Special Issue

Te-Hsuan Chen*, Armin Alaghi, and John P. Hayes

Behavior of stochastic circuits under severe error conditions

Abstract: Stochastic computing is an old and unconventional computing technique that is finding promising new applications in image processing and the handling of complex error-correcting codes. Stochastic circuits offer an alternative to conventional digital circuits because of their extremely small size and inherent noise tolerance. They are also well-suited to meeting the requirements of emerging nanoscale technologies where non-deterministic behavior due to manufacturing defects and soft errors cannot be ignored. Error analysis of stochastic circuits, however, has received little attention and remains a largely open problem, especially when multiple errors affecting both the data sources and the stochastic circuits can occur in the course of a computation. This paper attempts to analyze stochastic circuits under various error conditions, and to compare their behavior to that of conventional circuits under similar error conditions. We use probabilistic transfer matrices for this analysis, complemented by circuit simulation. Our results indicate that stochastic circuits provide significantly better error tolerance under severe error conditions.

Keywords: ACM CCS → Hardware → Hardware validation, ACM CCS → Hardware~Emerging technologies → Analysis and design of emerging devices and systems, ACM CCS → Computer systems organization → Architectures, Digital circuits, error analysis, soft errors, stochastic computing.

***Corresponding Author: Te-Hsuan Chen,** Advanced Computer Architecture Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, e-mail: tehsuan@eecs.umich.edu
Armin Alaghi, John P. Hayes: Advanced Computer Architecture Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan

1 Introduction

Non-deterministic behavior is becoming ubiquitous in digital circuits implemented using conventional CMOS transistors or various novel nanotechnologies [5]. Because of their small size, these circuits are easily affected by small manufacturing defects and by transient errors due to environmental noise, both of which tend to be non-deterministic in nature. For example, carbon nanotube field-effect transistors (CNFETs), an emerging alternative to CMOS, exhibit behavioral variations that are difficult to identify and control [19]. Methods of designing circuits that tolerate errors are also of increasing interest. Von Neumann took the first steps in designing reliable circuits using unreliable switches in the 1950s [21]. Since then, many error-tolerant techniques have been proposed, ranging from error-correcting codes [15], to replication of hardware, software and/or data [12, 17]. Most of these approaches impose high circuit overhead, and tend to be used only in the most cost-insensitive applications.

Stochastic computing (SC) processes data in the form of (pseudo) random sequences of 0 s and 1 s [2, 7]. These bit-streams are referred to as stochastic numbers (SNs) and are interpreted as probabilities. In its basic “unipolar” form, a bit-stream X of length n containing n_1 1 s denotes the probability $p_X = n_1/n$, i. e., the probability of a randomly-chosen bit of X being 1. The same value p_X is represented by many different bit-streams, so this number format is highly redundant. For example, 0010, 01000001 and 00001100 all denote the same number 1/4. Unipolar SNs directly approximate numbers lying in the interval [0, 1], but they can be scaled to other ranges such as the interval [−1, 1] used for signed numbers (the so-called “bipolar” SNs) [2].

The key advantage of SC is that arithmetic operations can be performed by extremely simple and standard logic circuits using any desired hardware implementation technology. Figure 1 shows stochastic circuits for multiplication, addition, and number format conversion. In this figure (and in the rest of the paper), circuit wires and the signals (Boolean variables) they carry are denoted by small letters x, y, z,…, bit-streams or SNs are denoted by big

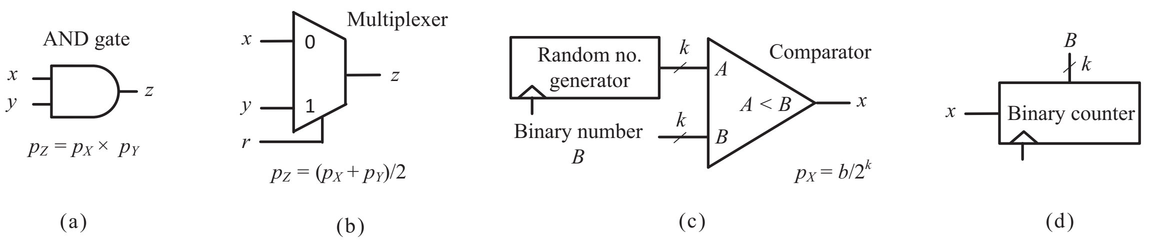


Figure 1: A selection of components for (unipolar) stochastic computing: (a) multiplier, (b) scaled adder, (c) binary-to-stochastic converter, and (d) stochastic-to-binary converter.

letters $X, Y, Z, \dots$, and their numerical values are denoted by p_X, p_Y, p_Z, etc. For example, the two-input AND gate performs the multiplication $p_X \times p_Y$ in n clock cycles, with the caveat that the input bit-streams X and Y must be statistically independent (uncorrelated) and sufficiently long to provide acceptable precision. Addition is implemented by the two-way multiplexer in the scaled form $0.5(p_X + p_Y)$, which ensures that the sum always lies in the probability interval [0, 1]. Observe that the scaling factor 0.5 is supplied by a stochastic number R of constant value 0.5, i. e., a purely random sequence of 0 s and 1 s. The remaining circuits convert numbers between conventional binary and stochastic forms. They serve to interface standard and stochastic digital circuits, and to re-randomize SNs that have become unduly correlated. The (pseudo) random number generator in Figure 1c is generally implemented by a linear feedback shift register (LFSR). Number conversion circuits sometimes constitute the largest part of an SC system.

SC has several features that limit its usefulness, including long bit-streams and computing times, inaccuracies caused by random bit fluctuations, and awkward scaling requirements. However, because of their inherently redundant encoding format, errors of the bit-flip type have little effect on the value of an SN. For example, a single bit-flip occurring in an n-bit SN changes the output value by $1/n$, a relatively small error whose significance diminishes as n increases. Furthermore, if the errors are bidirectional, i. e., if 0-to-1 and 1-to-0 bit-flips are equally likely, then the errors tend to cancel one another. This suggests that stochastic computing can outperform binary in applications such as noisy image processing [3, 14] and complex error-correcting decoding based on probability calculations [6, 9, 16]. Figure 2 compares the behavior of stochastic and conventional image-processing circuits in the presence of different levels of noise in the input data. It shows that when noise in the image data causes conventional image processing to fail, stochastic circuits can still produce acceptable results.

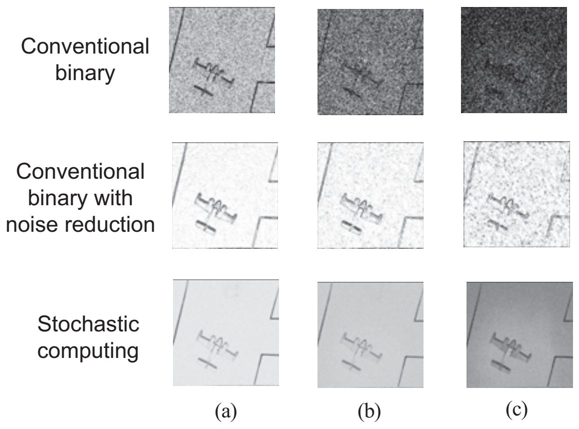


Figure 2: Comparison of stochastic and conventional edge-detection circuits for various noise rates in the input data: (a) 5%, (b) 10%, and (c) 20% [3].

Although the error-tolerance property of stochastic circuits has long been recognized, it has never been thoroughly analyzed. In [18], a soft error analysis of SC circuits is carried out that is limited to bit-flips occurring in the circuit's input data; the stochastic circuits themselves are assumed to be fault- or error-free. In this work, we aim to provide a more general error analysis for stochastic circuits, especially in the presence of high error rates such as are encountered in avionics or spacecraft instrumentation [8].

We first develop a mathematical framework for the analysis using Bernoulli random variables (BRVs) and probabilistic transfer matrices (PTMs). BRVs are widely employed in probability studies [20], and PTMs are used to determine probabilistic properties of logic circuits, such as their reliability or soft-error rate [13]. We also present a case study in image processing, which shows that stochastic circuits can outperform conventional ones under severe error conditions.

2 Basic concepts

This section discusses stochastic computations in terms of random variable theory, and reviews the key ideas of PTM algebra.

2.1 Stochastic numbers

A stochastic number X, as defined in [2], is a bit-stream carrying a probability value p_X that denotes 1 (success) or 0 (failure). It can therefore be viewed as a set of samples from a real-valued random variable (RV) with a Bernoulli distribution in which the probability of success is p_X [20]. Since probabilistic behavior can be easily modeled and analyzed in terms of Bernoulli RVs, we now use them to give a formal definition of a stochastic number that abstracts away from bit-stream formatting issues: a *stochastic number* X is a Bernoulli random variable with parameter p_X.

When dealing with RVs, we usually need to sample them in order to estimate their values. The sampling process is, in fact, a very basic form of stochastic computing. For instance, assume that the AND-gate multiplier of Figure 1a has two input SNs X and Y with known values p_X and p_Y, but the output is an SN Z of unknown value p_Z. Stochastic computation with this circuit involves generating samples for X and Y and measuring the success rate at z, and hence estimating p_Z. The expected rate of success at z can be calculated by the *expected value* operator denoted as $E[Z]$. Consequently,

$$p_Z = \mathbb{E}[Z] = \mathbb{E}[X \times Y] = \mathbb{E}[X] \times \mathbb{E}[Y] = p_X \times p_Y$$

assuming X and Y are independent RVs. For example, if $p_X = 0.2$ and $p_Y = 0.3$, then $p_Z = 0.06$, which is the expected rate of success at z. In practice, the success rate is affected by random fluctuations of the data, and usually has a different value $\hat{p}_Z$, which we refer to as the *estimated* value in contrast with the *exact* value p_Z. The estimated value $\hat{p}_Z$ is obtained by sampling the circuit/RV n times and recording the number n_1 of 1 s appearing at the output; this yields $\hat{p}_Z = n_1/n$. For example, if the RV Z, with the expected value $p_Z = 0.3$, is sampled 8 times, one possible outcome is 01100000, and the resulting estimate is $\hat{p}_Z = 2/8 = 0.25$.

In general, $\hat{p}_Z$ can be any of the 2^n different bit-streams derived from random sources, which allows p_Z and $\hat{p}_Z$ to differ, sometimes significantly, from one another. This difference between p_Z and $\hat{p}_Z$ is considered to be an error caused by randomness in the bit-stream representation of p_Z. Such *random fluctuation errors* are usually measured by the *mean square error* (MSE) $E_Z = E[(\hat{p}_Z - p_Z)^2]$. In the case of the Bernoulli RV's of interest here, we have [20]

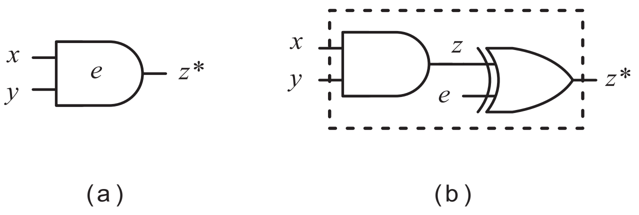


Figure 3: Circuit models for a stochastic multiplier with a bit-flip error e affecting its output: (a) internal or built-in error, and (b) externally injected error.

$$E_Z = p_Z(1 - p_Z)/n \qquad (1)$$

Thus the MSE of an SN estimate can be reduced by increasing the number of samples i. e., the bit-stream length n. Also note that E_Z is a function of p_Z and n only, implying that no matter what the circuit is (whether the AND of Figure 3 or any other circuit), once the expected rate of success p_Z at the output is calculated, we can use Equation (1) to calculate its MSE.

Besides the random fluctuations inherent in the selection of a particular bit-stream to represent Z in a stochastic circuit C, various non-deterministic physical phenomena associated with C itself and its environment affect the sampling process and distort the expected values of Z. It is convenient to lump such effects into a *bit-flip error* e that occurs with some probability p_e. For example, it is often assumed in the literature [4] that e causes bit-flips in Z, which affect 0 s and 1 s with equal probability p_e. Whatever, the error behavior assumed, two basic questions should be addressed: How do we model the impact of e on z, and how do we introduce e into a previously error-free stochastic circuit C? First, we assume the error e to be a Bernoulli RV with parameter p_e (the bit-flip rate), so it can be treated like another SN associated with C. The circuit's fault-free output z then changes to an erroneous function z^*, as illustrated in Figure 3a for the AND-gate stochastic multiplier. For simulation purposes, it is convenient to have a mechanism for injecting the error in a way that flips the normal signal z with probability p_e, resulting in the erroneous output z^*. Figure 3b shows how to do this by inserting an XOR gate with input e into C's output line. For example, a bit-flip rate of $p_e = 0.05$ with input values $p_X = 0.2$ and $p_Y = 0.3$ changes the expected success rate at the output of the AND multiplier from 0.06 to 0.104. An analytical method of calculating the expected value p_{Z^*} and its MSE will be developed next.

$$\begin{bmatrix} p_{e0} & 1-p_{e0} \\ p_{e1} & 1-p_{e1} \\ p_{e2} & 1-p_{e2} \\ 1-p_{e3} & p_{e3} \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) (b) (c) (d)

Figure 4: Representative PTMs: (a) NAND gate with four distinct input-dependent bit-flip error rates, (b) NAND gate with its first input stuck-at-1, (c) fanout wiring network with two output branches, and (d) swap or crossover gate that switches the order of two wires.

2.2 Probabilistic transfer matrices

A convenient tool for analyzing the probabilistic behavior of logic circuits is the probabilistic transfer matrix (PTM) and its associated algebra [13]. PTMs were introduced to analyze the reliability of conventional logic circuits. Their use may be limited by the fact that PTM size grows exponentially with circuit size. This is less of a problem with stochastic circuits, however, which typically consist of just a handful of gates.

In the PTM formulation, an n-input m-output combinational circuit's behavior is represented by a $2^n \times 2^m$ zero-one matrix whose rows correspond to all input combinations and whose columns correspond to all output combinations. This matrix, which is referred to an *ideal transfer matrix* (ITM), is a slightly modified truth table. For instance, a two-input AND gate has the ITM

$$J_{\text{AND}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

where the rows correspond to $xy = 00$, 01, 10, 11 and the columns correspond to $z = 0, 1$. A general PTM is obtained from the ITM by allowing the entry in row r and column c to become any real number in the interval $[0, 1]$ that denotes the conditional probability of producing output c in response to input r. In the AND gate's ITM J_{AND} shown in (2), the top row tells us that the AND produces output $z = 0$ with probability 1, and output $z = 1$ with probability 0, in response to the input $xy = 00$. By choosing suitable probability values, PTMs can be constructed to represent a remarkably wide range of error scenarios [13]. For example, the effect of a bit-flip error e with rate p_e on the output of the AND gate model in Figure 3a, is represented by the PTM

$$M_{\text{AND}} = \begin{bmatrix} 1-p_e & p_e \\ 1-p_e & p_e \\ 1-p_e & p_e \\ p_e & 1-p_e \end{bmatrix} \tag{3}$$

Input-dependent bit-flips can be modeled by associating a different p_e value with every row. Observe that a PTM must satisfy the stochastic requirement that all entries in each row sum to 1, and that the ITM is just the PTM for the error-free case.

Circuit PTMs can be manipulated by means of a well-defined algebra which loosely resembles linear algebra. Every element or circuit C is represented by a PTM that describes C's logic function and error status; see Figure 4. An n-output fanout gate F_n copies an input signal to its n outputs. Figure 4c shows the ITM for the 2-output fanout gate F_2. Wire permutations such as crossing wires are represented by the swap or crossover gate. The ITM for an adjacent wire swap is shown in Figure 4d; any permutation of wires can be modeled by a series of adjacent swaps. PTMs can be combined in two basic ways corresponding to the two basic circuit interconnection structures, series and parallel. The PTM of two circuits C_1 and C_2 connected in series is the ordinary matrix product of their PTMs, i. e., $M_1 \times M_2$. The PTM of two circuits connected in parallel is the tensor product of the PTMs, denoted $M_1 \otimes M_2$. In the tensor product, each element of the first matrix M_1 is multiplied by the entire second matrix M_2, which leads to rapid growth in matrix size. A wire corresponds to a 2×2 PTM; the ITM case is just the identity matrix. A signal is represented by a 1×2 row vector $[p_0 \;\; p_1]$, where p_0 and p_1 are the probabilities of the signal being 0 and 1, respectively. Signal vectors may be treated as a special kind of PTM, and can be manipulated with the same basic PTM operations.

For example, consider again the faulty AND-gate multiplier of Figure 3a with $p_X = 0.2$ and $p_Y = 0.3$, and $p_e = 0.05$. To determine the probability of getting a 1 at the gate's output, we first form the input vectors, namely, $M_x = [0.8 \;\; 0.2]$ and $M_y = [0.7 \;\; 0.3]$. These are then combined via the tensor product

$$M_{xy} = M_x \otimes M_y = \begin{bmatrix} 0.56 & 0.24 & 0.14 & 0.06 \end{bmatrix} \tag{4}$$

to give the probabilities associated with all four possible input combinations. The resulting input vector is multiplied by the PTM of the error-affected AND gate to obtain the circuit's output vector.

$$M_{z^*} = M_{xy} \times M_{\text{AND}} = \begin{bmatrix} 0.56 & 0.24 & 0.14 & 0.06 \end{bmatrix} \times \begin{bmatrix} 0.95 & 0.05 \\ 0.95 & 0.05 \\ 0.95 & 0.05 \\ 0.05 & 0.95 \end{bmatrix} = \begin{bmatrix} 0.896 & 0.104 \end{bmatrix}$$

From this, we conclude that p_{z^*}, i. e., the probability of seeing a 1 in z^*, is 0.104. Note that the PTM M_{AND} of the AND gate implicitly incorporates the bit-flip error, so there is no need for the XOR gate of Figure 3b, as is also the case in (3).

3 Error analysis

Next, we discuss the impact of bit-flips on stochastic numbers and stochastic circuits. We present a general analytical approach which can be implemented with PTMs to handle larger circuits.

3.1 Impact of soft errors on stochastic numbers

Consider a stochastic number X with the expected value $\mathbb{E}[X] = p_X$. In a noisy environment, if X is affected by bit-flip error e with expected value p_e, the SN becomes $X^* = X \oplus e$. We therefore have

$$p_{X^*} = \mathbb{E}\left[X^*\right] = p_X + p_e(1 - 2p_X) \tag{5}$$

Besides the expected value of X^*, we are interested in E_{X^*}, the mean square error of X^*, which denotes the average error occurring in a stochastic circuit, i. e., the average difference between the estimated value $\hat{p}_{X^*}$ and the exact value p_X.

$$E_{X^*} = \mathbb{E}\left[\left(\hat{p}_{X^*} - p_X\right)^2\right]$$

Note that E_{X^*} reflects both the random fluctuations of the bit-stream representation and the error e due to bit-flips. As mentioned earlier, X^* is a Bernoulli RV defined by its expected value, so using only (5), we should be able to find p_{X^*} and hence E_{X^*} analytically. Assuming the estimated value $\hat{p}_{X^*} = 1/n \sum_{i=1}^{n} X_i^*$ obtained by summing n independent samples of X^*, we get

$$\begin{aligned} E_{X^*} &= \mathbb{E}\left[\left(\hat{p}_{X^*} - p_X\right)^2\right] = \mathbb{E}\left[\hat{p}_{X^*}^2 + p_X^2 - 2p_X\hat{p}_{X^*}\right] \\ &= \mathbb{E}\left[\hat{p}_{X^*}^2\right] + \mathbb{E}\left[p_X^2\right] + \mathbb{E}\left[-2p_X\hat{p}_{X^*}\right] \\ &= \frac{n^2 p_{X^*}^2 + np_{X^*}(1 - p_{X^*})}{n^2} + p_X^2 - 2p_Xp_{X^*} \\ &= \left(p_{X^*} - p_X\right)^2 + \frac{p_{X^*}(1 - p_{X^*})}{n} \end{aligned} \tag{6}$$

There are two terms in the final result of (6). The first term is the difference between the expected values of X and X^*, and its only cause is the bit-flip e. The second term is a random fluctuation error that diminishes with increasing n. We can re-write E_{X^*} in terms of p_X and p_e by substituting (5) into (6) thus:

$$\begin{aligned} E_{X^*} &= p_e^2\left(1 - 2p_X\right)^2 \\ &+ \frac{1}{n}\left[p_X\left(1 - p_X\right) + p_e(1 - p_e)\left(1 - 4p_X\left(1 - p_X\right)\right)\right] \end{aligned} \tag{7}$$

Observe that the MSE error depends on both p_X and p_e. For sufficiently large n, E_{X^*} becomes 0 when $p_X = 1/2$, while it becomes p_e for $p_X = 1$.

In a similar way, we can analyze the effect of bit-flip errors on conventional (non-stochastic) binary numbers. An

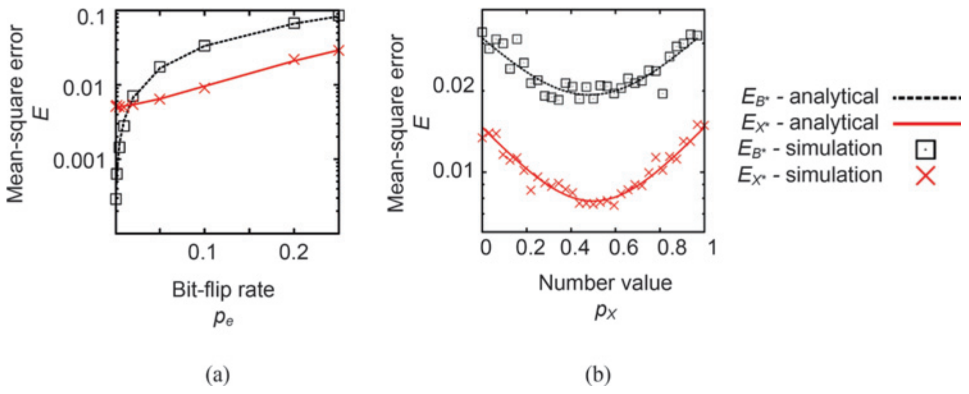


Figure 5: MSE of a stochastic number E_{X^*} and a binary number E_{B^*} in the presence of bit-flips calculated using analytical and simulation methods; (a) for different values of p_e and (b) for different values of p_X.

m-bit binary number B, affected by independent and identically distributed bit-flips on each bit becomes B^*, which can potentially be any m-bit number with some probability. The error of B^* and its probability of occurrence depend on the number of bit-flips m_{bf}. To find the MSE E_{B^*} in this case, we calculate the weighted average error over all possible B^* values.

$$E_{B^*} = \sum_{B_i^*=0}^{2^m} \left(B_i^* - B\right)^2 p_e^{m_{bf}} \left(1 - p_e\right)^{m-m_{bf}}$$

Using the above equations, we compare the effect of bit-flips on a 5-bit binary number and an SN of length 32. Figure 5a shows the MSEs E_{X^*} and E_{B^*} at different bit-flips rates. Initially, the SN has a higher error due to its random fluctuations. However, as p_e increases, SN outperforms the binary number with respect to error tolerance. Figure 5b shows the MSEs E_{X^*} and E_{B^*} at different values of p_X; the MSEs in this case are averaged over several bit-flip rates ranging from $p_e = 0.001$ to 0.25. As can be seen, E_{X^*} is approximately 50% less than E_{B^*}. These analytical results are also confirmed in Figure 5 by Monte Carlo simulation.

3.2 Impact of errors on stochastic circuits

A key feature of our error analysis is the use of PTMs to estimate the impact of errors on stochastic behavior. PTMs can be used to calculate the probability distribution of all output combinations of a stochastic circuit C. Given specific input signal probabilities, the input vector is multiplied by the PTM of the circuit C to obtain the output probabilities. For example, consider again the AND gate of Figure 3 which is multiplying two SNs X and Y and has output error p_e. Generalizing (4) gives the 1×4 input vector

$$M_{xy} = \big[\ (1 - p_X)(1 - p_Y) \quad (1 - p_X)\, p_Y \quad p_X (1 - p_Y) \quad p_X p_Y\ \big]$$

Now, let us consider two cases: first, the AND gate is error-free, and second, it contains the error e defined by the PTM in (3). In the error-free case, the output vector is

$$M_{xy} \times J_{\text{AND}} = \big[\ 1 - p_z \quad p_Z\ \big] = \big[\ 1 - p_X p_Y \quad p_X p_Y\ \big]$$

indicating that the probability of output 1 is $p_X p_Y$. If an error e is present, then using M_{AND} from (3), the output becomes

$$M_{xy} \times M_{\text{AND}} = \begin{bmatrix} (1 - p_e)\left((1 - p_X)(1 - p_Y) + (1 - p_X)\, p_Y + p_X (1 - p_Y)\right) + p_e p_X p_Y \\ p_e\left((1 - p_X)(1 - p_Y) + (1 - p_X)\, p_Y + p_X (1 - p_Y)\right) + (1 - p_e)\, p_X p_Y \end{bmatrix}^T$$

where T denotes matrix transposition (used to save space). This implies that the expected value of the output is $p_{Z^*} = p_e((1-p_X)(1-p_Y)+(1-p_X)p_Y+p_X(1-p_Y))+(1-p_e)p_X p_Y$. The MSE E_{Z^*} can now be calculated from Equation (6).

We can readily generalize the above technique to arbitrary stochastic circuits to analyze their stochastic behavior under single or multiple errors. First, generate the PTMs and ITMs for each individual logic or wiring gate. Then, apply the ordinary and tensor products repeatedly to calculate the PTM and ITM for the entire circuit [13]. Again, if the circuit has n inputs and m outputs, its final PTM and ITM will both be $2^n \times 2^m$ matrices.

Besides using the PTM method to analyze the behavior of a stochastic circuit in the presence of errors, we can employ gate-level circuit simulation to achieve the same goal. We inject the bit-flips into a gate via an XOR gate that flips the output signal z of C with probability p_e, resulting in a new erroneous signal z^*. For a circuit containing multiple gates, the error is injected into every gate.

Consider, for example, the stochastic realization of scaled addition. This operation can be implemented either by a majority circuit or a multiplexer [1], as shown in Figure 6a–b. The special input r is a constant scaling factor of value 0.5. The corresponding circuits with XOR gates added for error injection are shown in Figure 6c–d. To focus on the behavior of the computational hardware (the logic gates) in the presence of errors, we assume the data sources are not affected by errors.

Figure 7 presents error data obtained by PTM analysis and circuit simulation for the two basic gate types AND and NOT, as well as the scaled adder circuits of Figure 6. The error rates of all gates are assumed to be the same ($p_{e_1} = p_{e_2} = p_{e_3} = p_{e_4} = p_e$), but they are generated from independent random sources. We simulated the circuit with and without the added XOR gates to get the expected error-free values and the values affected by soft errors. The MSE is given by $E_{Z^*} = \mathbb{E}[(\hat{p}_{Z^*} - p_Z)^2]$. As Figure 7 shows, the analytical and simulation results are quite consistent.

We also constructed PTMs M_{MAJ} and M_{MUX} for the circuits of Figure 6a–b level by level from the PTMs of their component gates, including wiring gates, according to the method of [13]. The vertical lines on the figure separate the circuits into levels. In high-level symbolic form, we obtain the PTM expressions

$$\begin{aligned} M_{\text{MAJ}} &= \left(F_2 \otimes F_2 \otimes F_2\right) \times \left(I \otimes \text{swap} \otimes \text{swap} \otimes I\right) \\ &\quad \times \left(\text{AND2}_{p_e} \otimes \text{AND2}_{p_e} \otimes \text{AND2}_{p_e}\right) \times \left(\text{OR3}_{p_e}\right) \\ M_{\text{MUX}} &= \left(I \otimes F_2 \otimes I\right) \times \left(I \otimes \text{NOT}_{p_e} \otimes I \otimes I\right) \\ &\quad \times \left(\text{AND2}_{p_e} \otimes \text{AND2}_{p_e}\right) \times \left(\text{OR2}_{p_e}\right) \end{aligned}$$

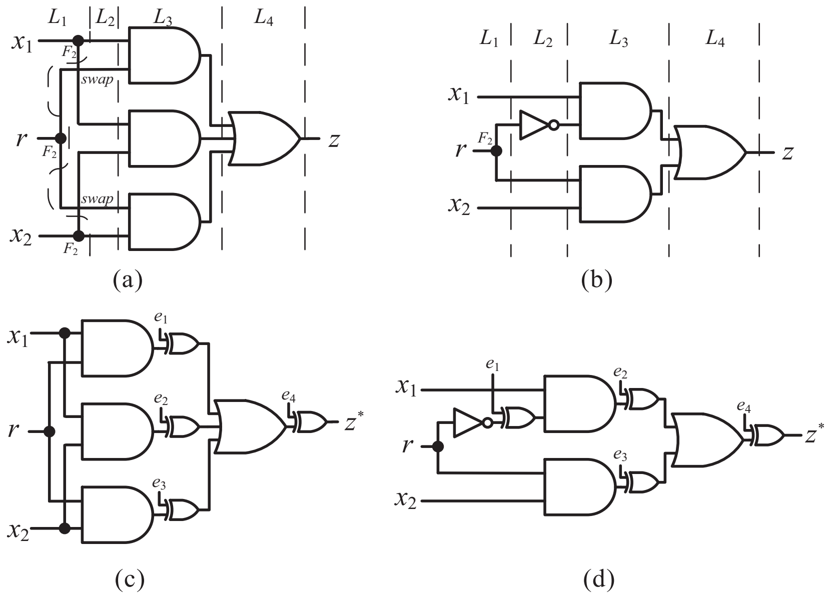


Figure 6: Stochastic circuits for the scaled addition $p_Z = 0.5(p_{X1} + p_{X2})$: (a) majority-based, (b) multiplexer-based, (c) majority-based with error injection, and (d) multiplexer-based with error injection; dashed vertical lines separate levels of the circuits.

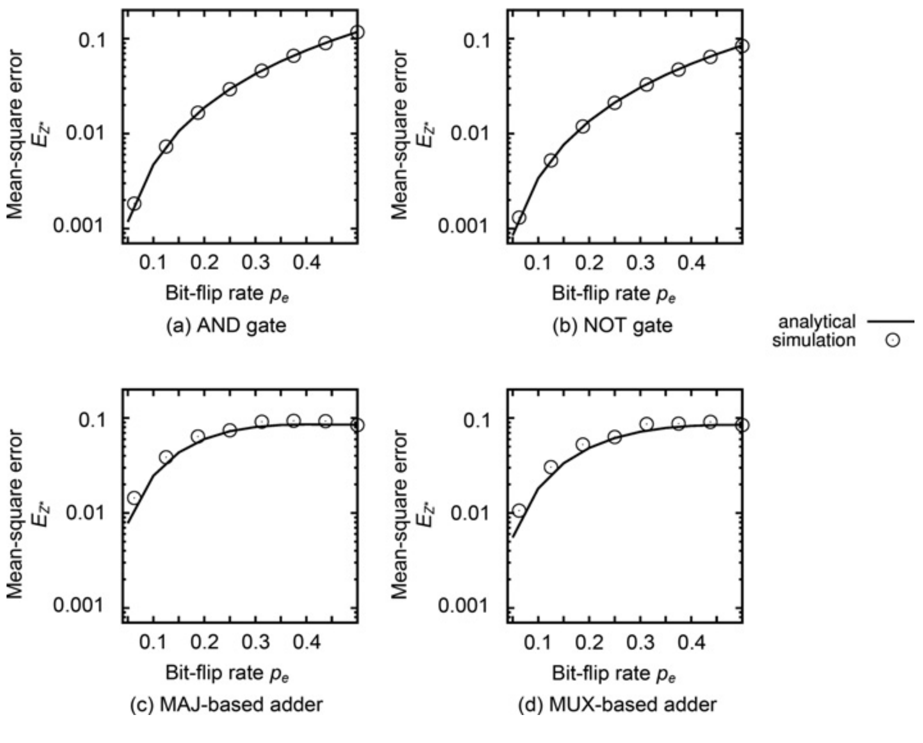


Figure 7: MSE at the outputs of representative stochastic circuits in the presence of soft-errors calculated using analytical and simulation methods.

Each parenthesized term in these equations corresponds to a circuit level, and the overall PTM of a circuit is obtained by multiplying the PTMs of all its levels. For example, the first level L_1 of the majority circuit consists of three F_2 gates in parallel, so $L_1 = (F_2 \otimes F_2 \otimes F_2)$. Similarly, we obtain $L_2 = (I \otimes \text{swap} \otimes \text{swap} \otimes I)$, $L_3 = (\text{AND2}_{p_e} \otimes \text{AND2}_{p_e} \otimes \text{AND2}_{p_e})$, and $L_4 = (\text{OR3}_{p_e})$. The PTM for the majority gate is then $M_{\text{MAJ}} = L_1 \times L_2 \times L_3 \times L_4$. We use the same method to construct the PTM for the multiplexer.

Fully expanded to binary form, M_{MAJ} and M_{MUX} become 8×2 matrices, which we derived from the above equations with the aid of GNU Octave [10]. The ITMs J_{MAJ}

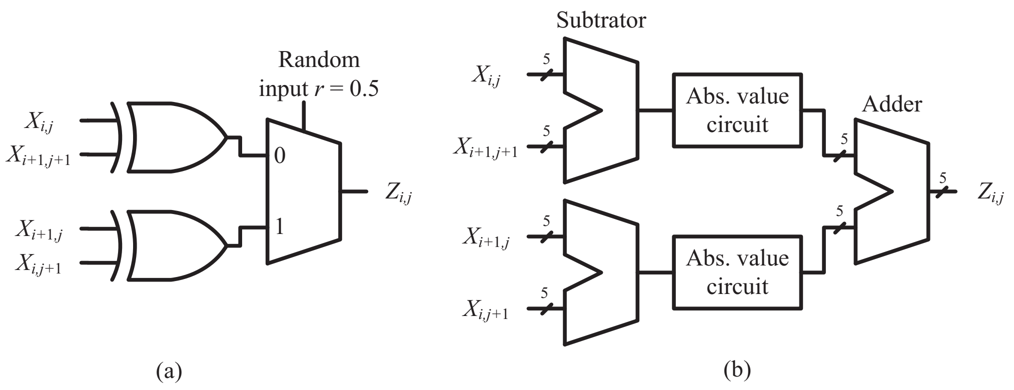


Figure 8: Edge detectors: (a) stochastic and (b) conventional.

and J_{MUX} for the two circuits, which have $p_e = 0$, take the form

$$J_{\mathrm{MAJ}} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}^T$$

$$J_{\mathrm{MUX}} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}^T$$

When errors are present, the 0–1 entries of J_{MAJ} and J_{MUX} must be replaced by complex polynomial expressions involving the variable p_e to obtain M_{MAJ} and M_{MUX} in expanded form.

Knowing both the erroneous PTMs and the ITMs, we can calculate the corresponding MSEs; see Figure 7. Again, the analytical results confirm the circuit simulations. In other words, both circuit simulation and PTM manipulation are valid methods for estimating soft-error effects in stochastic circuits. These results also show that when multiple errors are present, the errors accumulate. Hence, when the error rate is low, the multi-gate adders have worse MSE than single gates. When the error rate is high, for example, near 0.5, the behavior of all the circuits tends to appear random, so that they all have approximately the same MSE.

4 Case study: image edge detection

Edge detection is a fundamental operation in image processing and computer vision. Its goal is to identify significant local changes of intensity in digital images. Stochastic edge detectors have been shown to be significantly smaller, faster, more power-efficient, and more noise-tolerant than conventional ones in real-time image processing; see Figure 8 [3]. These designs, which are based on the Roberts cross algorithm [10], compute a moving average across a pixel window of size 2×2 for each pixel $p_{X_{i,j}}$ at row i and j of the image, and generate an output value $p_{Z_{i,j}}$ according to the formula

$$p_{Z_{i,j}} = 0.5\left(\left|p_{X_{i,j}} - p_{X_{i+1,j+1}}\right| + \left|p_{X_{i+1,j}} - p_{X_{i,j+1}}\right|\right) \tag{8}$$

The stochastic implementation takes advantage of certain correlation properties of stochastic numbers. An XOR gate $z = x \otimes y$ with uncorrelated (independent) inputs performs the function $p_Z = p_X(1 - p_Y) + (1 - p_X)p_Y$. However, if SNs X and Y are highly correlated with maximum overlap of 1 s, the XOR gate's function becomes $p_Z = |p_X - p_Y|$, which allows (8) to be realized by two XOR gates and a multiplexer as shown in Figure 8a. Assuming 5-bit precision, Figure 8b shows the corresponding binary design which contains several large arithmetic blocks, including addition, subtraction and absolute-value circuits. The stochastic edge detector is about two orders of magnitude smaller than the conventional binary design.

We now use PTMs to analyze the behavior of these circuits of under noisy conditions. The effect of a bit-flip rate of p_e on the output of every gate in the circuits is represented by a suitable PTM. Suppose the PTMs for the stochastic and conventional edge detectors are M_{sc} and M_{conv}, respectively. For each pixel and its 2×2 window,

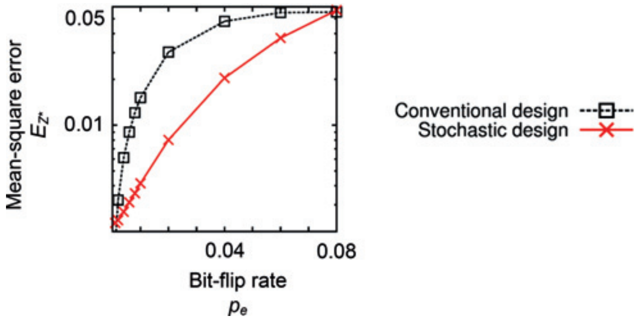


Figure 9: MSE of stochastic and conventional edge-detection circuits in the presence of soft-errors.

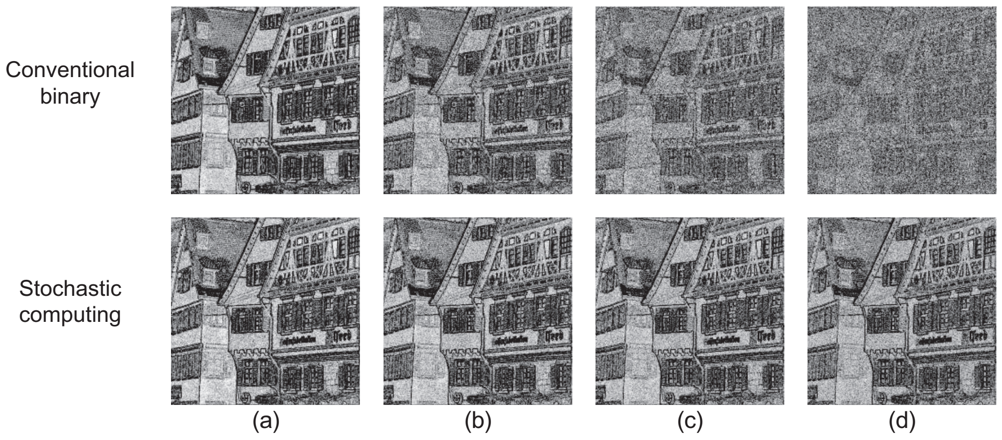


Figure 10: Comparison of stochastic and conventional edge detection for various soft-error rates (bit-flips percentages) in the edge-detection circuits: (a) 0.1%, (b) 0.5%, (c) 1% and (d) 2%.

we generate the corresponding input vectors M_{in} and M'_{in} for the stochastic and conventional edge detectors, respectively. The result of the edge-detection operation is then calculated as $M_{\text{in}} \times M_{\text{sc}}$ and $M'_{\text{in}} \times M_{\text{conv}}$. In this example, we assume that 5-bit precision is required, so the bit-stream length is $2^5 = 32$ for the stochastic design. We do not consider any additional circuits that might be needed for number conversion between the binary and stochastic formats.

Figure 9 compares the MSEs of the stochastic and conventional designs. As expected, when the error rate is low, the stochastic circuit is more affected by random fluctuation errors and performs worse than the conventional one. However, as the error rate increases, the MSE of the conventional design increases rapidly. When the error rate is very high, all the signal values become essentially random in both designs, so the MSEs coverage to the same value. Note that this result is consistent with the results shown in Figure 7.

Figure 10 compares the output image quality of the two edge detectors of Figure 8 in the presence of errors injected into them to simulate the impact of soft errors on the edge-detection hardware. It shows that when noise causes the output of the conventional circuit to become almost unrecognizable (at around $p_e = 2\%$), the stochastic circuit still produces acceptable results. Note that this experiment is different from the one shown in Figure 2, in which noise was only injected into the input image signals and the image-processing circuitry was assumed to be error-free. In this later experiment, noise is injected into the stochastic circuits themselves to demonstrate their fault-tolerant behavior. Together, these two experiments show that when the conventional design fails to produce recognizable results, stochastic computing can produce good results in the presence of severe noise that affects both the input image and the edge-detection circuit.

5 Conclusions

Stochastic circuits are finding major new applications because of their small size and error tolerance. We have presented a quantitative study of error tolerance that considers multiple error effects, and accounts for the inherent fluctuations in stochastic data as well as externally induced bit-flip errors affecting the data-processing circuits. We successfully used two complementary approaches: algebraic analysis with probabilistic transfer matrices (PTMs), and Monte Carlo circuit simulation. The algebraic analysis is more accurate than the simulation approach, but has the disadvantage of being infeasible for large circuits. Since stochastic circuits are very simple by nature, the algebraic approach is generally feasible for them, and is hence preferred. The simulation approach gives reasonably accurate results, and is feasible for circuits of any size. The experimental results we presented show that stochastic circuits have far better error tolerance than conventional binary circuits, especially at higher error rates.

Acknowledgement: This work was supported by Grant CCF-1318091 from the U.S. National Science Foundation.

Received December 16, 2013; accepted March 3, 2014.

References

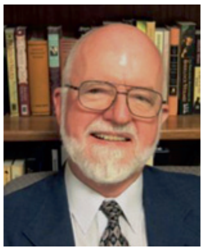
1. A. Alaghi and J. P. Hayes. A spectral transform approach to stochastic circuits. *Proc. Intl. Conf. Computer Design* (*ICCD*), pp. 315–321, Oct. 2012.
2. A. Alaghi and J. P. Hayes. Survey of stochastic computing. *ACM Trans. Embedded Computing Systems*, vol. 12, article 92 (19 pages), May 2013.
3. A. Alaghi, C. Li and J. P. Hayes. Stochastic circuits for real-time image-processing applications. *Proc. Design Automation Conf.* (*DAC*), article 136 (6 pages), June 2013.
4. R. C. Baumann, Radiation-induced soft errors in advanced semiconductor technologies.*IEEE Trans. Device & Materials Reliability*, vol. 5, pp. 305–316, 2005.
5. S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro*, vol. 25, pp. 10–16, 2005.
6. T.-H. Chen and J. P. Hayes. Design of stochastic Viterbi decoders for convolutional codes. *Proc. Design & Diagnostics of Electronic Circuits & Systems* (*DDECS*), pp. 66–71, 2013.
7. B. R. Gaines. Stochastic computing systems. *Advances in Information Systems Science*, vol. 2, pp. 37–172, 1969.
8. J. Gal-Edd and C. C. Fatig. L2-James Webb Space Telescope operationally friendly environment? *Proc. Aerospace Conf.*, pp. 105–110, 2004.
9. V. C. Gaudet and A. C. Rapley. Iterative decoding using stochastic computation. *Electron. Letters*, vol. 39, pp. 299–301, 2003.
10. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed., Prentice Hall, 2002.
11. GNU Octave. http://www.gnu.org/software/octave/.
12. I. Koren and C. Mani Krishna. *Fault-Tolerant Systems*, Morgan Kaufmann, 2007.
13. S. Krishnaswamy, G. F. Viamontes, I. L. Markov and J. P. Hayes. Probabilistic transfer matrices in symbolic reliability analysis of logic circuits. *ACM Trans. Design Autom. Electron. Sys.*, vol. 13, article 8 (35 pages), 2008.
14. P. Li and D. J. Lilja. Using stochastic computing to implement digital image processing algorithms. *Proc. Intl. Conf. Computer Design* (*ICCD*), pp. 154–161, 2011.
15. S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*, Prentice Hall, 2004.
16. A. Naderi, S. Mannor, M. Sawan and W. J. Gross. Delayed stochastic decoding of LDPC codes. *IEEE Trans. Signal Processing*, vol. 59, pp. 5617–5626, 2011.
17. B. Pratt et al. Fine-grain SEU mitigation for FPGAs using partial TMR. *IEEE Trans. Nuclear Science*, vol. 55, pp. 2274–2280, 2008.
18. W. Qian et al. An architecture for fault-tolerant computation with stochastic logic. *IEEE Trans. Computers*, vol. 60, pp. 93–105, 2011.
19. M. M. Shulaker et al. Carbon nanotube computer. *Nature*, vol. 501, pp. 526–530, 2013.
20. H. Stark and J. W. Woods. *Probability and Random Processes with Applications to Image Processing,* 3rd ed., Prentice Hall, 2002.
21. J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, Princeton Univ. Press, pp. 43–98. 1956.

Te-Hsuan Chen
2260 Hayward Street, University of Michigan, Ann Arbor, MI 48109, USA,
Tel.: +1 734 763-6411
tehsuan@eecs.umich.edu

Te-Hsuan Chen is a Ph. D. candidate in Computer Science and Engineering at the University of Michigan. He received his B. S. and M. S. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan. His research interests include computer-aided design, design and testing of VLSI circuits, error-correcting coding and unconventional computing methods.

Armin Alaghi
2260 Hayward Street, University of Michigan, Ann Arbor, MI 48109, USA,
Tel.: +1 734 763-6411
alaghi@eecs.umich.edu

Armin Alaghi received his B. Sc. degree in electrical engineering (2006) and M. Sc. degree in computer architecture (2009) from the University of Tehran, Iran. He is currently a Ph. D. candidate at the University of Michigan. His research interests include digital system design and testing, approximate computing, and embedded computing. His current research focuses on stochastic computing.



Prof. John P. Hayes
2260 Hayward Street, University of Michigan, Ann Arbor, MI 48109, USA,
Tel.: +1 734 763-0386,
Fax: +1 734 763-4617
jhayes@eecs.umich.edu

Prof. John P. Hayes received the B. E. degree from the National University of Ireland, Dublin, and the M. S. and Ph. D. degrees from the University of Illinois, Urbana-Champaign, all in electrical engineering. He is currently Professor of EECS and holder of the Claude E. Shannon Chair of Engineering Science at the University of Michigan. His teaching and research interests include computer-aided design, verification and testing of VLSI circuits, fault-tolerant computer architecture, and unconventional computing methods.