Exploiting Correlation in Stochastic Circuit Design

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) is a re-emerging computing paradigm which enables ultra-low power and massive parallelism in important applications like real-time image processing. It is characterized by its use of pseudo-random numbers implemented by 0-1 sequences called stochastic numbers (SNs) and interpreted as probabilities. Accuracy is usually assumed to depend on the interacting SNs being highly independent or uncorrelated in a loosely specified way. This paper introduces a new and rigorous SC correlation (SCC) measure for SNs, and shows that, contrary to intuition, correlation can be exploited as a resource in SC design. We propose a general framework for analyzing and designing combinational circuits with correlated inputs, and demonstrate that such circuits can be significantly more efficient and more accurate than traditional SC circuits. We also provide a method of analyzing stochastic sequential circuits, which tend to have inherently correlated state variables and have proven very hard to analyze.**

Keywords—Stochastic computing; signal correlation; low-power design; stochastic circuit design.

I. INTRODUCTION

Stochastic computing (SC) was introduced in the 1960s as a way to implement complex computing tasks at low hardware cost [2][6][17]. Its key feature is the use of long random bit-streams called stochastic numbers (SNs) to represent the data being processed. Fig. 1a shows how a single AND gate can multiply two n-bit SNs in n clock cycles. Here x, y, z denote logic functions, and the 8-bit sequences X, Y, Z denote the corresponding SNs with their probability values p_X, p_Y, p_Z in parenthesis. The probability p_Z of 1 appearing on z is equal to the probability p_X of 1 on x, multiplied by the probability p_Y of a 1 on y, assuming that the input signals are independent or uncorrelated. SC has seen little use in practice because it involves complex and poorly understood trade-offs among circuit size, computation time, and accuracy. There has been a recent resurgence of interest in SC as it has been shown to be very cost-effective in some useful applications, notably low-density parity check (LDPC) decoding [9][16] and image processing [3][13]. Stochastic circuits have other nice properties such as very low power and high error tolerance.

On the other hand, SC has several drawbacks that need to be addressed to obtain practical circuits. In particular, independent inputs have been seen as necessary for SC because correlated inputs can produce inaccurate results [2]. This is illustrated by Fig. 1b where the bit-streams X and Y are identical, and so are maximally correlated. The output Z now has the value p_X instead of the desired product $p_X p_Y$.

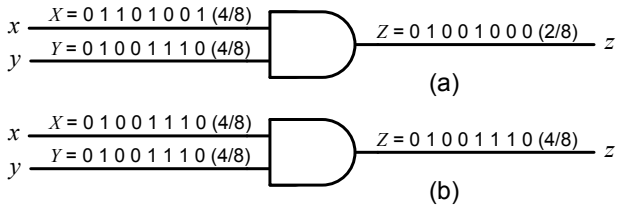

Figure 1. Stochastic multiplication; (a) accurate result with uncorrelated inputs; (b) inaccurate result due to correlated inputs.

Fig. 2 shows some basic components for constructing stochastic circuits. Addition is implemented by a two-way multiplexer in the scaled form $(p_X + p_Y)/2$, which ensures that the sum of two SNs lies in the probability interval [0,1]. Note that the scaling factor is supplied by an independent SN W of fixed value 0.5, i.e., a purely random bit-stream. The remaining two circuits convert numbers between ordinary "binary" form N and stochastic form X with $p_X = N/2^k$. The (pseudo) random number generator in Fig. 2c is typically implemented by a linear feedback shift register (LFSR) [10]. Signed numbers can be handled by bipolar notation, which maps the SN range from [0,1] to [−1,1]. If X is an SN with (unipolar) value p_X, its corresponding bipolar value is $\hat{p}_X = 2p_X - 1$. The circuits of Fig. 2 require very minor modification to handle bipolar numbers. For example, an XNOR gate performs the bipolar multiplication $\hat{p}_Z = \hat{p}_X \hat{p}_Y$, while the multiplexer of Fig. 2b continues to serve as a scaled adder for bipolar SNs.

Most SC circuits designed so far do not behave satisfactorily when their inputs are even moderately correlated. The general solution to this problem has been to avoid correlation as much as possible, either by using independent sources for all input SNs, or else by selectively re-randomizing correlated SNs. The latter involves first

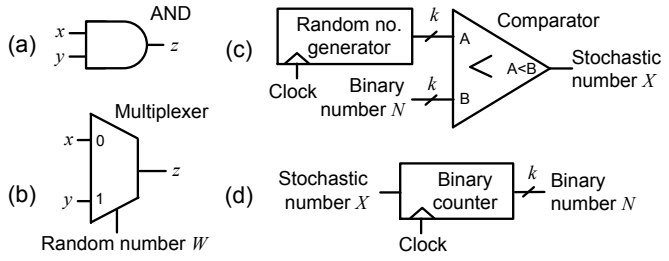

Figure 2. Unipolar SC components: (a) multiplier (b) scaled adder (c) binary-to-stochastic converter (d) stochastic-to-binary converter.

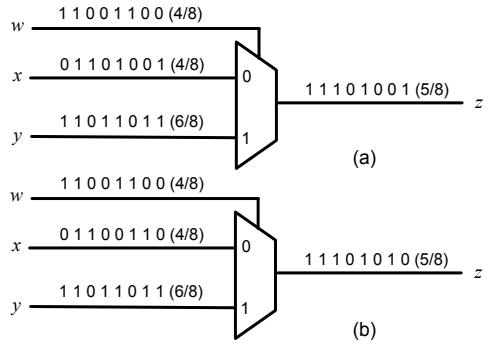

Figure 3. Stochastic addition showing accurate results with (a) uncorrelated and (b) correlated inputs *X* and *Y*.

converting an SN to binary form and then converting it back to stochastic form via circuits like those of Fig. 2c-d. These steps come at high cost, imposing as much as 80% area overhead on one stochastic image-processing circuit [18]. In an iterative LPDC decoder [16], SNs must be re-randomized in every iteration to avoid deadlocks.

Correlation is poorly understood in the SC context, in contrast with areas such as communications theory [10]. The only relevant study we know of is due to Jeavons et al. [11], who do not provide any measure of correlation among bit-streams or its impact on SC behavior. They define SNs *X* and *Y* to be *independent* if $p_{X\wedge Y} = p_X p_Y$, where $X \wedge Y$ is the SN obtained by ANDing *X* and *Y*. This, in effect, says that a stochastic multiplier's inputs are uncorrelated if the output is accurate. It gives no hint of what happens with correlation present. Ma et al. [15] discuss the propagation of inaccuracy through stochastic circuits under the assumption of independent inputs, and do not analyze correlated inputs.

This paper introduces a rigorous measure of correlation among SNs, and shows how to analyze circuits with correlated inputs. It leads to an interesting conclusion: contrary to general belief, correlation is not always harmful in SC. In fact, it can even be exploited to design better circuits! Fig. 3 gives a motivating example. The multiplexer implements $p_Z = (1 - p_W)p_X + p_W p_Y$, which, with p_W = 0.5, is the scaled sum $p_Z = (p_X + p_Y)/2$. If all inputs are independent as in Fig. 3a, the circuit performs the add operation accurately. In Fig. 3b, every 0 of *X* coincides with a 1 from *Y*, implying that *X* and *Y* are (negatively) correlated, but the circuit still computes accurately. This will be explained in Sec. II.

The next example illustrates a situation where, counter to most intuition, correlation is actually beneficial. The absolute-valued subtraction function $p_Z = |p_X - p_Y|$ is useful in image processing [3]. If it is implemented under the usual assumption of independent inputs, a large stochastic circuit is needed [13]. However, it can also be implemented by a single XOR gate fed with highly correlated inputs in which there is maximum overlap of 1s and 0s between the two input SNs. In that case, the probability of getting two 1s (or two 0s) on *x* and *y* is $\min(p_X, p_Y)$ (or $\min(1 - p_X, 1 - p_Y)$), implying that the probability of different values on *x* and *y* becomes $p_X - p_Y$ if

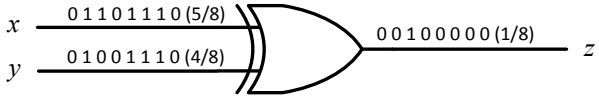

Figure 4. XOR gate with correlated inputs implementing absolute-valued subtraction.

$p_X > p_Y$, as in Fig. 4, or else $p_Y - p_X$ if $p_X < p_Y$. In other words, the output represents $|p_X - p_Y|$. Note that an XOR fed with independent SNs *X* and *Y* implements an entirely different function, namely, $p_Z = p_X(1 - p_Y) + p_Y(1 - p_X)$.

These examples demonstrate correlation's significance in SC, and suggest that advantage can be taken of correlation in certain cases. We introduce here a measure of correlation for SC and a method of analyzing SC circuits based on the probabilistic transfer matrix (PTM) methodology [12]. We also develop a method of synthesizing combinational circuits that work in the presence of correlation. Finally, we discuss the analysis of sequential stochastic circuits, which is very poorly understood at present. Prior work is limited to a few special cases with regular structures [4][14]. General sequential circuits are difficult to analyze because of correlation among the state variables. The correlation analysis of this paper provides a way to tackle these circuits.

The main contributions of this paper are

- A new correlation measure *SCC* for SNs
- Use of PTMs to analyze stochastic circuits
- A study of combinational circuits with correlated inputs and their application to some useful SC designs
- Analysis of general sequential stochastic circuits

The paper is organized as follows. Sec. II discusses PTMs and their role in SC. A rigorous analysis of correlation in the SC context is presented in Sec. III. Then Sec. IV deals with combinational stochastic circuits, while Sec. V addresses sequential circuits. Some conclusions are drawn in Sec.VI.

II. Probabilistic Transfer Matrices

The probabilistic transfer matrix (PTM) [12] has proven valuable in the probabilistic analysis of logic circuits, such as circuits subject to soft errors. Here we show that PTMs are also useful for analyzing correlation in stochastic circuits. In the usual SC scenario, a circuit with *m* inputs, $x_1, x_2, \ldots, x_m$ is analyzed using the probability values of *m* bit-streams applied to its input lines, assuming that these bit-streams are independent. In the PTM formulation, the same input data is represented by a stochastic vector *V* of size 2^m. The elements of *V* are the probabilities of the possible input combinations of $x_1x_2\ldots x_m$, which can vary from 00…0 to 11…1. For example, V_1 = [1/2 0 0 1/2] is the PTM corresponding to two inputs *x* and *y*, when the probability of *xy* = 00 and *xy* = 11 is 1/2 and the other probabilities are zero. The PTM has more information than conditional probability values—it also implicitly contains correlation information. For example, V_1 = [1/2 0 0 1/2] indicates that SNs *X* and *Y* are highly correlated because the probability of having different values on *x* and *y* is

always zero. The vector V_2 = [1/4 1/4 1/4 1/4], on the other hand, represents two completely uncorrelated SNs.

Every logic circuit has a PTM representing its error-free function. This is a matrix of size $2^m \times 2^l$, where m and l are the numbers of inputs and outputs of the circuit, respectively. The PTM of an AND gate implementing $z = x \wedge y$ is

$$\begin{array}{c} \\ xy \end{array} \begin{array}{c} z \\ \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

The entry $p_{i,j}$ is the (conditional) probability of input i producing output j. Multiplying an input PTM by a circuit PTM yields an output PTM, which for a single-output gate is a vector $[o_0\ o_1]$ where o_0 and o_1 are the probabilities of 0 and 1, respectively, appearing at the output. For example, the SC operation performed by the XOR circuit of Fig. 4 is described by the PTM calculation

$$[3/8 \quad 0 \quad 1/8 \quad 4/8] \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = [7/8 \quad 1/8]$$

It was stated earlier that the adder of Fig. 3 has input SNs X and Y which can be correlated without causing inaccuracy. We now prove this via PTM analysis. The input vector has the form $I = [i_0\ i_1\ i_2\ i_3]$, in which i_0, i_1, i_2 and i_3 denote the probability of xy being 00, 01, 10 and 11, respectively. The input SN W is a constant of value 1/2 and so is omitted from the PTM. The adder's PTM is therefore

$$\begin{bmatrix} 1 & 0 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \\ 0 & 1 \end{bmatrix}$$

The corresponding output vector is given by

$$[i_0 \quad i_1 \quad i_2 \quad i_3] \times \begin{bmatrix} 1 & 0 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \\ 0 & 1 \end{bmatrix} = \left[i_0 + \frac{i_1 + i_2}{2} \quad i_3 + \frac{i_1 + i_2}{2} \right]$$

Note that $p_X = i_2 + i_3$ and $p_Y = i_1 + i_3$ always holds, whether X and Y are independent or correlated. Hence, $p_Z = 1/2(p_X + p_Y)$. The input vectors [1/8 3/8 1/8 3/8] and [0 1/2 1/4 1/4] corresponding to the inputs X and Y of Figs. 3a and 3b, respectively, both lead to the same output vector [3/8 5/8].

III. Correlation of Stochastic Numbers

Correlation refers to statistical similarity between two phenomena. As discussed in detail in [10], the correlation of two sequences (bit-streams) is measured by some form of covariance or dot-product operation. With appropriate normalization, a correlation value of +1 means maximum similarity, a correlation value −1 means minimum similarity (maximum difference), and a correlation of 0 means the sequences are uncorrelated. While many measures of similarity exist [5], they are not very useful in the SC context. The standard definition of correlation (also known as the Pearson correlation [5]) $\rho(X,Y)$, in particular, is unsuitable because it imposes constraints on the expected value of the bit-streams. For example, $\rho = +1$ implies that the bit-streams must be identical. A suitable similarity measure should be independent of, or orthogonal to, the data values; in other words, it should not impose constraints on the data. We therefore propose a new correlation measure defined as follows.

Definition 1: The *SC correlation SCC*(X, Y) of two SNs X and Y is given by

$$SCC(X,Y) = \begin{cases} \dfrac{p_{X\wedge Y} - p_X p_Y}{\min(p_X, p_Y) - p_X p_Y} & \text{if } p_{X\wedge Y} > p_X p_Y \\ \dfrac{p_{X\wedge Y} - p_X p_Y}{p_X p_Y - \max(p_X + p_Y - 1, 0)} & \text{otherwise} \end{cases} \quad (1)$$

The starting point in constructing *SCC*(X, Y) is obtaining the bit-wise AND function $X \wedge Y$ (a kind of dot-product) of the SNs, that is, finding $p_{X\wedge Y}$. This is then centralized by the uncorrelated value $p_X p_Y$, yielding $p_{X\wedge Y} - p_X p_Y$. Finally, the centralized value is normalized by dividing it by the maximum possible values. The centralization makes *SCC* consistent with the definition of independence in [11] when $SCC(X,Y) = 0$, i.e., when $p_{X\wedge Y} - p_X p_Y = 0$. The normalization guarantees that for two maximally similar (or different) SNs X and Y, we get $SCC(X,Y) = +1$ (or −1). Unlike the standard correlation measure $\rho(X,Y)$, *SCC* does not vary with the SN values. All the intermediate values of *SCC* are linearly interpolated between the independent case and the maximum similarity (or difference) case. For example, $S(X,Y) = 0.5$ means that $p_{X\wedge Y}$ is half-way between $p_X p_Y$, i.e., the independent case, and $\min(p_X, p_Y)$, i.e., the maximum overlap case.

We can also define *SCC* using the notation of [5], which allows easy comparison with other correlation concepts. For two n-bit SNs X and Y, denote the number of overlapping 1s by a, the number of overlapping 1s of X and 0s of Y by b, the number of overlapping 0s of X and 1s of Y by c, and the number of overlapping 0s on both SNs by d. Clearly, $a + b + c + d = n$. We then have the following definition which is equivalent to Def. 1:

$$SCC(X,Y) = \begin{cases} \dfrac{ad - bc}{n.\min(a+b, a+c) - (a+b)(a+c)} & \text{if } ad > bc \\ \dfrac{ad - bc}{(a+b)(a+c) - n.\max(a-d, 0)} & \text{otherwise} \end{cases} \quad (2)$$

The numerator $ad - bc$ is common to many similarity measures including Pearson correlation

$$\rho(X,Y) = \frac{ad - bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$$

and it captures the overlap of 0s and 1s in the two bit-streams. The denominator, on the other hand, is simply a normalization factor. While Pearson correlation is normalized by the variance of the bit-streams, *SCC* is normalized so that the bit-streams with maximum (minimum) overlap of 1s and 0s lead to *SCC* = +1 (−1), independent of the values of the SNs.

Table I shows examples of bit-streams with their ρ and *SCC* values. Note that ρ and *SCC* are the same for independ-

TABLE I. SOME SNS WITH THEIR *SCC* AND STANDARD CORRELATION VALUES

Stochastic numbers	SC correlation $SCC(X,Y)$	Standard correlation $\rho(X,Y)$
X = 11110000 Y = 11001100	0	0
X = 11110000 Y = 11110000	+1	+1
X = 11110000 Y = 00001111	–1	–1
X = 11111100 Y = 11110000	1	0.58
X = 11111100 Y = 00001111	–1	–0.58
X = 11111100 Y = 11100001	0	0
X = 11000000 Y = 11111100	1	0.33

ent SNs, and for SNs with equal values. When the SNs have different values, *SCC* consistently gives the value +1 (or –1) for maximum (minimum) overlap of 1s and 0s between the bit-streams, while ρ gives different values. This shows that, unlike ρ, *SCC* is not affected by the values of the bit-streams.

As mentioned earlier, the function of a stochastic circuit can effectively be changed by enforcing correlations among its inputs. The XOR gate of Fig. 4 illustrates this. Fig. 5a shows the stochastic functions implemented by the same XOR gate at different levels of *SCC*. In all cases, the output of the function remains the same at the four corners, but the function changes greatly for the intermediate values. Fig. 5b shows the same function for various fixed values of p_Y and *SCC*.

IV. COMBINATIONAL CIRCUITS

Every stochastic circuit implements a real-valued function F, which is interpreted as its stochastic behavior. For example, the AND gate of Fig. 1 implements the multiplication function $p_Z = F(p_X, p_Y) = p_X p_Y$, assuming $SCC(X,Y) = 0$. The inputs of F are the values of the SNs p_X and p_Y. Hence, to obtain the stochastic behavior of a logic circuit, we need to determine its corresponding probability function F.

We saw earlier that a circuit's functionality can change in the presence of correlation. The AND gate, for example, implements $p_Z = F(p_X, p_Y) = \min(p_X, p_Y)$ if $SCC(X,Y) = +1$.

Table II. FUNCTIONS FOR THE PTM ELEMENTS OF A TWO-INPUT CIRCUIT

	F_0 (p_X, p_Y) SCC = 0	F_{-1} (p_X, p_Y) SCC = –1	F_{+1} (p_X, p_Y) SCC = +1
i_0	$(1-p_X).(1-p_Y)$	$\max(1-p_X-p_Y, 0)$	$\min(1-p_X, 1-p_Y)$
i_1	$(1-p_X).p_Y$	$\min(1-p_X, p_Y)$	$\max(p_Y-p_X, 0)$
i_2	$(1-p_Y).p_X$	$\min(1-p_Y, p_X)$	$\max(p_X-p_Y, 0)$
i_3	$p_X.p_Y$	$\max(p_X+p_Y-1, 0)$	$\min(p_X, p_Y)$

Based on Def. 1 and the subsequent discussion, the SC function of the AND gate for the case of, say $SCC(X,Y) = 0.5$, is half-way between its SC function for $SCC(X,Y) = +1$ and $SCC(X,Y) = 0$. In general, we can express a circuit's functionality as a linear combination of its functions at $SCC(X,Y) = 0$ and $SCC(X,Y) = +1$ or -1. Hence for any *SCC*, p_Z can be written as

$$p_Z(p_X,p_Y) = \begin{cases} (1+SCC).F_0(p_X,p_Y) - SCC.F_{-1}(p_X,p_Y) & \text{if } SCC < 0 \\ (1-SCC).F_0(p_X,p_Y) + SCC.F_{+1}(p_X,p_Y) & \text{otherwise} \end{cases} \quad (3)$$

where $F_0(p_X,p_Y)$, $F_{-1}(p_X,p_Y)$ and $F_{+1}(p_X,p_Y)$ are the functions of the same circuit at $SCC(X,Y) = 0, -1$ and $+1$, respectively. For the AND gate example, we have $F_0(p_X,p_Y) = p_X p_Y$, $F_{-1}(p_X,p_Y) = \max(p_X + p_Y - 1, 0)$, and $F_{+1}(p_X,p_Y) = \min(p_X,p_Y)$.

In order to derive the stochastic behavior of a circuit C with correlated inputs, we use the PTM tools discussed in Sec. II. If C has two input bit-streams X and Y with correlation *SCC*, construct the input PTM $I = [i_0\ i_1\ i_2\ i_3]$, in which the i_k's are expressed in the form of Eq. (3). The F_0's, F_{-1}'s, and F_{+1}'s corresponding to each i_k are defined in Table II. From these, we can extract C's stochastic behavior by multiplying the vector I by the circuit PTM. For instance, if C is an XOR gate, $[i_0\ \ i_1\ \ i_2\ \ i_3]\begin{bmatrix}1&0\\0&1\\0&1\\1&0\end{bmatrix}$ yields the functions illustrated in Fig. 5a for some representative values of *SCC*.

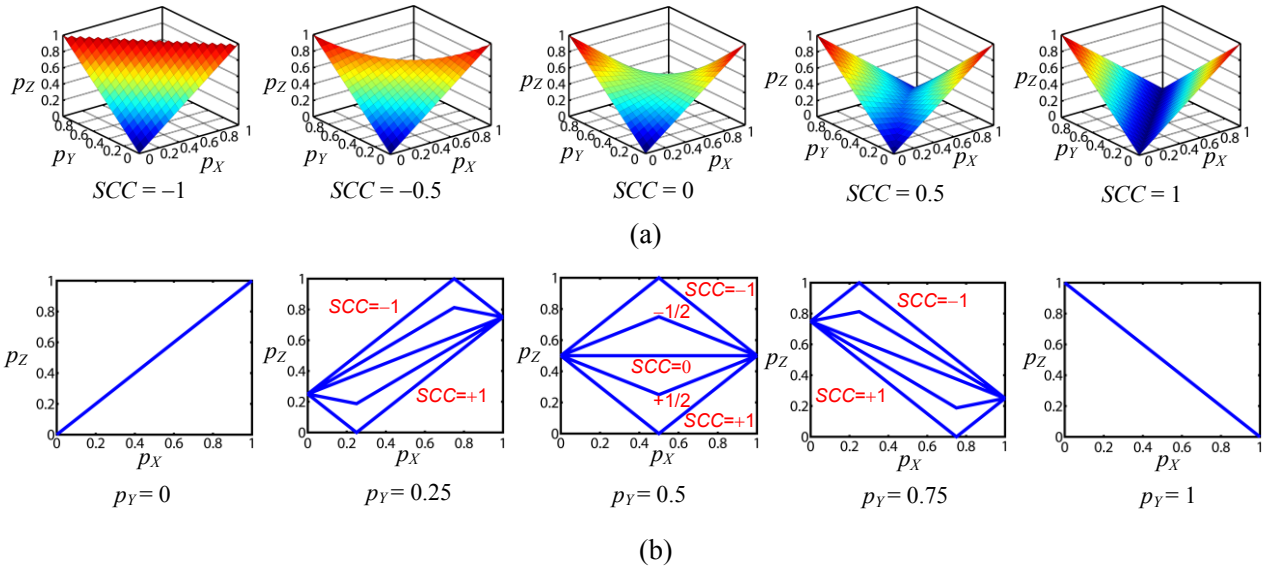

Figure 5. Functions implemented by an XOR gate (a) with different input *SCC* values, and (b) with fixed values of p_Y.

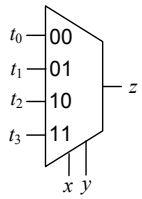

Figure 6. High-level structure of a synthesized two-input circuit with correlated inputs, prior to simplification.

Only correlation corresponding to the special case $SCC = 0$ has been considered in the literature. As we have just seen, other cases such as $SCC = +1$ lead to useful results. The PTM formulation of two-input SC circuits discussed above points to a method for synthesizing stochastic circuits with correlated inputs. In this approach, a target function $F(p_X, p_Y)$ is approximated by a function $p_Z = F'(p_X, p_Y)$ defined by

$$[i_0 \quad i_1 \quad i_2 \quad i_3] \times \begin{bmatrix} 1-t_0 & t_0 \\ 1-t_1 & t_1 \\ 1-t_2 & t_2 \\ 1-t_3 & t_3 \end{bmatrix} = [1-p_Z \quad p_Z] \qquad (4)$$

in which the t_k's are the parameters of F' to be determined and the i_k's are expressed in the form of Eq. (3) and Table II. The process of approximation is to find the best t_k's and the best *SCC* for which the following error function ε is minimized.

$$\varepsilon = \iint_{0,0}^{1,1} (F(p_X, p_Y) - F'(p_X, p_Y))^2 dp_X dp_Y \qquad (5)$$

This is done by adjusting the t_k's and the *SCC* using standard optimization techniques. Because of the limited number of parameters and the well-behaved error function, this problem is relatively easy to solve. Once the parameters are found, F' can be realized by a logic circuit with the overall multiplexer-style structure shown in Fig. 6, The constant SNs needed for the four t_k's can be generated by up to four copies of the circuit in Fig. 2c. The resulting circuit can then be further optimized using conventional logic synthesis methods and tools.

Generating two SNs *X* and *Y* with a desired level of *SCC* is a problem that has not been studied before. We propose to use the circuit structure shown in Fig. 7, which generates *X* by

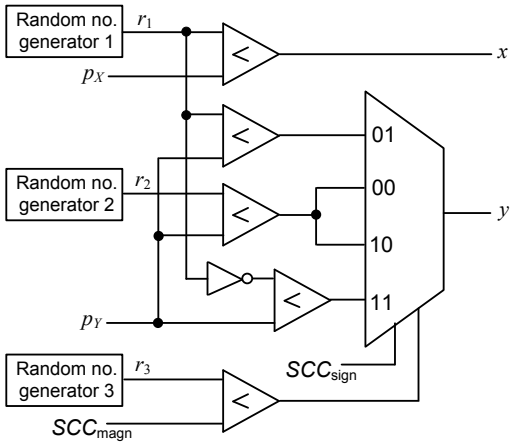

Figure 7. Generating SNs with a specified *SCC*.

Step 1: Determine a suitable approximating function $p_Z = F'(p_X, p_Y)$ according to Eq. (4) in which the input vector I is defined by Eq. (3) and Table II.

Step 2: Determine the error function ε given by Eq. (5).

Step 3: Minimize ε by adjusting the t_k and *SCC* parameters in F'.

Step 4: Insert these parameters into the structure of Fig. 6, and use standard logic synthesis methods to optimize the resulting circuit.

Figure 8. Algorithm CCC to synthesize a stochastic function $F(p_X, p_Y)$ with correlated inputs.

means of a standard SN generator and, depending on the sign SCC_{sign} and magnitude SCC_{magn} of *SCC*, mixes uncorrelated and correlated versions of *Y* together. For example, if *SCC* = +1, then *Y* is generated from the same random number source used by *X*, so *X* and *Y* become highly correlated. Note that Fig. 7 is a *programmable* structure, and not all the components shown are needed in every design. For example, if a circuit requires *X* and *Y* to be generated with *SCC* = +1, then the select inputs *xy* of the multiplexer are set to 01, implying that random number generators 2 and 3, and their associated circuits can be removed. Fig. 8 summarizes our proposed *correlated combinational circuit* (CCC) synthesis algorithm.

As an example, consider the problem of synthesizing a circuit for the target function $F(p_X, p_Y) = \min(p_X, p_Y)$. In Step 1 of CCC, $p_Z = F'(p_X, p_Y)$ is prepared in the form of Eq. (4), and in Step 2 the error function ε is prepared in the form of Eq. (5). Then, the t_k's and *SCC* are adjusted until ε is minimized. For the running example, $\varepsilon = 0$, i.e., the exact target function, can be achieved by assigning $t_0 = 0$, $t_1 = 0$, $t_2 = 0$, $t_3 = 1$, and *SCC* = +1. On plugging these values into the circuit of Fig. 6, we obtain that of Fig. 9a, i.e., an AND gate. Observe that the AND implements $\min(p_X, p_Y)$ only if its inputs have *SCC* = +1. In order to generate *X* and *Y*, we use the circuit of Fig. 7 and plug in *SCC* = +1 (SCC_{sign} = 0, SCC_{magn} = 1), yielding the circuit of Fig. 9b. This circuit is smaller than one employing two of the independent SN generators in Fig. 2c, so generating correlated SNs is cheaper than generating independent ones in this case.

Table III shows examples of circuits synthesized by the CCC algorithm. Most of the target functions are useful non-linear functions that have no efficient stochastic implementation when the inputs are uncorrelated. The last synthesized function in the table is the multiplexer-based scaled adder in which correlation of the input data does not matter. Another type of SC adder, a saturating adder, is also shown. This

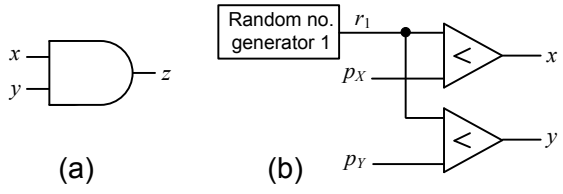

Figure 9. Implementing the function $F(p_X, p_Y) = \min(p_X, p_Y)$: (a) synthesized stochastic circuit, and (b) corresponding SN generator.

TABLE III. EXAMPLES OF SYNTHESIZED STOCHASTIC CIRCUITS EXPLOITING VARIOUS CORRELATION LEVELS

Target function	$[t_0 \quad t_1 \quad t_2 \quad t_3]$	*SCC*	Synthesized circuit
$p_Z = \min(p_X, p_Y)$	[0 0 0 1]	+1	AND gate with positively correlated inputs
$p_Z = \max(p_X, p_Y)$	[0 1 1 1]	+1	OR gate with positively correlated inputs
$p_Z = \lvert p_X - p_Y \rvert$	[0 1 1 0]	+1	XOR gate with positively correlated inputs; implements absolute-valued subtraction
$p_Z = \min(p_X + p_Y, 1)$	[0 1 1 1]	–1	OR gate with negatively correlated inputs; implements saturating add
$p_Z = 1/2(p_X + p_Y)$	[0 1/2 1/2 1]	Any	Multiplexer with arbitrary correlation among its data inputs; implements scaled add

circuit adds its inputs without scaling until the saturating value 1 is reached. Finally, observe that in some cases, the circuits synthesized by CCC are the same as the standard designs. For example, the smallest SC multiplier is the AND gate of Fig. 1, which requires uncorrelated inputs. This shows that the CCC is capable of replicating circuits synthesized by existing methods, because *SCC* = 0 is also allowed in CCC.

Table IV compares the circuits synthesized by CCC and those designed by the spectral synthesis method of [1], which makes the usual independent-inputs assumption, i.e., *SCC* = 0. In addition to the circuits of Table III, a few other functions were implemented. Since it is normally impossible to implement real-valued functions exactly, some are approximated before synthesis. Area is estimated by mapping the circuits to a generic library of cells using 0.35μm CMOS technology [19]. For a fair comparison, we also report the measured mean error between the synthesized and target functions *F′* and *F*. The results indicate that in most cases, the circuits synthesized by CCC are smaller and more accurate than those designed by the method of [1].

When dealing with more than two signals, considering their pairwise *SCC* values may be insufficient, as higher-order correlations can exist among groups of three or more of the signals. To handle such cases, we suggest using PTMs that are large enough to embed all the signal correlations of interest. Circuits with many inputs can also be designed by decomposing the target function into subfunctions of two variables. The CCC method can then be used to synthesize the pieces and put them back together. For example, consider the four-variable function $p_Z = 1/2(|p_X - p_Y| + |p_V - p_W|)$, which performs the very useful image-processing task of edge detection [3]. It decomposes into two absolute-valued subtraction functions p'_Z and p''_Z, which are then combined by a scaled add to produce p_Z:

$$p'_Z = |p_X - p_Y|, \quad p''_Z = |p_V - p_W|, \quad p_Z = 1/2(p'_Z + p''_Z)$$

All three of these functions can be synthesized by CCC, as indicated in Table III. Fig. 10 shows the result; the XOR gates perform absolute-valued subtraction, while the multiplexer performs scaled addition. Note that the select input of the multiplexer is fed by an auxiliary input *r* with $p_r = 1/2$.

TABLE IV. COMPARISON BETWEEN CIRCUITS SYNTHESIZED IN THIS PAPER AND THOSE SYNTHESIZED BY THE SPECTRAL METHOD OF [1]

Target function	Synthesis method and correlation assumption	Area* (μm^2)	Mean Error (%)
$p_Z = \min(p_X + p_Y, 1)$ Saturating adder	[1] with *SCC* = 0	1,628	10
	CCC with *SCC* = –1	1,091	0
$p_Z = \max(p_X - p_Y, 0)$ Saturating subtracter	[1] with *SCC* = 0	1,663	10
	CCC with *SCC* = +1	1,188	0
$p_Z = p_X \times p_Y$ Multiplier	[1] with *SCC* = 0	1,646	0
	CCC with *SCC* = 0	1,646	0
$p_Z = (p_X + p_Y)/2$ Scaled adder	[1] with *SCC* = 0	1,857	0
	CCC with *SCC* = +1	1,320	0
$p_Z = (p_X p_Y)^{0.45}$	[1] with *SCC* = 0	1,980	12
	CCC with *SCC* = +1	1,443	7
$p_Z = (1 - p_X - p_Y)^2$	[1] with *SCC* = 0	2,306	15
	CCC with *SCC* = –1	1,760	9
A multi-variate polynomial	[1] with *SCC* = 0	2,086	9
	CCC with *SCC* = –1/2	2,473	4

* Circuits with uncorrelated (independent) inputs were synthesized according to [1] with polynomials of degree 1. All the reported area numbers include stochastic number generators.

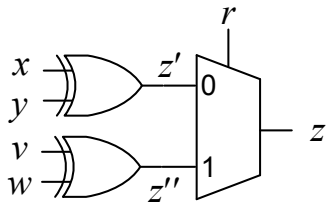

Figure 10. Stochastic circuit for image edge-detection [3].

V. SEQUENTIAL CIRCUITS

As with conventional binary logic, stochastic sequential circuits (stochastic FSMs) can lead to more efficient designs than combinational ones. For instance, Fig. 11 shows a sequential update node of the type commonly used in stochastic LDPC decoders [7]. It implements the function

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

for which no more efficient combinational equivalent is known. Sequential stochastic circuits have also been proposed to implement arithmetic functions such as division and hyperbolic tangent [4][6][14]. However, the proposed designs are mostly ad hoc, or require state transition diagrams of a very restricted structure [4][14]. Analyzing stochastic sequential circuits with arbitrary state transition diagrams is difficult since the state variables tend to be correlated. For example, in the three-state circuit of Fig. 12, the state 11 never occurs, so the SNs corresponding to state bits w_0 and w_1 are correlated. A general method to analyze and design arbitrary stochastic sequential circuits does not presently exist.

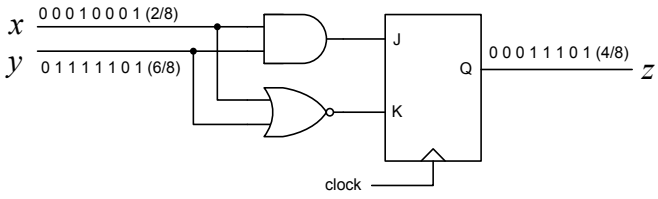

Figure 11. Stochastic update node used in LDPC decoders.

A sequential logic circuit implements two combinational functions: the next-state function δ that, given the current state and the current inputs, produces the next state, and the output function λ that produces the corresponding output signals. In the context of SC, next-state transitions are treated as probabilistic. Hence, we are mainly interested in the stochastic behavior of the state-defining function δ, since λ is a simple combinational function and can be analyzed by existing methods.

Like our approach to combinational circuits with correlation, we use PTMs to analyze sequential circuits. The state probability distribution of a sequential circuit is represented by a vector $S = [s_1\ s_2\ \dots\ s_l]$ in which s_i denotes the probability of being in state i. For example, the vector [1/3 1/3 1/3] corresponding to the circuit of Fig. 12 indicates that the probability of being in each state is 1/3. The state transition behavior of the sequential circuit can be expressed as a *transition matrix* T, in which element t_{ij} denotes the probability of a transition from state i to j. The transition matrix corresponding to Fig. 12 is

$$T = \begin{bmatrix} 1-p_X & p_X & 0 \\ 1-p_X & 0 & p_X \\ 1-p_X & 0 & p_X \end{bmatrix}$$

Given the state probability distribution $S(t)$ in a particular clock cycle t and the transition matrix T, we can write the state distribution of the next clock cycle $t+1$ as $S(t+1) = S(t) \times T$. For instance, with $S(t) = [1/3\ 1/3\ 1/3]$ and $p_X = 1$, we get $S(t+1) = [0\ 1/3\ 2/3]$.

After enough clock cycles, the state distribution of the sequential circuit typically converges to a stationary distribution π, which can represent the circuit's stochastic behavior. Fortunately, finding π is a well-known mathematical problem. The transition matrix T of a sequential circuit can be interpreted as a Markov chain [8], which is an FSM with probabilistic state transitions. The stationary state π of a Markov chain is an eigenvector of T with the defining property $\pi = \pi \times T$. For example, the stationary distribution of the circuit in Fig. 12 is given by

$$\pi = [1-p_X \quad p_X - {p_X}^2 \quad p_X^2] \times \begin{bmatrix} 1-p_X & p_X & 0 \\ 1-p_X & 0 & p_X \\ 1-p_X & 0 & p_X \end{bmatrix}$$
$$= [1-p_X \quad p_X - {p_X}^2 \quad p_X^2]$$

And since the output z only becomes 1 in the state $w_1w_0 = 01$, i.e., the second state, we conclude that $p_Z = p_X - {p_X}^2$.

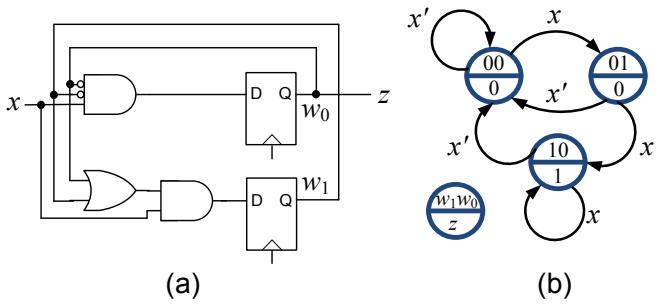

Figure 12. (a) A sequential stochastic circuit C and (b) its state transition diagram.

In order to find the stationary distribution of an l-state FSM with an arbitrary transition matrix T, we need to solve the equation $\pi = \pi \times T$ and compute the elements of $\pi = [s_1\ s_2\ \dots\ s_l]$. The equation corresponding to each state is of the form $s_i = G_i(s_1, \dots, s_l)$, in which the G_i's collectively represent the stochastic behavior of the next-state function δ of the FSM. As noted earlier, the function δ is combinational, and according to [1], its stochastic behavior can be represented as a polynomial. This implies that all the G_i's, and hence all the elements of T, are polynomial functions with respect to the input SN values. This fact allows us to obtain an analytical solution to $\pi = \pi \times T$ and leads to the following result.

Theorem: Given a sequential circuit with m inputs $x_1,\dots,x_m$, and a transition matrix T whose elements are polynomial functions of $p_{X_1}, \dots, p_{X_m}$, its stationary state distribution π has elements of the form $F_1(p_{X_1}, \dots, p_{X_m}) / F_2(p_{X_1}, \dots, p_{X_m})$, where F_1 and F_2 are polynomial functions.

The theorem follows from the fact that in solving $\pi = \pi \times T$, the coefficients of the variables s_i become parameterized polynomial functions. The standard row-transformation steps in solution methods like Gaussian elimination only apply the operations addition, multiplication, and division to the matrix elements. Since these are polynomial functions, adding and multiplying them produces polynomials, and dividing them gives rational polynomials. The degrees of the polynomials depend on the number of states l and the degrees of the polynomial elements of T. If these polynomials are linear, the final solution is of degree 2^{l-2}.

For example, the transfer matrix corresponding to the two-state LDPC update circuit of Fig. 11 is

$$T = \begin{bmatrix} 1-p_Xp_Y & p_Xp_Y \\ (1-p_X)(1-p_Y) & p_X+p_Y-p_Xp_Y \end{bmatrix}$$

To find the stationary distribution of T, we form the equations

$$s_0 = s_0(1-p_Xp_Y) + s_1((1-p_X)(1-p_Y))$$
$$s_1 = s_0(p_Xp_Y) + s_1(p_X+p_Y-p_Xp_Y)$$

Since the circuit is either in state s_0 or s_1, we have the additional constraint $s_0 + s_1 = 1$. Hence, $s_1 = 1 - s_0$, yielding

$$s_1 = (1-s_1)(p_Xp_Y) + s_1(p_X+p_Y-p_Xp_Y)$$
$$s_1(1+p_Xp_Y-p_X-p_Y+p_Xp_Y) = p_Xp_Y$$

Thus, we obtain the expected function [7]:

$$p_Z = s_1 = \frac{p_X p_Y}{p_X p_Y + (1-p_X)(1-p_Y)}$$

Another example is a four-state sequential circuit from [14], which has the transition matrix

$$T = \begin{bmatrix} 1-p_X & p_X & 0 & 0 \\ 1-p_X & 0 & p_X & 0 \\ 0 & 1-p_X & 0 & p_X \\ 0 & 0 & 1-p_X & p_X \end{bmatrix}$$

Now $\pi = \pi \times T$ yields the following set of equations:

$$s_0 = s_0(1-p_X) + s_1(1-p_X)$$
$$s_1 = s_0 p_X + s_2(1-p_X)$$
$$s_2 = s_1 p_X + s_3(1-p_X)$$
$$s_3 = s_2 p_X + s_3 p_X$$

with the additional constraint

$$s_0 + s_1 + s_2 + s_3 = 1$$

Solving these equations analytically we obtain $s_0 = (1-p_X)^3/\left(2{p_X}^2 - 2p_X + 1\right)$, etc., which is consistent with the analysis of [14].

VI. Conclusions

Stochastic computing (SC) has recently re-emerged as an attractive alternative technique for some important computing tasks with extreme demands for small size and low power. A key unsolved problem in designing stochastic circuits has been to overcome the computational inaccuracies that result from undefined and undesired correlations among signals. The usual solution has been to avoid correlation entirely at the cost of introducing many independent stochastic number sources or re-randomizers.

This paper has investigated in depth the impact of correlation on stochastic computing. We have shown that not all correlation is harmful. In fact, contrary to what one would intuitively expect, correlation can serve as a resource in designing stochastic circuits. We have given the first general and rigorous definition of correlation for SC, which has enabled us to analyze both combinational and sequential stochastic circuits in the presence of correlation. We demonstrated how probabilistic transfer matrices aid this analysis, and lead to a general approach to designing stochastic circuits with correlated inputs. We further demonstrated how to implement various useful functions such as saturating addition and subtraction that had no previous efficient SC implementations. We reported a comparative study indicating that the circuits with correlated inputs are generally smaller and more accurate than those with independent inputs. Finally, we presented the first systematic analysis of sequential stochastic circuits, a problem which has generally resisted attack since the 1960s.

Acknowledgements

This work was supported by Grant CCF-1017142 from the U.S. National Science Foundation. The authors are grateful to Igor L. Markov for helpful discussions.

References

[1] Alaghi, A. and Hayes, J.P., "A spectral transform approach to stochastic circuits," *Proc. ICCD*, pp. 315-321, 2012.

[2] Alaghi, A. and Hayes, J.P., "Survey of stochastic computing," *ACM Trans. Embedded Computing Systems*, 12, no. 2s, pp. 92:1-92:19, May 2013.

[3] Alaghi, A., Li, C. and Hayes, J.P., "Stochastic circuits for real-time image-processing applications," *Proc. DAC*, pp. 136:1-6, June 2013.

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

[5] Choi, S.S., Cha, S.H. and Tappert, C., "A survey of binary similarity and distance measures," *Journ. Systemics, Cybernetics and Informatics*, 8, pp. 43-48, 2010.

[6] Gaines, B.R., "Stochastic computing systems," *Advances in Information Systems Science*, 2, pp. 37-172, 1969.

[7] Gaudet, V.C. and Rapley, A.C., "Iterative decoding using stochastic computation," *Electronics Letters*, 39, 299-301, 2003.

[8] Grinstead, C.M., and Snell, L.J., *Grinstead and Snell's Introduction to Probability*, version of 4 July 2006, American Math. Soc., 2006.

[9] Gross, W.J., Gaudet, V.C. and Milner, A., "Stochastic implementation of LDPC decoders," *Proc. Asilomar Conf.*, pp. 713-717, 2005.

[10] Golomb, S.W. and Gong, G., *Signal Design for Good Correlation,* New York: Cambridge Univ. Press, 2004.

[11] Jeavons, P., Cohen, D.A. and Shawe-Taylor, J., "Generating binary sequences for stochastic computing," *IEEE Trans. Info. Theory*, 40, pp. 716-720, 1994.

[12] Krishnaswamy, S., Viamontes, G.F., Markov, I.L. and Hayes, J.P., "Probabilistic transfer matrices in symbolic reliability analysis of logic circuits," *ACM Trans. Design Automation of Electronic Systems*, 13, no. 1, pp.8:1-8:35, Jan. 2008.

[13] Li, P. and Lilja, D.J., "Using stochastic computing to implement digital image processing algorithms," *Proc. ICCD*, pp. 154-161, 2011.

[14] Li, P., Qian, W., Riedel, M.D., Bazargan, K. and Lilja, D.J., "The synthesis of linear finite state machine-based stochastic computational elements," *Proc. ASP-DAC.*, pp. 757-762, 2012.

[15] Ma, C., Zhong, S. and Dang, H., "Understanding variance propagation in stochastic computing systems," *Proc. ICCD*, pp. 213-218, 2012.

[16] Naderi, A., Mannor, S., Sawan, M. and Gross, W.J., "Delayed stochastic decoding of LDPC codes," *IEEE Tran. Signal Proc.,* 59, pp. 5617-5626, 2011.

[17] Poppelbaum, W.J., Afuso, C. and Esch, J.W., "Stochastic computing elements and systems," *Proc. AFIPS Fall Joint Computer Conf.*, pp. 635-644, 1967.

[18] Qian, W., LI, X., Riedel, M.D., Bazargan, K. and Lilja, D.J., "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. Computers*, 60, pp. 93-105, 2011.

[19] Sentovich, E.M. *et al*., "SIS: A System for sequential circuit synthesis," Univ. of California, Berkeley, Tech. Report UCB/ERL M92/41, Electronics Research Lab, 1992.