Brownian Circuits: Fundamentals

FERDINAND PEPER, National Institute of Information and Communications Technology, Japan
JIA LEE, Chongqing University, China
JOSEP CARMONA and JORDI CORTADELLA, Universitat Politècnica de Catalunya, Spain
KENICHI MORITA, Hiroshima University, Japan

Random fluctuations will be a major factor interfering with the operation of nanometer scale electronic devices. This article presents circuit architectures that can exploit such fluctuations, if signals have a particle-like (discrete, token-based) character. We define an abstract circuit primitive that, though lacking functionality when used with fluctuation-free signals, becomes universal when fluctuations are allowed. Key to the power of a signal's fluctuations is the ability to explore the state space of a circuit. This ability is used to resolve deadlock situations, which could otherwise only be averted by increased design complexity. The results in this article suggest that in the design of future computers, signal fluctuations, rather than being an impediment to be avoided at any cost, may be an important ingredient to achieve efficient operation.

Categories and Subject Descriptors: F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Probabilistic computation*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Unbounded-action devices*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Petri nets*

General Terms: Theory, Reliability, Algorithms

Additional Key Words and Phrases: Nanocomputing, asynchrony, fluctuation-driven computation, Brownian motion

ACM Reference Format:
Peper, F., Lee, J., Carmona, J., Cortadella, J., and Morita, K. 2013. Brownian circuits: Fundamentals. *ACM J. Emerg. Technol. Comput. Syst.* 9, 1, Article 3 (February 2013), 24 pages.
DOI: http://dx.doi.org/10.1145/2422094.2422097

1. INTRODUCTION

The trend toward reduced energy consumption of integrated circuits will eventually lead to devices that are switched by small numbers of particles. In such a regime near the thermal limit [Meindl et al. 2001]—when switching energy barely exceeds that of thermal noise—the law of large numbers ceases to hold, and fluctuations will play a prominent role [Ovchinnikov and Wang 2008]. Since noise and fluctuations compromise the reliability of VLSI systems, various strategies have been devised to counter their effects. Suppressing noise is the most conventional of those, but it only works if signal levels are well above the thermal limit by, say, a safety factor of 100 [Mead and Conway 1980, p. 358]. The detection and correction of errors through the redundant

This work has been partially supported by the project FORMALISM (TIN2007-66523) and NSFC (No. 61170036).
Authors' addresses: F. Peper, Brain ICT Laboratory, National Institute of Information and Communications Technology (NICT), Kobe, University of Hyogo, Himeji, and Osaka University, Japan; email: peper@nict.go.jp; J. Lee, College of Computer Science, Chongqing University, Chongqing, China; J. Carmona and J. Cortadella, Dept. of Software, Universitat Politècnica de Catalunya, Barcelona, Spain; K. Morita, Dept. Information Engineering, Hiroshima University, Higashi-Hiroshima, Japan.

DOI: http://dx.doi.org/10.1145/2422094.2422097

encoding of information is another strategy, but it becomes less useful if the safety factor decreases. Yet another strategy advocates limiting applications to those that have some inherent robustness to noise, like image processing, but this restricts the class of computation problems.

Noise and fluctuations have been part of the environment of biological organisms for as long as evolution has run its course. Organisms have mechanisms in place, not only to cope with fluctuations, but also to exploit them as a driving force in biological processes [Frey and Kroy 2005; Yanagida 2008]. The operation of molecular motors, for example, revolves to a great extent around Brownian motion [Hänggi et al. 2005; Reimann 2002; Yanagida 2008], which is the random motion of particles that results from collisions with the atoms or molecules of the medium in which the particles are suspended. In the original context of Brownian motion [Brown 1828], the medium is a fluid and the particles immersed in it are micrometer sized, but the term is used for fluctuations in systems with nanometer scale features as well [Hänggi and Ingold 2005]. In biological organisms, Brownian motion is thought to play a role in randomization [McAdams and Arkin 1999] and in facilitating processes involving small numbers of molecules [Frey and Kroy 2005]. The randomness of Brownian motion increases the likelihood that reactable molecules move to each others' vicinity, and thus provides a free search mechanism [Dasmahapatra et al. 2006; Yanagida 2008], without there being a need for an explicit control mechanism. Can similar mechanisms be used in the framework of nanoelectronics, and if so, how can we avoid negatively affecting operating time?

An early proposal, by Bennett [1982], of computation driven by Brownian motion uses a mechanical microscale Turing machine, in which signals randomly search their way through the machine's circuitry, restricted only by the circuit's topology. This process resembles a random walk through a maze, from which one will eventually emerge at an exit. Driven by thermal noise, computations in this model move forward and backward evenly, taking a long time to complete, but they can be sped up by biasing them forward through expending energy. Bennett points out similarities of Brownian computation with the tape-copying machine embodied by RNA polymerase [Bennett 1982].

Noise and fluctuations have also been used in simulated annealing on a Boltzmann machine based on Single Electron Tunneling devices [Yamada et al. 2001]. This method utilizes signal fluctuations to search in an energy landscape, thus showing similarities with the Brownian search in Bennett's Turing machine. Its main focus, however, is on optimization by neural networks, rather than on computation.

This article proposes a class of circuits that uses signal fluctuations as a fundamental mechanism underlying computations. Like the Turing Machine of Bennett [1982], the proposed circuits use a random search through the circuit topology to drive computation, but unlike Bennett's mechanical construct, the circuits are formulated in a more abstract framework (Section 2), which facilitates a mathematical analysis of their behavior. Lacking a clock, the circuits are asynchronous, and a wide range of formal models are available in this framework to analyze them, like the Petri Nets used in this article. We define a circuit primitive (Section 3) and prove that signal fluctuations are essential for this primitive to be universal for the proposed class of circuits (Sections 4 and 5). Universality implies that any arbitrary circuit in the class can be constructed from this primitive. The power of signal fluctuations derives from their ability to search and backtrack out of deadlocks (Section 5). The use of Brownian search to backtrack out of deadlock situations is illustrated by Video 2 of the Supplementary Online Materials (for details see Figure 11 in Section 4). The theory developed in this article on Brownian circuits includes conditions to decide when a computation is finished—an issue that is not trivial since a Brownian circuit may repeatedly move

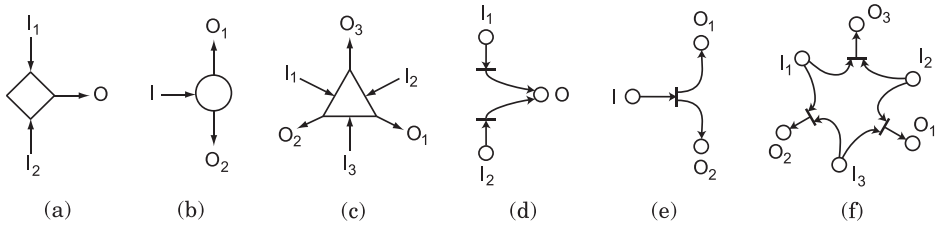


Fig. 1. Primitive modules for delay-insensitive token-based circuits (left half) and the corresponding Petri nets that define their functionality (right half). (a) Merge, which merges two streams of tokens on input lines I_1 and I_2 into one stream on output line O. (b) Fork, which duplicates every input token from input line I to its two output lines, O_1 and O_2. (c) Tria, which joins two input tokens into a single output token as follows. Upon receiving one token each from input lines I_i ($i \in \{1, 2, 3\}$) and I_j ($j \in \{1, 2, 3\} \setminus \{i\}$), it outputs a token to line O_{6-i-j}. If there is a token on only one of the input lines, it is kept pending, until a token on one more input line appears. (d) Net of the Merge, (e) Fork, and (f) Tria. Diagrams typical for Petri nets are used: places are represented by circles and transitions by short line segments orthogonal to the in- and out-going arrows that denote the flow relation.

in and out of its final state (Section 6). We show how a Brownian circuit can be sped up by placing so-called *ratchet*-devices—a kind of diode—at selected locations in a circuit (Section 7).

2. TOKEN-BASED CIRCUITS

2.1. Introduction

Tokens are discrete indivisible units that are used as signals in circuits or systems. They are graphically represented by fat dots, like in the left of Figure 5. Well-known token-based systems are *Petri nets* (e.g., see Murata [1989]), which are commonly used for modeling purposes. Token-based circuits are relevant for physical implementations in which signals have a discrete character, such as found in *charge-state logic* [Korotkov and Likharev 1998]. In such logic, the behavior of devices and circuits depends on the movement and interaction of individual electrical charges, like in Single Electron Tunneling circuits [Lageweg et al. 2004; Ono et al. 2005]. Charge-state logic is uncommon in current commercially available electronics; rather most circuits employ voltage-state logic, in which signals are represented by voltage levels.

The token-based circuits in this article are delay-insensitive, which means that they allow arbitrary delays of signals in lines or in circuit elements, without this compromising the circuit's operations. Delay-insensitive circuits are not governed by a clock, so they belong to the class of asynchronous circuits [Sparsø and Furber 2001].

Token-based delay-insensitive circuits can be constructed from a fixed set of circuit primitives, analogously to synchronous circuits, which use, for example, NOT-gates and AND-gates as primitives. When a set of primitives offers sufficient functionality to construct any circuit possible in a class of circuits, this set is *universal* for that class. Figures 1(a)–(c) shows one such universal set for the class of delay-insensitive token-based circuits [Lee et al. 2005]: the Merge (called P-Merge in [Lee et al. 2005]), the Fork, and the Tria [Patra and Fussell 1996]. A fundamental aspect of the Tria is its so-called *join functionality*, according to which an output token is only produced when there are two input tokens available. The presence of merely one input token will keep that token pending on its input terminal until the second input token arrives at another input terminal. Join functionality thus synchronizes tokens on a local scale. It is of fundamental importance in delay-insensitive circuits, because it substitutes for the clock in synchronous circuits.

2.2. Formalization

The Brownian circuits in this article are expressed in the framework of Petri Nets, which are described by the following definitions—most of them being standard.

Definition 2.1. A *Net* is a 3-tuple $N = (P, T, F)$, where:

(1) P (places) and T (transitions) are finite sets with $P \cap T = \emptyset$,
(2) $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation,
(3) $\forall t \in T\ \exists p, q \in P\colon (p,t), (t,q) \in F$,
(4) $\forall t \in T, p, q \in P : (p,t), (t,q) \in F \Rightarrow p \neq q$.

Condition (3)—no dangling transitions—and condition (4)—no self-loops of transitions—are included in this definition to avoid having to deal with special cases in proofs. Nets are represented graphically by the diagrams common for Petri nets. Figures 1(d), (e), and (f) show the nets of the Merge, Fork, and Tria, respectively.

Definition 2.2. A *marking* of a net $N = (P, T, F)$ is a subset of P.

The definition implies that a net is *safe*, i.e., that at most one token is contained in a place. A marking is represented graphically by putting a token in each place belonging to it.

Definition 2.3. Let $x \in X$, with $X = P$ or $X = T$, then ${}^\bullet_N x = \{y \in X | (y,x) \in F\}$ is the *input set* (or *pre-set*) of x in net N, and $x^\bullet_N = \{y \in X | (x,y) \in F\}$ is the *output set* (or *post-set*) of x (the subscript indicating a net is omitted if the context is clear). The expression $|x^\bullet|$ is called the *out-degree* of x. Let $Y \subseteq P$ or $Y \subseteq T$, then ${}^\bullet Y = \bigcup_{y \in Y} {}^\bullet y$ and $Y^\bullet = \bigcup_{y \in Y} y^\bullet$.

Definition 2.4. Let $N = (P, T, F)$ be a net and $C \subseteq P$ be a marking. Then transition $t \in T$ is *enabled* in C, written as $C\,[t\rangle$, if ${}^\bullet t \subseteq C$ and $t^\bullet \cap C = \emptyset$.

Definition 2.5. Let $N = (P, T, F)$ be a net, let $C, D \subseteq P$ be markings, and let T^* be the set of all sequences of transitions in T.

(1) The transition $t \in T$ *fires from C to D*, written as $C\,[t\rangle\, D$, if $C\,[t\rangle$ and $D = (C - {}^\bullet t) \cup t^\bullet$. When the transition is not specified, we write $C \to D$.
(2) Let $t_1, ..., t_n \in T$, with $n \geq 1$. Then the firing sequence $f = t_1 ... t_n \in T^*$ is said to *fire from C to D*, written as $C\,[t_1 ... t_n\rangle\, D$, if there exist markings $C_0, C_1, ..., C_n \subseteq P$ with $C_0 = C, C_n = D$, and $C_{i-1}\,[t_i\rangle\, C_i$ for all $i \in \{1, ..., n\}$. The *length of f*, written as $L(f)$, is the number n of transitions in it. The *set of firing sequences from C to D* is denoted by FS^D_C.

This definition extends the standard definition of a net's firing behavior to include the length of firing sequences. It plays an important role in the proof of Theorem 4.4.

Definition 2.6. Let $N = (P, T, F)$ be a net, let $C, D \subseteq P$ be markings, and let T^* be the set of all sequences of transitions in T.

(1) D is said to be *reachable from C*, written as $C \overset{*}{\to} D$, if there exists a firing sequence $f \in T^*$ such that $C\,[f\rangle\, D$.
(2) The *set of markings reachable from C*, written as $\mathrm{RE}(C)$, is defined by $\{D \subseteq P \mid C \overset{*}{\to} D\}$. The *set of markings reachable from C within n firings of transitions* is defined by $\mathrm{RE}^n(C) = \left\{D \in \mathrm{RE}(C) | \exists f \in \mathrm{FS}^D_C : L(f) \leq n\right\}$. By definition, $\mathrm{RE}^0(C) = C$.

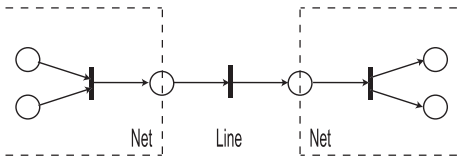


Fig. 2. Serial composition of two nets by a line.

Whether a marking is reachable from another marking is an important measure for establishing the presence or absence in the net of deadlocks (see Sections 4 and 5).

Definition 2.7. Given two nets $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, a net $N = (P, T, F)$ is called a *serial composition* of N_1 and N_2 if:

(1) $P = P_1 \cup P_2$,
(2) $T = T_1 \cup T_2 \cup T_c$, for a certain non-empty set of transitions T_c that satisfies $T_c \cap (T_1 \cup T_2) = \emptyset$,
(3) $F = F_1 \cup F_2 \cup F_c$, for a certain flow relation F_c for which $(i,j = 1, 2)$

$$F_c \subseteq \{(p,t),(t,q) \mid p \in P_i \wedge p^{\bullet}_{N_i} = \emptyset \wedge q \in P_j \wedge j = 3 - i \wedge {}^{\bullet}_{N_j} q = \emptyset \wedge t \in T_c\}$$

and for every $t \in T_c$, it holds $|{}^{\bullet}_{N} t| = |t^{\bullet}_{N}| = 1$.

Informally, a serial composition results in a net that maintains the flow relation of the two nets, and adds a new set of transitions (T_c in Definition 2.7) connecting *sink places* (i.e., with no successor transitions) of one subnet to *source places* (i.e., with no predecessor transitions) of the other. A transition in T_c allows tokens to flow from one subnet to the other in the composed net, and is therefore called a *line* (Figure 2). The serial composition of two lines is a multisegmented line.

Definition 2.8. An *Interfaced Elementary Net* (IEN) *system* is a 5-tuple $N = (P, T, F, S_i, S_f)$, where

(1) (P, T, F) is a net,
(2) S_i is the set of *initial markings*, and
(3) S_f is the set of *final markings*.
(4) For all $D \in S_f$ there exists a $C \in S_i$ such that $C \overset{*}{\rightarrow} D$.

Unlike in traditional EN systems [Rozenberg and Engelfriet 1998], there can be more than one initial marking in an IEN system. Each initial marking can be thought of as a particular input to the net. The final markings describe the possible outputs of the net.

Definition 2.9. A *Stochastic Interfaced Elementary Net* (SIEN) *system* is a 6-tuple $N = (P, T, F, S_i, S_f, \Lambda)$, where

(1) (P, T, F, S_i, S_f) is an IEN system,
(2) $\Lambda = \{\lambda_1, ..., \lambda_n\}$ is a set of non-negative real numbers, whereby λ_i is the transition rate of transition t_i in $T = \{t_1, ..., t_n\}$.

A SIEN system is an IEN system in which each transition is associated with an exponentially distributed random variable that expresses the delay from the enabling to the firing of the transition. If several transitions are simultaneously enabled, the transition with the shortest delay will fire first [Murata 1989]. This distribution of a transition's delay is assumed to be independent of the time that has elapsed and independent of the markings of the net. The stochasticity of a SIEN system guarantees

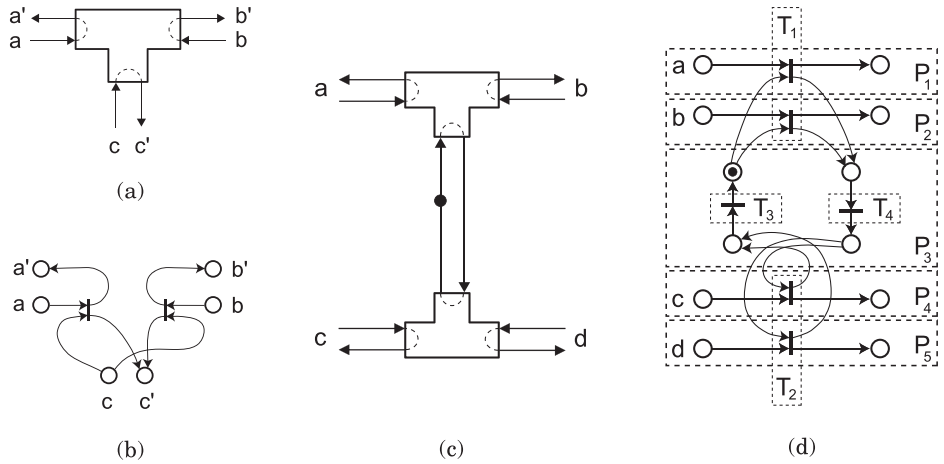


Fig. 3. (a) The T-element, which is a simple Token-Pass (TP) circuit. (b) Net of the T-element. The lines a–a' and b–b' in the arms of the T are the *choice lines*, while the line c–c' in the base of the T is the *base line*. The T-element requires one token as input on its base line and one token on one of its choice lines. The resulting transition moves the two tokens pairwise through the T-element to output them. If there are input tokens at the base line as well as at both choice lines, then the T-element makes an arbitrary choice as to which of the choice line tokens is processed, while leaving the token on the other choice line pending. (c) Circuit constructed from two T-elements, and (d) its net divided in paths P_1 to P_5 and Transition Clusters T_1 to T_4. Path P_3 represents the loop that corresponds to the pair of lines connecting the bases of the two T-elements.

that sufficient randomness is present in its transitions to avoid deadlocks and livelocks due to periodic behavior (see Section 5).

3. TOKEN-PASS CIRCUITS

A *Token-Pass circuit* is a circuit through which tokens pass via linear paths, on the way possibly interacting with one other, but never veering from their paths. Defined with physical plausibility in mind, Token-Pass circuits keep the number of tokens unchanged, be it on interconnection lines or inside modules.

Definition 3.1. Let $N = (P, T, F)$ be a net. A binary relation V is defined over the elements of T as follows. Let $t_1, t_2 \in T$, then $t_1 V t_2$ if and only if ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ or $t_1^\bullet \cap t_2^\bullet \neq \emptyset$. The transitive closure of V divides T into equivalence classes indicated by $T_1, ..., T_n$. These equivalence classes are called the *Transition Clusters* of N.

Definition 3.2. The net $N = (P, T, F)$ is *Token-Pass (TP)* if for all $i \in \{1, ..., n\}$, with n being the number of transition clusters in N, it holds that ${}^\bullet T_i \cap T_i^\bullet = \emptyset$ and there exists a bijection $\alpha_i : {}^\bullet T_i \to T_i^\bullet$ such that

(1) For all $p \in {}^\bullet T_i$ and all $t \in T_i$: $(p, t) \in F \iff (t, \alpha_i(p)) \in F$.
(2) For all $p \in {}^\bullet T_i$: $|p^\bullet| = |{}^\bullet \alpha_i(p)|$.

A circuit is called Token-Pass (TP) if its net is TP.

Definition 3.3. Let $N = (P, T, F)$ be a net that is Token-Pass. A binary relation W is defined over the elements of P as follows. Let $p_1, p_2 \in P$, then $p_1 \, W \, p_2$ if there exists an $i \in \{1, ..., n\}$ such that $\alpha_i(p_1) = p_2$ or $\alpha_i(p_2) = p_1$. The transitive closure of W divides P into equivalence classes indicated by $P_1, ..., P_m$. These equivalence classes are the *Paths* of N.

An example of a Token-Pass circuit is the *T-element* in Figure 3(a), which plays a pivotal role in this article. It has three lines passing through it, corresponding to the

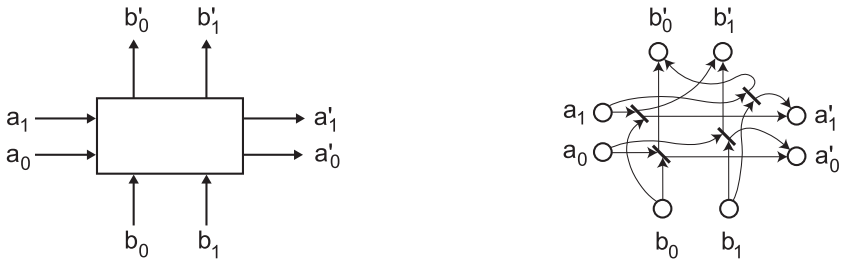


Fig. 4. (a) A module that is not Token-Pass, and (b) its representation as a net. The module assumes two tokens as input, one at one of the a-lines (left) and one at one of the b-lines (bottom). If a token is input to a_0, then the token at the b-lines passes through the module uncrossed, i.e. it moves from b_j to b'_j ($j \in \{0,1\}$), while the other token moves to a'_0. If a token is input to a_1, then the module operates as if the b-lines were crossed, i.e., the token at b_j moves to b'_{1-j}, while the token at a_1 moves to a'_1. If there is only one input token, then it remains pending until a second input token arrives.

paths $\{a, a'\}$, $\{b, b'\}$, and $\{c, c'\}$ in the net in Figure 3(b). The line c–c' passing through the base of the T-element is the *base line*. The other two lines, each passing through an arm of the T-element, are the *choice lines*. A bijection α for this T-element maps a to a', b to b', and c to c'. There is one transition cluster in the net in Figure 3(b), which includes both its transitions.

In a Token-Pass net the bijectivity of the α relations implies that the places in a path can be arranged as a linear structure, either as an open path, or as a loop. A token never moves from one path to another. It will enter a path as input, if the path is not a loop structure, and moves within a path until it is output. The circuit constructed from two T-elements in Figure 3(c) has five paths in its net (Figure 3(d)), one of which is a loop. This loop contains one token, which will never leave the loop. By being at a certain location in the loop at a certain instance, this token enforces a certain order at which other tokens may pass through the net. Apart from enforcing an order of operations in a circuit, a loop may also be used to store a circuit state, as in the 1-bit memory in Figure 8(b) or (c). In general, loops are used for internal processing in circuits.

The Token-Pass property is a strict one, as most nets lack it. The net of the module in Figure 4(a) is not Token-Pass, because it does not satisfy the condition in Definition 3.2: a bijection α from the input places a_0, a_1, b_0, and b_1 in Figure 4(b) to output places is impossible.

LEMMA 3.4. *The class of Token-Pass nets is closed under serial composition.*

PROOF. We proceed for the binary serial composition as described in Definition 2.7 for N_1, N_2, and N (the extension to k components is straightforward). First, the transition clusters of N_i ($i = 1, 2$) are preserved in N, because neither are transitions removed, nor are transitions added that have non-empty pre- or post-set intersections in N_i. This implies that the corresponding bijections apply to the transitions in N_1 and N_2, thus guaranteeing the conditions of Definition 3.2. The existence of bijections for the clusters $\widehat{T_1 \ldots T_m}$ in T_c ($m \leq |T_c|$) remains to be shown (c.f. Definition 3.2). By Definition 2.7, for every transition $t \in T_c$, the transition cluster $\widehat{T}_i$ ($i \in \{1, ..., m\}$) containing t satisfies $|{}^\bullet\widehat{T}_i| = |\widehat{T}_i^\bullet| = 1$ and ${}^\bullet\widehat{T}_i \cap \widehat{T}_i^\bullet = \emptyset$. Hence for each transition t in T_i, a bijection can be defined that associates the unique input place in ${}^\bullet t$ with the unique output place in $t^\bullet$. These bijections satisfy the two conditions of Definition 3.2. □

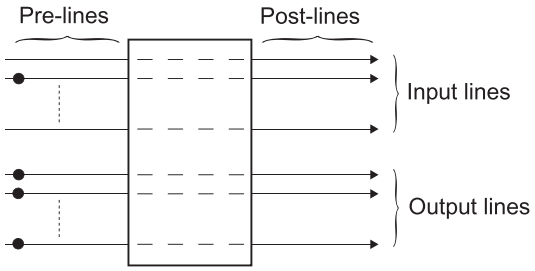


Fig. 5. Scheme of a Token-Pass circuit. Input lines and output lines all pass through the circuit.

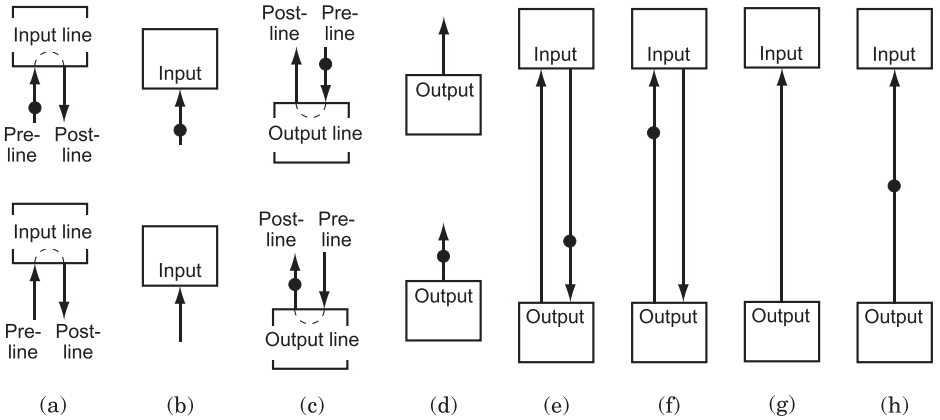


Fig. 6. Signaling protocol of a Token-Pass circuit and its token-based equivalent. (a) Token-Pass module with input (top) and without input (bottom), and (b) the token-based equivalents. (c) Token-Pass module before producing output (top) and after output (bottom), and (d) the token-based equivalents. (e) Token on a pair of lines connecting two modules is transferred through an operation of the bottom module from one line to (f), the other line. In terms of a conventional token-based circuit in which a bottom module produces output to a top module, the operations in (e) and (f) correspond to respectively (g) no token produced on a single interconnections line, and (h) one token produced.

Informally, the lemma proves that connecting two paths to each other results in a structure that is also a path. A Token-Pass circuit can be characterized as a collection of lines that either are connected in loops, or that run through the circuit to form input lines or output lines (Figure 5).

A token on the *pre-line* of an input line (left in Figure 5) denotes input to the circuit, and it passes to the corresponding *post-line* (right) as part of the circuit operation. All output lines have tokens on their pre-lines before the circuit operation, and—depending on the input to the circuit—only some of these tokens are passed to the corresponding post-lines to signify output.

Tokens are processed according to the protocol in Figures 6(a)–(d). A token on an input pre-line indicates input for the module from that line (top of Figure 6(a)). When processing this input, the module moves the token to the corresponding input post-line. While an input line may lack a token on its pre-line—when there is no input (bottom of Figure 6(a))—an output pre-line always contains a token before an operation (top of Figure 6(c)). To signify output of the operation, the module moves the token to the corresponding post-line (bottom of Figure 6(c)). In other words, depending on the pattern of tokens present on the input pre-lines, a circuit will move certain tokens on the output pre-lines to the output post-lines.

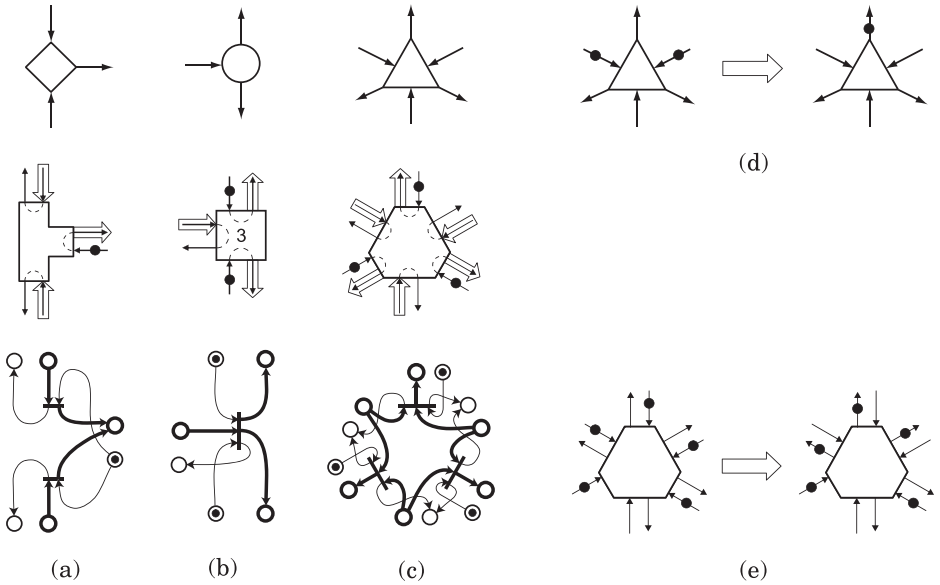


Fig. 7. (a) Merge (top), TP-Merge (center), and net of the TP-Merge (bottom); (b) Fork (top), TP-Fork (center), and net of the TP-Fork (bottom); (c) Tria (top), TP-Tria (center), and net of the TP-Tria (bottom). The hollow arrows of the Token-Pass modules in the center denote the corresponding input and output lines of the original modules at the top. The remaining line ends carry "dummy" tokens, necessary to guarantee that the number of tokens remains unchanged from input to output. For example, the TP-Fork, displayed with the symbol 3 in it (see Figure 9 for details), consumes three tokens as input (of which two are dummies) and it produces three tokens as output (of which one is a dummy). The thick lines in the nets at the bottom correspond to the nets of the respective token-based modules in Figures 1(d), 1(e), and 1(f). (d) Transition describing an operation of a Tria, and (e) the equivalent operation of a TP-Tria.

While the protocol in Figures 6(a)–(d) defines the external input/output behavior of Token-Pass circuits, it can also be used for interconnections between modules (Figure 6(e)(f)). Token-Pass modules are then connected to one other via a pair of lines that contains one token. The line among this pair on which the token resides determines the state of the signaling between the modules. There are two cases. In the first case (Figure 6(e)), the token is on the pre-line of the bottom module, waiting to be let through and become output of the module. This results in the second case (Figure 6(f)), in which the token is on the pre-line of the top module, waiting to be accepted as input to this module. Once the token is accepted and let through, the first case applies again. In terms of a conventional token-based circuit, these two cases correspond to no tokens (Figure 6(g)) respectively one token (Figure 6(h)) on an interconnection line between two modules.

The preceding correspondence between the signaling protocols of token-based circuits and Token-Pass circuits can be carried over to the modules from which circuits are made up. A Module in a Token-Pass circuit then has a pair of lines wherever the equivalent token-based module has a single line. When the Token-Pass equivalent of a token-based module M is denoted by *TP-M*, we then obtain the TP-Merge (Figure 7(a) center), the TP-Fork (Figure 7(b) center), and the TP-Tria (Figure 7(c) center). The TP-Merge turns out to be identical to the T-element, as a comparison of their nets reveals.

Token-Pass modules operate just like their token-based counterparts, though more lines and more tokens are involved. An operation of the TP-Tria is illustrated in

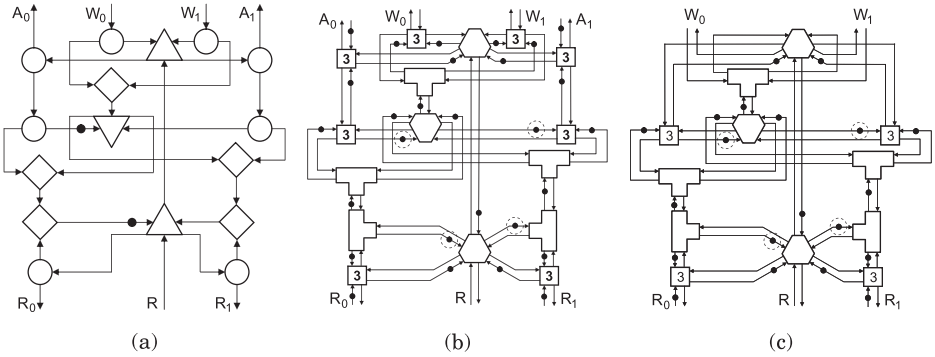


Fig. 8. (a) 1-Bit token-based memory. The state of the memory is stored by two tokens on the left of the two lower Trias (state 0, in this case), or on the right (state 1). State 0 or 1 is written into the memory by inputing a token to the input line W_0 or W_1 respectively. The writing operation is acknowledged by a token from line A_0 or A_1, respectively. By inputing a token to line R, the memory is read out, and a token is produced on either line R_0 or R_1, depending on the state of the memory. (b) Token-Pass equivalent of the 1-bit memory. Modules are connected to each other by pairs of lines, which effectively form loops, each containing a token. The memory's state is encoded by the positions of four tokens (encircled) in the loops on which they reside. (c) Simplified version of the Token-Pass 1-bit memory. Acknowledge lines are dropped, and their task is taken over by the post-lines of W_0 and W_1. See also Video 1 in the Supplementary Online Materials.

Figure 7(d), together with its token-based counterpart (Figure 7(e)). The equivalence of token-based circuits and Token-Pass circuits is used in the following theorem.

THEOREM 3.5. *The set {TP-Merge, TP-Fork, TP-Tria} is universal for the class of Token-Pass circuits.*

PROOF. Follows from the equivalence of token-based circuits and Token-Pass circuits, and universality of the set {Merge, Fork, Tria} for token-based circuits [Lee et al. 2005]. □

The circuit for the 1-bit memory [Peper et al. 2004] in Figure 8(a) is token-based. The corresponding Token-Pass circuit is constructed by replacing Merge, Fork, and Tria by their Token-Pass equivalents and connecting them by pairs of lines (Figure 8(b)). The same circuit, but with Acknowledge lines A_0 and A_1 left out, is shown in Figure 8(c); the function of the Acknowledge lines is taken over by the post-lines of the Writing lines W_0 and W_1. More efficient constructions for the 1-bit memory are given in Section 5.

4. THE T-ELEMENT

4.1. More Powerful than Expected?

Though the T-element introduced in the last section is the same as a TP-Merge, its functionality is more powerful than would be expected from the simple functionality of the Merge: the T-element can be used to construct the TP-Fork, as Figure 9 shows, while a conventional Fork cannot be constructed from a Merge.

The key to the increased power of the T-element is its inclusion of Join functionality (Section 2.1), its larger number of input and output lines, and the use of these lines to connect to other modules in a way that is richer than the pair-of-lines scheme in Figures 6(e) and (f). Can other modules be constructed from the T-element, in particular the TP-Tria? This question is important, because—as implied by Theorem 3.5—it determines whether the T-element qualifies as universal for the class of Token-Pass circuits or not. We show in this section that such a construction of the TP-Tria is

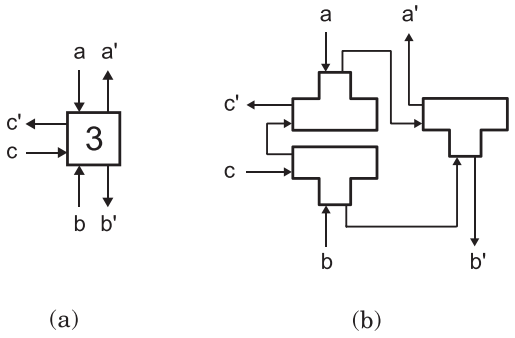


Fig. 9. (a) The TP-Fork, shown with the symbol 3 in it, has three tokens as input to the lines a, b, and c, and produces three tokens as output to the lines a', b', and c'. Input of less than three tokens results in the tokens being kept pending until there are three input tokens. (b) The TP-Fork constructed from three T-elements. In each T-element one arm remains unused.

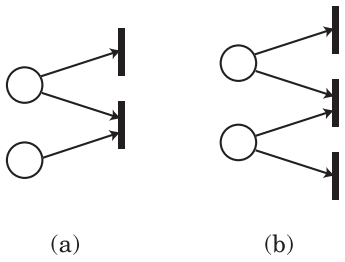


Fig. 10. (a) SPL-net: the two places share a transition in their post-sets, but at most one of the places has out-degree exceeding 1. (b) Not an SPL-net: the out-degree of both places exceeds one, while they share a transition in their post-sets.

impossible. It will become clear in Section 5, however, that adding fluctuations of tokens to the scheme completely alters this outcome.

4.2. Analyzing the Token-Pass Tria

A TP-Tria is constructed from T-elements by connecting the T-elements to each other such that the resulting circuit has the external input/output behavior of a TP-Tria. If in whatever arrangement of T-elements, there is a possibility of pathological input/output behavior, like a failure to produce output due to a deadlock, we consider the construction of the TP-Tria impossible. This section proves that there is no arrangement of T-elements that produces exact TP-Tria-like behavior. Key to this result is the underlying structure of the corresponding Petri net, which is largely limited by two properties: the Token-Pass property and the SPL property defined in the following.

Definition 4.1. A net $N = (P, T, F)$ is *Simple (SPL)* [Commoner 1972; Hack 1972] if for all $p, p' \in P$ it holds that $p \neq p' \Rightarrow (p^\bullet \cap p'^\bullet = \emptyset$ or $|p^\bullet| \leq 1$ or $|p'^\bullet| \leq 1)$.

SPL nets are a subclass of the more general *Extended Simple (ESPL) nets*, which are also denoted in the literature as *Asymmetric Choice* nets. Informally, an SPL net is a net in which, of all places having a transition in common in any of their post-sets, at most one of the places has out-degree larger than 1 (Figure 10(a) but not (b)). The T-element's net is SPL, as can be confirmed from Figure 3(b).

LEMMA 4.2. *The class of SPL nets is closed under serial composition.*

PROOF. Let T_c be the new set of transitions used to compose N_1 and N_2 (c.f. Definition 2.7). Since the only places in N_1 and N_2 that are modified in the composition are those with empty postsets, the outdegrees of these places will become at most 1 after the composition, so the conditions of SPL nets will be fulfilled for all places in N_1 and N_2. Finally, because every transition $t \in T_c$ satisfies $|t^\bullet| = |{}^\bullet t| = 1$, every two places $p, p' \in {}^\bullet T_c$ with $p \neq p'$ will satisfy $p^\bullet \cap p'^\bullet = \emptyset$. □

This lemma implies that all circuits constructed from T-elements have a net that is SPL. The significance of the SPL property is that it is a cause for deadlocks.

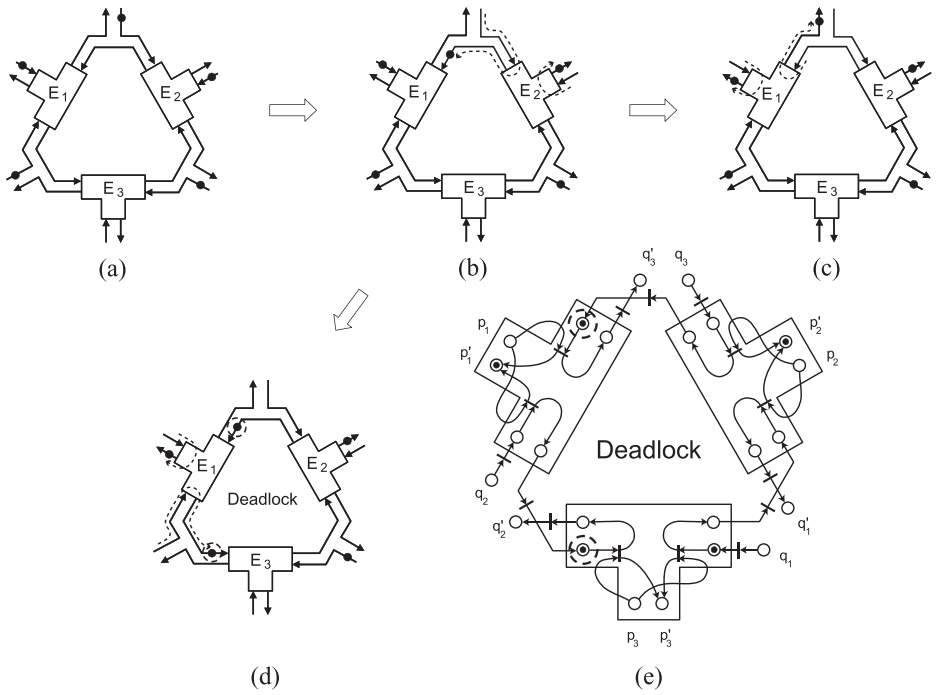


Fig. 11. (a) TP-Tria constructed from three T-elements. The TP-Tria has the same input tokens as in the left of Figure 7(e). (b) TP-Tria after one step in its operation. T-element E_1 now has tokens on all its prelines, implying that it has the choice between two ways to process its input. (c) One way leads to the correct output. (d) The other way causes a deadlock. The encircled tokens cannot be processed any further. (e) The net corresponding to the TP-Tria in deadlock. Labels p_i, p'_i, q_i, and q'_i ($i = 1, 2, 3$) are as in Figure 17 in the Appendix. The net is SPL. See also Video 2 in the Supplementary Online Materials.

Definition 4.3. A net $N = (P, T, F)$ is *deadlock-free* from the set of markings S to the set of markings S' if for all $C \in S$ there exists a $D \in S'$ such that for all markings $C' \in \mathrm{RE}(C)$: $C' \stackrel{*}{\rightarrow} D$. The net N has a *deadlock* from the set of markings S to the set of markings S' if N is not deadlock-free from S to S'.

Informally, a net is said to have a deadlock if, after receiving input, there is a possibility that the net fails to produce output.

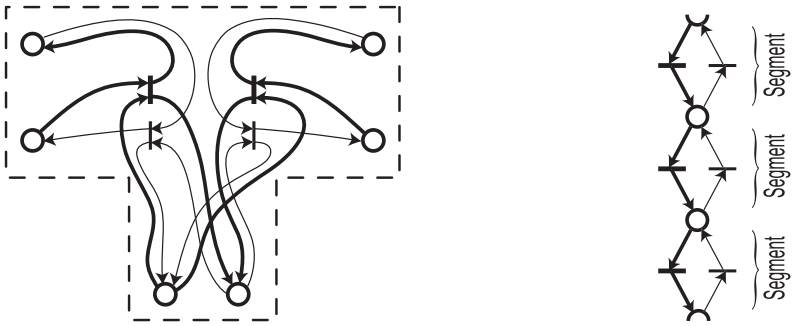
THEOREM 4.4. *Let $N = (P, T, F, S_i, S_f)$ be the IEN system of a TP-Tria. If N is Token-Pass and SPL, then N has a deadlock from S_i to S_f.*

PROOF. See Appendix. □

COROLLARY 4.5. *A deadlock-free TP-Tria can not be constructed from T-elements.*

PROOF. Every serial composition of T-elements is Token-Pass (Lemma 3.4) and SPL (Lemma 4.2), proving the corollary. □

Theorem 4.4 is illustrated by Figure 11, where a TP-Tria constructed from three T-elements runs into a deadlock via the sequence in Figure 11(a),(b),(d). The marking of the net in Figure 11(e) corresponds to the deadlock of the circuit in Figure 11(d). This marking results from a legal combination of inputs, but it will produce no outputs. The deadlocked tokens get stuck in places with out-degrees 1 (Figure 11(e)), like in the proof of Theorem 4.4. The same combination of inputs to the TP-Tria could also



(a) (b)

Fig. 12. (a) Brownian T-element (same symbol as non-Brownian T-element) represented as a net. The thick lines correspond to the non-Brownian equivalent of the net. (b) Net of a Brownian line (multisegmented). Transition rates are not indicated here.

result in an output marking without deadlock (Figure 11(c)), but such a marking is not guaranteed to occur, since the order of the transitions in the net is unpredictable. This result affects not only TP-Trias, but also circuits containing TP-Trias. The 1-bit Token-Pass memory in Figure 8(b) or Figure 8(c), for example, has no deadlock-free construction based on T-elements.

5. BROWNIAN CIRCUITS

Circuits are *Brownian* when tokens fluctuate in lines and modules. Unlike with the monotonous behavior of tokens in the previous sections, operations in a Brownian circuit are done and undone repeatedly, as tokens move forward and backward. The net of a Brownian circuit is constructed from its non-Brownian equivalent by inserting a reverse transition for each transition in the original net and assigning positive transition rates to all transitions. Figure 12 shows the nets of the Brownian equivalent of a T-element and of a multisegmented line.

Definition 5.1. Let $N = (P, T, F, S_i, S_f)$ be an IEN system. Then the *Brownian Equivalent* of N is a SIEN system $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{S}_i, \widetilde{S}_f, \Lambda)$ for which

(1) $\widetilde{P} = P$,
(2) $\widetilde{T} = T \cup \breve{T}$, whereby transition $\breve{t} \in \breve{T}$ if $\exists t \in T : ({}^\bullet\breve{t} = t^\bullet) \wedge (\breve{t}^\bullet = {}^\bullet t)$,
(3) $\widetilde{F} = F \cup \breve{F}$, whereby $(p, \breve{t}) \in \breve{F}$ if $\exists t \in T : (t, p) \in F$ and whereby $(\breve{t}, p) \in \breve{F}$ if $\exists t \in T : (p, t) \in F$,
(4) $\widetilde{S}_i = S_i$,
(5) $\widetilde{S}_f = S_f$,
(6) The transition rates $\lambda_1, ..., \lambda_n \in \Lambda$, with $n = |\widetilde{T}|$, are all positive.

The Brownian equivalent $\widetilde{N}$ of an IEN system N preserves all the transitions of N, as well as the flow relation and the initial and final markings. Furthermore, the new flow relation $\breve{F}$ added to obtain $\widetilde{F}$ does not introduce new states in the reachability graph of the net, rather it only adds new arrows, and it does not interfere with existing transitions. This implies that the functionality of $\widetilde{N}$ includes that of N, so circuits designed for a non-Brownian mode of operation, also work when fluctuations of tokens are allowed. Delays of tokens due to fluctuations do not affect the correctness of a circuit's operation, since the circuit is delay-insensitive.

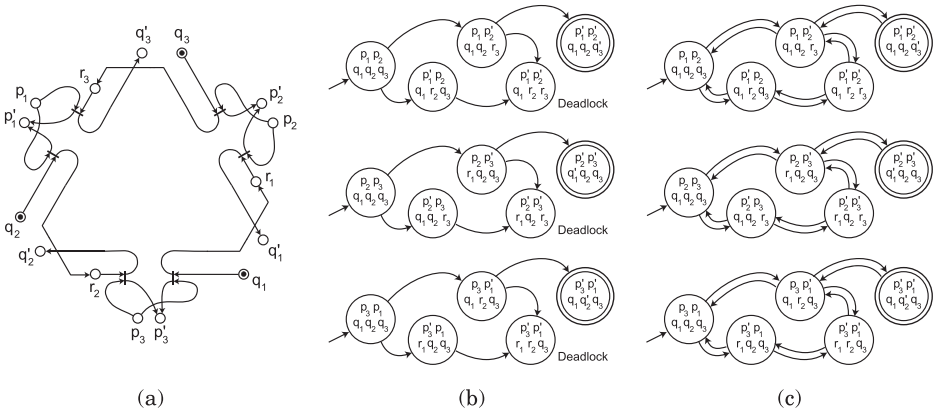


Fig. 13. (a) Simplified net of the TP-Tria in Figure 11. (b) Corresponding continuous-time Markov chain. The states are marked by the labels, each corresponding to a place that contains a token. The three initial states are pointed to by the arrows at the left, and the final states are indicated by double circles. (c) Continuous-time Markov-chain corresponding to the Brownian equivalent of the net in (a).

The added value of allowing tokens to fluctuate is that, due to the reverse transitions in the net of a Brownian circuit, it is possible to undo transitions, including those that lead to deadlocks. A circuit thus gains the ability to backtrack out of deadlocks, and this is why a TP-Tria constructed from three Brownian T-elements according to the circuit design in Figure 11 has no deadlocks.

Preventing deadlocks alone is not sufficient, though, because there is one more cause in a Petri Net of a process's failure to progress, and this is a so-called *livelock* [Sifakis 1980]. In terms of a Brownian circuit, a livelock occurs when for example, a token input to a choice line of a T-element becomes synchronized with a token input to a base line—possibly through some mutual interactions or some periodic behavior—in such a way that when one token moves away from the T-element, the other moves toward it, and the other way around. The T-element will then never produce output, since it requires both tokens to be at its input terminals at the same time. This type of pathological behavior is ruled out by imposing a sufficient degree of randomness on the model, which is achieved here by modeling the Brownian equivalent of a net as a Stochastic Interfaced Elementary Net system (see Definition 2.9). This allows the use of the extensive apparatus available for Continuous-Time Markov Chains (CTMC):

THEOREM 5.2 ([MOLLOY 1982]). *Any finite-place, finite-transition, marked stochastic Petri Net is isomorphic to a discrete-space continuous-time Markov chain.*

The isomorphism used in this theorem is a mapping between the state space of a stochastic Petri Net's reachability graph and the state space of a CTMC. Since every transition in the Brownian equivalence of a net is accompanied by a reverse transition—both transitions having a positive transition rate—the corresponding Markov chain contains a transition and its reverse transition—both also with positive rate—between the states representing the corresponding markings in the Petri net. Whenever a state S_1 of a Petri Net can be reached from a state S_2 and the other way around, the corresponding states M_1 and M_2 in the Markov chain are said to be *communicating*. Communication between states is an equivalence relation, so it divides the set of states of the Markov chain in equivalence classes. Each class contains at least one state corresponding to an initial marking in a SIEN system.

Figure 13(a) shows a non-Brownian stochastic net (with transition rate left out) and Figure 13(b) its corresponding CTMC, in which no states are communicating. When

reverse arrows are inserted in the CTMC, we obtain a CTMC with three equivalence classes of communicating states Figure 13(c)). This CTMC corresponds to the Brownian equivalent of the net in Figure 13(a). Each equivalence class in the CTMC in Figure 13(c) represents a single Markov chain that is *irreducible*, because the chain is aperiodic due to it being continuous-time (e.g. Steward [2009]). According to a well-known result, all states in an irreducible Markov chain with a finite number of states are *positive recurrent*. This means that every state will be visited infinitely often with probability 1, and that the mean time between two subsequent visits of a state is finite.

Definition 5.3. Let the SIEN system $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{S}_i, \widetilde{S}_f, \Lambda)$ be the Brownian equivalent of the IEN system $N = (P, T, F, S_i, S_f)$. Then the set $\widetilde{S}_i$ can be partitioned into two subsets, namely

(1) The set of *computable* markings $\widetilde{S}_i^{(c)}$, for which it holds that for all $C \in \widetilde{S}_i^{(c)}$ there exists a $D \in \widetilde{S}_f$ such that $C \overset{*}{\rightarrow} D$,
(2) The set of *noncomputable* markings $\widetilde{S}_i^{(nc)}$, i.e., the markings not in $\widetilde{S}_i^{(c)}$.

Since these two subsets of $\widetilde{S}_i$ are disjoint, the CTMC corresponding to the SIEN system can be partitioned in two disjoint CTMCs, one containing equivalence classes corresponding to the computable markings, and the other covering the noncomputable markings. Figure 13(c) shows equivalent classes that are all computable, because a final marking can be reached for each of the possible initial markings. Markings are noncomputable when the corresponding equivalence class in the CTMC contains no final state. In this case, transitions will take place between the states of the equivalence class associated with a certain input, without there being an end to the process. In other words, we will find ourselves in an infinite loop of transitions, none of which leads to a final state—a typical case of livelock. For this reason, the following theorem only covers initial markings that are computable.

THEOREM 5.4. *Let the SIEN system $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{S}_i, \widetilde{S}_f, \Lambda)$ be the Brownian equivalent of the IEN system $N = (P, T, F, S_i, S_f)$. Then $\widetilde{N}$ is deadlock-free from the computable markings $\widetilde{S}_i^{(c)}$ to $\widetilde{S}_f$.*

PROOF. Let $C \in \widetilde{S}_i^{(c)}$ and $C' \in \mathrm{RE}(C)$. Then all markings $C' \in \mathrm{RE}(C)$ represent states in the corresponding Markov chain that are in the same equivalence class. Let $D \in \widetilde{S}_f$ be a state for which $C \overset{*}{\rightarrow} D$, then D corresponds with a state in the Markov chain that is also in the same equivalence class. We then obtain $C' \overset{*}{\rightarrow} D$, which implies that $\widetilde{N}$ is deadlock-free from $\widetilde{S}_i^{(c)}$ to $\widetilde{S}_f$. □

Apart from guaranteeing the absence of deadlocks, Theorem 5.4 also rules out livelocks resulting from initial markings in the set $\widetilde{S}_i^{(c)}$. For initial markings in the set $\widetilde{S}_i^{(nc)}$, however, livelocks do occur, because the SIEN system will fail to reach a final marking in this case.

COROLLARY 5.5. *The Brownian T-element is universal for the class of Token-Pass circuits.*

PROOF. The TP-Merge is implemented as a Brownian T-element, and the TP-Fork is designed according to Figure 9(b). The TP-Tria is constructed from the Brownian

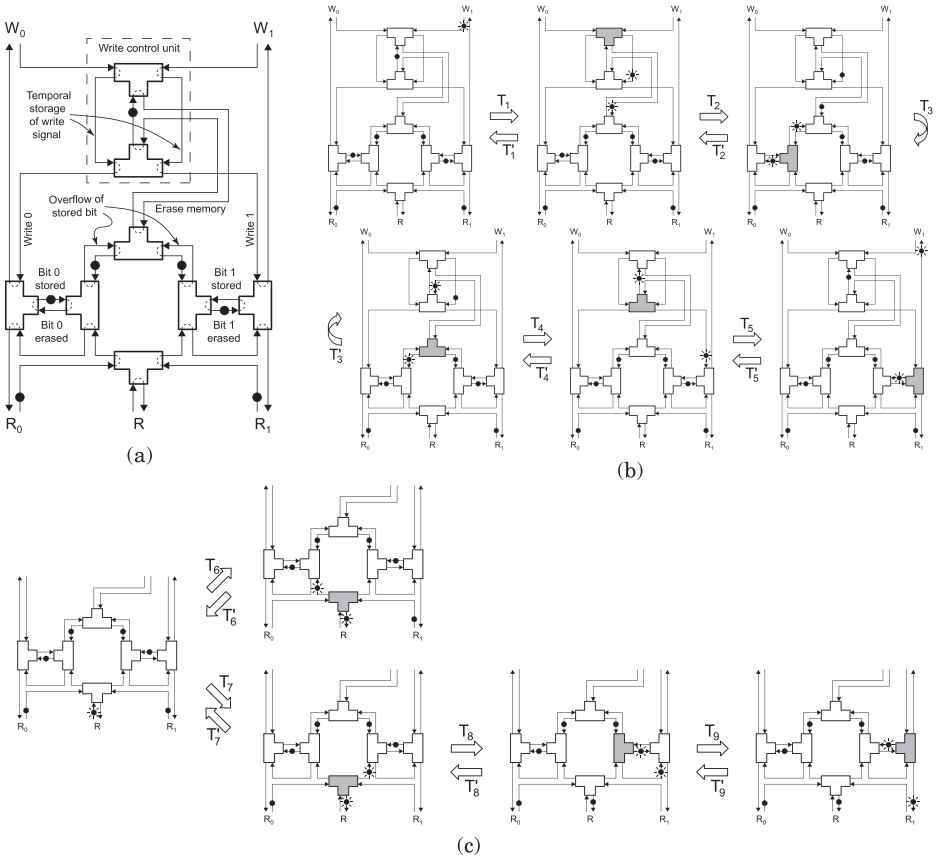


Fig. 14. Brownian 1-bit memory constructed from eight T-elements. The labels W_0, W_1, R, R_0, and R_1 have the same meanings as in Figure 8. (a) The memory in its initial state 0. The state is stored by the positions of two tokens in two different loops, one token at position *Bit 0 stored* and the other token at position *Bit 1 erased*. A token in the bit-stored position may cause a token to flow into the corresponding *Overflow of stored bit* line and back again, due to token fluctuations. This is an intermediate stage of erasing the memory during a writing operation. (b) Writing the state 1 in the memory. Tokens that have just moved to a new position are encircled by rays, and the T-element just used in the operation is colored gray. Tokens can fluctuate forward and backward as part of the writing operation, for example, through steps T_2 and T_2' before bit erasure. The write control unit ensures that erasure takes place (step T_3) before a new bit is written (step T_5). The writing operation is acknowledged by a token output at the post-line of W_1. (c) Reading out the memory (only lower half shown). Again, tokens fluctuate forward and backward, leading to dead ends in the search (via T_6), from which backtracking occurs (via T_6'), but eventually the memory's state is output to the post-line of R_1 after step T_9. See also Video 3 in the Supplementary Online Materials.

T-element according to the design in Figure 11. The initial markings of the TP-Tria are computable. The corollary then follows from Theorems 3.5 and 5.4. □

The Brownian equivalents of the 1-bit memories in Figures 8(b) and (c) built entirely from T-elements are deadlock-free and livelock-free. Being designed without considerations of token fluctuations in mind, however, these memories are inefficient, since the searching behavior of fluctuations is only exploited inside their three TP-Trias, and not in other parts of the circuits. The 26 T-elements required to build the memory in Figure 8(c)—five for the TP-Merges, three for each of the four TP-Forks, and three for each of the three TP-Trias—can be reduced to merely eight T-elements if the 1-bit memory is redesigned such that fluctuations are exploited to a fuller extent (Figure 14).

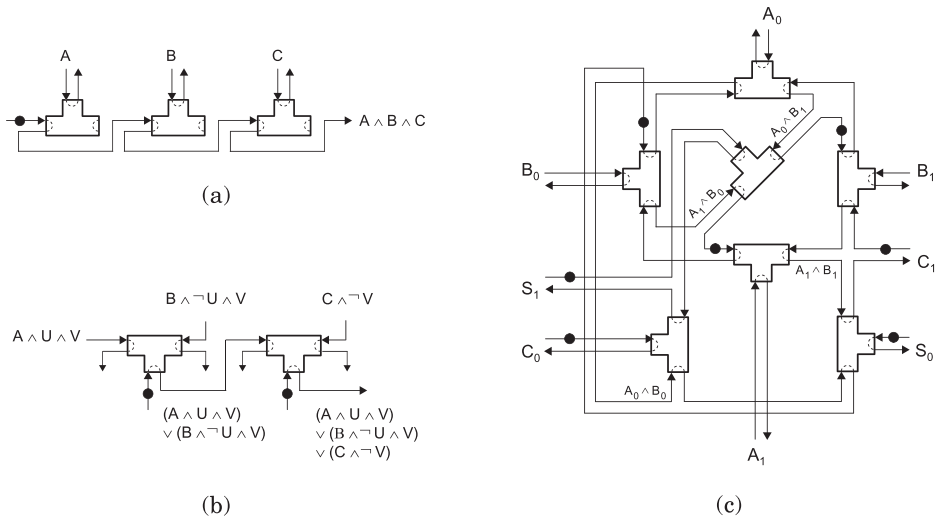


Fig. 15. (a) Logic circuit for the conjunction $A \wedge B \wedge C$. (b) Logic circuit for the disjunction of three lines that are logically exclusive: inputs are $A \wedge U \wedge V$, $A \wedge \neg U \wedge V$, and $C \wedge \neg V$, and the output is the disjunction of these three inputs at the bottom right of the circuit. The U and V variables are used to ensure that the three inputs are logically exclusive to each other. (c) Brownian Half-Adder constructed from seven T-elements. The lines A_0 and A_1 indicate the dual-rail encoded input A, and a similar representation is used for the other input B, the carry C, and the sum S. See also Video 4 in the Supplementary Online Materials.

Is there a systematic method to design Brownian circuits from T-elements? Unlike in conventional logic circuits, which use a single line to carry a binary value, two lines are required in token-based circuits. These lines, labeled 0 and 1, respectively, represent a binary value by the presence of a token on one of them. Called *dual-rail encoding* (e.g., Mead and Conway [1980, Ch. 7]), this scheme is also used in the previous 1-bit memories for the lines W_0 and W_1, as well as for the lines R_0 and R_1 (see Figures 8(c) and 14(a)). A dual-rail encoded AND-gate with input lines A_0 and A_1 to represent one input, input lines B_0 and B_1 to represent the other input, and output lines C_0 and C_1 to represent the gate's output is then defined by the terms $C_0 = (A_0 \wedge B_0) \vee (A_0 \wedge B_1) \vee (A_1 \wedge B_0)$ and $C_1 = A_1 \wedge B_1$. A logical function in a Boolean algebra can thus be expressed in canonical form as a sum of minterms of the individual dual-rail encoded values. Figure 15(a) illustrates the principle to construct a minterm of three variables from three T-elements. Generalized to a minterm of n variables, this results in a construction in which each variable (or complement of a variable) has the base line of a T-element assigned to it, whereas the minterm is represented by a line passing through one of the choice lines of each of the n T-elements. Since each minterm in a canonical form contains a unique combination of the input variables or their complements, only one minterm will have the logical value 1 for a given logical assignment of variables. In other words the minterms are logically exclusive to each other. A disjunction of three logically exclusive terms is implemented by the two T-elements connected to each other in the way shown in Figure 15(b), and this can be extended to more terms by feeding the right T-element's base line output side into a choice line of the next T-element.

Using these constructions, we design a half-adder in Figure 15(c). This half-adder consists of seven T-elements. It is left as an exercise to the reader to construct logic gates, like a (N)AND, (N)OR, or XOR, by stripping the half-adder. For these designs, only six T-elements per gate are required.

6. END CONDITIONS

When tokens are subject to fluctuations, a circuit may enter states repeatedly without it becoming clear when an output state is reached. What conditions should be satisfied in order to reliably read output from a Brownian circuit?

Definition 6.1. Let $N = (P, T, F, S_i, S_f, \Lambda)$ be a SIEN system. A non-empty set of non-empty markings E is an *End Condition* of N if

(1) for all markings $C \in E$ there exists a marking $D \in S_f$ such that $C \subseteq D$, and
(2) for all markings $D \in S_f$ there exists a marking $C \in E$ such that $C \subseteq D$.

An end condition can be interpreted as a set of representatives of the final markings of a SIEN system.

Example 6.2. The Brownian TP-Tria's SIEN system, labeled as in Figure 11(e), has the set of final markings

$$S_f = \{\{p'_2, p'_3, q'_1, q_2, q_3\}, \{p'_1, p'_3, q_1, q'_2, q_3\}, \{p'_1, p'_2, q_1, q_2, q'_3\}\}.$$

It has the end conditions (among others):

— $\{\{p'_2, p'_3\}, \{p'_1, p'_3\}, \{p'_1, p'_2\}\}$,
— $\{\{p'_2, p'_3, q'_1\}, \{p'_1, p'_3, q'_2\}, \{p'_1, p'_2, q'_3\}\}$,
— $\{\{q'_1, q_2, q_3\}, \{q'_2, q_1, q_3\}, \{q'_3, q_1, q_2\}, \{q'_1\}\}$,
— $\{\{q'_1\}, \{q'_2\}, \{q'_3\}\}$.

The following sets are not end conditions:

— $\{\{p'_2, p'_3, q'_1\}, \{p'_1, p'_3, q'_2\}\}$, because there is no marking that is a subset of $\{p'_1, p'_2, q_1, q_2, q'_3\}$,
— $\{\{q'_1\}, \{q'_2\}, \{q'_3\}, \{p_1\}\}$, because $\{p_1\}$ is not in a final marking.

Definition 6.3. Let $N = (P, T, F, S_i, S_f, \Lambda)$ be a SIEN system. An end condition E of N is *monotonous* if for all elements $B \in E$, for all $S \in S_i$, and for all markings $C \in \mathrm{RE}(S)$, it holds that there exists a $D \in S_f$ and a firing sequence $f = [t_1, ..., t_n\rangle \in \mathrm{FS}_C^D$ with the sequence of markings $C = C_0, ..., C_n = D$ determined by $C_{i-1} \left[t_i\right\rangle C_i$ for all $i \in \{1, ..., n\}$ such that: $C_{i-1} \cap B \subseteq C_i \cap B$.

A monotonous end condition indicates that a SIEN system is in a final marking, or close to a final marking to the extent that no backtracking of tokens already in a place that is part of a final marking is necessary to reach a completed final marking. This absence of the need to backtrack implies that once a token reaches a place that is in an element of a monotonous end condition, it can be kept there, without this preventing other tokens from reaching their places in the final marking.

The last two end conditions in Example 6.2 are monotonous, but the first two are not, because they contain the places p'_1, p'_2, and p'_3: some markings with tokens in these places require backtracking to reach a final marking (e.g. compare with Figure 11(e)).

Definition 6.4. A monotonous end condition E_m is *minimal* if for all markings $C_m \in E_m$ there is no monotonous end condition E with a configuration $C \in E$ such that $C \subsetneq C_m$.

Monotonous end conditions that are minimal allow us to decide whether a Brownian circuit has completed its operation through observing a minimal set of places for the

(a) (b)

Fig. 16. (a) Ratchet on a line and the corresponding net. (b) Ratchets on the output post-lines of a Brownian TP-Tria. These ratchets will not introduce deadlocks, while the ability to backtrack out of deadlocks inside the TP-Tria is preserved.

presence of tokens. A minimal end condition tends to correspond with the set of output post-lines in a Token-Pass circuit, or a subset thereof. The fourth end condition $\{\{q'_1\},\{q'_2\},\{q'_3\}\}$ in Example 6.2 is minimal. A minimal end-condition is not necessarily unique. The Brownian equivalent of the T-element's net in Figure 3(b) has the two minimal end conditions $E_m = \{\{c'\}\}$ and $E_m = \{\{a'\},\{b'\}\}$, if the set S_f of final markings is defined by $S_f = \{\{a',c'\},\{b',c'\}\}$.

7. SPEEDING UP BROWNIAN CIRCUITS

Fluctuations of tokens allow us to avoid deadlocks, but this is a mixed blessing, since it may require a long time to settle a circuit into its final state. How can Brownian circuits be sped up without giving up their advantages? One way of achieving this is to limit the extent to which tokens undergo fluctuations by minimizing the lengths of lines. Given that the mean 1-dimensional displacement of a particle through Brownian motion is proportional to the square root of time [Einstein 1956], the expected time for a token to move along a line of length L is of the order $O(L^2)$. This significant overhead can be mitigated by keeping lines short.

Further speed-ups can be achieved by using *ratchets*. A ratchet is a device that is attached to a line to restrict the motion of tokens to one direction, somewhat like a diode in an electronic circuit. Ratchets are well-known in nature, where they are thought to play a crucial role in the operation of molecular motors [Hänggi et al. 2005; Reimann 2002]. In the representation of a Brownian circuit as a net, the insertion of a ratchet results in the removal of a backward transition, as in Figure 16(a). Placement of ratchets at constant-order distances D from each other will speed up the circuit by a factor of $O(L/D)$.

When ratchets are placed on all lines of a Brownian circuit according to the direction that tokens are supposed to flow, the searching ability of the circuit will be taken away. This speeds up the circuit, but it leaves us where we departed: non-Brownian circuits, which may contain deadlocks. To avoid such situations, ratchets should be placed only at locations in which they do not interfere with backtracking. Section 6 suggests some candidates for such locations: at output post-lines that correspond to monotonous end-conditions.

Figure 16(b) shows the Brownian TP-Tria with ratchets placed in such a way. These ratcheted modules can be placed as-is in the Brownian equivalents of the 1-bit memories in Figures 8(b) and (c). In these memories, ratchets may be placed anywhere outside their three TP-Trias, because no backtracking is required there. Speeding up the Brownian 1-bit memory in Figure 14 may be done by placing ratchets at the post-lines of W_0, W_1, R_0, and R_1. Placing ratchets at the inside of this circuit, however, should be done with care, so as not to interfere with Brownian search processes.

In physical implementations the prodigal use of ratchets may be disadvantageous for yet another reason: ratchets consume energy, as shown by Feynman [2006]. There is thus an incentive to not overuse them, and to strive for a good trade-off between speed, energy consumption, and exploration ability.

8. CONCLUSIONS AND DISCUSSION

The presented Brownian circuit model uses fluctuations to explore the computational state space of token-based circuits. Central in this framework is a circuit primitive called T-element that is at the boundary of universality. Fluctuations provide tokens with the ability to search for computation paths in a circuit built from T-elements, allowing backtracking out of deadlocks and avoiding livelocks. It is this ability that sways circuits in favor of universality. The stochastic nature of search in Brownian circuits is confined to their internal operations, and is not reflected in their functionality, which is deterministic. Delays at which results emerge from output terminals of the circuits do not affect the correctness of the results; in other words, Brownian circuits are delay-insensitive.

Searching based on backtracking is a well-known strategy in computer science, and it is even the basis of the program language PROLOG [Clocksin and Mellish 2003]. The analogy goes even further: ratchets, proposed in this article to limit movements of tokens to one direction, are similar to the *cut predicates* in PROLOG. Both restrict search processes with the aim of speeding up computation, and both require a careful consideration of their placement so as to avoid adverse side effects to searching.

The placement of ratchets appears closely related to end conditions, as pointed out in Section 7. This article defines end conditions, but it lacks a method to compute them. Especially in large circuits, such methods will be important, since manual ad hoc methods are extremely time-consuming. Structural theory of Petri nets may be useful in this respect: *traps* in Petri nets, for example, appear to closely resemble end conditions, and algorithms are known to find them.

The searching and backtracking functionality afforded by fluctuations allows designs of circuits and circuit primitives to become simpler, since the functionality they supplement would otherwise need to be realized explicitly in the form of additional lines or states of circuit primitives, or in the form of more extensive circuitry. The Brownian 1-bit memory in Figure 14, for example, requires fewer resources—in terms of number and complexity of primitives—than the non-Brownian 1-bit memory in Figure 8(a).

The T-element—as useful as it is in the theoretical framework of this article—is not optimal from an implementation point of view. More simplicity is provided by the two Brownian circuit elements in Lee and Peper [2008], both of which have fewer input and output lines. These circuit elements, however, operate in only one mode, i.e., when tokens fluctuate. A nonfluctuating mode cannot be defined for them, unlike with circuits in this article that are based on T-elements. In other words, the main result of this article—that fluctuations of signals can make the difference between universality and the lack thereof—cannot be formally proved through the use of the simpler Brownian circuit elements in Lee and Peper [2008].

Brownian circuits may bring reduced energy consumption and relaxed operating temperatures in nanoelectronics a step closer, but there are still limitations to the fluctuations of tokens—and therefore to the allowable temperature under which a Brownian circuit operates. The model in this article requires tokens to fluctuate, but not to the extent that they leave a line or a module when they should not. Physical realizations of Brownian circuits will need to abide by these conditions. It is assumed in the proposed model that tokens do not interact with each other on lines or ratchets. In physical realizations, this condition may be difficult to satisfy for some types of tokens,

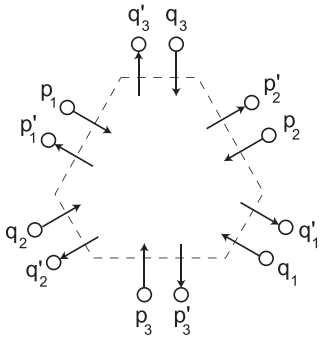


Fig. 17. Labeling of input and output places of TP-Tria's net for the proof of Theorem 4.4. The input pre-lines of the net are represented by the places p_1, p_2, and p_3. and the corresponding input post-lines by p'_1, p'_2, and p'_3 respectively. The output pre-lines are represented by q_1, q_2, and q_3, and the corresponding output post-lines by q'_1, q'_2, and q'_3 respectively.

for example for electrons, which are subject to electromagnetic interactions. Though ignored here, interactions between tokens can actually be exploited in certain Brownian circuits [Lee and Peper 2010], where repulsion of electrons could be used to steer them away from each other toward the proper input terminals of a module.

Petri nets—used for modeling Brownian circuits in this article—provide not only an abstract particle-like representation of signals, they are also compatible with the physical structures that will likely be employed to hold and propagate those signals. Quantum dots come to mind in this context [Reed et al. 1988], and their discrete nature is remarkably similar to the places in Petri nets. Tunneling of particles between Quantum dots is a stochastic phenomenon, and stochastic Petri nets (e.g., Murata [1989]) appear well-suited to describe such phenomena quantitatively, which will allow a more detailed evaluation of Brownian circuit performance, like speed. The fascinating journey beyond Moore's law has just begun, and the concepts discussed in this article may give a glimpse on the prominent role noise and fluctuations will play in it.

APPENDIX

Proof of Theorem 4.4

PROOF. Assume the net of a TP-Tria is Token-Pass and SPL. Let input and output places be labeled as in Figure 17. Throughout the proof, it is assumed that $i,j,k \in \{1,2,3\}$ and $i \neq j \neq k \neq i$, so $k = 6-i-j$. Define the following input markings of the net:

$$I_i = \{p_i, q_1, q_2, q_3\}$$
$$I_{ij} = \{p_i, p_j, q_1, q_2, q_3\}.$$

I_i corresponds to an incomplete (single) input to the TP-Tria, whereas I_{ij} corresponds to a complete (double) input to the TP-Tria. So, the set of initial markings is $S_i = \{I_{12}, I_{23}, I_{31}\}$. Likewise, the following output markings of the net are defined:

$$O_k = \{q'_k, q_i, q_j, p'_i, p'_j\}.$$

The set of final markings is $S_f = \{O_1, O_2, O_3\}$. According to the TP-Tria's definition, $I_{ij} \overset{*}{\rightarrow} O_k$ but the markings O_i and O_j are not reachable from marking I_{ij}. Moreover, O_i, O_j, and O_k are not reachable from any of the markings I_i, I_j, and I_k.

Since p_1, p_2, and p_3 are three different places, each with an empty pre-set, they belong to different paths. Let the path containing p_i be P_i ($i \in \{1, 2, 3\}$). Define the set T_{ij} of transitions as follows:

$$T_{ij} = \{t \in T \mid {}^\bullet t \cap P_i \neq \emptyset \ \wedge \ {}^\bullet t \cap P_j \neq \emptyset\}.$$

Let t_{ij} be a transition in T_{ij} with the shortest firing sequence from I_{ij}, including t_{ij} itself, and let the length of this firing sequence be L_{ij}. In a similar way, we define the sets T_{ik} and T_{kj}, and select corresponding transitions t_{ik} and t_{kj}, respectively. We then compare L_{ij}, L_{ik}, and L_{kj}, and select an element from $\{t_{ij}, t_{ik}, t_{kj}\}$ with a minimum-length firing sequence. Since i and j can be chosen freely at this point, we assume without loss of generality that t_{ij} is such an element.

Let $\hat{p}_i$ be the place in ${}^\bullet t_{ij} \cap P_i$ and $\hat{p}_j$ be the place in ${}^\bullet t_{ij} \cap P_j$. Due to the linearity of the paths P_i and P_j, these two places are uniquely determined. Since the net of the TP-Tria is SPL, at most one of the places in $\hat{p}_i$ and $\hat{p}_j$ has out-degree larger than 1, and the other place out-degree 1. Suppose (without loss of generality) that $|\hat{p}_i^\bullet| = 1$.

In the remainder of the proof, we show that the net N has a deadlock from the input marking I_{ik} to the output marking O_j, and that this deadlock occurs in $\hat{p}_i$. To this end it is shown that $\hat{p}_i$ is included in a marking in RE(I_{ik}) from which O_j cannot be reached.

First, we prove that transition t_{ij} is not enabled by any marking reachable from I_{ik}. Assume t_{ij} would be enabled by such a marking, then it would hold that $t_{ij} \in T_{ik}$ is an element with the shortest firing sequence in T_{ik} due to the definition of t_{ij}. We would then proceed in a similar way as for the input I_{ij}, and obtain the place $\hat{p}_k \in {}^\bullet t_{ij} \cap P_k$. Since P_i, P_j, and P_k are different paths, t_{ij} will be enabled by a marking containing all of the places $\hat{p}_i$, $\hat{p}_j$, and $\hat{p}_k$, but not by a marking that lacks $\hat{p}_k$. This contradicts the definition of t_{ij}, which implies that t_{ij} is in a firing sequence starting from I_{ij}. So, the assumption that t_{ij} can be enabled by a marking reachable from I_{ik} is false.

Second, we prove that place $\hat{p}_i$ is in a marking reachable from I_{ik}. Let $f_i = \tau_{i1}\tau_{i2}...$ be a firing sequence from I_i to a marking C_i, for which $\hat{p}_i \in C_i$, and let the associated sequence of markings be $I_i = C_{i0}, C_{i1}, ..., C_{in} = C_i$, whereby $C_{i,r-1} \left[\tau_{ir}\right\rangle C_{ir}$ ($r \in \{1, ..., n\}$). Inserting a token in p_k in the marking I_i results in the marking I_{ik}. Since p_k is an input place, it will not be in any of the markings C_{ir}, so f_i will also be a firing sequence from the marking I_{ik} to the marking $C_i \cup \{p_k\}$.

Place $\hat{p}_i$ is thus in a marking that is reachable from I_{ik}, but since t_{ij} is the only transition in $\hat{p}_i^\bullet$ (because of the out-degree being $|p_i'^\bullet| = 1$), it follows that a token ending up in place $\hat{p}_i$ due to input I_{ik} cannot be removed from $\hat{p}_i$ through the firing of t_{ij}. So, O_j is not reachable from marking $C_i \cup \{p_k\}$. Consequently, the net of the TP-Tria has a deadlock from S_i to S_f. □

ACKNOWLEDGMENTS

This work has greatly benefited from discussions with Professor Toshio Yanagida of Osaka University in Japan about the role of fluctuations in biology. We thank Dr. Kenji Leibnitz of the National Institute of Information and Communications Technology in Japan for his advice and feedback.

REFERENCES

Bennett, C. 1982. The thermodynamics of computation—a review. *Int. J. Theor. Phys. 21,* 12, 905–940.

Brown, R. 1828. A brief account of microscopical observations made in the months of June, July and August, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *Philosophical Mag. 4*, 161–173.

Clocksin, W. and Mellish, C. 2003. *Programming in Prolog*, 5th Ed. Springer-Verlag.

Commoner, F. 1972. Deadlocks in petri nets. Tech. rep. CA-7206/2311, Applied Data Research, Wakefield, Massachussets.

Dasmahapatra, S., Werner, J., and Zauner, K.-P. 2006. Noise as a computational resource. *Int. J. Unconventional Comput. 2,* 4, 305–319.

Einstein, A. 1956. *Investigations on the Theory of the Brownian Movement*. Dover Publications, New York, Chapter Translation of *Annalen der Physik 17,* 549–560, 1905, 1–18.

Feynman, R., Leighton, R., and Sands, M. 2006. *The Feynman Lectures on Physics*. Vol. I. Addison Wesley, 1–9.

Frey, E. and Kroy, K. 2005. Brownian motion: A paradigm of soft matter and biological physics. *Annalen der Physik 14,* 1–3, 20–50.

Hack, M. 1972. Analysis of production schemata by petri nets. Tech. rep. TR-94, Project MAC, MIT, Boston, MA.

Hänggi, P. and Ingold, G.-L. 2005. Fundamental aspects of quantum brownian motion. *Chaos 15,* 2, 026105.

Hänggi, P., Marchesoni, F., and Nori, F. 2005. Brownian motors. *Annalender Physik 14,* 1–3, 51–70.

Korotkov, A. and Likharev, K. 1998. Single-electron-parametron-based logic devices. *J. Appl. Phys. 84,* 11, 6114–6126.

Lageweg, C., Cotofana, S., and Vassiliadis, S. 2004. Single electron encoded latches and flip-flops. *IEEE Trans. Nanotechnol. 3,* 2, 237–248.

Lee, J. and Peper, F. 2008. On brownian cellular automata. In *Proceedings of Automata 2008*. Luniver Press, UK, 278–291.

Lee, J. and Peper, F. 2010. Efficient circuit construction in brownian cellular automata based on a new building block for delay-insensitive circuits. In *Proceedings of the 9th International Conference on Cellular Automata for Research and Industry (ACRI)*. S. Bandini, S. Manzoni, H. Umeo, and G. Vizzari Eds., Lecture Notes in Computer Science Series, vol. 6350, Springer Berkin, 356–364.

Lee, J., Peper, F., Adachi, S., and Mashiko, S. 2005. Universal delay-insensitive systems with buffering lines. *IEEE Trans. Circuits Syst. I, Reg. Papers 52*, 4, 742–754.

McAdams, H. and Arkin, A. 1999. It's a noisy bussiness! *Trends Genetics 15,* 2, 65–69.

Mead, C. and Conway, L. 1980. *Introduction to VLSI Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Meindl, J., Chen, Q., and Davis, J. 2001. Limits on silicon nanoelectronics for terascale integration. *Science 293,* 5537, 2044–2049.

Molloy, M. 1982. Performance analysis using stochastic petri nets. *IEEE Trans. Comput. 31,* 9, 913–917.

Murata, T. 1989. Petri nets: Properties, analysis and applications. *Proc. IEEE 77,* 4, 541–580.

Ono, Y., Fujiwara, A., Nishiguchi, K., Inokawa, H., and Takahashi, Y. 2005. Manipulation and detection of single electrons for future information processing. *J. Appl. Phys. 97,* 3, 031101–1–19.

Ovchinnikov, I. and Wang, K. 2008. Variability of electronics and spintronics nanoscale devices. *Appl. Phys. Lett. 92,* 9, 093503–1–3.

Patra, P. and Fussell, D. 1996. Conservative delay-insensitive circuits. In *Proceedings of the Workshop on Physics and Computation*. 248–259.

Peper, F., Lee, J., Abo, F., Isokawa, T., Adachi, S., Matsui, N., and Mashiko, S. 2004. Fault-tolerance in nanocomputers: A cellular array approach. *IEEE Trans. Nanotechnol. 3,* 1, 187–201.

Reed, M., Randall, J., Aggarwal, R., Matyi, R., Moore, T., and Wetsel, A. 1988. Observation of discrete electronic states in a zero-dimensional semiconductor nanostructure. *Phys. Rev. Lett. 60,* 6, 535–537.

Reimann, P. 2002. Brownian motors: Noisy transport far from equilibrium. *Phys. Rep. 361,* 2–4, 57–265.

Rozenberg, G. and Engelfriet, J. 1998. Elementary net systems. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, Lecture Notes in Computer Science Series, vol. 1491, Springer-Verlag, Berlin, UK, 12–121.

Sifakis, J. 1980. Deadlocks and livelocks in transition systems. In *Proceedings of the 9th Symposium on Mathematical Foundations of Computer Science (MFCS)*. Lecture Notes in Computer Science Series, vol. 88, Springer-Verlag, Berlin, UK, 587–600.

Sparsø, J. and Furber, S. 2001. *Principles of Asynchronous Circuit Design - A Systems Perspective*. Kluwer Academic Publishers.

Steward, W. 2009. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press.

Yamada, T., Akazawa, M., Asai, T., and Amemiya, Y. 2001. Boltzmann machine neural network devices using single-electron tunnelling. *Nanotechnol. 12,* 1, 60–67.

Yanagida, T. 2008. Fluctuation as a tool of biological molecular machines. *Biosyst. 93,* 1–2, 3–7.

Received March 2011; revised July 2011, September 2011; accepted February 2012