On Timing-Aware ATPG 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*—The shrinking feature sizes of the manufacturing process lead to high requirements on the post-production test. The high distribution of *Small Delay Defects* (SDDs) becomes a serious issue for the correct functionality of the manufactured design. Timing-aware ATPG targets the detection of faults through the longest paths in order to detect defects caused by distributed SDDs. However, this results in high CPU run times due to the large search space. This paper serves as the theoretical basis for the application of algorithms for *Pseudo-Boolean Optimization* (PBO) in order to leverage the recent advances in efficient search space pruning techniques in this field. We show how the problem of detecting a transition fault through the longest path can be formulated as a PBO problem. The proposed PBO formulation encodes the timing of the path into the minimization function using structural information. Additionally, the method is able to take transition-dependent delays into account to model a more realistic behavior.**

I. Introduction

Each manufactured chip is objected to a post-production test in order to filter out defective devices. A serious issue during this 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 delay 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 are typically more easier.

Timing-aware ATPG was proposed in [3]. Here, pre-calculated timing information is used during structural 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 very large. As a result, the run time of timing-aware ATPG increases significantly compared to regular ATPG as reported in [4]. This problem is expected to become even more serious in future due to the growing complexity of the designs. Furthermore, due to the high complexity of this task, simplifications are assumed to reduce the run time. As a result, the longest path might be missed.

An alternative to structural ATPG as used in timing-aware ATPG is ATPG based on *Boolean Satisfiability* (SAT) [5]. 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 [6], [7]. A key aspect for the robustness of SAT-based algorithms is the inherent conflict-driven learning [8] which efficiently prunes large parts of the search space. Additionally, information learned from one fault can be used to prune parts of the search space for other faults as well to strengthen the robustness of the overall ATPG process [9], [10]. 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.

One possibility is to apply solvers for *Pseudo-Boolean* (PB) SAT [11] or *Pseudo-Boolean Optimization* (PBO) [12], respectively. Many of the PBO solvers, i.e. clasp [13], 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 feasibility and the efficiency of the PBO solvers in the field of testing has been recently shown in [14], where PBO is used to generate high quality path delay tests.

However, the solving process is very different to the existing timing-aware ATPG approach which is based on structural ATPG. The efficiency of PBO solvers is based on the homogeneity of the problem formulation. This allows for the application of efficient procedures. In order to benefit from the efficient PBO techniques, the problem has to be formulated as a PBO problem. In particular, the path identification and the length calculation have to be part of the problem instance, i.e. the PB formula. This is mandatory for the application of PB-SAT or PBO solvers.

In this paper, we present the theoretical basis of how the the problem of detecting a transition fault through the longest path can be represented by PB constraints and a minimization function. Here, PB constraints are used to model the circuit's logic behavior, fault detection and the path identification while the minimization function is responsible for the longest path calculation. Structural information can be used to reduce the

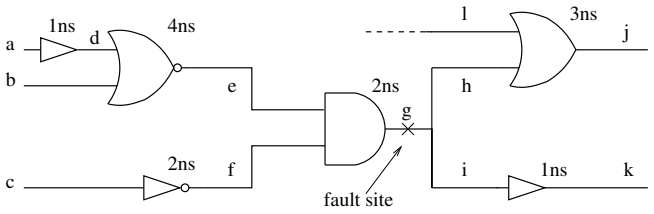

Fig. 1. Example circuit for timing-aware ATPG

size of the problem formulation which typically increases the efficiency. We first consider the unit delay model and extend this formulation to switching-dependent delays to model a more realistic behavior. The paper is structured as follows.

Section II presents the timing-aware ATPG problem and introduces basic information about PBO. Section III shows how the PB constraints and the minimization function are derived using the unit delay model. Section IV presents an improved formulation by using structural information. Section V extends the formulation for more realistic switching-dependent delays. Section VI gives the summary of this paper as well as an outlook.[1]

II. Preliminaries

This section gives the preliminaries of this work. First, the timing-aware ATPG problem is introduced in Section II-A and a corresponding algorithm is briefly reviewed. Then, basic information about the PBO problem and its application to circuit-oriented problems are given in Section II-B.

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.

[1]This paper is meant to present the theoretical basis of the application of PBO on timing-aware ATPG. However, experimental results may be included in future versions of this paper.

Timing-aware ATPG [3] 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 [3] 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.

B. Pseudo-Boolean Optimization

In this section, basic information about *Pseudo Boolean Optimization* (PBO) and the related *Pseudo-Boolean* (PB)-SAT problem is given. 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_{PB} is satisfied if and only if each constraint $\psi \in \Psi_{\text{PB}}$ is satisfied.

The question whether Ψ_{PB} is satisfiable is also known as the PB-SAT problem. The application of PB-SAT is related to the application of SAT. In order to transform a circuit-oriented problem into a PB-SAT problem, the circuit's logic behavior 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 [5], 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 I shows the representation of an AND gate in PB constraints as well 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:

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

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:

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

However, there are two different types of PB-SAT solvers. Solvers like Pueblo [15] directly supports PB constraints, while solvers like MiniSat+ [11] translate the PB-SAT problem

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

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

into a SAT instance and apply regular SAT algorithms to find a solution. Obviously, the latter type of solvers are particularly suited for problems, which can be modeled with many CNF clauses and a few pseudo-Boolean constraints [16].

The PBO problem consists of a pseudo-Boolean formula Ψ_{PB} and an objective function $\mathcal{F}$. 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 (solving the PB-SAT problem) and, at the same time, minimizes the given objective function $\mathcal{F}$.

In order to find the solution which minimizes the given objective function $\mathcal{F}$, 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 minimum 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. PSEUDO-BOOLEAN FORMULATION

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 technique 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:

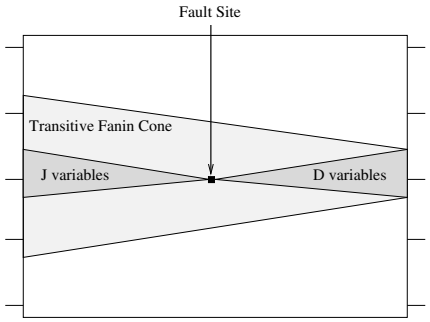

Fig. 2. D and J Variables in PB-SAT transformation

- Ψ_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 start 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 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 by static timing analysis) which represents the delay of the line as well as the delay of the predecessor gate:[2]

$$\mathcal{F}(y_1^D, \ldots, y_n^D, z_1^D, \ldots, z_m^D) \ = \ \sum_{i=1}^{n} -d^{y_i} \cdot y_i^D \ + \ \sum_{j=1}^{m} -d^{z_j} \cdot z_j^J$$

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 in the following 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 1. These constraints are described in the following 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}.

[2]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.

TABLE II
PB REPRESENTATION OF IMPLICATIONS

$y^D = 1 \rightarrow y^g \neq y^f$	$((1-y^D)+(1-y^g)+(1-y^f) \geq 1)$ $\cdot((1-y^D)+y^g+y^f \geq 1)$
$y^D = 1 \rightarrow (z_1^D + \ldots + z_n^D)$	$((1-y^D)+z_1^D+\ldots+z_n^D \geq 1)$
$x^J = 1 \rightarrow x_1 \neq x_2$	$((1-x^J)+(1-x_1)+(1-x_2) \geq 1)$ $\cdot((1-x^J)+x_1+x_2 \geq 1)$
$x^J = 1 \rightarrow (w_1^J + \ldots + w_n^J)$	$((1-x^J)+w_1^J+\ldots+w_n^J \geq 1)$

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

$$\Psi_{\text{PB}} = \Psi_C \cdot \Psi_F \cdot \Psi_{\text{path}} \cdot \Psi_{\text{one}} \cdot \Psi_{\text{tran}}$$

These constraints are given in detail in the following.

1) Ψ_{path}: First, in order to guarantee the propagation of the fault via at least one path from the fault site to an observation point, the following implications (which were used to accelerate the search in [17]) have to included as constraints Ψ_D in Ψ_{PB}.

$$\begin{aligned} y^D = 1 \ &\rightarrow \ y^g \neq y^f \\ y^D = 1 \ &\rightarrow \ (z_1^D + \ldots + z_n^D) \end{aligned}$$

The first implication ensures that the value of the good circuit (y^g) is different from the value of the faulty circuit (y^f). The second implication guarantees that if the D variable y^d of line y is 1, at least one D variable of the successors of y, i.e. $z_1, \ldots, z_n$, has to be 1 in order to ensure the detection of the fault. Similar implications have to be added for the activation path as constraints:

$$\begin{aligned} x^J = 1 \ &\rightarrow \ x_1 \neq x_2 \\ x^J = 1 \ &\rightarrow \ (w_1^J + \ldots + w_n^J) \end{aligned}$$

However, there is no good and faulty value for the fault activation. The desired transition is identified by comparing the value of the line in the initial time frame and the value in the final time frame. Therefore, the first implication ensures that the value of the first time frame (x_1) is different from the value of the second time frame (x_2), i.e. a transition occurs. The second implication guarantees a consistent activation path, since if the J variable of a line is assigned to 1, at least one predecessor of x, i.e. $w_1, \ldots, w_n$ have to assume a transition.

Table II shows the PB representation of the described implications. The implications presented below can be similarly transferred.

2) Ψ_{one}: In order to ensure that D and J variables of exactly one path are assigned to 1, the following implications are needed for a propagation path. Let Y be the set of lines in the output cone. For each line $y \in Y$, the direct predecessors which are itself in Y are given by $p_1, \ldots, p_m$. Note that the fault site is not part of Y. In addition, let Y^{fan} be the set of fanouts in the output cone and let $b_1, \ldots, b_n$ the branches of each fanout $y \in Y^{\text{fan}}$. Then, the following implications are included in Ψ_{one} for each branch and each gate in Y:

$$\begin{aligned} b_i^D = 1 \ &\rightarrow \ \bar{b}_1^D \cdot, \ldots, \cdot \bar{b}_{i-1}^D \cdot \bar{b}_{i+1}^D \cdot \ldots \cdot \bar{b}_n^D | 0 < i < (n+1) \\ y^D = 1 \ &\rightarrow \ p_1^D + \ldots + p_m^D \end{aligned}$$

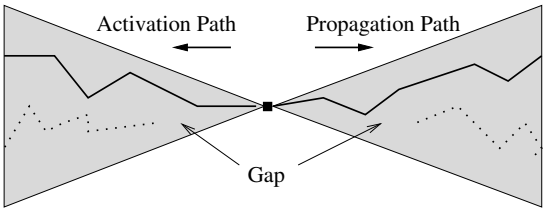

Fig. 3. Multiple paths

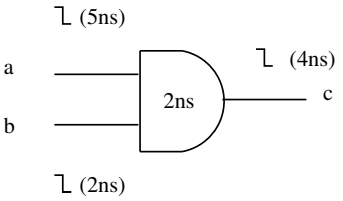

Fig. 4. Origin of transitions

The first implication ensures that for each fanout in the output cone, there is only one branch with a D variable assigned to 1. Note that the implications are unidirectional. Therefore, it is still possible that the fault is propagated along multiple paths. However, only the D variables of one path are activated. The second implication makes sure that D variables can only be assigned to 1 if the D variable of one predecessor is assigned to 1. This ensures that the beginning of the D-chain is the fault site. In summary, these constraints guarantee that the propagation path begins at the fault site and the D variables of exactly one consistent path are assigned to 1.

The constraints needed for the activation path are different, since the direction is contrary. Let Z be the set of lines in the support. For each line $z \in Z$, the direct predecessors are given by $p_1, \ldots, p_m$. In addition, the direct successors of z which are itself in the support are given by $s_1, \ldots, s_n$. Then, the following implications are included in Ψ_{one}:

$$p_i^J = 1 \quad \rightarrow \quad \overline{p}_1^J \cdot, \ldots, \cdot \overline{p}_{i-1}^J \cdot \overline{p}_{i+1}^J \cdot \ldots \cdot \overline{p}_m^J | 0 < i < (m+1)$$
$$z^J = 1 \quad \rightarrow \quad s_1^J + \ldots + s_n^J$$

Similar to the implications of the propagation path, the first implication ensures that only one input of each gate in the support has a J variable assigned to 1. The second implication ensures that the end of the activation path is the fault site.

The importance of especially the second implication is illustrated in Figure 3. The first implication guarantees that the path, i.e. a D and J path assignment, takes only one branch at each fanout or at each gate, respectively. The second implication excludes additional path assignments which do not have their source in the fault site as shown as pointed lines in the illustration.

3) Ψ_{tran}*:* In some cases, the origin of the transition is not clear from the value assignment. This problem is demonstrated in Figure 4. Here, an AND gate is shown which assumes a falling transition on the output c. However, the origin of the transition cannot be determined by the value assignment since both inputs switch in the same direction. Since the final value is the controlling value, the transition is caused by the first occurring transition. However, if the solver can freely choose the path, it would choose the longer path along input a which is wrong, since the transition is caused by input b.

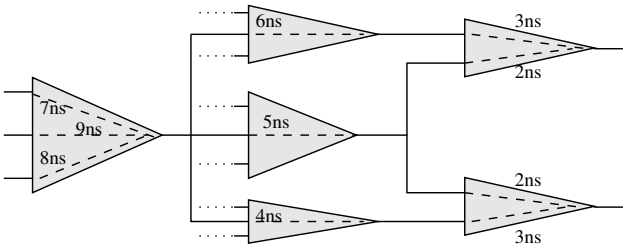

Fig. 5. Using FFRs

Therefore, for each predecessor p_i of line y in the output cone, let $q_1, \ldots, q_n$ be those predecessors of y whose arrival time is later than the arrival time of p_i. Let further cv (ncv) be the controlling value (non-controlling value) for the predecessor gate of y. In order to exclude that the incorrect transition path is chosen, the following implication is needed:

$$((p_i)_1 = ncv) \cdot ((p_i)_2 = cv) \cdot (y^D = 1) \rightarrow \overline{q_1}^D \cdot \ldots \cdot \overline{q_n}^D$$

That is, if the transition on p_i goes from the non-controlling value to the controlling value and y is on a D-chain, than all other inputs of the same gate cannot be on the D-chain. This ensures that the solver has to choose the correct switching input with the smallest arrival time. Note that this does not influence the value assignment of the signal but only the decision which path is the actual D-chain. The same implication is used for the activation path with the J variables instead of the D variables.

IV. Using Structural Information

The inclusion of the different implications in form of constraints in Ψ_{PB} increases the complexity of the PBO instance to solve. In addition, the large number of D and J variables increases the search space given by the minimization function. In order to reduce the search space, an improved formulation is proposed which makes use of structural information.

The circuit can be divided in *Fanout Free Regions* (FFRs). There is a unique path from every input of an FFR, i.e. a branch or primary input, to a fanout or output. Therefore, the basic idea behind the improved formulation is to restrict the D and J variables used in the minimization function to inputs of an FFR. The delay value which is associated with an input i of an FFR represents the delay of the path from i to the output of the FFR. This is illustrated in Figure 5.

This also leads to a reduction of the additional constraints. The formulation of the constraints included in Ψ_{one} and Ψ_{tran} – which are not needed for fault detection but for path identification – can be directly applied on the FFR level instead of the gate level. This is advantageous, since the reduction of constraints typically corresponds to a run time reduction of the solving process.

V. Considering Transition-Dependent Delays

So far, only the unit delay model has been considered. In this model, a static delay value is assumed for a specific line which is independent from the value assignment or possible transition. However, the actual delay is transition dependent,

e.g. the duration of a transition is typically different for a rising and a falling transition. In the following, we propose an extension of the PBO formulation for timing-aware ATPG in order to incorporate transition-dependent delays for a more realistic behavior.

The delay calculation is part of the minimization function $\mathcal{F}$ of the PBO formulation. Each D and J variable which is part of $\mathcal{F}$ are associated with a specific delay value. The delay values of the activated variables are then accumulated and represent the overall delay of the path. We propose to add another variable layer in order to incorporate transition-dependent delays. Each D and J variable are assigned two new variables D_F, D_R and J_F, J_R, respectively. These variables represent the direction of the transition: D_F, J_F (falling), D_R, J_R (rising). Then, the minimization function $\mathcal{F}$ is only defined over these variables and not over the D and J variables anymore.

Each of the variables D_F, D_R and J_F, J_R are then associated with the delay of the specific transition $d_{\{F,R\}}$ which is represented. Therefore, the new minimization function is formulated as follows:

$$\begin{aligned}\mathcal{F} \quad = \quad & \sum_{i=1}^{n} -d_F^{y_i} \cdot (y_i^D)_F + -d_R^{y_i} \cdot (y_i^D)_R \quad + \\ & \sum_{j=1}^{m} -d_F^{z_j} \cdot (z_j^J)_F + -d_R^{z_j} \cdot (z_j^J)_R\end{aligned}$$

In order to guarantee that the new variables D_F, D_R and J_F, J_R always assume the correct value, the following implications are added to the PB-SAT instance:

$$\begin{aligned}(y^D = 1) \cdot (y_g = 0) \quad &\rightarrow \quad y_F^D \\ y^D = 0 \quad &\rightarrow \quad \overline{y}_F^D \\ (y^D = 1) \cdot (y_g = 1) \quad &\rightarrow \quad y_R^D \\ y^D = 0 \quad &\rightarrow \quad \overline{y}_R^D \\ (z^D = 1) \cdot (z_g = 0) \quad &\rightarrow \quad z_F^D \\ z^D = 0 \quad &\rightarrow \quad \overline{z}_F^D \\ (z^D = 1) \cdot (z_g = 1) \quad &\rightarrow \quad z_R^D \\ z^D = 0 \quad &\rightarrow \quad \overline{z}_R^D\end{aligned}$$

These implications ensure that a rising transition and an activated $D(J)$ variable always lead to an activated $D_R(J_R)$ variable and that a falling transition and an activated $D(J)$ variable always lead to an activated $D_F(J_F)$ variable. Furthermore, if the $D(J)$ variable is assigned to 0, the corresponding transition-dependent variable is assigned to 0, too.

The proposed modification of $\mathcal{F}$ leads to a more realistic delay behavior. Extending the formulation for a more fine-grained delay model is easily possible by introducing more variables and adding the corresponding implications.

VI. Summary and Outlook

Timing-aware ATPG is an effective approach to increase the quality of the test set. This paper has given the theoretical basis for applying algorithms for *Pseudo-Boolean Optimization* (PBO) to this problem to make use of the effective solving techniques of this domain. A PBO solver requires as input a *Pseudo-Boolean* (PB)-SAT instance as well a minimization function. We have shown how the identification of consistent paths can be incorporated into a PB-SAT instance. Furthermore, it is shown how the minimization function has to be formulated in order to identify the longest paths through which a transition fault is detected. In addition, this method is extended to take transition-dependent delays into account to model a more realistic delay behavior.

Future work is the experimental evaluation of this approach on practical circuits. Due to the underlying solving techniques, it is expected that the evaluation shows an improvement especially for hard-to-detect faults compared to structural timing-aware ATPG. Furthermore, it is planned to transfer ATPG-specific solving techniques, e.g. [7], [10] from the *Boolean Satisfiability* (SAT) domain to the PBO domain to accelerate the search by structural information.

References

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

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

[3] 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 Symposium*, 2006, pp. 139–146.

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

[5] 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.

[6] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille, "On acceleration of SAT-based ATPG for industrial designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1329–1333, 2008.

[7] S. Eggersglüß and R. Drechsler, "Robust algorithms for high-quality test pattern generation using Boolean satisfiability," in *International Test Conference*, 2010, pp. 1–10.

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

[9] ——, "Robust search algorithms for test pattern generation," in *International Symposium on Fault-Tolerant Computing*, 1997, pp. 152–157.

[10] S. Eggersglüß and R. Drechsler, "Increasing robustness of SAT-based delay test generation using efficient dynamic learning techniques," in *IEEE European Test Symposium*, 2009, pp. 81–86.

[11] 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.

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

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

[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] 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.

[16] 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.

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