Design of Reliable Stochastic Number Generators Using Emerging Devices for Stochastic Computing

Meng Yang and Weikang Qian
University of Michigan-Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University, Shanghai, China
Email: {yangm.meng, qianwk}@sjtu.edu.cn

***Abstract*—Stochastic computing (SC), an unconventional computing paradigm that operates on stochastic bit streams, gains more and more attention recently because of the low area and power consumption of its computing core. SC relies on stochastic number generators (SNGs) to generate input stochastic bit streams. An SNG is composed of a random number source (RNS) and a comparator. However, conventional SNGs, which use linear feedback shift register (LFSR) as the RNS, consume much more area and power than the SC core, offsetting the small area and the low power advantages of the SC core. To mitigate this issue, some emerging devices, such as memristor and spintronic devices, were proposed to build SNGs. However, due to the large process variation in fabricating these devices, these SNGs are unreliable: their output probabilities could have a large error than the desired values. In this paper, we propose a general method to design reliable SNGs that use emerging device-based RNS's. Experimental results showed that our design is effective in improving the reliability.**

Keywords—stochastic computing; novel nanoscale devices; stochastic number generator (SNG)

I. INTRODUCTION

Stochastic computing (SC) was first introduced in 1960s as an unconventional computing paradigm [1]. Its biggest difference from traditional binary computing is that it operates on stochastic bit streams that encode real values through the probabilities of 1s in the streams. For example, the bit stream "01001100" represents 3/8. Due to the uniformly weighted encoding, SC has strong fault tolerance to bit flip errors. Moreover, the probabilistic way of encoding data allows very simple digital circuits to realize complex arithmetic operations. One notable example is that multiplication can be realized by an AND gate, as shown in Fig. 1(a). These circuits taking stochastic bit streams as inputs and outputs are referred to as *SC core*.

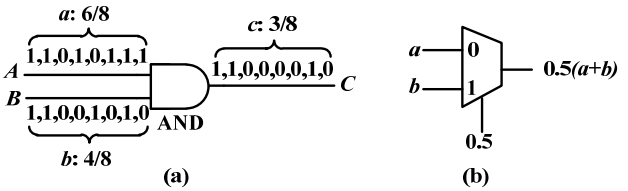

Fig. 1. Examples of stochastic computing elements. (a) Multiplication of the numbers encoded by stochastic bit streams can be realized by an AND gate; (b) Scaled addition of the numbers encoded by stochastic bit streams can be realized by a 2-to-1 multiplexer.

Because of its low area and power consumption, SC has been applied in many application domains, such as image processing [2][7], low-density parity-check (LDPC) decoding [3], artificial neural networks [6], and digital filters [5].

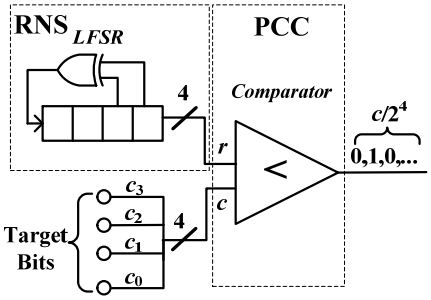

Fig. 2. A stochastic number generator composed of an LFSR and a comparator.

SC relies on stochastic number generators (SNGs) to generate input stochastic bit streams of the desired probabilities. A widely used SNG is composed of a pseudo random number source, such as a linear feedback shift register (LFSR), and a comparator, as shown in Fig. 2 (more details on it will be discussed in Section II). However, it consumes much more area and power than the SC core. Furthermore, to guarantee the correct functionality, the input stochastic bit streams should be mutually independent, which means the number of SNGs is equal to the number of inputs of the SC core. As a result, the existence of the large and power-hungry SNGs significantly mitigates the advantages of the SC core in area and power consumption.

To reduce the area and power consumption of the SNGs in a stochastic circuit, several solutions were proposed. One method is to share a single LFSR among multiple SNGs. For example, in [14], the circular shift of the output bits of the LFSR was proposed to produce stochastic bit streams with low correlation. In [15], the insertion of delay elements into the circuit was proposed to decorrelate the stochastic bit streams. However, these solutions cannot guarantee the perfect mutual independence of the input streams and hence, could introduce error to the output result.

Another method is to leverage the emerging devices, such as memristors [8][10] and spintronic devices [9][11][12]. These novel nanoscale devices usually have two different stable states that can be converted into binary values. For example, the memristor has a high-impedance state and a low-impedance state, which can be switched by applying a programming pulse to the device. Recent studies showed that the state switching of some emerging devices is random and the switching probability could be controlled by an input signal [8]. For example, the state switching probability of a memristor is

$$P(t) = 1 - e^{-t/\tau}, \quad (1)$$

where t is the width of the programming pulse and τ is a constant value determined by the device itself and the amplitude of the programming pulse. Therefore, by changing the value of t, a stochastic bit stream of an arbitrary probability can be generated. Previous studies showed that SNGs based on these emerging devices have much smaller area and lower power consumption than the conventional CMOS-based SNGs. For example, the power consumption of the design based on memristor is 16 times lower than that of CMOS [8], while the design based on spintronic devices can achieve a power reduction of 7 times [12].

However, the probabilities of the stochastic bit streams generated by emerging device-based SNGs are subject to a large error due to the noise in the input control signal and the large process variation occurred in manufacturing these devices. Take the memristor SNG as an example. As shown in Eq. (1), its switching probability depends both on the programming pulse and on its device parameters and thus, is vulnerable to both the noise in the programming pulse and variation in its device parameters. In order to fully take advantage of these emerging devices to build SNGs with small area and low power consumption, we should have a reliable design that guarantees the correct output probability even when the underlying devices are subject to large signal noise and process variation.

In this paper, we propose a general method to design such reliable SNGs using these emerging devices as the random sources. Our method is not restricted to any specific emerging devices as long as they can be used to build unbiased random sources. We demonstrate the basic principles to build reliable SNGs from these unreliable unbiased random sources and show a few implementations based on the design principles. Experimental results show that our proposed techniques can ensure the correct output probability of those emerging device-based SNGs while still taking their advantages in area and power consumption.

The rest of this paper is organized as follows. In Section II, we introduce the background on SNGs. In Section III, we present the proposed method to design a reliable SNG. In Section IV, we show the experimental results, which illustrate that our proposed design is very effective in improving the reliability. Finally, in Section V, we conclude the paper.

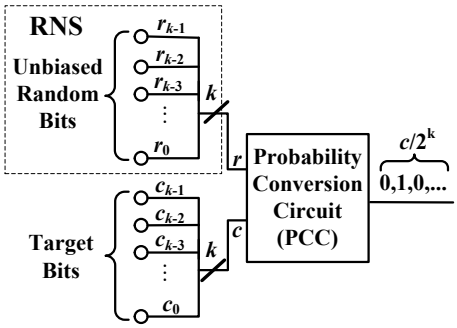

Fig. 3. A general stochastic number generator.

II. Background on Stochastic Number Generators

A general SNG is shown in Fig. 3. It is composed of a random number generator (RNS) and a probability conversion circuit (PCC). The RNS is composed of k independent unbiased random binary sources, each with probability of 0.5 to be a 1 and the same probability to be a 0. For simplicity, an unbiased random binary source is referred to as an *unbiased random bit*. The PCC takes k unbiased random bits $r_{k-1}, r_{k-2}, \dots, r_0$ and k target bits $c_{k-1}, c_{k-2}, \dots, c_0$ as inputs, where k gives the precision of the output probability. If the k target bits encoded a binary number $C = (c_{k-1}c_{k-2} \dots c_0)_2$, then the PCC outputs a stochastic bit stream with probability $C/2^k$. For simplicity, in what follows, we refer to a stochastic bit stream with probability p as a probability p.

One typical RNS is the LFSR shown in Fig. 2, since each bit of an LFSR is approximately an unbiased random bit and all of them are mutually independent. A commonly used PCC is a comparator as shown in Fig. 2. The comparator outputs a one if and only if the binary number $R = (r_{k-1}r_{k-2} \dots r_0)_2$ is less than C. Then, it is obvious that the output probability of the comparator is $C/2^k$.

Besides the comparator, there are two other widely-used PCCs proposed in literature. One is the weighted binary generator (WBG), first introduced by Gupta and Kumaresan in 1988 [13]. An example of WBG with the precision $k = 4$ is shown in Fig. 4. It consists of a number of AND gates to implement the stochastic multiplication and an OR gate at the output stage to add all weighted probabilities together. Specifically, given that $r_3, \dots r_0$ are unbiased random bits, we have

$$P(W_3) = \frac{1}{2}, P(W_2) = \frac{1}{2^2}, P(W_1) = \frac{1}{2^3}, P(W_0) = \frac{1}{2^4}.$$

Therefore, the output probability of the final OR gate is

$$P = P(W_3)c_3 + P(W_2)c_2 + P(W_1)c_1 + P(W_0)c_0$$
$$= \frac{1}{2}c_3 + \frac{1}{2^2}c_2 + \frac{1}{2^3}c_1 + \frac{1}{2^4}c_0 = \frac{C}{2^4}.$$

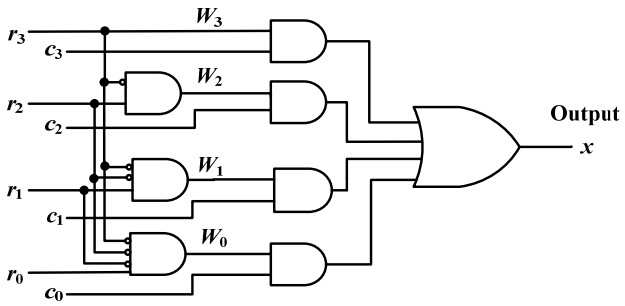

Fig. 4. A weighted binary generator with $k = 4$.

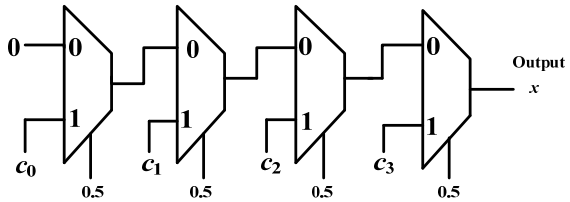

Fig. 5. A multiplexer chain with $k = 4$.

Another PCC is the multiplexer (MUX) chain. An example with the precision $k = 4$ is shown in Fig. 5 [4]. For a MUX, if the probabilities of its two data inputs are a and b and the probability of its selecting input is 0.5, then its output probability is $0.5(a + b)$, a scaled sum on the two input probabilities, as

shown in Fig. 1(b). Given the configuration shown in Fig. 5, we can easily see that the output probability of the MUX chain is

$$P = \frac{1}{2}\left(c_3 + \frac{1}{2}\left(c_2 + \frac{1}{2}\left(c_1 + \frac{1}{2}(c_0 + 0)\right)\right)\right) = \frac{C}{2^4}.$$

III. Design of Reliable Stochastic Number Generators

In this section, we first show the basic assumption used in our approach and give a motivating example. Then the design principles for reliable SNGs are presented.

A. Basic Assumption and Motivation

In our approach, we make the following assumption: Small-area and low-power unbiased random bits built from emerging devices are available, which are used as the RNS. This is a valid assumption. For example, we can configure the pulse width t to build a memristor-based unbiased random bit.

Note that our approach is different from that of [8], in which only one memristor-based random source is used. In our case, multiple memristor-based random sources are used. However, due to the small area and low power advantages of emerging device-based random sources, this is affordable. Furthermore, we argue that our approach has the benefit of reducing the overhead to generate input control signals. With the previous approach [8], different control signals should be provided to different SNGs. Furthermore, each control signal should be tunable in order to provide different probabilities at different time. In our case, only one fixed control signal is required, which produces the 0.5 output probability under the nominal conditions. For example, if we know the nominal value for the parameter τ in Eq. (1) is τ_0, then we only need one pulse with width $t_0 = \tau_0 \ln 2$, which controls all the memristor-based random sources. This ensures that each random source is unbiased when its parameter τ is equal to the nominal value τ_0. Although in normal cases, the actual τ for each random source fluctuates around τ_0, such a fluctuation can be well-tolerated by our proposed design technique, which will be introduced next.

The basic idea of our approach is to design the PCC properly so that it can guarantee the reliable output probability even if the unbiased random sources are unreliable. Next, we first discuss the issue of using a traditional PCC when the unbiased random bits are unreliable.

Due to the noise in the input control signals and the process variation of the devices, each random bit in the RNS cannot be guaranteed to produce a probability of 1/2. Instead, the output probability fluctuates around 1/2. For example, assume an RNS is composed of two random bits r_1 and r_0, which are implemented by emerging devices. Their probabilities are 0.55 and 0.59, respectively. Table I shows the probability of occurrence for each combination of r_1 and r_0. For example, the occurrence probability for the combination $r_1 = 0$ and $r_0 = 0$ is $(1 - 0.55) \times (1 - 0.59) = 0.1845$. In the case where each random bit is unbiased, we expect that each combination of r_1 and r_0 occurs with the probability of 0.25. The third column of Table I lists the difference of the actual occurrence probability for each combination from 0.25.

Table I. The probability of occurrence for each combination of r_1 and r_0. Assume $P(r_1 = 1) = 0.55$ and $P(r_0 = 1) = 0.59$. The fourth and fifth columns show the output values of two example Boolean functions f_1 and f_2, both producing the output probability 0.5 when $P(r_1 = 1) = P(r_0 = 1) = 0.5$.

$r_1 r_0$	probability p	$p - 0.25$	$f_1(r_1, r_0)$	$f_2(r_1, r_0)$
00	0.1845	−0.0655	1	1
01	0.2655	+0.0155	1	0
10	0.2255	−0.0245	0	0
11	0.3245	+0.0745	0	1

Now assume that a 2-bit comparator is used as the PCC to produce the output stochastic bit stream. For a given pair of target bits c_1 and c_0, the output probability is the probability that the binary number $(r_1 r_0)_2$ is strictly less than the binary number $(c_1 c_0)_2$. Table II lists the expected output probability and the actual probability for each pair of c_1 and c_0. For example, when $c_1 = 1$ and $c_0 = 1$, the expected output probability is 0.75, while the actual output probability, based on the values in Table I, is $0.1845 + 0.2655 + 0.2255 = 0.6755$. The fourth and the fifth columns of Table II show the absolute and the relative errors of the actual output probability over the expected value, respectively. We can see that using the comparator as a PCC, the output probability could be quite far away from the expected value when the unbiased random bits are unreliable. Similar conclusions can be drawn for other traditional PCCs.

The key to ensure the correctness of the output probability lies in the proper design of the PCC, which we will explore next.

Table II. Comparison between the actual and the expected output probabilities with the PCC as a comparator. The same assumption for Table I is used.

$c_1 c_0$	expected probability	actual probability	absolute error	relative error (%)
00	0	0	0	0
01	0.25	0.1845	0.0655	26.2
10	0.5	0.45	0.05	10
11	0.75	0.6755	0.0745	9.9

B. Design Principle for a Reliable Fixed Output Probability

In this section, we first demonstrate the principle to design a circuit to produce a reliable *fixed* output probability. Such a circuit takes a number of unreliable unbiased random bits as inputs and produces a given target probability $C/2^k$ as accurately as possible. In what follows, we focus on the Boolean function of such a circuit.

In the ideal case where all the random bits are with probability of 0.5 to be a 1, in order to produce an output probability $C/2^k$, we only need k random bits $r_0, \ldots, r_{k-1}$ and we can use any Boolean function $f(r_{k-1}, \ldots, r_0)$ with the on-set size as C [16]. Here, the *on-set* of a Boolean function refers to the set of input combinations that make the Boolean function evaluate to 1. For example, the fourth and the fifth columns of Table I show the output columns of two truth tables of Boolean functions that produce the output probability 2/4 for the ideal case. Indeed, for the ideal case, each input combination occurs with the probability $1/2^k$. With C minterms in the on-set of the Boolean function, the output probability is $C/2^k$.

However, in real situations, the probability of a random bit fluctuates around 0.5. Assume the probability of r_i $(0 \le i \le$

$k-1$) is $0.5+\varepsilon_i$, where $|\varepsilon_i| \ll 1$. Next we consider the probability that $r_{k-1}=a_{k-1},\dots,r_0=a_0$, where $(a_{k-1},\dots,a_0)$ is any fixed input combination in the Boolean space $\{0,1\}^k$. We denote such a probability as $P(a_{k-1},\dots a_0)$.

For illustration purpose, consider $k=3$ and $a_2=0,a_1=1,a_0=0$. The probability of $r_2=0,r_1=1,r_0=0$ is

$$\begin{aligned}&P(0,1,0)=P(r_2=0,r_1=1,r_0=0)\\&=(1-0.5-\varepsilon_2)(0.5+\varepsilon_1)(1-0.5-\varepsilon_0)\\&=(0.5-\varepsilon_2)(0.5+\varepsilon_1)(0.5-\varepsilon_0)\\&=\frac{1}{8}+\frac{1}{4}(-\varepsilon_2+\varepsilon_1-\varepsilon_0)+\Delta,\end{aligned}$$

where Δ contains higher-order products on ε_i's. Given that ε_i's are small, we can ignore Δ. Therefore, we have

$$P(0,1,0)\approx\frac{1}{8}+\frac{1}{4}(-\varepsilon_2+\varepsilon_1-\varepsilon_0).$$

In the general case, we have

$$P(a_{k-1},\dots a_0)\approx\frac{1}{2^k}+\frac{1}{2^{k-1}}\sum_{i=0}^{k-1}(2a_i-1)\varepsilon_i.$$

In the ideal case, the probability $P(a_{k-1},\dots a_0)$ should be $1/2^k$. We denote the error of the probability $P(a_{k-1},\dots a_0)$ compared to its ideal value as $E(a_{k-1},\dots a_0)$. Therefore, we have

$$E(a_{k-1},\dots a_0)\approx\frac{1}{2^{k-1}}\sum_{i=0}^{k-1}(2a_i-1)\varepsilon_i.$$

From the above equation, we can also obtain

$$\begin{aligned}&E(\overline{a_{k-1}},\dots\overline{a_0})=E(1-a_{k-1},\dots,1-a_0)\\&\approx\frac{1}{2^{k-1}}\sum_{i=0}^{k-1}(1-2a_i)\varepsilon_i=-E(a_{k-1},\dots,a_0).\end{aligned}$$

This indicates that the error of the occurrence probability of the input combination $(a_{k-1},\dots,a_0)$ could exactly cancel that of the input combination $(\overline{a_{k-1}},\dots,\overline{a_0})$ under the first order approximation. Therefore, if we assign the pair of input combinations $(a_{k-1},\dots,a_0)$ and $(\overline{a_{k-1}},\dots,\overline{a_0})$ into the on-set of the Boolean function, they will contribute a value of $1/2^{k-1}$ to the output probability with a very small error. Therefore, when C is an even number, we can arbitrarily assign $C/2$ distinct pairs of the input combinations of the form $(a_{k-1},\dots,a_0)$ and $(\overline{a_{k-1}},\dots,\overline{a_0})$ into the on-set of the Boolean function. This will realize the output probability $C/2^k$ very closely.

For the example in Table I, if the desired output probability is 1/2 (i.e., $C=2$), one possible assignment of the input combinations to the on-set of the Boolean function is $(r_1,r_0)=(0,0)$ and $(1,1)$, as shown by the fifth column in Table I. With this assignment, the actual output probability is

$$\begin{aligned}&P\big((r_1,r_0)=(0,0)\big)+P\big((r_1,r_0)=(1,1)\big)\\&=0.1845+0.3245=0.509,\end{aligned}$$

which is only 1.8% away from the desired value of 1/2. In contrast, if the input combinations $(r_1,r_0)=(0,0)$ and $(0,1)$ are chosen, which corresponds to using a comparator as the PCC, the actual output probability is

$$\begin{aligned}&P\big((r_1,r_0)=(0,0)\big)+P\big((r_1,r_0)=(0,1)\big)\\&=0.1845+0.2655=0.45,\end{aligned}$$

which is 10% away from the desired value (see Table II).

However, the above principle cannot be followed exactly when C is odd, since no matter how the input combinations are assigned to the on-set, there always exists one input combination $(a_{k-1},\dots,a_0)$ in the on-set for which the matching one $(\overline{a_{k-1}},\dots,\overline{a_0})$ is not in the on-set. This could introduce a large error when C is odd. To further address this problem, we propose to use $k+1$ unbiased random bits to realize an arbitrary target probability $C/2^k$. In this case, in order to realize the output probability $C/2^k$, we can arbitrarily assign C distinct pairs of the input combinations of the form $(a_k,\dots,a_0)$ and $(\overline{a_k},\dots,\overline{a_0})$ into the on-set of the Boolean function. It is clear that a Boolean function $f(r_k,\dots,r_0)$ constructed by the above way has no output error under the first order approximation, due to the error cancelling effect of each pair of matching input combinations in the on-set. Therefore, we refer to this type of Boolean function as *error cancelling function (ECF)* for the output probability $C/2^k$.

The following theorem gives a characterization of an ECF.

Theorem 1

A $(k+1)$-input Boolean function $f(r_k,\dots,r_0)$ is an ECF for the output probability $C/2^k$ $(0\le C\le 2^k-1)$ if and only if it satisfies that $f(r_k,\dots,r_0)=f(\overline{r_k},\dots,\overline{r_0})$ and the size of its on-set is $2C$. ■

Proof: "only if" part: by the way of constructing an ECF for the output probability $C/2^k$, it is clear that $f(r_k,\dots,r_0)=f(\overline{r_k},\dots,\overline{r_0})$ and the size of its on-set is $2C$.

"if" part: since $f(r_k,\dots,r_0)=f(\overline{r_k},\dots,\overline{r_0})$ and the size of the on-set of f is $2C$, the on-set can be partitioned into C distinct pairs of the input combinations of the form $(a_k,\dots,a_0)$ and $(\overline{a_k},\dots,\overline{a_0})$. By the definition, f is an ECF for the output probability $C/2^k$. ■

In summary, in order to reliably generate an output probability $C/2^k$, a solution is to use an ECF $f(r_k,\dots,r_0)$ for that probability, taking $k+1$ unbiased random bits as inputs.

C. Design Principle for Reliable Probability Conversion Circuits

In this section, we extend the design principle for a reliable fixed output probability to design reliable PCCs. Again, we focus on the Boolean function of a reliable PCC.

Based on the design principle for a reliable fixed output probability, a reliable PCC should take $k+1$ unbiased random bits $r_k,\dots,r_0$ as inputs. Besides, same as a traditional PCC, a reliable PCC should take k extra target bits $c_{k-1},\dots,c_0$ as inputs. Therefore, the Boolean function of a reliable PCC has $2k+1$ inputs. We assume it is $f(r_k,\dots,r_0,c_{k-1},\dots,c_0)$.

Now consider any fixed target bit combination $(a_{k-1},\dots,a_0)\in\{0,1\}^k$. By the basic function of a PCC, we require that the output probability of the function $f(r_k,\dots,r_0,a_{k-1},\dots,a_0)$ is $A/2^k$, where $A=(a_{k-1}\dots a_0)_2$. Note that given that $a_{k-1},\dots,a_0$ are all fixed, the function

$f(r_k, \dots, r_0, a_{k-1}, \dots, a_0)$ is a function that only depends on variables $r_k, \dots, r_0$. Indeed, the function $f(r_k, \dots, r_0, a_{k-1}, \dots, a_0)$ is just a *cofactor* of the function f, denoted as $f_{a_{k-1}\dots a_0}(r_k, \dots, r_0)$. Therefore, we require that the output probability of the cofactor of f, $f_{a_{k-1}\dots a_0}(r_k, \dots, r_0)$, be $A/2^k$, when $r_k, \dots, r_0$ are all unbiased random bits. Furthermore, we require the output probability of the cofactor $f_{a_{k-1}\dots a_0}(r_k, \dots, r_0)$ to be reliable when the input random bits are unreliable. Therefore, an immediate solution to design a reliable PCC is to let the cofactor of f, $f_{a_{k-1}\dots a_0}(r_k, \dots, r_0)$, be an ECF for the probability $A/2^k$, for all input combinations $(a_{k-1}, \dots, a_0) \in \{0,1\}^k$. To be formal, we first give the following definition.

Definition 1

A $(2k+1)$-input Boolean function $f(r_k, \dots, r_0, c_{k-1}, \dots, c_0)$ is an *error cancelling probability conversion function (ECPCF)* for a precision k if it satisfies the condition that for all $(a_{k-1}, \dots, a_0) \in \{0,1\}^k$, the cofactor $f_{a_{k-1}\dots a_0}(r_k, \dots, r_0)$ is an ECF for the probability $A/2^k$, where $A = (a_{k-1} \dots a_0)_2$. ■

Based on the above discussion, we conclude that one valid way to design a reliable PCC is to use an ECPCF.

Next, we show that there exists an important link between an ECPCF and the Boolean function of a traditional PCC. On the one hand, given an arbitrary traditional PCC, we can construct an ECPCF from the Boolean function of the traditional PCC. On the other hand, given an arbitrary ECPCF, we can extract from it the Boolean function of a traditional PCC. The above two claims are more formally presented by Theorems 2 and 3 below. For simplicity, these two theorems are described assuming $k = 2$.

Theorem 2

Given a traditional PCC with its Boolean function as $g(r_1, r_0, c_1, c_0)$, where r_1 and r_0 are the inputs for the random bits and c_1 and c_0 are the inputs for the target bits, then the function

$$f(r_2, r_1, r_0, c_1, c_0) = r_2 g(\overline{r_1}, \overline{r_0}, c_1, c_0) + \overline{r_2} g(r_1, r_0, c_1, c_0)$$

is an ECPCF. ■

Proof: By Definition 1, we only need to show that for all $(a_1, a_0) \in \{0,1\}^2$, the cofactor $f_{a_1a_0}(r_2, r_1, r_0)$ is an ECF for the probability $A/2^2$, where $A = (a_1a_0)_2$.

Consider any $(a_1, a_0) \in \{0,1\}^2$. We have

$$\begin{aligned} f_{a_1a_0}(r_2, r_1, r_0) &= f(r_2, r_1, r_0, a_1, a_0) \\ &= r_2 g(\overline{r_1}, \overline{r_0}, a_1, a_0) + \overline{r_2} g(r_1, r_0, a_1, a_0) \\ &= r_2 g_{a_1a_0}(\overline{r_1}, \overline{r_0}) + \overline{r_2} g_{a_1a_0}(r_1, r_0). \end{aligned}$$

Thus, we have

$$\begin{aligned} f_{a_1a_0}(\overline{r_2}, \overline{r_1}, \overline{r_0}) &= \overline{r_2} g_{a_1a_0}(r_1, r_0) + r_2 g_{a_1a_0}(\overline{r_1}, \overline{r_0}) \\ &= f_{a_1a_0}(r_2, r_1, r_0). \end{aligned}$$

Since $g(r_1, r_0, c_1, c_0)$ is the Boolean function of a traditional PCC, the size of the on-set of the cofactor $g_{a_1a_0}(r_1, r_0)$ is $A = (a_1a_0)_2$. Note that when $r_2 = 1$, $f_{a_1a_0}(r_2, r_1, r_0) = g_{a_1a_0}(\overline{r_1}, \overline{r_0})$; when $r_2 = 0$, $f_{a_1a_0}(r_2, r_1, r_0) = g_{a_1a_0}(r_1, r_0)$. Therefore, the size of the on-set of the function $f_{a_1a_0}(r_2, r_1, r_0)$ is $2A$. Thus, the output probability of the function $f_{a_1a_0}(r_2, r_1, r_0)$ is $A/2^2$. Based on Theorem 1, we conclude that $f_{a_1a_0}(r_2, r_1, r_0)$ is an ECF for the probability $A/2^2$. This finishes the proof. ■

Theorem 3

Given an ECPCF $f(r_2, r_1, r_0, c_1, c_0)$, where r_2, r_1, and r_0 are the inputs for the unbiased random bits and c_1 and c_0 are the inputs for the target bits, then the function

$$g(r_1, r_0, c_1, c_0) = f(0, r_1, r_0, c_1, c_0).$$

is the Boolean function of a traditional PCC of precision $k = 2$. ■

Proof: In order to prove the claim, we only need to show that for any $(a_1, a_0) \in \{0,1\}^2$, the output probability of the cofactor $g_{a_1a_0}(r_1, r_0) = g(r_1, r_0, a_1, a_0)$ is $A/2^2$, where $A = (a_1a_0)_2$.

Now, consider any $(a_1, a_0) \in \{0,1\}^2$. By the definition of an ECPCF, the cofactor $f_{a_1a_0}(r_2, r_1, r_0)$ of f is an ECF for the probability $A/2^2$. Given this, by Theorem 1, we have $f_{a_1a_0}(r_2, r_1, r_0) = f_{a_1a_0}(\overline{r_2}, \overline{r_1}, \overline{r_0})$ and the size of the on-set of $f_{a_1a_0}(r_2, r_1, r_0)$ is $2A$. Therefore, the size of the on-set of the function $f_{a_1a_0}(0, r_1, r_0)$ is A. Since $f_{a_1a_0}(0, r_1, r_0) = g_{a_1a_0}(r_1, r_0)$, the output probability of the function $g_{a_1a_0}(r_1, r_0)$ is $A/2^2$. This concludes the proof. ■

Theorem 2 shows a concrete way to construct a circuit for an ECPCF from a traditional PCC. An example of this for $k = 2$ is shown in Fig. 6. The circuit contains two copies of a traditional PCC, which can be a comparator, a WBG, or a MUX chain. One copy takes unbiased random bits r_1 and r_0 and the other takes their negations as the random input bits. The third unbiased random bit is used as the selection input to a MUX. Given that an ECPCF is a valid way to realize a reliable PCC, the circuit shown in Fig. 6 is a reliable PCC. Note that by our design, the unbiased random bit as the selection input to the MUX can also be unreliable.

The proposed design of a reliable PCC roughly doubles the area and power consumption of those of the underlying traditional PCC. However, as long as the area and power consumption of the RNS using emerging devices are much smaller than those of the LFSR, the overall area and power consumption of the SNG composed of emerging device-based RNS and the reliable PCC will still be smaller than those of the traditional CMOS-based SNG.

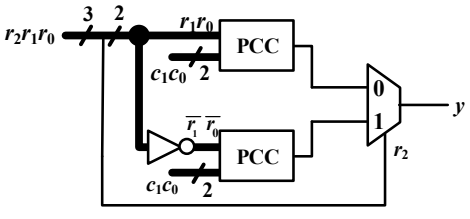

Fig. 6. Proposed design of a reliable probability conversion circuit.

IV. Experimental Results

In this section, we show the experimental results on our proposed reliable design of SNG.

A. *Reliability of the Output Probability*

We first study the reliability of the output probability when the input unbiased random bits are unreliable. To model these unreliable unbiased random bits, we assume that the probability of each is $0.5 + \varepsilon$, where ε is the noise. We assume ε is a Gaussian random variable with mean of 0 and standard deviation σ. For all the experiments in this section, the precision k is chosen as 8.

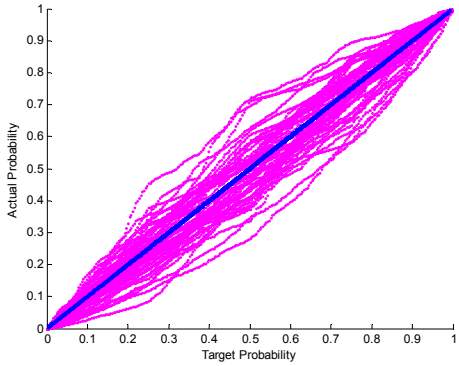

Fig. 7. The actual output probability generated by a comparator versus the target output probability.

Fig. 7 shows the curves of the actual output probability versus the target probability for a traditional PCC using a comparator. Fig. 8 shows the same curves for our proposed reliable PCC based on comparators. The standard deviation σ for each random bit is set as 0.1. In both figures, the blue curve, which is a straight line, indicates the expected relation between the actual output probability and the target output probability. There are also 50 purple curves in both figures. Each purple curve corresponds to the real relation between the actual output probability and the target output probability for a set of 8 random bits with randomly generated noises added to their expected values. Clearly, our proposed reliable PCC generates more accurate output probability than the traditional PCC for any target probability.

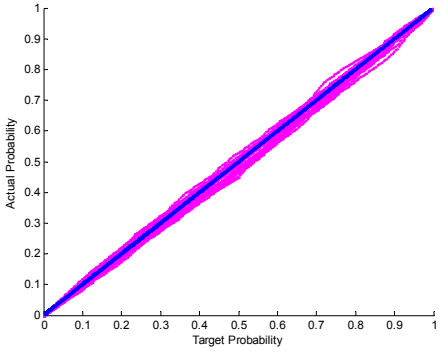

Fig. 8. The actual output probability generated by our proposed reliable comparator-based PCC versus the target output probability.

In Fig. 9, we vary the standard deviation of the injected noise from 0.025 to 0.25 and compare the average absolute error of the output probability for the traditional PCC and the reliable PCC. To obtain the average absolute error for each standard deviation σ, we generate 8000 sets of 8 random bits with the standard deviation of the injected noise as σ. For each set of fixed random bits, we obtain the average absolute error over all the target probabilities of 8-bit precision. The final absolute error is calculated as the average of these 8000 runs. It can be seen that the average absolute error increases with the noise level for both PCCs, which is expected. The proposed reliable PCC is much more accurate than the traditional PCC when the noise level is low. However, the slope of the curve for the reliable PCC increases with the noise level. This is because when the noise level is large, the higher-order errors caused by the noise in the random source are not ignorable, which causes the absolute error to increase faster. Nevertheless, the slope of the curve for the reliable PCC is always smaller than that for the traditional PCC, due to the ability of the reliable PCC to cancel the first-order errors induced by the noise in the random source.

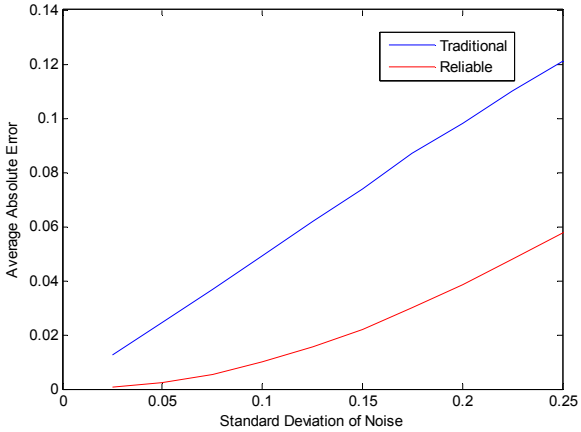

Fig. 9. The average absolute error of the output probability versus the standard deviation of the injected noise for the traditional PCC and our proposed reliable PCC.

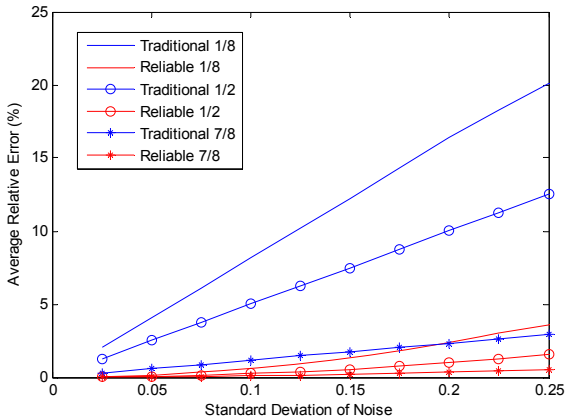

Fig. 10. The average relative error of the output probability versus the standard deviation of the injected noise for the traditional PCC and our proposed reliable PCC for three target probabilities, 1/8, 1/2, and 7/8.

In Fig. 10, we further show how the average relative error changes with the standard deviation of the injected noise for three specific target probabilities, 1/8, 1/2, and 7/8. Again, we vary the standard deviation from 0.025 to 0.25. To calculate the

relative error for a specific target probability and a standard deviation σ, we generate 8000 sets of 8 random bits with the standard deviation of the injected noise as σ. For each set of fixed random bits, we calculate the relative error for the specific target probability. The final average relative error is the average of these 8000 runs. It can be seen that for both PCCs, as the target probability decreases, the relative error increases, which is expected. For our proposed PCC, even if the target probability is as small as 1/8 and the noise level is as large as 0.25, the relative error is still within 4%. In contrast, for the traditional PCC, when the target probability is small, such as 1/8, the relative error could be more than 20% for a large noise level.

B. Area and Power Consumption

In this section, we study the area and power consumption of our proposed reliable PCC. Obviously, it increases the area and power consumption for the PCC part. However, with the small area and low power advantages of those emerging device-based random bits, it is possible that the entire area and power consumption of the reliable emerging device-based SNG (RESNG), which is built with the emerging random source and our reliable PCC, are still less than those of the traditional CMOS-based SNG (TCSNG), which is built with an LFSR and a traditional PCC. As our method is generally applicable to any emerging device-based random bits, in this section, we will obtain the limits on the area and power consumption for those random bits showing that our proposed SNG has smaller area and lower power consumption than the TCSNG.

We consider three PCCs, which are comparator, WBG, and MUX chain. For each PCC, we synthesize a TCSNG and a reliable PCC (RPCC) using that PCC. All the circuits are synthesized by Synopsys design compiler [17] and placed and routed by Cadence SoC encounter [18]. A 45nm Nangate open cell library is used as the technology library [19]. Precisions $k = 4,8,12,16$ are considered. Table III shows the simulation results of the area and power consumption of the TCSNG and the RPCC for each chosen PCC and precision k. The seventh column and the eighth column show the area budget and the power budget for one random bit with emerging device. If the area consumptions of the TCSNG and the RPCC are A and B, respectively, and the precision is k, then the area budget for the emerging device-based random source is $A - B$ and that for one single random bit is $(A - B)/(k + 1)$. Similarly, we can obtain the power budget for a single random bit.

From these results, we can clearly see that different types of PCC have different area and power budgets. For area budget, on average, using a MUX chain as the PCC gives the smallest area budget, while using a comparator as the PCC gives the largest one. For power consumption, using a WBG as the PCC gives the smallest power budget, while using a comparator as the PCC gives the largest one. Based on the preliminary data reported in some research works, we find that our proposed approach is very promising in reducing the area and power consumption of the SNGs. For example, the work [9] proposes an implementation of random bit using magnetic tunnel junction (MTJ) device. In this work, a 64×64 MTJ array-based multi-bit true random number generator is reported to have an area of 139.96 μm^2 with 45 nm technology node. Therefore, the area of a single MTJ-based random bit is 0.034 μm^2, which is much smaller than all the area budgets reported in Table III. Another work [8] estimated that a single memristor-based unbiased random bit consumes $5\mu W$, which is much smaller than some power budgets reported in Table III.

Table III. Area and power consumption comparison between the traditional PCC and our proposed reliable PCC.

PCC	Precision	Clock period (ns)	Circuit	Area (μm^2)	Power (μW)	Area budget for one random bit (μm^2)	Power budget for one random bit (μW)
Comparator	4	2.16	TCSNG	34.58	55.67	2.61	5.87
			RPCC	21.54	26.29		
	8	2.25	TCSNG	80.33	99.92	4.17	6.93
			RPCC	42.82	37.56		
	12	2.56	TCSNG	114.38	146.99	3.85	8.70
			RPCC	64.37	33.84		
	16	2.65	TCSNG	155.87	188.72	4.51	8.96
			RPCC	79.26	36.39		
WBG	4	2.44	TCSNG	34.04	63.14	2.39	2.77
			RPCC	22.07	49.29		
	8	2.70	TCSNG	71.28	107.00	3.58	6.16
			RPCC	39.10	51.54		
	12	3.10	TCSNG	110.92	70.82	3.74	1.01
			RPCC	62.24	57.70		
	16	3.38	TCSNG	145.23	92.10	3.71	1.02
			RPCC	82.19	74.78		
MUX Chain	4	2.52	TCSNG	31.65	65.17	2.13	5.30
			RPCC	21.01	38.69		
	8	3.35	TCSNG	65.70	92.97	3.31	3.92
			RPCC	35.91	57.68		
	12	4.18	TCSNG	96.55	139.70	3.52	6.42
			RPCC	50.80	56.20		
	16	5.00	TCSNG	127.41	164.55	3.63	6.83
			RPCC	65.70	48.49		

V. CONCLUSION

Due to their small area and low power consumption, emerging device-based random sources are a promising choice for building the stochastic number generators (SNGs) for stochastic computing. However, one challenge to use them to build SNGs is that the output probability of the SNG is not reliable, due to the noise in the input control signal and the large process variation for these emerging devices. In this work, we propose a general method to design reliable SNGs with these emerging devices-based random sources. The key is to build a reliable probability conversion circuit (PCC). We propose general principles to design such a PCC. Based on these principles, we find that a reliable PCC can be easily constructed from a traditional PCC. Experimental results showed that our proposed PCC generates much more accurate output probability than the traditional PCC. We also evaluated area and power consumption of the proposed PCC, and obtained the area and power budget for the random sources used in the SNG, which can be used as a guideline to

design novel emerging device-based random sources for stochastic computing.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61472243 and 61204042.

REFERENCES

[1] B.R. Gaines, "Stochastic Computing," *AFIPS Spring Joint Computer Conf.*, 1967, pp. 149-156.

[2] A. Alaghi, C. Li, and J.P. Hayes, "Stochastic circuits for real-time image-processing applications," *Design Autom. Conf.,* 2013, pp. 136:1-136:6.

[3] V.C. Gaudet and A.C. Rapley, "Iterative Decoding Using Stochastic Computation," *Electron. Lett.*, vol. 39, pp. 299-301, 2003.

[4] B.D. Brown and H.C. Card, "Stochastic neural computation I: computational elements," *IEEE Trans. Comp.*, vol. 50, pp. 891-905, 2001.

[5] N. Saraf, K. Bazargan, D. J. Lilja and M. D. Riedel, "IIR filters using stochastic arithmetic," *Design, Autom. & Test in Europe,* 2014, pp. 1-6.

[6] B. Li, M. H. Najafi, and D. J. Lilja, "Using stochastic computing to reduce the hardware requirements for a restricted Boltzmann machine classifier," *Intl. Symp. on FPGA*, 2016, pp. 36-41.

[7] P. Li et al., "Computation on stochastic bit streams: Digital image processing case studies,". *IEEE Trans. on VLSI,* vol. 22, pp. 449-462, 2014.

[8] P. Knag, W. Lu, and Z. Zhang, "A native stochastic computing architecture enabled by memristors," *IEEE Trans. Nanotech.*, vol. 13, pp. 283-293, 2014.

[9] H. Lee et al., "Design of high-throughput and low-power true random number generator utilizing perpendicularly magnetized voltage-controlled magnetic tunnel junction," *AIP Advances*, vol. 7, no. 5, 055934, 2017.

[10] Y. Wang et al., "A novel true random number generator design leveraging emerging memristor technology," *Great Lakes Symp. on VLSI*, 2015, pp. 271-276.

[11] A. Fukushima et al., "Spin dice: A scalable truly random number generator based on spintronics," *Applied Physics Express*, vol. 7, no. 8, pp. 083001, 2014.

[12] R. Venkatesan et al., "Spintastic: spin-based stochastic logic for energy-efficient computing," *Design, Autom. & Test in Europe,* 2015, pp. 1575-1578.

[13] P. K. Gupta and R. Kumaresan, "Binary multiplication with PN sequences," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 36, no. 4, pp. 603-606, 1988.

[14] H. Ichihara et al., "Compact and accurate stochastic circuits with shared random number sources," *Intl. Conf. Comp. Design,* 2014, pp. 361-366.

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

[16] W. Qian and M.D. Riedel, "Two-level logic synthesis for probabilistic computation," in *Intl. Workshop on Logic & Synthesis,* 2010, pp. 95-102.

[17] Synopsys Inc., http://www. synopsys. com.

[18] Cadence Inc., http://www.cadence.com.

[19] Nangate Inc., http://www.nangate.com