INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 13(3), 353-364, June 2018.

Implementation of Arithmetic Operations by SN P Systems with Communication on Request

Y. Jiang, Y. Kong, C. Zhu

Yun Jiang*, Chaoping Zhu
1. Detection and Control of Integrated Systems Engineering Laboratory
2. School of Computer Science and Information Engineering
Chongqing Technology and Business University
Chongqing 400067, China
*Corresponding author: jiangyun@email.ctbu.edu.cn
jsjzcp@163.com

Yuan Kong
College of Mathematics and System Science
Shandong University of Science and Technology
Qingdao 266590, China
kongyuan1122@126.com

Abstract: Spiking neural P systems (SN P systems, for short) are a class of distributed and parallel computing devices inspired from the way neurons communicate by means of spikes. In most of the SN P systems investigated so far, the system communicates on command, and the application of evolution rules depends on the contents of a neuron. However, inspired from the parallel-cooperating grammar systems, it is natural to consider the opposite strategy: the system communicates on request, which means spikes are requested from neighboring neurons, depending on the contents of the neuron. Therefore, SN P systems with communication on request were proposed, where the spikes should be moved from a neuron to another one when the receiving neuron requests that. In this paper, we consider implementing arithmetical operations by means of SN P systems with communication on request. Specifically, adder, subtracter and multiplier are constructed by using SN P systems with communication on request.
Keywords: membrane computing, spiking neural P system, communication on request, arithmetic operation.

1 Introduction

Since Gh. Păun first circulated his idea of membrane computing in 1998 [3] [22] (first circulated as a Turku Center for Computer Science (TUCS) Report 208, 1998), membrane computing has developed rapidly for almost two decades. As a branch of natural computing, membrane computing aims on abstract computing ideas from the structure and the functioning of a single cell, and also from complexes of cells, such as tissues and organs (including the brain) [23]. The computational devices in membrane computing are known as membrane systems (P systems, for short). Till now, three main classes of P systems have been investigated: cell-like P systems [22], tissue-like P systems [13, 50] and neural-like P systems [9]. The present paper deals with a class of neural-like P system, called spiking neural P systems (SN P systems, for short) [9].

SN P systems are a class of distributed parallel computing devices inspired from the way the neurons communicate by sending spikes to each other. In SN P systems, neurons (in the form of membranes) are placed in the nodes of a directed graph, with the edges representing synapses. Each neuron contains a number of identical objects, denoted by a and called spikes. Each neuron may also contain a number of firing rules and forgetting rules. When the contents

of a neuron satisfy some regular expression, a firing rule allows a neuron to send information to other neurons in the form of spikes. On the other hand, forgetting rule removes from the neuron a specified number of spikes. The system evolves by means of firing rules and forgetting rules. And it evolves synchronously, in each time unit, each neuron which can use a rule, no matter firing or forgetting, should use one. When the computation halts, no further rule can be used, and a result is obtained, e.g., in the form of the distance between the first two spikes of the output neuron, or the number of spikes present in a specified neuron in the halting configuration.

Since 2006 there have been quite a few research efforts put forward to SN P systems. Many variants of SN P systems have been proposed, such as asynchronous SN P systems [3], sequential SN P systems [8], SN P systems with anti-spikes [16], homogenous SN P systems [47], SN P systems with astrocytes [19], SN P systems with weighted synapses [21], SN P systems with rules on synapses [32], SN P systems with weights [36], SN P systems with a generalized use of rules [52], SN P systems with white hole neurons [27], SN P systems with request rules [30], cell-like SN P systems [43], extended SN P systems [1], SN P systems with scheduled synapses [2], SN P systems with polarizations [19]. Most of the classes of SN P systems obtained are computationally universal, equivalent in power to Turing machines [4, 11, 15, 31, 33, 35, 40, 42, 46, 51, 53]. An interesting topic is to find small universal SN P systems [14,20,25,28,29,44,54]. In certain cases, polynomial solutions to computationally hard problems can also be obtained in this framework [10, 17]. Moreover, SN P systems have been applied to solve real-life problems [24] [21], for example, to design logic gates, logic circuits [34] and databases [5], to represent knowledge [38], to diagnose faults [26,37,39], or to approximately solve combinatorial optimization problems [49].

SN P systems can also be applied in a very different way, where they are viewed as components of a restricted Arithmetic Logic Unit. Some SN P systems were constructed for dealing with basic arithmetic operations. These systems apply different encoding method. In [7], the binary number is encoded as a sequence of spikes: at each time unit, zero or one spike will be provided to the input neuron, depending on the corresponding bit being 0 or 1. The numbers used in [45] are encoded as the interval of time elapsed between two spikes. Under the third encoding mechanism, natural numbers are encoded in the form of spike train and introduced in the system through the input neurons, while the results of arithmetic operations are encoded in the form of the number of spikes emitted to the environment [13].

These SN P systems mentioned above perform communication on command, that is the initiative for communication belongs to the emitting neuron. Specifically speaking, the application of evolution rules depends on the contents of a neuron, (as mentioned above, checked by a regular expression), a specified number of spikes are consumed and a specified number of spikes are produced, and then sent to each neurons linked to the evolving neuron by a synapse. Inspired from parallel-cooperating grammar systems, it is natural to consider the opposite strategy – communication on request. In this case, spikes are requested from neighboring neurons, depending on the contents of the neighboring neuron (also checked by a regular expression). On the other hand, no spike is consumed or created, they are only moved from a neuron to another one along synapses when the receiving neuron requests that. This request-response communication is an important concept in software engineering, and computers use it as a basic method to communicate with each other. Recently, communication on request was introduced into SN P systems by Pan et al., and this variant of SN P systems is shortly called SNQ P systems [18]. Communication on request is a powerful feature in SN P systems: SNQ P systems using two types of spikes are proved to be universal, equivalent with Turing machines, and it is reported that 49 neurons are sufficient for SNQ P systems to achieve Turing universality.

In this work, SN P systems with communication on request for performing the arithmetic operations are introduced. The arithmetic operations we will consider are addition, subtraction and multiplication. Natural numbers can be encoded in the form of the number of spikes and

introduced in the system through input neurons. And then by performing the computation of the system, a number of spikes are present in the output neuron when the system halts. By analyzing the number of specific spikes in the output neuron, we can obtain the result of this arithmetic operation.

The paper is organized as follows. In the next section we recall some preliminaries that will be used in the following, including the formal definition of SN P systems with communication on request. In Section 3.1 we present an SN P system with communication on request that is used to add two natural numbers. A subtracter based on SN P system with communication on request is given in Section 3.2. An SN P system with communication on request for multiplication is constructed in Section 3.3. Conclusions and some open problems for future works are present in Section 4.

2 Spiking neural P systems with communication by request

Formally, a spiking neural P system with communication on request (shortly, SNQ P system), with k types of spikes, is a construct of the form (this form is almost the same as the one appearing in [18], except one reasonable change by introducing input neurons)

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, a_{i_{in}}, a_{i_{out}}, in, out),$$

where:

1. $O = \{a_1, a_2, \ldots, a_k\}$ is an alphabet (a_i is a type of spikes), $k \geq 1$;
2. $\sigma_1, \sigma_2, \ldots, \sigma_m$ are neurons, of the form

$$\sigma_i = (a_1^{n_1} a_2^{n_2} \ldots a_k^{n_k}, R_i), 1 \leq i \leq m, n_j \geq 0, 1 \leq j \leq k,$$

   where:

   a) $n_j \geq 0$ is the initial number of spikes of type a_j contained in the neuron σ_j, $1 \leq j \leq k$;
   b) R_i is a finite set of rules of the form E/Qw, with w a finite non-empty list of queries of the forms (a_s^p, j) and (a_s^∞, j), $1 \leq s \leq k$, $p \geq 0$, $1 \leq j \leq m$, or $j = env$;

3. $a_{i_{in}}$, $a_{i_{out}}$, $1 \leq i_{in}, i_{out} \leq k$, are the types of input spikes and output spikes,
4. $in \subseteq \{1, 2, \ldots, m\}$ indicate the input neurons, and $out \in \{1, 2, \ldots, m\}$ indicates the output neurons, respectively.

A query (a_s^p, j) means that neuron σ_i requests p copies of a_s from neuron σ_j, while the meaning of (a_s^∞, j) is that all spikes of type a_s from σ_j, no matter how many they are, are requested by σ_i. Specifically, a query of the form (a, env) is allowed to be used, which means that one copy of a is requested from the environment – with the environment supposed to contain arbitrarily many copies of a. This kind of rules can be removed [18], so it will not effect the arithmetic operations.

A rule of the form E/Qw can be used if both of the following conditions are satisfied: (1) the contents of the neuron are described by the regular expression E; (2) all queries formulated in w are satisfied (for example, if σ_j contains strictly less than p spikes, then the query (a_s^p, j) is not satisfiable). Specifically, there is a situation called the conflicting queries, where two different neurons σ_{i_1}, σ_{i_2} ask different numbers of occurrences of the same spike a_s from the same neuron σ_j (namely, two queries of the forms (a_s^p, j), (a_s^r, j) with $p \neq r$, or of the forms (a_s^p, j), (a_s^∞, j) for

p a given number). In the case of conflicting queries, the two rules cannot be used simultaneously, but one of them, non-deterministically chosen, can be used.

In SNQ P systems, the definition of a computational step is quite delicate because of the interplay of the queries. A computational step is described in terms of three sub-steps: (1) In each neuron, a rule is chosen to be applied, and its applicability is checked; (2) The requested spikes are removed from the neurons where they were present. (3) The queries are satisfied, the requested spikes are moved to the requesting neuron. The three sub-steps together form a step, which lasts one time unit.

An SNQ P system starts from the initial configuration, which is described by the number of spikes of each type present in each neuron in the beginning of the computation. Then it proceeds by applying the rules synchronously, which means that in each neuron if a rule can be used, then it is applied according to the procedure described above. After a computation step, we can define transitions configurations. Any sequence of transitions starting from the initial configuration is called a computation. A computation halts if reaches a configuration where no rule can be used. The result of a halting computation is the number of copies of spikes $a_{i_{out}}$ present in neuron σ_{out} in the halting configuration.

In order to perform arithmetic operations, it is necessary to introduce the numbers to be computed into the system, which may be encoded in many different ways. Here, we use the way discussed in [7]. A positive integer number is given as input to a specified input neuron. The number is specified as the number of input spikes initially contained in the input neuron. The result of the operation is encoded as the number of output spikes present in the output neuron when a computation halts.

In the next sections SNQ P systems are represented graphically, which is easy to understand. An oval with the initial number of spikes and rules inside is used to represent a neuron. The input neurons have incoming arrows and the output neuron have outgoing arrows, suggesting their communication with other devices (or the environment).

3 Performing arithmetic operations by SN P systems with communication on request

3.1 An SNQ P system for addition

In this section we present an SN P system with communication on request, as shown in Fig. 1, for dealing with the addition of two arbitrary natural numbers. System Π_{add} is composed of 5 neurons: two specified neurons are used as input neurons, where the summand and addend are introduced, and one neuron is used for giving the obtained result.

Theorem 1. *For two arbitrary natural numbers x and y, SN P system with communication on request Π_{add} computes the addition of x and y.*

Proof: We construct a system Π_{add} of the form

$$\Pi_{add} = (\{a, b, c_1\}, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \{a, b\}, \{a\}, \{1, 2\}, 5),$$

where:
$\sigma_1 = (a^x, \emptyset)$,
$\sigma_2 = (b^y, \emptyset)$,
$\sigma_3 = (\lambda, \{\lambda/Q(c_1, env), c_1/Q(a^\infty, 1)\})$,
$\sigma_4 = (\lambda, \{\lambda/Q(c_1, env), c_1/Q(b, 2), bc_1/Q(a, env)\})$,
$\sigma_5 = (\lambda, \{\lambda/Q(c_1, env), a^*b^*c_1/Q(a^\infty, 3)(ab, 4)\})$.

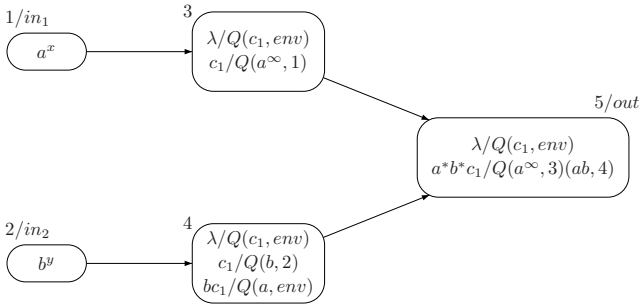

Figure 1: The structure of adder Π_{add}

SNQ P system Π_{add} functions as follows. In the initial configuration of Π_{add}, all neurons are empty. The summand x and the addend y are encoded as spikes a^x and b^y, and are provided to neuron σ_{in_1} and neuron σ_{in_2}, respectively. Since neurons σ_3, σ_4 and σ_{out} are empty, all of them can use the rule $\lambda/Q(c_1, env)$, the spikes c_1 arrives into them, and these neurons become active. With spike c_1 inside, neuron σ_3 can absorb from neuron σ_{in_1} all spikes a , which will be requested by neuron σ_{out} in a later step. In the meantime, neuron σ_4 absorb spike b, one by one, from neuron σ_{in_2}. In the next step, the spike b in neuron σ_4 will absorb from the environment one spike a, and then both the spike a and spike b are requested by neuron σ_{out} together. In this way, through neuron σ_3, the spikes a in σ_{in_1} move to σ_{out} at one time, and through neuron σ_4, the spikes b in σ_{in_2} move to σ_{out} one by one, together with a spike a every time. When the last spike b in σ_{in_2} is requested by neuron σ_4, and then moves to σ_{out} together with a spike a, all the spikes a and b in neurons σ_{in_1}, σ_{in_2}, σ_3 and σ_4 are exhausted. The computation halts, because there is no rule that can be used in the system. At this time, the spikes present in neuron σ_{out} are $a^{x+y}b^y$ (a^x absorbed from neuron σ_3 and $(ab)^y$ absorbed from neuron σ_4). The number of spikes a from the output neuron is $x + y$, which means that the result computed by the system is $x + y$.

Table 1: Spikes in each neuron of Π_{add} at each step during the computation of the addition $5 + 2 = 7$

step	$1/in_1$	$2/in_2$	3	4	$5/out$
0	$\mathbf{a^5}$	$\mathbf{b^2}$	λ	λ	λ
1	a^5	b^2	c_1	c_1	c_1
2	λ	b	a^5c_1	bc_1	c_1
3	λ	b	a^5c_1	abc_1	c_1
4	λ	b	c_1	c_1	a^6bc_1
5	λ	λ	c_1	bc_1	a^6bc_1
6	λ	λ	c_1	abc_1	a^6bc_1
7	λ	λ	c_1	c_1	$\mathbf{a^7}b^2c_1$

With the explanation above, readers can check that, for given $x, y > 0$, system Π_{add} can correctly compute the addition of x and y, which completes the proof. □

As an example, let us consider the addition $5 + 2 = 7$. Table 1 reports the spikes contained in each neuron of Π_{add} at each step during the computation. The input and output spikes are written in bold.

3.2 An SNQ P system for subtraction

We now describe an SN P system with communication on request Π_{sub} used as subtracter, which is shown in Fig. 2. System Π_{sub} is composed of 6 neurons, where two specified neurons are used to introduce the minuend and subtrahend into the system, and one neuron is used for giving the obtained result.

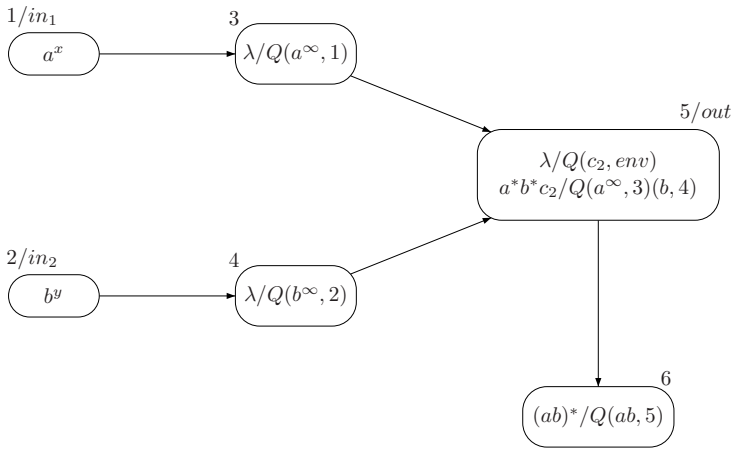

Figure 2: The structure of subtracter Π_{sub}

Theorem 2. *For two arbitrary natural numbers x and y, where $x > y > 0$, SN P system with communication on request Π_{sub} computes the subtraction of x and y.*

Proof: We construct a system Π_{sub} of the form

$$\Pi_{sub} = (\{a, b, c_2\}, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \{a, b\}, \{a\}, \{1, 2\}, 5),$$

where:
$\sigma_1 = (a^x, \emptyset)$,
$\sigma_2 = (b^y, \emptyset)$,
$\sigma_3 = (\lambda, \{\lambda/Q(a^\infty, 1)\})$,
$\sigma_4 = (\lambda, \{\lambda/Q(b^\infty, 2)\})$,
$\sigma_5 = (\lambda, \{\lambda/Q(c_2, env), a^*b^*c_2/Q(a^\infty, 3)(b, 4)\})$,
$\sigma_6 = (\lambda, \{(ab)^*/Q(ab, 5)\})$.

SNQ P system Π_{sub} functions as follows. In the initial configuration of Π_{sub}, all neurons are empty. The minuend x and the subtrahend y are encoded as spikes a^x and b^y, and are provided to neuron σ_{in_1} and neuron σ_{in_2}, respectively. Since neurons σ_3, σ_4 are empty, the spikes a^x will be absorbed by σ_3, and the spikes b^y by σ_4, respectively. In the meantime, neuron σ_5 can use the rule $\lambda/Q(c_2, env)$, and spike c_2 arrives in it. With spike c_2 inside, the neuron σ_5 will absorb one spike a from neuron σ_3 and one spike b from σ_4. In the next step, this pair of spikes a and b will move to the neuron σ_6, and spike c_2 remains in the neuron σ_5. So the absorbability of pairs of spikes a and b continues. When the spikes b in σ_2 get exhausted, the last spike b will be absorbed by σ_4, moves to σ_5 together with one spike a from σ_3, and this last pair of spikes of a and b moves to σ_6 at last. At this time, there is no rule can be used in the system, so the computation halts. During the computation there are y pairs of a and b moves to σ_6, the spikes b get exhausted in neuron σ_4, and there are $x - y$ spikes of a left in neuron σ_3. At the end of computation, the number of spikes a present in the output neuron is $x - y$, which means that the result computed by the system is $x - y$.

Table 2: Spikes in each neuron of Π_{sub} at each step during the computation of the subtraction $5-2=3$

step	$1/in_1$	$2/in_2$	3	4	$5/out$	6
0	$\mathbf{a^5}$	$\mathbf{b^2}$	λ	λ	λ	λ
1	λ	λ	a^5	b^2	c_2	λ
2	λ	λ	λ	b	a^5bc_2	λ
3	λ	λ	λ	λ	a^4bc_2	ab
4	λ	λ	λ	λ	$\mathbf{a^3}c_2$	a^2b^2

With the explanation above, readers can check that, for given $x > y > 0$, system Π_{sub} can correctly compute the subtraction of x and y, which completes the proof. □

As an example let us calculate $5-2=3$. Table 2 reports the spikes that occur in each neuron of Π_{sub} at each step during the computation. Also, the input and output spikes are written in bold.

3.3 An SNQ P System for multiplication

In this section, we present an SN P system with communication on request Π_{mul}, as shown in Fig. 3 with 6 neurons, for implementing the multiplication of two arbitrary natural numbers.

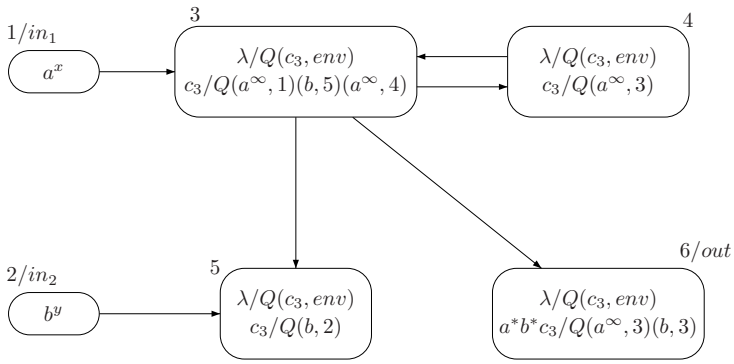

Figure 3: The structure of multiplier Π_{mul}

Theorem 3. *For two arbitrary natural numbers x and y, where $x, y > 0$, SN P system with communication on request Π_{mul} computes the multiplication of x and y.*

Proof: We construct a system Π_{mul} of the form

$$\Pi_{mul} = (\{a, b, c_3\}, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \{a, b\}, \{a\}, \{1, 2\}, 6),$$

where:
$\sigma_1 = (a^x, \emptyset)$,
$\sigma_2 = (b^y, \emptyset)$,
$\sigma_3 = (\lambda, \{\lambda/Q(c_3, env), c_3/Q(a^\infty, 1)(b, 5)(a^\infty, 4)\})$,
$\sigma_4 = (\lambda, \{\lambda/Q(c_3, env), c_3/Q(a^\infty, 3)\})$,
$\sigma_5 = (\lambda, \{\lambda/Q(c_3, env), c_3/Q(b, 2)\})$,
$\sigma_6 = (\lambda, \{\lambda/Q(c_3, env), a^*b^*c_3/Q(a^\infty, 3)(b, 3)\})$.

SNQ P system Π_{mul} functions as follows. In the initial configuration of Π_{mul}, all neurons are empty. The multiplicand x and the multiplier y are encoded as spikes a^x and b^y, and are provided to neuron σ_{in_1} and neuron σ_{in_2}, respectively. Since neurons σ_3, σ_4, σ_5 and σ_{out} are empty, all of them can use the rule $\lambda/Q(c_3, env)$, the spikes c_3 arrive in them, and these neurons become active.

With spike c_3 inside, neuron σ_5 can absorb from neuron σ_{in_2} one spike b. Now spike b is present in σ_5, so the rule $c_3/Q(a^\infty, 1)(b, 5)(a^\infty, 4)$ in neuron σ_3 can be used: all spikes a in neuron σ_{in_1} and the one spike b in neuron σ_5 are requested by neuron σ_3 (there is no spike a in neuron σ_4, so no spike a requested). After this rule is used, neuron σ_{in_1} becomes empty, and spike c_3 is left in neuron σ_5, which means neuron σ_5 can use again the rule $c_3/Q(b, 2)$. In the next step, the spikes a^x and b are requested by the neuron σ_{out}, and the spike c_3 is left in neuron σ_5. This is the first time the spikes a^x arrive in the output neuron. In the meantime, with spike c_3 present in neuron σ_4 and σ_5, neuron σ_5 absorb the second spike b from neuron σ_{in_2}, neuron σ_4 absorb the spikes a^x from neuron σ_3, Also with the present of spike c_3, these spikes a^x and b will be requested by neuron σ_3 in the next step, and then moves to neuron σ_{out}, which is the second time the spikes a^x arrive in the output neuron.

Table 3: Spikes in each neuron of Π_{mul} at each step during the computation of the subtraction $5 \times 2 = 10$

step	$1/in_1$	$2/in_2$	3	4	5	$6/out$
0	$\mathbf{a^5}$	$\mathbf{b^2}$	λ	λ	λ	λ
1	a^5	b^2	c_3	c_3	c_3	c_3
2	a^5	b	c_3	c_3	bc_3	c_3
3	λ	b	a^5bc_3	c_3	c_3	c_3
4	λ	λ	c_3	a^5c_3	bc_3	a^5bc_3
5	λ	λ	a^5bc_3	c_3	c_3	a^5bc_3
6	λ	λ	c_3	a^5c_3	c_3	$\mathbf{a^{10}}b^2c_3$

With the above explanation, readers can check that, the spikes a^x in neuron σ_{in_1} finally arrive y times at the output neuron, and at the end of the computation, the number of spikes a present in the output neuron is xy, which means that the result computed by the system is xy. So for given $x, y > 0$, system Π_{mul} can correctly compute the product of x and y, which completes the proof. □

For example let us consider $5 \times 2 = 10$. Table 3 reports the spikes that occur in each neuron of Π_{mul} at each step during the computation. Also, the input and output spikes are written in bold.

4 Conclusions and future work

Using the SN P systems with communication on request instead of the traditional SN P systems communicating on command, we have restudied the problem of considering SN P systems as components of an arithmetic logic unit. Specifically speaking, we have proposed three SN P systems with communication on request to implement addition, subtraction and multiplication of two arbitrary natural numbers, respectively. In these systems, natural numbers are introduced into the system as the number of some spike in input neuron, while the result of an arithmetic operation is the number of a specified spike present in output neuron at the end of computation.

First of all, it is an urgent task to propose an SNQ P system to compute the division between two natural numbers, and this one is probably the most difficult to design. In this work, the adder, subtracter and multiplier contain 5 neurons, 6 neurons and 6 neurons, respectively. The number of neurons is less than that is used in [45] (10 neurons, 12 neurons and 26 neurons, respectively), but it has no obvious advantage when compared to that is used in [13] (2 neurons, 2 neurons and 11 neurons, respectively). Therefore, it deserves to be investigated whether the SNQ P systems for arithmetic operations can be simplified by carefully examining the structure, or by using a different construction. Besides, for the further investigation, it is natural to mention this problem: how to construct an SNQ P system to implement arithmetic operations with signed number.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61502063 and 61602188) and Chongqing Social Science Planning Project (2017YBGL142).

Bibliography

[1] Alhazov A., Freund R., Ivanov S., Oswald M., Verlan S. (2017); Extended spiking neural P systems with hole rules and their red-green variants. *Natural Computing*, 2-3, 1–14, 2017.

[2] Cabarle F., Adorna H., Jiang M., Zeng X. (2017); Spiking neural p systems with scheduled synapses. *IEEE Transactions on Nanobioscience*, 16, 792–801, 2017.

[3] Cavaliere M., Ibarra O.H., Păun Gh., Egecioglu O., Ionescu M., Woodworth S. (2009); Asynchronous spiking neural P systems. *Theoretical Computer Science*, 410, 2352–2364, 2009.

[4] Chen H., Freund R., Ionescu M. (2007); On string languages generated by spiking neural P systems, *Fundamenta Informaticae*, 75, 141–162, 2007.

[5] Díaz-Pernil, D., Gutiérrez-Naranjo, M.A. (2018); Semantics of Deductive Databases with Spiking Neural P Systems, *Neurocomputing*, 272, 365-373, 2018

[6] Dzitac, I. (2015); Impact of Membrane Computing and P Systems in ISI WoS. Celebrating the 65th Birthday of Gheorghe Păun, *International Journal of Computers Communications & Control*, 10(5), 617–626, 2015.

[7] Gutiérrez-Naranjo, M.A., Leporati, A. (2009); First steps towards a CPU made of spiking neural P systems, *International Journal of Computers Communications & Control*, 4(3), 244–252, 2009.

[8] Ibarra O.H., Păun A., Rodríguez-Patón A. (2009); Sequential SN P systems based on min/-max spike number, *Theoretical Computer Science*, 410, 2982–2991, 2009.

[9] Ionescu M., Păun Gh., Yokomori T. (2006); Spiking neural P systems, *Fundamenta Informaticae*, 71, 279–308, 2006.

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

[11] Krithivasan K., Metta V.P., Garg D. (2011); On string languages generated by spiking neural P systems with anti-spikes. *International Journal of Foundations of Computer Science*, 22, 15–27, 2011.

[12] Liu X., Li Z., Liu J., Liu L., Zeng X. (2015); Implementation of arithmetic operations with time-free spiking neural P systems, *IEEE Transactions on Nanobioscience*, 14, 617–624, 2015.

[13] Martín-Vide C., Păun Gh., Pazos J., Rodríguez-Patón A. (2003); Tissue P systems, *Theoretical Computer Science*, 296, 295–326, 2003.

[14] Metta V.P., Raghuraman S., Krithivasan K. (2014); Small universal simple spiking neural P systems with cooperating rules as function computing devices, *Lecture Notes in Computer Science*, 8961, 300–313, 2014.

[15] Neary T. (2009); A boundary between universality and non-universality in extended spiking neural P systems, *Lecture Notes in Computer Science*, 6031, 475–487, 2009.

[16] Pan L., Păun Gh. (2009); Spiking neural P systems with anti-spikes, *International Journal of Computers Communnication & Control*, 4(3), 273–282, 2009.

[17] Pan L., Păun Gh., Pérez-Jiménez M.J. (2011); Spiking neural P systems with neuron division and budding, *Science China Information Sciences*, 54, 1596–1607, 2011.

[18] Pan L., Păun Gh., Zhang G., Neri F. (2017); Spiking neural P systems with communication on request, *International Journal of Neural Systems*, 27, 1750042, 2017.

[19] Pan L., Wang J., Hoogeboom H.J. (2012); Spiking neural P systems with astrocytes, *Neural Computation*, 24, 805–825, 2012.

[20] Pan L., Zeng X. (2010); A note on small universal spiking neural P systems, *Lecture Notes in Computer Science*, 5957, 436–447, 2010.

[21] Pan L., Zeng X., Zhang X., Jiang Y. (2012); Spiking neural P systems with weighted synapses, *Neural Processing Letters*, 35, 13–27, 2012.

[22] Păun Gh. (2000); Computing with membranes, *Journal of Computer and System Sciences*, 61, 108–143, 2000.

[23] Păun Gh. (2002); *Membrane Computing: An Introduction*, Springer, 2002.

[24] Păun Gh. (2016); Membrane Computing and Economics: A General View, *International Journal of Computers Communnication & Control*, 11, 105–112, 2016.

[25] Păun Gh., Păun A. (2007); Small universal spiking neural P systems, *Biosystems*, 90, 48–60, 2007.

[26] Peng H., Wang J., Pérez-Jiménez M.J., Wang H., Shao J., Wang T. (2013); Fuzzy reasoning spiking neural P systems for fault diagnosis, *Information Sciences*, 235, 106–116, 2013.

[27] Song T., Gong F., Liu X., Zhao Y., Zhang X. (2016); Spiking neural P systems with white hole neurons, *IEEE Transactions on Nanobioscience*, 15, 666-673, 2016.

[28] Song T., Jiang Y., Shi X., Zeng X. (2013); Small universal spiking neural P systems with anti-spikes, *Journal of Computational and Theoretical Nanoscience*, 10, 999–1006, 2013.

[29] Song T., Pan L. (2014); A small universal spiking neural P systems with cooperating rules, *Romanian Journal of Information Science and Technology*, 17, 177–189, 2014.

[30] Song T., Pan L. (2016); Spiking neural P systems with request rules, *Neurocomputing*, 193, 193–200, 2016.

[31] Song T., Pan L., Jiang K., Song B., Chen W. (2013); Normal forms for some classes of sequential spiking neural P systems, *IEEE Transactions on Nanobioscience*, 12, 255–264, 2013.

[32] Song T., Pan L., Păun Gh. (2014); Spiking neural P systems with rules on synapses, *Theoretical Computer Science*, 529, 82–95, 2014.

[33] Song T., Xu J., Pan L. (2015); On the universality and non-universality of spiking neural P systems with rules on synapses, *IEEE Transactions on Nanobioscience*, 14, 960–966, 2015.

[34] Song T., Zheng P., Wong M.L., Wang X. (2016); Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control, *Information Sciences*, 372, 380–391, 2016.

[35] Su Y., Wu T., Xu F., Păun A. (2017); Spiking neural p systems with rules on synapses working in sum spikes consumption strategy, *Fundamenta Informaticae*, 156, 187–208, 2017.

[36] Wang J., Hoogeboom H.J., Pan L., Păun Gh., Pérez-Jiménez M.J. (2014); Spiking neural P systems with weights, *Neural Computation*, 22, 2615–2646, 2014.

[37] Wang J., Peng H. (2013); Adaptive fuzzy spiking neural P systems for fuzzy inference and learning, *International Journal of Computer Mathematics*, 90, 857–868, 2013.

[38] Wang J., Shi P., Peng H., Pérez-Jiménez M.J., Wang T. (2013); Weighted fuzzy spiking neural P systems, *IEEE Transactions on Fuzzy Systems*, 21, 209–220, 2013.

[39] Wang T., Zhang G., Zhao J., He Z., Wang J., Pérez-Jiménez M.J. (2015); Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, *IEEE Transactions on Power Systems*, 30, 1182–1194, 2015.

[40] Wang X., Song T., Gong F., Zheng P. (2016); On the computational power of spiking neural P systems with self-organization, *Scientific Reports*, 6: 27624, 2016.

[41] Wu T., Păun A., Zhang Z., Pan L. (2017); Spiking neural P systems with polarizations, *IEEE Transactions on Neural Networks and Learning Systems*, 1–12, 2017.

[42] Wu T., Zhang Z., Pan L. (2016); On languages generated by cell-like spiking neural P systems, *IEEE Transactions on Nanobioscience*, 15, 455–467, 2016.

[43] Wu T., Zhang Z., Păun Gh., Pan L. (2016); Cell-like spiking neural P systems, *Theoretical Computer Science*, 623, 180–189, 2016.

[44] Zeng X., Pan L., Pérez-Jiménez M.J. (2014); Small universal simple spiking neural P systems with weights, *Science China Information Sciences*, 57, 1–11, 2014.

[45] Zeng X., Song T., Zhang X., Pan L. (2012); Performing four basic arithmetic operations with spiking neural P systems, *IEEE Transactions on Nanobioscience*, 11, 366–374, 2012.

[46] Zeng X., Xu L., Liu X. (2014); On string languages generated by spiking neural P systems with weights, *Information Sciences*, 278, 423–433, 2014.

[47] Zeng X., Zhang X., Pan L. (2009); Homogenous spiking neural P systems, *Fundamenta Informaticae*, 97, 275–294, 2009.

[48] Zhang G. (2017); *Real-life applications with membrane computing*, Springer, 2017.

[49] Zhang G., Rong H., Neri F., Pérez-Jiménez M.J. (2014); An optimization spiking neural P system for approximately solving combinatorial optimization problems, *International Journal of Neural Systems*, 24, 1440006, 2014.

[50] Zhang X., Liu Y., Luo B., Pan L. (2014); Computational power of tissue P systems for generating control languages, *Information Sciences*, 278, 285–297, 2014.

[51] Zhang X., Pan L., Păun A. (2015); On universality of axon P systems, *IEEE Transactions on Neural Networks and Learning Systems*, 26, 2816–2829, 2015.

[52] Zhang X., Wang B., Pan L. (2014); Spiking neural P systems with a generalized use of rules, *Neural Computation*, 26, 2925–2943, 2014.

[53] Zhang X., Zeng X., Luo B., Pan L. (2014); On some classes of sequential spiking neural P systems, *Neural Computation*, 26, 974–997, 2014.

[54] Zhang X., Zeng X., Pan L. (2008); Smaller universal spiking neural P systems, *Fundamenta Informaticae*, 87, 117–136, 2008.