2016 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2016, Yugawara, Japan, November 27th-30th, 2016

Spiking neural network simulation on FPGAs with automatic and intensive pipelining

Taro Kawao† Masato Neishi Tomohiro Okamoto Amir Masoud Gharehbaghi
Takashi Kohno Masahiro Fujita

University of Tokyo, Japan
†Now at Renesas Electronics, Japan

Abstract—There are lots of scientific interests in the behaviors of large spiking neural networks. One way to understand the behaviors is to simulate them as fast as possible with customized hardware, such as FPGA. This paper shows highly pipelined implementations of spiking neural networks on FPGA using a high level synthesis tool. Our accelerator allows 256 neurons to operate at 280 times faster than real time brain operations, which is around 8 times faster than the previous reports on similar directions. It is also designed for a multi-FPGA implementation which can simulate up to 3,000 neurons.

1. Introduction

There have been a series of researches on the use of specialized and customized hardware for speeding up particular scientific computations. Completely customized hardware, such as ASIC (Application Specific Integrated Circuit) implemented in silicon does not have any flexibility after fabrication, and there is no way to modify operations even if the target computations need to be changed. On the other hand, programmable and reconfigurable hardware, such as FPGA (Field Programmable Gate Arrays), can change their functionality at anytime, and so it is one of the ideal devices which realize the customized computing machines.

In this paper we show highly pipelined implementation of spiking neural network simulation on FPGA. Neural networks are mathematical models on the functionality of neuron and synapse, and consist of nodes for neurons and edges for synapses. By changing the weights for edges based on learning, various information processing can be realized. By using sufficiently large networks, robust computing, such as computations under various noises, can be achieved for pattern recognition, data classification, and others. There have been developed many FPGA-based spiking neuronal network simulators [1][3][7] [6][2][4].

Here, we use DSSN network model [5] which uses relatively simple computations for ease of hardware implementations but reproduces the behaviors of biological neural networks with sufficient accuracy. Our work is based on the previous works [6][7] but improves the simulation speed by 8 times and also can deal with larger networks. The previous works [6][7] realize the same speed as human brain up to around 1,500 neurons. The scope of our research covers the accelerated simulation of neuronal networks composed of tens of thousands of neurons. This number is comparable to that of the insect brains which realize intelligent and adaptive processing. Thus, speeding up simulations of that size is practically very important.

In our framework, the highly pipelined implementations of spiking neural network simulations are generated through the use of a high level synthesis tool, called Max Compiler [8]. High level synthesis tools can automatically refine designs in high level, such as in C or Java, into the ones in RTL (Register Transfer Level). Once designs in RTL are obtained, there are established tool flows by which FPGA implementations can be automatically obtained. Max Compiler can introduce intensive pipelined designs utilizing their special IPs (pre-designed functional units). The numbers of pipeline stages can be more than 1,000, which means more than 1,000 data parallel operations are processed simultaneously. With the use of high level synthesis tools, high performance neural network simulation on FPGA can be relatively easily realized.

The rest of the paper is organized as follows. In the next section the neural network model we are going to implement is reviewed. Then the previous works are briefly examined in the following section. Our proposed implementation is presented next with its experimental results. The final section gives concluding remarks.

2. Target neural network model

In our target model, there are N neurons which are connected to one another through synapse networks as shown in Figure 1. All neurons are connected to all the other neurons. For each neuron, the input from synapses to the neuron is computed as shown in the expression 5. The computations inside a neuron is shown in the expressions 1, 2, and 3. The output from a neuron to synapses is computed as shown in the expression 4. The learning on the spiking neural network is defined in the expressions 6 and 7. Some of the expressions originally defined as differential equations are converted into the corresponding difference equations.

Here we implement the spiking neural network model called DSSN model [5]. A spike is defined as a narrow

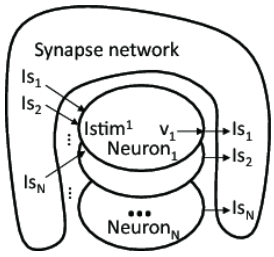

Figure 1: Target neural network

impulse with some amount of voltage swing. The variables v, n, and q are introduced for the modeling. By manipulating the variables, v and n, spikes are generated, and the variable, q adjusts the intervals of spikes. I_0 is a constant and I_{stim} is the weighted sum of synaptic inputs to a neuron and computed as shown in the expression 5. While v is above a threshold, [T] is 1 and else [T] is 0. φ and τ are time constants, and the others, $v_0, \alpha, a_n, a_p, b_n, b_p, c_n, c_p, k_n, k_p, l_n, l_p\ , m_n, m_p$, are control parameters for the behaviors of spikes. For a neuron, I_{stim} is the input and v is the output.

$$v(t+\Delta t) = v(t) + \Delta t \frac{\varphi}{\tau} \times \begin{cases} \left(a_n(v-b_n)^2 + c_n - n - q + I_0 + I_{stim}\right) & (v < 0) \\ \left(a_p(v-b_p)^2 + c_p - n - q + I_0 + I_{stim}\right) & (v \geq 0) \end{cases} \quad (1)$$

$$n(t+\Delta t) = n(t) + \Delta t \frac{1}{\tau} \times \begin{cases} \left(k_n(v-l_n)^2 + m_n - n\right) & (v < r) \\ \left(k_p(v-l_p)^2 + m_p - n\right) & (v \geq r) \end{cases} \quad (2)$$

$$q(t+\Delta t) = q(t) + \Delta t \frac{\varepsilon}{\tau}(v - v_0 - \alpha q) \quad (3)$$

The model of synapse is based on [9], and it is defined with the expressions, 4 and 5.

$$I_s(t+dt) = I_s + \Delta t \times \begin{cases} \alpha(1 - I_s(t)) & ([T] = 1) \\ -\beta I_s(t) & ([T] = 0) \end{cases} \quad (4)$$

$$I^i_{stim} = c \sum_{j=1}^{N} W_{ij} I^j_s \quad (5)$$

As for more details, please refer to [5].

2.1. Hebbian learning

The Hebbian learning is a learning method based on timings of spikes. When neuron i and j generate spikes in a similar timing, neuron j's influence on neuron i, W_{ij}, is strengthened. This is represented as follows.

$$\Delta W = A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) \quad (6)$$

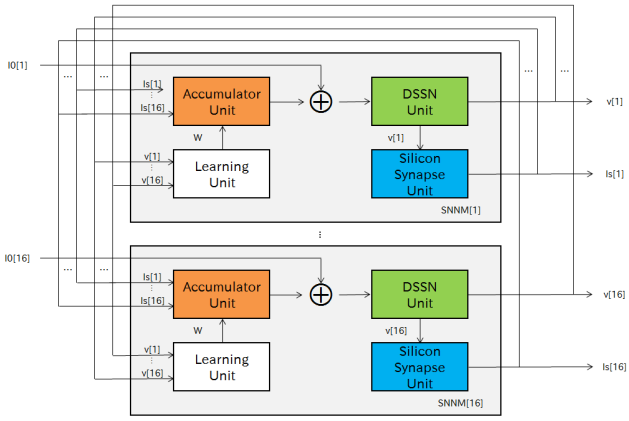

Figure 2: Circuit architecture of the previous work [7]

In the case that a spike generates an influence on both directions of increasing and decreasing values, the expression becomes as follows.

$$\Delta W = A_+ \exp\left(\frac{-|\Delta t|}{\tau_+}\right) - A_- \exp\left(\frac{-|\Delta t|}{\tau_-}\right) \quad (7)$$

where Δt is the time difference between the spikes and A_+, A_-, τ_+, and $\tau-$ are constants.

In the experiments, we use the above computation for the learning.

3. Previous work

DSSN model was first implemented on FPGA in [7]. It is based on the circuit architecture shown in Figure 2, and it runs in the pipelined way shown in Figure 3. The FPGA chip used is Virtex 6 XC6VSX315T. The parameter values are shown in Figure 2. Some scalar multiplications are replaced with additions and so the number of multiplications is less than what are shown in the expression 1–4.

In this implementation, according to [7] Accumulator Unit can generate one output every 4 cycles. As shown in Figure 3, DSSN Unit and Silicon Synapse Unit actually operate only 3% of the entire computation time. Nevertheless Accumulator Unit and other units must exist for parallel computations, as Accumulator Unit generates outputs only at specific small numbers of cycles.

4. Our implementation

In this section we introduce our implementation of the DSSN model and its networks. First we show the FPGA system and associated high level synthesis tool that we use for our implementation.

4.1. Target FPGA system

The FPGA system is connected to a host Xeon-based PC through PCI express as shown in Figure 4. The FPGA

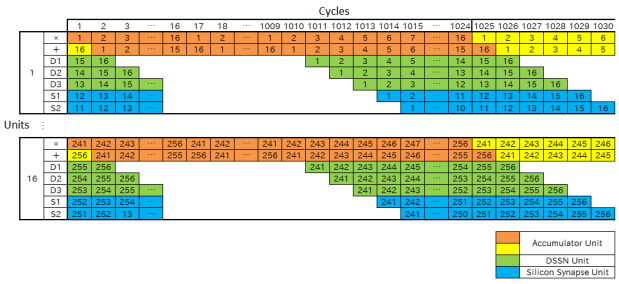

Figure 3: Pipeline operations of the previous work [7]

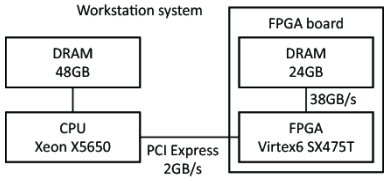

Figure 4: Target FPGA sytem

board has Virtex 6 XC6VSX475T and 24GB of DRAM memories. The host PC download the FPGA configurations and then host PC and FPGA can run at the same time with possible communication. In our implementation, after the DSSN models and networks are downloaded into the FPGA, spiking neural networks are simulated on the FPGA board only. After finishing all simulations, the results are transferred to the host PC.

We use a high level synthesis tool, Maxcompiler [8] from Maxeler. It can generate highly pipelined VHDL codes from data flow graphs represented in Java syntax. The VHDL codes are further compiled by Xilinx tools. Maxcompiler does not accept any conditional statements. The inputs to the compiler must be purely data flow graphs for highly pipelining. So, the conditional statements shown in the expressions (1–2) are manually converted into the ones which always compute both cases and selecting the right ones through multiplexers. Maxcompiler can accept the target clock speed. The numbers of pipeline stages can easily exceed 1,000 in typical compilations including in this implementation.

4.2. Overall architecture

We are following the previous implementation [7], but improved the performance by 8 times through more intensive pipelining as well as the use of larger FPGA chips. The overall data flow graph for the computation is shown in Figure 5. The inputs to the circuit are the initial values of I_s, W and several control signals, and the outputs are updated I_s.

Simulations are performed based on the expressions 1–5 for 64 steps with a fixed value of W, and then learnings are performed for 2 steps. This process is repeated until the value of W does not change for some time. When there are N neurons, 1 step is computed with $N + 12$ clock cycles.

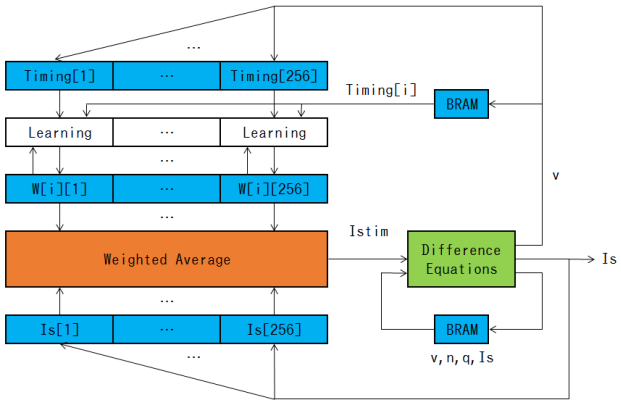

Figure 5: Overall data flow of the computation

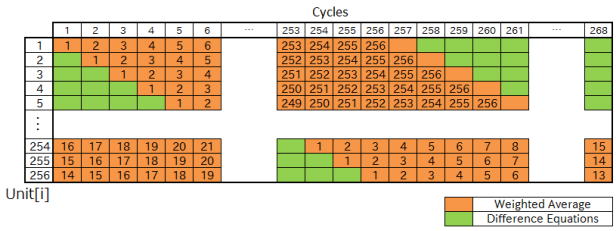

Figure 6: Operations in each cycle when N=256

Figure 6 shows the case of N=256.

The entire circuit is decomposed into the weighted sum unit, which computes the expression 5, the difference computation unit, which computes the expression 1–4, the learning unit, and the storage unit. These are different from Figure 2.

The weighted sum unit receives $2N$ of Wij as inputs, and so for speedy computations Wij are stored in the block RAM in the FPGA chip.

The difference computation unit operates on the variables, v, n, q, I_s, which are also stored in the block RAM. The learning unit and the storage unit are checking when a spike happens and use it for learning.

4.3. Fast computation of weighted sum

The computation for the expression 5 dominates the total computation time. In the previous implementation [7], Accumulator unit does this computation, but it can generate only one output at every 4 cycles, which is the speed bottleneck.

On the other hand, in our implementation, we introduce the pipeline operations shown in Figure 6 so that more pipeline stages can be introduced as the numbers of neurons to be simulated increases. The computation of $Istim^1$ takes $N + 1$ cycles, and then I_s^1 is computed in 11 cycles. This is pipelined with respect to i. The computations in terms of dataflow are aligned for easier layout inside the FPGA chip. The exception on this is the communication

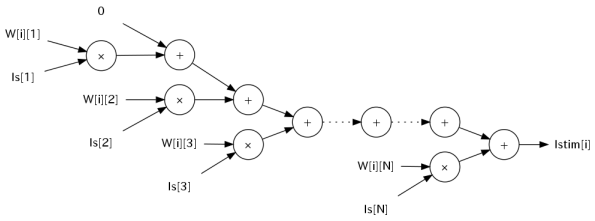

Figure 7: Structure of weighted sum unit

Table 1:

N	256	512	768	Total in chip
LUTs	19,515	58,298	109,443	297,600
FFs	39,934	101,215	168,136	297,600
DSPs	268	524	780	2,016
BRAMs	522	1,034	1,767	2,128

from the difference computation unit to the storage unit. Although there are a number of wires for this communication, right now there is no significant delay caused from them.

4.4. Bit width

Time interval, Δt, and bit-widths of variables are important parameters, as they influence the simulation time and accuracy. A software simulator has been implemented in C to determine those values. The results say Δt should be $3/8000$, and $v, n, q, Istim$ needs 18 bits and I_s, W needs 16 bits, which are used in our FPGA implementation.

4.5. Experimental results

The Java description based on the above dataflow has around 200 logical lines of codes. It has been synthesized to the FPGA netlists by Max Compiler [8], which runs at 200MHz. The synthesized results are shown in Table 1. There can be up to 768 neurons in one FPGA chip.

Compared with the existing implementation [7], the number of cycles is reduced to 1/4 and clock speed becomes double, and so in total around 8 times speed up has been observed. Our implementation is 280 times faster than the actual brain.

5. Concluding remarks

We have presented our single FPGA chip implementation of DSSN network models which runs at 200MHz. When there are N neurons and we prepare p computing units whose details are shown above, the computation complexity is proportional to N^2 and so the computation time becomes N^2/p. With the largest Virtex6 chip from Xilinx, we can implement 768 computing units in a single chip.

We are working on implementations with multiple FPGA chips by decomposing the dataflow graph. With 4 FPGA boards which are interconnected by a ring network, we should be able to simulate around 3,000 neurons at almost the same speed as the case of single FPGA chip.

Table 2: Parameter values

Parameter	Value	Parameter	Value
a_n	8.0	a_p	8.0
b_n	0.25	b_p	0.25
c_n	0.5	c_p	0.5
k_n	2.0	k_p	16.0
p_n	$-2^{-2} - 2^{-4}$	p_p	$2^{-5} - 2^{-2}$
q_n	-0.705795601	q_p	-0.6875
φ	1.0	τ	0.003
r	-0.205357142	I_0	-0.205
c	0.060546875	Δt	0.000375
A_+	2^{-6}	A_-	$2^{-7} + 2^{-8}$
τ_+	11.25	τ_-	22.5

References

[1] K. Cheung et al.: NeuroFlow: A General Purpose Spiking Neural Network Simulation Platform using Customizable Processors," Frontiers in Neuroscience, Vol. 9, Article 516, pp.1–15, Jan. 2016.

[2] S. W. Moore, P. J Fox, S. J. Marsh, A. T. Markettos, A. Mujumdar: Bluehive - a field-programable custom computing machine for extreme- scale real-time neural network simulation, *IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines*, Toronto, 2012.

[3] H. T Blair, J. Cong, D. Wu: FPGA simulation engine for customized construction of neural microcircuits, *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2013.

[4] D.B. Thomas, W. Luk: FPGA accelerated simulation of biologically plausible spiking neural networks, *17th IEEE Symposium on Field Programmable Custom Computing Machines*, 2009.

[5] T. Kohno and K. Aihara: Digital spiking silicon neuron: concept and behaviors in GJ-coupled network, *Proceedings of International Symposium on Artificial Life and Robotics*, Beppu, OS3-OS6, 2007.

[6] Jing Li, Yuichi Katori and Takashi Kohno: An FPGA-based silicon neuronal network with selectable excitability silicon neurons, *Frontiers in NEUROSCIENCE*, 2012, Volume 6, Article 183.

[7] Jing Li, Yuichi Katori and Takashi Kohno: Hebbian Learning in FPGA Silicon Neuronal Network, i*Proceedings of the 1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing*, 2013.

[8] MaxCompiler | Maxeler Technologies. `http://www.maxeler.com/products/software/maxcompiler/`.

[9] Destexhe, A., Mainen, Z. F., and Sejnowski, T. J.(1998). "Kinetic models of synaptic transmission," in Methods in Neuronal Modeling, eds C. Koch and I. Segev (Cambridge, FL:MIT Press),1-25.