Fluctuation-Driven Computing on Number-Conserving Cellular Automata

Jia Lee[*,a], Katsunobu Imai[b], Qing-sheng Zhu[a]

[a] *College of Computer Science, Chongqing University, Chongqing, China*
[b] *Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima, Japan*

Abstract

A number-conserving cellular automaton (NCCA) is a cellular automaton where the states of cells are denoted by integers, and the sum of all numbers in a configuration is conserved throughout its evolution. It has been widely used to model physical systems ruled by conservation law of mass or energy. Imai *et al.* (2002) showed that an NCCA's local transition function can be effectively translated into sum of a binary flow function over pairs of neighboring cells. In this paper, we explore the computability of NCCAs where the pairwise number flows are performed at fully asynchronous timings. Despite the randomness associated with asynchronous transition, useful computation still can be accomplished efficiently in cellular automata, through active exploitation of fluctuations (Lee, *et al.*, 2008). In particular, certain numbers may flow randomly fluctuating between forward and backward directions in the cellular space, as if they were subject to Brownian motion. Since random fluctuation promises a powerful resource for searching through the computational state space, the Brownian-like flow of numbers allows efficient embedding of logic circuits into our novel asynchronous NCCA.

Key words: cellular automaton, number-conserving, Brownian motion, asynchronous circuit, Petri net, universal computation

1. Introduction

Cellular automata (CAs) are discrete dynamical systems that are widely used to model complex physical phenomena resulting from simple interactions at local scale. A *number-conserving cellular automaton* (NCCA) is a CA where the states of cells are denoted by integers, and the sum of all states in a configuration is conserved throughout its evolution. As number conservation possibly reflects the fundamental conservative-law of mass in physics, various NCCAs

[*] Corresponding author
Email address: lijia@cqu.edu.cn (Jia Lee)

have been extensively studied so far as models of highway traffics [14, 29], fluid dynamics [6], abstract machines [12, 25, 26, 28], and so on.

Boccara and Fukś [3] formalized conditions for 1-dimensional CAs to be number-conserving, based upon a general theorem on additive conserved quantities [9]. Durand *et al.* [5] generalized the results for higher dimensions, and provided a linear time algorithm to decide whether a CA is number-conserving or not. Moreover, focusing on structural characteristic of transition rules, Imai *et al.* [12, 39] showed an effective characterization of 2-dimensional NCCAs with von Neumann neighborhood, by which the local transition function of an NCCA is translated into a linear summation of a binary function over pairs of neighboring cells (see Theorem 1). The binary function indicates the number (maybe negative) flowing into a cell from one of its neighbors, which will result in an increment as well as a decrement in the states of these two cells, respectively.

The above pairwise flow-based characterization demonstrated that when we construct NCCAs to conduct some intended work, such as computing, what we need to do is design an appropriate binary function. Though reduction in the number of free parameters of the local function often tends to complicate the design work, Imai *et al.* [12] succeeded at constructing an NCCA with von Neumann neighborhood that is capable of universal computation. Their model uses 26 states per cell and holds permutation symmetry. Furthermore, number flows between all pairs of neighboring cells must be performed simultaneously at discrete time steps, thereby their universal NCCA is synchronous.

Relaxing the synchronous requirement of state transitions leads to the use of various asynchronous schemes [38] to iterate cells, giving rise to models called *asynchronous cellular automata* (ACAs). For example, by a stochastic updating scheme, each pairwise number flow in an NCCA is subject to a positive probability p ($0 < p \leq 1$) at every time step (see also [7]). In this case, a cell's state turns out to be a random variable which may possibly follow a binomial distribution with success probability p. No matter what the p is, however, the total sum of numbers in a configuration after one step iteration remains the same as that before the iteration, i.e., the resulting model is still number-conserving. In particular, as $p \to 0$, transitions of cells approach a Poisson process after one unit of time is remeasured by p, whereby at each time a cell as well as one of its neighbors are selected randomly from a configuration, after which a number will flow from the neighbor to the cell in accordance with the binary function.

Though randomizing the timings of pairwise number flows does not disturb the conservation property of originally synchronous NCCAs, it often brings difficulty in designing models to carry out computation. In general, a special timing mechanism is required by ACAs in order to avoid unexpected behavior during the randomly-timed transitions [30, 18], which in turn usually causes the increase in the complexities of ACAs as compared to their synchronous counterpart. Nevertheless, inclusion of fluctuation into ACAs promises models with less complexity, e.g., the Brownian cellular automata (BCAs) [17, 21, 35]. A BCA is an ACA where certain configurations, as signals, may move randomly fluctuating between forward and backward directions, as if they were subject to Brownian motion. Because Brownian motion offers an effective resource for

searching through the computational state space [2, 4, 15], the BCA [17] requires much lower complexity than other non-Brownian ACAs with computational universality achieved thus far [19, 32, 33]. Furthermore, though the number of each state in a configuration is conserved throughout the iteration, the BCA is not exactly the same as an NCCA because its transition function is unable to be translated into the sum of a binary function.

Cellular automata have gained much attentions as a promising architecture for future nanocomputers, because their regular structures potentially allow manufacturing techniques based on molecular self-assembly [1, 22, 32, 33]. With an aim to verify fluctuation-driven computation in more general and physics-like models (e.g., [23, 25, 41, 42]), this paper focuses on the framework of NCCAs. For this purpose, we present a novel NCCA where pairwise flows of numbers are performed at fully asynchronous timings. In this case, Brownian-like behavior is emulated by random flow of numbers fluctuating back and forth in the cellular space. Asynchronous systems, like Petri net [11], usually suffer from the issue of deadlocks. Since Brownian motion provides a natural way to backtrack from deadlock situations, the fluctuation-driven computing scheme promises more effective circuit constructions [34, 20] and offers the potential for physical implementation by future nano-electronics [37]. As a result, our novel NCCA model requires merely 11 cell states as well as a simple binary function, and is able to implement any arbitrary logic circuit in its cellular space, thereby it is computationally universal.

It is worthy pointing out an alternative efficient and intuitive way to express number-conservation of CAs, based on the notion of *particle automata* (PAs) [27]. By PA, each cell's state is interpreted as the number of particles included in it, and a rule is used to decide the motion of each particle in a cell, which may either move to one of the cell's neighbors or simply stay in that cell. In this case, according to the notion of *motion representation* [3, 27] for PAs, the binary flow function in our NCCA can be transformed straightforward into a list of local configurations, each of which consists of two adjacent cells in a horizontal or vertical direction, associated with an arrow indicating the movement of particles from one cell to the other cell. Furthermore, one pair of adjacent cells is randomly selected at every time step, between which a certain number of particles is allowed to move. Thus, our NCCA model may also be formalized as an asynchronous PA with a minimal neighborhood.

This paper is organized as follows: Section 2 outlines the basic notion of NCCAs. More details can be found in [3, 5, 7, 12, 26, 27]. Section 3 describes a set of simple circuit elements [20, 17] that can actively exploit the fluctuation of signals. After that, Section 4 introduces an asynchronous NCCA model and shows its universality for constructing logic circuits. Besides our totally asynchronous NCCA, Section 5 demonstrates that it is still possible to emulate Brownian motion in a stochastic updating NCCA, by using a more complicated flow function. This paper finishes with the conclusion and further discussions given in Section 6.

2. Number-Conserving Cellular Automata

Definition 1. A *deterministic 2-dimensional cellular automaton with von Neumann neighborhood* is defined as (Z^2, Q, f, q_0), in which Z is the set of all integers, Q is a finite set of states, and $q_0 \in Q$ is a special state called *quiescent* state. In addition, $f : \ Q^5 \to Q$ is a local transition function which satisfies $f(q_0, q_0, q_0, q_0, q_0) = q_0$.

Definition 2. Assume a cellular automaton (Z^2, Q, f, q_0). a *configuration* over Q is a mapping $c : \ Z^2 \to Q$. Moreover, the set of all configurations is defined as $\Gamma(Q) = \{c \mid c : \ Z^2 \to Q\}$.

Assume $c, c' \in \Gamma(Q)$. When all cells undergo state transitions simultaneously, we say there is a *global transition* from c to c' written as $c \longrightarrow c'$ if

$$\forall (x, y) \in Z^2, \ c'(x, y) = f(c(x, y), c(x, y-1), c(x+1, y), c(x, y+1), c(x-1, y)).$$

On the other hand, if cells do transitions asynchronously such that one randomly selected cell is updated at a time, $c \longrightarrow c'$ holds if

$$\exists (x', y') \in Z^2, \forall (x, y) \in Z^2 \setminus (x', y'), \quad c'(x, y) = c(x, y) \wedge$$
$$c'(x', y') = f(c(x', y'), c(x', y'-1), c(x'+1, y'), c(x', y'+1), c(x'-1, y')).$$

Definition 3. Assume a CA (Z^2, Q, f, q_0). The function f satisfies *rotation symmetry* iff

$$\forall c, u, r, d, l \in Q, \ f(c, u, r, d, l) = f(c, r, d, l, u).$$

In addition, function f satisfies *permutation symmetry* iff it is rotation symmetry and

$$\forall c, u, r, d, l \in Q,$$
$$f(c, u, r, d, l) = f(c, u, d, l, r) = f(c, u, l, r, d) = f(c, u, d, r, l) = f(c, u, l, d, r) = f(c, u, r, l, d).$$

For convenience, let N denote a finite set of integers, such that $N \subset Z$ and $0 \in N$. Thus, we have the next definition.

Definition 4. A CA $(Z^2, N, f, 0)$ is called *number-conserving* if

$$\forall c, c' \in \Gamma(N), \quad c \longrightarrow c' \implies \sum_{(x,y) \in Z^2} (c(x, y) - c'(x, y)) = 0.$$

Assume all cells do state transitions simultaneously, the following theorem provides a sufficient and necessary condition for a von Neumann neighborhood NCCA.

Theorem 1. [12] *Assume a CA* $(Z^2, N, f, 0)$ *where* $f : N^5 \to N$ *satisfies permutation symmetry. The CA is number-conserving iff*

$$\forall c, u, r, d, l \in N, \quad f(c, u, r, d, l) = c + \sum_{\alpha \in \{u,r,d,l\}} g(c, \alpha)$$

where $g : Z^2 \to Z$ *is a binary function satisfying* $g(b, a) = -g(a, b)$ *for all* $(a, b) \in Z^2$.

The binary function g indicates a number (maybe negative) flowing into a cell from one of its non-diagonal adjacent cells, which will cause an increment as well as a decrement in the states of these two cells, respectively. Since the local function f of an NCCA is perfectly described in terms of a binary function g, it is possible to redefine an NCCA as $(Z^2, N, g, 0)$.

Let $(Z^2, N, g, 0)$ be an NCCA with von Neumann neighborhood. Suppose this NCCA is iterated asynchronously, such that flow of numbers takes place between utmost one pair of neighboring cells at a time, and the pair may be selected randomly from the entire configuration. In this case, for any $c, c' \in \Gamma(N)$, $c \longrightarrow c'$ holds if

$$\exists \vec{\mathbf{a}}, \vec{\mathbf{b}} \in Z^2, \forall \vec{\mathbf{r}} \in Z^2 \setminus \{\vec{\mathbf{a}}, \vec{\mathbf{b}}\}, \quad |\vec{\mathbf{a}} - \vec{\mathbf{b}}| = 1 \wedge$$
$$c'(\vec{\mathbf{r}}) = c(\vec{\mathbf{r}}) \ \wedge \ c'(\vec{\mathbf{a}}) - c(\vec{\mathbf{a}}) = g(c(\vec{\mathbf{a}}), c(\vec{\mathbf{b}})) \ \wedge \ c'(\vec{\mathbf{b}}) - c(\vec{\mathbf{b}}) = g(c(\vec{\mathbf{b}}), c(\vec{\mathbf{a}})).$$

Because $g(c(\vec{\mathbf{a}}), c(\vec{\mathbf{b}})) = -g(c(\vec{\mathbf{b}}), c(\vec{\mathbf{a}}))$, we obtain $\sum_{\vec{\mathbf{a}} \in Z^2} (c'(\vec{\mathbf{a}}) - c(\vec{\mathbf{a}})) = 0$. Therefore, according to Definition 4, the CA remains number-conserving even when cells are updated independently at random timings.

3. Token-Based Circuits and Brownian Operators

Tokens are discrete indivisible units that are represented graphically as black dots. Logic circuits that are concerned in this paper are token-based, such that communications between a circuit and the outside world are done via exchanging tokens through the input/output lines. They are also *asynchronous* in the sense that their correct operations are unaffected by arbitrary delays involved in interconnection lines or operators [13, 36, 10]. A well-known semantical model for the description of token-based systems is the *Petri net* (e.g., [8]). Below, a brief overview of Petri nets is given.

Definition 5. A *Petri net* (or simply *net*) is defined as (P, T, F) where P and T are finite sets of places and transitions ($P \cap T = \emptyset$), respectively. In addition, $F \subseteq (P \times T) \cup (T \times P)$ denotes the flow relation between places and transitions.

Definition 6. A *marking* of a net (P, T, F) is a function $\mu : P \to \{0, 1\}$, which assigns zero or one token to each place in the net. For simplicity, the marking can also be defined by $M = \{p \in P \mid \mu(p) = 1\}$.

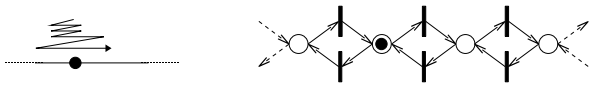
The graphical representation of a Petri net usually uses circles and bars to denote the places and transitions, respectively. Assume a net (P, T, F). For each $< \alpha, \beta > \in F$ with $\alpha, \beta \in P \cup T$, this is indicated by an arc from α to β. Also, Let M be a marking of the net. Suppose a place $p \in M$, i.e., p contains a token, it is represented by placing a black dot in the corresponding circle.

Definition 7. Assume a net (P, T, F) and $t \in T$. Let $\bullet t = \{p \in P \mid < p, t > \in F\}$ and $t\bullet = \{p \in P \mid < t, p > \in F\}$ denote the input and output sets of transition t, respectively.

Definition 8. Let M be a marking of a net (P, T, F). A transition $t \in T$ is called *firable* at M, if $\bullet t \subseteq M$ and $t\bullet \cap M = \emptyset$. Moreover, *firing* of transition t results in a new marking M' where $M' = (M - \bullet t) + t\bullet$.

In general, when several transitions are firable simultaneously, any one of them may fire. That is, the firing in a net is nondeterministic (asynchronous). In particular, assume $t_1, t_2 \in T$ are both firable in a net (P, T, F). If $\bullet t_1 \cap \bullet t_2 \neq \emptyset$, then the firing of t_1 or t_2 will disable the other transition temporarily, which gives rise to a *conflict* situation. Such situations may arise in token-based asynchronous circuits, for example, when two tokens try to access a shared component at the same time. The resulting conflict usually requires a purely nondeterministic or even stochastic functionality, called *arbitration* or *choice*, to resolve them [13, 36, 10, 31].

A special type of asynchronous circuits is the *Brownian circuit* [17, 34, 20], in which the movements of tokens on lines may fluctuate back and forth at random. A net representation of a (bi-directional) line where a token may fluctuate between going leftward and rightward directions is illustrated as follows:



The Brownian movements of tokens enable a circuit to backtrack from the conflict states, thereby providing the arbitration as part of their nature. This actually allows for more effective design of primitive operators and circuit constructions [17, 34, 20, 21]. As a result, three kinds of simple operators that can actively exploit the fluctuations of tokens are shown in Fig. 1, each of which has much less complex functionality than the primitive elements [31] of conventional asynchronous circuits. Furthermore, any arbitrary asynchronous circuit can be constructed by these elements, according to the following theorem.

Theorem 2. [17, 20] *{CJoin, Hub, Ratchet} is universal for asynchronous circuits.*

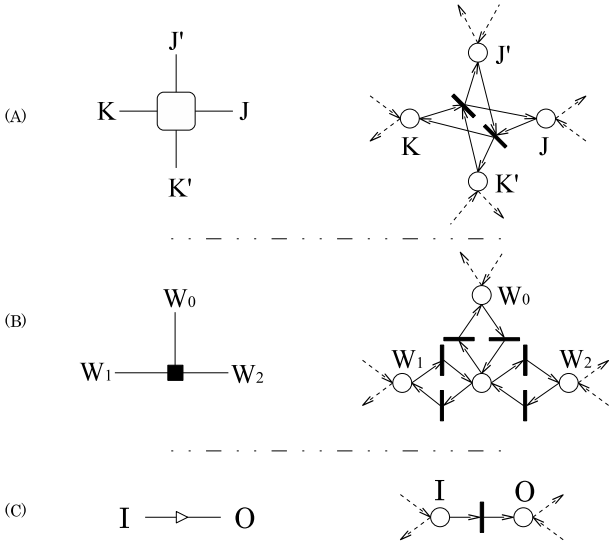


Figure 1: Brownian operators [17, 20] and their net representations. (A) **CJoin** (Conservative Join): Two tokens with one arriving on line J (or K) and another one on line J' (resp. K') are processed and give rise to one token on each of the lines K and K' (resp. J and J'), respectively. (B) **Hub**: A token arriving on line W_i will be transferred to one of the other lines W_j, where $i, j \in \{0, 1, 2\}$ and $i \neq j$. (C) **Ratchet**: A token arriving on line I is transferred to line O. This element works as a diode such that once a token passes it from I to O, the token cannot go back.

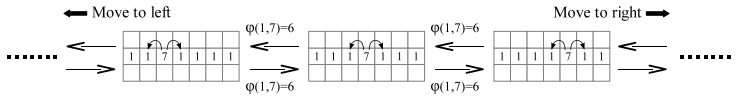


Figure 2: Sequence of configurations where, due to the randomness in the selection of neighboring cells, the number 6 from a 7-state cell fluctuates between flowing into the cell's left neighbor and right neighbor, which possibly resembles the Brownian-like movements of a token on a line. For simplicity, quiescent cells are denoted by blanks. Following the notation of motion representation [3, 27], each curved arrow in a configuration indicates that a positive number (of particles) is ready to flow (move) from a source cell to an adjacent destination cell, what implies is a global transition from that configuration if these two cells are selected. Moreover, each time a flow function $\varphi(x, y)$ (as well as $\varphi(y, x)$) is used, it is denoted along with the straight arrow depicting a global transition.

4. Universality of An Asynchronous Number-Conserving Cellular Automaton

4.1. Definition

Our novel NCCA with von Neumann neighborhood is defined by $A_{11} = (Z^2, N_{11}, \varphi, 0)$, where $N_{11} = \{0, 1, \cdots, 9\} \cup \{15\}$. The flow function $\varphi : Z^2 \to Z$ is given as follows:

$$\begin{array}{llll} \varphi(1,7) = 6, & \varphi(3,7) = 6, & \varphi(1,9) = 6, & \varphi(2,7) = 6, \\ \varphi(3,8) = 3, & \varphi(2,6) = 3, & \varphi(3,5) = 3, & \varphi(5,6) = 3, \\ \varphi(1,8) = 6, & \varphi(4,9) = 4, & \varphi(2,5) = 2, & \varphi(9,8) = 6, \\ \varphi(3,15) = 12, & \varphi(2,15) = 6. & & \end{array}$$

For all $x, y \in Z$, if $\varphi(x, y)$ or $\varphi(y, x)$ is defined above, then $\varphi(x, y) = -\varphi(y, x)$; otherwise $\varphi(x, y) = 0$. Moreover, we apply the *random choice* [38] scheme to iterate cells, such that at each time step, one cell together with one of its neighbors are selected randomly with uniform probability from the configuration, after which a number will flow from the neighbor to the cell in accordance with φ. The A_{11}, therefore, is asynchronous.

Universal computation in A_{11} is accomplished through the embedding of logic circuits into its cellular space. In particular, circuit constructions are based upon the primitive operators given in Fig. 1.

4.2. Embedding Brownian token and operators into A_{11}

In A_{11}, a line is represented as a linearly continuous sequence of cells in state 1, on which a token is represented by changing a 1-state cell into state 7, as shown in Fig. 2. Since a token has no distinct head and tail, driven by the flow function $\varphi(1, 7) = 6$, the token will propagate back and forth randomly on a line, resembling a particle undergoing Brownian motion. In addition, the layout of crossing lines is represented by the configuration in Fig. 3, through which a token running on either line can pass properly.

Local configurations each of which behaves like one of the Brownian operators in Fig. 1, are illustrated in Figs. 4–6, respectively. Thus, due to the

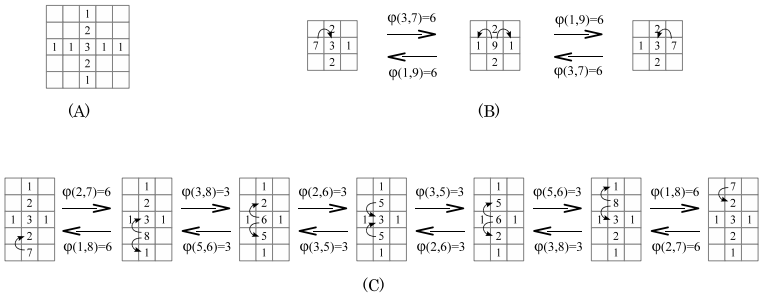


Figure 3: (A) Local configuration of two crossing lines. Sequences of configurations where a token arriving on either the (B) horizontal line or the (C) vertical line but without another token on the orthogonal line, passes through the crossing point freely.

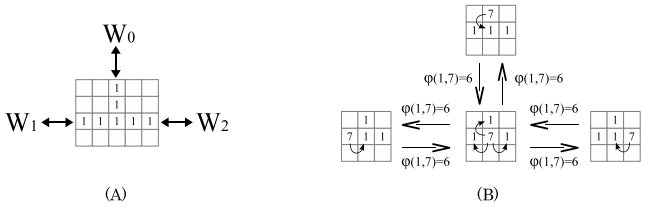


Figure 4: (A) Local configuration representing a Hub. (B) Sequence of configurations where the Hub receives one token from one of its lines, after which it may transfer the token to either of other lines.

.

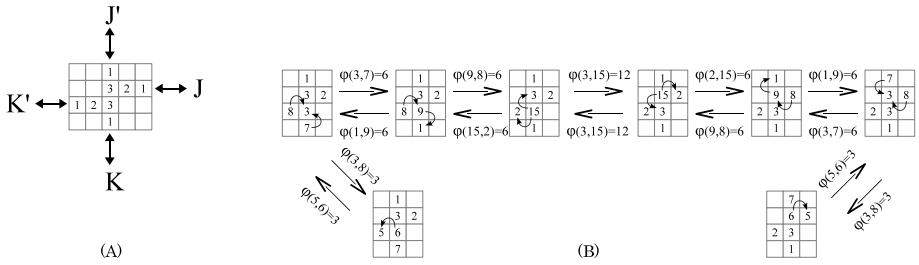


Figure 5: (A) Local configuration representing a CJoin. (B) Sequence of configurations where the CJoin receives one token from each of its lines K' and K (resp. J' and J), after which it produces one token to each of the lines J and J' (resp. K and K'), respectively. Due to asynchronicity of the CA, several different sequences of configurations are possible.

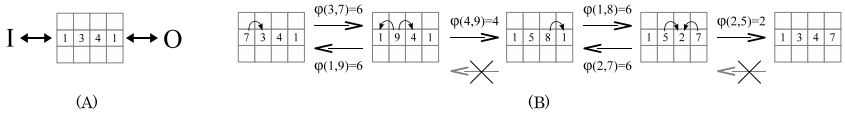


Figure 6: (A) Local configuration representing a Ratchet. (B) Sequence of configurations where the Ratchet receives one token from line I, after which it transfers the token to line O. Though a token may fluctuate on lines I and O before and after passing the Ratchet, it cannot go back from line O to line I.

universality of these operators, it is possible to construct any arbitrary asynchronous circuit in the cellular space of A_{11}.

Moreover, Fig. 3 demonstrates that a single token can pass the crossing point of two orthogonal lines, no matter it propagates on the horizontal line or the vertical line. But what would happen if one token appears on each of the orthogonal lines at the same time? Because of the asynchronicity of A_{11}, collision between tokens at the crossing point may take place (see Fig. 7A). Previous BCA models [17, 21, 35] simply treat such a collision as deadlock and do nothing, since the Brownian motion of tokens will eventually move one token away from the crossing point, so as to allow another token to pass first. In A_{11}, however, the process to deal with collision tends to be much more complicated, as demonstrated in Fig. 7B. This is because A_{11} can access no more than two adjacent cells at a time, and hence, it is difficult to recognize the collision shown in Fig. 7A as a deadlock immediately in a straightforward way.

In conclusion, any arbitrary asynchronous circuit can be constructed in A_{11}, and hence, the following theorem holds.

Theorem 3. *A_{11} is computationally universal.*

The next subsection demonstrates the embedding of a well-known logic circuit, called Half-Adder, in the cellular space of our A_{11}.

4.3. Embedding Half-Adder into A_{11}

A Half-Adder is a logic circuit that performs an additional operation over two one-bit binary numbers. It has two inputs $\{x, y\}$ and two outputs $\{z, c\}$, along with the truth table given below.

x	y	z	c
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Conventionally, the output $z = x \oplus y$ is used to represent the sum of x and y, as well as $c = x \cdot y$ representing the carry. To implement the Half-Adder in token-based systems, a robust scheme employs a pair of lines to express each

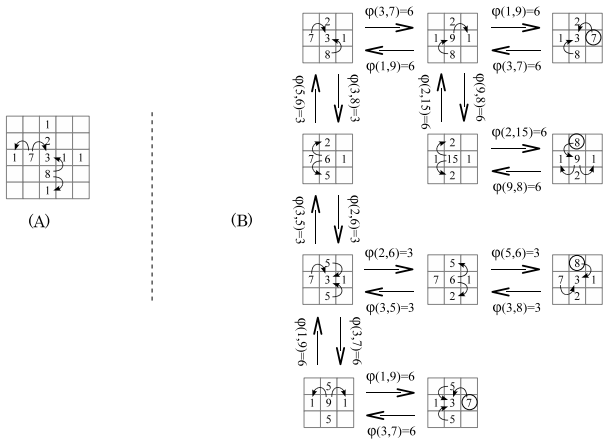


Figure 7: (A) Collision between two tokens at a crossing point. (B) Typical sequences of configurations where one token will pass through the crossing point first, after which another token becomes able to pass. Due to asynchronicity of the CA, there are so many different sequences in which this process may occur. For clarity, tokens which passed first at the right end of each sequence are indicated by circles, respectively. No matter in which sequence the process takes place (in which order the tokens pass), tokens running on different lines can always propagate safely and individually without interference from each other.

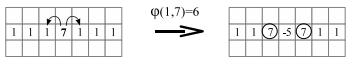
binary number. This scheme, called *dual-rail encoding*, encodes the value 1 by putting a token on one line, and value 0 by putting a token on the other line.

Figure 8A provides the design of a Half-Adder using the Brownian operators in Fig. 1, in accordance with which configuration representing a Half-Adder is illustrated in Fig. 8B.

5. More on Brownian Tokens

The configuration representing a token on a line, given in Fig. 2, is really so simple with rotational symmetry, that there is no preferred direction for the token to move on the line. The number 6, therefore, from a 7-state cell (representing a token) takes an equal chance to flow into neighboring cells in the left and right directions, albeit not at the same time. The choice between the left and right movements may be done by sorting all pairs of adjacent cells (that are ready for number flows) in a configuration, in accordance with which cells are updated sequentially. In this sequence, if the pair composed of the token (7-state) cell and its left neighbor precedes the other pair of the same token cell and its right neighbor, the token will move to the left direction, as shown in Fig. 2; otherwise the toke will move to right. Since the order among all pairs of neighboring cells will be shuffled randomly at the beginning of each round of cell transitions, the resulting token will run randomly on the line, fluctuating between going leftward and rightward.

What would happen, however, if we loose the fully asynchronous updating scheme, by allowing the 7-state cell in Fig. 2 to flow numbers to both its left and right neighbors simultaneously? The following figure shows the result of such concurrent flows of numbers, where a single token is divided into two, which is actually prohibited by the indivisible principle of tokens (see Section 3).



As a result, our asynchronous NCCA model will lose its computing ability when numbers from a single cell may flow into multiple neighbors at the same time. In other words, to realize a Brownian token in the face of concurrent number flows, we must design an alternative binary function against the function φ in our NCCA (Section 4).

For example, an alternative flow function used for Brownian movements of tokens can be defined as follows: Assume $\tau : \ Z^2 \to Z$ such that

$$\tau(1,7) = 6, \quad \tau(5,2) = 1, \quad \tau(2,6) = 5.$$

Given in Fig. 9, the configuration of a line on which a token is put is the same as that in Fig. 2. As said before, an effective scheme for emulating concurrent behavior is to assume that at every time step, each pair of neighboring cells is subject to a certain probability p $(0 < p < 1)$ to flow a number between them. Since $p > 0$, stating from the configuration (a) in Fig. 9, there is always a positive likelihood at any time that number flows between each pair of adjacent cells are performed simultaneously, which implies a successive sequence of global transitions alternating between from configuration (a) to configuration (b) and vice versa, where the token will simply stay on the line without move. Fortunately, such sequence cannot last for long whenever $p < 1$, and instead, only a random portion of cells on average will be chosen to undergo number flows at each time step. In this case, the asynchrony of the updating timings, together with the randomness in the selections of cells, will eventually result in Brownian-like movements of a token on a line, as illustrated in Fig. 9.

Furthermore, in Fig. 9, the token (7-state) cell in configuration (a) takes a possibility p^2 (or $(1-p)^2$) to flow a number into both (resp. neither) of its left and right neighbors, whereas the possibility to flow into either of its neighbors is $2p(1-p)$. The case for configuration (b) is similar. It turns out that to move a token one cell in either direction of a straight line, would take about $\frac{1}{2p(1-p)}$ time steps on average. In particular, such expected time steps get minimal when $p = 0.5$. That is, a token tends to move most intensely on a line in a Brownian fashion, when the occurrence of actual number flow between any pair of neighboring cells becomes completely unpredictable.

6. Conclusions and Discussions

Although by intuition the pure asynchronism in updates of individual cells seems incompatible with the conservation law of sum of all cells' states, Fukś [7]

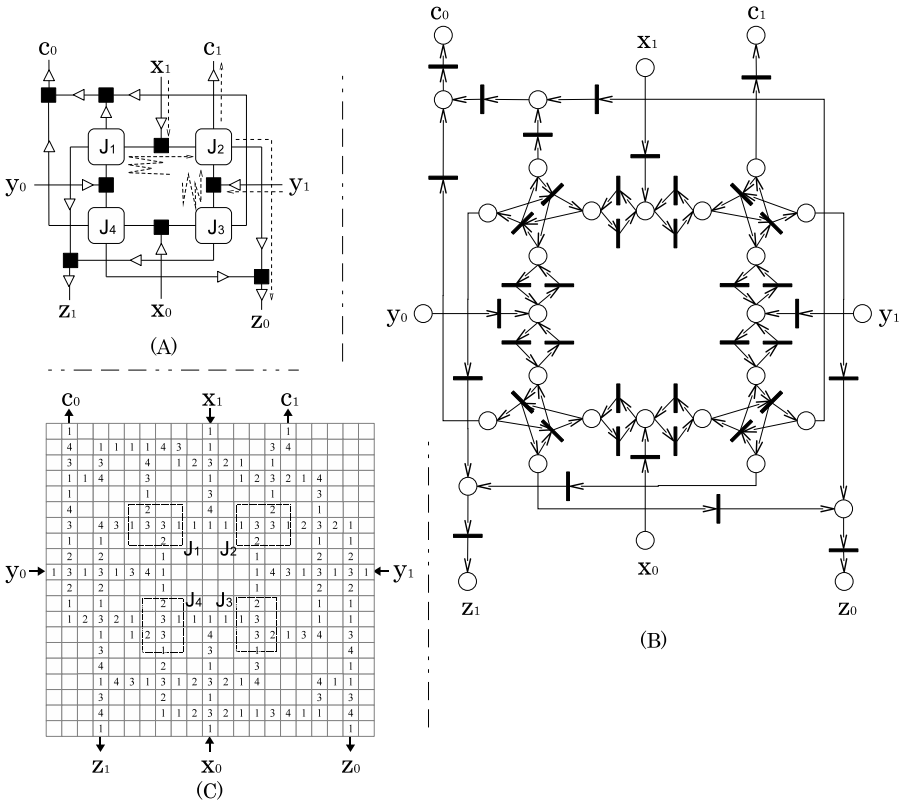


Figure 8: (A) Construction of a dual-rail encoded Half-Adder using Brownian operators in Fig. 1, and (B) its net representation. For each $\alpha \in \{x, y, z, c\}$, a token appearing on line α_1 (resp. α_0) represents $\alpha = 1$ (resp. $\alpha = 0$). For convenience, traces of tokens in response to an input coming from each of the lines x_1 and y_1 (i.e., $x = 1$ and $y = 1$) are depicted by dashed lines. Roughly speaking, after a token from line x_1 (or y_1) is received, it may move around in a random walk manner on the line between the CJoins J_1 (resp. J_3) and J_2, searching alternately for the CJoin at either end of the line which gets ready to process tokens (see Fig. 1A). Thus, the Brownian behavior of tokens allows a natural way to realize the arbitration function. Moreover, the Brownian motion will eventually move both the tokens to reach the J_2 simultaneously, whereby the tokens are able to activate the operation of J_2 and finally result in one token appearing on each of the output lines z_0 and c_1 (i.e., $z = 0$ and $c = 1$). In addition, the placement of Ratchets on lines is crucial to restrict the movements of tokens to one direction, which will substantially speed up the circuit's operation. (C) Construction of a dual-rail encoded Half-Adder in A_{11} in accordance with the circuit scheme in (A). Within the configuration, each region where a CJoin operator is placed is surrounded by a dashed-box. (A simulation of the dual-rail encoded Half-Adder can be found in the attached movie file: half-adder.wmv, starting by one signal being put on each of the lines x_1 and y_1.)

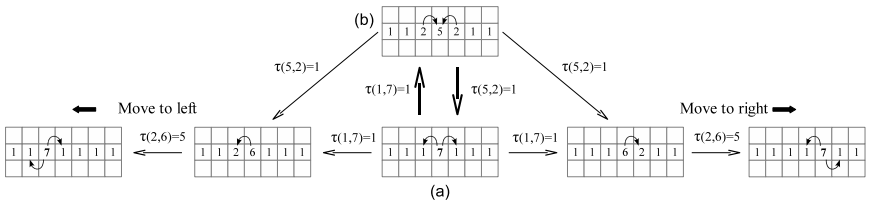


Figure 9: Propagation sequence of a token on a line where number flows may take place between multiple pairs of cells simultaneously at a time. For convenience, a bold arrow expresses that more than one pair of cells are chosen to perform number flows at the same time, whereas a thin arrow indicates that at most one pair of cells is selected out from all candidates to undergo number flow. In particular, both configurations (a) and (b) contain two pairs of cells that are ready for number flows, due to the flow function τ. Hence, simultaneous flows of numbers between the two pairs result in a global transition from configuration (a) (or (b)) to configuration (b) (resp. (a)). On the other hand, whenever a time lag occurs between the pairs, i.e., one pair precedes the other in flowing a number, the token will eventually move one cell and the direction depends on which pair is chosen first.

pointed out that the sufficient and necessary condition for synchronous CAs to be number-conserving can be naturally extended for probabilistic cellular automata. Likewise, the characterization of synchronous NCCAs (Theorem 1) by Imai *et al.* [12] ensures the conservation of CAs even when the pairwise number flows take place independently at random times. This paper explored the computability of asynchronous NCCAs, via a novel model which employs 11 states per cell and a simple binary flow function. Universal computation in our asynchronous NCCA was achieved through embedding of logic circuits, like the Half-Adder, into the cellular space. Circuit constructions were based upon primitive operators [20, 17] that can actively exploit the fluctuations of tokens.

Random fluctuation provides an effective and powerful resource for biological systems [40]. The BCA models designed in [17, 21, 35] demonstrated that they can be incorporated efficiently into the searching process associated with computation. In particular, the Brownian-like behavior of tokens allows a natural realization of arbitration and choice, a functionality that is essential for asynchronous systems but usually hard to implement without the use of Brownian motion [32, 34, 20, 17]. The effectiveness of Brownian motion, for example, may be well explained by the extremely simple representation of a token as well as a simple function used to drive its propagation (see Fig. 2). This characteristic, which seems having no counterpart in other models [12, 39], actually contributed to the small number of states and the simplicity of transition function in our asynchronous NCCA.

Our model uses a fully asynchronous scheme to iterate cells, by which all pairs of neighboring cells undergo transitions sequentially in a random order. As mentioned in the Introduction, a more general scheme assumes that each pair of neighboring cells is subject to a positive probability. Though it is still possible, as shown in Fig. 9, to emulate Brownian motion even when a number may

possibly flow into both the leftward and rightward directions simultaneously, a more complicated binary function (Section 5) seems needed as compared to the function of our NCCA model. Thus, how to compute efficiently on the stochastic updating NCCAs, especially how to realize those Brownian operators in Fig. 1, still needs further investigation.

Moreover, the pairwise flow-based characterization requires the NCCA models to satisfy permutation symmetry [12], which is so strong as the outer-totalistic ones. Relaxing the symmetry condition results in an additional binary function used to describe the local function of an NCCA [39]. In this case, the number flowing into a cell not only comes directly from one of its non-diagonal neighbors, but also is affected by the interaction between two neighbors in a diagonal direction. Though the additional function may contribute to reduce the complexity of synchronous NCCAs [39], it is yet unknown how to guarantee the total sum of numbers in a CA never change in the face of such indirect influence, when time lags between the transitions of cells may be involved.

Acknowledgments

We are grateful to the anonymous reviewers' helpful comments and valuable suggestions. This research work was supported by the Program for New Century Excellent Talents in University (No. NCET-09-0840).

References

[1] A. Bandyopadhyay, R. Pati, S. Sahu, F. Peper, D. Fujita, Massively parallel computing on an organic molecular layer, Nature Physics 6(5) (2010) 369–375.

[2] C.H. Bennett, The thermodynamics of computation—a review, Int. J. of Theoretical Physics 21(12) (1982) 905–940.

[3] N. Boccara, H. Fukś, Number-conserving cellular automaton rules, Fundam. Inform. 52 (2001) 1–13.

[4] S. Dasmahapatra, J. Werner, K.P. Zauner, Noise as a computational resource, Int. J. of Unconventional Computing 2(4) (2006) 305–319.

[5] B. Durand, E. Formenti, Z. Róka, Number-conserving cellular automata I: decidability, Theor. Comp. Sci. 299 (2003) 523–535.

[6] U. Frisch, B. Hasslacher, Y. Pomeau, Lattice-Gas Automata for the Navier-Stokes equation, Phys. Rev. Lett. 56 (1986) 1505–1508.

[7] H. Fukś, Probabilistic cellular automata with conserved quantities, Nonlinearity 17(1) (2004) 159–173.

[8] C. Girault, R. Valk, Petri Nets for Systems Engineering, Springer, 2003.

[9] T. Hattori, S. Takesue, Additive conserved quantities in discrete-time lattice dynamical systems, Physica D 49 (1991) 295–322.

[10] S. Hauck, Asynchronous design methodologies: an overview, Proc. IEEE 83 (1) (1995) 69–93.

[11] L.E. Holloway, B.H. Krogh, A. Giua, A survey of Petri net methods for controlled discrete event systems, Discrete Event Dynamic Systems 7(2) (1997) 151–190.

[12] K. Imai, K. Fujita, C. Iwamoto, K. Morita, Embedding a logically universal model and a self-reproducing model into number-conserving cellular automata, Proc. UMC'02, Kobe, LNCS 2509 (2002) 164–175.

[13] R.M. Keller, Towards a theory of universal speed-independent modules, IEEE Trans. Comput. C-23 (1) (1974) 21–33.

[14] B. Kerner, S. Klenov, D. Wolf, Cellular automata approach to three-phase traffic theory, J. Phys. A: Math. Gen. 35 (2002) 9971–10013.

[15] L.B. Kish, Thermal noise driven computing, Applied Physics Letters 89(14) (2006) 144104–1–3.

[16] K. Laurio, F. Linaker, A. Narayanan, Regular biosequence pattern matching with cellular automata, Inf. Sci. 146 (2002) 89-101.

[17] J. Lee, F. Peper, On Brownian cellular automata, In: A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G.J. Martinez, K. Morita, T. Worsch (Eds.), Theory and Application of Cellular Automata, Luniver Press, pp. 278–291, 2008.

[18] J. Lee, S. Adachi, F. Peper, K. Morita, Asynchronous Game of Life, Physica D 194 (2004) 369–384.

[19] J. Lee, S. Adachi, F. Peper, S. Mashiko, Delay-insensitive computation in asynchronous cellular automata, J. Comput. System Sci. 70 (2005) 201–220.

[20] J. Lee, F. Peper, M. Naruse, M. Ohtsu, T. Kawazoe, S. Cotofana, Y. Takahashi, L. Kish, T. Kubota, Brownian circuits: designs, In preparation.

[21] J. Lee, A simple model of asynchronous cellular automata exploiting fluctuation, J. Cell. Autom., in press.

[22] C.S. Lent, P.D. Tougaw, W. Pored, G.H. Bernstein, Quantum cellular automata, Nanotechnology 4 (1993) 49–57.

[23] J.C. Lusth, B. Dixon, A characterization of important algorithms for quantum-dot cellular automata, J. Inf. Sci. 113 (1999) 193–204.

[24] P. Maji, P.P. Chaudhuri, Non-uniform cellular automata based associative memory: Evolutionary design and basins of attraction, Inf. Sci. 178 (2008) 2315–2336.

[25] N. Margolus, Physics-like models of computation, Physica D 10 (1984) 81–95.

[26] A. Moreira, Universality and decidability of number-conserving cellular automata, Theor. Comput. Sci. 292 (2003) 711–721.

[27] A. Moreira, N. Boccara, E. Goles, On conservative and monotone one-dimensional cellular automata and their particle representation, Theor. Comput. Sci. 325 (2004) 285–316.

[28] K. Morita, Reversible computing and cellular automata - A survey, Theor. Comput. Sci. 395(1) (2008) 101–131.

[29] K. Nagel, Particle hopping models and traffic flow theory, Phys. Rev. E 53 (1996) 4655–4672.

[30] K. Nakamura, Synchronous to asynchronous transformation of polyautomata, J. Comput. System Sci. 23 (1981) 22–37.

[31] P. Patra, D.S. Fussell, Conservative delay-insensitive circuits, Workshop on Physics and Computation (1996) 248–259.

[32] F. Peper, J. Lee, S. Adachi, S. Mashiko, Laying out circuits on asynchronous cellular arrays: a step towards feasible nanocomputers? Nanotechnology 14 (4) (2003) 469–485.

[33] F. Peper, J. Lee, F. Abo, T. Isokawa, S. Adachi, N. Matsui, S. Mashiko, Fault-tolerance in nanocomputers: a cellular array approach, IEEE Trans. Nanotechnol. 3 (1) (2004) 187–201.

[34] F. Peper, J. Lee, J. Carmona, J. Cortadella, K. Morita, Brownian circuits: fundamentals, Submitted, 2011.

[35] F. Peper, J. Lee, T. Isokawa, Brownian cellular automata, J. Cell. Autom. 5(3) (2010) 185–206.

[36] L. Priese, Automata and concurrency, Theor. Comp. Sci. 25 (1983) 221–265.

[37] S. Saffiruddin, S.D. Cotofana, F. Peper, J. Lee, Building blocks for fluctuation based calculation in single electron tunneling technology. Proc. 8th IEEE Conf. Nanotechnology (Nano'08), TX, 358–361, 2008.

[38] B. Schönfisch, A. de Roos, Synchronous and asynchronous updating in cellular automata, BioSystems 51 (1999) 123–143.

[39] N. Tanimoto, K. Imai, A characterization of von Neumann neighbor number-conserving cellular automata, J. Cell. Autom. 4(1) (2009) 39–54.

[40] T. Yanagida, M. Ueda, T. Murata, S. Esaki, Y. Ishii, Brownian motion, fluctuation and life, Biosystems 88(3) (2007) 228–242.

[41] Y. Zhai, Z. Yi, P.M. Deng, On behavior of two-dimensional cellular automata with an exceptional rule, Inf. Sci. 179 (2009) 613-622.

[42] B. Zhang, J.C. Lusth, Computing with random quantum dot repulsion, Inf. Sci. 178 (2008) 1519–1532.