Bayesian Inference With Muller C-Elements

Joseph S Friedman, Laurie E Calvet, Pierre Bessière, Jacques Droulez, Damien Querlioz

Bayesian Inference With Muller C-Elements

Joseph S. Friedman, *Member, IEEE*, Laurie E. Calvet, *Member, IEEE*, Pierre Bessière, Jacques Droulez, and Damien Querlioz, *Member, IEEE*

***Abstract*—Bayesian inference is a powerful approach for integrating independent conflicting information for decision-making. Though an important component of robotic, biological, and other sensory-motors systems, general-purpose computers perform Bayesian inference with limited efficiency. Here we show that Bayesian inference can be efficiently performed with stochastic signals, which are particularly adapted to novel low power nanodevices that exhibit faults and device variations. A simple Muller C-element directly implements Bayes' rule. Complex inferences are performed by C-element trees, which compute the probability of an event based on multiple independent sources of evidence. A naïve Bayesian spam filter circuit is demonstrated as a pedagogical application, and design techniques for improving circuit functionality are described. Limitations of this structure are discussed in terms of signal autocorrelation. The stochastic inference structure is exceptionally robust to faults, an essential feature of decision circuits, and can therefore leverage the increased efficiency of emerging nanodevices. This hardware implementation of Bayesian inference is extremely area and power efficient, with an area-energy-delay product several orders of magnitude less than the conventional floating point implementation. These results open a pathway for a direct stochastic hardware implementation of Bayesian inference, enabling a new class of embedded decision circuits for robotics and medical applications.**

***Index Terms*—Fault-tolerant circuit design, Muller C-element, nanotechnology, stochastic computing, variation-prone devices.**

I. Introduction

BAYESIAN inference permits decision-making with maximal information integration from distinct inputs [1]. It computes the probability of an event through the incorporation of information from numerous sources. Bayesian inference has been suggested as a fundamental component of biological systems [2]–[4], and has been successfully applied to robotics and other sensory-motor systems [5].

Conventionally, Bayesian inference is performed by general-purpose processors in which the operations on probability values are executed by floating point arithmetic logic units. These systems provide high precision, but consume significant energy and area. In addition to the substantial resources required by floating point operations, conventional approaches do not make efficient use of the high parallelism of Bayesian inference [6]. Furthermore, conventional systems require exact values throughout the computation, impeding the exploitation of variation and error-prone emerging nanodevices that can compute with reduced power consumption [7]. It is therefore highly desirable to devise a specialized hardware structure to directly compute Bayesian inference that is robust to the imperfections characteristic of novel nanotechnology.

Several distinct approaches have been proposed for a specialized Bayesian hardware, including fully digital systems with dedicated architectures [8], [9], and analog computation of probabilistic inference [10], [11]. Here we propose the stochastic computation of Bayesian inference, with probabilistic signals encoded as clocked bitstreams. Stochastic computing permits complex computations with minimal hardware, but its inherent imprecision and correlations have impeded its development.

We show that stochastic Bayesian inference can be performed directly with Muller C-elements, using surprisingly simple circuits. The increase in precision with run-time inherent to stochastic computation [12], [13] is particularly well-suited to decision circuits. The C-element circuit performs useful inferences despite the imprecision and correlations of stochastic computing and process variations of scaled nanodevices [14]–[18]. Furthermore, the fault tolerant nature of stochastic computing permits the exploitation of the extreme low energy, high speed, and small size of emerging nanotechnology that can serve as low-area, low-power computing [19] and random number generation units [20], [21].

In this work, we thus leverage stochastic computing to perform Bayesian inference with a simple circuit structure. We use the spam-detection problem as an instructive example to analyze circuit performance, as it is a common application of Bayesian inference in commercial products [6]. We explore limitations to the inference circuit structure, and discuss the mitigation of these concerns. Finally, we highlight that in comparison to the conventional computation of Bayesian inference, the proposed circuit structure can achieve sufficient precision with orders-of-magnitude increases in performance and energy efficiency.

II. Stochastic Bitstreams

Stochastic computing uses traditional logic gates in a unique manner to perform operations on clocked binary stochastic single-bit streams rather than standard multi-bit binary signals.

Manuscript received July 17, 2015; revised December 2, 2015; accepted February 25, 2016. This work was supported by the FP7 ICT BAMBI (FP7-ICT-2013-C) project and a public grant overseen by the French National Research Agency (ANR) as part of the "Investissements d'Avenir" program (Labex NanoSaclay, reference: ANR-10-LABX-0035). This paper was recommended by Associate Editor I. Belykh.

J. S. Friedman, L. E. Calvet, and D. Querlioz are with the Institut d'Electronique Fondamentale, Université Paris-Sud, CNRS, 91405 Orsay, France (e-mail joseph.friedman@u-psud.fr).

P. Bessière and J. Droulez are with the Institut des Systèmes Intelligents et de Robotique, Université Pierre et Marie Curie, CNRS, 75005 Paris, France.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2016.2546064

This allows for the compact implementation of many complex operations—for example, the product of probabilities is computed by a simple AND gate [13]. Stochastic computing is therefore an attractive alternative to CMOS for inherently probabilistic computation, and its data encoding makes it particularly effective for exploiting error-prone nanodevices.

A. Encoded Probability

In a randomly generated stochastic bitstream, the dynamics of a $0 -> 1$ $(1 -> 0)$ switch can be defined by a state switching probability a (b). The probability R encoded by a stochastic bitstream is

$$R = \frac{a}{a+b} \tag{1}$$

which, for an uncorrelated bitstream (i.e., $a + b = 1$), is equivalent to

$$R = a. \tag{2}$$

Each bit in this bitstream thus has a probability R of being "1," and a probability $1 - R$ of being "0." Throughout this work, the percentage of "1"s in a bitstream encodes its probability.

B. Switching Rate

A particular probability can be represented by several distinct bitstreams that all encode the same value. For example, both bitstreams "0101010101" and "0000011111" encode the value 0.5, but the second bitstream contains longer "domains" of consecutive "0"s and "1"s. This first bitstream therefore allows for an evaluation of the value with significantly fewer bits of information. For this contrived example, the value encoded by the first N bits of the bitstream is closer to 0.5 for the first bitstream than the second bitstream, for any value of N. As will be seen in the following sections, this autocorrelation has profound implications for Bayesian inference.

It should be noted that the switching rate of a stochastic bitstream is significantly higher for signals with probabilities close to 0.5 than for extreme probabilities close to 0 or 1. For an uncorrelated stochastic bitstream with value R, the switching rate is given by

$$S = 2R(1 - R) = 2a(1 - a). \tag{3}$$

C. Precision

The probabilistic nature of stochastic bitstreams implies that there is no guarantee that the bitstream encodes the precisely correct value. In a properly operating circuit, however, the probability encoded by the bitstream converges toward the correct value as its length increases. The variance of an uncorrelated stochastic bitstream of length N is

$$\mathrm{Var} = \frac{R(1 - R)}{N}. \tag{4}$$

The expected variance of a bitstream thus decreases with $1/N$, with smaller values for extreme probabilities. The strong effect of probability R on variance creates a bias that distorts the precision analysis, thereby making variance a poor metric of bitstream precision. A more appropriate metric for the error of a probabilistic signal is the Kullback-Leibler (KL) divergence

$$KL = (1 - R) \ln \frac{1 - R}{1 - Q} + R \ln \frac{R}{Q} \tag{5}$$

which measures the information lost when a probability Q is used to approximate a probability R. The KL divergence is designed to measure error independently of R and, interestingly, the bitstream divergence approaches $1/2N$ for increasing N.

X	Y	Z
0	0	0
0	1	Z_{prev}
1	0	Z_{prev}
1	1	1

(a)

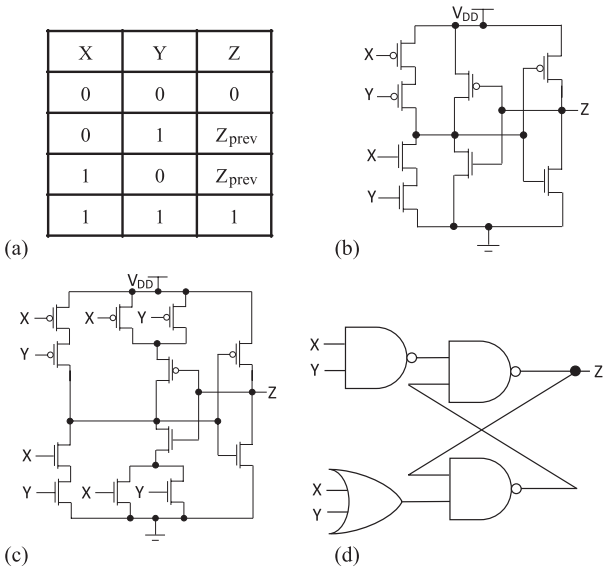

Fig. 1. (a) Muller C-element truth table. (b) Compact C-element with weak latch. (c) Efficient one-stage C-element. (d) Standard cell C-element design.

III. Two-Input C-Element Inference

To obtain the probability of a particular event V, Bayesian inference incorporates the probability of V given the prior $P(V)$ and evidence input E_1 as

$$P(V|E_1) = \frac{P(E_1|V)P(V)}{P(E_1|V)P(V) + P(E_1|\overline{V})P(\overline{V})} \tag{6}$$

which may be rewritten

$$P(V|E_1) = \frac{P^*(E_1)P(V)}{P^*(E_1)P(V) + (1 - P^*(E_1))\,(1 - P(V))} \tag{7}$$

with parameter $P^*(E_1)$ defined by

$$P^*(E_1) \equiv \frac{P(E_1|V)}{P(E_1|V) + P(E_1|\overline{V})}. \tag{8}$$

We now show that (7) is computed by a Muller C-element, enabling efficient inference circuits.

A. Muller C-Element

The Muller C-element, characterized by the truth table of Fig. 1(a), is a two-input memory element [22]. The output Z maintains its state Z_{prev} unless both inputs X and Y are opposite the current output state, in which case the output switches

to the shared input value. Several circuits perform this function with eight to twelve transistors, as shown in Fig. 1(b)–(d), each with its own tradeoffs [23]. Individual C-elements have recently found use in stochastic computing, particularly for error correction applications [14], [15], [24].

B. Stochastic C-Element Inference

For C-element input signals i with no autocorrelation ($a_i + b_i = 1$ for switching probabilities a_i and b_i), the output probability is

$$P(Z) = \frac{P(X)P(Y)}{P(X)P(Y) + (1 - P(X))\,(1 - P(Y))}. \quad (9)$$

Substituting $P^*(E_1)$ for $P(X)$, $P(V)$ for $P(Y)$, and $P(V|E_1)$ for $P(Z)$, (9) is equivalent to (7), demonstrating the Bayesian inference performed by C-elements. This equivalence motivates the implementation of Bayesian inference with stochastic computing, and shows that *a simple C-element performs the complete inference of Bayes' rule*.

C. C-Element Switching

Due to the inertia inherent to C-elements, the output switches equally often or or less frequently than the inputs. In contrast to a randomly generated bitstream, the switching rate of the C-element output bitstream is not uniquely determined by the probability it encodes, but is dependent on the combination of input probabilities. The upper bound of the output switching rate is the maximum switching rate of the two inputs. In particular, for uncorrelated bitstreams with opposite values ($a_i + b_i = 1,\ R_X + R_Y = 1$) and therefore equal switching rates, the expected switching rate of the output is one half of the input switching rate. Furthermore, the expected output switching rate when one input signal carries the value 0.5 is one half of the switching rate of the other input. The decreased switching causes a memory effect that results in autocorrelation- that is, the state of each bit is correlated to the states of other bits in the bitstream.

This autocorrelation causes a decrease in the amount of precision provided by a bitstream. The stochastic bitstream output of the C-element is also probabilistic and imprecise, but converges more slowly toward the precise Bayesian inference. In particular, the lower the switching rate of the output, the longer the "domains" of consecutive "0"s and "1"s, and the more imprecise the bitstream. Therefore, as can be seen in the simulation results shown in Fig. 2, the error is significantly increased for extreme inputs with low switching rates. The output signals polynomially converge to the exact Bayesian inference as N increases, but have consistently larger errors for opposite extreme input combinations. As these extreme opposite inputs can be considered as strong conflicting evidence, this error can be understood as inference uncertainty.

IV. Multi-Input Cascaded C-Element Inference

A. Cascaded C-Element Tree

The integration of additional inputs provides the capability for more complex inferences. Incorporating E_2 into the inference gives the probability of event V:

$$P(V|E_1,E_2) = \frac{P(V|E_1)P(E_2|V,E_1)}{P(V|E_1)P(E_2|V,E_1) + (1 - P(V|E_1))\,P(E_2|\overline{V},E_1)}. \quad (10)$$

Under the naïve assumption that evidence inputs E_1 and E_2 are conditionally independent, $P(E_2|V,E_1) = P(E_2|V)$, and (10) reduces to

$$P(V|E_1,E_2) = \frac{P(V|E_1)P^*(E_2)}{P(V|E_1)P^*(E_2) + (1 - P(V|E_1))\,(1 - P^*(E_2))}. \quad (11)$$

This can in theory be computed using two cascaded C-elements, as illustrated in Fig. 3(a).

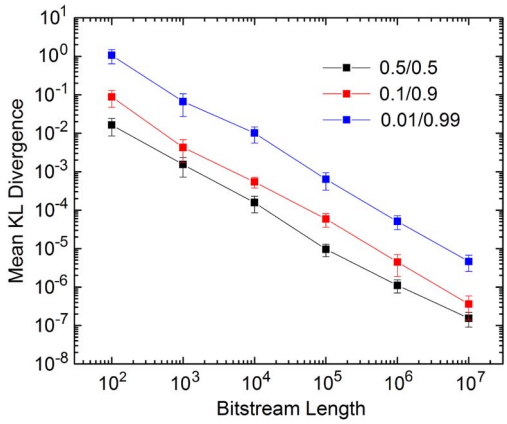

Fig. 2. Mean simulated precision (KL divergence from exact value) of C-element inference for various input signal combinations as a function of observed bitstream length. Error bars depict 95% confidence interval.

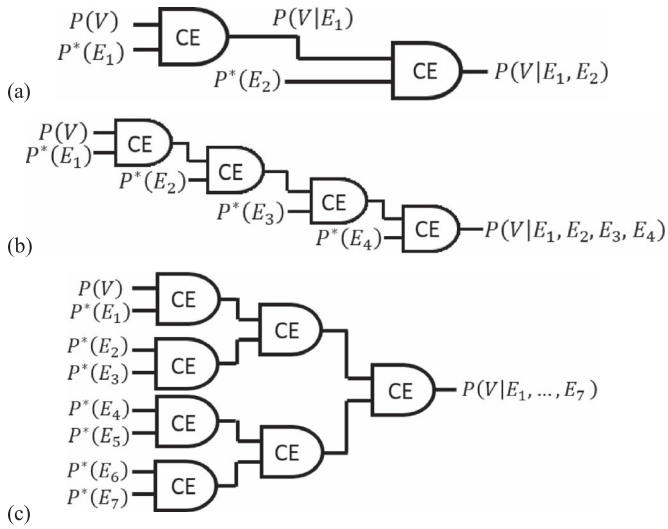

Fig. 3. Cascaded C-element Bayesian inference circuits that integrate a prior and (a) two inputs; (b) four evidence inputs in a sequential structure; (c) seven inputs in a tree structure.

TABLE I
SAMPLE MESSAGES FOR SPAM-DETECTION

Message	Message Text
A	Do **you** want to get **pizza** for lunch?
B	**You** should **check** out these **stochastic** simulations.
C	If **you** want to earn a **fortune**, send a $100 **check** to **Nigeria** and we will **transfer** $10,000 to your account.
D	My **commute** to **Nigeria** includes a **transfer** in Morocco. I will **check** if my flight is on schedule- if so, do **you** want to get **pizza** when I arrive?
E	There is a $10 fee for all **check transfers**.
F	**You!**

This inference can be extended to multiple independent evidence inputs $E_1, E_2, \ldots, E_i$ to give (12), shown at the bottom of the page. This multi-input inference can be approximated with cascaded C-elements [Fig. 3(b)]. The final output is expected to be

$$P\left(V|\{E_1,\ldots,E_i\}\right) = \frac{P(V)\prod_{x=1}^{i}P^*(E_x)}{P(V)\prod_{x=1}^{i}P^*(E_x)+(1-P(V))\prod_{x=1}^{i}(1-P^*(E_x))}. \quad (13)$$

Straightforward arithmetic shows that the C-elements can equivalently be organized in trees as in Fig. 3(c).

B. Example: Bayesian Spam Filter Circuit

To illustrate the feasibility of multi-input Bayesian inference with cascaded C-elements, we evaluate its use for spam filtering [6], a canonical Bayesian inference. Final applications of our proposal will target situations in which decisions are automated, such as brain-machine interfaces and robotics, and for which an efficient direct hardware implementation would fuel tremendous advances.

We consider the messages in Table I: A and B are likely to be ham (i.e., not spam), C is almost definitely spam, D seems to be ham but contains many words that are associated with spam, E is probably spam, and F is entirely inconclusive. To infer the probability that a message is spam, a simple Bayesian spam filter considers the "spamicity" $P^*(W_x^{m_x})$ of each word W_x given its presence $(m_x = 1)$ or absence $(m_x = 0)$. Statistics are provided in Appendix A. The probability that a message is spam given the presence or absence of various keywords is computed in accordance with Section IV-A. Assuming no prior information $(P(S) = 0.5)$ the posterior probability reads

$$P\left(S|\{W_1^{m_1},\ldots,W_i^{m_i}\}\right) = \frac{\prod_{x=1}^{i}P^*\left(W_x^{m_x}\right)}{\prod_{x=1}^{i}P^*\left(W_x^{m_x}\right)+\prod_{x=1}^{i}\left(1-P^*\left(W_x^{m_x}\right)\right)}. \quad (14)$$

The three-stage C-element tree of Fig. 3 can be used to perform the spam filter operation on messages A-F for eight key words. The presence or absence of each word provides a particular probability that a message is spam. These $P^*(W_x^{m_x})$ values are encoded as stochastic bitstreams and are used as inputs to the first stage of the C-element tree.

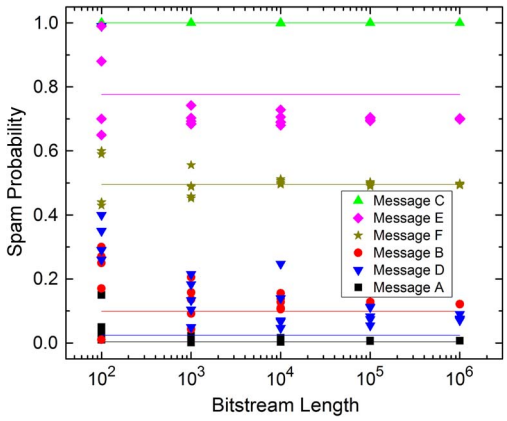

Fig. 4. Symbols: Spam probability value for messages A-E of Table I obtained by cascaded C-elements for several simulation runs as a function of observed bitstream length. Horizontal lines: exact inference. The output of the C-elments converges toward an approximated Bayesian inference as bitstream length increases.

Simulation results for several runs of the spam filter are shown as points in the plot of Fig. 4, with the exact naïve Bayesian inference shown as a solid line. This simple Bayesian system successfully filters the messages, with high outputs for message C and E, low outputs for messages A, B, and D, and an ambiguous output for F. These decisions can be reached reliably by observing bitstreams of 1000 bits. Despite the inertia and autocorrelation, these simulation results show that the inference performed by individual C-elements can be extended to multi-input Bayesian inference with cascaded C-elements.

As can be seen in the figure, there is significant variation of the output values for short bitstreams. As the bitstream length increases, the outputs converge toward values that are quite close to the exact Bayesian inference. However, the output is not perfectly exact, due to the inertia resulting from the autocorrelated domains. The largest discrepancy is for message E: the output converges to a value of 0.7, while an exact inferences gives a value of 0.78. It should further be noted that message D converges slower than other messages, which is a result of the extreme values at its input. We now discuss the origin of this non-ideal behavior.

C. Cascaded C-Element Switching & Inertia

Beyond the first stage of the tree, the inputs to C-elements are autocorrelated; that is, $a_i + b_i \neq 1$. More specifically, following the reasoning of Section III-C, $a_i + b_i \leq 2^{-n}$, where n is the number of C-elements through which information has passed since the original stimulus. The C-elements can therefore no longer be considered as isolated black boxes, and

$$P\left(V|\{E_1,\ldots,E_i\}\right) = \frac{P\left(V|\{E_1,\ldots,E_{i-1}\}\right)P^*(E_i)}{P\left(V|\{E_1,\ldots,E_{i-1}\}\right)P^*(E_i)+\left(1-P\left(V|\{E_1,\ldots,E_i\}\right)\right)\left(1-P^*(E_i)\right)} \quad (12)$$

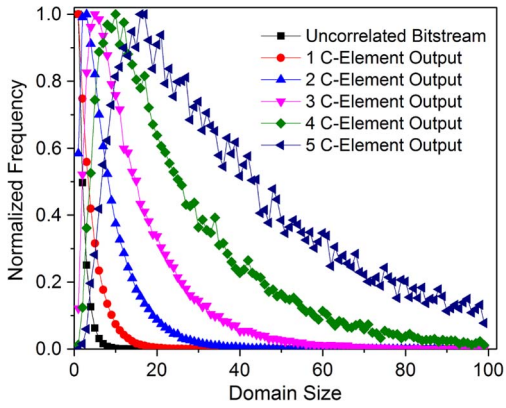

Fig. 5. Normalized distribution of bitstream domain sizes for trees of various depths and input values of 0.5. The domain sizes of the output bitstreams increase with the number of C-element stages.

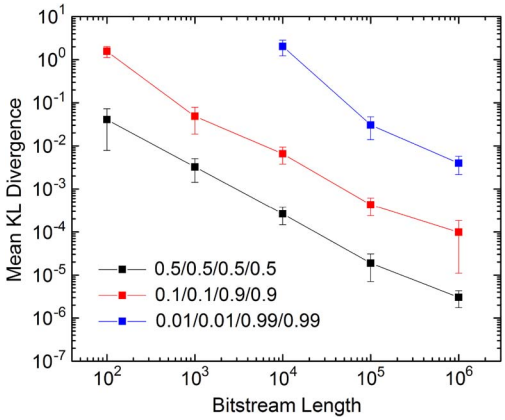

Fig. 6. Mean simulated precision of a two-stage C-element inference circuit, for various input signal combinations as a function of observed bitstream length. Error bars depict 95% confidence interval.

the autocorrelation between consecutive bits B^t and B^{t+1} of an output bitstream B can be described in terms of the mutual information I as

$$I(B) = 1 + (1 - S)\log_2(1 - S) + S\log_2(S) \qquad (15)$$

for a bitstream encoding a value 0.5 with switching rate S defined as

$$S = P(B^t = 1 \wedge B^{t+1} = 0) + P(B^t = 0 \wedge B^{t+1} = 1). \qquad (16)$$

As signals propagate through successive stages, the inertial nature of the C-elements induces strings of consecutive "1"s and "0"s. As mentioned previously, these "domains" suggest an increase in autocorrelation, and are therefore detrimental to the operation of the inference circuit. This domain growth is shown in Fig. 5, which presents the distribution of domain sizes at each stage for a tree with input values of 0.5. Though the input bitstream domains are quite short, the domains grow as the signals propagate through the circuit. By the fifth stage, which is the result of 32 input signals, the domain distribution deviates from that expected for uncorrelated stochastic bitstreams, with the most common domain length being 17 bits.

As the output of a C-element only switches in the case of both inputs having values opposite that of the output, the C-elements have a tendency to propagate signals with values biased toward the input values. More specifically, "inertia" causes the C-elements to be biased toward strongly correlated inputs with large domains. These correlation issues have been investigated for stochastic decoding applications, and several techniques have been described to decorrelate stochastic bitstreams [25]. The proposed counter and moving average circuits can reshuffle the bitstream and decrease the correlation, but come at the cost of increased area, power dissipation, and delay. This overhead therefore reduces the efficiency of the cascaded C-element inference circuit. Inherently probabilistic nanodevices [17], [20], [21], [26], [27] may be envisioned for the random number generators necessary in these decorrelation circuits, potentially bringing a substantial reduction in overhead. Decorrelation circuits can be periodically inserted between cascaded C-elements, with optimal design dependent on the tradeoff between efficiency and accuracy. Decorrelation circuits should therefore be considered for applications that require a level of accuracy unavailable with the simple cascaded C-element circuit.

D. Cascaded C-Element Precision

The importance of domain size is evident from Fig. 6, which shows the error for various bitstream lengths for several combinations of inputs. Similar to the case of a single C-element, the use of extreme probabilities leads to large domains and slow convergence. With cascaded gates, the intermediate signals can take on extreme values, resulting in massive errors for the green line with inputs of 0.01, 0.01, 0.99, and 0.99. The intermediate values are 10^{-4} and $1–10^{-4}$, preventing the representation of meaningful information with fewer than 10 000 bits. There are thus large output errors despite the expected output value being 0.5.

The effect of cascading on precision is studied in more depth in Fig. 7. Here, there is a tree structure with n stages and 2^n input stochastic bitstreams. The input values for one half of the bitstreams are $0.5 - k$, where k is a constant plotted on the x-axis. The second half of the bitstreams have input values of $0.5 + k$. This is the least favorable way to organize inputs, as the inputs of the final C-element encode extreme probabilities $(0.5 - k)^{n-1}$ and $(0.5 + k)^{n-1}$. Fig. 7(b) shows that the error and autocorrelation increases with increasing n and k, as the signals propagated by the second-to-last stage grow larger domains and become more extreme. The simulations thus show the importance of intermediate signals in determining output precision, and explores the autocorrelation limitations on the number of inputs and their values. Therefore, when possible, the inputs should be organized to minimize the potential for inertia and extreme values at intermediate nodes in the circuit.

Because of these considerations, the behavior of cascaded C-elements deviates from that of a single C-element, and the output of the C-element tree can diverge from the Bayesian inference of (13). Despite the challenges posed by autocorrelation, we have shown with the pedagogical example of spam filtering that cascaded C-elements can give useful results in practical situations.

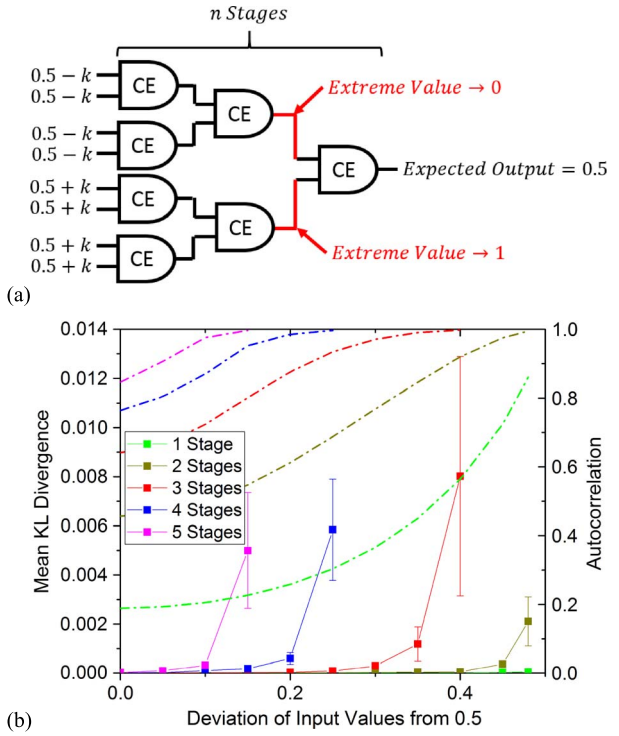

Fig. 7. (a) Cascaded C-element tree circuit with $n = 3$ stages to evaluate worst-case input probability combination. (b) Precision (solid lines) and autocorrelation (dashed lines) of tree as a function of input values for various depths with 500 000-bit bitstreams. Error bars depict 95% confidence interval.

V. Computing Implications

The use of stochastic computation brings exceptional fault tolerance, and the application-specific hardware enables high efficiency. This direct hardware implementation of Bayesian inference provides significant computing advantages which are particularly pronounced with error-prone nanodevices.

A. Fault Tolerance

For use with emerging nanotechnology, circuits should be designed to tolerate faults. Stochastic computation was originally designed for imprecise components, and fault tolerance is one of its primary strengths. Each incorrect bit in a stochastic bitstream causes a deviation from the correct value by only $1/N$, where N is the length of the bitstream. Additionally, if it is equally likely for a "0" to improperly represented by a "1" as for a "1" to be represented by a "0," then these faults may cancel out.

1) Not Switching: For the various C-element circuits shown in Fig. 1, the most likely fault is a missed switching event due to an insufficient current being provided to overpower the latch [28]. This type of fault will be referred to as a "not switching" fault, and can result from the use of a low supply voltage for circuits composed of variability-prone nanodevices.

The effect of this "not switching" fault can be seen in Fig. 8, which shows the impact of fault rates up to 99%. The simulation reflects the arbitrary case of the two inputs X and Y having probability values of 0.7 and 0.1; the expected output value from (9) is 0.206. One million bits are used in this simulation.

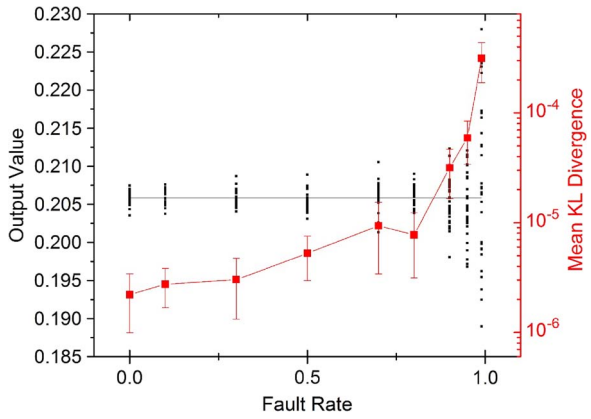

Fig. 8. The simulated (black dots) and expected (black line) output value for "not switching" faults. The mean KL divergence (red) is below 10^{-3} for a 99% fault rate. Error bars depict 95% confidence interval.

As seen in the graph, even for extremely high error rates, the output is close to the expected value. The KL divergence remains below 0.01, which may be considered a "correct" value, as discussed in the next subsection.

The error resulting from the "not switching" fault is unbiased: the output deviates from the correct value, but is not consistently larger or smaller. This is due to the nature of the "not switching" fault: as this fault also causes incorrect values in proceeding bits and there are necessarily equal numbers of switching events from "0" to "1" as from "1" to "0," there is an increase in domain length without biasing the output. This "not switching" fault is therefore equivalent to decreasing the bitstream length by the fault rate. Replacing the x-axis of Fig. 8 by $1\,000\,000 \times (1 - F_{\text{notswitching}})$, these results can be compared to the C-element outputs of Fig. 2. In both cases, the KL divergence rises from slightly greater than 10^{-6} to slightly greater than 10^{-4} as the bitstream length decreases from 1 000 000 to 10 000.

The exceptional tolerance of the "not switching" fault permits the consideration of advanced low-power design techniques with emerging nanotechnology. Reduction of the supply voltage provides a drastic decrease in power dissipation, but increases the likelihood of faults. Additionally, the clock speed can be increased to improve performance in exchange for an increase in faults. This fault tolerance thus provides an opportunity to optimize the power dissipation and speed for various applications and required levels of precision.

2) False Switching: If the latch itself is faulty, there is a possibility for the C-element to switch in the absence of a proper switching stimulus. This "false switching" fault is less likely than the non-switching fault. For realistic fault rates, far less than 1% in conventional circuits, the C-element structure provides the approximately correct output (KL divergence less than 0.001 for 1% fault rate). When the fault rate is increased toward 100%, the output approaches a random bitstream with probability 0.5. This can be understood as the expected result of the output switching every clock cycle—both those in which it is expected to switch, and those in which no switching event is expected. While this is not a realistic concern, this bias toward 0.5 can be interpreted in the context of a decision-making

process as uncertainty arising from unreliable nanoelectronic components.

B. Inference Efficiency

This C-element Bayesian inference structure uses a simple circuit to stochastically compute inferences. The precision increases as more bits are used for the inference, at the cost of increased energy and delay. In comparison to conventional implementations with floating point operations, the C-element inference structure provides significant improvements in area, performance, and energy consumption. However, when performing benchmark tests, the "correctness" of a Bayesian inference is application-specific. Here, a KL divergence of less than 0.01 is considered "correct" for an unspecified general circuit. Using (5), this divergence corresponds to a value of 0.5 being approximated as 0.57, a value of 0.1 being approximated as 0.15, and a value of 0.01 being approximated as 0.03. Clearly, more precision is necessary for some applications; for others, less precision is necessary.

To perform a comparison, the circuit efficiency is evaluated under identical technology node and process. Standard cells from Synopsys' SAED_EDK90_CORE library are used [29]; the efficiency of each component is considered at its ideal operating conditions. For the C-element, the circuit of Fig. 1(d) is chosen despite its non-ideality, as it can most easily be translated to standard cells. For the conventional floating point units, the parameters are optimized for the SAED_EDK90_CORE library by Tziantzioulis *et al.* [30]. These parameters are described in Appendix B.

1) Area: For a two-input Bayesian inference, this stochastic circuit requires just one C-element. This is in contrast to the conventional floating point implementation, which requires addition, multiplication, and division units. With the standard cells described above, the floating point circuit consumes $\sim$16 000 times more area than a C-element. This estimate ignores the overhead associated with random number generation, which may be efficiently achieved with emerging nanodevices [16], [20], [21] and will therefore become essential elements of stochastic computers.

For multi-input Bayesian inference, a single floating point circuit can be used. The stochastic C-element inference structure, however, requires $T-1$ C-elements, where T is the number of inputs. Therefore, the C-element provides an area reduction of $\sim$5400x for a two-stage (four-input) inference and $\sim$2300x for a three-stage (eight-input) inference. Similar calculations can be performed for inferences with additional inputs.

2) Performance: Whereas the stochastic C-element inference structure provides increasing precision with increasing time, the floating point implementation provides a highly-precise inference after its characteristic latency period. Therefore, the performance benefit provided by the C-element structure is dependent on the necessary precision for the specific application.

For multi-input Bayesian inference, the multi-stage C-element circuit needs only one additional cycle per stage to compute a bitstream with a particular length. However, the autocorrelation and domain growth contribute to a loss of precision, requiring longer bitstreams and slightly more computation time to achieve a precise output. The floating point circuit, by contrast, performs multiple pipelined computations and computes a highly precise output after its characteristic delay time. These processes are compared in Fig. 9, which considers a tree with n stages and 2^n input bitstreams all encoding the probability 0.5. As can be seen in the figure, the times required to achieve a KL divergence of 0.01 are of the same order of magnitude.

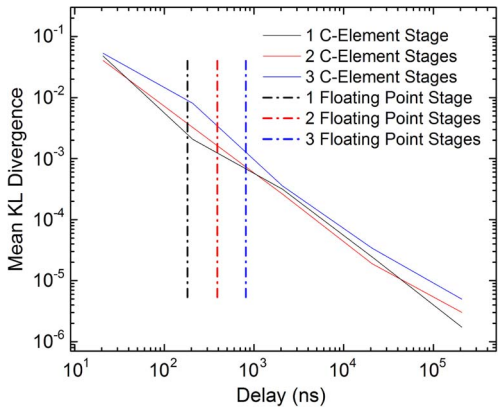

Fig. 9. Precision (KL divergence from exact inference) as a function of computing time for various inference circuits.

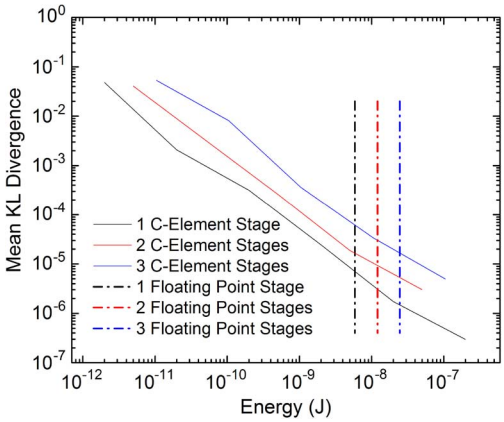

Fig. 10. Precision as a function of energy consumed by various inference circuits.

3) Energy Consumption: The C-element inference circuits consume a static energy proportional to the bitstream length in addition to a dynamic energy related to the switching rate, with the dynamic energy consumption largely dominant under typical conditions. Therefore, the energy required for a C-element to output a given value with a particular level of precision is nearly independent of input probability values. For extreme inputs, a large number of bits is required, but the effect is counteracted by the low switching rate. In comparison to the floating point implementation, there is an enormous reduction in the energy required to produce a KL divergence of 0.01, of roughly three orders of magnitude.

The multi-input C-element inference provides similarly large reductions in energy consumption compared to the floating point implementation. As shown in Fig. 10, the energy consumption of both approaches increases with an increasing

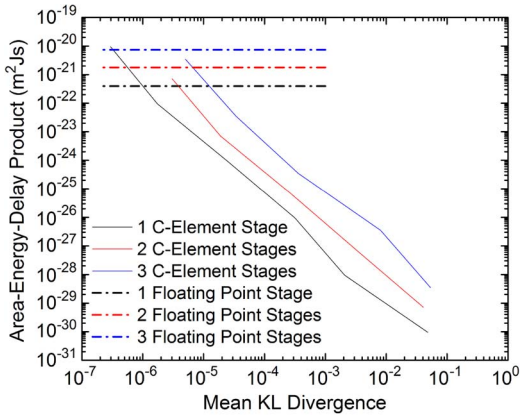

Fig. 11. AEDP as a function of precision for various inference circuits.

number of stages. These results show a ∼1000x reduction in energy consumption to produce an output with a KL divergence of 0.01.

VI. Conclusion

This work highlights the ability to perform Bayesian inference with simple circuits that can exploit emerging nanotechnology. In a stochastic approach, a circuit composed of Muller C-elements can integrate conflicting evidence from independent sources to provide an increasingly precise approximation of the probability of an event. Although extensions that provide a more formal analysis of the autocorrelation and explore inertia mitigation through signal rerandomization are envisioned, the C-element inference structure introduced here already achieves sufficient accuracy for some practical applications, and validates the basic concept of using stochastic computing to perform Bayesian inference.

The efficiency and fault tolerance of the C-element inference circuit provides significant advantages over conventional software implementations using floating point hardware. As shown in Fig. 11, the area-energy-delay product (AEDP) for producing a KL divergence of 0.01 is roughly one billionth of the floating point implementation. In addition, the exceptional fault tolerance provide an impetus for the use of stochastic C-elements in nanoelectronic decision circuits based on Bayesian inference.

For decision circuits, it is useful to have a constantly increasing precision with an initial approximation. Though floating point operations remain more effective for applications involving extreme values or requiring exact outputs, stochastic computing may provide important benefits for applications such as robotics. It also inspires a pathway toward compact embedded inference circuits with long battery life suitable for medical applications. The stochastic circuit structure proposed here has the potential to move Bayesian inference from software to nano-hardware, thereby enabling enormous improvements in decision-making efficiency.

Appendix A
Bayesian Spam Filter

A naïve Bayesian spam filter searches a message for particular key words and computes the probability of the supposition that the message is spam based on their presence or absence. As shown in Table II, eight key words are chosen and used for evaluating the messages. For each word, Laplace's rule of succession is applied to correctly handle incomplete observation:

$$P\left(W_x^1|S\right) = \frac{1 + \#\text{SpamWith}W_x}{2 + \#\text{Spams}} \tag{17}$$

$$P\left(W_x^1|H\right) = \frac{1 + \#\text{HamWith}W_x}{2 + \#\text{Hams}}. \tag{18}$$

The spamicity is calculated in accordance with Bayes' rule:

$$P\left(S|W_x^1\right) = \frac{P(S)P\left(W_x^1|S\right)}{P(S)P\left(W_x^1|S\right) + P(H)P\left(W_x^1|H\right)}. \tag{19}$$

Assuming that $P(S) = P(H)$ gives

$$P^*\left(W_x^1\right) = \frac{P\left(W_x^1|S\right)}{P\left(W_x^1|S\right) + P\left(W_x^1|H\right)}. \tag{20}$$

The probability that a message is spam given the absence of a particular key word is calculated similarly:

$$P^*\left(W_x^0\right) = \frac{P\left(W_x^0|S\right)}{P\left(W_x^0|S\right) + P\left(W_x^0|H\right)}. \tag{21}$$

The probability that a message is spam given the presence or absence of the various key words is calculated according to (14). For the case of message D, the equation is

$$P(S|\text{Message } D) = P\left(S \left| \begin{array}{l} \text{Nigeria}^1, \text{Pizza}^1, \text{Stochastic}^0, \text{Check}^1, \\ \text{You}^1, \text{Transfer}^1, \text{Commute}^1, \text{Fortune}^0 \end{array} \right.\right). \tag{22}$$

TABLE II
Word Count Statistics for Spamicity Calculations

Word	Spam Count (750,000 messages)	$P(W_x^1\|S)$	$P^*(W_x^1)$	Ham Count (250,000 messages)	$P(W_x^1\|H)$	$P^*(W_x^0)$
Nigeria	1,000	0.1%	99.4%	1	0.0%	50.0%
Pizza	10	0.0%	0.4%	1,000	0.4%	50.1%
Stochastic	0	0.0%	2.9%	10	0.0%	50.0%
Check	100,000	13.3%	76.9%	10,000	4.0%	47.4%
You	700,000	93.3%	53.8%	200,000	80.0%	25.0%
Transfer	10,000	1.3%	76.9%	1,000	0.4%	49.8%
Commute	0	0.0%	0.3%	100	0.0%	50.0%
Fortune	50,000	6.7%	94.3%	1,000	0.4%	48.4%

TABLE III
Inference Circuit Area Comparison

Circuit	Area (μm²)
Addition	126,189
Multiplication	200,463
Division	47,832
Total: Conventional Inference	374,483
C-Element	23

Appendix B
Comparison to Floating Point Inference

A. Area Comparison

The areas of the various circuits are listed in Table III. The circuit design has one hardware implementation for each function, and the results are pipelined when a particular function is used on multiple occasions. Whereas the area of the

TABLE IV
INFERENCE CIRCUIT PERFORMANCE COMPARISON

Circuit	Delay (ns)
Addition	35.5
Multiplication	18.4
Division	93.6
Total: Conventional Inference	218.5
C-Element (per bit)	0.21

TABLE V
INFERENCE CIRCUIT ENERGY CONSUMPTION COMPARISON

Circuit	Energy (J)
Addition	8.4 E-10
Multiplication	1.4 E-9
Division	5.0 E-10
Total: Conventional Inference	5.9 E-9
C-Element (per bit, inputs = 0.5)	2.0 E-14

conventional inference circuit is independent of the number of inputs to the inference, the C-element approach requires 2^{n-1} C-elements for an inference circuit with n stages.

B. Performance Comparison

The delays of the various circuits are listed in Table IV. For the conventional approach, ideal pipelining is assumed in which the operations are pipelined with a spacing equivalent to the number of issue cycles; there is therefore a decrease in delay per inference when more than one stage is combined for a cascaded circuit. These delay figures apply under the assumption that the separate hardware units of the floating point inference circuit each operate at their own ideal clock frequency. These assumptions are likely to underestimate the delay of the conventional inference.

C. Ernergy Consumption Comparison

The energy consumptions of the various circuits are listed in Table V. For the C-element, the expected energy consumed per bit is given for the worst case, with input values of 0.5. It is assumed that switching energy (but not power) is independent of frequency. For the conventional approach, it is assumed that the circuit dissipates static power throughout the complete inference calculation. In lieu of an accurate evaluation of the activity rate of the various circuit components, it is assumed that each cell of each arithmetic circuit switches once per arithmetic operation. These assumptions are likely to underestimate the energy consumption of the conventional inference.

ACKNOWLEDGMENT

The authors would like to thank N. Hardavellas, A.M. Gok, G. Tziantzioulis, D. Colliaux, E. Mazer, and J. Grollier for fruitful discussions.

REFERENCES

[1] P. Bessière, C. Laugier, and R. Siegwart, *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems*. Berlin, Germany: Springer-Verlag, 2008.
[2] W. J. Ma, J. M. Beck, P. E. Latham, and A. Pouget, "Bayesian inference with probabilistic population codes," *Nat. Neurosci.*, vol. 9, pp. 1432–1438, Nov. 2006.
[3] J. Laurens and J. Droulez, "Bayesian processing of vestibular information," *Biol. Cybern.*, vol. 96, pp. 389–404, Apr. 2007.
[4] A. Houillon, P. Bessière, and J. Droulez, "The probabilistic cell: Implementation of a probabilistic inference by the biochemical mechanisms of phototransduction," *Acta Biotheor.*, vol. 58, pp. 103–120, Sep. 2010.
[5] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer, "Bayesian robot programming," *Auton. Robots*, vol. 16, pp. 49–79, 2004.
[6] P. Bessière, E. Mazer, J.-M. Ahuactzin, and K. Mekhnacha, *Bayesian Programming*, Boca Raton, FL, USA: CRC, 2014.
[7] K. Bernstein, R. K. Cavin, W. Porod, A. Seabaugh, and J. Welser, "Device and architecture outlook for beyond CMOS switches," *Proc. IEEE*, vol. 98, no. 12, pp. 2169–2184, Dec. 2010.
[8] I. Pournara, C.-S. Bouganis, and G. A. Constantinides, "FPGA-accelerated Bayesian learning for reconstruction of gene regulatory networks," in *Proc. Int. Conf. Field Program. Logic Appl. (FPL)*, 2005, pp. 323–328.
[9] M. Lin, I. Lebedev, and J. Wawrzynek, "High-throughput Bayesian computing machine with reconfigurable hardware," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2010, pp. 73–82.
[10] B. Vigoda, "Analog logic: Continuous-time analog circuits for statistical signal processing," Ph.D. dissertation, Program Media Arts Sci., School Archit. Plan., Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.
[11] P. Mroszczyk and P. Dudek, " The accuracy and scalability of continuous-time Bayesian inference in analogue CMOS circuits," in *Proc. ISCAS*, 2014, pp. 1576–1579.
[12] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, C. Shannon, and J. McCarthy, Eds. Princeton, NJ, USA: Princeton Univ. Press, 1956.
[13] B. Gaines, "Stochastic computing systems," in *Advances in Information Systems Science* J. T. Tou, Ed. New York, NY, USA: Springer-Verlag, 1969, vol. 2, pp. 37–172.
[14] V. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," *Electron. Lett.*, vol. 39, no. 3, pp. 299–301, 2003.
[15] C. Winstead and S. Howard, "A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 6, pp. 484–488, Jun. 2009.
[16] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, Jul. 2013.
[17] P. Knag, W. Lu, and Z. Zhang, "A native stochastic computing architecture enabled by memristors," *IEEE Trans. Nanotechnol.*, vol. 13, no. 2, pp. 283–293, 2014.
[18] D. Querlioz, O. Bichler, A. F. Vincent, and C. Gamrat, "Bioinspired programming of memory devices for implementing an inference engine," *Proc. IEEE*, vol. 103, no. 8, pp. 1398–1416, 2015.
[19] T. Hamilton, S. Afshar, A. van Schaik, and J. Tapson, "Stochastic electronics: A neuro-inspired design paradigm for integrated circuits," *Proc. IEEE*, vol. 102, no. 5, pp. 843–859, 2014.
[20] A. Fukushima, T. Seki, K. Yakushiji, H. Kubota, H. Imamura, S. Yuasa, and K. Ando, "Spin dice: A scalable truly random number generator based on spintronics," *Appl. Phys. Exp.*, vol. 7, 2014, Art. no. 083001.
[21] W. H. Choi, Y. Lv, J. Kim, A. Deshpande, G. Kang, J. Wang, and C. H. Kim, "A magnetic tunnel junction based true random number generator with conditional perturb and real-time output probability tracking," in *Proc. IEDM*, 2014, pp. 315–318.
[22] D. E.Muller, "Theory of Asynchronous Circuits," University of Illinois, Graduate College, Digital Computer Laboratory, Internal Report no. 66, 1955.
[23] M. Shams, J. C. Ebergen, and M. I. Elmasry, "Modeling and comparing CMOS implementations of the C-element," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 4, pp. 563–567, 1998.
[24] C. Winstead, "C-element multiplexing for fault-tolerant logic circuits," *Electron. Lett.*, vol. 45, no. 19, pp. 969–970, 2009.
[25] S. Sharifi Tehrani, C. Winstead, W. J. Gross, S. Mannor, S. L. Howard, and V. C. Gaudet, "Relaxation dynamics in stochastic iterative decoders," *IEEE Trans. Signal Process.*, vol. 58, no. 11, pp. 5955–5961, 2010.
[26] S. Saighi, C. G. Mayr, T. Serrano-Gotarredona, H. Schmidt, G. Lecerf, J. Tomas, J. Grollier, S. Boyn, A. F. Vincent, D. Querlioz, S. La Barbera, F. Alibart, D. Vuillaume, O. Bichler, C. Gamrat, and B. Linares-Barranco, "Plasticity in memristive devices for spiking neural networks," *Front. Neurosci.*, vol. 9, p. 51, 2015.
[27] A. F. Vincent, J. Larroque, W. S. Zhao, N. Ben Romdhane, O. Bichler, C. Gamrat, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for

neuromorphic systems," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 2, pp. 166–174, 2015.
[28] J. A. Brzozowski and K. Raahemifar, "Testing C-elements is not elementary," in *Proc. 2nd Working Conf. Asynchronous Design Methodologies*, 1995, pp. 150–159.
[29] Synopsys, *Digital Standard Cell Library: SAED_EDK90_CORE Databook*, 2012.
[30] G. Tziantzioulis, A. M. Gok, S. M. Faisal, N. Hardavellas, S. Memik, and S. Parthasarathy, "b-HiVE: A bit-level history-based error model with value correlation for voltage-scaled integer and floating point units," in *Proc. DAC*, 2015, Art. no. 105.

Joseph S. Friedman (S'09–M'14) received the A.B. and B.E. degrees from Dartmouth College, Hanover, NH, USA, in 2009, and the M.S. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 2010 and 2014, respectively.

He joined the Institut d'Electronique Fondamentale in 2014, where he is a Research Associate with Université Paris-Sud, France, and the Centre National de la Recherche Scientifique, Orsay, France. He was a Guest Scientist at RWTH Aachen University, and worked on logic design automation for the Jaketown and Ivy Bridge processors as an intern at Intel Corporation. His research interests include the invention and design of novel cascaded computing structures based on nanoscale and quantum mechanical phenomena, with particular emphasis on spintronic logic families.

Dr. Friedman has been awarded a Fulbright Postdoctoral Fellowship and is a member of the editorial board of the *Microelectronics Journal*.

Laurie E. Calvet (M'97) received the B.S. degree in applied physics from Columbia University, New York, in 1995 and the Ph.D. degree in applied physics from Yale University, New Haven, CT, USA, in 2001.

In 2007 she joined the French Centre National de la Recherche Scientifique and carries out research at the Insititute d'Electronique Fondamentale, Université Paris-Sud, France. Her major fields of research include device physics of semiconductors and nanodevices and more recently hardware implementations of novel computing paradigms.

Pierre Bessière was born in 1958. He received the engineering and Ph.D. degrees in computer science from the Institut National Polytechnique de Grenoble (INPG), France, in 1981 and 1983, respectively.

He did a Post-Doctorate at SRI International (Stanford Research Institute) working on a project for National Aeronautics and Space Administration (NASA). He then worked for five years in an industrial company as the leader of different artificial intelligence projects. He came back to research in 1989. He has been a Senior Researcher at Centre National de la Recherche Scientifique (CNRS), Université Pierre et Marie Curie, Paris since 1992. His main research concerns have been evolutionary algorithms and probabilistic reasoning for perception, inference and action. He leads the Bayesian Programming research group (Bayesian-Programming.org) on these two subjects. Fifteen Ph.D. diplomas and numerous international publications are fruits of the activity of this group during the last 15 years. He also leads the BIBA (Bayesian Inspired Brain and Artefact) and was a Partner in the BACS (Bayesian Approach to Cognitive Systems) European project. He is a Cofounder and Scientific Adviser of the ProBAYES Company, which develops and sells Bayesian solutions for the industry.

Jacques Droulez (born in 1950) received a mathematical and engineer training (Ecole Polytechnique, Paris), a complete medical training (MD: Lariboisière—St Louis Hospital, Paris), a Master's in biochemistry, and a Habilitation to supervise research in cognitive sciences (Université Pierre et Marie Curie, Paris). He has a fellowship from the Centre National d'Etudes Spatiales (1978–1982). He is now Director of Research at the Centre National de la Recherche Scientifique (CNRS), and head of the research team "Active Perception and Probabilistic Approach" at the Laboratory of Physiology of Perception and Action (CNRS—Collège de France). His main research themes are the perception of 3D motion and objects, the theoretical study of models for multi-sensory interactions and the adaptive motor control. He has about 90 publications in international journals including one in PNAS on sensory-motor integration model and one in *Nature* on object perception during self-motion. He is involved in several European and national research programs and in multidisciplinary scientific networks.

Damien Querlioz (M'09) received the M.S. degree from the Ecole Normale Supérieure, Paris, France, in 2005 and the Ph.D. degree in device physics from the Université Paris-Sud, Orsay, France, in 2008.

After postdoctoral study at Stanford University and at CEA LIST, he became a Centre National de la Recherche Scientifique (CNRS) Research Scientist at the Université Paris-Sud, France, in 2010. He develops new concepts in nanoelectronics and spintronics relying on bioinspiration. His research interests have also included the physics of advanced nanodevices. He leads the ANR CogniSpin project, which investigates the use of magnetic memory as synapses. He leads the CNRS/MI DEFIBAYES project and is a one of the lead Principal Investigators of the FP7 FETOPEN BAMBI project, which explore the new paradigms for nanoelectronics based on Bayesian inference.