Reduction of Test Vectors Volume by Means of Gate-Level Reconfiguration

Lukas Starecek, Lukas Sekanina and Zdenek Kotasek
Faculty of Information Technology
Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic
Email: starecek@fit.vutbr.cz, sekanina@fit.vutbr.cz, kotasek@fit.vutbr.cz

***Abstract*— In this paper, a new concept which allows the reduction of test vectors volume is presented. The concept is based on reconfiguration of some gates of circuit under test. Instead of testing the original circuit, a circuit which has the same topology (but some of its gate functions are reconfigured) is actually tested. Two possible implementations of the reconfiguration are investigated. Preliminary experiments indicate that test length can be reduced to approx. 70% of its initial value while the increase in transistors is moderate.**

I. Introduction

There are many reasons forcing designers to reduce the number of test vectors needed to test a digital circuit and thus reduce the test application time. Automatic test pattern generator tools have been used in this area for a long time. High-quality test sequences generated by these tools, in addition to the usage of full/partial scan techniques and other methods allowed designers to reduce the test application time significantly [6], [12], [3], [5]. Another reason exists for reducing the number of test vectors – namely permanently increasing number of embedded systems implementations. In these systems, power consumption is important not only during their functional mode but during the test application as well. Numerous methodologies for the reduction of power consumption during test application were published, usually aiming at the reduction of the number of transitions during test application [4]. From this point of view, any reduction of test data volume can be seen as a contribution to this objective. Other activities the objective of which is the reduction of test data volume, can be also recognized in the field of test data compaction [2], [7].

The aim of this paper is to present a new concept which allows to reduce the number of test vectors. The approach is built on experiments that we performed with simple combinational circuits and FlexTest, a commercial automatic test pattern generator (ATPG) tool. For a given circuit C, ATPG can generate k-vector test sequence T leading to $p\%$ fault coverage where k and p depends on structure and properties of C, fault model used and user requirements in particular.

We will show that if logic function of some gates of C can be changed then a much shorter test T' can be generated, i.e. k can be significantly decreased (tens of percent, depending on circuit) simultaneously with keeping p unchanged or even increased. In fact, the circuit is reconfigured before test is applied. The first configuration is used for normal operation of the circuit. The second configuration is used during test application to reduce test application time. This observation will be demonstrated in Section II on a simple combinational circuit. It is important to note that circuit topology remains unchanged during this reconfiguration. In Section II (and also in Section V) we will also deal with the major objection which can be formulated against this approach: as test is not generated for the original circuit it says nothing about the original circuit.

The proposed method will be defined in Section III. In particular, we will describe a heuristic algorithm enabling to identify those gates of original circuits which should be reconfigured in order to reduce test application time. The method will be evaluated using some circuits from the ISCAS85 benchmark suite in Section IV. Section V discusses possible implementations of the reconfiguration mechanism. Conclusions are given in Section VI.

II. Preliminary Study

Figure 1a shows c17 circuit which consists of six two-input NAND gates. We used FlexTest to derive the test with 100% fault coverage. A standard stuck-at-fault model was utilized for AMI 1.2 micron technology[1]. The test consists of nine test vectors.

After some experiments in which we evaluated various replacements of gates, logic functions of three gates of this circuit were modified as shown in Figure 1b. Again, we used FlexTest to derive the test with 100% fault coverage. The new test contains only five test vectors which represents a significant reduction.

There are several problems related to this method:

- How to recognize those gates which have to be reconfigured and how to find their new logic functions in order to decrease test length as much as possible. This problem will be addressed in Section III.
- How to implement the reconfiguration. The reconfiguration should not influence the result of test and its implementation should cost small area on a chip. This problem will be addressed in Section V.

[1] http://germanium.cs.wustl.edu/HEP/ADK/HTMLdatabook/AMI12databook.htm

978-1-4244-2277-7/08/$25.00 ©2008 IEEE

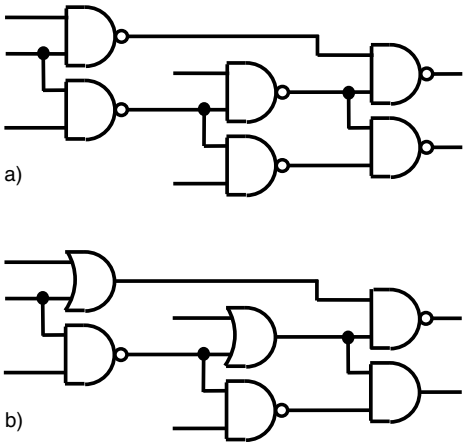

Fig. 1. C17 circuit (a) and its modification before test is applied (b)

- How to ensure that by performing the reconfiguration and testing the reconfigured circuit, the original circuit is also tested properly.

A possible answer to the third item which justifies proposed method is as follows. The basic assumption of the proposed method is that ATPG tools do not inspect internal failures of gates because only the circuit structure is tested in which gates are considered as black boxes. In general, test generation methods and tools are based on the assumption that failures which arise in the internal structure of a component are propagated to component output as either stuck-at-zero or stuck-at-one values. This is the principle which is widely accepted and expresses the relation between internal and external failures of an electronic component (e. g. a gate) and thus allows to generate test vectors for its external nodes only.

It is a fact, that one possibility how to generate a test is based on developing a test sequence for stuck-at-zero and stuck-at-one faults on the internal connections of the component (this is the case of Mentor Graphics FlexTest tool). It is evident that these faults are then the same for both versions of the circuit (i. e. the original circuit and the modified one) as far as internal connections and I/O ports of both of them are concerned. Thus, it can be stated that the test of one of them covers the faults of the other one. Problems may appear if the internal implementations of both types of gates differ in some way. Then, it might happen that in one mode a gate operates properly while the other mode is damaged and its operation is not correct. This is a situation which deserves to be investigated and analyzed. Section V will give more details relevant for a particular implementation of reconfigurable gates.

III. Proposed Method: Selection of Gates for Reconfiguration

Let Γ denote a set of available logic gates. Given a circuit which performs f_1 function and whose topology is represented by graph $G^{(1)} = (V, E, \varepsilon)$, $\varepsilon : V \rightarrow \Gamma$, where V is set of nodes, E is a set of interconnections and ε assigns a logic function to every node, the problem targeted in this paper is to find φ which constitutes $G = (V, E, \varphi)$, $\varphi : V \rightarrow \Gamma$ such that G can be tested using fewer test vectors than $G^{(1)}$; however, still providing at least the same fault coverage. In addition, we require that the gate replacements do not change the number of inputs and outputs of nodes in V (for example, a two-input/one-output gate can be replaced only by a two-input/one-output gate).

As the number of possible replacements is $|V|^r$, where r is the average number of possible replacements for a particular gate, the exhaustive search for the optimal replacement is intractable. Hence a heuristic search method developed on the basis of the hill-climbing algorithm is proposed to solve this problem. The search algorithm can be described using the following semi-formal description. For benchmark circuit C which contains gates from $\Gamma = \{g_1, \ldots g_n\}$, the procedure is as follows:

1) Set the best solution $B := C$.
2) Set current solution $X := C$.
3) Create an empty set of modified gates $S := \oslash$.
4) For X, find the minimum number of test vectors, $N^{min}(X)$, which ensures $p\%$ fault coverage.
5) For each gate g_i which is in X but not in S, replace it by all gates from Γ (one by another) that have the same number of inputs and outputs. For every replacement find the minimum number of test vectors $N^{min}(X^{ij})$ which ensures at least $p\%$ fault coverage[2].
6) If such j exists for which $N^{min}(X^{ij}) \leq N^{min}(B)$ then $B := X^{ij}$.
7) If such j exists for which $N^{min}(X^{ij}) \leq N^{min}(X)$ then run this algorithm recursively from step 4 with setting $X := X^{ij}$ and insert g_i to S. When this recursion is terminated, remove g_i from S.
8) Repeat 4 – 7 steps until all gates of C are checked for replacement.

The solution is represented by circuit B. In steps 6 and 7, we do not compare only test length, but also the number of reconfigured gates. Hence, if two circuits have the same test length (for a given fault coverage), the search proceeds from the circuit which requires fewer gates which differ from C.

IV. Experimental Results

For evaluation of proposed method we used c432, c1355, c2670 and c6288 from ISCAS85 suite [1]. Γ contains 56 standard gates (with up to 4 inputs) which are also supported by AMIS library. In addition, 7 gates (with up to 8 inputs) were included to Γ to cover all the gates used in ISCAS85 circuits. FlexTest tool was used to calculate test length. For each circuit the algorithm was executed for 4 days on an Opteron 2220 processor. In some cases, the computation was not terminated within this time limit. But for example, the computation took 7 min 31 sec for c17 circuit. The results are as follows:

[2] X^{ij} means circuit X whose i-th gate was changed to operate as j-th gate of Γ.

- The most interesting reduction was obtained for c6288 circuit (16-bit multiplier). Only 11 out of its 2416 gates have to be reconfigured to reduce test length by 21.7% while fault coverage remains unchanged and the number of transistors is increased by less than 1% (estimated). Fault coverage remains unchanged – 99.56%.
- In case of c432 circuit, 27 out of 160 gates have to be reconfigured to reduce 102 test vectors to 57 test vectors (i.e. the reduction is 44.1%). The number of transistors increased by approx. 37%. Fault coverage increased from 99.24% to 99.82%.
- In case of c2670 circuit, 43 out of 1269 gates have to be reconfigured to reduce 189 test vectors to 124 test vectors (i.e. the reduction is 34.4%). The number of transistors increased by approx. 7%. Fault coverage increased from 95.74% to 96.56%.
- No reduction was obtained for c1355 circuit (32b SEC).

These preliminary experiments indicate that test length can be reduced to approx. 70% of its initial value. Also, the relative area overhead decreases with growing test circuits. How is the area overhead calculated? We firstly calculated the number of transistors of original circuits. Then, we calculated the number of transistors of modified circuits (assuming that they contain ordinary gates, i.e. no reconfigurable gates are used). The number of transistors is given according to AMIS library. Finally, we estimated the number of transistors for circuits which contain reconfigurable gates that allow the change of logic function before test is performed. The number of transistors of a reconfigurable gate is estimated as

$$\alpha = d + max(a^{tr}, b^{tr}), \quad (1)$$

where d is a constant and a^{tr} (resp. b^{tr}) is the number of transistors of original (resp. modified) gate. For example, $d = 6$ is considered for conducted experiments.

V. Possible Physical Implementations

We investigated two approaches to the implementation of the reconfiguration mechanism: multiplexing of logic functions and polymorphic gates.

A. Reconfiguration by Multiplexing

Figure 2 shows an example of reconfigurable logic function NAND/NOR which could be used to implement reconfiguration for proposed circuits. Logic function is selected using a control logic signal (Sel), the third input of the circuit. In comparison with a standard CMOS implementation of NAND (or NOR) gate, this implementation uses 6 additional transistors. In order to reconfigure whole circuit before test is applied, control signals (Sel) of all reconfigurable gates must be connected together to form a global reconfiguration signal (GRS) which is set/reset by a controller.

However, reconfigurable circuits controlled by GRS exhibit one important disadvantage for our method. In fact, the existence of GRS signal should be reflected by ATPG during test generation which does not happen. In our future research we will investigate whether this fact really influences the set of test vectors generated by FlexTest.

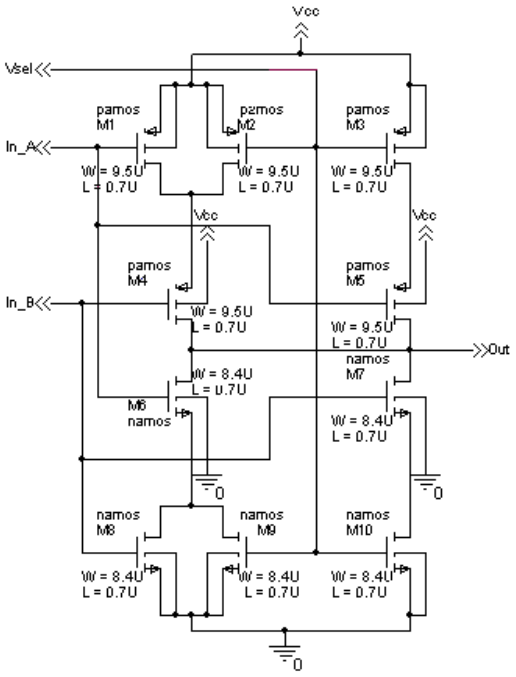

Fig. 2. Optimized transistor-level implementation of NAND/NOR gate which is controlled by Sel signal [8]

B. Reconfiguration by Polymorphic Gates

Polymorphic gates [9], [11], [10] exhibit another option to implement reconfiguration for the proposed method. Polymorphic gates are configurable; their functionality depends on some external factors, for example, on the level of the power supply voltage (Vdd), temperature, light etc. For instance, NAND/NOR gate which operates as NAND when Vdd=3.3V and as NOR when Vdd=1.8V is the most famous example [9]. This gate consists of 6 transistors only. It was fabricated in a 0.5-micron CMOS technology.

In particular, polymorphic gates controlled by Vdd could perform the reconfiguration for proposed method. Then, the polymorphic circuit works as follows: For one level of Vdd the circuit operates in the normal mode. Once the level of Vdd is changed, the circuit is switched to test mode which allows to perform test using reduced number of test vectors. This approach does not require any additional signal such as GRS and thus seems to be ideal for our method. Unfortunately, not all polymorphic gates which could be possibly used in our experiments exist now. Hence, the cost of implementation can be only estimated. The value of constant d (i.e. $d = 6$ in equation 1) was determined on the basis of the analysis performed on existing polymorphic gates.

C. Possible Problems with Test Application

During the test application, selected gates are reconfigured and the following faulty situations can be identified:

(A) If the test fails due to a fault on connections between gates, then no problem arises. The test generated by ATPG

covers these faults and during test application the faults will be properly detected by the test.

(B) The test is applied to a circuit which contains reconfigurable gates that are based on multiplexing logic functions (Figure 2). As each gate contains two elements (for example, NAND and NOR element), two situations can be recognized:

- A fault is propagated onto the gate output from the internal element which is active in test mode, the element is faulty – then the gate is identified as a faulty one although in functional mode it operates properly – an incorrect conclusion.
- An element which is active in functional mode and is faulty (so that it is not able to cover the required function) is replaced by the other element during the test application mode which operates correctly – the output of this gate is propagated to the element output, then the element is seen as properly working – an incorrect conclusion.

Another drawback of this approach can be certainly seen in the way in which the element is controlled, i. e. by an external signal. In fact, the existence of this signal should be reflected by ATPG during test generation which does not happen because the control signals were connected to the selected elements after the analysis of the test sequence generated by ATPG was done.

(C) Both logic functions are provided by an indifferentiable circuit (such as the polymorphic gate) whose components are active in normal as well as test mode. Hence, for this approach it holds that the drawbacks described in B are not valid.

To summarize, the paper deals with research in the environment described under B. The objective of our research was to verify whether the test vector set can be reduced if gates in the circuit can possibly perform two functions. For this purpose, the use of reconfigurable gates as described in this paper was justified and the results gained can be seen as convincing.

As a 100% fault coverage is not achievable for complex real-world circuits, designers do respect that some faults can remain unrecognized. Although the proposed method can also leave some faults unrecognizable, it allows the reduction of test vectors volume with simultaneous high fault coverage. This idea was verified and can be seen as the main contribution of this paper.

VI. Conclusions

In this paper, a new concept which allows the reduction of test vectors volume was presented. The concept is based on reconfiguration of some gates of circuit under test. We have shown that this circuit modification is not recognizable for commonly used ATPG tools when a commonly used fault model is utilized. Two possible implementations of the reconfiguration were investigated. The usage of polymorphic gates seems to be more advantageous than the usage of reconfigurable gates controlled by a logic signal. Preliminary experiments indicate that test length can be reduced to approx. 70% of its initial value while the increase in transistors is moderate. The method presented in this paper can be combined with various existing methodologies which result in test application time reduction. The method can be classified as "test generation strategies", which when combined with "test application strategies" can reduce both test application time and power consumption during its application.

Acknowledgements

This work was partially supported by the Grant Agency of the Czech Republic under contract No. 102/06/0599 *Methods of polymorphic digital circuit design* and the Research Plan No. MSM 0021630528 – *Security-Oriented Research in Information Technology*.

References

[1] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target simulator in Fortran. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*, pages 695–698, Kyoto, Japan, 1985.

[2] S. R. Das, C. V. Ramamoorthy, M. H. Assaf, E. M. Petriu, J. Wen-Ben, and M. Sahinoglu. Fault simulation and response compaction in full scan circuits using HOPE. *IEEE Transactions on Instrumentation and Measurement*, 54(6):2310–2328, 2005.

[3] A. Efthymiou, J. Bainbridge, and D. A. Edwards. Test pattern generation and partial-scan methodology for an asynchronous soc interconnect. *IEEE Trans. VLSI Syst.*, 13(12):1384–1393, 2005.

[4] J. Lee and N. A. Touba. Low power test data compression based on lfsr reseeding. In *22nd IEEE International Conference on Computer Design: VLSI in Computers & Processors (ICCD 2004)*, pages 180–185. IEEE Computer Society, 2004.

[5] Y. Makris and A. Orailoglu. Property-based testability analysis for hierarchical rtl designs. In *Proceedings of IEEE ICECS99, 6th IEEE International Conference on Electronics, Circuits and Systems*, pages 1089–1092. IEEE Computer Society, 1999.

[6] S. Park. A partial scan design unifying structural analysis and testabilities. *Int. J. Electronics*, 88(12):1237–1245, 2001.

[7] I. Pomeranz and S. M. Reddy. Static test compaction for multiple full-scan circuits. In *21st International Conference on Computer Design (ICCD 2003), VLSI in Computers and Processors*, pages 393–396. IEEE Computer Society, 2003.

[8] L. Starecek, L. Sekanina, Z. Gajda, Z. Kotasek, R. Prokop, and V. Musil. On properties and utilization of some polymorphic gates. In *Proc. of 6th Electronic Circuits and Systems Conference*, pages 77–81. Fakulta informatiky a informacnych technologii, Bratislava, 2007.

[9] A. Stoica, R. Zebulum, X. Guo, D. Keymeulen, I. Ferguson, and V. Duong. Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration. *IEE Proc. Comp. Digit. Tech.*, 151(4):295–300, 2004.

[10] A. Stoica, R. S. Zebulum, and D. Keymeulen. Polymorphic electronics. In *Proc. of Evolvable Systems: From Biology to Hardware Conference*, volume 2210 of *LNCS*, pages 291–302. Springer, 2001.

[11] A. Stoica, R. S. Zebulum, D. Keymeulen, and J. Lohn. On polymorphic circuits and their design using evolutionary algorithms. In *Proc. of IASTED International Conference on Applied Informatics AI2002*, Insbruck, Austria, 2002.

[12] D. Xiang and J. H. Patel. Partial scan design based on circuit state information and functional analysis. *IEEE Trans. Computers*, 53(3):276–287, 2004.