Simplifying Brownian Cellular Automata: Two States and an Average of Two Rules per Cell

Ferdinand Peper
Brain ICT Laboratory
National Institute of Information and Communications Technology (NICT)
588-2 Iwaoka, Iwaoka-cho, Nishi-ku
Kobe, 651-2492, Japan
Email: peper@nict.go.jp

Abstract—Brownian Cellular Automata (BCA) are asynchronous cellular automata in which local configurations are allowed to fluctuate in cell space. Used to drive the operations of BCA, these fluctuations facilitate a drastic reduction in the complexity of cells, with the best BCA models achieved up to now having three states and two transition rules. This paper proposes a further reduction to two states per cell, by employing cells of three different types. These three types require respectively three, two, and zero transition rules per cell. This result may bring physical implementations of BCA one step closer.

Keywords—Cellular Automata, Fluctuations, Computational Universality.

I. INTRODUCTION

Cellular Automata (CA) have been investigated for more than half a century as models in which simple cells available in large numbers and arranged uniformly in a cell space are able to conduct complex operations, such as computation and self-replication of patterns. There have also been proposals [1], [2], [3], especially in recent years, of highly distributed computers based on physical implementations of CA. The regular structure of CA make them very suitable for manufacturing based on bottom-up methods, like molecular self-assembly, and it is with this in mind that various proposals for CA-based nanocomputers have been made.

One obstacle to the realization of such nanocomputers is the relative complexity of cells. CA offer many advantages, like scalability and local connectivity, in architectures based on devices with feature sizes of a few nanometers, but notwithstanding these, a hardware overhead of a factor 10 to 100 can be expected as compared to conventional CMOS-based von Neumann architectures—a comparison favoring the latter provided of course that they can still be manufactured at such tiny scales. There is thus a big incentive to design CA with cells that are as simple as possible, while allowing a wide range of computations, preferably computations that are universal.

Brownian Cellular Automata (BCA) have been introduced with low complexity of cells in mind. Being asynchronously timed—i.e., lacking a clock according to which all cells are updated synchronously—they allow truly localized operation. Each cell operates according to its own timing, with only local conditions to take care of, unlike in conventional synchronous CA. Moreover, BCA tend to require less complexity in each cell, as compared to Asynchronous Cellular Automata (ACA). Though both of these models are timed asynchronously, there is a subtle difference between them: in BCA there is an active effort to exploit fluctuations of local configurations. BCA use fluctuations of signals to explore the topologies of circuits laid out on cell space by a random search that finds a computational path between inputs and outputs. As in Brownian Circuits [4], [5], BCA thus use fluctuations as a computational resource, without which they would be unable to operate. This computational resource comes in the place of wires, states, of whatever elements that add complexity to a model, like cell states or transition rules in the case of CA. Transition rules add an important degree of complexity to a CA model, especially if large numbers of them are required, though traditionally this factor has been considered irrelevant to the validity of models. However, when physically implementing CA whereby each cell is controlled locally, the number of transition rules becomes important, because they need to be implemented in each cell. BCA facilitate cells with low complexity: in their initial proposal [5], BCA use cells with three states and three transition rules. Later, this was improved to three states and two rules in [6], [7].

This paper proposes a new model of BCA that requires only two states to achieve computational universality. This result is achieved by employing a non-uniform cell space, in which cells are divided in three types: two active types, with respectively three and two transition rules per cell, and one passive type lacking any transition rules. This allows each cell to have a somewhat specialized function, which limits not only the number of states, but also the number of rules.

This paper is organized as follows. Section II gives preliminaries on Brownian circuits, such as primitive circuit elements for them. Section III describes the new BCA model: each primitive circuit element is implemented on the cell space, and wiring elements, such as curves and crossings, are described to connect these circuit elements. This paper finishes with conclusions and a discussion in Section IV.

II. PRELIMINARIES: BROWNIAN CIRCUITS

Brownian circuits are circuits in which signals take the form of discrete (particle-like) tokens that fluctuate forward...
and backward on wires and other circuit elements. These fluctuations are used to drive tokens through a Brownian circuit from input to output according to a random search, as if finding their way through a maze. Like many token-based circuits, Brownian circuits are \textit{Delay-Insensitive}, which means that any delay of tokens will not influence the correctness of operations. Delay-insensitive circuits are not governed by a clock, so they are \textit{asynchronous}.

Due to the lack of a clock, circuit elements employed in Delay-Insensitive circuits are different from those in synchronous circuits. Synchronization functionality, which is accomplished in conventional logic circuits through clocking action, needs to be explicitly implemented in Delay-Insensitive circuits. \textit{Join-functionality} is the usual name for this type of action. The circuit elements of Delay-Insensitive circuits tend to be more complex than the elements usually employed in synchronous circuits, like \textit{AND} and \textit{NOT}, due to the required added functionality. Brownian circuits allow for simpler circuit elements, however, because they exploit fluctuations as a resource for searching in a circuit. This (or equivalent) functionality would otherwise need to be implemented through additional states and input and output wires of circuit elements. The reduced complexity of Brownian circuits as compared to conventional Delay-Insensitive circuits is the reason why BCA can be implemented with less states and less transition rules as compared to conventional ACA.

This paper uses the primitive circuit elements proposed in [5] to construct BCA based on Brownian circuits. These elements are the \textit{Hub}, the \textit{Conservative Join (CJoin)}, and the \textit{Ratchet}, and they form a universal set of circuit primitives. That means that any computable function can be realized based on a circuit realized from these circuit primitives.

The Hub (Fig. 1) is an element with three wires, all of which act as both input and output. Tokens entering one of the wires fluctuate on and between the wires, and may exit any of the wires.

The CJoin (Fig. 2) is an element with four wires, two of which are used for input and two for output. Once two tokens are available as inputs, the CJoin moves them in a pairwise way to the outputs. Due to fluctuations, the operation may also reverse. The CJoin requires both input tokens to be present before it undergoes an operation. If only one input token is available, it will remain on its wire, until another input token becomes available, or until the token fluctuates away toward another part of the circuit.

The Ratchet (Fig. 3) is placed on a wire with the purpose to impose a direction on the flow of tokens on that wire. A \textit{hard ratchet} prevents tokens to go back through the ratchet, and a \textit{soft ratchet} merely biases tokens in one direction with a certain probability. This paper only employs hard ratchets. Any wire terminal of the above Brownian circuit elements can be connected with any other wire terminal in principle, though whether a certain interconnection scheme is useful is a different matter, and it generally depends on the particular design. This paper will not dwell on designs of Brownian circuits or their embedding on the cell space. Rather, the implementations of the above circuit elements on the cell space will be shown, as well as the valid ways to connect them to each other. Designs in other papers (e.g. [5]) may be used to inspire the construction of more complex computing structures on the cell space of the proposed BCA.

### III. The Brownian Cellular Automaton Model

The cell space of the model consists of three types of cells, called \textit{A-cells}, \textit{B-cells}, and \textit{X-cells}, respectively (Fig. 4). Each type of cells has two states: state 0, indicated by white (apart from the cell type-indicating features), and state 1, indicated by grey. Each cell has a von Neumann-neighborhood, which consists of the cells at orthogonal distance 1 from the cell. Cells are updated one by one, randomly picked, like in [5]. A signal on the cell space is presented as a cell in state 0 against a background of some selected cells in state 1. The transition rules for the cells are given in Table I. All rules are rotation-symmetric as well as reflection-symmetric, meaning that any

---

**Fig. 1.** Hub and its possible transitions on a token, denoted by a black blob. Fluctuations cause a token to move between the Hub’s three wires \(W_1, W_2, \) and \(W_3\) in any order.

**Fig. 2.** CJoin and its possible transitions on tokens. If there is a token on only one input wire \((I_1 \ or \ I_2)\), this token remains pending until a signal arrives on the other wire. Input of one token on each of the input wires \(I_1\) and \(I_2\) will result in one token on each of the two output wires \(O_1\) and \(O_2\). The CJoin operation may also be reversed.

**Fig. 3.** Ratchet and its possible transition. The token on the wire \(W\) may fluctuate before the ratchet as well as after the ratchet, but once it moves over the ratchet it cannot return. The ratchet thus imposes a direction on a (originally) bi-directional wire.
Fig. 4. Cell space of the proposed Brownian Cellular Automaton (BCA), consisting of three types of cells. Cells without any features are A-cells, cells with a blob are B-cells, and cells with an X in them are X-cells.

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="" /></td>
<td><img src="image2" alt="" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3" alt="" /></td>
<td><img src="image4" alt="" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image5" alt="" /></td>
<td><img src="image6" alt="" /></td>
</tr>
</tbody>
</table>

Fig. 5. Wire on the Brownian Cellular Automaton (cells in state 1), with a signal on it (cell in state 0 with state-1 cells on two sides). The signal moves forward and backward due to transition rules 1A and 1B, as indicated by the labels on the arrows.

Fig. 6. Curve on the Brownian Cellular Automaton, with a signal on it. After the signal is brought to the terminal of the curve by rule 1B, it moves through the curve through rule 2A. After passing the curve, the signal moves away due to rule 1A (not pictured). The signal can also move backward due to the same rule 2A, so signal fluctuations are also possible in the curve.

Fig. 7. Hub on the Brownian Cellular Automaton, with a signal on it. In a similar way as on the curve a signal fluctuates on the Hub through rule 2A.

Fig. 8. Ratchet on the Brownian Cellular Automaton, with a signal on it. The signal moves through the Ratchet through rule 2A, followed by rule 1A (transition indicated by arrow in the center). Once rule 1A is applied, there is no way back.

Fig. 9. CJoin on the Brownian Cellular Automaton, with two signals switched by it. After the two signals are brought to the terminals of the CJoin by rule 1A, they are switched through rule 2B, after which they follow their course. Fluctuation of the two signals forward and backward through the CJoin is also possible through rule 2B.

Fig. 10. Signal crossing on the Brownian Cellular Automaton, with a signal crossing another signal. This is facilitated by the natural behavior of fluctuations, as demonstrated in the BCA in [5].

Table I

**Transition rules of the Brownian Cellular Automaton.** The left column shows the three rules for A-cells, and the right column the two rules for B-cells. X-cells have no transition rules. Rules 1A and 1B are used for signal propagation. Rule 2A makes signals jump one cell and is used to implement curves, hubs, and signal crossings. Rule 2B switches a pair of signals, and is used to implement CJoins. Rule 3A moves a state-1 marker on an X-cell to another X-cell and is used to facilitate signal crossings. X-cells have no transition rules.

rotation over a multiple of 90 degrees or any reflection of the transition rules are also transition rules.

The circuit elements are now implemented on the cell space in the following way. First, a wire is a linear structure of alternating A-cells and B-cells in state 1. Fig. 5 shows a wire with a signal on it, as well as transitions that move the signal in one or the other direction. Fluctuation of the signal on the wire follow naturally from rules 1A and 1B, which allow interactions into both directions of the wire.

For circuits to be laid out on the two-dimensional cell space, wires need to run in all four directions, necessitating a curve structure to connect them (Fig. 6). The Hub (Fig. 7) is implemented in a similar way as the curve.

The Ratchet uses rule 2A in combination with rule 1A (Fig. 8). After application of rule 1A, the signal is unable to return to its previous location by any of the rules, so effectively the structure acts like a Ratchet.

The CJoin is based on rule 2B, which switches two signals on orthogonal wires to their mirror image (Fig. 9).

There is one circuit element remaining, and that is the signal crossing. Most circuit designs require crossings of signals, since only few designs can be transformed into planar structures. Crossing signals is a somewhat difficult operation in asynchronous CA, since it requires a way to arbitrate a shared resource (the crossing) between two competing processes (the signals). The BCA in [5] solves this problem in a simple way through the natural arbitration behavior of fluctuations, but unfortunately this method cannot be applied in an exactly
Proc. of macro-cells, each containing two A-cells, one B-cell, and per cell is limited. It is calculated as the sum of the products over the cell types, but on average the number of rules required cause unwanted interactions. This allows cells to contain only those rules that are strictly necessary for their operation, and by employing three different types of cells in the cell space.

IV. CONCLUSIONS AND DISCUSSION

This paper describes a new Brownian Cellular Automaton model that achieves a decreased number (two) of cell states, by employing three different types of cells in the cell space. This allows cells to contain only those rules that are strictly necessary for their operation, and excludes rules that would cause unwanted interactions.

The number of rules required per cell varies substantially over the cell types, but on average the number of rules required per cell is limited. It is calculated as the sum of the products of the number of rules per cell type and the frequency of occurrence of those types in the cell space. For the BCA model proposed in this paper, this computes to 3 rules for A-cells times their frequency of 1/2 plus 2 rules for B-cells times their frequency of 1/4 plus 0 rules for X-cells, giving a total of 2 rules on average per cell. This number can likely be improved, since this is just the first attempt of using a non-uniform cell model to construct BCA. Though non-uniform, the cell space can be reformulated in a uniform framework by dividing it into \(2 \times 2\) macro-cells, each containing two A-cells, one B-cell, and one X-cell, as well as five transition rules.

ACKNOWLEDGMENT

I would like to thank Prof. Ken’ichi Morita of Hiroshima University for all his encouragements and inspiration he has provided for my research on Cellular Automata over the years.

REFERENCES