A Modularization Hardware Implementation Approach for Artificial Neural Network

Tong WANG[1, a], Lianming WANG[*2, b]

[1]School of Physics, Northeast Normal University, Changchun, 130024, China

[2]School of Physics, Northeast Normal University, Changchun, 130024, China

[a]email: wangt443@nenu.edu.cn, [b]email:wanglm703@nenu.edu.cn(Correspondence)

Keywords: Artificial Neural Network; Modularization; Digitization; FPGA

Abstract. Hardware implementation has been proven to be an effective way to take full advantage of the parallel and distributed computation ability of artificial neural network. To simplify the hardware implementation process of different kinds of neural networks, a modularization and digitization implementation method based on FPGA is proposed. Firstly, some commonly used artificial neural network structures are divided into several functional modules, which are then digitized with HDL. Finally, the hardware implementation of an expected neural network can be achieved by combining those related modules with ease in FPGA. The modularization construction and hardware implementation process of a discrete Hopfield neural network is taken as an example to validate the feasibility and effectiveness of the method.

1 Introduction

Artificial neural networks (ANN) with parallel process, distributed storage and self-learning characteristics are widely used in the information processing field. Currently, there are two ways to implement the functions of neural networks; one is by performing algorithms with computer software simulation, while the other is by implementing structure and algorithms with hardware. Since the software simulation implementation method is characterized by low speed and low parallelism, the performance of a neural network cannot be fully exerted. Therefore, the hardware implementation methods based on electronic techniques have attracted more and more attention in recent years.

Performing neural network algorithms with general purpose processor units [1] [2] has been a general method for hardware implementation of neural network. Nevertheless, it is essentially a hybrid method. Complete hardware implementation is based on very large scale integrated circuit (VLSI) [3] [4]; or based on Field Programmable Gate Array (FPGA) [5] [6] [7] [8]. To a certain degree, although these methods above can take full advantage of the performance of neural networks, most achievements are dedicated to specific neural network models or stationary network structures. Therefore, there is lack of universal and flexible hardware implementation method in practical applications.

To simplify the hardware implementation process of different kinds of neural networks, a modularization and digitization implementation method based on FPGA is proposed, in which some commonly used artificial neural network structures are firstly divided into several functional modules; then the functional modules are digitized with Hardware Description Language (HDL); finally, an expected neural network can be implemented by combining those related modules with ease in FPGA.

2 Module Division

Considering commonly used artificial neural network structures, we extracted 5 functional modules as basic modules to construct various artificial neural network models, as shown in Figure 1, including input processing module (IPM), network connection and synaptic weight memory module (NWM), neuron module (NM), learning module (LM) and output processing module

(OPM).

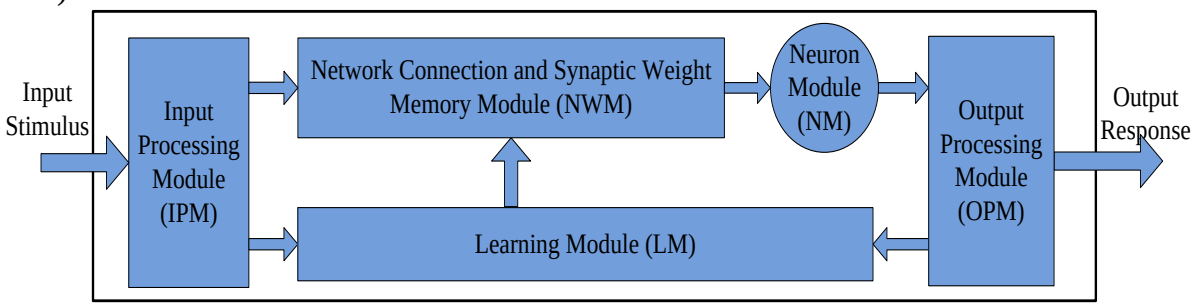

Fig.1. ANN functional module division

Functions of each module are shown as follows:

(1) Input processing module (IPM): It is designed for translating input signals or input stimulus representing the external information into the signals processed by the neural network.

(2) Network connection and synaptic weight memory module (NWM): It shows the connections among neurons inside the neural network and stores the corresponding connection weights (synapse weight), thresholds and state values. Different neural network structures can be constituted by altering the connections between the neurons, such as the monolayer (multilayer) feedforward neural network, the feedback neural network, the fully connected neural network, the partially connected neural network, the local connection neural network and so on.

(3) Neuron module (NM): It is the most basic information processing unit in the neural network.

(4) Learning module (LM): It controls the neural network training or learning process, which regulates the synaptic weight and determines the neuronal threshold.

(5) Output processing module (OPM): It estimates whether the neural network has reached a stable status. If the neural network reaches the condition, then OPM will convert the signals processed by the neural network into the corresponding output signals, or else it will return a signal to NM for another calculation.

3 Module Implementation

To realize each module in FPGA and make the connection between modules easily, VHDL (Very High Speed Integrated Circuit Hardware Description Language) is used to describe each module. Since VHDL code can be synthesized to fit any FPGA, this digitization method tends to form a systematic and generally accepted method for constituting an expected neural network in FPGA easily and quickly with those modules.

3.1 NWM Digitization

When the neural networks size is large and the connections among neurons are complicated, especially when the network structure (the number of neurons and the connections between neurons) needs to be variable, a flexible network connection and synaptic weight memory structure is necessary. In this case, a linked list structure with two memory, initial memory and topology structure memory is adopted to express connection topology and store thresholds, state values and weights, as shown in Figure 2.

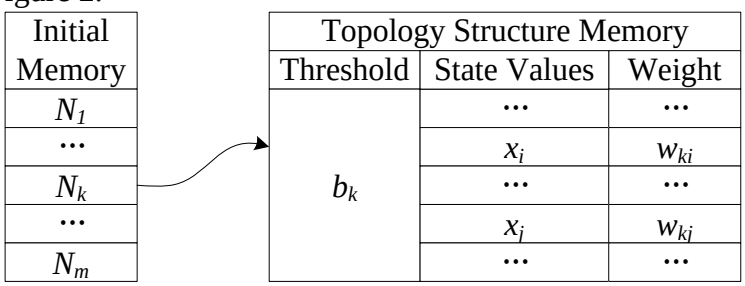

Fig.2. NWM structure

The figure shows the connections among m neurons, N1 to Nm. The signs of N1 to Nm are stored in the initial memory area and at the same time each sign points to a topology structure

memory area where the threshold, state values and weights related to this neuron are stored in consecutive memory blocks.

3.2 Neuron Module (NM)

An artificial neuron model [9] can be expressed as equation (1):

$$y_i = f\left(\sum_{j=1}^{N} w_{ij} x_j + b_i\right) \tag{1}$$

Where x_j $(j=1,2,...,N)$ is the input of neuron i, w_{ij} is the connection strength or synaptic weight between neuron i and j, b_i is the threshold of neuron i, $f(\cdot)$ is the excitation function and y_i is the output of neuron i. The neuron model can be divided into two functional sub-modules according to the arithmetic expression: multiplier/accumulator sub-module $\left(\sum_{j=1}^{N} w_{ij} x_j + b_i\right)$ and excitation function sub-module $f(\cdot)$.

Since multiplier/accumulator sub-module needs a large number of multiplication and addition operations and may occupy vast FPGA resources, a parameterized multiplier/accumulator structure is designed, by which the connection between the multipliers and the accumulators can be configured to be completely parallel, completely serial or partially parallel and partially serial to adapt FPGAs with different quantities of resources.

Provided neuron i is connected with other N neurons, to achieve the fastest processing speed, a completely parallel multiplier/accumulator sub-module with $2N+1$ data input ports (1 threshold, N state values and corresponding N weights) can be formed as Figure 3 by setting corresponding parameters. A total of N multipliers, $N/2$ adders, a data selector and an accumulator are used. The data selector is to select the output data of the adders to pass to the accumulator, by which the data are accumulated.

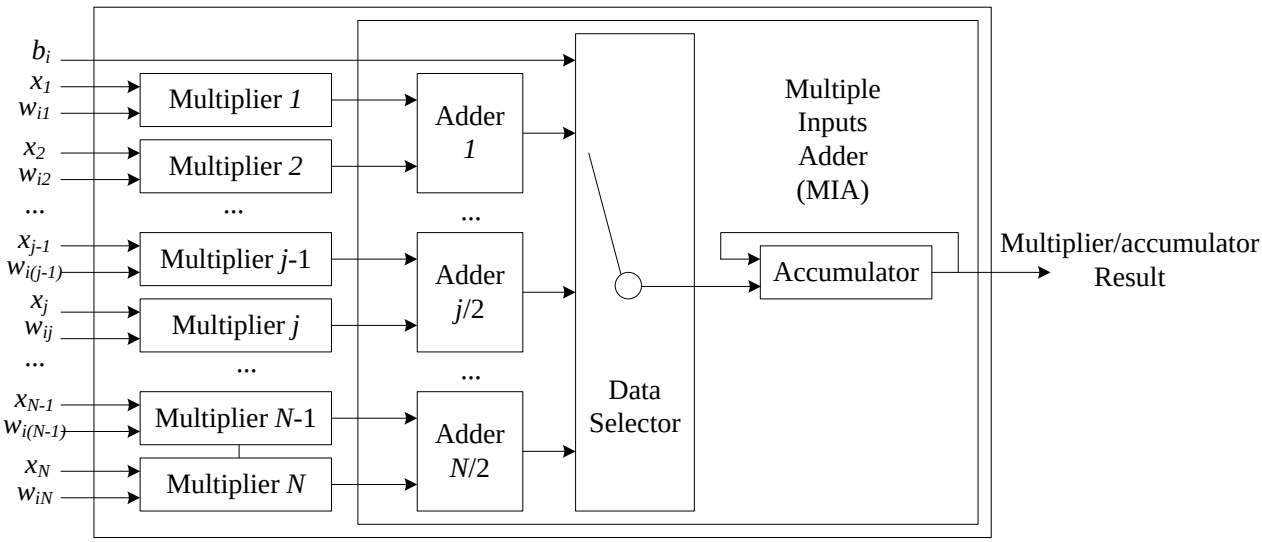

Fig.3. Completely parallel multiplier/accumulator sub-module structure

Although the completely parallel structure has the highest processing speed, it occupies most FPGA logic resources. Due to this reason, a partially parallel and partially serial structure may be a better tradeoff between the speed and resource, as shown in Figure 4, where the multiplexer is to divide the input into N/M groups and then successively transfer the data into the multiplier/accumulator sub-module, which consists of M Multipliers, a multiple inputs adder (MIA), a data selector and an accumulator.

By changing the parameter M, a completely serial structure may be acquired to occupy the least logic resources of FPGA, while the processing speed may be the lowest.

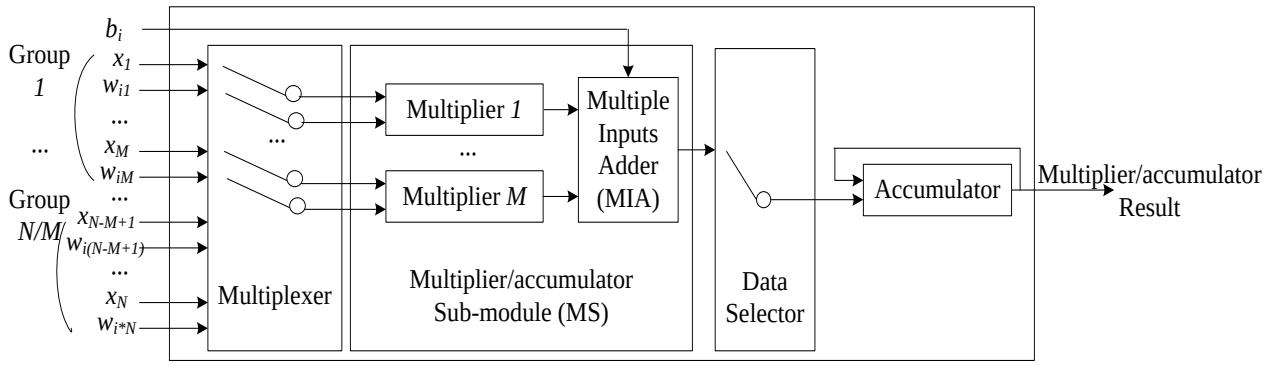

Fig.4. Partially parallel and partially serial multiply-accumulate structure

Since most calculations performed in neural network model are based on continuous value, the above-mentioned multiplication and addition operations are inevitably applied to floating-point numbers. To meet the demands of floating point number accuracy for different neural networks, a so-called custom floating point number format is adopted to express the floating point number by two's complement, as shown in equation (2).

$$N = F \times 2^{E} \tag{2}$$

Wherein F is a decimal, E is the index and 2 is the base number. The decimal part includes the sign bit and the practical decimal bits. F is defined as a decimal less than 1 and E is an integer. It is obvious that the given binary digits for F and E determine the accuracy and the range of the floating point number N respectively. During the calculation process, all numbers should be normalized according to the format.

For excitation function sub-module, the frequently-used excitation functions, such as threshold function, piecewise linear function and Sigmoid function are implemented by unified lookup table (LUT) method with high accuracy.

3.3 Learning Module (LM)

Generally, a learning process for a neural network is to determine the weights and thresholds of the network according to some learning rules, such as Hebb learning rule, error correction learning rule, memory-based learning rule, random learning rule and competitive learning rule [9]. Since most learning rules are too complicated to be implemented in FPGA, and different neural networks adopt different learning rules, off-line learning based on MATLAB is applied to obtain weights and thresholds firstly and then store them into the hardware topology structure memory.

4 System Test by Discrete Hopfield Neural Network

To test and verify the functions of the modularization hardware implementation system for neural network, a discrete Hopfield neural network containing 64 neurons is constructed in a ALTERA EP4CE115F29C7 FPGA to perform the recognition of the digits 0-9 with the size 8pixel$\times$8pixel. The constructed hardware structure is shown in Figure 5, where the system control module is designed to coordinate the time sequence of other modules.

The frequency of the system clock is 20MHz which is inputted into the system from the port *clk*. The port *start* is to start the system by applying a falling edge. The associated results are outputted from the port *associate_data*[63..0]. For testing the correctness of the associated result, the port *standard_data*[63..0] and *noisy_data*[63..0] are used to output standard sample data and noise-polluted sample data respectively. The port *over* is to indicate the end of association process.

Ultimately, the test results show that the system can complete one-time digit recognition in about 2ms. And what is more important, it validates the feasibility and effectiveness of the modularization hardware implementation approach for artificial neural network.

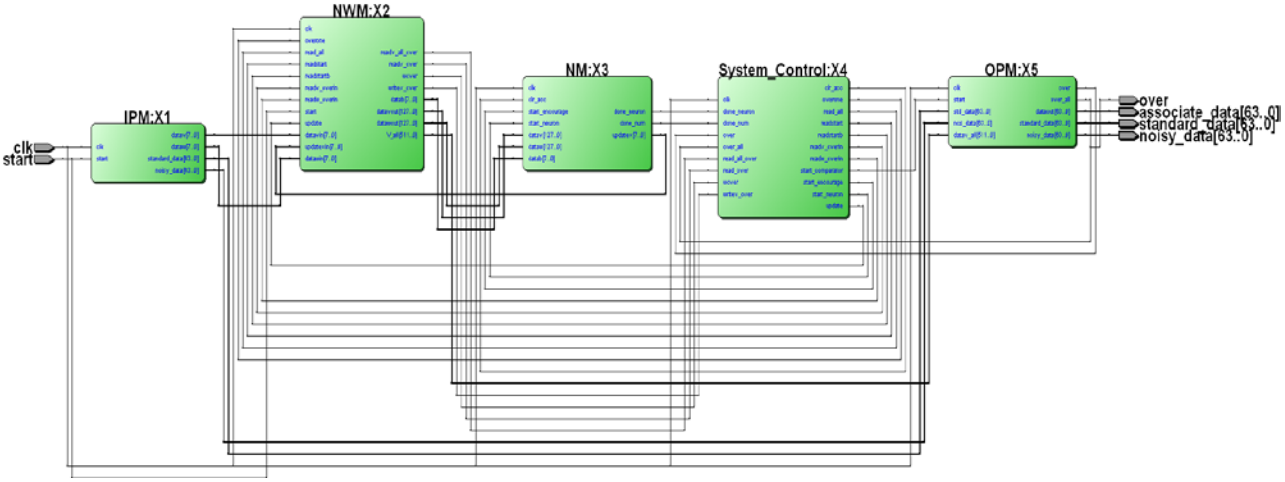

Fig.5. Modularization hardware structure of discrete Hopfield neural network

5 Conclusion

How to simplify the hardware implementation process of different kinds of neural networks based on FPGA has been studied in this paper. The proposed modularization and digitization can not only take full advantage of the parallel and distributed computation ability of artificial neural network, but also construct the desired neural network with ease. It provides a novel approach for hardware implementation of artificial neural network.

Acknowledgement

In this paper, the research was sponsored by the National Science Foundation of China (Project No. 21227008) and Jilin Provincial Science and Technology Agency (Project No. 20130102028JC).

References

[1] Seul Jung, Sung Su Kim. Hardware Implementation of a Real-Time Neural Network Controller with a DSP and an FPGA for Nonlinear Systems [J]. IEEE Transactions on Industrial Electronics. 2007:54 (1):265-271.

[2] Geun-Hyung, Sung-su Kim, Seul Jung. Hardware Implementation of a RBF Neural Network Controller with a DSP 2812 and an FPGA for Controlling Nonlinear Systems [C]. International Conference on Smart Manufacturing Application. Gyeonggi-do, 2008:167-171.

[3] Wang Jian, Wan Dongmei, Mao Zongyuan. Hardware Implementation Study of A New Associative Memory Neural Network [J]. Computer Engineering and Applications. 2003:14:25-29.

[4] Murray, A.F.; Del Corso, D.; Tarassenko, L. Pulse-stream VLSI neural networks mixing analog and digital techniques [J]. IEEE Transactions on Neural Networks. 1991:2 (2):193-204.

[5] Granado, J.M.; Extremadura Univ.; Caceres; Vega, M.A.; Perez, R.; Sanchez, J.M. Using FPGAs to Implement Artificial Neural Networks [C]. 13th IEEE International Conference on Electronics, Circuits and Systems. Nice, 2006:934-937.

[6] Abramson, D.; Smith, K.; Logothetis, P.. FPGA Based Implementation of a Hopfield Neural Network for Solving Constraint Satisfaction Problems [C]. Proceedings 24th Euromicro Conference. Vasteras, 1998:2:688-693.

[7] Liu Peilong. The Research and Design of Hardware Implementation of Neural Network Based on FPGA [D]. Master Degree Thesis, University of Electronic Science and Technology of China. 2012.

[8] XUE Wei-qin, LI Li-hua, DAI Ming. Implementation method of ANN system based on FPGA [J]. Electronic Design Engineering. 2010:18 (9):151-154.

[9] Gao Juan. Theory and Simulation Examples of Artificial Neural Network Second Edition [M]. Beijing: China Machine Press. 2007:6-17.