TEST PATTERNS COMPRESSION TECHNIQUES BASED ON SAT SOLVING FOR SCAN-BASED 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, FEL, Dept. of Computer Science & Engineering
Karlovo nám. 13, Praha 2, Czech Republic
E- mail: balcaj3@fel.cvut.cz

Abstract: In the paper we propose a new method of test patterns compression based on SAT (SATisfiability) solving. By test patterns compression we can dramatically decrease test memory requirements for test patterns storing. This compression method is very suitable for scan-based digital circuits. Test patterns are decompressed in the scan chain during the test, no additional hardware is required. By this way we can also decrease the data bandwidth between ATE (Automatic Test Equipment) and the internal test mechanism. The main idea is based on test patterns overlapping introduced in the COMPAS (COMpressed Pattern Sequencer) compression tool [1]. Our proposed algorithm is based, as well as COMPAS, on patterns overlapping. During the test generation, we are trying to *efficiently generate* vectors as candidates for an overlap, unlike COMPAS, which is based on efficient overlapping of pre-generated test patterns. We introduce our basic algorithm and show results obtained for standard ISCAS'85 and '89 benchmark circuits. The results are compared with the COMPAS compression tool.

Keywords. Test patterns compression, testing, ATPG, SAT.

1 Introduction

Testing of digital circuits is nowadays quite a difficult task. With the growing complexity of designs, scan-based techniques of testing are becoming very popular. The chain of registers (scan chain) is fed with test patterns through a serial interface. It means that each test pattern is shifted into the scan chain and then the circuit under test (CUT) response is shifted out to the response compactor. Generally, we can use a deterministic or pseudorandom test pattern set for testing. Both have their advantages and disadvantages. Pseudorandom testing may be easily realized by a linear-feedback shift register (LSFR) or by other automata. Test patterns generated in such a way cover most of easily-detectable faults, but the patterns can be quite inefficient in covering random pattern resistant faults. Also a great number of test patterns are needed to generate. It means that the time consumption may grow up significantly. On the other hand, we can generate deterministic test patterns able to detect all detectable faults. Its size may also be considerable and we need to send these test patterns to scan chains. This data transfer is realized by TAM (Test Access Mechanism), which creates an interface between ATE (Automatic Test Equipment) and the on-chip test mechanism. Design requirements force us to make the TAM as narrow as possible, but sending test patterns through a narrow TAM may cause a considerable growth of the time consumption. That is why we are trying to compress test patterns and by this way decrease the bandwidth between ATE and TAM. There are several methods used for a test patterns compression. One of the most interesting ones is based on a test patterns compaction and test patterns overlapping. The test patterns compressed by this way may be easily decompressed by exploiting scan chains. No additional hardware is required and the bandwidth between ATE and TAM sags significantly.

In this paper we introduce a new algorithm for test patterns compression based on SAT solving. This new algorithm doesn't use test patterns pre-generated by an ATPG (Automatic Test Pattern Generator), but it gradually generates most suitable test patterns in the process, to reach the best overlap and maximize the compression. A SAT-ATPG [5] based algorithm is used for the test patterns generation. A CNF for each fault is generated. This CNF represents a set of all test patterns detecting a given fault. Its particular SAT solution represents one test pattern detecting the respective fault. Thus, test patterns for a given circuit are represented implicitly, by a set of SAT instances. A proper selection of processed faults and their respective SAT solutions could yield optimum test patterns compression, in general. However, an exhaustive state space exploration is computationally unfeasible. We propose a heuristic method using these principles, yielding a competitive test patterns compression. Experimental results compared with the COMPAS compression tool are shown.

1.1 Test Patterns Compresion Technique Based on SAT Solving

The basic compression idea is the same as for the COMPAS compression tool. We try to find the best overlap of test patterns that are serially shifted into the scan chain. This approach was firstly described in [2]. This algorithm generally tries to find contiguous and consecutive test patterns for the test patterns currently present in the scan-chain. These patterns are checked whether they match with one or more remaining test patterns, which were previously generated and compacted with the help of some ATPG and which were not used in the scan chain sequence yet. The compacted test patterns are reordered by a heuristic algorithm to obtain maximum overlapping. The main disadvantage of these methods [7] is that they are either computationally complicated and thus they aren't usable for large circuits, or the size of the compressed test patterns is greater than the size of the test patterns compressed by other compression methods. The COMPAS compression tool is based on a similar approach, but it doesn't use compacted test patterns. Here the test patterns that are to be compressed are pre-generated by an external ATPG. These test patterns should contain as many don't care bits (DC bits, means not specified) as possible. Greater number of don't care bits grants the algorithm much more possibilities to combine test patterns and reach better compression. Another improvement is a simulation after every test pattern application and searching for best successors of a given starting test pattern (usually an all-zero pattern). These improvements make the COMPAS algorithm very efficient. The only weakness of the COMPAS algorithm is the need for don't care bits. If the input test generated by the ATPG doesn't include lots of don't care bits, it is much harder for COMPAS to find the best overlap and the compression efficiency decreases significantly.

In our new approach we try to eliminate the previously mentioned weakness of the COMPAS algorithm. The main idea is not to overlap test patterns pre-generated by an ATPG, but to generate most suitable test patterns in the process, to reach the best overlap. The basic question is how to find these test patterns. Each fault in the fault list has its set of test patterns by which it is detected. If we realize this, we can assume that if we were able to pick the right (granting the best overlap) pattern for each fault, we can reach the best possible compression of the test patterns. Because computation and storing of all these test patterns is quite inefficient, we were forced to find out another, more efficient way of representation. We research possibilities of *implicit representations* of test patterns. We have found that we can take advantage of principles of SAT-based ATPGs [5, 8] and efficiently represent all test patterns for one fault by an instance of a SAT problem in a CNF (Conjunctive Normal Form). The CNF test set representation is much less memory consuming than a standard tabular representation of the set of test patters. Our new algorithm generates a SAT instance in CNF for each fault and by solving this instance we acquire the next pattern for overlap. It is obvious that for a method based on a SAT solving we can expect a greater time consumption than for the COMPAS compression tool, but our main goal is to find the best overlap and maximize the test patterns compression.

1.2 Digital Circuit-To-CNF Conversion

Conversion of a digital circuit to CNF [5, 8] is an essential task for our SAT-based compression algorithm. The main idea of this conversion is shown in Figure 1. Each gate is described by its characteristic function and the whole circuit is the conjunction of these characteristic functions. Generation of the SAT instance in CNF for a fault means to make XOR of faulty and fault-free circuit outputs (X xor X‘).

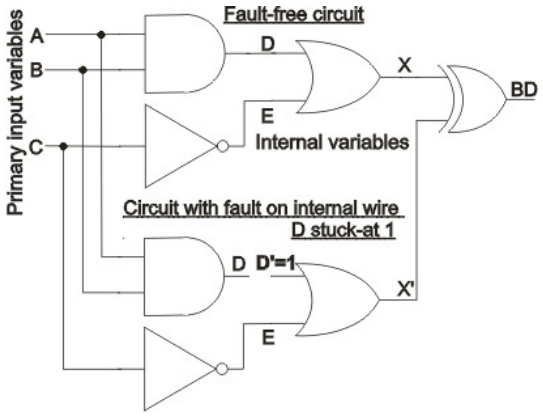

Figure 1: Faulty and fault-free circuit and its CNF example [5]

The characteristic function is derived from the basic function of the gate. We show the procedure for an AND gate and its basic function D = A ∧ B. Each function P=Q is logicaly equivalent to (P⇒Q)∧(Q⇒P) and that’s why we can transform the basic AND function into (D⇒(A∧B))∧((A∧B) ⇒D). Next we transform all implications into disjunctions by using the fact that P⇒Q is logicaly equivalent to ¬P∨Q and get the characteristic function for an AND gate (¬D∨A)∧(¬D∨B)∧(D∨¬A∨¬B) [5]. Then the CNF function for the fault-free circuit is:

(¬D∨A)∧(¬D∨B)∧(D∨¬A∨¬B)∧(C∨E)∧(¬C∨¬E)∧(X∨¬D)∧(X∨¬E)∧(¬X∨D∨E), (1)

and the CNF for a faulty circuit is:

(D’)∧(X’∨¬D’)∧(X’∨¬E)∧(¬X’∨D’∨E)∧(C∨E)∧(¬C∨¬E). (2)

The CNF of the output XOR of a faulty and fault-free circuit is:

(¬X∨X’∨BD)∧ (X∨¬X’∨BD)∧(X∨X’∨¬BD) ∧(¬X∨¬X’∨¬BD). (3)

Finally, the CNF for a stuck-at-1 fault on internal wire D is a conjunction of faulty-free (1), faulty (2) and the XOR of their outputs CNFs (3):

(¬D∨A)∧(¬D∨B)∧(D∨¬A∨¬B)∧(C∨E)∧(¬C∨¬E)∧(X∨¬D)∧(X∨¬E)∧(¬X∨D∨E)
∧(D’)∧(X’∨¬D’)∧(X’∨¬E)∧(¬X’∨D’∨E)∧(C∨E)∧(¬C∨¬E)∧
(¬X∨X’∨BD)∧ (X∨¬X’∨BD)∧(X∨X’∨¬BD) ∧(¬X∨¬X’∨¬BD) (4)

If there exists an assignment of variables, for which this CNF is satisfied, the fault is detectable. The assignment of input variables corresponds to the test pattern detecting this fault. In particular, the example CNF (4) is satisfied by assignment A=1, B=0, C=1, D=0, D’=1, E=0, X=0,X’=1, BD=1. Therefore, the stuck-at-1 at D fault is detected by the pattern (A, B, C) = (101).

2 The SAT-Compress Algorithm Principles

2.1 Best Overlap Finding

We try to find the best overlap by gradually building the compressed test patterns bitstream for the scan chain. The initial test pattern must be generated first. To do this, a fault from the fault list is picked, its CNF is generated and the SAT problem is solved for it. By this way we obtain a vector containing primary and internal variables set. Primary variables represent the primary inputs of the

CUT, the internal variables represent the CUT internal signals (interconnection), see Fig. 1. The test pattern has only primary input variables set, thus values of internal variables are not important for us. Bits of the first pattern are set according to the CNF solution. If the generated CNF (and its solution) doesn't contain some primary input variable, it is set as a DC. Figure 2 shows the next pattern generation example. The first pattern is simulated and all detected faults are deleted form the fault list. Then the pattern is shifted left and the first bitstream bit is shifted out. On the other side of the pattern, a DC bit is shifted in. By this way we obtain a new pattern. The DC bits might be set to logic 0 or 1. The best of DC bits setting is obtained by searching the fault list for a fault, whose CNF satisfies the input variable setting. A new test pattern is obtained, simulated and the search for a next pattern continues.

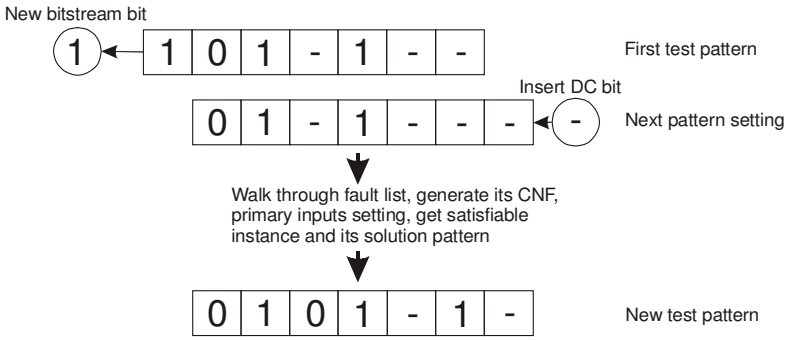

Figure 2: Next pattern generation example

2.2 Basic Algorithm Description

The basic algorithm is shown in Figure 3. First, we generate a complete fault list (FL) and pick one initial fault (the first one or any we choose). For it we generate its CNF. As a result of first CNF solving, the initial test pattern is obtained. The first pattern is simulated and all detected faults are deleted from the fault list. The following operation is shifting the first pattern left (see Figure 2). The first bit of the compressed bitstream and the next primary variable inputs setting is obtained. In the following steps we pick the next fault, generate its CNF and set primary input variables according to the previously obtained pattern. If satisfiable CNF is found for the given primary input setting, a next test pattern is obtained. In case that there doesn't exist a satisfiable solution for a given primary input variables setting, the pattern is shifted left by one bit, by which a new pattern is obtained (and a next bit of the bitstream is produced). A fault having a satisfiable CNF for this new input variable setting is looked for. Note that the new pattern contains more don't care bits, thus there is a bigger chance to find such a fault. These operations are performed while the fault list is not empty or until all care bits from the primary inputs setting are shifted out. The second case indicates, that the rest of faults in the fault list are undetectable (no variables are set and there is still no satisfiable CNF).

3 Experimental Results

In this part, results of our experiments are presented. Table 1 presents the compressed pattern bitstream lengths obtained by COMPAS and our SAT based algorithm, its compression ratio (compared to test lengths generated by Atalanta [4]) and the run time for our SAT based algorithm. The "Diff" row indicates the relative improvement (positive values) or deterioration (negative values) compared to COMPAS.

Table 1: Bitstream length and compression ratio reached by COMPAS and SAT based algorithm.

Circuit			c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552	s1196	s1238	s5378	s9234
Atalanta	Bit-stream	[bit]	1872	2214	3180	3444	3828	25164	7350	20292	1056	44505	4608	4704	54570	95095
COMPAS	Bit-stream	[bit]	195	224	412	1040	979	6091	726	1217	82	7115	717	783	1989	10589
	Comp. ratio	[%]	89,6	**89,9**	87	**69,8**	**74,43**	**75,79**	90,12	**94**	92,2	**84,01**	84,44	83,4	96,36	88,86
SAT based alg.	Bit-stream	[bit]	202	187	643	317	547	2064	1947	840	93	4779	1373	1423	2397	10858
	Comp. ratio	[%]	89,2	**91,6**	79,8	**90,8**	**85,71**	**91,8**	73,51	**95,86**	91,2	**89,26**	70,2	69,7	95,61	88,58
	Diff	[%]	-3,5	**16,5**	-36	**69,5**	**44,13**	**66,11**	-62,7	**30,98**	-11,8	**32,83**	-47,8	-45	-17	-2,48
	Time	[s]	10	8	25	81	169	1690	4541	207	87	7661	281	407	1007	77751

Table 1 presents compression results and run times for the SAT based algorithm. It is obvious that by our SAT based algorithm we are able to reach quite an interesting compression ratio. As we expected, the time for the compressed test patterns generating grows up significantly, but there are still much more possibilities to speed up the algorithm. Time results can not be compared with the COMPAS compression tool, because an on-line [6] application was used to compressed test patterns generation and the time measurement was not possible to handle. Generally, the speed of the COMPAS compression tool is higher than our SAT based algorithm and the difference grows with the circuit size. The time for solving of the hardest circuit (s9234) we measured was about one minute for the COMPAS algorithm.

4 Future work

Our place of interest are implicit methods in generating test patterns. In the previously mentioned algorithm, the CNF representation of test patterns for each fault was presented. The COMPAS compression algorithm processes previously generated test patterns and tries to combine them to reach the best compression ratio. Even if we were able to find the best compression of these test patterns, it needs not be the best compression at all. The result of the compression process is highly dependent on the input set of the test patterns. Our main goal is to explore possibilities of generating test patterns granting the best compression ratio. Each fault is represented by a set of all its test patterns (CNF in the presented algorithm). It is obvious, that if one pattern from each set is picked, we obtain the test patterns with full fault coverage. By complete exploring of the search space (all set of patterns and its permutations) we can guarantee generating the best set of the test patterns for overlap and reach the best compression ratio possible. Of course, the whole search space is huge and its complete exploring is not possible. In our future work we would like to explore possibilities of extracting the best set of the test patterns for overlap to reach better compression ratio. The proposed algorithm shows one of possible applications in this idea. The results imply that this approach may be quite efficient and offers a wide field for future improvements and possible applications in practice.

5 Conclusion

A new SAT based test patterns compression algorithm was proposed, implemented, and tested on a chosen set of ISCAS benchmarks. The compression ratio of this algorithm is as well as for the COMPAS tool quite high and in some cases even higher, but as expected, the run time may grow up significantly especially for larger circuits. The performance and scalability of this SAT based

algorithm depends on a trade-off between the compression ratio, time consumption and memory requirements. This new approach shows us another interesting way in the field of the test patterns compression.

References

[1] Novák, O. – Zahrádka, J.: COMPAS – Compressed Test Pattern Sequencer for Scan Based Circuits, Proc. of EDCC 2005, p. 403-414.

[2] Daehn, W., Mucha, J,: Hardware Test Pattern Generation for Built-in Testing. Proc. of ITC, 1981, p. 110-113.

[3] Balcárek, J.: Řešení problému splnitelnosti booleovské formule (SAT) pomocí binárních rozhodovacích diagramů (BDD). Bakalářská práce, FEL ČVUT v Praze, 2007, p. 69.

[4] Fišer, P.: Atalanta-m 2.0, 2005. http://service.felk.cvut.cz/vlsi/prj/Atalanta-M.

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

[6] Novák, O.: COMPAS - COMpressed PAttern Sequencer. http://iko.kes.tul.cz.

[7] Su, C., and Hwang, K.: A Serial Scan Test Vector Compression Methodology. Proc. of ITC 1993, p. 981-988.

[8] 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.

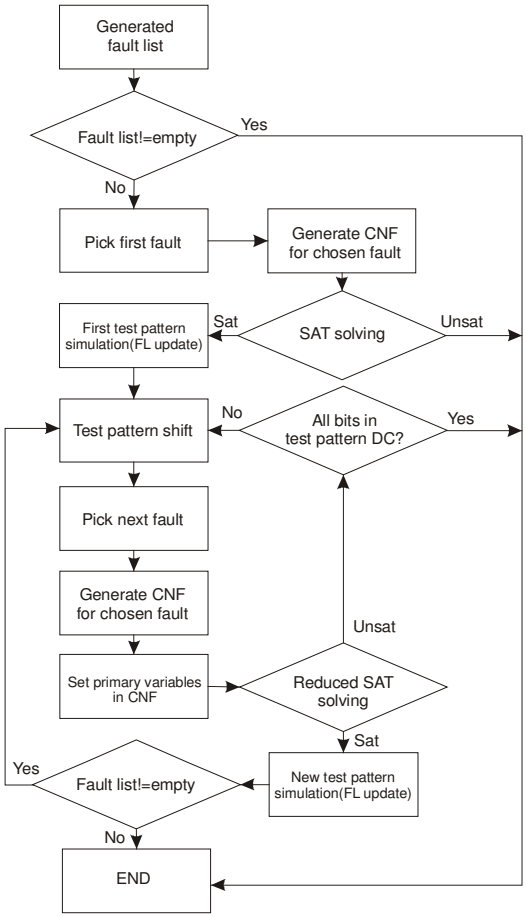

Figure 3: Basic algorithm