Analog Integrated Circuits and Signal Processing, 18, 153–162 (1999)

Hardware Implementation of a Pulse Density Neural Network Using Simultaneous Perturbation Learning Rule

YUTAKA MAEDA, ATSUSHI NAKAZAWA AND YAKICHI KANATA

Kansai University, Faculty of Engineering, Department of Electrical Engineering, 3-3-35, Yamate-cho, Suita, Osaka 564 Japan
maedayut@kansai-u.ac.jp

Received May 1, 1997; Accepted February 2, 1998

Abstract. The choice of the learning scheme is very important in the implementation of neural networks to take advantage of their learning ability. Usually, the back-propagation method is widely used as a learning rule in neural networks. Since back-propagation requires so-called error back propagation to update weights, it is relatively difficult to realize hardware neural networks using the back-propagation method. In this paper, we present a pulse density neural network system with learning ability. As a learning rule, the simultaneous perturbation method is used. The learning rule requires only forward operations of networks to update weights instead of the error back-propagation. Thus, we can construct the network system with learning ability without the need for a complicated circuit that calculates gradients of an error function. Pulse density is used to represent the basic quantities in this system. The pulse system has some attractive properties which includes robustness against a noisy environment. A combination of the simultaneous perturbation learning rule and the pulse density system results in an interesting architecture of hardware neural systems. Results for the exclusive OR problem and a simple identity problem are shown.

Key Words: neural networks, pulse density, simultaneous perturbation

1. Introduction

Artificial neural networks (NNs) are modelled on biological neural systems. NNs have a number of applications nowadays in which the NNs are usually implemented on an ordinary digital computer. However, this kind of an implementation does not take advantage of the essential idea; parallelism in biological NNs. In order to fully utilize this feature, suitable implementation is required. Implementation using hardware elements such as very large scale integration (VLSI) is advantageous in this respect.

When we consider the hardware implementation of NNs, one of the difficulties is to realize a learning mechanism. The concept of "Learning on Silicon" is very important [1,2].

The back-propagation method is widely used in many cases. However, it is very complex to implement the back-propagation method, for example, in an electronic system, since the learning rule uses derivatives of an error function to update weights. In particular, it is difficult to implement large scale NNs with learning ability via the gradient method because of the complexity of the realizing mechanism that derives the gradient. From this point of view, we need a learning rule that is easy to realize.

The simplest solution for the problem is the finite difference known as the weight perturbation in the field of NNs [3].

$$\Delta w^i = \alpha \frac{J(\boldsymbol{w} + c\boldsymbol{e}^i) - J(\boldsymbol{w})}{c} \tag{1}$$

where, Δw^i is a modifying quantity for the i-th weight. $\boldsymbol{w}$ is a weight vector, $J(\cdot)$ denotes an error function. c represents a magnitude of the perturbation and α is a positive coefficient. $\boldsymbol{e}^i$ is the fundamental vector whose i-th component is 1 and the others are 0.

However, the learning rule is not efficient for large scale NNs. A. J. Montalvo et al. described an analog VLSI NN using the chain rule perturbation method [4] that is an extension of the simple finite difference technique. In this method, instead of simple serial weight perturbation [3], outputs of neurons Δo are used as follows;

$$\Delta w^i = \frac{J(\boldsymbol{w} + c\boldsymbol{e}^i) - J(\boldsymbol{w})}{\Delta o} \cdot \frac{\Delta o}{c} \tag{2}$$

where, Δo is an output of a corresponding neuron.

On the other hand, the simultaneous perturbation method was introduced by J. C. Spall [5,6]. J. Alespector et al. [7] and G. Cauwenberghs [8] presented the same idea., Y. Maeda also independently proposed a learning rule using the simultaneous perturbation and reported a feasibility of the learning rule [9–11]. At the same time, the merit of the learning rule was demonstrated in VLSI implementation of analog NNs using this rule [12,13].

The advantage of the simultaneous perturbation method is its simplicity. The method is even simpler than the chain rule perturbation method, since there is no need to use outputs of neurons for the chain rule. From a hardware implementation point of view, the exclusion of the intricate mechanism required for deriving the gradient of the error function is preferable. The simultaneous perturbation can estimate the gradient using only the values of the error function. Therefore, the implementation of this learning rule is relatively easy, compared with that of the other learning rule. The details of the learning rule will be given in Section 2.

Pulse techniques including pulse width and pulse stream are also investigated to realize artificial NNs. For example, E.I.El-Masry et al. presented a current-mode pulse width modulation artificial NN that enables efficient implementation of artificial NNs [14]. A.F.Murray et al. also proposed a VLSI NN using analog and digital techniques [15].

The pulse density type of NNs has attractive properties. For example, the system is invulnerable to noisy conditions. Moreover, we can handle quantized analog quantities, based on the digital technology that is relatively effortless to implement. The analogy to biological neural systems is intriguing as well. Eguchi et al. reported a pulse density neural system [16]. They used the back-propagation method as a learning rule.

In this paper, we present a pulse density type of a NN system with learning ability using the simultaneous perturbation method. The combination of the pulse density system and the simultaneous perturbation learning rule results in an easy configuration which can be implemented in the hardware neural network system. In our hardware implementation, pulse density is used to represent the weight values, outputs and inputs.

2. Simultaneous Perturbation Learning Rule

The simultaneous perturbation learning rule is described as follows;

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \Delta \boldsymbol{w}_t \tag{3}$$

$$\Delta w_t^i = \frac{J(\boldsymbol{w}_t + c\boldsymbol{s}_t) - J(\boldsymbol{w}_t)}{c} s_t^i \tag{4}$$

where, $\boldsymbol{w}$ is the weight vector including the threshold of all neurons, α is a positive constant. $\Delta \boldsymbol{w}$ is a modifying vector and Δw^i represents the i-th element of the vector $\Delta \boldsymbol{w}$. $\boldsymbol{s}_t$ and s_t^i denote the sign vector and its i-th element that is 1 or -1, respectively. The sign of s_t^i is randomly determined. Moreover, the sign of s_t^i is independent of the sign of the j-th element s_t^j of the sign vector. That is, $E(s_t^i) = 0$, $E(s_t^i s_t^j) = 0 (i \neq j)$. E denotes the expectation, c is a magnitude of the perturbation. $J(\boldsymbol{w})(\equiv |o - d|)$ denotes the error function defined by an output of the NN o and a corresponding teaching signal d. The details of this algorithm are shown in Fig. 1.

When we expand the right-hand side of equation (4) at the point $\boldsymbol{w}_t$, there exist $\boldsymbol{w}_{s1}$ such that

$$\Delta w_t^i = s_t^i \boldsymbol{s}_t^T \frac{\partial J(\boldsymbol{w}_t)}{\partial \boldsymbol{w}} + \frac{C s_t^i}{2} \boldsymbol{s}_t^T \frac{\partial^2 J(u(\boldsymbol{w}_{S1}))}{\partial \boldsymbol{w}^2} \boldsymbol{s}_t \tag{5}$$

We take an expectation of the above quantity. From the conditions of the sign vector $\boldsymbol{s}_t$, we have

$$E(\Delta w_t^i) = \frac{\partial J(\boldsymbol{w}_t)}{\partial w_t^i} \tag{6}$$

That is, Δw_t^i approximates $\partial J_p(\boldsymbol{w}_t)/\partial w_t^i$. Since the right-hand side of equation (4) is an estimated value of thė first-differential coefficient in the sense of the expectation, the learning rule is a type of a schismatic gradient method [11,12].

An important point is that this learning rule requires only two values of the error function. That is, it requires only two forward operations of the NN in order to obtain estimators of the first differential coefficients of the error function with respect to all weights in the network.

3. Architecture of the Neural Network System

Fig. 2 shows the configuration of our pulse density NN system. The system consists of three kinds of units; weight unit, neuron unit and learning unit. The weight unit calculates the multiplication of the input signals

```
for $p$:=1 to $P_{max}$ do (* $P_{max}$ is a total number of patterns *)
  begin
  ● Add the perturbation $cs_t$ to the weight vector $w_t$.
  ● Obtain a value of the error function $J(w_t + cs_t)$.
  ● Subtract the perturbation $cs_t$.
  ● Obtain a value of the error function $J(w_t)$.
  ● Calculate the difference between these values. (* $J(w_t + cs_t) - J(w_t)$ *)
    for $i$:=1 to $n$ do (* $n$ is a total number of the weight *)
      begin
      ● Multiply the difference by $\alpha s_t^i/c$.
      end;
  ● Update the weight vector.
  ● Renew the iteration.
  end.
```

Fig. 1. Procedure for the learning rule.

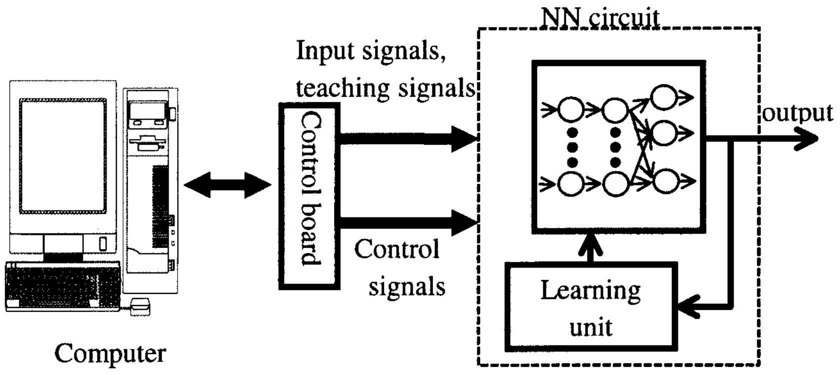


Fig. 2. Configuration of the NN system.

and weight values. The neuron unit gives the weighted sum of outputs. The learning unit achieves the so-called learning process using the simultaneous perturbation method and sends modifying quantities to the weight units. In this system, the input signals, the teaching signals and some timing signals used in all the units are generated by a personal computer.

Figs. 3 and 4 are photos of our system. Fig. 3 shows the overall system. Basically, the system is composed of individual elements such as ICs, resistors and capacitors. Fig. 4 shows the circuit board in which the weight units and the neuron units are implemented.

Next, we detail the features of these units.

3.1. The Weight Unit

After a modification of a weight value stored in the weight unit, the weight unit adds the perturbation to the value. Then, the unit multiplies the weight value and an output of a neuron in the previous layer.

The weight unit consists of a weight calculation part and a pulse generation part. The weight calculation part carries out addition or subtraction of the perturbation and updates the weight value based on the quantity by the learning unit. The pulse generation part generates a pulse stream for which the

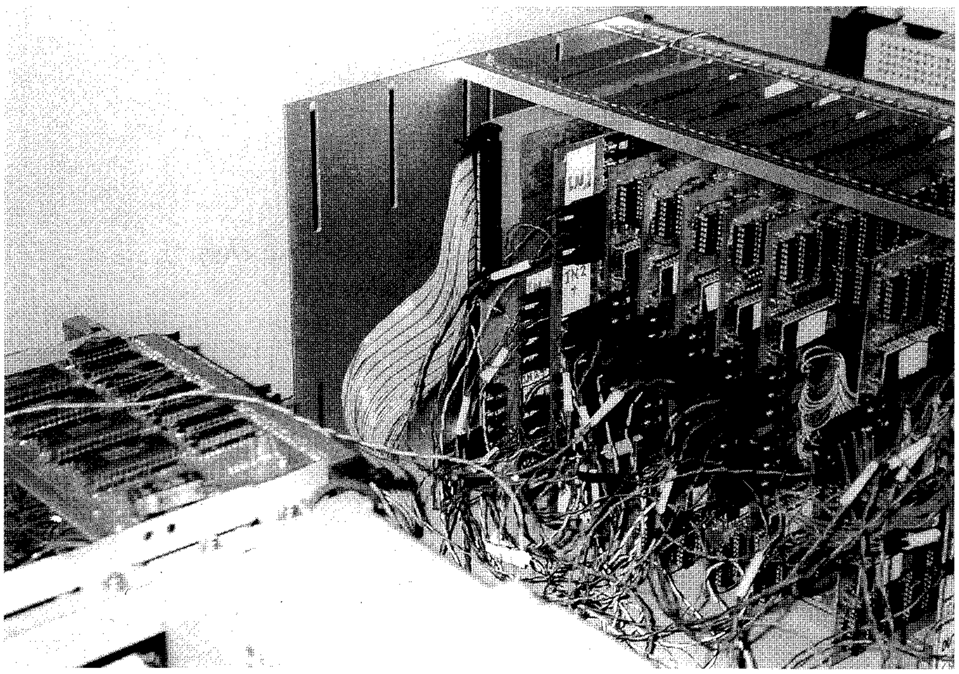


Fig. 3. Fabricated pulse density neural network system.

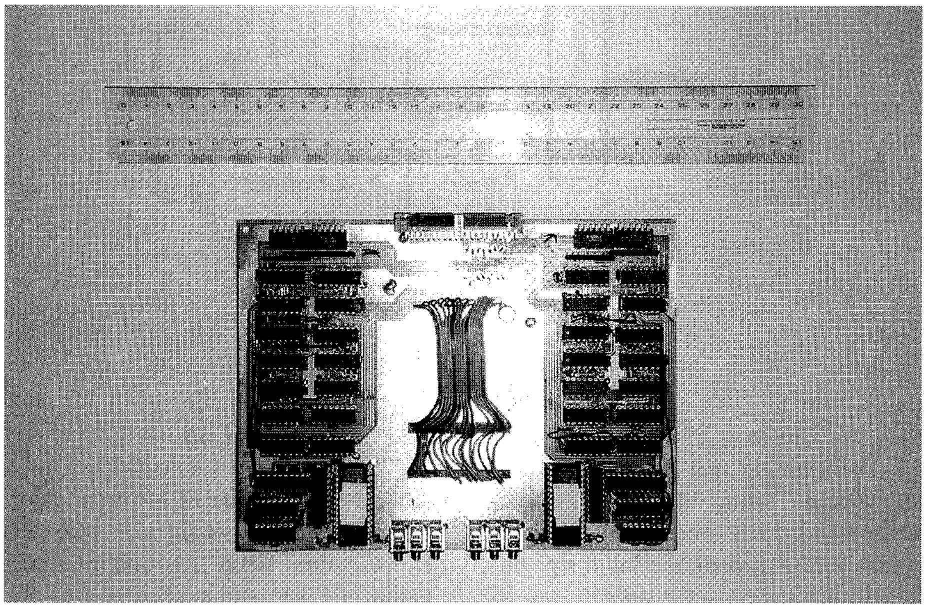


Fig. 4. A board for weight units and neuron units.

number of pulses per unit time is proportional to the weight value. An AND operation of these pulses and the input pulses means a multiplication of the weight value and the input value.

If the sign of the result is positive, the output is sent to a positive side of the neuron unit. If the sign is negative, the output is sent to a negative side. The block diagram is shown in Fig. 5.

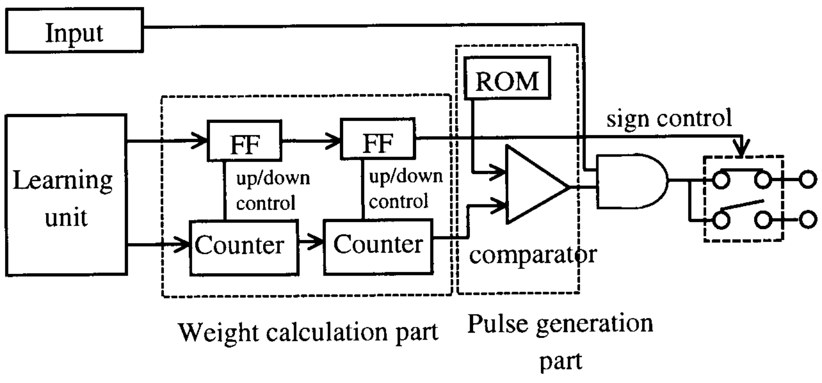


Fig. 5. The weight unit.

3.1.1. The Weight Calculation Part. This part consists of counters, flip-flops (FFs) and the other logical elements and performs the modification of the weights. Moreover, it achieves an addition or a subtraction of the perturbation.

First of all, we store the initial values of all weights and their corresponding signs in counters and FFs, respectively. The modifying quantity corresponding to each weight is sent from the learning unit. This quantity is connected to one of the counters. Counting up or down is decided by the sign of the modifying quantity stored in the FF. This procedure results in a new weight value.

Another role of the weight calculation part is to add a perturbation to the weights. This is simultaneously done for all the weights. Counters are used for this operation as well. The sign of the perturbation is generated in the pulse generation part described below.

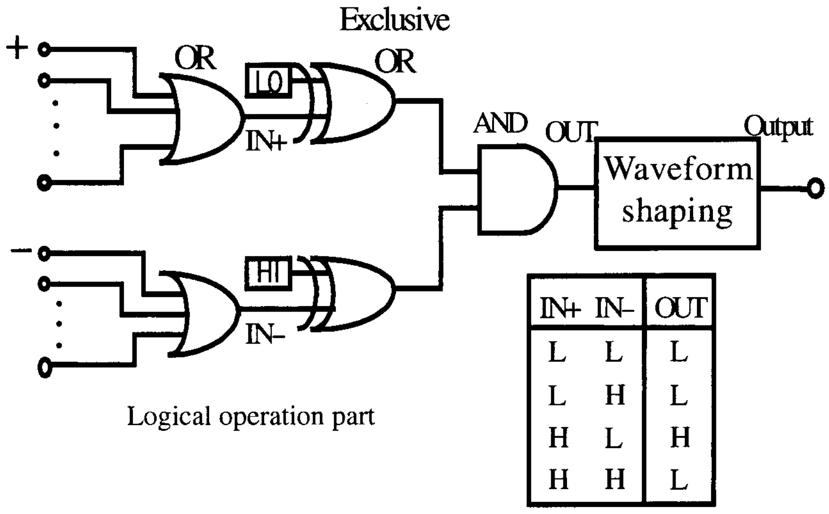
3.1.2. Pulse Generation Part. This part converts the weight values calculated in the weight calculation part into pulse series. It consists of comparators and a ROM. 32768 uniform random numbers are stored in the ROM. We compare a weight value with a random value stored in the ROM. Then, if the weight is larger than the random number, this circuit generates a single pulse, if not, no pulse is generated. We repeat this procedure for unit time. New random numbers are used step by step. Therefore, a large weight results in many pulses and a small weight results in very few pulses. This means that the weights in our system are represented by pulse density.

3.2. *Neuron Unit*

This unit consists of a logical operation part and a waveform shaping part. The former part calculates the sum of all inputs by means of a simple OR operator. The input-output characteristics are realized by the saturation of pulse density. That is, even if the weighted sum for a neuron is extremely large, the maximum number of pulses per unit time is restricted. In our system, the maximum number of pulses per unit time is 256. No pulse means a negative limitation of the weight value. Thus, the system uses the linear function with limitation instead of the sigmoid function. The block diagram is shown in Fig. 6.

In Fig. 6, $+$ or $-$ symbols indicate the sign of the weight; outputs of the weight unit. That is, positive and negative weights are transmitted separately. IN $+$ and IN $-$ represent the positive and negative sum of weighted inputs, respectively.

Digital elements have some sort of time delay. It is important to take this into account, when we design a digital circuit. To avoid malfunction caused by this time delay, we use the waveform shaping part. The part consists of two FFs. When the first FF latches a signal of the logical operation part at time t, another FF outputs a signal of time $(t-1)$. At the next sampling time $(t+1)$, this FF latches a signal of time $(t+1)$ and the first FF outputs the signal of time t. As a result, one clock time delay occurs. However, this prevents malfunction caused by the time delay of the other elements.



IN+	IN-	OUT
L	L	L
L	H	L
H	L	H
H	H	L

Fig. 6. The neuron unit.

3.3. Learning Unit

Learning ability is the most important factor for NNs. Therefore, it is crucial to realize a hardware NN system with learning ability. This unit actualizes this ability. Concretely, this unit gives modifying quantities for all weights using teaching signals and outputs of the NN. The block diagram is shown in Fig. 7. The pulse generation part in this unit has the same structure as that in the weight unit. The learning of this unit is based on the simultaneous perturbation learning rule. One of the features of this learning rule is that it requires only forward operations of the NN. In other words, if we use the ordinary back-propagation method, we have to realize so-called error back propagation in order to obtain a derivative of the error function. However, this learning rule requires only values of the error function. This means that we need only forward operation of the network to carry out the learning of the weights. Details of the operations of this unit are as follows.

1. Reset the counters 1 and 2 in the learning unit.
2. Forward operation of the NN with perturbations. The output counts up the counter 1. Next, the teaching signals count down the counter 1. The pulse generation part produces pulses using the

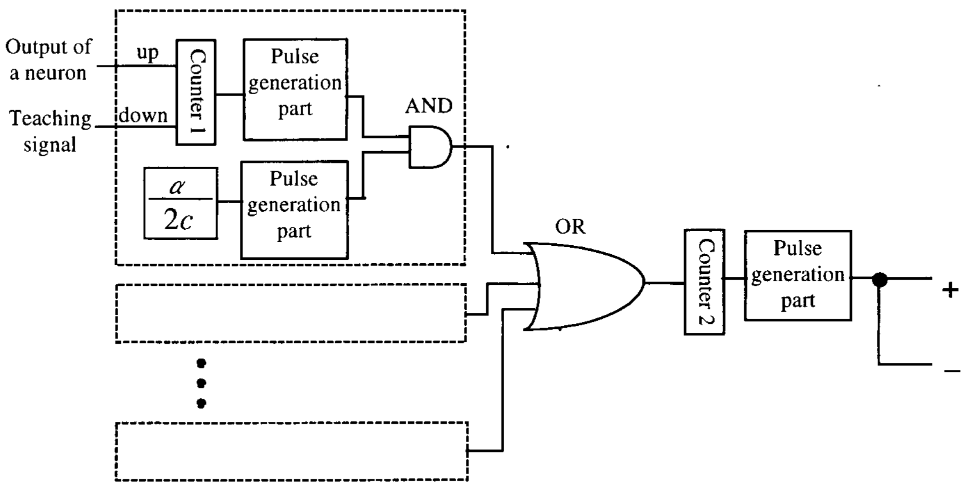


Fig. 7. The learning unit.

value stored in the counter 1. After AND operation of these pulses and pulses corresponding to the constant $\alpha/2c$, we set the value in the counter 2.

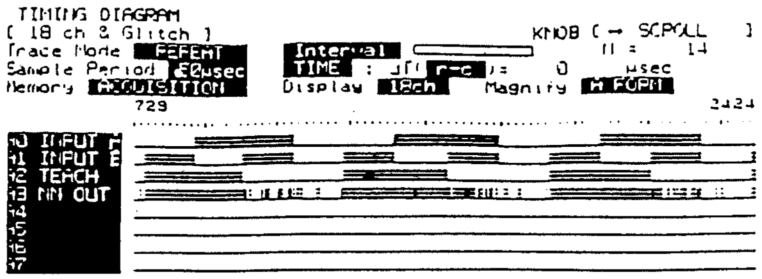
3. Reset counter 1. Forward operation without the perturbations. The output counts up the counter 1. Next, the teaching signals count down the counter 1. The pulse generation part produces pulses using the value stored in the counter 1. After AND operation of these pulses and pulses corresponding to the constant $\alpha/2c$, we set the value in the counter 2.
4. Using a counter, we obtain the difference of these values.
5. The result is sent to all weight units in order to update the weight values.

4. Results

Now the Exclusive OR problem is considered. The network is a three layered feed forward type. The numbers of neurons in each layer are 2, 2 and 1. The inputs, teaching signals and the results are shown in Figs. 8 and 9. Logical representations 0 and 1, are realized in the pulse density form as shown in Figs. 8 and 9. 256 pulses for unit time denote logical 1 and no pulse for unit time means logical 0. We can find a good agreement between the teaching signal and output of the system, which means that the system works well. The system requires about 60 seconds to learn the relation on average. The clock frequency in this example is about 50 KHz. Fig. 10 shows the case that the system could not learn the exclusive OR relation. In this case, the system was captured in the so-called local minimum.

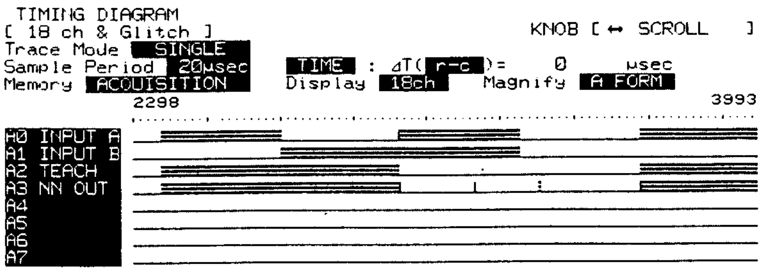
Next, we handle the learning of the linear function $y = x$ and the function $y = 1 - x$. We select five learning points; 0.0, 0.3, 0.5, 0.7 and 1.0. The numbers of the neurons in each layer are 1, 4 and 1. The density of pulses for a unit time denotes a value of x or y. The results are shown in Figs. 11 and 12. On an average, we require 90 seconds and 150 seconds for $y = x$ and $y = 1 - x$, respectively. The clock frequency is about 10 kHz. Details of the output pulses are shown in Fig. 13.

Since this system is fabricated using individual discrete parts, it was difficult to work under a high clock frequency. However, our purpose is to confirm the feasibility of the idea of the pulse density NN system via the simultaneous perturbation learning rule. From this point of view, we could confirm a viability of the idea. Implementation by FPGA or ASIC will contribute to the realization of a high-speed and high-stability system using this idea.



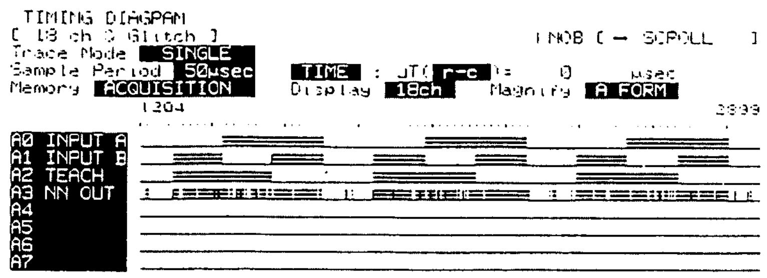
```
TIMING DIAGRAM                                              KNOB [↔ SCROLL ]
[ 18 ch & Glitch ]
Trace Mode REPEAT        Interval                              II =    14
Sample Period 20µsec     TIME : ΔT( r-c )=        0      µsec
Memory ACQUISITION       Display 18ch       Magnify  A FORM
           729                                                        2424
A0 INPUT A
A1 INPUT B
A2 TEACH
A3 NN OUT
A4
A5
A6
A7
```

Fig. 8. Exclusive OR during training.



```
TIMING DIAGRAM                                              KNOB [↔ SCROLL ]
[ 18 ch & Glitch ]
Trace Mode SINGLE
Sample Period 20µsec     TIME : ΔT( r-c )=        0      µsec
Memory ACQUISITION       Display 18ch       Magnify  A FORM
           2298                                                       3993
A0 INPUT A
A1 INPUT B
A2 TEACH
A3 NN OUT
A4
A5
A6
A7
```

Fig. 9. Exclusive OR after training.



```
 TIMING DIAGPAM
[ 18 ch 0 Glitch ]                                  KNOB [ → SCROLL    ]
Trace Mode  SINGLE
Sample Period  50μsec    TIME  : ΔT( r-c )=    0      μsec
Memory  ACQUISITION      Display  18ch    Magnify  A FORM
          1204                                                    2999
```

Fig. 10. Exclusive OR caught in a local minimum.

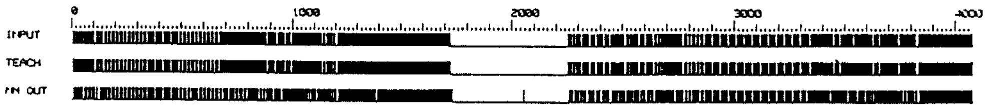


Fig. 11. Result for the function $y = x$.

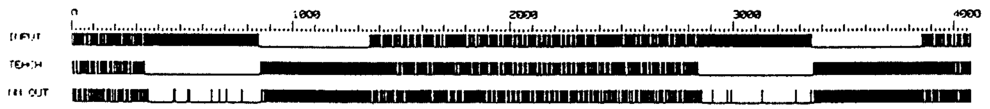
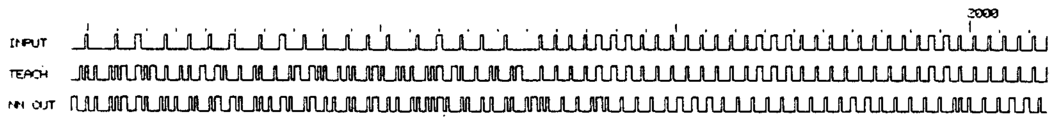


Fig. 12. Result for the function $y = 1 - x$.



5. Conclusion

In this paper, we described the fabrication of a pulse density NN system using the simultaneous perturbation learning rule. The pulse density system converts complicated operations using ordinary analog NN systems into simple digital gate operations. Combined with the simplicity of the simultaneous perturbation learning rule, the pulse stream NN became a fascinating alternative. Furthermore, Y. Maeda and J. C. Spall propose one-measurement type of simultaneous perturbation techniques; the time difference simultaneous perturbation [17] and the one-measurement simultaneous perturbation [18], respectively. These methods require only one forward operation of a NN to obtain the first derivatives of an error function with respect to all weights in the network. This will make hardware implementations more tractable.

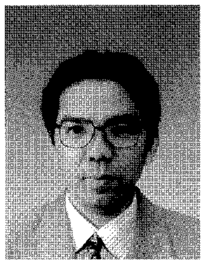
Acknowledgment

A part of this research was financially supported by the Kansai University Special Research Fund, 1997 and the Frontier Sciences Center of Kansai University.

References

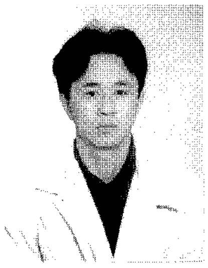
1. B. J. Sheu and J Choi, "Neural information processing and VLSI." Kluwer Academic Publishers, 1995.
2. S. M. Fakhraie and K. C. Smith, "VLSI-Compatible implementation for artificial neural networks." Kluwer Academic Publishers, 1997.
3. M. Jabri and B. Flower, "Weight perturbation : An optimal

architecture and learning technique for analog VLSI feed forward and recurrent multi layer networks.'' *IEEE Trans. on Neural Networks* 3(1), pp. 154–157, 1992.
4. A. J. Montalvo, R. S. Gyurcsik, and J. J. Paulos, ''Toward a general-purpose analog VLSI neural network with on-chip learning.'' *IEEE Trans. on Neural Networks* 8(2), pp. 413–423, 1997.
5. J. C. Spall, ''A schismatic approximation technique for generating maximum likelihood parameter estimates.'' *Proc. of the 1987 American Control Conference* pp. 1161–1167, 1987.
6. J. C. Spall, ''Multivariable schismatic approximation using a simultaneous perturbation gradient approximation.'' *IEEE Trans.* AC-37, pp. 332–341, 1992.
7. J. Alespector, R. Meir, B. Yuhas, A. Jayakumar, and D. Lippe, ''A parallel gradient descent method for learning in analog VLSI neural networks.'' in S. J. Hanson, J. D. Cowan and C. Lee (eds.), *Advances in neural information processing systems 5*. Morgan Kaufmann Publisher, pp. 836–844, 1993.
8. G. Cauwenberghs, ''A fast schismatic error-descent algorithm for supervised learning and optimization.'' in S. J. Hanson, J. D. Cowan and C. Lee (eds.), *Advances in neural information processing systems 5*. Morgan Kaufmann Publisher, pp. 244–251, 1993.
9. Y. Maeda and Y. Kanata, ''Learning rules for recurrent neural networks using perturbation and their application to neuro-control.'' *Trans. of the Institute of Electrical Engineers of Japan* 113-C, pp. 402–408 (in Japanese), 1993.
10. Y. Maeda and Y. Kanata, ''A learning rule of neural networks for neuro-controller.'' *Proc. of the 1995 World Congress of Neural Networks* 2, pp. II-402–II-405, 1995.
11. Y. Maeda and R. J. P. de Figueiredo, ''Learning rules for neuro-controller via simultaneous perturbation.'' *IEEE Trans. on Neural Networks* 8, pp. 1119–1130, 1997.
12. Y. Maeda, H. Hirano, and Y. Kanata, ''A learning rule of neural networks via simultaneous perturbation and its hardware implementation.'' *Neural Networks* 8, pp. 251–259, 1995.
13. G. Cauwenberghs, ''An analog VLSI recurrent neural network learning a continuous-time trajectory.'' *IEEE Trans. on Neural Networks* 7(2), pp. 346–361, 1996.
14. E. I. El-Masry, H. Yang, and M. A. Yakout, ''Implementations of artificial neural networks using current-mode pulse width modulation technique.'' *IEEE Trans. on Neural Networks* 8(3), pp. 532–548, 1997.
15. A. F. Murray, D. D. Corso, and L. Tarssenko, ''Pulse-stream VLSI neural networks mixing analog and digital techniques.'' *IEEE Trans. on Neural Networks* 2(2), pp. 193–204, 1991.
16. H. Eguchi, T. Furuta, H. Horiguchi, and S. Oteki, ''Neuron model utilizing pulse density with learning circuits.'' *IEICE Technical Report* 90, pp. 63–70, (in Japanese) 1990.
17. Y. Maeda, ''Time difference simultaneous perturbation method.'' *ELECTRONICS LETTERS* 32, pp. 1016–1017, 1996.
18. J. C. Spall, ''A one-measurement form of simultaneous perturbation schismatic approximation.'' *Automatica* 33, pp. 109–112, 1997.



Yutaka Maeda He received the B.E., M.E. and D.E. degrees in Electronic Engineering from Osaka Prefecture University in 1979, 1981 and 1990, respectively. He joined Kansai University, Faculty of Engineering in 1987, where he is an Associate Professor. In August to September 1993, he was a Visiting Researcher in Automation Center of Northeastern University, P. R. China. From 1995 to 1996, he was a Visiting Researcher in Electrical and Computer Engineering Department, University of California at Irvine. His research interests include control theory, artificial neural networks and robotics.

Dr. Maeda serves as a reviewer for some journals and international conferences.



Atsushi Nakazawa He graduated from Kansai University with M.E. degrees in 1997. His academic activities were in the area of neural networks implementation. After graduation of the university, he is working for Denso Co. in Japan. He belongs to the Electronic Engineering Department 1.

Yakichi Kanata He was born in Nara, on August 24, 1928. He received the M.E. and D.E. degrees both from Osaka University in 1959 and 1975, respectively. He joined Kansai University in 1959. Since April 1984, he has been a Professor of Kansai University, Faculty of Engineering. He is mainly engaging research on switching regulator and visual information processing systems. He is a member of the Institute of Electrical Engineers of Japan; the Institute of Electronics, Information and Communication Engineers of Japan; the Institute of System, Control and Information Engineers.