Supervised Learning Using Spike-Timing-Dependent Plasticity of Memristive Synapses

Yu Nishitani, Yukihiro Kaneko, *Member, IEEE*, and Michihito Ueda

***Abstract*— We propose a supervised learning model that enables error backpropagation for spiking neural network hardware. The method is modeled by modifying an existing model to suit the hardware implementation. An example of a network circuit for the model is also presented. In this circuit, a three-terminal ferroelectric memristor (3T-FeMEM), which is a field-effect transistor with a gate insulator composed of ferroelectric materials, is used as an electric synapse device to store the analog synaptic weight. Our model can be implemented by reflecting the network error to the write voltage of the 3T-FeMEMs and introducing a spike-timing-dependent learning function to the device. An XOR problem was successfully demonstrated as a benchmark learning by numerical simulations using the circuit properties to estimate the learning performance. In principle, the learning time per step of this supervised learning model and the circuit is independent of the number of neurons in each layer, promising a high-speed and low-power calculation in large-scale neural networks.**

***Index Terms*— Memristor, spike-timing-dependent plasticity (STDP), spiking neural network (SNN) hardware, supervised learning.**

I. Introduction

SPIKING neural networks (SNNs) are designed to closely emulate the architecture of biological nervous systems [1]. In this network, the neurons communicate with one another using electric pulses called spikes. The neural information is expressed by the relative timing of the spikes, rather than by the specific shape of the spikes [2]–[4]. SNNs have received much interest because not only their similarity to biological neurons helps us analyze the elementary processes in the brain, but also SNNs can be potentially applied to an adaptable information processing system with low computing power [1]. SNNs have been theoretically proved to have a larger processing ability than the conventional non-SNN [5]. However, software simulations of SNNs require much computational efforts because primarily, the computation is performed by a sequentially computing von Neumann machine, and the temporal dynamics of the neuron predominant in the neural computation must be calculated. Consequently, we are not able to capitalize on the full benefit of SNNs. To overcome this problem, SNN hardware has to be developed in which the neuron dynamics are realized in the electric signal of the devices.

One of the biggest issues in constructing SNN hardware is developing the artificial synapse devices, which store the analog synaptic weights. Recently, reports not only on the conventional floating-gate transistor-based synapses [6] but also on the novel analog memory-based synapses have been presented [7]–[12]. These analog memory devices are called memristors. Memristors possess the characteristic in which their electric conductance changes depending on the applied voltage or current, and the modified conductance is retained after the bias is cutoff. If the conductance is regarded as a synaptic weight, the memristors can act as a synapse device capable of electrical weight control. In addition, analog circuit-based implementations of neurons and synapses using memristors have been reported as superior to digital circuit-based implementation in terms of circuit area and power consumption [13]. Meanwhile, we have developed an original analog memory device as a three-terminal ferroelectric memristor (3T-FeMEM) [14], [15] and demonstrated its synaptic application [16]–[18].

Network functionality has been also demonstrated in SNNs using these memristors as synapse devices. In the reports, the potential of memristors for neural applications is shown in winner-take-all feedforward networks [19], [20] and Hopfield networks [21], [22] with unsupervised learning. However, no practical hardware implementation has been made on SNN-based supervised learning, e.g., error backpropagation (BP).

Some BP algorithms for SNNs have been presented recently as among the most popular supervised learning algorithms [23]–[27]. In these studies, the algorithm performance was investigated by software simulations. However, these approaches required much time to complete all learning processes for essentially the same reason as the problem in the SNN software simulation mentioned above. On the other hand, although the hardware implementation of these models could be a solution, such implementation has not been reported because in these proposed algorithms, the weight adjustment is expressed in a too complex equation to be completely developed into an analog circuit.

In this paper, we propose a simplified supervised SNN learning model suitable for hardware implementation. In addition, an analog SNN circuit for the model is presented as an example. In the circuit, we use a 3T-FeMEM with a brain-like learning function as synapse devices. We demonstrate the learning performance of the model

Manuscript received April 24, 2014; revised January 26, 2015; accepted January 29, 2015. Date of publication February 24, 2015; date of current version November 16, 2015.

The authors are with the Advanced Research Division, Panasonic Corporation, Kyoto 619-0237, Japan (e-mail: nishitani.yu@jp.panasonic.com; kaneko.yukihiro001@jp.panasonic.com; ueda.michihito@jp.panasonic.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2399491

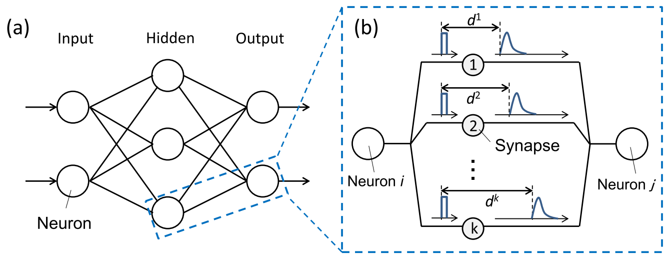

Fig. 1. (a) Spiking neuron-based feedforward neural network composed of input, hidden, and output layers. (b) Multiple synapses transmitting spikes from presynaptic neuron i to postsynaptic neuron j with some delay d^k.

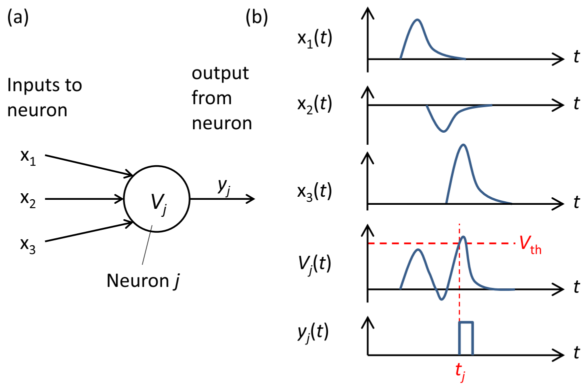

Fig. 2. (a) Schematic of a spiking neuron with multiple input and output. (b) Temporal changes in input x_1–x_3, resulting integrated voltage V_j and output y_j.

and the SNN circuit using an XOR problem as a learning benchmark.

II. Error Backpropagation for Spiking Neural Networks

A. Network of Spiking Neurons

SNN is a network that consists of spiking neurons. Fig. 1(a) shows the multilayer feedforward neural network as an SNN example. In SNNs, the input and output signals are expressed as spikes (pulse signals), where the analog information is represented by the relative spike timing. In the case of a supervised learning, the desired output signals (hereafter called teach signals) are also input to the SNN as spikes. The input spikes are applied to the input layer. Subsequently, these spikes propagate to the hidden and output layers. In the SNNs, each of the spiking neurons is connected by synapses in the same manner as that in conventional neural networks. In general, neurons i and j can be connected by parallelly connected multiple synapses, as shown in Fig. 1(b). Each of the synapses, e.g., kth synapse, has its own synaptic weight w_{ij}^k and delay d^k. In other words, a synapse weighs the signal from presynaptic neuron i. The postsynaptic neuron j receives the weighted signal at $t = t_i + d^k$, where t_i is the output timing of the presynaptic neuron i. Hereafter, we explain the details of the spiking neuron model by taking a leaky integrate-and-fire (LIF) neuron as an example. Fig. 2 shows the schematic of an LIF neuron. The inner potential of the neuron changes with respect to the signal input from the presynaptic neurons. The potential of neuron j is expressed as follows:

$$V_j(t) = \sum_i \sum_k w_{ij}^k \varepsilon(t - t_i - d^k) \tag{1}$$

where $\varepsilon(t)$ represents the spike response function, which is an unweighted response of the inner potential of a neuron with respect to an input spike. For example, the so-called α function can be used as $\varepsilon(t)$ as follows [4]:

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau}\exp\left(1 - \frac{t}{\tau}\right) & \text{when } t > 0 \\ 0 \quad \text{when } t \le 0 \end{cases} \tag{2}$$

where τ is the time constant of the neuron discharging and charging. If V_j exceeds a threshold value V_{th} at $t = t_j$, neuron j outputs a spike at $t = t_j$, as shown in Fig. 2(b).

B. Error Backpropagation

Some types of BP algorithms for the SNN have been proposed so far [23]–[27]. Traditional BPs are among the most known supervised learning algorithms for pattern recognition, function estimation, classification problems, and so on. The BP models for the SNN are based on the same concept as the traditional BP that employs the gradient descent method [28]. In the following, we briefly explain the algorithm using a feedforward network with input, hidden, and output layers. In the SNN BP, the input and teach signals are applied to the network, where the input timing of the signals expresses the neural information. If neuron j in the output layer fires at $t = t_j$ (which corresponds to the network output) and the desired output timing is $t = t_j^d$, network error E is computed as follows:

$$E = \frac{1}{2}\sum_j \left(t_j - t_j^d\right)^2. \tag{3}$$

The summation of j neurons is applied for all output-layer neurons. In the BP, the weights are adjusted to minimize E. The weight adjustment of the kth synapse between presynaptic neuron i and postsynaptic neuron j for every learning step is computed as

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \tag{4}$$

where η is the learning rate. In other words, Δw_{ij}^k is proportional to the network error gradient with respect to the weight. Using (4), the weight adjustment of the synapse between each layer can be calculated as follows [23], [24]: the weight adjustment for the kth synapse between hidden-layer neuron i and output-layer neuron j is

$$\begin{aligned}\Delta w_{ij}^k &= -\eta \frac{\partial E}{\partial w_{ij}^k} \\ &= -\eta \frac{\left(t_j - t_j^d\right)\varepsilon\left(t_j - t_i - d^k\right)}{\sum_i \sum_k w_{ij}^k\ \varepsilon\left(t_j - t_i - d^k\right)\left(\frac{1}{t_j - t_i - d^k} - \frac{1}{\tau}\right)}.\end{aligned} \tag{5}$$

The weight adjustment for the kth synapse between input-layer neuron h and hidden-layer neuron i is presented

$$\Delta w_{hi}^{k} = -\eta \frac{\partial E}{\partial w_{hi}^{k}} = \eta\, \varepsilon\left(t_i - t_h - d^k\right) \times \sum_j \left\{ \left(t_j - t_j^d\right) \frac{\sum_k w_{ij}\, \varepsilon\left(t_j - t_i - d^k\right) \left(\frac{1}{t_j - t_i - d^k} - \frac{1}{\tau}\right)}{\sum_h \sum_k w_{hi}\, \varepsilon\left(t_i - t_h - d^k\right) \left(\frac{1}{t_i - t_h - d^k} - \frac{1}{\tau}\right) \times \sum_i \sum_k w_{ij}\, \varepsilon\left(t_j - t_i - d^k\right) \left(\frac{1}{t_j - t_i - d^k} - \frac{1}{\tau}\right)} \right\}. \quad (6)$$

in (6), as shown at the top of the page. This method is known as the SpikeProp [23], [24]. Some reports have shown the learning ability of using some benchmark learning sets, e.g., a binary XOR problem [23], [24]. However, the expressions of the weight adjustment shown in (5) and (6) are too complex to be definitely implemented on hardware neural networks. They contain many parameters as well as their summations, which require us to probe all the signals related to the terms and to wire the detected signals to reproduce the functions of (5) and (6) when the SNN BP is developed as hardware. Because a hardware neural network is generally already complicated owing to the increase in the wiring, further increase in the wiring for the BP is inappropriate for a hardware neural network. Therefore, we propose a simpler model that enables hardware implementation while keeping its convergence performance to some extents.

Here, we propose a simplified BP model appropriate for implementation as hardware SNNs. In this model, some terms related to the summations of the inverse variables in (5) and (6) are approximated as constants. In the model described by (5) and (6), these terms strictly change as the learning progresses, because the variables change at each learning step. However, from a practical perspective, these terms are constrained not to exceed a given value as a heuristic rule to avoid drastic weight changes, which will cause late convergence [23], [24]. Therefore, we put forward a hypothesis that these terms do not contribute much to the learning convergence. In the following, we propose a new weight adjustment suitable for hardware implementation as the simplified model. The weight adjustment for the kth synapse between hidden-layer neuron i and output-layer neuron j is expressed as

$$\Delta w_{ij}^{k} = -\eta \frac{\partial E}{\partial w_{ij}^{k}} = -\eta' \left(t_j - t_j^d\right) \varepsilon\left(t_j - t_i - d^k\right) \quad (7)$$

where the term $t_j - t_j^d$ is the error factor of the output-layer neuron j. The weight adjustment for the kth synapse between input-layer neuron h and hidden-layer neuron i is

$$\Delta w_{hi}^{k} = -\eta \frac{\partial E}{\partial w_{hi}^{k}} = \eta'' \varepsilon(t_i - t_h - d^k) \sum_j \left(t_j - t_j^d\right). \quad (8)$$

This equation contains the summation of the error factors in all postsynaptic neurons j, which means that errors in output-layer neurons are propagated to the hidden-layer neurons. Here, η' and η'' are the learning rate constants for the simplified BP model. In Section III, we present the hardware that performs this learning.

III. Spiking Neural Network Circuit

A. Memristor Synapse Device With Spike-Timing-Dependent Synaptic Plasticity Learning Function

Various types of memristors have been reported so far [8]–[18]. In most memristors, their conductance can be modulated in an analog manner by applying a pulse voltage with varying magnitude. This characteristic is suitable for the application of a synapse device in the SNN for BPs because the weight can be controlled depending on the network error by reflecting the error to the write voltage of the memristors. In other words, when the network error is larger, we modulate the write voltage of the memristors to become larger to make their conductance larger, and vice versa.

In this paper, to show the implementation of the simplified model on hardware SNNs, we use our originally developed 3T-FeMEM as a synapse device. The 3T-FeMEM has a field-effect transistor structure with source, drain, and gate electrodes [Fig. 3(a)]. This device is characterized by its gate insulator composed of ferroelectric materials, meaning that basically the device is a type of ferroelectric-gate transistors [29]–[31]. The application of gate voltage V_{GS} controls the channel electric conductance (source–drain current), which is physically brought by the change in the direction of the spontaneous electric dipoles in the gate insulator. Thus, V_{GS} corresponds to the write voltage in the 3T-FeMEM. Because of the nonvolatility of the spontaneous dipole, the changed conductance is retained after V_{GS} is turned OFF, which means that the 3T-FeMEM can be used to store the synaptic weight as an electrically controllable conductance. In this paper, we did not go into the details of the structure and device preparation, which are described in [14] and [15].

Fig. 3(b) shows the experimentally measured typical 3T-FeMEM transfer curve. The arrows indicate the sweep direction of the gate–source voltage V_{GS}. The 3T-FeMEM is turned ON under positive V_{GS}, which accumulates electrons in the channel, indicating that the device has an n-type conduction. When $V_{GS} = 0$ after being swept toward the positive direction, a higher channel conductance G (~60 μS) is observed. When $V_{GS} = 0$ after being swept toward the negative direction, a lower G (<0.1 nS) is observed. This conductance memory function is caused by the nonvolatility of the spontaneous electric dipole in the ferroelectric gate insulator. This V_{GS} sweep results in a logarithmic wide-range modulation of the conductance. However, in the synaptic application, the analog (linear range) control is more responsible. Such a continuous linear-range conductance control can be realized using pulse signals as the gate voltage, as shown in Fig. 3(c). After a rectangular pulse voltage is

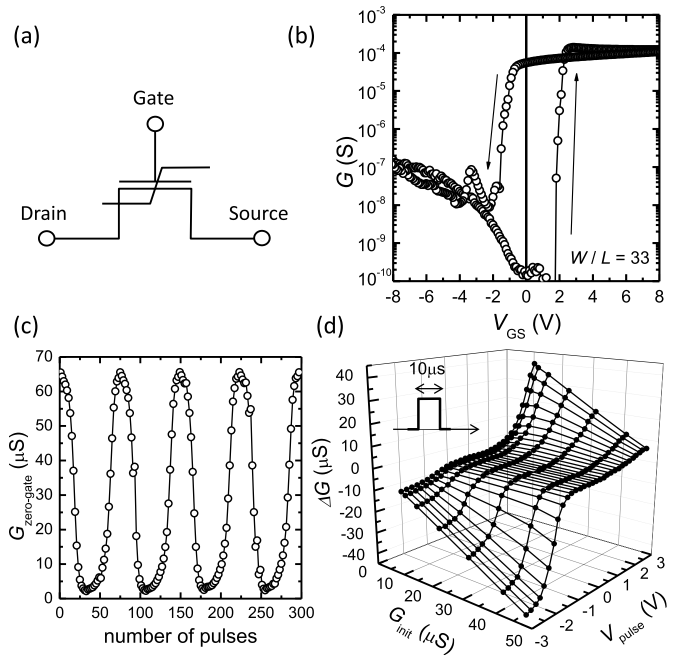

Fig. 3. (a) Circuit symbol of a 3T-FeFEM. (b) Channel conductance–gate voltage characteristics. (c) Continuous change in the memorized conductance of the 3T-FeMEM by applying a rectangular pulse voltage (pulsewidth $t_{\text{pulse}} = 10\ \mu$s) to the gate. The negative and positive gate pulses decrease and increase the conductance, respectively. (d) Pulse-height-dependence of the conductance change ΔG for various initialized conductance G_{init}.

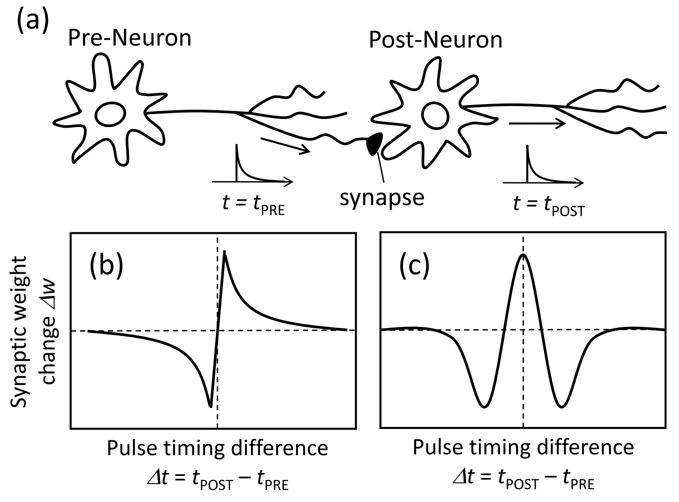

Fig. 4. (a) Schematics of the neurons (preneuron and postneuron) connected via a synapse. Characteristics of the (b) asymmetric and (c) symmetric STDP.

applied to the 3T-FeMEM gate, the conductance is measured when the gate voltage is equal to zero. The pulsewidth used here is 10 μs. The negative and positive gate pulses reduce and increase the conductance, respectively. The conductance modulation can be adjusted by varying the pulsewidth and/or the pulse height. Fig. 3(d) shows the conductance modulation ΔG before and after a rectangular pulse is applied to the 3T-FeMEM gate with varying pulse height V_{pulse}. Before applying a pulse, the conductance is initialized to a certain value G_{init}. Fig. 3(d) shows the ΔG–V_{pulse} curves in the case, where $G_{\text{init}} = 10.1 \pm 0.1, 11.5 \pm 0.2, 15.3 \pm 0.3, 22.6 \pm 0.4, 31.6 \pm 0.5, 41.4 \pm 0.7$, and $46.8 \pm 0.8\ \mu$S. When G_{init} is smaller, for example, $G_{\text{init}} = 10.1 \pm 0.1\ \mu$S, the negative (positive) V_{pulse} slightly decreases (significantly increase) the conductance. On the other hand, when G_{init} is larger, for example, $G_{\text{init}} = 46.8 \pm 0.8\ \mu$S, the negative (positive) V_{pulse} significantly decreases (slightly increases) the conductance. As stated above, the ΔG–V_{pulse} curve becomes more asymmetric as G_{init} approaches the minimum or maximum conductance value. This result is consistent with the fact that the conductance of 3T-FeMEM is a finite value. As mentioned above, ΔG is expressed as a function of two variables, such as $\Delta G(V_{\text{pulse}}, G_{\text{init}})$. In the later part of this paper, we present the BP simulation performed using this characteristic.

Referring back to (7) and (8), these equations comprise the terms concerning the timing difference between the pulse output of a hidden-layer neuron and the pulse arrival to the neuron $t_i - (t_h + d^k)$ or the timing difference between the pulse output of an output-layer neuron and the pulse arrival to the neuron $t_j - (t_i + d^k)$. This result indicates that the weight adjustment indeed depends on the spike timing of the presynaptic neuron and postsynaptic neuron. This characteristic is called spike-timing-dependent synaptic plasticity (STDP), which is observed in biological synapses [32]–[34]. In physiology, STDP is considered to be related to the learning mechanisms of neural systems [34]. The STDP characteristic is explained in Fig. 4. Fig. 4(a) shows the schematics of two neurons connected by a synapse. The presynaptic neuron sends a spike to the postsynaptic neuron via the synapse at $t = t_{\text{PRE}}$. The postsynaptic neuron fires in response to the spike at $t = t_{\text{POST}}$. The STDP characteristic means that the weight change of the synapse Δw depends on the timing difference between the spikes $\Delta t = t_{\text{POST}} - t_{\text{PRE}}$. Two main types of STDP are observed in a biological synapse. One is called the asymmetric STDP where Δw depends on the firing order and timing difference of the presynaptic neuron and postsynaptic neuron, as shown in Fig. 4(b). The other is the symmetric STDP, where the change in w is dominated by the absolute timing difference.

According to (7) and (8), the weight adjustment of our BP model also depends on the α-function with a spike-timing difference as the variable. Such α-function-like STDP learning function can be implemented to the 3T-FeMEM by changing the amplitude of V_{GS} as a write voltage according to Δt. In the following, the α-function is approximated as a triangular-wave function for simplicity. The STDP learning can be realized by applying V_{POST} to the 3T-FeMEM gate through a selector circuit, where its switching action is controlled by the V_{PRE} input, as shown in Fig. 5(a). An analog multiplexer can be used as a selector. Here, a triangular-shaped analog pulse is used as V_{POST} [Fig. 5(b)]. A part of V_{POST} is applied to the 3T-FeMEM only when V_{PRE} connects the selector V_{POST} terminal and the 3T-FeMEM gate. Because the V_{POST} amplitude changes with time, the applied pulse voltage to the gate V_{GS} depends on the timing difference between V_{PRE} and V_{POST}, namely, Δt, as shown in Fig. 5(b). As discussed earlier, the conductance change of the 3T-FeMEM is a function of

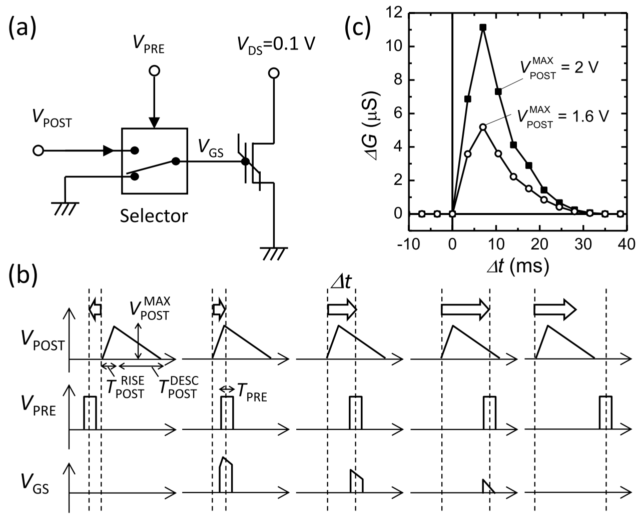

Fig. 5. (a) Circuit diagram of the synapse device composed of a 3T-FeMEM and a selector. (b) Schematics of the operation of the selector with applied V_{PRE} and V_{POST}. (c) Pulse-timing difference Δt dependence on the conductance change of the 3T-FeMEM.

the applied pulse height. Consequently, this driving method allows realization of an α-function-like STDP learning using the 3T-FeMEM. Fig. 5(c) shows the learning results under the following cases: 1) $T_{POST}^{RISE} = 7$ ms; 2) $T_{POST}^{DESC} = 28$ ms; 3) $T_{PRE} = 10$ μs; 4) $V_{POST}^{MAX} = 2$ or 1.6 V; and 5) $G_{init} = 31.6 \pm 0.5$ μS. The ΔG–Δt curve shows the pulse-timing-dependent conductance modulation that reflects the pulse shape of V_{POST}.

The weight adjustments are also proportional to the network error factor $\sum(t_j - t_j^d)$, as shown in (7) and (8). Here, j is the index of the output-layer neurons. Therefore, we only have to change the V_{POST} height depending on the error factor to reflect this error contribution to Δw because ΔG depends on the V_{POST} height, as shown in Fig. 5(c). Because the error factor is the timing difference between the output timing of output-layer neuron t_j and the input timing of teach pulse t_j^d, we can detect the error factor as the height of the voltage using a selector circuit and an analog pulse again. Fig. 6(a) shows an example of such an error-detection circuit. An analog-shaped voltage V_j^d, shown in Fig. 6(b), and a rectangle pulse V_j^{out} are input to the circuit at $t = t_j^d$ and t_j, respectively. V_j^{out} controls whether V_j^d will be applied or not by switching the selector. The output of the selector V_j^{sample} is applied to the peak-hold circuit, whose output is the dc voltage V_j^{error}. The input-timing-dependent behavior of the error-detection circuit is shown in Fig. 6(c). The smaller the absolute timing difference is, the smaller is the V_j^{error} magnitude, which means that the timing difference Δt_j^{error} can be converted to V_j^{error}. Here, V_j^{error} saturates when $|\Delta t_j^{error}| > 2T^d$ [Fig. 6(b)], because the maximum voltage is constrained by the supplied voltage to the circuit. The peak-hold circuit outputs a dc maximum voltage that can possibly be reached. The peak-hold circuit outputs the maximum height of the pulse as a dc voltage even after V_j^{out} goes back to 0 V. Using V_j^{error}, we generate V_{POST}, which reflects the error factor that changes the 3T-FeMEM conductance.

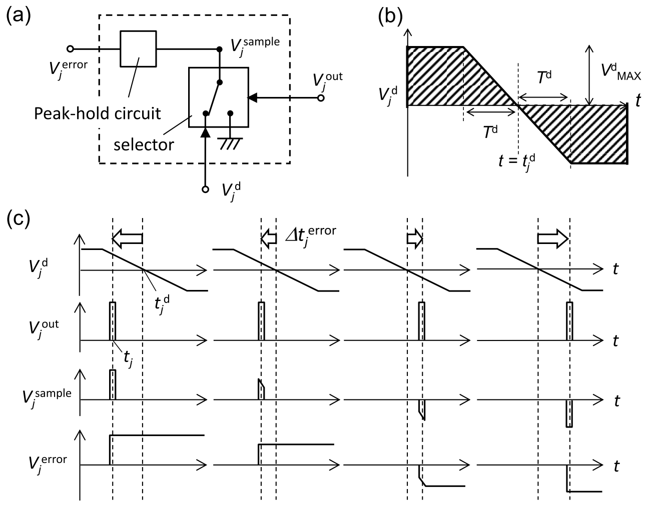

Fig. 6. (a) Circuit diagram of an error detector with input and output of the output-layer neuron V_j^{out} and teach pulse V_j^d. (b) Pulse shape of $V_j{}^d$. (c) Schematics of the operation of the error detector.

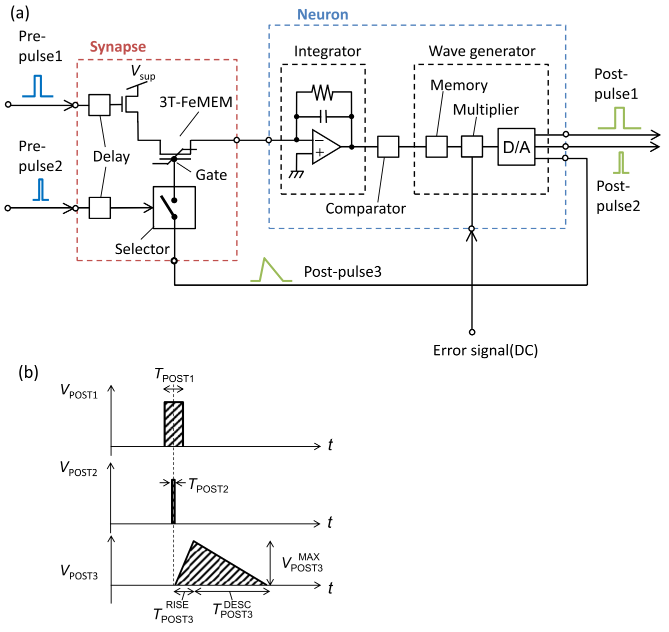

Fig. 7. (a) Example of 3T-FeMEM based synapse and LIF neuron circuit. (b) Pulse shape of the output of the neuron V_{POST1}, V_{POST2}, and V_{POST3}.

As discussed earlier, the 3T-FeMEM-based synapse with STDP and error-reflected learning function is suitable for electronic synapse for SNN circuits capable of BP learning. In addition, we have reported that our 3T-FeMEM can be embedded on such a CMOS circuit [22]. Using this technique, on-chip neural network circuits comprising 3TFeMEM synapses can be constructed.

B. Spiking Backpropagation Circuit

The spiking BP circuit can be constructed using the 3T-FeMEM synapses with STDP learning function and LIF spiking neuron circuits. Fig. 7(a) shows an example of the neuron circuit connected to the 3T-FeMEM synapse. Pulse voltages from a presynaptic neuron circuit are input as prepulses1 and 2 to a neuron circuit via the synapses. Although the neuron circuit receives multiple signals from other presynaptic neurons via multiple synapses, only one input and one synapse are shown in Fig. 7 for simplicity.

The synapse circuit example comprises a selector, a switch, two delay circuits, and a 3T-FeMEM. These delay circuits with the same delay time represent the synaptic delay shown as d^k in Fig. 1(b). Prepulse2 is used for learning. It acts as V_{PRE} in the STDP learning scheme mentioned previously [Fig. 5(a)]. Prepulse1 is used to produce a weighted signal, which is transmitted to the presynaptic neuron circuit. The pulse closes the switch, resulting in the flow of pulse current with an amplitude proportional to the 3T-FeMEM conductance. This process means that supply voltage V_{sup} is converted to current i, which is weighted by the 3T-FeMEM conductance G

$$i = V_{\mathrm{sup}} \times G. \tag{9}$$

Although G is always positive, the weights can be both positive and negative. To express a negative weight, the 3T-FeMEM applied with positive V_{sup} and that applied with negative V_{sup} are connected in parallel (not shown in the figure)

$$i(t) = \left(V_{\mathrm{sup}}^{+} \times G^{+} + V_{\mathrm{sup}}^{-} \times G^{-}\right) \tag{10}$$

where V_{sup}^{+} and V_{sup}^{-} are the positive and negative supply voltages, respectively. G^{+} and G^{-} represent the exciter and inhibitor circuits, respectively.

The spiking neuron circuit comprises an integrator with an op-amp, a comparator, and a wave generator. When current pulses are input to the neuron circuit, the capacitor connected parallel to the op-amp is charged, resulting in the change in the integrated voltage. The integrated voltage of this type of neuron circuit can well reflect the 3T-FeMEM conductance [21]. For the following BP simulations, the integrated voltage $V(t)$, which corresponds to the inner potential of the neuron, is formulated as follows:

$$C\frac{dV(t)}{dt} + \frac{1}{R}V(t) + \sum_{i} i_i(t) = 0 \tag{11}$$

where C and R are the capacitance and resistance of the capacitor and resistor, respectively, connected parallel to the op-amp in the integrator.

The comparator outputs a trigger when $V(t)$ exceeds a threshold in the neuroncircuit, shown in Fig. 7(a) as an example. Using this trigger, a digital memory in the wave generator outputs pulse-shaped data to the multiplier, where the pulse amplitude is multiplied with the error signal described earlier. The multiplied digital signal is converted to an analog pulse by a digital/analog converter, and subsequently output as postpulses. As explained in Section II, this multiplication indicates that the weight adjustment signal is modified according to the error factor. Two of the three output pulses (postpulses1 and 2 with a rectangle shape) are applied to the postsynaptic neuron circuits. Another pulse (postpulse3) is fed back to the synapse circuit, which acts as a postpulse in the STDP learning scheme (Fig. 5). Postpulse3 is a triangular-shaped pulse, which represents the characteristics

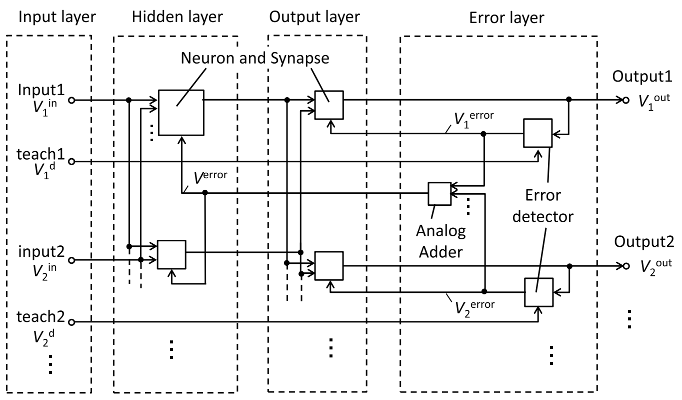

Fig. 8. SNN for error-BP learning with input, hidden, output, and error layers.

of the STDP, e.g., α-function. The shapes of these pulses are shown in Fig. 7(b). These pulses are output at the time when the temporal centers of the pulse (shown as a dotted line) are identical.

A feedforward neural network can be formed using these spiking neuron circuits and synapse circuits, as shown in Fig. 8. The network has multiple input terminals for the learn and teach signals. What we want the network to learn is represented by learn signals V_h^{in}, where h is the index of the input. The teach signals V_j^d correspond to the desired outputs for the learn signals. In general, a feedforward network has input, hidden, and output layers (Fig. 1). Our circuit has not only these layers but also an error layer, which comprises the error detectors and an analog adder. This layer generates signals to reflect the errors defined in (7) and (8). The error detector is shown in Fig. 6. The output of the error detector V_j^{error} is applied to the corresponding output-layer neuron j as an error signal. All error signals V_j^{error} are also summed up in the analog adder, whose output V^{error} is applied to all hidden layer-neuron circuits as an error signal. Using this circuit, the gate voltage applied to the 3T-FeMEM to update its conductance as a synaptic weight is proportional to the weight adjustment expressed in (7) and (8). The conductance change in the 3T-FeMEM depends on the applied V_{pulse}, as shown in Fig. 3(d). Therefore, the weight of the synapse is updated to reduce the error.

In this section, we have shown a spiking BP circuit with 3T-FeMEM synapses. The bias-magnitude-dependent conductance change and the STDP-like learning function have been reported in other memristors, including the two-terminal ones, as in the case with the 3T-FeMEM. Therefore, the concept of constructing the spiking BP circuit shown here can be applied for other memristor-based synapses.

C. Learning Sequence

At the start, we explained the input and output coding. Because we use SNNs, the input and output information has to be coded as pulse timings. For the input coding, two schemes are known [23]. One is the linear coding in which the values of each input are proportionally converted to a pulse input timing in a predetermined time range. The other is the

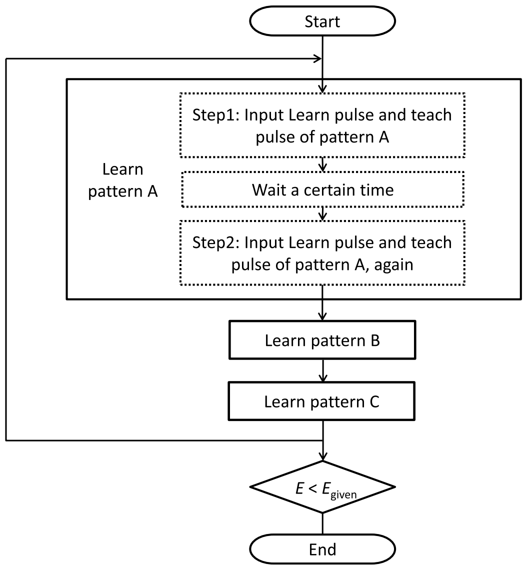
Start
Step1: Input Learn pulse and teach pulse of pattern A
Learn pattern A
Wait a certain time
Step2: Input Learn pulse and teach pulse of pattern A, again
Learn pattern B
Learn pattern C
$E < E_{\text{given}}$
End

Fig. 9. Flowchart of the spiking error-BP learning. The patterns to learn are A, B, and C.

population coding. In this scheme, each value is converted to multiple pulses by a function composed of several Gaussian functions. Here, the detail is not mentioned. For the output coding, we have two available choices [24], [25]. In the first scheme, multiple classes are separately encoded by multiple output neurons where the output of each neuron is 1 or 0. In other words, the output timing is early or late. The second scheme uses only one output neuron where multiple classes are represented by multiple output pulse timings. For example, if the class is three, the output timings are early middle and late. These coding schemes are employed relative to the problem.

In the following, the spiking BP learning sequence is explained. Similar to the case of the conventional BP algorithm, the learning process of all learning patterns is repeated in the spiking BP until the network error is reduced to a desired value. Fig. 9 shows the flowchart of the learning sequence in the case where the learning patterns are A, B, and C. First, learn and teach signals for pattern A are input. Then, the weights of the synapses are updated to reduce network error E. Next, the operations for patterns B and C are carried out in a same manner. These processes are repeated before E becomes lower than a predetermined value E_{given}. We call the repetition as loops. This type of learning is a type of online learning rather than batch learning.

Next, we explain the details of the learning of each pattern, which corresponds to, for example, the box labeled learn pattern A in Fig. 9. The learning is divided into two main steps, as shown in Fig. 9. In the first step, only the weight of the output-layer neuron is updated. In the second step, the weights of both output- and hidden-layer synapses are updated. The time chart for learning of the patterns is shown in Fig. 10.

To start the first step, the learn signal as voltage pulses V_h^{in} and teach signal as voltage pulses V_j^d are input to the input layer of the SNN. For V_h^{in}, we can use the rectangular-shaped pulse whose width and input timing are

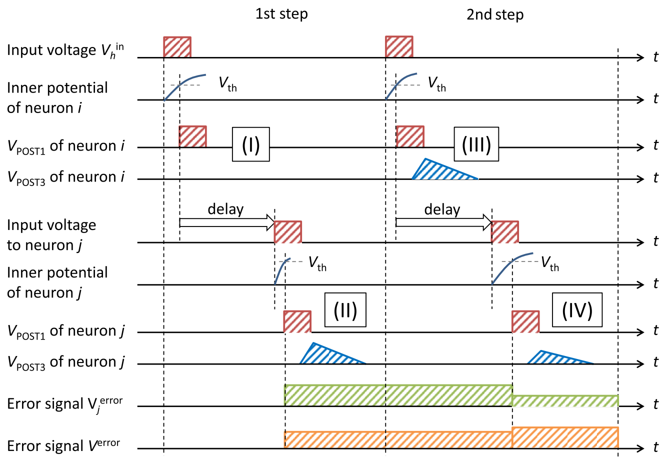
1st step
2nd step
Input voltage V_h^{in}
Inner potential of neuron i
V_{th}
V_{POST1} of neuron i
(I)
(III)
V_{POST3} of neuron i
delay
Input voltage to neuron j
Inner potential of neuron j
V_{POST1} of neuron j
(II)
(IV)
V_{POST3} of neuron j
Error signal V_j^{error}
Error signal V^{error}
t

Fig. 10. Schematic of the time chart during the first and second steps.

T^{in} and t_h^{in}, respectively. The V_h^{in} pulses act as prepulses in the hidden-layer neurons. These pulses can induce the output of postpulses (area labeled as I in Fig. 10). We note that at this moment, V_{error}, which determines the amplitude of V_{POST3} is zero, resulting in $V_{\text{POST3}}^{\text{MAX}} = 0$. As shown in Fig. 5(b) and (c), for $V_{\text{POST}}^{\text{MAX}} = 0$, the conductance change is zero and is independent from Δt, which means that the weight of the hidden-layer synapse does not change.

The postpulses from the hidden layer transmitted to the output layer, which induces the output of the postpulses from the output layer neurons (area labeled as II in Fig. 10). At this moment on the other hand, V_{POST3} of the output-layer neuron is not zero. The input of prepulse2 at $V_{\text{POST3}} \neq 0$ modulates the synaptic weight of the output layer (Fig. 5), i.e., the output neuron can be learned by the input in the first step.

In the second step, the weights of both hidden-layer synapse and output-layer synapse are updated. Because error signals V_j^{error} and V^{error} have already been generated at the first step, V_{POST3}s, which are proportional to the error signals are output when the inner potentials of any neurons exceed a threshold (areas labeled as III and IV in Fig. 10). Therefore, the weight of any synapse is updated depending on Δt, as shown in Fig. 5(c).

In this spiking BP method and circuits, signals are applied to all synapses in the same layer to update the weight. Thus, in principle, the time required to perform the first and second steps does not change irrespective of how many neurons are present in a layer as long as the number of layers is the same. Because the conventional method requires more learning time in networks with more neurons, our method is advantageous, especially when the number of neurons is large.

D. Benchmark Learning

We demonstrate the XOR learning using our spiking BP as a benchmark for testing out the learning model and the SNN circuit. The XOR function is a fundamental example of a nonlinear problem. We have known that a network needs a hidden layer to learn a nonlinear function, such as XOR. Therefore, the function is well used as a benchmark for learning performance.

TABLE I

(a) INPUT AND DESIRED OUTPUT OF THE XOR PROBLEM. (b) INPUT TIMING OF THE INPUT AND TEACH PULSES FOR THE XOR LEARNING IN OUR SPIKING ERROR BP

(a)

Input 1	Input 2	Desired output
0	0	0
0	1	1
1	0	1
1	1	0

(b)

t_1^{in}	t_2^{in}	t_3^{in}	t_1^{d}
6 ms	6 ms	0 ms	16 ms
6 ms	0 ms	0 ms	10 ms
0 ms	6 ms	0 ms	10 ms
0 ms	0 ms	0 ms	16 ms

The XOR data have two binary inputs and one desired binary output. The data set consists of four training samples, as listed in Table I(a). Similar to the previous studies based on software simulation [23], [24], we employ the linear coding. In particular, the input value one is coded as an early spike input t ($t = 0$ ms), whereas zero is coded as a late spike input at $t = 6$ ms. In this demonstration, we add a bias input with an input timing at $t = 0$ ms corresponding to the value of one to improve the learning convergence, which is a well-known technique in the BP. The output values zero and one are coded as an early output spike at $t = 10$ ms and a late spike at $t = 16$ ms, respectively. This spike timing coding for the XOR learning is listed in Table I(b). We use a set of four input pulses composed of two inputs, one bias input, and one input of the desired output to express one training sample. When we start the learning, the four types of pulse sets are sequentially applied to the network. In particular, a pulse set corresponding to one training sample is applied twice, where the double inputs represent the first and second steps. Subsequently, the input of the pulse sets for the other training samples follows in the same manner. These procedures correspond to the first loop. Here, if the network error is more than the predetermined value, the second loop is started, which is the same as the first loop.

For the learning, we use the BP network composed of three input terminals, five hidden neurons, and one output neuron. One of the input terminals in the input layer corresponds to the bias spike. Each of the three input terminals is connected to all neurons in the hidden layer. Each hidden-layer neuron circuit is connected to all output-layer neuron circuits. The number of synapses connecting the neuron circuits is 16, similar to that in [23] and [24]. The 16 synapse circuits have their own delay times, which are $d^k = k$ ms ($k = 1, 2, \ldots, 16$). V_{sup} in the synapse circuit is assumed to be -0.1 and 0.1 V for the exciter and inhibitor synapses, respectively. The resistance and capacitance of the resistor and the capacitor in the integrator are $R = 1$ MΩ and $C = 1$ nF, respectively. The input pulses V^{in} and V_{POST1} of each neuron are square-shaped pulses with $T^{\text{in}} = 7$ ms and $V^{\text{in}} = 10$ V, whereas the teach signal V^d has a liner slope with $T^d = 20$ ms and $V^d_{\text{MAX}} = 3$ V, as shown in Fig. 6(b). Fig. 7(b) shows that V_{POST1} and V_{POST2} of each neuron are square-shaped pulses with $T_{\text{POST1}} = 7$ ms and $T_{\text{POST2}} = 10$ μs, respectively. V_{POST3} is an asymmetric triangular-shaped pulse, where $T^{\text{RISE}}_{\text{POST3}} = 7$ ms, $T^{\text{DESC}}_{\text{POST3}} = 28$ ms, and $V^{\text{MAX}}_{\text{POST3}} = 3$ V. These parameters are adjusted for the learning to finely work by trial and error.

The whole learning development was examined by numerical calculation. In the calculation, we needed the variation with time of the integrated voltage in all neurons with respect to the input pulses from presynaptic neurons. To calculate the voltage variation, we used the difference equation obtained from

$$V(t+\delta t) = V(t) - \frac{V(t) + R\sum_j i_j(t)}{RC}\delta t \tag{12}$$

where δt is the time step of the simulation, which is sufficiently smaller than the pulsewidth. At the instant when $V(t)$ exceeds the threshold voltage, the resulting output spike timing of each neuron is obtained. The output is again applied to the postsynaptic neuron. Then, the network output, which is the output of the output-layer neuron, is obtained. Here, when the integrated potential of the neuron circuit does not exceed the threshold value for a certain period (30 ms), the neuron is forced to output V_{POST} at a given time, for example, by decreasing the threshold value [35]. This method is different from the other BP methods in terms of the software simulation in which the neuron fires when the inner potential reaches its maximum value, which is described as a heuristic rule [23], [24]. The reason why we use this rule is that in hardware SNNs, the input spikes propagate from the input to the output layer. In other words, the rule where the output timing is determined after verifying whether $V(t)$ has exceeded its threshold in a given period cannot be applied to hardware SNNs.

The pulse output timing can be calculated by the behavior of all neurons. From the pulse timings, the pulse voltage applied to synapses V_{pulse} can be obtained by considering the behavior of the error layers and the selectors in the synapse circuits. The change in conductance ΔG for arbitrary G_{init} and V_{pulse} was obtained by linear interpolation of the nearest data points plotted in Fig. 3(d).

The typical result is shown in Fig. 11. Fig. 11(a) and (b) shows the change in the output timing of the output-layer neuron t_{output} and error E for all learning patterns during the learning, respectively. In this trial, the initial conductance of all 3T-FeMEMs in the synapse circuit is set to a random value <4 μS. Fig. 11(a) shows that t_1^{out} approaches the desired timing, and the resulting errors rapidly drop to 1×10^{-6} s^2

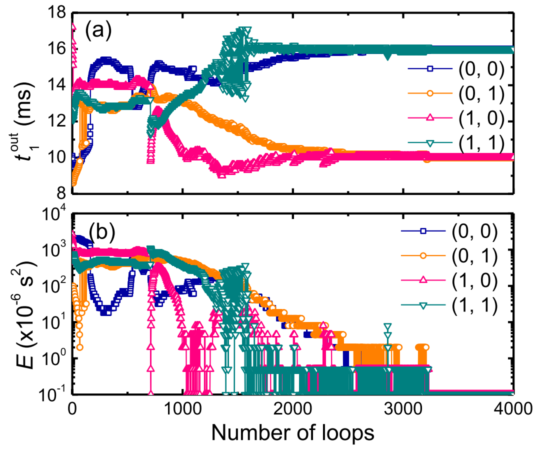

Fig. 11. Changes in the (a) output timing of the output-layer neurons t_1^{out} and (b) network error E during the learning.

in the vicinity of $n \sim 2500$, which means that the learning succeeded at this time. We also performed this simulation by changing the initial synaptic weight values. The convergence rate is ~50%. Even if we introduced a 10% conductance variation for each synapse, which was observed in the fabricated 3T-FeMEM in the same chip, XOR learning was successfully demonstrated, which means that the variation does not greatly affect the convergence speed. As mentioned above, our spiking BP method and circuit enable the spiking neuron-based supervised learning.

IV. Conclusion

In this paper, a simplified error-BP model based on SNNs, which is suitable for hardware implementation, has been proposed. We demonstrated a circuit example to implement the model. In the circuit, 3T-FeMEMs were used to obtain the synapse function in which the analog synaptic weights stored as 3T-FeMEM conductance were adjusted to decrease the network error according to the model. We also utilized the spike-timing learning function of the 3T-FeMEM, which appeared in the expression of the weight adjustment of our model. Using the model and the circuit, we successfully demonstrated an XOR learning as a benchmark for the learning performance evaluation. This BP model and the basic concept for constructing the BP hardware SNN can be applied to the case where other types of memristors are used as synapses. In this method, the learning time for the pattern is, in principle, independent from the number of neurons in each layer, meaning that a high-speed learning function can be expected, especially in large-scale SNNs. Application to medium- or large-scale problem using fabricated prototype circuits is the next challenge. The door has been thus opened for practical applications of SNN with supervised learning by means of this model and SNN circuits with memristor-based synapses.

Acknowledgment

The authors would like to thank Dr. D. Ueda, Dr. E. Fujii, and Dr. A. Omote for the meaningful discussions throughout this paper.

References

[1] C. Mead, *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley, 1989.
[2] S. M. Schuetze, "The discovery of the action potential," *Trends Neurosci.*, vol. 6, pp. 164–168, 1983.
[3] E. R. Kandel, J. H. Schwartz, and T. M. Jessel, Eds., *Principles of Neural Sciences*. New York, NY, USA: Elsevier, 1991.
[4] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[5] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.
[6] S. Ramakrishnan, P. E. Hasler, and C. Gordon, "Floating gate synapses with spike-time-dependent plasticity," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 3, pp. 244–252, Jun. 2011.
[7] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, Anaheim, CA, USA, Jun. 2008, pp. 85–92.
[8] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.
[9] A. Chanthbouala *et al.*, "A ferroelectric memristor," *Nature Mater.*, vol. 11, pp. 860–864, Sep. 2012.
[10] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, "Spin-based neuron model with domain-wall magnets as synapse," *IEEE Trans. Nanotechnol.*, vol. 11, no. 4, pp. 843–853, Jul. 2012.
[11] J. Shi, S. D. Ha, Y. Zhou, F. Schoofs, and S. Ramanathan, "A correlated nickelate synaptic transistor," *Nature Commun.*, vol. 4, Oct. 2013, Art. ID 2676.
[12] L. Q. Zhu, C. J. Wan, L. Q. Guo, Y. Shi, and Q. Wan, "Artificial synapse network on inorganic proton conductor for neuromorphic systems," *Nature Commun.*, vol. 5, Jan. 2014, Art. ID 3158.
[13] B. Rajendran *et al.*, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *IEEE Trans. Electron Devices*, vol. 60, no. 1, pp. 246–253, Jan. 2013.
[14] Y. Kato, Y. Kaneko, H. Tanaka, and Y. Shimada, "Nonvolatile memory using epitaxially grown composite-oxide-film technology," *Jpn. J. Appl. Phys.*, vol. 47, no. 4S, pp. 2719–2724, 2008.
[15] Y. Kaneko, Y. Nishitani, M. Ueda, E. Tokumitsu, and E. Fujii, "A 60 nm channel length ferroelectric-gate field-effect transistor capable of fast switching and multilevel programming," *Appl. Phys. Lett.*, vol. 99, no. 18, pp. 182902-1–182902-3, 2011.
[16] M. Ueda, Y. Kaneko, Y. Nishitani, and E. Fujii, "A neural network circuit using persistent interfacial conducting heterostructures," *J. Appl. Phys.*, vol. 110, no. 8, pp. 086104-1–086104-3, 2011.
[17] Y. Nishitani, Y. Kaneko, M. Ueda, T. Morie, and E. Fujii, "Three-terminal ferroelectric synapse device with concurrent learning function for artificial neural networks," *J. Appl. Phys.*, vol. 111, no. 12, pp. 124108-1–124108-6, 2012.
[18] Y. Nishitani, Y. Kaneko, M. Ueda, E. Fujii, and A. Tsujimura, "Dynamic observation of brain-like learning in a ferroelectric synapse device," *Jpn. J. Appl. Phys.*, vol. 52, no. 4S, pp. 04CE06-1–04CE06-6, 2013.
[19] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, Dec. 2012, pp. 10.4.1–10.4.4.
[20] O. Bichler, M. Suri, D. Querlionz, D. Vuillaume, B. DeSalvo, and C. Gamrat, "Visual pattern extraction using energy-efficient '2-PCM synapse' neuromorphic architecture," *IEEE Trans. Electron Devices*, vol. 59, no. 8, pp. 2206–2214, Aug. 2012.
[21] D. Kuzum, R. G. D. Jeyasingh, S. Yu, and H.-S. P. Wong, "Low-energy robust neuromorphic computation using synaptic devices," *IEEE Trans. Electron Devices*, vol. 59, no. 12, pp. 3489–3494, Dec. 2012.
[22] Y. Kaneko, Y. Nishitani, and M. Ueda, "Ferroelectric artificial synapses for recognition of a multishaded image," *IEEE Trans. Electron Devices*, vol. 61, no. 8, pp. 2827–2833, Aug. 2014.
[23] S. M. Bohte, J. N. Kok, and H. L. Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, 2002.
[24] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for EEG classification and epilepsy and seizure detection," *Integr. Comput.-Aided Eng.*, vol. 14, no. 3, pp. 187–212, 2007.
[25] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural Netw.*, vol. 22, no. 10, pp. 1419–1431, 2009.
[26] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, San Francisco, CA, USA, Mar. 1993, pp. 586–591.
[27] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm," in *Proc. Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, Jul. 2006, pp. 3970–3977.
[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[29] H. Ishiwara, "Proposal of adaptive-learning neuron circuits with ferroelectric analog-memory weights," *Jpn. J. Appl. Phys.*, vol. 32, no. 1B, pp. 442–446, 1993.
[30] S.-M. Yoon, E. Tokumitsu, and H. Ishiwara, "An electrically modifiable synapse array composed of metal-ferroelectric-semiconductor (MFS) FET's using $SrBi_2Ta_2O_9$ thin films," *IEEE Electron Device Lett.*, vol. 20, no. 5, pp. 229–231, May 1999.
[31] S.-M. Yoon, E. Tokumitsu, and H. Ishiwara, "Adaptive-learning neuron integrated circuits using metal-ferroelectric ($SrBi_2Ta_2O_9$)-semiconductor (MFS) FET's," *IEEE Electron Device Lett.*, vol. 20, no. 10, pp. 526–528, Oct. 1999.

[32] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, 1998.

[33] M. Nishiyama, K. Hong, K. Mikoshiba, M. M. Poo, and K. Kato, "Calcium stores regulate the polarity and input specificity of synaptic modification," *Nature*, vol. 408, no. 6812, pp. 584–588, 2000.

[34] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: Taming the beast," *Nature Neurosci.*, vol. 3, no. 11, pp. 1178–1183, 2000.

[35] H. Tanaka, T. Morie, and K. Aihara, "A CMOS spiking neural network circuit with symmetric/asymmetric STDP function," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E92-A, no. 7, pp. 1690–1698, 2009.

Yu Nishitani received the Ph.D. degree in electrical engineering from Tohoku University, Sendai, Japan, in 2010. His Ph.D. research focused on semiconductor spintronic materials and devices.

He has been with Panasonic Corporation, Kyoto, Japan, since 2010. He is currently involved in ferroelectric thin films and devices for neuromorphic applications.

Yukihiro Kaneko (M'10) received the B.S. and M.S. degrees in engineering from Nagoya University, Nagoya, Japan, in 2003 and 2005, respectively, and the Ph.D. degree in electronics and applied physics from the Tokyo Institute of Technology, Tokyo, Japan, in 2012.

He joined Panasonic Corporation, Kyoto, Japan, in 2005. His current research interests include the designing, fabrication, characterization, modeling, and applications of semiconductor and functional oxide devices.

Michihito Ueda received the B.Eng., M.Eng., and Ph.D. degrees from Kyoto University, Kyoto, Japan, in 1988, 1992, and 2006, respectively.

He joined Panasonic Corporation, Kyoto, in 1992. His current research interests include micromachined optical devices, semiconductor memory devices using high-k thin films, neural network devices using ferroelectric thin films, and stochastic information processing.