Is “Shannon-capacity of noisy computing” zero?

Pulkit Grover
pulkit@cmu.edu

Abstract
Towards understanding energy requirements for computation with noisy elements, we consider the computation of an arbitrary $k$-input, $k$-output binary invertible function. In our model, not all gates need to be noisy. However, the input nodes on the computation graph must be separated from the output nodes by a noisy cut. For this setup, we show that for the “information-friction” model proposed recently for energy consumed in circuits, and for binary-input AWGN noise in the computational nodes (with fixed communication schedule), the total (gate+ info-friction) energy consumption for reliable computation diverges to infinity as the target error probability is lowered to zero. Thus the capacity of noisy computing in this model is zero regardless of how “rate” of computation is defined: the required energy is unbounded for reliable computation regardless of how efficiently the available resources (e.g. gates, wires, etc.) are used.

I. INTRODUCTION
The success of information theory in advancing reliable communication is in large part due to the surprising result of Shannon that noisy channels can have non-zero Shannon-capacity, i.e., using bounded transmit power, unboundedly low error probabilities can be achieved across a noisy channel while communicating at a fixed communication rate. Many an attempt have been made at emulating its success in noisy computational systems. More recently, computation with noisy elements has had a surge of interest because of three reasons:
1) To save energy by allowing for errors in modern circuits by lowering the circuit voltage, and operating in the “sub-threshold regime.” Energy required by computation is beginning to saturate: the energy requirements no longer reduce simply because the devices are smaller.
2) It is hard to construct reliable circuit components as they get smaller.
3) Biological systems (e.g. neurons) appear to compute in noisily: the spike-train response of a neuron to the same stimulus can vary significantly over trials. This phenomenon has often been interpreted as a reflection of noise in neuronal computation.

However, reliable computation in Shannon sense — that is, with arbitrarily low error probability — is impossible in problems where every gate is erroneous independently with the same probability. This is because of the “last-gate effect”: the error probability is lower bounded by the error probability of the last (i.e., output) gate.

What if one could mix a few noiseless gates with noisy gates? Perhaps motivated by this observation, most of the classical results in noisy computing (e.g. [1]–[5]) have focused on a reliability-goal where only a small output bias towards the correct output is desired (e.g., it is sufficient to compute a boolean function correctly with probability $p = 0.5 + \epsilon$, for some $\epsilon > 0$). If such a bias is there, one can obtain arbitrarily high reliability by using, for instance, noiseless majority-logic gates to aggregate information from biased outputs.

Another body of work explores fundamental limits on “rate” of computation in noisy situations (in silicon or neuronal circuits). The “rate” is defined in various ways, for instance, the more information-theoretically motivated definition of the ratio of number of inputs to the number of gates used, or the more speed-oriented definition of bits processed per second (e.g. [6]–[8]). Often, the perspective is communication-driven: the circuit is viewed as a computational network that succeeds in spews the correct outputs at the end of the computation. This inspires our model with a noisy-cut, detailed in the following paragraph.

In this paper, we provide fundamental limits on energy requirements a circuit-inspired model of noisy computing where (as in [1]–[4]) noiseless gates are also made available. To capture an important aspect of noisy computing, we impose the constraint that a “noisy cut” must separate the input and the output nodes (see Fig. 1) so that no path is available for noiseless information exchange between input and
A noisy cut separates input nodes from output nodes such that all links across the cut are noisy. Further, our model assumes that energy is consumed in two ways in the circuit. The first is the “gate” or the “node” energy that is examined using an example relation between a node’s energy and its reliability that is an AWGN-inspired exponential function. The second accounts for the energy consumed in moving information using the “information-friction” model recently proposed in [9]. For this circuit-inspired model of implementation and energy consumption (detailed in Section II), we show (in Section III) that the total (node + info-friction) energy must diverge to infinity as the target error probability is lowered, even when noiseless gates are available, regardless of the chosen implementation architecture.

The result is analogous to our work in the classical communication setup (along with a circuit implementation model for encoding/deoding) where we show that the total (transmit + encoding/decoding circuit) energy [9], [10] must diverge to infinity. Intuitively, communication is similar to computing an identity function. However, computation on a circuit allows for more flexibility than our model in [9], [10] does. Restricted to computing the identity function, the contributions of this paper are:

- Allowing for different energy use by different gates, and hence using gates of different reliabilities (in [9], [10], all the channel outputs have the same reliability).
- Allowing for “interactions”: the communication problem addressed in [9], [10] does not allow feedback from the receiver. However, any realistic model of computation must allow information exchange in all directions between different parts of the circuit.

We must acknowledge that ours are not the first negative results on the problem. Elias [11] examined the maximum possible “rate” of reliable computing with noisy elements under restrictive conditions, with “encoding” and “decoding” allowed to be noiseless. He shows that the maximum rate is zero. In contrast with Elias’s results, our results are directly on energy consumption, and apply to a larger class of computations.

It is our view that our results, as negative as they are, are not suggesting that there is little utility of error-correcting codes in noisy circuits, or even of information theoretic ideas to obtain fundamental limits on computation. This is for three reasons. First, our lower bounds on energy are in order-sense lower than the energy required for an “uncoded” computation strategy in our model, i.e., a strategy that uses circuit architectures designed for noiseless computing along with sufficiently high SNR to simulate a noiseless circuit (for a given target error probability). Thus, our results do not rule out the possibility that there is significant room for improvement on today’s circuit implementations by allowing for errors (for instance, by reducing the supply voltage of the circuit and operating in the “sub-threshold regime” of the transistors). Second, the implementation model is constrained: the energy-noise tradeoff is captured using an exponential relation that could be improved on using device-design. Further, our implementation model also does not allow for flexibility in size and order of messages, which could be of benefit in noisy computing. Third, we believe even if the Shannon-capacity of computing is zero in the case of interest, mutual information across the computational network can yield useful limits on the performance.

\[1\] This is because the achievable distortion may not change significantly even with a gap from capacity (especially when the distortion-rate function is smooth).
Nevertheless, our results could serve as a cautionary note on instinctive use of Shannon-capacity obtain insights on computation in circuits, neuronal computation, molecular computation, etc. Shannon-capacity would always yield limits on energy requirements and attainable rates, but insights obtained from these limits can be misleading from a total energy viewpoint. For instance, modeling the neuronal computation system with axons as wires along which the neurons communicate, it appears highly inefficient for neurons to obsess over communicating at the axonal-channel’s capacity if their goal is to minimize the system’s energy with transmission of information at a fixed rate. Experimentally, there are many works that test the input distribution in to a biological “channel” to see if the distribution approaches the input distribution that maximizes the mutual information. Since communication over such channels may be far from capacity, we believe that adopting this approach may not be pertinent in general.

II. NOTATION, AND MODELS OF IMPLEMENTATION, COMPUTATION, NOISE, AND ENERGY

We use asymptotic analysis to obtain an intuitive understanding. We use the family of Bachmann-Landau notation [12] in our asymptotic lower bounds: $f(n) = \Omega(g(n))$ if $f(n) \geq c_1 g(n)$ as $n \to \infty$ and $f(n) = \Theta(g(n))$ if $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for some positive $c_1, c_2$. Vectors are denoted in bold, with the vector length appearing in superscript (e.g. $X^n_i$ is a vector of length $n$). $X_i$ denotes the $i$-th element of the vector $X^n_i$. The computation is performed using a “circuit” on a “substrate.” This section formally defines these terms allowing for analysis in Section III.

**Definition 1 (Substrate):** A Substrate is a square $Sq(l)$ of side $l$ in $\mathbb{R}^2$ with vertices at $(0,0)$, $(0,l)$, $(l,0)$, $(l,l)$.

**Definition 2 (SquareLattice($\lambda$)):** A SquareLattice($\lambda$) is the collection of points $(s\lambda,t\lambda) \in \mathbb{R}^2$ for all $s,t \in \mathbb{Z}$.

**Definition 3 (Grid($\lambda$)):** Grid($\lambda$) is the intersection of SquareLattice($\lambda$) with the substrate $Sq(l)$, that is, it is the set of the lattice-points of the square lattice that lie in the substrate.

The parameter $\lambda$ determines how close computational nodes in the circuit can be brought to each other, and depends on the technology of implementation. For large circuits, $\lambda \ll l$.

**Definition 4 (Circuit, computational nodes):** The substrate $Sq(l)$ together with a collection $S \subset \text{Grid}(\lambda)$ of points (called computational nodes, or simply nodes) inside $Sq(l)$, is called a Circuit, and is denoted by $\text{Ckt} = (Sq(l), S)$.

For instance, $Sq(10\lambda)$ along with the set $S = \{(\lambda, \lambda), (5\lambda, 4\lambda)\}$ constitutes a Circuit.

Nodes can be input nodes, output nodes, or helper nodes. Physically, the nodes help perform the computation by computing functions of received messages. Each node is accompanied with a finite storage memory. Input nodes store the input of computation (one bit each; at the beginning of computation), output nodes store the output (one bit each; at the end of computation), and helper nodes help perform the computation.

**Definition 5 (Subcircuit):** A subcircuit $\text{SubCkt}_1 = (F_1, S_1)$ of a circuit $\text{Ckt} = (Sq(l), S)$ is constituted by an open and convex subset $F_1$ of $Sq(l)$ and by the subset of computational nodes $S_1 = F_1 \cap S$.

That is, all the computational nodes within the sub-substrate $F_1$ must lie in the subcircuit $\text{SubCkt}_1$.

**Definition 6 (Link):** A (unidirectional) link connects two nodes in that it allows for communication of binary strings between nodes in one direction. Each binary string is a function of all the messages (and inputs) received at the transmitting node until the start of the message-transmission.

In a circuit with $n$ nodes, there are $n(n-1)$ links, which can be used more than once during a computation. If multiple bits are to be communicated, the nodes can use a link multiple times.

**Definition 7 (Communication on a circuit):** Computational nodes use messages received thus far in computation, and stored memory values, to generate messages that can be communicated to other nodes over links.

Our model assumes that the messages are of fixed-length and and are communicated in a predetermined and fixed order.

**Definition 8 (Fixed-message-length computation):** The computation starts with the arrival of the input of computation at the input nodes. Each input node stores one bit of the input. The computation then
proceeds with communication of messages of predetermined size and order. Each message is a function of the messages that the transmitting computational node has received thus far in the computation (including one bit of the input if the transmitting node is an input node). At the (pre-determined) end of the computation, the output is available in the memories of the output nodes.

We will now define the associated costs.

**Definition 9 (bit-meters of a link and of a circuit):** The bit-meters cost of a link in a computation $\text{Comp}$ on a circuit $\text{Ckt}$ is the product of the total number of bits carried by the messages on the link and the Euclidean distance between the nodes at the ends of the link. The bit-meters for the entire circuit $\text{Ckt}$ is the sum of bit-meters for all the links in $\text{Comp}$.

Fixing the order of messages makes sure that “silence” [13] cannot be used for communicating messages between nodes.

**Definition 10 (bit-meters for a link in a subcircuit):** For a link that connects two nodes within a subcircuit in a computation $\text{Comp}$, the bit-meters for that link within the subcircuit is the same as the bit-meters for the link in the original circuit. However, if only one of the nodes lies within the subcircuit, then bit-meters for this link in the subcircuit is the product of the number of bits of the message passed along this link and the length of link from the node inside the subcircuit to the boundary of the subcircuit.

**Definition 11 (bit-meters for a subcircuit):** The bit-meters for a subcircuit $\text{SubCkt}_1 = (F_1, S_1)$ in computation $\text{Comp}$ is the sum of bit-meters for all the links within the subcircuit (wholly or partially, as defined in Definition 10), and is denoted by bit-meters($\text{SubCkt}_1$).

The definition also holds for bit-meters for the entire circuit.

**Definition 12 (Coefficient of information-friction ($\mu$)):** The coefficient of information-friction, denoted by $\mu$, characterizes the energy required for computation in our model. This energy is given by $E = \mu \times bm$, where $bm$ is the number of bit-meters expended in executing the given computation on a circuit.

**Definition 13 (Computation graph):** The computation-graph is a directed graph with edges as the links that are used in a computation on a circuit. The direction of an edge is the direction of communication on the link.

Two nodes can be connected with edges in both directions.

**Noise model and the noisy cut:** We assume that there exists a cut on the computation-graph that separates input nodes from output nodes such that all the transmissions across this cut are noisy (see Fig. 1), with a raw bit-error probability given by $p_{\text{node}} = e^{-\gamma E_{\text{node}}}$, where $E_{\text{node}}$ is the energy expended by the transmitting node in computing a bit, and $\gamma$ is a technology-dependent constant. These errors are also assumed to be independent across time and across nodes. We further assume that for each transmission, a node needs to recompute the bit to be transmitted.

The motivation for this model is a binary input AWGN channel. For this channel, operating at high SNR, the bit-error probability (obtained by making hard-decisions at the receiving end) goes down exponentially to zero with increase in energy. This is a fairly good model for errors in circuits that are thermal-noise-limited. A deeper study is needed to understand how the total energy scales in other models.

The goal is to compute the function so that the correct output $b^k_{\text{out}}$ and the obtained noisy output $\hat{b}^k_{\text{out}}$ differ with a maximum error probability of $P_e^{\text{blk}} = \Pr(b^k_{\text{out}} = \hat{b}^k_{\text{out}})$.

**Definition 14 (Noisy Implementation Model ($\lambda, \mu, \gamma$)):** Noisy Implementation Model ($\lambda, \mu, \gamma$) denotes the implementation model as described in this section with $\lambda$ being the minimum distance between computational nodes, and $\mu$ being the coefficient of information-friction, and $\gamma$ being the noise parameter.

### III. LOWER BOUNDS ON ENERGY OF NOISY COMPUTING

In this section, we provide lower bounds on the total (nodes + information-friction) energy of noisy computing of a $k$-input, $k$-output invertible binary function.
A. Preliminary analysis

The paper shares much of its preliminary analysis with [9], [10]. We include it here for completeness. Similar to analysis in [9], [10], [14], [15], we will need to cut the circuit under consideration into many disjoint subcircuits. The following definitions and lemmas set up the technical background needed for circuit-cutting and ensuing analysis.

**Definition 15 (Disjoint subcircuits):** Two subcircuits \( \text{SubCkt}_1 = (F_1, S_1) \) and \( \text{SubCkt}_2 = (F_2, S_2) \) of a circuit \( \text{Ckt} = (S(q(l)), S) \) are said to be disjoint subcircuits if \( F_1 \cap F_2 = \phi \), the null set. Similarly, \( \{\text{SubCkt}_i\}_{i=1}^{N_{\text{subckt}}} \) are said to be mutually disjoint subcircuits if \( F_i \cap F_j = \phi \) for every \( i, j \in \{1, 2, \ldots, N_{\text{subckt}}\}, i \neq j \).

It follows that any two disjoint subcircuits cannot share computational nodes or communication links that connect two nodes within one of the subcircuits. In fact, two disjoint subcircuits do not “share” bit-meters of computation:

**Lemma 1 ([9], [10]):** Let \( \{\text{SubCkt}_i\}_{i=1}^{N_{\text{subckt}}} \) be a set of mutually disjoint subcircuits of the circuit \( \text{Ckt} = (S(q(l)), S) \). Then for any computation \( \text{Comp} \),

\[
\text{bit-meters}(\text{Ckt}) \geq \sum_{i=1}^{N_{\text{subckt}}} \text{bit-meters}(\text{SubCkt}_i). \tag{1}
\]

**Proof:** See [9], [10]. The lemma follows from the observation that in Definition 9, no bit-meters are double-counted in disjoint subcircuits. We note that there are potential situations when \( \bigcup_{i=1}^{N_{\text{subckt}}} F_i = S(q(l)) \) for which (1) is not satisfied with equality. This happens when there is a long link in a circuit which has a part that does not lie within either of the subcircuits that contain the two nodes at the ends of the link.

The circuit is partitioned into multiple subcircuits via a “Stencil” that can be “moved” over the circuit by changing its origin.

**Definition 16 (Stencil):** A Stencil \((a, \eta, O)\) in \( \mathbb{R}^2 \), for \( \eta < \frac{1}{2} \), is a pattern of equally spaced “inner” squares that are concentric with “outer” squares which form a grid (as shown in Fig. 2). The length of a side of each outer square is \( a \), and the origin \( O \) lies in the center of an “inner” square. The side of each inner square is of length \( s = (1 - 2\eta)a \).

We use the term “Stencil” in analogy with the classic stencil instrument used to produce letters or designs on an underlying surface. A stencil can be slid on the surface to produce the design at any location on the surface, effectively shifting the origin-point of the design. In this case, a pattern of inner and outer squares is produced on the computational substrate.

A node in a circuit is said to be covered by a Stencil that is overlaid on the circuit substrate if it lies inside an inner-square of the Stencil. Inside the \( i \)-th subcircuit, let \( k_i^{in} \) denote the number of output nodes that lie inside the inner square, and \( k_i \) denote the number of output nodes that lie inside the outer square (i.e., anywhere inside the \( i \)-th subcircuit).

**Definition 17 (Stencil-partition):** The outer squares of Stencil \((a, \eta, O)\) induce a partition (see Fig. 2) of a circuit into subcircuits, each occupying substrate area at most \( a^2 \). If any computational node lies on the boundary of an outer square, then it is arbitrarily included in one of the subcircuits.

The next lemma (from [9], [10]) shows that by moving the Stencil over the substrate, we can find at least one position of the Stencil so that the average number of nodes (over random locations of the Stencil) are covered.

**Lemma 2 ([9], [10]):** For any circuit implemented in Implementation Model \((\lambda, \mu, \gamma)\), for any \( \eta > 0 \), there exists an origin \( O \) of Stencil \((a, \eta, O)\) such that the number of output nodes covered by the Stencil is lower bounded as follows:

\[
k_i^{in} := \sum_i k_i^{in} \geq k(1 - 2\eta)^2. \tag{2}
\]

**Lemma 3 ([9], [10]):** For the Noisy Implementation Model \((\lambda, \mu, \gamma)\), for Stencil-partition with outer-squares of side-length \( a \), if \( \frac{a^2}{N} \geq 25 \), the maximum number of computational nodes in a subcircuit is
Consider the Stencil shown in Fig. 2. The distance between the inner and the outer squares is $\eta a$. $B$ bits are said to be communicated from the “transmitting” part of the circuit to the “receiving part” if the values stored in the receiving part are independent of the $B$ bits prior to communication, and the bits can be recovered (in an error-free manner) from the messages received at the receiving part during the process of communication. Notice that this definition is looser than the traditional understanding of communication: we do not stipulate that the stored values at the receiving part post-communication be able to recover the $B$ bits.

The following lemma captures the cost of communicating $B$ bits from outside an outer square to inside an inner square in a subcircuit.

**Lemma 4** (bit-meters in computations [9], [10]): Consider a circuit implemented in Noisy Implementation Model ($\lambda, \mu, \gamma$), and any subcircuit $\text{SubCkt}$ obtained using the Stencil-partition defined in Definition 17. For communicating $B$ bits of information from outside an outer-square to inside the corresponding inner-square, bit-meters$(\text{SubCkt}) \geq \eta a B$.

The following lemma follows in a straightforward manner from Fano’s inequality:

**Lemma 5** ([9], [10]): If at most $\frac{r}{3}$ bits of information is available to obtain an estimate $\hat{M}$ of a variable $M$ that is distributed uniformly on the set $M := \{1, 2, \ldots, 2^r\}$, $r$ being a positive integer, then $\text{Pr}(\hat{M} \neq M) \geq \frac{1}{9}$.

**B. Main result**

The next theorem is our main result. It proves that within our implementation and energy model of Section II, the total *per-output-bit* energy for computing an invertible $k$-input, $k$-output binary function must diverge to infinity as the error-probability of computation is lowered to zero, regardless of what implementation architecture is chosen.

The intuition for the proof as follows: intuitively, an error-correction strategy for noisy computing would attain robustness against noise by relying on the laws of large numbers. That is, if a large number of noisy nodes are used, the probability that a significant fraction of them are in error is small. However, such error-correcting mechanisms work by “diffusing” available information redundantly across the available space, and aggregating diffused information to produce reliable outputs would require some circuit-communication, which is what we lower bound.

More formally, to capture this communication-cost at the time of aggregation, we (conceptually) split the circuit using a stencil with parameter $a$ chosen to ensure that an “average” subcircuit receives only a fraction of the total node-energy. The probability that this information is highly corrupted can now be high. When such local error-events happen, the information supplied by other parts of the circuit
can be used for correction. After all, if each part operated on its own information, then we are really not exploiting the benefits of the laws of large numbers. But this information obtained from rest of the circuit needs to be paid with information-frictional energy.

**Theorem 1:** The total energy per-bit of computing a $k$-input, $k$-output invertible binary function in the Noisy Implementation Model ($\lambda, \mu, \gamma$) with an error probability of $P_{blk}^{\text{e}}$ is lower bounded$^2$ by:

$$E_{\text{total per-bit}} = \Omega \left( \sqrt[3]{\log \frac{1}{P_{blk}^{\text{e}}}} \right),$$

in the limit of $P_{blk}^{\text{e}} \to 0$, where $P_{blk}^{\text{e}}$ is the probability with which the entire transform is computed correctly. The result holds for all values of $k$.

**Proof:** Let $E_{\text{nodes}}$ be the total energy consumed by the noisy nodes in the computation. Use a Stencil($a, \eta, O$) of parameters $a = \frac{1}{4} \sqrt{\frac{\log \frac{1}{\gamma \cdot E_{\text{nodes}}^{\text{per-bit}}}}{\gamma \cdot E_{\text{nodes}}^{\text{per-bit}}}} \lambda$ where $E_{\text{nodes}} := \frac{E_{\text{nodes}}^{\text{total}}}{k}$ and $\eta = \frac{1}{4}$, and an origin $O$ so that $k_{\text{in}} := \sum_{i=1}^{n} k_{i, \text{in}} \geq (1 - 2\eta)^2 k = \frac{k}{4}$ (see Lemma 2).

**Case 1:** $\frac{a^2}{\lambda^2} < 25$. This implies that

$$\frac{1}{16} \frac{\log \frac{1}{\gamma \cdot E_{\text{nodes}}^{\text{per-bit}}}}{\gamma \cdot E_{\text{nodes}}^{\text{per-bit}}} < 25,$$

and thus $E_{\text{nodes}} = \Omega \left( \log \frac{1}{\gamma \cdot E_{\text{nodes}}^{\text{per-bit}}} \right)$. Since $E_{\text{nodes}}^{\text{per-bit}}$ is the per-bit node-energy, it is a lower bound on total per-bit energy which (in this case) is larger than the lower bound in (4), and thus (4) is satisfied.

**Case 2:** $\frac{a^2}{\lambda^2} \geq 25$

How many bit-meters are needed? To understand this, we first need to understand just how many bits must be communicated from outside the subcircuit to inside the inner square.

First let us consider the outputs of the noisy-cut that are received inside the $i$-th subcircuit (some of the nodes that generate the inputs to the links across the noisy cut might themselves be in the $i$-th subcircuit). Let the total energy corresponding to these outputs be denoted by $E_i = \sum_{j=1}^{n_i} E_{ij}$ where $E_{ij}$ is the energy expended by the node that sends the $j$-th message to the $i$-th subcircuit. Observe that there is no limitation on the number of these messages, but there is an energy-limitation: only $E_i$ amount of node-energy is available.

We use the observation that a BSC($p$) is a degraded version of a BEC(2$p$) erasure channel (see [9], [10]), and supply nodes that receive a BSC($p_{ij}$) output with a BEC(2$p_{ij}$) output instead. This information will only further reduce the lower bound as these nodes can easily simulate a BSC($p_{ij}$) by randomly generating a 0, 1 value (with probability 0.5) for an erased output.

Under this relaxation, the probability that all the bits transmitted across the noisy cut to the $i$-th subcircuit get erased is lower bounded by

$$\Pr(\text{all transmissions in the } i\text{-th subcircuit erased}) \geq \prod_{i=1}^{n} (2p_{ij}) > \prod_{i=1}^{n} e^{-\gamma E_{ij}} = e^{-\gamma E_i},$$

where $p_{ij} := e^{-\gamma E_{ij}}$ is the error-probability of the $(i, j)$-th transmission, and $E_i = \sum_{j=1}^{n_i} E_{ij}$ is the energy of the node-transmissions received by the $i$-th subcircuit. In this erasure event, the subcircuit needs to rely on information obtained from outside the subcircuit to compute the $k_{i, \text{in}}^{\text{output}}$ output bits. These bits can take all $2^{k_{i, \text{in}}^{\text{output}}}$ values because of the invertibility of the function.

Let $Err := \{ i : \text{bit-meters}_i \leq \frac{k_{i, \text{in}}^{\text{output}}}{12} a \}$. Assuming $k_{i, \text{in}}^{\text{output}} \geq 1$, if the number of bits received from outside the $i$-th subcircuit to inside the inner square of the subcircuit is smaller than $\frac{k_{i, \text{in}}^{\text{output}}}{3}$, and $k_{i, \text{in}}^{\text{output}} \geq 1$, then the

$^2$As a reminder, the symbol $\Omega$ denotes an order-sense lower bound.
error probability for the $i$-th subcircuit ($i \in \text{Err}$), $P_{e,i}$, is lower bounded by (using Lemma 5 and (6)):

$$P_{e,i} \geq \frac{1}{9} e^{-\gamma E_i}. \quad (7)$$

Now, assume that bit-meters $< \frac{k}{384} \sqrt{\frac{\log \frac{1}{10P_{blk}^a}}{\gamma E_{node}}} \lambda$, i.e.,

$$\text{bit-meters(Ckt)} < \frac{k}{384} \sqrt{\frac{\log \frac{1}{10P_{blk}^a}}{\gamma E_{node}}} \lambda = \frac{1}{24} k \eta a = \frac{1}{96} k a$$

for $\eta = \frac{1}{3}$. Under this assumption, we first claim (and prove via contradiction) that at least $\frac{k_{in}}{2}$ output-nodes (where $k_{in} = \sum_i k_{i}^{in}$) lie in subcircuits with indices in $Err$. Suppose our claim is not correct. Then for at least $\frac{k_{in}}{2}$ bits, the subcircuits they lie in have bit-meters $\geq \frac{k_{in}}{12} a$, and thus,

$$\text{bit-meters(Ckt)} \geq \sum_{i \in \text{Err}} \text{bit-meters}_i \geq \sum_{i \in \text{Err}} \frac{k_{i}^{in} a}{12} = \frac{a}{12} \sum_{i \in \text{Err}} k_{i}^{in} > \frac{a}{12} \times \frac{k_{in}}{2},$$

i.e., the total number of bit-meters is larger than $\frac{k_{in}}{24} a$, leading to a contradiction with (8). Thus at least $\frac{k_{in}}{2}$ output-nodes lie in subcircuits with bit-meters $\leq \frac{k_{in}}{12} a = \frac{k_{in}}{3} \eta a$.

Now notice that the information inside every subcircuit needs to be recovered in order to recover the entire block. Thus the error probability is lower bounded by average error probability across subcircuits in $Err$, i.e.,

$$P_{e}^{blk} \geq \frac{1}{|Err|} \sum_{i=1}^{|Err|} P_{e,i} \geq \left( \prod_{i=1}^{|Err|} P_{e,i} \right)^{1/|Err|} \geq \left( \prod_{i=1}^{|Err|} \frac{1}{9} e^{-\gamma E_i} \right)^{1/|Err|} \geq \frac{1}{9} e^{-\gamma E_{nodes}/|Err|}, \quad (9)$$

where (a) uses arithmetic mean $\geq$ geometric mean; and (b) uses $\sum_{i \in \text{Err}} E_i \leq \sum_i E_i = E_{nodes}$. Because at least $\frac{k_{in}}{2}$ number of bits lie in subcircuits with indices in $Err$, and because $k_{i}^{in} \leq 2a^2 \leq \frac{1}{8} \log \frac{1}{10P_{blk}^a}$ (from Lemma 3, and observing that in this case, $\frac{a^2}{\lambda^2} \geq 25$), the cardinality of the set $Err$ is lower bounded as

$$|Err| \geq \frac{k_{in}}{2} \geq \frac{\log \frac{1}{10P_{blk}^a}}{\gamma E_{nodes}} \geq \frac{k \gamma E_{nodes}}{\log \frac{1}{10P_{blk}^a}} = \frac{\gamma E_{nodes}}{\log \frac{1}{10P_{blk}^a}}, \quad (10)$$

where (a) uses $k_{in} \geq \frac{k}{4}$, and (b) uses $kE_{nodes} = E_{nodes}$. From (9),

$$P_{e}^{blk} \geq \frac{1}{9} e^{-\gamma E_{nodes}/|Err|} \geq \frac{1}{9} e^{-\gamma E_{nodes}/\gamma E_{nodes}} = 10 P_{e}^{blk} \quad (11)$$

a contradiction.

Thus the assumption in (8) is incorrect, and the total number of bit-meters is lower bounded as

$$\text{bit-meters(Ckt)} \geq \frac{k}{384} \sqrt{\frac{\log \frac{1}{10P_{blk}^a}}{\gamma E_{node}}} \lambda, \quad (12)$$
**Total energy:** The total energy per-bit is lower bounded by:

\[
\frac{E_{\text{total}}}{k} \geq \frac{E_{\text{nodes}} + E_{\text{info-friction}}}{k} \geq E_{\text{nodes}} + \mu \frac{\lambda}{384} \sqrt{\log \frac{1}{p_{\text{err}}}}.
\]

Minimizing over the possible values of \(E_{\text{nodes}}\), the minimizing value of \(E_{\text{nodes}}\) is (in order sense), \(\left(3\sqrt{\log \frac{1}{p_{\text{err}}}}\right)^{\frac{1}{3}}\), and thus \(E_{\text{total-per-bit}} = \Omega\left(\frac{3}{\sqrt{\log \frac{1}{p_{\text{err}}}}}\right)\).

**IV. CONCLUSIONS AND DISCUSSIONS**

By accounting for energy required to aggregate information in a noisy computational system, we obtain lower bounds on total (nodes + information-friction) energy for computing that diverge to infinity as the target error probability is lowered. These bounds hold for any circuit architecture implemented in our implementation model. In deriving these bounds, we assume that the order and size of the messages passed in the circuit is fixed and known in advance. This model thus does not allow full exploitation of circuit flexibility in presence of noise. More detailed studies, including allowing more flexibility in the model of computation and obtaining upper bounds (achievability), are needed in order to fully understand the potential energy-savings of allowing noise in computing.

**ACKNOWLEDGMENTS**

We acknowledge the generous support of NSF-ECCS-1343324, seed grants from the NSF Center for Science of Information NSF-CCF-0939370, and a startup grant from Carnegie Mellon University. We also thank Andrew Singer and Naresh Shanbhag for motivating discussions.

**REFERENCES**