Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 2, 2004

A Complete Laboratory on Evolutionary Electronics

Rustem Popa[1]

Abstract – Evolutionary design might be a promising option to the conventional design of electronic circuits. Each project is assembled from a number of component parts and then is tested in the frame of an evolutionary algorithm. We have presented in this paper some evolutionary experiments of digital and analog electronic circuits design, both by simulating evolution in software and by true evolution in hardware. These experiments are conducted with our students in the laboratory of Evolutionary Systems.
Keywords: genetic algorithms, evolvable hardware, electronic circuits, evolutionary computation

I. INTRODUCTION

Evolutionary systems are designed by the means of evolutionary computation. These "designs" are evolved by a process of natural selection, like in the living matter. The mechanism of evolution is entirely blind and has no particular object other than survivability. The survivability of the organism can be seen as a process of assembling a larger system from a number of component parts and then testing the organism in the environment in which it finds itself. The concept of assemble-and-test together with an evolutionary algorithm can explore the entire design space because of the absence of imposed rules of design. In this way, in electronics, evolutionary design generates new unexpected and usually useful electronic circuits.

The building of new electronic circuits by evolutionary computation has been created the concept of Evolvable Hardware (EHW). The usual design process is in a top-down way and begins with a precise specification. EHW is applicable even when no hardware specification is known before. Its implementation is determined through a genetic learning in a bottom-up way. A Genetic Algorithm (GA) is intended to mimic Darwinian evolution. A population of solutions, called chromosomes, is maintained, and goes through a series of generations. For each new generation, some of the existing chromosomes survive, while others are created by a type of reproduction and mutation from a set of parents. EHW combine knowledge of both GA and electronic circuits design to evolve new circuits.

Research in EHW can be divided into *intrinsic evolution*, which refers to an evolutionary process in which each circuit is built in electronic hardware and tested, and *extrinsic evolution*, that uses a model of the hardware and evaluates it by simulation in software.

We are convinced that very soon, as reprogrammable integrated circuits will become larger and larger, and the design techniques will be improved, EHW will be dominant in electronics, and the electronic engineer must be ready for this future evolution. It is true that, for the time being, the complexity of evolved circuits is so far small. The main problem is the representation of the circuit in chromosomes, because complex circuits need a great number of architecture bits, which directly influences the GA search space.

We have prepared some experiments of extrinsic evolution in digital and analog circuits, and for solving of some wellknown optimization problems, like the generation of test vectors in digital circuits, the finding of the global minima in a multimodal function, or the solving of the Traveling Salesman Problem (TSP). We have also prepared two experiments of intrinsec EHW in digital circuits by using common digital CMOS circuits. The first one is a test platform with controlled switches for experiments with simple building blocks made-up from few transistors. The second platform consists of CMOS switches, some simple logic gates, and three JK flip-flops for experiments with registers and counters. Finally, a real Xilinx XCR3064 CoolRunner CPLD mounted on a XCRP board may be used to implement some extrinsic EHW circuits.

The remaining sections of the paper are organised as follows: Section II describes in more detail the genetic learning component of the EHW. Section III shows some examples of digital circuits designed by software with a GA and then implemented in a 64 macrocell Xilinx CPLD. Section IV shows some examples of simple analog and digital circuits implemented by evolution on real hardware configurable boards. Finally, Section V provides the conclusions and future work.

[1] Universitatea „Dunărea de Jos" din Galaţi, Departamentul Electronică şi Telecomunicaţii, Str. Domnească Nr. 111, 800201, Galaţi, e-mail: Rustem.Popa@ugal.ro

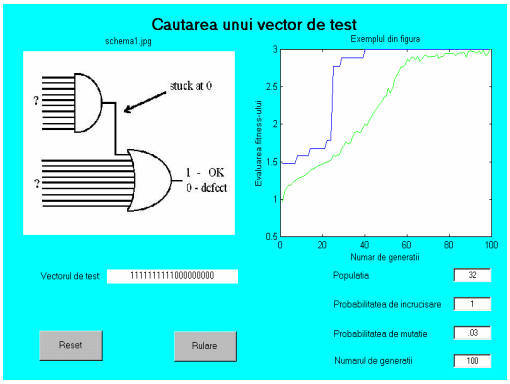

Fig. 1. A GUI for the search of a test vector which points a stuck-at 0 fault in the marked node

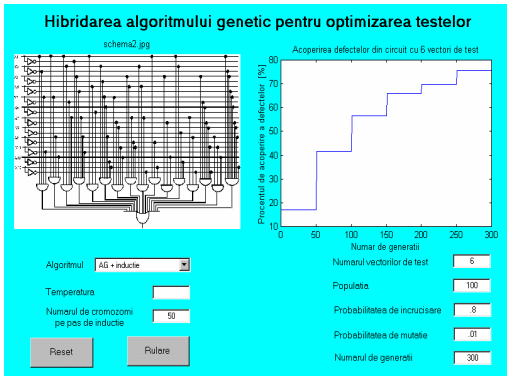

Fig. 2. A GUI for the generation of an optimum set of test vectors by hibridation of a GA

II. GENETIC LEARNING IN EHW

All the developed algorithms are based on GAs, an adaptive searching technique for solving optimisation problems based on the mechanics of natural genetics and natural selection. The success of the application of GAs to an optimisation problem depends on the representation of chromosomes, fitness function, method of crossover, mutation operation, and on the diverse information from the chromosomes. When the diversity is lost before the global optimum solution is found, the performance of GAs deteriorates and their solution processes converge prematurely. Moreover, the mutation operation is important. While the mutation operation adds new information to a chromosome, it can also destroy useful information held in the chromosome.

In GAs the search is conducted using information of a population of candidate solutions, called chromosomes, so that the chance of the search being settled in a local optimum can be significantly reduced. Four essential components need to be designed in applying a GA for an optimisation problem: chromosomes representation, crossover operator, mutation operator and fitness function.

In a reconfigurable circuit, each bit of a chromosome represents usually the state of a programmable switch. The entire chromosome represents the state of all switches, that is a complete circuit, which may be good or bad, according with his fitness. The initial population of chromosomes (bit strings) is generated randomly. All these potential solutions are evaluated using a fitness function. In our case, for a single boolean function, fitness is the ratio between the number of the correct values of the function and the number of all possible values (which is 2^n, if the boolean function has n input variables). A well-designed circuit will be obtained only when the value of fitness is 100%. An approximately value of the fitness is unacceptable here.

The next step is selection and reproduction. For each individual, a number of copies are made, proportional to its fitness, while keeping the population size constant. The least fit individuals are deleted. This is the survival of the fittest part of the GA.

The next step is crossover, where individuals are chosen two at a time, as parents. They are converted into two new individuals, called offsprings, by exchanging parts of their structure. Thus, each offspring inherits a combination of features from both parents. We have obtained the best results with one point crossover, with a probability of 80%. This operator may be used more times on different selected pairs of chromosomes in a generation.

The next step is mutation. A small change is made to each resultant offspring, with a small probability. After mutation is performed on an individual, it no longer has just the combination of features inherited from its two parents, but also incorporates the additional change caused by mutation. This ensures that the algorithm can explore new features that may not yet be in the population. It makes the entire search space reachable despite the finite population size. The whole process is repeated for several generations, and, if the best chromosome in population will have the fitness of 100%, then this bit string represents a good solution for our function.

III. EXPERIMENTS WITH EXTRINSIC EHW

The first set of experiments show the generation of complexity with very simple rules in unidimensional and bidimensional Cellular Automata (CA), and the solving of some complex NP-problems (the finding of the global minima in a multimodal function, or the solving of the TSP) with GAs. These experiments have been ample described in [7].

Another set of experiments have been prepared for the purpose of automated generation of test vectors in digital circuits. If we want to generate a test to detect a stuck-at 0 fault in the marked node of the circuit represented in the Fig.1., the required vector is 1111111111000000000, a combination of bits nearly impossible to find using a random approach ([6]). As

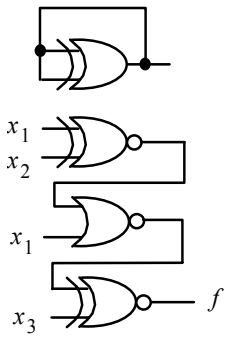

Fig. 3. The circuit achieved by evolution for the boolean function from the equation (1)

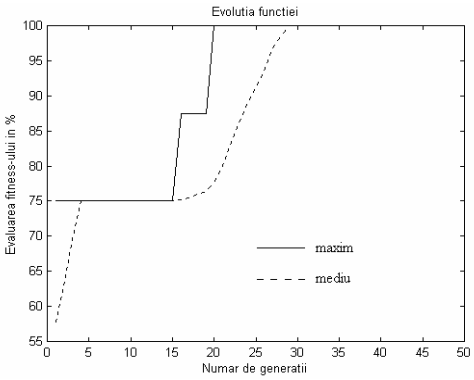

Fig. 4. The evolution of the fitness across 50 generations

we can see in the Graphic User Interface (GUI) from the Fig.1, the GA used to solve this problem has found the correct solution in 40 generations. The algorithm uses a population of 32 chromosomes and a mutation rate of 3%. Fitness was calculated as the sum of (1 if fault is excited or 0 otherwise) + (fraction of inputs in AND gate set to 1) + (fraction of inputs in OR gate set to 0). The maximum value of the fitness defined in this way is 3.

By using the GUI from the Fig.2., we can solve the Fault Coverage Code Generation Problem for a more complex combinational logic. The problem consists in finding of a given number of test vectors that maximizes the fault coverage of the circuit. We have chosen two ways of hibridation of the standard GA: by using the inductive search, like in the Fig.2., or by using the simulated annealing algorithm. The example from the Fig.2. shows that only 6 test vectors could cover more than 75% from the total number of stuck-at 0 faults in the circuit. All these GUIs (and also those from the first set of experiments) have been developed in Matlab 5.3.

A. *The Implementation of a Boolean Function*

We have considered a boolean function represented in a minimal disjunctive form by using a Karnaugh map:

$$f = \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_3 + \bar{x}_2 \cdot \bar{x}_3 \qquad (1)$$

This representation has a cost of 7 gates and 13 inputs, including inverters. By applying some switching-algebra theorems our function may be written in the next form:

$$f = x_3 \oplus \overline{\bar{x}_1 \cdot x_2} \qquad (2)$$

Now, the cost of implementation is of only 3 gates and 5 inputs. Unfortunately, there is no algorithm to find this convenient form of the function, only the heuristics and experience of the human designer.

We have tried to find another representation of this function by evolutionary design. We have used the idea given in [2]. Each combinational circuit is represented as a rectangular array of logic gates. Each of these gates has two inputs and one output, and the logic operator may be selected from a list. At the beginning of the search, all the gates from the matrix are disposable to implement a functional circuit. Once a functional solution appears, then the fitness function is modified such that any valid designs produced are rewarded for each gate which is replaced by a simple wire. The algorithm tries to find the circuit with the maximum number of gates replaced by wires that performs the function required.

The chromosome defines the connection in the network between the primary inputs and primary outputs. We have used a network of 4 gates, a population of 32 chromosomes, 10 of them beeing changed each generation, a single point 100% crossover and 5% rate mutation. A feasible solution has been obtained in less than 50 generations, as we can see in the Fig.3. and in the Fig.4. The cost is given now by 3 inverting gates and 6 inputs (one of the gates in the network is useless), and this solution has the minimum delay time between any input and the output of the circuit, in a gate level implementation.

B. *The Implementation of a Finite State Machine*

The Finite State Machine (FSM) represented in the Fig.5. is a sequence detector with one-input, one-output and 6-internal states. When the input sequence 011 occurs, the output becomes 1 and remains on this logic value until sequence 011 occur again. In this case, the output returns to 0, and maintain this value, until a new sequence 011 appears.

Initially a GA has been used to find optimal state assignment. The chromosome represents the FSM as a list of states. The initial population is generated randomly. The goal of the GA is to extract the optimum state assignment, which requires the least number of logic gates. For that reason the number of 2-inputs AND/OR logic gates are used to define the fitness function. The optimum state assignment is given in the Fig.5. A more detailed description of this problem is presented in [1].

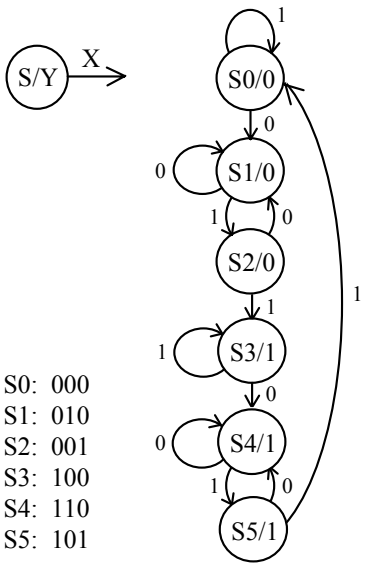

Fig. 5. A sequence detector described as state transition graph and GA state assignment

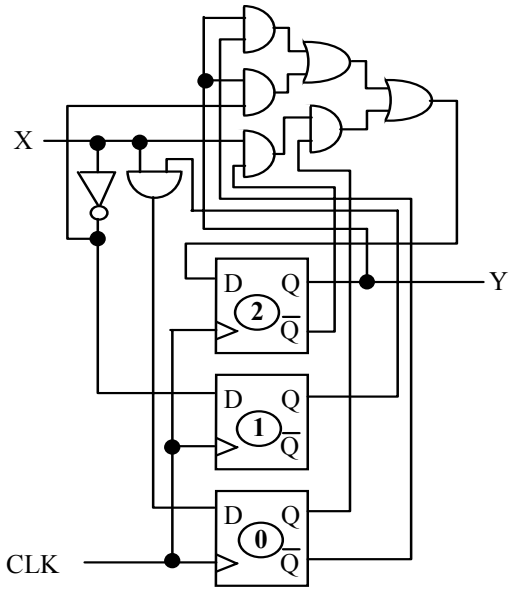

Fig. 6. Evolved optimal circuit solution of the sequence detector

Then, the extrinsic EHW has been used to find the functional design of combinational parts of the sequence detector. We have used the same method presented in the subsection A and in [2].

The equations of the evolved optimal combinational circuit are the following:

$$D_2 = Q_2 \cdot \overline{Q}_0 + \overline{x} \cdot Q_2 + x \cdot \overline{Q}_2 \cdot Q_0 \qquad (3)$$

$$D_1 = \overline{x} \qquad (4)$$

$$D_0 = x \cdot Q_1 \qquad (5)$$

$$y = Q_2 \qquad (6)$$

The schematic diagram of the circuit is given in the Fig.6. A bad state assignment may conduct to much more complex equations for the combinational circuit of the FSM.

C. *Some experiments with Xilinx XCR3064XL CPLD*

The circuit XCR3064XL, is a Xilinx CPLD with 64 macrocells and 1500 usable gates, providing low-power and very high speed, and beeing in-system programmable through JTAG IEEE 1149.1 Interface. Unfortunately, this circuit has only 1000 erase/ programming cycles guaranteed, so it can not be used with intrinsic EHW.

This programmable circuit is mounted on a board, called Digilab XCRP, delivered by Digilent, Inc. This low cost platform can be used to implement a wide variety of digital circuits. The programming pins of the circuit are directly connected to the parallel port pins of the computer.

The software we have used is Xilinx Integrated Software Environment (ISE) 6.1i, a complete CAD environment for implementation of complex digital circuits. We have generated the source file of the new project (schematic diagram or VHDL) and have obtained the fitter report and the timing report.

We have implemented the boolean function from the subsection A on the basis of equations (1) and (2) and the circuit from the Fig.3. We have obtained the same results, so we can assume that our software finds an optimal way in connecting the hardware resources of the circuit, even if the function is not done in a minimal form. The circuit has used a single macrocell from the maximum number of 64 (that is 1/64), only two product terms from the maximum number of 224 (that is 2/224), and only 3 function block inputs from the total number of 160 (that is 3/160). The pad to pad delay is 6 ns, and the total delay of the circuit is not more than this value. For more complicated functions, evolutionary design may offer a better fitting of circuit resources (a less number of product terms).

In sequential circuits, the optimal state assignment is crucial. The sequence detector from the subsection B, implemented with the equations 3,4,5 and 6, has used only 3/64 macrocells, 3/224 product terms, and 3/160 function block inputs. The same circuit, implemented with a non optimal state assignment has used 4/64 macrocells, 9/224 product terms, and 4/160 function block inputs. Even the combinational time delay is different for these circuits (4.7ns in the first case and 7,2ns in the second case). It's true that the main differences in the complexity of these circuits are given by the state assignment, but it seems that evolutionary design is more efficient even for the combinational part of a FSM.

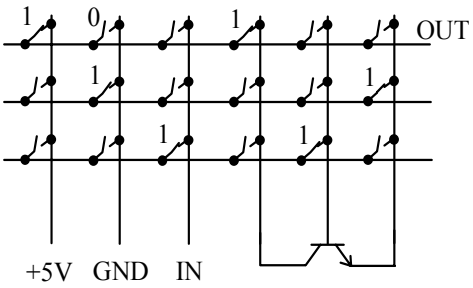

Fig. 7. Instantiation of the NOT gate on the evolvable testbed

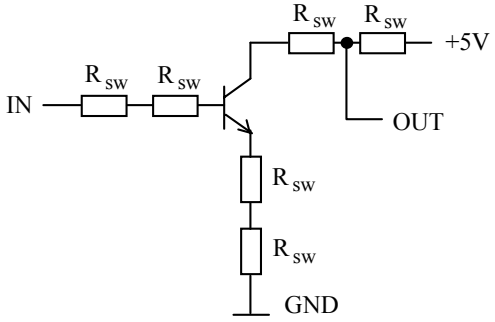

Fig. 8. The equivalent circuit diagram for the NOT evolved gate

D. *The Implementation of Analog Circuits*

Analog circuit synthesis entails the creation of both the topology and the sizing (numerical values) of all of the circuit's components. The difficulty of this problem is well known and the first auspicious approach, based on genetic programming, was presented in [3]. Another method of automatically generating analog circuit designs based on a parallel GA and a set of circuit constructing primitives is presented in [5].

Both methods need a huge computation power (few days on a parallel computer with 64 processors in the first case, or a network of workstations in the second case). We have only verified through PSpice simulation some of the circuits presented in [3] and [5]. Interesting is the fact that not all given circuits have been successful in our simulations.

IV. EXPERIMENTS WITH INTRINSIC EHW

We have prepared two set of experiments on intrinsic evolvable hardware of digital circuits by using common digital CMOS circuits.

A. *A Test Platform for Intrinsic Analogic EHW*

The first one is a test platform designed specifically for simple experiments into intrinsic hardware evolution. Based on an idea from [4], this testbed is in fact a matrix of analogue switches, connected to some plug-in boards, which contain the desired building-blocks for experimentation. In [4] is described a great motherboard with 12 integrated circuits (IC) CD22M3494, each of them beeing a matrix of 16×8 analogue switches. These ICs are very expensive, so we have built a much smaller board with only 20 ICs 4066, each of them having only 4 analogue switches, that is a total number of 80 programmable switches.

As a starting point for experimentation, bipolar transistors were used as the evolutionary building-block, and the task was to evolve a NOT gate. The digital input to the testbed is provided by a computer via a digital input/output board, and the output is connected to an A/D converter on the board.

We have used a standard GA, with a population of 50 chromosomes, 18 bit each of them, with a single point crossover, proportional selection and elitism. The mutation rate was 5%. The evolved circuit with a single transistor is shown in the Fig.8. The corresponding chromosome that build up the state of the analogue switches represented in the Fig.7. is 100100010001001010.

The on-switches resistances might be about 50Ω for CD22M3494 circuit, or about 300Ω for 4066 IC. The circuit from the Fig.8. conforms to the NOT function in that its output corresponding to 0 input is of slightly higher voltage than that corresponding to a 1 input, however this difference (only 1V) is too small to be of any practical use. If we repeat evolution, with much more switches, we can see that these additional switches are placed in parallel, reducing the combined resistance from the emiter. This configuration will give a good voltage swing and the circuit will become an inverter with a NPN transistor, like we know.

B. *A Test Platform for Intrinsic Digital EHW*

The second platform consists of CMOS switches, some AND gates and three JK flip-flops for experiments with registers and counters. The most suitable way to connect each data input of a flip-flop to a lot of different signals from the circuit, is by using CMOS analog multiplexers/demultiplexers.

The schematic diagram from the Fig.9. shows the building block used to design this board. Each data input of a JK flip-flop has an 8-channel analog multiplexer 4051. The first two inputs in the multiplexer are constants 1 and 0. The next 3 inputs are the direct or inverse outputs of the flip-flops, selected by 2-channel 4053 multiplexers. Finally, the last three inputs in the multiplexer produce AND functions between any two different flip-flop outputs.

A building-block uses 12 bits, so the length of a chromosome is 36 bits. We have used a standard GA with a population of 100 chromosomes, with a single point crossover, the mutation rate of 1%, proportional selection and elitism.

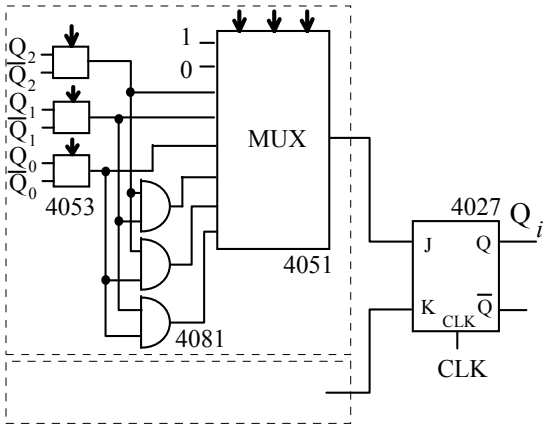

Fig. 9. The building-block for intrinsic digital EHW

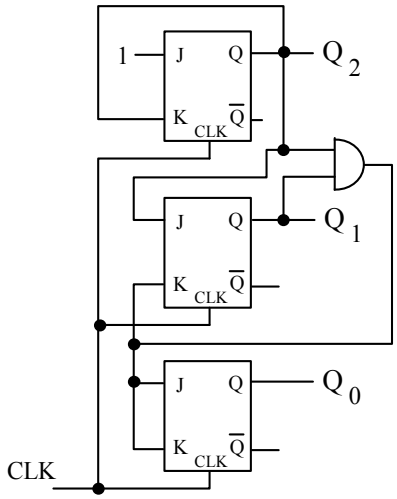

Fig. 10. An example of an evolved counter with 5 states

An example of an evolved counter with 5 states is presented in the Fig.10. We have used all the three building-blocks from the board, including an AND gate.

V. CONCLUSIONS

Evolutionary design is in fact a creative machine for new designs and may be useful for electronic engineers. The experiments presented here display the generation of complexity with very simple rules, and the solving of complex NP-problems with simple GAs. GAs may be useful for automated generation of test vectors and for synthesis of digital and analog circuits. Analog circuit synthesis usually needs more powerful computers, but in the near future this impediment will be certainly avoided.

Continued research on simple, even though unimpressive circuits, is a major factor on the development of EHW. Analysis of such circuits is „far from impractical, and is likely to contribute to the understanding of the properties that evolution can and cannot exploit, and why" ([4]).

In intrinsic EHW experiments, students have the entire control over the architecture or type of basic configurable element, and this may be an advantage over a board equiped with a FPGA device. Evolution may be able to exploit a different system of interconnections or architecture. But is understandable that more complex circuits might be evolved in a FPGA board and these new experiments are a goal for the future.

Future research must be done in this area. Firstly it is important to find a better representation of the circuit in chromosomes, because complex functions need a great number of architecture bits, which directly influences the GA search space. EHW successfully succeeds only when fitness reaches 100% and in huge search spaces this condition may be not always possible. This is the main reason that for the time being the complexity of evolved circuits is so far small.

ACKNOWLEDGMENT

The author would like to thank the Xilinx, Inc. for their academic donation, which consists in Xilinx Integrated Software Environment (ISE) 6.1i software and the Digilab XCRP circuit board provided by Digilent, Inc.

REFERENCES

[1] B. Ali, A. E. A. Almaini, T. Kalganova, "Evolutionary Algorithms and Their Use in the Design of Sequential Logic Circuits", *Genetic Programming and Evolvable Machines*, nr. 5, 2004, pp. 11-29.

[2] C. C. Coello, A. D. Christiansen, A. H. Aguirre, "Use of Evolutionary Techniques to Automate the Design of Combinational Circuits", *International Journal of Smart Engineering System Design*, nr. 4, 2000, pp. 299-314.

[3] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, F. Dunlap, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming", *IEEE Transactions on Evolutionary Computation*, vol. 1, nr. 2, 1997, pp. 109-127.

[4] P. Layzell, *The Evolvable Motherboard – A Test Platform for the Research of Intrinsic Hardware Evolution*, Technical Report CSRP 479, January 1998, School of Cognitive and Computing Sciences, University of Sussex, UK

[5] J. D. Lohn, S. P. Colombano, "A Circuit Representation Technique for Automated Circuit Design", *IEEE Transactions on Evolutionary Computation*, vol. 3, nr. 3, 1999, pp. 205-219.

[6] P. Mazumder, E. M. Rudnick, *Genetic Algorithms for VLSI Design, Layout & Test Automation*, Prentice Hall PTR, 1999.

[7] R. Popa, M. Iliev, V. Nicolau, "Evolutionary Systems for Electronic Engineers", *Proc. of the Scientific Conf. with Int. Participation INTER-ING 2003,* 6-7 November 2003, Tg. Mureş, Romania, vol. II, pp. 187-192.