Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 547 – 564
© TÜBİTAK
doi:10.3906/elk-1805-195

Accurate and compact stochastic computations by exploiting correlation

Hamdan ABDELLATEF[*], **Mohamed KHALIL-HANI**, **Nasir SHAIKH-HUSIN**
VeCAD Research Laboratory, School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia

Received: 29.05.2018 • **Accepted/Published Online:** 05.09.2018 • **Final Version:** 22.01.2019

Abstract: Recent studies have shown, contrary to what was previously believed, that by exploiting correlation in stochastic computing (SC) designs, more accurate SC circuits with low area cost can be realized. However, if these basic SC circuits or blocks are cascaded in series to form a large complex system, correlation between stochastic numbers (SNs) from one block to the next would be lost; thus, inaccuracies are introduced. In this study, we propose correlating circuits to be used in building complex correlated SC systems. One of the circuits is the correlator that restores lost correlations between two SNs due to previous processing. In addition, a correlated SN generator is introduced to generate SN correlated to a specific SN. Experimental results show that our methods have improved the accuracy of stochastic computation and preserved the stochastic computing correlation without the need for conversion from SC to the conventional binary-encoded computing, and vice versa. Consequently, lower latency and lower area cost are achieved.

Key words: Stochastic computing, correlation, image processing

1. Introduction

Stochastic computing (SC) is a reemerging computing paradigm that was first introduced by Gaines [1] in the 1960s as an alternative to the conventional binary-encoded deterministic computing (DC) technique. In SC, data being processed are represented by random number bit-streams (referred to as stochastic numbers), and the value of the data is encoded as the probability of 1s appearing in the bit-stream. For example, the data bit-stream $X = 1001$ encodes the value of 0.5 since the probability of 1s appearing in X is 0.5 (=2/4; there are two ones and the bit-stream is 4 bits long).

The main advantage of an SC-based design is its low hardware cost and high tolerance for soft errors. Hence, today there is renewed interest in SC for applications in mobile and embedded devices that usually demand error-tolerant solutions with low area and low power. Consequently, in recent years, there has been more active research conducted to adopt SC in a wide range of embedded solutions for image processing [2], neural networks [3], and digital filters [4].

However, SC has significant drawbacks that have to be addressed before it can be viable for application in designing complex practical circuits [5]. One key weakness is that an SC implementation can have a long latency arising from long input bit-streams. Data precision depends on bit-stream length; hence, higher precision requires a longer bit-stream. The second key drawback of SC is due to the fact that, unlike DC computation,

[*]Correspondence: hamdan.abdellatef@gmail.com

SC operations (which are based on random numbers) do not necessarily yield consistent results, giving rise to the issue of accuracy.

Aside from quantization errors, correlation between stochastic numbers (SNs) is also a source of inaccuracy in SC circuits. To operate correctly, some SC circuits require uncorrelated data inputs; others require correlated inputs. Hence, the inaccuracies due to correlation arise because of overcorrelated operands in the former case, and in the latter case, because of operands that are not sufficiently correlated. Research work in [6] has shown that circuits that exploit correlation can result in improved accuracy in SC-based designs. It has also shown that, by exploiting correlation, further gains can be made in area savings and latency reduction.

However, previous works on utilizing correlation in SC circuit design were limited to the design of basic circuits or functional blocks, such as an edge detection filter in [7]. Typically, a large complex system would be a cascade or pipeline of a number of these basic functional blocks. For example, an image processing pipeline would consist of a series of circuits that include a median filter block, a smoothing filter, an edge detector, and, finally, a thresholding stage. Such a system could not be realized previously, because the SC-based functional blocks with correlated input bit-stream produced uncorrelated output bit-streams. Consequently, correlation among SNs between the blocks was reduced or lost, resulting in significant errors. To prevent these errors from occurring, correlation has to be maintained across the complete system, end-to-end. We will refer to such a system whereby correlation is maintained as a correlated stochastic computing (CSC) system.

One may think that there is an on-the-fly solution. The designer simply inserts conversion circuits whenever inputs have to be correlated to restore any lost correlation. However, this solution is infeasible since it introduces long conversion latency and significantly increases area cost (or resource utilization). This paper proposes a methodology to build a design in which correlation across a CSC system is maintained without utilizing multiple conversion circuits. Our contributions are as follows: 1) An SN correlator that restores the lost correlation between SNs and eliminates the need for conversion circuits between the functional blocks. It utilizes a novel SC-based comparator. 2) A correlated stochastic number generator (CSNG) that can generate an SN that is correlated to a specific SN, allowing both to be employed in a circuit that exploits correlation. 3) A design methodology for building a more compact, accurate, low-latency CSC system that utilizes SC circuit blocks that exploit correlation. 4) This work classifies the SC processing elements concerning their effect on correlation. This is an extension of the study on correlation sensitivity classification introduced in [6].

The rest of this paper is organized as follows. Section 2 reviews related previous works and presents the basics of SC. It then addresses the correlation problem. Section 3 presents the SC correlator and the CSNG. Section 4 discusses the experimental work done and results obtained to demonstrate the effectiveness of the proposed methods in terms of accuracy. Section 5 concludes the paper.

2. Stochastic computing

2.1. Stochastic computing basics

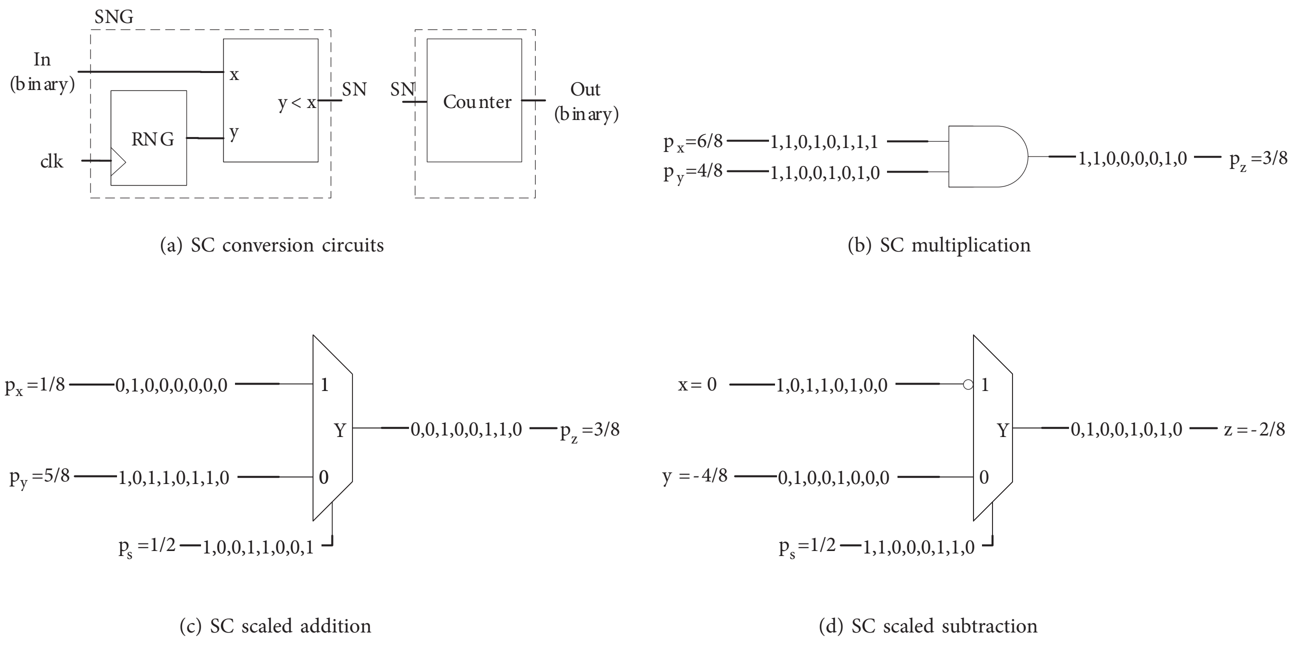
Stochastic computing is a computing paradigm that represents and processes information in the form of digitized probabilities. In SC, an SN is a binary bit stream with the probability p of 1s [5]. An SN has neither fixed length nor structure; SNs 1001, 1010, and 10100101 all encode the value of $\frac{1}{2}$ since probability p of 1s appearing in the SN is $\frac{1}{2}$. SNs either use unipolar (UP) or bipolar (BP) encoding. Bipolar encoding allows negative values while unipolar does not. Table 1 defines the UP and BP encoding. In the table, N_0, N_1, and N represent the number of zeros, number of ones, and total number of bits in the SN, respectively. Examples of SNs are

SN1 = 11001111 and SN2 = 10100110, with UP values $p_1 = \frac{6}{8}$ and $p_2 = \frac{4}{8}$. The corresponding BP values are $x_1 = \frac{6-2}{8} = \frac{4}{8}$, which satisfies $x_1 = 2 \times p_1 - 1$ and $x_2 = 0$.

Table 1. SN encoding.

Encoding type	Value	Interval	Relation
UP	$p = \frac{N_1}{N}$	[0,1]	$p = \frac{1+x}{2}$
BP	$x = \frac{N_1 - N_0}{N}$	[-1,1]	$x = 2p - 1$

To convert from binary to stochastic, a stochastic number generator (SNG) is used. The SNG comprises a random number generator (RNG) and a comparator. To produce correlated SNs, one RNG is shared among the SNGs. On the other hand, to convert from stochastic to binary, a counter is used. Figure 1a gives the block diagrams of these conversion circuits.



(a) SC conversion circuits (b) SC multiplication

(c) SC scaled addition (d) SC scaled subtraction

Figure 1. SC basic circuits.

A major advantage of SC is that it employs very low-complexity arithmetic units [5]. The AND gate in Figure 1b performs the multiplication of two uncorrelated UP-encoded SNs, X and Y, with probabilities $p_x = \frac{6}{8}$ and $p_y = \frac{4}{8}$. The output Z has probability $p_z = \frac{3}{8}$ as expected. If the independence (uncorrelated) condition between the inputs is not met, the accuracy of the output is degraded. As a result, Gaines [1] stated that SNs should be independent. If the input SNs are BP-encoded then SC multiplication is performed using an XNOR gate.

It is clear that when UP-encoded SNs are applied, the values are constrained between 0 and 1. Arithmetic operations such as addition or subtraction may produce a probability that is under 0 or exceeds 1. Thus, there is no direct addition in SC, but instead scaled addition is used. A 2-to-1 multiplexer (MUX) performs scaled addition with probability of the selector bit-stream $p_s = \frac{1}{2}$. The MUX operates according to $p_z =$

$p_s p_x + (1 - p_s)p_y$ where the value of the selector acts as a weight for the inputs. For the case $p_s = \frac{1}{2}$, $p_z = \frac{1}{2}(p_x + p_y)$. In Figure 1c the MUX acts as a scaled adder where $p_z = \frac{1}{2}(p_x + p_y) = \frac{3}{8}$. It should be noted that the MUX selector should be uncorrelated with the MUX inputs to prevent correlation-induced error. However, the MUX inputs are correlation-insensitive and can have any value of correlation. To perform subtraction, a NOT gate is used to negate the subtracted value before it is added to the other input by 2-to-1 MUX as shown in Figure 1d. In the subtraction example, BP encoding is used to represent the negative value. The stochastic number X is subtracted from Y. Thus, the BP output value z should be $\frac{1}{2}(y-x) = -\frac{2}{8}$, which is the case as $z = \frac{N_1 - N_0}{N} = -\frac{2}{8}$.

According to [8], SNs should be independent and uncorrelated bit-streams. The independence definition in probability theory is $P(\cap_{i\epsilon S} A_i) = \prod_{i\epsilon S} P(A_i)$. The N events A_1, A_2, ..., and A_N are said to be independent if the probability of their intersection equals the multiplications of their probabilities for every subset S of $\{1, 2, ..., N\}$ [9]. However, a recent study [6] stated that unlike what was formerly believed, the correlation can serve as a resource in designing stochastic circuits. For edge detection, after taking advantage of correlation, the circuit area was reduced $2.1\times$ and $2.57\times$ compared to independent inputs SC and conventional binary DC, respectively [10]. Also, the latency is reduced by $3\times$ compared to independent inputs SC.

2.2. Related work

The main advantage of SC is the cheap hardware footprint. Many applications such as digital filters, image processing, and neural networks have been implemented successfully in SC. These applications share the fact that they possess a high error tolerance and require a massive number of low-precision operations [11]. However, there are still limitations in SC, especially in accuracy, latency, and circuit size. In our work, we emphasize the accuracy of SC and the hardware footprint.

Although SC's benefit is the low area cost, the conversion circuits can take up to 80% of the overall area, particularly when independent RNGs are needed for SN generations [12]. Some works proposed sharing schemes to share one RNG with different SNGs [4, 13–15]. For example, if k different SNGs share only one RNG as in the case in the circular shifting in [4], the area cost of the RNGs is decreased by $1/k$. On the other hand, if the circuit is correlation-insensitive or requires correlated inputs, all SNGs could share one RNG, which also reduces the cost of conversion circuits.

Many research works have been conducted to increase the accuracy of SC circuits. There are three main methods in the literature to improve SC accuracy. The first approach is exploiting SC's progressive precision property by using other types of RNGs instead of the linear feedback shift register (LFSR) [16–18]. This approach decreases the random fluctuations and enables the designer to use part of the SNs to obtain acceptable results that improve the latency. Secondly, accuracy is improved by using circuits with correlated inputs (CSC) [6, 19], which reduce the area cost. Thirdly, accuracy can be enhanced by performing SC operations using deterministic bit-streams [20, 21]. These works obtained accurate arithmetic operations, but the entire bit-stream should be processed to avoid significant truncation error [22]. The three preceding approaches hold the same drawback in that they are not appropriate for multilevel designs [23]. If SC functional blocks are cascaded, outputs of the first block cannot be directly applied as an input of the second block. However, our work solves this problem for CSC circuits by proposing a methodology to create a multilevel CSC system.

Other works can be found in the literature addressing SC accuracy. The work in [24] dealt with random fluctuation errors by eliminating the constants by introducing memory in the SC circuits. To decorrelate the

SN to obtain more accurate results, [25] proposed the isolation-based decorrelator. Accurate SC operations were proposed in [23, 26]. These works obtained accurate results, but they introduced more latency as they increased the output bit-stream length. Also, a scaling-free accurate SC adder was proposed in [14]. The main selling point of SC is the cheap hardware footprint. Although these works are cheaper than conventional DC, CSC circuits are much cheaper with acceptable accuracy.

The accuracy in SC is affected by multiple sources. The sources of these inaccuracies are the low precision, random number fluctuations, conversion error, and correlation-induced inaccuracies [19]. In this work, we concentrate on the correlation-induced inaccuracies. The other sources of inaccuracies are mostly removed by using uniform RNG and the SN length $L = 2^n$ where n is the conventional DC precision.

2.3. Stochastic computing correlation problem

Alaghi and Hayes [6] introduced a parameter that determines the significance of the correlation between two SNs called stochastic computing correlation (SCC), as shown in Eq. (1). Also, they argued that correlation can be exploited to obtain more efficient circuits. They proposed an XOR gate for UP absolute subtraction. Then they categorized the basic processing elements according to their correlation sensitivity. Our work extends the categorization further to show that some elements have an effect on correlation and certain methods are proposed to allow cascading circuits to utilize correlation or create complex CSC circuits without accuracy loss, although the correlation will be lost.

$$SCC(X,Y) = \begin{cases} \frac{p_{X\cap Y} - p_X p_Y}{min(p_X, p_Y) - p_X p_Y} & p_{X\cap Y} - p_X p_Y > 0 \\ 0 & p_{X\cap Y} - p_X p_Y = 0 \\ \frac{p_{X\cap Y} - p_X p_Y}{p_X p_Y - max(p_X + p_Y - 1, 0)} & p_{X\cap Y} - p_X p_Y < 0 \end{cases} \tag{1}$$

If we want to realize the equation $p_Z = \frac{1}{2} max(p_A + p_B, p_C + p_D)$ using correlated inputs, generated using the same RNG, we would use 2-to-1 MUX to perform scaled addition. Then an OR gate is used to perform the max operation. The OR gate is SCC-sensitive; hence, when SCC varies, its functionality changes. For example, if SCC = 0, $p_{OR,SCC=0} = p_1 + p_2 - p_1 p_2$. However, if SCC = 1, $p_{OR,SCC=1} = max(p_1, p_2)$. Figure 2 shows the circuit that computes p_z. The four inputs A, B, C, and D are correlated. After the scaled addition operation, the correlation is lost. As a result, the OR operation does not perform the max operation, obtaining a wrong result. Thus, when a designer wants to create a CSC circuit, the SCC variation due to processing elements should be taken into consideration. In Table 2, the sensitivity to correlation and variation of correlation for basic SC processing elements are indicated.

If SC bit-stream length $L = 256$ bits are used, the mean absolute error generated due to the correlation loss is 5.45% in the example shown in Figure 2. The mean SCC of OR gate inputs is 0.75 in 10,000 iterations of randomly generated inputs. In this experiment, other inaccuracy sources such as conversion and correlation with MUX selector are almost eliminated by using uniform RNG and independent RNGs for the selector.

From previous discussion, to design a CSC block that utilizes correlation, the following order of processing elements should be used:

1. The processing elements that do not vary the correlation.

2. The processing elements that vary correlation but are correlation-sensitive.

3. The correlation-insensitive processing elements.

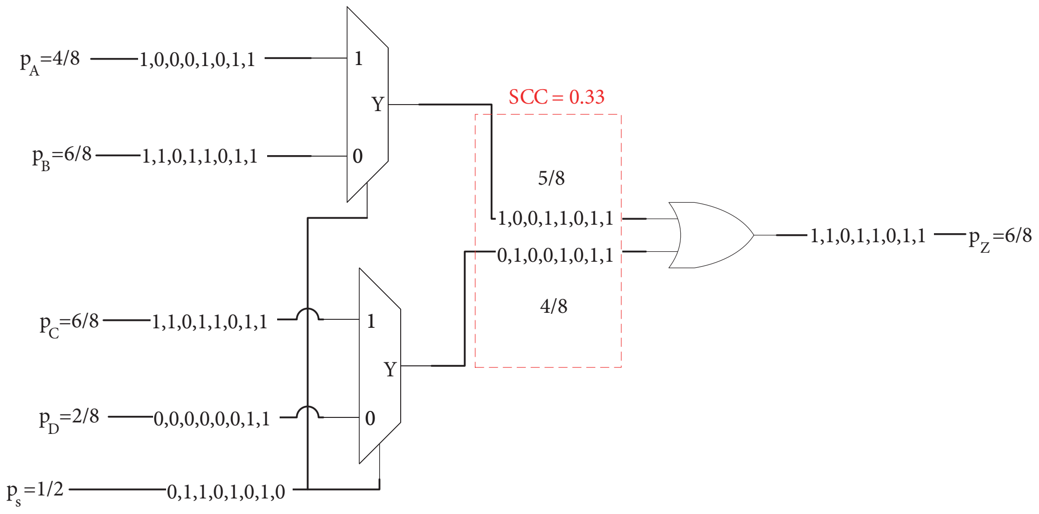


Figure 2. Correlation loss problem.

Table 2. Correlation sensitivity and variation.

SC element	Correlation sensitivity	Correlation variation
NOT	no	yes
AND	yes	no
OR	yes	no
XOR	yes	yes
MUX	inputs: no selector: yes	yes
FF	no	yes

Following the above sequence is the best for designing a block, but sometimes it cannot be done to achieve the required functionality. The designer thus achieves less accuracy due to correlation loss. Therefore, it is essential to create SC circuits that maintain the correlation throughout the system to obtain more efficient SC systems in terms of area and latency.

3. Correlating stochastic numbers method

To enable designing a more complex CSC circuit composed of multiple blocks that exploit correlation, a method to correlate SNs is proposed. All components of the circuitry should preserve correlation. Also, they should not introduce high latency or hardware footprint. It should be noted that the CSC circuit will use only one RNG to minimize area cost. Also, using one RNG will avoid one of the drawbacks mentioned by Hayes [12], which is conversion circuit area cost. In the case that requires an uncorrelated random number sequence such as generating MUX selector SN, the circular shift RNG sharing scheme is used as proposed in [13].

3.1. The correlator

As discussed in Section 2.3, SC requires a method to restore any lost correlation. Suppose we have two SNs, X and Y. According to Eq. 1, if we wish to obtain $SCC(X,Y) = 1$, the probability of $X \cap Y$ should be equal to the minimum of the probabilities of X and Y, i.e. $p_{X \cap Y} = min(p_X, p_Y)$, so that the numerator and denominator of Eq. 1 are equal.

Since X and Y are Bernoulli sequences, $p_{X\cap Y} = min(p_X, p_Y)$ cannot be attained unless, in all positions for 1s in the minimum SN, the maximum SN also has 1 in the corresponding position. Otherwise, the SNs would not be correlated. Figure 1 illustrates the relationship between 1's position and SCC where $X > Y$. Figure 1a shows two correlated SNs while Figure 1b illustrates how SCC is lost due to one position misalignment. We can see that a circuit varies the correlation if the processing elements have results where 1's position changes among SNs.

To recorrelate two uncorrelated SNs like in Figure 1c, two tasks should be performed. First, determine the minimum between the two SNs. Then change the positions of the minimum SN ones to be in the positions of maximum SN ones. Thus, a min relocate algorithm is required. It is important to note that in this work we want to recorrelate; in other words, the SNs to be correlated are a result of an operation of correlated inputs, which are generated initially from the same RNG, as in the example in Figure 2. In that figure, the two SNs that we desire to recorrelate are the outputs of the MUXs. Those SNs still have a high correlation, $SCC = 0.75$, if SC bit-stream length $L = 256$ bits are used.

To find the min SN, we propose a novel SC comparator. If we have two correlated SNs, the greater SN would have '1' in certain positions while the lower number has '0'. To find the smaller value we should determine the position where X and Y are '1' and '0', respectively. This can be obtained by XORing X and Y, so if $X_i \oplus Y_i = 1$ then we can find which is greater between X and Y by examining values at position i. The comparator output should be '1' if one number is greater than the other or '0' otherwise. Thus, the second stage in the comparator is to evaluate $X_i\bar{Y}_i$ to determine which input is smaller. The algorithm for this comparator is shown in Algorithm 1, which will be combined with the relocate algorithm to obtain the correlator.

Algorithm 1 The min operation.

```
1:  for $i = 1 : L$ do
2:      if $X_i \oplus Y_i$ then
3:          $idx = i$
4:          if $X_i\bar{Y}_i$ then
5:              Y is min
6:              break
7:          else
8:              X is min
9:              break
10:         end if
11:     end if
12: end for
```

Since we have two highly correlated SNs, the minimum can be determined when it has '0' where the other SN has '1'. The relocation of minimum ones is performed using a counter (CTR). This is a specially designed counter that increments only when increment signal is '1' and decrements only if the decrement signal is '1'; otherwise, it remains in the same state. At a certain position, if max SN is '0' and min SN is '1', the min SN is changed to '0' and the counter increments by 1. On the other hand, at other specific positions, if max SN is '1', min SN is '0', and $CTR > 0$ (grt0), the min SN will be changed to '1'. As a result, we relocate the min SN '1' to a position where max SN is '1'. Therefore, the two SNs become completely correlated. The relocate algorithm is shown in Algorithm 2. The relocate circuit is shown in Figure 4a. The correlator is obtained by combining Algorithm 1 and Algorithm 2. We notice that the algorithm accuracy depends on the min operation to find the correct bit position to ensure that the relocation part is 100% accurate.

Algorithm 2 Relocate algorithm.

```
1:  for $i = idx : L$ do
2:      if $min_i = 1$ then
3:          if $max_i = 0$ then
4:              $min_i = 0$
5:              $CTR = CTR + 1$
6:          end if
7:      else
8:          if $max_i = 1$ and $CTR > 0$ then
9:              $min_i = 1$
10:             $CTR = CTR - 1$
11:         end if
12:     end if
13: end for
```

To demonstrate the correlator, we use the example in Figure 2. If we use the correlator for the MUX outputs, the absolute error will be decreased by 3.2% from 5.45% to 2.25%. The resource utilization of the correlator when using a 4-bit precision counter is just 5 FFs and 7 LUTs when ZYNQ ZC706 FPGA is used. The 4-bit counter is more than enough for this example for a very long bit-stream since the SNs have sufficient amounts of correlation.

There should be a systematic way to choose the bit-width of the correlator counter. The bit-width of the correlator counter is of $log_2(N_r)$ where N_r is the number of ones in the minimum bit-stream that should be relocated. Suppose we have two bit-streams A and B with probabilities a and b, respectively. If the SN length is L, then $N_r = (min(a,b) - p_{A\cap B}) \times L$, which is equivalent to Eq. 2 where N_A, N_B, and $N_{A\cap B}$ are the number of ones of A, B, and $A \cap B$ bit-streams, respectively. From Eq. 1, taking into consideration that $SCC > 0$, we get Eq. 3. Substituting Eq. 3 in Eq. 2, we get Eq. 4. To find the maximum value for N_r, we first set $SCC = 0$, which corresponds to the worst case scenario, i.e. A and B are uncorrelated. Also, L is fixed to a certain value per iteration. Then N_A and N_B are varied from 0 to L. For all values of L the maximum is $N_r = L/4$ if $SCC = 0$. Thus, the maximum counter bit-width (BW_{ctr}) is given in Eq. 5. However, the counter bit-width is dependent on the SCC value estimate, since the term $log_2(1 - SCC)$ is negative if $SCC > 0$. For the previous example where $SCC = 0.75$ and $L = 256$, $BW_{ctr} = 4$ according to Eq. 5.

$$N_r = min(N_A, N_B) - N_{A\cap B} \tag{2}$$

$$N_{A\cap B} = min(N_A, N_B) \times SCC + \frac{N_A N_B}{L}(1 - SCC) \tag{3}$$

$$N_r = (1 - SCC)(min(N_A, N_B) - \frac{N_A N_B}{L}) \tag{4}$$

$$BW_{ctr} = log_2(1 - SCC) + log_2(L) - 2 \tag{5}$$

3.2. Generating correlated stochastic number

Many algorithms require parameters to be processed with the intermediate data that have been processed earlier. In SC, if we generate those parameters using the initial SNGs, the correlation will be low. Hence, CSNG is required to generate a correlated SN to a specific SN. Algorithm 3 describes CSNG operation. Different from correlating two SNs, CSNG does not need to find the max or min SN. The CSNG is a special local down-counter (LCTR) with initial state defining the number of 1s (N_1) to be generated. This LCTR decrements if decrement signal is '1'; otherwise, it does not change state. Suppose X is the SN to be generated using the CSNG with UP probability p_X where X is to be correlated with Y. Then $N_1 = p_X \times L$, where L is the SN length. From discussion of Figure 3, CSNG generates '1' when Y is '1'. However, if $X > Y$, the number of 1s in X is greater than that of Y. Therefore, if the remaining number of X 1s to be generated is equal to the remaining bits in the bit-stream $L - i$, all the remaining bits in X bit-stream will be 1s. The CSNG always generates correlated bit-streams. The global counter is a down counter that counts the bit-stream length L.

Algorithm 3 CSNG.

```
1:  $N_1 = p_X L$
2:  for $i = 1 : L$ do
3:      if $N_1 = 0$ then
4:          $X_i = 0$
5:      else if $N_1 = L - (i-1)$ then                ▷ useful when $X > Y$
6:          $X_i = 1$
7:          $N_1 - 1$
8:      else if $Y_i = 1$ then
9:          $X_i = 1$
10:         $N_1 - 1$
11:     else
12:         $X_i = 0$
13:     end if
14: end for
```

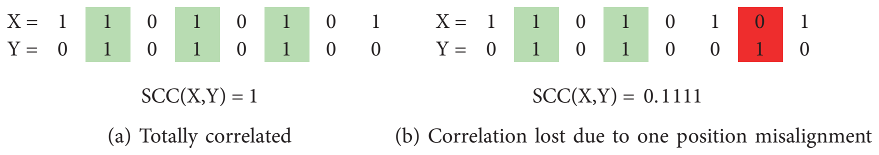
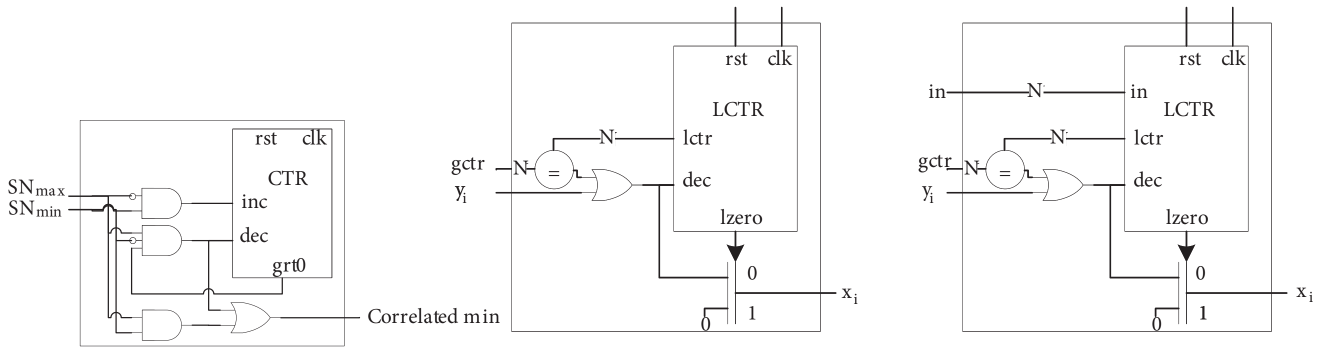


Figure 3. SCC and relation to 1's position.

We propose two types of CSNG. The first one is the constant CSNG that always generates an SN with a fixed probability, as shown in Figure 4b. Resource utilization for this CSNG is 9 FFs and 13 LUTs for 256-bit SN. On the other hand, the variable CSNG is shown in Figure 4c. This CSNG generates SN with the specified probability, and its hardware cost is 25 FFs and 43 LUTs for 256-bit SN. However, more resources can be saved if the counter bit width is reduced, which entirely depends on the values to be generated. The type of CSNG to be used depends on the application.

3.3. Correlated stochastic computing system design methodology

To develop a complete CSC system, first the functional blocks should be designed exploiting the correlation. In the literature, many works proposed blocks that utilize correlation, such as image processing [7, 10, 27] and digital filters [4]. In these previous works, for example, the SC image filter is proposed, but no multiple image processing CSC blocks are tested together due to the correlation loss. In this work we propose a methodology to design a complete CSC system using SC circuits exploiting correlation. The CSC design is more efficient



(a) The relocate circuit used in the correlator

(b) Constant CSNG

(c) Variable CSNG

Figure 4. The correlating circuits.

than SC with independent input SNs in terms of accuracy and area cost. Also, the correlating circuits can be considered as the counter parts of the randomizers in conventional SC implementations.

The input SNs to CSC system are generated using SNGs sharing one RNG, so they are correlated. There are three factors to be considered in this methodology: the SCC, correlation-sensitivity, and correlation-variation. The methodology to build a complete CSC system is shown in Table 3. If the block is first in the system, SCC would be 1, so the input SNs will be directly applied to the current block. Similarly, if the previous block does not change the SCC, the input SNs would be directly applied to the current block. Also, if the current block is correlation-insensitive, such as MUXs, the designer would not need any correlating circuits. On the other hand, in the case of a block sensitive to correlation preceded by a block that changes correlation, the correlator should be applied to the inputs before the current block. Finally, if an SN of a certain probability has to be generated, the CSNG should be used to produce a correlated SN.

Table 3. CSC design methodology.

Previous block	Current block	Action
-	Any	-
No variation in SCC	Any	-
Variation in SCC	Insensitive to SCC	-
Variation in SCC	Sensitive to SCC	Apply correlator to the inputs
Any	Certain probability required to be generated	Generate SN using CSNG

First, the sequence mentioned in Section 2.3 to design a CSC functional block should be used to design the basic blocks. Then, using the proposed methodology to obtain a CSC system, a more accurate and compact system will be obtained compared to conventional SC circuits and systems. For the conversion circuitry, one SNG stage is required at the start where all SNGs share the same RNG, and one counter stage is required at the end of the stochastic computations to convert the result into DC format. However, sometimes uncorrelated bit-streams are required, such as the MUX selector SN. In this case the circular shifting RNG sharing scheme should be used as proposed in [13]. Therefore, a single RNG is enough for the complete CSC system.

4. Experimental results

To evaluate the effectiveness of the proposed correlator or CSNG, they are tested in an image processing system composed of multiple filters. The accuracy of the CSC filters and circuit area are the measured metrics. To evaluate accuracy, the absolute error is computed. The applied SN length L is 256 bits. To assess the circuit area of the correlator and CSNG designs, we synthesize the correlator and the CSNG circuits using the Vivado design suite targeting Xilinx ZYNQ Z706.

The intention of this work is to build a complete CSC system that is composed of multiple functional blocks. Suppose we have the system shown in Figure 5. The system receives a DC image as input and outputs the edges of the input image. The emphasis is not on the image processing algorithm, but rather the ability to design a complete system in stochastic computing maintaining the correlation to obtain accurate results. Before starting the creation of the CSC circuit for the entire case study, the design of each function block is elaborated. The blocks are cascaded according to the proposed methodology. Before applying the inputs to the CSC circuits, SNGs are required to generate the input SNs. The SNGs share one RNG to generate correlated SNs.

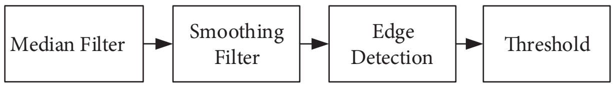


Figure 5. Image processing case study in testing the effectiveness of the proposed correlator and CSNG.

The complete experimental setup is shown in Figure 6. The input is in conventional DC, and the SNs representing the input pixel values are generated by SNGs that share one RNG to obtain complete correlation. At the output, a specially designed binary counter is utilized. Since the output after the threshold will be a binary image, this binary counter will output 0 if all SN bits are zero; otherwise, the output will be 1. The

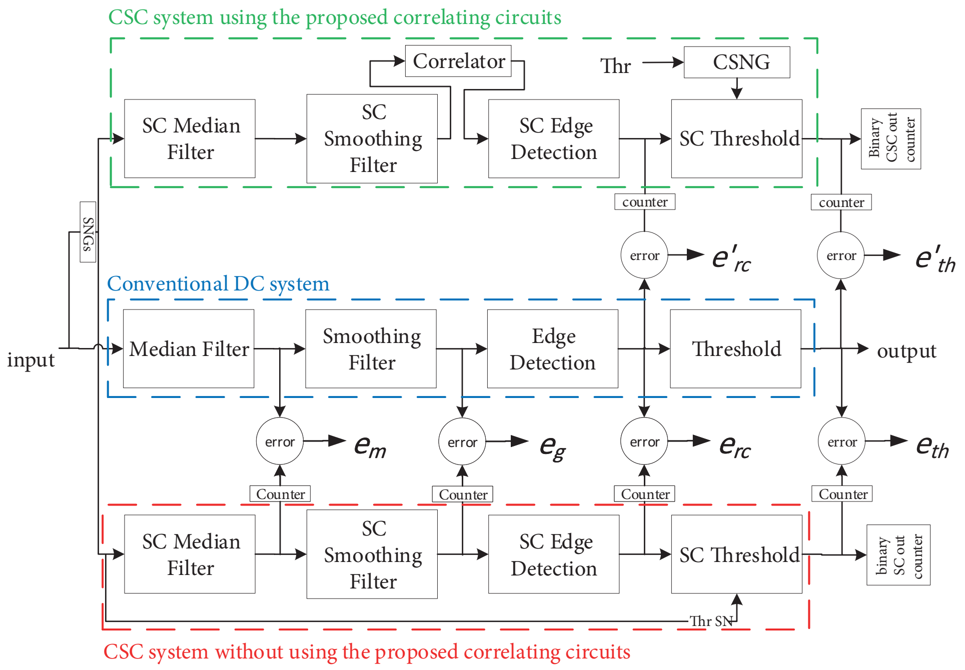


Figure 6. Experimental setup for examining the effectiveness of correlator and CSNG proposed circuits.

experimental setup contains three similar systems. In the middle system in Figure 6, the DC image processing system is used to serve as a reference. The upper system in the same figure is a CSC system that uses the methodology we proposed in this work to maintain the correlation. In this work, a complete CSC system is realized by utilizing the proposed correlator and CSNG. The third system at the bottom of Figure 6 shows the same CSC system without using the proposed correlator and CSNG. To compute the accuracy of the SC circuits, the absolute error as specified by Eq. 6 is computed, where x is the output image pixel value from the SC or DC filter, and R and C are the dimensions of the image. As shown in Figure 6, the error is calculated after converting the SC filter output to DC using a counter.

$$e = \frac{1}{RC}\sum_{i=0}^{R}\sum_{j=0}^{C} \mid x_{i,j}^{(SC)} - x_{i,j}^{(DC)} \mid \tag{6}$$

Applying a good quality image to the system would not demonstrate completely the effectiveness of either SC or the methodology proposed in this work. Before the image is supplied to the system, salt-and-pepper and Gaussian noises are added to erode the quality of the image. Those two noises require the median and smoothing filters to remove them. The original image and the noisy input image are shown in Figures 7a and 7b, respectively.

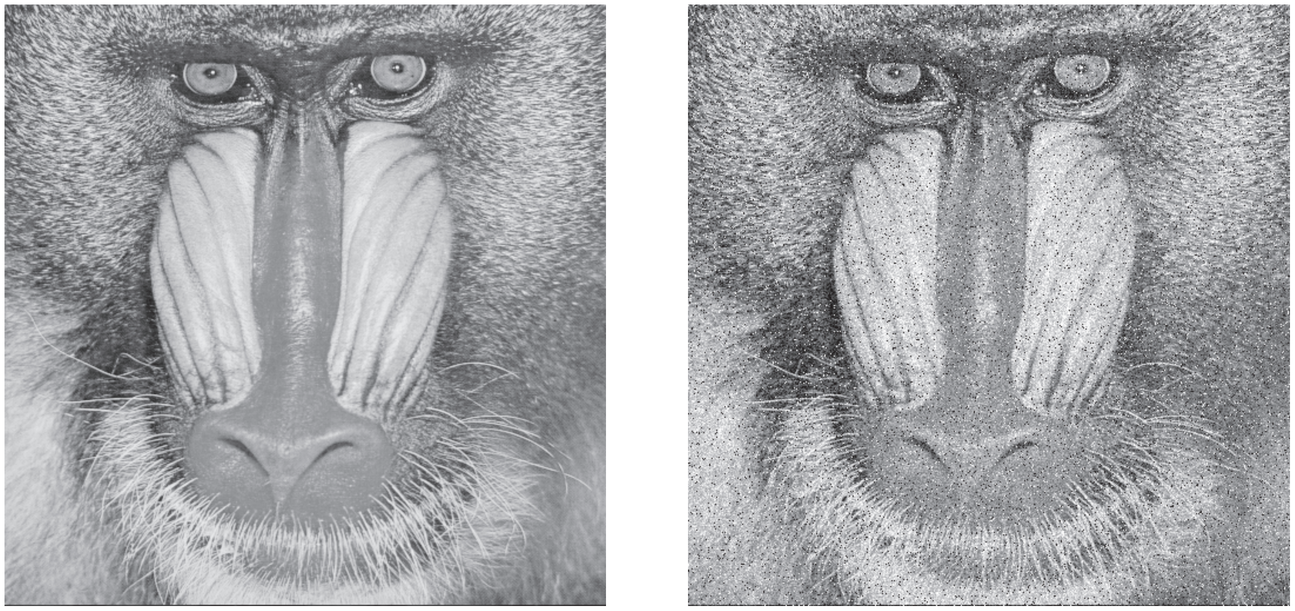
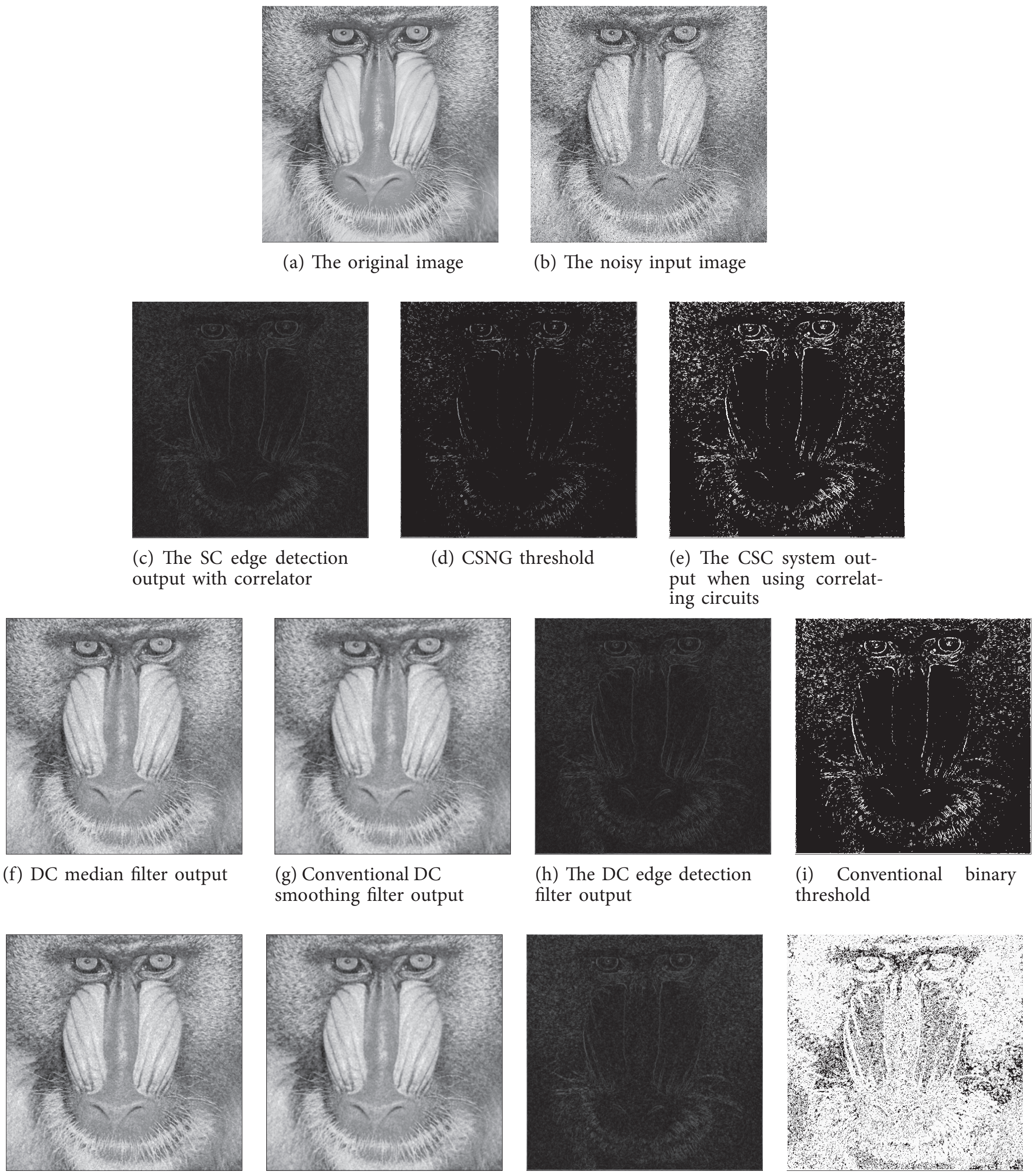

(a) The original image (b) The noisy input image

Figure 7. The experimental setup input image.

The median filter is usually used in image processing to remove salt-and-pepper noise. The SC median filter used in this study was previously proposed in [2] based on the sorting network shown in Figure 8a. However, the SC median filter, by exploiting correlation in the sorting unit, could be more efficient in terms of accuracy, area, and delay. The CSC median filter sorting unit was proposed in [10, 27], which consists of only AND and OR gates as shown in Figure 8b. The CSC median filter has 9.56× and 4.68× area savings with respect to the previous SC median filter [2] and DC median filter, respectively [10]. Since the median filter is the first block, the input SNs are correlated. The CSC median filter is composed of elements that do not vary



(a) The original image

(b) The noisy input image

(c) The SC edge detection output with correlator

(d) CSNG threshold

(e) The CSC system output when using correlating circuits

(f) DC median filter output

(g) Conventional DC smoothing filter output

(h) The DC edge detection filter output

(i) Conventional binary threshold

(j) CSC median filter output

(k) SC smoothing filter output

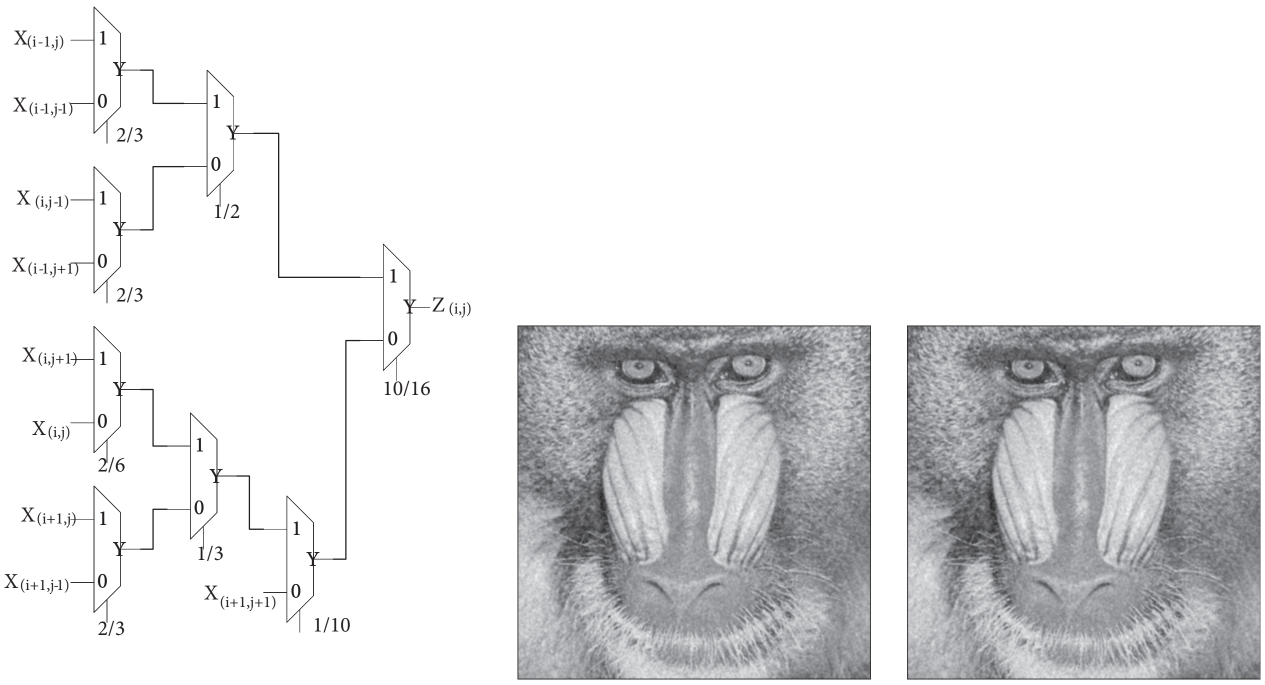
(l) The SC edge detection filter output without correlator

(m) Thres hold output when threshold SN is obtained from the initial SNG

Figure 8. The CSC median filter circuit and the median filter results of the experimental setup.

the correlation. Thus, the output SNs of this filter are correlated, and the correlator is not required after this block. The output images of the SC and DC median filters in the experimental setup of Figure 6 are shown in Figures 8c and 8d, respectively. The absolute error of the median filter e_m is zero since the two outputs are exactly similar. This accuracy is achieved since the processing involves only correlated SNs; hence, the median filter does not suffer from random or correlation-induced errors. In addition, there is no rounding error since the DC image pixel values are 8 bits and the SN length is $L = 2^8 = 256$.

The second filter is the Gaussian smoothing filter to remove the Gaussian noise in the image. We used the 3×3 SC Gaussian filter shown in Figure 9a based on the SC digital filter design methodology presented in [4]. In that work, different orders of the SC digital filters were examined where the hardware footprint was reduced $5\times$ to $7.8\times$ compared to DC filters. The output of this filter is the 2D convolution of the input image with the Gaussian kernel. The selector values are derived from the kernel values. It should be noted that the selector SNs are uncorrelated with the input SNs. Also, if the output of one MUX becomes the input for another, the two MUX selector SNs should be uncorrelated. The uncorrelated SNs are generated by a RNG sharing scheme. The sharing scheme used is the circular shift that creates a new random number sequence to be fed to the SNG comparator without any hardware footprint. The outputs of the SC and DC Gaussian filters are shown in Figures 9b and 9c, respectively. The absolute error of our SC Gaussian filter e_g is 0.0065. The output SNs are not correlated, since the MUX varies the correlation. Therefore, if the following CSC block is correlation-sensitive, a correlator must be used, which is the case in the example system.



(a) SC Gaussianfilter

(b) SC smoothing filter output

(c) Conventional DC smoothing filter output

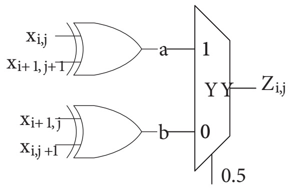
Figure 9. SC Gaussian filter circuit and the Gaussian filter results in the experimental setup.

The first design of an image processing algorithm in stochastic computing exploiting correlation was Robert–Cross (RC) edge detection by Alaghi et al. [7], shown in Figure 10a. The RC edge detection algorithm, shown in Eq. 7, was implemented previously in SC by Li et al. [2]. The saving of area-delay factor was $470\times$ when the correlations were exploited. Later works improved the implementation of Li et al. [2] and used correlation to obtain more efficient implementations in SC for image processing algorithms.

$$z_{i,j} = \frac{1}{2} \times (\mid x_{i,j} - x_{i+1,j+1} \mid + \mid x_{i,j+1} - x_{i+1,j} \mid) \tag{7}$$

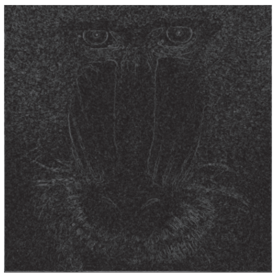
According to our classification in Table 2, the XOR gate that performs absolute subtraction with correlated inputs varies the correlation causing the SNs $SCC(a,b) \neq 1$. However, the MUX is correlation-insensitive to the inputs, and it performs the scaled addition for any value of SCC between the inputs with the condition that they are uncorrelated with the selector. Therefore, the output $z_{i,j}$ fulfills Eq. 7. The design of the SC RC block follows the arrangement we introduced. To evaluate the effectiveness of the correlator, we compare the edge detection output with and without using the correlator, as shown in Figure 6. Since the image intensity values are specified in 8 bits, the precision of the correlator is 6 bits only based on Eq. 5, where the worst case scenario is assumed, i.e. the SNs are independent.

The output images from CSC edge detection with and without correlator and DC edge detection are shown in Figures 10b, 10c, and 10d, respectively. As can be observed, it is clear that the edge detection CSC output using the correlator is more similar to the DC image. The absolute error of the SC edge detection when using correlator e'_{rc} is 0.0096 while that without correlator e_{rc} is 0.0201. The error is reduced by more than $2\times$, and the error is reduced further if the complexity of the circuitry is increased by including the correlator. The improvement in accuracy is obtained at the expense of a small increase in area cost due to the correlator. The correlator resource utilization is shown in Table 4. However, if conversion circuits were used to recorrelate SNs, more hardware footprint would be used, and higher latency by SN length L clock cycles would be introduced. On the other hand, the correlator increases the overall system latency by one clock cycle.



(a) SC Robert-Cross edge detection [7]

(b) The SC edge detection output with correlator


(c) The SC edge detection output without correlator

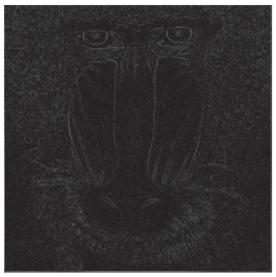

(d) The DC edge detection output

Figure 10. The CSC edge detection circuit and the edge detection results.

The threshold is an image processing point operation that generates a binary image of ‘1’ when the image is greater than the threshold. To do this operation, a comparator is used. If the input SNs are independent, the SC comparator would be very complex. However, the threshold operation can be performed in CSC easily. The novel SC comparator proposed in this work is used. The comparator outputs zeros if $x < y$; otherwise, the comparator outputs zeros until $X_i\bar{Y_i} = 1$, after which the output will be ones. It should be noted that the threshold SN should be totally correlated with the input SNs. Thus, the CSNG is essential to generate the threshold SN correlated with the input SN.

Suppose we want a threshold of 0.1 for the output of the RC edge detection. The DC threshold output is shown in Figure 11a, which serves as the reference, as discussed earlier. Also, Figure 11a is the final output of the DC system of the experimental setup shown in Figure 6. When CSNG is used to generate the threshold SN, the output of the CSC threshold block is as shown in Figure 11b. As can be observed, the CSC threshold output is similar to the DC output, but the edges is not that bright. The output image of the CSC threshold block when using CSNG is not a binary image. This issue is resolved by utilizing the special binary counter to be described later in this section. The error of the CSC threshold operation when using CSNG is $e'_{th} = 0.0507$.

Table 4. Resource utilization comparison.

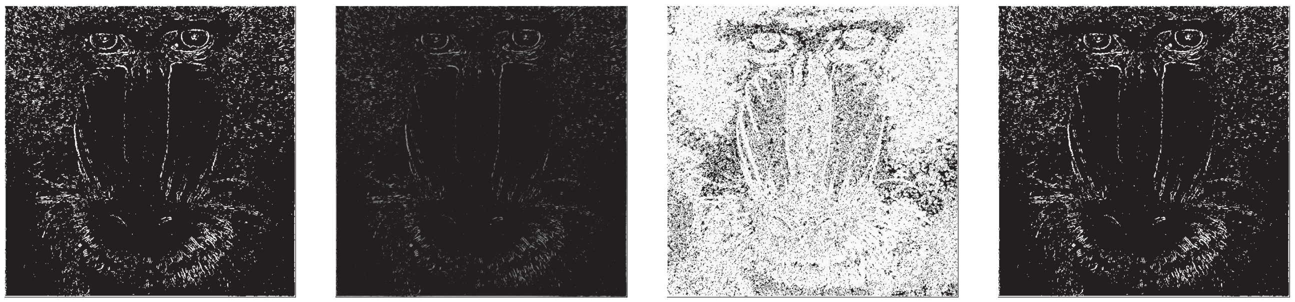
Approach	Precision	FF	LUT
Correlator	4	4	7
	6	8	11
Constant CSNG	8	9	13
Variable CSNG	8	25	43
SC - DC counter	8	8	7
DC - SC conventional SNG	8	8	9

Table 5. The absolute errors of each filter.

	Median filter	Gaussian filter	Edge detection	Threshold	Output error
Proposed CSC	0	0.0065	0.0096	0.0507	<0.001
CSC	0	0.0065	0.0201	0.7346	0.76

For the CSC threshold without using the CSNG, the output is shown in Figure 11c, where major inaccuracies due to the correlation loss can be observed. The threshold SN is generated using the same initial SNGs that generate the input SNs. Although the threshold SN is correlated initially with the inputs, the correlation at the threshold block is lost due to previous processing performed on the input SNs. The error value $e_{th} = 0.7346$ is very high, which proves the need for CSNG.

Finally, to produce the output of the CSC system utilizing the proposed correlating circuits, a special binary counter is used. As discussed earlier, this counter will output 0 if all the SN bits are zero; otherwise, the output will be one. The output of the CSC system using the methodology proposed in this work is shown in Figure 11d with negligible error (< 0.001). Table 5 summarizes the absolute error of each level in the CSC systems. This result proves that the proposed work can be used to build accurate and compact CSC systems.



(a) Conventionalbinary threshold

(b) CSNG threshold

(c) Threshold output when threshold SN is obtained from the initial SNG

(d) The CSC system output when using correlating circuits

Figure 11. The thresholding and the experimental setup outputs.

5. Conclusion

In this paper, a design methodology for a CSC system using correlating circuits is proposed. Two correlating circuits were designed to take advantage of correlation fully, especially when CSC circuits are cascaded to build a bigger system. The first is the correlator, which is used to correlate two SNs by the proposed min-relocate algorithm. Secondly, a CSNG is proposed to generate an SN wholly correlated with a target SN. The experimental results using an image processing case study showed that the proposed method increased the SC system accuracy, eliminated the need for intermediate conversion circuits between the CSC blocks, and obtained some hardware area savings. The proposed methodology will enable creation of efficient systems composed of different consecutive blocks in CSC rather than conventional SC with independent SNs. SC systems built using the methodology are more accurate and efficient in terms of area, power, and latency. Our future work investigates CSC implementations for more complex algorithms like convolutional neural networks.

References

[1] Gaines BR. Stochastic computing. In: Proceedings of the Spring Joint Computer Conference; 18–20 April 1967; Atlantic City, NJ, USA. New York, NY, USA: ACM. pp. 149-156.

[2] Li P, Lilja DJ, Qian W, Bazargan K, Riedel MD. Computation on stochastic bit streams digital image processing case studies. IEEE T VLSI Syst 2014; 22: 449-462.

[3] Canals V, Morro A, Oliver A, Alomar ML, Rossell JL. A new stochastic computing methodology for efficient neural network implementation. IEEE T Neur Net Lear 2016; 27: 551-564.

[4] Ichihara H, Sugino T, Ishii S, Iwagaki T, Inoue T. Compact and accurate digital filters based on stochastic computing. IEEE T Emerg Top Com (in press).

[5] Alaghi A, Hayes JP. Survey of stochastic computing. ACM T Embed Comput S 2013; 12: 1-19.

[6] Alaghi A, Hayes JP. Exploiting correlation in stochastic circuit design. In: 2013 IEEE 31st International Conference on Computer Design; 6–9 October 2013; Asheville, NC, USA. New York, NY, USA: IEEE. pp. 39-46.

[7] Alaghi A, Li C, Hayes JP. Stochastic circuits for real-time image-processing applications. In: 2013 50th ACM/EDAC/IEEE Design Automation Conference; 29 May–7 June 2013; Austin, TX, USA. New York, NY, USA: IEEE. pp. 1-6.

[8] Gaines BR. Stochastic computing systems. In: Tou JT, editor. Advances In Information Systems Science. Boston, MA, USA: Springer, 1969. pp. 37-172.

[9] Bertsekas DP, Tsitsiklis JN. Introduction to Probability. 2nd ed. Belmont, MA, USA: Athena Scientific, 2002.

[10] Budhwani RK, Ragavan R, Sentieys O. Taking advantage of correlation in stochastic computing. In: 2017 IEEE International Symposium on Circuits and Systems; 28–31 May 2017; Baltimore, MD, USA. New York, NY, USA: IEEE. pp. 1-4.

[11] Alaghi A, Qian W, Hayes JP. The promise and challenge of stochastic computing. IEEE T Comput Aid D 2017; 37: 1515-1531.

[12] Hayes JP. Introduction to stochastic computing and its challenges. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference; 8–12 June 2015; San Francisco, CA, USA. New York, NY, USA: IEEE. pp. 1-3.

[13] Ichihara H, Ishii S, Sunamori D, Iwagaki T, Inoue T. Compact and accurate stochastic circuits with shared random number sources. In: 2014 IEEE 32nd International Conference on Computer Design; 19–22 October 2014; Seoul, South Korea. New York, NY, USA: IEEE. pp. 361-366.

[14] Yuan B, Wang Y, Wang Z. Area-efficient scaling-free DFT/FFT design using stochastic computing. IEEE T Circuits-II 2016; 63: 1131-1135.

[15] Yuan B, Zhang C, Wang Z. Design space exploration for hardware-efficient stochastic computing: a case study on discrete cosine transformation. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing; 20–25 March 2016; Shanghai, China. New York, NY, USA: IEEE. pp. 6555-6559.

[16] Alaghi A, Hayes JP. Fast and accurate computation using stochastic circuits. In: Proceedings of the Conference on Design, Automation and Test in Europe; 24–28 March 2014; Dresden, Germany. Leuven, Belgium: European Design and Automation Association. p. 76.

[17] Kim K, Lee J, Choi K. An energy-efficient random number generator for stochastic circuits. In: 2016 21st Asia and South Pacific Design Automation Conference; 25–28 January 2016; Macau, China. New York, NY, USA: IEEE. pp. 256-261.

[18] Chen TH, Ting P, Hayes JP. Achieving progressive precision in stochastic computing. In: 2017 IEEE Global Conference on Signal and Information Processing; 14–16 November 2017; Montreal, Canada. New York, NY, USA: IEEE. pp. 1320-1324.

[19] Alaghi A, Hayes JP. On the functions realized by stochastic computing circuits. In: Proceedings of the 25th Great Lakes Symposium on VLSI; 20–22 May 2015; Pittsburgh, PA, USA. New York, NY, USA: ACM. pp. 331-336.

[20] Jenson D, Riedel M. A deterministic approach to stochastic computation. In: Proceedings of the 35th International Conference on Computer-Aided Design; 7–10 November 2016; Austin, TX, USA. New York, NY, USA: ACM. p. 102.

[21] Najafi MH, Jamali-Zavareh S, Lilja DJ, Riedel MD, Bazargan K, Harjani R. Time-encoded values for highly efficient stochastic circuits. IEEE T VLSI Syst 2017; 25: 1644-1657.

[22] Najafi MH, Lilja D. High quality down-sampling for deterministic approaches to stochastic computing. IEEE T Emerg Top Com (in press).

[23] Vahapoglu E, Altun M. From stochastic to bit stream computing: accurate implementation of arithmetic circuits and applications in neural networks. arXiv preprint, arXiv:180506262, 2018.

[24] Ting P, Hayes JP. Eliminating a hidden error source in stochastic circuits. In: 2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems; 23–25 October 2017; Cambridge, UK. New York, NY, USA: IEEE. pp. 1-6.

[25] Ting PS, Hayes JP. Isolation-based decorrelation of stochastic circuits. In: 2016 IEEE 34th International Conference on Computer Design; 2–5 October 2016; Scottsdale, AZ, USA. New York, NY, USA: IEEE. pp. 88-95.

[26] Vahapoglu E, Altun M. Accurate synthesis of arithmetic operations with stochastic logic. In: 2016 IEEE Computer Society Annual Symposium on VLSI; 11–13 July 2016; Pittsburgh, PA, USA. New York, NY, USA: IEEE. pp. 415-420.

[27] Najafi MH, Lilja DJ. High-speed stochastic circuits using synchronous analog pulses. In: 2017 22nd Asia and South Pacific Design Automation Conference; 16–19 January 2017; Chiba, Japan. New York, NY, USA: IEEE. pp. 481-487.