Bayesian Accuracy Analysis of Stochastic Circuits

Timothy J. Baker and John P. Hayes
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, 48109
{bakertim, jhayes}@umich.edu

ABSTRACT

Understanding accuracy and the tradeoffs it entails is key to evaluating the growing list of stochastic computing (SC) circuit designs. Due to shortcomings of current SC error theory, simulation has become the standard way to estimate a circuit's accuracy. However, simulation can demand large computational resources and lead to uncertain, misleading, or unexplainable results. A soundly based analytic approach is therefore preferable to simulation. In this work, we first show the input value distribution's large influence on circuit accuracy. Then we develop a Bayesian error analysis methodology which uses the input value distribution as a prior to inform better accuracy estimates. This error formulation introduces concepts new to SC such as estimator dominance and points to ways of improving simulation-based accuracy estimates. Orthogonal to the Bayesian ideas, we also show how to use bias-variance decomposition to simplify and aggregate the effects of SC's many error sources. We present techniques that use the beta distribution to model the stochastic number value distribution. Finally, we demonstrate the use of these ideas to improve the accuracy and analysis of an SC-based neural network.

KEYWORDS

Stochastic computing, Bayesian analysis, beta distribution, bias-variance decomposition, error analysis, neural networks

1 Introduction

Stochastic computing (SC) is a form of approximate computing that employs bit-streams known as stochastic numbers (SNs) to represent data [2][9]. The key parameter of an N-bit SN $\mathbf{X} = x_1x_2 \dots, x_N$ is p_x, the probability that an arbitrary bit in $\mathbf{X}$ is 1. $\mathbf{X}$'s value X is determined by p_x and by the chosen SN format. The basic (unipolar) format defines $X = p_x$ and allows for very simple arithmetic computing elements. Consider an AND gate with two uncorrelated N-bit inputs, $\mathbf{X}$ and $\mathbf{Y}$ and output $\mathbf{Z}$. The AND gate's functionality implies that $p_z = p_xp_y$ which makes $Z = XY$, so the AND gate serves as a SN multiplier. An important caveat is that Z is not available directly, but instead must be estimated by observing the AND gate's output bit-stream $\mathbf{Z} = z_1z_2 \dots z_N$. The usual and most straightforward estimator $\hat{Z}$ of Z is defined as the number of 1s in $\mathbf{Z}$ divided by $\mathbf{Z}$'s length, i.e., $\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} z_i$.

ICCAD '20, November 2–5, 2020, Virtual Event, USA © 2020 Association for computing Machinery. ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415703

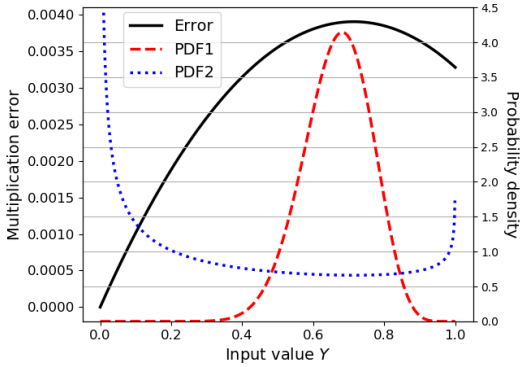

Figure 1: Simulated error $\hat{Z} - Z^*$ (solid black line) for an AND-gate multiplier with inputs $\mathbf{X}$, $\mathbf{Y}$ and SN length $N = 64$. X is set to 0.7 and Y is randomly sampled with the indicated PDFs (dashed lines). MSE = 0.00382 with PDF1 (red); MSE = 0.00228 with PDF2 (blue).

In general, *SN error* is associated with some difference between the estimator $\hat{Z}$ and Z^*, the exact or target output value. Errors arise from a variety sources including random fluctuations of the SN bits and unwanted correlations between them [1][6]. Most error sources can be mitigated by increasing SN length N or by using higher quality SN generation. However, such simple approaches tend to increase latency exponentially or require large amounts of extra hardware. Thus, for cost-effective error mitigation, it is essential to better understand accuracy-latency and accuracy-area tradeoffs. Key to this understanding is the biggest, but somewhat subtle, influencer of error—the value distribution of the input SNs. This influence is evident in the variance of an N-bit SN $\mathbf{X}$ which, for a binomial distribution of 0s and 1s, is $X(1 - X)/N$ and varies from 0.0 when $X = 0$ or 1, to $0.25/N$ when $X = 0.5$. This implies that, assuming $\mathbf{Z}$ is binomially distributed, an AND multiplier's output error is a maximum when $Z = XY$ is near 0.5, and a minimum when XY is near 0.0 and 1.0, as is readily verified experimentally.

Example 1: Consider an AND gate multiplier with two uncorrelated N-bit input bit-streams $\mathbf{X}$, $\mathbf{Y}$ and output bit-stream $\mathbf{Z}$. To gain a better sense of the influence of the input value distribution (IVD) on the multiplier's average error, we simulated it for $N = 64$ with one input value X held fixed and the other input Y assigned

various possible probability density functions (PDFs). Fig. 1 shows a typical set of results. For each simulation run, X was set to 0.70 and Y was chosen randomly from the unit interval [0.0, 1.0] according to two different PDFs, which are shown by dashed lines in Fig. 1. PDF1 (red) is a peaked IVD centered at 0.70 while PDF2 (blue) is a U-shaped IVD with most of its density concentrated near 0.0 and 1.0. If R is the number of simulation runs (50,000 in the case of Fig. 1) and $\hat{Z}_i$ is $\mathbf{Z}$'s estimated value during run i, the mean squared simulation error defined by

$$\text{MSE} = \frac{1}{R}\sum_{i=1}^{R}\left(\hat{Z}_i - Z\right)^2 \quad (1)$$

is $3.82 \times 10^{-3}$when using PDF1 to choose values for Y. However, it is 2.28×10^{-3} (40% lower) when using PDF2 to choose values for Y.

This large difference in error is due to PDF2's more favorable IVD for Y. PDF2 is higher for Y values that result in very low multiplication error while PDF1 has more density for Y values that result in high error. Thus, the circuit is more accurate on average when Y is drawn from PDF2. Evidently, Y's PDF exerts a large influence on the average output error. This leads to an important conclusion: having knowledge of the IVD enables a better understanding of average accuracy which can lead to a more informed choice about SN length and thus circuit latency. For instance, in this particular example, SN length could be set as low as 39 bits if Y is known to be distributed as in PDF2 and the average error would still be less than the average error when Y is distributed according to PDF1.

Several studies have derived error as a function of input values [3][4][6][15][16]. For example, Ex. 1's multiplication MSE for N-bit binomially distributed input SNs can be shown to be $XY(1 - XY)/N$. However, the impact of the SN IVD on average error has not been rigorously considered in previous SC error analyses, but it has very significant design implications. As Ex. 1 shows, when the input SNs' PDFs are favorable, bit-streams can be shortened and still meet a given accuracy threshold. Another place where the IVD is important is in constructing simulation experiments and interpreting simulation results.

Due to shortcomings in SC error theory, simulation has become the dominant method, and in many cases the only method, to determine and compare the accuracy of different stochastic designs. To simulate a circuit, SN values must be chosen, and this choice implicitly defines the IVD. Simulation accuracy is tied to this distribution choice, and the conclusions drawn may or may not remain valid when a different IVD is used.

To illustrate, it seems natural (and inconsequential) when performing simulation experiments to choose random SN values that are uniformly distributed, i.e., each SN value has the same probability of being selected. However, many real-world applications of SC, such as image processing, involve data that is far from uniform. A case in point is the MNIST database of handwritten characters which consists of thousands of images in the form of 28 × 28 arrays of grayscale pixels, and is widely used as a benchmark for image classification by neural networks (NNs) [14]. In SC-based image classifiers [8][17], the input SNs have grayscale values denoting pixel intensity that range from 0.0 (black) to 1.0 (white). The vast majority of these values are close to zero (similar to PDF2 in Fig. 1) because the corresponding images have a small pale foreground and large black background. A circuit's simulation-based accuracy performance on uniformly random data may not be representative of how the circuit will perform on unevenly distributed data such as MNIST. MNIST and SC-based NNs are further explored in Sec. 5.

To develop a quantitative understanding of how knowledge of the IVD impacts accuracy, we introduce a new Bayesian formulation of SC accuracy (which should not be confused with recent, unrelated work on the use of SC-based hardware to implement Bayesian inference [7]). A central idea in Bayesian probability theory is the use of prior knowledge, as in Bayes' theorem [10]

$$P(H|D) = P(D|H)P(H)/P(D) \quad (2)$$

where, for instance, H might be an error hypothesis and D might be some data influencing belief in the hypothesis H. Eq. (2) indicates how the prior probability or belief $P(H)$ in H directly affects the posterior belief $P(H|D)$ in the hypothesis given the data. Our analysis is Bayesian in the sense that it uses the IVD as informative prior data to better estimate overall error in a stochastic circuit.

Orthogonal to its application of Bayesian ideas, this paper aims to improve the analytic approach to SC error measurement when input values are known. It does so by grouping errors into systematic and random types and using bias-variance decomposition of mean squared error. The result is a consistent methodology to compute the error of a stochastic circuit. This loosens dependence on simulation to measure accuracy and enables a better understanding of accuracy trade-offs.

The main contributions of this work are:

1. A general Bayesian formulation of stochastic circuit accuracy that identifies limitations of current simulation approaches and suggests ways to improve them.
2. Introduction of bias-variance decomposition and the beta distribution to SC error analysis and their use to simplify SC and improve accuracy measurement.
3. Application of the forgoing ideas to accuracy analysis of SC-based NNs.

This paper is organized as follows. Sec. 2 reviews relevant background related to SC and probability theory, while Sec. 3 details a new Bayesian formulation of stochastic circuit accuracy. Sec. 4 then introduces techniques to model input value distribution. Finally, Sec. 5 gives a case study demonstrating the usefulness of these ideas.

2 Background

Here we review relevant concepts in SC, including basic component types and error analysis from an accuracy perspective.

2.1 SC Components

A compelling feature of SC is the simplicity of its basic arithmetic components [2][9]. For example, AND gates function as unipolar multipliers. (Henceforth, we will assume that only unipolar format is used, but all this paper's results can easily be modified to apply to bipolar SNs.) Fig. 2 shows two other basic components which

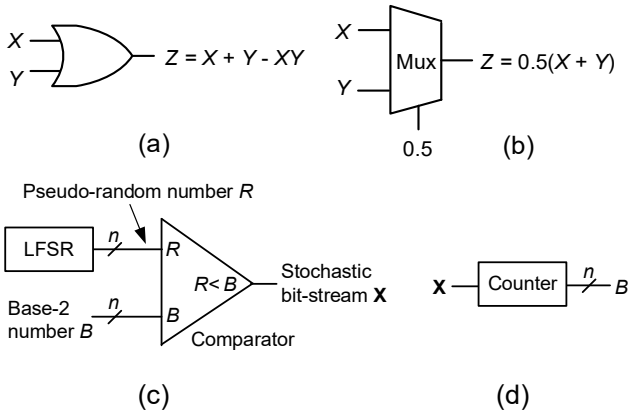

Figure 2: SC adders and data conversion: (a) OR gate computing the sum $X + Y - XY$; (b) mux computing the sum $0.5(X + Y)$; (c) stochastic number generator (SNG); (d) counter serving as a stochastic-to-binary converter.

function as SC adders; note their different ways of confining the sum Z to the unit interval. The OR gate is a biased adder computing $Z = X + Y - XY$, while the multiplexer (mux) serves as a scaled adder that computes $Z = 0.5(X + Y)$.

Fig. 2 also shows two basic circuits used for data conversion in SC. The SN generator (SNG) converts an n-bit binary number B to an N-bit stochastic bit-stream $\mathbf{X}$ of value $B/2^n$; the counter estimates $\mathbf{X}$'s value. The SNG contains a pseudo-random number source (RNS), which typically consists of an n-bit linear feedback shift register (LFSR) that outputs an integer in the interval $(0, 2^{n-1}]$ each clock cycle as it traverses its $2^n - 1$ states. Since an LFSR lacks the all-0 state, it never outputs the integer zero, which imposes a small bias on $\mathbf{X}$'s value.

2.2 SC Errors

Errors in stochastic circuits come from many sources. Due to the discrete nature of digital computers, stochastic circuits incur errors from function approximation and quantization in a similar manner to traditional binary computing. Two other error sources unique to stochastic computing are correlation and random fluctuation errors.

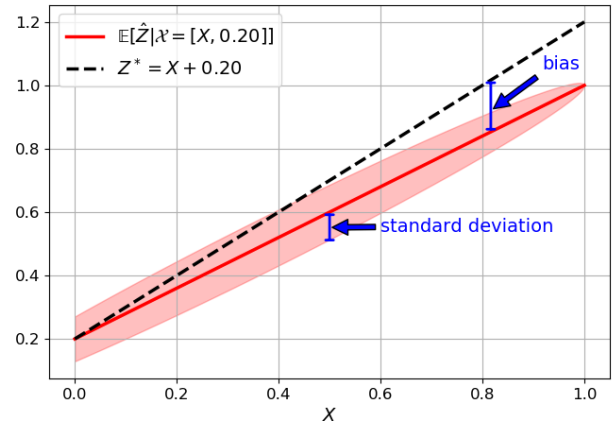

Figure 3: Visualization of bias-variance decomposition for an OR-gate adder with inputs $\mathbf{X}$, $\mathbf{Y}$, output $\mathbf{Z}$ and SN length $N = 32$. Y is set to 0.2 and plotted quantities are in terms of X. Standard deviation is the square root of variance.

Approximation error is the difference between the target function and the function implemented by the circuit. Approximating a target function can help simplify hardware if the approximating function is simpler than the target function or it can enable reconfigurable architectures where target functions must be approximated by a given functional form, such as a Bernstein polynomial [18]. Quantization error is due to the limited precision of SNGs. For example, a 4-bit SNG can only generate SNs with values $\left\{0, \frac{1}{16}, \frac{2}{16}, \ldots, 1\right\}$ and so an input SN with target value $\frac{7}{64}$ must be quantized to either $\frac{1}{16}$ or $\frac{2}{16}$.

Another error source is unwanted correlations. Most SC circuits are designed assuming that all input SNs are independent. In practice though, the input SNs may be correlated thus changing the circuit's expected output and introducing error [1][6]. Sometimes correlation is helpful. For example, in [3] it is shown that, contrary to expectation, suitably correlating the input SNs to a mux adder can improve accuracy.

To summarize, approximation, correlation, and quantization all affect the accuracy of a circuit; in other words, they change the circuit's expected output value. These errors are *systematic* and can be controlled or eliminated by careful design. They can also be grouped together under the single concept of circuit *bias*. In addition to systematic errors, there are errors due to the stochastic or probabilistic nature of the input signals. These are independent of the circuit itself and may be termed *random* errors. They are characterized by properties like expected value and variance and are addressed by statistical analysis and simulation.

3 Stochastic Circuit Accuracy

Stochastic circuits can be viewed as estimators of a target value. Through this lens, results and insights from statistical estimation theory can be used to improve the circuit design process. In many estimation problems, the random data source is fixed, and the goal is to derive an optimal estimation function that maps data to an estimate of the target value. In stochastic circuit design, we have the reverse situation. The estimation function is fixed (e.g., for unipolar output, it is the frequency of ones in the output) and the goal is instead to optimize the statistical properties of the output bit-stream which serves as the random data source.

In general, estimation error is defined as

$$\epsilon = \hat{Z} - Z^* \tag{3}$$

where $\hat{Z}$ is the circuit's estimate for the designer's target value Z^*. A cost or loss function $L(\epsilon)$ can be used to determine the severity of estimation error. No standard cost function exists for SC, but $L(\epsilon) = \epsilon^2$ (quadratic error) and $L(\epsilon) = |\epsilon|$ (absolute error) are common. We focus on quadratic cost due to its useful bias-variance decomposition, but the following formulation, especially (4) and (8), can readily be framed in terms of other cost functions.

For consistency with Bayesian theory, from here on we will treat the unipolar value $X_1 = p_x$, of an SN $\mathbf{X}_1$ as a random variable. The vector of input values to an M-input circuit is denoted by $\boldsymbol{\mathcal{X}} = [X_1, X_2, \ldots, X_M]$ and is treated as a random vector. $\mathbf{X}$ and $\boldsymbol{\mathcal{X}}$ should not be confused: the SN $\mathbf{X}$ is a random vector of bits, while $\boldsymbol{\mathcal{X}}$ is a random vector of SN values. An outcome of $\boldsymbol{\mathcal{X}}$ is denoted as $\boldsymbol{x}$.

3.1 General Error Formulation

Consider an M-input stochastic circuit C with random target input values $\boldsymbol{X} = [X_1, X_2, \ldots, X_M]$ and target function Z^* that maps $\boldsymbol{X}$ to a real number. C produces an N-bit output SN $\mathbf{Z} = z_1 z_2 \ldots z_N$ from which an estimator $\hat{Z}$ for Z^* is derived. The *mean squared error* (MSE) of $\hat{Z}$ given the input values, $\boldsymbol{X}$, is defined as

$$\mathrm{MSE}(\hat{Z}, Z^* | \boldsymbol{X}) = \mathbb{E}\left[\left(\hat{Z} - Z^*(\boldsymbol{X})\right)^2 \middle| \boldsymbol{X}\right]. \quad (4)$$

In standard statistical analysis [12], the MSE of an estimator such as $\hat{Z}$ can be decomposed into a combination of the estimator's variance and bias:

$$\mathrm{MSE}(\hat{Z}, Z^* | \boldsymbol{X}) = \mathrm{Var}(\hat{Z} | \boldsymbol{X}) + \mathrm{Bias}(\hat{Z}, Z^* | \boldsymbol{X})^2 \quad (5)$$

where

$$\mathrm{Var}(\hat{Z} | \boldsymbol{X}) = \mathbb{E}\left[(\hat{Z} - \mathbb{E}[\hat{Z} | \boldsymbol{X}])^2 \middle| \boldsymbol{X}\right] \quad (6)$$

$$\mathrm{Bias}(\hat{Z}, Z^* | \boldsymbol{X}) = \mathbb{E}[\hat{Z} | \boldsymbol{X}] - Z^*(\boldsymbol{X}). \quad (7)$$

Example 2: The foregoing bias-variance decomposition is illustrated in Fig. 3 which considers the use of an OR gate as an adder to approximate $Z^* = X + Y$, as in Fig. 2*a*. Here, $\hat{Z}$'s expected value $\mathbb{E}[\hat{Z} | \boldsymbol{X}] = X + Y - XY$ and Z^* are plotted against X when $Y = 0.20$. $\hat{Z}$'s variation is also shown. The difference between $\mathbb{E}[\hat{Z} | \boldsymbol{X}]$ and Z^* is $\hat{Z}$'s bias (7) which, in this case, is $XY = -0.2X$. Bias measures $\hat{Z}$'s systematic error with respect to approximating Z^*. The amount that $\hat{Z}$ fluctuates around its expected value is $\hat{Z}$'s variance (6), which measures $\hat{Z}$'s random error and is completely independent of Z^*. Note that the bias and variance depend on the input values. Sometimes bias is larger than standard deviation (the square root of variance) and sometimes the reverse is true. This is visualized in Fig. 3.

In the standard estimation theory embodied in (4), the target parameter Z^* is treated as a deterministic but unknown constant. But, as we now show, it can be very useful to view it as a random variable whose realization we must estimate. To handle this situation, a Bayesian approach is employed where Z^*'s distribution (or a related one) serves as a prior and the error conditioned on Z^* (or a related variable) is computed.

As Ex. 1 suggests, in SC $\boldsymbol{X}$ is generally a random vector whose distribution is application dependent. The target value Z^* is also random due to its dependence on $\boldsymbol{X}$, making a Bayesian approach to error analysis appropriate. Instead of using $Z^{*\prime}s$ distribution as a prior, $\boldsymbol{X}$'s distribution will be used since error is normally expressed in terms of input values. If our goal is to compute average error, then *Bayes mean square error* (BMSE) [12] denoted

$$\mathrm{BMSE}(\hat{Z}, Z^*) = \mathbb{E}[\mathrm{MSE}(\hat{Z}, Z^* | \boldsymbol{X})] \quad (8)$$

can be used where the expectation is taken with respect to $\boldsymbol{X}$. Intuitively, BMSE is a weighted sum where the summands are MSE values given $\boldsymbol{X} = \boldsymbol{x}$ (for all valid $\boldsymbol{x}$) and the weights are $\boldsymbol{X}$'s probability mass or density function evaluated at $\boldsymbol{x}$. The advantage of BMSE is that it measures the expected estimation error of a circuit *before* the input values are known. Combining Eqs. (5) and (8), BMSE can also be written as a bias-variance decomposition

$$\mathrm{BMSE}(\hat{Z}, Z^*) = \mathbb{E}[\mathrm{Var}(\hat{Z} | \boldsymbol{X})] + \mathbb{E}\left[\mathrm{Bias}(\hat{Z}, Z^* | \boldsymbol{X})^2\right]. \quad (9)$$

BMSE is well-known in estimation theory [12] and, as we show in this paper, it is a natural choice of error metric for SC. In fact, although not mentioned by name, it is what many simulation studies in SC already measure [4][11][18]. These studies begin by defining a rule for sampling circuit input values (e.g., the inputs are set to some fixed values like 0.0, 0.1, 0.2, 0.3, …, 1.0, or else they are randomly chosen from some fixed range). This step implicitly defines the distribution of $\boldsymbol{X}$. Then for each set of input values, $\boldsymbol{x}$, the circuit is simulated and the resulting error is measured many times thus producing estimates for $\mathrm{MSE}(\hat{Z}, Z^* | \boldsymbol{X} = \boldsymbol{x})$. The estimated MSE values for the $\boldsymbol{x}'$s are then averaged to produce a metric that represents the circuit's inherent accuracy. This metric is precisely an estimate for the circuit's BMSE. If another cost function is used in place of MSE, such as mean absolute error, this simulation approach is said to compute a *Bayesian risk*, a term that is more general than BMSE [12].

3.2 Comparing Circuit Accuracy

One way to compare two circuits' accuracy is by comparing their BMSEs, e.g., by computing the difference or ratio of the BMSEs. A somewhat limiting factor, though, is that BMSE depends on the IVD so one circuit may have a lower BMSE than another circuit for some IVDs, but not for others. This points to a major potential problem with current simulation approaches in SC. In many cases, only one method of choosing input values is used and thus only one input distribution is implicitly chosen. Thus, one BMSE per design is calculated and compared, but the comparison may not generalize well to other input distributions.

In some cases, however, it may be that one circuit is *never* less accurate (in terms of BMSE) than another circuit. This leads to the notion of *estimator dominance*. Formally, we say an estimator $\hat{Z}_1$ dominates another estimator $\hat{Z}_2$ with respect to estimating Z^* if $\mathrm{MSE}(\hat{Z}_1, Z^* | \boldsymbol{X} = \boldsymbol{x}) < \mathrm{MSE}(\hat{Z}_2, Z^* | \boldsymbol{X} = \boldsymbol{x})$ for at least one valid value $\boldsymbol{x}$, and $\mathrm{MSE}(\hat{Z}_1, Z^* | \boldsymbol{X} = \boldsymbol{x}) \leq \mathrm{MSE}(\hat{Z}_2, Z^* | \boldsymbol{X} = \boldsymbol{x})$ for all other values of $\boldsymbol{x}$. In other words, $\hat{Z}_1$ dominates $\hat{Z}_2$ when $\hat{Z}_1$ is more accurate for at least one input combination $\boldsymbol{x}$ and no less accurate than $\hat{Z}_2$ for all other input combinations, where accuracy is measured by MSE. (Dominance can also be phrased in terms of other cost functions.) The following result considers the case where two distinct estimators $\hat{Z}_1$ and $\hat{Z}_2$ are derived from circuits with input $\boldsymbol{X}$ and the same target function Z^*.

Theorem: If $\hat{Z}_1$ dominates $\hat{Z}_2$ then $\hat{Z}_1$ never has a greater BMSE than $\hat{Z}_2$ for any distribution of $\boldsymbol{X}$.

The theorem follows from the fact that BMSE is computed as a weighted sum of MSEs where all weights are positive and, in addition, the fact that $\hat{Z}_1$'s MSE never exceeds $\hat{Z}_2$ for any set of input values due to the definition of dominance. It implies the usefulness of proving dominance between estimators of two SC circuits. When a circuit's estimator dominates another circuit's estimator, then the former circuit is *always* at least as accurate as the latter circuit for any IVD.

3.3 Implications for Simulation Error Analysis

The foregoing formulation for SC errors points to two important considerations for error analysis. The first is that multiple IVDs should be considered when accessing the accuracy of a circuit design. These IVDs should be representative of data in the application domain of the circuit, or, in the case of general designs, should represent a variety of different distributions. The beta distribution presented in Sec. 4 and visualized in Figs. 1 and 5 can be useful for constructing and fitting input distributions with a variety of shapes. Alternatively, multiple datasets can be used to assess a circuit's accuracy (each dataset represents an IVD). In Sec. 5, we show an example where an SC-based neural network (NN) is evaluated using two datasets. We will see that their differences lead to strikingly different BMSEs.

The second consideration is that there are various ways to compute BMSE for a given IVD $f_{\boldsymbol{X}}$ depending whether the expected circuit output value $\mathbb{E}[\hat{Z}|\boldsymbol{X}]$ and variance $\mathrm{Var}(\hat{Z}|\boldsymbol{X})$ are known. There are several possibilities. For example, if both $\mathbb{E}[\hat{Z}|\boldsymbol{X}]$ and $\mathrm{Var}(\hat{Z}|\boldsymbol{X})$ are known, then bias can be found from (7) and BMSE from (9). If both are unknown, simulation and sampling can be used to estimate BMSE.

Example 3: Consider again the OR-gate adder of Fig. 2*a* and Ex. 2 that uses 32-bit SNs and fixes Y to 0.2. The MSE of this circuit assuming that X's value is uniformly sampled from the unit interval [0,1] can be found via simulation. Each simulation run, X is sampled randomly from [0,1] and the OR gate adder is simulated with X and $Y = 0.2$. The squared output error is recorded and then averaged across all 10,000 simulation runs to estimate MSE as in (1). In our experiment, we measured MSE = 0.0190.

According to analysis presented in Sec. 3.1, this MSE found using simulation is a Bayesian MSE and it can be found analytically using (9). To do this, the circuit's bias, variance and IVD must be determined as functions of X. Then integration can be used to calculate BMSE. The bias of the OR gate adder with $Y = 0.2$ is $-XY = -0.2X$. While the variance is

$$\mathrm{Var}(\hat{Z}|\boldsymbol{X}) = \frac{(X+Y-XY)(1-(X+Y-XY))}{N} \tag{10}$$

where N is the SN length [16]. This simplifies to $(-0.64X^2 + 0.48X + 0.16)/32$ since $Y = 0.2$ and $N = 32$ for this example. Finally, since X is sampled uniformly from [0,1], X's probability density is $f_X(x) = 1$ for $0 \le x \le 1$ and $f_X(x) = 0$ otherwise.

Eq. (9) can be re-expressed as

$$\mathrm{BMSE}(\hat{Z}, Z^*) = \int f_{\boldsymbol{X}}(\boldsymbol{x})\mathrm{Var}(\hat{Z}|\boldsymbol{X} = \boldsymbol{x})d\boldsymbol{x} + \int f_{\boldsymbol{X}}(\boldsymbol{x})\mathrm{Bias}(\hat{Z}, Z^*|\boldsymbol{X} = \boldsymbol{x})^2 d\boldsymbol{x} \tag{11}$$

by using the definition of the expectation operator. In this example, (11) becomes

$$\mathrm{BMSE}(\hat{Z}, Z^*) = \frac{1}{32}\int_0^1(-0.64x^2 + 0.48x + 0.16)dx - \int_0^1(0.2x)^2 dx \tag{12}$$

which evaluates to 0.0192 and is in close agreement to the simulation results.

Both the simulation approach and analytic approach based on (9) arrived at the same conclusion. The advantage of the analytic approach is that it is computationally more efficient which is important for large circuits that have many inputs and thus many parameters. Furthermore, the analytic approach can grant further insights. For example, the bias in Ex. 3 is $-0.2X$ and so we would expect the circuit to be more accurate when X is likely to take small values and less accurate when X is likely to take large values. These likelihoods are expressed mathematically as the distribution f_X of X. In the following section, we introduce ways to model f_X.

3.4 Application of Error Analysis

One key application of error analysis is determining the minimum latency (or equivalently, the SN length) required to achieve a user-specified level of accuracy. Since error varies with the circuit's input values, latency can be determined in terms of the worst-case error across any set of input values [16] or in terms of the expected error by considering the IVD. As we will see, these two approaches can give significantly different latencies.

The minimum latency required to meet a specified accuracy level depends on many factors that influence error such as SNG precision and correlation. For our next example, we assume that there is no correlation error and SN length is set to 2^n, where n is the SNG precision as is typical in SC experiments. Constraining the SN length to a power-of-two implies that a costly division circuit is not needed in addition to a counter for output SN value estimation.

Example 4: Consider an AND gate that multiplies two inputs **X** and **Y** which are generated with an *n*-bit SNG and whose target values X^* and Y^* are drawn from two independent PDFs and are rounded to n-bit precision by truncation. Therefore, **X** (**Y**) is generated with expected value $X = \lfloor X^*2^n \rfloor/2^n$ ($Y = \lfloor Y^*2^n \rfloor/2^n$) and the bias of the multiplier is $XY - X^*Y^*$. The variance of the multiplier is $XY(1 - XY)/N$ assuming **X** and **Y** are streams of $N = 2^n$ independent bits.

Under these assumptions, we derived the BMSE when X^* and Y^* are both independently drawn from either PDF1 or from PDF2 of Fig. 1 and plotted the results in Fig. 4a. Also shown in Fig. 4a is the predicted BMSE assuming the worst-case target input values occur with probability 1 or assuming a uniform distribution of target input values. The error curve is lowest for the PDF2 case because target SN values that result in low error occur with high probability. The opposite situation occurs for PDF1 where the error curve is high (nearly the worst case) because target SN values that result in high error occur with high probability. Finally, the average case error curve falls between PDF1's and PDF2's error curve, which highlights the fact that average case analysis is not representative of these (and many other) IVDs.

The relationship between error and latency can be used to derive minimum latency for a given accuracy. For instance, if the target MSE is 3×10^{-3}, then Fig. 4a shows that 32 bit SNs are needed when X^* and Y^* are drawn from PDF2 whereas 128 bit SNs are needed in the worst case or when X^* and Y^* are drawn from PDF1. This factor of 4 difference in required latency demonstrates the influence that the IVD can have on the error.

A major advantage of an analytic approach over a simulation-based approach to error analysis is that analysis can provide results that have better explanations, insights, and confidence levels. For example, Fig. 4b shows the bias-variance decomposition of MSE for both the PDF1 and PDF2 case of Ex. 4. It can be seen in both

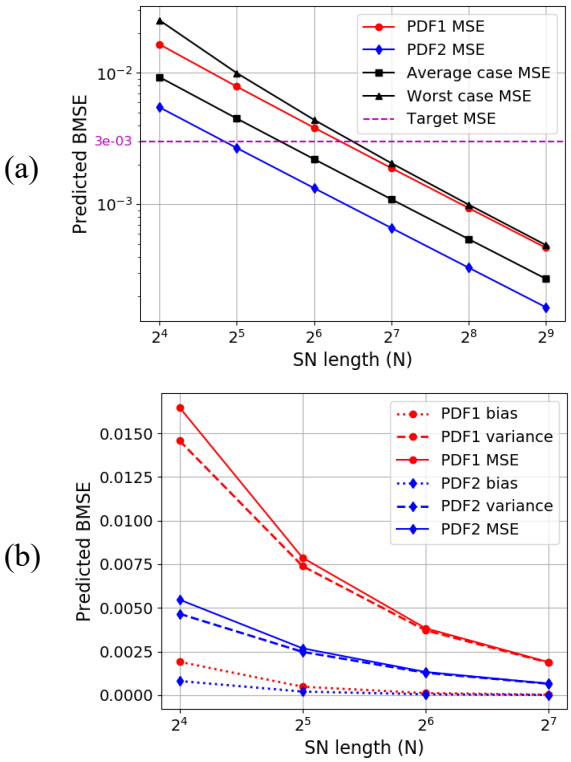

Figure 4: BMSE of an AND multiplier with target input values drawn from either PDF1 or from PDF2. (a) BMSE vs SN length for various IVDs; (b) Bias-variance decomposition of MSE for two IVDs. Note the log scales.

cases that bias due to quantization error is a small contributor to MSE compared to variance due to random fluctuations. This supports the common intuition that quantization error is negligible for multiplication with long bit-streams.

4 Flexible Input Value Distribution Model

Modeling the input SN value distribution for $f_{\boldsymbol{X}}$ requires some knowledge about $\boldsymbol{X}$ in the form of data or in the form of prior belief. As usual, we assume the unipolar format so that $X = p_x$. We begin by discussing how to use the beta distribution to model the value of a single input SN $\mathbf{X}$ that has a value X. We then generalize to modeling a vector of input values $\boldsymbol{X}$. Note that all SN values are treated as random variables.

4.1 Modeling a Single SN Value

The beta distribution has two shape parameters $\alpha > 0$ and $\beta > 0$, and its PDF is defined as

$$f_{beta}(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)} \tag{13}$$

where B is the beta function used to normalize the distribution [10][19]. B can be written in terms of the gamma function, Γ, the complex-valued generalization of the factorial function:

$$\mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{14}$$

The beta distribution is defined on the $(0,1)$ interval if $\alpha,\beta < 1$, or on the $[0,1]$ interval otherwise. It is thus a suitable distribution for modeling probabilities or, in this case, SN values.

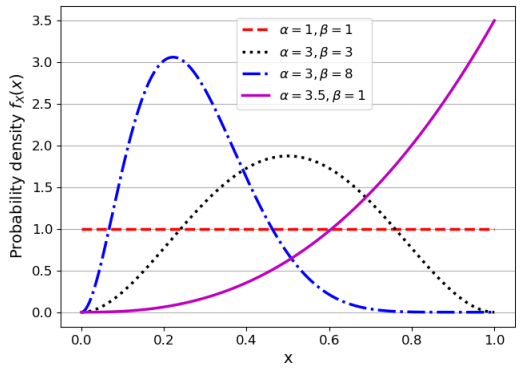

Figure 5: Beta PDFs for various alpha and beta values. If alpha and beta are swapped, the corresponding PDF is reflected around $x = 0.5$.

Fig. 5 shows the versatility of the beta distribution as α and β vary. The PDFs in Fig. 1 are also beta distributions where PDF1 has $\alpha = 16, \beta = 8$ and PDF2 has $\alpha = 0.5, \beta = 0.8$. The expected value and variance of the beta distribution are

$$\mathbb{E}[X] = \frac{\alpha}{\alpha+\beta} \tag{15}$$

$$\mathrm{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}. \tag{16}$$

Some intuition can be gained from these equations. For example, multiplying both α and β by a scalar s will not affect the mean, but will decrease the variance if $s > 1$. Therefore, if the beta distribution represents our belief about an SN's value, higher values of α and β imply more certainty about the value of the SN.

A limitation of the beta distribution is that it, if it is bimodal, the peaks must be at 0 and 1. In other words, the beta distribution cannot represent a bimodal PDF with a peaks at, say, 0.3 and 0.8. However, this limitation can be side-stepped by using a mixture model [19]. Rather than assume X is distributed as a single beta distribution, suppose it is distributed as a mixture of two beta distributions with parameters (α_1,β_1) and (α_2,β_2). X's PDF can then be defined as

$$f_X(x|\alpha_1,\alpha_2,\beta_1,\beta_2,\pi) = (1-\pi)f_{\mathrm{beta}}(x|\alpha_1\beta_1) + \pi f_{\mathrm{beta}}(x|\alpha_2,\beta_2) \tag{17}$$

where $0 \le \pi \le 1$ can be thought of as the probability or proportion of time that x is sampled from the second beta distribution with parameters α_2,β_2, rather than the first beta distribution with parameters α_1,β_1. This beta mixture model (17) has 2 components, but it can be extended to an arbitrary number of component beta distributions, where each has a coefficient π_i that determines its influence on X's overall distribution and the sum of π_i's is 1. Increasing the number of components increases the expressive power of the model. Parameter estimation for beta mixture models can be done using a modified expectation maximization algorithm presented in [19].

4.2 Modeling a Set of SN Values

One way to go from a modeling a single input value X to a set of input values $\boldsymbol{X} = [X_1, X_2, , \ldots, X_M]$ is to assume all X_i in $\boldsymbol{X}$ are independent thus implying the joint distribution of the elements of $\boldsymbol{X}$ is simply the product of the marginal distributions.

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{X_1}(x_1) f_{X_2}(x_2) \dots f_{X_M}(x_M) \quad (18)$$

where $\boldsymbol{x} = [x_1, x_2, \dots, x_M]$. Then, each f_{X_i} can be modeled separately using techniques for modeling a single SN value.

However, in some settings, the values to the circuit may be correlated. For example, values of adjacent pixels in an image tend to be highly correlated. It is important to note that two SNs having correlated values is not the same as the bits of two SNs being correlated. Two SNs can have correlated values but have independent bits. Likewise, two SNs can have correlated bits but independent values. In situations like these, modeling the input distribution becomes a domain-specific task. Sec. 5 gives an examples of modeling image data with a beta mixture model.

The IVD $f_{\boldsymbol{X}}$ is of interest because of its use to compute BMSE. Given a dataset, it is not necessary to construct an accurate model of the data's distribution to estimate BMSE but doing so provides a useful low-dimensional characterization of the dataset. Further, in cases where data cannot be made accessible, a parameterized model can instead be published to give other researchers the ability to understand the data used to test circuit accuracy.

The beta distribution is just one useful way to model an IVD. An advantage of this modeling approach is that *statistics* of the data, such as the expected value and variance, are used to fit the distribution to the data [19]. In some ways, it is like fitting data to a Gaussian except that beta distributions are defined on the bounded interval $[0,1]$ rather than over all real numbers like Gaussians (thus Beta distributions are useful for modeling uncertain probabilities or proportions [10]). Other IVD models such as a piecewise polynomial, may require fitting directly to the PDF or to the cumulative density function of the data rather than statistics of the data.

5 Case Study: Neural Networks

Next, we apply the foregoing methodology to an SC-based convolutional neural network (CNN) designed for image classification. This is a large-scale application of SC that usually demands high levels of classification accuracy [5][13][17].

Artificial neural networks have long been seen as a promising application for SC because of the huge number of multiplications they require to implement inner-product operations of the form $\sum_1^N W_i X_i$, where the W_i are constant weights and the X_i are variable inputs. The full use of SC in all layers of an CNN has been avoided mainly due to the fact that errors that tend to accumulate from layer to layer [17]. Identifying and measuring such errors is difficult and their mitigation is costly, requiring, for example, excessively long bit-streams, or hybrid designs that combine stochastic and deterministic features [8].

A basic part of a high-performing SC-based CNN design is shown in Fig. 6 [8]. To reduce errors and latency, the convolution layer is largely implemented in the SC domain where LFSRs (not shown) are used to generate the SNs. In this design style, positive and negative weights are separated into two groups and multiplication is performed using the unipolar format (absolute value of negative weights are used for SN generation). Both groups of products are

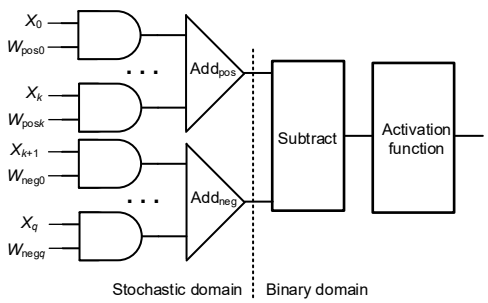

Figure 6: Convolution layer of a CNN based on the design in [8].

TABLE I
MEAN SQUARED ERROR (MSE) FOR MULTIPLICATIONS IN THE MNIST AND CIFAR10 CNNS OBTAINED BY SIMULATION AND ANALYSIS.

CNN	Simulated MSE	Analyzed BMSE
MNIST	1.18×10^{-5}	1.083×10^{-5}
CIFAR10	5.61×10^{-5}	5.55×10^{-5}

then summed and the difference is taken between the positive weight group's sum and negative weight group's sum to compute the overall inner product between weights and inputs, as shown in Fig. 6. Our goal is to characterize the multiplication accuracy of the CNN using simulation and the formalism presented in this paper.

5.1 Simulation

We trained two CNNs with structures similar to the classic LeNet-5 in [14]: one for the grayscale MNIST handwritten digit dataset [14] and one for the colored CIFAR10 natural image dataset [13]. These are popular 10-class benchmarks used in machine learning. For each CNN we simulated the first convolution layer in the SC-domain using the corresponding test dataset as input. SNs of length 256 and generated with 8-bit LFSRs were used as circuit inputs, and we recorded the squared error of all multiplication operations. The multiplication mean squared error for the MNIST and CIFAR10 CNNs are reported in Table I. Despite both CNNs being very similar, the MSE is four times higher for multiplication in the CIFAR10 CNN. This difference, as we will explain next, is almost entirely due to the distinct IVDs of MNIST and CIFAR10.

5.2 Error Analysis

To better understand the simulation results, the error formalism from Sec. 3, and the SN value modeling techniques from Sec. 4 will be used.

The variance of an AND multiplier with LFSR-generated inputs **X** and **W** and output **Z** was derived in [3] using the technique in [15].

$$\mathrm{Var}\left(\hat{\mathrm{Z}} \middle| X, W\right) = \frac{X(1-X)W(1-W)}{N-1} \quad (19)$$

The AND gate multiplier has no approximation error and there is no correlation present. Since the SC-based NNs use 8-bit LFSRs for SN generation, there is also very little quantization error. Thus, bias is approximately 0. The BMSE (9) can then be expressed as

$$\mathrm{BMSE} = \int_0^1 \int_0^1 f_{XW}(x,w) \mathrm{Var}\left(\hat{Z} \middle| X = x, W = w\right) dw dx \quad (20)$$

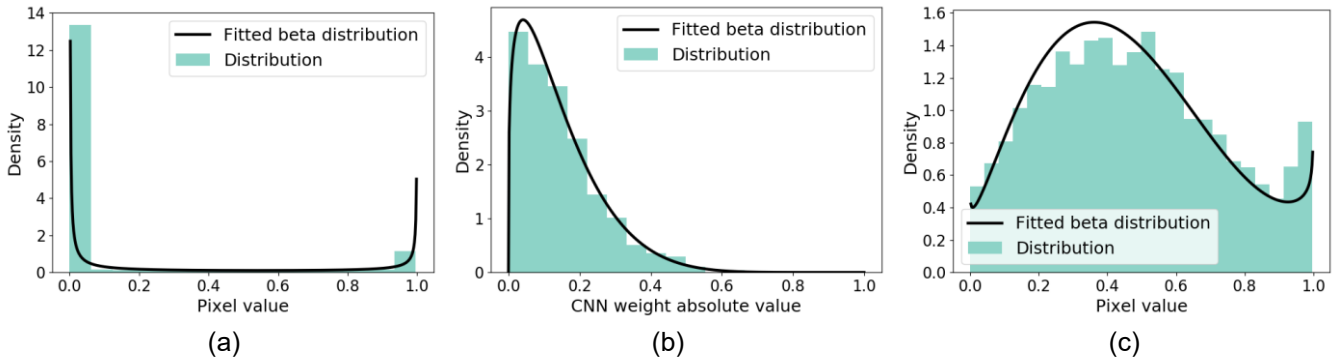

Figure 7: Distributions and their beta distribution fit: (a) MNIST pixel values; (b) MNIST CNN weights (absolute values); (c) CIFAR10 pixel values. Parameters for the beta distributions appear in Table II.

We will assume that the weight values and pixel values are approximately independent so that $f_{XW}(x,w) = f_X(x)f_W(w)$. This simplification and (19) can then be used to re-express (20).

$$\text{BMSE} = \frac{1}{\text{N}-1}\int_0^1 f_X(x)x(1-x)dx \int_0^1 f_W(w)w(1-w)dw \quad (21)$$

In this case, due to the form of (19), BMSE can be written as the product of two integrals of the same form as (21). The distributions f_X and f_W will be modeled as mixtures of k_X and k_W beta distributions, respectively. Each integral in (21) can then be re-expressed using the beta distribution PDF (13) and the k-component version of (17). For example

$$\int_0^1 f_X(x)x(1-x)dx = \sum_{i=1}^{k_X} \pi_i^{[X]} \int_0^1 \frac{x^{\alpha_i^{[X]}}(1-x)^{\beta_i^{[X]}}}{\text{B}(\alpha_i^{[X]},\beta_i^{[X]})}dx \quad (22)$$

where $\pi_i^{[X]}$, $\alpha_i^{[X]}$ and $\beta_i^{[X]}$ are the coefficient and parameters of the i-th component of the $X's$ beta mixture model. Eq. (22) can be further simplified by computing the definite integral.

$$\int_0^1 f_X(x)x(1-x)dx = \sum_{i=1}^{k_X} \pi_i^{[X]} \frac{\alpha_i^{[X]}\beta_i^{[X]}}{\alpha_i^{[X]}+\beta_i^{[X]}+1} \quad (23)$$

And finally, (23) can be used to re-express the two integrals in (21).

$$\text{BMSE} = \frac{1}{N-1}\sum_{i=1}^{k_X}\sum_{j}^{k_W} \pi_i^{[X]}\pi_j^{[W]} \frac{\alpha_i^{[X]}\beta_i^{[X]}\alpha_j^{[W]}\beta_j^{[W]}}{(\alpha_i^{[X]}+\beta_i^{[X]}+1)(\alpha_j^{[W]}+\beta_j^{[W]}+1)} \quad (24)$$

where $\pi_j^{[W]}$, $\alpha_j^{[W]}$ and $\beta_j^{[W]}$ are the coefficient and parameters for the j-th component of W's beta mixture model. Eq. (24) is the closed-form solution for the BMSE of an AND gate multiplier with LFSR-generated inputs whose values are modeled as independent beta mixture models. The possibility of such closed-form solutions is another advantage of using the beta distribution.

Next, we fit beta mixture models to f_X and f_W for both the MNIST and CIFAR10 distributions using training data and the algorithm in [19]. We considered beta mixture models of 1, 2 and 3 components and selected the best-fitting model of the three. The computed parameters are reported in Table II, while the distributions and fits are shown in Fig. 7. The CIFAR10 pixel distribution was fit with a two-component mixture model whereas all other distributions were fit with a single component mixture model (i.e., a single beta distribution). The CIFAR10 CNN weight distribution is similar to the MNIST case and is not shown in Fig. 7. The beta mixture model fits are representative of the target distribution in all cases.

BMSE can now be derived analytically using the fitted model parameters found in Table II with (24). The BMSE derived from this analysis is reported in Table I. There is close agreement between the MSE measured during simulation (also shown in Table I) and BMSE derived analytically with the foregoing analysis.

Some insights can be gained from the derivation. For example, Eq. (19) implies that multiplication variance (and thus error) is highest with $X = W = 0.5$ and lowest when $X = 0, 1$ or $W = 0, 1$. As Fig. 7 shows, pixel values, X, in the MNIST dataset almost always take value 0 or 1 and very rarely take values in between, implying the multiplication will be very accurate. In the CIFAR10 case, however, pixel values vary across the whole interval [0,1], but the weights are concentrated towards 0, implying that multiplication will be fairly accurate, but not as accurate as the MNIST case.

TABLE II
FITTED BETA DISTRIBUTION PARAMETERS FOR THE MNIST AND CIFAR PIXEL DISTRIBUTIONS.

Distribution	π_i	α_i	β_i
MNIST pixels	1	0.0362	0.1817
CIFAR pixels	0.6653	2.2919	3.2944
	0.3347	0.9236	0.8315
MNIST weights	1	1.2800	7.5486
CIFAR weights	1	1.6704	17.782

6 Conclusion

We introduced a methodology for Bayesian error analysis in SC that overcomes some limitations of simulation-based accuracy determination, e.g., measuring the influence of IVD. We showed how the Bayesian methodology treats input values as probabilistic and estimates average output error. We also introduced the use of beta distributions to model IVD. Orthogonal to these ideas, we used bias-variance decomposition to group SC errors into two distinct classes, systematic and random, thus providing a deeper understanding of circuit error sources and their effects. Finally, we validated all the above with a case study which clearly shows that IVD can have a major impact on the accuracy of SC-based CNNs.

Acknowledgement. This research was supported by the U.S. National Science Foundation under Grant CCF-2006704.

REFERENCES

[1] Alaghi, A. and J.P. Hayes, "Exploiting correlation in stochastic circuit design." Proc. Intl. Conf. Computer Design, 39–46, 2013.

[2] Alaghi, A., W. Qian and J.P. Hayes. "The promise and challenge of stochastic computing." *IEEE Trans. CAD*, 37, 1515-1531, 2018.

[3] Baker T.J. and J.P. Hayes. "The hypergeometric distribution as a more accurate model for stochastic computing." *Proc. Design Autom. and Test in Europe Conf.,* 592-597, 2020.

[4] Braendler, J. D., T. Hendtlass and P. O'Donoghue, "Deterministic bit-stream digital neurons," *IEEE Trans. Neural Nets*., **13**, 1514–1525, 2002.

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

[6] Chen, T-H. and J.P. Hayes. "Analyzing and controlling accuracy in stochastic circuits." *Proc. Intl Conf. Comp. Design*, 367–373, 2014.

[7] Faix, M. et al. "Design of stochastic machines dedicated to approximate Bayesian inferences," *IEEE Trans. Emerging Topics in Comp,* **7**, 60-66, 2019.

[8] Faraji, S.R. et al. "Energy-efficient convolutional neural networks with deterministic bit-stream processing." *Proc. Design Autom. and Test in Europe,* 1757-1762, 2019.

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

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

[11] Ichihara, H. et al. "Compact and accurate digital filters based on stochastic computing." *IEEE Trans. Emerging Topics in Comp.*, **7**, 31-43, 2019.

[12] Kay, S.M. *Fundamentals of Statistical Signal Processing: Estimation Theory,* Prentice Hall, Upper Saddle River, NJ, 1993.

[13] Krizhevsky, A. "Learning multiple layers of features from tiny images." Master's Thesis, Dept. of Comp. Sci., Univ. of Toronto, 2009.

[14] LeCun, Y. et al. "Gradient-based learning applied to document recognition, *Proc. IEEE*, **86**, 2278-2324, 1998.

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

[16] Neugebauer, F., I. Polian and J.P. Hayes. "Framework for quantifying and managing accuracy in stochastic circuit design." *ACM Jour. Emerging Technologies in Comp. Sys*.. **14**, article 31, 2018.

[17] Neugebauer, F., I. Polian and J.P. Hayes. "On the limits of stochastic computing," *Proc. Intl. Conf. Rebooting Computing*, 98-106, 2019.

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

[19] Schröder, C. and S. Rahmann. "A hybrid parameter estimation algorithm for beta mixtures and applications to methylation state classification." *Algorithms Mol. Biol*. **12** article 21, 2017.