Hardware Implemented Neural Networks used for Hand Gestures Recognition

S. Oniga*, I.Orha**
*University of Debrecen/Informatics Systems and Networks, Debrecen, Hungary, oniga.istvan@inf.unideb.hu
** North University of Baia Mare/Electronic and Computer Engineering, Baia Mare, Romania, ioan.orha@ubm.ro

***Abstract*— This paper presents hardware implementation of Artificial Neural Networks (ANN) that are used for human hand's natural gestures recognition. Main goal of this project is to implement a recognition system that recognizes data gathered from various sensors placed on a bracelet. This easy to use interface can be used even by elderly or impaired people to control electronic/electric devices only with hand gestures.**

Keywords: Neural networks, FPGA, Gesture recognition.

I. INTRODUCTION

Today's user interfaces it seems to be not enough natural, because of the lack of adaptation and learning capabilities. That's why the main goal of this project is to implement an easy to use interface that can be used even by elderly or impaired people. The data gathered from various sensors placed on a bracelet is sent in raw form, via radio waves to the control unit based on field programmable gate arrays (FPGA).

The use of neural networks to add learning and adaptive behavior is essential and the FPGA implementation is an easy an attractive way for hardware implementation.

This paper presents the successful implementation in FPGAs of some FF-BP neural networks with neuron parallelism used in pattern recognition tasks.

The network design was carried out using the System Generator tool for Simulink, developed by Xilinx Inc., which is also used to generate the hardware Description Language (HDL) code for the network.

The presented interface can be used in medical applications or in ambient assisted applications. Among possible applications are:

- smart devices with embedded and hidden intelligence, for the prosthetic, automotive, "domotic" and automation field where the trend is to produce easy-to-use devices
- intelligent computer peripherals enabling people with any kind of handicap to use computer and communicate,

any kind of industrial or domestic device with learning and adaptive capabilities.

II. GESTURE BASED CONTROL SYSTEM

This paper presents a part of a larger project that offers a solution under the form of a universal HMI: a bracelet. The gestures based control system is composed by two subsystems that communicated via radio waves (Figure 1).

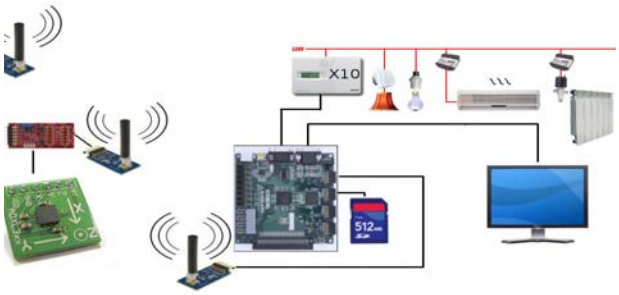


Figure 1. Gesture based control system

The first subsystem is a bracelet (Fig. 2) that captures the movement of the hand using accelerometers. The accelerometer chosen for implementation is ADXL345 from Analog Devices due to its 13 bits resolution and the ability to measure acceleration up to 16 g. These data is transmitted to the second subsystem via radio waves (communication range easily covers a house). The data are sent in raw form, data processing being done on the second subsystem. The bracelet is equipped with an Atmega168 microcontroller that receives data from the accelerometer and relays these data to the control box subsystem using an At86RF212 radio transceiver.

The second subsystem is the control box on which the data processing takes places. The data received by radio waves are processed using artificial neural networks in order to recognize hand gestures. These hand movements allow the user to navigate into a menu displayed onto a computer screen or a TV. The menu entries are real world actions that can control electronic devices directly through relays or through X10 network (existing power lines). These menu entries are highly configurable and easy to change because are stored on a SD card.

The control box is built around a Spartan 3E FPGA board (Nexys 2 from Digilent Inc). This assures easy upgrade and a cheap yet powerful platform for data processing and ANN hardware implementation. Also it can be successfully used as glue logic between the different types of input and output protocols.

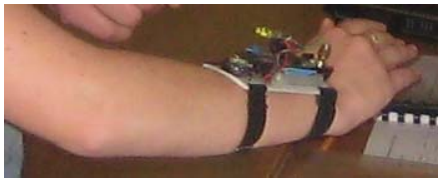


Figure 2. The bracelet

The FPGA board has a VGA output that is used for displaying the menu on any VGA compatible monitor or using an external converter even on a TV set. SPI protocol is used to access the SD card. The FPGA board has a build in serial port that is used for communicating with an X10 controller (CM11 from Marmitek). The controller transmits commands through power lines to any X10 compatible receivers in order to control household appliances.

III. HAND GESTURE

The number of gestures that need to be recognized depends on the field of the application:

- To manipulate in a 3D virtual space, 3 gestures are normally sufficient: grab, start, stop,
- In more complex systems, 5 to 10 gestures need to be recognized,
- In order to recognize the sign language 26 to 51 gestures must be taken into consideration.

The navigation in the menu is done with different hand movements: front, back, left, right. These movements can be replaces by inclining the bracelet, especially for those impaired. The purpose of this work is to create a system capable of recognizing 5 gestures used to navigate through a menu which is displayed on a screen.

- horizontal = idle
- front (up) = enter in submenu or activate the attached command
- back (down) = exits from current menu and go back to the parent menu
- left, right = navigate within the menu left/right

The menu consists of different entries that can have an external action attached, meaning that different electronic devices can be controlled by accessing the menu and select the appropriate action, eg: the light is turn on by selecting the TURN LIGHT ON option from the menu. The front movement of the hand enters a submenu of the current menu or activates an external action (if attached); the back movement exits from a menu to the parent menu, the left and right movements navigate within the menu left/right.

Sample of data representing different types of gesture provided by the bracelet are shown in Table 1.

TABLE I.
DATA SAMPLES

	Horizontal	Right	Left	Front	Back
Sensor 1	26	3600	12784	15	16383
Sensor 2	13	16367	38	3600	12784
Sensor 3	47	4118	12198	26	16215
Sensor 4	23	16354	67	4183	12246
Sensor 5	6178	9203	38	3612	12876

IV. GESTURE RECOGNITION METHODS

From amongst the known methods of recognition, the solution that utilizes neural network was chosen, because it offers the best results regarding the functioning precision.

There are several methods used to recognize gestures, including angle filtering, statistical methods, principal component analysis, neural networks, testing pattern proximity, etc. These methods can be divided into two large groups: those that can learn to recognize simple gestures and those that cannot.

Well known for their success in the recognition of patterns, the neural networks are being used in numerous gesture recognition systems. There are multiple advantages that the neural networks have to offer. Training based on examples, recognizing gestures even if noise is present or the data is incomplete, and last but not least the generalization. This last characteristic plays a crucial role in the systems performance due to the fact that not even the same user will reproduce its gestures accurately.

It is difficult to compare the recognition methods because the implemented systems do not work with the same gestures and were not tested in the same way, etc. Even so, it is obvious that the recognition method that uses neural networks offers the best recognition rate.

V. NEURAL NETWORK DESIGN

The neural networks used to recognize gestures are different depending on the gestures that had to be recognized

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

The design and implementation of the hardware model is made using System Generator and Xilinx ISE.

A. Neural Network Model in Simulink

In order to choose a NN capable to recognize as accurately as possible the gestures of a hand, several types of NN were simulated, as well as combination of these. Finaly we have chosen a feedforward with 2 layers architecture:

- First layer has 20 neurons and sigmoid activation function
- The second layer has 5 neurons purelin activation function.

The software (Simulink) model of the neural network designed, trained and simulated using Neural Network Toolbox is presented in Fig. 3.

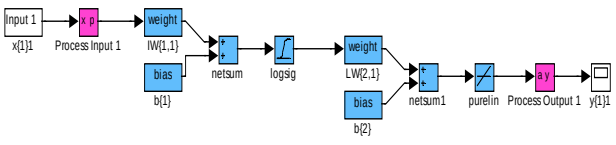


Figure 3. Software model of the ANN used to recognize hand gestures

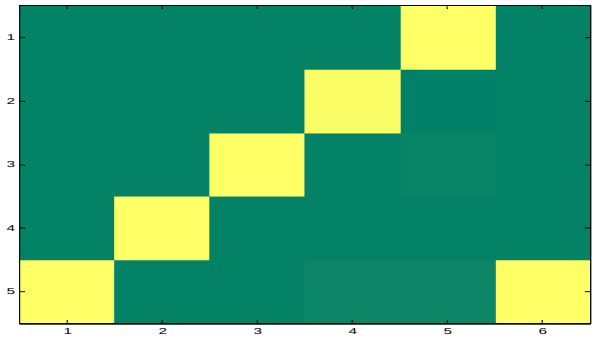


Figure 4. Simulation results of the neural network

Training of the NN was made using TRAINLM Levenberg-Marquardt backpropagation training function and a data set consisting in 1500x5 vectors.

Fig. 4 shows the simulation results of the NN used for gestures recognition. Simulation of the network was made using 6x5 vectors data set. It could be seen that for each applied set of vectors (from 1 to 6 on horizontal) only one neuron output is active. Number of neuron represents the number of the gestures that was recognized. For the first set, gesture 5 for the second gesture 4, and so on. For the last set of vectors once again gesture number 5 was recognized.

B. Hardware Implementation

The designed neuron model in System Generator (Fig. 5) is equivalent with the well know McCulloch-Pitts model and implement (1).

$$y(x) = f(a - \theta) = f\left(\sum_{i=1}^{N} w_i x_i - \theta\right). \quad (1)$$

The model was implemented using blocks from Xilinx Blockset library and user created blocks. Among the blocks created can be mentioned:

- Weights memory (ROM)
- MAC block (weighted sum)
- Bias block
- Activation function block.

The activation function block is common to all the neurons and is implemented using look-up tables.

Using this model for a neuron we have implemented in hardware the model presented in Fig. 3. The complete model of the ANN used for gesture recognition is presented in Fig. 6.

The A NN is composed of following blocks:

- Control blocks
- Input layer
- Preprocessing (normalization)
- Hidden layer (20 neurons)
- Activation function block,
- Output layer (5 neurons)

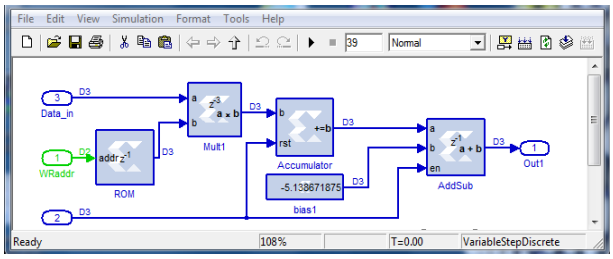


Figure 5. The hardware implemented neuron model

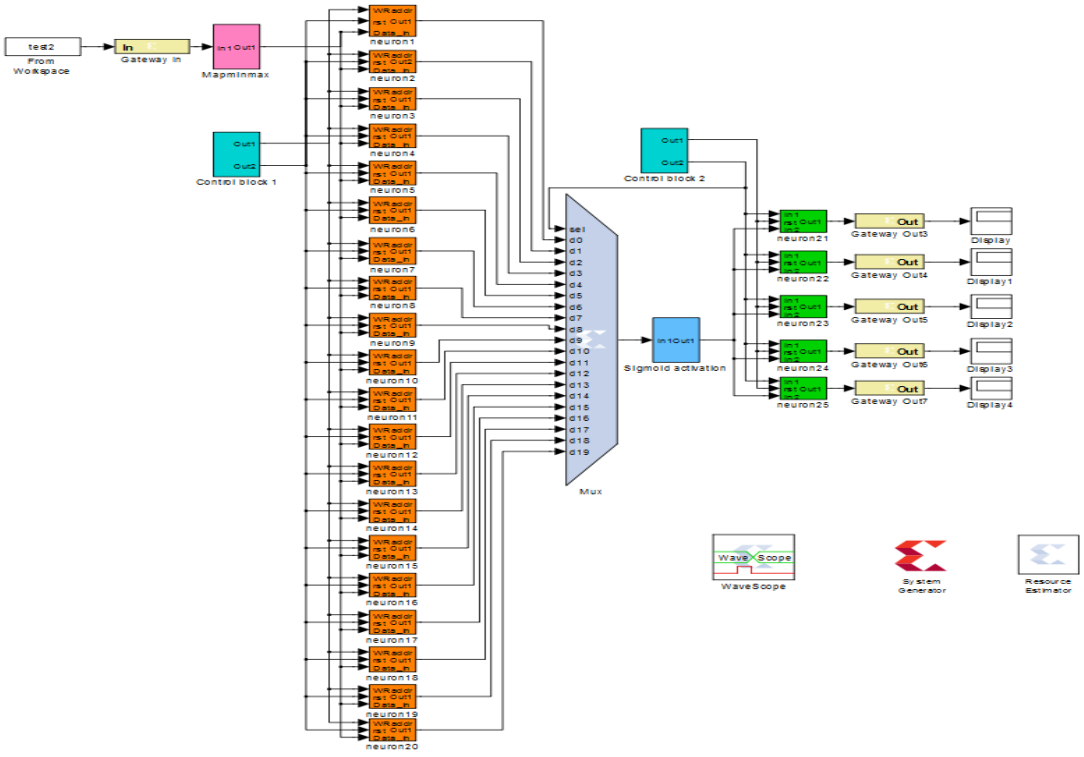
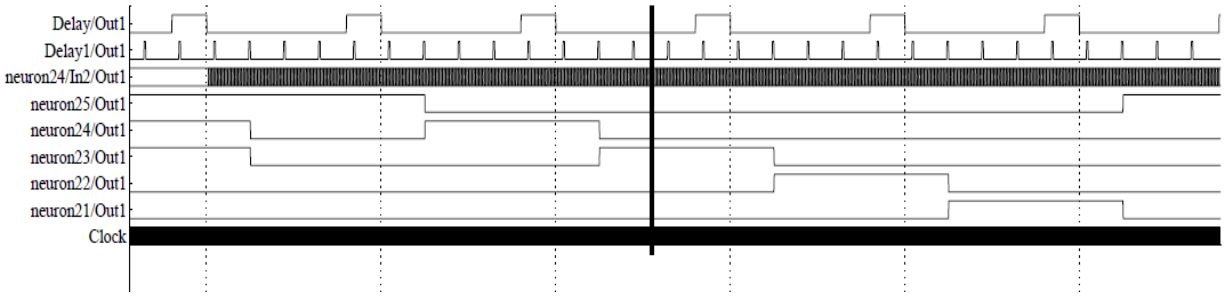
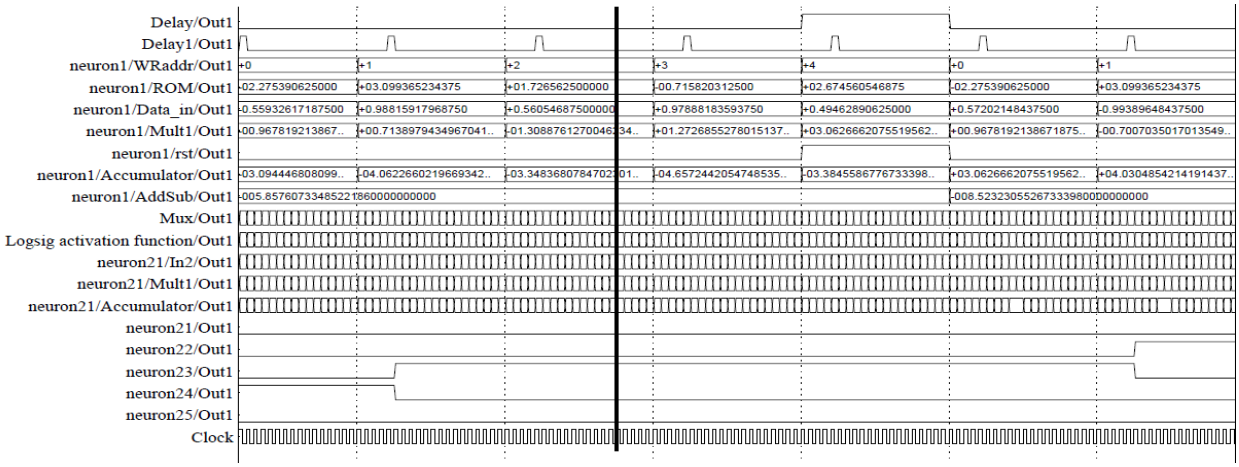


Figure 6. Neural network model in System Generator



a) Outputs of the neurons from the output layer



b) Detail on the computation for the gesture No. 3

Figure 7. Simulation of the hardware model of the ANN

VI. RESULTS

A. Simulations

The results obtained simulating the hardware model of ANN are presented in Fig. 7. Fig. 7a. presents the outputs of neurons from second (output) layer. For a given input only one output is active, those corresponding to the recognized gesture. Fig. 7b. presents a details on the computation for gesture nr. 3.

B. Resources

The estimated resources used for the implementation of a neuron using full precision of the multipliers and accumulators are presented in Fig. 8.

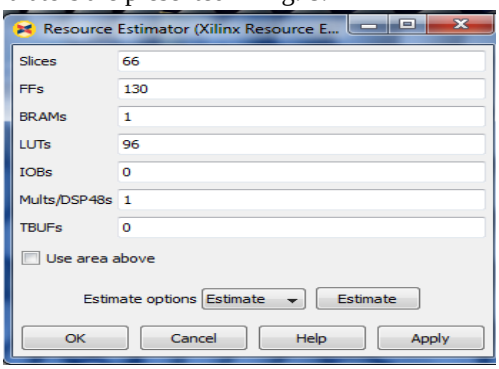


Figure 8. Resource estimation for implementation of a neuron

The resources depend in a grate measure on the number of bits used to represent data and weights. The shown data are for 14 bits representation of data and 16 bits used for weights. The multiply-accumulate operation is the bottleneck of NNs implementation in FPGA, because requires a large amount of logic blocks. Using various techniques presented in [8] regarding the reduction of number of bits that the multiplier and the accumulator use, the resource needed for the implementation of a neuron could be as low as 12-22 slices.

The resources needed to implement the 25 neurons FF-BP ANN with neuron parallelism using full precision in a Spartan 3E XC3S500E are presented in the Fig. 9.

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	4,218	9,312	45%	
Number of 4 input LUTs	5,384	9,312	57%	
Number of occupied Slices	3,523	4,656	75%	
Number of Slices containing only related logic	3,523	3,523	100%	
Number of Slices containing unrelated logic	0	3,523	0%	
Total Number of 4 input LUTs	6,123	9,312	65%	
Number used as logic	5,372			
Number used as a route-thru	739			
Number used as Shift registers	12			
Number of bonded IOBs	20	232	8%	
Number of BUFGMUXs	1	24	4%	
Number of MULT18X18SIOs	20	20	100%	
Average Fanout of Non-Clock Nets	2.78			

Figure 9.

In a bigger device having some hundreds to 2128 DSP blocks and BRAMs we can implement hundreds and even some thousands of neurons with neuron parallelism

VII. CONCLUSIONS

This paper presented the successful implementation of some simple FF-BP neural networks with neuron parallelism used in model classification tasks.

The implemented network has as many processing blocks as neurons are within the network, and only one activation function block per layer.

This network correctly recognizes all the training and test vectors supplied.

The maximum number of competitive neurons that can be implemented into a Spartan 3E device could be from some tens to some hundreds.

For larger networks and higher working frequencies, lager FPGA devices with higher working frequencies can be used.

For example, the Virtex 6 family contains over 2000 DSP blocks and over 74000 slices, which would allow the implementation of over 2000 neurons.

REFERENCES

[1] L. Marchese, "Digital neural network based smart devices", Project proposal, FP7 European Research Program, http://www.synaptics.org, December 2006, unpublished.

[2] S. Oniga, A. Tisan, C. Lung, A. Buchman, I. Orha, Adaptive hardware-software co-design platform for fast prototyping of embedded systems, Optimization of Electrical and Electronic Equipment (OPTIM), 2010 12th International Conference on, 2010 , Brasov, Romania, pp: 1004 – 1009.

[3] S. Oniga, A. Tisan, D. Mic, C. Lung, I. Orha, A. Buchman, A. Vida-Ratiu, "FPGA Implementation of Feed-Forward Neural Networks for Smart Devices Development", Proceedings of the ISSCS 2009 International Symposium on Signals, Circuits and Systems, July 9-10, 2009, Iasi, Romania, pp. 401-404

[4] A. Tisan, S. Oniga, A. Buchman, C. Gavrincea, "Architecture and Algorithms for Synthesizable Neural Networks with On-Chip Learning", International Symposium on Signals, Circuits and Systems, ISSCS 2007, July 12-13, 2007, Iasi, Romania, vol.1, pp 265 – 268.

[5] J. Torresen, Sh.Tomita, „A Review of Implementation of Backpropagation Neural Networks", Chapter 2 in the book by N. Sundararajan and P. Saratchandran (editors): Parallel Architectures for Artificial Neural Networks, IEEE CS Press, 1998. ISBN 0-8186-8399-6

[6] J. Zhu, P. Sutton, „FPGA Implementations of Neural Networks – a Survey of a Decade of Progress", Proceedings of 13th International Conference on Field Programmable Logic and Applications (FPL 2003), Lisbon, Sep 2003

[7] S. Dragici, "On the capabilities of neural networks using limited precision weights", Neural networks, Volume 15 , Issue 3, April 2002 pp. 395-414

[8] S. Oniga, A. Tisan, D. Mic, A. Buchman, A. Vida-Ratiu, "Optimizing FPGA implementation of feed-forward neural networks",, Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment OPTIM 2008, 2008, Brasov, Romania, May 22-23, pp. 31-36.