

Operating binary strings using gliders and eaters in reaction-diffusion cellular automaton

Andrew Adamatzky*, Genaro Martinez, Liang Zhang, Andrew Wuensche†

August 6, 2009

Center for Unconventional Computing and Department of Computer Science
University of the West of England, Bristol, United Kingdom

Abstract

We study transformations of 2-, 4- and 6-bit numbers in interactions between traveling and stationary localizations in the Spiral Rule reaction-diffusion cellular automaton. The Spiral Rule automaton is a hexagonal ternary-state two-dimensional cellular automaton – a finite-state machine imitation of an activator-inhibitor reaction-diffusion system. The activator is self-inhibited in certain concentrations. The inhibitor dissociates in the absence of the activator. The Spiral Rule cellular automaton has rich spatio-temporal dynamics of traveling (glider) and stationary (eater) patterns. When a glider brushes an eater the eater may slightly change its configuration, which is updated once more every next hit. We encode binary strings in the states of eaters and sequences of gliders. We study what types of binary compositions of binary strings are implementable by sequences of gliders brushing an eater. The models developed will be used in future laboratory designs of reaction-diffusion chemical computers.

Keywords: cellular automata, reaction-diffusion computing, gliders, collision-based computing

1 Introduction

Many physical, chemical and biological spatially extended non-linear systems exhibit a wide range of stationary and mobile localizations: solitons, kinks, breathers, excitons, defects and wave-fragments. The localizations can be used to transmit and transform information, and ultimately to perform computation [1]. A unit of information, such as the value of a Boolean variable, is decoded into presence (logical truth) or absence (logical false) of a localization in some specified site of space at a specified moment of time. When two localizations (representing the values of two logical variables) collide, they change their trajectories (or annihilate, reproduce, or change their shape). The

*Contact author: andrew.adamatzky@uwe.ac.uk

†www.ddlab.org

new trajectories of the localizations encode the values of some logical function over the two variables. This is how most collision-based computing devices work [7, 10, 13, 1, 8, 19, 20, 14].

The collision-based, or free-space, computing devices typically do not have wires and — in principle — are not supposed to use any other stationary components to perform computation. Any point of the computing media can act as a wire, a trajectory of a traveling localization can be seen as a momentary wire. Any site where two or more localizations collide is a logical gate. Thus space can be used efficiently and nothing is wasted. However, there is a price to pay. Initial positions and launch time of the traveling localizations should be precisely specified: one wrong time step destroys the whole computing scheme. The need for perfect timing is the weakest point of collision-based computing and the subject of constant criticism by “classical” computation schools.

Can we do without perfect timing? Asynchronous cellular-automaton based computers do pretty well [11, 15, 12] by using predetermined wires and valves. In the present paper we are trying to combine pure collision-based computing ideas (gliders only) with stationary architectures (breather-like localization) to implement computing schemes with relaxed timing. We choose the reaction-diffusion Spiral Rule cellular automaton [18, 4, 3, 16] as a testbed for ideas of asynchronous collision-based computing.

What is our rationale behind selecting the Spiral Rule automaton to study novel concepts of the collision-based computing? The Spiral Rule cellular automaton [18] plays a unique role in unconventional computing. On the one hand, this is a simple ternary state hexagonal automaton with Conway’s Game of Life type of behavior: it has gliders, still lives and eaters, and glider. Therefore it is very suitable for experimenting with collision-based computing schemes. On the other hand, the Spiral Rule automaton is a unique discrete model of a non-linear reaction-diffusion chemical system with an activator and inhibitor. The gliders and glider guns in the Spiral Rule automaton are analogues of excitation wave-fragments and generators of wave-fragments in a light-sensitive sub-excitable Belousov-Zhabotinsky medium [9]. This means that prototypes of computing schemes designed in the Spiral Rule automaton can then be almost straightforwardly implemented in chemical laboratory prototypes of reaction-diffusion computers.

The paper is structured as follows. The Spiral Rule reaction-diffusion cellular automaton is defined in Sect. 2. Encoding of two- and four-bit binary strings in states of an eater and transformation of the strings by gliders are presented in Sect. 3. Section 4 studies compositions of six-bit binary strings and speculates on the algebraic properties that the compositions impose on a set of six-bit digital numbers. We envisage future developments in Sect. 5.

2 Spiral rule reaction-diffusion cellular automaton

A detailed description, including the background and rationale, of the Spiral Rule reaction-diffusion cellular automaton are provided in [18, 4, 3, 16], therefore here we just give a self-consistent outline. The Spiral Rule automaton is a discrete approximation of an abstract reaction-diffusion chemical system. The

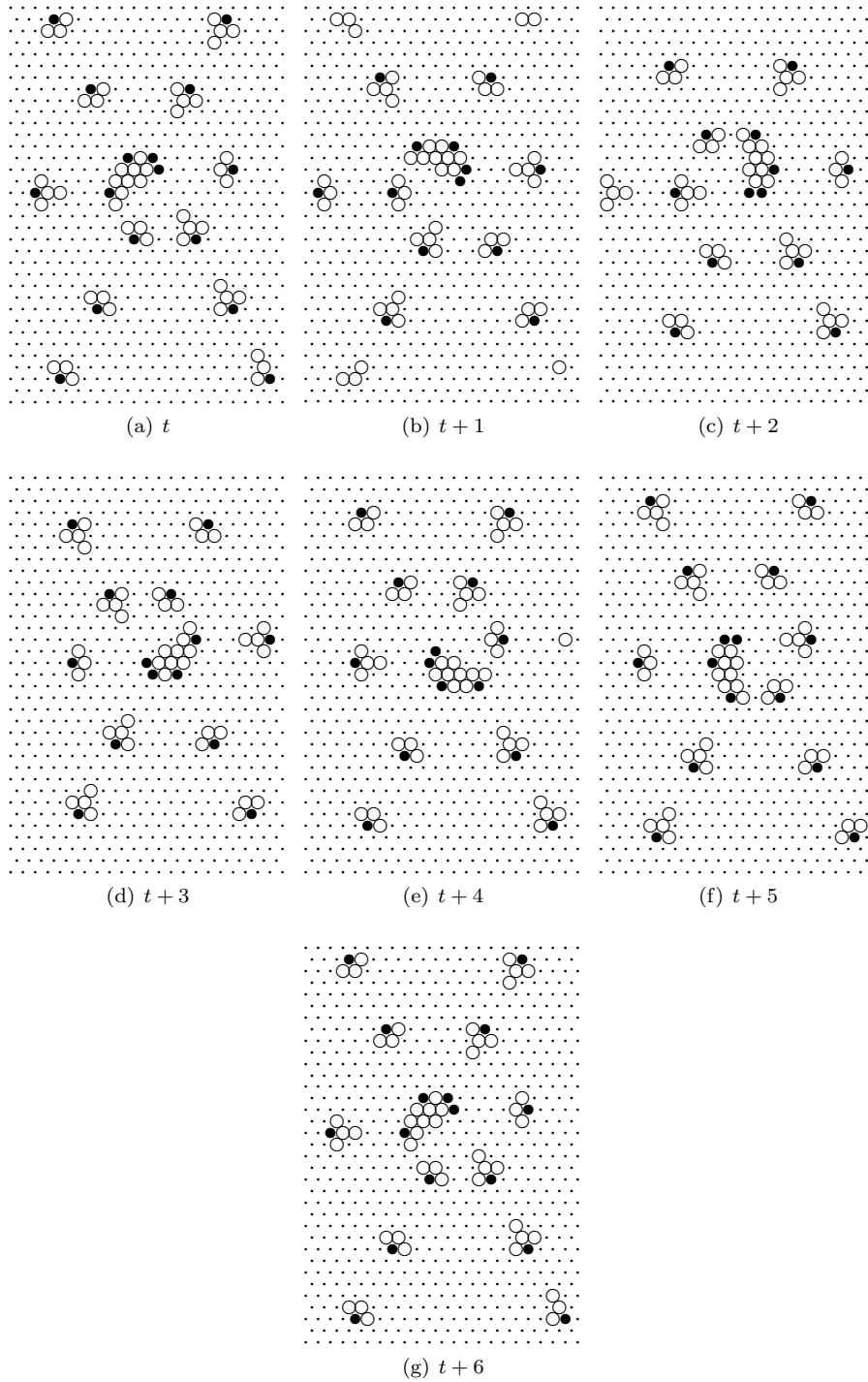


Figure 1: The full cycle of the high-frequency spiral glider gun. The “Core” of the gun is shown rotating (similarly to spiral waves in a sub-excitable Belousov-Zhabotinsky medium [14]), but can rotate in either direction. The gun emits six streams of gliders. An alternative, low-frequency, spiral glider gun also emerges in the Spiral Rule [18, 16].

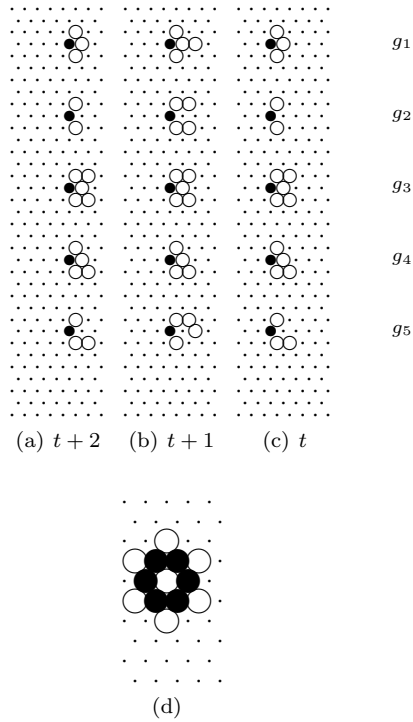


Figure 2: Basic localizations in Spiral Rule. Mobile localizations, gliders types $g1$ – $g5$, and the stationary localization, eater (d). Gliders are shown moving from right to left – time steps (c) to (a). Gliders $g1$, $g2$ and $g5$ change shape between alternate time-steps, gliders $g4$ and $g5$ are asymmetric with both chiral forms present. Cell-state I (inhibitor) is shown by a circle, cell-state A (activator) by a solid disk, and cell-state S (substrate) by a dot.

The Spiral Rule supports a rich population of other structures — compound gliders, mobile glider guns, compound eaters [17, 18], in addition to the basic ones discussed. There is also another basic stable localization SL1 [17, 18], which can modify gliders brushing past, but does not have “memory” in its outer shell like the eater discussed in the present paper.

3 Transformation of two- and four-bit strings

In [4] we discovered that substrate-sites between inhibitor-sites of the eater can be switched to an inhibitor-state by a colliding glider, or even a glider just brushing past. Thus the eater can play a role of a non-volatile memory device and also implement an asynchronous XOR gate [4]. What if we consider the states of several substrate-sites at once? Then gliders brushing the eater can change these states recursively and thus modify the string content represented by the states.

Let us encode two substrate-states in the northern part of the eater as x_1 and x_2 (Fig. 3a). If the cell x_i takes the inhibitor state it represents a bit-up state x_i of the string (x_1x_2) , if the cell x_i is in the substrate-state then it represents a bit-down state of the string (x_1x_2) . We have four possible combination of cell-states x_1 and x_2 : SS , SI , IS , II , which corresponds to binary string 00, 01, 10 and 11, respectively (Fig. 3b).

An example of a binary string transformation is shown in Fig. 4. Initially, the eater represents string 00, cells x_1 and x_2 are in the substrate state (Fig. 4a). We launch glider g_1 traveling West (Fig. 4b). When the glider brushes along the outer edge of the eater (Fig. 4a)f–k) it causes a minor perturbation of the marginal cell-states. The result of the perturbation is that one substrate-cell x_1 becomes an inhibitor (Fig. 4a)k). The glider changes its state as well — $g_1 \rightarrow g_5$ — because of its interaction with the eater (Fig. 4a)j–o).

All configurations, including those modified by gliders, are still configurations, they are not evolving by themselves but can only be changed again by another brushing glider. When we apply a sequence of transformation we do not have to worry about the precise timing of glider generation or a time interval between gliders (as far as the gliderphase is preserved). Therefore, eater-glider based transformations can be classified as asynchronous.

Finding 1 *The following transformations of two-bit strings, encoded below in digital numbers, are implemented by gliders and eaters in the Spiral Rule cellular automaton:*

$$\begin{array}{ll}
 L_1 : & L_2 : \\
 0 \rightarrow 0 & 0 \rightarrow 2 \rightarrow 0 & L_3 : 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \\
 1 \rightarrow 3 \rightarrow 1 & 1 \rightarrow 1 & L_4 : 0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0 \\
 2 \rightarrow 2 & 3 \rightarrow 3
 \end{array}$$

Assuming states '0' and '1' are Boolean values, transformations $L_1 \cdots L_3$ can be expressed in the following manner:

$$L_1(x_i, x_{i-1}) = (x_i \vee y_i \wedge (x_i \oplus x_{i-1}), x_{i-1} \vee y_i \wedge x_{i-1}) \quad (1)$$

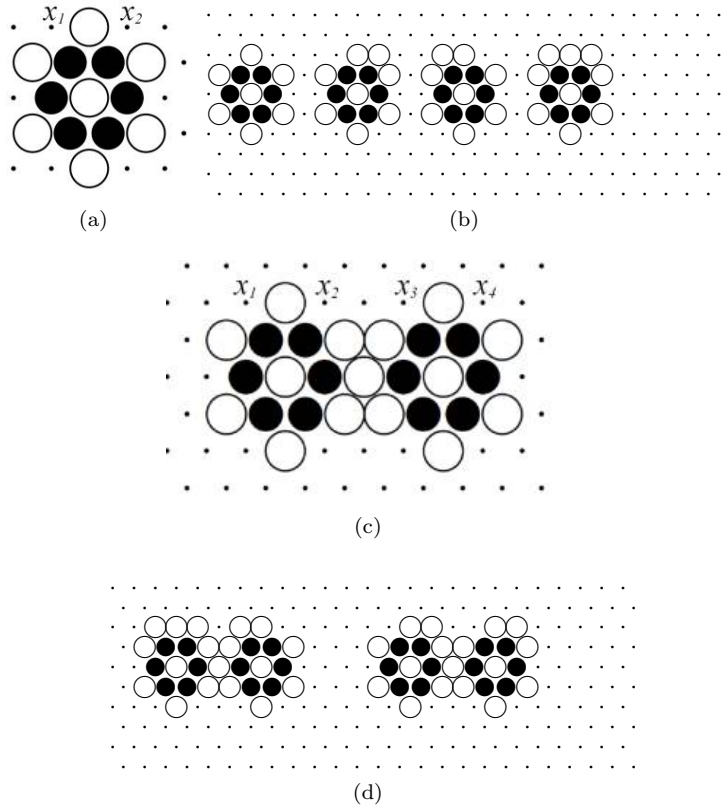


Figure 3: Encoding two-bit strings in the states of eaters. The state of cells x_1 and x_4 for a single eater encoding (a) and cells $x_1 \dots x_4$ for the eater couple (c) encoding represent bits of strings $(x_1 x_2)$ and $(x_1 x_2 x_3 x_4)$ respectively. Cell-state I (inhibitor) is shown by a circle, cell-state A (activator) by a solid disk, and cell-state S (substrate) by a dot. Cell x_i in state I represents '1', cell in state S represents '0'. Examples of encoding strings 00, 01, 10, 11 in a single eater are shown in (b). Examples of encoding strings 1110 and 0110 in the eater couple are shown in (d).

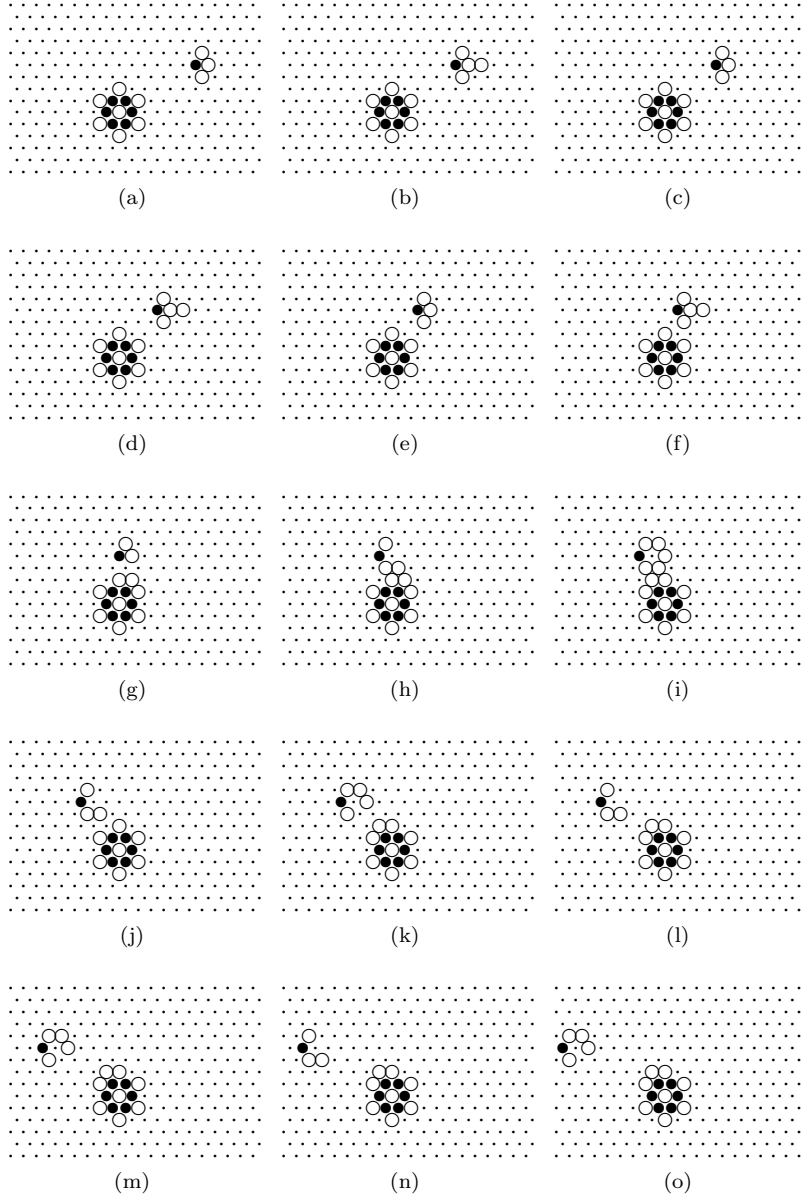


Figure 4: Transformation $00 \rightarrow 10$ implemented by glider g_1 .

$$L_2(x_i, x_{i-1}) = (x_i \vee y_i \wedge (\overline{x_i \oplus x_{i-1}}), x_{i-1} \vee y_i \wedge x_{i-1}) \quad (2)$$

$$L_3(x_i, x_{i-1}) = (x_i \vee y_i \wedge (x_i \oplus x_{i-1}), x_{i-1} \vee y_i \wedge \overline{x_{i-1}}) \quad (3)$$

$$L_4(x_i, x_{i-1}) = (x_i \vee y_i \wedge (\overline{x_i \oplus x_{i-1}}), x_{i-1} \vee y_i \wedge \overline{x_{i-1}}). \quad (4)$$

Finding 2 *One can implement increment modulo 4 using transformation L_3 and decrement modulo 4 using transformation L_3*

This follows directly from Finding 1 and Boolean representation (3).

Similarly by positioning two eaters close to each other — the distance between central states of eaters is 4 — we can change four substrate-states just with one glider. The positions of substrate-states $x_1 \dots x_4$ used for encoding are shown in Fig. 3c; examples of encoding four-bit binary sequences are demonstrated in Fig. 3d.

Example of transformation $0000 \rightarrow 1100$ is shown in Fig. 5, and transformation $1100 \rightarrow 0010$ in Fig. 6.

Finding 3 *The following transformations of four-bit strings, encoded below in digital numbers, are implemented by gliders and eaters in the Spiral Rule cellular automaton:*

$$\begin{array}{ll} T_1 : & T_2 : \\ 0 \rightarrow 12 \rightarrow 8 \rightarrow 4 \rightarrow 0 & 0 \rightarrow 6 \rightarrow 4 \rightarrow 10 \rightarrow 0 \\ 1 \rightarrow 3 \rightarrow 13 \rightarrow 7 \rightarrow 1 & 1 \rightarrow 9 \rightarrow 1 \\ 2 \rightarrow 2 & 2 \rightarrow 8 \rightarrow 14 \rightarrow 12 \rightarrow 2 \\ 5 \rightarrow 15 \rightarrow 9 \rightarrow 11 \rightarrow 5 & 3 \rightarrow 7 \rightarrow 11 \rightarrow 15 \rightarrow 3 \\ 6 \rightarrow 14 \rightarrow 6 & 5 \rightarrow 5 \\ 10 \rightarrow 10 & 13 \rightarrow 13 \end{array}$$

$$\begin{array}{l} T3 : \\ 0 \rightarrow 9 \rightarrow 14 \rightarrow 3 \rightarrow 8 \rightarrow 1 \rightarrow 6 \rightarrow 11 \rightarrow 0 \\ 2 \rightarrow 7 \rightarrow 4 \rightarrow 5 \rightarrow 10 \rightarrow 15 \rightarrow 12 \rightarrow 13 \rightarrow 2 \end{array}$$

$$\begin{array}{l} T4 : \\ 0 \rightarrow 3 \rightarrow 2 \rightarrow 13 \rightarrow 8 \rightarrow 11 \rightarrow 10 \rightarrow 5 \rightarrow 0 \\ 1 \rightarrow 12 \rightarrow 7 \rightarrow 14 \rightarrow 9 \rightarrow 4 \rightarrow 15 \rightarrow 6 \rightarrow 1 \end{array}$$

The exact conditions $C(T_i)$ or $C(L_i)$ of transformations T_i and L_i are determined by two parameters (g, d) , where a type g glider brushes the eater, and the initial distance d between the eater and the glider is even.

Finding 4 *The conditions of the transformations L_i and T_i , $i = 1 \dots 4$ are as follows:*

- $C(L_1) = C(T_1) = (g', d')$, $g' = g_3, g_4$ and $d' = \text{even, odd}$.

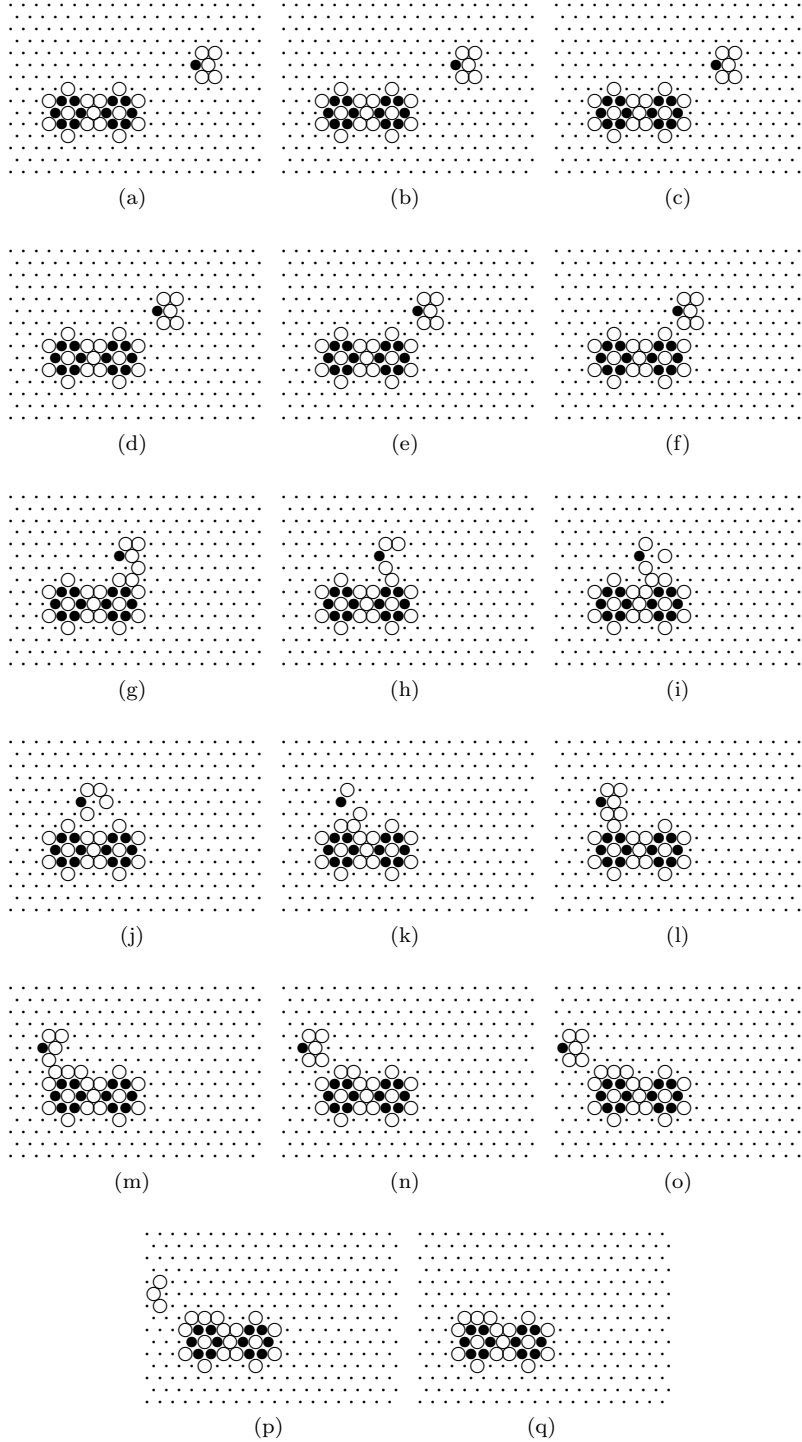


Figure 5: Transformation $0000 \rightarrow 1100$ implemented by glider g_3 .

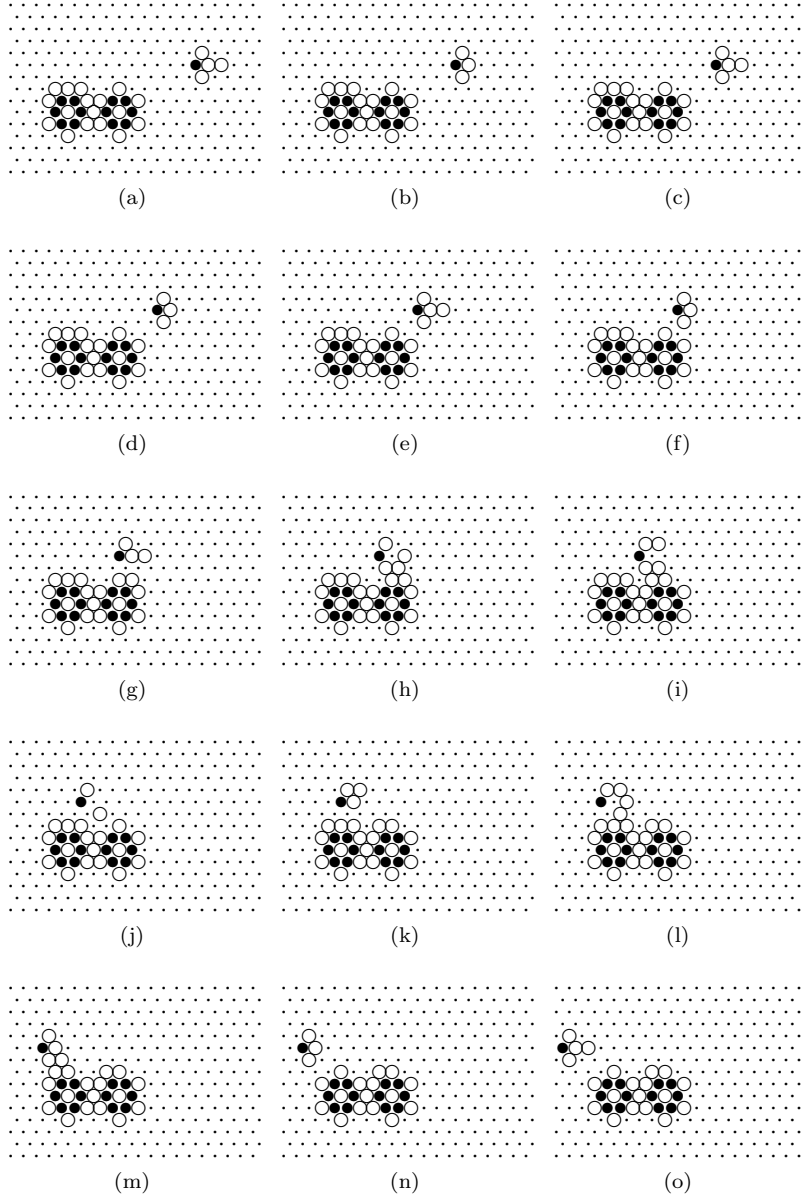


Figure 6: Transformation $1100 \rightarrow 0010$ implemented by glider g_1 .

- $C(L_2) = C(T_2) = (g', d')$, $g' = g_1$ and $d' = \text{even, odd}$.
- $C(L_3) = C(T_3) = \{(g_2, \text{even}), (g_7, \text{even}), (g_5, \text{odd})\}$
- $C(L_4) = C(T_4) = \{(g_2, \text{odd}), (g_7, \text{odd}), (g_5, \text{even})\}$.

Finding 5 Transformations T_1 and T_2 split the set $\{0, \dots, 15\}$ into stable subsets of exclusively even numbers and subsets of exclusively odd numbers.

This is an outcome of the experimental implementation of the transformations T_1 and T_2 , and analysis of chains generated by the transformations in Finding 3.

Finding 6 By using a series of transformations T_1, \dots, T_4 one can transform a number from $\{0, \dots, 15\}$ to any other number from $\{0, \dots, 15\}$ in at most three steps.

To prove this statement we compute a distance matrix $D = (d_{ij})_{0 \leq i, j \leq 15}$, where d_{ij} represents the minimal number of transformations T_1, \dots, T_4 that one needs to apply to get from number i to number j . The distance matrix has the following form:

$$D = \begin{pmatrix} 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 & 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 \\ 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 & 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 \\ 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 & 1 & 3 & 3 & 2 & 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 & 1 & 3 & 3 & 2 & 2 & 1 & 2 & 3 \\ 1 & 3 & 3 & 2 & 3 & 1 & 2 & 3 & 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 \\ 1 & 3 & 3 & 2 & 3 & 1 & 2 & 3 & 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 \\ 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 \\ 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 \\ 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 \\ 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 \\ 1 & 3 & 3 & 2 & 3 & 1 & 2 & 3 & 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 \\ 1 & 3 & 3 & 2 & 3 & 1 & 2 & 3 & 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 \\ 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 & 1 & 3 & 3 & 2 & 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 3 & 2 & 3 & 3 & 1 & 1 & 3 & 3 & 2 & 1 & 2 & 2 & 3 \\ 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 & 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 \\ 3 & 2 & 2 & 1 & 2 & 3 & 1 & 2 & 2 & 1 & 3 & 2 & 1 & 2 & 2 & 3 \end{pmatrix}$$

For example, to get 8 from 7 we must apply T_1 twice and then T_3 once: $T_3(T_1(T_1(7))) = 8$. From D we can also tell that one needs 34 to generate a glider, which brushes the eater to generate a monotonous ordered sequence of numbers $(0, 1, \dots, 15)$.

4 Six bit coding

Let us now consider the composition $\mathbf{x} \circ \mathbf{y}$ of two six bit binary strings $\mathbf{x} = (x_1 \cdot x_6)$ and $\mathbf{y} = (y_1 \cdot y_6)$. The string \mathbf{x} is represented by states of six cells $x_1 \cdot \dots \cdot x_6$ surrounding the eater, as shown in Fig. 7a. The string \mathbf{y} is represented by gliders brushing the eater (as discussed in the previous sections). The trajectories of gliders are indicated in Fig. 7: $y_i = 1$ if there is a glider traveling along trajectory y_i (Fig. 7a). Examples of encoding binary strings in the eater's states are shown in (Fig. 7b).

Gliders representing string \mathbf{y} are sent in the order y_1, \dots, y_6 . Values of Boolean variables y_i , $i = 1, \dots, 6$, are represented by the presence or absence of a glider g , traveling along trajectory y_i . A minimum interval between two subsequent gliders is selected so the next glider is sent after the current glider has already passed the eater. There is no maximal interval between the gliders. The exact structure of composition $\mathbf{x} \circ \mathbf{y}$ is determined by transformations $L_1 \cdot \dots \cdot L_4$.

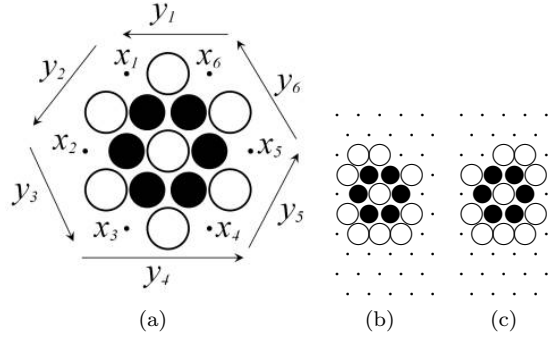


Figure 7: Encoding six-bit strings in states of the eaters. (a) States of cells $x_1 \cdots x_6$ represent bits of strings $(x_1 \cdots x_6)$. Cell-state I (inhibitor) is shown by a black disk, cell-state A (activator) by a circle, and cell-state S (substrate) by a dot. Cell x_i in state I represents '1', cell in state S represents '0'. (b)–(c) Examples of encoding strings 101100 (a) and 001101 (b). Vectors $y_1 \cdots y_6$ in (a) show which direction gliders representing input string $y_1 \cdots y_6$ travel.

For example, if the composition $\mathbf{x} \circ \mathbf{y}$ is based on L_4 then the result $\mathbf{x}' \leftarrow \mathbf{x} \circ \mathbf{y}$ can be calculated as follows:

$$\begin{aligned} x'_1 &= x_1 \vee y_1 \wedge (\overline{x_1 \oplus x_6}) \\ x'_6 &= x_6 \vee y_1 \wedge \overline{x_6} \end{aligned}$$

and for $i = 2, \dots, 6$

$$\begin{aligned} x'_i &= x_i \vee y_i \wedge (\overline{x_i \oplus x_{i-1}}) \\ x'_i &= x_{i-1} \vee y_i \wedge \overline{x_{i-1}} \end{aligned}$$

We represent the composition $\mathbf{x} \circ \mathbf{y}$ implemented by the eater and glider g_z , $z = 1, \dots, 6$, by matrix $M(L_i) = (m_{ij})_{0 \leq i, j \leq 63}$. Each entry m_{ij} of the matrix is an integer from $\{0, \dots, 63\}$. The entry m_{ij} is a digital representation of the string $\mathbf{x} \leftarrow \mathbf{x} \circ \mathbf{y}$, and its indexes i and j are digital representations of strings \mathbf{x} and \mathbf{y} , respectively.

Matrices of the compositions $\mathbf{x} \circ \mathbf{y}$ are shown in Appendix (Fig. 8–11). Structures of the matrices do not indicate any immediate implications for practical applications. There is however a range of properties which may be useful in further implementations of arithmetical operations with gliders and eaters.

Finding 7 Let $\kappa(L_i)$, $i = 1, \dots, 4$, be a ratio of commutative pairs in $\mathbf{x} \circ \mathbf{y}$ determined by transformation L_i , and $\alpha(L_i)$ a ratio of associative triples, then the following hierarchy of compositions can be drawn:

$$\begin{aligned} \kappa(L_3) &> \kappa(L_1) > \kappa(L_4) > \kappa(L_2) \\ \alpha(L_2) &> \alpha(L_3) > \alpha(L_1) > \alpha(L_4) \end{aligned}$$

This follows from straightforward analysis of the matrices $M(L_1), \dots, M(L_4)$ in Tab. 1.

Finding 8 For compositions determined by transformations L_1 and L_2 we have $\mathbf{x} \circ \mathbf{x} = \mathbf{x}$ for any \mathbf{x} from $\{0, \dots, 63\}$.

L_i	$\kappa(L_i)$	$\alpha(L_i)$
L_1	0.33	0.79
L_2	0.11	0.85
L_3	0.35	0.81
L_4	0.15	0.63

Table 1: Ratios of commutative pairs $\kappa(\cdot)$ and associative triples $\alpha(\cdot)$ in $M(L_1), \dots, M(L_4)$.

The following finding relates to identities and absorbing elements (nulls) of the compositions.

Finding 9 *Number 63 is the absorbing element of compositions determined by L_3 and L_4 . The composition determined by L_1 has two left absorbing elements: 0 and 63. The composition determined by L_2 has eighteen left absorbing elements: 21, 23, 27, 29, 31, 42, 43, 45, 46, 47, 53, 54, 55, 58, 59, 61, 62, 63. None of the composition has left or right identities.*

The structure of the first two rows of $M(L_1)$ shows that stationary and traveling localizations in the Spiral Rule automaton can implement at least one sensible arithmetical operation: transformation of arbitrary stream of numbers to streams of even or odd numbers.

Finding 10 *Let number $a \in \{0, 1\}$, encoded in a state of the eater, and number $b \in \{0, \dots, 31\}$, encoded by a stream of gliders traveling along trajectories as as per Fig. 7, then the eater-gliders system executes operation $2b + a$; the result of the operation is encoded in state of the eater.*

The resultant state of the eater can be read and erased by additional stream of gliders, as have been already demonstrated in [4].

5 Discussion

Using a cellular-automaton model [4] of a reaction-diffusion system we uncovered ways of implementing asynchronous operations over sets of binary strings. We employ a sequence of mobile localizations (gliders, excitation wave-fragments) to represent one binary string, and a stationary localization (eater, breather, standing wave) to represent another string in the composition. The operations are asynchronous in the sense that there is no precise time for a glider to collide with the eater, and there is no fixed time interval between gliders in a glider train representing a binary string. The order in which gliders hit certain parts of the eater is nevertheless important and must be executed precisely during the computation.

We found that by colliding gliders with eaters we can implement transformations of sets of two-, four- and six-bit integers, which demonstrates collision based computation in the Spiral Rule. The methods could find applications for the design arithmetical chips.

We did not discover any sophisticated operations on digital numbers. Most impressive so far were increments and decrements modulo four and subdivision of the digital number set onto prime and even numbers. We found operations

on a set of six-bit integers which are rich yet irregular and require detailed study. Nevertheless are we convinced that by combining operations (based on L_1, \dots, L_4) one can implement typical arithmetical operations. This will be subject of further analysis.

What are the potential practical outcomes of our designs and findings? As we demonstrated in [14], the Spiral Rule automaton is an ideal discrete approximation of excitation dynamics in a light-sensitive sub-excitabile chemical medium on a patterned environment. Gliders are discrete analogs of localized wave-fragments. This does not mean that our computing schemes can be implemented directly in an excitable chemical medium, not least because there are no excitation-wave analogues of eaters. More feasible implementations would be in other substrates, possibly at the nano-scales as e.g. excitation patterns in networks of single-electron oscillators [5], excitons in mono-molecular layers and breathers in polymers and crystals [1], and dynamically evolving molecular circuits [6].

References

- [1] Adamatzky A. (Ed.) *Collision-Based Computing* (Springer, 2003).
- [2] Adamatzky A., De Lacy Costello B., Asai T. *Reaction-Diffusion Computers* (Elsevier, 2005).
- [3] Adamatzky, A. Wuensche, A. and De Lacy Costello, B., Glider-based computation in reaction-diffusion hexagonal cellular automata, *Chaos, Solitons & Fractals* 27 (2006) 287–295.
- [4] Adamatzky, A., and A. Wuensche, Computing in Spiral Rule reaction-diffusion hexagonal cellular automaton, *Complex Systems* 16 (2007) 4.
- [5] Asai T. (Editor). Single-electron circuits. Special issue of the *Int. J. of Nanotechnology and Molecular Computation*. 1 (2009) 2, 1–92.
- [6] Bandyopadhyay A. and Acharya S. A 16-bit parallel processing in a molecular assembly. *Proc. Natl. Acad. Sci.* 105 (2008) 3668–3672.
- [7] Berlekamp E., J. Conway and R. Guy *Winning Ways*, vol. 2, (Academic Press, 1982).
- [8] De Lacy Costello B. and Adamatzky A. Experimental implementation of collision-based gates in BelousovZhabotinsky medium *Chaos, Solitons & Fractals* 25 (2005) 535–544.
- [9] De Lacy Costello B., Toth R., Stone C., Adamatzky A., Bull B. Implementation of glider guns in the light-sensitive Belousov-Zhabotinsky medium. *Phys. Rev. E* 79, 026114 (2009).
- [10] Fredkin E. and Toffoli T. Conservative logic *Int. J. Theor. Phys.* 21 (1982) 219–253.
- [11] Isokawa T., F. Abo, F. Peper, S. Adachi, J. Lee, N. Matsui, S. Mashiko. Fault-tolerant nanocomputers based on asynchronous cellular automata. *Int J Modern Physics C* 15 (2004) 893–915.

- [12] Lee J. and Peper F. On Brownian cellular automata. In: Adamatzky A. et al (Eds). AUTOMATA-2008: Theory and Applications of Cellular Automata (Luniver Press, 2008) 278–302.
- [13] Margolus N. Physics-like models of computation. *Physica D* 10 (1984) 81–95.
- [14] Toth T., Stone C., de Lacy Costello B., Adamatzky A., Bull L. Simple collision-based chemical logic gates with adaptive computing. *Int J Nanotechnology and Molecular Computation* 1 (2009) 1–17.
- [15] Takada Y., Isokawa T., Peper F., Matsui N. Construction universality in purely asynchronous cellular automata. *J Computer System Sci* 72 (2006) 1368–1385.
- [16] Wuensche, A. The spiral glider-gun rule (2005) http://www.cogs.susx.ac.uk/users/andywu/multi_value/spiral_rule.html
- [17] Wuensche, A. Glider dynamics in 3-value hexagonal cellular automata: the beehive rule. *Int. J. Unconventional Computing* 1 (2005) 375–398.
- [18] Wuensche, A. and A. Adamatzky, On spiral glider-guns in hexagonal cellular automata: activator-inhibitor paradigm, *Int J Modern Physics C* 17 (2006) 1009–1026.
- [19] Zhang L. and Adamatzky A. Collision-based implementation of a two-bit adder in excitable cellular automaton *Chaos, Solitons & Fractals* (2008), in press.
- [20] Zhang L. and Adamatzky A. Towards arithmetical chips in sub-excitable media: Cellular automaton models. *Int J Nanotechnology and Molecular Computation* 1 (2009) 64–82.

Appendix

