Column-Matching Based BIST Design Method

Petr Fiser, Jan Hlavicka
Department of Computer Science and Engineering
Czech Technical University
Karlovo nám. 13, 121 35 Prague 2
e-mail: fiserp@fel.cvut.cz, hlavicka@fel.cvut.cz

Abstract

A new method of test-per-clock BIST design for combinational circuits is proposed. The fundamental problem of matching the PRPG outputs with the required test patterns is solved as a general design problem in the field of combinational logic. A test set generated by an ATPG is compared with the PRPG generated sequence. The solution is based on a novel search algorithm, which identifies the best matches between the pairs of columns of the two sets.

Introduction

Use of the BIST (Built-in self-test) equipment is becoming inevitable with the growth of the complexity of the VLSI devices. Presently the attention concentrates above all on the use of the so-called algorithmic test pattern generators based on PRPG (Pseudo-Random Pattern Generator), which may be a LFSR (Linear Feedback Shift Register) or a cellular array. However, the sequence of the code words generated by a PRPG practically never fulfils the requirements stated for the test sequence. Thus we have to translate the LFSR outputs into the required test patterns by some combinational logic.

In this paper we propose a test-per-clock BIST method, where the CUT is fed with test patterns pre-computed by an ATPG tool. These patterns are generated by a circuit consisting of two parts: first, pseudorandom vectors are produced by a PRPG and then these vectors are converted into the required test patterns by an additional output decoder.

Principle of the Method

Our task is to design a simple LFSR output decoder, which transforms matrices. A code matrix **C**, with the dimensions (n, p) generated by an n-bit PRPG running for p cycles, is to be transformed into a testing matrix **T** with dimensions (r, s), where r is the number of CUT inputs and s is the number of test patterns.

When testing combinational circuits, the patterns can be reordered. In other words, any vector (row) from the **T** matrix can be assigned to any vector of the **C** matrix. Moreover, the rows in the **C** matrix need not form a compact block. The excessive patterns not transformed into tests just represent idle cycles of the PRPG. Thus, finding a transformation from **C** matrix to **T** matrix means finding a *matching* of all s rows of the **T** matrix with any s rows of the **C** matrix.

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. It is represented by a Boolean function with n inputs and r outputs, where only values of s terms are defined, the rest are don't cares. This Boolean function can be described by a truth table and then processed by some two-level Boolean minimizer (ESPRESSO [1], BOOM [2, 3]) to produce a sum-of-product (SOP) form.

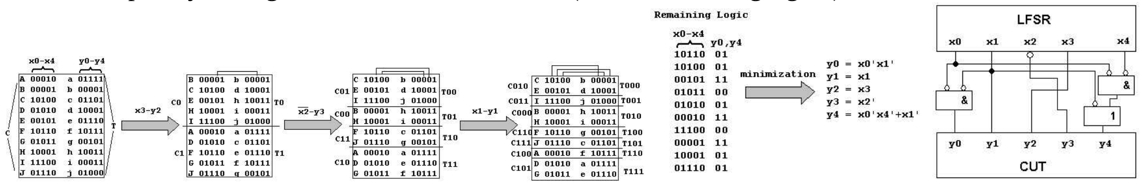
Column Matching

The assignment of the rows of T matrix to the rows of **C** matrix is of a key importance for the design of the output decoder. Our task is to find such an assignment that reduces the combinational logic of the output decoder to minimum.

The proposed method is based on the fact, that if in the final assignment i-th column of the matrix **C** is exactly the same as j-th column of the matrix **T**, there is no combinational logic required to implement j-th variable in hardware. Then the *column matching* is found. As the PRPG outputs are drawn directly from the outputs of flip-flops which normally provide also the negative value of the outputs, also the negative matching is possible – i.e., the value of the matched output variable is a complement to the value of some input variable in all care terms.

If there is $p = s$, all the PRPG vectors are assigned to the test vectors and no idle PRPG cycles are present. This will be denoted as one-to-one assignment. It is the fastest BIST method, however, the amount of logic needed to implement the output decoder is often large. Determining a column match for one column is a simple task: this match is possible if the counts of ones (and zeros) in the corresponding columns are *equal*. After finding one column match, the two matrices are decomposed into two disjoint parts: the rows with zeros and ones respectively in the corresponding columns, let them be denoted as $\mathbf{C_0}$, $\mathbf{C_1}$ and $\mathbf{T_0}$, $\mathbf{T_1}$. Then any vector from the sub-matrix $\mathbf{T_0}$ can be assigned to any vector from $\mathbf{C_0}$, and any vector from the sub-matrix $\mathbf{T_1}$ can be assigned to any vector from $\mathbf{C_1}$, but not otherwise. Finding all possible column matches consists in the successive decomposition of both matrices into systems of subsets, until no decomposition is possible. Then some row-matching method, e.g. [4], is applied to all the subsets to make the final assignment.

To illustrate the method we have chosen the c17 ISCAS benchmark [5] for its simplicity. As input we have a complete test set generated by an ATPG tool. This test set consists of 10 test patterns a-j, which are to be assigned to the vectors A-J generated by a LFSR with generating polynomial $x^5 + x^2 + 1$ seeded with the vector 00010. Our goal is to implement a BIST structure completely testing the c17 benchmark circuit (see the following figure).



One-to-one column matching example

Both methods proposed above assumed one-to-one row matching where $p = s$. In practice, it is often more advantageous to let the PRPG run more cycles than needed and pick out only several suitable vectors. Then some idle test cycles are present, however it significantly reduces the complexity of the output decoder. The column matching method is very efficiently applicable also to this case. Again, both matrices **C** and **T** are divided into two disjoint parts, while in this case their sizes are not equal; the number of vectors in each $\mathbf{C_i}$ must be *greater or equal* to the number of vectors in the corresponding $\mathbf{T_i}$. For our example, 19 LFSR cycles are needed to match *all* the columns and no additional logic is needed.

Experimental Results - ISCAS Benchmarks

The results of the column matching method are illustrated on several combinational ISCAS benchmarks. The complete test sets were generated by the APTG tool ATOM [6] for all benchmark files. An LFSR with *r* stages seeded with a random vector was used as a pseudo-random pattern generator. The following table shows the results. For each benchmark circuit the number of exact column matches obtained (out of *r*) is shown, as well as the complexity (cost) of the resulting output decoder in terms of the sum of the number of literals and the output cost.

benchmark	LFSR (*n* / *p*)	test size (*r* / *s*)	matches	cost
c1355	41 x 5000	41 x 192	8	1475
c1908	33 x 5000	33 x 210	10	2043
c432	36 x 5000	36 x 100	10	1180
c499	41 x 5000	41 x 127	9	698
c880	60 x 5000	60 x 133	10	3024

Conclusions

A new test-per-clock BIST design method for combinational circuits was described. The pseudorandom patterns are generated by a PRPG and then transformed by a combinational block into given test patterns. The method aims at simplifying this decoder as much as possible by proper assigning the test patterns to the PRPG patterns. It is based on the column matching approach, where as many outputs as possible are directly matched to the inputs, which minimizes the combinational logic. The method is applicable to one-to-one matching where the PRPG runs only the necessary number of cycles, as well as to the situation where idle PRPG cycles are inserted. The method was tested on a set of combinational ISCAS benchmarks.

Acknowledgment

This research was in part supported by grant #102/01/0566 Built-in Self-Test Equipment Optimization Methods in Integrated Circuits of the Czech Grant Agency (GACR).

References

[1] Brayton, R.K., et al.: Logic minimization algorithms for VLSI synthesis. Boston, MA, Kluwer Academic Publishers, 1984
[2] Hlavička, J. - Fišer, P.: BOOM - a Heuristic Boolean Minimizer, Proc. ICCAD-2001, San Jose, Cal. (USA), 4-8.11.2001, pp. 439-442
[3] http://service.felk.cvut.cz/vlsi/prj/BOOM/
[4] Chatterjee, M. - Pradhan, D.J.: A novel pattern generator for near-perfect fault coverage. Proc. of VLSI Test Symp. 1995, pp. 417-425
[5] 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
[6] http://www.crhc.uiuc.edu/IGATE/