Cellular Automata-Based Development of Combinational and Polymorphic Circuits: A Comparative Study

Michal Bidlo and Zdenek Vasicek

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 61266 Brno, Czech republic
{bidlom,vasicek}@fit.vutbr.cz

Abstract. Cellular automata-based evolutionary development is presented for the design of single-function and polymorphic (two-function) combinational circuits. The impact of evolution of the cellular automaton initial state on the success rate of the evolved solutions is investigated. The experiments show that it is more suitable to fix a proper initial state in order to increase the successfulness and speed of evolution. The proposed developmental model is capable to design a wide range of both single-function and polymorphic circuits.

1 Introduction

In recent years, many approaches were introduced for the evolutionary design of digital circuits. Probably the most popular approach is Miller's cartesian genetic programming [1]. His approach represents typical direct mapping between genotypes and phenotypes in the genetic algorithm for the evolution of digital circuits. Developmental systems represent an other class of systems that may be utilized for the circuits design. For example, Miller's developmental cartesian genetic programming [2], Tufte's FPGA-based approach for evolving functionality in cellular systems [3] or Gordon's developmental approach in evolvable hardware [4] represent instances of evolutionary developmental systems.

1.1 Polymorphic circuits

Polymorphic circuits, introduced by Stoica's team at JPL, are in fact multifunctional circuits [5]. The change of their behavior 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. (they are able to work in several modes of operation corresponding to different operational conditions). It means that polymorphic circuits are inherently multifunctional. The change of function does not require any reconfiguration or multiplexing.

Some applications of polymorphic circuits are discussed in [6]. Polymorphic electronics should allow engineers to build inherently adaptable digital circuits.

By changing the temperature, Vdd or some other conditions a circuit can change its functionality immediately, with no reconfiguration overhead. The potential applications include: 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 and novel solutions for reconfigurable cells and function generators in reconfigurable devices (such as FPGA and CPLD).

The design of polymorphic circuits is considered as a crucial problem because these circuits typically utilize normally unused characteristics of electronic devices and working environment; conventional design techniques are not able to deal with that. Therefore, evolutionary techniques have often been utilized. Thompson has shown that unconstrained evolutionary design is able to produce innovative designs that effectively utilize that characteristics [7]. Stoica et al. introduced an evolutionary approach for the design of polymorphic gates at the transistor level. Sekanina proposed a method for the evolutionary design of gate-level digital polymorphic circuits in which polymorphic gates are considered as building blocks [8]. In his paper, Miller's cartesian genetic programming was applied. For instance, a circuit was evolved operating as 2-bit adder in environment E1. This circuit can also work as 2-bit multiplier in environment E2. A typical feature of polymorphic gate-level circuits is that their topology (i.e. connection of components) is fixed; however, the components can change the functionality. In other approach, Bidlo et al. applied an instruction-based development for the evolutionary design of arbitrarily large polymorphic sorting networks [9]. The polymorphic sorting networks are able to sort the input sequences into the non-decreasing order in environment E1 and into the non-increasing order in environment E2. There is no innovation; human designer would construct the circuit in the same way. However, the utilization of resources is potentially interesting. The implementation of traditional AND as well as OR gate costs 6 transistors in the standard CMOS technology. The cost of polymorphic AND/OR gate controlled by temperature is also 6 transistors [5]. If one were able to build OR/AND gate with the same cost, the resulting polymorphic sorting network would consist of the same number of transistors as the original one whose behavior cannot be changed.

1.2 Development

In nature, the development is a biological process of ontogeny representing the formation of a multicellular organism from a zygote. It is influenced by the genetic information of the organism and the environment in which the development is carried out.

In the area of computer science and evolutionary algorithms in particular, the computational development has been inspired by that biological phenomena. Computational development is usually considered as a non-trivial and indirect mapping from genotypes to phenotypes in an evolutionary algorithm. In such case the genotype has to contain a prescription for the construction of target

object. While the genetic operators work with the genotypes, the fitness calculation (evaluation of the candidate solutions) is applied on phenotypes created by means of the development. The principles of the computational development together with a brief biological background and selected application of this bio-inspired approach are summarized in [10].

1.3 Cellular automata

Cellular automata, originally invented by Ulam and von Neumann in 1966 [11], represent a mathematical model originaly intended as a formal framework to study the behavior of complex systems, especially the questions of whether computers can self-replicate. Cellular automata may also be considered as a biologically inspired technique to model and simulate the cellular development.

The fundamental principles of CAs is as follows. A cellular automaton (CA) consists of a regular structure of cells, each of which can occur in one state from a finite set of states. The states are updated synchronously in parallel according to a local transition function. Let us call a developmental step of the CA the synchronous update of all the cells of the CA. The next state of a cell depends on the combination of states in the cellular neighborhood. In this paper we consider the cellular neighborhood consisting of the cell and its two immediate neighbors. Moreover, cyclic boundary conditions will be considered, i.e. the left neighbor of the first cell is the last cell of the CA. Similarly, the right neighbor of the last cell is the first cell of the CA. The local transition function defines a next state of the cell being updated for every possible combination of states in the cellular neighborhood. Let us denote $c_1c_2c_3 \rightarrow c_n$ as a rule of the local transition function, where $c_1c_2c_3$ represents the combination of states of the cells in the cellular neighborhood and c_n denotes the next state of the middle cell. In case of uniform cellular automata, the local transition function is identical for all the cells.

Cellular automata have been applied to solve many complex problems in different areas. A detailed survey of the principles and analysis of various types of cellular automata and their applications is summarized in [12]. Sipper [13] investigated the computational properties of CAs and proposed an original evolutionary design method for the "programming" the cellular automata called cellular programming. He demonstrated the successfulness of this approach to solve some typical problems related to the cellular automata, e.g. synchronization, ordering or the random number generation. In the recent years, scientists have been interested in the design of cellular automata for solving different tasks using the evolutionary algorithms. Dellaert et al. introduced a method for the evolutionary development of complete autonomous agents using random boolean networks. In fact, random boolean network can be understood as a binary cellular automaton whose cellular neighborhood is not limited by the structure of the automaton. The successful evolutionary development was presented that constructs complete autonomous agents which perform the line following task [14]. Corno et al. applied the cellular automaton as a generator of the binary test vectors for BIST (Built-In Self Test) units to detect stuck-at faults inside

a Finite State Machine circuit. According to the results presented in [15], this method is able to overcome the fault coverage than that can be achieved using current engineering practice. Miller investigated the problem of evolving a developmental program inside a cell to create multicellular organism of arbitrary size and characteristic. He presented a system in which the organism organizes itself into well defined patterns of differentiated cell types (e.g. the French Flag) [16]. Tufte and Haddow utilized a FPGA-based platform of Sblocks [17] for the online evolution of digital circuits. The system actually implements a cellular automaton whose development determines the functions and interconnention of the Sblock cells in order to realize a function. Note that the evolutionary algorithm is utilized to design the rules for the development of the cellular automaton [3].

1.4 Objectives of the paper

Since the polymorphic gates were discovered and the concept of polymorphic circuits was introduced [5], the design of polymorphic circuits represents a popular research domain. In order to identify the possibilities of different design methods with respect to the relations to the conventional circuits, more research is needed in this area.

In this paper, an evolutionary developmental method based on uniform one-dimensional cellular automata is presented for the design of combinational circuits at the gate level. The experiments are devoted to design (1) single-function combinational circuits and (2) two-function polymorphic circuits consisting of conventional and polymorphic gates. The results of both sorts of experiments are compared. A genetic algorithm is utilized to design a suitable cellular automaton by means of which a given circuit is developed. It is demonstrated that the presented developmental approach is able to design both classes of combinational circuits. Moreover, the impact of evolution of initial state of the cellular automaton on the success rate and computational effort of the evolutionary process is investigated.

2 Development of Digital Circuits

In order to design combinational circuits using a cellular automaton, a logic gate is assigned to each rule of the local transition function. Therefore, the rule of the CA that is capable to generate the circuits is in the form $c_1c_2c_3 \rightarrow c_n : f\ i_1\ i_2$, where the part on the right of the colon specifies the function (f) of the gate and the indices of its two inputs (i_1, i_2). This specification corresponds to the cellular automata generating single-function combinational circuits. In order to generate polymorphic circuits, it is needed to specify other functions the circuit should be able to perform. In this paper, only two-function polymorphic circuits will be considered. The form of a rule of the local transition function for polymorphic circuits possesses the form $c_1c_2c_3 \rightarrow c_n : f_1\ f_2\ i_1\ i_2$, where f_1 and f_2 denote the functions of the polymorhic gate and the rest of the rule has the same meaning as stated above.

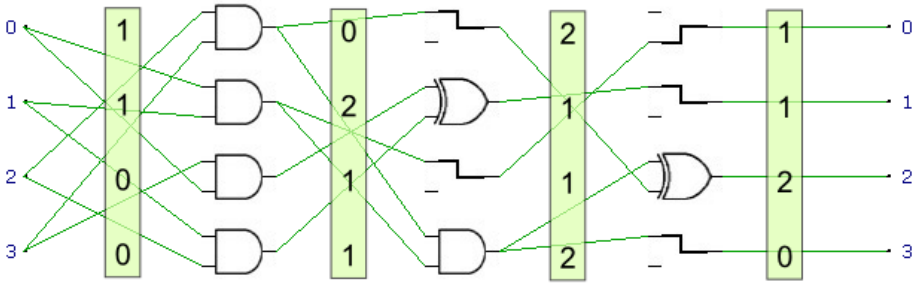
A gate is generated by each cell during the development of the CA. The gate to be generated is specified by the rule that is applied to determine the next state of the cell depending on the combination of states in the cellular neighborhood. Therefore, one level of the circuit is generated in one developmental step of the CA. The number of cells of the CA equals to the number of primary inputs of the circuit. In case of the first developmental step, the gates being generated connect their inputs to the primary inputs of the target circuit and the outputs of the gates generated in the last step are connected directly to the appropriate primary outputs of the circuit. Note that the utilization of the outputs depends on the type of the circuit to be developed. The inputs are referenced by the indices. Similarly, the outputs of the gates generated by the cells are denoted by the indices of the cells in the CA (the indices are identical to the indices of the primary inputs of the circuit). The inputs of the gates generated since the second step of the CA are connected to the outputs of the gates generated in the previous developmental step (in fact, it corresponds to $l - back$ parameter in Miller's cartesian genetic programming whose value equals 1 in this case). This restriction reduces the search space substantially while preserving the possibility of generating various instances of circuits for the comparative study presented in this paper.

Table 1 shows the set of basic logic gates utilized for the development, which also constitute the building blocks for two-function polymorphic gates.

Gate	**Inputs**	**Description**
0: AND	a, b	two-input AND gate
1: OR	a, b	two-input OR gate
2: XOR	a, b	two-input exclusive-OR gate
3: $NAND$	a, b	two-input inverted AND gate
4: NOR	a, b	two-input inverted OR gate
5: $NXOR$	a, b	two-input inverted XOR gate
6: IDA	a, x	one-bit buffer (identity function) of the first input
7: IDB	x, b	one-bit buffer (identity function) of the second input

Table 1. Gates utilized for the development. Note that x represents an unused input. Every arbitrary pair of gates may constitute a polymorphic gate utilized for the development of polymorphic circuits (including the gates with two identical functions which actually represent conventional logic gates in polymorphic circuits).

Figure 1a shows an example of the cellular automaton generating two-level 2x2-bit combinational multiplier. The primary inputs of the multiplier and the cells of the CA are denoted by the indices 0, 1, 2 and 3. The development of the circuit is performed as follows. At the beginning of the development, the CA is initialized by a suitable initial state, in this case 1 1 0 0. Considering the cyclic boundary conditions, the state of each cell is updated according to the local



(a)

$0\ 0\ 1 \rightarrow 1 : AND\ 1\ 2$	$0\ 1\ 1 \rightarrow 0 : AND\ 2\ 3$
$0\ 2\ 1 \rightarrow 1 : XOR\ 2\ 3$	$1\ 0\ 0 \rightarrow 1 : AND\ 3\ 0$
$1\ 0\ 2 \rightarrow 2 : IDA\ 0\ 1$	$1\ 1\ 0 \rightarrow 2 : AND\ 0\ 1$
$1\ 1\ 2 \rightarrow 2 : XOR\ 3\ 0$	$1\ 2\ 2 \rightarrow 0 : IDA\ 3\ 3$
$2\ 1\ 1 \rightarrow 1 : IDA\ 1\ 0$	$2\ 2\ 1 \rightarrow 1 : IDB\ 0\ 2$

(b)

Fig. 1. Example of the circuit development using a cellular automaton from the initial state 1100: (a) developed 2x2-bit multiplier, (b) rules of the local transition function of the CA applied to the development of the multiplier.

transition function (see Fig. 1b). During the first developmental step, the actual state 1 of the first (top) cell is updated according the rule $0\ 1\ 1 \rightarrow 0 : AND\ 2\ 3$. The AND gate is generated having its inputs connected to the primary inputs 2 and 3. The next state of the second cell is computed according to the rule $1\ 1\ 0 \rightarrow 2 : AND\ 0\ 1$, generating the AND gate whose inputs are connected to the primary inputs 0 and 1. The same principle is applied to generate the other gates in the first developmental step. After the first step the state of the CA is 0 2 1 1. In the second developmental step, for instance, the XOR 2 3 is generated by the rule $0\ 2\ 1 \rightarrow 1 : XOR\ 2\ 3$ and the identity function of the first gate input (IDA 1) is generated according to the rule $2\ 1\ 1 \rightarrow 1 : IDA\ 1\ 0$. Note that the input index 0 is meaningless since the IDA gate passes only the first input (labeled by 1) which is connected to the output of the AND gate generated by the cell 1 in the previous developmental step. After the next (and last, third) developmental step, the circuit is completed and the outputs of the gates generated in this step represent the primary outputs of the multiplier.

3 Evolutionary System Setup

The simple genetic algorithm was utilized for the evolutionary design of the cellular automaton that generates a specified circuit. Basically, there are two sorts of experiments presented regarding to the development of single-function combinational circuits and two-function polymorphic combinational circuits. Moreover, each of these sorts of experiments is divided into two subparts that differ in

the subject of evolution of the CA. In the first sort, the initial state and the local transition function of the CA is evolved. In the second sort, only the local transition function is evolved and the initial state is fixed at the beginning of evolution according to the experience from the evolution of both parts of the CA. The experiments showed that the number of different combinations of states in cellular neighborhods of the CA should be maximized in order to develop a working circuits. This is not surprising sice there are more rules to be applied, hence more different gates are to be generated and more complex behavior of the CA is possible to exhibit a satisfactory complexity required for the target circuit. If the number of different combinations in the initial state was low, the CA would be not able to develop complex structures and no working circuit would emerge.

The chromosome of the genetic algorithm contains the rules of the local transition function and the initial state of the CA (if intended to evolve). The initial state is encoded in the chromosome as a finite sequence of integers representing the initial states of cells. An encoded rule of the local transition function consists of the next state, a function of the gate to be generated and two indices of inputs of the gate. The structures of chromosomes regarding all the sorts of experiments are shown in Figure 2. The index (position in the genome) is specified implicitly by means of the value expressed by the number representing the combination of states in the cellular neighborhood. The base of this number equals the number of possible states of the cell. Therefore, if we consider the general form of the rule $c_1\ c_2\ c_3 \rightarrow c_n : f\ i_1\ i_2$, only the part on the right of the arrow is encoded in the genome. For example, if a cellular automaton with 2 different states and the cellular neighborhood consisting of 3 cells ought to be evolved, there are 2^3 rules of the local transition function. Consider the rule $0\ 1\ 1 \rightarrow 0 : OR\ 0\ 1$. Since the combination of states 0 1 1 corresponds to the binary representation of number 3, this rule will be placed in the chromosome at the position 3 of the local transition function.

In all the experiments, the population consists of 200 chromosomes which are initialized randomly at the beginning of evolution. The chromosomes are selected by means of the tournament operator with the base 4. The crossover operator is not applied. The following mutation operator is utilized. In each chromosome selected by the tournament operator, 5 genes are chosen randomly and each of them is mutated with the probability 0.96. A gene is understood as a single value representing the state or the gate function or the input index. If the initial state of the CA is mutated, then the two initial states being evolved are compared in order to avoid the evolution of two identical initial states.

The fitness function is calculated as the number of correct output bits of the target circuit using all the binary input test vectors. For example, if a 4-input circuit ought to be developed, there are 2^4 test vectors. Therefore, the fitness of a perfect solution possessing 4 primary outputs equals $4 \cdot 2^4 = 64$. If no solution is evolved in a given limit of the number of generations (which is specific for different sorts of experiments) the evolution is finished.

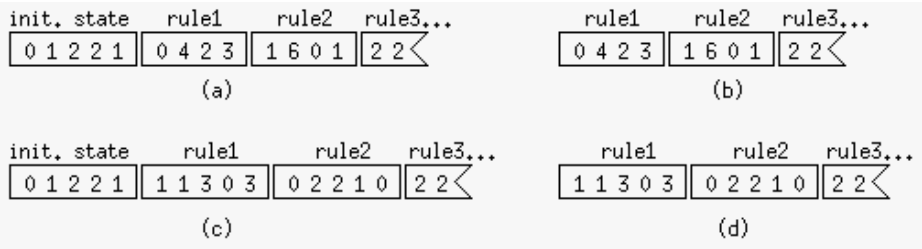


Fig. 2. Structures of the chromosomes for experiments regarding to the design of (a) combinational circuits including evolution of the initial state of the CA, (b) combinational circuits dealing with a fixed initial state, (c) polymorphic circuits including evolution of the initial state of the CA, (b) polymorphic circuits dealing with a fixed initial state. In cases (a) and (b) a rule of the local transition function consists of the next state, logic function of the gate to be generated and indices of two inputs of the gate respectively. In cases (c) and (d), the next state is followed by two logic functions of a polymorphic gate and indices of its two inputs.

4 Experimental Results

Experiments have been conducted in order to demonstrate the ability of the proposed developmental system to design single-function and two-function polymorphic combinational circuits at the gate level. Moreover, the initial state of the CA and its impact on the circuit development has been investigated. Two different sorts of experiments have been conducted for both the design of single-function and polymorphic circuits. In the first sort, the initial state has been fixed at the beginning of evolution and only the local transition function has been evolved. In the second sort, both the initial state and the local transition function have been evolved.

If the initial state is fixed, it is important to choose a proper one from which a working circuit may be developed. For the number of different combinations of states in the cellular neighborhood that is lower than the number of cells of the CA, the circuit development is expected to be very difficult (or even impossible) because the number of different gates that may be generated by the cells is low (more identical rules of the local transition function have to be applied) and more complex circuit structure can not be developed. This assumption was confirmed by the conducted experiments. Therefore, the initial state of the CA is chosen in which cells possess mutually different combinations of states in their neighborhood.

For example, if cyclic boundary conditions are considered, the initial state $1, 1, 0, 0$ of a four-cell CA develops according to the rules based on combinations of states in the cellular neighborhoods 011, 110, 100, 001. As evident, the next state of each cell is determined according to different rule, different gates may potentially be generated by each cell in a single step (i.e. more complex circuit structure) and therefore sequences possessing this feature is considered as proper initial states with respect to the information stated in the previous paragraph.

4.1 Development of single-function combinational circuits

The first sort of experiments deals with the development of single-function combinational circuits by means of a cellular automaton. In all these experiments, the whole set of gates shown in Table 1 was considered for the development. Thousand independent experiments were conducted for each type of circuit and for the two approaches to the specification of initial state of the CA (evolved and fixed. If no working solution is found in one million of generations, the evolution is finished.

The results of evolution are summarized in Table 2. The abbreviations in the first column of Table 2 correspond to the following circuits: median circuit (med), multiplier (mult), even parity circuit (par), sorter (sort), half adder (h.adder) and full adder (f.adder). As the results show, the success rate is better in most cases for a fixed initial state of the CA. Similarly, the average number of generations needed to evolve a working solution is lower if the initial state is fixed, i.e. it leads to a speed-up of evolution.

	Parameters			**Succ. rate**		**Avg. #gener.**		**Min. rules used**	
Circuit	#inputs	#steps	#states	**evolved**	**fixed**	**evolved**	**fixed**	**evolved**	**fixed**
med	5	6	2	97.8	*100*	120453	*70920*	5	5
mult	4	3	3	84.2	*88.8*	159408	*135028*	7	7
par	9	4	3	100	100	409	*309*	6	8
sort	4	3	2	85	*98.2*	105081	*60535*	6	6
sort	5	5	4	5.2	*9.6*	*274141*	375871	10	10
h.adder	4	3	4	*88.8*	86	85363	*82364*	7	8
h.adder	5	4	4	52	*59.2*	256543	*224348*	11	14
f.adder	5	4	4	34	*45.2*	291329	*264390*	10	12

Table 2. Comparison of the success rate and the number of generations needed to develop working single-function circuits by means of an evolved CA for evolving and fixed initial state of the CA. Values marked in italic represent the better results with respect to the given manner of specifying the initial state of the CA.

4.2 Development of two-function polymorphic circuits

The second sort of experiments have been devoted to the development of two-function polymorphic circuits. Although only a few polymorphic gates have still been realized, we consider an arbitrary two-function polymorphic gate to be a building block of the target polymorphic circuits. Therefore, every arbitrary pair of gates from Table 1 may constitute a polymorphic gate and the pairs consisting of two identical gates represent conventional gates in the poltmorphic circuits. The function of polymorphic gates is determined by a control bit which is common to all polymorphic gates in the circuit. Figure 3 illustrates the function of a sample polymorphic gate by means of the control bit.

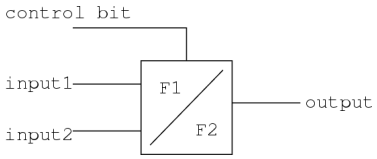


Fig. 3. The concept of polymorphic gate utilized in the experiments. If control bit is in logic 0, the gate performs function F1, otherwise it performs function F2.

In general, the proposed developmental approach based on the uniform 1D cellular automaton showed the ability to develop polymorphic circuits of different classes. Similarly to the development of single-function combinational circuits, the impact of the CA initial state on the success rate and the number of generations of the genetic algorithm needed to evolve a solution was investigated. Table 3 summarizes the statistics of the evolutionary development of polymorphic circuits. In each sort of experiments (i.e. the design of a given circuit with a given approach to the initial state), 1000 independent experiments were conducted. The maximal number of generations for the evolution was set to 1.5 million. Although the success rate of the design of circuits possessing more outputs is very low in comparison with the evolution of single-function circuits, it is not surprising since two different functions are required in a single circuit topology. The statistics show that in most cases the success rate is higher if the CA initial state is fixed during the evolution. Moreover, the average number of generations needed to evolve a working solution is lower in this case.

	Parameters			**Succ. rate**		**Avg. #gener.**		**Min. rules used**	
Circuit	#inputs	#steps	#states	**evolved**	**fixed**	**evolved**	**fixed**	**evolved**	**fixed**
med/par	5	5	4	57,1	*68,3*	143046	*111645*	5	5
med/par	7	7	5	41,9	*47,1*	391925	*362545*	20	19
sort/med	5	6	5	22,1	*22,6*	*561783*	568536	19	23
sort/mult	6	6	4	1,3	*1,4*	785710	*687403*	19	16
sort/mult	7	7	4	*0,9*	0,8	*608839*	905209	15	16
sort/parity	5	5	5	14,5	*16,2*	482018	*462758*	20	20
sort/parity	6	6	6	8,2	*9*	672306	*598862*	33	28
sort/h.adder	4	4	3	4,6	*4,6*	515076	*441353*	8	7
sort/h.adder	4	4	4	44,1	*47,5*	413029	*353368*	11	12
sort/f.adder	5	7	5	3,6	*5,6*	935358	*867781*	28	27
h.adder/mult	4	5	4	3,6	*4,2*	682657	*643104*	16	16
f.adder/med	5	5	5	*19,3*	18,3	*388150*	509391	18	19

Table 3. Comparison of the success rate and the number of generations needed to develop working polymorphic circuit by means of an evolved CA for evolving and fixed initial state of the CA. Values marked in italic represent the better results with respect to the given manner of specifying the initial state of the CA.

5 Conclusions

We presented an evolutionary developmental method based on the concept of cellular automata for the design of combinational circuits. It was demonstrated that this approach is capable to design both single-function and polymorphic combinational circuits at the gate level. The proposed method was tested on a wide range of circuits, e.g. multipliers, adders, sorters, median and parity circuits and their combinations in case of development of two-function polymorphic circuits. The experiments showed that it is more suitable in most cases to fix the initial state of the CA at the beginning of evolution in order to increase the success rate and decrease the number of generations needed to evolve a working solution.

The results obtained herein and in [18] show that cellular automata provides capabilities for generating gate-level digital circuits. In addition, it is possible to utilize other useful features of the developmental system, e.g. adaptation to different initial states. Although the design of polymorphic circuits is very difficult in general and this approach is not scalable (i.e. no working solution was developed for increasing number of inputs of the target circuit), the results indicate that this area is worth of future research. Therefore, the current results will be analyzed and the knowledges obtained herein will be applied in order to determine their impact on the possibility of designing more complicated structures (e.g. for more number of inputs, at a higher level of abstraction etc.). Moreover, the possibility of increasign the $l-back$ parameter and decreasing the restrictions of the development process will be investigated.

Acknowledgement

This work was partially supported by the Grant Agency of the Czech Republic under contract No. 102/06/0599 *Methods of Polymorphic Digital Circuit Design*, No. 102/05/H050 *Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems* and the Research Plan No. MSM 0021630528 *Security-Oriented Research in Information Technology.*

References

1. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Proc. of the 3rd European Conference on Genetic Porgramming, Lecture Notes in Computer Science, vol 1802, Berlin Heidelberg New York, Springer (2000) 121–132
2. Miller, J.F., Thomson, P.: A developmental method for growing graphs and circuits. In: Proc. of the 5th Conf. on Evolvable Systems: From Biology to Hardware (ICES 2003), Lecture Notes in Computer Science, vol. 2606, Berlin DE, Springer-Verlag (2003) 93–104
3. Tufte, G., Haddow, P.C.: Towards development on a silicon-based cellular computing machine. Natural Computing **4**(4) (2005) 387–416

4. Gordon, T.G.W., Bentley, P.J.: Towards development in evolvable hardware. In: Proc. of the 2002 NASA/DoD Conference on Evolvable Hardware, Washington D.C., US, IEEE Press (2002) 241–250
5. Stoica, A., Zebulum, R.S., Keymeulen, D.: Polymorphic electronics. In: Proc. of International Conference on Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science, volume 2210, Springer-Verlag (2001) 291–302
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 AU (2002)
7. Thompson, A., Layzell, P., Zebulum, R.: Explorations in design space: Unconventional electronics design through artificial evolution. **3**(3) (1999) 167–196
8. Sekanina, L.: Evolutionary design of gate-level polymorphic digital circuits. In: 2nd European Workshop on Evolutionary Computation in Hardware Optimisation (EvoHOT 2005), Lecture Notes in Computer Science vol. 3449, Springer (2005)
9. Bidlo, M., Sekanina, L.: Providing information from the environment for growing electronic circuits through polymorphic gates. In: Proc. of Genetic and Evolutionary Computation Conference – Workshops 2005, Association for Computing Machinery (2005) 242–248
10. S. Kumar (ed.), P. J. Bentley (ed.): On Growth, Form and Computers. Elsevier Academic Press (2003)
11. von Neumann, J.: The Theory of Self-Reproducing Automata. A. W. Burks (ed.), University of Illinois Press (1966)
12. Wolfram, S.: A New Kind of Science. Wolfram Media, Champaign IL (2002)
13. Sipper, M.: Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, volume 1194. Springer-Verlag, Berlin (1997)
14. Dellaert, F., Beer, R.: A developmental model for the evolution of complete autonomous agents. In: Proc. of the 4th International Conference on Simulation of Adaptive Behavior, Cambridge, MA, MIT Press-Bradford Books (1996) 393–401
15. Corno, F., Reorda, M.S., Squillero, G.: Evolving cellular automata for self-testing hardware. In: Proc. of the International Conference on Evolvable Systems: From Biology to Hardware, ICES 2000, Lecture Notes in Computer Science, volume 1801, Springer (2000) 31–39
16. Miller, J.F.: Evolving developmental programs for adaptation, morphogenesis and self-repair. In: Advances in Artificial Life. 7th European Conference on Artificial Life, Lecture Notes in Artificial Intelligence, volume 2801, Dortmund DE, Springer (2003) 256–265
17. Haddow, P.C., Tufte, G.: Bridging the genotype–phenotype mapping for digital fpgas. In: Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware, Los Alamitos, CA, US, IEEE Computer Society (2001) 109–115
18. Bidlo, M., Vasicek, Z.: Gate-level evolutionary development using cellular automata. In: Proc. of The 3rd NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2008, IEEE Computer Society (2008)