Evolutionary Design of Gate-Level Polymorphic Digital Circuits

Lukáš Sekanina

Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
sekanina@fit.vutbr.cz

Abstract. A method for the evolutionary design of polymorphic digital combinational circuits is proposed. These circuits are able to perform different functions (e.g. to switch between the adder and multiplier) only as a consequence of the change of a sensitive variable, which can be a power supply voltage, temperature etc. However, multiplexing of standard solutions is not utilized. The evolved circuits exhibit a unique structure composed of multifunctional polymorphic gates considered as building blocks instead. In many cases the area-efficient solutions were discovered for typical tasks of the digital design. We demonstrated that it is useful to combine polymorphic gates and conventional gates in order to obtain the required functionality.

1 Introduction

Evolutionary algorithms have been utilized to design analog as well digital circuits in the recent years [1, 2, 4]. In many cases the evolutionary approach discovered new and creative solutions to hard problems. New solutions, methods and techniques have been developed that designers did not know before. For example, the intrinsic evolution allowed engineers to exploit physical features of reconfigurable devices in order to utilize hardware effectively in a given particular situation [8]. One of these achievements is called *polymorphic electronics*.

The papers [5, 6, 7] show that it is possible to design and effectively implement multifunctional digital gates whose functionally can be controlled in a non-traditional way: by temperature, power supply voltage (Vdd), some external signals etc. As an example, we can mention a novel topology created for the multifunctional NAND/NOR gate which operates as NOR in the case that Vdd=1.8V and as NAND in the case that Vdd=3.3V. No conventional design is available with this logic function controlled by Vdd [5]. There is a great potential for various applications of polymorphic electronics in many areas because the devices composed of these gates are inherently adaptable to the changes of a particular environment and this feature is practically for free; with no reconfiguration overhead.

The available literature describes only the implementations of polymorphic gates. According to the knowledge of the author of this paper no concrete circuits composed of these gates have been reported so far. The design using these

F. Rothlauf et al. (Eds.): EvoWorkshops 2005, LNCS 3449, pp. 185–194, 2005.

unconventional gates seems to be a difficult task because no conventional design technique is available. We believe that compact circuits can be created by using the multifunctional gates as building blocks rather than by trivial multiplexing the outputs of several conventional modules using a polymorphic multiplexer. Hence the use of evolutionary algorithms could be a promising approach for the design of compact and useful adaptive circuits.

The objective of this research is to evolve multifunctional digital combinational circuits using the multifunctional gates as building blocks. It is assumed that suitable multifunctional gates exist. Only the bi-functional gates are considered in this paper; however, the proposed concept is general. The evolved circuits will perform the first required function in the first environment and the second required function in the second environment. The change of their functionality will be determined by the change of a control variable. For example, a circuit should operate as the adder for the temperature 30C and as the multiplier for 200C, i.e. the temperature is the sensitive variable in that case. The problem will be approached using cartesian genetic programming (CGP) which has already demonstrated its success for designing digital electronic circuits [4].

The rest of the paper is organized as follows. Section 2 introduces the area of polymorphic electronics. In Section 3 the polymorphic circuit design problem is formulated formally and a design approach is specified in detail. While Section 4 summarizes the obtained results, Section 5 discusses them. Conclusions are given in Section 6.

2 Polymorphic Electronics

In polymorphic electronics a function change does not require reconfiguration as in traditional approaches in which n different modules and a switch are needed to perform n different functions. Instead the change comes from modifications in the characteristics of components (e.g. in the transistor's operation point) involved in the circuit in response to controls such as temperature, power supply voltage, light, etc. [6]. Polymorphic circuits are able to work in several modes of operation. In the most straightforward approach, there are only two modes (it will also be our case). The existence of digital polymorphic circuits is based on polymorphic gates. Table 1 gives examples of the polymorphic gates reported in literature. Most of them have been designed by means of evolutionary techniques.

The NAND/NOR gate is the most famous example [5]. The evolution obtained a creative novel topology more compact than by multiplexing NAND/NOR gate which is a conventional solution using a standard digital library with external voltage control. No conventional design is available with the logic function controlled by Vdd for this task. The design of a 6-transistor NAND/NOR gate controlled by Vdd is a complex task for a human designer. The circuit was fabricated in a 0.5-micron CMOS technology and silicon tests showed a good correspondence with the simulations. The circuit is stable for $\pm 10\%$ variations of Vdd and for temperatures in the range 20C – 200C.

Table 1. Examples of existing polymorphic gates

Gate	control values	control method	ref.
AND/OR	27/125C	temperature	[7]
AND/OR/XOR	3.3/0.0/1.5V	external voltage	[7]
AND/OR	1.2/3.3V	Vdd	[6]
NAND/NOR	3.3/1.8V	Vdd	[5]

There are also conventionally designed multifunctional gates (e.g. a four transistor XOR/OR/AND gate described in the US patent 042335245); however these are not usually considered as polymorphic (see discussion in [6]).

Potential applications involve special circuits that are able to decrease resolution of digital/analog converters or speed/resolution of a data transmission when a battery voltage decreases, circuits with a hidden/secret function that can be used to ensure security, intelligent sensors, novel solutions for reconfigurable cells and function generators in reconfigurable devices (such as FPGA and CPLD), circuits of random number generators changing distributions of the probability according to the external environment and some others circuits as discussed in [6].

The design of complex polymorphic circuits is a difficult task for engineers since these circuits typically utilize normally unused characteristics of electronic devices and working environment. A. Thompson has shown that unconstrained evolutionary design is able to produce innovative designs that effectively utilize these characteristics [8]. However, only relatively small circuits have been evolved successfully in the field of evolvable hardware so far. Hence the evolution of useful polymorphic circuits (i.e. the circuits larger than a simple gate) is also difficult directly at the transistor level. So we will use the polymorphic gates as building blocks to evolve larger polymorphic circuits.

3 Problem Formulation and Design Approach

In this work we assume that suitable polymorphic gates exist and can be used as standard building blocks. We propose an EA-based approach to design non-trivial combinational modules. An alternative approach could be to design a complete complex module as a polymorphic circuit with an undistinguishable internal structure at the gate level. This is a more challenging task since very compact solutions could be obtained. However, it is difficult to manually control the design process and, furthermore, the evolutionary design is not scalable.

3.1 Problem Formulation

Let P be a set of polymorphic gates. Each of them is able to implement up to K functions (K is specified beforehand) according to a control signal which holds up to K different values. A gate is in *mode* j (and so performing the j-th function) in the case that j-th value of the control signal is activated. For

purposes of this paper we will denote a polymorphic gate as $X_1/X_2/\ldots/X_K$, where X_i is its i-th logic function. For example, NAND/NOR denotes the gate operating as NAND in *mode 1* and as NOR in *mode 2*. Note that some gates can perform less than K different functions; however, their function must be fully defined for each mode. For example, the conventional NAND gate considered for polymorphic circuits will perform the same function in all modes (denoted as NAND/NAND in Section 4).

A polymorphic circuit can formally be represented by the graph $G = (V, E, \varphi)$, where V is a set of vertices, and E is a set of edges between the vertices $E = \{(a, b)|a, b \in V\}$ and φ is a mapping assigning a function (polymorphic gate) to each vertex, $\varphi : V \rightarrow P$. As usually, V models the gates and E models the connections of the gates. A circuit (and also its graph) is in the *mode j* in the case that all gates are in the *mode j*.

Given P and logic functions $f_1 \ldots f_K$ required in different modes, the problem of the polymorphic circuit design at the gate level is formulated as follows: Find a graph G representing the digital circuit which performs functions $f_1 \ldots f_K$ in its modes $1 \ldots K$. Additional requirements can be specified, e.g. to minimize delay, area, power consumption etc.

3.2 Cartesian Genetic Programming

In other words, we are looking for a single circuit topology which will work for all functions of the circuit. The topology will be sought by CGP which is defined over graphs while the standard genetic programming operates with trees [3].

CGP models a reconfigurable circuit, in which digital circuits are evolved, as an array of u (columns) $\times$ v (rows) of programmable elements (gates). The number of the circuit inputs and outputs is fixed. Feedback is not allowed. Each gate input can be connected to the output of some gate placed in the previous columns or to some of the circuit inputs. L-back parameter defines the level of connectivity and thus reduces/extends the search space. For example, if L=1 only neighboring columns may be connected; if L=u, the full connectivity is enabled. We have to define for a given application the following: the number of inputs and outputs, L, u, v and the set of functions performed by programmable elements. Figure 1 shows an example and the corresponding chromosome. Similarly to this example, we will use $v = 1$ and $u = L$ in the proposed experiments.

Miller and Thomson originally used a very simple variant of evolutionary algorithm to produce configurations for the programmable circuit [3]. Our algorithm is based on their evolutionary technique. It operates with the population of 128 individuals; every new population consists of mutants of the best four individuals. Only the mutation operator has been utilized that modifies one randomly selected gene of an individual. In case that evolution has found a solution which produces the correct outputs for all possible input combinations, the number of gates is getting to minimize. Delay is not optimized. The computation is terminated in case that no improvement of the best fitness value has been reported in a given number of last generations (typically in 50000 generations).

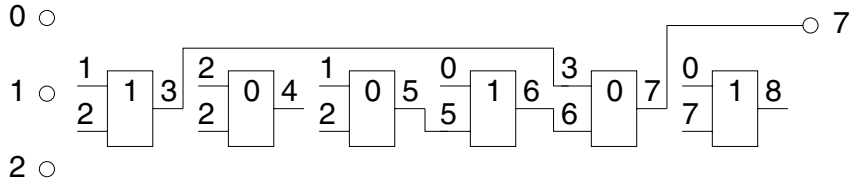

Fig. 1. An example of a 3-input circuit. CGP parameters are as follows: $L = 6$, $u = 6$, $v = 1$, functions AND (0) and OR (1). Gates 4 and 8 are not utilized. Chromosome: 1,2,1, 2,2,0, 1,2,0, 0,5,1 3,6,0, 0,7,1, 7. The last integer indicates the output of the circuit

As this paper deals with bi-functional gates, i.e. with bi-functional polymorphic circuits, the fitness value is obtained as follows:

1. Set all gates of a candidate circuit into *mode 1*.
2. Apply all possible input combinations at the circuit inputs and calculate the number of correct output bits B_1 obtained as response for these input values for the required function f_1.
3. Set all gates into *mode 2*.
4. Apply all possible input combinations at the circuit inputs and calculate the number of correct output bits B_2 obtained as response for these input values for the required function f_2.
5. Calculate $F_1 = B_1 + B_2$.

After achieving the required behavior for f_1 and f_2, the number of gates is being minimized and the fitness value is defined as:

$$F_2 = F_1 + u - g \tag{1}$$

where g denotes the number of gates utilized in a particular candidate circuit and u is the total number of gates available.

4 Experimental Results

This section presents some interesting examples of polymorphic combinational modules that we evolved. We dealt with small combinational circuits (up to 5 inputs and 4 outputs) with various types of bi-functional polymorphic gates. In particular the following circuits will be presented:

- 5-bit parity circuit vs 5-bit median circuit (denoted as 5b-parity-median)
- 2-bit multiplier vs 4-input sorting network (mult2b-sn4b)
- 2-bit multiplier vs 2-bit adder (2b-mult-add)

The following list shows the target circuits (considered as single circuits) and their conventional implementation costs measured in the number of two-input combinational gates.

Table 2. Evolved polymorphic combinational circuits

Circuit	u	run	corr.	opt. gates	gate1	gate2
5b-parity-median	24	200	16	14	NAND/NOR	XOR/XOR
5b-parity-median	20	900	48	13	NAND/XOR	XOR/NOR
Mult2b-sn4b	40	1000	8	25	NAND/NOR	AND/AND
Mult2b-sn4b	40	50	1	27	$(a \vee \overline{b})$/XOR	XOR/$(a \wedge \overline{b})$
2b-mult-add	40	200	11	20	NAND/NOR	OR/XOR
2b-mult-add	40	200	5	23	NAND/NOR	AND/AND

- 4-bit sorting network (sorts a 4-bit input vector) – 18 gates (9 AND, 9 OR)
- 2-bit adder (adds two 2-bit operands, 3-bit output) – 10 gates (a 1b full adder requires 2 XOR, 2 AND, 1 OR)
- 2-bit multiplier (multiplies two 2-bit operands, 4-bit output) – 7 gates (5 AND, 2 XOR)
- 5-bit parity (calculates even parity) – 5 gates (4 XOR, 1 NOT)
- 5-bit median circuit (returns the middle bit of a sorted 5-bit input vector) – 10 gates (5 AND, 5 OR)

Table 2 summarizes the circuits obtained using the algorithm described in Section 3.2. The symbol u denotes the number of gates used in CGP. The column *corr.* gives the number of correct circuits (i.e. perfectly working ones in both modes) out of *run* runs. The minimal number of gates we obtained for a particular circuit is given in the column denoted as *opt. gates*. *Gate1* and *gate2* are the polymorphic gates utilized in the design process. On average, tens of thousands of generations are needed to find a solution. We learned that the correct circuits represent only a fraction of all circuits generated by evolution. In the following figures Fig. 2, 3, 4 and 5, the label **0** denotes the first polymorphic gate and label **1** is the second gate of the two used. The bit 0 is LSB. These figures are taken from our design tool that we have developed.

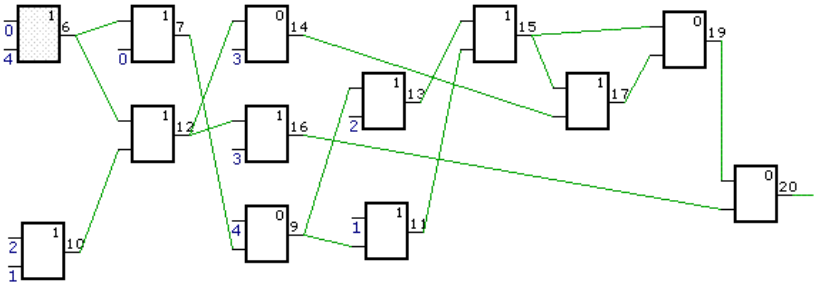

Fig. 2. Polymorphic median – parity circuit (the inputs: 0–4; gates: 0 – NAND/XOR, 1 – XOR/NOR)

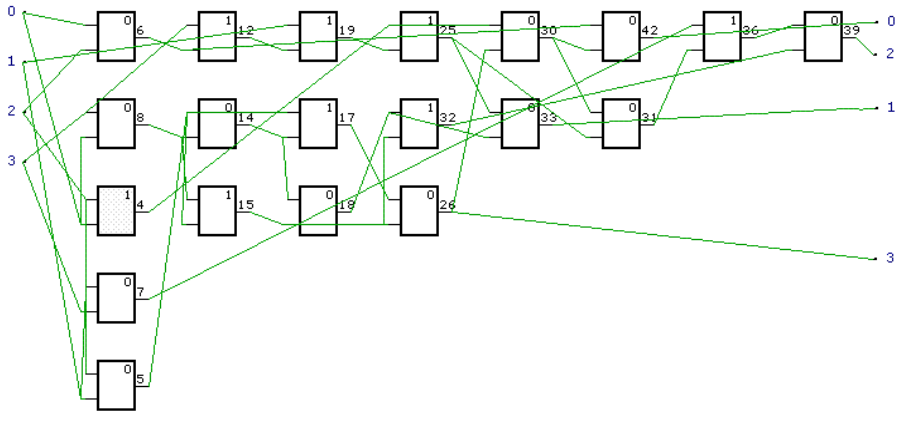

Fig. 3. Polymorphic multiplier – adder circuit (the inputs: A (0–1), B(2–3); the outputs: 0–3 (0–2 in case of adder); gates: 0 – NAND/NOR, 1 – OR/XOR)

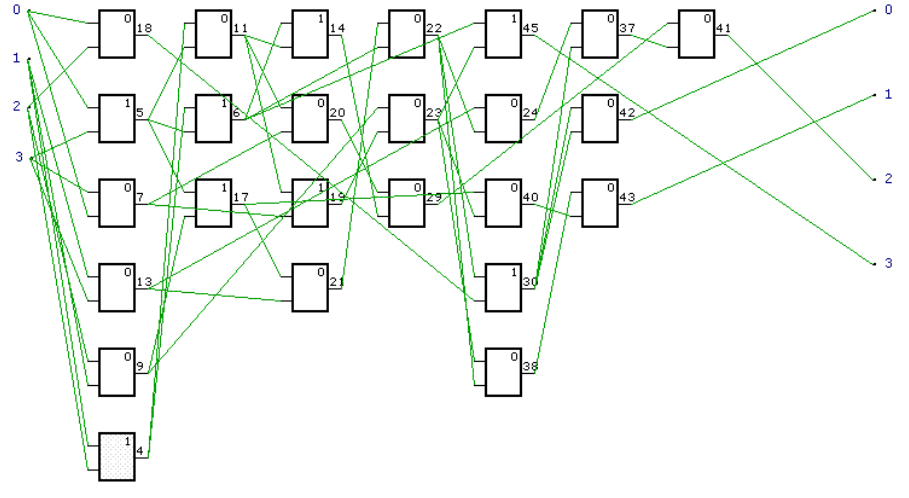

Fig. 4. Polymorphic multiplier – sorting network circuit (the inputs: A (0–1), B(2–3) in case of multiplier; the outputs: 0–3; gates: 0 – NAND/NOR, 1 – AND/AND)

5 Discussion

No useful circuits utilizing only a single type of a polymorphic gate were obtained. For instance, in case of circuits composed only of JPL's NAND/NOR gate, the same functionality of the circuit is obtained by exchanging NAND with NOR or vice versa; however, the circuit operates in the negative logic and with reordered signals. Therefore, no useful additional functionality can be achieved

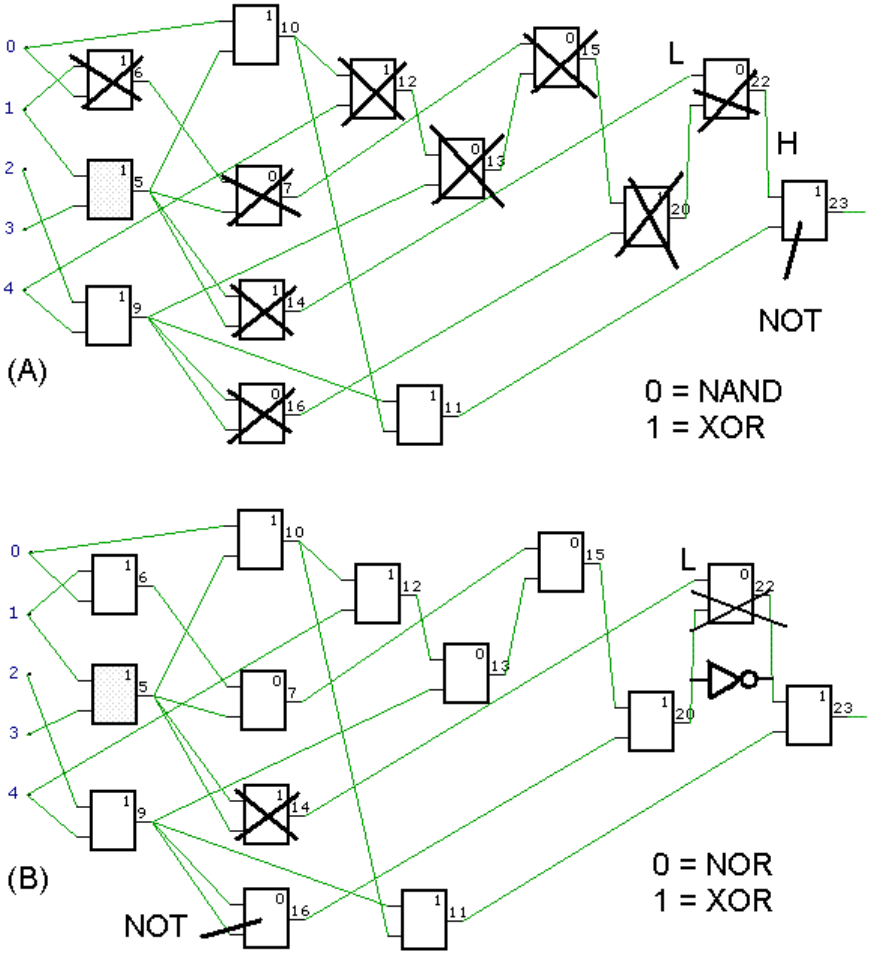

Fig. 5. Analysis of the polymorphic median – parity circuit (gates: 0 – NAND/NOR, 1 – XOR/XOR)

and so more than one type of (polymorphic) gate is needed. As Table 2 indicates, it seems useful to combine polymorphic gates with conventional gates.

The selection of suitable gates is also important for a particular polymorphic circuit. After our experimental work we have identified suitable polymorphic gates for the presented circuits (see Table 2 for concrete circuits). We performed the selection manually; next research will be conducted to use the evolution to accomplish this task. We recognized that only a few combinations of gates are suitable for a given problem; most combinations do not lead to a solution. As an example, we can mention the Mult2b-sn4b problem. We run 100 experiments for each combination of polymorphic gates taken from the following list:

- 1: NAND/NOR and XOR/XOR
- 2: NAND/NOR and OR/OR
- 3: NAND/NOR and $(a \vee \overline{b})/(a \vee \overline{b})$
- 4: NAND/NOR and AND/AND
- 5: NAND/NOR and $(a \wedge \overline{b})/(a \wedge \overline{b})$
- 6: NAND/NOR and OR/XOR
- 7: NAND/NOR and XOR/AND
- 8: $(a \vee \overline{b})$/XOR and XOR/$(a \wedge \overline{b})$
- 9: $(a \vee \overline{b})$/OR and XOR/$(a \wedge \overline{b})$
- 10: $(a \vee \overline{b})$/XOR and NXOR/$(a \wedge \overline{b})$

and obtained the perfect solution only for combinations (4) and (8). An open theoretical issue remains which combinations of polymorphic gates are sufficient to implement the required multifunctional behavior.

In some cases the evolved circuits have shown the lower number of gates when compared to a naive implementation multiplexing of two conventional modules. Note that the conventional two-input multiplexer requires 2 AND, OR and NOT gates. Hence the evolutionary approach seems to be a promising method for the design of polymorphic digital circuits. It is important to mention again that a typical polymorphic electronic device does not have a classical digital control signal. Hence the performed comparison is illustrative only.

In most cases we are not able to understand the topology of the evolved circuits since implementations of required behaviors are entangled. Fig 5 shows the analysis performed on the first circuit listed in Table 2. In the mode 1 (Fig. 5A) the realization is based on a classical implementation of parity circuits. In mode 2 (Fig. 5B) an inefficient implementation of the 5-input median circuit was evolved. For comparison, the implementation of the same behavior depicted in Fig. 2 is much more compact and difficult to decode.

The CGP is not scalable in its basic form, which means that neither our approach can be scaled. In order to illustrate the computational effort of the proposed method, we measured the average time of evolution for 500 runs of the multiplier – sorting network problem. In average 118,985 generations were produced in a single run, which corresponds to 143 seconds of the computational time at Pentium IV (2.6 GHz, 512MB RAM).

6 Conclusions

An evolutionary approach to the design of polymorphic combinational modules has been introduced. Area-efficient implementations have been discovered for various polymorphic gate-level circuits. We learned that it is useful to combine polymorphic gates and conventional gates in order to obtain the required functionality. The results of experiments allow us to predict that the approach could be useful for the design of real-world applications of polymorphic electronics. However, we have utilized hypothetic polymorphic gates. Hence a new development is needed in the basic polymorphic gate design.

Acknowledgment

The research was performed with the Grant Agency of the Czech Republic under No. 102/03/P004 *Evolvable hardware based application design methods.*

References

1. Higuchi, T. et al.: Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine. In: Proc. of the 2nd International Conference on Simulated Adaptive Behaviour, MIT Press, Cambridge MA 1993, p. 417–424
2. Koza, J. R., Keane, M. A., Streeter, M. J.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results. IEEE Intelligent Systems, May/June 2003, p. 25–31
3. Miller, J., Thomson, P.: Cartesian Genetic Programming. In: Proc. of the 3rd European Conference on Genetic Programming, LNCS 1802, Springer Verlag, Berlin 2000, p. 121–132
4. Miller, J., Job, D., Vassilev, V.: Principles in the Evolutionary Design of Digital Circuits – Part I. In: Genetic Programming and Evolvable Machines, Vol. 1(1), 2000, p. 8–35
5. Stoica, A., Zebulum, R. S., Guo X., Keymeulen, D., Ferguson, I., Duong, V.: Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration. IEE Proc.-Comp. Digit. Tech. Vol. 151(4) (2004) 295–300
6. Stoica, A., Zebulum, R. S., Keymeulen, D., Lohn, J.: On Polymorphic Circuits and Their Design Using Evolutionary Algorithms. In Proc. of IASTED International Conference on Applied Informatics (AI2002), Innsbruck, Austria 2002
7. Stoica, A., Zebulum, R., Keymeulen, D.: Polymorphic Electronics. In Proc. of International Conference on Evolvable Systems: From Biology to Hardware, LNCS 2210, Springer Verlag, 2001, p. 291–302
8. Thompson, A., Layzell, P., Zebulum, R., S.: Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. IEEE Transactions on Evolutionary Computation. Vol 3(3), 1999, p. 167–196