Hardware Self-Organizing Map Based on Digital Frequency-Locked Loop and Triangular Neighborhood Function

Hiroomi Hikawa, *Member, IEEE*

***Abstract*— This paper proposes a unique hardware architecture for a self-organizing map (SOM) that mimics the biological brain by using pulse mode operation. In the proposed SOM, vector elements are given as in the form of frequency modulated signals, and digital frequency-locked loops (DFLLs) in neurons handle the computations of the vector elements. The SOM is trained by unsupervised learning, where the winner neuron that has the nearest weight vector is found first. In the proposed SOM, the winner neuron is found by counting cycle slips between the signals that carry input and weight vectors. After the winner neuron is found, weight vectors selected by a neighborhood function are updated toward the input vector. Triangular neighborhood function that is implemented by using an attenuating enable signal for the DFLLs, is employed. To evaluate the proposed SOM and its building components, VHDL simulations and experiments using an FPGA were conducted. Compared to the previous work, the operation speed and learning capability were significantly improved. Novelty of the proposed architecture is it uniquely uses a pulse-based operation that mimics the biological brain, and it was verified that unsupervised learning can be realized with neurons communicating with each other using frequency modulated pulse signals.**

***Index Terms*— Self-organizing map, digital frequency-locked loop, frequency modulated pulse signal, triangular neighborhood function, FPGA.**

I. Introduction

THE self-organizing map (SOM) that was proposed by Kohonen [1] is a special type of artificial neural network (ANN). The SOM, which is trained by an unsupervised learning algorithm, performs a nonlinear mapping from a given high-dimensional input vector space to a lower-dimensional map of neurons, and it has been used to visualize, interpret, and classify large high-dimensional data.

In general, substantial parallelism is found in the algorithms of ANNs including the SOM. Therefore, a parallel hardware architecture is a suitable platform for implementing the ANNs and SOM. Many researchers have been developing VLSI implementations of the neural networks using various techniques, ranging from digital to analog and even optical. Dedicated digital or analog VLSI implementation is crucial for building fast ANNs. The primary disadvantage of analog implementation is low design flexibility even though it can possibly provide higher speed with low hardware cost. On the other hand, digital ANN implementation can take advantage of the benefits of the state-of-the-art VLSI and ULSI techniques [2]–[5]. As for the SOM, various hardware SOMs aiming to speed up the computation have been proposed [6]–[29].

One of the effective platforms to implement the SOM is an array processor system consisting of processing elements (PEs) working in parallel; thus, it can take advantage of the parallelism embedded in the SOM algorithm. Ienne *et al.* [6] implemented the SOM on a massively parallel computing system based on systolic array architecture. Some researchers used single instruction multiple data (SIMD) array processors. One example is the SOM that was proposed by Porrmann *et al.* [7]. The SOM was implemented on a rapid prototyping system with PEs working in a SIMD manner. Hendry *et al.* [8] also exploited the SIMD array accelerator consisting of 256 PEs. Porrmann *et al.* [9] implemented the SOM on a massively parallel computer that included SIMD processors.

A field programmable gate array (FPGA) is a suitable platform for implementing a digital system due to its reconfigurability, and the FPGA has been the most popular platform to implement the SOM. Tamukoh *et al.* [10] proposed a new fast learning algorithm for hardware SOMs, and its massively parallel architecture design was implemented on an FPGA to demonstrate its on-chip learning performance. Peña *et al.* [11] proposed a hardware-friendly SOM architecture. One-dimensional and two-dimensional SOMs were designed based on the proposed architecture, and they were implemented on the FPGA. Manolakos *et al.* [12] proposed an SOM IP core for the FPGA implementation, which was based on systolic array architecture. Ramirez-Agundis *et al.* [13] proposed a modular architecture to implement the SOM. In the proposed architecture, 16 processing units were distributed in a module, and a maximum of 16 modules could be included. By using the proposed architecture, a vector quantizer for real-time video coding was implemented on an FPGA and standard-cell devices.

Lachmair *et al.* [14] used a SIMD array processor system called gNBXe to implement the SOM. The gNBXe consisted of several FPGAs including the PEs. The local controller in

Manuscript received May 8, 2020; revised August 5, 2020 and September 15, 2020; accepted December 10, 2020. Date of publication January 12, 2021; date of current version February 23, 2021. This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP20K11999. This article was recommended by Associate Editor J. Juillard.

The author is with the Graduate School of Science and Engineering, Kansai University, Suita 564-8680, Japan (e-mail: hikawa@kansai-u.ac.jp).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2020.3046795.

Digital Object Identifier 10.1109/TCSI.2020.3046795

each PE translated macro commands from the global controller so that the PE performed the calculations of the SOM's neurons. In Hikawa et al.'s study [15], a massively parallel SOM architecture with a novel neighborhood function was proposed. The improvement of the learning performance was verified by the hardware SOM that was implemented on an FPGA. Huang *et al.* [16] proposed a vector quantizer based on the hardware SOM. The proposed vector quantizer was implemented on an FPGA and was used for high-speed image compression.

Sousa *et al.* [17] investigated different conceptions of FPGA circuits for implementing SOMs. Three models of FPGA implementation (centralized, distributed, and hybrid architectures) were compared to support the system design choices. The centralized model outperformed the other models in terms of chip area occupation and maximum operating frequency. A distributed architecture to implement SOM learning and recall algorithms were proposed by Sousa *et al.* [18]. Since the distributed model does not employ any type of central control unit, redesign process during the configuration of the neural system was simplified.

Appiah *et al.* [19] introduced a tri-state logic SOM called bSOM. The bSOM took binary inputs and maintained tri-state weights. The bSOM was well suited to FPGA implementation, and was applied to character recognition and appearance-based object identification. In [20] a real-time hand sign recognition system based on hardware SOM was proposed. Training of the SOM was carried out off-chip, and recall operation of the SOM was implemented on FPGA. Kim *et al.* [21] investigated an energy-efficient hardware architecture of SOM for electrocardiogram (ECG), and they presented its implementation in 65nm CMOS. Quadrature Amplitude Modulation (QAM) has been applied in many communication systems, and Sousa *et al.* [22] proposed a new FPGA-based SOM to detect 64-QAM symbols. The use of SOM in the Quadrature/In-phase pair detection allowed the continuous adjustment to the QAM constellation with no supervision, and bandwidth could be saved and training data retransmission was no longer necessary.

To implement various network topology, Sun *et al.* [23] proposed a new hardware SOM accelerator. The proposed accelerator circuit supported training of SOM with 3D cube and binary tree type network structures in addition to the traditional 1D- or 2D-array SOM. The proposed design was applied to three applications, and FPGA validation was conducted. The speedup factor ranged from 7.7 to 32.2 compared with software.

A new modular architecture for a SOM called systolic-SOM (SSOM) which is based on the use of a generic model inspired by a systolic movement was proposed by Khalifa *et al.* [24]. The model was formed by two levels of nested parallelism of neurons and connections that provided a distributed set of independent computations between the processing units.

Caradarilli et al. proposed a modified SOM algorithm called all winner-SOM (AW-SOM) [25]. In the AW-SOM, weight update of a neuron was performed without the neighborhood function, which uses the positional relationship between the neuron and winner neuron. Thus comparator tree for the winner search was not required, and its processing speed was almost independent of the number of neurons. However, the topology-preserving nature of the SOM was omited because the topological relations between neurons were not considered during the weight update.

Sousa *et al.* [26] proposed a high performance FPGA-based SOM architecture called SOMprocessor. The processor explored two different computational strategies for increasing the performance. The first improvement was achieved by including multiplexers, which supported alternating processing of neuron sets by the arithmetic circuits. This strategy resulted in a more flexible use of the chip, in which larger networks could be processed in low-density FPGAs. The second improvement was the inclusion of pipeline architecture for the training algorithm so that different parts of the circuit could process data at the same time. The computational acceleration due to this improvement achieved acceleration of 3 to 4 orders of magnitude as compared to CPU executions.

Researchers have been focusing on the improvement of scalability of the hardware SOM. A scalable and adaptable hardware SOM architecture was proposed by Abadi *et al.* [27], [28]. The scalability of the SOM was achieved by using the network-on-a-chip (NoC) communication approach, and the winner search operation was carried out in a systolic manner distributed all over the network. The proposed SOM architecture was designed for the FPGA implementation and was validated through simulation where the SOM was applied to image compression. Another hardware SOM architecture aiming for scalability has been proposed [29]. By using a nested structure for the SOM architecture, it became easier to scale up the SOM.

Operation of the SOM is divided into two phases: learning and recall. In the learning process of SOM, a winner neuron that has the nearest weight vector to input vector is searched for. This is called a winner-take-all (WTA) computation, which performs classification in the recall phase. The weight vectors of the winner neuron and its neighborhood neurons are updated so that their weight vectors are closer to the input vector. The neighborhood neurons are selected by a neighborhood function that determines the amount of the vector update.

Various hardware WTA computation circuits have been realized. One of the WTA circuits is MAXNET [30]. In MAXNET, neurons in the network mutually inhibited each other while activating themselves, resulting in only one neuron being chosen as the winner. Lazzaro et al. proposed a CMOS WTA circuit, where signals were represented as analog current [31]. Oster et al. examined analytically the ability of a spike-based WTA network [32]. Other examples of spiking WTANNs with temporal coding have been reported [33]- [36]. Regarding digitally implemented WTA, a bit-serial parallel minimum search circuit was being reported in [1]. WTA implementation in hardware based on a binary-tree search algorithm is popular [37], in which a global WTA circuit collects vector distance data from all neurons, and the winner was selected by tournament selections.

A very important feature of the SOM is its topology-preserving nature, where two adjacent vectors in the input vector space are mapped onto adjacent neurons

on the map. This topology-preserving nature is realized by the weight update with the neighborhood function. Thus, the neighborhood function affects the performance of the SOM in the recall phase. In the original algorithm, a Gaussian neighborhood function was used, but the Gaussian neighborhood function is not suitable for the hardware implementation because of its high hardware cost. Thus, most of the hardware SOMs used simplified neighborhood functions, like a square step function [8]. To reduce the hardware cost of the function, Hikawa *et al.* [15] used an internal bus to broadcast Precalculated neighborhood function values to all neurons. A widely used simplification applied to the neighborhood function is to restrict the values of the function to negative powers of two so that the multiplication is replaced by shift operation. This simplification was used in many hardware SOMs [8], [9], [11]–[13], [29].

Dlugosz *et al.* [38], [39] proposed new neighborhood mechanisms that were based on a triangular neighborhood function. The effect of their triangular neighborhood function was studied by Hspice simulation [40]. It was revealed that the triangular function can be used instead of the Gaussian neighborhood function, and the sound performance of the SOM was confirmed even though the signal resolution of the neighborhood function was low. However, the triangular neighborhood function required multipliers that are expensive in terms of hardware cost.

ANNs are computer systems based on or inspired by the biological brain; conversely, neural networks can be used as brain models. One of the important objectives in brain modeling is to explain how the organizational order emerges by itself in the various brain maps. Kohonen [41] demonstrated that the SOM has similar self-ordering to that found in the biological brain. In biological neural systems, information is conveyed by electric pulses. Pulse-mode neural netowrk hardware architectures have been proposed [42]- [46]. These hardware models used pulse density, pulse width, firing rate, or frequency to represent the signal level and used stochastic computations. The advantage of the pulse-mode architecture is that the computing elements can be realized by a stochastic arithmetic circuit that is smaller than the conventional arithmetic circuit.

In recent years, experimental evidence indicates that many biological neural systems use the timing of single spikes to encode and process information. This method, known as temporal coding, is considered to be the coding mechanism in biological neural systems. These spiking neuron models have become increasingly popular as they mimic the spiking nature of real neurons. The Hodgkin-Huxley model [47] is one of the first detailed spiking neuron models. A simple model is the leaky-integrate-and-fire (LIF) model [48], which is computationally cheaper. In hardware spiking neural networks with temporal coding, coincidence or synchrony detection plays an important role in their neural information processing.

An interesting approach to implement a hardware neural network uses pulse signal synchronization. Hoppensteadt [49] proposed a novel associative memory architecture using a phase-locked loop (PLL), which could memorize and reproduce complex oscillatory patterns. The hardware SOM that was proposed in [50] used a phase-modulated signal to convey neuron signals, and a digital phase-locked loop (DPLL) was used to handle the neuron's computation. This DPLL-SOM's scalability was improved by introducing a new WTA circuit in [51]. This SOM took advantage of the similarity between the operations of the DPLL and the computations of the SOM's neurons. The WTA operation was implemented by using phase comparators distributed among all neurons, which made it easier to increase the number of neurons compared to the global WTA approach with the binary-tree search.

The problem in using DPLL is that the computing precision was proportional to the clock frequency because the phases of the carrier signals were modulated by clock signal. For example, the frequency of the clock must be 2^{16} times higher than that of the carrier signal if 16-bit precision was required to represent vector element values.

In Hikawa's study in [52], digital frequency-locked loop (DFLL)-based WTANN with a frequency-modulated signal, was proposed to cope with the problem of the DPLL-based SOM. The DFLL controls the output signal so that its frequency matches that of the input signal. In the system, vector values were transmitted and processed by using carrier signals. The frequency of the carrier signals represented the vector elements, and the DFLL processed the vectors by adjusting the frequencies of the carrier signals. The DFLL used a direct digital frequency synthesizer (DDS) as its local oscillator. A strong point of DDS is that its frequency precision is governed by the bit-length of its internal register. Therefore, the use of the DFLL and frequency modulated signal could provide higher precision computation with lower clock frequency.

The DFLL-based WTANN circuit proposed in [52] was improved in [53] in terms of its circuit size, and it was applied to develop the DFLL-based hardware SOM [54]. Winner search was carried out by all neurons competitively by using frequency comparators distributed among neurons. This distributed winner search architecture made it easier to increase the number of neurons.

The problems of this SOM were its neighborhood function and operation speed. In this SOM, amount of the weight update carried out by the DFLL was proportional to frequency of a weight update pulse fed to the DFLL. The winner neuron generated the update pulse, which was propagated between the neurons with decreasing frequency. As a result, the function that was similar to the original neighborhood function was realized. However, the update pulse signal propagation circuit used in the SOM was rather complicated, which made it difficult to raise its clock frequency. Since the performance of the hardware SOM is proportional to the clock frequency, simple update pulse propagation circuit that can work with the higher clock frequency, is desired.

This paper proposes a new DFLL-based hardware SOM architecture that mimics the biological brain by using pulse mode operation. Like the previous work, the proposed SOM uses the DFLLs to process vector elements, and the frequency-modulated pulses convey the vector elements.

To improve the learning performance and speed of the SOM, the triangular neighborhood function is employed. As mentioned in the previous page, Dlugosz et al. well studied and

proved that the triangular neighborhood function can provide quite good learning performance [38], [39]. In the straightforward implementation of the triangular function, it requires, in terms of hardware cost, very expensive multipliers and subtraction circuits. Like the pulse operation carried out for the main computation of the SOM, the neighborhood function is implemented by pulse signal. A counter is the only major component of the proposed circuit for the triangular neighborhood function, and no multiplier or subtraction circuit is needed.

In the proposed SOM, vector distance computation is carried out by frequency detectors because all vector elements are represented as frequencies of pulse signals. Each DFLL then updates the weight vector element according to the frequency difference, and changes the frequency of its internal signal. While the enable signal is applied, the DFLL updates its vector element to bring the frequency of the internal signal closer to that of the input signal. Therefore, the amount of frequency change can be controlled by the period during which the enable signal is active. Using this characteristic of DFLL, the triangular neighborhood function is implemented by attenuating the duration of the enable signal.

The proposed components are described with VHDL, and their performances are examined by simulations. Then, the proposed SOM is implemented on an FPGA, and its on-chip learning performance is examined by experiments. As discussed before, information is conveyed by electric pulses in biological neural systems. The novelty of the proposed SOM is that it mimics the biological brain in that all information is conveyed by pulse signals and the SOM algorithm is implemented by pulse-based computation. The use of DFLL allows the neurons in the SOM to learn and perform the nonlinear mapping by communicating with each other via frequency-modulated pulse signals without a conventional numerical computation circuit.

The remainder of this paper is organized as follows. Section II describes the SOM algorithm, and the details of the proposed SOM architecture are discussed in Section III. The results of the simulation and experiments are presented in Section IV, followed by the conclusions in Section V.

II. Self-Organizing Map

The SOM consists of neurons that are usually placed in a two dimensional grid. Every neuron-i includes a D-dimensional vector $\vec{m}_i$ that is called the weight vector.

$$\vec{m}_i = \{\mu_{i,0}, \mu_{i,1}, \ldots \mu_{i,j}, \ldots \mu_{i,D-1}\} \in \Re^D \tag{1}$$

In the initial learning phase, the map is trained with a set of training vectors. The learning phase starts with an appropriate initialization of the weight vectors. Subsequently, the training vectors, $\vec{x} \in \Re^D$, are fed to the SOM in multiple iterations.

$$\vec{x} = \{\xi_0, \xi_1, \ldots \xi_j, \ldots \xi_{D-1}\} \in \Re^D \tag{2}$$

For each input vector, all neurons calculate the distances of their weight vectors to the input vector. Then, the winner neuron-C that has the closest weight vector to the input vector is determined from the vector distances of all neurons.

$$C = \arg\min_i \{||\vec{x} - \vec{m}_i||\} \tag{3}$$

In Kohonen's study [1], the Euclidean metric was used as the vector distance.

$$||\vec{x} - \vec{m}_i|| = \sqrt{(\xi_0 - \mu_{i,0})^2 + \cdots + (\xi_{D-1} - \mu_{i,D-1})^2} \tag{4}$$

However, most hardware SOMs have used Manhattan metric d_i instead of the Euclidean distance to reduce the hardware cost.

$$d_i = \sum_{j=0}^{D-1} |\xi_j - \mu_{ij}| \tag{5}$$

Since no squaring or square root circuit is required, the silicon area saving using the Manhattan metric is significant. After the winner neuron is determined, the weight vectors of the neurons in the winner's neighborhood are updated toward the input vector.

$$\vec{m}_i(t+1) = \vec{m}_i(t) + h_{ci}\{\vec{x}(t) - \vec{m}_i(t)\ \} \tag{6}$$

The original neighborhood function h_{ci} is defined as follows [1]:

$$h_{ci} = \alpha(t) \exp\left(-\frac{||\vec{r_C} - \vec{r_i}||}{2\sigma^2(t)}\right) \tag{7}$$

where, $\vec{r_C} \in \Re^2$ and $\vec{r_i} \in \Re^2$ are position vectors of the winner neuron-C and neuron-i. $\alpha(t)$, $\sigma(t)$ are learning rate and neighborhood radius, respectively. Because its hardware cost is very high, the neighborhood function in (7) is not suitable for the hardware implementation, and the hardware SOMs in the literature used simplified neighborhood functions, such as a negative power of two function [9], [11]–[13], square step function [8], and triangular neighborhood function [38], [39].

After the learning phase, the weights of the map are retained and are used in the recall mode, where only the winner search is carried out.

III. Hardware Self-Organizing Map Based on Frequency-Modulated Signals

A. Neuron

The proposed SOM consists of neurons, that are placed in lattice. Block diagram of the neuron used in the proposed SOM is shown in Figure 1. The neurons are placed in a two-dimensional grid. The input to the neuron consists of D signals ($X_0 - X_{D-1}$) that are frequency-modulated signals carrying the vector elements $\xi_0 - \xi_{D-1}$ given in (2). Thus the frequencies of the signals X_0, X_1, and X_2 represent ξ_0, ξ_1, and ξ_2, respectively. Each neuron contains an update controller, DFLLs, and a winner search circuit that provides the WTA function. Each DFLL processes one of the vector elements. As is discussed in Section III-B, DFLLs include internal signals M's, which are the frequency modulated carrier signals carrying the weight vector elements $\mu_{i,0} - \mu_{i,D-1}$ in (1).

Signals PN_I, PE_I, PE_O, PS_O, PW_I, and PW_O give priority among neurons in the winner search. Detail of these signals is explained in Section III-C. Signals UN_I, UN_O,

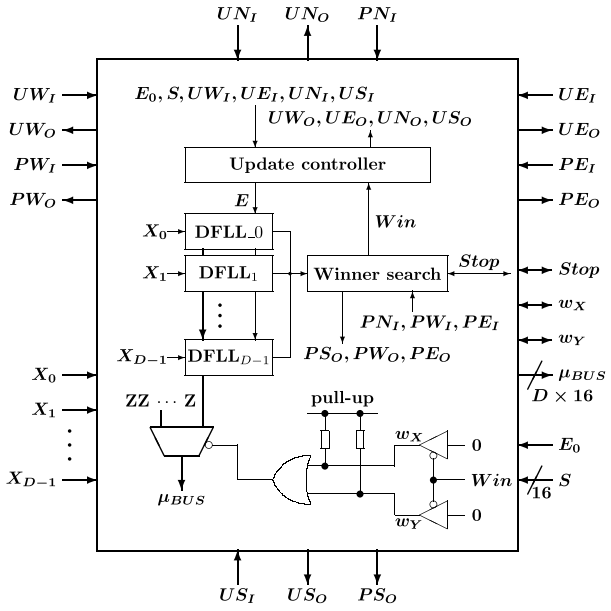

Fig. 1. Neuron used in the proposed SOM.

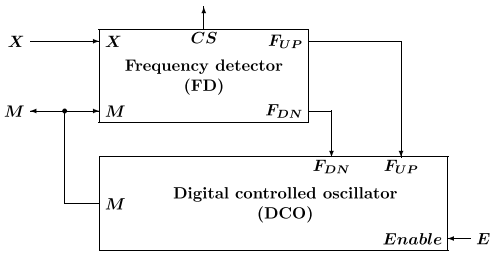

Fig. 2. Digital frequency-locked loop.

$UE_I, UE_O, US_I, US_O, UW_I$, and UW_O carry update signals that control amount of the weight update, and they realize the triangular neighborhood function. The detail is discussed in Section III-D. Since SOM applications use winner neuron information and its weight vector, winner signals w_X, w_Y and a weight vector bus called μ_{BUS} are provided for this purpose. Detail of these signals is explained in Section III-C.

B. Digital Frequency-Locked Loop

Fig. 2 shows the DFLL consisting of a frequency detector (FD) and a digitally controlled oscillator (DCO). The FD detects the frequency difference between two signals X and M. The DFLL controls its DCO so that the output frequency of M matches that of the input signal X. This operation is very similar to the learning process of the SOM that is given by (6). The DFLL used in the previous study [54] included a phase control circuit so that the signal of the DFLL was synchronized in both phase and frequency, but the DFLL in this work is simplified to have no phase control.

A block diagram of the FD is shown in Fig. 3 (A), which is a modified version of the FD used in [54]. Since phase synchronization is not performed, only the frequency error is detected. F_{UP} and F_{DN} are frequency control signals fed to

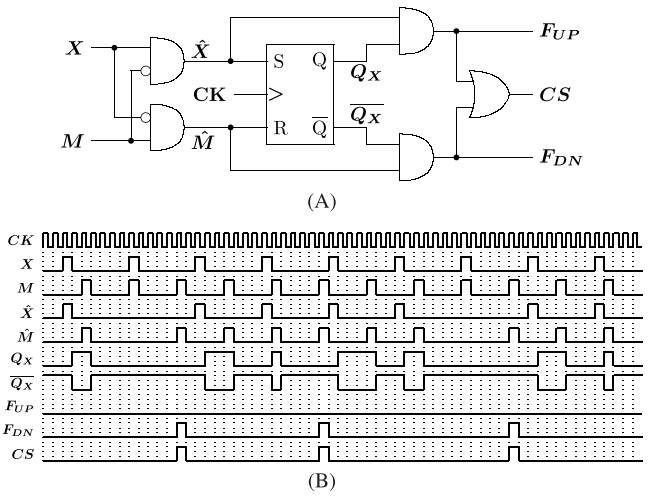

Fig. 3. Frequency detector (FD), (A) block diagram and (B) signal example.

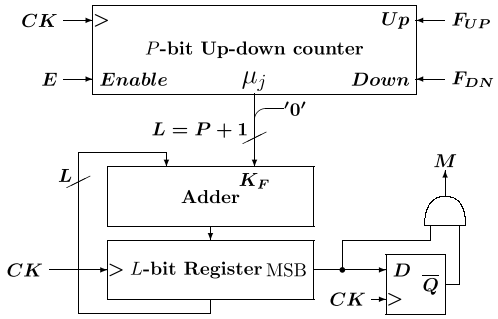

Fig. 4. Digital controlled oscillator.

the DCO. The former is a signal that raises the frequency of the output signal and the latter lowers the frequency.

An example of the FD signals is shown in Fig. 3 (B). In this example, the frequency of the internal signal M is higher than that of input signal X D The number of cycles of X is 9 while M has 12 cycles; thus, 3 cycle slips occur, which yields 3 cycle slip (CS) pulses. The CS pulse is used as an F_{UP} or F_{DN} signal, and 3 pulses are outputted as an F_{DN} signal in this example. This CS pulse is used in the winner search circuit.

As shown in Fig. 4, the DCO is made of an up-down counter and a direct digital synthesizer (DDS) that generates the internal signal M. The DDS is made of an adder, a register, a flip-flop and an AND gate.

The P-bit up-down counter holds the weight vector element μ_j. By concatenating 0 with μ_j, P-bit μ_j is extended to L-bit, apparently, $L = P + 1$. The extended μ_j is fed to the adder, and the μ_j value is used as a frequency control word for the DDS. The frequency of the DDS output signal M is given by the following equation:

$$f_M = \frac{\mu_j}{2^L} \cdot f_{ck} \tag{8}$$

As this equation shows, the frequency of the signal M is modulated by μ_j, and the signal M works as a carrier signal for μ_j value. Note that the frequency resolution of the DDS is defined by L, which is the bit-length of the register.

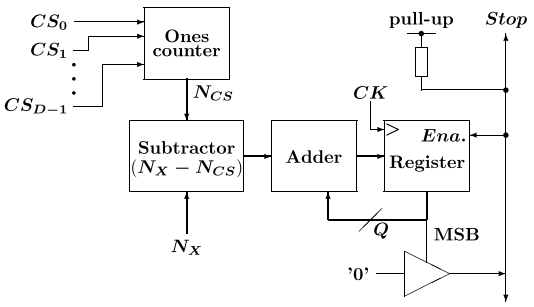

Fig. 5. Winner search circuit.

The contents of the up-down counter μ_j is adjusted by F_{UP} and F_{DN} signals.

$$\mu_j = \begin{cases} \mu_j + 1 & \text{if } F_{UP} = 1 \\ \mu_j - 1 & \text{if } F_{DN} = 1 \end{cases} \tag{9}$$

The change of μ_j is reflected in the frequency change of output signal M. Note that this frequency control is carried out while the enable signal E is activated, and this signal is used to implement the triangular neighborhood function.

C. Winner Search Circuit

A common approach used in hardware SOM is global winner search which collects vector distance data from all neurons to find the smallest one. To do so, the global winner search circuit needs to communicate with all neurons, which requires communication links with a wide bandwidth. When increasing the number of neurons, the global search module must be redesigned or reconfigured, which reduces the scalability of the hardware SOM.

As mentioned before, the proposed winner search circuit uses the cycle slip signal (CS) from the FD. Fig. 5 outlines the winner search circuit. The circuit measures the frequency difference between X and M by counting the CS pulses. The number of CS pulses (N_{CS}) simultaneously given from the DFLLs is counted by a ones-counter. Since the frequency of the cycle slip is proportional to the frequency difference between X_i and M_i, the frequency of the CS signal represents the vector distance. From (8), the average N_{CS} per clock cycle is given by:

$$N_{CS} = \sum_{i=0}^{D-1} |\frac{\xi_j}{2^L} - \frac{\mu_j}{2^L}| \tag{10}$$

Then, $N_X - N_{CS}$ is accumulated in the Q-bit register, where N_X is a constant value. When the MSB of a neuron's register rises, the neuron activates the *Stop* signal by making it to 0, which is broadcast to all other neurons to end the winner search. The value of N_X is chosen so that the accumulation of the register does not stop. Therefore $N_X \geq D$.

The fewer cycle slips yield the quicker accumulation in the register. As mentioned above, frequency of the cycle slip is proportional to the frequency difference between X and M. Thus, neuron that activates the *Stop* signal is the one that has the smallest frequency difference than other neurons, i.e., the neuron with the closest weight vector to input vector. Resulting that the neuron that activates the *Stop* signal, is the winner neuron. In this way, the winner neuron is determined by the speed competition among the neurons that can activate the *Stop* signal.

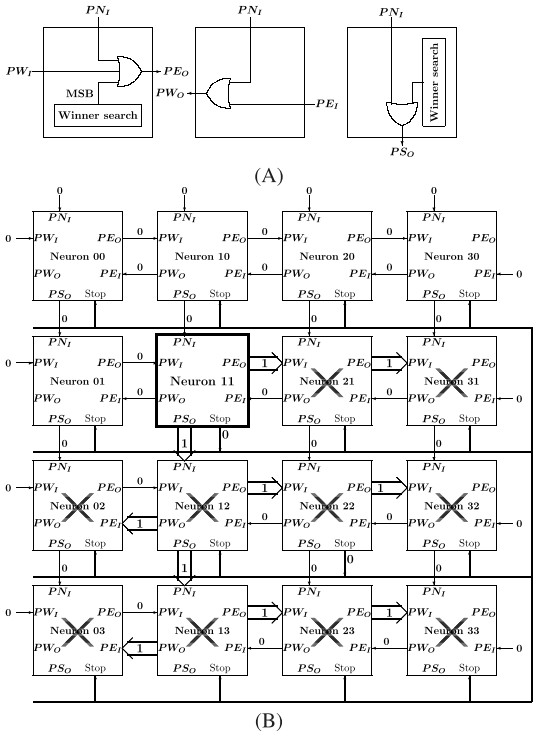

Fig. 6. Priority signal (A) generation and path in the neuron, and (B) example of signal propagation.

The normalized response time N_{win} that is the number of clocks for the winner search circuit to activate the *Stop* signal is calculated as follows:

$$\begin{aligned} N_{win} &= \frac{2^{Q-1}}{N_X - N_{CS}} \\ &= \frac{2^L \cdot 2^{Q-1}}{2^L \cdot N_X - \sum_{j=0}^{D-1} |\xi_j - \mu_j|} \end{aligned} \tag{11}$$

However, multiple neurons can activate the *Stop* signal simultaneously, and they all try to become the winner. To prevent the situation where multiple neurons become winners, priority ranking is assigned to the neurons by propagating the priority signal through PE, PW, PN, and PS. Winner neuron candidates that activate their *Stop* signals send the priority signal to neurons in their East and South. Neurons receiving the priority signal are not allowed to become the winner, and they forward the signal as shown in Fig. 6 (A). These priority signals give the neuron placed in the Northwest corner the highest priority. In the end, the candidate neuron that does not receive the priority signal becomes the winner. In the example shown in Fig. 6 (B), neuron 11 is the winner, and all neurons placed East and South of the winner (marked with $\times$) cannot become the winner because they receive the priority signals. In this example, Neuron 22 also activates the *Stop* signal, but it cannot be the winner because it receives the priority signal.

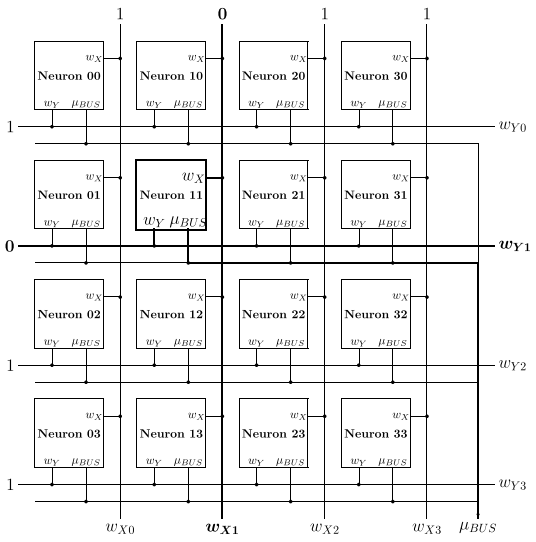

Fig. 7. w_X, w_Y, and μ_{BUS} signals.

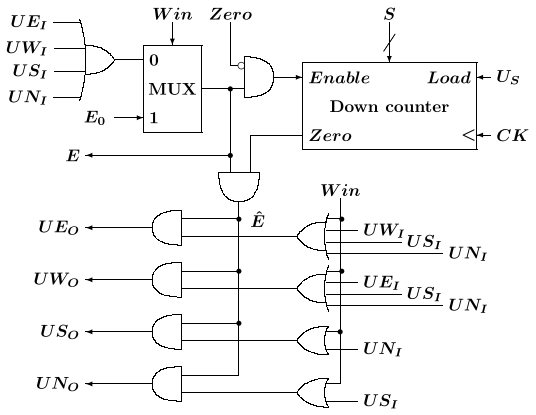

Fig. 8. Update controller.

When a neuron placed at (X, Y) wins, its winner search circuit makes $w_X = w_Y = 0$ to notify the winner neuron's position. In the SOM, these signals are connected as shown in Fig. 7. Neurons in the same row share the same w_Y signal, and neurons in the same column share the same w_X signal. Thus, the winner neuron can be identified by scanning the w_X and w_Y lines. In the example in Fig. 7, it is indicated that neuron-11 is the winner because $w_{X1} = 0$ and $w_{Y1} = 0$.

Meanwhile, the winner neuron outputs its weight vector to the μ_{BUS}. In Fig. 7, weight vector of the neuron-11 is outputted through μ_{BUS}. Signals w_X and w_Y are bidirectional and also function as an input signals that force a specific neuron to be winner. By setting $w_X = w_Y = 0$ from outside the SOM, the weight vector of the neuron placed at (X, Y) can be read through the μ_{BUS}.

D. Triangular Neighborhood Function

After the winner search, the weight vectors of the winner and its neighborhood neurons are adjusted so that they are closer to the input vector. As shown in (6) and (7), the amount of change for the winner neuron is the largest, and it decreases as the distance to the winner neuron increases. This weight update method provides topology-preserving mapping that is one of the most important features of SOM.

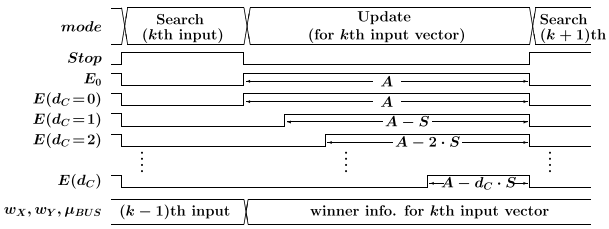

Fig. 9. Enable signals and d_C.

The amount of frequency change carried out by the DFLL is controlled by the duration of the enable signal ($E = 1$) applied to the up-down counter in the DCO. The longer the counter is enabled, the closer the frequency of the DCO signal is made to that of the input signal. Therefore, the weight update that is similar to (6) can be realized by changing the duration of the enable signal. This enable signal control and its distribution are handled by the update controller shown in Fig. 8.

An example of the E signal is shown in Fig. 9. When the $Stop$ signal is activated ($Stop = 0$), the winner search terminates, and resulting winner information (w_X, w_Y, and μ_{BUS}) is updated. Then operation mode switches to Update mode. The winner neuron uses a base enable signal E_0 as its enable signal E, and the duration of E_0 is represented by A. Meanwhile, by using a down counter, the update controller generates the $\hat{E}$ signal. The initial value for the down counter is S, and it is counted down. Once the counter reaches 0, it stops the count and $\hat{E}$ is set to 1, resulting in the duration of $\hat{E} = 1$ being shortened by S from A. The $\hat{E}$ is fed to adjacent neurons through output ports (UN_O, US_O, UW_O, and UE_O). Non-winner neurons receive the signal from one of its adjacent neurons via input ports (UN_I, UN_I, UW_I, and UE_I) and use the signal as their enable signal E. These neurons further shorten the duration of $E = 1$ by S and forward the signal to the neighboring neurons.

The update signal paths in the neuron are summarized in Fig. 10 (A). An example of the update signal propagation is shown in Fig. 10 (B), in which neuron 11 is the winner. The duration of $E = Enable = 1$ is made shorter as the distance d_C to the winner neuron increases, as shown in Fig. 9. Since the duration of $E = 1$ is never less than 0, the duration is expressed as $\max(\ A - S \cdot d_C,\ 0\ )$. As Fig. 10 (B) shows, d_C is the Manhattan distance between the winner neuron placed at (m, n) and a neuron at (p, q).

$$d_C = |p - m| + |q - n| \tag{12}$$

Since the amount of update carried out by the DFLL is proportional to the duration of $E = 1$, the change of $\mu_{i,j}$ per single clock cycle is given by the following equation:

$$\delta\mu_{i,j} = \frac{\xi_j - \mu_{i,j}}{2^L} \cdot \max(\ A - S \cdot d_C,\ 0\ ) \tag{13}$$

This equation shows that the proposed neighborhood function has the same characteristic as the original function, i.e., the amount of weight vector updating performed in the winner neuron is the largest and is reduced as the distance to the

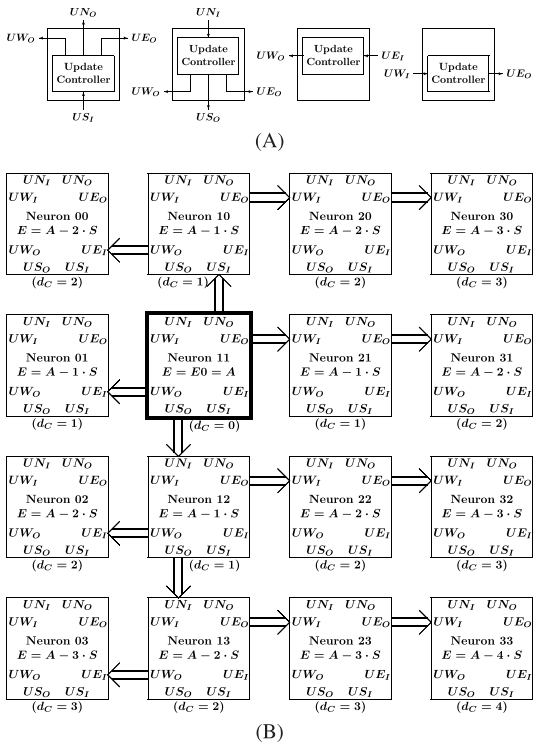

Fig. 10. Update signal (A) generation and path in neurons, and (B) example of signal propagation.

winner increases. A major difference of the proposed neighborhood function from the original one is that the amount of the update linearly decreases as d_C increases, and the proposed function is triangular shaped.

Note that A corresponds to the learning rate α in (7). From (13), if d_C is bigger than A/S, no weight update is carried out. Thus, the radius of the neighborhood of the proposed neighborhood function is A/S. In the original SOM algorithm, magnitude α and radius σ of (7) are decreased as the learning progresses. Similarly A is made smaller while S is enlarged to reduce the radius as the learning progresses.

IV. Experiments

The building components discussed in the previous section and a hardware SOM containing 16 × 16 neurons were described by VHDL. Then, the operations of the components were tested by simulation, and the SOM was implemented on an FPGA to investigate its on-chip learning as well as its circuit properties.

A. VHDL Simulation

The operations of the proposed winner search and neighborhood function circuits were examined by VHDL simulations.

1) Winner Search Circuit: Fig. 11 shows the relation between the register transition of the winner search circuit versus vector distance d_i. This simulation was carried out with $D = 3$, $L = 17$, $N_X = 3$, and $Q = 13,\ 16$. The figure shows that the slopes of the transitions are proportional to the vector distance d_i, and the shorter the d_i, the quicker the register reaches 4096 (2^{12}) or 32768 (2^{15}). Therefore, the neuron

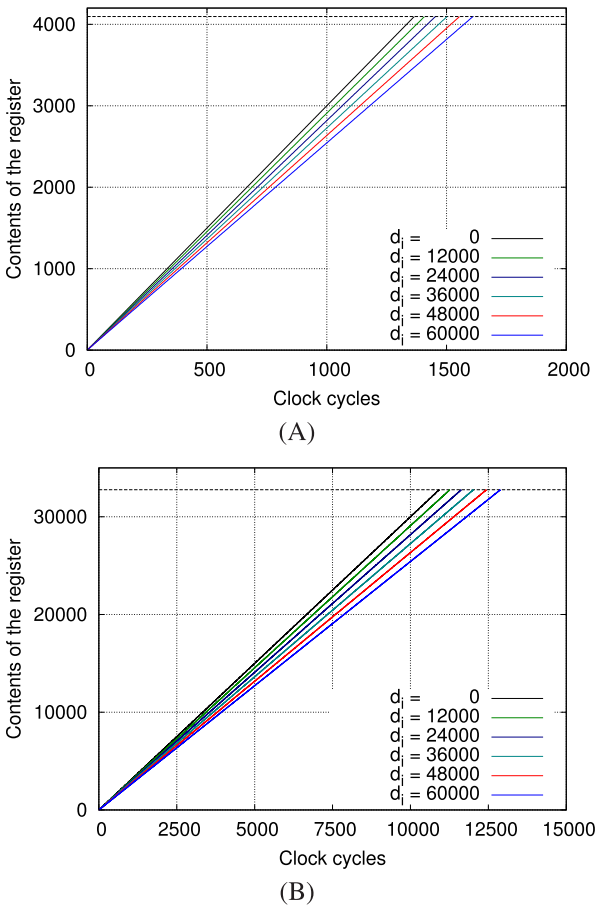

Fig. 11. Transitions of registers in the winner search circuit during the winner search, (A) $Q = 13$, (B) $Q = 16$.

with the shortest d_i activates the *Stop* signal and becomes the winner.

The relation between the response time N_{win} and the vector distance d_i is shown in Fig. 12. The theoretical N_{win} defined in (11) is also plotted in the figure, and the simulation results agree well with the theoretical values. Since N_{win} reflects the vector distance correctly, it was verified that the proposed circuit worked as expected.

2) Neighborhood Function: The neighborhood function was examined by using 16 neurons connected in a row. All weight vector elements of the neurons were initialized to have $\vec{m} = (\mu_0, \mu_1) = (32768, 32768)$, and input vector $\vec{x} = (\xi_0, \xi_1) = (65535, 0)$ was fed to the neurons. The neuron placed at the left end was set to be the winner neuron, and all weight vector values after the training of the period of 2^{18} clock cycles were measured. The amounts of the weight changes represent the neighborhood function.

From the theoretical update amount expressed in (13), $\Delta\mu_j$ that is the actual change of the weight vector element after the 2^{18} clock cycles is computed as:

$$\Delta\mu_j = \delta\mu_j \cdot 2^{18} \tag{14}$$

Fig. 13 shows the simulation results, in which μ_0's are made bigger toward 65536 while μ_1's get smaller toward 0. Fig. 13 (A) shows that the slope of the neighborhood function can be controlled by the parameter S, and another

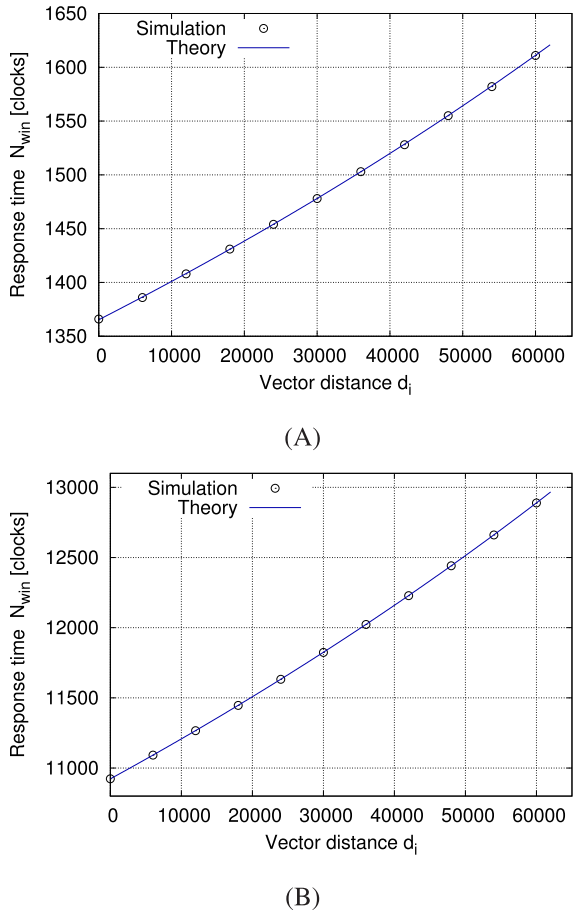

Fig. 12. Response time of the winner search circuit vs. vector distance, (A) $Q = 13$, (B) $Q = 16$.

TABLE I
CIRCUIT SIZE

No. of Slice registers	35938
No. of Slice LUTs	95699
No. of occuied Slices	28113
No. of LUT Flip Flop pairs used	95747

parameter A can change the magnitude of the function, as shown in Fig. 13 (B). The simulation results agree well with the theoretical value, showing that the triangular neighborhood function was realized by the proposed circuit.

B. FPGA Implementation

The proposed SOM was implemented on an FPGA, and the experiment was conducted to investigate the on-chip learning of the proposed hardware SOM. Fig. 14 shows the experimental system implemented on an FPGA (Xilinx Virtex-6 / XC6VSX315TFF1759-2). The system included the hardware SOM, training controller, memory, and frequency modulators. The hardware SOM consisted of the neurons examined by the VHDL simulations in the previous section, and the training controller supervised the training of the SOM.

The implemented SOM was configured as follows:

- the number of neurons was 256 (16×16),
- the dimension of the vectors was $D = 3$,
- the precision of ξ_i and $\mu_{i,j}$ was $P = 16$,

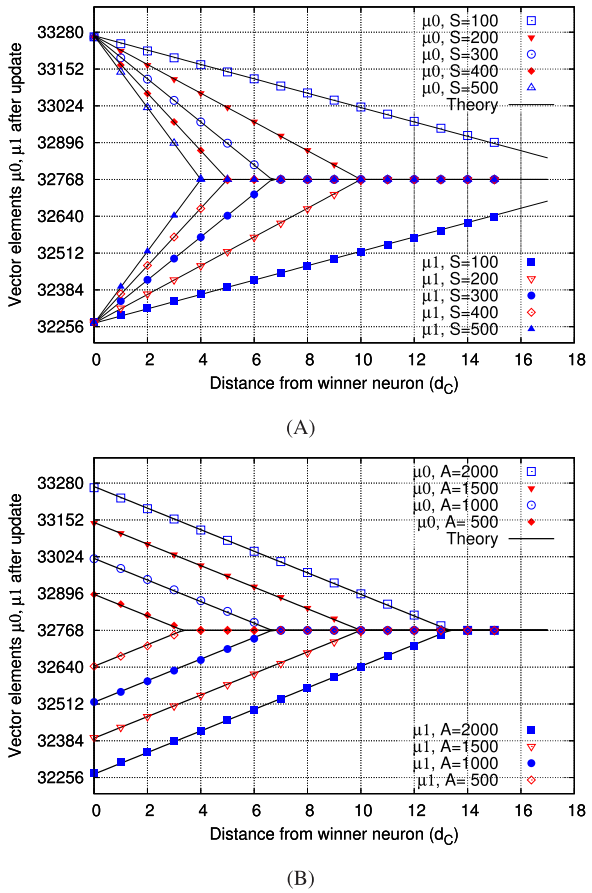

Fig. 13. Amount of frequency shift change vs. distance winner neuron, (A) A=2000, (B) S=150.

TABLE II
PERFORMANCE OF THE PROPOSED SOM (f_{CK} = 67.0 MHz)

	Number of clocks ($\times 10^6$)	Performance (MCUPS)
Random	3597.1	15.00
Triangle	3595.1	15.01
16x16mesh	3591.4	15.02
Donut	3593.0	15.02
Butterfly	3591.4	15.02

- the bit-length of the register in the DCO was $L = 17$,
- the constant value of the winner search circuit was $N_X = 3 (= D)$,
- the register size of the winner search circuit was $Q = 12$,
- the number of training iterations was 256 ($H = 0 - 255$), where H is the epoch count,
- using H as its argument, the neighborhood function parameters were generated by training controller as follows:

$$A = 3072 - 8 \times H \quad (A : 3072 \rightarrow 1024),$$
$$S = 512 + 2 \times H \quad (S : 512 \rightarrow 1024).$$

The memory contained 4096 3-dimensional training vectors, which was addressed by the training controller. The vector elements ξ_0, ξ_1, and ξ_2 read from the memory are sent to the frequency modulators to generate the carrier signals X_0, X_1, and X_2. The DCO discussed in Section III-B was used as the frequency modulator. The mode of the system was initially

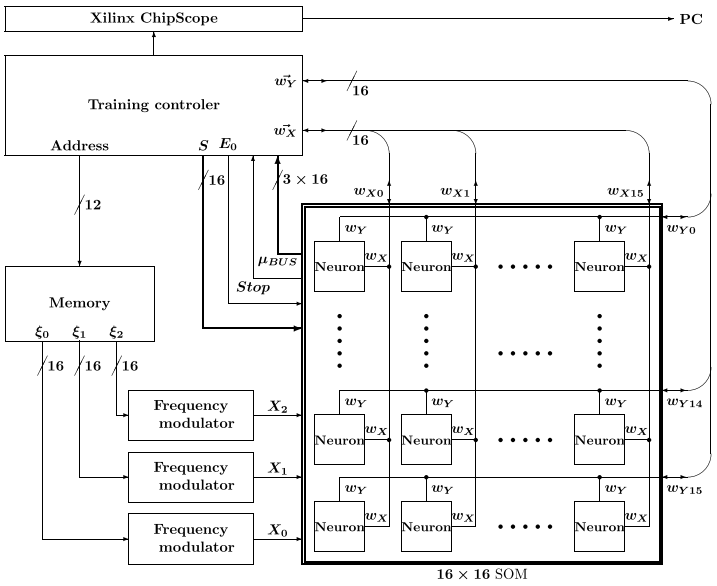

Fig. 14. FPGA configuration.

TABLE III
COMPARISON WITH OTHER HARDWARE SOMS

Design	Map size	Vec. dimension	Archiecture	Technology	MCUPS
[6]	6×10	76	Systolic array	MANTRA I system with Genes-IV chips	13.9
[11]	5×5	2	Custom	FPGA	28.38
[18]	25	3 (16-bit)	Custom	FPGA (Xilinx Virtex 5 xc5vlx50t)	113.3
[9]	16×8	128	SIMD NBX processor	0.8-μm CMOS standard cell	383
[27]	16×16	3	Systolic array using a NoC architecture	FPGA (Xilinx VC707 FPGA board)	480
[8]	256	16	SIMD array	0.65-μm CMOS	661
[12]	100	2048	Systolic array	FPGA	3467
[7]	250×250	9	SIMD	RAPTOR2000 (FPGA)	5700
[17]	25	2 (8-bit)	Custom	FPGA	5847.5
[22]	64	2 (16-bit)	Custom	FPGA (Xilinx Kintex U xcku035)	8050
[13]	256	16	Custom	0.35-μm CMOS standard cell	9102
[10]	64	4	Custom	FPGA	17500
[14]	6050	194	Custom	FPGA	20604
[24]	16×16	32 (8-bit)	Systolic architecture	FPGA (Xilinx Virtex-7 xc7vx485t)	23997
[15]	16×16	3 (16-bit)	Custom	FPGA (Xilinx xc6vsx315t-ff1759-2)	25344
[16]	16×16	3	Custom	FPGA	28494 (MCPS)
[29]	16×16	3 (16-bit)	Custom	FPGA (Xilinx xc7vx690ttfg1761-2)	33024
[26]	10×10	79 (16-bit)	Custom	FPGA (Xilinx Virtex-U xcvu440-flga20892)	37620
[25]	256	3	Custom	FPGA (Xilinx xc7vx690)	109800
[50]	5×5	2 (10-bit)	DPLL based processor	FPGA (Altera Ep20K400EBC625)	4.89
[54]	16×16	3 (16-bit)	DFLL based processor	FPGA (Xilinx xc6vsx315t-ff1759-2)	5.71
This work	16×16	3 (16-bit)	DFLL based processor	FPGA (Xilinx xc6vsx315t-ff1759-2)	15.0

set to Search mode. When the *Stop* signal was activated, the mode switched to Update mode, and the S values and signal $E_0 = 1$ were fed to the SOM for A clock cycles, and the weight vectors were updated. On completion of the update, the memory address increased and the mode was switched to Search mode again. On-chip learning proceeded by repeating these two modes. After 256 epochs, the mode was switched to Dump mode, where the controller generated $\vec{w_X}$ and $\vec{w_Y}$ signals so that all weight vectors were read one by one through the μ_{BUS}. Further, the vectors and other selected signals were transferred to a PC through Xilinx ChipScope Pro.

Hardware resource utilization of the implemented FPGA is summarized in Table I.

C. On-Chip Learning

The on-chip learning performance of the proposed SOM was investigated using five types of learning datasets (Random, Triangle, 16×16 mesh, Donut, Butterfly), each of which consisted of 4096 vectors. Fig. 15 shows the transitions of the weight vectors during the on-chip learning. As shown in the figure, the proposed SOM extends the weight vector well from the initial value while maintaining the topology between the neurons so as to cover the entire training vector space, and the SOM successfully learned five vector sets. Of these five datasets, the 16×16 mesh was the most difficult one for SOM to learn because the SOM not only extends the weight vectors to the training vector space, but also the individual weight vectors must be placed in the corresponding 256 vector clusters. Thus, use of this dataset made it easy to determine whether the learning succeeded. Failure of the learning could be easily judged if any of the weight vectors are misplaced. Therefore, this dataset was used as reference data for experimental evaluation, and the neighborhood function parameters A and S used in this experiment were chosen so

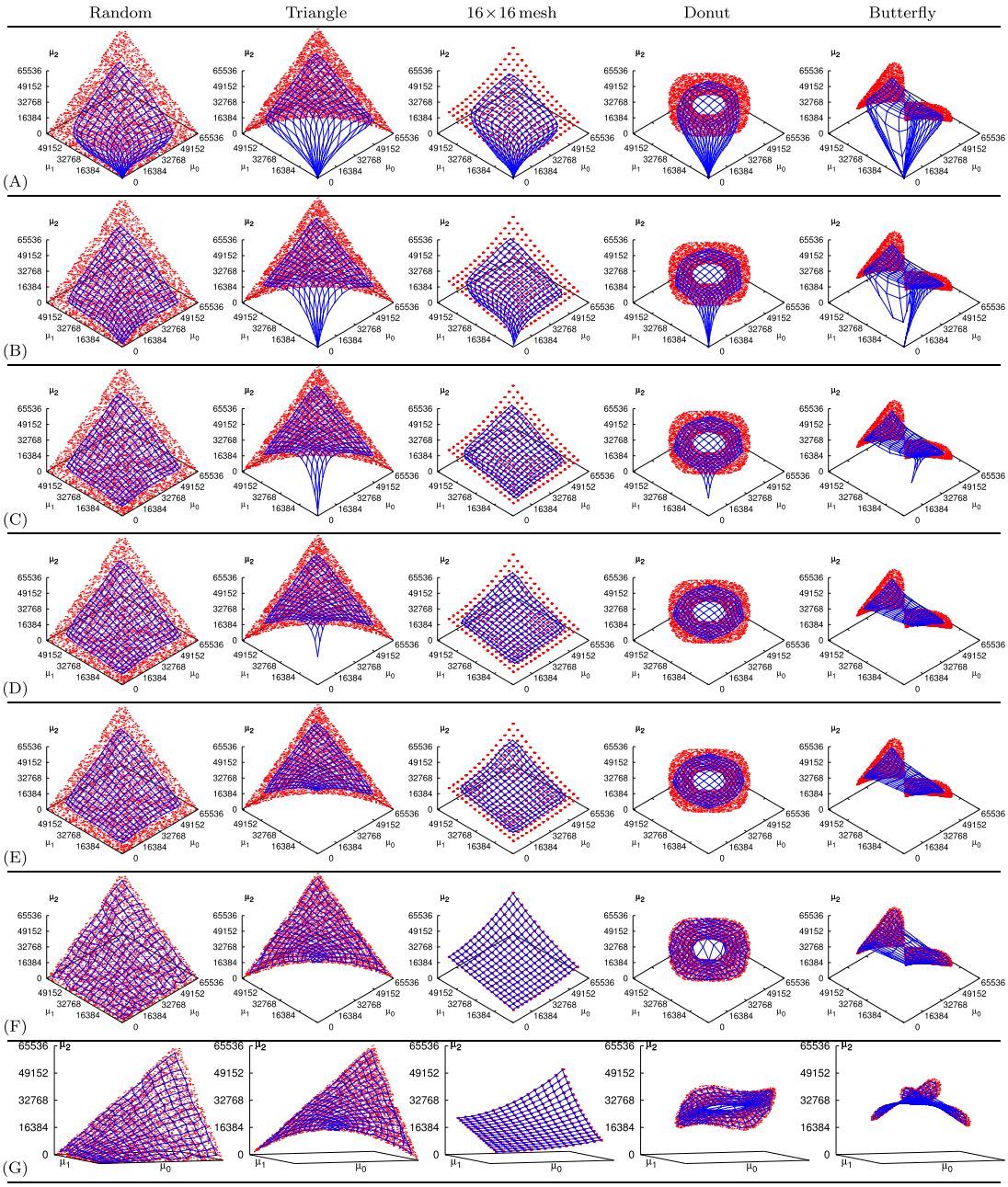

Fig. 15. Weight vectors during the on-chip learning of the proposed SOM, at (A) $H = 1$, (B) $H = 2$, (C) $H = 3$, (D) $H = 4$, (E) $H = 8$, (F) $H = 256$, (G) $H = 256$ (different angle view).

that the SOM could successfully learn the 16×16 mesh dataset. It should be noted that the previous DFLL-SOM [54] could not successfully learn this 16×16 mesh dataset.

All weight vectors were initialized to the same value. Thus, all vector distances to the very first input vector were the same, and it was impossible to determine the winner in this case. However, all the on-chip learning was successful because the proposed priority circuit worked well and selected one winner from the winner candidates.

For comparison purposes, learning behavior of the original SOM algorithm discussed in Section II was examined by using the same datasets used in Fig. 15. The original algorithm was implemented in software using the Euclidean vector distance shown in (4) and the Gaussian neighborhood function in (7). All computations for the SOM were carried out by floating-point arithmetic. The learning rate α and radius σ were programmed as follows:

$$\alpha = 0.7 \cdot (1 - H/256) \quad (\alpha : 0.07 \rightarrow 0.0),$$
$$\sigma = 2.0 \cdot (1 - H/256) \quad (\sigma : 2.0 \rightarrow 0.0),$$

where $H : 0 \rightarrow 255$.

Transitions of the weight vectors during the learning are shown in Fig. 16, which are very similar to those of the proposed SOM in Fig. 15. Therefore, it is considered that the proposed SOM based on the triangular neighborhood function and pulse mode operation has compatible functionality as the original SOM algorithm.

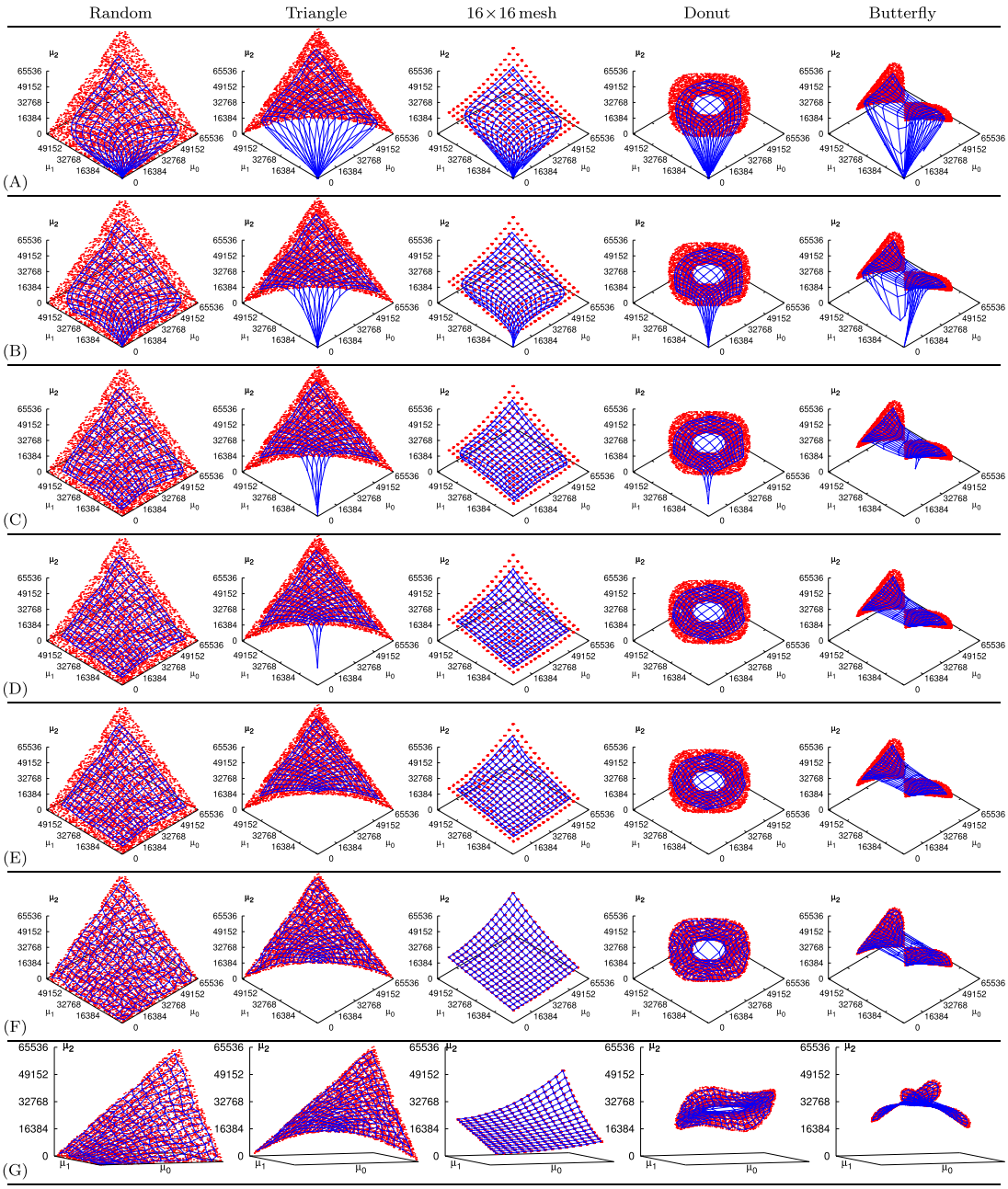

Fig. 16. Weight vectors during the learning of original SOM algorithm, at (A) $H = 1$, (B) $H = 2$, (C) $H = 3$, (D) $H = 4$, (E) $H = 8$, (F) $H = 256$, (G) $H = 256$ (different angle view).

As shown in (11) and Fig. 12, the search time of the proposed winner search circuit is variable with respect to the vector distance, so the actual learning time differs for different learning data. The details of the number of clock cycles for the learning is shown in Fig. 17 (A). The average numbers of clocks used in both the winner search and vector update for one input vector are plotted in the figure. The number of clocks for vector update was determined by the neighborhood function parameter A, which decreased as the learning progressed. Thus, the update time decreased as the learning progressed. Fig. 17 (B) is an enlarged view of the learning shown in Fig. 17 (A), showing that it took a longer time for the SOM to find winners because of the large vector distances in the early stage of the learning. Note that the weight vectors corresponding to Fig. 17 (B) are shown in Fig. 15 (A)-(E). Fig. 17 (B) indicates that the winner search is gradually getting shorter as learning progresses.

The highest operating clock frequency was measured by using the 16×16 mesh as the reference dataset, and the highest frequency was 67 MHz. Speed of the proposed SOM and the number of clocks required for the learning of five datasets are presented in Tab. II. The speed is expressed in million updates per second (MCUPS). The performance is computed from the number of weight vector elements, the number of the learning iterations, the number of clock cycles, and the clock frequency.

In terms of the operating speed, the performances of the hardware SOMs in the literature are summarized in Tab. III. Compared to the previous work [54], and the operating speed

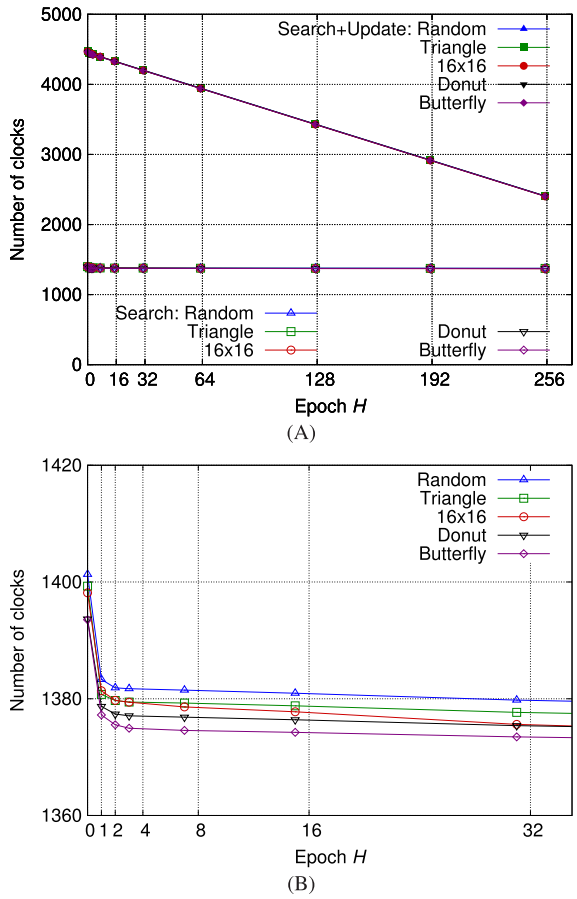

Fig. 17. Average number of clocks for each instance at epoch G. (A) The number of clocks for search and update operations, (B) The number of clocks for search operation in early stage of learning.

of the proposed SOM was improved to be about twice as fast. However, it can be seen that the operation speed is considerably slower than that of other hardware SOMs. The other hardware SOMs were based on the arithmetic computation architectures. Whereas [50], [54], and the proposed hardware SOMs were based on the frequency/phase modulated pulse signals, and computation was done by DPLL/DFLL which requires a large number of clocks. Thus the number of clocks to process a single input vector is quite large, which is the reason of the low performance.

V. Conclusion

This paper proposed a very unique hardware SOM architecture, in which vectors were represented by frequency modulated pulse signals, and the SOM computation was carried out by controlling the DFLLs. Winner search was carried out by taking advantage of the frequency modulated signal. The triangular neighborhood function was implemented by distributing enable signals for the DFLLs. The proposed winner search and neighborhood function circuits were tested by VHDL simulations. It was confirmed that the winner search circuit responded in proportion to the vector distance, and the response time was consistent with the theoretical value. The winner neuron was determined by competition between neurons in terms of the fastest response time.

The computing components for the SOM were described by VHDL and examined by the VHDL simulations, then the SOM was implemented on the FPGA to test the on-chip learning of the SOM. The VHDL simulation of the neighborhood function revealed that the proposed circuit provided the triangular function that agreed well with theory. Learning experiments were conducted by using five types of learning datasets. The results of the experiments showed the SOM's on-chip learning capability with the topology-preserving nature.

Since the DFLL used in the proposed SOM controlled the frequency of the carrier signal digitally, its clock frequency must be much higher than that of the carrier signal. Therefore, the frequency of the carrier signals were much slower than the clock frequency, and the operating speed of the SOM could not be made higher. Though its speed needs to be improved, the proposed architecture is a unique one based on pulse signals. The purpose of this study was to develop the hardware SOM that mimics the biological brain by using pulse mode operation like a spiking neural network. It was verified that the SOM with unsupervised learning can be realized with neurons communicating with each other using frequency-modulated pulse signals.

From the viewpoint of mimicking the brain, analog circuit implementation, i.e., use of analog PLL, is desired as future work.

References

[1] T. Kohonen, *Self-Organizing Maps*. New York, NY, USA: Springer, 2001.
[2] S. Y. Kung, *Digital Neural Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
[3] B. A. White and M. I. Elmasry, "The digi-neocognitron: A digital neocognitron neural network model for VLSI," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 73–85, Jan. 1992.
[4] T. Ikenaga and T. Ogura, "A DTCNN universal machine based on highly parallel 2-D cellular automata CAM/sup 2," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 5, pp. 538–546, May 1998.
[5] G. Frank, G. Hartmann, A. Jahnke, and M. Schafer, "An accelerator for neural networks with pulse-coded model neurons," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 527–538, May 1999.
[6] P. Ienne, P. Thiran, and N. Vassilas, "Modified self-organizing feature map algorithms for efficient digital hardware implementation," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 315–330, Mar. 1997.
[7] M. Porrmann, M. Franzmeier, H. Kalte, U. Witkowski, and U. Rückert, "A reconfigurable SOM hardware accelerator," *Proc. Eur. Symp. Artif. Neural Netw. (ESANN)*, 2002, pp. 337–342.
[8] D. C. Hendry, A. A. Duncan, and N. Lightowler, "IP core implementation of a self-organizing neural network," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1085–1096, Sep. 2003.
[9] M. Porrmann, U. Witkowski, and U. Ruckert, "A massively parallel architecture for self-organizing feature maps," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1110–1121, Sep. 2003.
[10] H. Tamukoh, T. Aso, K. Horio, and T. Yamakawa, "Self-organizing map hardware accelerator system and its application to realtime image enlargement," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Sep. 2003, pp. 2683–2687.
[11] J. Pena, M. Vanegas, and A. Valencia, "Digital hardware architectures of Kohonen's self organizing feature maps with exponential neighboring function," in *Proc. IEEE Int. Conf. Reconfigurable Comput. FPGA's (ReConFig)*, Sep. 2006, pp. 1–8.
[12] I. Manolakos and E. Logaras, "High throughput systolic SOM IP core for FPGAs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. ICASSP*, Apr. 2007, p. 61.
[13] A. Ramirez-Agundis, R. Gadea-Girones, and R. Colom-Palero, "A hardware design of a massive-parallel, modular NN-based vector quantizer for real-time video coding," *Microprocessors Microsyst.*, vol. 32, no. 1, pp. 33–44, Feb. 2008.

[14] J. Lachmair, E. Merényi, M. Porrmann, and U. Rückert, "A reconfigurable neuroprocessor for self-organizing feature maps," *Neurocomputing*, vol. 112, pp. 189–199, Jul. 2013.

[15] H. Hikawa and Y. Maeda, "Improved learning performance of hardware self-organizing map using a novel neighborhood function," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2861–2873, Nov. 2015, doi: 10.1109/TNNLS.2015.2398932.

[16] Z. Huang *et al.*, "A hardware-efficient vector quantizer based on self-organizing map for high-speed image compression," *Appl. Sci.*, vol. 7, no. 11, p. 1106, Oct. 2017, doi: 10.3390/app7111106.

[17] M. A. de Abreu de Sousa and E. Del-Moral-Hernandez, "Comparison of three FPGA architectures for embedded multidimensional categorization through Kohonen's self-organizing maps," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050799.

[18] M. A. de Abreu de Sousa and E. Del-Moral-Hernandez, "An FPGA distributed implementation model for embedded SOM with on-line learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3930–3937, doi: 10.1109/IJCNN.2017.7966351.

[19] K. Appiah, A. Hunter, P. Dickinson, and H. Meng, "Implementation and applications of tri-state self-organizing maps on FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 8, pp. 1150–1160, Aug. 2012.

[20] H. Hikawa and K. Kaida, "Novel FPGA implementation of hand sign recognition system with SOM–Hebb classifier," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 1, pp. 153–166, Jan. 2015, doi: 10.1109/TCSVT.2014.2335831.

[21] J. Kim and P. Mazumder, "Energy-efficient hardware architecture of self-organizing map for ECG clustering in 65-nm CMOS," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 9, pp. 1097–1101, Sep. 2017, doi: 10.1109/TCSII.2017.2672789.

[22] M. A. de Abreu de Sousa, R. Pires, S. D. D. S. Perseghini, and E. Del-Moral-Hernandez, "An FPGA-based SOM circuit architecture for online learning of 64-QAM data streams," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8489518.

[23] Y.-H. Sun and T.-D. Chiueh, "A flexible and high-performance self-organizing feature map training acceleration circuit and its applications," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Mar. 2019, pp. 92–96, doi: 10.1109/AICAS.2019.8771556.

[24] K. B. Khalifa, A. G. Blaiech, and M. H. Bedoui, "A novel hardware systolic architecture of a self-organizing map neural network," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–14, Apr. 2019, doi: 10.1155/2019/8212867.

[25] G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re, and S. Spano, "AW-SOM, an algorithm for high-speed learning in hardware self-organizing maps," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 2, pp. 380–384, Feb. 2020, doi: 10.1109/TCSII.2019.2909117.

[26] M. A. D. A. D. Sousa, R. Pires, and E. Del-Moral-Hernandez, "SOM-processor: A high throughput FPGA-based architecture for implementing self-organizing maps and its application to video processing," *Neural Netw.*, vol. 125, pp. 349–362, May 2020, doi: 10.1016/j.neunet.2020.02.019.

[27] M. Abadi, S. Jovanovic, K. Ben Khalifa, S. Weber, and M. H. Bedoui, "A hardware configurable self-organizing map for real-time color quantization," in *Proc. IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2016, pp. 336–339, doi: 10.1109/ICECS.2016.7841201.

[28] M. Abadi, S. Jovanovic, K. B. Khalifa, S. Weber, and M. H. Bedoui, "A scalable and adaptable hardware NoC-based self organizing map," *Microprocessors Microsyst.*, vol. 57, pp. 1–14, Mar. 2018.

[29] H. Hikawa, "Nested hardware architecture for self-organizing map," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–7.

[30] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

[31] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(N) complexity," in *Proc. Adv. Neural Inf. Process. Syst.*, San Mateo, CA, USA: Morgan Kaufmann, 1989, pp. 703–711.

[32] M. Oster, R. Douglas, and S.-C. Liu, "Computation with spikes in a winner-take-all network," *Neural Comput.*, vol. 21, pp. 2347–2465, Sep. 2009.

[33] B. Ruf and M. Schmitt, "Self-organization of spiking neurons using action potential timing," *IEEE Trans. Neural Netw.*, vol. 9, no. 3, pp. 575–578, May 1998.

[34] D. T. Pham, M. S. Packianather, and E. Y. A. Charles, "A self-organising spiking neural network trained using delay adaptation," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jun. 2007, pp. 3441–3446.

[35] A. Gupta and L. N. Long, "Hebbian learning with winner take all for spiking neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 1054–1060.

[36] T. K. Horiuchi, "A spike-latency model for sonar-based navigation in obstacle fields," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 11, pp. 2393–2401, Nov. 2009, doi: 10.1109/TCSI.2009.2015597.

[37] B. Mailachalam and T. Srikanthan, "Area–time issues in the VLSI implementation of self organizing map neural networks," *Microprocessors Microsyst.*, vol. 26, nos. 9–10, pp. 399–406, Dec. 2002.

[38] R. Dlugosz, M. Kolasa, and K. B. W. Electronics, "Programmable triangle nighborhood function for Kohonen self-organizing map implemented on chip," in *Proc. Int. Conf. Mixed Design Integr. Circuits Syst. (MIXDES)*, Jun. 2010, pp. 328–332.

[39] R. Dlugosz, M. Kolasa, W. Pedrycz, and M. Szulc, "Parallel programmable asynchronous neighborhood mechanism for Kohonen SOM implemented in CMOS technology," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2091–2102, Dec. 2011.

[40] M. Kolasa, R. Dlugosz, W. Pedrycz, and M. Szulc, "A programmable triangular neighborhood function for a Kohonen self-organizing map implemented on chip," *Neural Netw.*, vol. 25, pp. 146–160, Jan. 2012.

[41] T. Kohonen, "The self-organizing map, a possible model of brain maps," *Med. Biol. Eng. Comput.*, vol. 34, p. 204, Aug. 1996.

[42] L. M. Reyneri, "A performance analysis of pulse stream neural and fuzzy computing systems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 10, pp. 642–660, Oct. 1995.

[43] Y.-C. Kim and M. A. Shanblatt, "Random noise effects in pulse-mode digital multilayer neural networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 220–229, Jan. 1995.

[44] G. Moon, M. E. Zaghloul, and R. W. Newcomb, "VLSI implementation of synaptic weighting and summing in pulse coded neural-type cells," *IEEE Trans. Neural Netw.*, vol. 3, no. 3, pp. 394–403, May 1992.

[45] H. Hikawa, "A new digital pulse-mode neuron with adjustable activation function," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 236–242, Jan. 2003.

[46] H. Hikawa, "A digital hardware pulse-mode neuron with piecewise linear activation function," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1028–1037, Sep. 2003.

[47] A. L. Hodgkin and A. F. Huxley, "A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes," *J. Physiol.*, vol. 117, pp. 500–544, Aug. 1952.

[48] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[49] F. C. Hoppensteadt and E. M. Izhikevich, "Pattern recognition via synchronization in phase-locked loop neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 734–738, May 2000.

[50] H. Hikawa, "FPGA implementation of self organizing map with digital phase locked loops," *Neural Netw.*, vol. 18, nos. 5–6, pp. 514–522, Jul. 2005.

[51] H. Hikawa, "DPLL based hardware SOM with a new winner-take-all circuit," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 2641–2648.

[52] H. Hikawa, "Vector classification by a winner-take-all neural network with digital frequency-locked loop," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 2517–2520.

[53] H. Hikawa, "Improved winner-take-all circuit for neural network based on frequency-modulated signals," in *Proc. IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2016, pp. 85–88.

[54] H. Hikawa, H. Ito, and Y. Maeda, "Hardware self-organizing map based on frequency-modulated signal and digital frequency-locked loop," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5, doi: 10.1109/iscas.2018.8351364.

Hiroomi Hikawa (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Japan, in 1984, 1986, and 1989, respectively. Then, he served as a Post-Doctoral Researcher with the University of South Florida. From 1992 to 2008, he worked at the Computer Science and Intelligent Systems Department, Oita University. In 2008, he joined the Department of Electrical and Electronic Engineering, Kansai University, where he is currently a Professor. His research interest includes architecture for signal processing and neural networks.