A Low-Cost Stochastic Computing-based Fuzzy Filtering for Image Noise Reduction

Seyedeh Newsha Estiri*, Amir Hossein Jalilvand*, Samaneh Naderi+, M. Hassan Najafi*, Mahdi Fazeli**
*University of Louisiana at Lafayette, +Iran University of Science and Technology, **Halmstad University
Corresponding Author: seyedeh-newsha.estiri1@louisiana.edu

2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC) | 978-1-6654-6550-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/IGSC55832.2022.9969358

***Abstract*— Images are often corrupted with noise. As a result, noise reduction is an important task in image processing. Common noise reduction techniques, such as mean or median filtering, lead to blurring of the edges in the image, while fuzzy filters are able to preserve the edge information. In this work, we implement an efficient hardware design for a well-known fuzzy noise reduction filter based on stochastic computing. The filter consists of two main stages: edge detection and fuzzy smoothing. The fuzzy difference, which is encoded as bit-streams, is used to detect edges. Then, fuzzy smoothing is done to average the pixel value based on eight directions. Our experimental results show a significant reduction in the hardware area and power consumption compared to the conventional binary implementation while preserving the quality of the results.**

***Index Terms*—Stochastic computing, fuzzy logic, noise reduction, low-cost design.**

I. INTRODUCTION

Today, hardware-based data processing is bounded by some strict design constraints such as low power consumption, small circuit area, and reliability. Power and area costs, in particular, are the main concerns in designing embedded systems. Weighted binary radix has been the dominant format for the representation of data in these systems. Computation on this representation is rather complex and hence costly as each bit has its own weight according to its position. Considering the complexity of conventional binary designs, unconventional design techniques are receiving more and more attention. Stochastic computing (SC) is one of the unconventional technique that offers low-cost design and high tolerance to noise [3], [9], [20]. In SC, numbers in the $[0, 1]$ interval are presented using streams of random bits. The input data is encoded by the probability of obtaining a one versus a zero.

Complex arithmetic operations can be implemented using simple logic gates in SC. For example, multiplication operation can be performed using a single AND gate fed with uncorrelated (independent) bit-streams. This provides a significantly lower hardware cost compared to conventional binary multiplication. SC has been used for implementing low-cost designs for different application domains, from image and video processing [16] to sound processing [21], neural networks [12], sorting [13], and fuzzy [14], to name a few.

The theory of fuzzy logic [10] has been investigated in numerous applications, including control systems, real-time embedded systems, robotics, security, image and signal processing, telecommunications, decision-making support systems, and chemical industry [6]. In particular, fuzzy techniques have produced promising results for different image processing applications [15] with numerous practical works such as in industrial and medical image processing [11], [7]. In this work, as the first study of its kind, we apply the concept of SC to fuzzy logic-based filtering for image noise reduction. In contrast to the two-valued logic in the binary sets (true or false), fuzzy-logic variables have truth values in the $[0, 1]$ interval, the acceptable range of data in SC. In this work, we exploit the concept of SC for the hardware-efficient design of fuzzy filtering image noise reduction. Inspired by the fuzzy noise reduction technique of [22], our system estimates a fuzzy derivative to distinguish between local variations due to noise and image structure. Our proposed SC design takes advantage of the state-of-the-art low-discrepancy (LD) bit-streams [19] for low-latency yet high accuracy processing. Our synthesis results show a significant reduction in the hardware area, power, and energy consumption compared to the conventional binary implementation.

The rest of this paper is organized as follows. Section II presents background information on SC and the implemented fuzzy filter technique. Section III describes our proposed SC-based design. Section IV evaluates the hardware efficiency and the performance of the proposed architecture. Finally, Section V concludes the paper.

II. BACKGROUND

A. Stochastic Computing

Stochastic computing (SC) is an unconventional computing paradigm operating on random bit-streams. Independent of the length, the ratio of the number of ones to the length of the stream determines the bit-stream value. For example, 1101011101 is a representation for 0.7 in the stochastic domain. Conventionally, to convert data from conventional binary to stochastic representation, a random number from a random number source is compared with a constant number (i.e., the input data). The output of this comparison produces one bit of the bit-stream. For an N-bit bit-stream, the input number is compared with N random numbers. Implementing complex operations with simple hardware and the ability to tolerate high rates of noise are the primary advantages of SC. Minimum and maximum operations, for example, are two operations widely used in the fuzzy system with simple implementation in the stochastic domain. The minimum/maximum operation is implemented with a single `AND`/`OR` gate when fed with two *correlated* bit-streams, i.e., two bit-streams with a maximum overlap in the position of ones [3]. The conventional binary implementation of these operations, however, requires an n-bit comparator and an n-bit multiplexer (MUX). This results in a higher hardware area and power cost with the binary implementation. SC-based designs are independent of the precision of data; the same design can process input data with higher precision by processing longer bit-streams.

Conventionally, pseudo-random number generators are used in SC systems to convert data from binary to stochastic bit-streams. Recently, quasi-random number generators, such as Sobol [17] and Halton [4] sequence generators, have been used to generate high-quality low discrepancy (LD) bit-streams. LD bit-streams provide higher accuracy with significantly shorter bit-streams compared to conventional pseudo-random bit-streams. 1s and 0s are uniformly spaced in LD bit-streams, so the bit-streams do not suffer from the random fluctuations error [17]. The bit-streams converge faster to the expected results, resulting in a lower processing time and energy consumption.

In this work, we use Sobol sequences to generate LD bit-streams. The first 2^N numbers of any Sobol sequence can precisely present all possible N-bit precision numbers in the [0,1] interval. Hence, the only error in converting an N-bit precision data to a 2^N Sobol-based LD bit-stream is the quantization error [18]. Sobol sequence generators, however, are costly to implement in hardware. For a lower bit-stream generation cost compared to conventional comparator-based LD bit-stream generator that requires a costly Sobol sequence generator, we use the finite-state machine (FSM)-based LD bit-stream generator proposed in [5] to generate Sobol-based LD bit-streams.

B. Fuzzy Filtering Technique

Noise reduction with feature preservation is a fundamental problem in image processing. One of the main types of noise is additive noise. This noise is defined when a value with a specific distribution (e.g., Gaussian distribution) is added to each image pixel. The fuzzy filter of this work aims to remove the additive noise from input images. In contrast to the mean and median filter-based noise reduction techniques, which result in loss of edge information, the selected filter can preserve edge information and details of the image. The first stage for processing each image pixel is to compute a fuzzy derivative. A set of 16 *fuzzy rules* is then fired to determine a correction term for the processed pixel value. These rules use the fuzzy derivative as input. `Small`, `negative` and `positive` *membership functions* are used in our fuzzy filter. The `small` membership function can be adapted for more iterations of noise reduction. In this approach, detecting the edges near the target pixel is the first step in removing noise. Consider a 3×3 neighborhood of pixel(x, y) as shown in Fig. 1. The derivative in direction D ($D \in dir = \{NW, W, SW, S, SE, E, NE, N\}$) is defined as the difference between pixel(x, y) and its neighbor in the D direction. This derivative value is denoted by $\Delta_D(x, y)$.

NW	*N*	*NE*
W	*(x,y)*	*E*
SW	*S*	*SE*

Fig. 1. 3×3 neighborhood of pixel (x, y).

Consider an edge passing through the neighborhood of a pixel(x, y) in the $SW - NE$ direction. The derivative value $\Delta_{NW}(x, y)$ will be large, but also the derivative values of the neighboring pixels perpendicular to the edge's direction can be large. For example, in the NW direction, we can calculate $\Delta_{NW}(x, y)$, $\Delta_{NW}(x - 1, y + 1)$, and $\Delta_{NW}(x + 1, y - 1)$. The idea is to reduce the effect of one derivative value, which is high due to noise. Therefore, if two out of three derivative values are small, we can assume that no edge is present in this direction [22]. This observation will be considered when we formulate the fuzzy rule to calculate the fuzzy derivative values. We define the following membership function:

$$m(a) = \begin{cases} 1 - \frac{|a|}{Sd}, & 0 \leq a \leq Sd \\ 0, & |a| > Sd \end{cases} \tag{1}$$

where Sd is an adaptive parameter. For example, the value of the fuzzy derivative $\Delta^F_{NW}(x, y)$ for pixel(x, y) in the NW-direction is calculated by applying the following rule:

if ($\Delta_{NW}(x, y)$ is small and $\Delta_{NW}(x{-}1, y{+}1)$ is small)
or ($\Delta_{NW}(x, y)$ is small and $\Delta_{NW}(x{+}1, y{-}1)$ is small)
or ($\Delta_{NW}(x{-}1, y{+}1)$ is small and $\Delta_{NW}(x{+}1, y{-}1)$ is small)
then $\Delta^F_{NW}(x, y)$ is small.

Sd determines the spread of the `small` *membership function* and ultimately the threshold for edge detection. Instead of sampling the whole image for standard deviation (STD), we take $k \times k$ blocks of the image and find their STD. We then take the minimum STD across all blocks. We choose $K = 6$ for efficiency [2]. This minimum deviation Sd will always be less than the deviation in case of an edge. Finally, we multiply STD by an amplification parameter (α) to increase noise reduction:

$$Sd = \alpha \times \text{standard deviation} \tag{2}$$

We use a pair of fuzzy rules for each direction to compute the correction term for the processed pixel value. The idea behind these rules is as follows: if no edge is present in a specific direction, the derivative value in that direction can and will be used to compute the correction term. The first part (edge assumption) can be realized by using the fuzzy derivative value. For the second part (filtering), we must distinguish between the positive and negative values of the correction term. We can specify the following fuzzy rules to

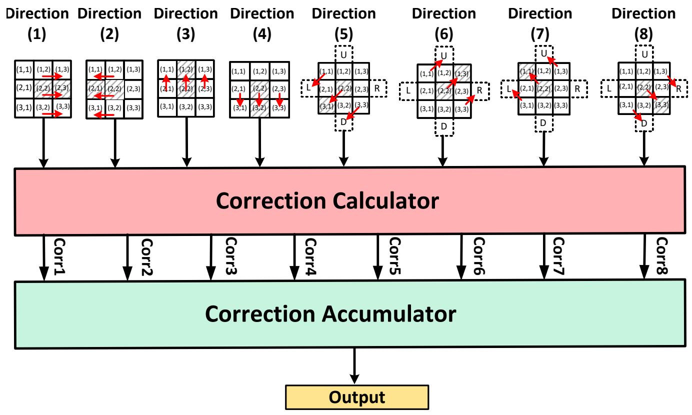


Fig. 2. General design of the fuzzy filtering technique.

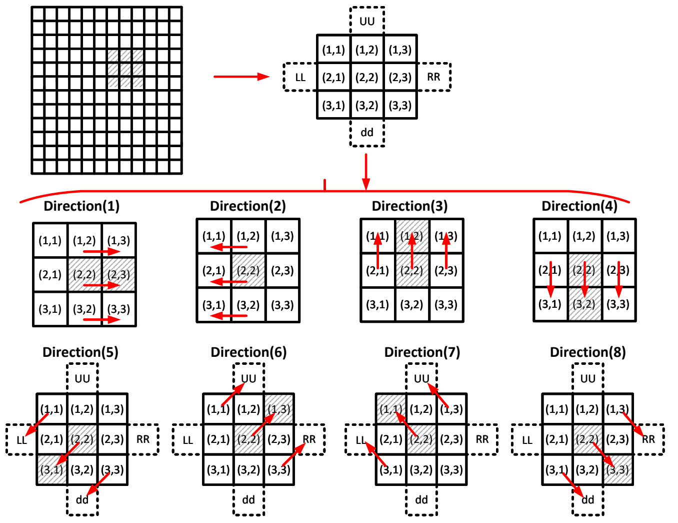


Fig. 3. An example a 3×3 block with the center pixel $(2,2)$. Eight directions with their representative pixel values are illustrated.

obtain the final correction for a direction:

C_{D_p}:*if* $\Delta^F_{NW}(x,y)$ is small and $\Delta_{NW}(x,y)$ is positive)
then c is positive
C_{D_n}:*if* $\Delta^F_{NW}(x,y)$ is small and $\Delta_{NW}(x,y)$ is negative)
then c is negative

After obtaining C_{D_p} and C_{D_n} for all directions, we can average their value to obtain the final correction:

$$\Delta C(x,y) = \Sigma_{D \in \text{directions}}(C_{D_p} - C_{D_n})/8 \qquad (3)$$

III. PROPOSED SC-BASED DESIGN

In this section, we present our proposed SC hardware architecture for the discussed fuzzy filtering noise reduction technique. As shown in Fig. 2, the design consists of two main parts: 1) correction calculator and 2) correction accumulator. We consider a 3×3 block window. For each center pixel, the *correction value* is calculated in eight directions. Fig. 3 shows an example of a 3×3 block window with eight directions. The calculated correction values of each direction are summed up together and the produced result will be added to the input pixel value.

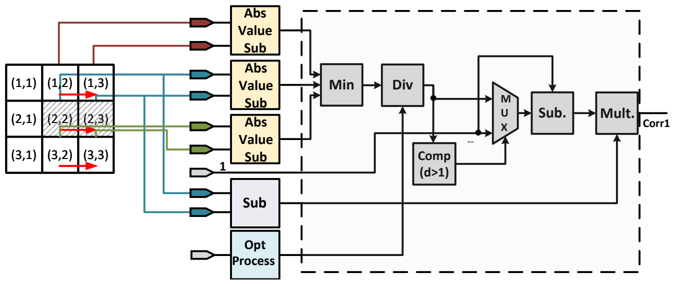


Fig. 4. Conventional Binary Implementation of the Correction Calculator.

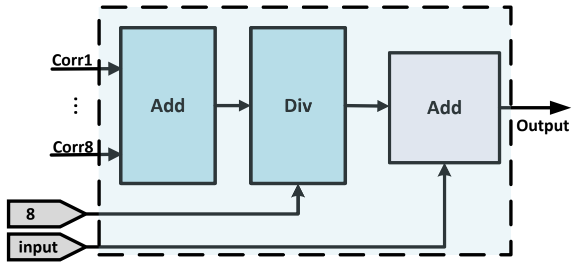


Fig. 5. Conventional Binary Implementation of the Correction Accumulator.

A. Correction Calculator

Fig. 4 shows the conventional binary architecture of the correction calculator. Fig. 6 illustrates the corresponding SC-based implementation. First, the input values (Δ(x,y) values for eight directions) are converted to stochastic bit-streams using a stochastic number generator (SNG) unit. Fig. 8 shows the structure of the SNG unit. The inputs are converted to Sobol-based LD bit-streams using the FSM-based bit-stream generator proposed in [5]. One FSM is shared for converting all inputs. However, each input is connected to a different MUX unit. Considering a 3×3 block window, we compute the fuzzy derivative values for eight directions. Also, we have an adaptive parameter, Sd. So, in total, we need 3×8+1=25 MUX units in the SNG unit. The fuzzy rule, a min-based function, is implemented using a standard `AND` gate. According to Equation 1, the `small` membership function consists of a division, a comparator, and a subtraction unit. For division, we employ a state-of-the-art SC division circuit, called CORDIV [8], that exploits the correlation between input bit-streams to realize the division operation. CORDIV not only has a lower hardware cost than the previous stochastic division circuits but also provides higher accuracy. In the conventional binary design, the comparator unit checks the division output to see if the value is less than one. A comparator unit is unnecessary in the stochastic design as the values are in the $[0,1]$ interval. The subtraction unit can be implemented with a standard `OR`-gate in the stochastic design. The last operation in the correction calculator circuit is multiplication. As shown in Equation 2, the final result is the correction component of direction D and is a fraction of Δ(x,y).

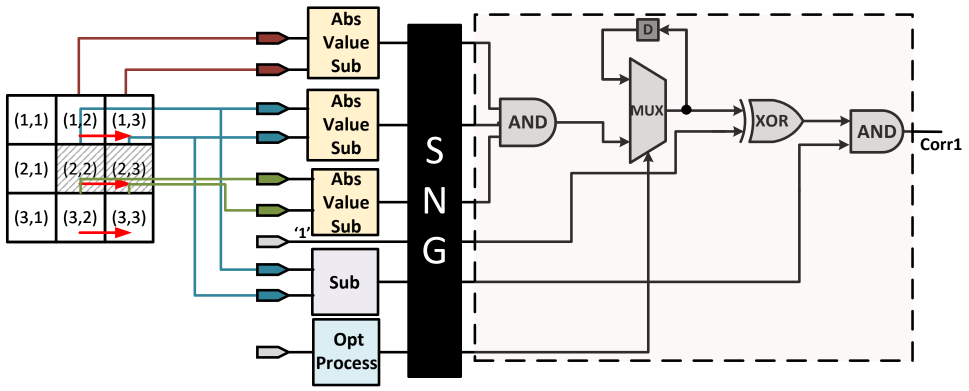


Fig. 6. Stochastic-based Implementation of the Correction Calculator.

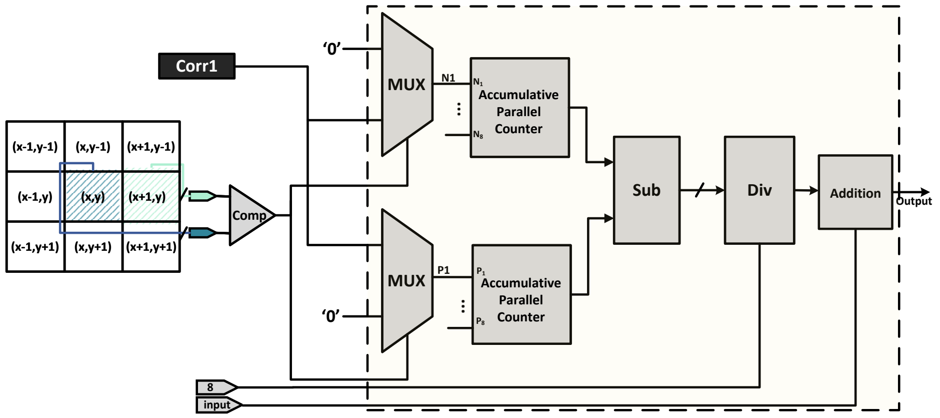


Fig. 7. Stochastic-based Implementation of the Correction Accumulator.

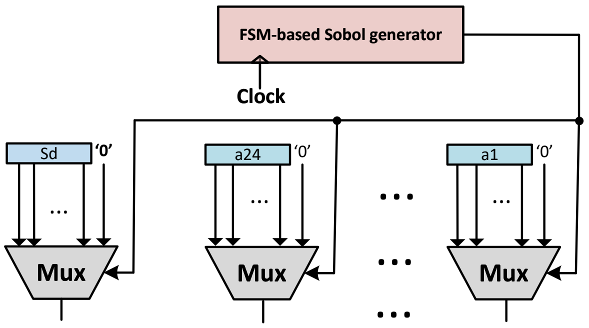


Fig. 8. SNG unit design. (a1,...,a24) are the $\Delta(x,y)$ values calculated for pixel(x,y) and its neighbouring pixels in 8 directions. The other input is the adaptive parameter Sd.

B. Correction Accumulator

Fig. 5 shows the block diagram of the conventional binary design for the correction accumulator. After calculating the correction values of all directions, we average their value to obtain the final correction value. The final value result will then be added to the pixel value. The correction values of each direction are either positive or negative depending on the sign of the Δ(x,y) value of the direction. A common approach for handling negative data in the stochastic domain is by extending the range of numbers from $[0,1]$ to $[-1,1]$ using a linear transformation in a so-called bipolar encoding [3]. Bipolar SC, however, requires twice bit-stream length and so twice processing time for the same accuracy compared to stochastic unipolar encoding. We divide the correction values into positive and negative subsets to handle negative correction values in the proposed correction accumulator. As shown in Fig. 7, a comparator is used to determine the sign of Δ(x,y). For example, if the value of pixel (x+1,y) is greater than the value of pixel (x,y), ΔE(x,y) and so the correction value are positive. In the accumulation step (Accumulative Parallel Counter (APC)), the correction values in the "positive" subset and the "negative" subset are accumulated separately using binary adders, implicitly converting them from bit-stream to binary representation. The outputs of the two APC units are then subtracted from each other. In the last step, the final correction value is divided by eight and is added to the original input pixel.

IV. Design Evaluation

In this section, we evaluate the hardware cost and the performance of the proposed SC design compared to the conventional binary implementation of the discussed fuzzy noise reduction technique.

A. Cost Comparison

For hardware cost comparison, we developed RTL VHDL descriptions for the proposed SC-based and the conventional binary design. The designs were synthesized using the Synopsys Design Compiler v2018.06 with the 45nm FreePDK

TABLE I
HARDWARE COST COMPARISON OF THE PROPOSED SC AND CONVENTIONAL BINARY DESIGN.
THE BIT-STREAM LENGTH FOR THE SC DESIGNS IS 2^{BW}, WHERE BW IS THE BIT WIDTH.

Bit Width	Area (μm^2)		Power @Max Frequency (mW)		Critical Path (ns)		Energy (pj)	
	Proposed	Conventional	**Proposed**	Conventional	**Proposed**	Conventional	**Proposed**	Conventional
2	**2,652**	30,552	**1.47**	7.97	**0.99**	5.42	**5.82**	43.24
3	**2,907**	33,209	**1.52**	8.66	**0.99**	5.42	**12.05**	47.00
4	**3,049**	34,957	**1.54**	9.07	**1.00**	5.59	**24.68**	50.74
5	**3,367**	38,841	**1.65**	10.08	**1.05**	5.83	**55.59**	58.73
6	**4,175**	51,787	**1.75**	10.81	**1.05**	5.83	**117.6**	62.98
7	**4,348**	53,389	**1.71**	11.23	**1.05**	5.83	**229.6**	65.50
8	**4,895**	59,988	**1.75**	11.54	**1.06**	5.89	**476.1**	67.98

TABLE II
ACCURACY EVALUATION (MSE) OF THE PROPOSED SC WITH DIFFERENT BIT-STREAM LENGTHS (BL) AND THE CONVENTIONAL BINARY DESIGN

	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$
Noisy image	1377	2381	3175
Conventional Binary Fuzzy Filter	310	605	883
Proposed Fuzzy Filter (BL=8)	335	661	901
Proposed Fuzzy Filter (BL=16)	321	631	897
Proposed Fuzzy Filter (BL=32)	317	629	891

library [1]. We report the synthesis results for different data bit-widths (i.e., $M = 2, 3, 4, 5, 6, 7$ and 8). Table I reports the synthesis results in terms of hardware footprint area, power consumption at maximum working frequency, critical path latency, and energy consumption. The energy consumption of the proposed design is calculated by finding power × critical path latency × number of clock cycles. As it can be seen, the proposed design achieves up to 91.8% savings in the hardware area and 84.7% reduction in power consumption. The proposed design achieves a lower area and power cost for all data bit-widths. However, in terms of energy consumption, the proposed design provides lower energy for bit-widths less than six. The energy saving rate decreases by increasing the data-width as the number of processing cycles in the proposed design increases by increasing the precision of data.

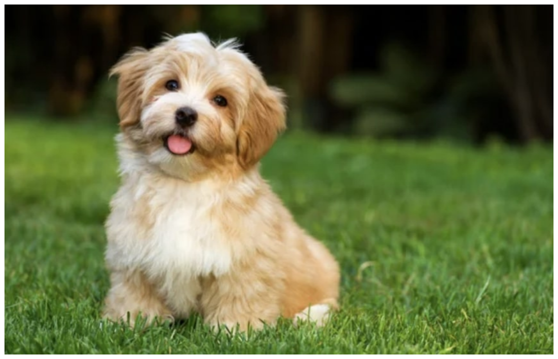


Fig. 9. Original Test Image.

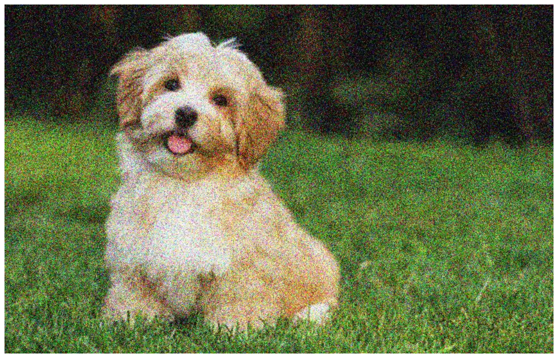
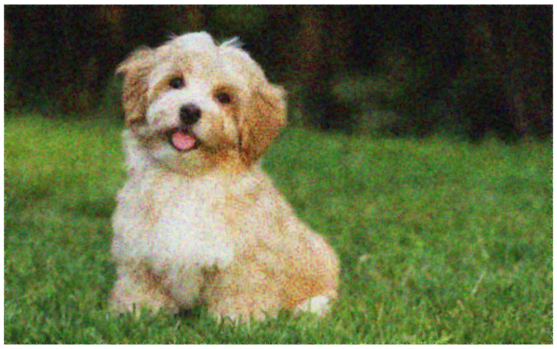


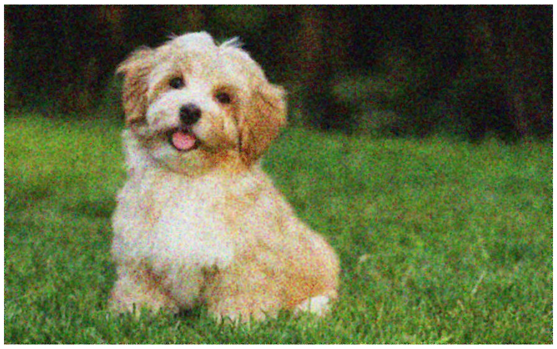
Fig. 10. Noisy image with additive Gaussian noise of variance = 10.

B. Accuracy Comparison

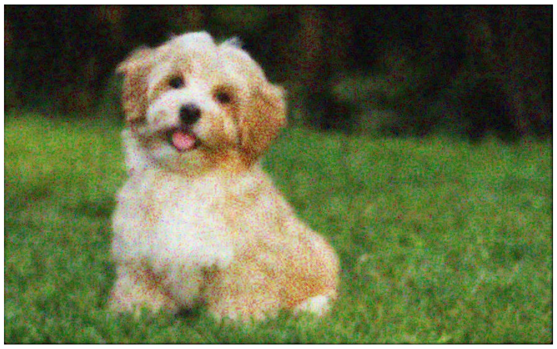
The fixed-point baseline design and the proposed bit-stream-based design were implemented in MATLAB for accuracy evaluation. Both approaches were evaluated with a test image after adding different levels of Gaussian noise. Figure 9 shows the representative test image. We corrupted the image with the variance of Gaussian noise equal to 5, 10, and 15. To evaluate the results, we calculated the mean squared error (MSE) between the original image and the filtered image. For high noise levels, we need to apply more iterations to reach an MSE close to the median filter. However, one or two iterations give us an acceptable noise reduction for low noise levels. For more noise reduction, we can increase the amplification α factor. Table II reports the MSE results. For the variance equal to 5, both the baseline and proposed stochastic filters are applied in only one iteration. Three iterations are done to achieve a lower MSE for the variance of Gaussian noise equal to 10 and 15. The proposed SC fuzzy filter design performs as well as the conventional binary-based fuzzy filter. The MSE obtained from the proposed design is negligibly higher than the baseline design, mainly due to the quantization errors. Figure 10 shows the dog image with noise $var = 10$. A 7×7 Median filter and the fuzzy filter in stochastic and binary domains are applied to the noisy image. As seen in Figure 11, the median filter was unable to preserve the image's details, such as the grass, the border of the dog body and the image background is blurred. However, both designs of the fuzzy filter were able to keep the small details, and the output is sharper.



(a)



(b)



(c)

Fig. 11. The output image after applying (a) conventional fuzzy filtering, (b) stochastic-based fuzzy filtering, and (c) median filtering (7 × 7).

V. Conclusion

This work proposed a low-cost hardware design for a fuzzy noise reduction filter based on stochastic computing. The main idea is to distinguish between local variations due to noise and due to image structures such as edges. Synthesis results confirm the efficiency of the proposed design. The stochastic-based design provides significant saving in the hardware area and power costs compared to the conventional binary implementation while preserving the quality of the results.

Acknowledgments

This work was supported in part by National Science Foundation (NSF) grant #2019511, the Louisiana Board of Regents Support Fund #LEQSF(2020-23)-RD-A-26, and generous gifts from Cisco, NVidia, and Xilinx.

References

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

[2] Noise reduction using fuzzy filtering. https://devendrapratapyadav.github.io/Fuzzy_Image_processing/, 2018.

[3] A. Alaghi, W. Qian, and J. P. Hayes. The Promise and Challenge of Stochastic Computing. *IEEE Trans. on Computer-Aided Design of Integ. Circ. and Sys.*, 37(8):1515–1531, Aug 2018.

[4] Armin Alaghi and John P Hayes. Fast and accurate computation using stochastic circuits. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–4. IEEE, 2014.

[5] Sina Asadi, M. Hassan Najafi, and Mohsen Imani. A low-cost fsm-based bit-stream generator for low-discrepancy stochastic computing. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 908–913, 2021.

[6] Zeungnam Bien and Kyung Chan Min. Fuzzy logic and its applications to engineering. *Information Sciences and Intelligent Systems. 1st Edn., Kluwer Academic Publishers, ISBN*, 10:0792337557, 1995.

[7] Sylvie Bothorel, Bernadette Bouchon Meunier, and Serge Muller. A fuzzy logic based approach for semiological analysis of microcalcifications in mammographic images. *International Journal of Intelligent Systems*, 12(11-12):819–848, 1997.

[8] Te-Hsuan Chen and John P. Hayes. Design of division circuits for stochastic computing. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 116–121, 2016.

[9] Brian R Gaines. Stochastic computing systems. *Advances in information systems science*, pages 37–172, 1969.

[10] JA Goguen. La zadeh. fuzzy sets. information and control, vol. 8 (1965), pp. 338–353.-la zadeh. similarity relations and fuzzy orderings. information sciences, vol. 3 (1971), pp. 177–200. *The Journal of Symbolic Logic*, 38(4):656–657, 1973.

[11] Lars Hildebrand and Madjid Fathi. Soft computing as a methodology for color processing. In *EUSFLAT-ESTYLF Joint Conf.*, pages 263–266. Citeseer, 1999.

[12] R. Hojabr, K. Givaki, S. R. Tayaranian, P. Esfahanian, A. Khonsari, D. Rahmati, and M. H. Najafi. Skippynn: An embedded stochastic-computing accelerator for convolutional neural networks. In *56th Design Automation Conference (DAC)*, 2019.

[13] Amir Hossein Jalilvand, Seyedeh Newsha Estiri, Samaneh Naderi, M. Hassan Najafi, and Mohsen Imani. A fast and low-cost comparison-free sorting engine with unary computing: Late breaking results. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, page 1390–1391, 2022.

[14] Amir Hossein Jalilvand, M. Hassan Najafi, and Mahdi Fazeli. Fuzzy-logic using unary bit-stream processing. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[15] Etienne E Kerre and Mike Nachtegael. *Fuzzy techniques in image processing*, volume 52. Springer Science & Business Media, 2000.

[16] Peng Li, D.J. Lilja, Weikang Qian, K. Bazargan, and M.D. Riedel. Computation on stochastic bit streams digital image processing case studies. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(3):449–462, 2014.

[17] Siting Liu and Jie Han. Energy efficient stochastic computing with sobol sequences. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 650–653. IEEE, 2017.

[18] M. Hassan Najafi, Devon Jenson, David J. Lilja, and Marc D. Riedel. Performing stochastic computation deterministically. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 27(12):2925–2938, 2019.

[19] M Hassan Najafi, David J Lilja, and Marc Riedel. Deterministic methods for stochastic computing using low-discrepancy sequences. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[20] Weikang Qian, Xin Li, Marc D Riedel, Kia Bazargan, and David J Lilja. An architecture for fault-tolerant computation with stochastic logic. *IEEE transactions on computers*, 60(1):93–105, 2010.

[21] Peter Schober, Seyedeh Newsha Estiri, Sercan Aygun, Nima TaheriNejad, and Hassan Najafi. Sound Source Localization using Stochastic Computing. In *The 2022 International Conference on Computer-Aided Design (ICCAD)*, pages 1–9, 2022.

[22] Dimitri Van De Ville, Mike Nachtegael, Dietrich Van der Weken, Etienne E Kerre, Wilfried Philips, and Ignace Lemahieu. Noise reduction by fuzzy image filtering. *IEEE transactions on fuzzy systems*, 11(4):429–436, 2003.