

# Design of A Stochastic Computing Architecture for The Phansalkar Algorithm

Yongqiang Zhang, *Member, IEEE*, Jiao Qin, Jie Han, *Senior Member, IEEE*, and Guangjun Xie, *Senior Member, IEEE*

**Abstract**—Binarization plays a key role in image processing. Its performance directly affects the success of subsequent character segmentation and recognition. The Phansalkar algorithm performs excellent in processing heavily degraded or poor-quality images. However, this algorithm requires significant hardware costs. In this paper, efficient stochastic computing (SC) function and architecture are proposed for the Phansalkar algorithm. Highly accurate stochastic elements are designed for this architecture, including a stochastic mean circuit (SMC), a stochastic unipolar subtractor (USUB), a stochastic square root circuit (SQRT), and a stochastic exponential circuit (SEXP). Simulation results show that the SC architecture using 64-bit streams for the Phansalkar algorithm provides sufficient accuracy. Physical implementation indicates the effectiveness of the proposed architecture in lowering hardware costs for this algorithm compared to the binary counterpart.

**Index Terms**—Image binarization, Phansalkar algorithm, stochastic computing.

## I. INTRODUCTION

**B**INARIZATION is an initial step in some image analyses to separate target areas from the background. It is a complex task for heavily degraded or poor-quality images, which are affected by factors such as non-uniform intensity, shadows, smear, smudge, and low contrast. For example, confocal images are often non-uniformly illuminated [1]. Therefore, an effective image binarization algorithm is essential for processing such images. Binarization is the process of finding one or more ideal thresholds to divide pixels in an image into two groups, namely (1) the foreground (including text, characters, and shapes) and (2) the background (including contextual surfaces). In general, the methods of image binarization are classified as either global thresholding or local thresholding. A global method searches for a threshold for an entire image, which performs well for images with clear separations between the foreground and the background, such

This work was supported in part by the Fundamental Research Funds for the Central Universities of China under Grant JZ2020HGQA0162; and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RES0048688. (Corresponding author: Guangjun Xie)

Yongqiang Zhang, Jiao Qin, and Guangjun Xie are with the School of Microelectronics, Hefei University of Technology, Hefei 230009, China (e-mail: ahzhangyq@hfut.edu.cn; 1540436201@qq.com; gjxie8005@hfut.edu.cn)

Jie Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: jhan8@ualberta.ca)

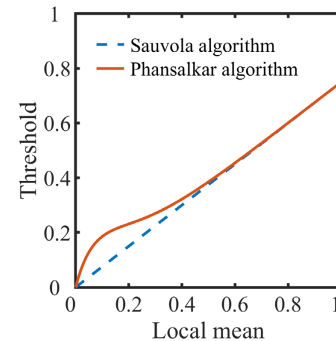
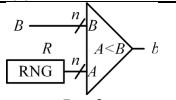

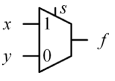
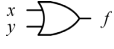


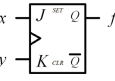


Fig. 1. The thresholds of the Phansalkar and Sauvola algorithms for different local means.

as the Otsu algorithm [2]. However, such separations are weakened in the images exposed to a gross environment. For these cases, a local thresholding method aims at estimating a threshold for each pixel based on its neighboring pixels, such as the Sauvola and Bernsen algorithms [3, 4]. Local thresholding methods have been widely employed for image analyses since they produce better results even for severely degraded or poor-quality images. The weighted binary implementations of these binarization algorithms usually consume large hardware costs and are highly susceptible to soft errors caused by cosmic radiation or noises caused by process, voltage, and temperature variations [5]. Tolerating these noises becomes increasingly important as devices continue to scale down to the nano regime.

Stochastic computing (SC) has been initiated to lower hardware costs for specific applications that can be designed loosely [6, 7], such as image processing algorithms [8, 9], neural networks [10, 11, 12, 13], and polynomial computation [14, 15]. The applications of SC in image edge detection, contrast stretching, filtering, segmentation, and sharpening have been fully investigated by optimizing related stochastic components, relating to finite-state machines, nonscaled adders, mean circuits, and inner-product units [9, 16, 17]. In these works, the superiority of the fault tolerance of stochastic components to a variety of noises over weighted binary counterparts has already been deeply demonstrated, by injecting random bit flip errors into components. As for image binarization, the Sauvola algorithm has been implemented by using the designed stochastic mean circuits and comparators [18]. In degraded or poor-quality images, the thresholds computed by the Sauvola algorithm are generally very small, so many foreground or background pixels are wrongly categorized

TABLE I  
STOCHASTIC COMPONENTS

Operation	(a) D/S converter	(b) S/D converter	(b) Scaled addition	(c) Addition	(d) Addition	(e) Multiplication	(f) Division
Component							
Function	$B \rightarrow b$	$b \rightarrow B$	$f = s \cdot x + (1-s) \cdot y$	$f = x + y$	$f = x + y$	$f = x \cdot y$	$f = x / (x + y)$
Correlation	NA	NA	Uncorrelated	Uncorrelated	Negatively correlated	Uncorrelated	Uncorrelated

[3, 18]. For the Phansalkar algorithm [19], the thresholds are much larger than those found by the Sauvola algorithm for a small local mean, thus providing better performance in binarizing images, as shown in Fig. 1. However, the weighted binary implementation of this algorithm requires large hardware resources.

To alleviate the issue that occurred in the implementation of the Phansalkar algorithm, a novel architecture is proposed for the algorithm in this paper to exploit the advantages of SC in reducing hardware costs. To this end, the parameters including  $q$ ,  $p$ , and  $k$ , and window size  $W$  in the Phansalkar algorithm are searched by using the simulation method. The peak signal to noise ratio (PSNR), mean squared error (MSE), maximum squared error (MAXERR), and ratio of squared norms (L2RAT) using the command ‘measerr’ in MATLAB are used to measure the best parameters. With these parameters, an SC function for the algorithm is then formulated based on the basic stochastic components. For example, a stochastic unipolar subtractor (USUB), a stochastic square root circuit (SQRT), and a stochastic exponential circuit (SEXP) are respectively proposed to build the architecture for the Phansalkar algorithm, aiming at a higher computing accuracy and lower hardware costs. All the proposed stochastic components are evaluated using the MSE for computing accuracy and physically synthesized using the Design Compiler (DC) with a TSMC 40-nm gate library under 100 MHz frequency and typical design corners.

The main contributions of this paper are summarized as follows. 1) A nonscaled USUB consisting of an up/down counter and a comparator is proposed. Compared with the previous one which is based on a stochastic divider and needs multiple loops to generate stable results, the proposed USUB computes results with lower MSEs in one loop and reduces area-delay product (ADP) and power-delay product (PDP) by 12.09% and 32.57%, respectively. 2) An SQRT is proposed by inserting a D flip flop to decorrelate the correlation to provide higher computing accuracy by sacrificing some hardware costs regarding a D flip flop. 3) An SEXP is proposed by saving three D flip-flops while realizing lower MSEs than previous designs. 4) With these components, an SC function and architecture for the Phansalkar algorithm are derived and built to provide an MSE of  $4.40 \times 10^{-2}$ , if using a 6-bit linear feedback shift register (LFSR). The SC architecture reduces ADP and PDP by approximately 98.43% and power by approximately 98.66% compared to the binary design implemented using the coordinate rotation digital computer (CORDIC) algorithm. The proposed architecture surpasses the designs composed of previous stochastic components in both hardware costs and

computing accuracy.

This work proceeds as follows. Section II introduces the basic concepts of the Phansalkar algorithm and SC. Section III presents the parameter setting, the formulated SC function, the proposed SC architecture, and the proposed stochastic components for the Phansalkar algorithm. Section IV illustrates the experimental results. Section V concludes this paper.

## II. BACKGROUND

### A. Phansalkar Algorithm

For a pixel located at coordinates  $(x, y)$  and centered on a  $W \times W$  window ( $W$  is an odd number), the Phansalkar algorithm computes its local threshold  $T(x, y)$  by using the local mean  $m(x, y)$  and standard deviation  $s(x, y)$  of neighboring pixels [19], as

$$T(x, y) = m(x, y) \cdot \left[ 1 + p \cdot e^{-q \cdot m(x, y)} + k \cdot \left( \frac{s(x, y)}{R} - 1 \right) \right], \quad (1)$$

where  $k \in [0.2, 0.5]$ ,  $p$  and  $q$  are bias constants, and  $R$  is the maximum value of the standard deviation (often 128 for 8-bit gray-scale images) [19]. If the value of  $q$  is too large, the exponential term becomes negligible, and (1) functions as the Sauvola algorithm. The constant  $p$  determines the degree of the effect of the exponential term on the computed thresholds. For a very small  $p$ , the performance of this algorithm is almost the same as the Sauvola algorithm. For a very large  $p$ , the threshold becomes too high, and too many background pixels are classified as the foreground. Thus, the values of  $q$ ,  $p$ , and  $k$  are key preconditions of an SC architecture for the Phansalkar algorithm.

### B. Stochastic Computing

SC is a re-emerging computing paradigm with the potential to outperform the weighted binary computing in terms of hardware efficiency and fault tolerance, because it relies on logic operations on stochastic bitstreams. Stochastic bitstreams, referred to as stochastic numbers (SNs), are generated by a digital-to-stochastic (D/S) converter as listed in the 2<sup>nd</sup> column in TABLE I, by comparing a given  $n$ -bit binary number  $B$  and a random number  $R$  generated by a random number generator (RNG) [20]. It produces a 1 or a 0 if  $B$  is larger or smaller than  $R$  in each clock cycle to generate an SN with a length of  $2^n$  bits. The RNG is usually a  $2^n$ -bit LFSR or Sobol sequence generator (SSG), instead of a truly random number source [21]. Compared to pseudorandom numbers generated by an LFSR, an SSG generates low-discrepancy numbers and generally improves computing accuracy [22]. To convert an SN back to its binary encoding format, a stochastic-to-digital (S/D)

TABLE II  
PSNR, MSE, MAXERR, AND L2RAT FOR DIFFERENT  $p$  VALUES

$p$	PSNR	MSE	MAXERR	L2RAT
2	7.0856	13835.45	240	3.2122
3	6.9174	14523.11	241	3.3439
4	6.7853	15023.31	241	3.4170
5	6.6883	15377.30	242	3.4512

TABLE III  
PSNR, MSE, MAXERR, AND L2RAT FOR DIFFERENT  $k$  VALUES

$k$	PSNR	MSE	MAXERR	L2RAT
0.2	7.0856	13835.45	240	3.2122
0.3	6.9174	14523.11	241	3.3439
0.4	6.7853	15023.31	241	3.4170
0.5	6.6883	15377.30	242	3.4512

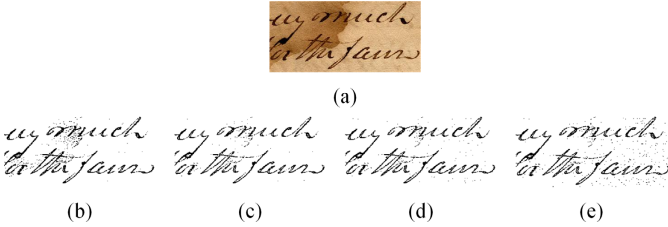


Fig. 2. (a) Original degraded document image. (b) 3×3, (c) 5×5, (d) 7×7, and (e) 9×9 window sizes using the Phansalkar algorithm.

TABLE IV  
THE MSEs ( $\times 10^{-4}$ ) OF THE PHANSALKAR ALGORITHM FOR DIFFERENT WINDOW SIZES

Algorithm	3×3	5×5	7×7	9×9	11×11	13×13	15×15
Phansalkar	1.30	1.23	1.20	1.19	1.18	1.18	1.16

converter, for example, a counter, is employed to sum up each bit in an SN, as in the 3<sup>rd</sup> column [23]. In the unipolar format, the probability  $p$  of each bit being 1 in a bitstream equals the corresponding binary number  $B$  within  $[0, 1]$ , while the probability  $p$  is  $(A+1)/2$  in the bipolar format, where  $A$  is within  $[-1, 1]$ . Since this work focuses on image binarization with positive pixel values, we consider only the unipolar format later. We use lowercases to indicate bitstreams, corresponding binary values, and logic values for simplicity unless otherwise specified. One can distinguish the represented meaning from the context easily.

The multiplexer (MUX)-based scaled adder performs  $f=s \cdot x+(1-s) \cdot y$ , where  $s$  is the select and has to be uncorrelated with inputs  $x$  and  $y$ , and  $x$  and  $y$  can be correlated with each other (In the 4<sup>th</sup> column). In addition, an OR gate with uncorrelated or negatively correlated bitstreams respectively performs  $f=x+y \cdot x \cdot y$  or  $f=x+y$  (In the 5<sup>th</sup> and 6<sup>th</sup> columns). Thus, a nonscaled adder can be designed through an OR gate by manipulating the correlation. An AND gate performs  $f=x \cdot y$  if bitstreams  $x$  and  $y$  are uncorrelated (In the 7<sup>th</sup> column). An interesting stochastic component is the JK flip flop (JKFF), which performs stochastic division  $x/(x+y)$  provided that the uncorrelated input bitstreams are long enough (In the 8<sup>th</sup> column).

As can be seen in TABLE I, the stochastic computing correlation (SCC) significantly affects the performed function for a stochastic component. The SCC value of two bitstreams  $x$

**Algorithm 1** Algorithm for Finding Best Parameters  $q, p$ , and  $k$ , and window size  $W \times W$  for the Phansalkar Algorithm

**Input:** 8-bit grayscale images,  $n$  (an odd number)

**Output:** Parameters  $q, p$ , and  $k$ , and window size  $W \times W$

for  $q=1$  to 15 do

AE= $\exp(-0.5q) < 0.01$  then break;

for  $i=1$  to  $n$  do

$W=2i+1$ ;

for  $k=0.2$  to 0.5 do

for  $p=2$  to 5 do

PSNR=psnr;

MSE=mse;

MAXERR=maxerr;

L2RAT=l2rat;

$[p, k, W]=\text{find}(\max(\text{PSNR}), \min(\text{MSE}), \min(\text{MAXERR}), \min(\text{L2RAT}))$

and  $y$  is defined as [24]

$$\text{SCC}(x, y) = \begin{cases} \frac{\delta(x, y)}{\min(p_x, p_y) - p_x p_y}, & \delta(x, y) > 0 \\ 0, & \delta(x, y) = 0, \\ \frac{\delta(x, y)}{p_x p_y - \max(p_x + p_y - 1, 0)}, & \delta(x, y) < 0 \end{cases} \quad (2)$$

where  $\delta(x, y) = p_{xy} - p_x p_y$  ( $p_{xy}$  is the probability of the ANDed results of  $x$  and  $y$ ). A value of 0 means two bitstreams are ideally independent, while a value of +1 or -1 respectively indicates a maximally positive or negative correlation.

### III. A STOCHASTIC COMPUTING ARCHITECTURE

#### A. Parameter Setting

Considering (1), an SC architecture for the Phansalkar algorithm relates to: 1) the parameters  $q, p$ , and  $k$ ; and 2) a selected window size  $W \times W$ .

The algorithm for finding the best parameters  $q, p$ , and  $k$ , and window size  $W \times W$  is described in Algorithm 1. The input images are a dozen gray-scale images and the outputs are the desired parameters [25, 26]. The mean value of pixels of 8-bit grayscale images is 128, which has to be scaled down to 0.5 in SC because all pixel values are scaled down from  $[0, 255]$  to  $[0, 1]$ . Assume that the exponent of the exponential term  $-q \cdot m(x, y)$  in (1) is -5 since  $e^{-q \cdot m(x, y)} = e^{-5} = 0.0067$ , which is small enough and considered a 0. The resulting absolute error from  $e^{-5}$  is only 0.0067, compared to 0. If setting  $e^{-q \cdot m(x, y)} = e^{-4}$ , the absolute error is about 0.0183, which is about 2.72 times larger than that of  $e^{-5}$ . On the other hand, if  $e^{-q \cdot m(x, y)} = e^{-6}$ , the related stochastic implementation of this function will be more complicated, as can be seen later. For a mean value of 0.5, therefore,  $q=10$  will be used in the SC architecture for the Phansalkar algorithm. Thus, the local mean values less than 0.5 will have an effect on the computation of local thresholds. When the local mean values are larger than 0.5, the effect of the exponential term in (1) is diminishing. This can also be seen from the convergence in Fig. 1. The parameter  $p$  determines the magnitude of the exponential term influencing the thresholds. When  $p$  is within  $[0, 1]$ , the Phansalkar and Sauvola algorithms achieve almost the same results; when  $p$  is greater than 5, the thresholds are too high and more background pixels will be classified as the

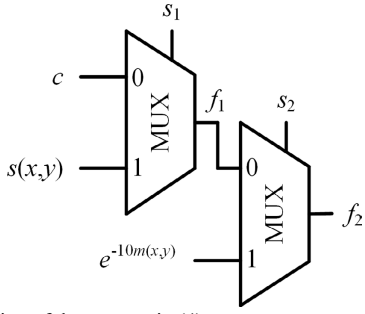


Fig. 3. The addition of three terms in (4).

foreground, or vice versa [4]. A dozen gray-scale images are tested to measure the PSNR, MSE, MAXERR, and L2RAT using the command ‘measerr’ in MATLAB [25]. Experimental results show that  $p=2$  provides the best performance in terms of the four metrics, as listed in TABLE II. In the same way as determining the parameter  $p$ , the parameter  $k$  is within  $[0.2, 0.5]$ , and the experimental results show that  $k=0.2$  reaches the best efficiency in terms of the four metrics, as listed in TABLE III.

The quality of binarized images processed by the Phansalkar algorithm also depends on a predefined window size. An oversized window would not significantly improve the processed image quality, even at the cost of additional hardware resources [19]. To verify the effect of window size on the quality of binarized images and determine an appropriate window size, the Phansalkar algorithm is evaluated using the binary method. Fig. 2(a) shows an original degraded image selected from the dataset document image binarization contest (DIBCO) [26]. Fig. 2(b), (c), (d), and (e) are respectively the images processed using the Phansalkar algorithm with  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$  and  $9 \times 9$  window sizes. According to TABLE IV, the MSEs of the Phansalkar algorithm decrease as the window size increases. The downtrend of the MSEs becomes slower when the window size exceeds  $7 \times 7$ . To balance hardware costs and computing accuracy, a  $7 \times 7$  window size is chosen in this work.

### B. Formulation

To build the SC architecture for the Phansalkar algorithm, all pixel values must be scaled down from  $[0, 255]$  to  $[0, 1]$ . All generated results have also to be limited within  $[0, 1]$ . Thus, the maximum pixel value of the standard deviation  $R$  is 1 and the mean value of pixels is 0.5 in SC. According to these defined parameters above, the realized function for the Phansalkar algorithm is

$$T'(x, y) = m(x, y) \cdot \left[ 1 + 2e^{-10m(x, y)} + \frac{1}{5}(s(x, y) - 1) \right]. \quad (3)$$

Note that, the maximum value of the thresholds computed by (3) is 3, so a factor of  $1/3$  has to be added to this function to scale down computed results within  $[0, 1]$ . Thus, the proposed SC function for the Phansalkar algorithm is

$$T(x, y) = m(x, y) \cdot \left[ \frac{2}{3}e^{-10m(x, y)} + \frac{1}{15}s(x, y) + \frac{4}{15} \right]. \quad (4)$$

### C. The Overall Architecture

With the proposed SC function for the Phansalkar algorithm (4), it has to be transformed into a format that can be

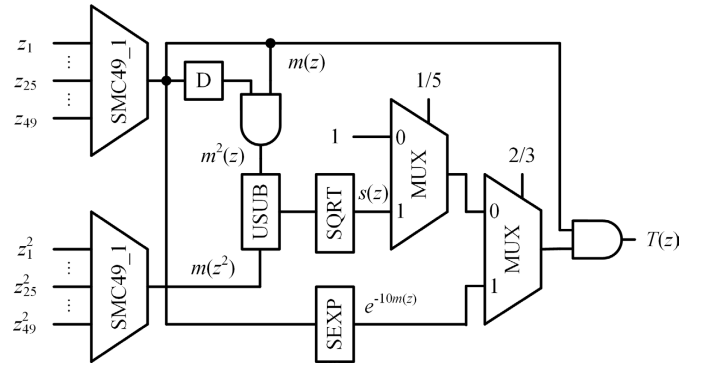


Fig. 4. The proposed stochastic computing architecture for the Phansalkar algorithm.

implemented through stochastic components. The computation of the threshold relates to the multiplication of a local mean value  $m(x, y)$  and a term in the square bracket in (4), which can be realized through an AND gate. The local mean value  $m(x, y)$  can be computed by using multi-input MUX-based stochastic mean circuits or other methods relating to correlated bitstreams as [17, 18, 27].

The point in (4) is the terms in the square bracket, which contains the addition of three terms and the computation of the standard deviation and exponential function. The addition needs at least two MUX-based scaled adders to be cascaded, of which the input bitstreams are  $e^{-10m(x, y)}$  and  $s(x, y)$ , and other select and input bitstreams are to be configured as follows. Assume the select and input bitstreams of the first scaled adder are  $s_1$ ,  $c$ , and  $s(x, y)$ . Its output  $f_1$  is fed into the second adder with select  $s_2$  and input bitstreams  $e^{-10m(x, y)}$ , generating an output  $f_2$  as shown in Fig. 3. Thus, we have

$$\begin{aligned} f_2 &= s_2 \cdot e^{-10m(x, y)} + (1 - s_2) \cdot f_1 \\ &= s_2 \cdot e^{-10m(x, y)} + (1 - s_2) \cdot [s_1 \cdot s(x, y) + (1 - s_1) \cdot c] \quad (5) \\ &= s_2 \cdot e^{-10m(x, y)} + (1 - s_2) \cdot s_1 \cdot s(x, y) + (1 - s_2) \cdot (1 - s_1) \cdot c \end{aligned}$$

Taking both (4) and (5) into consideration, we have an equation group containing three equations, as

$$\begin{aligned} \frac{2}{3} &= s_2 \\ \frac{1}{15} &= (1 - s_2) \cdot s_1 \\ \frac{4}{15} &= (1 - s_2) \cdot (1 - s_1) \cdot c \end{aligned} \quad (6)$$

Thus,  $s_2=2/3$ ,  $s_1=1/5$ , and  $c=1$ . With these computed constants, the proposed SC function for the Phansalkar algorithm (4) can be modified to

$$\begin{aligned} T(x, y) &= \\ &= m(x, y) \cdot \left[ \frac{2}{3}e^{-10m(x, y)} + \left(1 - \frac{2}{3}\right) \cdot \left(\frac{1}{5}s(x, y) + \left(1 - \frac{1}{5}\right) \cdot 1\right) \right] \quad (7) \end{aligned}$$

According to (7), the proposed SC architecture for the Phansalkar algorithm is shown in Fig. 4, where  $z_1, z_2, \dots, z_{49}$  are input pixels in a  $7 \times 7$  window and  $T(z)$  is the computed threshold. It mainly consists of two 49-to-1 stochastic mean circuits (SMC49\_1s), a stochastic unipolar subtractor (USUB),

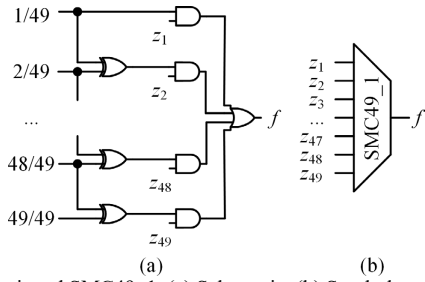


Fig. 5. The designed SMC49\_1. (a) Schematic. (b) Symbol.

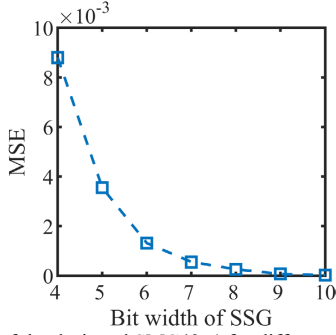


Fig. 6. The MSEs of the designed SMC49\_1 for different bit widths of SSGs.

a stochastic square root circuit (SQRT), and a stochastic exponential circuit (SEXP). The SC architecture works as follows. The SMC49\_1s are respectively used to generate the mean value  $m(z)$  of 49 input bitstreams  $z_1, z_2, \dots, z_{49}$ , and the mean value  $m(z^2)$  of 49 squared input bitstreams  $z_1^2, z_2^2, \dots, z_{49}^2$ . The mean value  $m(z)$  is delayed by one clock cycle and passed through an AND gate to generate  $m^2(z)$ . The results  $m^2(z)$  and  $m(z^2)$  are then passed through the USUB and SQRT to compute the standard deviation  $s(z)$  of 49 neighboring pixel values. The mean value  $m(z)$  is also processed through the SEXP to compute the exponential term  $e^{-10m(z)}$ . The result is then imported into the scaled adders, along with the standard deviation  $s(z)$ . The result of the last scaled adder is multiplied by the mean value  $m(z)$  to compute the threshold  $T(z)$ . Four main stochastic components including the SMC49\_1, USUB, SQRT, and SEXP are described as follows.

#### D. A Stochastic Mean Circuit (SMC)

An SMC is to average the sum of its input bitstreams by dividing their count, which has been mainly designed through cascaded MUXs by controlling their select bitstreams, as in [18]. However, while the input bitstreams can be correlated, multiple uncorrelated select bitstreams have to be generated, which will incur large hardware costs. We have demonstrated in [17, 27] that positively correlated input bitstreams can be averaged through a very simple stochastic component to realize a high computing accuracy. In this work, the method in [17] is used to design the required SMC to average 49 input bitstreams in a  $7 \times 7$  window (Abbreviated as SMC49\_1), as shown in Fig. 5(a). Fig. 5(b) is the symbol for simplicity. In this design,  $1/49, 2/49, \dots, 49/49$  are generated through sharing the same SSG, of which the bitstreams are positively correlated [17]. Thus, each output of an XOR gate with positively correlated input bitstreams, performing absolute subtraction, is  $1/49$ . In

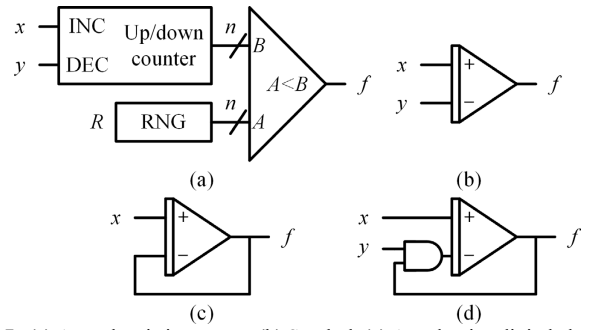


Fig. 7. (a) A stochastic integrator. (b) Symbol. (c) An adaptive digital element. (d) A stochastic divider.

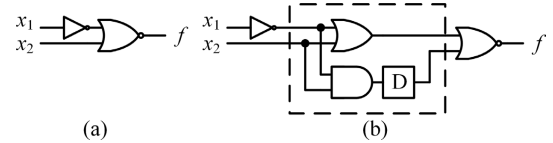


Fig. 8. USUB<sub>NOR</sub> [33]. (a) Schematic. (b) Enhanced USUB<sub>NOR</sub>.

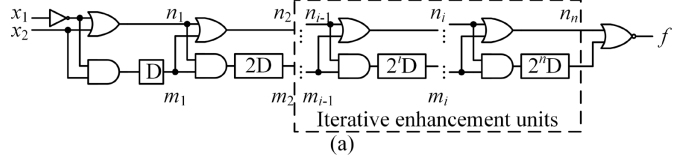


Fig. 9. USUBs. (a) USUB<sub>Hi</sub> [33]. (b) USUB<sub>Div</sub> [33]. (c) The proposed USUB.

addition, the 49 input bitstreams are produced through another direct vector array (DVA) for the SSG [22]. The input bitstreams and those of  $1/49$  are uncorrelated theoretically. Thus, each output of an AND gate is the multiplication of  $z_i$  and  $1/49$  ( $1 \leq i \leq 49$ ), which is negatively correlated as demonstrated in [17]. They are summed through an OR gate with negatively correlated bitstreams to perform  $(z_1 + z_2 + \dots + z_{49})/49$  at last. The MSEs for the designed SMC49\_1 using an SSG are shown in Fig. 6. The MSE reaches  $2.56 \times 10^{-4}$  using an 8-bit SSG.

#### E. A Unipolar Subtractor (USUB)

The standard deviation  $s(z)$  of the threshold is  $(m(z^2) - m(z)^2)^{0.5}$ , for which a nonscaled unipolar subtractor (USUB) is used. From the theorem [28], it is known that  $m(z^2) \geq m(z)^2$ . A unipolar to bipolar subtractor has been proposed in [29], but it is a scaled subtractor. A counter-based scaled unipolar absolute subtractor has also been proposed in [30]. Both of the results for these two subtractors are scaled by a factor of  $1/2$ , which will decrease the computing accuracy that applications need.

Before the introduction of nonscaled subtractors, it is beneficial to detail the stochastic integrator [6, 31, 32], as shown in Fig. 7(a), where  $x, y$ , and  $f$  are input and output bitstreams. Fig. 7(b) is the symbol. The integrator consists of an up/down counter and a D/S converter containing an RNG and a comparator. The  $n$ -bit counter stores an initial value  $c_0$ , which is increased by 1 if  $x_i=1$  and  $y_i=0$ , decreased by 1 if  $x_i=0$  and  $y_i=1$ , or stays unchanged at clock cycle  $i$ . The resulting value  $c$  is compared with a random number  $R$  in the RNG to generate a 1

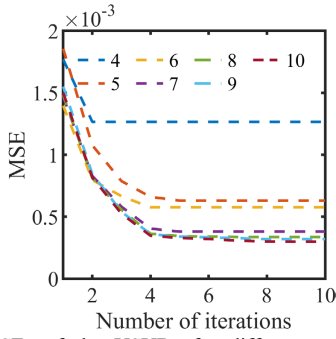


Fig. 10. The MSEs of the USUB<sub>It</sub> for different numbers of iterative enhancement units.

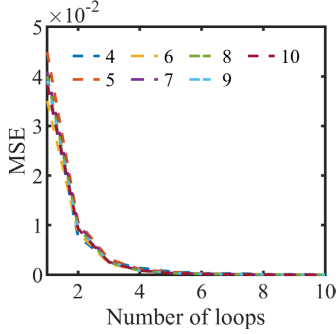


Fig. 11. The MSEs of the USUB<sub>Div</sub> for different numbers of loops.

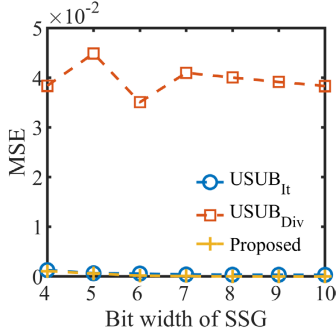


Fig. 12. The MSEs of the three USUBs for different bit widths of SSGs.

with a probability of  $c/2^n$ . The stochastic integrator functions  $f = c_0 + \frac{1}{2^n} \sum_{i=0}^{k-1} (x_i - y_i)$ , after  $k$  clock cycles. The integrator with a feedback loop can be used as an adaptive digital element (ADDIE) to estimate the probability of its input bitstream [6, 31, 32], as shown in Fig. 7(c). An important application of the stochastic integrator is the stochastic divider, by looping the output bitstream through an AND gate [6], as shown in Fig. 7(d). The stochastic divider implements  $f=x/y$  (Assume  $x \leq y$ ) when an equilibrium state is reached in the up/down counter [31]. Experimental results have shown that the equilibrium state can only be realized through looping the input bitstreams repeatedly, which will take a much longer computation time for applications using this divider.

Two types of nonscaled unipolar subtractors have been designed in [33]. The first one, denoted as USUB<sub>NOR</sub>, is based on a NOR gate, as shown in Fig. 8(a). The output  $f$  can be written as

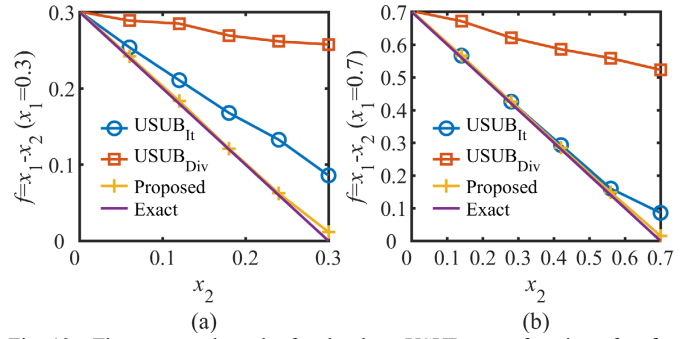


Fig. 13. The computed results for the three USUBs as a function of  $x_2$  for different values of  $x_1$  using 8-bit SSGs. (a)  $x_1=0.3$ . (b)  $x_1=0.7$ .

TABLE V  
THE HARDWARE COSTS OF THE THREE USUBS USING 8-BIT SSGS

Design	Area	Power	CPD	ADP	PDP
USUB <sub>It</sub>	542.78	0.05685	1.44	781.61	0.0819
USUB <sub>Div</sub>	416.30	0.03235	1.44	599.48	0.0466
Proposed	364.97	0.02657	1.44	525.56	0.0383

Area:  $\mu\text{m}^2$ ; Power: mW; CPD: ns; ADP:  $\mu\text{m}^2\text{-ns}$ ; PDP: mW $\cdot$ ns.

$$f = 1 - \left( (1-x_1) + x_2 - (1-x_1) \cdot x_2 \right), \quad (8)$$

$$= x_1 - x_2 + (1-x_1) \cdot x_2$$

where  $x_1$ ,  $x_2$ , and  $f$  are input and output bitstreams, and  $x_1$  and  $x_2$  are uncorrelated. It is a feasible approximation of  $x_1-x_2$ , if  $x_1$  is large enough and  $x_2$  is very small. The computing accuracy can be improved by adding an enhancement unit as shown in the dashed box in Fig. 8(b), where a D flip flop (DFF) is for decorrelation. The output  $f$  in Fig. 8(b) is written as

$$f = 1 - \left( \frac{\left( (1-x_1) + x_2 - (1-x_1) \cdot x_2 \right) + (1-x_1) \cdot x_2 - \left( (1-x_1) + x_2 - (1-x_1) \cdot x_2 \right) \cdot \left( (1-x_1) \cdot x_2 \right)}{\left( (1-x_1) + x_2 - (1-x_1) \cdot x_2 \right)} \right). \quad (9)$$

$$= x_1 - x_2 + (1-x_1) \cdot x_2 (1-x_1 \cdot (1-x_2))$$

Thus, comparing (9) with (8), the enhancement unit can decrease the error to some extent. Thus, a unipolar subtractor using iterative enhancement units, denoted as USUB<sub>It</sub>, can be designed in this way, as shown in Fig. 9(a). However, it has the disadvantage that many iterative enhancement units have to be used to obtain a high computing accuracy.

The second one, named USUB<sub>Div</sub>, has been designed using the stochastic divider in Fig. 7(d) based on the stochastic integrator [33]. The circuit structure is shown in Fig. 9(b), where the output  $z$  is  $z=x_2/x_1$ , assuming  $x_2 \leq x_1$ . Thus, the output  $f$  is

$$f = \left( 1 - \frac{x_2}{x_1} \right) \cdot x_1. \quad (10)$$

$$= x_1 - x_2$$

Its disadvantage is that it consumes many loops and a large area to reach a high accuracy because of the D/S converter.

In this paper, a nonscaled USUB composed of an up/down counter and a comparator is proposed, as shown in Fig. 9(c). Assume that the bitstreams  $x_1$  and  $x_2$  are unipolar and  $x_2 \leq x_1$ . The up/down counter increases by 1 in each clock cycle when  $w < x_2$ , while the up/down counter decreases by 1 if  $w > x_2$ . The up/down counter reaches an equilibrium state when  $x_2 = w$ , which

indicates the output of the up/down counter is  $c=x_2=w$ . The output  $c=x_2=w$  of the up/down counter is compared to 0 to generate a 1 if the value is greater than 0 and a 0 otherwise. This means the output of the comparator is 1 with a probability of  $c=x_2=w$ . That is, the output of the comparator is  $z=c=x_2=w$  in this stable state. In addition, since  $w=x_1 \cdot z$ , the output  $z$  will converge to  $z=x_2/x_1$ . Therefore, the output  $f$  of the proposed USUB is

$$f = \left(1 - \frac{x_2}{x_1}\right) \cdot x_1 = x_1 - x_2 \quad (11)$$

The computing accuracy of the proposed USUB and existing USUB<sub>It</sub> and USUB<sub>Div</sub> is evaluated by using the MSE, where two input bitstreams are generated by different DVAs for an SSG. Fig. 10 shows the MSEs of the USUB<sub>It</sub> for different numbers of iterative enhancement units, using 4-, 5-, ..., and 10-bit SSGs. It is concluded that when the number of iterative enhancement units reaches 6, the MSEs stabilize. Thus, the USUB<sub>It</sub> with 6 iterative enhancement units is used later. The MSEs of the USUB<sub>Div</sub> for different numbers of loops are shown in Fig. 11, using 4-, 5-, ..., and 10-bit SSGs. The number of loops means the number of repetitions that input bitstreams are repeatedly computed. The USUB<sub>Div</sub> needs at least 6 loops to reach a stable MSE. However, the clock delay of this design is 6 times longer than a design with one loop. Thus, the number of loops is set to 1 for further consideration.

Fig. 12 shows the MSEs of the three USUBs for different bit widths of SSGs. The USUB<sub>Div</sub> with one loop produces the largest MSEs, resulting in the worst computing accuracy. The proposed USUB provides slightly lower MSEs than the USUB<sub>It</sub>. This can also be seen from the plots for the three USUBs as a function of  $x_2$  for different values of  $x_1$  in Fig. 13 using 8-bit SSGs, where the values of  $x_1$  are given as 0.3 and 0.7. The errors of the USUB<sub>Div</sub> with one loop are the largest regardless of the values of  $x_1$ . The proposed USUB almost obtains the exact results, while the USUB<sub>It</sub> reaches large or small errors for  $x_1=0.3$  or  $x_1=0.7$ , respectively.

The hardware costs of the three USUBs using 8-bit SSGs are listed in TABLE V, including area, power, critical path delay (CPD), area-delay product (ADP), and power-delay product (PDP). The proposed USUB has the same CPD as the USUB<sub>It</sub> and USUB<sub>Div</sub>. The ADP and PDP are respectively reduced by up to 17.81% and 53.24%, compared to those of the USUB<sub>It</sub>. These metrics of the proposed USUB are respectively reduced by about 12.09% and 32.57%, compared to those of the USUB<sub>Div</sub>.

#### F. A Stochastic Square Root Circuit (SQRT)

The standard deviation  $s(z)=(m(z^2)-m(z)^2)^{0.5}$  of the threshold relates to the stochastic square root (SQRT) function.

SQRT<sub>SR</sub>: A shift register (SR) based SQRT abbreviated as SQRT<sub>SR</sub> has been designed in [34], as shown in Fig. 14(a). The left MUX realizes scaled addition as

$$f = 1 \cdot (1 - n_5) + x \cdot n_5, \quad (12)$$

where  $x$  and  $f$  are input and output bitstreams, and  $n_5$  is the select. The SR consisting of a DFF and the neighboring MUX is

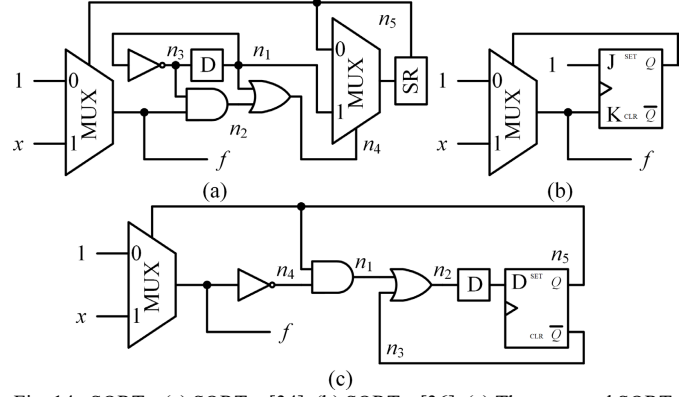


Fig. 14. SQRTs. (a) SQRT<sub>SR</sub> [34]. (b) SQRT<sub>JK</sub> [36]. (c) The proposed SQRT.

the division kernel, which realizes  $n_1/n_4$  in an equilibrium state [35]. That is,

$$n_5 = \frac{n_1}{n_4}, \quad (13)$$

where  $n_1$  is the input connected to the pin '1' of the MUX and  $n_4$  is the select. The DFF and the inverter make up a random number source to provide a value of 1/2. Thus,  $n_1=n_3=1/2$ . According to the definition of SCC (2),  $\delta(n_1, n_2)$  is

$$\begin{aligned} \delta(n_1, n_2) &= p_{n_1 n_2} - p_{n_1} p_{n_2} \\ &= p_{n_1 n_3 f} - p_{n_1} p_{n_3 f} \\ &= -p_{n_1} p_{n_3 f} \end{aligned} \quad (14)$$

Further, since

$$p_{n_3 f} \leq p_{n_3} = 1 - p_{n_1}, \quad (15)$$

$$\max(p_{n_1} + p_{n_3 f} - 1, 0) = 0. \quad (16)$$

Thus, the SCC between  $n_1$  and  $n_2$  is -1. The OR gate with negatively correlated bitstreams performs the addition of  $n_1$  and  $n_2$  as

$$\begin{aligned} n_4 &= n_1 + n_2 \\ &= \frac{1}{2} + \frac{1}{2} \cdot f \end{aligned} \quad (17)$$

Taking (13) into consideration, we have

$$n_5 = \frac{1}{1+f}. \quad (18)$$

Take (18) into (12), we have  $f=x^{0.5}$  for the SQRT<sub>SR</sub>.

SQRT<sub>JK</sub>: A JK flip flop (JKFF) based SQRT denoted as SQRT<sub>JK</sub> has been proposed [36], as shown in Fig. 14(b). Its working principle is that the output of an SQRT is always greater than or equal to its input. The JKFF performs stochastic division as

$$q = \frac{j}{j+k}, \quad (19)$$

where  $j$ ,  $k$ , and  $q$  are the input and output bitstreams of the JKFF. If  $j=1$ ,  $q$  equals  $1/(1+k)$ . As for the MUX, it performs scaled addition as

$$f = 1 \cdot (1 - q) + xq, \quad (20)$$

where  $q$  is the select of the MUX in Fig. 14(b), and  $x$  and  $f$  are input and output bitstreams. Take (19) into (20), we have

$$\begin{aligned}
f &= 1 \cdot \left(1 - \frac{j}{j+k}\right) + x \cdot \frac{j}{j+k} \\
&= 1 - \frac{1}{1+k} + x \cdot \frac{1}{1+k} \\
&= 1 - \frac{1}{1+f} + x \cdot \frac{1}{1+f}
\end{aligned} \tag{21}$$

Thus,  $f=x^{0.5}$  for the SQRT<sub>JK</sub>.

The proposed SQRT: Fig. 14(c) shows the proposed SQRT. The output of the MUX is

$$f = 1 \cdot (1-q) + xq, \tag{22}$$

where  $q$  is the output of the right DFF and thus  $q=(1-f)/(1-x)$ . For simplicity, five nodes  $n_1, n_2, n_3, n_4,$  and  $n_5$  are marked in Fig. 14(c). The values of nodes  $n_3, n_4,$  and  $n_5$  equals  $1-q, 1-f,$  and  $q,$  respectively. Since the AND gate with uncorrelated input bitstreams realizes multiplication, the value of node  $n_1$  can be computed as

$$\begin{aligned}
n_1 &= n_4 \cdot n_5 \\
&= (1-f) \cdot q
\end{aligned} \tag{23}$$

The computation of node  $n_2$  relies on the SCC between the bitstreams of nodes  $n_1$  and  $n_3$ . According to (2),  $\delta(n_1, n_3)$  is

$$\begin{aligned}
\delta(n_1, n_3) &= p_{n_1 n_3} - p_{n_1} p_{n_3} \\
&= p_{Q(1-f)\bar{Q}} - p_{Q(1-f)} p_{\bar{Q}} \\
&= -p_{Q(1-f)} p_{\bar{Q}}
\end{aligned} \tag{24}$$

where  $1-f$  means the inversion of  $f$ . Since

$$\begin{aligned}
p_{Q(1-f)} &\leq p_Q \\
p_{Q(1-f)} + p_{\bar{Q}} &\leq p_Q + p_{\bar{Q}} = 1,
\end{aligned} \tag{25}$$

we have

$$\max(p_{Q(1-f)} + p_{\bar{Q}} - 1, 0) = 0. \tag{26}$$

Thus, the SCC between nodes  $n_1$  and  $n_3$  is  $-1$ , which indicates the bitstreams of nodes  $n_1$  and  $n_3$  are negatively correlated. With this support, the OR gate with negatively correlated bitstreams realizes the addition of  $n_1$  and  $n_3$ , that is

$$\begin{aligned}
n_2 &= n_1 + n_3 \\
&= (1-f) \cdot q + (1-q)
\end{aligned} \tag{27}$$

Since a DFF is just a buffer,  $n_5=q=n_2$ . According to (22) and (27), we have

$$q = \frac{1-f}{1-x} = \frac{1}{1+f}. \tag{28}$$

Thus, the proposed SQRT computes  $f=x^{0.5}$ .

Fig. 15 shows the trends of the MSEs of the proposed SQRT for different numbers of DFFs, using different bit widths of SSGs and LFSRs. The insertion of a reasonable number of DFFs significantly affects the computing accuracy of the proposed SQRT. For example, ultra-high MSEs are reached if 10 DFFs are inserted. Taking both accuracy and hardware costs into consideration, a DFF is adopted in the proposed SQRT.

The computing accuracy of the proposed SQRT and existing SQRT<sub>SR</sub> and SQRT<sub>JK</sub> are compared via the MSE, as shown in Fig. 16, using 8-bit SSGs and LFSRs, respectively. The proposed SQRT outperforms the SQRT<sub>SR</sub> having the largest MSEs and the SQRT<sub>JK</sub> ranking second in two cases. However,

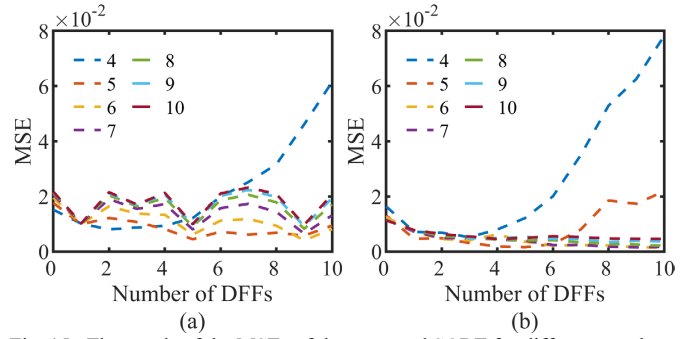


Fig. 15. The trends of the MSEs of the proposed SQRT for different numbers of DFFs. (a) SSG. (b) LFSR.

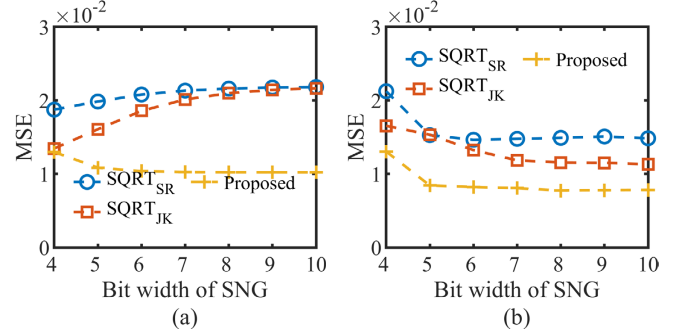


Fig. 16. The MSEs of the three SQRTs for different bit widths of SNGs. (a) SSG (b) LFSR.

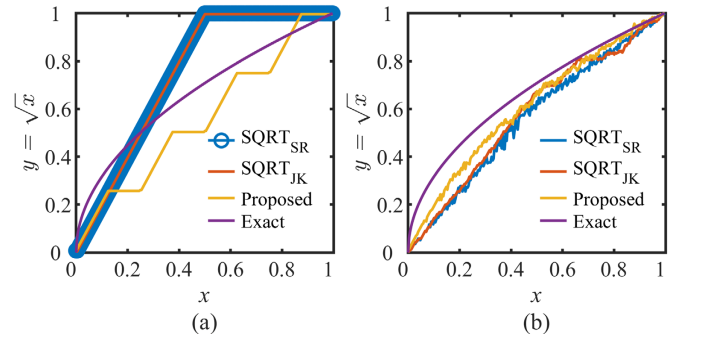


Fig. 17. The computed results for the three SQRTs using 8-bit SNGs. (a) SSG. (b) LFSR.

TABLE VI  
THE HARDWARE COSTS OF THE THREE SQRTs USING 8-BIT SSGs

Method	Area	Power	CPD	ADP	PDP
SQRT <sub>SR</sub>	210.09	0.01732	1.12	235.30	0.0188
SQRT <sub>JK</sub>	203.04	0.01634	1.15	233.49	0.0194
Proposed	207.45	0.01710	1.15	238.56	0.0197

Area:  $\mu\text{m}^2$ ; Power: mW; CPD: ns; ADP:  $\mu\text{m}^2\text{-ns}$ ; PDP: mW $\cdot\text{ns}$ .

since SSGs generating dependent random numbers are more suitable for stochastic combinational logic circuits and integrators than circuits using looped flip flops, the trends of the MSEs seem to be counterintuitive for the designs using SSGs, but they are normal, as demonstrated in [22]. This abnormality can also be seen from the plots for the three SQRTs using 8-bit SSGs and LFSRs in Fig. 17, where hard-threshold functions for the three SQRTs using SSGs are created. All the designs using 8-bit SSGs deviate from the exact results more than those using 8-bit LFSRs. The proposed SQRT using LFSRs is closer to the exact results compared to the SQRT<sub>SR</sub> and SQRT<sub>JK</sub>.



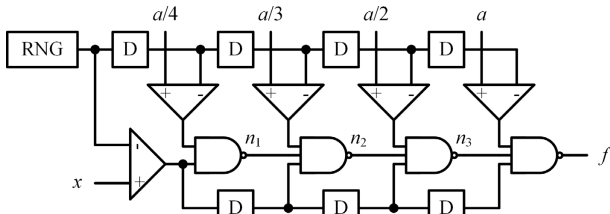


Fig. 18. The schematic of the SEXP<sub>a</sub> using 4<sup>th</sup>-order Maclaurin series expansion and Horner's rule [37].

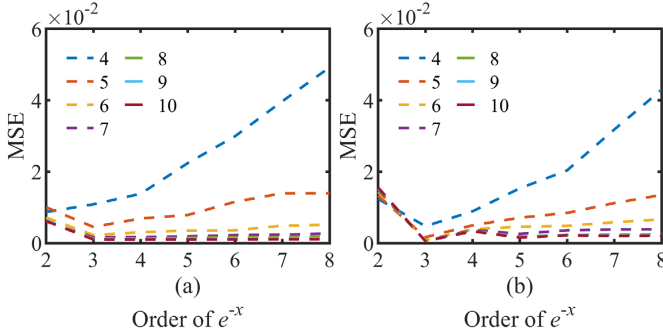


Fig. 19. The MSEs of the SEXP1 for different orders. (a) SSG. (b) LFSR.

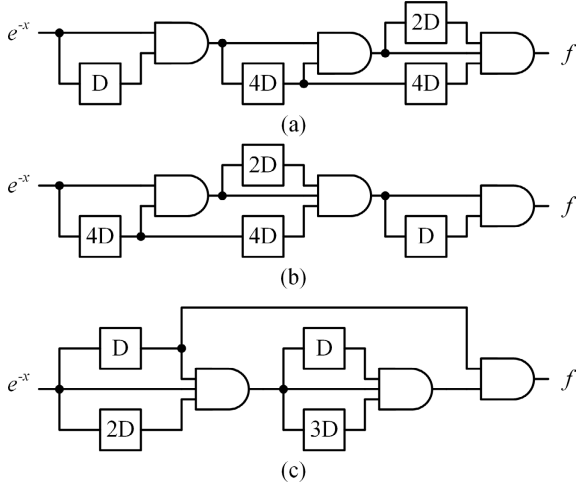


Fig. 20. The schematics of the SEXP10 using Maclaurin series expansion and Horner's rule. (a) SEXP10\_1 in [37]. (b) SEXP10\_2 in [37]. (c) The proposed SEXP10.

The hardware costs of the three SQRTs using 8-bit SSGs are listed in TABLE VI. The proposed SQRT is slightly weaker than the other two designs since a DFF is inserted to increase computing accuracy.

### G. A Stochastic Exponential Circuit (SEXP)

Stochastic exponential circuits (SEXP) have been realized through truncated Maclaurin series expansion and Horner's rule [37]. A method using correlated bitstreams has also been studied for implementing SEXPs in [15]. However, a correlation manipulation circuit has to be used if employing this method in an intermediate stage in a stochastic circuit [20], which will increase hardware costs. Since the SEXPs using truncated Maclaurin series expansion and Horner's rule show better performance than those using correlation [15], Bernstein polynomials [38], and finite state machines (FSMs) [23, 39, 40], the method of designing the SEXPs in [37] is used in this paper.

The Maclaurin series expansion of an exponential function  $e^{-ax}$  (denoted as SEXP<sub>a</sub>) is

$$e^{-ax} = \sum_{n=0}^{\infty} \frac{(-ax)^n}{n!} = 1 - ax + \frac{a^2}{2!}x^2 - \frac{a^3}{3!}x^3 + \frac{a^4}{4!}x^4 - \dots \quad (29)$$

where  $0 < a \leq 1$ . Its 4<sup>th</sup>-order truncated Maclaurin series expansion using Horner's rule is

$$e^{-ax} \approx 1 - ax + \frac{a^2}{2!}x^2 - \frac{a^3}{3!}x^3 + \frac{a^4}{4!}x^4 = 1 - ax \left( 1 - \frac{ax}{2} \left( 1 - \frac{ax}{3} \left( 1 - \frac{ax}{4} \right) \right) \right) \quad (30)$$

Fig. 18 shows the schematic of the SEXP<sub>a</sub> ( $0 < a \leq 1$ ) using 4<sup>th</sup>-order truncated Maclaurin series expansion and Horner's rule. The input bitstream is generated by comparing original random numbers from an RNG with input values, while the coefficient bitstreams are produced by sharing delayed random numbers and then comparing delayed numbers with coefficients, as suggested in [37]. Since there are several coefficient bitstreams to be generated, DFFs in every stage are inserted for decorrelation. In addition, a DFF is also inserted in each stage to decorrelate the correlation of the input bitstream. Each stage is computed to realize (30) as follows according to the method in [41], where the impact of the DFFs inserted should be considered comprehensively. For the first stage,

$$n_1^{[0]} = 1 - \frac{a^{[1]}}{4} x^{[0]}, \quad (31)$$

where a DFF is inserted to decorrelate the correlation between  $x$  and  $a/4$  and the values in the superscripts are the numbers of clock cycles delayed by random numbers or generated bitstreams. For example, 0 means the current clock cycle, and 1 represents that each bit in the bitstreams  $a/4$  is delayed by 1 clock cycle. The second and third stages are computed as

$$\begin{aligned} n_2^{[0]} &= 1 - \frac{a^{[2]}}{3} x^{[1]} n_1^{[0]} \\ &= 1 - \frac{a^{[2]}}{3} x^{[1]} \left( 1 - \frac{a^{[1]}}{4} x^{[0]} \right) \\ n_3^{[0]} &= 1 - \frac{a^{[3]}}{2} x^{[2]} n_2^{[0]} \\ &= 1 - \frac{a^{[3]}}{2} x^{[2]} \left( 1 - \frac{a^{[2]}}{3} x^{[1]} \left( 1 - \frac{a^{[1]}}{4} x^{[0]} \right) \right) \end{aligned} \quad (32)$$

Note that, the insertion of DFFs in this way in [37] incurs the correlations between  $x^{[1]}$  and  $\frac{a^{[1]}}{4}$ , and between  $x^{[2]}$  and  $\frac{a^{[2]}}{3}$  in (32). The output  $f$  of Fig. 18 is computed as

$$\begin{aligned} f^{[0]} &= 1 - a^{[4]} x^{[3]} n_3^{[0]} \\ &= 1 - a^{[4]} x^{[3]} \left( 1 - \frac{a^{[3]}}{2} x^{[2]} \left( 1 - \frac{a^{[2]}}{3} x^{[1]} \left( 1 - \frac{a^{[1]}}{4} x^{[0]} \right) \right) \right) \end{aligned} \quad (33)$$

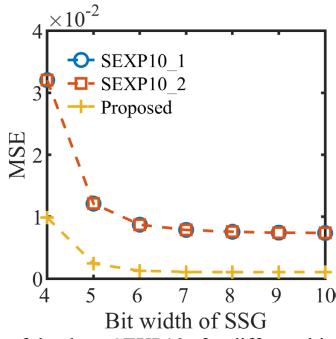


Fig. 21. The MSEs of the three SEXP10s for different bit widths of SSGs.

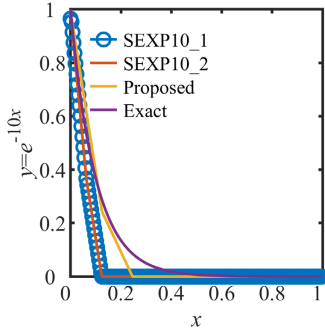


Fig. 22. The computed results for the three 4<sup>th</sup>-order SEXP10s using 8-bit SSGs.

TABLE VII  
THE HARDWARE COSTS OF THE THREE SEXP10s USING 8-BIT SSGs

Method	Area	Power	CPD	ADP	PDP
SEXP10_1	387.37	0.0302	1.00	387.37	0.0302
SEXP10_2	387.37	0.0306	1.00	387.37	0.0306
Proposed	361.44	0.0266	0.96	346.99	0.0256

Area:  $\mu\text{m}^2$ ; Power: mW; CPD: ns; ADP:  $\mu\text{m}^2\cdot\text{ns}$ ; PDP: mW $\cdot\text{ns}$ .

where the correlation between  $x^{[3]}$  and  $\frac{a^{[3]}}{2}$  and the correlations

between  $x^{[1]}$  and  $\frac{a^{[1]}}{4}$ , and between  $x^{[2]}$  and  $\frac{a^{[2]}}{3}$  introduce

deviations from the exact results in (30). That is, the implementation in Fig. 18 approximately realizes the SEXP $a$  ( $0 < a \leq 1$ ) using 4<sup>th</sup>-order truncated Maclaurin series expansion and Horner's rule to approximate SEXP $a$  ( $0 < a \leq 1$ ). By setting  $a=1$  in (30), the exponential function  $e^{-x}$  (denoted as SEXP1) is realized. Fig. 19 shows the MSEs of the SEXP1 with orders from 2 to 8, using different bit widths of SSGs and LFSRs. The design using a small bit width of RNG results in larger MSEs, such as the fluctuation of the MSEs generated by 4-bit SSGs and LFSRs. Generally, the SEXP1 with an order of 3 provides a good balance between computing accuracy and hardware costs in two cases, so it is used later.

The required  $e^{-10x}$  (denoted as SEXP10) in (7) can be rewritten as

$$e^{-10x} = (e^{-x})^{10}. \quad (34)$$

Two stochastic implementations of (34) have been proposed in [37], as shown in Fig. 20(a) and (b), where  $e^{-x}$  represents the output of the SEXP1. Eleven DFFs, two 2-input AND gates, and one 3-input AND gate are respectively used in both

designs.

An optimized implementation of the SEXP10 is proposed, as shown in Fig. 20(c), where seven DFFs, one 2-input AND gate, and two 3-input AND gates are used.

Fig. 21 shows the MSEs of the three SEXP10s using different bit widths of SSGs. The MSEs of the proposed SEXP10 are lower than those of the SEXP10\_1 and SEXP10\_2. Thus, the proposed SEXP10 outperforms the SEXP10\_1 and SEXP10\_2 in computing accuracy. Fig. 22 presents the plots for the three 3<sup>rd</sup>-order SEXP10s using 8-bit SSGs. The computed results of the SEXP10\_1 and SEXP10\_2 overlap, while the proposed SEXP10 is closer to the exact results.

TABLE VII lists the hardware costs of the three SEXP10s using 8-bit SSGs. The SEXP10\_1 and SEXP10\_2 use almost the same hardware resources, while the proposed design requires less costs than others in all respects. For example, the ADP and PDP of the proposed SEXP10 are respectively reduced by 14.42% and 15.79% on average compared to those of the SEXP10\_1 and SEXP10\_2.

#### IV. EXPERIMENTAL RESULTS

From the simulation results of the USUBs, SQRTs, and SEXP $s$ , it is obvious that an SSG cannot provide sufficient computing accuracy for the SQRTs with feedback loops. Thus, an LFSR is used in evaluating the SC architecture for the Phansalkar algorithm. Take previous stochastic components into comparison in computing accuracy as follows. The USUB<sub>Div</sub> with one loop provides much lower accuracy than the USUB<sub>It</sub> and the proposed design, so it is not considered in comparison. In addition, since the SEXP10\_1 and SEXP10\_2 share the same computing accuracy, the SEXP10\_1 is compared with the proposed design. So, two SC architectures using previous stochastic components and an SC architecture using the proposed stochastic components for the Phansalkar algorithm are built as follows.

Design\_1: the USUB<sub>It</sub>, SQRT<sub>SR</sub>, and SEXP10\_1.

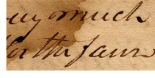
Design\_2: the USUB<sub>It</sub>, SQRT<sub>JK</sub>, and SEXP10\_1.

Proposed: the proposed USUB, SQRT, and SEXP10.

All three designs share the designed SMC49\_1.

##### A. Accuracy Comparison

Fig. 23(a) shows the original degraded image, from DIBCO, to be processed using the Phansalkar algorithm [26]. Fig. 23(b) shows the image processed by using the original expression (1) in MATLAB to provide an exact result. Fig. 23(c), (d), (e), (f), (g), (h), and (i) are obtained by using the proposed SC architecture for the Phansalkar algorithm, respectively using 4-, 5-, ..., and 10-bit LFSRs. Fig. 24 shows the corresponding MSEs for the three SC architectures, which decrease as the bit widths of LFSRs increase. The Design\_1 and Design\_2 share almost the same MSEs, which are consistently larger than those of the proposed architecture. The proposed SC architecture with 6-bit LFSRs provides an MSE of  $4.40 \times 10^{-2}$ , which is considered sufficient for the Phansalkar algorithm since an error of less than 5% is considered to be acceptable in many digital image processing algorithms [42].



(a)



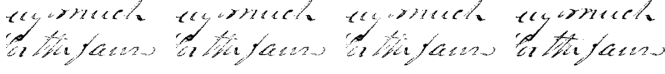
(b)

(c)

(d)

(e)

(f)



(g)

(h)

(i)

(j)

(k)

Fig. 23. (a) Original degraded image. (b) Exact computing result. Stochastic computing results with bit length (c) 16, (d) 32, (e) 64, (f) 128, (g) 256, (h) 512, (i) 1024.

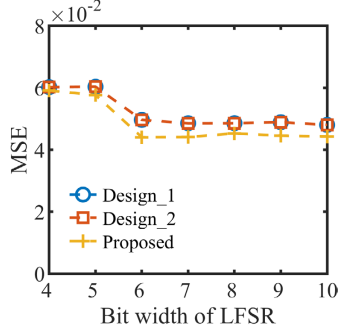


Fig. 24. The MSEs of the three SC architectures for the Phansalkar algorithm.

TABLE VIII  
THE HARDWARE MEASUREMENTS OF THE PHANSALKAR ALGORITHM

Design	Area	Power	CPD	ADP	PDP
Design_1	1833.325	0.1657	1.42	2603.32	0.2353
Design_2	1829.092	0.1641	1.38	2524.15	0.2265
Proposed	1627.819	0.1331	1.41	2295.23	0.1877
Exact	18597.15	1.7828	7.85	145987.60	13.9950

Area:  $\mu\text{m}^2$ ; Power: mW; CPD: ns; ADP:  $\mu\text{m}^2\text{-ns}$ ; PDP: mW $\cdot$ ns.

### B. Hardware Comparison

TABLE VIII lists the hardware measurements of the three designs for the Phansalkar algorithm using 8-bit LFSRs and an 8-bit binary design. Since the simplicity of the coordinate rotation digital computer (CORDIC) algorithm, it can be physically implemented by simple hardware through repeated shift-add operations and is regarded as an attractive and effective method for nonlinear functions [43]. The square root and exponential circuits in the 8-bit binary design of the Phansalkar algorithm are implemented using the CORDIC method in the hyperbolic coordinate through 16 iterations that are deemed to be sufficient enough generally [44, 45]. Experimental results show that the stochastic architecture requires much less area, CPD, and power than the binary design. For example, the SC design reduces ADP and PDP by approximately 98.43% and power by approximately 98.66% compared to the binary one. In addition, they are reduced by 10.45% and 18.18% on average compared with the Design\_1 and Design\_2. Note that the area in the proposed SC architecture includes all the D/S and S/D converters, such as LFSRs and a counter.

## V. CONCLUSION

In this paper, an efficient stochastic computing (SC) architecture is proposed for the Phansalkar algorithm to binarize degraded images and to explore the feasibility of SC in lowering hardware costs. To this end, a stochastic mean circuit (SMC), a stochastic unipolar subtractor (USUB), a stochastic square root circuit (SQRT), and a stochastic exponential circuit (SEXP) are respectively designed. Experimental results show that the proposed components outperform previous stochastic designs and the proposed SC architecture achieves high computing accuracy with fewer hardware costs compared to the binary ones. However, the energy efficiency of stochastic components is also a problem until now compared to the binary designs. In addition, an SSG generally provides higher computing accuracy than an LFSR. Its application in stochastic components with feedback loops is up in the air and the selection of DVA significantly affects the accuracy of stochastic components, which will be deeply investigated further in our future works.

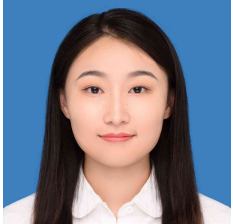
## REFERENCES

- [1] L. Nichele, V. Persichetti, M. Lucidi, and G. Cincotti, "Thresholding algorithms for microbial cell counting," in 21st International Conference on Transparent Optical Networks (ICTON), Angers, France, 2019, pp. 1-4.
- [2] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62-66, Jan. 1979.
- [3] J. Sauvola, and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognit.*, vol. 33, no. 2, pp. 225-236, Feb. 2000.
- [4] J. Bernsen, "Dynamic thresholding of gray level image," in International Conference on Pattern Recognition, Berlin, Germany, 1986, pp. 1251-1255.
- [5] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi, "A stochastic computational approach for accurate and efficient reliability evaluation," *IEEE Trans. Comput.*, vol. 63, no. 6, pp. 1336-1350, Jun. 2014.
- [6] B. Gaines, "Stochastic computing systems," *Advances in information systems science*, Advances in information systems science J. Tou, ed., pp. 37-172: Springer, Boston, MA, 1969.
- [7] W. Poppelbaum, C. Afuso, and J. Esch, "Stochastic computing elements and systems," in Proceedings of the Fall Joint Computer Conference on - AFIPS'67, Anaheim, California, 1967, pp. 635-644.
- [8] S. Aygun, M. Najafi, M. Imani, and E. Gunes, "Agile simulation of stochastic computing image processing with contingency tables," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 42, no. 10, pp. 3474-3478, Oct. 2023.
- [9] N. Temenos, and P. Sotiriadis, "Modeling a stochastic computing nonscaling adder and its application in image sharpening," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 69, no. 5, pp. 2543-2547, May. 2022.
- [10] N. Temenos, and P. Sotiriadis, "A stochastic computing sigma-delta adder architecture for efficient neural network design," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 13, no. 1, pp. 285-294, Mar. 2023.
- [11] J. Rosselló, J. Font-Rosselló, C. Frasser, A. Morán, E. Skibinsky-Gitlin, V. Canals, and M. Roca, "Highly optimized hardware morphological neural network through stochastic computing and tropical pruning," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 13, no. 1, pp. 249-256, Mar. 2023.
- [12] T. Li, W. Romaszkan, S. Pamarti, and P. Gupta, "Rex-SC: Range-extended stochastic computing accumulation for neural network acceleration," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 42, no. 12, pp. 4423-4435, Jun. 2023.
- [13] A. Morán, L. Parrilla, M. Roca, J. Font-Rosselló, E. Isern, and V. Canals, "Digital implementation of radial basis function neural networks based on stochastic computing," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 13, no. 1, pp. 257-269, Dec. 2023.
- [14] S. Wang, G. Xie, X. Cheng, and Y. Zhang, "Weighted-adder-based polynomial computation using correlated unipolar stochastic bitstreams,"

- IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 69, no. 11, pp. 4528-4532, Nov. 2022.
- [15] S. Chu, C. Wu, T. Nguyen, and B. Liu, "Polynomial computation using unipolar stochastic logic and correlation technique," *IEEE Trans. Comput.*, vol. 71, no. 6, pp. 1358-1373, May. 2021.
- [16] P. Li, D. Lilja, W. Qian, K. Bazargan, and M. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 22, no. 3, pp. 449-462, Mar. 2014.
- [17] Y. Zhang, X. Chen, J. Han, and G. Xie, "Stochastic mean circuits based on inner-product units using correlated bitstreams," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, pp. 1-1, Sep. 2023.
- [18] 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.*, vol. 24, no. 2, pp. 808-812, Feb. 2016.
- [19] N. Phansalkar, S. More, A. Sabale, and M. Joshi, "Adaptive local thresholding for detection of nuclei in diversity stained cytology images," in International Conference on Communications and Signal Processing, Kerala, India, 2011, pp. 218-220.
- [20] V. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2018, pp. 1417-1422.
- [21] A. Alaghi, W. Qian, and J. Hayes, "The promise and challenge of stochastic computing," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1515-1531, Aug. 2018.
- [22] S. Liu, and J. Han, "Toward energy-efficient stochastic circuits using parallel Sobol sequences," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 26, no. 7, pp. 1326-1339, Jul. 2018.
- [23] B. Brown, and H. Card, "Stochastic neural computation I: Computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891-905, Sep. 2001.
- [24] A. Alaghi, and J. Hayes, "Exploiting correlation in stochastic circuit design," in 2013 IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, 2013, pp. 39-46.
- [25] "The USC-SIPI image database." 1981. [Online]. Available: <https://sipi.usc.edu/database/>.
- [26] I. Pratikakis, B. Gatos, and K. Ntirogiannis, "ICDAR 2011 document image binarization contest (DIBCO 2011)," in 11th International Conference on Document Analysis and Recognition (ICDAR 2011), Beijing, China, 2011, pp. 1506-1510.
- [27] S. Wang, G. Xie, W. Xu, X. Cheng, and Y. Zhang, "High-accuracy mean circuits design by manipulating correlation for stochastic computing," *Int. J. Circuit Theory Appl.*, vol. 50, no. 10, pp. 3692-3703, Jun. 2022.
- [28] M. Steele, *The Cauchy-Schwarz master class: An introduction to the art of mathematical inequalities*, New York, NY, United States: Cambridge University Press, 2004.
- [29] Z. Zhang, W. Zhang, S. Xiong, and Y. Zhao, "An accurate and time-efficient subtractor by cross format coding in stochastic computing," in 2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Shenzhen, China, 2022, pp. 1-5.
- [30] Y. Zhou, G. Xie, J. Han, and Y. Zhang, "Absolute subtraction and division circuits using uncorrelated random bitstreams in stochastic computing," in 2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2021, pp. 1-6.
- [31] S. Liu, W. Gross, and J. Han, "Introduction to dynamic stochastic computing," *IEEE Circuits Syst. Mag.*, vol. 20, no. 3, pp. 19-33, Aug. 2020.
- [32] S. Liu, and J. Han, "Dynamic stochastic computing for digital signal processing applications," in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 604-609.
- [33] Y. Liu, and K. Parhi, "Computing polynomials using unipolar stochastic logic," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1-30, May. 2017.
- [34] D. Wu, and J. Miguel, "In-stream stochastic division and square root via correlation," in 2019 56th ACM/EDAC/IEEE Design Automation Conference, Las Vegas, NV, USA, 2019, pp. 1-6.
- [35] T. Chen, and J. Hayes, "Design of division circuits for stochastic computing," in 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, USA, 2016, pp. 116-121.
- [36] D. Wu, R. Yin, and J. Miguel, "In-stream correlation-based division and bit-inserting square root in stochastic computing," *IEEE Des. Test*, vol. 38, no. 6, pp. 53-59, Dec. 2021.
- [37] K. Parhi, and Y. Liu, "Computing arithmetic functions using stochastic logic by series expansion," *IEEE Trans. Emerging Top. Comput.*, vol. 7, no. 1, pp. 44-59, Mar. 2019.
- [38] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 93-105, Jan. 2011.
- [39] W. Qian, and M. Riedel, "The synthesis of robust polynomial arithmetic with stochastic logic," in 2008 45th ACM/IEEE Design Automation Conference, Anaheim, CA, USA, 2008, pp. 648-653.
- [40] P. Li, D. Lilja, W. Qian, K. Bazargan, and M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," in 2012 IEEE/ACM International Conference on Computer-Aided Design, New York, 2012, pp. 480-487.
- [41] Z. Li, Z. Chen, Y. Zhang, Z. Huang, and W. Qian, "Simultaneous area and latency optimization for stochastic circuits by d flip-flop insertion," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1251-1264, Jul. 2019.
- [42] P. Li, and D. Lilja, "Accelerating the performance of stochastic encoding-based computations by sharing bits in consecutive bit streams," in 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors, New York, 2013, pp. 257-260.
- [43] P. Meher, and T. Stouraitis, *Arithmetic circuits for DSP applications*, Hoboken, New Jersey: John Wiley & Sons, Inc., 2017.
- [44] P. Meher, J. Valls, T. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 56, no. 9, pp. 1893-1907, Sep. 2009.
- [45] L. Chen, "Low power designs using approximate computing and emerging memory at nanoscales," The Department of Electrical and Computer Engineering, Northeastern University, Boston, Massachusetts, 2021.



**Yongqiang Zhang** received the B.E. degree in electronic science and technology from Anhui Jianzhu University, Hefei, China, in 2013, and the Ph.D. degree in integrated circuits and systems from Hefei University of Technology, Hefei, in 2018. He was a Visiting Student with the Department of Electrical and Computer Engineering, University of Alberta, for one year. He is currently with the School of Microelectronics, Hefei University of Technology. His research interests include digital IC, inexact computing, and nanoelectronic systems.



**Jiao Qin** received the B.S. degree in electronic information science and technology from Lanzhou University of Technology, Lanzhou, China, in 2021. She is currently pursuing an M.S. degree in the new generation of information technology with the Hefei University of Technology. Her research interests include integrated circuit design and stochastic computing.



**Jie Han** received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from the Delft University of Technology, The Netherlands, in 2004. He is currently a Professor and Director of Computer Engineering in the Department of Electrical and Computer Engineering at the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computation, reliability and fault tolerance, nanoelectronic circuits and systems, and novel computational models for learning and biological applications.



**Guangjun Xie** received the Ph.D. degree in signal and information processing from the University of Science and Technology of China, Hefei, China, in 2002. He worked as a post-doctor in optics at the University of Science and Technology of China, Hefei, China, from 2003 to 2005. He was a senior visitor at IMEC in 2007 and ASIC in 2011. He is currently a Professor and Dean at the School of Microelectronics, Hefei University of Technology, Hefei, China. His research interests include VLSI design, reliability and fault tolerance, nanoelectronic circuits and systems.