Pseudorandom Testability – 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

In a pseudo-random testing of combinational circuits the pattern generator produces test vectors that are being applied to the tested circuit. The nature of the generator thus directly influences the fault coverage achieved. In this paper we discuss an influence of the type of the pseudo-random pattern generator on the fault coverage. In most cases the LFSR is used as a pattern generator, while its generating polynomial is primitive to ensure a maximal period. We show that using primitive polynomials is not necessary, and in most cases even undesirable. This fact is documented by statistical graphs. The necessity of properly selecting a generating polynomial and a LFSR seed is shown here, by designing a mixed-mode BIST for the ISCAS benchmarks.

1. Introduction

As the complexity of the present VLSI devices increases to millions of gates, the chips are becoming untestable using standard manufacture external ATE testers. The test lengths rapidly increase, and so do the testing times and ATE memory requirements. Hence the built-in self-test (BIST) was established as a necessary part of VLSI circuits. Then the circuit is able to test itself without using any ATE equipment, or when used together with an external tester, the BIST significantly reduces the test time and tester memory demands.

Up to now many BIST techniques were developed [1, 2]. A 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, the test vectors are applied either externally, or they are generated by the BIST equipment itself.

As a PRPG a linear feedback shift register (LFSR) is mostly used, for its simplicity.

A general structure of a BIST 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.

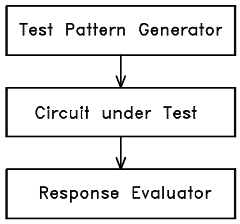

Figure 1: The BIST scheme

Test patterns might be applied to the circuit in parallel, which is denoted as a *test-per-clock* BIST, or serially (*test-per-scan*).

The design of the TPG has a 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. In some approaches the LFSR code word sequence is being modified, to produce patterns detecting more faults. These methods imply reseeding the LFSR during the test, eventually the generating polynomial is being modified too [3], or the LFSR patterns are being modified by an additional logic [4, 5].

Best results are being 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].

In a case of a mixed-mode testing, properly selecting a PRPG is very important. Detecting as many faults as possible by a PRPG is desired, so the additional logic is maximally reduced. This is the main issue addressed in this paper. We introduce statistics on the fault coverages for the ISCAS benchmarks, using different PRPGs. Influence of the PRPG on the total BIST area overhead is shown for the column-matching method [7].

The paper is organized as follows: basic principles of the PRPGs are introduced in Section 2, the statistics of the 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, the PRPGs are easily implementable circuits that generate deterministic code words having random characteristics. 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 with internal XORs is shown in Fig. 2.

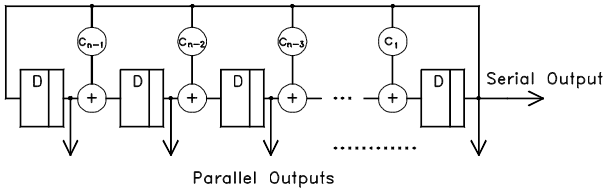

Figure 2. LFSR structure

The register has *n* parallel outputs drawn from the outputs of the D flip-flops, 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 not (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 referenced as *taps*.

The sequence of code words that are produced by a LFSR can be described by a *generating polynomial* g(x) in GF(2^n):

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

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 a register (initial values of the flip-flops) is called a *seed*.

3. Fault Coverage Statistics

In order to determine the stuck-at fault coverage reached by a pseudo-random test sequence generated by a PRPG, we have made extensive experiments on the standard ISCAS benchmarks, both the combinational ones [10] and the full-scan versions of the sequential ones [11]. In all the examples the FSIM fault simulator [12] was used.

3.1. Pseudo-Random Testability of the Circuits

In order to design a fast BIST with a low area overhead, it is necessary to thoroughly study the nature of a circuit, for which the BIST is being designed. Pseudo-random testability of a particular circuit strictly depends on the number of the hard-to-detect faults. For some circuits, it is possible to apply an unmodified LFSR sequence of code words to fully test it in a reasonable number of cycles, while some circuits are particularly untestable by this way.

We have studied the pseudo-random testability of the ISCAS benchmarks, using standard LFSRs. Each benchmark was tested 1000-times using different LFSR polynomials and seeds. Both the polynomials and seeds were randomly generated, however a satisfactory period length was ensured. The

number of the LFSRs stages was set to the number of the 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, *"range"* indicates the range of the encountered number of the test patterns to fully test the circuit (in those 1000 samples), while the statistic average value is shown in the last column.

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
s1494	14	2200 – 23 K	5 K
s5378	214	50 K – 196 K	82 K
s9234.1	247	> 10 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	

It can be seen that the number of pseudo-random patterns needed to fully test the circuits notably vary. The distribution of the number of required patterns follows the curve shown in Fig. 3. This particular curve corresponds to the c1908 circuit.

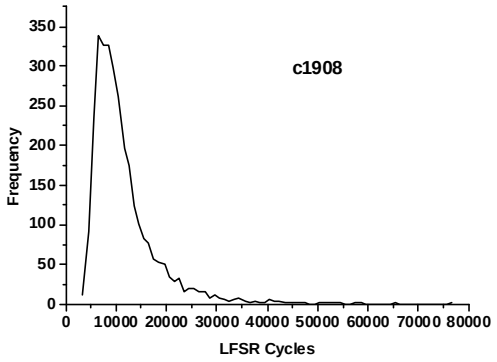

Figure 3. Distribution of the number of the patterns to reach full fault coverage for c1908

3.2. Influence of the PRPG on the Test Length

In most methods exploiting a LFSR as a pseudo-random pattern generator its generating polynomial is chosen to be primitive to provide the longest period of the code words generated. In this Subsection we show that it is not necessary to use primitive polynomials. We have investigated the influence of the number of the LFSR taps on the testing capability. In particular, we have studied the necessary number of patterns needed to cover all the faults in a circuit (like in Subsection 3.1), while varying the number of the LFSR taps. For each LFSR a satisfactory period was ensured, by a simulation of its run. The results of the experiment are shown in Fig. 4. Here the number of LFSR cycles needed to cover all the faults in the c1355 circuit is shown. For each LFSR size 100 different LFSRs were produced, differing both in the tap positions and the seed. Thus, for the circuit used (having 40 inputs) 3900 different LFSRs were produced. The LFSRs 0-99 correspond to the 1-tap LFSRs, 100-199 to the 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 to use is one of 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 taps is mostly bigger than one here and they do not bring any contribution.

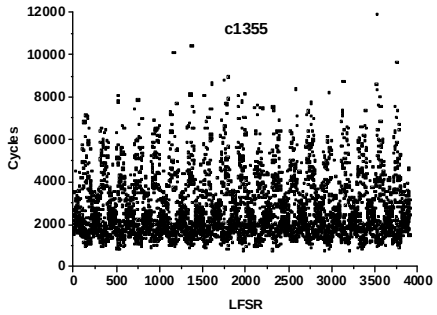

Figure 4. Influence of the LFSR

4. Mixed-Mode BIST Principles

When pseudo-random patterns are being successively applied to the CUT, the number of faults detected by these patterns follows the saturation curve, as it is shown in Fig. 5. 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 reach a complete fault coverage.

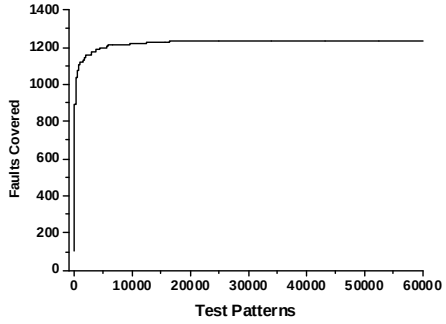

Figure 5. Fault Coverage Curve

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*.

There has to be found a trade-off between the number of pseudo-random and deterministic patterns. A probability of covering a given number of faults by a PRPG is illustrated by Fig. 6. Here sets of 50, 100, 500 and 1000 LFSR patterns were applied to the c3540 circuit, 1000 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. With an increasing number of the test patterns the number of undetected faults rapidly decreases, while the variance of this number decreases as well.

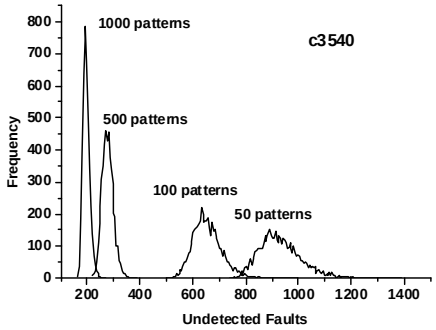

Figure 6. Pseudo-random fault coverage

4.1. Column-Matching BIST

The column-matching BIST method is based on a transformation of the LFSR code words into deterministic test patterns pre-computed by some ATPG tool. This transformation is being done by a combinational block, the *Output Decoder*. The method is designed for combinational or full-scan circuits, thus the order of the test patterns applied to the CUT is insignificant. Moreover, not all the LFSR patterns have to be transformed info test patterns; the excessive ones just do not test any faults.

In our column-matching method we try to assign the LFSR code words to the deterministic patterns, so that some of the columns 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 [13, 14].

This principle was further extended to support the 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. For the rest of the faults deterministic test patterns are computed by Atalanta ATPG tool [15]. These vectors are to be produced by the Decoder. There has to be some additional logic present, 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 MUXes as well, by introducing *direct matches* [7]. The structure of a mixed-mode BIST is shown in Fig. 7.

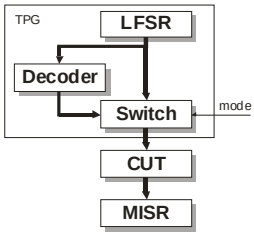

Figure 7. Mixed-mode BIST structure

4.2. The Test Lengths

It is clear that choosing appropriate lengths of the two phases is of a key importance. Maximum faults should be detected in the pseudo-random phase, while its length should be acceptable. According to Fig. 5 the majority of 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 an area overhead. It can be compensated by a longer run of the deterministic phase to some extent, however 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 are applied to the CUT, *"ud."* faults were left undetected and *"vct."* deterministic vectors were produced to detect them. These 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 given in terms of gate equivalents [16]. Only the logic of the decoder and the switching logic is considered, the overhead of the LFSR is not included here.

Table 2. Influence of the 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 when increasing the length of the pseudo-random phase the BIST overhead decreases to some extent. In some cases a significant decrease of the overhead is reached for a small increase of the test length (c1355, c1908, s820). In some cases the improvement is negligible even when the test length is significantly increased (s641, s838). Sometimes a longer pseudo-random phase causes even an increase of the area overhead (s420, s641). This is due to the fact that the amount of the test don't cares decrease for smaller test size, which complicates the decoder synthesis.

4.3. Influence of the LFSR

Not only the length of the pseudo-random test influences the fault coverage reached in the first phase. The number of detected faults also depends on the pseudo-random sequence, thus it is influenced by the LFSR polynomial and seed. This is illustrated by Figures 4 and 6. For different LFSRs, significantly different results are produced, even when the lengths of the phases are retained. For illustration, we have 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 have 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, *"GEs"* shows the complexity of the BIST. 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 selected – it varies from 7.5 GEs up to 69 GEs.

Computing a proper LFSR seed and/or generating polynomial analytically is impossible for practical examples, due to the complexity of this problem. Thus, in praxis we repeatedly reseed the polynomial and conduct a fault simulation several times, while we pick out the best seed for further processing. The 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 presented a discussion on the influence of the pseudo-random pattern generator type on its fault detection capability, which directly influences the complexity of the resulting BIST circuit. We have shown that selecting a 1-tap LFSR is mostly a good solution, for its satisfactory period length and fault coverage, while the area overhead is minimum.

The claims were confirmed experimentally on a BIST design for ISCAS benchmarks, however the conclusions made can be applied to any circuits.

Acknowledgement

This research was supported by a grant GA 102/04/0737 and MSM 212300014

References

[1] Agarwal, 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., Hlavicka, 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 Paralel 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 Europian 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 Europian 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.