A New Lightweight and High Fault Tolerance Sobel Edge Detection Using Stochastic Computing

Ming Ming Wong[1*], Dennis M. L. D. Wong[2], Cishen Zhang[3], Ismat Hijazin[3]

[1] Faculty of Engineering, Computing and Science, Swinburne University of Technology Sarawak, Malaysia.
[2] Heriot Watt University Malaysia, Wilayah Persekutuan Putrajaya, Malaysia.
[3] Faculty of Science, Engineering and Technology, Swinburne University of Technology, Hawthorn, Australia.

* Corresponding author. Email: wmingming7@gmail.com
Manuscript submitted March 24, 2017; accepted June 23, 2017.
doi: 10.17706/ijcee.2017.9.2.403-420

Abstract: A new Stochastic Computing (SC) circuit design paradigm for image processing system is presented in this work. Two improved SC computational functions are derived, which are namely the stochastic scaled addition and stochastic absolute value of difference. Data correlation is also incorporated in the design for effective circuit size reduction without imposing accuracy degradation in the hardware implementations. The proposed SC functions are next employed to design the new and lightweight Sobel edge detection. Experimental results obtained from detailed test analysis have proven that new implementation has satisfactory accuracy level and higher fault tolerance capability in comparison with their conventional counterparts. The works proposed are also implemented on an Altera Cyclone V 5CGXFC7D6F31C6 FPGA for hardware complexity evaluation.

Key words: Stochastic computing, scaled addition, absolute value of difference, Sobel edge detection, fault tolerance.

1. Introduction

Stochastic computing (SC) [1] was first introduced several decades ago and it has recently gained a fair amount of attentions in the integrated circuit (IC) design community. This technique, which incorporates elements from probability theory, has proven to be able to handle computation uncertainties in a more effective and efficient manner [2]. Thus, it emerges as an unconventional and non-deterministic computational technique with high fault tolerance [3], [4]. Furthermore, SC is particularly attractive in IC design as it requires low complexity computation blocks. In general, SC is seen as a promising alternative in comparison to its conventional binary computing counterpart, which usually has a higher computational cost.

Though SC has been known for decades, very few physical realizations have been proposed. Initially, SC applications were limited to the field of neural networks [5] and machine controls [6]. Until recent years, it was discovered that SC efficiently simplifies some mathematical functions which are computational expensive in binary computation. These functions can be efficiently approximated using stochastic logic with minimal hardware requirements and without significant accuracy degradation. Ever since, SC implementation has been extended to image processing [7]-[9], error control coding applications [10] and digital filter design [11]-[14].

The main contributions of this work are three-folds. First, we presented the improved designs for

stochastic scaled addition and stochastic absolute value of difference. The proposed designs are proven to be smaller than the existing works in the literature.

Second, the proposed SC functions are employed to design new SC circuit for image processing system, which is namely the Sobel edge detection. The work also incorporated data correlation to achieve further circuit size reduction without imposing accuracy degradation in SC hardware implementations.

Third, detailed test analysis are performed on the new SC circuit implementation in order to measure their accuracy level, fault tolerance capability and also the cost and performance in hardware. Based on the obtained result, it shows that our work has high accuracy level, better in fault tolerance and is more hardware cost effective compared to the conventional counterparts.

The rest of the paper is organized as follows. Section 2 explains the fundamental of SC theory which includes the SC circuit design as well as the stochastic computational elements. Detailed study of data correlation in SC circuit implementation is discussed next in Section 3. The improved stochastic functions, specifically the stochastic scaled addition and stochastic absolute value of difference are derived in Section 4 and 5 respectively.

Section 6 reviews the detailed design and implementation of the proposed stochastic Sobel edge detection. The experimental results (accuracy and fault tolerance analysis) as well as the hardware implementation are reported and discussed in Section 7. Finally, some conclusion remarks are drawn in Section 8.

2. Basic Theory of Stochastic Computation

The basic rule of SC is that the computational data (in bit-streams) are represented as stochastic sequences and are processed in the form on digitized probabilities [3]. Naturally, the representations and all the involved computations always lie within the real-number unit interval [0, 1]. Stochastic representation can be coded in two formats: *SC-unipolar* and *SC-bipolar* [1].

In *SC*-unipolar format, the input s is a real number within the unit interval, i.e. $0 \le s \le 1$. As an example, a 2's complement binary input bit-stream $\{0011\}_2$ is represented in stochastic bit-streams S, consisted of 3 of bit '1' out of $2^4 = 16$ bits (remaining bits are zeros). This stochastic bit-streams S, is also interpreted as $p = P(S = 1) = 3/16$. On the other hand, in *SC*-bipolar format, the range of the real number input, s, is extended to $-1 \le s \le 1$. Consider a 2's complement binary input bit-stream $\{1100\}_2$. In SC-bipolar bit-streams S, the deterministic value is mapped to $p = P(S = 1)/2 = 12/(16 \times 2) = 6/16$.

In other words, stochastic representation observes the probability of 1s at arbitrary bit position in S. Such representation serves as the main reason for having high fault tolerance in SC. A single bit-flip in a long bit-stream causes a minor change in original logical value. On the contrary, a single bit-flip in the conventional 2's complement computation will result in huge error especially if the bit-flip occurs on the higher-order bit.

A general SC architecture consists of three major components: the stochastic number generator (SNG), the stochastic computing elements (SCE) and the de-randomizer. SNG is used to convert the deterministic binary value into stochastic bit-streams by using the (pseudo) random number generator (PRNG) and a comparator. The PRNG is usually implemented using linear feedback shift register (LFSR). Meanwhile, the de-randomizer, which is usually a binary counter, is used to decode the output bit-stream back into deterministic binary value. Both the architectures of SNG and de-randomizer are as illustrated in Fig. 1. The SCEs are arithmetic functions such as multiplier and adder/subtractor which are computed using SC.

Multiplication of two inputs streams, which is computational intensive in conventional signed binary computing, can be performed using single logical gate in SC. Assuming that the stochastic input bit-streams, X_1 and X_2 are suitably uncorrelated, the output for their multiplication, Y, is derived as,

$$y = P(Y=1)$$
$$= P(X_1=1)P(X_2=1)+(1-P(X_1=1))(1-P(X_2=1))$$

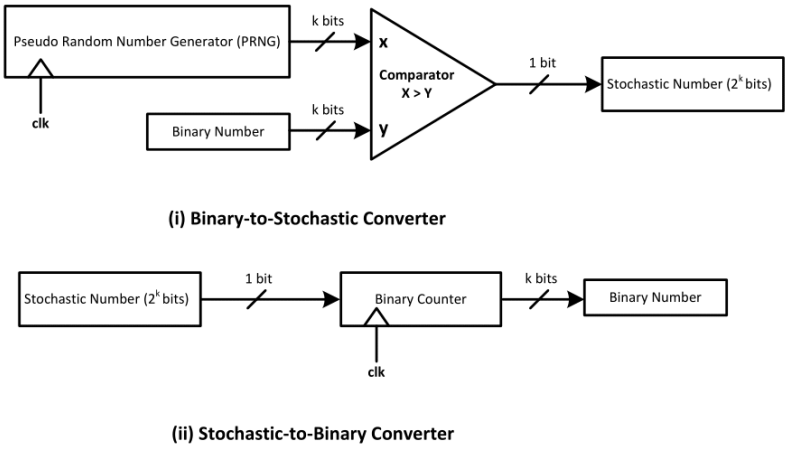


Fig. 1. *SC* components: (i) Stochastic number generator (SNG) (ii) De-randomizer.

Stochastic multiplication in bipolar format is clearly a logical XNOR operation between input bit-streams, X_1 and X_2 in digital circuit. For unipolar format, the multiplication is performed using a logical AND operation instead. Stochastic multiplier for both unipolar and bipolar formats are as depicted in Fig. 2.

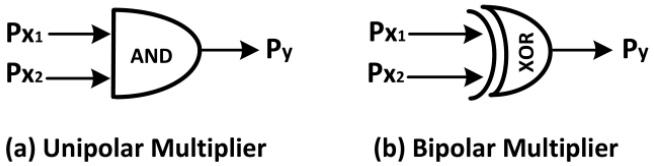


Fig. 2. Stochastic multiplier for *SC*-unipolar and SC-bipolar formats.

Addition in SC is performed using a special operation, termed as scaled addition. The addition is scaled such that the value always lies between the probability interval [0, 1]. With S is a constant scale, the sum of two independent stochastic bit-streams X_1 and X_2, Y, is defined as,

$$y = P(Y=1)$$
$$= P(S=1)P(X_1=1)+(1-P(S=1))P(X_2=1)$$
$$= SX_1+(1-S)X_2$$

Thus, multiplexer with conditional select line S, set as $P(S) = 1/2$ can be used to realize the scaled addition of two stochastic bit-streams in digital circuit. Subtraction in SC is similar to the adder except that the stochastic scaled substractor requires an additional inverter and this only feasible in SC-bipolar format. Both the stochastic scaled adder and scaled substractor are illustrated in Fig. 3.

In general, by using SC, arithmetic functions can be efficiently computed through simple circuit using standard logic elements. In term of hardware realizations, stochastic multiplication is implemented using XNOR (or AND) gate and multiplexer is used to compute stochastic scaled addition. In addition, computational data represented in stochastic sequences resulted in high fault tolerance level as they are less susceptible towards bit-flipping errors. Hence, it is evident that SC circuit design is hardware cost-effective with high fault tolerance.

However, based on the conventional belief, SC hardware circuits will only perform properly when the computational data are uncorrelated. In the next section, this issue will be addressed and analyzed in detail.

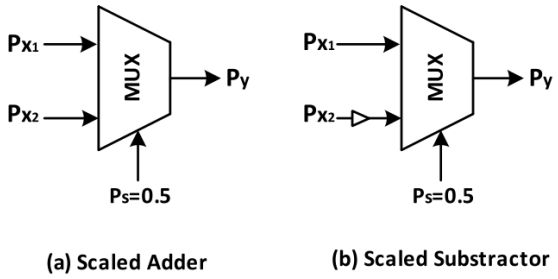


Fig. 3. Stochastic scaled adder/substractor.

3. Data Correlation in SC Circuit Implementation

One of the concerns in SC circuit design is that the accuracy of the systems is affected by the correlation of its computational data. Based on the common understanding stated in the literature, the accuracy level of the SC circuit tends to degrade significantly even when the data are moderately correlated. Technically, correlation in the data bit streams alters the expected output of a stochastic logic circuit.

Such implication can be clearly observed in stochastic multiplication. For instance, multiplication of two identical (massively correlated) stochastic input sequences X and Y produces an output Z, with its probability as P_x, instead of the desired product P_xP_y. Similar error occurs upon the multiplication of two stochastic input sequences X and Y which happen to be the inverse of one another. This produces a stream of zeros as the output, Z. Both examples are as depicted in (b) and (c) in Fig. 4.

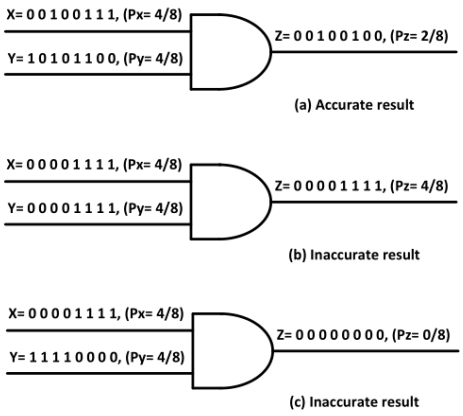


Fig. 4. Stochastic multiplication with accurate result using uncorrelated inputs in (a) and inaccurate results using correlated inputs in (b) and (c).

The real context of data correlation and its effect in SC remained unclear until the recent work reported by [15], [16]. Both studies have performed an in-depth analysis of SC circuit with correlated data. The study by Parhi et. Al has precisely reported the closed form output expressions for stochastic logic elements [15]. The findings showed that the expressions derived using logical AND gates deviate between the correlated and uncorrelated input data. This explains the accuracy degradation that occurs during the realization of stochastic multiplication using AND gate. The study further reported that multiplexer, on the other hand, shows no output deviation in both correlated and uncorrelated data.

Therefore, unlike stochastic multiplication, data correlation would not degrade the computational

accuracy in stochastic scaled addition, which is implemented using multiplexer (refer Fig. 5). Furthermore, the work by Alaghi and Hayes has also stated that correlation is not always harmful in SC circuits [16]. The authors proved that correlation in SC data can actually be exploited for better SC circuit designs.

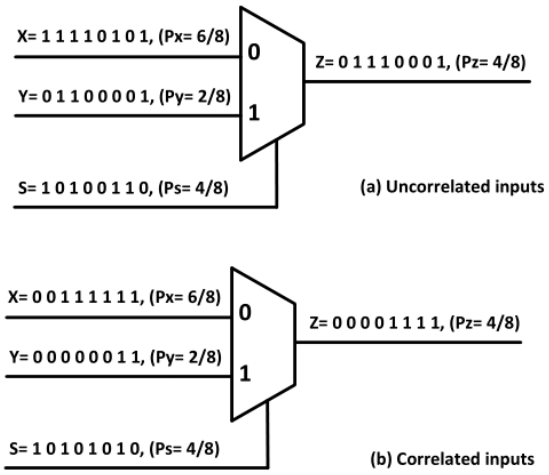


Fig. 5. Accurate result in stochastic addition using both (a) uncorrelated inputs and (b) correlated inputs.

These new findings defy conventional understandings and provide insight for further enhancement and optimization in SC circuit implementations. In this work, we attempt to design new stochastic scaled adder/subtractor, that will be suitable for image processing applications. These computations are namely the **stochastic scaled addition** and **stochastic absolute value of difference**. While stochastic scaled subtraction is basically the same as the stochastic scaled addition (only having one input's polarity inverted), both operations will be referred as stochastic scaled addition from this point onwards.

In the consecutive sections, we will first discuss the implementation issues of the above-mentioned functions in stochastic circuits which are followed by the proposed solutions

4. Stochastic Scaled Addition

As discussed in Section 2, addition is performed as scaled addition in SC, which is realized using multiplexer in hardware implementation. Consider a set X of four inputs as $\{X_1, X_2, X_3, X_4\}$. Its summation Y using SC is derived as follows.

$$\begin{aligned} Y &= X_1 + X_2 + X_3 + X_4 \\ Y_{sc} &= \frac{1}{4}\left(P(X_1) + P(X_2) + P(X_3) + P(X_4)\right) \end{aligned} \tag{1}$$

$$Y_{sc} = \frac{1}{2}\left(\frac{1}{2}\left(P(X_1) + P(X_2)\right) + \frac{1}{2}\left(P(X_3) + P(X_4)\right)\right) \tag{2}$$

Y_{sc} in both (1) and (2) presents the conventional stochastic scaled addition. In order to retain the value in the probability interval of [0, 1] the summation value has to be scaled down accordingly. The scaled addition in hardware implementation can be performed in two approaches as described in (1) and (2) where the difference only lies in the number and the size of the multiplexers needed (refer Fig. 6). The prior requires single 4-1 multiplexer where the latter uses three but smaller size 2-1 multiplexers. The combinatorial circuit resultant from (2) will be in the form of tree structure and hence shorter in its critical

path. In term of functionality, both architectures performed the same computation.

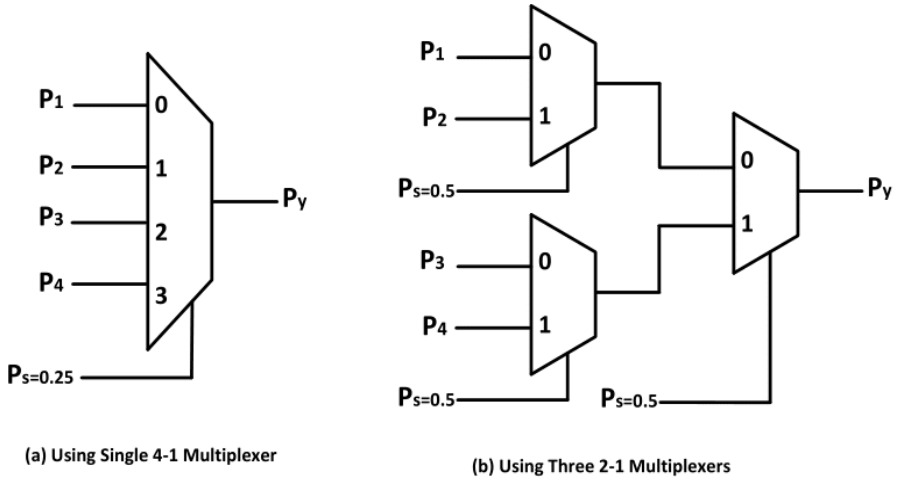


Fig. 6. Stochastic scaled addition using (a) Three 2-1 Multiplexers and (b) Single 4-1 Multiplexer. Both architectures are same in functionality but different in hardware cost and performance.

By comparing the derivation of Y_{sc} ((1) and (2)) and its implementation in Fig. 6, it is evident that the scaling is achieved through the multiplexer with the scaling factor controlled by the selection line. Therefore, should there be a large amount of summations to be performed in SC, the amount multiplexers required will be increased as well. Take note that a single 2-1 multiplexer is comprised of 2 AND, 1NOT and 1 XOR gates, while a single 4-1 multiplexer is made up of 4 AND, 4 NOT and 3 XOR gates. Subsequently, this will become a problem in lightweight applications if massive simultaneous additions are required.

4.1. New Stochastic Scaled Addition

In this work, we proposed a new stochastic scaled addition which does not require any multiplexer. Using the previous example and given that $n = N/4$, the scaled addition can be further factored as the following.

$$\begin{aligned} Y_{sc} &= \frac{1}{4}\left(P(X_1) + P(X_2) + P(X_3) + P(X_4)\right) \\ Y_{newSC} &= \frac{1}{4}\left[\left(\frac{N+X_1}{2N}\right)+\left(\frac{N+X_2}{2N}\right)+\left(\frac{N+X_3}{2N}\right)+\left(\frac{N+X_4}{2N}\right)\right] \\ &= \frac{1}{4}\left[\left(\frac{n+\frac{X_1}{4}}{2n}\right)+\left(\frac{n+\frac{X_2}{4}}{2n}\right)+\left(\frac{n+\frac{X_3}{4}}{2n}\right)+\left(\frac{n+\frac{X_4}{4}}{2n}\right)\right] \\ &= \frac{1}{4}\left[P\left(\frac{X_1}{4}\right)+P\left(\frac{X_2}{4}\right)+P\left(\frac{X_3}{4}\right)+P\left(\frac{X_4}{4}\right)\right] \end{aligned} \tag{3}$$

Using the superposition principle, the conventional stochastic scaled addition can be written as Y_{newSC} in (3). With that, the scaling is performed prior to addition and this is illustrated in the following example. Given a set X of four inputs, $\{X_1, X_2, X_3, X_4\}$ as {28, 4, 12, 16}, their correspondent representation SC-bipolar format, $\{P(X_1), P(X_2), P(X_3), P(X_4)\}$ will be denoted as $\left\{\frac{30}{32},\frac{18}{32},\frac{22}{32},\frac{24}{32}\right\}$. With the input scaled by ¼, the SC-bipolar representation of $\left\{P\left(\frac{X_1}{4}\right),P\left(\frac{X_2}{4}\right),P\left(\frac{X_3}{4}\right),P\left(\frac{X_4}{4}\right)\right\}$ will be denoted as $\left\{\frac{7}{8},\frac{4}{8},\frac{5}{8},\frac{6}{8}\right\}$. The summation of these inputs in SC is then listed as follows.

$$Y = X_1 + X_2 + X_3 + X_4$$

$$Y_{sc} = \frac{1}{4}\left(P(X_1) + P(X_2) + P(X_3) + P(X_4)\right)$$

$$= \frac{1}{4}\left(\frac{30}{32} + \frac{18}{32} + \frac{22}{32} + \frac{24}{32}\right)$$

$$= \frac{23}{32}$$

$$Y_{newSC} = \frac{1}{4}\left(P\left(\frac{X_1}{4}\right) + P\left(\frac{X_2}{4}\right) + P\left(\frac{X_3}{4}\right) + P\left(\frac{X_4}{4}\right)\right)$$

$$= \frac{1}{4}\left(\frac{7}{8} + \frac{4}{8} + \frac{5}{8} + \frac{6}{8}\right) = \frac{22}{32} \approx Y_{SC}$$

The proposed computation can be directly applied to the additions that appeared at the upfront of digital circuits. This is where the additions are required to perform first on the system's inputs. In this case, the scaling is done directly on the binary inputs prior to SNG conversion. Scaling by ½ in binary sequence is a binary right shift. Thus it is much simpler compared to using a multiplexer in SC.

Furthermore, as a result of the scaling, the initial *N*-bits inputs are reduced in size to *N*-1 bits. Hence, the complexity of the required SNG modules will be reduced accordingly as the length of the stochastic sequence is now 2^{N-1} bits (instead of 2^N bits). Addition will eventually become as simple as concatenating both stochastic sequences of 2^{N-1} bits to produce an output sequence of original length, 2^N bits.

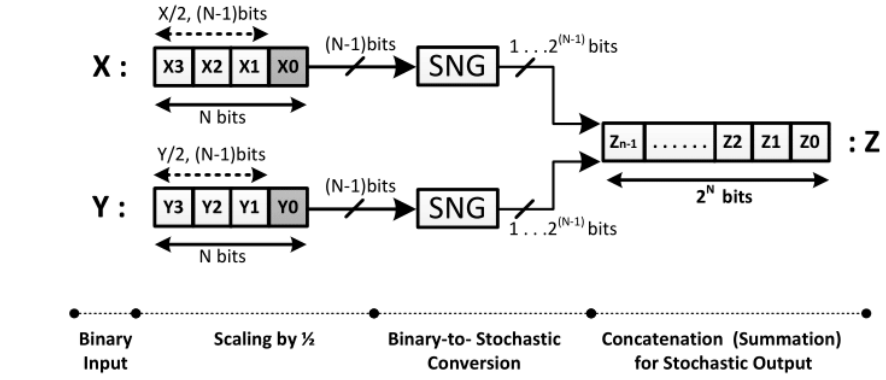
In any other part of the SC circuit, the proposed scaled addition can be implemented with a slight modification. The difference now is that the inputs to the additions are stochastic sequences instead of primary binary inputs. Therefore, scaling by binary right shifting is no longer feasible. However, correlation can be exploited to the good use in this case.

As discussed in Section 3 (refer Fig. 5), scaled addition using multiplexer works equally effective for both correlated and uncorrelated computational data. While the input stochastic sequences are in correlated form, the scaling can be done by simply sampling the alternate bits from both sequences. This measure can be easily deployed in hardware as it only requires re-wiring without any additional hardware. Both the sequences of 2^N bits are sampled down to 2^{N-1} bits and are concatenated to produce the addition output of 2^N bits.

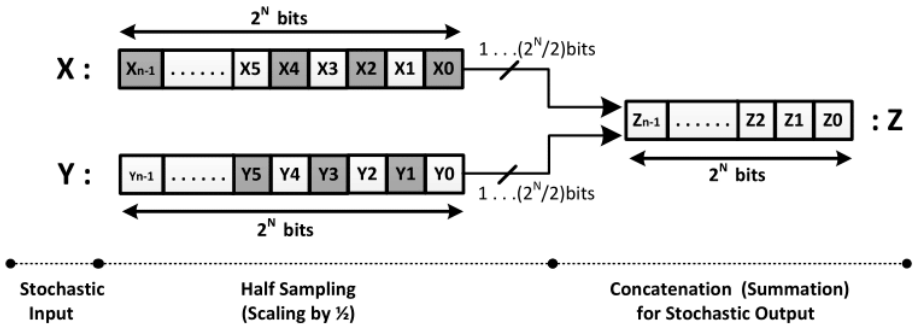
The architectures of both of the proposed scaled addition are depicted in Fig. 7. In this new stochastic scaled addition, the hardware required is significantly less than the conventional stochastic scaled addition. Multiplexer is no longer required to perform the scaling and the addition becomes just a mere bit stream concatenation. These new stochastic scaled additions are employed in digital Sobel edge detection, which will be presented in Section 6.

5. Stochastic Absolute Value of Difference

In stochastic computing, basic arithmetic operations such as addition, subtraction, multiplication and inversion can be conveniently implemented through combinational logic gates such as multiplexer, NOT and AND gates. Other operations such as computing an absolute value does not have a straightforward implementation in SC [9]. In [9], the authors demonstrated that these stochastic computational elements can be implemented through FSM. This has unfortunately produced a relatively complicated and large sequential circuit to perform the absolute value of difference, even though it just a simple operation.



(a) New Scaled Addition for front part of SC circuit



(b) New Scaled Addition for midpart of SC circuit

Fig. 7. The proposed stochastic scaled addition which could be employed at the front part of the SC circuit (a) and in any other part of the SC circuit (b). Both stochastic adders do not require multiplexer.

Similar to stochastic scaled addition in previous section, data correlation can be exploited for complexity reduction in SC circuit design [16]. To be precise, absolute value of difference of two correlated inputs can be computed by using a simple XOR gate, as shown in Fig. 8. As an example, with X and Y are correlated inputs, the probability of the difference between X and Y is derived as P_X - P_Y if $P_X > P_Y$ or else $P_Y - P_X$. In other words, the output by XOR-ing two correlated inputs produces $|P_Y - P_X|$. Take note that XOR fed with independent input produces a different function, $P_x(1-P_y)+ P_y(1-P_x)$.

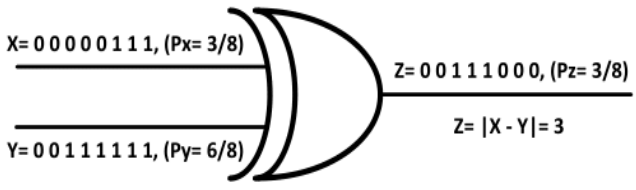


Fig. 8. Stochastic absolute value of difference using XOR gate for correlated inputs in SC unipolar format.

This new stochastic absolute value of difference has been applied in an image processing algorithm, the edge detections, which is reported in [17]. The resulted circuit is demonstrated to be smaller than using the conventional stochastic absolute value of difference through FSM.

5.1. New Stochastic Absolute Value of Difference

Stochastic absolute value of difference using FSM [9] involves a significant amount of hardware resources (registers). Thus, the single XOR approach proposed in [17] is definitely a wiser selection for SC circuit design. Unfortunately, such exploitation is only feasible in SC-unipolar format. While several digital signal and image processing applications deal with real numbers data, *SC*-bipolar format is often needed (instead of *SC*-unipolar) in the overall SC circuit. Should the computation be required for *SC*-bipolar data, a conversion from bipolar to unipolar correlated data is required prior the operation and vice versa after the operation. As a result, this will impose an increase the hardware circuit size.

In this work, we propose a different architecture for stochastic absolute value of difference, which can be employed effectively for data in *SC*-bipolar format. The first step of the computation involves stochastic scaled addition, of which is presented in the previous section. The second and also the last step is to determine the absolute value of the summation.

Table 1 shows an example of a list of real number with the precision of 3-bits and represented in SC-bipolar format. Based on listing given, we can observe that any positive real number, x, in SC-bipolar format has a probability $P(x) \geq ½$. Therefore, in term of hardware circuitry, the polarity of the bipolar stochastic sequence can be easily determined by using a binary counter.

Table 1. Stochastic Bipolar Sequences with Precision of 3-Bits

Real Number	Stochastic Number	Real Number	Stochastic Number
-8	0/8	8	8/8
-7	0/8	7	7/8
-6	1/8	6	7/8
-5	1/8	5	6/8
-4	2/8	4	6/8
-3	2/8	3	5/8
-2	3/8	2	5/8
-1	3/8	1	4/8
0	4/8		

If the probability of a sequence x is found to be $P(x) \geq ½$. The sequence shall remain unchanged and this is also its absolute value. On the other hand, if $P(x) < ½$ is found instead, the absolute value of x can be computed by bit inverting the sequence. Therefore, given -x is a negative number, its absolute value in SC is denoted as $P(-x) = 1-P(x)$.

In term of hardware cost, our proposed circuit requires additional binary counter in comparison to the absolute value of difference for SC-unipolar data reported in [17]. However, it is worth noting that the bit counting is operating concurrently with the addition of the stochastic sequences. Therefore, in term of computational performance, our proposed work is similar to the work in [17]. Upon the completion of stochastic addition (which also marks the end of the counting process), if the total number of '1' is larger than $N/2$, we can deduce the output sequence is positive and hence is also the final absolute value. Otherwise, a negative output sequence is determined and its absolute value can be easily derived by inverting the entire output bit sequence.

The proposed stochastic absolute value of difference is depicted in Fig. 9 and is also applied to Sobel edge detection, which will be explained further in the next section.

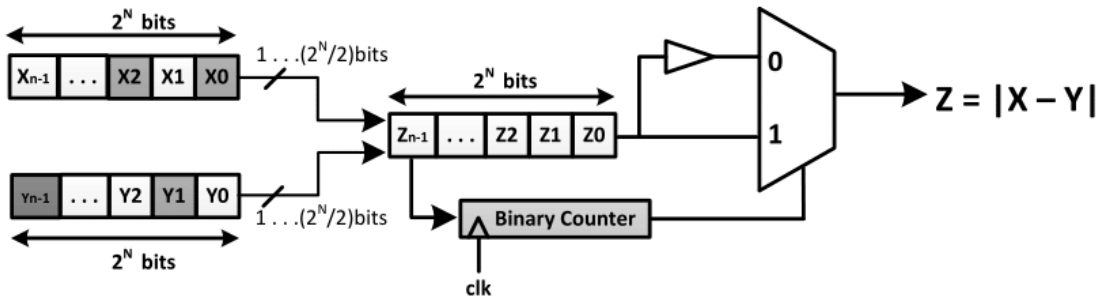


Fig. 9. The layout of the new stochastic absolute value of difference for correlated inputs in SC-bipolar format.

6. The Proposed Stochastic Sobel Edge Detection

Basic edge detection algorithms operate by performing the first-order derivatives of digital grey level images, which is computed by using the vertical and horizontal gradients, G_x and G_y. Precisely, an image gradient is determined by calculating the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ of an image pixel f in position (x, y). This can be obtained through the application of different type of masks around the pixel f.

Prewitt and Sobel masks are by far the most common choices in digital image gradient computation and each offering different advantages. Prewitt is simpler in nature while Sobel, which is implemented in this work, is superior in noise removal. With the weighting at the central pixel in Sobel mask (refer Fig. 10), the gradients computation is as shown in (4) and (5).

$$G_x = \frac{\partial f}{\partial x} = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7) \tag{4}$$

$$G_y = \frac{\partial f}{\partial y} = (Z_1 + 2Z_2 + Z_3) - (Z_7 + 2Z_8 + Z_9) \tag{5}$$

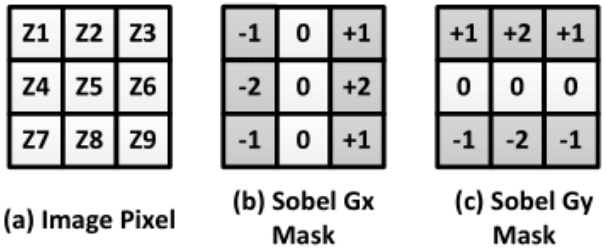


Fig. 10. Pixels numbering in an 3×3 image is shown in (a). The horizontal and vertical Sobel masks are depicted in (b) and (c) respectively. Both masks are needed to compute the G_x and G_y gradients.

While the masks are used to derive gradients G_x and G_y, the most important quantity here is to find the magnitude of these gradients as described in (6). This implementation is complex in hardware as it requires intensive computations, which are namely the squaring and square root operations. Alternatively, an approximation method which involved only derivation of the absolute value of difference and the summation of both the gradients are more commonly deployed such as stated in (7). The new SC architectures proposed in Section 4.1 and Section 5.1 are employed in this work to design a novel stochastic Sobel edge detection.

$$\nabla f = \sqrt{G_x^2 + G_y^2} \tag{6}$$

$$\nabla f = |G_x| + |G_y| \tag{7}$$

The magnitude of the gradients, ∇f in (7) using SC is generally derived as the following in (8). Take note that each image pixel, Zn with precision level of 8-bits is coded in *SC*-bipolar format $P_n = P(Z_n = 1)$. Using the new stochastic scaled addition, where the scaling is performed prior to addition, all the eight pixel entries for both horizontal and vertical gradients are scaled down by 1/16 prior to conversion into *SC*-bipolar format.

Such approach promotes a significant hardware saving as each of the scaled pixel is mapped to stochastic correlated stream of 24 = 16 bits. The conventional stochastic architecture, however, converts the 8-bit pixel entry to stochastic sequence of 2^8 = 256 bits stream. Not only that, the new architecture exploited data correlation in the computation and therefore the need of PRNG is no longer needed.

Furthermore, eight pixel entries of 16-bits correlated stochastic sequences are concatenated and form a stream of 128-bits sequence which represents the gradient value. The absolute value of each gradient can be easily determined from its probability value. If the gradient weightage is $P(G) < 0.5$, its absolute value is equivalent to the bit inverted of its stochastic sequence. Otherwise, the sequence stream shall remain unchanged. The process is performed using binary counter. Eventually, the absolute value of both gradients, G_x and G_y are again concatenated to produce a stochastic output sequence of 256-bits. In this approach, no multiplexer is required in the architecture and this enables further hardware cost saving.

$$\begin{aligned}
\nabla f &= |G_x| + |G_y| \\
&= \left|\left(Z_3 + Z_6 + Z_6 + Z_9\right) + \left(-Z_1 - Z_4 - Z_4 - Z_7\right)\right| \\
&\quad + \left|\left(Z_1 + Z_2 + Z_2 + Z_3\right) + \left(-Z_7 - Z_8 - Z_8 - Z_9\right)\right| \\
\nabla f_{SC} &= \frac{1}{2}\left[\frac{1}{2}\left|\frac{1}{4}\left(P_3 + P_6 + P_6 + P_9\right) + \frac{1}{4}\left(-P_1 - P_4 - P_4 - P_7\right)\right| + \frac{1}{2}\left|\frac{1}{4}\left(P_1 + P_2 + P_2 + P_3\right) + \frac{1}{4}\left(-P_7 - P_8 - P_8 - P_9\right)\right|\right] \\
&= \frac{1}{16}\left|P_3 + P_6 + P_6 + P_9 + \overline{P}_1 + \overline{P}_4 + \overline{P}_4 + \overline{P}_7\right| + \frac{1}{16}\left|P_1 + P_2 + P_2 + P_3 + \overline{P}_7 + \overline{P}_8 + \overline{P}_8 + \overline{P}_9\right|
\end{aligned} \tag{8}$$

The design of the above mentioned approach also is deduced in (9) and the architecture layout is as depicted in Fig. 11. Overall, the new stochastic Sobel edge detection serves as a better alternative for the existing SC architecture reported in [9] and [17]. The work in [9] is complex with large sequential circuit using FSM and the approach in [17] is simple but requires several conversion modules, which also consumes large amount of hardware resources.

Further verification and validation of the proposed stochastic Sobel edge detection, as well as its implementation on hardware platform are also performed in this work. The experimental results deduced will be presented in detail in Section 7.

$$\begin{aligned}
\nabla f_{SCNew} &= |G_x| + |G_y| \\
&= \left\{\begin{array}{l}
\left|P\left(\frac{Z_1}{16}\right), P\left(\frac{Z_2}{16}\right), P\left(\frac{Z_2}{16}\right), P\left(\frac{Z_3}{16}\right), \overline{P}\left(\frac{Z_7}{16}\right), \overline{P}\left(\frac{Z_8}{16}\right), \overline{P}\left(\frac{Z_8}{16}\right), \overline{P}\left(\frac{Z_9}{16}\right)\right|, \\
\left|P\left(\frac{Z_3}{16}\right), P\left(\frac{Z_6}{16}\right), P\left(\frac{Z_6}{16}\right), P\left(\frac{Z_9}{16}\right), \overline{P}\left(\frac{Z_1}{16}\right), \overline{P}\left(\frac{Z_4}{16}\right), \overline{P}\left(\frac{Z_4}{16}\right), \overline{P}\left(\frac{Z_8}{16}\right)\right|
\end{array}\right\}
\end{aligned} \tag{9}$$

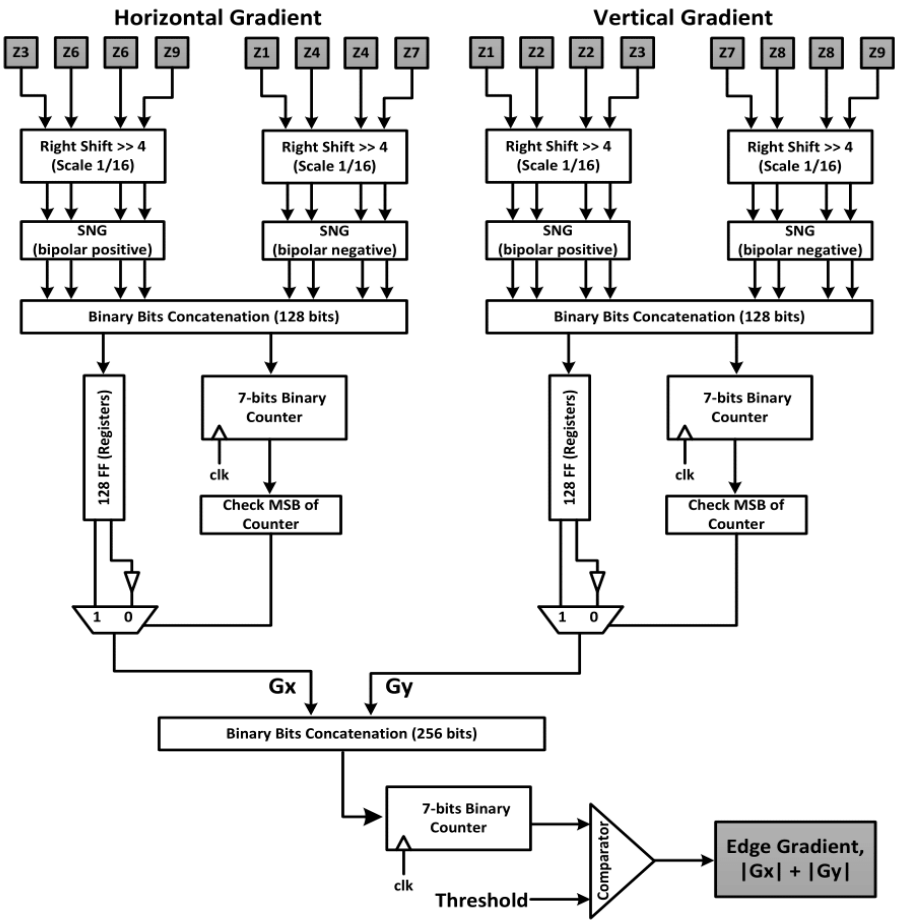


Fig. 11. The architecture layout of the new proposed stochastic Sobel edge detection.

7. Experimental Result

For verification and validation purposes, the new stochastic Sobel edge detection is tested and implemented on hardware circuit. The accuracy of the design is verified through MATLAB simulation using different images. The result obtained is benchmarked with the MATLAB Sobel edge detection module. In addition, fault tolerance analysis is performed on both algorithms to testify their susceptibility towards soft errors. The hardware requirement of the proposed architecture and its performance in FPGA implementation are also deduced in this study.

7.1. Accuracy Analysis

The results obtained from Sobel edge detection simulation using our new stochastic design and its conventional counterpart are shown in Fig. 12. The simulation of the MATLAB edge detection algorithms is performed using 8-bits unsigned binary bits. Meanwhile, the simulation of our stochastic design is executed on 2^4-bits SC-bipolar format but this comes with the precision level of 8-bits per image pixel, without using 2^8-bits stochastic sequences. Such reduction is possible due to the scaled-before-addition approach, which is presented in Section 4.1.

Based on the obtained simulation results, it is proven that the accuracy performance in our stochastic Sobel edge detection is not degraded even though it consumes substantially fewer computational elements than the conventional method.

7.2. Fault Tolerance Analysis

In this study, two fault tolerance analysis are performed on the proposed stochastic Sobel edge detection and the test results are compared with its conventional counterpart. The first testing is conducted by randomly injecting soft errors in the internal circuits and the corresponding average output error from each implementation is measured. The soft errors are injected into three different parts of the circuits; the computation of G_x, G_y and ∇f respectively.

This is done by randomly inflicting a substantial amount of bit-flip (ranging from 6% to 50%) in the internal circuit. In addition, various degree of bit-flipping errors are also injected randomly during the pixel processing and covering 5% to 50% of the entire image. Random bit-flipping is implemented by inserting XOR gates connected to a global random noise source, which is constructed using Linear Feedback Shift Register (LFSR) and a comparator.

The average output error deduced from both implementations under difference bit-flip and pixel error rates are tabulated in Table 2. The calculation of the average output error, E, is shown in (10) [9];

$$E = \frac{\sum_{i=1}^{H}\sum_{i=1}^{W}\left|P'_{i,j} - P_{i,j}\right|}{255 \cdot H \cdot W} \tag{10}$$

where P is the output image of the implementation without injected noise, P' is the output image of the implementation under fault tolerance testing and H and W are the height and the width of the image respectively. In the case of 30% pixel error rate, the output images obtained from both implementations with various bit-flip error rate is depicted in Fig. 13.

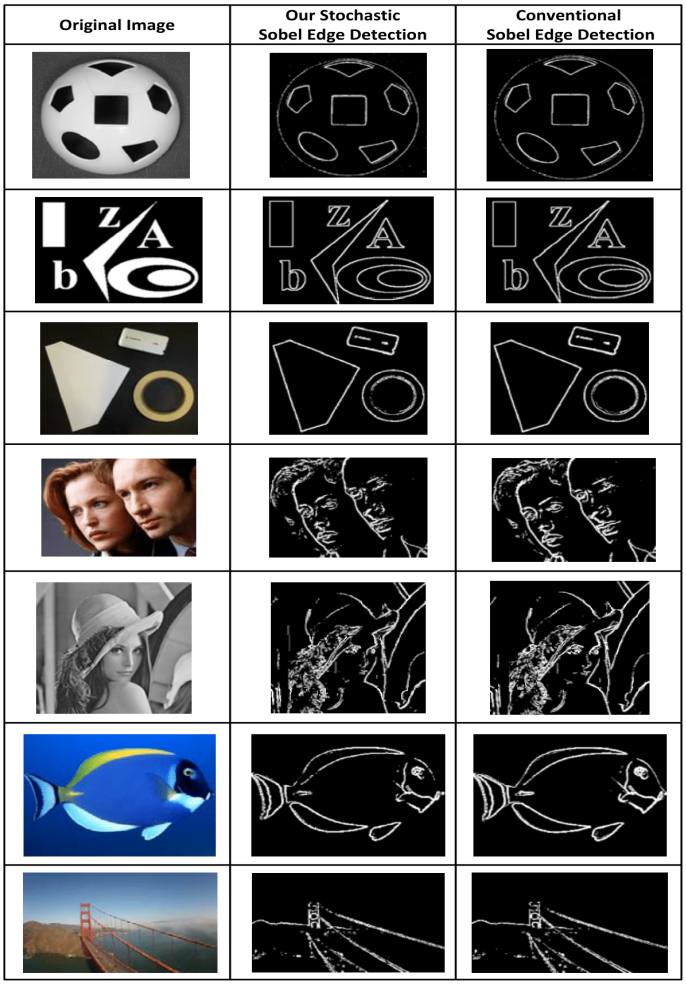


Fig. 12. The figure shows the original image and the simulation outcomes of Sobel edge detection derived using the presented new stochastic architecture and the conventional binary computing.

Table 2. Result of Fault Tolerance Testing toward Internal Circuit Bit-Flip Errors over Several Degree of Pixel Error Rate

Pixel Error Rate	**5%**				**15%**				**30%**				**50%**			
Bit-Flip Rate	**6%**	**12%**	**25%**	**50%**	**6%**	**6%**	**25%**	**50%**	**6%**	**12%**	**25%**	**50%**	**6%**	**12%**	**25%**	**50%**
Our Work	0.001414	0.001919	0.002525	0.00030	0.00422	0.00056	0.00077	0.00086	0.00082	0.00113	0.00151	0.00172	0.00137	0.00187	0.00247	0.00290
Conventional Sobel	2.714887	5.106801	9.160722	13.2	8.071919	15.3373	28.2754	38.2658	16.8048	31.3242	55.1058	75.8483	27.8181	52.0330	90.3886	127.8882

In the second testing, the susceptibility of the new stochastic Sobel edge detection design toward noisy images is tested and the performance is compared with conventional computing. This testing is performed through simulation using an image with additive zero-mean Gaussian noise of various variance value. The output images from the testing are depicted in Fig. 14 and the average output error (using (10)) are summarized in Table 3.

Table 3. Image Error-Ratio Obtained from Simulations Using Image with Additive Zero-Mean Gaussian Noise of Different Variance Values (also Refer to Fig. 14)

Noise Variance	0.008	0.020	0.050	0.080
Our Work	0.0252	0.0352	0.0510	0.0621
Conventional Sobel	0.4048	0.6082	0.8926	1.0852

Based on the result obtained in both testing, it is proven that the proposed stochastic Sobel edge detection has high fault tolerance compared to its conventional counterpart. In both cases, where the noise injection rate increases (in both the internal circuits and input cases), the conventional Sobel edge detection degrades rapidly. In the worst case scenario, the output image is no longer recognizable. On the contrary, our architecture is less susceptible towards noise and showing consistent accuracy level even as the error rate increases.

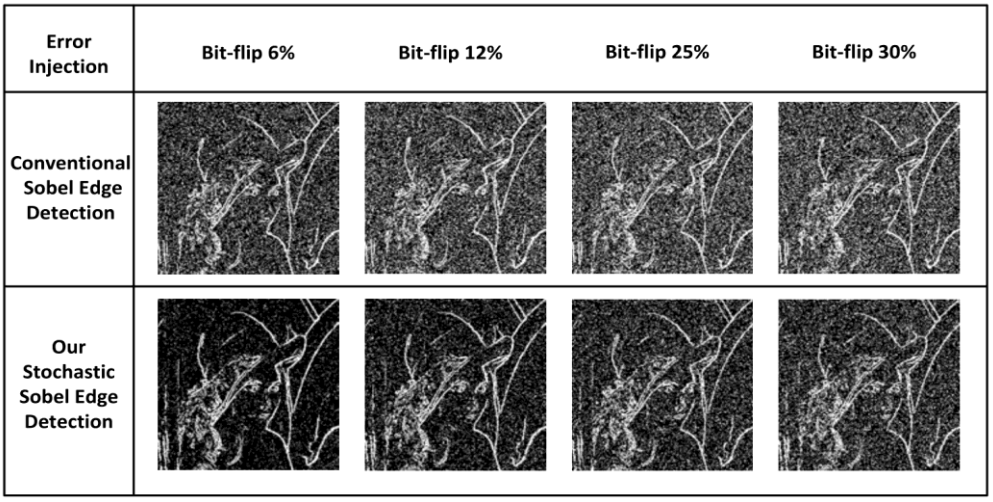


Fig. 13. Output images obtained from simulation using our new architecture and the conventional Sobel edge detection corresponding to pixel error rate of 30% and bit-flip error rate of 6%, 12%, 25% and 50%.

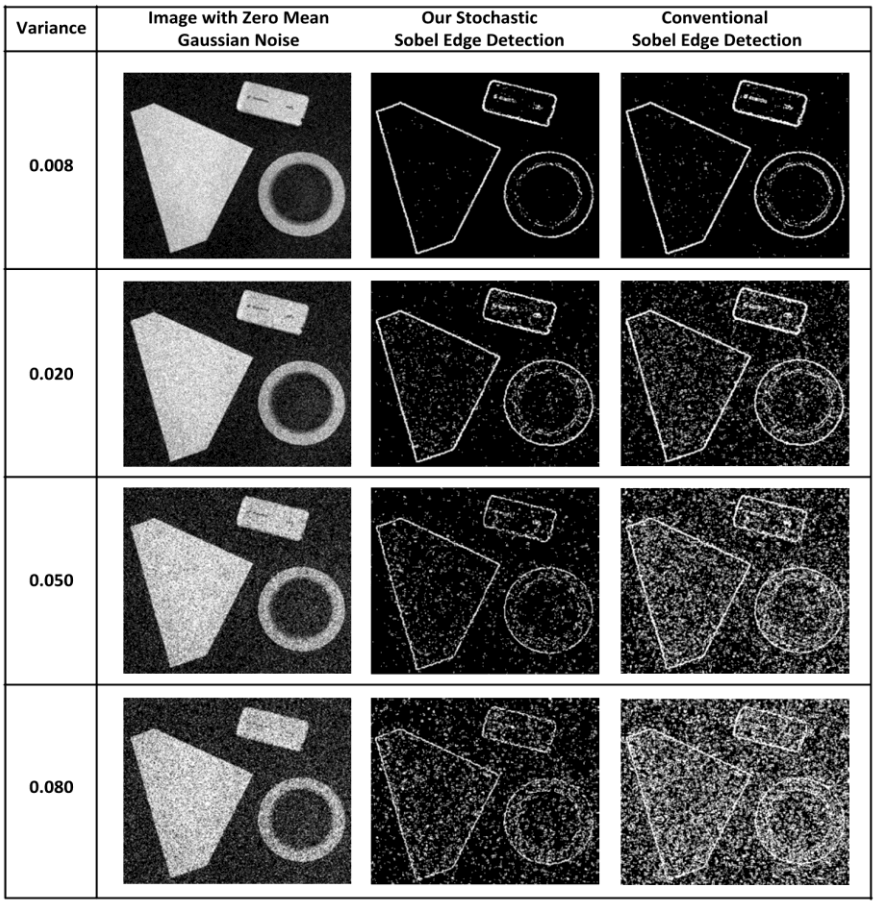


Fig. 14. The figure shows the original image with additive zero mean Gaussian noise of different variance values and the simulation outcomes from Sobel edge detection derived using the presented new stochastic architecture and the conventional binary computing.

7.3. Hardware Complexity

For hardware area evaluation, the proposed stochastic Sobel edge detection was implemented in Cyclone V 5CGXFC7D6F31C6 and synthesized using Quartus II 11.1. Having the architectures clocked at 100MHz, the timing analysis of the architectures is deduced using TimeQuest Timing Analyzer. The power consumption was estimated via PowerPlay Power Analyzer. The full hardware compilation result is summarized in Table 4.

Table 4. Hardware Review for the FPGA Implementation of the Stochastic Sobel Edge Detection.

Hardware Requirement/ Performance	Stochastic Sobel Edge Detection	Absolute Value of Difference	SNG Module
Combinational ALUTs (112,960)	56	59	7
Memory ALUTs (56,480)	0	0	0
Dedicated Logic Register (225,920)	170	39	5
Total Register (225,920)	170	39	5
Fmax (MHz)	213.68	326.05	631.31
Dynamic Thermal Power Dissipation (mW)	4.33	2.26	0.79

8. Conclusion

New and improved stochastic computational functions; stochastic scaled addition and stochastic absolute

difference of value are presented in this study. As opposed to the conventional belief in SC, our design incorporated data correlations to achieve efficient hardware cost reduction without jeopardizing the effectiveness of the original computations. For verification and validation, the proposed stochastic functions are implemented in an image processing system.

As a result, an optimized stochastic Sobel edge detection is designed using the new stochastic scaled addition and stochastic absolute value of difference. Through the scaled-before-addition approach, multipliers are no longer required. Furthermore, the length of the stochastic sequence is reduced significantly. Unlike the previous works, the proposed stochastic absolute value of difference is applicable directly to SC-bipolar format. Despite of the hardware reduction, the experimental results have proven that the accuracy level of our design is on par with its conventional counterpart. Not only that, based on two fault tolerant testing, it is clear that our architecture is less susceptible towards soft errors insertions in the internal circuit and also towards noisy images.

In conclusion, dedicated design of the stochastic computational elements enables efficient implementation of high fault tolerance stochastic Sobel edge detection on hardware without an excessive trade-off between the computational bit-length and the output accuracy level.

References

[1] Gaines, B. R. (1967). Stochastic computing. *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference* (pp. 149–156). New York, NY, USA: AFIPS '67 (Spring), ACM.

[2] Alaghi, A., & Hayes, J. P. (2013). Survey of stochastic computing. *ACM Trans. Embed. Comput. Syst., 12(2s)*, 92:1–92:19.

[3] Qian, W., & Riedel, M. D. (2008). The synthesis of robust polynomial arithmetic with stochastic logic. *Proceedings of 45th ACM/IEEE Design Automation Conf.* (pp. 648-653).

[4] Moons, B., & Verhelst, M. (2014). Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 4(4)*,475 – 486.

[5] Brown, B. D., & Card, H. C. (2001). Stochastic neural computation. i. computational elements. *IEEE Transactions on Computers, 50(9),* 891–905.

[6] Dinu, A., Cirstea, M. N., & McCormick, M. (202). Stochastic implementation of motor controllers. *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics, 2002, ISIE: Vol. 2,* (pp. 639–644).

[7] Qian, W., Li, X., Riedel, M. D., Bazargan, K., & Lilja, D. (2011). An architecture for fault-tolerant computation with stochastic logic. *IEEE Transactions on Computers, 60(1),* 93–105.

[8] Alaghi, A., Li, C., & Hayes, J. P. (2013). Stochastic circuits for real-time image-processing applications. *Proceedings of 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)* (pp. 1–6).

[9] Li, P., & Lilja, D. J. (2011). Using stochastic computing to implement digital image processing algorithms. *Proceedings of 2011 IEEE 29th International Conference on Computer Design (ICCD)* (pp. 154–161).

[10] Naderi, A., Mannor, S., Sawan, M., & Gross, W. J. (2011). Delayed stochastic decoding of LDPC codes. *IEEE Transactions on Signal Processing, 59(11),* 5617–5626.

[11] Chang, Y. N., & Parhi, K. K. (May 2013). Architectures for digital filters using stochastic computing. *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2697–2701).

[12] Parhi, K. K., & Liu, Y. (June 2014). Architectures for IIR digital filters using stochastic computing. *Proceedings of 2014 IEEE International Symposium on in Circuits and Systems (ISCAS)* (pp. 373–376).

[13] Liu, Y., & Parhi, K. K. (April 2015). Lattice fir digital filter architectures using stochastic computing.

Proceedings of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1027–1031).

[14] Saraf, N., Bazargan, K., Lilja, D. J., & Riedel, M. D. (March 2014). IIR filters using stochastic arithmetic. *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE)* (pp. 1–6).

[15] Parhi, M., Riedel, M. D., & Parhi, K. K. (July 2015). Effect of bit-level correlation in stochastic computing. *Proceedings of 2015 IEEE International Conference on Digital Signal Processing (DSP)* (pp. 463–467).

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

[17] Ranjbar, M., Salehi, M. E., & Najafi, M. H. (May 2015). Using stochastic architectures for edge detection algorithms. *Proceedings of 2015 23rd Iranian Conference on Electrical Engineering (ICEE)* (pp. 723–728).

Ming Ming Wong received her B.E. degree in computer systems engineering from Curtin University of Technology (Sarawak Campus) in 2008. After her first degree, she joined Faculty of Engineering, Computing and Science (FECS) at Swinburne University of Technology (Sarawak Campus) as a Ph.D. research student. She completed her Ph.D in July 2012 and she is currently a lecturer in the same institution. Her research interests include VLSI design for cryptology application, digital signal processing and hardware description and implementation.

Dennis M. L. Wong received his BEng(Hons) in electronic and communication engineering and Ph.D in signal processing from the Department of Electrical Engineering and Electronics in University of Liverpool, Liverpool, UK.

In 2004, Dr. Wong joined the School of Engineering, Swinburne University of Technology Sarawak Campus as a lecturer. Subsequently, he was appointed a senior lecturer in 2007 at the same institution. From 2011 to mid-2012, he was an associate professor at Xi'an Jiaotong Liverpool University, Suzhou, China. From June 2012 to Sept 2016, he was the dean, Faculty of Engineering, Computing and Science at Swinburne Sarawak. Since October 2016, professor Wong has been appointed as a professor and deputy provost for Heriot-Watt University Malaysia. Professor Wong is also a fellow of IET, a fellow and chartered professional engineer with IEAust, a chartered engineer with Engineering Council UK; and a senior member of IEEE. In 2014, he was elected as the inaugural treasurer of the IEEE CIS Malaysia Chapter and in 2015 he was elected as the vice chair for the chapter. In 2016, he was elected as the inaugural chair for IEEE Malaysia Sarawak Subsection. His research interests include statistical signal processing and pattern classification, machine condition monitoring, and VLSI for digital signal processing.

Cishen Zhang received the B.Eng. degree in computer engineering from Tsinghua University, China, in 1982 and Ph.D. degree in electrical engineering from Newcastle University, Australia, in 1990.

He was an electrician with ChangXinDian (February Seven) Locomotive Manufactory, Beijing, China during 1970-1978, carried out research on control systems during 1983-1985 at Delft University of Technology, the Netherlands, was with the Department of Electrical and Electronic Engineering at the University of Melbourne, Australia, as a lecturer, senior lecturer, and associate professor and reader in 1989-2002 and with the School of Electrical and Electronic Engineering and School of Chemical and Biomedical Engineering during 2002-2010 at

Nanyang Technological University, Singapore. Since 2010, he has been the professor of electrical and electronic engineering with the School of Software and Electrical Engineering at Swinburne University of Technology in Melbourne, Australia. His research interests include control, signal processing and medical imaging.

Ismat Hijazin is the quality education and academic accreditation director of the Faculty of Science, Engineering and Technology at Swinburne University of Technology in Melbourne Australia. In addition, he is also the program coordinator for electrical and electronic engineering. He completed his undergraduate and postgraduate studies at Bradley University, Peoria Illinois all in Electrical Engineering. Ismat Hijazin lectures and supervises students in electrical engineering at Swinburne. His research interests include error control coding, VLSI digital signal processing and hardware description and implementation.