Investigating Gate-Level Evolutionary Development of Combinational Multipliers Using Enhanced Cellular Automata-Based Model

Michal Bidlo and Zdenek Vasicek

***Abstract*— Cellular automata represent a computational model that is based on updating the states of the cells, that are arranged in a regular structure, by means of local interactions between the cells. Cellular automata have often been utilized as a developmental model in engineering areas to solve many complex problems. In the area of the evolutionary algorithms, cellular automata can be applied as an indirect mapping between genotypes and phenotypes. In the recent years, this approach has successfully been applied on the evolutionary development of digital circuits at the gate level. Combinational multipliers represent a class of circuits that is usually considered as hard task for the design using the evolutionary techniques. In our previous research regarding the cellular automata-based development, 2x2-bit multipliers were successfully evolved using this approach. Combinational multipliers have been chosen in this paper to demonstrate capabilities of an advanced developmental system that allows to apply cellular automata of different sizes in order to design larger instances of this kind of circuits. In the experiments presented herein, the 2x3-bit and 3x3-bit multipliers will be considered which represent the first case when such instances of multipliers have been successfully developed at the gate level using cellular automata. The proposed developmental model is investigated in detail with respect to the success rate of the evolutionary experiments for different experimental setups (such as the cellular automata size, the number of cell states and developmental steps). Moreover, it will be demonstrated that different ways of connections of the circuit outputs can be utilized without a significant influence on the successfulness of the evolutionary process.**

I. Introduction

In the area of evolutionary design, the biologically inspired development has often been utilized as indirect construction process of the target objects. This has been accomplished by means of algorithms that were designed using the evolution. Cellular automata represent one of the methods to perform the so called computational development inspired by principles observed in biology (e.g. cell division, differentiation and growth). In addition to their original purpose — the study of the behavior of complex systems and simulation of those biological phenomena [1] — cellular automata represent a universal computational model applicable to various fields [2]. 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 [3].

Sipper [2] 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 [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 [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 [6]. 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) [7]. 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 [9].

In this paper we present an approach that allows to generate combinational multipliers at the gate level. We will show that by increasing the number of cells of the cellular automaton larger multipliers can be developed (note that during our previous research, only 2x2-bit multipliers were successfully designed using the cellular automata that possess the same number of cells as the number of input of the target circuit [10]). We will investigate the impact of the cellular automata size (i.e. the number of cells) on the success rate of the experiments. Finally, it will be shown that different ways of connection of the outputs of the multipliers can be considered during the evolutionary process. The impact of this issue on the success rate will be investigated.

Michal Bidlo and Zdeněk Vašíček are with the Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 61266 Brno, Czech Republic (email: bidlom@fit.vutbr.cz, vasicek@fit.vutbr.cz).

The paper is structured as follows. Section II summarizes the basic principles of the biological development and highlights the aspect which represent the crucial features of the computational development. Section III summarizes the basic principles of the cellular automaton that will be utilized in the experiments. In Section IV the cellular automata-based developmental model is described by means of which the combinational multipliers will be generated. Section V provides the information on the evolutionary system setup and the configuration of the developmental process with respect to the construction. Section VI provides the obtained results and discussion. Finally, concluding remarks are given in Section VII.

II. 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 [11].

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 [12]). Several researchers have dealt with the development applied to the field of digital circuits, e.g. [13]. 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 [9] involves higher-level functions by means of which the target circuits are implemented.

III. Cellular automata

The fundamental principle of CAs is as follows [1]. 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. Let us call a developmental step of the CA the synchronous update of all the cells of the CA. 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 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, that will be applied in this paper, the local transition function is identical for all the cells.

IV. Proposed CA-based Developmental System

A CA-based developmental model will be utilized to generate combinational multipliers at the gate level. This model is based on the approach introduced in [10].

A logic gate is assigned to each rule of the local transition function that will be generated by the cells during the CA development. 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, i_2). Since we consider 3-cell neighbourhood of each cell (i.e. the next state of a cell depends on the present state of this cell and the states of its immediate neighbours on the left and right side), the state space that determines the number of different rules of the local transition function can be calculated as $rules_count = s^3$, where s denotes the number of possible cell states. 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 inputs of the gates being generated are connected to the primary inputs of the target circuit. Otherwise the gate inputs are connected to the outputs of the gates generated in the previous developmental step, which corresponds to the $l - back$ parameter of the value 1 if the cartesian genetic programming is considered [14].

The enhancements of the developmental model (in comparison with the approach introduced in [10]) lies in the utilization of higher number of cells of the CA than the number of primary inputs of the circuit to be developed. Obviously, we will consider only the case when the number of cells is greater than the number of inputs of the circuit. Since this feature enables to select different ways of connection of the circuit outputs, we will investigate this issue in more detail in order to determine its impact on the successfulness and computational effort of the evolutionary process.

A suitable method of connection of the gates to the primary inputs in the first developmental step must be

considered because of different number of gates that can be generated in a developmental step. The following approach will be utilized. Let N be the number of inputs of the circuit (and also the number of outputs because multipliers possess the same number of inputs and outputs) and C the number of cells of the CA. If $C > N$, the index range of the primary inputs is lower than the index range of the gates generated by the cells. In order to ensure correct connection in the first developmental step, the modulo operation will be utilized. For example, if an input of a gate has its input specified by the index 10 and a 6-input circuit ought to be developed, this input of the gate will be connected to the primary input of the index $4 = 10\ mod\ 6$ (i.e. the range of cell indices $0 \ldots 9$ is mapped into the range of input indices $0 \ldots 5$. The number of gates generated in the next developmental steps is equivalent, therefore, the inputs of the gates generated in a developmental step can be directly connected to the given outputs of the gates generated in the previous developmental step.

The specified outputs of the gates generated in the last step will directly be connected to the primary outputs of the circuit. However, because the number of cells of the CA is greater than the number of outputs of the circuit, there are several variants of what outputs of the gates will be considered as primary outputs of the circuit. Since the number of gates exceeds the number of primary outputs of the circuit, there are more options how to connect these outputs to the gates generated in the last developmental step. Four different ways of connection will be investigated in this paper: (1) the outputs are considered at the beginning of the range of the cell indices, (2) the outputs are considered at the middle of the range, (3) the outputs are considered at the end of the range and (4) the outputs are considered to be regularly spread within the range of the cell indices. In the second variant, the starting index of the outputs corresponds to the integer value $(C-N)/2$ and the last output possesses the index $(C - N)/2 + N - 1$. In the fourth variant, a special multiplicator P (an integer parameter) is introduced to determine the number of cells of the CA according to the number of inputs of the multiplier as $C = P \cdot (N-1)+1$ in order to ensure regular spreading of the outputs into the index range greater than the number of outputs. In addition, this parameter will be utilized to determine the number of cells of the CAs considered in the experiments for all the variants of the outputs selection and the sizes of the multipliers to be developed. Figure 1 shows an example of selecting outputs of a 4-output circuit developed by means of a 7-cell CA. Note that $P = 2$ in this example, therefore, the cellular automaton consists of $C = 2 \cdot (4 - 1) + 1 = 7$ cells.

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

Figure 2 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

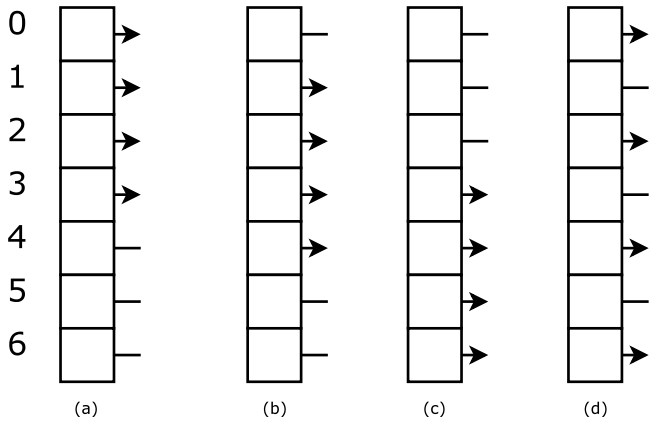


Fig. 1. Configurations for different outputs selections illustrated on the example of 4-output circuit developed by a 7-cell cellular automaton. The pins with arrows denote the selected outputs. (a) Selection from the beginning of the index range, (b) selection from the middle, (c) selection from the end and (d) selection of outputs that are spread regularly within the index range.

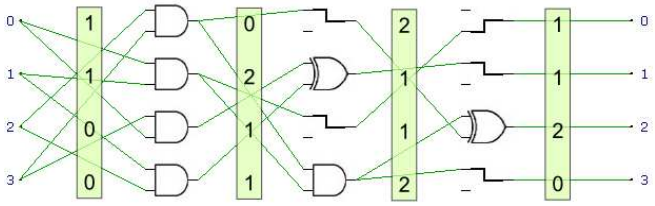
Gate	Inputs	Description
0: AND	a, b	two-input AND gate
1: OR	a, b	two-input OR gate
2: XOR	a, b	two-input exclusive-OR gate
3: $NAND$	a, b	two-input $NAND$ gate
4: NOR	a, b	two-input NOR gate
5: $NXOR$	a, b	two-input gate of equivalence function
6: IDA	a, x	one-bit buffer (identity function) of the first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

TABLE I

GATES UTILIZED FOR THE DEVELOPMENT. NOTE THAT x REPRESENTS AN UNUSED INPUT.

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. 2a). 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)

Fig. 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.

V. Evolutionary System Setup

The simple genetic algorithm was utilized for the evolutionary design of the cellular automaton that generates a specified circuit. The objective is to evolve the initial state of the CA and the local transition function. The form of the chromosome is shown in Figure 3. 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 $0\ 1\ 1 \rightarrow 0 : XOR\ 0\ 1$. Since the combination of states $0\ 1\ 1$ corresponds to the binary representation of number 3, this rule will be placed in the chromosome at the position 3 of the local transition function. Note that the rule is encoded as a sequence of integers $0\ 2\ 0\ 1$.

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, 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

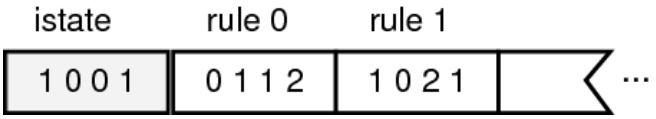


Fig. 3. A chromosome consists of the initial states of the CA to be evolved (denoted as istate) and the set of rules of the local transition function. Each rule contains the next state, gate function and indices of two inputs of the gate respectively. The combinations of states in the cellular neighborhood are encoded implicitly by the indices of rules in the chromosome.

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 genome undergoing changes in order 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.

VI. Experimental results and discussion

The goal was to design a uniform cellular automaton (i.e. its initial state and the local transition function) by means of which a working multiplier of a given number of inputs could be developed. Two sets of experiments were conducted, considering the development of 2x3-bit and 3x3-bit multipliers. We tried to develop as large circuits as possible using the proposed model. The evolution of 2x3-bit and 3x3-bit multipliers exhibits a reasonable computational effort that allows us to perform sufficient number of experiments for several different experimental setups that allows us to identify suitable setup parameters and their impact on the success rate of the evolutionary process. In each set of experiments, different sizes of the cellular automata (let us denote it W), number of possible states of the cells (S), and ways of connections of the outputs of the multipliers were considered. Let us denote the outputs connections by the abbreviations as follows: the connection at the beginning of the gate sequence (B), the connection at the middle of the gate sequence (M), at the end (E) and the spread outputs (P). For each setup of these parameters, the 2x3-bit combinational multipliers were developed using 3 and 4 developmental

steps (corresponding to the delay of the resulting circuits) and the 3x3-bit multipliers were developed using 4 and 5 developmental steps. These values showed to be suitable to develop a working multiplier of the given size by means of the cellular automaton, using the building blocks from table I.

The experimental results demonstrate the success of the development of 2x3-bit and 3x3-bit multipliers at the gate level which confirmed our initial assumption that larger multipliers can be developed by increasing the number of cells of the CA. We were not able to obtain such sizes of multipliers using the original developmental model [10]. One of the best 3x3-bit multiplier developed by means of this approach is shown in Figure 4. This circuit has been optimized by removing the meaningless gates (e.g. AND gates containing identical inputs, gates whose outputs are not connected to any input of another gate or primary output etc.). After this optimization, 30 gates are needed and the resulting multiplier possesses delay of 5 gates. Although this result is not optimal in comparison with the best known solution, it represents a significant contribution of this paper because such size of multipliers has not been obtained before using the cellular automata.

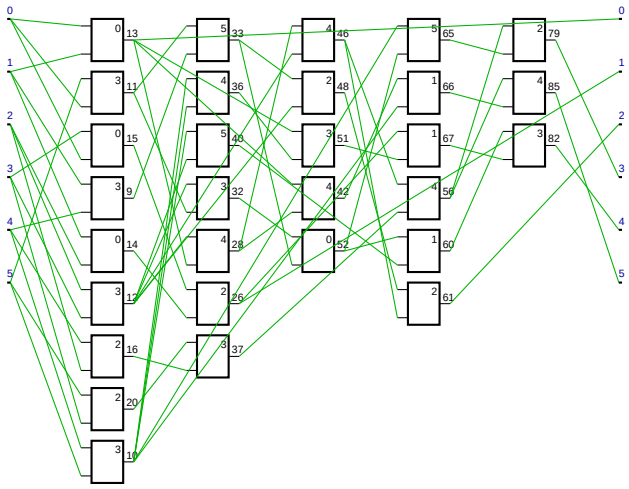


Fig. 4. One of the best 3x3-bit multipliers developed by means of cellular automata designed by the evolutionary algorithm. Note that the arrangement of the gates against the structure generated by the cells was altered by the optimization (i.e. removing the unused and meaningless gates) performed by our visualisation software.

For each experimental setup (i.e. different values of parameters W, S, T, size of the target multipliers and the way of outputs connection) 200 independent runs of the evolutionary process were executed. If no solution is found in 250k generations in case of 2x3-bit multipliers and in 1M5 generations in case of 3x3-bit multipliers, the evolution is stopped. The success rate and the average number of generations needed to evolve a working solution were measured. The experimental results for the evolution of 2x3-bit multipliers are summarized in Table II and for the evolution of 3x3-bit circuits in Table III.

TABLE II

SUMMARY OF THE SUCCESS RATE AND AVERAGE NUMBER OF GENERATIONS FOR THE EXPERIMENTS OF EVOLUTION OF 2X3-BIT MULTIPLIERS USING CELLULAR AUTOMATA. SEVERAL DIFFERENT NUMBERS OF CELLS (DENOTED BY W) WERE CONSIDERED, FOR EACH SIZE OF THE CA THE NUMBER OF CELL STATES (S) AND THE NUMBER OF DEVELOPMENTAL STEPS (T) WERE DETERMINED EXPERIMENTALLY. DIFFERENT WAYS OF CONNECTION OF THE CIRCUIT OUTPUTS WERE INVESTIGATED (P, S, M, E).

			success rate (%)				avg. # generations			
W	S	T	P	S	M	E	P	S	M	E
9	3	3	8	7	12	9	157k	106k	156k	162k
		4	5	4	3	6	145k	182k	172k	177k
	4	3	40	55	54	48	105k	90k	113k	93k
		4	54	52	56	53	102k	98k	107k	102k
	5	3	63	64	53	62	77k	66k	72k	69k
		4	90	88	93	92	45k	48k	45k	56k
13	3	3	16	23	17	17	167k	152k	139k	125k
		4	15	12	14	7	146k	167k	146k	144k
	4	3	69	72	68	60	90k	97k	85k	85k
		4	87	81	77	78	73k	84k	84k	68k
	5	3	92	80	77	76	73k	66k	56k	66k
		4	96	97	98	99	41k	31k	36k	39k
17	3	3	24	23	24	17	161k	157k	139k	142k
		4	19	14	12	21	166k	175k	152k	146k
	4	3	68	67	57	75	103k	90k	109k	95k
		4	91	85	79	92	82k	83k	76k	76k
	5	3	94	84	87	86	83k	66k	73k	63k
		4	99	97	100	99	34k	37k	37k	33k
21	3	3	19	24	22	25	135k	124k	165k	152k
		4	21	14	16	20	174k	135k	164k	160k
	4	3	65	74	63	75	91k	88k	81k	89k
		4	96	87	85	83	77k	78k	72k	76k
	5	3	88	93	87	90	69k	67k	66k	69k
		4	100	97	98	100	32k	32k	38k	37k
25	3	3	25	25	16	19	122k	135k	142k	152k
		4	24	15	11	24	144k	136k	155k	135k
	4	3	64	69	71	76	85k	97k	82k	107k
		4	96	91	87	90	69k	77k	67k	89k
	5	3	91	90	88	87	70k	80k	74k	76k
		4	99	98	99	98	39k	33k	43k	35k

As the results show, the success rate exceeds 90% in many cases of experiments related to the development of 2x3-bit multipliers (see Table II). If four states (S=4) are utilized, the success rate is much higher (even several times) in comparison with the experimental setup possessing S=3. If the setups of S=4 and S=5 are compared, the increase of the success rate is lower. Surprisingly, the size of the CA does not lead to any significant increase of the success rate. In some cases the larger CA even causes a worse success rate in comparison with the smaller CAs. It is interesting because the higher number of cells (in comparison with the number of circuit inputs) just enabled to design larger multipliers. Note that very good success rates have already been achieved

TABLE III

SUMMARY OF THE SUCCESS RATE AND AVERAGE NUMBER OF GENERATIONS FOR THE EXPERIMENTS OF EVOLUTION OF 3X3-BIT MULTIPLIERS USING CELLULAR AUTOMATA. THE COLUMNS HAVE THE SAME MEANING AS IN TABLE II.

			success rate (%)				avg. # generations			
W	S	T	P	S	M	E	P	S	M	E
11	4	5	0	0	0	0	-	-	-	-
		6	0	0	0	0	-	-	-	-
	5	5	1	1	1	3	806k	1M	1M	660k
		6	0	0	3	2	-	-	818k	1M
	6	5	14	4	8	10	971k	718k	720k	812k
		6	15	10	9	7	537k	886k	653k	1M
16	4	5	0	1	0	0	-	848k	-	-
		6	0	0	1	0	-	-	1M	-
	5	5	18	14	16	16	882k	963k	836k	728k
		6	17	17	10	17	843k	792k	746k	873k
	6	5	47	42	47	39	816k	692k	631k	755k
		6	36	39	41	47	728k	668k	780k	752k
21	4	5	1	1	0	0	1M	892k	-	-
		6	0	0	0	0	-	-	-	-
	5	5	36	30	32	30	802k	899k	788k	755k
		6	28	28	33	28	831k	862k	853k	807k
	6	5	67	57	57	56	615k	699k	669k	635k
		6	61	64	69	62	658k	572k	552k	666k
26	4	5	1	2	1	0	816k	1M	996k	-
		6	1	0	1	0	285k	-	1M	-
	5	5	39	41	23	30	789k	821k	771k	737k
		6	34	26	37	26	782k	651k	863k	864k
	6	5	70	66	67	60	662k	680k	617k	609k
		6	78	73	74	74	565k	693k	587k	564k
31	4	5	2	2	2	2	1M	1M	1M	928k
		6	2	1	0	0	828k	1M	-	-
	5	5	45	36	33	40	778k	815k	773k	830k
		6	44	35	36	41	838k	735k	800k	852k
	6	5	78	63	66	75	679k	650k	607k	553k
		6	84	81	81	77	500k	518k	531k	599k

with 13 cells of the CA; if larger sizes of the CA is used, the success rate exhibits only small improvements. It is possible to observe similar behavior in the average number of generations (with contradictory values). If three cell states (S=3) are considered, there is a higher computational effort (i.e. higher number of generations is needed in average to evolve a working solution) in comparison with higher numbers of states. However, it is possible to observe significantly lower computational effort for 5 states and 4 developmental steps in comparison with the rest of the experimental setups though the increased number of states leads to exponential increase in the size of the search space. In the worst case, a working multiplier was evolved in 3 minutes. The smaller CAs with a lower number of cell states exhibit higher computational effort in comparison with larger CAs that provide more possibilities how the working solution can be developed.

As expected, the evolution of CAs for the development of 3x3-bit multipliers is more difficult. More states, developmental steps and larger CAs were needed to evolve working solutions (see Table III). In case of 4 cell states, only a few of functional multipliers were developed. In most cases, the evolution failed even in 1M5 generations using this experimental setup. However, it is possible to achieve significantly better success rate for the number of states 5 and higher. In addition, as evident from Table III, the success rate increases with increasing the size of the CA in most cases. For example, the 26- and 31-cell CAs with 6 states that construct 3x3-bit multipliers in 6 developmental steps can be evolved with the success rate exceeding 70% which is a perfect result if the complexity of the CA-based construction process and the gate-level structure of these multipliers is considered. The computational effort, measured as the average number of generations needed to evolve a working solution, exhibit similar rates like in the experiments dealing with the 2x3-bit multipliers. In the worst case, a working multiplier was evolved in 45 minutes. Such a high increase in the evolution time in comparison with the 2x3-bit multipliers is caused by the exponential increase of the evaluation time and the complexity of the structure of the 3x3-bit multipliers that needs to be designed.

As evident from both the experiments dealing with the construction of 2x3-bit multipliers (Table II) and 3x3-bit multipliers (Table III), the way of the outputs connection does not influence the success rate and computational effort very markedly. The difference is usually a few of percents only for the specific experimental setup. This is illustrated by Figure 5, 6 and 7 that shows the dependence of success rate on the cellular automaton size for the investigated connections of the circuit outputs, processed for the development of 3x3-bit multipliers using the experimental setup (1) S=5, T=6, (2) S=6, T=5 and (3) S=6, T=6 respectively. In fact, this feature is not surprising. If there is a sufficient number of possible combinations of states in the cellular neighborhood, the evolution may generate a variety of different gates and their interconnections (using different rules of the local transition function) which are required to design a working multiplier for the given outputs connections.

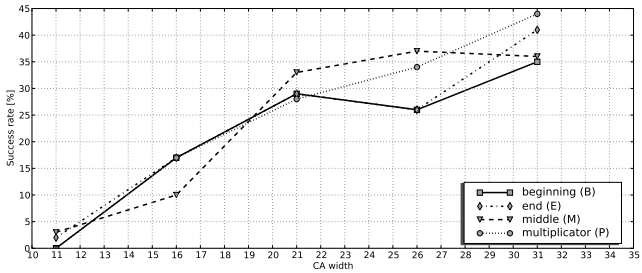


Fig. 5. Dependence of the evolution success rate on the size of the CA, considering different ways of the outputs connection. Experimental setup: S=5, T=6.

The experiments showed that it is possible to evolve a lot of different CA that develop various structures of the target circuit. In case of 2x3-bit multipliers, the convergence of the evolutionary process is very fast considering the large size

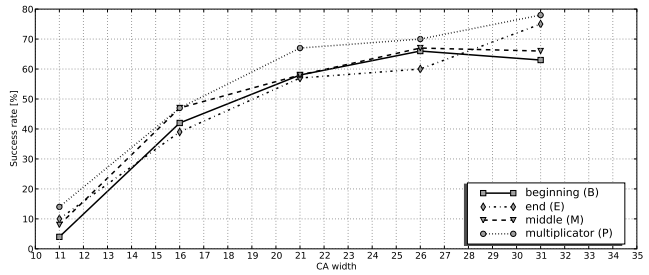


Fig. 6. Dependence of the evolution success rate on the size of the CA, considering different ways of the outputs connection. Experimental setup: S=6, T=5.

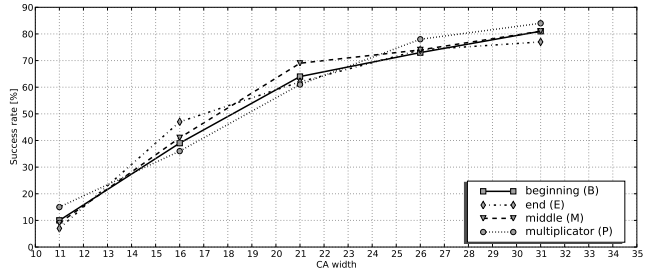


Fig. 7. Dependence of the evolution success rate on the size of the CA, considering different ways of the outputs connection. Experimental setup: S=6, T=6.

of the search space (its size is approximately $1.2 \cdot 10^{93}$ in the simplest case dealing with the 9-cell CA for the design of 2x3-bit multiplier). Similar behavior of the evolutionary process was observed in case of 3x3-bit multipliers when CA larger than 20 cells was applied (of course, there is a higher computational effort) which indicates a good level of evolvability using the CA-based generative representation.

Unfortunately, we have not been able to obtain any working solution for 3x4-bit multipliers. This is probably caused by the exponential increase in the search space with increasing the number of the cell states that is needed to generate more complex gate-level structures. It means that the problem of scale represents the main issue of this approach. In fact, the developmental system in the form in which it has been presented is not suitable to scale up the circuit during the development process which could enable to overcome this issue. For example, we could enable the CA to ”grow” during its development and allow to determine the size of the structure generated in a given developmental step by the stage of the development process (i.e. more advanced developmental steps could produce more gates). Moreover, the system works with basic logic gates that represent a certain amount of information supplied to the system that may not be sufficient to develop larger circuits. In fact, the increasing number of cells can be understood as increasing the amount of information supplied to the developmental system in the form of the resources that can be utilized for the circuit development. The results show that more resources lead to higher success rate (i.e. easier evolution of the CA) – see Table II and III. It could be useful to introduce a capability of supplying with domain-specific information to the developmental system in order to develop larger multipliers (e.g. more complex building blocks).

Despite of the problems mentioned in the previous paragraphs, the approach we presented demonstrates a variant of generative encoding as an alternative to the direct encoding in the gate-level evolutionary design of digital circuits. Since the primary experiments were successful in most cases, this approach may represent a basis for our next research in which those issues will be addressed.

VII. Conclusions

Cellular automata were utilized as the developmental model for the evolutionary design of combinational multipliers at the gate level. Since the multipliers are usually considered as the class of circuits that are difficult to design using evolutionary algorithms, we chose them in order to demonstrate capabilities of the proposed enhanced cellular automata-based developmental model.

By allowing higher number of cells in comparison with the number of circuit inputs, it is possible to develop larger multipliers (it was demonstrated on the development of 2x3-bit and 3x3-bit multipliers). Moreover, the obtained results represent the first case when such sizes of gate-level multipliers have successfully been developed using the cellular automata.

The research was focused on investigating the influence of the number of cells of the CA on the success rate and the number of generations that is needed to evolve a working solution. A perfect success rate was observed in some cases that exceeds 90% in the evolution of 2x3-bit multipliers and 70% in case of 3x3-bit multipliers. The success rate can significantly be influenced by the size (number of cells) of the utilized cellular automaton. Moreover, several different ways of connection of the multiplier outputs were considered. The experiments showed that the outputs connection has only a very small impact on the success rate and the computational effort.

The development of digital circuits by means of cellular automata represents a more difficult task in comparison with the case if a direct encoding is used (e.g. cartesian genetic programming). In fact, the evolution has to design both the initial state and the local transition function together with the gates (including their interconnections) to be generated during the CA development. This process would be very difficult to perform manually because no systematic approach exists how to design a cellular automaton to exhibit a given behavior. The construction of digital circuits using cellular automata is performed at the structural level, i.e. the circuit structure emerges as the “second” product of the CA development, in addition to updating the cell states. Because of this issue, the evolved solutions and the principles of the circuit construction process is often not fully understood. Therefore, more investigation is needed regarding the analysis of specific CAs and the resulting circuits. The multipliers produced by the proposed developmental system are not optimal if the number of gates or delay is considered

in comparison with the best known solutions. Actually, the optimization was not the goal of this paper, the circuits were only evaluated according to their functionality. Therefore, these issues represent possible topics for our next research.

ACKNOWLEDGMENT

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

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

[3] S. Wolfram, *A New Kind of Science*. Champaign IL: Wolfram Media, 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*. Cambridge, MA: MIT Press-Bradford Books, 1996, pp. 393–401.

[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*. Springer, 2000, pp. 31–39.

[6] 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*. IEEE Computer Society, 1995, p. 146.

[7] 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*. Dortmund DE: Springer, 2003, pp. 256–265.

[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*. Los Alamitos, CA, US: IEEE Computer Society, 2001, pp. 109–115.

[9] G. Tufte and P. C. Haddow, "Towards development on a silicon-based cellular computing machine," *Natural Computing*, vol. 4, no. 4, pp. 387–416, 2005.

[10] 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*. IEEE Computer Society, 2008, pp. 11–18.

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

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

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

[14] J. F. Miller and P. Thomson, "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, pp. 121–132.