Gate-Level Evolutionary Development Using Cellular Automata

Michal Bidlo and Zdeněk Vašíček
Brno University of Technology
Faculty of Information Technology
Božetěchova 2, 61266 Brno, Czech republic
bidlom@fit.vutbr.cz, vasicek@fit.vutbr.cz

Abstract

In this paper we present a novel evolutionary developmental technique for the design of the combinational circuits. This technique is based on the development one-dimensional uniform cellular automaton. The goal is to evolve a cellular automaton – its local transition function and two different initial states from which a combinational circuit with a given functionality at the gate-level may be developed. The two evolved initial states are intended to demonstrate the ability of the developmental process to construct the given circuit by means of a single local transition function. Moreover, it will be shown that the developmental process is able to adapt also to other initial states than that were originally evolved, i.e. a working circuit possessing a different structure is created. The circuit functionality may be preserved even if the development of the cellular automaton continues after the original circuit was developed.

1. Introduction

In nature, evolution has created various systems which exhibit emergent behavior, e.g. ant colonies, immune systems or cellular assemblies. The term emergent behavior can be understood as the appearance of global capability of the system that is not explicitly represented in the system's elementary components or in their interconnections [2]. Cellular automaton, inspired by the biological development of multicellular organisms, represents a model capable of exhibiting the emergent behavior.

Since the invention of the basic concept of the cellular automata in 1966 [14], this mathematical model has been successfuly 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 [15]. Sipper [11] investigated the computational properties of cellular automata and proposed an original evolutionary design method for "programming" the 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 [4]. 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 [1], this method is able to overcome the fault coverage that can be achieved using current engineering practice. De Garis and Korkin introduced the "CAM-Brain Machine", an FPGA-based piece of hardware that implements a genetic algorithm for the evolution of a cellular automaton-based neural network module consisting of aproximately 1,000 neurons [3]. Nandi et al. studied the theory and applications of cellular automata for synthesis of easily testable combinational logic [9]. 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) [8]. Tufte and Haddow utilized a FPGA-based platform of Sblocks [6] for the online evolution of digital circuits. The system actually implements a cellular automaton whose development determines the functions and interconnenition 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 [13].

In this paper we present a novel evolutionary developmental technique for the design of the combinational circuits. This technique is based on the development one-dimensional (1D) uniform cellular automaton (CA). The goal is to evolve a cellular automaton – its local transition function and two different initial states from which a combinational circuit with a given functionality at the gate-level may be developed. The two evolved initial states are intended to demonstrate the ability of the developmental process to construct the given circuit by means of a single local transition function. Moreover, it will be shown that the developmental process is able to adapt also to other initial states than that were originally evolved, i.e. a working circuit possessing a different structure is created. It is a case of the emergent behavior. The circuit functionality may be preserved even if the development of the cellular automaton continues after the original circuit was developed.

The paper is structured as follows. Section 2 summarizes the basic principless of the biological development and highlights the aspect which represent the crucial features of the computational development. In Section 3 the cellular automata-based developmental model in described by means of which combinational circuits are developed. Section 4 provides the information on the evolutionary system setup and the role of the developmental process with respect to the construction of digital circuits. Section 5 presents the obtained results and discussed their features. Finally, concluding remarks are given in Section 6.

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].

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 [7]). Several researchers have dealt with the development applied to the field of digital circuits, e.g. [12], [5]. 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 [12] and [13] involves higher-level functions by means of which the target circuits are implemented. In this paper we present a method for generating digital circuits at the gate level using one-dimensional uniform cellular automata.

3. Cellular Automata-Based Model

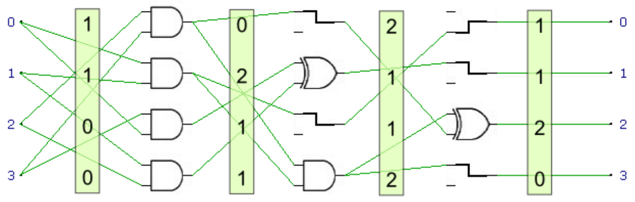
In this section, we present the developmental model based on the uniform 1D CA. Cellular automata are discrete dynamical systems consisting of a regular structure of cells, each of which may occur in one state from a finite set of states. The states are updated synchronously in parallel according to 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.

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). The gate is generated by each cell during the development of the CA. Note that the developmental step is considered as the calculation of the next state for each cell 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. In case of the first developmental step, the gates being generated connect their inputs to the primary inputs of the targer circuit. Otherwise the gate inputs are connected to the outputs of the gates generated in the previous developmental step. 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. The number of cells of the CA equals the number of primary inputs of the circuit. 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). 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: IDA	a, x	one-bit buffer (identity function) of the first input
4: 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 1 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 transition function (Fig. 1a). 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 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 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.



(a)

$$\begin{array}{ll}
0\,0\,1 \rightarrow 1 : AND\,1\,2 & 0\,1\,1 \rightarrow 0 : AND\,2\,3 \\
0\,2\,1 \rightarrow 1 : XOR\,2\,3 & 1\,0\,0 \rightarrow 1 : AND\,3\,0 \\
1\,0\,2 \rightarrow 2 : IDA\,0\,1 & 1\,1\,0 \rightarrow 2 : AND\,0\,1 \\
1\,1\,2 \rightarrow 2 : XOR\,3\,0 & 1\,2\,2 \rightarrow 0 : IDA\,3\,3 \\
2\,1\,1 \rightarrow 1 : IDA\,1\,0 & 2\,2\,1 \rightarrow 1 : IDB\,0\,2
\end{array}$$

(b)

Figure 1. 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.

4. Evolutionary System Setup

The simple genetic algorithm was utilized for the evolutionary design of the cellular automaton that generates a specified circuit. Since one of the goals is to demonstrate adaptation of the developmental process to more than one initial states of the CA, two initial states are evolved together with a single local transition function. The form of the chromosome is shown in Fig. 2. The rules of the transition function are represented by a 4-tuples, each of which contains the next state of the cell, the function and the indices of inputs of the gate to be generated when the rule is activated. 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. Note that the rule is encoded as a sequence of integers 0 1 0 1.

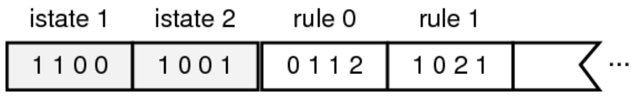


Figure 2. A chromosome consists of two initial states of the CA to be evolved (istate 1, istate 2) and the set of rules of the local transition function. Each rule contains the next state, gate function and indices of two inputs of the gate respectively. The combinations of states in the cellular neighborhood 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. 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. 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 5 genes per chromosome may be mutated. This number represents a sufficiently large part of genome undergoing changes in order the evolution converges in a reasonable time while preserving a good success rate in different sorts of experiments. 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. Two instances of the circuit are developed for two different initial states of the CA. 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 $2 \cdot 4 \cdot 2^4 = 128$. The experiments showed that it is difficult to determine exactly the number of developmental steps after which a working circuit is developed. Therefore, the fitness is computed for each developmental step, considering a limit of the number of developmental steps that is specific for a given experiment. If a working circuit is developed after one of the developmental steps for both the evolved initial states of the CA, the evolution is finished. If no solution is evolved in a limit of the number of generations (which is specific for different sort of experiments) the evolution is restarted with the new randomly initialized population.

5. Experimental Results and Discussion

The objective of the experiments is to evolve CAs that generates combinational circuits at the gate-level, for example 2x2-bit multipliers, 3+2-bit adders, 5-input sorters, 5-input median circuits. These circuits have usually been considered as typical benchmarks in the area of evolutionary design. The experiments were conducted on a cluster consisting of 100 PCs Pentium IV, 2.4GHz, 1GB RAM using the Sun Grid Engine (SGE) service software. Therefore, it is possible to run up to 100 independent experiments in parallel. The evolution time varied from a few seconds to tens of minutes depending on the type of the circuit and its number of inputs.

Tables 2a and 2b summarize the statistics of the experiments divided according to the number of inputs of the circuits developed. A perfect solutions were discovered for 2x2-bit multiplier, 4-bit and 5-bit sorters and median circuits, 2+2-bit and 3+2-bit adder. It is evident (and also expectable) that the higher number of possible cell states the higher success rate in the specified limit of the number of generations. Since the higher number of states implies the higher number of different combinations of states in the cell neighborhood, more different gates (with respect to the connection of their inputs) may be associated with the rules of the local transition function. Therefore, there are more possibilities of correct circuit structures with the same functionality to be generated during the development of the CA. Although the higher number of states causes the growth of the CA's search space, the average number of generations needed to evolve a working circuit decreases in most cases. However, an exception may be observed in case of the 5-input median circuits. There is a very good success rate (96%) if the cellular automaton utilizes only two states. For three states of the CA, there is a large decrease of the success rate – only 38% runs of experiments finished successfully. Since the structure of the median networks is actually very similar to the sorting networks and, moreover, medians may be obtained directly from the sorting networks, it is very difficult to explain the reason of such a low success rate in comparison with the results obtained in case of the sorting networks developed using a higher number of states.

Considering the quality of the evolved solutions, the following aspects may be optimized. (1) The number of developmental steps after which the first working circuit emerges is optimized (i.e. the delay of the circuit). (2) The number of states of the cells is reduced (i.e. the computational effort of the CA). (3) The number of different rules utilized for the circuit generation is minimized (i.e. more regular circuit

(a) Summary of the evolutionary development of 4-input circuits

#states	success rate [%]	average num. of generations in a successful experiment	#applied rules min	#applied rules max	#initial states the development adapts to	#devel. steps min	#devel. steps max
2+2-bit adder, 12 steps evaluated, max. 500k generations evolved							
3	21	177.1k	7	18	3	3	10
4	66	144.6k	7	20	2	3	7
4-bit sorter, 20 steps evaluated, max. 500k generations evolved							
2	86	151.8k	6	8	12	3	21
3	99	56.2k	6	18	12	3	25
2x2-bit multiplier, 12 steps evaluated, max. 500k generations evolved							
3	8	200.0k	10	16	2	3	5
4	70	131.7k	8	19	5	3	17

(b) Summary of the evolutionary development of 5-input circuits

#states	success rate [%]	average num. of generations in a successful experiment	#applied rules min	#applied rules max	#initial states the development adapts to	#devel. steps min	#devel. steps max
3+2-bit adder, 20 steps evaluated, max. 1.5M generations evolved							
3	1	686.0k	12	14	2	4	6
4	10	567.1k	12	17	2	4	10
5-bit sorter, 20 steps evaluated, max. 1M generations evolved							
3	10	278.6k	5	17	15	5	21
4	30	423.6k	5	24	20	5	18
5-bit median, 20 steps evaluated, max. 50k generations evolved							
2	96	10.8k	5	6	10	5	20
3	38	16.0k	5	24	15	5	19

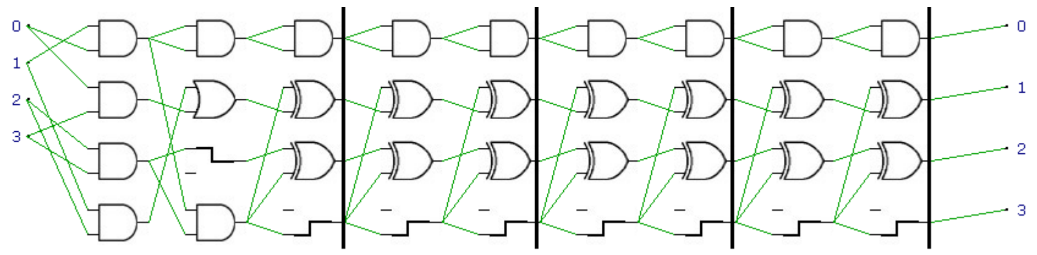
Table 2. Statistic summary of the experiments dealing with (a) 4-input circuits, (b) 5-input circuits.

structure may emerge). In fact, this aspect directly relates to (2) as more combinations of states in the cellular neighborhood exist which may be associated with different gates to be generated. (4) The period in which working circuits are developed (if the evolved CA exhibits this ability). The period is understood as the number of steps of the CA after which correct behavior of the circuit is observed during the CA development. This behavior is exhibited by some CAs if the development continues after the first correct behavior of the generated circuit was observed. The short period may lead to simpler and more regular circuit structures to be developed.

Figure 3 shows a 2x2-bit multiplier developed by means of one of the best CAs evolved in this sort of experiments. The working circuit is created after three developmental steps. The state of the CA does not change starting in the third step. Thus a highly regular structure of the circuit is generated if the development of the CA continues. Surprisingly, after each two developmental steps, the circuit outputs exhibit the required function, i.e. the working circuit is developed with period 2 (the correct functionality is denoted by thick vertical lines in Figure 3). In addition to the two different states of the CA which were evaluated by the fitness function and evolved to 2033 and 2303, the developmental process is able to adapt also to three other initial states, i.e. from five initial states in total a working circuit emerges. The initial states and other properties of the developmental process are shown in Table 3. Since states 0332, 3203 and 3320 represent rotated forms of the evolved state 2033 only, all these states might be considered as identical with respect to the cyclic boundary conditions of the CA. However, the circuit development is dependent on the linear representation of the CA as well as on the order of values in the initial state. Therefore, different circuit structures are generated as demonstrated by different properties of the developmental process and the circuits generated (see Table 3). This behavior indicates a certain degree of adaptation of the developmental process to different initial conditions.

Combinational multipliers have been usually considered to be difficult to evolve at the gate-level. In this case a successful development was demonstrated utilizing one-dimensional uniform CA exhibiting interesting properties of adaptation and repetition of a part of the circuit structure retainig the desired function of the circuit.

Similar abilities and properties of the developmental process may be observed also in the development of other classes of circuits. Figure 4 shows an example of a sorter together with the rules of the evolved CA by means of which



(a)

initial state: 2303

developmental rules:	$0\ 2\ 3 \rightarrow 2 : AND\ 0\ 0\ (8\times)$	$0\ 3\ 2 \rightarrow 0 : AND\ 1\ 2\ (1\times)$
$1\ 0\ 2 \rightarrow 0 : IDB\ 2\ 3\ (7\times)$	$2\ 3\ 0 \rightarrow 3 : AND\ 0\ 3\ (1\times)$	$2\ 3\ 1 \rightarrow 3 : XOR\ 3\ 1\ (7\times)$
$2\ 3\ 3 \rightarrow 3 : OR\ 3\ 1\ (1\times)$	$3\ 0\ 2 \rightarrow 0 : AND\ 0\ 2\ (1\times)$	$3\ 0\ 3 \rightarrow 3 : AND\ 2\ 3\ (1\times)$
$3\ 1\ 0 \rightarrow 1 : XOR\ 2\ 3\ (7\times)$	$3\ 2\ 3 \rightarrow 2 : AND\ 1\ 0\ (1\times)$	$3\ 3\ 0 \rightarrow 1 : IDA\ 2\ 3\ (1\times)$

(b)

Figure 3. (a) Example of a 2x2-bit multiplier developed by means of the evolved CA shown in part (b). The thick vertical lines denote the levels of circuit at which the correct behavior is observed. The rules shown in part (b) represent only the part of the local transition function that is applied for the development of the circuit shown in part (a). The numbers in parentheses states how often the rule was applied.

initial state	steps	period	rules
0332	9	7	15
2033 (evolved)	4	2	15
3203	9	7	15
3320	8	6	15
2303 (evolved)	3	2	11

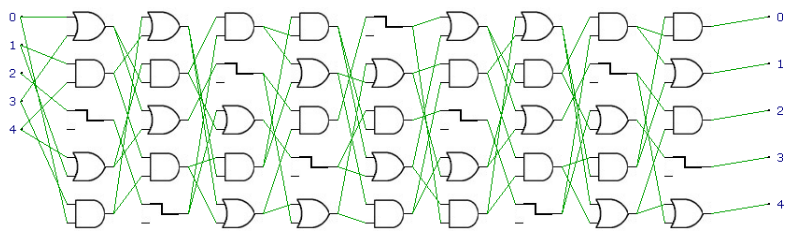
Table 3. List of initial states the CA from Fig. 3a is able to adapt to. Column steps represents the number of developmental steps after which the working circuit is created. Column period contains the lengtl of cycle of the CA at the end of which correct behavior of the circuit is observed and column rules denotes the number of rules of the local transition function utilized during the development from a given initial state.

the circuit was created. Note that the XOR gate was not utilized in the development of the sorting circuit. Table 4 shows four selected initial states the development is able to adapt to. Note, however, that the total number of adaptable initial states in this case is 20 because the development is able to adapt also to all the rotated forms of the states from Table 4. Although the delay of the developed sorter is not optimal, the development may continue with the period 5 (not shown in Figure 4). In addition, only 5 different rules of the local transition function utilized during the development of the CA.

initial state	steps	period	rules
30013 (evolved) and all its rotated forms	9	5	15
20103 and all its rotated forms	8	5	20
32010 (evolved)	10	5	20
01113 and all its rotated forms	9	5	5
30202 and all its rotated forms	8	5	10

Table 4. List of selected initial states the CA from Fig. 4a. is able to adapt to. The colums have the same meaning as in Table 3.

The next sort of experiments deals with the develpoment of adders. The initial experiments have shown that the concept of the full adder with carry is very difficult to design by means of cellular automata considering the adaptation to several initial states. However, a modified structures have been successfully developed utilizing the concept of $m+n$-bit input vector (without the carry being considered explicitly). In this case some results were evolved from which an example of a 3+2-bit adder is shown in Figure 5. Unfortunately, the ability of adaptation to other initial states of the CA is very low (only the two evolved initial states may be utilized for the development of working 3+2-bit adder). Similarly, in case of the design of 2+2-bit adder at most 3



(a)

initial state: 01113

developmental rules:

$0\ 1\ 1 \rightarrow 3 : AND\ 1\ 4\ (9\times)$

$1\ 1\ 1 \rightarrow 0 : IDA\ 2\ 2\ (9\times)$

$1\ 1\ 3 \rightarrow 1 : OR\ 4\ 1\ (9\times)$

$1\ 3\ 0 \rightarrow 1 : AND\ 3\ 0\ (9\times)$

$3\ 0\ 1 \rightarrow 1 : OR\ 0\ 3\ (9\times)$

(b)

Figure 4. (a) Example of a 5-bit sorter developed by means of the evolved CA shown in part (b). Note that only the basic circuit is shown without the levels generated during the cyclic development of the CA. The rules shown in part (b) represent only the part of the local transition function that is applied for the development of the circuit shown in part (a). The numbers in parentheses states how often the rule was applied.

initial states in total were determined to be suitable for the development of this class of circuits. The circuit shown in Figure 5 is fully developed after four steps. However, if the development continues, the correct function exhibits with period 5. It is evident that two different levels of the circuit are created during the third and fourth developmental step. This structure repeats periodically in the next steps of the CA, generating a working circuit after each period (denoted by thick vertical lines in Figure 5).

initial state	steps	period	rules
11102 (evolved)	4	2	13
11122 (evolved)	4	2	12

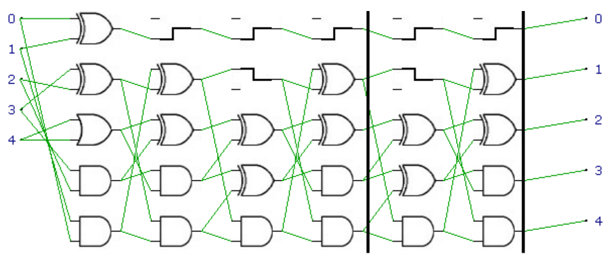
Table 5. List of initial states the CA from Fig. 5a. is able to adapt to. The colums have the same meaning as in Table 3.

6. Conclusions

A novel evolutionary developmental technique was introduced for the design of the combinational circuits by means of one-dimensional uniform cellular automata. The traditional concept of the cellular automata was modified in order to generate the circuits at the gate-level during the development of the CA. In particular, every rule of the local transition function contains the next state of a cell and a logic gate to be generated by the cell if the specific rule is applied. We showed that the genetic algorithm is able to design an CA (i.e. its initial state and the local transition function) for the development of a circuit with a given functionality.

The evolutionary experiments presented herein were focused on the adaptation of the developmental process to different initial states of the CA retaining the ability to develop the specified circuit. Therefore, two different initial states were evolved together with a single local transition function. In some cases a CA was evolved which generates working circuits also from other initial states than that were evaluated during the evolution. Since the the circuits being generated from the different initial states possess different structure, the development is actually able to adapt to different initial conditions. It is also a case of the emergent behavior because of local interactions of the cells during the development of the CA. Moreover, properties of the CAs were investigated related to the development of the circuits after different number of developmental steps. An interesting behavior was observed regarding the ability of the CA to develop combinational circuits which provide correct output at different levels (i.e. outputs of gates which were generated by different steps of the CA). In particular, a working circuit is developed after a specific number of steps of the CA. However, if the CA continues to develop, a periodic behavior of the CA is observed that may result in generating regular structure of the circuit which is fully functional after each period of the development.

The results presented in this paper demonstrated interesting abilities of 1D uniform cellular automata in the area of structural design of digital circuits. Unfortunately, the scalability of this approach is very difficult. Therefore, more research is needed in order to utilize the properties of the cellular automata-based model. The optimization of different aspects of the development and properties of the generated circuits was not included in the fitness calculation during evolution. Thus a modification may be introduced in order to investigate the specific features of the system, e.g. the regularity of the circuits, conditions on which the regularity may occur, relation with the repetition of the states during the CA development etc. Moreover, non-uniformity of the CA may be investigated in order to compare the ob-



(a)

initial state: 11102

developmental rules:	$0\ 0\ 1 \rightarrow 1: XOR\ 2\ 3\ (2\times)$
$0\ 1\ 1 \rightarrow 0: XOR\ 2\ 3\ (3\times)$	$0\ 1\ 2 \rightarrow 1: XOR\ 3\ 4\ (2\times)$
$0\ 2\ 1 \rightarrow 2: AND\ 1\ 0\ (1\times)$	$1\ 0\ 0 \rightarrow 0: IDA\ 1\ 3\ (2\times)$
$1\ 0\ 1 \rightarrow 0: XOR\ 4\ 1\ (3\times)$	$1\ 0\ 2 \rightarrow 1: AND\ 3\ 2\ (1\times)$
$1\ 1\ 0 \rightarrow 1: OR\ 4\ 4\ (1\times)$	$1\ 1\ 1 \rightarrow 0: XOR\ 3\ 2\ (1\times)$
$1\ 1\ 2 \rightarrow 1: AND\ 3\ 2\ (3\times)$	$1\ 2\ 1 \rightarrow 2: AND\ 1\ 4\ (5\times)$
$2\ 1\ 0 \rightarrow 1: IDB\ 4\ 0\ (5\times)$	$2\ 1\ 1 \rightarrow 1: XOR\ 0\ 1\ (1\times)$

(b)

Figure 5. (a) Example of a 3+2-bit adder developed by means of the evolved CA shown in part (b). The outputs 0, 1, 2 and 3 represent the primary outputs of the circuit. The thick vertical lines denote the levels of circuit at which the correct behavior is observed. The rules shown in part (b) represent only the part of the local transition function that is applied for the development of the circuit shown in part (a). The numbers in parentheses states how often the rule was applied.

tained results with the CAs of higher computational capabilities. These issues constitute potential areas for the future research.

Acknowledgement

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. 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] 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.

[2] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In *Proc. of the 3rd International Conference on Parallel Problem Solving from Nature, PPSN 1994, Lecture Notes in Computer Science, volume 866*, pages 344–353, Heidelberg, DE, 1994. Springer-Verlag.

[3] H. de Garis and M. Korkin. The cam-brain machine (cbm) an fpga based hardware tool which evolves a 1000 neuron net circuit module in seconds and updates a 75 million neuron artificial brain for real time robot control. *Neurocomputing*, 42(1–4):35–68, February 2002.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] S. Nandi and P. P. Chaudhury. Theory and application of cellular automata for synthesis of easily testable combinational logic. In *Proceedings of the 4th Asian Test Symposium*, pages 146–152. IEEE Computer Society, 1995.

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

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

[12] G. Tufte. Development of digital circuits on a virtual sblock fpga, PhD thesis. Technical report, Department of Computer and Information Science, Norwegian University of Science and Technology, 2004.

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

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

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