Adaptive Filter Design using Stochastic Circuits

Honglan Jiang*, Chengkun Shen†, Pieter Jonker‡, Fabrizio Lombardi§ and Jie Han*
*Department of Electrical and Computer Engineering
University of Alberta, Edmonton, AB T6G 1H9, Canada
Email: honglan@ualberta.ca, jhan8@ualberta.ca
†Harbin Institute of Technology, Harbin, Heilongjiang, China
‡Delft University of Technology, The Netherlands
§ Department of Electrical and Computer Engineering
Northeastern University, Boston, USA
Email: lombardi@ece.neu.edu.

***Abstract*—This paper proposes the design of an adaptive filter in stochastic circuits. The proposed circuit requires lower area and power than a conventional stochastic implementation. In the proposed design, the stochastic multiplier is implemented by an XNOR gate, as in a conventional scheme. However, the stochastic adder based on a multiplexer is not a very efficient implementation due to the three required stochastic number generators (SNGs) and the iterative operation required in the adaptive filter. Thus, a novel stochastic adder using a counter and a post processing unit is proposed. This adder avoids the use of SNGs, therefore it incurs a smaller area and power, while operating faster than the conventional (multiplexer-based) stochastic adder. In terms of accuracy and hardware efficiency, simulation results show that the adaptive filter using the proposed stochastic design outperforms the conventional stochastic implementation using linear feedback shift register (LFSR) based SNGs. Specifically, the proposed design consumes 35.81% less dynamic power and 21.34% less area than an LFSR-based implementation at a slightly higher accuracy.**

I. Introduction

An adaptive filter is a negative feedback system consisting of a linear filter with variable parameters adjusted by an adaptive algorithm. It is commonly used in applications such as image processing, signal prediction, system identification and echo suppression [1]. However, the hardware implementation of an adaptive filter is very complex because of the closed-loop adaptive process and the adaptive algorithm [2]. Therefore, it is important to reduce the area and power of an adaptive filter for applications in mobile devices. Albeit involved in a rather complicated computational process, the basic units in an adaptive filter are adders and multipliers. Since adders and multipliers can be efficiently implemented in stochastic circuits [3], stochastic adders and multipliers are natural candidates for consideration in the design of an adaptive filter.

In stochastic computing, a number is encoded into a random binary bit stream using a stochastic number generator (SNG). The SNG is a basic element in stochastic computing; a widely used SNG consists of a random number generator (RNG) and a comparator (Fig. 1). Unipolar and bipolar representations are the two widely used formats for stochastic numbers. In the unipolar representation, a real number $x \in [0, 1]$ is represented by the probability of observing a '1' at a bit position of a stochastic sequence. For the bipolar representation, a real number $y \in [-1, 1]$ is mapped from the unipolar representation by $y = 2x - 1$ [4].

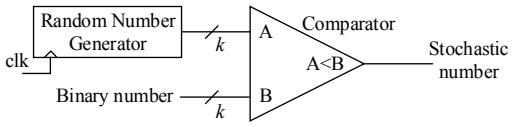
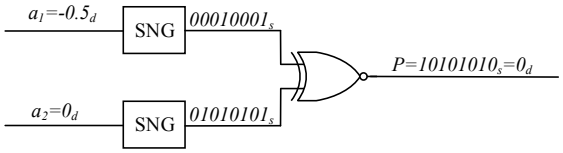


Fig. 1. A stochastic number generator.

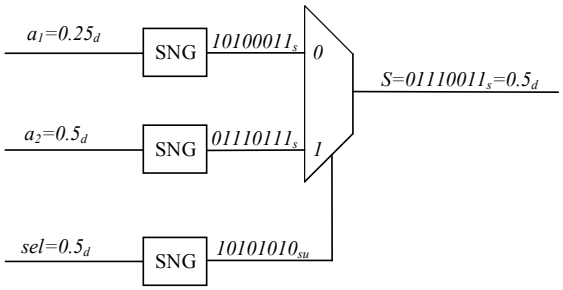
Stochastic addition and multiplication can be simply implemented by a multiplexer and an AND (for the unipolar representation) or an XNOR gate (for the bipolar representation). Fig. 2 shows the implementations of the stochastic multiplication and addition for the bipolar representation. The accuracy of a stochastic computing result is limited by the resolution and random fluctuations in stochastic sequences [5]. The random fluctuation is determined by the type of stochastic sequences. The resolution is defined as the minimum value that a stochastic sequence can represent. For a stochastic sequence (in bipolar representation) with a length of l bits, its resolution is $2/l$. For example, the length of the stochastic sequence in Fig. 2 is $l = 8$ and its resolution is $2/8 = 0.25$. For the stochastic multiplier in Fig. 2(a), therefore, the resolution of the multiplication result is 0.25. However, the resolution for the stochastic adder in Fig. 2(b) is reduced by $2\times$ because the addition result of the stochastic adder is $S = (a_1 + a_2)/2$, and the exact addition result is $2S$. Hence, the minimum value that can be represented in the addition result is 0.5, i.e., $2\times$ of the minimum value for the multiplier.

As shown in Fig. 2(a), a stochastic multiplier can effectively reduce the circuit area and power consumption compared to a binary array multiplier. However, a stochastic adder is not a very efficient implementation because it requires three SNGs that can be very complex compared to a binary adder. Additionally, as the resolution is reduced by $2\times$ in a stochastic adder, the resolution for an adder tree in an adaptive filter would be $2^K\times$ worse, where K is the number of stages of the adder tree. Therefore, the addition result from the adder in Fig. 2(b) cannot be directly used in the next iteration of the adaptive filter.

In this paper, a novel stochastic adder using a counter and a post processing unit is proposed. This adder avoids the use of SNGs with no loss on the resolution, therefore it incurs a smaller area and power dissipation, while still operating faster than the conventional multiplexer-based stochastic adder.



(a) A stochastic multiplier based on an XNOR gate. $P = a_1 a_2$.



(b) A stochastic adder based on a multiplexer. $S = (a_1 + a_2)/2$

Fig. 2. The stochastic (a) multiplier and (b) adder. The numbers with the subscripts of d, s and su are numbers in decimal, the bipolar stochastic and the unipolar stochastic formats, respectively. The result of the adder is not accurate due to random fluctuations and the limited resolution in the stochastic sequences.

Moreover, low discrepancy (LD) sequences [6] used in quasi-Monte Carlo (QMC) simulation rather than the commonly used pseudo-random sequences (usually generated by an LFSR) are applied. The LD sequence has a better uniformity, thus it suffers less from random fluctuations. In this paper, the Halton sequence is used due to its low complexity compared to other LD sequences [7].

The adaptive filter is then applied to system identification as an application. The simulation results show that the proposed design consumes significantly less power and requires a smaller area than the conventional stochastic implementation using multiplexer-based adders and LFSR-based SNGs. Furthermore, performance and energy consumption of the proposed design are also significantly better than the conventional design.

This paper is organized as follows. Section II presents the design of the adaptive filter using a stochastic multiplier and the newly proposed adder. Section III shows the simulation results of the stochastic adaptive filter for hardware and accuracy; a comparison with the conventional stochastic implementation is also pursued. The paper concludes in section IV.

II. Stochastic Adaptive Filter Design

An adaptive filter uses a negative feedback loop to adjust its parameters based on an adaptive algorithm. Fig. 3 shows the basic structure of an adaptive filter, in which the linear filter is implemented as a finite impulse response (FIR) filter by [8]

$$y(n) = \mathbf{x}(n) \cdot \mathbf{w}(n) = \sum_{i=0}^{M-1} w_i(n) \cdot x(n-i), \qquad (1)$$

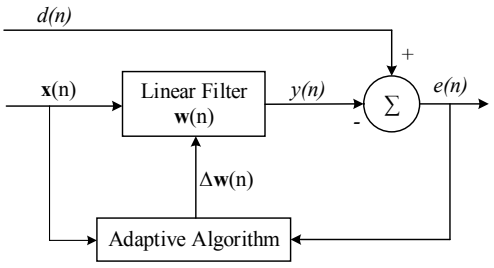


Fig. 3. An adaptive filter [10]. n is the iteration number, $\mathbf{x}(n)$ is the input vector, $y(n)$ is the output signal, $d(n)$ is the interfered desired signal with the undesired noise, $e(n)$ is the error output, and $\mathbf{w}(n)$ is the weight vector.

where $\mathbf{x}(n) = [x(n), x(n-1), \cdots, x(n-M+1)]$ is the input vector, $\mathbf{w}(n) = [w_0(n), w_1(n), \cdots, w_{M-1}(n)]^T$ is the weight vector at the n^{th} iteration, and M is the length of $\mathbf{w}(n)$. The weights of the linear filter are variable with the iteration number n as determined by the adaptive algorithm. They are updated until a set of optimized values are obtained. There are many adaptive algorithms, e.g. the least mean square (LMS), the normalized LMS (NLMS) and the recursive LMS (RLMS) algorithms [9]. The selection of an algorithm is based on a tradeoff between the computational complexity and the convergence speed. As the LMS algorithm is very simple and thus suitable for a hardware implementation, it is utilized in this paper. The LMS algorithm is formulated as

$$w_i(n+1) = w_i(n) + \mu \cdot e(n) \cdot x(n-i), i = 0, 1, \cdots, M-1 \quad (2)$$

where μ is the step size, and $e(n) = d(n) - y(n)$ is the error signal between the interfered (desired) signal $d(n)$ (with the undesired noise) and the filter output $y(n)$.

As per (1) and (2), a functional iteration of the adaptive filter (i.e. the update process of the weights) can be divided into a linear filter and LMS sections, which are implemented by multipliers and adders (as shown in Fig. 4). In Fig. 4, the error signal is computed by an adder tree within the linear filter. $3M$ multipliers and $2M$ adders (M adders for the adder tree in the linear filter and M adders for implementing the LMS algorithm) are required for each iteration; this process consumes a significant amount of power and incurs a large area for a binary implementation. Therefore, power and area efficient stochastic computing circuits are considered next. The bipolar representation is applied because both positive and negative numbers are processed by an adaptive filter.

A. Design of the Linear Filter

The multipliers in the linear filter are replaced by stochastic multipliers implemented by XNOR gates. In addition to the XNOR gate, two SNGs are required to convert binary numbers to stochastic numbers [11]. Stochastic circuits using Halton sequences have been shown to be faster and more accurate than those using traditional LFSR-based sequences [7]. Therefore, the Halton sequence is used as the stochastic sequence in this paper. Two RNGs are shared among the SNGs of the M multipliers (operating in parallel) to reduce the hardware overhead at no loss of accuracy because correlation will not

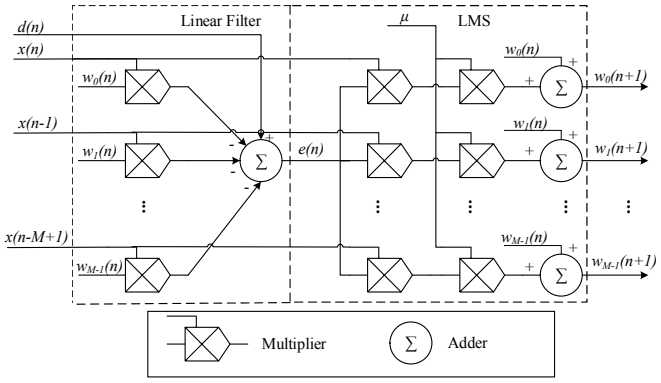


Fig. 4. An iteration of the adaptive filter implemented by multipliers and adders. The addition in the linear filter is implemented by an adder tree consisting of M adders.

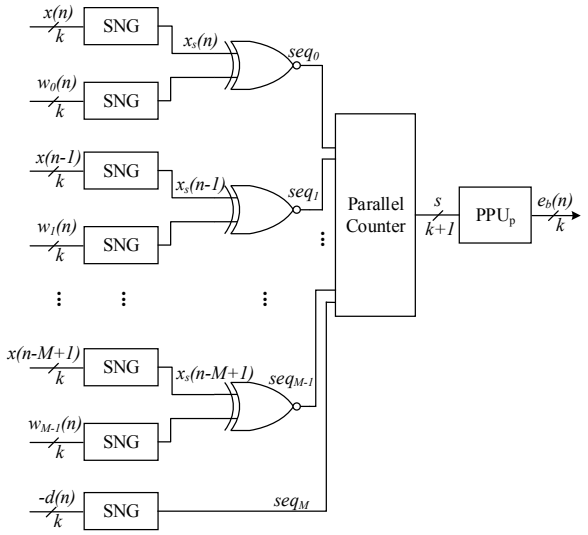


Fig. 5. Stochastic implementation of the linear filter. The post processing unit (PPU_p) converts the output of the parallel counter to a binary number used for the SNG of the LMS.

cause an error in the addition result by the counter based adder (as discussed next).

However, the conventional stochastic adder implemented by a multiplexer is not very efficient due to its SNGs and the loss of resolution as discussed in section I. Although the SNGs (for the select signals of the multiplexers) can be shared among adders at the same stage in an adder tree, the stochastic sequences at different stages cannot be correlated with each other to ensure an accurate addition. Therefore, many SNGs are required for an adder tree. To address this problem, a new stochastic adder is proposed; it consists of a multiple-bit parallel counter and a post processing unit (Fig. 5). The post processing unit is a simple circuit that performs shifting and other simple logic functions.

Let the real number encoded in the stochastic sequence seq_i of the multiplication result in Fig. 5 be $x_i, i = 0, 1, \cdots, M$ (where $x_i = w_i(n) \cdot x(n-i), i = 0, 1, \cdots, M-1$ and $x_M = -d(n)$ in Fig. 5). As discussed, the bipolar stochastic representation of a real number $x_i \in [-1, 1]$ is given by

$$x_i = \frac{2s_i}{l} - 1, \tag{3}$$

where s_i represents the number of 1's in the stochastic sequence seq_i and l is the length of the stochastic sequence. To obtain the sum, the addition is performed on both sides of (3)

$$\sum_{i=0}^{M} x_i = \frac{2\sum_{i=0}^{M} s_i}{l} - (M+1). \tag{4}$$

Dividing $(M+1)$ on both sides leads to

$$\frac{\sum_{i=0}^{M} x_i}{M+1} = \frac{2\sum_{i=0}^{M} s_i}{(M+1)l} - 1. \tag{5}$$

As per (3), the right side of (5) is another bipolar stochastic representation of a real number with a sequence length of $(M+1)l$, and the real number in the left side is the mean value of $x_i, i = 0, 1, \cdots, M$. It indicates that the direct concatenation of $(M+1)$ stochastic sequences gives the mean of the real numbers with a total sequence length of $(M+1)l$; thus, the resolution is preserved.

As $y(n) = \sum_{i=0}^{M-1} w_i(n) \cdot x(n-i) = \sum_{i=0}^{M-1} x_i$, then

$$\sum_{i=0}^{M} x_i = \sum_{i=0}^{M-1} x_i + x_M = y(n) - d(n). \tag{6}$$

Let $s = \sum_{i=0}^{M} s_i$ be the number obtained from the output of the parallel counter in Fig. 5. The error signal $e(n)$ used for updating the weights of the adaptive filter is obtained as

$$e(n) = d(n) - y(n) = (M+1) - \frac{2s}{l}. \tag{7}$$

(7) shows that subtraction and shifting can be used to obtain the error output. To further simplify the hardware implementation, a feature of the adaptive filter is utilized. For the adaptive filter, all of its outputs and intermediate results are in $[-1, 1]$ as long as its input operators are in $[-1, 1]$. Therefore, $e(n)$ is in $[-1, 1]$, and thus $\frac{2s}{l}$ is in $[M, M+2]$. Assuming $l = 2^k$, $\frac{2s}{l}$ can be obtained by right shifting s for $k-1$ bits. Let the binary format of s be $b_{m-1} \cdots b_1 b_0$, then $\frac{2s}{l} = b_{m-1} \cdots b_k b_{k-1}.b_{k-2} \cdots b_0$, where m is the width of s and $m > k$. Let $\frac{2s}{l} = x_{int} + x_\delta$, where $x_{int} = b_{m-1} \cdots b_{k-1}$ and $x_\delta = 0.b_{k-2} \cdots b_0$ are the integral and fractional parts of $\frac{2s}{l}$, respectively; then $e(n)$ can be represented by

$$e(n) = \begin{cases} 1 - x_\delta, & x_{int} = M \\ -x_\delta, & x_{int} = M+1 \\ -1, & x_{int} = M+2 \end{cases}. \tag{8}$$

In (8), the value of x_δ is 0 when $x_{int} = M+2$ because $e(n)$ must be in $[-1, 1]$. To implement the LMS algorithm, $e(n)$ is required to be mapped into the unipolar representation by $e_u(n) = \frac{e(n)+1}{2}$, as given by

$$e_u(n) = \begin{cases} 1 - \frac{x_\delta}{2}, & x_{int} = M \\ \frac{1-x_\delta}{2}, & x_{int} = M+1 \\ 0, & x_{int} = M+2 \end{cases}. \tag{9}$$

For ease in a hardware implementation, the k-bit binary format of $e_u(n)$ is given by

TABLE I. The Karnaugh map for estimating the value of x_{int}

$b_k b_{k-1}$ \ $m_1 m_0$	00	01	11	10
00	M	X	M+1	M+2
01	M+1	M	M+2	X
11	X	M+2	M	M+1
00	M+2	M+1	X	M

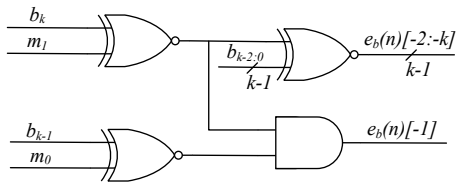


Fig. 6. The post processing unit of the parallel counter (denoted as PPU_p in Fig. 5).

$$e_b(n) = \begin{cases} 1.00\cdots0 - 0.0b_{k-2}\cdots b_0, & x_{int} = M \\ 0.10\cdots0 - 0.0b_{k-2}\cdots b_0, & x_{int} = M+1 \\ 0.00\cdots0, & x_{int} = M+2 \end{cases} \approx \begin{cases} 0.1\overline{b_{k-2}}\cdots\overline{b_0}, & x_{int} = M \\ 0.0\overline{b_{k-2}}\cdots\overline{b_0}, & x_{int} = M+1 \\ 0.0b_{k-2}\cdots b_0, & x_{int} = M+2 \end{cases}, \quad (10)$$

where the results for the first two cases are approximated by ignoring a bit of '1' at the least significant bit, while the result for the last case is accurate because $b_{k-2}\cdots b_0 = 00\cdots0$ when $x_{int} = M+2$.

As there are only three cases for x_{int}, the value of x_{int} can be obtained by comparing the two least significant bits of x_{int} and M. Table I shows the Karnaugh map for estimating the value of x_{int}, where m_1m_0 are the two least significant bits of M. The value of x_{int} is given by

$$d_M = \overline{b_k \oplus m_1} \wedge \overline{b_{k-1} \oplus m_0}, \quad (11)$$

$$d_{M+1} = \overline{b_k \oplus m_1} \wedge (b_{k-1} \oplus m_0), \quad (12)$$

$$d_{M+2} = (b_k \oplus m_1) \wedge \overline{b_{k-1} \oplus m_0}. \quad (13)$$

where d_M ='1' indicates $x_{int} = M$, d_{M+1}='1' indicates $x_{int} = M+1$, and d_{M+2}='1' indicates $x_{int} = M+2$. (10) shows that the first two cases can be combined because only their first fractional bits are different. Thus, $\overline{b_{k-1} \oplus m_0}$ is used to produce the first fractional bit in (10), and $\overline{b_k \oplus m_1}$ is used to determine the inverting operation in (10), as shown in Fig. 6. Hence, only $k+1$ bits of s are needed to obtain the error signal $e(n)$ and the width of the parallel counter can be set to $k+1$ because overflow does not affect the value of $e(n)$.

B. Implementation of the LMS algorithm

As per (2), two multiplications and one addition are required to update each weight of the adaptive filter. As the initial value of $w_i(M-1)$ is usually 0, the weight can be simplified to $w_i(n) = \mu \cdot \sum_{j=M-1}^{n-1} e(j)x(j-i)$ after n iterations. To further simplify the multiplication, the step size μ is set to 2^{-q} (q is an integer). Thus the multiplication by μ can be realized by a right shift operation. The remaining $\sum_{j=M-1}^{n-1} e(j)x(j-i)$ is implemented by a stochastic multiplier (XNOR) and counter based adder, as shown in Fig. 7. The stochastic sequences of the input vector $(x_s(n), \cdots, x_s(n-M+1))$ and the RNG of the error signal are shared with the linear filter, which significantly reduces the hardware of the stochastic multipliers. In this design, the counter is a serial counter and the post processing unit is similar to the one in Fig. 5 with some small differences that are explained next.

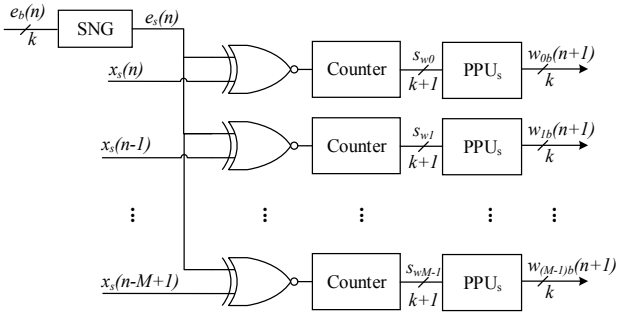


Fig. 7. Stochastic implementation of the LMS algorithm. The post processing unit (PPU_s) converts the output of the serial counter to a binary number used for the next iteration.

The post processing unit performs the function of

$$w_i(n) = \mu \cdot (\frac{2s_{wi}}{l} - (n - M + 1)), \quad (14)$$

where n is the iteration number of the adaptive filter and s_{wi} is the output obtained from the serial counter. Let the binary format of s_{wi} be $b_{i(m-1)}\cdots b_{i0}$, then $\frac{2s_{wi}}{l} = b_{i(m-1)}\cdots b_{ik}b_{i(k-1)}.b_{i(k-2)}\cdots b_{i0}$, where m is the width of s_{wi} and $m > k$. Let $\frac{2s_{wi}}{l} = w_{int} + w_\delta$, where $w_{int} = b_{i(m-1)}\cdots b_{i(k-1)}$ and $w_\delta = 0.b_{i(k-2)}\cdots b_{i0}$ are the integral and fractional parts of $\frac{2s_{wi}}{l}$, respectively; then $w_i(n)$ is given by

$$w_i(n) = \begin{cases} \mu \cdot (w_\delta - 1), & w_{int} = n - M \\ \mu \cdot w_\delta, & w_{int} = n - M + 1 \\ \mu, & w_{int} = n - M + 2 \end{cases}. \quad (15)$$

The unipolar representation of the real number $w_i(n)$ is given by

$$w_{iu}(n) = \begin{cases} \frac{\mu \cdot w_\delta + 1 - \mu}{2}, & w_{int} = n - M \\ \frac{\mu \cdot w_\delta + 1}{2}, & w_{int} = n - M + 1 \\ \frac{\mu + 1}{2}, & w_{int} = n - M + 2 \end{cases}. \quad (16)$$

Since $\mu = 2^{-q}$ (q is an integer), for an implementation in hardware the k-bit binary format of $w_{iu}(n)$ is given by

$$w_{ib}(n) = \begin{cases} 0.01\cdots1b_{i(k-2)}\cdots b_{iq}, & w_{int} = n - M \\ 0.10\cdots0b_{i(k-2)}\cdots b_{iq}, & w_{int} = n - M + 1 \\ 0.10\cdots01b_{i(k-2)}\cdots b_{iq}, & w_{int} = n - M + 2 \end{cases}, \quad (17)$$

where the more significant $(q+1)$ bits for each case are constant and the less significant $(k-q-1)$ bits are $b_{i(k-2)}\cdots b_{iq}$ for all cases. The constant bits are obtained as shown in Fig. 8. Likewise, the width of the serial counter is $k+1$ and the resolution is kept the same for the adder.

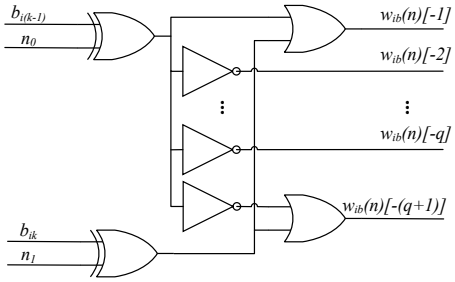


Fig. 8. The post processing unit of the serial counter (denoted as PPU$_s$ in Fig. 7). n_0 and n_1 are the two least significant bits of the iteration number n.

III. Simulation Results

The adaptive filter is employed to model an unknown system as an application of system identification [12]. The unknown system under consideration is a high-pass FIR filter with specifications shown in Table II. The tap of the filter is $M = 103$, and the step size for the adaptive algorithm is given by 2^{-10}. White Gaussian noise with a signal-to-noise ratio (SNR) of 40 dB is added to the output signals of the FIR filter system as interference noise.

The normalized misalignment is calculated to evaluate the convergence performance of the proposed adaptive filter. It is given by [13]

$$\eta(n) = \frac{E\{|\mathbf{h} - \mathbf{w}|^2\}}{E\{|\mathbf{h}^2|\}}, \tag{18}$$

where $\mathbf{h}$ is the impulse response of the unknown FIR system, and $\mathbf{w}$ is the adaptive weight vector. A threshold is set for the misalignment value to stop the simulation. For comparison purposes, the same stochastic design is implemented by using LFSR-based SNGs and the multiplexer-based adders.

In this simulation, the threshold value of the misalignment is -6 dB, and the maximum number of iterations is 17000. A sequence length of $l = 2^{12}$ bits is adopted for the proposed stochastic design (12-bit), which is compared to an LFSR-based implementation (20-bit) with a length of 2^{20} bits. The additional 8 bits for the LFSR-based design is for compensating the resolution loss caused by the multiplexer-based adder. The proposed stochastic design requires 16518 iterations to reach the misalignment threshold, while the LFSR-based stochastic design cannot reach the misalignment threshold before the number of iterations reaches the maximum value (17000). Fig. 9 shows the simulation results. The LFSR-based stochastic implementation shows an inferior performance compared to the proposed stochastic design at lower frequencies, while they have a similar performance at higher frequencies. Furthermore, the mean-squared error (MSE) and the misalignments of the two implementations are also computed to measure the average error and the speed of convergence. Fig. 10 shows that the LFSR-based implementation produces larger errors than the proposed stochastic design, while Fig. 11 shows that the proposed design converges faster than the LFSR-based implementation.

To evaluate hardware overhead, the two stochastic designs are implemented in VHDL and synthesized by the Synopsys design compiler in STM CMOS 28 nm technology. The supply voltage and temperature are 1.0 V and 25 °C, respectively. The area and timing results are reported by the Synopsys design compiler, while the power dissipation is estimated by the PrimeTime-PX tool using the value change dump file. The results are shown in Table III. For a better comparison, the values of the energy per operation (EPO) and throughput per

TABLE II. The specifications of simulated high-pass filter.

Specification	Value
Cut-off frequency f_c	800 Hz
Sampling frequency f_s	2000 Hz
Transition band bandwidth BW	20 Hz
Minimum stop-band attenuation	-40 dB
Maximum peak-to-peak pass-band ripple	3.937 dB

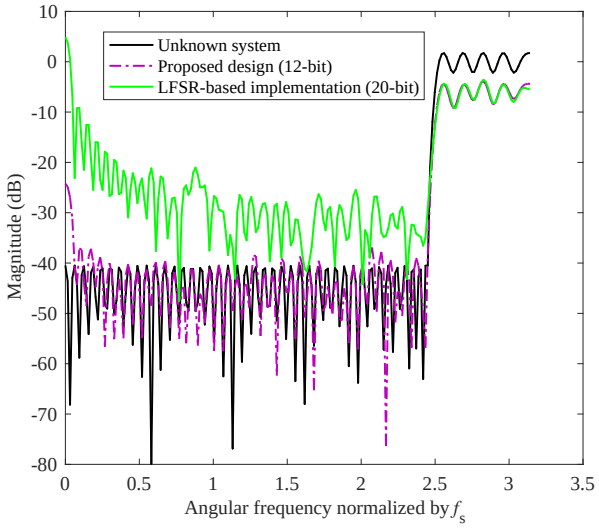


Fig. 9. The system identification results by different designs.

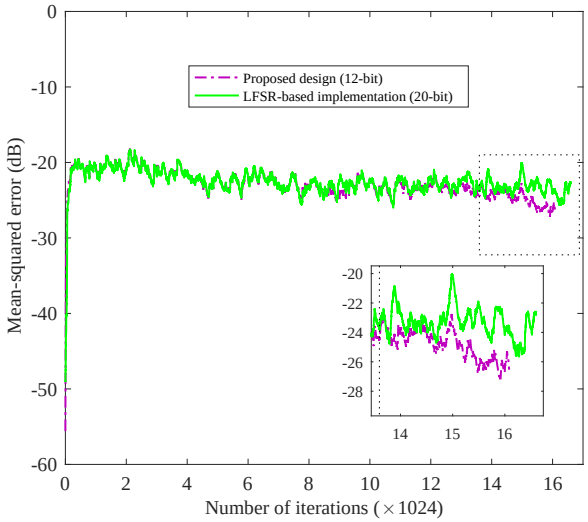


Fig. 10. The mean-squared error of the system identification application.

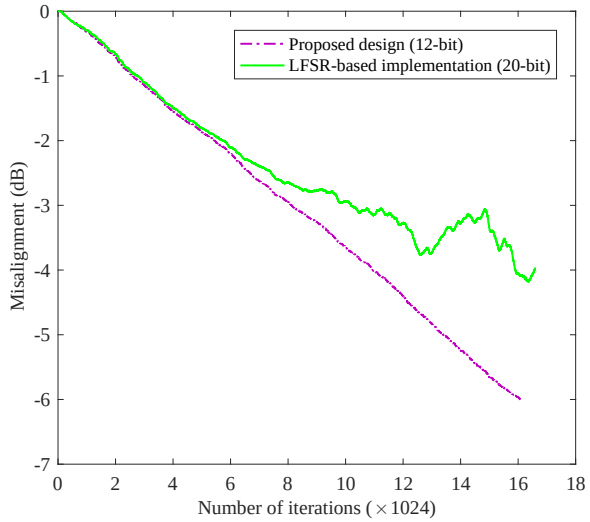


Fig. 11. The misalignment results of the system identification application.

TABLE III. The hardware characteristics of the adaptive filter designs.

Design	Clock period (ns)	Maximum frequency (MHz)	Area (μm^2)	Dynamic power (mW)	Leakage power (μW)	EPO $(nJ/operation)$	TPA $(operation/(s \cdot \mu m^2)$
proposed (12-bit)	3	1234.6	38,394	13.766	16.23	106.639	7.85
LFSR (20-bit)	3	877.2	48,811	21.456	20.43	34,033.946	0.02

area (TPA) [14] are computed for the considered designs. The EPO is defined as the energy consumed per binary operation during one clock period, and the TPA is defined as the number of binary operations per unit time and per unit area. They are given by

$$EPO = t_{op} \times Power, \tag{19}$$

and

$$TPA = 1/(t_{min} \times Area), \tag{20}$$

where t_{op} and t_{min} are the time required per operation (i.e., it is the clock period for the binary design, and Clock period $\times$ Sequence length for the stochastic design) and the minimum time required per operation, respectively. $Power$ is the total power consumption including the dynamic and leakage power. $Area$ is the utilized circuit area.

Table III shows that the proposed stochastic design has a larger maximum frequency by 40.74%. It requires 21.34% less area than the LFSR-based implementation and 35.81% and 20.56% less dynamic and leakage power, respectively. Moreover, the proposed design performs significantly better than the LFSR-based implementation in terms of EPO and TPA. Specifically, the LFSR-based implementation requires more than $300\times$ of energy and nearly $400\times$ of area per operation than the proposed design. These advantages are mainly due to the improved resolution of the proposed adder.

IV. Conclusion

This paper proposes the design of a low-power and area-efficient adaptive filter using stochastic computing circuits. A counter based stochastic adder with no SNG is utilized in the design to achieve a better computing accuracy because of its improved resolution compared to the conventional multiplexer-based adder. At a similar accuracy, the circuit area and power consumption of the proposed design are lower than a conventional LFSR-based stochastic implementation. The performance and energy consumption of the proposed design are also shown to be significantly better than those of the LFSR-based implementation.

References

[1] D. Comminiello, M. Scarpiniti, L. A. Azpicueta-Ruiz, J. Arenas-Garcia, and A. Uncini, "Functional link adaptive filters for nonlinear acoustic echo cancellation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1502–1512, 2013.

[2] Y. Xin, W. X. Li, Z. Zhang, R. C. Cheung, D. Song, and T. W. Berger, "An application specific instruction set processor (asip) for adaptive filters in neural prosthetics," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 5, pp. 1034–1047, 2015.

[3] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 449–462, 2014.

[4] B. R. Gaines, "Stochastic computing systems," in *Advances in Information Systems Science*, J. T. Tou, Ed. New York: Springer US, 1969, vol. 2, ch. 2, pp. 37–172.

[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 Transactions on Computers*, vol. 63, no. 6, pp. 1336–1350, 2014.

[6] I. L. Dalal, D. Stefan, and J. Harwayne-Gidansky, "Low discrepancy sequences for monte carlo simulations on reconfigurable platforms," in *2008 ASAP International Conference on Application-Specific Systems, Architectures and Processors*, 2008, pp. 108–113.

[7] A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *Proceedings of the conference on Design, Automation & Test in Europe*, 2014, p. 76.

[8] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive fir filter based on distributed arithmetic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 6, pp. 346–350, 2013.

[9] E. H. Krishna, M. Raghuram, K. V. Madhav, and K. A. Reddy, "Acoustic echo cancellation using a computationally efficient transform domain lms adaptive filter," in *2010 10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA)*, 2010, pp. 409–412.

[10] N. V. Thakor and Y.-S. Zhu, "Applications of adaptive filtering to ECG analysis: noise cancellation and arrhythmia detection," *IEEE Transactions on Biomedical Engineering*, vol. 38, no. 8, pp. 785–794, 1991.

[11] R. Wang, J. Han, B. Cockburn, and D. Elliott, "Design, evaluation and fault-tolerance analysis of stochastic fir filters," in *Microelectronics Reliability*, vol. 57, no. 2, 2016, pp. 111–127.

[12] B. Farhang-Boroujeny, *Adaptive filters: theory and applications*. John Wiley & Sons, 2013.

[13] M. Bekrani and A. W. Khong, "Convergence analysis of clipped input adaptive filters applied to system identification," in *2012 IEEE Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2012, pp. 801–805.

[14] R. Wang, J. Han, B. Cockburn, and D. Elliott, "Stochastic circuit design and performance evaluation of vector quantization for different error measures," *IEEE Transactions on VLSI Systems*, accepted for publication.