Volume 5, Issue 7, July 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Implementation of FIR Filter Using Stochastic Computing

[1]Marri Jithendra, [2]Sounak Samanta, [3]Dr. B. Chandra Mohan
[1]Department of ECE, Bapatla Engineering College, Bapatla, India
[2]Scientist"D" DLRL DRDO, Hyderabad, India
[3]Department of ECE, Dean of Academics, Bapatla Engineering College, Bapatla, India

Abstract: - Stochastic computing was proposed several decays ago. In stochastic computing numbers are represented by its probability of sequences. Because it requires large amount memory stochastic computing loses its attention, but due to its false tolerance property stochastic computing regained its attention. This paper addresses the implementation of FIR filters and its inner products using stochastic logic. Implementation of inner products by using traditional structure leads to significantly large errors. To overcome this problem this paper proposes a novel scaling method for efficient stochastic logic implementation of inner products and fir filters. The proposed weighted summation circuit achieves better signal scaling with lower cost by incorporating the filter coefficients into the probability of the selection signals of the multiplexer than the one derived from traditional structure. Our experiments results, the stochastic FIR filters can perform the desired filtering function, but their accuracy degrades with the increase of filter order.

Key words: - Stochastic computing, False tolerance, Stochastic logic, Scaling, Stochastic FIR filter

I. INTRODUCTION

Stochastic computing was first proposed in 1960's it has recently regained significant attention due to its false tolerance capability which is key requirement for deep sub-micron technology. The original motivation for considering computation with stochastic was the simplicity of computational elements. Stochastic computation held out the possibility of carrying out complex computations with very simple hardware. Stochastic circuits may also suffer from long processing latency and degradation of accuracy. Therefore in the past stochastic computing applications were limited in the fields of neural networks and control machines.

Different from the ordinary binary computing, the unconventional approach represents numbers using probability of bit streams. Note that an n-bit binary number requires 2^n bits in a stochastic implementation, for same resolution. Stochastic numbers are based on unary representation where each bit in the number has the same weight. For example, in a number with 256 bits, if 75 bits are 1, then the value of the number is 75/256. If any bit is flipped, the error introduced due to flipping is 1/512, independent of the location of the flipped bit. But in a two's complement number, the error introduced due to flipping of one bit can vary from 1/2 to 1/256, depending on the position of the bit. Thus, the stochastic implementations are inherently fault-tolerant and are less affected by errors due to bit flipping.

In recent years SC applied successfully to certain image processing and error controlling applications. The success is due to the fact, that the applications demand some special features, which are very complex to implement by using binary computing, but can be approximates efficiently by stochastic computing. In order to explore more SC opportunities, this paper addresses the SC implementations of one board field of applications: FIR filters.

This paper organized as follows. How stochastic numbers are generated is discussed in section2. Since most digital filters are implemented based on the inner products, so how to realize this inner products by using SC components, our proposed modified SC architecture which can improves the performance are describes in section 3. Section 4 deals with design of stochastic FIR filter. Finally, some conclusions are given in section 4.

II. STOCHASTIC NUMBER GENERATOR (SNG UNIT)

The Randomizer Unit is shown in Fig. 1. To generate a stochastic bit stream, a random number generator produces a number R in each clock cycle. If R is strictly less than the number C stored in the corresponding constant number register, then the comparator generates a one; otherwise, it generates a zero.

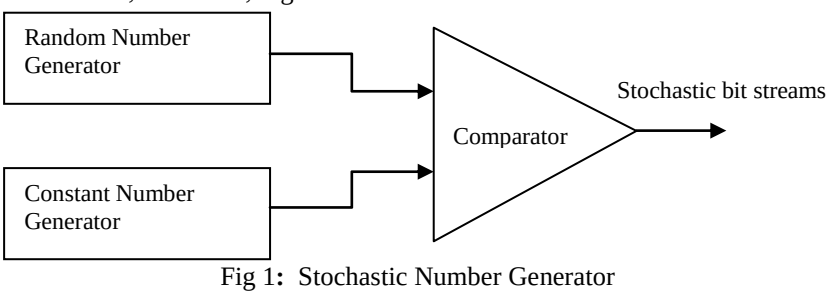


Fig 1: Stochastic Number Generator

In our implementation, we use linear feedback shift registers (LFSRs). Assume that the LFSR has L bits. Therefore it can generate pseudorandom numbers with a period of 2^{L-1}. We choose L so that $2^{L-1} \geq N$, where N is the length of the input random bit stream. The set of random numbers that can be generated by such an LFSR is $\{1, 2 \ldots 2^{L-1}\}$, and the probability that R equals the specific k in the set is:

$$P(R = K) = 1/2^{L} - 1$$

Given a constant integer $1 \leq C \leq 2^{L}$, the comparator generates a one with probability

$$P(R < C) = \sum_{k=1}^{C-1} P(R = k) = \frac{C-1}{2^{L}-1}$$

Thus, the set of probability values that can be generated by the randomizer unit is

$$S = \left\{0, 1, \frac{1}{2^{L}-1}, \ldots, 1\right\}$$

Given an arbitrary value 0<p<1 we can round it to the closest number p' in S. Hence, C is determined by p as

$$C = round\left(p\left(2^{L}-1\right)\right)+1$$

In our stochastic implementation, we require different input random bit streams to be independent. Therefore, LFSRs for generating different input random bit streams are configured to have different feedback functions

III. DESIGN OF INNER PRODUCT MODULE

Inner products are key requirement in the design of FIR filters. The design of the inner-product unit is quite straight forward as it contains several multiplication and addition operations. Fig. 2 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).

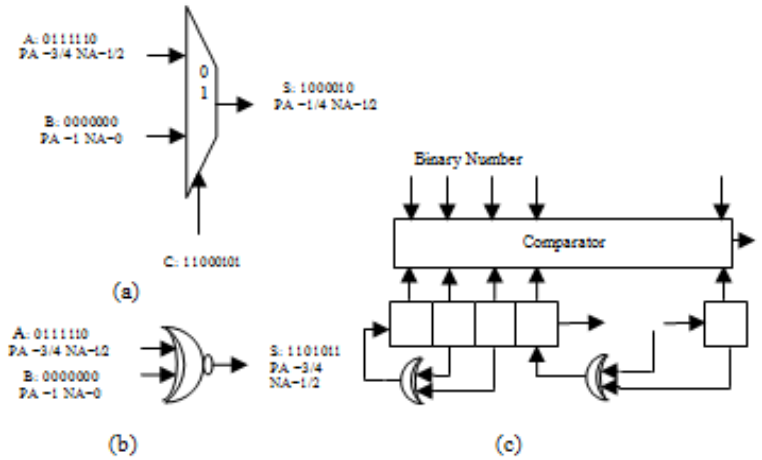


Fig 2: The basic SC arithmetic Units (a) Adder (b) Signed Multiplier (c) Stochastic Number Generator

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 factor of $1/2$ such that the sum of the inputs will be automatically scaled down by two. The output of the inner product module shown in Fig. 3(a) will equal $\frac{1}{2}(a_0 x_0 + a_1 x_1)$ whose magnitude is less than *max* ($|x_0|$; $|x_1|$). 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.

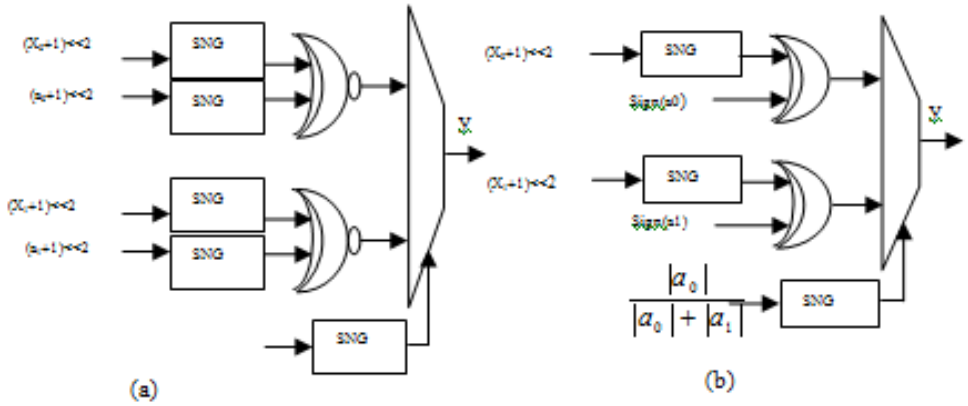


Fig 3: The circuit diagram of inner product module for input vectors of size two built by (a) General SC units and (b) Proposed weighted summation Architecture

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. 3(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 $a1$ 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 $\frac{|a_0|}{|a_0|+|a_1|} sign(a_0)x_0 + \left(1-\frac{|a_1|}{|a_0|+|a_1|}\right) sign(a_1)x_1$ which can be rewritten as $\frac{1}{|a_0|+|a_1|}(a_0x_0 + a_1x_1)$ Compared with the summation result of Fig. 3(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. In addition to better signal scaling, the other advantage of the proposed design is the reduced number of SNGs required.

IV. FIR FILTER DESIGN

A general M+1-tap finite-impulse response (FIR) filter is represented by $y[n] = b_0x[n] + b_1x[n-1] + \ldots + b_Mx[n-M]$ 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. 4 shows the circuit diagram for the N-tap filter whose output is the scaled filtering result $y[n] \Big/ \sum_{i=o}^{M} |b_i|$. 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. 4(a) first converts the input into the stochastic bit-stream, which then will pass through the delay line. In Fig. 4(b), the input signal first passes through the delay line, and then each of the signals 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. 4(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. 4(b) each delay requires *W*-bit memory elements where *W* represents the binary word-length of the input signal.

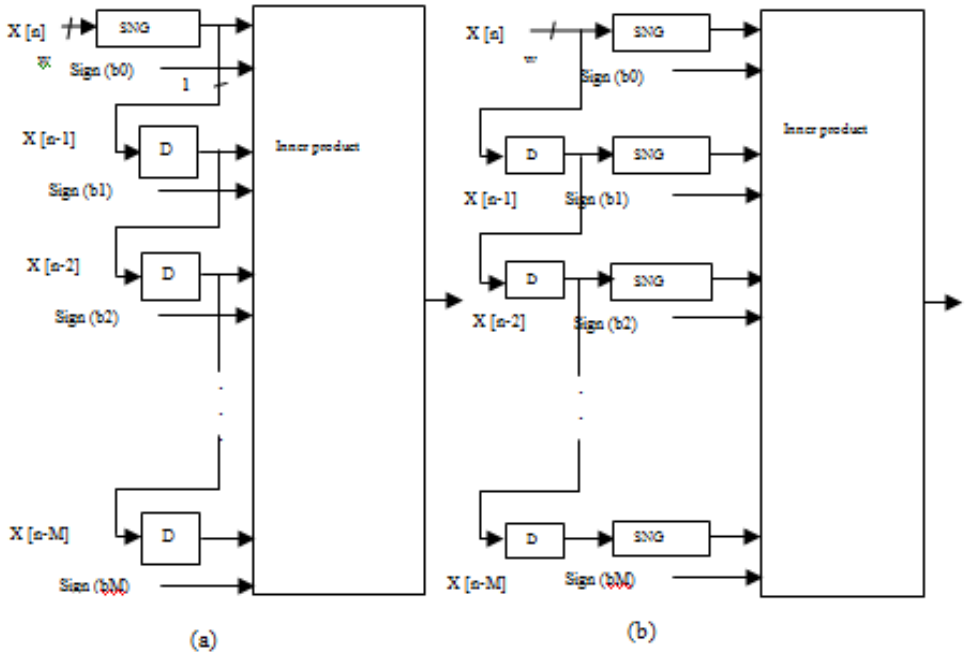
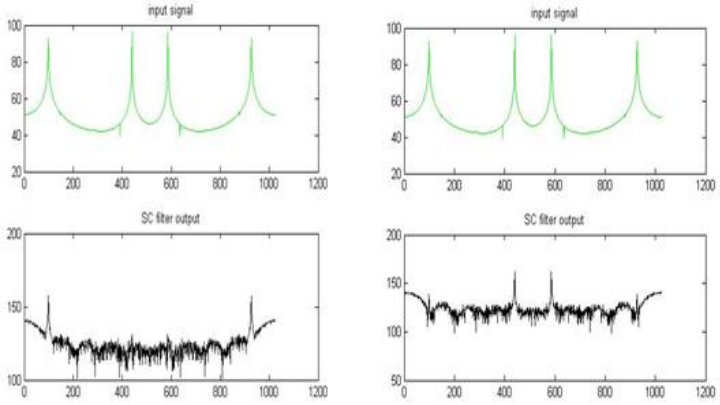


Fig 4: The circuit diagram of the FIR filter for (a) Delaying the stochastic sequence and (b) Delaying the binary input sequence

In addition to the trade-off between the delay cost and the number of SNGs required as shown in Fig. 4, the implementation of the filter should also take into account how the correlation of different stochastic sequences in the circuit can be reduced. In Fig. 4(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. 4(a) but it is realized by changing the seeds for different incoming signals.



(a) (b)

Fig 5: The filtering results (a) Low Pass 3rd order FIR filter (b) High Pass 3rd order FIR filter

V. CONCLUSION

In this paper the several design issues of stochastic logic are proposed. First of all, weighted multiplexer based summation architecture is proposed which can achieves better signal scaling by controlling the selection signals of the multiplexer with probability equal to the corresponding coefficient. Next, a modified inner product structure which can decreases the correlation effect between the probability sequences of coefficient. Finally FIR filters can be design based on the modified inner product module and additional storage elements. Different seeds are used to convert the input samples into stochastic sequences with reduced correlation. Further extension is design the IIR filters using the same approach.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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..

[5] 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.

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

[7] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, Hoboken, NJ: Wiley, 1999.2701.

