Gabor Filter Based on Stochastic Computation

Naoya Onizawa, *Member, IEEE*, Daisaku Katagiri, Kazumichi Matsumiya, Warren J. Gross, *Senior Member, IEEE*, and Takahiro Hanyu, *Senior Member, IEEE*

***Abstract*—This letter introduces a design and proof-of-concept implementation of Gabor filters based on stochastic computation for area-efficient hardware. The Gabor filter exhibits a powerful image feature extraction capability, but it requires significant computational power. Using stochastic computation, a sine function used in the Gabor filter is approximated by exploiting several stochastic tanh functions designed based on a state machine. A stochastic Gabor filter realized using the stochastic sine shaper and a stochastic exponential function is simulated and compared with the original Gabor filter that shows almost equivalent behaviour at various frequencies and variance. A root-mean-square error of 0.043 at most is observed. In order to reduce long latency due to stochastic computation, 68 parallel stochastic Gabor filters are implemented in Silterra 0.13 μm CMOS technology. As a result, the proposed Gabor filters achieve a 78% area reduction compared with a conventional Gabor filter while maintaining the comparable speed.**

***Index Terms*—Digital circuit implementation, stochastic computing.**

I. Introduction

THE Gabor filter [1] is a powerful tool that can extract oriented bars and edges of an image with a similar behaviour to the human visual system [2]–[4]. The Gabor filter designed using a multiplication of sin/cos function by a Gaussian function is often used for various image processing and computer vision applications, such as face recognition [5] and vehicle verification [6], [7].

However, real-time implementation of the Gabor filter is challenging, due to the complexity of the Gaussian and sine function implementation. Several approaches have been proposed, based decomposition algorithms, cellular neural networks or COordinate Rotation DIgital Computer (CORDIC) [8]–[10]. Digital hardware implementations have been shown to achieve high-speed, at the cost of large area due to the complex arithmetic function implementations. On the other hand, approaches using analog or mixed-signal circuit realizations of the Gaussian and sine functions can result in area-efficient hardware implementation [11]–[14]. However, analog implementations do not scale well to more advanced CMOS process nodes and under process variations.

In this letter we provide a design and proof-of-concept implementation of Gabor filters using *stochastic computation* [15], [16]. Stochastic computation is a purely-digital implementation technique that represents data as streams of random bits. Like analog circuits, stochastic computation can perform complex functions, such as the non-linear exponentiation function, with simple, area-efficient hardware, however it enjoys the scalability of digital circuits. To the best of our knowledge, this is the first hardware algorithm and architecture of a sine function and Gabor filter based on stochastic computation.

Manuscript received August 31, 2014; revised November 12, 2014; accepted January 09, 2015. Date of publication January 14, 2015; date of current version January 23, 2014. This work was supported by MEXT Brainware LSI Project and JSPS KAKENHI under Grant 26700003. This simulation was supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Joseph Cavallaro.

N. Onizawa is with Frontier Research Institute of Interdisciplinary Sciences, Tohoku University, Sendai, Japan (e-mail: nonizawa@m.tohoku.ac.jp).

K. Daisaku, K. Matsumiya, and T. Hanyu are with Research Institute of Electrical Communication, Tohoku University, Sendai, Japan (e-mail: katagiri@ngc.riec.tohoku.ac.jp; kmat@riec.tohoku.ac.jp; hanyu@ngc.riec.tohoku.ac.jp).

W. J. Gross is with Department of Electrical and Computer Engineering, McGill University, Montréal, QC H3A 0E9 Canada (e-mail: warren.gross@mcgill.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2015.2392123

II. Stochastic Sine Shaper

Stochastic computing was first introduced in the 1960 s [15]. There has recently been a revival of interest in this technique and stochastic implementations have been demonstrated for several applications, such as LDPC decoding [17]–[20] and image processing [21]–[23]. Stochastic computing represents information by sequences of random bits. Information is carried by the frequency of ones in a sequence. The representation is not unique; for example different sequences such as $\{1110\}$ and $\{1011\}$, represent the same information.

There are two mappings commonly used between the frequency of ones in a sequence and the represented information. For a sequence of bits $a(t)$, denote the probability of observing a '1' to be $P_a = \Pr(a(t) = 1)$. In *unipolar* coding, the represented value a is $a = P_a$, $(0 \leq a \leq 1)$. In *bipolar* coding, the represented value a is $a = (2 \cdot P_a - 1)$, $(-1 \leq a \leq 1)$. In this letter, the bipolar format is used. Fig. 1(a) shows a bipolar stochastic multiplier, which is simply a two-input XNOR gate [16]. The stochastic bit streams are generated using a digital-to-stochastic converter shown in Fig. 1(b), where the random number generator is often realized using a linear-feedback shift register. The input variable, a, is compared with a random number that generates a stochastic bit stream.

Stochastic computing, as originally proposed, uses tens or hundreds of clock cycles to perform an operation on bit sequences. However recent work has resulted in several stochastic hardware implementations achieve similar speed to conventional digital implementations by reducing the number

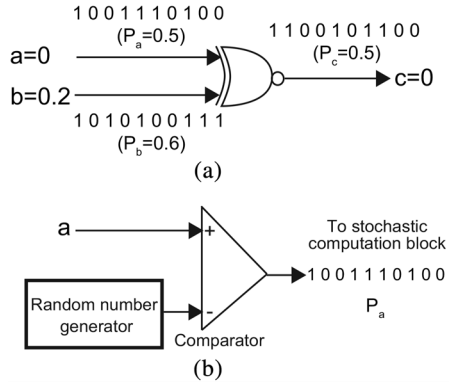

Fig. 1. Stochastic computation: (a) bipolar stochastic multiplier, and (b) digital-to-stochastic converter.

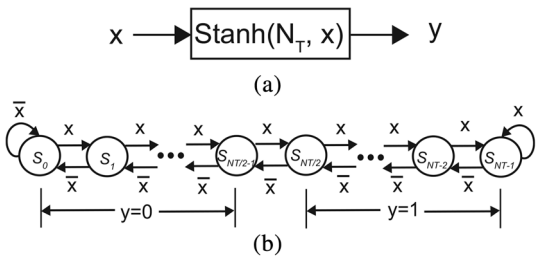

Fig. 2. Stochastic tanh function: (a) block diagram and (b) state transition diagram.

of required clock cycles, and maintain the same functionality [20], [22].

To implement a Gabor filter, we need to find efficient implementation of Gaussian and sine functions [2]. In this section, a sine shaper based on stochastic computation is presented and in the next section, the exponential function is considered. Inspired by the analog implementation of a sine shaper in [24], we can approximate the sine function with several tanh functions as:

$$\alpha \sin\left(\frac{1}{\beta}x\right) \approx \sum_{k=-\infty}^{\infty} (-1)^k \tan\mathrm{h}\left(\frac{1}{\beta}(x+\pi\beta k)\right), \quad (1)$$

where x is the input variable, α is a shape-fitting constant, and β is a constant that determines the cycle length of the sine

The tanh function can be approximated by a bipolar stochastic circuit that implements the function [16], [25]:

$$\mathrm{Stanh}(N_T, x) \approx \tan\mathrm{h}(xN_T/2), \quad (2)$$

and is implemented using a state machine with N_T states as shown in Fig. 2 (see detail in [16]). The hardware representation of the Stanh function is a $\lceil \log_2 N_T \rceil$-bit up/down counter. Using (1) and (2), the sine function is described as follows:

$$\alpha \sin\left(\frac{1}{\beta}x\right) \approx \sum_{k=\lceil -\frac{1}{\pi\beta} \rceil}^{\lfloor \frac{1}{\pi\beta} \rfloor} (-1)^k \mathrm{Stanh}\left(\frac{2}{\beta}, (x+\pi\beta k)\right), \quad (3)$$

where $-1 \leq x \leq 1$.

The summation is realized using a scaled addition [16] and is described as follows:

$$y = P_S \cdot a + (1 - P_S) \cdot b, \quad (4)$$

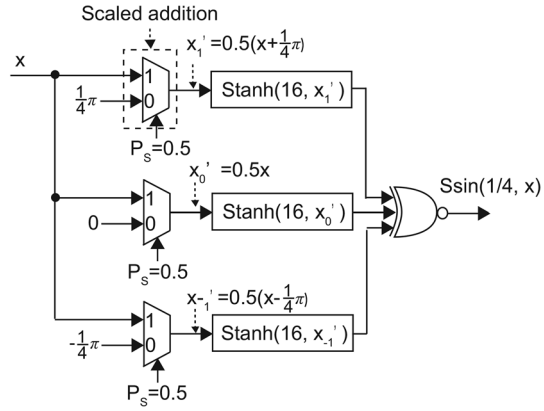

Fig. 3. Block diagram of the proposed stochastic sine shaper, where β is 1/4.

where a and b are input variables, y is an output variable, and P_S is a probability of selecting a as the output. Using (3) and (4), the sine function is described as follows:

$$\alpha \sin\left(\frac{1}{\beta}x\right) \approx \sum_{k=\lceil -\frac{1}{\pi\beta} \rceil}^{\lfloor \frac{1}{\pi\beta} \rfloor} (-1)^k \mathrm{Stanh}\left(\frac{4}{\beta}, (0.5(x+\pi\beta k))\right). \quad (5)$$

Now, we define the proposed stochastic sine shaper, Ssin(β, x), that is described as follows:

$$\mathrm{S}\sin(\beta, x) = \sum_{k=\lceil -\frac{1}{\pi\beta} \rceil}^{\lfloor \frac{1}{\pi\beta} \rfloor} (-1)^k \mathrm{Stanh}\left(\frac{4}{\beta}, (0.5(x+\pi\beta k))\right), \quad (6)$$

where $\beta = 4/N_T$. Note that a stochastic cos shaper can be also designed by replacing $(x + \pi\beta k)$ to $(x + \frac{\pi}{2}\beta(k+1))$ in equation (6).

Fig. 3 shows a block diagram of the proposed stochastic sine shaper, where $\beta = 1/4$. The value of $x + \pi\beta k$ is determined using the scaled addition realized using a multiplexor (see detail in [16]). The three Stanh function blocks are implemented with k ranging from -1 to 1. The Stanh function block is realized using a 16-state finite-state-machine implemented as a 4-bit up/down counter. The output of one of three blocks changes depending on x as each Stanh function performs in a different region of x. The outputs of the other two blocks are fixed to -1 or 1 that determines the sign of the output of the stochastic sine shaper. Hence, the summation of the outputs of the three Stanh functions can be simply realized using a three-input XNOR gate. Note that the summation is realized using an XNOR gate when $\lfloor \frac{1}{\pi\beta} \rfloor$ is odd and it is realized using an XOR gate when even.

Fig. 4 shows simulation results of the original sine function and the proposed stochastic sine shaper, where the curve-fitting variable, α, is set to 0.85. The simulation results are obtained using MATLAB. The frequency of the stochastic sine shaper is controlled by β in the Stanh function. The simulation results show that the stochastic sine shaper generates almost the same value of the original sine function. A RMS error between the original and the stochastic sine functions is summaized as a

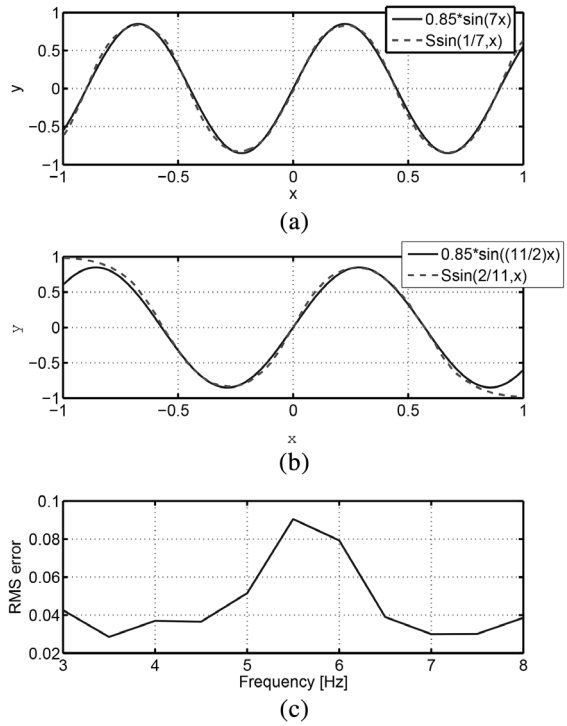

Fig. 4. Simulation results of the proposed stochastic sine shaper, where α is 0.85, with the original sine function: (a) Ssin(1/7,x), (b) Ssin(2/11,x) and (c) RMS error vs. frequency of sine function.

function of the frequency in Fig. 4(c). The RMS error at 5.5 Hz is the worst value because the sine wave of the proposed sine shaper is not generated at the edges as shown in Fig. 4(b). This situation occurs when $(\frac{1}{\pi\beta} - \lfloor\frac{1}{\pi\beta}\rfloor)$ is more than 0.5. However, the value differences at the edges are diminished by a Gaussian function of a Gabor filter.

III. Stochastic Gabor Filter

A one-dimensional sine- (or odd-) phase Gabor-filter function is a sine function multiplied by the Gaussian function [2] described as follows:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-x^2}{2\sigma^2}\right) \sin(2\pi f x), \tag{7}$$

where f is the carrier frequency for which this filter gives the greatest output and σ is the spread of the Gaussian function.

Stochastic implementations of the exponential function are presented in [16], [25] and like the tanh function, are realized using a state machine as shown in Fig. 5. The input signal, x, is encoded in the bipolar format and the output signal, y, is encoded in the unipolar format. G is a positive integer with $G \ll N_E$, where N_E is the number of states in the stochastic exponential function. The stochastic exponential function is approximated as follows:

$$y = \exp^{-2Gx} \approx \mathrm{SExp}(N_E, G, x). \tag{8}$$

To match the degrees of two exponential functions in equations (7) and (8), the input signal, x, with a scaling constant, γ, is squared and is then used in the SExp function as follows:

$$y = \exp^{-2G\gamma^2 x^2} \approx \mathrm{SExp}(N_E, G, \gamma^2 x^2). \tag{9}$$

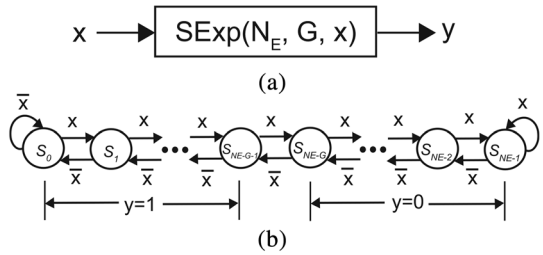

Fig. 5. Stochastic exponential function: (a) block diagram and (b) state transition diagram.

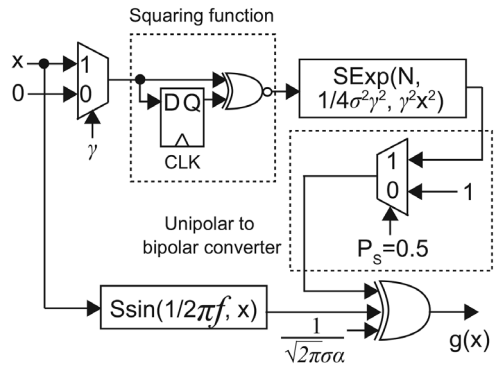

Fig. 6. Block diagram of the proposed stochastic Gabor filter.

As y is encoded using the unipolar format, the coding format of y is converted from unipolar to bipolar for the stochastic sine shaper realized using bipolar coding. y' in the bipolar format is described as follows:

$$y' \approx (\mathrm{SExp}(N_E, G, \gamma^2 x^2) + 1)/2. \tag{10}$$

Using equations (5), (6) and (10), the Gabor filter is approximated as follows:

$$g(x) \approx \frac{1}{\sqrt{2\pi}\sigma\alpha} \frac{\mathrm{SExp}(N_E, \frac{1}{4\sigma^2\gamma^2}, \gamma^2 x^2) + 1}{2} \cdot \mathrm{S}\sin\left(\frac{1}{2\pi f}, x\right). \tag{11}$$

Fig. 6 shows a block diagram of the proposed stochastic Gabor filter. Both input signal, x, and output signal, $g(x)$, are encoded using the bipolar format. x is scaled by γ and is then squared using a stochastic squaring circuit presented in [16]. The format of the output of the SExp function is converted from unipolar to bipolar using a unipolar-to-bipolar converter realized using the scaled addition. Simultaneously, the Ssin function is performed with $\beta = 1/2\pi$f. At the end, $g(x)$ is generated using a three-input multiplication realized by the three-input XOR gate.

Fig. 7(a) and (b) show simulated results of the original and the proposed stochastic Gabor filters at different frequencies and variances, where α is 0.9 and N_E is 64. The simulation results are obtained using MATLAB. γ can be selected to determine G in the stochastic exponential function while maintaining the same σ. The simulation results show that the stochastic Gabor filter approximates the original Gabor filter. The RMS errors are 0.043 in Fig. 7(a) and 0.040 in Fig. 7(b) when γ^2 is 1. Fig. 7(c) summarizes a relationship between RMS errors and N_E when f is 1.04. The simulation results show that large γ and N_E achieve small RMS errors.

Table I shows performance comparisons of Gabor filters. The proposed Gabor filter is designed based on parameters used in

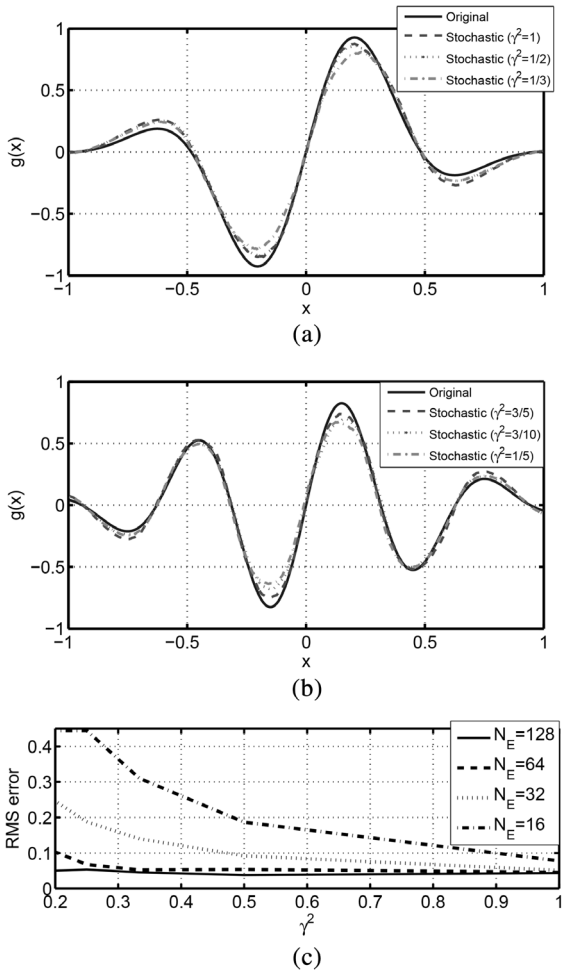

Fig. 7. Simulation results of the original and the proposed stochastic Gabor filters, where α is 0.9: (a) $f = 1.04$, $\sigma^2 = 0.125$ and $N_E = 64$, (b) $f = 1.59$, $\sigma^2 = 0.208$, and $N_E = 64$, and (c) RMS errors vs. γ^2 when f is 1.04.

TABLE I
PERFORMANCE COMPARISONS IN ASIC DESIGN

	This work		CORDIC [10]
	1 parallel	68 parallel	
Technology [μm]	0.13		
Clock frequency [MHz]	758		250
Delay [ns/pixel]	13,192	194	196
# gates	301	20,468	653,312
Kernel size	1x7		7x7
# gates/kernel	43	2,924	13,333

Fig. 7(b). In addition, 68 parallel stochastic Gabor filters are designed to reduce the long latency due to stochastic computation. The proposed Gabor filters are synthesized using Synopsys Design Compiler in Silterra 0.13 μm CMOS technology, where the number of clock cycles that is equivalent to the stochastic bit length is 10,000. The number 10,000 was chosen just for demonstration purposes that exhibit almost the same values as the original Gabor filter. Future work will explore the minimizing the number of clock cycles using several useful techniques for stochastic computation presented in [20], [22]. In related work, [9] realizes a high-speed Gabor filtering using large number of multipliers and memories, but it lacks the flexibility of local frequency and orientation of an image and the kernel size is limited to 3×3. In [10], a conventional CORDIC-based

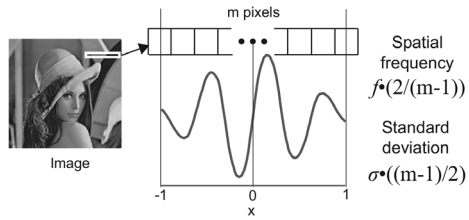

Fig. 8. Example of the spatial frequency and the standard deviation of the stochastic Gabor filter exploited in image processing. They have two degrees of freedom: f and σ in equation (11), and a pixel length, m, assigned to -1 to 1 of an input signal in stochastic computation, where m is odd.

Gabor filter is designed with small area while realizing kernel sizes and the flexibility, where the conventional filter is two-dimentional. As the proposed Gabor filter presented in this paper is one-dimentional. the area overhead from 1-D to 2-D is estimated. Suppose the area ovehead is negligibly small for the performance comparison as the proposed 1-D Gabor filter can be extended to the 2-D Gabor filter by adding two scaled additions and a squaring circuit. As a result, the 68 parallel stochastic Gabor filters achieve a 78% area reduction while maintaining the comparable delay time compared with the conventional Gabor filter.

Fig. 8 shows an example of the spatial frequency and the standard deviation of the stochastic Gabor filter. The spatial frequency is defined as the number of light and dark regions imaged within a given distance. In the stochastic Gabor filter, different lengths of pixels of an image can be assigned to -1 to 1 of an input signal, x. Suppose that the stochastic Gabor filter is applied to m pixels of an image, where m is odd. The spatial frequency is $f \cdot (2/(m-1))$ and the standard deviation is $\sigma \cdot ((m-1)/2)$. As a conclusion, the spatial frequency and the standard deviation of the stochastic Gabor filter exploited in image processing have two degrees of freedom: f and σ in equation (11), and a pixel length assigned to -1 to 1 of an input signal in stochastic computation.

IV. CONCLUSION

In this letter, the Gabor filter based on stochastic computation has been proposed for area-efficient hardware implementation. A stochastic sine shaper is approximated by using several stochastic tanh functions designed based on a state machine, where the state machine is simply realized using an up/down counter in hardware. The stochastic Gabor filter realized using the stochastic sine shaper and the exponential function is simulated and compared with the original Gabor filter that shows almost equivalent behaviours at different frequencies and variances are generated, where the RMS error of 0.043 at most is observed. The proposed filter implemented in Silterra 0.13 μm CMOS technology achieves a 78% area reduction compared with the conventional Gabor filter based on CORDIC while maintaining the comparable speed.

In future work, the stochastic Gabor filter will be extended to a two-dimensional Gabor filter for extracting oriented bars and edges of an image. In addition, the stochastic sine shaper would be useful to implement other algorithms that require a sinusoidal wave in hardware.

REFERENCES

[1] D. Gabor, "Theory of communications," *J. Inst. Elect. Eng. III, Radio Commun. Eng.*, vol. 93, no. 26, pp. 429–441, Nov. 1946.

[2] S. Marcelja, "Mathematical description of the responses of simple cortical cells," *J. Opt. Soc. Amer.*, vol. 70, no. 11, pp. 1297–1300, Nov. 1980.

[3] D. J. Heeger, "Model for the extraction of image flow," *J. Opt. Soc. Amer. A*, vol. 4, no. 8, pp. 1455–1471, Aug. 1987.

[4] G. DeAngelis, I. Ohzawa, and R. Freeman, "Depth is encoded in the visual cortex by a specialized receptive field structure," *Nature*, vol. 352, no. 11, pp. 156–159, Jul. 1991.

[5] J. Wu, G. An, and Q. Ruan, "Independent gabor analysis of discriminant features fusion for face recognition," *IEEE Signal Process. Lett.*, vol. 16, no. 2, pp. 97–100, Feb. 2009.

[6] Z. Sun, R. Miller, G. Bebis, and D. DiMeo, "A real-time precrash vehicle detection system," in *Proc. Sixth IEEE Workshop on Applications of Computer Vision, 2002 (WACV 2002)*, 2002, pp. 171–176.

[7] J.-M. Guo, H. Prasetyo, and K. Wong, "Vehicle verification using gabor filter magnitude with gamma distribution modeling," *IEEE Signal Process. Lett.*, vol. 21, no. 5, pp. 600–604, May 2014.

[8] X. Wang and B. Shi, "GPU implementation of fast gabor filters," in *Proc. 2010 IEEE Int. Symp. Circuits and Systems (ISCAS)*, May 2010, pp. 373–376.

[9] E. Cesur, N. Yildiz, and V. Tavsanoglu, "On an improved FPGA implementation of CNN-based Gabor-type filters," *IEEE Trans. Circuits Systems. II: Exp. Briefs*, vol. 59, no. 11, pp. 815–819, Nov. 2012.

[10] L. Jun-bao, W. Shuai, L. Yi, H. Jun, and Z. Xiao-Yang, "Configurable pipelined gabor filter implementation for fingerprint image enhancement," in *10th IEEE Int. Conf. Solid-State and Integrated Circuit Technology (ICSICT)*, Nov. 2010, pp. 584–586.

[11] B. Shi, T. Choi, and K. Boahen, "On-off differential current-mode circuits for gabor-type spatial filtering," *2002 IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 2, pp. II-724–II-727, 2002, Vol.2.

[12] F. Borghetti, P. Malcovati, and F. Maloberti, "A current-mode 64×1 programmable gabor filter for early vision systems," in *2001 Proc. 27th Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2001, pp. 205–208.

[13] M. Tuckwell and C. Papavassiliou, "An analog gabor transform using sub-threshold 180-nm CMOS devices," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 56, no. 12, pp. 2597–2608, Dec. 2009.

[14] T. Morie, J. Umezawa, and A. Iwata, "A pixel-parallel image processor for gabor filtering based on merged analog/digital architecture," in *Dig. Techn. Papers 2004 Symp. VLSI Circuits*, Jun. 2004, pp. 212–213.

[15] B. R. Gaines, "Stochastic computing systems," *Adv. Inf. Syst. Sci. Plenum*, vol. 2, no. 2, pp. 37–172, 1969.

[16] B. Brown and H. Card, "Stochastic neural computation. I. computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[17] V. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," *Electron. Lett.*, vol. 39, no. 3, pp. 299–301, Feb. 2003.

[18] S. Sharifi Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 10, no. 10, pp. 716–718, Oct. 2006.

[19] S. Sharifi Tehrani, S. Mannor, and W. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5692–5703, Nov. 2008.

[20] S. Sharifi Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4883–4896, Sep. 2010.

[21] P. Li and D. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *29th IEEE Int. Conf. Computer Design (ICCD)*, Oct. 2011, pp. 154–161.

[22] A. Alaghi, C. Li, and J. Hayes, "Stochastic circuits for real-time image-processing applications," in *50th ACM/EDAC/IEEE Design Automation Conf. (DAC)*, May 2013, pp. 1–6.

[23] 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.

[24] B. Gilbert, "Circuits for the precise synthesis of the sine function," *Electron. Lett.*, vol. 13, no. 17, pp. 506–508, Aug. 1977.

[25] P. Li, D. Lilja, W. Qian, M. Riedel, and K. Bazargan, "Logical computation on stochastic bit streams with linear finite state machines," *IEEE Trans. Comput.*, vol. 63, no. 6, pp. 1474–1486, Jun. 2014.