UDK621.3:(53+54+621+66), ISSN0352-9045

A NEW APPROACH TO OPTIMIZATION OF TEST PATTERN GENERATOR STRUCTURE

Gregor Papa[1], Tomasz Garbolino[2]

[1]Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
[2]Institute of electronics, Silesian University of Technology, Gliwice, Poland

Key words: test pattern generator, design, optimization, genetic algorithm.

Abstract: This paper presents a new approach to design and 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. Such a TPG is suitable for on-line built-in self-test (BIST) implementations where functional units are tested in their idle cycles. To reduce the gate count of the BIST structure a genetic algorithm (GA) is employed. This approach and its multi-objective nature allows concurrent optimization of multiple parameters within multiple design aspects (register cells type, patterns order in the generated test sequence, bit order of a test pattern), which influence the final solution. Experimental results on combinational ISCAS benchmarks demonstrate the efficiency of the proposed evolutionary approach.

Nov pristop k optimiranju strukture generatorja testnih vzorcev

Kjučne besede: generator testni vzorcev, načrtovanje, optimiranje, genetski algoritem.

Izvleček: V članku je predstavljen nov pristop k načrtovanju in optimiranju strukture generatorja testnih vzorcev (TPG). TPG je sestavljen iz linearnega registra in nelinearne kombinacijske funkcije, ki lahko invertira katerikoli bit generiranega vzorca. Tako lahko dobimo poljubno testno sekvenco. Takšen TPG je primeren za on-line built-in self-test (BIST) izvedbe, kjer se funkcijske enote testirajo v njihovih prostih ciklih. Za zmanjšanje števila logičnih vrat strukture BIST, je uporabljen genetski algoritem (GA). Večkriterijska narava tega pristopa omogoča sočasno optimiranje več parametrov na več načrtovalnih nivojih (tip pomnilnih registrov, vrstni red vzorcev, vrstni red bitov v vzorcih), kar vse vpliva na končno rešitev. Rezultati testiranja s kombinacijskimi testnimi vezji ISCAS so pokazali uspešnost uporabljenega pristopa.

1 Introduction

The complexity of modern integrated circuits and rapid changes in technology pose an ever-increasing number of challenges in testing electronic products. With the introduction of surface mounted devices, small pitch packaging becomes prevalent, which makes the access to the test points on a board either impossible or at least very costly. Traditional in-circuit test techniques that utilize a bed-of-nails to make contact to individual leads on a printed circuit board have become inadequate. To cope with this problem, boundary-scan approach has been developed and is now widely adopted in practice /3//25/. Another problem originates from the fact that the number of transistors in a chip increases faster than the pin count and consequently internal chip modules become more and more 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, /4/. One of the alternative solutions is to implement a built-in self-test (BIST) of the UUT, /28/, with on-chip test pattern generation (TPG) and on-chip output response analysis logic. In this way, the test circuitry is incorporated on-chip and communication with external ATE is reduced to test initiation and transfer of test results /22/. Besides, self-test can be performed at the circuit's normal clock rate. This may increase the coverage of faults that could otherwise be detected only during normal system operation. In addition, BIST can be used for periodic testing and/or to diagnose system failures in system maintenance. On the other hand, 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.

Different TPG approaches have been proposed. They can be classified as ROM-based deterministic, algorithmic, exhaustive and pseudo-random. In the first approach, deterministic patterns are stored in a ROM and a counter is used for their addressing, /10/. This simple approach is limited to small test pattern sets. Algorithmic TPG are mostly used for testing regular structures such as RAMs /30/. Exhaustive TPG is counter-based approach which suffers from the fact that it is not able to generate specific sequence of test vectors. With some modifications, however, counter-based solutions are able to generate deterministic test patterns, /5//16//18/. Pseudo-random TPG is most commonly applied technique in practice. In this approach Linear Feedback Shift Register (LFSR) or Cellular Automata (CA) are employed to generate pseudo-random test patterns. In order to decrease the complexity of a TPG, designers usually try to embed deterministic test patterns into the vector sequence generated by some linear register. Such embedding can be done either by re-seeding a TPG or modifying its feedback function /17/. There are also solutions that modify or transform the vector sequence produced by a LFSR in such a way that it contains deterministic test patterns /29//2/. Most proposed LFSR structures are based on D-type flip-flops. In recent years, LFSR composed of D-type and T-type flip-flops is gaining popularity due to its low area overhead and high operating speed /13//14/.

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. 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 /8//26/. 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 /9/.

This work was motivated by the need of deterministic test pattern generation for the on line BIST of structure composed of idle function units and registers, originally proposed in /27/. 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. Deterministic test sequences (i.e., in which non-useful test vectors are eliminated) may also considerably reduce diagnosis time in fault localization /19/.

2 Test pattern generator structure

A TPG can be regarded as an autonomous finite-state machine that is typically configured as a shift register with additional feedback connections. A TPG is said to be linear if its feedback logic is composed exclusively of XOR gates, otherwise it is said to be non-linear. A TPG is initialized to a known initial state, and the contents of its flip-flops in the initial state are called the seed. The flip-flops are clocked to cause the transitions, whose exact nature depends on the feedback connections. The values of the state variables in subsequent transitions are used as test patterns. Most reported TPGs are D flip-flop based linear feedback shift registers (LFSRs).

In a typical scenario, a TPG is initialized with a given deterministic seed and run until the desired fault coverage is achieved. The test application time using an LFSR is significantly larger than what is required for applying the test set generated using a deterministic TPG. This is due to the fact that vector set generated by a LFSR includes besides useful vectors also many other vectors that do not contribute to the fault coverage. In order to reduce test application time, current non-useful vectors should be replaced by useful vectors appearing later in the test sequence. This can be done in a number of ways. Most commonly used techniques are reseeding and weighted-random pattern generation.

In our approach, the goal is to develop a TPG that would generate only the required test vectors (i.e., with no intermittent non-useful vectors). 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 $\overline{\mathrm{D}}$ or $\overline{\mathrm{T}}$). Thus, the register may have one of *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.

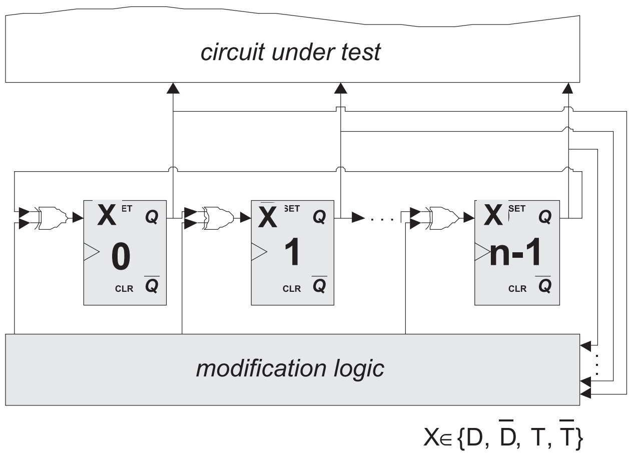

Fig. 1: Test pattern generator structure

In our case, MISR and the modification logic are application specific: they are synthesized according to the required test pattern set. 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.

One of the parameters that are important for practical implementations of TPGs is area overhead. It is influenced by the structure of each MISR stage, the order of the test patterns in a test sequence and the bit-order of the test patterns. While the first property influences the complexity of both the MISR and the modification logic, the remaining two impact only the area of the modification logic.

3 Genetic algorithm

The population-based evolutionary approach - employed through GA /1//7//15/ was used for optimization because of its intrinsic parallelism that allows searching within a broad database of solutions in the search space simultaneously, climbing many peaks in parallel. Therefore, the risk of converging to a local optimum is low. Besides, promising results of our research work obtained in other optimization problem areas /20//21//23//24/ encouraged us to consider GA approach as one of the possible alternatives in TPG synthesis optimization.

The implementation of genetic operators is described with more details in /24/.

4 Structure evaluation

Operation of the *j*-th cell of the TPG register during one clock cycle can be expressed by the following equation:

$$Q_j = t_j\, q_j \oplus q_{j-1-} \oplus i_j \oplus f_j$$

$$Q_1 = t_1\, q_1 \oplus q_{n-} \oplus i_1 \oplus f_1 \qquad (1)$$

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 , i.e., *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 , i.e., *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:

$$f_{j,} = t_j\, q_j \oplus q_{j-1-} \oplus i_j \oplus Q_j$$

$$f_1 = t_1\, q_1 \oplus q_{n-} \oplus i_1 \oplus Q_1 \qquad (2)$$

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.

Further, Espresso software /11/ was used for Boolean minimization of the modification logic and its approximate cost evaluation. This software takes a two-level representation of a two-valued (or multiple-valued) Boolean function as input, and produces a minimal equivalent representation (number of equivalent gates). It automatically verifies that the minimized function is equivalent to the original function. The algorithms used represent an advance in both speed and optimality of solution in heuristic Boolean minimization.

5 Results

The optimization process is shown in Figure 2, where the initialization phase determines the initial TPG structure through the desired sequence of test patterns. The GA tries to optimize the circuit (make new configuration) while checking the allowed TPG structure and using the external structure 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; a) 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, and b) 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. The final solution for each circuit was the best one found after a few repetitions of optimization. There were few repetitions due to the non-deterministic nature of the genetic algorithm.

In Table 1 the results of the evaluation of the optimization process with the ISCAS test-benchmark combinational circuits are presented. 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. In 1989 ISCAS symposium a set of sequential circuits was introduced, 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.

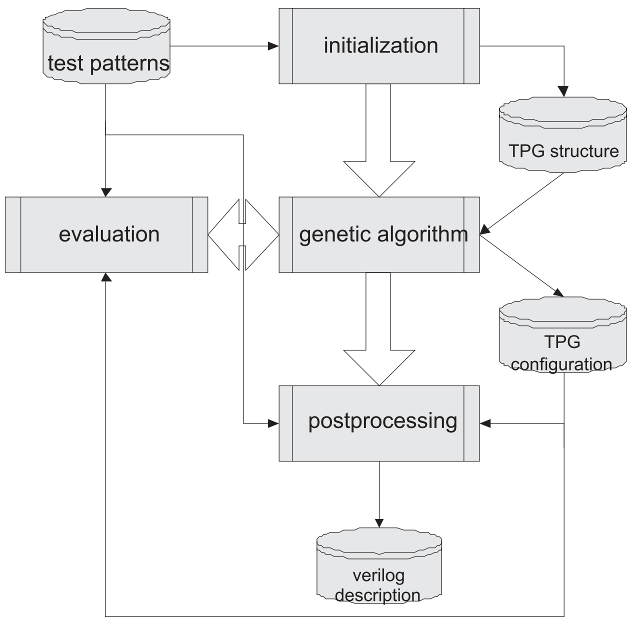

Fig. 2: The optimization process

All test circuits used in our evaluation were transformed by the input reduction procedure proposed in /6/. The test pattern width (denotes the number of the inputs) and the number of test patterns (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 (number of equivalent gates) of the modification logic reported by Espresso for the initial and optimized TPG structure. The last column shows the achieved improvement between initial and optimized structure. The execution time of the GA algorithm itself was always below a second, while the evaluation phase, performed by the external structure evaluation tool, took couple of seconds per evaluation. There is no report on total execution time, which in fact was measured in minutes, but since this is off-line optimization procedure, optimization effectiveness was considered more important as optimization time.

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

	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

Since 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, it might be interesting to compare the results also with the results of column matching algorithm /12/. Both approaches use MISR of similar complexity, while 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:

$$complexity = \frac{total\_cost}{test\_pattern\_width * number\_of\_test\_patterns} \quad (3)$$

Such a measure was applied because in experiments different test pattern sets were used than those reported in /12/.

Table 2. Comparison with results achieved in /12/

	complexity of TPG obtained by column matching	complexity of the proposed TPG obtimized 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

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. Another big difference is also 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.

In Table 3 the comparison of the area of TPG logic for AMS 0.35 µm technology for the implementations reported in /2/ and the GA based solutions is presented. The area is expressed in terms of equivalent two input NAND gates. As in Table 2, a specific measure of the area overhead of the TPGs was applied due to the fact that different deterministic patterns sets have been used for TPG synthesis. The proposed measure is expressed by the following formula:

$$area\_per\_bit = \frac{area}{test\_pattern\_width * number\_of\_test\_patterns} \quad (4)$$

Table 3. Comparison with results achieved in /2/

	area_per_bit of the TPG in [2]	area_per_bit of the proposed TPG obtimized 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

Experimental results 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 /2/ while for some other benchmarks the TPG in /2/ are better. This may be due to the fact that we used Espresso as a fast evaluation tool in the TPG optimization process and Synopsys as a tool for synthesizing the final solution. Therefore, applying Synopsys as both the evaluation tool and the final synthesis tool is likely to improve the results.

The above examples are good for illustrating the advantages of the proposed approach in comparison with the existing solutions. However, one should be aware that the employed benchmark circuits are relatively small. Realistic assessment of techniques for automatic deterministic test pattern generation requires more complex circuits. Since such examples are not reported in the referred papers, we performed GA optimization approach on some larger benchmark circuits. While the results regarding the complexity and the area per bit are in average comparable to the GA examples reported above, the computation time for larger circuits considerably increases and may represent a bottleneck in practical implementations. For example, the computation time for circuit s38417 was 140 times larger than for c880.

6 Conclusion

In many cases, pseudo random pattern generators provide reasonable fault coverage for different circuits-under-test. However, if a TPG fails to provide the desired fault coverage within the given test length, application specific deterministic TPGs are employed. Deterministic TPGs are by default more complex since they employ additional logic to prevent the generation of non-useful test patterns.

Area overhead is one of the important issues of the design of deterministic TPGs. In this paper, a new type of deterministic TPG is presented 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. The search for the optimal structure of the TPG is performed by a genetic algorithm and some illustrative case studies were performed on ISCAS test-benchmark circuits. Promising initial results have been obtained on small and medium benchmark circuits. The computation time for larger circuits considerably increases and may represent a bottleneck in practical implementations.

Acknowledgment

This work was supported by the Slovenian Research Agency program Computing structures and systems and the bilateral Slovenian-Polish project Development of on-line built-in self-test structures (BI-PL/03-04-012).

References

/1/ T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.

/2/ M. Bellos, D. Kagaris, D. Nikolos, Test set embedding based on phase shifters, in *Proc. Fourth European Dependable Computing Conference*, EDCC-4, 2002, pages 90–101.

/3/ H. Bleeker, P. van den Eijnden, F.de Jong, *Boundary-scan test, A practical approach*, Kluwer Academic Publishers, 1993.

/4/ M. Bushnell, V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal circuits*, Kluwer Academic Publishers, 2000.

/5/ K. Chakrabarty, B. Muray, V. Iyengar, Deterministic built in test pattern generation for high performance circuits using twisted ring counters, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 8, 2000, pp. 633–636.

/6/ C.-A. Chen, K. Gupta, Efficient bist tpg design and test set compaction via input reduction, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, 1998, pp. 692–705.

/7/ C.C. Coello, A comprehensive survey of evolutionary-based multiobjective optimization techniques, *Knowledge and Information Systems*, Vol. 1, 1999, pp. 269–308.

/8/ F. Corno, P. Prinetto, M. Rebaudengo, M.S. Reorda, Gatto: a genetic algorithm for automatic test pattern generation for large synchronous sequential circuits, *IEEE Transactions on Computer-Aided Design*, Vol. 15, 1996, pp. 943–951.

/9/ F. Corno, P. Prinetto, M.S. Reorda, A genetic algorithmfor automatic generation of test logic for digital circuits, in *Proc. IEEE International Conference On Tools with Artificial Intelligence*, 1996, pp. 10–16.

/10/ G. Edirisooriya, J. Robinson, Design of low cost ROM based test generators, in *Proc. IEEE VLSI Test Symposium*, 1992, pp. 61–66.

/11/ Espresso, UC Berkeley, 1988, *http://www-cad.eecs.berkeley.edu:80/software/software.html*.

/12/ P. Fišer, J. Hlavička, Column matching based bist design method, in *Proc. IEEE European Test Workshop*, 2002, pp. 15–16.

/13/ T. Garbolino, A. Hlawiczka, A new LFSR with D and T flip flops as an effective test pattern generator for VLSI circuits, Vol. 1667 of *Lecture Notes in Computer Science*, 1999, pp. 321–338.

/14/ T. Garbolino, A. Hlawiczka, A. Kristof, Fast and low area TPG based on T-type flip flops can be easily integrated to the scan path, in *Proc. IEEE European TestWorkshop*, 2000, pp. 161–166.

/15/ D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

/16/ Hellebrand, S., Liang, H., and Wunderlich, H. (2000). A mixed mode bist scheme based on reseeding of folding counters. In *Proc. International Test Conference 2000*, pages 778–784.

/17/ S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, B. Courtois, Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers, *IEEE Transaction on Computers*, Vol. 44, 1995, pp. 223–233.

/18/ D. Kagaris, S. Tragoudas, Generating deterministic unordered test patterns with counters, in *Proc. 14th VLSI Test Symposium*, 1996, pp. 374–379.

/19/ M. Khalil, C. Robach, F. Novak, Diagnosis strategies for hardware or software systems, *Journal of electronic testing: Theory and Applications*, Vol. 18, 2002, pp. 241–251.

/20/ B. Koroušić-Seljak, Timetable construction using general heuristic techniques, *Journal of Electrical Engineering*, Vol. 53, 2002, pp. 61–69.

/21/ P. Korošec, J. Šilc, B. Robič, Population-based methods as a form of metaheuristic combinatorial optimization, *Electrotechnical Review*, Vol. 72, 2005, pp. 214–219.

/22/ B. Nadeau-Dostie, *Design for at-speed test, diagnosis and measurement*, Kluwer Academic Publishers, 2000.

/23/ G. Papa, B. Koroušić-Seljak, An artificial intelligence approach to the efficiency improvement of a universal motor, *Engineering applications of artificial intelligence*, Vol. 18, 2005, pp. 47–55.

/24/ G. Papa, An evolutionary approach to chip design: an empirical evaluation, *Informacije MIDEM*, Vol. 33, 2003, pp. 142–148.

/25/ K. Parker, *The boundary-scan handbook, Third edition*, Kluwer Academic Publishers, 2003.

/26/ P. Prinetto, M. Rebaudengo, M.S. Reorda, An automatic test pattern generator for large sequential circuits based on genetic algorithms, in *Proc. IEEE International Test Conference*, 1994, pp. 240–249.

/27/ R. Singh, J. Knight, Concurrent testing in high level synthesis, in *Proc. 7th International Symposium on High-Level Synthesis*, 1994, pp. 96–103.

/28/ C. Stroud, *A designer's guide to built-in self-test*, Kluwer Academic Publishers, 2002.

/29/ N. Touba, E. McCluskey, Bit-fixing in pseudorandom sequences for scan BIST, *IEEE Transactions on Computer-Aided Design of Integrated Circuits And Systems*, Vol. 20, 2001, pp. 545–555.

/30/ A. van de Goor, *Testing semiconductor memories: theory and practice*, John Wiley & Sons, 1991.

Gregor Papa
Jožef Stefan Institute
Computer Systems Department
Jamova c. 39, 1000 Ljubljana, Slovenia
tel. +386 1 4773 514
fax. +386 1 4773 882
Email: gregor.papa@ijs.si

Prispelo (Arrived): 28.06.2007 Sprejeto (Accepted): 28.03.2008