Pseudo-Random Pattern Generator Design for Column-Matching BIST

Petr Fišer
Czech Technical University
Dept. of Computer Science and Engineering
Karlovo nám. 13, 121 35, Prague 2
e-mail: fiserp@fel.cvut.cz

Abstract

This paper discusses possibilities for a choice of a pseudorandom pattern generator that is to be used in combination with the column-matching based built-in self-test design method. The pattern generator should be as small as possible, whereas patterns generated by it should guarantee satisfactory fault coverage. Weighted random pattern generators offer this. Several weighted pattern generator designs are proposed and their effectiveness is evaluated in this paper. Moreover, two methods for computing the weights are compared.

The column-matching method is primarily intended for a test-per-clock BIST, i.e., test patterns are applied to the tested circuit in parallel. Pseudorandom vectors obtained by an LFSR are modified here by a combinational circuit, to obtain deterministic test patterns. The number of inputs of this block corresponds to the width of the LFSR, the outputs correspond to the tested circuit inputs. This paper discusses possibilities of a reduction of the LFSR width.

1. Introduction

As the complexity of VLSI circuits constantly increases, there is a need of a built-in self-test (BIST) to be used. Built-in self-test enables the chip to test itself and to evaluate the circuit's response. Thus, the very complex and expensive external ATE (Automatic Test Equipment) may be completely omitted, or its complexity significantly reduced. Moreover, BIST enables an easy access to internal structures of the tested circuit, which are extremely hard to reach from outside.

There have been proposed many BIST equipment design methods [1]. In most of the state-of-the-art methods some kind of a pseudorandom pattern generator (PRPG) is used to produce vectors to test the circuit. These vectors are applied to the circuit either as they are, or the vectors are modified by some additional circuitry in order to obtain better fault coverage. Then the circuit's response to these vectors is evaluated in a response analyzer.

Usually, linear feedback shift registers (LFSRs) or cellular automata (CA) [2] are used as PRPGs, for their simplicity. Patterns generated by simple LFSRs or CA often do not provide a satisfactory fault coverage. Thus, these patterns have to be modified somehow. One of the most known approaches is the *weighted random pattern testing* [3, 4]. Here the LFSR code words are modified by a weighting logic to produce a test with given probabilities of occurrence of 0's and 1's at the particular circuit under test (CUT) inputs. Many papers dealing with the computation of the weights and the design of the weighting logic have been published [4-7]. There are two problems in the weighted testing involved: the way how to compute the weights and the way how to design the weighting logic. These two aspects are discussed in this paper, compared and their influence on the overall column-matching based BIST design is evaluated.

We propose several ways of a design of the pseudorandom pattern generator that is used in the BIST equipment produced by the column-matching BIST method [8, 9] in this paper. The main contribution reached in this research is a significant reduction of the width of the LFSR used, thus reduction of the overall test pattern generator (TPG) logic. Originally, the LFSR width was equal to the number to the CUT inputs. Now the LFSR width may be arbitrarily scaled. The effectiveness of this scaling and its possible extents are documented on BIST equipment design examples for some of the standard ISCAS benchmarks.

The paper is structured as follows: basic principles of the mixed-mode column-matching BIST design method are described in Section 2. Section 3 discusses the weighted pattern testing issues, namely the influence of the weight computation and selection of the number of weights on the fault coverage. Then, LFSR width reduction possibilities are discussed in Section 4. Experimental results summarizing the effects of the proposed mechanisms are shown in Section 5. Section 6 concludes the paper.

2. Mixed-Mode Column-Matching

In the column-matching BIST design method the test pattern generator (TPG) consists of two parts: the pseudorandom pattern generator (PRPG), which is usually an LFSR and the *output decoder*. The output decoder is a combinational block transforming pseudorandom vectors into deterministic test patterns pre-computed by an ATPG tool. The method is designed for a test-per-clock BIST, i.e., the test patterns are fed to the circuit in parallel. Thus, the output decoder has as many inputs, as there are the PRPG outputs (LFSR bits) and as many outputs as there are CUT inputs.

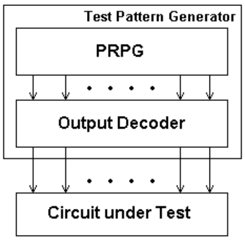

Figure 1. Test-per-clock BIST design

The decoder is constructed using the column-matching algorithm, proposed in [8]. The inputs of the decoder are the PRPG patterns, the outputs are deterministic test vectors. The algorithm is designed to test combinational circuits only, thus the order, in which are the test patterns applied to the circuit, is insignificant. Thus, the vectors may be reordered in any way, i.e., we can freely decide, which PRPG vector will be “decoded” to obtain a particular deterministic vector. The main principle of the algorithm consists in trying to “match” as many decoder outputs with its inputs, by finding a suitable vector ordering. If an output is matched with an input, there will be no logic needed to implement this output; it will be implemented as a mere wire. Finding these matches is a simple permutation problem. Let us have an n-bit PRPG and an m-output CUT. The decoder will be an n-input and m-output combinational block. There are n possibilities for a column match for each of the m outputs. Thus, there are n^m combinations to test, to obtain an optimum matching. Such an algorithm complexity is prohibitively large, thus some heuristic must be used instead of a brute force approach. We use a “*thorough search*” algorithm, having an asymptotic complexity $O(n \cdot m^2 \cdot p \cdot s^2)$, where p is the number of PRPG patterns and s the number of deterministic vectors. For more details see [8].

Then the algorithm has been extended to support a mixed-mode BIST [9]. Here the BIST is divided into two phases: the pseudorandom and deterministic one. The difference between our mixed-mode BIST method and the others is that the two phases are disjoint. First, the easy-to-detect faults are covered in the *pseudo-random* phase. Then, a set of deterministic test vectors covering the undetected faults is computed and these tests are then generated by a transformation of the subsequent PRPG patterns. This significantly reduces the decoder logic. A general scheme of the column-matching mixed-mode BIST is shown in Fig. 2. For sake of simplicity the number of LFSR bits (and thus the Decoder inputs) was set equal to the number of CUT inputs (m) here.

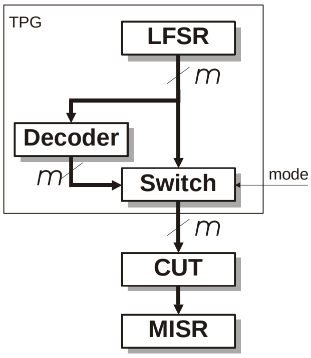

Figure 2. Column-matching BIST scheme

The whole mixed-mode column-matching based TPG design process can be summarized as follows:

1. Simulate several (*PR*) pseudo-random patterns for the CUT and determine the undetected faults (by fault simulation)
2. Compute deterministic test patterns detecting these faults by an ATPG tool
3. Perform the column-matching using the subsequent LFSR pseudo-random patterns and the deterministic tests
4. Synthesize the unmatched decoder outputs using a two-level Boolean minimizer.

3. Weighted Pattern Testing

There have been many weighted pattern testing approaches proposed up to now [4, 5, 6, 7, 10, 11, 12]. Basically, all of them are based on computing the probability of occurrence of ‘1’ and ‘0’ values on particular inputs and modifying the pseudorandom sequence (which usually has the probability 0.5 on all inputs) by some additional *weighting* logic, to meet these weights. In general, the weighting logic consists of AND and OR gates, which, when fed by LFSR outputs, produce weighted patterns. For example, when two LFSR outputs are connected into an AND gate, the resulting probability of occurrence of '1' at the output of the AND gate will be equal to 0.25.

It was shown that using simple weighted pattern testing only does not ensure sufficient fault coverage and multiple weight sets are needed [13]. Another alternative is to use a combination of weighted pattern testing with deterministic test, as shown, e.g., in [12]. We also propose such an approach too in this paper, particularly the combination of weighted pattern pseudorandom testing with column-matching.

3.1. Weight Set Computation

The weights are usually being computed from the deterministic test set derived for the tested circuit. A common approach is to find a set of so called *random pattern resistant faults (RPRFs),* which are faults that are difficult to be detected by random patterns. Then, a test vector set is computed to test these faults. The weights are then derived by computing respective 0/1 value ratios for each CUT input.

The RPRFs are determined by repeatedly applying pseudorandom vectors to the CUT and recording the undetected faults. The number of RPRFs obtained thus strongly depends on the number of pseudorandom patterns applied. The higher their number is, the less faults remains undetected. There arises a question what number of RPRFs should be considered in practice, to obtain optimum results. One limit approach is to consider *all* faults and to derive the weights from a complete test for the circuit. This approach is, unfortunately, unusable for very large circuits, since the complete test set computation would take a very long time. We have performed experiments to estimate what number of RPRFs should be used to compute the test weights. We have used the s9234.1 ISCAS'89 benchmark circuit [14] for the following measurement. We have varied the number of pseudorandom vectors applied to the CUT to detect RPRFs, from 0 (all faults are used to compute weights) to 100 000. Then a 3-weight set (see Subsection 3.2) was computed using the test vectors detecting these RPRFs. After that, the weighting logic was synthesized and the weighted PRPG was run for 10 000 cycles. The number of undetected faults was measured. Such an experiment was repeatedly run 10 000-times (using different LFSRs and keeping the computed weight set) and the average of the result computed (for the number of undetected faults), for higher precision of the measurement. The results are shown in Table 1. First the number of pseudorandom vectors used to determine RPRFs is shown, then the number of PRPFs obtained. The "*vcts*" column indicates the number of test vectors used for weight computation. Finally, the average number of faults undetected by the run of the resulting weighted pattern PRPG is shown ("*UD*"). The dependency of the number of undetected faults on the number of test vectors testing RPRFs is visualized by Fig. 3. There is an apparent global minimum to be seen, corresponding to the optimum number of test vectors to determine the test weights. However, it is hard to estimate this optimum in practice, for different circuits. It has to be found experimentally, by trying out different numbers of vectors that are applied to the circuit and picking the best trial.

Table 1. Computing test weights

vectors	*RPRFs*	*vcts.*	*UD*
0	6927	6470	852
100	3060	2605	817
1000	1997	1544	815
2000	1709	1257	805
2500	1614	1162	802
3000	1482	1030	781
4000	1347	895	762
5000	1280	828	808
10 000	1080	628	818
20 000	950	498	860
50 000	876	424	862
100 000	715	263	891

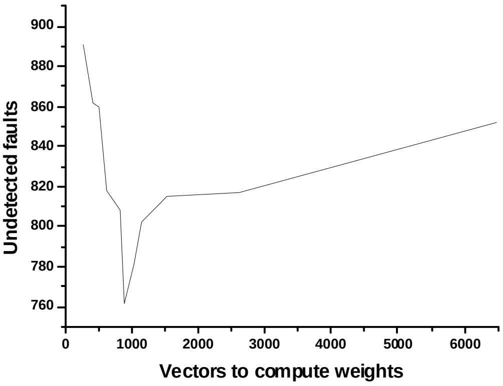

Figure 3. Number of undetected faults

3.2. The Weighting Logic

Another issue that has to be taken into account is the design of the weighting logic. The higher is the precision of weight generation, the higher the area overhead. For our examples we will limit ourselves to the 3-weight and 5-weight logic only, since more weights (or even more weight sets) would involve a large area overhead caused by the weighting logic, while the gain in quality (fault coverage) is negligible.

There have been several weighting logic designs proposed. The simplest one (see, e.g., [12]) proposes weights like 0, 0.5, 1 for 3-weighted logic, 0, 0.25, 0.5, 0.75, 1 for 5-weight logic, etc. This approach has been found very inefficient in terms of the fault coverage reached, see Fig. 4. This is most probably due to constant 0 and 1 values in the test. However, such a 3-weight test does not require any additional weighting hardware, since the 0 and 1 weights are implemented as hard-wired connections to the ground or the voltage supply, respectively, and the 0.5 weight is constructed as a direct connection to an LFSR output. We propose a 3-weight testing method where weights of 0.25, 0.5 and 0.75. Thus, the two weights are constructed by AND-ing and OR-ing two LFSR inputs and a 5-weight method (0.125, 0.25, 0.5, 0.625, 0.7) where the weights are generated by AND-ing and OR-ing two or three LFSR inputs. This approach involves some additional hardware, however the increase of the fault coverage reached by it is fully compensated by the reduction of the deterministic test generator logic.

The effect of the use of the weighting logic on the fault coverage is illustrated by Fig. 4. We have computed the weights for the ISCAS'89 s13207.1 circuit [14], repeatedly reseeded the LFSR and constructed the weighting logic (10 000-times). The number of faults that remained undetected by a sequence of 5000 vectors generated by the final TPG was measured. The curves represent the frequencies of the respective amounts of undetected faults (the area below the curves is equal to the number of tests, i.e., 10 000). Four curves are shown, representing four different weighting logics. The most efficient one is the 5-weight logic, without 0 and 1 weights used (the leftmost curve). Here minimum of vectors are left undetected. Similar 3-weight logic is illustrated by the neighboring curve. The number of undetected faults is almost the same. Then the case with no weighting logic used is shown, for comparison. It can be seen that the weighting logic significantly reduces the number of undetected faults, with respect to this case. The rightmost curve describes the 5-weight logic, where constant 0 and 1 weights are used. Here the number of undetected faults is even increased. The 3-weight logic with 0 and 1 constants is not shown in the graph, since the number of undetected faults is extremely high.

The results are summarized in Table 2. Minimum and average numbers of undetected faults are shown there, for all the five weighting logic cases. The s13207.1 circuit has 9815 faults in total.

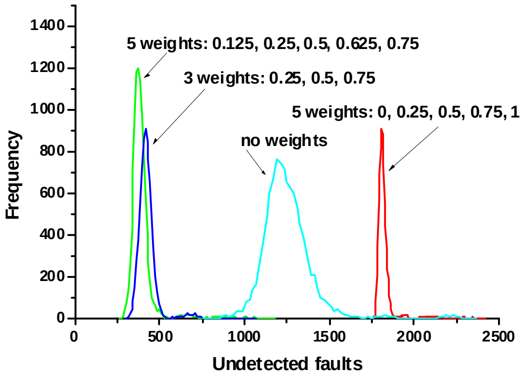

Figure 4. Weighting logic effects

Table 2. Weighting logic effects

Weighting logic	Min.	Avg.
5-weights (0.125, 0.25, 0.5, 0.625, 0.75)	269	392
3-weights (0.25, 0.5, 0.75)	299	425
no weights	742	1250
5-weights (0, 0.25, 0.5, 0.75, 1)	1761	1823
3-weights (0, 0.5, 1)	3762	3771

4. Scaling the LFSR Width

The column-matching algorithm is primarily designed for a test-per-clock BIST (as it was said before). Originally, the number of the LFSR bits had to be equal to the number of CUT inputs. Results presented here show that the LFSR width may be arbitrarily scaled. The effect of such a scaling will be shown in this Section. The LFSR scaling affects both the total TPG area overhead and the TPG design time. This is documented in the summary results presented in Section 5.

4.1. Weighted Pattern Testing

When the weighted pattern testing is used, a new block is introduced into the BIST design – the *weighting logic* block. See Fig. 5. The LFSR width (*r*) may be less than the number of CUT inputs (*m*), the number of TPG outputs is increased just by the weighting logic block.

In practice, an LFSR with a random polynomial and random seed is used. This ensures (to some extent) a uniform distribution of '1's and '0's in code words produced by it. In other words, the weights of all the *r* LFSR outputs are approximately 0.5. To generate the weighted patterns, the outputs of the weighting logic are generated by AND-ing or OR-ing randomly selected LFSR outputs. Thus, m weighted PRPG outputs are generated by this way.

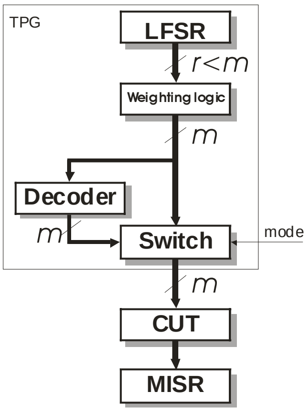

Figure 5. Weighted column-matching BIST scheme

4.2. Non-Weighted Testing

Another possibility how to reduce the LFSR width is to use a *splitter*. No weighting logic is involved here, however we need to synthesize an m-output pseudorandom pattern generator, to feed the CUT in the pseudorandom phase. The simplest way how to do this is to split the LFSR outputs among the PRPG outputs, thus feeding several CUT inputs by one LFSR output. This, of course, reduces the pseudorandom testing capabilities. On the other hand, a significant area overhead reduction is obtained by eliminating the weighting logic (and, more importantly, the number of LFSR flip-flops).

The LFSR width reduction has a significant influence on the deterministic phase as well: since there are less decoder inputs, the column-matching process is sped up (linearly with the number of inputs, see Section 2). The negative influence of a reduction of the LFSR input is an increase of the area overhead of the decoder. This is caused by a decrease of the number of possibilities for a column match (there are n^m column matches in total). However, this disadvantage is fully compensated by the LFSR area reduction.

The structure of the BIST using a splitter is shown in Fig. 6.

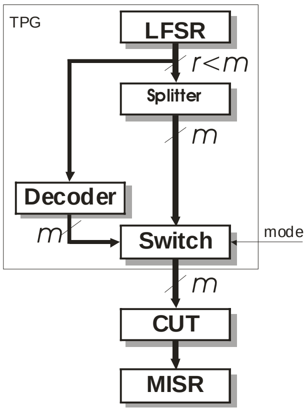

Figure 6. Column-matching BIST scheme with a splitter

5. Experimental Results

Experimental results for several standard ISCAS benchmarks are presented in this section. The experiments are summarized in one table, for better comparison. See Table 3, at the end of the paper. First, the benchmark circuit name is stated, then the number of its inputs ("*m*"). The "*LFSR*" column indicates the width of the LFSR used (r). Then the length of the pseudorandom phase follows ("*PR*"), the length of the deterministic phase was set to 1000 in all cases. A short description of the PRPG design method follows (type of weighting logic or a splitter). In a case of weighted pattern testing the number of gates needed for the weighting logic is shown in the next column ("*w*"). The number of faults that remained undetected by the pseudorandom phase is shown in the "*UD*" column. Then the number of deterministic vectors for which the output decoder had to be synthesized is indicated in the "*vcts*" column. Finally the column-matching results are shown. The number of matches obtained is shown in the "*M*" column, the area overhead of the Decoder and Switch is shown next ("*CM*"), then the column-matching process runtime in seconds and, at the end, the overall area overhead of the whole TPG, including the weighting logic and an LFSR area overhead ("*GEs*"). The area overhead is computed in terms of gate equivalents [15]. An n-input NAND gate counts for 0.5n GEs, the two-input XOR gate (used in LFSR) is 2.5 GEs. The size of a D flip-flop is considered to be 4 GEs, which complies with the design of a standard flip-flop in CMOS logic.

The pseudo-random patterns were simulated using HOPE fault simulator [16]. Test sets for the undetected faults were computed by Atalanta ATPG [17].

5.1. Comments on the Results

The first three lines of the table describe the results for the c2670 ISCAS benchmark, for different weighting logics. First, no weighting logic was used, and then the 3-weight and 5-weight logic was applied. The weighting logic without constant 0 and 1 values was used (see Subsection 3.2). There is an apparent reduction of the number of undetected faults to be seen when the weighting logic is used. Unfortunately, the number of deterministic vectors needed to cover the undetected faults is in the 3-weight case higher than in the unweighted case. This caused an increase of both the column-matching algorithm runtime and decoder area overhead. When the 5-weight logic is used, the number of deterministic test vectors is significantly less, thus the algorithm runtime is significantly reduced. The overall overhead is unfortunately increased, due to the weighting logic overhead.

An LFSR having the width equal to the number of CUT inputs was used in the previous example. Only a 50-bit LFSR is used in the next comparison example. Here the effect of the weighting logic is to be seen more apparently. Both the runtime and overall TPG logic is reduced when the weighting logic is used. Moreover, there is a very significant reduction of the TPG area overhead, when compared to the 233-bit LFSR case (41% in the unweighted case, 55% in the 3-weight case).

Similar weighting logic effect can be observed for the s838 benchmark circuit.

More thorough experiments have been performed using the s13207.1 benchmark circuit. First of all, the effect of the use of the weighing logic is shown. Very significant reduction of the number of deterministic test vectors is seen when the 3-weight logic is used. This yielded a column-matching runtime reduction (more than 19-times) and the decoder area reduction (more than 4-times). Unfortunately, the overall TPG logic is slightly increased, due to the weighting logic overhead.

Next, the LFSR scaling effects were observed, while the 3-weight logic is used. We have tried to scale the originally 700-bit LFSR down to 20 bits. It can be seen that optimum results are obtained for the 50-bit LFSR, the TPG area reduction is more than 70% with respect to the original (700-bit unweighted) case. When the LFSR width is scaled down more, the TPG area overhead rapidly increases, since the weighted PRPG is not able to cover enough faults, due to a reduced randomness of the patterns. The column-matching results for the 30- and 20-bit LFSR are not present, due to very high column-matching algorithm runtimes.

6. Conclusions

An analysis of several possible pseudo-random pattern generators used in connection with the column-matching BIST TPG design method is presented in this paper. The aim is to reduce both the test pattern generator area overhead and its design time. Formerly, the LFSR width used in the column-matching BIST design had to be equal to the number of the tested circuit inputs. Two methods reducing the LFSR width are proposed here.

Two approaches to do this are shown: the use of the weighted pattern testing and using a “splitter”. Several weight set generation methods are discussed and experimentally evaluated. The weighted pattern testing enables a reduction of the width of the LFSR used in our TPG design method, for a cost of the weighting logic overhead. Satisfactory trade-off between these two aspects has to be found here.

The second way to reduce the overall TPG logic proposed in this paper is the use of the “splitter”. The LFSR width is reduced here as well, however no additional weighting logic is involved here; the LFSR outputs are simply branched to feed several CUT inputs simultaneously. This reduces the pseudo-random fault coverage capabilities of the PRPG, which causes an increase of the amount of the combinational logic produced by column-matching. However, is a proper LFSR width scaling is found, this overhead increase is negligible when compared to the LFSR area reduction obtained. Moreover, the column-matching runtime is reduced as well.

Acknowledgement

This research has been supported by MSMT under research program MSM6840770014.

References

[1] V.K. Agrawal, C.R. Kime and K.K., Saluja, „A tutorial on BIST, part 1: Principles“, IEEE Design & Test of Computers, vol. 10, No.1 March 1993, pp.73-83

[2] P.P. Chaudhuri, et al., „Additive Cellular Automata Theory and Applications, Volume I“, IEEE Computer Society Press, 1997, 340 pp.

[3] P.H. Bardell, W.H. McAnney and J. Savir, „Built-In Test for VLSI: Pseudo-Random Techniques“, New York: John Wiley & Sons, 1987

[4] H.J. Wunderlich, „Self Test Using Unequiprobable Random Patterns“, International Symposium on Fault-Tolerant Computing, 1987

[5] F. Muradali, V.K. Agarwal and B. Nadeau-Dostie, “A New Procedure for Weighted Random Built-In-Self-Test,” Proc. International Test Conference (ITC‘90), pp. 660-668, 1990

[6] M.A. Miranda et al., “Generation of Optimized Single Distributions of Weights for Random BIST”, Proc. International Test Conf. (ITC’93), pp. 1023- 1030, 1993

[7] J. Hartmann and G. Kemnitz, “How to Do Weighted Random Testing for BIST”, Proc. International Konference on Computer-Aided Design (ICCAD), 1993

[8] Fišer, P., Hlavička, J., Kubátová, H.: „Column-Matching BIST Exploiting Test Don't-Cares“, Proc. 8th IEEE Europian Test Workshop, Maastricht, 2003, pp. 215-216

[9] Fišer, P., Kubátová, H.: „An Efficient Mixed-Mode BIST Technique“, Proc. 7th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop 2004, Tatranská Lomnica, SK, 18.-21.4.2004, pp. 227-230

[10] Kunzmann, A.: „Efficient random testing with global weights“. In Proceedings of the Conference on European Design Automation (Geneva, Switzerland). Computer Society Press, Los Alamitos, CA, 1996, pp. 227-232.

[11] Jas, A., Krishna, C. V. and Touba, N. A. 2001. Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme. In Proceedings of the 19th IEEE VLSI Test Symposium (March 29 - April 03, 2001). VTS. IEEE Computer Society, Washington, DC, 2.

[12] Wang, S. 2001. Low hardware overhead scan based 3-weight weighted random BIST. In Proceedings of the IEEE international Test Conference 2001 (October 30 - November 01, 2001). IEEE Computer Society, Washington, DC, 868-877.

[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] 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

[15] G. De Micheli, “Synthesis and Optimization of Digital Circuits”, McGraw-Hill, 1994

[16] H.K. Lee and D.S. Ha. An Efficient Forward Fault Simulation Algorithm Based on the Paralel Pattern Single Fault Propagation, Proc. of the 1991 International Test Conference, pp. 946-955, Oct. 1991

[17] 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, 199

[18] N.A. Touba and E.J. McCluskey: “Bit-Fixing in Pseudorandom Sequences for Scan BIST“, IEEE Transactions on CAD, Vol. 20, No. 4, April 2001, pp. 545-555

[19] E.M. Sentovich et al.: „SIS: A System for Sequential Circuit Synthesis“, Electronics Research Laboratory Memorandum No. UCB/ERL M92/41, University of California, Berkeley, CA 94720, 1992

Table 3. Experimental results

bench	*m*	*LFSR (r)*	*PR*	*method*	*w*	*UD*	*vcts*	*M*	*CM*	*time [s]*	*GEs*
c2670	233	233	4000	standard	-	320	104	197	219	606	1153.5
		233		3-w	216	282	130	202	264.5	960	1415
		233		5-w	406	159	72	224	226	171	1566.5
		50		splitter	-	334	120	193	323	475	677.5
		50		3-w	219	275	144	203	248.5	427	629.5
s838	67	67	5000	standard	-	101	61	43	107	50	377.5
		67		3-w	42	96	103	56	94.5	128	365
		67		5-w	87	74	85	61	86.5	60	357
s9234.1	247	247	50000	standard	-	783	331	211	514	1030	1504.5
		247		3-w	135	532	73	232	165.5	532	1291
		50		splitter	-	1226	526	188	1005.5	7343	1208
		50		3-w	135	713	185	223	333	463	535.5
s13207.1	700	700	10000	standard	-	917	492	696	172.5	4720	2975
		700		3-w	518	239	88	700	40.5	245	3361
		200		3-w	365	240	108	697	63.5	653	1231
		50		3-w	365	303	226	697	103.5	2835	671
		45		3-w	365	334	232	696	145.5	3423	693
		40		3-w	365	338	530	692	640.5	29434	1288
		30		3-w	365	1069	851	-	-	-	-
		20		3-w	365	1085	886	-	-	-	-