Accuracy Analysis for Stochastic Circuits with D Flip-Flop Insertion

Kuncai Zhong[1], Weikang Qian[1,2*]
[1]University of Michigan-Shanghai Jiao Tong University Joint Institute, [2]MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University, Shanghai, China (*email: qianwk@sjtu.edu.cn)

Abstract—**One of the challenges stochastic computing (SC) faces is the high cost of stochastic number generators (SNG). A solution to it is inserting D flip-flops (DFFs) into the circuit. However, the accuracy of the stochastic circuits would be affected and it is crucial to capture it. In this work, we propose an efficient method to analyze the accuracy of stochastic circuits with DFFs inserted. Furthermore, given the importance of multiplication, we apply this method to analyze stochastic multiplier with DFFs inserted. Several interesting claims are obtained about the use of probability conversion circuits. For example, using weighted binary generator is more accurate than using comparator. The experimental results show the correctness of the proposed method and the claims. Furthermore, the proposed method is up to $560\times$ faster than the simulation-based method.**

I. Introduction

As an unconventional computing paradigm, stochastic computing (SC) has drawn more attention recently. It uses digital circuits to compute on *stochastic bit streams (SBSs)*, which encode values through the ratios of 1s in the streams [1]. Compared to the conventional binary computing, SC has significant advantages in circuit area and fault tolerance due to its probabilistic nature. For example, with a single AND gate, it can implement multiplication. Such a circuit taking SBSs as inputs and outputs is called an *SC core* in this paper. With the features of low circuit area and strong fault tolerance, it has been successfully applied in image processing [2], digital filter design [3], and neural network [4].

SC needs a component called *stochastic number generators (SNGs)* to convert a binary number into an SBS as shown in Fig. 1. Generally, SNG is composed of a random number source (RNS) and a probability conversion circuit (PCC) [5]. RNS generates random numbers R and is usually implemented by a linear feedback shift register (LFSR). PCC converts an input binary number X into an SBS with the random numbers R. A typical choice of PCC is a comparator (CMP). In addition, there are some other PCCs, such as weighted binary generator (WBG) [6].

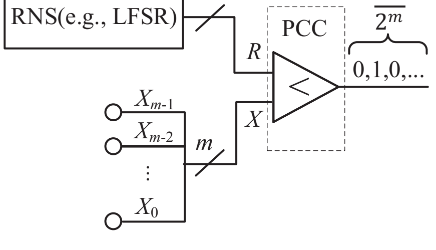

Fig. 1: The general architecture of an SNG [7].

In general, SNGs occupy most of the area of an SC circuit. Therefore, in order to reduce the area of an SC circuit, it is vital to reduce the area of SNGs. Many strategies have been proposed, such as sharing RNSs among SNGs [3] and sharing PCCs among SNGs [7]. In this work, we focus on another effective way, inserting D flip-flops (DFFs) into an SC circuit [1]. This is illustrated in Fig. 2. Fig. 2(a) shows a *basic SC multiplier* that uses two SNGs to generate two independent SBSs. However, we can reduce one costly RNS by inserting a few DFFs, as shown in Fig. 2(b). For example, with 1 DFF inserted, the area of the SC multiplier can be reduced by 28.5% [8]. In terms of the *expectation* of the value encoded by the output SBS, the circuit still correctly implements multiplication. This is because the random binary numbers generated by the RNS at different clock cycles are independent. Therefore, at each clock cycle, the two input bits of the AND gate are independent and hence, the probability of each bit in the output stream still equals the product of the two input probabilities.

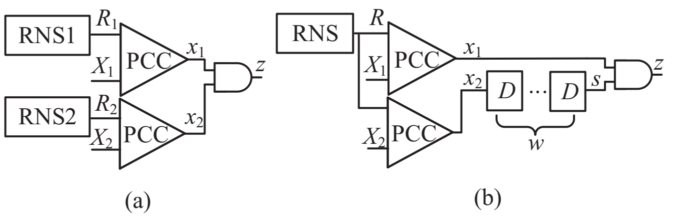

Fig. 2: (a) The basic SC multiplier with 2 independent SNGs; (b) The SC multiplier with DFF insertion.

Nevertheless, DFF insertion affects the computation accuracy, because output bits at different clock cycles become correlated. For example, for the SC multiplier in Fig. 2(b), with the bit-width of the input binary numbers as 8 and the length of SBS as 256, when only 1 DFF is inserted, the mean square error (MSE) of the output is 6.99×10^{-4}, higher than that of the basic SC multiplier, which is 5.49×10^{-4}. Therefore, it is very important to get a better understanding on the trade-off between hardware cost reduction and accuracy reduction for the DFF insertion technique, which can help us design SC circuits with a proper balance among accuracy, area, and energy consumption [9]. Thus, it is critical to analyze the accuracy of an SC circuit with DFF insertion. In the following, we refer to such a circuit as *SC-DFF* for short.

A few recent works considered analyzing the accuracy of SC-DFFs. Chen and Hayes analyzed the accuracy of SC-DFFs for power-form expressions [10]. Neugebauer *et al.* further proposed a method based on time-frame expansion to analyze the accuracy of arbitrary SC-DFFs [9]. However, this method fails to consider the impact of PCCs to the accuracy of SC-DFF, which we find to be an important factor that cannot be ignored. Furthermore, it relies on existing symbolic algebra systems to generate the accuracy expression, which may miss the opportunity of applying some domain-specific properties to speed up the computation.

To solve the above challenges, in this work, we propose a novel systematic method to analyze the accuracy of SC-DFFs. Since multiplication is an important operation in many applications, we further apply this method to analyze the stochastic multiplier shown in Fig. 2(b), which we refer to as *MUL-DFF*. We obtain several interesting claims. For example, MUL-DFF using CMPs as the PCCs is less accurate than that using WBGs.

In summary, we make the following contributions.

- We propose a method to analyze the accuracy of SC-DFFs. Our method exploits a few mathematical properties of SC-DFF to accelerate the analysis (see Section II).
- For the first time, we reveal that PCCs can affect the accuracy of SC-DFFs and our proposed method can characterize their impact (see Sections II-C and III).
- We apply our method to analyze the MUL-DFF and obtain some interesting claims (see Section III).

II. Accuracy Analysis Method for SC-DFF

In this section, we present a systematic method for analyzing the accuracy of SC-DFFs.

A. Model, Assumptions, and Notations

The general model of the SC-DFF we consider in this work is shown in Fig. 3. We make the following assumptions.

1) The circuit has just one RNS.
2) The PCCs of the circuit are all of the same type.
3) Before DFFs are inserted, the SC core is a combinational circuit.
4) The random binary numbers produced by the RNS at different clock cycles are independent and uniformly distributed.

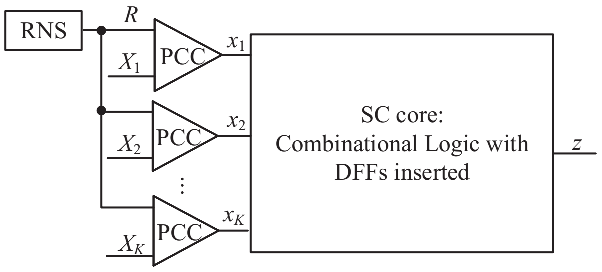

Fig. 3: General SC-DFF model.

Assumption 1 leads to the minimum RNS cost, which is usually desired. Assumption 2 follows the current design practice of an SC circuit. Furthermore, by making all the PCCs the same, it is possible to share some common parts of them and reduce the circuit area [7]. In general, an SC core can either be combinational or sequential. However, combinational design covers a widely-used class of SC core [11], [12]. Thus, we focus on combinational SC core here and we have Assumption 3. Assumption 4 is a mathematical model to simplify our analysis. Given Assumption 4, we have:

Claim 1 *The bits of any input SBS's at different clock cycles are independent. Moreover, the bits of the same input SBS at different clock cycles are independent and identically distributed.*

The following notations are used in the paper.

- We assume the number of binary inputs of an SC-DFF is K and they are $X_1, \dots, X_K$ (see Fig. 3). The bit-width of each binary input is m. We denote the jth ($0 \le j \le m-1$) least significant bit of X_i ($1 \le i \le K$) as $X_{i,j}$.
- We denote the random binary number produced by the RNS at clock cycle i as $R[i]$, which is also with m bits.
- We assume the SBS length is n. We denote the input SBS produced by the PCC over X_i as x_i and the output SBS as z. The jth bits in the streams x_i and z are denoted as $x_i[j]$ and $z[j]$, respectively.
- We use $Pr(A)$ to denote the probability of an event A.
- We let $p_{x_i} = X_i/2^m$. We have $Pr(x_i[j] = 1) = p_{x_i}$.
- We denote the accurate output as f and the output by the SC circuit as $\mathcal{Z}$. The value f is known. By the encoding rule of SC, we have

$$\mathcal{Z} = \frac{1}{n}\sum_{i=1}^{n} z[i]. \tag{1}$$

Since $z[i]$'s are random, the value $\mathcal{Z}$ is also a random variable. Thus, the output error $\mathcal{Z} - f$ is also a random variable. To capture how large it is on average, we consider the mean square error (MSE), i.e., $E[(\mathcal{Z} - f)^2]$. We have

$$E[(\mathcal{Z} - f)^2] = E[\mathcal{Z}^2] - 2E[\mathcal{Z}]f + f^2. \tag{2}$$

Given Eq. (2), in order to calculate the MSE, we need to obtain $E[\mathcal{Z}]$ and $E[\mathcal{Z}^2]$. In the following, we will first show how to derive the expression for the output $\mathcal{Z}$. Then, we will show how to calculate $E[\mathcal{Z}]$ and $E[\mathcal{Z}^2]$.

B. Output Expression of an SC-DFF

For later calculation of $E[\mathcal{Z}]$ and $E[\mathcal{Z}^2]$, we need to represent $\mathcal{Z}$ as an arithmetic expression of the input bits $x_j[k]$'s. By Eq. (1), in order to obtain the arithmetic expression, we only need to represent $z[i]$'s as an arithmetic expression of $x_j[k]$'s.

We apply the method proposed in [8] to obtain the expression for a $z[i]$. The method applies the arithmetic function of each gate in the circuit in a topological sorting order. The arithmetic function for a logic gate can be easily derived from the truth table of the gate. For example, the function for an inverter is $z(x) = 1-x$, while that for a 2-input AND gate is $z(x, y) = xy$. Furthermore, an SC-DFF has some DFFs inserted. Each time a DFF is visited, we should subtract all clock cycles in the input expression by 1. Note that since $x_j[i]$ can only be 0 or 1, we also have $(x_j[i])^k = x_j[i]$. By applying these rules, we can eventually derive the expression of $\mathcal{Z}$.

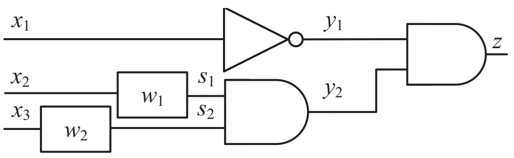

Fig. 4: A simple example of SC-DFF.

Example 1 *Consider the circuit shown in Fig. 4, where w_1 and w_2 indicate that there are w_1 and w_2 DFFs inserted at the corresponding places. By applying the arithmetic function for* *each gate in a topological sorting order and considering the effect of DFFs, we can obtain*

$$\begin{aligned}
&s_1[i] = x_2[i-w_1], \quad s_2[i] = x_3[i-w_2],\\
&y_1[i] = 1 - x_1[i], \quad y_2[i] = s_1[i]s_2[i] = x_2[i-w_1]x_3[i-w_2],\\
&z[i] = y_1[i]y_2[i] = (1-x_1[i])x_2[i-w_1]x_3[i-w_2]\\
&\quad = x_2[i-w_1]x_3[i-w_2] - x_1[i]x_2[i-w_1]x_3[i-w_2].
\end{aligned}$$

C. Expectation of the Output Value

In this section, we show how to calculate $E[\mathcal{Z}]$. Given Eq. (1), we have

$$E[\mathcal{Z}] = \frac{1}{n}\sum_{i=1}^{n} E[z[i]]. \tag{3}$$

First, based on Claim 1, we can prove that the probability mass distributions of the random variables $z[1]$ and $z[i]$ are the same. Thus, we have the following claim.

Claim 2 *For any i, we have $E[z[i]] = E[z[1]]$.*

Combining the above claim with Eq. (3), we have

$$E[\mathcal{Z}] = E[z[i]] = E[z[1]]. \tag{4}$$

Next, we show how to calculate $E[z[1]]$.

As Example 1 shows, $z[1]$ is a sum of products. Assume $z[1]$ consists of M products $H_1, \ldots, H_M$. By the linearity of expectation, we have $E[z[1]] = \sum_{i=1}^{M} E[H_i]$. Thus, the problem of obtaining $E[z[1]]$ reduces to obtaining each $E[H_i]$. Next, we show how to calculate it.

Assume a product term H can be represented as $H = cy_1[v_1]y_2[v_2]\cdots y_r[v_r]$, where c is a constant coefficient and for all $1 \le i \le r$, $y_i \in \{x_1, \ldots, x_K\}$ and v_i indicates the clock cycle. We call $y_i[v_i]$ *a factor* of the product. We partition all the factors $y_i[v_i]$'s into a number of subsets based on the clock cycles of the factors. Assume there are L distinct clock cycles among all the factors. Then, we will obtain L subsets $S_1, \ldots, S_L$. We denote the number of factors in subset S_i as d_i and the kth factor in S_i as S_i^k. Given that the factors with different clock cycles are independent, we have

$$E[H] = c\prod_{i=1}^{L} E[S_i^1 S_i^2 \cdots S_i^{d_i}]. \tag{5}$$

Example 2 *Consider $z[1]$ derived in Example 1. Assume that $w_1 = 0$ and $w_2 = 1$. Then, we have $z[1] = x_2[1]x_3[0] - x_1[1]x_2[1]x_3[0]$. It includes two products, $H_1 = x_2[1]x_3[0]$ and $H_2 = -x_1[1]x_2[1]x_3[0]$. Then, $E[z[1]] = E[H_1] + E[H_2]$.*

We further consider calculating $E[H_2]$ as an example. H_2 has 3 factors. They can be partitioned into 2 subsets, $S_1 = \{x_1[1], x_2[1]\}$ and $S_2 = \{x_3[0]\}$. Given that the bits at different clock cycles are independent, we have $E[H_2] = -E[x_1[1]x_2[1]]E[x_3[0]]$.

Given Eq. (5), the problem reduces to calculating each term $E[S_i^1 S_i^2 \cdots S_i^{d_i}]$. Since each factor S_i^k is either 0 or 1, the product $S_i^1 S_i^2 \cdots S_i^{d_i}$ is either 0 or 1. Thus, we have

$$E[S_i^1 S_i^2 \cdots S_i^{d_i}] = Pr(S_i^1 S_i^2 \cdots S_i^{d_i} = 1).$$

Assume the clock cycle of the factor S_i^k $(1 \le k \le d_i)$ is b and its corresponding input binary number is Q_i^k. In other words, the random bit S_i^k is produced by the PCC over the random number $R[b]$ and the input binary number Q_i^k. Based on the choice of the PCCs, we can further obtain the value $Pr(S_i^1 S_i^2 \cdots S_i^{d_i} = 1)$. We consider two types of PCCs in this paper, CMP and WBG.

CMP compares a random binary number R generated by the RNS with an input binary number X. It outputs a 1 if $R < X$ and 0 otherwise. With CMP as the PCC, $S_i^1 S_i^2 \cdots S_i^{d_i} = 1$ if and only if $R[b]$ is less than all Q_i^k's, which means $R[b] < \min\{Q_i^1, Q_i^2, \ldots, Q_i^{d_i}\}$. Given that $R[b]$ is uniformly distributed in the set $\{0, 1, \ldots, 2^m - 1\}$, we can obtain that

$$\begin{aligned}
&Pr(S_i^1 \cdots S_i^{d_i} = 1) = Pr(R[b] < \min\{Q_i^1, \ldots, Q_i^{d_i}\})\\
&= \min\left\{\frac{Q_i^1}{2^m}, \frac{Q_i^2}{2^m}, \ldots, \frac{Q_i^{d_i}}{2^m}\right\}.
\end{aligned} \tag{6}$$

WBG is another type of PCC. It first transforms the random binary number R produced by the RNS into m random bits of probabilities $\frac{1}{2}, \frac{1}{2^2}, \ldots, \frac{1}{2^m}$ to be a 1 and then applies a tree of AND and OR gates on these bits and the input binary number X to produce the final target probability [6]. In our case, for each $1 \le k \le d_i$, the bit S_i^k is produced by a WBG over the random binary number $R[b]$ and the input binary number Q_i^k. We denote the jth $(0 \le j \le m-1)$ least significant bit of $R[b]$ and Q_i^k as $R_j[b]$ and $Q_{i,j}^k$, respectively. Then, by the Boolean function of WBG, we have

$$\begin{aligned}
S_i^k = &(R_{m-1}[b] \wedge Q_{i,m-1}^k) \vee (\overline{R_{m-1}[b]} \wedge R_{m-2}[b] \wedge Q_{i,m-2}^k)\\
&\vee \cdots \vee (\overline{R_{m-1}[b]} \wedge \overline{R_{m-2}[b]} \wedge \cdots \wedge \overline{R_1[b]} \wedge R_0[b] \wedge Q_{i,0}^k),
\end{aligned}$$

where $\wedge$ and $\vee$ represent logical AND and OR, respectively. By the above equation, we can further obtain

$$\begin{aligned}
&S_i^1 \cdots S_i^{d_i} = (R_{m-1}[b] \wedge Q_{i,m-1}^1 \wedge Q_{i,m-1}^2 \wedge \cdots \wedge Q_{i,m-1}^{d_i})\\
&\vee (\overline{R_{m-1}[b]} \wedge R_{m-2}[b] \wedge Q_{i,m-2}^1 \wedge Q_{i,m-2}^2 \wedge \cdots \wedge Q_{i,m-2}^{d_i})\\
&\vee \cdots\\
&\vee (\overline{R_{m-1}[b]} \wedge \cdots \wedge \overline{R_1[b]} \wedge R_0[b] \wedge Q_{i,0}^1 \wedge Q_{i,0}^2 \wedge \cdots \wedge Q_{i,0}^{d_i}).
\end{aligned}$$

Since the random bits $R_0[b], \ldots, R_{m-1}[b]$ are independent and have probability 0.5 to be a 1, we can further derive that

$$\begin{aligned}
&Pr(S_i^1 \cdots S_i^{d_i} = 1) = \frac{1}{2}Q_{i,m-1}^1 Q_{i,m-1}^2 \cdots Q_{i,m-1}^{d_i}\\
&+ \frac{1}{2^2}Q_{i,m-2}^1 \cdots Q_{i,m-2}^{d_i} + \cdots + \frac{1}{2^m}Q_{i,0}^1 \cdots Q_{i,0}^{d_i}.
\end{aligned} \tag{7}$$

From Eqs. (6) and (7), it is obvious that $Pr(S_i^1 \cdots S_i^{d_i} = 1)$ is affected by the PCC choice. Therefore, $E[\mathcal{Z}]$ is also affected by it. Since the MSE calculation depends on $E[\mathcal{Z}]$, we can conclude that the MSE of an SC-DFF is affected by the PCC choice. An concrete example on this will be shown in Section III using the MUL-DFF.

D. Expectation of the Output Square

In this section, we will show how to calculate $E[\mathcal{Z}^2]$. Given that $E[\mathcal{Z}^2] = \frac{1}{n^2}E[(n\mathcal{Z})^2]$, we focus on calculating $E[(n\mathcal{Z})^2]$.

By Eq. (1), we have

$$E[(n\mathcal{Z})^2] = E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} z[i]z[j]\right] = \sum_{i=1}^{n}\sum_{j=1}^{n} E\left[z[i]z[j]\right].$$

The amount of computation by the above equation is $\Theta(n^2)$. Next, we derive a more efficient way to compute $E[(n\mathcal{Z})^2]$. By reorganizing the above double sum diagonally and observing that $E[z[i]z[j]] = E[z[j]z[i]]$, we further have

$$E[(n\mathcal{Z})^2] = \sum_{i=1}^{n} E[z[i]^2] + 2\sum_{d=1}^{n-1}\sum_{i=1}^{n-d} E[z[i]z[i+d]]. \quad (8)$$

Given that $z[i]$ is either 0 or 1, we have $z[i]^2 = z[i]$. Combining this with Eq. (3), we can simplify the first term in Eq. (8) as

$$\sum_{i=1}^{n} E[z[i]^2] = \sum_{i=1}^{n} E[z[i]] = nE[\mathcal{Z}]. \quad (9)$$

To calculate the second term in Eq. (8), we first have the following claim based on Claim 1.

Claim 3 *For any $1 \le d \le n-1$ and $2 \le i \le n-d$, we have $E[z[1]z[1+d]] = E[z[i]z[i+d]]$.*

By Claim 3, the second term in Eq. (8) can be simplified as

$$2\sum_{d=1}^{n-1}\sum_{i=1}^{n-d} E[z[i]z[i+d]] = 2\sum_{d=1}^{n-1}(n-d)E[z[1]z[1+d]].$$

Assume that the maximum number of DFFs along any path from an input to the output of the given SC-DFF is $t \ge 0$. It can be proved that when $d > t$, the random variables $z[1]$ and $z[1+d]$ are independent. Combining this fact with Eq. (4), we have the following claim.

Claim 4 *For any $t < d < n$, $E[z[1]z[1+d]] = E[\mathcal{Z}]^2$.*

By Claim 4, the second term in Eq. (8) can be further simplified as

$$\begin{aligned}2\sum_{d=1}^{n-1}\sum_{i=1}^{n-d} E[z[i]z[i+d]] &= 2\sum_{d=1}^{t}(n-d)E[z[1]z[1+d]] \\ &+ (n-t)(n-t-1)E[\mathcal{Z}]^2.\end{aligned} \quad (10)$$

Given Eqs. (9) and (10), we can finally reduce Eq. (8) to

$$\begin{aligned}E[(n\mathcal{Z})^2] &= nE[\mathcal{Z}] + (n-t)(n-t-1)E[\mathcal{Z}]^2 \\ &+ 2\sum_{d=1}^{t}(n-d)E[z[1]z[1+d]].\end{aligned} \quad (11)$$

In order to eventually obtain $E[(n\mathcal{Z})^2]$, we only need to obtain $E[z[1]z[1+d]]$ for $d = 1, \ldots, t$. We define $s[1] = z[1]z[1+d]$. Then, we can just apply the method described in Section II-C for obtaining $E[z[1]]$ to obtain $E[s[1]]$, which gives $E[z[1]z[1+d]]$.

Typically, in order to minimize the circuit area, the value t is much less than the SBS length n. From Eq. (11), we can see that by using the properties specified in Claims 3 and 4, we significantly reduce the amount of computation to get $E[(n\mathcal{Z})^2]$, i.e., from a function quadratic on n to one independent of n.

E. The Accuracy Analysis Flow

In this section, we summarize the flow of the accuracy analysis method for a general SC-DFF as follows.

1) Derive the expression for the output $\mathcal{Z}$ by the method described in Section II-B.
2) Calculate $E[\mathcal{Z}]$ by the method described in Section II-C.
3) Calculate $E[\mathcal{Z}^2]$ by the method described in Section II-D.
4) Obtain the MSE of the SC-DFF by Eq. (2).

III. Application: Accuracy Analysis for MUL-DFF

In this section, as an application, we will apply the proposed accuracy analysis method to analyze the accuracy of the MUL-DFF shown in Fig. 2(b). We assume $w \ge 1$ DFFs are inserted at the second input of the AND gate. The same notations used in Section II are adopted.

A. Accuracy Analysis

For the MUL-DFF, we have $z[i] = x_1[i]x_2[i-w]$, for any i. Then, based on Eq. (1), the expression for the output $\mathcal{Z}$ is

$$\mathcal{Z} = \frac{1}{n}\sum_{i=1}^{n} z[i] = \frac{1}{n}\sum_{i=1}^{n} x_1[i]x_2[i-w]. \quad (12)$$

We now derive $E[\mathcal{Z}]$. By Eq. (4), $E[\mathcal{Z}] = E[z[1]] = E[x_1[1]x_2[1-w]]$. Since $w \ge 1$, we further have

$$E[\mathcal{Z}] = E[x_1[1]]E[x_2[1-w]] = p_{x_1}p_{x_2}. \quad (13)$$

Thus, the expectation of the output of the MUL-DFF equals the expected value, the product of the two input probabilities.

We now derive $E[\mathcal{Z}^2]$. By the procedure shown in Section II-D, we first derive $E[(n\mathcal{Z})^2]$ by Eq. (11). The value t in Eq. (11) is the maximum number of DFFs along any path from an input to the output of the MUL-DFF. Clearly, $t = w$. Thus, we have

$$\begin{aligned}E[(n\mathcal{Z})^2] &= np_{x_1}p_{x_2} + (n-w)(n-w-1)p_{x_1}^2p_{x_2}^2 \\ &+ 2\sum_{d=1}^{w}(n-d)E[x_1[1]x_2[1-w]x_1[1+d]x_2[1+d-w]].\end{aligned} \quad (14)$$

The remaining problem is to obtain $E[x_1[1]x_2[1-w]x_1[1+d]x_2[1+d-w]]$ for any $1 \le d \le w$. As we stated in Section II-D, this can be obtained by the method shown in Section II-C. For any $1 \le d < w$, we have $1-w < 1+d-w < 1 < 1+d$. Thus, the four factors $x_1[1]$, $x_2[1-w]$, $x_1[1+d]$, and $x_2[1+d-w]$ have different clock cycles and they are independent. Therefore,

$$E[x_1[1]x_2[1-w]x_1[1+d]x_2[1+d-w]] = p_{x_1}^2p_{x_2}^2. \quad (15)$$

When $d = w$, we have $1-w < 1+d-w = 1 < 1+d$. Thus, the four factors can be partitioned into three subsets of different clock cycles, $\{x_2[1-w]\}$, $\{x_1[1+d]\}$, and $\{x_1[1], x_2[1]\}$. In this case, we have

$$\begin{aligned}&E[x_1[1]x_2[1-w]x_1[1+d]x_2[1+d-w]] \\ &= E[x_2[1-w]]E[x_1[1+d]]E[x_1[1]x_2[1]] \\ &= p_{x_1}p_{x_2}Pr(x_1[1]x_2[1]=1).\end{aligned} \quad (16)$$

By Eqs. (14), (15), and (16), we can get

$$\begin{aligned}E[(n\mathcal{Z})^2] &= np_{x_1}p_{x_2} + (n^2-3n+2w)p_{x_1}^2p_{x_2}^2 \\ &+ 2(n-w)(p_{x_1}p_{x_2}Pr(x_1[1]x_2[1]=1)).\end{aligned} \quad (17)$$

Note that $E[\mathcal{Z}^2] = \frac{1}{n^2}E[(n\mathcal{Z})^2]$. For MUL-DFF, the accurate output is $f = p_{x_1}p_{x_2}$. By Eqs. (2), (13), and (17), the MSE of the MUL-DFF can be eventually derived as

$$\begin{aligned}MSE &= \frac{1}{n}(p_{x_1}p_{x_2})(1-p_{x_1}p_{x_2}) \\ &+ \frac{2(n-w)}{n^2}p_{x_1}p_{x_2}(Pr(x_1[1]x_2[1]=1) - p_{x_1}p_{x_2}).\end{aligned} \quad (18)$$

As shown in Section II-C, the value $Pr(x_1[1]x_2[1]=1)$ is determined by the choice of the PCCs. We will further evaluate the effect of the PCCs on the MSE in the next section.

B. Claims for the Accuracy of MUL-DFF

In this section, we derive several claims from the MSE expression for MUL-DFF shown in Eq. (18).

We use the MSE of a basic SC multiplier, which is shown in Fig. 2(a), as the reference. For this multiplier, the two input SBSs to the AND gate are generated by two independent SNGs. Its MSE can be calculated as

$$MSE = \frac{1}{n}(p_{x_1}p_{x_2})(1 - p_{x_1}p_{x_2}). \tag{19}$$

Comparing Eq. (19) with Eq. (18), we can see that with DFFs inserted, the MSE of the MUL-DFF is increased by

$$\Delta = \frac{2(n-w)}{n^2}p_{x_1}p_{x_2}(Pr(x_1[1]x_2[1] = 1) - p_{x_1}p_{x_2}). \tag{20}$$

Clearly, Δ decreases with the number of inserted DFFs w. Furthermore, since $Pr(x_1[1]x_2[1] = 1)$ is affected by the PCC choice, as stated in Section II-C, Δ is also affected by it. We further consider two PCC choices, CMP and WBG.

When PCCs are CMP, by Eq. (6), we have $Pr(x_1[1]x_2[1] = 1) = \min\{p_{x_1}, p_{x_2}\}$. It is easy to see that $\min\{p_{x_1}, p_{x_2}\} \geq p_{x_1}p_{x_2}$. Thus, by Eq. (20), we have $\Delta \geq 0$. We can conclude the following claim.

Claim 5 *For any inputs X_1 and X_2, the output MSE of an MUL-DFF with CMP as the PCCs is no less than that of a basic SC multiplier.*

When PCCs are WBG, by Eq. (7), we have

$$Pr(x_1[1]x_2[1] = 1) = \sum_{i=1}^{m}\frac{1}{2^i}X_{1,m-i}X_{2,m-i}. \tag{21}$$

Given Eq. (21), we can see that for some input binary numbers $X_1 > 0$ and $X_2 > 0$, it is possible that $P(x_1[1]x_2[1] = 1) = 0$. For example, consider $m = 3$, $X_1 = (011)_2 > 0$, $X_2 = (100)_2 > 0$. Then, by Eq. (21), we have $Pr(x_1[1]x_2[1] = 1) = 0$. For this pair of X_1 and X_2, since $p_{x_1} = X_1/2^m > 0$ and $p_{x_2} = X_2/2^m > 0$, we have $p_{x_1}p_{x_2} > 0$. Thus, by Eq. (20), we have $\Delta < 0$. Therefore, we have the following claim.

Claim 6 *For some inputs X_1 and X_2, the output MSE of an MUL-DFF with WBG as the PCCs is less than that of a basic SC multiplier.*

This is a surprising result. It indicates that performing multiplication with an MUL-DFF that uses WBG as the PCC, the accuracy sometimes can be improved.

Finally, we compare the MSEs between the MUL-DFF using CMP as the PCCs and that using WBG as the PCCs. To make distinction, we denote the $Pr(x_1[1]x_2[1] = 1)$ value for the former and the latter as P_{CMP} and P_{WBG}, respectively.

For all $0 \leq i \leq m - 1$, since $X_{1,i}, X_{2,i} \in \{0, 1\}$, we have $X_{1,i}X_{2,i} \leq X_{1,i}$. Thus, by Eq. (21), we have

$$P_{WBG} = \sum_{i=1}^{m}\frac{1}{2^i}X_{1,m-i}X_{2,m-i} \leq \sum_{i=1}^{m}\frac{1}{2^i}X_{1,m-i} = p_{x_1}.$$

Similarly, we have $P_{WBG} \leq p_{x_2}$. Since $P_{CMP} = \min\{p_{x_1}, p_{x_2}\}$, we can further obtain that

$$P_{WBG} \leq \min\{p_{x_1}, p_{x_2}\} = P_{CMP}. \tag{22}$$

By Eqs. (18) and (22), we can obtain the following claim.

Claim 7 *For any inputs X_1 and X_2, the output MSE of an MUL-DFF with WBG as the PCCs is no more than that of an MUL-DFF using CMP as the PCCs.*

This is another surprising result, which shows that using WBG as the PCCs has better accuracy than using CMP as the PCCs for MUL-DFFs.

IV. Experimental Results

In this section, we present the experimental results to verify the correctness and efficiency of the proposed accuracy analysis method and the correctness of several claims on MUL-DFF described in Section III-B. We implemented the algorithms in C++. All the experiments were tested on a computer with 2.50GHz CPU and 8GB RAM. We used the random number function in C++ to simulate the RNS. In the following, we call our method *the analysis method*. In order to verify the correctness of the analysis method, we compared it with the simulation-based method. Since MSE is a statistical value, to obtain the output MSE for a set of inputs, the simulation method repeats the basic simulation of the SC-DFF for that set of inputs N times and obtains the MSE as $\frac{1}{N}\sum_{i=1}^{N} e_i^2$, where e_i is the error of the i-th simulation. According to our experiments, which is omitted here due to space limitation, when simulation number N is less than 1000, the MSE obtained by the simulation method has a large error over the true MSE. Therefore, in our experiments, we chose N as 1000.

A. Correctness of the Proposed Method on General SC-DFFs

In this section, we verify the correctness of the proposed analysis method for general SC-DFFs. We tested on 6 circuits from [13], which implement arithmetic functions $\sin(x)$, $\cos(x)$, $\tanh(x)$, $\log(1 + x)$, e^{-x}, and $\text{sigmoid}(x)$. To reduce the number of gate types we need to consider, the SC core of each circuit was transformed into an AND-inverter graph (AIG) by ABC [14]. Both the analysis and the simulation methods were run on the AIGs. The bit-width of the input binary numbers was set as 7. We chose 6 different SBS lengths in our experiments, which are 32, 64, 128, 256, 512, and 1024. For each circuit and each SBS length, we chose 1000 different sets of input probabilities and averaged the MSE values of these sets to derive the final result for each method.

Fig. 5 shows the experimental results for the SC-DFFs with CMP as the PCCs. The figure plots the average MSEs obtained from the proposed analysis method and the simulation method for different circuits and SBS lengths. As the figure shows, the results obtained by the analysis method and those obtained by the simulation method are very close, which verifies the correctness of the proposed analysis method. Similar conclusions can be obtained when the PCCs are WBG.

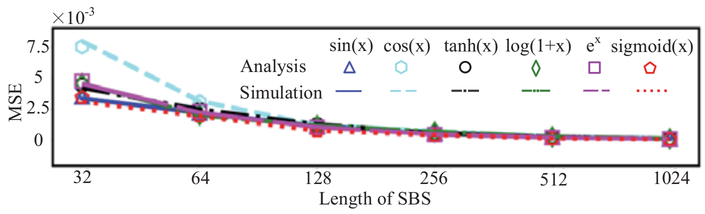

Fig. 5: The MSEs of the SC-DFFs using comparator as the PCCs.

B. Efficiency of the Proposed Method on General SC-DFFs

In this section, we compare the runtimes of the proposed analysis method and the simulation method. The experimental setup is same as that in Section IV-A. For a fair comparison, we also optimized the simulation method by implementing it using bit operations.

We chose CMP as the PCCs. Figs. 6(a) and (b) show the average runtimes of the proposed analysis method and the simulation method, respectively, for each circuit and each SBS length. We can see that the runtime of the analysis method is far less than that of the simulation method. For example, for obtaining the MSE of $\sin(x)$ with $n = 1024$, the proposed method is $560\times$ faster than the simulation method. Moreover, as shown in Fig. 6(a), the runtime of the analysis method changes little with the SBS length. In contrast, as shown in Fig. 6(b), the runtime of the simulation method increases exponentially with the logarithm of the SBS length. This is expected, since the SBS length is just a parameter in the analysis method, while the runtime of the simulation method is proportional to the SBS length. Similar conclusions can be obtained when the PCCs are WBG.

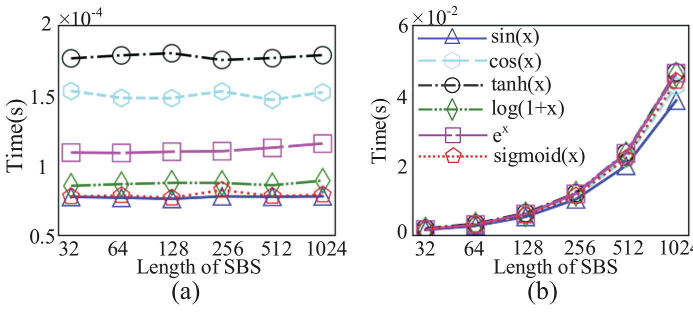

Fig. 6: The runtime of the proposed analysis method (a) and the simulation method (b).

C. Experimental Results on MUL-DFF

In this section, we verify the claims for MUL-DFF described in Section III-B. We chose the number of inserted DFFs as 1. We considered two choices of PCCs, CMP and WBG. We chose 4 products of input probabilities, which are 0.1×0.2, 0.3×0.8, 0.8×0.2, and 0.8×0.9. They represent various cases where a small or a large probability multiplies another small or a large probability. As a reference, we also considered a basic SC multiplier with CMP as the PCCs. We chose the bit-width of the input binary numbers as 10 and the bit stream length as 1024. The MSE comparison for the proposed analysis method and the simulation method and for various SC multipliers is shown in Fig. 7. In the figure, CMP and WBG refer to the MUL-DFFs with the PCCs implemented by CMP and WBG, respectively, while BSCM refers to the basic SC multiplier.

As shown in the figure, each MSE value calculated by the analysis method is close to the corresponding value obtained by the simulation method. This again verifies the correctness of the proposed analysis method. Each MSE of the MUL-DFF with CMP as the PCCs is neither less than the corresponding value of the basic SC multiplier, nor less than the corresponding value of the MUL-DFF with WBG as the PCCs. This verifies the correctness of Claims 5 and 7. For the input probability products 0.8×0.2 and 0.8×0.9, we can also see that the MSE of the MUL-DFF with WBG as the PCCs is less than the corresponding value of the basic SC multiplier. This verifies the correctness of Claim 6.

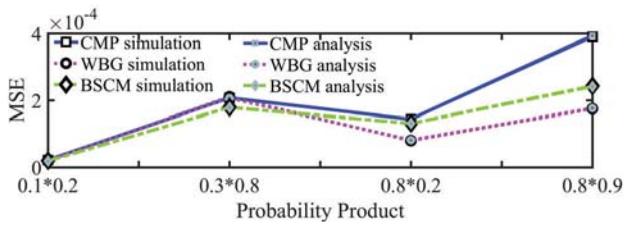

Fig. 7: The MSE comparison for the proposed analysis method and the simulation method and for the MUL-DFFs and the basic SC multiplier.

V. Conclusion

In this paper, we proposed a systematic method to analyze the accuracy of stochastic circuits with DFF insertion. As an important application, we analyzed the accuracy of the stochastic multiplier with DFF insertion and derive some interesting claims. Our experimental results verified the correctness of the proposed method and the significant runtime efficiency over the traditional simulation-based method. Besides, one important advantage of this method is that it gives an analytical formula on the accuracy. By knowing this, it is possible to further derive some general conclusions, as it is done for the MUL-DFF case in this study. The limitation of this work is that it assumes that the random binary numbers generated by the RNS at different cycles are independent and uniformly distributed. This is not strictly true for some widely-used pseudo-random number sources (PRNSs), such as linear feedback shift registers. In our future work, we will consider how to analyze the accuracy for SC-DFFs using these PRNSs.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (NSFC) under Grants 61204042 and 61472243.

References

[1] B. R. Gaines, "Stochastic computing systems," in *Advances in information systems science*. Springer, 1969, pp. 37–172.

[2] A. Alaghi *et al.*, "Stochastic circuits for real-time image-processing applications," in *DAC*, 2013, pp. 136:1–136:6.

[3] H. Ichihara *et al.*, "Compact and accurate digital filters based on stochastic computing," *IEEE TETC*, vol. 7, no. 1, pp. 31–43, 2019.

[4] A. Ren *et al.*, "SC-DCNN: highly-scalable deep convolutional neural network using stochastic computing," in *ASPLOS*, 2017, pp. 405–418.

[5] M. Yang *et al.*, "Design of accurate stochastic number generators with noisy emerging devices for stochastic computing," in *ICCAD*, 2017, pp. 638–644.

[6] P. K. Gupta and R. Kumaresan, "Binary multiplication with PN sequences," *IEEE TASSP*, vol. 36, no. 4, pp. 603–606, 1988.

[7] M. Yang *et al.*, "Towards theoretical cost limit of stochastic number generator for stochastic computing," in *ISVLSI*, 2018, pp. 154–159.

[8] Z. Li *et al.*, "Simultaneous area and latency optimization for stochastic circuits by D flip-flop insertion," *IEEE TCAD*, vol. 38, no. 7, pp. 1251–1264, 2019.

[9] F. Neugebauer *et al.*, "Framework for quantifying and managing accuracy in stochastic circuit design," *ACM JETC*, vol. 14, no. 2, pp. 31:1–31:21, 2018.

[10] T. Chen and J. P. Hayes, "Analyzing and controlling accuracy in stochastic circuits," in *ICCD*, 2014, pp. 367–373.

[11] W. Qian *et al.*, "An architecture for fault-tolerant computation with stochastic logic," *IEEE TC*, vol. 60, no. 1, pp. 93–105, 2011.

[12] A. Alaghi and J. P. Hayes, "STRAUSS: spectral transform use in stochastic circuit synthesis," *IEEE TCAD*, vol. 34, no. 11, pp. 1770–1783, 2015.

[13] K. Parhi and Y. Liu, "Computing arithmetic functions using stochastic logic by series expansion," *IEEE TETC*, vol. 7, no. 1, pp. 44–59, 2019.

[14] A. Mishchenko *et al.*, "ABC: a system for sequential synthesis and verification, release 80916," http://people.eecs.berkeley.edu/~alanmi/abc/.