Test Patterns Compression Technique Based on a Dedicated SAT-Based ATPG

Jiří Balcárek, Petr Fišer, Jan Schmidt
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
balcaji2@fit.cvut.cz, fiserp@fit.cvut.cz, schmidt@fit.cvut.cz

Abstract— **In this paper we propose a new method of test patterns compression based on a design of a dedicated SAT-based ATPG (Automatic Test Pattern Generator). This compression method is targeted to systems on chip (SoCs) provided with the P1500 test standard. The RESPIN architecture can be used for test patterns decompression. The main idea is based on finding the best overlap of test patterns during the test generation, unlike other methods, which are based on efficient overlapping of pre-generated test patterns. The proposed algorithm takes advantage of an implicit test representation as SAT problem instances. The results of test patterns compression obtained for standard ISCAS'85 and '89 benchmark circuits are shown and compared with competitive test compression methods.**

Keywords: Test patterns compression, testing, ATPG, SAT

I. INTRODUCTION

Testing of digital circuits is quite a difficult task, for a huge amount test data needed to be delivered to the circuit under test (CUT). With the growing complexity of designs, scan-based test techniques are becoming a standard. The test patterns are shifted into the chain of scan registers (scan chain) through a serial interface and the circuit under test response is shifted out to the response compactor.

The volume of test patterns generated by an ATPG can considerably grow with the size of the circuit. Processing and storing of such a huge amount of data can be difficult and also the time to transport test patterns through a serial interface to the CUT may cause a considerable growth of testing time. That is why compression of test patterns is needed.

In this paper we introduce a novel test patterns compression algorithm *SAT-Compress* based on a design of a dedicated SAT-ATPG [1, 2]. The compression algorithm exploits an overlapping of test patterns. Unlike other compression methods based on this principle [3, 4, 5, 6], test patterns are not pre-generated by an ATPG; they are generated on the fly to reach the locally best overlap and maximize the compression ratio. Test patterns compressed this way can be easily decompressed by the RESPIN decompression architecture [7], which is intended to test system on chip (SoC) cores by reusing scan chains.

II. PREVIOUS WORK

The proposed test patterns compression method is basically based on finding the best overlap of test patterns pre-generated by an ATPG. The test patterns are serially shifted into the scan chain. This idea was described in [6] for the first time. This algorithm generally tries to find contiguous and consecutive test patterns having the maximum overlap. Deterministic test patterns are generated by an ATPG and compacted. Patterns in the scan chain are checked whether they match with one or more test patterns which were not employed in the sequence yet. In [5], the pattern overlapping problem is converted into a Traveling Salesman Problem (TSP), for which different heuristics have been proposed.

The COMPAS [3] test patterns compression tool is based on a similar approach, but it does not use compacted test patterns. Test patterns that are to be compressed are pre-generated by an ATPG as well. However, these test patterns should contain as many don't care bits (DC bits, means not specified) as possible. For this purpose, one test pattern for each fault from the fault list is generated. Greater number of don't care bits grants the algorithm much more possibilities to combine test patterns and reach better compression. Additional techniques, like a fault simulation after every test pattern application and searching for best successors of a given starting pattern are employed, making COMPAS very efficient in comparison with other compression tools, see Table II. The only weakness of COMPAS is the need for don't care bits. The number of DCs and the fact that the algorithm fully relies on a pre-generated test set can also affect the test compression ratio. When the test patterns are highly specified (they contain only few don't care's), it is much harder for COMPAS to find a good overlap of the test patterns and the efficiency of the test patterns encoding decreases which cause greater memory consumption [8].

Another compression technique based on the RESPIN architecture is presented in [9]. Suitable test patterns are produced by a standard ATPG tool performing dynamic compaction, while constraints to circuit inputs are applied. In contrast to this, we apply these constraints to the SAT instances, in a SAT-based ATPG.

The state-of-the-art compression/decompression architecture used in industry is the Embedded Deterministic Test (EDT) [10]. Here a high test compression is achieved by employing a dedicated but generic test decompressor. The compressed test patterns serve as seeds for a pseudo-random pattern generator ("ring register"), where they are decompressed and further distributed to scan-chains using a XOR-network structure ("phase-shifter"). The compressed test patterns are obtained as a solution of a set of linear

equations. Similarly to the previously mentioned test compression methods, the test patterns to be compressed need to be pre-computed by an ATPG. Again, high amount of test don't cares is essential for achieving a good compression ratio [10].

III. THE SAT-COMPRESS ALGORITHM PRINCIPLES

A. Basic Idea

In our new approach we try to eliminate weaknesses of COMPAS. The main idea is not to overlap test patterns pre-generated by an ATPG, but to generate most suitable test patterns on the fly, to reach the (locally) best overlap. If we were able to pick the right pattern for each fault in the right order, we could have reached the best possible compression of the test patterns.

Because explicit computation and storing of all test patterns is inefficient (and mostly even infeasible), we were forced to find another, more efficient way of test patterns set representation.

We have researched possibilities of implicit representations of test patterns. We have found that we can take advantage of principles of SAT-based ATPGs and efficiently represent all test patterns for one fault *implicitly*, by one SAT problem instance in a CNF. The CNF test set representation is much less memory consuming than a standard tabular test set representation. Our proposed test set compression algorithm sequentially generates the compressed test bitstream. A SAT instance in CNF is generated for each fault [2]. This SAT instance contains both variables representing the circuit's primary inputs and variables representing internal signals. The test pattern is determined by values of the primary input variables in the SAT solution; values of internal variables are of no significance. The variables not present in the generated CNF or in the SAT solution are set as don't cares (DCs). A SAT instance is solved with constraints given by the test pattern generated in the preceding algorithm step. A satisfiable solution of a SAT instance forms the next suitable test pattern for an overlap.

B. The SAT-Compress Algorithm Description

We try to find the best overlap by gradually building the compressed test patterns bitstream for the scan chain. The basic algorithm is shown in Fig. 1 and test pattern generation example for *n* PIs is shown in Fig. 2.

1) First, we generate a complete fault list (FL).

2) A zero state (all-zero test pattern) or a test pattern covering a chosen fault is used as the initial test pattern.

3) The pattern is simulated and all detected faults are deleted from the fault list. The leftmost bit of the pattern goes to the resulting bitstream.

4) The pattern is shifted left and a DC bit is put to its rightmost position. This is the mask for the next pattern (next pattern mask). The mask constraints the values of primary inputs (PI) in subsequent SAT instances. Only care bits generate the PI constraints.

5) To generate the next test pattern having the highest overlap with the previously generated one, a CNF for each fault in the fault list is generated, while the primary input variables are set according the mask. The orders of faults are of no significance. If a CNF for a fault is satisfiable for a given assignment of primary variables, a new test pattern is obtained. If none of these CNFs is satisfiable, the pattern is shifted one more bit left, which generates a new mask of SAT constraints.

6) Steps 3-6 are performed while the fault list is not empty or until all the mask care bits are shifted out. The latter case indicates that all the remaining faults are undetectable.

```
1) Generate FL                                   (FL = fault list)
2) TP[1..n] = "0..0"     (TP = test pattern = all-zero seed)
   mask[1..n] = DC                  (DC = don't care vector)
3) FL = FL - detected_by_simulation(TP[1..n])
   compressed_bitstream += TP[1]
4) mask[1..n] = TP[2..n] + DC[1]
5) do {
       for each fault in FL {
          Create CNF
          Apply mask to PIs in CNF
          if CNF is SAT {
             break the for loop
          }
       }
       TP[1..n] = CNF_Solution[1..n]
       FL = FL - detected_by_simulation(TP[1..n])
       compressed_bitstream += TP[1]
       mask[1..n] = TP[2 .. n] + DC[1]
   }
6) while
   (FL!=0 or (mask[1..n]==DC and All_CNF==UNSAT))
```

Figure 1. The SAT-Compress algorithm

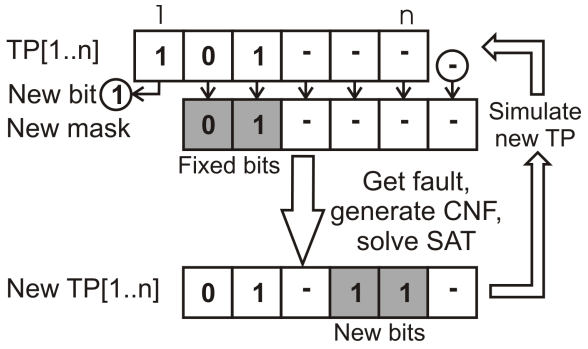

Figure 2. Test pattern generation example

IV. EXPERIMENTAL RESULTS

The experimental results and a comparison of our algorithm with similar compression methods are presented in the following two subsections. Unfortunately, comparison with [9] could not be made, since only results for 20 parallel scan-chains are presented in [9]. The measurement was performed on Intel Xenon CPU - 2GHz with 4GB RAM.

TABLE II. COMPAS AND SAT-COMPRESS TEST LENGTHS

Bench.	COMPAS			SAT-Compress			
	bits	*avg.*	*var.*	*bits*	*avg.*	*var.*	*time [s]*
c17	9	9	1.5	11	12.3	1	0
c432	195	218	30.6	189	198.2	21.8	3
c499	260	303.7	47.6	187	198.2	8.7	6
c880	540	412.7	44.3	410	561.5	60.99	10
c1355	1040	1126	68.6	349	-	-	74
c1908	1009	989.8	49.6	624	-	-	229
c2670	6553	5940	269.7	2223	-	-	2305
c3540	747	743.7	32.9	1622	-	-	3569
c5315	1255	1159.5	64.8	881	-	-	156
c6288	82	95.2	36.6	97	-	-	165
c7552	6005	6430.7	345.2	4840	-	-	7406
s27	16	13.2	2.3	23	21.5	2.1	0
s208	130	124.1	9.1	202	209	15.9	0
s298	101	79.6	5.5	137	139.7	8.2	0
s344	85	80.9	6.7	116	114.7	10.2	0
s349	85	80.1	6.8	116	114.4	10	0
s382	123	111.3	6.6	191	179.2	12.8	0
s386	264	255	10.8	304	319	12.5	4
s400	121	109	7.2	173	170.9	9.7	0
s420	352	315.9	29.1	624	574.8	46.4	11
s444	116	107	7.5	160	150.6	10.5	0
s510	160	156	8.9	210	211.8	10.3	2
s526	344	349.8	18.4	521	482.8	22.7	4
s526n	344	350.2	18.1	493	-	-	3
s641	397	393.3	24.2	710	670.7	28.4	15
s713	428	403.8	26.1	642	672	38.7	32
s820	460	504.6	21.9	697	697.7	26.3	28
s832	494	498.8	19.5	729	690.7	24.1	36
s838	920	762.3	79.6	1800	1638.3	103.5	136
s953	723	700.4	24.7	825	-	-	51
s1196	740	738.5	23.5	1211	1202.2	41.9	173
s1238	741	769.2	24.3	1300	1268.7	41.7	365
s1423	596	621.5	38.6	794	827.6	43.9	72
s1488	488	461.5	15.1	546	-	-	20
s1494	431	451.3	16.6	573	-	-	25
s5378	2148	1995.6	73.8	2407	-	-	631
s9234	11594	11309.6	310.7	9928	-	-	68119
s13207	4163	-	-	10457	-	-	67751
s15850	8234	-	-	12987	-	-	114932
s38417	24198	24926.3	1717.7	19291	-	-	91713
s38584	7291	-	-	14271	-	-	122143

A. Comparison of Different Compression Techniques

A comparison of SAT-Compress with other state-of-the-art test compression techniques is presented in Table I. The first column "*Bench*." represents the benchmark name. The compressed test lengths in bits, for seven different competitive methods, are shown then. A comparison of only seven biggest ISCAS'89 circuits is shown, since no more relevant data was available to us. The last column shows the compressed test data size in bits for our proposed compression tool SAT-Compress. An all-zero initial test pattern for both COMPAS and SAT-Compress is used, thus the results are not influenced by different initial states.

It can be concluded from Table I., that our proposed algorithm can reach similar and often even better compression of test patterns than most of the presented state-of-the-art compression methods. The time comparison of the presented tools was not possible to measure because of the unavailability of source codes.

B. Comparison COMPAS and SAT-Compress Algorithms

In this Subsection we will present a more detailed comparison of SAT-Compress and COMPAS, as a representative of test compression algorithms based on overlapping of patterns.

Results for ISCAS'85 and '89 benchmarks [11, 12] are presented in Table II. The first column "*Bench*." presents the name of the benchmark. The compressed test lengths generated by COMPAS and SAT-Compress, both starting with an all-zero test pattern, are shown in the column "*bits*". Then we have tried to repeatedly run the algorithms starting with different initial test patterns. The average compressed test lengths are shown in the columns "*avg.*" and average variations of compressed test lengths are shown in the columns "*var.*". The time consumptions in seconds for the SAT-Compress algorithm are shown in the "time" column.

The results of this measurement show that the bitstream length for both tools significantly depends on the initial test pattern and starting with an all-zero seed (which is the default setting for COMPAS) can produce outstandingly poor results, out of the range of the variation (see, e.g., s298 for COMPAS). In some cases, our tool reaches much better compression than COMPAS, but it may also fail. We are still investigating why the efficiency of proposed algorithm is much better for ISCAS'85 benchmarks than for ISCAS'89 ones. As can be seen from Table II., the time consumption may be considerable for larger circuits, but we suppose, that scalability of proposed algorithm can be improved by using of more scan chains and division of the circuit into smaller parts.

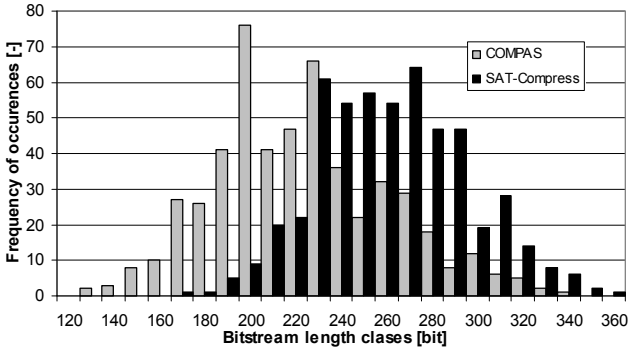

Figure 3. Frequency of bitstream length distribution (c432)

Fig. 3 shows an example of distribution of bitstream lengths using different starting patterns for COMPAS and SAT-Compress. In all measured cases we always start with a test pattern covering one particular fault. Thus, the number of restarts is equal to the number of faults.

As can be seen, the compressed bitstream length seems to have Gaussian-like distribution and the only difference

between SAT-Compress and COMPAS characteristics is their displacement. This Gaussian distribution of the compressed bitstream lengths for different starting seeds is a common characteristic of these two tools for all tested benchmarks. It indicates that the selection of the initial test pattern has a crucial impact on the resulting compressed test length, for both algorithms. The way of its proper choice will be a topic of our further investigation.

V. CONCLUSION

New test patterns compression algorithm (SAT-Compress) based on a modification of SAT-based ATPG is presented. This algorithm utilizes a CNF (Conjunctive Normal Form) implicit representation of test patterns and tries to compress the test patterns by overlapping. In contrast to competitive state-of-the-art test compression techniques, the proposed algorithm does not rely on a pre-generated test set; most suitable test patterns are being generated on the fly.

The test decompression is based on a generic RESPIN architecture, where the patterns are being decompressed in the tested circuit scan-chain. Thus, the circuit needs not be modified to apply the test.

The proposed method was compared with seven state-of-the-art test compression algorithms and a detailed comparison with COMPAS has been made. The comparison results seem to be promising – SAT-Compress achieved the best test compression ratio for many benchmark circuits. Moreover, there are yet many ways of possible improvements. These will be a topic of our further research. Possibilities of application of the method to multiple scan-chain designs will be also studied in the future.

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] T. Larrabee, “Test Pattern Generation Using Boolean Satisfiability,” IEEE Transactions on Computer-Aided Design, 1992, pp. 4-15.

[2] 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.

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

[4] C. Dufaza, H. Viallon, C. Chevalier, "BIST hardware generator for mixed test scheme," edtc, European Design and Test Conference (ED&TC '95), 1995, pp. 424-431.

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

[6] Daehn, W., Mucha, J,: Hardware Test Pattern Generation for Built-in Testing. Proc. of IEEE Test Conference, 1981, pp. 110-113.

[7] Schafer L., Dorsch R., Wunderlich H.J., “RESPIN++- Deterministic Embedded Test,“ Proc. of the European Test Workshop, 2002, pp.37-42.

[8] Jeníček J., Novák O.:A Test Pattern Compression Based on Pattern Overlapping, Proc. of DDECS 2007, Apr. 2007, Krakow, Poland, pp.29 - 34, ISBSN: 1-4244-1161-0.

[9] R. Dorsch and H.-J. Wunderlich, “Tailoring ATPG for embedded testing,” in Proc. ITC, 2001, pp. 530–537.

[10] Rajski, J., “Embedded Deterministic Test” IEEE Trans. on CAD, vol. 23, No. 5, 2004, pp. 776-792.

[11] 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, pp. 663-698, 1985.

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

[13] Janak H. Patel, Ilker Hamzaoglu, "Test Set Compaction Algorithms for Combinational Circuits," iccad, International Conference on Computer-Aided Design (ICCAD '98), 1998, pp.283-289.

[14] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, “Scan vector compression/decompression using statistical coding,” in Proc. VLSI Test Symp., 1999, pp. 114–120.

[15] C.V. Krishna, N.A. Touba, “Reducing Test Data Volume Using LFSR Reseeding with Seed Compression,” in Proc. of the International Test Conference, 2002, pp. 321-330.

[16] Irion, A.; Kiefer, G.; Vranken, H.; Wunderlich, H.-J., “CircuitPartitioning for Efficient Logic BIST Synthesis,” Proc. DATE, 2001, pp.88-93.

[17] Hamzaoglu, I., and J.H. Patel, “Reducing Test Application Time for Full Scan Embedded Cores,” Proc. Of Int. Symp. on Fault Tolerant Computing, 1999, pp. 260-267.

[18] A. Chandra, K. Chakrabarty, “Frequency-Directed Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data compression,” in Proc. of VLSI Test Symposium, 2001, pp. 42–47.

[19] A. Chandra and K. Chakrabarty, “Test Data Compression and Test Resource Partitioning for System-on-a-Chip Using Frequency-Directed Run-Length (FDR) Codes,” IEEE Transactions on Computers, vol. 52, May 2003, to appear.

TABLE I. COMPARISON OF THE TEST DATA AMOUNT FOR DIFFERENT COMPRESSION TECHNIQUES

Bench.	MinTest [13]	Stat. Coding [14]	LFSR Reseeding [15]	Illinois Scan [16, 17]	FDR Codes [18, 19]	EDT [10]	RESPIN++ [7]	COMPAS [3]	SAT-Compress
s5378	20,758	15,417	6,180	14,572	12,346	-	17,332	2,148	2,407
s9234	25,935	19,912	12,112	27,111	22,152	-	17,198	11,594	9,928
s13207	163,100	52,741	11,285	109,772	30,880	10,585	26,004	4,163	10,457
s15850	58,656	49,163	12,438	32,758	26,000	9,805	32,226	8,234	12,987
s35932	21,156	-	-	-	22,744	-	-	1,860	5,096
s38417	113,152	172,216	34,767	96,269	93,466	31,458	89,132	24,198	19,291
s38584	161,040	128,046	29,397	96,056	77,812	18,568	63,232	7,291	14,271