Global Control In Polymorphic Cellular Automata*

LUKAS SEKANINA†, TOMAS KOMENDA‡

Faculty of Information Technology, Brno University of Technology
Bozetechova 2, 612 66 Brno, Czech Republic

Received 10 March 2010; In final form 29 April 2010

The notion of locality is fundamental for cellular automata. However, introducing a kind of global information to some or even all the cells can significantly increase the effectiveness of computation. We used a two-value discrete global signal which allowed the cells to change the local transition function during computation. On the basis of the signal we could generate new patters in a simple 1D cellular automaton and accelerate self-replication of Byl's loop in a 2D cellular automaton (the speedup obtained is 48%). In the case of 1D automaton we demonstrated that the overhead introduced with the global control can be relatively small if the implementation is performed using polymorphic gates controlled by the level of power supply voltage.

Key words: Cellular Automata, Polymorphic Circuits, Self-Replication, Global Control, Cellular Hardware

1 INTRODUCTION

Cellular automata (CA) are spatially-distributed dynamical system models consisting of many relatively simple computing elements (cells) [26, 4, 25,

* This is authors' created manuscript of the paper: Sekanina Lukas, Komenda Tomas: Global Control In Polymorphic Cellular Automata. Journal of Cellular Automata, Vol. 6, No. 4-5, 2011, p. 301-321. For the final version, see Old City Publishing, Inc. at http://www.oldcitypublishing.com/JCA/JCAcontents/JCAv6n4-5contents.html

† email: sekanina@fit.vutbr.cz

‡ email: KomendaTomas@seznam.cz

27]. All interactions among the cells are purely local. The cells usually contain and exchange only a small amount of information. Despite of their simplicity, cellular automata can exhibit emergent properties, including dynamical behaviors, that are not deducible from their elementary components.

The notion of locality is one of the most crucial concepts behind the idea of cellular automata. One implication of this principle is that no one cell has a global view of the entire system, i.e. there is no central controller [20]. However, introducing a kind of global information to some or even all the cells can significantly increase the effectiveness of computation. For example, Chandler has introduced a cellular automaton that chooses the function to apply (by a particular cell) on the basis of the global state of the automaton. Resulting cellular automata with global control proved quite useful in modeling interactions in which behavior depends on the behavior of neighbors and on global information produced by some sort of aggregating mechanism [3]. The global control is also important when a cellular system has to be initialized to a particular state or some results of computations have to be stored at a given time point.

The implementation of a global control is not difficult when cellular automata are modeled in software because each cell is accessible in a constant time. However, as physical fabrication of cellular automata-based hardware truly utilizes the concept of locality (only neighboring cells are physically connected) there is limited or no way how to access each particular cell from boundary cells. Software implementations could be also slow in cases when really large grids (such as a cellular machine with 10^{23} cells proposed in [5]) have to be simulated. Various cellular automata-based platforms have been developed [25, 5, 9, 18, 19]. One of the approaches is to create an array of simple locally connected electronic circuits – cells (see CellMatrix [5] and Embryonics [9] platforms). The problem with the global control is that in addition to routing of power supply voltage and synchronization signal (if the cellular automaton is synchronous) to every cell, another global control signal has to be distributed on the chip. Having millions of cells, routing the additional signals will be very area expensive. Hence an ideal implementation from the hardware point of view should not use any wires connected to all the cells expect the power supply connections.

The first goal of this article is to show that there is another way how to globally change the local function of the cells. The solution is based on the implementation of local transition functions using polymorphic gates. A polymorphic gate is capable of switching among two or more logic functions as a response to some external factors, e.g. to the level of the power sup-

ply voltage (V_{dd}), temperature or light [22, 23, 21, 30, 13]. If the transition function is implemented as a digital circuit using polymorphic gates sensitive to V_{dd}, all the cells can in parallel reconfigure their local transition functions when V_{dd} is changed. Then, the global control can be achieved without the need of special routing signals. This phenomenon will be demonstrated using a polymorphic chip REPOMO32 [16]. Another interesting property of polymorphic gates is that they, in fact, act as sensors when controlled by temperature or light. Cellular automaton containing such polymorphic gates can be dynamically reconfigured as response to changing environments.

The second goal is to demonstrate that the global control can be utilized to accelerate some computations typically conducted using cellular automata. Particularly, we will show that the number of steps needed to self-replicate the Byl's loop [2] can significantly be reduced if the rule set is changed in precomputed time points. As self-replicating mechanisms are of great interest for artificial life, programming massively parallel computers, nanocomputing, computer viruses and other fields, the investigation of fast self-replicating structures is important.

The paper is organized as follows. Section 2 introduces one- and two-dimensional cellular automata and their basic modifications, including the global control. Section 2.3 surveys the approaches developed to self-replication in the framework of cellular automata. In Section 3, the concept of polymorphic circuits is described. Section 4 is devoted to the first case study: design of a simple one-dimensional globally controlled cellular automaton which is implemented by means of polymorphic gates controlled using V_{dd}. The second case study – Byl's loop and the acceleration of its self-replication using the concept of global control – is described in Section 5. Discussion of obtained results is presented in Section 6. Conclusions are given in Section 7.

2 CELLULAR AUTOMATA AND SELF-REPLICATION

2.1 Cellular Automata

A cellular automaton is a d-dimensional grid of finite automata (known as *cells*) that operate according to their *local transition functions*. In a classic scenario, the cells work synchronously – a new state of every cell is calculated from its previous state and the previous states of the cell's 'neighbors' at each time step. If all automata of the grid are identical (that is, the automata operate according the same local transition function, *rule*) the automaton is called *uniform*, otherwise it is *non-uniform*. By *configuration* of the cellular automaton we mean the states of all the cells at a given moment. The

global behavior is captured in the *global transition function*, which defines a transformation from one configuration to the next configuration of the cellular automaton. The sequence of configurations, determined by the global transition function, represents the *computation* of the cellular automaton.

In theory, the cellular automaton model supposes that the number of cells is infinite. However, in the case of practical applications the number of cells is finite. Then, it is necessary to define the *boundary conditions*, i.e. the setting of the boundary cells. One of the states is also usually used as *quiescent* or inactive state. A convention is that when a quiescent cell has an entirely quiescent neighborhood, it will remain quiescent at the next time step.

A simple one-dimensional uniform cellular automaton, with only two states and nearest neighbors neighborhood N = {-1, 0, 1} (only left and right neighbor cells together with the cell itself are relevant for the local transition function), can exhibit very complex behavior [27]. Each such cellular automaton is defined by a mapping $\{0,1\}^N \rightarrow \{0,1\}$ uniquely. Hence there are 2^8 such cellular automata, each of which is uniquely specified by the (transition) rule i ($0 \leq i < 256$).

In case of two-dimensional cellular automata, the neighborhood usually comprises of five or nine cells (see Fig. 1). We usually include the central cell into its own neighborhood. In the 5-neighbor model, the individual transition rules would be in the form CTRBL $\rightarrow C^{'}$ where CTRBL specifies the states of the Center, Top, Right, Bottom, and Left positions of the neighborhood's present state and $C^{'}$ represents the next state of the center cell.

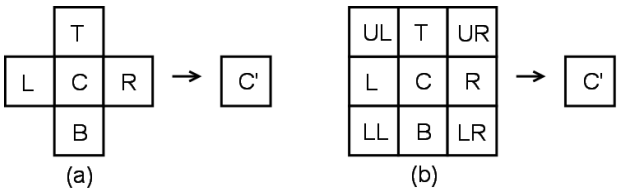

FIGURE 1
Typical neighborhoods in 2D cellular automata: (a) von Neumann neighborhood; (b) Moore neighborhood

Cellular automata have been applied in many scientific areas, especially for the modeling of complex (biological, chemical, computational, etc.) phenomena. Because the number of cells is typically very large, the cellular automaton can model a massively parallel computational system where some

useful computation can *emerge* only on the base of local interactions. The properties of cellular automata have been investigated by means of analytic as well as experimental methods [6, 19, 25, 27, 28]. In general, the objectives are twofold: (i) to find a method for the design of cellular automaton rules for a given application or (ii) to predict the global behavior of a given cellular automaton if the rules and the initial configuration are known.

2.2 Modifications of the Basic Model

In order to obtain specific behaviors, the concept of cellular automata has been modified in several ways, for example:

- *Non-uniform* cellular automata were introduced which do not employ the same rule in each cell. Sipper has shown that it is easier to achieve computational universality with non-uniform cellular automaton than with the uniform one [19].

- In another modification – *non-local cellular automaton* – the concept of locality is not satisfied [14]. Although the non-locality is not plausible from the point of view of physics, dynamical systems with non-local connections have potential applications to economic and biological systems.

- *Asynchronous* cellular automata do not update the cells in identical time points [1]. The asynchronous operation is typical for large cellular automata in which it is difficult to ensure precise synchronization for every cell.

We will mainly be interested in cellular automata with a *global control*. One of possible approaches to achieve it is to observe the global state of the automaton and perform scheduled actions when a particular global state is detected [3]. A possible strategy could be to observe the parity (or majority or any relevant function) of the cellular automaton at a given point. If the parity is, for example, even the rule is modified; otherwise the rule remains unchanged. Another option is to modify the rule when a particular step of the automaton is achieved. Finally, by combining mentioned strategies, rich spatio-temporal interactions can be obtained.

2.3 Self-Replication

Self-replication represents one of key principles which is intensively studied using cellular automata in the field of artificial life. The goal is to find such

rules for cellular automata which will be able to create a copy of a given configuration only on the basis of local interactions. Performing self-replication effectively is crucial for configuring and repairing some massively parallel systems such as bio-inspired hardware [5, 9].

The first study on self-replicating machines was conducted by von Neumann and Ulam who created a 29-state 5-neighbor cellular system with the ability of universal construction/computation, replication of structures and self-replication [26]. The automaton rules were designed in such a way that they simulated a digital circuit working in the manner similar to Turing machine. However, because of the overall complexity of the model it was impossible to simulate it on a computer at that time.

Another significant contribution to self-replication was due Langton who proposed self-reproducing loops (known as Langton loops or Q loops) [7]. Langton used 8-state 5-neigbor cellular automaton. Langton's loop consists of cells containing 'genetic information' (modelled using special states in the cells), which flows continuously around the loop and out along an 'arm' which is forming the daughter loop. These relatively small structures (in comparison to von Neumann automaton) were not designed to be universal since Langton postulated that the constructional universality of automata is not necessary condition for self-reproduction. This idea was generally accepted and the approaches to self-replication based on loops have dominated until now. We will describe this concept in greater detail in Section 5 where we will deal with Byl's loop which is a smaller version of Langton's self-replicating loop. While Langton's loop (consisting of 86 cells) replicates in 151 steps [7], Byl's loop (consisting of 13 cells) creates its own copy in 25 steps [2].

Table 1 surveys major work in the area of self-replication. In order to overcome difficult and time consuming design process of self-replicating structures, Lohn and Reggia have used genetic algorithms to design these structures automatically [8].

So far, we have discussed self-replication in the logic sense. Attempts to perform physical self-replication were also reported, see, for example, self-replicating robots [31].

3 POLYMORPHIC CIRCUITS

Polymorphic electronics was introduced by A. Stoica's group at NASA Jet Propulsion Laboratory as a new class of electronic devices that exhibits a new style of (re)configuration [22]. Polymorphic gates play the central role in the polymorphic electronics. For example, Stoica's polymorphic bifunc-

TABLE 1
Major approaches to self-replication using cellular automata

Loop	Neigborhood	Number of states	Numer of cells	Steps to replicate
Langton's loop [7]	von Neumann	8	86	151
Byl's loop [2]	von Neumann	6	13	25
Chou-Reggia's loop [12]	von Neumann	8	5	15
Tempesti's loop [24]	Moore	10	148	304
Perrier's loop [11]	von Neumann	64	158	235
SDSR loop [15]	von Neumann	9	86	151
Evoloop [15]	von Neumann	9	149	363
Sexyloop [10]	von Neumann	10	various	various

tional NAND/NOR gate controlled by V_{dd} operates as NOR for V_{dd} = 1.8 V and NAND for V_{dd} = 3.3 V [21]. Another gate was developed to operate as AND when temperature is 27°C and as OR when temperature is 125°C. For a survey of polymorphic electronics, see [17].

The main motivation for the use of polymorphic gates is to obtain reconfigurable (and thus potentially adaptive) circuits for a very low cost and without the need to implement the reconfiguration infrastructure (switches, multiplexers, configuration registers etc.). Figure 2 shows an example of polymorphic digital circuit and its equivalent behavior in both modes of the polymorphic NAND/NOR gate (i.e., $y = \overline{(a \oplus b) \wedge c}$ and $y = \overline{(a \oplus b) \vee c}$). Although polymorphic gates can be implemented relatively effectively using current CMOS technology, we can expect the expansion of polymorphic devices with further development of nanoelectronics and molecular electronics.

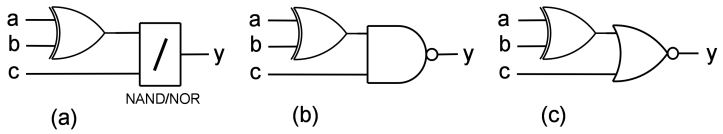

FIGURE 2
(a) Example of a polymorphic circuit; (b) Equivalent circuit in mode 1;
(c) Equivalent circuit in mode 2

Another NAND/NOR gate controlled by V_{dd} was utilized in the REPOMO32 chip which is an experimental reconfigurable platform for development and investigation of small combinational polymorphic circuits [16]. REPOMO32 consists of 32 two-input Configurable Logic Elements; each of them can be programmed to perform one of the following functions: AND, OR, XOR and polymorphic NAND/NOR (controlled by V_{dd}). When V_{dd} = 3.3V the NAND/NOR gate exhibits the NOR function and when V_{dd} = 5V the gate exhibits the NAND function. Remaining gates do not change their logic functions with the changes of V_{dd} (for 3–5 V). The chip was fabricated in a 0.7-micron AMIS technology. The REPOMO32 chip is used in experiments that are reported in Section 5.

4 GLOBAL CONTROL OF A SIMPLE CELLULAR AUTOMATON

The objective of this section is to present an example of cellular automaton implementation whose behavior can be controlled globally using the level of V_{dd}. We will consider a 1D cellular automaton whose cells are implemented as individual digital circuits. Figure 3 shows that the next state of the cell is computed using a combinational block. This block can be implemented either as a look up table (LUT) or, more efficiently, as a combinational logic.

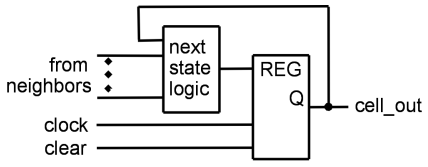

FIGURE 3
Circuit implementation of a cell

In order to create a cellular automaton which is globally controlled using polymorphic gates, we will use polymorphic gates to implement the local transition functions. The automaton is uniform, with only two states and nearest neighbors neighborhood. We will demonstrate how the changes in V_{dd} level affect the behavior of the automaton. Two rules were chosen for this example: Rule 150 (which is the 3-bit parity function) is the first rule which the automaton uses when V_{dd}=3.9-5 V. Once the V_{dd} drops below 3.8 V the automaton works according to rule 232 (the 3-bit majority function).

Figure 4 shows the implementation of the local transition function using ordinary and polymorphic gates. The circuit consists of two polymorphic NAND/NOR gates, two XOR gates, two OR gates and two AND gates. The circuit was designed using Cartesian Genetic Programming adopted for synthesis of polymorphic circuits [17]. We implemented the circuit in REPOMO32 and measured its response for all possible assignments to the inputs and for two V_{dd} levels (see Fig. 5). We can observe that the change of the local transition function can really be controlled globally and without the need for a special signal determining the function of the cells. The implementation using polymorphic gates is also relatively efficient. A conventional solution (without polymorphic gates but with a control signal used for function selection) would require two circuits – the 3-bit parity and 3-bit majority - multiplexed according to the global signal g using a two-input multiplexer (see Fig. 6). As the 3-bit parity is implemented using two two-input XOR gates, the majority using four two-input gates and the multiplexer using three two-input gates the conventional solution would cost 9 gates which is more expensive than the polymorphic solution if the number of two-input gates is counted.

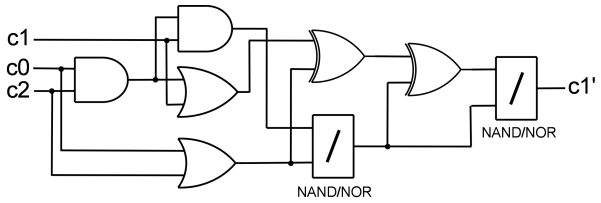

FIGURE 4
An implementation of polymorphic 3-bit majority/3-bit parity function (rule 150/232)

The global behavior of the automaton is shown in Fig. 7 for the following setting (i.e., for the V_{dd} levels): Rule 150 is applied for 11 steps, then rule 232 is used for 5 steps and the sequence is repeated five times. By introducing the global control we obtained the patterns that no single CA of this type can produce individually. The role of rule 232 is to keep some patterns unchanged and delete all independent 1s. More examples are visible in Fig. 8 and Fig. 9.

Finally, the implementation of the global signal controller is straightforward. In our case, a counter equipped with detector of a particular value will control the power supply unit.

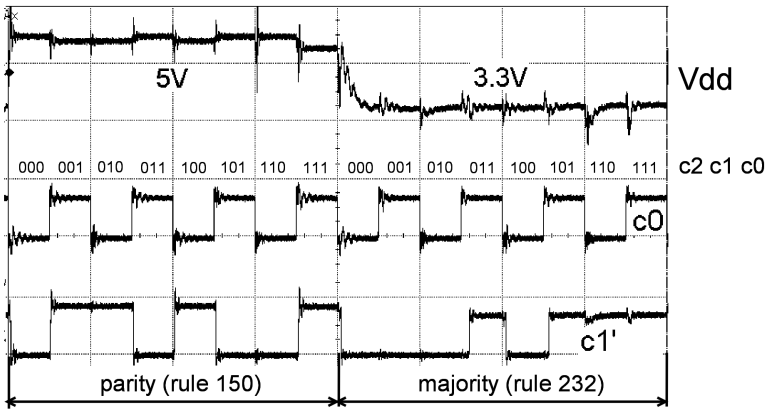

FIGURE 5
Behavior of the 3-bit parity/majority circuit (at 1 MHz). Only the least significant input bit (c0) is shown.

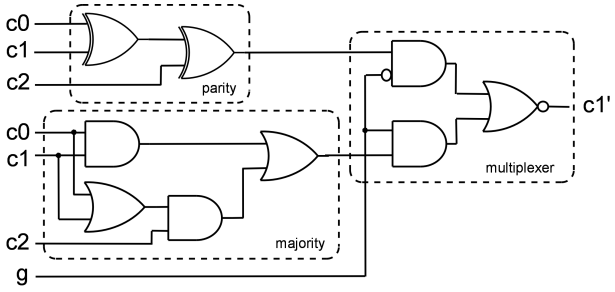

FIGURE 6
Conventional multiplexing of the 3-bit majority and 3-bit parity

5 GLOBALLY CONTROLLED SELF-REPLICATION

We will demonstrate that the idea of global control can be utilized to reduce the number of steps that have to be performed in order to replicate Byl's loop.

5.1 Byl's Self-Replicating Loop

Figure 10 shows that Byl's loop consists of 13 cells; each of them is in one of the six possible states (0–5), where 0 is depicted as white square [2].

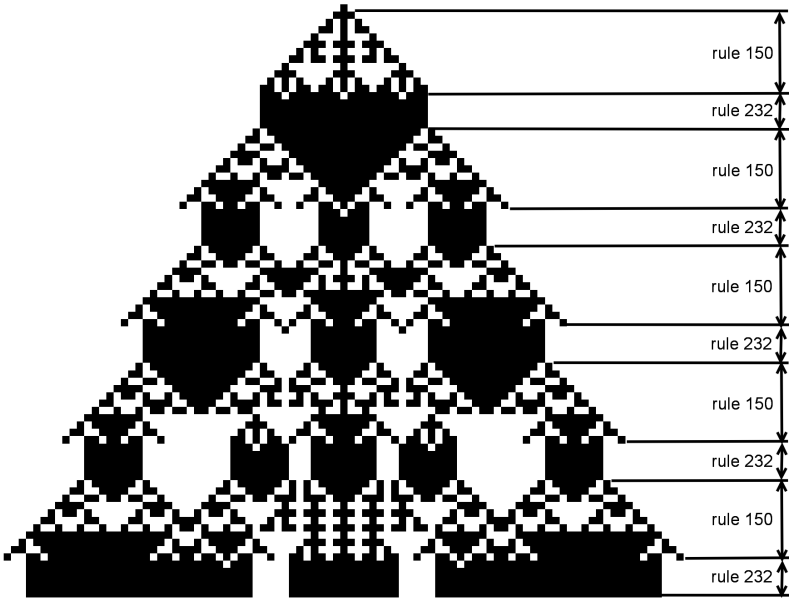

FIGURE 7
Development of 1D cellular automaton with rule 150 (11 steps), rule 232 (5 steps), 150 (11 steps), rule 232 (5 steps) etc.

The automaton uses a 5-neighbor neighborhood. The four inner cells create so-called *kernel* which rotates during automaton development. The set of rules that are required to replicate Byl's loop is given in Appendix. The process of self-replication is similar to that used by well-known Langton's self-replicating loop. The cell possessing state 5 (together with its neighbor) forms a gate where a new arm (and then a new cell) grows from. The self-replication is complete in 25 steps (see Fig. 10).

Figure 11 shows a colony of Byl's loops. There are three loops in the center of the colony that do not have space to replicate. Despite this fact, the kernels of these loops rotate, open their gates and try to replicate; however, unsuccessfully.

FIGURE 8
Development of 1D cellular automaton with rule 150 (15 steps), rule 232 (3 steps), 150 (15 steps), rule 232 (3 steps) etc.

5.2 Accelerated Replication of Byl's Loop

In order to accelerate the self-replication we will utilize two sets of rules. The first set is used in steps 1-9. Next four steps are performed using the second set of rules. At the end of the 13th step, the replication is finished and we obtain the same result as after 25 steps of the original approach (see Fig 12). Now, the first set of rules has to be activated for steps 14-22, the second set of rules takes place in steps 23-26 etc. Therefore, the pulse ratio of global control signal is 9:4 in this case. Note that the kernel does not rotate when the second set of rules is active. The overall reduction in time is 48% in comparison with the original Byl's loop (see the situation after 150 steps in Fig. 13 which also shows the state to color assignment on top-left).

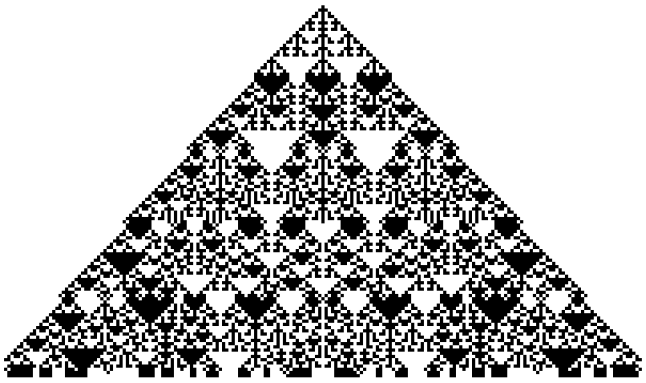

FIGURE 9
Development of 1D cellular automaton with rule 150 (21 steps), rule 232 (2 steps), 150 (21 steps), rule 232 (2 steps) etc.

We choose the step 9 to change the rules because the loop is in a suitable position: There is one disconnected cell in state 1 and the position of the kernel is advantageous. The first (i.e. original) set of rules consists of 238 rules, including all rotations. The second set of rules consists of 277 rules. All the changes in rule set were designed manually after numerous experiments. Appendix 1 summarizes all the rules.

Similarly to the simple automaton reported in Section 4 we can implement the transition function using a polymorphic circuit and control the replication using power supply voltage. In this case the circuit would have five 3-bit inputs and one 3-bit output as three bits are needed to encode the cell's state. However, the description of design and implementation of the polymorphic circuit is out of the scope of this article.

There is one significant difference of accelerated Byl's loop in comparison to its original version. In the case of accelerated Byl's loops, the kernels of the inner cells of the colony stop their rotations after five unsuccessful attempts to replicate. The first 'frozen' loop is visible at the 62nd step of simulation (see Fig 14). This situation is caused by the fact that the second set of rules contains the rule $12433 \rightarrow 4$ (CTRBL$\rightarrow C$) which yields the state 4. After getting back to the first set of rules, the loop does not have a way to go on and freezes.

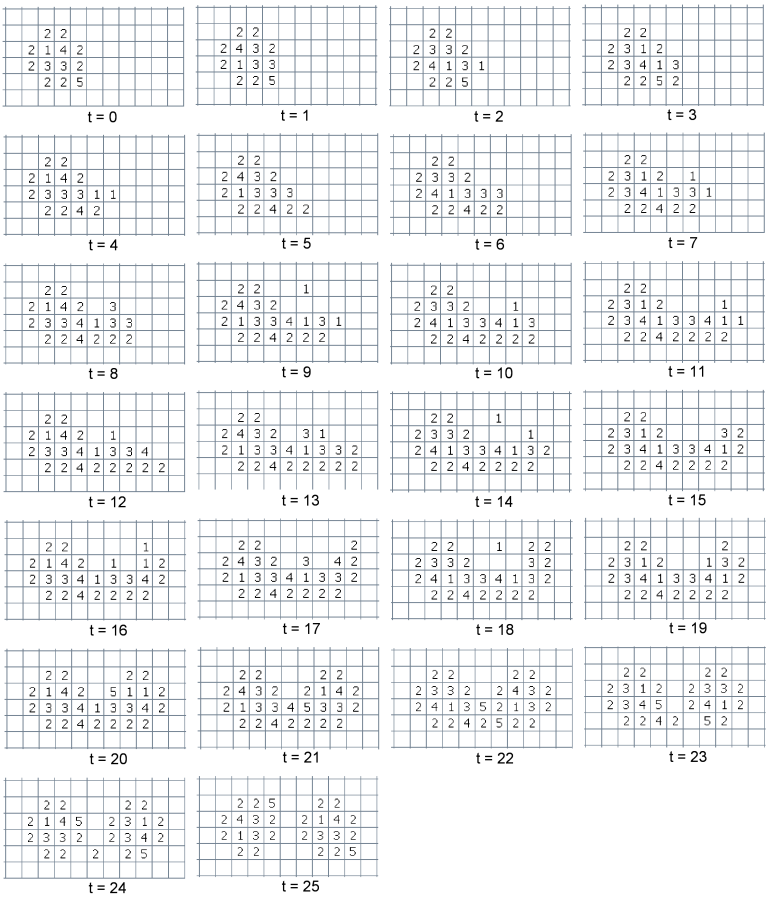

FIGURE 10
Byl's loop and its self-replication in 25 steps

5.3 Dissolvable Byl's Loops

With inspiration in SDSR (Structurally Dissolvable Self-Reproducing) loops which have a limited lifetime and dissolve at the end of their life cycle, we created another modification of accelerated Byl's loop. New rules were added with the following purpose: Once a loop gets frozen, its cells have to subsequently converge to a new dissolving state (denoted as 6) and then to a quiescent state (denoted as 0). The process takes 16 steps (see Figure 15) and

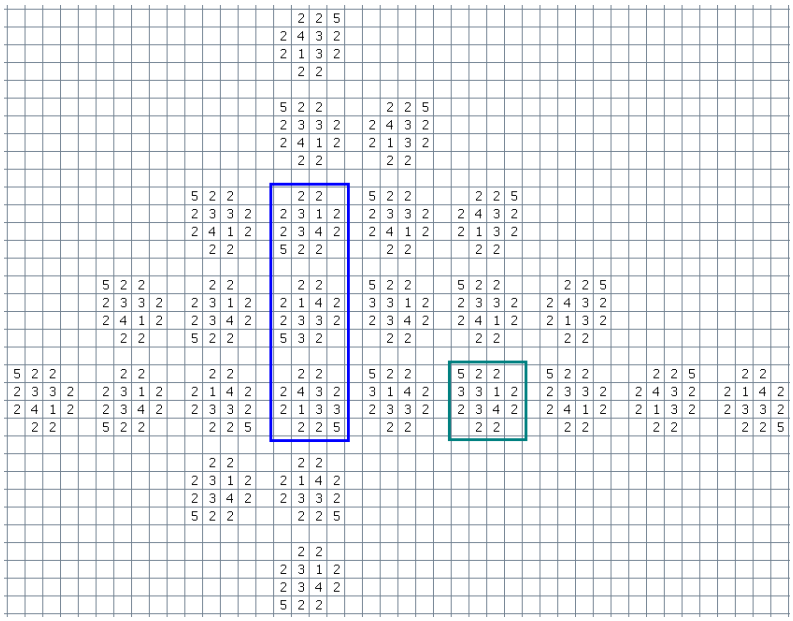

FIGURE 11
A colony of Byl's loops: Labeled triplet of cells does not have space to replicate. Labeled single cell can still be replicated.

its purpose is to free a space to vital loops. Figure 16 shows some dissolving loops and the first dissolved loop in step 78. Figure 17 shows the population of loops in step 260. We can observe that the loops form regular waves. There is a 'trash' in the middle of the colony as some loops have not finished their replication because of collisions with other loops.

6 DISCUSSION

There are numerous engineered as well as biological systems that are composed of many components which create locally interconnected patterns. The concept of locality allows creating massively parallel systems that can exhibit emergent behaviors. The components can be relatively simple and process a relatively small amount of information. However, as there is not a direct access to inner components from outside it can be difficult to set some component(s) to a particular state. The existence of this kind of globally working

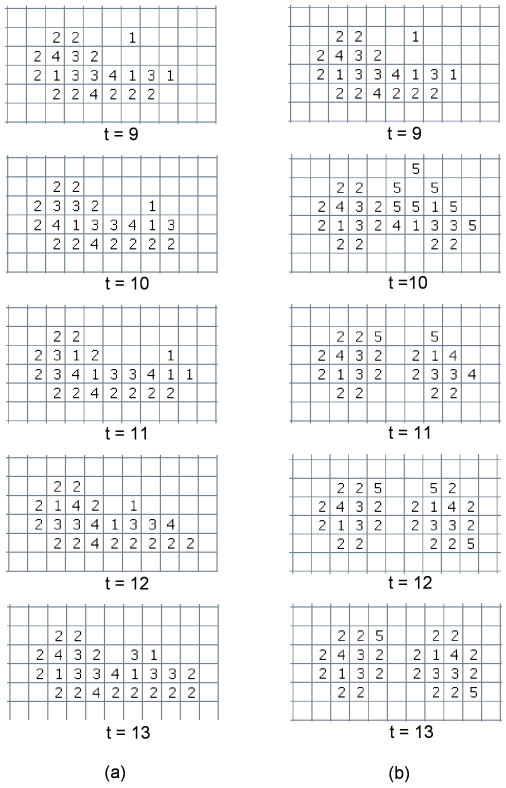

FIGURE 12
Steps 9 to 13 of the original Byl's loop (a) and its accelerated replication (b)

operation can be very useful, for example, when one has to set some components to a pre-specified state at the beginning of computation or configure some of the components.

For example, in biology, cells benefit from this type of global information in the process of differentiation. One of solutions existing in the embryo is to distribute morphogen into the system and let each cell decide its function on the basis of morphogen concentration in the place where the cell is located [29]. The morphogen can be seen as the global positional information which informs each cell what genes have to be synthesized, i.e. how to specialize. The positional information can be used to generate patterns in a growing embryo.

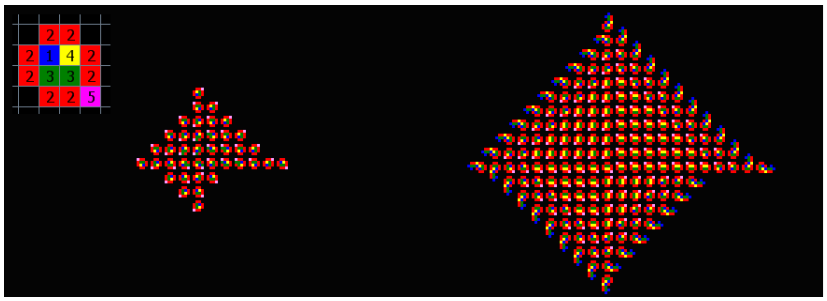

FIGURE 13
Comparison of colony of original (left) and accelerated (right) Byl's loops after 150 steps (the state 0 is shown in black)

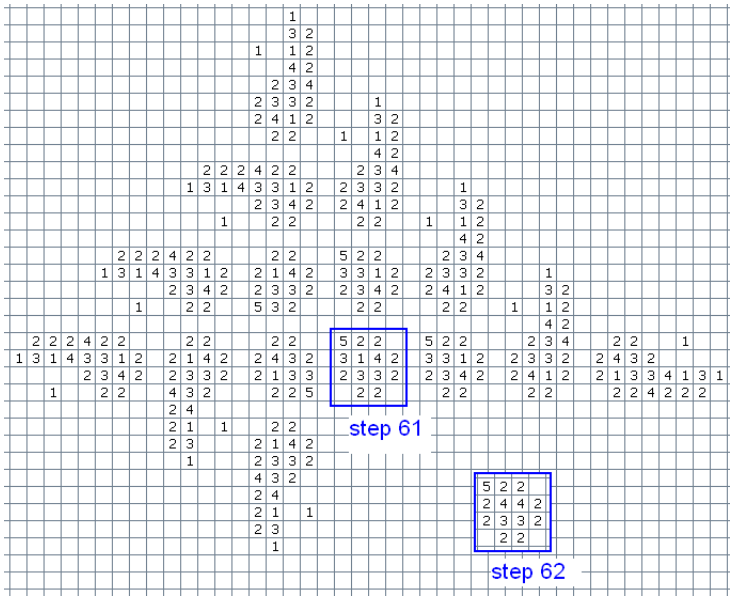

FIGURE 14
Labeled cell (step 61) has performed 5 unsuccessful attempts to replicate and will get frozen in step 62

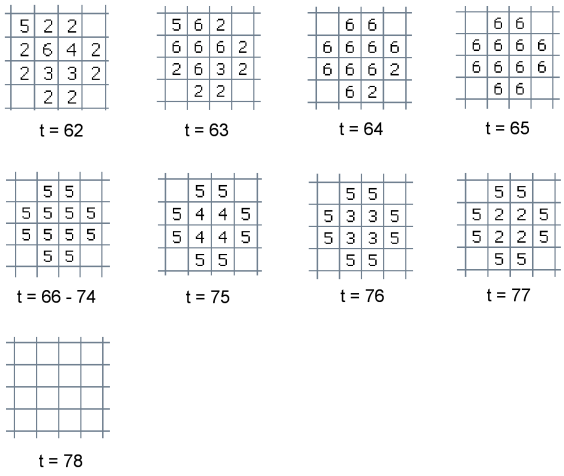

FIGURE 15
The process of loop dissolving

We have shown that benefits coming with the global control introduced to cellular automata are considerable and can be obtained for a relatively low cost in some cases. We consider the proposed concept as important mainly for physical implementations of very large scale cellular automata.

The concept of polymorphic components (polymorphic gates in our case) seems to be promising for practical applications. One can imagine a very large cellular automaton implemented using advanced nanotechnology in which elementary cells can be easily reconfigured using light, electromagnetic field, temperature or some other phenomena. Then, the possibility of the global control is actually for free. Nowadays, it is possible to implement in hardware, for example, a cellular automaton-based pseudorandom number generator with a programmable distribution controlled by means of the global control signal such as V_{dd} level.

In general, the problem with the global control is twofold: (i) It is necessary to know when and how the global control has to be activated. (ii) In order to recognize the global signal, the cells have to be equipped with additional capabilities (e.g., additional hardware). The designer's task is to find a suitable trade off between the benefits coming with the global control and the implementation overhead. Another issue is related to the synchronization of the cellular automaton. We have assumed that the computations are perfectly

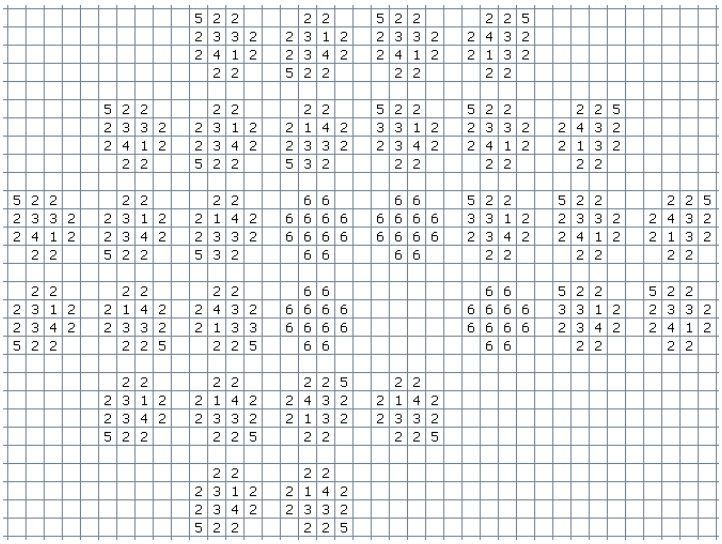

FIGURE 16
Several dissolving loops and one dissolved loop in a colony of modified Byl's loops in step 78

synchronized. The proposed approach will fail in the case of Byl's loops if all the loops are not in the same phase. We plan to study the global control of asynchronous cellular automata in our future work.

7 CONCLUSIONS

We have demonstrated that the global control signal can be useful for computations with cellular automata. In our very simplified case we, in fact, used a two-value discrete global signal which allowed the cells to change the local transition function. On the basis of the signal we could generate new patterns in a simple 1D cellular automaton and accelerate self-replication of Byl's loop in a 2D cellular automaton. In the case of 1D automaton we demonstrated that the overhead introduced with the global control can be relatively small if the implementation is performed using polymorphic gates controlled by the level of power supply voltage.

Future work will be devoted to searching for other applications of proposed concept. Another research could be devoted to analyzing properties of

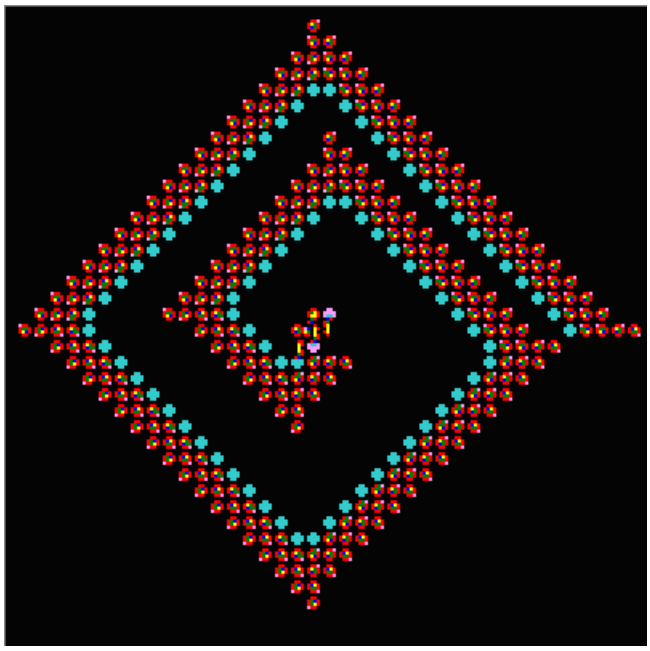

FIGURE 17
Dissolving and dissolved loops in colony of modified Byl's loops in step 260 (state 6 is shown in cyan)

globally controlled cellular automata using techniques developed for standard cellular automata.

8 ACKNOWLEDGMENTS

This work was partially supported by the grant Natural Computing on Unconventional Platforms GP103/10/1517, the BUT FIT grant FIT-S-10-1 and the research plan Security-Oriented Research in Information Technology, MSM 0021630528.

REFERENCES

[1] H. Bersini and V. Detours. (1994). Asynchrony induces stability in cellular automata based models. In *Proceedings of the IVth Conference on Artificial Life*, volume 204, pages 70–82. MIT Press.

[2] J. Byl. (1989). Self-reproduction in small cellular automata. *Physica D*, 34:295–299.

[3] S. J. Chandler, (2009). Cellular Automata with Global Control – from The Wolfram Demonstrations Project. URL: `http://demonstrations.wolfram.com/CellularAutomataWithGlobalControl`.

[4] E. Codd. (1968). *Cellular Automata*. Academic Press.

[5] L. Durbeck and N. Macias. (2001). The cell matrix: An architecture for nanocomputing. *Nanotechnology*, 12(3):217–230.

[6] M. Garzon. (1995). *Models of Massive Parallelism – Analysis of Cellular Automata and Neural Networks*. Springer, Berlin.

[7] C. G. Langton. (1984). Self-reproduction in cellular automata. *Physica D*, 10:135–144.

[8] J. D. Lohn and J. A. Reggia. (1997). Automatic discovery of self-replicating structures in cellular automata. *IEEE Transactions on Evolutionary Computation*, 1(3):165–178.

[9] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. (2000). Towards Robust Integrated Circuits: The Embryonics Approach. *Proceedings of IEEE*, 88(4):516–541.

[10] N. Oros and C. L. Nehaniv. (2007). Sexyloop: Self-reproduction, evolution and sex in cellular automata. In *The First IEEE Symposium on Artificial Life*, pages 130–138. IEEE.

[11] J.-Y. Perrier, M. Sipper, and J. Zahnd. (1996). Toward a viable, self-reproducing universal computer. *Physica D*, 97:335–352.

[12] J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng. (1993). Simple systems that exhibit self-directed replication. *Science*, 259:1282–1287.

[13] R. Ruzicka, L. Sekanina, and R. Prokop. (2008). Physical demonstration of polymorphic self-checking circuits. In *Proc. of 14th IEEE International On-Line Testing Symposium*, pages 31–36. IEEE.

[14] W. Li Santa and W. Li. (1992). Phenomenology of non-local cellular automata. In *Lectures in Complex Systems*, pages 317–327. Addison-Wesley.

[15] H. Sayama. (1998). *Constructing Evolutionary Systems on a Simple Deterministic Cellular Automata Space. PhD Thesis*. University of Tokio.

[16] L. Sekanina, R. Ruzicka, Z. Vasicek, R. Prokop, and L. Fujcik. (2009). Repomo32 – new reconfigurable polymorphic integrated circuit for adaptive hardware. In *2009 IEEE Workshop on Evolvable and Adaptive Hardware*, pages 39–46. IEEE Computational Intelligence Society.

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

[18] B. Shackleford, M. Tanaka, R. J. Carter, and G. Snider. (2002). Fpga implementation of neighborhood-of-four cellular automata random number generators. In *FPGA '02: Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 106–112, New York, NY, USA. ACM.

[19] M. Sipper. (1997). *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer Verlag, Heidelberg.

[20] M. Sipper. (1999). The emergence of cellular computing. *IEEE Computer*, 32(7):18–26.

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

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

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

[24] G. Tempesti. (1995). New self-reproducing cellular automaton capable of construction and computation. In *Proc. 3rd European Conference on Artificial Life*, volume 929 of *LNCS*, pages 555–563. Springer.

[25] T. Toffoli and N. Margolus. (1987). *Cellular Automata Machines: A new environment for modeling*. MIT Press, Cambridge MA.

[26] J. von Neumann. (1966). *Theory of Self-Reproducing Automata*. University of Illinios Press.

[27] S. Wolfram. (1994). *Cellular Automata and Complexity – Collected Papers*. Addison–Wesley.

[28] S. Wolfram. (2002). *A New Kind of Science*. Wolfram Media Inc.

[29] L. Wolpert. (2007). *Principles of Development, 3rd ed.* Oxford University Press, Oxford, UK.

[30] R. S. Zebulum and A. Stoica. (2006). Four-Function Logic Gate Controlled by Analog Voltage. *NASA Tech Briefs*, 30(3):8.

[31] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson. (2005). Self-reproducing machines. *Nature*, 435:163–164.

9 APPENDIX: RULES OF BYL'S LOOP

The original set of rules which is needed to replicate Byl's loop contains 238 rules (including rotations to ensure the replication in all directions). The second set of rules used to accelerate the self-replication in steps 10 - 13 contains 277 rules. Table 2 and Table 3 present 317 rules - the union of the two sets of rules. There are 190 rules that are identical for both sets.

TABLE 2
The union of the original rules and the rules for acceleration of Byl's loop self-replication (Part I). The individual rules are in the form CTRBLX-Y where CTRBL specifies the states of the Center, Top, Right, Bottom, and Left positions of the neighborhood's present state, X represents the next state for the first set of rules, '-' is separator, and Y represents the next state for the second set of rules. Symbol '!' just indicates that X differs from Y.

000010-5!	031001-1	130123-3	242042-0!	321331-3!	420234-0!	502402-2
000031-1	040022-5!	130451-2!	244202-0!	321511-1	420250-0	503005-4!
000100-5!	040040-2!	132244-1!	251005-5	321535-5	421433-3	503105-4!
000135-5	040050-2!	132404-1!	251205-5	322111-1	422313-4!	504202-2
000212-2	042000-5!	132414-4	252025-5	322131-3!	423003-3	504422-2
000233-5!	044000-2!	132434-4	253203-2!	323411-1	423013-4!	505005-0!
000242-5!	045000-2!	133000-0	300010-0	324433-2!	423055-5	510025-2!
000301-1	050040-2!	133244-4	300030-0	325131-1	423113-3	510035-4!
000311-1	051002-2	134424-4	300100-0	325415-5	423123-4!	511505-2!
000420-5!	052005-0!	135423-3	300110-0	330000-0	423153-3	512024-4
000440-2!	052200-5!	140124-4	300211-1	332101-1	423204-0!	513002-2
000450-2!	054000-2!	140324-1!	300300-0	332111-3!	425104-0!	515015-2!
000512-2	100000-0	141124-4	301000-0	332131-3!	425200-0	520042-2
000525-0!	100010-0	141224-4	301100-0	332155-5	425313-3	520214-4
000540-2!	100033-3	141324-4	301211-3!	332211-3!	430004-2!	520220-0
001000-5!	100100-0	142344-4	302101-1	332443-2!	430023-3	520442-2
001010-5!	100303-3	142353-3	303000-0	332511-1	430123-4!	521005-2!
001305-5	100330-0	143224-1!	303211-1	333211-3!	430525-5	521204-4
002102-2	101000-0	143324-4	305122-2	334121-1	431123-3	522020-0
002303-5!	101233-3	144234-4	310000-0	341231-1	431223-4!	522200-0
002402-5!	101244-4	145301-2!	310010-0	341255-5	431253-3	522300-0
003001-1	103003-3	153041-2!	310021-1	343243-2!	431523-3	523020-0
003101-1	103244-1!	154233-3	310121-3!	344323-2!	432024-0!	523242-2
004200-5!	103300-0	200000-0	310321-1	351202-2	432143-3	524002-2
004400-2!	104531-2!	200220-0	311000-0	351211-1	443213-3	524232-2
004500-2!	110000-0	200515-5	311221-1	351321-1	451024-0!	525545-0!
005102-2	111244-4	202122-0!	311321-3!	353215-5	452020-0	530005-4!
005205-0!	112244-4	202200-0	312052-2	354125-5	452305-5	530012-2
005220-5!	112303-3	202525-5	312101-3!	400034-2!	452313-3	530220-0
005400-2!	112404-4	204422-0!	312151-1	400233-3	453123-3	531005-4!
010000-5!	112414-4	205105-5	312211-1	400304-2!	500005-0!	532422-2
010022-2	113244-4	205125-5	312341-1	401233-4!	500035-4!	540022-2

TABLE 3
The union of the original rules and the rules for acceleration of Byl's loop self-replication (Part II).

010031-1	122414-4	205323-2!	312545-5	402303-3	500055-0!	542002-2
010052-2	122434-1!	210055-5	313211-3!	402324-0!	500132-2	542042-2
010100-5!	123013-3	212022-0!	313221-3!	402514-0!	500215-2!	542322-2
013005-5	123444-4	212055-5	313251-1	402520-0	500242-2	542555-0!
020040-5!	123543-3	220020-0	313321-3!	403004-2!	500305-4!	544202-2
020055-0!	124014-4	220212-0!	315121-1	405235-5	500315-4!	550005-0!
020520-5!	124034-1!	220255-5	315325-5	410254-0!	500422-2	550115-2!
021002-2	124114-4	220442-0!	320512-2	411233-3	500505-0!	554255-0!
022050-5!	124124-4	220515-5	321001-1	412233-4!	501155-2!	555425-0!
023003-5!	124134-4	220533-2!	321011-3!	412303-4!	501302-2	
024002-5!	124324-1!	221202-0!	321031-1	412313-3	502105-2!	
030001-1	124334-4	222000-0	321121-1	412533-3	502124-4	
030015-5	130003-3	225205-5	321131-3!	414323-3	502220-0	
030023-5!	130030-0	232053-2!	321321-3!	415233-3	502230-0	