特集

An architecture of Small-scaled Neuro-hardware Using Probabilistically-coded Pulse Neurons with On-Chip Learning*

川島 毅 Takeshi KAWASHIMA
石黒章夫 Akio ISHIGURO
大熊 繁 Shigeru OKUMA
菅原良一 Ryoichi SUGAWARA
田中裕章 Hiroaki TANAKA

An architecture of neuro-hardware with on-chip learning that when compared to the conventional approach can be realized on a small-scale digital circuit is proposed. In order to reduce the scale of the circuits, the architecture employs a new method of computing the membrane potential and the sigmoidal function by encapsulating the probability properties into a relative delay between two pulses. The back-propagation algorithm is applied to the network, which is also realized using pulse delay calculation.

Key words : Neuro-hardware, Pulsed neuron, Probabilistically coding, Digital circuit, Hardware miniaturization, On-chip learning

1. INTRODUCTION

The neuro-hardware has significant advantages over programmed neural network on computer due to its potential high-speed calculation by parallel processing. There are two major ways to implement neuro-hardware into electrical circuit. One is the analog implementation and the other is the digital implementation. The digital implementation is easier to realize and to connect to other circuit or chips. However, digital neuro-hardware generally requires larger circuit-scale on silicon than the analog hardware.[1] The largeness of digital neuro-hardware results from interneuron connection and two principal operations in the neuron, the synaptic multiplication and the sigmoidal calculation. To solve this problem, a pulsed neuron model has been proposed.[2] In this neuron model, the activity of the neuron is represented by pulse sequences. This model is much suitable because a pulse can be specified by 1-bit signal line without multiple-bit bus line, which means wiring area between neurons can be dramatically reduced. On the other hand, the model requires the signal encoding which appropriately represent the neuron activity.

In this article, we propose a novel relative delay encoding, and a novel neuron architecture as well, that focused on small-scale implementation of McCulloch-Pitts type neuron including sigmoidal-type nonlinear mapping by encapsulating the probability properties into relative delay between two pulses. The back-propagation algorithm is applied to the network, which is also implemented using pulse multiplication.

2. RELATIVE DELAY CODING OF INPUT PULSES

The equation of neural calculation we discuss is given by:

$$x = \sum_i w_i \cdot a_i \tag{1}$$

$$y = f(x) \tag{2}$$

where a_i denotes i th input of the neuron, w_i the synaptic weight between a_i and the output of previous neuron. f denotes the nonlinear function such as sigmoidal function:

$$f(x) = \frac{1}{1+\exp(-x)} \tag{3}$$

In the architecture, both an input value and an output value are represented by a relative delay between 2 pulses on different signal line A_i and T as illustrated in Fig. 1. On the additional signal line named standard pulse T, the pulses occur at constant interval:

$$T = \{T^1, T^2, \ldots\} \tag{4}$$

$$A_i = \{A_i^1, A_i^2, \ldots\} \tag{5}$$

Then, the input value a_i is defined by the relative delay between the corresponding k th pulses:

$$a_i^k = T^k - A_i^k \quad (k = 1,2,\ldots) \tag{6}$$

＊John Wiley & Sons, Inc. の了解を得て，EEJ Vol.139，No.4（2002）より一部加筆して転載

Actually the relative delay a_i^k is measured by counting the number of fast system clock, it is normalized from 0 to 1 corresponding to the neuron activity. For example, if the pulse T^k and A_i^k occur simultaneously, it means that the input value 0 at time interval k. Similarly, if the pulse A_i^k occurs just before the next standard pulse T^{k+1}, it means 1. In this way, the proposed encoding always has a set of pulses even if the input has no activity. In the pulse encoding, it requires minimal wiring area on silicon because the signal can be represented with 1-bit line compared to bus interface.

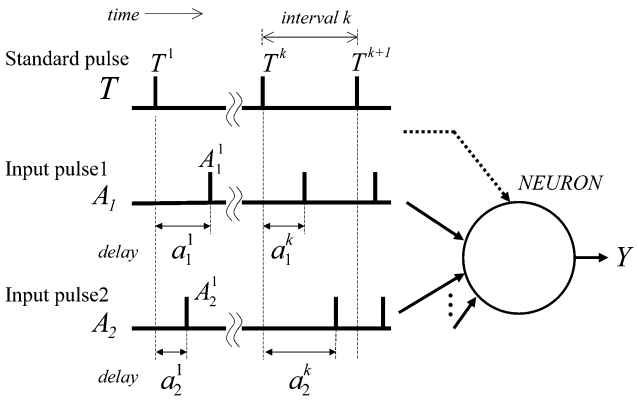

Fig. 1 Relative delay encoding

3. PROBABILISTIC CALCULATION OF NEURON

There are two key operations to prevent the neural calculation from realizing into small circuit on silicon. One is the multiplication of input value with synaptic weight. Since a parallel multiplier consists of many adders, it requires large amount of circuit. The other is the memory for nonlinear mapping of the sum of weighted inputs. For the digital hardware, counters, comparators, and random number generators are more simple and acceptable for implementation. In our approach, these two operations are calculated probabilistically which can be realized by these simple circuits.

3.1 Synaptic multiplication

The principle of synaptic multiplication is similar to coin-tossing trials. After tossing a fair coin repeatedly for 100 times, it will result 100×0.5=50 head-sides. By extending the principle, repetitive coin-tossing trials with head-side probability w_i/w_M will result a_i^k head-sides on average. From the mathematical point of view, the following result is obtained.

Theorem 1 Let a synaptic weight w_i ($0 \le w_i \le w_M$; w_M be constant) be constant integer through time interval k, let X be a uniform random variable with density function U (0, w_M). Then, after the repetitive trials of comparing X with w_i for a_i times, the expected value of the number L, in which each trial results $X \le w_i$, is $E(L) = w_i a_i / w_M$.

Proof A trial being successful if $X \le w_i$ is considered to be a Bernoulli trial of successful rate w_i/w_M. Thus, after repetitive trials for a_i times

$$L \sim B(a_i, w_i / w_M) \tag{7}$$

where B means binominal distribution. So, the expected value $E(L) = w_i a_i / w_M$, the variance

$V(L) = a_i w_i (w_M - w_i) / w_M^2$. □

Theorem 1 implies that the multiplication result is given probabilistically by the expected value. Since the inputs coming through a neuron are a set of pulses T^k and A_i^k, the comparing trial starts at the timing of occurrence T^k and ends at A_i^k in proportion to the relative delay a_i. The multiplication completes within the interval k for each input i. The number of the fast system clock can measure a_i with high accuracy if the clock frequency is sufficiently fast compared to the pulse interval k. If $w_i < 0$, the absolute value of w_i is calculated, then the sign of the resulting expected value is reversed.

Recall that L is random variable, it has some dispersion. From equation (7), the variance $V(L)$ takes the maximum value $a_{max} w_M / 4$ at $a_i = a_{max} = 1$ and $w_i = w_M / 2$. This worst dispersion spreads to 4.0±0.47 with 95% probability at the case $a_i w_i = 1.0 \times 4.0 = 4.0$ in theory. However, the experimental dispersion spreads only within ±0.03 for realized hardware due to devised uniform random generator as discussed later.

Finally, the sum x of weighted inputs in equation (1) is obtained by adding the outputs of all multipliers.

3.2 Nonlinear function

After the multiplication, nonlinear mapping $f(x)$ is carried out. The probabilistic technique is also applied

to the calculation to eliminate the large mapping memory.

Let us consider cumulative distribution function $F(x)$ of normal distribution function $g(t)$ with mean 0 and variance σ^2:

$$F(x) = \int_{-\infty}^{x} g(t)dt \tag{8}$$

The shape of $F(x)$ is quite similar to sigmoidal function $f(x)$ as illustrated in **Fig. 2**. Furthermore, it is convenient that it takes $F(0) = 0.5$, $F(\infty) = 1$, and $F(-\infty) = 0$, which is identical to $f(x)$. In our neuron architecture, the nonlinear mapping is carried out using $F(x)$. Following theorem gives us how to calculate the integration in equation (8).

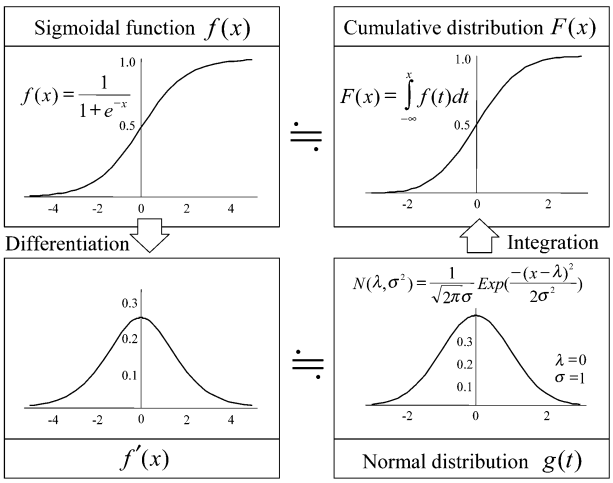

Fig. 2 Approximation of sigmoidal function by normal distribution function

Theorem 2 The cumulative distribution function $F(x)$ of random variable $X \sim N(0, \sigma^2)$ is equal to the probability $x \geq 0$ for random variable $G \sim N(0, \sigma^2)$.

Proof From the definition of $g(t)$ and line-symmetricity with $t = 0$, the equation (8) can be transformed to:

$$F(x) = \int_{-x}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-t^2}{2\sigma^2})dt$$

Replacing the variable t with $s = t - x$,

$$= \int_{0}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-(s-x)^2}{2\sigma^2})ds$$

$$= \int_{0}^{\infty} \tilde{g}(s)ds \tag{9}$$

Hence $\tilde{g}(s)$ represents normal distribution with expected value x, equation (9) implies that $F(x)$ is equal to the integration of probability density function $N(x, \sigma^2)$ with $x \geq 0$. □

Theorem 2 states that $F(x)$ is obtained just by checking the sign of the random numbers generated from normal distribution with expected value x.

Figure 3 illustrates **Theorem 2**. $F(x)$ corresponds to the area ratio of hatched area for total area (area size 1).

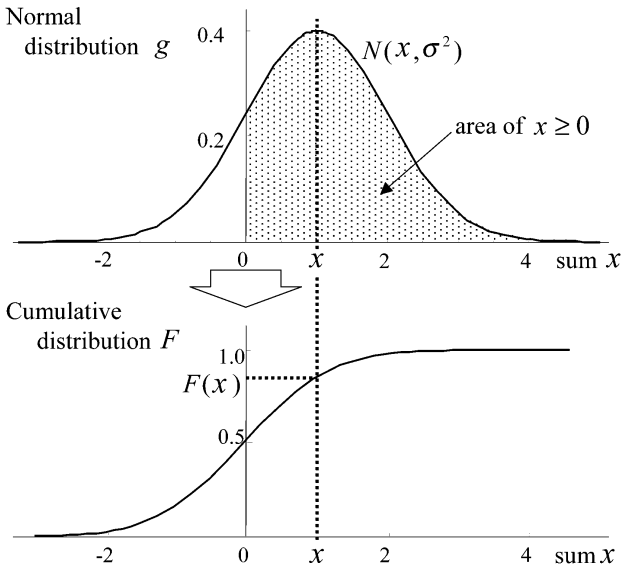

Fig. 3 Principle of the calculation of a sigmoidal function

With the digital circuit, $F(x)$ is approximately calculated by the following function $\hat{F}$:

$$\hat{F}(x_k) = \frac{1}{m}\sum_{k=1}^{m} sign(x_k) \tag{10}$$

where x_k ($k = 1,...,m$) is random number generated from the normal distribution $N(x, \sigma^2)$, and $sign(\cdot)$ is a sign function:

$$sign(t) = \begin{cases} 1 & if\,(t \geq 0) \\ 0 & if\,(t < 0) \end{cases} \tag{11}$$

In the following, we describe x_k a normal distributed random number. For example, if $x = 0$ then $\hat{F}(0) = (m/2)/m = 1/2$ because about half of the normal distributed random numbers are expected to be negative. This case corresponds to the hatched area in **Fig. 3** is equal to half of all area.

3.3 Generating normal distributed random numbers

As discussed in the previous section, normal distributed random variable with expected value x is indispensable in order to calculate the sigmoid-like nonlinear function. Such random variable can be generated by adding some uniform distributed random variables.

Lemma 1 Let $X_1, ..., X_n \sim U(-x_M, x_M)$ be independent uniform random variables. Then, the sum $S = X_1 + ... + X_n$ should be approximately normal distribution, that is, $S \sim N(0, n \cdot x_M^2/3)$. Where U means uniform distribution.

Proof Since each X_k is uniform distribution, its expected value is equal to 0 and variance $w_M^2/3$. From the Central Limit Theorem,[3] it results $S \sim N(0, n \cdot x_M^2/3)$ if n is sufficiently large. □

The **Lemma 1** states that S asymptotically be close to the normal distribution as n increases. The larger n gives the better approximation to the normal distribution. However it needs more circuit resources. In this paper, we set $n = 4$ in view of balance the approximation accuracy with the circuit scale. A Linear Feedback Shift Resister (LFSR) with a primitive polynomial can generate discrete uniform random number sequence on $[-x_M\text{-}1, x_M]$. Then the sum of 4 LFSR becomes as follows:

Theorem 3 The probability distribution of sum $\hat{S}$ of 4 discrete random variables $X_1, ..., X_4 \sim U(-x_M\text{-}1, x_M)$ becomes:

$$\hat{S} \sim \frac{1}{2^4 (x_M + 1)^4} \sum_{i=-4(x_M+1)}^{s+4x_M} p(i)\, p(s-i) \tag{12}$$

where x_M is a positive integer, and

$$p(t) = \min(t + x_M + 1, x_M) - \max(t - x_M, -x_M - 1) + 1 \tag{13}$$

Proof The sum $\hat{S}'$ of 2 independent random variables becomes:

$$\hat{S}' \sim \sum_{x_1 + x_2 = s'} f_1(x_1) f_2(x_2) \tag{14}$$

where $f_1(\cdot)$ as well $f_2(\cdot)$ is the probability distribution for the random variable respectively.[4] Recall X_1 and X_2 are independent variables, $X_1 + X_2$ are rewritten using the equation (14):

$$X_1 + X_2 \sim \sum_{x_1 + x_2 = \hat{S}'} f_1(x_1) f_2(x_2)$$

$$= \sum_{x_1 = \max(\hat{S}' - x_M, -x_M - 1)}^{x_1 = \min(x_M + \hat{S}' + 1, x_M)} \frac{1}{2^2 (x_M - 1)^2}$$

$$= \frac{1}{2^2 (x_M + 1)^2} \{\min(\hat{S}' + x_M + 1, x_M) - \max(\hat{S}' - x_M, -x_M - 1) + 1\}$$

$X_3 + X_4$ can be rewritten similarly. Then let $X_1 + X_2$, $X_3 + X_4$ be new random variables, and apply equation (14) again results equation (12). □

In equation (12), $\hat{S}'$ has expected value -2, and variance $4\,(x_M + 3)(x_M - 1)/3 + 5$ (induction; omit proof). The distribution is illustrated in **Fig.4** with $x_M = 127$. The curve has good agreement with the normal distribution S having same expected value and same variance. The expected value of $\hat{S}$ is almost 0 but not 0 strictly; because of the original uniform distribution is not 0.

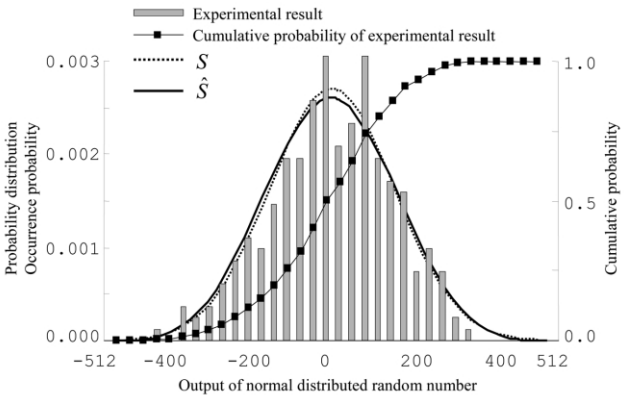

Fig. 4 The normal distribution and an example of normal random numbers

Finally, the desired distribution $\hat{G}$ with expected value x can be obtained by adding $\hat{S} + 2$ to the sum of weighted inputs x:

$$\hat{G} \sim N(x, (x_M + 3)(x_M - 1)/3 + 5) = N(x, \hat{\sigma}^2) \tag{15}$$

4. HARDWARE CONSTRUCTION

Proposed neuron architecture employs probabilistic calculation which can be realized by comparators, random-number generators, and counters unless parallel multipliers or memory. **Figure 5** illustrates the circuit diagram of a pulse neuron.

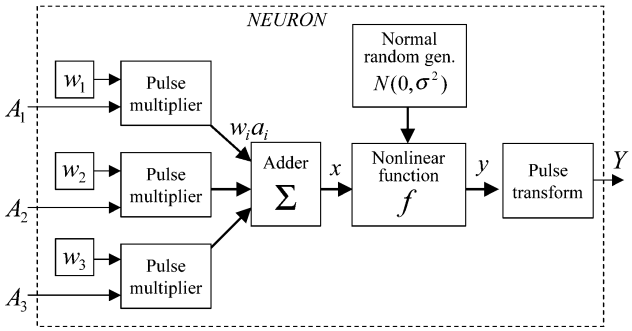

Fig. 5 Circuit diagram of the proposed pulse neuron

A neuron is composed of five blocks; Pulse multiplier block, an Adder block, Nonlinear function block, Normal distributed random number generation block, and Normal transform block.

Figure 6 shows more detailed circuit construction of a neuron. In **Fig. 6**, relative delay a_i is represented by number of the system clock, synaptic weight is stored with 8-bit register. Internal signal $w_i \cdot a_i$, x_j, and a_j are transmitted with 8-bit bus, while A_i and Y with 1-bit.

4.1 Synaptic multiplication

The synaptic multiplication explained in **Theorem 1** is easy to realize by random number generator, a comparator for comparing the random number with synaptic weight $w_i \cdot$ a counter for counting the number of the comparing result.

Uniform random number sequence can be generated by a LFSR. For example a 7-bit LFSR, which consists of 7 flip-flops, can generate pseudo random number on [1,127] at the edge of the clock. Then every random number is compared to wi by the comparator, and the up-down counter counts up if

$$\mathrm{LFSR} \leq w_i \tag{16}$$

The number of occurrence of $w_i \geq$ LFSR from T^k through A_i^k indicates the resultant multiplication. In the case of $w_i < 0$, the absolute value of w_i is compared to the output of LFSR, and the up-down counter counts down if LFSR $\leq |w_i|$. A multiplication result, that is the output of the counter, is fixed at the timing of A_i^k. The circuit scale is less than 1/10 compared to the parallel multiplier of same bit width.

We designed the cycle-length of LFSR 127, which is about half of the maximum delay $a_{max} = 255$, so as to reduce the dispersion. The experimental dispersion is reduced to 0.25 whereas the theorical variance $a_{max} w_M / 4 = 63.75$.

4.2 Nonlinear function

In Normal distributed random generator block, normal distributed random numbers $\hat{S}$ with expected value 0 are generated from **Theorem 3** $\hat{S}$ and x, the

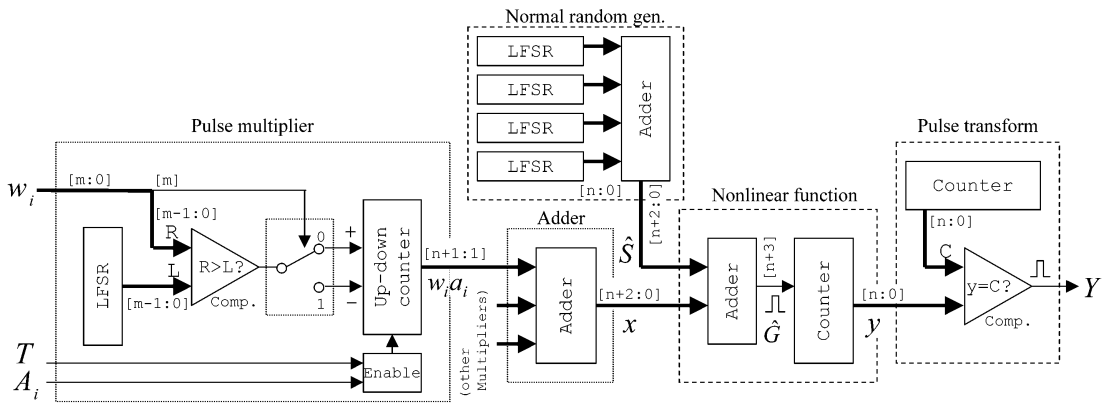

Fig. 6 Detailed circuit diagram of the proposed pulse neuron

output of the Adder, comes through the Nonlinear function block. The block is very simple since it includes only an adder and a counter. The normal random numbers $\hat{G}$ with expected value x are generated by adding $\hat{S}$ with x. The Counter counts up if $\hat{G}>0$ by checking the MSB, or sign bit of the adder. The output of the Counter after the pulse A_i indicates the neuron activity y. The calculation of this part is completed within an interval k.

Figure 4 illustrates the occurrence probability of 255 random numbers based on $\hat{S}$ and its cumulative frequency as well. Although the dispersion in occurrence seems to be rather large in contrast to theoretical $\hat{S}$, expected value -2.01 and variance 22017 almost agree with theory.

In this hardware construction, the division in equation (10) can be omitted because $0 \leq F(x) \leq 1$ in equation (8) is mapped into the integer $0 \leq \hat{F}(x) \leq m$.

Furthermore, the derivative of the nonlinear function is obtained by counting up the number of occurrence $x_j^k = 0$ since normal distribution $N(x, \sigma^2)$ is the derivation of cumulative distribution $F(x)$, that means it is easy to calculate Δw_i for back-propagation learning without large derivative memories.

In the Pulse transform block, obtained output y is transformed into the pulse signal Y.

5. LEARNING

The back-propagation rule is available on learning synaptic weights. The weight is corrected by:

$$\Delta w_{kj}^{(s)} = \eta \delta_k^{(s)} y_j^{(s-1)}$$
$$(k = 1,\ldots,N_s,\ j = 1,\ldots,N_{s-1}) \qquad (17)$$

$$\delta_k^{(s)} = \sigma_k^{(s)} F'(x_k^{(s)}) \quad (s = 1,\ldots,M)$$

$$\sigma_k^{(s)} = \begin{cases} t_k - y_k^{(s)} & (s = M) \\ \sum_{j=1}^{N_{s+1}} w_{kj}^{(s+1)} \delta_j^{(s+1)} & (s = 1,\ldots,M-1) \end{cases}$$

where $y_k^{(s)}$ denotes the output of the neuron in s th layer, t_k denotes training input for k th output neuron for the output layer, η the learning rate. Equation (17) includes derivative $F'(y_k^{(s)})$, which is generally hard to implement into digital circuit. In the architecture, it is clear that the derivative corresponds to $g(\cdot)$ in (8) because $F(\cdot)$ is integration of $g(\cdot)$. Then $F'(x_k^{(s)})$ is rewritten by the probability of $x_i\ (i=1,\ldots,m)$ to be $x_i = x_k^{(s)}$ for the set of normal distributed random numbers $x_1,\ldots,x_m$. It is represented by:

$$F'(x) = \frac{1}{m}\sum_{i=1}^{m} eql(x, x_i)$$

$$eql(x, x_i) = \begin{cases} 1 & (x = x_i) \\ 0 & (x \neq x_i) \end{cases} \qquad (18)$$

For instance, if 1 random numbers are equal to $x_k^{(s)}$ then $F'(x_k^{(s)}) = 1/m$. The circuit for calculating derivative needs only a comparator and a counter as illustrated in **Fig. 7**.

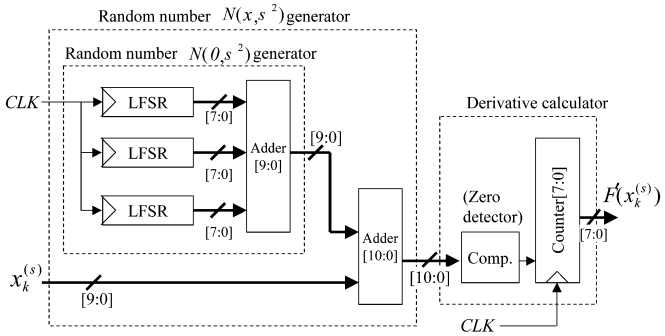

Fig. 7 Derivative calculator for learning

6. CONCLUSION

An architecture of a pulse based neuro-hardware that can be realized on by far a small-scaled circuit compared to the conventional hardware has developed. The hardware has capability of on-chip back-propagation learning. In order to reduce the scale of the circuits, the architecture employs a new method of computing the synaptic multiplication and the sigmoid-like nonlinear function by encapsulating the probability properties into relative delay between two pulses.

This research was supported in part by a grand from Emergent Soft Computer Project held by Nagoya Industrial Science Research Institute.

REFERENCES

1) T. Schoenauer, A. Jahnke, U. Roth, and H. Klar: Digital Neurohardware: Principles and Perspectives, Neuronal Networks in Applications-NN 98 (1998), pp.101-106.
2) Wolfgang Maass, Christofer M. Bishop: Pulsed Neural Networks, The MIT Press (1999)
3) K. Ito: Statistical Theory I,II, Iwanami Shoten (1976)
4) K. Kunisawa: Exercise of Probabilistics and Statistics 1, BaiFuKan (1966)

<著　者>

川島　毅
（かわしま　たけし）
基礎研究所
視覚情報処理，創発型ソフトコンピュータの開発に従事

大熊　繁
（おおくま　しげる）
名古屋大学大学院工学研究科
電子情報学専攻　教授
工学博士
電子情報学に関する教育・研究，主として，創発型ソフトコンピュータの開発，ロボティクス，パワーエレクトロニクスの研究に従事

田中　裕章
（たなか　ひろあき）
基礎研究所
自動車用LSI，IP，ASIC機能回路の研究開発に従事

石黒　章夫
（いしぐろ　あきお）
名古屋大学大学院工学研究科
計算理工学専攻　助教授
工学博士
創発システム，ロボット工学の研究に従事

菅原　良一
（すがわら　りょういち）
基礎研究所
視覚情報処理の研究開発に従事