ELECTRONICS' 2004 22-24 September, Sozopol, BULGARIA

FPGA IMPLEMENTATION OF ARTIFICIAL NEURONS

Ognyan Manchev, Blagomir Donchev, Kostadin Pavlitov

ECAD Laboratory, TU – sofia, 8 Climent Ochridsky str, 1700, Sofia, Bulgaria, phone: +359 2 9652190, e-mail: manchev@ecad.tu-sofia.bg, donchev@ecad.tu-sofia.bg, knp@tu-sofia.bg

Keywords: neural network, VHDL, field programmable gate arrays (FPGA), XILINX.

This paper presents the design of a digital artificial neuron, which is the base building block of modern systems for classification and recognition, called Artificial Neural Networks. The authors outline the benefits of programmable devices in use today and present an effective approach to designing parallel computing units and implementing them into FPGA devices. A digital neuron is developed and implemented in a Xilinx® FPGA.

1. INTRODUCTION

Controlling modern systems is a difficult to implement task. The number of controlled parameters and monitored values is greatly increasing and present means of computing become unusable, inefficient or inadequately expensive. Conventional computers suffer the lack of I/O connections, speed and stability in the area of industrial applications, parameter control, optical or sound recognition.

An interesting approach to solving problems concerning multiple input parameter handling and noise – resistance is processing data the way the human brain does. The brain is a highly complex, nonlinear and parallel information processing system with the capability of organizing neurons (basic processing units) such as to perform these computations on a speed, greater than the speed of the fastest algorithmic machine existing today. The neuron – a basic cell of this parallel computer comprises of receptors (dendrites), computing body (core) and synaptic connections to adjacent neurons. Synaptic connection properties define how the brain processes particular input information. The organization of these connections is referred to as knowledge.

A computational system modeling the way the brain processes information is called Artificial Neural Network (ANN) and therefore its components are called artificial neurons. Artificial neurons are organized in layers. Cascaded layers form an ANN which can classify objects or perform signal processing tasks.

Differences in principles of operation of conventional algorithmic computers and ANNs define fields of application for both of them. While algorithmic approach can be used to do explicitly defined tasks, such as word processing or accounting, parallel processing structures manage incompletely defined problems, concerning signal processing, sound and image recognition or shape classification.

Results from a comparison between conventional computers and neural networks can be summarized in the following table:

Characteristics	Conventional computers	Artificial Neural Networks
Popularity	Well known	In development
Methods for implementing knowledge	Developed algorithms	Specific education
Learning method	By rules	By example
Processing style	Sequential operation	Parallel processing
Result generation	Decision reproduction	Generalization
Speed	Require big processors	Require multiple custom-built chips
Attitude to noise	Noise – sensitive	Noise – tolerant
Error-proof	Susceptible to input errors and faults	Fault – tolerant
Field of application	Explicitly defined tasks (accounting, word processing, math)	Outlined problems (Sensor processing, speech recognition, image recognition)

Table 1 - ANNs vs algorithmic machines

Table 1 is based on the following properties and capabilities of ANNs [1]:

- Nonlinearity;
- Input – output mapping;
- Adaptivity;
- Evidential response;
- Contextual information;
- Fault tolerance;
- VLSI and PLD implementability;
- Uniformity of analysis and dsign;
- Neurobiological analogy.

The next section describes the functional description, mathematical model, functional blocks and interface signals of the neuron. Section III contains results from behavioral and timing simulation, and Section IV, the conclusion.

2. FUNCTIONAL DESCRIPTION

2.1. Mathematical model of an artificial neuron

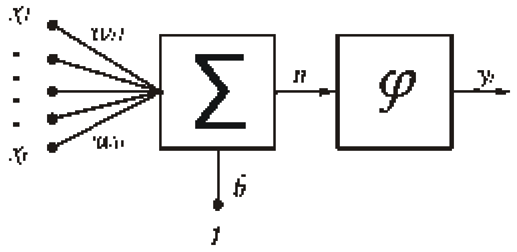

Figure 1 - Mathematical model of the neuron

Fig. 1. represents the common mathematical model of an artificial neuron**.** The linear combiner $\sum$ forms the activation potential n – linear product of inputs x_1 - x_r

and corresponding weights $wi_1 - wi_r$. Threshold b is then applied to form the input of the activation function φ. Network performance depends strongly on the type of this function, which may be threshold, linear or sigmoid. Equation (1) describes analytically neuron behavior:

$$y_i = \varphi(\sum_{j=1}^{R} w_{ij} x_{ij} - b) \qquad (1),$$

where i denotes the number of particular neuron, x_{ij} and w_{ij} – inputs and their weights.

2.2 Neuron structure

Developed neuron block structure, shown on Fig. 2, resembles the mathematical model. Inputs A_1 and A_2 are multiplied by corresponding weights w_1 and w_2 and products are added to the threshold b. The activation function is applied to the linear product and forms the output Y.

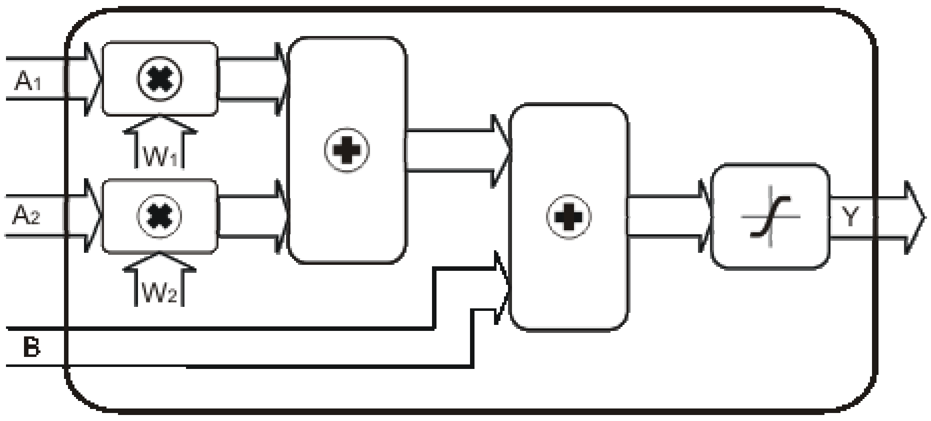

Figure 2 - Block diagram of the neuron

Neuron modules work as follows:

Multiplier:

The multiplier performs fast multiplication of two eight – bit signed binary numbers, considering the most significant bit as the sign and setting the most significant bit of the product in accordance to the seventh bits of inputs. Multiplication is performed on the rest of the bits.

The multiplier generated by the synthesizing software utilizes fast carry logic in Xilinx® FPGA. This special logic in modern FPGA circuits allows multiplication in just one clock cycle. Hardware pre – implemented pipelined multipliers in high – end platform FPGAs perform even faster multiplication for DSP designs.

Macros generated by synthesis tool:

7x7-bit multiplier: 2

Adder

The adding circuit produces a linear product of input vectors and corresponding weights to form the activation potential n. Numbers are represented in two's complement format, eight bit each for both inputs and output. The first adder operates on neuron information inputs and its output is added to the threshold of the neuron.

Adding circuits are recognized and implemented by synthesis tool, inferred macros:

8-bit adder: 2

Hyperbolic tangent

This artificial neuron uses hyperbolic tangent as its activation function. Hyperbolic tangent is defined by the expression

$$\tanh n = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \qquad (2),$$

where n is the activation potential (linear product of inputs and weights).

Hyperbolic tangent can be implemented using look-up table. Values for the activation potential in the interval (-5,5) are observed to form the following table:

№	**n**	**tanh(n)**	**n** $_{16}$	**tanh(n)** $_{16}$	**№**	**n**	**tanh(n)**	**n** $_{16}$	**tanh(n)** $_{16}$
1	0	0	$00	0	17	2,5	0,986614	$40	1A
2	0,15625	0,154991	$04	4	18	2,65625	0,990189	$44	1A
3	0,3125	0,30271	$08	8	19	2,8125	0,992813	$48	1A
4	0,46875	0,437189	$0C	B	20	2,96875	0,994737	$4C	1A
5	0,625	0,5546	$10	E	21	3,125	0,996147	$50	1A
6	0,78125	0,653424	$14	11	22	3,28125	0,997179	$54	1A
7	0,9375	0,734072	$18	13	23	3,4375	0,997936	$58	1A
8	1,09375	0,798243	$1C	14	24	3,59375	0,998489	$5C	1A
9	1,25	0,848284	$20	16	25	3,75	0,998894	$60	1A
10	1,40625	0,886695	$24	17	26	3,90625	0,999191	$64	1A
11	1,5625	0,915825	$28	17	27	4,0625	0,999408	$68	1A
12	1,71875	0,937712	$2C	18	28	4,21875	0,999567	$6C	1A
13	1,875	0,954045	$30	18	29	4,375	0,999683	$70	1A
14	2,03125	0,96617	$34	19	30	4,53125	0,999768	$74	1A
15	2,1875	0,975137	$38	19	31	4,6875	0,99983	$78	1A
16	2,34375	0,981749	$3C	1A	32	4,84375	0,999876	$7C	1A

Table 2 - hyperbolic tangent values

Column *n* lists discrete decimal values of the activation potential. Values are chosen by estimating the hexidecimal equivalent of their fraction and skipping three numbers. n_{16} lists *n* in hex format, while *tanh(n)* and $tanh(n)_{16}$ represent function output in both decimal and hexidecimal numbers.

Table 2 is used as truth table for an eight – bit decoder, implementing the hyperbolic tangent function (referred to as "tansig" in Matlab™). The table suggests the following optimizations:

- Input value 0x3C leads the function into saturation (1A) therefore the sixth bit of the argument can be ignored;

- Output is in range (0x00, 0x1A). Bits six and five can be tied to zero;
- The function is odd, so negative outputs are equal to positive ones, changing their most significant bit (sign). The sign bit of the function is tied to the seventh bit of *n*.
- A block RAM implementation of the hyperbolic tangent saves FPGA cell resources. In the current design target device – the Xilinx® Spartan™II FPGA the ten SelectRAM modules enable fast table lookup tanh computation and decrease occupied area.

2.3. Interface

Name	**Direction**	**Description**
A_{i1} (7:0)	In	Input 1
W_{i1} (7:0)	In	Weight 1
A_{i2} (7:0)	In	Input 2
W_{i2} (7:0)	In	Weight 2
B_i (7:0)	In	Threshold
ldAB	In	Valid input
clk	In	System clock
rst_b	In	System reset
Y_i (4:0)	Out	Output
Flag_signed	Out	Output sign

Table 3 - Artifical neron interface

Table 3 lists the module interface. Input and output busses are eight – bit wide, representing signed binary numbers. Control signals are operated as follows:

Clk: System clock signal. All input signals are registered with respect to clk;
Rst_b: Asynchronous system reset. Rregisters are reset regardless of clk value;
ldAB: Inputs are valid when ldAB is high;

3. SYNTHESIS AND SIMULATION RESULTS

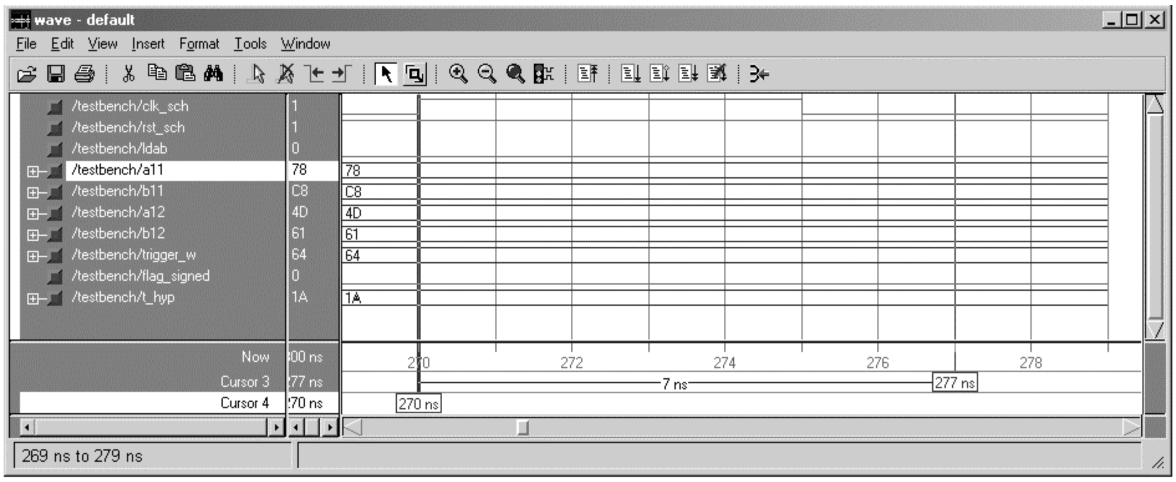

Figure 3 - Timing simulation results

Fig. 3 presents timing simulation of the VHDL model. Measured time is the maximum output required time after clk t_{out}=7ns. Table 4 summarizes device resource usage and timing parameters.

Parameter	**Value**
Device Utilization	
Number of External IOBs	48/ 140 (34%)
Number of SLICEs	86/1200(7%)
Number of GCLKs	1/ 4 (25%)
Timing summary	
Maximum Frequency	93.923 MHz
t_{setup}	10.961 ns
t_{hold}	5.140 ns
Maximum output required time after clock	max 7.589 ns
Clock to setup	11.047 ns

Table 4 - Synthesis results

4. CONCLUSION

This article presents the structure of common artificial neurons, the development of a two-input eight-bit digital artificial neuron and its implementation in an Xilinx® FPGA device. This platform provides a multitude of benefits for implementing ANNs such as incorporating an ANN and a control circuit in a single chip, generation of IP core library and operation at high frequencies. The neuron module's maximum frequency is over 90MHz and resource utilization summary shows that an artificial network comprising five to ten neurons can be implemented in a typical FPGA chip. These results ensure that the developed neuron can be used as a building block for modern control systems comprising of ANNs of the Feed-forward – Back-propagation type.

5. REFERENCES

[1] Hagan, M.T., Demuth, H.B., Beale, M., *Neural Network Design*, PWS Publishing Company, Boston, 1995.

[2] Cichoski, A., Unbehauen, R., *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons, New York, 1994.

[3] Haykin, S., *Neural Networks – A Comprehensive Foundation*, Macmillan College Publishing Company, NY, 1994.

[4] Pavlitov, K., *Application of the Spartan II FPGA circuits in artificial neural network realizations*.

[5] Pavlitov, K., *Xilinx Spartan II Library Module Signed Multiplier*.

[6] Pavlitov, K., *Nonlinear TanSig Converter Based on Spartan II Xilinx FPGA*.