As-Robust-As-Possible Test Generation in the Presence of Small Delay Defects using Pseudo-Boolean Optimization

Stephan Eggersglüß*†, Rolf Drechsler*
*Institute of Computer Science, University of Bremen,
28359 Bremen, Germany
{segg, drechsle}@informatik.uni-bremen.de
†German Research Center for Artificial Intelligence (DFKI)

***Abstract*—Delay testing is performed to guarantee that a manufactured chip is free of delay defects and meets its performance specification. However, only few delay faults are robustly testable. For robustly untestable faults, non-robust tests which are of lesser quality are typically generated. Due to significantly relaxed conditions, there is a large quality gap between non-robust and robust tests. This paper presents a test generation procedure for *As-Robust-As-Possible* (ARAP) tests to increase the overall quality of the test set. Instead of generating a non-robust test for a robustly untestable fault, an ARAP test is generated which maximizes the number of satisfiable conditions required for robust test generation by pseudo-Boolean optimization. Additionally, the problem formulation is extended to incorporate the increased significance of small delay defects. By this, the likeliness that small delay defects invalidate the test is reduced. Experimental results on large industrial circuits confirm the quality gap and show that the generated ARAP tests satisfy a large percentage of all robustness conditions on average which signifies a very high quality.**

I. Introduction

The significance of delay defects increases with the shrinking features sizes of today's designs. Therefore, delay testing is a major issue in the post-production test to filter out defective devices. Delay tests are widely used to check whether a manufactured chip is free of delay faults and meets its performance specification. The most accurate delay fault model is the *Path Delay Fault Model* (PDFM) [1], [2]. The PDFM captures small as well as large delay defects along one path in the circuit. However, the PDFM suffers from the large number of paths in a circuit. For that reason, typically only tests for critical paths are generated to ensure the correct timing behavior. Additionally, tests for the PDFM are used for diagnostic reasons. Here, high-quality tests are required.

Delay tests can be of different quality [2], [3]. These tests can be roughly classified into two categories. *Robust* tests guarantee the detection of the fault independently from the presence of other delay faults in the circuit. On the other hand, *non-robust* tests guarantee the detection of the fault only if no other delay fault is present in the circuit. Non-robust and robust tests differ in the sensitization conditions on the side inputs of the path under test. Robust test generation requires stringent conditions on the side inputs of the path. Static values on certain signals are required in order to guarantee the absence of activity which might invalidate the test. Nowadays, the quality of tests is of high concern in the industry. Obviously, robust tests are more desirable to obtain. However, only a small percentage of all paths can be tested robustly [4] due to the stringent conditions. For robustly untestable faults, non-robust tests are typically generated. These tests can generally be invalidated by other delay faults at the side inputs of the path under test.

Several approaches were developed in order to increase the quality of tests for robustly untestable paths. The approach in [5] orders all signals which require static values for a robust test in a static manner taking timing information into account. Then, static values are assigned successively to those signals if possible. However, the result is significantly influenced by the static order since no backtracking is performed. As a result, the solution with highest quality might be missed. The work presented in [6] describes a test generation approach which targets untestable critical paths by robustly testing their longest segment only. In [7], the *As-Late-As-Possible Transition Fault* (ALAPTF) model is proposed. In order to accumulate small delay defects, the transition is launched as late as possible via the longest robust segment. The computational effort using this method is very high.

In this paper, *As-Robust-As-Possible* (ARAP) test generation is proposed. If a *Path Delay Fault* (PDF) is robustly untestable, a test is generated with the maximal number of static side inputs possible instead of a non-robust test. As a result, the test is guaranteed to be as robust as possible and provides increased quality. To achieve this, the problem is formulated as a *Pseudo-Boolean Optimization* (PBO) problem which can be solved by a pseudo-Boolean SAT solver, e.g. [8]–[11]. In contrast to a static order (as used in [5] for a different aim), the PBO solver works in a dynamical manner and performs backtracking automatically to avoid local maximums in an efficient way.

Pseudo-Boolean SAT solvers benefit significantly from the recent advances in SAT solving techniques [12] and have already shown their feasibility in other domains like for instance the maximum circuit activity estimation [13], [14]. Another optimization problem is the test generation with maximal crosstalk-induced delay. In [15], this problem is targeted with an *Integer Linear Programming* (ILP) formulation which is generally similar to the PBO problem formulation. In [16],

978-3-9810801-7-9/DATE11 ©2011 EDAA

the problem of critical path selection considering coupling noise was formulated as a weighted partial Max-SAT problem. Here, the formulation distinguishes between hard and relaxable constraints.

A further issue in the field of delay testing is the increased distribution of *Small Delay Defects* (SDDs) caused by the shrinking manufacturing technologies. SDDs are distributed delay defects which are not able to cause a delay defect by its own, but only if accumulated. Delay defects caused by SDDs are more likely to manifest on long paths [17], [18]. Due to the flexibility of the proposed PBO formulation, it is easily possible to incorporate additional information into the PBO problem which decreases the likeliness that the test is invalidated by distributed delay defects. This is done by prioritizing static values on side inputs with fanin cones with large depth containing long paths.

The proposed approach for ARAP test generation was experimentally evaluated on large industrial circuits with up to 3.3 million elements. The results show that ARAP tests typically contain a high percentage of static side inputs resulting in a very high test quality compared to conventional non-robust tests.

The remainder of the paper is structured as follows. Section II presents the preliminaries of this work. ARAP tests are introduced in Section III. Section IV presents the proposed test generation methodology for generating ARAP tests using PBO. Section V shows the extension of the PBO formulation in order to decrease the likeliness that the test is invalidated by SDDs and Section VI provides experimental results on large industrial circuits. Finally, conclusions are drawn in Section VII.

II. Preliminaries

This section presents the preliminaries of this work. Section II-A describes the different sensitization criteria for PDFs. Section II-B introduces basic information about PBO and its application to circuit-oriented problems, and Section II-C provides the necessary information about robust test generation using *Boolean Satisfiability* (SAT).

A. Sensitization Criteria

A *Path Delay Fault* (PDF) models a distributed delay on a structural path $\mathcal{P}$ from an input to an output of a circuit. A structural path $\mathcal{P}$ is defined as the sequence of gates $g_1, \ldots, g_o$, where g_1 is an input and g_o is an output. The path $\mathcal{P}$ must be complete. That means, each gate g_i on $\mathcal{P}$ with $0 < i < k$ must be an input of g_{i+1}. If a gate g_i is located on path $\mathcal{P}$, it is denoted by $g_i \in \mathcal{P}$. According to the direction of the transition at the beginning of path $\mathcal{P}$, there is rising PDF as well as a falling PDF for each structural path.

A test for a PDF has to consider two time frames t_1, t_2. In order to generate a test for a PDF on path $\mathcal{P}$, the desired transition has to be launched at the input and $\mathcal{P}$ has to be sensitized to propagate the transition. For this, the side inputs of $\mathcal{P}$ have to be constrained to specific values according to the desired sensitization criterion. A side input of $\mathcal{P}$ is an input of gate $g_i \in \mathcal{P}$ with $1 < i \leq o$ which is not on $\mathcal{P}$. The set of side inputs of $\mathcal{P}$ is denoted by $S^{\mathcal{P}}$. The sensitization criteria is responsible for the test quality.

TABLE I
Sensitization criteria for robust and non-robust tests

Gate type	Robust rising	Robust falling	Non-robust
AND/NAND	$X1$	$S1$	$X1$
OR/NOR	$S0$	$X0$	$X0$

Table I shows the criteria for non-robust as well as robust sensitization. For a non-robust test, it is sufficient that all side inputs of $\mathcal{P}$ have to assume the non-controlling value of the gate in t_2 only (denoted by X0/X1). In order to avoid that other delay faults mask the fault on $\mathcal{P}$, static values have to be guaranteed for a robust test if the transition goes from a non-controlling value to a controlling value (denoted by S0/S1). The set of side inputs on which a static value is required is denoted by $S_s^{\mathcal{P}}$ in the following. A robust test is only possible when all side inputs $s \in S_s^{\mathcal{P}}$ are able to assume a static value.

B. Pseudo-Boolean Optimization

The *Pseudo Boolean Optimization* (PBO) problem consists of a pseudo-Boolean formula Ψ and an objective function $\mathcal{F}$. A pseudo-Boolean formula is a conjunction of pseudo-Boolean constraints. A pseudo-Boolean constraint ψ over Boolean variables $x_0, \ldots, x_{n-1}$ is an inequality of the form:

$$\sum_{i=0}^{n-1} c_i \dot{x}_i \geq c_n,$$

where $c_0, \ldots, c_n \in \mathbb{Z}$ and $\dot{x}_i \in \{0, 1\}$ (corresponding to the assignment of x_i). A pseudo-Boolean constraint ψ is satisfied if and only if the sum of the coefficients c_i with $0 \leq i < n$ for which the associated variable x_i is activated, that is $x_i = 1$, is greater or equal than c_n. A pseudo-Boolean formula Ψ is satisfied if and only if each constraint $\psi \in \Psi$ is satisfied.

The formula $\mathcal{F}$ is to minimize a given objective function of the form:

$$\mathcal{F}(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} m_i \dot{x}_i,$$

where $m_0, \ldots, m_{n-1} \in \mathbb{Z}$. Therefore, the PBO problem is to determine the solution which satisfies Ψ (also known as the *Pseudo-Boolean Satisfiability* (PB-SAT) problem) and, at the same time, minimizes the given objective function $\mathcal{F}$.

The solving process of a PBO solver can be regarded as an iterative application of a PB-SAT solver. At first, an initial solution is calculated and improved in the following search process until no further solution is possible. Due to the desired optimization, the search space which has to be explored is huge. However, PBO solvers use efficient conflict-based learning techniques during the search. As a result, the search space can typically be traversed very quickly, since a large part can be pruned by learned information.

The application of PB-SAT is related to the application of SAT. In order to transform a circuit-oriented problem into a PBO problem, the circuit has to be modeled in PB constraints. Each signal s_j in a circuit is assigned a Boolean variable x_j. Similar to the transformation into a SAT problem [19], the functionality of each gate g can be represented by a set of constraints ψ_g. In fact, each SAT constraint, i.e. a CNF clause, can be easily converted into a PB constraint. Table II shows the representation of an AND gate in PB constraints as well

TABLE II
PSEUDO-BOOLEAN AND CNF REPRESENTATION FOR AND GATE $a \cdot b = c$

PB	CNF
$((1-a)+(1-b)+c \geq 1)\cdot$ $(a+(1-c) \geq 1)\cdot$ $(b+(1-c) \geq 1)$	$(\overline{a}+\overline{b}+c)\cdot$ $(a+\overline{c})\cdot$ $(b+\overline{c})$

as in CNF. Note that a negative literal $\overline{x}_i$ is represented by the term $(1-x_i)$. The PB representation $\Psi_{\mathcal{C}}$ for circuit $\mathcal{C}$ with gates $g_1,\ldots,g_k$ is given by the following formula:

$$\Psi_{\mathcal{C}} = \prod_{j=0}^{k} \psi_{g_j}$$

After the problem has been formulated as a PB-SAT instance Ψ and an objective function $\mathcal{F}$ has been created, the PBO problem is solved by a dedicated PBO solver. However, there are two different types of solvers. Solvers like Pueblo [10] directly supports PB constraints, while solvers like MiniSat+ [9] translate the PB-SAT problem into a SAT instance and apply conventional SAT algorithms to find a solution. Obviously, the latter type of solvers are particularly suited for problems, which are modeled with many clauses and a few pseudo-Boolean constraints [20].

C. SAT-based ATPG for Robust Tests

The proposed PB-SAT formulation is based on the SAT formulation for robust PDF test generation. Therefore, the SAT formulation is briefly introduced in the following. The advantage of SAT-based ATPG algorithms is the high robustness of modern SAT solvers, which make the application particularly suitable for hard problems such as robust test generation. SAT-based test generation for robust tests was first introduced in [21] for combinational circuits. The approach MONSOON [22] presented a SAT formulation for sequential circuits considering tri-state elements and unknown values and shows the efficient application in industrial practice. In order to apply a SAT solver to a circuit-oriented problem, the problem has to be converted into a Boolean formula in *Conjunctive Normal Form* (CNF).

As described above, robust tests require the modeling of static values which cannot directly be represented by Boolean logic. Therefore, both approaches use a multiple-valued logic $\mathcal{L}$ to represent the sequential behavior of the circuit guaranteeing static values. In order to apply a Boolean SAT solver, a Boolean encoding is used. Here, several Boolean variables are used to represent one value of $\mathcal{L}$. Assume that a set of Boolean variables X is used to represent all values of $\mathcal{L}$.[1] The Boolean encoding is chosen such that one variable $x_S \in X$ determines whether the signal is guaranteed to be static or not.

Example 1: Assume that a signal x is represented by three Boolean variable x_1, x_2, x_S. The variable x_1 denotes the value of x in the initial time frame t_1 and x_2 denotes the value of x in the final time frame t_2. Additionally, x_S determines whether x is guaranteed to be static or not. If $x_1 \neq x_2$, a transition occurs and $x_S = 0$ holds. If $x_1 = x_2$, then $x_S = 1$ determines that x is guaranteed to be static. This is ensured by additional constraints as described in [22].

[1]The number of variables contained in this set differs according to the logic used. This is described in more detail in [22].

III. AS-ROBUST-AS-POSSIBLE TESTS

The quality of a delay test for path $\mathcal{P}$ is determined by the sensitization criterion applied to the side inputs of $\mathcal{P}$. However, in order to generate a robust test, the stringent conditions on the side inputs must hold for each gate $g \in \mathcal{P}$. Crucial is that all side inputs $s \in S_S^{\mathcal{P}}$ have to assume a static value in order to avoid that a test is invalidated by another delay fault. If at least one side input is not able to assume the required value, a robust test is not possible anymore. Then, a non-robust test has to be generated.

Typically, robust test generation fails because a subset of the side inputs (and not all) cannot be set to static values at the same time. For a non-robust test, the conditions on the side inputs are significantly relaxed for all side inputs on $\mathcal{P}$. Here, none of the side inputs of $\mathcal{P}$ requires a static value. As a result, an invalidation of the test is possible at any side input $s \in S_S^{\mathcal{P}}$. Therefore, there is a large quality gap between non-robust and robust tests.

Instead of a non-robust test, we propose to generate an *As-Robust-As-Possible* (ARAP) test for a robustly untestable fault. In contrast to a non-robust test, the number of side inputs $s \in S_S^{\mathcal{P}}$ which can be set to a static value is maximized. By this, the quality of the test is increased, since each side input set to a static value cannot be invalidated by other delay faults anymore.

Example 2: Figure 1 shows a small example circuit (including one flipflop FF) with tests for a PDF with falling transition on path $a-d-g-j-m$. This PDF is robustly untestable due to the sequential behavior. A non-robust test requires that the side inputs of the gates d, g, m are set to the non-controlling value of the gate in t_2, i.e. line b and g are set to $X1$, while line l assumes the value $X0$. This is shown in Figure 1(a).

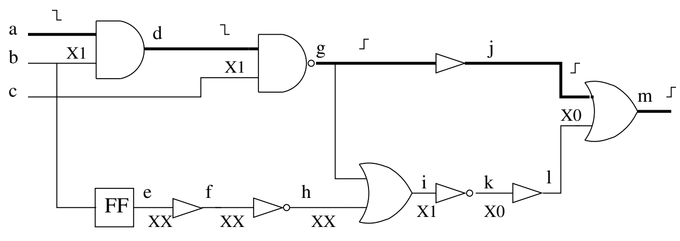
Although the PDF is robustly untestable, some of the inputs can be set to a static value. Figure 1(b) shows an ARAP test in which the number of static side inputs is maximized. Here, two of all three side inputs, i.e. b and c, are set to a static value and prevent that other delay faults invalidate the test at the corresponding gates. A static value at side input l is not possible, because of the assignment of side input b and the sequential behavior (flipflop e).

IV. TEST GENERATION FOR ARAP TESTS

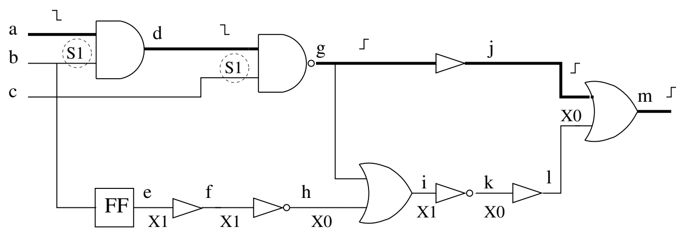
The following procedure is proposed to efficiently generate ARAP tests. In order to generate an ARAP test with a maximized number of static side inputs, we formulate the problem as a *Pseudo-Boolean Optimization* (PBO) problem and solve it by a dedicated PBO solver. For a circuit $\mathcal{C}$ and a PDF F on path $\mathcal{P}$, the problem is formulated as follows: First, the circuit $\mathcal{C}$ is transformed into PB constraints $\Psi_{\mathcal{C}}$ as described in Section II. Here, a circuit representation is used which is able to model static values (see Section II-C). Then, $\Psi_{\mathcal{C}}$ is extended with PB constraints Ψ_{NR} for non-robust sensitization along the side inputs $S^{\mathcal{P}}$. This results in a PB-SAT instance

$$\Psi_{\mathcal{C}}^F = \Psi_{\mathcal{C}} \cdot \Psi_{\text{NR}}$$

suitable for non-robust test generation. Note that this can still be modeled as a Boolean SAT instance.



(a) Non-robust test



(b) ARAP test

Fig. 1. Tests for path $a - d - g - j - m$

Next, the set of side inputs $S_S^{\mathcal{P}}$ which have to assume a static value for a robust test is identified and the objective function $\mathcal{F}$ is formulated. The objective function $\mathcal{F}$ is formulated over the variables of the side inputs $S_S^{\mathcal{P}}$. However, not all variables of the signals contained in $S_S^{\mathcal{P}}$ have to be included in $\mathcal{F}$. For signal x, it is sufficient to include the variable x_S in $\mathcal{F}$, since the assignment of x_S determines whether the signal is guaranteed to be static. The number of static side inputs have to be maximized. Therefore, the objective function $\mathcal{F}$ is the sum of all variables x_S of all signals $s_i \in S_S^{\mathcal{P}}$ multiplied by the constant (-1):

$$\mathcal{F}(x_S^0, \ldots, x_S^n) = \sum_{i=0}^{n} (-1) \cdot x_S^i,$$

where x_S^i is the variable x_S of signal x_i and $n = |S_S^{\mathcal{P}}|$. As a result, the constants associated with each variable x_S^i are accumulated, if x_S^i is active, i.e. $x_S^i = 1$, which means that the corresponding side input is guaranteed to be static.[2]

Finally, the resulting PBO problem, i.e. the PB-SAT instance $\Psi_{\mathcal{C}}^F$ and the objective function $\mathcal{F}$, is given to a PBO solver in order to calculate a solution. The solution directly provides the test with the maximum number of static side inputs. That is, if the test is robustly testable, a robust test is generated. Otherwise, an ARAP test is generated or the fault is proven untestable. Both test problems are integrated into one problem instance and the generated test is automatically of highest quality possible.

The following example demonstrates the procedure.

Example 3: Again, consider the circuit $\mathcal{C}$ shown in Figure 1 and the given falling PDF on path $a - d - g - j - m$. Each signal line x in $\mathcal{C}$ is represented by three Boolean variables x_1, x_2 and x_S. Here, x_1 (x_2) represents the value of x in the initial (final) time frame. The variable x_S determines whether the value at x is guaranteed to be static. In order to generate an ARAP test, the circuit is transformed into PB constraints gate by gate:

$$\Psi_{\mathcal{C}} = \psi_d \cdot \psi_e \cdot \psi_f \cdot \psi_g \cdot \psi_h \cdot \psi_i \cdot \psi_j \cdot \psi_k \cdot \psi_l \cdot \psi_m$$

[2]Note that state-of-the-art PBO solver try to minimize $\mathcal{F}$. Therefore, the negative constant is necessary to maximize the number of static side inputs.

The constraints for generating a non-robust test are added (b=X1,c=X1,l=X0):

$$\Psi_{\text{NR}} = (b \geq 1) \cdot (c \geq 1) \cdot ((1 - l) \geq 1)$$

Finally, the objective function is formulated:

$$\mathcal{F}(b_S, c_S, l_S) = (-1) \cdot b_S + (-1) \cdot c_S + (-1) \cdot l_S$$

A solution for $\Psi_{\mathcal{C}} \cdot \Psi_{\text{NR}}$ with respect to $\mathcal{F}$ determined by a PBO solver provides the ARAP test with maximized number of static side inputs:

$$\text{Test} = \{a = 10, b = S1, c = S1, e = X1\}$$

V. Considering the Presence of Small Delay Defects

The distribution of *Small Delay Defects* (SDDs) has significantly increased in the last years due to the shrinking feature sizes. The growing distribution of SDDs results in an increased likeliness of delay defects on long paths [7], [23] due to their possible accumulation. The test generation procedure described in the previous section considers all side inputs in the same manner. No side input is prioritized. However, this is not optimal if SDDs are considered. This section shows how the presence of SDDs can be incorporated into the proposed test generation procedure. Note that the aim of the proposed method is not to detect SDDs but to avoid that tests are invalidated by SDDs.

Crucial in this incorporation is the consideration of path lengths. When the maximum number of static side inputs is the only objective of the generation of ARAP tests, some longer paths ending at side inputs can be ignored in favor of short paths ending at side inputs. Since shorter paths are typically easier to justify than longer paths, the test generation procedure is more likely to set static values on side inputs with short paths than on side inputs with long paths. However, the likeliness that a test is invalidated by a delay defect caused by SDDs is higher if a longer path ends at a side input. Therefore, side inputs with longer paths should be prioritized. Consequently, it is desired that static values should be set to side inputs with longer paths.

Example 4: Figure 2 gives the example circuit which is also shown in Figure 1 and used in Example 2. A test is shown which also sets a static value to two of the three side inputs. However, this test avoids its possible invalidation at the side input l which is set to a static value. In the presence of SDDs, this test is to be preferred since the side input with longer paths is covered by a static value. The test of Example 2 sets the same number of static side inputs. However, primary inputs are chosen only. The likeliness that SDDs cause delay defects on inputs is very low due to the short path length. Unfortunately, test generation algorithms are more likely to generate tests with static values on side inputs with short paths since their justification is typically easier.

In order to prioritize side inputs with longer paths, the PBO formulation is modified. The PBO instance $\Psi_{\mathcal{C}}^F$ for fault F in $\mathcal{C}$ is built in the same way as described in Section IV. However, the objective function $\mathcal{F}$ is formulated differently, since $\mathcal{F}$ is responsible for the maximization criterion. In the procedure described above, the constant (-1) used for each variable in

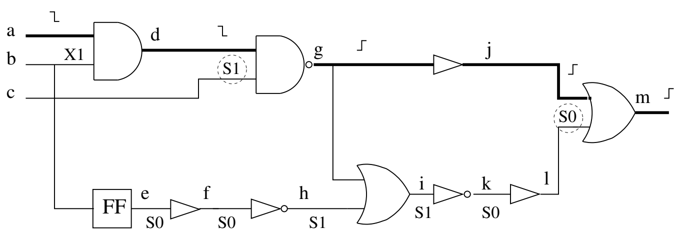


Fig. 2. ARAP test for path $a - d - g - j - m$ considering SDDs

$\mathcal{F}$ was responsible for the equal treatment of each side input. This constant is substituted by a value indicating the likeliness of the invalidation by SDDs. In this paper, the length of the longest path ending at side input $s \in S_S^{\mathcal{P}}$ was chosen as an indicator. A static value on a long paths avoid that the test can be invalidated via this path. The length is denoted by c_x for signal x in the following. The objective function is therefore formulated as follows.

$$\mathcal{F}(x_S^0, \ldots, x_S^n) = \sum_{i=0}^{n} (c_{x^i}) \cdot x_S^i$$

By this, longer paths are prioritized since static values at the end of longer paths cause a higher maximum value in $\mathcal{F}$. If the PBO solver has to decide between two side inputs which cannot be set to static values at the same time, the one with the higher constant is chosen. The altered formulation of $\mathcal{F}$ is shown in the following example.

Example 5: In Figure 2, the length of the longest path (under the unit delay model) for side input b and c is 1. Both side inputs are primary inputs. The length of side input l is 6 (path $e - f - h - i - k - l$). Therefore, $\mathcal{F}$ is formulated as follows:

$$\mathcal{F}(b_S, c_S, l_S) = (-1) \cdot b_S + (-1) \cdot c_S + (-6) \cdot l_S$$

Note that this is a heuristic measurement. Incorporating information about the path length leads to the possibility that a test is found which does not have the maximum number of static side inputs possible. However, experiments confirmed that the decrease in the number of static side inputs is negligible.

An advantage of using a PBO formulation is the flexibility of the objective function. Due to the objective function, the side inputs are dynamically ordered. In this paper, we used the path length as indicator for demonstrating the efficiency and feasibility of the technique. This value can easily be replaced by more technology-dependent parameters or *Standard Delay Format* (SDF) information, respectively. Additionally, it is easily possible to extend the formulation to generate high-quality non-robust tests [5]. Here, the side inputs are statically ordered according to the difference between the arrival time of the on input of the path under test and the earliest arrival time of the side input.

VI. Experimental Results

This section provides experimental results of the proposed approach. The experiments were conducted on ITC'99 benchmark circuits as well as on large industrial circuits provided by NXP Semiconductors. The PBO solver used was *clasp* [11]. This solver support both PB constraints as well as CNF constraints.[3] Therefore, clasp is also used as a SAT solver for comparison.

Table III shows the detailed results of the test generation which was performed using the launch-on-capture scheme. Column *Circ.* gives the name of the circuit. The name of the industrial circuits roughly denotes the size, i.e. p3327k contains over 3.3 million elements. For each circuit, 40000 long critical paths with a path length of over 50 elements were extracted (if exists). The total number of extracted paths is given in column *#Paths*. The number of non-robustly testable PDFs is presented in column *#NR*. Column *%Rob.* presents the percentage of robustly testable PDFs among the non-robustly testable PDFs.

Three different approaches were evaluated. The run time of these approaches is given in column *Run Time* in CPU seconds. Column *CNF* gives the run time of CNF-based test generation for generating robust as well as non-robust tests. However, no ARAP test is generated. Column *ARAP* gives the run time for the proposed ARAP test generation (including robust tests). The run time overhead for generating ARAP tests is in most cases only about 2X and ranges between 1.1X (b15) and 2.3X (b21) for the benchmark circuits and between 1.7X (p3327k) and 2.8X (p1330k) for the industrial circuits. In contrast, the quality of the tests significantly increases. Column *%ARAP* reports the percentage of generated ARAP tests. In fact, each of the robustly untestable faults could be tested by an ARAP test since at least one input could be set to a static value. The average percentage of side inputs which could be set to a static value is given in column *%St_Inp*. The results show that the average percentage is very high ranging between 74%–92%. Most of the side inputs can be set to static value. This confirms the large quality gap between non-robust and robust tests. The minimum percentage of static side inputs is also given in column *%Min_Inp*.

The run time for ARAP test generation considering SDDs is presented in column *SDD*. Here, the run time overhead is higher compared to ARAP test generation. The overhead ranges between 1.1X (b15) and 3.6X (b21) for the benchmark circuits and between 2.3X (p57k) and 6.9X (p1330k). This is due to the fact, that more solutions have to be considered because of the incorporation of different factors for each side input. In order to measure the quality improvement of this method, all side inputs set to a static value are considered in each test and the average path length of longest paths ending at these side inputs is calculated. On these paths, accumulated SDDs are not able to invalidate the test. The improvement of the average path length is presented in column *Av.Imp.* The results show that the average path length with static values is increased on average by up to 12%. Column *Max.Imp.* gives the highest improvement which could be achieved for one test. The highest factor of improvement can be obtained for p80k with an increase of 887% for one test.

In summary, the experiments confirm the large quality gap between non-robust and robust tests which can be significantly diminished by the proposed ARAP test generation. The results show that the generation of ARAP tests causes moderate run

[3]The SAT solver clasp won the gold medal in two categories of the SAT competition 2009 and in one category in the PB Evaluation 2009.

TABLE III
EXPERIMENTAL RESULTS - ARAP TEST GENERATION

				Run time			ARAP tests			Length	
Circ.	#Paths	#NR	%Rob.	CNF	ARAP	SDD	%ARAP	%St_Inp.	%Min_Inp.	Av.Imp.	Max.Imp.
b14	1666	354	1.7	35.6	60.7	105.5	98.3	89.3	15.2	+0%	+0%
b15	3696	346	0	83.4	91.6	94.7	100.0	83.5	8.6	+0%	+0%
b17	11396	2427	3.0	365.9	503.2	692.5	97.0	85.7	5.7	+2%	+11%
b18	29214	12198	2.5	2009.1	3577.2	4965.1	97.5	91.8	50.0	+8%	+130%
b20	4646	1648	7.7	275.8	578.5	927.0	92.3	92.6	41.9	+1%	+90%
b21	4510	1485	8.5	247.0	573.3	892.2	91.5	91.4	41.9	+1%	+108%
b22	6716	2235	9.5	344.7	737.4	1158.9	90.5	91.9	30.2	+1%	+7%
p57k	16042	4142	45.9	4646.1	9513.3	10625.0	54.1	87.0	1.0	+2%	+249%
p80k	22062	15034	5.4	6521.4	13698.6	15936.0	94.6	86.9	11.5	+12%	+887%
p99k	12710	7095	6.8	505.3	1393.5	1895.4	93.2	74.9	5.3	+5%	+46%
p462k	40000	8855	19.0	361.0	838.5	2227.1	81.0	87.7	8.3	+6%	+99%
p565k	40000	9909	14.2	951.2	1715.1	2166.7	85.8	81.8	31.3	+4%	+92%
p1330k	40000	8473	80.2	69.4	192.0	482.1	19.8	83.5	39.3	+10%	+70%
p3327k	40000	17851	22.0	2706.6	4631.1	7715.4	78.0	88.9	1.0	+4%	+66%

time overhead compared to non-robust/robust test generation. However, it is also shown that ARAP increases the quality of the generated tests significantly since ARAP tests contain a very high average number of static inputs. The extended formulation considering the presence of SDDs is able to increase the average path length of side inputs with static values. This decreases the likeliness that the test is invalidated by delay defects caused by SDDs.

VII. Conclusions

The quality of a delay test is determined by the sensitization criterion applied to the side inputs of the path. Since the same sensitization criterion is applied to each gate, there is a large quality gap between non-robust and robust tests. In the paper, we have proposed the generation of *As-Robust-As-Possible* (ARAP) tests in order to diminish the quality gap between non-robust and robust tests. An ARAP test is a test which satisfies the maximum number of conditions needed for a robust test. As a result, a robust test is generated if all robustness conditions can be met or a high-quality non-robust test. The problem is formulated as a *Pseudo-Boolean Optimization* (PBO) problem and solved by a dedicated PBO solver. By this, the test of highest quality possible is automatically generated with one single problem instance.

Additionally, the presence of *Small Delay Defects* (SDDs) can be easily incorporated due to the flexibility of the problem formulation. The problem formulation is modified to prioritize static values on long paths to avoid that SDDs are able to invalidate the test. Experimental results on large industrial circuits show the efficiency and feasibility of the proposed approach. Future work is the application of this technique to transition fault test generation and the evaluation of the impact on the test set size and switching activity.

References

[1] G. L. Smith, "Model for delay faults based upon paths," in *Proceedings of the International Test Conference*, 1985, pp. 342–349.

[2] C.-J. Lin and S. M. Reddy, "On delay fault testing in logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 694–703, 1987.

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

[4] K. Fuchs, M. Pabst, and T. Rössel, "RESIST: a recursive test pattern generation algorithm for path delay faults considering various test classes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 12, pp. 1550–1562, 1994.

[5] K.-T. Cheng and H.-C. Chen, "Generation of high-quality non-robust tests for path delay faults," in *Proceedings of the Design Automation Conference*, 1994, pp. 365–369.

[6] M. Sharma and J. H. Patel, "Testing of critical paths for delay faults," in *Proceedings of the International Test Conference*, 2001, pp. 634–641.

[7] P. Gupta and M. S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Proceedings of the International Test Conference*, 2004, pp. 1053–1060.

[8] V. M. Manquinho and J. P. Marques-Silva, "Satisfiability-based algorithms for pseudo-Boolean optimization using gomory cuts and search restarts," in *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, 2005, pp. 14–16.

[9] N. Eén and N. Sörensson, "Translating pseudo-boolean constraints into SAT," *Journal of Satisfiability, Boolean Modeling and Computation*, vol. 2, no. 1–4, pp. 1–26, 2006.

[10] H. M. Sheini and K. A. Sakallah, "Pueblo: A hybrid pseudo-boolean SAT solver," *Journal of Satisfiability, Boolean Modeling and Computation*, vol. 2, no. 1–4, pp. 165–189, 2006.

[11] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-driven answer set solving," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007, pp. 386–392.

[12] A. Biere, M. Heule, H. v. Maaren, and T. W. (Eds.), *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009.

[13] H. Mangassarian, A. Veneris, S. Safarpour, F. N. Najm, and M. S. Abadir, "Maximum circuit activity estimation using pseudo-boolean satisfiability," in *Proceedings of Design, Automation and Test in Europe*, 2007, pp. 1538–1543.

[14] S. Roy, P. P. Chakrabarti, and P. Dasgupta, "Satisfiability models for maximum transition power," *IEEE Transactions on VLSI Systems*, vol. 16, no. 8, pp. 941–951, 2008.

[15] K. Ganeshpure and S. Kundu, "On ATPG for multiple aggressor crosstalk faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 774–787, 2010.

[16] R. Tayade and J. A. Abraham, "Critical path selection for delay test considering coupling noise," in *Proceedings of the IEEE European Test Symposium*, 2008, pp. 119–124.

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

[18] M. Tehranipoor and N. Ahmed, *Nanometer Technology Designs: High-Quality Delay Tests*. Springer, 2007.

[19] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 1, pp. 4–15, 1992.

[20] M. Anjos, "Pseudo-boolean forms," in *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications, A. Biere, M. Heule, H. v. Maaren, and T. Walsh, Eds. IOS Press, 2009, pp. 49–51.

[21] C. Chen and S. K. Gupta, "A satisfiability-based test generator for path delay faults in combinational circuits," in *Proceedings of the Design Automation Conference*, 1996, pp. 209–214.

[22] S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and R. Drechsler, "MONSOON: SAT-based ATPG for path delay faults using multiple-valued logics," *Journal of Electronic Testing: Theory and Applications*, vol. 26, no. 3, pp. 307–322, 2010.

[23] B. Kruseman, A. K. Majhi, G. Gronthoud, and S. Eichenberger, "On hazard-free patterns for fine-delay fault testing," in *Proceedings of the International Test Conference*, 2004, pp. 213–222.