Majority adder implementation by competing patterns in Life-like rule $B2/S2345$

Genaro J. Martínez[1,3], Kenichi Morita[2], Andrew Adamatzky[3], and Maurice Margenstern[4]

[1] Instituto de Ciencias Nucleares and Centro de Ciencias de la Complejidad, Universidad Nacional Autónoma de México, México DF.
genaro.martinez@uwe.ac.uk
[2] Hiroshima University, Higashi-Hiroshima 739-8527, Japan.
morita@iec.hiroshima-u.ac.jp
[3] Bristol Institute of Technology, University of the West of England, Bristol, United Kingdom. andrew.adamatzky@uwe.ac.uk
[4] Laboratoire d'Informatique Théorique et Appliquée, Université de Metz, Metz Cedex, France.
margens@univ-metz.fr

Abstract. We study Life-like cellular automaton rule $B2/S2345$. This automaton exhibits a chaotic behavior yet capable for purposeful computation. The automaton implements Boolean gates via patterns which compete for the space when propagate in channels. Values of Boolean variables are encoded into two types of patterns — symmetric (False) and asymmetric (True). We construct basic logical gates and elementary arithmetical circuits by simulating logical signals using glider reactions taking place in the channels built of non-destructible still lifes. We design a binary adder of majority gates realised in rule $B2/S2345$.

1 Introduction

There is a plenty of computing devices 'made of' Conway's Game of Life (GoL) cellular automaton [13]. Examples include a complete set of logical functions [32], register machine [8], direct simulation of Turing machine [9, 31], and design of a universal constructor [16]. These implementations use principles of collision-based computing [8, 1] where information is transferred by gliders propagating in an architecture-less medium. Theoretical result regarding GoL universality is only a tiny step in a long journey towards real-world implementation of the collision-based computers [33].

GoL has a long history where a number of dedicated researchers obtained significant results on its complex dynamics and computing devices. The first one was published by Gardner [13] followed for a newsletter edited by Wainwright [35, 3]. A number of results in GoL is published, some of them really complicated as for example universal computers/constructors [1, 6, 8, 9, 12, 14–16, 26, 30, 31].

On the way, we have reported in phenomenological studies of semi-totalistic CA [5], a selected set rules named as *Life* $2c22$, identified by periodic struc-

tures [28]. The clan closest to the family $2c22$ and the *Diffusion Rule* (Life rule $B2/S7$) [21], all they also into of a big cluster named as *Life* $dc22$.[5]

In this paper we will exploit previous results on the constructions of feedback channels with still life patterns (previous studies since $B2/S23456$ [20] and $B2/S2345678$ [22]), reducing the number of cells in state 1 on the evolution rule. Hence every pattern propagation is stimulated since a glider reaction that will produce a specific static geometric pattern, thus their interactions when they compete shall yield a binary value representation. Finally we design specific initial configurations to get implementations of universal logic gates and a binary adder based on majority gates inside $B2/S2345$.

2 Life rule $B2/S2345$

Dynamics of Life rule $B2/S2345$ is described for the next conditions. Each cell takes two states '0' ('dead') and '1' ('alive'), and updates its state depending on its eight closest neighbours (Moore neighborhood):

a) Birth: a central cell in state 0 at time step t takes state 1 at time step $t+1$ if it has exactly two neighbours in state 1.
b) Survival: a central cell in state 1 at time t remains in the state 1 at time $t+1$ if it has two, three, four or five live neighbours.
c) Death: all other local situations.

Once a resting lattice is perturbed in $B2/S2345$ (few cells are assigned live states), patterns of states 1 emerge, grow and propagate on the lattice quickly. The main characteristic is that gliders and oscillators emerge but they do not survive for long time.

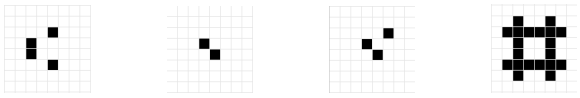

Fig. 1. Basic periodic structures in $B2/S2345$: (a) glider, (b) oscillator (flip-flop), (c) oscillator (blinker), and (d) still life configuration.

A set of minimal particles, or basic periodic structures, in rule $B2/S2345$ include one glider (period one), two oscillators (one blinker and one flip-flop, period two), and finally one still life configuration (see Fig. 1). The still life pattern [23, 11] in $B2/S2345$ has a relevant characteristic. They are not affected by their environment however they do affect their environment [20, 22]). Therefore the still life patterns can be used to build channels, or wires, for signal propagation.

[5] `http://uncomp.uwe.ac.uk/genaro/Life_dc22.html`

2.1 Indestructible still life pattern in $B2/S2345$

Some patterns amongst still life patterns in the rule $B2/S2345$ belong to a class of *indestructible patterns* (sometimes referred to as 'glider-proof' patterns in GoL) which cannot be destroyed by any perturbation, including collisions with gliders. A minimal indestructible pattern, still life occupying a square of 6×6 cells, is shown in Fig. 1d.

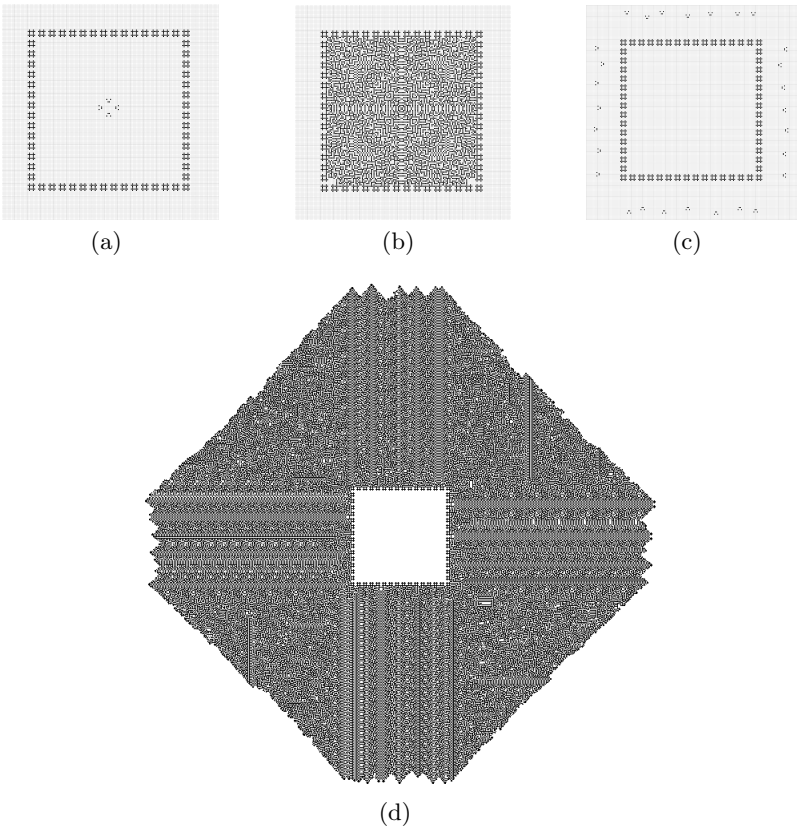

Fig. 2. Containment of growing pattern by indestructible patterns. (a) First example display an explosion reaction started from a collision between four gliders (see center), (b) display the final configuration stopping this growing pattern. (c) Display initial positions of a fleet of gliders outside the box walled by still life configurations in our second example, (d) show how interior of the box is protected from the growing pattern.

The indestructible patterns can be used to stop 'supernova' explosions in some Life-like rules. Usually a Life-like automaton development started at an arbitrary configuration exhibits unlimited growth (generally related to some kind

of nucleation phenomenon [17]). Hence a suitable concatenation of still life configurations avoid a continuos expansion.

In rule $B2/S2345$ such an 'uncontrollable' growth can be prevented by a regular arrangement of indestructible patterns. Examples are shown in Fig. 2. In the first example (Fig. 2a) four gliders collide inside a 'box' made of still life patterns. The collision between the gliders lead to formation of growing pattern. The propagation of the pattern is stopped by the indestructible wall (Fig. 2b). In the second example, a fleet of gliders collide outside the box (Fig. 2c) however interior of the box remains resting (Fig. 2c) due to impenetrable walls. Similarly, one can construct a colony of still life patterns immune to local perturbations.

Thus the indestructibility exemplified above allows us to use still life patterns to construct channel information in logical circuits.

3 Computing by competing patterns

The easiest way to control patterns propagating in a non-linear medium circuits is to constrain them geometrically. Constraining the media geometrically is a common technique used when designing computational schemes in spatially extended non-linear media. For example 'strips' or 'channels' are constructed within the medium (e.g. excitable medium) and connected together, typically using arrangements such as T-junctions. Fronts of propagating phase (excitation) or diffusive waves represent signals, or values of logical variables. When fronts interact at the junctions some fronts annihilate or new fronts emerge. The propagation in the output channels represent results of the computation.

Hence we built a computing scheme from channels — long areas of '0'-state cells walled by still life blocks, and T-junctions[6] — sites where two or more channels join together.

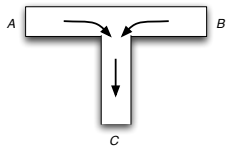

Fig. 3. T-shaped system processing information.

Each T-junction consists of two horizontal channels A and B (shoulders), acting as inputs, and a vertical channel, C, assigned as an output (Fig. 3). Such type of circuitry has been already used to implement XOR gate in chemical laboratory precipitating reaction-diffusion systems [4], and precipitating logical gates imitated in CA [20, 22]. A minimal width of each channel equals three widths of the still life block (Fig. 1d) and width of a glider (Fig. 1a).

[6] T-junction based control signals were suggested also in von Neumann [34] works, and used by Banks [7] and Codd [10] as well.

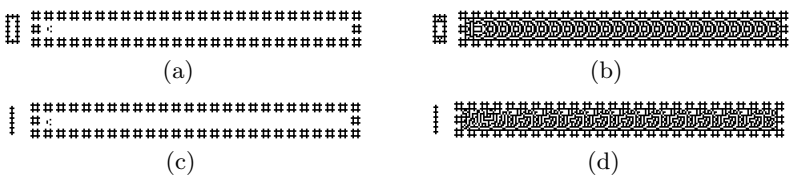

Fig. 4. Feedback channels constructed with still life patterns ((a) and (c)) show the initial state with the empty channel and one glider respectively. The symmetric pattern represent value 0 (b), and non-symmetric pattern represent value 1 (d) late of glider reaction.

Boolean values are represented by position of gliders, positioned initially in the middle of channel, value 0 (Fig. 4a), or slightly offset, value 1 (Fig. 4c). The initial positions of the gliders determine outcomes of their reaction. Glider, corresponding to the value 0 is transformed to a regular symmetric pattern, similar to frozen waves of excitation activity (Fig. 4b). Glider, representing signal value 1, is transformed to transversally asymmetric patterns (Fig. 4d). Both patterns propagate inside the channel with constant, advancing unit of channel length per step of discrete time.

3.1 Implementation of logic gates

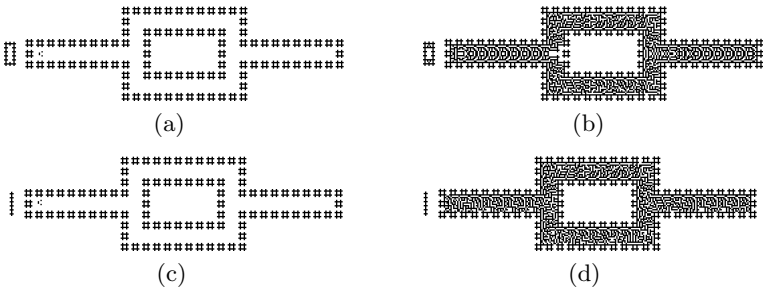

Fig. 5. Configurations of delay element for signal '0' (a) and (b), and signal '1' (c) and (d). Thus (a) and (c) shows initial configurations, (b) and (d) final states.

Our first stage is implement basic universal logic gates. When patterns, representing values 0 and 1, meet at T-junctions they compete for the output channel. So depending on initial distance between gliders, one of the patterns wins and propagates along the output channel.

On the way we can design a DELAY element as shown in Fig. 5. Useful to delay signals (wave propagations) and synchronize multiple collisions.

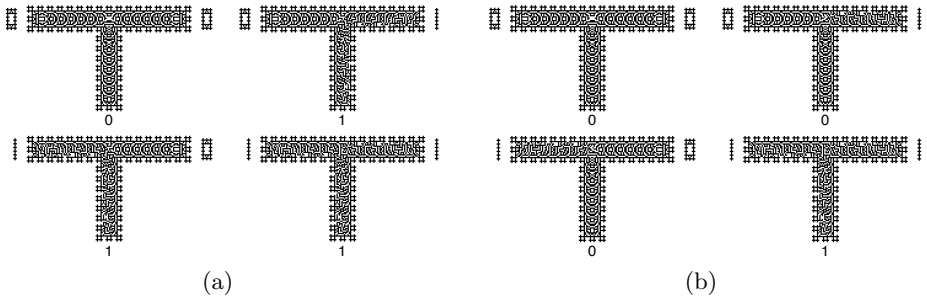

Fig. 6. Implementations of OR and AND gates at the Life rule $B2/S2345$. Input binary values A and B they are represented as 'In/0' or 'In/1,' output result C is represented by 'Out/0' or 'Out/1.' Thus (a) display OR gate, and (b) AND gate implementation.

Figure 6a shows a way to implement an OR gate. Due to different locations of gliders in initial configurations of gates, patterns in both implementations of gates are different however, results of computation are the same. Similarity a codification to implement AND gate is shown in Fig. 6b.

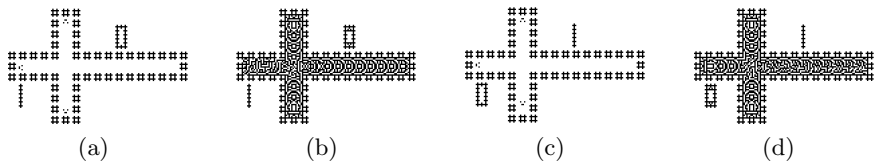

Fig. 7. NOT gate implementation for input '1' (a,b) and input '0' (c,d). This way (a) and (c) display initial configurations, (b) and (d) display final configurations.

The NOT gate is implemented using additional channel (as a trick), where control pattern is generated, propagate and interfere with data-signal pattern. Initial and final configurations of NOT gate are shown in Fig. 7. A consequence with this idea is that the number of control channels growth proportionally to number of gates in the circuit. Of course, we accept that it could not be the most elegant and efficient way of constructing NOT gate, but useful for our purposes at the moment.

3.2 Majority gate

MAJORITY gate implementation on three input values can be represented as a logical proposition: $(a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$, where the result is precisely the most frequently value on such variables [24].

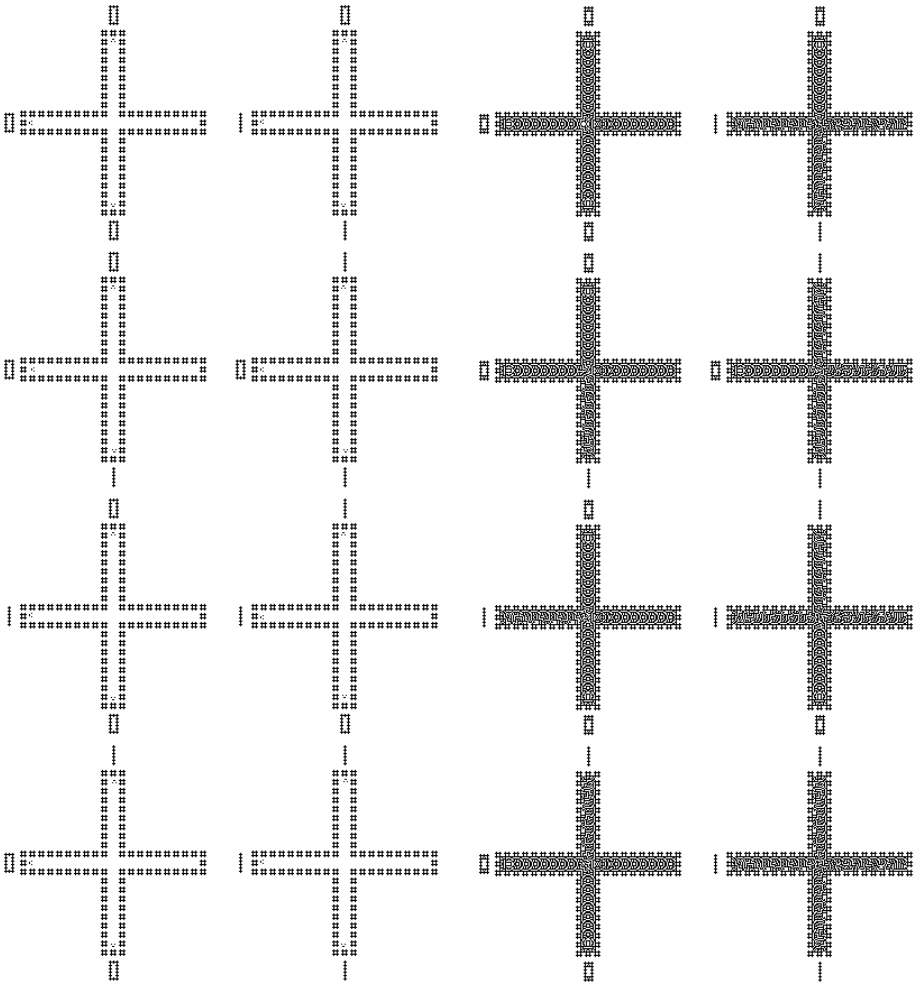

Fig. 8. (a) initial configuration: majority input values In/0 (first column), and majority input values In/1 (second column), and (b) final configurations of the majority gates.

Implementation of MAJORITY gate in $B2/S2345$ is shown in Fig. 8. The gate has three inputs: North, West and South channels, and one output: East channel. Three propagating pattern, which represent inputs, collide at the cross-junction of the gate. The resultant pattern is recorded at the output channel.

3.3 Implementation of binary adder with NOT-MAJORITY gates

Here we will implement a binary adder constructed of three NOT-MAJORITY gates and two inverters. Such type of adder appears in several publications, particularly in construction of the arithmetical circuits in quantum-dot cellular automata [29, 36]. Original version of the adder using NOT-MAJORITY gates was suggested by Minsky in his designs of artificial neural networks [24].

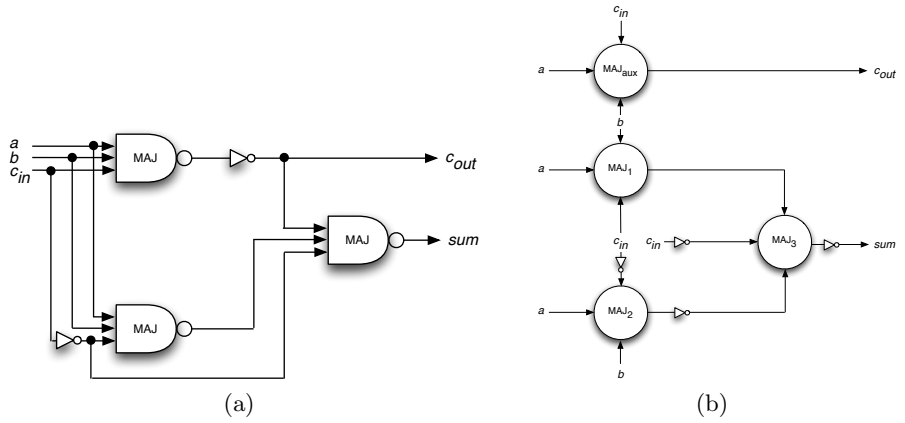

Fig. 9. Circuit and schematic diagram of a full binary adder comprised of NOT-MAJORITY gates. Delay elements are not shown.

Figure 9a shows the classic circuit illustrating the dynamics of this adder. This way Fig. 9b represents a scheme of the adder to implement in $B2/S2345$. The scheme highlights critical points where some extra gates/wires are necessary to adjust inputs and synchronize times of collisions.

Figure 10a presents most important stages of the full adder on $B2/S2345$ evolution space standing out DELAYS stages and NOT gates. The adder is implemented on $1,402 \times 662$ lattice that relates an square of 928,124 cells lattice with an initial population of 56,759 cells in state '1.' Final configurations of the adder for every initial configuration of inputs are shown in Figs. 10b–i with a final population of 129,923 alive cells on an average of 1,439 generations.[7]

[7] To look enlarge pictures or videos of the simulations please visit `http://uncomp.uwe.ac.uk/genaro/Life_dc22.html`

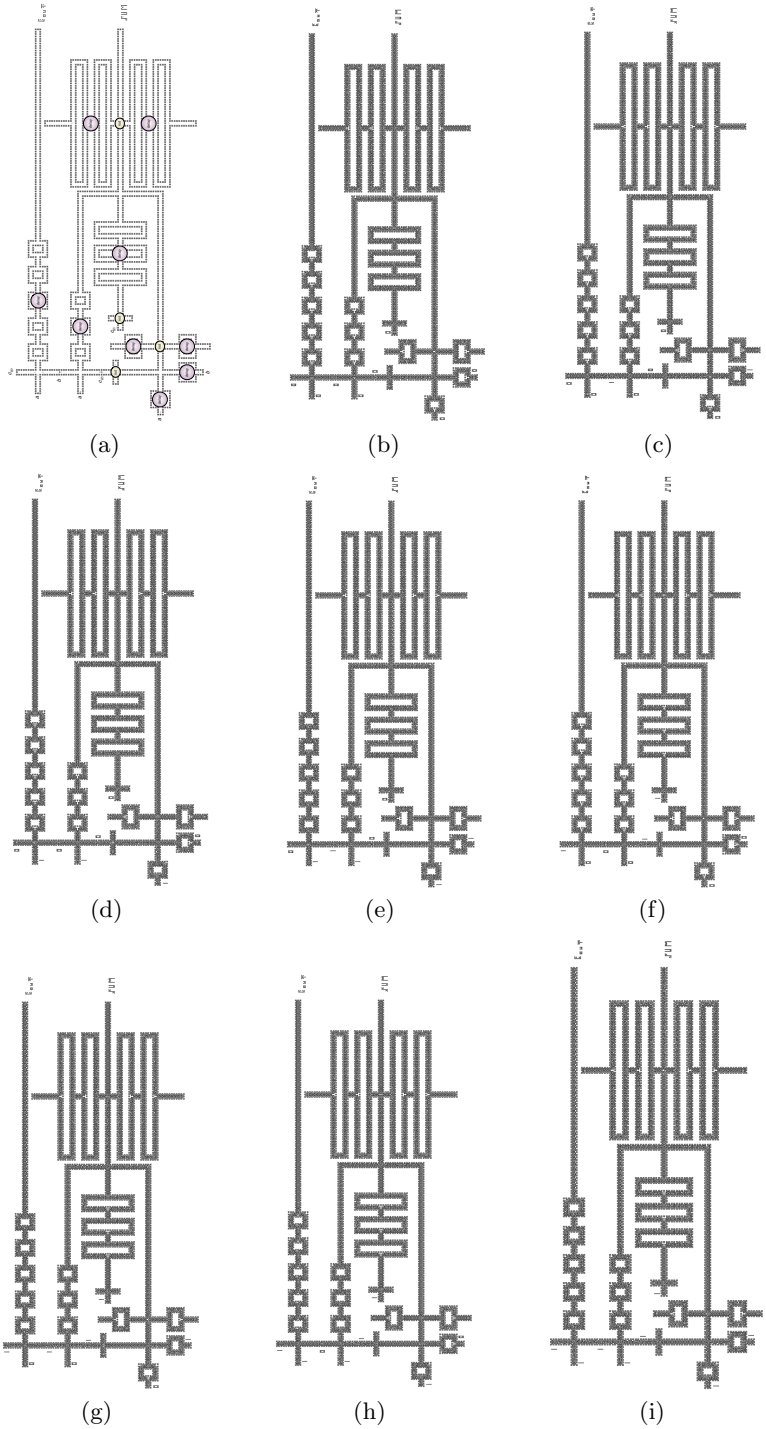

Fig. 10. Final configuration of the adder for inputs: (a) main stages; (b) $a = 0$, $b = 0$, $c_{in} = 0$; (c) $a = 0$, $b = 1$, $c_{in} = 0$; (d) $a = 1$, $b = 0$, $c_{in} = 0$; (e) $a = 1$, $b = 1$, $c_{in} = 0$; (f) $a = 0$, $b = 0$, $c_{in} = 1$; (g) $a = 0$, $b = 1$, $c_{in} = 1$; (h) $a = 1$, $b = 0$, $c_{in} = 1$; (i) $a = 1$, $b = 1$, $c_{in} = 1$.

4 Conclusions

We have demonstrated that chaotic rule $B2/S2345$ supports complex patterns. That relates another case where a chaotic CA contains non evident complex behaviour [21, 25], and how such systems could have some computing on its evolution space from particular initial conditions.

We have shown how construct basic logical gates and arithmetical circuits by restricting propagation of patterns in channels, constructed indestructible still life blocks.

However we have recognized a number of limitations on this model. Disadvantage of the approach presented is that computing space is geometrically constrained and the computation is one-time-use. Also actually we do not have a way to develop a crossing signal and FANOUT gate that are essential to complete a feedback full circuit operation.

Nevertheless the geometrical constraining brings some benefits as well. Most computing circuits in Life-like automata are using very complex dynamics collisions between gliders and still life [9, 31, 16]. In this case gliders are used only to 'ignite' propagating patterns in the channels [4, 35].

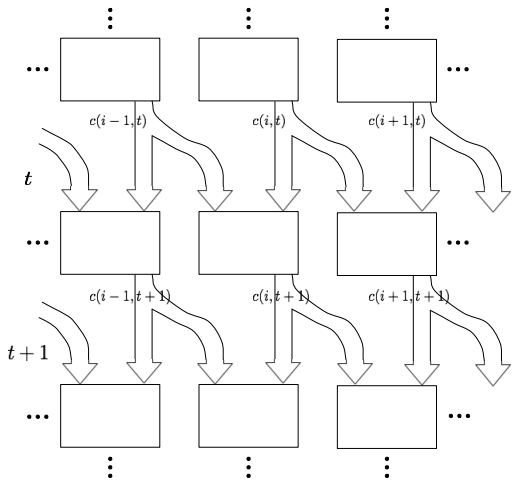

Fig. 11. Simulating a radius $\frac{1}{2}$ CA by an infinite (but periodic) cascade circuit.

Let us check how to exploit computations on one-time-use. We could employ the known *cascade circuits* concept (see Fig. 11). The cascade circuit is a one without feedback where each box contains a logic circuit that realizes a local function of the CA, which is also a cascade one. Since it has a no feedback because each logic gate is used only once. This way, an initial state of each cell should be set at the position of $t = 0$.

These results are potentially useful in the search of control of big volume on data in non-linear medium. Life-like rule $B2/S2345$ in this case is only an example of how is possible controller data as a single bit value in a deterministic version.

Also we will explore and develop more complex flows of data as were done in reversible devices [19, 27]. You can see the problem to control chemical wave propagation in reaction-diffusion computers [4] hence a competing pattern can represent a fragment of each wave.

In future studies we are planning to implement the computing architecture designed in the paper to manufacture experimental prototypes of precipitating chemical computers; they will be based on crystallization of 'hot ice' [2].

Implementations and constructions are done with *Golly* system.[8] Source configurations and specific initial condition (RLE files) to reproduce these results are available in 'Life *dc*22' home page.[9]

Acknowledgement

Genaro J. Martínez was partially funded by Engineering and Physical Sciences Research Council (EPSRC), United Kingdom, grant EP/F054343 and DGAPA UNAM, Mexico. Kenichi Morita was partially funded by Grant-in-Aid for Scientific Research (C) No. 21500015 from JSPS.

References

1. A. Adamatzky (Ed.): *Collision-Based Computing*, Springer (2002).
2. A. Adamatzky: Hot ice computer, *Physics Letters A* 374(2), 264–271 (2009).
3. A. Adamatzky (Ed.): *Game of Life Cellular Automata*, Springer (2010).
4. A. Adamatzky, B. L. Costello, T. Asai: *Reaction-Diffusion Computers*, Elsevier (2005).
5. A. Adamatzky, G. J. Martínez, J. C. Seck-Tuoh-Mora: Phenomenology of reaction-diffusion binary-state cellular automata, *Int. J. Bifurcation and Chaos* 16(10), 1–21 (2006).
6. S. Adachi, F. Peper, J. Lee, H. Umeo: Occurrence of gliders in an infinite class of Life-like cellular automata, *Lecture Notes in Computer Science* 5191, 32–41 (2008).
7. E. R. Banks: *Information Processing and Transmission in Cellular Automata*, Ph.D. thesis Department of Mechanical Engineering, MIT (1971).
8. E. R. Berlekamp, J. H. Conway, R. K. Guy: *Winning Ways for your Mathematical Plays*, Academic Press, (vol. 2, chapter 25) (1982).
9. P. Chapman: Life Universal Computer, `http://www.igblan.free-online.co.uk/igblan/ca/` (2002).
10. E. F. Codd: *Cellular Automata*, Academic Press (1968).
11. M. Cook: Still Life Theory, In [15], 93–118 (2003).
12. D. Eppstein: Growth and decay in Life-like cellular automata, arXiv:0911.2890v1 [nlin.CG], (2009).

[8] `http://golly.sourceforge.net/`
[9] `http://uncomp.uwe.ac.uk/genaro/Life_dc22.html`

13. M. Gardner: Mathematical Games — The fantastic combinations of John H. Conway's new solitaire game Life, *Scientific American* 223, 120–123 (1970).
14. D. Griffeath, C. Moore: Life Without Death is P-complete, *Complex Systems* 10, 437–447 (1996).
15. D. Griffeath, C. Moore (Eds.): *New constructions in cellular automata*, Oxford University Press (2003).
16. A. Goucher: Completed Universal Computer/Constructor (2009). In: `http://pentadecathlon.com/lifeNews/2009/08/post.html`.
17. J. Gravner: Growth Phenomena in Cellular Automata, In [15], 161–181 (2003).
18. S. R. Hameroff: *Ultimate Computing: Biomolecular Consciousness and Nanotechnology*, Elsevier Science Publishers BV (1987).
19. K. Imai, K. Morita: A computation-universal two-dimensional 8-state triangular reversible cellular automaton, *Theoret. Comput. Sci.* 231, 181–191 (2000).
20. G. J. Martínez, A. Adamatzky, B. L. Costello: On logical gates in precipitating medium: cellular automaton model, *Physics Letters A* 1(48), 1–5 (2008).
21. G. J. Martínez, A. Adamatzky, H. V. McIntosh: Localization dynamic in a binary two-dimensional cellular automaton: the Diffusion Rule, arXiv:0908.0828v1 [cs.FL], 2009.
22. G. J. Martínez, A. Adamatzky, H. V. McIntosh, B. L. Costello: Computation by competing patterns: Life rule $B2/S2345678$, In *Automata 2008: Theory and Applications of Cellular Automata*, Adamatzky, A. *et. al* (Eds.), Luniver Press (2008).
23. H. V. McIntosh: Life's Still Lifes, `http://delta.cs.cinvestav.mx/~mcintosh` (1988).
24. M. Minsky: *Computation: Finite and Infinite Machines*, Prentice Hall (1967).
25. M. Mitchell: Life and evolution in computers, *History and Philosophy of the Life Sciences* 23, 361–383 (2001).
26. M. Magnier, C. Lattaud, J.-K. Heudin: Complexity Classes in the Two-dimensional Life Cellular Automata Subspace, *Complex Systems* 11(6), 419–436 (1997).
27. K. Morita, M. Margenstern, K. Imai: Universality of reversible hexagonal cellular automata, *Theoret. Informatics Appl.* 33, 535–550 (1999).
28. G. J. Martínez, A. M. Méndez, M. M. Zambrano: Un subconjunto de autómata celular con comportamiento complejo en dos dimensiones, `http://uncomp.uwe.ac.uk/genaro/Papers/Papers_on_CA.html` (2005).
29. W. Porod, C. S. Lent, G. H. Bernstein, A. O. Orlov, I. Amlani, G. L. Snider, J. L. Merz: Quantum-dot cellular automata: computing with coupled quantum dots, *Int. J. Electronics* 86(5), 549–590 (1999).
30. N. Packard, S. Wolfram: Two-dimensional cellular automata, *J. Statistical Physics* 38, 901–946 (1985).
31. P. Rendell: Turing universality of the game of life, In [1], 513–540 (2002).
32. J. P. Rennard: Implementation of Logical Functions in the Game of Life, In [1], 491–512 (2002).
33. T. Toffoli: Non-Conventional Computers, *Encyclopedia of Electrical and Electronics Engineering* (John Webster Ed.) 14, 455–471, Wiley & Sons, (1998).
34. J. von Neumann: *Theory of Self-reproducing Automata* (edited and completed by A. W. Burks), University of Illinois Press, Urbana and London (1966).
35. R. Wainwright (ed.): Lifeline - A Quaterly Newsletter for Enthusiasts of John Conway's Game of Life, Issues 1 to 11, March 1971 to September 1973.
36. K. Walus, G. Schulhof, R. Zhang, W. Wang, G. A. Jullien: Circuit design based on majority gates for applications with quantum-dot cellular automata. In *Proceedings of IEEE Asilomar Conference on Signals, Systems, and Computers* (2004).