Dynamic Stochastic Computing for Digital Signal Processing Applications

Siting Liu
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, AB, Canada
siting2@ualberta.ca

Jie Han
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, AB, Canada
jhan8@ualberta.ca

***Abstract*—Stochastic computing (SC) utilizes a random binary bit stream to encode a number by counting the frequency of 1's in the stream (or sequence). Typically, a small circuit is used to perform a bit-wise logic operation on the stochastic sequences, which leads to significant hardware and power savings. Energy efficiency, however, is a challenge for SC due to the long sequences required for accurately encoding numbers. To overcome this challenge, we consider to use a stochastic sequence to encode a continuously variable signal instead of a number to achieve higher accuracy, higher energy efficiency and greater flexibility. Specifically, one single bit is used to encode a sample from a signal for efficient processing. This type of sequences encodes constantly variable values, so it is referred to as dynamic stochastic sequences (DSS's). The DSS enables the use of SC circuits to efficiently perform tasks such as frequency mixing and function estimation. It is shown that such a dynamic SC (DSC) system achieves savings up to 98.4% in energy and up to 96.8% in time with a slightly higher accuracy compared to conventional SC. It also achieves energy and time savings of up to 60% compared to a fixed-width binary implementation.**

***Index Terms*—stochastic computing, dynamic stochastic computing, dynamic stochastic sequence, frequency mixer, function estimation.**

I. Introduction

As the scaling of transistors continues, a variety of challenges emerge for the design of computing systems. Since emerging mobile applications such as wearable devices and self-driving cars, require embedded, low-power and high-performance processing units, energy efficiency has become a major concern. As an alternative computing paradigm, stochastic computing (SC) offers high computational density, low power and error tolerance, which can potentially help to overcome the aforementioned constraints [1].

In SC, a random binary bit stream or a stochastic sequence is used to encode a number. The arithmetic circuits that are used to process the sequences are area- and power-efficient because complex functions can be implemented by simple logic circuits to process one bit at each clock cycle. Due to this simplicity, SC has been considered in compute-intensive tasks such as reliability evaluation [2], [3], signal processing [4]–[6] and machine learning [7]–[11]. However, most of the designs rely on long stochastic bit streams to obtain a high accuracy, thus resulting in an inferior performance and low energy efficiency compared to conventional arithmetic circuits. To reduce the sequence length while maintaining a high accuracy, low-discrepancy (LD) sequences, including the Halton [12] and Sobol [13] sequences, have been considered. Although a relatively short LD sequence is sufficient to encode a number, the potential of SC has not yet been fully explored.

This performance bottleneck stems from the fundamental principle of the digital encoding of analog or continuous signals in SC, which, by itself, is likely overly ambitious to attain without efficient encoding techniques. In [14], [15], a $\Delta - \Sigma$ modulated bit stream is used as a stochastic sequence for filtering, multiplication and image processing. However, the $\Delta - \Sigma$ modulated bit streams may suffer from high signal correlations and degrade the accuracy of the results [15].

In this paper, we propose a new type of SC that uses a dynamic stochastic sequence (DSS) to encode a continuously variable signal rather than a static number. In such a sequence, the bit values show a dynamical pattern that constantly changes with the signal amplitude. This new encoding technique enables the use of single-bit SC circuits for low-power and high-performance computing of many digital signal processing (DSP) functions.

The contributions of this paper include the following.

1) A DSS is defined, based on which dynamic stochastic computing (DSC) is proposed for the first time.
2) The generation of a DSS is proposed and the reconstruction of DSS's is explicitly explained.
3) A stochastic multiplier is used to implement a frequency mixer using DSC.
4) A more complex function, the function composition of a Bernstein polynomial, is estimated by a multiplexing circuit in DSC.

II. Background

Typically, an SC system consists of three parts: the stochastic number generator (SNG) that converts binary numbers into stochastic sequences, the logic circuits processing the bit streams, and the probability estimator (PE) that converts the stochastic sequences back to binary numbers [16]. The SNG is usually implemented by a random number generator (RNG) and a comparator [1]; the logic circuits vary with the functions to be performed; the PE is often realized by using a counter.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) (Project Number: RES0025211). We thank CMC Microsystems for providing the simulation tools and technology library.

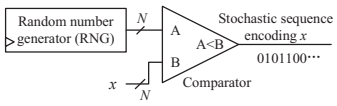


Fig. 1: A stochastic number generator (SNG).

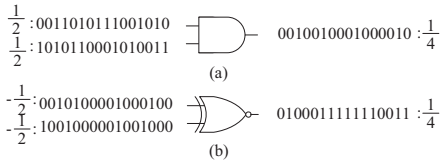


Fig. 2: (a) A unipolar and (b) a bipolar stochastic multiplier.

The circuit diagram of the SNG is shown in Fig. 1. A '1' is generated when a random number (RN) generated by the RNG is smaller than the number to be encoded, x, which is represented by an N-bit fixed-point fractional number. Since the RNs are uniformly distributed, the probability of obtaining a '1' is approximately the value of the number to be encoded, $x \in [0, 1]$. This encoding method is called the unipolar representation in SC. For a number, $x \in [-1, 1]$, the bipolar representation uses a linear mapping, $p = (x + 1)/2$, to scale the number to $p \in [0, 1]$ and then p is encoded by a stochastic sequence. To obtain a higher resolution and accuracy, a longer stochastic sequence is usually required.

The RNG in Fig. 1 can be implemented by a linear feedback shift register (LFSR) that generates pseudorandom numbers. Despite the inevitable bit correlation, the stochastic sequence generated by an LFSR is considered as approximately a Bernoulli sequence. Although the randomness in a Bernoulli sequence minimizes signal correlation, it leads to a rather wide distribution of the value encoded in the output stochastic sequence. A non-Bernoulli sequence is used to reduce the randomness in SC for reliability evaluation [2]. Recently, Halton and Sobol sequence generators are used to produce quasirandom numbers as the RNGs [12], [13]. The use of the LD sequences leads to a faster convergence of the results, thus achieving higher computation accuracy with shorter sequences.

The stochastic circuit can be combinational or sequential; a simple logic circuit can be used to implement a complex function. For example, an AND gate (or an XNOR gate) implements a unipolar (or bipolar) multiplier, as shown in Fig. 2. A multiplexing circuit has been used to implement Bernstein polynomials [17] and spectral transformation has been applied to generate a stochastic circuit that calculates a multi-linear polynomial [18]. There also exist sequential stochastic circuits including stochastic integrator-based and finite state machine (FSM)-based circuits [13].

The PE can be implemented by a counter to obtain the number of 1's in a sequence. This number is then divided by the sequence length. For ease of the division, the sequence length is usually selected to be a power of 2.

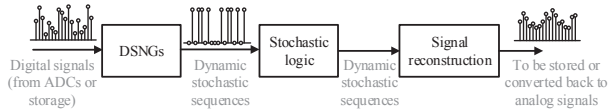


Fig. 3: A dynamic stochastic computing (DSC) system.

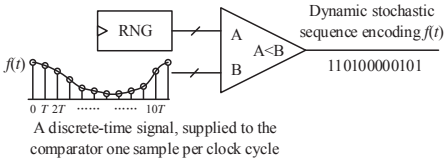


Fig. 4: A dynamic stochastic number generator (DSNG).

III. Dynamic Stochastic Computing (DSC) Systems

In a DSC system, a digital signal is first converted to a DSS by a dynamic stochastic number generator (DSNG). Then the bit sequence is processed by the stochastic circuits and converted back to a digital signal by a signal reconstruction unit, as shown in Fig. 3. The storage of a DSS is not required.

In this section, the generation of the DSS, the reconstruction of the output signal and the general aspects of DSC circuits are discussed.

A. Generation of the DSS

It is generally assumed that a digital signal is available from an analog-to-digital converter (ADC) or storage. If an analog signal is directly used as an input, the DSNG can be replaced with an analog design such as the SNG in [19].

Definition 1. Let $0 \leq f(t) \leq 1$ be a continuous signal. After a sampling with a clock period of T, it can be converted to a sequence of $\{f(kT)\}$, $k = 0, 1, \ldots$. A DSS $\{A_k\}$ encoding $f(t)$ satisfies that the kth bit in the random binary sequence has the expectation,

$$\mathbb{E}[A_k] = f(kT), \tag{1}$$

where T[1] is the sampling period and $1/T$ is the sampling rate. It is found in our experiments that a higher sampling rate generally leads to a higher encoding quality.

A DSS can be generated by comparing each element in the sequence ($\{f(kT)\}$) with a uniformly distributed RN within $[0, 1]$. Therefore, the expectation of the kth bit in a DSS is $f(kT)$. Also, DSS can also be generated by a $\Delta-\Sigma$ modulator. In this case, a $\Delta - \Sigma$ modulated signal can be considered to form approximately a Bernoulli sequence satisfying (1) [20]. Similar to a conventional SNG, the diagram of a DSNG is shown in Fig. 4. If the signal value is within $[-1, 1]$, a linear mapping can be applied for the bipolar representation.

B. Reconstruction of the DSS

The outputs of combinational stochastic circuits are still stochastic sequences. To convert a DSS back into a multi-bit digital signal, a reconstruction unit is required. The DSS

[1] T will be used as the notation for sampling period throughout this paper.

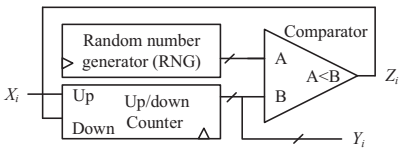


Fig. 5: An adaptive digital element (ADDIE).

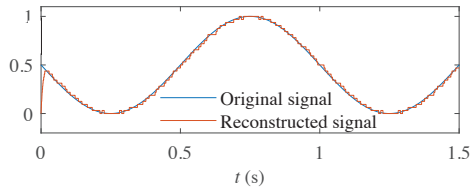


Fig. 6: Original and the reconstructed signals.

can be reconstructed to a multi-bit digital signal by using an exponential smoothing circuit [21]. Note that there exist other methodologies that can recover a 1-bit signal with better quality; however, this method is considered due to their ease of implementation in hardware and a relatively low hardware cost.

The exponential smoothing is implemented by an adaptive digital element (ADDIE) for a signal reconstruction [21], as shown in Fig. 5. Following [21], the exponential smoothing function is formulated as follows. Let the output of the comparator be Z_i (0 or 1) at clock cycle i, the input bit be X_i (0 or 1), and the multi-bit integer value stored in the counter be Y_i. The RNG and the comparator work as an SNG, so the probability of generating a '1' is $Y_i/2^N$, where N is the bit-width of the counter, i.e., $y_i = \mathbb{E}[Z_i] = Y_i/2^N$. As per the function of the up/down counter, $Y_{i+1} = Y_i + X_i - Z_i$ [14]. Therefore, given Y_i, the expectation of Y_{i+1} is

$$\mathbb{E}[Y_{i+1}] = (2^N - 1)Y_i/2^N + \mathbb{E}[X_i] \tag{2}$$

by taking expectations of X_i and Z_i. Assume Y_0 is 0, and let $Y_{i+1} \approx \mathbb{E}[Y_{i+1}]$ for $i = 0, 1, \ldots$, (2) can be rewritten as

$$y_{i+1} = \frac{1}{2^N} Y_{i+1} \approx \frac{1}{2^N} \sum_{k=0}^{i} \left[\left(\frac{2^N - 1}{2^N} \right)^k \mathbb{E}[X_{i-k}] \right], \tag{3}$$

where the coefficients of $\{\mathbb{E}[X_{i-k}]\}$ form a geometric sequence. It indicates an exponential smoothing. Then, $\{y_i\}$ is an estimate of the encoded signal. However, to reconstruct the signal instead of filtering it, the width of the counter N needs to be carefully selected because the same circuit can be used as a low-pass IIR filter [14], which may attenuate the encoded signal. An optimal width N can be obtained regarding the sampling rate and the frequency of the signal. A sinusoidal signal reconstructed by an ADDIE is shown in Fig. 6. At the start of the signal, a warm-up phase is required to allow the integrator to follow the signal if it is initialized with a random value other than the actual initial value.

C. DSC circuits

DSC circuits are efficient and flexible to implement a series of function compositions. For example, a combinational SC circuit implementing the function $f(x)$ can be used to implement the function composition $f[\chi(t)]$ with the input DSS encoding the signal $\chi(t)$. This principle also applies to multi-input combinational circuits.

For a finite-state machine (FSM)-based sequential circuit [22], however, the output does not accurately encode $f[\chi(t)]$ because an FSM-based circuit requires temporal independency within a sequence. The adjacent bits in a DSS are correlated and violate this condition [13]. On the other hand, a stochastic integrator does not require temporal independency due to the accumulation of bit values, so it works well with the DSS's.

IV. DSC-based Digital Signal Processing (DSP)

As first examples, two DSP functions, frequency mixing and function estimation, are considered as applications of DSC. These applications can be extended to include other functions that involve various arithmetic operations.

A. Frequency Mixer

Conventionally, a frequency mixer is implemented by an analog multiplier consisting of nonlinear components. New signals at the summation and difference of the original frequencies are then produced. Using the DSS's, a stochastic multiplier is proposed to implement a frequency mixer, as shown in Fig. 7(a). As per (1), if the input sequences X and Y are statistically independent and X_k, Y_k and Z_k are the kth bits in the sequences X, Y and Z, respectively, we obtain $\mathbb{E}[Z_k] = \mathbb{E}[X_k]\mathbb{E}[Y_k] = x(kT)y(kT)$ at clock cycle k for the output sequence. Therefore, the output sequence Z encodes $z(t) = x(t)y(t)$, which is the product of the two input signals.

Fig. 7(b) shows the output results for multiplying two sinusoidal signals with frequencies of 1 Hz and 6 Hz, both sampled at a rate of 2^{14} Hz. The DSS is reconstructed to a multi-bit digital signal by a 5-bit ADDIE.

As shown in Fig. 7(b), the results produced by the stochastic circuit are very close to the results produced by using double precision numbers with a signal-to-noise ratio[2] (SNR) of 22.6 dB. It also shows that there are more 1's (or 0's) in the DSS's when the original signal is closer to 1 (or 0). The DSS's are decimated for a clear view.

B. Approximation of functions

Bernstein or multilinear polynomials have been implemented in SC either by a multiplexing circuit or a Boolean function with auxiliary inputs [17], [18]. By using the DSS's, more complex functions can be implemented with the same circuit. The multiplexing circuit consisting of an accumulator and a multiplexer is discussed as an example. Fig. 8(a) shows a multiplexing circuit that computes a Bernstein polynomial, $f(x) = 1/11(2x^3 + 3x^2 + 6x)$, where all the input sequences, i.e.,

[2]SNR=$10\log_{10}(\mathbb{E}[z_n^2]/\mathbb{E}[e_n^2])$, where $\{z_n\}$ is the target signal while $\{e_n\}$ is the difference between the estimated signal and the target signal.

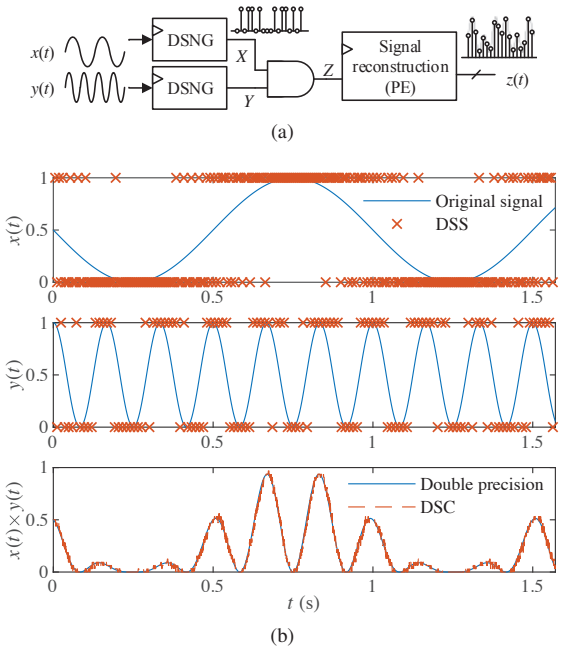


Fig. 7: A frequency mixer by using a stochastic multiplier with DSS's as the inputs, (a) circuit and (b) results.

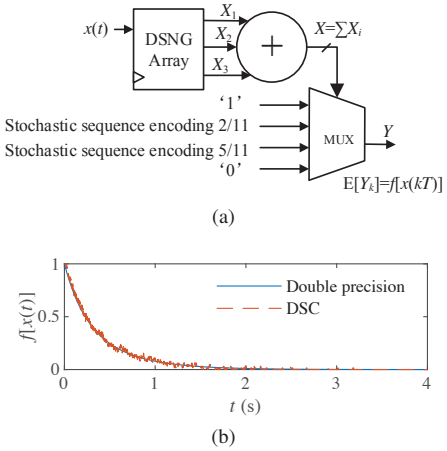


Fig. 8: A summation of several exponential functions computed by (a) a multiplexing circuit and (b) the output results.

X_1, X_2, X_3, to the accumulator independently encode a fixed value x. However, the same circuit can be used to compute the function composition, $f[x(t)]$, when DSS's encoding $x(t)$ serve as the inputs to the accumulator, i.e., $\mathbb{E}[X_{1,k}] = \mathbb{E}[X_{2,k}] = \mathbb{E}[X_{3,k}] = x(kT)$. The expectation of the kth bit in the output sequence Y is then $\mathbb{E}[Y_k] = \sum_{i=0}^{n} b_i \binom{n}{i} x^i(kT)(1-x(kT))^{n-i}$, where $\{b_i\}$ are the Bernstein coefficients ($\{1, 2/11, 5/11, 0\}$ in Fig. 8(a)) encoded by the inputs of the multiplexer and n is the order of the Bernstein polynomial [17]. The value of the function, $f[x(kT)]$, is then equal to $\mathbb{E}[Y_k]$, so the output sequence Y encodes $f[x(t)]$.

When $x(t) = e^{-2t}$, the output sequence of the multiplexer encodes $f[x(t)] = 1/11(2e^{-6t} + 3e^{-4t} + 6e^{-2t})$. Fig. 8(b) shows the reconstructed result by using a 6-bit ADDIE and a sampling rate of 2^{12} Hz for the input DSS. The reconstructed signal has an SNR of 23.6 dB. The sequences for the coefficients of the Bernstein polynomial (i.e., the inputs to the multiplexer) are generated by an SNG as in a conventional SC (CSC) circuit.

V. Hardware Efficiency Assessment

In this section, the hardware efficiency is evaluated for the frequency mixer and the function estimator. The area, power and critical path delay of all considered circuits are measured using the Synopsys Design Compiler with a 28-nm STM process by the default high-effort optimization for overall performance. The temperature is set to 25°C and the supply voltage is 1.0 V. Sobol sequences are used to generate the DSS's for a better accuracy. For the stochastic designs, the RNG (Sobol sequence generator) can be shared to generate multiple stochastic sequences, so the cost is negligible when considering a large SC system and it is not included in the hardware evaluation (as an example, only two independent RNGs are required to generate the stochastic sequences for performing many multiplications in parallel in [23]). However, the hardware cost of comparators are counted as the DSNG since every signal requires a comparator to generate a DSS and they cannot be shared. The signal reconstruction units/PEs are also counted for the dynamic/conventional stochastic designs.

A. Frequency mixer

Two sinusoidal signals with frequencies of 1 Hz and 6 Hz are multiplied to measure the SNR of the results produced by circuits as shown in Fig. 7(a). With a sampling rate of 2^{16} Hz (resulting in a dynamic sequence length of $k \cdot 2^{16}$ bits for a k-second signal) and a 5-bit ADDIE, the DSC circuit produces an SNR of 24.20 dB, as shown in Figure 9(a).

For the CSC circuit, the same sampling rate and the same LD sequences are applied. With a sequence of 2^5 bits encoding each sample, the CSC circuit produces an SNR of 23.3 dB, as shown in Fig. 9(b). A binary implementation using a 5-bit fixed-width multiplier is also considered and the result is shown in Fig. 9(c).

As shown in Fig. 9, the results produced by the DSC circuit show stronger variations because the signal reconstructor consistently tracks the input signal, which makes it very sensitive to the change in the input sequence. The DSC frequency mixer produces a similar or a slightly higher SNR than the CSC circuit. However, due to the effect of limited precision, the fixed-width multiplication by a binary circuit produces the lowest quality.

The hardware evaluation results are shown in Table I. As can be seen, the DSC circuits have a slightly higher area and power consumption than the conventional stochastic circuit due to the use of ADDIE-based signal reconstructor, while the SNG (comparator) and the multiplier are the same for those two circuits. However, this disadvantage is negligible

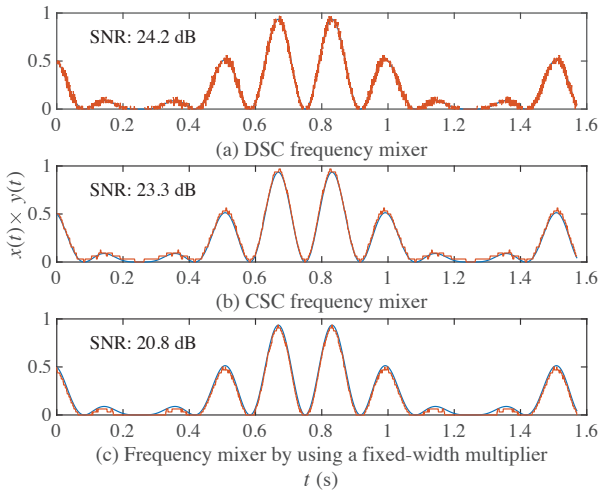


Fig. 9: Frequency mixer: dynamic SC (DSC) vs. conventional SC (CSC) vs. fixed-width binary implementation.

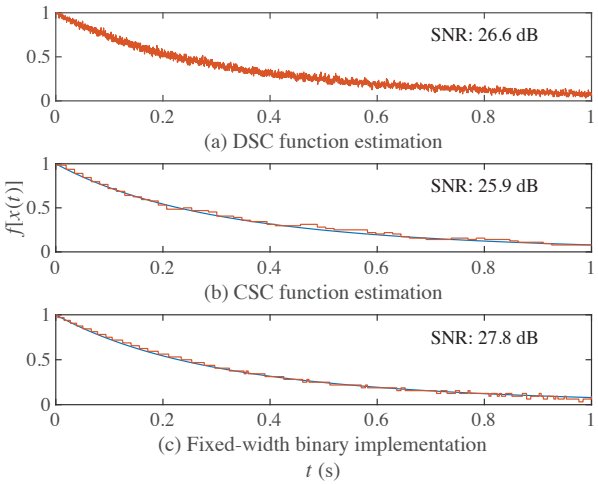


Fig. 10: Function estimator: dynamic SC (DSC) vs. conventional SC (CSC) vs. fixed-width binary implementation.

TABLE I: Hardware efficiency evaluation of DSC, CSC and binary frequency mixers producing one result at one sampling point.

	Area (μm^2)	Power (μW)	No. of clk cycles	Minimum time (ns)	Energy (fJ)	SNR (dB)
DSC	82	13	1	0.7	51	24.2
CSC	67	10	32	9.3	1315	23.3
Binary	80	14	1	0.7	55	20.8
Ratio[a]	0.82	0.77	32	13.29	25.78	–

[a]The ratio of CSC:DSC.

TABLE II: Hardware efficiency evaluation of DSC, CSC and binary function estimators producing one result at one sampling point.

	Area (μm^2)	Power (μW)	No. of clk cycles	Minimum time (ns)	Energy (fJ)	SNR (dB)
DSC	176	21	1	0.7	82	26.6
CSC	159	20	64	21.8	5125	25.9
Binary	302	51	1	1.8	203	27.8
Ratio[b]	0.90	0.95	64	31.14	62.50	–

[b]The ratio of CSC:DSC.

when considering the dominating factor, the sequence length or the number of clock cycles for producing one result: it is 2^5 for CSC, whereas it is only 1 for DSC. Due to the significant reduction in sequence length, the DSC frequency mixer consumes only 3.9% of the energy and 7.5% of the time required by the CSC design with a similar accuracy.

Compared to the fixed-width binary design, the DSC circuit has a slightly larger hardware cost, while it requires a lower energy and the same time to produce the result. Also, the SNR of the results produced by the proposed method exceeds that of the binary design by 3.4 dB. However, when a signal is sampled at a lower rate (e.g., at the Nyquist frequency of $(1+6)\times 2 = 14$ Hz in this case), the binary circuit would achieve a much lower latency and lower energy cost with a higher accuracy, while DSC is less effective at a relatively low sampling rate[3].

B. Function estimator using multiplexing circuits

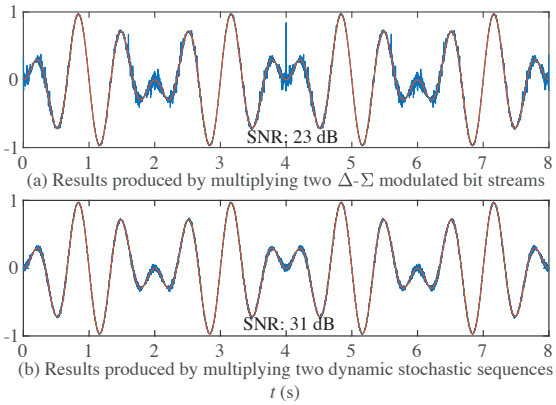
The function $y(t) = 1/11(2e^{-6t} + 3e^{-4t} + 6e^{-2t})$ is used for the accuracy evaluation of the dynamic and conventional SC systems. For the DSC system, the multiplexing circuit in Fig. 8(a) is used to perform the function estimation with DSS's encoding $x(t) = e^{-2t}$ as the inputs. With a sampling rate of 2^{16} Hz and a 6-bit ADDIE as the signal reconstructor, the SNR produced by the DSC system is 26.6 dB, as shown in Fig. 10(a).

For the CSC system, a multiplexing circuit proposed in [17] is used to approximate the function with a minimum-order Bernstein polynomial to reduce the hardware cost of the SNGs (comparators), which constitutes a major part of the SC system [24]. The Bernstein polynomial that is used to approximate $y(t)$ is optimized as $y(t) = 17/256t^3 + 159/256t^2(1-t) + 66/256t(1-t)^2 + 249/256(1-t)^3$. Using a sequence length of 64, the SNR is 25.9 dB, as shown in Fig. 10(b).

A fixed-width binary circuit is also considered to implement the polynomial using binary adders and multipliers. The binary circuit is optimized to use the least number of multipliers to reduce the hardware cost. The resulting 6-bit binary design produces an SNR of 27.8 dB, as shown in Fig. 10(c), which is slightly higher than that of the result produced by DSC.

The hardware evaluation of the two SC and the binary circuits is reported in Table II. As can be seen, the DSC circuit has a slightly higher hardware cost but with a similar power consumption compared to the CSC circuit. However, for the CSC circuit, the long sequence undermines the performance and energy efficiency. The DSC takes only about 3.2% of the time and 1.6% of the energy of the CSC to achieve a higher accuracy and attains about 60% savings in energy and time compared to the binary circuit.

[3]Our experiments indicate that the oversampling rate needs to be at least approximately 100 times of the Nyquist frequency to obtain a reconstructed signal with an SNR over 20 dB using a moving average filter.



(a) Results produced by multiplying two Δ-Σ modulated bit streams

(b) Results produced by multiplying two dynamic stochastic sequences

Fig. 11: $\Delta - \Sigma$ modulated vs. dynamic stochastic sequences for a frequency mixer multiplying signals at 0.5 Hz and 3 Hz. The sequences are generated by using the same sampling rate.

VI. Discussion

DSS's and $\Delta - \Sigma$ modulated (DSM) bit streams are similar in the sense that the generation of the streams can both be considered as Bernoulli processes [25]. However, the DSC can avoid correlation issues by using independent random sequences, whereas correlation exists in the DSM signals, which can seriously degrade the accuracy [15]. For example, for multiplying two signals with different frequencies, the DSC produces a much higher accuracy than using the DSM signals, as shown in Fig. 11. The DSC can also compute the product of two identical signals with the same frequency. However, multiplying two DSM signals encoding identical values can result in the original signal instead of their product [15]. Finally, DSC can be used to encode a stochastic signal such as the gradient in the training of a neural network [26], while it is not feasible to use $\Delta - \Sigma$ modulation to encode such signals because the input of a $\Delta - \Sigma$ modulator is usually an analog signal or an up-sampled digital signal [27].

VII. Conclusion

In this paper, dynamic stochastic computing (DSC) is proposed that leverages an efficient encoding technique using dynamic stochastic sequences (DSS's) and simplistic digital circuits to implement complex DSP functions. Frequency mixing and function approximation are implemented by using a stochastic multiplier and a multiplexing circuit, respectively. The generation and reconstruction of the DSS's are discussed. Compared to CSC, the proposed DSC can achieve a speedup and an energy efficiency improvement by more than 13× and 25×, respectively, with a better accuracy for signal multiplication. For function estimation, the improvement is even larger, which are 31× and 62×, respectively. With a similar accuracy, DSC also achieves a saving in energy and time by 60% compared to conventional binary circuits, when dealing with complex tasks such as function estimation. Theories of DSC will further be investigated in future work.

References

[1] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Trans. on CAD*, vol. 37, no. 8, pp. 1515–1531, Aug 2018.

[2] 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.

[3] P. Zhu, J. Han, L. Liu, and M. J. Zuo, "A stochastic approach for the analysis of fault trees with priority AND gates," *IEEE Trans. on Reliability*, vol. 63, no. 2, pp. 480–494, June 2014.

[4] N. Onizawa, D. Katagiri, K. Matsumiya, W. J. Gross, and T. Hanyu, "Gabor filter based on stochastic computation," *IEEE Signal Processing Letters*, vol. 22, no. 9, pp. 1224–1228, Sept 2015.

[5] Y. Liu and K. K. Parhi, "Architectures for recursive digital filters using stochastic computing," *IEEE Trans. on Signal Processing*, vol. 64, no. 14, pp. 3705–3718, July 2016.

[6] B. Yuan, Y. Wang, and Z. Wang, "Area-efficient scaling-free DFT/FFT design using stochastic computing," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 63, no. 12, pp. 1131–1135, Dec 2016.

[7] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Trans. on VLSI Systems*, vol. 25, no. 10, pp. 2688–2699, Oct 2017.

[8] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, "SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing," in *ASPLOS*, 2017.

[9] H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *DAC*, 2017.

[10] B. D. Brown and H. C. Card, "Stochastic neural computation. I. computational elements," *IEEE Trans. on Computers*, vol. 50, no. 9, pp. 891–905, Sept 2001.

[11] J. S. Friedman, L. E. Calvet, P. Bessire, J. Droulez, and D. Querlioz, "Bayesian inference with Muller C-Elements," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 63, no. 6, pp. 895–904, June 2016.

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

[13] S. Liu and J. Han, "Toward energy-efficient stochastic circuits using parallel Sobol sequences," *IEEE Trans. on VLSI Systems*, vol. 26, no. 7, pp. 1326–1339, July 2018.

[14] N. Saraf, K. Bazargan, D. J. Lilja, and M. D. Riedel, "IIR filters using stochastic arithmetic," in *DATE*, 2014.

[15] P. Gonzalez-Guerrero, X. Guo, and M. Stan, "SC-SD: Towards low power stochastic computing using sigma delta streams," in *ICRC*, 2018.

[16] R. Wang, J. Han, B. F. Cockburn, and D. G. Elliott, "Design, evaluation and fault-tolerance analysis of stochastic FIR filters," *Microelectronics Reliability*, vol. 57, pp. 111 – 127, 2016.

[17] 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, Jan 2011.

[18] A. Alaghi and J. P. Hayes, "STRAUSS: Spectral transform use in stochastic circuit synthesis," *IEEE Trans. on CAD*, vol. 34, no. 11, pp. 1770–1783, Nov 2015.

[19] A. Alaghi and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *DAC*, 2013.

[20] V. da Fonte Dias, "Sigma-delta signal processing," in *ISCAS*, 1994.

[21] P. Mars and W. J. Poppelbaum, *Stochastic and deterministic averaging processors*. Peter Peregrinus Press, 1981, no. 1.

[22] P. Li, W. Qian, M. D. Riedel, K. Bazargan, and D. J. Lilja, "The synthesis of linear finite state machine-based stochastic computational elements," in *ASP-DAC*, 2012.

[23] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A stochastic computational multi-layer perceptron with backward propagation," *IEEE Trans. on Computers*, vol. 67, no. 9, pp. 1273–1286, Sept 2018.

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

[25] F. Maloberti, "Non conventional signal processing by the use of sigma delta technique: a tutorial introduction," in *ISCAS*, 1992.

[26] S. Liu, H. Jiang, L. Liu, and J. Han, "Gradient descent using stochastic circuits for efficient training of learning machines," *IEEE Trans. on CAD*, vol. 37, no. 11, pp. 2530–2541, Nov 2018.

[27] E. Janssen and A. van Roermund, *Look-Ahead Based Sigma-Delta Modulation*, 1st ed. Springer Publishing Company, Incorporated, 2011.