Optimizing Stochastic Computing-Based FIR Filters

Kuncai Zhong, Meng Yang, Weikang Qian
University of Michigan-Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University, Shanghai, China
Email: {kczhong, yangm.meng, qianwk}@sjtu.edu.cn

***Abstract*—Stochastic computing (SC) is an unconventional computing paradigm based on digital computation on stochastic bit streams. It is promising for many applications, one of which is the digital filter design. A previous work proposed an area-efficient SC-based finite impulse response (FIR) filter by optimizing the stochastic number generators (SNGs), which are used to produce stochastic bit streams of the desired probabilities. An SNG is composed of a random number source (RNS) and a probability conversion circuit (PCC). The previous technique is based on reducing the total number of RNSs within all the SNGs. In this work, we exploit two techniques that reduce the area of PCCs to further reduce the area of the SC-based FIR filters. They optimize the PCCs for variable and constant input probabilities, respectively. With the PCC area significantly reduced, it allows us to add few RNSs back to reduce the computation error due to correlation. Our experimental results showed that our proposed design can further improve both the area and computation accuracy of stochastic implementations of FIR filters.**

I. Introduction

Stochastic computing (SC), as an unconventional computing paradigm, was proposed in 1960s. It has attracted much attention in recent years [1]. It converts binary numbers into stochastic bit streams and operates on these streams with conventional digital circuits. A stochastic bit stream encodes a value equal to the probability of a 1 in the stream. For example, the bit streams 11010111 and 11001010 encode the values $6/8$ and $4/8$, respectively. By ANDing these two streams bitwise, we obtain an output bit stream 11000010, which encodes the value $3/8$. This example shows one attractive feature of SC: we can perform multiplication by a simple AND gate. Indeed, SC enables realizing many other complex arithmetic functions by very simple circuits. Besides, SC has several other advantages such as strong tolerance to bit flip errors and allowing multiple precisions with the same circuit.

Due to its low area cost, SC has been applied to several applications, such as digital filter [2], [3], image processing [4], decoding of modern error-correction code [5], and machine learning [6]. In this work, we focus on SC-based *finite impulse response (FIR)* filter.

In order to generate a stochastic bit stream from a binary number, *stochastic number generator (SNG)* is used. It consists of a *random number source (RNS)* and a *probability conversion circuit (PCC)* [7]. An RNS can be viewed as a set of k unbiased random bits, where k determines the precision of the generated probability. A typical RNS is a *linear feedback shift register (LFSR)*. A PCC takes k unbiased random bits provided by an RNS and k *target bits* $c_{k-1}, c_{k-2}, \ldots, c_0$ as inputs. It produces an output bit stream of probability $C/2^k$, where C equals the binary number $(c_{k-1}c_{k-2}\ldots c_0)_2$. A typical PCC is a comparator.

Chang and Parhi proposed a general SC-based architecture to implement FIR filters [2]. However, Ichihara *et al.* found that the SNGs occupy most of the area of that design [3]. To reduce the area, they proposed an RNS sharing technique, which reduces the total number of RNSs in the filter to 1. However, as showed in their experimental results, now PCCs occupy most of the area. In this work, we propose two techniques to further reduce the area of the PCCs. They optimize the PCCs for variable and constant input probabilities, respectively. With the PCC area significantly reduced, it allows us to add few RNSs back to reduce the computation error due to correlation. Our experimental results showed that our proposed design can further improve both the area and computation accuracy of stochastic implementations of FIR filters.

The rest of the paper is organized as follows. Section II introduces the related works on SC-based FIR filters. Section III presents the proposed new SC filter architecture using two novel techniques to reduce the PCC area. Section IV shows the experimental results. Section V concludes the paper.

II. Background and Related Work

In this section, we first introduce correlation in SC. Then, we introduce a general architecture of SC-based FIR filters. Finally, we present a previously proposed improved architecture with a single RNS.

A. Correlation in Stochastic Computing

Since stochastic bit streams are generated by SNGs, if the RNSs in the SNGs are correlated, the produced bit streams will also be correlated. Such correlation may result in a large error in the computing process. For example, if the same bit stream 11001100 is fed into the two inputs of an AND gate, the output will be 11001100 as well. In this case, since the two input bit streams to the AND gate are maximally correlated, the output value is not the product of the two input values. Thus, mitigating the correlation is critical for SC. The detailed analysis of correlation in SC is introduced in [8].

B. A General Architecture of SC-Based FIR Filter

An nth-order FIR filter performs the following computation

$$y = \sum_{i=0}^{n} h_i x_i,$$

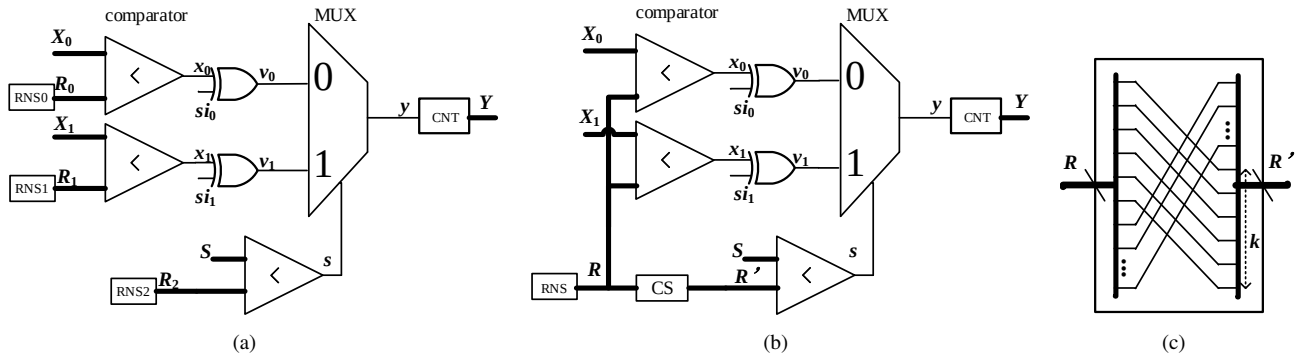

Figure 1: SC-based 1st-order FIR filter: (a) a general architecture proposed in [2]; (b) an improved architecture with a single RNS proposed in [3]; (c) a detailed illustration of the circular shift (CS) module used in Fig. 1(b) [3].

where y is the filter output of the current clock cycle, h_i $(0 \leq i \leq n)$ is the ith filter coefficient, and x_i $(0 \leq i \leq n)$ is the input signal i clock cycles before. Traditional binary implementation of an FIR filter uses binary multipliers and adders, which are costly in area and power.

Chang and Parhi proposed a general SC-based architecture to implement FIR filters [2]. It consists of RNSs, PCCs, XOR gates, MUXs, and a counter (CNT). Fig. 1(a) shows the architecture of a 1st-order FIR filter with function as $y = h_0x_0 + h_1x_1$. LFSRs are used as the RNSs and comparators are used as the PCCs. The final counter is used to convert a stochastic bit stream back to a binary number.

The input bit streams x_0 and x_1, the intermediate bit streams v_0 and v_1, and the output bit stream y are in the *bipolar format*. In such a format, a value p in the range $[-1, 1]$ is represented by a stochastic bit stream with probability $(1 + p)/2$ to be a 1. The signals si_0 and si_1 denote the sign of h_0 and h_1, respectively. That is, for $j = 0, 1$, si_j is 0 if $h_j \geq 0$ and 1 otherwise. Thus, if h_j $(j = 0, 1)$ is negative, the output of the jth XOR gate is the negation of its input and in biploar format, we have $v_j = -x_j$. Thus, for $i = 0, 1$, we can represent v_i as

$$v_i = x_i \cdot \text{sgn}(h_i), \tag{1}$$

where $\text{sgn}(x)$ is defined as follows

$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}.$$

The MUX in the circuit realizes a weighted sum. Its output value is $y = (1-s)v_0+sv_1$, where s is the probability encoded by the bit stream for the select input of the MUX. In the design, the probability of s is fixed as $\frac{|h_1|}{|h_0|+|h_1|}$. Thus, the output $y = \frac{|h_0|v_0+|h_1|v_1}{|h_0|+|h_1|}$. By Eq. (1), the final output is

$$y = \frac{h_0x_0 + h_1x_1}{|h_0| + |h_1|}.$$

Thus, the design shown in Fig. 1(a) realizes a scaled version of the original filter function. For a general nth-order FIR filter, the stochastic implementation is similar to the one shown in Fig. 1(a). The only difference is that the single MUX in the figure is replaced by a MUX tree.

C. An Improved Architecture of SC-Based FIR Filter with a Single RNS

In the design shown in Fig. 1(a), three RNSs are used. However, the data inputs of a MUX can be correlated, since at each clock cycle, the bit of only one data input is selected as the output. Based on this observation, Ichihara *et al.* proposed an improved SC-based FIR architecture with a single RNS [3]. We call it *single-RNS architecture* for short. For the same 1st-order FIR filter in Fig. 1(a), the single-RNS architecture is shown in Fig. 1(b). Notice that the bit streams v_0 and v_1 are generated from the same RNS. However, the bit streams for the data inputs of the MUX should be independent to that for the select input to avoid the correlation-induced error. Thus, ideally, the bit stream for the select input and those for the data inputs should be generated by two different RNSs. To further reduce the number of RNSs to 1, the authors proposed to do a *circular shift* (as shown in Fig 1(c)) on the outputs of the RNS for the bit stream for the data inputs and use that shifted version as the RNS for the bit stream for the select input. This reduces the error due to correlation.

For a general nth-order filter, the final output bit stream is produced by a MUX tree. Since the structure of the MUX tree is not unique and different MUX trees could lead to different correlation-induced error, an algorithm was proposed in [3] to design the optimal MUX tree with a low correlation-induced error. Table I shows the area comparison between the general architecture and the single-RNS architecture for a 31st-order FIR filter [3]. As it shows, the number of RNSs is reduced from 63 to 1 and the area is reduced from $4,837$ to $1,506$.

TABLE I: Area breakdown of the general architecture and the single-RNS architecture of a 31st-order FIR filter with 8-bit precision [3].

		general arch.			*single-RNS arch.*		
	area(μm^2)	#	area(μm^2)	%	#	area(μm^2)	%
RNS	53.73	63	3,385	70	1	53.73	3.6
PCC	20.22	63	1,274	26.4	63	1,274	84.6
XOR	1.596	32	51.07	1.1	32	51.07	3.4
MUX	1.862	31	57.72	1.2	31	57.72	3.9
CNT	69.69	1	69.69	1.5	1	69.69	4.7
Total			4,837	100		1,506	100

III. The Proposed Architecture for SC-based FIR

As we discussed in Section II-C, the single-RNS architecture reduces the area of the RNSs. However, the other part of the circuit is not changed. Specially, the area of the PCCs is not changed and it becomes the dominating part of the total area of the SC-based FIR filter. For example, as shown in Table I, it takes 84.6% of the total area for a 31st-order FIR filter. In order to further reduce the area of the filter, we introduce a new architecture in this section. As shown in Fig. 1(b), those input stochastic bit streams produced by the SNGs can be divided into two groups. The first group consists of those bit streams encoding the variable input signal x_i's. The second group consists of those bit streams fed into the select inputs of the MUXs. For a fixed filter, the probabilities of these bit streams are constant. Our proposed architecture involves two novel strategies. The first strategy optimizes the PCCs for those bit streams of variable probabilities, while the second strategy optimizes the PCCs for those bit streams of constant probabilities. We will present these two strategies in Sections III-A and III-B, respectively. Then, we will present the proposed architecture in Section III-C.

A. Optimizing PCCs for Variable Probabilities

We use a technique proposed in our previous work [9] to optimize the PCCs for generating bit streams of variable probabilities. It is based on a type of PCC called *weighted binary generator (WBG)* [10].

Fig. 2(a) shows a WBG with precision $k = 4$. It takes as inputs 4 unbiased random bits $r_3, \ldots, r_0$ of probability 0.5 to be a 1 and 4 target bits $c_3, \ldots, c_0$. The set of AND gates in the first level is referred to as *WBG Part 1* and the remaining part of the WBG is referred to as *WBG Part 2* [9]. WBG Part 1 produces signals $w_3, \ldots, w_0$, where w_i $(0 \leq i \leq 3)$ has the probability of $1/2^{4-i}$ to be a 1. WBG Part 2 takes $w_3, \ldots w_0$ and $c_3, \ldots, c_0$ as inputs and produces a stochastic bit stream with the probability equal to

$$P(w_3)c_3 + P(w_2)c_2 + P(w_1)c_1 + P(w_0)c_0 = \frac{(c_3c_2c_1c_0)_2}{2^4},$$

which is the desired probability.

To reduce the area of PCCs, we propose to replace the comparators in Fig. 1(b) for generating the variable probabilities x_i's by the WBGs. If WBGs are used as PCCs, normally, each input bit stream requires an individual WBG. Thus, the area is the sum of the areas of all WBGs including their Part 1's and Part 2's. However, given the architecture shown in Fig. 1(b) for the FIR application, if we replace the comparators for those x_i bit streams by the WBGs, the Part 1's of these WBGs are identical and hence, only one copy of WBG Part 1 is needed. Thus, the total area of the PCCs for those x_i bit streams reduces to the area of 1 WBG Part 1 plus $(n + 1)$ WBG Part 2's for an nth-order FIR filter. This mechanism of sharing WBG Part 1 is illustrated in Fig. 2(b) using an example of a 3rd-order FIR. Note that this mechanism of sharing WBG approaches the theoretical cost limit for SNGs [9]. Generally, it can be applied to reduce the area of PCCs for stochastic bit streams of variable probabilities, if these stochastic bit streams could be correlated. The stochastic implementation of FIR filter shown in Fig. 1(b) satisfies this condition.

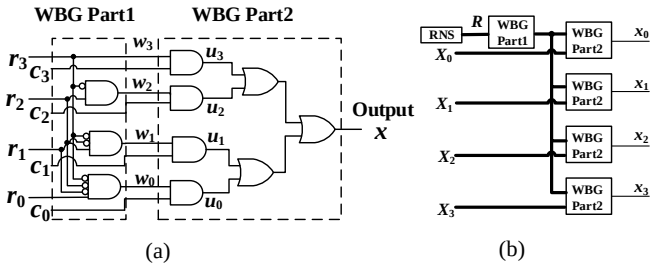

Figure 2: Weight binary generator (WBG) and its sharing mechanism: (a) a WBG for generating one stochastic bit stream [9]; (b) the share of WBG Part 1 for generating 4 stochastic bit streams that can be correlated.

B. Optimizing PCCs for Constant Probabilities

We use a technique proposed in our previous work [11] to optimize the PCCs for generating stochastic bit streams of constant probabilities, which are the streams to the select inputs of the MUXs in the architecture shown in Fig. 1(b). The technique proposed in [11] specifically optimizes the PCCs for generating constant-probability stochastic bit streams that can be correlated. In our case, the set of stochastic bit streams to the select inputs of the MUXs at the same level[1] in the MUX tree satisfies these conditions. First, they are constant probabilities for a fixed filter. Second, for any level in a MUX tree, only the output of one MUX in that level will be selected as the final output of the tree. Thus, the stochastic bit streams to the select inputs of the MUXs at the same level can be correlated without any influence to the final computation result.

Now, we briefly introduce the technique proposed in [11]. One idea is that since the probability of each target bit stream is constant, thus, we do not need to use a high-cost PCC for those variable probabilities, such as a comparator and a WBG. In [11], we first consider building a mincost combinational circuit that generates a single target probability from unbiased random bits. We use *AND-inverter graphs (AIGs)* [12] to represent combinational circuits. An AIG is a directed acyclic graph of 2-input AND gates and inverters. We measure the cost of an AIG by the number of AND gates in the graph. We showed that to generate a target probability $\frac{C}{2^k}$, where C is an odd number, from k unbiased random bits, the mincost AIG is a tree of $(k - 1)$ AND gates. We refer to such a tree as a *mincost AND-inverter tree (MAIT)*. Fig. 3(a) shows an example of a MAIT for generating the constant probability $1/16$ from 4 unbiased random bits. The MAIT contains 3 AND gates.

If we want to generate multiple constant-probability stochastic bit streams that can be correlated, the MAITs for all individual probabilities can share some common gates to reduce the total cost. For example, if we want to generate three

[1]The level is counted from the leaves to the root.

stochastic bit streams of probabilities $1/16$, $3/16$, and $5/16$ that can be correlated, we can merge the individual MAITs and build a final 3-output AIG as shown in Fig. 3(b). For this example, by merging the common part, the number of AND gates reduces from 9 to 6. The MAIT for a target probability is not unqiue and different choices of the MAIT for each individual target probability may lead to different amount of cost reduction. In our work [11], we proposed a *pairwise MAIT-merging algorithm* to synthesize a *low-cost AIG (LAIG)* to generate multiple constant-probability stochastic bit streams that can be correlated. We applied this technique to generate the stochastic bit streams for the select inputs of the MUXs at the same level.

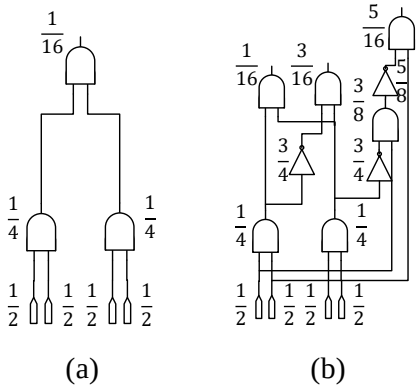

Figure 3: Mincost AND-inverter tree (MAIT) and low-cost AND-inverter graph (LAIG): (a) a MAIT for a single target probability; (b) a LAIG for generating 3 target probabilities that can be correlated.

C. The Proposed Architecture

In this section, we present a new architecture of SC-based FIR filter using the two techniques mentioned above. The architecture for a 4th-order FIR filter is illustrated in Fig. 4. It is based on the single-RNS architecture introduced in Section II-C. The stochastic bit streams x_i's are generated by the technique proposed in Section III-A. For the stochastic bit streams to the select inputs of the MUXs, they are generated by the technique proposed in Section III-B. In the example shown in Fig. 4, the MUX tree has three levels. The first level has two MUXs and their select bit streams can be correlated. Thus, they are generated by a 2-output LAIG. The second level has a single MUX and its select bit stream is generated by a single MAIT. The same for the third level.

In order to reduce the error due to correlation, we use two more RNSs than the single-RNS architecture. Thus, we use three different RNSs in total. The first is used to generate the input stochastic bit streams x_i's. The second is used to generate the select bit streams of the MUXs in the MUX tree except the last one. The third is used to generate the select bit stream of the last MUX in the MUX tree. Furthermore, we apply circular shift of the second RNS's outputs to each level in the MUX tree except the first and the last levels, as shown in the figure. For a general nth-order FIR filter, the proposed architecture contains 3 RNSs, 1 WBG Part 1, $(n+1)$ WBG Part 2's, $(n+1)$ XORs, n MUXs, $\lceil \log_2 (n+1) \rceil - 2$ circular shift blocks, $\lceil \log_2 (n+1) \rceil$ LAIGs or MAITs, and 1 counter.

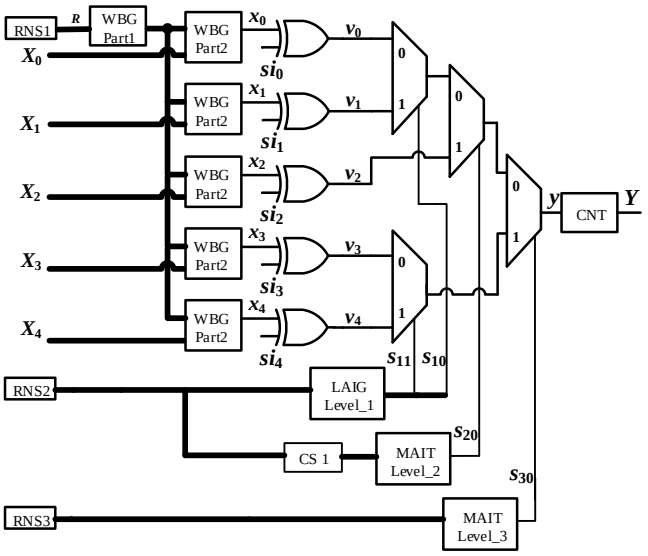

Figure 4: The proposed architecture of SC-based FIR filter using the two proposed techniques to reduce the area of the PCCs. The example is a 4th-order FIR filter. Each MAIT level generates a single output and the LAIG level generates more than 1 output.

IV. Experimental Results

In this section, we compared the proposed architecture with the single-RNS architecture presented in Section II-C. All the circuits were synthesized by Synopsys Design Compiler [13] and placed and routed by Cadence SoC Encounter [14]. The Nangate 45nm library was used [15]. The precision of the input probabilities x_i's and the select input probabilities are both chosen as 8-bit and the stochastic bit stream length is 256. The RNSs are realized by LFSRs. In the proposed architecture, three LFSRs have three different initial values to ensure the independence among them. The algorithm proposed in [3] was used to construct the optimal MUX tree with a low correlation-induced error. The kth circular shift module, CS k, has the shift amount as $(3k \mod 8)$. For example, CS 1 in Fig. 4 has its shift amount as 3.

TABLE II: The computation error and area for the single-RNS architecture and the proposed architecture for FIR filters with different orders.

	single-RNS arch.		*proposed arch.*	
# orders	error($\times 10^{-2}$)	area(μm^2)	error($\times 10^{-2}$)	area(μm^2)
4	4.52	423	1.69	359
8	3.48	668	2.11	437
16	4.92	1,129	3.40	575
32	5.36	2,028	3.02	843
average	4.57	1,062	2.56	554

Table II and Fig. 5 show the comparison on computation error and area between the single-RNS architecture and our proposed architecture for FIR filters of various orders. From

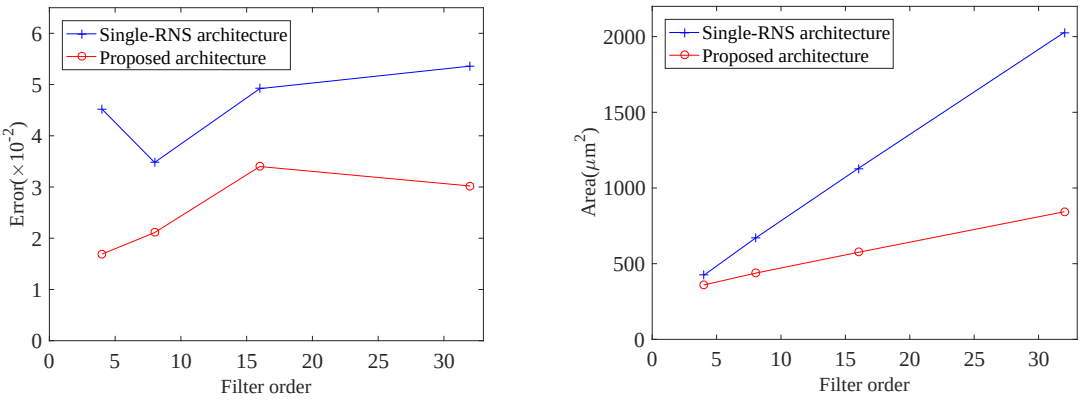

Figure 5: Comparison between the single-RNS architecture and the proposed architecture: (a) the computation error comparison; (b) the area comparison.

the table and Fig. 5(a), we can see that the proposed architecture has smaller error than the single-RNS architecture. Also, it has smaller area, as the table and Fig. 5(b) show. For a 32nd-order FIR filter, the proposed architecture saves the area by 58%.

In summary, the proposed architecture is better than the single-RNS architecture in both computation error and area. Although the proposed architecture uses 2 more RNSs than the single-RNS architecture, it significantly reduces the area of the PCCs. Thus, its entire area reduces. Furthermore, with the 2 extra RNSs, it can reduce the error due to correlation compared to the single-RNS architecture.

V. Conclusion

In this paper, we proposed a new architecture for stochastic computing-based FIR filter. It is an improved version over a state-of-the-art stochastic architecture for FIR filter. Specifically, we introduced two techniques to reduce the area of the probability conversion circuits (PCCs) used in the filter. The first technique shares the common part of the weighted binary generator to reduce the area of the PCCs for the variable probabilities, while the second technique synthesizes low-cost PCCs for the constant probabilities that can be correlated. With the PCC area significantly reduced, it allows us to add few RNSs back. This leads to a final design with both computation error and area reduced.

Acknowledgment

This work is supported by National Natural Science Foundation of China (NSFC) under grant no. 61472243 and 61204042.

References

[1] B. R. Gaines, "Stochastic computing," in *AFIPS Spring Joint Computer Conference*, 1967, pp. 149–156.

[2] Y.-N. Chang and K. K. Parhi, "Architecture for digital filters using stochastic computing," in *International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2697–2701.

[3] H. Ichihara, T. Sugino *et al.*, "Compact and accurate digital filters based on stochastic computing," *IEEE Transactions on Emerging Topics in Computer*, 2016.

[4] P. Li, D. J. Lilja *et al.*, "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.

[5] X.-R. Lee, C.-L. Chen *et al.*, "A 7.92 Gb/s 437.2 mW stochastic LDPC decoder chip for IEEE 802.15.3c applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 2, pp. 507–516, 2015.

[6] K. Kim, J. Kim *et al.*, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," in *Design Automation Conference*, 2016, pp. 124:1–124:6.

[7] M. Yang, J. P. Hayes *et al.*, "Design of accurate stochastic number generators with noisy emerging devices for stochastic computing," in *International Conference on Computer-Aided Design*, 2017, pp. 638–644.

[8] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *International Conference on Computer Design*, 2013, pp. 39–46.

[9] M. Yang, B. Li *et al.*, "Towards theoretical cost limit of stochastic number generator for stochastic computing," in *IEEE Computer Society Annual Symposium on VLSI*, 2018.

[10] P. K. Gupta and R. Kumaresan, "Binary multiplication with PN sequences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 603–606, 1988.

[11] Y. Ding, Y. Wu, and W. Qian, "Generating multiple correlated probabilities for MUX-based stochastic computing architecture," in *International Conference on Computer-Aided Design*, 2014, pp. 519–526.

[12] A. Mishchenko, S. Chatterjee, and R. Brayton, "DAG-aware AIG rewriting: a fresh look at combinational logic synthesis," in *Design Automation Conference*, 2006, pp. 532–535.

[13] Synopsys Inc., http://www.synopsys.com.

[14] Candence Inc., http://www.candence.com.

[15] Nangate Inc., http://www.nangate.com.