Implementation of Analog Neural Networks with Labview

Liljana Docheva [1] and Aleksander Bekiarski [2]

Abstract – **The ability of analog neural networks to process different types of information is well known, but for the real application such as image processing or recognition it is necessary to use modelling and analyzing of the methods chosen for a concrete image processing or recognition. The modelling and analysis included also the specific parameters of the electronic elements of the analog neural networks.**

In this article it is proposed to model and analyze a multi-layered analog neural network with LABVIEW, applying Back Propagation algorithm in learning process. These types of neural network are capable to perform just about any linear or non-linear computation, and can approximate any reasonable function arbitrarily well. LABVIEW graphical programming environment and capability in building analog artificial neural networks are discussed.

The result of the modelling and simulations of the proposed multi-layered analog neural network are shown and in the conclusion are made some remarks of the advantages and the proposition for the future investigations in this direction.

Keywords – Analog neural networks, LABVIEW.

I. INTRODUCTION

Analog neural networks have its wide application cause their high speed, low power consumption and compact implementation. But variation in the size of discrete transistors and the local mobility will cause random parameter variation. An increase in the precision of any component will lead to increase of its area.

Limited accuracy and nonlinear behavior, typical of analog neural networks, don't reduce their application because utilization of appropriate training algorithm (back propagation for example) decreases their influence to a certain degree.

Simulations made in LABVIEW will help to design and test analog neural network with aim to investigate influence of some analog neural network parameters onto its recognition ability.

LABVIEW is graphical programming software, used usually for data acquisition and control. In some articles have been successfully used LABVIEW for building neural networks [1,2] because one has the ability to develop data flow that are highly parallel in structure. Therefore LABVIEW seems to be a very effective approach for building and investigating neural networks.

In addition is very easy during building process to make that the parameters of real components to take part in neural networks. On this way the components of an analog neural network will be close to real ones.

In this article multi-layer feed-forward analog artificial neural network is investigated. The training algorithm applied to this network is back propagation. The aim is to discuss the LABVIEW graphical programming environment, its features and capability in building analog artificial neural networks.

[1]Liljana Docheva is with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, E-mail: docheva@tu-sofia.bg.

[2]Aleksander Bekiarski is with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria,aabbv@tu-sofia.b.

II. ANALOG NEURAL NETWORK

Many investigations in the implementation of analog neural networks field are known [3, 4, 5, 6]. In this article a VLSI implementation of an analog neural network, depicted in [5] is chosen for building with LABVIEW. The synapse chip consists of a number of MOS resistive circuit multiplier. On the base of the neuron and synapse equations described in [5], equations about analog parameter variation over analog neural network behavior have been worked out [7,8]. These equations will take place in LABVIEW implementation discussed in this article. On this way we have investigated parameter variation influence over behavior of a multi-layer feed-forward analog artificial neural network implemented with LABVIEW. The training algorithm applied to this network is back propagation. The aim isn't to decide some complex problem. We chose a simple one (AND problem) in the beginning and more difficult on the later stage of our research. On this way it will be shown how parameter variation and complexity of the task affects over behavior of an analog neural network.

III. APPLICATION OF LABVIEW FOR AN ANALOG NEURAL NETWORK IMPLEMENTATION

A. Block scheme

Graphical programming environment LABVIEW allows construction and investigations of complex systems analysis including neural networks. There are different ways to realize neural network with LABVIEW. We have clung to architecture of the back propagation network. This implementation with LABVIEW is easy to explain with the block diagram depicted on Fig. 1. Feed forward signal is accomplished from 1 to 5 numbered blocks. The blocks with numbers from 6 to 9 serve as purpose to back propagation error signal. Function of every one of them is explained bellow.

Block 1 is a data base. One includes input patterns that are applied to neural network. In our case the input pattern is a binary vector.

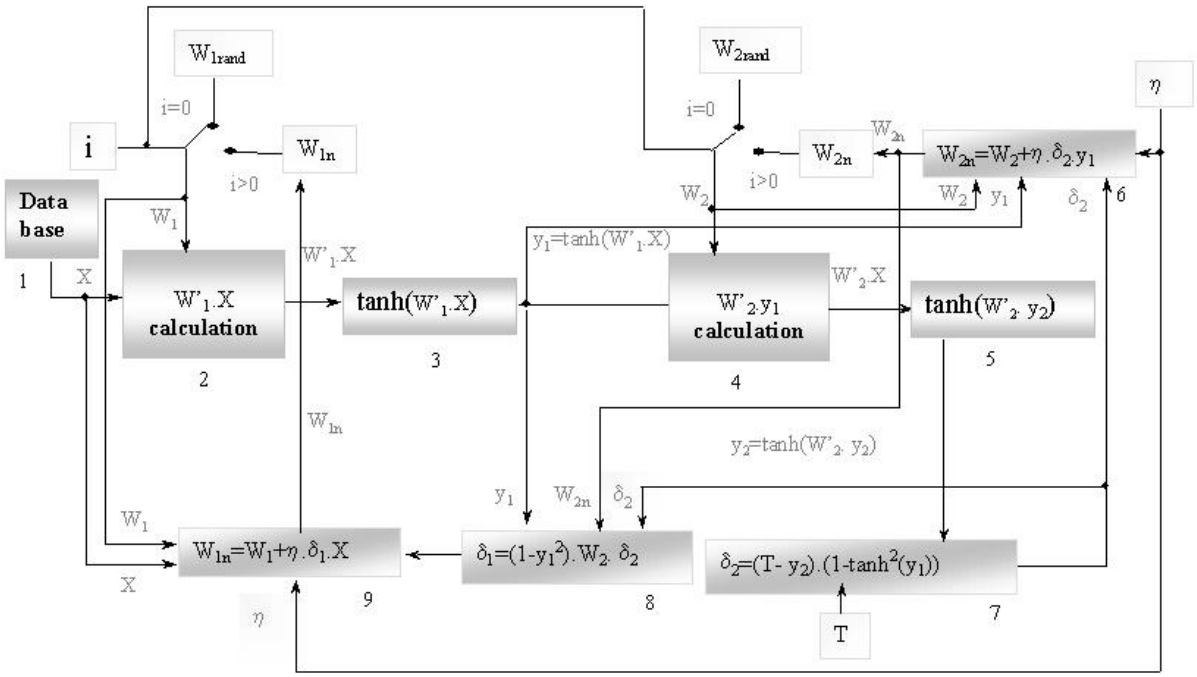

Fig. 1. The analog neural network LABVIEW implementation block diagram

Blocks 2 and 4: The aim of these blocks is calculating the weighted sum for the first layer and for the second layer correspondingly. First at all it mast to decide whether input vector will be multiplied by weight matrix of random numbers (first initialization - $\mathbf{W_{1rand}}$ or $\mathbf{W_{2rand}}$), or matrix of updated weights ($\mathbf{W_{1n}}$ or $\mathbf{W_{2n}}$). For this aim we use the number of iteration **i**. When **i** is equal to **0** the weight mast to initialize for first time with random numbers. For the all next iterations (**i>0**) input vector will be multiplied by the updated weight matrix.

The weighted sum takes part in output voltage of neuron calculating (blocks 3 and 5). Ones calculate activation function (tanh(.) in our case). On the outputs of these blocks are obtained $\mathbf{y_1}$ and $\mathbf{y_2}$. In this paper is considered analog neural network therefore we will denote **w** as $\mathbf{u_w}$ and **y** as $\mathbf{u_y}$ (Eq. 1). For the output layer it can be written [8]:

$$u_{yk}^{2}(t)=\left(\chi_k^2+\Delta\chi_k^2\right)\tanh\left(\left(\xi_k^2+\Delta\xi_k^2\right)\sum_{j=1}^{N}U_{w_{kj}}u_{yj}^{1}(t)\right) \quad (1)$$

where

u_{yk}^{l} is the neuron output voltage;

k – number of neuron,

l - neural network layer number,

χ and ξ - parameters of analog components constructing neuron.

Δ- symbol of variable variation.

After signal is fed forward the error back propagation follows. The firs step is to calculate the equivalent error of the k[th] neuron of the 2[th] layer δ^2_k (Eq. 2) and is accomplished of block 7:

$$\delta_k^2(t)=\left[d_k-\left(\chi_k^2+\Delta\chi_k^2\right)\tanh\left(\left(\xi_k^2+\Delta\xi_k^2\right)\sum_{j=1}^{N}U_{w_{kj}}u_{yj}^{1}(t)\right)\right]. \quad (2)$$
$$.\left(\chi_k^2+\Delta\chi_k^2\right)\left(\xi_k^2+\Delta\xi_k^2\right)\left(1-\tanh^2\left(\left(\xi_k^2+\Delta\xi_k^2\right)\sum_{j=1}^{N}U_{w_{kj}}u_{yj}^{1}(t)\right)\right).$$

Then in block 6 the weight update for the 2th layer is calculated (Eq. 3).

$$U_{wyk}^{2}(t+1)=U_{wyk}^{2}(t)+(\eta.\delta_k^2(t).u_{yj}^{1}(t)) \quad (3)$$

Calculating the equivalent error of the 1[th] layer (block 8) is computed (Eq. 4): on the base of equivalent error of the 2[th] layer:

$$\delta_j^1(t)=\left(\chi_j^1+\Delta\chi_j^1\right)\left(\xi_j^1+\Delta\xi_j^1\right)\left(1-\tanh^2\left(\left(\xi_j^1+\Delta\xi_j^1\right)\sum_{j=1}^{M}U_{w_{ij}}(t)u_{yz}(t)\right)\right)\sum_{j=1}^{K}U_{w_{kj}}(t)\delta_k^2(t), \quad (4)$$

The aim of block 9 is the weight update for the 1th layer (Eq. 5).

$$U_{wyk}^{1}(t+1)=U_{wyk}^{1}(t)+(\eta.\delta_k^1(t).u_{yz}(t)) \quad (5)$$

C. Implementation of an analog neural network with graphical programming software LABVIEW

It isn't difficult to implement the bocks mentioned above with *g*raphical programming software LABVIEW. Before blocks 2 and 4 implementation, decision whether input vector will be multiplied by weight matrix of random numbers mast be made. This is presented on Fig. 2 for first layer. The design for second layer is similar. When the number of iteration **i** is equal to **0** the weight mast to initialize for first time with random numbers. For the all next iterations (**i>0**) input vector will be multiplied by the updated weight matrix.

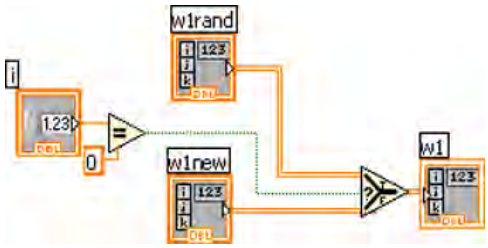

Fig. 2. The choice of weight matrix for the weighted sum of first layer.

Implementation of block 2 is depicted on Fig. 3.

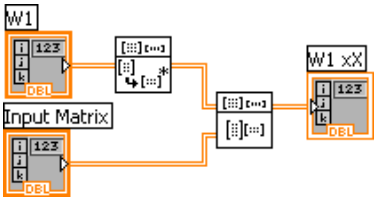

Fig. 3. Implementation of block 2.

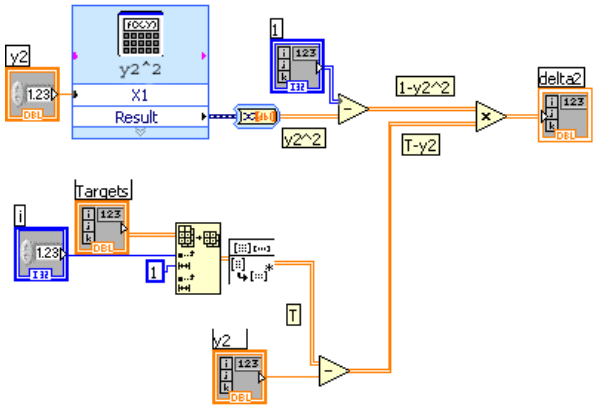

Fig. 4. Implementation of the equivalent error calculation for the second layer

For this aim are used two virtual instruments: **Transposes Input Matrix** and **AxB**. After weight matrix transposing the multiplication of input vector (pattern) and weight matrix is performed with virtual instrument **AxB**. The block 4 is

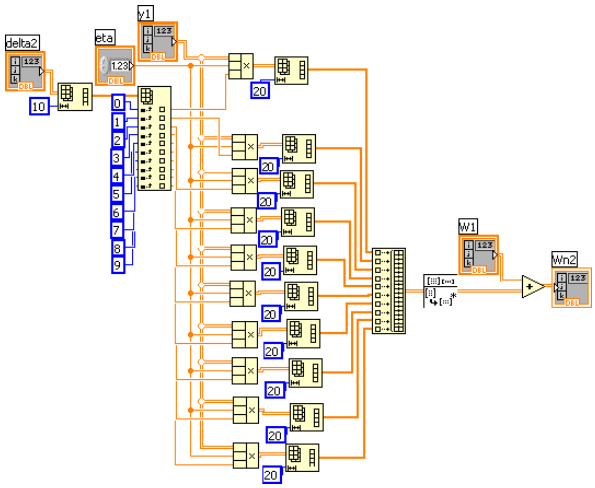

Fig. 5. Weight update for the 2th layer.

implemented on same way.

As it can be seen from Eq. 1 in calculation for output neuron voltage parameters of the real elements of analog neural network take part. The activation function (blocks 3 and 5) is realized by **Formula** virtual instrument.

Figure 4 depicts a block 7 implementation. One is done following Eq. 2.

The block 6 - the weight update for the 2th layer (Eq. 3) is given in Fig 5. After delta for second layer is calculated new weights can be computed. To obtain this operation mast to be used the follow virtual instruments: **Reshape Array** - Changes the dimension of an array, **Index Array** - Returns

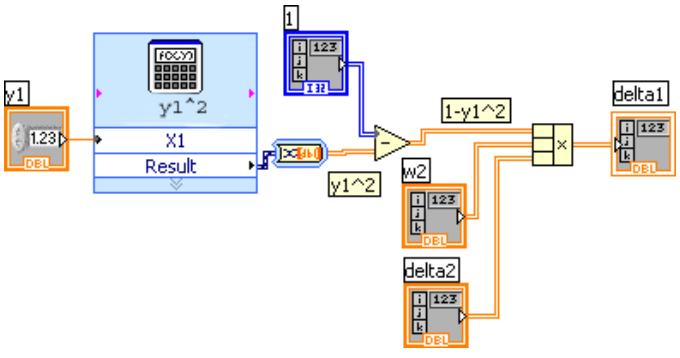

Fig. 6. Implementation of the equivalent error calculation for the first layer

the element of 1-dimension array, **Build Array** - Concatenates multiple vectors to an 2-dimensional array.

After operations of Eq. 3 are accomplished in an output of he block the updated weights are obtained.

Figure 6 depicts a block 8 implementation that follows Eq. 4 expression. Virtual instrument calculate square of neuron output but in addition voltage parameters of the real elements of analog neural network are included in accordance with Eq. 4.

The block 9 - the weight update for the 1th layer (Eq. 5) - is implemented on similar way to the block 6 implementation.

IV. Results

In order to be sure that analog neural network implemented with *g*raphical programming software LABVIEW works correctly we chose a simple task - AND problem. Fig.7 shows

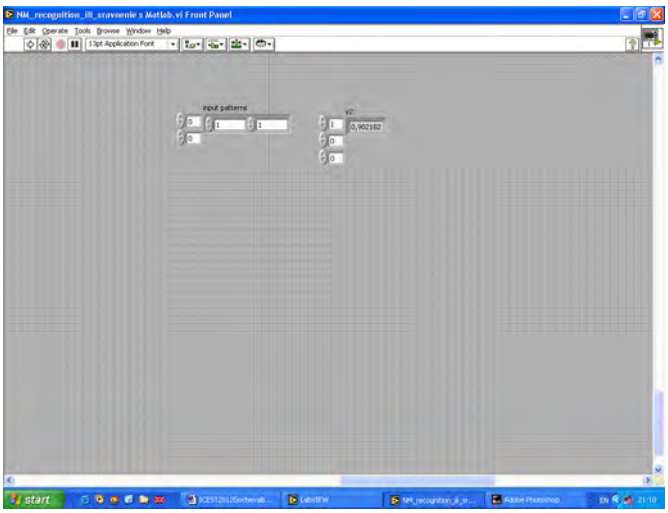

Fig. 7. LABVIEW front panel for one of the applied input patterns

LABVIEW front panel for one of the applied input patterns.

The results for all patterns are given in Table 1. Ones present that network work correctly despite of influence of the analog neural network parameters onto its recognition ability.

TABLE I
THE RESULT OF ANALOG NEURAL NETWORK LEARNING FOR AND PROBLEM

X	Y2
11	0.902
10	0.088
01	0.093
00	0.086

V. Conclusion

In this paper simulations made in LABVIEW are depicted. LABVIEW graphical programming environment, its features and capability in building analog artificial neural networks are discussed. The results present that network work correctly despite of influence of the analog neural network parameters onto its recognition ability.

On the later stage of our research we will examine analog neural network behaviour realized with LABVIEW when a task is more complex

References

[1] J. Fernandez de Canete, S. Gonzalez-Perez, P. del Saz-Orozco, “Artificial Neural Networks for Identification and Control of a Lab-Scale Distillation Column using LABVIEW”, World Academy of Science, Engineering and Technology, 2008.

[2] M.A. Panait, T. Tudorache, " A Simple Neural Network Solar Tracker for Optimizing Conversion Efficiency in Off-Grid Solar Generators", International conference on renewable energies and power quality (ICREPQ'08).2008.

[3] Cyril Prasanna Raj P, S.L. Pinjare, "Design and Analog VLSI Implementation of Neural Network Architecture for Signal Processing", European Journal of Scientific Research , Vol.27 No.2 (2009), pp.199-216,2009.

[4] G. Cauwenhberghs, "An Analog VlSI Recurrent Neural Network Learning a Continuous-Time Trajectory", [1]IEEE Transactions on Neural Networks, vol. 7, N:2, March 1996.

[5] [6] T. Lehman, " Hardware Learning in Analog VLSI Neural Networks ", Ph.D. thesis, Technical University of Denmark, 1994.

[6] S. Draghici, " Neural Networks in analog hardware-design and implementation issues ", Int. J. of Neural Systems, 2000, vol.10, no. 1,pp. 19-42.

[7] L. Docheva, A. Bekiarski, I. Dochev, "Analysis of Analog Neural Network Model with CMOS Multipliers", Radioendineering, vol. 16, N:3, september 2007.

[8] L. Docheva, A. Bekiarski, "Investigationof analog Neural Network used for number recognition in images", International Journal of Neural Networks and applications, 2(1) January-June 2009, pp. 15-18.