Reliable Arithmetic Circuit Design Inspired by SNP Systems

by

Pei An

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2013 by the
Graduate Supervisory Committee:

ARIZONA STATE UNIVERSITY

August 2013

ABSTRACT

Developing new non-traditional device models is gaining popularity as the silicon-based electrical device approaches its limitation when it scales down. Membrane systems, also called P systems, are a new class of biological computation model inspired by the way cells process chemical signals. Spiking Neural P systems (SNP systems), a certain kind of membrane systems, is inspired by the way the neurons in brain interact using electrical spikes. Compared to the traditional Boolean logic, SNP systems not only perform similar functions but also provide a more promising solution for reliable computation.

Two basic neuron types, Low Pass (LP) neurons and High Pass (HP) neurons, are introduced. These two basic types of neurons are capable to build an arbitrary SNP neuron. This leads to the conclusion that these two basic neuron types are Turing complete since SNP systems has been proved Turing complete. These two basic types of neurons are further used as the elements to construct general-purpose arithmetic circuits, such as adder, subtractor and comparator.

In this thesis, erroneous behaviors of neurons are discussed. Transmission error (spike loss) is proved to be equivalent to threshold error, which makes threshold error discussion more universal. To improve the reliability, a new structure called motif is proposed. Compared to Triple Modular Redundancy improvement, motif design presents its efficiency and effectiveness in both single neuron and arithmetic circuit analysis.

DRAM-based CMOS circuits are used to implement the two basic types of neurons. Functionality of basic type neurons is proved using the SPICE simulations. The motif improved adder and the comparator, as compared to conventional Boolean logic design, are much more reliable with lower leakage, and smaller silicon area. This leads to the conclusion that SNP system could provide a more promising solution for reliable computation than the conventional Boolean logic.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

As the silicon-based electrical devices scale down, it is more and more difficult to overcome the obstacles caused by the constraints of technology and manufacturing. In order to develop devices with higher performance, many new nontraditional device models are proposed. Membrane computing is one of the most popular research areas.

1.1 P SYSTEMS AND SNP SYSTEMS

Membrane system, also known as P system, is a computational model in the field of computer science that performs calculations using a biologically-inspired process. P system is defined as a distributed parallel computation model. Such a system is constructed by the objects in its membranes, specified evolution rules for objects, and the given input-output prescriptions. In P system, any object, alone or together with other objects, could perform many operations. They can evolve, be transformed into other objects, cross a membrane and dissolve the membrane where it is placed. The evolution rules are hierarchized by priority, the rule with the highest priority among the applicable rules is always the one actually applied. If there is only one object evolved, then the system is non-cooperative; if there are rules specifying the evolution of several objects at the same time, then the system is cooperative. Besides the previous cases, there is an intermediate case. This kind of cases involves certain objects (called catalysts) which do not evolve alone, but appear together with other objects in evolution rules and they are not modified by the use of the rules [1].

Membrane systems are sorted in three types: cell-like membrane system, tissue-like membrane system, neural-like membrane system. These three types are abstracted from corresponding biological systems. This thesis mainly studies SNP system (Spiking Neural P system), a certain kind of neural-like membrane system, which has a similar network structure like neural system.

As a branch of membrane computing, SNP systems define a set of rigorous rules that abstract the spiking behavior of biological neurons, place them in a directed graph, and process the input information. Compared to the complexity of membrane system, SNP system is much simpler and still follows the definition of membrane system: SNP system has only one object; the information passed from one neuron to another is in the form of electrical impulses, i.e., spikes, which are accumulated at the target neuron; the neuron which has only one membrane can hold an arbitrary number of spikes; each neuron operates the spikes in it under specified condition which are sorted in two types — firing and forgetting. The detailed mathematical definition of SNP system will be given in chapter two.

Firing rules: These rules allow a neuron to send a spike along its axon. This spike passes to all neurons connected by a synapse of the spiking neuron. Whether to fire or not is determined by checking if the total number of spikes collected in the neuron fulfills a firing rule equation. After firing, the neuron will consume some or all the spikes it has received in the previous step. As inspired from a biological neuron, when a neuron sends out spikes, it becomes inactive for a specified period of time, i.e., the refractory period, during which the neuron does not accept new inputs and cannot fire. When the neuron is firing, there is also a delay associated between the input and output spikes. In general, a set of firing rules, with different firing thresholds, can be defined for a SNP neuron.

Forgetting rules: Unlike the firing rules, the forgetting rules remove spikes from the neuron without firing any spikes. Forgetting rule and firing rule can’t be applied at the same time [2].

Figure 1 illustrates an example that abstracts a digital SNP neuron from a biological neuron: ‘a’ means a spike; ‘λ’ means no output spike. In this example, after the firing or the forgetting operation, all the spikes involved in the operation will be consumed and the neuron is going to be reset to the initial mode.

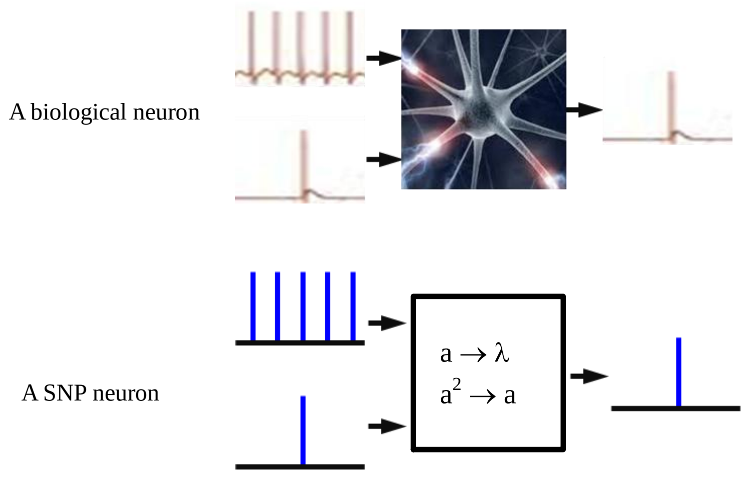


Figure 1. Example digital SNP neuron has a firing rule and a forgetting rule in it. It will fire a spike for every two input spikes and will not fire if there is only one input spike.

1.2 Spiking Neurons

As SNP system is abstracted from biological neurons, understanding how neurons operate will be very helpful in studying SNP system. A brief biological background of neurons is presented in this section.

The nervous system is unique in the vast complexity of thought processes and controls the actions it can perform. The central nervous system contains more than 100 billion neurons. Figure 2 shows a typical neuron of a type found in the brain motor cortex. Incoming signals enter this neuron through synapses. The synapse is the junction point from one neuron to the next. A special feature of most synapses is that the signal normally passes only in the forward direction, from the axon of a preceding neuron to dendrites on cell membranes of subsequent neurons. Conversely, the output signal travels along a single axon leaving the neuron. This axon has many separate branches to other parts of the nervous system or peripheral body.

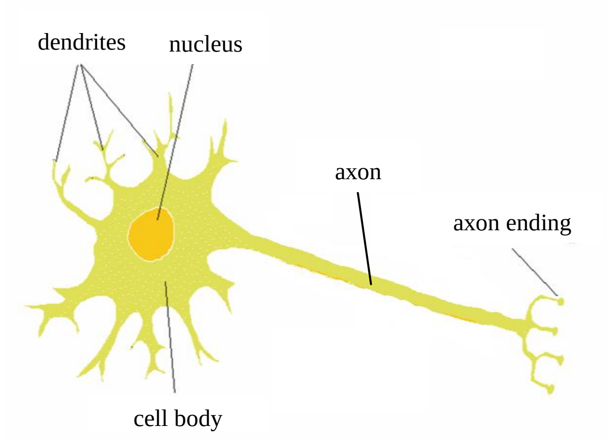


Figure 2. A neuron has dendrites, cell body and axon. A neuron can have many dendrites and axon endings but only one axon and one cell body [3].

The resting membrane potential of brain neurons when not transmitting nerve signals is about -70 millivolts. Nerve signals are transmitted by action potentials, which are rapid changes in the membrane potential that spread rapidly along the nerve fiber membrane. If any event causes enough initial rise in the membrane potential from -70 millivolts toward the zero level, the rising voltage itself causes many voltage-gated sodium channels to open. This allows rapid inflow of sodium ions, which causes a further rise in the membrane potential, thus opening more voltage-gated sodium channels and allowing more streaming of sodium ions to the interior of the fiber. This process is a positive-feedback cycle and once the feedback is strong enough it continues until all the voltage-gated sodium channels have become activated (opened). Then, within another fraction of a millisecond, the rising membrane potential causes the closure of the sodium channels and the opening of potassium channels and the action potential soon terminates. Then, rapid diffusion of potassium ions to the exterior re-establishes the normal negative resting membrane potential. The above process is shown in figure 3 and figure 4.

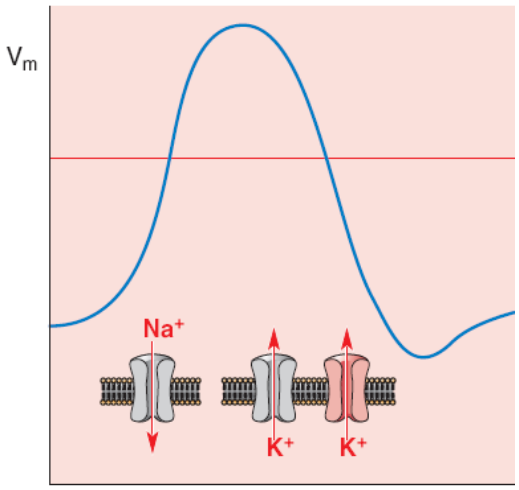


Figure 3. Neuron activation is controlled by the sodium channels and potassium channels [4]

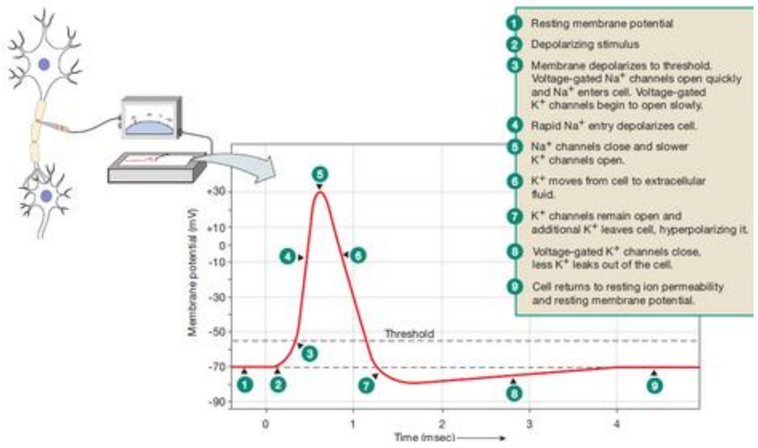


Figure 4. Neuron activation testing [5]

A new action potential cannot occur in an excitable fiber as long as the membrane is still depolarized from the preceding action potential. The reason is that shortly after the action potential is initiated, the sodium channels become inactivated and no amount of excitatory signal applied to these channels at this point will open the inactivation gates. The only condition that will allow them to reopen is for the membrane potential to return to or near the original resting membrane potential level. Then, within another small fraction of a second, the inactivation gates of the channels open and a new action potential can be initiated. The period during which a second action potential cannot be elicited, even with a strong stimulus, is called the absolute refractory period [6].

1.3 RELIABLE SNP SYSTEMS

The reliability of CMOS circuits has become one of the top concerns of future IC design, especially impacted by process variations and device unreliability. However, compared with biological neurons, silicon transistors and circuits are still quite reliable. At the 12nm node, the failure rate of a SRAM cell is expected to be less than 0.03% [7], on the other hand, for a spiking neuron, the coherence between the stimulus and the output signals is only ~70% [8][9]. Even with such a high failure rate, the brain is still robust enough to perform neural function and computation correctly, which raise a question on how the neural system overcomes this reliability challenge.

Many works have been inspired by this question. One of the well-known examples is neural networks, which focuses on how to adaptively tune the synapse, i.e., the interconnection between neurons, to achieve the robustness and the learning capability [10] [11]. However, this approach only reflects Pavlov's conditioned reflex experiment which mainly focuses on the synapse function of the receiver. Besides the biological constraints of this approach, the difficulty in hardware implementation also limits the applications, even with the help of emerging memory devices [12][13][14].

Instead of the synapse, SNP systems pay more attention to the neuron, which is the core component in neural computation. SNP system has been proved to be Turing complete [15], and capable of solving numerical NP-complete problems in polynomial time [16]. Indeed, as a computing device, the spiking neuron endows substantially larger computation power and better reliability over the conventional Boolean logic [17][18].

1.4 Specific Contributions of This Thesis

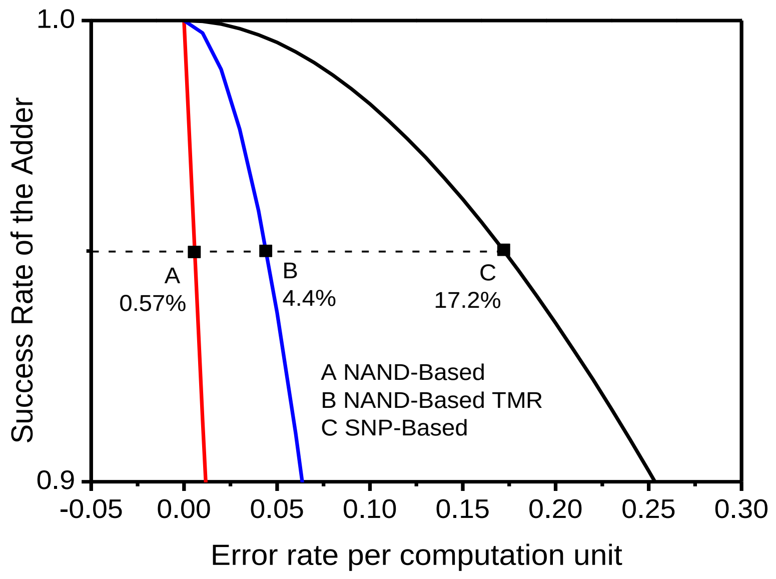


Figure 5. The SNP-based design has a significant advantage in circuit reliability over Boolean logic

This thesis attempts to transfer the theoretical SNP model into the practical circuit design, with the emphasis on circuit reliability management from unreliable devices. Figure 5 presents a 1-bit full adder as an example, applying different reliability improvement (chapter 3 and 4). NAND-based adder consists of 36 transistors, Triple

Modular Redundancy (TMR) improved NAND-based adder has more than 108 transisters, whereas the SNP-based adder has 83 transistors. Assuming the probabilities of a 1-bit error happening in a neuron and in a NAND gate are the same, the SNP-based design is clearly superior in circuit-level reliability. At the 95% success rate of the adder function, the SNP-based design tolerates 17.2% error rate per neuron, as compared to 0.57% in the single module implementation or 4.4% error rate per NAND gate in the case of TMR.

With this benefit, this work represents the first step toward reliable logic circuits design that is inspired by spiking neurons. The contents of this thesis include:

Basic neurons: Two types of spiking neurons are proved to be basic neurons which can build up a neuron with arbitrary rules in it. (Chapter 2)

SNP-based arithmetic circuits: Some basic arithmetic operations, addition and comparison, are synthesized for design evaluation. (Chapter 3)

Design of SNP motifs for reliability: Error model are proposed after discussing erroneous neurons. To deal with the error in a single neuron, appropriate redundancy is necessary. Compared to simply adding more copies of neurons (e.g., TMR), the proposed motif design (i.e., clusters of neurons) is able to utilize the unique operation of spiking neurons and to achieve high reliability. (Chapter 4)

CMOS implementation of neurons: Using DRAM-type circuits, the design realizes the function of neuron, without any Boolean logic. Design simplicity offers lower leakage and smaller area. The sacrifice of reliability at the neuron level is well recovered at the circuit level, through the SNP network with the help of motif design. The design is evaluated at 45nm in the presence of statistical variations. (Chapter 5)

CHAPTER 2

PRINCIPLES OF SNP SYSTEMS

2.1 Introduction to SNP Systems

Spiking neural P systems are inspired by membrane systems and based on the neurophysiological behavior of neurons [2]. In SNP systems, the processing elements are neurons, which are connected by a directed graph—the synapse graph. The SNP systems work only with spikes and all the spikes it deals with are identical. This section explains the basic rules of SNP neurons in a mathematical way.

For a Spiking Neural P system of degree $m \geq 1$, in the form

$$\Pi = (O, \sigma_1, \dots, \sigma_m, syn, i_0)$$

Where:

1. O={a} is a singleton alphabet and a represents a spike.
2. $\sigma_1, \dots, \sigma_m$ are neurons of the form

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m$$

Where:

a. $n_i \geq 0$, n_i is the initial number of spikes contained in the cell;

b. R_i is a finite set of rules of the following two forms:

(1) $E/a^r \rightarrow a; t$, where E is a regular expression over O, $r \geq 1$ and $t \geq 0$,;

(2) $a^s \rightarrow \lambda$, for $s \geq 1$, $a^s \notin L(E)$ for any rule $E/a^r \rightarrow a; t$ from R_i.

3. syn belongs to $\{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$;

4. $i_0 \in \{1, 2, \ldots, m\}$ represents the output neuron.

The rules of type (1) are firing rules, the contents of the neuron are described by the regular expression E, r spikes are consumed, the neuron is fired, and it produces a spike which will be sent to other neurons after t times units.

Neurons have two important actions which can happen in one step: getting fired and spiking.

A neuron gets fired when applying a rule $E/a^r \rightarrow a;\ t$, and this is true only if the neuron contains n spikes such that $a^n \in L(E)$ and $n \geq r$. This means that the regular expression E "covers" exactly the contents of the neuron.

For the spiking, the use of a rule $E/a^r \rightarrow a;\ t$ in a step q means firing in step q and spiking in step q + t. That is, if $t = 0$, then the spike is produced immediately, in the same step when the rule is used. If $t = 1$, the spike will leave the neuron in the next step, and so on.

The rules of type (2) are forgetting rules: s spikes are simply removed from the neuron when applying $a^s \rightarrow \lambda$. Like in the case of spiking rules, the left side of a forgetting rule must "cover" the contents of the neuron, i.e. $a^s \rightarrow \lambda$ is applied only if the neuron contains exactly s spikes. [2]

2.2 Basic type of neurons

From the previous sections, we have known the basic information of SNP systems. Since there's no restriction on the expression E, the combination of different rules in a neuron is infinite, which would complicate the fabrication of neuron circuits design.

Inspired by the fact that any Boolean function could be implemented by NAND gates, a hypothesis is proposed that there are some basic neurons which could implement a neuron with arbitrary rules in it.

Since neurons have two actions—getting fired and spiking, we are going to begin with finding neurons that could implement these two actions separately.

The first thing is to identify the number of input spikes. The number of input spikes could be expressed in binary notation. Inspired by the 1-bit full-adder in Boolean logic, two kinds of neurons called odd type and even type are presented in figure 6. The odd type performs the function of sum of Boolean 1-bit full-adder while the even type performs the function of carryout of 1-bit full-adder.

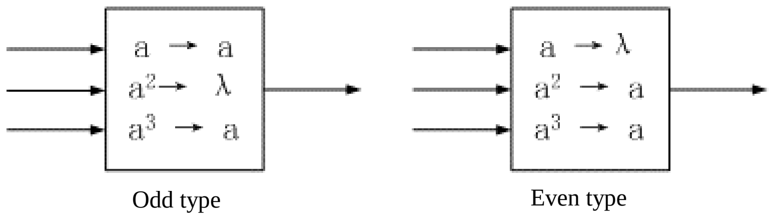


Figure 6. Odd-even type neurons can build up 1-bit adder. Odd type performs the sum function and the even type performs the carryout function

With one odd type neuron and one even type neuron, a 1-bit spikes converter which convert the number of input spikes into binary expression can be built (shown in figure 7): O corresponds to S_{out} in 1-bit full-adder and E corresponds to C_{out}. The number of input spikes will be expressed in binary code EO.

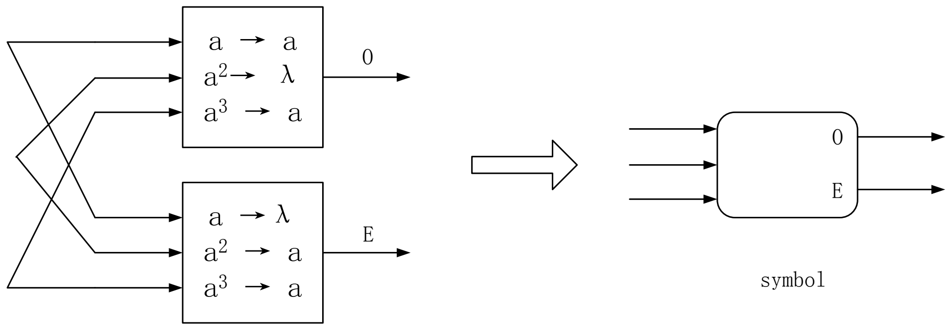


Figure 7. 1-bit spikes adder design and its symbol

This design can be expanded to sum up more input pulses. Figure 8 shows a design which can sum up 12 inputs and express the number of input spikes in a 4-bit binary code $a_0a_1a_2a_3$.These designs are named as converters since they can convert the number of input spikes into binary expression.

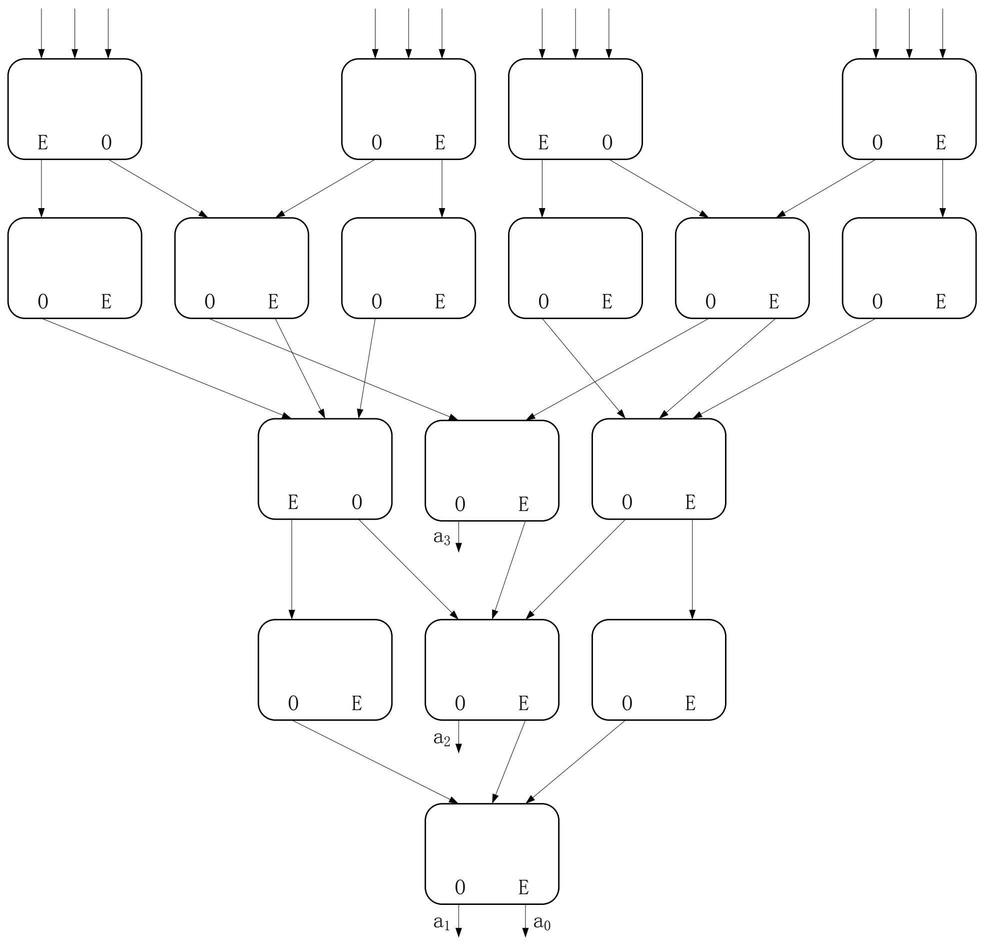


Figure 8. Multi-bit adder is the expansion of the 1-bit adder. This one can sum up at most 12 input spikes and convert it into binary code $a_0a_1a_2a_3$

However, the odd type neuron has two thresholds, which will be a problem in the future hardware design and implementation. In order to avoid this problem, some new basic type neurons that have only one threshold are needed. Moreover, the new basic neurons should also be capable of constructing the odd-even type to sum up input spikes.

Since the neurons have only one threshold, the number of inputs of the new basic type needs not to be set, the new basic neurons could have arbitrary number of inputs. And this is very reasonable because neurons in brain always have many inputs.

Firstly, the even type neuron fulfills the requirement of one threshold. We rename it High Pass neuron with threshold 2 (in short: HP neuron). Following the one threshold requirement, the complementary design of HP neuron is named as Low Pass neuron with threshold 2 (in short: LP neuron). HP and LP neurons are shown in figure 9. For simplicity, HP and LP neurons will be represented by their shapes shown in figure 9 without rules written in it for rest of this thesis.

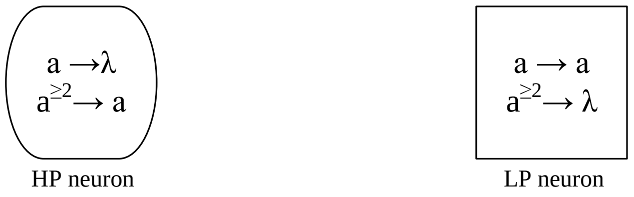


Figure 9. HP neuron and LP neuron are the basic neurons. HP neuron will fire a spike out if it receives more than two spikes and otherwise will forget input spikes. LP neuron works in the opposite way.

Figure 10 presents the odd-even type neuron constructed by HP and LP neurons. Verification on the function of designed odd neurons is as follows:

1. If there’s only one spike at the inputs, this spike will be passed to the final output.
2. If there are two spikes at the inputs, there will be two kinds of cases: (a) input 1 and input 2 have spikes or (b) input 3 and input 1 or input 2 have spikes. For case a, neuron B will give out nothing and neuron A will see nothing from the inputs, thus gives out nothing. For case b, both neuron B and C will send a spike to neuron A and neuron A will give out nothing at the output.

3. If there are three spikes at the inputs, neuron B will give out nothing and neuron C will pass the input spike to neuron A, and neuron A will receive only one spike at the inputs and thus fire a spike out.

From the above analysis, the design is proved to be valid.

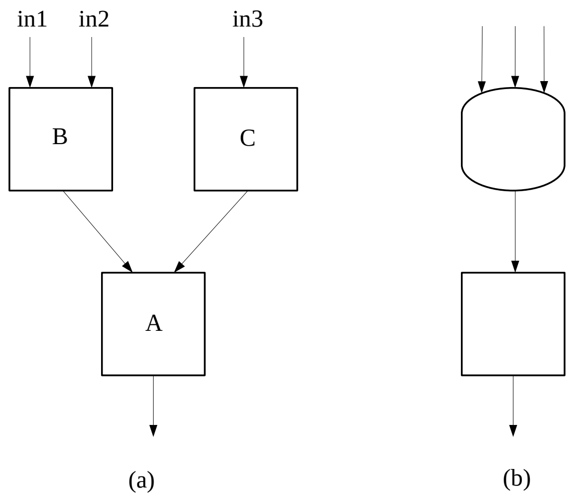


Figure 10. Odd-even neuron can be implemented by HP and LP neurons. (a) odd type, (b) even type

A one input LP neuron could work as another important unit—the delay unit. This delay unit with one clock delay is designed to synchronize the circuits. Neuron C in the LP neuron based odd-type neuron design is an example of delay unit.

From the above, we can conclude that the converter, which can sum up arbitrary number of input spikes and expressed them in binary code, could be constructed by LP and HP neurons.

For the rule part, some LP and HP neurons will be used to build up a judgment circuit which decides whether to fire or not based on the number of input spikes expressed in binary code generated by converter.

A special kind of rules called “do-nothing” should be considered. Unlike the firing and forgetting rules, this kind of rules makes the neurons do nothing, i.e., the input spikes will be remembered in the neuron with no spike output. This kind of rules even won’t be written in the neurons as they are not real rules.

Suppose a binary number $a_0a_1a_2a_3a_4a_5$ is transferred from the converter. For a firing rule $a^n \to a$, a set of HP and LP neurons are supposed to check the equation $n = a_0a_1a_2a_3a_4a_5$, if the equation was true, a spike should be sent to final output.

Similarly, for a forgetting rule $a^n \to \lambda$, another set of HP and LP neurons are supposed to check the equation $n = a_0a_1a_2a_3a_4a_5$, if the equation was true, a spike should be generated to make sure no firing rules would be activated. A detailed example will be given later.

For the do-nothing rule a^n, if the equation $n = a_0a_1a_2a_3a_4a_5$ was true, a spike will be generated and copied to n inputs to participate the next step calculation.

2.3 An Example of Building a Complex neuron

Figure 11 presents a neuron which will be implemented by HP and LP neurons. The neuron has both firing and forgetting rules in it, along with a do-nothing rule a^2.

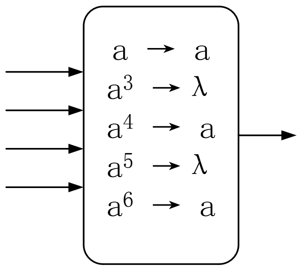


Figure 11. A representative neuron will be implemented by HP and LP neurons. This neuron has firing rules, forgetting rules and a special rule a^2 in it.

Figure 12 presents the converter design and its symbol which will be used in later. The converter generates a binary number $a_0a_1a_2$, a_0 is most significant digit and a_2 is the least significant digit.

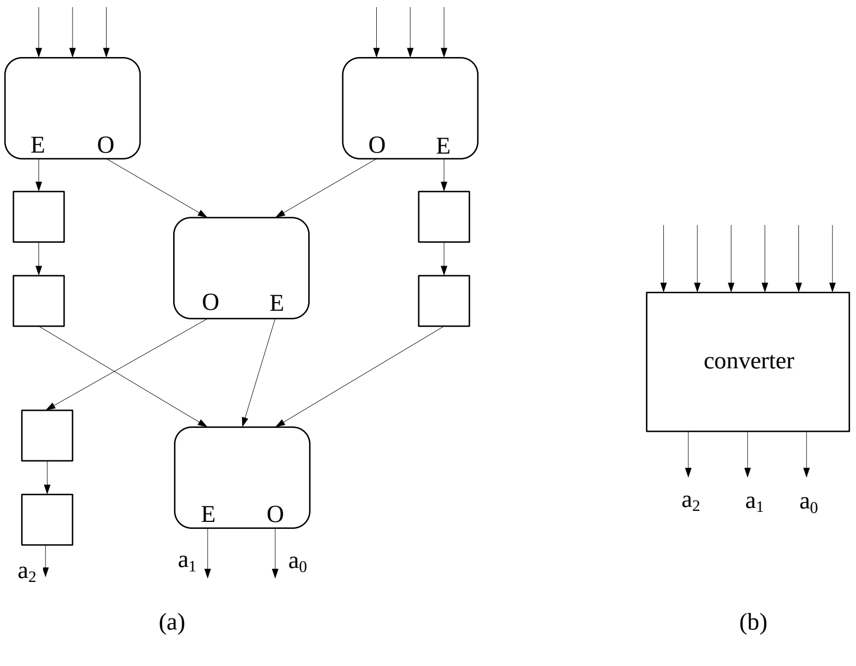


Figure 12. Converter design and symbol

Figure 13 illustrates the ruler design. This neuron has 5 rules in it: 1) $a \to a$; 2) $a^3 \to \lambda$; 3) $a^4 \to a$; 4) $a^5 \to \lambda$; 5) $a^6 \to a$. Since neurons would only be activated by spikes, some rules might be activated unexpectedly, i.e. rule 1 will be activated when applying rule 4. Thus rule 1 should be deactivated by rule 2 and rule 4; rule 3 should be deactivated by rule 4 and rule 5 and the do-nothing rule a^2 should be deactivated by rule 2 and rule 5.

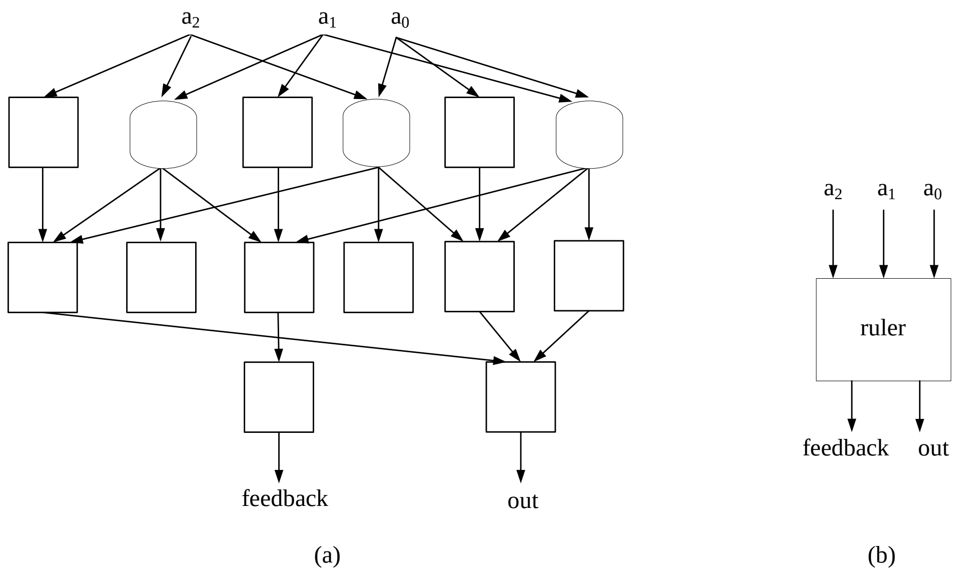


Figure 13. Ruler design and its symbol

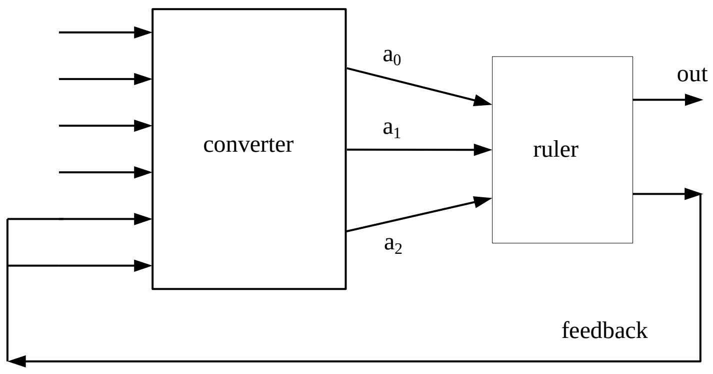


Figure 14. Final implementation of the proposed representative neuron

Figure 14 presents the final design of the target neuron, which leads to the conclusion that an arbitrary neuron could be implemented only with HP and LP neurons.

Since SNP systems have already been proved to be Turing complete [2] [15], the same conclusions could be expanded to the HP and LP neurons.

CHAPTER 3

DESIGN OF SNP-BASED ARITHMETIC CIRCUIT

The spiking neuron model has been proved having much more computational capacities in Boolean function processing than other traditional neuron models, such as threshold gates and sigma-pi units [17]. There have been several attempts in the synthesis of arithmetic operations with SNP neurons [19][20]. This chapter proposes designs of several arithmetic functions using only the LP and HP neurons.

3.1 Prerequisite of SNP Circuit Design

In order to perform arithmetic operation, it is necessary to determine the numbers to be computed in the system. A lot of works have been done on applying SNP systems in Boolean domain, dealing with binary codes [19]. However, the binary encoding method might not be suitable for SNP systems since it was designed appropriately for CMOS logic circuits. Instead of binary codes, the unary codes seem to be more appropriate for SNP systems. In most researches on SNP systems, the encoding mechanism is designed to be either the time interval between two spikes, or the number of continuous spikes. This mechanism accords with the fact that the time when neurons receive and fire out pulses is of great importance in encoding information in human brain [20].

Figure 15 illustrates these two representation methods, which could be easily converted into each other using the circuit shown in figure 16[21].

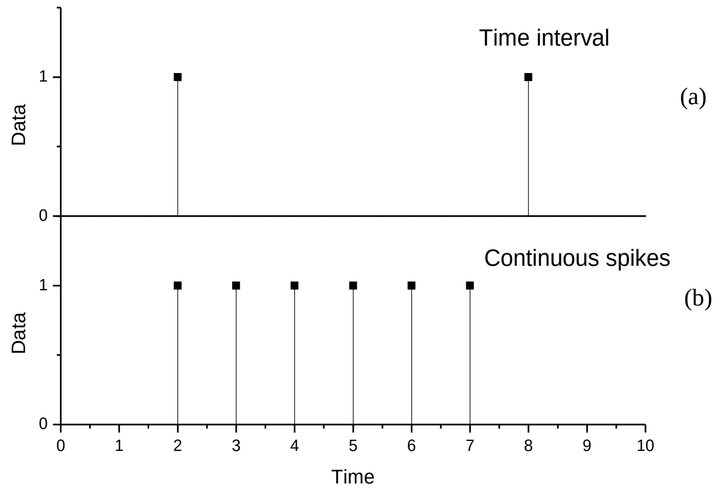


Figure 15. Six expressed in two methods. (a) Six expressed in time interval (b) Six expressed in continuous spikes

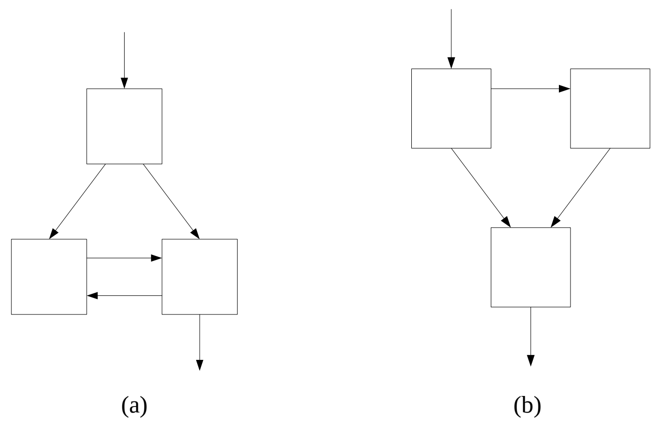


Figure 16. Two circuits that transform data between different representation methods: (a) Convert data represented in time interval into continuous spikes, (b) Convert data represented in continuous spikes into time interval [21].

In [20], the time interval represented data was computed after being transferred into continuous spikes and the result represented in continuous spikes was converted into the time interval. Since all the data being calculated was represented in continuous spikes, this method will be used in this thesis for simplicity.

Since the SNP model operates quite differently from the Boolean gates, the network of SNP systems that performs a certain arithmetic function will not be the same as the one of Boolean logic. Thus the SNP network design plays an essential role at the preliminary stage of SNP design for arithmetic functions.

3.2 Adder Design

Adder is the core component of arithmetic logic units. Indeed, in today's digital computer, most of the arithmetic functions, such as multiplication, subtraction, and division, are constructed from a binary adder with other smaller circuits (e.g., the inverter or the shifter).

One-bit full adder in Boolean logic has two outputs, one for carryout and one for the sum. Although our data representation is different from the binary method, the SNP based adder still needs to have these two outputs.

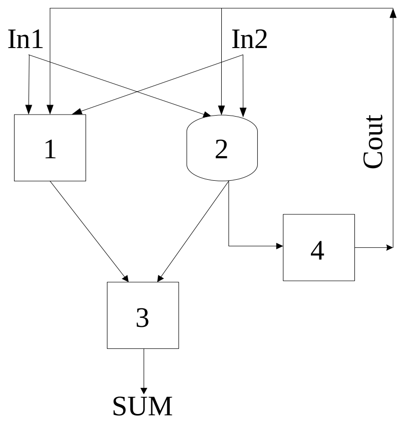


Figure 17. SNP-based adder design

Figure 17 presents the design of a SNP based adder: In1 and In2 are the primary input data; neuron 1 and neuron 2 are input neurons; neuron 3 is the output neuron; neuron4 is not only a delay neuron which synchronizes the calculation but also a feedback path that sends carryout back to the primary input neurons.

This design has a restriction that in1 or in2 should be one at most. This restriction is caused by the scale of the feedback path. If a stronger computing capability is needed, a larger and more complex feedback neuron circuit should be designed. Two assumptions are applied here and later in this thesis: input data are synchronized; all single neurons have the same signal delay from the input to the output.

Figure 18 shows the timing diagram of this adder design. The delay from the primary input to the final output equals 2 which is the number of operation levels in the SNP network.

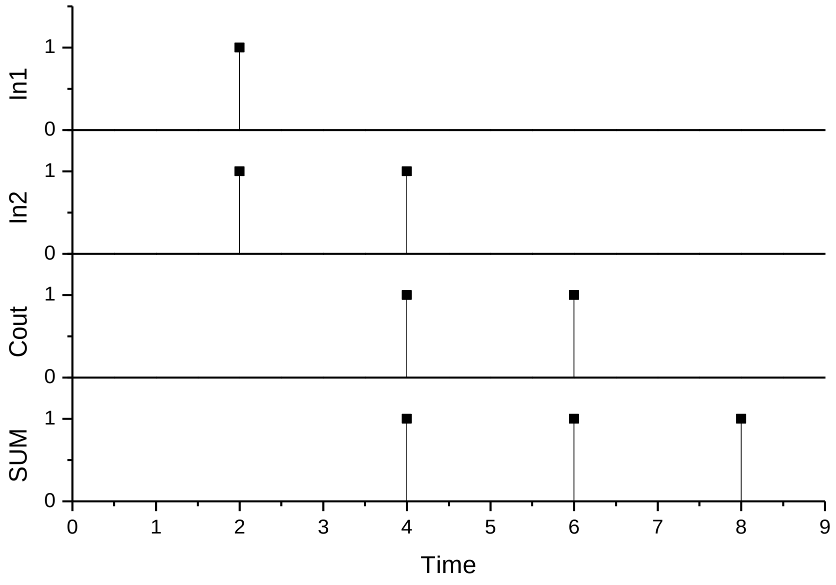


Figure 18. Timing diagram of adder

3.3 Subtractor Design

Subtraction is another fundamental arithmetic function. Unlike Boolean logic subtractor, the SNP based subtractor is quite simple. Actually, a single LP neuron can perform the subtraction.

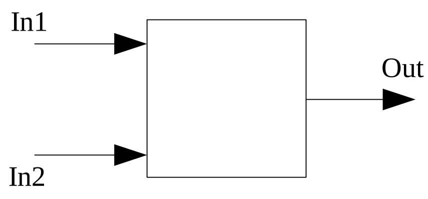


Figure 19. SNP based subtractor design

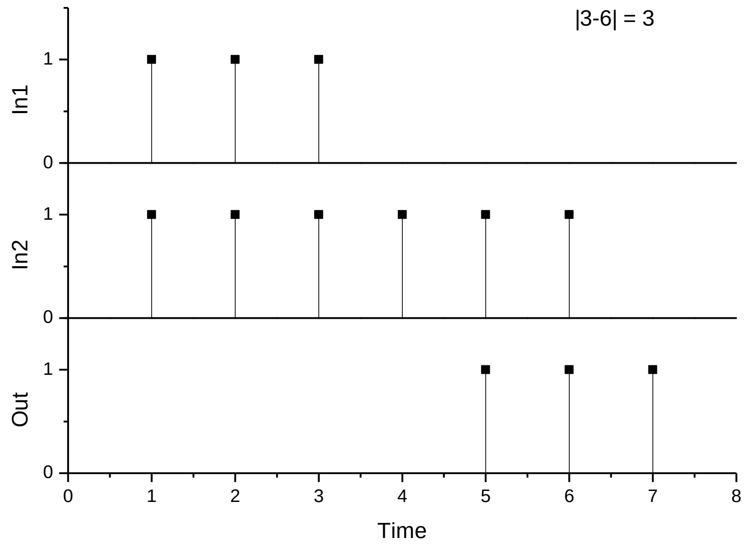


Figure 20. Timing diagram of subtractor

From section 3.2 and 3.3, we can find that SNP neurons circuit design differs from Boolean circuit design a lot. The SNP adder seems to be more complex than 1-bit full adder in Boolean logic while the SNP subtractor is much simpler than Boolean logic subtractor. This is because of the different rules applied to SNP systems and Boolean logic. What’s more, the disparate number representation method also leads the SNP neuron circuit design to a new way.

3.4 Comparator Design

Another representative arithmetic circuit is the comparator which is widely used in many areas, especially in image processing applications. The comparator calculates the absolute difference between two numbers and compares the value with a given threshold.

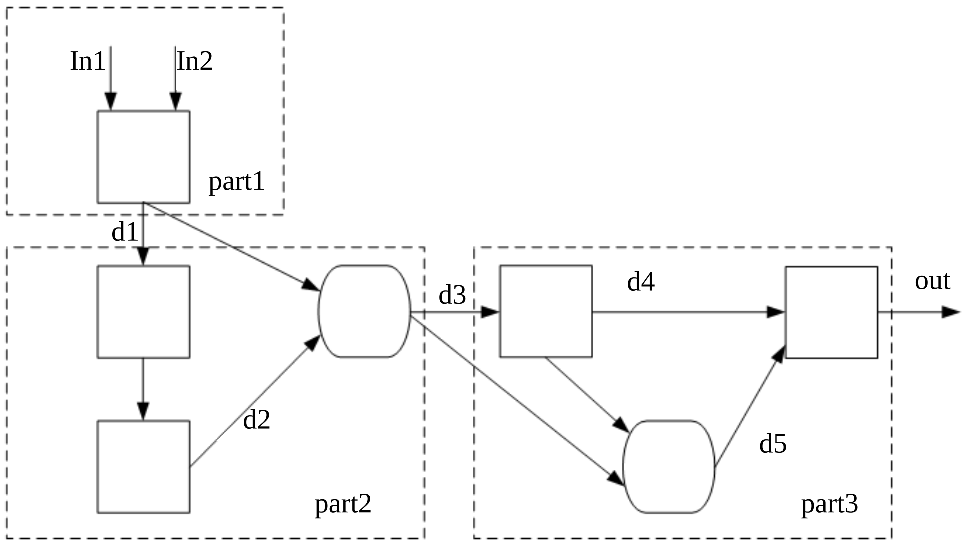


Figure 21. SNP based comparator design

Compared to conventional Boolean logic comparator, figure 21 presents a new design that is fully based on LP and HP neurons. It will output one spike if the difference between its two input spike sequences is greater than or equal to 3.

This SNP design contains three parts:

- Part1: Subtraction. This part evaluates the absolute value of the subtraction between two inputs. The function of this part is：

$$\mathrm{o_{p1}} = |\mathrm{in1} - \mathrm{in2}|$$

- Part2: Comparison. This part compares the subtraction value with a threshold 3. The function of this part is :

$$\mathrm{o_{p2}} = \mathrm{o_{p1}} - \mathrm{threshold} + 1$$

  The threshold value is determined by the number of levels in the comparison network. Thus, if another threshold value is needed, the chain of LP neurons in P2 should be proportionally adjusted. The number of LP neurons in the chain is $\mathrm{threshold} - 1$, so if the threshold is one, this part could be removed.

- Part3: Conversion. This part converts the input spike sequence into one spike if there’s a spike sequence at the input. The function of this part is:

$$\begin{cases} \mathrm{o_{p3}} = 1, \mathrm{if}\ \mathrm{o_{p2}} \geq 1 \\ \mathrm{o_{p3}} = 0, \mathrm{if}\ \mathrm{o_{p2}} = 0 \end{cases}$$

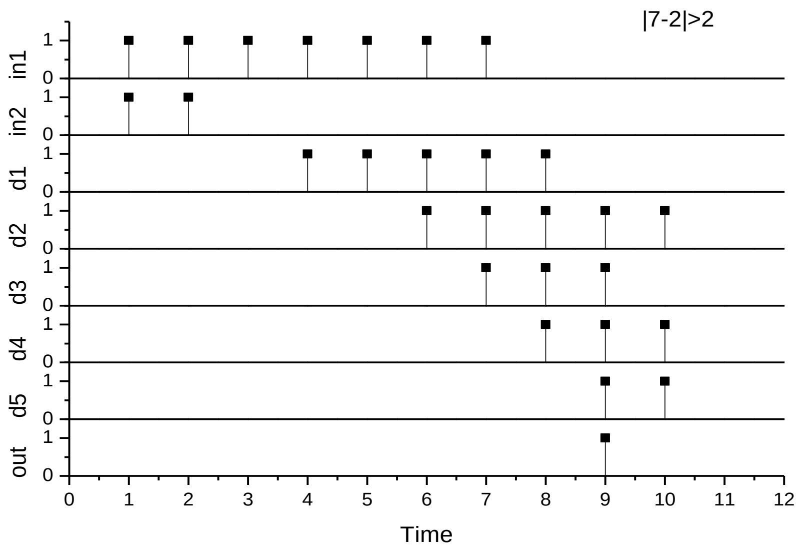


Figure 22. Timing diagram of SNP based comparator

Unlike the design of the previous SNP adder, in the design of a SNP comparator, the maximum length of input spike trains is actually not limited. This is because the function of a SNP comparator is decreasing the length of the input spike sequence while the adder is combining and increasing the length of the input spike sequence. As a result, the comparator design needs no feedback path in the network while the adder design does.

CHAPTER 4

RELIABLABLE DESIGN OF SNP SYSTEM

In this chapter, erroneous neuron is analyzed and error models are proposed; a motif design for neurons to improve reliability is applied in three arithmetic circuit designed in chapter 3.

4.1 ERROR MODEL DISCUSSION

In order to increase reliability, the SNP error types should be analyzed and SNP error models should be built.

The definition of SNP systems is clearly expressed in Chapter2. The firing rule is:

$E/a^r \rightarrow a;\ t$, where E is a regular expression over O, $r \geq 1$ and $t \geq 0$;

We can draw a conclusion that SNP neurons could only fire one spike at one time, which means SNP neurons will only receive no more than one spike at each input each time. Then the assumption that the most possible error on the arriving spike is missing seems reasonable. This spike missing problem could probably be caused by one spike arriving at the target neuron after the neuron evaluates its inputs or the synapse fails to work.

This assumption fits the biological background very well: After firing a spike, the excitability of a neuron would reduce to zero for a certain period and this hyperpolarized neuron could not be inspired during this period which is called absolute refractory period.

This means that the neurons could not generate two spikes in serial in a certain period which corresponds to one clock in SNP systems.

Since we assume the error rate would not be very high, the probability of two or more input spikes missing at the same time would be very small, which allows us to ignore this situation in this thesis.

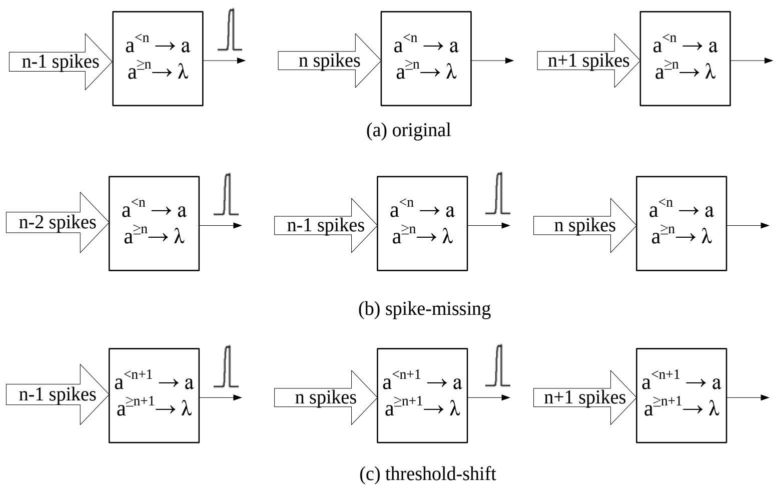


Figure 23. Error discussion on LP neurons with threshold n:
(a) LP neuron with threshold n having n-1,n,n+1 spikes input
(b) Same neuron having one input spike missing
(c) A neuron with threshold increased by one

Figure 23(a) shows a LP neuron with threshold n, for n input spikes case, the neuron will fire a spike; for n+1 input spikes case and n+2 input spikes case, the neuron will forget all the input spikes.

Figure 23(b) shows the same LP neuron with one input spike missing, n input spikes case and n+1 input spikes case become n-1 input spikes case and n input spikes case which would inspire the neuron to fire a spike; n+2 input spikes case becomes n+1 input spikes case while still triggering the forgetting rule.

Figure 23(c) shows a LP neuron with threshold n+1, for n input spikes case and n+1 input spikes case, the neuron will fire a spike; for n+2 input spikes case, the neuron will forget all the input spikes.

From the above, we can conclude that for LP neurons, one spike missing is equivalent to the threshold increasing by one.

There is one conflict which needs to be clarified. When the number of input spikes is one, for the threshold increasing erroneous neuron, it will fire, but for the spike-missing erroneous neuron, the neuron can’t receive the spike and thus would not fire. In this case, the threshold-shift error doesn’t match the spike-missing error. This conclusion is incorrect. Let’s recall from the reason of one spike missing. This error happens because one spike did not reach the target neuron before the neuron evaluates the input spikes. However, a neuron starts evaluating its input pulses after it receives its first spike which means spike-missing error would not occur to LP neurons if there was only one spike at the input.

As a result, the threshold-shift error is equivalent to the spike-missing error for LP neurons.

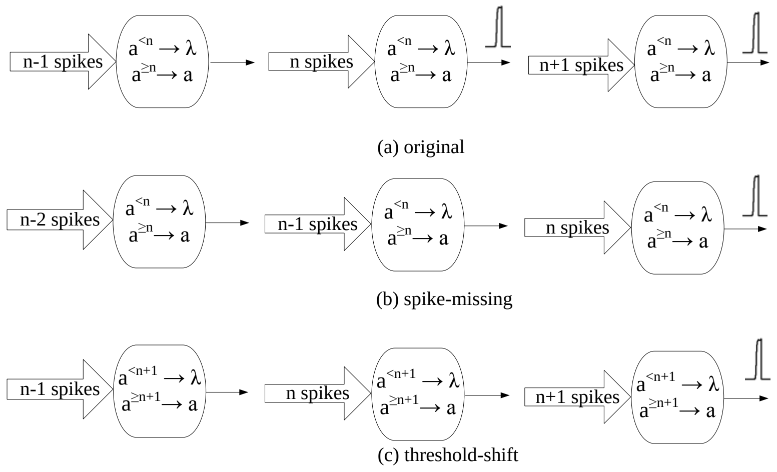


Figure 24. Error discussion on HP neurons with threshold n:

(a) HP neuron with threshold n having n-1,n,n+1 spikes input
(b) Same neuron with one input spike missing
(c) A neuron with threshold increased by one

Figure 24(a) shows a HP neuron with threshold n, for n-1 input spikes case, the neuron will forget the input spikes; for n input spikes case and n+1 input spikes case, the neuron will fire a spike.

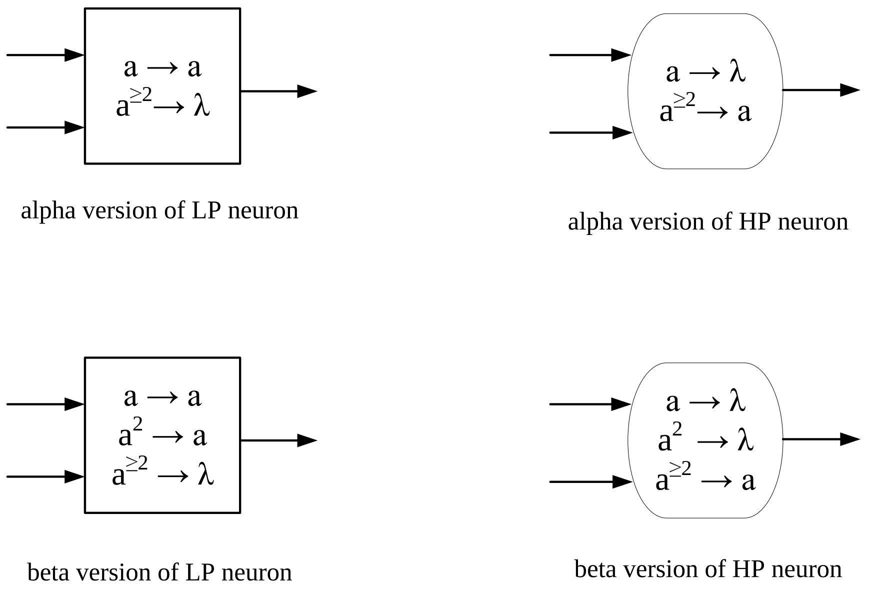
Figure 24(b) shows the same HP neuron with one input spike missing, n-1 input spikes case and n input spikes case become n-2 input spikes case and n -1 input spikes case which would trigger the forgetting rule; n+1 input spikes case becomes n input spikes case while still inspiring the neuron to fire a spike.

Figure 24(c) shows a HP neuron with threshold n+1, for n-1 input spikes case and n input spikes case, the neuron will forget its input spikes; for n+1 input spikes case, the neuron will fire a spike.

If n=1, the threshold increasing error still equals the one spike missing error.

From the above, we can conclude that the one spike missing error is equivalent to the threshold increasing by one error for both LP and HP neurons. Since the threshold-shift error is simpler on analyzing and modeling, it is an appropriate model to analyze the reliability in this thesis.

For simplicity, the original basic SNP neurons are named as alpha version, the threshold-shift erroneous SNP neurons is beta version. Figure 25 presents the alpha version and beta version of LP and HP neurons with threshold one, which will be used in this thesis.



$a \rightarrow a$
$a^{\geq 2} \rightarrow \lambda$
alpha version of LP neuron
$a \rightarrow \lambda$
$a^{\geq 2} \rightarrow a$
alpha version of HP neuron
$a \rightarrow a$
$a^{2} \rightarrow a$
$a^{\geq 2} \rightarrow \lambda$
beta version of LP neuron
$a \rightarrow \lambda$
$a^{2} \rightarrow \lambda$
$a^{\geq 2} \rightarrow a$
beta version of HP neuron

Figure 25. Alpha and beta version of LP and HP neurons. Alpha version represents the designed neurons, beta version represents the erroneous neurons.

4.2 Error Tolerated Improvement of A Single Neuron

In order to enhance the error tolerance of a SNP circuit design, our approach starts from increasing the reliability of a single LP or HP neuron. For this purpose, a traditional solution is to combine more identical units with a majority vote unit. In this way, the function of the unit will be correct only if the majority of these multiple copies work correctly. Besides, the majority vote unit should also be correct. Figure 26 presents an example of TMR (Triple Modular Redundancy) design for a LP neuron. Three copies of the LP neuron work in parallel and the HP neuron serves as the majority vote unit in this case.

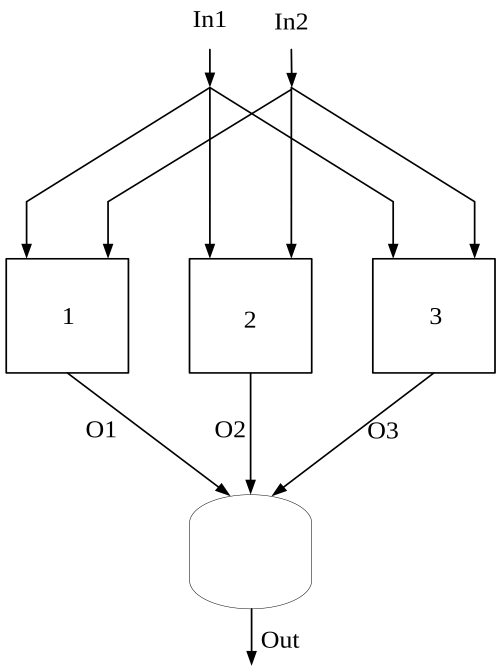


Figure 26. TMR improved LP neuron

Table 1 presents the outputs of all neurons in this circuit corresponding to all kinds of possible inputs of the circuit, 1 represents a spike at the input, and 0 represents nothing.

Table 1. Signal analysis in TMR improved LP neuron

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0

If one of the LP neurons (suppose neuron 3) changes from alpha version to beta version:

Table 2. Signal analysis in erroneous TMR improved LP neuron (1 erroneous LP)

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	0	1	0

If the HP neuron changes from alpha version to beta version:

Table 3. Signal analysis in erroneous TMR improved LP neuron(1 erroneous HP)

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0

If two of the LP neurons (suppose neuron 2 and 3) change from alpha version to beta version:

Table 4. Signal analysis in erroneous TMR improved LP neuron (2 erroneous LP)

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1

Table 1, 2 and 3 have the same functions while table 4 has a different function on case In1=In2=1. Thus, this design could tolerate at most one beta version error, no matter error happens to LP neuron or HP neuron.

Suppose the error rates $\mathrm{e_n}$ of the neurons in a design are identical, and the errors among the neurons are uncorrelated. (This assumption will be used later in this thesis.) The reliability of this TMR circuit is: $(1-\mathrm{e_n})^4 + 4\mathrm{e_n}(1-\mathrm{e_n})^3$

Although the TMR solution is widely used in many designs, it does have its defects. The cost of TMR is very high while the improvement of reliability is relatively low. In a word, the efficiency of its reliability improvement is low.

Compared to TMR, a better solution which is more efficient in reliability improvement is needed. This solution should be able to make use of the unique properties of the spiking neuron, as well as the error model.

This new design starts from analysis on the threshold-shift error model. When a basic neuron falls into beta version, the threshold of the neuron changes from two to three. Errors only happen to the two input spikes case. If the neuron has one input spike or three input spikes, the output will still be correct. Same conclusion could be made from Table 4. Thus avoiding two input spikes cases is a good choice on reliability improvement.

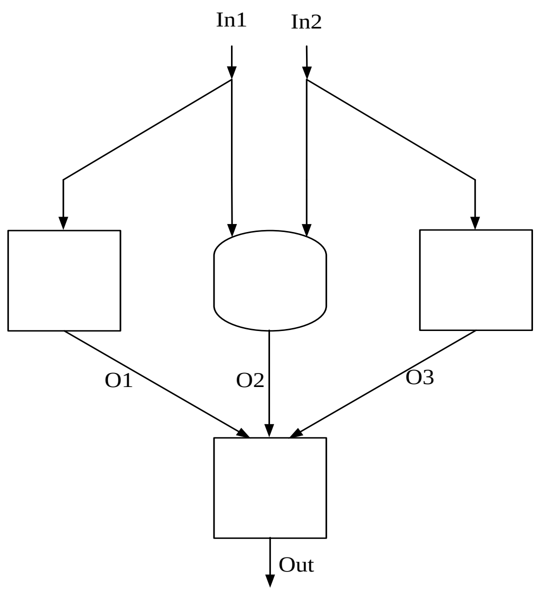


Figure 27. Motif improved LP neuron

Figure 27 presents a circuit called LP motif, which is designed to tolerate the threshold-shift error.

Table 5 presents the outputs of all neurons in this circuit corresponding to all kinds of possible inputs of the circuit.

Table 5. Signal analysis in Motif improved LP neuron

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	0	0	1	1
1	0	1	0	0	1
1	1	1	1	1	0

If the output LP neuron changes from alpha version to beta version:

Table 6. Signal analysis in erroneous Motif improved LP neuron (erroneous output LP)

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	0	0	1	1
1	0	1	0	0	1
1	1	1	1	1	0

If the HP neuron changes from alpha version to beta version:

Table 7. Signal analysis in erroneous Motif improved LP neuron (erroneous input HP)

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	0	0	1	1
1	0	1	0	0	1
1	1	1	1	0	0

If both the output LP neuron and the HP neuron change from alpha version to beta version:

Table 8. Signal analysis in erroneous Motif improved LP neuron (erroneous input HP and output LP)

In1	In2	O1	O2	O3	Out
0	0	0	0	0	0
0	1	0	0	1	1
1	0	1	0	0	1
1	1	1	0	1	1

From section 4.1, we know that one input LP neurons would not have threshold-shift error. Thus we can conclude that this motif design would fail only if both the HP neuron and output LP neuron failed. Therefore the reliability of this circuit is: $1 - \mathrm{e_n}^2$. This design uses the same number of neurons as the TMR design but in a different network.

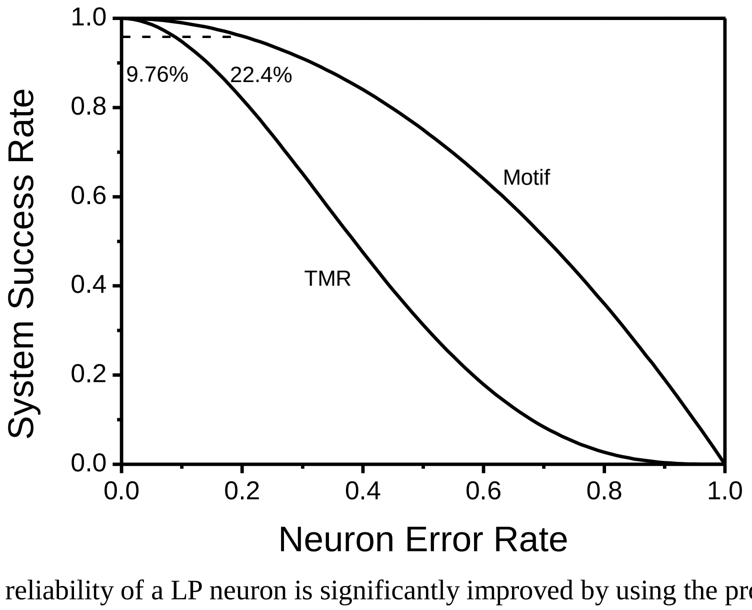


Figure 28. The reliability of a LP neuron is significantly improved by using the proposed motif, with a similar area cost of TMR

Figure 28 illustrates the success rate of the motif compared to the TMR design. At the 95% success rate, the error rate TMR could tolerate is 9.76% while that the motif could tolerate is 22.4%, which is much more effective than TMR.

In order to further enhance the reliability, TMR design usually adds TMR to each unit in the original design while the motif only needs to insert more HP neurons in parallel with the one in figure 27. Figure 29 illustrates a 2nd order improved LP motif with two HP neurons to improve the reliability, the LP motif with only one HP neuron in figure 27 is 1st order improved LP motif.

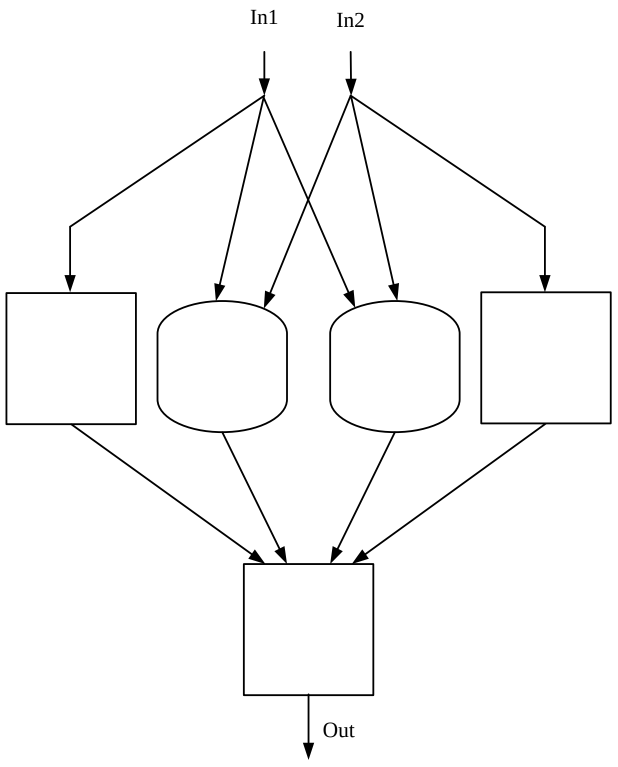


Figure 29. 2nd order motif improved LP neuron

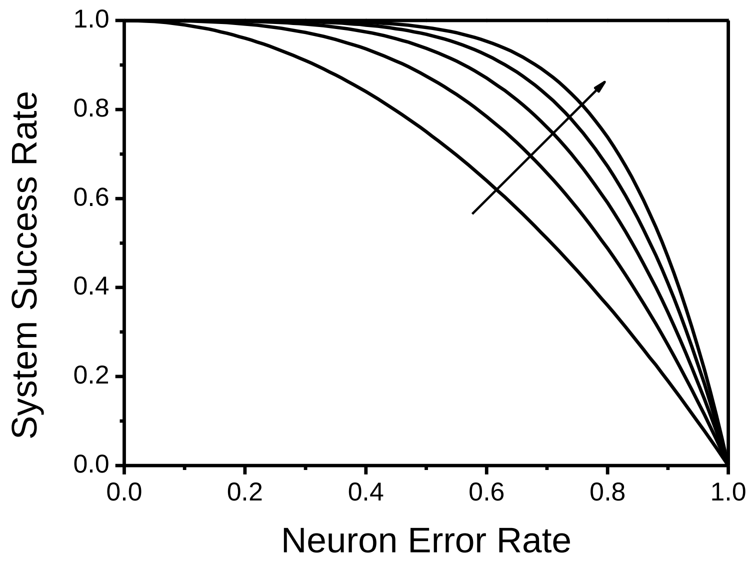


Figure 30. The reliability can be further improved by adding more HP neurons

With an increasing number of orders of the motif improvement, the asymptotic behavior of the motif improvement is presented in Figure 30. This solution provides the flexibility to meet various reliability demands, at the cost of more neurons.

There is another benefit of the motif—delay: the motif design only has two levels in the design no matter how many HP neurons are inserted while the number of levels in TMR increases linearly when adding more TMRs.

4.3 Motif Applied On Arithmetic Circuit Design

The motif design has been proved effective for one single neuron. However, just improving the reliability of a single neuron is not enough. The capability to improve reliability in complex circuit design is more important.

4.3.1 MOTIF APPLIED ON ADDER

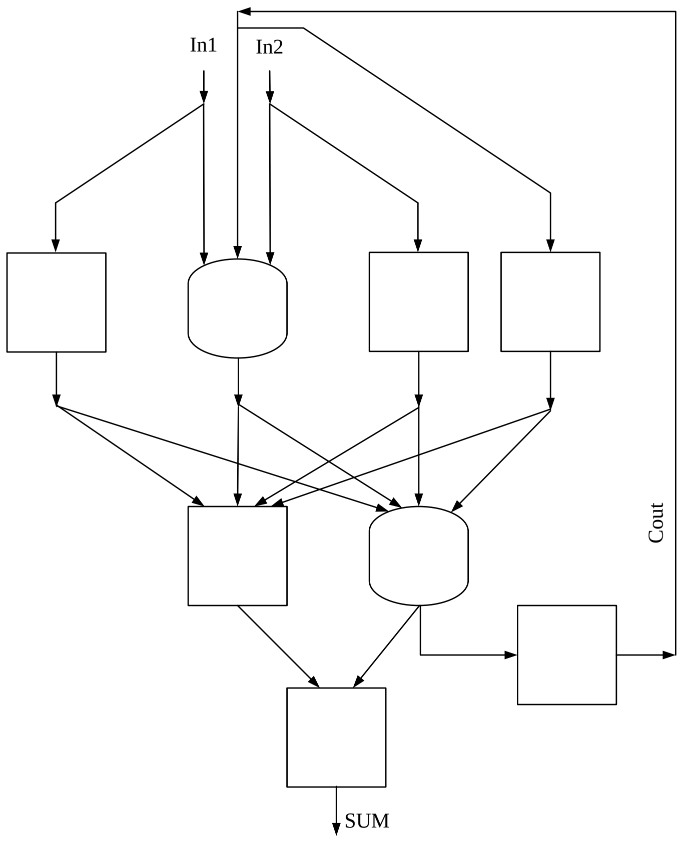


Figure 31. Motif improved adder

The motif improved 1-bit full SNP adder is presented in Figure 31. Since In1, In2 and feedback are sent into both neuron 1 and neuron 2 (in figure 17), these two neurons could share one set of motif.

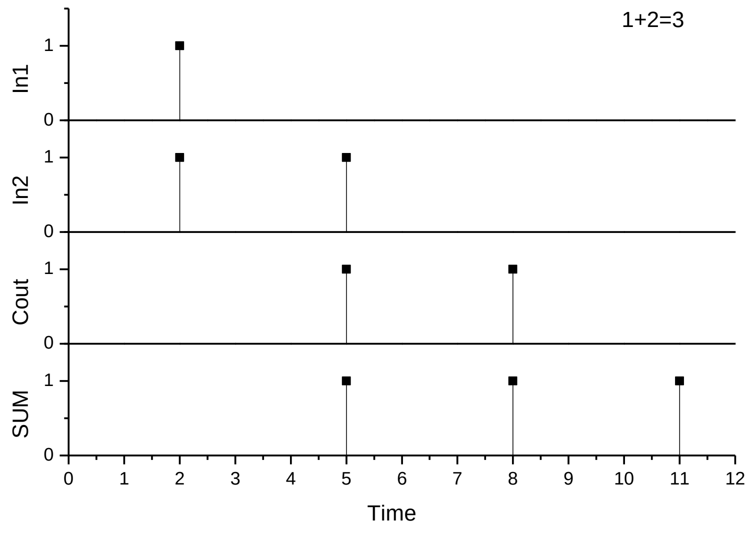


Figure 32. Timing diagram of motif improved adder

Figure 32 illustrates the timing diagram of the motif improved adder. The delay from the primary input to the final output represents the number of operation levels in the SNP network. There are three levels in this design, compared to two levels in the original design. Some neurons, such as the LP neurons in the first level, are introduced only as a buffer, in order to synchronize the operation across various branches.

The success rate of the original design is: $1 - \mathrm{e_n} - \mathrm{e_n}(1 - \mathrm{e_n})^2$; the success rate of the motif improved circuit is: $1 - \mathrm{e_n}^2 - \mathrm{e_n}^2(1 - \mathrm{e_n})^2$, which is shown in figure 33. At 95% system success rate, the original design could only tolerate 2.6% error while the motif improved design could tolerate 17.2%. Figure 34 illustrates the reliability comparison among adder motifs with different orders.

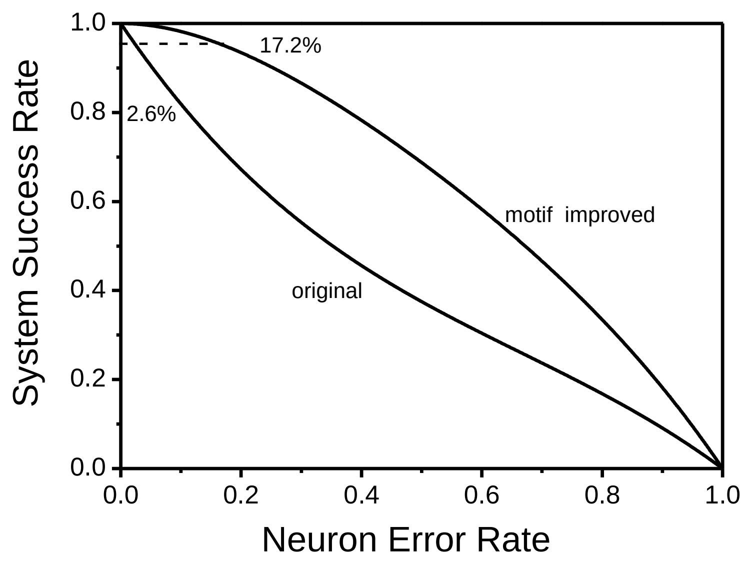


Figure 33. The reliability of the adder design is significantly improved by using the proposed motif.

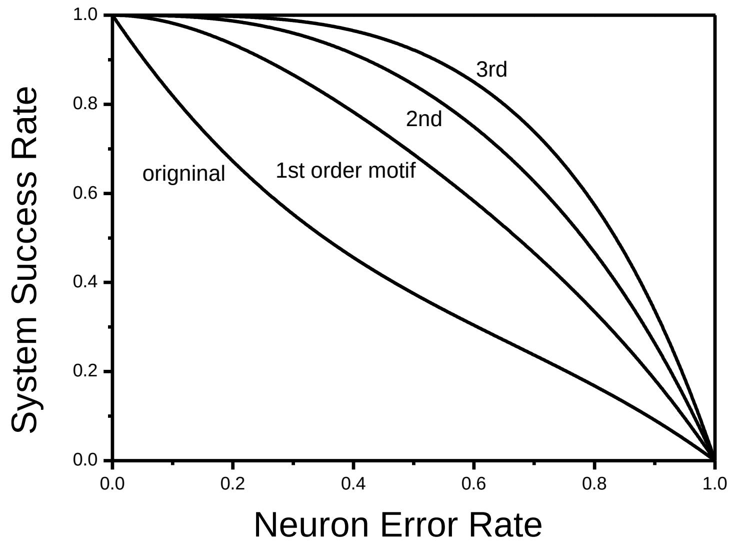


Figure 34. The reliability can be further improved by applying higher order motif structure.

4.3.2 Motif Applied On Comparator

Applying motif on comparator is more complex than on adder as it has three function unit and more critical neurons.

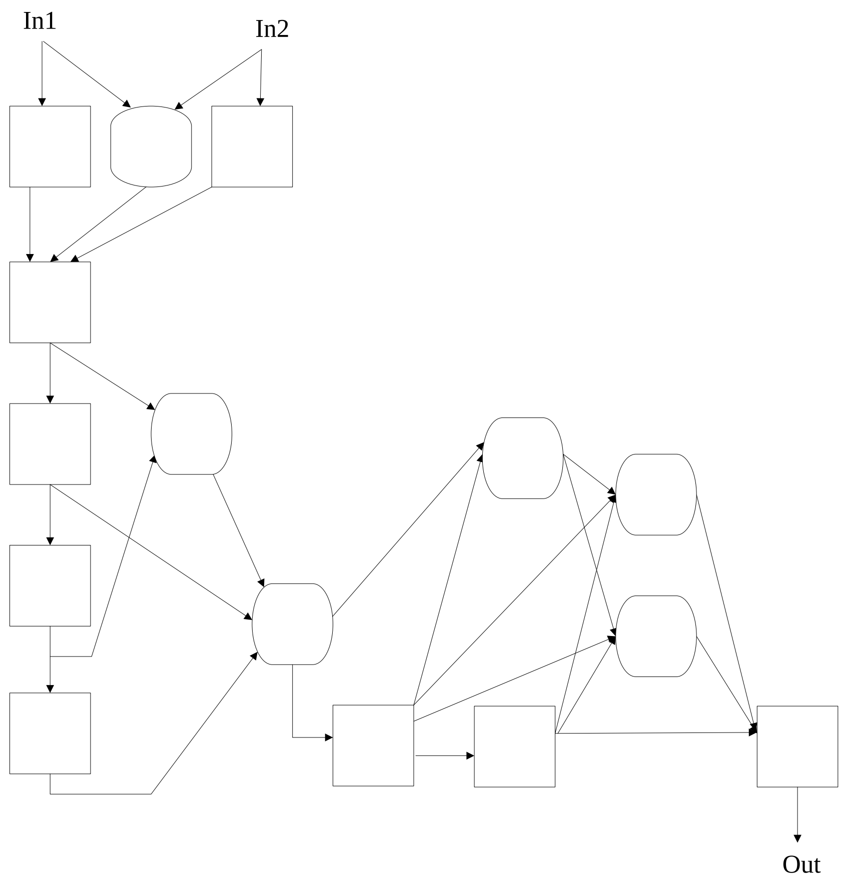


Figure 35. Motif improved comparator

Figure 35 presents motif improved comparator. In the original design, there are four critical neurons, which means an error that happens to any of these neurons leads to the failure of the whole circuit.

The success rate of the original design is: $(1-\mathrm{e_n})^4$; the success rate of the comparator motif is: $(1-\mathrm{e_n}^2)(1-\mathrm{e_n}^2)(1-\mathrm{e_n}^4-3\mathrm{e_n}^3(1-\mathrm{e_n}))$, which are depicted in figure 36.

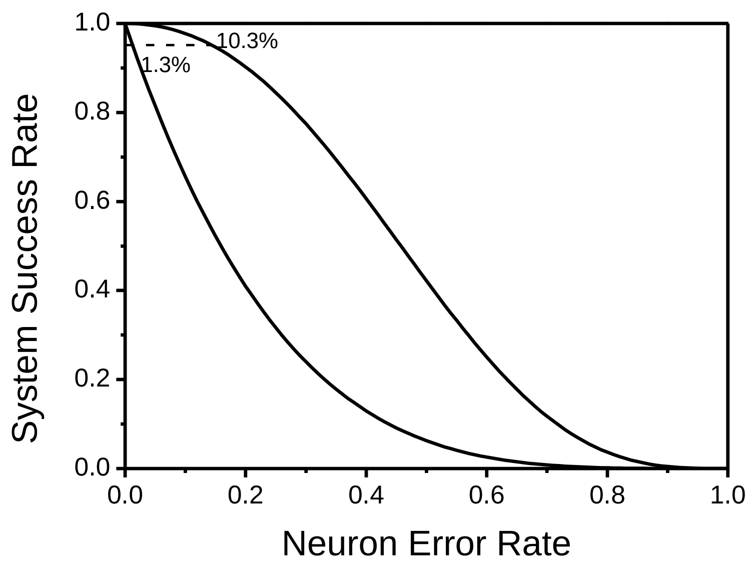


Figure 36. The reliability of the motif improved comparator design is significantly better than that of the original comparator design.

4.3.3 Motif Reliability Enhancement Analysis

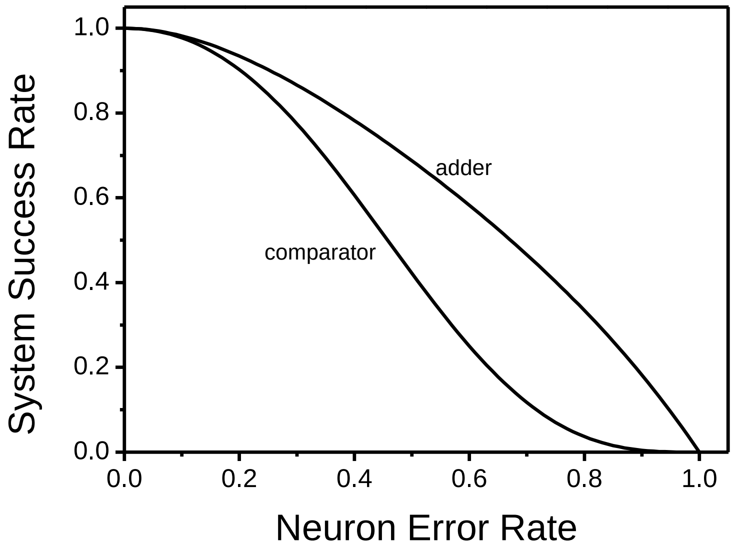


Figure 37. Reliability of comparator vs reliability of adder

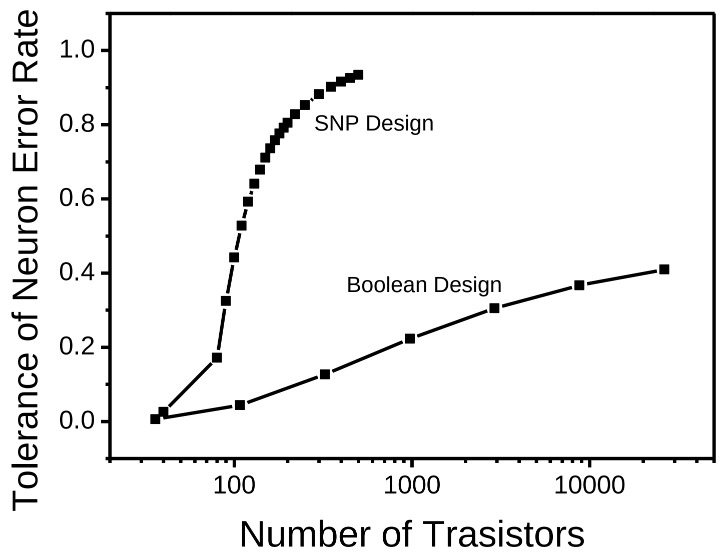


Figure 38. The reliability of the SNP design is effectively improved with smaller area than the Boolean design

Figure 37 evaluates the reliability of these two SNP designs, with an assumption that all LP and HP neurons have the same rate of the 1-bit error. Thus, this study examines the capability of error tolerance in the SNP network. As shown in Figure 37, the comparator has a lower reliability than the adder, due to its complexity. Since the reliability of the network can be improved by increasing the orders of motif, Figure 38 checks the hardware cost in reliability protection. It presents the error tolerance at 95% success rate of an adder in both CMOS-based SNP and Boolean designs. In the SNP design, the tolerance of a single neuron error is rapidly enhanced by moderately adding the motifs. Overall, the results reveal that the SNP network is more robust than the Boolean network, with a lower area cost.

CHAPTER 5

CMOS IMPLEMENTATION

5.1 DRAM-TYPE NEURON DESIGN

In the domain of designing neuromorphic silicon neuron circuits, a lot of analog and digital implementations of spiking neurons are proposed to simulate the realistic and complicated neural biological behavior [22]. However, paying too much attention to the detailed function of dendrite and axon may distract us from the importance of simplified neuron model which would benefit in power and density. This thesis focuses on designing simplified circuit of neurons while still maintaining the primary properties.

5.1.1 BASIC NEURON IMPLEMENTATION

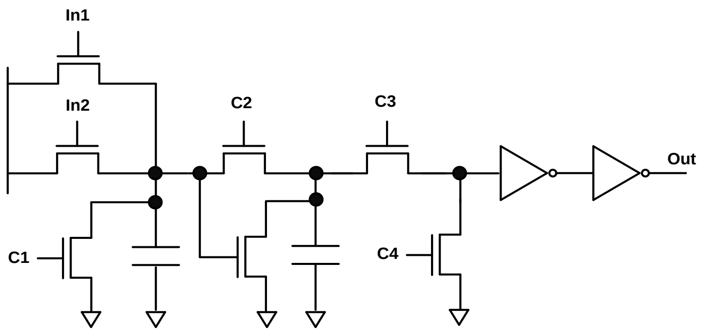
The circuit implementation starts from the design of the LP and HP neurons as they are the fundamental units. MOESFETs are used to build up basic neurons and 45nm PTM are used in all the simulations.

As shown in figure 39, the design is constructed by three parts:

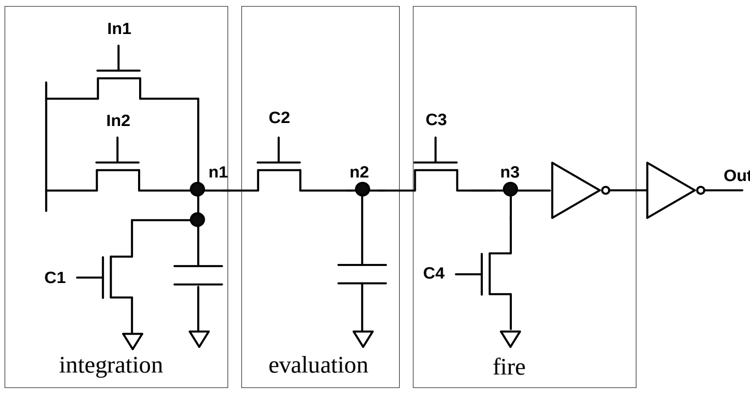
Integration: this DRAM structured part integrates all the input spikes in a capacitor. The voltage on the capacitor represents the number of input spikes. The number of the pass-gates corresponds to the input number of neuron. If more inputs are needed, the number of the pass-gates should be increased.

Evaluation: this part copies voltage at node n1 to voltage at node n2 which is realized by charge sharing between a large cap and a small cap.

Fire: this part compares the voltage at node n2 with the threshold voltage of an inverter and determines whether to fire or not. The second inverter is a buffer which also shapes the output pulse.



(a) 2 input LP neuron



(b) 2 input HP neuron

Figure 39. CMOS-based neuron design, using DRAM-type structures

5.1.2 Simulation Result

Figure 40 presents the SPICE simulation results of a 2-input HP neuron. C1, C2, C3 and C4 are four clocks; In1 and In2 are two input signals; n1, n2, n3 represent the corresponding nodes; Out is the final output which is also the input of the next neuron. This simulation proves the simulated HP neuron functionally correct: it fires out a pulse when there are two input pulses; it forgets the input when only one pulse arrives at the neuron.

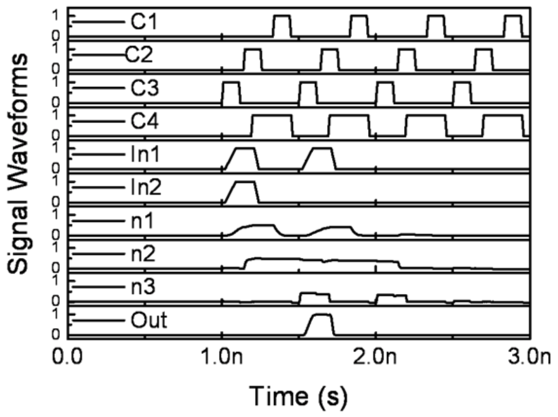
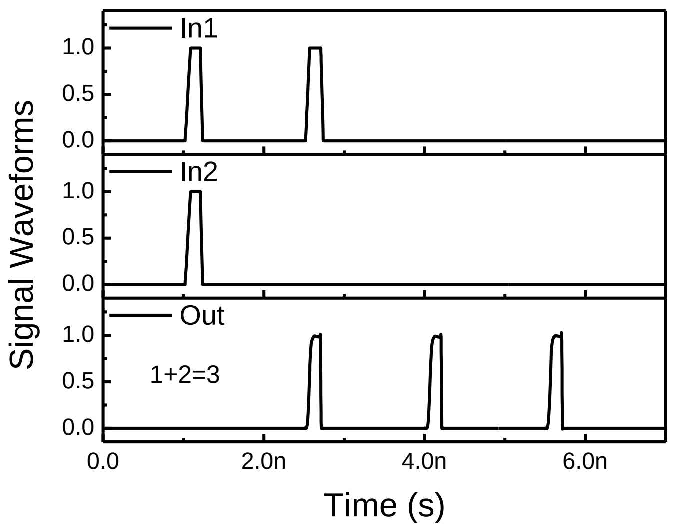


Figure 40. The simulation of 2-input HP neuron

The circuit operates in the following order: first of all, C3 comes to high and the fire stage applied fire/forget rules based on the voltage it reads from n2, meanwhile, In1 and In2 arrive at the integration stage and the capacitor accumulates electrical quantity. Then C2 comes and passes the voltage at n1 to n2; C4 discharges n3 until next C3 arrives. Finally, C1 resets n1 and ends the period.



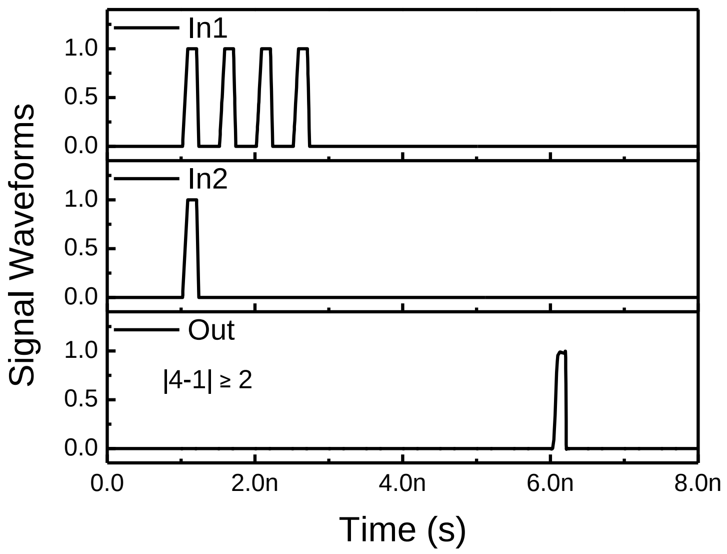


Figure 41. Timing diagram of adder and comparator in simulation

By applying the neuron circuits in SNP networks, various functions could be achieved. Figure 41 presents the timing diagram simulation of the adder and comparator proposed in the previous chapter.

5.2 Circuit Performance

The proposed CMOS design is simple, implying smaller area and lower power consumption. Even though the simplicity of neuron design costs reliability, we can still manage the system reliability via properly designed SNP networks. 45nm SPICE simulation with threshold voltage (V_{th}) variations is applied in the following simulations.

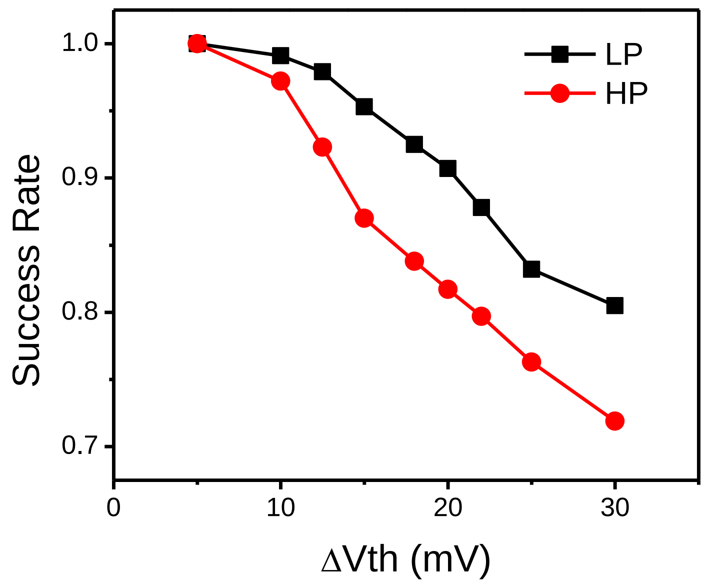


Figure 42. Reliability of a single neuron for various σ_{Vth}

Figure 42 illustrates the reliability of a single neuron for various V_{th} variances (σ_{Vth}). Because of the different circuit structure, LP and HP neurons have two different curves of σ_{Vth} tolerance.

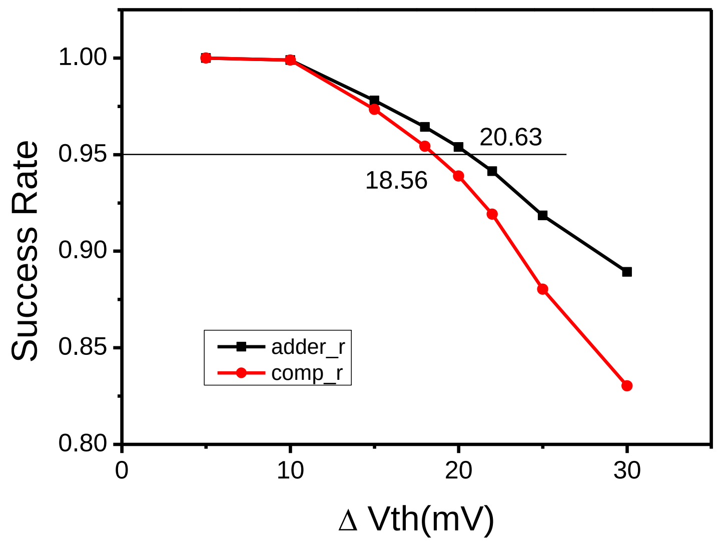


Figure 43. Tolerance of V_{th} variation for motif adder and motif comparator

Figure 43 presents the tolerance of σ_{Vth} for adder motif and comparator motif. The σ_{Vth} for adder and comparator is even larger than the one of single LP and HP neurons, implying that appropriate SNP network will guarantee the system reliability.

Table 9. Comparison between Boolean and SNP on energy and leakage

Function	Design type	Switching Energy (fJ)	Leakage Power (μ W)
Adder	Boolean	5.76	2.15
	SNP	83.8	0.55
Comparator	Boolean	83.73	38.52
	SNP	151.8	0.91

Table 9 compares the power consumption between SNP circuits and the Boolean design based on NAND gates. For a simple design like adder, the SNP design costs more switching energy than the Boolean logic design because it operates more complex. But for a complicated design like the comparator, the switching energy costs on Boolean design and SNP design are compatible. This might partially be due to the fact that the scale of a SNP design does not increase as fast as the function complexity.

In both design cases, the leakage power of SNP circuit is much smaller than that of Boolean circuits which profits from the reset operation at each cycle. All over all, the design of SNP proposes a tradeoff between active and leakage power.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This thesis presents a first step toward designing integrated circuit with SNP systems. The proposed LP (Low Pass) and HP (High Pass) neurons are proved capable of implementing arbitrary SNP neurons and Turing complete, which make them basic building neurons in circuit designs. Representative arithmetic functions are constructed by the circuit implemented by LP and HP neurons.

One spike missing error is proved to be equivalent to threshold increasing by one error. For simplicity, the threshold error model is analyzed through the thesis. The reliability discussion starts from improving the reliability of a single neuron. The motif structure is proposed and proved to be much more efficient than the traditional TMR improvement. The motif structure is also applied on the arithmetic circuits and presents its efficiency in reliable design.

The neuron model is implemented into DRAM-type CMOS circuits. The 45nm SPICE simulation results illustrate that the SNP network is able to reach high success rate while tolerating an excessive amount of unreliability at the device level. Besides, SNP based circuits have much smaller leakage than the Boolean logic circuit while maintaining compatible power consumption over the Boolean logic circuits.

Overall, SNP system presents its design potential for IC designs. Future work is needed in designing basic SNP functional unit, automatic generation of SNP neuron circuits, synthesis on SNP networks and design optimization of SNP neuron circuits.

REFERENCES

[1] Gheorghe Păun, “Computing with membranes,” *Journal of Computer and System Sciences 61*, 108_143 (2000) .

[2] M. Ionescu, G. Păun, T. Yokomori, “Spiking neural P systems,” *J. Fundamental Informaticae,* vol., 71, no. 2-3, pp. 279-308, Aug. 2006.

[3] “The Neuron - Ship” [Online], Available: webspace.ship.edu/cgboer/theneuron.html

[4] Mark F. Bear, Barry W. Connors, Michael A. Paradiso, “Neuroscience: Exploring the Brain” third edition, 2006.

[5] Dee Unglaub Silverthorn, “Human Physiology: An Integrated Approach” San Francisco: Pearson. p 249-263.

[6] John E. Hall, “Guyton and Hall Textbook of Medical Physiology,” 12th edition, 2010

[7] S. Nassif, N. Mehta, Y. Cao, “A resilience roadmap,” Design, Automation and Test in Europe, pp. 1011-1016, 2010

[8] G. B. Ermentrout, R. F. Galán, N. N. Urban, “Reliability, synchrony and noise,” *Trends Neurosci.*, vol. 31, no. 8, pp. 428-434, Aug. 2008.

[9] J. Haag, A. Borst, “Encoding of visual motion information and reliability in spiking and graded potential neurons,” *J. Neurosci.*, vol. 12, no. 12, pp. 4809-4819, Jun. 1997.

[10] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[11] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Trans. the Society for Computer Simulation*, vol. 14, no. 4, pp. 1659-1671, Dec. 1997.

[12] K. Bernstein, R. K. Cavin, W. Porod, A. Seabaugh, J. Welser, “Device and architecture outlook for beyond CMOS switches,” *Proc. IEEE*, vol. 98, no. 12, pp. 2169-2184, Dec. 2010.

[13] M. Sharad, C. Augustine, G. Panagopoulos, K. Roy, “Spin-based neuron model with domain-wall magnets as synapse,” *Trans. Nanotechnology*, vol. 11, no. 4, pp. 843-853,

Jul. 2012.

[14] D. Kuzum, R. G. D. Jeyasingh, B. Lee, H.-S. P. Wong, “Nanoelectronic programmable synapses based on phase change materials for brain inspired computing,” *Nano Lett.*, vol. 12, no. 5, pp. 2179-2186, Jun. 2011.

[15] H. Chen, M. Ionescu, T.-O. Isdorj, “On the efficiency of spikign neural P systems,” *Intern. Conf. on Electronics, Information, and Communication*, pp. 195-206, 2006.

[16] A. Leporati, C. Zandron, C. Ferretti, G. Mauri, “Solving numerial NPcomplete problems *with* spiking neural P systems,” *International Conference on Memberane Computing*, pp. 336-352, 2007.

[17] M. Schmitt, “On computing Boolean functions by a spiking neuron,” *J. Annals of Mathematics and Artificial Intelligence*, vol. 24, no. 1-4, pp. 181-191, 1998.

[18] M. Cavaliere, I. Mura, “Experiments on the relability of stochastic spiking neural P systems,” *J. Natual Computing*, vol. 7, no. 4, pp. 453-470, Dec. 2008.

[19] M. A. Gutiérrez-Naranjo, A. Leporati, “First steps towards a CPU made of spiking neural P systems,” *J. Comput. Commun. Control*, vol. 5, no. 3, pp. 244-252, 2009.

[20] X. Zeng, T. Song, X. Zhang, L. Pan, “Performing four basic arithmetic operations with spiking neural P systems,” *Trans. NanoBioscience,* vol. 11, no. 4, pp. 366-374, Dec. 2012.

[21] X. Zhang , X. Zeng, T. Song , L. Pan, “Spiking Neural P Systems for Arithmetic Operations,” Sixth International Conference on Bio-Inspired Computing: Theories and Applications, 2011

[22] G. Indiveri et al., “Neuromorphic silicon neuron circuits,” *Frontiers in Neuroscience*, vol. 5, no. 73, pp. 1-23, May 2011.