Design of an Efficient Parallel Random Number Generator using a Single LFSR for Stochastic Computing

Donghui Lee, Hyoju Seo and Yongtae Kim*
School of Computer Science and Engineering, Kyungpook National University, Daegu, Republic of Korea
{*thebock12, hyoju, yongtae*}*@knu.ac.kr*

***Abstract*—This paper proposes a parallel random number generator (RNG) using a single linear feedback shift register (LFSR) to generate two distinct random numbers, achieving twice the operational speed of a traditional serial RNG. The proposed RNG generates two distinct random numbers utilizing an LFSR. When implemented in a 65-nm CMOS technology, the proposed design leads to a 15.6% improvement in area and a 14.8% improvement in power efficiency, addressing the trade-off between accuracy and energy efficiency in stochastic computing (SC). Furthermore, the proposed design not only matches but surpasses the performance of serial SC in an edge-detection digital image processing application. Therefore, for enhanced hardware efficiency and improved accuracy, the proposed parallel RNG architecture can be effectively employed.**

Index Terms*—*stochastic computing (SC), parallel random number generator (RNG), linear feedback shift register (LFSR)

I. Introduction

Stochastic computing (SC), which performs probability-based computations, was first introduced in the late 1960s [1]. Unlike conventional binary computing, SC conducts operations probabilistically, facilitating the development of error-tolerant designs. These distinctive characteristics render SC widely applicable in various applications, including digital filters, image processing, and neural networks [2]–[8]. SC offers a low-power design paradigm by executing operations using simple logic circuits such as AND gates or XOR gates. In contrast to conventional binary computing, SC operates probabilistically, making it suitable for developing error-tolerant designs. The core of SC involves the use of a unary bitstream represented by probabilities, known as stochastic numbers (SN). The conversion of binary numbers into SNs is carried out by the stochastic number generator (SNG), which comprises a comparator and a random number generator (RNG). The RNG commonly employs a linear feedback shift register (LFSR) due to its simplicity and favorable random characteristics [9]. The SNG process involves assigning a value of 1 to the SN if the input binary number exceeds the randomly generated number by the RNG; otherwise, it assumes a value of 0. There are two encoding techniques for the SN bitstream: unipolar and bipolar. In the unipolar representation, 0 and 1 of SN have weights of 0 and 1, respectively, resulting in a range of [0, +1]. On the other hand, in the bipolar representation, 0 and 1 of SN have weights of -1 and 1, respectively, leading to a range of [-1, +1]. Consequently, the unipolar encoding method exclusively facilitates the representation of positive values, while the bipolar encoding method accommodates the representation of both positive and negative signed values. These encoding techniques provide flexibility in representing different value ranges within the SC domain.

The accuracy of SC is determined by the bitstream length, denoted as N. A stochastic number of length N, featuring equal weights for each bit, can represent a total of $N + 1$ distinct values. For instance, a length of 256 can express values ranging from 0 to 1 in increments of $1/256$. However, a notable drawback of SC is its dependence on longer SN bitstreams to achieve high accuracy, which consequently diminishes energy efficiency. While an SN with identical weighted bits enhances SC's robustness to bit errors, it necessitates an exponentially increased bitstream length. Specifically, achieving equivalent precision for an n-bit binary number requires a 2^n-bit SN bitstream in SC. SC can be implemented in two main ways: serial and parallel. Fig. 1 shows serial and parallel SC architectures. The parallel approach in SC serves as a strategic solution to mitigate the prolonged delays inherent in serial methods. However, this scheme exhibits exponential increases in hardware consumption.

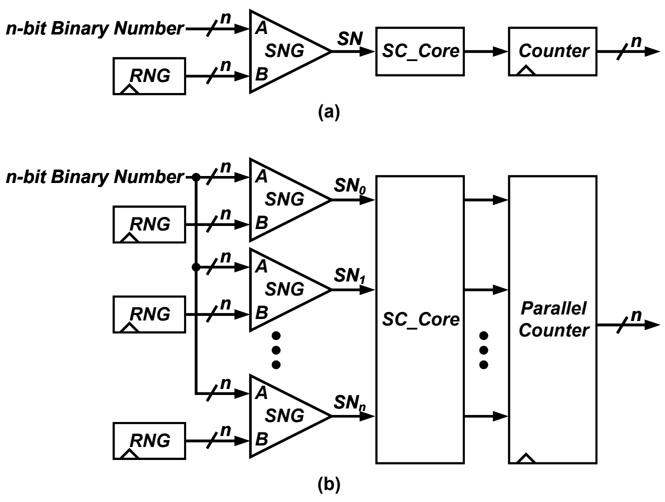
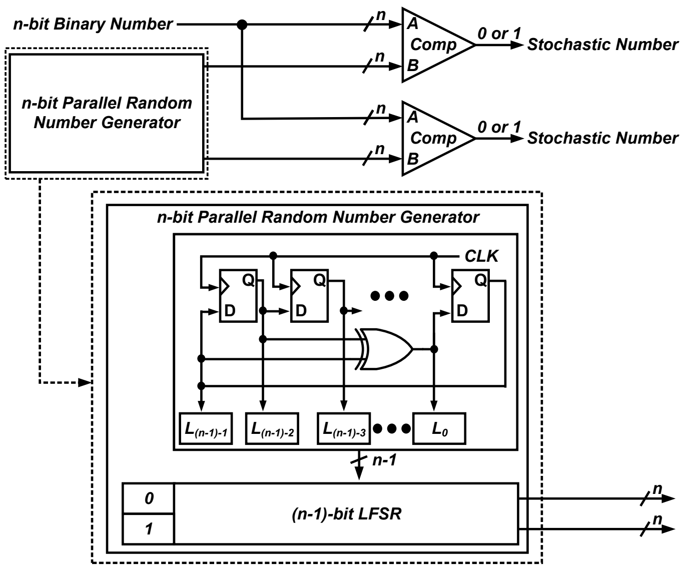


Fig. 1. Stochastic computing architectures; (a) serial and (b) parallel.

This paper proposes a new parallel RNG that utilizes a single LFSR to generate two distinct random numbers. The proposed parallel RNG achieves twice the operational speed compared to traditional serial RNG, leading to efficient energy improvements, with a 15.6% and a 14.8% improvement in area and power, respectively. This innovative approach addresses the trade-off between accuracy and energy efficiency in SC, making it a promising advancement in the field of SC architectures.

II. Proposed Parallel Random Number Generator

We address the problem of increased hardware consumption resulting from individual RNGs being used to generate SN simultaneously in parallel computing scenarios. To overcome this challenge, we propose a novel architecture for a parallel RNG that aims to increase parallel SC's efficiency. The ar-



n-bit Binary Number
n-bit Parallel Random Number Generator
Comp
0 or 1
Stochastic Number
n-bit Parallel Random Number Generator
CLK
(n-1)-bit LFSR
n-1

Fig. 2. Proposed parallel random number generator (RNG) architecture.

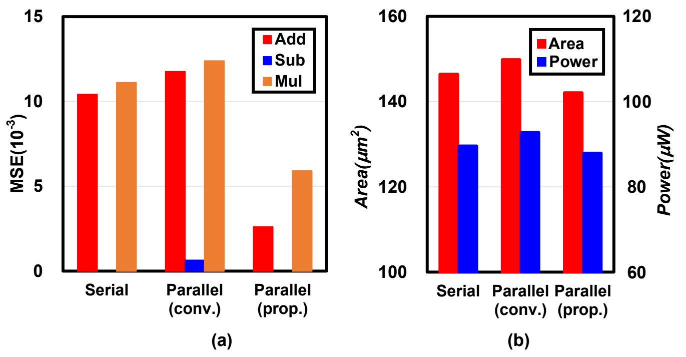


Fig. 3. Proposed parallel RNG performance: (a) mean squared error (MSE) and (b) area and power consumption.

TABLE I
HARDWARE RESOURCE CONSUMPTION OF ROBERT CROSS-EDGE DETECTOR HARDWARE.

RNG configuration		*Area* [μm^2]	*Power* [μW]
Serial SC	*no_share*	951.0	513.1
	all_share	369.6	156.2
Parallel SC	*all_share_8bit_LFSR*	500.5	187.5
	proposed design	477.1	168.9

chitecture of the proposed parallel RNG is visually depicted in Fig. 2, providing a detailed overview of its structure. This design represents a departure from the conventional practice of employing separate n-bit LFSRs for each parallel RNG. In contrast, the proposed RNG embraces a more resource-efficient strategy by utilizing a single $(n-1)$-bit LFSR to concurrently generate two distinct n-bit random numbers. The key idea lies in establishing fixed values for the most significant bit (MSB) of the generated random numbers. By assigning 0 and 1 to the MSB, the design enables the simultaneous generation of two unique n-bit random numbers. These fixed MSB values are integrated with the output of the $(n-1)$-bit LFSR, facilitating the concurrent generation of two n-bit random numbers. The n-bit random number with a fixed MSB of 0 uniformly distributes values within the range of 1 to $2^{(n-1)}-1$. In contrast, the random number with a fixed MSB of 1 generates values ranging from $2^{(n-1)}+1$ to 2^n-1. The conventional implementation of an n-bit LFSR typically involves XOR gates and registers. In contrast, the proposed design optimizes this architecture by reducing the number of XOR gates and registers, resulting in a significant decrease in hardware consumption. For example, an 8-bit LFSR conventionally requires three XOR gates and eight registers. However, the proposed design with a 7-bit LFSR requires only one XOR gate and seven registers. The proposed parallel RNG design introduces a novel and efficient approach to generating random numbers for parallel SC, effectively mitigating the associated increase in hardware consumption. By employing a single LFSR for the concurrent generation of two random numbers, the design optimizes resource utilization, offering a promising solution for parallel computing scenarios. This novel method addresses the issues concerning hardware usage while also offering a scalable and effective parallel SC solution. The use of a single LFSR to produce two random numbers streamlines the random number generation process, leading to improved efficiency and reduced resource overhead.

III. EXPERIMENTAL RESULTS

To evaluate the proposed design, we assess the accuracy of SC operations using the 8-bit RNG. The accuracy was measured by calculating the mean squared error (MSE) with all possible inputs. Additionally, for evaluating hardware performance, we implemented the various SC designs in Verilog HDL and synthesized them using a 65-nm CMOS technology.

In Fig. 3(a), the MSE for addition, subtraction, and multiplication operations is presented for serial SC and parallel SC using both the conventional 8-bit LFSR and the proposed parallel RNG design. For addition, utilizing the conventional 8-bit LFSR in parallel SC leads to an approximate 1.1× increase in MSE compared to serial SC. In contrast, the proposed parallel RNG design demonstrates a remarkable 4× improvement in MSE relative to serial SC. The proposed design yields an MSE nearly identical to that of serial SC for subtraction operations, showcasing its effectiveness. Conversely, parallel SC using the 8-bit LFSR shows a slight increase in MSE. In multiplication, parallel SC using conventional methods indicates a 1.1× increase in MSE compared to serial SC. However, the proposed parallel RNG design stands out with a noteworthy 1.5× decrease in MSE, highlighting its superior performance. In Fig. 3(b), the area and power consumption of serial RNG, parallel RNG, and the proposed RNG in 8-bit SC are illustrated. The proposed design exhibits a 9.4% reduction in area compared to serial RNG and a substantial 15.6% reduction compared to parallel RNG. In terms of power consumption, the proposed RNG design shows a 5.6% enhancement relative to the serial RNG and a significant 14.8% improvement compared to the parallel RNG. These results underscore the efficiency gains of the proposed RNG design, positioning it as a promising advancement in the realm of parallel SC.

To assess the practical performance of the proposed design, we implemented hardware for the Robert cross-edge detector, and Table I presents the corresponding area and power consumption for various SC architectures. For generating the input required for the Robert cross-edge detector operations, two approaches were considered: *all_share*, where a single RNG

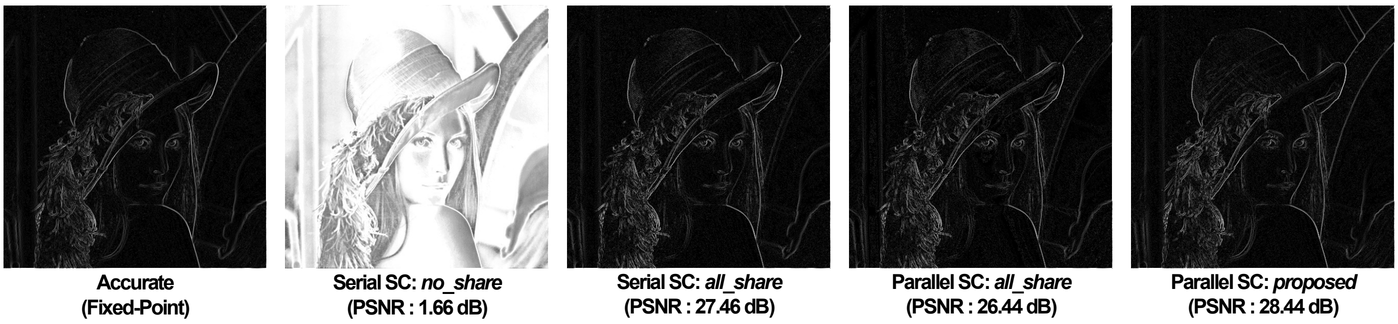


Fig. 4. Output images with PSNRs of SC-based Robert cross-edge detector.

is employed to generate SN, and *no_share*, where individual RNGs are used. When comparing the proposed design to serial SC with *no_share*, there is a substantial 49.8% improvement in area and a significant 67.1% reduction in power consumption. This emphasizes the efficiency gains achieved by the proposed design in comparison to the traditional serial SC configuration. Conversely, when compared to the *all_share* approach, there is a 29.1% increase in area and an 8.1% increase in power for the proposed design. The adoption of a parallel methodology in the proposed design leads to a simultaneous increase in both area and power consumption compared to the existing serial SC architecture. This enhancement in computational resources is, however, accompanied by a trade-off, as the delay required for calculations is halved compared to the serial SC configuration. Furthermore, in comparison to parallel SC with *all_share*, the proposed design demonstrates a decrease of 4.7% in area and 9.9% in power consumption. This exhibits the optimization achieved by the proposed design in terms of resource utilization and power efficiency in a parallel computing context.

Fig. 4 illustrates output images processed with accurate fixed-point arithmetic alongside those processed with both serial and parallel SC. To assess image processing performance, the peak signal-to-noise ratio (PSNR) was employed, comparing output images processed with accurate fixed-point arithmetic to those processed with SC methodologies. Given that subtraction operations in SC necessitate two correlated inputs, the PSNR for SC with the *no_share* RNG is observed to be the lowest. However, when compared to serial SC employing the *all_share* RNG, the proposed design exhibits a notable 3.6% improvement in PSNR. This improvement suggests that the proposed parallel RNG design contributes to enhanced accuracy and quality in image processing applications compared to traditional serial SC configurations. Furthermore, when contrasted with parallel SC utilizing an 8-bit LFSR, the proposed design demonstrates a substantial 7.6% enhancement in PSNR. This significant improvement underscores the efficacy of the proposed parallel RNG design in elevating image processing performance in SC domain. The higher PSNR values indicate superior fidelity and quality in the output images processed using the proposed design, highlighting its potential for improving the accuracy of image processing applications.

IV. Conclusion

In this paper, we introduced an energy-efficient parallel RNG utilizing an LFSR to generate two distinct random numbers. When compared to the serial RNG, the accuracy of the operation results shows a $4\times$ improvement in addition and a $1.5\times$ improvement in multiplication. Additionally, subtraction shows the same result. Implemented in a 65-nm CMOS technology, the proposed RNG exhibits significant advancements, including a 15.6% reduction in area and a 14.8% decrease in power consumption compared to existing parallel RNGs. When extended to image processing applications, the proposed RNG design, despite an increase in both area and power compared to serial SC, demonstrates the ability to execute operations twice as fast as the conventional delay. This accelerated processing capability is a crucial asset for real-time applications, contributing to improved overall efficiency. The PSNR results further underscore the enhanced image processing performance of the proposed RNG design compared to the serial SC method. This improvement in image quality establishes the proposed design as a promising solution for image processing applications. With notable reductions in area and power consumption, coupled with accelerated processing speeds and enhanced image processing results, the proposed RNG emerges as a promising solution for parallel SC applications. Its efficiency and performance enhancements make it a valuable contribution to the field, with potential applications in various parallel computing scenarios.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00279770).

References

[1] B. R. Gaines, *Stochastic Computing Systems*, pp. 37–172. 1969.
[2] Y. Liu *et al.*, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 2809–2824, 2021.
[3] S. Hu *et al.*, "Hybrid stochastic ldpc decoder with fully correlated stochastic computation," *IEEE Trans. on Circuits Syst. I: Reg. Papers*, vol. 69, no. 9, pp. 3643–3654, 2022.
[4] S. Liu and J. Han, "Dynamic stochastic computing for digital signal processing applications," in *Proc. Des. Autom. Test Eur. Conf. Exhib. (DATE)*, pp. 604–609, 2020.
[5] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic Circuits for Real-Time Image-Processing Applications," in *Proc. Design Autom. Conf. (DAC)*, (New York, NY, USA), pp. 1–6, 2013.
[6] D. Lee, J. Baik, and Y. Kim, "An accurate and efficient stochastic computing adder exploiting bit shuffle control scheme," in *Int. SoC Design Conf. (ISOCC)*, pp. 51–52, 2022.
[7] D. Lee, J. Baik, and Y. Kim, "Enhancing stochastic computing using a novel hybrid random number generator integrating lfsr and halton sequence," in *Int. SoC Design Conf. (ISOCC)*, pp. 7–8, 2023.
[8] D. Lee and Y. Kim, "Towards quantized stochastic computing by leveraging reduced precision binary numbers through bit truncation," in *Proc. Int. Conf. Comput. Design (ICCD)*, pp. 419–422, 2023.
[9] A. Zhakatayev *et al.*, "An efficient and accurate stochastic number generator using even-distribution coding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3056–3066, 2018.