Energy Efficient Stochastic Computing with Sobol Sequences

Siting Liu and Jie Han
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Alberta T6G 1H9, Canada
Email: {siting2, jhan8}@ualberta.ca

Abstract—**Energy efficiency presents a significant challenge for stochastic computing (SC) due to the long random binary bit streams required for accurate computation. In this paper, a type of low discrepancy (LD) sequences, the Sobol sequence, is considered for energy-efficient implementations of SC circuits. The use of Sobol sequences improves the output accuracy of a stochastic circuit with a reduced sequence length compared to the use of another type of LD sequences, the Halton sequence, and conventional linear feedback shift register (LFSR)-generated pseudorandom sequence. The use of Sobol sequences leads to a similar or higher accuracy than using Halton sequences for basic arithmetic operations. Sobol sequence generators cost less energy than the Halton counterparts when multiple random sequences are required in a circuit, thus the use of Sobol sequences can lead to a higher energy efficiency in an SC circuit than using Halton sequences.**

I. Introduction

Stochastic computing (SC) is an unconventional computing technique originally proposed to reduce the size of digital arithmetic circuits [1], [2]. It is more robust against noise and bit flip errors than analog and conventional digital circuits by using stochastic sequences to represent numbers. In a typical SC system, a binary integer is converted to a real value in the range $[0, 1]$ for the unipolar representation or $[-1, 1]$ for the bipolar representation of a stochastic sequence. Let N_1 denote the number of 1's in a stochastic sequence with N_L-bit length; then, this sequence encodes N_1/N_L in the unipolar representation or $(2N_1-N_L)/N_L$ in the bipolar representation.

An SC arithmetic unit is typically very small, which leads to significant saving in hardware and power consumption. A complex mathematical function such as a Bernstein polynomial can be implemented by just a few logic gates in SC [3], while it requires considerable hardware in a conventional design.

However, the disadvantages of SC include the long latency and a low energy efficiency. Since the stochastic sequence behaves similarly as a Bernoulli sequence, the variance of the computed results is proportional to $1/\sqrt{N_L}$ [4]. It indicates that a long sequence must be used to obtain a high accuracy. Although the hardware implementation of basic SC arithmetic units are simplistic, the required stochastic number generators (SNGs) consume significant hardware and energy. It has been shown that linear feedback shift register (LFSR)-based SNGs and the circuits that convert stochastic numbers to binary ones contribute to more than 80% of the total circuit cost [5].

For LFSR-based SNGs, the effect of seed selection is exploited in [6] and various strategies are applied to enhance the output accuracy of an SC circuit. Although the optimized sequences achieve a higher accuracy, a subset of the sequences suffers from large random fluctuations, thus the error can be large when the sequence length is smaller than the maximal length of an LFSR-generated sequence ($2^N - 1$ for an N-bit LFSR). In [7], the Halton sequence is introduced for use in SC. This low discrepancy (LD) sequence significantly reduces the sequence length for achieving the same accuracy compared to LFSR-generated pseudorandom sequences. However, Halton sequences rely on the use of counters with different radices when several independent sequences are required, thus a base conversion is indispensable for the current binary system [7]. The base conversion results in additional hardware overhead.

In this paper, another type of LD sequences, the Sobol sequence, is introduced for an energy-efficient implementation of SC. The generation of Sobol sequences only requires simple binary logic operations, so it does not incur a base-conversion overhead compared to Halton sequences. A multiplier and a multiplexing circuit implementing Bernstein polynomials, using Sobol and Halton sequences, are implemented for the evaluation of hardware efficiency, compared to conventional implementations using LFSR-generated sequences.

II. Low-discrepancy Sequences

A. Theory

LD sequences were first used to accelerate the convergence process of Monte-Carlo (MC) integration [8], [9]. MC integration requires S-dimensional random sequences to estimate the S-dimensional numerical integration. Given random sequences with sufficient length, the MC integration can provide a close estimation of the numerical integration. It has been shown that a lower discrepancy in random sequences indicates a smaller error in an MC integration [8]. An SC circuit can be considered as an MC problem. It is shown in [7] that a lower discrepancy also indicates a smaller error in SC.

The discrepancy of S-dimensional sequences P of N_L values can be quantitatively measured by the star discrepancy $D^*(P)$. "Dimension" here refers to the dimension of sample space in MC integration and it is equivalent to the number of independent random sequences in an SC circuit. The condition for being an LD sequence is that $D^*(P)$ has a convergence speed of $O(\log(N_L)^{S-1}/N_L)$. Thus a longer sequence (larger

N_L) and/or fewer independent sequences (smaller S) imply a smaller error in an SC circuit.

Several methodologies have been developed to generate different types of LD sequences, including Halton, Sobol and Faure sequences. Software-based generation methods have been developed, but few have been implemented in hardware. In this paper, Sobol and Halton sequences are considered for their ease in hardware generation. Also, the discrepancy of Sobol sequences is smaller than that of Halton sequences, especially when S is large and N_L is small [8].

B. Generators

The designs in [10] and [7] are adopted for Sobol and Halton sequence generation, respectively, as shown in Fig. 1.

The Sobol sequence generator consists of an address generator that detects the position of the least significant zero, a storage array storing the values of the direction vectors as intermediate variables for sequence generation, and the XOR gate and D flip-flop (DFF) pair for recursively generating quasi-random numbers. The algorithm for it is elaborated in [11].

Independent Halton sequences are generated by co-prime base numbers $\{b_i\}$ $(i = 1, 2, 3, \dots)$. The n-th random number (RN) using base b can be expressed as

$$\mathrm{RN}_{n,b} = \sum_{i=0}^{\infty} a_i b^{-i-1}, \tag{1}$$

where a_i is the i-th digit of the b-ary expansion of n. Thus the Halton sequence generator can be accordingly built by a reversely mapped b-ary counter, as shown in Fig. 1(b). Each digit of the b-ary counter is coded in the binary format. Compared to the design in [7], the base conversion is performed by converting a value to b-ary in advance instead of using a built-in hardware converter, thus saving hardware while creating some overhead to perform the conversion offline. The S smallest prime numbers serve as the bases for S-dimensional Halton sequences for a lower hardware cost.

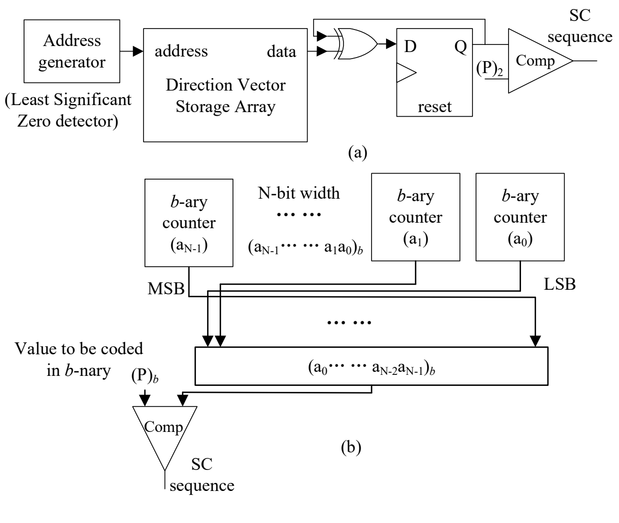

Address generator
(Least Significant Zero detector)
address
data
Direction Vector Storage Array
D
Q
reset
$(P)_2$
Comp
SC sequence
(a)
b-ary counter (a_{N-1})
N-bit width
⋯ ⋯
$(a_{N-1} \cdots \cdots a_1 a_0)_b$
b-ary counter (a_1)
b-ary counter (a_0)
MSB
LSB
⋯ ⋯
Value to be coded in b-nary
$(P)_b$
$(a_0 \cdots \cdots a_{N-2} a_{N-1})_b$
Comp
SC sequence
(b)

Fig. 1. (a) Sobol-based SNG; (b) Halton-based SNG.

Note that the simplest Sobol sequence is the same as a base-2 Halton sequence, so the inversely mapped counter is used for a single Sobol sequence generator to reduce hardware cost.

III. Hardware Simulation and Analysis

A. Basic Stochastic Computing Elements

To compare the efficiency of different types of sequences used in SC, two typical SC elements, a unipolar multiplier and a multiplexing circuit implementing a Bernstein polynomial, are considered. The schematics are shown in Fig. 2.

A unipolar stochastic multiplier is implemented by an AND gate with two inputs in Fig. 2(a). For a stochastic implementation of the Bernstein polynomial, the sequences $\{z_n\}$ $(n = 0, 1, \dots, N)$ are used to encode the Bernstein coefficients and the sum of multiple independent stochastic sequences of the input x serves as the selection signals for the multiplexer.

B. Metrics

The performance of SC elements are examined by energy per operation (EPO) and throughput per area (TPA) [12]. The root-mean-squared error (RMSE) is used to measure accuracy.

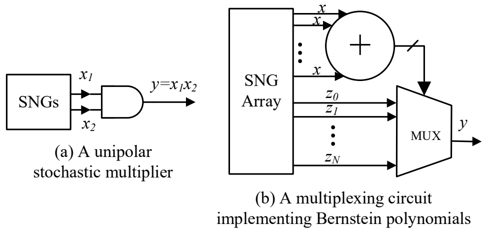
Given the sequence length N_L for a stochastic arithmetic operation, the EPO for an SC element can be calculated by

$$\text{EPO} = \text{Power} \times T_{clk} \times N_L, \tag{2}$$

where T_{clk} is the clock period and power is measured at the corresponding T_{clk}. Throughput is used to measure how much information a system can process during a unit time. In this case, the throughput is measured by how many effective bits are output in a unit time. The effective bits are the bit width for a binary output, and it is $\lfloor \log_2(N_L) \rfloor$ for an SC design. The time factor is measured by $t_c \times N_L$, where t_c is the critical path delay. For example, if an 8-bit result is produced by a sequence of 256 bits, the throughput is calculated by $8/256/t_c$. Subsequently, the TPA is given by

$$\text{TPA} = \text{Number of effective bits}/(t_c \times N_L)/\text{area}. \tag{3}$$

The critical path delay, area and power consumption are first obtained by the Synopsis Design Compiler with a 28nm STM process. The EPO and TPA are then computed. The RMSE is measured by using 10,000 random trials for each circuit.



(a) A unipolar stochastic multiplier

(b) A multiplexing circuit implementing Bernstein polynomials

Fig. 2. Basic stochastic computing elements.

C. Simulation Results

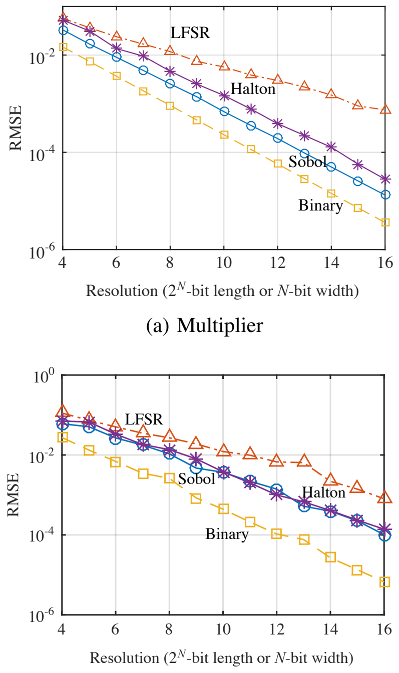
The SC circuits using Sobol, Halton and LFSR-generated sequences are compared with their conventional binary counterparts. Since the circuits for bipolar and unipolar representations show similar behaviors, only the unipolar SC elements are considered. Binary designs with different bit widths are considered for comparison: the multiplier is implemented by an array multiplier and the binary polynomial circuit is optimized to use the least number of multipliers. The comparisons are made in terms of RMSE, EPO and TPA.

1) Accuracy: The accuracy is measured with respect to the length and dimension of stochastic sequences in a multiplier and a Bernstein polynomial circuit. The results are shown in Figs. 3 and 4 repectively. The computed Bernstein polynomial is $f(x) = 6/11(x + 1/2x^2 + 1/3x^3)$ for Fig. 3(b).

As seen in Fig. 3, among SC designs, results produced by the LD sequences are consistently more accurate than that of LFSR-generated sequences using the maximal length.

For the multiplier, the RMSEs of the circuit using LD sequences decrease nearly linearly with the sequence length with the binary design as the reference line, while the output accuracy of the circuit using LFSRs converges much slower. To achieve a similar accuracy as an N-bit binary multiplier, Sobol sequences need 2^{N+2} bits and Halton sequences need approximately 2^{N+3} bits. It indicates that the Halton sequence-based multiplier takes about twice the sequence length to achieve a similar accuracy as the Sobol sequence-based design.

For the stochastic Bernstein polynomial circuits, the RMSE of the Sobol-based design is only slightly better than the Halton-based one when sequence length is smaller than 2^{10} bits. Otherwise, the performance is similar. However, both LD sequences outperform the LFSR-generated sequences. The accuracy of an N-bit binary design roughly matches that of a stochastic design using LD sequences of 2^{N+3} bits.



(a) Multiplier

(b) Bernstein polynomial

Fig. 3. Accuracy comparison: (a) a 2-input multiplier; (b) a third-order Bernstein polynomial, using different sequence lengths.

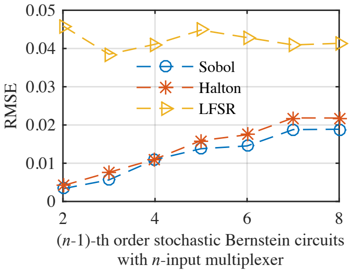


Fig. 4. Accuracy comparison: stochastic Bernstein polynomial circuits using different dimensions of random sequences with 256 bits.

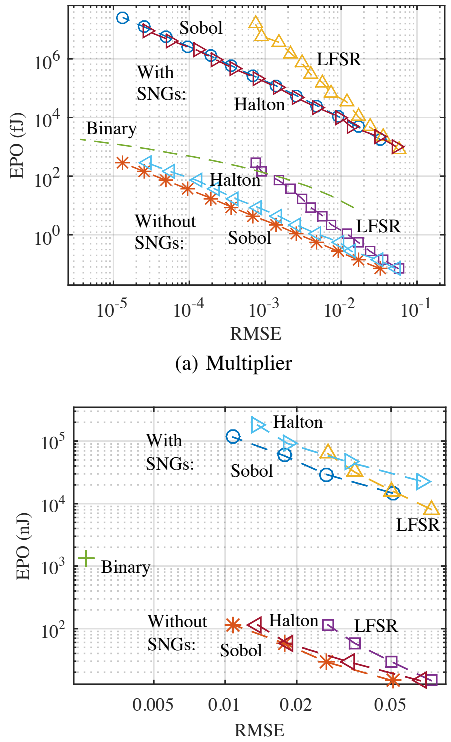
When the number of inputs or the dimension of stochastic sequences increases, as shown in Fig. 4, the output accuracy of the stochastic Bernstein polynomial circuit slightly decreases. This result is consistent with the theory of LD sequences. However, the output RMSE using LFSR-generated sequences is consistently larger than those using the LD sequences.

2) Hardware: For a fair comparison on hardware cost, EPO and TPA are measured against the same RMSE for two specific SC circuits: (1) a 2-input stochastic multiplier as a low dimensional SC circuit and (2) a 4-to-1 multiplexer along with an adder implementing the aforementioned third-order Bernstein polynomial as a relatively high dimensional circuit. In an SC circuit, both results for with and without the SNGs are reported. The EPO and TPA are obtained for using sequence lengths from 2^4 to 2^{16} for the multiplier and from 2^4 to 2^8 for the Bernstein polynomials circuit. For the binary designs, the bit width used is 4 to 16 for the multiplier, and 4 to 8 for the Bernstein polynomial circuit.

The results of EPO and TPA comparison are shown in Figs. 5 and 6 respectively. It can be seen that the SC circuits without considering SNGs mostly have a smaller energy consumption and a higher TPA than the binary circuits because SC arithmetic elements consist of only a few gates. The performance of SC elements are largely undermined by the costly SNGs. When the SNGs are counted, the SC designs have an inferior performance in both metrics for the same RMSE.

For the multiplier, when SNGs are considered, the higher cost of Sobol sequence generators results in a similar EPO and TPA compared to the use of Halton sequences, although half of the Sobol sequence length is sufficient to achieve a similar RMSE. Nevertheless, the performance of the Sobol-based multiplier is better than the one using Halton sequences due to the lower latency, without considering the SNGs.

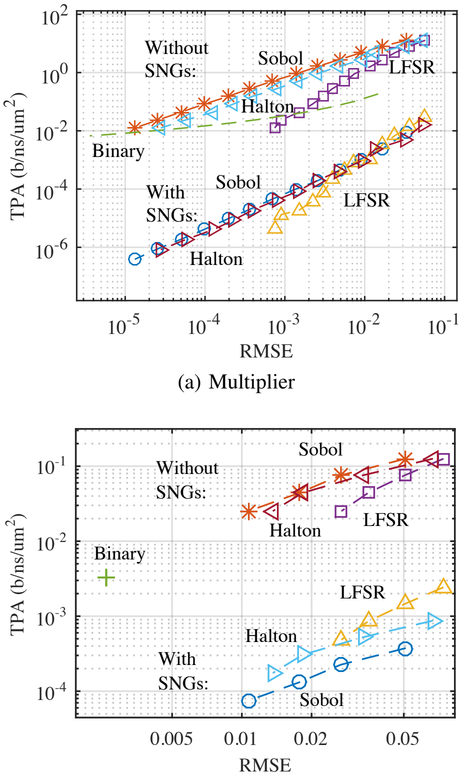
For the polynomial circuits, the design using Halton sequences consumes a higher energy than the one using Sobol sequences. This is due to the increased dimension for the Halton sequence generator. A higher dimension means a larger base that requires a higher hardware cost to convert a b-ary number to the binary representation. However, the Halton



(a) Multiplier

(b) Bernstein polynomial

Fig. 5. EPO comparison: (a) multiplier; (b) Bernstein polynomial.



(a) Multiplier

(b) Bernstein polynomial

Fig. 6. TPA comparison: (a) multiplier; (b) Bernstein polynomial.

sequence generator has a shorter critical path, rendering a larger TPA than the Sobol sequence-based design.

Generally, a Sobol sequence-based design consumes less energy (with a lower TPA when the SNGs are counted) than a Halton-based design for the same accuracy requirement, when multiple independent random number sequences are required. For a simple SC element such as a multiplier, the two LD sequences show similar performance. The scaled adder using a 2-to-1 multiplexer also shows a similar trend as the results for the multiplier, although they are not discussed in detail due to space limitations. For a design using LFSRs, it requires sequences that are several times longer to achieve the same accuracy, resulting in a larger energy consumption and a lower TPA in most cases.

IV. Conclusion

In this paper, the Sobol sequence is considered for improving the energy efficiency of SC circuits. For a simple computing element such as a multiplier, the energy consumptions of stochastic circuits using Sobol and Halton sequences are similar, especially when the SNGs are counted, while the computed results obtained by using Sobol sequences are more accurate. For circuits using multiple independent stochastic sequences such as a Bernstein polynomial circuit, the use of Sobol sequences provides a better energy efficiency than using Halton sequences due to the reduced sequence length and simpler SNGs.

References

[1] B. R. Gaines, *Stochastic Computing Systems*. Boston, MA: Springer US, 1969, pp. 37–172. [Online]. Available: http://dx.doi.org/10.1007/978-1-4899-5841-9_2

[2] W. Poppelbaum, C. Afuso, and J. Esch, "Stochastic computing elements and systems," in *Proceedings of the November 14-16, 1967, fall joint computer conference*. ACM, 1967, pp. 635–644.

[3] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. on Computers*, vol. 60, no. 1, pp. 93–105, 2011.

[4] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi, "A stochastic computational approach for accurate and efficient reliability evaluation," *IEEE Trans. on Computers*, vol. 63, no. 6, pp. 1336–1350, June 2014.

[5] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. on Embedded computing systems*, vol. 12, no. 2s, p. 92, 2013.

[6] J. H. Anderson, Y. Hara-Azumi, and S. Yamashita, "Effect of LFSR seeding, scrambling and feedback polynomial on stochastic computing accuracy," in *DATE*, pp. 1550–1555, 2016.

[7] A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *DATE*, article 76, 2014.

[8] H. Niederreiter, *Random Number Generation and quasi-Monte Carlo Methods*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

[9] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo methods*. John Wiley & Sons, 2013, vol. 706.

[10] I. L. Dalal, D. Stefan, and J. Harwayne-Gidansky, "Low discrepancy sequences for Monte Carlo simulations on reconfigurable platforms," in *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 108–113, 2008.

[11] P. Bratley and B. L. Fox, "Algorithm 659: Implementing Sobol's quasirandom sequence generator," *ACM Trans. on Mathematical Software (TOMS)*, vol. 14, no. 1, pp. 88–100, 1988.

[12] R. Wang, J. Han, B. F. Cockburn, and D. G. Elliott, "Stochastic circuit design and performance evaluation of vector quantization for different error measures," *IEEE Trans. on VLSI Systems*, vol. 24, no. 10, pp. 3169–3183, Oct 2016.