Trading Accuracy for Energy in Stochastic Circuit Design

ARMIN ALAGHI, University of Washington
WEI-TING J. CHAN, University of California, San Diego
JOHN P. HAYES, University of Michigan
ANDREW B. KAHNG and JIAJIA LI, University of California, San Diego

As we approach the limits of traditional Moore's-Law scaling, alternative computing techniques that consume energy more efficiently become attractive. Stochastic computing (SC), as a re-emerging computing technique, is a low-cost and error-tolerant alternative to conventional binary circuits in several important applications such as image processing and communications. SC allows a natural accuracy-energy tradeoff that has been exploited in the past. This article presents an accuracy-energy tradeoff technique for SC circuits that reduces their energy consumption with virtually no accuracy loss. To this end, we employ voltage or frequency scaling, which normally reduce energy consumption at the cost of timing errors. Then we show that due to their inherent error tolerance, SC circuits operate satisfactorily without significant accuracy loss even with aggressive scaling. This significantly improves their energy efficiency. In contrast, conventional binary circuits quickly fail as the supply voltage decreases. To find the most energy-efficient operating point of an SC circuit, we propose an error estimation method that allows us to quickly explore the circuit's design space. The error estimation method is based on Markov chain and least-squares regression. Furthermore, we investigate opportunities to optimize SC circuits under such aggressive scaling. We find that logical and physical design techniques can be combined to significantly expand the already-powerful accuracy-energy tradeoff possibilities of SC. In particular, we demonstrate that careful adjustment of path delays can lead to significant error reduction under voltage and frequency scaling. We perform buffer insertion and route detouring to achieve more balanced path delays. These techniques differ from conventional path-balancing techniques whose goal is to minimize power consumption by resizing the non-critical paths. The goal of our path-balancing approach is to increase error cancellation chances in voltage-/frequency-scaled SC circuits. Our circuit optimization comprehends the tradeoff between power overheads due to inserted buffers and wires versus the energy reduction from supply voltage downscaling enabled by more balanced path delays. Simulation results show that our optimized SC circuits can tolerate aggressive voltage scaling with no significant signal-to-noise ratio (SNR) degradation. In one example, a 40% supply voltage reduction (1V to 0.6V) on the SC circuit leads to 66% energy saving (20.7pJ to 6.9pJ) and makes it more efficient than its conventional binary counterpart. In the same example, a 100% frequency boosting (400ps to 200ps) of the optimized circuits leads to no significant SNR degradation. We also show that process variation and temperature variation have limited impact on optimized SC circuits. The error change is less than 5% when temperature changes by 100°C or process condition changes from worst case to best case.

This article is an extended and revised version of Alaghi et al. [2015].

This work was supported by Grant CCF-1318091 from the National Science Foundation, as well as by the IMPACT+ center, Samsung, Qualcomm, and NXP.

Authors' addresses: A. Alaghi, Department of Computer Science and Engineering, University of Washington, 185 Stevens Way, Seattle, WA, 98195-2350; email: armin@cs.washington.edu; W.-T. J. Chan and J. Li, Department of Electrical and Computer Engineering, University of California San Diego, 9500 Gilman Drive # 0407, La Jolla, CA, 92093-0407; emails: {wechan, jil150}@ucsd.edu; J. P. Hayes, Department of Electrical Engineering and Computer Science, University of Michigan, 2260 Hayward Street, Ann Arbor, MI, 48109-2121; email: jhayes@eecs.umich.edu; A. B. Kahng, Department of Computer Science and Engineering, University of California San Diego, 9500 Gilman Drive # 0404, La Jolla, CA, 92093-0404; email: abk@cs.ucsd.edu.

DOI: http://dx.doi.org/10.1145/2990503

CCS Concepts: • **Hardware → Physical design (EDA)**; **Fault tolerance**; **Process, voltage and temperature variations**; **Emerging architectures;**

Additional Key Words and Phrases: Stochastic computing, energy reduction, error modeling

ACM Reference Format:
Armin Alaghi, Wei-Ting J. Chan, John P. Hayes, Andrew B. Kahng, and Jiajia Li. 2016. Trading accuracy for energy in stochastic circuit design. J. Emerg. Technol. Comput. Syst. 13, 3, Article 47 (April 2017), 30 pages.
DOI: http://dx.doi.org/10.1145/2990503

1. INTRODUCTION

As we approach the limits of traditional Moore's-Law scaling, energy and power constraints pose major challenges for integrated-circuit (IC) designers. Many embedded systems, such as wearable devices and medical implants, have strict power and energy requirements due to battery capacity and physiological limitations [Lee et al. 2013]. For example, body tissue may be damaged by excessive power dissipation in a poorly designed implantable circuit [Chun et al. 2014]. Various approaches have been proposed to overcome such energy/power problems. Notably, embedded systems are usually designed for specific applications; this allows designers to use dedicated hardware with more desirable physical and/or logical characteristics than conventional designs.

Stochastic computing (SC) [Gaines 1969; Poppelbaum et al. 1967] has been proposed as an alternative low-power computing technique for several important embedded processing applications. SC circuits perform complex computations on (pseudo-)random bit-streams by means of simple logic gates. Figure 1 shows an SC circuit implementing the function $Z = \frac{1}{4} + \frac{1}{2}X_1X_2$. The number represented by each bit-stream is the probability of seeing a 1 in it. For example, the *stochastic numbers* (SNs) X_1, X_2, Z appearing at x_1, x_2, z represent $\frac{9}{12}, \frac{8}{12}, \frac{6}{12}$, respectively. The circuit has two primary inputs, x_1 and x_2, and two auxiliary inputs, r_1 and r_2. The auxiliary inputs are constant SNs of value $\frac{1}{2}$. The NAND gate of Figure 1 implements the stochastic function $Y_1 = 1 - X_1X_2$, which involves multiplication and subtraction. The OR gate implements $Y_2 = R_1 + R_2 - R_1R_2$, and since $R_1 = R_2 = \frac{1}{2}$, we have $Y_2 = \frac{3}{4}$. Finally, the XOR gate implements the function $Z = Y_1 + Y_2 - 2Y_1Y_2 = \frac{1}{4} + \frac{1}{2}X_1X_2$.

The main benefit of SC, as is evident from Figure 1, is that simple logic gates implement complicated arithmetic functions. For example, a single AND gate implements multiplication. Compared to a conventional binary multiplier, the SC multiplier is orders of magnitudes smaller in size. This, however, comes at the cost of speed. SC circuits need to operate on bit-streams that grow exponentially as the precision increases, and thus they take much more time to complete a computation. This has always been the main drawback of SC. The exponential loss in runtime not only hurts the performance but also leads to excessive energy consumption when long bit-streams are used [Moons and Verhelst 2014]. Consequently, SC circuits are mainly useful for low-precision computations [Aguiar and Khatri 2015]. Some recent work has focused on addressing this problem by reducing the runtime of SC circuits through the use of deterministic number sources [Alaghi and Hayes 2014] or by eliminating the power overhead of the clock distribution tree [Najafi et al. 2016].

This article exploits SC's error tolerance in order to reduce the energy consumption of SC circuits via voltage/frequency scaling. SC circuits are error tolerant, because a single error on one of the bits has minimal effect on the numerical value of a long bit-stream, and multiple errors tend to cancel each other out. Finally, SC circuits provide a natural energy-accuracy tradeoff: The bit-stream length N, that is, the number of clock cycles an SC circuit uses to perform a computation, directly affects its energy consumption and its accuracy.

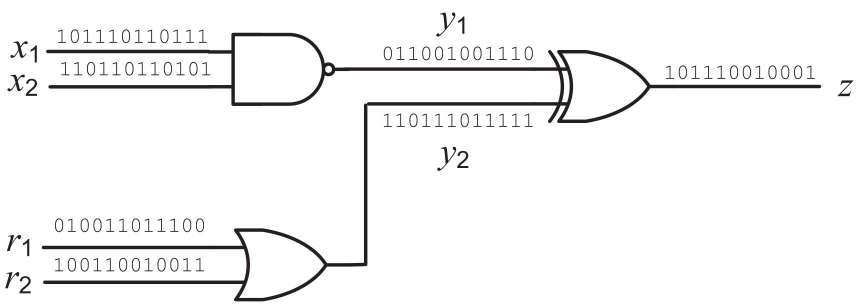

Fig. 1. Stochastic computing circuit implementing the function $Z = \frac{1}{4} + \frac{1}{2}X_1X_2$. The stochastic number represented by each bit-stream is the probability of seeing a 1 in a randomly chosen position.

SC, when first introduced in the 1960s, was attractive because it allowed simple implementation of arithmetic functions. However, it was dominated by conventional binary computing in the decades that followed, mainly because transistors became cheaper and performance became the primary design target. Throughout those decades, SC remained useful in certain applications, including efficient implementation of artificial neural networks [Kim and Shanblatt 1995; Brown and Card 2001; Canals et al. 2016].

After the turn of the century, SC regained attention because of its potential in low-power embedded processing applications. Some recent successful applications include low-density parity check (LDPC) decoding [Gaudet and Rapley 2003; Lee et al. 2015; Gross et al. 2005] and image processing [Li and Lilja 2011; Alaghi et al. 2013; Fick et al. 2014]. Other recent applications of SC include data recognition and mining [Chippa et al. 2014; Morro et al. 2015], machine learning [Gupta and Gopalakrishnan 2014], and dynamical systems [Wang et al. 2015]. Note that Lee et al. [2015] and Fick et al. [2014] have silicon-validated SC designs that outperform their conventional binary counterparts.

As mentioned, we investigate the application of low-power techniques such as voltage scaling to SC with the goal of obtaining circuits with ultra-low energy needs. Voltage scaling, that is, reducing the supply voltage of a circuit, reduces the circuit's energy consumption but increases its latency. If the application allows some latency overhead, then aggressive voltage scaling can be employed at the cost of occasional erroneous outputs. Thus, voltage scaling allows designers to trade accuracy for energy. This approach has been extensively studied in the non-SC literature [He et al. 2012; Hegde and Shanbhag 2001; Kahng et al. 2010], and methods of tolerating and/or correcting timing errors have been proposed. However, the probability of timing violations increases rapidly with voltage scaling, necessitating complicated error-correcting methods.

In this article, we show that representative SC circuits can tolerate up to 40% voltage reduction with no significant error. Figure 2 shows an example circuit and illustrates why, intuitively, we can aggressively scale the voltage/frequency of SC circuits. The idea is that we can apply new sets of inputs before the previous inputs have completely propagated through the circuit. In the ideal scenario (shown in Figure 2(c)), all the input signals propagate through different levels with the same speed; this scenario is very similar to the concept of wave pipelining [Burleson et al. 1998].

A major contribution of this work is an optimization method that improves the accuracy-energy tradeoff of SC circuits under voltage/frequency scaling. This is based on the observations of Figure 2. In order to achieve the ideal scenario shown in Figure 2(c), we need to modify the circuit to make sure signals propagate simultaneously. For this purpose, we employ synthesis and physical design techniques that keep the circuit's functionality intact, while effectively winning back any lost accuracy.

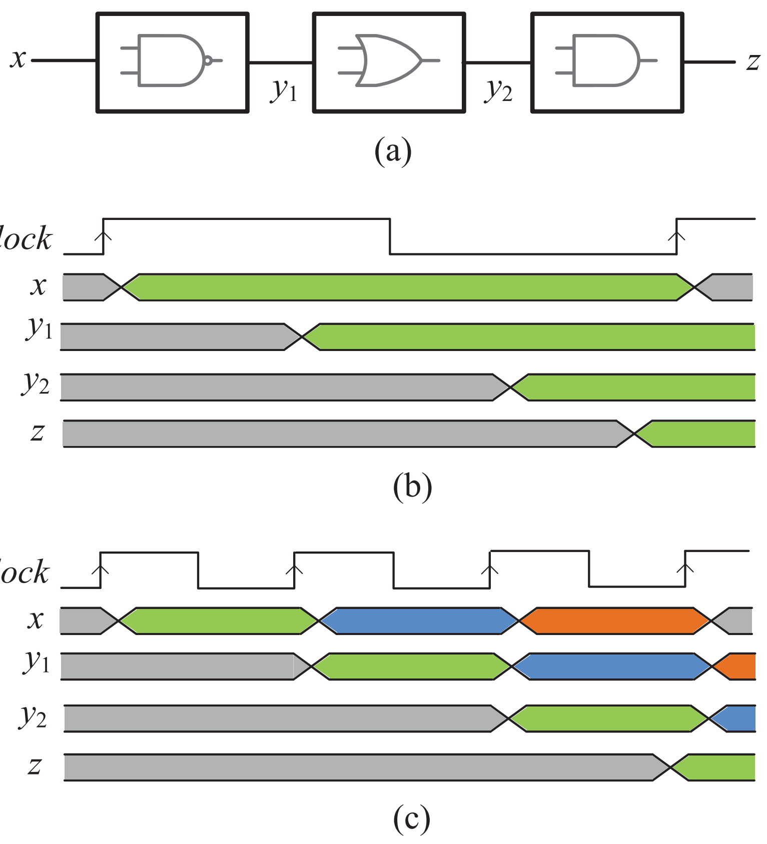

Fig. 2. Example showing how a representative SC circuit (a) operates under aggressive voltage/frequency scaling: (b) the normal mode of operation, in which the period of the clock is the same as the propagation delay of the circuit. In this case, the input remains unchanged until the propagation to the output is complete; (c) shows a voltage/frequency scaled scenario where the clock period is approximately 3 times smaller than the propagation delay of the circuit. As soon as the input propagates through the first level, a new input is applied. The circuit levels therefore operate like a pipeline.

In the synthesis step, we employ circuit structures that are naturally balanced. Note that this differs from logic-level balancing of circuits because we look at circuits that are stochastically equivalent [Chen and Hayes 2015], meaning that their underlying logic function may not be the same, even though they implement the same stochastic function. Existing logic-level tools do not understand such equivalences.

Ideally, a conventional place-and-route (P&R) flow balances path delays as much as possible (e.g., trading the slacks of non-critical paths for power and area reductions). However, due to design constraints (e.g., maximum transition or maximum capacitance) and the fact that gate sizes are limited, path delays are typically not perfectly balanced, especially when the paths have large differences in their depths. Figure 3 illustrates our proposed optimization. Assume that gate G_1 is already sized to its smallest size and is using high V_{th} (threshold voltage), but due to the difference in the depths of the paths, delays from $x_{\{1,2,3,4\}}$ to z and those from $x_{\{5,6\}}$ to z are still not completely balanced. Our study shows (in Section 4) that such unbalanced path delays increase the computation error of voltage/frequency-scaled SC circuits. To improve the computation accuracy of SC circuits, we propose to further balance path delays using buffer insertion and wire detouring. As indicated in Figure 3(b), we insert buffers (shown in red) into

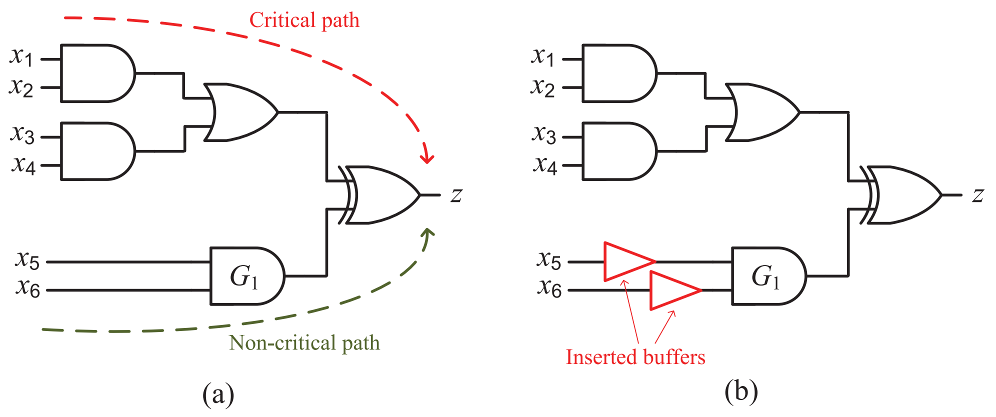

Fig. 3. (a) SC circuit implemented via conventional P&R tools, where the main objectives are area and power reduction. (b) SC circuit implemented via the proposed optimization flow, where path delays are balanced to improve accuracy. Inserted buffers are shown in red.

the non-critical path. Although the inserted buffers incur a power penalty, they enable more frequency and/or voltage scaling and hence reduce the latency and the energy consumption of the circuit for a given accuracy requirement. To our knowledge, this is the first time such techniques have been employed in the context of SC. In addition, to guide the design space exploration, we demonstrate an improved Markov chain model of computation errors in Section 3.3 based on the model from Alaghi et al. [2015]. We improve the modeling accuracy in Alaghi et al. [2015] by applying least-squares regression.

Note that the term "stochastic computing" has been also used recently to describe conventional circuits involving probabilistic behavior, including scenarios with voltage/frequency scaling [Sartori et al. 2011; Shanbhag et al. 2010]. What we refer to as SC is the computation technique that was proposed in the 1960s [Gaines 1969; Poppelbaum et al. 1967] and is unrelated to the concepts used in Sartori et al. [2011] and Shanbhag et al. [2010].

In this article, we focus only on combinational SC circuits. Qian and Riedel [2008] and Qian et al. [2011] show a connection between combinational SC circuits and Bernstein polynomials [Lorentz 1986] and prove that SC combinational circuits only implement certain types of polynomial functions. In this article, we use Reconfigurable Stochastic Computing (ReSC) [Qian et al. 2011] as the main method of implementing most of our testcases. As we will discuss later in the article, the rival design method [Alaghi and Hayes 2015] is not as energy efficient as ReSC.

We note that sequential SC circuits that implement a larger class of functions also exist in the literature [Li and Lilja 2011; Saraf et al. 2013; Brown and Card 2001]. Brown and Card [2001] are among the first to develop the theory of sequential SC circuits in the context of neural networks. In particular, they implement a sigmoid function by using a simple finite-state machine. However, addressing sequential circuits is beyond the scope of this article; we leave it as a subject for future work. The Sigmoid testcase used in this article is a combinational implementation and is unrelated to the circuit designed by Brown and Card [2001]. We also note that combinational circuits can implement non-polynomial functions by exploiting correlation [Alaghi and Hayes 2013] and that our optimization method is capable of handling them.

This article is organized as follows. Section 2 gives a brief review of SC, as well as an error analysis under voltage/frequency scaling conditions. It also illustrates the effect of stochastic number generation on the accuracy of SC circuits. Section 3 poses two

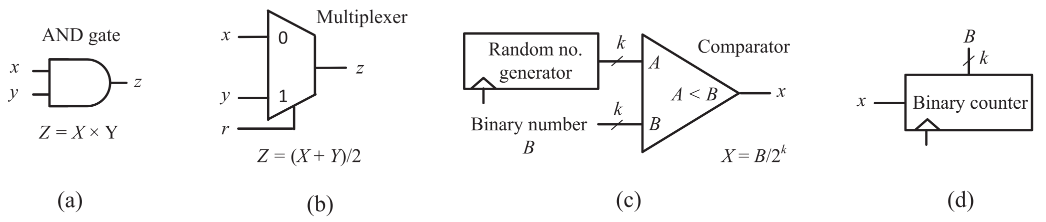

Fig. 4. Basic SC components: (a) multiplier, (b) scaled adder, (c) stochastic number generator, and (d) stochastic-to-binary converter.

related optimization problems, along with a straightforward way to find the minimum-energy operating point of an SC circuit for a desired accuracy level. It is also discusses a fast error estimation method for SC circuits. Section 4 examines the opportunities for optimizing SC circuits at different voltage levels. Section 5 presents experimental results, and, finally, conclusions are given in Section 6.

2. ERROR TOLERANCE AND NUMBER GENERATION IN STOCHASTIC COMPUTING

A stochastic circuit C is a logic circuit that operates on (pseudo-)random bit-streams, called *stochastic numbers* (SNs). Each wire x_i of C carries an SN X_i. The information conveyed by X_i, also conveniently denoted by X_i when no confusion is possible, is the rate or frequency of its 1-pulses and is independent of bit-stream length. Formally, a bit-stream of length N with N_1 1's and $N - N_1$ 0's is called an SN with value or magnitude $X_i = N_1/N$. This is usually interpreted as the probability of seeing a 1 in a randomly chosen position of the bit-stream [Gaines 1969]. SN values range over the unit interval $[0, 1]$, and their precision is determined by N.

Figure 4 shows several basic SC components. As mentioned earlier, a single AND-gate (shown in Figure 4(a)) implements SC multiplication. Figure 4(b) shows a multiplexer that implements SC addition. Since SNs are in the unit interval, the sum of two SNs falls into the interval $[0, 2]$ which cannot be represented by an SN. To mitigate this problem, a scaling factor of 1/2 is usually applied to bring the result back in the unit interval. Thus, the circuit of Figure 4(b) implements the scaled addition $Z = (X+Y)/2$.

When used along with conventional binary circuits, the inputs and outputs of stochastic circuits must go through a conversion process. Figure 4(c) shows a binary-to-stochastic converter that is usually referred to as a stochastic number generator (SNG). At each clock cycle, the SNG compares its k-bit binary input with a uniformly distributed random number. As a result, the probability of seeing a 1 at the output of the comparator becomes proportional to the binary input. Several studies have shown that the random number generator of Figure 4(c) can sometimes be replaced by simple counters [Alaghi and Hayes 2014]. As we will show later in this section, the choice of random number generators can impact the power and accuracy of an SC circuit. Converting SNs back to binary form can be done by counting the number of 1s, so the binary counter shown in Figure 4(d) suffices for this task.

2.1. Error Analysis

The inherent error tolerance of SC circuits stems from the fact that a single bit-flip in an SN of length N alters its magnitude by $1/N$, which is insignificant when N is sufficiently large. For example, the SN at the output of the circuit in Figure 1, where $N = 12$, represents $Z = \frac{6}{12}$. If one of the 1's or 0's of the bit-stream changes due to an error, then the erroneous SN is $Z^* = Z \pm \frac{1}{12}$, a minimal change. Furthermore, multiple errors tend to cancel each other out if they occur in opposite directions, since it is the number of 1's, but not their positions, that determines the magnitude of an SN [Chen et al. 2013; Qian 2011; Qian et al. 2011]. The probabilistic nature of SC circuits,

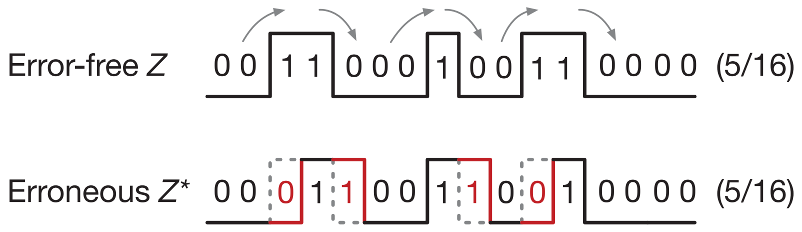

Fig. 5. Effect of delay errors on an SN.

along with the cancellation possibilities, makes it difficult to evaluate the accuracy of SC circuits.

Even though we will be dealing with the effect of timing errors in this article, it is worth mentioning how errors of the bit-flip type affect SNs. The mean square error (MSE) of an SN in the presence of bit-flips can be calculated by the following equation [Chen et al. 2013]. Assuming Z is the error-free SN and Z^* is the erroneous SN, we have

$$MSE = \mathbb{E}[(Z - Z^*)^2] = p_e^2 \cdot (1 - 2 \cdot Z)^2 + \frac{1}{N} \cdot \big(Z \cdot (1 - Z) + p_e \cdot (1 - p_e) \cdot (1 - 4 \cdot Z \cdot (1 - Z))\big), \quad (1)$$

where $\mathbb{E}[*]$ denotes the expected value (averaging) operator, p_e is the probability of getting a bit-flip on each bit of the SN, and N is the SN's length. Equation (1) reflects the effect of error cancellation: when $Z = 1/2$, the error becomes zero for large N. However, the cancellation does not help much when $Z \neq 1/2$. In the extreme case of $Z = 0$ (or $Z = 1$), the error is maximized because no cancellation occurs.

This work investigates the application of voltage and frequency scaling to SC circuits. Voltage scaling refers to the systematic reduction of the power supply voltage (i.e., "undervolting"), which is a standard technique used to reduce the power consumption of digital circuits. However, such scaling tends to produce timing violations that may cause output errors. Overly aggressive voltage scaling can induce many timing errors in conventional binary circuits and the resulting degradation of computational correctness can be catastrophic. By frequency scaling, we refer to the clocking of the circuit at higher than its nominal speed, at the cost of timing errors. It is possible to use design methods such as Razor [Ernst et al. 2003] to make conventional circuits more resilient to timing errors that are induced by frequency scaling. However, these techniques are only effective when the error rate is relatively low. SC circuits, on the other hand, have the potential to achieve graceful degradation of computational correctness when the voltage (or frequency) scaling is extremely aggressive and the timing error rate is relatively high. In addition, we may also be able to retrieve lost accuracy by employing the optimization method proposed in this work.

The types of errors that affect an SC circuit differ considerably from, bit-flips when voltage (or frequency) scaling is applied. In general, SC circuits tolerate errors of the bit-flip type, so one would expect them to tolerate scaling-induced timing errors as well. As we show next, SC circuits are much better at tolerating such timing errors.

Timing errors may occur in an SN Z when a transition from 0 to 1 is delayed, in which case the 1 will not be captured in time, and the magnitude of Z will be reduced by $1/N$, where N is the bit-stream length. Similarly, on a 1-to-0 transition, the 0 may be missed because of a timing error, and the magnitude of Z will increase by $1/N$. Since the numbers of 0-to-1 and 1-to-0 transitions are almost the same for any bit-stream (the difference is at most one), these timing errors tend to cancel each other out. Figure 5 shows an example of an SN affected by transition errors. In this figure, Z has three 0-to-1 and three 1-to-0 transitions denoted by arrows. Due to delay errors, some of the transitions are missed in the erroneous case Z^*, but the resulting number value remains the same due to error cancellation. In a very recent work, Najafi et al. [2016] show that SC circuits can tolerate timing variations caused by unsynchronized clocks. Their observation is in agreement with the results of our work. Similarly, Perez-Andrade et al. [2013] have shown that SC LDPC decoders can operate satisfactorily when clock-scaling-induced timing errors are present.

Now let us denote the 0-to-1 and 1-to-0 error rates by $e_{0\to1}$ and $e_{1\to0}$, respectively. We want to analyze their effect on an SN Z. We assume that the event of having a transition error at a certain clock cycle is an independent sample from a Bernoulli random variable with parameter $e_{0\to1}$ or $e_{1\to0}$, depending on the direction of the transition. This simplifying assumption is not a precise model, because, after all, the underlying circuit is deterministic. However, there are several phenomena that produce seemingly random behavior in voltage/frequency-scaled SC circuits. First, since (pseudo-)random bit-streams are used in most cases, signals in two consecutive clock cycles will be statistically independent. This implies that an input signal does not always activate the same circuit paths; the activated paths depend on the signals of the previous clock cycles, which are pseudo-random. Second, the output of an SC circuit is usually collected at a flip-flop of a counter, and, due to timing violations, the output flip-flop may become metastable and produce a seemingly random result. Third, when the supply voltage is reduced, the circuits become more susceptible to environment noise of a random nature. We do not consider the direct effect of each of these phenomena in this work; we model their collective effect by a Bernoulli random variable. We note that if deterministic bit-streams are used, as in our EdgeDetection testcase, the simplifying assumption of Bernoulli random variable fails to model the circuit behavior correctly.

For simplicity, we also assume that Z has equal numbers of 0-to-1 and 1-to-0 transitions. These numbers depend on two factors: the value of Z and its "activity." If $Z = 0$ or $Z = 1$, then the number of transitions will be zero. The maximum number of transitions usually occurs when $Z = 1/2$ if (pseudo-)random number sources are used in generating Z. But, as we will show in the next subsection, the choice of the random number source affects the number of transitions. We define the *activity factor* A as a number between 0 and 1 with the following properties. When $A = 1$ in Z, the number of transitions becomes maximum, and when it is 0, the number of transitions drops to the minimum. In the case of $Z = 1/2$, the maximum number of transitions will be N, where N is the length of the SN. This corresponds to an SN that transitions on every clock cycle, for example, 01010101010.... The SN $Z = 1/2$ with $A = 0$ corresponds to a bit-stream with the minimum number of transitions, for example, 00000001111....

We are now ready to calculate the effect of transition errors. Following the analysis in Chen et al. [2013], we observe that they cause output errors in two ways; they change the average value and the variance of Z^*. For the error-free number Z of length N and activity factor A we have

$$\mathbb{E}[Z^*] = \frac{1}{N} \cdot \mathbb{E}[Z \cdot N - N_{0\to1} \cdot e_{0\to1} + N_{1\to0} \cdot e_{1\to0}],$$

where $e_{i\to j}$ denotes the error rate on i-to-j transitions and $N_{i\to j}$ denotes the number of i-to-j transitions calculated by

$$N_{0\to1} = N_{1\to0} = 2 \cdot Z \cdot (1 - Z) \cdot A \cdot N.$$

Hence

$$\mathbb{E}[Z^*] = Z + 2 \cdot Z \cdot (1 - Z) \cdot \mathbb{E}[e_{1\to0} - e_{0\to1}]) \cdot A. \tag{2}$$

Equation (2) has two important implications. First, and more importantly, if $e_{0\to1} = e_{1\to0}$, then Z and Z^* will be equal on average. Second, the activity factor A has also a direct effect on the error. We will address impact of the activity factor in the next subsection. For now, let us assume that $e_{0\to 1} = e_{1\to 0} = e$ is correct. Even though the expected value of Z^* is the same as the error-free Z, we may still see random fluctuations due to the probabilistic nature of $e_{0\to 1}$ and $e_{1\to 0}$. Once again, if only deterministic signals are used, no random fluctuation will be seen. We compute the MSE using an approach similar to that in Chen et al. [2013], that is,

$$MSE = \mathbb{E}[(Z - Z^*)^2] = \frac{1}{N^2} \cdot \mathbb{E}\big[\big(Z \cdot N - (Z \cdot N - N^*_{0\to 1} + N^*_{1\to 0})\big)^2\big],$$

in which $N^*_{i\to j}$ is a random variable denoting the number of i-to-j transition errors. We now have

$$MSE = \frac{1}{N^2} \cdot \mathbb{E}\big[(N^*_{0\to 1} - N^*_{1\to 0})^2\big] = \frac{1}{N^2} \cdot \big(\mathbb{E}[(N^*_{0\to 1})^2] + \mathbb{E}[(N^*_{1\to 0})^2] - 2\mathbb{E}[N^*_{0\to 1} \cdot N^*_{1\to 0}]\big).$$

Since $N^*_{0\to 1}$ is a binomial random variable with parameters $N_{0\to 1}$ and e, we can evaluate the first term of the above equation by finding the second moment of $N^*_{0\to 1}$,

$$\mathbb{E}[(N^*_{0\to 1})^2] = e^2 \cdot N_{0\to 1} \cdot (N_{0\to 1} - 1) + e \cdot N_{0\to 1}$$

and, since $\mathbb{E}[(N^*_{0\to 1})^2] = \mathbb{E}[(N^*_{1\to 0})^2]$, we get

$$MSE = \frac{1}{N} \cdot \big(4 \cdot A \cdot Z \cdot (1 - Z) \cdot e \cdot (1 - e)\big). \tag{3}$$

Equation (3) has several important and counterintuitive implications:

—The errors due to voltage scaling can be reduced by increasing the number length N. While it is well known that increasing N reduces the random fluctuation errors in SC [Gaines 1969], Chen et al. have observed that when bit-flips are present, increasing N will not help [Chen et al. 2013].

—The activity factor A can also play an important role, since reducing A decreases the MSE.

—If the transition error rate $e = 1/2$, then the error is maximized. Obviously, if we set $e = 0$, we reduce the error to zero, but, somewhat surprisingly, if we increase the error rate to $e = 1$, then we also get $MSE = 0$. This is a scenario in which every transition of Z is erroneous, and since the numbers of transitions are equal, the errors all cancel each other out. Note that when $e = 1$, no random fluctuation is seen in the circuit, and the circuit behaves deterministically. This scenario is similar to the desirable behavior shown in Figure 2(c).

Among the three preceding implications, increasing N is the least desirable because it increases the runtime of the circuit and hence its energy consumption. The activity factor can be controlled by generating suitable SNs at the input (see Section 2.2). Balancing the transition errors $e_{0\to 1}$ and $e_{1\to 0}$ is the main target of this article, and this will be addressed in the following sections.

We emphasize that Equations (2) and (3) are based on several simplifying assumptions and only reflect the result of delay errors on a single signal Z. In reality, the assumptions may not hold due to circuit complications. For example, the assumption of having $e_{0\to 1} = e_{1\to 0}$, which is based on Equation (2), leads to desirable error reduction and cannot be easily achieved in real-world examples. We use Equation (2) as a guideline or ideal case that we want to approach. Similarly, Equation (3) sets guidelines for error reduction, some of which are not easily achieved. For instance, the $e = 1$ scenario where the final error becomes zero is never seen in our experiments. Also, $A = 0$ is another unachievable case that reduces the final error to zero. However, as we will show next, reducing A decreases the final error.

Low activity factor $A \simeq 0$	0000000011111111
Medium activity factor $A = 1/2$	0010011010111100
High activity factor $A = 1$	0101010101010101

Fig. 6. Three SNs with different activity factors representing $Z = 1/2$.

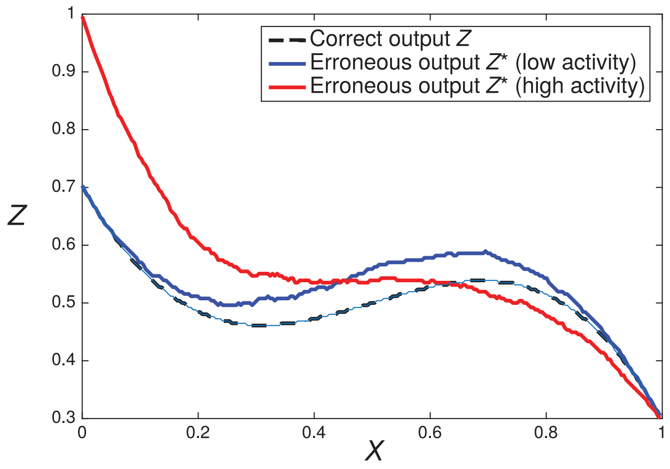

Fig. 7. The effect of SN generation on output errors; they are lower when a low-activity SNG is used.

2.2. Stochastic Number Generation

Stochastic number generation is an important step of a stochastic computation and directly affects parameters such as accuracy and area cost. It has been studied fairly extensively in the SC literature [Jeavons et al. 1994; Qian et al. 2009]. While most of the early work on SC assumed that SNs must be (pseudo-)random, several recent studies suggest that deterministic SNs can also be successfully employed in SC [Fick et al. 2014; Alaghi and Hayes 2014].

Equation (3) shows that the activity factor A of an SN directly affects its error. Interestingly, the activity factor also affects dynamic power consumption [Pedram 1994], so reducing A leads to both power and error reduction. While A cannot be completely fixed in a general SC circuit, it is possible to control it to some extent by a careful choice of stochastic number generators. Figure 6 shows three SNs with different activity factors representing 1/2. The most commonly used method of SN generation, that is, using (pseudo-)random number generators [Jeavons et al. 1994], yields numbers with medium activity factor. The SN generation method of Alaghi and Hayes [2014] yields SNs with high activity factors and hence is not suitable for the purposes of this work. The low activity factor SN shown in Figure 6 is generated by the method of Fick et al. [2014]. This number has only one transition, so it is very tolerant of transition errors and consumes very little dynamic power.

Figure 7 compares the impact of different SN generation methods on a voltage-overscaled SC circuit. The supply voltage of the circuit under test has been reduced from the nominal value of $V_{dd} = 1.0$V to $V_{dd} = 0.72$V. As a result, the output Z^* deviates from the correct output Z. As seen in the figure, the deviation is higher when a high-activity SNG is used. Based on this observation, we employ the SN generation method of Fick et al. [2014] to the extent allowed by the SC design. Note that SC circuits usually require independence (zero correlation) among their inputs, so in many cases it is not possible to generate all the inputs using the same method. In other words, we do not have complete control over the activity factors of all the signals.

3. PROBLEM DESCRIPTION AND PROPOSED SOLUTIONS

This section defines the two main problems that are addressed in this work. An error estimation method, which allows quick evaluation of SC circuits under voltage/frequency scaling, is also discussed.

3.1. Finding the Minimum Energy Point

First, we pose and answer the following question: "Given an SC circuit, what is the lowest energy required for computation with a given required accuracy (err_{goal})?" To quantify the accuracy of a circuit, several error metrics, such as maximum error, MSE, and so on, can be employed. From this point forward, we use the "average error" metric,

$$err = \mathbb{E}[|Z - Z^*|],$$

where Z is the correct or "golden" value of the circuit output and Z^* is the erroneous output. The difference between Z and Z^* is averaged over all possible inputs. This average-error metric will be used to measure the accuracy of both SC and conventional binary circuits. It is important to note that our choice of average error as the accuracy metric is because of its simplicity. The general approach that we propose below is not limited to a specific error metric. However, due to the probabilistic nature of SC circuits, all of the deduced error bounds will be probabilistic.

The length N of the SNs used in a stochastic computation controls the accuracy and the total energy consumed by the circuit. Thus, by decreasing N, one can trade away accuracy for energy or power savings. This natural tradeoff has been successfully used in the past [Alaghi et al. 2013]. Our work here shows that voltage/frequency scaling adds new dimensions to the accuracy-power tradeoff possibilities for SC circuits. In effect, SC circuits have three control knobs—(i) supply voltage V_{dd}, (ii) clock frequency f, and (iii) bit-stream length N (or, equivalently, clock cycle count)—that determine their accuracy and energy/power consumption. Finding the best operating point for a circuit is thus a new and challenging problem.

Previous methods search for the minimum N for which the average error is less than the given err_{goal}. However, as discussed, it is possible to adjust all three parameters (supply voltage V_{dd}, clock frequency f, and SN length N) concurrently in order to find the best solution. We will refer to the triplet (V_{dd}, f, N) as an *operating point* of an SC circuit. We now formalize the above question in the following problem statement.

Minimum-Energy Operating Point (MEOP) Problem. Given an SC circuit, find the operating point (V_{dd}, f, N) that has minimum energy consumption while satisfying the accuracy requirement of average error $\leq err_{goal}$.[1]

In addition to providing a solution to the MEOP problem, which we do in the next subsection, we also consider optimization to improve error behavior under voltage scaling conditions. Excessive supply-voltage downscaling and/or increase of the operating frequency can result in the misalignment of signal *actual arrival times* (AAT) at output z with respect to the clock capture phase. Without loss of generality, for any pair of timing paths from inputs x_i and x_j to output z in an SC circuit, we assume the corresponding arrival times at z are AAT_i and AAT_j, respectively, such that $k_i \cdot T \leq AAT_i \leq (k_i + 1) \cdot T$

[1]It is practically impossible to search the entire solution space given that f and V_{dd} are continuous and N is an arbitrary integer. Here "minimum energy" refers to the minimum energy point among a given set of (V_{dd}, f, N) combinations. Since varying N has been extensively studied in the literature, we only consider one choice of N in our search space. This means that the energy savings reported in this article are obtained only from voltage/frequency scaling.

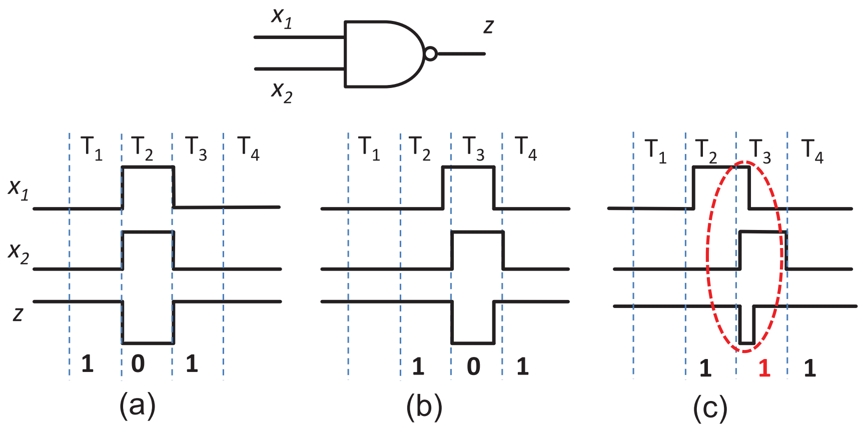

Fig. 8. Misalignment of arrival times at z with respect to the clock capture phase can lead to a computation error.

and $k_j \cdot T \leq AAT_j \leq (k_j + 1) \cdot T$, where T is the clock period. We say that these two timing paths exhibit *arrival time misalignment* if $k_i \neq k_j$. In other words, the two signals cannot be captured in the same clock cycle. We will show that the arrival time misalignment has significant impact on computation accuracy for SC circuits. Figure 8 shows one example of two timing paths (arcs $x_1 - z$, $x_2 - z$) to illustrate that arrival time alignment matters. In the example, Case (a) assumes no timing violation for both paths. This case generates the correct output sequence $1,\ 0,\ 1$. Case (b) has timing violations on both timing paths. However, the two arrival times are captured within the same clock cycle (i.e., T_2). Therefore, there is no arrival time misalignment. Although both signals are delayed by one cycle, the output sequence at z is still correct, that is, it is $1,\ 0,\ 1$.[2] In Case (c), due to unbalanced path delay, signals from x_1 and x_2 arrive at z in two different cycles. Thus, Case (c) has an arrival time misalignment that leads to a computation error, as shown by the red-dotted oval in Figure 8. Moreover, the output sequence cannot be recovered by adjusting the capture phase.

Motivated by the discussion above, we propose to employ logical and physical design techniques to align the arrival times at the output of an SC circuit. As observed in Equation (2), it is desirable to have equal error rates on 0-to-1 and 1-to-0 transitions ($e_{0\to1}$ and $e_{1\to0}$), because balanced errors reduce the average error. Accordingly, we define the following problem statement, whose solution is discussed in Section 4 below.

SC Circuit Optimization (SCOpt) Problem. Given a stochastic function and a range of supply voltages, find a circuit implementation that has minimum average error across the given supply voltage range.

3.2. Solution of the MEOP Problem

We now present our solution to the MEOP problem defined in the previous section. Briefly, given an SC circuit, we want to find the most energy-efficient operating point (V_{dd}, f, N) for a given accuracy metric. Our approach to this problem is a straightforward search within the operating-point space. In other words, we try different operating points and, for each, evaluate the accuracy and energy of the corresponding circuit.

[2]In Case (b), the output at the end of T_2 (derived from propagated signals in T_1 of Case (a)), can be incorrect. However, the corresponding impact on computational accuracy is negligible given that N is typically large, for example, $N = 4{,}096$.

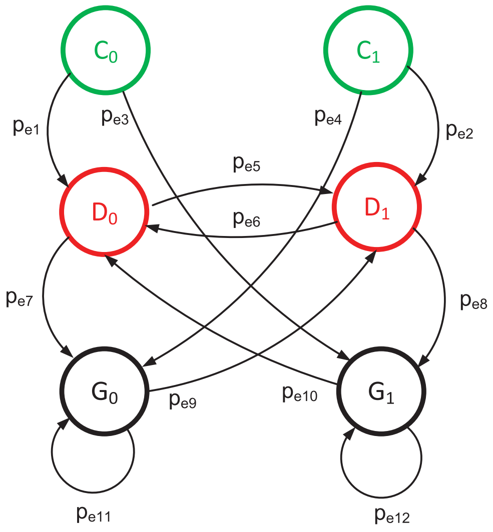

Fig. 9. Markov chain model for the proposed error estimation approach. The states are described in Table I.

We then choose the point that has the lowest energy while satisfying the accuracy requirements.

Unlike conventional binary circuits, errors in SC circuits tend to cancel each other out. In addition, SC circuits have a non-deterministic nature, that is, their behavior can be described by probabilities. For these reasons, evaluating the accuracy of SC circuits is *not* trivial. Exhaustive simulation can be used to evaluate the accuracy of small stochastic circuits. However, for larger circuits, it is impractical to perform exhaustive simulation for every operating point. With this in mind, we propose a method for fast error estimation. We note that our search strategy is not novel, but we are proposing a new model that enables fast exploration of the solution space.

3.3. Error Estimation Using a Markov Chain

We propose a Markov chain (MC) model [Grinstead and Snell 2003] to estimate errors. This model assumes that an SC circuit involving timing errors can be in *correct* or *incorrect* states. In a correct state, the circuit is producing the same output as the circuit with no timing errors. Since there are two possible output values, we have two correct states: C_0, in which the output is 0, and C_1, in which the output is 1 (Figure 9). In addition to the correct states, there are four incorrect ones. In an incorrect state, the SC circuit is producing an incorrect result due to a timing violation. Timing violations occur in two forms: (i) delay errors that appear when a 0-to-1 or 1-to-0 transition is missed at the output and (ii) glitches that appear when the output was not supposed to have a transition. We distinguish between these two error types and allocate different states to them. State D_i ($i \in \{0, 1\}$) is a state in which the output is the incorrect value i due to a delay error, and G_i is a state caused by a glitch in the output signal. Table I summarizes the MC model states.

The edges of the MC model indicate the transition probabilities between the states. For simplicity, we only show edges for the error cases and assume that the output magnitude is 0.5. In general, the magnitude of the output also affects the transition probabilities. Furthermore, there are implicit edges that are the complements of the shown edges and land on correct states. For instance, the implicit edge that goes from C_0 to C_1 is the complement of the edge that goes from C_0 to D_0 and so has transition

Table I. Description of Each State in the MC Model

Term	Meaning
C_0	Output is 0 and is correct
C_1	Output is 1 and is correct
D_0	Output is 0 and is incorrect due to a delay error
D_1	Output is 1 and is incorrect due to a delay error
G_0	Output is 0 and is incorrect due to a glitch
G_1	Output is 1 and is incorrect due to a glitch

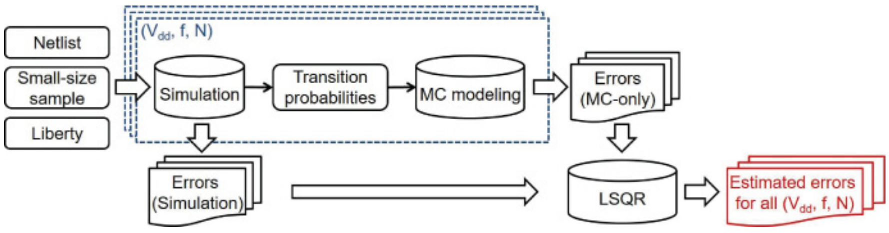

Fig. 10. Flow to construct MC model and estimate computation errors across various operating points.

probability $1 - p_{e1}$. As an example, let us assume that $p_{e1} = 0.1$. This means that if the circuit is in state C_0 and the next output is going to be 1, there is a 10% chance that the output transition is not captured due to a delay error, and hence the circuit lands in D_0 with probability p_{e1}. The other 90% of the time, the transition is successfully made and the circuit goes to the correct state C_1.

If the transition probabilities are known, then we can find the equilibrium probability, that is, the stationary distribution, of the MC and then we can evaluate the accuracy of the circuit in question. We do so by calculating the probability of seeing a 1 at the output of the circuit, that is, the probability of being in states C_1, D_1, or G_1, and comparing it with the correct output probability. Figure 10 illustrates our MC-based error estimation flow. We obtain the transition probabilities by generating a small input sample set (uniformly selected from the input space) and simulating the circuit. We perform logic simulation based on gate-level netlists. We then gather statistics for the transition rates between different states of the MC model. The size of the sample set determines the tradeoff between simulation runtime and the accuracy of the constructed MC model. We gradually increase the input sample size and find that in our testcases, the transition probabilities always converge when the input sample size is less than or equal to 20 (increasing the sample size to 21 does not lead to a significant change in the collected data). We therefore use 20 input samples in our experiments. Once the transition probabilities are estimated, we plug them into the MC model of Figure 9 and evaluate the accuracy of the circuit.

Our experimental results show that although the estimated errors from our MC model correlate well with the actual errors from simulation, they are typically pessimistic. We therefore apply a least-squares regression (LSQR) technique to improve the estimation accuracy. The LSQR step uses the final error values observed during the simulations. To further clarify, we simulate the circuits for 20 evenly distributed input samples from the big space of possible inputs. We collect two data sets from the simulations: (i) transition probabilities that are used to construct the MC model and (ii) final error values that are used to correct the MC model. Figure 11 shows an example where the LSQR technique improves the estimation accuracy. The MC model enables fast design space exploration by avoiding exhaustive simulation. Note that the MC model is constructed only once for each (design, operating point) combination.

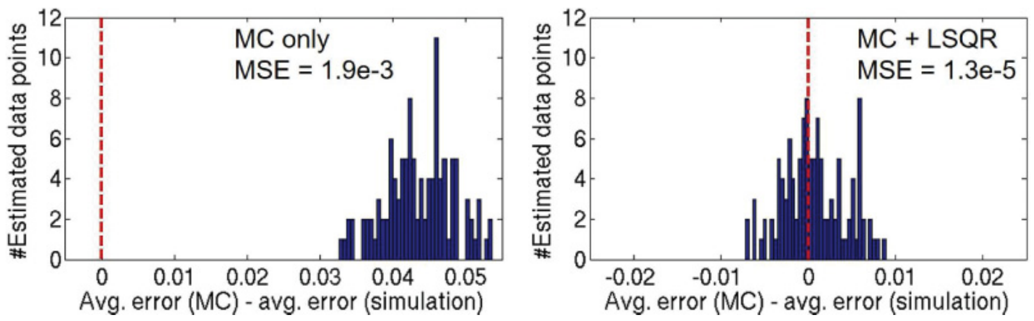

Fig. 11. Distribution of the estimation errors across different operating points. Left: Estimation from the MC model without LSQR (i.e., the method proposed earlier in Alaghi et al. [2015]) is pessimistic. Right: Application of the LSQR technique significantly improves the estimation accuracy.

We verify our modeling flow by comparing the average error values predicted by the MC model (i.e., MC + LSQR) and by post-layout simulation. We use six representative testcases throughout this article. These testcases are mostly from image processing and artificial neural network applications. A list of the testcases appears in Section 5. Although the MC model can be employed in all cases, it is mostly useful for the bigger testcases (GammaCorrection [Qian et al. 2011], Neuron [Brown and Card 2001], Sigmoid and BilateralFilter) since exhaustive simulation would be time consuming in these cases.

After logic synthesis, placement, and routing (SP&R), each testcase is simulated in *Cadence NC-Verilog* [Cadence 2011] with delays that are annotated from the SP&R flow results. To show the ability of the MC model to predict errors under aggressive voltage and frequency scaling, the circuit is signed off at $V_{dd} = 1.0$V, worst process corner, 125°C; it is then operated at lower voltages ($V_{dd} = \{0.6, 0.64, \ldots, 0.96\}$V) and boosted clock frequencies (e.g., up to 5× the signoff frequency).

Figure 12 shows that the predicted average errors are well correlated with the post-layout simulation results of the majority of the testcases. The MC model does not perform well for small testcases (EdgeDetection and PolySmall) mainly because they exhibit deterministic behavior (especially the EdgeDetection testcase). But, as noted, the MC model may not be useful in small testcases, since exhaustive simulation is feasible and fast.

The estimation error is relatively larger in the low-error cases (e.g., GammaCorrection) compared to the high-error cases (e.g., Neuron). The low-error cases happen when the transition errors $e_{0\to 1}$ and $e_{1\to 0}$ are either very small or very large. In such cases, the behavior of the circuit becomes mostly deterministic. For example, as discussed along with Equation (3), when $e_{0\to 1} = e_{1\to 0} = 1$, all the transitions acquire an error, yielding a zero error for the SN. Our MC model's main limitation is that it cannot model such deterministic scenarios.

Furthermore, the MC error estimation involves some discrepancies, as seen in Figure 12. We observed that in most of our testcases, a 25% margin is sufficient to guard against the discrepancies of the larger testcases. However, this makes the MC model pessimistic; we may discard acceptable operating points. As we discuss in Section 5, using the MC model does not always find the optimum operating point that is found via exhaustive simulation. Nevertheless, we believe that the MC model is still useful for the following scenarios: (i) estimating the error of a big circuit for which exhaustive simulation is impossible and (ii) exploring the huge space of operating points.

To further clarify the second scenario, consider an MEOP problem with $err_{goal} = 0.01$ on a relatively large circuit. Our approach is to search the space of possible

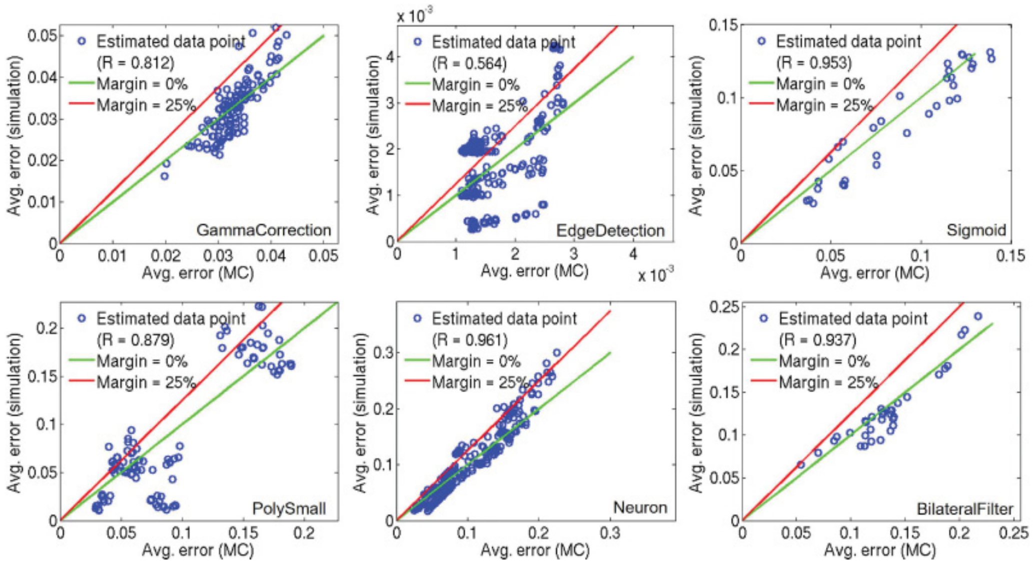

Fig. 12. Plots showing correlation between the errors estimated by our proposed MC model and the errors obtained from post-layout simulations. A 25% margin added to the MC estimated errors is sufficient to guard against small discrepancies for most of the testcases.

operating points and choose one that meets the err_{goal} with the least amount of energy consumption. For each operating point, we first run the MC model to quickly assess its error behavior. If the MC model shows a high error, say, 0.1, then we can safely dismiss the current operating point and move on to the next one. Otherwise, we perform exhaustive simulation on the operating point to precisely assess its error behavior. Thus, the MC model saves valuable simulation time for many operating points.

4. CIRCUIT-LEVEL OPTIMIZATION

The previous section dealt with a scenario in which an SC circuit is already implemented and we can only choose an operating point for it, that is, the MEOP problem defined in Section 2. In this section, we consider how to optimize the circuit using logic synthesis and physical design techniques to improve its energy efficiency (the SCOpt problem). We first discuss the timing behavior of SC circuits and highlight the main causes of errors, as well as the opportunities to eliminate them (Section 4.1). We then discuss the proposed optimization methods (Section 4.2).

4.1. Arrival Time Misalignment Matters

To examine the impact of arrival time misalignment on computation accuracy in an SC circuit, we insert and gradually increase the delays at the circuit's inputs; in the example shown in Figure 13, we sweep the delay from 0ps to 450ps, that is, $3\times$ the clock period, with a step size of 15ps. We record the change in the average computation error. We perform this experiment on two implementations of testcase PolySmall, where one implementation uses the conventional P&R flow and the other is optimized to have more balanced path delays. Figure 13(a) shows the path delay distribution of the two implementations. Note that the initial designs have the maximum path delay around 140ps. Therefore, the designs will have timing violations due to the inserted input delays.

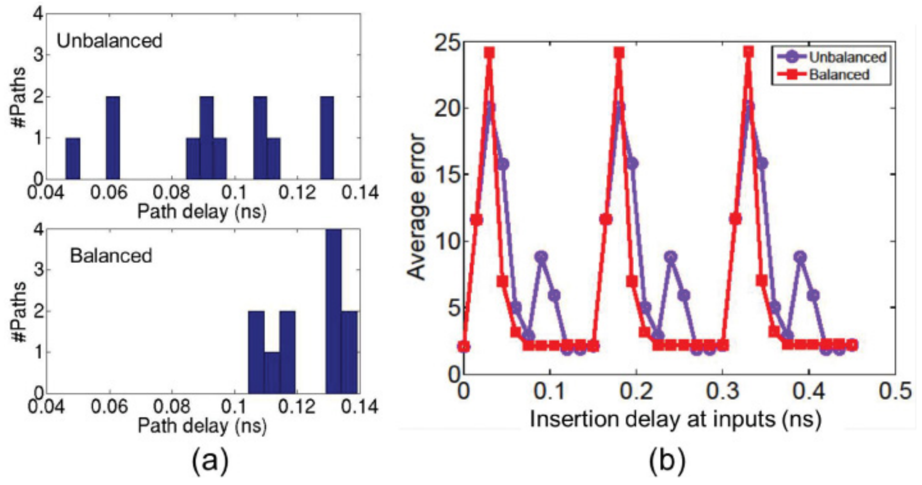

Fig. 13. Simulation results of the PolySmall testcase synthesized in 28nm FDSOI technology and clocked at 6.7GHz: (a) path delay distributions of two implementations and (b) computation error for different input delays.

The results in Figure 13(b) show that changing the input delay results in periodic fluctuation of computation accuracy, which indicates the impact of arrival time misalignment with respect to the capture phase. More specifically, when a large number of paths exhibit arrival time misalignment, for example, when the delay ranges between 15ps to 65ps for the balanced case, the corresponding computation error is large. On the other hand, when there is no arrival time misalignment, for example, when the delay ranges between 60ps to 150ps for the balanced case, although the design has larger timing violations, the computation error is small. Further, due to a wider range of path delays in the unbalanced case, the unbalanced implementation shows more data points with non-minimum average error (as seen in Figure 13(b)). Therefore, to reduce the likelihood of the misalignment of arrival times and to minimize the computation error, we propose buffer insertion and route detouring to minimize input-output path delay differences in SC circuits.

4.2. Optimization Methodologies

To resolve the arrival time misalignment issue and reduce the computation errors at a low supply voltage or with an overscaled frequency, we perform optimization during SC circuit implementation (i.e., SP&R) to balance the circuit's path delays.

First, we examine two major SC design styles: Spectral TRAnsform Use in Stochastic circuit Synthesis (STRAUSS) [Alaghi and Hayes 2015] and ReSC [Qian et al. 2011]. We then compare their path delays and computation errors for a given range of supply voltages. Figure 14 compares STRAUSS and ReSC for testcase PolySmall in 28nm fully depleted silicon on insulator (FDSOI) technology. We observe that the SC circuit implemented with ReSC tends to have more balanced path delays and smaller errors than that designed by STRAUSS. The ReSC architecture, which consists of an adder and a multiplexer, is very symmetric with respect to the primary inputs of the circuit. The STRAUSS-based circuits, on the other hand, have an asymmetric structure, which makes them smaller than the ReSC circuits but leads to unbalanced path delays and hence greater sensitivity to timing errors. We therefore only implement SC circuit designs based on ReSC in the experiments reported in Section 5.

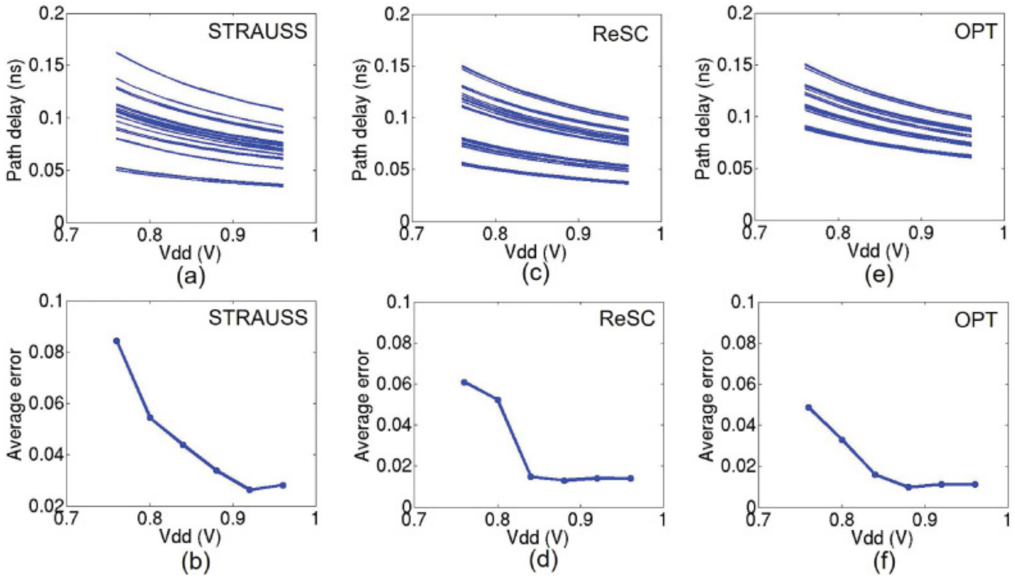

Fig. 14. Path delays (left) and average computation errors (right) at supply voltages ranging from 0.72V to 0.98V for testcase PolySmall at 28nm FDSOI. Each trace in (a), (c), and (e) denotes a timing path with a unique combination of rise/fall transitions. [(a) and (b)] STRAUSS [Alaghi and Hayes 2012], [(c) and (d)] ReSC [Qian et al. 2011], and [(e) and (f)] optimized circuit using our proposed MILP-based method.

To optimize the circuit, we perform buffer insertion and/or route detouring at the post-routing stage to balance path delays.[3] Various mathematical programming methods have been applied in the previous literature to guide the buffer insertion and wire sizing/route detouring for minimization of clock skew or data path delay [Chu and Wong 1999; Han et al. 2015]. Given that SC circuits typically have small sizes,[4] we formulate a Mixed Integer Linear Program (MILP) to search for the optimal solution based on a given set of buffering candidates.

We formulate our MILP as follows. The objective of the optimization is to minimize the normalized maximum delay difference (denoted by U) among timing paths of a design across a given range of supply voltages. Constraints are upper bounds on the maximum path delay and design leakage power. The notation used in the formulation is given in Table II.

$$\text{Minimize} \quad U, \tag{4}$$

$$\text{subject to} \quad D_i'^k = D_i^k + \sum_{1 \le i \le M, 1 \le j \le Q} c_{rj} \cdot d_j^k, \tag{5}$$

[3]We note that buffer insertion and route detouring techniques are not novel. They have been used to balance clock paths for skew minimization [Han et al. 2015] and to balance signal paths to prevent power side-channel attacks in smartcards [Tiri and Verbauwhede 2006]. However, we appear to be the first to apply such optimization techniques to balance path (datapath) delays in SC circuits in order to improve their accuracy.

[4]Typical image-processing SC circuits have only around 20 gates [Alaghi et al. 2013], while the largest known SC circuits have no more than 1,250 gate instances [Li and Lilja 2011].

Table II. Description of The Notation Used in the MILP

Term	Meaning
V_k	Supply voltage, ($1 \le k \le K$; V_K is the highest voltage)
P_i	Timing path, ($1 \le i \le M$)
D_i^k	Path delay of P_i at V_k
U	Upper bound on maximum normalized delay difference
G^k	Leakage power of the design at V_k
n_r	Wiring net ($1 \le r \le R$)
d_j^k	Delay increase due to buffer insertion and/or routing at V_k, ($1 \le j \le Q$)
g_j^k	Leakage power penalty of buffer insertion choice at V_k, ($1 \le j \le Q$)
c_{rj}	Indicator of buffer insertion and/or routing detour on n_r
α	Normalized upper bound on delay increase
β	Normalized upper bound on leakage power penalty

$$\sum_{1 \le j \le Q} c_{rj} \le 1, \qquad \forall\, 1 \le r \le R, \tag{6}$$

$$D_{max}^k = \max_{1 \le i \le M} D_i^k, \ \forall\, 1 \le k \le K, \tag{7}$$

$$\alpha \cdot D_{max}^k \ge D_i'^k, \qquad \forall\, 1 \le i \le M, \ 1 \le k \le K, \tag{8}$$

$$D_{max}'^k \ge D_i'^k, \qquad \forall\, 1 \le i \le M, \ 1 \le k \le J, \tag{9}$$

$$D_{min}'^k \le D_i'^k, \qquad \forall\, 1 \le i \le M, \ 1 \le k \le K, \tag{10}$$

$$U \ge \frac{D_{max}^K}{D_{max}^k} \cdot (D_{max}'^k - D_{min}'^k), \tag{11}$$

$$\beta \cdot G^k \ge \sum_{1 \le r \le R, 1 \le j \le Q} c_{rj} \cdot g_j^k, \ 1 \le k \le K, \tag{12}$$

where $D_i'^k$ is the optimized path delay of path P_i, with the buffer insertion and/or routing detour solution indicated by c_{rj}. $D_{max}'^k$ and $D_{min}'^k$ are, respectively, the maximum and minimum path delays at supply voltage V_k with buffer insertion and routing detour. U is the upper bound on the normalized path delay difference at all supply voltages. The MILP model minimizes U, thus minimizing the maximum normalized path delay difference at all supply voltages. In addition, G^k is the leakage power of the original design. The parameter g_j^k is the leakage power penalty of buffer insertion at supply voltage V_k. Our formulation constrains the optimization not to lead to more than α times the original maximum path delay and more than β times the original leakage power at each supply voltage.

Our initial studies attempted to include gate sizing and V_{th} swapping in the optimization process. However, this leads to a significant runtime and complexity increase in our MILP optimization, where each gate instance can have 6 to 22 candidate library cells for gate sizing and V_{th} swapping in the technology used. Furthermore, small sizing and/or V_{th}-swapping moves do not have a large impact on path delay and so are not

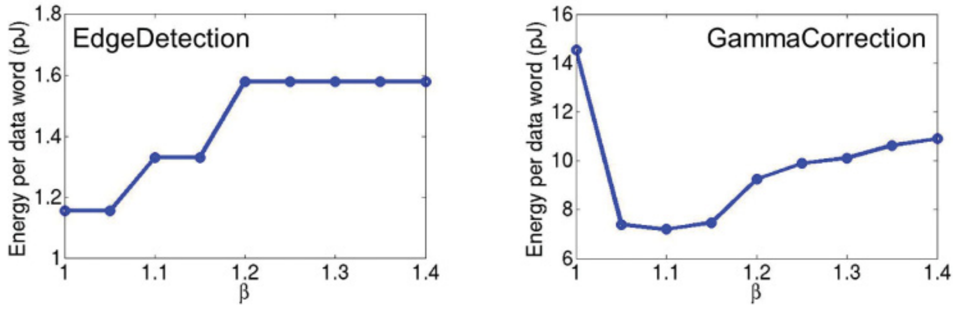

Fig. 15. Energy of designs optimized with different β values. The target average error rate is 0.02. Operating points are selected based on exhaustive simulation to minimize energy.

helpful in balancing path delays. On the other hand, large sizing and/or V_{th}-swapping moves might cause maximum-capacitance and maximum-transition violations, due to weak drive strength for downsizing and/or swapping to a higher V_{th}, or large input pin capacitance for upsizing. We therefore only apply buffer insertion and/or route detouring in our optimization.

Figures 14(e) and (f) show the resultant path delays and computation errors of the optimized SC circuit. They indicate significant improvement over both the unoptimized STRAUSS and ReSC implementations.

There is a tradeoff between power overhead due to inserted buffers and wire segments versus the energy benefits from improved accuracy with more balanced path delays (e.g., greater supply voltage downscaling is enabled by more balanced path delays). Our optimization reduces energy only when the power benefits from voltage downscaling outweigh the power overhead due to inserted buffers and wire segments. A small design with simple netlist structure might already have relatively balanced path delays and be sensitive to the power overhead of buffer insertion (i.e., the relative power overhead of buffer insertion is large), where the potential benefit from our optimization is small. On the other hand, power overhead due to buffer insertion is relatively small with respect to the total design power in a large design. Therefore, a large design is more likely to benefit from our optimization. Figure 15 shows the design energy optimized with different β values for a given accuracy requirement (i.e., $err_{goal} = 0.02$). We observe that for the small testcase EdgeDetection (with $\sim$5 instances), a larger β value always increases design energy.[5] On the other hand, for the relatively large testcase GammaCorrection (with $\sim$100 instances), the change of design energy with various β values shows a unimodal behavior due to the tradeoff between the power overhead of buffer insertion and the energy benefits from improved accuracy. These results support our intuition that large designs are more likely to benefit from optimization.

Figure 16 compares energy use across different accuracy requirements for testcase GammaCorrection, which is optimized with different β values. The black arrows in Figure 16 indicate the minimum achievable error for each optimized circuit. We observe that a larger β value leads to higher accuracy. However, due to the tradeoff between power overhead of buffer insertion and route detouring versus energy benefits from improved accuracy, a higher β value does not necessarily provide smaller design energy. We therefore sweep the value of β to explore such a tradeoff and to minimize design energy. This optimization procedure is illustrated in Algorithm 1, in which we

[5]Given that the EdgeDetection testcase only has about five instances, even insertion of the minimum-size buffer leads to relatively large power overhead.

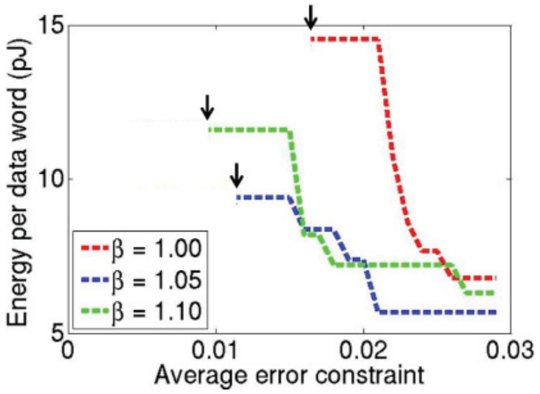

Fig. 16. Energy comparison of three GammaCorrection circuits optimized with different β values. The optimum operating point for a given accuracy requirement (err_{goal}) is selected via exhaustive simulation. Black arrows indicate the minimum achievable error of each circuit.

iteratively increase the value of β by δ (in our experiments $\delta = 0.05$) until there is no further energy reduction.

ALGORITHM 1: SC Circuit Optimization.

```
1: $\beta \leftarrow 1$; $Energy \leftarrow inf$; $is\_improved \leftarrow$ true
2: while $is\_improved$ do
3:    Solve MILP; Perform buffer insertion and/or route detouring as ECOs
4:    Perform exhaustive simulation to search for min-energy operating point for each $err_{goal}$
5:    Calculate average energy over all $err_{goal}$; Update $Energy$
6:    $\beta \leftarrow \beta + \delta$
7:    if $Energy$ reduces then
8:       $is\_improved \leftarrow$ true
9:    else
10:      $is\_improved \leftarrow$ false
11:   end if
12: end while
```

To evaluate the influence of process corners and temperature variation, we characterize standard cell libraries in different corner cases (worst corner and best corner) and temperatures (125°C and 25°C) using Synopsys SiliconSmart [Synopsys 2014]. We choose the same GammaCorrection testcase and simulate the circuits over different supply voltages. The result is shown in Figure 17. The differences among the corners are within 60% (normalized to the largest error at the same voltage) when the supply voltage decreases from 1.2V to 0.72V. This results in a maximum of 5% change in the output error.

Figure 18 further shows the path delays at different supply voltages for various corners. We observe that, due to smaller gate delays, the maximum path delay difference reduces in the best-corner cases, leading to smaller errors. In addition, lower temperature increases the maximum path delay difference (temperature inversion effect), especially at low supply voltages, which leads to larger errors as compared to the default (worst corner 125°C) case.

To ensure the feasibility of engineering change orders (ECOs), we characterize lookup tables (LUTs) based on buffer insertion and/or route detouring candidates with different input slew and load capacitance values. We then formulate our MILP and

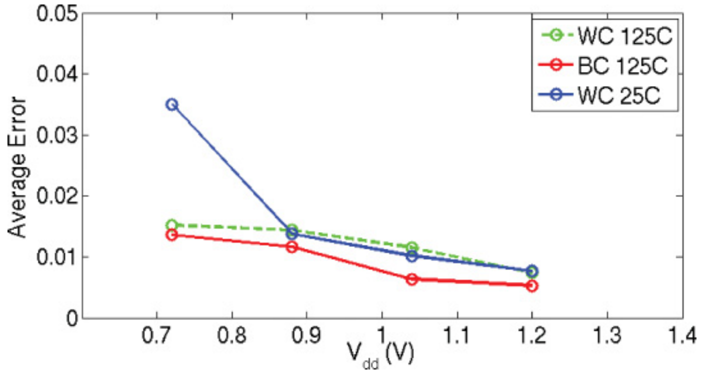

Fig. 17. Evaluation of the impact of process and temperature variations on the average error. The testcase GammaCorrection is optimized at 125°C and simulated at different corners and temperatures.

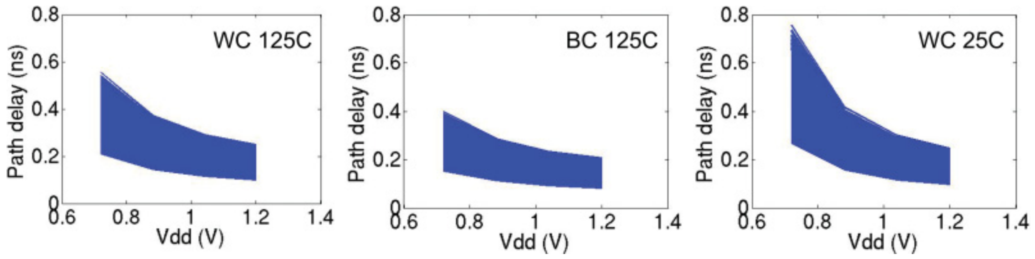

Fig. 18. Path delay for the four different corners in Figure 17.

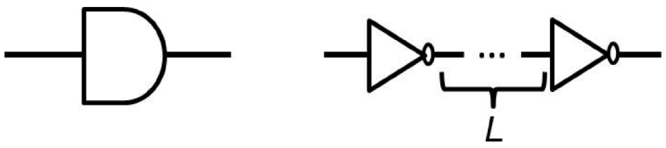

Fig. 19. Applied buffering styles: a single-stage non-inverting buffer, and an inverter pair with routing detour.

optimize circuits based on the characterized LUTs. The approach is similar to that of Han et al. [2015]. To balance path delays at a range of supply voltages and minimize the MILP runtime, we select buffer insertion and/or routing detour candidates that cover a wide range of delay-voltage tradeoffs but with a small set of choices. We study the delay-voltage tradeoffs with various gate types, gate sizes, threshold voltages, and wirelengths. We observe that the delay-voltage tradeoff is greatly affected by threshold voltage, gate size, and wirelength, which matches the observations made in Chan and Kahng [2012]. Therefore, we apply two buffering styles—a single-stage non-inverting buffer and an inverter pair with routing detour in between—as shown in Figure 19. Our approach selects from buffers and inverters of various sizes based on the delay requirements. We use both low V_{th} (LVT) and regular V_{th} (RVT) cells. The detoured wirelength, L, ranges from 10μm to 50μm with a step size of 10μm. Based on the LUTs, we further extend the buffering candidates with multiple cell stages (e.g., five stages of X100 buffers) to cover a wide range of delays. However, a large number of buffering candidates can significantly increase the runtime of a MILP. We therefore prune the candidates such that for a range of delay and delay-voltage tradeoffs, we uniformly divide the solution space into 4×4 sub-regions. We then select the buffering

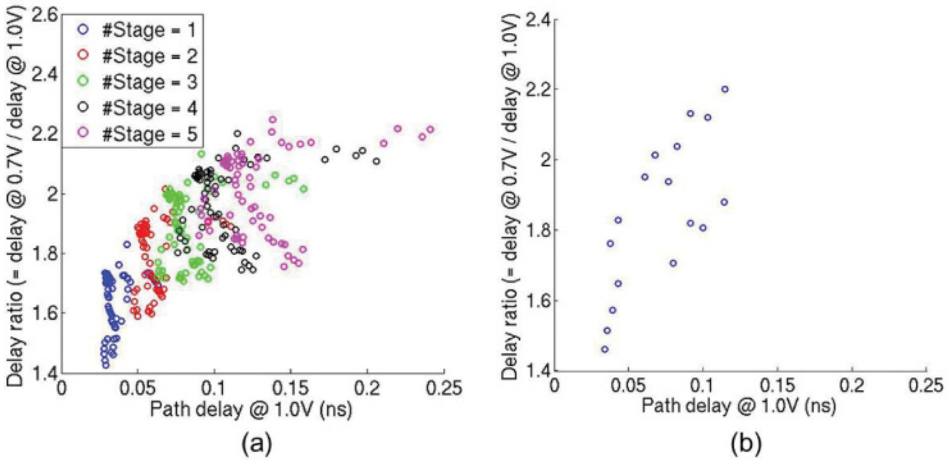

Fig. 20. (a) Buffering solution space, that is, delay range and delay-voltage tradeoff range, with multiple stages of buffers/inverter pairs. The circle colors denote different numbers of stages of buffers/inverter pairs. (b) Pruned buffering candidates.

solution with minimum leakage power from each sub-region. Figure 20(a) shows the solution space with up to five stages of buffering candidates. Figure 20(b) shows the pruned buffering candidates with delay ranges from 20ps to 120ps. Our experiments show that the pruning significantly reduces the runtime, while leading to negligible degradation in solution quality.[6]

Using the MILP solution, we perform buffer insertion and routing detour as ECO steps. Given that single-stage non-inverting buffer insertion is trivial, we use ECO commands from the P&R tools to perform buffer insertion and placement legalization. For insertion of an inverter pair with routing detour, we perform the ECO steps described in Algorithm 2. In the design flow, we start by inserting the first inverter. We then legalize the location of the inserted inverter so there is enough space for wire detour, for example, by moving the inverter away from the block boundary, and to ensure there is no overlap with previous routing detours. We then insert the second inverter such that the distance is 25 sites in the horizontal direction and two rows in the vertical direction with respect to the first inverter. Last, we perform routing detour with the single-width double-spacing (1W2S) routing rule on layers M3 and M4, between two inverters. An example of detoured routing is shown in Figure 21. Our current optimization method does not comprehend switching activity information. However, function-aware and input-pattern-aware optimization will be one of our future directions.

ALGORITHM 2: Insertion Flow of Inverter Pairs.

```
1: Place first inverter
2: Legalize the location of the first inverter
3: Insert second inverter such that its distance to the first inverter is 25 sites in horizontal
   direction and two rows in vertical direction
4: Perform routing detour with 1W2S
```

[6]For the largest design with ∼500 gate instances, the MILP runtime is less than 20s on a 24-core 2.5GHz Intel Xeon server.

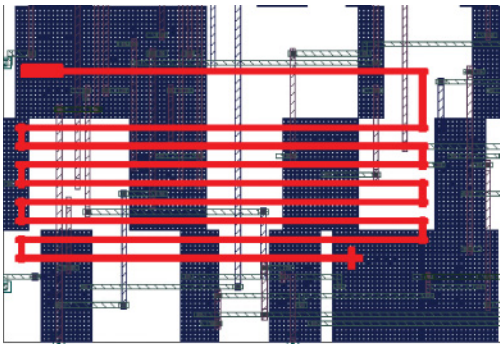

Fig. 21. Layout of routing detour (in red). The detoured wirelength is 40μm. Shaded blocks are standard cells.

Table III. Summary of Testcases

Testcase	# of cells	Description
GammaCorrection	~100	A common image processing task [Qian et al. 2011]
EdgeDetection	~5	A common image processing task [Alaghi et al. 2013]
PolySmall	~20	A simple polynomial of degree 3 implemented using methods of Alaghi and Hayes [2015] and Qian et al. [2011]
Neuron	~500	A 128-input neuron [Brown and Card 2001]
Sigmoid	~120	A common function used in artificial neural networks
BilateralFilter	~300	An edge-preserving smoothing filter; used in image processing

5. EXPERIMENTAL RESULTS

The experiments are implemented in 28nm FDSOI technology. We synthesize the testcases using *Synopsys Design Compiler vH2013.03-SP3* [Synopsys 2013a], and place and route them using *Synopsys IC Compiler vI-2013.12-SP1* [Synopsys 2013b]. We use *Synopsys PrimeTime vH-2013.06-SP2* [Synopsys 2013c] and *Synopsys PT-PX vH-2013.06-SP2* for timing and power analyses, respectively. We perform gate-level simulation using *Cadence NC-Verilog v8.2* [Cadence 2011]. We construct the Markov chain model using *MATLAB R2013a* [MathWorks 2013]. The MILP solver used in our optimization flow is *CPLEX v12.5* [IBM 2013]. Our testcases (see Table III) are representative circuits obtained from the SC literature and employed in typical applications such as image processing and neural network design. For input generation, we convert binary input vectors to pseudo-random bit-streams via SNGs.

5.1. Circuit Optimization Results

To evaluate the effectiveness of our optimization methods, we apply them to the testcases of Table III and compare the results with those of the unoptimized circuits. Figure 22, for example, shows how our optimization method changes the 0-to-1 and 1-to-0 error rates of the GammaCorrection testcase. It also shows that reducing the difference of the two error rates leads to output error reduction, even though the error rates are increased. For example, for the voltage range V_{dd} = 0.88–0.96V, we see that the optimized circuit has more 0-to-1 and 1-to-0 errors. The final results shown in Figure 23 also confirm the error reductions of the optimized circuits; that is, optimized circuits have less energy per data while we apply the same accuracy constraint. However, because the difference between the error rates is lower than that of the unoptimized circuit, we see a better output error behavior for the optimized circuit.

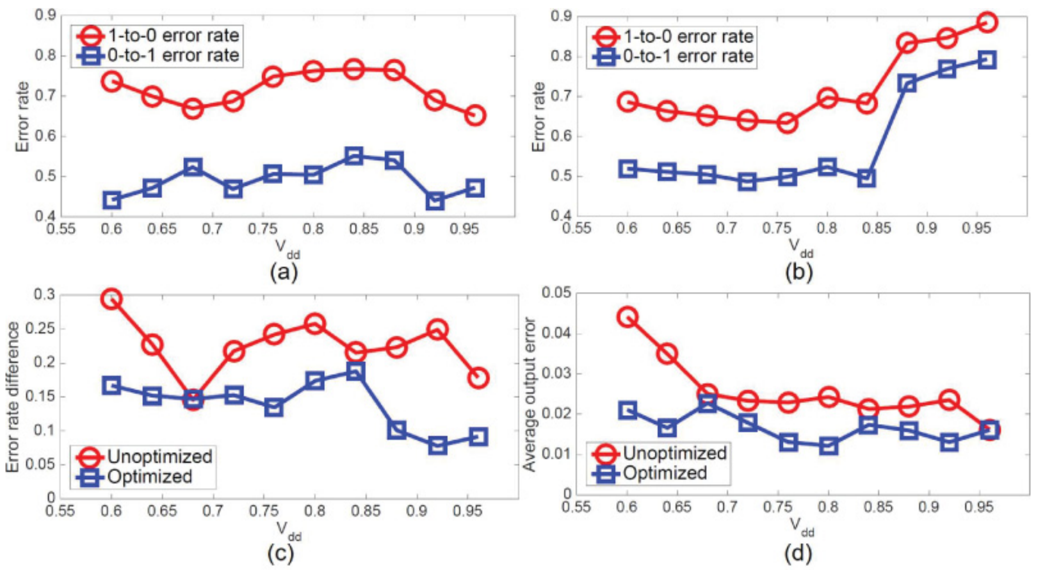

Fig. 22. Optimization of the GammaCorrection testcase: (a) 0-to-1 and 1-to-0 error rates of the unoptimized circuit, (b) 0-to-1 and 1-to-0 error rates of the optimized circuits, (c) error rate difference for both circuits (the optimized circuit has a lower error rate difference even though the error rates are higher for some cases), and (d) average output error for both circuits; the optimized circuit has a lower error.

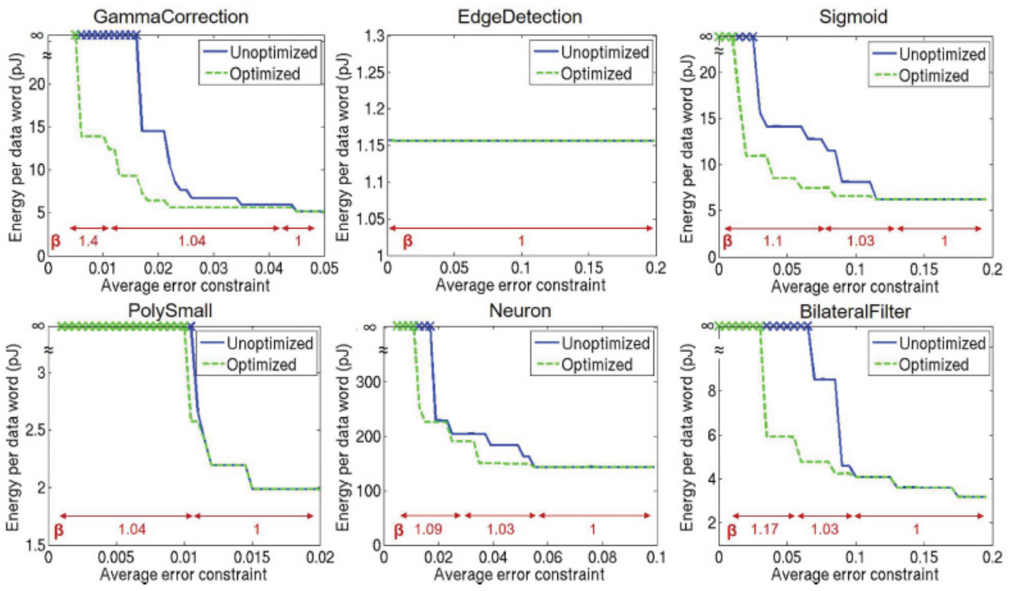

Fig. 23. Energy comparison for different accuracy requirements (err_{goal}) between unoptimized implementations (blue solid line) and optimized circuits (green dashed line). Operating points are selected based on exhaustive simulation. The V_{dd} range is 0.6V to 1.0V with a step size of 4mV. A cross sign ($\times$) indicates that no suitable operating point was found for the given err_{goal}. β values corresponding to the optimized circuits are shown in red font.

Figure 23 shows the minimum energy required for each design to meet a given average error constraint err_{goal} (within the space of possible operation points). A cross sign ($\times$) indicates that no suitable operating point was found for the given err_{goal}. Note that a circuit with a large number of inserted buffers (more balanced path delays) might have a small energy consumption when the error constraint is high because of voltage scaling, but it can have a large energy consumption when the error constraint is low due to the power penalty of the inserted buffers. Thus, it is difficult to find an optimized circuit that achieves minimum energy for all error constraints. To address this, we consider multiple optimized circuits, each optimized with a different β value. Note that we show different optimized circuits in Figure 23 to illustrate our optimization performance. Designers can choose their own accuracy requirements and use our method to find an optimized circuit tailored for their requirements. Furthermore, multiple accuracy-energy requirements are also supported in our optimization. A key observation here is that the optimized circuits are able to achieve lower err_{goal} values than the unoptimized circuits, especially for large testcases and/or tight error constraints.

In spite of the power overhead of added buffers and wires, the improved accuracy of the optimized circuits enables more aggressive voltage scaling yielding lower power. We observe that when the error constraints are tight, the optimized circuits can meet the constraints at a lower V_{dd}, leading to significant energy savings. For example, up to 49% energy reduction occurs in the GammaCorrection testcase with $err_{goal} = 0.02$. In addition, the results show that tighter error constraints require larger β values (i.e., more balanced path delays) for circuit optimization (especially in the large testcases). This indicates that SC circuits with more balanced path delays are able to achieve higher accuracy. On the other hand, when the error constraints are loose, or when the design is fairly balanced, for example, the EdgeDetection testcase, the unoptimized circuits (i.e., with $\beta = 1$) can also perform satisfactorily at low supply voltages. In such cases, especially for circuits with a small number of instances, where the relative power overhead of buffer insertion is large, optimizations with buffer insertion and route detouring are not efficient because the inserted buffers and wires increase the overall energy consumption. Therefore, our optimization flow (illustrated in Algorithm 1) chooses not to insert any buffer or wire for the small testcases EdgeDetection and PolySmall, hence the optimized circuits are the same as the unoptimized circuits.

5.2. Validation of MC Model

To gauge the effectiveness of our MC model, we perform an energy-accuracy comparison between the operating points selected by the MC-based flow and the ones selected via exhaustive simulation; see Figure 24. In this experiment, we only consider the unoptimized implementation of the testcases. The results show that our MC-based flow finds operating points that are similar in terms of energy to those selected via exhaustive simulations for most of the designs and error constraints. Moreover, Figure 24 shows that the MC-based flow reduces runtime significantly especially for large design, for example, ~50% for BilateralFilter. We observe that the energy penalty of using the MC-based flow is relatively high for small designs (e.g., PolySmall) where the computation errors are typically small and the MC-based estimation is less accurate.

5.3. Comparison with Conventional Circuits

This subsection performs a comparison between the optimized SC GammaCorrection and EdgeDetection and their binary counterparts. Figure 25 shows images generated by conventional binary and SC circuits at different supply voltage levels. We use both the signal-to-noise ratio (SNR) and multi-scale structure similarity (MS-SSIM) [Wang

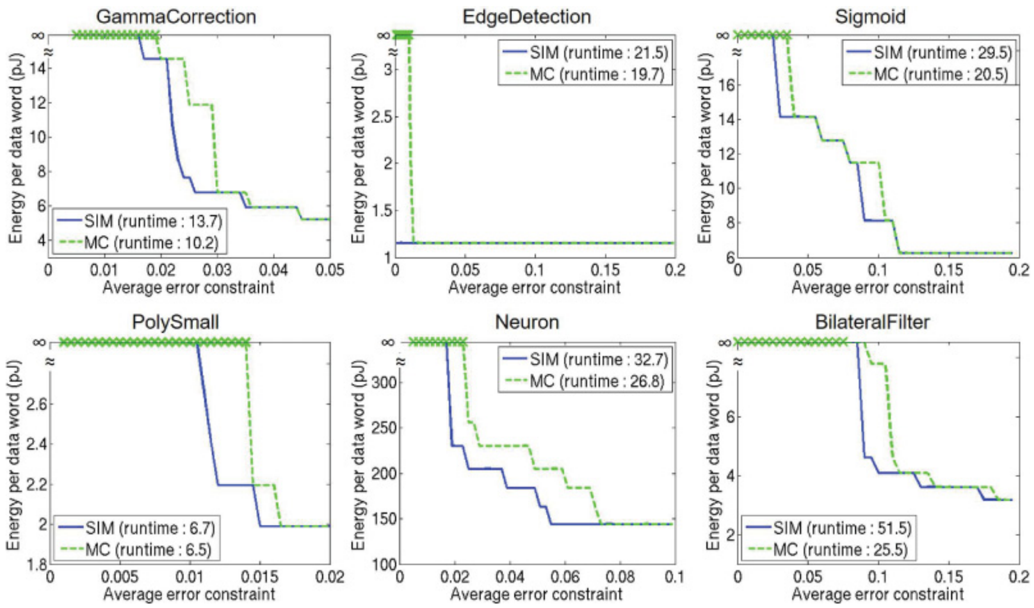

Fig. 24. Energy comparison between operating points from MC-based search (green dashed line) and exhaustive search (blue solid line) for different accuracy goals (err_{goal}). A cross sign ($\times$) indicates that no suitable operating point was found. The MC-based approach uses far fewer samples for circuit simulation, thus significantly reducing the simulation runtime. (Runtime unit: minutes.)

et al. 2003] error metrics to quantify the quality of the output images.[7] The results show that while the SC circuits tolerate aggressive voltage scaling, the binary circuits' output quality quickly drops, even with modest voltage changes.[8] This leads to significant energy savings in the SC case. Figure 25 shows that in the GammaCorrection testcase, the SC circuit consumes more energy than the conventional binary circuit at $V_{dd} = 1$V. However at $V_{dd} = 0.6$V, the SC circuit achieves the same accuracy as the conventional circuit at $V_{dd} = 1$V, making it more energy efficient (about 44% less energy). Similarly, in the EdgeDetection testcase, the SC circuit at $V_{dd} = 0.6$V achieves the same accuracy as the conventional circuit at $V_{dd} = 0.9$V, thus achieving 95% energy reduction. In Figure 25, we also report the area and the runtime of the circuits. As expected, the SC circuits have a lower area than their binary counterparts but have a longer runtime.

6. CONCLUSIONS

We present several optimization and modeling methodologies to exploit voltage/frequency scaling in SC circuits for reduced energy consumption at the cost of timing

[7]MS-SSIM evaluates the similarities of luminance, contrast, and structure components between the original image and the processed image. We use the MATLAB function from Chen et al. [2013] in our experiments to calculate MS-SSIM.

[8]SNR and MS-SSIM do not change monotonically with the supply voltage. There are several explanations for this. First, voltage reduction increases the rate of 0-to-1 and 1-to-0 errors ($e_{0\rightarrow1}$ and $e_{1\rightarrow0}$), but the changes are not necessarily linear, meaning that the difference $|e_{0\rightarrow1} - e_{1\rightarrow0}|$ may decrease due to voltage reduction, thus yielding a lower error. Second, based on Equation (3), the MSE decreases when e increases beyond 0.5. So a non-monotonic error behavior is not surprising. We also note that both GammaCorrection circuits (SC and binary) approximate the original gamma correction function (i.e., $Z = X^{0.45}$ [Qian et al. 2011]) and that the SC circuits have inherent random fluctuation errors. So even at a high supply voltage, some error exists. When timing errors are introduced via voltage reduction, they can cancel out the other errors and cause non-monotonic error behavior similar to that seen in Figure 25.

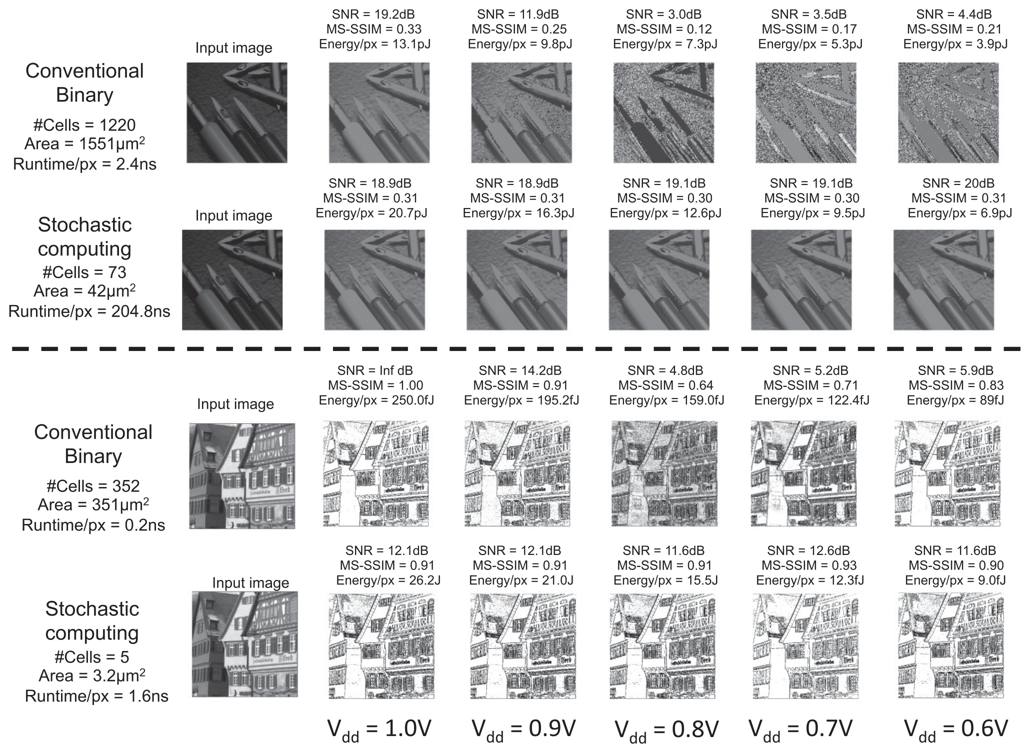

Fig. 25. Voltage scaling results of GammaCorrection (top, design obtained from Qian et al. [2011]) and EdgeDetection (bottom, design obtained from Alaghi et al. [2013]) executed by conventional and stochastic circuits, both implemented in 28nm FDSOI technology. After applying our proposed optimization techniques, the stochastic circuits show better tolerance against aggressive voltage scaling. Test images are from Kodak [1993] and Canon [2002].

errors. We also demonstrate that SC circuits are extremely tolerant of timing errors. Hence they can operate successfully under highly aggressive voltage/frequency scaling with very little loss of accuracy, unlike almost all conventional logic circuits. Based on these results, we define and solve the problem of finding the minimum-energy operating point of an SC circuit for a desired accuracy level. To enable rapid exploration of the operating-point space, we introduce a Markov chain-based technique for error estimation. We further observe that the accuracy of an SC circuit under scaled conditions can be improved by balancing its path delays. Accordingly, we perform optimizations during the logical and physical design phases to balance path delays. These methods have been successfully applied to several representative SC circuits, achieving substantial energy reduction without significant accuracy loss. To determine the robustness of our optimization approach, we also demonstrate that process and temperature variation have little impact on the error behavior of the optimized SC circuits, even under voltage scaling.

REFERENCES

A. M. Aguiar and S. P. Khatri. 2015. Exploring the viability of stochastic computing. In *Proc. ICCD*. 420–423.

A. Alaghi, W.-T. Chan, J. P. Hayes, A. B. Kahng, and J. Li. 2015. Optimizing stochastic circuits for accuracy-energy tradeoffs. In *Proc. ICCAD*. 178–185.

A. Alaghi and J. P. Hayes. 2012. A spectral transform approach to stochastic circuits. In *Proc. ICCD*. 315–321.

A. Alaghi and J. P. Hayes. 2013. Exploiting correlation in stochastic circuit design. In *Proc. ICCD*. 39–46.

A. Alaghi and J. P. Hayes. 2014. Fast and accurate computation using stochastic circuits. In *Proc. DATE*. 76:1–76:4.

A. Alaghi and J. P. Hayes. 2015. STRAUSS: Spectral transform use in stochastic circuit synthesis. *IEEE Trans. CAD* 34, 11 (2015), 1770–1783.

A. Alaghi, C. Li, and J. P. Hayes. 2013. Stochastic circuits for real-time image-processing applications. In *Proc. DAC*. 136:1–136:6.

B. D. Brown and H. C. Card. 2001. Stochastic neural computation. I. Computational elements. *IEEE Trans. Comput.* 50, 9 (2001), 891–905.

W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu. 1998. Wave-pipelining: A tutorial and research survey. *IEEE Trans. VLSI* 6, 3 (1998), 464–474.

Cadence. 2011. Cadence NC-Verilog User's Manual. Retrieved from http://www.cadence.com/.

V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselló. 2016. A new stochastic computing methodology for efficient neural network implementation. *IEEE Trans. Neur. Netw. Learn. Syst.* 27, 3 (2016), 551–564.

Canon. 2002. Canon Image Test Set. Retrieved from http://www.cipr.rpi.edu/resource/stills/canon.html.

T.-B. Chan and A. B. Kahng. 2012. Tunable sensors for process-aware voltage scaling. In *Proc. ICCAD*. 7–14.

M.-J. Chen, C.-C. Su, D.-K. Kwon, L. K. Cormack, and A. C. Bovik. 2013. Full-reference quality assessment of stereopairs accounting for rivalry. *Sign. Process.: Image Commun.* 28, 9 (2013), 1143–1155.

T. H. Chen, A. Alaghi, and J. P. Hayes. 2013. Behavior of stochastic circuits under severe error conditions. *Inform. Technol.* 56, 4 (2013), 182–191.

T. H. Chen and J. P. Hayes. 2015. Equivalence among stochastic logic circuits and its application. In *Proc. DAC*. 131:1–131:6.

V. K. Chippa, S. Venkataramani, K. Roy, and A. Raghunathan. 2014. StoRM: A stochastic recognition and mining processor. In *Proc. ISLPED*. 39–44.

C. C. N. Chu and D. F. Wong. 1999. A quadratic programming approach to simultaneous buffer insertion/sizing and wire sizing. *IEEE Trans. VLSI* 18, 6 (1999), 787–798.

H. Chun, Y. Yang, and T. Lehmann. 2014. Safety ensuring retinal prosthesis with precise charge balance and low power consumption. *IEEE Trans. Biomed. Circ. Syst.* 8, 1 (2014), 108–118.

D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, C. Ziesler, T. Pham, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. 2003. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proc. MICRO*. 7–18.

D. Fick, G. Kim, A. Wang, D. Blaauw, and D. Sylvester. 2014. Mixed-signal stochastic computation demonstrated in an image sensor with integrated 2d edge detection and noise filtering. In *Proc. CICC*. 1–4.

B. R. Gaines. 1969. Stochastic computing systems. In *Advances in Information Systems Science*, 2 (1969), 37–172.

V. C. Gaudet and A. C. Rapley. 2003. Iterative decoding using stochastic computation. *Electronics Letters* 39, 3 (2003), 299–301.

C. M. Grinstead and J. L. Snell. 2003. *Introduction to Probability* (2nd ed.). American Math. Soc. ISBN: 978-1886529236.

W. J. Gross, V. C. Gaudet, and A. Milner. 2005. Stochastic implementation of LDPC decoders. In *Proc. IEEE Asilomar Conf. Signals, Systems and Computers*. 713–717.

S. Gupta and K. Gopalakrishnan. 2014. Revisiting stochastic computation: Approximate estimation of machine learning kernels. In *Proc. Workshop on Approximate Computing Across the System Stack*. 1–2.

K. Han, A. B. Kahng, J. Lee, J. Li, and S. Nath. 2015. A global-local optimization framework for simultaneous multi-mode multi-corner skew variation reduction. In *Proc. DAC*. 26:1–26:6.

K. He, A. Gerstlauer, and M. Orshansky. 2012. Low-energy signal processing using circuit-level timing-error acceptance. In *Proc. Intl. Conf. on IC Design and Technology*. 1–4.

R. Hegde and N. R. Shanbhag. 2001. Soft digital signal processing. *IEEE Trans. VLSI* 9, 6 (2001), 813–823.

IBM. 2013. CPLEX Optimizer. Retrieved from http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

P. Jeavons, D. A. Cohen, and J. Shawe-Taylor. 1994. Generating binary sequences for stochastic computing. *IEEE Trans. Inform. Theory* 40, 3 (1994), 716–720.

A. B. Kahng, S. Kang, R. Kumar, and J. Sartori. 2010. Slack redistribution for graceful degradation under voltage overscaling. *Proc. ASP-DAC* (2010), 825–831.

Y. C. Kim and M. A. Shanblatt. 1995. Architecture and statistical model of a pulse-mode digital multilayer neural network. *IEEE Trans. Neur. Netw.* 6, 5 (1995), 1109–1118.

Kodak. 1993. Kodak Image Test Set. Retrieved from http://www.cipr.rpi.edu/resource/stills/kodak.html.

X. R. Lee, C. L. Chen, H. C. Chang, and C. Y. Lee. 2015. A 7.92 Gb/s 437.2 mW stochastic LDPC decoder chip for IEEE 802.15.3c applications. *IEEE Trans. Circ. Syst. I: Regul. Pap.* 62, 2 (2015), 507–516.

Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester, and D. Blaauw. 2013. A modular 1mm^3 die-stacked sensing platform with optical communication and multi-modal energy harvesting. *IEEE J. Solid-State Circ.* 48, 1 (2013), 229–243.

P. Li and D. J. Lilja. 2011. Using stochastic computing to implement digital image processing algorithms. In *Proc. ICCD*. 154–161.

G. G. Lorentz. 1986. *Bernstein Polynomials*. AMS Chelsea Publishing.

MathWorks. 2013. MATLAB and Statistics Toolbox Release. 2012b. MathWorks, Inc., Natick, Massachusetts.

B. Moons and M. Verhelst. 2014. Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies. In *IEEE J. Emerg. Select. Top. Circ.Syst*. 475–486.

A. Morro, V. Canals, A. Oliver, M. L. Alomar, and J. L. Rossello. 2015. Ultra-fast data-mining hardware architecture based on stochastic computing. In *PLoS ONE*, Vol. 10.

M. H. Najafi, D. J. Lilja, M. Riedel, and K. Bazargan. 2016. Polysynchronous stochastic circuits. In *Proc. ASP-DAC*. 492–498.

M. Pedram. 1994. Power estimation and optimization at the logic level. *J. High Speed Electron. Syst.* 5, 2 (1994), 179–202.

I. Perez-Andrade, X. Zuo, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo. 2013. Analysis of voltage- and clock-scaling-induced timing errors in stochastic LDPC decoders. In *Proc. IEEE Wireless Communications and Networking Conf.* 4293–4298.

W. J. Poppelbaum, C. Afuso, and J. W. Esch. 1967. Stochastic computing elements and systems. In *Proc. AFIPS Fall Joint Computer Conf.* 635–644.

W. Qian. 2011. *Digital yet Deliberately Random: Synthesizing Logical Computation on Stochastic Bit Streams*. Ph.D. Dissertation. University of Minnesota.

W. Qian and M. D. Riedel. 2008. The synthesis of robust polynomial arithmetic with stochastic logic. In *Proc. DAC*. 648–653.

W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. 2009. The synthesis of combinational logic to generate probabilities. In *Proc. ICCAD*. 367–374.

W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. 2011. An architecture for fault-tolerant computation with stochastic logic. *IEEE Trans. Comput.* 60, 1 (2011), 93–105.

N. Saraf, K. Bazargan, D. J. Lilja, and M. D. Riedel. 2013. Stochastic functions using sequential logic. In *Proc. ICCD*. 507–510.

J. Sartori, J. Sloan, and R. Kumar. 2011. Stochastic computing: Embracing errors in architecture and design of processors and applications. In *Proc. Intl. Conf. Compilers, Architectures and Synthesis for Embedded Systems*. 135–144.

N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones. 2010. Stochastic computation. In *Proc. DAC*. 859–864.

Synopsys. 2013a. Synopsys Design Compiler User's Manual. Retrieved from http://www.synopsys.com/.

Synopsys. 2013b. Synopsys IC Compiler User Guide. Retrieved from http://www.synopsys.com/.

Synopsys. 2013c. Synopsys PrimeTime User's Manual. Retrieved from http://www.synopsys.com/.

Synopsys. 2014. Synopsys SiliconSmart User's Manual. Retrieved from http://www.synopsys.com/.

K. Tiri and I. Verbauwhede. 2006. A digital design flow for secure integrated circuits. *IEEE Trans. CAD* 25, 7 (2006), 1197–1208.

Z. Wang, N. Saraf, K. Bazargan, and A. Scheel. 2015. Randomness meets feedback: Stochastic implementation of logistic map dynamical system. In *Proc. DAC*. 132:1–132:7.

Z. Wang, E. P. Simoncelli, and A. C. Bovik. 2003. Multi-scale structural similarity for image quality assessment. In *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*. 9–12.

Received December 2015; revised June 2016; accepted August 2016