SAT-based ATPG for Path Delay Faults in Sequential Circuits

Stephan Eggersglüß Görschwin Fey Rolf Drechsler
Institute of Computer Science, University of Bremen
28359 Bremen, Germany
{segg,fey,drechsle}@informatik.uni-bremen.de

Abstract

Due to the development of high speed circuits beyond the 2-GHz mark, the significance of automatic test pattern generation for Path Delay Faults *(PDFs) drastically increased in the last years. This paper describes an algorithm for generating robust and non-robust tests for PDFs based on* Boolean Satisfiability *(SAT). A new formulation for the robust path delay fault model as a SAT instance is introduced. Unlike previous SAT-based approaches our approach can cope with latches and is therefore applicable for sequential circuits. The formulation provides the possibility to apply the SAT technique* Incremental SAT *to accelerate the process. Experimental results show the efficiency of the approach.*

I. Introduction

The rapidly growing size and the increasing complexity of modern circuit designs cause a continuous need of improvement in the domain of *Automatic Test Pattern Generation* (ATPG). Classical ATPG-algorithms, such as FAN [2], reach their limits, when coping with large designs. Due to the development of high speed circuits beyond the 2-GHz mark, the significance of ATPG for dynamic fault models, e.g. the PDF model, increased in the last years.

Because of the development of efficient techniques in the domain of the *Boolean Satisfiability* (SAT) problem like dynamic conflict learning [6], efficient implementation of implication strategies [7] and efficient search heuristics [3] the significance of this field has grown in the last years. Several SAT-based approaches were presented in the field of ATPG, e.g. [8] for the stuck-at-fault model. For the PDF model, a SAT-based ATPG algorithm for robust tests has been presented in [1]. To model robustness a 7-valued logic is introduced. The mere applicability of the technique *Incremental SAT* (ISAT) [9], which can accelerate the solving process of related instances, for the PDF model was shown in [5]. The approaches listed so far are restricted to combinational circuits. An adequate modelling of sequential dependencies is not given. But this is crucial in industrial practice.

In this paper we consider adequate modelling of sequential dependencies and present a SAT-based algorithm for generating robust and non-robust test patterns. For the generation of robust tests three different encodings are presented and compared to each other. Furthermore the applicability of ISAT is demonstrated.

The paper is structured as follows: In the next section, the application of Boolean Satisfiability to circuit problems and the PDF model is shown. In Section III, we present the formulation for non-robust tests, while the formulation for robust tests is shown in Section IV. Section V introduces the use of ISAT. The experimental results are shown in Section VI. In Section VII we draw conclusions.

II. Preliminaries

A. Boolean Satisfiability

SAT-based algorithms are working on a formula in *Conjunctive Normal Form* (CNF). A CNF Φ in n binary variables is the conjunction of m clauses, each clause is the disjunction of literals. A literal is a variable in its positive or negative form. For modelling the circuit C in CNF, the following notations are needed.

The sequential dependencies of a circuit C are given by a set S of pairs of present states (pseudo primary inputs) and next states (pseudo primary outputs). A pair $(g,h) \in S$ means that g is the present state and h is the next state. A gate $g \in C$ modelled at time t is described by the variable g_t. The corresponding CNF is denoted by $\phi_{g,t}$ and given by a truthtable. The set of predecessors of g is given by $\mathcal{P}(g)$ and the non-controlling (controlling) value is given by $\nu_g^{nc}(\nu_g^c)$, e.g. $\nu_g^{nc} = 1, \nu_g^c = 0$, if g is an AND/NAND gate.

The CNF of a combinational circuit is given by the following equation:

$$\Phi_{C,t} = \prod_{g \in C} \phi_{g,t}$$

The sequential dependencies for two time frames are modelled by:

$$\Phi_{S,t1,t2} = \prod_{(g,h) \in S} g_{t2} \leftrightarrow h_{t1}$$

TABLE I. Off-path constraints

	rising-robust	*falling-robust*	*non-robust*
AND/NAND	$X1$	$S1$	$X1$
OR/NOR	$S0$	$X0$	$X0$

The following equation describes the CNF Φ_C for a circuit including sequential behaviour for two time frames:

$$\Phi_C = \Phi_{C,t1} \cdot \Phi_{C,t2} \cdot \Phi_{S,t1,t2}$$

B. Path Delay Fault Model

The *Path Delay Fault Model* (PDFM) describes a fault (delay) on a path from a (pseudo) primary input to a (pseudo) primary output. A test pattern for a PDF contains two vectors v_1, v_2, one for each time frame. To detect the delay, a transition, which is either *rising* or *falling*, is applied to the input of the path and determines the value of the signal in both time frames.

More formally, a PDF is described by $F = (T, P)$, where $T \in \{\uparrow, \downarrow\}$ determines the type of the transition and $P = (g_1, \ldots, g_n)$ is a path in the circuit, where g_1 is a (pseudo) primary input and g_n is a (pseudo) primary output.

The quality of the test depends on the off-path inputs of the gates along P. Table I shows the constraints of the off-path inputs for a non-robust and a robust test. The values are given in the 7-valued logic presented in [1] and describe the behaviour of the signal in both time frames.

The second element of each value in the table shows the value in $t2$ and the first element describes, whether the value is *static* (S) or *unknown* (X). A signal is static, if the values of $t1, t2$ are the same and no hazard occurs during the interval. In case of unknown, there is no assertion on the value of $t1$ and consequently no assertion, whether the signal is static or not.

Note, that the signal value during $t1$ cannot be clearly specified with the 7-valued logic. Previous approaches using this logic are therefore not able to handle sequential circuits. In contrast, we show a formulation, which can determine the values of both time frames in the following sections.

If there is neither a non-robust nor a robust test under the given constraints, the PDF is redundant.

III. Non-Robust Tests

The SAT-based approach to calculate a non-robust PDF test for a given circuit C and a PDF $F = (T, P)$ is presented in this section.

Essentially, a non-robust PDF test is calculated by solving a CNF formula $\Phi^N_{C,F}$. If $\Phi^N_{C,F}$ is unsatisfiable, the fault is not testable. If the formula is satisfiable, any satisfying assignment directly determines test patterns. The formula $\Phi^N_{C,F}$ is given as follows:

$$\Phi^N_{C,F} = \Phi_C \cdot \Phi_T \cdot \Phi_P$$

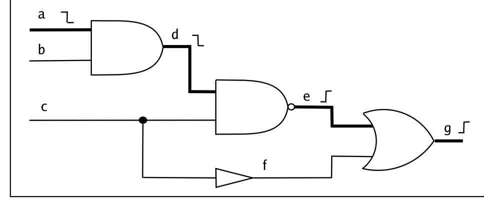

Fig. 1. Example circuit

As explained in Section II-A, Φ_C models the circuit during two consecutive time frames. The constraint Φ_T forces the transition, whereas Φ_P sets all off-path inputs of gates along P to non-controlling values in the second time frame. More formally, for a given fault $F = (T, P)$, where $P = (g_1, \ldots, g_n)$, the constraints Φ_T and Φ_P are described by:

$$\Phi_T = \begin{cases} \prod_{i=1}^{n} \overline{g}_{i,t1} \cdot g_{i,t2}, & \text{if } T = \uparrow \\ \prod_{i=1}^{n} g_{i,t1} \cdot \overline{g}_{i,t2}, & \text{if } T = \downarrow \end{cases}$$

$$\Phi_P = \prod_{i=2}^{n} \prod_{h \in \mathcal{P}(g_i), h \neq g_{i-1}} h_{t2} \leftrightarrow \nu^{nc}_{g_i}$$

Example 1: Assume that the path *a - d - e - g*, shown in Figure 1, with a falling edge is to be tested non-robustly. The falling transition is forced by the following assignments:
$a_{t1} = d_{t1} = e_{t2} = g_{t2} = 1, a_{t2} = d_{t2} = e_{t1} = g_{t1} = 0$
Note, that the transition must be inverted if the path passes an inverting gate. To set the off-path constraints, the non-controlling values of the gates are assigned:
$b_{t2} = c_{t2} = 1, f_{t2} = 0$
A corresponding test would be:
$\{a_{t1} = 1, b_{t1} = x, c_{t1} = x, a_{t2} = 0, b_{t2} = 1, c_{t2} = 1\}$

IV. Robust Tests

One approach to calculate robust tests is to extend the formula $\Phi^N_{C,F}$ by additional constraints Ψ_H that guarentee the absence of hazards. Φ^R_P denotes the additional constraints on off-path inputs for a robust test. The formula for robust tests is given as follows:

$$\Phi^R_{C,F} = \Phi^N_{C,F} \cdot \Psi_H \cdot \Phi^R_P$$

Then, $\Phi^R_{C,F}$ is satisfiable if and only if the assignment determines a robust PDF test. The constraint Ψ_H must guarentee, that a hazard-free signal, applied to an off-path input, is propagated backwards to the inputs. For any gate in C a hazard-free output value implies that at least one input value is controlling and hazard-free or that all input values are non-controlling and hazard-free.

Of course, there is no unique representation of Ψ_H. For this, we study three alternatives and discuss their properties:
(1) the *absolute encoding*
(2) the *extended (ext.) absolute encoding* and
(3) the *relational encoding*

TABLE II. Additional overhead of the different encodings

encoding	# clauses	# literals	# add. variables
1-input gates			
absolute	8	18	1
ext. absolute	10	20	2
relational	6	16	1
2-input gates			
absolute	7	19	1
ext. absolute	5	15	2
relational	11	33	3

The encodings differ in their representation of a hazard-free signal and therefore in the size of the CNF representation.

In the *absolute encoding*, for each gate g an additional variable g_h is introduced, which determines, whether a signal is hazard-free or not. The output value is hazard-free, if $g_h = 1$. This is modelled by the following constraint:

$$\begin{aligned}\Psi_H \;=\; \prod_{g\in C} \{ & (g_h) \\ & \rightarrow [\ \sum_{h\in\mathcal{P}(g)}(h_h = 1, h_{t_1} \leftrightarrow \nu_g^c) \\ & \quad + \prod_{h\in\mathcal{P}(g)}(h_h = 1, h_{t_1} \leftrightarrow \nu_g^{nc})\ ]\}\end{aligned}$$

The disadvantage of this encoding is the exponential number of clauses for gates with more than two inputs. Therefore in the *ext. absolute encoding* we introduce two additional variables g_{hnc}, g_{hc} for each gate. With two variables for each gate, the increase in the number of clauses for each gate is only linear and not exponential. The output value of a gate is hazard-free and has the non-controlling (controlling) value, if $g_{hnc} = 1$ ($g_{hc} = 1$). This is guarenteed by:

$$\begin{aligned}\Psi_H \;=\; \prod_{g\in C} \{ & (g_{hnc}) \\ & \rightarrow [\ \prod_{h\in\mathcal{P}(g)}(h_{hnc} = 1, h_{t_1} \leftrightarrow \nu_g^{nc})]\} \\ \{ & (g_{hc}) \\ & \rightarrow [\ \sum_{h\in\mathcal{P}(g)}(h_{hc} = 1, h_{t_1} \leftrightarrow \nu_g^{c})]\}\end{aligned}$$

Another possibility to encode a hazard-free signal is presented with the *relational encoding*. The additional variable g_{rel} is introduced and describes an arbitrary point in time between $t1$ and $t2$. Moreover, n temporary variables are needed for calculating the value of g_{rel}. where n is given by the number of inputs of g. The output value of a gate is hazard-free, if $g_{t1} \leftrightarrow g_{t_2} \leftrightarrow g_{rel}$. This is guarenteed by the following constraint:

$$\begin{aligned}\Psi_H \;=\; \prod_{g\in C} \{ & (g_{t1} \leftrightarrow g_{t_2} \leftrightarrow g_{rel}) \\ & \rightarrow [\ \sum_{h\in\mathcal{P}(g)}(h_{t1} \leftrightarrow h_{t_2} \leftrightarrow h_{rel} \leftrightarrow \nu_g^c) \\ & \quad + \prod_{h\in\mathcal{P}(g)}(h_{t1} \leftrightarrow h_{t_2} \leftrightarrow h_{rel} \leftrightarrow \nu_g^{nc})\ ]\}\end{aligned}$$

The advantage of this encoding is the smaller CNF representation of gates with only one input. To compare the additional overhead of the different encodings, Table II shows the number of clauses and literals as well as the number of additional variables for a 1-input (e.g. inverter) and 2-input gates.

As stated above, the number of clauses for the CNF representation is exponential in the number of inputs in the *absolute encoding*. Therefore gates with $n > 2$ inputs are modelled as $n-1$ gates with two inputs in this case.

It can be observed that the *relational encoding*, which has the largest number of variables and clauses when modelling 2-input gates, has the smallest number of variables and clauses when modelling 1-input gates.

Example 2: In contrast to Example 1, consider that the path *a* - *d* - *e* - *g* with a falling edge is to be tested robustly. The assignments to the path remain the same. Changes occur at the assignments to the off-path inputs. For a robust test the following assignments are applied:
$b_{t1} = b_{t2} = c_{t1} = c_{t2} = 1, f_{t1} = f_{t2} = 0$
Additionally, the off-path inputs have to be hazard-free. Depending on the encoding used, this is guarenteed by the following assignments:
• $b_{rel} = c_{rel} = 1, f_{rel} = 0$ (relational encoding)
• $b_h = c_h = f_h = 1$ (absolute encoding)
• $b_{hnc} = c_{hnc} = f_{hnc} = 1$ (extended absolute encoding)
A corresponding test would be:
$\{a_{t1} = 1, b_{t1} = 1, c_{t1} = 1, a_{t2} = 0, b_{t2} = 1, c_{t2} = 1\}$

V. Use of Incremental SAT

Generally, robust tests are more desirable than non-robust tests. If no robust test is available, a non-robust test is needed for a testable PDF.

But previous SAT-based approaches considered a multi-valued logic instead of a Boolean logic for PDF test generation. The multi-valued problem was then encoded into a Boolean problem. A single value in the multi-valued logic represents the value of a gate at the second time frame and the information about hazards, e.g. a 7-valued logic was applied in [1]. Therefore, determining the correspondences between an encoded Boolean SAT instance for robust tests and non-robust tests is difficult in this case. Moreover, latches were not modelled.

In contrast, the representations proposed here model the problem at the Boolean level. Hence, it allows to model latches directly. As a result, the model for non-robust test generation can be extended incrementally for robust tests. This can be exploited during the overall test pattern generation process and suggests a two-phase approach for each PDF, which relies on the notion of *Incremental Satisfiability* (ISAT) as introduced in [4], [10].

Essentially, the information that can be learned directly from the model of the circuit can be reused for all PDFs. By keeping the learned information, the efficiency of test pattern generation can be improved. Furthermore, learned information from the generation of a non-robust test can be re-used for efficiently generating a robust test.

In our approach the relation $\Phi_C \subset \Phi^N_{C,F} \subset \Phi^R_{C,F}$ holds. The use of ISAT is therefore restricted to incrementally adding the constraints in two steps:
(1) $\Phi_C \cdot \Psi_H$ is extended incrementally by $\Phi_T \cdot \Phi_P$ to $\Phi^N_{C,F}$
(2) $\Phi^N_{C,F}$ is extended incrementally by Φ^R_P to $\Phi^R_{C,F}$
That is in contrast to the approach presented in [5], where

the path segments are incrementally added to the problem instance.

VI. Experimental Results

ISCAS benchmark results are reported in the following. All experiments are carried out on a Pentium M (2.13 GHz) system (Linux) with 1024 MB RAM. As SAT solver, we used Zchaff [7]. The test procedure works as follows. All possible paths (rising and falling) in the circuit are tested. First a non-robust test is generated. If sucessful, the constraints for the generation of a robust test are added incrementally and a robust test generation follows.

Note that the number of testable paths may differ from those of the previous approaches, since previous approaches overestimate the number of testable paths. This is due to not modelling the sequential dependencies.

First, in Table III the advantage of the use of ISAT is presented using the *relational encoding*. In this table the name of the circuit (*circuit*), the number of paths in the circuit (*# paths*), the number of robust testable paths (*r.test.*) and the number of non-robust testable paths (*n-r.test.*) are shown. The run time for ATPG without using ISAT is presented in column *without ISAT* and the run time with ISAT is given in column *with ISAT*. It can be observed, that the application of ISAT accelerates the generation for all instances up to a factor of 5.

In Table IV the results for the benchmarks are presented for comparing the run time and the number of clauses for the different encodings. The circuit's name is given in the column (*circuit*). The run time (*time*) and the number of clauses (*# cls*) for each encoding are shown in the following columns below the name of the encoding.

It can be observed, that there is only little difference between the run times in the different encodings. The *absolute encoding* seems to be the slowest encoding in most circuits. The results of the *relational* and the *ext. absolute encoding* are almost the same. In a few benchmarks there are advantages for the *ext. absolute encoding*. The relation of the times for the different encodings are mostly reflected by the number of clauses. Therefore a compact representation is desirable for an instance.

VII. Conclusions

A new SAT-based algorithm for the generation of robust and non-robust tests for PDFs was presented. Instead of using a multi-valued logic to generate robusts tests, the problem is directly modelled at the Boolean level. As a result, for the first time a SAT-based algorithm for PDF test generation can handle sequential circuits. To speed up the generation the ISAT-technique was used. Experimental results show the efficiency of the approach.

VIII. Acknowledgement

This research work was supported in part by DFG grant DR 287/15-1.

TABLE III. Results - without ISAT vs. with ISAT

				relational enc.	
circuit	*# paths*	*r. test.*	*n-r.test.*	*without ISAT*	*with ISAT*
s344	710	253	259	2.08s	0.36s
s349	730	253	259	1.52s	0.36s
s382	800	154	165	0.70s	0.34s
s400	896	150	166	0.77s	0.40s
s420	184	37	38	0.08s	0.04s
s444	1070	148	166	0.94s	0.54s
s510	866	221	256	1.42s	0.63s
s641	3444	1007	1100	16.11s	14.35s
s713	43624	831	1097	2465.44s	2422.85s
s820	1166	358	437	3.80s	1.48s
s832	1234	389	429	4.04s	1.65s
s953	1634	533	540	6.02s	2.59s
s1196	2420	1243	1313	18.97s	8.84s
s1238	1044	496	527	6.64s	2.43s
s1488	2096	627	890	13.45s	6.38s
s1494	2246	641	929	15.31s	7.88s
s5378	26758	15886	16739	1461.69s	839.78s

TABLE IV. Results for the different encodings

	absolute enc.		ext. absolute enc.		relational enc.	
circuit	*time*	*# cls*	*time*	*# cls*	*time*	*# cls*
s344	0.84s	2491	0.35s	2717	0.36s	2541
s349	0.87s	2538	0.35s	2766	0.36s	2598
s382	0.52s	3094	0.33s	2934	0.34s	2866
s400	0.37s	3280	0.39s	3104	0.40s	3048
s420	0.06s	2248	0.04s	1394	0.04s	1782
s444	0.52s	3569	0.51s	3423	0.54s	3359
s510	0.80s	6683	0.48s	3609	0.63s	5261
s641	14.64s	4190	14.42s	4316	14.35s	3652
s713	2478.67s	4732	2422.90s	4810	2422.85s	4274
s820	2.22s	11410	1.08s	5482	1.48s	8426
s832	2.46s	11995	1.14s	5697	1.65s	8821
s953	2.28s	7288	2.62s	7678	2.59s	7550
s1196	8.02s	8818	8.27s	8592	8.84s	9124
s1238	2.04s	9211	2.08s	8781	2.43s	9673
s1488	9.76s	19097	4.05s	8663	6.38s	14071
s1494	11.93s	20306	5.06s	9142	7.88s	14922
s5378	831.83s	32917	841.40s	32887	839.78s	30395

References

[1] C. Chen and S. K. Gupta. A satisfiability-based test generator for path delay faults in combinational circuits. In *DAC '96: Proceedings of the 33rd annual conference on Design automation*, pages 209–214, 1996.

[2] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Trans. on Comp.*, 32:1137–1144, 1983.

[3] E. Goldberg and Y. Novikov. BerkMin: a fast and robust SAT-solver. In *Design, Automation and Test in Europe*, pages 142–149, 2002.

[4] J. N. Hooker. Solving the incremental satisfiability problem. *Journal of Logic Programming*, 15(1-2):177–186, 1993.

[5] J. Kim, J. Whittemore, J. P. Marques-Silva, and K. Sakallah. On applying incremental satisfiability to delay fault testing. In *DATE '00: Proceedings of the conference on Design, automation and test in Europe*, pages 380–384, 2000.

[6] J. Marques-Silva and K. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. on Comp.*, 48(5):506–521, 1999.

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

[8] J. Shi, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel. PASSAT: Effcient SAT-based test pattern generation for industrial circuits. In *IEEE Annual Symposium on VLSI*, pages 212–217, 2005.

[9] O. Shtrichman. Pruning techniques for the SAT-based bounded model checking problem. In *CHARME*, volume 2144 of *LNCS*, pages 58–70, 2001.

[10] J. Whittemore, J. Kim, and K. Sakallah. SATIRE: A new incremental satisfiability engine. In *Design Automation Conf.*, pages 542–545, 2001.