IMPLICIT REPRESENTATIONS IN CUSTOMIZED TESTING OF DIGITAL CIRCUITS

Jiří Balcárek
Informatics and Computer Science, 1-st class, full-time study
Supervisor: Ing. Jan Schmidt, Ph.D., Ing. Petr Fišer, Ph.D.
Czech Technical University in Prague, FIT, Dept. of Digital Design
Kolejní 550/2, Praha 6,Česká republika
balcaji2@fel.cvut.cz

Abstract. Despite of the growing number of analog and mixed signal parts in electronic devices, there are still many challenges in digital circuits testing. Many test patterns generation and compression methods for digital circuits have been developed, but their efficiency can dramatically depend on the representation of test vectors sets. Also the importance of additional requirements imposed upon a test set increases. In this paper we consider implicit representations of test patterns sets and possibilities of their use in test patterns generation, compression and constrained test generation. We suppose that by using a proper implicit representation and a set of possible constraints, a significant improvement in customized test generation is possible.

Keywords. Implicit test representation, testing, test compression.

1 Introduction

Digital circuits testing had been a great challenge in the past, but at present, there are many methods for Automatic Test Pattern Generation (ATPG) [1] and compression and also a great number of test architectures. It can seem that testing of digital circuits is a completely mastered part of the design flow and the future challenges are mixed signal and analog testing. Demands of industry are clear, but there are still some parts in digital testing waiting to be overmastered. There is a need to generate customized test sets, e.g., to decrease heat dissipation and power consumption during the test. Also more fault models are needed to cover greater number of defects.

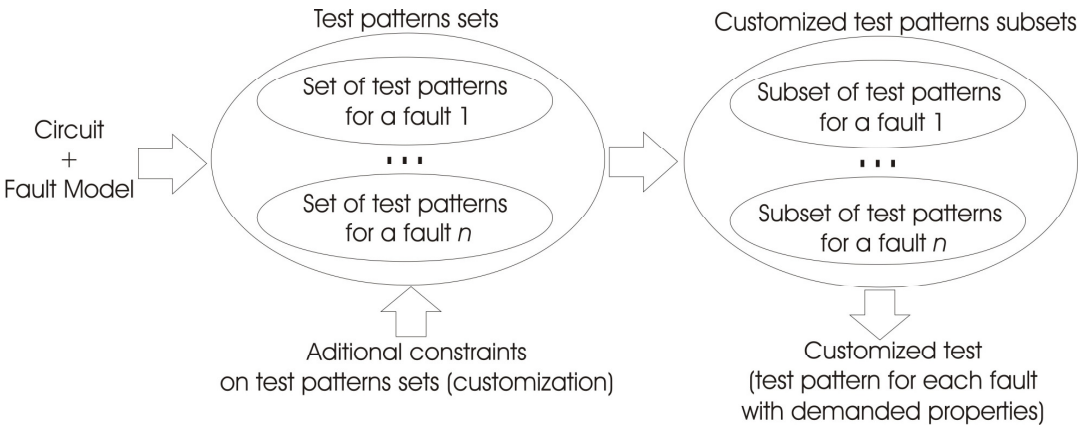

Figure 1: Basic concept of customized test generation

Test patterns set can be compactly and with lower memory requirements represented implicitly. A dedicated ATPG can benefit such implicit test representations for customized test generation. The best test pattern or a subset of best test patterns for a given application can be chosen by additional constraints, see Figure 1. Our aim is to find a general concept of a proper implicit test representation, constraints, and possible solution preserving operations independent of the fault model and customized by the target application. As far as we know, there have been no similar concepts of test customization based on implicit representations.

The SAT-Compress [2] algorithm doing the test patterns compression over an implicit test representation in CNF (Conjunctive Normal Form) has been introduced in our previous work [2]. Basic implicit representations are discussed in Section 2. Interesting observations about CNF based on our previous work are presented and discussed in Section 3. The assumed future work is described in Section 4 and a brief conclusion of this paper is in Section 5.

2 Basic Implicit Representations

This section gives a brief overview of two best known implicit representations of test patterns set: the Conjunctive Normal Form (CNF) [1, 3] and Reduced-Ordered Binary Decision Diagram (ROBDD) [4, 5, 6], and discusses their properties. Figure 2 shows CNF and ROBDD of the same simple function.

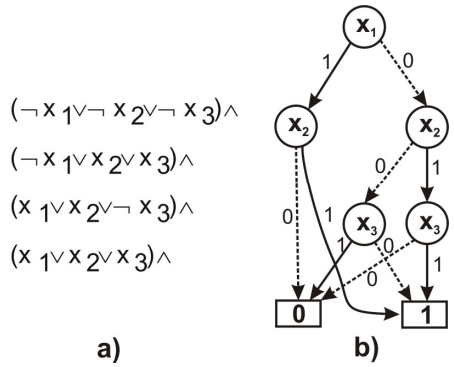

Figure 2: Basic implicit representations (a) CNF, (b) ROBDD.

Boolean function written in the CNF (see Figure 2a) is a conjunction of clauses, each clause is a disjunction of literals. A literal is a logic variable or its complement. The set of possible solutions (test patterns) is defined as an enumeration of satisfiable assignments of the variables. The set of solutions can be easily customized by a forced variables assignment or by additional clauses. A CNF can be easily obtained [3] from a Boolean network, its size grows linearly with circuit size.

Reduced-Ordered Binary Decision Diagrams [4, 5, 6] are another efficient representation of a Boolean function as a rooted Directed Acyclic Graph (DAG) (see Figure 2b). It consists of decision nodes (variables) and two terminal nodes (0-terminal and 1-terminal). Nodes are connected by directed edges. Each edge is evaluated with logical value which represents its assignment to the node (variable). Each path from the root node to the 1-terminal represents a variable assignment for which the represented Boolean function is satisfiable, thus the solution set is given by the number of paths from root node to 1-terminal. The solution set can be also easily constrained by fixing some variables to a given logical value. The number of BDD nodes strongly depends on the variable ordering and can easily grow exponentially during building or manipulating of the ROBDD. That is why this implicit representation can be unfeasible for larger circuits [5].

It can be assumed that the implicit representation in CNF might be the best one for our further investigation. It is a simple and scalable implicit test set representation, which can be easily constrained. A satisfiable solution (test pattern) can be obtained by a SAT-solving, which can be a bottle-neck of this representation, however the stat-of-the-art SAT-solvers are very efficient [1, 7, 8] and the ATPG instances are easy to be solved [9, 10].

3 Properties of an Implicit Test Representation in CNF

It has been assumed in Section 2, that CNF can be an efficient representation of test patterns set, thus it is the starting point of our investigation. The SAT-Compress algorithm [2] has been implemented to prove possibilities of the proposed approach based on an implicit representation of the test patterns set. This algorithm compresses the test patterns by their overlapping [11, 12]. The stuck-at fault model and combinational circuits are assumed. Test patterns to be compressed are not pre-generated by an ATPG, but generated on the fly. In each algorithm step, the locally best test pattern is chosen from a subset of possible test patterns represented by constrained Boolean formula in CNF. The CNFs for each fault are not stored in the memory because its size can be insufficient. Each test pattern is obtained as a SAT solution of a CNF constrained by the previous test pattern which is shifted one bit left. The Most Significant Bit (MSB) is a valid bit of the compressed bitstream and the Least Significant Bit (LSB) is set as a Don't Care (DC). The SAT-Compress algorithm benefits from an implicit representation of test patterns set in CNF. The compression ratio does not depend on pre-generated test patterns and it is comparable with state-of-the-art tools. However, our measurements show that the compression efficiency depends on the structure of the circuit. Test generation time can also grow significantly in comparison with the COMPAS [12] compression algorithm which, in contrast to SAT-Compress, is based on overlapping of *pre-generated* test patterns.

Let us focus on some properties of the SAT-Compress algorithm and produced CNF instances to find possibilities of the compression efficiency improvement (time consumption/compression ratio).

3.1 Possible Speedup of Test Generation Process

The SAT-Compress algorithm is much slower in comparison with COMPAS [12]. The CNF instances are not stored in the memory (because its size can be insufficient), but they are generated repeatedly in the test compression process, which can be time consuming. For example 1,279,654 CNFs is needed to be generated and solved for the c7552 ISCAS benchmark [13], but only 456 CNFs of them fulfill the constraints given by previous variables assignments. According to our extensive measurements, in average 98% of generated CNFs are unsatisfiable with given constraints. These CNFs do not contribute to the compressed test bitstream by any test pattern, but the time to generate and solve such not suitable (unsatisfiable) instances can cause a significant time overhead. Therefore, it might be helpful to store maximally simplified CNFs in memory instead of repeated generation. However, CNF reductions can cause a loss of possible solutions (test patterns). Reduced set of test patterns can cause a decrease of the compression quality. That is why the whole set of the SAT solutions must be preserved for each CNF instance.

Possible solution-set-preserving reductions were investigated [10]. We have found that 1-literal clauses often appear in the CNFs. Variables of these literals must be assigned to a value constant over all solutions, for the SAT to be satisfiable. Repeated application of 1-literal clause elimination will identify most of constant variables. The rest of them are detected by fixing the variable to a constant value and solving the SAT problem, see [14]. The set of variables fixed to a constant value for every SAT solution is called a *backbone* [15]. Next, we reduce the number of clauses by removing duplicities, absorbed clauses, and by creating resolution terms [16]. All these reductions preserve all SAT solutions.

Better understanding of CNF instances can be useful in their further processing. That is why more than 180000 ATPG CNF instances from ISCAS and ITC benchmarks had been generated and their properties and possible reductions were closely analyzed.

Experimental results show that in the average 60% of variables and 65% of clauses can be removed by solution-set-preserving reductions. Moreover, in average 57% of variables have the same values in each SAT solution (*backbone* size) [10, 15]. Constant variables are assigned and great numbers of clauses are satisfied by resolutions which simplify the SAT solution. All the CNF instances or their subset can be stored in the memory. Thus, the number of repeatedly generated CNF

instances is reduced and SAT-Compress algorithm can be sped up. The backbone can be further utilized to detect CNFs which cannot be satisfied with the constraints given by variables assignments.

Theoretical studies proved that ATPG SAT instances are easy to be solved [9]. The average percentages of clauses having a given number of literals were measured. Experiments show that ATPG CNFs consists of 3% of 1-literal clauses, 70% of 2-literal ones, 24% of 3-literal clauses, and 2% of 4-literal ones. It was shown that 2+p-SAT problems behave like 2-SATs for p<0.4 and like 3-SAT for p higher (p is the percentage of 3-literal clauses) [15]. Based on this observation it can be concluded that CNF instances (SAT instances) are similar to 2+p-SATs, with p=0.24. Therefore, they behave like 2-SATs and are easy to be solved. Our experimental results have confirmed theoretical studies.

From these observations it can be concluded, that a significant reduction of CNF instances is possible and the generated CNFs are also easy to be solved. Reduced instances can be stored in the memory to speed up the compression process. Moreover, the set of possible CNFs can be dynamically changed based on the learned backbone and its match with constrains. We assume, that dynamic activation of CNFs based on the backbone can dramatically decrease the number of processed CNF instances which can not fulfill constrains and speed up SAT-Compress algorithm.

3.2 Improvements of Compression Efficiency

Another considered aspect is the compression efficiency. SAT-Compress is a simple greedy algorithm. Each test pattern is generated based on variables assignments given by its predecessors, thus the initial pattern can affect bitstream lengths. Permutations of PIs (Primary Inputs) can also affect bitstream lengths. Each test pattern is generated pursuant to *mask*, which is made from a previous test pattern by its shift and the LSB is replaced by a DC. The LSB and nearby bits can be less constrained. We assume that the compression ratio can be further improved by ordering of PIs based on its occurrences in test patterns. Our extensive evaluation confirms that the result quality depends on the initial state as much as on the order of primary inputs. Figure 3 shows an example of distributions of bitstream lengths using different starting patterns (initial states) or different orders of primary inputs (PIs permutation). The compressed bitstream lengths seem to have Gaussian-like distribution for both cases and the only difference is displacement. As a default state for both measurements were used "initial state"/"PIs permutation" producing the bitstream length equal to median of the bitstream lengths.

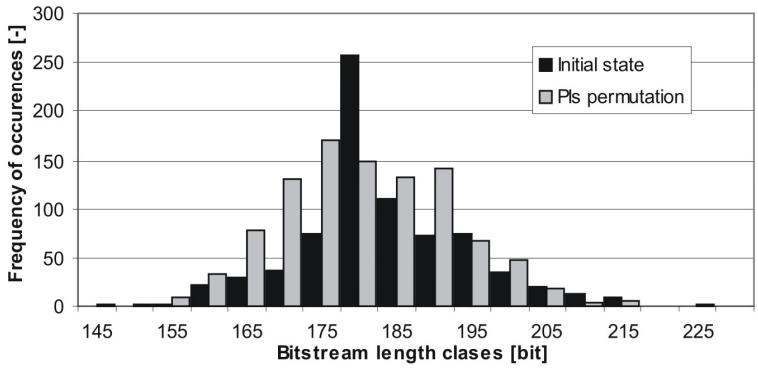

Figure 3: Frequency of bitstream lengths distribution for different initial state or permutation of PIs (c499)

We have found that the compressed bitstream lengths for some circuits are either better or worse in comparison with COMPAS [12] independently of the initial state. Also the compression ratio is surprisingly better for ISCAS'85 [13] than ISCAS'89 [17] benchmark set. We assume that it can be caused by the structure of the circuits. Numbers of paths from POs (Primary Outputs) to PIs (see Figure 4) have been measured and analyzed. Table 1 consists of the compression ratios of COMPAS and SAT-Compress algorithm (compared to compacted test lengths generated by Atalanta [18]), both

starting with an all-zero test pattern, in the columns "*ratio*". The "*diff.*" column indicates the difference of compression ratios of the SAT-Compress and COMPAS algorithm. A positive value indicates better compression ratio of SAT-Compress. The ratio of the number of paths from POs to PIs (e.g. Figure 4b) compared to the maximal number of paths from POs to PIs (e.g. Figure 2a) is presented in the column "*comp.*". Column "*Bench.*" presents the name of the benchmark. Table 1 signifies that a greedy algorithm based on implicit representation such as SAT-Compress can handle compression of test patterns for circuits with high number of paths from active POs to PIs more efficiently (some of ISCAS'85), but it is not able to benefit its lower number (most of ISCAS'89) such as COMPAS. Much more measurement is needed to prove this assumption and find solutions of this drawback.

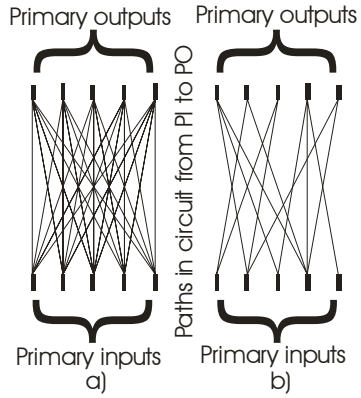

Figure 4: Number of paths from POs to PIs of the circuit (a) maximal, (b) less.

Bench.	COMPAS ratio [%]	SAT-Compress ratio [%]	diff. [%]	comp. [%]
C432	89.6	89.2	-0.4	89.2
C499	89.9	91.6	1.7	100
C880	87	79.8	-7.2	26.8
c1355	69.8	90.8	21	100
S344	82.2	75.8	-6.4	19.3
s1196	84.4	70.2	-14.2	37.7
s1423	90.6	87.5	-3.1	31

Table 1: Number of paths from active POs to PIs of circuits and compression ratios for COMPAS and SAT-Compress.

4 Future work

Our areas of interest are implicit representations and customized test patterns generation. In the future we will continue in CNF investigation and possibly find some efficient handling of the CNF instances based on their properties such as a backbone size or reductions possibilities. Utilization of implicit representations will be further explored on test patterns compression. SAT-Compress algorithm can be sped up according to proposals in Section 3 and more efficient heuristics can be applied to increase the compression ratio. Such a focused research should give us better understanding the behavior of implicit representations in specific applications, which can be beneficial in the future to create a general concept of customized test generation based on specific constrains.

Further investigation of circuit's properties such as communication complexity and its influence on testing are also needed to be explored. These observations can help us to find and overcome reasons of lower efficiency in test patterns compression for some circuits.

New implicit representation and set of constrains based on known representations and previous observation will be considered.

5 Conclusion

General concept of customized test patterns generation based on implicit representations of test patterns set has been proposed and discussed. Brief overview of implicit representations and their properties has been made. Implicit representation in CNF has been chosen to be very suitable as test patterns set representation and its properties has been discussed on formerly published SAT-Compress algorithm [2].

Acknowledgment

This research has been supported by MSMT under research program MSM6840770014, by the grant of the Czech Grant Agency GA102/09/1668 and the grant of the Czech Technical University in Prague, SGS10/118/OHK3/1T/18.

References

[1] Drechsler, R., Eggersglüß, S., Fey, G., Tille, D., "Test Pattern Generation using Boolean Proof Engines," Publisher Springer Netherlands, ISBN 978-90-481-2360-5, 2009, XII, p. 192.

[2] Balcárek, J., "Test Patterns Compression Techniques Based on SAT Solving for Scan-Based Digital Circuits," Počítačové architektury&diagnostika, Soláň (ČR), 2009, pp. 26-31.

[3] T. Larrabee. Test Pattern Generation Using Boolean Satisfiability. Test Pattern Generation Using Boolean Satisfiability. IEEE Transactions on Computer-Aided Design, pages 4-15, Jan, 1992.

[4] Bryant, R. E., "Graph-based algorithms for boolean function manipulation," IEEE Trans. Comput. 35, ISSN:0018-9340, 1986, pp. 677-691.

[5] Balcárek, J., "Řešení problému splnitelnosti booleovské formule (SAT) pomocí binárních rozhodovacích diagramů (BDD)," Bakalářská práce, FEL ČVUT v Praze, 2007, p. 71.

[6] R. Drechsler, D. Sieling, "Binary decision diagrams in theory and practice," Springer Trans., Berlin, 2001, pp. 112-136.

[7] J. Shi, G. Fey, R. Drechsler, A. Glowatz, J. Schoffel, and F. Hapke, "Experimental studies on SAT-based test pattern generation for industrial circuits," In Int'l Conf. On ASICS, 2005, pp. 967-970.

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

[9] Prasad, M. R., Chong, P., Keutzer, K.: Why is ATPG easy?. In Proc. of the 36th Annual ACM/IEEE Design Automation Conference, New Orleans, USA, June 21 - 25, 1999, pp. 22-28.

[10] Balcárek , J., Fišer, P., Schmidt, J., " On Properties of SAT Instances Produced by SAT-Based ATPGs," In Fifth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, ISBN 978-80-87342-04-6, Znojmo, 2009, pp. 3-10.

[11] C. Su, K. Hwang, "A Serial Scan Test Vector Compression Methodology," in Proc. ITC, 1993, pp. 981-988.

[12] O. Novák, J. Zahrádka, "COMPAS – Compressed Test Pattern Sequencer for Scan Based Circuits," in Proc. of EDCC, 2005, pp. 403-414.

[13] F. Brglez and H. Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan, Proc. Of International Symposium on Circuits and Systems, 1985, pp. 663-698.

[14] Singer, J., Gent, I.P., Smaill, A.: Local Search on Random 2+p-SAT. In Proc. of the 14th ECAI, 2000, pp. 113-117.

[15] Monasson, R., Zecchina, R., Kirkpatrick, S. Selman, B., and Troyansky, L.: 2+p-SAT: relation of typical-case Complexity to the nature of the phase transition. In Random Structures & Algorithms, Vol. 15, Issue 3-4, Oct.-Dec. 1999, pp. 414 – 435.

[16] Davis, M., Putnam, H.: A computing procedure for quanti_cation theory. Journal of the ACM (JACM), Vol. 7, Issue 3, July 1960, pp. 201-215.

[17] F. Brglez, D. Bryan and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits, Proc. of International Symposium of Circuits and Systems, 1989, pp. 1929-1934.

[18] H.K. Lee and D.S. Ha, "Atalanta: an Efficient ATPG for Combinational Circuits", Technical Report, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993.