Influence of the Test Lengths on Area Overhead in Mixed-Mode BIST

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 this paper we present a discussion on choosing the test lengths in our mixed-mode BIST technique. The BIST design method is based on the column-matching algorithm proposed before. The mixed-mode strategy divides the test sequence into two disjoint phases: first the pseudo-random phase detects the easy-to-detect faults, and the subsequent deterministic phase generates test vectors needed to fully test the circuit. The lengths of these two phases directly influence both the test time and the BIST area overhead, as well as the BIST design time. Some kind of trade-off has to be found, to design the BIST circuitry efficiently. The pseudo-random testability of the ISCAS benchmarks is studied here. The conclusions obtained here can be generalized to be applied to any circuit.

1 Introduction

As the complexity of present VLSI circuits rapidly grows, their testing using only external test equipment (ATE) is becoming impossible, mainly due to a huge amount of test vectors, long testing time and very expensive tester memory. Incorporating the Built-in Self-Test (BIST) becomes necessary. It requires no external tester to test the circuit, since all the circuitry needed to conduct a test is included in the very circuit. This is paid by an area overhead, long test time and often a low fault coverage. To achieve a complete stuck-at fault coverage, either a test time is prohibitively long (exhaustive test), or the area overhead is extremely large (when deterministic test patterns are stored in ROM). Thus, several compromise techniques were developed [1-7]. To the most efficient methods belong *mixed-mode BIST* techniques. Here the circuit-under-test (CUT) is being tested by several pseudo-random test patterns generated mostly by the linear feedback shift register (LFSR). These patterns cover the easy-to-detect faults. For the remainder of the faults deterministic patterns are generated, usually by modifying the non-detecting patterns [5-7].

Lately, we have proposed a mixed-mode BIST method based on a column-matching principle [8, 9]. Here the test is divided into two disjoint phases: the pseudo-random and the deterministic. Choosing proper lengths of these phases is of a key importance to design a good BIST, since the BIST area overhead highly depends on the lengths of both the phases.

The influence of the test lengths on the area overhead is studied on ISCAS benchmarks. Some general conclusions can be made from the results, which help us to design a universal BIST design method.

The paper is structured as follows: the principles of our mixed-mode BIST are described in Section 2. Section 3 shows the way how the length of the pseudo-random phase should be selected. The length of the deterministic phase is discussed in Section 4. Section 5 presents a comparison of our method with others. Section 6 concludes the paper.

2 Mixed-Mode BIST

Our mixed-mode BIST technique is intended for combinational or full-scan circuits. It is designed for a test-per-clock testing, thus the test vectors are fed to the CUT in parallel. The basic structure of the BIST is shown in Fig. 1. The test pattern generator (TPG) consists of the *LFSR*, the combinational *Decoder* and the *Switch*. In general, the Switch is an array of multiplexers, alternating between the two phases. At the beginning the circuit is tested by unmodified LFSR patterns. This enables us to detect the majority of faults. After that we switch to the deterministic phase and test the yet undetected faults by pre-computed test patterns. These are generated by the Decoder.

The logic needed to drive the switching signal represents a negligible area overhead. It can be implemented either as a counter or a LFSR pattern analyzer, thus with a constant area overhead.

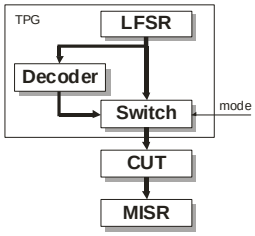

Figure 1. Mixed-mode BIST structure

The Decoder logic is synthesized using our column-matching algorithm [8]. The Decoder is a combinational block transforming some of the LFSR patterns into deterministic patterns pre-computed by an ATPG. Our aim is to design the decoder to be as small as possible. Its design is based on "matching" maximum of the decoder outputs with its inputs. Particularly, when the test vectors are reordered and assigned to the LFSR vectors in such a way that the values in the respective matched columns (i.e., input and output variables) are equal, the matched output will be implemented as a wire, without any logic. Since the BIST is designed for combinational circuits, any reordering can be freely done. Moreover, the deterministic test can be much longer than the computed test sequence. Only few of the LFSR patterns produce the required test vectors and the rest represent the non-testing "gaps". This gives us a big freedom how to select the appropriate matches. For more details see [8]. The values of the non-matched outputs have to be synthesized by some Boolean minimizer, i.e. BOOM [10, 11].

The algorithm was then slightly modified to support the mixed-mode BIST. Originally, when a column match is found, no decoder logic is needed, however the multiplexer has to be present in the Switch. The only case where absolutely no logic is necessary to implement an output is when an *i-th* output variable is matched with an *i-th* input variable. Then the values of the *i-th* output will be copied from the *i-th* input in both the phases. Such a special case of a column match will be denoted as a *direct column match*. These should be preferred by the algorithm. For more details see [9].

The BIST design process is divided into four phases:

1. Simulate several *PR* pseudo-random patterns for the CUT and determine the undetected faults (by a fault simulator).
2. Compute deterministic test patterns for these faults by an ATPG tool.
3. For the following *Det* pseudo-random LFSR patterns and the deterministic tests do the column matching.
4. Synthesize the unmatched decoder outputs by BOOM.

3 The Pseudo-Random Phase

The aim of the pseudo-random phase is to cover as many faults as possible, while keeping the test time acceptable. Two aspects play role here: the LFSR polynomial and seed and the test length. Computing a LFSR polynomial and seed in order to achieve a good fault coverage is an extremely computationally demanding problem, thus we select it at random and evaluate the effectiveness.

Selection of a LFSR and a seed might significantly influence the fault coverage. The frequency distribution of covering a particular number of faults is illustrated by Fig. 2. Here sets of 50, 100, 500 and 1000 LFSR patterns were applied to the c3540 circuit, 1000 samples for each test size (i.e., 4 curves in Fig. 2). Each LFSR and its seed were selected randomly. The distribution of the number of faults, which remained undetected, is shown. We can see that it follows the Gaussian distribution. For a low number of patterns many faults are left undetected, while also their number varies a lot. With an increase of the number of the test patterns the number of undetected faults rapidly decreases, while the variation of this number decreases as well. This means that when a high fault coverage is obtained by a long test sequence, the influence of the LFSR and seed on the fault coverage is negligible.

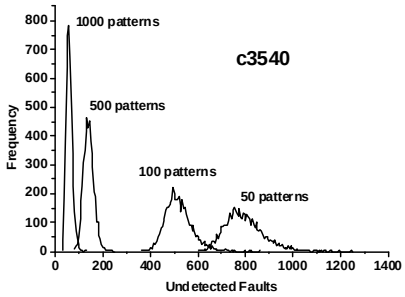

Figure 2. Pseudo-random fault coverage

The number of the covered faults as a function of the number of LFSR cycles applied to the CUT follows the well-known curve shown in Fig. 3 (for the c3540 circuit). First few vectors detect the majority of faults, and then the fault coverage increases only slightly. The total number of detectable stuck-at faults is 3428. This number was not reached even after applying 50 000 LFSR cycles.

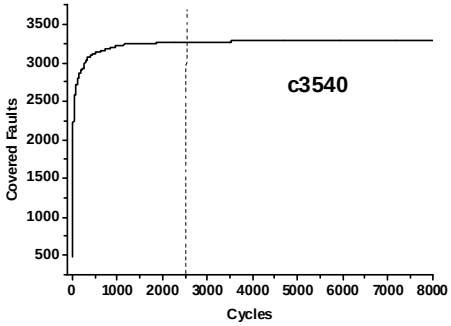

Figure 3. Fault coverage saturation curve

From these two graphs the following conclusion can be made: in order to reach a satisfactory fault coverage in the first phase, we should determine the fault coverage saturation curve for the CUT by simulation. The length of the *PR* phase can be easily observed from it. The pseudo-random phase should be stopped when the fault coverage is not improving for a given number of cycles.

This number can be freely adjusted, according to the application specific requirements (the trade-off between the test time and area overhead). Usually, we set this threshold to 1000 cycles. Thus, for the c3540 benchmark we determine *PR = 2500* cycles (see Fig. 3).

After the length of the pseudo-random phase (*PR*) is determined we repetitively select a LFSR polynomial and seed at random, and simulate the fault coverage. For each LFSR and seed the run of the *PR* cycles is simulated, to ensure a satisfactory period. From all the random LFSRs we pick out the one that covers the maximum of faults.

To illustrate the importance of properly choosing the parameters of the pseudo-random phase we have designed a BIST structure for several ISCAS benchmarks [12, 13]. We have varied the length of the pseudo-random phase, while the length of the deterministic phase was kept constant, 1000 cycles. As a fault simulator FSIM was used, as an ATPG we have used Atalanta [14]. The results are shown in Table 1. The benchmark name and the number of its inputs are shown in the first two columns. The *"PR"* column indicates the length of the pseudo-random phase, the *"UD"* column shows the number of s-a faults that were left undetected by this phase. *"vct."* gives then the number of deterministic vectors testing these faults. The *"M"* column shows the total number of column matches obtained, *"DM"* the number of direct column matches. The next column describes the complexity of the Switch and the Decoder, in terms of the gate equivalents [15]. The time needed to complete the column-matching procedure is indicated in the last column. The runtimes of the fault simulation and Boolean minimization were negligible comparing to the column-matching runtimes. The experiment was run on a PC with Athlon CPU, on 1 GHz, Windows XP.

4 The Deterministic Phase

In the deterministic phase we try to synthesize the deterministic vectors from some of the LFSR patterns that follow after the pseudo-random phase. With increasing number of LFSR patterns the chance of finding more column matches increases as well. This is due to having more freedom for selecting the LFSR vectors to be assigned to the deterministic vectors. However, the design runtime rapidly increases with the number of vectors.

This is illustrated by Table 2. Its format is retained from Table 1, the *"Det."* column indicates the length of the deterministic phase.

It can be observed that a trade-off between the test time and area overhead can be freely adjusted here too, according to the demands of the BIST designer.

The lengths of both the phases significantly influence the BIST design time as well. The design process is being sped up when increasing the length of the pseudo-random phase, since the number of deterministic vectors is being reduced this way. On the other hand, an increasing length of the deterministic phase slows down the process.

5 Comparison of the Results

We have compared our results with two state-of-the-art methods, namely the bit-fixing method [5] and the row matching method proposed in [7]. The comparison is shown in Table 3. The *"TL"* columns indicate the total length of the test, the *"GEs"* columns give the number of gate equivalents of the BIST combinational circuits. The column-matching GEs in bold indicate that our method was better than both the other methods, in terms of the complexity of the transforming combinational logic. Let us note here, that a special kind of a PRPG is used in the row-matching approach [7]. Such a circuit causes quite a large area overhead in most cases, for many XOR gates present. This overhead is not included in the table. Our method is independent on a PRPG used, in general, thus in all the cases we have used an LFSR with two XOR gates only, independently on its width. Thus, sometimes bigger area overhead of our method could be compensated by a small area of a PRPG used. The empty cells indicate that the data for the respective circuit was not available to us.

Table 3. Comparison results

	Column-matching		Bit-fixing		Row-matching	
Bench	*TL*	*GEs*	*TL*	*GEs*	*TL*	*GEs*
c880	1 K	**10.5**	1 K	27	1 K	21
c1355	2 K	15	3 K	11	2 K	0
c1908	3 K	**7.5**	4 K	12	4.5 K	8
c2670	5 K	172	5 K	121	5 K	119
c3540	5.5 K	**1.5**	4.5 K	13	4.5 K	4
c7552	8 K	586	10 K	186	8 K	297
s420	1 K	**24.5**	1 K	28	-	-
s641	4 K	15	10 K	12	10 K	6
s713	5 K	16.5	-	-	5 K	4
s838	6 K	130	10 K	37	-	-
s1196	10 K	**6**	-	-	10 K	36

6 Conclusions

We have proposed a study of the influence of the test lengths on the resulting circuitry for our mixed-mode column-matching based BIST method. The test is divided into two phases, the pseudo-random and deterministic. The lengths of both the phases might be freely adjusted, to find a trade-off between the test time and area overhead. We have shown that the length of the pseudo-random phase has a crucial impact on the result and present a methodology for choosing its length efficiently.

The length of the deterministic phase influences the result as well, however not too significantly. The impact

of the test lengths on the duration of the BIST design process is considered as well.

Our present experiments are intended to expand our conclusions to other the pseudo-random based BIST design methods.

Acknowledgement

This research was supported by a grant GA 102/04/2137 and MSM 212300014

References

[1] S. Hellebrand, 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

[2] S. Hellebrand, H. Liang, H.J. Wunderlich, "A Mixed Mode BIST Scheme Based on reseeding of Folding Counters". Proc. IEEE ITC, 2000, pp.778-784

[3] G. Kiefer, H. Vranken, E.J. Marinissen, H.J. Wunderlich, "Application of deterministic logic BIST on industrial circuits". Proc. Int. Test Conf. (ITC'00), Atlantic City, NJ, Oct. 2000, pp. 105-114.

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

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

[6] N.A. Touba, E.J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan-Based BIST". Proc. of International Test Conference, 1996, pp. 167-175

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

[8] P. Fišer, J. Hlavička, H. Kubátová, "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

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

[10] P. Fišer, J. Hlavička, "BOOM - a Heuristic Boolean Minimizer". Proc. International Conference on Computer-Aided Design ICCAD 2001, San Jose, California (USA), 4.-8.11.2001, pp. 439-442

[11] P. Fišer, J. Hlavička, "BOOM - A Heuristic Boolean Minimizer". Computers and Informatics, Vol. 22, 2003, No. 1, pp. 19-51

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

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

[14] 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

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

Table 1. Influence of the pseudo-random phase on the result

bench	*inps*	*PR*	*UD*	*vct.*	*M*	*DM*	*GEs*	*Time [s]*
c2670	233	1000	309	86	193	173	90 + 109.5 = 199.5	166
		2000	306	86	192	175	87 + 102.5 = 189.5	166
		5000	216	73	198	164	103.5 + 91 = 194.5	143
		10000	154	69	199	178	82.5 + 84 = 166.5	123
c3540	50	300	165	66	38	29	31.5 + 78 = 109.5	10.26
		500	92	42	44	29	31.5 + 25 = 56.5	3.88
		1000	36	26	49	32	27 + 1 = 28	1.02
		2000	9	9	50	41	13.5 + 0 = 13.5	0.19
		5000	1	1	50	49	1.5 + 0 = 1.5	0.02
s1196	32	200	228	104	26	25	10.5 + 100 = 110.5	5.05
		500	141	79	27	23	13.5 + 63.5 = 77	3.87
		1000	90	51	27	24	12 + 38.5 = 50.5	2.00
		2000	52	37	28	23	13.5 + 23.5 = 37	1.20
		5000	23	17	29	25	10.5 + 6.5 = 17	0.48
		10000	9	4	32	28	6 + 0 = 6	0.04

Table 2. Influence of the deterministic phase on the result

bench	*inps*	*PR*	*Det.*	*vct.*	*M*	*DM*	*GEs*	*Time [s]*
c3540	50	1000	200	26	48	31	28.5 + 5.5 = 34	0.32
			500		49	31	28.5 + 1 = 29.5	0.52
			1000		49	32	27 + 1 = 28	1.02
			2000		50	39	16.5 + 0 = 16.5	1.47
			5000		50	45	7.5 + 0 = 7.5	2.93
s1196	32	5000	200	23	27	22	15 + 10.5 = 25.5	0.17
			500		29	20	18 + 7 = 25	0.32
			1000		29	25	10.5 + 6.5 = 17	0.48
			2000		29	26	9 + 8 = 17	1.52
			5000		31	27	7.5 + 1.5 = 9	2.16
			10000		32	29	4.5 + 0 = 4.5	5.83