Journal of ELECTRICAL ENGINEERING, VOL. 58, NO. 3, 2007, 121–127

DETERMINISTIC TEST PATTERN GENERATOR DESIGN WITH GENETIC ALGORITHM APPROACH

Gregor Papa [*] — **Tomasz Garbolino** [**]
Franc Novak [*] — **Andrzej Hławiczka** [**]

The paper presents an automatic technique for structure optimization of a deterministic test pattern generator (TPG). The TPG is composed of a linear register and a non-linear combinational function that can invert any bit in the generated patterns. Consequently, any arbitrary test sequence can be produced. This kind of a TPG is suitable for on-line built-in self-test (BIST) implementations where a set of deterministic test patterns is required. In order to reduce the gate count of the BIST structure a genetic algorithm (GA) is employed. In contrast to conventional approaches, a GA concurrently optimizes multiple parameters that influence the final solution. Experimental results on ISCAS benchmarks demonstrate the efficiency of the approach.

K e y w o r d s: test pattern generator, design, evolutionary technique, genetic algorithm, optimization

1 INTRODUCTION

Testing of integrated circuits and systems has nowadays become a difficult problem for which conventional test approaches often prove to be inadequate. Part of the difficulties arises from the fact that the number of transistors in a chip increases faster than the pin count and consequently internal chip modules become increasingly difficult to access. Limited number of I/O pins represents a bottleneck in testing of complex embedded cores where transfers of large amounts of test patterns and test results between the automatic test equipment (ATE) and the unit-under-test (UUT) are required, [1]. One of the alternative solutions is to implement a built-in self-test (BIST) of the UUT, [2], with on-chip test pattern generation (TPG) and on-chip output response analysis logic. In this way, communication with external ATE is reduced to test initiation and transfer of test results [3]. The approach, however, also has some disadvantages. BIST implementation inevitably leads to area overhead, which typically results in performance penalties due to longer signal routing paths resulting from the inclusion of the BIST circuitry in the design. Minimization of the BIST logic is one of the commonly addressed problems in practice.

In the paper an approach for the generation of deterministic TPG logic based on a Linear Feedback Shift Register (LFSR) composed of D-type and T-type flip-flops is described [4]. The use of LFSR for TPG eliminates the need of a ROM for storing the seeds since the LFSR itself jumps from a state to the next required state (seed) by inverting the logic value of some of the bits of its next state. The approach for constructing the proper LFSR employs a genetic algorithm (GA) to find an acceptable practical solution in a large space of possible LFSR implementations. In the area of TPG, genetic algorithms have mainly been used for the derivation of test pattern sets for target UUTs [5], [6]. As for the synthesis of the TPG logic for actual generation of the derived test patters, GA approach has been used for the solutions based on cellular automata [7].

The work reported in this paper was motivated by the need of deterministic test pattern generation for the on line BIST structure composed of idle functional units and registers, originally proposed in [8]. In this approach, functional units and registers that are not used for the computations of the target application during individual time slots are organized into a structure that is continuously tested in parallel with normal system operation. Normally, pseudo-random test vectors can be employed for such on-line self-test. In critical applications, where low fault latency is required, test pattern generators (TPG) that generate deterministic test sequence are needed.

The rest of the paper is organized as follows: in Section 2 we describe a TPG structure, and give an example of area minimization through the modification of TPG structure and its test vectors; in Section 3 we describe the GA and the work of its operators; in Section 4 we present the evaluation tool, and its methodology used in our optimization procedure; in Section 5 we describe the whole optimization process and evaluate it; and in Section 6 we draw the conclusions.

[*] Computer Systems Department, Jožef Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia, E-mail: gregor.papa@ijs.si, franc.novak@ijs.si
[**] Institute of electronics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland, E-mail: tomasz.garbolino@polsl.pl, hlawicz@boss.iele.polsl.gliwice.pl

ISSN 1335-3632

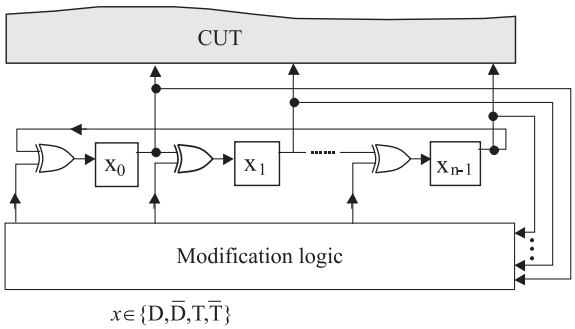
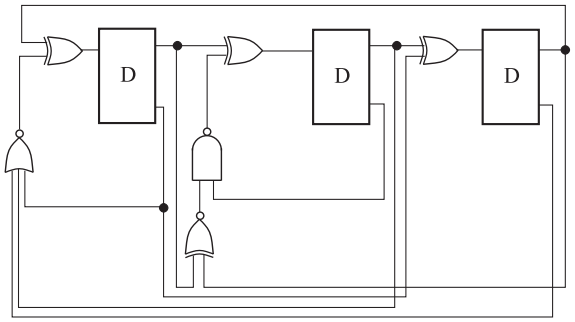


Fig. 1. Test pattern generator structure.

a)

1.	**1**	**0**	**0**
2.	**0**	**0**	**0**
3.	**0**	**0**	**1**
4.	**1**	**1**	**1**
5.	**1**	**0**	**1**
6.	**0**	**1**	**0**

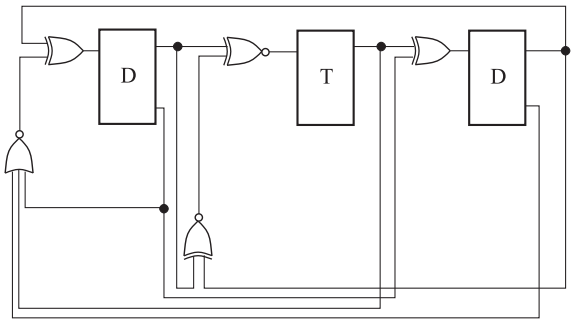


b)

Fig. 2. a) The six 3-bit test vectors, and b) the structure of the TPG for initial configuration.

a)

1.	**1**	**0**	**0**
2.	**0**	**0**	**0**
3.	**0**	**0**	**1**
4.	**1**	**1**	**1**
5.	**1**	**0**	**1**
6.	**0**	**1**	**0**



b)

Fig. 3. a) The six 3-bit test vectors, and b) the structure of the TPG for modified flip-flop type.

2 PROPOSED TPG STRUCTURE

The overall structure of the proposed n bit test pattern generator is presented in Fig. 1. It is composed of a Multiple-Input Signature Register (MISR) and a modification logic. The MISR has a form of a ring that is composed of n flip-flops with either active high or active low inputs. Any flip-flop of the MISR can be of T type or D type. Each flip-flop (D or T) can also have inverter on their input (denoted as $\overline{D}$ or $\overline{T}$). Thus, the register may have $4n$ different structures. The inputs of the MISR are controlled by the modification logic. The outputs of the MISR are fed back to the modification logic, which is a simple combinational logic and acts like a decoder.

The modification logic allows that in the subsequent clock cycles the contents of the MISR assume the values specified by the target test pattern set. Hence MISR and the modification logic are application specific: they are synthesized according to the required test pattern set.

The parameter that is particularly important in the case of deterministic test pattern generators is the area overhead. Important factors influencing the area of a TPG are:

- the structure of each MISR stage,
- the order of the test patterns in a test sequence,
- the bit-order of the test patterns.

The first property influences the complexity of both the MISR and the modification logic while the remaining two impact the area of the modification logic only. These relationships are illustrated by the following optimization example.

2.1 Initial structure and test vectors

Let us assume that the following test set composed of six 3-bit vectors (Fig. 2a) is to be produced by the TPG. The resulting structure of the TPG consists of D-type flip-flops in all stages of the MISR (Fig. 2b). Assuming that all the flip-flops are scannable and have asynchronous reset, the total area of the TPG simulated in AMS 0.35 μm technology is 1821 μm^2.

2.2 Flip-flop type replacement

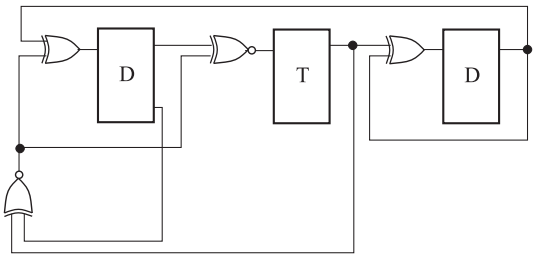
Suppose that we replace the second flip-flop with a T-type flip-flop having active low input (Fig. 3b). Since the standard cell library of the AMS 0.35 μm technology does not contain a T-type flip-flop with inverted input, the negation is implemented by replacing the XOR gate with an XNOR. The total area of the TPG is 1784 μm^2. Although a T-type flip-flop is more expensive than a D-type in terms of area, the reduction of the TPG area was achieved due to the simplified modification logic.

2.3 Column permutation

Further decrease in the area of the TPG can be obtained by permutation of columns of the test pattern sequence (*ie*, by simultaneous permutation of bits in all

a)

1.	**1**	**0**	**0**
2.	**0**	**0**	**0**
3.	**0**	**1**	**0**
4.	**1**	**1**	**1**
5.	**1**	**1**	**0**
6.	**0**	**0**	**1**

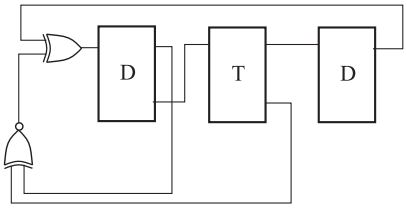


b)

Fig. 4. a) The six 3-bit test vectors, and b) the structure of the TPG for permutated columns.

a)

1.	**1**	**0**	**0**
2.	**0**	**0**	**0**
3.	**0**	**1**	**0**
4.	**0**	**0**	**1**
5.	**1**	**1**	**0**
6.	**1**	**1**	**1**



b)

Fig. 5. a) The six 3-bit test vectors, and b) the structure of the TPG for permutated test patterns.

test patterns). If we permute the 2nd and the 3rd column in the test sequence (Fig. 4a), the TPG is simplified to the structure shown in Fig. 4b. The area of the TPG is $1657\,\mu\text{m}^2$.

2.4 Vectors permutation

Another possibility is to permute test patterns in the test sequence. If we exchange the test patterns 4 and 6 in the test sequence (Fig. 5a) we get the modified sequence. The TPG is further simplified to the structure shown in Fig. 5b and with the area of $1421\,\mu\text{m}^2$.

Minimization of TPG area is a complex problem that can be addressed by different optimization approaches.

3 GENETIC ALGORITHM

We used GA optimization because of its intrinsic parallelism that allows working from a broad database of solutions in the search space simultaneously, climbing many peaks in parallel. Thus, the risk of converging to a local optimum is relatively low. Besides, promising results of our research work obtained in other optimization problem areas [9], [10], [11] encouraged us to consider GA approach as one of the possible alternatives in TPG synthesis optimization.

3.1 Encoding

The parameters of the TPG to be optimized were coded as integer values into three different chromosomes. With those three chromosomes we concurrently optimized the structure of the TPG, the order of the test patterns, and the bit order of test patterns. The first chromosome, which encodes the structure of n-bit TPG, looks like

$$C_1 = t_1 i_1 t_2 i_2 \ldots t_n i_n \tag{1}$$

where t_j $(j = 1, 2, \ldots, n)$ represents the type of the flip-flop (either D or T) and i_j $(j = 1, 2, \ldots, n)$ represents the presence of the inverter on the input of the j-th flip-flop.

The second and third chromosome, which encode the order of the test patterns, and the bit order of test patterns, look like

$$C_2 = a_1 a_2 \ldots a_m \tag{2}$$

where m is the number of test vectors and a_j $(j = 1, 2, \ldots, m)$ is the label number of the test pattern from the pattern list, and

$$C_3 = b_1 b_2 \ldots b_k \tag{3}$$

where k is the number of flip-flops in the structure and b_j $(j = 1, 2, \ldots, k)$ is the label number of the bit order of test patterns.

3.2 Initial population

The initial population consisted of n chromosomes — reproductions of the initial structure. To ensure versatile population some chromosomes were mirrored. The values on the left side (beginning) of the chromosome were mirrored to the right side (ending), while the values from the right side were mirrored to the left side; either type of registers or inverter presence or both values were mirrored in case of the first chromosome type. In case of other two chromosomes, their initial reproduction included mirroring of orders between the beginning and the ending positions.

3.3 Genetic operators

In the selection process most fit chromosomes were selected for reproduction. The elitism strategy was applied through the substitution of the least-fit members with the equal number of those best-ranked.

In a two-point crossover chromosome mates were chosen randomly and with a probability p_c all values between

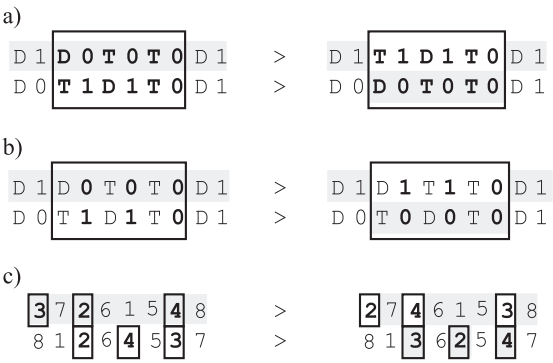


Fig. 6. Crossover operator: a) register type and inverter presence as one indivisible block, b) only the values of inverters are swapped, and c) interchange of positions that store the ordered numbers.

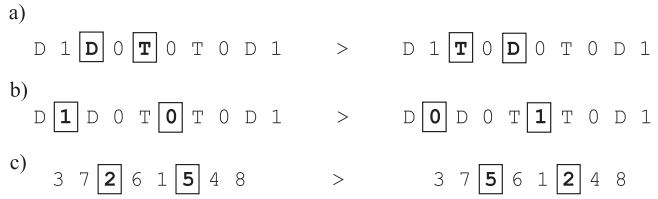


Fig. 7. Mutation operator: a) only flip-flop types are changed, b) inverter presences are changed, and c) pattern orders and test bit streams order are changed.

two randomly chosen positions were swapped which led to the two new solutions. For example, considering two strings with crossover points on positions 1 and 4 see Fig. 6a. In the first chromosome, register type and inverter presence are considered as one indivisible block (*ie*, two values for one position in the chromosome).

Moreover, with some probability p_r only the values of inverters in that swapping range were swapped. See Fig. 6b.

The crossover in case of test patterns order and bit-order of the test patterns was performed with the interchange of positions that store the ordered numbers within the range; for example within the range $[2, 4]$ see Fig. 6c (positions with orders 3, 2, and 4 in the first chromosome are interchanged with orders 2, 4, and 3 of the second chromosome).

In the mutation process each value of the string mutated with a probability p_m. However, since a high mutation rate resulted in a random walk through the GA search space, p_m had to be chosen to be somewhat low. Three different types of mutation were applied (see Fig. 7):

- D/T-type change, where only flip-flop types were changed with some probability on each position in the chromosome (Fig. 7a);
- inverter change, where inverter presences were changed with some probability on each position in the chromosome (Fig. 7b);
- order change, where pattern orders and test bit streams order were changed — after choosing the positions to be modified their values are interchanged (Fig. 7c).

With a possibility of annealing the mutation rate, p_m was a variable mutation probability. It was decreasing linearly with each new population. Since each new population generally was more fit than the previous one, we overcome a possible disruptive effect of mutation at the late stages of the optimization, and speed up the convergence of the GA to the optimal solution in the final optimization stages.

3.4 Fitness evaluation

After the recombination operators modified the solutions, the whole new population was ready to be evaluated. Here, the external evaluation tool (see Section 4) was used to evaluate each new string created by the GA.

3.5 Termination criteria

In our implementation the GA operated repetitively, with an idea that, on average, solutions of the population defining the current generation had to be as good (or better) at improving the fitness function as those of the previous generation. When a certain number of populations had been generated and evaluated, the system was assumed to be in a non-converging state. The fittest member within all generations was taken to be the solution of the design problem.

4 EVALUATION TOOL

Operation of the jth cell of the TPG register during one clock cycle can be expressed by the following equation:

$$\begin{aligned} Q_j &= t_j q_j \oplus q_{j-1} \oplus i_j \oplus f_j \\ Q_1 &= t_1 q_1 \oplus q_n \oplus i_1 \oplus f_1 \end{aligned} \tag{4}$$

where q_{j-1} is the current state of the cell number $j-1$, q_j is the current state of the j-th cell, Q_j is the next state of the j-th cell, t_j is the coefficient determining type of the flip-flop in the j-th cell, *ie*, 0 for D-type flip-flop, and 1 for T-type flip-flop, i_j is the coefficient determining whether there is an inverter at the input of the flip-flop in the j-th cell, *ie*, 0 for absence of inverter, and 1 for presence of inverter, and f_j is the value of the j-th output of the modification logic. Thus, the value of the j-th output of the modification logic is:

$$\begin{aligned} f_{j,} &= t_j q_j \oplus q_{j-1} \oplus i_j \oplus Q_j \\ f_1 &= t_1 q_1 \oplus q_n \oplus i_1 \oplus Q_1 \,. \end{aligned} \tag{5}$$

On the basis of these equations one can derive values of the outputs of the modification logic for each vector but last in the test sequence. In that way ON-set and OFF-set of the modification logic are defined.

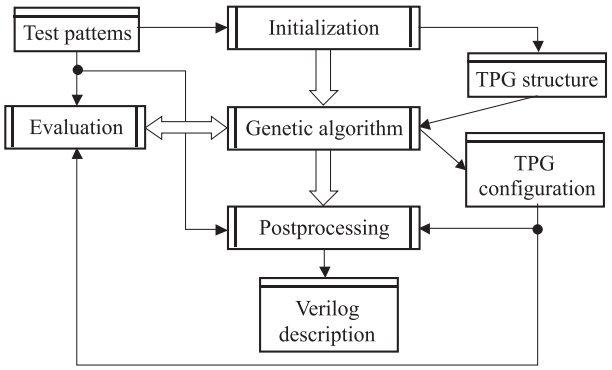


Fig. 8. The optimization process

Table 1. Results of modification logic size (in total cost by ESPRESSO)

	test pattern width	number of test patterns	initial TPG	optimized TPG	improvement in %
c432	36	27	348	280	19.5
c499	41	52	312	164	47.4
c880	60	16	536	402	25.0
c1355	41	84	584	488	16.4
c1908	33	106	2077	1840	11.4
c6288	11	12	74	49	33.8

Table 2. Comparison with results achieved in [14]

	complexity of TPG obtained by column matching	complexity of the proposed TPG optimized by GA approach
c432	0.33	0.29
c499	0.13	0.08
c880	0.38	0.35
c1355	0.19	0.14
c1908	0.29	0.53
c6288	–	0.44

Table 3. Comparison with results achieved in [15]

	area_per_bit of the TPG in [15]	area_per_bit of the proposed TPG optimized by GA approach
c432	0.22	0.11
c499	0.21	0.10
c880	0.19	0.29
c1355	0.20	0.09
c1908	0.19	0.32
c6288	0.24	0.54

They are further minimized and the cost of the logic is estimated. We use Espresso software [12] for Boolean minimization of the modification logic and its approximate cost evaluation. Espresso takes as input a two-level representation of a two-valued (or multiple-valued) Boolean function, and produces a minimal equivalent representation. The algorithms used represent an advance in both speed and optimality of solution in heuristic Boolean minimization. Espresso reads the file provided, performs the minimization, and writes the minimized result. It automatically verifies that the minimized function is equivalent to the original function.

5 RESULTS

The TPG structure optimization process is shown in Fig. 8. First, the initialization phase determines the initial TPG structure through the desired sequence of test patterns. Then the GA tries to optimize the circuit (make new configuration) while checking the allowed TPG structure and using the external evaluation tool. The evaluation tool calculates the cost of a given structure through the input test patterns and TPG configuration. After a number of iterations the best structure is chosen and implemented through the hardware description language. Parameters of the GA used in our experiments are:

- for first three circuits: number of generations is 50, population size is 10, probability of crossover is 0.8, and probability of mutation is 0.01,
- for the next three circuits: number of generations is 100, population size is 50, probability of crossover is 0.7, and probability of mutation is 0.05.

Table 1 presents the results of the evaluation of the optimization process with the ISCAS test-benchmark combinational circuits. The widely accepted ISCAS benchmark suite has been in use since being introduced in simple netlist format at the International Symposium of Circuits and Systems in 1985 (ISCAS '85). The 1989 ISCAS symposium introduced a set of sequential circuits, that were similar to the 1985 circuits, but with the addition of a D-type flip-flop element. These simple combinatorial circuits are used to benchmark various test pattern generation systems.

The test circuits used in our evaluation were transformed by the input reduction procedure proposed in [13]. The test pattern width (*ie*, the number of the inputs) and the number of test patterns (*ie*, the number of different input test vectors to cover all possible faults) are presented in the second and the third column, respectively, for each benchmark. The next two columns present the total cost of the modification logic reported by Espresso for the initial and optimized TPG structure. The last column shows the achieved improvement. The execution time of the GA algorithm itself was always below one second, while the evaluation phase, performed by the external evaluation tool, took couple of seconds per evaluation. We do not report total execution time, which in fact was measured in minutes, but since this is off-line and one-time optimization procedure, optimization effectiveness was considered more important as optimization time.

As mentioned before, the bit-order of the test patterns and the order of the test patterns in a test sequence influence the area of the modification logic. In this respect it

may be interesting to compare the results also with the results of column matching algorithm [14]. Both approaches use MISR of similar complexity, the main differences are in the design of the modification logic. Table 2 shows the results of the comparison of the two approaches for the same benchmark circuits.

The complexity figures in the 2nd and 3rd columns of Table 2 are expressed in terms of a total cost reported by Espresso per bit of the produced test pattern:

$$\text{complexity} = \frac{\text{total\_cost}}{\text{test\_pattern\_width} * \text{number\_of\_test\_patterns}}. \quad (6)$$

We need to apply such a measure because we used in experiments different test pattern sets than those reported in [14].

The comparison presented in Table 2 indicates that the proposed approach has a higher potential to provide solutions of TPG generating deterministic test patterns than column matching. There is also a big difference in testing time. In column matching solution all deterministic test patterns are embedded in a long test sequence composed of 5000 test vectors, which contains a lot of patterns not contributing to the fault coverage in the CUT. On the other hand, the GA based solution produces all deterministic test patterns as a one short test sequence that does not contain any superfluous vectors.

Table 3 presents the area of TPG logic for AMS $0.35\,\mu$m technology for the implementations reported in [15] and the GA based solutions. The area is expressed in terms of equivalent two input NAND gates. Like in Table 2, we need to apply a specific measure of the area overhead of the TPGs due to the fact that different deterministic patterns sets have been used for TPG synthesis. The proposed measure is expressed by the following formula:

$$\text{area\_per\_bit} = \frac{\text{area}}{\text{test\_pattern\_width} * \text{number\_of\_test\_patterns}}. \quad (7)$$

Experimental results shown in Table 3 indicate that for some benchmarks the proposed TPG and the GA optimization procedure provide solutions with lower area overhead than the TPG presented in [15] while for some other benchmarks the TPG in [15] are better. This may be due to the fact that we used ESPRESSO as a fast evaluation tool in the TPG optimization process and SNOPSYS as a tool for synthesizing the final solution. Applying SYNOPSYS as both the evaluation tool and the final synthesis tool is likely to improve the results.

6 CONCLUSION

A new type of deterministic TPG is presented in the paper. It is based on a feedback shift register composed of D- and T-type flip-flops and inverters. It is also equipped with a modification logic that can invert any bit in any pattern generated by the register. A genetic algorithm which minimizes the area overhead of the TPG for the given deterministic test set is also described. The initial structure of the TPG is encoded and multiplied with some variations to form the initial population. The search for the optimal structure of the TPG is performed by selection, crossover, and mutation operators, while each solution is evaluated by the evaluation tool. TPG area optimization performed on ISCAS test-benchmark circuits gave promising initial results.

References

[1] BUSHNELL, M. L.—AGRAWAL, V. D.: Essentials of Electronic Testing for Digital, Memory and Mixed-Signal Circuits, Kluwer Academic Publishers, 2000..

[2] STROUD, C. E.: A Designer's Guide to Built-in Self-Test, Kluwer Academic Publishers, 2002.

[3] NADEAU-DOSTIE, B.: Design for at-Speed Test, Diagnosis and Measurement, Kluwer Academic Publishers, 2000.

[4] GARBOLINO, T.—HLAWICZKA, A.: A New LFSR with D and T Flip Flops as an Effective Test Pattern Generator for VLSI Circuits, Lecture Notes in Computer Science, vol. 1667, 1999, pp. 321–338.

[5] PRINETTO, P.—REBAUDENGO, M.—SONZA REORDA, M.: An Automatic Test Pattern Generator for Large Sequential Circuits Based on Genetic Algorithms, in Proc. ITC94: IEEE International Test Conference, Washington D. C., USA, 1994, pp. 240–249.

[6] CORNO, F.—PRINETTO, P.—REBAUDENGO, M.—SONZA REORDA, M.: GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits, IEEE Transactions on Computer-Aided Design **15** No. 8 (1996), 943–951.

[7] CORNO, F.—PRINETTO, P.—SONZA REORDA, M.: A Genetic Algorithm for Automatic Generation of Test Logic for Digital Circuits, in Proc. IEEE International Conference on Tools with Artificial Intelligence, Toulouse, France, November 1996, pp. 10–16.

[8] SINGH, R.—KNIGHT, J.: Concurrent Testing in High Level Synthesis, in Proc. 7th Int. Symposium on High-Level Synthesis, pp. 96–103, 1994.

[9] KOROUŠIĆ-SELJAK, B.: Timetable construction using general heuristic techniques, Journal of Electrical Engineering **53** (2002), 61–69.

[10] PAPA, G.—KOROUŠIĆ-SELJAK, B.: An Artificial Intelligence Approach to the Efficiency Improvement of a Universal Motor, Engineering Applications of Artificial Intelligence **18** (2005), 47–55.

[11] PAPA, G.—ŠILC, J.: Automatic Large-Scale Integrated Circuit Synthesis Using Allocation-Based Scheduling Algorithm, Microprocessors and Microsystems **26** (2002), 139–147.

[12] Espresso, Version 2.3, Release date 01/31/88, UC Berkeley [Online]. Available: http://www-cad.eecs.berkeley.edu/Software/software.html.

[13] CHEN, C. A.—GUPTA, K.: Efficient BIST TPG Design and Test Set Compaction via Input Reduction, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **17** (1998), 692–705.

[14] FISER, P.—HLAVICKA, J.: Column Matching Based BIST Design Method, in Proc. IEEE European Test Workshop, Corfu, Greece, 2002, pp. 15–16.

[15] BELLOS, M.—KAGARIS, D.—NIKOLOS, D.: Test Set Embedding Based on Phase Shifters, in Proc. The Fourth European Dependable Computing Conference EDCC-4, Toulouse, France, 2002, pp. 90–101.

Received 24 April 2006

Gregor Papa (PhD) is a researcher at the Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia. He received his MSc and PhD degrees in Electrical Engineering from the University of Ljubljana, Slovenia, in 2000 and 2002, respectively. His research interests include optimization techniques, meta-heuristic algorithms, high-level synthesis of integrated circuits, hardware implementations of high-complexity algorithms, and industrial product improvements. His work is published in several international journals.

Tomasz Garbolino (PhD) is an assistant professor in the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology at Gliwice, Poland. He received his MS and PhD degrees (with honors) in Electronics from the Technical University of Gliwice in 1993 and 2002, respectively. His research interests encompass built-in self-test structures for digital circuits and SOCs, with particular focus on test pattern generators and test pattern decompression techniques, as well as design for testability issues. He is a co-author of several papers that have been published in proceedings of international conferences and journals.

Franc Novak (Prof, PhD) gained the BSc, MSc, and PhD degrees in electrical engineering from the University in Ljubljana in 1975, 1977, and 1988, respectively. Since 1975 he has been with the Jožef Stefan Institute, where he is currently head of Computer Systems Department. Since 2001 he is also associate professor, at Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests are in the areas of electronic testing and diagnosis, and fault-tolerant computing. His most recent assignment has been on design for testability of analogue circuits.

Andrzej Hławiczka (Prof, PhD) is a professor in the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology at Gliwice, Poland. He received the MS degree in Electrical Engineering from the Technical University of Gliwice in 1965. His PhD degree, in Computer Engineering, and DSc, in Electronics, were received from the Silesian University of Technology at Gliwice, in 1973 and 1998, respectively. From 1965 to 1979 he was mainly with the Institute of Mathematical Machines and Institute of Control Systems, Katowice, Poland, where he published research related to hazards detection, logic simulation, and fault test generation. His research interests include digital circuits and SoCs built-in self-testing, fault-tolerant computing, and design for testability. He is the editor, co-editor, and co-author of several scientific books published in Polish, Russian and English. He has published over 120 scientific papers in international journals and conferences, and has supervised several PhD dissertations.