Analysis of Reconfiguration Options for a Reconfigurable Polymorphic Circuit

Zdenek Vasicek and Ladislav Capka and Lukas Sekanina
Faculty of Information Technology, Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic
E-mail: {vasicek;icapka;sekanina}@fit.vutbr.cz

Abstract

REconfigurable POlymorphic MOdule (REPOMO) will be a new reconfigurable chip intended for experimental applications of evolvable and polymorphic hardware. In this paper, we analyze various reconfiguration options for this platform with the aim of finding such a reconfiguration subsystem which maximizes the success rate of evolutionary circuit design conducted using REPOMO. An interesting outcome of this analysis is that a relatively high success rate of evolutionary design can be achieved using relatively simple reconfiguration options which have to be implemented in hardware. These results are also relevant for evolutionary circuit design which is performed using Cartesian Genetic Programming.

1 Introduction

In the field of evolvable hardware, evolutionary algorithms are combined with reconfigurable devices in order to create a new generation of hardware – adaptive, self-modifying and self-repairing hardware [5, 1].

Various reconfigurable platforms have been proposed and studied for evolvable hardware in the recent years [4, 17]. In the area of intrinsic evolution of digital circuits, majority of experiments were performed using Field Programmable Gate Arrays (FPGAs). *Intrinsic evolution* means that candidate circuits are evaluated using a physical hardware and in a real environment. On the other hand, the term *extrinsic evolution* is used when candidate circuits are evaluated using a circuit simulator.

In general, a reconfigurable digital circuit consists of an array of programmable logic elements, programmable interconnects and programmable I/O ports. The function of programmable logic elements and their interconnection (i.e. the circuit functionality) is defined using a configuration bitstream. The configuration bitstream is stored in a configuration register (or memory) whose bits directly control the configurable switches and multiplexers of the platform.

Nowadays, there are two main approaches how to build evolvable hardware using FPGAs: (1) Evolution can work at the level of logic blocks available in the FPGA. In other words, it operates directly with the configuration bitstream of the FPGA, for example, using ICAP interface available in some Xilinx FPGAs [17]. This solution requires the knowledge of the internal structure of the FPGA and configuration bitstream. It is usually very efficient in terms of resources; however, it can be slow. (2) A new reconfigurable circuit is created on the top of an FPGA [11]. Using this method, sometimes called Virtual Reconfigurable Circuit (VRC), a very powerful reconfigurable device can be created for a given application. However, its implementation cost can be significant, as everything must be implemented using resources available in the FPGA. In particular, it is necessary to implement programmable elements, multiplexer-based reconfigurable network, programmable I/Os and configuration memory (which is usually implemented as a register array). Another advantage of VRC is that the design is available at the level of HDL source code which can be synthesized for various target platforms.

When one is going to create, for some reasons, a new reconfigurable circuit as an Application-Specific Integrated Circuit (ASIC), the implementation can be based on the HDL source code which describes a particular VRC. The advantage is that a prototype of this new reconfigurable ASIC can be implemented in the FPGA and tested before the chip is fabricated.

In both cases (i.e. VRC and ASIC), designer has to come up with suitable configurable logic blocks and configurable interconnections with respect to the target application. Designer has to also define the organization of the configuration register (memory) in order to maximize the efficiency of evolutionary algorithm. The selection of suitable structure and parameters of the reconfigurable circuit is one of the most difficult tasks evolvable hardware designers are faced with. It can be stated that there is no simple approach which could help the designer to determine suitable structure and parameters of the reconfigurable device for a given application. Similarly to the design of evolutionary algo-

rithm (which includes the selection of the type of search method, population size, genetic operators, parameters of the algorithm etc.) the design of reconfigurable device remains a time-consuming experimental work.

Paper [9] deals with initial considerations and description of a new reconfigurable ASIC called REconfigurable POlymorphic MOdule (REPOMO). This chip consists of a small array of configurable blocks. Each of them can be programmed to perform one of four two-input logic functions. Interconnection of programmable logic blocks is also configurable. Therefore, REPOMO can be used to realize small combinational circuits. The most important feature, which distinguishes REPOMO from other small programmable circuits (such as PALs and GALs) is, that configurable logic blocks also contain *polymorphic* NAND/NOR gates controlled by the level of power supply voltage (Vdd). While the polymorphic gate operates as NAND for a certain level of Vdd, for another level of Vdd the same gate operates as NOR (see Figure 1). The usage of polymorphic gates thus allows a very interesting interaction of digital logic with an external world [13]. These interactions will be investigated using REPOMO in our future research. Note that the concept of polymorphic gates was introduced by Stoica's team at JPL [14, 16, 15].

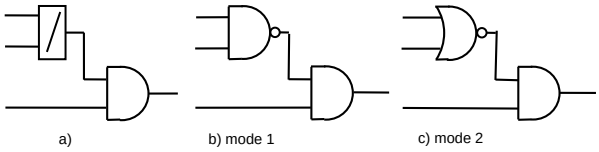

Figure 1. (a) A polymorphic gate-level circuit which contains AND gate and polymorphic NAND/NOR gate, (b) its behavior in the first mode, (c) its behavior in the second mode

Before the fabrication of REPOMO, it is necessary to exactly define the internal structure of the chip. It is supposed that an evolutionary algorithm will generate REPOMOS's configurations in order to find the functionality required by specification. In this paper, we present a set of experiments that were performed to determine suitable parameters of REPOMO. In particular, the goal is to (1) define the set of functions that should be available in configurable blocks and (2) determine reconfiguration options of REPOMO (i.e. possible connection points for inputs of configurable blocks and primary outputs). The proposed method assumes that REPOMO can be modeled as a particular instance of Cartesian Genetic Programming (CGP) – a method widely used to evolve small combinational circuits [8, 6]. Such a specification of REPOMO is sought which maximizes the performance of the evolutionary algorithm over a set of benchmark problems.

The paper is organized as follows. Section 2 describes REPOMO organization. In Section 3, the instance of CGP which will be used to evaluate REPOMO is introduced. While Section 4 presents the experimental setup, Section 5 includes experiments which were performed and their evaluation. Finally, conclusions are given in Section 6.

2 Reconfigurable polymorphic module

2.1 Original version of REPOMO

Figure 2 shows the original architecture of REPOMO according to [9]. REPOMO consists of an array of configurable blocks whose interconnection is also configurable. The initial design assumed that there will be 4 x 4 configurable blocks on the chip. Each of them can be programmed to operate as NAND/NOR, XOR, AND or a simple wire (see Fig. 3). Each gate input can be connected to one of the outputs of the configurable blocks which are placed in the previous two columns. The inputs of configurable blocks of the first and second column can be also connected to the primary inputs. REPOMO utilizes 4 primary inputs and 4 primary outputs. Reconfigurability is achieved using multiplexers. We have already fabricated and tested polymorphic NAND/NOR gates controlled by Vdd [10]. These gates will be also utilized in REPOMO. As configurations of REPOMO will be mostly designed by an evolutionary algorithm, it is required that any (even randomly) generated configuration bitstream is valid. The configuration is stored in a 120 bit shift-register. Note that REPOMO has to work correctly for both levels of Vdd which are used by polymorphic gates. Assumed fabrication technology is AMIS CMOS 0.7.

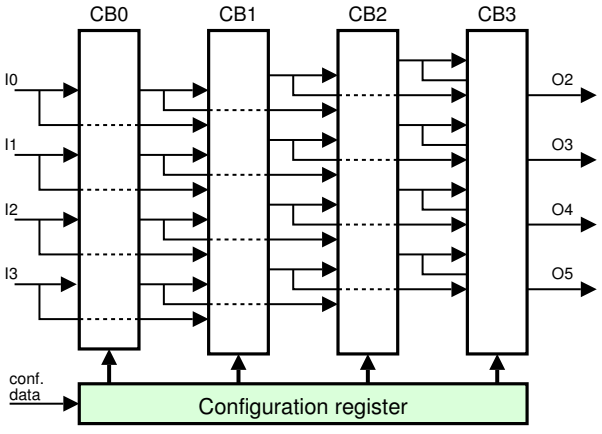

Figure 2. Original design of REPOMO

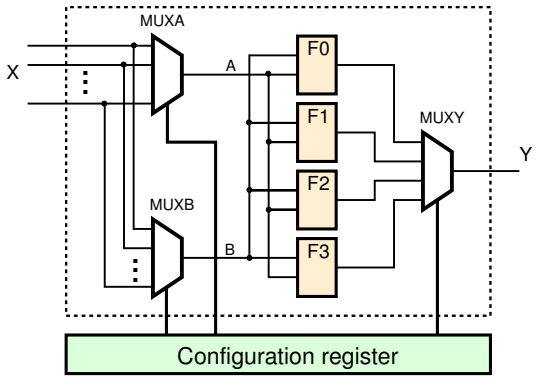

Figure 3. A configurable block which can realize four different functions (F0 - F3). MUXA and MUXB select the inputs for the logic function selected by MUXY.

Table 1. List of logic functions tested in configurable blocks

code	function	description
f0	$Y = 0$	constant
f1	$Y = A$	identity
f2	$Y = A \wedge B$	bitwise AND
f3	$Y = A \vee B$	bitwise OR
f4	$Y = A \oplus B$	bitwise XOR
f5	$Y = \overline{A \wedge B} / \overline{A \vee B}$	polymorphic NAND/NOR

2.2 REPOMO Extended

The original specification of REPOMO was changed. A new version, which we call REPOMOx, will contain more configurable elements and more interconnection options. Figure 4 shows the architecture of REPOMOx which now utilizes 8x8 configurable elements (each of them supports 4 functions), 6 primary inputs and 6 primary outputs. These parameters reflect target applications assumed for this platform.

In order to maximize the performance of evolutionary circuit design in REPOMOx and utilize a reasonable number of configurable bits, it is necessary to determine:

- Logic functions that should be included into configurable blocks: Note the NAND/NOR gate controlled by Vdd has to be included into REPOMO as this is the only polymorphic gate which is now available for us. Table 1 gives all logic functions that we will consider in this analysis. In addition to four logic functions already used in REPOMO, bitwise OR and constant function are also included as a possible replacement for the identity function (wire).

- Connection points for inputs of configurable blocks: The question is whether only neighboring columns of configurable blocks can be interconnected or whether more connectivity is needed.

- Connection points for primary outputs: In REPOMO, the connection of primary outputs was fixed to particular outputs of configurable blocks placed in the last (i.e. rightmost) column. We will investigate whether there should be more options for configuration of primary outputs.

- Connection points for primary inputs: In REPOMO, primary inputs could be connected to the inputs of configurable blocks placed in the first two columns. The question is whether more connection options are needed for primary inputs.

It is evident that additional configuration options imply longer configuration bitstreams which then lead to larger search spaces and, potentially, to slower convergence of the evolutionary algorithm. Additional configuration options will also cost some area on the chip. On the other hand, more configuration options are usually beneficial for the user of the chip. The selection of suitable parameters of REPOMOx will be investigated using CGP.

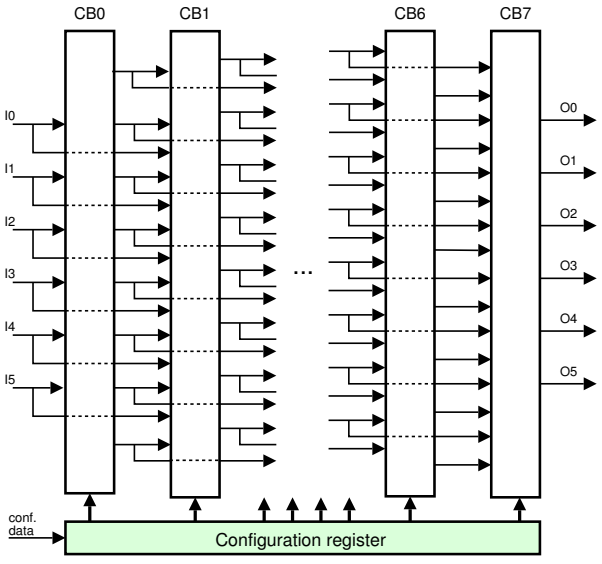

Figure 4. Architecture of modified version of REPOMO – REPOMOx

3 Cartesian Genetic Programming and Its Extension

3.1 Basic version of CGP

In CGP [8, 6], a digital circuit is modeled as an array of u (columns) $\times$ v (rows) of programmable elements (gates). The number of inputs, n_i, and outputs, n_o, is fixed. Feedback is not allowed. Each gate input can be connected either to the output of a gate placed in the previous L columns or to some of primary inputs. The L-back parameter, in fact, 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. Each programmable gate is programmed to perform one of functions defined in the function set. As Figure 5 shows, while the size of chromosome is fixed, the size of phenotype is variable (i.e. some nodes are not used). Every individual is encoded using $u \times v \times 3 + n_o$ integers.

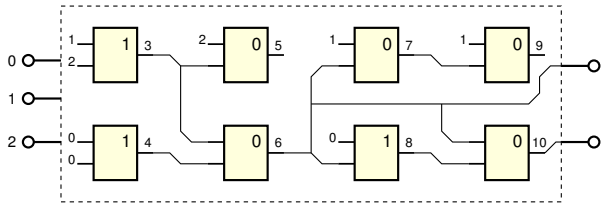

Figure 5. An example of a candidate circuit in CGP with parameters: $L = 3$, $u = 4$, $v = 2$, Function Set = {AND (0), OR (1)}. Nodes 5 and 9 are not utilized. Chromosome: 1,2,1, 0,0,1, 2,3,0, 3,4,0 1,6,0, 0,6,1, 1,7,0, 6,8,0, 6, 10. The last two integers indicate the outputs of the program.

CGP operates with the population of λ individuals (typically, $\lambda = 5 - 20$). The initial population is randomly generated. Every new population consists of the best individual and its mutants. In case when two or more individuals have received the same fitness score in the previous population, the individual which did not serve as a parent in the previous population will be selected as a new parent. This strategy is used to ensure the diversity of population.

The fitness function usually takes the following form: For evolution of logic circuits, all possible input combinations are applied at the candidate circuit inputs, the outputs are collected and the goal is to minimize the difference between obtained truth table and required truth table. In case when the evolution has found a solution which produces correct outputs for all possible input combinations, other parameters, such the number of components or delay are getting to minimize. The evolution is stopped when the best fitness value stagnates or the maximum number of generations is exhausted.

3.2 Proposed extension

CGP was used as a template for implementation of VRCs [19, 18, 11, 3, 2]. In these implementations, L-back parameter is typically set to 1 in order to allow pipeline processing. Similarly, the connection of primary outputs is fixed to some of configurable blocks of the last column.

In order to analyze the effect of the connection of primary outputs, we introduce the so-called output-back (o-back) parameter. This parameter defines a limit in the number of preceding columns, which a primary output may be connected to. For example, if o-back=1 then a primary output can be connected to one the outputs of configurable blocks placed in the last (i.e. rightmost) column. If o-back=u, a primary output may be connected to any configurable block of the array. Let o-back = 0 denote the situation in which the connection of a primary output is fixed (i.e. non-configurable) to just one of configurable blocks of the last column.

Similarly, we introduce i-forward parameter for primary inputs. This parameter defines a limit in the number of columns which a primary input may be connected to.

In next experiments we will show that the setting of L-back, o-back and i-forward significantly influences the convergence of evolutionary algorithm.

4 Experimental setup

In order to find suitable parameters of REPOMOx, CGP with the following setup is used: $u = 8, v = 8, n_i = 6, n_o = 6, \lambda = 5$ and 1–5 genes of the chromosome are mutated. Setting of L-back, o-back and i-forward parameters will be described for each experiment separately. The set of benchmark circuits include:

- 3×3-bit multiplier,
- 6-input/6-output sorting network,
- 6-input majority circuit and
- polymorphic 6-input majority/even parity circuit.

These circuits were chosen with the aim to (1) maximally utilize the available number of inputs in REPOMOx, (2) test different number of outputs and (3) evolve also polymorphic circuits. Note that the 3-bit multiplier may be nowadays considered as a standard test circuit for digital evolvable hardware. The polymorphic 6-input majority/even parity circuit computes the majority function in the first mode of polymorphic gates and the parity function in the second mode of polymorphic gates [12].

The number of generations is 500k for majority circuit and polymorphic majority/parity circuit, 1.5M for multiplier and 1M for sorting network. When a perfectly functional solution is obtained, the number of gates is getting to minimize until the limit in the number of generations is exhausted. Each experiment was repeated 100 times and more than 500k runs of CGP were performed in total.

5 Experimental results

First series of experiments is devoted to searching for a suitable set of functions for configurable elements. Second series of experiments deals with a determination of the most effective configuration options of the chip. Following tables show the average values calculated over all benchmark circuits, all considered function sets and selected values of L-back, o-back and i-forward parameters (a particular setup will be specified in each subchapter).

5.1 Function set

The original function set of REPOMO includes AND, XOR, NAND/NOR and the identity function (denoted as FS1 in Table 2). Firstly, we removed the identity function (and obtained FS2 according to Table 2) because the AND can play the role of identity when its inputs are interconnected. Results summarized in Table 3 show that the usage of FS2 leads to a higher success rate than FS1. Our explanation is that the problem is easier for CGP because the search space is reduced.

The number of functions should be a power of two (2^n) in order to effectively utilize all combinations resulting from n configuration bits devoted to a selection of function in REPOMOx. Table 3 shows that including the OR instead of the identity (denoted as FS6) leads to a much better performance than any other setup that we tested. This high success rate is mainly caused by existence such circuits (sorting network, majority) in our benchmark set which require the OR gate for their effective implementation.

Table 2. Combinations of logic functions tested in configurable blocks

Func. set	Utilized functions (codes)			
FS1	f1	f2	f4	f5
FS2	f2	f4	f5	
FS3	f0	f2	f4	f5
FS4	f2	f2	f4	f5
FS5	f4	f2	f4	f5
FS6	f2	f3	f4	f5

Table 3. Averaged results for different sets of functions: success rate, the number of generations and utilized gates

func.	succ.	# generations				used gates	
set	rate	min	max	avg	dev	min	max
FS1	23	3763	1499k	461k	161k	15	46
FS2	29	3405	1499k	433k	161k	15	54
FS3	17	6518	1499k	406k	131k	15	51
FS4	25	3485	1499k	429k	152k	16	54
FS5	28	2677	1499k	416k	153k	16	55
FS6	71	474	1499k	276k	144k	14	53

5.2 Analysis of interconnection options

In the first experiment of this category, we analyzed all combinations of L-back and o-back parameters. Tables 4 and 5 give success rates averaged over all benchmark circuits, function sets and i-forward parameters. Surprisingly, the best results are obtained neither for L-back = u (as usually used in Miller's papers [6]) nor for L-back=1 (as used in pipeline hardware implementations of VRC [19]). The best success rate is clearly achieved when L-back=2. That is a very good outcome because it tells us that by a small extension of reconfiguration options (which is easily implementable in hardware), significant improvements can be achieved in the success rate. Tables 6 and 7 show averaged results for one of benchmark circuits – the majority/parity circuit. Figure 6 shows example of evolved polymorphic majority/parity circuit.

Experimental results indicate that the value of o-back parameter should be very low (Tables 4 and 5). The best results were obtained if o-back=0, i.e. the connection of primary outputs is not configurable at all. Tables 8 and 9 provide detailed results for L-back=2. Our explanation of this phenomenon is that mutations of output connections are very disruptive; hence, it is better to have a fixed connection of primary outputs. In addition, the search space is also reduced. This outcome is again important from the implementation point of view. It simply says that reconfiguration of output connections is not useful and should not be implemented.

In order to determine, whether it is useful to allow the connection of primary inputs either to all configurable blocks or to some of them only, we analyzed the influence of i-forward parameter. Table 10 shows averaged experimental results for different values of the i-forward parameter. A reasonable compromise is to allow the configurable elements that are placed in the first two columns to connect their inputs to the primary inputs. Increasing the value of i-forward parameter requires including much larger multiplexers to configurable elements which costs a considerable area on the chip.

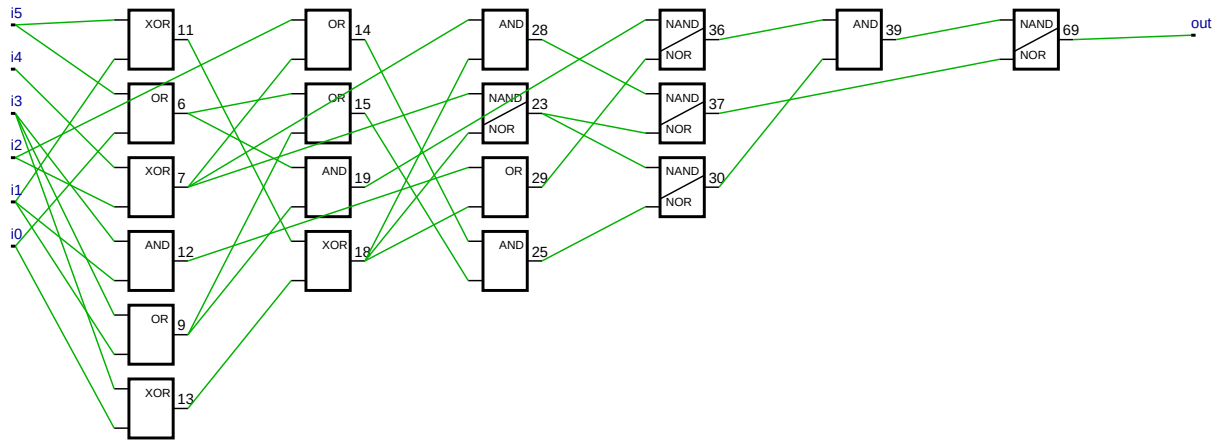

Figure 6. Example of evolved six-bit polymorphic majority/parity circuit

Table 4. The average number of correctly evolved circuits for different combinations of o-back and L-back.

L-back	o-back 0	1	2	3	4	5	6	7	8
1	36	36	36	35	33	34	32	33	35
2	50	49	49	46	46	46	46	46	46
3	44	44	41	40	40	39	38	39	38
4	36	36	34	32	30	31	31	31	31
5	32	31	29	28	26	26	26	26	27
6	29	29	27	26	25	24	24	24	24
7	29	29	27	25	24	23	23	23	23
8	30	29	27	25	23	23	23	23	23

Table 5. The average number of correctly evolved circuits for different combinations of o-back and L-back. Only the FS6 is considered.

L-back	o-back 0	1	2	3	4	5	6	7	8
1	65	63	67	68	67	67	66	67	69
2	80	79	80	80	78	80	81	80	80
3	78	79	77	79	78	78	76	76	78
4	75	74	74	73	72	73	72	72	72
5	71	72	71	69	69	69	70	68	69
6	70	70	69	67	68	67	66	66	67
7	70	70	70	67	67	67	66	65	67
8	71	71	69	67	65	66	67	66	66

5.3 Summary of Results

A new version of REPOMO, whose specification requires an array of 8x8 configurable blocks, 6 inputs, 6 outputs and 4 functions per a configurable block was described. Preformed experiments suggest that the function set available in configurable gates should contain NAND/NOR, AND, OR and XOR. In order to maximize the success rate of the evolutionary design, the following reconfiguration capabilities are recommended:

- Primary inputs should be available at first two columns of configurable blocks (i-forward=2). This requires the usage of 8-input multiplexers in the first column and 16-input multiplexers in the second column of configurable blocks.
- Inputs of configurable blocks should be connectable to the outputs of configurable blocks placed in two preceding columns (L-back=2). This requires the usage of 16-input multiplexers in configurable blocks.
- Primary outputs should be directly connected (without the possibility of reconfiguration) to selected configurable blocks of the last column (o-back=0).

6 Conclusions

In this paper, a new version of REPOMO platform was described. We analyzed various reconfiguration options for this platform with the aim of finding such a reconfiguration subsystem which maximizes the success rate of evolutionary circuit design conducted using REPOMO. Performed analysis of the o-back, L-back and i-forward parameters extends the analysis of CGP published previously (for example, [7, 18, 6]). An interesting outcome is that a relatively high success rate of evolutionary design can be achieved using relatively simple reconfiguration options in hardware. These results are very important for the final design of REPOMOx's architecture.

Table 6. The average number of correctly evolved circuits for different combinations of o-back and L-back for the majority/parity circuit.

	o-back								
L-back	0	1	2	3	4	5	6	7	8
1	40	38	39	35	32	34	31	32	32
2	34	34	34	30	30	29	30	28	29
3	18	19	19	17	16	17	14	14	15
4	10	11	10	10	9	9	9	8	9
5	7	7	8	5	6	6	6	5	5
6	6	7	6	5	5	5	4	3	4
7	6	5	6	5	4	4	4	3	3
8	7	7	6	5	3	4	4	3	4

Table 7. The average number of correctly evolved circuits for different combinations of o-back and L-back for the majority/parity circuit. Only the FS6 is considered.

	o-back								
L-back	0	1	2	3	4	5	6	7	8
1	68	65	67	68	64	62	64	62	62
2	69	66	65	67	66	67	69	66	63
3	48	51	50	50	50	47	46	44	48
4	35	36	35	34	33	31	32	29	29
5	29	28	27	23	23	22	21	20	20
6	23	25	24	20	18	19	19	15	18
7	23	22	24	18	18	17	14	15	15
8	23	23	22	19	16	16	18	14	16

Table 8. Averaged success rates, generations and used gates for L-back=2 and different values of o-back

	succ.	# generations				used gates	
o-back	rate	min	max	avg	dev	min	max
0	50	1334	1492k	371k	186k	14	54
1	49	1419	1499k	372k	181k	14	55
2	49	539	1497k	373k	181k	14	54
3	46	489	1492k	376k	173k	14	53
4	46	1056	1493k	366k	179k	14	51
5	46	1393	1497k	398k	172k	14	52
6	46	474	1499k	381k	177k	14	49
7	46	1458	1495k	382k	168k	14	49
8	46	1135	1477k	365k	171k	14	50

Table 9. Averaged success rates, generations and used gates for L-back=2 and different values of o-back. Only the FS6 is considered.

	succ.	# generations				used gates	
o-back	rate	min	max	avg	dev	min	max
0	80	1334	1492k	250k	144k	14	53
1	79	1419	1476k	280k	140k	14	47
2	80	539	1480k	254k	139k	14	52
3	80	489	1423k	174k	131k	14	47
4	78	1056	1468k	169k	118k	14	49
5	80	1393	1488k	212k	145k	14	49
6	81	474	1496k	172k	123k	14	48
7	80	1458	1492k	251k	120k	14	49
8	80	1135	1448k	248k	150k	14	45

Acknowledgements

This research was partially supported by the Grant Agency of the Czech Republic under No. 102/06/0599 *Methods of Polymorphic Digital Circuit Design* and the Research Plan No. MSM 0021630528 *Security-Oriented Research in Information Technology*.

References

[1] H. de Garis. Evolvable hardware – genetic programming of a darwin. In *International Conference on Artificial Neural Networks and Genetic Algorithms*, Innsbruck, Austria, 1993. Springer Verlag.

[2] K. Glette, J. Torresen, M. Yasunaga, and Y. Yamaguchi. On-Chip Evolution Using a Soft Processor Core Applied to Image Recognition. In *The 1st NASA/ESA Conference on Adaptive Hardware and Systems*, pages 373–380, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[3] D. Gwaltney and K. Dutton. A VHDL Core for Intrinsic Evolution of Discrete Time Filters with Signal Feedback. In *Proc. of the 2005 NASA/DoD Conference on Evolvable Hardware*, pages 43–50, Washington D.C., USA, 2005. IEEE Computer Society.

[4] T. Higuchi, Y. Liu, and X. Yao. *Evolvable Hardware*. Springer, 2006.

[5] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya. Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine. In *Proc. of the 2nd International Conference on Simulated Adaptive Behaviour*, pages 417–424. MIT Press, 1993.

[6] J. Miller, D. Job, and V. Vassilev. Principles in the Evolutionary Design of Digital Circuits – Part I. *Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[7] J. F. Miller and S. L. Smith. Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2):167–174, 2006.

[8] J. F. Miller and P. Thomson. Cartesian genetic programming. In *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 121–132, Edinburgh, 15-16 2000. Springer-Verlag.

[9] R. Ruzicka and L. Sekanina. Evolutionary circuit design in repomo - reconfigurable polymorphic module. In *Proceedings of the Second IASTED International Conference on Computational Intelligence*, pages 237–241. ACTA Press, 2006.

[10] R. Ruzicka, L. Sekanina, and R. Prokop. Physical demonstration of polymorphic self-checking circuits. In *IEEE International On-Line Testing Symposium 2008*, page 6. IEEE Computer Society – Accepted, 2008.

Table 10. Averaged success rates, generations and used gates for different values of i-forward parameter. Only the FS6 is considered.

	succ.	# generations				used gates	
i-fwd	rate	min	max	avg	dev	min	max
1	60	474	1479k	281k	109k	14	52
2	81	489	1498k	236k	149k	14	53
3	80	461	1498k	259k	151k	13	55
4	78	622	1498k	271k	158k	14	52
5	77	468	1496k	281k	164k	14	51
6	75	717	1497k	298k	168k	14	53
7	74	1800	1499k	305k	171k	14	53
8	73	1757	1499k	310k	174k	14	52

[11] L. Sekanina. *Evolvable components: From Theory to Hardware Implementations*. Natural Computing. Springer-Verlag Berlin, 2004.

[12] L. Sekanina, L. Starecek, Z. Gajda, and Z. Kotasek. Evolution of multifunctional combinational modules controlled by the power supply voltage. In *Proc. of the 1st NASA/ESA Conference on Adaptive Hardware and Systems*, pages 186–193. IEEE Computer Society, 2006.

[13] L. Sekanina, L. Starecek, Z. Kotasek, and Z. Gajda. Polymorphic gates in design and test of digital circuits. *International Journal of Unconventional Computing*, 4(2):125–142, 2008.

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

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

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

[17] A. Upegui and E. Sanchez. Evolvable FPGAs. In S. Hauck and A. Dehon, editors, *Reconfigurable Computing*, pages 725–752. Morgan Kaufmann, 2008.

[18] Z. Vasicek and L. Sekanina. Evaluation of a new platform for image filter evolution. In *Proc. of the 2007 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 577–584. IEEE Computer Society, 2007.

[19] Z. Vasicek and L. Sekanina. An evolvable hardware system in Xilinx Virtex II Pro FPGA. *International Journal of Innovative Computing and Applications*, 1(1):63–73, 2007.