Comparison of the Uniform and Non-Uniform Cellular Automata-Based Approach to the Development of Combinational Circuits

Michal Bidlo
Faculty of Information Technology
Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic
E-mail: bidlom@fit.vutbr.cz

Zdenek Vasicek
Faculty of Information Technology
Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic
E-mail: vasicek@fit.vutbr.cz

Abstract

Cellular automata-based model has been shown as a useful developmental model in the evolutionary design of digital circuits at the gate level. Uniform one-dimensional cellular automata have been successfully applied to the circuit design task so far. Moreover, the initial experiments performed during our previous research have demonstrated the possibility of applying non-uniform cellular automata to the circuits design which is the main objective of the proposed paper. We will investigate this approach considering several classes of combinational circuits, provide an analysis of the obtained results and their comparison with the results of the uniform cellular automata-based model. It will be shown that evolution is able to find (in general a different) local transition function for each cell of the automaton according to which the target circuit is developed. Two different case studies will be presented in order to demonstrate the abilities of the proposed method. The first case study deals with the development of combinational multipliers and the second one is intended to develop combinational dividers. The obtained experimental results will be compared to our previous approach in which uniform cellular automata were applied. The proposed non-uniform approach enables to design circuits that we were not able to develop successfully using the uniform cellular automata.

1 Introduction

The biologically inspired development represent a powerful technique in the area of evolutionary design. A non-trivial indirect construction process (i.e. the genotype–phenotype mapping in the evolutionary algorithm) represents the main feature of this approach. Recently, the development has successfully been applied in solving various complex problems not only in engineering areas.

Cellular automata (CA) represent one of the methods to perform the computational development (i.e. to simulate the basic principles of that biological phenomena by means of digital computational devices). Since the invention of the basic concept of the cellular automata in 1966 [15], this mathematical model has been successfully applied to investigate many complex problems in different areas. The detailed survey of the principles and analysis of various types of cellular automata and their applications (including emulation of circuits and computer systems) is summarized in [16]. Sipper [13] investigated the computational properties of cellular automata and proposed an original evolutionary design method for cellular automata called the 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 [6]. 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 [5], this method is able to overcome the fault coverage that can be achieved using current engineering practice. Nandi et al. studied the theory and applications of cellular automata for synthesis of easily testable combinational logic [11]. 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) [10]. Tufte and Haddow utilized a FPGA-based platform of Sblocks [8] for the online evolution of digital circuits. The system actually implements a cellular automaton whose development determines the functions and interconnection 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 [14]. Recently, cellular automata have successfully been applied in the area of gate-level evolutionary development of digital circuits. One-dimensional uniform CA have been investigated in [2, 1, 3] and the initial experiments of applying the non-uniform approach were introduced in [4].

In this paper, adavanced experiments of the gate-level combinational circuits development using the non-uniform cellular automata will be presented and an analysis of the obtained results will be proposed. The development will be performed by means of one-dimensional cellular automata. The impact of the cellular automata size (i.e. the number of cells) on the success rate and computational effort of the evolutionary processes will be investigated. The obtained results will be compared to the results of our previous research published in [3] and the advantages and disadvantages of the proposed approach will be discussed. Moreover, it will be demonstrated that the non-uniform CA-based approach allows us to develop circuits whose development have not been successful by means of uniform cellular automata. The case studies deal with the development of combinational multipliers and dividers at the gate level.

The paper is structured as follows. Section 2 summarizes the basic principles of the biological development and highlights the aspect which represent the crucial features of the computational development. Section 3 summarizes the basic principles of the cellular automaton that will be utilized in the experiments. In Section 4 the cellular automata-based developmental model is described by means of which the target circuits will be generated. Section 5 provides the information about the experimental setup (the evolutionary algorithm and the configuration of the developmental system) with respect to the process of circuits construction. Section 6 presents the obtained results and discussion. Finally, concluding remarks are summarized in Section 7.

2 Biological and computational 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 [12].

The utilization of the computational development is motivated by the fact that natural development is one of the phenomena which is primarily responsible for the extraordinary diversity and sophistication of living creatures. It is assumed that the computational development (inspired by natural development) in connection with an evolutionary algorithm might be utilized to achieve the evolution of complex artificial objects and other objectives desired by evolutionary design systems, including evolvability, adaptation, regulation, repetition or robustness (as discussed in [9]). Several researchers have dealt with the development applied to the field of digital circuits, e.g. [7]. In fact, Tufte's work represents one of the few cellular automata-based models applied to the design of digital circuits. However, the approach presented in [14] involves higher-level functions by means of which the target circuits are implemented.

3 Cellular automata

The fundamental principle of CAs is as follows [15]. A cellular automaton (CA) consists of a regular structure of cells, each of which can possess one state from a finite set of states at a given time. The states are updated synchronously in parallel according to a local transition function. The next state of a given cell depends on the combination of states in the cellular neighborhood. In this paper the cellular neighborhood of each cell consists of the cell itself and its two immediate neighbors. Cyclic boundary conditions of the CA will be considered, i.e. the left neighbor of the first cell is the last cell of the CA and the right neighbor of the last cell is the first cell of the CA. The local transition function defines a next state of a cell 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 the CA consists of. In this paper, however, the non-uniform CA will be considered. Therefore, each cell possesses its own local transition function for updating the cell state. In general, the local transition function may be different for the cells.

4 Developmental system using cellular automata

The CA-based developmental system is based on the approach introduced in [3] with the difference that non-uniform cellular automata are considered herein.

A logic gate is assigned to each rule of the local transition function of each cell. The gate will be generated by the cells during the development of the cellular automaton. Therefore, in general, the rule of the local transition function possesses 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 to be generated and the indices of its two inputs (i_1 and i_2). The developmental step is considered as the calculation of the next state for each cell of the CA together with generating a gate by the cells. The gate to be generated is specified by the rule that is applied to determine the next state of the given cell depending on the combination of states in the cellular neighborhood. Considering this developmental scheme, one level of the circuit is generated in one developmental step of the CA. In case of the first developmental step, the inputs of the gates being generated are connected to the primary inputs of the target circuit. The inputs of gates generated in the next developmental steps are connected to the specified outputs of the gates generated in the previous developmental step. The outputs of the appropriate gates generated in the last developmental step represent the primary outputs of the target circuit. No other interconnection of the gates are allowed. This development strategy was chosen with respect to our previous experiments using the uniform CA [3, 1, 2] in order to perform a comparison of the uniform and non-uniform CA-based approach.

There are several possibilities how to connect the circuit outputs to the outputs of the gates generated in the laste developmental step if the number of cells of the CA is greather than the number of outputs of the circuit. The experiments showed that there is no significant influence of different ways of inputs connection on the design process. Therefore, we will consider the simplest way when the primary outputs of the circuit are successively connected to the gates generated by the cells from the cell index 0 to $p-1$, where p denotes the number of outputs of the circuit. The connection scheme is illustrated in Figure 1.

In order to ensure correct connection of gates generated in the first developmental step, the modulo operation will be utilized. For example, if a 3-input circuit is generated by means of a 5-cell CA and a gate generated in the first developmental step should have its input connected to index 4, it will be connected to the primary input of index $4\ mod\ 3 = 1$. (i.e. the range of cell indices $0 \ldots 4$ is mapped into the range of input indices $0 \ldots 2$.) The number of gates generated in each of the next developmental steps corresponds to the number of cells, therefore, the inputs of

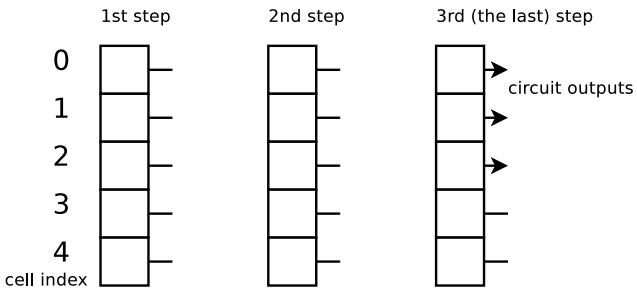


Figure 1. An example of an abstract 3-output circuit generated in 3 developmental steps. The connection scheme of the outputs utilized during the experiments is illustrated (the circuit outputs are denoted by arrows, the remaining outputs are meaningless).

the gates generated in a developmental step can be directly connected to the given outputs of the gates generated in the previous developmental step (using the index range $0 \ldots 4$ in this example).

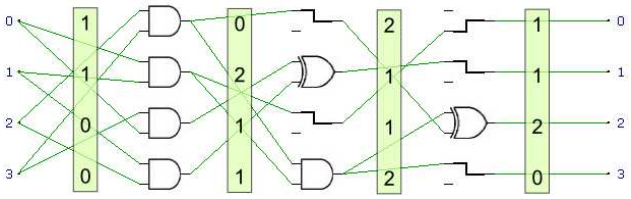
Table 1 shows the set of gates utilized for the experiments presented in this paper.

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 $NAND$ gate
4: NOR	a, b	two-input NOR gate
5: $NXOR$	a, b	two-input gate of equivalence function
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.

Figure 2a shows an example of the cellular automaton generating a three-level 2x2-bit combinational multiplier. Note that for simplicity the developmental process is explained using an uniform CA whose number of cells corresponds to the number of inputs of the target circuit. The primary inputs of the multiplier and the cells of the CA are denoted by indices 0, 1, 2 and 3. However, in real experiments presented herein, non-uniform cellular automata are evolved that may possess different cells in comparison with the number of circuit inputs and outputs. The design process 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

transition function (Fig. 2b). 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 $2\ mod\ 4 = 2$ and $3\ mod\ 4 = 3$ (the modulo operation is performed because, in general, the number of inputs may be smaller than the number of cells of the CA). The next state of the second cell in 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 $0\ mod\ 4 = 0$ and $1\ mod\ 4 = 1$. 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.



(a)

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

(b)

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

5 Evolutionary System Setup

The simple genetic algorithm was utilized for the evolutionary design of a non-uniform cellular automaton that generates a specified circuit. The objective is to design using the genetic algorithm (1) the local transition function for each cell together with the initial state of the CA and (2) the local transition function for each cell whereas the initial state is fixed to a suitable value at the beginning of the evolutionary process.

The form of the chromosome is shown in Figure 3. Each cell of the CA possesses a complete local transition function in the genome. The local transition function of a cell is represented by its rules that are encoded by 4-tuples. These rules contain the next state of the cell, the gate (function) and the indices of inputs of that gate that is generated when the rule is activated, i.e. when the cell determines its next state according to the given rule. The index of a rule inside a given local transition function is determined implicitly by means of the value calculated from the combination of states in the cellular neighborhood. This combination is interpreted as a number whose base corresponds to the number of possible cell states of the CA. Therefore, if the general rule of the local transition function is considered in the form $c_1\ c_2\ c_3 \rightarrow c_n : f\ i_1\ i_2$, only the part on the right of the arrow is encoded in the genome. For example, if a cellular automaton with 2 possible cell states and the cellular neighborhood consisting of 3 cells is considered, there are $2^3 = 8$ rules of the local transition function. Consider the rule $0\ 1\ 1 \rightarrow 0 : XOR\ 0\ 1$. Since the combination of states $0\ 1\ 1$ corresponds to the binary representation of number 3, this rule will be placed in the third 4-tuple (counted from zero) of the local transition function corresponding to the given cell. Note that this rule would be encoded as a sequence of integers $0\ 2\ 0\ 1$ (0 denotes the next state and 2, 0, 1 denotes the function of the gate and indices of its two inputs respectively).

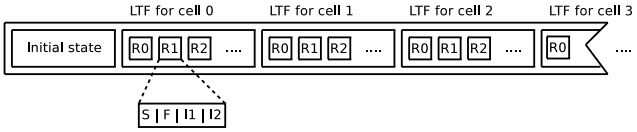


Figure 3. A genome consists of the initial states of the CA and the set of rules (R0, R1,...) of the local transition function (LTF) for each cell. Each rule contains the next state (S), gate function (F) and indices of the gate inputs (I1, I2) respectively. The combinations of states in the cellular neighborhood are encoded implicitly by the indices of rules in the chromosome.

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. No crossover is applied because the experimentation with various types of this operator has not shown any positive impact on the evolutionary process. The following mutation operator is utilized. In each chromosome selected by the tournament operator, 6 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. The high mutation rate was chosen in order to enable a larger change in the genome because no crossover operator is applied. The experiments showed that if only one gene per chromosome is mutated, then the convergence of the evolution is very slow. Therefore, up to 6 genes per chromosome may be mutated. This number represents a sufficiently large part of the genome undergoing changes in order that the evolution converges in a reasonable time while preserving a good success rate in different sorts of experiments.

The fitness function is calculated as the number of correct output bits of the target circuit using all the binary 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 $4 \cdot 2^4 = 64$. The number of developmental steps for developing a working circuit (i.e. the number of steps after which the resulting circuit is evaluated) is determined on the basis of the delay of conventional multipliers designed at the gate level.

The experiments were performed on common PCs running RedHat-based Linux operating system. The hardware configuration consists of a 2.0 GHz processor and 512 MB RAM. Sun Grid Engine (SGE) system was utilized so that several independent experiments could be run on different PCs in parallel.

6 Experimental results and discussion

Since the non-uniform cellular automata, in general, may apply different local transition function for each cell, a more complex behavior is possible to observe in comparison with the uniform CA, considering the same number of cell states [13]. However, because of several local transition functions are evolved in case of non-uniform CA, the search space grows exponentially with the number of states and, moreover, with the number of cells (size) of the CA because each additional cell extends the genome by all the rules of a local transition function. For example, the search space of a binary CA consisting of four cells for the development of a 4-input circuit composed of 7 different building blocks approximately possesses $1.2 \cdot 10^{73}$ candidate solutions. Several sets of experiments were conducted. These experiments were focused on the evolutionary design of CAs of different sizes for the development of the multipliers and dividers.

Table 2. Success rate and computational effort of the evolutionary dividers development using non-uniform cellular automata. A fixed and evolved initial state of the CA is considered.

				Evolved init. st.		Fixed init. st.	
I	W	S	T	S. rate	C. effort	S. rate	C. effort
5	10	3	4	35	16k	28	16k
			5	72	16k	70	11k
		4	4	27	17k	20	21k
			5	74	17k	65	19k
	15	3	4	51	15k	51	15k
			5	83	10k	89	7k
		4	4	36	20k	32	22k
			5	81	13k	81	13k
	20	3	4	53	18k	55	14k
			5	96	8k	96	8k
		4	4	36	22k	37	19k
			5	81	14k	86	15k
6	12	3	5	16	67k	16	75k
			6	33	70k	35	62k
		4	5	6	100k	1	133k
			6	43	79k	37	78k
	18	3	5	33	60k	27	54k
			6	67	50k	61	45k
		4	5	16	84k	17	77k
			6	60	69k	52	72k
	24	3	5	35	67k	31	59k
			6	85	41k	70	42k
		4	5	19	77k	22	73k
			6	77	55k	68	48k

In each class of circuits, two different sets of experiments will be presented. In the first one, the local transition functions of non-uniform cellular automata are evolved together with the initial states of the CAs. In the second set of experiments, only the local transition functions are evolved whereas the initial state is set to a fixed combination that is invariable during the evolutionary process. In this case the genetic algorithm actually searches the rules of the CA for the development from the given initial state. Since the nun-uniform cellular automata are considered, it is supposed that a working circuit can be developed from the simplest form of the initial state in which all the cells are set to identical initial state. The experiments were performed for various parameters of the CA (i.e. the number of cells, the number of states and the number of developmental steps) using the gates from Table 1.

For each experimental setup (i.e. the number of circuit inputs, cells, states, developmental steps and circuit type) 100 independent evolutionary experiments were conducted. The results are summarized in tables 2, 3 and 4. In each table, the number of inputs of the target circuit is denoted as I, W denotes the number of cells (width) of the CA, S

denotes the number of possible cell states and T denotes the number of developmental steps. In each experimental setup specified by those parameters we measured the success rate (the number of successfully evolved solutions out of 100 independent experiments) and computational effort (expressed as an average number of generations needed to evolve a working solution).

Combinational dividers represent a class of circuits that has not been successfully evolved by means of uniform cellular automata so far. As we assumed, the non-uniform approach is able to develop different instances of these circuits up to 7 inputs. Note that a divider possessing N input bits considers $D = N/2 + N \ mod \ 2$ bits of a dividend and the rest $d = N - D$ bits of a divisor. The quotient is then expressed by D bits. If the divisor equals zero, the quotient is not defined (i.e. its bits are considered as don't care values during the circuit evaluation). The experimental results related to the dividers development are summarized in Table 2. The maximal number of generations for measuring the success rate was chooses as follows: 50k for 5-input dividers, 150k for 6-input dividers and 750k for 7-input dividers. Although the non-uniform approach to the development of 7-input dividers was successful (with approx. 10% success rate), only 5-input and 6-input dividers will be presented herein since no 7-input working multiplier has been evolved yet so there are no results to compare. As Table 2 shows, the success rate as well as the computational effort does not differ very markedly if compared the evolving and fixed initial states. Note that in case of experiment considering the fixed initial state, the initial state of all the cells was set to zeros and only the local transition function was evolved. In contrast to [2], where the fixed initial state provided better results in comparison with the evolving initial state using uniform CA, the experiments presented herein exhibit worse success rate if the fixed initial state is considered using the non-uniform approach. It is surprising because the fixed initial state reduces the search space. Moreover, it could be claimed that evolving initial state makes the evolution more difficult because if the initial state is mutated, the evolution has to design suitable local transition functions for the cells in order that a working circuit can be developed. This might require a substantial computational effort. However, the evolving initial state evidently does not represent a problem, since the success rate reaches over 50% in many cases. Similar behavior can also be observed in the computational effort. The values do not exhibit big differences and there is not any substantial influence of the evolving/fixed initial state. Similarly to the results presented in [3], the success rate increases with increased number of cells for given values of cell states and number of developmental steps. However, the computational effort is lower for larger numbers of cells in many cases which is surprising because of significant increase of the search space (the evolution has to design a separate local transition function for each cell).

Similar features of the evolutionary process can be observed in most cases of the multipliers development whose results are summarized in Table 3. The maximal number of generations for measuring the success rate was chosen as 50k for 4-input multipliers, 250k for 5-input and 1M5 for 6-input multipliers. Unfortunately, no larger multipliers have been developed yet using CA. As could be expected, the development of larger multipliers is more difficult in comparison with the dividers. For example, there are several cases in the design of 6-input dividers where the success rate reaches substantially over 60%. However, the best success rate in case of 6-input multipliers is 60% if the initial state was fixed and 55% in case of evolving initial state. It may be caused by different structures of these classes of circuit, by the fact that in case of division by zero the result may be arbitrary and also by the differences in setting the parameters of the CA because determining optimal values for a given problem is not trivial (no systematic approach has been published yet so it is needed to perform it experimentally or empirically).

Since the combinational multipliers have already been evolved in [3] using uniform cellular automata, we will compare the results with those obtained herein by means of the non-uniform approach. Note that only the cellular automata with evolving initial states will be included. The comparison, that is shown in Table 4, provides several interesting features. It can be observed that the success rate increases with the increasing number of cells in case of the uniform CA [3], while this phenomen is not present if non-uniform CA were utilized. This feature is possible to make clear as follows. In case of uniform CA the local transition function is identical for all the cells. If a more complex circuit structure ought to be developed, it is easier for the design process to involve more cells in order to generate more gates in a developmental step from which the suitable interconnection of the target circuit emerges during other developmental steps (i.e. some gates generated by the CA may be unused). Therefore, there are more possibilities how to develop a working circuit if the number of cells is higher which leads to increased success rate. The non-uniform CA is able to perform more complex behavior in comparison with the uniform approach that enables the CA to develop sufficient number of working circuit structures which leads to increased success rate already with less number of cells. The next interesting feature is that in more than 50% of experiments the computational effort of the evolution of non-uniform CA is substantially lower in comparison with the evolution of uniform CA. Moreover, the non-uniform approach was able to find working solutions of 3x3-bit multipliers using the same parameters of the CA that were unsuccessful by means of the uniform approach.

Table 3. Success rate and computational effort of the evolutionary multipliers development using non-uniform cellular automata. A fixed and evolved initial state of the CA is considered.

				Evolved init. st.		Fixed init. st.	
I	W	S	T	S. rate	C. effort	S. rate	C. effort
4	7	2	2	97	3k	94	3k
			3	100	1k	100	1k
		3	2	80	1k	76	1k
			3	94	2k	93	2k
		4	2	74	4k	63	3k
			3	95	3k	92	4k
	10	2	2	93	2k	87	2k
			3	100	1k	100	777
		3	2	84	3k	74	1k
			3	99	1k	100	2k
		4	2	83	6k	78	3k
			3	94	2k	92	4k
	13	2	2	93	6k	81	5k
			3	100	468	100	438
		3	2	81	4k	75	2k
			3	100	1k	100	2k
		4	2	78	4k	74	5k
			3	96	3k	97	3k
5	9	3	3	53	54k	50	52k
			4	81	41k	86	37k
		4	3	39	77k	39	79k
			4	82	61k	82	54k
		5	3	41	107k	35	95k
			4	59	65k	65	68k
	13	3	3	49	40k	57	57k
			4	97	27k	94	20k
		4	3	44	88k	50	80k
			4	95	44k	92	35k
		5	3	44	85k	37	92k
			4	76	52k	83	59k
	17	3	3	51	58k	63	70k
			4	96	23k	98	25k
		4	3	54	78k	56	77k
			4	95	39k	95	39k
		5	3	44	84k	40	90k
			4	91	66k	92	63k
6	11	4	5	0	-	6	822k
			6	2	382k	3	376k
		5	5	0	-	0	-
			6	0	-	1	821k
		6	5	1	900k	1	1M
			6	1	1M	1	1M
	16	4	5	16	554k	9	624k
			6	27	589k	32	673k
		5	5	14	854k	7	1M
			6	20	758k	26	701k
		6	5	5	887k	4	697k
			6	10	837k	9	675k
	21	4	5	33	617k	27	767k
			6	55	499k	60	577k
		5	5	25	845k	21	919k
			6	40	617k	41	685k
		6	5	4	688k	10	836k
			6	19	639k	18	598k

Table 4. Comparison of success rate and computational effort of the evolutionary multipliers development using non-uniform and uniform CA

				Non-uniform CA		Uniform CA	
I	W	S	T	S. rate	C. effort	S. rate	C. effort
4	7	2	2	97	3k	44	18k
			3	100	1k	3	17k
		3	2	80	1k	100	4k
			3	94	2k	92	10k
		4	2	74	4k	99	2k
			3	95	3k	99	1k
	10	2	2	93	2k	77	16k
			3	100	1k	16	27k
		3	2	84	3k	100	3k
			3	99	1k	99	7k
		4	2	83	6k	100	752
			3	94	2k	99	1k
	13	2	2	93	6k	85	13k
			3	100	468	23	28k
		3	2	81	4k	100	2k
			3	100	1k	98	5k
		4	2	78	4k	100	550
			3	96	3k	100	2k
5	9	3	3	53	54k	8	157k
			4	81	41k	5	145k
		4	3	39	77k	40	105k
			4	82	61k	54	102k
		5	3	41	107k	63	77k
			4	59	65k	90	45k
	13	3	3	49	40k	16	167k
			4	97	27k	15	146k
		4	3	44	88k	69	90k
			4	95	44k	87	73k
		5	3	44	85k	92	73k
			4	76	52k	96	41k
	17	3	3	51	58k	24	161k
			4	96	23k	19	166k
		4	3	54	78k	68	103k
			4	95	39k	91	82k
		5	3	44	84k	94	83k
			4	91	66k	99	34k
6	11	4	5	0	-	0	-
			6	2	382k	0	-
		5	5	0	-	1	1M
			6	0	-	0	-
		6	5	1	900k	9	666k
			6	1	1M	9	632k
	16	4	5	16	554k	0	-
			6	27	589k	0	-
		5	5	14	854k	18	887k
			6	20	758k	18	842k
		6	5	5	887k	32	890k
			6	10	837k	56	792k
	21	4	5	33	617k	0	-
			6	55	499k	0	-
		5	5	25	845k	22	938k
			6	40	617k	27	975k
		6	5	4	688k	42	658k
			6	19	639k	62	797k

It means that, although the non-uniform search space is substantially larger if compared to the uniform CA, the number of correct solutions and the conditions for the evolutionary convergence is probably more advantageous than in case of uniform CA. The low computational effort of the non-uniform setups in comparison with the corresponding uniform setups may also indicate suitable setup values of the cellular automata for the given circuit to be developed.

7 Conclusions

We proposed a developmental method based on non-uniform cellular automata for generating combinational circuits at the gate level. The evolutionary design of the CA was demonstrated on the problems of dividers and multipliers development which could be considered as typical benchmark problems. The impact of the cellular automata size on the success rate and computational effort was investigated. The results showed that the number of cells of the non-uniform CA has lower influence on the success rate than the uniform CA. The non-uniform approach enabled us to evolve circuits that has not been successfully designed using the uniform CA. It is the case of the combinational dividers of different sizes (we performed successful development of 5-input, 6-input and 7-input dividers) and 6-input multipliers using several CA setups that were unsuccessful in the development using uniform CA. We also provided a comparative study of the multipliers development using non-uniform and uniform approach. The results showed that the computational effort of the non-uniform CA evolution is substantially lower in most cases in comparison with the uniform approach.

The next research will be focused on a more in-depth analysis of the evolved developmental rules of the CA, optimization of the developing circuits during the evolutionary process and investigation of other development strategies considering different sets of building blocks.

8 Acknowledgments

This work was partially supported by the Grant Agency of the Czech Republic under contract No. GA102/07/0850 Design and hardware implementation of a patent-invention machine, No. GD102/09/H042 Mathematical and Engineering Approaches to Developing Reliable and Secure Concurrent and Distributed Computer Systems and the Research Plan No. MSM 0021630528 Security-Oriented Research in Information Technology.

References

[1] M. Bidlo and Z. Vasicek. Cellular automata-based development of combinational and polymorphic circuits: A comparative study. In *Proc. of the 8th International Conference on Evolvable Systems: From Biology to Hardware (ICES 2008), Lecture Notes in Computer Science, vol. 5216*, pages 106–117, Berlin Heidelberg New York, 2008. Springer.

[2] M. Bidlo and Z. Vasicek. Gate-level evolutionary development using cellular automata. In *Proc. of The 3nd NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2008*, pages 11–18. IEEE Computer Society, 2008.

[3] M. Bidlo and Z. Vasicek. Investigating gate-level evolutionary development of combinational multipliers using enhanced cellular automata-based model. In *Proc. of The 2009 IEEE Congress on Evolutionary Computatio, CEC 2009*. IEEE Computer Society, 2009.

[4] M. Bidlo and Z. Vasicek. Development of combinational circuits using non-uniform cellular automata-based approach: Initial results. In *Proc. of The Genetic and Evolutionary Computation Conference, GECCO 2009*. ACM Press, 2009 (to appear).

[5] F. Corno, M. S. Reorda, and G. Squillero. 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*, pages 31–39. Springer, 2000.

[6] F. Dellaert and R. Beer. A developmental model for the evolution of complete autonomous agents. In *Proc. of the 4th International Conference on Simulation of Adaptive Behavior*, pages 393–401, Cambridge, MA, 1996. MIT Press-Bradford Books.

[7] T. G. W. Gordon. Exploiting development to enhance the scalability of hardware evolution, PhD thesis. Department of Computer Science, University College London, 2005.

[8] P. C. Haddow and G. Tufte. Bridging the genotype–phenotype mapping for digital FPGAs. In *Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware*, pages 109–115, Los Alamitos, CA, US, 2001. IEEE Computer Society.

[9] S. Kumar. Investigating computational models of development for the construction of shape and form, PhD thesis. Department of Computer Science, University College London, 2004.

[10] J. F. Miller. 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*, pages 256–265, Dortmund DE, 2003. Springer.

[11] S. Nandi and P. P. Chaudhuri. Theory and applications of cellular automata for synthesis of easily testable combinational logic. In *Proceedings of the 10th Anniversary Compendium of Papers from Asian Test Symposium 1992-2001*, page 146. IEEE Computer Society, 1995.

[12] S. Kumar and P. J. Bentley (eds.). *On Growth, Form and Computers*. Elsevier Academic Press, 2003.

[13] M. Sipper. *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, volume 1194*. Springer-Verlag, Berlin, 1997.

[14] G. Tufte and P. C. Haddow. Towards development on a silicon-based cellular computing machine. *Natural Computing*, 4(4):387–416, 2005.

[15] J. von Neumann. *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[16] S. Wolfram. *A New Kind of Science*. Wolfram Media, Champaign IL, 2002.