Efficient Test Generation with Maximal Crosstalk-Induced Noise using Unconstrained Aggressor Excitation

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

***Abstract*— The influence of crosstalk noise grows as the feature sizes in modern designs decrease. Crosstalk-induced effects are able to cause major timing violations, especially if multiple aggressors affect certain lines. However, conventional *Automatic Test Pattern Generation* (ATPG) algorithms for delay test do not consider these effects during test generation. This increases the possibility that chips which passed the testing phase might fail due to crosstalk-induced effects.**

In this paper, we propose a new efficient ATPG approach for generating delay tests considering crosstalk-induced effects using *Boolean Satisfiability* (SAT). Previous approaches used a two-step procedure to increase the crosstalk-induced noise. As a result, the search space is highly restricted. In contrast, the proposed approach is able to do test generation and excite multiple aggressors in one step. By this, more aggressor combinations can be found and the generated test potentially induce more crosstalk noise on the victim. In order to maximize the crosstalk-induced effects of the test, an exact branch-and-bound algorithm and a static aggressor ordering heuristic are applied and compared. Experimental results demonstrate the efficiency and effectiveness of the approach.

I. Introduction

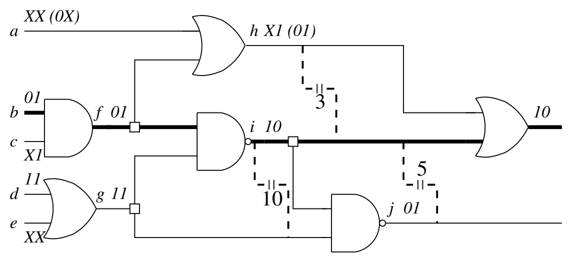
As the feature sizes continue to scale down, the influence of crosstalk-induced effects increases. Crosstalk-induced effects may cause circuit malfunction problems and may be the reason for timing violations [1]. If two or more lines are physically adjacent, crosstalk noise from so-called *aggressor lines* can influence the behavior on the *victim line*. The effects can roughly be categorized into two types: crosstalk-induced glitches and crosstalk-induced delay. A crosstalk-induced glitch is invoked when the victim line is in a static state and the aggressor lines switch. If the victim and the aggressors switch simultaneously, the delay of the transition on the victim line is influenced. Switching in the same direction results in a speed-up of the transition. In contrast, if the aggressors and the victim switch in the opposite direction, the transition delay of the victim line increases.

Crosstalk-induced effects can be reduced by, for instance, redesign techniques [2]. The complete elimination of crosstalk may not be possible due to stringent area and performance requirements. Furthermore, process variations are shown to aggravate these effects [3]. As a result, testing for crosstalk-induced faults is necessary to ensure the correct circuit operation. Though, conventional *Automatic Test Pattern Algorithms* (ATPG) algorithms do not consider crosstalk-induced effects during test generation. First approaches that incorporate these effects, e.g. [4], [5], were focused mostly on single aggressor scenarios. However, it was reported in [6] that long signal nets are typically coupled with 40-50 other lines and maximal crosstalk noise is produced by the simultaneous excitation of multiple aggressors. Due to logical constraints, it may not be possible to excite all aggressors at the same time. Therefore, ATPG approaches dealing with multiple aggressor scenarios [7]–[12] are mainly focused on determining a subset of aggressors which can be excited simultaneously and produces maximal crosstalk noise on the victim line.

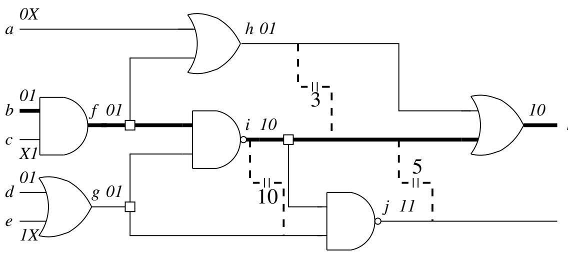
The approach presented in [7] formulates the problem as an implication graph and uses a PODEM-based algorithm to generate the tests. However, the approach produces many aborts, since the underlying engine is not robust enough. In [8], the problem of finding the subset of aggressors exciting maximal crosstalk noise is formulated as an ILP problem. However, a common test is generated at first to activate the propagation path. The generated test is then used as constraint in the ILP formulation. The approach presented in [12] solves the problem using a structural ATPG algorithm. Again, a common test is generated at first and given as constraint to the ATPG tool. Here, a *Static Aggressor Ordering* (SAO) is used to heuristically choose a promising aggressor subset. Furthermore, static timing analysis techniques are used to prune aggressors whose timing windows do not overlap with the victim's timing window. Both approaches have in common, that they use a two-step procedure. In the first step, a test pattern is generated which is used as constraint in the second step. By this, the search space is reduced and the problem is simplified. However, the subset of aggressors exciting maximal crosstalk noise may not be found.

The work presented in [13], [14] employs an algorithm based on *Boolean Satisfiability* (SAT) wrapped by a *Branch-and-Bound* (B&B) algorithm to find the subset of aggressors exciting maximal crosstalk noise on a victim line. As a result, false noise can be reduced in order to provide a more accurate static timing analysis. Since no previously generated test is given as constraint, this approach is guaranteed to calculate the subset of aggressors providing maximal crosstalk noise but without taking test generation into account. The drawback of this approach is the long run time.

In this paper, we propose an efficient SAT-based test generation algorithm for delay tests inducing maximal crosstalk noise on the victim. The new SAT formulation is based on the SAT formulation introduced in [13] but incorporating test generation constraints. In contrast to previous test generation approaches, the approach is able to generate a test and excite multiple aggressors in one step. By this, the search space is not restricted and a potentially larger number of logically valid aggressor combinations can be found. Efficient SAT-based ATPG techniques are used to handle the increased complexity. As a result, the amount of induced crosstalk noise on the victim is significantly increased with acceptable run time overhead. In order to find the aggressor combination exciting maximal crosstalk, two different schemes – an exact



(a) Constrained



(b) Test with maximal induced crosstalk

Fig. 1. Example circuit

B&B procedure [13] and an SAO heuristic [12] – are employed and compared to each other.

The paper is structured as follows. The next section gives a motivational example introducing the drawbacks of previous approaches. Section III briefly introduces SAT and presents the concept of unconstrained aggressor excitation, while Section IV introduces different schemes for aggressor selection. Experimental results are given in Section V and the conclusion is drawn in Section VI.

II. Motivational Example

Consider the example circuit shown in Figure 1. The victim path in the circuit is path *p = (b – f – i – k)* with a rising edge on input b. Three aggressors were previously identified for p: h, g, j. All aggressors are able to induce crosstalk noise on line i on p. The aggressor strengths are $3, 10$ and 5, respectively. Assume that the target is to produce maximal crosstalk-induced delay on the victim.[1] Therefore, the maximal effect is provoked if all aggressors switch in the opposite directions, i.e. all aggressors have a rising edge. In previous approaches, a test pattern which sensitizes p is generated at first. Assume that the generated test is $\{a = XX, b = 01, c = X1, d = 11, e = XX\}$ as shown in Figure 1(a). Here, the first position shows the value in the first time frame, whereas the second position denotes the value in the second time frame.

In order to maximize the crosstalk-induced delay, the aggressors have to be excited in such a way that the effect on the victim is maximized. If applying the SAO heuristic proposed in [12], the aggressors are excited in descending order of their strength. Since the generated test is taken as constraint – as done in previous approaches to reduce search space – it is not possible to excite aggressor g, aggressor j is already excited and aggressor h can be excited by justifying the value 01 on line h. This results in a cumulative strength of 8. However, the aggressor combination inducing maximal crosstalk delay is missed by taking the generated test into account. A test inducing maximal crosstalk delay on the victim path is $\{a = 0X, b = 01, c = X1, d = 01, e = 0X\}$ as shown in Figure 1(b). Here, aggressor g and h are excited, whereas aggressor j cannot be excited due to logical correlations. This results in a cumulative strength of 13 which is more than the strength of the test given in Figure 1(b).

In the following, it is explained how to efficiently find the logically valid aggressor combination inducing maximal crosstalk noise using SAT.

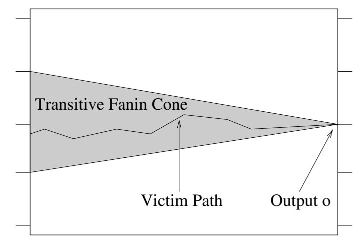


Fig. 2. Influenced circuit parts

III. SAT Formulation: Unconstrained Aggressor Excitation

In this section, a brief introduction of SAT is given. Furthermore, it is explained how an ATPG problem is formulated as a SAT problem. Contrary to classical structural ATPG algorithms, SAT-based ATPG algorithms do not work on the circuit structure but on a Boolean formula in *Conjunctive Normal Form* (CNF).[2] A CNF is a conjunction of clauses and each clause is a disjunction of literals. A literal is a Boolean variable in its positive or negative form. A CNF is satisfied if all clauses are satisfied. A clause is satisfied if at least one of its literals is satisfied. A SAT solver is used to solve the CNF. For more information about SAT solving techniques, it is referred to [15].

In order to apply a SAT solver to an ATPG problem, the problem has to be converted into a CNF Φ_{Test}. Generally, the SAT instance Φ_{Test} consists of two different parts: the considered circuit and the fault-specific constraints. However, it is not necessary to include the complete circuit in Φ_{Test}. For a specific path delay fault f on path p on output o, only the transitive fanin cone $\mathcal{F}(o)$ of output o has to be included (demonstrated in Figure 2). Additionally, the constraints for path sensitization Φ_f must be considered. As a result, the SAT instance for generating a test for F is composed by: $\Phi_{Test} = \Phi_{\mathcal{F}(o)} \cdot \Phi_f$.

To create the circuit CNF $\Phi_{\mathcal{F}(o)}$, each connection $c \in \mathcal{F}(o)$ is associated with a Boolean variable x_c and each gate $g \in \mathcal{F}(o)$ is converted to a set of clauses Φ_g: $\Phi_{\mathcal{F}(o)} = \prod_{g \in \mathcal{F}(o)} \Phi_g$. The CNF Φ_f contains constraints for path sensitization. For a detailed description about Φ_F and for more information about SAT-based ATPG, it is referred to [16], [17]. If the SAT solver determines that Φ_{Test} is satisfiable, the fault is testable. The test pattern can directly be extracted from the solution provided by the SAT solver. If the SAT solver proves that Φ_{Test} is unsatisfiable, the fault is untestable.

A. Exciting Multiple Aggressors

In previous work, test generation and aggressor excitation were divided into two steps. In the first step, a test is generated sensitizing the victim path. Then taking this test as constraint,

[1]For reasons of simplicity, the descriptions of the techniques are restricted to crosstalk-induced delay. However, it is easily possible to apply them to crosstalk-induced speed-up or glitches.

[2]In preliminary studies in the field of ATPG, we also experimented with a circuit-based SAT solver but did not observe improvements compared to a CNF-based SAT solver.

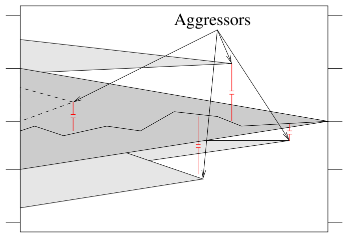


Fig. 3. Multiple Aggressors

aggressors are excited to enhance the crosstalk-induced effects of this test. However, the scenario inducing maximal crosstalk might be missed using this two-step procedure. In this section, it is described how test generation and aggressor excitation can be done in parallel using a SAT formulation.

The SAT instance Φ_{Test} has to be augmented if multiple aggressors are to be guaranteed to be excited simultaneously. Since a rising or falling edge is needed to excite an aggressor a, the test generation algorithm has to justify this value additional to the fault-specific constraints. This is schematically depicted in Figure 3. The fanin cone of each aggressor which is to be excited has to be included in Φ_{Test}. Additionally, constraints must be formulated to set the desired edges on the aggressor lines. For an aggressor a, these constraints are denoted by Φ_F^a. Therefore, the SAT instance Φ_{Test}^A is built for a given fault F and a set of aggressors $A = \{a_1, \dots, a_n\}$ as follows:

$$\Phi_{Test}^A = \Phi_{Test} \cdot \underbrace{\prod_{i=1}^{n} \Phi_{\mathcal{F}(a_i)}}_{\text{Circuit}} \cdot \underbrace{\prod_{i=1}^{n} \Phi_F^{a_i}}_{\text{Excitation}}$$

Evaluating Φ_{Test}^A yields a test which excites all aggressors $a_1, \dots, a_n$ or proves that such a test does not exist.

In contrast to the two-step approach described above, both test generation and aggressor excitation is done in parallel using only one SAT instance. Since aggressor excitation is done unconstrained, evaluating Φ_{Test}^A yields a test which excites all aggressors $a_i \in A$ or proves that no such test exists. Using a one-step approach increases complexity of test generation, i.e. the search space is larger. However, the experiments show that SAT-based ATPG techniques are able to deal with the increased complexity in reasonable time.

IV. Aggressor Selection Schemes

A SAT formulation for unconstrained aggressor excitation was presented in the last section. Typically, there are many potential aggressors coupled to a victim path. However, due to logical correlations, it is very unlikely that all of them can be excited at the same time. Therefore, an aggressor combination has to be found which is logically valid and provides maximal induced crosstalk on the victim path. Checking all possible aggressor combinations is not feasible, since for n aggressors, 2^n possible aggressor combinations exist. Two different aggressor selection schemes were adopted and evaluated in this work: an exact *Branch-and-Bound* (B&B) procedure similar to [13] and a *Static Aggressor Ordering* (SAO) heuristic as presented in [12].

A. Branch-and-Bound

The branch-and-bound algorithm used explores a binary search tree of aggressors as depicted in Figure 4. Each node in this tree corresponds to a specific aggressor combination and each branch denotes whether the corresponding aggressor is included in the current combination ("in") or not ("out").

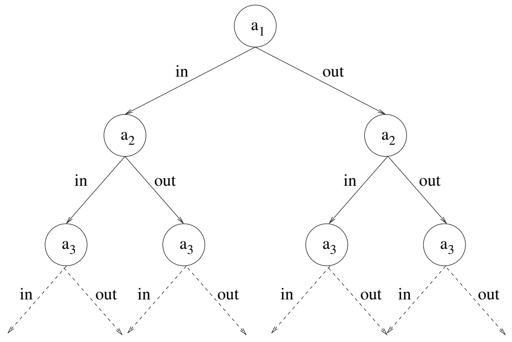


Fig. 4. Example branch-and-bound search tree

A complete path through the binary tree corresponds to an aggressor combination where each aggressor is either "in" or "out". The validity of each node has to be checked using the SAT method described in Section III-A. If the SAT solver determines that the current aggressor combination cannot be excited, the branch can be bounded. If a complete path is found, i.e. an aggressor combination, the result is saved. A branch can also be bounded if the best possible aggressor combination has less strength than an already found one. Using the B&B algorithm, it is guaranteed that the aggressor combination inducing maximal crosstalk is found. The drawback of this method is the very large number of SAT instances which have to be solved.

B. Static Aggressor Ordering

Instead of a B&B procedure, a static aggressor ordering is used in [12] to overcome the drawback of the large number of possible aggressor combinations. At first, a test which sensitizes the victim path is generated. Then, all aggressors are ordered according to their strength. Each aggressor a_i is excited consecutively taking the generated test into account ("constrained test generation"). If a test is found which excites a_i, the test is updated. This scheme was adopted for constrained as well as for unconstrained aggressor excitation.

In contrast to the B&B algorithm, this procedure does not explore the complete search tree. When one aggressor a_k is found to be able to be excited, this aggressor cannot be removed anymore. Although if one possible combination of aggressors exists which does not excite a_k, but has a cumulative strength greater than any combination including a_k, this combination will not be found.

V. Experimental Results

The experimental results are presented in this section. The proposed approach was implemented in C++. All experiments were conducted on an AMD Athlon64 X2 6000+ (4,096MB, 3GHz, GNU/Linux). The approach was tested on publicly available benchmark circuits from the ISCAS'85, ISCAS'89 and ITC'99 benchmark suite. The underlying SAT-based ATPG engine was DynamicSAT [18]. All experiments were performed using the launch-on-capture scheme. The longest 1,000 non-robustly testable paths were extracted and chosen as victim paths. The aggressors and their strength were selected randomly as done in [8], [12]. Table I shows the experimental results of the test generation procedure. In order to show the efficiency of the approach, column *Single* presents the results of the unconstrained test generation with a single aggressor line. Here, each victim path is coupled with only one aggressor line. Column *Agg* shows how many of these paths are still testable with this aggressor excited. Column *Time* presents the

TABLE I
EXPERIMENTAL RESULTS

	Single		Constrained						Unconstrained					
			Static ordering			B&B			Static ordering			B&B		
Circ	Agg	Time	#Agg	Str.	Time	#Agg	Str.	Time	#Agg	Str.	Time	#Agg	Str.	Time
c1908	72%	0.6	12	1.0x	12.9	21	1.65x	178.4	41	3.27x	30.3	41	3.28x	309.6
c3540	63%	0.6	11	1.0x	33.4	16	1.48x	561.1	34	3.13x	73.1	35	3.23x	857.3
c5315	93%	0.2	68	1.0x	73.1	68	1.00x	312.5	69	1.02x	76.0	69	1.03x	318.6
c7552	92%	0.3	71	1.0x	133.4	71	1.01x	497.6	72	1.01x	134.8	71	1.01x	502.2
s13207	76%	1.2	4	1.0x	39.0	4	1.21x	1,309.2	69	15.39x	215.8	68	15.39x	935.4
s15850	84%	1.6	12	1.0x	59.5	62	4.76x	658.9	70	5.44x	212.1	69	5.44x	942.5
s35932	91%	0.3	84	1.0x	217.1	83	1.00x	443.7	85	1.01x	220.9	84	1.01x	442.0
s38417	98%	1.5	77	1.0x	458.5	82	1.06x	1,177.4	84	1.09x	514.5	83	1.09x	1,187.0
s38584	92%	2.6	64	1.0x	278.5	65	1.03x	1,405.7	75	1.17x	342.6	74	1.18x	1,354.5
b15	69%	10.7	4	1.0x	137.8	37	8.01x	3,990.5	51	11.39x	834.2	51	11.39x	5,918.8
b17	64%	8.5	66	1.0x	5,537.7	66	1.01x	28,126.5	66	1.02x	5,526,0	69	1.05x	27,645.2
b20	87%	14.4	34	1.0x	1,882,0	37	1.09x	8,814.6	78	2.30x	5,070.8	78	2.30x	14,946.4
b21	85%	15.6	4	1.0x	249.4	12	2.54x	4,389.8	75	15.61x	5,059.1	74	15.64x	17,259.6
b22	90%	13.3	13	1.0x	817.2	31	2.33x	6,047.4	83	6.31x	5,463.6	82	6.32x	12,884.0
av.	83%	–	37	1.0x	1.0x	47	2.08x	11.2x	68	4.94x	4.0x	68	4.95x	17.7x

run time in CPU seconds. Even for large benchmark circuits, our approach needs only a few seconds.

Column *Constrained* presents the results for the constrained test generation. Here, 100 potential aggressors were selected and the task was to find the aggressor combination inducing maximal crosstalk. The results for the SAO heuristic are given in column *Static ordering*, whereas the results for the B&B procedure are presented in column *B&B*. Column *#Agg* reports the average number of aggressors that could be excited for one victim path. The average strength of the aggressor combination inducing maximal crosstalk is given in column *Str.*. The strength is given in relation to the results of the constrained test generation with SAO to highlight the improvement of the proposed approach. First, the results of the constrained test generation are discussed. As expected, tests found by the SAO heuristic have less aggressor strength than tests found by the B&B algorithm. The average factor of aggressor strength increase is 2. However, the drawback of the branch-and-bound approach is the long run time which is increased on average by a factor of 11.

The results for the proposed unconstrained test generation are presented in column *Unconstrained*. The unconstrained approach is able to excite significant more aggressors than the constrained approach. The average aggressor strength is increased by a factor of 4.94 and 4.95, respectively. The aggressor strength of the SAO heuristic is even greater than the strength of the constrained B&B approach. The difference in aggressor strength between unconstrained SAO heuristic and unconstrained B&B algorithm is very small. However, the run time increase of the B&B algorithm is about a factor of 17, whereas the run time of the SAO heuristic increases only by a factor of 4. Therefore, unconstrained test generation using the SAO heuristic yields the best trade-off between increasing crosstalk-induced noise and run time of all evaluated configurations.

VI. Conclusion

Previous approaches for test generation with induced crosstalk noise suffer from the fact that they increase the crosstalk noise of a previously generated test. This "constrained" procedure restricts the search space and typically avoids that the test with maximal induced crosstalk noise is found. In this paper, we presented an "unconstrained" procedure which generates a test and excites multiple aggressors in one step. By this, the complete search space is traversed and a solution is guaranteed to be found, if existent. Efficient SAT-based ATPG techniques are used to generate tests exciting multiple aggressors. Two different schemes for finding the aggressor combination with maximal induced crosstalk are evaluated: a static aggressor ordering heuristic and an exact branch-and-bound algorithm. The unconstrained test generation is shown to induce significantly more crosstalk noise than the constrained approach with acceptable run time overhead. Future work is the integration of static timing analysis techniques.

References

[1] W. Chen, S. K. Gupta, and M. A. Breuer, "Analytic models for crosstalk delay and pulse analysis under non-ideal inputs," in *Int'l Test Conf.*, 1997, pp. 809–818.

[2] A. Vittal and M. Marek-Sadowska, "Crosstalk reduction for VLSI," *IEEE Trans. on Computer-Aided Design for Circuits and Systems*, vol. 16, no. 3, pp. 290–298, 1997.

[3] M. A. Breuer and S. K. Gupta, "Process aggravated noise (PAN): New validation and test problems," in *Int'l Test Conf.*, 1996, pp. 914–923.

[4] W. Chen, S. K. Gupta, and M. A. Breuer, "Test generation in VLSI circuits for crosstalk noise," in *Int'l Test Conf.*, 1998, pp. 641–650.

[5] W.-Y. Chen, S. K. Gupta, and M. A. Breuer, "Test generation for crosstalk-induced delay in integrated circuits," in *Int'l Test Conf.*, 1999, pp. 191–200.

[6] S. Kundu, S. T. Zachariah, Y.-S. Chang, and C. Tirumurti, "On modeling crosstalk faults," *IEEE Trans. on Computer-Aided Design for Circuits and Systems*, vol. 24, no. 12, pp. 1909–1915, 2005.

[7] X. Bai, S. Dey, and A. Krstić, "HyAC: A hybrid structural SAT based ATPG for crosstalk," in *Int'l Test Conf.*, 2003, pp. 112–121.

[8] K. P. Ganeshpure and S. Kundu, "Automatic test pattern generation for maximal circuit noise in multiple aggressor crosstalk faults," in *Design, Automation and Test in Europe*, 2007, pp. 540–545.

[9] ——, "On ATPG for multiple aggressor crosstalk faults in presence of gate delays," in *Int'l Test Conf.*, 2007, pp. 1–7.

[10] M. Zhang and X. Li, "Test generation for crosstalk glitches considering multiple coupling effects," in *IEEE Asian Test Symp.*, 2007, pp. 259–264.

[11] J. Lee and M. Tehranipoor, "A novel pattern generation framework for inducing maximum crosstalk effects on delay-sensitive paths," in *Int'l Test Conf.*, 2008, pp. 1–10.

[12] S. Chun, T. Kim, and S. Kang, "ATPG-XP: test generation for maximal crosstalk-induced faults," *IEEE Trans. on Computer-Aided Design for Circuits and Systems*, vol. 28, no. 9, pp. 1401–1413, 2009.

[13] M. Palla, J. Bargfrede, K. Koch, W. Anheier, and R. Drechsler, "Adaptive branch and bound using SAT to estimate false crosstalk," in *Int'l Symp. on Quality Electronic Design*, 2008, pp. 508–513.

[14] M. Palla, J. Bargfrede, S. Eggersglüß, W. Anheier, and R. Drechsler, "Timing arc based logic analysis for false noise reduction," in *Int'l Conf. on Computer-Aided Design*, 2009, pp. 225–230.

[15] A. Biere, M. Heule, H. v. Maaren, and T. Walsh, *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009.

[16] 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 for Circuits and Systems*, vol. 27, no. 7, pp. 1329–1333, 2008.

[17] R. Drechsler, S. Eggersglüß, G. Fey, and D. Tille, *Test Pattern Generation using Boolean Proof Engines*. Springer, 2009.

[18] S. Eggersglüß, D. Tille, and R. Drechsler, "Speeding up SAT-based ATPG using dynamic clause activation," in *IEEE Asian Test Symp.*, 2009, pp. 177–182.