1$. By this procedure, the longest sensitizable propagation path is found without including timing information in the problem instance which would slow down the search process.

However, a large number of outputs on a potential D-chain would result in significant overhead, if the longest paths are not testable adding the fanin cone of each single output in one step. Therefore, an $n$-steps approach is proposed. This approach restricts the number of incremental steps to $n$ and add more than the fanin cone of one output in each single step.

The number of outputs added in one single step depends on $n$. For example, if $n = 4$, than 25% of the outputs on a potential D-chain are added in one single step. This results in at most four incremental steps. As the experiments in the next section will show, the $n$-steps approach speeds up the search process and at the same time degrades the length of the propagation path only slightly.

## V. Experimental Results

In this section, the experimental results for generating robust tests for the TF model with long propagation

TABLE II
CIRCUIT STATISTICS

| circuit | #PI | #FF | av. #PO | #Faults |
|---|---|---|---|---|
| c2670 | 157 | 0 | 4.13 | 2802 |
| c3540 | 50 | 0 | 7.56 | 3742 |
| c5315 | 178 | 0 | 6.83 | 6016 |
| c6288 | 32 | 0 | 15.96 | 7744 |
| c7552 | 206 | 0 | 6.02 | 8078 |
| s641 | 36 | 20 | 3.80 | 324 |
| s5378 | 35 | 179 | 2.92 | 3166 |
| s13207 | 62 | 638 | 3.89 | 3322 |
| s15850 | 77 | 534 | 5.56 | 2446 |
| s38417 | 28 | 1636 | 2.02 | 794 |
| s38584 | 38 | 1426 | 2.52 | 5148 |
| p44k | 739 | 2175 | 55.49 | 10000 |
| p77k | 171 | 2977 | 21.58 | 10000 |
| p80k | 152 | 3878 | 37.30 | 10000 |
| p99k | 167 | 5747 | 17.29 | 10000 |
| p177k | 768 | 10507 | 444.88 | 10000 |
| p462k | 1815 | 29205 | 79.46 | 10000 |

paths are presented. The algorithm is executed for ISCAS benchmarks and for industrial circuits provided by NXP Semiconductors Hamburg, Germany; both in full-scan version. As SAT solver, we used MiniSat v1.14 [13]. All experiments were carried out on a Dual DualCore Xeon (3000 MHz, 32768 MByte RAM) running GNU/Linux.

The experimental setup is as follows. For ISCAS benchmarks, robust tests are generated for each input and each fanout branch targeting the falling and the rising TF. For the industrial circuits, due to the large number of fault locations and due to the absence of a fault simulator targeting robust test for TFs[3], the number of faults is limited to 10,000. If the total number of faults exceeds the limit, the number of faults on the inputs and the number of faults on the fanout branches are each limited to 5,000. Furthermore, it was tried to achieve a large structural coverage of the circuit, i.e. only few targeted faults are located in the same region. The timeout for each fault was set to 13 MiniSat restarts (further details can be found in [13]).

The name of the industrial circuit roughly denotes the size of the circuit, e.g. p462k contains nearly half a million gates. Further information about the circuits can be found in Table II. The first columns gives the name of the circuit, whereas in column *#PI* and column *#FF*, the number of primary inputs and the number of flipflops are shown, respectively. The average number of outputs on a potential D-chains for each targeted fault is presented in column *av. #PO* and the number of targeted faults is given in the last column.

The experimental results are presented in Table III. In the first column, the name of the circuit is given, whereas the results for the classical approach, where the outputs are considered altogether are given in the column entitled *non-incremental*. The column *incremental* provides the results for the incremental approach in which each single output is added incrementally in one step, whereas the columns *4-steps, 8-steps* and *32-steps* present configurations of the $n$-steps approach with $n = 4$, $n = 8$ and $n = 32$, respectively. Note, that the performance of the incremental approach is very poor for the industrial circuits. Therefore, it is not applicable, but the results are presented anyway for the purposes of comparison.

The run time needed for targeting all faults are given in columns *time*, whereas the number of faults that could not be classified within the restart limit are given in column *ab*. The average length of the propagation paths of all robustly testable faults are presented for each circuit in columns entitled *D-len*. In the upper part of the table, the results for the ISCAS benchmarks are shown, whereas in the lower part the results for the industrial circuits are presented.

The experiments show that all targeted faults in the ISCAS benchmarks could be classified in less than two minutes (except c6288 for which nearly seven minutes are needed). The run times for the more complex industrial circuits are generally higher and vary between under three minutes (p99k) and approximately seven hours (p177k) for the non-incremental approach. It can be concluded that this approach achieves a very good performance even on industrial circuits. But due to the large SAT instances considering the outputs altogether, not all faults could be classified in the industrial circuits, e.g. about 13% in case of p177k and about 2% in case of p462k.

However, an analysis of the length of the propagation paths shows that the non-incremental approach sensitizes rather short paths compared to the length of the propagation paths of the incremental approach. The length of the latter is increased up to a factor of more than two (s641) for the ISCAS benchmarks and up to a factor of nearly two (p80k) for the industrial circuits.

Considering the $n$-steps approaches that add more than one output in each incremental step to the SAT instance (4-steps, 8-steps, 32-steps), there are only slight differences in run time for the ISCAS benchmarks. But already for the 4-step approach, the length of the propagation paths is increased significantly compared to the non-incremental approach. It reaches almost the maximum length given by the incremental approach. The other $n$-steps approaches with a finer granularity (8-steps, 32-steps) still improve slightly the length and 32-steps provides the same length as the incremental approach.

Concerning the industrial circuits, the run time of the $n$-steps approaches generally grows with the increasing of $n$ due to the higher number of incremental steps. Compared to the run times of the non-incremental approach, the results of 4-steps and 8-steps perform generally better, whereas the results of 32-steps are in the same range. An exception is p177k, for which the number of not classified faults could be significantly reduced. By this, the run time could also be reduced.

For the length of the propagation paths, it can be observed that, similar to the ISCAS benchmarks, the length is increased significantly already for the 4-steps

[3]As common in industrial practice, a fault simulator is called for each test pattern. If it detects more faults, these are dropped from the fault list. By this, the number of targeted faults is reduced.

TABLE III
EXPERIMENTAL RESULTS

| | non-incremental | | | incremental | | | 4-steps | | | 8-steps | | | 32-steps | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| circ | time | ab. | D-len | time | ab. | D-len | time | ab. | D-len | time | ab. | D-len | time | ab. | D-len |
| c2670 | 0:31m | 0 | 6.93 | 0:31m | 0 | 7.95 | 0:31m | 0 | 7.87 | 0:31m | 0 | 7.95 | 0:31m | 0 | 7.95 |
| c3540 | 1:26m | 0 | 11.58 | 1:28m | 0 | 13.19 | 1:25m | 0 | 13.10 | 1:27m | 0 | 13.13 | 1:29m | 0 | 13.19 |
| c5315 | 0:59m | 0 | 6.41 | 0:57m | 0 | 7.23 | 0:55m | 0 | 7.21 | 0:55m | 0 | 7.21 | 0:57m | 0 | 7.23 |
| c6288 | 6:46m | 0 | 16.27 | 6:36m | 0 | 16.27 | 6:17m | 0 | 16.27 | 6:23m | 0 | 16.27 | 6:37m | 0 | 16.27 |
| c7552 | 1:52m | 0 | 7.25 | 1:52m | 0 | 7.40 | 1:48m | 0 | 7.40 | 1:49m | 0 | 7.40 | 1:52m | 0 | 7.40 |
| s641 | 0:01m | 0 | 9.87 | 0:01m | 0 | 20.47 | 0:01m | 0 | 19.83 | 0:01m | 0 | 20.47 | 0:01m | 0 | 20.47 |
| s5378 | 0:13m | 0 | 13.52 | 0:12m | 0 | 14.10 | 0:11m | 0 | 14.10 | 0:11m | 0 | 14.10 | 0:11m | 0 | 14.10 |
| s13207 | 0:52m | 0 | 18.65 | 0:54m | 0 | 19.49 | 0:53m | 0 | 19.48 | 0:54m | 0 | 19.49 | 0:54m | 0 | 19.49 |
| s15850 | 0:27m | 0 | 14.46 | 0:28m | 0 | 21.31 | 0:27m | 0 | 21.30 | 0:28m | 0 | 21.30 | 0:28m | 0 | 21.31 |
| s38417 | 0:01m | 0 | 10.07 | 0:01m | 0 | 10:16 | 0:01m | 0 | 10.16 | 0:01m | 0 | 10.16 | 0:01m | 0 | 10.16 |
| s38584 | 0:28m | 0 | 15.67 | 0:25m | 0 | 16.86 | 0:24m | 0 | 16.85 | 0:24m | 0 | 16.86 | 0:25m | 0 | 16.86 |
| p44k | 4:41h | 0 | 10.24 | 14:20h | 0 | 11.22 | 4:51h | 0 | 11.16 | 4:42h | 0 | 11.17 | 4:47h | 0 | 11.19 |
| p77k | 3:09m | 0 | 6.57 | 4:42m | 0 | 6.60 | 2:58m | 0 | 6.59 | 3:01m | 0 | 6.57 | 3:36m | 0 | 6.60 |
| p80k | 32:36m | 0 | 7.30 | 1:32h | 0 | 14.31 | 12:57m | 0 | 13.54 | 15:09m | 0 | 14.11 | 30:58m | 0 | 14.29 |
| p99k | 2:47m | 0 | 8.38 | 1:57h | 0 | 12.02 | 4:21m | 0 | 11.89 | 4:49m | 0 | 11.96 | 7:34m | 0 | 12.01 |
| p177k | 7:09h | 1331 | 7.84 | 7:50h | 713 | 12.91 | 5:49h | 904 | 11.21 | 5:03h | 760 | 12.50 | 5:01h | 718 | 12.87 |
| p462k | 1:08h | 221 | 10.38 | 9:43h | 151 | 12.49 | 1:01h | 182 | 12.21 | 1:05h | 167 | 12.39 | 1:10h | 156 | 12.47 |

approach. The 32-steps approach then nearly reaches the maximum length.

The experiments have shown that the length of the propagation paths can be increased significantly using the incremental approach. Due to the large run time, this approach is not feasible. The proposed $n$-steps approaches however reduce the run time significantly compared to the incremental approach. At the same time, the length of the propagation paths is only very slightly decreased.

## VI. Conclusions

In this paper, a new efficient SAT formulation for generating robust test patterns for the TF model was presented. The algorithm is based on a multiple-valued logic for modeling static values and adds structural information for detecting robustly sensitzed paths from the fault site to an output. Furthermore, a SAT technique was presented that prioritizes longer paths such that small delay defects are more likely to be found. The experiments have shown that the length of the propagation path could be increased by a factor of more than two using this technique.

## Acknowledgment

Parts of this research work were supported by the German Federal Ministry of Education and Research (BMBF) in the Project MAYA under contract number 01M3172B and by the German Research Foundation (DFG) under contract number DR 287/15-1.

## References

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

[2] C.-J. Lin and S. Reddy, "On delay fault testing in logic circuits," *IEEE Trans. on CAD*, vol. 6, no. 5, pp. 694–703, 1987.

[3] J. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar, "Transition fault simulation," *IEEE Design & Test of Computers*, pp. 32–38, 1987.

[4] K.-T. Cheng, "Transition fault testing for sequential circuits," *IEEE Trans. on CAD*, vol. 12, no. 12, pp. 1971–1983, 1993.

[5] K. Cheng and H. Chen, "Classification and identification of nonrobust untestable path delay faults," *IEEE Trans. on CAD*, vol. 15, no. 8, pp. 845–853, 1996.

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

[7] Y. Shao, I. Pomeranz, and S. Reddy, "On generating high quality tests for transitions faults," in *Asian Test Symp.*, 2002.

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

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

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

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

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

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

[14] K. Yang, K.-T. Cheng, and L.-C. Wang, "Trangen: a SAT-based ATPG for path-oriented transition faults," in *ASP Design Automation Conf.*, 2004, pp. 92–97.

[15] S.-Y. Lu, M.-T. Hsieh, and J.-J. Liou, "An efficient SAT-based path delay fault ATPG with an unified sensitization model," in *Int'l Test Conf.*, 2007.

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

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

[18] P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Trans. on CAD*, vol. 15, pp. 1167–1176, 1996.

[19] J. Hooker, "Solving the incremental satisfiability problem," *Journal of Logic Programming*, vol. 15, no. 1-2, pp. 177–186, 1993.

[20] O. Shtrichman, "Pruning techniques for the SAT-based bounded model checking problem," in *CHARME*, ser. LNCS, vol. 2144, 2001, pp. 58–70.

[21] S. Disch and C. Scholl, "Combinational equivalence checking using incremental SAT solving, output ordering, and resets," in *ASP Design Automation Conf.*, 2007, pp. 938–943.

[22] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. on CAD*, vol. 11, no. 1, pp. 4–15, 1992.