Pseudorandom Testing – A Study of the Effect of the Generator Type

Petr Fišer, Hana Kubátová
Czech Technical University in Prague
Dept. of Computer Science & Engineering
Karlovo nám. 13, CZ-121 35, Prague 2, Czech Rep.
E-mail: fiserp@fel.cvut.cz, kubatova@fel.cvut.cz

Abstract*: The test pattern generator produces test vectors that are applied to the tested circuit during pseudo-random testing of combinational circuits. The nature of the generator thus directly influences the fault coverage achieved. In this paper we discuss the influence of the type of pseudo-random pattern generator on stuck-at fault coverage. Linear feedback shift registers (LFSRs) are mostly used as test pattern generators, and the generating polynomial is primitive to ensure the maximum period. We have shown that it is not necessary to use primitive polynomials, and moreover that their using is even undesirable in most cases. This fact is documented by statistical graphs. The necessity of the proper choice of a generating polynomial and an LFSR seed is shown here, by designing a mixed-mode BIST for the ISCAS benchmarks.*

An alternative to LFSRs are cellular automata (CA). We study the effectiveness of CA when used as pseudo-random pattern generators. The observations are documented by statistical results.

Keywords: built-in self-test, diagnostics, testability, LFSR, test pattern generators, column-matching

1. Introduction

The complexity of present-day VLSI devices has risen to millions of gates, and the chips are therefore becoming untestable by standard manufacture external ATE (Automated Test Equipment) testers. The test lengths are rapidly increasing, as are the testing times and the ATE memory requirements. Hence the built-in self-test (BIST) has been established as a necessary part of VLSI circuits. The circuit is able to test itself by BIST without using any ATE equipment, or when used together with an external tester, BIST significantly reduces the test time and tester memory demands.

Many BIST techniques have been developed [1, 2]. The vast majority of them use a pseudo-random pattern generator (PRPG) to produce test vectors that detect the easy-to-detect faults, which mostly represent more than 90% of the total faults. For the remaining faults, test vectors are either applied externally, or they are generated by the BIST structure itself.

Linear feedback shift registers (LFSR) or cellular automata (CA) are mostly used as PRPGs, due to their simplicity and good properties concerning implementation space demands and the good fault coverage.

A general BIST structure is shown in Fig. 1. The patterns are generated by a test pattern generator (TPG), then they are fed to the circuit-under-test (CUT) and the circuit's responses are evaluated.

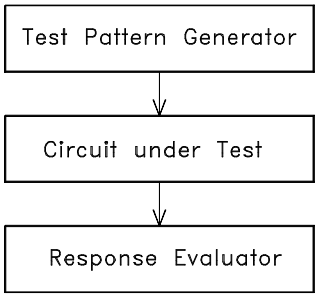

Figure 1: The BIST scheme

Test patterns may be applied to the circuit in parallel, which is denoted as a

test-per-clock BIST, or serially (*test-per-scan*) [1].

The design of the TPG is of key importance for the whole BIST, since it determines the fault coverage achieved and the area overhead of the BIST equipment. A simple LFSR often cannot ensure satisfactory fault coverage, thus it has to be augmented in some way. The LFSR code word sequence is modified in some approaches to produce patterns that detect more faults. These methods imply reseeding the LFSR during the test, or possibly the generating polynomial is also modified [3], or the LFSR patterns are modified by an additional logic [4, 5, 7, 13].

The best results are produced by *mixed-mode BIST* methods. Here some of the PRPG patterns are applied to the circuit unmodified to detect the easy-to-detect faults. After that, either deterministic or somehow modified PRPG patterns are generated to detect the remaining faults [2, 5, 6, 7].

The proper choice of a PRPG is very important in a case of mixed-mode testing. It is desirable to detect as many faults as possible by the PRPG, so that the additional logic is maximally reduced. This is the main issue addressed in this paper. We introduce statistics on the stuck-at fault coverages for the ISCAS [10, 11] and ITC'99 benchmarks [21], using different PRPGs. The influence of the PRPG on the total BIST area overhead is shown for the column-matching method [7, 13, 14], since this method enables high scalability, and the effects of the generator type and test lengths can be demonstrated here very well.

The paper is organized as follows: the basic principles of PRPGs are introduced in Section 2, the statistics of fault coverages are presented in Section 3, Section 4 briefly describes the mixed-mode BIST principles, together with the column-matching BIST method and the results obtained using this method. Section 5 concludes the paper.

2. The PRPG Structure

Generally, PRPGs are simple sequential circuits generating code words, according to the generating polynomial [23]. These code words are then either fed directly to the CUT inputs, or they are modified by some additional circuitry.

The most common PRPG structures are *linear feedback shift registers* (LFSRs) or *cellular automata* (CA). An *n*-bit (*n*-stage) LFSR is a linear sequential circuit consisting of D flip-flops and XOR gates generating code words (patterns) of a cyclic code. The structure of an *n*-stage LFSR-I (with internal XORs) is shown in Fig. 2.

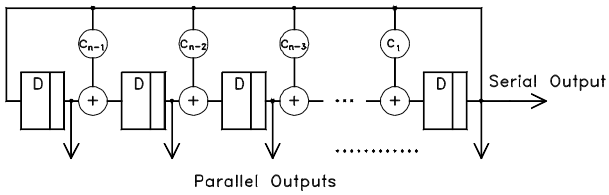

Figure 2. LFSR structure

The register has *n* parallel outputs corresponding to the outputs of the D flip-flops, and one flip-flop output can be used as a serial output of a register.

The coefficients $c_1 - c_{n-1}$ express whether there exists (1) a connection from the feedback to the corresponding XOR gate or no connection (0). Thus it determines whether there is a respective XOR gate present or the flip-flops are connected directly. The feedbacks leading to the XOR gates are also called *taps*.

The sequence of code words produced by an LFSR can be described by a *generating polynomial* g(x) in GF(2^n), [22].

$$g(x) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + ... + c_1x^1 + 1$$

If the generating polynomial is primitive, the LFSR has a maximum period 2^n-1, thus it produces 2^n-1 different patterns.

The initial state of the register (initial values of the flip-flops) is called the seed.

The second LFSR type, the LFSR-II is implemented with XORs in the feedback. Its generating polynomial is dual to the LFSR-I polynomial. Only LFSR-I will be considered in this paper, since these LFSRs are mutually convertible.

Cellular automata are sequential structures similar to LFSRs. Their periods are often shorter, but code words generated by CA are sometimes more suitable for test

patterns with preferred numbers of ones or zeros at the outputs.

An example of a CA performing multiplication of the polynomials corresponding to code words by the polynomial x+1 (rule 60 for each cell [19]) is shown in Fig. 3.

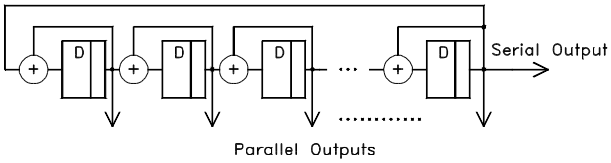

Figure 3. Example of a cellular automaton

3. Fault Coverage Statistics

We have performed extensive experiments on the standard ISCAS benchmarks, both combinational benchmarks [10] and full-scan versions of sequential benchmarks [11], to determine the fault coverage achieved by a pseudo-random test sequence generated by a PRPG. The FSIM fault simulator [12] has been used in all the examples to determine the fault coverage.

First we show that the testability and the fault coverage achieved by a certain number of pseudo-random test vectors strictly depend on the tested circuit. Knowledge of the testability of the circuit for which the BIST is being designed can help us to select properly the lengths of the BIST phases [20].

Then we demonstrate the effect of the generator type on the stuck-at fault coverage, and show that the simplest LFSR is sufficient for most of applications.

3.1. Pseudo-Random Testability of the Circuits

A low area overhead and good speed of the designed BIST strictly depend on the nature of the circuit, for which the BIST is being designed. Pseudo-random testability of a particular circuit strictly depends on the number of hard-to-detect faults. It is possible to apply an unmodified sequence of LFSR code words to fully test some circuits in a reasonable number of cycles, while some other circuits are particularly untestable by this way.

We have studied the pseudo-random testability of the ISCAS [10, 11] and ITC'99 [21] benchmarks, using standard LFSRs. All the benchmarks were in their full-scan versions, thus turned into combinational. Each benchmark was tested 1000 times using different LFSR polynomials and seeds. Both the polynomials and the seeds were randomly generated, while a satisfactory period length was ensured by a simulation of the PRPG run. The number of LFSR bits was set to be equal to the number of CUT inputs. The results of a simulation of a selected set of benchmarks are shown in Table 1. The "*i*" column shows the number of the benchmark inputs (including the scan path for sequential circuits), "*range*" indicates the range of the encountered number of test patterns to fully test the circuit (in those 1000 samples), while the statistical average value is shown in the last column. "K" stands for thousands of patterns, "M" for millions, "G" for billions.

For some benchmarks the range has not been evaluated, for an extremely large number of patterns needed to fully test the circuit (more than 10 M).

Table 1. Pseudo-random testability

bench	*i*	range	avg
c17	5	2 – 33	4
c432	36	250 – 120	600
c499	41	300 – 6 K	1 200
c880	60	2 500 – 57 K	13 K
c1355	41	800 – 12 K	2 800
c1908	33	3 K – 77 K	12 K
c2670	233	2.4 M – 12.5 M	4.4 M
c3540	50	5 K – 174 K	32 K
c5315	178	1 400 – 5 K	2 500
c6288	32	33 – 474	131
c7552	207	> 100 M	
s27	7	2 – 192	29
s208.1	18	1 400 – 26 K	6 K
s298	17	100 – 1000	500
s344	24	60 – 1000	250
s349	24	70 – 1000	250
s382	24	150 – 2000	500
s386	13	1 400 – 15 K	3 600
s400	24	120 – 2000	500
s420.1	34	165 K – 4 M	1.4 M
s444	24	130 – 2000	500
s510	25	300 – 2500	900
s526	24	5 K – 67 K	19 K
s641	54	196 K – 3.2 M	1 M
s713	54	294 K – 3.4 M	1 M
s820	23	10 K - 78 K	27 K
s832	23	9 K – 75 K	27 K
s838	67	> 100 M	
s953	45	15 K – 98 K	46 K
s1196	32	196 K – 3.2 M	1 M
s1238	32	21 K – 489 K	118 K
s1423	91	9 K – 138 K	55 K
s1488	14	2500 – 24 K	6 800

bench	*i*	range	avg
s1494	14	2200 – 23 K	5 K
s5378	214	50 K – 196 K	82 K
s9234.1	247	6 M – 30 M	15 M
s13207.1	700	97 K – 879 K	329 K
s15850.1	611	> 10 M	
s35932	1763	150 – 500	230
s38417	1664	> 10 M	
s38584.1	1464	> 1 G	
b01	7	1 – 1100	350
b02	5	1 – 1000	200
b03	34	30 – 2600	700
b04	77	14 K – 330 K	60 K
b05	35	10 K – 70 K	25 K
b06	11	1 – 1200	330
b07	50	220 K – 10 M	3 M
b08	30	2 K – 60 K	13 K
b09	29	4 K – 42 K	16 K
b10	28	300 – 5 K	1700
b11	38	12 K – 160 K	50 K
b12	126	5 – 44 M	13 M
b13	63	700 – 15 K	5 K
b14	277	> 100 M	
b15	485	> 100 M	
b17	1452	> 100 M	
b18	3307	> 100 M	
b19	6666	> 100 M	
b20	522	> 100 M	
b21	522	> 100 M	
b22	767	> 100 M	

It can be seen that the number of pseudo-random patterns needed to fully test the circuits varies considerably. The distribution of the number of required patterns follows the curve shown in Fig.4. This particular curve corresponds to the ISCAS c1908 circuit.

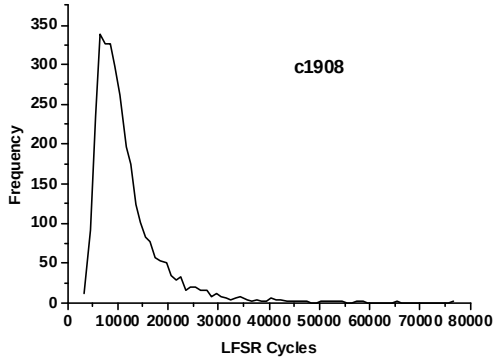

Figure 4. Distribution of the number of patterns to achieve full fault coverage for c1908

3.2. Influence of LFSR type on Test Length

An LFSR used as a pseudo-random pattern generator is mostly based on the primitive generating polynomial to provide the longest period of the code generated. In this Subsection we show that it is not necessary to use primitive polynomials. We investigated the influence of the number of LFSR taps on the testing capability. In particular, we studied the number of patterns needed to test all the faults in a circuit (like in Subsection 3.1), while varying the number and the position of the LFSR taps. A satisfactory period for each generated LFSR was ensured by simulating its run. The results of the experiment are shown in Fig. 5. Here the number of LFSR cycles needed to cover all the stuck-at faults in the c1355 circuit is shown. 100 different LFSRs were generated randomly for each LFSR size, differing both in the tap positions and the seed. Thus, for the circuit used (having 40 inputs) 3900 different LFSRs were produced (the x-axis – LFSR in Fig. 5). LFSRs 0-99 correspond to 1-tap LFSRs, 100-199 correspond to 2-tap LFSRs, and so on. It can be observed that the number of taps does not influence the fault coverage capability at all; the test lengths are steadily distributed. Thus, we can conclude that the most advantageous LFSR is one of the 1-tap LFSRs, since its area overhead is the smallest (only one feedback). A 1-tap LFSR having a satisfactory period can be found in most cases. The use of primitive polynomials thus becomes counterproductive, since the number of taps is mostly greater than one here, and they do not make any contribution.

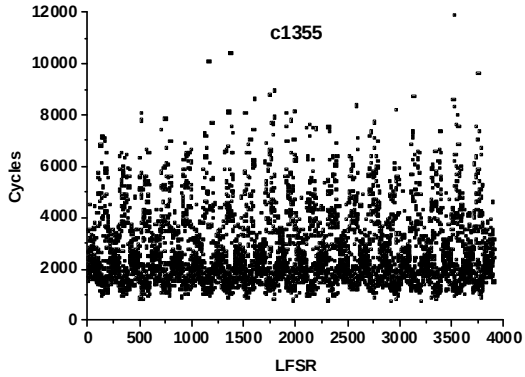

Figure 5. Influence of the LFSR

4. Mixed-Mode BIST Principles

The number of faults detected by pseudo-random patterns successively applied to the CUT follows the saturation curve, see Fig. 6. Here the LFSR patterns were gradually applied to the s1196 ISCAS benchmark,

while the number of covered faults was recorded. It can be observed that 90% of the faults were covered in the first 1000 cycles, while 60 000 cycles were needed to achieve a complete fault coverage.

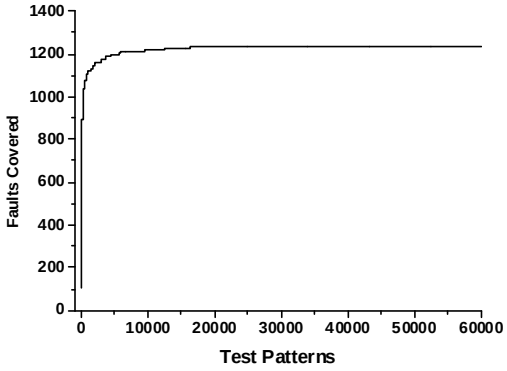

Figure 6. Fault Coverage Curve

Thus, it is advantageous to apply a relatively small number of pseudo-random patterns to cover the easy-to-detect faults, and then produce several deterministic patterns to cover the rest. This approach is called a *mixed-mode BIST*.

It is necessary to find a trade-off between the number of pseudo-random and deterministic patterns. Moreover, the PRPG has to be properly chosen, since the fault coverage differs notably with PRPG type, as we have shown in Subsection 3.1.

The probability of covering a given number of faults by a PRPG is illustrated by Fig. 7. Here, sets of 10, 50, 100, 500, 1000 and 5000 LFSR patterns were applied to the c3540 circuit, 10 000 samples for each test size. The seed and the tap position were selected randomly for each sample. The distribution of the number of faults that remained undetected is shown here. For a low number of patterns many faults are left undetected, and their number also varies considerably. As the number of test patterns increases, the number of undetected faults rapidly decreases, while the standard deviation of this number decreases as well.

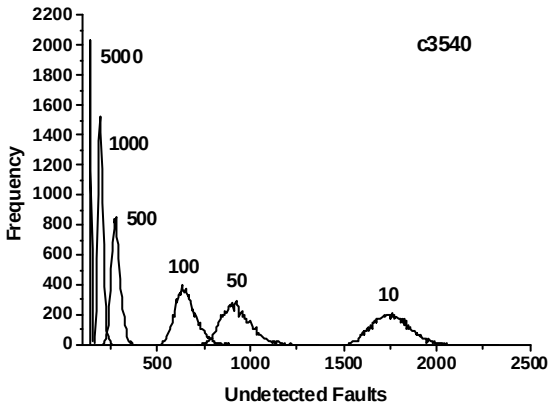

Figure 7. Pseudo-random fault coverage

4.1. Cellular Automata vs. LFSRs

Many methods using cellular automata instead of LFSR have been proposed [17, 18]. In general, pseudo-random patterns generated by a CA have a more random nature than those generated by an LFSR. The *weights* of the particular PRPG outputs (i.e., the ratios of zeroes and ones) are balanced in LFSRs, approaching the value 0.5. Cellular automata often have the weights misbalanced, according to the seed. This property gives an advantage to CA, since they can be advantageously exploited as generators of weighted test patterns [18].

We studied the fault covering capabilities of cellular automata seeded with a random vector and compared the results with random LFSRs (randomly generated polynomial and seed). A very interesting observation was made.

The distribution of the number of undetected faults, exactly as in Fig. 7, was studied for LFSRs and CA, rule 60 (see Fig. 3). The distribution curves for these two types of generators are shown in Fig. 8. Here, 500 patterns generated by random LFSRs and random CA 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 CA instead of an LFSR does not increase the number of covered faults on an average, but the probability that more faults will be covered by the CA vectors is increased.

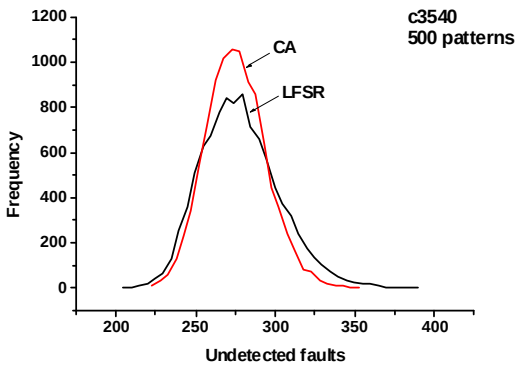

Figure 8. Comparison of the fault coverage obtained by LFSR and CA

This experiment shows that the use of randomly seeded CA instead of LFSRs does not make any contribution to the fault coverage achieved. 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 chosen for a cellular automaton, its fault covering properties will change dramatically.

We performed an experiment similar to that shown in Fig. 8, for the s838 ISCAS circuit using an LFSR and a CA, once with a steady distribution of values in its seed, and once with a seed having *only one* "1" value at a random position, so that 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. 9. We can observe that the fault coverage of the LFSR decreased rapidly for this special seed, but on the other hand the variability of fault coverage of the CA increased, while in some cases many more faults were covered by the vectors produced by this CA (left-hand side).

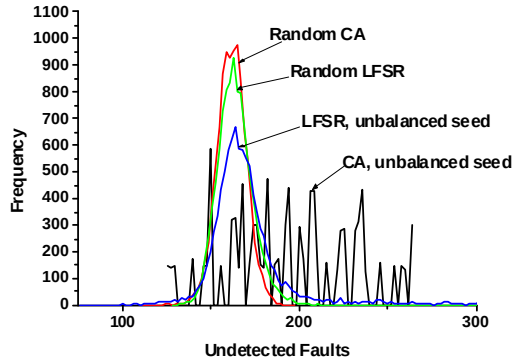

Figure 9. Fault coverage of a "specially" seeded CA and LFSR

This observation can be explained by unsteady distribution of weights, i.e. the probabilities of occurrence of the "1" value on the PRPG outputs. The distribution of weights for four 100-bit PRPGs running 1000 cycles is shown in Figures 10a-10d.

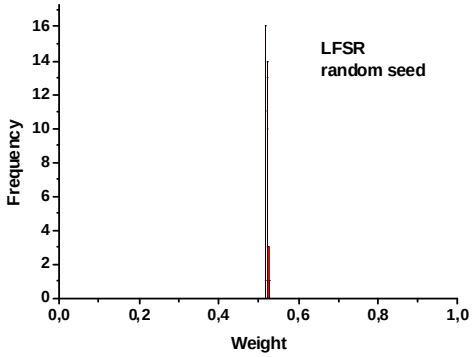

Fig. 10a. Distribution of weights for different PRPGs – LFSR with a balanced random seed

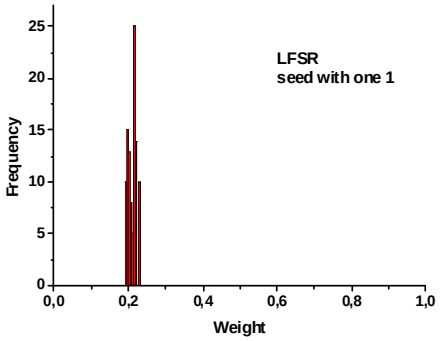

Fig. 10b. Distribution of weights for different PRPGs – LFSR with an unbalanced random seed

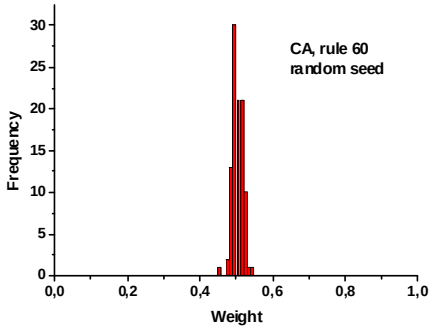

Fig. 10c. Distribution of weights for different PRPGs – CA with a balanced random seed

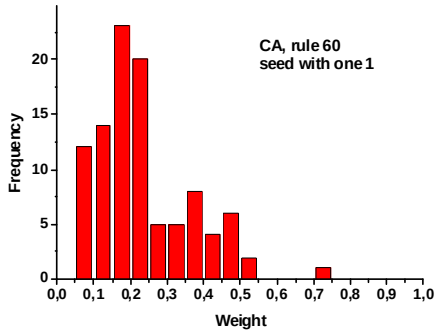

Fig. 10d. Distribution of weights for different PRPGs – CA with an unbalanced random seed

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. 10a and c). 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 for this particular case (Fig. 10b). 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 it has been for all of our experiments. If a LFSR with a bigger number of taps was chosen, all the weights would near to 0.5, similarly to the case of a balanced seed. In Figure 10d the CA seeded with an unbalanced seed (having one "1" value) is shown. The weights range from negligible values (all zeroes) to more than 0.7. This is the case where the weighted pattern testing can be advantageously applied.

4.2. 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 block, called *Output Decoder*. The method is designed for combinational or sequential full-scan circuits, thus the order of the test patterns applied to the CUT is insignificant. Moreover, not all the PRPG patterns have to be transformed into test patterns; the excessive ones just do not test any new faults.

In our column-matching method we try to assign the PRPG code words to the deterministic patterns, so that some of the columns are equal. Then the decoding logic needed to implement the matched column will 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 a more detailed description, see [13, 14].

This principle has been further extended to support mixed-mode testing [7]. The BIST run is divided into two disjoint phases. First, the circuit is tested using an unmodified sequence of LFSR code words, detecting the easy-to-detect faults. The deterministic test patterns for the rest of the faults are computed by the Atalanta ATPG tool [15]. These vectors are to be generated by several consecutive LFSR code words and modified by the Decoder to obtain deterministic vectors. There has to be some additional logic to control the switch between the two phases. The *switch* is implemented as an array of multiplexers, one for each CUT input. However we attempt to eliminate the MUXes as well, by introducing *direct matches* [7]. The structure of a mixed-mode BIST is shown in Fig. 11.

The sequence of patterns is fed to the tested circuit and its response is then evaluated by a multi-input shift register (MISR).

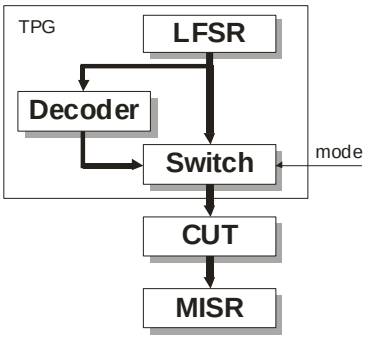

Figure 11. Mixed-mode BIST structure

4.3. Test Lengths

It is clear that the choice of appropriate lengths of these two phases is of key importance. The maximum number of faults should be detected in the pseudo-random phase, while its length should be acceptable. According to Fig. 6, most of the faults can be

detected by a few initial patterns, and for the remaining faults deterministic patterns have to be produced. The more faults remain undetected, the more ATPG vectors are needed, which also complicates the Decoder design, in terms of the area overhead. This can be compensated by a longer run of the deterministic phase to some extent, but not significantly.

The influence of the length of the initial phase on the final result is illustrated by Table 2. The lengths of the two phases are shown in the *"rand / det."* column. After the *"rand"* pseudo-random (unmodified) patterns were applied to the CUT, *"ud."* faults were left undetected and *"vct."* deterministic vectors were produced by the Atalanta ATPG tool to detect them. 100% coverage of detectable stuck-at faults is considered in all the cases. Thus, the *"ud."* column does not include the number of redundant faults, which cannot be detected, from the nature of the tested circuit. The deterministic vectors are to be generated from additional *"det."* LFSR patterns by the Decoder. The area overhead of the BIST decoder synthesized by a column-matching method is indicated in the *"GEs"* column. It is described in terms of gate equivalents [16]. Only the logic of the decoder and the switching logic is considered and stated in this column, while the overhead of the PRPG and BIST control logic is not included here.

Table 2. Influence of test length

bench	*rand / det.*	*ud.*	*vct.*	*GEs*
c1355	500 / 500	31	12	70
	1000 / 1000	8	1	15
c1908	1000 / 1000	46	30	46.5
	2000 / 1000	19	10	7.5
c3540	1000 / 1000	33	22	15
	2000 / 1000	8	8	7.5
	5000 / 1000	3	3	6
s420	400 / 600	40	30	24.5
	1000 / 1000	35	19	25.5
	3000 / 1000	29	17	27
s526	500 / 500	21	17	30.5
	1000 / 1000	12	11	4.5
	2000 / 1000	7	6	4.5
s641	1000 / 500	12	9	21
	3000 / 1000	8	7	15
	5000 / 1000	7	6	16.5
s820	1000 / 1000	70	28	63
	5000 / 5000	34	14	0
s838	1000 / 1000	129	72	120
	5000 / 1000	105	56	130
	10000 / 1000	106	62	110
s1196	1000 / 1000	89	54	50.5
	5000 / 1000	25	19	28.5

It can be seen that increasing length of the pseudo-random phase decreases the BIST overhead to some extent. A significant decrease in the overhead is achieved for a small increase in the test length in some cases (c1355, c1908, s820). Sometimes the improvement is negligible even when the test length is increased significantly (s641, s838). Sometimes a longer pseudo-random phase even causes an increase in the area overhead (s420, s641). This is due to the fact that the amount of test don't cares decreases for smaller test size and complicates the decoder synthesis [14].

4.4. Influence of the LFSR

The fault coverage achieved in the first phase is influenced not only by the number of pseudo-random test patterns (the length of the pseudo-random phase). The number of detected faults also depends on the pseudo-random sequence, so it is influenced by the LFSR (CA) polynomial and seed. This is illustrated by Figures 4 and 7. Significantly different results are produced for different LFSRs, even when the lengths of the phases are retained. For illustration, we designed a BIST for the c1908 circuit. The pseudo-random phase was run for 2000 cycles, the LFSR polynomial was set constant (1-tap) and we repeatedly randomly reseeded it. Then the deterministic phase was run for 1000 clock cycles. The simulation results are shown in Table 3. Again, the *"ud."* column indicates the number of undetected faults in the first phase, *"vct."* gives the number of deterministic vectors and *"GEs"* shows the complexity of the final BIST logic. The entries are sorted by the number of undetected faults.

We can see that the complexity of the final circuit strictly depends on the LFSR seed – it varies from 7.5 GEs up to 69 GEs.

It is not possible to compute a proper LFSR seed and/or generating polynomial analytically for practical examples, due to the complexity of this problem. Thus, in practice

we repeatedly reseed the polynomial and conduct the fault simulation several times, while we pick out the best seed for further processing. Fault simulation is often a very fast process, thus it does not significantly influence the BIST design time.

Table 3. Influence of the LFSR seed

ud.	*vct.*	*GEs*	*ud.*	*vct.*	*GEs*
19	10	7.5	33	15	37
21	9	19.5	34	16	33
24	13	23.5	36	18	38
26	15	28	37	20	40.5
26	13	25	39	22	53
28	15	37.5	44	26	40
28	14	22.5	46	22	42.5
30	14	36	48	24	44
32	16	31	52	28	63.5
33	17	27.5	62	34	69

5. Conclusions

We have discussed the influence of the pseudo-random pattern generator type on its fault detection capability. Both LFSRs and CA are studied, with either a random or a “special” seed. The distribution of weights on the individual PRPG outputs is shown for all cases, together with the fault coverage curves obtained by the PRPGs.

We have shown that for a pseudo-random test-pattern generation phase a 1-tap LFSR is mostly a good choice, due to its satisfactory period length, fault coverage and minimal area overhead.

The pseudo-random testability of the standard ISCAS and ITC benchmarks is summarized in this paper, to help BIST designers properly choose the desired pseudo-random test lengths for these circuits.

The effects of generator type are illustrated on a mixed-mode column-matching BIST synthesis. It directly influences the total complexity of the resulting BIST circuitry. The claims were confirmed experimentally on a BIST design for several ISCAS benchmarks, but the conclusions made can be applied to any circuits.

Acknowledgement

This research was supported by grant GA 102/04/2137, "Design of Highly Reliable Control Systems Built on Dynamically Reconfigurable FPGAs" and MSM6840770014

References

[1] Agrawal, V., K. - Kime, C., R. - Saluja, K., K.: *A tutorial on BIST, part 1: Principles*. IEEE Design & Test of Computers, **vol. 10, No.1** March 1993, pp.73-83, part 2: Applications, No.2 June 1993, pp.69-77

[2] Touba, N., A. - McCluskey, E., J.: *Synthesis Techniques for Pseudo-Random Built-In Self-Test*. Technical Report, (CSL TR # 96-704), Dept. of Electrical Engineering and Computer Science, Stanford University, August 1996

[3] Hellebrand, S. et al.: *Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers*. IEEE Trans. on Comp., vol. 44, No. 2, February 1995, pp. 223-233

[4] Hartmann, J., Kemnitz, G.: How to Do Weighted Random Testing for BIST, Proc. of International Conference on Computer-Aided Design (ICCAD), pp. 568-571, 1993

[5] Chatterjee, M. - Pradhan, D.K.: *A BIST Pattern Generator Design for Near-Perfect Fault Coverage.* IEEE Transactions on Computers, **vol. 52, no. 12**, December 2003, pp. 1543-1558

[6] Touba, N.A.: *Synthesis of mapping logic for generating transformed pseudo-random patterns for BIST*, Proc. of International Test Conference, pp. 674-682, 1995

[7] Fišer, P. - Kubátová, H.: *An Efficient Mixed-Mode BIST Technique.* DDECS'04, Tatranská Lomnica, SK, 18.-21.4.2004, pp. 227-230

[8] Aloke, K. - Chaudhuri, D.P.: *Vector Space Theoretic Analysis of Additive Cellular Automata and Its Application of Pseudoexhaustive Test Pattern Generation*. IEEE Transactions on Computers, **Vol. 42, No. 3**, March 1993, pp. 340-352

[9] Novák, O. - Hlavička, J.: *Design of a Cellular Automaton for Efficient Test Pattern Generation*. Proc. IEEE ETW 1998, Barcelona, Spain, pp. 30-31

[10] Brglez, F. - Fujiwara, H.: *A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan.* Proc. of International Symposium on Circuits and Systems, pp. 663-698, 1985

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

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

[13] Fišer, P. - Hlavička, J.: *Column-Matching Based BIST Design Method*. Proc. 7th IEEE European Test Workshop (ETW'02), Corfu (Greece), 26.-29.5.2002, pp. 15-16

[14] Fišer, P. - Hlavička, J. - Kubátová, H.: *Column-Matching BIST Exploiting Test Don't-Cares*. Proc. 8th IEEE European Test Workshop (ETW'03), Maastricht (The Netherlands), 25.-28.5.2003, pp. 215-216

[15] Lee, H.K. - Ha, D.S.: *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

[16] De Micheli, G.: *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[17] Hortensius, et al. *Cellular automata circuits for BIST*. IBM J. R&Dev, vol 34, no 2/3, pp. 389-405, 1990

[18] Novák, O.: *Pseudorandom, Weighted Random and Pseudoexhaustive Test Patterns Generated in Universal Cellular automata*. Springer: Lecture Notes in Computer Science **1667**, pp. 303-320, September 1999

[19] Chaudhuri, P.P. et al.: *Additive Cellular Automata Theory and Applications*. Volume I. IEEE Computer Society Press, 1997, 340 p.

[20] Fišer, P. - Kubátová, H.: *Influence of the Test Lengths on Area Overhead in Mixed-Mode BIST*. Proc. 9th Biennial Baltic Electronics Conference (BEC'04), Tallinn (Estonia), 3.-6.10.2004, pp. 201-204

[21] Corno, F. - Sonza Reorda, M. - Squillero, G.: *RT-Level ITC 99 Benchmarks and First ATPG Results*. IEEE Design & Test of Computers, July-August 2000, pp. 44-53

[22] Adamek, J.: *Foundations of Coding*. John Wiley & Sons, Inc. 1991, 336 p.

[23] Stroud, Ch. E.: *A Designer's Guide to Built-In Self-Test*. Kluwer Academic Publisher, London, 2002