An FPGA Implementation of On-Chip Trainable Multilayer SAM Spiking Neural Network

Minoru Motoki^{a,*}, Ryuji Waseda^b, Terumitsu Nishimuta^b

^aNational Institute of Technology, Kumamoto College, 2659-2 Suya, Koshi-city, Kumamoto 861-1102, Japan

^bMaviss Design Co.,Ltd, 5-1-1 Minamikumamoto, Cyuou-ku, Kumamoto-city, Kumamoto 860-0812, Japan

*motoki@kumamoto-nct.ac.jp

Abstract

This paper describes an implementation of a multilayer SAM spiking neural network into the PL logic part (FPGA) in a Xilinx Zynq processor. The SAM neuron model is a type of one of the most popular LIF spiking neuron model. The SAM neural network can be an on-chip trainable model because the model does not require any multiplier under our proposed Back Propagation (BP) base training algorithm. As a result, the model achieved an XOR logic element in a unit of spikes. Moreover, we achieved a multiplier-less implementation with our intended algorithm and architecture. The design allows an arbitrary number setting for the hidden and output neurons.

Keywords: Spiking Neural Network, SAM model, supervised training, multiplier-less, FPGA, Zynq processor.

1. Introduction

Recently, researches and industrial applying of AI are very progressing, especially the neural network research area of AI is generating a wide variety of results⁽¹⁾. Researches are reducing the precision of calculation information of neuron outputs and neuron parameters for reducing the scales of the circuit and improving the performances. Therefore, researches of spiking neural networks (SNN) is substantially expected because the information of neuron output of the SNN is regarded as only 1 bit. Moreover, SNN is a nearer model to a biological neuron than normal artificial neural networks (ANNs)^(2,3,4).

We have been proposed a supervised training algorithm for the multilayer SAM spiking neural network⁽⁵⁾. The algorithm is based on BP. Because of allowing ‘multiplier-less’ implementation to digital circuits such as FPGAs by an approximation without using $\mathbf{R} \times \mathbf{R}$, the model can achieve

‘on-chip training’ on digital circuits.

The reports of SAM-SNN have been only simulation results of AND logic and 3rd poly-nominal function approximation for Intel-Altera FPGAs so far^(6,7). This time, we implemented XOR logic in a unit of spike timing using the SAM-SNN into the PL logic part (FPGA) in a Xilinx Zynq processor. Moreover, we designed the SAM-SNN as allowing arbitrary number setting for the hidden neurons and output neurons.

2. SAM spiking neural network

2.1 SAM neuron model

The Spike Accumulation and Modulation (SAM) neuron model is a type of LIF spiking neuron model. The behavior of the SAM neuron model is as follows. At a discrete time t , if the j -th neuron receives a spike $X_i(t) \in \mathbf{B} = \{0,1\}$ from the i -th neuron, then the inner potential $U_j(t) \in \mathbf{R}$ of the j -th neuron is calculated by the sum of product between the link weights $W_{ji} \in \mathbf{R}$ and input spikes $X_i(t)$, plus the addition of the weighted previous inner potential $aV_j(t-1)$ (where a is a decay parameter). Thus,

$$U_j(t) = \sum_{i=1}^{n_1} W_{ji} X_i(t) + aV_j(t-1) \quad (1)$$

The j -th neuron output $X_j(t) \in \mathbf{B}$ is obtained by applying the activation function $g()$ to $U_j(t)$, where $g()$ is a step function such that if $U_j(t)$ is less than θ the output is 0, corresponding to no spike; if $U_j(t)$ is greater than or equal to θ the output is 1 and a spike is generated. θ is the threshold, i.e.,

$$X_j(t) = g(U_j(t) - \theta), \quad (2)$$

$$u = U_j(t) - \theta, \quad (3)$$

Fig. 1. Multilayer SAM Spiking Neural Network.

Fig. 2. Derivative function $g'(u)$.

$$g(u) = \begin{cases} 0 & (u < 0) \\ 1 & (u \geq 0) \end{cases} \quad (4)$$

Moreover, the inner potential is decreased by an amount p upon activation, such that

$$V_j(t) = U_j(t) - pX_j(t). \quad (5)$$

The model has the advantage of treating time information within a single neuron body, avoiding a reduction in the frequency of output spikes even if the sampling interval is wide, comparing to the LIF neuron model. Therefore, the SAM neuron model is suitable for implementation on digital circuits (FPGAs).

2.2 Supervised training algorithm of Multilayer SAM neural network

The training approach for the multilayer SAM neural network is based on an approximation of BP. We define the objective function E as RMS error loss between output spike

$X_{p,k}^{(3)}(t)$ and teacher spike $T_{p,k}^{(3)}(t)$,

$$E \stackrel{\text{def}}{=} \frac{1}{N} \sum_{P=1}^N \sum_{t=1}^{t_N} \frac{1}{2} \sum_{k=1}^{n_3} \left(X_{P,k}^{(3)}(t) - T_{P,k}^{(3)}(t) \right)^2. \quad (6)$$

Generally, for the l -th layer, the link weight $W_{nm}^{(l)}$ is updated by gradient, such as

$$W_{nm}^{(l)}(t+1) \leftarrow W_{nm}^{(l)}(t) - \eta \frac{\partial E(t)}{\partial W_{nm}^{(l)}(t)} \quad (7)$$

where,

Fig. 3. A schematic design of the SAM neuron model (2inputs, 1output). The link weights $W1, W2$ are updated by the values with regard to the difference between the output spike X_j and the teacher spike T_j , and the design is multiplier-less.

$$\frac{\partial E(t)}{\partial W_{nm}^{(l)}(t)} = \left(X_n^{(l)}(t) - T_n^{(l)}(t) \right) g' \left(u_n^{(l)}(t) \right) H_{nm}^{(l)}(t), \quad (8)$$

$$H_{nm}^{(l)}(t) = aH_{nm}^{(l)}(t-1) + X_m^{(l-1)}(t). \quad (9)$$

Here, when the layer l is a hidden layer, $T_n^{(l)}(t)$ is the teacher signal for the hidden layer calculated by the notion of approximation of BP as

$$T_n^{(l)}(t) = \begin{cases} 1 & \left(X_n^{(l)}(t) - \frac{\partial E(t)}{\partial X_n^{(l)}(t)} \right) \geq 0.5 \\ 0 & \left(X_n^{(l)}(t) - \frac{\partial E(t)}{\partial X_n^{(l)}(t)} \right) < 0.5. \end{cases} \quad (10)$$

Because the differential of the step function $g'(u)$ is a delta function that has infinite value, we instead use the pseudo derivative function shown in figure 2.

3. Design and implementation of the SAM neural network for Zynq processor

3.1 Design of the SAM model and the SAM neural network

The schematic design for digital circuitry based on the training algorithm described in chapter 2 is shown in figure 3. Figure 3 shows the SAM neuron model circuit is achieved without any multiplier (namely ‘multiplier-less’). The multiplication of decay constant a is realized by shift

Fig. 4. A schematic design of the Multilayer SAM neural network. The network consists of some SAM neuron models.

(trapezoid symbol).

Figure 4 shows the multilayer SAM neural network design, which has switches of inferring mode and training mode. “Learning” module produces the teacher spike signal in each layer. The HDL design is parameterized for generating circuits by specifying hyper parameter N_2 (the number of hidden units) automatically. We used Verilog-HDL for the design.

This time, we chose the XOR problem as a task. As the number expression, we adopted fixed point number of 17 bits (m.n format). We set $m=5$ and $n=12$.

As the parameters for the SAM neuron model, we set $a = 0.5$, $p = 0.5$. The initial value of link weights $W_{nm}^{(l)}$ is set on random, and $\theta_n^{(l)}(t) = 0.25$. The learning coefficient $\eta = 0.0625$.

The state design is as follows. There are 3 states, such as initial state(S0), inferring and training states(G0L0), and next time step(TNEXT). After the start, the controller repeats G0L0 and TNEXT alternately.

Moreover, this time, we introduced a parameterized designed to set an arbitrary number of hidden neuron units and output neuron units by using generate sentence of Verilog-HDL.

3.2 Simulation and Implementation results

The tool for designing, implementation and simulation is Xilinx Vivado Suite 2018.3⁽⁸⁾, and the assumed FPGA is a Zynq processor (xc7z020clg484-1) which has 17600 LUTs and 35200 FFs.

The result of the 2-2-1 (2 inputs, 2 hidden neurons, 1 output neuron) SAM neural network for the XOR task is shown in figure 5. Figure 5 shows 6 blocks that are the spike result of input(X1, X2)/output(Xj)/teacher signal(Tj) of the net, i/o/t of the output layer, i/o/t of the hidden neuron 1, and the hidden neuron 2, from top to bottom. At 459 training epoch (signal l[9:0]) the XOR task is realized. Moreover, the figure shows that the hidden neuron 1 performs OR logic element and the hidden neuron 2 performs AND logic element.

Figure 6 shows an implementation report of the design. The model required small circuit scale such that 72 LUTs (0.41% of 17600LUTs) and 57 FFs(0.16% of 35200FFs) total for the implementation. In addition, there was no DSP48E report, that is, a multiplier-less design was achieved.

Fig. 5. A design and simulation result of XOR problem into the PL block (FPGA) of Zynq processor. The SAM-SNN output X_j performs as the same spikes as T_j in the ‘Outlayer’ at 459 training epoch on 13rd row.

4. Conclusions

This time, we implemented the 3 layers SAM spiking neural network into the Xilinx Zynq processor. As concluding remarks,

- In a simulation, the model performs XOR logic element in a unit of spike timing.
- The implemented circuit scale is small, and it is confirmed that the implementation does not use any multiplier. Thus, the design allows ‘on-chip training’ circuitry.
- The design is parameterized in the setting of hidden layer neuron units and output layer neuron units. This allows a setting arbitrary number of neuron units.

Recently, an embedded trainable AI board Jetson Nano using GPU has been on sale by NVIDIA Co., Ltd⁽⁸⁾. However, generally speaking, GPU requires somewhat electrical energy (Jetson Nano needs 5 to 10 watt). This research aims to develop trainable AI devices that have characteristics of energy efficiency. It is considered that the applications of this research are autonomous robots, intelligent sensors, and

Resource	Utilization	Available	Utilization %
LUT	72	17600	0.41
FF	57	35200	0.16
IO	6	100	6.00

Fig. 6. Utilization of the implementation for Zynq processor (xc7z020clg484-1). The report result of ‘DSP48E’ for multiplier usage does not report.

myoelectric prosthetic hands.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP19K12176.

References

- (1) Krizhevsky, A., Sutskever, I., Hinton, G.E.: “ImageNet Classification with Deep Convolutional Neural Networks”, Advances In Neural Information Processing

Systems , 1–9arXiv:1102.0183, 2012

- (2) Catherine D. Schuman et.al.: “A Survey of Neuromorphic Computing and Neural Networks in Hardware.”, arXiv:1705.06963, 2017
- (3) Merolla, P. et.al.:”A million spiking-neuron integrated circuit with a scalable communication network and interface (IBM TureNorth)”, Supplementary Materials on Science online 345, 668, 2014
- (4) Kulkarni, S.R., Rajendran, B. :”Spiking neural networks for handwritten digit recognition - supervised learning and network optimization”. Neural Networks 103, pp. 118–127, 2018
- (5) M.Motoki et.al.: “Function Approximation Using Multilayer SAM Spiking Neural Network”, Proc. of IEEE 8th International Innovative Computing Technology (INTECH2018), pp.65-70, 2018
- (6) M.Motoki, M., Matsuo, K.: “An error-correction learning algorithm for SAM neuron model and its HDL design” (in Japanese), IEICE Workshop on Circuits and Systems 27, pp.244–249, 2014
- (7) Y.Aso et.al.:”A study on fixed point number representation on FPGA implementation of SAM spiking neural network” (in Japanese), IEICE Workshop on Circuits and Systems 31, pp.276–279, 2017
- (8) Xilinx Vivado Design Suite HLx Editions design tools, <https://www.xilinx.com/products/design-tools/vivado.html> (refer Nov 2020)
- (9) NVIDIA Jetson Nano Developer Kit, tutorial, <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>, (refer June 2020)