Int. Journ. of Unconventional Computing, Vol. 12, pp. 341–362
Reprints available directly from the publisher
Photocopying permitted by license only

Brownian Circuits: Designs

JIA LEE[1], FERDINAND PEPER[2,3,4,5,*], SORIN D. COTOFANA[6], MAKOTO NARUSE[2], MOTOICHI OHTSU[7], TADASHI KAWAZOE[8], YASUO TAKAHASHI[9], TETSUYA SHIMOKAWA[2,3,5], LASZLO B. KISH[10] AND TOHRU KUBOTA[2]

[1]*College of Computer Science, ChongChing University, China*
[2]*National Institute of Information and Communications Technology, Japan*
[3]*Osaka University, Japan*
[4]*The University of Hyogo, Japan*
[5]*Kobe University, Japan*
[6]*Delft University of Technology, the Netherlands*
[7]*The University of Tokyo, Japan*
[8]*Nanophotonics Engineering Organization, Japan*
[9]*Graduate School of Information Science and Technology, Hokkaido University, Japan*
[10]*Department of Electrical and Computer Engineering, Texas A&M University, College Station, USA.*

Received: May 2, 2016. Accepted: May 26, 2016.

The ongoing miniaturization of electronic circuits will eventually lead to signals consisting of only a few particles or molecules, but fluctuations will be a major interference in the operation of such circuits. Brownian circuits have been shown to exploit fluctuations by finding computational paths in circuits through a random search mechanism. This paper discusses Brownian circuits with decreased complexity, and shows designs of circuits with functionalities like counting, testing of conditional statements, memory, and arbitration of shared resources. We also discuss the potential of Brownian circuits for implementations by Single Electron Tunneling technology.

Keywords: Fluctuation-driven computation, Brownian motion, circuit designs, nanocomputing

1 INTRODUCTION

Brownian circuits employ fluctuating tokens as signals to search randomly for computational paths in the state space defined by the circuit topology.

* Contact author: E-mail: peper@nict.go.jp

Fluctuations are not just a nuisance factor in such circuits, rather they are actively exploited in assisting circuit operations. Brownian circuits provide an alternative to more conventional strategies to deal with noise, which usually amounts to suppressing noise or employing error correction. While such methods work when signal levels are well above noise by a large factor, they fall short in a regime near the thermal limit [15], where the energy to operate a switch barely exceeds that of thermal noise. This issue has become more serious with the increased integration densities of microelectronics, which will eventually lead to circuits employing signals that consist of only a small number of particles. Fluctuations in such circuits will be difficult to avoid [18].

One of the first proposals to use Brownian motion of signals in computation originates with Bennett [4]. It takes the form of a miniature mechanical Turing machine, in which signals move around randomly, driven by thermal noise, and searching their way through the machine's circuit topology. Later proposals have employed fluctuations with the eye of making a tradeoff between energy use and reliability [8, 19], but these approaches tend to require extensive error correction, and may thus fall in the realm of more-or-less traditional methods. Noise and fluctuations have also been used in the simulated annealing process of a Boltzmann machine [26]. Based on Single Electron Tunneling devices, this architecture has been shown by computer simulations to be able to exploit signal fluctuations to search in an energy landscape. This proposal revolves around a neural network that conducts optimization, but it is not suitable for arithmetic computation.

In biological systems, noise and fluctuations play an important role in facilitating transitions between energy states that are separated by a high energy barrier. Whereas designers of engineered systems usually aim to achieve high S/N ratios, biological systems differ in that they have evolved to take advantage of noise and fluctuations, allowing high energy efficiencies, like in rotary protein motors, which work at efficiencies close to 100 percent [7]. Fluctuations are also used in a biological system to exert effective control on molecular scales through a trial-and-error mechanism that employs stochastic search to make the system converge to a desirable state. This phenomenon is called *Brownian search*.

Brownian circuits use a similar mechanism to conduct their operations in a controlled way. This is formalized in [23] through the definition of a circuit element—*the T-element*—that has been proven to be universal for a class of circuits called *Token-Pass circuits*, under the necessary and sufficient condition that tokens undergo fluctuations. Key to the power of token fluctuations is their ability to backtrack out of deadlocks, which are a common problem in token-based circuits [23]. Token-Pass circuits, along with the T-element, form a suitable framework to formally prove the capability of fluctuations, but they have a somewhat rigid structure, being basically a set of lines that can interact

with each other at certain locations in very restricted ways. This poses the question whether more efficient—in terms of hardware and time resources—constructions are possible if more flexibility is allowed in the definition of circuits.

The pay-off of such simplified constructions lies in the potential for efficient designs in physical implementations. The omnipresence of fluctuations at nanometer scales, combined with the trend in microelectronics of signals consisting of less and less particles, underlines the importance of the formulation of such abstract circuit models.

This paper describes circuits that are based on elements that are less complex than the T-element, yet can be used as primitives of a universal class of Brownian circuits. Called *Hub*, *Conservative Join (CJoin)*, and *Ratchet* (see Section 2), these elements have three, four, and two input and output lines, respectively, which is less than the six lines of the T-element [23]. Token fluctuations are fundamental to operations of circuits based on the Hub, the CJoin, and the Ratchet, but compared to the Brownian circuits in [23], circuits tend to become more straightforward in their designs. We introduce designs of some standard circuits, such as a 1-bit memory cell and a Half-Adder (Section 3), and an arbitration operator for shared resources (Section 4), and combine these into more complex functionalities (Section 5). We also discuss six conditions that are important for the realization of Brownian circuits in terms of physical implementations (Section 6). This is followed by a brief review of Single Electron Tunneling technology designs based on Brownian circuits (Section 7). The discreteness of single electrons, as well as the stochastic nature of electron tunneling, both fit well in the framework of Brownian circuits. Other technologies based on different mechanisms [16], however, may also be suitable for physical implementations, provided they employ signals that have a discrete character and the tendency to undergo fluctuations. This paper finishes with conclusions and a discussion in Section 8.

2 BROWNIAN CIRCUITS

A Brownian circuit is a token-based circuit, in which tokens are allowed to fluctuate forwards and backwards on lines and across operators. Notwithstanding their ability to fluctuate, tokens are bound by the transition rules governing circuits. An operator that requires a particular combination of inputs will not produce outputs unless that combination is available on the operator's input lines at a certain time instant. For every other combination the tokens will remain fluctuating on the input lines, where they may again serve as input to the operator at some future time. The use of fluctuations has an important merit: it allows Brownian circuits to backtrack out of states

in which no forward computational path exists [23]. *Deadlocks* are the usual term for such situations, and they are likely to arise in conventional (non-Brownian) token-based circuits when a circuit element—or, *module* in our terminology—requires at least two tokens as input. Absent the full number of input tokens, the input to a module will stay pending, waiting for the remaining input tokens to arrive, and when that does not happen, a deadlock results. The usual way out of a deadlock is to reroute pending tokens to alternative locations in the circuit, where they can be processed without deadlocks. Given that tokens can only move forward in conventional token-based circuits, however, this requires additional pathways and additional control mechanisms, which increases the complexities of modules and inter-module connection patterns. Brownian circuits do not require these additional resources, since they use fluctuations to make tokens backtrack their way out of deadlocks, and in the process reach other modules that can accept them.

The Brownian circuits in [23] are very useful for the formal analysis of fluctuations in a computational framework, but they have some overhead in terms of interconnection lines, making them less suitable for efficient implementations. This motivates the definition of a class of circuits facing less restrictions, the so-called *Conservative Delay-Insensitive (CDI) circuits* [22]. Promising for their physical plausability, these circuits are robust to delays of tokens, and they conserve tokens in operations, i.e., the number of input tokens to a module equals the number of output tokens.

A universal set of primitive modules for CDI circuits consists of the *Merge* and the 2×2-*CJoin*. Figure 1 shows these modules, together with their functionalities, expressed in terms of Petri nets (see also [23]). The universality of this set of primitive modules can be shown by constructing a so-called $n \times m$-CJoin from it according to the design in [22]. The $n \times m$-CJoin is a generalization of a 2×2-CJoin and it is possible to construct a Finite State Machine (FSM) from it with n states and m inputs (e.g. see [11, 21]). Since any logic circuit can be expressed in terms of an FSM, the universality of the set consisting of the Merge and the 2×2-CJoin follows (see also [6]).

When CDI circuits have fluctuating tokens, they are Brownian. They can be constructed from a set of three primitive modules. The first module is the *Hub*, which has three lines that are bidirectional (Figure 2). There is at most one token at a time on any of the Hub's lines, and this token can move to any of the lines due to its fluctuations.

The second module is the *Conservative Join (CJoin)*, which has two input lines and two output lines (Figure 3). The two tokens on its input lines (one token on each line) pass through the CJoin in a pairwise manner, so the CJoin in fact behaves like a synchronizer. Tokens may fluctuate on the input lines, and when processed by the CJoin, they move to the output lines where they may also fluctuate. The operation of the CJoin may also be reversed, and the

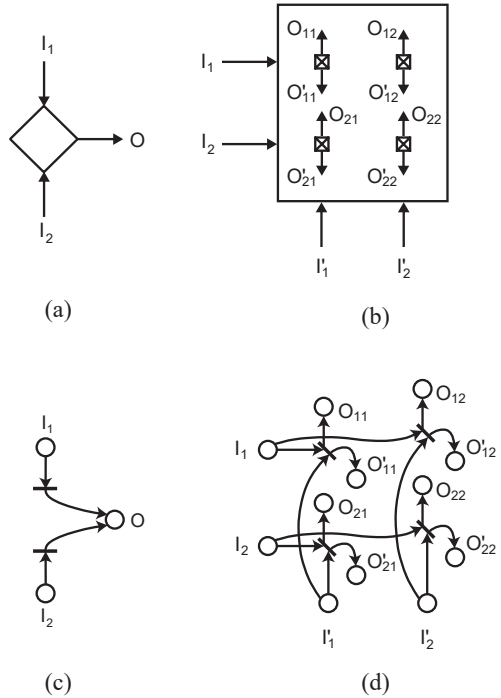

FIGURE 1
Primitive elements for Conservative Delay-Insensitive (CDI) circuits, and the corresponding Petri nets. (a) Merge merges two streams of input tokens I_1 and I_2 into one output stream O. (b) 2×2-CJoin joins two input tokens resulting in two output tokens as follows. Upon receiving one token from line I_i and one token from line I'_j $(i, j \in \{1, 2\})$, the module outputs one token to each of the lines O_{ij} and O'_{ij}. If there is only one token input to the module, it remains pending until a second token is input to the module. (c) Petri net of the Merge, and (d) of the 2×2-CJoin.

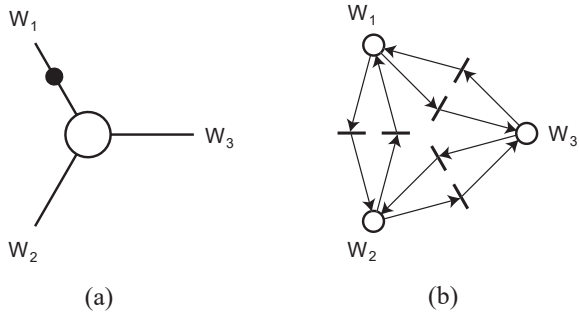

FIGURE 2
(a) The Hub, with a token on one of its lines (W_1), denoted by a black blob. All lines are bidirectional and are indicated without arrow heads. The token can be on any of the three lines W_1, W_2, and W_3, and it can fluctuate between the lines in any order. (b) Petri net of the Hub.

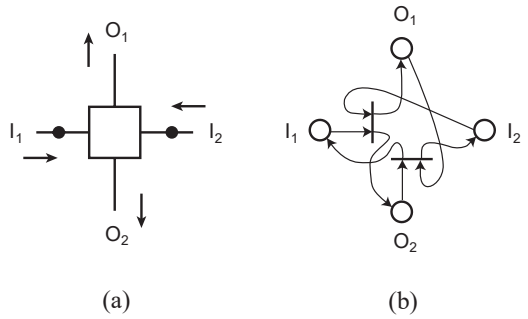

FIGURE 3
(a) The CJoin, with one token on each of its two input lines I_1 and I_2. A transition moves the tokens from the input lines to the two output lines O_1 and O_2, or in the reverse direction if both tokens are on the output lines. If there is a token on only one input line (I_1 or I_2), this token remains pending (and fluctuating) until a token arrives on the other input line. All lines are bidirectional and lack arrow heads, but since there is a bias from input to output, there are small arrows to indicate the preferred direction of token flow. (b) Petri net of the CJoin.

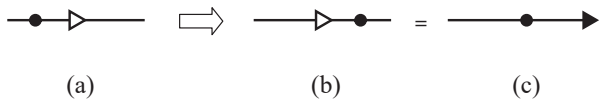

FIGURE 4
Ratchet and its possible transition. (a) The token on the line may fluctuate before the Ratchet as well as after the Ratchet, but (b) once it is at the right side of the Ratchet it cannot return. The Ratchet thus imposes a direction on an (originally bidirectional) line. (c) The resulting unidirectional line is denoted by an arrow.

forward / backward movement of the two tokens through it may be repeated an unlimited number of times. Due to this bidirectionality, there is strictly spoken no difference between input and output lines of the CJoin, though we still use the terminology of input and output, since the direction of the process is eventually biased forward. We call this the *preferred* direction of the CJoin or of the associated lines.

The third module is the *Ratchet*, which allows a token to freely pass through in one direction, but blocks it in the opposite direction (Figure 4). Thus, a bidirectional line with a Ratchet on it effectively becomes unidirectional. Ratchets can be used to limit the searching behavior of a circuit at selected points, as a result of which the circuit is sped up. However, the placement of Ratchets should be carefully considered to avoid the creation of deadlocks at locations where Brownian search is required.

To show that the set consisting of the Hub, CJoin, and Ratchet is universal for the class of CDI circuits, we construct a Merge and a 2×2-CJoin from

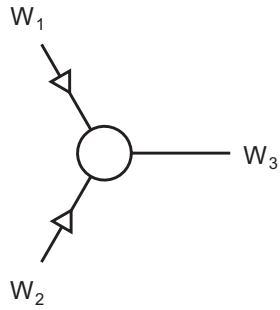

FIGURE 5
The Merge constructed from a Hub and two Ratchets. The Ratchets are used to guarantee that any token input to the Merge will always be output to line W_3, and never to either of the input lines W_1 and W_2.

them. Since the Merge and the 2×2-CJoin have only unidirectional input and output lines, Ratchets are used to mimic this characteristic. The Merge is constructed from a Hub and two Ratchets, whereby two of the Hub's bidirectional lines are equipped with Ratchets to act as input lines for the Merge, and the remaining bidirectional line of the Hub is used for output of the Merge (Figure 5). The construction of the 2×2-CJoin requires four Hubs, four CJoins, and twelve Ratchets. The more general construction of the $n \times m$-CJoin is given in Figure 6. Brownian search in this construction takes place at the trees forming the input lines of the individual CJoin modules.

Universality of the set consisting of the Hub, CJoin, and Ratchet can also be shown in a different way, i.e., by constructing a so-called Conservative Tria (CTria) from these modules [12], but we employ the construction based on the $n \times m$-CJoin, because it results in more straightforward and efficient designs.

3 CIRCUIT DESIGNS

The universality of the Hub, CJoin, and Ratchet set opens the way for constructions of circuits like a Half-Adder and a 1-bit memory. The basic idea behind these constructions is to use the CJoins in an $n \times m$-CJoin as minterms in a canonical form of a Boolean expression, and to use the Hubs for summing the minterms into the desired results, following similar ideas as in [23]. A Half-Adder, for example, is constructed from one 2×2-CJoin and four Hubs (Figure 7). The inputs of the Half-Adder—represented by the 0-line and the 1-line at the top and a similar pair of lines at the right according to a dual-rail encoding scheme ([13], Chapter 7)—are fed to the four CJoins,

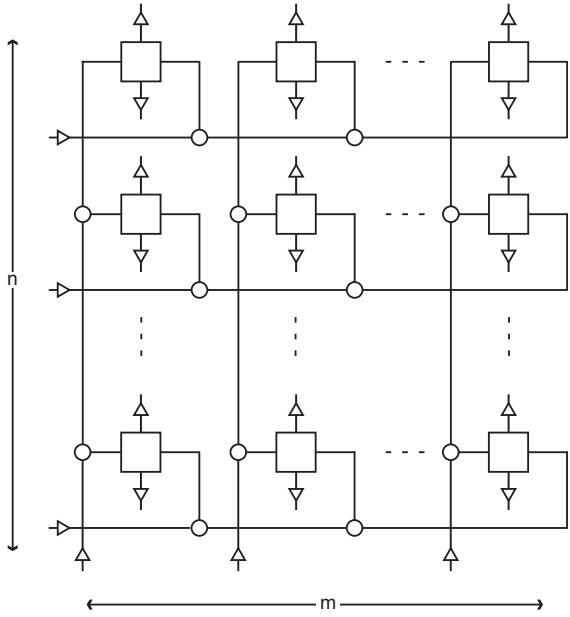

FIGURE 6
The $n \times m$-CJoin constructed as an array of CJoin modules with n rows and m columns. Each row contains—apart from CJoin modules—a total of $m - 1$ Hubs that allow a token to conduct a Brownian search in the row of m CJoins. Similarly, each column contains $n - 1$ Hubs for searching in the column of n CJoins. This gives a total of $n(m - 1) + (n - 1)m = 2nm - n - m$ Hubs. When one token is received from the input line at row i and one token from the input line at column j, the tokens find their way through Brownian search to the CJoin in row i and column j. After accepting these tokens, this CJoin produces one token at each of its two corresponding output lines. Ratchets are placed at locations where Brownian search is not necessary, which in this case is at the output sides of the CJoins. A total of $2mn + m + n$ Ratchets is used.

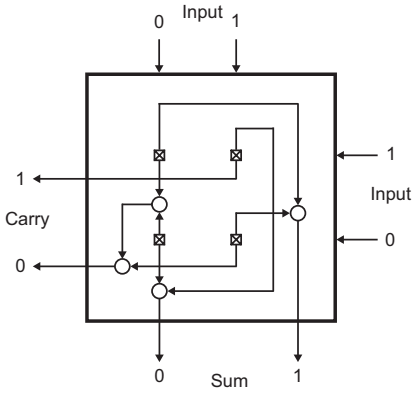

FIGURE 7
Half-Adder constructed from a 2×2-CJoin and four Hubs.

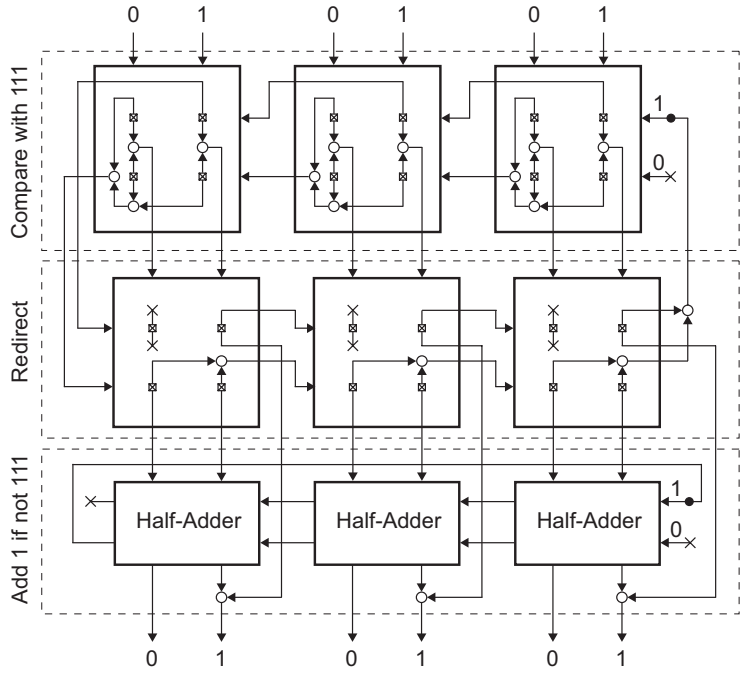

FIGURE 8
3-bit Conditional Counter. The 3-bit input received by the Counter is compared with the bit-string '111' in the top stage. Depending on the outcome of the comparison, the second stage either redirects the input to the output (when the input equals '111'), or to the third stage to increase it by one before being output. Due to the comparator in the first stage, there will never be overflow in the third stage, so the carry bit 1 in the most-significant Half-Adder is not connected to other modules.

and the resulting four min-terms are combined through the Hubs to produce the Half-Adder's sum at the bottom and the carry at the left.

Figure 8 combines three Half-Adders into a 3-bit Conditional Counter that adds 1 to its input in an operation, provided the input is less than the maximal value '111'. The first stage of this counter compares the input with the bit string '111' and if the input equals this string, it is redirected toward the output; otherwise it is increased by one by the Half-Adders in the bottom stage before the results are output.

Another useful circuit is a 1-bit memory, which requires a 2×3-CJoin, as shown in Figure 9(a). The construction principle is similar to that of the Half-Adder, with minterms produced by the six CJoins in the 2×3-CJoin being summed by six Hubs. This construction involves feedback connections to store the memory state, and Ratchets are used to retain the state. A 1-bit memory can also be constructed from two 2×2-CJoins, like in Figure 9(b).

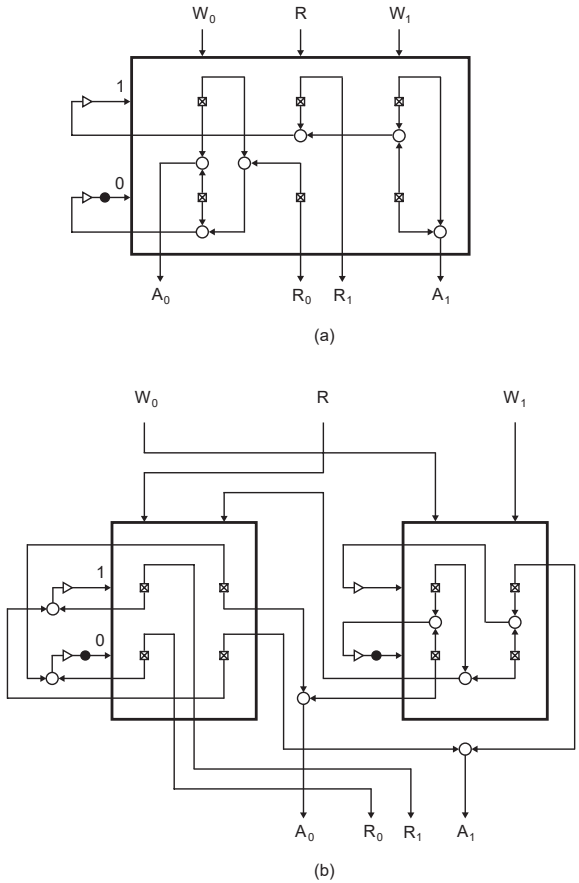

FIGURE 9
(a) One-bit memory constructed from a 2×3-CJoin and six Hubs. The state of the memory is stored by one token residing on one of the input lines at the left, whereby a token on the lower line (like in the figure) denotes the state 0 and a token on the upper line denotes the state 1. The value 0 resp. 1 is written into the memory by putting a token on the line W_0 resp. W_1. After the required value is written, a token is output to the corresponding acknowledge line A_0 or A_1. Reading from the memory is done by inputing a token to line R, which results in a token output to line R_0 or line R_1, depending on the memory's state. (b) One-bit memory constructed from two 2×2-CJoins and seven Hubs. For technical reasons, the memory's state is stored by two tokens in parallel.

4 ARBITRATION

When there are parallel processes competing for a shared resource, the assignment of that resource is accomplished through *arbitration*. The circuits in the previous section all work fine in the absence of arbitration, since

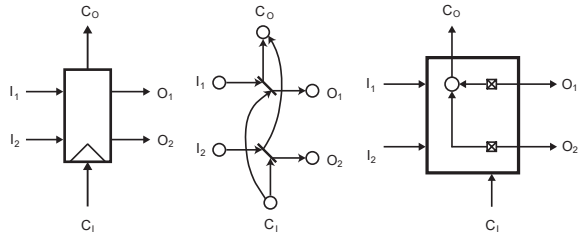

FIGURE 10
(a) CSequencer facilitates arbitration of shared resources between parallel processes. An input token on line I_1 (resp. I_2) together with an input token on line C_I but without an input token on line I_2 (resp. I_1) results in one output token on line O_1 (resp. O_2) and one on line C_O. If there are input tokens on both I_1 and I_2 at the same time as well as an input token on C_I, then only one of the tokens on I_1 and I_2 (possibly chosen arbitrarily) is taken together with the token on C_I, resulting in an output token on the corresponding O_1 or O_2 line and on line C_O. The remaining input token may be processed at a later time, when a new token is available on line C_I. (b) Petri net of the CSequencer. (c) CSequencer constructed from a 2×1-CJoin.

no resource sharing is needed. A module capable of arbitration behavior is the *Conservative Sequencer (CSequencer)* in Figure 10(a), with functionality as defined by the Petri-net in Figure 10(b). This module is basically a 2×1-CJoin (Figure 10(c)) in which simultaneous input tokens to the two lines I_1 and I_2 are allowed, one of which passes to the corresponding output line either O_1 or O_2 respectively, if a token is input to line C_I. This behavior is a direct consequence of the searching process in the underlying Brownian circuit. The fluctuations of the token input to C_I drive a search inside the module to match it with a second token input from either line I_1 or line I_2. As long as the token from C_I fails to find a token to match with, no operations take place in the 2×1-CJoin, and the C_I token continues its search. This circuit is capable of arbitration as a result of the tokens being subject to fluctuations. Absent those, the circuit may end up in a deadlock.

The CSequencer comes in various formats. The version that arbitrates n processes is called the n-CSequencer. When $n = 2$, like above, the prefix is left out. The n-CSequencer can be implemented by an $n \times 1$-CJoin.

5 ARBITRATION-BASED CIRCUITS

When the Conditional Counter in Figure 8 is extended with memories to store the counted value, we obtain a circuit like in Figure 11. This circuit, which we call *Counting Memory*, includes functionality to count up or down by one, or read out the memory value. An attempt to conduct these operations at the

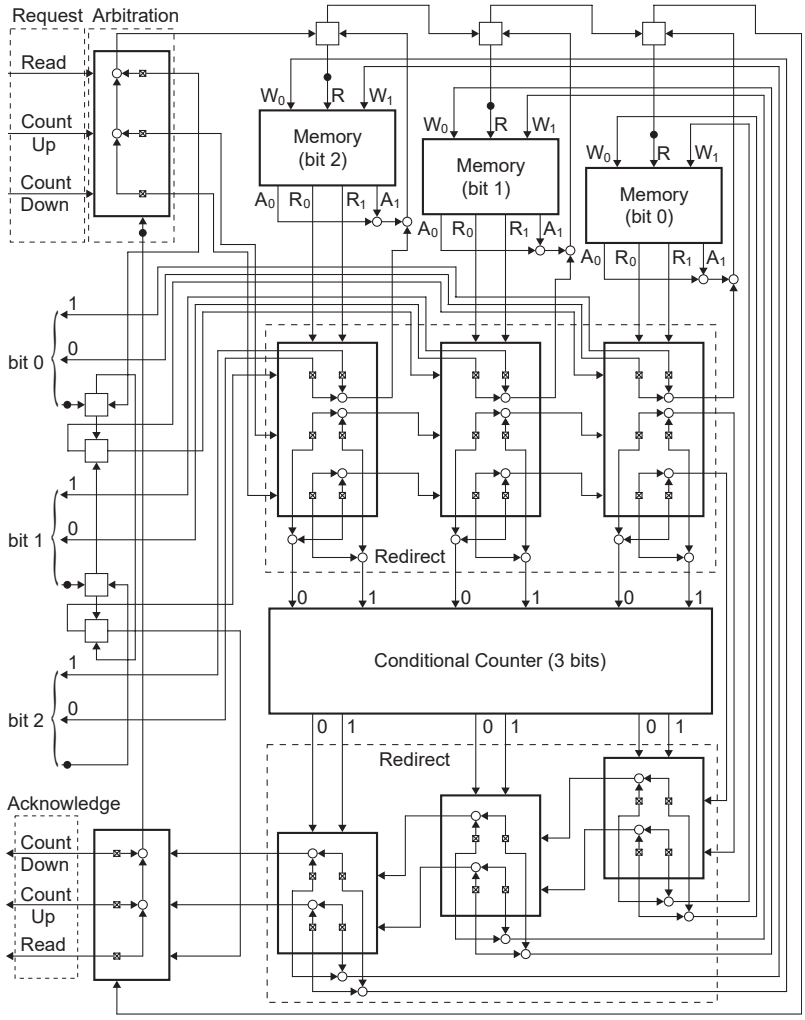

FIGURE 11
Counting Memory storing three bits, the value of which can be read out, or, alternatively, be increased by one up to a maximum of '111' or decreased by one down to a minimum of '000'. A token input to the Read line at the top left results in the memory's value being output to the pairs of 0- or 1-lines of bits 0, 1, and 2 at the center left. This operation requires one token to be input for each bit to ensure that the number of tokens is conserved (lowest line of each bit). A token input to the Count-Up line resp. Count-Down line results in the Counting Memory's value being increased resp. decreased by one. All three operations generate an acknowledge-token at the lower left.

same time in a non-arbitrating circuit would would give unpredictable results, but this is prevented by arbitrating simultaneous calls to the circuit through a 3-CSequencer (top left), which is implemented as a 3×1-CJoin.

The Counting Memory output is given in dual-rail encoded form through three pairs of 0- and 1-lines (bits 0, 1, and 2 at the center left of Figure 11). As a result of a reading operation, tokens are output to these lines so that they reflect the values stored in the 1-bit memories at the top of the figure. Since the circuit conserves tokens, any number of tokens it outputs should be compensated by input of the same number of tokens. This is the reason why each of the output bits has a single token accompanying it on the line below the bit's two output lines. These input tokens are absorbed by the circuit when a Read signal passes the arbitrator at the top left.

Increasing or decreasing the number stored in the memory is accomplished through the 3-bit Conditional Counter just below the center. While an increase in value is straightforward, a decrease is done by first redirecting the number's bits via the three modules directly above the Conditional Counter. These modules include functionality to convert the number represented by the bits into its negative equivalent in two's complement representation. This is followed by an operation of the Conditional Counter to increase the number by one, and then convert the result back by the three Redirect modules at the bottom. No output signals are generated by a counting operation, except for a token at an output line at the bottom left in the figure to acknowledge the end of the operation. Such an acknowledge signal is also generated at the end of a reading operation.

Figure 12 shows a circuit that calls the Counting Memory from three parallel threads. Though the Counting Memory in itself is able to arbitrate Read, Count-Up, and Count-Down calls, the three threads may coincidentally call one and the same operation at the same time, which may give undesirable results. Thread 1 and thread 3, for example, both have calls to read out the value of the Counting Memory, but only one such call can be handled at a time, requiring an additional level of arbitration. The modules at the bottom center of Figure 12 provide this arbitration functionality. For each type of operation—Read, Count-Up, or Count-Down—it allows only one call to be made at a time. Any other calls are delayed until the Counting Memory is available again.

The Read operation is the most complex since it requires input and output tokens of the bits to be routed from and to the thread that made the Read call. The part of the circuit in Figure 12 located just left of the Counting Memory is responsible for this. It collects a Read operation's tokens from threads, and decides from which thread it redirects tokens to the Counting Memory in accordance with the decision of the arbitration part at the bottom of the

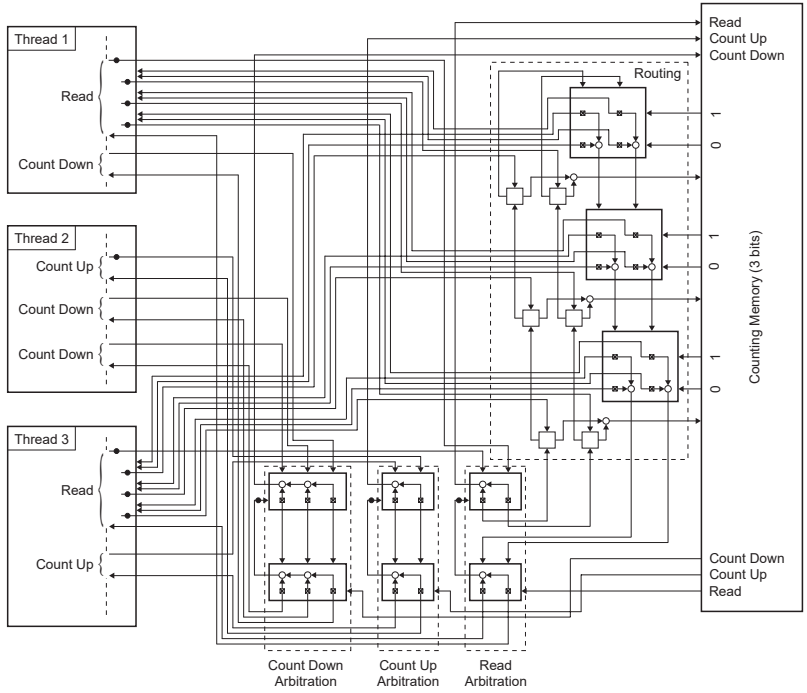

FIGURE 12
Arbitration of calls by three parallel threads to a 3-bit Counting Memory. Each thread is executed from top to bottom and when a call to the Counting Memory is made, the correponding tokens are submitted to the Counting Memory, but not before they pass arbitration circuitry at the bottom center. The part of the circuitry located just left of the Counting Memory is an extension of this arbitration circuitry, and it takes care for routing bits associated with the Read operation between the Counting Memory and the thread requesting the operation.

figure. Similarly, the output bits of the Read operation are routed back to the corresponding thread that made the call.

The lines in Figures 11 and 12 are all unidirectional, which allows Ratchets to be placed on them. The presence of Ratchets on these lines causes no problem, since no Brownian search needs to take place on them. All Brownian searching behavior is restricted to a local level within $n \times m$-CJoin modules. Ratchets placed outside these modules do not affect the Brownian search process inside modules. Consequently, the only penalties in time complexity likely to be incurred are those due to Brownian searching within $n \times m$-CJoin modules, but since the dimensions of these modules are small in our case, this overhead is limited.

Sections 3 to 5 have presented designs of elementary circuits, like counters and memories, as well as examples combining these circuits into more

complex circuits. In principle, circuits for every computable function can be constructed, since the basis set, consisting of the Hub, the CJoin, and the Ratchet, is universal for the class of CDI circuits. Arbitration in circuits is facilitated by Brownian search in the underlying Brownian circuits.

6 REALIZATIONS OF BROWNIAN CIRCUITS

For Brownian circuits to be realized by a technology or a model certain conditions must be met. This section discusses such conditions for circuits based on the Hub, CJoin, and Ratchet.

Token-Based Condition. Signals are represented as tokens: it is impossible to divide a signal into more signals, or to fuse multiple signals into one signal when on a line, i.e. outside a module. In practice this condition implies that signals behave like particles.

Line-Search Condition. A signal is able to explore the state space of a line by Brownian search based on fluctuations.

Hub-Search Condition. A signal is able to explore the state space of a Hub by Brownian search based on fluctuations. This condition is similar to the Line-Search condition, but then extended to three lines coming together in the Hub.

CJoin-Pair Condition. Two signals input to the CJoin (at two different lines), pass through the CJoin in a pairwise manner. If the two signals go back through the CJoin they do so in a pairwise manner too. This condition does not require signals to pass through the CJoin at the same time: it just means that when one signal passes through the CJoin, the other will also pass. Figure 13 shows a sequence of events in which signals pass pairwise, but not simultaneously.

In this case the CJoin has six states that are assumed in certain orders to ensure that the correct sequences from input to output are followed. After both signals have passed, the CJoin's state reverts to the initial state S_0, i.e., to the state before the signals were input. If the CJoin passes its two signals simultaneously, only one state is required.

Modularity Condition. The functionalities of the circuit elements (Hub, CJoin, or Ratchet) do not change when they are connected to each other. In other words, there should be no interference on the behavior inside a module from processes at the outside.

Ratchet Condition. Ratchets are to be placed at positions where no Brownian search is required. Feasible positions to place Ratchets are at output sides

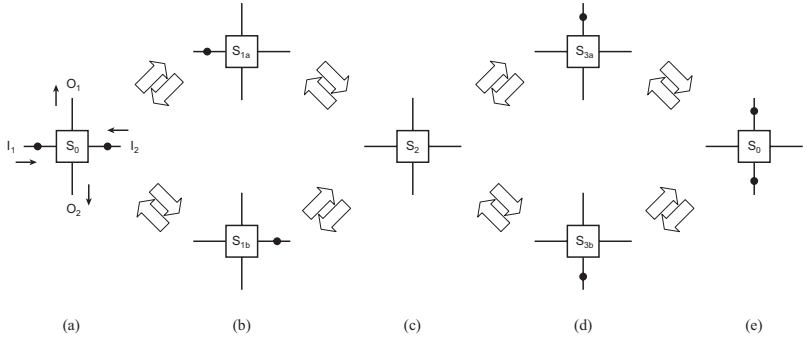

FIGURE 13
Sequences of states assumed by a CJoin when two signals are passed through it pairwise but not necessarily simultaneously. Transitions may take place both forward or in reverse. (a) CJoin is initially in state S_0, which indicates that no signals are being processed. The presence or absence of signals on the input and output lines is irrelevant to the state here. (b) Depending on which input signal is absorbed first, the CJoin will assume the state S_{1a} or S_{1b}. (c) After both input signals are accepted by the CJoin, the state S_2 is assumed. (d) Depending on which signal is output first, the CJoin will assume the state S_{3a} or S_{3b}. (e) After outputing both signals, the CJoin returns to its initial state S_0.

of CJoins. This tends to block signals from going backwards through CJoins, making their functionality forward-only. For the designs in this paper this does not impede the Brownian search process. In a search tree constructed from Hubs, Ratchets are usually placed at the root of the search tree to confine tokens to a minimal search space, like in the $n \times m$-CJoin in Figure 6.

7 SINGLE ELECTRON TUNNELING CIRCUITS

Single Electron Tunneling (SET) circuits use tunneling of electrons as the underlying operating mechanism. The fundamental element in a SET circuit is a *tunneling junction*, which is an thin layer with high electrical resistance separating two Coulomb islands. Tunneling through a junction becomes possible when the voltage V_j over the junction exceeds a critical voltage V_c that depends on the capacitance of the junction and of the remainder of the circuit. Tunneling is a stochastic phenomenon: the transport of an electron over the junction experiences a delay of

$$t_d = \frac{-\log P_{\text{err}} R_j q_e}{|V_j| - V_c},$$

where R_j is the junction resistance and P_{err} is the probability that no transport has occurred after t_d seconds. Though tunneling is extremely fast (in the order

of picoseconds), it is unpredictable, making it less suitable for synchronously timed architectures. For this reason, research has been initiated on how SET could be used in the framework of token-based delay-insensitive circuits [24]. SET technology is especially suited to the token-based nature of such circuits, with tokens representing electrons or vacancies of electrons.

Can Brownian circuits be implemented with SET technology? We will first examine the assumptions made in more conventional SET circuits, before addressing their potential to implement Brownian circuits. In order to be used as tokens in Brownian circuits, electrons need to have a token-based character, but their wave-particle duality implies that this only holds under certain conditions. An electron's wave function extends through a potential barrier, thus spreading the electron over the Coulomb islands at both sides of the barrier. If this effect was to prevail there would be no localized charges of electrons, rendering token-based computations impossible. The quantized nature of an electron can be made more prominent by making the tunneling resistance sufficiently high. This ensures that the charging energy dominates over the quantum charge fluctuations. In other words:

$$\frac{{q_e}^2}{2 \cdot C_j} \cdot R_j \cdot C_j \gg h \Longrightarrow R_j \gg h/{q_e}^2 = 25.8k\Omega$$

where h is Planck's constant, C_j is the tunnel capacitance and R_j is the tunneling resistance. The resistance of tunneling junctions in designs is usually chosen as 100kΩ [2, 5, 9, 10, 14]. The token-based nature is fundamental to Brownian circuits, so the above requirement is important for them.

Another requirement for SET circuits concerns the thermal energy. If the thermal energy dominates over the charging energy E_c, tunneling is very likely to take place spontaneously, rather than controlled. The condition to avoid this is $E_c = \frac{{q_e}^2}{2 \cdot C} \gg k_B \cdot T$, where k_B is Boltzmann's constant and T is the absolute temperature. For a temperature of $1K$, for example, this equation implies that the capacitance of a Coulomb island should not exceed $926aF$. For higher temperatures, this capacitance is lower, which imposes a strict upper bound to the size of a quantum dot: For room temperature the capacitance should be less than $1aF$, which corresponds to an approximate diameter of $1nm$ in silicon. In the context of Brownian circuits, the above limitation is less strict, since spontaneous tunneling behavior has the potential to be employed for Brownian search. Implementations of SET circuits in terms of Brownian circuits may thus allow higher operating temperatures or larger feature sizes.

A preliminary study on implementations of Brownian circuits by SET technology has been conducted in [1,25] (see Figures 14 and 15). This study focuses on a design of the Hub, as well as of a CJoin that is one-way, i.e., that

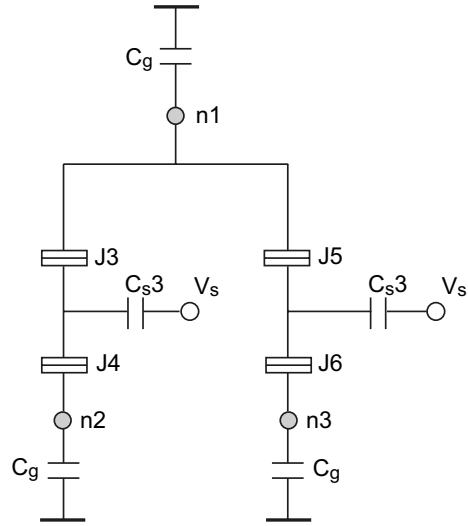

FIGURE 14
SET circuit of the Hub based on the design in [25]. The three wire terminals of the Hub are modeled by the Coulomb islands $n1$, $n2$, and $n3$. When there is an electron vacancy on n_1, an electron is supplied from either n_2 or n_3 via junctions J_4 and J_3, or J_6 and J_5. Capacitance C_s3 is chosen to be sufficiently low such that the thermal energy of the electron allows it to tunnel between n_1 and n_2, or between n_1 and n_3 in a random fashion.

cannot reverse its tokens once they have been output. Though the CJoin in this study is different from the (two-way) CJoin in Figure 3, it can be considered as having ratchets attached to its output lines. Strictly spoken, the CJoin does not need Brownian search at its output sides (see Ratchet Condition in Section 6). Therefore, incorporating ratchets in a CJoin does not compromise the correctness of circuit designs. The one-way CJoin in combination with the Hub can thus be used as the basis of a universal set of primitives. The one-way design of the CJoin implies that fluctuations do not play an active role in its operation, unlike in the design of the Hub. The Hub and the CJoin thus work under somewhat different (but compatible) regimes of circuit parameters, the parameter settings of the Hub facilitating Brownian search, and the settings of the CJoin being more in line with deterministic behavior. Fluctuations in the Hub arise when the voltages over its tunneling junctions are brought close enough to their critical voltages such that electrons will tunnel forward and backwards through the junctions due to thermal energy. The thermal energy is thus effectively used as a random control voltage. The CJoin on the other hand is designed with buffering techniques in mind [10], such that the thermal energy has little or no effect on its behavior. In [1] a Half-Adder based on these SET-designs is simulated, confirming its correct behavior.

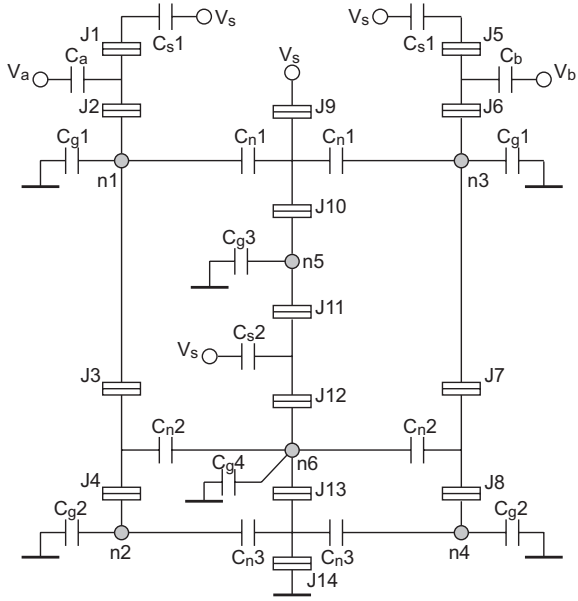

FIGURE 15
SET circuit of the CJoin [1]. Signals are input to the terminals V_a and V_b and the output terminals are modeled by the Coulomb islands $n2$ and $n4$. When input V_a (resp. V_b) goes high, an electron will tunnel through J_2 and J_1 (resp. J_6 and J_5), resulting in a positive charge on n_1 (resp. n_3). When both V_a and V_b are high and thus n_1 and n_3 containing a positive charge, an electron will tunnel from n_5 through J_{10} and J_9 to the source V_s. This triggers the tunneling of an electron from n_6 to n_5. The resulting positive charge at n_6 will cause tunneling of an electron from n_2 to n_1 through J_4 and J_3 and an electron from n_4 to n_3 through J_8 and J_7. The resulting positive charges at n_2 and n_4 will then cause the charge in n_6 to flow to ground after an electron tunnels through J_{14} and J_{13}. This restores the circuit to its initial state.

8 CONCLUSIONS AND DISCUSSION

The trend towards nanocircuits that are switched by ever-decreasing numbers of particles will eventually lead to a regime where the law of large numbers ceases to hold. Fluctuations of signals will play a major role, requiring new ways to deal with the inherent decrease in reliability. Brownian circuits offer a new perspective in this context: they actively exploit fluctuations by searching through the state space of a circuit. This strategy allows decreased complexity of circuit primitives and circuit topologies, because it obviates the use of added circuitry to avoid the deadlocks common in conventional token-based circuits. The Brownian circuits in this paper are based on three primitives—the Hub, CJoin, and Ratchet—that are less complex than the primitives in [23], and that allow more straightforward circuit designs.

Simplicity of primitives and circuits are important factors contributing to the efficiency by which the circuits can be implemented physically.

Though Brownian circuits depend on fluctuations in their operation, they also need to restrict fluctuations to bias Brownian search away from directions not leading to output states. This is accomplished through the controlled use of Ratchets, which are placed so that they do not interfere with Brownian search, while also speeding up computations. When placed at the very beginning of the input lines of a circuit, Ratchets will lock in the input tokens, thus preventing the tokens from backing out from the circuit through the input lines. In general, Ratchets are placed so that they limit the Brownian search space to the bare minimum, yet do not block Brownian search where it is needed. At locations where Brownian search is indispensable Ratchets are left out. In [23] it is pointed out that the expected time for a token with unbiased fluctuations to move from one end of a line with length L to the other end scales with L^2. When Ratchets are placed at constant distances D from each other, the token will be sped up by a factor of L/D on that line. Placement of Ratchets on output lines of circuits allows us to fix output of a circuit, so that signals cannot reenter it. Ratchets thus serve as buffers that isolate output of a circuit from the random fluctuations inside the circuit.

Single Electron Tunneling (SET) circuits, discussed in Section 7, are promising candidates for implementing Brownian circuits, because of the discreteness of their signals, which are represented as individual electrons, and because of the stochastic nature of tunneling, which fits well with the asynchronous nature of Brownian circuits. Discreteness in SET circuits manifests itself not only in the physical representation of a signal, but also in the position of a signal when it is physically represented by electrons held in Quantum dots. Quantum dots can be implemented in silicon [17], but molecular implementations are also possible, like in [20], where a single Co-ion bonded to a polypyridyl ligand behaves like a Coulomb island. In [3] tokens are represented by electrons in the bonds of molecules that are organized on a discrete grid.

Brownian circuits, unconventional as they may be, are no different from traditional circuits with respect to the range of functionalities they can realize. The logic design style for the Brownian circuits in this paper is based on Canonical Disjunctive Normal Forms, with CJoins encoding minterms and Hubs summing them. This implies that the complexity of designs is of the same order as in traditional logic, apart from a constant factor. There are differences too: whereas traditional electronic circuits employ voltage-encoded signals, Brownian circuits use tokens and they work without a clock. Due to this different framework, Brownian circuits need dual-rail encoding to represent logical values of signals, which requires a larger number of lines. Arbitration, on the other hand, is implemented more efficiently in Brownian

circuits, since it is inherent in the stochastic nature of Brownian search. Since Brownian circuits have the potential for implementations at integration densities exceeding conventional technology, the area required to implement a circuit may be significantly decreased, even when taking into account the overhead of dual-rail encoded lines.

Physical implementations of Brownian circuits are within the realm of possibilities, as simulations of designs based on Single Electron Tunneling technologies confirm. Other technologies may also be suitable for implementations, like spintronics, in which information is encoded by the spins of electrons. Though spintronics devices are less sensitive to quantum fluctuations as compared to SET, they are still subject to thermal fluctuations [18]. Another candidate is nanophotonics [16], where a token would represent an exciton—a quasi-particle consisting of an electron and electron hole, which results from the absorption of a photon in a semiconductor. In general, for a technology to be suitable for implementations of Brownian circuits, it is necessary that it supports a representation of signals that is token-based and that are subject to fluctuations.

REFERENCES

[1] I. Agbo, S. Safiruddin, and S.D. Cotofana. (2009). Implementable building blocks for fluctuation based calculation in single electron tunneling technology. In *Proc. 9th IEEE Int. Conf. on Nanotechnology (IEEE NANO)*, pages 450–453.

[2] N. Asahi, M. Akazawa, and Y. Amemiya. (January 1998). Single-electron logic systems based on the binary decision diagram. *IEICE Transactions on Electronics*, E81-C(1):49–56.

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

[4] C.H. Bennett. (1982). The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 21(12):905–940.

[5] S. Cotofana, C. Lageweg, and S. Vassiliadis. (March 2005). Addition related arithmetic operations via controlled transport of charge. *IEEE Transactions of Computers*, 54(3):243–256.

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

[7] K. Kinosita, R. Yasuda, H. Noji, and K. Adachi. (April 2000). A rotary molecular motor that can work at near 100% efficiency. *Philosophical Transactions of the Royal Society of London*, 355(1396):473–489.

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

[9] C. Lageweg, S. Cotofana, and S. Vassiliadis. (2001). A linear threshold gate implementation in single electron technology. In *WVLSI '01: Proceedings of the IEEE Computer Society Workshop on VLSI 2001*, pages 93–98, Washington, DC, USA. IEEE Computer Society.

[10] C. Lageweg, S. Cotofana, and S. Vassiliadis. (2002). Static buffered set based logic gates. In *Proceedings of the 2nd IEEE International Conference on Nanotechnology (IEEE Nano)*, pages 491–494, Arlington, USA.

[11] J. Lee, S. Adachi, and F. Peper. (2011). A partitioned cellular automaton approach for efficient implementation of asynchronous circuits. *The Computer Journal*, 54(7):1211–1220.

[12] J. Lee and F. Peper. (2008). On brownian cellular automata. In *Proc. of Automata 2008*, pages 278–291, UK. Luniver Press.

[13] C. Mead and L. Conway. (1980). *Introduction to VLSI Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[14] C. Meenderinck and S. Cotofana. (July 2007). Computing Division Using Single-Electron Tunneling Technology. *IEEE Transactions on Nanotechnology*, 6(4):451–457.

[15] J.D. Meindl, Q. Chen, and J.A. Davis. (2001). Limits on Silicon Nanoelectronics for Terascale Integration. *Science*, 293(5537):2044–2049.

[16] M. Ohtsu, K. Kobayashi, T. Kawazoe, T. Yatsui, and M. Naruse. (2008). *Principles of Nanophotonics*. Taylor & Francis.

[17] Y. Ono, Y. Takahashi, K. Yamazaki, M. Nagase, H. Namatsu, K. Kurihara, and K. Murase. (2000). Fabrication method for IC-oriented Si single-electron transistors. *IEEE Trans. Electron Devices*, 47(1):147–153.

[18] I.V. Ovchinnikov and K.L. Wang. (2008). Variability of electronics and spintronics nanoscale devices. *Applied Physics Letters*, 92(9):093503–1–3.

[19] K.V. Palem. (2005). Energy aware computing through probabilistic switching: a study of limits. *IEEE Trans. Computers*, 54(9):1123–1137.

[20] J. Park, A.N. Pasupathy, J.I. Goldsmith, C. Chang, Y. Yaish, J.R. Petta, M. Rinkoski, J.P. Sethna, H.D. Abruña, P.L. McEuen, and D.C. Ralph. (2002). Coulomb blockade and the Kondo effect in single-atom transistors. *Nature*, 417(6890):722–725.

[21] P. Patra and D.S. Fussell. (1994). Efficient building blocks for delay insensitive circuits. In *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 196–205.

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

[23] F. Peper, J. Lee, J. Carmona, J. Cortadella, and K. Morita. (2013). Brownian circuits: Fundamentals. *ACM Journal on Emerging Technologies in Computing Systems*, 9(1):3:1–24.

[24] S. Safiruddin and S.D. Cotofana. (2007). Building blocks for delay-insensitive circuits using single electron tunneling devices. In *Proc. 7th IEEE Int. Conf. on Nanotechnology (IEEE NANO)*, pages 704–708.

[25] S. Safiruddin, S.D. Cotofana, F. Peper, and J. Lee. (2008). Building blocks for fluctuation based calculation in single electron tunneling technology. In *Proc. 8th IEEE Int. Conf. on Nanotechnology (IEEE NANO)*, pages 358–361.

[26] T. Yamada, M. Akazawa, T. Asai, and Y. Amemiya. (2001). Boltzmann machine neural network devices using single-electron tunnelling. *Nanotechnology*, 12(1):60–67.