Implementation of Fixed-point Neuron Models with Threshold, Ramp and Sigmoid Activation Functions

Lei Zhang
School of Engineering and Applied Science, University of Regina, Regina, Saskatchewan, Canada, S4S0A2
lei.zhang@uregina.ca

Abstract. This paper presents the hardware implementation of single-neuron models with three types of activation functions using fixed-point data format on Field Programmable Gate Arrays (FPGA). Activation function defines the transfer behavior of a neuron model and consequently the Artificial Neural Network (ANN) constructed using it. This paper compared single neuron models designed with bipolar ramp, threshold and sigmoid activation functions. It is also demonstrated that the FPGA hardware implementation performance can be significantly improved by using 16-bit fixed-point data format instead of 32-bit floating-point data format for the neuron model with sigmoid activation function.

1. Introduction
Although the calculation speed of computer hardware has been growing exponentially for decades, human brain is still far more efficient than any sequential computer in terms of power consumption per calculation. A human brain has 100 billions of neurons and over 100 trillions of connection, forming the largest and most complex parallel architecture, even a tenth of which is hard to simulate. Recent research shows the brain simulation of 20,000 neurons and 51 billions synapses on an ARM processor at speeds close to biological real-time[1]. This is an infinitesimal fraction of the human brain. Parallel computing devices such as FPGA can better represent the parallel architecture of the neural network and provide hardware acceleration in order to simulate a larger brain neural network. There has been a number of hardware-based neuron models published in recent research. An FPGA implementation of a neuron model with 64-bit double precision floating point data format is presented in [2], and a second-order piecewise nonlinear approximation function was used to simplify the calculation. An FPGA implementation of a single artificial neuron using various activation functions and XOR Gate is presented in [3]. A high accuracy adaptive exponential integrated neuron model is presented in [4]. An efficient neural architecture for FPGA-based spiking neural networks is presented in [5]. An FPGA implementation of Hodgkin-Huxley Neuron model is presented in [6]. Sigmoid function is the most important and commonly used activation functions in an artificial neural network (ANN) architecture [7]. There are several approximation methods used for simplify the implementation of sigmoid function, such as piece-wise linear (PWL), look-up table and CORDIC. The FPGA implementation of bipolar sigmoid activation function is presented in [8]. The previous research commonly represents neuron models with floating-point data format to achieve high precision, which is however inefficient in FPGA hardware implementation. In this paper, 16-bit fixed-point data format is used for the design of a single neuron model with three different types of activation functions: threshold, ramp and sigmoid.
2. General Neuron Model for ANN

An ANN is a network of interconnected neurons arranged in multiple layers, including one input layer, one output layer and a number of hidden layers. A neuron is the basic element in the construction of an ANN. The model of a single neuron is illustrated in Fig. 1. A neuron has multiple synaptic inputs with individual weight associated with each input. A simple 3-layer ANN architecture is illustrated by Fig. 2, with 2 inputs, 4 neurons in the hidden layer and 1 output. A general mathematic representation of an individual neuron within an ANN architecture is shown by equation (1).

\[ a_j^l = \sum_{i=1}^{N_l} w_{ij} x_i^l + b_{j,0}^l, \quad j = 1, 2, \ldots, N_{l+1}, \quad y_j^l = f_i(a_j^l) \]  

where \( N_l \) is the number of neurons at \( l \)-layer. \( j \) is the index of the neurons at \( l \)-layer. Each hidden neuron \( j \) receives the output of each input neuron \( i \) from the input layer multiplied with a weight of \( w_{ij} \). The sum of all weighted inputs is used by an activation function \( f_i \) to produce the output of the hidden layer neuron and feed it forward to the output layer. A similar weighted sum is generated for each output neuron. \( b_{j,0}^l \) is the bias of the \( j \)th neuron at the \( l \)th layer, which are added as noise to randomize the initial condition in order to get better chance to converge when training ANN with back propagation algorithm. FPGA is particularly suitable for hardware implementation of the successive multiply-accumulation (MAC) operation of multiple neurons in an ANN.

3. Activation Functions

The accumulated synaptic inputs of a neuron, represented by the summed weights, is transferred to output via an activation function. Common activation functions include threshold, ramp and sigmoid function. These functions can be used to represent either unipolar or bipolar neuron model. Unipolar model has a single polarity, normally positive, while bipolar model has two polarities, both positive and negative. The output of the unipolar activation function ranges from 0 to +1 and is symmetrical to the (0,0.5) point. The corresponding bipolar activation functions are illustrated in Fig. 3. The output of the bipolar activation function ranges from -1 to +1 and is symmetrical to the origin.

3.1. Threshold Activation Function

The ramp function is a piece-wise linear function saturated at upper and lower limits. The unipolar and bipolar ramp activation functions are shown in equation (2). The ramp function is used to design ANN for fitting applications. The summed weights in the ANN architecture need to be normalized within the linear range of the activation function in order to represent linear patterns.

\[ f_u(x) = \begin{cases} 1.0, & x \geq x_{ih} \\ 0.0, & x < x_{ih} \end{cases}, \quad f_b(x) = \begin{cases} 1.0, & x \geq x_{ih} \\ -1.0, & x < x_{ih} \end{cases} \]  

Figure 1. Single Neuron  
Figure 2. A 3-layer ANN  
Figure 3. Bipolar Activation
3.2. Ramp Activation Function
The ramp function is a piece-wise linear function saturated at upper and lower limits. The unipolar and bipolar ramp activation functions are shown in equation (3). The ramp function is used to design ANN for fitting applications. The summed weights in the ANN architecture need to be normalized within the linear range of the activation function in order to represent linear patterns.

\[
\begin{align*}
  f_u(x) &= \begin{cases} 
  1, & x > 1 \\
  x, & 0 \leq x \leq 1 \\
  0, & x < 0
  \end{cases} \\
  f_b(x) &= \begin{cases} 
  1, & x > 1 \\
  x, & -1 \leq x \leq 1 \\
  -1, & x < -1
  \end{cases}
\end{align*}
\]  

(3)

3.3. Sigmoid Activation Function
The unipolar and bipolar sigmoid activation functions are represented by equation (4). The bipolar sigmoid function is also called a hyperbolic tangent sigmoid function or a logistic sigmoid function.

\[
f_u(x) = \frac{1}{1 + e^{-\beta x}}; \quad f_b(x) = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}} = \frac{2}{1 + e^{-\beta x}} - 1
\]  

(4)

where \( \beta \) represents the slope of the sigmoid function. The outputs of sigmoid functions (unipolar and bipolar) approach to the threshold function when the value of \( \beta \) increases to the positive infinity \((\beta \to +\infty)\). When \( \beta = 0 \), the outputs is a horizontal line (x axis). Sigmoid function is a continuous non-linear function, which maps an infinite input domain \((-\infty, \infty)\) to a finite output domain: (0,1) for unipolar and (-1,1) for bipolar. This property of sigmoid function provides generalization in ANN models to approximate any function. Sigmoid function is biologically plausible and commonly used for ANN design with back propagation training. Back propagation algorithm is based on gradient decent which requires calculating the differential output of the activation function. The sigmoid function is easy to differentiate and its differential function is also continuous. The derivative of sigmoid function is shown in equation (5), which can be simply implemented on an FPGA using a multiplier and a subtracter (2's complement adder).

\[
f'(x) = \frac{df(x)}{dx} = \left( \frac{1}{1 + e^{-x}} \right)' = f(x)(1 - f(x))
\]  

(5)

4. Hardware Models and Implementation
A model-based hardware design approach is employed for the neuron model FPGA implementation. A hardware model is created using Simulink and Xilinx System Generator for threshold, ramp and sigmoid activation function respectively. These models are firstly simulated in Matlab/Simulink software environment. The neuron models are designed with two inputs, which can be easily extended by adding inputs and weights. The multiplier block `Mult' is used for calculating the product of input and weight. The adder block `Add' is used to calculate the summed weights. The input \( x_{in} \) is a series of linear increment values range between (-2, 2) for evaluating the outputs of the activation function. The 16-bit fixed-point bipolar threshold, ramp and sigmoid models are shown in Figure 4(a), (b) and (c) respectively. After the design function is verified by the behavior simulation, a design file (netlist) is then exported to generate a VHDL-based Vivado project for FPGA implementation. The neuron models are implemented on a Zeboard with a Zed7020 FPGA device. Xilinx Vivado 2015.4 was used to implement the model-based designs. The FPGA implementation results are shown in Table.1.
Figure 4. 16-bit Fixed-point System Generator Neuron Models

### Table 1. FPGA Implementation Results.

| Activation Function | Little Format | 16-bit Fixed-point | 16-bit Fixed-point | 32-bit Floating-point | 16-bit Fixed-point |
|---------------------|---------------|---------------------|---------------------|----------------------|---------------------|
|                     | Option        | Syn                 | Mux                 | Max(DSP)             | Multi(logic)        | Multi(DSP)          | D-RAM               | BRAM                |
| Worst Negative Slack (ns) | (-0.269) | 0.034 | 1.195 | (-0.388) | 1.28 | -44.537 | -43.312 | 0.703 | 1.165 |
| Maximum Frequency (MHz) | (97.38) | 100.34 | 113.57 | (96.26) | 114.68 | (18.34) | (18.76) | 107.56 | 113.19 |
| Look-up Table (LUT) | 590 | 590 | 24 | 598 | 32 | 3133 | 2018 | 272 | 97 |
| Register’s Flip-flop (FF) | 98 | 100 | 100 | 112 | 112 | 356 | 356 | 147 | 155 |
| Slices | 192 | 193 | 37 | 196 | 40 | 1001 | 712 | 112 | 52 |
| DSP Elements | 0 | 0 | 2 | 0 | 2 | 15 | 21 | 2 | 2 |
| Block RAM | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0.5 |
| Total On-chip Power (W) | 0.132 | 0.132 | 0.126 | 0.148 | 0.141 | 0.232 | 0.218 | 0.146 | 0.137 |
5. Conclusion and Future Research

In conclusion, this paper presented the 16-bit fixed-point FPGA hardware implementation of a single neuron model with three different activation functions: threshold, ramp and sigmoid function. Xilinx System Generator is used to create hardware-based models for the hardware implementation. The performance of the sigmoid neuron model is significantly improved by using 16-bit fixed-point data format and memory based design compared to 32-bit floating point model. The neuron models are further optimized by using FPGA dedicated hardware resource DSP48E1 and block RAM. The maximum operating frequency is increased by more than six times from 18 MHz to 113 MHz. The FPGA logic resource utilization is reduced by 93% from 712 to 52. The optimized performance achieved by the 16-bit fixed-point models also reflects the characteristics of biological neurons, which need to maintain simplification in order to deliver efficiency. The simplified neuron model will increase the processing speed for an ANN. The bipolar activation function models can be easily modified for unipolar activation functions. These models will be use in future research for the design and FPGA hardware implementation of ANN topologies in brain simulation and machine learning applications. The fixed-point sigmoid hardware implementation will also be used for the design of large ANN topology.

References

[1] J. C. Knight, P. J. Tully, B. A. Kaplan, A. Lansner, and S. B. Furber, “Large-scale simulations of plastic neural networks on neuromorphic hardware,” Frontiers in Neuroanatomy, vol. 10, p. 37, 2016.
[2] S. Kawamura, M. Saito, and H. Yoshida, FPGA Implementation of Neuron Model Using Piecewise Nonlinear Function on Double-Precision Floating-Point Format. Springer International Publishing, 2016, pp. 620–629.
[3] S. Abrol and R. Mahajan, “Implementation of single artificial neuron using various activation functions and xor gate on fpga chip,” in 2015 Second International Conference on Advances in Computing and Communication Engineering, May 2015, pp. 118–123.
[4] A. Makhlooghpour, H. Soleimani, A. Ahmadi, M. Zwolinski, and M. Saif, “High accuracy implementation of adaptive exponential integrated and fire neuron model,” in 2016 International Joint Conference on Neural Networks (IJCNN), July 2016, pp. 192–197.
[5] L. Wan, Y. Luo, S. Song, J. Harkin, and J. Liu, “Efficient neuron architecture for fpga-based spiking neural networks,” in 2016 27th Irish Signals and Systems Conference (ISSC), June 2016, pp. 1–6.
[6] M. Lu, J.-L. Wang, J. Wen, and X.-W. Dong, “Implementation of hodgkin-huxley neuron model in fpgas,” in 2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEC), vol. 01, May 2016, pp. 1115–1117.
[7] A. N. Ide and J. H. Saito, FPGA Implementations of Neural Network. Springer, 2006, ch. 7. FPGA Implementations of Neocognitrons, pp. 197–224.
[8] I. Sahin and I. Koyuncu, “Design and implementation of neural networks neurons with radbas, logsig, and tansig activation functions on fpga,” Elektronika ir elektrotechnika, vol. 120, no. 4, pp. 51–54, Apr. 2012.