On the Functions Realized by Stochastic Computing Circuits

Armin Alaghi and John P. Hayes
Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI, 48109, USA
{alaghi, jhayes}@eecs.umich.edu

ABSTRACT
Stochastic computing (SC) employs conventional logic circuits to implement analog-style arithmetic functions acting on digital bit-streams. It exploits the advantages of analog computation —powerful basic operations, high operating speed, and error tolerance—in important applications such as sensory image processing and neuromorphic systems. At the same time, SC exhibits the analog drawbacks of low precision and complex underlying behavior. Although studied since the 1960s, many of SC's fundamental properties are not well known or well understood. This paper presents, in a uniform manner and notation, what is known about the relations between the logical and stochastic behavior of stochastic circuits. It also considers how correlation among input bit-streams and the presence of memory elements influences stochastic behavior. Some related research challenges posed by SC are also discussed.

Categories and Subject Descriptors
B.2.1 [**Arithmetic and Logic Structures**]: Design styles.

General Terms
Algorithms, design, theory.

Keywords
Stochastic computing, Boolean functions, logic design.

1. INTRODUCTION
Computing hardware has traditionally been partitioned into two broad classes: *analog* acting on continuous data and *digital* acting on discrete data, with real and integer arithmetic being the corresponding mathematical methods. In the digital case where the fundamental data units are the bits 0 and 1, Boolean algebra plays a key role. Analog computing was eclipsed by digital in the mid-twentieth century due to the latter's greater generality, higher precision, and ease of use [19]. Nevertheless, analog computing continues to be found in applications that can exploit its performance advantages (high speed, complex basic operations, and error insensitivity) while tolerating its disadvantages. Of interest here are *hybrid* systems that combine analog and digital features. These include biological systems, in which digital neural signals control analog functions like motion, and a class of artificial computing systems called stochastic [1][12], which are the topic of this paper. Stochastic computing (SC) is so called because it computes with analog probabilities, but represents them by digital bit-streams and processes them with conventional logic circuits.

GLSVLSI '15, May 20 - 22, 2015, Pittsburgh, PA, USA

http://dx.doi.org/10.1145/2742060.2743758

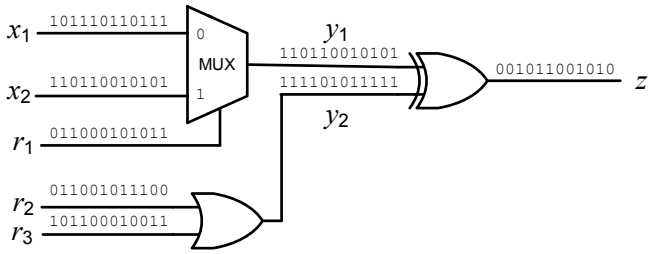

Figure 1. Stochastic circuit implementing the arithmetic function $Z = -0.25(X_1 + X_2)$ using bipolar format.

Figure 1 shows a typical stochastic circuit with bit-streams of length N = 12. It comprises an OR gate, an XOR gate, and a multiplexer (MUX). On one level, this is just a simple logic circuit realizing the Boolean function

$$z(x_1, x_2, r_1, r_2, r_3) = (x_1 \wedge \bar{r}_1 \vee x_2 \wedge r_1) \oplus (r_2 \vee r_3) \quad (1)$$

On another level, it is a relatively powerful stochastic circuit realizing the arithmetic function

$$Z(X_1, X_2) = -0.25(X_1 + X_2) \quad (2)$$

Here, X_1, X_2 and Z denote *stochastic numbers* (SNs) implemented by (pseudo) random bit-streams applied to lines x_1, x_2 and z, respectively.

An N-bit SN X containing N_1 1s and $N - N_1$ 0s has the (*unipolar*) *value* $p_X = N_1/N$. For example, the SN on output line z of Figure 1 has the value p_Z = 5/12. Since p_X always lies in the real-number interval [0,1], it can be seen as the probability of observing 1 in any randomly selected position of X. The value p_X is also referred to as signal intensity, pulse rate, or frequency in different contexts. In neurobiology, for instance, a neural spike train can be modeled by a bit-stream X and its intensity can be represented by p_X [8].

The foregoing probabilistic interpretation along with the randomness of SNs are at the heart of SC, and effectively convert logic gates into analog-style arithmetic components operating on probabilities. For example, a MUX serves as a scaled adder computing the function $0.5(p_{X_1} + p_{X_2})$. With suitable off-line scaling and value approximation, stochastic circuits can be applied to numbers over various ranges. For example, to handle signed numbers, we map p_X to $2p_X - 1$, which changes the SN range from [0,1] to [−1,1]. This is the *bipolar* format and is the interpretation needed for Figure 1 to implement Eq. (2). The output bit-stream Z of Figure 1 then represents 2(5/12) − 1 = −2/12. Figure 1's XOR gate computes $-(2\,p_{Y_1} - 1)(2\,p_{Y_2} - 1)$ and so contributes both multiplication and negation to Eq. (2). This circuit implements a fairly complex arithmetic operation using just a handful of logic gates. A conventional implementation operating on ordinary binary numbers requires many more gates to implement Eq. (2). The hardware simplicity illustrated by Figure 1 is SC's chief attraction.

The three SNs R_1, R_2 and R_3 appearing on inputs r_1, r_2 and r_3, respectively, of Figure 1 are examples of *stochastic constants*. They are typically seen as auxiliary inputs and have the unipolar value 0.5. This value requires an SN with equal numbers of 0s and 1s, which is easily generated by (pseudo) random sources. It can be transformed

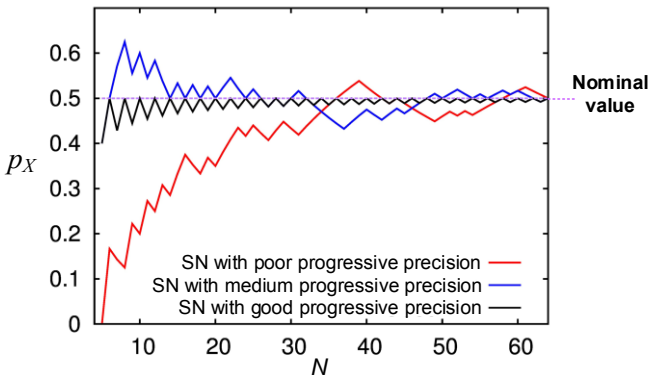

Figure 2. Fluctuations in p_X for 3 bit-streams as their length N increases.

to other constant values via suitable logic circuits [24]. For instance, the OR gate of Figure 1 generates the constant $p_{R_2} + p_{R_3} - p_{R_2}p_{R_3}$ = 0.75 (unipolar) = 0.5 (bipolar), which combined with R_1, gives the 0.25 coefficient required by Eq. (2).

Two other key factors affecting stochastic circuit behavior are the length N and the randomness of the bit-streams. SN formats are highly redundant since $p_X = N_1/N$ is represented by $\binom{N}{N_1}$ different bit patterns. This skewed distribution implies low arithmetic precision. To represent p_X with a precision of n bits, requires X to have length $N \geq 2^n$. Consequently, stochastic circuits tend to have low precision and/or very large N. This can offset the speed advantage of SC's relatively simple arithmetic components. Moreover, such basic questions as— What should N be to guarantee that results are correct to k bits?—are surprisingly hard to answer. Figure 2 shows how p_X fluctuates as N increases for SNs of nominal value $p = 1/2$ generated by a typical stochastic number generator (SNG). SNs that rapidly converge to p are said to have (good) *progressive precision*.

Yet another basic problem in SC is that interacting bit-streams are usually required to be independent or *uncorrelated*, otherwise the results can be unacceptably inaccurate. For example, if two identical (maximally correlated) copies of a bipolar SN X are applied to an XOR gate, the result will be the all-0 bit-stream instead of the expected bipolar product $-X^2$. Ensuring that SNs are sufficiently long and uncorrelated can require an excessive number of SNGs and become the major cost factor in SC [25].

Compensating somewhat for these drawbacks is the fact that long SNs are inherently far less sensitive than ordinary binary numbers to errors caused by environmental noise or hardware faults. Not only do bit-flips occurring in SN X have a small effect on p_X, but two bit-flips in opposite directions cancel one another. Progressive precision can be exploited to speed-up applications where variable degrees of precision are acceptable [2]. Correlation may also be less of a problem than it seems at first sight. While the XOR multiplier is sensitive to correlation among its inputs, the MUX adder is insensitive to input correlations. Moreover, correlation can sometimes be deliberately used to increase the functional range of stochastic circuits and reduce their complexity [4], as will be discussed in Sec. 3. Figure 3 summarizes the advantages and disadvantages of stochastic computing.

Stochastic computing can be traced back to the pioneering ideas of Gaines and Poppelbaum in the 1960s [12][23]. Since then, it has found its major applications in control systems [29] and artificial neural networks [7][26]. More recently, new applications have appeared that involve probabilistic or error-tolerance issues for which SC is well suited, such as image processing [2][17], simulation of probabilistic systems [9][21], data recognition and mining [11], and decoders for channel codes ranging from LDPC to polar codes [13][28]. Furthermore, novel physical technologies are emerging such as memristors that have native stochastic features [16]. Despite these successes, many gaps exist in our understanding of SC and its potential applications.

Feature	Advantages	Disadvantages
Circuit size and power	Tiny arithmetic components	Many random number sources and stochastic-binary conversion circuits
Operating speed	Short clock periods Massive parallelism	Very long bit-streams
Result quality	High error tolerance Progressive precision	Low precision Random number fluctuations Correlation-induced inaccuracies
Design issues	Rich set of arithmetic components	Theory not fully understood Little CAD tool support at present

Figure 3. Advantages and disadvantages of stochastic computing.

The goal of this paper is review and unify recent results on the behavioral or functional properties of stochastic circuits. Section 2 defines stochastic behavior formally, and examines the links between a circuit's logical and stochastic properties, as exemplified by Eqs. (1) and (2). Correlation is examined in Sec. 3, while Sec. 4 addresses sequential design issues. Section 5 draws some conclusions and discusses challenges for future SC research.

2. STOCHASTIC FUNCTIONS

In this section, we discuss the links between combinational logic circuits and their stochastic functions (SFs). We show that that the SFs are functions over the real numbers, which can be expressed in many forms. Later, we will see how factors like correlation and the presence of memory elements can change the SFs.

Suppose some n-input single-output combinational circuit C realizes the Boolean function (BF) $z(x_1, x_2, \ldots, x_n)$. This function has the canonical sum-of-minterms form

$$z(x_1, x_2 \ldots, x_n) = \bigvee_{i=0}^{2^n-1} c_i \wedge m_i \qquad (3)$$

where the c_i's are 0-1 constants. The m_i's are *minterms* of the form $\tilde{x}_{i,1} \wedge \tilde{x}_{i,2} \wedge \cdots \wedge \tilde{x}_{i,n}$ where $\tilde{x}_{i,j}$ is either $x_{i,j}$ or $\bar{x}_{i,j}$. For example, the sum-of-minterms representation of the XOR function $z_{\mathrm{XOR}}(x_1, x_2)$ is $m_2 \vee m_3 = (x_1 \wedge \bar{x}_2) \vee (\bar{x}_1 \wedge x_2)$.

Now suppose n SNs are applied to the inputs of C. If X_i is the (unipolar) probability value of the SN on x_i, i.e., $X_i = p_{X_i}$, then $\bar{x}_i$ has the probability value $1 - X_i$. The following theorem gives C's output probability Z, and so defines its stochastic function (SF).

Theorem 1: Let $z(x_1, x_2 \ldots, x_n)$ be a Boolean function defined by Eq. (3). The stochastic function $Z(X_1, X_2 \ldots, X_n)$ implemented by z, assuming all input SNs are independent, is

$$Z(X_1, X_2 \ldots, X_n) = \sum_{i=0}^{2^n-1} c_i M_i \qquad (4)$$

where $M_i = \tilde{M}_{i,1}\tilde{M}_{i,2} \cdots \tilde{M}_{i,n}$ with $\tilde{M}_{i,j} = p_{X_{i,j}} = X_{i,j}$ if the corresponding minterm m_i of Eq. (3) has $\tilde{x}_{i,j} = x_{i,j}$; $\tilde{M}_{i,j}$ is $1 - p_{X_{i,j}} = 1 - X_{i,j}$ if $\tilde{x}_{i,j} = \bar{x}_{i,j}$.

This key result was first shown by Parker and McCluskey [22] using rather ad hoc notation. Note that each M_i corresponds to a minterm and is the probability of the corresponding input combination. These probabilities have the form stated in Thm. 1 when the input SNs are independent. As will be shown in Sec. 3, if the input SNs are correlated, the M_i's may take a different form. For the XOR gate with independent inputs, Thm. 1 implies

$$Z_{\mathrm{XOR}}(X_1, X_2) = X_1(1 - X_2) + (1 - X_1)X_2$$

which, when multiplied out, becomes

$$Z_{\mathrm{XOR}}(X_1, X_2) = X_1 + X_2 - 2X_1X_2 \qquad (5)$$

The sum-of-minterms-style probability expression (4) can be seen as a *canonical representation* of the stochastic function Z realized by the Boolean function z. It thus captures z's stochastic behavior with respect to the basic unipolar format. When the X_i's are restricted to 0 and 1, and sum is interpreted as OR, Eq. (4) reduces to Eq. (3), so Z is effectively an interpolation of z in the real-number domain. Equation (4) is also easily converted to other SN formats. To convert from

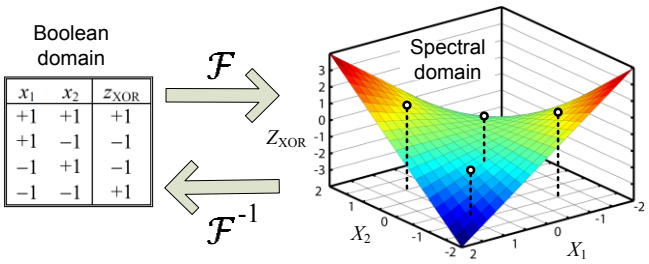

Figure 4. Spectral transformation of the XOR function.

unipolar to bipolar, for instance, replace $p_{X_{i,j}}$ by $2p_{X_{i,j}} - 1$, and re-define the SN value $X_{i,j}$ to be $2p_{X_{i,j}} - 1$.

The canonical representation of Eq. (4) can also be expressed as the inner product of two vectors. The first is the truth-table vector $\boldsymbol{C}_z = [c_0 \;\; c_1 \;\; \cdots \;\; c_{2^n-1}]$ defining z in terms of the constant coefficients in Eq. (3). The second is the input vector $\boldsymbol{M} = [M_0 \;\; M_1 \;\; \cdots \;\; M_{2^n-1}]$ specifying the probability distribution of the input combinations, or equivalently, the stochastic minterm functions. We can now rewrite Eq. (4) as follows, where "·" denotes the inner-product operation:

$$Z(X_1, X_2 \ldots, X_n) = \boldsymbol{C}_z \cdot \boldsymbol{M} = [c_0 \;\; c_1 \;\; \cdots \;\; c_{2^n-1}] \cdot [M_0 \;\; M_1 \;\; \cdots \;\; M_{2^n-1}] \quad (6)$$

The c_i elements in Eqs. (3), (4) and (6) are the same and belong to the binary set {0,1}. Since SFs deal with real numbers, we can further generalize Thm. 1 by allowing the c_i's to be any numbers in the real interval [0,1]. Such generalized c_i coefficients can be interpreted as constant SNs applied to the circuit when the corresponding minterm m_i is activated or set to 1. For example, if $c_0 = 0$, $c_1 = 0.5$, $c_2 = 0.5$, and $c_3 = 1$ (or in vector form [0 0.5 0.5 1]), then Eq. (4) becomes

$$Z(X_1, X_2) = 0.5X_1(1 - X_2) + 0.5(1 - X_1)X_2 + X_1X_2 \quad (7)$$

which is the scaled add function $0.5(X_1 + X_2)$. The coefficients $c_1 = c_2 = 0.5$ in (7) imply that when minterms m_1 and m_2 are activated, a constant SN of value 0.5 should propagate to the output. Such constant probabilities can be obtained from (pseudo) random number sources. These sources often appear as auxiliary inputs in the corresponding circuit. For example, the r_i inputs of Figure 1 are auxiliary inputs that are fed with SNs of value 0.5.

Like BFs, SFs can be expressed and interpreted in different forms, which are associated with different, and sometimes useful, circuit design styles. When Eq. (4) is expanded in the manner illustrated by Eq, (5), Z takes the form of a *multi-linear polynomial*, i.e., one which can contain products of X_i variables, but no variable appears with a power of two or higher. If bipolar instead of unipolar format is used then, as we saw in the case of Figure 1, Eq. (5) changes to $Z_{XOR} = -X_1X_2$, a different multi-linear polynomial. Qian et al. observed that these expressions can be replaced by another interesting class of polynomials called Bernstein polynomials [25].

Alaghi and Hayes showed that the spectrum of a BF z obtained via the Fourier transform reveals z's stochastic behavior in useful ways [3]. First, to facilitate the use of spectral transforms, we map the usual 0 and 1 values of $\boldsymbol{C}_z$ into the real numbers +1 and –1, respectively; see Figure 4. Then we multiply $\boldsymbol{C}_z$ by an appropriate matrix, such as the Walsh-Hadamard matrix $\boldsymbol{H}_n$. This produces a 2^n-dimensional vector (z's spectrum) which defines yet another polynomial form of Z. In the case of z_{XOR}, we get $Z_{XOR} = X_1X_2$.

Spectral transformation can be expressed symbolically as $Z = \mathcal{F}(z)$. An advantage of the spectral viewpoint is that the design problem of finding a z to implement a given SF Z reduces to computing the inverse spectral transform

$$z = \mathcal{F}^{-1}(Z) \quad (8)$$

A difficulty here is that there may be no BF z satisfying Eq. (8). This problem can be resolved by approximating Z by another function Z^* for which Eq. (8) has a solution in the Boolean domain. This entails expressing Z^* in a suitable (polynomial) form and introducing new

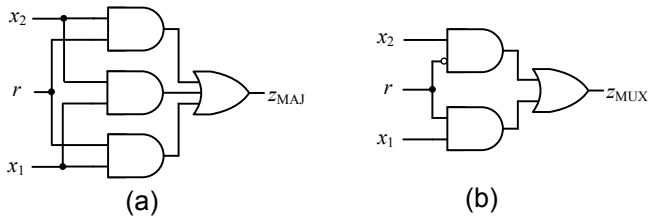

Figure 5. Two circuits implementing scaled addition when $R = 0.5$.

stochastic variables and constants, a complex process for which only heuristic methods are known [3][10][25]. For example, the function $Z = X^{0.45}$, commonly used in image processing, has no suitable polynomial form. However, it is approximated by $Z^* = -0.75X^2 + 1.5X + 0.25$ [3]. Applying the inverse spectral transform to Z^* yields

$$z(x_1, x_2, r_1, r_2) = x_1 \vee x_2 \vee r_1 \wedge r_2$$

which includes several new inputs. Variables x_1 and x_2 are supplied with two independent SNs representing X, while r_1 and r_2 are auxiliary inputs supplied with constant SNs of value 0.5.

Although every BF has a unique sum-of-minterms form (3), it turns out, surprisingly, that several different BFs can lead to the same SF [3][10]. This happens when generalized minterm coefficients, i.e., real-valued constant inputs, are allowed in stochastic functions. Consider, for instance, the majority function $z_{MAJ} = (x_1 \wedge x_2) \vee (x_1 \wedge r) \vee (x_2 \wedge r)$ and the multiplexer function $z_{MUX} = (x_1 \wedge r) \vee (x_2 \wedge \bar{r})$. They map to two different SFs $Z_{MAJ}(X_1, X_2, R)$ and $Z_{MUX}(X_1, X_2, R)$, as is easily shown using Eq. (4). However, if R is set to 0.5, both SFs become the same, i.e.

$$Z_{MAJ}(X_1, X_2, 0.5) = Z_{MUX}(X_1, X_2, 0.5) = 0.5(X_1 + X_2) \quad (9)$$

This, again, is the scaled addition operation of SC. Figure 5 shows two-level realizations of z_{MAJ} and z_{MUX}. Although each is optimal in the usual circuit-cost sense, the multiplexer has somewhat lower cost. However, in some emerging nanotechnologies, majority gates are the fundamental building block [15] so a majority-based scaled adder might be preferred.

The preceding discussion shows that SC adds an interesting new twist to logic optimization, namely: Find the "best" Boolean function z that implements a target SF Z (or an approximation thereto) in the form $z(X_V; X_C)$, where X_V denotes inputs to which variable SNs are applied, and X_C denotes auxiliary inputs to which constant SNs are applied. (For notational simplicity. X_V may refer either to Boolean variables or SNs.) With slight loss of generality, we assume all members of X_C are 0.5, i.e., 0s and 1s are applied with equal probability to constant inputs. This reflects the nature of the random sources normally used in SC.

With these assumptions, we can now define various types of stochastic equivalence among Boolean functions. For example, two BFs $z_1(X_V; X_C)$ and $z_2(X_V; X_C)$are *stochastically equivalent*, denoted $z_1 \equiv z_1$, if $Z_1(X_V; X_C) = Z_2(X_V; X_C)$ [10]. Equation (9) shows that $z_{MAJ} \equiv z_{MUX}$. For any given size parameters $|X_V| = s$ and $|X_C| = t$, the $\equiv$ relation partitions the set of SFs $Z(X_V; X_C)$ into stochastic equivalence classes (SECs). With $s = 2$ and $t = 1$, the $2^{2^3} = 256$ distinct BFs form 81 SECs, including a 4-member class E containing z_{MAJ} and z_{MUX}. (The other two members of E result from replacing r by $\bar{r}$, whose value is also 0.5) Each SEC represents a potentially useful arithmetic component or circuit for designing stochastic circuits. Note that although X_C is usually seen as a set of secondary inputs, they form an intrinsic part of an SF and consume significant circuit resources. This is clear from Figure 1 where the two-input SF of Eq. (2) requires a five-input logic circuit in which X_C with $t = 3$ has a non-trivial role comparable in complexity to that of X_V.

The foregoing SEC concept can also play a useful role in optimizing stochastic circuits [10]. For small s and t, an SEC representing some SF $Z(X_V; X_C)$ can be searched systematically either to find an optimal BF $z(X_V; X_C)$ implementing Z, or else to

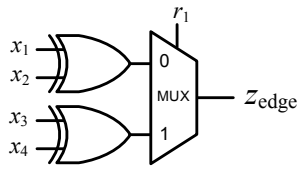

Figure 6. Stochastic edge detector [2].

evaluate the optimality of a known *z*. To illustrate, consider the stochastic function

$$Z_{\text{edge}} = 0.5 \times (|X_1 - X_2| + |X_3 - X_4|) \quad (10)$$

which defines the Roberts Cross function for edge detection in black-and-white images. It employs a four-pixel window which outputs four Boolean variables x_1,x_2,x_3,x_4 on which random bit-streams appear that measure light intensity. Figure 6 shows a stochastic logic circuit implementing Eq. (10) which was derived by ad hoc means [2]. (Its input SNs must meet certain correlation requirements, which we consider in Sec. 3, but they do not affect SEC membership.) This circuit's area cost is about 100x less than that of a non-stochastic implementation of Eq. (10), and was thought to be optimal. Its BF is

$$z_{\text{edge}} = \bar{x}_1 \wedge x_2 \wedge \bar{r}_1 \vee x_1 \wedge \bar{x}_2 \wedge \bar{r}_1 \vee \bar{x}_3 \wedge x_4 \wedge r_1 \vee x_3 \wedge \bar{x}_4 \wedge r_1 \quad (11)$$

whose inputs x_1, x_2, x_3 and x_4 define X_V, while r_1 defines X_C. The SEC for z_{edge} contains 256 functions. The implementation cost of these functions was computed in terms of literal count [14] for an optimal two-level design using a conventional CAD tool, and found to range from 16 to 45. Since z_{edge}'s literal cost is 16, as can be seen from Eq. (11), the optimality of the edge-detector design in Figure 6 is confirmed.

3. IMPACT OF CORRELATION

In stochastic computations, it is often necessary to convert inputs from a number style such as analog or weighted-binary to stochastic form. This requires *stochastic number generators* (SNGs) which tend to cost far more than other SC components. The large numbers of them found in traditional designs—Figure 1 needs up to five SNGs for its five inputs—render many such designs impractical.

As shown in Figure 7, a typical SNG comprises a comparator and a random number source (RNS). In each clock cycle, a new random number is compared with the input number X^* and a bit of the corresponding SN X appears at the output. Over the years, many variants of this design have been proposed. Most implement the RNS by a deterministic sequential circuit such as a linear feedback shift register (LFSR) that produces pseudo-random outputs. Alternative SNG designs can be found in [1][5]. It is also possible to combine non-random and pseudo-random bit-streams, but the results have been unpromising. "True" random sources, made possible by nanotechnologies like memristors [16] and magnetic-tunnel junction devices [20] have also been proposed recently for SC.

As noted earlier, the inputs of a stochastic circuit must usually be independent or uncorrelated in order to achieve the desired functionality. Correlation is caused by insufficient randomness among SNs and is a key source of inaccuracy. Reducing correlation requires many costly SNGs with independent RNSs. Alaghi and Hayes [4][6] however, show that some circuits are inherently *correlation insensitive* (CI), meaning that correlation among their input SNs does not alter their stochastic function. A formal definition of CI is given in [6], where it is shown that exploiting correlation insensitivity can reduce stochastic circuit area substantially.

Correlation insensitivity is most readily seen in the scaled adder realized by the multiplexer of Figure 5b. The output z_{MUX} is x_1 if $r = 1$ and x_2 if $r = 0$. Hence, the inputs x_1 and x_2 never affect z_{MUX} simultaneously, so any correlation between the SNs X_1 and X_2 is masked by the circuit. Knowledge of this kind can be used to reduce the SNG costs. For instance, it implies that a scaled adder's inputs x_1 and x_2 can share an RNS, as shown in Figure 8.

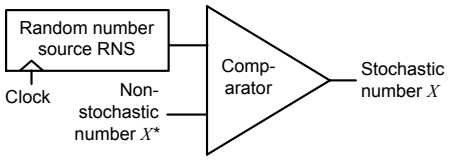

Figure 7. Generic stochastic number generator (SNG).

The stochastic function of a circuit with correlated inputs can be expressed using modified versions of Eqs. (4) or (6). The M_i's of these equations are the probabilities of the circuit's various input combinations. For example, when $n = 2$, $\boldsymbol{M} = [M_0 \quad M_1 \quad M_2 \quad M_3]$ denotes the probability of x_1x_2 being 00, 01, 10, and 11, respectively. Suppose X_1 and X_2 are constant SNs with values 0.3 and 0.2, respectively. By Thm. 1, $\boldsymbol{M}$ is $\boldsymbol{M}_1 = [0.56 \quad 0.14 \quad 0.24 \quad 0.06]$ when X_1 and X_2 are independent. But if X_1 and X_2 are correlated, the M_i's generally take different values. Suppose that, in the current example, whenever X_2 applies 1 to x_2, X_1 always applies a 1 is to x_1, implying that X_1 and X_2 have a high degree of correlation. Then $\boldsymbol{M}_1$ changes to $\boldsymbol{M}_2 = [0.7 \quad 0 \quad 0.1 \quad 0.2]$.

Earlier we observed that a MUX-based scaled adder is CI with respect to X_1 and X_2 using the intuitive argument that the output function z_{MUX} does not depend on both inputs and at the same time. This argument does not apply to a MAJ-based scaled adder. To determine whether it too is CI, consider the majority circuit of Figure 5a. On assigning a constant SN of value 0.5 to its *r* input (assuming it is independent of the other input SNs), its truth-table vector becomes $\boldsymbol{C}_{\text{MAJ}} = [0 \quad 0.5 \quad 0.5 \quad 1]$, which implements stochastic scaled addition if the inputs are independent. Now assume a generic input vector $\boldsymbol{M} = [M_0 \quad M_1 \quad M_2 \quad M_3]$ with no specific assumptions about correlation between X_1 and X_2. From Eq. (6), we can write

$$\begin{aligned} Z_{\text{MAJ}}(X_1, X_2) &= [M_0 \quad M_1 \quad M_2 \quad M_3] \cdot [0 \quad 0.5 \quad 0.5 \quad 1] \\ &= 0.5M_1 + 0.5M_2 + M_3 \\ &= 0.5(M_1 + M_3) + 0.5(M_2 + M_3) \end{aligned}$$

Noting that $M_1 + M_3 = X_1$ and $M_2 + M_3 = X_2$ for any possible level of correlation between X_1 and X_2, we get $Z_{\text{MAJ}}(X_1, X_2) = 0.5(X_1 + X_2)$. This implies that the majority gate is CI with respect to X_1 and X_2 when the SN constant 0.5 is assigned to *r*.

Systematic correlation among input SNs of a circuit is not necessarily a source of inaccuracy. In fact, it can change a circuit's underlying stochastic function to a more desirable one [4]. For example, consider the upper XOR gate of Figure 6. As shown by Eq. (5), it has the stochastic function $Z_{\text{XOR}}(X_1, X_2) = X_1 + X_2 - 2X_1X_2$ when its inputs are independent SNs. However, with maximally correlated input SNs, the XOR gate implements the stochastic function $Z_{XOR}(X_1, X_2) = |X_1 - X_2|$, which turns out to be a key part of the Z_{edge} function implemented by the circuit of Figure 6.

To quantify systematic correlation among SNs, a measure called SCC (*stochastic computing correlation*) is proposed in [4]. Zero *SCC* between two SNs implies their independence. If *SCC* = +1, then the SNs have maximum overlap of 1s and 0s; if SCC = –1, then the SNs have a minimum overlap of 1s and 0s. It is important to note that these conditions hold for SNs of arbitrary value. For example, if X_1 = 11110000 and X_2 = 1100000, then $SCC(X_1, X_2)$ = +1, while if X_1 = 11110000 and X_2= 0000011, we get $SCC(X_1, X_2)$ = –1. In contrast, the standard Pearson correlation measure imposes constraints on the

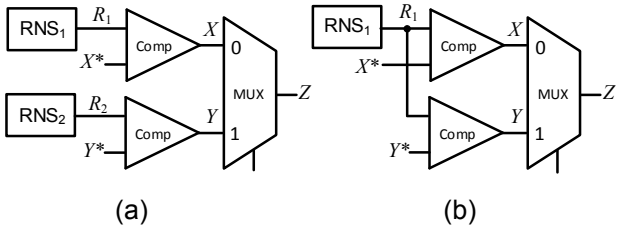

Figure 8. Stochastic number generation for a scaled adder with (a) two independent RNSs, and (b) one shared RNS.

SN values. For instance, a correlation of +1 implies that the SNs are identical, and hence must have the same value. Thus SCC is a more suitable correlation measure for SC. More importantly, we can incorporate SCC into Thm. 1 and extend it as follows.

Theorem 2: The correlation-dependent stochastic function $Z(X_1, X_2)$ implemented by the Boolean function $z(x_1, x_2)$ defined by Eq. (3) with $n = 2$ is

$$Z(X_1, X_2) = \sum_{i=0}^{3} c_i M_i'$$

where representative M_i''s are given by Figure 9.

	$SCC(X_1, X_2) = 0$	$SCC(X_1, X_2) = -1$	$SCC(X_1, X_2) = +1$
M_0'	$(1 - X_1)(1 - X_2)$	$\max(1 - X_1 - X_2, 0)$	$\min(1 - X_1, 1 - X_2)$
M_1'	$(1 - X_1)X_2$	$\min(1 - X_1, X_2)$	$\max(X_2 - X_1, 0)$
M_2'	$(1 - X_2)X_1$	$\min(1 - X_2, X_1)$	$\max(X_1 - X_2, 0)$
M_3'	X_1X_2	$\max(X_1 + X_2 - 1, 0)$	$\min(X_1, X_2)$

Figure 9. Correlation-dependent probabilities for Thm. 2.

To illustrate this theorem, consider the upper XOR gate of Figure 6 which has the two minterms m_1 and m_2. When supplied by SNs with *SCC* = +1, it implements the stochastic function

$$Z_{\text{XOR}} = \max(X_2 - X_1, 0) + \max(X_1 - X_2, 0) = |X_1 - X_2|$$

In contrast, the SF implemented by the multiplexer of Figure 5b remains the same for all possible SCC values among its inputs.

4. SEQUENTIAL STOCHASTIC CIRCUITS

Stochastic circuits are highly sequential in that their behavior is determined by long sequences of binary data involving synchronous sequential components like SNGs and I/O registers. So far in this paper (and throughout the SC literature) it has been assumed that the data-processing functions are fully defined by combinational circuits. Introducing memory into these circuits changes the picture.

Consider the circuit of Figure 10a which combines an AND gate with a D-flip-flop. The AND acts as a stochastic multiplier implementing the function $Z = X(1 - Y)$. The D-flip-flop simply shifts its input bit-stream by 1 bit, and implements the stochastic function $Y = Z$. Eliminating Y from the preceding equations, gives $Z = X/(1 + X)$, which is the SF implemented by the circuit of Figure 10a. This function does not have an appropriate polynomial form, and so cannot be directly implemented by combinational stochastic circuits. A similar example is the JK-flip-flop shown in Figure 10b, which has the SF $Z = X_1/(X_1 + X_2)$, and is used to approximate stochastic division.

Figure 11 shows the general structure of an n-input sequential circuit with k flip-flops. The combinational block generates the output z and the next state variables $y_1^+, \ldots, y_k^+$ based on the inputs and the current state variables $y_1, \ldots, y_k$. The memory block merely copies the y_i^+'s values to y_i at the active clock edge. The stochastic functions implemented by a sequential circuit C are defined by the stationary distribution Y of its states and the primary output Z, which can be derived by solving the Markov chain equations for C [12].

As an example, let $n = k = 1$ in the circuit of Figure 11, and assume that the Boolean functions $y^+(x, y)$ and $z(x, y)$ realized by the combinational block, have the following truth-table vectors.

$$\boldsymbol{C}_{y^+} = [c_0^y \quad c_1^y \quad c_2^y \quad c_3^y] \text{ and } \boldsymbol{C}_z = [c_0^z \quad c_1^z \quad c_2^z \quad c_3^z]$$

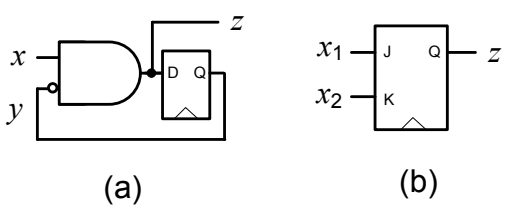

Figure 10. Sequential stochastic circuits implementing (a) $Z = X/(1 + X)$ and (b) $Z = X_1/(X_1 + X_2)$.

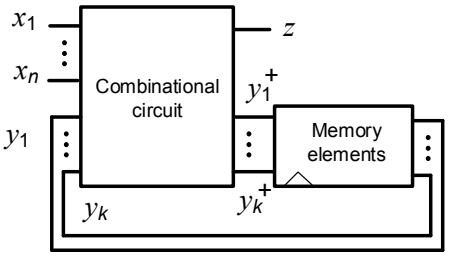

Figure 11. An n-input sequential circuit with k flip-flops.

Then the stationary state distribution at Y is obtained by assigning $Y^+ = Y$ in the following equation

$$Y^+(X, Y) = \boldsymbol{C}_{y^+} \cdot [(1 - X)(1 - Y) \quad (1 - X)Y \quad X(1 - Y) \quad XY]$$

Solving this equation gives Y's SF.

$$Y(X) = \frac{c_0^y + X(c_2^y - c_0^y)}{1 + c_0^y - c_1^y - X(c_0^y - c_1^y - c_2^y + c_3^y)}$$

The SF at Z can be derived similarly.

By assigning different values to the M_i^y's and M_j^z's, one can obtain all the possible stochastic sequential circuits with one input and one flip-flop. Interestingly, many of these circuits implement similar stochastic functions, showing that there are many equivalence classes of sequential stochastic circuits analogous to the combinational SECs mentioned in Sec. 2. For instance, the two circuits shown in Figure 12 implement exactly the same SF $Z(X) = (2X - 2)/(X - 2)$, so they are stochastically equivalent. Within the 256 single-input single-flip-flop sequential circuits, there are only 55 distinguished equivalent classes of SFs.

These small examples show that sequential circuits implement a larger class of SFs than combinational circuits, namely, rational functions of the form

$$Z(X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n)}{Q(X_1, \ldots, X_n)}$$

in which P and Q are polynomials.

As noted, the stochastic function of sequential circuits can be obtained by solving the corresponding equations for Y and Z, but this can be very difficult when many state variables are involved. To sidestep this problem, Gaines proposed restricting attention to finite-state machines (FSMs) with a chain structure in which the states are ordered and transitions only occur between adjacent states; jumping over states is not allowed [12]. This restriction allows easy Markov chain analysis. Figure 13 shows the state behavior of one such chain-structured FSM, the ADDIE (*ADaptive Digital Element*). Gaines also argued that state transitions should be local to avoid excessive fluctuations in SF values. ADDIEs have been used in various analog-style stochastic circuits such as filters [12]. Similar chain structured sequential circuits can implement non-polynomial functions such as *tanh* and *exp* efficiently [7][18].

Variations and extensions of Gaines's ADDIE model have been proposed over the years. A 2-dimensional extension of the chain-structured FSM was proposed by Li et al. [18]. A more general form of ADDIE was used by Saraf et al. [27] to implement SFs such as trigonometric functions. Evidently, sequential implementations can be more efficient than combinational for certain classes of SFs.

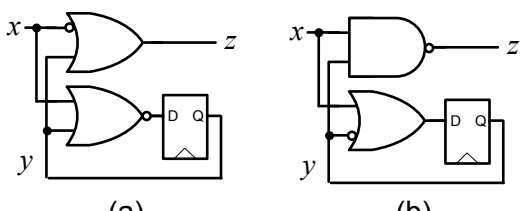

Figure 12. Two sequential circuits implementing $Z(X) = (2X - 2)/(X - 2)$.

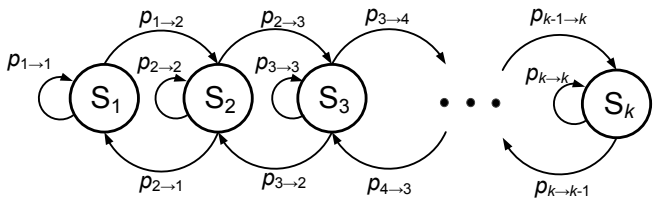

Figure 13. State diagram of a generalized ADDIE [12].

However, many optimal combinational stochastic circuits exist. For example, the sequential edge-detection circuit in [17] is more than 20 times larger than the combinational edge detector of Figure 6.

A drawback of sequential circuits is that they require a transition (warm-up) period before settling to the desired stationary distribution. During this period, which can be quite long, the circuit may produce inaccurate results. Another disadvantage of sequential circuits is that their behavior is affected by *auto-correlation* among the SNs. This refers to the correlation between a SN and its shifted or delayed versions. Auto-correlation imposes new requirements for the SNGs used for sequential circuits. Combinational circuits, being memory-less, are not affected by auto-correlation.

Finally, we note that unlike combinational circuits, a general design methodology for sequential stochastic circuits is not known. Most existing methods are limited to chain-structured designs.

5. DISCUSSION

Stochastic computing is a fascinating blend of analog and digital concepts. By associating data values with signal probabilities, SC enables analog computation to be performed using digital bit-streams and circuits. This hybrid approach tends to merge the advantages and disadvantages of analog and digital. Unsurprisingly, SC is best suited to applications that benefit most from the advantages (powerful low-cost primitives and error tolerance) and are least affected by the disadvantages (low precision, scaling issues, and complex behavior).

We have examined the dual nature of SC from a functional perspective, starting from the fact that a stochastic circuit implements both a Boolean function z and a stochastic function Z. Theorem 1 states the basic connection between z and Z. This connection can be expressed in many different, but equivalent, ways (multi-linear polynomials, spectral transforms, etc.), some of which have interesting and potentially useful design implications. By introducing auxiliary variables and constants, and accounting for phenomena like correlation and sequential behavior, the range of stochastic functions and their implementations can be greatly expanded.

Many questions, old and new, about SC remain unanswered. We have only begun to investigate the full range of stochastic functions that are realizable, even by relatively small logic circuits. Sequential stochastic circuits still present many challenges. Most of what we know about correlation, precision, scalability and the like derives from studies of circuits involving just a few gates and flip-flops. It seems likely that many of the problems of building large stochastic systems have not yet been fully recognized, let alone solved.

ACKNOWLEDGEMENT

This work was supported by Grant CCF-1318091 from the National Science Foundation.

REFERENCES

[1] Alaghi, A. and Hayes, J.P. 2013. Survey of stochastic computing. *ACM Trans. Embed.Comp. Syst.*12, 92:1-92:12.

[2] Alaghi, A. et al. 2013. Stochastic circuits for real-time image-processing applications. In *Proc. DAC,* 136:1-136-6.

[3] Alaghi, A. and Hayes, J.P. 2012. A spectral transform approach to stochastic circuits. In *Proc. ICCD,* 315-312.

[4] Alaghi, A. and Hayes, J.P. 2013. Exploiting correlation in stochastic circuit design. In *Proc. ICCD*, 39-46.

[5] Alaghi, A. and Hayes, J.P. 2014. Fast and accurate computation using stochastic circuits. In *Proc. DATE*. 1-4.

[6] Alaghi, A. and Hayes, J.P. 2015. Dimension reduction in statistical simulation of digital circuits. In *Proc. Symp. Theory of Modeling and Simulation* (TMS). To appear.

[7] Brown, B.D. and Card, H.C. 2001. Stochastic neural computation I: computational elements. *IEEE Trans. Computers*. 50, 891-905.

[8] Brown, E.N. 2004. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience.*7, 456-461.

[9] Chen, H. and Han, J. 2010. Stochastic computational models for accurate reliability evaluation of logic circuits. In *Proc. GLSVLSI*, 61-66.

[10] Chen, T.H. and Hayes, J.P. 2015. Equivalence among stochastic logic circuits and its application. In *Proc DAC*. To appear.

[11] Chippa V.K. et al. 2014. StoRM: a stochastic recognition and mining processor. In *Proc. ISLPED*, 39-44.

[12] Gaines, B.R. 1969. Stochastic computing systems. In *Advances in Information Systems Science*, Springer. 2, 37-172.

[13] Gross, W.J. et al. 2005. Stochastic implementation of LDPC decoders. In *Proc. Asilomar Conf.* 713-717.

[14] Hachtel, G.D. and Somenzi, F. 1996. *Logic Synthesis and Verification Algorithms.* Kluwer.

[15] Zhang, R. et al. 2007. Majority and minority network synthesis with application to QCA-, SET-, and TPL-based nanotechnologies. *IEEE Trans. CAD*. 26, 1233-1245.

[16] Knag, P. et al. 2014. A native stochastic computing architecture enabled by memristors. *IEEE Trans. Nanotech.* 13, 283-293.

[17] Li, P. et al. 2013. Computation on stochastic bit streams: digital image processing case studies. *IEEE Trans. VLSI*. 22, 449-462.

[18] Li, P. et al. 2014. Logical computation on stochastic bit streams with linear finite-state machines. *IEEE Trans. Computers*. 63, 1473-1485.

[19] MacLennan, B.J. 2009. Analog computation. In *Encyclopedia of Complexity and System Science*, Springer, 271-294.

[20] Onizawa, N. et al. 2014. Analog-to-stochastic converter using magnetic-tunnel junction devices. In *Proc. NANOARCH*, 59-64.

[21] Paler, A. et al. 2013. Approximate simulation of circuits with probabilistic behavior. In *Proc. DFT Symp.* 95-100.

[22] Parker, K.P. and McCluskey, E.J. 1975. Probabilistic treatment of general combinational networks. *IEEE Trans. Computers.* C-24, 668-670.

[23] Poppelbaum, W.J. 1976. Statistical processors. In *Advances in Computers*, Academic Press, 187-230, 1976.

[24] Qian, W. et al. 2011. Transforming probabilities with combinational logic. *IEEE Trans. CAD*. 30, 1279-1292.

[25] Qian, W. et al. 2011. An architecture for fault-tolerant computation with stochastic logic. *IEEE Trans. Computers*. 60, 93-105.

[26] Rossello, J.L. et al. 2010. Hardware implementation of stochastic-based neural networks. In *Proc. IJCNN*. 1-4.

[27] Saraf, N. et al. 2013. Stochastic functions using sequential logic. In *Proc. ICCD,* 507-510.

[28] Yuan, B and Parhi, K.K. 2015. Successive cancellation decoding of polar codes using stochastic computing. In *Proc. ISCAS.* To appear.

[29] Zhang D. and Li, H. 2008. A stochastic-based FPGA controller for an induction motor drive with integrated neural network algorithms. *IEEE Trans. Industrial Electronics*, 55, 551-561.