Journal of Circuits, Systems, and Computers

World Scientific
www.worldscientific.com

A Low-Cost and Fault-Tolerant Stochastic Architecture for The Bernsen Algorithm Using Bitstream Correlation

Shaowei Wang[†], Guangjun Xie[‡], Wenbing Xu[§], Yongqiang Zhang[‖]

School of Microelectronics,
Hefei University of Technology,
Hefei, Anhui, 230009, P.R. China
[†]*2019010121@mail.hfut.edu.cn*
[‡]*gjxie8005@hfut.edu.cn*
[§]*wbxu@mail.hfut.edu.cn*
[‖]*ahzhangyq@hfut.edu.cn*

Jie Han

Department of Electrical and Computer Engineering,
University of Alberta, Edmonton,
Edmonton, AB, T6G1H9, Canada
jhan8@ualberta.ca

Received (25 March 2022)
Revised (7 September 2023)
Accepted (23 November 2023)

Many algorithms for image processing do not require particularly high precision, but they rely on complicated arithmetic operations for every pixel in an image. The Bernsen algorithm is a typical local thresholding algorithm for solving the problem of uneven lighting. However, this algorithm requires a significant computing overhead and is extremely sensitive to noise. In this work, two stochastic computing architectures are proposed for implementing the Bernsen algorithm by using, respectively, uncorrelated and correlated input bitstreams. Experimental results show that both designs, especially the one using correlated bitstreams, present high fault tolerance of soft errors and low hardware cost in comparison with its conventional binary implementation. However, SC logic with uncorrelated inputs is not always superior to its corresponding binary circuit in energy consumption, especially the circuit that needs long input bitstreams. That means that a reasonable use of correlation can further optimize the SC circuit design.

Keywords Bernsen algorithm, fault-tolerance, hardware cost, stochastic computing.

1. Introduction

In digital image processing, binarization algorithms significantly reduce the amount of data in an image, so the contour can readily be highlighted.[1] Commonly used methods in image binarization algorithms are divided into global and local thresholding ones. The former is used when the target and background in an image are clearly separated, such as the Otsu algorithm and the average grey thresholding method;[2] otherwise, the latter is usually

applied, such as the Bernsen algorithm[3] and the Niblick algorithm.[4] Generally, a global binarization algorithm is faster than a local one, although it results in inferior image qualities to the latter.

Among these algorithms, the Bernsen algorithm is a typical local thresholding algorithm for solving the problem of uneven illumination, thus an important approach to image binarization.[5] Since this algorithm requires complex computations for each pixel in a target image, the physical implementation of this algorithm results in a significant hardware cost. Meanwhile, the Bernsen algorithm is sensitive to noises, which can lead to a decrease in the accuracy of the output results. The noises include those caused by soft errors, triggered by the environment, or generated by voltage and thermal fluctuations, which may make the computation results deviate from the expected ones.[6, 7] These issues above for the Bernsen algorithm can be potentially addressed by Stochastic Computing (SC). The main advantages of an SC-based design are low hardware cost and high fault tolerance for soft errors.[8] The principle of SC uses a proportion of 1s in a stochastic bitstream to encode a target value[9], so that a value in the numerical domain is mapped to the probabilistic domain, making it an approximate computation.[10]

Compared with a conventional binary number, each bit in a stochastic bitstream has the same weight. Noises caused by bit flipping in a bitstream have a negligible impact on the computation accuracy, so SC improves the fault tolerance of circuits.[11] Furthermore, SC uses simpler elements to implement complex operations than conventional binary circuits. This feature greatly reduces the hardware and power consumption in stochastic circuits. An architecture with low-power and fault-tolerance has been presented in SC for the kernel density estimation-based image segmentation algorithm.[12] This work demonstrates the significant advantages of SC architecture in fault tolerance. A low-power and fault-tolerant stochastic architecture has been designed for the Sauvola algorithm.[13] The literature designs an SC structure that can efficiently compute the mean of multiple inputs, while highlighting the low hardware overhead characteristics of SC circuits compared to traditional binary circuits. We have developed the deterministic Halton sequence (DHS)-based stochastic number generators for the Bernsen algorithm.[14] However, this method is not sufficient enough to highlight the strengths of SC in image processing. A hybrid bit-splitting generator has been designed to produce parallel stochastic bitstreams to reduce delay, but the application of parallel technology in neural network applications results in a significant hardware overhead.[15] A novel stochastic number generator has been presented to optimize stochastic multipliers for implementing the Izhikevich spiking neuron model.[16] However, the adoption of the Omega-flip structure results in a higher hardware cost. In the relevant literature above, the SC circuits designed require the use of uncorrelated input bitstreams. In this paper, the efficiency of bitstream correlation in this aspect through implementing the Bernsen algorithm in SC is comprehensively explored, by comparing two scenarios.

In this paper, we propose two stochastic circuits for the Bernsen algorithm. The main contributions include 1. High-accuracy and low-cost stochastic circuits using, respectively, uncorrelated and correlated stochastic input bitstreams are proposed. The impacts of LD

sequence and LFSR on the accuracy, hardware overhead, and fault tolerance of Bernsen circuits under correlated and uncorrelated logic are analyzed. The experimental results verify that the reasonable use of correlation can greatly improve the performance of SC circuits. 2. A decoder that can simplify the proposed stochastic Bernsen circuits is designed. The proposed decoder can directly convert the logical value into binary numbers, and there is no need to convert to stochastic bitstreams. This paper proceeds as follows. Sec. 2 introduces the background for the Bernsen algorithm and stochastic computing. Sec. 3 presents the proposed stochastic circuits for the Bernsen algorithm. Sec. 4 reports the experiments and results. Sec. 5 concludes this paper.

2. Background

2.1. Bernsen Algorithm

In the local thresholding Bernsen algorithm, for a pixel value indexed by its coordinate (i,j), $f(i,j)$, the mean value or threshold $T(i,j)$, is computed by considering the maximum and minimum pixel values in a $(2k+1)\times(2k+1)$ window centered on $f(i,j)$ as

$$T(i,j)=\frac{\max\limits_{-k\le x,y\le k} f(i+x,j+y)+\min\limits_{-k\le x,y\le k} f(i+x,j+y)}{2}, \tag{1}$$

where k is a positive integer as the radius of the selected window.

The maximum and minimum pixel values in a selected window are denoted as M and N respectively, and A is the difference between M and N. The parameters S and tt are thresholds to control the gray level difference in the selected window. The values of these two parameters are not set in stone, different values can accordingly be selected according to the specific required output results. If $A\ge S$, it means that the gray level in the selected window has a large difference, thus a further comparison between $f(i,j)$ and $T(i,j)$ is required. Otherwise, tt is directly compared with $T(i,j)$. Fig. 1 shows a schematic for computing the mean value $T(i,j)$ and the difference A for the Bernsen algorithm.

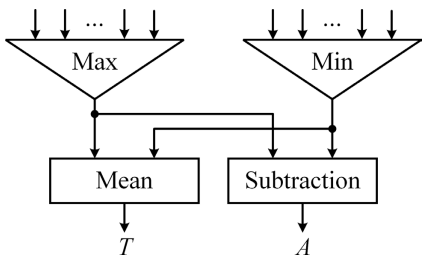

Fig. 1. A schematic for computing the mean value and the difference.

2.2. Stochastic Computing

Two encoding methods are widely used in SC, the unipolar and bipolar formats.[8] In the unipolar representation, the probability of 1s occurring in a bitstream represents a real value x; for example, a bitstream Sx=00010111 encodes $x=P(Sx)=4/8=0.5$. The position of 1s in a bitstream is not fixed, which allows a value to have different bitstream forms. Bitstream Sx'=11100001 can also represent 0.5. Numbers in the unipolar representation range in [0,1], while they range in [−1,+1] in the bipolar representation. A real value in the bipolar representation is interpreted as $x=2P(Sx)-1$, so the same bitstream Sx=00010111 encodes $x=2P(Sx)-1=0$. Thus, both positive and negative real values can be encoded in the bipolar representation.[8] In this paper, x is indiscriminately used to represent both a real value and the bitstream for representing it.

Simple logic gates using stochastic bitstreams enable low-cost circuits to implement complex arithmetic functions. However, for a logic gate, the correlation between bitstreams that represent the same values could make it present different logical functions.[17] This property is called Stochastic Computing Correlation. Intuitively speaking, if the 1s of bitstreams overlap with each other to the greatest extent, then these bitstreams can be referred to as having the maximal positive correlation, and if the 1s of bitstreams almost do not overlap with each other, they can be referred to as having the maximal negative correlation. If the 1s of bitstreams randomly overlap with each other, then they are ideal stochastic bitstreams, or they can be called uncorrelated bitstreams.

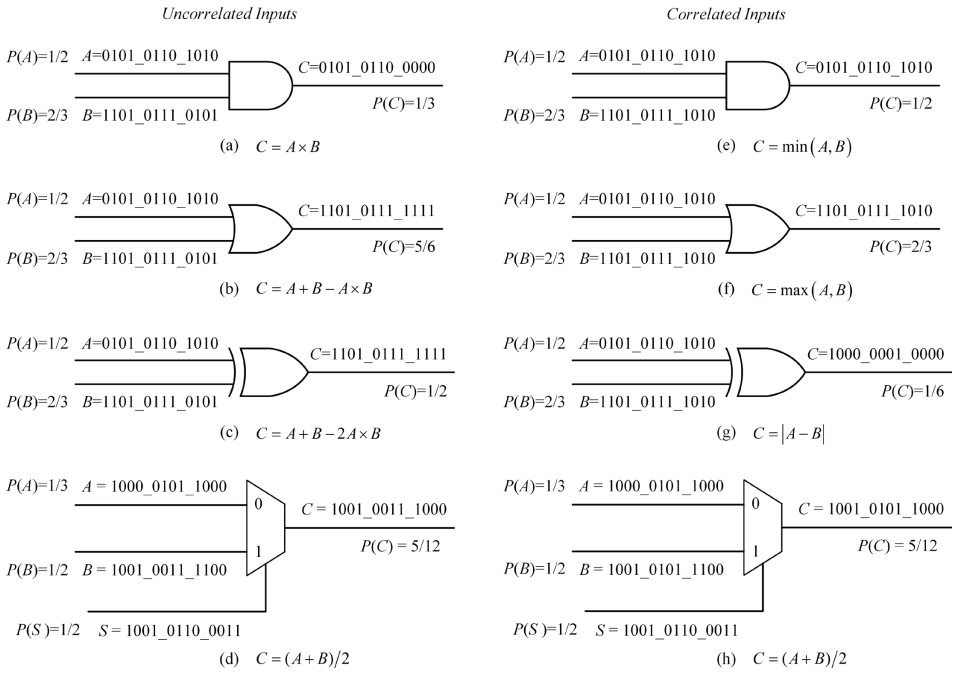

Fig. 2. Stochastic elements with different input bitstreams.

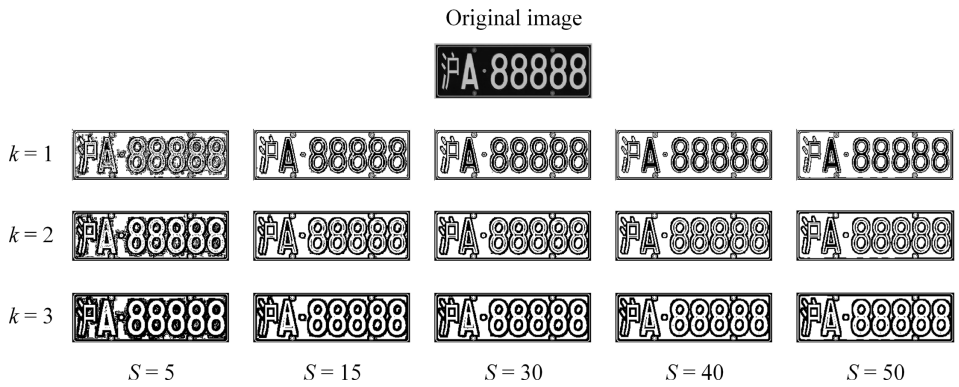

Fig. 3. The binarization results under different Bernsen parameters.

In unipolar representation, while two input bitstreams are uncorrelated, an AND gate performs as a stochastic multiplier. For example, two input bitstreams A=01101001 and B=11011011 respectively represent values $P(A)$=4/8, $P(B)$=6/8, and its output bitstream is C=01001001 representing value $P(C)$=3/8, so satisfying $C=A\times B$. An OR gate can achieve the function of $C=A+B-A\times B$. An XOR gate implements the function $C=A+B-2A\times B$. While two inputs are maximally correlated, an AND gate can achieve the function of calculating the minimum value, an OR gate can achieve the function of calculating the maximum value, and an XOR gate can be used as an absolute value subtractor.[18] A multiplexer or MUX with a select signal set to ½ realizes the function of mixing two stochastic bitstreams, resulting in a scaled adder.[8] The inputs of MUX can be correlated or not, while the select bitstream must be uncorrelated with the inputs. It is worth noting that the MUX completes an addition operation of inputs A and B for both unipolar and bipolar representations. The corresponding examples are shown in Fig. 2.

3. The proposed circuits for Bernsen algorithm

In this section, two stochastic architectures are proposed for implementing the Bernsen algorithm by respectively using uncorrelated and correlated input bitstreams. For an image, it is necessary to normalize all pixels ranging from [0, 255] to [0, 1] when implementing the algorithm in SC.

3.1. Bernsen Parameters

Three important parameters should be preset to implement the Bernsen algorithm: the parameters S and tt, and the selected window size $(2k+1)\times(2k+1)$. Parameter tt adopts the preset value of 128 in the Bernsen algorithm. The parameter S and the selected window size $(2k+1)\times(2k+1)$ will have varying degrees of impact on the results. For the Bernsen algorithm, the window size has a significant impact on results: the larger the selected window size is, the higher the accuracy. If the selected window size is too large, however, it may not further improve the quality of processed images and will cause additional costs.[13] The results are shown in Fig. 3 for the Bernsen algorithm processing gray-scale images

under different window sizes. Experiments show that the algorithm using a 5×5 window size achieves acceptable binarized results and requires modest computation. Thus, a 5×5 window size is adopted in this paper, by comprehensively considering the computing accuracy and computation cost.

3.2. Design Process

According to the Bernsen algorithm described in Sec. 2 and the selected window size described above, the stochastic architectures are realized in the following three steps,

1. Computing the maximum and minimum pixel values M and N in a selected window, respectively. The adopted window **F** is shown in Fig. 4. Thus, the maximum pixel value can be written as,

$$M = \max\{\mathbf{F}\}. \tag{2}$$

The minimum pixel value can be written as,

$$N = \min\{\mathbf{F}\}. \tag{3}$$

2. Computing the mean T and the difference A between the maximum and minimum pixel values, respectively. The mean value can be computed as,

$$T = \frac{M+N}{2}. \tag{4}$$

The difference value can be computed as,

$$A = M - N. \tag{5}$$

$f(i\text{-}2, j\text{+}2)$	$f(i\text{-}1, j\text{+}2)$	$f(i, j\text{+}2)$	$f(i\text{+}1, j\text{+}2)$	$f(i\text{+}2, j\text{+}2)$
$f(i\text{-}2, j\text{+}1)$	$f(i\text{-}1, j\text{+}1)$	$f(i, j\text{+}1)$	$f(i\text{+}1, j\text{+}1)$	$f(i\text{+}2, j\text{+}1)$
$f(i\text{-}2, j)$	$f(i\text{-}1, j)$	$f(i, j)$	$f(i\text{+}1, j)$	$f(i\text{+}2, j)$
$f(i\text{-}2, j\text{-}1)$	$f(i\text{-}1, j\text{-}1)$	$f(i, j\text{-}1)$	$f(i\text{+}1, j\text{-}1)$	$f(i\text{+}2, j\text{-}1)$
$f(i\text{-}2, j\text{-}2)$	$f(i\text{-}1, j\text{-}2)$	$f(i, j\text{-}2)$	$f(i\text{+}1, j\text{-}2)$	$f(i\text{+}2, j\text{-}2)$

Fig. 4. The diagram of a selected window **F**.

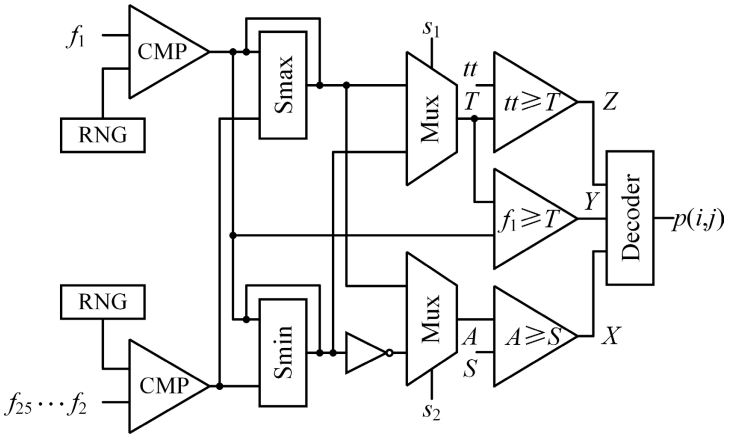

Fig. 5. The proposed stochastic circuit using uncorrelated input bitstreams for the Bernsen algorithm.

3. Generating output pixel values.

These three computation steps involve maximum, minimum, addition, and subtraction operations, respectively. Based on the computation steps in subsection 3.2 and the parameters identified in subsection 3.3, we design the SC circuits for the Bernsen algorithm in the next.

3.3. The Proposed Circuit using Uncorrelated Bitstreams

We propose a stochastic circuit for implementing the Bernsen algorithm by using uncorrelated input bitstreams. Fig. 5 shows the proposed stochastic circuit. Independent stochastic number generators (SNG) are used to generate uncorrelated bitstreams. The SNG consists of a random number generator (RNG) and a comparator (CMP). The linear feedback shift register (LFSR) has the advantage of low hardware complexity, making it an extensively used RNG. The Low-discrepancy bitstreams based on Sobol sequence[19] and Halton sequence[20] have been introduced for SC design. These three different random sequences are used to generate bitstreams in this paper.

3.3.1. Maximum and Minimum

Because a 5×5 window size is selected, 25 input bitstreams are needed to compute M and N. A stochastic max (Smax) function circuit[21] that is composed of an XOR gate, a multiplexer, and a stochastic Tanh structure can compute the max one in two uncorrelated bitstreams. This circuit is used in the proposed stochastic Bernsen structure. A Smax has two inputs, one of which is connected to pixel $f(i,j)$, and the other is connected to the other 24 pixels in sequence so that M in a selected window can be computed. Pixel $f(i,j)$ is conveniently denoted as f_1, and the remaining 24 pixels are represented as $f_2, f_3 \ldots f_{25}$ in turn. A Smin structure for computing N is obtained by changing the connection of 0 and 1 in the MUX in the stochastic Tanh structure.

3.3.2. Mean and Difference

The mean T and the difference A between the maximum and minimum pixel values in a selected window are respectively computed. A MUX with a fixed select input ½ can compute the mean of two bitstreams, such as the mean of M and N is $(M+N)/2$. The difference A relates to a stochastic subtraction that requires a NOT gate to implement negation in bipolar representation. A MUX with a logically inverted input can realize the difference between the maximum and minimum, as $(M-N)/2$, and bipolar format representation is needed here. A MUX-based subtractor cannot directly compute the difference between M and N, thus a scaled version result is adopted here.

3.3.3. Generating Output Pixel Values

This process needs to be implemented with a stochastic comparator. A stochastic comparator[22] produces less accurate bitstreams approximately representing 0 or 1, and then the bitstreams need to be converted to binary numbers. A comparator[13] has the ability to produce more accurate results, which the results also need to be converted into binary numbers. We propose a method that can directly output real binary pixel values, by using the two preset parameters S and tt and considering the principle of the Bernsen algorithm. Suppose the length of input bitstreams is 255, set S and tt to be 40 and 128 as discussed previously. By mapping them to [0,1], they become 40/255 and 128/255, then multiplying the length of the input bitstreams, so 40 and 128 1s can respectively be counted. If the difference A is larger than S, that is, the number of 1s in the bitstream A is greater than 40, the comparison result X is logically true, as shown in Fig. 5. To compare f_1 and T, up/down counter is used. The result Y indicates whether f_1 is larger than T or not. Similarly, tt and T are also compared in the same way. Therefore, the comparison results are determined by the logical values of X, Y, and Z. Table 1 shows the truth table for the output pixel value according to X, Y, Z. The proposed method directly converts the logical value into binary numbers, and there is no need to convert to stochastic bitstreams. Compared with the two stochastic comparators above, the proposed one is easy to be designed and significantly lowers the hardware cost of the circuit.

Table 1. The truth table of output pixel value obtained by X, Y, Z

X	*Y*	*Z*	*Output*
1	1	x	1
1	0	x	0
0	x	1	1
0	x	0	0

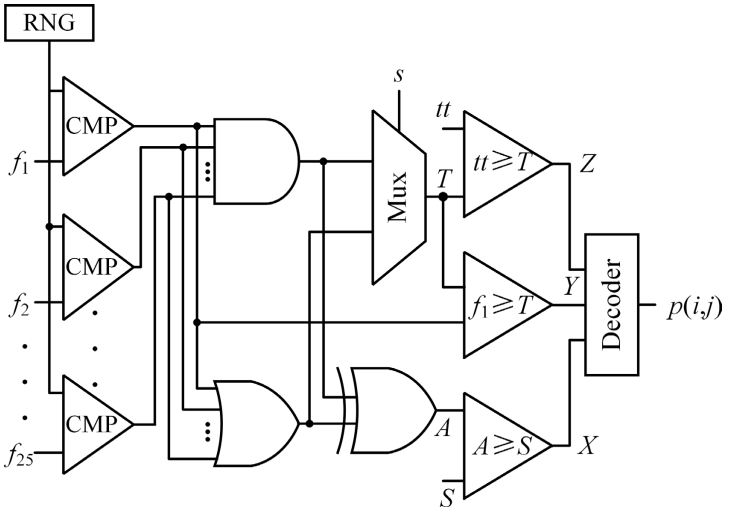

Fig. 6. The proposed stochastic circuit using correlated input bitstreams for the Bernsen algorithm.

3.4. The Proposed Circuit using Correlated Bitstreams

The correlation between stochastic bitstreams will affect the accuracy of stochastic circuits so that most stochastic circuits require uncorrelated bitstreams to perform computation. However, the appropriate use of correlation enables stochastic basic units to realize new logic functions, which can further simplify the structures of stochastic circuits. Therefore, in this subsection, we propose a stochastic architecture for the Bernsen algorithm by using correlated bitstreams as shown in Fig. 6. By directly sharing an RNG, bitstreams with the maximal correlation can be obtained, which can further reduce the hardware cost of the stochastic circuit.

3.4.1. Maximum and Minimum

A stochastic AND gate serving as a multiplier reduces its hardware cost and simplifies logic, compared with traditional methods to realize a multiplier. The prerequisite for that is that its input bitstreams must be highly uncorrelated. An AND gate using correlated inputs will result in the minimum among two inputs. An OR gate using correlated bitstreams can get the maximum among two inputs. Hence, the maximum and minimum pixel values M and N in a selected window can be computed by using simple logic gates in the proposed stochastic circuit.

3.4.2. Mean and Difference

The mean T and the difference A between M and N are computed in this step. Similarly, a MUX can implement the function of averaging M and N. It should be noted that, although input bitstreams are generated by sharing an RNG directly in this method, a separate RNG is provided for generating the select bitstream to ensure the MUX accurately computes the

average of two bitstreams. An XOR gate can realize the function $C=|A-B|$ when the input bitstreams have the maximum correlation. Therefore, an XOR gate is used as an absolute value subtractor to compute the difference A between M and N in this method to implement the Bernsen algorithm.

3.4.3. Generating Output Pixel Values

The method to generate output pixel values is the same as the proposed one using uncorrelated bitstreams.

In summary, the output bitstreams processed by the AND gate and the OR gate still are the original bitstreams generated by an RNG, so they have the maximum correlation. Therefore, the XOR gate also accurately computes results. That is, the entire computing process of this circuit from the initial inputs to the XOR gate is error-free. Consequently, the Bernsen algorithm is suitable to be implemented by using correlated bitstreams.

4. Experiment and results

In this section, the performance of the proposed circuits will be compared and described through different dimensions such as accuracy, hardware resource consumption, and fault tolerance. Accuracy is described by computing the error rate of the circuit output results. The hardware resource consumption is evaluated by comparing the area, power consumption, latency, and other data of Bernsen circuits implemented in different ways. The fault tolerance performance of the circuits is proved by detecting its error rate under input noises. MATLAB and Verilog hardware description languages are used to simulate and evaluate the accuracy of the proposed stochastic circuits. All circuits are synthesized by Synopsys Design Compiler and TSMC 40nm standard cell library at a frequency of 100MHz and a typical corner.

Table 2. The accuracy of the conventional and stochastic circuits for the Bernsen algorithm under different bitstream lengths (%)

	Length		4-bit	5-bit	6-bit	7-bit	8-bit
	Conventional		30.85	29.68	9.28	3.72	0
SC	Length		256-bit	512-bit	1024-bit	2048-bit	4096-bit
	Uncorrelated	LFSR	21.90	13.52	6.97	3.93	2.20
		Halton	21.74	13.50	6.91	3.89	2.19
		Sobol	21.62	13.07	6.20	3.78	2.19
	Length		8-bit	16-bit	32-bit	64-bit	128-bit
	Correlated	LFSR	5.50	2.36	1.12	0.62	0.32
		Halton	4.38	2.17	1.12	0.52	0.31
		Sobol	4.33	2.01	0.71	0.38	0.15

4.1. Accuracy

To compare the accuracy of the proposed stochastic circuits, the average output error rate is computed as

$$Error = \sum_{i=1}^{H}\sum_{j=1}^{W}\frac{\left|T_{i,j} - E_{i,j}\right|}{H \cdot W} \times 100\%. \tag{6}$$

where H and W represent the height and width of an image to be processed, T and E are the theoretical results and corresponding experiment results, respectively. The conventional circuit is simulated using 4-, 5-, 6-, 7-, 8-bit binary numbers. For stochastic circuits, LFSRs will cause fluctuating results, so 1000 trials of the Monte Carlo experiment are simulated and then their average is recorded. Table 2 shows the accuracy of the proposed stochastic architectures under various lengths of bitstreams.

For most digital processing algorithms, errors less than 5% that would not be observed by human eyes can be tolerated up for computation results.[23] The data in Table 2 indicates that the minimum precision for the conventional binary approach to achieve the acceptable error rate is 7-bit. The stochastic design using uncorrelated inputs requires 2048-bitstreams to make the output results acceptable. Meanwhile, the one using correlated inputs generated with 8-bit streams can produce almost acceptable results.

The main reason for this is that the proposed stochastic circuits have different structures for calculating the maximum and minimum values for the selected window. Specifically, the uncorrelated method uses a Smax structure to find the maximum and minimum values that require a longer sequence to generate results. The correlated method using an AND gate and an OR gate achieves the same function even with more accurate results by taking advantage of the correlation between bitstreams.

4.2. Hardware Resource Consumption

The proposed stochastic circuits are tested by synthesizing with various lengths of bitstreams, and then compared in terms of their hardware performance to a conventional circuit. Their area, power consumption, critical path delay, and total delay are shown in Table 3. For the stochastic circuit using uncorrelated bitstreams, since acceptable results are obtained when 2048-bit streams are used as input, only experimental results using 2048-bit streams are presented. The results of conventional binary circuits and other proposed stochastic circuits are fully presented with different precisions.

Compared with the conventional 8-bit precision circuit, the LFSR-based stochastic circuit with uncorrelated bitstreams has a 76.2% improvement in area and 75.2% in power. The one with correlated bitstreams has a 96.5% improvement in area, 96.3% in power, and 69.5% in critical path delay when 8-bit LFSR streams are used. When 128-bit LFSRs are used, the circuit has a 91.9% improvement in area, 92.3% in power, and 42.0% in critical path delay. Compared with the LFSR-based design, the Halton- and Sobol-based approaches have higher hardware overhead. The hardware resource consumption of the correlated Sobol-based design is higher than that of the conventional 8-bit precision design.

Table 3. Performance comparison of the proposed stochastic circuits and conventional circuits for the Bernsen algorithm

Method	Length (bit)		Area (um^2)	Power (uW)	Critical path delay (ns)	Delay (ns)	ADP ($um^2 \times ns$)	PDP (10^{-3} pJ)
Conventional	4		1368.51	116.64	0.94	0.94	1286.40	109.64
	5		1745.12	137.68	0.95	0.95	1657.86	130.80
	6		2127.91	161.18	1.05	1.05	2234.31	169.24
	7		2490.42	180.39	1.13	1.13	2814.17	203.84
	8		2865.27	196.24	1.31	1.31	3,753.50	257.07
Uncorrelated	LFSR	2048	680.73	48.69	1.64	$24\times1.64\times2^{11}$	54873155.28	3924865.84
	Halton	2048	2105.51	117.11	2.45	$24\times2.45\times2^{11}$	253550567.42	14102667.26
	Sobol	2048	3233.59	349.20	2.85	$24\times2.85\times2^{11}$	452971634.69	48917053.44
Correlated	LFSR	8	101.07	7.30	0.40	0.40×2^{3}	323.42	23.36
		16	134.24	9.33	0.48	0.48×2^{4}	1030.96	71.65
		32	166.17	11.90	0.58	0.58×2^{5}	3084.12	220.86
		64	206.04	13.83	0.65	0.65×2^{6}	8571.26	575.33
		128	231.96	15.13	0.76	0.76×2^{7}	22565.07	1471.85
	Halton	8	289.83	31.72	0.67	0.67×2^{3}	1553.49	170.02
		16	400.43	34.99	0.78	0.78×2^{4}	4997.37	436.68
		32	551.78	39.61	1.05	1.05×2^{5}	18539.81	1330.90
		64	637.33	41.38	1.19	1.19×2^{6}	48539.05	3151.50
		128	821.14	45.81	1.45	1.45×2^{7}	152403.58	8502.34
	Sobol	8	339.46	37.81	0.67	0.67×2^{3}	1819.51	202.66
		16	491.98	49.37	0.79	0.79×2^{4}	6218.63	624.04
		32	683.55	63.12	0.89	0.89×2^{5}	19467.50	1797.66
		64	886.23	79.94	1.23	1.23×2^{6}	69764.03	6292.88
		128	1102.15	98.50	1.74	1.74×2^{7}	245470.85	21937.92

Table 4. The average output error of the proposed and conventional circuits versus injected noise levels (%)

Noise Levels	Length	0	2	5	10	15	20	30
Conventional	-	0.0	26.19	35.70	38.19	39.39	40.88	43.09
Uncorrelated	2048-bit	3.9	4.1	4.2	4.3	4.7	6.6	14.1
Correlated	8-bit	5.5	5.5	5.5	9.9	10.2	10.3	25.6
	16-bit	2.4	2.4	4.6	6.8	7.1	8.9	20.4
	32-bit	1.1	1.6	4.4	5.0	6.8	7.8	10.6
	64-bit	0.6	1.3	2.7	4.6	5.8	7.2	9.2
	128-bit	0.3	1.0	2.2	4.4	5.6	6.7	8.5

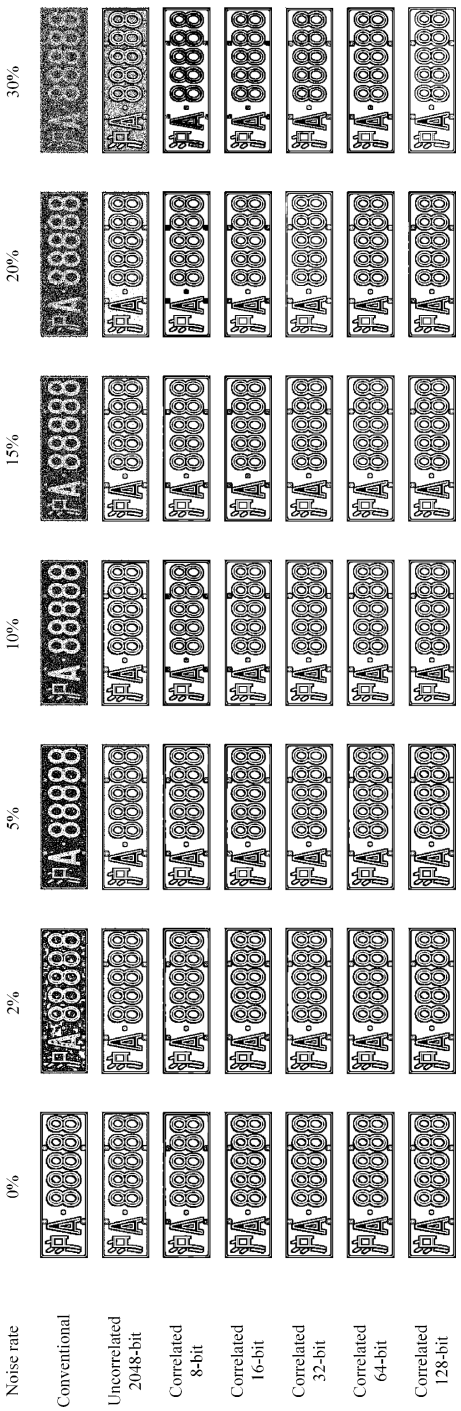

Fig. 7. Fault-tolerance comparison of the proposed stochastic circuits and conventional circuits for the Bernsen algorithm.

Since stochastic circuits require long bitstreams, their delay is larger than that of the conventional circuit. As shown in Table 3, Area Delay Product (ADP), and Power Delay Product (PDP) of the aforementioned three types of circuits are compared. The proposed LFSR-based stochastic circuit using correlated bitstreams enjoys lower ADP, and PDP than the conventional one until 64-bit streams are used. However, the circuit designed with uncorrelated inputs has a much higher energy consumption compared to its corresponding binary circuit. The comparison of the two proposed circuits for the Bernsen algorithms indicates that the correlation should be paid more attention to circuit design. Speaking cautiously, these results, again, illustrate that the circuit designed using correlated bitstreams is more suitable for implementing the Bernsen algorithm.

4.3. Fault Tolerance

Soft errors can be approximated by flipping input bits independently and randomly. For example, if a length of 1000-bitstream is injected for 5% noise, then 50 bits are randomly selected and flipped. By injecting different error rates into input bitstreams in this way to evaluate the fault tolerance of the proposed stochastic architectures. As the data shown in Table 2, the accuracy of the LFSR-based approach is slightly inferior to that of the other two SC approaches. However, it has advantages in hardware overhead. The 8-bit precision conventional design and LFSR-based design are compared, and the processed images are shown in Fig. 7. The results dealing with different noise levels are numerically shown in Table 4. It can be seen that the proposed stochastic circuits have better fault tolerance than the conventional circuit.

5. Conclusion

In this paper, two stochastic architectures are proposed for implementing the Bernsen algorithm, and these circuits are evaluated through hardware implementations and computing accuracy. A new method to generate output pixel values and concurrently convert them to binary values is also proposed to effectively reduce the resource consumption of the proposed stochastic circuits. Experimental results show that the proposed stochastic circuits outperform their conventional binary circuits in terms of area and fault tolerance. Specifically, the one using uncorrelated input bitstreams has low performance in energy efficiency, while the one using correlated input bitstreams produces higher accuracy and lower hardware costs. It demonstrates that simpler and more efficient stochastic circuits can be designed by using correlated bitstreams for certain logic functions. In future work, we will further explore the impact of bitstream correlation on SC.

Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities of China (Grant No. JZ2020HGQA0162, No. JZ2020HGTA0085), and by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Number: RES0048688).

References

1. 1. M. Villegas, V. Romero, and J. Andreu Sanchez, On the modification of binarization algorithms to retain grayscale information for handwritten text recognition, eds. R. Paredes, J. S. Cardoso and X. M. Pardo, 2015), 208-215.
2. 2. Y. Chen, D. Chen, Y. Li, and L. Chen, "Otsu's thresholding method based on gray level-gradient two-dimensional histogram," in 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics, Wuhan, China, 2010, pp. 282-285.
3. 3. J. Bersen, "Dynamic thresholding of grey-level images," in Eighth International Conference on Pattern Recognition, Paris, France, 1986, pp. 1251-5.
4. 4. O. Samorodova, and A. Samorodov, Fast implementation of the Niblack binarization algorithm for microscope image segmentation, *Pattern Recognit Image Anal.* **26** (2016) 548-51.
5. 5. S. Lokhande, and N. Dawande, "A survey on document image binarization techniques," in 1st International Conference on Computing Communication Control and Automation, Pune, India, 2015, pp. 742-746.
6. 6. S. P. J. V. Rani, J. R. L. Jennifer, and P. Sudhanya, Approximate multipliers design using approximate adders for image processing applications, *Journal of Circuits, Systems, and Computers.* **31** (2022) 2250256.
7. 7. Sakali Raghavendra Kumar, P Balasubramanian, Ramesh Reddy, Sreehari Veeramachaneni, and N. M. Sk, Optimized fault-tolerant adder design using error analysis, *Journal of Circuits, Systems, and Computers.* **32** (2023) 2350091.
8. 8. M. Alawad, and M. Lin, Survey of stochastic-based computation paradigms, *IEEE Trans. Emerging Top. Comput.* **7** (2019) 98-114.
9. 9. S. I. Chu, C. L. Wu, T. N. Nguyen, and B. H. Liu, Polynomial computation using unipolar stochastic logic and correlation technique, *IEEE Trans. Comput.* (2021) 1-1.
10. 10. N. Onizawa, and T. Hanyu, Cmos invertible logic: Bidirectional operation based on the probabilistic device model and stochastic computing, *IEEE Nanatechnol. Mag.* **16** (2022) 33-46.
11. 11. A. Alaghi, W. Qian, and J. Hayes, The promise and challenge of stochastic computing, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **37** (2018) 1515-1531.
12. 12. P. Li, and J. Lilja, "A low power fault-tolerance architecture for the kernel density estimation based image segmentation algorithm," in the IEEE International Conference on Application-specific Systems, Architectures and Processors, Santa Monica, CA, USA, 2011, pp. 161-168.
13. 13. M. Najafi, and M. Salehi, A fast fault-tolerant architecture for sauvola local image thresholding algorithm using stochastic computing, *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **24** (2016) 808-812.
14. 14. Z. Lin, G. Xie, W. Xu, J. Han, and Y. Zhang, Accelerating stochastic computing using deterministic Halton sequences, *IEEE Trans. Circuits Syst. II Express Briefs.* **68** (2021) 3351-3355.
15. 15. Y. Zhang, S. Liu, J. Han, Z. Lin, S. Wang, X. Cheng, and G. Xie, An energy-efficient binary-interfaced stochastic multiplier using parallel datapaths, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.* (2023) 1 - 5.
16. 16. M. A. Hedayatpour, M. A. Karami, and J. Shamsi, Implementation of izhikevich neuron based on stochastic computing using a novel inspired omega‐flip stochastic number generator, *International Journal of Circuit Theory and Applications.* (2022)
17. 17. S.-I. Chu, C.-L. Wu, T. N. Nguyen, and B.-H. Liu, Polynomial computation using unipolar stochastic logic and correlation technique, *IEEE Transactions on Computers.* **71** (2021) 1358 - 1373.

18. 18. S. Wang, G. Xie, X. Cheng, and Y. Zhang, Weighted-adder based polynomial computation using correlated unipolar stochastic bitstreams, *IEEE Transactions on Circuits and Systems II: Express Briefs.* **69** (2022) 4528-4532.
19. 19. S. Liu, and J. Han, Toward energy-efficient stochastic circuits using parallel sobol sequences, *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **26** (2018) 1326-1339.
20. 20. A. Alaghi, and J. Hayes, "Fast and accurate computation using stochastic circuits," in the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2014, pp. 1-4.
21. 21. J. Yu, K. Kim, J. Lee, and K. Choi, "Accurate and efficient stochastic computing hardware for convolutional neural networks," in 2017 IEEE International Conference on Computer Design (ICCD), Boston, MA, USA, 2017, pp. 105-112.
22. 22. P. Li, and D. Lilja, "Using stochastic computing to implement digital image processing algorithms," in the 2011 IEEE 29th International Conference on Computer Design (ICCD), Amherst, MA, USA, 2011, pp. 154-161.
23. 23. P. Li, and D. Lilja, "Accelerating the performance of stochastic encoding-based computations by sharing bits in consecutive bit streams," in the Proceedings of the 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors, New York, 2013, pp. 257-260.