This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

The Promise and Challenge of Stochastic Computing

Armin Alaghi, *Member, IEEE*, Weikang Qian, *Member, IEEE*, and John P. Hayes, *Life Fellow, IEEE*

***Abstract*—Stochastic computing (SC) is an unconventional method of computation that treats data as probabilities. Typically, each bit of an *N*-bit stochastic number (SN) *X* is randomly chosen to be 1 with some probability p_X, and *X* is generated and processed by conventional logic circuits. For instance, a single AND gate performs multiplication. The value *X* of an SN is measured by the density of 1s in it, an information-coding scheme also found in biological neural systems. SC has uses in massively parallel systems and is very tolerant of soft errors. Its drawbacks include low accuracy, slow processing, and complex design needs. Its ability to efficiently perform tasks like communication decoding and neural network inference has rekindled interest in the field. Many challenges remain to be overcome, however, before SC becomes widespread. In this paper, we discuss the evolution of SC, mostly focusing on recent developments. We highlight the main challenges and discuss potential methods of overcoming them.**

***Index Terms*—Approximate computing, pulse circuits, stochastic circuits, unconventional computing methods**

I. Introduction

FROM its beginnings in the 1940s, electronic computing has relied on weighted binary numbers of the form $X = x_1 x_2 \cdots x_k$ to represent numerical data [16]. Typical is the use of *X* to denote a fixed-point fraction $v = \sum_{i=1}^{k} 2^{-i} x_i$ lying in the unit interval [0,1]. Efficient arithmetic circuits for processing such *binary numbers* (BNs) were soon developed. There were, however, concerns about the cost and reliability of these circuits, which led to the consideration of alternative number formats. Notable among the latter are *stochastic numbers* (SNs), where the x_i bits are randomly chosen to make *X*'s value be the probability p_X that $x_i = 1$. Again the resulting data values are in the unit interval [0,1]. In the late 1960s, research groups led by Gaines in the U.K. [28][29] and Poppelbaum in the U.S. [74] investigated data processing with SNs, a field that soon came to be known as *stochastic computing* (*SC*). Their pioneering work identified key features of SC, including its ability to implement arithmetic operations by means of tiny logic circuits, its redundant and highly error-tolerant data formats, and its low precision levels comparable to analog computing.

Like some early binary computers, stochastic circuits process data serially in the form of bit-streams. Figure 1 shows a stochastic number generator (SNG) that converts a given BN *B* to stochastic bit-stream form. The SNG samples a random BN *R* which it compares with *B*, and outputs an SN of probability $B/2^k$ at a rate of one bit per clock cycle. After *N* clock cycles, it has produced an *N*-bit SN *X* with $p_X \approx B/2^k$. The value p_X is the frequency or rate at which 1s appear, so an estimate $\hat{p}_X$ of p_X can be made simply by counting the 1s in *X*. In general, the estimate's accuracy depends on the randomness of *X*'s bit-pattern, as well as its length *N*. Rather than a true random source, an SNG normally employs a logic circuit like a linear feedback shift register (LFSR) whose outputs are repeatable and have many of the characteristics of true random numbers [32]. Mathematically speaking, the SNG approximates a Bernoulli process that generates random binary sequences of the coin-flipping type, where each new bit is independent of all earlier bits.

The essence of SC can be seen in how it is used to perform basic multiplication. Let *X* and *Y* be two *N*-bit SNs that are applied synchronously to a two-input AND gate, as in Figure 2. A 1 appears in the AND's output bit-stream *Z* if and only if the corresponding values of *X* and *Y* are both 1, hence

$$\hat{p}_Z \approx p_X \times p_Y \tag{1}$$

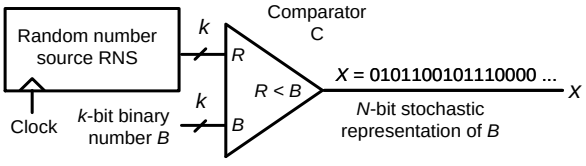


Figure 1. Stochastic number generator (SNG).

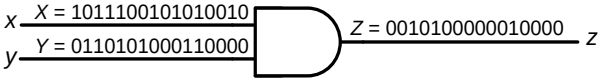


Figure 2. AND gate as a stochastic multiplier, with $p_X = 8/16$, $p_Y = 6/16$ and $p_Z = p_X \times p_Y = 3/16$. Equivalently, $Z = X \times Y = 3/16$.

March 2017: manuscript was submitted for review.

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grants No. 61472243 and 61204042, and by Grant CCF-1318091 from the U.S. National Science Foundation.

Armin Alaghi is with the Computer Science and Engineering Department of the University of Washington, Seattle, WA 98195. E-mail: amin@cs.washington.edu.

Weikang Qian is with University of Michigan-Shanghai Jiao Tong University Joint Institute at Shanghai Jiao Tong University, Shanghai, China 200240. E-mail: qianwk@sjtu.edu.cn.

John P. Hayes is with the Computer Science and Engineering Division, University of Michigan, Ann Arbor, MI 48109. E-mail: jhayes@umich.edu.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

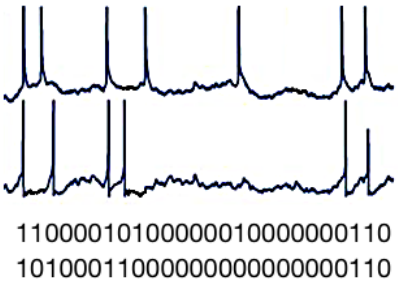


Figure 3. Spike trains in a biological neural network and equivalent SNs.

In other words, the AND gate serves as a multiplier of probabilities, and can be orders of magnitude smaller than a comparable BN multiplier. The SC multiplier's output bit-pattern Z varies with the randomness of the SNGs generating X and Y. These variations have little impact on the multiplier's output value $\hat{p}_Z$, however, indicating a naturally high degree of error tolerance. On the other hand, the precision with which $\hat{p}_Z$ reflects $p_X \times p_Y$ tends to be rather low. This situation can be improved by increasing N, but N must be doubled for every desired extra bit of precision, a property that leads to very long bit-streams and slow computations. For example, BNs of length $k = 8$ provide 8-bit precision. To obtain similar precision with SC requires SNs of length $N = 2^k = 256$ or more. Hence, SC tends to be restricted to low-precision applications where the bit-streams are not excessively long. More troublesome is the need for X and Y to have statistically independent or uncorrelated bit-patterns in order for $\hat{p}_X$ and $\hat{p}_Y$ to be treated as independent probabilities, as required by Equation (1). In the extreme case where exactly the same bit-pattern X is applied to both inputs of the AND gate, the output bit-stream's value becomes p_X instead of p_X^2, implying a potentially large computation error which cannot be corrected simply by extending N.

As the cost and reliability of conventional *binary computing* (BC) improved in the 1960s and '70s with the development of integrated circuits tracked by Moore's Law, interest in SC waned. It was seen as poorly suited to general-purpose computation, where high speed, accuracy, and compact storage were routinely expected. However, SC continued to find niche applications in areas such as image processing, control systems, and models of neural networks, which can take advantage of some of its unique features.

Neural networks, both natural and artificial, constitute an interesting case. As Figure 3 suggests, biological neurons process noisy sequences of voltage spikes which loosely resemble SNs [31][57]. Information is encoded in both the timing and the frequency of the spikes—the exact nature of the neural code is one of nature's mysteries. However, significant information, such as the intensity of a muscular action, is embedded SN-like in the spike rate over some time window; the spike positions also exhibit SN-style randomness. Moreover, the operation of a single neuron is commonly modeled by an inner-product function of the form

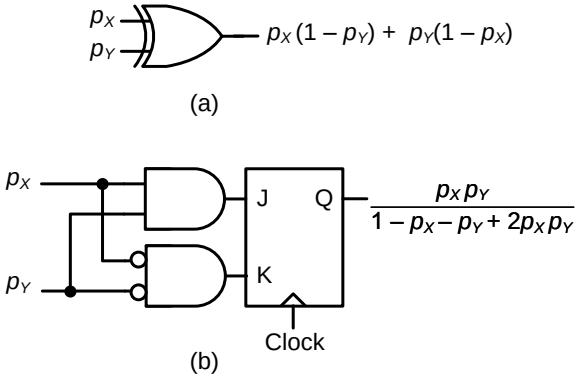


Figure 4. Stochastic circuits for LDPC decoding [30]: (a) parity check node; (b) equality node.

$$F = \sum_{i=1}^{N} W_i \times X_i \quad (2)$$

where the X_i's are signals from other neurons, and the W_i's are synaptic weights denoting the influence of those neurons. Since the number of interneural connections and multiplications N can be in the thousands, SC-based implementations of Equation (2) are attractive because of their relatively low hardware cost [14][46].

The state of SC circa 2000 can be characterized as focused on a handful of old and specialized applications [3][59]. The situation changed dramatically when Gaudet and Rapley observed that SC could be applied successfully to the difficult task of decoding low-density parity check (LDPC) codes [30]. Although LDPC codes, like SC, were discovered in the 1960s, there was little practical interest in them until the advent of suitable decoding methods and circuits, as well as the inclusion of LDPC codes in new wireless communication standards such as digital video broadcasting (DVB-S2) and WiMAX (IEEE 802.16). LDPC decoding employs a probabilistic algorithm that passes messages around a code representation called a Tanner graph, while repeatedly performing two basic operations, parity checking and equality checking. It turns out that these operations are implemented efficiently by the stochastic circuits in Figure 4. Many copies of these circuits can be operated in parallel, resulting in fast, low-cost decoding, and demonstrating the potential of SC to provide massive parallelism. Recent developments have shown that SC-based LDPC decoders are competitive in performance and cost with conventional binary designs [47].

Other new applications and technology developments supported this revival of interest in SC. With the emergence of mobile devices such as smart phones and medical implants, extremely small size and power, as well as low-cost digital signal processing, have become major system goals [48]. An illustrative application of SC in the medical field is the design of retinal implants to aid the blind. An implant chip can be placed in the eye to receive

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

and process images and transfer the results via pulse trains through the optic nerve directly to the brain. The chip must satisfy extraordinarily severe size and power constraints, which SC is particularly well-suited to meet [4].

Significant aspects of SC that had been ignored in the past—Why does the apparently simple logic circuit of Figure 4b implement such a complex arithmetic function?— now began to receive attention. The relation between logic circuits and the stochastic functions they implement has been clarified, resulting in general design procedures for implementing arithmetic operations [75]. Correlation effects in SC have recently been quantified, leading to the surprising conclusion that correlation can serve as a valuable computational resource [5]. Bit-stream length can be reduced by careful management of correlation and precision (progressive precision [6]). The high contribution of stochastic-binary number conversion circuits to overall SC costs [75] is being recognized and addressed. New technologies, notably memristors, have appeared that have naturally stochastic properties which reduce data-conversion needs [43].

Despite these successes, SC still has limitations that must be considered when used in certain applications. Most importantly, the run time of SC circuits increases prohibitively when high precision or highly accurate computations are needed. Recent investigations have shown that the long computation time may lead to excessive energy consumption, thus making low-precision BC a better choice [1][58][62]. Manohar [58] provides a theoretical comparison between SC and BC and shows that even for multiplication, SC ends up having more *gate invocations* (i.e., the number of times an AND gate is called). Aguiar and Khatri [1] perform a similar comparison but instead of comparing the number of gate invocations, they actually implement BC and SC multipliers with different bit widths. They conclude that SC multiplication is more energy efficient for computations that require 6 bits of precision (or lower). However, if conversion circuits are needed, SC is almost always worse than BC [1].

This poses an important challenge to SC designers: their designs must be competitive in terms of energy efficiency with BC circuits of similar accuracy/precision. Some of the topics that can potentially address this problem are (i) exploiting progressive precision to reduce overall run time, (ii) exploiting SC's error tolerance to improve energy usage, and (iii) reducing or eliminating the cost of data conversion. Examples of these techniques appear in the current literature.

This paper focuses on more recent SC work than the survey [3], and attempts to highlight the big challenges facing SC and their potential solutions. The remainder of the paper is organized as follows. Section II provides a formal introduction to SC and its terminology, including SC data formats, basic operations, and randomness requirements. Readers familiar with the topic can skip this section. General synthesis methods for combinational and sequential SC circuits are discussed in Section III. Section IV examines the application domains of SC, as well as some emerging new applications. Concluding remarks and future challenges of SC are discussed in Section V.

II. Basic Concepts

Probabilities are inherently analog quantities that correspond to continuous real numbers. Stochastic circuits can be therefore interpreted as hybrid analog-digital circuits because they employ digital components and signals to process analog data. Theoretically, the AND gate of Figure 2 can perform multiplication on numbers with arbitrary precision. However, to find the probability $p_Z = p_X \times p_Y$ we must obtain a finite number of discrete samples of the circuit's output from which to estimate p_Z. The estimation's accuracy increases slowly with the number of samples, and is limited by noise considerations, making it impractical to estimate p_Z with high precision.

A. Stochastic number formats

Interpreting SNs as probabilities is natural, but it limits them to the unit interval [0,1]. To implement arithmetic operations outside this interval, we need to scale the number range in application-dependent ways. For example, integers in the range [0,256] can be mapped to [0,1] by dividing them by a scaling factor of 256, so that {0, 1, 2, …, 255, 256} is replaced by {0, 1/256, 2/256, …, 255/256, 1}. Such scaling can be considered as a pre-processing step required by SC.

SC can readily be defined to handle signed numbers. An SN X whose numerical value is interpreted in the most obvious fashion as p_X is said to have the *unipolar* format. To accommodate negative numbers, many SC systems employ the *bipolar* format where the value of X is interpreted as $2p_X - 1$, so the SC range effectively becomes [−1, 1]. Thus, an all-0 bit-stream has unipolar value 0 and bipolar value −1, while a bit-stream with equal numbers of 0's and 1's has unipolar value 0.5, but bipolar value 0. Note that the function of an SC circuit usually changes with the data format used. For instance, the AND gate of Figure 2 does not perform multiplication in the bipolar domain. Instead, an XNOR gate must be used, as shown in Example 1 below. On the other hand, both formats can use the same adder circuit. In what follows, to reduce confusion, we use X to denote the numerical value of the SN X. With this convention, $X = p_X$ in the unipolar domain, while $X = 2p_X - 1$ in the bipolar domain.

Several other SN formats have appeared in the literature [60]. *Inverted bipolar* is used in [2] to simplify the notation for spectral transforms. In [61] the value of a bit-stream is interpreted as the ratio of 1's to 0's, which creates a very wide, albeit sparse, number range. Table I shows the various number formats mentioned so far.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Table I. Possible interpretations of a bit-stream of length N containing N_1 1's and N_0 0's.

Format	Number value	Number range	Relation to unipolar value p_X
Unipolar (UP)	N_1/N	[0, 1]	p_X
Bipolar (BP)	$(N_1 - N_0)/N$	[−1, 1]	$2p_X - 1$
Inverted bipolar (IBP)	$(N_0 - N_1)/N$	[−1, 1]	$1 - 2p_X$
Ratio of 1's to 0's	N_1/N_0	[0, +∞]	$p_X/(1 - p_X)$

These formats deal with single bit-stream only. Dual-rail and multi-rail representations have also been proposed. Gaines [29], for example, presents dual-rail unipolar and bipolar number formats, along with the basic circuits for each format. Toral et al. propose another dual-rail encoding that represents a ternary stochastic number $X = x_1x_2 \ldots x_N$, where each $x_i \in \{-1,0,1\}$ [94]; it will be discussed in Section IV-A. The binomial distribution generator of [75], which is discussed in Section III, produces a multi-rail SN.

B. Stochastic number generation

We can map an ordinary binary number to an SN in unipolar format using the SNG in Figure 1. To convert the unipolar SN back to the binary, it suffices to count the number of 1's in the bit-stream using a plain (up) counter. Slight changes to these circuits allow for conversion between bipolar SNs and binary numbers. In SC, number-conversion circuits tend to cost much more than number-processing circuits. For example, to multiply two 8-bit binary numbers using the SC multiplier of Figure 2, we need two SNGs and a counter. A rough gate count reveals that the conversion circuits have about 250 gates while the computation part, is just a single AND gate. Extensive use of conversion circuits can severely affect the cost of SC circuits. Qian et al. [76] report that the conversion circuits consume up to 80% of the total area of several representative designs. For this reason, it is highly desirable to reduce the cost of conversion circuits.

Methods to reduce the cost of constant number generation are investigated in [25][79]. For massively parallel applications such as LDPC decoding, a single copy of random number generator can be shared among multiple copies of SC circuits to provide random inputs, thus effectively amortizing the cost of conversion circuits [21][89]. Furthermore, inherently stochastic nanotechnologies like memristors offer the promise of very low-cost SNGs [43]. The cost of data conversion can also be lowered if analog inputs are provided to the SC circuit. In this case, it may be feasible to directly convert the inputs from analog to stochastic using ramp-compare analog-to-digital converters [46][64] or delta-sigma converters [83].

C. Accuracy and randomness

The generation of an SN X resembles an ideal Bernoulli process producing an infinite sequence of random 0's and 1's. In such a process, each 1 is generated independently with fixed probability p_X; 0's thus appear with probability $1 - p_X$. The difference between the exact value p_X and its estimated value $\hat{p}_X$ (estimated over N samples) indicates the accuracy of X. This difference is usually expressed by the mean square error (MSE) E_X given by

$$E_X = \mathbb{E}[(\hat{p}_X - p_X)^2] = \frac{p_X(1 - p_X)}{N} \quad (3)$$

Equation (3) implies that inaccuracies due to random fluctuations in the SN bit-patterns can be reduced as much as desired by increasing the bit-stream length N. Hence the precision of X can be increased by increasing N or, loosely speaking, the quality of a stochastic computation tends to improve over time. This property is termed *progressive precision*, and is a feature of SC that will be discussed further later.

Stochastic circuits are subject to another error source which is much harder to deal with, namely insufficient independence or correlation among the input bit-streams of a stochastic circuit. Correlation is due to signal reuse caused by reconvergent fanout, shared randomness sources, and the like. As noted in Section I, if a bit-stream representing X is fanned out to both inputs of the AND gate in Figure 2, the gate computes X instead of X squared. This major error is due to maximal (positive) correlation between the AND's input signals. In general, if correlation changes the output number, the resulting error does not necessarily go toward zero as N increases.

It is instructive to interpret SN generation as a Monte Carlo sampling process [6]. Consider again the SNG of Figure 1 and, for simplicity, assume that both the input B and the random source R have arbitrary precision. Assume further that the value p_X of B is unknown. The SNG effectively generates a sequence X of N samples, and we can get an estimate $\hat{p}_X$ of p_X by counting the number of 1's in X. It is known that $\hat{p}_X$ converges to the exact value p_X at the rate of $O(1/\sqrt{N})$.

For most stochastic designs, LFSRs are used as the random number sources to produce stochastic bit-streams. Although these random sources are, strictly speaking deterministic, they pass various randomness tests [32][44] and so are considered pseudo-random. Such tests measure certain properties of a bit-stream, e.g., the frequency of 1's, the frequency of runs of k 1's, etc., and check the extent to which these properties match the behavior of a true random number generator.

Despite what is commonly believed, SNs do not need to pass many randomness tests. In fact, in order to have $\hat{p}_X = p_X$ we only need X to have the correct frequency

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

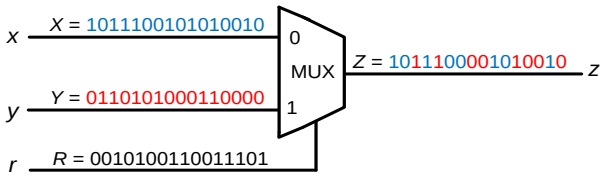


Figure 5. Multiplexer serving as a stochastic adder, with $p_X = 8/16$, $p_Y = 6/16$, $p_R = 8/16$, and $p_Z = 1/2(p_X + p_Y) = 7/16$.

of 1's. So it is possible to replace random number sources by so-called *deterministic* sources, which employ predictable patterns and lack most of the usual randomness attributes [6][38]. An example of a deterministic format is where all the 1's of an SN are grouped together and followed by all the 0's, as in 111111100000 [13].

To generate a deterministic bit-stream of the above form, we can use a counter to generate a sequence of deterministic values $0, 1/N, 2/N, \ldots, (N-1)/N$ and feed it to the comparator of Figure 1. It can be proved that the difference between $\hat{p}_X$ (the value of the generated bit-stream) and p_X (the constant number fed to the comparator) is no more than $1/N$, implying that $\hat{p}_X$ converges to p_X at the faster rate of $O(1/N)$. This motivates the use of deterministic number sources in SC, and indeed some SC circuits use such deterministic numbers [6]. However, there are several challenges to overcome when deterministic number formats are used, including limited scalability, and the cost of number generation to conserve the deterministic formats.

When many mutually uncorrelated SNs are needed, we can still extend the foregoing deterministic number generation approach, but its cost significantly increases with number of inputs. Gupta and Kumaresan [34] described an SN multiplier that produces exact results for any given input precision. However, to multiply k m-bit numbers using their method requires bit-streams of length 2^{km}, which becomes impractical for circuits with a large number of inputs.

By employing the deterministic approach, one gains a better control over the progressive precision of the SNs. Random number sources provide this property naturally to some degree. To fully exploit it, quasi-random or low-discrepancy sources may be used [6]. SNs generated via low-discrepancy sequences converge with the rate of $O(1/N)$. However, the benefits of using low-discrepancy sequences also diminish as the number of inputs increase, because the cost of generating them is much higher than pseudo-random number generation.

In summary, it may be beneficial to use deterministic number sources for SC circuits that have few inputs (three or fewer uncorrelated inputs). For circuits with more number sources, it appears better to use LFSRs and settle for the slower $O(1/\sqrt{N})$ convergence rate.

D. Basic arithmetic operations

SC multiplication was discussed in the previous sections. SC addition is usually performed by a multiplexer (MUX) implementing the Boolean function $z = (x \wedge r') \vee (y \wedge r)$, where x and y are the primary (data) inputs and r is the select input. A purely random bit-stream of probability $p_R = 0.5$ is applied to r. The bit-streams X, Y, and Z can be interpreted either as unipolar or bipolar. As Figure 5 shows, half the output bit-stream Z comes from X (blue) and the other half from Y (red), as decided by R. It follows that $p_Z = 0.5(p_X) + 0.5(p_Y)$. Therefore, with either the unipolar or bipolar format, the output value $Z = 0.5X + 0.5Y$. Notice that R provides a scaling factor of 0.5 and maps the sum to [0, 1] in the unipolar case, or to [−1, 1] in the bipolar case. This type of scaled addition entails a loss of precision since half of the information in the input bit-streams is effectively discarded. Thus, in the case of Figure 5 where the input precision is $\log_2 N = 4$ bits, the precision of the output also drops to 4 bits (as opposed to the expected 5 bits of precision). To ensure that Z has precision of 5, the length of all the bit-streams would have to be doubled to 32. It should also be noted that the probability p_Z can be expected to fluctuate around $0.5(p_X + p_Y)$ due to random fluctuations in R.

Several other adder designs have been proposed in the literature. A novel, scaling-free stochastic adder is proposed in [99], which operates on the ternary stochastic encoding proposed in [94]. Its key idea is to use a counter to remember carries of 1 and -1 and release them at a later time slot. Lee et al. [46] describe an adder that eliminates the need for a separate random source. Since adding is expensive, Ting and Hayes [91] propose using accumulative parallel counters (APCs) [71] in computations that end with an adding reduction e.g., matrix multiplication. An APC performs addition and stochastic-to-binary conversion simultaneously.

SC subtraction is easily implemented in the bipolar domain. Because inverting a bit-stream negates its bipolar value, we can use an inverter and a MUX to implement a bipolar subtractor. However, with unipolar encoding, since the value range [0, 1] does not include negative numbers, implementing subtraction becomes complicated. Various methods of approximating unipolar subtraction exist in the literature [5][27].

SC division is the most difficult of the basic operations. First, the result $Z = X_1/X_2$ falls outside the range [0, 1] if $X_1 > X_2$, so we must assume that $X_1 \leq X_2$. Second, as will be discussed in Section II.E, SC combinational circuits are only capable of implementing multilinear functions, but division is naturally a non-linear function. Nevertheless, SC circuits that implement division have been proposed in the literature. These circuits either include sequential elements, or exploit correlation among the input SNs.

Gaines [29] implemented division using a feedback loop (Figure 6). First, an initial guess of the result is stored in a binary variable p_Z, and then $Y = Z \times X_2$ is calculated using an SC multiplier. If Z were a correct guess of the division result X_1/X_2, then $X_1 = Y$ must hold. So based on the observed value of Y, the guessed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

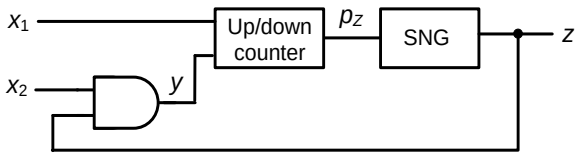


Figure 6. Counter-based stochastic divider.

result p_Z is updated. If $Y > X_1$, p_Z is reduced, and if $Y < X_1$, p_Z is increased. Given sufficient time, p_Z eventually converges to X_1/X_2. Note that this method needs a binary register to hold the guessed result p_Z, and an SNG to generate an SN representing Z. Furthermore, the convergence time of the circuit can be long.

An approximate divider can be implemented by a JK flip-flop. If we connect the J and K inputs to X_1 and X_2, respectively, then the SN appearing at the output of the flip-flop implements $Z = X_1/(X_1 + X_2)$. The JK flip-flop of Figure 4b is used for the purpose of division. Recently, a new SC divider has been proposed by Chen and Hayes [20]. This divider exploits correlation among its inputs and implements an exact division function.

E. Stochastic functions

As shown in the previous sections, SC addition and multiplication can be implemented by simple combinational circuits. A related question is: Given an arbitrary combinational circuit, what SC function does it compute?

Consider a combinational circuit C implementing the Boolean function $f(x_1, \dots, x_k)$. When supplied with uncorrelated SNs, C implements the (unipolar) stochastic function $F(X_1, \dots, X_k)$ defined by

$$\begin{aligned} F(X_1, \dots, X_k) = \\ (1 - X_1)(1 - X_2) \dots (1 - X_k) f(0,0, \dots, 0) \\ + (1 - X_1)(1 - X_2) \dots (X_k) f(0,0, \dots, 1) \\ \dots \\ + (X_1)(X_2) \dots (X_k) f(1,1, \dots, 1) \end{aligned} \quad (4)$$

When expanded out, Equation (4) takes the form of a multilinear polynomial. Consequently, combinational circuits with uncorrelated inputs can only approximate their target function via a suitable multilinear polynomial (see Section III).

Example 1: Let $f(x_1, x_2)$ be the logic function of an XOR gate. Then from Equation (4)

$$\begin{aligned} Z = F(x_1, x_2) &= (1 - X_1)X_2 + X_1(1 - X_2) \\ &= X_1 + X_1 - 2X_1X_2 \end{aligned}$$

or, equivalently,

$$p_Z = p_{X_1} + p_{X_2} - 2\, p_{X_1} p_{X_2} \quad (5)$$

Thus $X_1 + X_2 - 2X_1X_2$ is the unipolar stochasic function of XOR. If we treat the SNs as bipolar numbers, where $X = 2p_X - 1$, Equation (5) can be rewritten as $2p_Z - 1 = -(2p_{X_1} - 1)(2p_{X_2} - 1)$, i.e., $Z = -X_1X_2$. Hence, an XOR gate serves as a bipolar multiplier with negation.

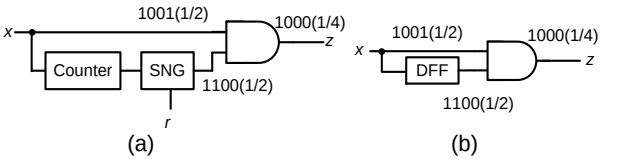


Figure 7. (a) Regeneration-based and (b) isolation-based decorrelation of a squarer circuit. DFF denotes a D flip-flop isolator.

Using the inverted bipolar or IBP format (see Table I), we get $F = X_1X_2$, and the XOR gate becomes an IBP multiplier without negation. Clearly, an XNOR gate is the basic bipolar multiplier. □

We can extend the functionality of SC circuits by incorporating sequential elements, as in the examples of Figure 4. In particular, sequential elements enable implementation of rational functions. Section III shows how arbitrary functions can be implemented efficiently using sequential SC circuits.

F. Correlation in stochastic operations

Although correlation in input SNs is usually detrimental to the functional correctness of stochastic circuits, careful use of correlation may be beneficial. Indeed, by feeding a circuit with inputs that are intentionally correlated, we obtain a different SC function, which may sometimes be very useful. For example, an XOR gate with maximally correlated inputs X and Y implements the absolute difference function $|X - Y|$, as shown in [5].

To measure the correlation between SNs, Alaghi and Hayes [5] introduced a similarity measure called *SC correlation* (SCC), which is quite different from the more usual Pearson correlation measure [22]. It is claimed in [5] that SCC is more suitable for SC circuit design because unlike the Pearson correlation, it is independent from the value of SNs. However, SCC cannot be easily extended to more than two SNs.

Maintaining a desired level of correlation between SNs is difficult. Consider the problem of *decorrelation*, i.e., systematic elimination of undesired correlation. There are two main ways to reduce correlation. One is regeneration, which converts a corrupted SN to binary form and then back to stochastic using a new SNG. An example of this is shown in Figure 7a, which computes $Z = X^2$. This decorrelation method has very high hardware cost, and may eliminate desirable properties such as progressive precision. An alternative method called isolation is illustrated in Figure 7b. A D-flip-flop (DFF) is inserted into line x and clocked at the bit-stream frequency, so it delays X by one clock cycle. If the bits of X are independent, as is normally the case, then X and a k-cycle delayed version of X are statistically independent in any given clock period. In general, isolation-based decorrelation has far lower cost than regeneration, but the numbers and positions of the isolators must be carefully chosen. Ting and Hayes [92] have developed a the-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

ory for placing isolators and have obtained conditions for a placement to be valid.

As noted earlier, a stochastic multiplier requires independent inputs for correct operation. However, Alaghi and Hayes noticed that some operations, including MUX-based addition, do not require their inputs to be independent [8]; such circuits are called *correlation insensitive* (CI). Figure 8 shows how correlation insensitivity can be exploited in an SC adder. The original design of Figure 8a assumes that inputs X and Y are generated independently. Because an SC adder is CI, the input random number sources (RNSs) can be shared as shown in Figure 8b. Correlation between X and Y does not lead to errors, since the output bit z at any time is taken from X or Y, but not both; cf. Figure 5.

III. Design Methods

Until recently, stochastic circuits were designed manually. The circuits of Figure 4 are examples of clever designs that implement complex functions with a handful of gates. Designing stochastic circuits for arbitrary functions is not easy. This problem has been studied intensively in the last few years, and several general synthesis methods have been proposed [2][7][19][49][52][82][101]. These methods can be classified into two types depending on whether the target design is reconfigurable or fixed. A reconfigurable design has some programmable inputs that allow the same design to be reused for different functions. A fixed design can only implement one target function. In this section, unless otherwise specified, we only discuss SC design in the unipolar domain.

A. Reconfigurable stochastic circuits

The basic form of a reconfigurable stochastic circuit is shown in Figure 9. Its computing core consists of a distribution-generating circuit (DGC) and a MUX. The DGC has m inputs $x_1, \dots, x_m$. It outputs a binary value s in the range $\{0,1,\dots,n\}$. The inputs $x_1, \dots, x_m$ are fed with independent SNs $X_1, X_2, \dots, X_m$, all encoding the same variable value X. Then the port s outputs a sequence of random numbers. The probability of s to assume the value i $(0 \le i \le n)$ is a function of the variable X, denoted by $F_i(X)$. With different DGCs, different probability distributions $F_0(X), \dots, F_n(X)$ of s can be achieved. The signal s is used as the select input of the MUX. The data inputs of the MUX are $n+1$ SNs $B_0, \dots, B_n$, which encode constant probabilities $B_0, \dots, B_n$. The value of the output SN Y of the MUX can be expressed as

$$Y = P(y=1) = \sum_{i=0}^{n} P(y=1|s=i)P(s=i) = \sum_{i=0}^{n} B_i F_i(X) \quad (6)$$

As Equation (6) shows, the final output is a linear combination of the distribution functions $F_0(X), \dots, F_n(X)$. This type of circuit is reconfigurable because with different sets of constant values B_i, different functions can be realized using the same design. Of course, not every function can be realized exactly by Equation (6). Given a target function $G(X)$, an optimal set of constant values are determined by minimizing the approximation error between the linear combination and $G(X)$ [76].

Prior research on synthesizing reconfigurable stochastic circuits can be distinguished by the form of the DGC proposed. The first work in this category employed an adder as the DGC [77]. The adder takes n Boolean inputs and computes their sum as the output signal s. Given that the n Boolean inputs are independent and have the same probability X of being 1, the output s follows the well-known binomial distribution

$$P(s=i) = \binom{n}{i}(1-X)^{n-i}X^i$$

for $i = 0,1,\dots,n$. Therefore, the computation realized has the following form:

$$Y = \sum_{i=0}^{n} B_i \binom{n}{i}(1-X)^{n-i}X^i \quad (7)$$

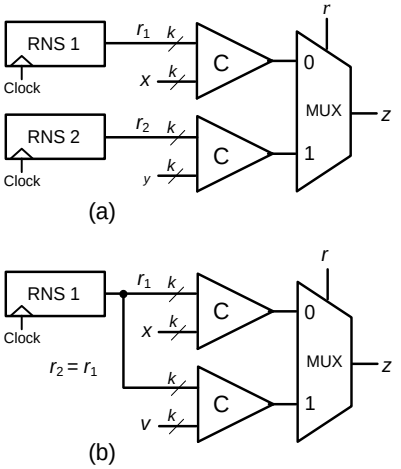


Figure 8. Exploiting correlation insensitivity in an SC adder: (a) original design and (b) design sharing an RNS.

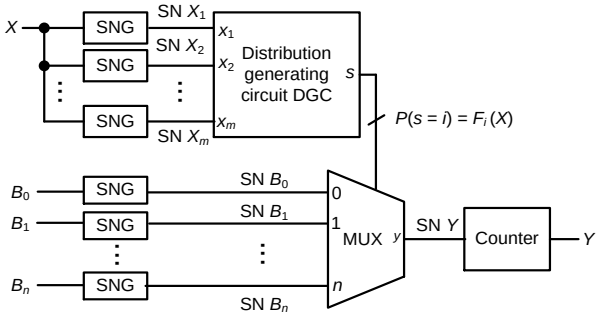


Figure 9. Reconfigurable stochastic circuit; examples of the distribution generating circuit include an adder [77] and an up/down counter [52].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

which is known as a Bernstein polynomial [55][78]. The approach of [76] finds a Bernstein polynomial that is closest to the target function and realizes it using the reconfigurable stochastic circuit. The drawback of this method is that n SNGs are required to generate n SNs $X_1, \dots, X_n$. To address this issue, later work explored the use of sequential circuits as the DGC. The key is to find a simple circuit which produces a distribution that approximates arbitrary functions closely.

Li et al. [52] first studied the use of an up/down counter as the DGC. The counter has a Boolean input x and outputs the current count value. If $x = 1$, the count increases by one, otherwise it decreases by one. The count value remains unchanged for $x = 1$ if it has reached its maximal value. Also, it remains unchanged for $x = 0$ if it has reached its minimal value.

If the input x carries an SN X, the state behavior of the counter can be modeled as a time-homogeneous Markov chain [82]. A Markov chain has an equilibrium distribution $(\pi_0(X), \dots, \pi_n(X))$, where $\pi_i(X)$ is the probability of the state i at equilibrium, which is a function of the input value X. The equilibrium probability distribution can be used as the DGC of Figure 9, yielding $F_i(X) = \pi_i(X)$ and

$$Y = \sum_{i=0}^{n} B_i \pi_i(X)$$

However, the reconfigurable stochastic circuit using the counter as the DGC is not able to approximate a wide range of functions. To enhance the representation capability, extensions were proposed in [49][84]. These extensions use FSMs with extra degrees of freedom, thus allowing a wider range of functions to be implemented.

B. Fixed stochastic circuits

In many applications, the computation does not change, so a fixed stochastic circuit is enough. The design of fixed stochastic circuits based on combinational logic has been studied in several recent papers [2][7][101].

The work in [7] proposes a synthesis method STRAUSS based on the Fourier transform of a Boolean function. The Fourier transform maps a vector $\vec{T}$ representing a Boolean function into a spectrum vector $\vec{S}$ as follows

$$\vec{S} = \frac{1}{2^n} H_n \times \vec{T} \tag{8}$$

Here $\vec{T}$ is obtained by replacing 0 and 1 in the output column of the truth table by $+1$ and -1, respectively, and H_n is the Walsh matrix recursively defined as

$$H_1 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix},\ H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}$$

The authors of [7] first demonstrate a fundamental relation between the computation of a stochastic circuit and its spectrum vector. They use the IBP format for SNs defined in Table I. The Boolean function $f(x_1, \dots, x_n)$ then corresponds to the stochastic function

$$F(X_1, \dots, X_n) = \sum_{(a_1,\dots,a_n)\in\{0,1\}^n} c(a_1, \dots, a_n) \prod_{j=1}^{n} X_j^{a_j}$$

where the $c(a_1, \dots, a_n)$'s are constant coefficients. This is a multilinear polynomial on $X_1, \dots, X_n$, cf. Equation (4). An important finding in [7] is that the coefficient vector $\vec{c} = [c(0, \dots, 0), \dots, c(1, \dots, 1)]^T$ is the spectrum vector $\vec{S}$ specified by Equation (8).

Example 2. Consider an XOR gate, which serves as a multiplier in IBP format (see Example 1). Its original truth table vector is $[0\ 1\ 1\ 0]^{\mathrm{T}}$. Replacing 0s and 1s by $+1$s and -1s, we get the vector $\vec{T} = [+1\ -1\ -1\ +1]^{\mathrm{T}}$. Applying Equation (8) to perform the Fourier transform yields the spectrum vector

$$\vec{S} = \frac{1}{2^2}\begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}\begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

This again shows that the stochastic function of XOR is IBP multiplication. □

Based on the relation between spectral transforms and stochastic circuits, a method to synthesize a stochastic circuit for a target function $\vec{S}$ is proposed in [7]. The basic idea is to apply the inverse Fourier transform $\vec{T} = H_n\vec{S}$ to obtain the vector $\vec{T}$. However, this vector may contain entries that are neither $+1$ nor -1, implying that $\vec{S}$ does not correspond to a Boolean function. For example, consider the scaled addition function $1/2(X_1 + X_2)$. Its $\vec{S}$ (coefficient) vector is $[0\ 1/2\ 1/2\ 0]^{\mathrm{T}}$, and the inverse Fourier transform $\vec{T} = H_2\vec{S}$ yields $\vec{T} = [1\ 0\ 0\ -1]^{\mathrm{T}}$, which contains the non-Boolean element zero. This problem is implicitly resolved in the standard MUX-based scaled adder (Figure 5) which has a third input r that introduces the constant probability 0.5.

In general, an entry $-1 < q < 1$ in the $\vec{T}$ vector corresponds to an SN of constant probability $(1 - q)/2$. STRAUSS employs extra SNs of probability 0.5 to generate these SNs, since a probability of 0.5 can be easily obtained from an LFSR. A heuristic method is introduced to synthesize a low-cost circuit to produce multiple constant probabilities simultaneously.

A synthesis problem similar to that of [7] is addressed in [101]. The authors first analyze the stochastic behavior of a general combinational circuit whose inputs comprise n variable SNs $X_1, \dots, X_n$ and m constant input SNs $R_1, \dots, R_m$ of value 0.5, as shown in Figure 10. If the Boolean function of the combinational circuit is $f(x_1, \dots x_n, r_1, \dots, r_m)$, then the stochastic circuit in Figure 10 realizes a polynomial of the form

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

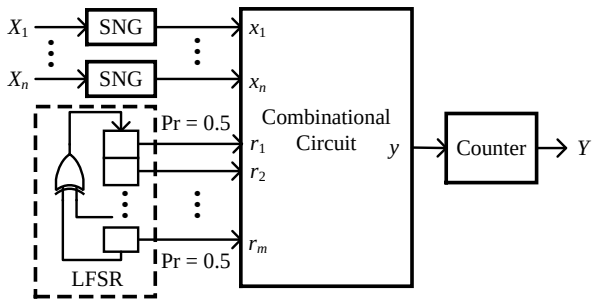


Figure 10. The general form of a fixed stochastic circuit based on a combinational circuit.

$$F(X_1, \dots, X_n) = \sum_{\substack{(a_1,\dots,a_n) \\ \in\{0,1\}^n}} \frac{g(a_1, \dots, a_n)}{2^m} \prod_{j=1}^{n} X_j^{a_j} (1 - X_j)^{1-a_j} \quad (9)$$

In this equation, for any $(a_1, \dots, a_n) \in \{0,1\}^n$, $g(a_1, \dots, a_n)$ denotes the *weight* of the Boolean function $f(a_1, \dots, a_n, r_1, \dots, r_m)$ on $r, \dots r_m$, i.e., the number of input vectors $(b_1, \dots, b_m) \in \{0,1\}^m$ such that $f(a_1, \dots, a_n, b_1, \dots, b_m) = 1$.

Example 3. Consider the case where the combinational circuit in Figure 10 is a MUX, with x_1 and x_2 as its data inputs and r_1 as its select input. Then, the circuit's Boolean function is $f(x_1, x_2, r_1) = (x_1 \wedge \overline{r_1}) \vee (x_2 \wedge r_1)$. We have $f(0,0,r_1) = 0$, $f(0,1,r_1) = r_1$, $f(1,0,r_1) = \overline{r_1}$, and $f(1,1,r_1) = 1$. Correspondingly, we have $g(0,0) = 0$, $g(0,1) = g(1,0) = 1$ and $g(1,1) = 2$. According to Equation (9), the circuit's stochastic function is

$$F(X_1, X_2) = \frac{1}{2}(1 - X_1)X_2 + \frac{1}{2}X_1(1 - X_2) + \frac{2}{2}X_1X_2 = 1/2(X_1 + X_2)$$

This again shows that the stochastic function of MUX is a scaled addition. □

A synthesis method is further proposed in [101] to realize a general polynomial. It first converts the target to a multilinear polynomial. Then, it transforms the multilinear polynomial to a polynomial of the form shown in Equation (9). This transformation is unique and can be easily obtained. After that, the problem reduces to finding an optimal Boolean function $f^*(x_1, \dots, x_n, r_1, \dots, r_m)$ such that for each $(a_1, \dots, a_n) \in \{0,1\}^n$, the weight of $f^*(a_1, \dots, a_n, r_1, \dots, r_m)$ is equal to the value $g(a_1, \dots, a_n)$ specified by the multilinear polynomial. A greedy method is applied to find a good Boolean function. The authors also find that in synthesizing polynomials of degree more than 1, all $(a_1, \dots, a_n) \in \{0,1\}^n$ can be partitioned into a number of equivalent classes and the weight constraint can be relaxed so that the sum of the weights $f(a_1, \dots, a_n, r_1, \dots, r_m)$ over all $(a_1, \dots, a_n)$'s in each equivalence class is equal to a fixed value derived from the target polynomial. The authors of [101] exploit this freedom to further reduce the circuit cost.

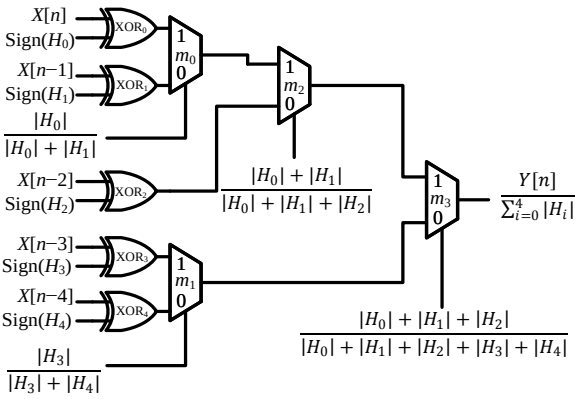


Figure 11. Stochastic implementation of a 5-tap FIR filter with an uneven-weighted MUX tree.

IV. APPLICATIONS

Stochastic computing has been applied to a variety of application domains, including artificial neural networks (ANNs) [12][14][15][17][24][39][46][93][95], control systems [59], [100], reliability estimation [35], data mining [21], digital signal processing (DSP) [4][18][40][48][50][54][83], and decoding of modern error-correcting codes [26][30][47][63][85][86][89][90][96][97]. Most of these applications are characterized by the need of a large amount of arithmetic computation, which can leverage the simple circuitry provided by SC. They also have low precision requirements for the final results, which can avoid the use of the excessively long SNs to represent data values. In this section, we review four important applications for which SC has had some success: filter design, image processing, LDPC decoding, and artificial neural networks.

A. Filter design

The design of finite impulse response (FIR) filters is considered in [18][36]. A general M-tap FIR filter computes an output based on the M most recent inputs as follows:

$$Y[n] = H_0X[n] + H_1X[n-1] + \cdots + H_{M-1}X[n-M+1] \quad (10)$$

where $X[n]$ is the input signal, $Y[n]$ is the output signal, and H_i is the filter coefficient. The FIR filter thus computes the inner product of two vectors, cf. Equation (2). A conventional binary implementation of Equation (10) requires M multipliers and $M-1$ adders, which has high hardware complexity. SC-based designs can potentially mitigate this problem.

Since the values of H, X, and Y may be negative, bipolar SNs are used to encode them. A straightforward way to implement Equation (10) uses M XNOR gates for multiplications and an M-to-1 MUX for additions. However, this implementation has the problem that the output of the MUX is $1/M$ times the desired output. Such downscaling causes severe accuracy loss when M is large.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

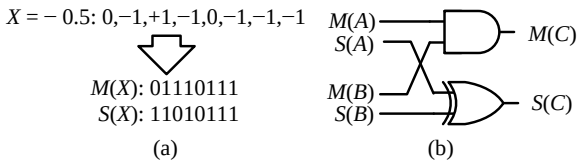


Figure 12. Two-line stochastic encoding: (a) An example of encoding the value −0.5; (b) Multiplier for the encoding.

To address the foregoing problem, a stochastic design based on an uneven-weighted MUX tree has been proposed [18][36]. Figure 11 shows such a design for a 5-tap FIR filter. The input $\mathrm{Sign}(H_i)$ is a stream of bits, each equal to the sign bit of H_i in its 2's complement binary representation. The probability for the select input of each MUX is shown in the figure. The output probability of the design is $Y[n]/\sum_{i=0}^{4}|H_i|$. In the general case, the output probability of an uneven-weighted MUX tree is $Y[n]/\sum_{i=0}^{M-1}|H_i|$. Note that the scaling factor is reduced to $\sum_{i=0}^{M-1}|H_i| \le M$. In the case where $\sum_{i=0}^{M-1}|H_i| < 1$, the proposed design will even scale up the result.

Although the datapath of the stochastic FIR filter consists of just a few logic gates as shown in Figure 11, the interface SNGs (not shown) may occupy a large area, offsetting the potential area benefit brought by the simple datapath. To further reduce the area of SNGs, techniques of sharing the RNSs used in the SNGs and circularly shifting the outputs of the RNS to generate multiple random numbers with low correlation are proposed in [36].

Area-efficient stochastic designs for the discrete Fourier transform (DFT) and the fast Fourier transform (FFT), which are important transformation techniques between the time and frequency domains, are described in [99]. An M-point DFT for discrete signals $X[n]$ $(n = 0,1,\dots,M-1)$ computes the frequency domain values $Y[k]$ $(k = 0,1,\dots,M-1)$ as follows:

$$Y[k] = \sum_{n=0}^{M-1} X[n]W_M^{kN}$$

where $W_M = e^{-j(2\pi/M)}$. The FFT is an efficient way to realize the DFT by using a butterfly architecture [70].

The basic DFT computation resembles that of an FIR filter. Although the technique of the uneven-weighted MUX tree can be applied [98], the accuracy of the result degrades as the number of points becomes larger due to the growing scaling factor. To address this problem, the work in [99] proposes a scaling-free stochastic adder based on a two-line stochastic encoding scheme [94]. This encoding represents a value in the interval $[-1,1]$ by a magnitude stream $M(X)$ and a sign stream $S(X)$. Figure 12a shows an example of encoding the value -0.5. Indeed, this encoding can be viewed as employing a ternary stochastic stream $X = x_1x_2 \dots x_N$ with each $x_i \in \{-1,0,1\}$. The magnitude and the sign of x_i are represented by the i-th bit in the magnitude stream and the sign stream, respectively. If the sign bit is 0 (1), the value is positive (negative). Figure 12b shows the multiplier for this encoding. Experimental results indicate that using the stochastic multiplier and the special stochastic adder to implement DFT/FFT can achieve much higher accuracy than an implementation based on the uneven-weighted MUX tree when the number of points M is large.

The design of infinite impulse response (IIR) filters is considered in [53][54][72]. Compared to FIR filters, the implementation of IIR filters using SC is more challenging. The main difficulty is the feedback loop in the IIR filter, which causes correlation in the stochastic bit-streams. However, the correct computation of SC usually requires the independence of the stochastic bit-streams. To address this problem, Liu and Parhi [54] propose transforming the IIR filter into a lattice structure via the Schur algorithm [88]. The benefit of such a lattice structure is that its states are orthogonal and hence, are uncorrelated, which makes the design suitable for stochastic implementation. To reduce error due to the state overflow (where the state value may be outside of the range $[-1, 1]$ of the bipolar stochastic representation), the authors further propose a scaling method that derives a normalized lattice structure as the implementation target.

B. Image processing

A DSP application that is well-suited to SC is image processing [4][50][67]. It can exploit the massive parallelism provided by simple stochastic circuits, because many image-processing operations are applied pixel-wise or block-wise across an entire image [33]. Also, long SNs are not required for image-processing applications, because the precision demands are low; in many cases, 8-bit precision is enough.

Li et al. [50] propose stochastic implementations for five image processing tasks: edge detection, median filter-based noise reduction, image contrast stretching, frame difference-based image segmentation, and kernel density estimation (KDE)-based image segmentation. Their designs introduce some novel SC elements based on sequential logic. All the designs show smaller area than their conventional counterparts. The reduction in area is greatest for KDE-based image segmentation, due to its high computational complexity. This work demonstrates that stochastic designs are advantageous for relatively complicated computations.

Najafi and Salehi [67] apply SC to a local image thresholding algorithm called the Sauvola method [87]. Image thresholding is an important step in optical character recognition. It selects a threshold and uses that threshold to determine whether a pixel should be set to 1 (background) or 0 (foreground). The Sauvola method determines the threshold for each pixel in an image and involves calculating product, sum, mean, square, absolute difference, and square root. All these operations can be realized efficiently by SC units.

Improved stochastic designs for several image-processing applications were also proposed in [4]. An example is real-time edge detection. The authors consider the Robert cross operator, which takes an input image and produces an output image with edges highlighted. Let $X_{i,j}$ and $Z_{i,j}$ denote the pixel values at row i and column j in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

input and the output images, respectively. The operator calculates $Z_{i,j}$ in the following way:

$$Z_{i,j} = 0.5\left(\left|X_{i,j} - X_{i+1,j+1}\right| + \left|X_{i,j+1} - X_{i+1,j}\right|\right) \quad (11)$$

A stochastic implementation of Equation (11) is shown in Figure 13a. It consists of only two XOR gates and one MUX. By deliberately correlating the two input SNs of an XOR gate so they have the maximum overlap of 0s and 1s, the XOR computes the absolute difference between the two input SNs [5]. The MUX further performs a scaled addition on two absolute differences. In contrast, a conventional implementation of Equation (11) on BNs is much more complicated, as suggested by Figure 13b; it has two subtractors, two absolute value calculators, and an adder.

Although a stochastic implementation often needs a large number of clock cycles to obtain the final result, the critical path delay of the stochastic implementation is much smaller than a conventional implementation's due to the simplicity of the stochastic circuit. For instance, the overall delay of the circuit of Figure 13a is only 3× higher than the delay of its binary counterpart (Figure 13b).

Another benefit of a stochastic implementation is its error tolerance. Figure 14 visually demonstrates this advantage by comparing the stochastic implementation of edge detection with conventional binary implementations for different levels of noise injected into the input sensor [4]. As shown in the first row of Figure 14, when the noise level is 10% to 20%, the conventional design generates useless outputs. In contrast, the SC implementation in the second row is almost unaffected by noise and is able to detect the edges even at a noise level of 20%.

C. Decoding Error-Correcting Codes

One successful application of SC is the decoding of certain modern error-correcting codes (ECCs). Researchers have proposed stochastic decoder designs for several ECCs, such as turbo code [26], polar code [96][97], binary low-density parity-check (LDPC) codes [30][47][89][90], and non-binary LDPC codes [85][86].

The earliest stochastic decoder was proposed for binary LDPC codes (for simplicity, hereafter referred to as LDPC codes), which have very efficient decoding performance that approaches the Shannon capacity limit [81]. They have been adopted in several recent digital communication standards, such as the DVB-S2, the IEEE 802.16e (WiMAX), and the IEEE 802.11n (WiFi) standards.

A binary LDPC code is characterized by a bipartite factor graph consisting of two groups of nodes: variable nodes (VNs) and parity-check nodes (PNs). A widely-used method to decode an LDPC code applies the sum-product algorithm (SPA) to the factor graph. The SPA iteratively passes a probability value, which represents the belief that a bit in the code block is 1, from a VN to a connected PN, or vice versa. The codeword is determined by comparing the final probabilities against a threshold.

The major computation in the decoder involves the following two operations on probabilities:

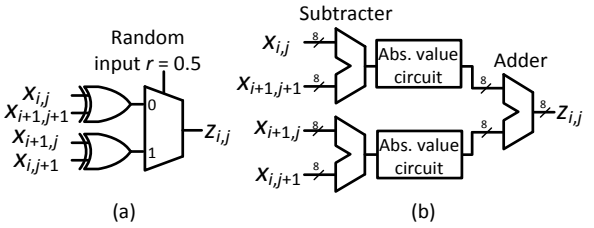


Figure 13. Two implementations of Roberts cross operator: (a) stochastic, and (b) conventional.

$$p_C = p_A(1 - p_B) + p_B(1 - p_A) \quad (12)$$

$$p_Z = \frac{p_X p_Y}{p_X p_Y + (1 - p_X)(1 - p_Y)} \quad (13)$$

Binary implementation of Equations (12) and (13) requires complicated arithmetic circuits, such as adders, multipliers, and dividers. To alleviate this problem, Gaudet and Rapley proposed a stochastic implementation of LDPC decoding in which Equations (12) and (13) are realized efficiently by the circuits in Figure 4a and 4b, respectively [30].

Besides reducing the area of the processing units, SC also reduces routing area. In a conventional binary implementation, the communication of probability values of precision k between two nodes requires k wires connecting the two nodes, which leads to a large routing area. However, with SC, due to its bit-serial nature, communication between two nodes only requires a single wire. Another benefit of SC is its support of an asynchronous pipeline. In SN representation, bit order does not matter, so we do not require the input of the PNs and VNs to be the output bits of the immediately previous cycle. This allows different edges to use different numbers of pipeline stages, thus increasing the clock frequency and throughput [89].

To improve the SPA convergence rate, the authors of [89] add a module called edge memory (EM) to each edge in the factor graph. Since one EM is assigned to each edge, the hardware usage of EMs can be large. To further reduce this hardware cost, Tehrani et al. [90] introduce a module called a majority-based tracking forecast memory (MTFM), which is assigned to each VN. This method has been integrated into a fully parallel stochastic decoder ASIC that decodes the (2048, 1723) LDPC code from the IEEE 802.3an (10GBASE-T) standard [90]. This decoder turns out to be one of the most area-efficient fully parallel LDPC decoders.

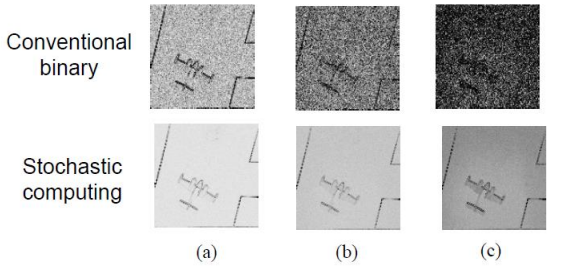


Figure 14. Edge-detection performance for two implementation methods with noise levels of (a) 5%, (b) 10% and (c) 20%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Stochastic LDPC decoders essentially implement the belief propagation algorithm [73]. This fundamental approach can also be used to decode other ECCs, such as polar codes and non-binary LDPC codes. Given their algorithm-level similarity, researchers have proposed SC-based decoders for these codes [85][86][96][97]. For example, to resolve the slow convergence problem of a pure stochastic decoder for non-binary LDPC codes, a way of mixing the binary computation and stochastic computation units has been proposed [86]. A technique of splitting and shuffling stochastic bit-streams is described in [97] to simultaneously mitigate the costs of long stochastic bit-streams and re-randomization of a stochastic decoder for polar codes.

D. Artificial neural networks

Artificial neural networks (ANNs), mimicking aspects of biological neural networks, are an early application of SC [14][15][24][68][93]. Only recently, with advances in machine learning algorithms and computer hardware technology, have they found commercial success in applications such as computer vision and speech recognition [45]. ANNs are usually implemented in software on warehouse-scale computing platforms, which are extremely costly in size and energy needs. These shortcomings have stimulated renewed interest in using SC in ANNs [10][11][41][46][80]. Furthermore, many classification tasks such as ANNs do not require high accuracy; it suffices that their classification decisions be correct most of the time [51]. Hence, SC's drawbacks of low precision and stochastic variability are well-tolerated in ANN applications.

A widely used type of ANN is the feed-forward network shown in Figure 15 [37]. It is composed of an input layer, several hidden layers, and an output layer. A node in the network is referred to as a *neuron*. Each neuron in a hidden or an output layer is connected to a number of neurons in the previous layer via weighted edges. The output 0 (inactive) or 1 (active) of a neuron is determined by applying an activation function to the weighted sum of its inputs. For example, the output of the neuron Y_1 in Figure 15 is given by

$$Y_1 = F\left(\sum_{i=1}^{n} W_i X_i\right) \tag{14}$$

where X_i is the signal produced by the i-th input neuron of Y_1, W_i is the weight of the edge from X_i to Y_1, and $F(Z)$ is the activation function. A frequent choice for F is the sigmoid function defined by

$$F(Z) = \frac{1}{1 + e^{-\beta Z}}$$

where β is the slope parameter.

A key problem in ANN design is the addition of a large number of items supplied to a neuron; a similar problem occurs in FIR filters with a large number of taps. The straightforward use of MUX-based adders to perform the scaled addition is not a good solution, because the scaling factor is proportional to the number of a neuron's connections. When rescaling the final MUX output, even a very small error due to stochastic variation may be enlarged significantly. To address this problem, Li et al. [51] revive the old idea of using an OR gate as an adder [29]. OR combines two unipolar SNs X and Y as follows:

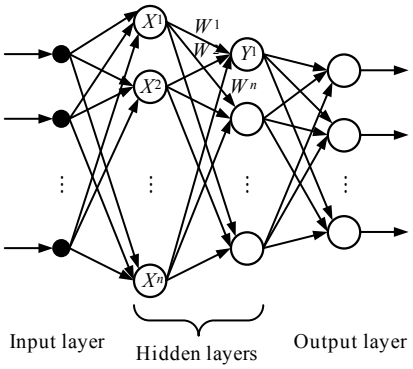


Figure 15. A typical feed-forward network structure

$$Z = X + Y - XY$$

This is not strictly addition, but when either $X \ll 1$ or $Y \ll 1$, the output Z is approximately the sum of the two inputs. To make the inputs close to zero, the authors of [51] apply a moderate scaling factor to scale down the inputs.

Some other studies have addressed the addition problem with new stochastic data representations [11][17]. In [17], an encoding scheme called extended stochastic logic (ESL) is proposed which uses two bipolar SNs X and Y to represent the number X/Y. ESL addition has the advantage of being exact, with no scaling factor. Moreover ESL encoding allows easy implementation of multiplication, division, and the sigmoid function. Together, these operations lead to an efficient neuron design.

Ardakani et al. have proposed the concept of integer stochastic number (ISN) in which a sequence of random integers represents a value equal to the mean of these integers [11]. For example, the sequence 2,0,4,1 represents 7/4. With this encoding, any real number can be represented without prior scaling. The weights in an ANN, which can lie outside the range $[-1,1]$, do not need to be scaled. The addition of two ISNs uses a conventional binary adder, which makes the sum exact. Multiplication of two ISNs requires a conventional binary multiplier, which is expensive. Fortunately, in the ANN implementation proposed in [11], one input to the multiplier, which corresponds to the neuron signal, is always a binary SN. Then, the conventional multiplier is reduced to several AND gates. The sigmoid activation function is implemented by a counter similar to that in [14]. Although the hardware cost of the ISN implementation is larger than that of a binary stochastic implementation, the former has much lower latency and energy consumption. Compared to the conventional binary design, the ISN design produces fewer misclassification errors, while reducing energy and area cost substantially.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Another recent work [41] proposes two new ways to design ANNs with SC. The first considers training in the design phase to make the network friendly to a stochastic implementation. The authors observe that weights close to zero, which correspond to (bipolar) SNs of probability 0.5, contribute the most to random fluctuation errors. Therefore, they propose to iteratively drop near-zero weights and then re-train the network to derive a network with high classification accuracy but no near-zero weights. The second technique is to exploit the progressive precision property of SC. The authors observe that most of the input data can be classified easily because they are far from the decision boundary. For these input data, computation with low-precision SNs is enough to obtain the correct results. Based on this, the authors devise an early decision termination (EDT) strategy which adaptively selects the number of bits to use in the computation depending on the difficulty of the classification tasks. The resulting design has a misclassification error rate very close to the conventional implementation. Furthermore, EDT reduces energy consumption with a slight increase in misclassification errors.

Efficient stochastic implementation of convolutional neural networks (CNNs), a special type of feed-forward ANN, is the focus of [10]. In a CNN, the signals of all the neurons in a layer are obtained by first convolving a kernel with the signals in the input layer—a special kind of filtering operation—and then applying an activation function. The size of the kernel is much less than that of the input layer, which means a neuron signal only depends on a subset of the neurons in its input layer. CNNs have been successfully applied to machine learning tasks such as face and speech recognition. A major contribution of [10] is an efficient stochastic implementation of the convolution operation. Unlike SC that uses SNs to encode real values, the proposed method uses the probability mass function of a random variable to represent an array of real values. An efficient implementation of convolution is developed based on this representation. Furthermore, a few other techniques are introduced in [10] to implement other components of a CNN, such as the pooling and nonlinear activation components. Compared to a conventional binary CNN, the proposed SC implementation achieves large improvements in performance and power efficiency.

Efficient stochastic implementation of CNNs has also been studied by Ren et al. [80]. They perform a comprehensive study of SC operators and how they should be optimized to obtain energy-efficient CNNs. Ren at al. adopt the approximate accumulative parallel counter (APC) of [42] to add a large number of input stochastic bit-streams. Kim et al. report that the approximate APC has negligible accuracy loss and is about 40% smaller than the exact APC [42].

V. Discussion

Since the turn of the present century, significant progress has been made in developing the theory and application of stochastic computing. New questions and challenges have emerged, many of which still need to be addressed. With the notable exception of LDPC decoder chips, few large-scale SC-based systems have actually been built and evaluated. As a result, real-world experience with SC is limited, making it likely that many practical aspects of SC such as its true design costs, run-time performance, and energy consumption are not yet fully appreciated. Small-scale theoretical and simulation-based studies are fairly plentiful, but they often consider only a narrow range of issues under restrictive assumptions.

A. Conclusions

Based on what is now known, we can draw some general conclusions about what SC is, and is not, good for.

Precision and errors: SC is inherently approximate and inexact. Its probability-based and redundant data encoding makes it a relatively low-precision technology, but one that is very tolerant of errors. It has been successfully applied to image-processing using 256-bit stochastic numbers (SNs), which correspond roughly to 8-bit (fixed-point) BNs. SC is unsuited to the very high 32- or 64-bit precision error-sensitive calculations that are the domain of BNs and binary computing (BC). This is seen in the random noise-like fluctuations that are normal to SNs, in the way SNs are squeezed into the unit interval producing errors near the boundaries, and in the fact that SNs grow in length exponentially faster than BNs as the desired level of precision increases. Also the stochastic encoding of numbers does not provide a dynamic range, similar to the one provided by floating point numbers.

On the other hand, low precision and error tolerance have definite advantages. They have evolved in the natural world for use by the human brain and nervous system. Similar features are increasingly seen in artificial constructs like deep learning networks that aim to mimic brain operations [23]. Thus it seems pointless to compare SC and BC purely on the basis of precision or precision-related costs alone [1][58].

Finally, we observe that while BC circuits have fixed precision, SC circuits have the advantage of inherently variable precision in their bit-streams. Moreover, the bit-streams can be endowed with progressive precision where accuracy improves monotonically as computation proceeds, as has been demonstrated for some image-processing tasks [4]. If a variable precision cannot be exploited, a simple bit-reduction technique in BC often provides better energy efficiency over SC. As reported in recent work, with fixed precision, SC becomes worse for designs above 6 bits of precision [1][46].

Area-related costs: The use of tiny circuits for operations like multiplication and addition remains SC's strongest selling point. A stochastic multiplier contains orders-of-magnitude fewer gates than a typical BC mul-

tiplier. However, many arithmetic operations including multiplication require uncorrelated inputs to function correctly. This implies a need for randomization or decorrelation circuits incorporating many independent random sources or phase-shifting delay elements (isolators), whose combined area can easily exceed that of the arithmetic logic [92]. The low-power benefit of stochastic components must be weighed against the additional power consumed by their randomization circuitry.

Speed-related costs: Perhaps the clearest drawback of SC is its need for long, multicycle SNs to generate satisfactory results. This leads to long run-times, which are compensated for, in part, by the fact that the clock cycles tend to be very short. Parallel processing, where long bitstreams are partitioned into segments that are processed in parallel is a speed-up possibility that has often been proposed, but not be studied much [21]. The same can be said of progressive precision.

Small stochastic circuits have relatively low power consumption. However, since *energy = power × time*, the longer run-times of stochastic circuits can lead to higher energy use than their BC counterparts [62]. Reducing energy usage is therefore emerging as a significant challenge for SC.

Design issues: Until recently, SC design was an ad hoc process with little theory to guide it. However, thanks to a deeper understanding of the properties of stochastic functions and circuits, several general synthesis techniques have been developed, which can variously be classified as reconfigurable or fixed, and combinational or sequential [7][49][76]. The new understanding has revealed unexpected and novel solutions to some of SC's basic problems.

For example, it has come to be recognized that different circuits realizing different logic functions can have the same stochastic behavior [19]. Far from just being the enemy, correlation can sometimes be harnessed as a design resource to reduce circuit size and cost, as the edge detectors of Figure 13a vividly illustrate. Common circuits like the MUX-based scaled adder turn out to have correlation insensitivity that enables RNSs to be removed or shared; see Figure 8. A fundamental redesign of the SC scaled adder itself is shown in Figure 16, which converts it from a three-input to a two-input element, while improving both its accuracy and correlation properties [46]. Despite such progress, many questions concerning the properties of stochastic circuits that influence design requirements, remain unanswered.

Circuit level aspects: Since SC employs digital components, conventional digital design process (synthesis, automatic placement and routing, timing closure, etc.) have been used to implement SC ASIC and FPGA-based designs. However, as discussed in this paper, SC shares similarities with analog circuits, so the digital design aspects of it may differ from conventional digital circuits.

Various circuit-level aspects of SC designs have been investigated very recently as a means of improving SC's energy efficiency [9][65]. They suggest that SC circuits are probably not optimal if they are designed using standard digital design tools. Najafi et al. [65] demonstrate that SC circuits do not need clock trees. Eliminating the clock tree significantly reduces the energy consumption of the circuit. In fact, employing analog components, rather than digital, can lead to significant energy savings [66]. One example is the use of analog integrators, instead of counters, to collect the computation results.

Alaghi et al. [9] have investigated a different circuit-level aspect of SC. They show that SC's inherent error-tolerance makes it robust against errors caused by voltage overscaling. Voltage overscaling, i.e., the process of reducing the power consumption of the circuit without reducing the frequency, usually leads to critical path timing failures and catastrophic errors in regular digital circuits. However, timing violations in SC manifest as extra or missing pulses on the output SN. The extra and missing pulses tend to cancel each other out, leading to negligible error. An optimization method is described in [9] that balances the circuit paths to guarantee maximum error cancellation. It is worth noting that the observations of [9] have been confirmed through a fabricated chip.

The new results suggest that circuit-level aspects of SC must be considered at design time, as they provide valuable sources of energy saving. As a result, SC circuits should be either manually designed [64] or new CAD tools must be provided [9].

Applications: As discussed in detail in Section IV, SC has been successfully applied to a relatively small range of applications, notably filter design, image processing, LDPC decoding, and ANN design. A common aspect of these applications is a need for very large numbers of low-precision arithmetic operations, which can take advantage of the small size of stochastic circuits. They also typically have a high degree of error tolerance. It is worth noting that current trends in embedded and cloud computing, e.g., the increasing use of fast on-line image recognition and machine learning techniques by smartphones and automobiles, call for algorithms for which SC is well suited. The so-called Internet of Things is likely to create a big demand for tiny, ultra-low-cost processing circuits with many of the characteristics of SC.

B. Future Challenges

The issues covered in the preceding sections are by no means completely understood, and many of them deserve further study. There are however, other important topics that have received little recognition or attention; we briefly discuss four of them next.

Accuracy management: In conventional BC, the accuracy goals of a new design, such as its precision level and error bounds, are determined a priori during the specification phase. As the design progresses and prototypes are produced, fine tuning may be needed to ensure that these goals and related performance requirements

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2778107, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

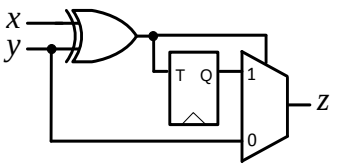


Figure 16. An SC adder built around a toggle flip-flop [46].

are actually met. This approach is much harder to apply to stochastic circuit design. Interacting factors including bit-stream length, RNS placement, and correlation can drastically affect accuracy in complex ways. For example, it is pointed out in [92] that cascading two well-designed squarer circuits, each computing X^2, does not implement X^4, as might be expected; instead the cascaded circuit implements X^3.

Because of hard-to-predict behavior like this, extensive simulation is almost always used to determine the basic accuracy limits and error sensitivities of a new SC design. SC projects often have a cut-and-try flavor which involves multiple design-and-simulate iterations that resemble design-space exploration rather than the fine tuning of well-founded designs. It would be very useful to be able to incorporate into an SC design flow an "accuracy manager" that can comprehend and automatically adjust the relations among the design parameters affecting accuracy. A first step in this direction can be found in [69], while automatic decorrelation methods to enhance accuracy are addressed in [92].

Design optimization. Despite recent advances in SC synthesis, a number of open problems remain. It is now recognized that many different Boolean functions can realize the same computation [7][19][101]. For instance, the Boolean functions $f_1(x_1, x_2, r_1) = (x_1 \wedge \overline{r_1}) \vee (x_2 \wedge r_1)$ and $f_2(x_1, x_2, r_1) = (x_1 \wedge r_1) \vee (x_2 \wedge r_1) \vee (x_1 \wedge x_2)$ both realize the same stochastic addition function $F(X_1, X_2) = 1/2(X_1 + X_2)$. An open question is: Among numerous Boolean functions that have the same stochatic behavior, how can we find an optimal one? All the previous work on synthesis assumes that the input SNs are independent. However, as shown in [5][20], sometimes taking the advantage of correlated input SNs helps reduce circuit area. Another open problem is how to develop a synthesis approach that takes correlation into consideration and exploits it when necessary. Finally, most work on synthesis has been restricted to combinational logic. This has led to a deeper understanding of combinational synthesis, for example, the existence of stochastic equivalence classes [19]. In contrast, far fewer theoretical advances have been made in understanding sequential stochastic design. How to synthesize optimal stochastic circuits based on sequential logic therefore remains an unsolved problem.

***Energy harvesting*:** With the development of the Internet-of-Things, many future computing systems are expected to be powered by energy harvested from the environment. The potential energy sources include solar energy, as well as ambient RF, motion, and temperature energy [56]. A difficulty with such energy sources is that they tend to be highly variable and unstable. This can significantly degrade the performance of BC systems. SC, on the other hand, has strong tolerance of errors caused by the random fluctuation of the supply voltage [9]. A problem for SC is the potentially large energy needs of its many randomness sources for number conversion and decorrelation. This may be solved by emerging technologies that have naturally stochastic behaviors. For example, very compact random sources can be constructed from memristors. Moreover, a single memristor source can supply independent random bit-streams to multiple destinations simultaneously [43].

Biomedical devices: It was remarked in Sec. 1 that stochastic bit-streams can mimic the low-power spike trains used for communication in natural neural networks; see Figure 3. This has suggested the use of SC in implantable devices such as retinal implants to treat the visually impaired [4]. Retinal implants are ICs that are placed directly on the retina, sense visual images in the form of pixel arrays, and convert the pixel information into bit-streams that are sent directly to the brain via the optic nerve where they produce flashes of light that the brain can be trained to interpret. With better understanding of the information coding and data processing involved, SC may be found applicable to other applications that involve interfacing stochastic circuits with natural neural networks. A particular advantage of SC in this domain is its very low power consumption which is necessary to avoid heat damage to human tissue. So far, however, we know of no current work to incorporate SC into implantable medical devices.

REFERENCES

[1] J. M. de Aguiar and S. P. Khatri, "Exploring the Viability of Stochastic Computing," *Proc. Intl. Conf. Computer Design (ICCD)*, pp. 391-394, 2015.

[2] A. Alaghi and J.P. Hayes, "A Spectral Transform Approach to Stochastic Circuits," *Proc. Intl. Conf. Computer Design (ICCD)*, pp. 315–312, 2012.

[3] A. Alaghi and J.P. Hayes, "Survey of Stochastic Computing," *ACM Trans. Embed. Comp. Syst.*, vol. 12, no. 2s, pp. 92:1–92:19, May 2013.

[4] A. Alaghi, C. Li and J.P. Hayes, "Stochastic Circuits for Real-time Image-processing Applications," *Proc. Design Autom. Conf.* (DAC), article 136, 6p, 2013.

[5] A. Alaghi and J.P. Hayes, "Exploiting Correlation in Stochastic Circuit Design," *Proc. Intl Conf. on Computer Design* (ICCD), pp. 39–46, Oct. 2013.

[6] A. Alaghi and J. P. Hayes, "Fast and Accurate Computation Using Stochastic Circuits," *Proc. Design, Automation, and Test in Europe Conf.* (DATE), pp. 1-4, 2014.

[7] A. Alaghi and J.P. Hayes, "STRAUSS: Spectral Transform Use in Stochastic Circuit Synthesis," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 34, pp. 1770-1783, 2015.

[8] A. Alaghi and John P. Hayes, "Dimension Reduction in Statistical Simulation of Digital Circuits," *Proc. Symp. on Theory of Modeling & Simulation* (*TMS-DEVS*), pp. 1-8, 2015.

[9] A. Alaghi, W. T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Optimizing Stochastic Circuits for Accuracy-Energy Tradeoffs," *Proc. ICCAD*, pp. 178-185, 2015.

[10] M. Alawad and M. Lin, "Stochastic-Based Deep Convolutional Networks with Reconfigurable Logic Fabric," *IEEE. Trans. Multi-Scale Comp. Syst.* (to appear).

[11] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu and W.J. Gross, "VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing," *IEEE. Trans. VLSI*, 2017 (IEEE Early Access Article).

[12] S.L. Bade and B.L. Hutchings, "FPGA-Based Stochastic Neural Networks-Implementation," *Proc. IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 189-198, 1994.

[13] D. Braendler, T. Hendtlass and P. O'Donoghue, "Deterministic Bit-Stream Digital Neurons," *IEEE Trans. Neural Nets*., vol. 13, pp. 1514-1525, Nov. 2002.

[14] B.D. Brown and H.C. Card, "Stochastic Neural Computation I: Computational Elements," *IEEE Trans. Comp*., vol. 50, pp. 891-905, 2001.

[15] B.D. Brown and H.C. Card, "Stochastic Neural Computation II: Soft Competitive Learning," *IEEE Trans. Comp.*, vol. 50, pp. 906-920, 2001.

[16] A.W. Burks, H.H. Goldstine and J. Von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computer Instrument," Institute for Advanced Study, Princeton, Jan. 1946.

[17] V. Canals, A. Morro, A. Oliver, M.L. Alomar and J.L. Rossello, "A New Stochastic Computing Methodology for Efficient Neural Network Implementation," *IEEE Trans. Neural Networks and Learning Systems*, vol. 27, pp. 551-564, 2016.

[18] Y-N. Chang and K.K. Parhi, "Architectures for Digital Filters Using Stochastic Computing," *Proc. Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2697-2701, 2013.

[19] T-H. Chen and J.P. Hayes, "Equivalence among Stochastic Logic Circuits and its Application," *Proc. Design Autom. Conf. (DAC)*, pp. 131-136, 2015.

[20] T-H. Chen and J.P. Hayes, "Design of Division Circuits for Stochastic Computing," *Proc. IEEE Symp. on VLSI* (*ISVLSI*), pp. 116-121, 2016.

[21] V.K. Chippa, S. Venkataramani, K. Roy and A. Raghunathan, "StoRM: a Stochastic Recognition and Mining Processor," *Proc. Intl. Symp. Low Power Electronics and Design (ISLPED)*, pp. 39-44, 2014.

[22] S.S. Choi, S.H. Cha and C. Tappert, "A Survey of Binary Similarity and Distance Measures," *Jour. Systemics, Cybernetics and Informatics*, vol. 8, pp. 43-48, 2010.

[23] M. Courbariaux, Y. Bengio and J. P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," *Proc. Intl. Conf. Neural Information Processing Systems* (NIPS), pp. 3123-3131, 2015.

[24] J.A. Dickson, R.D. McLeod and H.C. Card, "Stochastic Arithmetic Implementations of Neural Networks with In Situ Learning," *Proc. Intl. Conf. Neural Networks*, pp. 711–716, 1993.

[25] Y. Ding, Y. Wu and W. Qian, "Generating Multiple Correlated Probabilities for MUX-based Stochastic Computing Architecture," *Proc. ICCAD*, pp. 519-526, 2014.

[26] Q. Dong, M. Arzel, C. Jego, and W. Gross, "Stochastic Decoding of Turbo Codes," *IEEE Trans. Signal Processing*, vol. 58, pp. 6421–6425, 2010.

[27] D. Fick, G. Kim, A. Wang, D. Blaauw and D. Sylvester, "Mixed-Signal Stochastic Computation Demonstrated in an Image Sensor with Integrated 2D Edge Detection and Noise Filtering", *Proc. IEEE Custom Integrated Circuits Conf.* (CICC), 2014, pp. 1–4.

[28] B.R. Gaines, "Stochastic Computing," *Proc. AFIPS Spring Joint Computer Conf*., pp. 149-156, 1967.

[29] B.R. Gaines, "Stochastic Computing Systems," *Advances in Information Systems Science*, vol. 2, J.T. Tou (ed.), Springer-Verlag, pp. 37-172, 1969.

[30] V.C. Gaudet and A.C. Rapley, "Iterative Decoding Using Stochastic Computation," *Electron. Lett*., vol. 39, pp. 299-301, 2003.

[31] W. Gerstner and W.M. Kistler, *Spiking Neuron Models*, Cambridge University Press, 2002.

[32] S.W. Golomb, *Shift Register Sequences*. Revised ed., Aegean Park Press, Laguna Hills, CA, 1982.

[33] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd ed., Prentice Hall, 2002.

[34] P. K. Gupta and R. Kumaresan, "Binary Multiplication with PN Sequences," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 603-606, 1988.

[35] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi, "A Stochastic Computational Approach for Accurate and Efficient Reliability Evaluation," *IEEE Trans. Comp.*, vol. 63, pp. 1336-1350, 2014.

[36] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, T. Inoue, "Compact and Accurate Digital Filters Based on Stochastic Computing," *IEEE Trans. Emerging Topics in Comp.*, vol. 99, pp. 1-1, 2016.

[37] A.K. Jain, J. Mao and K.M. Mohiuddin, "Artificial Neural Networks: a Tutorial," *Computer*, vol. 29, no. 3, pp. 31-44, 1996.

[38] D. Jenson and M. Riedel, "A Deterministic Approach to Stochastic Computation," *Proc. Intl. Conf. Computer-Aided Design* (ICCAD), pp. 1-8, 2016.

[39] Y. Ji, F. Ran, C. Ma and D.J. Lilja, "A Hardware Implementation of a Radial Basis Function Neural Network Using Stochastic Logic," *Proc. Design, Automation, and Test in Europe Conf.* (DATE), pp. 880–883, 2015.

[40] H. Jiang et al., "Adaptive Filter Design Using Stochastic Circuits," *Proc. IEEE Symp. on VLSI* (ISVLSI), pp. 122-127, 2016.

[41] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee and K. Choi, "Dynamic Energy-Accuracy Trade-off Using Stochastic Computing in Deep Neural Networks," *Proc. Design Autom. Conf.* (DAC), article 124, 2016.

[42] K. Kim, J. Lee, and K. Choi, "Approximate De-randomizer for Stochastic Circuits," *Proc. Intl. SoC Design Conf.*, pp. 123-124, 2015.

[43] P. Knag, W. Lu and Z. Zhang, "A Native Stochastic Computing Architecture Enabled by Memristors," *IEEE Trans. Nanotech*., vol. 13, pp. 283-293, 2014.

[44] D.E. Knuth, *The Art of Computer Programming*, Vol. 2: (2nd Ed.) Seminumerical Algorithms. Addison Wesley Longman Publishing Co., Redwood City, CA, 1998.

[45] Y. LeCun, Y. Bengio, G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436-444, 2015.

[46] V.T. Lee, A. Alaghi, J.P. Hayes, V. Sathe and L. Ceze, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," *Proc. Design, Automation and Test in Europe Conf.* (DATE), pp. 13-18, 2017.

[47] X-R. Lee, C-L. Chen, H-C. Chang and C-Y. Lee, "A 7.92 Gb/s 437.2 mW Stochastic LDPC Decoder Chip for IEEE 802.15.3c Applications," *IEEE Trans. Ccts. and Syst. I: Reg.Papers*, vol. 62. pp. 507-516, 2015.

[48] P. Li and D.J. Lilja, "Using Stochastic Computing to Implement Digital Image Processing Algorithms," *Proc. Intl. Conf. Computer Design* (ICCD), pp. 154-161, 2011.

[49] P. Li, D.J. Lilja, W. Qian, K. Bazargan and M. Riedel, "The Synthesis of Complex Arithmetic Computation on Stochastic Bit Streams Using Sequential Logic," *Proc. Intl. Conf. Computer-Aided Design* (ICCAD), pp. 480-487, 2012.

[50] P. Li, D.J. Lilja, W. Qian, K. Bazargan and M. Riedel, "Computation on Stochastic Bit Streams: Digital Image Processing Case Studies," *IEEE Trans. VLSI*, vol. 22, pp. 449-462, 2014.

[51] B. Li, M. H. Najafi, and D. J. Lilja, "Using Stochastic Computing to Reduce the Hardware Requirements for a Restricted Boltzmann Machine Classifier," *Proc. Intl. Symp. on FPGA*, pp. 36-41, 2016.

[52] P. Li, W. Qian, M. Riedel, K. Bazargan and D.J. Lilja, "The Synthesis of Linear Finite State Machine-based Stochastic Computational Elements," *Proc. Asia and South Pacific Design Automation Conf.* (ASP-DAC), pp. 757-762, 2012.

[53] Y. Liu and K. Parhi, "Architectures for Stochastic Normalized and Modified Lattice IIR Filters," *Proc. Asilomar Conf. on Signals, Systems & Computers*, pp. 1351-1381, 2015.

[54] Y. Liu and K. Parhi, "Architectures for Recursive Digital Filters Using Stochastic Computing," *IEEE Trans. Signal Processing*, vol. 64, pp. 3705–3718, 2016.

[55] G.G. Lorentz, *Bernstein Polynomials, 2nd Ed.* New York: AMS Chelsea, 1986.

[56] K. Ma, et al. "Architecture Exploration for Ambient Energy Harvesting Nonvolatile Processors," *Proc. Intl. Symp. High Performance Computer Architecture* (HPCA), pp. 526-537, 2015.

[57] W. Maass and C.M. Bishop (eds.), *Pulsed Neural Networks,* MIT Press, 1999.

[58] R. Manohar, "Comparing Stochastic and Deterministic Computing," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 119-122, 2015.

[59] S. L. T. Marin, J. M. Q. Reboul and L. G. Franquelo, "Digital Stochastic Realization of Complex Analog Controllers," *IEEE Trans. Industrial Electronics*, vol. 49, pp. 1101–1109, 2002.

[60] P. Mars and W.J. Poppelbaum, *Stochastic and Deterministic Averaging Processors*, London: Peter Peregrinus, 1981.

[61] S.-J. Min, E.-W. Lee and S.-I. Chae, "A Study on the Stochastic Computation Using the Ratio of One Pulses and Zero Pulses," *Proc. Intl. Symp. Circuits and Systems* (ISCAS), pp. 471–474, 1994.

[62] B. Moons and M. Verhelst, "Energy-Efficiency and Accuracy of Stochastic Computing Circuits in Emerging Technologies," *IEEE Jour. Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 4, pp. 475-486, 2014.

[63] A. Naderi, S. Mannor, M. Sawan and W.J. Gross, "Delayed Stochastic Decoding of LDPC Codes," *IEEE Trans. Signal Processing*, vol. 59, pp. 5617–5626, 2011.

[64] M.H. Najafi, S. Jamali-Zavareh, D.J. Lilja, M.D. Riedel, K. Bazargan, and R. Harjani, "Time-Encoded Values for Highly Efficient Stochastic Circuits," *IEEE Trans VLSI*, vol. 99, pp.1-14, 2016.

[65] M.H. Najafi, D.J. Lilja, M. Riedel and K. Bazargan, "Polysynchronous Stochastic Circuits," *Proc. Asia and South Pacific Design Automation Conf.* (ASP-DAC), pp.492-498, 2016.

[66] M.H. Najafi and D.J. Lilja, "High-Speed Stochastic Circuits Using Synchronous Analog Pulses," *Proc. Asia and South Pacific Design Automation Conf.* (ASP-DAC), 2017.

[67] M.H. Najafi and M.E. Salehi, "A Fast Fault-Tolerant Architecture for Sauvola Local Image Thresholding Algorithm Using Stochastic Computing," IEEE Trans. VLSI, vol. 24, pp. 808-812, 2016.

[68] N. Nedjah and L. de Macedo Mourelle, "Stochastic Reconfigurable Hardware For Neural Networks," *Proc. Euromicro Conf. on Digital System Design* (DSD), pp. 438-442, 2003.

[69] F. Neugebauer I. Polian and J. P. Hayes, "Framework for Quantifying and Managing Accuracy in Stochastic Circuit Design," *Proc. Design, Automation and Test in Europe Conf.* (DATE), pp. 1-6, 2017.

[70] A. V. Oppenheim, A. S. Willsky and S. H. Nawab, *Signals & Systems* (2nd Ed.). Prentice-Hall, Upper Saddle River, NJ, 1996.

[71] B. Parhami and C.-H. Yeh, "Accumulative Parallel Counters," *Proc. Asilomar Conf. Signals, Systems & Computers*, pp. 966-970, 1995.

[72] K. Parhi and Y. Liu, "Architectures for IIR Digital Filters Using Stochastic Computing," *Proc. Intl. Symp. Circuits and Systems* (ISCAS), pp. 373-376, 2014.

[73] J. Pearl, Probabilistic *Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[74] W.J. Poppelbaum, C. Afuso and J.W. Esch, J.W, "Stochastic Computing Elements and Systems," *Proc. AFIPS Fall Joint Computer Conf.*, pp. 635-644, 1967.

[75] W. Qian, *Digital yet Deliberately Random: Synthesizing Logical Computation on Stochastic Bit Streams*, PhD Dissertation, University of Minnesota, 2011.

[76] W. Qian, X. Li, M. Riedel, K. Bazargan and D.J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," *IEEE Trans. Comp.*, vol. 60, pp. 93-105, 2011.

[77] W. Qian and M. D. Riedel, "The Synthesis of Robust Polynomial Arithmetic with Stochastic Logic," *Proc. Design Autom. Conf.* (DAC), pp. 648-653, 2008.

[78] W. Qian, M.D. Riedel and I. Rosenberg, "Uniform Approximation and Bernstein Polynomials with Coefficients in the Unit Interval," in *European Jour. Combinatorics*, vol. 32, pp. 448-463, 2011.

[79] W. Qian, M.D. Riedel, H. Zhou and J. Bruck, "Transforming Probabilities with Combinational Logic," *IEEE Trans. CAD* , vol. 30, pp. 1279-1292, 2011.

[80] A. Ren, J. Li, Z. Li, C. Ding, X. Qian, Q. Qiu, B. Yuan, and Y. Wang, "SC-DCNN: Highly-Scalable Deep Convolutional Neural Network using Stochastic Computing," *Proc. Intl. Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS), pp. 405-418, 2017.

[81] T.J. Richardson and R.L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. Info. Theory*, vol. 47, pp. 599-618, 2001

[82] N. Saraf, K. Bazargan, D.J. Lilja and M. Riedel, "Stochastic Functions using Sequential Logic," *Proc. Intl. Conf. Computer Design* (ICCD), pp. 507–510, 2013.

[83] N. Saraf, K. Bazargan, D.J. Lilja and M. Riedel, "IIR Filters Using Stochastic Arithmetic," *Proc. Design, Automation and Test in Europe Conf.* (DATE), pp. 1–6, 2014.

[84] N. Saraf and K. Bazargan, "Polynomial arithmetic using sequential stochastic logic," *Proc. Great Lakes Symposium on VLSI* (GLSVLSI), pp. 245-250, 2016.

[85] G. Sarkis and W. Gross, "Efficient Stochastic Decoding of Non-Binary LDPC Codes with Degree-Two Variable Nodes," *IEEE Communications Letters*, vol. 12, pp. 389-391, 2012.

[86] G. Sarkis, S. Hemati, S. Mannor, and W. Gross, "Stochastic Decoding of LDPC Codes over GF(q)," *IEEE Trans. Communications*, vol. 61, pp. 939-950, 2013.

[87] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recog.*, vol. 33, no. 2, pp. 225–236, 2000.

[88] I. Schur, "Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind," *Jour. Reine Angew. Math*., vol. 147, pp. 205–232, 1917.

[89] S.S. Tehrani, S. Mannor and W.J. Gross, "Fully Parallel Stochastic LDPC Decoders," *IEEE Trans. Signal Processing*, vol. 56, pp. 5692–5703, 2008.

[90] S.S. Tehrani, A. Naderi., G.-A. Kamendje, S. Hemati, S. Mannor and W.J. Gross, "Majority-Based Tracking Forecast Memories for Stochastic LDPC Decoding," *IEEE Trans. Signal Processing*, vol. 58, pp. 4883–4896, 2010.

[91] P. S. Ting and J. P. Hayes, "Stochastic Logic Realization of Matrix Operations," *Proc. Euromicro Conf. Digital System Design (DSD)*, pp. 356-364, 2014.

[92] P. S. Ting and J. P. Hayes, "Isolation-Based Decorrelation of Stochastic Circuits," *Proc. Intl. Conf. Computer Design (ICCD)*, pp. 88-95, 2016.

[93] J.E. Tomberg and K.K.K. Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms," *IEEE Jour. Solid-State Circuits*, vol. 25, pp. 1277-1286, 1990.

[94] S. L. Toral, J. M. Quero and L. G. Franquelo, "Stochastic Pulse Coded Arithmetic," *Proc. Intl. Symp. Circuits and Systems* (ISCAS), pp. 599-602, 2000.

[95] D.E. Van Den Bout and T. K. Miller, III, "A Digital Architecture Employing Stochasticism for the Simulation of Hopfield Neural Nets," *IEEE Trans. Circuits & Syst*., vol. 36, pp. 732–738, 1989.

[96] B. Yuan and K. Parhi, "Successive Cancellation Decoding of Polar Codes using Stochastic Computing," *Proc. Intl. Symp. Circuits and Systems* (ISCAS), pp. 3040-3043, 2015.

[97] B. Yuan and K. Parhi, "Belief Propagation Decoding of Polar Codes using Stochastic Computing," *Proc. Intl. Symp. Circuits and Systems* (ISCAS), pp. 157-160, 2016.

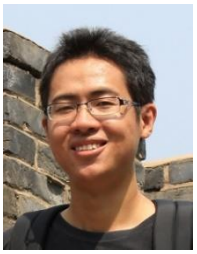
[98] B. Yuan, Y. Wang and Z. Wang, "Area-Efficient Error-Resilient Discrete Fourier Transformation Design using Stochastic Computing," *Proc. Great Lakes Symp. on VLSI* (GLSVLSI), pp. 33-38, 2016.

[99] B. Yuan, Y. Wang and Z. Wang, "Area-Efficient Scaling-Free DFT/FFT Design Using Stochastic Computing," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 63, pp. 1131-1135, 2016.

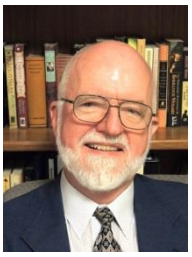
[100] D. Zhang and H. Li, "A Stochastic-based FPGA Controller for an Induction Motor Drive with Integrated Neural Network Algorithms," *IEEE Trans. Industrial Electronics*, vol. 55, pp. 551–561, 2008.

[101] Z. Zhao and W. Qian, "A General Design of Stochastic Circuit and its Synthesis," *Proc. Design, Automation and Test in Europe Conf.* (DATE), pp. 1467-1472, 2015.

Armin Alaghi (S'06–M'15) received the B.Sc. degree in electrical engineering (2006) and M.Sc. degree in computer architecture (2009) from the University of Tehran, Iran. He received his Ph.D. from the Electrical Engineering and Computer Science Department at the University of Michigan (2015). From 2005 to 2009, he was a research assistant in the Field-Programmable Gate-Array (FPGA) Lab and the Computer-Aided Design (CAD) Lab at the University of Tehran, where he worked on FPGA testing and Network-on-Chip (NoC) testing. From 2009-2015, he was with the Advanced Computer Architecture Lab (ACAL) at the University of Michigan. He is currently a research associate at the University of Washington. His research interests include digital system design, embedded systems, VLSI circuits, computer architecture and electronic design automation.



Weikang Qian (S'07–M'11) is an assistant professor in the University of Michigan-Shanghai Jiao Tong University Joint Institute at Shanghai Jiao Tong University. He received his Ph.D. degree in Electrical Engineering at the University of Minnesota in 2011 and his B.Eng. degree in Automation at Tsinghua University in 2006. His main research interests include electronic design automation and digital design for emerging technologies. His research works were nominated for the Best Paper Awards at the 2009 International Conference on Computer-Aided Design (ICCAD) and the 2016 International Workshop on Logic and Synthesis (IWLS).



John P. Hayes (S'67–M'70–SM'81–F'85–LF'10) received the B.E. degree from the National University of Ireland, Dublin, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana-Champaign, all in electrical engineering. While at the University of Illinois, he participated in the design of the ILLIAC III computer. In 1970 he joined the Operations Research Group at the Shell Benelux Computing Center in The Hague, where he worked on mathematical

programming and software development. From 1972 to 1982 he was a faculty member of the Departments of Electrical Engineering–Systems and Computer Science of the University of Southern California. Since 1982 he has been with the Electrical Engineering and Computer Science Department of the University of Michigan, Ann Arbor, where he holds the Claude E. Shannon Chair in Engineering Science. His teaching and research interests include computer-aided design, verification, and testing; VLSI circuits; computer architecture; and unconventional computing systems. Professor Hayes has authored over 300 technical papers, several patents, and seven books, including *Computer Architecture and Organization* (3rd ed., 1998), and *Design, Analysis and Test of Logic Circuits under Uncertainty* (with S. Krishnaswamy and I.L. Markov, 2012). He has served as editor of the *Communications of the ACM* and the *IEEE Transactions on Parallel and Distributed Systems*. Professor Hayes was elected a Fellow of the ACM in 2001. His awards and honors include the University of Michigan's Distinguished Faculty Achievement Award (1999) and the Alexander von Humboldt Foundation's Research Award (2004). He received the IEEE Lifetime Contribution Medal for outstanding contributions to test technology in 2013, and the ACM Pioneering Achievement Award for contributions to logic design, fault tolerant computing, and testing in 2014.