MULTI-CORE COMPUTING UNIT FOR ARTIFICIAL NEURAL NETWORKS IN FPGA CHIP

Marek Bohrn and Lukas Fujcik
Department of Microelectronics, Brno University of Technology, Udolni 53, Brno, Czech Republic

Keywords: Artificial neural networks, Computation acceleration, FPGA, VHDL, Spartan-3.

Abstract: This article describes a design and features of a multi-core unit for performing computing operations required for artificial neural network functioning. Its purpose is to speed up computing operations of the neural network. The number of computing cores can be altered as needed to achieve the required performance. VHDL language has been used to build this module. It has been optimized for the Spartan-3 family FPGA chips from Xilinx. These chips are favorable because of their low price and a high number of on-chip multipliers and block memory units. Spartan-3 chips facilitate parallel computing operations within neural networks to a very high level and thus help to achieve high computing power.

1 INTRODUCTION

Artificial neural networks are suitable for various tasks, such as pattern recognition, signal processing, classification, function approximation, prediction, data compression, etc.

It is possible to apply analog circuits, micro-controllers, computers, signal processors, and FPGA chips of neural networks implementation. Analogue neural networks are not used because of their implementation complexity and computing accuracy. Micro-controllers are cheap and easy to use, however, they do not achieve the required computing performance. Implementation of a neural network in a computer is easy, but it is unsuitable for use in industry or as embedded application due to the high cost and/or enormous size.

Signal processors and FPGA chips are suitable platforms for the implementation of neural networks. Signal processors are easier to use, they can be programmed in C language and they are very cheap for this purpose. Their disadvantage is that their computing power cannot be increased easily due to their fixed structure.

FPGA chips, on the other hand, have the capacity to facilitate parallel computing operations at a very high level and their structure can be adjusted to target application. The programming language used for FPGA chips is usually VHDL which is suitable for the description of parallel structures and pipe-line circuits. Application of these techniques allows for achieving the maximum computing power and speed and thus competes with all remaining alternatives of neural networks implementation.

The unit described in this article takes advantage of all benefits of FPGA chips and has been optimized for Spartan-3 family from Xilinx. The solution provided herein performs parallel computing operations and use pipe-line structure. The number of computational cores can be adjusted as needed. The biggest chip of Spartan-3 family (XC3S-5000) can employ up to 100 computational cores (Xilinx, 2006). The XC3S-200 chip was chosen to carry out the implementation and testing of the unit.

2 THEORETICAL ANALYSIS

2.1 Artificial Neuron

Artificial neuron – perceptron (Figure 1) is composed of an input vector, synaptic weight vector, summation unit, activation function and output.

Its function is as follows:

The input vector is multiplied by the weight vector. The results are summed up and they compose the inner potential of the neuron as depicted in Equation 1 (Fausett, 1994).

$$i = \sum_{n=1}^{x} w_n * in_n \quad (1)$$

Bohrn M. and Fujcik L.
MULTI-CORE COMPUTING UNIT FOR ARTIFICIAL NEURAL NETWORKS IN FPGA CHIP.
DOI: 10.5220/0002172101490152
In *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2009)*, page
ISBN: 978-989-8111-99-9

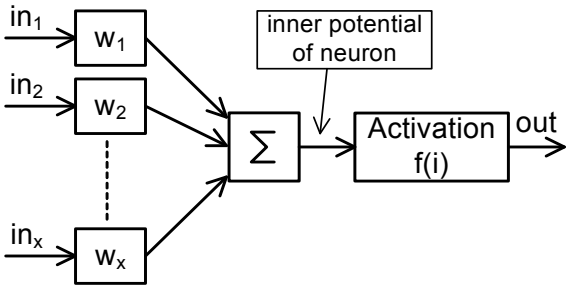

Figure 1: Structure of artificial neuron.

This value is fed into the activation function. The activation function is usually a step function, a sigmoid or ramp function; its task is to saturate the output value within the set limits, usually between 0 and 1.

2.2 Computations Inside Neural Network

Multiplication and summation are the most important computations within the neural network. These can be carried out using the multiply-accumulate (MAC) function, however, the chosen target chip Spartan-3 does not support this particular function. Instead, the MAC operation is replaced by sequential multiplication and summation operations. Spartan-3 chips include hardware multipliers and the summation operations are performed by gates.

The computation of inner potential of a neuron is always performed by one computational block. These blocks are in a parallel arrangement which increases the data throughput capacity as well as the overall computing power of the entire unit. The Spartan-3 chips can work up to 375 MHz under ideal conditions; however, the actual frequency of operation is slightly lower.

For the activation function, the unipolar sigmoid has been chosen (Figure 2) because it is the most common function for neural computations. The activation function is carried out using interpolation from the lookup table. It is possible to add other activation functions to the chip or replace the unipolar sigmoid by another function, if needed.

2.3 Format of Data

The data format for the weight vectors and input information transfer and the computational operations are the same within the neural network. The 18-bit mode is the most suitable width for buses employed in the Spartan-3 chips because of the 18-bit inputs of the embedded multipliers. The block

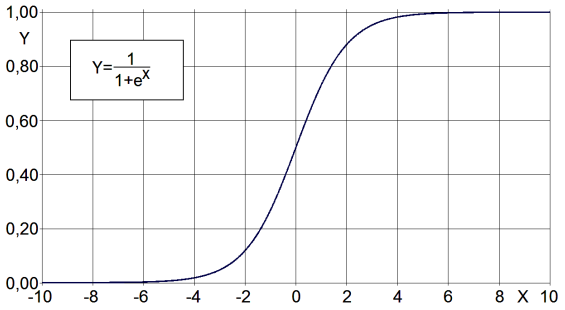

Figure 2: Sigmoid activation function.

memory units can also be adjusted to the 18-bit mode.

The data are represented as a fixed point number and are stored in two's complement. The 6 most significant bits of the data word represent the integer part and the next 12 bits represent the fraction part as illustrated in Figure 3 (Suhap, Becerikili, Yazici, 2006).

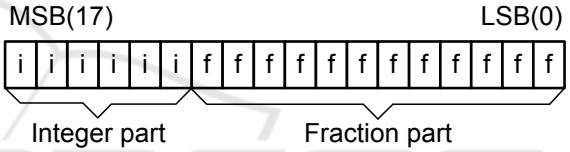

Figure 3: Format of number representation.

3 REALIZATION IN FPGA CHIP

3.1 Circuit Overview

Figure 4 shows a block diagram of the entire circuit for neural network computations. The control logic block, neuron computation blocks, and activation function are the main parts of the circuit.

The control logic includes a map of the neural network which is the basis of data and commands sent out to neuron computation blocks. The selection of weight is done by an address sent to the bus controlling the memory blocks. Weight coefficients are stored in these memory blocks. Each memory block can store up to 1,024 weight coefficients. In the XC3S-200 chip used for testing the design, up to 10,240 synapses can be implemented.

Data coefficients are transmitted on the basis of an external requirement and a neural network computational status. Either data stored in the neuron data memory or data from the input port are used. The selected neuron data memory configuration allows for storage of 1,024 neuron

outputs which means that the unit can handle a neural network consisting of up to 1,024 neurons.

The control logic can also send the following commands to the computational blocks: reset, MAC, BIAS, send result. Each neuron computation block is connected to an individual command bus. By separating the command buses of individual neurons, it is possible to arrange them in a way so that each neuron performs computational operations simultaneously at a different layer of the neural network. Possibly some of the neuron cores can be disconnected by sending only the reset command. MAC command together with BIAS command calculates the inner potential of the given neuron. Once the commands are received by the neuron cores, the computational core starts to multiply the weight coefficients by data inputs and stores the results in the internal register. The send result command initiates the transfer of the inner potential value to the next register. Subsequently, the neuron waits for an impulse to send the data to the activation function.

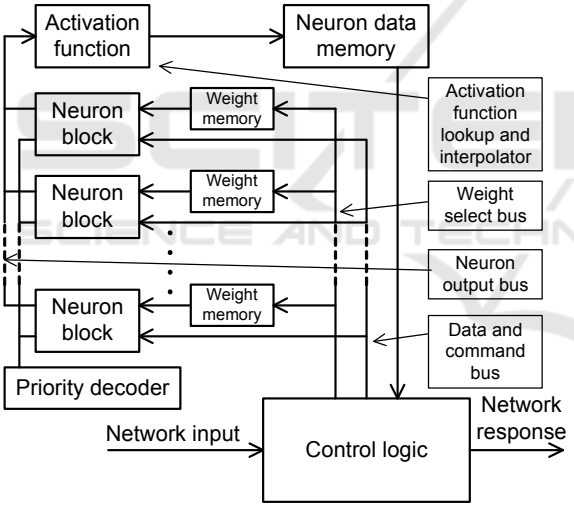

Figure 4: Block diagram of a complete unit.

The output bus of neuron computing blocks is shared by all computing blocks because its capacity utilization should be the low. When required to send the data to the activation function, neurons wait for the priority decoder's impulse. Data are sequentially brought from computing blocks to the activation function and no internal FIFO buffer implementation in necessary.

After the data pass through the activation function, they are stored in the neuron data memory - dual port RAM. If needed they are read and sent from this memory via the data bus to the computing blocks. If the stored data belong to the last neuron layer, they are transferred to the unit output.

3.2 Neuron Computation Block

The block diagram of a neuron computing block is illustrated in the Figure 5. The block is arranged as a three stage pipe-line circuit to increase the speed. The input data of this block are command, data and weight.

In the first stage, the input data are multiplied according to the chosen command, the second stage is composed as a summation block. These two together implement the MAC function and calculate the internal neuron potential. When the send result command is received, the internal potential is sent to the last pipe-line stage and waits for the impulse to be transferred to the activation function.

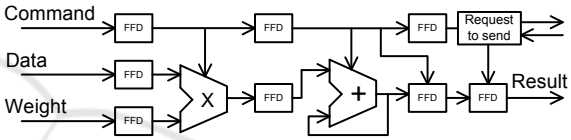

Figure 5: Block diagram of a neuron computation block.

After receiving the reset command, the internal potential is set to zero and the block can start computing another neuron regardless of the fact whether the result was sent or is waiting in the last stage, the result remains in an unchanged state.

3.3 Activation Function

Activation function is composed by an interpolated lookup table. The activation function circuit structure is pipe-line similar to computing block, it includes a multiplier and an adder.

The activation function has only one input which is divided into two parts. The upper part (bits 17 .. 9) is transferred into two lookup memory units. The lookup value for interval offset is stored in one of them and gradient of interval in the other. Gradient is multiplied by the lower part of the input (bits 8 .. 0) and is transferred to the summation part. Offset is delayed by a one clock period and it is also transferred to the summation block. The interpolated value of the activation function connected to the output is achieved in this way. The activation function output is stored in the neuron data memory which can be accessed by the control logic.

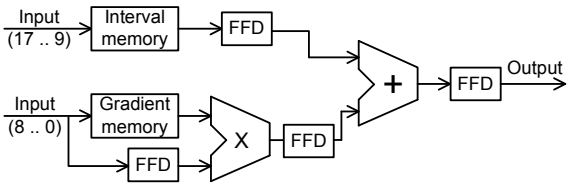

Figure 6: Block diagram of an activation function interpolator.

Since the activation function is realized by a lookup table, it is easy to change its shape using different coefficients. The circuit function can be extended by adding another block with a different activation function. For example, the activation function for output layer will be a step function if we require binary result.

3.4 Testing of Complete Circuit

The circuit was described in VHDL language, the source code is modular and optimized for the Spartan-3 family chips from Xilinx and fully synthesizable. Since most of the circuit is arranged into pipe-lines, it can work with the clock frequency up to 133MHz.

The target XC3S-200 chip includes 12 blocks of RAM and 12 dedicated multipliers. Activation function requires at least one block RAM and one multiplier. Neuron data memory needs one block RAM. For this reason it is possible to implement 10 computing blocks, needing one block of on-chip RAM for storing weight coefficients and one multiplier for computing.

The circuit function was tested on an application recognizing hand-written numbers. A network with 88 neurons in the input layer, 40 neurons in the hidden layer and 10 in the output layer was realized. The network model and the training algorithm were realized in a computer and weight coefficients were transferred to weight memory blocks on FPGA chip (Masters, 1993).

Calculation of response of the neural network require 88 times 4 clock periods for the hidden layer, 40 clock periods for the output layer plus latency of the activation function of 4 periods. For one character recognition, 396 clock cycles were needed. At the working frequency of 133 MHz the circuit can recognize 336000 numbers per a second. In comparison to the single core signal processor which would need for such a calculation at least 3920 clock cycles, this is an excellent result.

4 CONCLUSIONS

The neural network function is very demanding regarding to the computing power. In the FPGA chips, however, it is possible to parallelize the calculations very efficiently. The designed circuit speeds up the computation for neural network approximately 10 times in comparison to signal processor. With using the biggest chip XC3S-5000 from the Spartan-3 family, it would be possible to implement up to 100 computing cores into the circuit and thus increase the computing power theoretically up to 100 times.

The entire circuit is very modular and allows for realization of neural networks in various configurations. From very small networks up to networks consisting of thousands of neurons and hundreds of thousands synapses with a very high computing power kept.

Spartan-3 family chip was chosen for the implementation because of its low price and good accessibility of development kits. Its disadvantage is that it cannot process the MAC command in one clock cycle. This problem is solved by higher FPGA families, such as Virtex-5 family chips are able to process up to 580 Giga MAC per second while their computing of MAC command is performed within one clock cycle.

ACKNOWLEDGEMENTS

This research has been supported by the Czech Ministry of Education in the frame of long term Program Plan MSM 0021630503 MIKROSYN New Trends in Microelectronic Systems and Nanotechnology and by the Czech Science Foundation as the project GA102/08/1116.

REFERENCES

Ohomondi, Amos, R., Rajapasake, Jagath, C., 2006. *FPGA Implementations of Neural Networks*, Springer Netherlands

Fausett, L., 1994. *Fundamentals of Neural Networks*, Prentice Hall New Jersey

Masters, T., 1993. *Practical Neural Network Recipes in C++*, Academic Press California

Xilinx, 2006. Spartan-3 FPGA Family: Complete Data Sheet, Xilinx company

Suhap, S., Becerikili, Y., Yazici, S., 2006. *Neural Network Implementation in Hardware Using PFGAs, 13th International Conference, ICONIP 2006*, Springer-Verlag Germany