Improvement of the Fault Coverage of the Pseudo-Random Phase in Column-Matching BIST

Petr Fišer, Hana Kubátová
Department of Computer Science and Engineering
Czech Technical University
e-mail: fiserp@fel.cvut.cz, kubatova@fel.cvut.cz

Abstract

Several methods improving the fault coverage in mixed-mode BIST are presented in this paper. The test is divided into two phases: the pseudo-random and deterministic. Maximum of faults should be detected by the pseudo-random phase, to reduce the number of faults to be covered in the deterministic one. We study the properties of different pseudo-random pattern generators. Their successfulness in fault covering strictly depends on the tested circuit. We examine properties of LFSRs and cellular automata. Four methods enhancing the pseudo-random fault coverage have been proposed. Then we propose a universal method to efficiently compute test weights.

The observations are documented on some of the standard ISCAS benchmarks and the final BIST circuitry is synthesized using the Column-Matching method.

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) can 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.

Many BIST methods have been proposed [1], all of them have been finding a trade-off between the BIST area overhead, fault coverage achieved and the test time. Generally, some kind of a pseudo-random pattern generator (PRPG) is used to produce test patterns. These are being applied to the tested circuit and the responses are then evaluated by a signature analyzer. Usually, a linear feedback shift registers (LFSRs) or cellular automata (CA) [2] are used as PRPGs, for their simplicity. However, 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 has been published [4-7]. It has been found that multiple set of weights have to be used to achieve satisfactory results [8]. The weighted random pattern testing is aimed to increase the probability that vectors testing the hard-to-detect faults will be generated. However, there is no guarantee that the hard-to-detect faults will be really detected.

Other methods ensure 100% fault coverage by modifying the pseudo-random LFSR code words, in order to produce deterministic test vectors. Such an approach is exploited in the bit-fixing method [9, 10], bit-flipping [11] and the method proposed in [12, 13]. A similar principle is exploited in our column-matching method [14, 15] as well. In all these methods only some of the PRPG vectors are being transformed into deterministic patterns. Preferably, only vectors that do not detect any faults are being modified. Such an approach, where the pseudo-random vectors are mixed together with the deterministic ones, is called a *mixed-mode BIST*.

The difference between our mixed-mode BIST method – the column-matching - and the others is that the whole test is divided into two disjoint phases. 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 produced by a transformation of the following PRPG patterns, done by the Decoder. A general scheme of the column-matching mixed-mode BIST is shown in Fig. 1.

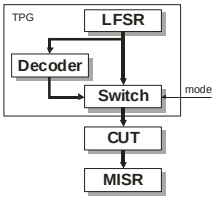


Figure 1. Column-matching BIST scheme

The more faults are covered in the pseudo-random phase, the smaller the resulting BIST logic overhead is, see [16]. Thus, our aim is to increase the pseudo-random fault coverage to a maximum. There are several techniques improving the fault coverage. However, a universal and simple technique cannot be found yet. The effects of the individual techniques significantly differ for different circuits. This fact is documented in this paper. We have examined four methods increasing the number of faults covered in the pseudo-random phase and compared the results. We propose an *enhanced column-matching* scheme. Here the column-matching method proposed before [14, 15] is augmented by methods increasing the number of faults detected in the pseudo-random phase, which reduces both the test time and BIST area overhead. The results obtained by the best methods are presented and compared with other state-of-the-art methods.

The proposed techniques are based on a random selection of the PRPG parameters. Such a fast approach, when applied many times repeatedly, could mostly outperform time-demanding deterministic techniques based on genetic algorithms [23] or a repeated simulation of all the test patterns [24].

This paper is primarily intended as a case study of a possibility of a combination of standard BIST methods involving a modification of a PRPG and a deterministic BIST. Moreover, in the beginning of the paper we introduce a straightforward overview of the fault coverage capabilities of different PRPGs.

The paper is organized as follows: first the pseudo-random fault coverage properties and weight distributions for standard LFSRs and CA will be discussed in Section 2. Principles of the column-matching method will be briefly described in Section 3 and techniques enhancing the fault coverage will be described and evaluated in Section 4. Section 5 shows a comparison of the enhanced column-matching method with other state-of-the-art methods. Section 6 concludes the paper.

2. Pseudo-Random Fault Coverage and Weight Distributions

A certain number of pseudo-random patterns are being applied to the CUT in the first phase of our column-matching mixed-mode BIST method, to detect the easily testable faults. The more faults are detected by these patterns, the simpler the logic needed to produce deterministic test vectors is. Thus, a good PRPG choice is of a key importance for the whole BIST design process.

We have studied several different PRPGs, namely LFSRs with different generating polynomials and seeds and cellular automata with different seeds. The fault coverage and the distribution of weights, thus the probabilities of occurrence of 1's and 0's in the generated code words will be evaluated in this section.

In all the following experiments we have used a cellular automaton based on a rule 60 for each cell [2], due to its simplicity. However, the results can be generalized for most of other automata. The structure of this CA is shown in Fig. 2.

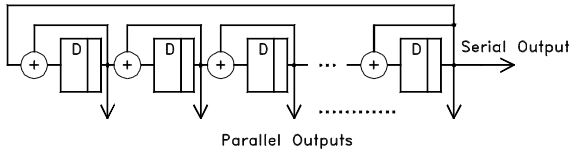


Figure 2. Cellular automaton used

2.1. Influence of the LFSR Generating Polynomial on Fault Coverage

In order to thoroughly evaluate the fault coverage of different PRPGs, we have made a vast number of experiments. The first group of experiments searches for the optimal number of LFSR "taps" (i.e., the number of XOR gates). In other words, we ask a question: "Is it necessary to use a primitive polynomial to generate the pseudo-random test patterns?" We have performed the following experiment: we have repeatedly applied 500 pseudo-random patterns to the c3540 ISCAS benchmark circuit [17]. The sets of test patterns were generated by LFSRs with different generating polynomials and seeds, both randomly generated. All the polynomials were chosen such that a satisfactory period was ensured. The LFSR width was set equal to the number of the CUT primary inputs, thus 50 in a case of the c3540 benchmark. We have gradually increased the number of LFSR taps, from 1 to 49. For each LFSR size 100 different random LFSRs were produced, differing both in the tap positions and the seed. Thus, for the circuit used, 5000 different LFSRs were produced. The results of the experiment are shown in Fig. 3. The horizontal axis corresponds to the number of LFSR taps, the vertical axis shows the fault coverage reached (number of undetected faults). In each horizontal position 100 points showing the fault coverage reached are drawn.

It can be observed that the number of taps does not influence the fault coverage capability at all; the fault coverage is steadily distributed. Thus, we can conclude that the most advantageous LFSR is one from the 1-tap LFSRs, since its area overhead is the smallest one. In most cases a 1-tap LFSR having a satisfactory period can be found. Using primitive polynomials thus becomes counterproductive, since the number of their taps is mostly bigger than one and they do not bring any contribution, considering that each LFSR tap introduces one XOR gate to the overall BIST overhead.

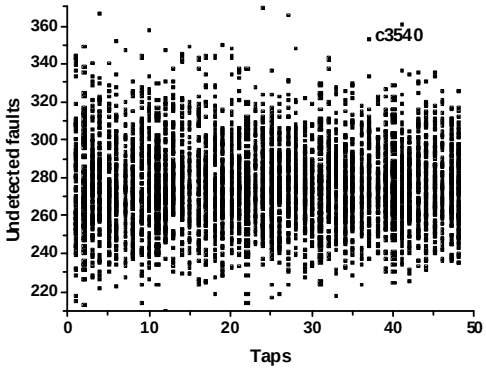


Figure 3. Fault coverage of different LFSRs

2.2. Fault Coverage Probability

We have studied the measure of a probability of detecting a given number of faults by a randomly generated LFSR, with respect to the number of pseudo-random test patterns. We have performed several sets of experiments, each experiment with a different, randomly generated LFSR (both in the polynomial and seed). The results are shown in Fig. 4. Here sets of 10, 50, 100, 500, 1000 and 5000 LFSR patterns were gradually applied to the c3540 circuit [17], 10 000 samples for each test size. The distribution of the number of faults which remained undetected is shown here. For a low number of patterns many faults are left undetected, while also their number varies a lot. When increasing the number of the test patterns, the number of undetected faults rapidly decreases, while the standard deviation of this number decreases as well. It can be derived from this example, that a good choice of a PRPG becomes very important when the number of the applied patterns is low (right side of the figure). For a relatively high number of pseudo-random patterns applied, the influence of their (non-)randomness is suppressed (left side of the figure).

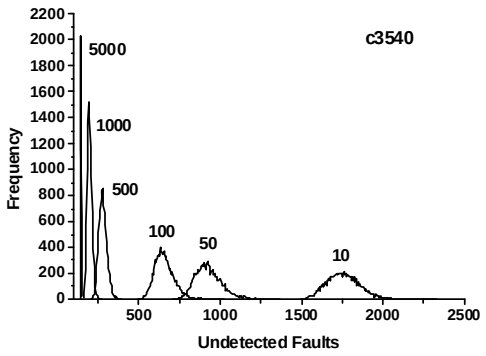


Figure 4. Pseudo-random fault coverage

2.3. Cellular Automata vs. LFSRs – Fault Coverage

The fault covering capabilities of cellular automata and LFSRs are compared in this Subsection. The distribution of the number of undetected faults, exactly as in Fig. 4, has been studied for LFSRs and CA, rule 60. Such a CA has been chosen for all our experiments that can be easily implemented by T-flip-flops. The distribution curves for these two types of generators are shown in Fig. 5. Here 500 patterns generated by random LFSRs seeded with random seeds were repeatedly applied to the c3540 circuit (10 000 times). It can be observed that the mean value of the number of undetected faults does not change when a CA is used, but the standard deviation *is decreased*. Hence an important conclusion can be derived: using a randomly seeded CA does not increase the number of covered faults in average, but the probability of detecting more faults by the CA vectors is higher with respect to an LFSR.

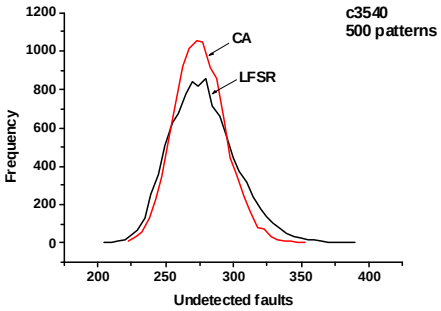


Figure 5. Comparison of the fault coverage obtained by LFSR and CA

2.4. Cellular Automata with Unbalanced Seeds

The previous experiment has shown that using a randomly seeded CA instead LFSRs does not bring any contribution to the fault coverage reached. In all of these experiments the seeds were generated randomly, with a steady distribution of 1's and 0's. On the other hand, when a "special" seed is selected for a cellular automaton, its fault covering properties will dramatically change.

We have performed an experiment similar to the one presented in Fig. 5, for the s838 ISCAS circuit [18] applying LFSR and CA patterns, once with a steady distribution of values in its seed, and once with a seed having *only one* "1" value at a random position, thus the weight of this seed was unbalanced. The four tests were run for 500 cycles and the distribution of the number of undetected faults was measured. The results are shown in Fig. 6. We can observe that for this special seed the fault coverage of a LFSR has rapidly decreased, but on the other hand, the variability of fault coverage of a CA has increased, while in some cases much more faults were covered by vectors produced by this CA (left-hand side).

This observation can be explained by an unsteady distribution of weights, as it will be shown in the following Subsection. These properties of cellular automata have been presented in other papers as well [19, 20].

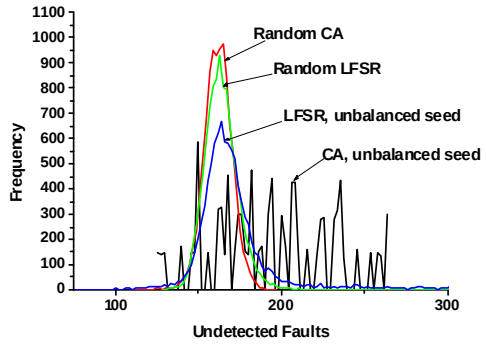
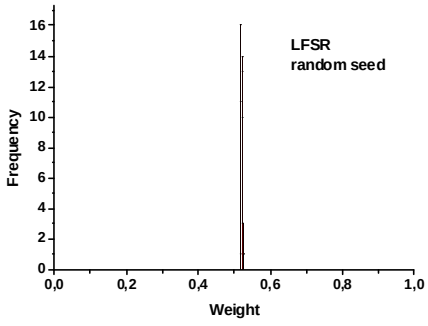


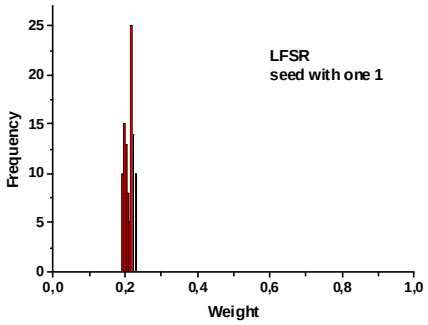
Figure 6. Fault coverage of a "specially" seeded CA and LFSR

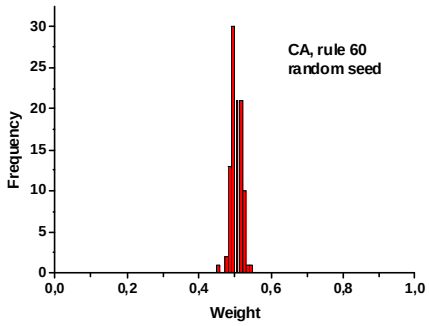
2.5. Distribution of Weights

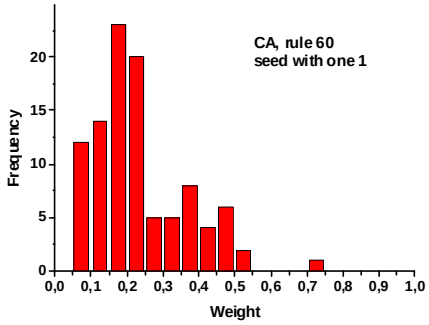
When CA are seeded properly, the distribution of weights on their outputs is non-uniform; it often varies throughout the whole weight scale. This cannot be observed by LFSRs. Thus, cellular automata can be used for weighted pattern BIST, without using any additional weighting logic.

The distribution of weights for four 100-bit PRPGs running 1000 cycles is shown in Figures 7-10.









Figures 7-10. Distribution of weights for different PRPGs

We can see that for a randomly generated seed, for both the LFSR and CA, the weights near 0.5, thus there is a balanced distribution of zeroes and ones in a test (Fig. 7 and 9). When a LFSR is unbalanced by a seed having only one "1" value and the rest are zeroes, the weights at the outputs are shifted to the weight of the seed in this particular case (Fig. 8). The weights do not differ from each other too much; the probabilities of zeroes and ones at all the outputs are approximately equal. A 1-tap LFSR has been chosen here, as we did for all of our experiments. If a LFSR with greater number of taps were chosen, all weights would near to 0.5, similarly to the case of a balanced seed. In Figure 10 the CA seeded with an unbalanced seed (having one "1" value) is shown. The weights vary from negligible values (all zeroes) to more than 0.7. This is the case where the weighted pattern testing can be advantageously applied.

3. Column-Matching BIST

The column-matching BIST method is based on a transformation of the PRPG code words into deterministic test patterns pre-computed by some ATPG tool. This transformation is being done by a combinational logic.

The method is designed for combinational or full-scan circuits, thus the order of the patterns applied to the tested circuit is not significant. In our column-matching method

we try to assign the PRPG patterns to the deterministic vectors, so that some of the columns of the PRPG patterns and ATPG tests are equal. Then the decoding logic needed to implement the "matched" column would be reduced to a mere wire connecting the decoder output with its respective input. The unmatched outputs have to be synthesized by some Boolean minimizer. For more detailed description see [14]. The column-matching was originally developed for a test-per-clock BIST, however it can be easily modified for the test-per-scan BIST, as it is proposed in [13]. The original principle has been further extended to support the mixed-mode testing [15]. The BIST run is divided into two disjoint phases here. First, the circuit is tested using an unmodified sequence of LFSR code words, to detect the easy-to-detect faults. For the rest of the faults deterministic test patterns are computed by Atalanta ATPG tool [21]. These vectors are to be produced by the Decoder. There has to be some additional logic to switch between the two phases. It is implemented as an array of multiplexers, one for each CUT input, however we try to eliminate the switching logic as well, by introducing *direct matches* [15]. A simple example of a mixed-mode column-matching BIST principle is shown in Fig. 11. A 5-bit PRPG is run for 5 cycles first and the easily testable faults are detected. Then we run the fault simulation to find the undetected faults, for which the test vectors are generated by an ATPG. At the end the Decoder logic is synthesized for these tests and the subsequent PRPG patterns. The resulting circuitry is shown in Fig. 12.

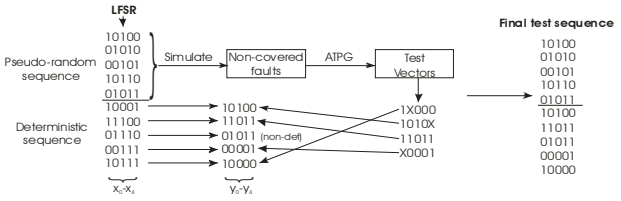


Figure 11. Mixed-mode column-matching BIST example

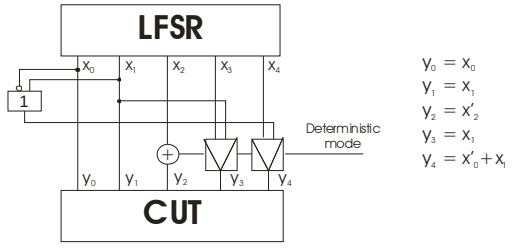


Figure 12. Resulting BIST circuit

4. Enhancing Pseudo-Random Fault Coverage

We have thoroughly examined several fault coverage improving methods and compared the results. We have found that none of these simple methods can be used universally. Their effectiveness strictly depends on the nature of the tested circuit.

Four methods will be proposed here: the *repetitive balanced LFSR reseeding*, *the repetitive unbalanced CA reseeding*, *test weight-based wire reordering* and a standard *weighted random pattern testing* [3].

4.1. Repetitive Balanced LFSR Reseeding

As we have shown in Fig. 4, the number of faults covered by a particular LFSR seeded by a random vector notably varies. It is computationally infeasible to compute an LFSR polynomial and seed so that the code words will cover a maximum of faults, hence we have chosen a randomized method. We randomly choose the LFSR polynomial (1-tap) and randomly generate the seed having a balanced number of zeroes and ones. Such an LFSR will then produce code words with balanced weights at all its outputs (see Subsection 2.5). Then we apply a certain number of vectors produced by this LFSR to the tested circuit [16] and determine the number of undetected faults, using a fault simulator. We repeat this procedure several times, while the "most successful" polynomial and seed are remembered and at the end they are used to generate the patterns. Using this simple method, we try to randomly choose an LFSR that lies on the left side of the fault coverage distribution curve shown in Fig. 4. The experimental results for some of the ISCAS benchmarks [17, 18] are shown in Table 1. The LFSR has been repeatedly reseeded 100 times. The number of repetitions required has been derived from our observation: we have found that the number of faults that remain undetected after 100 repetitions does not vary too much for our benchmark circuits. The number of undetected faults as a function of the number of repetitions is shown in Fig. 13. The c3540 benchmark has been tested for 3000 pseudo-random cycles here, up to 1000 repetitions. It can be seen that the number of undetected faults after 1000 iterations sinks by two only, with respect to the 100 repetitions.

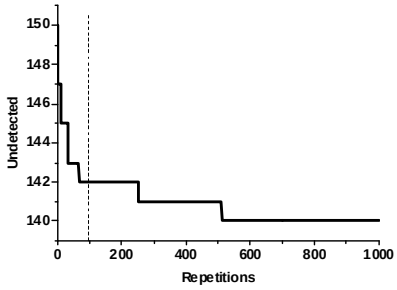


Figure 13. Improvement of fault coverage by repeated reseeding

In Table 1, the "*inps.*" column indicates the number of benchmark inputs, "*PR / det.*" the lengths of the pseudo-random and deterministic phases [16], the "*UD / vct.*" shows the number of undetected faults in the pseudo-random phase and the number of deterministic vectors produced by an ATPG, the "*M / DM*" column indicates the total number of column matches and the number of direct matches. The last column shows the total complexity of the resulting BIST, in terms of gate equivalents [15].

Table 1. Repetitive balanced LFSR reseeding

bench	*inps.*	*PR/det.*	*UD/vct*	*M/DM*	*GEs*
c880	60	500/500	12/5	60/50	15
c1355	41	1K/1K	11/1	41/28	19.5
c1908	33	2K/1K	21/9	32/24	14.5
c3540	50	3K/1K	139/2	50/48	3
s420	35	500/500	40/23	32/17	31
s641	54	1K/1K	14/11	54/39	22.5
s713	54	1K/1K	52/11	54/37	25.5
s838	67	5K/1K	104/61	43/13	119.5
s1196	32	5K/1K	23/16	30/24	15

4.2. Repetitive Unbalanced CA Reseeding

According to the experiments described in Subsection 2.4, a properly seeded cellular automaton could cover a significantly greater number of faults than a LFSR. We have found that seeding a CA with a balanced seed yields results very similar to those obtained by a LFSR, hence such a case will not be discussed here. On the other hand, a LFSR seeded with an unbalanced seed produces code words with very low (or high) weights on all its outputs. This is disadvantageous for most of circuits, which we have found experimentally. Thus, this case will not be studied either.

Similarly as in the previous subsection, we have repeatedly reseeded a cellular automaton (rule 60) 100 times and the best seed was recorded. In all cases a random seed having only one "1" value was selected. The results of the experiment are shown in Table 2. The table format is retained from Table 1.

Table 2. Repetitive unbalanced CA reseeding

bench	*inps.*	*PR/det.*	*UD/vct.*	*M/DM*	*GEs*
c880	60	the CA period is too short			
c1355	41	1000/1000	11/1	41/33	12
c1908	33	the CA period is too short			
c3540	50	3000/1000	148/7	50/43	10.5
s420	35	500/500	17/12	35/26	13.5
s641	54	1000/1000	53/22	51/32	42.5
s713	54	1000/1000	93/22	49/32	45.5
s838	67	5000/1000	24/17	67/52	22.5
s1196	32	the CA period is too short			

4.3. Determining Test Weights

The two previously described methods were based on a repeated random selection of a PRPG. The two following methods are based on deterministic decisions – we try to generate pseudo-random patterns, so that their weights will be in correlation with the required test weights.

At first, the weight set for the test has to be computed. Most of the present mixed-mode or weighted BIST methods target the *hard-to-detect* faults. Here either the deterministic test vectors detecting these faults [13] are generated or the pseudo-random patterns are modified by weights, so that the weights correspond to weights of the test set detecting these faults [3]. Usually, these hard-to-detect faults are identified first and then the test set is computed for them. Typically, several sets of pseudo-random patterns are applied to the CUT, the fault simulation is performed, and faults that were detected by each set of patterns are assigned as easy-to-detect, the rest are assigned as hard-to-detect faults [13].

We have applied a different approach to determine the test weights (i.e., the probabilities of occurrence of 1's and 0's on individual test bits). We do not compute the set for the hard-to-detect faults; the weights are derived directly from the test sets. Similarly to the previously mentioned method, we apply several sets of pseudo-random patterns to the CUT. For each pattern set we generate a set of test vectors covering the faults that remained undetected. For each test set, the weights are computed and after all the weights are averaged together, to obtain the final weight set. This method allows us to avoid fluctuations and weight differences between different test sets for the hard-to-detect faults. Moreover, weight sets generated by this weight do not reflect the hard-to-detect faults only; the testability of all faults is considered.

Usually, the weights of the individual bits do not differ between the test sets. We have found experimentally that the average value of the ranges of weight values (difference between the highest and lowest weight) does not exceed 0.2.

4.4. Test Weight-Based Wire Reordering

In the Repetitive Unbalanced CA Reseeding we have tried to randomly select the CA seed, and hoped that it will "fit" the weights. There is a more sophisticated approach to do so – the Test Weight-Based Wire Reordering. First, we compute the test weights using the procedure described in the previous Subsection. Then we randomly generate a seed for a cellular automaton. The seed should be selected so that the CA weights are unbalanced, thus e.g., the seed having only one "1" value. Then we compute the weights of the PRPG patterns and *reorder* the CA outputs, so that the CA weights correlate with the test weights. This is done by sorting both the CA outputs and test bits according their weights. Then they are assigned to each other according their order. This approach is in many cases better than a simple random CA reseeding, however there

are cases where it fails as well. The experimental results are shown in Table 3.

Table 3. Test weight-based wire reordering

bench	*inps.*	*PR/det.*	*UD/vct.*	*M/DM*	*GEs*
c880	60	the CA period is too short			
c1355	41	1000/1000	11/1	41/35	9
c1908	33	the CA period is too short			
c3540	50	3000/1000	150/5	50/44	9
s420	35	500/500	32/15	35/27	12
s641	54	1000/1000	23/15	43/42	18
s713	54	1000/1000	74/20	52/39	25.5
s838	67	5000/1000	13/12	67/52	22.5
s1196	32	the CA period is too short			

4.5. Weighted Random Pattern Testing

We have selected a common weighted random pattern testing method as the last one to try. Instead of reordering the wires, we use an additional logic to generate weighted outputs of the PRPG. A standard balanced LFSR can be advantageously used here, since all its weights are approximately 0.5, thus a computation of additional weights becomes simple. Using an AND gate connecting two different LFSR outputs, we obtain an output having a weight 0.25. Similarly, OR-ing two outputs we get a weight 0.75. The weights 0 and 1 can be acquired by connecting the CUT input to the ground or the power.

In all our experiments we have used only this set of weights (0, 0.25, 0.5, 0.75, 1). The results are shown in Table 4.

Table 4. Weighted random pattern testing

bench	*inps.*	*PR/det.*	*UD/vct.*	*M/DM*	*GEs*
c880	60	500/500	20/6	60/46	21
c1355	41	1000/1000	11/1	41/28	19.5
c1908	33	2000/1000	45/28	27/11	52
c3540	50	3000/1000	141/4	50/46	6
s420	35	500/500	18/7	35/25	15
s641	54	1000/1000	12/5	54/52	3
s713	54	1000/1000	50/6	54/51	4.5
s838	67	5000/1000	38/20	60/39	49
s1196	32	5000/1000	208/71	27/21	73

4.6. Comparison of the Results

A comparison of the four methods is presented in this section, see Table 5. The *a)* column stands for the Repetitive Balanced LFSR Reseeding, *b)* for the Repetitive Unbalanced CA Reseeding, *c)* for Test the Weight-Based Wire Reordering and *d)* for the Weighted Random Pattern Testing. The entries in the cells correspond to the complexity of the BIST, in terms of GEs. The respective field is shadowed where the method gave best results. It can be seen that no single method can be universally used; one has to be chosen according the properties of the tested circuit. Unfortunately, we have no clue which method to chose, other than trying out all of them and pick the best one. Finding out some selection rules will be the aim of our further research.

Table 5. Comparison of the results

bench	*a)*	*b)*	*c)*	*d)*
c880	15	-	-	21
c1355	19.5	12	9	19.5
c1908	14.5	-	-	52
c3540	3	10.5	9	6
s420	31	13.5	12	15
s641	22.5	42.5	18	3
s713	25.5	45.5	25.5	4.5
s838	119.5	22.5	22.5	49
s1196	15	-	-	73

5. Comparison with Other Methods

We have made a comparison of our enhanced column-matching mixed-mode method with other stand-or-the-art methods, namely the bit-fixing [9] and the method proposed in [13]. The experimental results are shown in Table 6. The "*TL*" column gives the length of the test (clock cycles), the "*GEs*" column the complexity of the resulting BIST logic, in terms of gate equivalents [22]. The "*method*" column indicates the enhancement method used, similarly as in the previous Subsection.

The empty cells indicate that the data for the respective circuit was not available to us.

6. Conclusions

We have proposed an enhancement of our column-matching mixed-mode method. The pseudo-random phase has been altered so that more faults are covered by it. Four methods enhancing the pseudo-random fault coverage have been proposed and their results evaluated. We have found that no universal method can be used, unless the area overhead caused by it is prohibitive.

The study of the PRPG type effect on the fault coverage has been given, showing that using a one-tap LFSR is sufficient to reach a satisfactory fault coverage.

We have proposed a new method to compute the weight set. Not only the hard-to-detect faults are being targeted here, the obtained weight set reflect the testability of all the faults.

Acknowledgement

This research was supported by grant GA 102/04/2137 and 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] H. Wunderlich, “Multiple Distributions of Biased Random Test Patterns”, IEEE Trans. Computer-Aided Design, vol. 9, no. 6, June 1990
[9] N.A. Touba, „Synthesis of mapping logic for generating transformed pseudo-random patterns for BIST“, Proc. of International Test Conference, pp. 674-682, 1995
[10] N.A. Touba and E.J. McCluskey, “Bit-Fixing in Pseudorandom Sequences for Scan BIST”, IEEE Trans. Computer-Aided Design, vol. 20, no. 4, pp 545-555, 2001
[11] H.J. Wunderlich and G. Keifer, „Bit-Flipping BIST“, Proc. ACM/IEEE International Conference on CAD-96, San Jose, California, November 1996, pp. 337-343
[12] M. Chatterjee and D.K. Pradhan, „A novel pattern generator for near-perfect fault coverage“, Proc. of VLSI Test Symposium 1995, pp. 417-425
[13] M. Chatterjee and D.K. Pradhan, „A BIST Pattern Generator Design for Near-Perfect Fault Coverage“, IEEE Transactions on Computers, vol. 52, no. 12, 2003, pp. 1543-1558
[14] P. Fišer, J. Hlavička and H. Kubátová, „Column-Matching BIST Exploiting Test Don't-Cares“, Proc. 8th IEEE Europian Test Workshop, Maastricht, 2003, pp. 215-216
[15] P. Fišer and H. Kubátová, „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
[16] P. Fišer and H. Kubátová, „Influence of the Test Lengths on Area Overhead in Mixed-Mode BIST“, Proc. 9th Biennial Conference on Electronics and Microsystem Technology 2004 (BEC'04), Tallinn (Estonia), 3.-6.10.2004, pp. 201-204
[17] 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
[18] 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
[19] Hortensius, et al. Cellular automata circuits for BIST, IBM J. R&Dev, vol 34, no 2/3, pp. 389-405, 1990
[20] O. Novák, „Pseudorandom, Weighted Random and Pseudoexhaustive Test Patterns Generated in Universal Cellular automata“, Springer: Lecture Notes in Computer Science 1667, pp. 303-320, 1999
[21] H.K. Lee, D.S. Ha, „Atalanta: an Efficient ATPG for Combinational Circuits“, Technical Report, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993
[22] G. De Micheli, “Synthesis and Optimization of Digital Circuits”, McGraw-Hill, 1994
[23] S. Chiusano, F. Corno, P. Prinetto and M. Sonza Reorda, „Cellular Automata for Sequential Test Pattern Generation“, 15th IEEE VLSI Test Symposium, Monterey, CA (USA), April 1997, pp. 60-65
[24] C. Fagot, O. Gascuel, P. Girard and C. Landrault, „On calculating efficient LFSR seeds for built-in self test“, Proc. IEEE European Test Workshop 1999, Constance, Germany, 1999, pp. 7-14

Table 3. Comparison the results

	Column-matching			Bit-fixing		Row-matching	
Bench	*TL*	*GEs*	*method*	*TL*	*GEs*	*TL*	*GEs*
c880	1 K	**15**	*a)*	1 K	27	1 K	21
c1355	2 K	9	*c)*	3 K	11	2 K	0
c1908	3 K	**7.5**	*a)*	4 K	12	4.5 K	8
c3540	4 K	**3**	*a)*	4.5 K	13	4.5 K	4
s420	1 K	**12**	*c)*	1 K	28	-	-
s641	2 K	**3**	*d)*	10 K	12	10 K	6
s713	2 K	4.5	*d)*	-	-	5 K	4
s838	6 K	**22.5**	*c)*	10 K	37	-	-
s1196	6 K	**15**	*a)*	-	-	10 K	36