

Phase transitions in computation time of Brownian circuits

Kota Okajima^{1,*} and Koji Hukushima^{1,2}¹Graduate School of Arts and Sciences, *The University of Tokyo*, Komaba, Meguro-ku, Tokyo 153-8902, Japan²Komaba Institute for Science, *The University of Tokyo*, 3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan

(Received 12 March 2026; accepted 19 September 2026; published 16 October 2026)

Brownian circuits perform computations using stochastic transitions driven by thermal fluctuations. While the energetic costs of such fluctuation-driven computation have been extensively studied within stochastic thermodynamics, much less is known about its computational complexity, in particular, how computation time scales with circuit size. In this work, the computation time for explicitly designed Brownian circuits is numerically investigated via the first-passage time to a completed state. For arithmetic circuits such as adders, varying the forward transition rate induces a sharp change in the scaling behavior of the mean computation time with circuit size, from linear to exponential. This change can be interpreted as an easy-hard transition in computational time complexity. The transition suggests that, for meaningful computational tasks, achieving efficient polynomial-time computation generally requires a finite forward bias corresponding to a nonzero energy input. As a counterexample, we show that arbitrary logical operations can be reduced to an effective one-dimensional stochastic process in which the zero-bias limit lies within the computationally efficient (easy) regime. However, achieving such a one-dimensional normal form unavoidably leads to an exponential increase in circuit size. These results reveal a fundamental trade-off between computation time, circuit size, and energy input in Brownian circuits and demonstrate that phase transitions in time complexity provide a natural framework for characterizing the cost of fluctuation-driven computation.

DOI: [10.1103/zhgy-wgv4](https://doi.org/10.1103/zhgy-wgv4)

I. INTRODUCTION

Computation is the process of transforming symbols according to prescribed rules. These rules, called algorithms, can be implemented in various physical systems. In any such implementation, a computer encodes symbols into physical states and executes the computation through the time evolution of these states. The physical limitations of computation have been discussed from several perspectives. Feynman emphasized the intrinsic difficulty of simulating physical phenomena on conventional digital computers and proposed quantum computation as a form of natural computation [1]. Lloyd analyzed fundamental bounds on computational speed resulting from the limits imposed by quantum mechanics [2]. These studies demonstrated that computational performance is governed by both mathematical computational models and physical laws such as thermodynamics, quantum mechanics, and energy constraints. Among these, thermodynamics plays a central role in constraining computations implemented in noisy, fluctuation-driven physical systems.

From a thermodynamic perspective, computers differ fundamentally from conventional heat engines. While engines convert energy into useful mechanical work, computers expend energy to process information, and in most physical

implementations, the supplied energy is ultimately dissipated as heat [3–6]. For thermodynamic engines, the efficiency limits and trade-offs are rigorously constrained by the second law of thermodynamics. Because computation is also a physical process, it must obey the second law, implying unavoidable entropy production and fundamental trade-offs among energy dissipation, speed, and reliability. Thus, thermodynamics is an effective tool for analyzing the energy cost of information processing. However, how these thermodynamic constraints translate into computational cost in the sense of computational complexity, namely, how the computation time scales with a problem or circuit size, remains an open and actively debated problem [6].

The pioneering work of Landauer and Bennett established the foundations of computation physics by explicitly connecting information processing with thermodynamics. Landauer argued that information is physical and that logically irreversible operation necessarily imposes a minimum thermodynamic cost, which is known as Landauer's principle [3,7]. Bennett extended this perspective by investigating computational processes within idealized physical systems and clarifying how computation can proceed reliably despite being driven by microscopic fluctuations [4]. This research line emphasized the significant impact of thermal noise on computational dynamics and motivated the development of computational models that are both robust against and driven by such fluctuations, together with analyses of their computational cost.

Standard mathematical models of computation include Turing machines and logic circuits. Bennett proposed a physical realization of such models based on particles or molecules

*Contact author: k.okajima.phys@gmail.com

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

in Brownian motion [4]. In this framework, the computational state is encoded in particle configurations and coupling, and state transitions occur only when the particles satisfy specific local conditions. Thermal fluctuations drive system dynamics. As the transition rules are fixed by the circuit design, thermal fluctuations do not yield incorrect results; instead, they induce stochastic forward and backward motions along valid computational paths. With a sufficient forward bias, the system eventually reaches the completed state.

Additionally, extensive research on Brownian motors and molecular machines has revealed that useful tasks can be performed autonomously by actively utilizing thermal fluctuations. In such systems, directional processing emerges from stochastic transitions combined with structural asymmetries without requiring externally imposed step-by-step control. These fluctuation-driven mechanisms provide a concrete physical basis for Brownian computations and demonstrate how thermal fluctuations themselves can serve as computational resources [8–11]. Related concepts are also being actively investigated in biological contexts, such as the energy cost of information processing, including DNA replication and transcription in living cells.

Such systems are inherently nonequilibrium. In recent years, information thermodynamics has developed as a framework to systematically and quantitatively describe the interplay between information processing and thermodynamics [12–17]. Within this framework, entropy production and work associated with fundamental information-processing operations, such as measurement, memory operations, and feedback control, can be expressed in terms of information-theoretic quantities. Furthermore, advances in stochastic thermodynamics have yielded universal inequalities related to speed, precision, and energy consumption in stochastic processes, including the fluctuation theorem and thermodynamic uncertainty relations. Related analyses have also been applied to computational models driven by Brownian motion [5, 18–23].

Several recent studies have further developed the connection between stochastic thermodynamics and computation. Thermodynamically consistent stochastic circuit models based on elementary logic gates have been used to analyze trade-offs among accuracy, speed, and dissipation in low-dissipation computing [24]. More general thermodynamic frameworks have also been developed for computations with stochastic stopping times, unidirectional transitions, and absolute irreversibility, leading to universal relations and bounds on thermodynamic costs [25]. In the context of Brownian circuits, computation-time distributions have also been studied by defining the computation time as a first-passage time (FPT) and relating its fluctuations to thermodynamic costs such as output detection and resetting [21]. These studies provide important foundations for understanding the energetic and thermodynamic costs of stochastic computation.

However, these stochastic thermodynamic analyses do not characterize how the physical computation time scales with circuit size, which is the central question from the perspective of computational complexity theory. Motivated by this question, we investigate the physical computation time of Brownian circuits from both theoretical and numerical perspectives. We define the physical computation time as the first-passage time from an initial input state to the final output

state and develop a general framework for analyzing its dependence on circuit size and transition rates. We then apply this framework to explicitly designed Brownian circuits that implement concrete arithmetic operations. By analyzing several adder circuits, we observe that varying the forward transition rates leads to a qualitative change in the scaling behavior of the mean first-passage time. This scaling changes from linear to exponential scaling with circuit size, which we interpret as an easy-hard phase transition.

Here, we use the term “phase transition” in a dynamical and computational sense. The term does not refer to an equilibrium thermodynamic singularity, but to a qualitative change in the asymptotic scaling of a dynamical observable in the large-circuit-size limit. This limit plays a role analogous to the thermodynamic limit. For finite circuits, the change is rounded, whereas in the asymptotic limit distinct scaling regimes can be identified. This usage is analogous to easy-hard phase transitions discussed in statistical-physics approaches to computational problems, where typical computational behavior can change sharply with control parameters, including changes between polynomial and exponential scaling of computation time [26, 27].

These observations emphasize a fundamental trade-off between time, space, and energy in the context of Brownian computations. Although similar trade-offs are familiar from thermodynamic considerations, our analysis suggests that they can also be understood from a complexity-inspired perspective by focusing on the scaling of physical computational time with circuit size. This computation-based perspective provides a complementary framework for understanding the performance limits of Brownian circuits beyond purely thermodynamic bounds.

The remainder of this paper is organized as follows. Section II introduces the Brownian-circuit model, including the conservative join (CJoin) gate, the valid computational state space, and the computational cost used in this work. Section III defines the Brownian-circuit architectures considered below. We first introduce circuit modules and the boxing representation, and then describe modular adder circuits and the sum-of-products (SoP) construction as a one-dimensional architecture. Section IV formulates the stochastic dynamics of these circuits in terms of state-transition diagrams and first-passage time. This section also reviews the one-dimensional first-passage problem, introduces an embedding method for structured graphs, and discusses an effective balance condition for general circuits. Section V presents numerical results for the first-passage computational times. We first use the one-dimensional and sum-of-products circuits as a benchmark for the finite-size scaling (FSS) analysis, and then apply the same framework to modular adders to examine architecture-dependent easy-hard transitions. Finally, Sec. VI discusses the implications of these results for computational time, circuit size, energy bias, and circuit design, and concludes the paper. Technical derivations and additional examples are provided in the Appendixes.

II. BROWNIAN-CIRCUIT MODEL AND COMPUTATIONAL COST

This section introduces the fundamental components of Brownian circuits and discusses their computational costs.

First, we describe the basic building blocks that support particle movement and logical operations. Next, we examine how these elements form a state-transition diagram that represents the circuit's dynamic behavior. Finally, we discuss how to measure the computational cost, focusing on how circuit size and structure affect the overall computational performance.

A. CJoin gate and valid computational states

A Brownian circuit [4,28] is a computation model constructed using Brownian particles, pathways (referred to as “wires”), and gates. Brownian particles move along these wires, transitioning from one gate to others along the wires. The gates control these transitions by switching the connections between the wires on or off. As in conventional computational circuits, a Brownian circuit is designed to implement logic mapping from inputs to the corresponding outputs. Logic states are represented by the positions of multiple Brownian particles in the circuit. Gates define the transitions between these states by enabling particles to move from one state to another along the pathways. Because Brownian particles exhibit random motion owing to thermal fluctuations, active control of the particles is not necessary. Instead, the logical operations of the circuit are realized through the careful design of wires and gates.

In this study, we use the CJoin gate as the primary component of our Brownian circuits [28,29]. The CJoin gate comprises of two pairs of wires, as shown in Fig. 1. The gate has four wires with two inputs and two outputs, grouped into two pairs. A transition occurs when two particles simultaneously occupy one pair of wires, causing them to move toward the other pair.

Here, we restrict the dynamics to configurations that encode valid computational states, as shown in Fig. 2. Each logical variable is represented by a pair of complementary wires such as “A” and “not A,” and exactly one particle occupies one of the two wires. Configurations with no particles on either wire, or with particles on both complementary wires, are considered invalid logical configurations and are not included in the computational state space. We assume that the initial state is prepared within this valid subspace. Since CJoin transitions conserve the number of particles and move particles only between prescribed wire pairs, a properly designed circuit maps valid configurations to valid configurations. Thus, starting from a valid initial state, the dynamics remains within the valid computational subspace.

Each CJoin gate transition is bidirectional, with forward and backward directions. A forward transition progresses the computation, representing the movement from a prelogic state to a postlogic state as part of the logic mapping step. Although in principle each gate may have its own transition rate, we assume throughout this study that all the gates share a common forward transition rate and a common backward transition rate, which may differ from each other.

To build a Brownian circuit, CJoin gates are interconnected via branching wires, allowing particles on a wire to interact with multiple connected gates. Because each gate's transition depends only on the simultaneous presence of two particles in the corresponding wire pair, such wire bifurcations do not interfere with logical operations when the circuit is designed

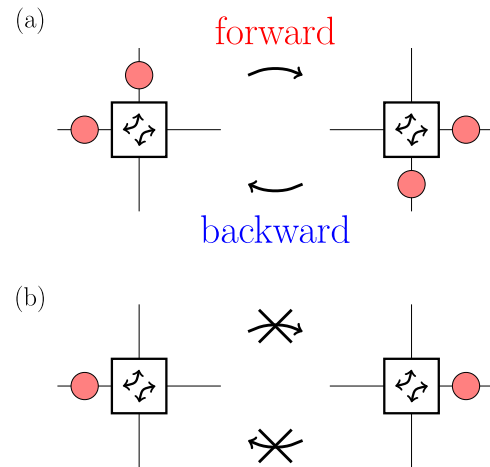


FIG. 1. Schematic diagram of the CJoin gate and the configurations considered in the present Brownian-circuit model. (a) Valid CJoin configurations and the bidirectional transition between them. The gate has four wires, top, bottom, left, and right, grouped into two pairs: top-left and bottom-right, as indicated by the arrows inside the box. A CJoin transition occurs only when two particles simultaneously occupy one of these designated pairs, and the transition moves them to the other pair. The forward transition moves the particles from the input pair to the output pair, whereas the backward transition corresponds to the reverse process. (b) Examples of configurations that are not included as valid CJoin transitions in the computational state space. A configuration with only a single particle does not satisfy the two-particle condition for activating the CJoin gate. A configuration in which all four wires are occupied is outside the valid logical state space assumed in this work. Starting from a properly prepared logical input state, the CJoin dynamics preserves the number of particles and does not generate such invalid configurations.

to preserve the valid computational subspace. Even when the gates and wires are entirely functional, certain factors may interfere with circuit operations. For example, multiple particles may occupy a single wire, or a gate could become inoperative owing to improper input. These problems can be avoided by

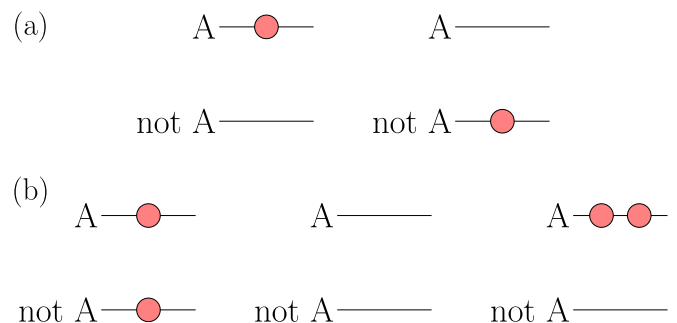


FIG. 2. Valid and invalid configurations for a logical variable represented by a pair of complementary wires. (a) Valid logical states. Exactly one particle is placed on either A or $\bar{A}$. (b) Invalid configurations. The three examples shown correspond to both complementary wires occupied, neither wire occupied, and two particles occupying the same wire. These configurations are excluded from the valid computational state space.

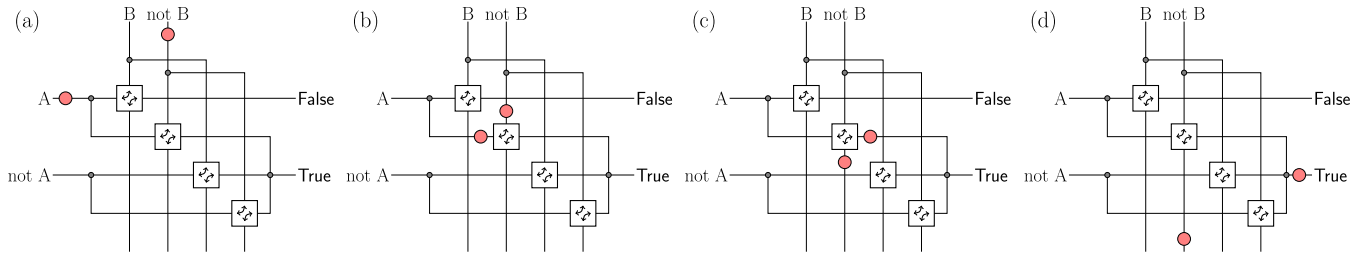


FIG. 3. NAND circuit and its transition process in a Brownian circuit: A and $\text{not } B$ are the inputs. The circuit progresses through four states to reach the True output. White boxes represent CJoin gates, black circles represent connected junctions of wires, and blue circles represent Brownian particles. (a) Initial state: Brownian particles are placed on wires A and “not B .” (b) Particles search for the correct gate by moving along their respective wires. (c) A gate transition moves the particles to the next wires. (d) Final state: The particles reach the output wires, resulting in the True output.

appropriately designing the Brownian circuit, as discussed in Sec. III.

As a concrete example of construction using CJoin gates, we demonstrate a NAND logic gate implementation [30]. The NAND circuit, shown in Fig. 3, embeds two logic variables, A and B in four wires: “ A ,” “not A ,” “ B ,” and “not B .” The presence of a particle on a wire corresponds to the logic state True, whereas its absence corresponds to False. Each logic state is represented by a specific combination of two particles placed on four wires, and only four of the $4C_2 = 6$ possible configurations correspond to valid logic configurations.

In Fig. 3(a), particles are initially set on the wires labeled “ A ” and “not B ” corresponding to the input state $A = \text{True}$ and $B = \text{False}$. This is a valid logic state from which the corresponding gate operation can begin. This approach resembles differential logic in electronic logic circuits, though instead of comparing wire pairs, particles exclusively ensure logical consistency during circuit operations. Figures 3(b) and 3(c) show how specific input configurations activate particular gates. For instance, if $A = \text{True}$ and $B = \text{False}$, then the second gate from the top left becomes active. As Brownian particles move through the circuit, they eventually reach the output wires, yielding a logic output True for the NAND function, as shown in Fig. 3(d). The NAND function is implemented by connecting the output wires as specified to produce the corresponding logic output. Logical operations are expressed using standard Boolean notation: “+” denotes logical sum: OR, “-” denotes logical product: AND, and the negation operator “-” generates the logical complement of a given logical variable. For example, $\bar{A}$ denotes the negation of A .

The concept of “simple rules and components” enables various implementations of Brownian circuits. Although our discussion focuses on the CJoin gate, other gate types and design rules can also be employed. Brownian circuits using CJoin gates are computationally universal [28,29], and mathematical equivalences among different gate types have been established [28]. However, in this study, we restrict our discussion to circuits based on the CJoin gate and concentrate on analyzing their dynamical behavior.

B. Physical computational time, circuit size, and energetic bias

Once the thermodynamic model of a circuit is defined, we can discuss not only the energy cost but also the circuit’s mathematical properties. Computational complexity is

a mathematically defined measure of computational cost; for logic circuits, it is referred to as circuit complexity. In computational complexity theory, complexity is characterized by a function $f(n)$, which represents the maximum size of the required computational resource as a function of the input size n . Although complexity is often measured in a single dimension, such as time or space, real physical systems do not always allow a straightforward equivalence between time and space. In practice, a computing system can be redesigned to reduce the computation time at the expense of increased system size; however, this trade-off depends on the design strategy. Therefore, no universal measure exists for comparing the original and redesigned systems using a single complexity measure determined only from the behavior of systems.

Previous studies [6] identified multiple methods for classifying computational costs by distinguishing between the different physics quantities involved in the computational process. Their classifications are broadly applicable to physical computing systems and are not limited to Brownian circuits. However, by restricting our focus to Brownian circuits, we can interpret the components of computational cost more concretely in physical terms. Because Brownian circuits can be constructed using various gate types and can be interpreted as Brownian motion on a state-transition diagram, we consider the following three aspects to provide both mathematical and physical descriptions of the computational cost.

1. Computation time

We define the computation time of a Brownian circuit in terms of the FPT of Brownian particles. The FPT is the time required for all particles in the circuit to move from their initial positions to the completion of the computation. As described in Sec. IV A, a Brownian circuit can be represented as the motion of a single particle on a state-transition diagram. Thus, we consider the FPT of this effective single-particle Brownian motion in this representation. Because FPT is a probabilistic quantity, we define the computation time as its mean value.

2. Circuit size

We define circuit size as the total number of gates. In this work, a module denotes a subcircuit with specified input and output wires that performs a prescribed logical operation. A module can be composed of several gates, and a circuit

constructed by connecting multiple modules can itself be regarded as a large module. Thus, when N identical modules, each consisting of m gates, are connected in series, the total number of gates is mN , which indicates the overall scale of the circuit. As the circuit size can vary significantly depending on the design, we introduce a practical circuit size in Sec. III for meaningful comparisons. This is analogous to circuit complexity in computational theory [31], where $\text{Size}(f)$ denotes the size of the circuit required to implement the mapping $f : \{0, 1\}^N \rightarrow \{0, 1\}$. The set of functions that can be implemented with size at most $G(n)$ is denoted as the complexity class $\text{SIZE}(G(n))$. Although the inherently probabilistic nature of Brownian circuits means that circuit complexity does not fully characterize their behavior, it remains a useful measure of computational cost.

3. Energy

To account for energy costs, we assume that transitions in CJoin gates obey the local detailed balance condition. Consider two states, i and j , in the state-transition diagram, with corresponding energies of E_i and E_j , and transition rates γ_{ij} from i to j , and γ_{ji} from j to i . These rates satisfy

$$E_i - E_j = \beta^{-1} \ln \frac{\gamma_{ij}}{\gamma_{ji}}, \quad (1)$$

where β denotes inverse temperature. This local detailed balance condition relates the energy difference between the two states to the ratio of their transition rates, thereby quantifying the energy cost per computational step [4]. In this study, we assume that all the gates within a given circuit have identical properties: Each gate has the same forward and backward transition rates, which may differ, and all the gates operate in precisely the same manner.

In the following analysis, we formulate the dynamics mainly in terms of the transition rates and study the scaling of the first-passage computation time. Thus, the easy-hard transition discussed below is primarily a dynamical transition. The local detailed balance condition provides its energetic interpretation: For a CJoin transition, changing the ratio between the rate in the computationally forward direction and that in the reverse direction corresponds to changing the energetic bias that drives the computation. In particular, a critical value of this rate ratio can be interpreted as the energetic bias required to compensate for backward stochastic motion. We do not attempt to evaluate the full thermodynamic cost of computation, such as costs associated with output detection or resetting, but focus on the relation between physical computation time, circuit size, and energetic bias.

III. BROWNIAN-CIRCUIT ARCHITECTURES

An important property of Brownian circuits is that they can implement arbitrary logical mappings; that is, they are functionally complete [28,29]. This means that any Boolean function can, in principle, be realized using Brownian circuits. In this section, we define the Brownian-circuit architectures considered in this work and explain how their sizes are counted. We first introduce circuit modules and the boxing representation used to describe them. We then present modular adder circuits, which provide the main examples for the

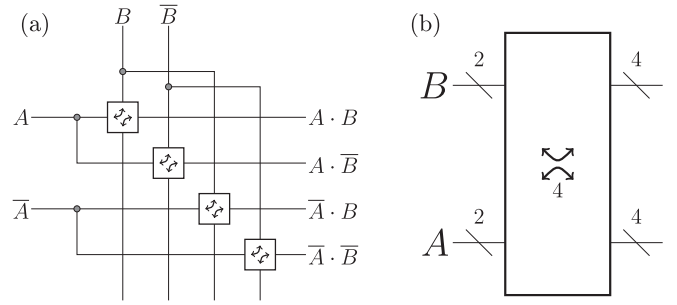


FIG. 4. (a) Two-particle I/O circuit explicitly constructed using CJoin gates. (b) Boxing representation of the same circuit. The number 4 inside the box indicates the number of gates used in this module. Each input logical variable is represented by a pair of wires, with the number 2 indicating the number of input wires. The eight output wires [four on the right and four on the bottom in panel (a)] are grouped into two pairs and represented compactly. The interpretation of each output is context dependent and specified as required.

numerical analysis. Finally, we describe the sum-of-products construction. Since the sum-of-products form is a standard normal form for Boolean functions, this construction provides a systematic way to realize arbitrary logical mappings. In the present work, it also provides an effectively one-dimensional reference architecture for the later dynamical analysis.

A. Circuit modules and boxing representation

Because CJoin gates operate only when two particles arrive simultaneously at a gate, each transition updates only part of the overall system state, even if the global state remains uniquely defined. Consequently, some circuit elements can operate independently in parallel, which we refer to as parallel elements. Conversely, certain gates must operate sequentially after the input state is set to complete the computation. The longest sequence of dependent gates is called the critical path, and the number of gates along this path is denoted by N_g . In deterministic circuits, such as conventional electronic circuits, the total computational time is typically proportional to the length of the critical path.

We construct CJoin-based circuits incrementally, beginning with a two-particle input/output (I/O) circuit. For two logical input variables and their corresponding outputs, $2^4 \times 2^4 = 256$ configurations are possible. Figure 4 shows how inputs A and B are converted into four possible logic products: $A \cdot B$, $A \cdot \bar{B}$, $\bar{A} \cdot B$, $\bar{A} \cdot \bar{B}$. By appropriately rearranging the connections between these outputs, all 16 possible two-variable logic functions can be achieved.

To simplify the schematic representation of complex circuits, we introduce an abstract graphical notation referred to as the boxing representation. Figure 4 presents this notation using a two-particle input/output circuit. The detailed circuit in Fig. 4(a) has a specific geometric layout, in which input and output wires may enter or leave the circuit from different directions. In the corresponding boxing representation shown in Fig. 4(b), this geometrical information is suppressed: By convention, input bundles are drawn on the left side of the box, and output bundles are drawn on the right side. The box represents the same subcircuit or module, but its internal gate-

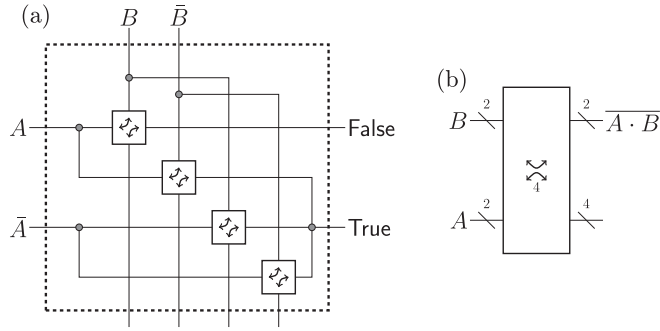


FIG. 5. NAND circuit and its boxing representation. (a) Detailed CJoin-gate implementation of the NAND circuit. The two-particle I/O circuit first separates the four wires corresponding to $A \cdot B$, $A \cdot \bar{B}$, $\bar{A} \cdot B$, and $\bar{A} \cdot \bar{B}$. The $A \cdot B$ wire is connected to the False output, whereas the other three wires are merged into the True output, thus implementing $\overline{A \cdot B}$. (b) Boxing representation of the same NAND module. The dotted box in panel (a) corresponds to the box in panel (b). The two input variables “A” and “B” are each represented by a pair of complementary wires, labeled 2, and the NAND output is represented by a pair of wires corresponding to False and True. The additional bundle labeled 4 represents output wires of the underlying two-particle I/O circuit that are not used for the NAND output in this particular module.

level structure is omitted. The number attached to the input and output ports indicates how many wires are grouped into each bundle. For example, a logical variable is represented by two complementary wires, such as “A” and “not A,” and the label 2 denotes such a pair of wires. More generally, a bundle of 2^k wires can encode k -bit logical information through the positions of particles on the corresponding wires. Thus, in Fig. 4(b), the two input variables “A” and “B” are represented by two input bundles of two wires, while the possible two-bit output configurations are represented by output bundles of four wires. When some wires in the complete output bundle are not used for the logical function assigned to a particular module, they are omitted from the functional description. Such wires are still part of the underlying circuit, but they are not used when the module is interpreted as a specific logical gate.

An example of this boxing representation applied to a NAND circuit is shown in Fig. 5. Figure 5 shows the detailed CJoin-gate circuit, and the dotted box indicates the subcircuit that is represented abstractly by the box in Fig. 5(b). The NAND operation is implemented by grouping the output wires of the two-particle I/O circuit according to the truth table. The circuit first distinguishes the four wires corresponding to the four input combinations, $A \cdot B$, $A \cdot \bar{B}$, $\bar{A} \cdot B$, and $\bar{A} \cdot \bar{B}$. Since the NAND output is False only when $A = B = \text{True}$, the wire corresponding to $A \cdot B$ is connected to the False output wire. The other three wires are connected to the True output wire. In this way, the four input-combination wires are grouped into two output wires representing the Boolean value of $\overline{A \cdot B}$.

In the boxing representation in Fig. 4(b), the two input variables A and B are drawn on the left side as two bundles of complementary wires, each labeled 2. The output $\overline{A \cdot B}$ is drawn on the right side and is also represented by a pair of wires corresponding to True and False. The additional bundles

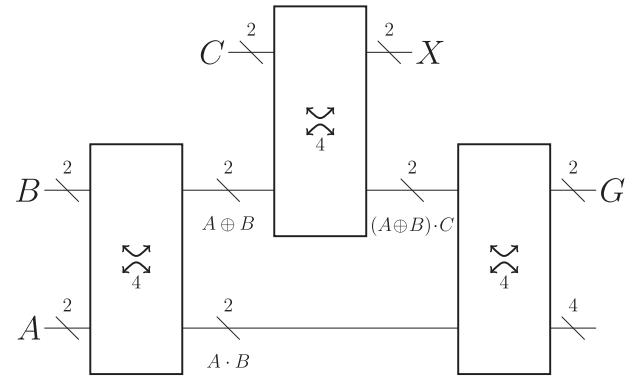


FIG. 6. Full adder circuit. A conventional full adder adapted to a Brownian circuit. A and B are the bits to be added, C is the “carry-in” from the lower digit, $X = A \oplus B \oplus C$ is the output of addition, and $G = A \cdot B + (A \oplus B) \cdot C$ is the “carry-out” to the upper digit.

labeled 4 represent output wires of the underlying two-particle I/O circuit that are not used for the NAND output in this particular module. These wires remain part of the underlying circuit and could be connected to other modules in a different circuit design.

B. Modular adder circuits

In this subsection, we present and compare three modular designs for implementing adders in Brownian circuits: the full, precede, and product adders. Although different adder implementations achieve the same logical function for binary addition, they differ in how the carry signals propagate through the circuit, resulting in different critical path lengths. These differences are not only structurally significant but also lead to marked differences in computational time, particularly in how it scales with the number of digits, as we will demonstrate in the following section. First, we introduce standard logic gate designs, including fundamental components such as AND, OR, and NAND [21]. Using the boxing representation described in the previous subsection, we present their schematic structures as follows.

The adder is one of the most fundamental components of logic circuits. Adders perform integer addition and, like other logic circuits, typically handle binary numbers. Consider an adder module: Each module takes a two-bit input and produces both the sum and the carry-out bit as output. A module without a carry-in bit is called a half adder, whereas one that receives a carry-in bit is called a full adder. By connecting multiple adder modules in series with carry-in and carry-out bits, we can construct a circuit that enables multidigit binary addition. In this study, we employ a module-design approach to control the overall circuit size by changing the number of modules. The total circuit size is proportional to the number of modules, with the proportionality constant depending on the circuit design used. Figure 6 shows a conventional adder circuit adapted for Brownian implementation. We refer to this circuit as a full adder.

Motivated by empirical insights from conventional circuits, where the implementation of carry propagation significantly affects performance, we introduce two alternative adder designs: the precede and product adders. Although these designs

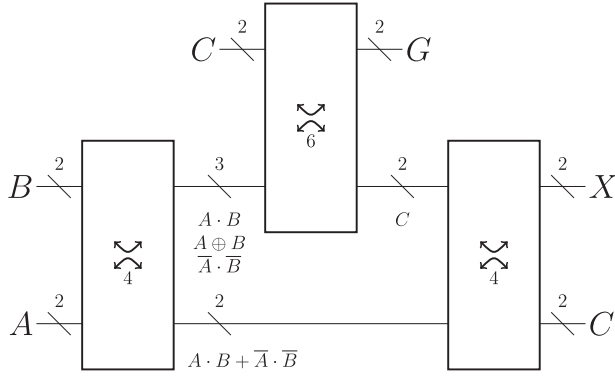


FIG. 7. Precede adder circuit. A modular adder design in which the carry-out signal is processed before the sum output. This circuit layout provides precedence to carry propagation within each module.

achieve the same logic function, they differ in terms of their structure and critical path length. By comparing their behavior in a Brownian circuit framework, we aim to examine how such architectural differences manifest in a thermally driven computational model.

The addition logic function common to all three designs is given by the following expressions for inputs A , B , and carry-in C :

$$X = A \oplus B \oplus C, \quad (2)$$

$$G = A \cdot B + (A \oplus B) \cdot C, \quad (3)$$

where X is the sum output and G denotes the carry-out. We embed this function into a Brownian circuit. In the standard full adder, the sum output X is computed before the carry-out G , which means that the next digit cannot proceed until the carry has propagated. Consequently, an adder consisting of N such modules has a critical path length of $2N + 1$, which includes both the computation and propagation delay of the carry signals. To examine the effect of this ordering on the computation in Brownian circuits, we design the precede adder, in which the carry-out G is computed and propagated before the sum output X . The circuit structure is shown in Fig. 7.

In conventional electronic logic circuits, a design such as the precede adder is generally not feasible because a single-step CJoin gate circuit can represent logic functions using only two logical variables (see Fig. 4). To further investigate the effect of the circuit structure on the computational performance, we introduce a third design: the product adder, as shown in Fig. 8. This design is based on the SoP logic design methodology, which will be discussed in detail in Sec. III C. In contrast to the full and precede adders, the product adder simultaneously produces both the sum X and carry-out G , resulting in a simpler structure with a shorter critical path of length $N + 1$.

Among the three adder designs discussed, the product adder has the shortest critical path. In conventional deterministic logic circuits, the computational time is largely determined by the critical path length, which is the longest path from input to output. In Brownian circuits, this notion corresponds to the longest path that a particle must travel

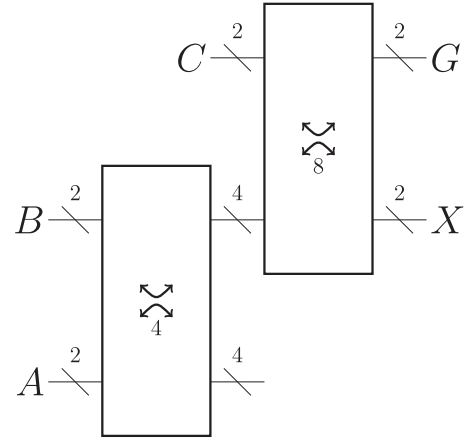


FIG. 8. Product adder circuit. A modular adder based on the sum-of-products logic design. Both the sum and carry outputs are produced simultaneously.

to complete a computation. Thus, the critical path length is a useful indicator of expected computation time. From this structural viewpoint, the product adder is expected to be the most efficient of the three. However, as we investigate in the following section through numerical simulations, the actual scaling behavior of the computation time with digit size can deviate from this expectation because of the stochastic dynamics of Brownian motion.

C. Sum-of-products construction as a one-dimensional architecture

In the previous subsection, we presented several concrete modular adder designs to demonstrate the implementation of specific logic functions in Brownian circuits. Although these designs demonstrate the validity of Brownian computation for specific tasks, how arbitrary logical mappings can be systematically implemented should be understood.

Here, we examine the functional completeness of Brownian circuits by demonstrating that any logic function can be realized using the SoP normal form. The SoP form, also referred to as the disjunctive normal form, is a standard representation in logic design. By explicitly constructing an SoP-based circuit using CJoin gates, we establish a universal construction scheme and clarify how its circuit size and computational cost scale with the input size. Although it is well known that CJoin gates form a universal gate set for Brownian circuits [28,29], our focus is on analyzing this universality from the perspective of scalability and efficiency.

In the SoP normal form, a logic function is expressed as a logical SoP in terms of the logic input variables. For example, the following function is expressed in SoP form:

$$f(A, B, C) = A \cdot B \cdot C + \bar{A} \cdot B \cdot C + B \cdot \bar{C}.$$

Because any logic function can be transformed into this SoP normal form, constructing a Brownian circuit for an arbitrary SoP function is sufficient for implementing all possible logic mappings. Below, we demonstrate the construction of such circuits using CJoin gates.

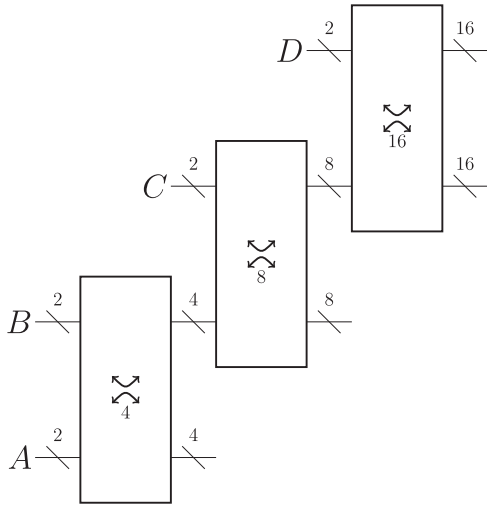


FIG. 9. SoP circuit with four variables.

Figure 9 shows an SoP circuit with four variables. These inputs are processed sequentially, with each stage maintaining the accumulated logic information. At each state n , the number of wires increases by 2^{n+1} , reflecting all possible combinations of input variables up to that point. Therefore, the final stage encodes complete $n + 1$ bit input information across the wires. This structure yields a one-dimensional state-transition diagram with a linear depth in the input size but an exponential number of gates and wires. Specifically, an SoP circuit with N inputs requires $N + 1$ stages and 2^N wires to represent all the input combinations. The number of gates required to construct such a circuit increases as follows:

$$\text{Size}(f) = 4 \times (2^{N-1} - 1) = \mathcal{O}(2^N).$$

The required size is at least when extending the circuit to produce N independent outputs,

$$\text{Size}(f) = 4(2^{N-1} - 1) + (N - 1)2^N = \mathcal{O}(N \cdot 2^N).$$

A detailed derivation of these expressions is provided in Appendix C.

IV. DYNAMICAL FORMULATION BY STATE-TRANSITION DIAGRAMS

Having introduced the circuit architectures, here we formulate their stochastic dynamics. In Brownian circuits, computation is carried out through the stochastic motion of Brownian particles along wires, with CJoin gates inducing transitions between computational states. Rather than explicitly tracking individual particle trajectories, we describe the dynamics in terms of a state-transition diagram. This diagram is a finite graph whose nodes represent computational states and whose edges represent allowed CJoin transitions. In this representation, Brownian-circuit computation becomes a continuous-time Markov process on a finite graph, and the FPT to the final output state defines the physical computation time.

This section develops the dynamical framework used in the numerical analysis. We first describe the state-transition diagram associated with Brownian circuits. We then review

the one-dimensional first-passage problem, which provides a reference case that exhibits a dynamical phase transition between polynomial and exponential scaling of the FPT. Next, we introduce an embedding method that maps the dynamics of a structured state-transition graph onto an effective one-dimensional process, clarifying how graph structure influences the balance between forward and backward transitions. Finally, we discuss how this balance condition can be extended to more general Brownian-circuit architectures.

A. State-transition diagram of Brownian circuits

As Brownian particles are classical, we treat each particle as localized at a discrete point along a wire. In practice, Brownian motion causes each particle to relax to an equilibrium distribution on its wire, and it remains on the wire until a CJoin gate transfers it to a neighboring wire. When the transition rate of the gates is sufficiently slow compared with the relaxation time of the Brownian motion along the wire, the system can be reasonably modeled as a Markov process in which transitions occur between wire-localized equilibrium states. In this coarse-grained picture, the position of a particle is fully described by the wire it occupies. By assigning logical variables to the wires, we can map the circuit dynamics onto a discrete logical state space, and the resulting stochastic evolution is governed by a master equation. This representation naturally leads to a state-transition diagram, where nodes represent discrete logic states determined by particle configurations, and edges represent allowed transitions induced by gates.

A Brownian circuit can be represented as discrete Brownian motion on a state-transition diagram. By adopting a coarse-grained representation of particle positions as logical variables of the wires, the particle arrangement on the wires directly corresponds to a set of logic variables, thereby forming a circuit logic state. Consequently, the Brownian time evolution of the circuit is equivalent to the motion of a single particle in the state-transition diagram, with the logic states serving as discrete coordinates.

Brownian circuits exhibit several key structural properties in their state-transition diagrams. First, for any connected diagram, a unique output state exists that represents completion of computation. Second, the input states belong to a single connected component in the same diagram if and only if they are converted to the same output state. These properties arise from the mathematical structure of logic circuits; every input deterministically maps to a unique output, although the mapping need not be injective, meaning that multiple input states may correspond to the same output. The specific structure of a state-transition diagram depends on the underlying circuit design, and different circuit architectures that implement the same logic function may produce distinct diagrams. However, some design methods yield diagrams with fixed structural patterns, even when logic mapping is not fully understood.

In this study, we adopt a modular design approach that allows the circuit size to be systematically varied. Each module is a functional unit with a fixed number of inputs and outputs that performs a prescribed local operation. We construct circuits that support multibit operations by connecting identical modules in series. The corresponding state-transition diagram

of such a modular circuit has a fixed depth, defined as the number of gates that a particle transfers from any initial state to the final output state, because all input states pass through the same sequence of operations to reach the output state.

B. One-dimensional first-passage analysis

We begin by introducing the notion of the FPT in a general context independent of any specific Brownian circuit implementation. In this subsection, we discuss a discrete and finite-state space consisting of N states [32,33].

Let $\mathbf{p}$ be a probability vector whose component p_i denotes the probability of being in the state i . The time evolution of $\mathbf{p}$ is governed by the master equation

$$\frac{d\mathbf{p}}{dt} = \Gamma \mathbf{p}, \quad (4)$$

where Γ is the transition rate matrix. We focus on the one-dimensional Brownian motion with forward and backward transition rates γ_+ and γ_- , respectively. The corresponding transition matrix has the following tridiagonal form:

$$\Gamma = \begin{pmatrix} -\gamma_+ & \gamma_- & 0 & \dots & 0 \\ \gamma_+ & -\gamma_+ - \gamma_- & \gamma_- & \dots & 0 \\ 0 & \gamma_+ & -\gamma_+ - \gamma_- & \dots & 0 \\ 0 & \dots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & \gamma_+ & -\gamma_- \end{pmatrix}. \quad (5)$$

The FPT is defined as the random time τ at which a trajectory first reaches the target state N starting from an initial state:

$$\{x(t) \mid x(\tau) = N; 0 \leq t < \tau, x(t) \neq N\}. \quad (6)$$

The probability distribution of τ is denoted by $P_f(\tau)$, and its l th moment is given by

$$\langle \tau^l \rangle = \int_0^\infty \tau^l P_f(\tau) d\tau. \quad (7)$$

In the deterministic limit $\gamma_+ \gg \gamma_-$, the FPT distribution is reduced to the convolution of individual exponential waiting times:

$$P_f(\tau) = \int_0^\tau dt_{N-1} \dots \int_0^{t_2} dt_1 P(\tau - t_{N-1}) \dots P(t_2 - t_1) P(t_1),$$

where each waiting time is distributed as

$$P(t) = (\gamma_+ + \gamma_-) e^{-(\gamma_+ + \gamma_-)t}.$$

This convolution simplifies the gamma distribution:

$$P_f(\tau) = \frac{(\gamma_+ + \gamma_-)^N \tau^{N-1}}{(N-1)!} e^{-(\gamma_+ + \gamma_-)\tau}. \quad (8)$$

Hence, the first moment and variance are

$$\langle \tau \rangle = \frac{N}{\gamma_+ + \gamma_-}, \quad (9)$$

$$\sigma^2 = \langle \tau^2 \rangle - \langle \tau \rangle^2 = \frac{N}{(\gamma_+ + \gamma_-)^2}. \quad (10)$$

Generally, in Brownian circuits, N corresponds to the critical path length, and the mean FPT is determined by the number of exponential waiting times that must be transitioned along that path.

In the continuous limit, which is valid when $\gamma_+ \simeq \gamma_-$, the dynamics of the one-dimensional master equation can be approximated using the Fokker-Planck equation (see Appendix A). Introducing the effective drift and diffusion coefficients

$$v \approx \gamma_+ - \gamma_-, \quad D \approx \gamma_+ + \gamma_-, \quad (11)$$

the mean FPT from the reflecting boundary at $-N$ and to the absorbing boundary at the origin has the classical asymptotic form summarized in Appendix B. For an asymptotically large N , this expression yields three distinct scaling behaviors depending on the sign of the drift:

$$\langle \tau \rangle = \begin{cases} \frac{N}{v}, & \frac{vN}{D} \gg 1, \\ \frac{N^2}{2D}, & \frac{vN}{D} \simeq 0, \\ \frac{D}{v^2} e^{|v|N/D}, & \frac{vN}{D} \ll -1. \end{cases} \quad (12)$$

The threshold separating these regimes is given by $v = 0$, i.e., by the condition $\gamma_+ = \gamma_-$. Thus, even in this simplest one-dimensional setting, the FPT exhibits a sharp change in its dependence on system size: The mean FPT increases linearly with N when $\gamma_+ > \gamma_-$, quadratically at $\gamma_+ = \gamma_-$, and exponentially when $\gamma_+ < \gamma_-$.

Although this crossover is a well-known classical feature of drift-diffusion processes, its interpretation in terms of computational complexity is noteworthy. In the context of Brownian circuits, FPT scaling with respect to N directly corresponds to the scaling of the computational time with respect to the circuit size. From this perspective, the change in asymptotic behavior at $\gamma_+ = \gamma_-$ naturally represents an easy-hard transition: Polynomial-time computation is possible only when the drift is non-negative, whereas negative drift imposes an exponential-time computation. The relationship between the computational costs of Brownian circuits is discussed in detail subsequently.

A similar qualitative change appears in the variance of the FPT. For $\gamma_+ > \gamma_-$, the variance increases only as $\mathcal{O}(\sqrt{N})$, implying that fluctuations in computation time remain relatively small even as the circuit size increases. In contrast, for $\gamma_+ < \gamma_-$, the variance increases exponentially, indicating that not only the mean computation time but also its uncertainty becomes exponentially large. From a computational perspective, this distinction is essential in the polynomial-time regime, where the computation is not only fast on average but also dynamically stable, whereas in the exponentially slow regime, the computation becomes intrinsically unreliable owing to enormous temporal fluctuations.

C. Embedding method for structured graphs

Generally, the state-transition diagram of a Brownian circuit is not strictly one dimensional. Nevertheless, many circuits have structural features that enable their dynamics to be coarse grained in an effective one-dimensional process. This reduction relies on the observation that the state-transition diagrams of Brownian circuits are N -partite graphs in which independent sets can be naturally ordered by their graph-theoretic distance from the final state.

The general embedding procedure is as follows:

(1) Label all states according to their distance from the final state.

(2) Group states with the same distance and sum their master equations to obtain an effective one-dimensional chain.

To illustrate this procedure, we examine two essential classes of graphs relevant to circuit architectures: complete trees and hypercubes.

As the first example, we consider a complete m -ary tree in which the root represents the final state and the leaves represent the initial states. Each nonleaf node has exactly m children. Thus, the tree is filled at every depth level, containing m^d nodes at depth d . To label the nodes, we assign to each state a word σ over an alphabet Σ . The alphabet has a sufficient number of symbols to assign to the children of each parent; for example, $\Sigma = \{0, 1, \dots, m-1\}$. The root node is labeled the empty word ϵ , and each child of state σ is labeled by appending a unique symbol $a \in \Sigma$ to produce the state σa . Conversely, if σ is nonempty, we denote σ^- the word obtained by removing its last symbol. Thus, σ^- is the parent of σ , and $(\sigma a)^- = \sigma$.

With this label convention, the master equation for the probability p_σ of being in state σ is

$$\frac{d}{dt}p_\sigma = \gamma_+ \sum_{a \in \Sigma} p_{\sigma a} - (\gamma_+ + m\gamma_-)p_\sigma + \gamma_- p_{\sigma^-}. \quad (13)$$

If l_σ denotes the depth (word length of σ), we define the probability of the state in $\{\sigma | l_\sigma = l\}$ as

$$p_l = \sum_{\sigma: l_\sigma = l} p_\sigma. \quad (14)$$

Summing the master equation over all the states at depth l yields a coarse-grained one-dimensional process,

$$\frac{d}{dt}p_l = \gamma_+ p_{l+1} - (\gamma_+ + m\gamma_-)p_l + m\gamma_- p_{l-1}. \quad (15)$$

At the boundaries $l = 0$ and $l = d$, we obtain

$$\frac{d}{dt}p_0 = \gamma_+ p_1 - m\gamma_- p_0, \quad (16)$$

$$\frac{d}{dt}p_d = -\gamma_+ p_d + m\gamma_- p_{d-1}. \quad (17)$$

This effective one-dimensional chain has one forward transition and m backward transitions per layer. Consequently, an easy-hard transition in FPT occurs when

$$\gamma_+ = m\gamma_-.$$

For $m = 1$, this is reduced to the conventional one-dimensional threshold $\gamma_+ = \gamma_-$. In contrast to the strictly one-dimensional case, where the easy-hard transition occurs at zero-energy cost, this tree-structured system exhibits a transition only at a finite energy cost determined by the branch number m .

As another example, we consider an N -dimensional hypercube. Each state is labeled by a bit string $\sigma \in \{0, 1\}^N$, and the Hamming weight n_σ counts the number of 1s in the string. We define the sets of neighboring states obtained by flipping a single bit:

$$\begin{aligned} S_\sigma^+ &= \{\sigma' | \text{obtained by flipping a } 0 \rightarrow 1\}, \\ S_\sigma^- &= \{\sigma' | \text{obtained by flipping a } 1 \rightarrow 0\}, \end{aligned}$$

such that the sizes of these sets are given by

$$|S_\sigma^+| = N - n_\sigma, \quad |S_\sigma^-| = n_\sigma.$$

Therefore, the master equation for the probability p_σ of being in state σ is

$$\begin{aligned} \frac{dp_\sigma(t)}{dt} &= -(|S_\sigma^+|\gamma_+ + |S_\sigma^-|\gamma_-)p_\sigma(t) \\ &\quad + \sum_{\sigma' \in S_\sigma^+} \gamma_- p_{\sigma'}(t) + \sum_{\sigma' \in S_\sigma^-} \gamma_+ p_{\sigma'}(t). \end{aligned} \quad (18)$$

To coarse grain the state, we group the states using their Hamming weight,

$$p_l = \sum_{\sigma: n_\sigma = l} p_\sigma,$$

and obtain the effective one-dimensional equation

$$\begin{aligned} \frac{d}{dt}p_l(t) &= -((N-l)\gamma_+ + l\gamma_-)p_l(t) + (l+1)\gamma_- p_{l+1}(t) \\ &\quad + (N-l+1)\gamma_+ p_{l-1}(t). \end{aligned} \quad (19)$$

In contrast to the complete tree, the hypercube does not exhibit a global balance between forward and backward transitions: The effective drift changes sign as l varies. Consequently, as will be shown later, the hypercube does not exhibit an easy-hard phase transition. Instead, the mean FPT increases exponentially with N for all values of γ_+/γ_- .

D. Effective balance condition for general circuits

The purpose of this subsection is to extract a general guiding principle from the exactly or partially reducible cases discussed above. The one-dimensional chain provides an exactly solvable reference case, in which the easy-hard transition occurs at $\gamma_+ = \gamma_-$. Complete tree structures provide another controlled example: After grouping states by their distance from the final output state, the dynamics closes into an effective one-dimensional process, and the transition point is shifted by the branching factor. These examples suggest that the transition is governed not simply by the microscopic forward and backward rates, but by their effective balance after the structure of the state-transition graph is taken into account. The following argument should therefore be understood as an effective criterion for general Brownian-circuit architectures, rather than as a theorem for arbitrary state-transition graphs.

Generally, state-transition diagrams of Brownian circuits have complex structures involving multiple branches and merges. Such complexity makes an exact reduction to a simple one-dimensional Markov process rarely possible. Nevertheless, Brownian circuits obey structural constraints that enable us to formulate a useful conjecture regarding the scaling behavior of their FPT.

Although a full reduction to one dimension is not possible for arbitrary circuits, we can still define a coarse-grained one-dimensional description by grouping states according to their distance from the final output state and summing the master equations within each group. This procedure yields an effective escape rate that governs the probability flow between adjacent groups. The ratio of the effective forward and

backward contributions can then be interpreted analogously to the ratio γ_+/γ_- in the strictly one-dimensional setting.

Because Brownian circuits implement deterministic logical mapping, each input corresponds to a unique output. As a result, multiple distinct initial states are mapped to unique output states via intermediate transitions. This structural feature implies that each intermediate state typically has exactly one forward transition but multiple backward transitions. The total backward transition rate, therefore, tends to exceed the forward transition rate. This does not mean that backward transitions are dominant. Rather, it indicates a generic structural bias toward backward motion, which must be compensated for by a sufficiently large forward rate to ensure efficient computation.

To quantify this imbalance, we introduce an effective factor a representing the average number of backward transitions per forward transition. This leads to the following generalized balance condition for the easy-hard transition:

$$\gamma_+ = a\gamma_- \quad (20)$$

In the purely one-dimensional case discussed in Sec. IV B, we have $a = 1$, recovering the standard threshold $\gamma_+ = \gamma_-$. In a complete m -ary tree, the exact coarse-grained dynamics gives $a = m$. These two limiting cases provide direct evidence that the critical point is controlled by the effective balance between forward and backward probability flows.

For more general Brownian-circuit architectures, where converging paths are unavoidable and backward transitions are more frequent, we expect that the convergence of computational paths generically gives $a > 1$, unless the state-transition graph can be effectively reduced to a one-dimensional chain. This consideration leads to the conjecture that, for general Brownian circuits, the easy-hard transition occurs when the forward rate compensates for the graph-induced backward bias characterized by a . Equivalently, unless the state-transition graph is effectively reduced to a one-dimensional chain ($a = 1$), the transition point shifts from $\gamma_+ = \gamma_-$ to $\gamma_+ = a\gamma_-$ with $a > 1$.

The SoP circuit discussed in Sec. III C is an important exception. Its state-transition diagram is effectively reducible to a one-dimensional structure and thus satisfies $a = 1$. However, as discussed later in Sec. VI, achieving this effective one-dimensionality requires an exponentially large circuit. Thus, the same viewpoint also clarifies the trade-off between dynamical efficiency and circuit-size cost.

V. NUMERICAL RESULTS FOR FIRST-PASSAGE COMPUTATIONAL TIME

We now apply the dynamical framework developed in Sec. IV to the Brownian-circuit architectures introduced in Sec. III. The computation time is evaluated as the FPT from an initial input state to the final output state in the corresponding state-transition diagram. We generate stochastic trajectories using the Gillespie algorithm [34] and analyze how the mean FPT scales with circuit size. For each pair of circuit size and the forward transition rate, the mean FPT is estimated from 10^2 – 10^3 independent stochastic trajectories, depending on the circuit size. For modular circuits, each parameter set is evaluated for a single randomly chosen input instance, and

the averaging is performed only over stochastic trajectories. The qualitative scaling behavior was confirmed to be insensitive to the choice of input configuration. In contrast, abstract circuit constructions such as the SoP circuit do not involve an ensemble of randomly chosen input instances [35]. This section has two purposes. First, we use one-dimensional and sum-of-products circuits as benchmark cases, where the expected transition between polynomial and exponential scaling can be directly tested by finite-size scaling. Second, we apply the same numerical framework to modular adder circuits and examine how the circuit architecture affects the existence and location of the easy-hard transition. This comparison allows us to identify which feature of the state-transition diagram controls the scaling of the physical computation time.

A. One-dimensional and sum-of-product circuits: Finite-size scaling benchmark

As discussed in Sec. III C, Brownian circuits can generally be represented using the SoP normal form with a one-dimensional state-transition diagram. Furthermore, as shown in Sec. IV B, the FPT in one dimension exhibits a qualitative change in the asymptotic behavior with respect to the circuit size, indicating a phase transition as a function of the gate transition rate in the continuous spatial limit. To confirm this phenomenon numerically, we performed stochastic simulations of Brownian circuits on a one-dimensional discrete lattice and examined how the FPT scales with the circuit size.

We employ the Gillespie algorithm [34] to generate stochastic trajectories of Brownian motion from the input to the output, following the transition rules defined in Sec. IV D. This approach provides the empirical FPT distributions. Specifically, we performed stochastic simulations for various values of the forward transition rate γ_+ while fixing the backward transition rate at $\gamma_- = 1$. The resulting mean FPT values for the circuit size averaged over multiple trajectories are shown in Fig. 10. Here, the circuit size of the SoP construction is characterized by the effective chain length corresponding to the number of logical states rather than the number of CJoin gates.

In this setup, the phase transition occurs at $\gamma_+ = \gamma_c = 1$. Hereafter, we denote the transition point for γ_+ by γ_c . At this transition point, the mean FPT follows a power law, $\langle \tau \rangle \propto N_g^\alpha$, with scaling exponent $\alpha = 2$, which is consistent with the continuous-space result in Eq. (12). For $\gamma_+ > \gamma_c$, the mean FPT scales linearly with N_g , including the deterministic limit. For $\gamma_+ < \gamma_c$, the FPT scales exponentially. These discrete-space simulations therefore support the analytical findings of the continuous-space model, confirming both the scaling regimes and critical transition rate.

To determine the value of γ_c more systematically, we use FSS analysis, which can also be applied to state-transition diagrams beyond one dimension, as we will discuss in the next subsection. If the mean FPT, $\langle \tau \rangle$, is obtained numerically as a function of γ_+ and critical path length N_g , we assume that, near γ_c , the mean FPT follows the FSS form

$$\langle \tau \rangle(\gamma_+, N_g) = N_g^\alpha f(N_g^\beta (\gamma_+ - \gamma_c)), \quad (21)$$

where α and β are scaling exponents, and f is a universal scaling function. The exponent α characterizes the algebraic

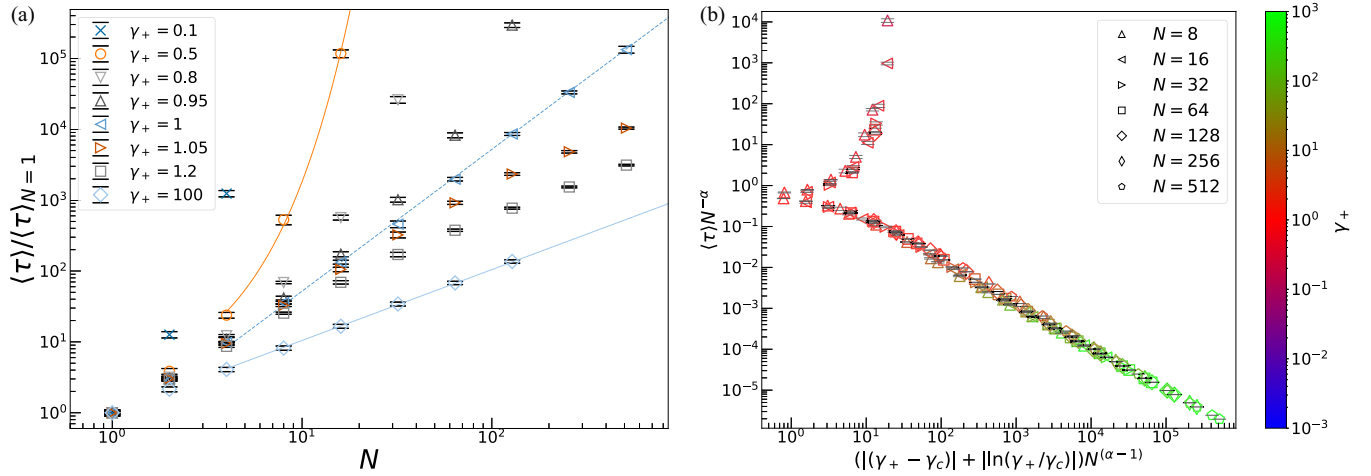


FIG. 10. Numerical results for the sum-of-products circuits obtained by Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed as the unit of time, $\gamma_- = 1$, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the initial computational state to the final output state. (a) Mean first-passage time as a function of the effective circuit length N , normalized by its estimate at $N = 1$. The reference curves represent exponential, quadratic, and linear scaling with N , corresponding, respectively, to the hard regime, the critical point, and the easy regime. (b) Finite-size scaling of the mean first-passage time using the scaling exponent $\alpha = 2.0$ and the critical transition rate $\gamma_c = 1.0$.

divergence of the mean FPT at the transition point. In contrast, the exponent β determines how the characteristic crossover size scales with the distance to the transition point, $N_g^* \simeq |\gamma_+ - \gamma_c|^{-1/\beta}$.

Consistent with deterministic-limit behavior, $\langle \tau \rangle \simeq N/\gamma_+$ requires that $f(x) \simeq 1/x$ as $x \gg 1$ and implies $\beta = \alpha - 1$. By applying this FSS form to our one-dimensional simulation results, we find that the scaling works well with $\alpha = 2$ and $\gamma_c = 1$, as shown in Fig. 10(b). In these scaling plots, we empirically include a small logarithmic correction to the scaling variable to improve the data collapse. This correction is irrelevant for $\gamma_+ > \gamma_c$; however, for $\gamma_+ < \gamma_c$ it slightly improves the visual collapse, where deviations from ideal scaling become more pronounced. Even with this adjustment, the resulting asymptotic scaling behavior remains consistent with the theoretical predictions.

These results confirm that the phase transition occurs at $\gamma_c = 1$, which agrees with the continuous-limit analysis. Moreover, the FSS procedure with $\alpha = 2$ not only reproduces the expected algebraic divergence at the transition point but also captures the deterministic-limit behavior, indicating that the framework is robust and can be extended to more complex circuits. Indeed, as shown by two examples in Appendix D, the critical point shifts to $\gamma_c = 2$ in complete binary trees, whereas fully parallel hypercube circuits exhibit no phase transition. These examples demonstrate that FSS analysis successfully detects easy-hard transitions depending on the structural properties of the underlying circuits.

B. Modular adders: Architecture-dependent transitions

To further examine computation-time scaling in more realistic Brownian circuits, we apply the same numerical procedure as in the previous subsection to modular arithmetic circuits. Specifically, we investigate three types of N -digit adders: ordinary full, precede, and product adders. Although all these adders implement the same logical function and

consist of N connected modules, their circuit designs differ, resulting in distinct state-transition diagrams. These differences affect their Brownian stochastic dynamics and, in turn, the scaling behavior of the FPT. In this subsection, numerical simulations of these adders are presented, and the scaling of the mean FPT with respect to both the circuit size and transition rate is analyzed.

Figure 11(a) shows the mean FPT for full adders at various values of γ_+ . A transition in the scaling behavior is clearly observed as γ_+ changes. At the critical value $\gamma_c \simeq 1.43$, the mean FPT scales quadratically with the number of critical path lengths N_g , $\langle \tau \rangle \sim N_g^2$, as confirmed by the FSS collapse in Fig. 11(b), with the exponent $\alpha = 2$. For $\gamma_+ \gg \gamma_c$, the FPT scales linearly with N_g , whereas for $\gamma_+ \ll \gamma_c$, it increases exponentially. These results indicate a clear easy-hard transition governed by the balance between the forward and backward transition pathways in the state-transition diagram.

Notably, the qualitative behavior of the transition is the same as that observed in the one-dimensional SoP circuit discussed in Sec. V A, whereas the critical value γ_c shifts to a significantly larger value. This shift reflects the more complex branching-and-merging structure inherent in the full-adder state-transition graph, which requires a stronger forward bias to ensure efficient computation.

In contrast, the precede adder exhibits a qualitatively different behavior. As shown in Fig. 12(a), the mean FPT does not asymptotically follow either the linear or quadratic lines. For all example values of γ_+ , the FPT eventually grows exponentially with N_g . For small circuit sizes, the scaling may appear approximately linear; however, as N_g increases, a clear crossover to exponential behavior is observed. This crossover behavior further indicates that no intermediate regime exists in which polynomial-size scaling occurs in the precede adder circuits. Finite-size scaling also does not yield a data collapse for any value of γ_c . These results demonstrate that the precede adder remains in the hard-computational regime for all γ_+ . Its

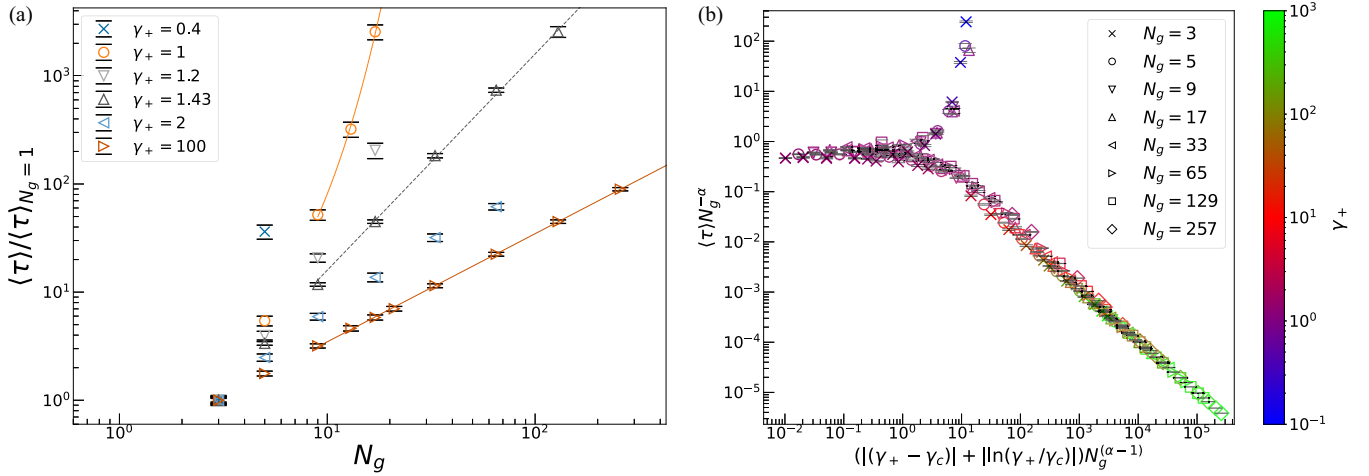


FIG. 11. Numerical results for the full-adder circuits obtained from Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed as the unit of time, $\gamma_- = 1$, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the initial computational state to the final output state. (a) Mean first-passage time as a function of the number of CJoin gates along the critical path, N_g . (b) Finite-size scaling of the mean first-passage time using the scaling exponent $\alpha = 2.0$ and the critical transition rate $\gamma_c = 1.43$.

circuit structure creates bottlenecks or unfavorable transition pathways that hinder efficient propagation in Brownian dynamics.

The product adder behaves similarly to the full adder but with a higher transition threshold. As shown in Fig. 13, the FPT transitions from linear to exponential growth at a critical rate $\gamma_c \simeq 2.34$. Finite-size scaling again confirms the quadratic divergence at the transition point. This higher threshold implies that the product adder requires a stronger forward rate to maintain efficient computation compared with the full adder.

These simulations demonstrate that the design of a modular circuit crucially influences its dynamical behavior under Brownian motion. The existence and value of the critical

rate γ_c vary significantly across circuit types. In particular, a clear easy-hard transition appears in the full and the product adders, but not in the precede adder. Notably, at the transition point, all circuits that exhibit a transition share a universal scaling behavior, $\langle \tau \rangle = \mathcal{O}(N_g^2)$. This quadratic scaling is a signature of an underlying effective one-dimensional structure in state-transition diagrams. In other words, although these adders differ in their local gate connectivity, the dominant transition path that governs computation time behaves as a one-dimensional chain, leading to universal N_g^2 scaling at γ_c . In this sense, the precede adder exhibits exceptional behavior that deviates from this universal trend, the implication of which is discussed in the following section.

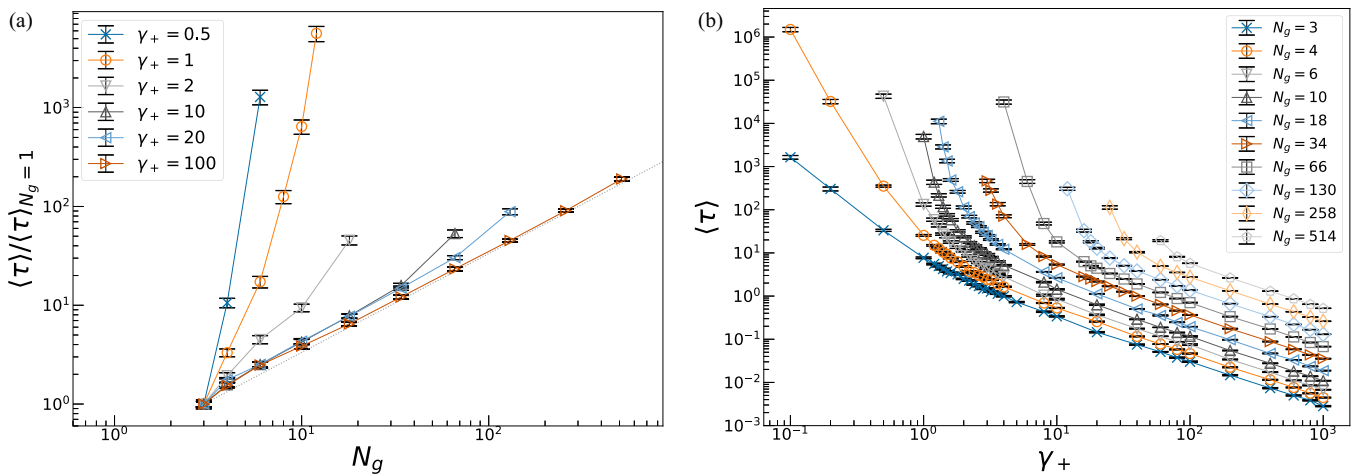


FIG. 12. Numerical results for the precede-adder circuits obtained from Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed as the unit of time, $\gamma_- = 1$, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the initial computational state to the final output state. (a) Mean first-passage time as a function of the number of CJoin gates along the critical path, N_g . (b) Mean first-passage time as a function of the forward transition rate γ_+ for several values of N_g .

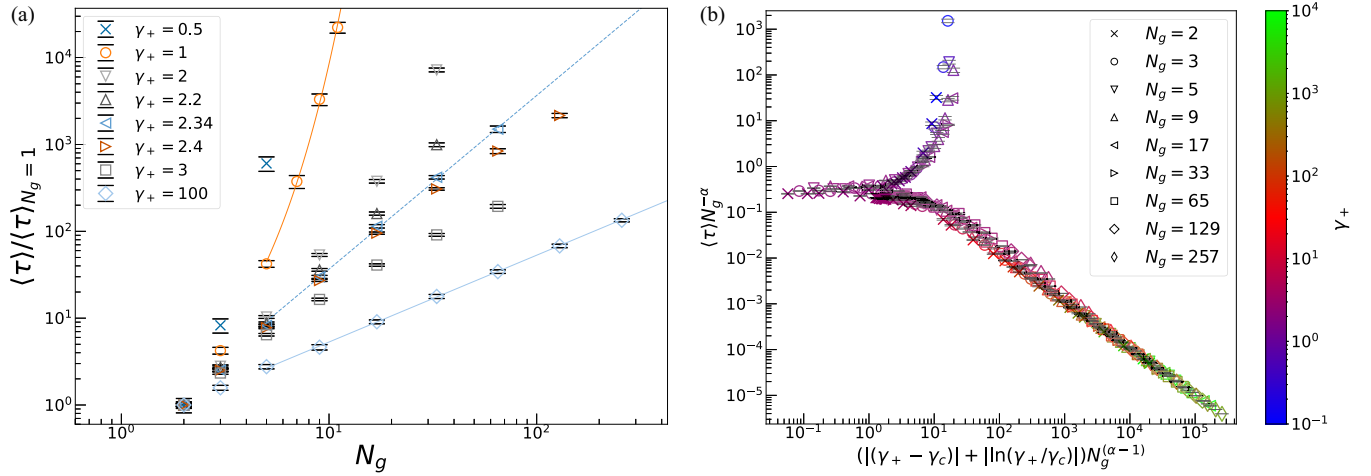


FIG. 13. Numerical results for the product-adder circuits obtained from Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed as the unit of time, $\gamma_- = 1$, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the initial computational state to the final output state. (a) Mean first-passage time as a function of the number of CJoin gates along the critical path, N_g . (b) Finite-size scaling of the mean first-passage time using the scaling exponent $\alpha = 2.0$ and the critical transition rate $\gamma_c = 2.34$.

VI. DISCUSSION AND SUMMARY

A. Dynamical origin of the transition: Balance of forward and backward transitions

In this study, we have identified a dynamical phase transition in Brownian circuits that separates the regimes of linear and exponential scaling of the mean FPT with respect to the circuit size. Finite-size scaling analyses confirm the existence of a critical forward transition rate, denoted by γ_c , at which this change in scaling behavior occurs. Throughout this work, γ_c refers to the critical value of the forward rate γ_+ at which the scaling of the mean FPT changes, whereas the backward rate γ_- is held fixed and sets the unit of time in the numerical evaluation. The associated scaling exponent is found to be universal, taking the value $\alpha = 2$ independent of circuit-specific details. This value agrees with the analytical results of Eq. (12) for a one-dimensional continuous Brownian motion, indicating that Brownian circuits exhibiting such a transition share the same critical scaling behavior as effective one-dimensional drift-diffusion processes.

Generally, by applying an embedding method (Sec. IV C), the dynamics of a Brownian circuit can be reduced to an effective one-dimensional master equation defined in the computational-state space. Within this reduced description, the imbalance between the forward and backward transitions can be quantified explicitly. The characteristic transition rate γ_c is then determined by the condition under which the effective forward and backward probability fluxes are balanced.

In the SoP circuit, the forward and backward transition rates in the intermediate states are exactly symmetric, leading to $\gamma_c = \gamma_-$. In contrast, in modular circuits such as adders, state-transition diagrams always contain merging points where multiple computational paths converge to a single state. In such merging states, the number of backward transitions exceeds that of the forward transitions, which shifts the balance condition and results in $\gamma_c > \gamma_-$. Because no practical circuit exhibits a strict one-to-one correspondence

between the forward and backward transitions, except for the idealized SoP construction, most Brownian circuits are expected to satisfy $\gamma_c > \gamma_-$. We emphasize that this inequality should be understood as a general conjecture rather than a strict theorem. This reflects a structural bias toward backward transitions that arises universally in Brownian circuits that implement nontrivial computations.

B. Time, space, and energy through local detailed balance

The easy-hard transition identified in this work is primarily a dynamical transition. This is determined by observing the scaling behavior of the mean first-passage time with circuit size. The role of the local detailed balance condition is to connect this dynamical transition to an energetic interpretation. In particular, it relates the kinetic bias between forward and backward transitions to the energy cost associated with a computational step. In this subsection, we discuss how the computation time, circuit size, and energy input determine the overall computational cost of a Brownian circuit.

In our numerical analyses, we controlled two key parameters: the circuit size and the forward transition rate γ_+ , while fixing the backward transition rate γ_- as the unit of time. The local detailed balance condition relates the energy cost E associated with each forward transition to

$$E = \beta^{-1} \ln \frac{\gamma_+}{\gamma_-}. \quad (22)$$

Thus, increasing γ_+ at fixed γ_- corresponds to supplying energetic bias to the circuit. Because γ_+ strongly influences the computation time, quantified by the mean FPT, the energy cost is intrinsically related to the dynamical computational performance of the circuit. Meanwhile, circuit complexity, as measured by the circuit size, also constrains the overall computational time. Consequently, the computational cost of a Brownian circuit must be understood in terms of both energy input and circuit size.

For SoP circuits, the state-transition diagram is effectively one dimensional, and the forward/backward balance is exactly symmetric at every intermediate state. Consequently, a dynamical transition occurs at $\gamma_c = \gamma_-$, which corresponds to the zero-energy limit. This initially suggests an energy advantage. However, this benefit is compensated for by the exponential growth of the circuit size, such that a function with N inputs requires $\mathcal{O}(2^N)$ gates and wires. Although the computation time remains polynomial for $\gamma_+ \geq \gamma_-$, linear for $\gamma_+ > \gamma_-$, and quadratic at $\gamma_+ = \gamma_-$, the exponential growth in circuit size dominates the total computational cost. Thus, even though zero-energy computation is formally achieved at $\gamma_+ = \gamma_-$, SoP circuits are not computationally efficient because their space complexity becomes considerably more cumbersome than any advantages gained from the energy-balanced design.

In contrast, modular circuits achieve arithmetic operations with a circuit size that increases only linearly with the input size N . However, their state-transition diagrams contain merging points where multiple computational paths converge, leading to an excess of backward transitions, and therefore, a critical rate satisfying $\gamma_c > \gamma_-$. Our numerical study of three types of N -digit adders, the full, product, and precede adders, quantitatively demonstrates these characteristics.

The precede adder exhibits no phase transition in its FPT scaling and shows an exponential growth of the mean FPT for all values of γ_+ , indicating that its effective transition graph strongly suppresses forward propagation. In contrast, both the full and product adders exhibit clear transition points at which the scaling of the mean FPT changes from linear to exponential in N . The transition points are numerically estimated as $\gamma_c = 1.43$ for the full adder and $\gamma_c = 2.34$ for the product adder, where $\gamma_- = 1$ is the unit of time. Using the local detailed balance condition in Eq. (22), these values correspond to finite energy thresholds, explicitly demonstrating that efficient computation requires a nonzero energy input.

Although the adders examined here represent only specific examples of modular circuits, their state-transition diagrams share a structural motif: the repeated merging of computational pathways. This structure generally increases the total backward transition rate and therefore raises γ_c above unity. Therefore, we expect the qualitative scaling behavior observed here to hold broadly across modular circuit architectures.

These observations reveal a fundamental trade-off. SoP circuits achieve $\gamma_c = \gamma_-$ but require an exponential space, whereas modular circuits achieve a linear space but necessarily satisfy $\gamma_c > \gamma_-$. Consequently, no Brownian circuit can simultaneously achieve a polynomial circuit size and energy-free computation time. More precisely, for a modular circuit of size $\Theta(N)$, an efficient computation in terms of the polynomial mean FPT requires

$$\gamma_+ \geq \gamma_c > \gamma_- \quad (23)$$

At the transition point $\gamma_+ = \gamma_c$, the scaling becomes quadratic for input size, $\langle \tau \rangle = \Theta(N^2)$, whereas for $\gamma_+ < \gamma_c$ the mean FPT increases exponentially. Therefore, energy-free computation ($\gamma_+ = \gamma_-$) is possible only in circuits whose size already scales exponentially. For Brownian circuits with polynomial

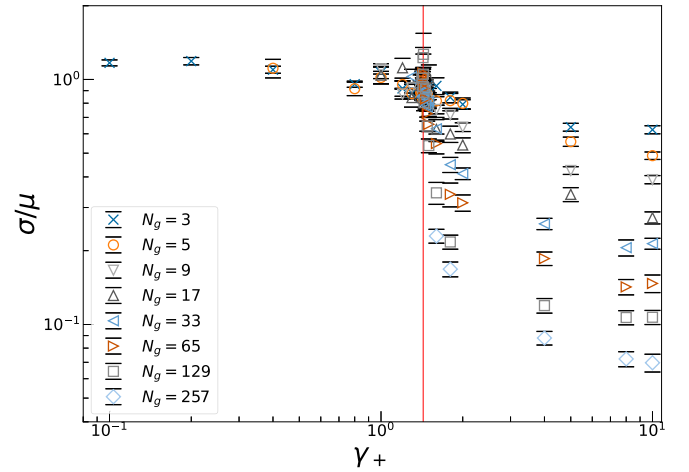


FIG. 14. Coefficient of variation of the first-passage time for the full adder as a function of the forward transition rate γ_+ , obtained from Gillespie simulations of the state-transition dynamics with $\gamma_- = 1$. The coefficient of variation is defined as the standard deviation divided by the mean of the first-passage time. The vertical line indicates the estimated transition point, $\gamma_c \simeq 1.43$, evaluated by the scaling of the mean first-passage time.

complexity, a strictly positive energy input is fundamentally required to maintain efficient computation.

In the computationally efficient regime, the computation time not only scales polynomially with circuit size but also exhibits small relative fluctuations. We quantify the relative fluctuations in the computation time using the coefficient of variation, defined as the ratio of the standard deviation to the mean of the FPT. Figure 14 shows the dependence of the coefficient of variation in the FPT on the forward transition rate for the full adder. In the regime $\gamma_+ > \gamma_c$, the coefficient of variation decreases with increasing circuit size and asymptotically approaches zero in the large-size limit. This behavior indicates that efficient computation in Brownian circuits is characterized not only by short mean computation times but also by high temporal reliability.

This observation is also related to thermodynamic uncertainty relations [36–38], which express a general trade-off between dissipation and precision in stochastic dynamics. Since the computation time considered here is a first-passage time, the most directly related results are uncertainty relations formulated for first-passage processes. We do not attempt to derive a thermodynamic uncertainty relation for the present Brownian circuits. Nevertheless, our results are consistent with the general perspective. In the efficient regime, $\gamma_+ > \gamma_c$, the forward bias corresponds through local detailed balance to finite energetic driving, and the coefficient of variation of the FPT vanishes in the large-size limit. In contrast, in the hard regime, $\gamma_+ < \gamma_c$, where the mean FPT grows exponentially, the relative fluctuations remain of order unity, as in the one-dimensional drift-diffusion processes summarized in Appendix B. Thus, the easy-hard transition is accompanied by a simultaneous change in both the scaling of the mean computation time and its statistical variability.

The relation between Brownian circuits and thermodynamic uncertainty relations has also been studied in terms of

computation-time distributions, where the computation time is defined as a first-passage time, together with thermodynamic costs associated with output detection and resetting [21]. Our analysis is complementary to this approach. Rather than comparing the coefficient of variation with a thermodynamic uncertainty bound for a fixed circuit, we study how the mean FPT and its relative fluctuations scale with circuit size as the forward transition rate is varied. This allows us to identify the critical value γ_c that separates the polynomial- and exponential-time-scaling regimes.

This perspective provides an alternative characterization of the computational costs of Brownian circuits. In contrast to the conventional approach of allowing arbitrarily long computation times, this study employs scaling laws based on the computational size to evaluate the computational efficiency. This approach follows the concepts of computational complexity theory. From this perspective, Brownian circuits exhibit a clear and unavoidable trade-off between the circuit size and the energy input required for efficient computation.

C. Serial versus parallel architectures

Circuits operate according to the order in which information propagates. In Brownian circuits, this ordering is determined by the sequence of the stochastic particle's motion. Modular circuits, whose components are connected in series, generally operate sequentially. However, depending on the circuit structure, paths may exist along which information propagates independently, allowing certain circuit elements to operate in parallel. In deterministic electronic circuits, such parallelism typically reduces the computation time. However, the behavior of parallel elements in Brownian circuits is fundamentally different and is analyzed in Appendix D, where fully parallel architectures are explicitly discussed.

The distinction between serial and parallel architectures significantly influences the scaling of the computational cost. A completely serial design, such as a one-dimensional chain or SoP circuit, exhibits the same scaling behavior as discussed earlier: linear scaling of the mean FPT for $\gamma_+ > \gamma_-$, quadratic scaling at $\gamma_+ = \gamma_-$, and exponential divergence for $\gamma_+ < \gamma_-$. In contrast, a completely parallel design, in which all modules operate independently, generates a state-transition diagram equivalent to a hypercube. For such circuits, the mean FPT increases exponentially with the input size for all values of γ_+ , indicating that the no-parameter regime yields a polynomial-time computation.

This difference indicates the fundamental design principle of Brownian circuits. In deterministic electronic circuits, the addition of parallel components generally reduces the computation time because multiple logic elements operate simultaneously, thereby shortening the critical path. Brownian circuits behave quite differently; because forward progress is governed by probabilistic transitions, simultaneous forward movements across many parallel components are extremely rare events. Consequently, highly parallel architectures lead to longer computation times, with FPT scaling exponentially even when the energy input is large.

These considerations imply that efficient Brownian circuits require a serial organization of computational modules. Parallel elements should be minimized or, when unavoidable,

constrained such that their influence does not accumulate across the multiple stages of computation. Thus, the conventional circuit-design strategies developed for deterministic electronic systems cannot be directly applied to Brownian circuits. Instead, Brownian circuits must be designed using their probabilistic dynamics, ensuring that the effective state-transition structure remains close to one dimensional to maintain polynomial-time computation.

D. Critical points and general implications

Our analysis shows that the existence of a dynamical phase transition in the FPT of a Brownian circuit is determined by the balance between the forward and backward transition rates. Circuits whose state-transition diagrams can be effectively approximated by one-dimensional chains exhibit a clear transition at a characteristic rate γ_c , whereas circuits with strong branching or extensive parallelism may exhibit no such phase transition. Therefore, determining whether a given circuit exhibits a phase transition requires an understanding of how its state-transition diagram can be reduced to an effective one-dimensional stochastic process, and whether the forward and backward probability fluxes can be balanced at any point.

In simple structural diagrams, such as those discussed in Appendix E, this balance can be calculated explicitly because the equilibrium distribution is analytically tractable. However, for general Brownian circuits, state-transition diagrams increase rapidly in terms of both size and structural complexity. This makes computing or even sampling the equilibrium distribution required to determine γ_c . Consequently, characterizing the existence and location of a critical point remains nontrivial for large or irregular circuit architectures.

Nevertheless, the conceptual framework developed in this study applies not only to Brownian circuits composed of CJoin gates but also to a more general class of Brownian computation systems driven by Brownian motion, whose logical gates are encoded by the positions of Brownian particles. In such systems, the computational process can generally be interpreted as the stochastic motion of a particle in a discrete state-transition diagram. Whenever the effective topology of this diagram is sufficiently close to one dimension, a similar easy-hard phase transition in computational time is expected.

Finally, the phase-transition picture developed in this work is not limited to artificial Brownian circuits. Related phenomena are likely to occur in real-world systems that rely on Brownian motion for information processing or sequential control. A notable example is DNA transcription, which proceeds as a Brownian process that is effectively one dimensional and inherently serial, taking place along a one-dimensional template. This architecture can be interpreted as a design choice that avoids considerable slowdowns associated with excessive branching and parallelism. Future research into these connections could yield significant insights, particularly by identifying biological and soft-matter systems that either exploit or avoid easy-hard transitions. In this broader context, our results suggest that maintaining fast and reliable Brownian computation generally requires a finite energetic bias, making the energy input an inherent component of fluctuation-driven information processing.

ACKNOWLEDGMENTS

We would like to thank T. Sakaue for useful discussions on the first-passage time problem and Y. Utsumi for valuable guidance on Brownian circuits. We are grateful to J. Takahashi and K. Itao for continuous encouragement throughout this work. This work was supported by JSPS KAKENHI Grant No. 23H01095 and JST Grant No. JPMJPF2221.

DATA AVAILABILITY

The data that support the findings of this article are openly available [35].

APPENDIX A: FOKKER-PLANCK EQUATION AND MASTER EQUATION

In this Appendix, we summarize the relationship between the master equation and the Fokker-Planck equation to clarify the notation and assumptions [39]. The master equation describes the stochastic dynamics in a discrete state space, whereas the Fokker-Planck equation provides its continuous limit in the diffusive regime. Here, we derive the Fokker-Planck equation from the master equation in a one-dimensional system with N discrete states.

The master equation for discrete space is given by Eq. (4). To take the continuous limit, we introduce a scaling transformation for position x and displacement Δx in terms of the state index i :

$$x = \frac{i}{N}, \quad \Delta x = \frac{1}{N}. \quad (\text{A1})$$

We define the probability density $p(x, t)$ as

$$p(x, t)\Delta x = p_i(t). \quad (\text{A2})$$

The Taylor expansion of $p(x, t)$ yields

$$p(x \pm \Delta x, t) = p(x, t) \pm \frac{1}{N} \frac{\partial}{\partial x} p(x, t) + \frac{1}{2N^2} \frac{\partial^2}{\partial x^2} p(x, t) \cdots \quad (\text{A3})$$

Substituting this into the master equation and taking $N \rightarrow \infty$, we obtain the one-dimensional Fokker-Planck equation:

$$\frac{\partial}{\partial t} p(x, t) = -v \frac{\partial}{\partial x} p(x, t) + D \frac{\partial^2}{\partial x^2} p(x, t), \quad (\text{A4})$$

where the drift velocity v and diffusion constant D are given by

$$v = \frac{\gamma_+ - \gamma_-}{N}, \quad D = \frac{\gamma_+ + \gamma_-}{2N^2}. \quad (\text{A5})$$

Thus, continuous relaxation is valid when

$$\gamma_+ - \gamma_- = \mathcal{O}(N), \quad \gamma_+ + \gamma_- = \mathcal{O}(N^2), \quad (\text{A6})$$

ensuring that v and D remain finite as $N \rightarrow \infty$. This corresponds to the regime in which $\gamma_+ \simeq \gamma_-$, that is, the small bias limit in which the drift-diffusion approximation is appropriate.

Note that, in the numerical experiments presented in Sec. V, we set $\gamma_- = 1$ for convenience because the overall timescale does not affect scaling analysis. The continuous

limit results derived in this study apply after rescaling using this choice.

We define the probability flux as

$$j(x, t) := vp(x, t) - D \frac{\partial}{\partial x} p(x, t). \quad (\text{A7})$$

The corresponding boundary conditions can also be expressed in continuum form. For the absorbing boundary at the final state, the discrete master equation contains the term

$$\frac{d}{dt} p(N, t) = \gamma_+ p(N-1, t) - \gamma_- p(N, t) + \gamma_+ p(N+1, t). \quad (\text{A8})$$

Imposing absorption requires $p(N+1, t) = 0$, which yields, in the continuum limit,

$$p(1, t) = 0. \quad (\text{A9})$$

The reflection boundary at the initial state can be treated in an analogous manner. In the discrete description, the master equation at the boundary takes the form

$$\frac{d}{dt} p(1, t) = \gamma_- p(2, t) - \gamma_+ p(1, t). \quad (\text{A10})$$

Using the definition of the probability flux in the continuum limit, this condition corresponds to

$$j(0, t) = 0. \quad (\text{A11})$$

These results confirm that the Fokker-Planck equation with an absorbing boundary at $x = 1$ and a reflection boundary at $x = 0$ provides the correct continuum analog of the original discrete master equation.

APPENDIX B: FIRST-PASSAGE TIMES OF ONE-DIMENSIONAL CONTINUOUS SPACE

In this Appendix, we present the classical results of the FPTs of a one-dimensional drift-diffusion process following standard Refs. [32,33]. In the main text, we have summarized the known analytical expressions for FPT moments under drift and diffusion, which form the basis for the discussion of phase transitions in Brownian circuits.

We consider the Fokker-Planck equation given in Eq. (A4), which is defined in the interval $x \in [-N, 0]$ with an absorbing boundary at the origin and a reflecting boundary at $x = -N$:

$$p(0, t) = 0, \quad (\text{B1})$$

$$j(-N, t) := \left(vp - D \frac{\partial p}{\partial x} \right) \Big|_{x=-N} = 0. \quad (\text{B2})$$

A particle starts at $x_0 \in [-N, 0]$ at time $t = 0$ under the initial condition

$$p(x, 0) = \delta(x - x_0) \quad (\text{B3})$$

and eventually reaches the absorbing boundary at $x = 0$. We focus on the FPT moments for a specific initial position $x_0 = -N$. The survival probability is defined by

$$P_s := \int_{-N}^0 p(x, t) dx, \quad (\text{B4})$$

and the FPT distribution $P_t(t)$ is given by the standard relationship

$$P_t(t) = -\frac{d}{dt}P_s(t). \quad (\text{B5})$$

To compute FPT moments, we use the Laplace transform

$$\mathcal{L}[f(t)] = \tilde{f}(z) := \int_0^\infty f(t)e^{-zt} dt, \quad (\text{B6})$$

where z is a complex parameter. Applying Laplace transformation to P_s yields

$$\tilde{P}_s(z) = \int_0^\infty P_s(\tau)e^{-z\tau} d\tau = \int_{-N}^0 \tilde{p}(x, z) dx. \quad (\text{B7})$$

Using the Laplace transformation of $p(x, t)$, the FPT moments can be expressed as

$$\langle \tau^l \rangle = \left(-\frac{d}{dz} \right)^{l-1} \int_{-N}^0 \tilde{p}(x, z) dx \Big|_{z=0}. \quad (\text{B8})$$

Furthermore, using the Laplace transform of the time derivative of $p(x, t)$,

$$\mathcal{L}\left[\frac{\partial p}{\partial t}\right] = z\tilde{p}(x, z) - p(x, 0), \quad (\text{B9})$$

we substitute into Eq. (A4) to obtain the differential equation for $\tilde{p}(x, z)$:

$$\frac{\partial^2}{\partial x^2} \tilde{p} - \frac{v}{D} \frac{\partial}{\partial x} \tilde{p} - \frac{z}{D} \tilde{p} = -\frac{1}{D} \delta(x + N). \quad (\text{B10})$$

The boundary conditions in Eqs. (B1) and (B2) become

$$\tilde{p}(0, z) = 0, \quad (\text{B11})$$

$$\tilde{j}(-N, z) := \left(v\tilde{p} - D \frac{\partial \tilde{p}}{\partial x} \right) \Big|_{x=-N} = 0. \quad (\text{B12})$$

Solving this ordinary differential equation on $[-N, 0]$ yields

$$\tilde{p}(x, z) = \frac{e^{-k_+ x} - e^{-k_- x}}{(k_+^* e^{k_+ N} - k_-^* e^{k_- N}) D}, \quad (\text{B13})$$

where

$$k_\pm = \frac{-v \pm \sqrt{v^2 + 4Dz}}{2D}, \quad k_\pm^* = \frac{v \pm \sqrt{v^2 + 4Dz}}{2D}.$$

The mean FPT is obtained from $\tilde{p}(x, 0)$:

$$\mu = \langle \tau \rangle = \frac{D}{v^2} \left(e^{-\frac{vN}{D}} - 1 \right) + \frac{N}{v}. \quad (\text{B14})$$

From this closed-form expression, the asymptotic behavior of the mean FPT follows directly:

$$\langle \tau \rangle = \begin{cases} \frac{N}{v}, & \frac{vN}{D} \gg 1, \\ \frac{N^2}{2D}, & \frac{vN}{D} \simeq 0, \\ \frac{D}{v^2} e^{\left| \frac{vN}{D} \right|}, & \frac{vN}{D} \ll -1. \end{cases} \quad (\text{B15})$$

Thus, as is well known in classical FPT theory, the mean FPT $\langle \tau \rangle$ transitions from linear growth in N when $v > 0$ to exponential growth when $v < 0$, with quadratic growth at the critical point $v = 0$.

Similarly, the second moment is derived as follows:

$$\langle \tau^2 \rangle = \frac{N^2}{v^2} + \frac{6DN}{v^3} e^{-\frac{vN}{D}} + \frac{2D^2}{v^4} \left(-2 + e^{-\frac{vN}{D}} + e^{-\frac{2vN}{D}} \right), \quad (\text{B16})$$

and the variance $\sigma^2 = \langle \tau^2 \rangle - \langle \tau \rangle^2$ becomes

$$\sigma^2 = \frac{DN}{v^3} \left(4e^{-\frac{vN}{D}} + 2 \right) + \frac{D^2}{v^4} \left(-5 + 4e^{-\frac{vN}{D}} + e^{-\frac{2vN}{D}} \right). \quad (\text{B17})$$

All higher-order moments exhibit the same qualitative change at $v = 0$, separating the linear and exponential growth regimes.

Finally, the coefficient of variation is given by

$$\sigma/\mu = \begin{cases} \sqrt{\frac{2D}{v^4 N}}, & \frac{vN}{D} \gg 1, \\ \sqrt{\frac{2}{3}}, & \frac{vN}{D} \approx 0, \\ 1, & \frac{vN}{D} \ll -1. \end{cases} \quad (\text{B18})$$

In the linear regime ($v > 0$), σ/μ vanishes as $N \rightarrow \infty$, which implies that the relative fluctuations disappear when the mean FPT increases linearly with N .

APPENDIX C: SoP CIRCUIT SIZE

In this Appendix, we count the circuit size required to implement a function in the SoP form using CJoin gates. For the two input variables, the SoP circuit contains four gates (see Fig. 9). Each additional input variable introduces an extra processing stage and the number of gates added at the N th stage is 2^N . Therefore, the total number of gates required to implement an SoP function f_N with N inputs is

$$\text{Size}(f_N) = \sum_{i=2}^N 2^i = 4 \sum_{i=0}^{N-2} 2^i = 4(2^{N-1} - 1). \quad (\text{C1})$$

To construct a circuit with N inputs and N outputs, we extend the single-output SoP circuit by duplicating the product terms at the final stage. In this stage, each product term appears in two particles: One particle is used as the output and the other remains available for further interaction. The remaining particles interact with the particles preserved in the early stages, allowing the product terms to be copied as required. Finally, the duplicated particles are linked through junctions to produce N -independent outputs. The resulting circuit with N inputs and N outputs has the following size:

$$\begin{aligned} \text{Size}(f_N) &= 4(2^{N-1} - 1) + (N - 1)2^N \\ &= N2^N + 2^N - 4 = \mathcal{O}(N \cdot 2^N). \end{aligned} \quad (\text{C2})$$

Thus, although the SoP representation provides a universal construction method for Brownian circuits, this universality comes at the cost of an exponentially increasing circuit size with the number of input variables.

APPENDIX D: COMPLETE BINARY TREE AND FULLY PARALLEL CIRCUITS

In this Appendix, we present two examples to clarify the influence of the structure of a state-transition diagram on the scaling behavior of the FPT in Brownian circuits. The first example is a complete binary tree, which represents the

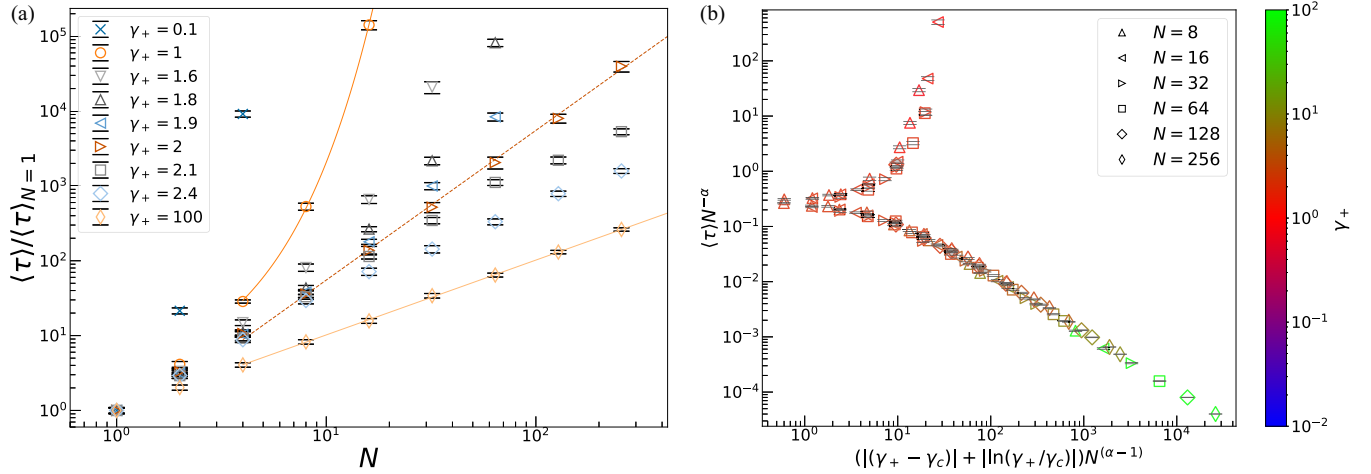


FIG. 15. Numerical results for the complete binary tree obtained from Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed as the unit of time, $\gamma_- = 1$, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the root state to the leaf state. (a) Mean first-passage time as a function of the tree depth N , which is the distance from the root to the leaves. The three guide curves indicate exponential, quadratic, and linear scaling with N , corresponding, respectively, to the hard regime, the critical point, and the easy regime. (b) Finite-size scaling plot of the mean first-passage time using the scaling exponent $\alpha = 2.0$ and the critical rate $\gamma_c = 2.0$, consistent with the effective balance condition for a binary tree.

simplest case in which multiple backward transitions occur at every intermediate state. As discussed in Sec. IV D, an m -ary branching structure is expected to exhibit an easy-hard transition under the generalized balance condition $\gamma_+ = m\gamma_-$; therefore, the complete binary tree ($m = 2$) provides a minimal nontrivial test case for the predicted critical rate $\gamma_c = 2\gamma_-$.

Under the embedding method, a Brownian circuit whose state-transition diagram forms a complete binary tree is equivalent to a one-dimensional diagram in which the backward transition rate is twice the forward rate. Therefore, the predicted critical point for scaling switching is $\gamma_c = 2\gamma_-$. Figure 15 shows the numerical results for the mean FPT and

its finite-size scaling. The results clearly exhibit a dynamical phase transition at $\gamma_c = 2\gamma_-$, which is consistent with the theoretical prediction for a binary-branching structure. This provides direct numerical evidence supporting the conjecture proposed in Sec. IV D.

To provide a complementary example, we examine a fully parallel circuit in which all modules operate independently, and each module consists of a single gate. The circuit accepts an input only when all the modules have reached their final state. The corresponding state-transition graph is an N -dimensional hypercube, as discussed in Sec. IV C. In this architecture, computational progress requires simultaneous forward transitions across multiple independent components.

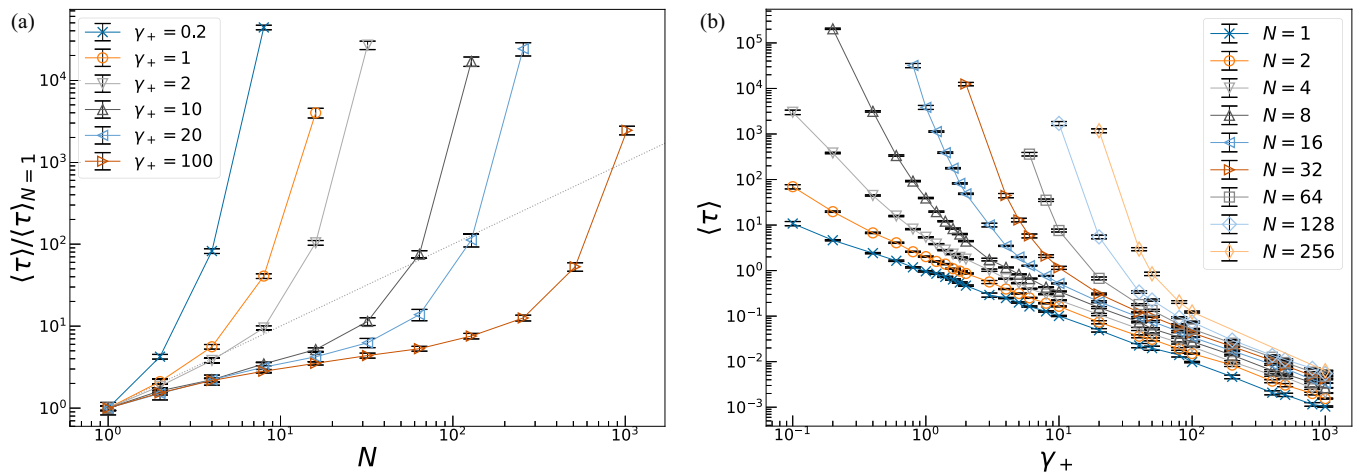


FIG. 16. Numerical results for the fully parallel circuit whose state-transition diagram forms an N -dimensional hypercube, obtained from Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed at $\gamma_- = 1$, which sets the unit of time, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the initial computational state to the final output state. (a) Mean first-passage time as a function of the number of parallel modules N . (b) Mean first-passage time as a function of the forward transition rate γ_+ for several values of N .

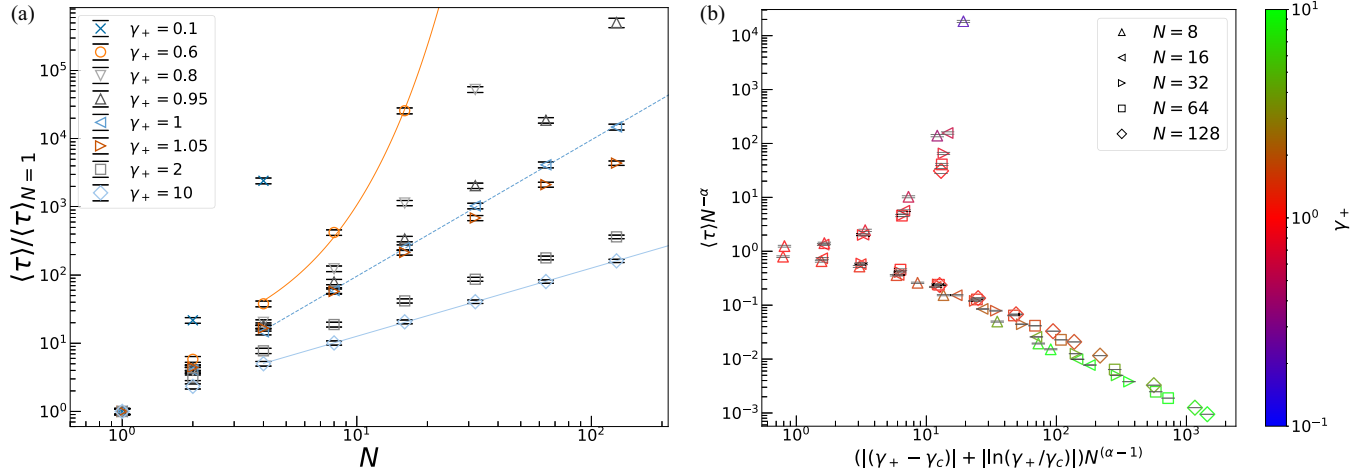


FIG. 17. Numerical results for the one-dimensional chain with one-step backward subbranches, obtained from Gillespie simulations of the state-transition dynamics. The backward transition rate is fixed at $\gamma_- = 1$, which sets the unit of time, and the forward transition rate γ_+ is varied. The plotted quantity is the mean first-passage time $\langle \tau \rangle$ from the initial state to the final state. (a) Mean first-passage time as a function of the length of the main chain, N . The reference curves indicate exponential, quadratic, and linear scaling with N , corresponding, respectively, to the hard regime, the critical point, and the easy regime. In particular, at the predicted transition point $\gamma_c = 1.0$, the mean first-passage time exhibits the expected quadratic scaling. (b) Finite-size scaling plot of the mean first-passage time using the scaling exponent $\alpha = 2.0$ and the critical transition rate $\gamma_c = 1.0$, consistent with the effective balance condition.

Consequently, there exists no finite forward transition rate for which the FPT scales polynomially with N . In other words, the system exhibits no easy-hard transition formally corresponding to $\gamma_c \rightarrow \infty$.

Figure 16 shows numerical results that confirm this behavior. For small system sizes, the mean FPT initially increases logarithmically, reflecting a crossover regime in which simultaneous forward updates are not strongly suppressed. However, as the number of modules increases, the scaling becomes dominated by the exponentially many backward pathways inherent in the hypercube structure. Consequently, the FPT increases exponentially with N_g for all values of γ_+/γ_- . No polynomial-time regime was observed even near the crossover.

APPENDIX E: MAIN CHAIN WITH ONE-STEP SUBBRANCHES

To illustrate the effects of local branching on the effective one-dimensional reduction, we consider a model in which each node in a one-dimensional chain has a single backward branch of length one. Let $p_i^{(m)}$ denote the probability of being on the main chain at level i and $p_i^{(s)}$ denote the probability of being on its one-step subbranch. The master equations at level i are

$$\frac{d}{dt} p_i^{(m)} = \gamma_+ (p_{i-1}^{(m)} + p_{i-1}^{(s)}) + \gamma_- p_{i+1}^{(m)} - \gamma_+ p_i^{(m)} - 2\gamma_- p_i^{(m)}, \quad (\text{E1})$$

$$\frac{d}{dt} p_i^{(s)} = \gamma_- p_{i+1}^{(m)} - \gamma_+ p_i^{(s)}. \quad (\text{E2})$$

By defining the total probability at level i as $p_i = p_i^{(m)} + p_i^{(s)}$, we follow the same coarse-graining concept of the embedding method discussed in Sec. IV C to obtain the effective equation

$$\frac{d}{dt} p_i = \gamma_+ p_{i-1} + 2\gamma_- p_{i+1}^{(m)} - \gamma_+ p_i - 2\gamma_- p_i^{(m)}. \quad (\text{E3})$$

Introducing the fractions

$$a_i = p_i^{(m)} / p_i, \quad b_i = p_i^{(s)} / p_i,$$

with $a_i + b_i = 1$, the above equation becomes

$$\frac{d}{dt} p_i = \gamma_+ p_{i-1} + 2\gamma_- a_{i+1} p_{i+1} - (\gamma_+ + 2a_i \gamma_-) p_i. \quad (\text{E4})$$

Thus, the effective backward rate at level i is $2\gamma_- a_i$, and the balance condition defining the transition point is $\gamma_c = 2a_i \gamma_-$. To estimate the value of γ_c , we assume that, for a sufficiently long chain in a quasistationary regime, the probability weight between the main and subbranches is equally distributed, yielding $a_i = \frac{1}{2}$. Under this assumption,

$$\gamma_c = \gamma_-,$$

coinciding with the transition conditions for a purely one-dimensional chain.

This result suggests that the presence of a single one-step backward branch at each level does not shift the critical point unless it causes additional backward transitions. To test this prediction, we performed numerical simulations for a one-step subbranch model, as shown in Fig. 17. The simulations confirmed that the mean FPT exhibits the same scaling transition at $\gamma_c = 1$ (with $\gamma_- = 1$) as in the purely one-dimensional case, despite additional local branching.

[1] R. P. Feynman, Simulating physics with computers, *Int. J. Theor. Phys.* **21**, 467 (1982).

[2] S. Lloyd, Ultimate physical limits to computation, *Nature (London)* **406**, 1047 (2000).

- [3] R. Landauer, Irreversibility and heat generation in the computing process, *IBM J. Res. Dev.* **5**, 183 (1961).
- [4] C. H. Bennett, The thermodynamics of computation—A review, *Int. J. Theor. Phys.* **21**, 905 (1982).
- [5] D. H. Wolpert, The stochastic thermodynamics of computation, *J. Phys. A: Math. Theor.* **52**, 193001 (2019).
- [6] D. H. Wolpert, J. Korbelt, C. W. Lynn, F. Tasnim, J. A. Grochow, G. Kardeş, J. B. Aimone, V. Balasubramanian, E. De Giuli, D. Doty, N. Freitas, M. Marsili, T. E. Ouldrige, A. W. Richa, P. Riechers, E. Roldán, B. Rubenstein, Z. Toroczkai, and J. Paradiso, Is stochastic thermodynamics the key to understanding the energy costs of computation? *Proc. Natl. Acad. Sci. USA* **121**, e2321112121 (2024).
- [7] R. Landauer, Information is physical, *Phys. Today* **44**, 23 (1991).
- [8] F. Jülicher, A. Ajdari, and J. Prost, Modeling molecular motors, *Rev. Mod. Phys.* **69**, 1269 (1997).
- [9] R. D. Astumian, Thermodynamics and kinetics of a Brownian motor, *Science* **276**, 917 (1997).
- [10] A. B. Boyd, D. Mandal, and J. P. Crutchfield, Identifying functional thermodynamics in autonomous Maxwellian Ratchets, *New J. Phys.* **18**, 023049 (2016).
- [11] P. Reimann, Brownian motors: Noisy transport far from equilibrium, *Phys. Rep.* **361**, 57 (2002).
- [12] J. M. R. Parrondo, J. M. Horowitz, and T. Sagawa, Thermodynamics of information, *Nat. Phys.* **11**, 131 (2015).
- [13] T. Sagawa and M. Ueda, Generalized Jarzynski equality under nonequilibrium feedback control, *Phys. Rev. Lett.* **104**, 090602 (2010).
- [14] T. Sagawa and M. Ueda, Nonequilibrium thermodynamics of feedback control, *Phys. Rev. E* **85**, 021104 (2012).
- [15] T. Sagawa, Second law, entropy production, and reversibility in thermodynamics of information, [arXiv:1712.06858](https://arxiv.org/abs/1712.06858).
- [16] A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz, Experimental verification of Landauer's principle linking information and thermodynamics, *Nature (London)* **483**, 187 (2012).
- [17] N. Shiraishi, *An Introduction to Stochastic Thermodynamics: From Basic to Advanced*, Fundamental Theories of Physics (Springer, Singapore, 2023), Vol. 212.
- [18] P. Strasberg, J. Cerrillo, G. Schaller, and T. Brandes, Thermodynamics of stochastic Turing machines, *Phys. Rev. E* **92**, 042104 (2015).
- [19] D. Yoshino and Y. Tokura, Thermodynamics of computation for CMOS NAND gate, *J. Phys. Soc. Jpn.* **92**, 124004 (2023).
- [20] A. Kolchinsky and D. H. Wolpert, Thermodynamic costs of Turing machines, *Phys. Rev. Res.* **2**, 033312 (2020).
- [21] Y. Utsumi, Y. Ito, D. Golubev, and F. Peper, Computation time and thermodynamic uncertainty relation of Brownian circuits, [arXiv:2205.10735](https://arxiv.org/abs/2205.10735).
- [22] Y. Utsumi, D. Golubev, and F. Peper, Thermodynamic cost of Brownian computers in the stochastic thermodynamics of resetting, *Eur. Phys. J.: Spec. Top.* **232**, 3259 (2023).
- [23] Y. Utsumi, Join gate with memory in token-conserving Brownian circuits and the thermodynamic cost, *Phys. Rev. E* **113**, 024129 (2026).
- [24] C. Y. Gao and D. T. Limmer, Principles of low dissipation computing from a stochastic circuit model, *Phys. Rev. Res.* **3**, 033169 (2021).
- [25] G. Manzano, G. Kardeş, É. Roldán, and D. H. Wolpert, Thermodynamics of computations with absolute irreversibility, unidirectional transitions, and stochastic computation times, *Phys. Rev. X* **14**, 021026 (2024).
- [26] S. Kirkpatrick and B. Selman, Critical behavior in the satisfiability of random Boolean expressions, *Science* **264**, 1297 (1994).
- [27] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, Determining computational complexity from characteristic phase transitions, *Nature (London)* **400**, 133 (1999).
- [28] F. Peper, J. Lee, J. Carmona, J. Cortadella, and K. Morita, Brownian circuits: Fundamentals, *J. Emerg. Technol. Comput. Syst.* **9**, 1 (2013).
- [29] J. Lee, F. Peper, S. Cotofana, M. Naruse, M. Ohtsu, T. Kawazoe, Y. Takahashi, T. Shimokawa, L. B. Kish, and T. Kubota, Brownian circuits: Designs, *Int. J. Unconv. Comput.* **12**, 341 (2016).
- [30] İ. Ercan and E. Suyabatmaz, Fundamental energy limits of SET-based Brownian NAND and half-adder circuits, *Eur. Phys. J. B* **91**, 113 (2018).
- [31] C. E. Shannon, The synthesis of two-terminal switching circuits, *Bell Syst. Tech. J.* **28**, 59 (1949).
- [32] C. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry, and the Natural Sciences* (Springer, Berlin, 1985).
- [33] S. Redner, *A Guide to First-Passage Processes* (Cambridge University Press, Cambridge, 2001).
- [34] D. T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, *J. Comput. Phys.* **22**, 403 (1976).
- [35] K. Okajima, brownian-circuit-gillespie-mc, version 1.0.1, GitHub, 2026, <https://github.com/wokajima/brownian-circuit-gillespie-mc/releases/tag/v1.0.1>.
- [36] A. C. Barato and U. Seifert, Thermodynamic uncertainty relation for biomolecular processes, *Phys. Rev. Lett.* **114**, 158101 (2015).
- [37] J. M. Horowitz and T. R. Gingrich, Thermodynamic uncertainty relations constrain non-equilibrium fluctuations, *Nat. Phys.* **16**, 15 (2020).
- [38] A. Pal, S. Reuveni, and S. Rahav, Thermodynamic uncertainty relation for first-passage times on Markov chains, *Phys. Rev. Res.* **3**, L032034 (2021).
- [39] N. van Kampen, *Stochastic Processes in Physics and Chemistry* (Elsevier Science, Amsterdam, 1992).