Improved Circuit-to-CNF Transformation for SAT-based ATPG

Daniel Tille[1]* René Krenz-Bååth[2] Juergen Schloeffel[2] Rolf Drechsler[1]

[1] Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
{tille,drechsle}@informatik.uni-bremen.de

[2] NXP Semiconductors Germany GmbH, 21147 Hamburg, Germany
{rene.krenz-baath,juergen.schloeffel}@nxp.com

Abstract—**SAT-based ATPG has proven to be a beneficial complement to traditional ATPG techniques. The generation of a CNF-representation is a vital issue in SAT-based test pattern generation. Firstly, the generation of the problem instances for SAT-based ATPG requires a significant portion of the overall runtime. Secondly, the performance of the SAT solver strongly depends on the properties of the resulting CNF-representation.**

The contribution of this paper is a new approach to generate CNF-representations for SAT-based ATPG. The objective of the proposed technique is to speed up the generation process and to optimize the resulting CNF-representation with respect to the SAT computation. The experimental results, obtained on large industrial designs, show that the accomplished optimizations result in a significant reduction of the overall runtime of the SAT-based test pattern generation process. Finally we discuss how this contribution enables some promising future work.

I. Introduction

The continuous growth of today's circuit designs requires a constant improvement of state-of-the-art computer-aided design (CAD) and computer-aided test (CAT) tools. During recent years SAT-based ATPG algorithms became a promising alternative to traditional ATPG techniques such as FAN [1] and PODEM [2]. In particular for hard-to-solve problem instances SAT-based methods proved to be highly advantageous [3], [4], [5], [6]. However, SAT-based ATPG algorithms suffer from a number of disadvantages. Most modern SAT-solvers [7], [8], [9], [10] require the modeling of the problem in *Conjunctive Normal Form* (CNF). Hence the ATPG problem needs to be converted into one or several CNF-representations. Furthermore SAT-based ATPG techniques tend to deliver overdetermined test pattern which is disadvantageous with respect to pattern compaction and runtime.

The contribution of this paper is a new approach to efficiently generate CNF-representations for SAT-based ATPG. The proposed technique employs *Reduced Ordered Binary Decision Diagrams* (ROBDDs) [11] in order to generate optimized CNF-instances during test pattern generation. The new approach constructs ROBDD-representations of *Fanout-Free Regions* (FFRs) in the circuit graph, where every FFR is treated as an individual sub-circuit. To derive CNF-representations directly from these ROBDDs provides several advantages over traditional approaches. Functional redundancies contained in the FFR will not be reflected by the resulting CNF. Furthermore the number of CNF-variables is reduced drastically. Finally the number of CNF-clauses in the problem instances is considerably decreased. The experimental results show that the proposed approach is a promising alternative to traditional circuit-to-CNF transformation techniques.

The paper is structured as follows. Previous work is discussed in Section II. The proposed approach is the objective of Section III. Experimental results are presented in Section IV. Section V concludes the paper and discusses future work.

*This work was done during a research visit at the NXP development center in Hamburg, Germany.

II. Previous Work

A. SAT-based ATPG

Test pattern generation with respect to some *Stuck-At Fault* (SAF) is the search for an input assignment, which conducts different values at some primary output between the faulty circuit and the correct circuit. SAT-based ATPG was initially proposed by Larrabee in [3]. Significant improvements were achieved by Stephan et al. in [12] and by Silva and Sakallah in [13]. In SAT-based ATPG the problem of finding a sufficient input assignment is transformed into a Boolean satisfiability problem such that if a test for a particular SAF exists, then the corresponding problem instance is satisfiable and the resulting test pattern can be directly derived from the satisfying assignment. If the fault is undetectable the SAT solver concludes unsatisfiability.

In the following the circuit-to-CNF transformation for an SAF is reviewed. Figure 1 illustrates a combinational circuit. The *fault location* denotes a connection c where an SAF is assumed. The area denoted as *output cone*, contains all gates belonging to some path P from c to some primary output in the reflexive fanout of c, $fanout^*(\{c\})$. Let us denote the set of primary outputs reachable from c by O_c. Next the reflexive

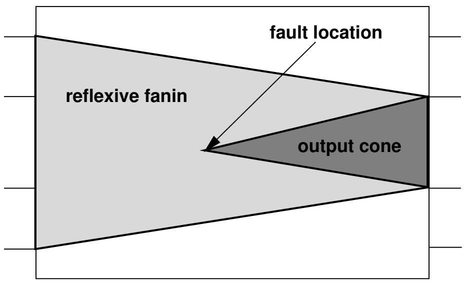


Fig. 1. Illustration of influenced circuit areas.

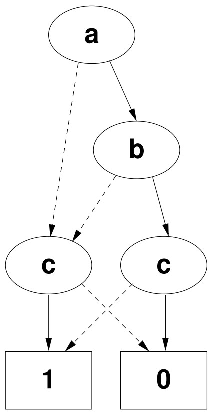
fanin, $fanin^*(O_c)$, of all primary outputs in O_c is computed, see Figure 1.

As introduced in [4], two Boolean variables $G(v)$ and $G_f(v)$ are assigned to every gate v in the reflexive fanout of c, $v \in fanout^*(\{c\})$, to represent the fault free circuit and the faulty circuit, respectively. Both circuits are generated by building the characteristic function for every gate. Clearly all gates belonging to the reflexive fanin of some primary output in the set O_c and are not contained in $fanout^*(\{c\})$ only need to be modeled once, since the SAF at c does not influence their behaviour. To express a difference of the values at $G(v)$ and $G_f(v)$, additionally a Boolean variable $G_d(v)$ is assigned to every gate $v \in fanout^*(\{c\})$. If $G_d(v)$ is true, then the values $G(v)$ and $G_f(v)$ differ. A test pattern to detect the SAF at c is found if it is possible to compute an assignment such that there exists a path P from c to some primary output, where the variable $G_d(v)$ for each gate $v \in P$ is true. This path is called *D-chain*.

B. ROBDDs

The proposed technique employs ROBDDs as a *canonical* representation of Boolean functions [11]. Canonicity allows significant performance improvements for operations such as equivalence checking or satisfiability checking. Additionally ROBDDs are a highly effective representation for large combinational sets, which is crucial for model checking tasks [14].

Given an ROBDD representing some Boolean function a CNF of this function can be generated as follows: Let P be a path from the root node to the zero-terminal node. A clause can be derived by the disjunction of the complement of each variable occurring in P. In Figure 2 an example for this ROBDD-to-CNF conversion is given. An ROBDD consisting of three variables is depicted in Figure 2(a). The solid lines represent the high-edges and the dashed lines represent the low-edges. The CNF describing the same Boolean function as the ROBDD is given in Figure 2(b). To generate the CNF all paths from the root node (labeled with a) to the zero-terminal node (labeled with 0) have to be traversed. Since there exist three such paths in the ROBDD the CNF consists of three clauses. The first clause is derived by applying the approach explained above to the path consisting of all high-edges. The second clause (along $a_{high} - b_{low} - c_{low}$) and the third clause (along $a_{low} - c_{low}$) are built the same way.



$(\bar{a} + \bar{b} + \bar{c})$
$(\bar{a} + b + c)$
$(a + c)$

(a) (b)

Fig. 2. Example for the ROBDD-to-CNF conversion: (a) ROBDD, (b) CNF.

As mentioned above the number of clauses in the CNF is equal to the number of zero-paths in the ROBDD. Unfortunately in the worst case the number of zero-paths is exponential with respect to the number of nodes (e.g., the EXOR function). A sophisticated approach which introduces auxiliary variables in order to reduce the number of paths in the ROBDD is proposed in [15]. The minimization of the number of paths in an ROBDD using different sifting strategies is described in [16].

III. Improved Circuit-to-CNF Transformation

The traditional approach to derive a CNF-representation from a circuit graph is described in Section II. The CNF-representation of every single gate is generated without consideration of the adjacent circuit structure. It is obvious that this concept, although simple, does not generate a compact CNF-representation of the problem instance.

The goal of the proposed technique is to increase the efficiency of the CNF-generation process itself and to improve the properties of the generated problem instance with respect to the succeeding reasoning process. This is accomplished by first generating an ROBDD-based representation of specific parts of the circuit graph, and later on deriving the corresponding CNFs from these ROBDDs. The circuit is decomposed along *Fanout-Free Regions* (FFRs) in the circuit graph, where each FFR is treated as an individual sub-circuit. The D-chains (cf. Subsection II-A) are build along FFRs, i.e. variables G_d are only assigned to FFR-output gates.

A. Observations

The proposed approach is based on the following observations:

1) Kuehlmann and Krohm showed in [17] that industrial circuits contain a large amount of functional redundancies. A CNF derived from the corresponding ROBDD

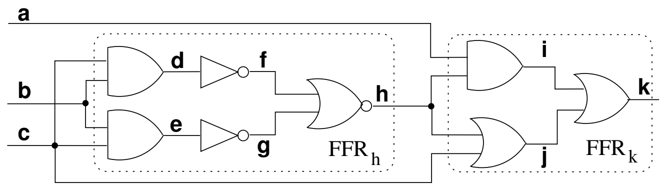


Fig. 3. Illustrative example.

will not reflect these redundancies. This yields a reduction in the number of clauses and the amount of CNF-variables in the resulting problem instance.

2) The presented technique works similar to the concept of supergates, where a set of gates in a circuit graph is merged to a larger gate [18]. Thus the number of CNF-variables required to generate the CNF of this set of gates is significantly smaller than the amount of CNF-variables produced by the traditional approach.
3) We observed that numerous FFRs or their subfunctions are functionally equivalent. Treating FFRs as independent sub-circuits leads to an increased sharing of ROBDD-structures for the representation of several functionally equivalent circuit-structures. Note that we refer to the functional equivalence of gates with respect to the inputs of individual FFRs.

B. Illustrative Example

The following example clarifies the observations discussed in Subsection III-A. The circuit graph depicted in Figure 3 contains two FFRs, named *FFR*$_h$ and *FFR*$_k$ corresponding to the names of their output gates. The set of gates belonging to the individual FFRs is indicated by dotted lines.

Considering *FFR*$_h$ it is easy to see that gates d, e, and h implement the same Boolean function, $b \wedge c$, with respect to the FFR-inputs b and c. Transforming these gates in the traditional way would require 7 CNF-variables and 13 clauses. The proposed techniques would instead generate only 3 CNF-variables and 3 clauses. These improvements are explicitly obtained through redundancy removal as described in the first observation in Subsection III-A.

Figure 4 presents two alternative CNF-representations of the Boolean function implemented by *FFR*$_h$ in Figure 3. Part (a) of Figure 4 shows the CNF-representation obtained using the traditional approach, where a CNF is generated for every single gate in the circuit graph. This technique introduces a CNF-variable for every single gate contained in the FFR and ignores possible reductions with respect to adjacent gates. Figure 4(b) contains the result of the ROBDD-to-CNF transformation, where the CNF-representation is derived from the BDD-based representation of *FFR*$_h$ as described in Subsection II-B.

Generating the CNF-representation of the Boolean function implemented by *FFR*$_k$ in the traditional way would require 6 CNF-variables. As described in the second observation, it is possible to reduce the number of required CNF-variables to 4

$$\boxed{\begin{array}{lll} (d+\overline{b}+\overline{c}) & \cdot\,(\overline{d}+b) & \cdot\,(\overline{d}+c)\,\cdot \\ (e+\overline{b}+\overline{c}) & \cdot\,(\overline{e}+b) & \cdot\,(\overline{e}+c)\,\cdot \\ (h+f+g) & \cdot\,(\overline{h}+\overline{f}) & \cdot\,(\overline{h}+\overline{g})\,\cdot \\ (f+d) & \cdot\,(\overline{f}+\overline{d})\cdot & \\ (g+e) & \cdot\,(\overline{g}+\overline{e}\,) & \end{array}}$$

(a)

$$\boxed{(h+\overline{b}+\overline{c})\cdot(\overline{h}+b)\cdot(\overline{h}+c)}$$

(b)

Fig. 4. CNFs of the example: (a) Traditionally derived CNF, (b) ROBDD-derived CNF.

by treating the FFR as a single but more complex gate.

As mentioned above every FFR is treated as an individual Boolean function. Variables used in the corresponding ROBDD-representations are re-used for every individual FFR. Let us assume that the ROBDD-variables x_1, x_2, and x_3 are assigned to the FFR-inputs a, h, and c, respectively, of *FFR*$_k$. Furthermore we assume the order of the ROBDD-variables to be x_1, x_2, x_3. Then the generated ROBDD-structure can be partially re-used to also represent the function implemented by *FFR*$_h$, assuming that the FFR-inputs b and c are assigned to x_1 and x_2, respectively. In practice this feature contributes to a dramatic reduction of the number of generated ROBDD-nodes and hence a significant improvement with respect to the memory consumption.

C. Implementation Details

In the following we discuss issues regarding the implementation of the proposed technique.

The construction of the ROBDD-based representations is accomplished during a preprocessing step. This means the circuit-to-ROBDD conversion is performed only once. As BDD package we choose to apply the well-known CUDD package [19] version 2.4.1. CNF-representations of individual FFRs are generated multiple times during a complete ATPG run. In the current implementation the initially computed ROBDDs will be repeatedly used during the entire ATPG run. The derivation of the CNF from the corresponding ROBDD-representation is discussed in Section II. The number of literals contained in the generated CNF-clauses is significantly reduced by using the prime implicants in the ROBDDs. The mapping of ROBDD-variables onto the corresponding CNF-variables is accomplished during every individual CNF-generation.

During first experiments it was observed that the number of prime implicants and hence the number of clauses derived from an ROBDD often exceed the number of clauses generated by the traditional approach. We observed that this situation appears frequently for ROBDD-representations of FFRs with more than 16 inputs. Therefore it was decided to add two limitations in order to prevent such an increase of the CNF size. Firstly a circuit-to-ROBDD transformation is only performed for FFRs with 16 or less inputs. Secondly the number of clauses generated by the traditional approach is estimated. FFRs whose ROBDD-representation exceeds this limit are not considered during the succeeding ROBDD-to-CNF transformation. The estimation is based on the following two assumptions:

- the FFR does only contain two-input gates, and
- the CNF-representation of every individual gate in the FFR would require three clauses, e.g., AND, NAND, OR, NOR.

Based on these assumptions the following cost function can be formulated:

$$n_{clauses} = 3(n_{inputs} - 1),$$

where $n_{clauses}$ denotes the estimated number of clauses, and n_{inputs} represents the number of FFR-inputs. For example the estimated amount of resulting clauses for a 4-input FFR and for an 11-input FFR would be 9 clauses and 30 clauses, respectively.

Furthermore, the current implementation does only construct ROBDD-representations of FFRs which do not require an encoding for four-valued logic.

IV. Experimental Results

This section contains an experimental evaluation of the proposed technique. We will demonstrate that the new approach reduces the number of variables and the number of clauses in the problem instance significantly. The obtained runtime improvements confirm that the proposed optimizations result in a considerable speed-up of the actual pattern generation.

The new technique was integrated into a prototype version of the NXP Semiconductors ATPG tool *AMSAL* and applied to a set of benchmarks consisting of four large industrial designs. The used SAT-solver is MiniSat version 1.14 [10]. The experiments were performed on a PC equipped with a 2.4 GHz AMD Opteron 880 CPU and 64 GByte main memory running RedHat Enterprise 4.

Table I provides a first set of experimental results. Columns one to four contain information about the circuits, such as benchmark name, number of inputs, number of outputs, and number of targets, respectively. The set of targets contains all remaining faults after fault collapsing. The name of the benchmark reflects the approximate number of gates contained in the circuit. For example benchmark p141k contains roughly 141,000 gates.

Building all ROBDDs for the tested designs as proposed in Section III requires less than two seconds of CPU time and the additional memory consumption does not exceed 50 MByte. Further details about the number of FFRs transformed into ROBDDs and the CNF-size savings are given in Table II.

TABLE I
Benchmark statistics and results obtained by the presented algorithm in comparison to the traditional approach using a set of industrial benchmarks.

Benchmarks	Inputs	Output	Targets	runtime without ROBDDs	runtime with ROBDDs
p141k	11,290	10,502	267,946	5:23h	3:44h
p267k	17,332	16,621	366,773	16:27m	10:53m
p330k	18,010	17,468	540,756	1:48h	1:38h
p418k	30,430	29,809	674,022	3:42h	3:33h

TABLE II
Further results obtained by the presented algorithm in comparison to the traditional approach.

Benchmarks	% treated FFRs	% saved clauses	% saved CNF-vars
p141k	10.2	35.72	38.18
p267k	14.9	31.49	33.82
p330k	8.7	29.16	33.39
p418k	10.2	29.01	33.48

Although only a subset of FFRs in the circuit graph has been considered, see second column in Table II, runtime reductions of 30.65% and 33.84% for the designs p141k and p267k, respectively, have been obtained. However, the overall runtimes, listed in the most right column of Table I, do not reflect the achieved improvements with respect to the actual runtime required to find a sufficient input assignment. In order to provide this information a set of diagrams is depicted which contains individual SAT-runtimes for every computed target. Figure 5 comprises four diagrams, which directly compare the runtimes required by the SAT-solver with and without the proposed preprocessing technique. Each entry represents the computation times consumed for a single target. The runtime is considerably reduced for a large number of targets.

The second set of diagrams, given in Figure 6, illustrates the achieved improvements with respect to the number of clauses to model the Boolean function implemented by an individual FFR. Note that these results only reflect the clause reduction of FFRs treated by the proposed technique. The number of clauses is considerably reduced in nearly all cases. In particular for larger FFRs the number of clauses could be reduced significantly. One reason for that is the higher likelihood of redundancies contained in these FFRs. The reduction of the number of clauses and the number of CNF-variables due to redundancy removal is discussed in Section III. Columns three and four in Table II report the percental reduction of the number of clauses and the number of CNF-variables with respect to all FFRs treated by the proposed technique.

V. Conclusions and Future Work

The contribution of this paper is a new approach to efficiently translate a circuit-based ATPG problem into a corresponding CNF-representation. In contrast to traditional approaches, which construct the CNF-representation of every single gate without considering the adjacent circuit structure, the new technique generates the CNF-representation of a set

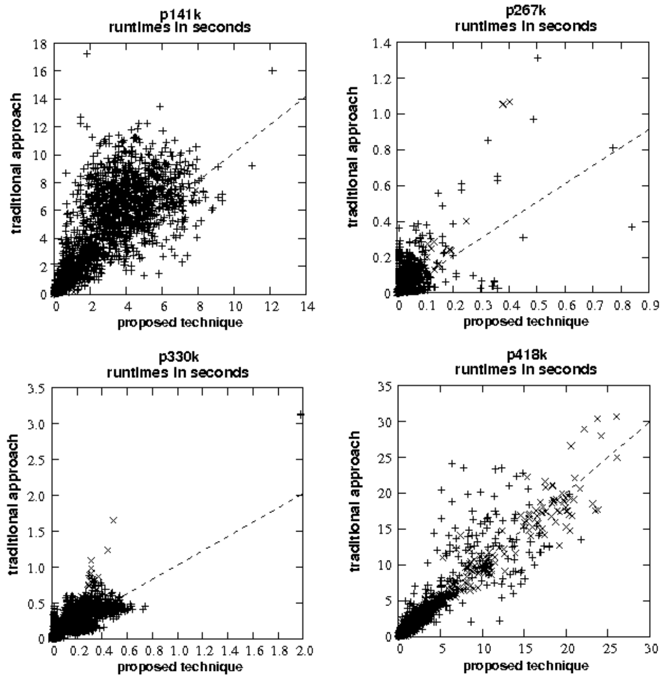


Fig. 5. Runtime comparison for individual targets.

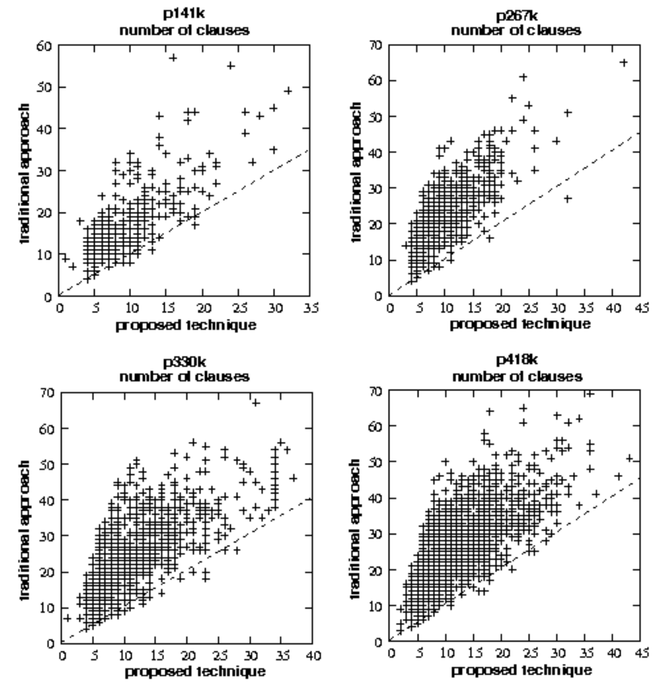


Fig. 6. Number of clauses contained in the CNF-representation of individual FFRs.

of gates. The proposed technique employs ROBDDs in order to reduce the resulting problem instance with respect to the number of clauses and the amount of CNF-variables. The experimental results confirm that the overall runtime of the ATPG computation can be significantly reduced using the new technique.

Our future work involves the application of the proposed concept to other fault models, e.g. the gate delay fault model and the path delay fault model. Additionally the technique can be extended to handle multiple-valued logic which is contained in many industrial circuits.

ACKNOWLEDGEMENTS

Parts of this research work were supported by the German Federal Ministry of Education and Research (BMBF) in the Project MAYA under the contract number 01M3172B.

REFERENCES

[1] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms", *IEEE Trans. on Comp.*, vol. 32, pp. 1137–1144, 1983.
[2] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic", *IEEE Trans. on Comp.*, vol. 30, pp. 215–222, 1981.
[3] T. Larrabee, "Test pattern generation using Boolean satisfiability", *IEEE Trans. on CAD*, vol. 11, pp. 4–15, 1992.
[4] P. Stephan, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability", *IEEE Trans. on CAD*, vol. 15, pp. 1167–1176, 1996.
[5] 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*, 2005, pp. 212–217.
[6] D. Tille, S. Eggersglüß, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel, "Studies on integrating SAT-based ATPG in an industrial environment", in *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, 2007.
[7] J.P. Marques-Silva and K.A. Sakallah, "GRASP – a new search algorithm for satisfiability", in *Int'l Conf. on CAD*, 1996, pp. 220–227.
[8] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver", in *Design Automation Conf.*, 2001, pp. 530–535.
[9] E. Goldberg and Y. Novikov, "BerkMin: a fast and robust SAT-solver", in *Design, Automation and Test in Europe*, 2002, pp. 142–149.
[10] N. Eén and N. Sörensson, "An extensible SAT solver", in *SAT 2003*, 2004, vol. 2919 of *LNCS*, pp. 502–518.
[11] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation", *IEEE Trans. on Comp.*, vol. 35, no. 8, pp. 677–691, 1986.
[12] P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability", Tech. Rep. UCB/ERL M92/112, Dept. of EECS, Univ. of California, Berkeley, October 1992.
[13] J.P. Marques-Silva and K.A. Sakallah, "Robust search algorithms for test pattern generation", Tech. Rep. RT/02/97, Dept. of Informatics, Technical University of Lisbon, Lisbon, Protugal, January 1997.
[14] K.L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publisher, 1993.
[15] G. Cabodi, S. Nocco, and S. Quer, "SAT-based bounded model checking by means of BDD-based approximate traversals", in *Design, Automation and Test in Europe*, 2003, pp. 898–903.
[16] G. Fey and R. Drechsler, "Minimizing the number of paths in BDDs: Theory and algorithm", in *IEEE Trans. on CAD*, 2006, pp. 4–11.
[17] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps", in *Design Automation Conf.*, 1997, pp. 263–268.
[18] S.C. Seth, L. Pan, and V.D. Agrawal, "Predict - probabilistic estimation of digital circuit testability", in *Int'l Symp. on Fault-Tolerant Comp.*, 1985, pp. 220–225.
[19] F. Somenzi, *CUDD: CU Decision Diagram Package Release 2.3.1*, University of Colorado at Boulder, 2001.