Hebbian Learning in FPGA Silicon Neuronal Network

Jing Li^{a*}, Yuichi Katori^{b,c}, Takashi Kohno^b,

^aGraduate School of Engineering, The University of Tokyo, Tokyo, Japan

^bInstitute of Industrial Science, The University of Tokyo, Tokyo, Japan

^cFIRST, Aihara Innovative Mathematical Modelling Project, Japan Science and Technology Agency, Tokyo, Japan

*Corresponding Author: li@sat.t.u-tokyo.ac.jp

Abstract

This paper describes a digital silicon neuronal network trained by the Hebbian learning rule that can execute the auto-associative memory. In our previous work, a fully connected network of 256 silicon neurons based on the digital spiking silicon neuron (DSSN) model and kinetic-model-based silicon synapses were implemented. In this work, we added circuit modules that append Hebbian learning function and fitted it to a Xilinx Virtex 6 XC6VSX315T FPGA device. The performances of auto-associative memory with several spike-time-dependent Hebbian learning rules and the correlation rule are compared. The results show that Hebbian learning rules that model both synaptic potentiation and depression improve the retrieval probability in our silicon neuronal network.

Keywords: Hebbian learning, silicon neuronal network, FPGA, associative memory.

1. Introduction

Sensory experiences are thought to be configuring the nerve system. The synaptic plasticity underlies this phenomenon, which refers to the change of the connection strength between two neurons. From the view point of molecular biology, the modification of the neurotransmitter release and the receptors can cause the plasticity of the synapses^(1,2). The modification that lasts for long time has two types: long-term potentiation (LTP) and long-term depression (LTD) which respectively leads to reinforced and weakened synapses. Physiological experiments revealed LTP and LTD in the hippocampus and the cortex^(3,4). The Hebbian learning rule firstly described that the connection from one cell to another is facilitated if their activities are correlated⁽⁵⁾. Several artificial neural networks

with the Hebbian rule are simulated on PCs or designed by electronic circuits^(7,8). However, these kinds of systems mimic restricted aspects of the nervous system because they simplify the timing information of spikes by using discrete time models such as the two-state neuron model. Continuous-time models such as integrate-and-fire (I&F) model are chosen to build neuronal networks in order to overcome the above issue^(9,10).

We proposed a silicon neuronal network implemented in an FPGA device. It is composed of silicon neurons and synapses that are optimized for implementation by digital arithmetical circuits and capable of real-time operation in entry-level FPGA devices. The model of these silicon neurons was designed in the viewpoint of nonlinear dynamics and can reproduce the graded spike response in the Class II neurons in the Hodgkin's classification⁽¹¹⁾. The silicon synapse model describes the transmitter release, receptor activation and the generation of synaptic current. In this work, we train our silicon neuronal network by the Hebbian learning rule and estimate the performance of associative memory.

Section 2 explains the model of the Hebbian learning rule used in this paper. Section 3 presents the implementation of the circuit of our silicon neuronal network with the Hebbian learning ability. Section 4 describes the associative memory we performed and compares their performances. Conclusion and the future plan of our work are presented in section 5.

2. Hebbian learning rules

The Hebbian learning rule, proposed in 1949, supposes that if input from neuron A causes the firing of neuron B, then the synapse from A to B tends to be strengthened⁽⁵⁾. Exponential learning windows are a natural choice for

describing the Hebbian rule for spike-timing dependent learning⁽⁶⁾. A most simple formulation is,

$$\Delta W = A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right), \quad (1)$$

where ΔW is the change of synaptic weight W and Δt is the time difference between the pre- and post-synaptic spikes. Parameters A_+ and τ_+ represent the amplitude and the decay constant.

The above basic Hebbian rule only considers the synaptic potentiation. The synaptic depression, which is also an important process of learning, can be modeled by exponential weight decay⁽⁶⁾. A simple formulation of a model with both of the potentiation and the depression is as follows.

$$\Delta W = A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - A_- \exp\left(\frac{-|\Delta t|}{\tau_-}\right), \quad (2)$$

where A_- and τ_- are the amplitude and the decay constant for the synaptic depression. We also use more simplified model as in Eq. (3) because simpler models require less hardware resources.

$$\Delta W = A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - A_-. \quad (3)$$

Figure 1 shows the learning windows of Eqs. (1), (2), and (3). Learning rules in Eqs. (2) and (3) (we refer to them as anti-combined (AC) and decay-combined (DC) rules, respectively) can lead to decreasing synaptic change while the basic one cannot. When the time difference becomes large, the decreasing rate is constant in the DC rule while it decays in the AC rule.

Fig. 1. Learning windows of the basic Hebbian rule shown in Eq. (1) with $A_+ = 1, \tau_+ = 10\text{ms}$ (blue), the AC rule shown in Eq. (2) with $A_+ = 1, A_- = 0.5, \tau_+ = 10\text{ms}, \tau_- = 20\text{ms}$ (green) and the DC rule shown in Eq. (3) with $A_+ = 1, A_- = 0.1, \tau_+ = 10\text{ms}$ (cyan).

3. FPGA silicon neuronal network

3.1 Model of silicon neuronal network

The model of our silicon neuronal network is composed of the Digital Spiking Silicon Neuron (DSSN) model⁽¹²⁾ and a silicon synapse model proposed in our previous work⁽¹³⁾. The DSSN model is two-variable differential equations optimized for digital arithmetic circuit implementation. Its dynamics can be configured by selecting appropriate parameter values. Both Class I and II neurons in the Hodgkin's classification⁽¹⁴⁾ can be reproduced by the two sets of parameter values listed in appendix. The equations of this model are,

$$\frac{dv}{dt} = \frac{\varphi}{\tau} (f(v) - n + I_0 + I_{stim}), \quad (4)$$

$$\frac{dn}{dt} = \frac{1}{\tau} (g(v) - n), \quad (5)$$

$$f(v) = \begin{cases} a_n(v + b_n)^2 - c_n & \text{when } v < 0, \\ -a_p(v - b_p)^2 + c_p & \text{when } v \geq 0, \end{cases} \quad (6)$$

$$g(v) = \begin{cases} k_n(v - p_n)^2 + q_n & \text{when } v < r, \\ k_p(v - p_p)^2 + q_p & \text{when } v \geq r, \end{cases} \quad (7)$$

where v and n denote the membrane potential and a slow variable that represents the activity of ionic channels, respectively. Parameter I_0 is a bias constant and I_{stim} is the weighed sum of postsynaptic input from silicon synapse. Parameters φ and τ are time constants. Parameters $r, a_x, b_x, c_x, k_x, p_x, q_x$ for $x=n$ and p , are the parameters that are responsible to the dynamical behaviors.

The synaptic model is a kinetic-model-based one that describes the transmitter release and the postsynaptic current generation in response to the presynaptic membrane potential. Its equation is,

$$\frac{dI_s}{dt} = \begin{cases} \alpha(1 - I_s) & [T] = 1, \\ -\beta I_s & [T] = 0, \end{cases} \quad (8)$$

where, I_s and $[T]$ are the postsynaptic current and the amount of the released transmitter per impinging spike, respectively. Parameters α and β are the forward and backward rate constants which represent the rate of the receptors transitioning from the closed state to the open state and its opposite, respectively. The DSSN model in the Class II mode responds to stimulus input in a graded manner: the peak of v in a spike varies depending on the strength of stimulus current. The amount of the released transmitter is controlled by the stimulus current due to this graded response property.

The stimulus input to the i -th silicon neuron I_{stim}^i is calculated by adding the postsynaptic currents as follows,

$$I_{stim}^i = c \sum_{j=1}^N W_{i,j} I_s^j, \quad (9)$$

where $W_{i,j}$ is the connection weight from the j -th to the i -th silicon neurons. And c is a coefficient used to restrict I_{stim}^i to an appropriate range and ensure that neurons fire regularly. The values of parameters for Class I and II neurons are listed in the appendix.

3.2 FPGA implementation

We designed a fully connected network which has the pipelined and parallel structure as shown in Fig. 2. It contains multiple (N_f) silicon neuronal network modules (SNNMs). Each SNNM calculates the system variables for N_v silicon neurons in sequence. The SNNM is composed of four units: a DSSN, a silicon synapse, an accumulator and a learning unit. The DSSN and the silicon synapse units update the states of the silicon neurons and the silicon synapses in response to stimulus input I_{stim}^i from accumulator unit. The learning unit modifies the connection weights $W_{i,x}$ according to the Hebbian learning rule when it detects spikes.

Fig. 2. Structure of our silicon neuronal network

In order to reduce the hardware cost of the learning unit, two exponential functions are realized by Look Up Tables (LUTs) instead of multipliers. The weights are 12-bit signed fixed point numbers with 11-bit fraction part. We constructed a 256-neuron silicon neuronal network ($N_f = 16, N_v = 16$) in a Virtex 6 XC6VSX315T FPGA

device. Table 1 lists the device utilization of the circuit synthesized by the Xilinx ISE tool. The fixed point number, shift operation for avoiding multiplier and LUTs for exponential functions are adopted to reduce the hardware utilization. The fast speed of our network can reach up to 97.972MHz according to the synthesize report.

Table 1. Device utilization on FPGA device

Logic utilization	Utilization	Available
Slice registers	6,723(1%)	393,600
LUTs	46,833(23%)	196,800
Block RAMs	81(11%)	704
DSPs	16(1%)	1,344

4. Auto-associative memory

4.1 Configuration of the network

The auto-associative memory retrieves a stored pattern with the help of its fragments. It was originally investigated in a fully connected neuronal networks with the correlation learning⁽¹⁵⁾. Then some online learning rules such as Hebbian learning rules were introduced^(16,17,18). We executed the auto-associative memory by our silicon neuronal network with the correlation learning in the previous work⁽¹³⁾. In this paper, we report the performance in this task of our silicon neuronal network equipped with the Hebbian learning ability.

Fig. 3. Stored patterns (A) and a set of input patterns (B) generated by randomly reversing pixels of stored pattern 1.

We put our network firstly in the learning stage and then in the retrieval stage. In the learning stage, four stored patterns shown in Fig.3(A) and their reversed patterns were presented to our network alternately in the following order: $1, \bar{1}, 1, \bar{1} \dots, 2, \bar{2}, 2, \bar{2} \dots, 3, \bar{3}, 3, \bar{3} \dots, 4, \bar{4}, 4, \bar{4} \dots, 1, \bar{1}, 1, \bar{1} \dots$. Here, the reversed patterns ($\bar{x}$ corresponds to x) are used to increase the contrast between two pixels with different colors to ensure an efficient learning by the Hebbian rule. The coupled patterns x and $\bar{x}$ (for $x = 1, 2, 3, 4$) repeat 8 times and interval between two different coupled patterns is set to 24ms. This period was selected so that stimuli avoid

Fig. 4. Transition of connection weights with L1-5 rules.

the refractory period of silicon neurons induced by the last stimulus. One pixel in patterns corresponds to one silicon neuron in our network. A pulse current is added to the neuron if its corresponding pixel is black. The amplitude and the high and the rest periods of this pulse current are 0.125, 6ms and 18ms, respectively. In this paper, each silicon neuron is in the Class I mode. We designed 5 types of Hebbian learning rules L1 to L5 based on Eqs. (1)-(3) with soft or hard bounds that limit the weights in a finite range. Their update equations are,

$$\text{L1: } \Delta W = K_a(W)A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right), \quad (10)$$

$$\text{L2: } \Delta W = K_a(W)A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - K_i(W)A_- \exp\left(\frac{-|\Delta t|}{\tau_-}\right), \quad (11)$$

$$\text{L3: } \Delta W = K_a(W)A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - K_i(W)A_- \exp\left(\frac{-|\Delta t|}{\tau_-}\right), \quad (12)$$

$$\text{L4: } \Delta W = K_a(W)K_i(W)[A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - A_- \exp\left(\frac{-|\Delta t|}{\tau_-}\right)], \quad (13)$$

$$\text{L5: } \Delta W = A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - A_- \exp\left(\frac{-|\Delta t|}{\tau_-}\right). \quad (14)$$

Here, $K_a(W) = \frac{W_{max}-W}{W_{max}-W_{min}}$ and $K_i(W) = \frac{W-W_{min}}{W_{max}-W_{min}}$ are soft bounds in L1-4. The hard bound in L5 is that the update of weights is stopped when one of the weights reaches a limit, W_{min} or W_{max} which equal to -1 and 1 respectively. The weights are updated from their initial value 0 according to the above 5 Hebbian learning rules and the update process stops when weights are unchanged for 600ms. The values of parameters in Eqs. (10)-(14) are listed in the appendix.

4.2 Simulation results

We simulated our silicon neuronal network with C++ programs to estimate the performance. Figure 4 shows the transition of connection weights to the silicon neuron 1 ($W_{1,1}, \dots, W_{1,256}$) when our network is trained by the above 5 Hebbian learning rules. With the basic Hebbian learning rule with soft bounds (L1), the weights increased and reached their maximum value, 1 (Fig. 4(A)). The weights also increased based on the DC Hebbian learning rule with soft bounds (L2) in Fig. 4(B). However, the decay factor A_- and its relevant soft bound K_i slowed down their growth speed and decreased their maximum values. By the AC Hebbian learning rule with two soft bounds

Fig. 5. Averaged difference of weights between the Hebbian learning rules (L1-5) and the correlation learning rule.

corresponding to LTP and LTD (L3), the weights stayed within the range of $[-0.4, 0.2]$ (Fig. 4(C)). In another AC Hebbian learning rule (L4), two soft bounds commonly effect on LTP and LTD together. This updated some of the weights to their boundary values (Fig. 4(D)). The AC Hebbian learning rule with hard bound (L5) modified the weights more quickly than the soft bounds rules (L1-4) (Fig. 4(E)).

We have verified that our silicon neuronal network with the correlation learning rule can successfully execute auto-associative memory in our previous work⁽¹³⁾. Here, the connection weights were calculated by the correlation learning rule according to the following equations:

$$W_{ij} = \begin{cases} \frac{1}{p} \sum_{u=1}^p x_i^u x_j^u & \text{when } i \neq j, \\ 0 & \text{when } i = j, \end{cases} \quad (15)$$

where p is the number of stored patterns, which is 4 in this work, and x_i^u represents the i -th pixel in the u -th stored pattern, which is 1 or -1 when it is black or white, respectively.

Figure 5 plots the averaged difference of weights between the Hebbian learning rules (L1-5) and the correlation learning rule (dW). For L1 and L2 rules, dW grew monotonically as the learning proceeds. It indicates the weights produced by L1 and L2 are completely different from those by the correlation rule (Fig. 5(A, B)). For L3, dW decreased but remained over 0.3 (Fig. 5(C)). With L4, the weights approached to the results of the correlation learning in the early phase then dW increased upto about 0.3 (Fig. 5(D)). With L5, the weights approached to the correlation learning results and stayed close to it but maintained small difference (Fig. 5(E)).

In the retrieval stage, input patterns with different levels of errors (Fig. 3(B)) were presented to our network for 120ms by adding pulse currents which have the same parameter setting to the one in the learning stage. Here, 120ms (5 spikes) allowed our network to receive sufficient amount of stimulus input. The time length of the retrieval stage was 800ms, which was sufficiently long for silicon neurons to reach stable states. We calculated overlap index to evaluate the similarity between the state of silicon neurons and the stored patterns. The overlap M_u between the state of silicon neurons and the u -th stored pattern is obtained by the following equations:

$$M_u(t) = \frac{1}{N} \left| \sum_{j=1}^N x_j^u \exp(i\phi_j(t)) \right|, \quad (16)$$

$$\phi_j(t) = 2\pi k + 2\pi \frac{t - t_j^k}{t_j^{k+1} - t_j^k}, (t_j^k \leq t < t_j^{k+1}). \quad (17)$$

where N is the number of silicon neurons, x_j^u represents the j -th pixel in the u -th stored patterns with value of 1 or -1 that correspond to black or white, respectively. Function $\phi_j(t)$ is the phase value of the j -th neuron, which is defined by Eq. (17). And t_j^k is the start time of the k -th spike for the j -th silicon neuron. When the network successfully retrieves the stored pattern, the relevant overlap equals to 1. Figure 6 shows the raster plot (A) and overlaps (B) when our silicon neuronal network is trained by the L5 rule in the learning stage and the pattern with 20% errors in Fig. 3(B) was applied in the retrieval stage. The input pattern was observed for 5 spikes firstly and then stored pattern 1 and its reversed pattern alternately appeared from 0.4s (Fig. 6(A)). Then the overlap M_1 reached to 1 when the stored pattern 1 was retrieved in the network (Fig. 6(B)).

Fig. 6. Raster plot (50 out of 256 neurons are displayed) and overlap of the memory retrieval in the Class I mode network trained by the L5 rule. The input pattern includes 20% errors shown in Fig. 3(B).

We calculated the retrieval probability in the auto-associative memory when our silicon neuronal network is trained by the Hebbian learning rules (L1)-(L5) and the correlation learning. Totally 100 input patterns were applied; 10 patterns with 10 different error rates from 5% to 50% by 5% step. We counted the success of memory retrieval if M_u reached 1 at the end of the retrieval stage (Fig. 7). The L1-3 rules could not retrieve at all. The L4 and L5 rules had better performance than the correlation learning rule when the error rate was larger than 15%. The L5 rule showed higher retrieval probability than L4 rule when input pattern contained 35% errors.

Fig. 7. Retrieval probability with the correlation learning rule and the Hebbian learning rules.

5. Conclusions

The spike-time-dependent Hebbian learning modifies the synaptic weights according to the difference of spike timing between pre- and post-synaptic neurons. Our silicon neuronal network succeeded to execute auto-associative memory based on this learning concept.

We trained our silicon neuronal network with 5 types of Hebbian learning rules and evaluated them by the auto-associative memory. The simulation results suggest that the AC Hebbian learning rules (L4, L5) boost the performance of the auto-associative memory in our network in comparison to the correlation learning. The difference of performance between L4 and L5 is not large in comparison to this boost, and we expect it is brought by a kind of distortion of ΔW over the distribution of the weights caused by the soft bounding terms. This will to be analyzed in detail in our future works. Our silicon neuronal network model is optimized for implementation by digital arithmetic circuits. Moreover, many effects were made in the circuit design to reduce the number of LUTs for the Hebbian

learning module. The system is estimated to be able to run at up to 97.972MHz, which realizes over 35 times faster than real-time of the nerve system.

In the future, we will complete our VHDL design and download our silicon neuronal network to an FPGA device and verify its functionality. The spike-timing-dependent plasticity (STDP) with asymmetric learning curves is another popular learning rule that modifies the connection weights depending on the difference between the spike timings of pre- and post-synaptic neurons. We will evaluate the performance of our silicon neuronal network trained with this rule by the spatio-temporal memory. We will also expand our network to 1000 neurons to enhance the amount of the storable data. Because the sequential aspect of our silicon neuronal network model is increased by larger number of silicon neurons in a network, the calculation speed in FPGA devices will be reduced. Though, we expect we can keep about 35 times faster than real-time by exploiting parallelism of the FPGA circuits.

Acknowledgment

This work was partially supported by JSPS Grant-in-Aid for scientific Exploratory Research Number 25240045 and Grant-in-Aid for Young Scientist (B) 25730175. This work was also partially supported by Aihara Innovative Mathematical Modelling Project, the Japan Society for the Promotion of Science (JSPS) through its "Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program)."

References

- (1) Cline H.T., Constantine-Paton, M. and Debski., E : "Patterned activity, synaptic convergence and the NMDA receptor in developing visual pathways", *Neurosci.*, 13, 129–154, 1990
- (2) Caillard O. Gaiarsa, J.L. and Y. Ben-Ari. : "Long-term plasticity at gabaergic and glycinergic synapses: mechanisms and functional significance", *Trends Neurosci.*, 25, 564–570, 2002
- (3) T.V. Bliss and G.L. Collingridge. : "Synaptic model of memory: long-term potentiation in the hippocampus", *Nature*, 361, 31–39, 1993
- (4) Burke J.P. Smith-D. Fregnac, Y. and M.J. Friedlander. : "Temporal covariance of pre and postsynaptic activity regulates functional connectivity in the visual cortex", *J. Neurophysiol.*, 71, 1403–1421, 1994
- (5) Hebb, D. O : "Organization of Behavior" John Wiley & Sons, New York. 1949
- (6) Wulfram Gerstner and Werner M. Kistler : "Spiking neuron models single neurons, populations, plasticity", Cambridge university press, 2002
- (7) James, Ting-Ho Lo : "Unsupervised Hebbian learning by recurrent multilayer neural networks for temporal hierarchical pattern recognition", *Annual Conference on Information Sciences and Systems (CISS)*, 1–6, 2010
- (8) Oniga, S., Tisan, A., Mic, D., Buchman A. and Vida-Ratiu, A. : "Optimizing FPGA Implementation of Feed-Forward Neural Networks", *International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, 31–36, 2008
- (9) Andrés Upegui, Carlos Andrés Peña-Reyes, Eduardo Sánchez : "A hardware implementation of a network of functional spiking neurons with Hebbian learning", *Biologically Inspired Approaches to Advanced Information Technology*, Springer Berlin Heidelberg, 233–243, 2004
- (10) H. Kolle Riis and P. Hafliger : "spike based learning with weak multi-level static memory", *International Symposium on Circuits and Systems (ISCAS)*, 5, 393–396, 2004
- (11) John Rinzel and Bard Ermentrout : "Analysis of excitability and oscillations", *Methods in neuronal modeling*, MIT Press, 1989
- (12) Takashi Kohno and Kazuyuki Aihara : "Digital spiking silicon neuron: concept and behaviors in GJ-coupled network", *International Symposium on Artificial Life and Robotics*, OS3-6, 2007
- (13) Jing Li, Yuichi Katori and Takashi Kohno : "An FPGA-based silicon neuronal network with selectable excitability silicon neurons", *Frontiers in neuroscience*, 6, 183, 2012
- (14) Hodgkin, A.L. : "The local electric changes associated with repetitive action in a non-medullated axon", *J. Physiol.*, 107, 165–181, 1948
- (15) Hopfield, J.J. : "Neural networks and physical systems with emergent collective computational abilities", *Proc. Natl. Acad. Sci.*, 79, 2254–2258, 1982
- (16) Tsodyks, M.B. : "Associative memory in neural networks with the Hebbian learning rule", *Modern Physics Letters B*, 3, 555–560, 1989
- (17) Molter, C., Salihoglu, U. and Bersini, H. : "Learning cycles brings chaos in continuous hopfield networks", *IJCNN*, 2, 770–775, 2005

(18)Chen, J., Liu, Y., Chen, Q. and Cui, J. : “A neural network model for concept formation”, ICNC, 3, 24-28, 2008

Appendix

Common parameters for Class I and II mode

Par.	Value	Par.	Value
a_n	8.0	k_p	16
b_n	0.25	p_p	$2^{-5} - 2^{-2}$
c_n	0.5	q_p	-0.6875
a_p	8.0	τ	0.003
b_p	0.25	α	83.3
c_p	0.5	β	333.3

Different parameters for Class I and II mode

Par.		Value	Par.		Value
k_n	I	2.0	r	I	-0.205357142
	II	4.0		II	-0.104166
p_n	I	$-2^{-2} - 2^{-4}$	I_0	I	-0.205
	II	$-2^{-1} - 2^{-4}$		II	-0.23
q_n	I	-0.705795601	c	I	0.060546875
	II	-1.317708517		II	0.03125
φ	I	1.0			
	II	0.5			

Parameters for Hebbian learning rules L1-5

	L1	L2	L3	L4	L5
A_+	2^{-6}	2^{-6}	2^{-6}	2^{-6}	2^{-6}
A_-		2^{-7}	$2^{-7} + 2^{-8}$	$2^{-7} + 2^{-8}$	$2^{-7} + 2^{-8}$
τ_+ (ms)	11.25	11.25	11.25	11.25	11.25
τ_- (ms)			22.5	22.5	22.5
$W_{\max}$	1	1	1	1	
$W_{\min}$	-1	-1	-1	-1	