MIXED-MODE BIST BASED ON COLUMN MATCHING

Petr Fišer
Informatics and Computer Science, 3-rd class, full-time study
Supervisor: Hana Kubátová

Czech Technical University
Dept. of Computer Science & Engineering
Karlovo nám. 13, Prague 2, Czech Rep.

fiserp@fel.cvut.cz

Abstract. A test-per-clock BIST method for combinational or full-scan circuits is proposed. The method is based on a design of a combinational block - the decoder, transforming pseudo-random LFSR code words into deterministic test patterns. A Column-Matching algorithm to design the decoder is proposed. The Column-Matching method modified to support a mixed-mode BIST is proposed as well. Here the BIST is divided into two disjoint phases – the pseudo-random phase, where the LFSR patterns are being applied to the circuit unmodified, and the deterministic phase detecting all the yet undetected faults. This enables us to reach a high fault coverage in a short test time and with a low area overhead.

Keywords. built-in self-test, diagnostics, test generation, logic synthesis

1 Introduction

The complexity of present VLSI circuits rapidly grows. Using only external test equipment (ATE) is becoming impossible, mainly due to a huge amount of test vectors, long testing time and very expensive test equipment. Incorporating Built-in Self-Test (BIST) methods becomes inevitable. Up to now, many BIST methods were developed [1-5], all of them trying to find some trade-off between four aspects that are mutually antipodal: the fault coverage, test time, an area overhead and the BIST design time. A high fault coverage means either a long test time (exhaustive test), or a high area overhead (deterministic ROM based BIST). A pseudo-random testing established the simplest trade-off between all these three criteria. A combination of a pseudo-random and deterministic BIST is being referred to as a mixed-mode BIST. Easy-to-detect faults are tested by pseudo-random test patterns, and the deterministic patterns are generated to test the remaining, undetected faults. The bit-fixing [4] and bit-flipping [2] techniques belong to this category.

A mixed-mode BIST method is proposed here too. It is based on a Column-Matching principle, where the pseudo-random patterns are being transformed by a combinational block into deterministic patterns precomputed by an ATPG (Automatic Test Pattern Generator) tool. The test is divided into two disjoint phases: the pseudo-random one and the deterministic one. This enables to significantly reduce the decoder logic as well.

The paper is structured as follows: basic principles of our method are described in Section 2, the mixed-mode extension is shown in Section 3. Section 4 contains the experimental results, Section 5 concludes the paper.

2 Principles of the Method

The method is primarily intended for a test-per-clock BIST, thus the test patterns are applied to the primary inputs of the circuit-under-test (CUT) in parallel; one test vector is being processed in each clock cycle. However, it can be modified for a test-per-scan as well, as it was proposed in [5].

The method aims at the decrease of the area overhead that may be achieved by the simplification of the test pattern generator (TPG). Deterministic test patterns generated by some ATPG tool are used, thus the fault coverage achieved strictly depends on these patterns. No memory is used for their storage, since the memory mostly causes a big area overhead on a chip.

The test pattern generator consists of two blocks: the pseudo-random pattern generator (PRPG) and the Output Decoder, which is a combinational block transforming the PRPG patterns into deterministic tests. The PRPG is mostly constructed as a linear feedback shift register (LFSR) with an appropriate generating polynomial, or as a cellular automaton.

In the deterministic phase, vectors are synthesized from some of the LFSR patterns that follow the pseudo-random phase. To do so, the *Column-Matching* algorithm is used [10, 11].

2.1 Problem Statement

Let us have an *n*-bit PRPG running for *p* clock cycles in the deterministic phase. The code words generated by this PRPG can be described by a **C** matrix (*code matrix*) of dimensions (*p, n*). These code words are to be transformed into the test patterns pre-computed by some ATPG tool. They are described by a **T** matrix (*test matrix*). For an *r*-input CUT and the test consisting of *s* vectors the **T** matrix has dimensions (*s, r*). The rows of the matrices will be denoted as *vectors*.

The tests can be presented either in a form of deterministic patterns (minterms) or they may contain don't care values, depending on the ATPG algorithm used for the test set generation [6]. The presence of don't cares can significantly reduce the Output Decoder complexity, since they give us more freedom to select the column matches.

The output decoder logic modifies the **C** matrix vectors in order to obtain all the **T** matrix vectors. Since the proposed method is restricted to combinational circuits, the order in which the test patterns are fed to the CUT is insignificant. Thus, the **T** matrix vectors can be reordered in any way. Finding a transformation from the **C** matrix to the **T** matrix means finding a pairing of each of the *s* rows of **T** matrix with rows of the **C** matrix – thus finding a *row assignment* (see Fig. 1), i.e., to determine which **C** matrix rows will be transformed to **T** matrix rows and how. Here, the five 5-bit test vectors are to be assigned to ten 5-bit PRPG patterns.

The ***Output Decoder*** is a combinational block that converts *s n*-dimensional vectors of the **C** matrix into *s r*-dimensional vectors of the **T** matrix. The decoder is represented by a Boolean function having *n* inputs and *r* outputs, where only values of *s* terms are defined and the rest are don't cares. This Boolean function can be easily described by a truth table, where the output part corresponds to the **T** matrix, while the input part consists of *s* **C** matrix vectors assigned to the **T** matrix rows. The set of such vectors will be denoted as a *pruned **C** matrix*.

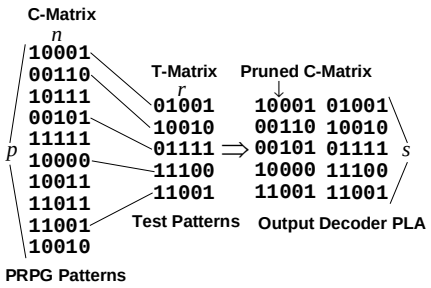
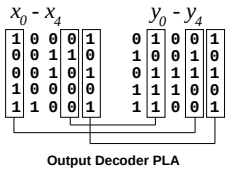


Figure 1: Assignment of the rows

2.2 The Column-Matching Method

The column-matching method is based on assigning all the **T** matrix rows to some of the **C** matrix rows so that some columns of the **T** matrix will be ***equal*** to some of the pruned **C** matrix columns. This yields no logic necessary to implement these **T** matrix columns (output variables of the decoder); they are implemented as mere wires. This idea can be extended to a *negative matching*. Since most of the D flip-flops are provided with the negated output as well, columns with all opposite values can be matches as well. An illustrative example is shown in Fig. 2. The matched columns of the pruned **C** matrix and **T** matrix from Fig. 1 are shown here. The **T** matrix column y_1 is matched with the **C** matrix column x_3 (negatively), then y_3 with x_1 (negatively) and y_4 with x_4 (positively). Thus, the outputs y_1, y_3 and y_4 are implemented without any combinational logic, while the remaining outputs have to by synthesized using some standard two-level Boolean minimization tools, like ESPRESSO [7] or BOOM [8, 9].



$y_0 = x_4' + x_1$
$\mathbf{y_1 = x_3'}$
$y_2 = x_2 x_3' + x_2' x_4'$
$\mathbf{y_3 = x_0'}$
$\mathbf{y_4 = x_4}$

Figure 2: Column matching example

2.3 Selection of Columns

Good choice of the columns to be matched is not a trivial task. The number of combinations grows exponentially with the number of columns and thus computing the optimum combination of matches is computationally infeasible. Hence, some kind of a heuristic has to be used. In practice, the number of PRPG patterns to be transformed is usually much bigger than the number of test vectors ($p >> s$). Then, almost any two columns can be matched together at the beginning. Thus, we select the columns to be matched purely at random, one by one, until there is no possibility of a match.

If there are no don't care values in the test set, the algorithm is straightforward. Each column match selected divides both the **C** and **T** matrices into a disjoint pair of two submatrices, where the respective rows can be assigned to each other. This division is being gradually performed, until no further division is possible. This basic method was presented in [10].

When the don't cares are present in the test set, the divided sets become non-disjoint. Thus the algorithm consists of two linked NP-hard problems. We have found that using the set decomposition based approach here is rather time-consuming, although using it is not impossible. An efficient heuristic based on a ***blocking matrix B*** has been proposed in [11]. The blocking matrix is a binary matrix (it contains only "0" and "1" values) of dimensions (p, s). Thus, it has as many columns as there are **T** matrix rows and as many rows as there are **C** matrix rows. The value "1" in the cell **B**[k, l] indicates that the k-th **C** matrix row may be assigned to the l-th **T** matrix row, "0" value indicates the contrary. At the beginning of the algorithm all the **B** matrix cells are filled with a "1" value, since there are no restrictions for row assignments. After the i-th **C** matrix column is matched with the j-th **T** matrix column, the **B** matrix cells [k, l] are set to "0" when the k-th input row contains in an i-th column an opposite value to the l-th output row in a j-th column. Thus, rows containing opposite values in the matched columns cannot be assigned to each other. The row assignment consists of the selection of one row from the possible ones. We have investigated several more methods, for more details see [12]. Strategy of choosing the columns to be matched is important as well. Two major techniques were proposed: ***the fast search*** and ***the thorough search*** [13].

3 Mixed-Mode Column-Matching BIST

The basic Column-Matching algorithm was later extended to a mixed-mode BIST. Most of the mixed-mode BIST techniques use some kind of transformation and switching logic accompanying the PRPG. A general structure of our mixed-mode BIST design is shown in Fig. 3. The pseudo-random code words are produced by an LSFR. Then they are transformed by the Output Decoder into deterministic vectors. The *Switching logic* selects the patterns that are to be applied to the CUT. After that the circuit's response is evaluated, usually in the multi-input shift register (MISR).

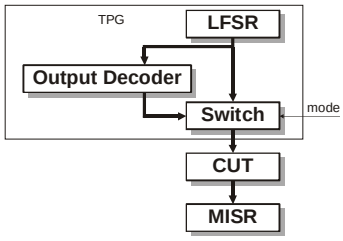


Figure 3: Mixed-mode BIST structure

The main difference between this algorithm and the competitive methods [3, 4, 5] is a separation of the pseudo-random and deterministic phases. In praxis, several initial pseudo-random vectors detect faults, but the fault detection capability of the latter ones quickly drops to zero. Thus, it could be more advantageous to run the unmodified pseudo-random phase for several clock cycles and then switch to the deterministic one at once. Then in the most general case, the switching logic consists of multiplexers. The area overhead caused by the switching logic is not big, since these multiplexers are not necessary to be present in many cases, when a modified column matching method is applied [11].

In the pseudo-random phase, all the multiplexers are set to feed the circuit with unmodified LFSR patterns. Subsequently, in the deterministic phase, all the multiplexers switch to the Decoder outputs and the modified patterns are applied to the CUT. The area overhead of the mode switching logic can be negligible, since the BIST controller pattern counter can be exploited very efficiently. For instance, when the lengths of the two phases are equal, the *mode* signal can be driven by one additional stage (D flip-flop) of a counter only. If not, just an extra comparator logic has to be present. Hence, the separation of the two BIST phases eliminates the pattern recognition logic [4].

The whole BIST design process can be divided into four consecutive phases:

1. Simulation of several *(PR)* pseudo-random patterns and determination of undetected faults.
2. Computation of the deterministic test patterns for these faults by an ATPG tool.
3. Running the column matching for the subsequent LFSR patterns and the deterministic tests.
4. Synthesis of the decoder for the unmatched outputs.

The lengths of the two phases can be freely adjusted, according to the needs of the BIST designer. The trade-offs between the test length, area overhead and the BIST design time have to be found. For more detailed discussion on this issue see [14]. The PRPG can be also augmented to achieve higher fault coverage, which yields to a significant reduction of the area overhead and design time [15, 16].

4 Experimental Results

4.1 Comparison with Other Methods

The Column-Matching method is compared with two state-of-the-art methods here, namely the bit-fixing method [3] and [5]. The comparison is shown in Table 1. The *"TL"* columns indicate the total length of the test, the *"GEs"* columns give the number of gate equivalents of the BIST combinational circuits. Let us note here that a special kind of a PRPG is used in the row-matching

approach [5]. Such circuit involves quite a large area overhead in most cases, due to many necessary XOR gates used. This overhead is not included in the table. Our method is independent on a PRPG used, thus in all the cases we have used an LFSR with two XOR gates only [15], independently on its width. The empty cells indicate that the data for the respective circuit was not available to us.

Table 1: Comparison results

	Column-matching		Bit-fixing [3]		Row-matching [5]	
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

4.2 Results for Standard Benchmarks

Since the comparison shown in Table 1 describes results for a few benchmark circuits only, a more exhaustive result table for some hard-to-test ISCAS [17, 18] and ITC'99 [19] benchmarks is presented in Table 2. The *"inps"* column indicates the number of the benchmark inputs, in the *"100% FC*" column the number of pseudo-random vectors needed to be applied to the CUT to achieve 100% fault coverage is shown, just to show the effectiveness of the method. The *"TL"* column gives the lengths of the pseudo-random and deterministic phases. The next columns show the number of column matches reached. The complexity of the switching logic is shown in the *"SW GEs*" column, the complexity of the output decoder in *"OD GEs"*. These numbers are summed together in the *"Total GEs*" column and the area overhead of the Output Decoder and Switch, with respect to the CUT GEs is shown in the *"BIST Overhead*" column. The runtime needed to complete the column-matching process is indicated in the last column. The experiments have been run on a PC with Athlon 2600 MHz processor.

Table 2: ISCAS and ITC benchmarks

Bench	*inps*	*100% FC*	*TL (PR+Det.)*	*M*	*SW GEs*	*OD GEs*	*Total GEs*	*BIST Overhead*	*Time [s]*
c2670	233	2.5 M	1 K + 1 K	193	90	109.5	199.5	19 %	166
c7552	207	> 100 M	7 K + 1 K	131	261	325	586	19 %	500
s420	34	150 K	3 K + 1 K	35	21	0	21	11 %	0.41
s641	54	200 K	500 + 500	52	21	2	23	9 %	0.47
s713	54	300 K	500 + 500	52	24	3	27	8 %	0.56
s838	67	> 100 M	1 K + 1 K	37	81	45	126	32 %	26.20
s1196	32	200 K	9 K + 1 K	32	6	0	6	1 %	0.04
s5378	214	80 K	20 K + 1 K	214	13.5	0	13.5	1 %	0.98
s9234	247	10 M	50 K + 1 K	208	163.5	156	319.5	8 %	350
s13207.1	700	100 K	10 K + 1 K	696	300	26.5	326.5	6 %	137
s15850.1	611	> 10 M	100 K + 2 K	553	306	66.5	372.5	5 %	1244
s38417	1664	> 10 M	100 K + 2 K	1503	1245	489	1734	11 %	17650
s38584.1	1464	> 1 G	100 K + 1 K	1464	165	0	165	1 %	34
b07	50	200 K	10 K + 1 K	50	24	0	24	6 %	0.5
b12	126	5 M	10 K + 1 K	118	33	34	67	7 %	25

5 Conclusions

A mixed-mode BIST method based on the *column matching* approach has been proposed. Here the pseudorandom LFSR code words are being transformed into deterministic test patterns computed by some ATPG tool. The transformation is being done by a purely combinational block.

The pseudo-random and deterministic phases are separated, which enables to reach smaller area overhead. The method is based on a design of a decoder transforming the LFSR code words into deterministic test vectors testing the hard-to-detect faults.

Acknowledgment

This research was supported by a grant GA 102/04/2137 and MSM6840770014.

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] Wunderlich, H.J., Keifer, G.: Bit-Flipping BIST, Proc. ACM/IEEE International Conference on CAD-96 (ICCAD96), San Jose, California, November 1996, pp. 337-343

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

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

[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] 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

[7] Brayton, R.K, et al.: Logic Minimization Algorithms for VLSI Synthesis, Boston, MA, Kluwer Academic Pub., 1984

[8] Fišer, P., Hlavička, J.: BOOM - A Heuristic Boolean Minimizer, Comp. & Informatics, Vol. 22, 2003, No. 1, pp. 19-51

[9] Hlavička, J., Fišer, P.: 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

[10] 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

[11] 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

[12] Fišer, P., Kubátová, H.: Survey of the Algorithms in the Column-Matching BIST Method, Proc. 10th International On-Line Testing Symposium 2004 (IOLTS'04), Madeira, Portugal, 12.-14.7.2004, pp. 181

[13] Fišer, P.: Mixed-Mode BIST Based on Column Matching, Postgraduate Study Report, Prague, CTU, Sept. 2004, 45 pp.

[14] 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

[15] Fišer, P., Kubátová, H.: Pseudorandom Testability - Study of the Effect of the Generator Type, Proc. 6th International Scientific Conference on Electronic Computers and Informatics 2004 (ECI'04), Herľany, SR, 22.-24.9.04

[16] Fišer, P. - Kubátová, H.: Improvement of the Fault Coverage of the Pseudo-Random Phase in Column Matching BIST, Proc. 31th Euromicro Symposium on Digital Systems Design (DSD'05), Porto, (Portugal), 30.8. - 3.9.05

[17] 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

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

[19] 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