IMPLICIT REPRESENTATIONS IN THE DIAGNOSTICS OF THE DIGITAL CIRCUITS

Jiří Balcárek
Informatics and Computer Science, 2-th 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@fit.cvut.cz

Abstract. This paper discusses the utilization of implicit representations of test vector set and techniques of their processing in the field of diagnostics of the digital circuits. Many tasks in the diagnostics of digital circuits must deal with processing of the vector sets whose size can be considerable for large circuits. The memory requirements and also difficult manipulation of large sets of vectors can decrease the efficiency of diagnostic algorithms significantly. The set of vectors can be compactly and with low memory requirements represented implicitly, which can increase the robustness of diagnostic algorithms. The efficiency can be improved as well by proper implicit techniques of its processing. Previous work on implicit representations in the diagnostics is briefly summarized and some properties of an implicit representation of test vectors in the CNF (Conjunctive Normal form) are discussed as well as techniques of its processing. The summary of our observations forms a simple background for a general methodology of processing of implicit representations.

Keywords. Implicit representation, implicit techniques, SAT, ATPG.

1 Introduction

Despite of the growing number of analog and mixed signal parts in electronic devices, there are still many challenges in digital circuits diagnostic and dependability. Many algorithms in diagnostics and dependability of the digital circuits have to process a great number of vectors or their sets. The efficiency of these algorithms can dramatically depend on the representation of test vectors sets. Great amount of vectors must be processed in effort to find the best solution and memory or time requirements can be unfeasible. Implicit representations such as BDDs (Binary Decision Diagrams), ZDDs (Zero Decision Diagrams), CNF (Conjunctive Normal Form) and similar ones can compactly and with low memory requirements represent the whole set of vectors, which can be further processed by simple operations to possibly reach the desired solution. We suppose that a proper implicit representation and implicit techniques themselves can naturally increase the robustness of these algorithms and show us a new way of vector sets processing.

The paper is organized as follows. A brief overview of the previous work on implicit representations in the field of diagnostics is shown in Section 2. Section 3 discusses the implicit representation of the test vector set in CNF, its properties, possible utilization, bounds and beneficial techniques of its processing. Future work on a general methodology of implicit representation processing is suggested in Section 4. Section 5 gives a brief conclusion.

2 Previous Work

Implicit representations of test vector sets and techniques of their processing have been explored for a long time and they are already utilized by many algorithms. This section gives a brief overview of implicit representations and their utilization in the field of diagnostics of digital circuits.

A common application of implicit representations in the diagnostics is a generation of test patterns (test vectors). A set of all test patterns for a fault in the digital circuit is implicitly described as an instance of SAT (SATisfiability) problem by a Boolean formula in the CNF (Conjunctive Normal Form). Searching for a test pattern for a fault is thus transformed into solving an instance of the satisfiability problem. This method is utilized in a SAT-based ATPG (Automatic Test Pattern Generator) [1].

Generation of test patterns with some constraints is also a common process in digital designs testing. Test patterns are constrained to be better compressed, tailored for a scan-based designs, to speed up the test generation process, to avoid illegal test patterns (on primary inputs, buses, tri-state elements), etc.

Test patterns compression based on overlapping of test patterns [2] is a particular application of the constrained test patterns generation. The set of test vectors for a fault is represented implicitly as a Boolean formula in the CNF as in the SAT-based ATPG. The next test vector to be overlapped is constrained by specified bits in the previously generated test vector (the initial test vector is an all-zero seed). This modification of the SAT-based ATPG is known as SAT-Compress algorithm [2] and its compression efficiency is comparable with state-of-the-art compression tools, but the time of the compression grows up significantly with the size of the circuit.

In [3], an implicit representation of all test sequences for a fault is used to check for conflicts with rule matrices (of the cellular automata) during the test patterns generation constrained to the cellular automata. This technique is used for testing of midsize controllers. The entire set of the test sequences of the controller under test is implicitly represented by a BDD (Binary Decision Diagram). The BDD is explored and only those sequences which can be reproduced by cellular automata are selected. A specific algorithm and the way of application of the constraints were not described in the paper.

Low power tests are mostly built from pre-generated test patterns [4] by their reordering. Another approach is to represent all test patterns for each fault implicitly and form constraints to choose the best sequence of test patterns by a heuristic algorithm (test sequence is built incrementally). The best test patterns to be ordered into the test sequence are searched instead of the ordering of the pre-generated test patterns [4].

3 The CNF implicit representation

We have focused on the implicit representation of the test vectors set in the CNF, because it is simple and scalable (its size grows linearly with the number of gates). The CNF is also used by many state-of-the-art ATPG tools [1][2].

Properties of the CNF and techniques of its processing were examined in the SAT-Compress algorithm [2]. This research can give us much better idea about its possible utilization in different algorithms. Moreover, further analysis should show the limits of usability of the CNF and also its strengths and weaknesses.

3.1 The CNF analysis

Difficulty of SAT Instances Produced by SAT-based ATPGs

The SAT problem is generally known to be NP-complete, thus its solution is considered to be time-consuming. However, SAT-based tools used for the diagnostics of the digital circuits are fast enough to be comparable with other state-of-the-art algorithms. This observation implies that SAT instances produced by SAT-based ATPGs (or other SAT-based algorithm) have some specific properties (unique structure) which makes them easy to be solved.

The theoretical proof that SAT instances produced by SAT-based ATPGs are easy to be solved is presented in [5]. Our further experimental evaluation [6] confirmed that SAT instances produced by SAT-based ATPGs consist of a majority of 2-literal clauses (1-literal 3%, 2-literal 70%, 3-literal 24%, 4-literal 2% and 5 or more literal less then 1%) and thus are easy to be solved. This observation is also validated in [1][7], which claims that SAT-ATPG spend on average 80% of the CPU time by generating of CNFs, instead of solving them. This experimental evaluation shows that solving the SAT is easy enough, but the circuit-to-CNF transformation can be a bottle-neck, thus some advanced transformation techniques and efficient CNF processing could be useful (e.g., dynamic clause activation [8]).

Satisfiability of SAT Instances Produced by SAT-based ATPG

The CNF implicitly represents the whole set of test patterns for a fault. If the fault cannot be detected, the set of test patterns is empty, thus a satisfiable solution of the CNF does not exist. Our research [6] shows that 99% of CNFs produced by the SAT-based ATPG are satisfiable on average. The number of SAT solutions (test patterns detecting a fault) is also very high [9] because the major part of the fault list usually consists of faults which are easy to be detected.

We have also generated random `clones' of the real instances, by substituting real variables in the SAT by random ones. While retaining the SAT instances structure, only 10% of them were satisfiable. This observation shows that SAT instances produced by the SAT-based ATPG are much more satisfiable than its random ‘clones’. Up to now, we have no fully relevant clue to explain this phenomenon. We have measured the connectivity of the SAT instances, when expressed as graphs [10]. However, we haven't found any significant difference between ATPG and random SAT instances of equal parameters.

The satisfiability of the constrained ATPG SAT instances in test patterns compression (by SAT-Compress algorithm) was also explored. For example 1,279,654 CNFs are needed to be generated and solved for the c7552 ISCAS benchmark [11], but only 456 CNFs of them fulfill the constraints given by variables assignments. According to our extensive measurements, 98% of generated CNFs are unsatisfiable with given constraints on average. This can be caused by the ‘structure’ of the ATPG SAT instances. According to [12], conflicts between variable assignments in the ATPG SAT instances are locally bounded. Our research on difficulty of SAT instances mentioned in the previous section shows that ATPG SAT instances contain 70% of 2-literals clauses on average [6]. Variables whose values must be fixed can be easily propagated through the chain of implications made of 2-literal clauses and force conflict assignments of variables. This implication chain can be used as CNFs unsatisfiability detection engine and thus speed their processing up [12][13][14].

Solution-Set-Preserving SAT Reductions

Possible solution-set-preserving reductions were investigated [6], because the number of the processed CNFs can be considerable, thus the size of the CNF can also affect the speed of its processing and the memory requirements. We have found that 1-literal clauses often appear in CNFs. Variables of these literals must be assigned to a constant value in all satisfiable solutions. Repeated application of 1-literal clause elimination will identify most of the constant variables. The rest of them are detected by fixing the variable to a constant value and solving the SAT problem, see [15]. The set of variables fixed to a constant value for every SAT solution is called a backbone [16]. Next, we reduce the number of clauses by removing duplicities, absorbed clauses, and by creating resolution terms [17]. All these reductions preserve all SAT solutions.

Experimental results show that 60% of variables and 65% of clauses can be removed by solution-set-preserving reductions on average. Moreover, 57% of variables have equal values in each SAT solution (*backbone* size) on average [6][16]. The backbone size supports our assumption about high unsatisfiability of the constrained ATPG SAT instances. Great number of variables must be fixed, thus the probability of a conflict assignment or implication is high. The backbone can be further utilized to detect CNFs which cannot be satisfied with the constraints given by variables assignments. It is also obvious that a significant reduction of the size of the ATPG SAT instances is possible, but searching

for the backbone and application of other solution-set-preserving SAT reductions can be unfeasible. Only simple solution-set-preserving reductions such as implications of the 1-literal clauses are efficient, but the reduction of the CNF size is much less.

3.2 Techniques of CNF processing

Circuit to SAT Transformation Techniques

The first technique of the circuit to SAT transformation was described in [13]. The difference between each primary output (PO) of the circuit exhibiting a given fault and its counterpart in a fault-free copy is detected by a XOR gate. Difference signals for all outputs are OR-ed to obtain a general difference signal. If there exists an input vector that sets this signal to 1, the fault is detectable. Each gate in this circuit is described by its characteristic function which is added to the CNF. The variable corresponding to the output of the OR gate is set to logic 1. This simple CNF transformation generates large CNFs with a great number of redundancies, thus it is not usually used by state-of-the-art tools.

Another approach [1] of the circuit to SAT transformation tries to reduce the CNF size by elimination of redundancies. Gates in the input cone of the fault in the fault-free and faulty circuit are the same, thus they are added to the CNF only once. Moreover, only gates in the input and output cone of the fault are added to the CNF. Primary outputs of the output cones are handled as in previous case. This transformation of the circuit to the SAT produces much smaller CNFs and their size can be further decreased by additional structural information. The value of the variable corresponding to the faulty signal can be assigned to a complement of the value assigned to variable corresponding to the fault-free signal. This transformation of the circuit to CNF is simple and produces small SAT instances, but their solving can be time-consuming for large circuits (large CNFs).

The previous approach to the circuit to SAT transformation produces CNFs without redundancies, but it can be further extended [1][18] by additional clauses to simplify SAT solving. The set of additional clauses describing possible active path (D-chain) is added to CNF for each gate [18]. These clauses increase the size of the CNF significantly (e.g., for c6288 [18], several CNFs have over 50 000 literals), thus some reduction of the CNF is needed. The CNF instances built this way grant a high performance of the designed tool, because additional structural information can significantly speedup solution of hard SAT instances (SAT instances for the faults which are hard to be detected).

On-the-fly CNF generation vs. CNF storing

The class of algorithms such as constrained test patterns generators must solve a set of the same CNFs in the loop. These CNFs can be stored in the memory or generated ‘on-the-fly’. Our experimental results [14] indicate that the time consumption of the algorithm is almost the same for both CNFs storing and their generation on-the-fly, but CNFs storing can be unfeasible for large circuits. In fact, both these techniques can be unfeasible because of time or memory requirements. These drawbacks can be overcome by a combination of both, the *dynamic clause activation* [8]. The SAT solver works with a dynamic SAT instance. The complete list of clauses for a fault-free circuit is stored in the memory and forms a clause database. The SAT solver generates the CNF for the faulty part of the circuit and runs the decision heuristic guiding the search for the SAT solution. Additional necessary clauses from the fault-free circuit are sent from the clause database on request of the decision heuristic. The dynamic clause activation engine is used in the SAT-based tools in the industry, because it is highly efficient even though the size of the clause database can be a bottle-neck for large industrial circuits [1].

Early UNSAT Detection

Early detection of the unsatisfiable CNFs can be a vital technique for many algorithms such as constrained test patterns generation. High number of unsatisfiable CNFs to be generated and solved causes a significant time overhead. Several techniques of early unsatisfiability detection have been proposed.

The first class of techniques is based on the implication engine [13] which can quickly propagate constraints given to the SAT solver (or decisions of the SAT solver) and detect unsatisfiability.

Another approach [12] is based on the claim that all conflicts in the CNF are locally bounded. This algorithm does not generate the whole CNF for a fault, but the SAT instance is created from gates close to the faulty signal. Only characteristic functions of two levels of gates from the output cone of the faulty signal are added to the CNF with the corresponding input cone. The SAT solving of this small CNF is obviously much faster then solving of the whole CNF, thus the detection of unsatisfiability is fast. The main disadvantage of both these techniques is a need of extra CNF generation and solving to detect unsatisfiability of the CNF.

Our technique is based on a powerful implication engine with a table of implications [14]. The table of implications is generated once when the algorithm starts and consists of 2-literal clauses from the CNF of the fault-free circuit. The assigned constraints and structural information of the fault are propagated through implications and conflicts in the variable assignments indicate unsatisfiability. This technique is able to detect 42% of unsatisfiable SAT instances on average and speed the application (the SAT-Compress algorithm) up more then twice. It can be further improved by searching for additional implications coming from the previous variable assignment. This new 'dynamic' implications can detect additional 11% of unsatisfiable instances, but the speedup is not as high, because searching for dynamic implications causes an additional time overhead. The main advantage of this approach is detection of unsatisfiability within CNFs generation and solving. However, its efficiency strongly depends on the number of constraints and implications, thus its usability is limited to a specific class of applications.

4 Future Work

The evaluation of the CNF properties and techniques of its processing should give us much better idea about its possible utilization in different algorithms. Moreover, this analysis should show the usability limits of the implicit representation and also its strengths and weaknesses.

Another promising application of implicit techniques is processing of Boolean networks consisting of general purpose blocks (such as LUTs in FPGAs) with described fault behavior. An efficient processing of such networks can be utilized by a wide range of applications from diagnostics and dependability. These networks consist of a great number of faults to be processed and also the wide range of the fault models must be considered.

The information about processing of implicit representations learned when designing the SAT-Compress algorithm and its usability in Boolean networks composed of general blocks will be revised and extended by further observations. The exact application of the implicit techniques and selection of the proper implicit representation will be considered.

The research on implicit representations applied in these exact tasks from the diagnostics and dependability of the digital circuits and the differences between its processing by standard algorithms (not using implicit techniques) should give us reliable knowledge of implicit representations behavior and techniques of their proper processing. This information can be further generalized on a class of the similar tasks from diagnostics and dependability and finally the methodology of utilization of implicit representations for a class of the algorithms will be defined.

5 Conclusion

This paper discusses properties of the implicit representation in the CNF such as hardness, satisfiability and possibilities of solution-set-preserving reductions. It also describes some known techniques of the circuit to SAT transformations, its processing and detection of unsatisfiable instances. This brief overview forms a background for further analysis and methodology of implicit representation processing in the diagnostics of digital circuits.

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, SGS11/089/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., Fišer, P., Schmidt, J., Test Patterns Compression Technique Based on a Dedicated SAT-based ATPG, Proc. of 13th Euromicro Conference on Digital Systems Design (DSD'10), Lille (France), 1.-3.9.2010, pp. 805-808.

[3] Fummi, F., Sciuto, D., "Implicit test pattern generation constrained to cellular automata embedding," Proc. of 15th IEEE VLSI Test Symposium (VTS'97), 1997, pp.54.

[4] Girard, P., Nicolici, N., Wen, X., "Power-Aware Testing and Test Strategies for Low Power Devices," Publisher Springer Netherlands, ISBN: 1441909273, 2009, p.353.

[5] 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.

[6] 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.

[7] Marques-Silva, J. P., Sakallah, K. A., "Robust search algorithms for test pattern generation," in Int'l Symp. on Fault-Tolerant Computing, 1997, pp. 152–157.

[8] Eggersglüß, S., Tille, D., Drechsler, R., "Speeding up SAT-Based ATPG Using Dynamic Clause Activation," In Asian Test Symposium, 2009, pp. 177-182.

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

[10] Kravchuk, O., Pullan, W., Thornton, J., Sattar, A., : An investigation of variable relationships in 3-SAT problems. Al 2002: Advances In Arti_cial Intelligence, 2557, 2002, pp. 579-590.

[11] Brglez, F., Fujiwara, H., "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.

[12] Tille, D., Drechsler, R., "A fast untestability proof for SAT-based ATPG." In Proceedings of the IEEE Symposium on Design and Diagnosis of Electronic Circuits and Systems, pages 38–43, 2009.

[13] Larrabee, T., "Test pattern generation using Boolean satisfiability," IEEE Transactions on Computer-Aided Design for Circuits and Systems, 11(1):4–15, 1992.

[14] Balcárek, J., Fišer, P., Schmidt, J. "Techniques for SAT-Based Constrained Test Patterns Generation," Proc. 14th Euromicro Conference on Digital Systéme Design (DSD'11), Oulu (Finland), 31.8.-2.9.2011, p. 8.

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

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

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

[18] Stephan, P., Brayton, R. K., Sangiovanni-Vincentelli, A. L., "Combinational test generation using satisfiability." IEEE Transactions on Computer-Aided Design for Circuits and Systems, 15:1167–1176, 1996.