Hardware Implementation of a PCA Learning Network by an Asynchronous PDM Digital Circuit

Yuzo Hirai† and Kuninori Nishizawa††
†Institute of Information Sciences and Electronics, University of Tsukuba
††Doctoral Program in Engineering, University of Tsukuba
1-1-1 Ten-nodai, Tsukuba, Ibaraki 305-8573 Japan
E-mail : hirai@is.tsukuba.ac.jp, kuninori@viplab.is.tsukuba.ac.jp

Abstract

We have fabricated a PCA (Principal Component Analysis) learning network in a FPGA (Field Programmable Gate Array) by using an asynchronous PDM (Pulse Density Modulation) digital circuit. The generalized Hebbian algorithm is expressed in a set of ordinary differential equations and the circuits solve them in a fully parallel and continuous manner. The performance of the circuits was tested by a network with two-microphone inputs and two-speaker outputs. By moving a sound source right and left in front of the microphones, the first principal weight vector could continuously track the sound direction in real time.

1 Introduction

A primary object of the hardware implementation of neural networks is to realize real-time operation of neural functions in VLSI chips. Since the mid eighties, many VLSI neural chips and systems have been reported in the literature, e.g. [1, 2]. We also developed a hardware system that consisted of 1,008 neurons fully interconnected via more than one million 7-bit physical synapses [3]. An asynchronous digital circuit was used to implement the neuron circuits. The behavior of each neuron faithfully obeys a nonlinear first-order differential equation, so that the system solves more than one thousand simultaneous differential equations in a fully parallel and continuous manner. The magnitude of neuron output is encoded by a pulse density as our real neurons do. Synaptic weights can be down loaded from a host computer after learning has been finished on it. The processing speed is ten thousand times faster than a latest workstation.

Most of the research on on-chip learning have focused on supervised learning, e.g. [4]. There were a few cases for unsupervised or self-organizing learning networks including analog chips for independent component analysis reported in [5]. In this paper an asynchronous PDM digital circuit is applied to a principal component analysis (PCA) learning network.

PCA is a standard technique widely used for dimension reduction in statistical pattern recognition. The task of PCA is to find a set of eigenvectors whose eigenvalues are the largest among those obtained from the correlation matrix of input data [6]. Oja [7] devised an on-line Hebbian learning algorithm which could find the first principal component of the input without recourse to the correlation matrix. Sanger [8] proposed a generalized Hebbian algorithm (GHA) which could find the first m principal components at the m output neurons in a single-layer feedforward linear network. Kung and Diamantaras [9] also invented a PCA learning network, but their network included an anti-Hebbian learning rule as well as a Hebbian rule [6].

In this paper GHA is chosen as the target because the structure of the learning algorithm, especially its local implementability, is suited for hardware implementation as compared with other candidates. The discrete form of the original GHA is directly converted to a continuous form, since our circuits operate continuously in time. This does not mean to use ordinary differential equations, which will appear in the stability analysis of stochastic difference equations, because they contain a correlation matrix of multivariate signals. In order to check if our circuit can find principal components in real time, a small learning network with two inputs and two outputs was fabricated in a FPGA. It was observed that the circuit could continuously track the correct directions of principal weight vectors in real time.

2 Generalized Hebbian Algorithm

Here we consider a single-layer feedforward network with M inputs and N outputs. The output of the ith neuron at discrete time T is given by

$$V_i(T) = \sum_{j=1}^{M} w_{ij}(T)\xi_j(T) \tag{1}$$

where $w_{ij}(T)$ is a synaptic weight from the jth input to the ith output neuron and $\xi_j(T)$ is the jth input at time T. According to GHA [8] the synaptic weight $w_{ij}(T)$ is updated by a small change as given by

$$\Delta w_{ij}(T) = \eta \left[V_i(T)\xi_j(T) - V_i(T)\sum_{k=1}^{i} w_{ik}(T)V_k(T) \right] \tag{2}$$

where η is the parameter which determines the learning rate.

The prominent feature of GHA is in its local implementability. By rewriting Eq.(2) the following iterative equations can be obtained:

$$\begin{aligned} \Delta w_{ij}(T) &= \eta V_i(T)\mu_{ij}(T), \text{ and} & (3) \\ \mu_{ij}(T) &= \mu_{i-1,j}(T) - V_i(T)w_{ij}(T), & (4) \end{aligned}$$

where $\mu_{0j}(T) = \xi_j(T)$. As seen in the above equations $\mu_{ij}(T)$ can be considered as a modified input for a Hebbian rule, and it can be calculated by using a learning signal $\mu_{i-1,j}(T)$ supplied from its neighbor and a local negataive term $V_i(T)w_{ij}(T)$. Convergence properties of this algorithm were rigorously investigated in [10].

3 Implementation in an Asynchronous PDM Digital Circuit

We employed an asynchronous PDM digital circuit to implement the GHA because (1) by using asynchronous circuits, it is not necessary to supply a common clock to the entire circuits and a large system can be designed easily, (2) by using pulse stream, faithful analog data transmission by a single signal line will be achieved as actual neurons do, which relaxes wiring problem, and (3) by using digital circuits, standard CMOS technologies such as FPGAs can be used and their rapid developments can be incorporated into the design of system. All of these merits have already been verified in the development of our 1,000-neuron system [3].

3.1 Continuous Learning Algorithm

Since in PDM architecture an output from each neuron is expressed as a pulse stream, integration is necessary to recover the analog value from the pulse stream. We invented a PDM digital circuit which could perform leaky integration and showed that it could solve first-order differential equations in a continuous manner. In order to apply GHA to the circuit, we modified the learning equations from Eq.(1) to Eq.(4) in the following differential equations:

$$\begin{aligned} \frac{dw_{ij}(t)}{dt} &= \eta_{ij} V_i(t)\mu_{ij}(t), & (5) \\ \tau_\mu \frac{d\mu_{ij}(t)}{dt} &= -\mu_{ij}(t) + \left(\mu_{i-1,j}(t) - V_i(t)w_{ij}(t)\right), & (6) \end{aligned}$$

where $\mu_{0j}(t) = \xi_j(t)$, and

$$\tau_V \frac{dV_i(t)}{dt} = -V_i(t) + \sum_{j=1}^{N} w_{ij}(t)\xi_j(t), \tag{7}$$

where τ_μ and τ_V are time constants of a learning signal and a neuron output, respectively. The time constant τ_μ must be faster than τ_V because as seen in Eq.(6) the learning signal $\mu_{i-1,j}(t)$ is used in $w_{ij}(t)$ and it should propagate fast enough to ensure the correct interaction between $V_i(t)$ and $\mu_{ij}(t)$ as seen in Eq.(5). Although the number of principal components that the network can find will be limited by this propagation delay, the time constants can be adjusted as described below. This limitation is not explicit in the discrete GHA.

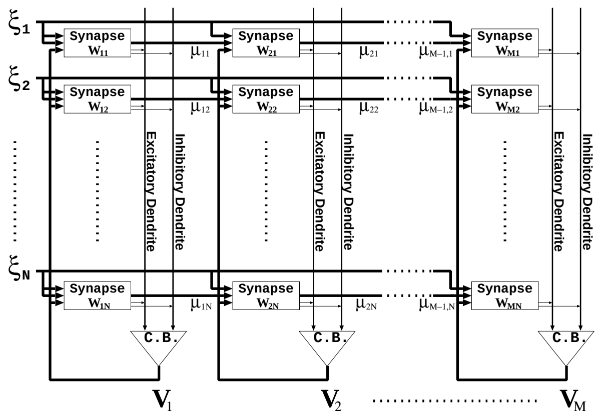

Figure 1: Structure of the network. Details are described in text.

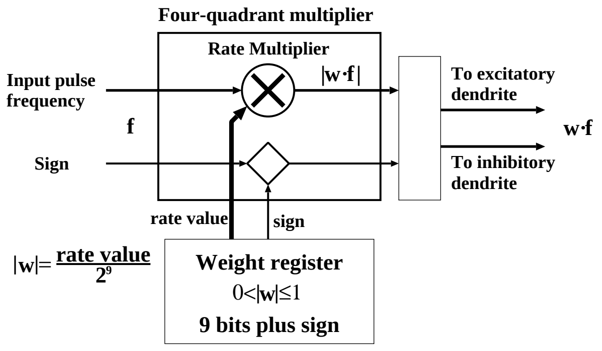

Figure 2: Structure of the synapse circuit. The symbol ⊗ designates a rate multiplier and ◇ designates an exclusive OR circuit.

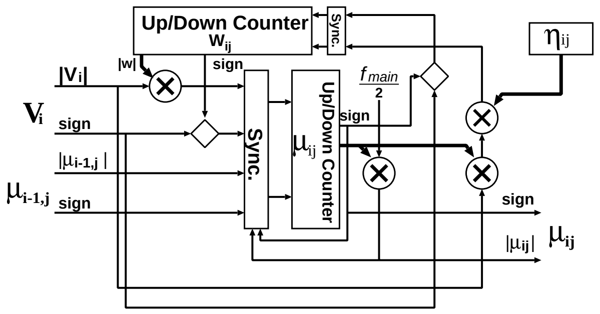

Figure 3: Structure of the circuit which updates synaptic weight. The notations are the same as in Figure 2.

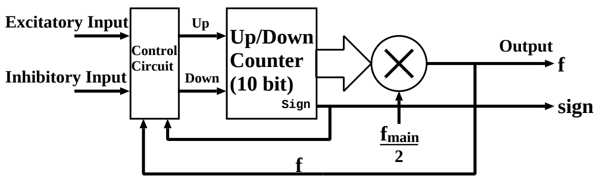

Figure 4: Structure of the cell body circuit. The notations are the same as in Figure 2. The control circuit resolves the conflict between simultaneous up- and down-input to the counter.

3.2 Structure of the Network

The structure of PCA learning network is schematically illustrated in Figure 1. The network consists of three circuits. They are synapse circuit, dendrite circuit, and cell body circuit. The input denoted by ξ_j is fed to each neuron via the synaptic weight denoted by w_{ij}. Learning signal $\mu_{i-1,j}$ that is calculated in the synapse $w_{i-1,j}$ is sent to the next synapse w_{ij} to make the learning signal μ_{ij} as described in Eq.(6). Linear output V_i of each neuron is fed back to all of its synapses to calculate a learning signal in each synapse.

3.2.1 Synapse Circuit

Each synapse circuit consists of two parts. One is to multiply a synaptic weight to the input. This multiplication is carried out by transforming input pulse frequency to the frequency which is proportional to the synaptic weight as shown in Figure 2. This transformation is carried out by a 9-bit rate multiplier and output frequency is given by

$$f_{output} = \frac{\text{rate value}}{2^9} \times f_{input}. \tag{8}$$

Since rate value is smaller than 2^9, the magnitude of the synaptic weight is smaller than 1.

Since GHA uses a linear network, four-quadrant multiplication is necessary. Multiplication between the absolute value of an input and that of a synaptic weight is carried out by a rate multiplier. The result is fed to an excitatory dendrite circuit when the signs of input and weight are the same or is fed to an inhibitory one when both signs are different.

The other part is a learning circuit which consists of four rate multipliers and two up-down counters as shown in Figure 3. It faithfully realizes the computation defined by Eq.(5) and Eq.(6). In this circuit Eq.(6) is solved by the following integral form:

$$\mu_{ij}(t) = \frac{1}{\tau_\mu}\int_0^t \left[-\mu_{ij}(\tau) + (\mu_{i-1,j}(\tau) - V_i(\tau)w_{ij}(\tau))\right] d\tau + \mu_{ij}(0), \tag{9}$$

where $\mu_{ij}(0)$ is the initial value of learning signal.

The integration is carried out as follows. First we find $V_i \cdot w_{ij}$ by four-quadrant multiplication and the result is fed either to up or down input of the up-down counter denoted by μ_{ij} in the figure. The learning signal denoted by $\mu_{i-1,j}$ from the jth synapse of the $(i-1)$th neuron is also fed to either up or down input of the counter according to the sign. The absolute value of the counter is transformed to a pulse stream whose frequency is proportional to it. The negative feedback term $-\mu_{ij}$ in the above equation is realized by feeding the output pulses to the up input of the counter when the content is negative or to the down input when it is positive. The circuit block denoted by *Sync.* in the figure resolves the conflict between simultaneous up- and down-input to the counter.

Since we use 6-bit up-down counter, the time constant τ_μ is given by

$$\tau_\mu = \frac{2^6}{f_{\max}} = \frac{2^6}{10\text{MHz}} = 6.4\ \mu\text{sec}, \tag{10}$$

where $f_{\max}$ is the maximum output frequency.

In realizing Eq.(5) by a digital circuit, a pulse stream of $|V_i|$ is fed to another rate multiplier where a rate value is given by a binary number of $|\mu_{ij}|$ from the up-down counter. In order to multiply a learning constant, the output pulse is fed to another rate multiplier whose rate value is given by the register denoted by η_{ij}. Then, the output pulse from this rate multiplier is fed to either up or down input of the other up-down counter, according to the signs of V_i and μ_{ij}, that is denoted by w_{ij} and that stores the weight. In the following, every η_{ij} is set to $\frac{1}{8}$.

3.3 Dendrite Circuit

A dendrite circuit spatially sums synaptic output pulses by OR gates. Because the pulse stream does not have polarity, spatial summation of excitatory and inhibitory synaptic output pulses must be done separately. Although when more than one pulse come simultaneously, they are counted as one and linear summation cannot be taken place, asynchronous circuits can relax this problem. By driving each neuron by individual clock, output pulses from different synapses tend to be asynchronous especially when their frequencies are low. We have already theoretically and experimentally analyzed the summation characteristics and it has been shown that the linear summation occurs in a wide range of input frequencies [3].

3.4 Cell Body Circuit

In cell body circuit Eq.(7) is solved in the following integral form in the same way as the learning circuit:

$$V_i = \frac{1}{\tau_V}\int_0^t \left[-V_i + \sum_{k=1}^{N} w_{ik}\xi_k\right] d\tau + V_i(0). \tag{11}$$

The structure of the circuit is shown in Figure 4. Since each neuron output must be linear, not only the output pulse stream whose frequency is proportional to the absolute value of the counter, but also the sign signal is sent to the other components of the network.

Since we used a 10-bit up-down counter, the time constant τ_V is 102.4 μsec.

4 Performance of the Network

4.1 Structure of the Network

The circuits described above were designed by VHDL (Very-high-speed-integrated-circuits Hardware Description Language) and were fabricated in a FPGA (Xilinx XC4085XL). In order to evaluate the real-time

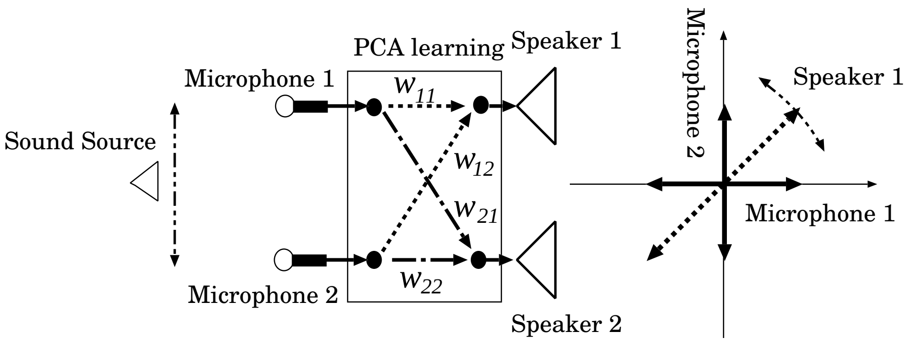

Figure 5: A PCA learning network with two-microphone inputs and two-speaker outputs.

performance of the circuits, a network with two-microphone inputs and two-speaker outputs is created as illustrated in Figure 5. The total number of gates that the circuits used was about 12,000. A 20 MHz common clock drives the cell body circuit of each output neuron and the synapse circuits which are connect to the cell body circuit. Two output neurons are driven by different crystal oscillators so that they operate asynchronously. A sixteen-bit A/D converter, which samples microphone input at 44.1 KHz, is used for each input, but the signal is resampled at 4 KHz and 10 MSBs among sixteen bits are used as the input to the learning network. A sixteen-bit D/A converter is used to produce analog output signal for each speaker. 10 MSBs are used as in the case of A/D conversion.

Since in this case a single sound source is used, a composite vector of the two microphone signals will move along a diagonal line in the two-dimensional input space as shown in the right part of the figure. When the sound source is moved left and right in front of the microphones, the angle of the composite vector will change accordingly. By applying PCA learning to this input space, the eigenvector corresponding to the first principal component will indicate the direction of the diagonal line and will follow the change in the direction caused by the sound source movement. In this case the second principal component will, however, degenerate because variance in the orthogonal direction to the first principal vector is minimal. Therefore, the output sound will always appear at Speaker 1 and will be minimal at Speaker 2. It was shown that the network could find the second principal component in nonsingular cases successfully [11].

4.2 The Performance

An example of the time evolution of four synaptic weights in the network is shown in Figure 6. In this case identical signals are supplied to both inputs. The weights w_{11} and w_{12} constituting the first principal component converged within 10 msec. They consist of a vector with unit length. On the other hand, the weights w_{21} and w_{22} constituting the second principal component were degenerated. They converged to zero weights at a slower rate than the first principal component. Even when the second principal component is not degenerated, the convergence takes a much longer time than that for the first principal component [11].

The loci of weight vectors for the first and the second principal components are shown in Figure 7 when the sound source was moved left and right in front of the microphones. The first and the second principal vectors obtained at the same sound position are designated by the same symbols. The loci on the unit circle are those for the first principal components and those inside the circle are for the second principal components. In all cases the output sound appeared only at Speaker 1.

5 Conclusions

We have designed and fabricated a PCA learning algorithm in a FPGA by using an asynchronous PDM digital circuit. The performance of the circuit was evaluated by a learning network with two inputs and two outputs. It was verified that the circuit could find principal components and could follow changes in the statistical nature of multivariate inputs in real time. We are planning to apply the circuit to high-dimensional

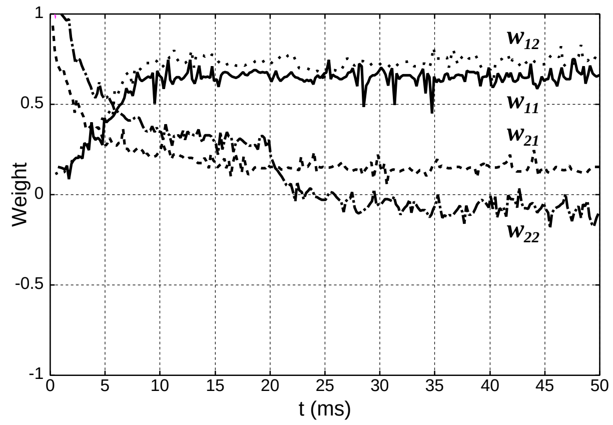

Figure 6: Time evolution of synaptic weights during PCA learning.

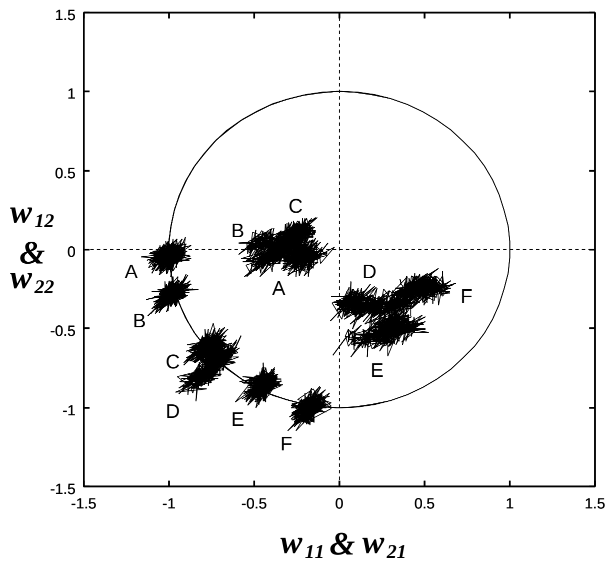

Figure 7: Loci of weight vectors following environmental changes in real time.

input cases and to signal processing functions such as a real-time whitening filter.

References

[1] Mead, C.: "Analog VLSI and Neural Systems." Addison-Wesley Publishing Company, Massachusetts, 1989.

[2] Przytula, K.W. and Prasanna, V.K., Eds.: " Parallel Digital Implementations of Neural Networks." Prentice Hall, New Jersey, 1993.

[3] Hirai, Y.: "A 1,000-Neuron System with One Million 7-bit Physical Interconnections." In Advances in Neural Information Processing Systems 10, ed. Jordan, M.I., Kearns, M.J. and Solla, S.A., pp.705–711, The MIT Press, 1998. Web site: http://www.viplab.is.tsukuba.ac.jp/

[4] G. Cauwenberghs: A learning analog neural network chip with continuous-time recurrent dynamics. In J. D. Cowan, G. Tesauro and J. Alspector, Eds., *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann Publishers, San Mateo, CA, pp.858-865, 1994

[5] Jutten, C. and Herault, J.: "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture." Signal Processing, Vol.24, pp.1–10, 1991.

[6] Haykin, S.: "Neural Networks: A Comprehensive Foundation." 2nd edition, Prentice Hall, New Jersey, 1999.

[7] Oja, E.: "A simplified neuron model as a principal component analyzer." J. Math. Biology, Vol.15, pp.267–273, 1982.

[8] Sanger, T.D.: "Optimal unsupervised learning in a single-layer linear feedforward neural network," Neural Networks, Vol.12, pp.459–473, 1989.

[9] Kung, S.Y. and Diamantaras, K.I.: "A neural network learning algorithm for adaptive principal component extraction (APEX)." Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal processing, Vol.2, pp.861–864, 1990.

[10] Chatterjee, C., Roychowdhury, V.P. and Chong, E.K.P.: "On relative convergence properties of principal component analysis algorithms." IEEE Trans. on Neural Networks, Vol.9, No.2, pp.319–329, 1998.

[11] Nishizawa, K. and Hirai, Y.: "Hardware implementation of PCA neural network." Proceedings of ICONIP'98, pp.85–88, 1998.