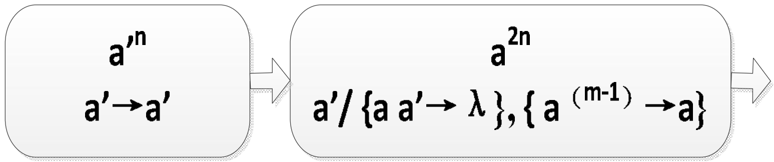

Fig. 5: Division in SNPC system generating string language

Table2 Process of division in SNPC system generating string language

Step	Neuron1			Neuron2			output
	start		end	start		end	
1	a'^{n}	a'→a'	$a'^{(n-1)}$	a' (1)		a'	
2	$a'^{(n-1)}$	a'→a'	$a'^{(n-2)}$	a' $a^{(2n)}$	a'/ {a a'→ λ, $a^{(m-1)}$→a}	$a^{(2n-m)}$	a
3	$a'^{(n-2)}$	a'→a'	$a'^{(n-3)}$	a' $a^{(2n-m)}$	a'/ {a a'→ λ, $a^{(m-1)}$→a}	$a^{(2n-2m)}$	2a
4	$a'^{(n-3)}$	a'→a'	$a'^{(n-4)}$	a' $a^{(2n-2m)}$	a'/ {a a'→ λ, $a^{(m-1)}$→a}	$a^{(2n-3m)}$	3a
i-1	$a'^{(n-(i-2))}$	a'→a'	$a'^{(n-(i-1))}$	a' $a^{(2n-(i-3)m)}$	a'/ {a a'→ λ, $a^{(m-1)}$→a}	$a^{(2n-(i-2)m)}$	(i-2)a
i	$a'^{(n-(i-1))}$	a'→a'	$a'^{(n-i)}$	a' $a^{(2n-(i-2)m)}$	a'/ {a a'→ λ, $a^{(m-1)}$→a}	$a^{(2n-(i-1)m)}$	(i-1)a
i+1	$a'^{(n-i)}$	a'→a'	$a'^{(n-(i+1))}$	a' $a^{(2n-(i-1)m)}$	a'/ {a a'→ λ, $a^{(m-1)}$→a}	$a^{(2n-im)}$	ia
k-1	$a'^{(n-(k-2))}$	a'→a'	$a'^{(n-(k-1))}$	a^{3m}	a'/ {a a'→ λ, $a^{(m-1)}$→a}	a^{2m}	(k-2)a
k	$a'^{(n-(k-1))}$	a'→a'	$a'^{(n-k)}$	a^{2m}	a'/ {a a'→ λ, $a^{(m-1)}$→a}	a^{m}	(k-1)a
k+1	$a'^{(n-k)}$	a'→a'	$a'^{(n-(k+1))}$	a^{m}	a'/ {a a'→ λ, $a^{(m-1)}$→a}	φ	ka

As shown in Table 2, the SNPC system completes the division (m/n) of two arbitrary natural numbers *m* and *n* ($n \geq m$).

4 Arithmetic Operation in SNPC System Generating Datasets

The SNPC system generating datasets is referred to the outputs are defined as the time lag of output neuron exporting the first two spikes. In this paper, the one step of the computing process is seen as a time unit, and the result is the steps of output neuron exporting the first two spikes. Due to the irreversibility of the time, the SNPC system generating datasets is more suitable for the addition and multiplication. Now the algorithm that performing arithmetic operation (addition and multiplication) of two arbitrary natural numbers in SNPC system generating datasets is given.

4.1 Addition

For achieving the addition of two arbitrary natural numbers in SNPC system generating datasets, the structure of the membrane system is designed as followed:

$\prod(Addition) = (O, \sigma1, \sigma2, \sigma3, syn, in, out)$, where,

- $O = \{a\}$;
- $\sigma1 = \{n1, R1\}, \sigma2 = \{n2, R2\}, \sigma3 = \{n3, R3\}$;
- $n1 = \{1|\alpha\}$, $n2 = \{O|\Phi\}$, $n3 = \{1|\alpha\}$;
- $R1=\{a/a\rightarrow a;m-1\}, R2=\{a\rightarrow a;n-1\}, R3=\{a\rightarrow a;0\}$;
- in=l, out=3

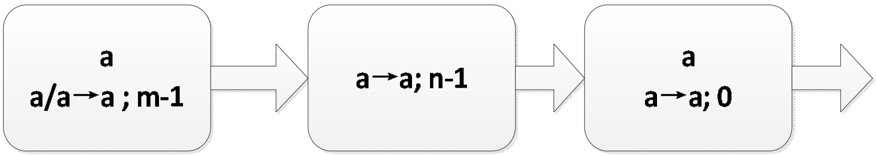

Fig. 6: Addition in S NPC system generating datasets

As shown in Fig.6, there are one object a and one firing rule a/a → a; m−1 (m−1 is the delay)in membrane σ1. There is object a in membrane σ1, the restrain a met, so the rule a/a→a; m−1 can be executed, consuming one a spike and generating one a spike. Because of the delay m−1, from the step 2 to step m, the membrane σ1 is closed, not performing the rules, receiving and sending spikes. In membrane σ2, due to the lack of object a, the rule a→a;n−1 can't be fired until receives the object a from membrane σ1. There are one object a and one firing rule a→a;0 in membrane σ3. Due to the existence of object a, at the step one, the rule a→a;0 can be inspired, and send one a spike to environment at once. This is the first time the system exports objects to environment. When receive the a spike from membrane σ2, the rule a→a;0 will be inspired again, and send one a spike to environment at once. This is the second time the system exports objects to environment. The step lag between the two exports is the computing result.
The specific execution is as shown in Table 3.

Table 3 The process of addition in SNPC system generating datasets

Step	Neuron1			Neuron2			Neuron3		
	start		end	start		end	start		end
1	a	a→a	a				a	a→a	φ
2	a	inactive	a						
3	a	inactive	a						
4	a	inactive	a						
m	a	inactive release a	φ	a		a			
m+1				a	a→a	a			
m+2				a	inactive	a			
m+3				a	inactive	a			
m+n				a	inactive release a	φ	a		a
m+n+1							a	a→a	φ

As shown in Table 3, the SNPC system completes the addition (m+n) of two arbitrary natural numbers *m* and *n*.

4.2 Multiplication

For achieving the multiplication of two arbitrary natural numbers in SNPC system generating datasets, the structure of the membrane system is designed as followed:

$\prod(Multiplication) = (O, \sigma 1, \sigma 2, \sigma 3, syn, in, out)$, where,

- $O = \{a, a'\}$;
- $\sigma 1 = \{n1, R1\}$, $\sigma 2 = \{n2, R2\}$, $\sigma 3 = \{n3, R3\}$;
- $n1 = \{m|a\}$, $n2 = \{0|\phi\}$, $n3 = \{1|a'\}$;
- $R1 = \{a/a \rightarrow a; n-2\}$, $R2 = \{a \rightarrow a; m\}$, $R3 = \{a' \rightarrow a', a^m \rightarrow a^m; 0\}$;
- in =1, out=3.

As shown in Fig.7, there are *m* objects *a* and one firing rule $a/a \rightarrow a$; $n-2$ ($n-2$ is the delay) in membrane $\sigma 1$. There is object *a* in membrane $\sigma 1$, the restrain *a* met, so the rule $a \rightarrow a$; $n-2$ can be executed, consuming one *a* spike and generating one *a* spike. Because of the delay $n-2$, from the step 2 to step $n-1$, the membrane $\sigma1$ is closed, not performing the rules, receiving and sending spikes. In membrane $\sigma2$, due to the lack of object *a*, the rule $a \rightarrow a$; *m* can't be fired until receives the object *a* from membrane $\sigma1$. There are one object a' and one firing rule $a' \rightarrow a'$; 0 in membrane $\sigma3$. Due to the existence of object a', at the step one, the rule $a' \rightarrow a'$; 0 can be inspired, and send one a' spike to environment at once. This is the first time the system exports objects to environment. When receive *m a* spikes from membrane $\sigma2$, the rule $a^m \rightarrow a^m$; 0 will be inspired, and send *m a* spikes to environment at once. This is the second time the system exports objects to environment. The step lag between the two exports is the computing result.

For performing multiplication in SNPC system generating datasets, m=3, n=6 as example, the specific execution is as shown in Table 4.

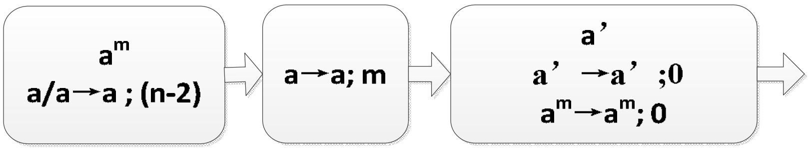

Fig. 7: Multiplication in SNPC system generating datasets

Table 4 The process of Multiplication in SNPC system generating datasets

Step	Neuron1			Neuron2			Neuron3		
	start		end	start		end	start		end
1	a^3	a→a	a^3				a'	a'→a'	φ
2	a^3	inactive	a^3						
3	a^3	inactive	a^3						
4	a^3	inactive	a^3						
5	a^3	inactive release a	a^2	a		a			
6	a^2	a→a	a^2	a	a→a	a			
7	a^2	inactive	a^2	a	inactive	a			
8	a^2	inactive	a^2	a	inactive	a			
9	a^2	inactive	a^2	a	inactive release a	φ	a		a
10	a^2	inactive release a	a	a		a			
11	a	a→a	a	a	a→a	a			
12	a	inactive	a	a	inactive	a			
13	a	inactive	a	a	inactive	a			
14	a	inactive	a	a	inactive release a	φ	a^2		a^2
15	a	inactive release a	φ	a		a			
16				a	a→a	a			
17				a	inactive	a			
18				a	inactive	a			
19				a	inactive release a	φ	a^3		a^3
20							a^3	$a^3 \rightarrow a^3$	φ

As shown in Table 4, the SNPC system completes the multiplication (m+n) of two arbitrary natural numbers *m* and *n*.

5 Conclusion

Due to the objects of each membrane in membrane system can evolve in the way of maximum parallel principle with high parallelism, P system, a computational model, is with the computing power to solve complex problems. From the above deduction process, we can see the SNPC system with high parallelism can solve arithmetic operation of any two natural numbers in polynomial time. Compared with the algorithm in P system, cell-like P system, proposed in literature [3], for the algorithms in this paper, the design of P system is easier, and the efficiency of the computation is improved. The two algorithms proposed in this paper, one based on generating string language, resulted as the number of strings exported, and the other based on generating datasets, resulted as the time lag between the first two exports. For addition, the time complexity of the two algorithms is linear, but the executive efficiency of SNPC system generating string language is higher. For multiplication, the time complexity of the algorithm in SNPC system generating string language is linear, but in SNPC system generating datasets is polynomial time, so the algorithm in SNPC system generating string language is easier and more efficient with high universality, which is suited for subtraction and division. For performing arithmetic operation in SNP system, how to import the natural numbers to the system is the question considered firstly, and the binary encoding is the frequently-used encoding method currently [4]. In this paper, the problem isn't considered when perform arithmetic operation in SNPC system which is the deficiencies of algorithm design and need to be improved. There are some other questions needed to be discussed, expect the design of reasonable hardware based on SNPC system, such as performing all possible arithmetic operation in SNPC system, more precise, when the objects aren't natural numbers, but negative numbers for example, how to design the SNPC system to solve the problem.

Membrane computing is considered to be the most promising in solving NP problems, of course, SNPC system also can used to solve NP problems. Literature [9] proposed solving TSP problem by membrane computing, using the algorithm MCTSP, and on this basis, we can use the SNPC system to find a more efficient algorithm for solving TSP. All of these are the future direction and points of the study. Obviously, due to the current of the analog implementation of the membrane system and in reality the corresponding technical support of realizing are too little, calculations based on membrane computing models and methods are more difficult to verify, which makes to solving the problem by membrane computing relatively vague. These have also become challenges and efforts for the study of SNPC system.

References:

[1] T. O., Ishdorj. "Membrane Computing, Neural Inspirations, Gene Assembly in Ciliates". Ph.D. Thesis, Universidad de Sevilla, 2006.
[2] Zhang Jian-ping, Liu Xi-yu. Optimization Model Based on Discrete Morse theory. System Engineering- Theory & Practice, vol.31, 2011.
[3] Ping Guo, Jing Chen. Arithmetic Operation in Membrane System. 2008 International Conference on BioMedical Engineering and Informatics.
[4] M.A. Gutirrez-Naranjo, A. Leporati. Performing arithmetic operations with spiking neural P systems[C]. Preproceedings of the 7th Brainstorming Week on Membrane Computing, 2009:181-198.
[5] Ionescu M, Paun Gh, Yokomori T. Spiking neural P systems. Fundamenta Informaticae, 2006, 71(2-3) : 279-308.
[6] Linqiang Pan, Gheorghe Paun. Spiking Neural P Systems with Anti-Spikes. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. IV (2009), No. 3, pp. 273-282.
[7] Forman Robin. Users Guide to Discrete Morse Theory. S'eminaire Lotharingien de Combinatoire, Article B48c,vol.48, pp.1-35, 2002.
[8] Dragos Sburlan. P Systems with Chained Rules. Twelfth International Conference on Membrane Computing ,pp.447-458.
[9] Chen Hai-zhu. Application Research on Membrane Computing[D]. Chongqing: Chongqing Uni- versity, 2011.