ARCHITECTURES FOR DIGITAL FILTERS USING STOCHASTIC COMPUTING

Yun-Nan Chang

National Sun Yat-Sen University
Department of Computer Science
and Engineering
Kaohsiung, Taiwan

Keshab K. Parhi

University of Minnesota
Department of Electrical and
Computer Engineering
Minneapolis, MN, USA

ABSTRACT

Stochastic computing has recently gained attention due to its fault-tolerance property. In stochastic computing, numbers are represented by probabilities of sequences. This paper addresses implementation of inner products and digital filters using stochastic logic. Straightforward implementations of stochastic inner products and digital filters lead to significantly large output error. To overcome this, this paper proposes a *novel scaling method* for efficient stochastic logic implementations of inner products and digital filters. By incorporating the filter coefficients into the probability of the selection signals of the multiplexors, the proposed weighted summation circuit can achieve better signal scaling with lower cost than the one derived from a traditional structure. This paper also presents how to vary the seeds in stochastic filters in order to reduce the correlation. Implementing IIR filters using stochastic logic limits possible pole locations. To overcome this, a new stochastic IIR filter structure is presented that includes a binary multiplier and stochastic-to-binary and binary-to-stochastic converters. Our experimental results show that the proposed architecture for the inner-product unit can lead to more than 12 times reduction in the error-to-power ratio. The stochastic FIR filters can perform the desired filtering function, but their accuracy degrades with the increase of filter order. The direct-form stochastic IIR filters may fail for large filter orders, but their performance can be improved by using cascade-form filter architecture.

Index Terms— Stochastic logic, stochastic filter, stochastic FIR filter, stochastic IIR filter, scaling.

1. INTRODUCTION

Stochastic computing (SC), first proposed several decades ago [1], has recently regained a lot of attention mainly due to its fault-tolerance capability which is a key requirement for the deep sub-micron technology. Different from the ordinary binary computing, this unconventional approach represents numbers using the probability value of bit-streams. Very low-cost circuits that are highly resistant to manufacturing process variations and soft errors can thus be designed by stochastic logic. However, the stochastic circuits may also suffer from long processing latency and accuracy degradation. Therefore, in the past, SC applications have been limited to the fields of neural networks [2] and control machines [3]. In recent years, several researches illustrate that SC can also be applied successfully in certain image processing [4], [5] and error-control coding [6] applications. The success of these new applications is due to the fact that they demand some special functions, which are very complex to implement using binary logic, but can be approximated efficiently by stochastic logic. In order to explore more SC opportunities, this paper addresses the SC implementations of one broad field of applications: digital filters.

The remainder of this paper is organized as follows. Since most digital filters are based on the inner-product function, Section 2 first describes how to realize this function based on general SC components, followed by our proposed modified SC architecture which can improve the performance. Section 3 discusses the design of stochastic FIR filters. Implementation of stochastic IIR filters is addressed in Section 4. Finally, some conclusions are given in Section 5.

2. DESIGN OF SC INNER-PRODUCT UNIT

Inner-products are key components of FIR and IIR digital filters. The design of the inner-product unit is quite straightforward as it contains several multiplication and addition operations. Fig. 1 shows some of the most fundamental building blocks used in SC circuits, which include the XNOR gate as the SC multiplier, the multiplexor as the scaled adder, and the linear-feedback-shift-register (LFSR) and a comparator as the stochastic number generator (SNG). Therefore, an inner-product module can be built by using these units. Fig. 2(a) shows the circuit diagram of an inner-product module for input vectors (x_0, x_1) and (a_0, a_1) based on the components shown in Fig. 1. We assume that all the binary inputs are converted into stochastic sequences.

Due to the nature of SC, the number represented by the probability of a sequence cannot be greater than one. Therefore, the SC adder shown in Fig. 1 has an implicit scale fac-

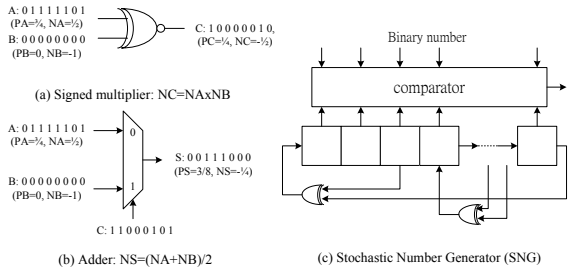


Fig. 1. The basic SC arithmetic units.

tor of $\frac{1}{2}$ such that the sum of the inputs will be automatically scaled down by two. The output of the inner-product module shown in Fig. 2(a) will equal $\frac{1}{2}(a_0x_0 + a_1x_1)$ whose magnitude is less than $max(|x_0|, |x1|)$. For the summation of more stochastic numbers, it can be expected the sum will become even smaller due to the implicit scaling effect. For small stochastic numbers, their variance may increase due to the imprecision of SC caused by insufficient sequence length or signal correlation. Therefore, to enhance the scaling effect, this paper proposes an alternative SC logic implementation of inner-product when the vector (a_0, a_1) is a constant.

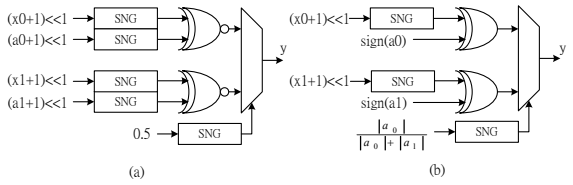


Fig. 2. The circuit diagram of inner-product module for input vectors of size two built by (a) general SC units and (b) proposed weighted-summation architecture.

Fig. 2(b) shows our proposed inner-product circuit based on the uneven weighted multiplexors. Here the probability of the selection signal of the multiplexor is no longer fixed at 0.5. If a_0 and a_1 are constants, this control signal probability will be set to $\frac{|a_0|}{|a_0|+|a_1|}$ whose value depends on the relative magnitude ratio of the coefficients. Since the constants a_0 and a_1 can be negative, the input sequences which represent the other vector may have to be inverted. The output of our proposed inner-product circuit shown in Fig. 2(b) equals $\frac{|a_0|}{|a_0|+|a_1|}sign(a_0)x_0 + (1 - \frac{|a_0|}{|a_0|+|a_1|})sign(a_1)x_1$ which can be rewritten as $\frac{1}{|a_0|+|a_1|}(a_0x_0 + a_1x_1)$. Compared with the summation result of Fig. 2(a), the proposed circuit scales the result better according to the magnitude of the constants. When $|a_0| + |a_1|$ is smaller than one, it will even scale-up the result. The scaling is similar to the use of Horner's rule in bit-serial implementations [7].

In addition to better signal scaling, the other advantage of the proposed design is the reduced number of SNGs required. Furthermore, the input sequence will pass through fewer levels of gates leading to less signal variation due to the SC operations. Table 1 compares the accuracy of results for different sizes of input vectors using the metric of error-to-signal power ratio. This ratio is derived from dividing the average output error power by the average output signal power. The error ratio is compared with the floating-point implementation results. For each vector size, 1000 different vector pairs are tested. The input vectors are generated randomly. The sequence length for each number is 1024. All the seeds used in SNG are different in order to reduce the correlation of stochastic sequences. As shown in this table, our proposed circuit has significantly less error than the other built from the basic SC components.

Table 1. Error comparison of two inner product circuits.

Vector Size	2	4	6	8
Proposed Design	0.0033	0.0040	0.0034	0.0052
Conventional Design	0.1783	0.0934	0.1206	0.0656

3. FINITE IMPULSE RESPONSE FILTER DESIGN

A general M+1-tap finite-impulse response (FIR) filter is represented by $y[n] = b_0x[n] + b_1x[n-1] + \cdots + b_Mx[n-M]$ which can be realized based on the proposed inner-product module design. The only exception of the filter to the inner-product module is that one of the input vectors corresponds to the filter coefficients while the other input vector is obtained through the delay line. Here we assume the filter input is in the binary form although it may be in the bit-stream form if it comes directly from a sigma-delta AD converter. Fig. 3 shows the circuit diagram for the N-tap filter whose output is the scaled filtering result $y[n]/\sum_{i=0}^{M}|b_i|$. The scaling of the FIR output will not alter the relative frequency contents of the signal.

There are two alternative approaches to generate the delayed version of the input signals. Fig. 3(a) first converts the input into the stochastic bit-stream, which then will pass through the delay line. In Fig. 3(b), the input signal first passes through the delay line, and then each of the signal from the delay line is converted separately to the stochastic bit sequence. Although it seems that the latter one will require more SNG modules, it should be noted that the total memory elements used to implement the delay line of these two approaches are different. In Fig. 3(a), each delay tap requires L-bit memory elements where L represents the length of stochastic sequence used to represent a signal sample. In Fig. 3(b) each delay requires W-bit memory elements where W represents the binary word-length of the input signal.

In addition to the tradeoff between the delay cost and the number of SNGs required as shown in Fig. 3, the implemen-

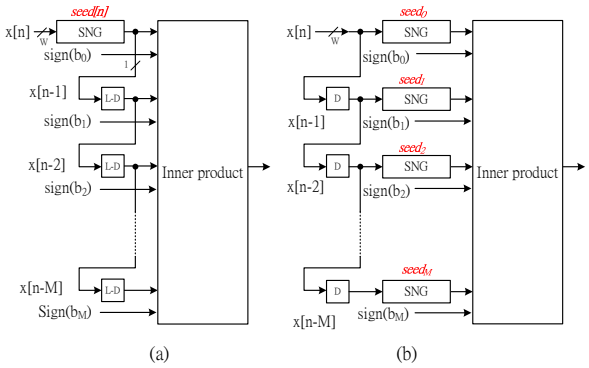


Fig. 3. The circuit diagram of the FIR filter for (a) delaying the stochastic sequence and (b) delaying the binary input sequence.

tation of the filter should also take into account how the correlation of different stochastic sequences in the circuit can be reduced. In Fig. 3(b), each SNG used to convert each delayed input signal should adopt a different seed for its initial LFSR value to generate the random sequence. The principle is the same for Fig. 3(a) but it is realized by changing the seeds for different incoming signals.

Several simulations were performed to test the accuracy of the stochastic FIR filters. An input test signal consisting of a mixture of five sinusoidal waves of different frequencies and random noise is used. Table 2 shows the error-to-signal power ratio for low-pass and high-pass filters with four different cut-off frequency settings and three different filter orders. In our simulation, the length of the stochastic sequence is 1024. A total of 256 input samples are used for simulation. It can be found in general the error will increase with the filter order because the data correlation is more likely to increase for larger circuit size. The result of the last column of Table 2 is for the filter where we do not vary the seed for the input signal. Fig. 4 further shows the spectrum of input and output signals obtained from stochastic and ideal filters for a high-pass and a low-pass filter. The frequency responses of these two filters are also shown in this figure. It can be found that the spectrum of the stochastic filter is very close to that of the ideal filter.

4. INFINITE IMPULSE RESPONSE FILTER DESIGN

The other category of digital filters is the infinite-impulse response (IIR) filter. Different from FIR filters, IIR filters contain recursive loops that impose strict constraints on its implementation. The transfer function of an N^{th}-order IIR can be represented in Z-domain as $H(z) = \frac{Y(z)}{X(z)} = \frac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots b_Mz^M}{1-a_1z^{-1}-a_2z^{-2}-\cdots-a_Nz^N}$. In order to implement the IIR filter, its time-domain representation is preferred, where output $y[n]$ equals $(b_0x[n] + b_1x[n-1] + \cdots + b_Mx[n-M]) + (a_1y[n-1] + a_2y[n-2] + \cdots + a_Ny[n-N])$. This equation

Table 2. Accuracy test results of stochastic FIR filters of different orders. (* use the constant seed)

Filter Order	Low-Pass Cut-Off Frequency (π) 0.2	0.4	0.6	0.8	0.8*
2	0.0037	0.0025	0.0013	0.0004	0.0032
4	0.0597	0.0465	0.0314	0.0145	0.0291
6	0.0648	0.0462	0.0637	0.0626	0.0943
Filter Order	**High-Pass Cut-Off Frequency (π) 0.2**	**0.4**	**0.6**	**0.8**	**0.8***
2	0.0008	0.0015	0.0023	0.0028	0.1742
4	0.0097	0.0127	0.0137	0.0161	0.0803
6	0.0491	0.0316	0.0114	0.0137	0.0609

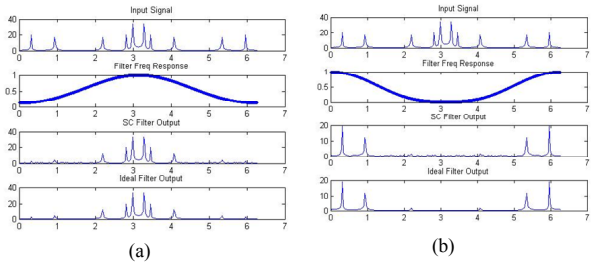


Fig. 4. The filtering results of (a) a high-pass 4^{th}-order FIR filter with cutoff-frequency 0.6 π, and (b) a low-pass 6^{th}-order FIR filter with cutoff-frequency 0.4 π.

can be divided into two parts. This first part involves the input signals $x[.]$, while the other one involves the previous output signals $y[.]$. Both parts can be realized by the inner-product module as shown in Fig. 5. The summation of each part will be scaled by a factor. The terms $scaleA$ and $scaleB$, which equal $1/\sum_{i=1}^{N-1}|a_i|$ and $1/\sum_{i=0}^{M}|b_i|$, respectively, represent the scaling factors of the corresponding inner-product modules. The summation results of both parts are added together by another multiplexor with the selection signal whose probability is set to $\frac{scaleB}{scaleA+scaleB}$.

The design flow of IIR so far is similar to FIR. However, it should be noted that in FIR, the relative frequency content is not changed by scaling the output by a constant factor. However, in IIR design, the filter output $y[n]$ will be fed-back for further computation of the following outputs. If the output is scaled, it will affect the result of the next output sample, and eventually alter the filter function. If we take into account the scale factors, the filter transfer function will now be changed to $H'(z) = \frac{\frac{1}{scaleB}(b_0+b_1z^{-1}+b_2z^{-2}+\cdots b_Mz^M)}{1-\frac{1}{scaleA}(a_1z^{-1}+a_2z^{-2}+\cdots a_Nz^N)}$. The zeros of this new transfer function are the same as the original, but the poles are not the same. Therefore, the final summation result shown in the middle of Fig. 5 has to be multiplied by a scale factor. Since the scale factor can be greater than one while the multiplication of factor larger than one cannot be imple-

mented very accurately using stochastic logic, the summation sequence has to be converted back to a binary number first by stochastic-number-to-binary-number converter (S2BC). This binary number will then be multiplied by the scale-factor, and the result will be converted into stochastic sequence again. It should be noted that it is possible that the result of this binary multiplication can be greater than one, which cannot be represented by any stochastic sequence. Therefore, the product of the binary multiplier has to be clamped between ± 1.

Based on the proposed IIR architecture, Table 3 shows the error-to-signal power ratio for some IIR examples. It can be found that even for small filter orders, the stochastic IIR filters perform much worse than FIR filters. The reason is mainly due to error accumulation in the recursive IIR filters. In addition, the error resulting due to SC will be magnified by the binary multiplier. The error will increase significantly in the filter order as illustrated in Table 3. For the case of a 4^{th} order low-pass IIR filter with cut-off frequency 0.2π, the error itself is larger than the desired signal; therefore, this stochastic filter fails to function properly. Table 3 also lists the factor $scaleA$, the loop gain, in different IIR simulation cases. For those IIR filters which exhibit large errors tend to have large $scaleA$. Therefore, the direct implementation of a stochastic IIR filter for an order larger than four is very likely to fail. However, the filter can be implemented in several ways. Fig. 5 represents a so-called direct-form implementation of IIR filters. However, the filter can also be implemented as the cascaded form by decomposing a high-order filter transfer function into the product of several first and second-order sections. For example, Fig. 6(a) shows the experimental results of the direct-form stochastic IIR architecture for the transfer function given by: $\frac{0.0007378(1+6z^{-1}+15z^{-2}+20z^{-3}+15z^{-4}+6z^{-5}+6z^{-6})}{1+3.183z^{-1}-4.622z^{-2}+3.769z^{-3}-1.791z^{-4}+0.4593z^{-5}-0.0453z^{-6}}$. The output signal is all wrong. However, if we decompose this transfer function as follows: $\frac{0.12(1+2z^{-1}+z^{-2})}{1-1.268^{z}-1+0.7051z^{-2}} \frac{0.12(1+2z^{-1}+z^{-2})}{1-1.01^{z}-1+0.358z^{-2}} \frac{0.12(1+2z^{-1}+z^{-2})}{1-0.904^{z}-1+0.215z^{-2}}$, we can implement the same transfer function as the cascade of three second-order IIR filters. As shown in Fig. 6(b), the result is significantly more accurate than the direct-form.

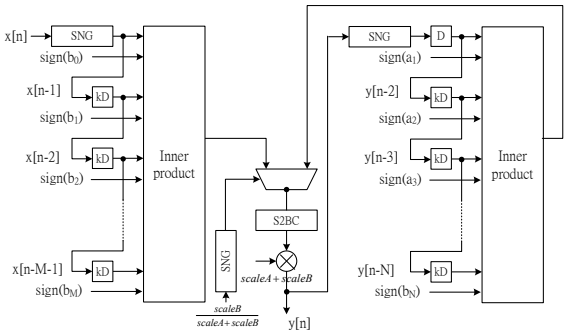
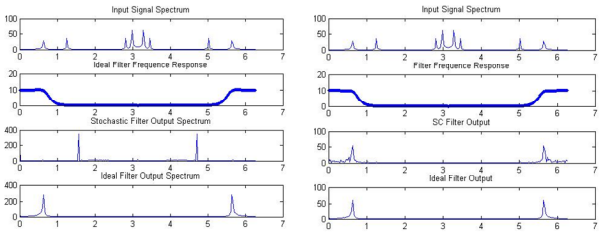


Fig. 5. The circuit diagram of the stochastic IIR filter.

Table 3. Accuracy test results of stochastic IIR filters of different orders.

Filter Order	Low-Pass Cut-Off Frequency (π) 0.2	0.4	0.6	0.8
2	0.3161	0.0364	0.0613	0.3822
scaleA	1.55	0.56	0.56	1.55
4	15.4	0.1978	0.55	8.78
scaleA	5.92	1.674	1.674	5.92
Filter Order	High-Pass Cut-Off Frequency (π) 0.2	0.4	0.6	0.8
2	0.1058	0.0213	0.012	0.0187
scaleA	1.5558	0.5653	0.5683	1.558
4	3.3355	0.1211	0.0489	1.4175
scaleA	5.9255	1.6749	1.6749	5.9225



(a) (b)

Fig. 6. The simulation result of a 6-order stochastic low-pass IIR filter based on (a) the direct-form implementation, and (b) a cascade-form implementation.

5. CONCLUSION

This paper addresses several design issues of stochastic filters. First of all, a weighted multiplexor based summation-tree architecture is proposed which can achieve better signal scaling by controlling the multiplexor selection signal with probability equal to the corresponding coefficient ratio. Next, FIR filters can be built based on the core inner-product module and additional storage elements. Different seeds are used to convert the input samples to stochastic sequences with reduced correlation. The same approach can be extended to the design of IIR filters. Due to the recursive nature of IIR, an additional binary multiplier may be required to provide the signal gain of the recursive loop. Our experimental results show that higher-order IIR stochastic filters do not perform well if implemented by the direct-form. It is suggested that the cascade IIR form can be a much better alternative. According to our experimental results, the implementation of digital filters based on SC is feasible. However, how to further enhance the accuracy of stochastic IIR filters needs further investigation.

6. REFERENCES

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

[2] B. D. Brown and H. C. Card, "Stochastic neural computation I: Compputational elements," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, September 2001.

[3] A. Dinu, M. N. Cirstea, and M. Mccormick, "Stochastic implementation of motor controllers," in *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics*, 2002, pp. 639–644.

[4] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 93–105, January 2011.

[5] W. Qian, M. D. Riedel, H. Zhou, and J. Bruck, "Transforming probabilities with combinational logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 9, pp. 1279–1292, September 2011.

[6] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5617–5626, November 2011.

[7] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, Hoboken, NJ: Wiley, 1999.