Energy Aware Computing through Probabilistic Switching: A Study of Limits

Krishna V. Palem, *Fellow*, *IEEE*

Abstract—The main result in this paper establishes the energy savings derived by using *probabilistic* AND as well as NOT gates constructed from an idealized *switch* that produces a *probabilistic* bit (PBIT). A probabilistic switch produces the desired value as an output that is 0 or 1 with probability p, represented as a PBIT, and, hence, can produce the wrong output value with a probability of $(1-p)$. In contrast with a probabilistic switch, a conventional deterministic switch produces a BIT whose value is always correct. Our switch-based gate constructions are a particular case of a systematic methodology developed here for building *energy-aware networks* for computing, using PBITs. Interesting examples of such networks include AND, OR, and NOT gates (or, as functions, Boolean conjunction, disjunction, and negation, respectively). To quantify the energy savings, novel measures of "technology independent" *energy complexity* are also introduced here—these measures parallel conventional machine-independent notions of computational complexity such as the algorithm's running time and space. Networks of switches can be related to Turing machines and to Boolean *circuits*, both of which are widely known and well-understood models of computation. Our gate and network constructions lend substance to the following thesis (established for the first time by this author [1], [2], [3]): The mathematical technique referred to as *randomization* yielding *probabilistic algorithms* results in energy savings through a physical interpretation based on statistical thermodynamics and, hence, can serve as a basis for energy-aware computing. While the estimates of the energy saved through PBIT-based probabilistic computing switches and networks developed here rely on the constructs and thermodynamic models due to Boltzmann, Gibbs, and Planck, this work has also led to the innovation of probabilistic CMOS-based devices and computing frameworks. Thus, for completeness, the relationship between the physical models on which this work is based and the electrical domain of CMOS-based switching will be discussed.

Index Terms—Energy-aware systems, low-power design, probabilistic computation.

1 introduction

Concerns of *power* (or *energy*) consumption have become increasingly significant in the context of the design as well as the use of embedded and high-performance computing systems. To paraphrase Trevor Mudge, "Power (and energy) are first-class citizens in current considerations of computer system design." While devices, computer architecture, and the layers of software that reside in and execute at higher levels of abstraction (such as operating systems, runtime, compilers, and programming languages) all afford opportunities for being *energy-aware*, the most fundamental limits are truly rooted in the physics of energy consumption—specifically in *thermodynamics*. Based on this premise, this paper embodies the innovation of models of computing for energy-aware algorithm design and analysis, for the first time, based on the following thesis central to this work: *The computational technique referred to as randomization yielding probabilistic algorithms, now ubiquitous to the mathematical theory of probabilistic algorithm design and analysis, when interpreted as a physical phenomenon through classical statistical thermodynamics, yields to energy savings that decrease with the probability* p *with which each primitive computational step is guaranteed to be correct (or, equivalently, increase with the probability of error,* $(1-p)$*).*

Historically, probabilistic algorithms were viewed as a mathematically very promising approach to algorithmic design, as elegantly stated by Schwartz [4]: "The startling success of the Rabin-Strassen-Solovay (see Rabin [5]) algorithm, together with the intriguing foundational possibility that axioms of randomness may constitute a useful fundamental source of mathematical truth independent of, but supplementary to, the standard axiomatic structure of mathematics (see Chaitin and Schwartz [6]), suggests that probabilistic algorithms ought to be sought vigorously." Since this prediction, probabilistic algorithms have proliferated in a range of areas centered around the theoretical foundations of computer science.

At its heart, the work described in this paper is based on the definition of an abstract *energy-aware switch*, or *switch* for short (in Section 4.1), which is the *first* contribution. A switch, denoted sw, is a device for realizing computations that are functions of a single bit. While switches are provably building blocks for constructing Boolean gates as well as for describing algorithms, in our context, they also serve as idealizations for modeling energy consumption. Our basic idealizations of a "switch" and "switching" model transformations to the *state* of a physical device capable of altering its Boltzmann (physical) entropy, with a well-defined accompanying expenditure of energy consistent with the laws of physical domain—electrical, gaseous, or others in which the switch is operating.

- *The author is with the Center for Research on Embedded Systems and Technology, School of Electrical and Computer Engineering, Georgia Institute of Technology, 777 Atlantic Drive NW, Atlanta, GA 30332-0250. E-mail: palem@ece.gatech.edu.*

Manuscript received 2 Aug. 2003; revised 13 Aug. 2004; accepted 6 Oct. 2004; published online 15 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0115-0803.

0018-9340/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

We note that, to realize computations, switching will be used to alter the current bit or value, say 0, to some other value, say 1. All computation can be realized as compositions of such elemental one-bit switching changes. In this work, the characterization of probabilistic switching as well as its associated energy consumption will be based on physical realizations characterized by (classical) statistical thermodynamics, (see Balian [7], for example). This statistical foundation is essential to proving two fundamental theorems (stated in Section 9) which serve as a basis here for determining the energy consumption of computations constructed from networks of switches. These physical principles characterizing the energy changes associated with switching were established by this author in [1], [3]: *The energy consumed by deterministic switching is never less than* $(\kappa t \ln 2)$ *joules, referred to often as the "fundamental (or thermodynamic) limit." In contrast, the energy consumed by an idealized probabilistic switch with an associated probability of error of (1-p) is lower, and can be as low as* $(\kappa t \ln 2p)$ *joules for each switching step*. Here, k is the well-known Boltzmann's constant, t is the *temperature* of the thermodynamic system, and $\ln$ is the natural logarithm. This result established that, at the fundamental limit, probabilistic algorithms offer the potential for energy savings of $kt \ln(\frac{1}{p})$ joules per primitive switching step. In keeping with traditional idealizations, our switches are not lossy since switching is always performed at thermal equilibrium. By basing our developments here on this notion of a probabilistic switch modeled using statistical thermodynamics, rather than on the more familiar deterministic models of energy consumption known previously (see Section 2), our switches are inherently probabilistic; they do not need an explicit random source, typically realized as a pseudorandom number generator central to the earlier development of the theory of probabilistic algorithms (see Vazirani and Vazirani [8] for example).

Given a switch sw, our *second* contribution (in Section 5) involves a systematic method for constructing *networks* of switches to realize AND gates. Each switch sw has an *input* value which is either 0 or 1 and an explicit *enabling signal* that determines whether a switch is "active" or "inactive." In turn, sw is capable of producing an output value from the set $\{0, 1\}$ and, possibly, an output enabling signal to a successor switch. *Thus, the output value of switch sw is correct only with a probability p (and is erroneous with probability (1-p)); this output value will be referred to as a* probabilistic bit *or a* PBIT. A distinguishing feature of this switch is that depending on the input to the associated network, it is active only in the context of being "required" to compute a value and is enabled by a predecessor switch; it is inactive otherwise. For this reason, we will refer to such a switch as an *introverted* switch.

A network $\mathcal{N}$ is central to defining a technology independent *energy complexity* of a switch and constitutes our *third* contribution; these complexity measures are introduced in Section 5.3 in the deterministic context and in Section 8.2 for probabilistic networks. Informally, the energy complexity of a network is the number of steps involved in switching from the start of the computation till it is completed and, thus, is a measure of the *dynamic energy* consumed by the computation in the context of a given input. In the case of a probabilistic network, given a particular input, it will be the average energy consumed over the "family" of executions induced by the probabilistic nature of the network. Through a straightforward construction and for completeness, we also show in Section 6 that logical negation or a NOT gate can be realized through a single switch and, therefore, its energy behavior in the deterministic and probabilistic cases is identical to those claimed above for a switch in isolation.

In Section 7, and moving to the *fourth* contribution, we prove that a standard two-input AND function requires at least two "energy consuming" switches in the deterministic case and, hence, its *energy complexity* is bounded below by 2. Identical results can be obtained for OR gate constructions.

While the above results are specific to deterministic networks and computations, by construction, a switch sw can be *naturally* or *explicitly* probabilistic. Thus, the energy characteristics of a switch sw, where its output value and associated enabling signal are only guaranteed to be correct with a probability p, are the topic of Section 8 and constitute the *fifth* contribution of this work. In this section, probabilistic versions of the AND gates from Section 5 are described and their energy complexity characteristics are derived. By contrast with a lower bound of 2 in the deterministic case, we show that probabilistic AND gates can be realized with an associated *expected energy complexity* of $(1+p)$ with an associated probability of error $(1-p)$. This technology independent notion of energy complexity can be converted into physical energy estimates by interpreting the switches and hence switching, in an appropriate physical domain. For example, using estimates based on the thermodynamics of an ideal gas and based on the switch construction of Szilard [9] (also, see Section 11), each probabilistic switching step can be realized using $\kappa t \ln(2p)$ joules and, therefore, the expected energy consumed by an AND network is $(1+p) \cdot \kappa t \ln(2p)$ joules.

Reminiscent of combinational logic "networks" whose power has been systematically studied relative to models of computing such as Turing machines (see early characterizations due to Pippenger and Fischer [10] and Pippenger [11]), the model of a network $\mathcal{N}$ introduced in this paper uses an enabling signal, in the absence of which a switch is quiescent in that it does not switch and, hence, compute. In keeping with current convention, throughout this paper, we will use the term *circuit* to denote this widely studied combinational logic "network" of Boolean gates wherein measures such as *size, depth*, and *width* were of concern [12], [13], [14], [15], whereas energy was not; by contrast, the term *network* will be used to denote a structure built using our introverted switches with enabling signals. Using this terminology, every constituent "element" (in this case, a Boolean gate) in a circuit is "extroverted" and switches once for every input, whereas an introverted switch only switches when it is activated, as determined by an input enabling signal.

In Section 10, we outline relationships between networks of switches and circuits, as well as alternate models of energy-aware algorithm design and analysis introduced by this author in earlier work [2] [16]—the *ra*ndomized *b*it-level *r*andom *a*ccess *m*achine (or RABRAM). In Section 11, we shall address the increased relevance of energy consumed statically due to leakage in realizing giga and terascale circuits and the value of our definition of a network of introverted switches in this increasingly significant context.

2 Historical Remarks, Comparisons, and Perspectives

The energy characteristics of switching have their roots in thermodynamics, the history of which, traces back to the works of Carnot [17] and Clausius [18] in the early nineteenth century, leading to significant developments in the early part of the twentieth century. Influenced significantly by Maxwell (see [19] and [20], for example), the current statistical interpretation of thermodynamics was first introduced by Boltzmann and Brush [21] and was later developed by Gibbs [22] and Planck [23]. In the context of the considerations of energy consumption in this paper, the work of Boltzmann leading to the definition of *thermodynamic entropy* is especially relevant.

The notion of a value such as 0 or 1 being modeled in a physical system with a single molecule dates back to 1929 and can be attributed to Szilard [9]. In particular, his work and that of several subsequent physicists was motivated by a need to explain the celebrated *Maxwell's demon* [9] paradox, which purported to, through a thought experiment, violate the inviolate *second law* of thermodynamics. Subsequent authors credit Szilard with having envisioned the modern notion of a "bit" and a machine with two "states." While other celebrated researchers, including von Neumann [24], observed that the minimum energy needed to compute a bit is $\kappa t \ln 2$ joules, it was Landauer [25] who took a very big step toward clarifying the Maxwell's demon paradox in his widely known work. In doing so, he also explicitly laid the foundations for the (more) modern field of the thermodynamics of computation. At the heart of Landauer's work is the characterization of *erasure*, which is also a property of switching in this work.

Bennett [26] pioneered *logically reversible* computations, leading to the widely known models for reversible computing that admit computations with *energy recovery*. Subsequently, Fredkin and Toffoli [27] demonstrated logical gates that exhibit the same property. By contrast, in all of our work, we model switching as being based on *nonrecovering* modes of energy consumption and computation—energy once expended by a switching step is not recovered, even if such a recovery is physically feasible through reversible thermodynamic processes from a physical standpoint.

Moving toward realizations of "electrical switches" based on which modern computers are built—transistors being the ubiquitous vehicles—within the context of studying the inherent energy needed by deterministic switching, Meindl [28], and Meindl and Davis [29] in the context of CMOS-based switches, established fundamental limits and derived energy lower-bounds. The significance of Meindl's paper, which continues the philosophical tradition set by Szilard, von Neumann, and Landauer, among others, is the ability, for the first time, to model a switch in an idealized manner—without lossy dissipation, for example, much as we do—while, at the same time, reconciling the delicate and pragmatic balance needed to model the realities of modern semiconductor devices. In doing so (see Meindl [28]), novel techniques based on an inductive inverter-chain-based argument were developed. This technique is a crucial step in Meindl's arguments for bounding the energy consumed from below. While previous work does so implicitly, Meindl [28] is also the first in this series to provide an *explicit* physical construction of a switch, in this case as an inverter, within the context of proving a bound on the minimum energy needed, thus making the bounds rather concrete.

The inherent energy bounds for deterministic switching outlined in this work (in Section 9), while having the same conceptual goal as Meindl's approach, are distinct in a three-fold manner. First, following Szilard [9], our switches and, hence, energy limits are based on the more fundamental energy behavior of idealized monoatomic gases [21]. Second, our representations are the first characterizations of a PBIT and concomitant energy bounds associated with a probabilistic switch computing a PBIT. Thus, a study of the energy characteristics of probabilistic switching is an entirely novel contribution of this work. Finally, as outlined in Section 11, at the thermodynamic limit, the gas-based idealizations used in this paper yield potentially greater energy savings in the context of computing PBITs than those using the transistor based electrical switches following Stein [30] and Meindl [28].

A consistent theme in all of the previous work is that computation and, hence, the value of a bit being computed is deterministic—since computation, starting with Turing, was considered to be essentially a deterministic activity—and, thus, traditionally, its physical instantiation has been treated *macrophysically* [7] and has not been subjected to a statistical interpretation. Perhaps it is not an exaggeration to say that determinism is deeply ingrained, if not essential to human intuition in considerations of computing. This is perhaps the best explanation as to why an alternate style of computing—the counterintuitive notion of probabilistic algorithms that compute a value that could sometimes be wrong in their Monte Carlo "flavor"—was not considered until the 1970s from a mathematical perspective by computer scientists; this despite the readily available statistical interpretation of the value of a bit in Szilard's own construction of a molecular realization of a PBIT, based on its location in a volume of gas. As a result, previous work on modeling switches from the perspective of energy analysis is not readily applicable to a probabilistic switch designed to be deliberately erroneous.

In the context of mathematical theories of computing, Rabin and Scott's influential paper [31] introduced *nondeterminism* and broke with the tradition of determinism in a definitive manner. Subsequently, Rabin [32] also took the important step of explicitly introducing probability into the definition of an automaton [32] and studied its expressive power and relationship to a deterministic finite-state machine. Attributable perhaps to the deep seated belief in determinism in computing in general, over a decade elapsed before the role of probability became prevalent in the algorithmic computing domain following the influential

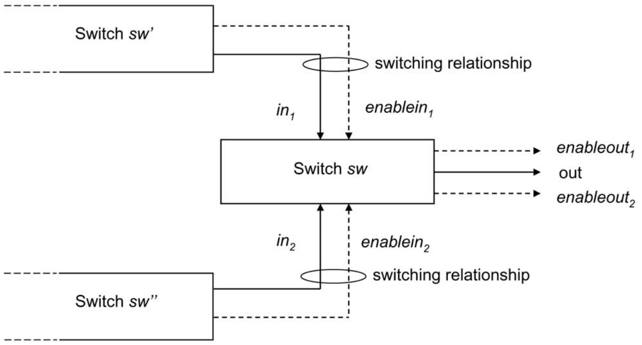


Fig. 1. The inputs to a switch and its output.

Input	output
0	0
1	1

Identity Function

Input	output
0	1
1	0

Complement Function

Input	output
0	0
1	0

Constant Function

Input	output
0	1
1	1

Constant Function

Fig. 2. The four choices for switching of which sw realizes one choice.

work of Karp within the context of average-case analysis [33], Gill's characterizations of probabilistic complexity classes [34] as well as the innovation of the Rabin-Solovay-Strassen algorithm referred to earlier.

Feynman's exposition of the thermodynamics of computing [35] uses the gas-based model for switching, albeit in the deterministic context. All of the energy estimates in this paper are based on a physical system (and a phase-space) akin to that associated with Szilard's construction and Feynman's expositions, although our modeling of a state and a PBIT that it represents is statistical and, hence, microphysical, whereas previous approaches interpreted them to be deterministic and, therefore, macrophysical considerations suffice.

3 Roadmap and Reading Guide

In Section 4.1, we define a switch and the associated notion of switching is defined in Section 4.2. Using these characteristics, in Section 5, we characterize the construction of networks of switches and, hence, Boolean gates. A novel technology or realization-independent measure of energy complexity is also introduced in Section 5. In Section 6, we outline a construction of deterministic and probabilistic *NOT* gates and outline their energy characteristics. Then, in Section 7, we prove a lower bound on the energy complexity of any deterministic network realizing an *AND* gate. In Section 8, we introduce networks of probabilistic switches, as well as the associated measure of expected energy complexity, and demonstrate energy savings relative to their deterministic counterparts, derived from *probabilistic AND* gates. Based on the definitions of a switch and of switching from Section 4, the (physical) energy characteristics of switching will be summarized in Section 9; details can be found in earlier work by this author [3]. These energy characteristics (from Section 9) allow the abstract notion of energy complexity to be interpreted in a physical domain in terms of actual energy measured in joules—in our case, through a switch realized as a Szilard engine based on an idealized monoatomic gas. In Section 10, we characterize the relationship of our networks to established models of computing such as Turing machines and circuits, as well as traditional complexity measures such as running time, space, and size. In Section 11, we contrast the energy savings derived through the PBIT realizations used in this paper and those derived using conventional transistor-based switch realizations in the electrical domain. Finally, in Section 12, we summarize the implications of probabilistic switching to the pragmatic considerations of semiconductor devices that are "nonideal" in that they are lossy and dissipate energy, as well as the potential impact on the national semiconductor (ITRS) roadmap.

4 A Switch and Switching

In Section 4.1, we will define the basic elements of a single switch and its behavior. Following this, the notion of switching will be introduced in Section 4.2. This definition will be entirely in "logical" terms without recourse to the considerations of physical realization. This formalism is reminiscent of earlier developments in the classical field, referred to as switching theory, described in Kohavi [36].

4.1 Defining a Switch

Starting with an informal introduction based on Fig. 1, each switch sw has (up to) two alternate choices for "input values" as well as "enabling signals." Each input value and enabling signal of sw is in turn the output of a distinct switch (from the set of all switches, **SW**), sw' and sw'' in the example. In this example, the outputs of switch sw' are identified with the input value in_1 and the input enabling signal $enablein_1$, whereas the outputs of sw'' are identified with in_2 and $enablein_2$. Any switch sw in turn has two possible (mutually exclusive) enabling signals as output, denoted by $enableout_1$ and $enableout_2$, as well a single output value out.

During the entire lifetime of a switch sw, each of its enabling signals $enablein_i, i \in \{1, 2\}$, is "associated with" exactly one $in_j, j \in \{1, 2\}$. Subsequently, these associations will be formalized as "switching relationships." As shown in Fig. 1, $enablein_1$ is associated with in_1 and, similarly, $enablein_2$ is associated with in_2; in general, one of the four possible associations are allowed. Finally, in this example, switch sw produces an output value as a function of in_1 whenever $enablein_1$ is active, whereas it produces an output value as a function of in_2 whenever $enablein_2$ is active, where each switch sw realizes one of the four possible 1-bit functions shown in Fig. 2. In any legal *switching* of sw, *exactly one* of its two enabling signals must be "active," indicated by associating the value 1 with it.

To further understand switching, let us suppose that $enablein_1 = 1$. Recall from Fig. 1 that the association or

Signal	Symbol
enablein	w
in	x
enableout	y
out	z

Fig. 3. Symbols for the inputs and outputs of a switch.

switching relationship between $enablein_1$ and in_1 implies that, whenever $enablein_1 = 1$, the value of *out* is determined by in_1. In this case, *sw switches* and produces an output using f; the function f is one of the four possible choices shown in Fig. 2: *identity, complement*, and the two *constant* functions—say the complement function in this example. Thus, $\text{out} = \overline{in_1}$ whenever $enable_1 = 1$, whereas $\text{out} = \overline{in_2}$ whenever $enable_2 = 1$.

For succinctness, we will use the notation in Fig. 3 to denote the values associated with the various inputs and outputs of sw. Formally, let $I_{sw,1}$ and $I_{sw,2}$ be the two *inputs* to sw wherein $I_{sw,i}$ is the ordered pair $< w_{sw,i}, x_{sw,i'} >$, where $w_{sw,i} \in \{0,1\}$ is an input enabling signal and $x_{sw,i'} \in \{0,1,\Gamma\}$ corresponds to the input value, with Γ denoting an undefined value. Thus, each switch sw is associated with a pair of *switching relationships* which are ordered pairs of the form $< w_{sw,i}, x_{sw,1} >$ and $< w_{sw,i'}, x_{sw,2} >$, where $i \neq i'$ and $i, i' \in \{1,2\}$. Intuitively, these ordered pairs capture the association between an input value and an enabling signal—as shown in the example in Fig. 1, each of the two distinct input enabling signals of sw_i is in a switching relationship with each of the two inputs. To reiterate, when an input enabling signal, say $w_{sw,i} = 1$, the corresponding value $x_{sw,1}$ is used as the input whenever $< w_{sw,i}, x_{sw,1} >$ is a valid switching relationship.

By definition, while a switch sw can produce two mutually exclusive output enabling signals $y_{sw,j}$ and $y_{sw,j'}$ with different values as outputs to two possible successor switches, it must always have the same output value z to all of its successors. In what follows, the subscript sw of w, x, y, z will be omitted whenever the use of the symbols is unambiguous.

4.2 The Process of Switching

Recall from Fig. 2 that a switch computes a fixed one-bit function f. We will now introduce a few preliminary definitions. A switch sw is *oblivious* whenever the associated function f that it computes is a constant function. It is *nonoblivious* otherwise.

Given a switch sw as above with an associated function f, a *switching* (step) is defined as follows:

1. $z = \Gamma$ and $y_1 = y_2 = 0$ whenever both its input enabling signals w_1 and w_2 have an identical value.
2. Whenever exactly one input enabling signal, say $w_i = 1$ (and $w_{i'} = 0$ for $i \neq i'$), and $< w_i, x_j >$ is a valid switching relationship,

    a. if $x_j = 0$, then $z = f(0)$, $y_1 = z$, and $y_2 = \bar{z}$,
    b. if $x_j = 1$, then $z = f(1)$, $y_1 = z$, and $y_2 = \bar{z}$.

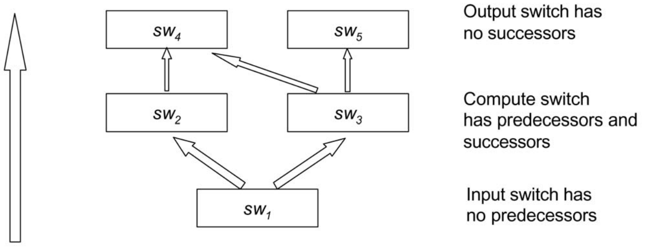

sw4
sw5
sw2
sw3
sw1
Output switch has no successors
Compute switch has predecessors and successors
Input switch has no predecessors

Fig. 4. Three types of switches.

Let $f(x) = z$ be the *deterministic* switching (step) realized by sw as above. A *probabilistic switching* (step) with a probability parameter $p \geq \frac{1}{2}$ is defined identically except $f(x) = z$ with a probability p and $f(x) = \bar{z}$ with probability $(1-p)$.

4.3 Composing Switches

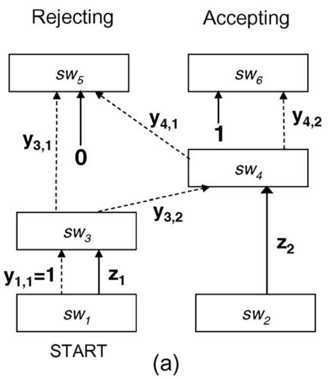
Consider a switch sw with its inputs defined by the switching relationships $< w_i, x_1 >$ and $< w_{i'}, x_2 >$ as before. An *instant of time* $\tau \in \Re^+ \cup 0$, where $\Re^+$ is the set of positive reals. Now, let τ' be the earliest instant of time when one of the two input enabling signals of switch sw, say $w_i = 1$, that is $w_i = w_{i'} = 0$, for $i \neq i'$ at any time $0 \leq \tau < \tau'$. Also, $\tau'' > \tau'$ exists and is the earliest instant when either y_i or $y_{i'}$ is 1; switch sw has *completed* switching by time τ''. Then, $\hat{\tau} = (\tau', \tau'')$ is the finite *switching interval* associated with switch sw.

Consider a switch sw' (distinct from switch sw) with output enabling signals y'_j, $y'_{j'}$, where $j, j' \in \{1,2\}$ as before, and output value z'. Switch sw' is *composed* to switch sw, denoted by $sw' \prec sw$ if and only if, for all $\tau > 0$, at least one of the following conditions applies:

1. exactly one of the values x_1 or x_2 equals z' or
2. exactly one of the input enabling signals w_1 or w_2 of sw is identical to one of the the output enabling signals y'_j or $y'_{j'}$ of sw'.

Whenever $sw' \prec sw$, switch sw' is said to *drive* switch sw or, equivalently, sw is said to be *driven by* switch sw'. Let $sw' \prec sw$. Whenever $z' = x_i$, the ordered pair $< z', x_i >$ is referred to as a *wire* and x_i is said to be *connected* to z' through the wire $< z', x_i >$. Similarly, whenever $w_i = y'_j$, we again say that the wire $< y'_j, w_i >$ connects y'_j to w_i. Finally, when a switch sw' drives sw, sw' is defined to be the *predecessor* of sw and sw is a *successor* of sw'. A switch sw is *well-connected* if and only if both the elements of at least one of its input switching relationships is connected to those of a predecessor switch through wires. Given a switch sw, note that having a predecessor switch does not guarantee that sw is *well-connected*.

To develop structures meant to realize entire computations, we will identify three types of switches: INPUT SWITCH, OUTPUT SWITCH, and COMPUTE SWITCH, as shown in Fig. 4. An INPUT SWITCH sw_1 has no predecessors and drives at least one switch of type COMPUTE SWITCH or of type OUTPUT SWITCH. An input switch does not have w and x values and its y and z values are fixed throughout the life of the network. A switch such as sw_2 in our example, which is a COMPUTE SWITCH, is driven by a switch sw_1



Switch	(Input) switching relationship	Input	Output enabling signals	
sw_3	$< w_{3,1}, x_{3,1}>$	$x_{3,1} = z_1$	$y_{3,1}$	$y_{3,2}$
		0	$1 = \bar{z}_1$	$0 = z_1$
		1	$0 = \bar{z}_1$	$1 = z_1$
sw_4	$< w_{3,2}, x_{4,1}>$	$x_{4,1} = z_2$	$y_{4,1}$	$y_{4,2}$
		0	$1 = \bar{z}_2$	$0 = z_2$
		1	$0 = \bar{z}_2$	$1 = z_2$

(b)

Fig. 5. A deterministic 2-canonical network resolving the AND function where the compute-switches sw_3 and sw_4 with input switching relationships $< w_{3,1}, x_{3,1} >$ and $< w_{4,1}, x_{4,1} >$, respectively, as well as wires $< y_{1,1}, w_{3,1} >, < z_1, x_{3,1} >, < y_{3,2}, w_{4,1} >, < z_2, x_{4,1} >$ drive the *accepting* switch sw_6 and *rejecting switch* sw_5.

which is an INPUT SWITCH. A COMPUTE SWITCH can in turn drive one or more switches that are either of a type COMPUTE SWITCH or of a type OUTPUT SWITCH. In our example, COMPUTE SWITCH sw_2 drives sw_4 which is an OUTPUT SWITCH; an output switch has no successors and is driven by at least one switch of type INPUT SWITCH or COMPUTE SWITCH. Also, when convenient, we will refer to a switch(es) as being input switch(es), output switch(es), and compute switch(es) to refer to switches of the respective types.

5 Networks for Deterministic Computations

In this section, we will first introduce a *network* of switches for deterministic computation (Sections 5.1 and 5.2) and define their energy complexity (in Section 5.3). In Section 7, we will prove a nontrivial lower bound on the energy consumed by any deterministic network that can compute the logical AND function.

5.1 Defining a Network

A *network* of switches is a connected directed acyclic graph $\mathcal{N} = (\mathbf{SW}, \mathrm{WIRES})$ such that the vertices are switches, the edges are wires, and the switches that are of the type OUTPUT SWITCH as well as those of type COMPUTE SWITCH are all well-connected. Also, each switch has no more than two predecessors and no more than two successors. This assumption of bounded (two) degree entails no loss of generality and is integrated into the definition of energy complexity introduced in Section 5.3 below. A switch is defined to be *extrinsic* if and only if at least one of its inputs is driven by an input switch. It is defined to be *intrinsic* otherwise. A network is said to be *k-canonical* for $k \geq 1$ whenever it has exactly:

1. k input switches,
2. two output switches, and
3. one compute switch sw with one of its input enabling signals $w_i \equiv y'_j = 1$ at $\tau = 0$, where y'_j is the output enabling signal of some input switch sw' driving sw.

   For convenience we will refer to switch sw' switch as the START switch.

From the perspective of the theory of computation (see Manna [37], Kohavi [36]), when the network is used as a basis for (formal) language recognition, it is convenient to view the two output switches as being either an *accepting* switch or a *rejecting* switch and hence oblivious. For simplicity and through abuse of notation, given any switch sw_i, we will use the notation $x_{i,j}$, $w_{i,j}$, and $y_{i,j}$ where $j \in \{1, 2\}$, to respectively denote its jth input value, input enabling signal, and the output enable signal, respectively. Similarly, z_i shall denote the unique output value of switch sw_i. For further notational simplicity, whenever the index i of the switch is obvious, it shall be omitted. Thus, x_j, w_j, and y_j shall denote the jth input, input enabling signal, and output enabling signal values associated with switch sw, whereas z will denote its unique output (value). Also, given an input to $\mathcal{N}$, henceforth k-canonical, which is a binary string determined by the settings of its input switches at time $\tau = 0$, the START switch sw_i, by definition, has exactly one output enabling signal, say $w_{i,j} = 1$, thus "triggering" the computation.

To illustrate the idea of a network, a 2-canonical network computing the logical AND function is sketched in Fig. 5a. Whereas switches sw_1, sw_2 are input switches, switches sw_3, sw_4 are compute switches (implementing the complement function), and switches sw_5 and sw_6 are output switches, sw_5 is the rejecting switch and sw_6 is the accepting switch. We note that every switch in this network, unless it is an input-switch, is well-connected and the network is directed and acyclic. It is a simple exercise to verify that this network is 2-canonical since, in addition to the above constraints, exactly one compute switch sw_3 has an input enabling signal $y_{1,1} \equiv w_{3,1} = 1$. In Fig. 5b, we show the crucial relationships between the input values to sw_3 and sw_4 and their output and enabling signals in a "truth-table-like" structure to help understand the structure of the wires and the connectivity.

5.2 Execution of a Network

A switch sw is said to have *switched by time* τ if and only if its switching interval (τ', τ'') is such that $\tau'' \leq \tau$. In what follows, the input switches are all assumed to have switched by time $\tau = 0$. Continuing with our example from Fig. 5, at time $t = 0$, $y_{1,1}$ is 1 since the input-switch sw_1 has switched by time $\tau = 0$ during interval $\hat{\tau}_1$ (Fig. 6). (While this implies possibly negative switching times, this is

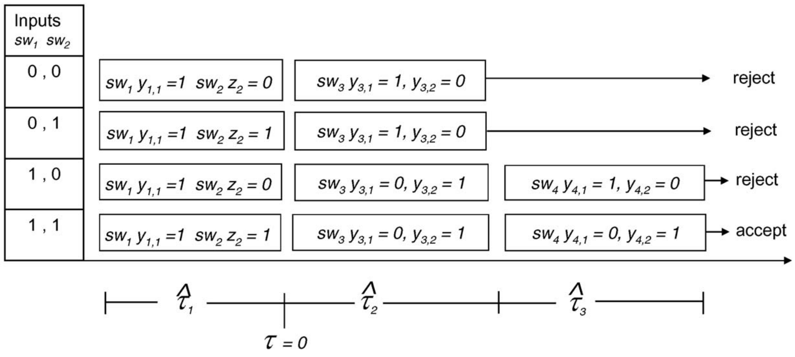

Inputs
sw_1 sw_2
0 , 0
0 , 1
1 , 0
1 , 1
$sw_1\ y_{1,1} = 1\ \ sw_2\ z_2 = 0$
$sw_3\ y_{3,1} = 1,\ y_{3,2} = 0$
reject
$sw_1\ y_{1,1} = 1\ \ sw_2\ z_2 = 1$
$sw_3\ y_{3,1} = 1,\ y_{3,2} = 0$
reject
$sw_1\ y_{1,1} = 1\ \ sw_2\ z_2 = 0$
$sw_3\ y_{3,1} = 0,\ y_{3,2} = 1$
$sw_4\ y_{4,1} = 1,\ y_{4,2} = 0$
reject
$sw_1\ y_{1,1} = 1\ \ sw_2\ z_2 = 1$
$sw_3\ y_{3,1} = 0,\ y_{3,2} = 1$
$sw_4\ y_{4,1} = 0,\ y_{4,2} = 1$
accept
$\hat{\tau}_1$
$\hat{\tau}_2$
$\hat{\tau}_3$
$\tau = 0$

Fig. 6. A trace for the deterministic network resolving the AND function.

merely a technicality and can be trivially altered to yield nonnegative switching times.) Thus, $\tau = 0$ is interpreted to denote the time when the first compute switch starts switching. In our example detailed in Fig. 6, this switch is sw_3, which is enabled at $\tau = 0$ and switches during interval $\hat{\tau}_2$. Depending on its input values, one of the two output enabling signals $y_{3,1}$ or $y_{3,2}$ assumes a value of 1. If the first input, also determined by switch sw_1, $x_{3,1} \equiv z_1 = 1$, then $y_{3,2} = 1$ at the end of the interval $\hat{\tau}_2$; by contrast, $y_{3,1} = 1$ whenever $x_{3,1} \equiv z_1 = 0$. Returning to the case where $x_{3,1} \equiv z_1 = 1$, in the next interval, $\hat{\tau}_3$ and, enabled by $y_{3,2}$, switch sw_4 switches and, upon switching, computes an output value and enables either switch sw_5 or switch sw_6. Specifically, whenever $x_{4,1} \equiv z_2$ has a value of 1, $y_{4,2} = 1$, thereby leading to an accepting computation. On the other hand, whenever $x_{4,1} \equiv z_2 = 0$, $y_{4,1} = 1$, resulting in a rejecting computation. On the other hand, when $y_{3,1} = 1$ (whenever $x_{3,1} = 0$), leading to switch sw_5 switching during interval $\hat{\tau}_3$, the input is rejected. As shown in Fig. 6, an accepting computation from network $\mathcal{N}$ corresponds to the AND gate with an input of $(1, 1)$, thus having an output value of 1, whereas a rejecting computation corresponds to an input with at least one of the two values being a 0 and, hence, an output value of 0.

More generally, let $\mathcal{N} = (\mathbf{SW}, \mathrm{WIRES})$ be a k-canonical network. An *input binding*, or input for short, is a function $I_{\mathcal{N},k} : \mathbf{SW}_{IN} \rightarrow \{0, 1\}$, where $\mathbf{SW}_{IN} \subseteq sw$ is the set of all switches of type INPUT SWITCH and $I_{\mathcal{N},k}(sw \mid sw \in \mathbf{SW}_{IN})$ is the (input) value x of switch sw, which, by definition, is either 0 or 1. An *execution* of $\mathcal{N}$ *determined by* input $I_{\mathcal{N},k}$ is a partial function $\mathcal{E} : \mathbf{SW} \rightarrow INT$, where $\mathbf{SW}' = \{\mathbf{SW} - \mathbf{SW}_{IN}\}$ and INT is the set of all intervals and $\mathcal{E}(sw) = \hat{\tau}$ is defined for a switch sw whenever

1. one of the input enabling signals w of sw is 1 at time τ, where $\hat{\tau} = (\tau, \tau')$,
2. τ is the smallest value for which this is true, and
3. switch sw has switched by time τ'.

Continuing and depending on its input value, one of the two output enabling signals of switch sw will have a value of 1 at time τ'. For example, with an input of $(1, 0)$ to the example network in Fig. 6, we see that, during interval $\hat{\tau}_2 = (\tau_2, \tau_2')$, switch sw_3 switches and that $y_{3,2} = 1$ at time τ_2'.

Let $\mathbf{SW}_{\mathcal{E}} \subseteq \mathbf{SW}'$ be such that $sw \in \mathbf{SW}_{\mathcal{E}}$ whenever $\mathcal{E}(sw)$ is defined. Given this fact, let $\hat{\tau}_f = (\tau_f, {\tau'}_f)$ be the final interval, that is, the interval with the largest starting value of time $\tau = \tau_f$ in an execution of $\mathcal{N}$ (with input $I_{\mathcal{N},k}$) and let $\mathcal{E}(sw_f) = \hat{\tau}_f$. It follows from the definition of a *k-canonical* network that

Observation 5.1. sw_f is either an accepting or a rejecting switch.

Now, consider a finite sequence of intervals $< \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \cdots, \hat{\tau}_l >$ such that $\hat{\tau}_1 = (0, \tau' > 0)$ and, for any $\hat{\tau}_i$, where $1 < i \leq l$, $\tau'_{i-1} = \tau_i$, where $\hat{\tau}_i = (\tau_i, \tau'_i)$. A sequence $\mathbf{t} =< \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \cdots, \hat{\tau}_l >$ is said to be the *trace* of a network $\mathcal{N}$ induced by execution $\mathcal{E}$ (with associated input $I_{\mathcal{N},k}$) whenever $\mathcal{E}$ is a bijection from $\mathbf{SW}_{\mathcal{E}}$ to $\mathbf{t}$. For completeness, in Fig. 6, all of the traces associated with inputs $< 0, 0 >$, $< 0, 1 >$, $< 1, 0 >$, and $< 1, 1 >$ to the network in Fig. 5 are shown. Given an input, traces will be used to characterize the execution of a network. We note in passing that, in the deterministic network from our example, each of the four input bindings is associated with an execution $\mathcal{E}$ and an associated trace.

5.3 Energy Complexity of Deterministic Networks

We will now introduce the definition of the *energy complexity* of a network. An interval $\hat{\tau}$ is oblivious in a trace $\mathbf{t}$ whenever switch sw in this interval, determined by $\mathcal{E}^{-1}(\hat{\tau})$, is oblivious. It is nonoblivious otherwise. We define the *energy characteristic* of $\mathcal{N}$ in an execution $\mathcal{E}$ represented by the function $EC : (\mathbf{SW} \times \mathbf{T}) \rightarrow \{0, 1\}$, where $\mathbf{T}$ is the set of all traces included by the execution of N to be

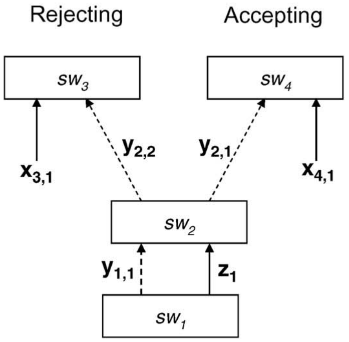
$$EC(sw, \mathbf{t}) = \begin{cases} 1 & \text{whenever } sw \text{ is a compute switch} \\ & \text{and interval } \mathcal{E}(sw) = \hat{\tau} \in \mathbf{t} \text{ is nonoblivious} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{t} =< \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \cdots, \hat{\tau}_l >$ be a trace induced by execution $\mathcal{E}$ of a network $\mathcal{N}$, corresponding to input $I_{\mathcal{N},k}$. Then, the *effective energy* of $\mathbf{t}$ is

$$EE(\mathbf{t}) = \sum_{\forall sw \in \mathbf{SW}} EC(sw, \mathbf{t}).$$

The *deterministic energy complexity* $\mathbf{E}$ of a network with respect to a family of traces $\bar{T}$ induced by input bindings $\bar{I}'_{\mathcal{N},k}, \bar{I}''_{\mathcal{N},k}, \cdots$ is

$$\mathbf{E}(\mathcal{N}) = \max_{\forall \mathbf{t} \in \bar{T}} EE(\mathbf{t}).$$



Switch	(Input) switching relationship	Input	Output	Output enabling signals	
sw2	< y1,1 , z1>	z1	z2	y2,1	y2,2
		0	1	1	0
		1	0	0	1

Fig. 7. A NOT gate as a network of switches.

The technology-independent characterization of the energy complexity of a network (of switches) can be easily extended to account for the physical energy consumed by the network. Consider a network $\mathcal{N}$ to be homogeneous if all of its compute switches consume the same switching energy, say β joules. Then,

Observation 5.2. There exists an input binding $I_{\mathcal{N},k}$ and execution $\mathcal{E}$ that induces a trace $\mathbf{t}$ such that the total energy expended by the compute switches in $\mathcal{N}$ during the trace $\mathbf{t}$ is $\mathbf{E}(\mathcal{N}) \cdot \beta$ joules.

6 Constructing NOT Gates as Networks

A NOT gate can be realized as a single compute-switch switch sw driven by one input switch which, in turn, drives an accepting and a rejecting switch, as shown in Fig. 7. Thus, the energy characteristics and complexity of this network are identical to those of a single nonoblivious switch sw with the associated function f being the complement function, as described in Section 4. We note that, in both the deterministic and the probabilistic cases, the energy complexity of this network is 1. Thus, following Section 9, a deterministic network for computing negation will cost at least $\kappa t \ln 2$ joules, whereas a probabilistic variant (following Theorem 9.3) can be realized with energy that is as low as $\kappa t \ln 2p$ joules; recall that p denotes the probability with which sw computes the output correctly.

7 Lower Bound on the Energy Complexity of an AND Network

In the sections above, the notion of a network and the energy complexity of such a network have been defined. In this section, we will prove a lower bound on the energy complexity of any network that resolves the (logical) AND function.

7.1 Some Basic Definitions

We will first introduce some helpful technical definitions to help prove the lower bound in Section 7.2. Formally, $\mathbf{B}$ is a set of k-vectors from $\{0,1\}^k$. Consider a fixed vector $\mathbf{b} \in \mathbf{B}$, $\mathbf{b} =< b_1, b_2, b_3, \cdots, b_k >$, where $b_i \in \{0,1\}$ and $k \geq 1$. Let $\mathcal{B}$ be a k-input (or k-ary) Boolean function with $\mathbf{B}$ as its domain and let $sw'_1, sw'_2, \cdots, sw'_k \in \mathbf{SW}_{IN}$ denote the input switches of a k-canonical network $\mathcal{N}$. An input $I_{\mathcal{N},k}$ *binds* $\mathcal{N}$ oo $\mathbf{b}$ if and only if, for each sw'_j, there exists a unique bit $b_{j'} \in \mathbf{b}$ for $1 \leq j, j' \leq k$ $I_{\mathcal{N},k}(sw'_j) = b_{j'}$. Recall from Section 5 that we have a unique execution $\mathcal{E}$ associated with each input $I_{\mathcal{N},k}$. We will say that, bound by $I_{\mathcal{N},k}$, $\mathcal{N}$ *resolves* $\mathbf{b}$ *under* $I_{\mathcal{N},k}$ *with respect to* $\mathcal{B}$ whenever the final switch sw_f in the trace $\mathbf{t}$ induced by execution $\mathcal{E}$ is an accepting (rejecting) switch whenever $\mathcal{B}(\mathbf{b}) = 1$ (0, respectively).

Now, consider a family of input bindings $\mathbf{I} = \bar{I}_{\mathcal{N},k}, \bar{I}'_{\mathcal{N},k}, \bar{I}''_{\mathcal{N},k}, \cdots$. A family of bindings $\mathbf{I}$ binds $\mathcal{N}$ to $\mathbf{B}$ if and only if, given any $\mathbf{b} \in \mathbf{B}$, there exists a $I_{\mathcal{N},k} \in \mathbf{I}$ such that $I_{\mathcal{N},k}$ binds $\mathcal{N}$ to $\mathbf{b}$. A network $\mathcal{N}$ *resolves a Boolean function* $\mathcal{B}$ with an input family of bindings $\mathbf{I}$ (that binds $\mathbf{B}$ to it) provided, given any $I_{\mathcal{N},k}$ that binds $\mathbf{b} \in \mathbf{B}$ to it, $\mathcal{N}$ resolves $\mathbf{b}$ under $I_{\mathcal{N},k}$ *with with respect to*$\mathcal{B}$. We define $\mathbf{I}$ to be *consistent* if and only if, given any pair of inputs $\bar{I}'_{\mathcal{N},k}$ and $\bar{I}''_{\mathcal{N},k}$ and corresponding vectors, $\mathbf{b}$ and $\mathbf{b}'$ such that $\bar{I}'_{\mathcal{N},k}(sw) = b_j \in \mathbf{b}$, then $\bar{I}''_{\mathcal{N},k}(sw) = b'_j \in \mathbf{b}'$. Informally, all consistent bindings associate the same index from the inputs $\mathbf{b}$ and $\mathbf{b}'$ with the same input switch switch sw. In what follows, we will only consider families of consistent bindings $\mathbf{I}$. For example, the bindings in Fig. 8a are consistent, whereas those in Fig. 8b are not consistent since the second element of $\mathbf{b_3}$ is bound to switch sw_1 (row 3 in Fig. 8b), whereas the first element of input vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ is bound to the same switch (in rows 1 and 2 of Fig. 8b).

Given a trace $\mathbf{t} =< \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \cdots, \hat{\tau}_l >$, a maximally oblivious subsequence $< \hat{\tau}_j, \hat{\tau}_{j+1}, \cdots, \hat{\tau}_{j'} >$ is any subsequence of intervals such that every interval in the subsequence is oblivious and, furthermore, whenever $j > 1$, $\hat{\tau}_{j-1}$ and, whenever $j' < l$, $\hat{\tau}_{j'+1}$ are nonoblivious intervals. Finally, a Boolean function $\mathcal{B}$ is nontrivial if and only if there exist $\mathbf{b}, \mathbf{b}' \in \mathbf{B}$ such that $\mathcal{B}(\mathbf{b}) \neq \mathcal{B}(\mathbf{b}')$.

Input Vector	Binding	
b1=<0,0>	sw1=0	sw2=0
b2=<0,1>	sw1=0	sw2=1
b3=<1,0>	sw1=1	sw2=0
b4=<1,1>	sw1=1	sw2=1

(a)

Input Vector	Binding		
b1=<0,0>	sw1=0	sw2=0	
b2=<0,1>	sw1=0	sw2=1	
b3=<1,0>	sw1=0	sw2=1	← Inconsistent binding
b4=<1,1>	sw1=1	sw2=1	

(b)

Fig. 8. (a) Consistent and (b) inconsistent bindings.

7.2 The Lower Bound on the Energy Complexity of an AND Network

First, we will show that

Lemma 7.1. *Given $\mathcal{N}$, let $\mathbf{t}$ be a trace and $\bar{\mathbf{t}} =< \hat{\tau}_j, \hat{\tau}_{j+1}, \cdots, \hat{\tau}_{j'} >$, where $j' = j + \lambda$ for $1 \leq j \leq j' \leq l$ be any maximally oblivious subsequence of $\mathbf{t}$. Then, $\mathcal{N}$ cannot resolve a nontrivial Boolean function if $\lambda = l - 1$ for every trace $\mathbf{t} \in \mathbf{T}$.*

Proof. The proof follows a straightforward induction on the length of the subsequence and the definition of an oblivious function. □

Lemma 7.2. *Let $(\mathbf{t}, \mathbf{t}', \mathbf{t}'', \cdots)$ be a set of traces of $\mathcal{N}$ induced by executions $(\mathcal{E}, \mathcal{E}', \mathcal{E}'', \cdots)$ associated, respectively, with inputs $(\bar{I}_{\mathcal{N},k}, \bar{I}'_{\mathcal{N},k}, \bar{I}''_{\mathcal{N},k}, \cdots)$. Also, $\bar{\mathbf{t}} =< \hat{\tau}_j, \hat{\tau}_{j+1}, \cdots, \hat{\tau}_{j'} >$ is a maximally oblivious subsequence of $\mathbf{t}$ where $\mathcal{E}^{-1}(\hat{\tau}_j) = sw$ and $\mathcal{E}^{-1}(\hat{\tau}_{j'}) = sw'$ with z' as its output. Then, whenever $(\mathcal{E}^{-1}(\hat{\tau}_j) \equiv \mathcal{E}'^{-1}(\hat{\tau'}_j) \equiv \mathcal{E}''^{-1}(\hat{\tau''}_j) \cdots)$, respectively, in traces $(\mathbf{t}, \mathbf{t}', \mathbf{t}'', \cdots)$, $\mathcal{E}^{-1}(\hat{\tau}_{j'}) \equiv \mathcal{E}'^{-1}(\hat{\tau}'_{j'}) \equiv \mathcal{E}''^{-1}(\hat{\tau}''_{j'}) \cdots \equiv sw'$ and z' is a constant in all cases.*

Proof. Follows from an induction on the length of the traces using the definitions of an input, an output to a switch, and the definition of switching. □

This lemma essentially says that, given any number of different inputs to the network $\mathcal{N}$, if the input to a switch at the "head" (in interval $\hat{\tau}_j$) of an oblivious sequence of steps sw is not changed in the respective traces, then the sequence of switches traversed by the computations from that step will be identical across all the traces leading to the same final switch at the end, switch sw', with the same output value z'. These two lemmas will enable us to prove the lower bound now.

Theorem 7.1. $\mathbf{E}(\mathcal{N}) \geq 2$ *for any 2-canonical network $\mathcal{N}$ that resolves the Boolean AND function.*

Proof (sketch). If the theorem is false for a network $\mathcal{N}$, in any trace $\mathbf{t}$ induced by any execution $\mathcal{E}$ with input binding $I_{\mathcal{N},k}$, there can be no more than one nonoblivious interval. Without loss of generality, let this interval be τ_j for $1 \leq j \leq l$, where $\mathbf{t} =< \tau_1, \tau_2, \tau_3, \cdots, \tau_l >$. If there is no such interval, $\mathcal{N}$ cannot resolve a nontrivial Boolean function from Lemma 7.1. Let $\mathcal{E}^{-1}(\tau_j) = sw$. Switch sw must be extrinsic or else, from Lemma 7.2, once again, $\mathcal{N}$ cannot resolve a nontrivial Boolean function. Since the output of $\mathcal{N}$ is a constant across all inputs.

Without loss of generality, let input switch sw' drive sw and consider inputs $\mathbf{b_1} \equiv< 1, 1 >$, $\mathbf{b_2} \equiv< 0, 1 >$, and $\mathbf{b_3} \equiv< 1, 0 >$ to $\mathcal{B}$, which is an AND function. From pigeon-holing, there exist two inputs in any consistent family of input bindings, say binding $\mathbf{b_1}$ and $\mathbf{b_2}$ to $\mathcal{N}$ without loss of generality, such that the input value to sw derived from sw' is identical in both cases. However, from the definition of a Boolean AND, $\mathcal{B}(\mathbf{b_1}) \neq \mathcal{B}(\mathbf{b_2})$. Then, from Lemma 7.2 and the fact that $\mathcal{N}$ is 2-canonical, we know that, in both cases, the corresponding traces have the same accepting or rejecting switch sw in their final interval, which contradicts the fact that $\mathcal{N}$ resolves the Boolean AND function since $\mathcal{B}(\mathbf{b_1}) \neq \mathcal{B}(\mathbf{b_2})$. □

8 Networks for Randomized Computations

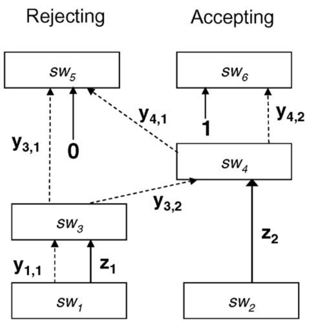
In Section 5, the notion of a *deterministic* network that can resolve a Boolean function was introduced and its energy complexity defined. Implicit to this definition is the fact that every switch sw in network $\mathcal{N}$ is a deterministic switch. In Section 4.2, the notion of a probabilistic switch was introduced. Analogous to deterministic network constructions, we will now use this definition to construct probabilistic networks and characterize their energy complexity.

8.1 Probabilistic Networks and Their Execution

A k-canonical probabilistic network $\mathcal{R}$ is any network with the property that a COMPUTE SWITCH can either be deterministic or probabilistic. A *probabilistic execution* is $\hat{\mathcal{E}} : \mathbf{SW} \rightarrow INT$ as before, where the individual switches in sw are probabilistic with some probability parameter p. Given a k-canonical probabilistic network $\mathcal{R}$ and one input binding $I_{\mathcal{R},k}$, we have an associated family of *probabilistic executions* $\hat{\mathcal{E}}_1, \hat{\mathcal{E}}_2, \hat{\mathcal{E}}_3, \cdots, \hat{\mathcal{E}}_{l'}$ inducing a corresponding *family* of traces $\mathcal{F} \equiv \{\rho_1, \rho_2, \rho_3, \cdots, \rho_{l'}\}$, with respective probabilities $r_1, r_2, r_3, \cdots, r_{l'}$. Let $\hat{\mathcal{E}}_i(sw) = \hat{\tau}_j$ as before and let sw' be the switch in interval $\hat{\tau}'_{(j-1)}$ immediately preceding $\hat{\tau}_j$ in trace ρ_i, where $j > 1$, that is, $\hat{\tau}_{(j-1)}$ and $\hat{\tau}$ are of the form (τ'', τ) and (τ, τ'), respectively. Also, let x' be the input value of switch sw' at time τ''. (We recall that, in any legal execution, the input value to a switch is defined and one of its input enabling signals is asserted.) Now, the conditional probability q_j associated with $\hat{\tau}_j$ in trace ρ_i is the probability that the output enabling signal from sw' driving sw equals one at time τ given the input to sw' as determined by $I_{\mathcal{R},k}$ is x'. Informally, each member of this family of traces is generated by the probabilistic or randomized nature of the switches in $\mathcal{R}$. This is in contrast to a deterministic network $\mathcal{N}$, which has a unique trace given an input binding $I_{\mathcal{N},k}$. The following observation is immediate.

Observation 8.1. By definition, $r_i = \prod_{1 \leq j \leq l} q_j$, where $\rho_i =< \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \cdots, \hat{\tau}_{l'} > \tau_l$. Furthermore, $\sum_{1 \leq j' \leq l'} r_{j'} = 1$.

In Fig. 9, we show a probabilistic network $\mathcal{R}$ that can resolve the logical AND function. The details of its construction are identical to those in the deterministic case from Fig. 5. Switches sw_3 and sw_4 are probabilistic compute switches, with a probability parameter p. As before, they compute the complement function probabilistically. As in the case of the deterministic network, the output enabling signals y are shown as a function of the inputs in the table in Fig. 9. A probabilistic network can, depending on the execution, invoke different switches with varying probability parameters each leading to a distinct trace with the same input. Thus, with an input of $x_{3,1} = 1$, switch sw_3 (top row in the table of Fig. 9) can result in two distinct traces, one enabling output switch sw_5 incorrectly with a probability of $(1 - p)$ and the other enabling compute switch sw_4 correctly with a probability of p, respectively, through output enabling signals $y_{3,1}$ and $y_{3,2}$. This is in contrast to the deterministic case, where a fixed input is associated with a single (or unique) trace, whereas, in the probabilistic case, it is associated with a family of traces whose relative probabilities are characterized by Observation 8.1.



Switch	(Input) switching relationship	Input	Output enabling signals	
sw_3	$< y_{1,1}, z_{1,1} >$	z_1	$y_{3,1}$	$y_{3,2}$
		0	$1 = \bar{z}_1$ (p) $0 = z_1$ (1-p)	$0 = z_1$ (p) $1 = \bar{z}_1$ (1-p)
		1	$0 = \bar{z}_1$ (p) $1 = z_1$ (1-p)	$1 = z_1$ (p) $0 = \bar{z}_1$ (1-p)
sw_4	$< y_{3,2}, z_{2,1} >$	z_2	$y_{4,1}$	$y_{4,2}$
		0	$1 = \bar{z}_2$ (p) $0 = z_2$ (1-p)	$0 = z_2$ (p) $1 = \bar{z}_2$ (1-p)
		1	$0 = \bar{z}_2$ (p) $1 = z_2$ (1-p)	$1 = z_2$ (p) $0 = \bar{z}_2$ (1-p)

Fig. 9. A probabilistic 2-canonical network resolving the AND function.

8.2 Energy Complexity of Probabilistic Networks

We will now introduce the notion of the *energy complexity* of a probabilistic network. Let $\rho =< \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \cdots, \hat{\tau}_l >$ be a trace from the family $\mathcal{F}$ induced by executions associated with a binding $I_{\mathcal{R},k}$. Then, the *expected effective energy* of $\mathcal{F}$ is

$$REE(\mathcal{F}) = \sum_{\rho_i \in \mathcal{F}} r_i \cdot EE(\rho_i),$$

where r_i is the probability of trace ρ_i. These definitions are an adaptation of the measure of *logical work* introduced by Palem [2] in the context of a RABRAM. The *randomized energy complexity* **RE** of a network is

$$\mathbf{RE}(\mathcal{R}) = \max_{\forall \mathcal{F} \in \mathbf{F}} REE(\mathcal{F}),$$

where **F** is the *set of all trace families* induced by executions associated with all the consistent input bindings $\bar{I}_{\mathcal{R},k}, \bar{I}'_{\mathcal{R},k}, \bar{I}''_{\mathcal{R},k}, \cdots$. It is a simple exercise to determine that:

Observation 8.2. The randomized energy complexity **RE** of the 2-canonical network for resolving the Boolean AND function (in Fig. 9) is $(1 + p)$ and, hence, is lower than the minimum energy complexity of 2 associated with any deterministic network.

Proof. Immediate from Theorem 7.1), whenever $p < 1$. □

9 Energy Characteristics of Switching

We will now relate the "realization" independent notion of energy complexity from the previous sections to physical energy. Recall that $\hat{\tau}$ is the switching interval associated with switch sw. Given the set of all switches **SW**, let $\mathrm{ENERGY} : \mathbf{SW} \rightarrow \Re^+$ be the *switching energy*, representing the energy consumed in joules by a switch sw during its associated switching interval $\hat{\tau}$.

Based on the analysis presented in detail in this author's earlier work [3], the switching energy associated with deterministic nonoblivious switching can be bound from below as follows:

Theorem 9.1. *The switching energy* $\mathrm{ENERGY}(sw) \geq \kappa t \ln 2$ *joules whenever* $\hat{\tau}$ *is nonoblivious and* sw *is a deterministic switch.*

Combining this theorem with the lower bound (Theorem 7.1) on energy complexity, we have

Corollary 9.2. *There exist inputs to any 2-canonical network that resolve the Boolean* AND *function such that the energy consumed is at least* $2\kappa t \ln 2$ *joules.*

Also, given a probability parameter $p \geq \frac{1}{2}$, the following theorem (again from [3]) characterizes the energy consumed by probabilistic switching.

Corollary 9.3. *The switching energy* $\mathrm{ENERGY}(sw)$ *can be as low as* $\kappa t \ln 2p$ *joules whenever* sw *is a nonoblivious probabilistic switch with a probability parameter* p.

From this, we can immediately deduce:

Corollary 9.4. *The potential for saving through probabilistic switching over deterministic switching is* $\kappa t ln \frac{1}{p}$ *joules per switching step.*

Moving from the deterministic to the randomized or probabilistic context, let a probabilistic network $\mathcal{R}$ be r-homogeneous if and only if it is homogeneous and all of its compute switches are randomized with an identical probability parameter, p.

Observation 9.5. The maximum energy in joules consumed by the compute-switches of a r-homogeneous probabilistic network $\mathcal{R}$, across all of its inputs, averaged for each input $I_{\mathcal{R},k}$ over all the traces induced by executions $\hat{\mathcal{E}}_1, \hat{\mathcal{E}}_2, \cdots$, can be as low as $\mathbf{RE}(\mathcal{R}) \cdot \kappa t \ln 2p$ joules.

From Theorem 9.3 and in any switching, a probabilistic switch with probability parameter p consumes $\kappa t \ln 2p$ joules. Thus, using this fact and since $\mathbf{RE} = (1 + p)$ for a 2-canonical probabilistic network for computing the AND function, we have, from Observation 9.5:

Corollary 9.6. *The expected energy consumed by switches of type compute-switch in the probabilistic network for computing the* AND *function can be as low as* $-(1 + p)\kappa t \ln(2p)$ *joules. When* $p < 1$*, this is less than its deterministic counterpart, which is a minimum of* $-2\kappa t \ln(2)$ *joules.*

10 Complexity Theoretic Characterization of the Power of Networks

As shown earlier, a switch sw can be used to realize AND as well as NOT "gates." *Disjunction or* OR *gates can be similarly realized whose energy complexity in the deterministic and* *probabilistic cases are identical to those established for AND gates*. It will be useful to extend these foundational constructs and results to the broader scope of realizing entire computations and designing energy-aware algorithms, using these switch constructs as building blocks. To accomplish this goal, we will sketch relationships below, between a network $\mathcal{N}$ and established models of computation such as Turing machines and circuits, from the theory of computation; Papadimitriou [12] and Sipser [13] provide introductions to the topic. Our goal in providing the characterization sketched below is primarily to help place our notion of a network and its associated energy complexity in familiar terrain.

10.1 Switching and the RABRAM XModel of Computing

In an earlier paper [2], this author introduced the RABRAM model for energy-aware algorithm analysis and design. The RABRAM and the model of a network as introduced in this paper are equivalent—in the RABRAM, the address decoder is abstracted away and is a potential source of additional computational power, where a single RABRAM program can correspond to an unbounded family of networks, one for each input length. A central contribution of this earlier work is the demonstration of asymptotic energy savings in the RABRAM model, in the context of the basic question of detecting whether a given vector of n elements which are drawn from the set $\{0, 1\}$, contains at least one element which is equal to 0. This problem is referred to as the *distinct vector problem*, for variants for which the following results are established (in [2]). Using lower bounds for the deterministic case and upper bounds for the probabilistic case, which has a probability of error bound from above by $\frac{1}{n^c}$, *energy savings* were shown to grow asymptotically in n, using a *probabilistic value amplification* technique. An interesting aspect of this result is that (as far as can be determined) it is the first asymptotic demonstration of energy savings derived from a probabilistic algorithm when compared to any deterministic counterpart wherein the complexity of the running time is asymptotically the same $(\Theta(n))$ in both cases. This result demonstrated that the energy savings are due to probabilistic "switching" as opposed to being a byproduct of an improvement to the running time achieved by randomization since, intuitively, a lower running time may imply lower energy consumption.

10.2 Networks and Circuit Complexity

To reiterate, we will use the term *circuit* to refer to the form of Boolean circuits that have become ubiquitous in the study of the complexity of computing, and *networks* to refer to the particular model for computing introduced in this paper. A crucial body of work in the context of circuits, using this terminology, is that of Pippenger and Fischer [10]; we will use this paper as a basis for definitions. We recognize that, since this early characterization, significant strides have been made in circuit complexity, notably in clarifying the power of "monotonicity," as in the work of Razborov [14], [15].

Following Pippenger and Fischer [10], two Turing machines M, M' *simulate* each other if, when they are started with the same string of symbols on their input tapes, they produce the same string of symbols *online* on their output tapes; two machines that simulate each other do so online if the shifts of the input and output heads occur in the same order (but not necessarily at the same steps) for both machines. This notion of simulation can be naturally extended to involve *intersimulation* between Turing machines, networks (in the sense used in this paper), and circuits, viewed as language recognizers within stated resource bounds. As in the case of circuits, a potentially infinite set of networks $\mathcal{N}_1, \mathcal{N}_2, \ldots$ correspond to a single Turing machine, in one-one correspondence with each distinct input size $1, 2, \ldots$ to M. In the context of online simulation, the energy complexity of networks is related to the number of steps taken by any Turing machine, whereas the size of a circuit is related to the number of steps of an *oblivious* Turing machine. Thus, the energy complexity of a network and the size of a circuit are related by a gap determined by the separation between oblivious and nonoblivious Turing machines using time-complexity as a measure.

11 Physical Representation of PBITS and Energy Savings

In this work (in Section 9) and as developed earlier by this author in [1] and in [2], a bit is represented by a group of classical *microstates* and the output value of switching is determined through an instantaneous (classical) "measurement" that detects the existence of a "witness" microstate to the value of 0 or 1. An example of such an instantaneous measurement is the detection of the position of a molecule of gas in a cylinder containing it, as described by Szilard [9]. This approach to representing the value of a PBIT is to be contrasted with the traditional approach to representing PBITs as voltages and measured as averages as characterized by Stein [30] and by Meindl [28]; we will briefly outline the relationship between these alternate representations here.

A representation of a PBIT in the electrical domain would use a value such as voltage, whose *mean* is the value of the PBIT as shown in Fig. 10. A normally distributed noise signal whose mean value is the intended PBIT value determines the behavior of the system. Then, the probability of detecting a particular voltage—assuming an arbitrarily precise instantaneous measurement device—is determined by the normal distribution and the actual value will be centered around the current mean. In this representation and as shown in Fig. 10, a value $\text{PBIT} = 0$ is the interval of the normalized voltage scale $(00, 0.5]$, whereas $\text{PBIT} = 1$ corresponds to the interval $[0.5, 00)$. The overlap in the two normally distributed random variables between these two intervals, denoted by the hatched area $\mathbf{A}$ in the figure, represents the region where a measurement can lead to an erroneous value. Quantitatively, this area of overlap between the two density functions, $\mathbf{A}$, represents the error of a PBIT value of 0 being erroneously detected to be 1 and vice versa. Thus, using the notation from this paper, the one-sided error $(1 - p) = \frac{A}{2}$. Comparisons between the energy savings using probabilistic computing, between the canonical realization of a PBIT and its novel counterpart

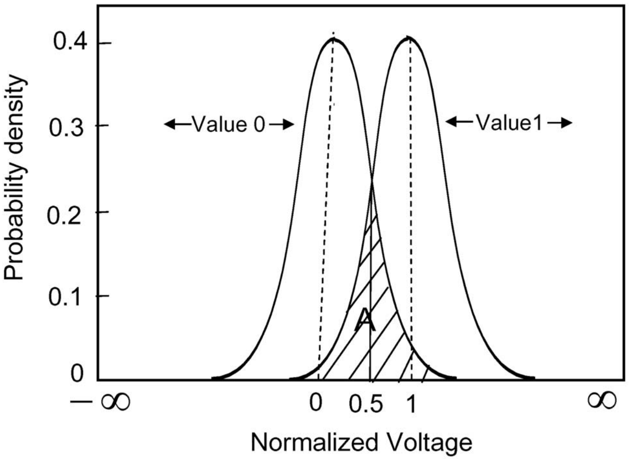


Fig. 10. Realization of a PBIT where a value of 0 is defined to be a voltage in the interval $(00, 0.5]$, whereas a 1 is defined to be a value in the interval $[0.5, 00)$.

were determined in collaboration with Palem et al. [38] and Suresh et al. [39]; for a detailed discussion, the reader is referred there. As shown in Fig. 11, while energy savings are possible in both realizations, for a fixed probability p, in an idealized thermodynamic sense, a realization of a PBIT based on an approach inspired by a Szilard engine requires lower energy than the canonical representation in the electrical domain following the approach of Stein [30].

12 PBITS, LEAKAGE, AND THE NATIONAL SEMICONDUCTOR ROADMAP

In this section, we will outline issues related to the energy savings gleaned from probabilistic switching and devices for realizing them. First, in Section 12.1, we provide an overview of the anticipated impact that probabilistic switching has on Moore's law as projected by the current roadmaps. An emerging and increasingly important aspect of these roadmap projections is the energy consumption attributed to *leakage*. In Section 12.2, we will outline the potential that the switching constructs outlined in this paper have for overcoming the challenges posed by leakage. This has to be contrasted with the fact that, as described thus far, our probabilistic switching elements and their associated complexity framework address energy consumption due to *dynamic switching*.

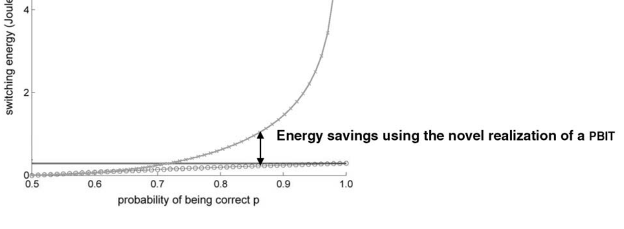


Fig. 11. Comparing energy savings due to probabilistic switching based on PBIT realizations through a monoatomic gas referred to as the novel representation and the more conventional realization through voltages referred to as the canonical representation.

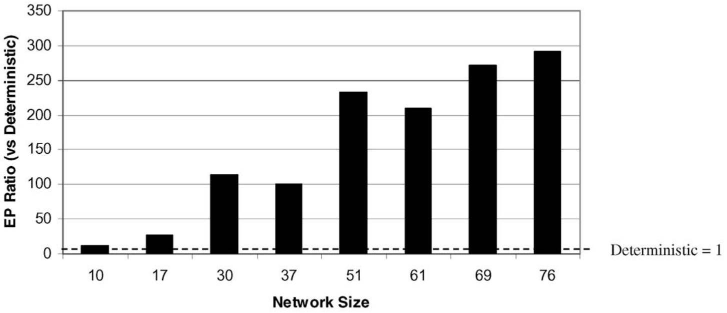


Fig. 12. The improvements to *energy X performance (running time)* in the context of a Bayesian network as a ratio of the deterministic case to the case wherein the probabilistic computing steps are realized in CMOS with thermal noise being used as a source of randomness.

12.1 Probabilistic Switches as a Basis for Accelerating Moore's Law

A significant issue in modern computing devices, largely based on CMOS semiconductor material, is lossy *dissipation*. Feynman [35, Chapter 7] discusses this topic and its impact on the idealized devices that are the basis for the energy estimates in this paper. In devices available today, the switching energy tends to be several orders of magnitude above the ideal dissipationless value of $\kappa t \ln 2\dot{p}$ based on the idealized models used in this paper. Briefly, if lossy dissipation is factored in, the savings due to probabilistic switching are expected to be even greater than those outlined at the fundamental limit in Section 9. We will briefly discuss this phenomenon and its impact on energy savings now.

In Fig. 12, we show the trends in energy consumed by realistic CMOS devices with a feature size if 0.25μ. Using the well-known Trimaran framework and in collaboration with Chakrapani and Seshasayee [40], we have shown the energy savings through probabilistic switching savings can be significant indeed and much higher than the amounts characterized at the limit in Corollary 9.4. In this figure, the energy savings in the context of a well-known probabilistic algorithm, the Bayesian network, are illustrated. Thus, with a Bayesian network of greater than 60 nodes and as shown in the figure, the $(energy \times performance(numberofcycles))$ is better by a (multiplicative) factor of over 120 over deterministic switching—deterministic computing is based on a StrongArm SA1100 processor whose energy per-switching step can be estimated from jouleTrack [41] —through probabilistic CMOS devices with $p = 0.85$.

One way of interpreting the trends illustrated in Fig. 12 is to view probabilistic CMOS-based switching (devices) as a basis for accelerating Moore's law as projected by the national semiconductor (ITRS) roadmap, wherein the exact amount of acceleration depends on the feature size of the technology and, hence, the point in time under consideration. The term "acceleration" in this context implies that the energy-performance benefits that Moore's law characterizes in the context of deterministic or conventional CMOS devices and which is at the heart of the tremendous growth

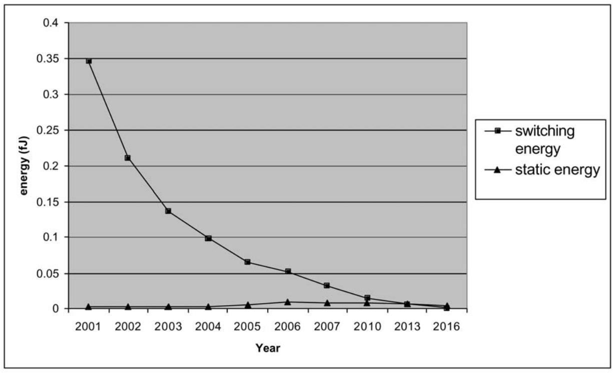


Fig. 13. The relationship between switching energy and static power dissipation.

of the computing industry can be gleaned by an earlier generation of probabilistic CMOS devices within the context of probabilistic applications or workloads. Thus, probabilistic switching can be a basis for achieving the benefits of Moore's law earlier, albeit with an associated probability of being correct $p < 1$; for a detailed discussion of this trend, please see the work of Chakrapani et al. [40]. All of the above predictions are empirical; for those interested in a theoretical approach to modeling and analyzing dissipation and its impact on switching, the reader is referred to Gupta's work [42].

12.2 Coping with Increased Leakage

In Fig. 13, we have shown the rate at which deterministic switching energy drops with progress in CMOS technology, based on the 2002 ITRS roadmap. As shown there, *static energy consumed* as a result of leakage is increasing relative to that consumed by switching. Thus, while switching energy tends to be the dominant issue now and the improvements projected by probabilistic switching shown in Fig. 12 will be a significant factor for some time, after 2010, additional improvements to overcome static energy consumption—efforts underway by technologists in the semiconductor arena now—become increasingly significant. In this context, a feature of switching and networks that constituted most of the contents of this paper offers an unanticipated and potential remedy. Specifically, a switch sw in our formulation is *not* enabled or is dormant unless its input enabling signal w is active, denoted by a value of 1. Thus, switches that are not enabled in this sense are quiescent and, hence, are *introverted*—they are active or respond only when enabled.

In interpreting such a device in CMOS terms [43], this in turn implies that networks realized from such introverted switches can be a basis for minimizing static energy consumption due to leakage. Conceptually, this notion is similar to the idea of enabling entire blocks on logic only when needed, leading to a style of circuits and computing referred to as *guarded evaluation* by Tiwari et al. [44]. One way of viewing an introverted switch is that it allows guarded evaluation at the finest level of granularity possible—that of one-bit functions—wherein the switches are off when they are not in use. The entire methodology of realizing networks (or circuits) based on our notion of introverted switches to cope with the challenge posed by leakage and the concomitant and novel challenges raised to logic synthesis and design using them as building blocks is an unexplored and potentially valuable direction of research from the viewpoint of ameliorating the problems posed by static power dissipation due to leakage. If our networks are interpreted to be asynchronous or clock-free—a context to which the development in this paper is readily applicable—then the notion of an introverted switch can, yet again, be viewed as a one-bit version of a "reactive gate" that is at the core of designing asynchronous processors (see Martin et al. [45]).

13 Concluding Remarks

The work described in this paper clearly builds substantially on the surprising success of the use of probability in deriving efficient algorithms wherein running time was the primary criterion for success. Thus, Karp's average-case analysis [33], as well as the Rabin-Solovay-Strassen algorithm referred to earlier on, which led to the field of probabilistic computing, are important examples that spawned the use of probability in algorithm design and analysis. The work in this paper takes a different view from these breakthroughs by interpreting probability to be a byproduct of a physical phenomenon, ubiquitous to nature. Please see Kish [46] for an excellent analysis of the impact of noise viewed as an impediment to sustaining the projections of Moore's law. In contrast to the universal view, wherein noise is thus viewed as an impediment, our approach takes the diametrically opposed view of viewing it as a "resource" as an aid to achieving low-energy probabilistic devices and computing. Thus, most devices based on CMOS or other physical media are, by their very nature, "unstable" "noisy" or, from our perspective, *inherently* probabilistic. With these naturally unstable devices as a starting point, energy is spent in deriving stability so that what is popularly considered to be (deterministic) computing is realized. If this energy is not spent quite to the same extent and thus saved, these devices provide a naturally probabilistic switch. Such a switch is described and the concomitant energy savings are quantified in this work for the first time. A detailed discussion of the relationship between this work and the rich history of the role of probability in computing can be found in a paper by this author [16].

The field of thermodynamics of computing, starting with Landauer's work [25] and leading to the reversible computing concepts of Bennett [26] as well as Fredkin and Toffoli [27], had much to offer in terms of a philosophy for reasoning about physical models of computing that are simple and idealized—this author credits Feynman [35] with providing a helpful exposition for characterizing energy-aware computing abstractly using the thermodynamics of monoatomic gases based on Szilard's [9] work as a foundation. However, the work presented in this paper departs from the reversible computing framework, which can, in the ideal case, realize computations with no energy expenditure at all—specifically, the work described in this paper deals with the style of *nonrecovering* computing ubiquitous to any switch in any commercially available modern-day computer, while the analytical models used here aand idealizations following Feynman's approach. Thus, while it is not reasonable to expect that computers will be built in Silicon Valley and elsewhere using models based on the physics of monoatomic gases used here (also see [1], [2], [3] for details), as Feynman argues, these models provides a clean and abstract foundation for reasoning about energy-aware computing without the significant nonlinearities introduced by the more practical CMOS-based models (briefly discussed in Section 11 and developed further in [38]). Thus, our thermodynamic formulations based on the models of a perfect gas can serve as a convenient basis for identifying significant trends and physical limits, even though the precise and quantitative energy estimates will not be directly applicable to the context of CMOS-based switches.

With this work as a starting point, four distinct areas of research seem to hold promise in terms of further exploration. The first direction concerns gaining a deeper understanding of the inherent energy consumed within the style of *nonrecovering* computing advocated in this paper. Thus, the results claimed in Section 9 (and proved in [3]) can be viewed as "at least" estimates in the context of energy savings achieved through probabilistic switching. It will be interesting to understand the true behavior in terms of "at most" estimates of the savings using realistic CMOS devices, further developing the approach outlined in Section 12.1. The second direction involves a systematic study of the energy-complexity of algorithms as well as complexity theoretic work, based on the models presented here and this author's earlier work [2], [39] that introduced the RABRAM model. A third and, from a pragmatic perspective, extremely important direction involves the issue of realizing CMOS-based probabilistic switches as well as their integration into circuits, thus forming a substrate for computing. An important first in this direction has been taken by interpreting the results from the previous sections in the context of the electrical domain, as well as plausible realizations as devices [38], [39], [47]. The fourth and final direction concerns using an introverted switch in conventional deterministic hardware (digital) designs as well as in probabilistic designs (as discussed in Section 12.2) as a basis for energy savings and for dealing with the increasing and significant challenge posed by static dissipation due to leakage.

Acknowledgments

The author wishes to thank Lakshmi Chakrapani, Yogesh Chobe, Pinar Korkmaz, Bilge Ebru Akgul, and Rodric Rabbah, in that order, for helping with typesetting and other related editorial issues. Rodric Rabbah helped verify the results pertaining to the relationships between networks, circuits, and Turing machines. Zeke Zalcstein's comments on the previous paper and his criticism of the abstract notion of a logical mechanism introduced by this author were crucial in motivating this paper. Jack Schwartz clinched this issue by insisting on a concrete notion of a *switch* to help clarify the subtleties of this work. He also pointed out the importance of clarifying the deeper issues underlying the ergodicity of the gaseous system, as well as the concerns caused by dissipation in the domain of actual switching when it is not idealized. For all of this, the author is very grateful. Suresh Cheemalavagu has been of immense help as a *sounding board* in ensuring the engineering relevance of this work, and in verifying the correctness of several of the subtler relationships between the physical domain and the mathematical formalisms; he also helped with numerous editorial comments and provided the data used in this work. A very special thank you to Bob Graybill who supported this work in part through US Defense Advanced Research Projects Agency seedling contract #F30602-02-2-0124, which was critical to its reaching maturity.

References

[1] K.V. Palem, "Thermodynamics of Randomized Computing, a Discipline for Energy Aware Algorithm Design and Analysis," Technical Report GIT-CC-02-56, Georgia Inst. of Technology, Nov. 2002.

[2] K.V. Palem, "Energy Aware Computation: From Algorithms and Thermodynamics to Randomized (Semiconductor) Devices," Technical Report GIT-CC-03-10, Georgia Inst. of Technology, Feb. 2003.

[3] K.V. Palem, "Energy Aware Computing through Randomized Switching," Technical Report GIT-CC-03-16, Georgia Inst. of Technology, May 2003.

[4] J.T. Schwartz, "Fast Probabilistic Algorithms for Verification of Polynomial Identities," *J. ACM,* vol. 27, pp. 701-717, 1980.

[5] M.O. Rabin, "Probabilistic Algorithms," *Algorithms and Complexity, New Directions and Recent Trends,* J.F. Traub, ed., pp. 29-39, 1976.

[6] G.J. Chaitin and J.T. Schwartz, "A Note on Monte Carlo Primality Tests and Algorithmic Information Theory," *Comm. Pure and Applied Math.*, vol. 31, pp. 521-527, 1978.

[7] R. Balian, *From Microphysics to Macrophysics,* vols. 1, 2. Springer-Verlag, 1991.

[8] U.V. Vazirani and V.V. Vazirani, "Efficient and Secure Pseudo-Random Number Generation (Extended Abstract)," *Proc. Ann. Symp. Foundations of Computer Science,* pp. 458-463, 1984.

[9] L. Szilard, "On the Decrease of Entropy in a Thermodynamic System by the Intervention of Intelligent Beings," *Maxwell's Demon: Why Warmth Disperses and Time Passes,* H. Leff and E. Rex, 1998.

[10] N. Pippenger and M.J. Fischer, "Relations among Complexity Measures," *J. ACM,* vol. 26, pp. 361-381, Apr. 1979.

[11] N. Pippenger, "On Simultaneous Resource Bounds (Preliminary Version)," *Proc. 20th Ann. Symp. Foundations of Computer Science,* pp. 307-311, 1979.

[12] C.H. Papadimitriou, *Computational Complexity.* Addison-Wesley, 1994.

[13] M. Sipser, *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.

[14] A.A. Razborov, "A Lower Bound on the Monote Complexity of the Logical Permanent," *Mat. Zametki,* vol. 37, no. 6, English translation in *Math. Notes of the Academy of Sciences of the USSR,* 1985.

[15] A.A. Razborov, "Lower Bounds on the Monote Complexity of Some Boolean Functions," *Dokl. Akad. Nauk SSSR,* English translation in *Soviet Math. Dokl.*, vol. 31, pp. 798-801, 1985.

[16] K. Palem, "Proof as Experiment: Computation from a Thermodynamic Perspective," *Proc. Int'l Symp. Verification: Theory and Practice,* June 2003.

[17] S. Carnot, *Reflections on the Motive Power of Fire (Reflexions sur la Puissance Motrice du Feu et sur les Machines Propres a Developper Cette Puissance),* 1824.

[18] R. Clausius, "Uber die Bewegende Kraft der Warme," *Annalen der Physik,* 1850.

[19] J.C. Maxwell, "Illustrations of the Dynamical Theory of Gases," *Phil. Mag.,* vol. 19, pp. 19-32, 1860.

[20] J.C. Maxwell, "On the Dynamical Theory of Gases," *Philosophical Trans. Royal Soc. London,* vol. 157, pp. 49-88, 1867.

[21] L. Boltzmann and S.G. Brush, *Lectures on Gas Theory,* English translation by S.G. Brush. Dover Publications, 1995.

[22] J.W. Gibbs, *Elementary Principles in Statistical Mechanics.* New York: Scribner, 1902.

[23] M. Planck, *Treatise on Thermodynamics.* Dover Publications, 1922.

[24] J. von Neumann, *Fourth University of Illinois Lecture in Theory of Self-Reproducing Automata,* A. W. Burks, ed. Univ. of Illinois Press, 1966.

[25] R. Landauer, “Irreversibility and Heat Generation in the Computing Process,” *IBM J. Research and Development,* vol. 3, pp. 183-191, July 1961.

[26] C.H. Bennett, “Logical Reversibility of Computation,” *IBM J. Research and Development,* vol. 17, pp. 525-532, Nov. 1973.

[27] E. Fredkin and T. Toffoli, “Conservative Logic,” *Proc. Physics of Computation Conf.,* pp. 219-253, 1982.

[28] J.D. Meindl, “Low Power Microelectronics: Retrospect and Prospect,” *Proc. IEEE,* pp. 619-635, Apr. 1995.

[29] J.D. Meindl and J.A. Davis, “The Fundamental Limit on Binary Switching Energy for Terescale Integration (TSI),” *IEEE. J. Solid-State Circuits,* pp. 1515-1516, Oct. 2000.

[30] K.-U. Stein, “Noise-Induced Error Rate as Limiting Factor for Energy per Operation in Digital ICS,” *IEEE J. Solid-State Circuits,* vol. 31, no. 5, 1977.

[31] M.O. Rabin and D.S. Scott, “Finite Automata and Their Decision Problems,” *IBM J. Research and Development,* vol. 3, no. 2, pp. 115-125, 1959.

[32] M.O. Rabin, “Probabilistic Automata,” *Information and Control,* vol. 6, pp. 230-245, 1963.

[33] R.M. Karp, “Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane,” *Math. Operations Research,* vol. 2, no. 3, pp. 209-224, Aug. 1977.

[34] J. Gill, “Computational Complexity of Probabilistic Turing Machines,” *SIAM J. Computing,* vol. 6, no. 4, pp. 675-695, 1977.

[35] R. Feynman, *Feynman Lectures on Computation.* Addison-Wesley, 1996.

[36] Z. Kohavi, *Switching and Finite Automata Theory.* New York: McGraw-Hill, 1970.

[37] Z. Manna, *Mathematical Theory of Computation.* New York: McGraw-Hill, 1974.

[38] K. Palem, S. Cheemalavagu, and P. Korkmaz, “The Physical Representation of Probabilistic Bits (PBITs) and the Energy Consumption of Randomized Switching,” CREST technical report, June 2003.

[39] C. Cheemalavagu, P. Korkmaz, and K. Palem, “Ultra Low-Energy Computing via Probabilistic Algorithms and Devices: CMOS Device Primitives and the Energy-Probability Relationship,” *Proc. 2004 Int'l Conf. Solid State Devices and Materials,* Sept. 2004.

[40] K.V. Palem, L.N. Chakrapani, B.E.S. Akgul, P. Korkmaz, and B. Seshasayee, “Ultra Energy-Performance Efficient Embedded SoC Architectures Based on Probabilistic CMOS (PCMOS),” *Int'l Conf. Computers, Architectures, and Synthesis for Embedded Systems (CASES) 2005,* submitted for publication.

[41] A. Sinha and A. Chandrakasan, “Jouletrack: A Web Based Tool for Software Energy Profiling,” *Proc. 38th Conf. Design Automation,* 2001.

[42] M.S. Gupta, “Fluctuations and Dissipation in Elementary One-Bit Information Storage System,” *Int'l J. Theoretical Physics,* vol. 21, nos. 3/4, pp. 275-282, Apr. 1982.

[43] K. Palem, S. Cheemalavagu, and P. Korkmaz, “Introverted Switches as a Basis for Minimizing Leakage in CMOS,” US patent invention disclosure, in preparation.

[44] V. Tiwari, S. Malik, and P. Ashar, “Guarded Evaluation: Pushing Power Management in Logic Synthesis/Design,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 17, Oct. 1998.

[45] A.J. Martin, M. Nystrom, and C.G. Wong, “Tutorial: Asynchronous Microprocessor Design,” *Proc. 35th Int'l Symp. Microarchitecture,* Nov. 2003.

[46] B. Kish, “End of Moore's Law: Thermal (Noise) Death of Integration in Micro and Nano Electronics,” *Physics Letters A,* vol. 305, 2002.

[47] K.V. Palem, S. Cheemalavagu, P. Korkmaz, and B.E.S. Akgul, “Probabilistic and Introverted Switching to Conserve Energy in a Digital System,” US Patent application, 27 Apr. 2005.

Krishna V. Palem holds professorships in electrical and computer engineering and in computer science in the College of Computing, a senior research leadership in the College of Engineering, and is the founding director of the Center for Research in Embedded Systems and Technology (CREST) (www.crest.gatech.edu) at the Georgia Institute of Technology. His earlier work led to the widely used TRIMARAN system (www.trimaran.org), codeveloped with the CAR group of Hewlett Packard Labs and the Impact project of the University of Illinois. The efforts of the ReaCT ILP Laboratory were recognized with awards for excellence from Hewlett Packard, IBM, and Panasonic. Another highlight of the research accomplishments along this dimension is the award-winning dissertation of his PhD advisee, Suren Talla. As part of this research, Palem laid the foundations of *architecture assembly*. The prestigious Analysts' Choice Awards recognized this technology by nominating it as one of the outstanding technologies of 2002. He has chaired bodies whose advice has led to funding initiatives in embedded and hybrid systems in the US, as well as in Singapore. With Guang Gao, he started the *Compilers, Architectures, and Synthesis for Embedded Systems* (CASES) workshop series in 1998. Since then, this workshop has blossomed into an international conference sponsored by ACM SIGs, serving the community as a point of focus for top quality research, driven exclusively by concerns of the embedded computing domain. From 1986 to 1994, he was with the IBM T.J. Watson Research Center. He was a Schonbrunn visiting professor at the Hebrew University of Jerusalem, Israel, where he was recognized for *excellence in teaching*. He is a fellow of the IEEE.