Linear Feedback Shift Register Reseeding for Stochastic Circuit Repairing and Minimization

Chen Wang[1], Weikang Qian[1,2,*]
[1]University of Michigan-Shanghai Jiao Tong University Joint Institute, [2]MoE Key Laboratory of Artificial Intelligence
Shanghai Jiao Tong University, Shanghai, China
Email: wangchen_2011@sjtu.edu.cn, qianwk@sjtu.edu.cn; *corresponding author

***Abstract*—Stochastic computing (SC) is a re-emerging paradigm to realize complicated computation by simple circuitry. Although SC has strong tolerance to bit flip errors, manufacturing defects may still cause unacceptably large computation errors. SC circuits commonly adopt linear feedback shift registers (LFSRs) for stochastic bit stream generation. In this study, we observe that the computation error of a faulty LFSR-based SC circuit can be reduced by LFSR reseeding. We propose novel methods to use LFSR reseeding to 1) repair a faulty SC circuit and 2) minimize an SC circuit by constant replacement. Our experiments show the effectiveness of our proposed methods. Notably, the proposed SC circuit minimization method achieves an average 36% area-delay product reduction over the state-of-the-art fully-shared LFSR design with no reduction of the computation accuracy.**

I. Introduction

Stochastic computing (SC) is a re-emerging paradigm to perform complicated computation by simple circuitry [1]. Unlike conventional binary computing based on radix-2 numbers, SC operates on stochastic bit streams, which encode data through the ratios of 1s in the streams. SC circuits generally retain fairly low complexity. For example, the multiplication of two numbers requires only one *AND2* gate by SC. Moreover, SC enjoys strong tolerance to bit flip errors, due to the equal weight for all the bits in the encoding. However, SC is subject to errors due to random fluctuation [2]. Therefore, SC is usually applied to error-tolerant applications such as image processing [3], neural networks [4], and decoding of error-correcting codes [5].

Just like conventional digital circuits, an SC circuit can be faulty due to manufacturing defects, which may lead to a large systematic error. Therefore, even if an SC circuit is applied in a fault-tolerant scenario, the error caused by the fault may still violate the error constraint of the application, causing the circuit to be discarded. Thus, a lightweight repair method for faulty SC circuits is desirable.

Another issue with SC is that it usually requires long bit streams to achieve high computation accuracy. For each computation, the same number of clock cycles as the bit stream length is needed. Thus, although an SC circuit has a small critical path delay and power consumption, its overall computation latency and energy consumption are still large. Therefore, further minimization of the SC circuit is still desirable to reduce its latency and energy consumption.

One common type of SC design uses linear feedback shift registers (LFSRs) to provide stochastic bit streams [1] and it is the focus of this work. An LFSR outputs a pseudo-random number sequence controlled by its seed, i.e., the initial state of its registers. In this study, we consider selecting a new LFSR seed (hereafter called *LFSR reseeding*) as a new dimension to reduce the output error of a faulty LFSR-based SC circuit. As adopted by the test community for a long time, LFSR reseeding has become widely used for test compression by skipping inefficient test patterns [6]. It is worth noting that a previous work applied LFSR reseeding to improve the accuracy of SC circuit [7]. However, it focused on fault-free SC circuits. In our work, we focus on faulty SC circuits. By changing the seeds of the LFSRs in an SC circuit, we can change the stochastic bit streams and hence, the computation accuracy. Thus, a faulty SC circuit may be repaired without any modification to its hardware.

Our major contributions are listed as follows.

1) We observe a unique property of SC that a faulty LFSR-based SC circuit can be repaired by LFSR reseeding without any hardware modification.
2) We propose an efficient method to repair a faulty LFSR-based SC circuit by LFSR reseeding (see Section IV). Through a one-time offline characterization, it significantly reduces the number of LFSR seed vectors that need to be considered during the online repair stage by $585\times$ over a straightforward repairing method.
3) We propose a method to minimize an SC circuit by iteratively setting an internal signal to a constant 0 or 1 (hereafter called *constant replacement*) at design time (see Section V). Although it introduces error, we reduce the error by LFSR reseeding. Our experimental results show that this method can reduce the area-delay product (ADP) of the state-of-the-art shared LFSR-based SC circuit by 36% on average without increasing error.

II. Related Works

In this section, we discuss related works on SC circuit testing and minimization.

1) SC Circuit Testing: A previous work proposed a *tomographic testing method* to test and diagnose the faults existing in an SC circuit [8]. The method tests a manufactured SC circuit multiple times to obtain the distribution of its output values, which is then compared to the theoretical output distribution. However, it is only applicable to SC circuits based on true random number sources, which are not as mature as SC circuits based on LFSRs. Furthermore, faulty SC circuit

repair is not considered. In contrast, our work considers test and repair for the more common LFSR-based SC designs.

2) SC Circuit Minimization: A number of prior works propose various minimization methods for the SC circuit. Some of them focus on the minimization of the SC core, which is the main computing unit of an SC circuit. For example, the works [9], [10] propose heuristic methods to minimize the SC core. Recent works [11], [12] further explore target function approximation to reduce the hardware cost of the SC core. Other works consider improving the stochastic number generator (SNG) by methods like LFSR output scrambling [7], [13], D flip-flop insertion [14], [15], and the application of low-discrepancy (LD) sequence-based random number generators [16], [17]. The SC circuit minimization method proposed in our work is orthogonal to those prior methods. For instance, it can be applied to the LD sequence-based SC circuit design by reconfiguring the address counters or the direction vectors prestored in the memory to reduce the output error, thus improving the SC design.

III. Preliminaries

The general architecture of the SC circuits considered in this work is shown in Fig. 1, which consists of an LFSR-based SNG, an SC core, and a stochastic-to-binary converter (SBC). Specifically, the LFSR-based SNG takes the binary variables and coefficients as input, and generates the corresponding stochastic bit streams that are processed by the following SC core. The output bit stream of the SC core is then converted back to a binary output by the SBC. For any SC circuit design satisfying the general architecture such as the ones in [2], [9], [18]–[20], our approach is applicable. For clarity, in the following, we illustrate our method using an SC circuit design proposed in [18], which is shown in Fig. 2.

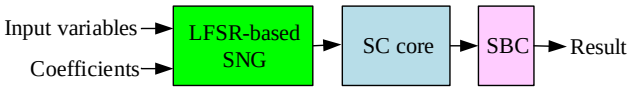

Fig. 1. The general architecture of the SC circuit considered in this study.

The circuit realizes a degree-n univariate Bernstein polynomial with precision parameter m approximating a target arithmetic function. It satisfies the general architecture shown in Fig. 1. The SNG includes n pairs of k-bit LFSR and *probability conversion circuit (PCC)* implemented by a comparator (CMP). They generate n bit streams for a k-bit binary input variable x by comparing x with the pseudo-random numbers produced by the LFSRs. The SNG also includes an m-bit LFSR. Its m cells provide m bit streams with value 0.5 as the coefficients. All the bit streams are then fed into an SC core that transforms them into an output bit stream realizing the target function. The output stream is finally converted to a binary number by the SBC implemented by a k-bit counter. The bit stream length is set as (2^k-1), as each k-bit LFSR has a period of (2^k-1). Since this design has $(n+1)$ LFSRs, it needs $(n+1)$ seeds for these LFSRs. We define *a seed vector* as $(s_1, s_2, \cdots, s_{n+1})$ to represent these seeds. Since the accuracy of an LFSR-based SC circuit depends on the seeds of its LFSRs [7], we assume that each LFSR has extra ports to support the load of a given seed. Thus, LFSR reseeding can be supported by these extra ports with no hardware modification.

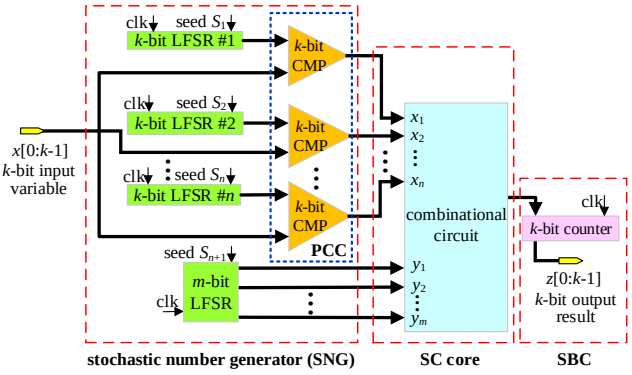

Fig. 2. An SC circuit design proposed in [18].

For the SC design, ideally, each input bit stream of the SC core should be independent. Therefore, in this example, each of the first n LFSRs for the input variable is based on a unique maximum-length feedback polynomial, which determines its output. Note that these input bit streams can also be generated by a single LFSR with its output scrambled [13]. This gives a much smaller SNG, but at the cost of accuracy loss. In what follows, we call the design with different LFSRs for different inputs the *unshared-LFSR design* and the one with an LFSR shared by multiple inputs the *shared-LFSR design*.

Although we illustrate our methods using a design shown in Fig. 2 that only implements univariate functions, it can be extended to handle multivariate polynomials [10]. Besides, many univariate functions are important, e.g., $\tanh(x)$ and the sigmoid function $1/(1+\exp(-x))$, which are used as the activation functions in machine learning.

IV. Repair Faulty SC Circuits by LFSR Reseeding

In this section, we present the details on repairing faulty SC circuits by LFSR reseeding. Since an SC circuit is relatively simple, it is extremely rare for it to have multiple faults simultaneously. Thus, we assume that only a single fault exists in a manufactured SC circuit. Our proposed method is not limited to any specific fault model. The output error of the SC circuit is measured by either the *worst-case absolute error (WCAE)* or the *mean absolute error (MAE)* [10] over all possible values of the input variable for a specific seed vector.

Assume v^* is the optimal seed vector that gives the minimum output error over all possible seed vectors for the fault-free SC circuit and e^* is the corresponding output error. We call them the *reference seed vector (RSV)* and the *reference output error (ROE)*, respectively. An SC circuit is claimed *faulty* if its output error under the RSV exceeds the given error bound e_b.

One straightforward method to repair a faulty SC circuit C_f by reseeding is to randomly generate a large set of seed vectors and try them one by one on C_f. If a seed vector reduces the output error below the given error bound, we find a repair to the circuit. Otherwise, if none of the seed vectors succeeds,

C_f is considered irreparable. One drawback of this method is that a large number of seed vectors need to be tried to achieve a decent repair rate.

To reduce the number of seed vectors to be tried, we propose a two-stage method consisting of an offline characterization stage and an online test-and-repair stage. The offline stage characterizes all the possible faults, while the online stage further selects a small set V_p of seed vectors to efficiently repair the faulty SC circuit. Next, we describe the details.

A. Offline Characterization Stage

Algorithm 1: The proposed offline characterization algorithm for SC circuit repair.

```
   Input  : a fault-free SC circuit C_0, the reference output error e*,
            and the reference seed vector v*.
   Output : set of faults S_1.
1  construct a set V of candidate seed vectors;
2  S_1 ← ∅;
3  foreach possible fault f in C_0 do
4      inject f into C_0 to obtain the SC circuit C_f;
5      simulate C_f under v* and obtain output error e; f.e_{v*} ← e;
6      if e ≤ e* then continue;
7      flag ← false; e_min ← +∞; v_min ← null;
8      foreach seed vector v in V do
9          simulate C_f under v and obtain output error e;
10         if e ≤ e* then flag ← true; f.e_r ← e; f.v_r ← v; break;
11         else if e < e_min then e_min ← e; v_min ← v;
12     if flag is false then f.e_r ← e_min; f.v_r ← v_min;
13     S_1 ← S_1 ∪ {f};
14 return S_1;
```

The offline characterization procedure is shown in Algorithm 1. Line 1 builds a set V of candidate seed vectors. To make our proposed method flexible, we assume that the given error bound e_b varies with the application and hence, is not known during the offline stage. However, one thing for sure is that e_b must be no less than the ROE e^*, i.e., $e_b \geq e^*$. Otherwise, even the fault-free SC circuit cannot be used.

We associate each fault f with three entries, namely, e_{v^*}, v_r, and e_r. Denote the SC circuit with fault f as C_f. Specifically, e_{v^*} holds the output error of C_f under the RSV v^*, v_r is a special seed vector in V described below, and e_r keeps the output error of C_f under v_r. The vector v_r is special in that *if it cannot repair C_f (i.e., $e_r > e_b$), then no other vector in V can repair C_f*. We call such a vector *decisive seed vector (DSV)*. As shown in Fig. 3(a), all the faults of the SC circuit can be partitioned into two subsets, S_0 and S_1, where S_0 consists of all the faults with $e_{v^*} \leq e^*$, while S_1 keeps the remaining faults. Clearly, a circuit C_f with the fault $f \in S_0$ needs not be repaired, since its $e_{v^*} \leq e^* \leq e_b$. Thus, we only need to consider the faults in S_1. The offline stage identifies S_1 together with the three entries of each fault in S_1.

To achieve the target, for each possible fault f in the input fault-free circuit C_0, Line 4 first obtains the circuit C_f by injecting f into C_0. Line 5 simulates C_f under v^* to obtain the output error e, which is then assigned to the entry e_{v^*} of fault f. If $e \leq e^*$, by definition, the fault f belongs to S_0 and it can be skipped (Line 6). Line 7 initializes *flag* to *false*. Then, for each seed vector v in V, Line 9 simulates C_f under v to obtain its output error e. If $e \leq e^*$, Line 10 sets *flag* to *true*, assigns e and v to the entries e_r and v_r of fault f, respectively, and exits the inner loop. As *flag* is now *true*, Line 12 is skipped and Line 13 adds fault f into set S_1. Otherwise, if e is smaller than the running minimum output error e_{min}, e_{min} and v_{min} are updated by e and v, respectively (Line 11). After all the seed vectors in V have been evaluated, if variable *flag* is still *false*, the two entries e_r and v_r of f are assigned with e_{min} and v_{min}, respectively (Line 12). Then, Line 13 adds fault f into set S_1. Finally, Line 14 returns set S_1.

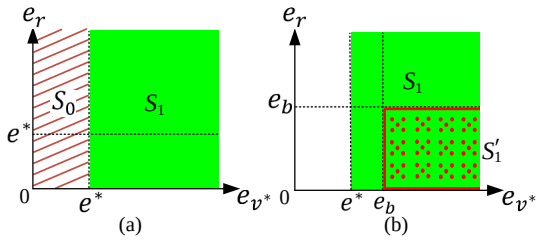

Fig. 3. Fault space partition for an SC circuit. (a): a bi-partition considered in the offline characterization stage; (b): a refined partition considered in the online test-and-repair stage when a specific error bound e_b is known. The dotted region represents the set of faults for deriving the seed vectors for online repair.

Note that the seed vector v_r found by the above procedure for each fault f is a DSV. Clearly, the corresponding e_r belongs to one of the following two cases: 1) $e_r \leq e^*$; 2) $e_r > e^*$. For the first case, v_r clearly can repair the faulty circuit C_f, as $e_r \leq e^* \leq e_b$. For the second case, v_r actually is the seed vector in V with the minimum output error. Thus, if v_r cannot repair C_f (i.e., $e_r > e_b$), neither can any other seed vector in V. By definition, v_r is a DSV.

The offline characterization is a one-time step. Ideally, set V used in the offline characterization should contain all possible seed vectors, but this can cause a long runtime of the offline characterization. To reduce the runtime, we propose a speedup technique to reduce the size of V. For the unshared-LFSR design with n k-bit LFSRs and a single m-bit LFSR, the total number of seed vectors is $(2^k-1)^n(2^m-1)$. If (2^k-1) is a multiple of (2^m-1), then all these seed vectors can be divided into $(2^k-1)^{n-1}(2^m-1)$ equivalent classes, each containing (2^k-1) seed vectors with the same output error. If such a condition holds, we can select a *representative seed vector* for each equivalent class, which reduces the number of seed vectors by (2^k-1) times. To further reduce the runtime, a user-defined threshold M is applied. If the total number of representative seed vectors does not exceed M, they will all be put into the set V for evaluation. Otherwise, we randomly generate M representative seed vectors and put them into V. Usually, with a larger M and hence a larger set V, more faults can be repaired, however at the cost of higher runtime.

B. Online Test-and-Repair Stage

In this stage, the user-defined output error bound e_b is given. The first step of this stage is to perform online testing on each fabricated SC circuit to obtain its output error under the RSV v^*. If its output error exceeds e_b, it is identified as faulty.

After testing, we perform online repair to the identified faulty SC circuits. Note that *we do not diagnose the fault*.

Instead, we directly try seed vectors from a small set V_p to check whether the circuit can be repaired. For this purpose, we first identify a set of faults $f \in S_1$ so that the faulty circuit C_f needs repair and can be repaired. For the other faults, the circuits with the faults either do not need repair or are irreparable.

Since e_b is known now, the faults of the faulty circuits C_f that need repair are those in S_1 with the entry e_{v^*} larger than e_b. Moreover, by the definition of DSV, C_f can be repaired if and only if the entry e_r of f does not exceed e_b. Therefore, C_f needs repair and can be repaired if and only if f is a fault in S_1 with $e_r \leq e_b$ and $e_{v^*} > e_b$, which is located in the dotted region shown in Fig. 3(b). Then, the entries v_r of all the faults in that region are collected to form the set V_p. These seed vectors in V_p are then sequentially applied to the circuit under repair. As soon as a seed vector reduces the output error below the given error bound, the circuit is repaired. Otherwise, the circuit is declared irreparable.

V. Minimize SC Circuit by Constant Replacement with LFSR Reseeding

In this section, we elaborate our method that minimizes the SC circuit by constant replacement at design time together with LFSR reseeding. Usually, LFSRs occupy a large part of an SC circuit area. The LFSR sharing approach can effectively reduce the LFSR area [13], [19]. With the LFSR area reduced, PCC becomes the most area-consuming part in an SC circuit. For example, as reported in [19], in a shared-LFSR SC filter circuit, the PCC occupies 84.6% of the total area. Thus, it is imperative to reduce the PCC area, which is our target.

The basic operation we use to minimize PCC is replacing an internal signal by a constant 0 or 1, a technique used in approximate circuit design [21]. It can further lead to the simplification of the adjacent gates of the signal, which effectively reduces the hardware cost of the circuit.

To guide our optimization procedure, we use ADP to measure the hardware cost. For the unshared-LFSR design, we denote the area of its ith LFSR as $A_{LFSR,i}$, and the area and delay of the counter as A_{CNT} and D_{CNT}, respectively. Since both the CMP-based PCC and the SC core are combinational circuits and they are connected together, we consider them as a whole combinational circuit with area and delay denoted as A_{comb} and D_{comb}, respectively. By analyzing the circuit, we find that the ADP of the unshared-LFSR design can be evaluated as

$$\left(\sum_{i=1}^{n+1} A_{LFSR,i} + A_{comb} + A_{CNT}\right) \cdot (D_{comb} + D_{CNT}) .$$

Note that the above ADP model can also be extended to the shared-LFSR design.

Now, we present the entire minimization procedure shown in Algorithm 2. Given an input SC circuit C_0 and an error bound e_b, we minimize the PCCs in the SC circuit by iterative constant replacement. Line 1 initializes the current circuit C as C_0 and the best seed vector v_{best} as the RSV v^*. In each iteration of the main loop, all possible constant

Algorithm 2: The proposed algorithm of SC circuit minimization by constant replacement with LFSR reseeding.

Input : an SC circuit C_0, the given output error bound e_b, and the reference seed vector v^*.
Output : the minimized SC circuit C and the best seed vector v_{best}.

1 initialize: $C \leftarrow C_0$; $v_{best} \leftarrow v^*$;
2 **while** *true* **do**
3 $S \leftarrow \emptyset$;
4 **foreach** *possible constant replacement r in the PCCs of C* **do**
5 apply r to C to yield a new SC circuit C_r;
6 apply LFSR reseeding to C_r to get the minimum output error e_{min} and its corresponding seed vector v_{min};
7 $r.v_{min} \leftarrow v_{min}$; $r.ADP \leftarrow$ ADP of C_r;
8 **if** $e_{min} \leq e_b$ **then** add r into S;
9 **if** $S \neq \emptyset$ **then**
10 identify $r^* \in S$ leading to the minimum ADP and apply r^* to C to update C;
11 $v_{best} \leftarrow r^*.v_{min}$;
12 **else return** C, v_{best};

replacements for the PCCs of the circuit C are evaluated one after another (Lines 4–8). By constant replacement, the output error may exceed the given error bound. In order to recover the computation accuracy, Line 6 applies LFSR reseeding to obtain the seed vector v_{min} that leads to the minimum output error e_{min}. Line 7 associates the replacement r with v_{min} and the ADP of the circuit with r applied. If e_{min} does not exceed e_b, this replacement is added into set S (Line 8). After all the constant replacements are evaluated, if set S is not empty, then the replacement r^* in S leading to the minimum ADP is selected to be actually applied to C to achieve the new circuit (Line 10). We also assign $r^*.v_{min}$ to v_{best} (Line 11). Then, we continue to the next iteration, working on the new circuit. Otherwise, if S is empty, the process terminates by returning the current SC circuit C and v_{best} (Line 12). Note that a constant replacement may eliminate an entire PCC. In this case, the LFSR connected to the PCC is also removed.

VI. Experimental Results

In this section, we show the experimental results obtained by a computer with a 6-core Intel CPU of 4.1 GHz and a 16GB memory. Logic synthesis and technology mapping were done by ABC [22] with the commands "*resyn2;map*" using the Nangate 45 nm standard cell library [23].

A. Unshared-LFSR Design

In this section, we present the experimental results for the unshared-LFSR SC design shown in Fig. 2. The benchmarks are the SC circuits synthesized by [10] to realize different target functions listed in Table I. We set k as 8. Therefore, the bit stream length is 255.

TABLE I
The benchmark information. n and m are parameters of the SC design in Fig. 2.

ID	(n,m)	function	ID	(n,m)	function	ID	(n,m)	function
1	(4,4)	$\cos(x)$	5	(6,6)	$\cos(x)$	9	(6,6)	$\tanh(x)$
2	(4,4)	$\exp(-x)$	6	(6,6)	$\exp(-x)$	10	(6,6)	$\exp(-2x)$
3	(4,4)	$\tanh(4x)$	7	(6,6)	$\tanh(4x)$	11	(6,6)	$1/(1+\exp(-x))$
4	(4,4)	$x^{2.2}$	8	(6,6)	$x^{2.2}$	12	(6,6)	$0.5\cos(\pi x)+0.5$

For each benchmark, we need to obtain the RSV and the ROE. For those benchmarks with $(n, m) = (4, 4)$, we

enumerate all the $255^3 \cdot 15$ representative seed vectors to get the RSV and the ROE. For the remaining benchmarks with $(n, m) = (6, 6)$, the number of seed vectors is prohibitively large. We obtain the RSV and the ROE from 3E+8 randomly generated seed vectors.

1) Repairing Faulty SC Circuits by LFSR Reseeding: In this section, we first check the performance of the proposed faulty SC circuit repair method for handling stuck-at faults (SAFs). For each benchmark, all possible SAFs existing in the LFSRs, the CMPs, the SC core, and the counter are considered. We apply the method described in Section IV to obtain the repaired SAFs. The parameter M used in the offline characterization stage is set as 1E+4. Since the number of all possible representative seed vectors exceeds M, set V in Algorithm 1 consists of M randomly generated representative seed vectors.

TABLE II
SAF REPAIRING RATES (%) BY LFSR RESEEDING FOR DIFFERENT WCAE AND MAE BOUNDS.

BM	WCAE bound					MAE bound				
ID	0.02	0.04	0.06	0.08	0.1	0.02	0.04	0.06	0.08	0.1
1	15.5	15.3	12.7	5.8	9.0	18.8	13.6	8.4	8.3	13.6
2	–	15.0	18.4	15.8	7.2	–	13.6	22.3	19.2	20.7
3	0	3.5	8.2	4.8	2.7	0	3.8	8.9	5.3	3.0
4	32.0	20.1	5.3	8.6	9.3	35.6	27.1	9.7	9.4	8.8
5	44.2	28.8	8.3	29.9	39.3	11.2	20.8	37.6	32.4	14.8
6	19.4	10.6	24.3	18.3	17.3	14.6	21.5	13.5	4.4	8.1
7	28.7	32.4	51.1	40.9	28.1	40.2	20.2	7.1	12.0	11.5
8	37.9	40.6	39.8	30.4	42.3	33.5	28.8	33.3	20	35.1
9	31.1	30.0	34.0	31.8	26.7	25.4	32.4	25.9	26.6	42.3
10	26.7	11.5	13.2	17.8	25.3	26.5	27.4	20.2	10.6	2.3
11	24.8	25.0	8.4	6.4	10.5	21.6	23.9	33.7	27.3	32.1
12	38.2	23.7	27.9	25.7	9.1	25.4	24.5	9.1	46.1	37.7

In order to quantitatively evaluate the repairing effectiveness, we define the metric *repairing rate* as the ratio of the number of repaired SAFs after LFSR reseeding over the number of SAFs causing the error violation before LFSR reseeding. Table II shows the repairing rate for each benchmark for different WCAE and MAE bounds. Notably, 6 and 7 out of the 12 benchmarks have the maximum repairing rate exceeding 30% for WCAE and MAE, respectively. Therefore, the proposed LFSR reseeding method can effectively repair the SAFs existing in these benchmarks.

We further compare the cost of the online repair stage of our proposed method to that of the straightforward method for those WCAE bounds listed in Table II. Assuming each SAF is equally likely to occur in the fabricated circuits, we measure the cost by the number of seed vectors evaluated, which is proportional to the amount of computational effort needed. For both methods, the same set V of 1E+4 random seed vectors are evaluated, giving the same repairing rates shown in Table II. However, our proposed method achieves an average cost reduction of $585\times$, showing its very high efficiency.

2) Minimizing SC Circuit by Constant Replacement with LFSR Reseeding: We apply constant replacement to the PCCs of the SC circuit, and obtain the minimized SC circuit satisfying the given error bound. The error bound is set as αe^*, where α is the relative error bound and e^* is the ROE. We consider WCAE here, and similar evaluation can also be done for MAE. After each constant replacement, 1E+3 randomly generated representative seed vectors are evaluated for LFSR reseeding.

TABLE III
EXPERIMENTAL RESULTS FOR SC CIRCUIT MINIMIZATION BY CONSTANT REPLACEMENT WITH LFSR RESEEDING.

BM ID	SC circuit under test	WCAE bound	minimum WCAE	#8-bit LFSR	area (μm^2)	delay (ps)	total ADP (improvement)	run-time(s)
1	original	-	8.84E-3	4	337.3	408.8	137892	–
	$\alpha = 1.1$	9.73E-3	8.84E-3	4	337.3	408.8	137892 (0%)	210
	$\alpha = 1.5$	1.33E-2	1.26E-2	2	195.0	335.3	65367 (52.6%)	748
2	original	-	2.95E-2	4	331.4	415.2	137597	–
	$\alpha = 1.1$	3.24E-2	2.92E-2	2	192.3	331.3	63721 (53.7%)	1158
	$\alpha = 1.5$	4.42E-2	4.16E-2	2	186.7	316.4	59086 (57.1%)	1314
3	original	-	1.84E-2	4	326.4	399.6	130409	–
	$\alpha = 1.1$	2.02E-2	1.84E-2	4	326.4	399.6	130409 (0%)	88
	$\alpha = 1.5$	2.75E-2	1.84E-2	4	326.4	399.6	130409 (0%)	101
4	original	-	7.99E-3	4	337.0	394.7	133033	–
	$\alpha = 1.1$	8.78E-3	7.99E-3	4	337.0	394.7	133033 (0%)	184
	$\alpha = 1.5$	1.20E-2	1.15E-2	3	267.1	374.6	100046 (24.8%)	633
5	original	-	7.89E-3	6	492.6	458.1	225666	–
	$\alpha = 1.1$	8.68E-3	7.89E-3	6	492.6	458.1	225666 (0%)	1610
	$\alpha = 1.5$	1.18E-2	1.10E-2	2	213.3	353.1	75320 (66.6%)	15443
6	original	-	7.54E-3	6	492.9	468.2	230778	–
	$\alpha = 1.1$	8.29E-3	7.54E-3	6	492.9	468.2	230778 (0%)	1622
	$\alpha = 1.5$	1.13E-2	1.06E-2	2	216.3	359.1	77658 (66.3%)	5874
7	original	-	1.56E-2	6	508.3	443.5	225437	–
	$\alpha = 1.1$	1.72E-2	1.72E-2	5	383.8	381.5	146439 (35%)	3216
	$\alpha = 1.5$	2.34E-2	2.15E-2	5	365.2	363.9	132907 (41%)	4972
8	original	-	9.60E-3	6	482.3	535.8	258368	–
	$\alpha = 1.1$	1.06E-2	9.60E-3	6	482.3	535.8	258368 (0%)	1742
	$\alpha = 1.5$	1.44E-2	1.42E-2	6	439.7	436.6	191979 (25.7%)	11614

Table III shows the experimental results for the first 8 benchmarks. Under a strict relative error bound $\alpha = 1.1$, both benchmarks 2 and 7 achieve an ADP improvement. As α increases to 1.5, all the benchmarks except the 3rd one gain significant ADP improvement. Notably, benchmarks 5 and 6 have more than 66% ADP reduction. The increase of α or (n, m) leads to the runtime increase, since more constant replacements are evaluated.

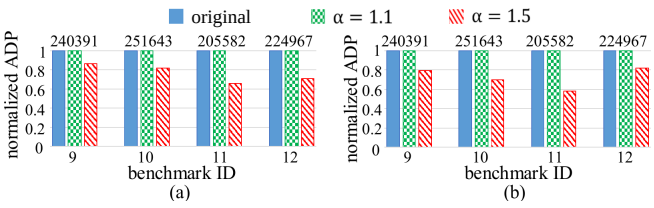

Fig. 4. Normalized ADP of the minimized SC circuit with different relative (a) WCAE and (b) MAE bounds over that of the fault-free case. The ADPs of the fault-free case are listed at the top of each category.

Fig. 4 shows the ADPs of the remaining benchmarks 9 to 12 minimized under different WCAE and MAE bounds. For $\alpha = 1.1$, none of these benchmarks have ADP reduction. However, as α increase to 1.5, 23.7% and 27.6% average ADP reductions are achieved for WCAE and MAE bounds, respectively.

Therefore, together with LFSR reseeding, applying constant replacement to the PCCs can significantly minimize the SC

circuit while meeting the given error bound.

3) Case Study on Gamma Correction: We evaluate our proposed SC circuit minimization method on gamma correction for image processing. The LFSR-based SC circuit implementing the gamma correction function $x^{0.45}$ is first built, where the SC core is implemented by [10] with parameters $(n, m) = (4, 4)$. 10 sample images of size 256×256 from [24] are evaluated.

TABLE IV
COMPARISON BETWEEN THE ORIGINAL AND THE MINIMIZED GAMMA CORRECTION SC CIRCUITS UNDER 1.02 RELATIVE WCAE BOUND.

SC circuit	WCAE bound	actual WCAE	#const. replace	#8-bit LFSR	area (μm^2)	delay (ps)	ADP (improv.)
original	–	6.64E-2	0	4	337.3	456.2	154014
minimized	6.77E-2	6.70E-2	12	3	246.8	350.3	86476 (43.85%)

We consider WCAE, and set the output error bound as 1.02 times the ROE. Table IV compares the original circuit and the minimized one. For the minimized circuit, its actual WCAE satisfies the error bound, while achieving a 43.85% improvement in ADP. Thus, the hardware cost is effectively reduced. Fig. 5 visualizes the image processing results for one sample image. Comparing the processed image shown in Fig. 5(c) to the reference image shown in Fig. 5(b), we can see that there is no significant difference.

(a) (b) (c)

Fig. 5. One sample image for the gamma correction application. (a): the original image; (b): the reference image by mathematical calculation; (c): the image processed by the minimized SC circuit after reseeding.

TABLE V
AVERAGE QUALITY METRICS OF 10 SAMPLE IMAGES PROCESSED BY DIFFERENT SC CIRCUITS.

SC circuit	PSNR (dB)	MSE	WCAE	MAE
original	30.43	9.61E-4	6.62E-2	2.53E-2
minimized before reseeding	26.76	2.18E-3	9.71E-2	3.99E-2
minimized after reseeding	28.12	1.57E-3	6.70E-2	3.46E-2

Next, four quality metrics, namely peak signal-to-noise ratio (PSNR), mean square error (MSE), WCAE, and MAE, are evaluated for each processed image with regard to the reference image obtained by mathematical calculation. As shown in Table V, after minimization, all the 4 metrics degrade from that of the original circuit. By reseeding, all these metrics are recovered but are still worse than that of the original circuit. Nevertheless, WCAE of the image processed by the minimized circuit after reseeding is only 0.9% worse than the ROE. However, the hardware cost of the SC circuit measured by ADP is reduced by 43.85%. This shows the effectiveness of the proposed minimization method.

B. Shared-LFSR Design

As mentioned in Section III, the state-of-the-art shared-LFSR design is more area efficient than the unshared-LFSR design. In this section, we study the effectiveness of our proposed techniques for shared-LFSR designs.

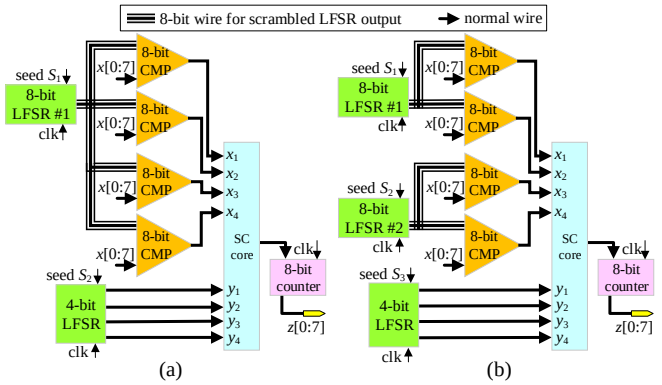

Fig. 6. (a) Fully shared LFSR design and (b) partially shared LFSR design.

Fig. 6(a) shows the *fully-shared design* of an SC circuit to realize a target arithmetic function with parameters $(n, m) = (4, 4)$. All the 4 CMP-based PCCs share a single 8-bit LFSR. The outputs of the LFSR are permuted by the pattern with the minimum average stochastic circuit correlation among 5E+5 random permutations similar to the method in [13].

The fully-shared design effectively reduces the area of the SC circuit. However, its computation accuracy suffers a large degradation. Therefore, we propose a partially-shared design as shown in Fig. 6(b) with an additional 8-bit LFSR. Each 8-bit LFSR has a unique feedback polynomial, and its two output permutations are in reverse order to maximize their independence [13]. Among the 4 benchmarks in Table I with $(n, m) = (4, 4)$ (i.e., benchmarks 1 to 4), the fully-shared design for benchmark 3 has a very large WCAE. Thus, in the following experiments, we ignore benchmark 3 and only test on benchmarks 1, 2, and 4.

We first evaluate the effect of LFSR reseeding to repair the faulty shared-LFSR designs. SAFs are considered. Note that the numbers of representative seed vectors for the fully-shared and the partially-shared designs are 15 and $255 \cdot 15$, respectively. Since the numbers are relatively small, all of these vectors are evaluated for LFSR reseeding.

TABLE VI
REPAIRING RATES (%) BY LFSR RESEEDING FOR DIFFERENT WCAE BOUNDS.

	BM ID	WCAE bound 0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20
fully shared	1	–	**7.32**	3.31	3.42	4.17	4.35	0	0	0	1.69
	2	–	**7.69**	4.17	0.69	2.11	2.22	0	0	0.76	0
	4	–	4.52	2.25	0	4.88	0	0.67	0.67	2.03	**10.64**
partially shared	1	**25.55**	10.19	10.96	7.44	5.08	20.35	6.32	0	0	2.27
	2	–	13.5	6.63	13.64	**15.65**	13.53	5.04	4.27	4.35	5.26
	4	14.35	8.99	8.44	5.71	6.57	1.54	2.5	5	17.24	**18.92**

Table VI shows the repairing rates of each design under different WCAE bounds. Comparing the maximum repairing rates denoted in bold, those for the partially-shared design are larger than those for the fully-shared design. This indicates that

LFSR reseeding is more effective to repair the partially-shared design because of its larger LFSR reseeding search space.

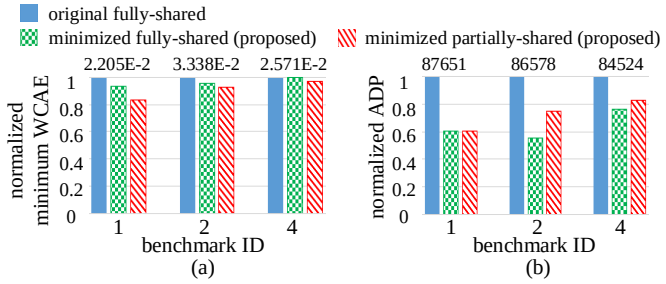

Fig. 7. Normalized (a) minimum WCAE and (b) ADP of the minimized fully-shared and partially-shared SC designs by constant replacement. The absolute values for the original fully-shared design are listed at the top of each category.

We further evaluate the effect of LFSR reseeding in minimizing shared-LFSR designs by constant replacement. We consider WCAE here, and set the error bound as the ROE of the original fully-shared design. As the results in Fig. 7 show, for the fully-shared design, despite its rather limited LFSR reseeding search space consisting of only 15 seed vectors, the ADP on average has a 36% reduction over the original design within the given error bound. This surprising result indicates that our method can further reduce the hardware cost of the state-of-the-art fully-shared design.

For the partially-shared design, as it has a larger LFSR reseeding search space than the fully-shared one, we can see that by constant replacement together with LFSR reseeding, its output WCAE further reduces over the fully-shared design with constant replacement. However, its average ADP degrades a little. Nevertheless, its ADP is on average 27.5% smaller than that of the original fully-shared design. This result shows a new way to design low-cost high-accuracy SC circuit. Compared to the fully-shared design, although an additional LFSR is introduced, the area increase is compensated by the PCC minimization due to constant replacement. Meanwhile, the additional LFSR greatly enlarges the LFSR reseeding search space, which can lead to a lower output error by LFSR reseeding.

VII. Conclusions

In this work, we proposed an LFSR reseeding method to repair faulty LFSR-based SC circuits. We further exploited this idea at design time to minimize SC circuits by constant replacement, followed by accuracy recovery through LFSR reseeding. These techniques are enabled by a unique property of an LFSR-based SC circuit, that is, a faulty circuit can be repaired by LFSR reseeding without any hardware modification.

Although this paper focuses on LFSR-based SC circuits, the basic idea is also applicable to recently proposed SC circuits based on Sobol sequence generator [16], [17]. Specifically, we can reduce the output error of a faulty SC circuit based on Sobol sequence generator by changing its counter's initial states or the direction vectors without any hardware modification. In conclusion, the possibility of reducing the error of a faulty circuit without any hardware modification is another significant advantage of SC over traditional binary computing, which can be exploited in SC circuit design and test.

Acknowledgments

This work is supported by the National Key R&D Program of China under Grant 2020YFB2205501.

References

[1] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE TCAD*, vol. 37, no. 8, pp. 1515–1531, 2018.
[2] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE TC*, vol. 60, no. 1, pp. 93–105, 2011.
[3] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams: Digital image processing case studies," *IEEE TVLSI*, vol. 22, no. 3, pp. 449–462, 2014.
[4] S. Lee, H. Sim, J. Choi, and J. Lee, "Successive log quantization for cost-efficient neural networks using stochastic computing," in *DAC*, 2019, pp. 1–6.
[5] 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 TCAS-I*, vol. 62, no. 2, pp. 507–516, 2015.
[6] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *IEEE TC*, vol. 44, no. 2, pp. 223–233, 1995.
[7] J. H. Anderson, Y. Hara-Azumi, and S. Yamashita, "Effect of LFSR seedings, scrambling and feedback polynomial on stochastic computing accuracy," in *DATE*, 2016, pp. 1550–1555.
[8] A. Paler, A. Alaghi, I. Polian, and J. P. Hayes, "Tomographic testing and validation of probabilistic circuits," in *ETS*, 2011, pp. 63–68.
[9] A. Alaghi and J. P. Hayes, "STRAUSS: Spectral transform use in stochastic circuit synthesis," *IEEE TCAD*, vol. 34, no. 11, pp. 1770–1783, 2015.
[10] X. Peng and W. Qian, "Stochastic circuit synthesis by cube assignment," *IEEE TCAD*, vol. 37, no. 12, pp. 3109–3122, 2018.
[11] C. Wang, W. Xiao, J. P. Hayes, and W. Qian, "Exploring target function approximation for stochastic circuit minimization," in *ICCAD*, 2020, pp. 122:1–122:9.
[12] X. Wang, Z. Chu, and W. Qian, "MinSC: An exact synthesis-based method for minimal-area stochastic circuits under relaxed error bound," in *ICCAD*, 2021, pp. 1–9.
[13] S. A. Salehi, "Low-cost stochastic number generators for stochastic computing," *IEEE TVLSI*, vol. 28, no. 4, pp. 992–1001, 2020.
[14] P. Ting and J. P. Hayes, "Isolation-based decorrelation of stochastic circuits," in *ICCD*, 2016, pp. 88–95.
[15] Z. Li, Z. Chen, Y. Zhang, Z. Huang, and W. Qian, "Simultaneous area and latency optimization for stochastic circuits by D flip-flop insertion," *IEEE TCAD*, vol. 38, no. 7, pp. 1251–1264, 2019.
[16] M. H. Najafi, D. J. Lilja, and M. Riedel, "Deterministic methods for stochastic computing using low-discrepancy sequences," in *ICCAD*, 2018, pp. 1–8.
[17] S. Liu and J. Han, "Toward energy-efficient stochastic circuits using parallel Sobol sequences," *IEEE TVLSI*, vol. 26, no. 7, pp. 1326–1339, 2018.
[18] Z. Zhao and W. Qian, "A general design of stochastic circuit and its synthesis," in *DATE*, 2015, pp. 1467–1472.
[19] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, and T. Inoue, "Compact and accurate digital filters based on stochastic computing," *IEEE TETC*, vol. 7, no. 1, pp. 31–43, 2019.
[20] K. K. Parhi and Y. Liu, "Computing arithmetic functions using stochastic logic by series expansion," *IEEE TETC*, vol. 7, no. 1, pp. 44–59, 2019.
[21] D. Shin and S. K. Gupta, "A new circuit simplification method for error tolerant applications," in *DATE*, 2011, pp. 1–6.
[22] A. Mishchenko, "ABC logic synthesis package," 2012. [Online]. Available: https://people.eecs.berkeley.edu/ alanmi/abc/abc.htm
[23] Nangate, Inc., "Nangate 45 nm open cell library," 2020. [Online]. Available: https://si2.org/open-cell-library/
[24] Imageprocessingplace, "Image database," 2019, http://imageprocessingplace.com/root_files_V3/image_databases.htm.