IOP Conf. Series: Materials Science and Engineering 466 (2018) 012094 doi:10.1088/1757-899X/466/1/012094

Design and Implementation of Intelligent Hardware of Neural Network Based on FPGA

Haiming Jing and Ning Zhao

Department of Information Science and Technology, Department of Economic and Management, Shijiazhuang Tiedao University, 17 Northeast, Second Inner Ring, Shijiazhuang, China.
Email: jimperlov@yeah.net

Abstract. Neural networks, with its inherent parallel nature, is not suitable to realize in micro-processor, however, FPGA is widely applied in the hardware realization of neural network, for its parallel nature, abundant embedded hardware core unit with multiplying and adding function, and memory resources. According to the hardware circuit design, the neural network system is divided into several modules, and then the hardware implementation of each module is designed. The hardware circuit is realized by Verilog language, and the simulation test is carried out in Quartus II. According to the simulation results, whether the expected results are achieved is analyzed.

1. Introduction

A great deal of research work has been carried out on the hardware implementation of neural network at home and abroad, mainly focusing on the development of neuron chip and the implementation of neural network on FPGA[1]. The FPGA-based approach uses the advantages of both hardware and software models[2]. FPGA is the product of further development based on PAL, GAL, CPLD and other programmable devices. It is a semi-custom circuit in the field of application specific integrated circuit (ASIC). It not only solves the defect of custom circuit, but also overcomes the shortcoming of limited number of programmable devices[3].

In 1994, Botros N M and Abdul-Aziz M implemented a fully digital multi-layer perceptron network on Xilinx FPGA.Each neuron was implemented on two XC3042 FPGAs, and the network was offline trained[4].

An intelligent hardware implementation method of BP neural network based on FPGA is attempted, and it is applied to the recognition of 26 upper-case English letters.

2. Implementation Methods of Each Computing Module

The main operations in the system are the multiplication and accumulation of inputs and weights, and the conversion of the preliminary results to nonlinear continuous values through the excitation function or its derivatives. Some efficient IP cores are designed, which not only improves the speed of system operation, but also saves hardware resources.

2.1. Implementation Method of MAC Module

MAC module is the core component of neural network, which is used to achieve multiplication and addition of connections and inputs of weights of neuron. It is convenient to calculate the error through the excitation function. Therefore, according to the characteristics of the sigmoid function, the derivative of the sigmoid function is used in the back propagation stage.

MAC output requires 16 bits to guarantee accuracy, so the multiplier and adder need to be 16. When two 16 digits are multiplied, the 32 digits outcome will not affect the accuracy of the lower 16 digits, so after the multiplier is finished, the 32 digits will be turned into 16 digits by the intercept operation and then enter the later operation. Since the weights are stored in a two-dimensional array, if you want to extract the corresponding weights, a multiplier will be used to accurately send the required values into the multiplier. Since the neural network input is composed of a large number of neurons and longer vectors, we will design an 8-way 16-bit data gating device, which will simplify the whole complex input slightly and facilitate the subsequent operation. The design structure of the MAC module is shown in Figure 1.

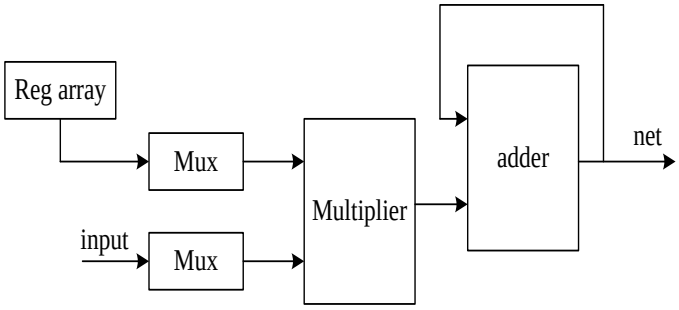

Figure 1. Schematic diagram of MAC module structure.

2.2. Implementation Method of Weight Update Module

Weight update module is still a multiplication and addition operation, which is different from MAC module in that it has less input and output. In addition, the result of weight update module should be sent to the array of stored weights to overwrite the original weights. Therefore, the weight accumulator module first sends the learning rate n, error and output to the multiplier, calculates the amount of update, then adds the amount of update and the original value to the adder to get a new weight, and finally sends the updated weight into the storage unit through the multi-channel data selector to complete the training.

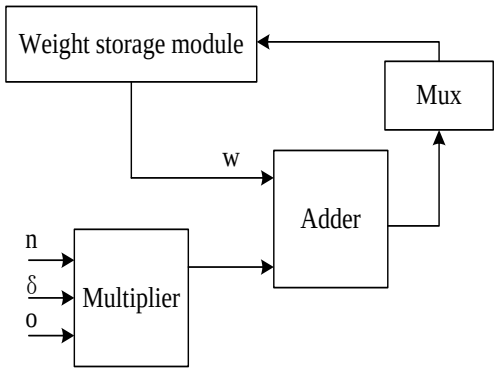

Figure 2. Schematic diagram of weight updating module.

2.3. Implementation Method of Excitation Function Module

PWL approximation method is used to realize the excitation function, and a certain number of broken line segments are used to approximate a function. It is still necessary to look up the table when choosing broken line segments, but because the number of broken line segments is limited, it takes less resources. The more the broken lines are selected, the more it can approximate to the original function. But in order to improve efficiency, we should ensure that the function does not distort with less broken lines. Because the sigmoid function is symmetrical about (0,0.5), it can be well represented by eight broken line. Each broken line is determined by slope and starting point. The most important parameter in FPGA implementation is the slope of broken line. Slope is saved by binary in circuit, and the approximate broken line is adjacent to each other and the slope between lines is two

times. Therefore, the slope of the next segment can be obtained by simply shifting the representation data of the previous broken line one bit to the right[5]. The correspondence of input and output of functions is shown in Table 1.

Table 1. Correspondence of broken line of sigmoid simulation line.

Input range	Input	Output	Output range
x→x	xxxx.xxxxxxxxxxx	x.xxxxxxxxxxxxxx	x→x
0→1	0000.abcdefghijkl	0.10abcdefghijkl0	0.5→0.75
1→2	0001.abcdefghijkl	0.110abcdefghijkl	0.75→0.875
2→3	0010.abcdefghijkl	0.1110abcdefghijk	0.875→0.9375
3→4	0011.abcdefghijkl	0.11110abcdefghij	0.9375→0.96875
4→5	0100.abcdefghijkl	0.111110abcdefghi	0.96875→0.984375
5→6	0101.abcdefghijkl	0.1111110abcdefgh	0.984375→0.9921875
6→7	0110.abcdefghijkl	0.11111110abcdefg	0.9921875→0.99609375
7→8	0111.abcdefghijkl	0.11111111abcdefg	0.99609375→1

3. Functional Simulation and Error Analysis

Function implementation and simulation are implemented in Quartus II environment, and error analysis of simulation results will be done.

3.1. Simulation Results of MAC Module

As shown in Figure 3, it is the function simulation result of a single MAC unit in the forward propagation process. clk is the clock signal, opa is as the input interface of the connection weights of neurons, opb is the output of the former layer and as the input of this layer. out is the output of the MAC module in the forward propagation process. out is the multiplication by accumulation of opa and opb. they are both 16-bit binary numbers, the result of multiplication is 32 bits. In order to ensure the uniformity of input in the front and back layers, the output results are stored in a 32-bit register, then the high 16 bits of the 32-bit register are assigned to the out, and the low 16 bits are discarded, because in the excitation function, The size of the last sixteen places has little effect on the final output. There are 32 neurons in the hidden layer and 26 neurons in the output layer of the neural network, so it is necessary to use a large number of mac modules for multiplication and accumulation.

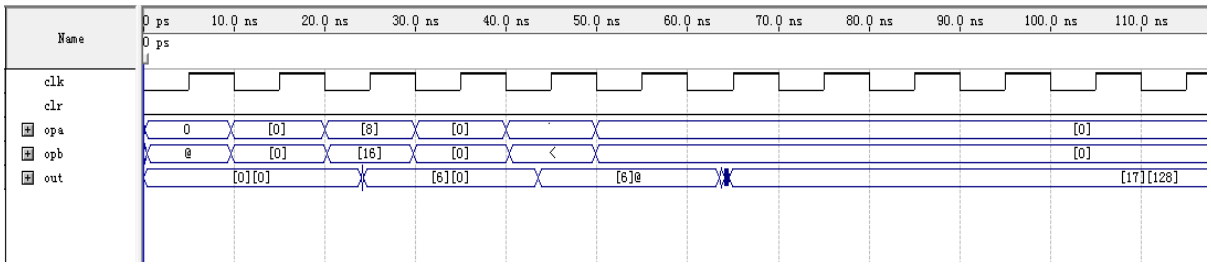

Figure 3. MAC module simulation image.

Through image analysis, we can see that the MAC module has precisely completed the output of net. But in the actual hardware circuit, the error still exists.

3.2. Simulation Results of Sigmoid Module

Figure 4 shows the simulation image of the excitation function module. The input of the excitation function is the output net in the MAC module. It is named fin in the module. In Figure 5, the output of the excitation function is fout, the output of fout_der is the derivative of the excitation function. The fin signal has a 16-bit binary number, the highest bit is a symbol bit, the next 3 digits are integers, and the last 12 digits are decimal digits. Through looking up the table, according to the input conditions, we can find the corresponding output relationship, complete the output of the excitation function. According to the characteristics of the excitation function, no matter how big the input is, the output is

always between - 1 and 1. In order to ensure the 16-bit input, the highest bit of the output is still set to be a symbol bit, and the last 15 bits are all decimal digits.

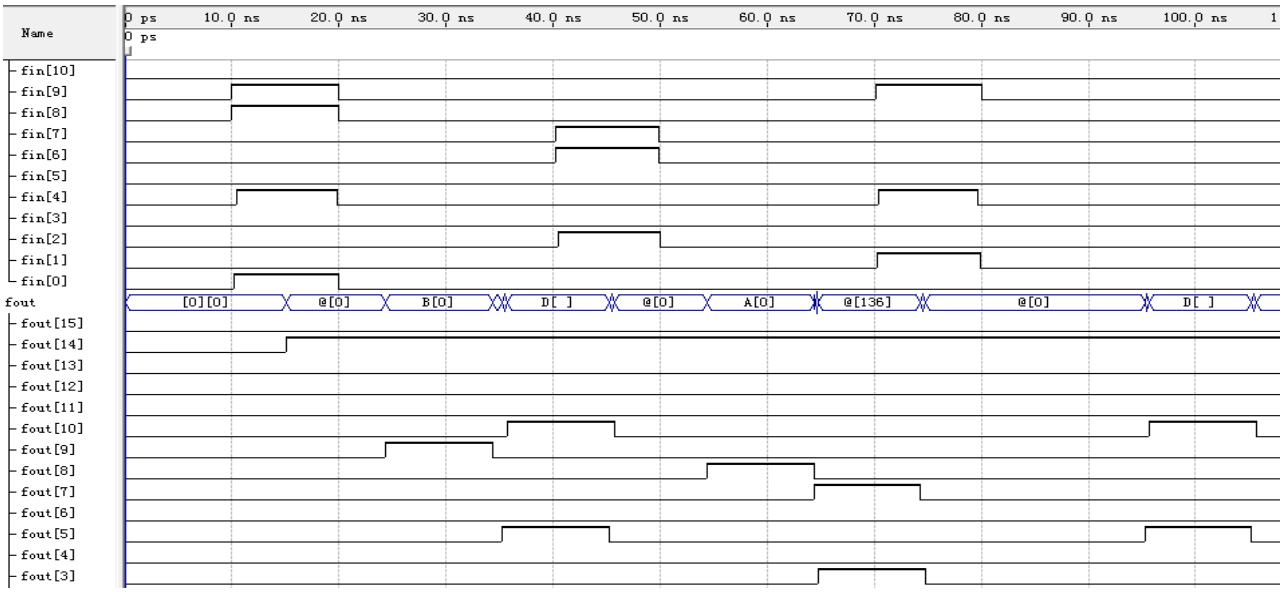

Figure 4. Sigmoid module simulation waveforms.

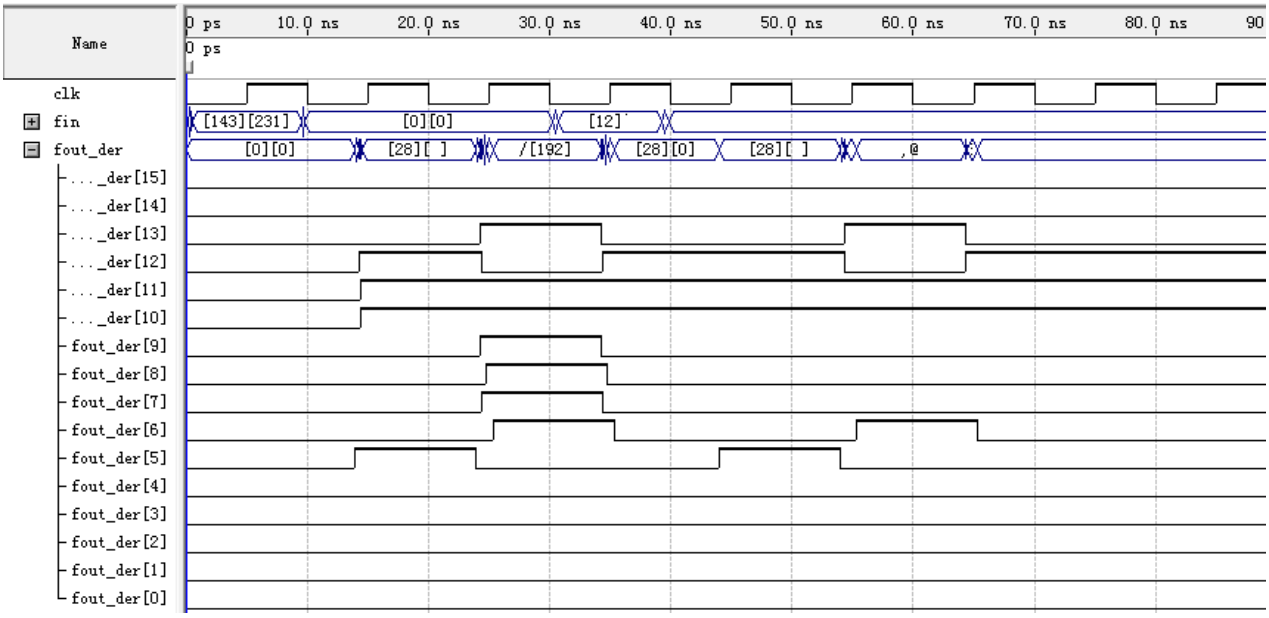

Figure 5. simulation waveforms of derivative of sigmoid function.

3.3. Simulation Results of Weight Update Module

Figure 6 is the simulation result of the weight updating module, in which fd is the learning rate of neural network, it is set to 0.2, namely 0001100110011001binary form, wu is the learning error, w is the weight value before updating, out is the updating amount of the weight value, it is the result of multiplication of three inputs, it is originally a 48 bit output, through the interception operation, the input and output is kept at high 16 bit. By analyzing the image, we can see that the module realizes the expected function.

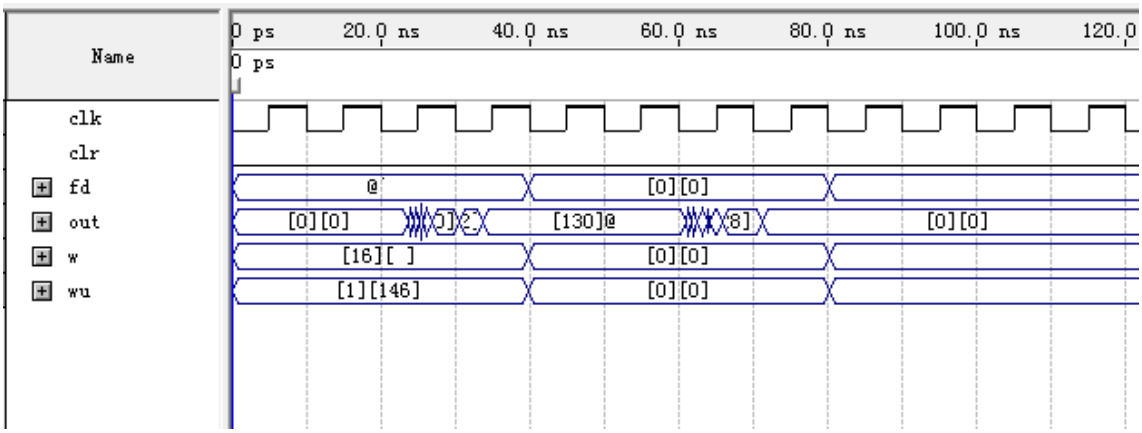

Figure 6. Weight update module emulate image.

4. Conclusion

Through the above design, A simple three-layer BP neural network is constructed by using neuron and excitation function. By inputting training samples, the output results are obtained and compared with the expected output to get the error. The training is completed according to the error correction weight. A more perfect neural network system is realized, which has certain autonomous learning ability and achieves the preliminary goal. There are still many shortcomings in this paper, for example, in the process of hardware circuit design, a large number of multiplication and accumulation operations are invoked. The multiplication operation in this design is serial, but the efficiency is still insufficient.

5. Acknowledgement

This research was financially supported by the National Science Foundation(F020502) and Youth Foundation by Education Department of Hebei Provine(Z6750124)

6. References

[1] Dinu A, Cirstea M N, Cirstea S E. Direct Neural-Network Hardware-Implementation Algorithm. *IEEE Transactions on Industrial Electronics*, 2010, 57(5): 1845-1848.

[2] Remzi Tuntas. A new intelligent hardware implementation based on field programmable gate array for chaotic systems. *Applied Soft Computing*, 2015, 35: 237-246.

[3] Xiong Zhang. Hardware implementation of BP neural network in image recognition [D]. *Lan Zhou JiaoTong University*, 2014.

[4] Botros N M, Abdul-Aziz M. Hardware implementation of an artificial neural network using field programmable gate arrays (FPGA's). *IEEE Transactions on Industrial Electronics*, 1994, 41(6): 65-667.

[5] Joon-sik SON. Apply for Back-Propagation Neural Network to Control a GMA Welding Process[A]. *Science and Engineering Research Center.Proceedings of 2015 International Conference on Education, Management and Systems Engineering*(EMSE 2015)[C].