CORLD: In-Stream Correlation Manipulation for Low-Discrepancy Stochastic Computing

Sina Asadi*, M. Hassan Najafi* and Mohsen Imani†
sina.asadi1@louisiana.edu, najafi@louisiana.edu, m.imani@uci.edu
*School of Computing and Informatics, University of Louisiana at Lafayette, LA, USA
†Department of Computer Science, University of California Irvine, CA, USA

2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 978-1-6654-4507-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCAD51958.2021.9643450

***Abstract*—Stochastic computing (SC) is a re-emerging computing paradigm providing low-cost and noise-tolerant designs for a wide range of arithmetic operations. SC circuits operate on uniform bit-streams with the value determined by the probability of observing 1's in the bit-stream. The accuracy of SC operations highly depends on the correlation between input bit-streams. While some operations such as minimum and maximum value functions require highly correlated inputs, some other such as multiplication operation need uncorrelated or independent inputs for accurate computation. Developing low-cost and accurate correlation manipulation circuits is an important research in SC as these circuits can manage correlation between bit-streams without expensive bit-stream regeneration. This work proposes a novel in-stream *correlator* and *decorrelator* circuit that manages 1) correlation between stochastic bit-streams, and 2) distribution of 1's in the output bit-streams. Compared to state-of-the-art solutions, our designs achieve lower hardware cost and higher accuracy. The output bit-streams enjoy a low-discrepancy distribution of bits which leads to higher quality of results. The effectiveness of the proposed circuits is shown with two case studies: SC design of sorting and median filtering.**

I. Introduction

Stochastic computing (SC) [1], [2] is an unconventional computing paradigm offering low-cost and noise-tolerant solutions for a wide range of arithmetic operations from multiplication [3], [4] to division [5]–[7], square root [8], scaled addition and subtraction, minimum and maximum value functions [9], [10], and trigonometric, logarithmic, and exponential functions [11], [12]. SC treats data as probabilities presented by streams of random bits. The value of a SC bit-stream is determined by the ratio of 1's in the bit-stream, i.e., a bit-stream with P ones and length L represents P/L value. For example, 10011000 with P=3 and L=8 represents 3/8 in the stochastic domain. This unconventional method of representing data leads to extremely simple computation circuits for complex arithmetic operations. For example, multiplication operation can be realized by bit-wise ANDing stochastic bit-streams [3]. But this simple method of multiplying data is accurate only if the input bit-streams are statistically *independent* or *uncorrelated*. Bit-wise ANDing two correlated bit-streams with high overlap between the position of 1's in the bit-streams gives the minimum of the two bit-streams and not their product. Correlation, hence, plays an important role in correct functionality of SC circuits [9], [10], [13].

In stochastic systems, both correlated and uncorrelated bit-streams are needed subject to desired function. Two bit-streams are *positively correlated* when all 1's in the two bit-streams are completely overlapped, i.e., they are exactly in the same bit positions. For example, 101100 and 10100 are two positively correlated bit-streams. Stochastic operations such as minimum using bit-wise AND, maximum using bit-wise OR, absolute difference using bit-wise XOR, and division [5] are examples of operations that need positively correlated inputs. When there is no or minimum overlap between the bit positions of 1's in the bit-streams, the bit-streams are called *negatively correlated*. For example, 101100 and 010011 are two negatively correlated bit-streams. Saturating addition using bit-wise OR is an example of a stochastic operation that needs negatively correlated inputs [13]. Operations such as multiplication using bit-wise AND and scaled addition using multiplexer need *uncorrelated* inputs. For example, 1010 and 1001 are two uncorrelated bit-streams both representing 0.5 value. Bit-wise ANDing these two bit-streams gives 1000, the expected value (0.25) from multiplying the two inputs.

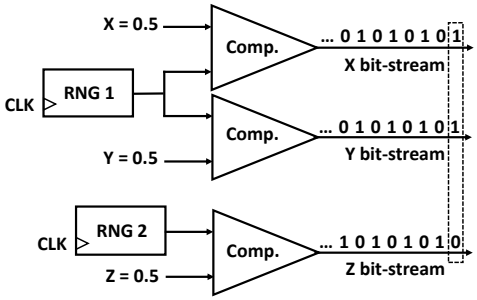

Fig. 1: Example of generating correlated and uncorrelated bit-streams. X and Y bit-streams are correlated, and both uncorrelated with bit-stream Z due to using the same and different RNGs in generating them.

An important part of a stochastic system is the data conversion unit that converts the input data from positional binary to bit-stream representation. The common approach for implementing this unit is by using a binary comparator and a random number generator (RNG). In each clock cycle, the input data in binary format is compared with a random number from the RNG. A 1 is produced at the output of the comparator if the random number is smaller than the input data. Correlation between input bit-streams can be controlled at this stage when the bit-streams are generated. Sharing the same RNG between different data conversion units results in generating correlated bit-streams. Using different RNGs, on the other hand, leads to generating uncorrelated bit-streams. Fig. 1 shows this using an example circuit.

978-1-6654-4507-8/21/$31.00 ©2021 IEEE

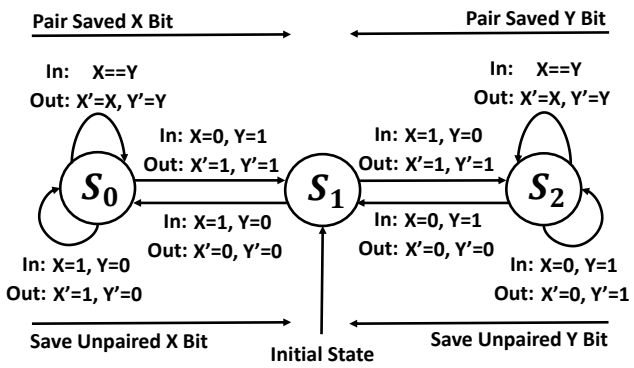

Fig. 2: Synchronizer circuit of [13] with depth = 1.

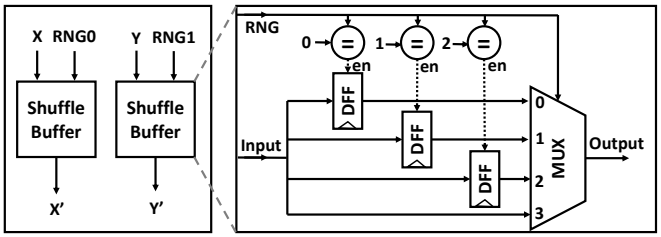

Fig. 3: Decorrelator circuit of [13] with depth = 4.

While this method is common for controlling correlation between the inputs of the stochastic system at the first stage of computations, it cannot be used in the next stages of the system as the data are already in the bit-stream form. A naive method for manipulating correlation in the intermediate stages is to convert the bit-streams back to the positional binary format and then re-generate them with the desired correlation. But this approach incurs significant area and power overheads. Developing low-cost correlation manipulation circuits for in-stream (i.e., without re-generating bit-streams) managing of correlation between bit-streams is an ongoing research.

Lee et al. [13] developed a novel synchronizer and desynchronizer circuit for increasing positive and negative correlation respectively between two bit-streams. Fig. 2 depicts the synchronizer (correlator) circuit of [13] with depth size = 1. Given two stochastic bit-streams X and Y, the synchronizer produces two bit-streams which are more positively correlated. They also developed a novel decorrelator circuit (Fig. 3) to reduce correlation between two SC bit-streams. At each cycle, the decorrelator circuit either passes the current input bit or stores it in a shuffle buffer and emits a previously stored bit by scrambling the stored bits using an RNG. They use these circuits to propose improved SC maximum, minimum, and saturating adder designs. Wu and Miguel [6] also developed an in-stream correlation-based division and square root circuit by introducing a skewed synchronizer.

This work proposes an improved correlator that can increase positive correlation between stochastic bit-streams. We focus on positive correlation, and leave negative correlation for our future work, as positive correlation is more common and needed by more SC operations. We further develop a decorrelator circuit that can make input bit-streams uncorrelated with negligible impact on their value. An important advantage

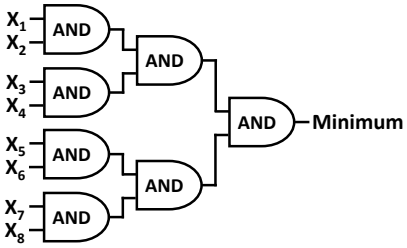

Fig. 4: 8-input SC Minimum Circuit

of our proposed circuits is that the output bit-streams have low-discrepancy (LD) distribution [14]. With LD distribution, 1's and 0's are uniformly spaced. Random fluctuations are removed from bit-streams and the bit-streams converge to target values significantly faster than the true- or pseudo-random bit-streams [15], [16]. This significantly improves the accuracy and reduce the processing time of SC operations. In summary, the main contributions of this work are as follows:

- A novel correlator and decorrelator circuit for in-stream correlation manipulation of stochastic bit-streams with no significant impact on the data values.
- The proposed circuits output high-quality LD bit-streams irrespective of the distribution of the input bit-streams.
- The proposed circuits can manipulate correlation of any number of inputs.
- The accuracy, performance, and hardware cost are all improved compared to those of the state-of-the-art (SoA) correlation manipulation techniques.
- A finite-state machine (FSM)-based correlator circuit for SC division.
- Significant improvement in the accuracy and hardware cost of SC sorting and median filtering circuits.

The rest of the paper is organized as follows: Section II demonstrates the proposed LD correlator. Section III introduces our LD decorrelator. Section IV discusses our correlator circuit for SC division. Section V evaluates the accuracy and the hardware cost of the proposed techniques. Section VI evaluates the proposed techniques in two application case studies. Finally, Section VII concludes the paper.

II. Proposed Low-Discrepancy Correlator

As we discussed in Section I, the common approach for generating correlated bit-streams is to use the same RNG in the data conversion units when generating bit-streams. When correlated bit-streams with LD distribution are desired the same Sobol [15] or Halton [16] number generator can be shared between the data conversion units to generate LD bit-streams that have maximum overlap between the bit positions of 1's, i.e., LD bit-streams with maximum positive correlation. But this approach is only applicable to the input stage of the SC system where the input data are converted from binary to bit-stream representation.

Consider a simple SC circuit, consisted of seven `AND` gates, that finds the minimum value between eight input numbers (Fig. 4). The data are split into four pairs of two numbers, each pair connected to one `AND` gate. Each `AND` gate finds

the minimum of its two inputs. An advantage of the common approach of generating correlated bit-streams is that because correlated input bit-streams are directly generated, the output bit-streams from operations such as minimum using AND or maximum using OR are also correlated and so can directly be fed to the next stages (if correlated bit-streams are needed) with no need for any correlation manipulation. So, with this approach, the minimum of the four produced output bit-streams are found by connecting them to three AND gates in two more stages as shown in Fig. 4. But the conventional approach can be used in this design example when the inputs are in the binary format and we have control over bit-stream generation.

The question is what if the inputs to the SC system are already in the bit-stream form and they are uncorrelated or the correlation level is unknown? A solution is to convert the bit-streams back to binary format and then re-generate them by sharing the same RNG. But this approach costs a significant latency, area, and power overhead. The in-stream synchronizer proposed in [13] can be used in such designs to increase the positive correlation between input bit-streams with no need for re-generating them. While the synchronizer can manipulate correlation at a lower cost compared to the bit-stream re-generation method, it has two weaknesses: 1) it lacks the advantage of the conventional approach; the output bit-streams from operations such as minimum and maximum operation are not necessarily correlated. An additional synchronizer is needed to make any two output bit-streams correlated. This will lead to a significant correlation manipulation cost for SC systems with multiple stages of computation. 2) the bit-streams produced by the synchronizer are random and so suffer from random fluctuations [2]. Even in cases that the inputs to the synthesizer have LD distribution, there is no guarantee that the synchronizer provides the same uniform distribution in its output bit-streams.

Here we propose an in-stream correlator called CORLD-C that addresses the weaknesses of the SoA synchronizer technique. Fig. 5 shows our proposed correlator with a *fixer size (FS)* of 2. The proposed correlator processes the input bit-stream in segments of 2^{FS} bits and converts the value of each segment into an LD bit-stream. CORLD-C consists of two counters (an input-controlled counter and an independent counter), a register, and a binary comparator. In the correlator of Fig. 5, FS is 2 indicating that the number of bits that can be taken in each segment is 4. The circuit counts the number of 1's in each 4-bit segment of the bit-stream, stores the value in a register, and then restarts. While the top counter processes the next segment, concurrently the bottom part converts the counted segment to bit-stream using the bottom counter and comparator. A counter is a Sobol sequence generator if reversing its output bits [15]. So the output bits of the bottom counter are connected to the comparator in reverse order to compare the value stored in the register with a new Sobol number in each cycle. The comparator produces one bit of the final bit-stream in each cycle. Since there are at most 4 ones in each 4-bit segment, the proposed circuit should be able to count from 0 to 4 and then restart. Therefore, when the bottom counter reaches its first state (CN=0), an *Enable*

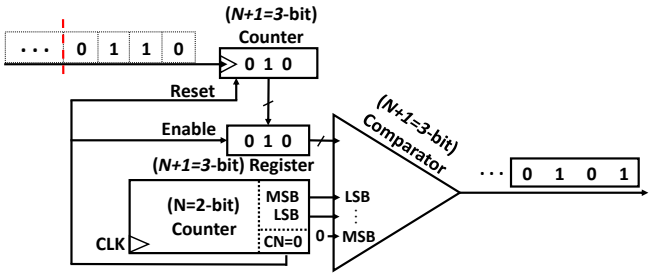

Fig. 5: Proposed Correlator (CORLD-C) with FS=2

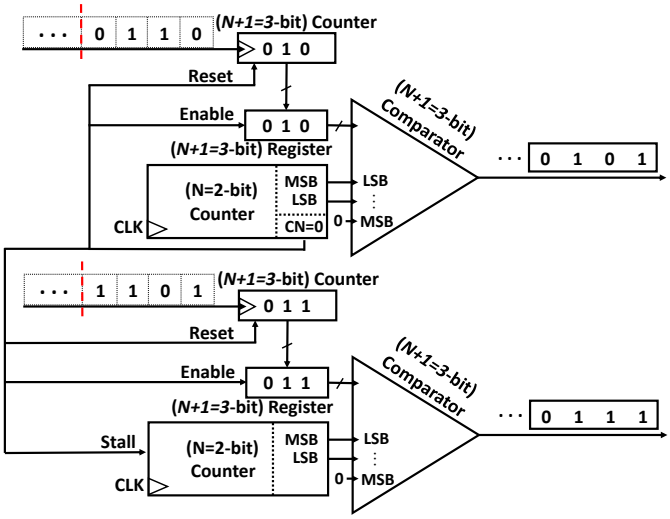

Fig. 6: Proposed Decorrelator (CORLD-D) with *FS*=2

signal is set to copy the value of the top counter into the register, and then the top counter restarts. It should be noted that both *Enable* and *Reset* signals are clock edge sensitive; *Enable* is sensitive to the positive edge and *Reset* changes with the negative edge of the clock signal. The output bit-streams produced by the proposed correlator are all LD and correlated as they are all manipulated based on the same Sobol sequence.

III. Proposed Low-Discrepancy Decorrelator

Conventionally, different RNGs are used in the data conversion units to generate uncorrelated (independent) stochastic bit-streams. By comparing the input data with Sobol numbers from different Sobol sequences, the data can be converted to independent LD bit-streams [15], [17]. But similar to the discussion on correlation, this method can only be used and is efficient if we have control over the data conversion units. If the input bit-streams are already generated, e.g., they are the outputs from other stochastic circuits, and our SC system requires uncorrelated input bit-streams, in-stream decorrelator circuits are needed to minimize correlation between input bit-streams and guarantee correct functionality of the system.

Fig. 6 shows our proposed in-stream decorrelator, CORLD-D, with FS = 2. The proposed circuit consists of binary counters, registers, and comparators, and is capable of converting two bit-streams, whether they are correlated or not, into two independent LD bit-streams. The input bit-streams are segmented into sub-parts of 2^{FS} bits and processed by the decorrelator circuit to produce independent bit-streams. The

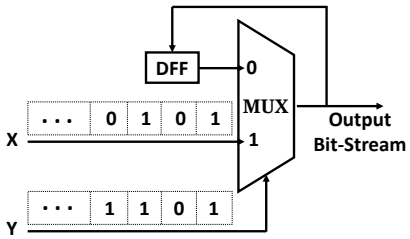

Fig. 7: CORDIV SC Division Circuit [5]

first bit-stream is manipulated using the same approach utilized in CORLD-C. The second bit-stream is manipulated using a similar approach but with this difference that the bottom counter, which is connected to the comparator, is halted for one clock cycle after each round of counting the bits. This is inspired by the rotation technique of [18].

IV. Correlation and SC Division

Before evaluating the proposed correlator and decorrelator, we discuss the SoA correlation-based SC division circuits, CORDIV [5] and ISCBDIV [6], and propose a correlator circuit to improve the performance of the CORDIV design.

Chen and Hayes developed a novel SC division circuit, called CORDIV [5], that exploits the correlation between input bit-streams to realize division operation. The architecture of CORDIV is shown in Fig. 7. The result of X/Y falls outside [0,1] – the accepted range of numbers in SC – if $X > Y$. So for the SC division circuit, it is assumed that the divisor Y is greater than the dividend X. CORDIV requires that the X and Y bit-streams are positively correlated. Our simulations show that the Mean Absolute Error (MAE) rate of the division operation when connecting two 2^8-bit correlated LD bit-streams (generated by sharing the simplest Sobol sequence as the source of the random number in the data conversion unit) to CORDIV is no less than 2.79% (see the last column of Table I).

A weakness of CORDIV is that it needs an expensive organization to regenerate bit-streams to guarantee correlation between them [5]. Inspired by CORDIV, Wu and Miguel developed an in-stream correlation-based division technique called ISCBDIV [6]. Fig. 8 shows the architecture of ISCBDIV. ISCBDIV uses a skewed synchronizer (SS) to correlate the input bit-streams for the CORDIV kernel. Two bit-streams are fed into the SS unit regardless of their correlation. The SS unit leverages the assumption that the divisor is larger than the dividend and reorders the dividend bit-stream based on the divisor. The 1's in the dividend bit-stream are recorded when the divisor bit is 0 and then the saved 1's are paired with the upcoming 1's in the divisor bit-stream. The D-Flip Flop of CORDIV is replaced with a 2-bit shift register (SR). Some extra logic gates are also used to generate a single bit random number (RN) which is connected to the select input of the multiplexer (MUX). The MAE rates of ISCBDIV are reported in Table I.

In this section, we propose an FSM-based correlator that improves the accuracy of the SC division when the bit-streams

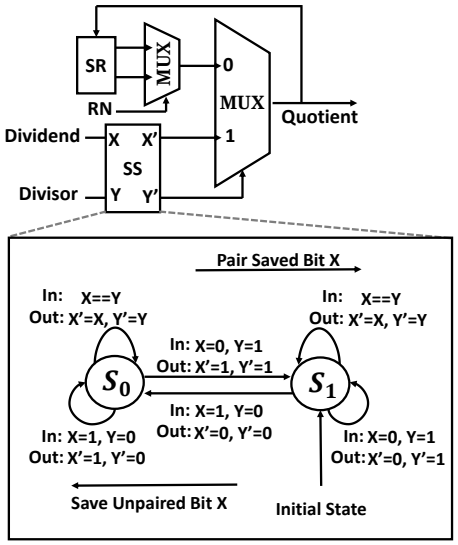

Fig. 8: ISCBDIV SC Division Circuit [6]

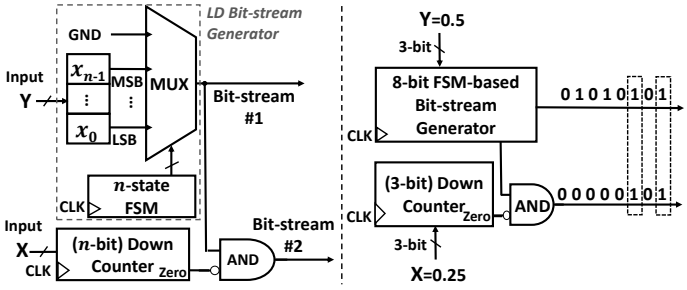

Fig. 9: (left) Proposed FSM-based correlator for SC division (right) An example for n=3.

are connected to the CORDIV circuit. The proposed correlator receives data in binary format and generates correlated bit-streams. At the cost of a down-counter and an AND gate, and by using the SoA FSM-based LD bit-stream generator [14], an LD bit-stream (divisor) correlated with another LD bit-stream (dividend) is generated. The proposed correlator is shown in Fig. 9. A similar assumption here is that $X < Y$. The MAE rates of the proposed FSM-based correlator when connected to CORDIV are reported in Table I. As it can be seen, the proposed correlator achieves the minimum MAE rate among the SoA SC division techniques.

V. Evaluation

In this section, we evaluate the accuracy and the hardware cost of the proposed correlator and decorrelator circuit compared to the SoA correlation manipulation techniques.

A. Accuracy Comparison

1) Correlator: The proposed correlator is evaluated and compared with the SoA synchronizer technique [13] for minimum (bit-wise AND), maximum (bit-wise OR), and absolute difference (bit-wise XOR) function. All these functions require correlated input bit-streams for correct functionality. As the input, we first use two uncorrelated LD bit-streams with

TABLE I: Accuracy Evaluation of the Proposed Correlator with LD Input Bit-streams of 2^N bits (N=Input Width).

Input Width	Input SCC	Method	Depth /FS	Output SCC	Minimum Function MAE (%)	Maximum Function MAE (%)	ABS-Diff Function MAE (%)	Division Function MAE (%)	Division Method	MAE (%)	CORDIV MAE (%) Correlated Sobol 1 Based Inputs
6	1.58	Synchronizer [13]	1	99.48	0.1	1.37	1.47	7.13	Proposed FSM-based Correlator + CORDIV	0.95	2.82
			2	99.97	0	2.83	2.83	9.25			
			3	99.97	0	4.31	4.31	11.36			
			4	99.97	0	5.74	5.74	13.43			
			5	99.97	0	7.11	7.11	15.41			
		CORLD-C	2	93.59	0.51	0.51	1.03	3.39	ISCBDIV (SS Depth=FS)	3.15	
			3	98.29	0.12	0.12	0.24	2.84		3.1	
			4	99.65	0.02	0.02	0.04	2.82		2.98	
			5	100	0	0	0	2.82		2.96	
7	1.73	Synchronizer [13]	1	99.63	0.08	0.62	0.72	5.93	Proposed FSM-based Correlator + CORDIV	0.52	2.8
			2	99.99	∼0	1.35	1.35	7.06			
			3	99.99	0	2.12	2.12	8.18			
			4	99.99	0	2.87	2.87	9.31			
			5	99.99	0	3.6	3.6	10.42			
		CORLD-C	2	93.71	0.52	0.52	1.04	3.47	ISCBDIV (SS Depth=FS)	2.46	
			3	98.30	0.13	0.13	0.26	2.84		2.28	
			4	99.56	0.03	0.03	0.06	2.77		2.23	
			5	99.90	∼0	∼0	0.01	2.77		2.2	
8	0.5	Synchronizer [13]	1	99.58	0.08	0.31	0.41	5.68	Proposed FSM-based Correlator + CORDIV	0.25	2.79
			2	99.997	∼0	0.64	0.64	6.25			
			3	100	0	1.02	1.02	6.83			
			4	100	0	1.41	1.41	7.42			
			5	100	0	1.79	1.79	8.01			
		CORLD-C	2	93.42	0.51	0.51	1.04	3.56	ISCBDIV (SS Depth=FS)	1.97	
			3	98.31	0.12	0.12	0.26	2.87		1.67	
			4	99.63	0.03	0.03	0.06	2.76		1.64	
			5	99.92	∼0	∼0	0.01	2.76		1.63	

different lengths of 2^N bits where N is the input bit-width = 6, 7, and 8. MAE rates between the expected and the produced results are provided in Table I for different *FS* ranging from 2 to 5 for the proposed correlator, and different *depths* ranging from 1 to 5 for the synchronizer technique.

The correlation between two bit-streams X and Y is quantified using the SC Correlation (SCC) as defined in [9]:

$$SCC(X,Y) = \begin{cases} \frac{ad-bc}{N\times min(a+b,a+c)-(a+b)(a+c))} & ad > bc \\ \\ \frac{ad-bc}{(a+b)(a+c)-N\times max(a-d,0)} & else \end{cases}$$

In this formula, a is the number of bit positions where both bit-streams (X and Y) are 1, b is the number of bit positions where X is 1 and Y is 0, c is the number of bit positions where X is 0 and Y is 1, and d is the number of bit positions where both bit-streams are 0. A SCC equal to 0 means the bit-streams are completely uncorrelated. SCC values close to +1 or -1 show high positive or negative correlation, respectively. A positive correlation of +1 means that there is maximal correlation between the two bit-streams. Notice that because the SCC values are small numbers, *the numbers we report in the tables are SCCs multiplied by 100.*

The MAE of the minimum, maximum, and absolute difference functions is zero when their input bit-streams are positively correlated, i.e. SCC = 1.0 (100.0 when multiplying SCC by 100). As the results presented in Table I suggest, the SCC value could be deceptive where some number of 1's are removed from the input bit-stream by the correlation manipulation technique. This is especially the case with the maximum value function. When *depth* increases in the synchronizer technique, the MAE of the maximum function increases due to the number of 1's stuck in the FSM states in the final cycles. This similarly affects the absolute difference function since the bias (defined as the deviation from the original value [13]) of the larger bit-stream increases as *depth* increases. With our proposed correlator, although in some cases the output SCC is less than that of the synchronizer, the bias of both input bit-streams is zero, resulting in accurate minimum, maximum, and absolute difference function.

We further evaluate the performance of the proposed correlator when inputs are pseudo-random bit-streams. Maximal period linear-feedback shift registers (LFSRs) are used to generate pseudo-random bit-streams corresponding to each input bit-width. The SCC values and the MAE rates for the minimum and maximum functions are reported in Table III. As it can be seen, similar to the case with LD bit-streams, the synchronizer faces an increase in the MAE rates when the maximum function is applied on pseudo-random bit-streams and *depth* increases. The performance of CORLD-C, on the other hand, improves (MAE decreases) as *FS* increases.

Multiplication (using `AND`) is another common SC operation. As a different type of input, we evaluate the performance of CORLD-C in a common case that the inputs to the correlator are the outputs from some SC multiplication operations. The SoA work uses LD Sobol-based input bit-streams for accurate multiplication [2], [18]. The output bit-streams however are random and do not follow LD distribution. Table II reports the input and output SCC, and the MAE rates of the minimum and maximum functions for the case that the inputs to the correlator are the outputs of bit-wise `AND`ing two pairs of 256-bit LD bit-streams generated by using MATLAB built-in Sobol 1 and Sobol 2 sequences. We also evaluate a case that one pair of the inputs is generated using Sobol 3 and Sobol 4 sequences. As reported in Table II, CORLD-C provides accurate outputs

TABLE II: Accuracy Evaluation of the Proposed Correlator and Decorrelator when Inputs are 2^8-bit Outputs from SC Multiplication

Input Type	Input SCC	Multiplication. Function MAE (%)	Depth/ FS	Synchronizer [13]			CORLD-C		Decorrelator [13]		CORLD-D	
				Output SCC	Min. Func. MAE (%)	Max. Func. MAE (%)	Output SCC	Min. or Max. Func.s MAE (%)	Output SCC	Multiplication. Function MAE (%)	Output SCC	Multiplication. Function MAE (%)
Sobol based 1&2 and 1&2	80.29	18.96	1	99.82	0.04	0.38	N/A	N/A	N/A	N/A	N/A	N/A
			2	100	0.003	0.74	96	0.63	24.17	2.63	4.87	1.48
			3	100	∼0	1.11	98.74	0.23	14.28	1.7	2.3	0.84
			4	100	0	1.51	99.58	0.07	9.29	2.15	1.23	0.61
			5	100	0	1.88	99.86	0.02	10.55	1.47	0.78	0.35
Sobol based 1&2 and 3&4	1.27	20.08	1	98.2	0.37	0.54	N/A	N/A	N/A	N/A	N/A	N/A
			2	99.92	0.025	0.64	90.7	1.4	0.51	0.87	1.46	1.05
			3	99.99	0.005	1.03	97.48	0.44	3.95	1	0.78	0.69
			4	100	∼0	1.4	99.27	0.13	2.72	0.87	0.51	0.53
			5	100	0	1.78	99.67	0.05	5.21	1.02	0.38	0.43

TABLE III: Accuracy Evaluation of the Proposed Correlator (CORLD-C) with Pseudo-random Input Bit-streams.

Input Width	Input SCC	Method	Depth /FS	Output SCC	Minimum Function MAE (%)	Maximum Function MAE (%)
6	2.35	Synchronizer [13]	1	88.35	1.73	1.77
			2	98.74	0.28	2.54
			3	99.65	0.07	3.91
			4	99.92	0.02	5.34
			5	99.97	0.005	6.77
		CORLD-C	2	43.81	4.67	4.67
			3	79.37	2.39	2.39
			4	96.68	0.55	0.55
			5	99.68	0.05	0.05
7	1.81	Synchronizer [13]	1	77.04	2.70	2.47
			2	92.27	1.10	1.53
			3	97.26	0.48	2.03
			4	98.88	0.25	2.72
			5	99.56	0.14	3.48
		CORLD-C	2	46.65	4.90	4.90
			3	58.41	3.97	3.97
			4	81.02	2.05	2.05
			5	91.35	1.09	1.09
8	1.63	Synchronizer [13]	1	79.18	2.52	2.32
			2	94.27	0.95	1.08
			3	98.11	0.4	1.02
			4	99.33	0.2	1.28
			5	99.73	0.11	1.63
		CORLD-C	2	46.39	4.74	4.74
			3	69.77	3.14	3.14
			4	85.20	1.64	1.64
			5	95.57	0.5	0.5

for both minimum and maximum functions and its MAE rates decrease by increasing *FS*.

2) Decorrelator: We evaluate the accuracy of the proposed decorrelator (CORLD-D) compared to the decorrelator design of [13] when connecting correlated pseudo-random and LD bit-streams of 2^N-bit length (N=6, 7, 8) to the inputs of the decorrelator circuit. The accuracy evaluation results are shown in Table IV. An *Input SCC* of 100 means that the input bit-streams are completely correlated. A lower *Output SCC* (closer to 0) means a better independence between output bit-streams. We evaluate the quality of the output bit-streams by measuring the MAE rate of performing SC multiplication (bit-wise `AND`) on the produced bit-streams. The reported MAE is the mean of the measured error rates when multiplying all possible pairs of input values (e.g., 256 × 256 combinations for 8-bit input width). Note that using different random number sequences (e.g., different Sobol sequences) in generating the input bit-streams can result in different MAEs. Hence, in our evaluation, we selected different pairs of Sobol sequences from a large set of more than 1,100 Sobol sequences to generate LD bit-streams. The reported MAEs in Table IV are the averages of the measured values.

The last column of Table IV reports the MAE rate of multiplying two numbers represented by two independent Sobol-based bit-streams as a standard for our comparison. Notice that for 0 percent error, bit-streams of 2^{2N}-bit length must be processed [18] [19] but here we process 2^N-bit bit-streams to have the same precision for both input and output. As can be seen in Table IV, CORLD-D achieves comparable accuracy compared to the independent LD Sobol-based bit-streams for different input widths. In most cases, our decorrelator performs better than the decorrelator design of [13] for both LD and pseudo-random bit-streams. We further evaluated CORLD-D for the case that the inputs are outputs bit-streams from multiplication operation. The output SCC and the MAE rates of the multiplication function are reported in Table II. We can see that in most cases CORLD-D provides a lower output SCC and MAE compared to the SoA decorrelator [13].

B. Cost Comparison

The hardware cost of the proposed correlator and decorrelator is compared with the SoA designs in Table V and Table VI. We synthesized the designs using the Synopsys Design Compiler v2018.06 with the 45nm FreePDK gate library [20]. The proposed designs consist of two parts: a static part which is independent of the number of input bit-streams and so its area is fixed, and a part that its area depends on and increases with the number of inputs. The synchronizer and decorrelator design of [13] have different structures. The minimum *depth* for the decorrelator is 2 while it is 1 for the synchronizer. The hardware area of the decorrelator varies by the input width as the size of the RNGs depends on the number of bits in the input bit-streams. Although the hardware cost of the synchronizer is less than that of CORLD-C, the total cost of using the synchronizer technique depends on the number of input bit-streams. At each stage, new synchronizers are needed as the synchronizer circuit only makes two bit-streams correlated with respect to each other. Therefore, correlating a large number of bit-streams in a SC system with multiple stages of computations needs many synchronizers. We will show this in Section VI with two applications of correlated

TABLE IV: Accuracy Evaluation of the Proposed Decorrelator (CORLD-D) for Pseudo-random and LD Bit-streams of 2^Nbit.

Input Width	Input SCC	Method	Depth/ Fixer Size	Pseudo-random Bit-streams: Output SCC	Pseudo-random Bit-streams: MAE (%)	LD Bit-streams: Output SCC	LD Bit-streams: MAE (%)	MAE (%) Independent Sobol-based Bit-streams
6	100	Decorrelator [13]	2	64.54	3.83	42.43	2.59	0.65
			3	59.46	4.19	30.75	2.31	
			4	63.98	4.30	25.08	2.68	
			5	57.19	4.86	33.04	2.67	
		CORLD-D	2	52.28	3.90	9.92	1.24	
			3	22.39	2.77	1.79	0.90	
			4	8.84	2.33	-2.19	0.79	
			5	25.96	3.09	-11.94	1.84	
7	100	Decorrelator [13]	2	67.13	4.35	39.25	2.36	0.36
			3	56.12	3.48	29.33	1.85	
			4	56.81	3.32	24.87	2.64	
			5	46.23	2.52	23.39	1.66	
		CORLD-D	2	59.55	4.07	12.66	0.87	
			3	49.85	3.12	2.89	0.64	
			4	29.8	2	-0.25	0.47	
			5	32.63	1.7	-2.21	0.57	
8	100	Decorrelator [13]	2	67.67	4.54	39.32	2.49	0.19
			3	54.70	3.46	22.76	1.48	
			4	52.18	3.17	24.36	2.52	
			5	42.49	2.51	21.21	1.25	
		CORLD-D	2	49.14	3.53	13.98	0.69	
			3	35.1	2.25	3.86	0.4	
			4	32.62	1.81	0.45	0.3	
			5	5.44	0.98	-0.49	0.25	

TABLE V: Hardware area (μm^2) of the proposed CORLD-C and CORLD-D for different FSs

	Static Area		# of Input Dependent Area		Total Area For 1 Input		Total Area For 2 Inputs	
FS	-C	-D	-C	-D	-C	-D	-C	-D
2	45	83	170		215	253	386	424
3	67	129	238		306	367	544	605
4	90	175	298		389	474	687	772
5	112	220	354		467	574	822	929

TABLE VI: Hardware area (μm^2) of the synchronizer and decorrelator circuit of [13] for different depths

Depth	Decorrelator [13]			Synchronizer [13]
1	N/A	N/A	N/A	107
2	1225	1589	1969	166
3	1288	1652	2033	168
4	1357	1721	2101	237
5	1393	1757	2137	241
SSG Area	569	751	941	
Input Width	6-bit	7-bit	8-bit	

bit-streams, SC design of sorting and median filtering. Our correlator, on the other hand, is able to correlate all bit-streams by manipulating each one only once. This leads to a significant hardware cost saving compared to the synchronizer technique. The proposed decorrelator is also advantageous from the hardware cost point of view. The hardware cost of the decorrelator design of [13] depends on the area of the Sobol sequence generator (SSG) which is more than that of other types of RNGs. However, selecting SSG as the RNG significantly improves the accuracy.

VI. Case Studies

In this section, we evaluate the performance and the hardware cost of the proposed correlator in the SC design of

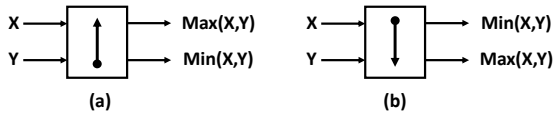

Fig. 10: Schematic representation of a CAS block (a) Ascending order, (b) Descending order

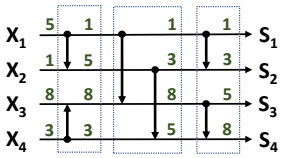

Fig. 11: 4-input Sorting Network

the sorting and median filtering circuits which include both minimum and maximum operations. These circuits consist of multiple stages of computations requiring correlated bit-streams for correct functionality.

A. Sorting

A sorting network is a combination of some compare-and-swap (CAS) blocks that sorts a set of input data [10]. A CAS block comprises a minimum and a maximum operation in ascending or descending order as shown in Fig. 10. A 4-input sorting network made of 6 CAS blocks is shown in Fig. 11. A low-cost SC design for hardware implementation of sorting networks is proposed in [10]. In this design, each CAS block is implemented using one `AND` (for minimum operation) and one `OR` (for maximum operation) gate. For correct and accurate functionality, the input bit-streams to the CAS blocks must be correlated. Connecting uncorrelated bit-streams to the SC sorting circuit and so CAS blocks results in inaccurate and

TABLE VII: Accuracy Evaluation of the Stochastic Sorting System with the Proposed and SoA Correlator Technique when Sorting 256-bit independent LD input bit-streams

	Synchronizer [13]		CORLD-C	
Depth/ FS	8 inputs MAE (%)	16 Inputs MAE (%)	8 Inputs MAE (%)	16 Inputs MAE (%)
1	1.05	2.19	N/A	N/A
2	2.06	3.74	2.25	3.11
3	3.18	5.41	0.69	1.10
4	4.29	7.09	0.16	0.32
5	5.38	8.75	0.03	0.07

TABLE VIII: Hardware Cost (μm^2) Comparison of the SC Sorting and Median Filtering System with different number of inputs utilizing proposed CORLD-C and SoA synchronizer (# of Cor.: # of correlator needed for the design.)

		Synchronizer [13]			CORLD-C			
# of Inputs	# of CAS	# of Cor.	Area Depth=1	Area Depth=2	# of Cor.	Area FS=2	Area FS=3	
8	24	24	2,580	3,998	8	1,408	1,974	Sorting
16	80	80	8,600	13,328	16	2,771	4,850	
32	240	240	25,800	39,984	32	5,498	7,696	
64	672	672	72,240	111,955	64	10,951	15,325	
128	1,792	1,792	192,640	298,547	128	21,856	30,582	
256	4,608	4,608	495,360	767,693	256	43,667	61,098	
9 (3×3)	19	19	2,043	3,165	9	1,579	2,213	Median Filter
25 (5×5)	246	246	26,445	40,984	25	4,305	6,027	

wrong output data. Correlator circuits are therefore needed to manipulate and guarantee correlation between bit-streams.

The overhead cost of using the synchronizer technique [13] depends on, and significantly increases with, the number of CAS blocks. This is because the output bit-streams from different CAS blocks are not necessarily correlated when using this technique. Therefore, each pair of inputs to a CAS block in each stage of computations needs a separate synchronizer. On the other hand, our proposed CORLD-C needs to correlate the input bit-streams only once, in the first stage of computations, where the input bit-streams arrive.

Table VII shows the MAE rates of an 8-input and a 16-input SC sorting system [10] processing data received in the form of independent LD bit-streams. The MAEs are reported for two different approaches for manipulating correlation: 1) using the SoA synchronizer with different *depth*s 2) using the proposed CORLD-C with different *FS*. As can be seen in the reported numbers, the proposed technique outperforms the SoA technique particularly when *FS* increases. Increasing *depth* in the synchronizer technique increases the MAE rates mainly due to the weak performance of the synchronizer technique in performing the maximum operation. Table VIII compares the number of needed correlator units and the hardware cost of the implemented designs. As it can be seen, the hardware area saving provided by CORLD-C increases significantly when the number of inputs increases. For example, for the 256-input SC sorting system, CORLD-C with *FS*=2 reduces the hardware area cost by more than 17$\times$ compared to the synchronizer technique with *Depth*=2.

B. Median Filtering

A median filter is a nonlinear filter which removes impulse noises from images and videos by replacing each pixel value

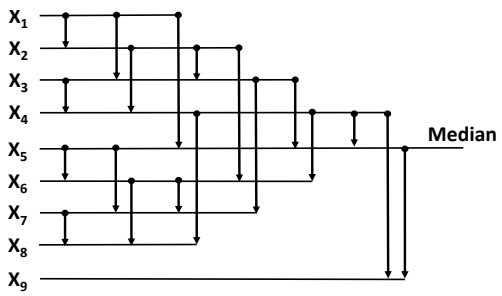

Fig. 12: 3×3 median filter using 19 CAS blocks [10]

TABLE IX: Accuracy Evaluation of SC Median Filtering System with 256-bit indepdendent LD input bit-streams

	Synchronizer [13]		CORLD-C	
Depth/ FS	9 (3×3) inputs MAE (%)	25 (5×5) Inputs MAE (%)	9 inputs MAE (%)	25 Inputs MAE (%)
1	1.09	4.07	N/A	N/A
2	2.05	8.92	2.52	3.61
3	3.25	13.68	0.8	1.36
4	4.43	18.38	0.2	0.44
5	5.59	22.9	0.04	0.11

by the median of all pixel values in the local neighborhood. Fig. 12 shows the structure of a 3$\times$3 median filter circuit made of 19 CAS blocks. The design of a 5$\times$5 median filtering circuit can be found in [10]. For correct functionality, similar to the SC sorting system, the CAS blocks need correlated inputs. Correlation manipulation units are therefore needed to guarantee correlation between the inputs of the CAS blocks. The hardware cost and the accuracy of the SC median filtering circuit implemented with different correlation manipulation approaches are reported in Table VIII and Table IX. As shown, CORLD-C achieves significantly lower MAE rates with considerable saving in the hardware area cost.

VII. Conclusion

In this work, we proposed two in-stream correlator (CORLD-C) and decorrelator (CORLD-D) techniques to increase and decrease correlation between SC bit-streams. The proposed techniques produce high-quality LD bit-streams with no impact on the data values. The accuracy, performance, and hardware cost are all improved compared to the SoA correlation manipulation techniques. We further proposed an FSM-based correlator that improves the accuracy of the SoA SC division circuit. Evaluating the proposed techniques in the SC design of the sorting and median filtering circuits showed a significant reduction in the MAE rates and hardware area cost. The synthesizable Verilog HDL codes of this work including the proposed correlator and decorrelator circuits are made publicly available at *https://github.com/asadisina/CORLD*.

Acknowledgment

This work was supported in part by the Louisiana Board of Regents Support Fund no. LEQSF(2020-23)-RD-A-26, National Science Foundation grants #2019511 and #2127780, Semiconductor Research Corporation (SRC) Task No. 2988.001, and Department of the Navy, Office of Naval Research, grant #N00014-21-1-2225.

REFERENCES

[1] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM, 1967, pp. 149–156.

[2] A. Alaghi, W. Qian, and J. P. Hayes, "The Promise and Challenge of Stochastic Computing," *IEEE Trans. on Computer-Aided Design of Integ. Circ. and Sys.*, vol. 37, no. 8, pp. 1515–1531, Aug 2018.

[3] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," *IEEE Trans. on Comp.*, vol. 60, no. 1, pp. 93–105, Jan 2011.

[4] H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *the 54th DAC, 2017*, 2017, pp. 1–6.

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

[6] D. Wu and J. San Miguel, "In-stream stochastic division and square root via correlation," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[7] 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, 2020.

[8] S. Mitra, D. Banerjee, and M. K. Naskar, "A low latency stochastic square root circuit," in *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*, 2021, pp. 7–12.

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

[10] M. H. Najafi, D. J. Lilja, M. D. Riedel, and K. Bazargan, "Low-Cost Sorting Network Circuits Using Unary Processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 8, pp. 1471–1480, Aug 2018.

[11] M. H. Najafi, P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. Riedel, "A Reconfigurable Architecture with Sequential Logic-Based Stochastic Computing," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, pp. 57:1–57:28, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3060537

[12] S. Gupta, M. Imani, J. Sim, A. Huang, F. Wu, M. H. Najafi, and T. Rosing, "SCRIMP: A General Stochastic Computing Architecture using ReRAM in-Memory Processing," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1598–1601.

[13] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *DATE'18*, 2018, pp. 1417–1422.

[14] S. Asadi, M. H. Najafi, and M. Imani, "A Low-Cost FSM-based Bit-Stream Generator for Low-Discrepancy Stochastic Computing," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 908–913.

[15] S. Liu and J. Han, "Energy Efficient Stochastic Computing with Sobol Sequences," in *DATE'17*, March 2017, pp. 650–653.

[16] A. Alaghi and J. Hayes, "Fast and accurate computation using stochastic circuits," in *DATE'14*, March 2014, pp. 1–4.

[17] S. Asadi and M. H. Najafi, "LDFSM: A Low-Cost Bit-Stream Generator for Low-Discrepancy Stochastic Computing: Late Breaking Results," in *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, ser. DAC '20. IEEE Press, 2020.

[18] 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.

[19] S. Asadi and M. H. Najafi, "Accelerating deterministic stochastic computing with context-aware bit-stream generator," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, New York, NY, USA, 2020, p. 157–162.

[20] "NCSU FreePDK 45nm Library," https://research.ece.ncsu.edu/eda/freepdk/freepdk45/.