Delta Sigma Modulator-based Dividers for Accurate and Low Latency Stochastic Computing Systems

Xiaochen Tang, Shanshan Liu, Farzad Niknia, Ziheng Wang, Siting Liu, Pedro Reviriego, and Fabrizio Lombardi

***Abstract*—Stochastic computing (SC) has received considerable research interest in the past decade. Significant efforts have been devoted to reducing computation latency for the stochastic divider, which is the most complex unit in SC. However, current SC systems still lack dividers that can timely operate with other SC units by aligned processing periods. Moreover, all existing stochastic dividers cannot perform accurate division for input values near the center of the SC computation range. This paper proposes two Delta Sigma Modulator (DSM) based stochastic dividers. The proposed first-order DSM-based divider significantly reduces the additional clock cycles needed for division, and also slightly increases the accuracy (e.g., compared with the fastest existing divider of 10-bit resolution, a reduction of 87.5% in the number of additional clock cycles is accomplished, with an average mean square error (MSE) that is decreased from $10^{-3.9}$ to $10^{-4.0}$). Moreover, a fully compatible second-order DSM-based divider is proposed. It achieves a higher division accuracy (e.g., MSE of $10^{-4.7}$ for 10-bit resolution) and does not require additional clock cycles, at the cost of a slightly increased hardware overhead. As an emerging application, SC-based neural networks are implemented as a case study to evaluate the advantages of the proposed designs. The synthesis results show that compared to the network implementation with the most efficient existing stochastic divider, the use of the proposed dividers reduces the total hardware overhead of the network by 32.0% to 46.6%, and slightly improves the classification accuracy. Overall, the proposed divider designs enable an SC system to operate with aligned timing, so resulting in a better implementation.**

***Index Terms*—Stochastic computing, divider, Delta Sigma Modulator, neural network, compatible SC units.**

I. Introduction

As a large volume of data is collected, technologies such as Machine Learning (ML) and Artificial Intelligence (AI), utilize more complicated algorithms than ever. Thus, computation complexity of hardware plays a crucial role in current computing systems. Chip area and power dissipation are becoming critical for ML/AI-based designs. Especially for some applications that perform data processing on the edge of the network, such as Internet-of-Things (IoT) devices, a low complexity hardware design is preferred. In such devices, the hardware and power are limited, and conventional arithmetic logic units may prevent the use of large-scale algorithms.

As a promising solution to address these concerns, stochastic computing (SC) has aroused research interest for its low hardware complexity implementation and inherent error tolerance [1]. For example, an SC scheme has been adopted for implementing neural network accelerators [2], [3]. In an SC system, real numbers are represented by stochastic sequences that are binary coded (i.e., composed of "1" and "0"), and their values are converted to the occurrence probability of "1" within the sequences. Since computations are performed based on sequences, SC arithmetic logic is rather simple [4], [5]; for example, a stochastic multiplier is implemented using an AND gate for the unipolar representation or an XNOR gate for the bipolar representation. Therefore, the use of SC enables high hardware efficiency for implementing systems that consist of a large amount of arithmetic computations, such as neural networks [2], [3], digital filters [6], [7], image processing [8], and decoders for error control codes [9], [10].

Among the different SC arithmetic blocks, the stochastic divider is regarded as challenging and crucial because of the significant latency it introduces. For an N-bit resolution SC system (i.e., the occurrence probability of "1" in sequences is represented using a N-bit number), the computing period of most SC elements such as the adders and multipliers is equal to the length of the stochastic sequence, so given by 2^N clock cycles. However, a stochastic divider usually requires more cycles to generate the corresponding quotient sequence, or a large deviation from the correct quotient may appear [5]. This process significantly increases the entire computational latency of an SC system; it may also introduce in the design a rather complicated timing due to the different numbers of clock cycles required by the divider and other SC units. Even though improved schemes for reducing computational latency have been designed by calculating the quotient region by region [11], [12] or utilizing a parallel scheme [13], the required number of clock cycles is still considerably larger than for the remaining blocks of an SC system.

In addition to latency, computational accuracy is another important issue of existing stochastic dividers. The mechanism of performing division in these designs is based on the multiplication of the divisor and the quotient-tracking dividend sequences. Hence, the limited accuracy is originated from the process of inefficiently adjusting unmatched errors between the two sequences. Moreover, the inaccurate computation of

Manuscript received October 14, 2022, revised December 19, 2022 and January 16, 2023. The work was supported by NSF under Grant 1953961, and by the Spanish Agencia Estatal de Investigation under Grant PID2019-104207RB-I00 and Grant TSI-063000-2021-127. *Corresponding author: Shanshan Liu* (email: ssliu@nmsu.edu).

Xiaochen Tang and Shanshan Liu are with the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM 88001, USA.

Farzad Niknia , Ziheng Wang and Fabrizio Lombardi are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02215, USA.

Siting Liu is with School of Information Science and Technology, Shanghai Tech University, Shanghai, 201210, China.

Pedro Reviriego is with the Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 28040 Madrid, Spain.

stochastic multipliers for inputs near the center of the SC computation range [14] also leads to an accuracy loss in the divider, when the divisor and dividend are in such value region.

Recently, a fast-converging divider based on a Delta sigma modulator (DSM) has been proposed [15]; this circuit operates on an efficient error adjustment process that is different from the traditional process. It provides significant improvement in both number of clock cycles and average computational accuracy compared to other stochastic dividers. However, the accuracy issue when dividend and divisor values are near the center of computation range, as well as the incompatibility issue in timing for the divider with other SC elements due to the additional clock cycles, are still not fully resolved.

The above challenges of existing dividers motivate this paper to investigate stochastic divider designs with less processing time and compatibility with other elements for implementing an SC system. Moreover, when utilizing such short sequences, the proposed dividers still achieve a very high division accuracy. The main contributions of this paper are as follows:

- An improved first-order DSM-based stochastic divider (named as DSM-U1) is proposed to perform an accurate division and generate shorter sequences compared to the existing DSM divider (named as DSM-PRE), which is the most efficient design among all existing dividers. For example, when N = 10 the proposed design reduces the required number of clock cycles from 1428 to 1040 (i.e., a reduction of 87.5% in the number of additional clock cycles compared to 2^N as sequence length); it also improves the average mean square error (MSE) of computation from $10^{-3.9}$ to $10^{-4.0}$.
- A second-order DSM-based divider (named as DSM-U2) is then proposed. It achieves a fully timing compatible stochastic unit with a very competitive computing accuracy, especially for divisor and dividend values centered around the SC range. For example, it requires precisely 1024 clock cycles for N = 10 and achieves an MSE of $10^{-4.7}$.
- The mechanism of error suppression of a DSM-based divider is analytically investigated; this proves that the proposed DSM-U2 provides high accuracy using shorter sequences.
- Extensive simulations using both evenly and Gaussian distributed datasets (with different standard deviation values) are conducted to evaluate the computational performance of the proposed dividers under different conditions.
- SC-based neural networks are implemented as an application to assess the advantages of the proposed dividers. Evaluation results show that when employing the proposed dividers, the SC network with DSM-U2 (DSM-U1) achieves a reduction of more than 35.9% (32.0%) in the number of clock cycles compared to a network with DSM-PRE.

The rest of the paper is organized as follows. In Section II, the SC representation and essential SC elements are briefly reviewed. Section III presents the proposed DSM-based dividers and analyzes their error suppression mechanisms. Then, the performance of the proposed designs is evaluated and compared with existing dividers in Section IV. In Section V, to assess the advantages and evaluate the performance of the proposed designs, SC-based neural networks with the proposed dividers are implemented as examples. Finally, the paper ends in Section VI with the conclusion.

II. Preliminaries

A. Stochastic Sequence Representation

In an SC system, a real number X can be encoded in the unipolar or bipolar stochastic sequence representations within the value range of [0, 1] or [-1, 1], respectively. Define the number of bits "1" as q and the length of a stochastic sequence x as 2^N; the real number X can be represented by $X = p(x) = q/2^N$ for the unipolar representation and $X = 2 \cdot p(x)\text{-}1 = (2 \cdot q\text{-}2^N)/2^N$ for the bipolar representation, where $p(x)$ is the occurrence probability of "1" in sequence x. Moreover, an additional sign bit associated with a unipolar stochastic sequence can also be used to represent a real number in [-1, 1], which is proposed for obtaining more accurate calculation results [16]. In general, SC utilizes sequences with no or low correlation to achieve a high computational accuracy [5]. Even though the correlation between multiple sequences has also been studied for computation [17], there is likely a significant difficulty when employing correlated sequences and implementing a large-size SC system. Therefore, an uncorrelated-based SC is targeted in this paper, and the correlated-based dividers (such as found in [18]-[21]) are not considered for comparison.

The limited computational range of SC may result in an inaccurate result for some applications. To address this issue, the extended stochastic logic (ESL) has been proposed [22]; ESL utilizes the quotient of two stochastic sequences to represent a real number. Although both the numerator (x_h) and the denominator sequences (x_l) still represent numbers within the range of [0, 1] or [-1, 1], the range of the quotient x_h/x_l can be extended to [0, 2^N) or (-2^{N-1}, 2^{N-1}) for the unipolar and bipolar representations, respectively; this is correct because X can be considered as $\frac{\pm 1}{1/X}$ when it is smaller than -1 or larger than 1, so essentially extending the value range of SC. However, the use of two sequences also doubles the circuit size of the SC units. Therefore, a trade-off is often pursued between computational accuracy and hardware. For example, ESL units are only used to perform partial computations in an SC system and the remaining parts are calculated using standard SC units; this is explained in more detail in the case study presented in Section V.

B. Basic Stochastic Logic Elements

Prior to performing stochastic computation, real numbers are converted to stochastic sequences by using the stochastic number generator (SNG, shown in Fig. 1). For a probability $p(x)$ (N-bit binary number), pseudorandom numbers from 0 to 2^N-1 are generated by a random number generator (RNG) and compared with $p(x)$; if the pseudorandom number is smaller, a "1" is generated, and vice versa. After the entire conversion period of 2^N clock cycles, the stochastic sequence representing $p(x)$ is generated. The RNG is usually implemented by a linear feedback shift register (LFSR) or low-discrepancy (LD) sequences (e.g., Halton or Sobol sequences) [13], and the location of bits "1" and "0" is usually observed as evenly

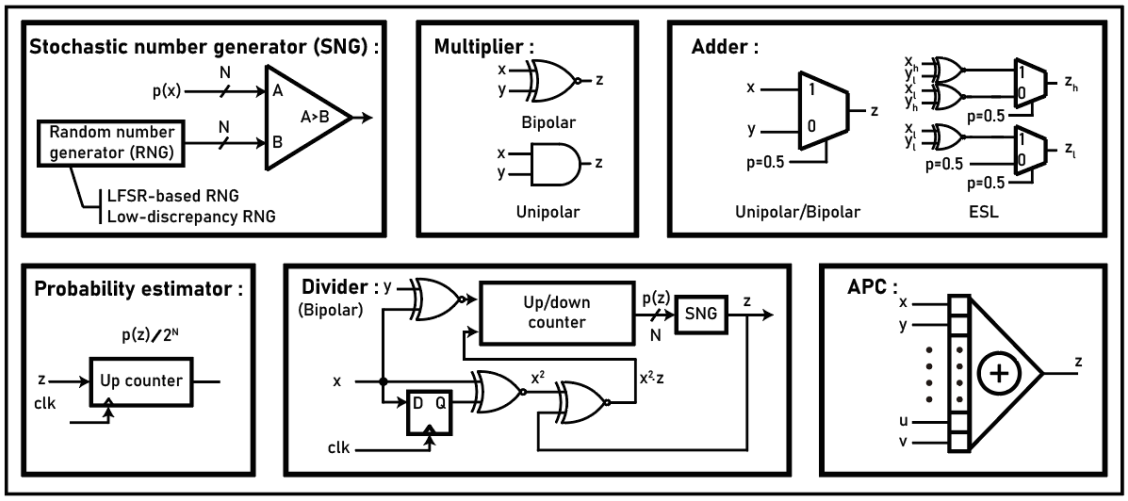


Figure 1. Stochastic computing elements (x, y, z, u, v, x_h, x_l, y_h, y_l, z_h, z_l are stochastic sequences).

distributed [5]. Compared with an LFSR-based stochastic sequence, an LD-based sequence performs better in terms of accuracy for combinational and integrator-based SC units (e.g., an adder, multiplier, or divider). Therefore, when achieving the same computational accuracy for these units, a Sobol-based sequence tends to require a shorter length, i.e., a smaller number of clock cycles to complete the computation. However, Sobol-based sequences are not ready suitable to some SC units based on Finite State Machines (FSM) [13] that are usually used to implement the activation functions of an SC neural network. Additionally, deterministic sequences with arranged positions of "1" bits can be used for computation with a very high accuracy, but they usually require a significantly longer sequence length [23], [24].

Once the stochastic computation is completed, the so-called probability estimator (PE) estimates the binary numbers from the sequences, which can be seen as the opposite process of the SNG. As shown in Fig. 1, a PE can be implemented by a counter. A stochastic multiplier is implemented by an AND (XNOR) gate for the unipolar (bipolar) representation; the ESL version of multiplier uses two AND or XNOR gates in each case. A typical unipolar/bipolar stochastic adder is implemented by a multiplexer; however, such an adder generates an output that scales the addition result by half. To address this issue, the ESL version of the adder can be used, because the scaling issue does not occur when appearing on both the numerator and denominator. An alternative solution is the use of an accumulation parallel counter (APC). APC calculates the sum of bits on the same position of paralleled input sequences and generates a result sequence that consists of binary values [25]. The stochastic divider and improved designs are reviewed in the next subsection.

C. Stochastic Dividers

A conventional stochastic divider in a binary representation is also shown in Fig. 1. It is formed by a feedback loop that computes $x^2 \cdot z$ to keep track of $x \cdot y$, where z is the quotient sequence of y/x; therefore, the quotient is obtained once the loop converges and the result sequence consists of the last 2^N bits. However, generating an accurate result requires a very long period to converge; for example, for an N = 10 LFSR-based sequence, the simulated convergence requires 46341 clock cycles [12]. This number is significantly larger than for other units (such as an adder and multiplier) that require only 2^N clock cycles. This will cause a very large computational latency for the entire SC system when different elements are involved. Several techniques have been proposed to shorten the division period, as introduced next.[1]

1) Triple Modular Redundancy Based Divider

The triple modular redundancy (TMR) technique is applied to reduce the latency introduced by the divider convergence process [11]. The binary searching TMR (BS-TMR) technique has been proposed to reduce latency by estimating the quotient probability of the up/down counter in binary ranges. In such case, the entire calculation process includes N steps of iterations of probability estimate for an N-bit quotient probability. Within each iteration step, three up/down counters with three SNGs (i.e., TMR blocks) are employed in parallel. After each iteration period, a voter decides the corresponding bit of the quotient. Thus, the N-step iteration process can estimate the quotient from the most significant to the least significant bits. After the estimate process, a stabilization phase is required by utilizing additional clock cycles for the final fine-tuning of the quotient; this is performed using a conventional divider by disabling two of the three TMR blocks.

Similarly, a decimal searching TMR (DS-TMR) divider is proposed to reduce the latency further [12]. Compared to the BS-TMR, the iteration step of DS-TMR is decided by the system's required resolution. For example, if N = 10, the resolution of the quotient is 10^{-3}; thus, the iteration is set to 3 steps, within each step ten sections of quotient probability are calculated in parallel to decide the corresponding decimal bit. The DS-TMR also contains voters for deciding in each iteration step as well as the stabilization phase for final fine-tuning.

Although BS-TMR and DS-TMR have significantly reduced the required division time, for example, only requiring 9214 and 4300 clock cycles when N = 10, the latency is still significantly higher than for other basic stochastic logic units (adders and multipliers). Moreover, due to the TMR blocks, the hardware overhead is higher than for the conventional divider.

[1] Due to page limitation, the existing dividers are briefly reviewed in this paper. For detailed information, please refer to the corresponding references [11], [12], [15].

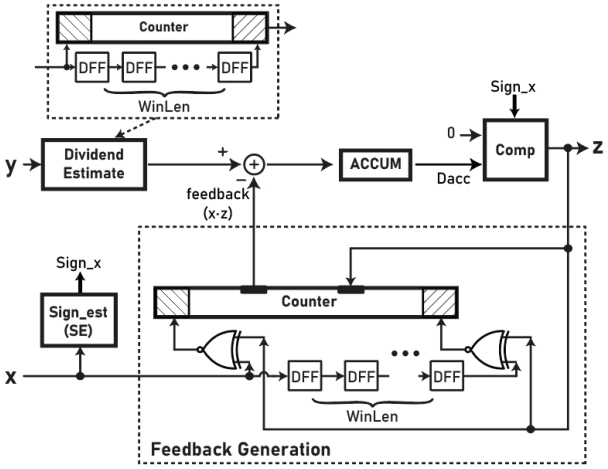


Figure 2. The existing DSM-based divider, DSM-PRE [15].

2) DSM-Based Divider

A fast-converging divider (i.e., the DSM-based divider, referred to as DSM-PRE in this paper), has been proposed to reduce the latency further [15]. As shown in Fig. 2, a negative feedback loop also exists in the DSM-PRE design, but the feedback data origins from the $x{\cdot}z$ estimate within a *WinLen*-bit window. Since the "1" bits in a stochastic sequence are uniformly distributed, checking the number of "1"s in a small segment (i.e., the so-called window) can approximately estimate the real number represented by this sequence. The integer-based accumulator stores the accumulated unmatched error between the estimated y and $x{\cdot}z$; it then adjusts the output bit of z to reduce the error. The DSM-PRE is designed for a bipolar stochastic sequence; thus, a sign estimate block (Sign_est) is applied. This block is implemented by a counter; this circuit counts the number of "1"s in the first or second *WinLen*-bit window to estimate the sign of the sequence x.

The divider converges fast due to its highly efficient scheme of controlling the unmatched errors between y and $x \cdot z$. Compared to the abovementioned dividers that adjust the quotient probability based mostly on the current values of y and $x{\cdot}z$ bits and the pseudorandom output bit of the SNG, the DSM-PRE adjustment introduces fewer errors. However, the Dividend Estimate block still requires *WinLen* clock cycles, and the Sign_est block also incurs in an additional number of cycles. Moreover, the Sign_est block may erroneously estimate the sign of the sequences for small dividend and divisor values that are nearly 0. Therefore, a divider design without such a Sign_est block could potentially achieve a better computational latency and accuracy; this is discussed and evaluated in the following sections of this paper.

D. Challenges and Motivation

The challenges of stochastic divider circuit design focus on improving accuracy and increasing the computation speed while maintaining a low hardware overhead.

- ***Accuracy.*** The existing stochastic dividers are inaccurate when the inputs are near the center of the entire possible range (e.g., [-0.2, 0.2] for a bipolar sequence). This is expected, because the fundamental operation of a divider is based on multiplication, so with a lower computational accuracy for inputs centered in the possible range [14]. This issue is mitigated in DSM-PRE by the continuous calibration of the accumulated error. However, it is still confronted with errors brought by the sign estimate.
- ***Computation clock cycles.*** The other challenge of a stochastic divider design is the required convergence time of the computation, which results in significant latency and power dissipation for the entire system. For example, the conventional, BS-TMR, and DS-TMR dividers require 46341, 9214, and 4300 clock cycles to complete the division respectively for $N = 10$ [11], [12]. The DSM-PRE divider (with a *WinLen* of 2^7) requires fewer clock cycles, but it still needs 1438 cycles (which is larger than $2^N = 1024$) to complete the computation [15]. Major concerns are found on the sign estimate algorithm, which needs one or two times the number of *WinLen* cycles, and the dividend estimate process requiring an additional *WinLen* cycles.

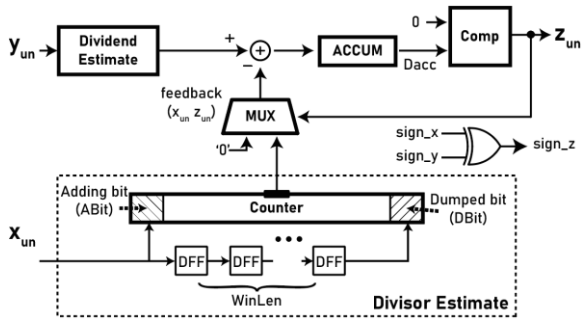


Figure 3. The proposed DSM-U1 stochastic divider with independent sign representation.

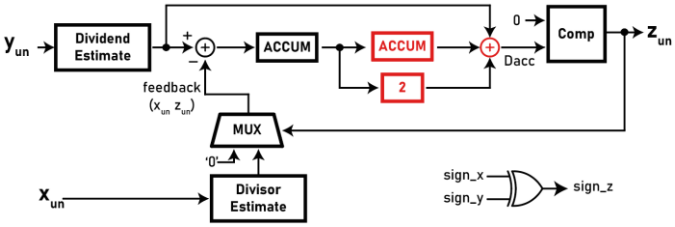


Figure 4. The proposed DSM-U2 stochastic divider (blocks that are different from DSM-U1 shown in Fig. 3 are marked in red; the Divisor Estimate block is the same as that in DSM-U1).

To mitigate the abovementioned issues and improve the performance of stochastic dividers, in this paper, we propose two types of DSM-based dividers: the improved first-order DSM divider (DSM-U1) to address the accuracy issue in a hardware-efficient way and the feed-forward second-order DSM divider (DSM-U2) to address the computation clock cycles issue.

Compared to all existing stochastic dividers, the proposed dividers improve computation accuracy, especially with inputs near the center of the entire range. Additionally, the proposed DSM-U2 reduces the number of clocks to 2^N by considering a trade-off between *WinLen* and accuracy. This design either does not require additional clock cycles compared to addition and multiplication with an acceptable accuracy, or offers higher accuracy with a smaller number of clock cycles than existing stochastic dividers.

The existing DSM-PRE divider uses a block to estimate the sign of the inputs, which decides the feedback calculation. This scheme may face difficulties if the input values are near 0 (for

Algorithm 1 Feedback generation process of DSM-U1

```
Counter = Counter + ABit - DBit;
if z_un = 1
    feedback = Counter;
else
    feedback = 0;
end
Dacc = Dacc+ Input - feedback;
if Dacc >= 0
    z_un = 1;
else
    z_un = 0;
end
```

Algorithm 2 Feedback generation process of DSM-U2

```
Counter = Counter + ABit - DBit;
if z_un = 1
    feedback = Counter;
else
    feedback = 0;
end
ACCUM_stage1 = ACCUM_stage1 + Input – feedback; ACCUM_stage2 = ACCUM_stage2 + ACCUM_stage1;
Dacc = Input + 2×ACCUM_stage1 + ACCUM_stage2;
if Dacc >= 0
    z_un = 1;
else
    z_un = 0;
end
```

the bipolar representation). Due to this limitation, the DSM-PRE has similar performance with input values in this range compared to other stochastic dividers. A high-order DSM can mitigate this issue, while achieving a higher resolution; in a way like the definition of order in DSM analog to digital converters [26], the order in this design is defined as the number of stages of accumulator in the feedback loop. Thus, in this paper, a second-order DSM divider with feed-forward scheme is also proposed and discussed.

III. Proposed DSM-based Stochastic Dividers

In this section, we propose an improved first-order DSM-based divider DSM-U1 and a second-order DSM divider DSM-U2 for accurate and low latency SC systems. The DSM-U1 and DSM-U2 use an additional sign bit and unipolar representation to achieve bipolar computation, which can solve potential errors introduced by sign estimate in the existing DSM-PRE design and reduce the hardware overhead. The DSM-U2 further reduces the size of *WinLen*, which leads to a shorter stochastic sequence length. Both proposed dividers achieve a better division computing accuracy.

A. Proposed Circuit Design of DSM-U1

The proposed DSM-U1 is shown in Fig. 3. Compared to DSM-PRE, an additional bit of sign representation is added; the input values are converted to absolute values, and the operands during the bipolar computation are then converted to the unipolar representation. As shown in Fig. 3, the input value pair X and Y are now represented by x_{un} with *sign_x*, and y_{un} with *sign_y*, respectively; x_{un} and y_{un} denote the unipolar represented stochastic sequence of the absolute values of X and Y, and *sign_x* and *sign_y* denote the signs of X and Y, respectively.

Similarly, the Dividend/Divisor Estimate block estimates the value of the input sequence by using a counter to check the number of "1" bits in a *WinLen*-bit segment. For a low

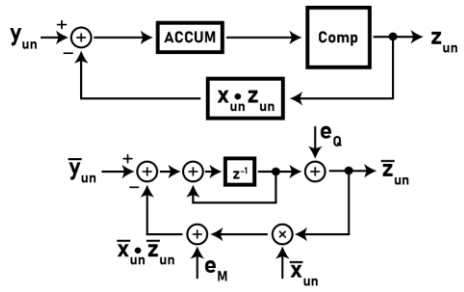


Figure 5. Simplified block diagram and corresponding z-transform error analysis model of the proposed DSM-U1 divider.

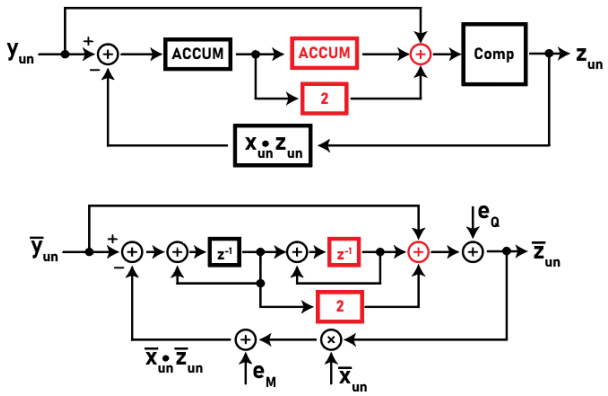


Figure 6. Simplified block diagram and corresponding z-transform error analysis model of the proposed DSM-U2 divider (blocks that are different from DSM-U1 shown in Fig. 5 are marked in red).

complexity design, only the adding bit and the dumped bit of each segment are provided as input to the counter, so that the counter acts as a timing window moving along with the input sequence. Due to the unipolar operation, the multiplication of the stochastic sequence changes from XNOR to AND, thus, the feedback generation is simplified by eliminating the two XNOR gates and adding a multiplexer (MUX) that has two $\log_2$*WinLen*-bit channels and is controlled by the output bitstream of the divider z_{un}. The MUX generates the feedback value by selecting one channel data either from the Divisor Estimate block or "0", thus forming the product of x_{un} and z_{un}. Note that the MUX can also be implemented by an AND array as an alternative solution.

Then, the difference between the estimated y_{un} and $x_{un}{\cdot}z_{un}$ is sent to the integer-based accumulator (ACCUM) and added to the previously accumulated difference (*Dacc*). The bit width of the accumulator can be designed based on either a targeted level of computational accuracy, or the value range of a given application (e.g., checking the possible range of difference accumulation by simulation); in this paper, it is designed with 16 bits as an example for all DSM-based dividers as sufficient to achieve a high accuracy. Since the goal of the divider design is to obtain a convergent feedback loop (i.e., making $y_{un} = x_{un}{\cdot}z_{un}$), the quotient sequence z is adjusted by comparing the accumulated difference with 0. Specifically, if $D_{acc} > 0$, which means $y_{un} > x_{un}{\cdot}z_{un}$, then the comparator generates a bit "1" to increase the value of z and make $x_{un}{\cdot}z_{un}$ closer to y_{un}. Here, the convergence mechanism of the loop can also be considered as pushing the accumulated difference to 0 (i.e., by eliminating these differences). In this case, when $D_{acc} > 0$, the "1" bit

generated by the Comp unit is used (as an enable signal in the MUX unit) to choose a smaller value between the two possible results of the adder, that is added to D_{acc} in the next clock cycle. Specifically, the adder result is either equal to the counter value for y_{un} subtracted by that for x_{un}·z_{un}, or equal to the counter value for y_{un} itself; since the first result is always smaller, it is selected by the enable signal in the MUX unit to be added to D_{acc}. Similarly, a "0" bit is generated by the comparator when D_{acc} < 0 to either decrease the value of z_{un} or choose a larger value between the possible outputs of the adder to be added in the accumulator. Finally, the loop converges based on the adjustment in each clock and the accurate quotient sequence z_{un} is obtained. This process is given in Algorithm 1.

Note that as the same as all existing dividers, the proposed designs also work when the dividend is larger than the divisor. In this case, the quotient sequence is the one reporesenting a real number of 1 or -1, which are the boundaries of the computaitonal range for standard bipolar SC.

B. *Proposed Circuit Design of DSM-U2*

In oversampling ADC design, higher-order DSMs provide a higher signal-to-noise ratio (SNR), leading to a higher resolution [26]. This feature is exploited in the proposed second-order DSM divider with feed-forward paths to attain a higher accuracy with input values near 0. Moreover, due to its higher accuracy, the proposed DSM-U2 can be designed with a smaller *WinLen*. Thus, it makes DSM-U2 suitable to low latency SC applications.

Fig. 4 shows the proposed DSM-U2 stochastic divider. It is implemented by applying another stage of the accumulator (ACCUM). The proposed scheme has one negative feedback loop with two additional feed-forward paths, from the estimated input y_{un} and the output of the first stage of ACCUM, to the adding node prior to the comparator. Unlike DSM-U1, *Dacc* in DSM-U2 is calculated by adding y_{un}, 2×$ACCUM_{stage1}$, and $ACCUM_{stage2}$. This process is given in Algorithm 2; note that 2×$ACCUM_{stage1}$ is implemented by a simple one-bit shift to the left.

C. *Analysis of Expected Advantages*

Like DSM, a higher order design (so with an additional stage of ACCUM in the loop) suppresses the accumulated error and adjusts the output bit with more flexibility, rather than utilizing rigid adjustments of the first order DSM-based dividers. To analyze error suppression, simplified models of the proposed DSM-U1 and DSM-U2 are investigated to obtain the signal transfer (STF) and error transfer (ETF) functions.

As shown in Fig. 5, the DSM-U1 circuit can be simplified with three major blocks, ACCUM, Comp as a quantizer that quantizes the accumulated error, and the multiply operation of x_{un} and z_{un}. Then, based on the simplified circuit, the corresponding *z*-transform model is established to obtain the ETF. From the *z* model, the error originates from two sources, the quantizer and multiplier. The quantization error (e_Q) is the difference between the accumulated error and the output z_{un}. While the multiplication error is from the computation of x_{un} · z_{un} and the estimate of the stochastic sequence segments of x_{un} and z_{un} through the *WinLen*-bit window. The analysis of ETF is required to investigate the relationship between one of the

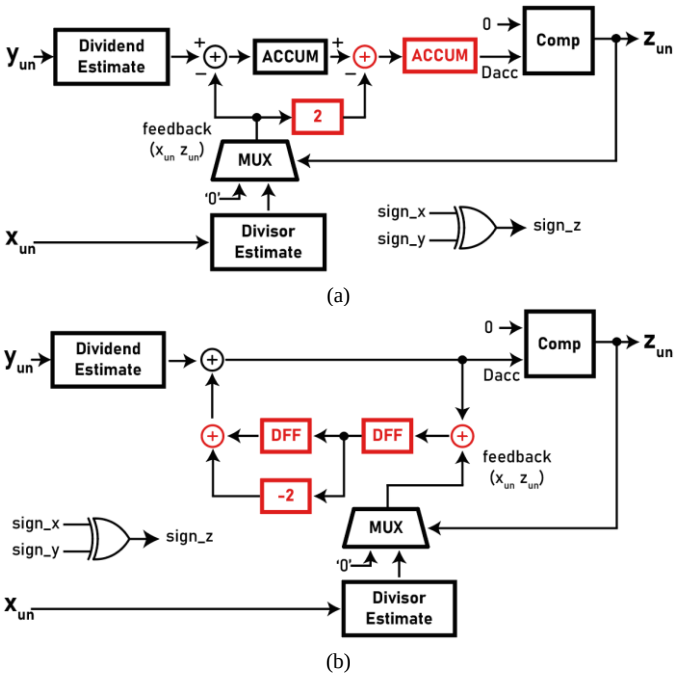


Figure 7. Other schemes for second-order DSM divider implementation (blocks that are different from DSM-U1 shown in Fig. 3 are marked in red): (a) Single loop second-order; (b) Error feedback second-order.

inputs ($I_{\bar{y}}$), and the output of the multiplier ($O_{\overline{xz}}$). Thus, we can obtain the relationship as shown in the following equation:

$$O_{\overline{xz}}(z) = \left(I_{\bar{y}}(z) - O_{\overline{xz}}(z)\right) \cdot \frac{z^{-1}}{1-z^{-1}} + e_Q(z) + e_M(z), \quad (1)$$

where e_Q and e_M are the quantization error and multiplication error, respectively. Then we can obtain the STF and ETF from

$$O_{\overline{xz}}(z) = z^{-1} \cdot I_{\bar{y}} + (1-z^{-1}) \cdot e_Q(z) + (1-z^{-1}) \cdot e_M(z). \quad (2)$$

Finally, from (2), we observe that y_{un} is unaltered but delayed, while e_Q and e_M are differentiated and suppressed. Thus, we obtain the unaltered input with a differentiated quantization error and multiplication error.

Compared to the proposed DSM-U1, DSM-U2 has an additional stage of ACCUM, which suppresses the quantization and multiplication error further. From the simplified circuit block diagram and the corresponding z-transform model (shown in Fig. 6), we obtain the relationship between $O_{\overline{xz}}$ and $I_{\bar{y}}$ as per

$$O_{\overline{xz}}(z) = \left(I_{\bar{y}}(z) - O_{\overline{xz}}(z)\right) \cdot \left(\frac{z^{-1}}{1-z^{-1}}\right)^2 + \left(I_{\bar{y}}(z) - O_{\overline{xz}}(z)\right) \cdot 2 \cdot \frac{z^{-1}}{1-z^{-1}} + I_{\bar{y}}(z) + e_Q(z) + e_M(z). \quad (3)$$

Then the STF and ETF of DSM-U2 are found from the following equation:

$$O_{\overline{xz}}(z) = I_{\bar{y}} + (1-z^{-1})^2 \cdot e_Q(z) + (1-z^{-1})^2 \cdot e_M(z). \quad (4)$$

We obtain the unaltered input from (4), with a squared differentiated quantization error and multiplication error. This implies that the second-order DSM divider can withstand more errors and suppress errors further. Thus, the ETF indicates that the DSM-U2 can be implemented with a smaller *WinLen* (this causes more errors, but they can be suppressed by the enhanced

proposed second-order based process) while maintaining a high computation accuracy.

As per (4), the second-order design results in two poles due to the additional order in the loop; thus, it may be less stable than the first-order design. Such an issue may degrade the divider's performance when the inputs are close to the maximum values (i.e., the value of 1 for a stochastic sequence), and then result in a low computational accuracy, or even make the divider fail. This potentially may occur because the maximum value of the second stage accumulator keeps increasing to an unduly large value and is unable to recover. However, due to the reduced *WinLen* and lower number of clock cycles required, and the mechanism of the accumulator (automatically resetting to 0 when there is an overflow), the proposed DSM-U2 remains stable over the entire input ranges. This stability is also verified by the simulation results presented in Section IV-A (which show a very high computational accuracy for all value ranges).

D. Considerations between Different Topologies

The second-order DSM divider can be implemented in several different ways; for example, as shown in Fig. 7, two additional schemes are designed with the same STF and ETF of (4). The scheme in Fig. 7 (a) does not have a feed-forward path, and all the sum nodes have only two inputs, which leads to slightly less hardware. The scheme in Fig. 7 (b) eliminates the ACCUM blocks, thus, it may relax timing constraint requirements and perform the calculation at a higher speed. Considering that both the STF and ETF are the same for all second-order DSM dividers, and their performances are similar, we choose the proposed feed-forward scheme based on its performance (evaluated by simulations in section V).

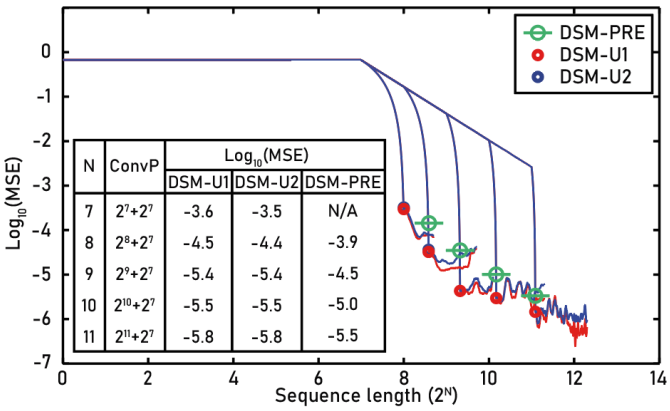
Higher-order DSM dividers are supposed to theoretically provide more error suppression; however, such type of scheme also induces stability issues, input range limitation, and more hardware due to the additional stage of ACCUM. Thus, the advantages caused by the higher order may be overwhelmed by more disadvantages. In this paper, the investigation of DSM dividers beyond second order is not further pursued.

IV. Evaluation

In this section, the performance of the proposed DSM-based stochastic dividers is evaluated and compared with existing designs discussed previously, including conventional [5], BS-TMR [11], DS-TMR [12], and DSM-PRE [15] dividers.

A. Convergence and Accuracy

The proposed dividers generate the quotient sequence starting from the *WinLen*th clock cycle. The convergence point (ConvP) is defined as the finish point of the division computation, i.e., at the clock cycle of 2^N+*WinLen*. Compared with the LFSR-based sequence (that is susceptible to seed choices and pseudorandom number distribution), the Sobol sequence is usually used to evaluate the close-to-theoretical performance, which is also applied in this paper to investigate ConvP. Fig. 8 shows the ConvPs for the proposed dividers at different sequence lengths (i.e., N = 7 to 11), with the same number of *WinLen* = 2^7 as that in the DSM-PRE for comparison purposes. Compared with DSM-PRE, the proposed dividers can achieve a higher accuracy as shown in Fig. 8.



N	ConvP	Log10(MSE)		
		DSM-U1	DSM-U2	DSM-PRE
7	2^7+2^7	-3.6	-3.5	N/A
8	2^8+2^7	-4.5	-4.4	-3.9
9	2^9+2^7	-5.4	-5.4	-4.5
10	$2^{10}+2^7$	-5.5	-5.5	-5.0
11	$2^{11}+2^7$	-5.8	-5.8	-5.5

Figure 8. Convergence points of the proposed DSM dividers with different N and fixed *WinLen* of 2^7 for 10000 random input pairs (the points for DSM-U1 and DSM-U2 overlap).

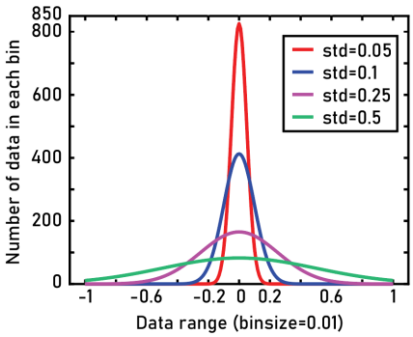


Figure 9. Datasets of random numbers in Gaussian distribution with different standard deviations as used for simulation.

The impact of *WinLen* on computation accuracy is also investigated using four datasets that follow a Gaussian distribution; this evaluates the statistical performance of the proposed dividers in different divisor and dividend ranges. The datasets include random number pairs for the input dividend and divisor that follow Gaussian distribution. They have different standard deviations (μ = 0, σ = 0.05, 0.1, 0.25, 0.5), as shown in Fig. 9. The results for both DSM-PRE and the proposed dividers with N = 10 and different *WinLen* values are plotted in Fig. 10. From the results, we can observe that all dividers perform better for a more uniformly distributed dataset. The trends of computation accuracy with *WinLen* values can be seen for DSM-PRE; with larger *WinLen*, the accuracy is better. DSM-U1 and DSM-U2 follow the same trend with *WinLen* until it reaches 2^6 to 2^7. For larger *WinLen* (*WinLen* > 2^7), both DSM-U1 and DSM-U2 show a slight accuracy deterioration as *WinLen* increases. Moreover, the accuracy curve tendency of DSM-U2 does not have significant variations over the entire *WinLen* range. It indicates that for divisor and dividend with small absolute values, the length of the segment sequence used for the value estimate does not significantly affect the performance of DSM-U2. Due to its further error suppression capability, DSM-U2 can operate with a small *WinLen*. Thus, the proposed DSM-U2 can be fully compatible (*WinLen* = 2^0) with stochastic adders and multipliers, i.e., no additional clock cycles are needed.

Based on the *WinLen* simulation results, we set the *WinLen* values to 2^4 and 2^0 for DSM-U1 and DSM-U2, respectively; this value is 2^7 for DSM-PRE from [15]. The simulation results of

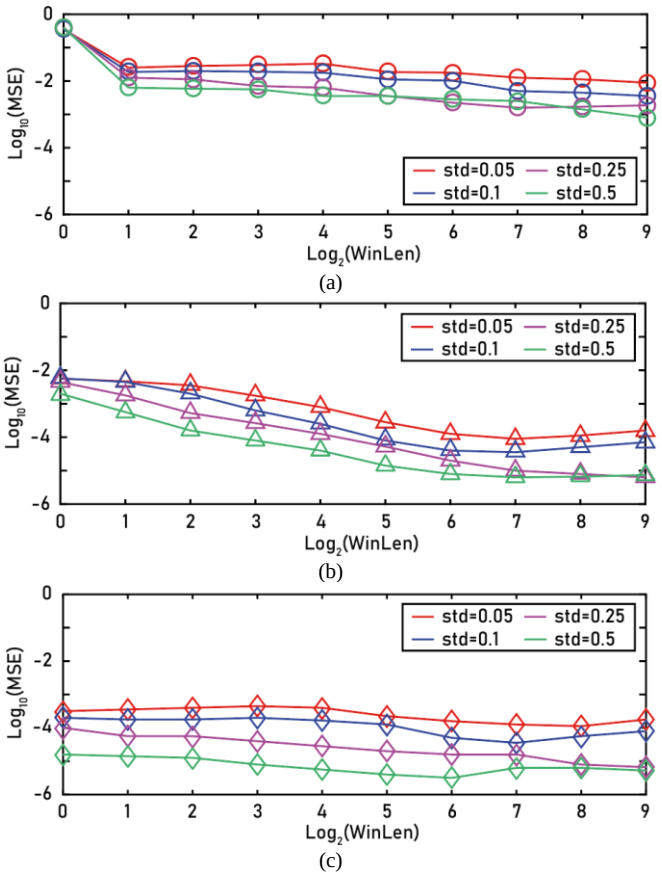


Figure 10. MSE of all DSM-based dividers with N = 10 and different *WinLen* values for 10000 random input pairs: (a) DSM-PRE, (b) the proposed DSM-U1, (c) the proposed DSM-U2.

all DSM-based dividers using such *WinLen* settings with Sobol-based sequences are shown in Table I; it reports the total averaged MSE and that for both dividend and divisor within the range of [μ-σ, μ+σ] for the four Gaussian distributed datasets. Both proposed dividers achieve better accuracy than DSM-PRE and the conventional divider, especially for the dataset with σ = 0.05, in which most dividend and divisor values are around 0. This is reasonable because DSM-PRE is susceptible to sign estimate errors when the Sign_est block works with divisors that have smaller absolute values. The conventional divider suffers from an ineffective error adjustment and in particular, errors introduced by the multipliers within the feedback loop.

A representative example of small absolute value division computation (0.1/0.11) is shown in Fig. 11, which depicts the convergence and accuracy of different dividers with N = 10. In this case, BS-TMR and DS-TMR require more clock cycles in each iterative searching step to achieve an acceptable accuracy.

TABLE I
SUMMARIZED AVERAGED MSE OF DIFFERENT DIVIDERS WITH GAUSSIAN DISTRIBUTED DATA WITH DIFFERENT STD FOR DIVIDEND AND DIVISOR

Divider	**Averaged $\log_{10}$(MSE)**							
	σ = 0.05		σ = 0.1		σ = 0.25		σ = 0.5	
	$\mu \pm \sigma$	Total	$\mu \pm \sigma$	Total	$\mu \pm \sigma$	Total	$\mu \pm \sigma$	Total
Conventional [5]	-1.1	-1.5	-1.6	-1.9	-2.0	-2.3	-2.3	-2.5
DSM-PRE [15]	-1.6	-1.9	-2.0	-2.3	-2.4	-2.8	-2.3	-2.6
Proposed DSM-U1	-2.8	-3.1	-3.3	-3.6	-3.6	-3.9	-4.2	-4.4
Proposed DSM-U2	**-3.2**	**-3.5**	**-3.5**	**-3.7**	**-3.7**	**-4.0**	**-4.5**	**-4.8**

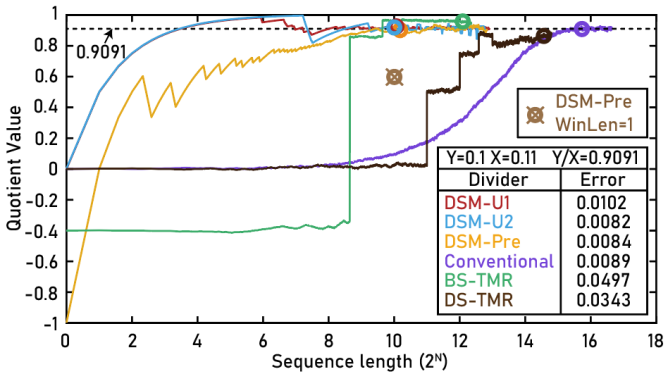


Figure 11. Quotient for 0.1/0.11 obtained by different stochastic dividers with N = 10.

The conventional stochastic divider also works in this case, but due to its ineffective error adjustment, it needs more clock cycles to converge. DSM-PRE achieves similar accuracy compared with the conventional divider and requires 2^N+*WinLen* to 2^N+2·*WinLen* clock cycles (*WinLen* = 2^7 [15], i.e., a 128-bit shift register is needed) to complete the computation. To compare with the proposed DSM-U2 (that will be discussed in the following), the result of DSM-PRE with a reduced *WinLen* that is equal to 2^0 is also plotted in Fig. 11; a significant decrease in accuracy is observed as per the results. Due to the elimination of the Sign_est block and the corresponding induced error in the proposed DSM-U1 design, *WinLen* of DSM-U1 can be reduced to 2^4 to achieve a similar error level compared with DSM-PRE with *WinLen* = 2^7. Moreover, due to the error suppression capability of introducing an additional stage, DSM-U2 achieves a better accuracy even with *WinLen* = 2^0. Overall, the proposed DSM-U2 is more accurate and faster than other dividers. Similar accuracy results are achieved for DSM-U1 with *WinLen* = 2^4, DSM-PRE with *WinLen* = 2^7, and the conventional divider with close to $2^{15.5}$ clock cycles.

As a further evaluation, the accuracy comparison of different stochastic dividers and the required number of clock cycles for inputs with a uniform distribution are considered and

TABLE II
SUMMARIZED AVERAGED MSE OF DIFFERENT DIVIDERS WITH UNIFORMLY DISTRIBUTED DATA FOR DIFFERENT SC SEQUENCE LENGTH

Divider	**Sequence type**	**# of clock cycles**	**$\log_{10}$(MSE) for different N**				
			7	8	9	10	11
Conventional [5]	LFSR	$2^{N+5.5}$	-	-2.9	-3.1	-3.4	-3.8
BS-TMR [11]	LFSR	$2^N + N$•round(2^{N+3}/10)	-	-2.9	-3.1	-3.4	-3.8
DS-TMR [12]	LFSR	$2^N + 2$•round(2^{N+4}/10)	-	-2.9	-3.1	-3.4	-3.8
DSM-PRE [15]	Sobol	$2^N + 2^7$ to $2^N + 2^8$	-3.3	-3.9	-4.5	-5.0	-5.5
	LFSR	$2^N + 2^7$ to $2^N + 2^8$	-2.8	-3.2	-3.5	-3.9	-4.2
Proposed DSM-U1	Sobol	$2^N + 16$	-3.8	-4.2	-4.5	-5.0	-5.5
	LFSR	$2^N + 16$	-2.9	-3.4	-3.6	-4.0	-4.3
Proposed DSM-U2	Sobol	$\mathbf{2^N}$	**-4.5**	**-4.9**	**-5.4**	**-5.6**	**-5.9**
	LFSR	$\mathbf{2^N}$	**-3.1**	**-3.8**	**-4.6**	**-4.7**	**-4.9**

summarized in Table II. As per the considered application that is studied in Section-V, both LFSR and Sobol-based sequences are utilized. Compared to all existing dividers, the proposed DSM-U2 achieves the best computational accuracy, requiring no additional clock cycles; this feature makes this divider completely (timing-wise) compatible with stochastic adders and multipliers. The elimination of the sign estimate in both proposed dividers further reduces potential errors. Thus, compared to the existing DSM-PRE, the proposed dividers achieve a higher accuracy while still having a stable convergence process (i.e., a fixed number of clock cycles for convergence). Therefore, the proposed dividers operate with shorter stochastic sequences, that abide by SC trends and can potentially be implemented in large-scale systems.

B. *Hardware overhead*

The hardware overhead required by the proposed stochastic dividers is evaluated next. To achieve a high accuracy, the proposed DSM-U1 divider is designed with *WinLen* = 2^4 as per the analysis presented in Section IV-A, while *WinLen* = 2^0 is taken for the proposed DSM-U2 divider due to its excellent accuracy. The circuits are implemented using Verilog and synthesized using Synopsys Design Compiler by mapping the design to an ASAP 7 nm technology library [27]. The operational frequency of the designs is set to 200 MHz, and the synthesis constraints are set accordingly for different hardware figures of merit to evaluate the best performance in terms of the corresponding metric. The hardware metrics including the circuit area, latency for completing a division, power dissipation, required number of clock cycles are evaluated.

For comparison purposes, the existing stochastic dividers evaluated in the previous section and a widely used Newton-Raphson-based FP divider (i.e., a binary/non-SC unit) [28] found in the technical literature are also implemented. For the existing DSM-PRE, *WinLen* = 2^7 is employed for achieving a high accuracy as per the previous discussion in Section IV-A and from [15]. A 16-bit accumulator is used in all DSM-based dividers for a fair comparison. Moreover, a parallel scheme to implement the Sobol-based conventional divider is also considered; it provides the same accuracy as the conventional divider, but it requires half the number of clock cycles (with 2x parallelization) [13]. N = 10 (i.e., a sequence length of 1024-bit) is taken as an example in all stochastic divider implementations; both LFSR and Sobol based sequences are utilized for all designs (when applicable). The synthesis results of different designs are reported in Table III.

Consider the comparison among all stochastic dividers first.

- The proposed DSM-U1 divider is superior to all other designs exclusive of the LFSR-based conventional divider in terms of area overhead and power dissipation. This is expected due to the low complexity of its circuit. Specifically, the DSM-U1 divider only uses 16 flip-flops to implement *Winlen* = 2^4 (i.e., significantly less than the existing DSM-PRE with *Winlen* = 2^7). Moreover, it achieves a small latency due to the removal of the block for sign estimate (which is required in the DSM-PRE divider, the fastest circuit among all existing designs), as well as the fast convergence process (i.e., requiring a small number of clock cycles).

TABLE III
SYNTHESIS RESULTS OF DIFFERENT DIVIDERS

Divider[*1]		Area (μm^2)	Latency (ns)	Power (μW)	# clock cycles
Floating-point divider [28]		8248.8	100.0	10.0	50
LFSR-based SC (N=10)	Conventional [5]	**653.6**	10380.3	**1.2**	46341
	BS-TMR [11]	2770.6	1233.7	4.0	9214
	DS-TMR [12]	13900.5	463.4	19.7	4300
	DSM-PRE [15]	1150.7	377.6	1.8	1428
	Proposed DSM-U1	780.9	270.4	1.3	1040
	Proposed DSM-U2	1081.6	**256.0**	1.7	**1024**
Sobol-based SC (N=10)	Conventional [5]	1210.6	7286.1	2.7	23171
	Parallel divider[*2] [13]	1566.3	4194.1	3.5	11586
	DSM-PRE [15]	1150.7	304.4	1.8	1159
	Proposed DSM-U1	**780.9**	270.4	**1.3**	1040
	Proposed DSM-U2	1081.6	**256.0**	1.7	**1024**

[*1]For existing dividers, only available results from the corresponding references are reported; [*2]The parallel divider has been proposed only for Sobol-based sequences [13].

- The proposed DSM-U2 requires a larger area and power dissipation compared to DSM-U1 and the conventional stochastic divider due to its second-order configuration. However, since DSM-U2 reduces the implementation cost for *WinLen*, its area and power are still lower than DSM-PRE and all other dividers. Importantly, *WinLen* = 2^0 also permits that DSM-U2 achieves exactly 2^N clock cycles (i.e., the same as a stochastic adder/multiplier), and thus the smallest latency compared to all existing designs. This feature makes the proposed DSM-U2 extremely attractive for implementing an SC-based system, because it does not introduce any additional clocks to the system (and thus to the entire latency and energy) compared to the other types of SC units.
- The advantages of the proposed dividers apply to both LFSR and Sobol based sequences; moreover, their hardware overhead is independent of the sequence type for a given value of N, because the designs do not use an SNG. Instead, for existing dividers that require an SNG (e.g., the conventional design), the version using Sobol-based sequences tends to require a larger area and power, but at a smaller latency/number of clocks compared to the LFSR version; this occurs because the Sobol-based SNG has a larger circuit size and its generated sequences offer a better computation performance [13].

Next, consider the comparison between stochastic dividers and the FP divider. Table III shows that the FP divider requires a significantly larger area and power dissipation compared to all stochastic dividers exclusive of the DS-TMR design[2] but the smallest latency and number of clock cycles. This is expected because SC is known to have a low-hardware complexity by incurring in an additional computation latency for the sequences. However, note that the latency and clock cycles of SC designs given in Table III are for an example of N = 10, while these results are approximately proportional to the sequence length 2^N. This means that some stochastic dividers can potentially have a smaller latency than the FP design (due to their better delay per cycle), when their accuracy is acceptable. For example, when N = 7, the proposed DSM-U1

[2] The DS-TMR stochastic divider of [12] has been proposed as a trade-off solution between the latency/clocks and area/power for an SC-based system; so, it is less meaningful to compare its overhead with a floating point divider.

(DSM-U2) divider that also offers a very low MSE of $10^{-2.9}$ or $10^{-3.8}$ ($10^{-3.1}$ or $10^{-4.5}$) only requires a latency of approximately 36.0 ns (32.0 ns) to complete a division, i.e., they are superior to an FP divider also in latency in addition to the area and power.

C. Discussion on Alternative Implementations

The proposed DSM-based stochastic divider designs can also be implemented using a bipolar representation (i.e., without the sign bit). This is applicable when we can manipulate the representation of the inputs and place the negative sign of an input pair (if it exists) in the dividend, such as for the ESL sequences. In this alternative implementation, the DSM-U1 and DSM-U2 designs do not require the XOR gate to calculate the sign bit of the quotient sequence (as shown in Figs. 3 and 4), and the input "0" of MUX needs to be changed to *~Counter*; the remaining parts of the circuits are kept unaltered.

In the DSM-based divider design, the computation is based on making x•z to track y, which focuses on the probability of "1" bits in a segment of the input sequences (instead of the corresponding real numbers and the way that they are represented); hence, such transformation from signed unipolar to bipolar has no impact on the computational accuracy of the proposed dividers. Moreover, this alternative implementation has nearly the same hardware overhead compared to the designs in Figs. 3 and 4, because only an XOR gate is replaced by at most $\log_2 WinLen$ (the width of the counter) inverters.

V. Application: SC-based Neural Networks

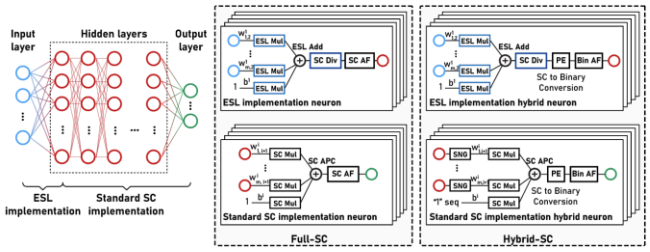
A. Network Implementation

SC has been extensively investigated to achieve power-efficient implementation for different types of neural networks, such as multi-layer perceptron (MLP) [11], convolutional neural networks (CNN) [29], deep belief networks (DBN) [30], and recurrent neural networks (RNN) [31]. In this section, MLP is taken as an example to study the use of the proposed stochastic dividers in an SC-based neural network as an application and demonstrate their effectiveness.

MLPs are one of the most widely used neural networks for performing classification tasks, because they have a very high flexibility for learning the mapping between inputs and outputs and are capable of processing nearly all types of datasets [32]. Fig. 12 (a) shows the structure of an MLP; it consists of an input layer, one or multiple hidden layers, and an output layer, among which each two neighbor layers are fully connected. In the inference process of an MLP, the valid features of a sample being classified are fed into the network (i.e., as the neurons in the input layer shown in Fig. 12 (a)). Then, neuron computations start by performing (5). Specifically, to obtain the n^{th} neuron in the $i+1^{\text{th}}$ layer, each of m neurons in the i^{th} layer is initially multiplied by a weight w; then, the multiply-accumulation result for all m neurons is combined with a bias value b of the i^{th} layer and activated by a function Φ.

$$Neuron_{n,i+1} = \Phi\left(\sum_{j=1}^{m} w_{j,n}^{i} \cdot Neuron_{j,i} + b^{i}\right) \qquad (5)$$

In a typical SC-based neural network implementation, all arithmetic computations of the network are processed using stochastic sequences (i.e., a full-SC design). Fig. 12 (b) shows the hardware diagram of a full-SC MLP implementation for performing high performance inference (as recently proposed in [11] and further improved in [12], [15] by using more efficient dividers). Since the inputs of an MLP and the multiply-accumulation results of neurons can exceed the traditional SC computation range of [-1, +1], stochastic multipliers and adders of the ESL version are utilized to extend the value range and pursue an accurate computation. Moreover, ESL units are only employed for the mapping between the input layer and the first hidden layer for a trade-off between accuracy and hardware overhead [11]. Note that each pair of ESL sequences requires two SNGs; this is also considered in the hardware evaluation conducted in this section. The SC-based activation function (e.g., clamped ReLU or tanh) is typically implemented using an FSM with a single input sequence. Therefore, a stochastic divider is required to convert each multiply-accumulation result (represented using two sequences in ESLs) to a single stochastic sequence as input to the FSM. Since the divider generates 1 or -1 when the dividend is larger than the divisor, it can bound the computational range for ESLs to the one for the standard SC. For neurons in the other layers, standard SC multipliers and APCs are employed for performing multiply-accumulation computation.



(a) (b) (c)

Figure 12. SC implementation of an MLP: (a) the MLP network; (b) the full-SC implementation; (c) the hybrid-SC implementation.

In a full-SC implementation, LFSR-based sequences are always utilized, because the other types of sequences (e.g., Sobol) are not capable of providing accurate computation for the FSM-based activation functions [13]. However, the use of the LFSR poses a challenge in terms of clock cycles for the implementation, because sequences of large length are often required for the entire network to achieve high accuracy. Moreover, an accuracy loss may occur or increase due to the accumulated correlation issue among stochastic sequences after they perform a large number of continuous computations; this also limits the use of a full-SC design for implementing large-size networks.

An alternative solution to implement neural networks using SC circuits is to utilize a hybrid configuration as shown in Fig. 12 (c). In this case, the multiply-accumulation of neurons is computed using stochastic sequences. Then, the results are converted to conventional binary numbers for the activation computation using for example floating-point hardware. Once computation for one layer is completed, the activation results are converted back to stochastic sequences for conducting the subsequent layer computation. Therefore, in a hybrid-SC design, SNGs and PEs are required for data conversion between adjacent layers; even though they can be shared by different layers without increasing the circuit area compared to a full-SC implementation, additional latency and an increased number of clock cycles are incurred for data conversion.

However, data conversion during network computation mitigates the issue of error accumulation/accuracy loss caused by using stochastic sequences for continuous computation in a full-SC design (especially when the network has a large size). Moreover, since the activation functions are implemented using conventional non-SC circuits, sequences that offer a higher computation accuracy such as Sobol-based, can be utilized for the multiply-accumulation computations. These two features permit a hybrid-SC design to use shorter sequence lengths than in a full-SC design for achieving a satisfactory accuracy; this can offset the impact of data conversion on the computation latency/clock cycles.

B. Evaluation

The MLPs for two widely-used datasets, including MNIST [33] and SVHN [34], are implemented using both full-SC design and hybrid-SC design to evaluate the advantages of the proposed dividers. Specifically, the full-SC implementation utilizes LFSR-based stochastic sequences with N = 10 (i.e., length of 1024 bits) and the hybrid-SC implementation utilizes Sobol-based sequences with N = 7 (i.e., length of 128 bits); these types and lengths of sequences are set for achieving high classification accuracy with low hardware overhead. In all SC MLP implementations, ESLs are employed for the first hidden layer (the other layers); the proposed dividers and the DSM-PRE of [15] (that is the most efficient design among all existing dividers as corroborated in Table III) are used for converting ESL sequences to single stochastic sequences.

A conventional MLP implementation using the widely used 32-bit single-precision floating-point data and 16-bit fixed-point data is also considered and compared. Table IV describes the network models being implemented (trained and obtained using floating-point data in Matlab); the network structure (i.e., the number of layers and neurons per layer, and the activation functions) for each dataset is determined during training for achieving a high classification accuracy. Table V reports the classification accuracy and synthesized hardware results of different MLP implementations with an operational frequency of 200 MHz (obtained by using the same synthesis method and constraints presented in Section IV-B) for different datasets. In addition to the hardware metrics of area, power, latency and number of clock cycles for completing an inference, a combined metric defined as the product of area, latency, power and clocks (PALPC) is also considered for a comprehensive evaluation.

TABLE IV
MLP MODELS OF DIFFERENT ML DATASETS

Dataset	# layers	# neurons per layer	Activation function		Classification accuracy
			Hidden layer	Output Layer	
MNIST	4	784, 256, 128, 10	Clamped Relu	Tanh	98.3%
SVHN	5	704, 512, 512, 512, 10	Clamped Relu	Tanh	85.5%

Table V shows the advantages of employing SC for MLP implementation, as well as the advantages of the proposed stochastic dividers in such application. The results are summarized as follows. Consider the comparison between SC and conventional floating-point implementations. Even though a small accuracy loss is incurred[3], the use of SC for MLP implementation provides a reduction in all hardware metrics evaluated in this paper, exclusive of the number of clock cycles[4]. This is expected because SC hardware has a lower complexity, but computation using sequences requires a considerable number of clock cycles and reduces data resolution. As a comprehensive evaluation for the hardware overhead, SC MLPs achieve a significant PALPC reduction compared to the floating-point version, that is 28.8% to 63.8% (61.7% to 81.8%) for the MNIST (SVHN) dataset.

Next, consider fixed-point implementations (which also have a considerable lower complexity than floating-point units). Their classification accuracy is slightly lower than the floating-point MLPs due to the lower data precision and similar to the SC version. In terms of hardware metrics, the SC MLPs are shown to have larger PALPC values[5] due to the larger number of clock cycles, but they offer a significantly better accuracy. Moreover, a significant reduction in power is achieved for the SC MLPs when the network size is large. These comparison results show that the SC implementation is better suitable for power-constraint applications that require high accuracy.

Consider a comparison between a full-SC design using LFSR-based sequences and a hybrid-SC design using Sobol-

TABLE V
CLASSIFICATION ACCURACY AND SYNTHESIZED HARDWARE RESULTS OF DIFFERENT MLP IMPLEMENTATIONS FOR DIFFERENT DATASETS

Implementation Scheme			Classification accuracy	Area		Latency		Power		# Clock Cycles	PALPC (%)
				mm^2	%	ns	%	mW	%		
MNIST	Floating-point MLP		98.3%	21.7	100	1475.0	100	1011.4	100	295	100
	Fixed-point MLP		93.3%	16.6	76.5	**350.0**	**23.7**	700.9	69.3	**70**	**3.0**
	LFSR-based SC MLP with different dividers (N = 10)	DSM-PRE [15]	97.8%	15.8	72.8	1452.1	98.4	207.6	20.5	1428	71.2
		Proposed DSM-U1	97.8%	**15.7**	**72.4**	1104.3	74.9	**207.4**	**20.5**	1040	39.2
		Proposed DSM-U2	**98.0%**	15.8	72.8	1079.9	73.2	207.5	20.5	1024	38.0
	Sobol-based SC MLP with different dividers (N = 7)	DSM-PRE [15]	97.5%	21.2	97.7	839.7	56.3	679.9	67.2	512	64.9
		Proposed DSM-U1	97.5%	21.1	97.2	654.4	44.4	679.7	67.2	400	39.3
		Proposed DSM-U2	97.6%	21.2	97.7	624.4	42.3	679.8	67.2	384	36.2
SVHN	Floating-point MLP		85.5%	76.0	100	2050.0	100	3536.1	100	410	100
	Fixed-point MLP		82.8%	69.0	90.8	**470.0**	**22.9**	2913.1	82.4	**94**	**3.9**
	LFSR-based SC MLP with different dividers (N = 10)	DSM-PRE [15]	85.0%	54.5	71.7	1605.0	78.3	693.0	19.6	1428	38.3
		Proposed DSM-U1	85.0%	**54.3**	**71.4**	1281.6	62.5	**692.8**	**19.6**	1040	22.2
		Proposed DSM-U2	**85.1%**	54.5	71.7	1258.2	61.4	692.9	19.6	1024	21.5
	Sobol-based SC MLP with different dividers (N = 7)	DSM-PRE [15]	84.5%	59.4	78.2	1339.5	65.3	1258.7	35.6	640	28.4
		Proposed DSM-U1	84.5%	59.2	77.9	1109.1	54.1	1258.5	35.6	528	19.3
		Proposed DSM-U2	84.7%	59.4	78.2	1071.6	52.3	1258.6	35.6	512	18.2

based sequences. For each MLP, the hybrid-SC version incurs in an additional area overhead due to the floating-point activation function circuits; it also dissipates significantly more power due to the conversion between SC and floating-point data for each layer. The number of clock cycles of hybrid-SC MLPs is smaller than for the full-SC implementations, even though an additional data conversion is required; this occurs because when providing a similar classification accuracy, Sobol-based sequences with a significantly shorter length can be utilized in the hybrid-SC MLPs (this can compensate the impact of data conversion, as discussed previously). Therefore, the use of Sobol-based sequences leads to an improvement in the entire computational latency for performing an inference process.

Consider a comparison between the proposed stochastic dividers and the existing DSM-PRE divider (which is the most efficient design of all existing stochastic dividers as evaluated in Section IV). The evaluation results show that the SC MLPs with these dividers require nearly the same area and power, even though the difference in these metrics for a single divider is not small (as per Table III). This occurs because the dividers are only used for the first hidden layer of an MLP, i.e., its impact on area/power is very low. However, since the divider is on the critical computational path of the network computation, it has a significant impact on the entire latency and number of clock cycles for performing inference, and thus on the comprehensive performance in terms of PALPC. Therefore, the SC MLPs with the proposed DSM-U2 achieve the best PALPC; followed by the MLPs with the proposed DSM-U1. Specifically, compared to the SC MLPs with the existing DSM-PRE divider, the SC MLPs using the proposed DSM-U2 divider offer a PALPC reduction of 46.6% and 44.2% (43.9% and 35.9%) for the MNIST (SVHN) dataset when using LFSR-based and Sobol-based sequences, respectively; this reduction for the DSM-U1 version is 44.9% and 39.4% (42.0% and 32.0%) for the MNIST (SVHN) dataset and when using LFSR-based and Sobol-based sequences, respectively. Moreover, the proposed DSM-U2 divider that permits a very accurate computation, also helps improving the classification accuracy of the SC MLPs.

Overall, Table V shows that SC is more attractive for implementing neural networks in hardware/power constrained platforms compared to conventional floating-point circuits in terms of complexity and to fixed-point circuits in terms of power dissipation. Moreover, the full-SC design is more efficient in terms of area and power, while the hybrid-SC design is more efficient in terms of latency and number of clock cycles. The proposed dividers further improve the performance of an SC implementation in terms of both computational accuracy and hardware overhead (with a significant reduction in latency and PALPC). Therefore, they are very promising for SC applications; for systems with a tight requirement on latency, the proposed dividers may even permit the use of SC implementation that would otherwise be not applicable.

[3]A potential solution to address the issue of accuracy loss is to use SC units to train the MLP models. This investigation, interesting in its own, goes beyond the scope of this paper that focuses on the stochastic dividers and only takes SC MLPs as an application; therefore, it is left for future work.

[4]Since the operation of an SC unit is performed bit by bit (and it is independent to each other), a later unit does not wait to receive the complete sequence from its previous unit; instead, it starts to compute after receiving the first bit of the sequence. Therefore, the number of clock cycles required to complete the network computation depends on the divider, because the divider requires the largest number of clock cycles compared to the other arithmetic units and it is in the critical computational path. Moreover, when data conversion between SC and non-SC is performed during computation (such as in a hybrid-SC MLP design), it introduces an additional number of clock cycles to the path.

[5]There are some potential improvement schemes to design SC MLPs with smaller PALPC values. For example, sequences with a shorter length can be utilized when the accuracy is acceptable. Moreover, in a hybrid-SC scheme, the arithmetic units (both SC and non-SC) can be implemented only for the layer with the largest number of neurons and then reused for other layers; this is feasible because data conversion is performed between different layers and computations using stochastic sequences are independent in each layer. Moreover, the use of correlated sequences can also be investigated to reduce the hardware overhead of SNGs. These investigations are also not the focus of this paper, so they are left for future work.

VI. Conclusion

In this paper, two DSM-based stochastic dividers have been proposed; operationally, these dividers are more compatible to other SC units and achieve a higher division accuracy compared with existing stochastic dividers. By employing an additional sign bit and a sequence that eliminates the sign estimation process of existing DSM-based dividers, the proposed DSM-U1 significantly reduces the required additional clock cycles and improves the accuracy. In addition, a second-order DSM based divider, DSM-U2, has been proposed for implementing a fully compatible SC system. The proposed DSM-U2 realizes division with the same number of clock cycles as other SC units require (e.g., adders, multipliers). In addition, DSM-U2 also achieves the highest accuracy. Moreover, the mechanism of error suppression in DSM based dividers is investigated; this provides a theoretical analysis for the performance of DSM-U2 to achieve a high accuracy in a shortened stochastic sequence. SC-based neural networks (MLPs specifically) have been implemented as a case study to evaluate the advantages of the proposed dividers. Compared to existing stochastic dividers, the use of the proposed dividers offers a significant hardware overhead reduction for the entire network implementation and also helps improving the classification accuracy.

References

[1] B. Moons, and M. Verhelst, "Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 4, pp. 475-486, Dec. 2014.

[2] Y, Liu, S. T. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2809-2924, Aug. 2020.

[3] Y. Y. Lee and Z. A. Halim, "Stochastic computing in conventional neural network implementation: a review," *PeerJ Computer Science*, vol. 6, pp. e309, Nov. 2020.

[4] B. R. Gaines, Stochastic Computing Systems. Boston, MA, USA: Springer, 1969, pp. 37–172.

[5] B. D. Brown and H. C. Card, "Stochastic neural computation I: Computational elements," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[6] N. Onizawa, D. Katagiri, K. Matsumiya, W. J. Gross, and T. Hanyu, "An accuracy/energy-flexible configurable Gabor-flter chip based on stochastic computation with dynamic voltage-frequency-length scaling," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 444-453, Jun. 2018.

[7] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, and T. Inoue, "Compact and accurate digital filters based on stochastic computing," *IEEE Transactions on Emerging Topics Computing*, vol. 7, no. 1, pp. 31–43, Jan. 2019.

[8] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 449-462, Mar. 2014.

[9] B. Yuan and K. K. Parhi, "Belief propagation decoding of polar codes using stochastic computing," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 157–160.

[10] G. Sarkis, S. Hemati, S. Mannor, and W. J. Gross, "Stochastic decoding of LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 939–950, Mar. 2013.

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

[12] S. Liu, X. Tang, F. Niknia, P. Reviriego, W. Liu, A. Louri, and F. Lombardi, "Stochastic dividers for low latency neural networks," *IEEE Trans. Circuits Systems I: Regular Papers*, vol. 68, no. 10, pp. 4102–4115, Oct. 2021.

[13] S. T. Liu and J. Han, "Toward energy-efficient stochastic circuits using parallel Sobol sequences," *IEEE Transactions on Very Large Scale Integration (VLSI) System*, vol. 26, no. 7, pp. 1226–1339, Mar. 2018.

[14] T. J. Baker and J. P. Hayes, "The hypergeometric distribution as a more accurate model for stochastic computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2020, pp. 592–597.

[15] X. Tang, S. Liu, F. Niknia, P. Reviriego, Z. Wang, A. Louri, and F. Lombardi, "A Delta sigma modulator-based stochastic divider," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 8, pp. 3272-3283, Apr. 2022.

[16] S. T. Liu, H. Jiang, L. Liu, and J. Han, "Gradient descent using stochastic circuits for efficient training of learning machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2530-2541, Nov. 2018.

[17] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *IEEE 31st International Conference on Computer Design (ICCD)*, Oct. 2013, pp. 39-46.

[18] S. Wang, G. Xie, J. Han, and Y. Zhang, "Highly accurate division and square root circuits by exploiting signal correlation in stochastic computing," *International Journal of Circuit Theory and Applications*, vol. 50, no. 4, pp. 1375-1385, Apr. 2022.

[19] S. I. Chu, "New divider design for stochastic computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 147–151, Jan. 2020.

[20] T. H. Chen and J. P. Hayes, "Design of division circuits for stochastic computing," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2016, pp. 116-121.

[21] D. Wu and J. S. Miguel, "In-stream stochastic division and square root via correlation," in *Proceedings of the 56th Annual Design Automation Conference*, Jun. 2019, pp. 1-6.

[22] V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselló, "A new stochastic computing methodology for efficient neural network implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 551–564, Mar. 2016.

[23] D. Jenson and M. Riedel, "A deterministic approach to stochastic computation," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2016, pp. 1–8.

[24] M. H. Najafi, D. Jenson, D. J. Lilja, and M. D. Riedel, "Performing stochastic computation deterministically," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2925–2938, Dec. 2019.

[25] P. S. Ting and J. P. Hayes, "Stochastic logic realization of matrix operations," in *17th Euromicro Conference on Digital System Design*, Aug. 2014, pp. 356-364.

[26] P. Shanthi, R. Schreier, and G. Temes, Understanding delta-sigma data converters. Hoboken, NJ, USA: Wiley, 2017.

[27] ASAP (2021): Arizona State Predictive PDK. [Online]. Available: http://asap.asu.edu/asap/.

[28] Y. Li, Computer principles and design in Verilog HDL: Floating-point algorithms and FPU design in Verilog HDL. Hoboken, NJ, USA: Wiley, 2015.

[29] Z. Li, A. Ren, J. Li, Q. Qiu, Y. Wang, and B. Yuan, "Dscnn: hardware-oriented optimization for stochastic computing based deep convolutional neural networks," in *IEEE 34th International Conference on Computer Design (ICCD)*, Oct. 2016, pp. 678–681.

[30] Y. Liu, Y. Wang, F. Lombardi, and J. Han, "An energy-efficient online-learning stochastic computational deep belief network," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 454–465, Jul. 2018.

[31] A. Zhakatayev, S. Lee, H. Sim, and J. Lee, "Sign-magnitude SC: Getting 10X accuracy for free in stochastic computing for deep neural networks," in *55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, Jun. 2018, pp. 1–6.

[32] J. Brownlee, When to Use MLP, CNN, and RNN Neural Networks. Montpelier, VT, Australia: Machine Learning Mastery, 2018. [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998

[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proceedings of NIPS Workshop Deep Learning*, 2011, pp. 1-9.