Accurate and Energy-Efficient Stochastic Computing with Van Der Corput Sequences

Mehran Shoushtari Moghadam
School of Computing and Informatics, University of Louisiana at Lafayette
Lafayette, Louisiana, USA
m.moghadam@louisiana.edu

Sercan Aygun
School of Computing and Informatics, University of Louisiana at Lafayette
Lafayette, Louisiana, USA
sercan.aygun@louisiana.edu

Mohsen Riahi Alam
School of Computing and Informatics, University of Louisiana at Lafayette
Lafayette, Louisiana, USA
mohsen.riahi-alam@louisiana.edu

Jonas I Schmidt
School of Computing and Informatics, University of Louisiana at Lafayette
Lafayette, Louisiana, USA
jonas.schmidt1@louisiana.edu

M. Hassan Najafi
School of Computing and Informatics, University of Louisiana at Lafayette
Lafayette, Louisiana, USA
najafi@louisiana.edu

Nima TaheriNejad
Institute of Computer Engineering, Heidelberg University
Heidelberg, Germany
nima.taherinejad@ziti.uniheidelberg.de

ABSTRACT

In stochastic computing (SC), data is represented using random bit-streams. The efficiency and accuracy of SC systems rely heavily on the stochastic number generator (SNG), which converts data from binary to stochastic bit-streams. While previous research has shown the benefits of using low-discrepancy (LD) sequences like Sobol and Halton in the SNG, the potential of other well-known random sequences remains unexplored. This study investigates new random sequences for potential use in SC. We find that Van Der Corput (VDC) sequences hold promise as a random number generator for accurate and energy-efficient SC, exhibiting intriguing correlation properties. Our evaluation of VDC-based bit-streams includes basic SC operations (multiplication and addition) and image processing tasks like image scaling. Our experimental results demonstrate high accuracy, reduced hardware cost, and lower energy consumption compared to state-of-the-art methods.

CCS CONCEPTS

• **Hardware** → **Emerging technologies**; *Logic circuits*; *Very large scale integration design*; • **Computing methodologies** → *Computer vision.*

KEYWORDS

Correlation, image scaling, low-discrepancy sequences, random number generator, stochastic computing

ACM Reference Format:
Mehran Shoushtari Moghadam, Sercan Aygun, Mohsen Riahi Alam, Jonas I Schmidt, M. Hassan Najafi, and Nima TaheriNejad. 2023. Accurate and Energy-Efficient Stochastic Computing with Van Der Corput Sequences. In *18th ACM International Symposium on Nanoscale Architectures (NANOARCH '23), December 18–20, 2023, Dresden, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3611315.3633265

NANOARCH '23, December 18–20, 2023, Dresden, Germany

ACM ISBN 979-8-4007-0325-6/23/12…$15.00
https://doi.org/10.1145/3611315.3633265

1 INTRODUCTION

Stochastic computing (SC) is a re-emerging computing paradigm offering low-cost hardware designs and high tolerance to noise. In contrast to traditional binary computing, which operates on positional binary radix numbers, SC designs process uniform bit-streams of "0"s and "1"s with no significant digits. While the paradigm was known for approximate computations for years, recent works showed deterministic and completely accurate computation using SC circuits [19, 22]. Encoding data from traditional binary to stochastic bit-streams is an important step in any SC system. The data are encoded by the probability of observing a "1" in the bit-stream. For example, a bit-stream with 25% "1" represents the data value of 0.25. The accuracy of the computations and the energy efficiency of the SC designs highly depend on this encoding step, particularly on the distribution of "1"s and "0"s in the bit-streams.

A stochastic number generator (SNG), which encodes a data value in binary format to a stochastic bit-stream, consists of a random number generator (RNG) and a binary comparator. At each cycle, the output of comparing the input data with the random number from the RNG unit produces one bit of the bit-stream. The distribution of the bits in the encoded bit-streams is directly affected and controlled by the RNG component of the SNG. While traditionally pseudo-random sequences; generated by linear-feedback shift registers (LFSRs), were used for the RNG unit, the state-of-the-art (SOTA) studies demonstrate the importance of using quasi-random sequences, such as `Sobol` [16, 19] and `Halton` [2, 15] sequences, for high-quality generation of stochastic bit-streams. These sequences remove an important source of error in SC operations, namely the random fluctuation error [23] in generating bit-streams and produce *Low-Discrepancy* (LD) bit-streams. LD bit-streams quickly converge to the target value, reducing the length of bit-streams and, consequently, the latency of stochastic computations. This latency reduction directly translates to savings in energy consumption (i.e., power × latency), a critical metric in the hardware efficiency of the SC systems. A challenge with the SOTA SNGs using sequences such as `Sobol` and `Halton` is their relatively high hardware cost. This high hardware cost limits the maximum energy savings achievable using these sequences. This study extends the SOTA random sequences for the high-quality encoding of data in SC. We analyze some well-known random sequences in the literature for possible improvement in the performance and hardware

efficiency of the SNG units. We explore Weyl (W), R2 (R), Kasami (K), Latin Hypercube (L), Gold Code (G), Hadamard (HD), Faure (F), Hammersly (HM), Zadoff–Chu (Z), Niederreiter (N), and Poisson Disk (P) sequences in the context of SC. We also evaluate and study the Van Der Corput (VDC) [24] sequences as a promising alternative to prior LD sequences.

2 BACKGROUND

2.1 Random Sequences

Random sequences are widely used in various research domains. All of the mentioned sequences except the complex-valued Zhadoff–Chu sequence, have LD properties. *Discrepancy* means how much the sequence points deviate from uniformity [17]. The *recurrence* property (i.e., the constructibility of further-indexed sequences from the previous-indexed ones) in LD sequences is beneficial for cross-correlation. This is particularly advantageous for SC systems that require uncorrelated bit-streams [1].

The Weyl sequence belongs to the class of additive recurrence sequences, characterized by their generation through the iteration of multiples of an irrational number modulo 1. Specifically, by considering $\alpha \in \mathbb{R}$ as an irrational number and $x_i \in \{0, \alpha, 2\alpha, ..., k\alpha\}$, the sequence $x_i - \lfloor x_i \rfloor$ (x_i modulo 1) produces an equidistributed sequence within the interval $(0, 1)$. Another example of an additive recurrence sequence is the R sequence, which is based on the *Plastic Constant* (the unique real solution of the cubic equation) [18]. The Latin Hypercube sequences involve partitioning the sampling space into equally sized intervals and randomly selecting a point within each interval [14].

The VDC sequence serves as the foundation for many LD sequences. It is constructed by reversing the digits of the number in the corresponding base, representing each integer value as a fraction within the $[0, 1)$ interval. For instance, the decimal value 11 in base-3 is represented by $(102)_3$. The corresponding value for the base-3 VDC is $2 \times 3^{-1} + 0 \times 3^{-2} + 1 \times 3^{-3} = \frac{19}{27}$.

For the rest of this work, we will use LD-type and LFSR-based random sequences, and leave the orthogonal and complex-valued sequences for our future work on other emerging technologies, such as hyperdimensional computing, that require high orthogonality [4].

2.2 Stochastic Computing (SC)

SC has gained attention due to its robustness to noise, high parallelism, and power efficiency. Complex arithmetic operations are realized with simple logic gates, achieving significant savings in implementation costs for a range of applications, from image processing [12] to machine learning [13].

An essential step in SC systems is data conversion. Real numbers must be converted to bit-streams, where each bit position has equal significance, distinguishing it from conventional binary representation. SC supports data in the unit interval, i.e., [0, 1]. This coding format is known as unipolar encoding (UPE). In this encoding, the probability of observing a "1" in the bit-stream X or $P(X = 1)$ equals the input value. The common method for generating a bit-stream with a length of N involves generating N random numbers $(R_1...R_N)$ and comparing them with the input value in N cycles. A logic-1 is produced at the output if the input value is greater than the random number; A logic-0 is produced otherwise. The occurrence of logic-1s in the produced bit-stream depends on the sequence of random numbers.

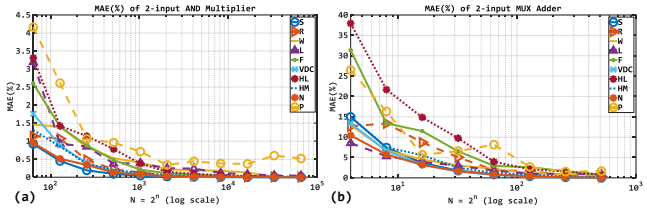

Figure 1: MAE (%) of SC operation on two 8-bit precision input (a) SC Multiplication and (b) SC Scaled Addition.

SC operations often consist of simple bit-wise logic operations. Multiplication of bit-streams in UPE is achieved by bit-wise AND operation [1]. For accurate multiplication, the input bit-streams must be *uncorrelated* with each other. Performing bit-wise AND on *correlated* bit-streams, i.e., bit-streams with a maximum overlap in the position of 1s, gives the minimum of the input bit-stream. Scaled addition is realized in SC by using a multiplexer (MUX) unit [7]. For scaled subtraction, a MUX with one inverter is utilized [12].

3 DESIGN SPACE EXPLORATION

As one of the main contributions of this study, we comprehensively and comparatively examine the use of the random sequences discussed in Section 2 for SC. We first analyze these sequences for basic SC operations before extending the evaluations to more complex case studies. The numbers provided by these sequences are used as the required random numbers $(R_1...R_N)$ during bit-stream generation. These sequences can be pre-stored and read from memory or dynamically generated by using custom digital circuits. In summary, if the to-be-encoded input is greater than the random number a "1" is produced for the bit-stream. Otherwise, a "0" is generated. Prior work in SC has used Sobol [16], Halton [2], and VDC [11] sequences for LD bit-stream generation. Sobol-based LD bit-streams, in particular, has shown promising performance and fast-converging property compared to other sequences. Overall, Sobol-based SC designs are more energy-efficient than the Halton-based designs [16]. VDC is a generalized version of the Halton sequence. In this work, we reveal a new correlation property of the VDC sequences and propose a lightweight hardware design for VDC number generation.

3.1 Benchmark-I: SC Multiplication

We first evaluate the performance of the selected sequences for 2-input multiplication. Two input values ($x1$ and $x2$) are converted to bit-stream representation ($X1$ and $X2$) by using the random sequences, and the generated bit-streams are bit-wise ANDed to produce an output bit-stream. The resulting bit-stream is converted back to standard representation (by counting the number of 1s and dividing by the length of the bit-stream) and compared with the expected multiplication result to find the absolute error. Here, the expected value is $P_{x1} \times P_{x2}$. For accurate multiplication, the input

bit-streams must be *uncorrelated*. In SC literature, *Stochastic Cross-Correlation* (*SCC*) is used to quantify the correlation between bit-streams [1]. We exhaustively evaluated the multiplication accuracy for all cartesian combinations of the $x1$ and $x2$ values where the inputs are 8-bit precision values in the [0,1) interval (i.e., 0/256, 1/256,..., 255/256) with UPE encoding. The Mean Absolute Error (MAE) of the multiplication results are presented in Fig. 1 (a). We multiply the measured mean values by 100 and report them as percentages. Two different sequences are selected for each case to satisfy the uncorrelation requirement. For the `Sobol` sequence, the first two `Sobol` sequences from the `MATLAB` built-in `Sobol` sequence generator are used. For the `Faure` sequence, two sequences are created using a `VDC` base-7. The `Halton` sequence involves two dimensions generated using `VDC` bases of 11 and 13 with the `MATLAB` built-in `Halton` function. For the `Hammersley` sequence, we use the `VDC` sequence with bases 2 and 3 to generate the respective sequences. The `Latin Hypercube` sequence was also generated using its `MATLAB` built-in function. For the `Weyl` sequence, π and the *Silver Ratio* (i.e. $\sqrt{2}-1$) were chosen as the irrational numbers. The first dimension of the `VDC` sequence is selected as base-2, while base-N is selected for the other dimension depending on the length of the bit-stream (N).

As the length of the bit-streams increases, the accuracy of the results improves. Notably, the `VDC`-related sequences exhibit favorable convergence rates. Specifically, after 2^{10} operation cycles, the `VDC` sequence surpasses the `Niederreiter` sequence and approaches proximity to the `Sobol` sequence in terms of accuracy. For approximate results, the `Sobol`, `Niederreiter`, and `VDC` sequences emerge as the top performers. As can be seen in Fig. 1 (a), compared to `Sobol` sequence, `VDC` sequence shows a better convergence when increasing the length of the bit-streams.

3.2 Benchmark-II: SC Addition

Next, we evaluate the accuracy of the SC *Scaled-Addition* operation. We utilize a 2-to-1 `MUX` with two 8-bit precision input operands similar to the multiplication operation. For this SC operation, the two addends (the main inputs of the `MUX`) are correlated ($SCC = 1$), while the `MUX` select input is uncorrelated to the addends [1]. To meet this requirement, we use a random sequence to generate the main input bit-streams and another sequence to generate the bit-stream corresponding to the `MUX` select input. For two-input addition, a bit-stream corresponding to 0.5 value is generated for the select input. Fig. 1 (b) presents the accuracy results of SC addition in terms of MAE for different bit-stream lengths for each sequence. Accurate output (0.0% MAE) can be achieved with a bit-stream length of 2^9 by using sequences such as `Sobol`, `Niederreiter`, and `VDC`. After 2^4 bit-stream lengths, the `VDC` sequence achieves the minimum MAE among the other sequences. By increasing the bit-stream length (N), we can see that the MAE tends to be zero.

3.3 Mid-Level Stream Correlation

Based on the accuracy analysis presented in Sections 3.1 and 3.2, we can see that the standout sequences in terms of MAE for 2-input multiplication and scaled addition are `VDC`, `Sobol`, and `Niederreiter` sequences. But this accuracy is guaranteed for single-stage computation. Still, how random sequences affect the mid-level computations

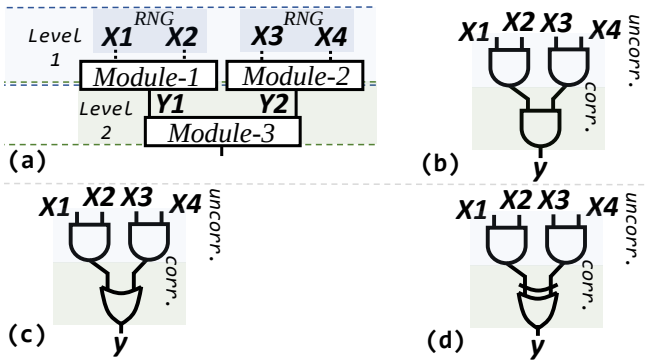

Figure 2: (a) Generic two-level circuit with no correlation control in the mid-level (b) SC Design 1: *Minimum of multiplications*: $Py = Min((P_{X1} \times P_{X2}), (P_{X3} \times P_{X4}))$ (c) SC Design 2: *Maximum of multiplications*: $Py = Max((P_{X1} \times P_{X2}), (P_{X3} \times P_{X4}))$ (d) SC Design 3: *Absolute subtraction of multiplications*: $Py = |(P_{X1} \times P_{X2}) - (P_{X3} \times P_{X4})|$.

of cascaded logic systems remains unexplored in the literature. In this regard, we provide additional analysis for the well-performing `Sobol` and `VDC` sequences with relatively decent `Neiderreiter` sequences. Here, we also include the conventional RNG unit of the SC systems, LFSR, in our evaluations.

Remind that for correct and accurate operation, some SC circuits require uncorrelated bit-streams (e.g., multiplication using bit-wise `AND`). In contrast, others require correlated inputs (e.g., minimum using bit-wise `AND`). Satisfying the correlation requirements in the intermediate stages of some cascaded computations is challenging. While the correlation between the input bit-streams of the first stage can be controlled in the bit-stream generation, the outputs of the previous stages are processed for the intermediate stages. For instance, consider the two-level circuit structure shown in Fig. 2 (a), where the logic elements in the first layer operate on uncorrelated bit-streams. In contrast, the circuit module in the second layer requires correlated bit-streams. Determining the correlation level between the outputs ($Y1$ and $Y2$) is difficult when generating bit-streams independently from different random sequences.

Prior works proposed two methods for correlation manipulation of the bit-streams in the intermediate stages: 1) In-stream correlation manipulation [3], and 2) SC-to-binary conversion and regeneration of bit-streams. Both of these solutions take additional hardware costs. Here, we provide a statistical analysis of the first and second levels of cascaded circuits with pairs of `AND`, `OR`, and `XOR` gates when using different random sequences. To this end, *Module-1* and *Module-2* in the design of Fig. 2 (a) are replaced with `AND-AND`, `OR-OR`, `XOR-XOR`, `AND-OR`, `AND-XOR`, and `OR-XOR` gates, respectively, with uncorrelated bit-stream inputs $X1$, $X2$, $X3$, and $X4$. We examine the correlation level between the output bit-streams $Y1$ and $Y2$. Fig. 3 presents the *SCC* results from over 10^5 test runs. We made an interesting observation: the designs with `Sobol`, `Niederreiter`, and `LFSR`-based sequences exhibit a correlation around zero, forming a bell-shaped distribution for the $Y1$ and $Y2$ outputs. However, we observed that the circuit constructed with the `VDC` sequences generates positively correlated $Y1$ and $Y2$ outputs after the first-level stage with `AND-AND`, `OR-OR`, and `AND-OR`.

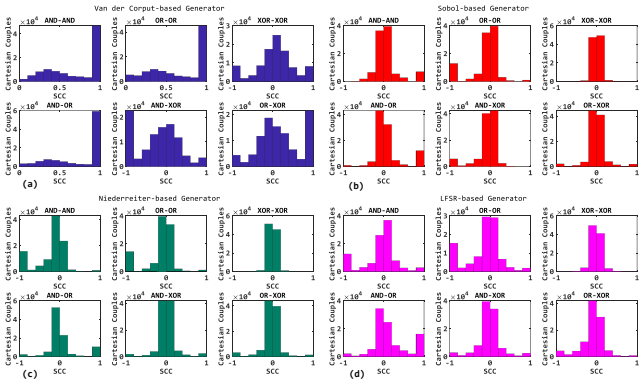

Figure 3: VDC (a), Sobol (b), Niederreiter, and (d) LFSR-based generator results of mid-level circuit output correlations (*SCC*). Sobol uses sequences 1, 2, 3, and 4. VDC uses bases 2, 4, 8, 16. Niederreiter uses sequences 1, 2, 3, and 4. LFSR uses four different maximal lengths LFSRs with polynomials: (i) $x^8+x^6+x^5+x^4+1$, (ii) $x^8+x^4+x^3+x^2+1$, (iii) $x^8+x^6+x^3+x^2+1$, and (iv) $x^8+x^6+x^5+x^1+1$. Bit-stream length, N, is 256.

Table 1: MAE (%) Comparison of the Circuits Shown in Fig. 2.

MAE (%)	VDC	S	N	LFSR
Min. Selector $P_y = Min((P_{X1} \times P_{X2}), (P_{X3} \times P_{X4}))$	**3.7**	6.6	6.9	6.7
Max. Selector $P_y = Max((P_{X1} \times P_{X2}), (P_{X3} \times P_{X4}))$	**5.8**	7.0	6.7	6.2
Absolute Subtractor $P_y = \lvert(P_{X1} \times P_{X2}) - (P_{X3} \times P_{X4})\rvert$	**9.2**	13.6	13.6	12.9

Based on our observations, the VDC sequences can be utilized in different configurations of the cascaded circuits where the design requires different levels of correlations. In this regard, the three circuit topologies presented in Figs. 2 (b), (c), and (d) are anticipated to yield fewer errors. In these circuit topologies, multiplication is performed in the first layer on uncorrelated bit-streams, and correlated bit-streams are required for the SC operations in the second layer. Fig. 2 (b) presents a circuit that selects the minimum of two multiplication results, Fig. 2 (c) presents a circuit that selects the maximum of two multiplication results, and Figs. 2 (d) presents a circuit that computes the absolute difference of the multiplication results. Table 1 provides the MAE results of the circuits shown in Fig. 2 based on 10^5 test runs. As it can be seen, the VDC sequences achieve the best results (lower MAEs).

4 PROPOSED SEQUENCE GENERATOR

In this section, we propose a novel hardware design for generating VDC sequences and evaluate the implementation cost compared to prior LD sequence generators. Alaghi and Hayes [2] implemented a Halton sequence generator consisting of mod counters, digit converters, and an adder. Liu and Han [17] proposed a Sobol sequence generator by using some Direction Vectors (DVs). The DVs ($V_x(x = 0, 1, ..., N-1)$) are generated using some primitive polynomials and stored in a Direction Vector Array (DVA). By employing different DVs, different Sobol sequences can be produced. At any

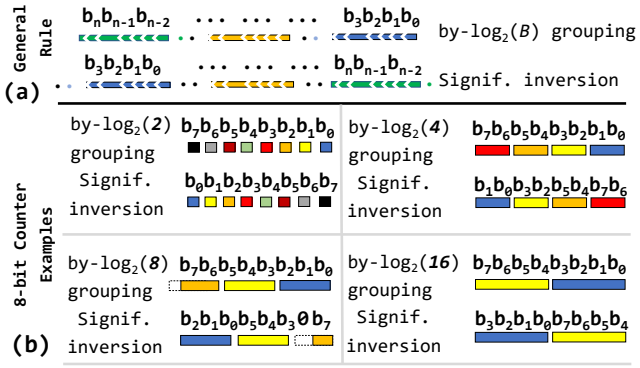

Figure 4: Proposed VDC sequence generator. (a) The general rule to hard-wiring bits for reversing operation (significance inversion), and (b) Example for the 8-bit counter to generate base-2, base-4, base-8, and base-16 (up to base-256 is possible) VDC sequences.

cycle, a priority encoder finds the least significant zero (LSZ) in the output of a counter. Depending on the position of the LSZ, a DV is selected from the DVA. A new Sobol number is recursively generated by XORing the respective DV and the previous Sobol number.

Prior work suggested a look-up table-based approach for VDC sequence generation [10]. In this work, we propose a low-cost VDC sequence generator with simple hardware implementation for efficient and lightweight generation of the VDC sequences, specifically for powers-of-2 bases. Our design involves using $log_2(N)$-bit counters for bases of 2^N, where N is the bit-precision of the bit-stream. To conduct a fair comparison with previous random generators and assess the performance for various image processing applications, we target bit-streams of up to N=256 (sufficient for representing 8-bit grayscale image data). Therefore, we require up to 256 random numbers from the VDC sequence generator to generate each bit-stream. The general algorithm to generate a base-B VDC sequence consists of five steps:

① Generating integer numbers. ② Converting an integer number to its base-B representation. ③ Reversing the base-B representation. ④ Converting the base-B representation to a binary number. ⑤ Converting the input number within the $[0, 1)$ interval to corresponding 8-bit binary number in the $[0, 256)$ range, to be connected to the binary comparator.

The complexity of the hardware design for this algorithm is closely tied to the chosen base. We classify the hardware designs into two categories depending on the base values.

4.1 Class-I: VDC Generator with *Non-Powers-of-2*

To implement this type of VDC sequence generator, we combine the first two steps, ① and ②, by utilizing a base-B counter to generate integer numbers in a specific base representation. For instance, a *Binary Coded Decimal* (BCD) counter can be employed for a base-10 representation. Step ③ can be done by using a wiring technique in the hardware design. Step ④ can be implemented by employing adders and MUXs. This step is relatively complex and takes more

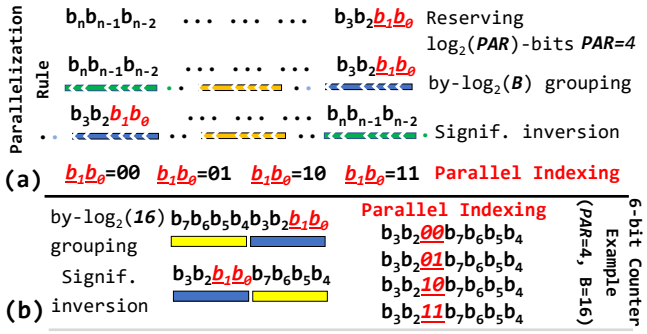

Figure 5: Parallel VDC sequence generator. (a) The general rule to assign parallel indexing bits, and (b) Base-16 example with PAR=4 concurrent generation.

hardware resources compared to the other steps. Step ⑤ can simply be achieved by an 8-bit shift operation.

The Hammersley and Halton sequences extend the VDC sequence to higher dimensions, representing each dimension in a different prime base-B. Consequently, the hardware implementation of these sequences falls under this particular type of sequence generator. The need for counters with prime radices and base conversion make the Halton sequence generator of [2] complex to implement in hardware. The hardware limitations of the design of [2] motivate us to explore the second class of VDC generators for the powers-of-2 bases.

4.2 Class-II: VDC Generator with *Powers-of-2*

To implement this type of sequence generator, a binary counter with sufficient bits is utilized to represent the desired range of integer numbers in step ①. To convert the value of a binary counter to its base-B representation (step ②), we consider groups of $log_2(B)$ bits, starting from the least significant bit. If the last group lacks enough bits, some additional 0 bits are appended via zero padding to ensure it forms a complete group. The reversing operation in step ③ is done by hard-wiring each group of bits, treating them as a single digit in base-B. The process of converting a base-B number to its equivalent binary representation is the inverse of step ②. In this process, each group or base-B digit is considered as equivalent $log_2(B)$ bits of binary representation and any exceeding bits beyond the counter in step ① is discarded. For instance, consider a simple 8-bit precision VDC sequence generator for base-16. An Up-Counter counts toward 256, and the resulting sequence is obtained by hard-wiring the output of each T Flip-Flop (T-FF) in a reverse manner. This is done to inverse the significance of each group, i.e., the least significant group becomes the most significant group, and vice versa. Fig. 4 (a) shows the overall idea behind the proposed VDC sequence generator. After grouping each bit from the counter; the inversion (via hard-wiring) reverses the bit significance, the new binary output is ready for comparison in the SNG block. Fig. 4 (b) illustrates examples of different bases. Assuming that the target N is known in the SC system at the beginning of each operation, B is constructed by B=N, and the grouping-inversion steps with hard-wires are implemented in advance.

Our proposed *Class-II* VDC sequence generator can also operate in parallel. Fig. 5 illustrates how more than one sequence element of a VDC sequence (in any base) can be generated in parallel at any time. Let us define PAR as the number of sequence elements to be generated in parallel. First, $log_2(PAR)$ bits are reserved at the least significant positions. The remaining bits require a reduced precision counter (e.g., 8→6 in Fig. 5). At any clock cycle, the reserved bits are filled with $2^{(log_2(PAR))}$ possible logic values (parallel indexing). Fig. 5 shows an example for PAR=4. In this example, each output repeats four times to fill the reserved bits with 00, 01, 10, and 11. The outputs at any cycle produce four consecutive VDC numbers. Fig. 5 (b) illustrates another example of PAR=4 for VDC base-16.

5 SC IMAGE PROCESSING

In this section, we evaluate the performance of the VDC sequences and the hardware efficiency of the proposed VDC sequence generator in SC image processing case study. Prior work has used SC for low-cost implementation of different computer vision tasks from depth perception to interpolation [5, 6, 12, 21, 25]. We evaluate the sequence generator in an image scaling application.

Interpolation refers to the process of estimating or calculating values between two known data points. Linear interpolation is a method used to estimate values between two known values based on a linear relationship [8]. It assumes a straight line between the available values and calculates intermediate values along that line. In image processing, linear interpolation is used to estimate pixel values between two neighbouring pixels. It is commonly employed when performing operations such as rotation, translation, or affine transformations on images. Bilinear interpolation is a specific case of linear interpolation applied in two dimensions. Instead of estimating values along a straight line, it estimates values within a two-dimensional grid of pixels [9]. Bilinear interpolation considers the four nearest pixels to the target location and calculates a weighted average based on their values. The weights are determined by the distances between the target location and the surrounding pixels; thereby, an image scaling task can be performed [20].

Assume we have an original image, I, with pixel values represented by a 2-D array. We want to estimate the pixel value at a non-integer coordinate (x, y) in the image. The four surrounding pixels to consider are (x_1, y_1), (x_1, y_2), (x_2, y_1), and (x_2, y_2), where (x_1, y_1) represents the pixel at the bottom-left corner of the target location, and (x_2, y_2) represents the pixel at the top-right corner. Let us denote the pixel values as $I(\mathrm{x}, \mathrm{y})$, $I(x_1, y_1)$, $I(x_1, y_2)$, $I(x_2, y_1)$, and $I(x_2, y_2)$. The bilinear interpolation formula to estimate the pixel value $I(\mathrm{x}, \mathrm{y})$ is as follows: $I(\mathrm{x}, \mathrm{y}) = (1-u)(1-v)\times I(x_1, y_1)+(1-u)v\times I(x_1, y_2)+u(1-v)\times I(x_2, y_1)+uv\times I(x_2, y_2)$, where $u = x - x_1$ (fractional distance between x and x_1) and $v = y - y_1$ (fractional distance between y and y_1). The values $(1-u)(1-v)$, $(1-u)v$, $u(1-v)$, and uv are the weights assigned to each surrounding pixel. These weights represent the contribution of each pixel to the interpolated value. The interpolation formula can be compared to a multiplication-based SC MUX structure [7], where neighbouring pixels are fed into the main MUX inputs, and the location information is fed into the selection ports. In this scenario, a 4-to-1 MUX can be expressed in terms of probabilities as follows: $P_{I(\mathrm{x},\mathrm{y})} = (1 - P_{\boldsymbol{u}})(1 - P_{\boldsymbol{v}})P_{I_{11}} + (1 - P_{\boldsymbol{u}})(P_{\boldsymbol{v}})P_{I_{12}} + (P_{\boldsymbol{u}})(1 - P_{\boldsymbol{v}})P_{I_{21}} + (P_{\boldsymbol{u}})(P_{\boldsymbol{v}})P_{I_{22}}$.

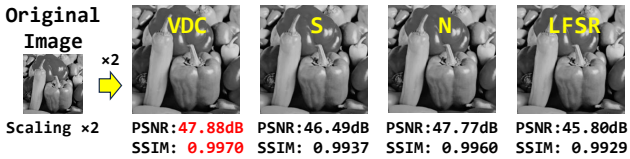

Figure 6: Visual results of SC image scaling using different SNGs (with VDC, Sobol, Niederreiter, or LFSR) and a 4-to-1 MUX.

Table 2: Hardware Cost Comparison of the Image Scaling process.

LD Sequence	Area (μm^2)	Energy* (pJ)	Delay* (ns)	Total Energy (μJ)
Sobol	2017	17.55	366	0.781
Parallel 4× Sobol	4548	16	91.5	0.713
Proposed VDC	**715**	**5.60**	**317**	**0.250**
Parallel 4× VDC	**2040**	**2.40**	**71**	**0.107**

* Energy and Delay for producing each output pixel. Bit-stream Length (N) is 256.

Fig. 6 visually demonstrates the results of 2× image scaling with an SC circuit composed of SNGs for data conversion and a 4-to-1 MUX unit. We evaluated the SC circuit for the cases of using the VDC, Sobol, Niederreiter, and LFSR random sequences in the SNG units. The values shown in Fig. 6 exhibit the superior performance of the VDC sequences. Furthermore, we evaluate the performance and energy consumption in 45nm CMOS technology when processing the *Pepper* image (107 × 104 image size) for the two cases of VDC and Sobol. The results are reported in Table 2. As reported, the non-parallel and the 4× parallel designs of the VDC-based implementation save area by 64% and 55%, energy by 67% and 85%, and delay by 13% and 22% compared to the non-parallel and 4× parallel Sobol-based implementation, respectively.

6 CONCLUSIONS

This study explores new design possibilities for SC by analyzing some well-known random sequences in the literature. As a promising random sequence for the SNG unit of SC systems, we evaluated the performance of the Van der Corput (VDC) sequences, revealing their interesting correlation properties. We proposed a lightweight hardware design for VDC sequence generation. The proposed generator provides a higher hardware efficiency compared to the SOTA LD sequence generators. We evaluated the performance of the VDC-based SNGs in an image scaling case study. Our performance evaluation and hardware cost comparison show comparable or better numbers compared to the SOTA. Our finding opens possibilities for incorporating the VDC sequences in other emerging paradigms that require orthogonal vectors, such as hyperdimensional computing. The VDC sequences can be utilized to address the computational needs and improve the performance of such paradigms. We leave studying this aspect for our future work.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation (NSF) grant #2019511, the Louisiana Board of Regents Support Fund #LEQSF(2020-23)-RD-A-26, the Louisiana Space & Sea Grant Opportunities (LaSSO) award #NA22OAR4170105, and generous gifts from Cisco, Xilinx, and Nvidia.

REFERENCES

[1] Armin Alaghi and John P. Hayes. 2013. Exploiting correlation in stochastic circuit design. In *ICCD*. Asheville, NC, USA, 39–46.
[2] Armin Alaghi and John P. Hayes. 2014. Fast and accurate computation using stochastic circuits. In *2014 DATE*. 1–4. https://doi.org/10.7873/DATE.2014.089
[3] Sina Asadi, M. Hassan Najafi, and Mohsen Imani. 2021. CORLD: In-Stream Correlation Manipulation for Low-Discrepancy Stochastic Computing. In *2021 ICCAD*. 1–9. https://doi.org/10.1109/ICCAD51958.2021.9643450
[4] Sercan Aygun, Mehran Shoushtari Moghadam, M. Hassan Najafi, and Mohsen Imani. 2023. Learning from Hypervectors: A Survey on Hypervector Encoding. arXiv:2308.00685 [cs.LG]
[5] Sercan Aygun, M. Hassan Najafi, Mohsen Imani, and Ece Olcay Gunes. 2023. Agile Simulation of Stochastic Computing Image Processing with Contingency Tables. *IEEE TCAD* (2023). https://doi.org/10.1109/TCAD.2023.3243136
[6] Sercan Aygün, Mustafa Altun, and Ece Olcay Güneş. 2017. Sobel filter operation in image processing via stochastic arithmetic-logic unit design. In *2017 IEEE SIU*. https://doi.org/10.1109/SIU.2017.7960479
[7] Timothy J. Baker and John P. Hayes. 2022. CeMux: Maximizing the Accuracy of Stochastic Mux Adders and an Application to Filter Design. *ACM TDAES* 27, 3, Article 27 (jan 2022), 26 pages. https://doi.org/10.1145/3491213
[8] T. Blu, P. Thevenaz, and M. Unser. 2004. Linear interpolation revitalized. *IEEE Transactions on Image Processing* 13, 5 (2004), 710–719. https://doi.org/10.1109/TIP.2004.826093
[9] K.T. Gribbon and D.G. Bailey. 2004. A novel approach to real-time bilinear interpolation. In *DELTA 2004*. 126–131. https://doi.org/10.1109/DELTA.2004.10055
[10] Vincent T. Lee. 2019. *Towards practical stochastic computing architectures for emerging applications*. Ph. D. Dissertation. University of Washington, Seattle, USA. http://hdl.handle.net/1773/43658
[11] Vincent T. Lee, Armin Alaghi, Rajesh Pamula, Visvesh S. Sathe, Luis Ceze, and Mark Oskin. 2018. Architecture Considerations for Stochastic Computing Accelerators. *IEEE TCAD* 37, 11 (2018), 2277–2289. https://doi.org/10.1109/TCAD.2018.2858338
[12] Peng Li, David J. Lilja, Weikang Qian, Kia Bazargan, and Marc D. Riedel. 2014. Computation on Stochastic Bit Streams Digital Image Processing Case Studies. *IEEE TVLSI* 22, 3 (2014).
[13] Zhe Li, Ji Li, Ao Ren, Ruizhe Cai, Caiwen Ding, Xuehai Qian, Jeffrey Draper, Bo Yuan, Jian Tang, Qinru Qiu, and Yanzhi Wang. 2019. HEIF: Highly efficient stochastic computing-based inference framework for deep neural networks. *IEEE TCAD* 38, 8 (2019), 1543–1556. https://doi.org/10.1109/TCAD.2018.2852752
[14] C. Devon Lin and Boxin Tang. 2022. Latin Hypercubes and Space-filling Designs. arXiv:2203.06334 [stat.ME]
[15] Zhendong Lin, Guangjun Xie, Wenbing Xu, Jie Han, and Yongqiang Zhang. 2021. Accelerating Stochastic Computing Using Deterministic Halton Sequences. *IEEE TCAS II* 68, 10 (2021), 3351–3355. https://doi.org/10.1109/TCSII.2021.3073680
[16] Siting Liu and Jie Han. 2017. Energy efficient stochastic computing with Sobol sequences. In *2017 DATE*. 650–653. https://doi.org/10.23919/DATE.2017.7927069
[17] S. Liu and J. Han. 2018. Toward energy-efficient stochastic circuits using parallel sobol sequences. *IEEE TVLSI* 26, 7 (2018).
[18] Luka Marohnic et al. 2012. Plastic Number: Construction and Applications.
[19] M. Hassan Najafi, Devon Jenson, David J. Lilja, and Marc D. Riedel. 2019. Performing Stochastic Computation Deterministically. *IEEE TVLSI* 27, 12 (2019), 2925–2938. https://doi.org/10.1109/TVLSI.2019.2929354
[20] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. 1995. *Numerical Recipes in C: The Art of Scientific Computing*. Camb. Univ. Press, New York, NY, USA.
[21] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. 2011. An Architecture for Fault-Tolerant Computation with Stochastic Logic. *Computers, IEEE Trans. on* 60, 1 (Jan 2011), 93–105. https://doi.org/10.1109/TC.2010.202
[22] Mohsen Riahi Alam, M. Hassan Najafi, and Nima TaheriNejad. 2021. Exact Stochastic Computing Multiplication in Memristive Memory. *IEEE Design & Test* 38, 6 (2021), 36–43. https://doi.org/10.1109/MDAT.2021.3051296
[23] Paishun Ting and John P. Hayes. 2017. Eliminating a hidden error source in stochastic circuits. In *2017 IEEE DFT*. 1–6. https://doi.org/10.1109/DFT.2017.8244436
[24] J. G. van der Corput. 1935. Verteilungsfunktionen. I. *Proc. Akad. Wet. Amsterdam* 38 (1935), 813–821.
[25] Ran Wang, Jie Han, Bruce F. Cockburn, and Duncan G. Elliott. 2016. Stochastic Circuit Design and Performance Evaluation of Vector Quantization for Different Error Measures. *IEEE TVLSI* 24, 10 (2016), 3169–3183. https://doi.org/10.1109/TVLSI.2016.2535313