Optimizing Stochastic Circuits for Accuracy-Energy Tradeoffs

Armin Alaghi[†], Wei-Ting J. Chan[‡], John P. Hayes[†], Andrew B. Kahng[‡+] and Jiajia Li[‡]
UC San Diego, [‡]ECE and [+]CSE Depts., La Jolla, CA 92093, {wechan, abk, jil150}@ucsd.edu
[†]University of Michigan, EECS Dept., Ann Arbor, MI 48109, {alaghi, jhayes}@eecs.umich.edu

***Abstract*—Stochastic computing (SC) acts on data encoded by bit-streams, and is an attractive, low-cost and error-tolerant alternative to conventional binary circuits in some important applications such as image processing and communications. We study the use of energy reduction techniques such as voltage or frequency scaling in SC circuits. We show that due to their inherent error-tolerance, SC circuits operate satisfactorily without significant accuracy loss even with aggressive scaling that improves their energy efficiency by orders of magnitude. To find the minimum-energy operating point of an SC circuit, we propose a Markov chain model that allows us to quickly explore the space of operating points. We also investigate opportunities to optimize SC circuits under such aggressive scaling. We find that logical and physical design techniques can be used to significantly expand the already powerful accuracy-energy tradeoff possibilities in SC circuits. Our simulation results show that our optimized SC circuits can tolerate aggressive voltage scaling with no significant SNR degradation after 40% supply voltage reduction (1V to 0.6V), leading to 66% energy saving (20.7pJ to 6.9pJ). Similarly, a 100% frequency boosting (400ps to 200ps) of the optimized circuits leads to no significant SNR degradation for several representative circuits.**

I. Introduction

Energy and power constraints have become a major challenge to IC design in recent years. Many embedded systems such as wearable devices and medical implants have strict power and energy requirements due to battery capacity and physiological limitations [20]. For example, body tissue may be damaged by excessive power dissipation in a poorly designed implantable circuit [10]. 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) [12] has been proposed as an alternative low-power computing technique for important applications such as error correction [19], image processing [3] [21], and neural networks [5]. 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 are $\frac{9}{12}, \frac{8}{12}, \frac{6}{12}$, respectively.[1] The main benefit of SC, as evident from Figure 1, is that simple logic gates implement complicated arithmetic functions. For example, a single AND gate implements multiplication. Furthermore, SC circuits are error-tolerant because errors of bit-flip type have minimal effect on the numerical value of a long bit-stream and tend to cancel each other out. Finally, SC circuits provide a natural energy-accuracy tradeoff: the bit-stream length N, i.e., the number of clock cycles an SC circuit uses to perform a computation, directly affects its energy consumption and its accuracy.

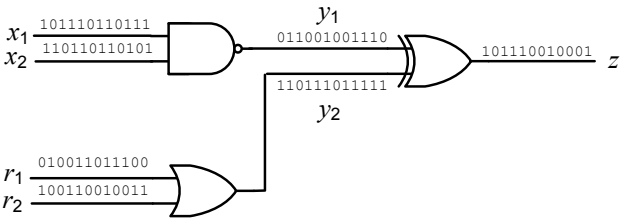


Fig. 1: Stochastic computing circuit implementing the function $Z = \frac{1}{4}+\frac{1}{2}X_1X_2$. The stochastic number represented by each bit-stream is the probability of seeing a 1 in a randomly chosen position.

In this paper, 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, i.e., reducing the supply voltage of a circuit, reduces the circuit's energy consumption but increases its latency. If latency overhead is allowable by the application context, aggressive voltage scaling can be applied at the cost of occasionally erroneous results. Thus, voltage scaling allows designers to trade accuracy for energy. This approach has been extensively studied in the non-SC literature [15] [16] [18], 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. Our work reported here shows that SC circuits can tolerate up to 40% voltage reduction with no significant error. To our knowledge, this is the first time such techniques have been applied to SC circuits. Note that the term "stochastic computing" has been also used (more recently) to describe conventional circuits involving probabilistic behavior, including scenarios with voltage/frequency scaling [24] [25]. What we refer to as SC is the computation technique that has been around since the 1960s [12], and that is unrelated to the concepts used in [24] and [25].

Figure 2 compares images generated by conventional binary and SC circuits at different supply voltage levels. It can be seen that while the SC circuits tolerate aggressive voltage scaling, the binary circuits' output quality quickly drops, even with modest voltage changes. This enables significant energy savings in SC circuits. In the example of Figure 2, the SC circuit at $V_{dd} = 0.6$V achieves the same accuracy as the conventional circuit at $V_{dd} = 1$V, thus the SC circuits consume about 44% less energy. To analyze SC behavior under voltage scaling, we develop a method of accuracy and error evaluation based on Markov chains. This method is then used to find the minimum-energy operating point of an SC circuit for a given accuracy metric. Furthermore, we present synthesis and physical design techniques that can improve SC's accuracy-power tradeoff possibilities.

[1]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 $Z = Y_1+Y_2-2Y_1Y_2 = \frac{1}{4}+\frac{1}{2}X_1X_2$.

978-1-4673-8388-2/15/$31.00 ©2015 IEEE

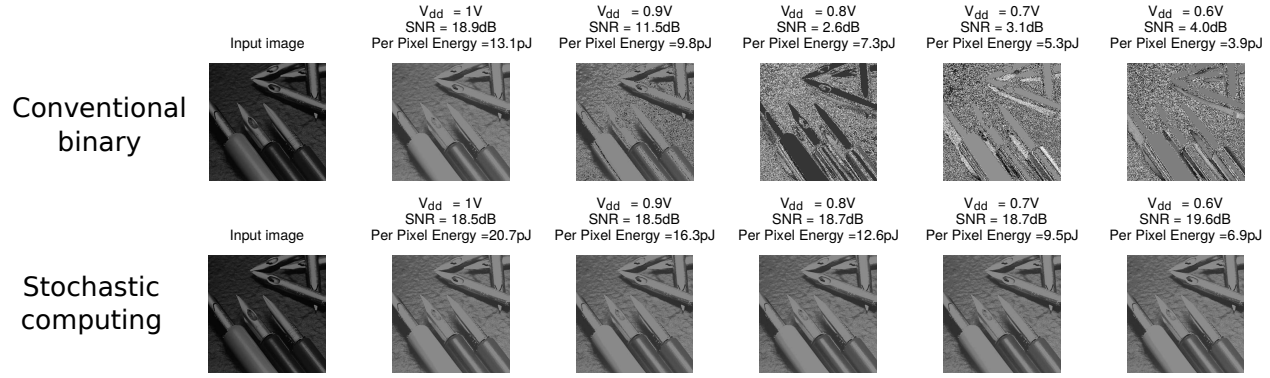


Fig. 2: Voltage scaling results of gamma correction (design obtained from [23]) executed by conventional and stochastic circuits (both implemented in 28nm FDSOI technology). After applying our proposed optimization techniques, the stochastic circuits show better tolerance against aggressive voltage scaling. The SNR of SC is not monotonic when the voltages are scaled. This is because the random bit-stream sensitizes different paths when delays scale.

This paper is organized as follows. Section II gives a brief review of SC and its error tolerance. In addition, two related problems are defined in this section. Section III discusses our proposed method of finding the minimum-energy operating point of an SC circuit for a desired accuracy level. Section IV shows the opportunities and the proposed methods of optimizing SC circuits at different voltage levels. Section V presents experimental results and finally, conclusions are given in Section VI.

II. Error Tolerance in Stochastic Computing

Recall that 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 [2]. SN values range over the unit interval $[0, 1]$, and their precision is determined by N.

The inherent error tolerance of SC circuits stems from the fact that a 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, 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 [8]. The probabilistic nature of SC circuits, along with the cancellation possibilities, makes it difficult to evaluate the accuracy of SC circuits.

To quantify the accuracy of a circuit, several error metrics, such as maximum error, mean square error, etc. can be employed. We use the "average error" metric

$$err = mean(|Z - Z^*|)$$

where Z is the exact 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 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 the deduced error bounds will be probabilistic.

Our work investigates the application of voltage and frequency scaling to SC circuits. By voltage scaling, we refer to the the systematic reduction of the power supply voltage (i.e., "undervolting"), which is a standard technique used to reduce 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 a speed higher than its nominal speed, at the cost of timing errors. It is possible to use design methods such as Razor [11] 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 computation correctness when the voltage (or frequency) scaling is extremely aggressive and the error rate is relatively high.

While SC circuits easily tolerate errors of the bit-flip type, they are also tolerant of timing errors induced by voltage/frequency scaling. 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 $\frac{1}{N}$, 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 $\frac{1}{N}$. Since the numbers of 0-to-1 and 1-to-0 transitions are almost the same for any bit-stream, these timing errors tend to cancel each other out. This error cancellation is maximized if the rates of 0-to-1 and 1-to-0 errors are exactly the same. Figure 3 shows the average error on an SN (e.g., the output of an SC circuit) for different rates of 0-to-1 and 1-to-0 timing errors on that signal. It can be seen that the error magnitude is reduced when the rates of delay errors are the same.

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 [3]. 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

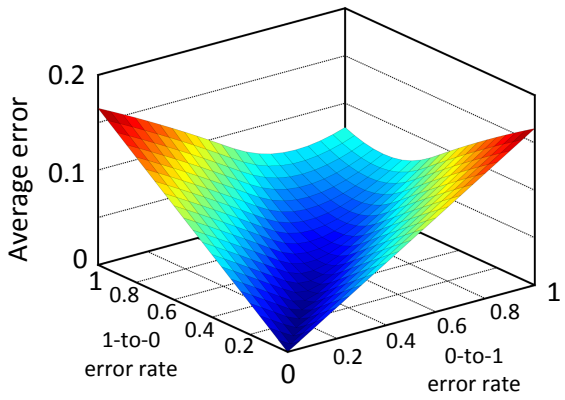


Fig. 3: Error in the magnitude of an SN for different 0-to-1 and 1-to-0 timing error rates.

length N (or, equivalently, clock cycle count) – that control their accuracy and energy/power consumption. Finding the best operating point for a circuit is thus a new and challenging problem.

Here we pose and answer the following question: "Given an SC circuit, what is the lowest energy required for computation with an average error of err_{goal}?" Previous methods search for the minimum N for which the average error is less than 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 answer to the question. We will refer to the triplet (V_{dd}, f, N) as an *operating point* of an SC circuit. Now we 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 $\leq err_{goal}$.

In addition to providing a solution to the MEOP problem, which we do in the next section, we also consider the optimization of SC circuits so that their error behavior improves 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 $k_i \cdot T \leq AAT_i \leq (k_i + 1) \cdot T$ and $k_j \cdot T \leq AAT_j \leq (k_j + 1) \cdot T$, where T is the clock period. We say 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 4 shows one example of two timing paths (arcs) 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 of 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

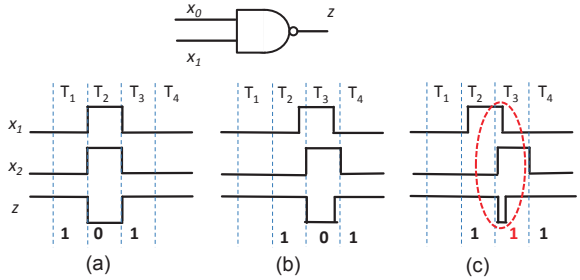


Fig. 4: Misalignment of arrival times at z with respect to clock capture phase can lead to a computation error.

sequence at z is still correct i.e., 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 which leads to computation error, as shown by the red-dotted oval in Figure 4. 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 a given SC circuit. Based on the observation made in Figure 3, our goal will be to balance the 0-to-1 and 1-to-0 error rates, thus minimizing the error. Accordingly, we define the following problem statement, whose solution is discussed in Section IV.

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

III. Error-accuracy tradeoff in SC

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 will try different operating points and, for each, evaluate the accuracy and energy of the corresponding circuit. We will 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, i.e., 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 to estimate the accuracy of the circuit by creating a Markov chain (MC) model [13].

Our proposed MC model assumes that the 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 5). 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 appear in

[2]In Case (b), the output at the first cycle, i.e., T_1, can be incorrect. However, the corresponding impact on computation accuracy is negligible given that N is typically large, e.g., N = 4,096.

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 timing delay error
D_1	Output is 1 and is incorrect due to a timing delay error
G_0	Output is 0 and is incorrect due to a timing glitch
G_1	Output is 1 and is incorrect due to a timing glitch

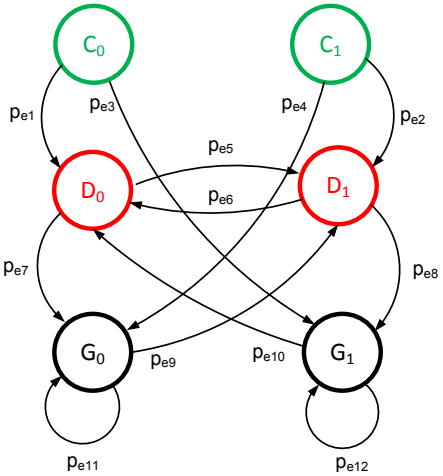


Fig. 5: Markov chain (MC) model for the proposed error estimation approach. The description of each state is given in Table I.

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 fault, 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 they 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, i.e., with transition 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, we can find the equilibrium probability (i.e., stationary) distribution of the MC, and then evaluate the accuracy of the circuit in question. We do this by calculating the probability of seeing a 1 at the output of the circuit, i.e., the probability of being in states C_1, D_1, or G_1, and comparing it with the correct output probability. To construct the MC model of a given circuit, we obtain the transition probabilities by generating a small input sample set and simulating the circuit. We then gather statistics of the transition rates between the states. We continue the simulation and incrementally increase the input sample sizes until these transition probabilities converge (i.e., when the difference between predicted errors in consecutive simulations is small). Once the transition probabilities are estimated, we plug them into the MC model of Figure 5 and evaluate the accuracy of the circuit. We note that the MC model construction is performed only once for each (design, operating point) combination. Further, design space exploration with our MC model is less time-consuming since it avoids exhaustive simulations.

We verify our modeling flow by comparing the average error values predicted by the MC model and by post-layout simulation. We use the gamma correction circuit [23] as the testcase for this verification. (A complete list of testcases that we use in our studies is given in Section V below.) After logic synthesis, placement and routing (SP&R), the circuit is simulated in Cadence NC-Verilog [6] 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} = 1.0V, worst process corner, 125°C, and clock period = 400ps; it is then operated at lower voltages (V_{dd} = $\{0.7,\ 0.68,\ ...0.6\}$V) and boosted clock frequencies (clock period = $\{400,\ 350,\ ...,\ 200\}$ps).

Figure 6 shows that the predicted average errors are well-correlated to the post-layout simulation when the errors are relatively large (> 0.1). For small errors, the MC model's prediction becomes pessimistic. Applying a margin allows the MC model to guardband accuracy for larger errors, but makes the MC model more pessimistic for the small errors. Since quickly exploring the space of operating points is more important when the error is large, the MC model's prediction becomes more important in large-error cases. We therefore apply a margin (e.g., 10%) in our experiments.

IV. Circuit-Level Optimization

The previous section dealt with a scenario in which the SC circuit is already implemented and we can only choose an operating point for it, i.e., the MEOP problem defined in Section II. In this section, we consider a scenario where we can optimize the SC 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

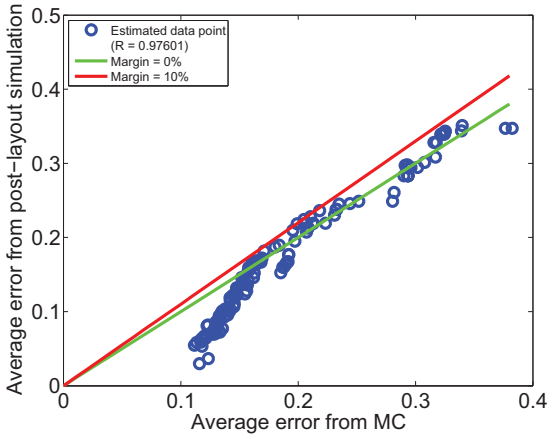


Fig. 6: Plot showing good correlation between the errors estimated by the Markov chain (MC) model and the errors obtained from post-layout simulations. A 10% margin added to the MC estimated errors is sufficient to guard against small discrepancies.

as the opportunities to eliminate them (Section IV-A). We then discuss proposed optimization methods (Section IV-B).

A. Arrival Time Misalignment Matters

To examine the impact of arrival time misalignment on computation accuracy in an SC circuit, we insert and incrementally increase the input delays at the circuit's inputs from 0ps to 450ps, i.e., 3× 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 7(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.

The results in Figure 7(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, e.g., 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, e.g., 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 7(b)). Therefore, to reduce the likelihood of the misalignment of arrival times and to minimize the computation error, we propose some circuit optimization methods to minimize input-output path delay differences in SC circuits.

B. 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 a circuit's path delays.

First, we examine two major SC design styles: those based on the STRAUSS (Spectral TRAnsform Use in Stochastic circuit Synthesis) method [1], and those based on the ReSC (Reconfigurable Stochastic Computing) architecture [23]. We then compare their path delays and computation errors for a given range of supply voltages. Figure 8 compares STRAUSS and ReSC for testcase PolySmall in 28nm FDSOI technology. We observe that the SC circuit implemented with ReSC tends to have more balanced path delays and smaller errors than 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 a less symmetric 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 V.

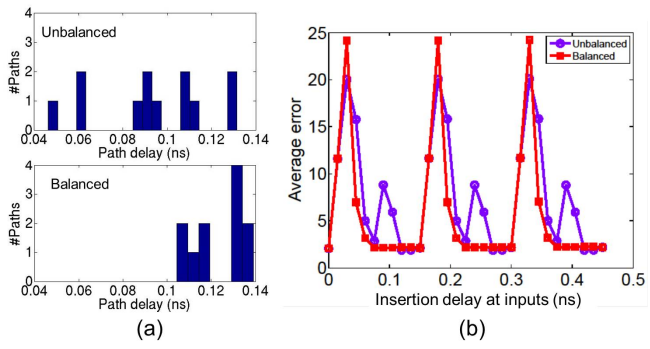


Fig. 7: Design: PolySmall. Technology: 28nm FDSOI. Clock period = 150ps. (a) Path delay distributions of two implementations. (b) Computation error changes with an increase of the input delay.

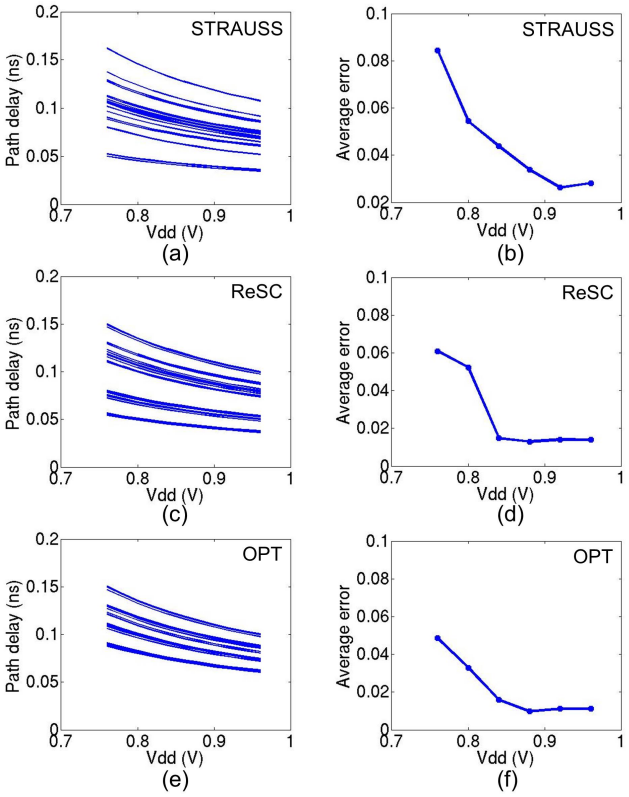


Fig. 8: Path delays (left) and average computation errors (right) at supply voltages ranging from 0.72V to 0.98V for testcase PolySmall at 28nm FDSOI. Each trace in (a)(c)(e) denotes a timing path with unique combination of rise/fall transitions. (a-b) STRAUSS [4]; (c-d) ReSC [23]; and (e-f) optimized circuit using our proposed MILP-based method.

We further perform buffer insertion and/or route detouring at the post-routing stage to balance path delays. 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 [9] [14]. Given that SC circuits typically have small sizes[3], we formulate a Mixed Integer Linear Program (MILP) to search for the optimal buffer insertion and/or route detouring solution based on a given set of buffering candidates. Figures 8(e-f) show the resultant path delays and computation errors of the optimized SC circuit; we

[3]Typical image-processing SC circuits have only around 20 gates [3]; and to our knowledge the largest SC circuits have no more than 1,250 gate instances [21].

TABLE II: Description of notations used in the MILP.

Term	Meaning
V_k	Supply voltage, ($1 \leq k \leq K$; V_K is the highest voltage)
P_i	Timing path, ($1 \leq i \leq M$)
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 ($1 \leq r \leq R$)
d_j^k	Delay increase due to buffer insertion and/or routing at V_k, ($1 \leq j \leq Q$)
g_j^k	Leakage power penalty of buffer insertion choice at V_k, ($1 \leq j \leq Q$)
c_{rj}	Indicator of buffer insertion and/or routing detour on n_r

observe significant improvement over both the unoptimized STRAUSS and ReSC implementations.

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 notations used in our formulation are given in Table II.)

$$\text{Minimize} \quad U \tag{1}$$

$$\text{subject to} \quad D_i'^k = D_i^k + \sum_{1 \leq i \leq M, 1 \leq j \leq Q} c_{rj} \cdot d_j^k \tag{2}$$

$$\sum_{1 \leq j \leq Q} c_{rj} \leq 1, \qquad \forall\ 1 \leq r \leq R \tag{3}$$

$$D_{max}^k = \max_{1 \leq i \leq M} D_i^k, \ \forall\ 1 \leq k \leq K \tag{4}$$

$$\alpha \cdot D_{max}^k \geq D_i'^k, \qquad \forall\ 1 \leq i \leq M,\ 1 \leq k \leq K \tag{5}$$

$$D_{max}'^k \geq D_i'^k, \qquad \forall\ 1 \leq i \leq M,\ 1 \leq k \leq J \tag{6}$$

$$D_{min}'^k \leq D_i'^k, \qquad \forall\ 1 \leq i \leq M,\ 1 \leq k \leq K \tag{7}$$

$$U \geq \frac{D_{max}^K}{D_{max}^k} \cdot (D_{max}'^k - D_{min}'^k) \tag{8}$$

$$\beta \cdot G^k \geq \sum_{1 \leq r \leq R, 1 \leq j \leq Q} c_{rj} \cdot g_j^k,\ 1 \leq k \leq K \tag{9}$$

where $D_i'^k$ is the optimized path delay of path P_i, with 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. We use the empirical values $\alpha = 1.1$ and $\beta = 1.2$ in the experiments.

To ensure the feasibility of ECOs (engineering change orders), we characterize lookup tables (LUTs) based on buffer insertion and/or routing detour candidates with different input slew and load capacitance values, which are needed for the MILP formulation. We formulate our MILP and optimize circuits based on the characterized LUTs. The approach is similar to what is applied in [14]. To balance path delays at a range of supply voltages and minimize the MILP runtime, we select buffer insertion and/or routing detour candidates such that they 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 [7]. 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 9. Our approach selects from buffers and inverters of various sizes based on the delay requirements. We use both low V_t (LVT) and regular V_t (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 buffer) 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 solution with minimum leakage power from each sub-region. Figure 10(a) shows the solution space with up to five stages of buffering candidates. Figure 10(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.[4]

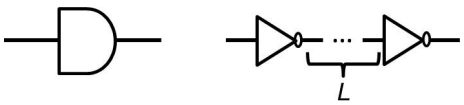


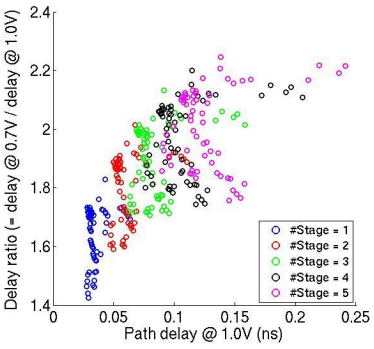
Fig. 9: Applied buffering styles: a single-stage non-inverting buffer, and an inverter pair with routing detour.

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 1. In the design flow, we first insert the first inverter. We then legalize the location of the inserted inverter so that there is enough space for wire detour, e.g., to move the inverter away from the die 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 1W2S (single-width double-spacing) routing rule on layers M3 and M4, between two inverters. An example of detoured routing is shown in Figure 11.

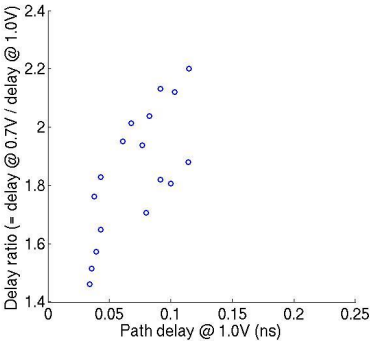
Algorithm 1 Insertion flow of inverter pairs.

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 and two rows in horizontal and vertical directions
4: Perform routing detour with 1W2S

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



(a) Buffering solution space.



(b) Pruned buffering candidates.

Fig. 10: (a) Buffering solution space, i.e., delay range and delay-voltage tradeoff range, with multiple stages of buffers/inverter pairs. The colors of circles denote different numbers of stages of buffers/inverter pairs. (b) Pruned buffering candidates.

V. Experimental Results

Our experiments are implemented in foundry 28nm FDSOI technology. We synthesize our designs using *Synopsys Design Compiler vH2013.03-SP3* [26], and place and route them using *Synopsys IC Compiler vI-2013.12-SP1* [27]. We use *Synopsys PrimeTime vH-2013.06-SP2* [28] 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* [6]. We construct the Markov chain model using *MATLAB R2013a* [22]. The MILP solver used in our optimization flow is *CPLEX v12.5* [17]. 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.

To evaluate the effectiveness of our optimization methods, we apply them to the testcases and compare the results with those of the unoptimized circuits. Figure 12 shows

TABLE III: Summary of testcases.

Testcase	# of cells	Description
GammaCorrection	∼100	A common image processing task [23]
EdgeDetection	∼5	A common image processing task [3]
PolySmall	∼20	A simple polynomial of degree 3 implemented using methods of [1] and [23]
Neuron	∼500	A 128-input neuron [5]

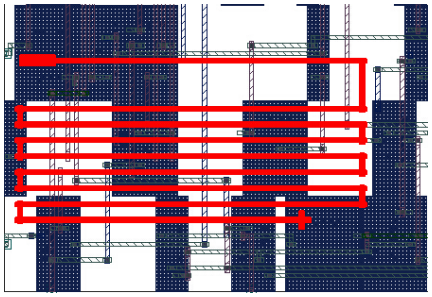


Fig. 11: Layout of routing detour (in red). The detoured wirelength is 40µm. Shaded blocks are standard cells.

the minimum energy required for each design to meet a given average error constraint err_{goal}. In spite of the power overhead due to added buffers and wires, the improved accuracy of the optimized circuits allows more aggressive voltage scaling to achieve lower circuit power. We see 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 43% energy reduction occurs in the GammaCorrection testcase with $err_{goal} = 0.07$. When the error constraints are loose, or when the design is fairly balanced, e.g., EdgeDetection [3], the unoptimized circuits can also perform satisfactorily at low supply voltages. In such cases, optimizing the circuit is not as efficient because the inserted buffers increase the overall energy consumption. Studying the tradeoff between the cost (power overhead) of buffer insertion versus its benefits (i.e., the fact that it allows voltage downscaling) is among our future tasks.

To gauge the effectiveness of our MC model, we perform an energy-accuracy comparison between the operating points selected by the MC-based flow versus the points selected by exhaustive simulation; see Figure 13. We observe that when the average error is relatively large (greater than 0.1), the MC model estimates the error correctly, and hence selects

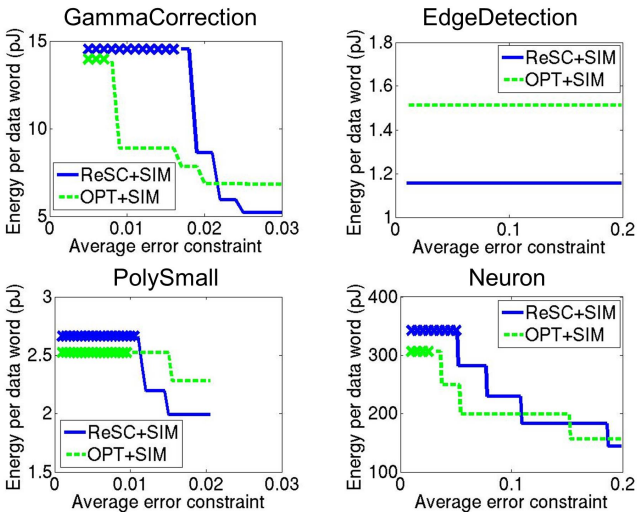


Fig. 12: Energy comparison for different accuracy requirements (err_{goal}) between unoptimized implementation (blue solid line) and our optimized circuit (green dashed line). Operating points are selected based on exhaustive simulation. The V_{dd} range is 0.7V to 1.0V. A cross sign ($\times$) indicates that no suitable operating point was found for the given err_{goal}.

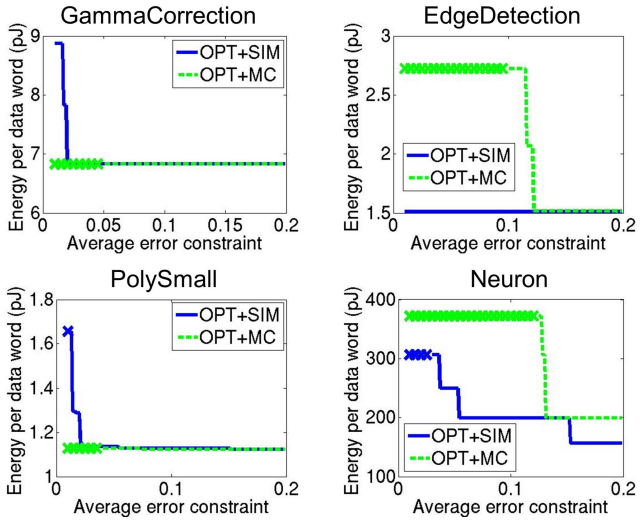


Fig. 13: Energy comparison between operating points from MC model-based search flow (green dashed line) versus exhaustive search (blue solid line) for different accuracy requirements (err_{goal}). A cross sign ($\times$) indicates that no suitable operating point was found. Runtimes of MC-based searching and exhaustive simulation for each testcase are shown in Table IV.

TABLE IV: Runtime comparison between exhaustive simulation and MC model-based search.

	#Cycles (Ex.)	#Cycles (MC)
GammaCorr	1024	10
EdgeDetection*	1000	10
Polysmall	256	10
Neuron*	100	10

*Due to the large number of possible inputs, which makes exhaustive simulation infeasible, we utilize a smaller sample set for the MC model-based flow.

an operating point that is very similar to the one selected by the exhaustive simulation. In such cases, the MC-based flow runs much faster than the exhaustive simulation, while achieving the same result. However, when the average error is low, the MC model's error estimation becomes pessimistic, leading to a suboptimal operation point selection. This has been observed and discussed in Section III.

VI. Conclusions

In this paper we present novel methods of exploiting accuracy-energy tradeoffs in SC circuits. We employ voltage and frequency scaling to reduce energy consumption at the cost of timing errors. We show, for the first time, that SC circuits are very tolerant of timing errors, and hence can tolerate aggressive voltage/frequency scaling without much loss of accuracy, unlike most conventional binary circuits. Based on this result, we define and solve the problem of finding the minimum-energy operating point of an SC circuit for a desired accuracy level. To quickly explore the operating point space, we propose a Markov chain model for SC circuits. Furthermore, we observe that the accuracy of SC circuits, under scaled conditions, can be improved by balancing the path delays. Accordingly, we propose methods of optimizing SC circuits during the logical and the physical design steps. Our proposed methods have been successfully applied to several representative SC circuits, achieving substantial energy reduction without significant accuracy loss. Our future work will include studying the tradeoffs among computation latency, process variation, temperature fluctuation and voltage/frequency scaling, and considering these tradeoffs in our optimization process. We also plan to study reliability issues (e.g., aging) in SC circuits.

Acknowledgments

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

References

[1] A. Alaghi and J. P. Hayes, "STRAUSS: Spectral Transform Use in Stochastic Circuit Synthesis," *IEEE Trans. CAD*, 2015.
[2] A. Alaghi and J. P. Hayes, "Survey of Stochastic Computing," *ACM Trans. Embedded Computing Systems* 12(2), pp. 92:1–92:12, 2013.
[3] A. Alaghi, C. Li and J. P. Hayes, "Stochastic Circuits for Real-Time Image-Processing Applications," *Proc. DAC*, pp. 136:1–136:6, 2013.
[4] A. Alaghi and J. P. Hayes, "A Spectral Transform Approach to Stochastic Circuits," *Proc. ICCD*, pp. 315–312, 2012.
[5] B. D. Brown and H. C. Card, "Stochastic Neural Computation. I. Computational Elements," *IEEE Trans. Computers* 50(9), pp. 891–905, 2001.
[6] Cadence, *NC-Verilog User's Manual.* http://www.cadence.com/
[7] T.-B. Chan and A. B. Kahng, "Tunable Sensors for Process-Aware Voltage Scaling," *Proc. ICCAD*, pp. 7–14, 2012.
[8] T. H. Chen, A. Alaghi and J. P. Hayes, "Behavior of Stochastic Circuits under Severe Error Conditions," *it - Information Technology* 56(4), pp. 182–191, 2014.
[9] C. C. N. Chu and D. F. Wong, "A Quadratic Programming Approach to Simultaneous Buffer Insertion/Sizing and Wire Sizing", *IEEE Trans. VLSI* 18(6), pp. 787–798, 1999.
[10] H. Chun, Y. Yang and T. Lehmann, "Safety Ensuring Retinal Prosthesis With Precise Charge Balance and Low Power Consumption," *IEEE Trans. Biomedical Circuits and Systems* 8(1), pp. 108–118, 2014.
[11] D. Ernst et al., "Razor: a Low-Power Pipeline Based on Circuit-Level Timing Speculation," *Proc. MICRO*, pp. 7–18, 2003.
[12] B. R. Gaines, "Stochastic Computing Systems", *Advances in Information Systems Science*, pp. 37–172, 1969.
[13] C. M. Grinstead and L. J. Snell, *Introduction to Probability, 2nd ed.*, American Math. Soc., 2003.
[14] K. Han, A. B. Kahng, J. Lee, J. Li and S. Nath, "A Global-Local Optimization Framework for Simultaneous Multi-Mode Multi-Corner Skew Variation Reduction", *Proc. DAC*, pp. 26:1–26:6, 2015.
[15] K. He, A. Gerstlauer and M. Orshansky, "Low-Energy Signal Processing Using Circuit-Level Timing-Error Acceptance," *Proc. Intl. Conf. on IC Design and Technology*, pp. 1–4, 2012.
[16] R. Hegde and N. R. Shanbhag, "Soft Digital Signal Processing," *IEEE Trans. VLSI* 9(6), pp. 813–823, 2001.
[17] IBM *CPLEX Optimizer.* http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[18] A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Slack Redistribution for Graceful Degradation Under Voltage Overscaling," *Proc. ASP-DAC*, pp. 825–831, 2010.
[19] X. R. Lee, C. L. Chen, H. C. Chang and C. Y. Lee, "A 7.92 Gb/s 437.2 mW Stochastic LDPC Decoder Chip for IEEE 802.15.3c Applications," *IEEE Trans. Circuits and Systems I: Regular Papers* 62(2), pp. 507–516, 2015.
[20] Y. Lee et al., "A Modular $1mm^3$ Die-Stacked Sensing Platform with Optical Communication and Multi-Modal Energy Harvesting," *IEEE J. of Solid-State Circuits (JSSC)* 48(1), pp. 229–243, 2013.
[21] P. Li and D. J. Lilja, "Using Stochastic Computing to Implement Digital Image Processing Algorithms", *Proc. ICCD*, pp. 154–161, 2011.
[22] MathWorks, Inc., *MATLAB and Statistics Toolbox Release 2012b*, Natick, Massachusetts.
[23] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," *IEEE Trans. Computers* 60(1), pp. 93–105, 2011.
[24] J. Sartori, J. Sloan and R. Kumar, "Stochastic Computing: Embracing Errors in Architecture and Design of Processors and Applications," *Proc. Intl. Conf. Compilers, Architectures and Synthesis for Embedded Systems*, pp. 135–144, 2011.
[25] N. R. Shanbhag, R. A. Abdallah, R. Kumar and D. L. Jones, "Stochastic Computation," *Proc. DAC*, pp. 859–864, 2010.
[26] Synopsys, *Design Compiler User's Manual.* http://www.synopsys.com/
[27] Synopsys, *IC Compiler User Guide.* http://www.synopsys.com/
[28] Synopsys, *PrimeTime User's Manual.* http://www.synopsys.com/