Robust Timing-Aware Test Generation Using Pseudo-Boolean Optimization

Stephan Eggersglüß*†
*University of Bremen, Germany
†German Research Center for Artificial Intelligence (DFKI), Bremen - Cyber-Physical Systems
{Stephan.Eggersgluess@dfki.de}

Mahmut Yilmaz‡
‡NVIDIA, Santa Clara, CA 95050, USA
{myilmaz@nvidia.com}

Krishnendu Chakrabarty§
§Electrical and Computer Engineering
Duke University, NC 27708, USA
{krish@ee.duke.edu}

***Abstract*—Advances in the chip manufacturing process impose new requirements for post-production test. *Small Delay Defects* (SDDs) have become a serious problem during chip testing. Timing-aware ATPG is typically used to generate tests for this kind of defects. Here, the faults are detected through the longest path. In this paper, a novel timing-aware ATPG approach is proposed which is based on *Pseudo-Boolean Optimization* (PBO) in order to leverage the recent advances in solving techniques in this field. Additionally, the PBO-based approach is able to cope with the generation of hazard-free robust tests by extending the problem formulation. As a result, the faults are detected through the longest robustly testable path, i.e. independently from other delay faults. Experimental results show that a hazard-free robust test can be efficiently found for most testable timing-critical faults without much reduction in path length.**

I. Introduction

Delay testing is typically performed to ensure that a produced chip meets its timing specification. A serious issue during the post-production test is the growing distribution of *Small Delay Defects* (SDDs). A SDD is a defect with defect size not large enough to cause a timing failure on its own. However, SDDs might cause a timing violation when many of them are accumulated. Due to the shrinking feature sizes and the increased speed of today's circuits, the likelihood of failures caused by SDDs increases and their detection has become a critical issue [1]. An SDD might escape during test application when a short path is sensitized since the accumulated delay of the distributed defect is not large enough to cause a timing violation. In contrast, the same SDD might be detected if a long path is sensitized [1], [2]. Unfortunately, common ATPG algorithms usually prefer short paths since the sensitization of these paths is typically much easier.

Timing-aware ATPG was proposed in [3], [4]. Here, pre-calculated timing information is used during ATPG to guarantee sensitization of the longest path. By this, the test is more likely to detect SDDs. However, timing-aware ATPG is a computationally intensive task, since the search space is huge. As a result, the run time of timing-aware ATPG increases significantly compared to regular ATPG as reported in [5].

Besides the length of a path, a delay test can be classified with respect to different quality levels [6], e.g. non-robust and robust. A (hazard-free) robust test promises highest quality since it guarantees the detection of the target fault independently from other delay faults. Robust tests are more desirable but typically harder to obtain. Classical timing-aware ATPG does not consider robust test generation. Therefore, the fault might not be detected via the longest path since it may be masked by other delay faults present in the circuit. A drawback of commercial ATPG tools is that they do not provide robust tests. The fault model ALAPTF [2] was proposed to launch the transition via one of the longest robust segments ending at the fault site. However, a computational intensive genetic algorithm and a recursive structural ATPG algorithm were used to find long paths. A different metric to judge the quality of a test set is the statistical delay quality level (SDQL) metric [7]. It measures the test escapes for a test set with respect to a given delay defect probability. However, it cannot be used in an early phase since the collection of data from the testing and manufacturing processes is necessary.

An alternative to structural ATPG as used in timing-aware ATPG is ATPG based on *Boolean Satisfiability* (SAT) [8]. Here, the search process does not work on a structural netlist but on a Boolean formula typically in *Conjunctive Normal Form* (CNF). Recent advances in SAT solving techniques led to highly efficient SAT solvers. SAT-based ATPG was shown to be highly fault efficient and the application results in significantly increased fault coverage for large industrial circuits [9], [10]. A key aspect for the robustness of SAT-based algorithms is the inherent conflict-driven learning which efficiently prunes large parts of the search space. Therefore, it is desirable to employ these techniques to timing-aware ATPG as well that these benefits can be leveraged. However, SAT solvers can not directly be used since they do not have the ability to process natural number which is mandatory for incorporating timing information. The approaches presented in [11], [12] use a Boolean encoding to encode natural numbers into a SAT instance. Then, a series of SAT solver calls with different timing assumptions is used in order to find the longest sensitizable path through the fault site or all paths in a specified delay range, respectively. However, these approaches are not able to generate hazard-free robust tests.

A different approach is to apply solvers for *Pseudo-Boolean* (PB) SAT or *Pseudo-Boolean Optimization* (PBO) [13], respectively. Many of the PBO solvers strongly rely on the efficient SAT techniques but in addition are able to process natural numbers in a specific manner. In fact, SAT solvers often form the core engine of state-of-the-art PBO solvers. The applicability of PBO in the field of ATPG has been shown in [14]. Here, As-Robust-As-Possible tests are generated for the path delay fault model. An optimization function is used to satisfy as much robust sensitization conditions as possible for a specified path.

In this paper, we present a novel PBO-based timing-aware ATPG approach which is also able to generate high-quality hazard-free robust tests for the transition fault model. PB-

The work of K. Chakrabarty was supported in part by the US National Science Foundation under grant OISE-1157536.

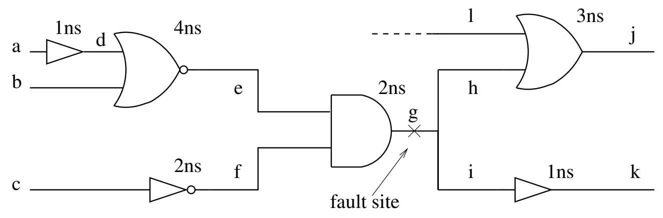

Fig. 1. Example circuit for timing-aware ATPG

SAT constraints are used to model the circuit's logic behavior, fault detection and the path identification while an optimization function is responsible for the longest path calculation. Additionally, the ability of encoding static values which is necessary for the robust sensitization condition is integrated into the problem formulation and coupled with the path identification. As a result, the approach is able – for the first time – to generate a test sensitizing the longest robustly testable path and, by this, increases the quality of the test set. Experimental results for robust and classical PBO-based timing-aware ATPG show that robust tests can be obtained with only small run time overhead or even speed-up.

The paper is structured as follows. Section II presents the timing-aware ATPG problem, different sensitization conditions, and introduces basic information about PBO. Section III shows how the PB constraints and the minimization function are derived for the timing-aware ATPG problem. Static value encoding and the integration into the PBO-based timing-aware ATPG formulation is presented in Section IV. Section V presents experimental results and Section VI gives the summary of this paper as well as an outlook.

II. Preliminaries

A. Timing-Aware ATPG

Common ATPG algorithms tend to sensitize short paths during test generation due to reasons of complexity. However, this is disadvantageous for detecting SDDs. Delay defects based on SDDs are more likely to occur on longer paths, since more SDDs can be potentially accumulated and the slack margin is smaller. This is demonstrated by the following example.

Example 1: Consider the simple example circuit shown in Figure 1. Each gate is associated with a specific delay. Assume that the fault site is line g. There are six possible paths through g on which the transition could be propagated:

- p_1 = a–d–e–g–h–j (10ns)
- p_2 = b–e–g–h–j (9ns)
- p_3 = a–d–e–g–i–k (8ns)
- p_4 = b–e–g–i–k (7ns)
- p_5 = c–f–g–h–j (7ns)
- p_6 = c–f–g–i–k (5ns)

Regular ATPG tools try to find a path on which the transition is propagated as fast as possible. So, it is most likely that a regular ATPG algorithm sensitizes the shortest path p_6, since this is the easiest path to sensitize. If the value is sampled for example at 11ns, the slack margin is very high, i.e. the accumulated defect size has to be at least 7ns for p_6 to detect a delay defect. However, if the ATPG algorithm chooses path p_1, the defect size has to be only 2ns for a detection.

Timing-aware ATPG [4] was developed to enhance the quality of the delay test. Here, a test is generated to detect the transition fault through the longest path by using timing information during the search. The algorithm proposed in [4] is based on structural ATPG and consists of two tasks: fault propagation and fault activation. Each task uses the path delay timing information as a heuristic to propagate (activate) the fault through the path with maximal static propagation delay (maximal static arrival time). However, due to complexity reasons, both tasks are carried out independently and the longest path might be missed. Furthermore, simplifications are assumed to further reduce the complexity. This motivates the need for new techniques that can cope with the high complexity.

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	$S0$	$X0$	$X0$

B. Sensitization Criteria

A test for a delay fault has to consider two time frames t_1, t_2. In order to generate a test for a path delay fault on path p, the desired transition has to be launched at the input and p has to be sensitized to propagate the transition. For this, the side inputs of p, i.e. all connections which are not on the path but feed a gate on a path, have to be constrained to specific values according to the desired sensitization criterion. The sensitization criterion used is responsible for the test quality.

Table I shows the conditions for non-robust as well as robust sensitization [6]. For a non-robust test, it is sufficient that all side inputs of 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 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). Note that often two Boolean variables are used to denote the values of a signal in two discrete points of time, i.e. at t_1 and t_2. Setting both values to either 0 or 1 is not sufficient for the robust sensitization criterion, since the time between t_1 and t_2 is not considered and hazards or glitches could mask the delay fault.

C. Pseudo-Boolean Optimization

In this section, basic information about *Pseudo Boolean Optimization* (PBO) and the related *Pseudo-Boolean* (PB)-SAT problem is given [15] (cf. [14]). A pseudo-Boolean formula Ψ 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:

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

where $c_0, \ldots, c_n \in \mathbb{Z}$ and $x_i \in \{0,1\}$. 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_{PB} is satisfied if and only if each constraint $\psi \in \Psi$ is satisfied.

The PB-SAT problem is to find an assignment that satisfies Ψ_{PB} or to prove that no such assignment exists. The PBO problem is to find the satisfying assignment of Ψ_{PB} which

TABLE II
PB AND CNF REPRESENTATION FOR AN AND GATE $a \cdot b = c$

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

is optimal, e.g. minimal, with respect to a given objective function $\mathcal{F}$:

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

where $m_0,\ldots,m_{n-1} \in \mathbb{Z}$. The task of a PBO solver is therefore to find the assignment which satisfies Ψ_{PB} and, at the same time, minimizes $\mathcal{F}$ or to prove that no satisfying assignment exists.

A circuit-oriented problem can be formulated as a PB-SAT problem as follows (similar to the SAT transformation [8]): each signal s_j in a circuit is assigned a Boolean variable x_j. Then, the circuit's logic has to be transformed into PB constraints gate by gate by creating a set of constraints ψ_g for each gate g. The similarity to SAT transformation is shown in Table II, where the CNF as well as the PB-SAT representation of an AND gate is given. Each CNF clause corresponds to a PB constraint. 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:

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

In practice, $\Psi_{\mathcal{C}}$ is then extended with problem-specific constraints Ψ_F which are for example needed for fault propagation and activation. Then, the derived PB-SAT instance Ψ_{PB} which can be given to a PB-SAT solver to compute a test is as follows:

$$\Psi_{\text{PB}} = \Psi_{\mathcal{C}} \cdot \Psi_F$$

Typically, PBO solvers like clasp [16] internally translate the problem into a SAT instance and work in a iterative manner: a PBO solver calculates an initial solution at first (corresponding to a PB-SAT solution) which is then improved in the following until no better solution can be found. Generally, the search space of such a problem is huge and typically many iterations are needed to find the minimal solution. However, PBO solvers use efficient conflict-based learning techniques and effective heuristics 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. Therefore, PBO solvers have the potential to cope with the high complexity of the timing-aware ATPG problem.

III. PBO-BASED TIMING-AWARE ATPG

This section describes how the timing-aware ATPG problem is represented as a PBO problem, i.e. as a PB-SAT instance Ψ_{PB} and a minimization function $\mathcal{F}$. We first describe in Section III-A how the PB-SAT instance is composed and how the minimization function is derived. Afterwards, Section III-B presents details about the implications and constraints which have to added to the PB-SAT instance in order to guarantee a consistent path representation.

A. PB-SAT and Minimization Function

The use of PB-SAT and PBO, respectively, has the advantage that the efficient solving and search space pruning techniques of state-of-the-art solvers can be applied to solve the specific problem. However, the correct and complete formulation as a PBO problem instance is crucial for the efficient application. As stated above, the use of a PBO solver requires the creation of a PB-SAT instance Ψ_{PB} and a minimization function $\mathcal{F}$. The proposed PB-SAT formulation is based on the SAT formulation for ATPG proposed in TEGUS [17]. As shown above, any SAT instance can be transformed into a PB-SAT instance in a straightforward manner but not vice versa. The test generation formulation consists of the following parts:

- $\Psi_{\mathcal{C}}$ describes the logic of the necessary circuit parts. Note that two consecutive time frames t_1, t_2 have to be considered for transition test generation. A signal x is therefore associated with two variables x_1, x_2 representing the value of the line in the corresponding time frame.
- Ψ_F describes the faulty part of the circuit. That is the fault site as well as the logic of the faulty output cone. An additional variable y^f is assigned to each signal y in the faulty output cone which represents the value of y in the faulty part.
- Ψ_D describes additional constraints necessary for fault propagation and fault observation. In particular, these constraints make sure that a D-chain exists, i.e. there exists a path from the fault site to an observation point along which the fault is propagated. An additional variable y^D (also called D variable) is associated with each signal y in the faulty output cone. This variable is 1 if the fault is propagated to an observation point along this line.

This formulation is extended for the problem of finding the longest path through the fault site. Here, a clear path representation is needed for identifying the longest path automatically by the solver used. The last part of the formulation, i.e. Ψ_D already includes a propagation path representation by the D variables of the output cone. When the variable y^D of signal y is assigned to 1, the fault is propagated along line y. Therefore, the propagation path is represented by the set of lines whose D variable is 1. More formally, let Y be the set of lines in the output cone of the fault site, then the propagation path P^p is represented as follows:

$$P^p = \{y \in Y : \; y^D = 1\}$$

However, this representation has to be extended, since it covers the propagation path only. The activation path has to be considered for identifying the longest path, too. Generally, setting the desired transition value at the fault site is sufficient for the solver used to create an activation path. However, additional information is required for path identification. Therefore, a J variable z^J is assigned to each line z in the support of the fault site. This is illustrated in Figure 2. Note that the signal line of the fault site is assigned a D variable as well as a J variable. Both variables of the fault site are fixed to 1 in the problem formulation to trigger the search.

The J variable z^J of line z is 1 if the line carries a transition along the activation path. Therefore, similar to the representation of the propagation path P^p, the activation path P^a is represented by those lines whose J variable is assigned to 1. More formally, let Z be the set of lines in the support of the fault site, then the activation path P^a is represented as

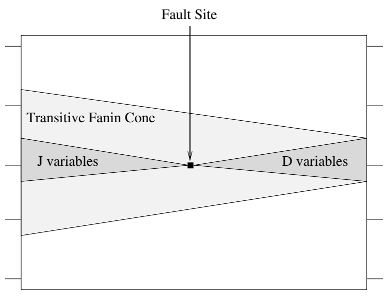

Fig. 2. D and J Variables in PB-SAT transformation

follows:

$$P^a = \{z \in Z : \ z^J = 1\}$$

Note that the constraints which guarantee the correct assignment of the D and J variable are given in Section III-B. Eventually, the complete path P^f for fault activation as well as for fault propagation is derived by the union of P^a and P^p:

$$P^f = P^a \cup P^p$$

This path representation allows the solver to identify the path by checking the assignment of the D and J variables. This is then used to create the minimization function which is responsible for identifying the longest path. Therefore, the minimization function $\mathcal{F}$ consists of the D as well as of the J variables of the given instance. In addition, to incorporate the delay aspect, each variable x in the minimization function is associated with a static delay value d^x (obtained for instance by static timing analysis) which represents the delay of the line as well as the delay of the predecessor gate:[1]

$$\mathcal{F}(Y^D, Z^J) \ = \ \sum_{i=1}^{n} -d^{y_i} \cdot y_i^D \ + \ \sum_{j=1}^{m} -d^{z_j} \cdot z_j^J$$

The result of $\mathcal{F}$ is the accumulation of the delay values of the activated variables, i.e. those variables which are assigned to 1 in the current assignment. Given to a PBO solver, the ultimate solution is the assignment which minimizes $\mathcal{F}$. This directly corresponds to the longest path through which the transition fault is detected.

B. Constraints for Consistent Path Representation

This section shows which constraints or implications have to be added to the PB-SAT instance to guarantee a correct and consistent path representation. This includes the following properties:

- It has to be guaranteed that the transition is activated and propagated along at least one path. These constraints are needed for fault detection and are described by Ψ_{path}.
- It has to be ensured that the D and J variables of *exactly one path* are assigned to 1, although there exist multiple paths along which the transition is propagated or activated, respectively. This is especially important since the minimization function $\mathcal{F}$ is defined over *all* D and J variables. The solver tries to assign as many as possible of these variables with the value 1. These constraints are described by Ψ_{one}.
- Different arrival times of transitions at gate inputs have to be considered in order to make sure that the correct path which causes the transition at the output is identified. This is described by Ψ_{tran}.

[1]Note that the delay value is given in $\mathcal{F}$ as a negative value, since state-of-the-art PBO solvers typically perform minimization but not maximization.

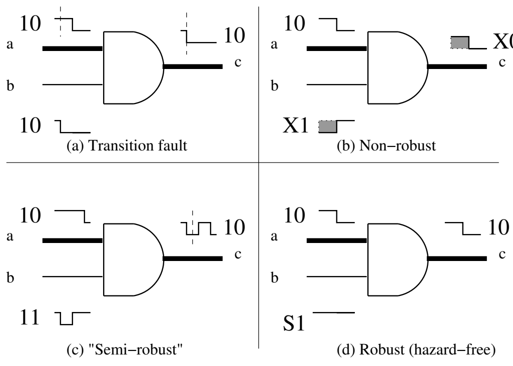

Fig. 3. Different sensitization examples

In summary, the PB-SAT instance Ψ_{PB} which incorporates these properties is derived as follows:

$$\Psi_{\text{PB}} = \Psi_{\mathcal{C}} \cdot \Psi_{F} \cdot \Psi_{\text{path}} \cdot \Psi_{\text{one}} \cdot \Psi_{\text{tran}}$$

The precise implications or constraints, respectively, in PB form included in Ψ_{path}, Ψ_{one} and Ψ_{tran} cannot be given here due to page limitation.

IV. Longest Robustly Testable Paths

Although the longest path has been sensitized, a different delay fault present in the circuit might mask the targeted delay fault on the longest path. Different sensitization criteria were developed to prevent this. This is demonstrated in the following example.

Example 2: Consider the AND gate $a \cdot b = c$ in Figure 3. Here, different sensitization possibilities and the associated problems are presented. Typically, transition fault testing (a) constrains the output only. That is, only a transition has to be assumed at the output c. In particular, this leads to the possibility that a delay fault on the path is masked if the transition on the side input b arrives earlier than the transition on the on-path input a. Non-robust sensitization (b) ignores the behavior on the side input during the initial time frame. Therefore, a delay fault on a could be masked by a delay fault on side input b.

Semi-robust sensitization (c) is often used to prevent the invalidation of the test by other delay faults. Here, the initial and the final value of the side inputs are constrained to the non-controlling value. However, the behavior between the initial and final time points is unknown. Therefore, a glitch is able to invalidate the test as shown in Figure 3(c). Here, the glitch (originated from the side input b) is propagated on a and the correct value is (wrongly) sampled at the output. Robust sensitization (d) requires the guarantee of a static value on all side inputs. By this, the test cannot be invalidated nor masked out by other delay faults or glitches (“hazard-free”).

In [18], a third Boolean variable x^S is assigned to each signal line x (*S variable*). If $x^S = 1$, the signal is guaranteed to be static. Constraints Ψ_S are added to ensure that no glitch or hazard occurs by means of static value justification. Basically, these constraints guarantee that a static value on the output of a gate is caused by its gate input assignment. That is either by static non-controlling values on all gate inputs for a non-controlling static value on the output or by a static controlling value at least on one gate input for a controlling value on the gate output. By this, the static values are justified by the assignment of the primary or pseudo primary inputs. However, robust test generation is typically applied for the path delay fault model where the path under test is completely specified as in [14]. For timing-aware ATPG, the path is not specified. The search for the longest path is explicitly part of the problem. The following formulation is proposed to integrate the robust sensitization conditions into the proposed PBO-based timing-aware ATPG approach in order produce a test with the longest robustly testable path through the fault site.

At first, each gate g in the considered circuit part is assigned an additional variable g^S as described above. The constraints Ψ_S are added for each gate to the circuit formula in order to enable static value representation. Note that Ψ_S is encapsulated from the original circuit formula and serves only for the correct calculation of the S variables. The aim is now to couple the path representation proposed in the previous section with the ability to guarantee and justify static values in order to enable the robust sensitization criterion. That means, whenever a gate is on a sensitized path, the robust sensitization condition has to be satisfied in addition to the conditions for fault detection presented in the previous section.

Since a consistent path representation exists, i.e. whenever a D or J variable is assigned with 1, the path segment is sensitized, then this variable has to be linked with the robust sensitization condition. In other words, if a gate g or the corresponding output connection, respectively, is on a sensitized path and the transition is from the non-controlling value to the controlling value of the successor gate h, then all other side inputs of h have to assume a static non-controlling value. More formally, given a connection g and a successor gate h. Let Q be the set of gate inputs of h and $P = Q \setminus \{g\} = \{p_1, \ldots, p_{n-1}\}$, the following implications have to be added to the problem instance[2]:

$$(g_2 = cv) \cdot (g^D = 1) \rightarrow (p_1^S) \cdot \ldots \cdot (p_{n-1}^S)$$

The transformation of this implication for a gate h with n inputs results in $n \cdot (n-1)$ PB-SAT constraints and is denoted by Ψ_R. The problem formulation for obtaining the longest robustly testable path is therefore given by the following formula. The parentheses are used to highlight the encapsulation of the problem into a fault detection part and a robust sensitization part.

$$\Psi_{\text{PB}} = (\Psi_C \cdot \Psi_F \cdot \Psi_{\text{path}} \cdot \Psi_{\text{one}} \cdot \Psi_{\text{tran}}) \cdot (\Psi_S \cdot \Psi_R)$$

Because the formulation of the robust sensitization conditions is encapsulated, the optimization function has not to be modified, since the path representation has not been changed in the problem instance. In order to keep the fault coverage high, it is easily possible to deactivate the constraints Ψ_R by incremental assumptions [19] if the fault is generally testable but robustly untestable. By this, a classical timing-aware test can be generated using the information learned in the previous search process.

V. Experimental Results

This section presents the experimental results for the proposed PBO-based timing-aware ATPG approach. The approach was implemented in C++ and clasp [16] was used as underlying PBO solver. The experiments were conducted on an AMD Phenom (3400MHz, 8192MB, GNU/Linux) using the IWLS 2005 benchmarks without any test-related modifications, e.g. test point insertion. Timing information for these circuits were obtained by HSpice and Monte Carlo simulation using 45 nm technology. Test generation was performed using the launch-on-capture scheme (broadside tests).

Table III shows the experimental results. Two different setups were used. The upper part shows results for performing test generation for all targets, i.e. transition faults on inputs and branches. Fault dropping is disabled, i.e. 263,440 targets for circuit ethernet corresponds to 263,440 ATPG calls. The integration of timing-aware fault dropping and test set compaction is future work.

The lower part shows the results for timing-critical faults only as proposed in [20]. Here, only transition faults are targeted that lie on structural longest paths within 25% of the clock cycle. The timing of the structural longest path is assumed to be the duration of a clock cycle (column *clk* in *ps*). The average (*av.*) and the maximum (*max*) length of the sensitized path for classical transition fault test generation are given in column *Classic TF* for comparison. Results of (PBO-based) timing-aware ATPG are given in column *Classic Timing-aware* while column *Robust Timing-aware* shows the results for the approach using robust sensitization to increase the quality.

Column *test.* gives the percentage of testable faults, while column *ab.* presents the number of aborts, i.e. those faults for which no test could be generated within the limit of 10,000 conflicts. Column *opt.* gives the percentage of testable faults for which no guarantee is given that the optimal solution was found. Note that this does not imply that the generated test is not the optimal solution, i.e. not the longest path. This means that the proof that there is no better solution has not been finished. The total run time in cpu seconds is given in column *cpu*. The percentage of robustly testable faults is given in *rob.*

The results show that PBO-based timing-aware ATPG is very robust. Only very few faults were aborted. The percentage of tests for which the proof of optimality has not been finished is also low.[3] Concerning the average and maximal path length, timing-aware tests are – as expected – significantly longer than classical transition tests.

Concerning robust timing-aware test generation, it has to be pointed out that the number of testable faults is decreased only slightly for most circuits. Another important perception is that the path length of robust timing-aware tests is only marginally decreased in terms of average as well as maximum length compared to classical timing-aware tests. In few cases, robust tests are even longer than classical timing-aware tests, e.g. for mem_ctrl. Here, classical timing-aware ATPG could not find the optimal solution. Also, the cpu time is reduced for a few circuits, i.e. tv80 and usb_funct although there is large overhead in terms of formula size. This can be explained because the solving process has to traverse the complete solution space. Typically, the solution space for robust tests is much smaller than the solution space for classical tests.

If only the timing-critical faults are targeted as often done in practice, similar observations can be made. Although, the num-

[2]The implication is given for the propagation path only. However, the implication for the activation path is obtained by simply substituting the D variable by the J variable.

[3]For those circuits where the opt. value is higher, e.g. systemcaes and aes_core, increasing resources led to higher run time and to significantly decreased opt. value. However, the path length could only marginally improved. This indicates that often the optimal solution has already been found but not proven to be optimal.

TABLE III
EXPERIMENTAL RESULTS ON ATPG RUNS

All faults targeted																
			Classic TF		Classic Timing-aware (proposed, PBO-based)						Robust timing-aware (proposed)					
circ.	targets	clk	av.	max.	test.	ab.	opt.	av.	max	cpu	rob.	ab.	opt.	av.	max	cpu
tv80	26,228	7,677	2,201	5,150	56.4%	7	9.1%	2,899	6,960	6,354	51.6%	0	4.2%	2,773	6,960	4,490
systemcaes	31,758	12,078	2,740	10,740	69.6%	191	13.2%	3,697	11,890	8,751	61.9%	140	7.8%	3,398	11,890	11,904
mem_ctrl	43,924	5,780	1,110	4,300	43.3%	0	0.8%	1,526	5,000	3,916	42,6%	0	0,1%	1,506	5,150	4,455
ac97_ctrl	47,796	5,664	904	5,460	57.8%	0	<0.1%	982	5,664	65	57.3%	0	0%	981	5,664	64
usb_funct	48,878	5,284	1,107	5,070	81.8%	0	1.0%	1,389	5,284	726	80.7%	10	0.3%	1,372	5,284	602
pci_bridge32	73,978	7,432	1,150	6,660	65,6%	5	1.4%	1,423	7,432	2,520	64,2%	0	1.0%	1,401	7,432	2,756
DMA	75,364	6,650	1,585	4,600	72.3%	490	3.5%	2,137	6,430	17,241	71.6%	65	1.7%	2,094	6,490	27,116
wb_conmax	120,322	4,528	1,407	4,220	76.4%	5	3.0%	1,650	4,528	8,736	71.8%	0	2.7%	1,673	4,528	11,382
ethernet	263,440	12,192	1,819	11,000	65.5%	12	0.4%	1,945	11,500	193,479	64.8%	5	0.4%	1,932	11,500	322,750
des_perf	292,020	6,386	1,141	6,240	100%	0	0,3%	1,983	6,240	21,102	99.7%	0	0.2%	1,969	6,240	34,358
Timing-critical faults only																
tv80	3,414	7,677	2,625	5150	42.4%	2	16.9%	3,312	6,960	1,646	38.6%	0	5.8%	3,379	6,960	1,081
systemcaes	862	12,078	6,862	10,740	65.5%	9	18.8%	9,862	11,890	254	65.3%	0	16.8%	9,310	11,890	271
mem_ctrl	2,564	5,780	1,486	4,300	34.1%	0	6.2%	2,377	5,000	1,302	33,5%	0	0,7%	2,358	5,150	994
ac97_ctrl	670	5,664	5,664	5,664	50.0%	0	0%	5,664	5,664	<1	50.0%	0	0%	5,664	5,664	<1
usb_funct	1,296	5,284	4,377	5,070	88.0%	0	<0.1%	4,514	5,284	8	88.0%	0	0%	4,514	5,284	8
pci_bridge32	936	7,432	4,768	6,660	53.5%	0	2.8%	5,546	7,370	75	53.5%	0	1.8%	5,654	7,370	106
DMA	3,090	6,650	2,888	4,600	57.5%	0	15.0%	4,371	6,430	2,687	55.1%	0	7.8%	4,362	6,430	4,432
wb_conmax	14,584	4,528	3,035	4,160	76.2%	0	3.1%	3,479	4,420	1,740	68.1%	0	6.4%	3,746	4,420	2,482
ethernet	1,544	12,192	9,936	11,000	39.9%	0	0%	10,448	11,500	24	39.2%	0	<0.1%	10,464	11,500	21
des_perf	550	6,386	4,744	6,240	100%	0	4.7%	4,967	6,240	2,799	100%	0	3.6%	4,976	6,240	4,899

ber of targets is significantly decreased, the total run time is quite high since timing-critical faults are often hard-to-detect. However, the number of aborted faults is negligible. Therefore, the time consuming part is the optimization procedure for these faults. The opt. value is also increased compared to the upper part. It is also shown that most of the testable timing-critical faults are robustly testable and the average and the maximum path length of classical timing-aware test generation and robust timing-aware test generation is very similar. Therefore, the robustness of the test set is significantly strengthened while the ability to detect SDDs is hardly compromised.

VI. CONCLUSIONS

Timing-aware ATPG is important to detect small delay defects. This paper presents a timing-aware ATPG approach based on Pseudo-Boolean Optimization (PBO) in order to leverage the powerful solving techniques in this field. A PBO formulation for the timing-aware ATPG problem is given. Furthermore, it is shown how static values necessary for generating robust tests can be integrated in this formulation. As a result, the approach is able to generate tests which detect the fault through the longest robustly testable path. The experimental results show that for most testable timing-critical faults a robust test can be efficiently generated with few or even no impact on the path length. By combining timing-aware ATPG and robust test generation, the quality of the test set can be significantly increased.

Future work is the development of novel fault dropping schemes based on timing and sensitization criterion as well as novel ATPG-specific PBO solving techniques in order accelerate to the search process and diminish the time to find the optimal solution.

REFERENCES

[1] B. Kruseman, A. K. Majhi, G. Gronthoud, and S. Eichenberger, "On hazard-free patterns for fine-delay fault testing," in *Int'l Test Conf.*, 2004, pp. 213–222.

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

[3] E. S. Park, M. R. Mercer, and T. W. Williams, "The total delay fault model and statistical delay fault coverage," *IEEE Trans. on Computers*, vol. 41, no. 6, pp. 688–698, 1992.

[4] X. Lin, K.-H. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware ATPG for high quality at-speed testing of small delay defects," in *IEEE Asian Test Symp.*, 2006, pp. 139–146.

[5] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test pattern selection for screening small-delay defects in very-deep submicron integrated circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 760–773, 2010.

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

[7] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara, "Invisible delay quality – SDQM model lights up what could not be seen," in *Int'l Test Conf.*, 2005, pp. 1202–1210.

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

[9] 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 Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1329–1333, 2008.

[10] S. Eggersglüß and R. Drechsler, "Efficient data structures and methodologies for SAT-based ATPG providing high fault coverage in industrial application," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 9, pp. 1411–1415, 2011.

[11] M. Sauer, A. Czutro, T. Schubert, S. Hillebrecht, I. Polian, and B. Becker, "SAT-based analysis of sensitisable paths," in *IEEE Symp. on Design and Diagnosis of Electronic Circuits and Systems*, 2011, pp. 93–98.

[12] M. Sauer, J. Jiang, A. Czutro, I. Polian, and B. Becker, "Efficient SAT-based search for longest sensitisable paths," in *IEEE Asian Test Symp.*, 2011, pp. 108–113.

[13] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 155–225, 2002.

[14] S. Eggersglüß and R. Drechsler, "As-Robust-As-Possible test generation in the presence of small delay defects using pseudo-Boolean optimization," in *Design, Automation and Test in Europe*, 2011, pp. 1291–1297.

[15] 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.

[16] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-driven answer set solving," in *Int'l Joint Conf. on Artificial Intelligence*, 2007, pp. 386–392.

[17] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1167–1176, 1996.

[18] 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.

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

[20] X. Lin, M. Kassab, and J. Rajski, "Test generation for timing-critical transition faults," in *IEEE Asian Test Symp.*, 2007, pp. 487–492.