A New Method for Hardware Implementation of Artificial Neural Network Used in Smart Sensors

Ştefan ONIGA, Atilla BUCHMAN

Electrotechnical Department

North University of Baia Mare, Faculty of Engineering

62A Victor Babes Str., 430083 Baia Mare, Romania

Phone: +40-262-218509 e-mail: onigas@ubm.ro

Abstract

The use of neural networks to add learning and adaptive behavior to smart sensors is essential and the FPGA implementation is an easy an attractive way for hardware implementation. This paper presents a new method for hardware implementation of neural network using the System Generator tool for Simulink developed by Xilinx Inc. to implement high-performance DSP systems in FPGA. This method allow the easy generation of hardware Description Language (HDL) code from a system representation in Simulink. This VHDL design can then be synthesized for implementation in the Xilinx family of FPGA (Field Programmable Gate Arrays) devices. With this method it is possible to add neural network specific, user-created, Simulink blocks to the Xilinx Blockset. The Matlab is used to perform the off-chip learning task and the neural network weights are transferred automatically from Matlab workspace to weight (ROM) memory. It this way is possible to create "application specific neural networks" in an easy and fast time to market way.

1. INTRODUCTION

In recent years, computers have penetrated in almost any field of activity, but many applications and conventionally input devices such us keyboards, mice, joysticks, etc., are not enough "user friendly" and many times require parameters setting that can't be made by a regular user, without some knowledge.

The efforts made world wide by the large numbers of universities and research organizations that are involved in designing and building natural user interfaces it seems to be not enough because of the lack of adaptation and learning capabilities. The use of neural networks to add learning and adaptive behavior to smart sensors is essential and the FPGA implementation is an easy an attractive way for hardware implementation.

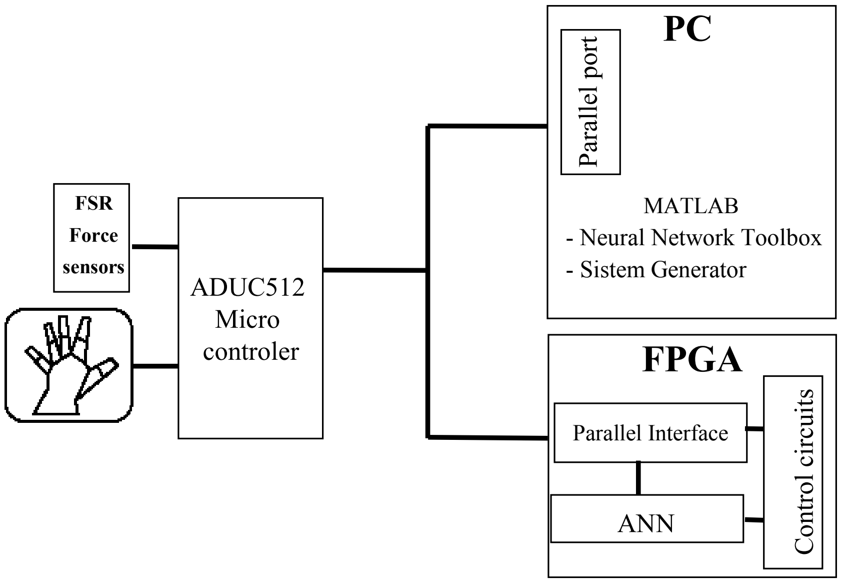

Fig. 1 The codesign platform

This paper presents a new method for hardware implementation of artificial neural networks (ANN) in field programmable logic devices (FPGA) that can be used in intelligent sensors development. Among possible applications are intelligent computer peripherals enabling people with any kind of handicap to use computer and communicate, as any kind of industrial or domestic device with learning and adaptive capabilities.

The goal of this work was to develop hardware-software codesign platform enabling the fast development of intelligent interfaces with the addition of hardware sensors and VHDL modules.

2. THE HARWARE-SOFTWARE CODESIGN PLATFORM

This work use a platform, shown in Figure 1, developed to facilitate the use of codesign techniques. The platform was developed in order to provide a fast prototyping environment. The Aduc 512 microcontroler is used to implement the Data Acquisition System and to adapt signal sensors to neural network input requirements. The reconfigurable device (XC2S50 Xilinx) is used to implement the neural networks and other logic blocks of the same application. The System Generator tool for Simulink developed by Xilinx Inc. allow the easy generation of hardware Description Language (HDL) code from a system representation in Simulink. This VHDL design can then be synthesized for implementation in the Xilinx family of FPGA devices. The developed framework allows device communication with a PC in order, to perform off-chip training task or, to transfer data for analysis. Software is designed to manage the communication protocol with Matlab via parallel port.

The platform may be used in three ways. The first one is the neural network simulation and learning phase of the weights, the second is the network design and hardware implementation using System Generator tool for Simulink and Xilinx ISE, and the third is normal use of the network (propagation phase).

2.1. Learning phase

Training of the neural network can be executed using a given set of inputs with the corresponding outputs. The inputs for training are collected via parallel port of a personal computer running Matlab, and a data acquisition program developed by author. Input and output sets are stored in a file and will be used to determine neural network weights.

The desired network architecture is simulated using Neural Network Toolbox and the neural network weights are transferred automatically from Matlab workspace to weight (ROM) memory represented in Simulink. Many networks architecture trained with different methods could be simulated and the network that is best performing for given application is choused for hardware implementation.

2.2. Implementation phase

First step for transfer the neural network from software simulation to hardware implementation is the network modeling with System Generator tool for Simulink, using Xilinx blocks or user created, neural network specific blocks. One layer could be created using only one ANN block from user created libraries and the block parameters (number of neurons, weights, bias) are loaded automatically from Matlab workspace. If the designed system is well performing in simulation it could be transformed in VHDL code that is made automatically by System Generator Tool for Simulink, developed by Xilinx.

To increase hardware performance, most System Generator blocks are implemented in hardware using Xilinx Smart-IP (Intellectual Property) LogiCOREs. These modules make optimal use of FPGA resources to maximize performance.

During code generation, the System Generator creates all project files that are necessary for use in Xilinx 6.2i ISE. Opening Project Navigator project file, it is possible to import System Generator design into the Project Navigator, and from there, it can be synthesized, simulated, and implemented in the Xilinx 6.2i software tools environment.

Configuration ".bit" file is then downloaded in FPGA using for example the Parallel cable IV and Xilinx download program iMPACT.

2.3. Propagation phase

The sensorial outputs from ADUC512 microcontroller represented on 8 bits parallel format and sampled at 10 ms are loaded in the neural network implemented FPGA. For testing the developed method we have used a sensorial system for an artificial hand composed from:

- Data glove as signal source related with fingers and hand position
- Force sensing resistors (FSR) to detect contact with an object and the force being exerted
- Data acquisition system made up with ADUC512 microconverter

Analog signals from FSR are converted in digital signals by microconverter. Also it receives serial data from glove and output both signals time multiplexed in 8 bits parallel format. More precisely outputs 7 bytes of information about 5 fingers position and 2 about hand position (pitch and roll), followed by 6 bytes of information supplied by 6 touch-pressure sensors located on the fingertips as well as on the palm. The FPGA module serves as implementation framework for neural networks. It receives data from data acquisition system in 8 bits parallel format and outputs the recognized posture number.

The recognized posture can serve as feedback in a control system, or can be transmitted via a signal generator to the peripheral nervous system for the persons with loss of sensory nerve function, or can be used for teleoperating a robotic hand.

3. NEURAL NETWORK DESIGN

As mentioned above, with this method neural networks could be realized using the specific modules created with blocks from Xilinx Blockset.

The neural network model presented in Figure 2 has inputs (X_i), plus all the outputs (Y) from other neurons, feeding into a summing junction (Σ) whose output feeds an activation function (A). W_{Xi} represents the matrix of weights for inputs of neurons and W_Y the matrix of weights for neuron outputs connected to inputs.

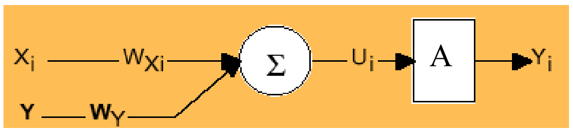

Fig. 2 Neural Network Model

Figure 3 shows the neural network model in Simulink. The main element of neuron is the multiply-accumulate (MAC) block. This block could be implemented efficiently using existing dedicated multipliers in Virtex II, Virtex II Pro or Spartan III FPGAs. For example XC2V250 (a Virtex II FPGA) has 24 dedicated 18 bits MAC blocks. They can be implemented efficiently even in other FPGAs without dedicated MAC blocks, using Xilinx LogiCORE Generator.

Figure 4 presents a MAC block realized using blocks from Xilinx Blockset library.

Control logic block determines neural network architecture. For example determines number of neurons and the correspondence between inputs and weights. For simplicity we have considered that all neurons from a layer are connected to all neurons outputs from previous layer. In other cases the not necessary connections could be deactivated setting corresponding weights to zero.

ROM memory is used for storage of neurons inputs weights, and the RAM memory as a data buffer.

Transfer function is implemented using look-up tables.

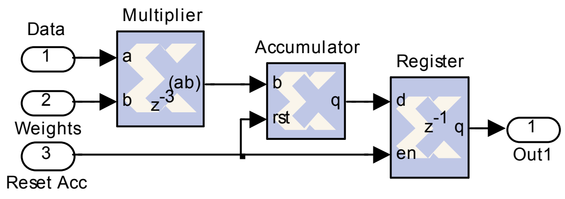

Fig. 4 Multiply-accumulate block

The resources consumed by a very simple network with one layer of 7 neurons are presented in Table 1. Between parenthesis are shown resources used by the 16 bits multiply-accumulate block.

Used resources	MAC implemented with	
	VIRTEX-II dedicated multipliers	Xilinx LogiCORE multipliers
Slices	55 (29)	89 (63)
Flip Flops	56 (39)	123 (106)
Block RAMs	0	0
Look-up tables	66 (17)	170 (121)
Dedicated multipliers	1 (1)	0
% from a 50.000 gates Spartan-II	--	11,58 %
% from a 250.000 gates Virtex-II	3.58 %	5,79
% from a 1.000.000 gates Virtex-II	1,07%	1,73%

Tab.1 Resource consumed by a simple network

The multiply-accumulate operation is the bottleneck of ANNs FPGA implementation, because require a large amount of logic blocks.

The resources depend in a grate measure from the number of bits used to represent data and weights. The shown data are for 8 bits representation of data and 12 bits used for weights.

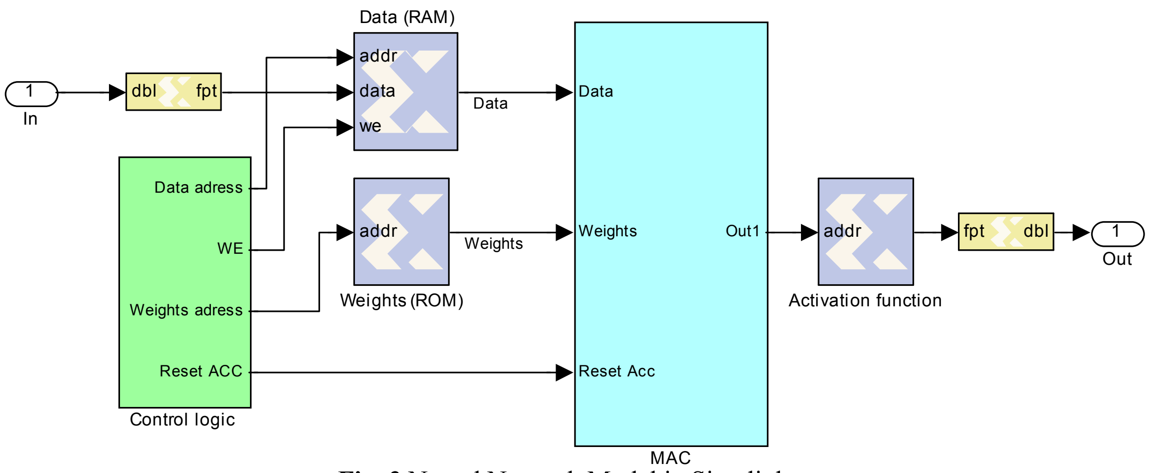

Fig. 3 Neural Network Model in Simulink

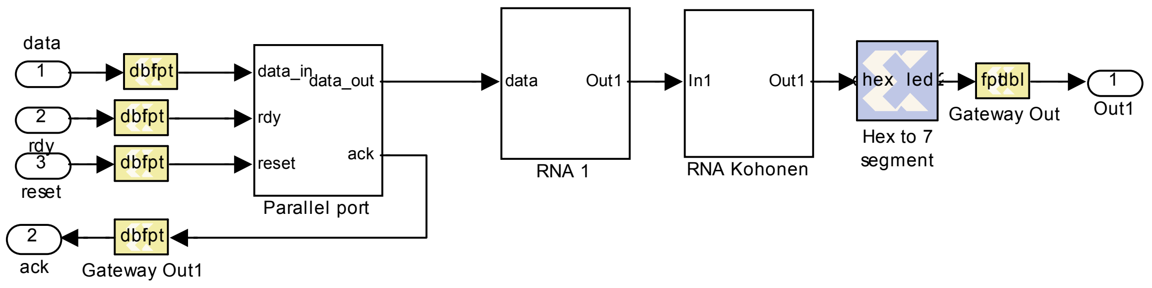

Fig. 5 Gesture recognition network

Definition of system elements is made automatically using variables that are taken from Matlab workspace. In this way dimension of the memories, registers, counters, as constants and number of bits/word are automatically loaded in Simulink representation of the ANN after the simulation of the neural network in Matlab.

4. RESULTS

As presented earlier the method was developed for easy implementation of neural network used in smart sensors. The chosen application for testing the method was static hand gesture recognition using a data glove equipped with optical fiber flex sensors. Figure 4 presents the implemented configuration for gesture recognition.

First block is a parallel port implementation and ensure the correct data transfer between data acquisition system and gesture recognition neural network.

RNA1 is Feed-Forward network that can be trained in many different ways but one of the most common methods is gradient based learning using back propagation. Other very used training method is Hebbian learning rule. We have tested both of them with good results. RNA 1 is used for input data preprocessing and is build from one layer of N1 neurons, where N1 represent the number of sensorial inputs.

The second network used for classification task is a simple competitive network with one layer of N2 neurons, one for each of N2 gesture to be recognized.

Last block is a BCD to 7 segment decoder and it displays the recognized gesture number.

5. CONCLUSIONS

This paper has presented a new method for the implementation of neural networks in FPGAs.

The main contribution of this work is the creation of the framework that permit rapid development of smart sensors with learning capability and adaptive behavior.

An other contribution is the creation of neural network specific modules such as MAC units, activation function.

The proposed method permit to easily adapt the number of neurons per layer, the weight of each input and the activation function.

A testbench was developed for application and it permit to implement different types of neural network with different kinds of architecture.

Future work will focus on development of other neural nework specific modules, optimization of implemented modules, and implementation of on-chip learning capability.

6. REFERENCES

[1] Jihan Zhu and Peter Sutton: FPGA Implementations of Neural Networks – a Survey of a Decade of Progress, 2003.

[2] H. Ossoinig, E. Reisinger, Ch. Steger, R. Weiss: Design and FPGA-Implementation of a Neural Network. Proceedings of the 7th International Conference on Signal Processing Applications & Technology, pages 939-943, Boston, USA, October 1996.

[3] Dr. M. Turhan Taner: Kohonen's Self Organizing Networks With "Conscience", Rock Solid Images, November 1997.

[4] K. Boehm, W. Broll, M. Sokolewicz: Dynamic Gesture Recognition Using Neural Networks; A Fundament for Advanced Interaction Construction, SPIE Conference Electronic Imaging Science & Technology, San Jose California, USA, Feb. 1994.

[5] Rolf F. Molz, Paulo M. Engel, Fernando G. Moraes, Lionel Torres, Michel Robert: Codesign of Fully Parallel Neural Network for a Classification Problem, International Conference on Information Systems, Analysis and Synthesis, Orlando, USA, 2000.

[6] R. Gadea, J. Cerda, F. Ballester, A. Mocholi: "Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation", Proceedings of the 13th International Symposium on System Synthesis (ISSS'00), pp 225-230, Madrid, Spain, 2000..