Mitigating the Correlation Problem in Multi-Layer Stochastic Circuits

Owen Hoffend & John P. Hayes
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109 USA
{ohoffend, jhayes}@umich.edu

2023 IEEE International Conference on Rebooting Computing (ICRC) | 979-8-3503-8204-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRC60800.2023.10386939

***Abstract*—Stochastic computing is a low-cost non-standard computer architecture that processes pseudo-random bitstreams. Its effectiveness, and that of other probabilistic methods, requires maintaining desired levels of correlation among interacting input bitstreams, for example, $SCC = 0$ or $SCC = +1$, where SCC is the stochastic cross-correlation metric. Correlation errors are systematic (bias-causing) errors that cannot be corrected by increasing bitstream length. A typical stochastic design C_1 only controls correlation at its primary input lines. This is a fairly straightforward task, however it limits the scope of SC to "single layer," usually combinational, designs. In situations where a second processing layer C_2 follows C_1, the output correlation of C_1 must satisfy the input correlation needs of C_2. This can be done by inserting a (sequential) correlation control layer S_{12} between C_1 and C_2, which incurs high area and delay overhead. S_{12} transforms intralayer bitstreams Z with unknown or undesired SCC values into numerically equivalent ones Z* with desired correlation. The fundamental problem of designing C_1 to produce Z* directly, thereby dispensing with S_{12}, which apparently has not been considered before, is addressed in this paper. We focus on two-layer designs C_1C_2 requiring SCC = +1 between layers, and present a method called COMAX for (re)designing C_1 so that it outputs bitstreams with correlation that is as close as possible to +1. We demonstrate on a representative image processing application that, compared to alternative correlation control techniques, COMAX reduces area by about 50% without reducing output image quality.**

Keywords—stochastic computing, approximate computing, correlation control, logic synthesis, image processing, edge detection

I. Introduction

Stochastic computing (SC) is a probabilistic and approximate design paradigm that computes with sequences of randomized bitstreams known as stochastic numbers (SNs), instead of conventional "binary" or base-2 numbers [1][2]. SC systems typically have much lower hardware area requirements than binary, i.e., non-stochastic, systems, at the cost of lower accuracy. Although originally proposed in the 1960s [1], SC has only recently gained research traction due to the growing need for hardware-intensive "smart" applications. SC's area-accuracy trade-offs offer considerable advantages over binary systems for computing tasks that tolerate small inaccuracies such as digital filtering [3] [4], image processing [5] [6], and neural networks [7] [8] [9]. However, SC's randomness also entails new behavior issues such as correlation, which is very hard to manage and is addressed in this paper.

A typical SC system must interface with a conventional binary system. It therefore begins with a binary-to-stochastic converter (BSC) that maps the input binary data values $\boldsymbol{B} = B_1, B_2, \ldots, B_n$ into SNs of the form $X_i = 101110011$, where X_i's expected numerical value P_{X_i} is proportional to B_i, and can be interpreted in terms of a probability. For example, with unipolar encoding, P_{X_i} is the probability that an arbitrary bit of X_i is 1. The BSC is the primary source of randomness. The SNs $\mathbf{X}$ it generates are fed to a stochastic circuit C for processing, and the resulting output SNs $\boldsymbol{Z}$ are converted back to binary using a stochastic-to-binary converter (SBC). Fig. 1 illustrates this basic SC architecture. Note that C's hardware is a conventional Boolean logic circuit designed to process SNs, so it can be constructed from any standard CMOS technology.

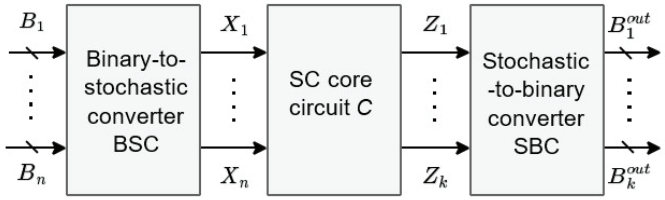

Fig. 1. Basic structure of an SC system: n binary inputs are converted into SNs by a BSC, processed by a stochastic circuit C into k output SNs, then converted back to binary by an SBC.

The stochastic circuit C has both a logic function and an arithmetic (stochastic) function associated with it. For example, if C is an XOR logic gate, it performs a type of arithmetic subtraction on its input SNs. Thus, subtraction is a stochastic function associated with the XOR logic function. In general, the type and accuracy of C's stochastic function depend on both the length N of its SNs and the correlations between them. A consequence of these dependencies is that stochastic functions are approximate, and controlling accuracy is a challenging and application-dependent problem in SC.

Correlation among C's input SNs greatly influences the stochastic function it computes. For example, consider the circuit shown in Fig. 2a where $N = 8$. If the input SNs $\boldsymbol{X}$ and $\boldsymbol{Y}$ are uncorrelated, it implements a probabilistic subtract-and-multiply operation:

$$P_Z = P_X(1 - P_Y) \tag{1}$$

The AND gate in Fig. 2a serves as a stochastic multiplier, and the inverter performs negation. Using the standard stochastic cross-correlation metric SCC [10], complete independence or lack of correlation between SNs X and Y, as in Fig. 2a, is indicated by $SCC = 0$. In Fig. 2b, on the other hand, the SNs have maximum correlation $SCC = +1$ since $X = 1$ whenever $Y =$

1. If all input SN pairs are maximally correlated in this way, the same circuit computes an entirely different stochastic function, namely, saturating subtraction:

$$P_Z = max(0, P_X - P_Y) \quad (2)$$

Most SC operations considered in the literature require $SCC \approx 0$ between all pairs of inputs for acceptable accuracy, and assume, often implicitly, that the BSC is a source of suitably uncorrelated SNs. Correlation-based operations like Eq. (2) which requires $SCC = +1$ are also useful in many applications [4][6][8-10], but operations using most other possible values of SCC (which must lie between +1 and –1) are not. If the input SCC is not properly controlled to the required value, the design may implement a function that is entirely different than that intended. Note that the range of achievable SCC values depends on the BSC design, including the length N of the SNs it produces and their target probability values. Note too that an approximation to a target SCC value may be useful, reflecting the probabilistic nature of SC, but little is known about this.

SCC is most easily controlled at the primary inputs X_i of a stochastic circuit C when the SNs come immediately from a suitably designed BSC. The authors of [10] give a general BSC design method to accomplish this task for specified values of SCC among the input lines $\boldsymbol{X}$ of C. However, it's much more challenging to control the SCC values among the *output* lines $\boldsymbol{Z}$ of C, without repeatedly re-generating the required SCC using SBC and BSC blocks [11]. We call the general task of designing C with specific SCC output values the *correlation control problem* (CCP).

To illustrate the CCP, consider Fig. 3a, where the core SC computation block C from Fig. 1 is broken into two subcircuits, C_1 and C_2, such that the input SNs for C_2 come from the output SNs of C_1. Circuits composed in this layered way are found throughout digital design for both the SC and conventional binary domains. For example, an SC image processing circuit might start with a smoothing filter for C_1, followed by an edge detector for C_2, as in [12]. If, however, the outputs of C_1 are not properly correlated to satisfy the input SCC requirements of C_2, ($SCC = +1$ in the edge detection case) then the C_2 layer can be expected to have a significant functional error. Correlation errors are systematic or bias-causing errors that cannot be mitigated by increasing the length of the SN bitstreams. Unlike other forms of SC error [4], therefore, they must be anticipated and corrected during the circuit design phase.

The standard approach to solving the CCP in the SC literature has been to introduce some type of sequential correlation correction block S_{12} between layers C_1 and C_2, as shown in Fig. 3b. A brute-force approach is to completely re-generate the bitstreams by inserting an extra SBC followed by a BSC [11]. This method does not require knowing the output SCC values produced by C_1, and can generate the exact SCC required by C_2, but at a very high area and latency cost [23]. Some previous work has proposed smaller and/or faster sequential correlation correction designs by approximating the target SCC instead of guaranteeing it [11-14].

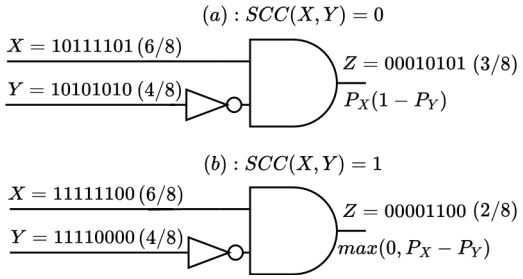

Fig. 2. (a): AND gate circuit $Z = XY'$ computing subtract-and-multiply $P_Z = P_X(1 - P_Y)$ when SCC = 0. (b): Saturating subtraction function $P_Z = max(0, P_X - P_Y)$ computed when SCC = 1.

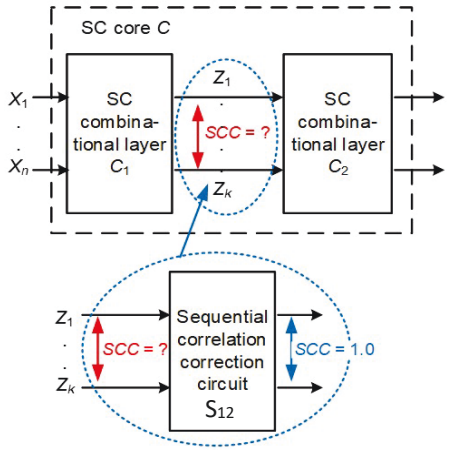

Fig. 3. (a): Stochastic circuit broken into two sub-circuit layers C_1 and C_2. (b): Sequential correlation correction circuit S_{12} inserted to improve SCC between internal signals, in this case increasing SCC toward $SCC = +1$. COMAX designs C_1 in a way that eliminates the need for the correlation correction circuit.

For example, [12] uses a finite-state machine called a "synchronizer" to increase SCC between two SNs X and Y without changing their probability values by saving and later pairing up unpaired 1s. However, these types of designs usually have a very large area footprint relative to C_1 and C_2, owing mainly to their sequential nature. For instance, the design in [12] increases the area cost of its image processing application by a factor of 1.49x. There appears to be no prior work on designing C_1 directly to achieve both a desired stochastic function and a desired output SCC.

This paper presents a systematic way to directly improve the output correlation of C_1, eliminating the need for sequential correlation correction by S_{12}. We focus on the case where C_2 requires maximally correlated inputs, indicated by $SCC = +1$. The $SCC = +1$ case is unique because it is the only SCC value that can always be achieved between any number of SNs, no matter their length N or their values P_X. This property allows circuit optimizations not possible at other SCC values. While traditional SC circuits like an AND-gate multiplier require uncorrelated inputs ($SCC = 0$) [2], some recent SC research has utilized circuits with maximally correlated inputs ($SCC = +1$) to great advantage due to their RNS area savings and ability to implement new, useful functions such as saturating addition/subtraction, minimum, and maximum [10]. Large examples include SC image processing circuits [6][24], neural networks [8] [9], and digital filters [4].

We present a method called COMAX for reforming the input layer C_1 into one that outputs the desired function with the highest possible SCC. We refer to this type of CCP as the *correlation maximization problem* (CMP). COMAX does not rely on ad-hoc searching or simulation, rather it employs the theory of stochastic equivalence [15] to modify the target circuit.

The main contributions of this paper are:

1. Introduction and investigation of the combinational correlation maximization problem (CMP) in SC.
2. An algorithm, COMAX, for solving the CMP that, given a circuit C_1, finds the functionally equivalent design that best maximizes C_1's output SCC.
3. A case study applying COMAX to a representative SC-based image processing application, with area and performance comparisons between COMAX and existing sequential re-correlation designs.

II. Background

First, we review some basics of SC relating to data conversion and correlation.

A. Conversion Between SC and Binary

A binary number B_X is converted to a numerically equivalent SN X using a BSC, such as the one shown in Fig. 4a. This common type of BSC compares B_X with a pseudo-random value R produced by a random number source (RNS) to produce a (pseudo) random sequence of bits X. Commonly, the RNS is implemented with a linear feedback shift register (LFSR) [2]. Sharing one RNS among two or more binary comparators produces correlated output SNs and lowers the RNS area cost [16], whereas having a separate RNS for each BSC produces uncorrelated SNs but requires much greater area. The binary counter in Fig. 4b samples the number of 1s in an SN, effectively converting it back to binary. These two data conversion methods enable SC hardware to interface with traditional binary (base-2) components.

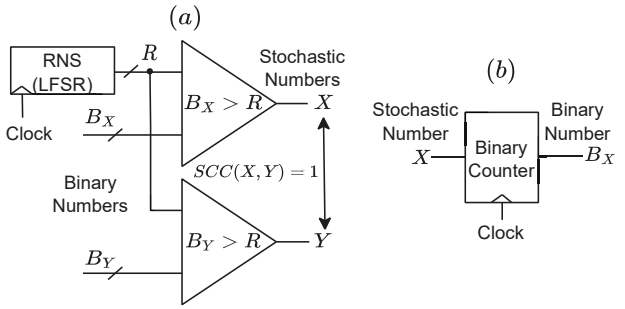

Fig. 4. (a): Binary-to-stochastic converter (BSC) employing a shared random number source (RNS) to produce two correlated SNs X and Y. (b): Binary counter acting as an SBC, the inverse of a BSC.

B. Correlation in SC

Next, to motivate COMAX, we review existing theory on measuring correlation in SC and quantifying its impact on stochastic circuit behavior. Intuitively, the correlation between two SNs with values P_X and P_Y relates to the probability of encountering a 1 in both bitstreams at the same time, denoted $P_{X\wedge Y}$. Table I shows three ways of aligning the 1s and 0s in two example SNs. Each distinct alignment differs only in the value of $P_{X\wedge Y}$. An AND gate, such as that in Fig. 2, performs multiplication without correlation error when $P_{X\wedge Y} = P_X P_Y$, in which case the covariance $\Delta = P_{X\wedge Y} - P_X P_Y$ is zero.

TABLE I
SELECTED SCC VALUES FOR $P_X = 6/10$ AND $P_Y = 5/10$.

X	Y	$SCC(X,Y)$	$P_{X\wedge Y}$
1111110000	1111100000	1	5/10
1011110100	1010000111	0	3/10
1111110000	0000011111	−1	1/10

The dominant correlation measure in the SC literature is the stochastic cross correlation (*SCC*) [10], which takes on values in the interval $[-1,+1]$, where -1 indicates maximum anti-correlation, 0 indicates no correlation, and $+1$ indicates maximum correlation. It is convenient here to slightly restate SCC in terms of covariance. *SCC* compares the covariance Δ of the two bitstreams with the minimum and maximum covariance values possible, Δ_{max} and Δ_{min}, where:

$$\Delta_{max} = min(P_X, P_Y) - P_X P_Y \tag{3}$$

$$\Delta_{min} = max(P_X + P_Y - 1, 0) - P_X P_Y \tag{4}$$

The definition of SCC is then given by Eq. (5).

$$SCC(X,Y) = \begin{cases} \Delta/\Delta_{max} & if\ \Delta > 0 \\ -\Delta/\Delta_{min} & otherwise \end{cases} \tag{5}$$

In general, non-integer SCC values are avoided because such values are less commonly achievable in practice (a fact we explore more later), so few applications for circuits that use these values are known. Instead, non-integer SCC values nearly always represent a deviation from a target integer value like $SCC = +1$, and are therefore a major source of error. To see this, suppose c is the actual measured input SCC value for a two-input circuit. Let F_0, F_{+1}, and F_{-1} be the ideal stochastic functions implemented by the circuit at $c = 0$, $c = +1$, and $c = -1$. Then F_c is given as a piece-wise linear combination [10]:

$$F_c = \begin{cases} (1+c)F_0 - cF_{-1} & if\ c < 0 \\ (1-c)F_0 + cF_{+1} & otherwise \end{cases} \tag{6}$$

Eq. (6) implies that the correlation error for a circuit requiring $SCC = +1$ can be computed directly as a function of c via $\epsilon(1,c) = |F_{+1} - F_c|$. It also follows from [10] that $\epsilon(1,-1) \geq \epsilon(1,0) \geq \epsilon(1,1)$ and consequently that $\epsilon(1,c)$ is a non-increasing function of c. Thus, increasing the input SCC will always reduce correlation error (or keep it the same) for designs requiring $SCC = +1$, even when the new SCC is still less than 1.

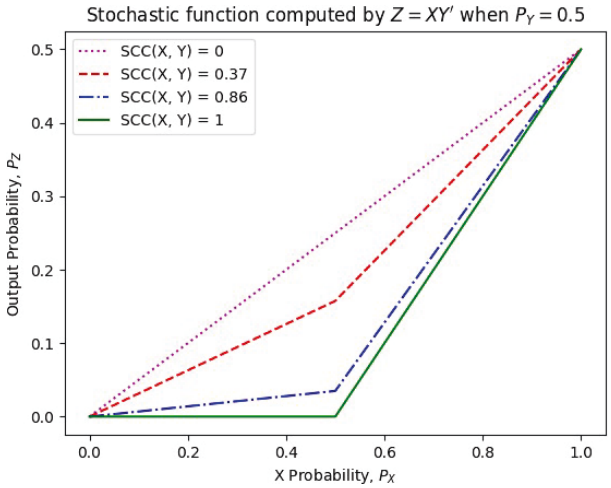

Fig. 5. Stochastic function $Z = XY'$ when $P_Y = 0.5$ under various SCC values.

For example, consider Fig. 5, which shows the stochastic function computed by the AND-gate circuit from Fig. 2. The green (solid) trace shows how the circuit operates as a saturating subtractor when $SCC = +1$, whereas the straight purple (dotted) trace shows that it operates as a multiplier when $SCC = 0$. As the SCC value approaches $SCC = +1$, the corresponding curve also more closely approximates the $SCC = +1$ curve, indicating less correlation error (bias). Most existing sequential re-correlation designs, as well as the COMAX method proposed in this paper do not guarantee that the output has $SCC = +1$ exactly. Instead, they try to approximate $SCC = +1$ as closely as possible to achieve sufficiently low correlation error.

III. Proposed Method

A. Correlation Maximization Problem (CMP)

First, the correlation maximization problem (CMP) is formally defined and justified. This paper focuses on maximizing the output SCC towards $SCC = +1$ because $SCC = +1$ is used in most of the known non-zero SCC applications [4][6][8-10][24], and has seen growing research interest recently. Furthermore, unlike $SCC = 0$ or other SCC values, pairwise $SCC = +1$ is always achievable between any number of circuit outputs, regardless of the SNs' lengths and values. For example, Fig. 6 presents a few 3D scatterplots showing the P_X, P_Y, P_Z values at which three SNs of length $N = 16$ can all be exactly correlated with the given SCC value. Fig. 6a demonstrates how $SCC = +1$ is always possible for all P_X, P_Y, P_Z, however the plots are sparser for the other SCC values like $SCC = 0$ (Fig. 6b), as these are not always achievable. In general, only the integer SCC values are possible often enough to base SC computation on, as non-integer values like $SCC = 0.5$ are very rarely possible, as shown in Fig. 6d.

To see why bitstreams can always be made to have $SCC = +1$, consider the following three SNs: X_1 = 10100110, X_2 = 11101101, and X_3 = 01000100. Now suppose that the 1s in these bitstreams are aligned on the left-hand side such they all come first, followed by all the 0s thus:

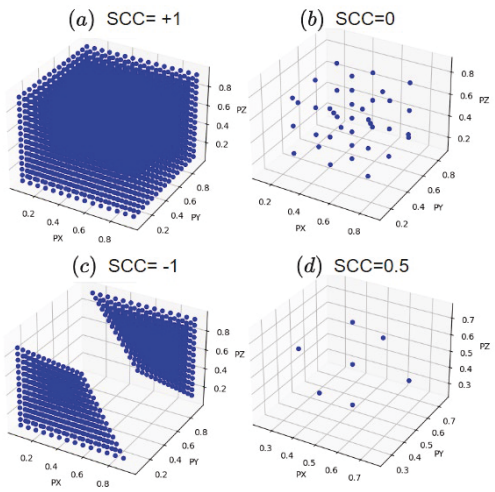

Fig. 6. 3D scatterplots showing the values of P_X, P_Y and P_Z at which three $N = 16$ bit SNs can be correlated with each other at $SCC(X,Y) = SCC(Y,Z) = SCC(X,Z)$.

$$X_1 = 11110000$$
$$X_2 = 11111100 \quad (7)$$
$$X_3 = 11000000$$

It can readily be verified with Eq. (5) that the pairwise SCC is 1 between all pairs of bitstreams in Eq. (7). It can be easily seen that left alignment like this always induces $SCC = +1$. Since +1 is the only SCC value that can always be achieved, the CMP is the only directly solvable type of CCP.

To formalize the CMP, let $\boldsymbol{Z} = f(\boldsymbol{X})$ be a Boolean function defining the logic (non-stochastic) behavior of a stochastic circuit C_1 that takes a random vector $\boldsymbol{X}$ of n SNs as input, and outputs a random vector $\boldsymbol{Z}$ of k SNs. A Boolean function $\boldsymbol{Z}^* = f^*(\boldsymbol{X})$ is said to be *stochastically equivalent* (SE) to f if the two circuits compute the same stochastic function. In other words, $P_{Z_1} = P_{Z_1^*}, P_{Z_2} = P_{Z_2^*}, \ldots, P_{Z_k} = P_{Z_k^*}$ for all $\boldsymbol{X}$.

The set of all such f^* functions constitutes a *stochastic equivalence class* (SEC) [15]. This type of equivalence only guarantees that the marginal output probabilities match; Generally, $\boldsymbol{Z} \neq \boldsymbol{Z}^*$ because the correlations between the random variables in $\boldsymbol{Z}$ may be different than those between the ones in $\boldsymbol{Z}^*$. An example is the stochastic scaled addition function $P_Z = \frac{1}{2}(P_{X_1} + P_{X_2})$, which can be implemented using either a MUX gate or a SE majority (MAJ) gate [15], as shown in Figs. 7a and 7b, respectively.

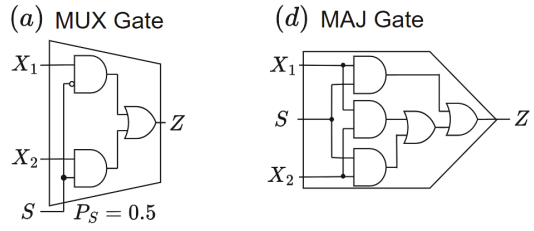

Fig. 7. Two SE circuits computing $P_Z = \frac{1}{2}(P_{X_1} + P_{X_2})$: (a) The multiplexer (MUX) gate. (b) The majority (MAJ) gate, which outputs 1 if any two inputs are 1.

Despite being SE, circuits employing MAJ-based adders have been shown to output higher SCC than MUX-based ones [17] [18]. For example, in [17] replacing MUX with MAJ gates within a stochastic image processing circuit raises the average output SCC from $SCC = 0.32$ to $SCC = 0.48$. Thus, using MAJ instead of MUX can reduce correlation error for subsequent C_2 layers requiring $SCC = +1$. The goal of the CMP is to generalize this observation to find a correlation-maximizing replacement for any combinational stochastic circuit. This should be an SE Boolean function f^* that causes $\boldsymbol{Z}^*$ to have the highest possible expected output SCC among all pairs of outputs. This improves the output correlation while leaving the circuit's stochastic function unchanged.

Formally, the CMP is defined as finding

$$f_{opt} = \arg\max_{f^*}(SCC(f^*_{\ell_1}(\mathbf{X}), f^*_{\ell_2}(\mathbf{X}))) \tag{8}$$

where $(\ell_1, \ell_2) \in [1..k]^2$ such that Eq. (8) requires the SCC to be as close to 1 as possible between all pairs of outputs. Any Boolean function that solves the CMP maximizes the output correlation(s) of f^* under any distribution of the inputs $\boldsymbol{X}$. It's important to note that solving the CMP does not necessarily guarantee $SCC = +1$, but it does ensure f_{opt} has the highest output *SCC* among all (combinational) SE designs implementing the same stochastic function.

B. COMAX: Solving the CMP

Next, an algorithm COMAX for directly solving the CMP is proposed. A pseudo-code description of COMAX appears in Alg. 1 and this section explains the theory behind it. The goal of COMAX is to manipulate a Boolean function f to yield a function f_{opt} that is stochastically equivalent to f and achieves the maximum possible correlation between all pairs of outputs. f is defined by a $2^n \times k$ Boolean matrix or truth table comprising k vectors $f = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k]$, where $\mathbf{f}_\ell$ represents the truth-table for the ℓth output. These are supplied as input to Alg. 1, whereas the outputs are a new truth-table representing the SE function $f_{opt} = [\mathbf{f}^*_1, \mathbf{f}^*_2, \ldots, \mathbf{f}^*_k]$.

To understand the operation of COMAX, first assume that the n input bitstreams $\boldsymbol{X}$ are organized into a set of n_c constant SN inputs $\boldsymbol{X}_C$ and a set of n_v variable SN inputs $\boldsymbol{X}_V$, such that $\boldsymbol{X} = [\boldsymbol{X}_C, \boldsymbol{X}_V]^T$. The constant bitstreams in $\boldsymbol{X}_C$ are uncorrelated and have the same fixed probability value 0.5. Generally, any SC circuit can be modeled in this way [15][1]. For example, the MUX/MAJ circuits in Fig. 7 have a constant input of $S = 0.5$. This distinction is useful because the 0.5-valued constant inputs are all interchangeable; they can be swapped and/or inverted anywhere they are used in the circuit to produce a new Boolean function f^* that is SE to f but may have different output correlation behavior. If C_i is a layer of a multi-layer circuit, then inputs from prior layers C_{i-1} are treated as variable inputs, while additional inputs can be either variable or constant.

Alg. 1: COMAX to solve the CMP given in Eq. (8)

Input: $f = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k]$, n_c, n_v
Output: $f_{opt} = [\mathbf{f}^*_1, \mathbf{f}^*_2, \ldots, \mathbf{f}^*_k]$

```
1 for ℓ = 1 to k do
2     Let F_ℓ = 0, F_ℓ* = 0 be 2^{n_v} × 2^{n_c} binary matrices
3     F_ℓ = reshape(f_ℓ, 2^{n_v} × 2^{n_c})
      // Reshape f_ℓ into F_ℓ via column-major ordering
4     for i = 1 to n_v do
5         Let w = sum(F_{ℓ_i})
          // Sum of ith row
6         for j = 1 to w do
7             F*_{ℓ_ij} = 1
              // Set first w values of ith row to 1
8     f*_ℓ = reshape(F*_ℓ, 2^n × 1)
      // Reshape F_ℓ* into f*_ℓ via column-major ordering
9 return f_opt = [f*_1, f*_2, ..., f*_k]
```

The first step of COMAX is to reshape each of the k truth-table vectors $\mathbf{f}_\ell$ from a $2^n \times 1$ vector into a matrix $\mathbf{F}_\ell$ with shape $2^{n_v} \times 2^{n_c}$ via column-major ordering, noting that $n = n_c + n_v$. This is done in lines 2 and 3 of the pseudo-code. The reshaping process is illustrated for a truth-table vector representing a MUX-based stochastic adder in Eq. (9). The circuit has two variable inputs X and Y (data) and one constant input S (select), so $\mathbf{F}_M$ has shape 4 × 2:

$$\mathbf{f}_M = \begin{matrix} SXY \\ 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{matrix} \begin{bmatrix} Z \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \Longrightarrow \mathbf{F}_M = \begin{matrix} XY \\ 00 \\ 01 \\ 10 \\ 11 \end{matrix} \begin{bmatrix} S=0 & S=1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{9}$$

This reshaping is done because all entries in $\boldsymbol{X}_C$ have the value 0.5, so every column of $\mathbf{F}_\ell$ has the same probability of being sampled, namely, $1/2^{n_c}$. It therefore follows that re-ordering the 1s and 0s in one or more rows of $\mathbf{F}_\ell$ results in a new matrix $\mathbf{F}^*_\ell$ representing a Boolean function f^* that is SE to f, since the relative probability of sampling a 1 versus a 0 for a given assignment of variable input values $\boldsymbol{X}_V$ does not change. For this reason, $\mathbf{F}_\ell$ and $\mathbf{F}^*_\ell$ are called *stochastically equivalent matrices* (SEMs).

Formally, SEMs are defined such that if f and f^* are SE, then Eq. (10) holds true for all their SEM rows $i \in [1..2^{n_v}]$ and outputs $\ell \in [1..k]$:

[1] As discussed in [15], this assumption is general because constants other than 0.5 can be derived from multiple uncorrelated 0.5 sources using relatively simple combinational circuits. By design, nearly all SC random sources used in the SC literature, such as the LFSR, produce uncorrelated 0.5-valued SNs to a high degree of approximation.

$$\sum_{j=1}^{2^{n_c}} (\mathbf{F}_\ell)_{ij} = \sum_{j=1}^{2^{n_c}} (\mathbf{F}^*_\ell)_{ij} = \mathbf{W}_{i\ell} \tag{10}$$

Here $\mathbf{W}$ is a *weight matrix* that corresponds to a unique stochastic function. For example, the weight matrix for both the MUX and MAJ gates is $\mathbf{W}_M = [0\ 1\ 1\ 2]^T$. It corresponds to the stochastic addition function $P_Z = \frac{1}{2}(P_{X_1} + P_{X_2})$.

COMAX uses SEMs to optimize the circuit's correlation without changing the stochastic function. The next step after reshaping the input truth-table vector $\mathbf{f}_\ell$ into the SEM $\mathbf{F}_\ell$ is to compute the weight of each row according to Eq. (10), as shown in line 5 of Alg. 1. This yields a $w = \mathbf{W}_{i\ell}$. The following lines, 6 and 7, construct the new, optimized SEM by placing the w required 1s such that they are aligned to the left side of $\mathbf{F}^*_\ell$. This step is illustrated with an example in Fig. 8, where Figs. 8c and 8d are the SEMs before and after optimization, respectively. It can be shown that left-aligning the 1s in this way maximizes the number of possible locations where a 1 in $\mathbf{F}^*_1$ overlaps with a 1 at the same location in $\mathbf{F}^*_2$. In other words, the new matrices have the highest possible number of locations where $\mathbf{F}^*_1 \wedge \mathbf{F}^*_2 = 1$. Since $\mathbf{F}^*_1$ and $\mathbf{F}^*_2$ are SEMs, this maximizes $P_{Z_1 \wedge Z_2}$ without changing P_{Z_1} or P_{Z_2} individually, therefore maximizing *SCC*. Based on these insights, the solution to the CMP is given in the following theorem (a proof is in Appendix A).

Theorem 1: Let $\mathbf{F}^*_1, \ldots, \mathbf{F}^*_k$ be the SEMs for a k-output Boolean function f^*. Then f^* is a solution to the CMP if Eq. (11) holds true for all output pairs $(\ell_1, \ell_2) \in [1..k]^2$ and SEM rows $i \in [1 \ldots 2^{n_v}]$.

$$SCC\left((\mathbf{F}^*_{\ell_1})_i, (\mathbf{F}^*_{\ell_2})_i\right) = 1 \tag{11}$$

In other words, each row i of the SEMs should be reordered so that the SCC measured between the row vectors is 1. Crucially, it is always possible to satisfy Eq. (11) for any set of SEMs via left-alignment, as done in Eq. (7), so this is the solution used by COMAX.

After generating the optimized SEMs $\mathbf{F}^*_\ell$, the final step of COMAX is to reshape these SEMs back into truth-table vectors $\mathbf{f}^*_\ell$. This is done in line 8 of Alg. 1 and is the reverse process of line 3. An example is given in Eq. (12), which continues the MUX example from Eq. (9), showing how the left-alignment and final reshaping steps of COMAX are applied.

$$\mathbf{F}^*_M = \begin{array}{c} XY \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} S=0 \quad S=1 \\ \left[\begin{array}{cc} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{array}\right] \end{array} \Longrightarrow \mathbf{f}^*_M = \begin{array}{c} SXY \quad Z \\ \left[\begin{array}{cc} 000 & 0 \\ 001 & 1 \\ 010 & 1 \\ 011 & 1 \\ 100 & 0 \\ 101 & 0 \\ 110 & 0 \\ 111 & 1 \end{array}\right] \end{array} \tag{12}$$

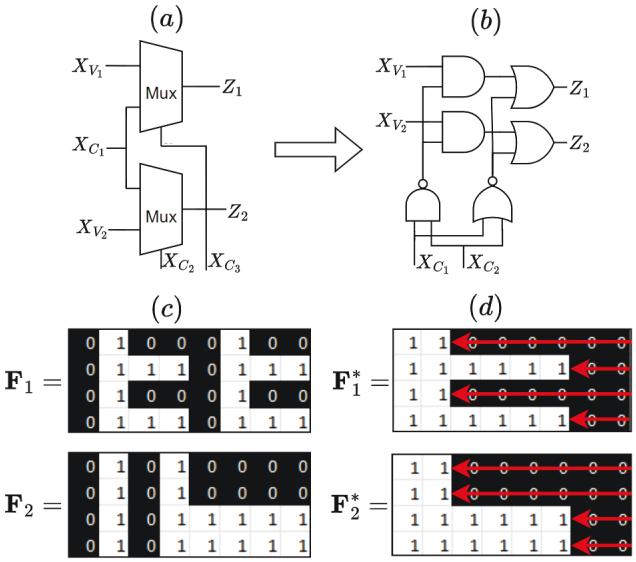

Fig. 8. Example of COMAX using SEMs. (a) Original circuit, computing $P_{Z_1} = 0.5P_{X_{V_1}} + 0.25$ and $P_{Z_2} = 0.5P_{X_{V_2}} + 0.25$, when $P_{X_{C_1}} = P_{X_{C_2}} = P_{X_{C_3}} = 0.5$. (b) SE AND-gate circuit solving the CMP. (c) Original SEMs. (d) Optimized SEMs, with 1s aligned on the left so that $SCC\left((\mathbf{F}^*_{\ell_1})_{i'}, (\mathbf{F}^*_{\ell_2})_i\right) = 1$.

The Boolean sum-of-minterms expression for Eq. (12) is

$$f_{M_{opt}} = S'X'Y + S'XY' + S'XY + SXY \tag{13}$$

Eq. (13) can be simplified to $f_{M_{opt}} = S'X + S'Y + XY$. This result (which is generalized in Appendix B) defines a MAJ gate with an inverted select input. Consequently, this example application of COMAX proves that the MAJ gate implementation of stochastic addition solves the CMP. It's important to observe that COMAX can yield optimizations beyond simple MUX-MAJ substitution. For example, in Fig. 8 it produces a circuit implementation with one less constant input, and an entirely different gate structure.

Overall, the runtime complexity of the entire COMAX algorithm is $O(k2^n)$, which is consistent with the fact that it is a truth-table based method. Here, we remark that SC designs, which are inherently bit-serial, usually have a small number of input lines. This makes truth-table and other exponential-time algorithms much more viable in the SC context than for bit-parallel paradigms like binary computing. For example, probability transfer matrices (PTMs) are a well-established method of representing stochastic circuits using matrices of shape $2^n \times 2^k$ [22]. In [17], the relationship between PTMs and truth-tables is explored.

IV. Image Processing Case Study

Next, as a case study we evaluate the effectiveness of COMAX at reducing correlation error for a two-layer image processing pipeline consisting of a Gaussian blur (GB) operation followed by edge detection. This task is similar to those in [12], [19], and [23], all of which are used for evaluating SC re-correlation performance, thus it is useful for evaluating COMAX. The GB operation consists of a 3x3 matrix which acts

as a sliding window filter that averages local pixels to produce a blurring effect. The filter computes an operation of the form:

$$Z_{11} = \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix}^T \begin{bmatrix} X_{V_{11}} & X_{V_{12}} & X_{V_{13}} \\ X_{V_{21}} & X_{V_{22}} & X_{V_{23}} \\ X_{V_{31}} & X_{V_{32}} & X_{V_{33}} \end{bmatrix} \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix} \tag{14}$$

The stochastic implementation of Eq. (14) employs a tree of MUXes, as shown in Fig. 9b. For edge detection, a 2x2 Roberts cross edge detector (RCED) implementing Eq. (15) is used:

$$Z = \frac{1}{2}|Z_{11} - Z_{22}| + \frac{1}{2}|Z_{12} + Z_{21}| \tag{15}$$

Both Eqs. (14) and (15) are examples of non-trivial SC functions. RCEDs have an area-efficient SC implementation relying on $SCC = +1$ [6] and consisting of two XOR gates and one MUX, as depicted in Fig. 9d. Each of the four RCED inputs Z_{11}, Z_{22}, Z_{12}, Z_{21} is generated by a separate instance of the 3x3 GB circuit, such that the overall input image tile is 4x4 pixels, as shown in Fig. 9a. The full circuit receives 16 variable inputs (pixel intensities) from the image tile, 4 constant inputs shared among the GB kernels, and 1 constant input for the RCED. Henceforth, this combined Gaussian-blur and edge detection architecture is referred to as GBED.

To measure COMAX's impact on the edge detection performance of the GBED circuit, the algorithm is applied to the GB layer and the entire circuit is simulated on a dataset of ten grey-scale test images (0 to 1 range) from the MATLAB image processing toolbox. Since GB is intended to filter out Gaussian noise, each test image is made noisy by adding Gaussian noise with $\mu = 0$ and $\sigma = 0.1$. The images are broken up into many 4x4 pixel windows, which are then processed by the GBED circuit by converting the pixel intensities into SN bitstreams of length $N = 256$ or $N = 16$ using simulated LFSRs.

First, the SCC at the output of the GB layer is calculated over all possible 4x4 pixel windows in the dataset. For SCC calculation, a bitstream length of $N = 256$ is used. The frequency histogram in Fig. 10 shows the resulting distribution of SCC values both before and after applying COMAX (higher SCC is better). It directly demonstrates the SCC benefits of COMAX, showing that the output SCC is greater than 0.92 about 45% more often, and there are far fewer instances of SCC < 0.75. The unmodified 3x3 GB circuit only outputs $SCC = 0.72$ on average, while the GB circuit optimized by COMAX outputs $SCC = 0.97$. To visualize the effect this improvement in SCC has on the circuit's edge detection performance, Fig. 11 shows the GBED results on one of the noisy test images before and after applying COMAX, with Figs. 11c and 11f showing the case where $N = 256$. Observe that COMAX results in much improved edge detection, with far fewer erroneous edges being produced. This result is visually similar to that produced by a 32-bit floating-point GBED implementation (Fig. 11d).

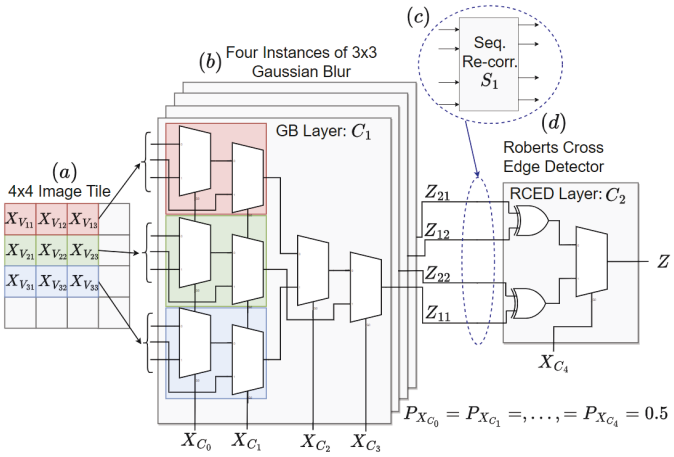

Fig. 9. GBED circuit. (a) 4x4 input pixel tile. (b) Four instances of the 3x3 Gaussian blur circuit. (c) Optional recorrelator to increase correlation (d) 2x2 Roberts cross edge detection circuit, requiring correlated inputs.

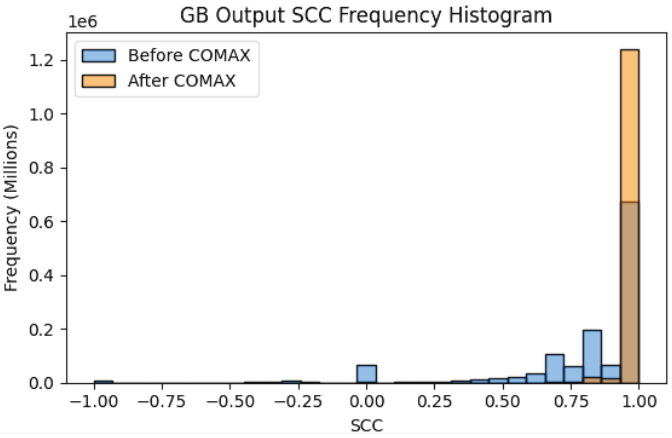

Fig. 10: Frequency histogram of SCC values from GB for all possible 4x4 pixel windows in the dataset. It shows that COMAX substantially increases SCC.

To contextualize COMAX with existing correlation correction techniques, we compare our design against three sequential techniques that have been used to achieve re-correlation: full bitstream re-generation [11], the synchronizer circuit from [12] with a save depth of $D = 1$, and the correlator circuit from [13]. The synchronizer works to increase SCC between two SNs X and Y by remembering up to D unpaired 1s, such as $XY = 10$, and attempting to pair them with unpaired 1s on the other bitstream, such as $XY = 01$, leading to $XY = 11$, thus increasing $P_{X \wedge Y}$ and SCC. In contrast, the correlator uses a counter to dynamically estimate which SN has the min/max value, then relocates unpaired 1s on the lower-valued bitstream to align with those on the higher-valued one. Each re-correlation design is inserted into the circuit as shown in Fig. 9c. For completeness, we also include cases where both COMAX and sequential re-correlation are used together.

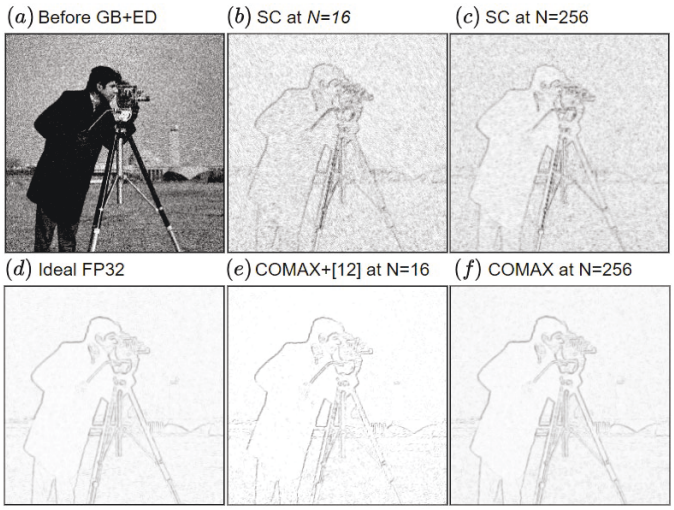

Fig. 11. (a) Original test image with added Gaussian noise. (b) Un-optimized GBED with N = 16. (c) Un-optimized GBED with N = 256. (d) Ideal 32-bit software-based GBED (e) COMAX with N = 16 in combination with the design from [12] with D = 1. (f) COMAX alone with N = 256.

For example, Fig. 11e shows the result of combining [12] with COMAX, which achieves edge detection of comparable quality to floating-point even with short bitstreams of length $N = 16$.

Next, the area cost of COMAX and existing methods is evaluated using Synopsys Design Compiler with the FreePDK45 45nm cell library [20]. Each design is implemented in SystemVerilog and synthesized with a 500MHz clock and Design Compiler's default optimization parameters. Table II summarizes these area results (lower is better).

TABLE II
GBED AREA RESULTS

Method of Correlation Control	Area (μm^2)
None (GBED only)	253
COMAX	**224**
[11] Full bitstream regeneration	664
[12] Synchronizer	459
[12] Synchronizer and COMAX	**433**
[13] Correlator	575
[13] Correlator and COMAX	**540**

Table II indicates that applying COMAX to the GBED circuit does not increase its area footprint; in fact, the area decreases slightly. These results imply that COMAX can be applied to the GBED circuit to improve its edge detection performance without sacrificing any area to do so. The area cost of [12] is 2x higher than COMAX, and for [13] it is 2.5x higher.

Lastly, output image quality is measured using the mean structural similarity index measure (MSSIM) [21]. MSSIM can quantify the human-perceptible differences between images more effectively than traditional measures such as mean squared error (MSE) or the related PSNR [21]. Fig. 12 plots the MSSIM before and after applying COMAX to each design (higher is better).

Observe from the data in Fig. 12 that applying COMAX to any of the circuit cases strictly increases MSSIM, except the very costly full regeneration case which already outputs the maximum $SCC = +1$. For example, when N = 256, applying COMAX only to the original circuit improves the MSSIM from

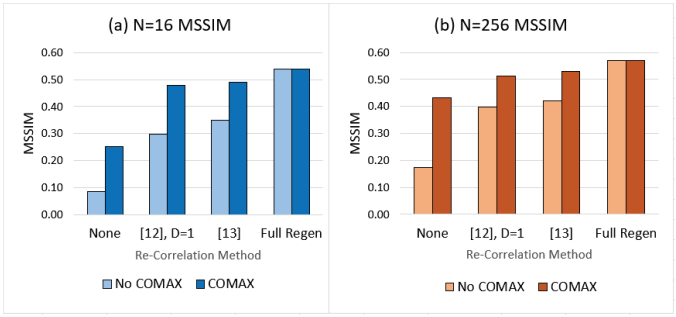

Fig. 12. MSSIM image quality results sorted by MSSIM performance for (a) bitstream length N = 16, and (b) N = 256. Light and dark-colored bars represent MSSIM before and after applying COMAX, respectively; higher is better.

0.17 to 0.43, a 2.5x increase. The best existing method that achieves a MSSIM close to this is the synchronizer from [12], but according to Table II this comes at the additional cost of 2x higher area, demonstrating COMAX's big area-cost advantage. COMAX also excels when used in combination with synchronizers. This configuration works especially well when $N = 16$, as it produces a MSSIM of 0.48, which is better than that produced when synchronizers are used on their own even at $N = 256$, despite the bitstream length being 16x shorter. The general trend of Fig. 12 indicates that COMAX achieves higher relative MSSIM gains when the initial MSSIM is low.

V. CONCLUSION

In this work, we formally defined the problem of combinational correlation maximization (CMP) in multi-layer stochastic circuit design and presented a novel method, COMAX, to solve it. This problem is central to developing efficient multi-layer stochastic circuits C_1C_2 for use in the common practical SC design setting where C_2 requires an input SCC of +1. Unlike existing re-correlation techniques, which rely on expensive sequential hardware, COMAX directly produces a combinational SC circuit that achieves the highest possible output correlation without any ad-hoc design space searching or simulation. In an image-processing case study consisting of a Gaussian blur filter followed by a correlation-dependent edge detector, we demonstrated that COMAX increases average output SCC from 0.72 to 0.97 and output image quality, measured by MSSIM, by a factor of 2.5x, at no additional area cost. Achieving this same performance gain via conventional re-correlation circuits requires twice the area footprint of the COMAX implementation.

APPENDIX A
PROOF OF THEOREM 1

Suppose we are given the k Boolean functions for a k-output stochastic circuit, where the ℓth output is $Z_\ell = f_\ell(\boldsymbol{X})$. The goal is to find the SE functions $f^*_{\ell_1}(\boldsymbol{X})$ and $f^*_{\ell_2}(\boldsymbol{X})$ that make $SCC\left(Z^*_{\ell_1}, Z^*_{\ell_2}\right)$ to as close to 1 as possible for all pairs of outputs $(\ell_1, \ell_2) \in [1..k]^2$ and distributions of $\boldsymbol{X}$, thus solving the CMP. For notational brevity, let $P_{Z_{\ell_1}} = P_A$, $P_{Z_{\ell_2}} = P_B$, and $P_{Z_{\ell_1} \wedge Z_{\ell_2}} = P_{A \wedge B}$. Then, using the definition of SCC given in Eq. (5) we obtain:

$$SCC(Z_{\ell_1}, Z_{\ell_2}) = \begin{cases} \frac{P_{A\wedge B} - P_A P_B}{min(P_A, P_B) - P_A P_B} & if\ P_{A\wedge B} > P_A P_B \\ \frac{P_{A\wedge B} - P_A P_B}{P_A P_B - max(P_A + P_B - 1, 0)} & otherwise \end{cases} \quad (16)$$

Recall that the Boolean function $Z_\ell^* = f_\ell^*(\boldsymbol{X})$ is SE to $Z_\ell = f_\ell(\boldsymbol{X})$ if $P_{Z_\ell^*} = P_{Z_\ell}$. To define $SCC\left(Z_{\ell_1}^*, Z_{\ell_2}^*\right)$, the only term from Eq. (16) that changes is the output overlap probability, which goes from $P_{Z_{\ell_1} \wedge Z_{\ell_2}}$ to $P_{Z_{\ell_1}^* \wedge Z_{\ell_2}^*}$; the others remain the same because of stochastic equivalence. Observe that in both piecewise cases, increasing $P_{Z_{\ell_1}^* \wedge Z_{\ell_2}^*}$ linearly increases SCC. Therefore, the optimization problem can be restated as:

$$f_{opt} = \arg\max_{f^*}(P_{Z_{\ell_1}^* \wedge Z_{\ell_2}^*}) \quad (17)$$

The output overlap probability of a stochastic circuit is related to its truth-table vectors and the input distribution of $\boldsymbol{X}$ via

$$P_{Z_{\ell_1}^* \wedge Z_{\ell_2}^*} = (\mathbf{f}_{\ell_1}^* \wedge \mathbf{f}_{\ell_2}^*)^T \mathbf{v}_{\mathbf{X}} \quad (18)$$

where $\mathbf{f}_{\ell_1}^* \wedge \mathbf{f}_{\ell_2}^*$ is the element-wise logical AND between truth-table vectors $\mathbf{f}_{\ell_1}^*$ and $\mathbf{f}_{\ell_2}^*$, and $\mathbf{v}_{\boldsymbol{X}}$ is a probability transfer vector (PTV) [17] that defines the distribution of $\boldsymbol{X}$. If $\mathbf{v}_{\boldsymbol{X}}$ is separated into its independent variable and constant components: $\mathbf{v}_{\boldsymbol{X}} = \mathbf{v}_{\boldsymbol{X_V}} \otimes \mathbf{v}_{\boldsymbol{X_C}}$, then Eq. (18) can be rewritten using SEMs as:

$$P_{Z_{\ell_1}^* \wedge Z_{\ell_2}^*} = \mathbf{v}_{\mathbf{X}_V}^T (\mathbf{F}_{\ell_1}^* \wedge \mathbf{F}_{\ell_2}^*) \mathbf{v}_{\mathbf{X}_C} \quad (19)$$

Now note that $\mathbf{v}_{\boldsymbol{X_C}} = [2^{-n_c}, 2^{-n_c}, \ldots]^T$ because it describes a distribution of n_c independent bitstreams with probability 0.5. Therefore:

$$P_{Z_{\ell_1}^* \wedge Z_{\ell_2}^*} = 2^{-n_c} \sum_{i=1}^{2^{n_v}} (\mathbf{v}_{\mathbf{X}_V})_i \sum_{j=1}^{2^{n_c}} (\mathbf{F}_{\ell_1}^* \wedge \mathbf{F}_{\ell_2}^*)_{ij} \quad (20)$$

where i is the row index corresponding to value assignments of the variable inputs, and j is the column index corresponding to value assignments of the constant inputs. Since f^* must be optimal for all possible $\mathbf{v}_{\boldsymbol{X_V}}$, the optimization problem can be rewritten as:

$$f_{opt} = \arg\max_{f^*}(\sum_{j=1}^{2^{n_c}} (\mathbf{F}_{\ell_1}^* \wedge \mathbf{F}_{\ell_2}^*)_{ij})\ \forall_i \quad (21)$$

Eq. (21) is maximized if, for each row i, the 1s between the ℓ_1th and ℓ_2th SEM have the highest possible number of overlaps. This is exactly equivalent to the condition of having maximum SCC between these SEM rows: $SCC\left((\mathbf{F}_{\ell_1}^*)_i, (\mathbf{F}_{\ell_2}^*)_i\right) = 1$. This concludes the proof.

APPENDIX B
SUM-OF-PRODUCTS FORM FOR COMAX

The result returned by COMAX can be expressed rather elegantly as a Boolean sum-of-products (SOP) involving the weight matrix $\mathbf{W}$. For the ℓth output, this is:

$$f_{opt_\ell} = \bigvee_{i=1}^{2^{n_v}} \bigvee_{j=1}^{\mathbf{W}_{i\ell}} m_{v_{i-1}} m_{c_{j-1}} \quad (22)$$

where m_{v_i} and m_{c_j} are the ith and jth minterms for the sets of variable and constant inputs, respectively. For instance, $m_{v_0} = X'_{V_1} X'_{V_2} X'_{V_3}$, $m_{c_5} = X'_{C_1} X_{C_2} X'_{C_3}$, etc.

Eq. (22) is an alternative way of representing COMAX and provides insight into the gate-level implementation of circuits optimized with COMAX. Following the previous MUX example, Eq. (22) yields the following sum-of-products expression for a stochastic scaled-adder that solves the CMP:

$$\begin{aligned} f_{M_{opt}} &= m_{v_1} m_{c_0} + m_{v_2} m_{c_0} + m_{v_3} m_{c_0} + m_{v_3} m_{c_1} \\ &= XY'S' + X'YS' + XYS' + XYS \end{aligned} \quad (23)$$

The first two terms of Eq. (23) come from the two 1's in the MUX weight matrix $\mathbf{W}_M = [0\ 1\ 1\ 2]^T$, where the minterms $m_{v_1} = XY'$ and $m_{v_2} = X'Y$ for the variable inputs are each weighted by 1 and therefore each share the minterm, $m_{c_0} = S'$. Conversely, the last two terms of Eq. (23) weight the minterm $m_{v_3} = XY$ by 2, so the first utilizes $m_{c_0} = S'$ while the second utilizes $m_{c_1} = S$. Observe that the result from Eq. (23) is equal to Eq. (13), derived previously using Alg. 1.

ACKNOWLEDGMENT

This research was supported by the U.S. National Science Foundation under grant no. CCF-2006704.

REFERENCES

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

[2] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, May 2013.

[3] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, and T. Inoue, "Compact and accurate digital filters based on stochastic computing," *IEEE Trans. on Emerging Topics in Computing*, vol. 7, pp. 31–43, Jan 2019.

[4] T. J. Baker and J. P. Hayes, "CeMux: Maximizing the accuracy of stochastic mux adders and an application to filter design," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 27, no. 3, Jan 2022.

[5] P. Li, D. J. Lilja, W. Qian, K. Bazargan and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Trans. on VLSI Systems*, vol. 22, pp. 449–462, 2014.

[6] A. Alaghi, C. Liand J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *50th Design Automation Conf. (DAC),* Article 136, 2013.

[7] Y. Liu, S. Liu, Y. Wang, F. Lombardi and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, pp. 2809–2824, 2021.

[8] H. Abdellatef, M. Khalil-Hani, N. Shaikh-Husin and S. O. Ayat, "Stochastic computing correlation utilization in convolutional neural network basic functions," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 16, pp. 2835–2843, 2018.

[9] C. F. Frasser, P. Linares-Serrano, A. Moran, J. Font-Rossello, V. Canals, M. Roca, T. Serrano-Gotarredona and J. L. Rossello, “Exploiting correlation in stochastic computing based deep neural networks,” in *2021 Conf. on Design of Circuits and Integrated Systems (DCIS)*, 2021.

[10] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 39-46, 2013.

[11] P. -S. Ting and J. P. Hayes, "Isolation-based decorrelation of stochastic circuits," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pp. 88-95, 2016.

[12] V. T. Lee, A. Alaghi and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1417-1422, 2018.

[13] H. Abdellatef, A. Abdellatif and N. Ramaha, “A new correlation manipulation circuit for stochastic computing,” in *2022 International Conference on Smart Systems and Power Management (IC2SPM)*, pp. 182–186, 2022.

[14] S. Asadi, M. H. Najafi, and M. Imani, “Corld: In-stream correlation manipulation for low-discrepancy stochastic computing,” in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

[15] T.-H. Chen and J. P. Hayes, “Equivalence among stochastic logic circuits and its application to synthesis,” in *IEEE Trans. on Emerging Topics in Computing*, vol. 7, pp. 67–79, 2019.

[16] M. Yang, B. Li, D. J. Lilja, B. Yuan and W. Qian, “Towards theoretical cost limit of stochastic number generators for stochastic computing,” in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 154–159, 2018.

[17] O. Hoffend and J. P. Hayes, "Analyzing multilevel stochastic circuits using correlation matrices," in *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 130-135, 2022.

[18] T. Baker, O. Hoffend, and J. Hayes, “Multiplexer-majority chains: Managing correlation and cost in stochastic number generation,” in *Proceedings of the 17th ACM International Symposium on Nanoscale Architectures (NANOARCH),* 2022.

[19] H. Abdellatef, M. Khalil-Hani, and N. S. Husin, “Accurate and compact stochastic computations by exploiting correlation,” in *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 27, pp. 547–564, 12 2019.

[20] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, “Freepdk: An open-source variation-aware design kit,” in *2007 IEEE International Conference on Microelectronic Systems Education (MSE’07)*, pp. 173–174, 2007.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. on Image Processing*, vol. 13, pp. 600–612, 2004.

[22] S. Krishnaswamy et al., “Probabilistic transfer matrices in symbolic reliability analysis of logic circuits.” *ACM Trans. Design Automation of Elec. Sys.,* vol.13, issue 1, pp.1-35, 2008.

[23] Alaghi, A., Ting, P., Lee, V.T., Hayes, J.P. “Accuracy and Correlation in Stochastic Computing,” in *Gross, W., Gaudet, V. (eds) Stochastic Computing: Techniques and Applications*. Springer, 2019.

[24] K. Budhwani, R. Ragavan and O. Sentieys, "Taking advantage of correlation in stochastic computing," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4, 2017.