Journal of Universal Computer Science, vol. 23, no. 7 (2017), 589-602
submitted: 31/3/17, accepted: 30/6/17, appeared: 28/7/17 © *J.UCS*

Stochastic Computing with Spiking Neural P Systems

Ming Ming Wong
(School of Computer Science and Engineering
Hardware & Embedded Systems Lab (HESL)
Nanyang Technological University, Singapore
mmwong@ntu.edu.sg)

Mou Ling Dennis Wong
(Institute of Sensors, Signals and Systems
Heriot-Watt University Malaysia
Wilayah Persekutuan Putrajaya Malaysia
d.wong@hw.ac.uk)

Abstract: This paper presents a new computational framework to address the challenges in deeply scaled technologies by implementing stochastic computing (SC) using the Spiking Neural P (SN P) Systems. SC is well known for its high fault tolerance and its ability to compute complex mathematical operations using minimal amount of resources. However, one of the key issues for SC is data correlation. This computation can be abstracted and elegantly modeled by using SN P systems where the stochastic bit-stream can be generated through the neurons spiking. Furthermore, since SN P systems are not affected by data correlations, this effectively mitigate the accuracy issue in the ordinary SC circuitry. A new stochastic scaled addition realized using SN P systems is reported at the end of this paper. Though the work is still at the early stage of investigation, we believe this study will provide insights to future IC design development.

Key Words: Stochastic Computing, Membrane Computing, Spiking Neural P System, Fault Tolerance, Integrated Circuits
Category: I.1.2, I.2

1 Introduction

The rapid emergence of Internet of Things (IoT) has driven the integrated circuit (IC) technology to scale with greater depth. Vast variety of applications with complex computations are expected to be fabricated onto nano-devices with stringent hardware and power constraints. Such development trend gives rise to mounting concerns over variability, noises and uncertainties in IC manufacturing.

Subsequently, this imposes a great challenge in maintaining the paradigm of the deterministic computing in IC design. Deterministic computation are sensitives towards noises and variations and these could lead towards instability in ICs. As a direct result, the abstraction process of mapping the logical Boolean computations to the physical layer has become extremely complicated, expensive and unsustainable. In near future, it is foreseen that the conventional deterministic computing may no longer be practical and feasible in IC design.

These problems have later motivated fellow researchers and IC designers to deploy a different computation domain to mitigate the shortcomings of deterministic computation in handling noisy signals and computation uncertainties. Stochastic computing (SC) [Gaines 1967] that processes signals in a probabilistic approach has therefore appeared to be an potential alternative.

This computation framework which incorporates probability theory, is proven to be able to handle computation uncertainties in a more effective and efficient manner [Alaghi and Hayes 2013]. With that, SC has brought the emergence of an unconventional and non-deterministic computation with high fault tolerance, which is the key requirement for deep sub-micron technology [Qian et al. 2011, Moons and Verhelst 2014, Zhang et al. 2016]. Furthermore, SC is particularly attractive in IC design for it uses low complexity computation blocks. On the downside, the accuracy level of SC is subjected to the randomness of stochastic bit-streams i.e. the data has to be uncorrelated.

In this study, we intend to take this research to another level by incorporating the membrane computing, specifically the Spiking Neural P (SN P) systems with stochastic computation. The main objective here is to take the best out of both computation frameworks to present a new computational model that is; 1, small in resource consumption, 2, insusceptible towards noises, variations and uncertainties, and 3, high in computational accuracy.

SN P systems is a class of distributed and parallel computing models inspired by the neuro-biological behaviour of spiking neurons [Ionescu et al. 2006]. From scientific point of view, neurons resemble electrical devices whereby they send electrical pulses of identical voltage (termed as spikes) to the neighbouring neurons. Generally, SN P systems can be visualized as a system consisted of a set of neurons placed in the nodes of directed graph, where neurons send spikes i.e. signals along the arcs of the graph.

In fact, the idea of using neurons to generate logic and to signal each other is not new in circuit design. However, note that neurons' activities rely on voltage pulses. Hence, SN P systems is different from the level sensitive logic systems applied in ordinary deterministic computing. This characteristic is suitable for the stochastic computation as the neurons spiking can be used to generate stochastic bit-streams effectively. Besides, the spiking rules which govern the computation procedures in the system are not affected by the correlation between processing data. Therefore, the computation accuracy is guaranteed.

In a nutshell, SC that is low in complexity and high fault tolerance is modeled using SN P systems in order to improve the computation accuracy. Along with this concept, a new computational model, a stochastic scaled addition using SN P systems is presented in this study.

The remaining of this paper is organized as follows. Section 2 describes the background overview of both SC and SN P systems. Detailed mathematical explanations of SC is then elaborated in Section 3, while the in-depth descriptions of SN P systems is presented in Section 4. A new computational model, which is the stochastic scaled addition modeled using SN P systems is reported in Section 5. Finally, some discussion and future works are drawn in Section 6.

2 Background

Stochastic computing (SC) [Gaines 1967], which was introduced several decades ago has recently gained a fair amount of attentions in IC manufacturing. Unlike the deterministic computing, SC employs the principle of randomness whereby the value of numbers are conveyed through the statistical distribution of its logical value. In other words, the logical Boolean computation are transformed into stochastic computations framework, represented in probability values of interval $[0, 1]$.

Such characteristics enables SC to offer **high fault tolerance capability** and **low computation hardware area**, while maintaining the equivalent performance as the conventional deterministic computing. Overall, SC serves as an promising alternative in comparison to the conventional deterministic computing which are high in computational cost and also high inclination towards accuracy degradation in physical implementation.

Though SC has been known for decades, very few physical realization have been proposed. Initially, SC applications were limited to the field of neural networks [Brown and Card 2001] and machine controls [Dinu et al. 2002]. Until recent years, it was discovered that SC efficiently simplifies some mathematical functions which are computational expensive in binary computation. These functions can be efficiently approximated using stochastic logic with minimal hardware requirements and without significant accuracy degradation. Ever since, SC implementation has been extended to image processing [Alaghi et al. 2013, Li and Lilja 2011], error control coding applications [Naderi et al. 2011] and digital filter design [Chang and Parhi 2013,Parhi and Liu 2014,Liu and Parhi 2015,Saraf et al. 2014].

At the same time, SC comes with a few drawbacks as well. These are such as the variance inherent in estimating the value of a stochastic signal and the increased number of clock cycles required to accomplish a given computation [Brown and Card 2001]. In simple, the existing SC systems suffer from large computation latency and inaccuracy problems. However, the SC driven ICs are relatively small in areas. This in turn provides high opportunities for massive parallelism and this may actually alleviate the latency issue. In addition to that, retiming (pipelining) could be employed easily in order to maximize system clock rate. With that, this leaves the accuracy of SC as the major consideration issue in nano-scaled IC design.

In SC, the data are transformed and presented as stochastic sequences via the use of stochastic number generator (SNG). The accuracy of SC framework is highly dependent on the quality of the generated stochastic bit-streams, of which, the data has to be both **accurate** and **uncorrelated**. Therefore, it is concluded that the two main sources of inaccuracies in SC systems are: 1, initial **conversion error** introduced by SNG; and 2, computation error caused by **data correlations** of different bit-streams [Lifeng and Chakrabarti 2013].

The existing works on SC attempted to mitigate this problem by improving the design of SNG for better accuracy and highly uncorrelated randomized stochastic sequences generations [Lifeng and Chakrabarti 2013]. Meanwhile, many other related studies proposed optimizations in SC circuitry design in order to avoid correlations between the bit-streams throughout the computations as well as error propagations [Chang and Parhi 2013, Parhi and Liu 2014, Liu and Parhi 2015, Saraf et al. 2014].

In this study, we run against above-mentioned trend by incorporating bio-inspired computing as a solution to improve the computational accuracy in SC. Bio-inspired computing, short for biologically inspired computing, is a major candidate in natural computation, whose aim is to abstract computing ideas from biological systems to construct high-performance computing models and algorithms. The abstract computing ideas include data structures, information encoding/decoding strategy, operations with data, ways to control operations, computing intelligence, and many more.

Membrane computing, initiated by Gh. Păun, is a new branch of bio-inspired computing, which seeks to discover new computational models from the study of biological cells, particularly of the cellular membranes [Păun 2000, Păun et al. 2010]. The obtained models are distributed and parallel bio-inspired computing devices, usually called P systems.

In 2006, spiking neural P systems, namely **SN P systems**, were proposed through modeling the way neurons communicate via electrical impulses (spikes) [Ionescu et al. 2006]. Such systems are also known as a specific group of ingredients in membrane computing [Păun 2002], and corresponding to a shift from cell-like to neural-like architectures.

In recent years, SN P systems became the popular subject of investigation for developing powerful neural-like computing models. In terms of motivation of models, SN P systems fall into the third generation of neural network models [Maass 1997]. SN P systems are capable of generating and accepting the sets of Turing computable natural numbers [Ionescu et al. 2006], generating the recursively enumerable languages [Chen et al. 2007] and computing the sets of Turing computable functions [Păun and Păun 2007].

With different biological and mathematical inspirations, lots of variants of SN P systems have been proposed, such as SN P systems with anti-spikes [Pan and Păun 2009, Song et al. 2013], asynchronous SN P systems [Cavaliere et al. 2009], asynchronous SN P systems with local synchronization [Song et al. 2012], homogeneous SN P systems [Zeng et al. 2009, Song et al. 2009], sequential SN P systems [Ibarra et al. 2009], SN P systems with rules on synapses [Song et al. 2014, Song and Pan 2015]. The latest studies in this area are such as SN P systems with self-organizations [Wang et al. 2016], SN P systems with request rules [Song and Pan 2016] and SN P systems with white hole neurons [Song et al. 2016].

For applications, SN P systems are used to design logic gates, logic circuitries [Song and Pan 2016] and operating systems [Adl et al. 2010], perform basic arithmetic operations [Zeng et al. 2012], solve combinatorial optimization problems [Zhang et al. 2014], diagnose fault of electric power systems [Wang et al. 2010]. SN P systems with neuron budding and division and space-time tradeoff strategy can theoretically solve computationally hard problems in a feasible (polynomial or linear) time [Ishdorj et al. 2010, Leporati et al. 2009, Pan et al. 2009].

3 Stochastic Computation (SC)

The basic rule of SC is that the computational data (in bit-streams) are represented as stochastic sequences and are then processed in the form on digitized probabilities [Qian et al. 2011]. Naturally, the representations and all the involved computations always lie within the real-number interval $[0, 1]$. Stochastic representation can be coded in two formats: *SC-unipolar* and *SC-bipolar* [Gaines 1967].

As an example, a 2's complement binary input bit-stream $\{0011\}_2$ is represented in stochastic bit-streams S, consisted of 3 of bit '1' out of $2^4 = 16$ bits (remaining bits are zeros). This stochastic bit-streams S, is also interpreted as $p = P(S = 1) = 3/16$. Such coding is termed as SC-unipolar format. On the other hand, consider a 2's complement binary input bit-stream $\{1101\}_2$ represented in stochastic bit-streams S. By using SC-bipolar format, the deterministic value is mapped to $p = P(S = 1)/2 = 13/(16 \times 2) = 13/32$.

In other words, stochastic representation observes the probability of 1s at arbitrary bit position in S. Such representation serves as the main reason for having high fault tolerance in SC. A single bit-flip in a long bit-stream causes only a minor change in original logical value. On the contrary, a single bit-flip in the conventional 2's complement computation will result in huge error especially if the bit-flip occurs on higher-order bit.

A general SC architecture consists of three major components: the stochastic number generator (SNG), the stochastic computing elements (SCE) and the de-randomizer (as depicted in Figure 1). The SNG is used to convert the deterministic binary value into stochastic bit-streams by using the (pseudo) random number generator and a comparator [Lifeng and Chakrabarti 2013]. Meanwhile, the de-randomizer, which is usually a binary counter, is used to decode the output bit-stream back into deterministic binary value. The SCEs are the arithmetic operators such as the multiplier, adder or subtractor which are computed using SC.

Multiplication of two inputs streams, which is computational intensive in conventional signed binary computing, can be performed using single XNOR gate in SC. Assuming that the stochastic input bit-streams, X_1 and X_2 are suitably uncorrelated, the output for their multiplication, Y, is derived as,

$$\begin{aligned} y &= P(Y = 1) \\ &= P(X_1 = 1)P(X_2 = 1) \\ &\quad +(1 - P(X_1 = 1))(1 - P(X_2 = 1)) \end{aligned}$$

SC multiplication in bipolar format is clearly a logical XNOR operation between input bit-streams, X_1 and X_2 in digital circuit. For unipolar format, the multiplication is performed using a logical AND operation instead. SC multiplier for both unipolar and bipolar formats are as depicted in Figure 2.

Addition in SC is performed using a special operation, termed as scaled addition. The addition is scaled such that the value always lies between the probability interval $[0, 1]$. With S is a constant scale, the sum of two independent stochastic bit-streams X_1 and X_2, Y, is defined as,

$$\begin{aligned} y &= P(Y = 1) \\ &= P(S = 1)P(X_1 = 1) + (1 - P(S))(P(X_2 = 1)) \\ &= SX_1 + (1 - S)X_2 \end{aligned}$$

Thus, multiplexer with conditional select line S, set as $P(S) = \frac{1}{2}$ can be used to realize the scaled addition of two stochastic bit-streams in digital circuit. **Subtraction** in SC is similar to the adder except that the scaled substractor requires an additional inverter and such processing is only applicable in bipolar format. Both the SC scaled adder and scaled substractor are illustrated in Figure 3.

Generally, through SC, arithmetic operations can be efficiently computed in simple circuit using standard logic elements. Thus, realization of an SC system is in fact very low in hardware cost.

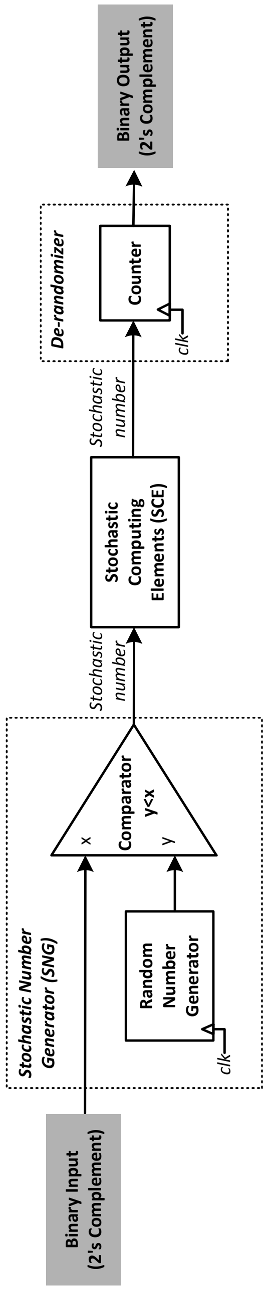

Figure 1: General Architecture of Stochastic Computing (SC) system.

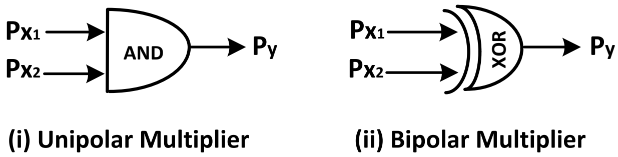

Figure 2: Stochastic Multiplier for (i) Unipolar and (ii) Bipolar rformats.

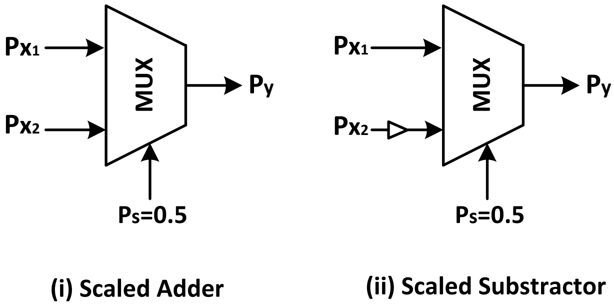

Figure 3: Stochastic Scaled Adder/Substractor

4 Spiking Neural P (SN P) Systems

An SN P system degree $m \geq 1$ is a construct of the form

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, in, out),$$

where

- $O = \{a\}$ is a singleton alphabet (a denotes spike);
- $\sigma_1, \sigma_2, \ldots, \sigma_m$ are neurons of the form $\sigma_i = (n_i, R_i)$ with $1 \leq i \leq m$, where
  (1) $n_i \geq 0$ is the number of spikes initially placed in σ_i;
  (2) R_i is a finite set of rules to process information of the following forms:
    - Spiking rule: $E/a^c \to a^p; d$, where E is a regular expression over $\{a\}$, $c \geq p \geq 1$ and $d \geq 0$;
    - Forgetting rule: $a^s \to \lambda$ with $s \geq 1$ and $a^s \notin L(E) \cup L(E')$ for any any spiking rule $E/a^c \to a^p; d$ and request rule $E'/a^q \leftarrow a^r$;
- $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ is the set of synapses between neurons, with restriction $(i, i) \notin syn$ (no self-loop synapse);
- $in, out \in \{1, 2, \ldots, m\}$ indicate the input and output neuron; where the input neuron can read spike trains from the environment, and the output neuron can emit spikes to the environment.

The spiking rule from any R_i of the form $E/a^c \to a^p; d$ with $c \geq p \geq 1$ is called an *extended spiking rule* or *extended rule*; if $p = 1$, the rule is called a *standard spiking rule* or *standard rule*. If $L(E) = \{a^c\}$, the rule can be simply written as $a^c \to a^p; d$; in addition, if $d = 0$, the rule can be simply written as $a^c \to a^p$.

Spiking rules are used as follows. If the neuron σ_i contains k spikes, and $a^k \in L(E), k \geq c$, then spiking rule $E/a^c \to a^p; d \in R_i$ can be applied. By using the rule, neuron σ_i fires with consuming c spikes ($k - c$ spikes remain in the neuron) and sends p spikes to its neighboring neurons (having synapses connections) after d time units (as usual in membrane computing, a global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized). If $d = 0$, then the p spikes are emitted immediately; if $d = 1$, then the p spikes will be emitted one time unit later, etc. In general, if the rule is used in step t and $d \geq 1$, then in steps $t, t+1, \ldots, t+d-1$ the neuron is in close status (this corresponds to the refractory period from neurobiology), so that it cannot receive new spikes (if a neuron has a synapse to a closed neuron and tries to send several spikes along it, then these particular spikes are lost). In the step $t+d$, the neuron emits the p spikes to each of its neighboring neuron and becomes again open, so that it can receive spikes and may fire again in step $t + d + 1$ with applying the rule. Because two spiking rules, $E_1/a^{c_1} \to a^{p_1}; d_1$ and $E_2/a^{c_2} \to a^{p_2}; d_2$, can have $L(E_1) \cap L(E_2) \neq \emptyset$, it is possible that two or more spiking rules can be used in a neuron at some moment, while only one of them is chosen non-deterministically to apply.

It is important to notice that the applicability of the spiking rules are controlled by checking the number of spikes contained in the neuron against a regular expression associated with the rule. The regular expressions associated with the rules can be considered as the circumstance for the application of the rules.

The rule of the form $a^s \to \lambda$ is called a forgetting rule, by which a predefined number of spikes will be removed out of the neuron (i.e., removed from the system). It has $a^s \notin L(E) \cup L(E')$ for any any spiking rule $E/a^c \to a^p; d$ and request rule $E'/a^q \leftarrow a^r$, which indicates when a forgetting rule can be used, no spiking and request rule is applicable.

The configuration of the system is described by both the number of spikes associated with each neuron and by the number of steps to wait until it becomes open (this number is zero if the neuron is already open). Using the rules as described above, we can define *transitions* among configurations. Any sequence of transitions starting from the initial configuration is called a *computation*. A computation, starting from reading spikes through input neurons, finally halts if it reaches a configuration where no rule in the neuron can be used. With any computation, halting or not, it is associated a spike train, that is, a binary sequence with occurrences of 1 indicating time instances when the output neuron emits a spike. The spike train can be taken as the computation result of the system.

5 A New Model: Stochastic Scaled Addition using SN P System

In this section, a new model of computing $x + y = \lceil \frac{x+y}{2} \rceil$ (stochastic scaled addition) by SN P systems is given, where x and y stochastic bit-streams (in binary form) with length n. The system is shown in Figure 4.

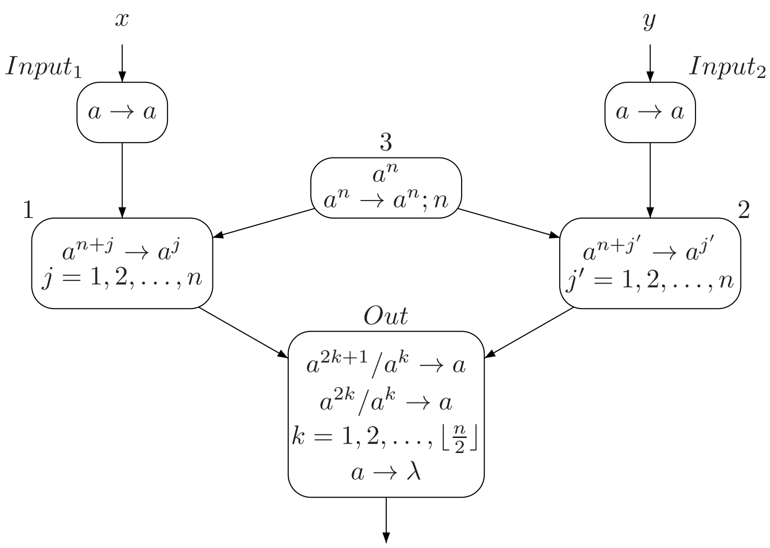

Figure 4: An SN system Π computing $x + y = \lceil \frac{x+y}{2} \rceil$

Initially, all the neurons have no spike inside, with the exception that neuron σ_3 holds n spikes. It fires by using spiking rule $a^n \to a^n; n$ and produces n spikes, but emits the n spikes to neurons σ_1 and σ_2 after n steps, i.e., at step n, due to the delay n.

Assume stochastic numbers x and y are of the forms $x = x_1 x_2 \ldots x_n$ and $y = y_1 y_2 \ldots y_n$ with $x_i, y_i \in \{0, 1\}$. The value represented by x is $\frac{\sum_i^n x_i}{n}$ and y is associated with value $\frac{\sum_i^n y_i}{n}$. The two spike trains can be read as spike trains through input neurons σ_{Input_1} and σ_{Input_2} accordingly. The process of reading spike train x through input neuron σ_{Input_1} is as follows, which is assumed starting at the very beginning. If $x_1 = 1$, neuron σ_{Input_1} receives one spike from the environment at step 1. With the spike, neuron σ_{Input_1} fires and sends one spike to neuron σ_1. If $x_1 = 0$, neuron σ_{Input} receives no spike from the environment at step 1. Similar cases happens at steps $2, 3, \ldots, n$. After n steps (all the bits have been read), neuron σ_1 has a number of spikes which is equivalent with the number of bits with value 1 in x, at most n (in case that every bit of x is 1). This happens also to neuron σ_2 with accumulate a certain number of spikes (the number of bits with value 1 in y), after input neuron σ_{Input_2} reading spike train y. With spikes less than n, neurons σ_1 and σ_2 keep inactive for no rule can be used.

At step $n+1$, neuron σ_3 emits the produced n spikes to each of neurons σ_1 and σ_2, such that the two neurons will fire in the next step with using spiking rules $a^{n+j} \to a^j$ and $a^{n+j'} \to a^{j'}$. Neuron σ_1 sends j spikes and neuron $\sigma_{j'}$ sends j' spikes to neuron σ_{out}. Note that, number j is exactly the number of bits with value 1 in x, and j' is the number of bits with value 1 in y. In this way, neuron σ_{out} accumulate $j + j'$ spikes inside, which can be odd or even number.

With spiking rules $a^{2k+1}/a^k \to a$ and $a^{2k}/a^k \to a$ with $k = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$ in neuron σ_{out}, it is not hard to find that neuron σ_{out} will emit $\lceil \frac{j+j'}{2} \rceil$ spikes. Hence, the spike train generated by the system Π contains $\lceil \frac{j+j'}{2} \rceil$ bits with value be 1, which is exactly the value of $\lceil \frac{x+y}{2} \rceil$.

6 Discussion and Future Works

Today's integration circuits are built using silicon technologies with deterministic computation which are sensitive towards noises and uncertainties often result in instability and unreliability in digital systems. The problem is further amplified as the semiconductor device fabrication node is getting smaller and smaller. This issues has led to the need of fault tolerant computational model.

In this work, a new computational model that performs non-deterministic addition through SC is constructed using SN P systems. Performing basic arithmetic operations using SN P systems is no longer new in the literature. However, to the best of our knowledge, this is the first work to construct stochastic scaled adder using SN P systems. This non-deterministic computation is able to compute arithmetic operations more efficiently compared to the conventional deterministic computing. Therefore, the resultant SN P system is observed to be relatively small and simple in its structure.

To be exact, the proposed stochastic scaled addition using SN P systems is consisted of 6 neurons where two specified neurons are use for inputting the addends and one neuron is used for outputting the obtained result. In other words, only 3 neurons are required to perform the computation.

In comparison to the deterministic adder modeled using SN P systems, such as reported in [Zeng et al. 2012, Tian et al. 2016], the system is composed of 10 neurons. Furthermore, in the same work, the subtractor system is different from the adder system [Zhang et al. 2016]. The proposed SN P subtractor system is more complex with 12 neurons in total. On the other hand, in SC, scaled adder and scaled substractor shared the same computation in bipolar format except that one of the inputs needs to be inverted (refer Figure 3). This implication is important in IC design as this enable resource sharing and hence promote effective hardware size reduction.

Overall, the aim of this work is to deploy SC which computes arithmetic operation probabilistically (results in high fault tolerance) on a bio-inspired system; the SN P systems. SN P spiking rules are applied and subjected to the neuron spiking activities only and thus the system accuracy will not be affected by the correlation or the randomness of the processing data. As an end result, a stochastic scaled addition which is smaller in its SN P system structure is realized and presented in this study.

The concept presented in this research is still at the early stage of investigation, where there are several issues and problems that ought to be put into considerations. A potential next research direction would be to model a stochastic multiplier in SN P systems. Both multipliers and adders are needed to construct the inner-product core, which is the main computational unit in FIR and IIR digital filter designs. Therefore, it would be interesting to study on the feasibility to model a digital filter using SN P systems.

Acknowledgments

The authors thank Tao Song for his valuable comments and suggestions to this work during his visiting fellowship at Swinburne University of Technology Sarawak Campus.

References

[Adl et al. 2010] Adl,A., Badr,A., and Farag, I. Towards a spiking neural P systems OS, arXiv preprint arXiv:1012.0326.

[Alaghi and Hayes 2013] Alaghi, A. and Hayes, J.-P. "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 92:1–92:19, May 2013.

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

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

[Cavaliere et al. 2009] Cavaliere, M., Ibarra, O. H., Păun, G., Egecioglu, O., Ionescu, M. and Woodworth, S. Asynchronous spiking neural P systems, Theoretical Computer Science 410 (24) (2009) 2352–2364.

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

[Chen et al. 2007] Chen, H., Freund, R., Ionescu, M., Păun, G. and Pérez-Jiménez, M. J. On string languages generated by spiking neural P systems, Fundamenta Informaticae 75 (1) (2007) 141–162.

[Dinu et al. 2002] Dinu, A., Cirstea, M.-N. and McCormick, M. "Stochastic implementation of motor controllers," in *Industrial Electronics, 2002. ISIE 2002. Proceedings of the 2002 IEEE International Symposium on*, 2002, vol. 2, pp. 639–644 vol.2.

[Gaines 1967] Gaines, B. R. "Stochastic computing," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, New York, NY, USA, 1967, AFIPS '67 (Spring), pp. 149–156, ACM.

[Ibarra et al. 2009] Ibarra, O. H., Păun, A. and Rodríguez-Patón, A. Sequential SNP systems based on min/max spike number, Theoretical Computer Science 410 (30) (2009) 2982–2991.

[Ionescu et al. 2006] Ionescu, M., Păun, G., and Yokomori, T. Spiking neural P systems, Fundamental Informaticae 71 (2) (2006) 279–308.

[Song and Pan 2016] Song, T., Pan, Z. Wong, D.M., Wang, X., Design of Logic Gates Using Spiking Neural P Systems with Homogeneous Neurons and Astrocytes-like Control, Information Sciences, 372, (2016) 380-C391

[Ishdorj et al. 2010] Ishdorj,T.-O., Leporati, A., Pan, L., Zeng, X. and Zhang, X. Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with precomputed resources, Theoretical Computer Science 411 (25) (2010) 2345–2358.

[Leporati et al. 2009] Leporati, A., Mauri, G., Zandron, C., Păun, G. and Pérez-Jiménez, M. J. Uniform solutions to SAT and Subset Sum by spiking neural P systems, Natural Computing 8 (4) (2009) 681–702.

[Lifeng and Chakrabarti 2013] Lifeng, M. and Chakrabarti, C. "A parallel stochastic computing system with improved accuracy," in *Signal Processing Systems (SiPS), 2013 IEEE Workshop*, October 2013, pp. 195–200.

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

[Liu and Parhi 2015] Liu, Y. and Parhi, K.-K. "Lattice FIR digital filter architectures using stochastic computing," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, April 2015, pp. 1027–1031.

[Maass 1997] Maass, W. Networks of spiking neurons: the third generation of neural network models, Neural Networks 10 (9) 1659–1671.

[Moons and Verhelst 2014] Moons, B. and Verhelst, M. "Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 4, no. 4, pp. 475–486, Dec 2014.

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

[Pan and Păun 2009] Pan, L. and Păun, G. Spiking neural p systems with anti-spikes, International Journal of Computers, Communications & Control, IV (3) (2009) 273–282.

[Pan et al. 2009] Pan,L., Păun, G. and Pérez-Jiménez, M. J. Spiking neural P systems with neuron division and budding, Science China Information Sciences 54 (8) (2011) 1596–1607.

[Parhi and Liu 2014] Parhi, K.-K. and Liu, Y. "Architectures for IIR digital filters using stochastic computing," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, June 2014, pp. 373–376.

[Păun and Păun 2007] Păun, A. and Păun, G. Small universal spiking neural P systems, BioSystems 90 (1) (2007) 48–60.

[Păun 2000] Păun, G. Computing with membranes, Journal of Computer and System Sciences 61 (1) (2000) 108–143.

[Păun 2002] Păun, G.Membrane computing: an introduction, Springer, 2002.

[Păun et al. 2010] Păun, G.,Rozenberg, G., and Salomaa, A. The Oxford handbook of membrane computing, Oxford University Press, 2010.

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

[Saraf et al. 2014] Saraf, N., Bazargan, K., Lilja, D.-J. and Riedel, M.-D. "IIR filters using stochastic arithmetic," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, March 2014, pp. 1–6.

[Song et al. 2016] Song, T., Gong, F., Liu, X., Zhao, Y. and Zhang, X. Spiking Neural P Systems With White Hole Neurons, IEEE Transactions on NanoBioscience 15 (07) (2016) 666–673.

[Song et al. 2013] Song, T., Pan, L., Jiang, K., Song, B. and Chen, W. Normal forms for some classes of sequential spiking neural P systems, IEEE Transactions on NanoBioscience 12 (3) (2013) 255–264.

[Song and Pan 2015] Song,T. and Pan,L. Spiking neural p systems with rules on synapses working in maximum spikes consumption strategy, IEEE Transactions on NanoBioscience 1 (2015) 38–44.

[Song and Pan 2016] Song, T. and Pan, L. Spiking neural P Systems with Request Rules, Neurocomputing 193 (2016) 193–200.

[Song et al. 2012] Song,T., Pan, L. and Păun, G. Asynchronous spiking neural P systems with local synchronization, Information Sciences 219 (2012) 197–207.

[Song et al. 2014] Song, T., Pan, L. and Păun, G. Spiking neural P systems with rules on synapses, Theoretical Computer Science 529 (2014) 82–95.

[Song et al. 2009] Song,T., Wang, X., Zhang, Z. and Chen, Z. Homogenous spiking neural P systems with anti-spikes, Neural Computing and Applications, (2009).

[Wang et al. 2016] Wang, X., Song, T., Gong, F. and Zheng, P. On the Computational Power of Spiking Neural P Systems with Self-Organization, Scientific reports 6-27624 (2016).

[Wang et al. 2010] Wang, T., Zhang, G., Zhao, J., He, Z., Wang,J. and Pérez-Jiménez, M. J. Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, IEEE Transactions on Power Systems 30 (3) (2014) 1182–1194.

[Zeng et al. 2012] Zeng, X., Song, T., Zhang, X. and Pan, L. Performing four basic arithmetic operations with spiking neural P systems, IEEE Transactions on NanoBioscience 11 (4) (2012) 366–374.

[Zeng et al. 2009] Zeng, X., Zhang, X. and Pan,L. Homogeneous spiking neural P systems, Fundamental Informaticae 97 (1) (2009) 275–294.

[Zhang et al. 2014] Ye Tian, Ran Cheng, Xingyi Zhang, Fan Cheng, Yaochu Jin, An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility, IEEE Transactions on Evolutionary Computation, 2017, in press.

[Zhang et al. 2016] Lei Zhang, Hebin Pan, Yansen Su, Xingyi Zhang, Yunyun Niu, A mixed representation based multi-objective evolutionary algorithm for overlapping community detection, IEEE Transactions on Cybernetics, 2017, 47(9): 2703-2716.

[Tian et al. 2016] Ye Tian, Handing Wang, Xingyi Zhang*, Yaochu Jin. Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization, Complex & Intelligent Systems, 2017, in press.

[Zhang et al. 2016] Xingyi Zhang, Ye Tian, Ran Cheng, Yaochu Jin. A decision variable clustering based evolutionary algorithm for large-scale many-objective optimization. IEEE Transactions on Evolutionary Computation, 2016, in press.