Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium, Volume 23, No.1, ISSN 2304-1382
ISBN 978-3-901509-91-9, CDROM version, Ed. B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2012
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2012

IMPLEMENTATION OF THE NEURAL NETWORKS FOR ADAPTIVE CONTROL SYSTEM ON FPGA

KONDRATENKO, Y[uriy] P[anteliyovych] & GORDIENKO, E[vgeniy]

Abstract: *This work deals with the parametric identification of control objects based on application of developed neural network. Special attention paid to the structure of embedded identification system with FPGA (Field Programmable Gate Arrays) realisation.Two kinds of FPGA-based neural networks are synthesized and tested for identification of the non-stationary parameters of caterpillar turn control system with corresponding adjusting of the adaptive regulator parameters. The modeling results confirm the efficiency of the suggested approach and developed neural networks.*
Keywords: *neural network, identification, adaptation, control system, neurochip, FPGA*

1. INTRODUCTION

Over the past 30 years a lot of attention has been paid to adaptive control of objects with unknown parameters. Under this approach Neural Networks (NN) were used for solving control problems in the 1980-s [1]. At the time being, NN are used in control problems as neural emulators and predictors [2]. In practice it is necessary to identify the object parameters that are changed under the influence of internal or external disurbances. The identification problem can be considered as a multivariate function approximation when the response (the values of the object's dynamic parameters) for the input request (the input and output signals' values) is received. Utilizing NN has been proved useful for solving this kind of problems [3].

In this paper, the Field Programmable Gate Arrays (FPGA) application of neural networks for identification and adaptaion of the control system is considered. During the development of a neural network algorithm, accuracy and processing speed are the main effectiveness criteria. Processing speed is the critical factor in complex real time tasks. Using general-purpose processors (GPP) to implement the NN algorithm proves not to be effective from the point of view of productivity-to-cost ratio [5]. The reason is that the NN size can be large enough in case of a complicated task, and the capacities of input signals and weights do not coincide with the GPP standard (32 bits now). In this case neurochip development is used. A number of papers prove FPGA to be one of the most effective applications of Neural Networks. FPGA can be programmed using special programming languages such as Verilog, VHDL, AHDL etc.

The number of FPGA pins and the advanced structure of fast-acting connections allow developing FPGA-based regular fragments of NN and then creating different NN using FPGA cascading [5].

2. NEURAL NETWORK FRAGMENT STRUCTURE

In [5], the utilization of an arbitrary FPGA-based NN is considered and the method of cascade FPGA-based NN development is proposed.

Due to the fact that NN allocation on the chip is difficult because of chip resources' scantiness, the NN has to be divided into cascading parts [5]. A fragment of a feedforward NN with 8 neurons was adopted (Fig. 1). The capacity of input signals and weights vectors is 8; consequently, every neuron has 8 inputs. The method of vertical cascading with an increasing number of neurons in each layer was chosen [6].

Cascading can be implemented using physical (each fragment is allocated on a different chip), virtual (each fragment is looped at one chip), and mixed methods. The given task solution time is the reason to choose one of the cascading methods. The physical cascading solution takes less time than the virtual cascade solution with a ratio roughly equal to the number of physically implemented fragments.

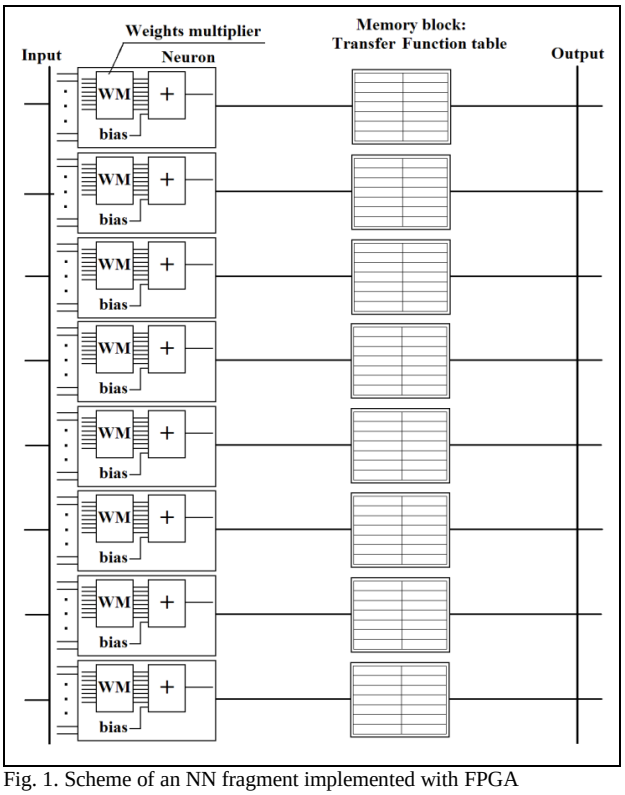

Fig. 1. Scheme of an NN fragment implemented with FPGA

3. NN DEVELOPMENT

The NN fragment was developed using VHDL. The block diagram of the developed neuron is presented on fig. 2.

In [5], a parallel approach to NN development is proposed. This approach provides faster NN utilization but makes NN take more space on a chip, so the serial approach to NN development was chosen. In this case the NN takes only 46% of the chip resources against 86% described in [5]. The described NN fragment takes 46% of the chip Xilinx XCV400E-pq240 resources (305 equivalent logic gates). The fragment consists of the following elements:

1. 8 serial 8-bit multipliers;
2. 8 two-input summators;
3. single-port memory block volume of 256x15 bit for each neuron (8 blocks amount) that implements the transfer function;
4. mode control block.

The chip has 158 single-bit pins. In the current project 143 pins were used: 64 input pins, 64 output pins, 1 synchronizing signal pin and 15 pins for circuit elements and modes control.The clocking rate of the circuit is 90 MHz. The time to calculate the output signal from the input signal is 70 ns.

The developed fragment is destined for:

- parallel input of 8 components of the input vector;
- serial multiplication by weight constants from the internal memory blocks of multipliers (weight multipliers on Fig. 1., *weights_ram* on Fig. 2);
- serial addition of 8 products;
- parallel activation of each sum in each neuron (8 high-order bits of the sum are fed to the memory address port where the table with discrete values of the transfer function is stored).

The neurons don't have any special input for weights' changing. The weights can be changed using the neuron input signals and setting the *WE* (write enable) bit. The multiplied numbers differ only in the speed of change: one of numbers can be changed fast, while the second one can only be changed slowly. It was developed in such a way in order to make the NN invariant under the task. In some tasks it is necessary to change inputs faster than weights (classification, identification), and some tasks require fast changing of weights and slow changing of inputs (solving a system of algebraic equations).

The multiplier constant that is stored in the internal multiplier memory can be changed in 8 time steps. Accordingly, weights can be adjusted in real-time. This functionality is implemented by feeding signals of mode control, choosing neurons and writing permission.

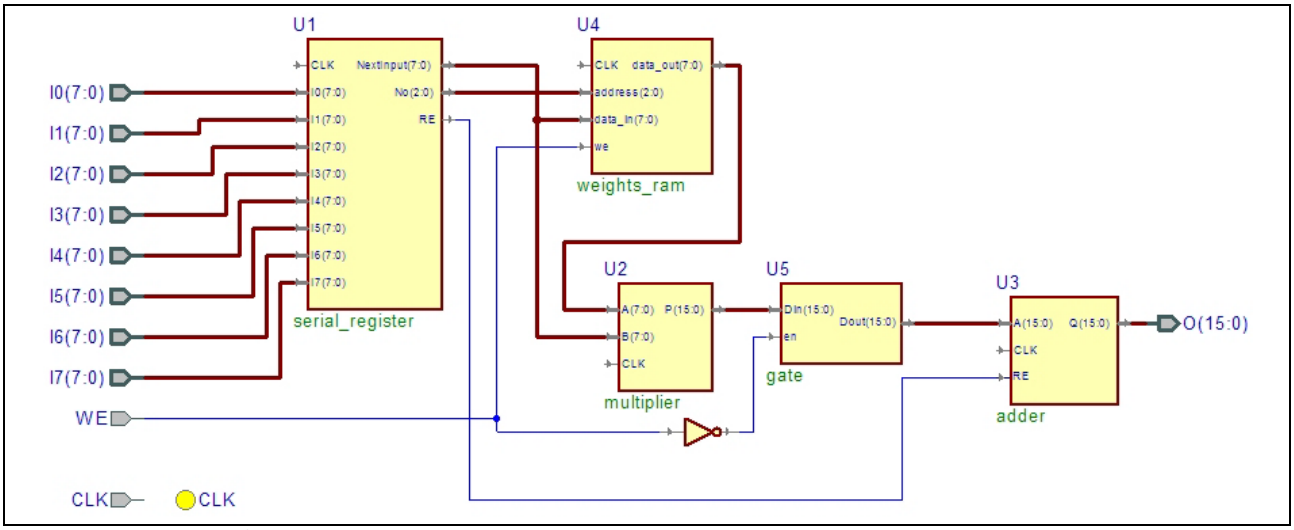

Fig. 2. The block diagram of the neuron

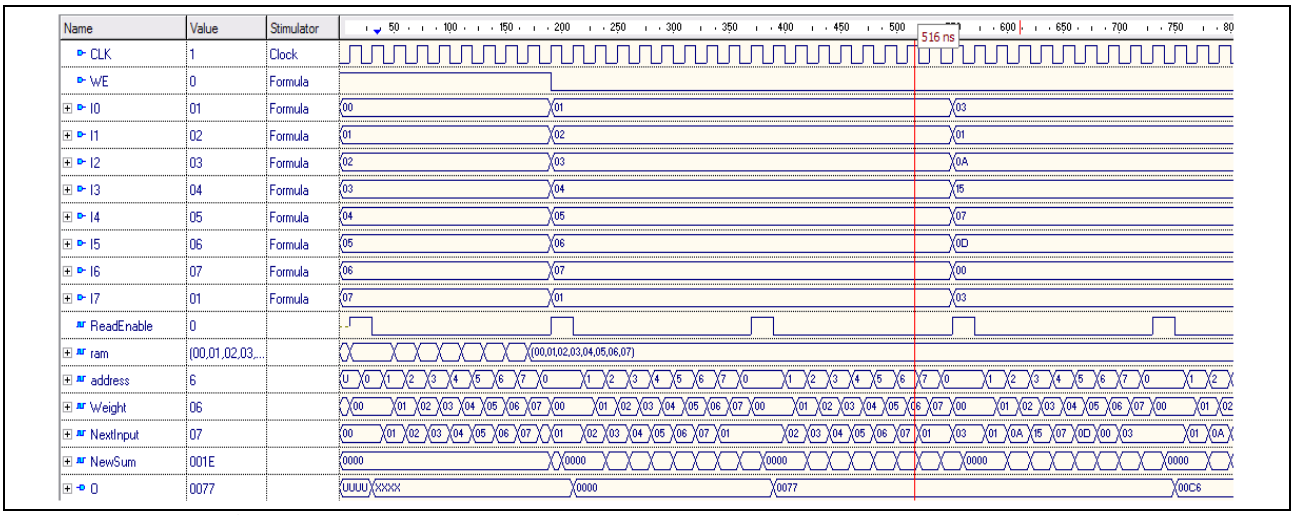

Fig. 3. The time diagram of the neuron modeling results

Data that is stored in the single-port memory block of the transfer function can also be changed. The number of time steps for modification is defined by the number of the modified values. For this task, the mode control and write permission signals are also provided. The values are changed in every of the 8 memory blocks because the transfer function is shared for all the neurons in the layer. A neuron consists of several blocks that are described further.

Serial_register is **a** shifting block that returns every input signal and its number serially with each *CLK* raising. When the last input signal is returned, the *RE* (read enable) output bit is set. It means that all the input signals were processed and the output signal can be read.

Weights_ram is the storage of current neuron weights. It returns the weight for each input by its number (*address* signal). Weights can be changed using the *WE* (write enable) bit. While this bit is set, the weights in the storage are being changed and the neuron output value is not calculated, its value being "0".

Multiplier is a two-input block that returns the product of two input signals by every *CLK* reduce.

Gate is a buffer with a condition: it returns its input signal if *EN* (enable) bit is set and "0" if it is not.

Adder is an accumulative summation block that stores the sum of the previous input signals by every *CLK* rising and returns the current sum when the *RE* bit is set. This means it calculates a weighted sum of the input signals. Its output is the output of the neuron. Also, if the *RE* bit is set, the *bias* of the current neuron is added to the weighted sum and the current sum value is set to 0.

4. NEURAL NETWORK FOR IDENTIFICATION

In a previous paper [1], two kinds of neural networks for adaptation of a caterpillar turn control system (CS) were developed and trained. The proposed method of NN implementation with FPGA was tested on that CS.

An adaptive loop for the caterpillar CS was synthesized for the purpose of researching the proposed approach to identification and adaptive control of non-stationary technological objects. An example of the caterpillar turn CS and its synthesis are shown in [4] where it is possible to consider two non-stationary parameters: *K* and *a* (Fig. 4).

Since the caterpillar is usually employed in rather complex cases, the turn gear is influenced by internal and external disurbances of stochastic nature. This leads to non-stationary change of object parameters (1).

$$G(p) = \frac{1}{p(p+\widetilde{c_1})(p+\widetilde{c_2})}, \qquad (1)$$

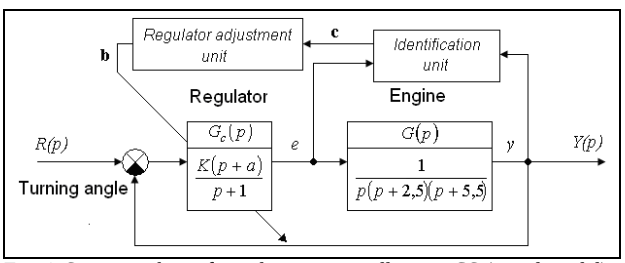

Fig. 4. Structure chart of an adaptive caterpillar turn CS (initial model)

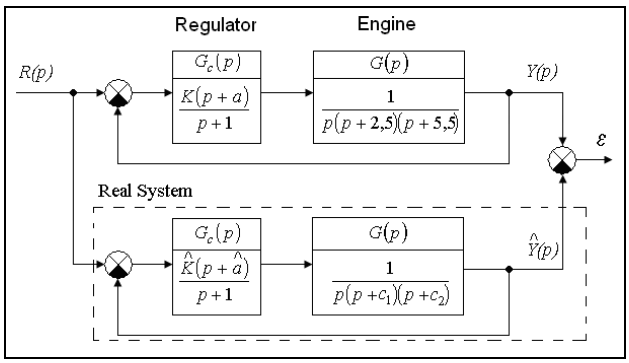

Fig. 5. Error estimation

where $\widetilde{c_1}$ and $\widetilde{c_2}$ are non-stationary parameters which can vary in time. This is initially related to changing parameters of the turn gear transfer function (TF) influenced by the following factors:

- damage caused by external objects and natural phenomena (e.g. sand, dirt, dust or corrosion);
- deterioration or insufficient lubrication of rubbing parts;
- failure of some internal systems or gears;
- other types of damage.

Thus, promptly adjusting the regulator parameters while the CS operates is necessary. An adaptive loop consisting of an identification unit and a regulator adjustment unit was added to the CS in order to solve that problem (Fig. 4).

An effective NN-structure for solving such problems is a multilayer perceptron (MLP) [3]. The NN in question must determine the vector c by processing several serial discrete values of signals *e(k)* and *y(k)* (Fig. 4). The required vector is the TF's parameters vector (1). The moment of the transient end is the moment after which the output signal value differs from the set value by no more than 2%. The stepwise signal is fed to the system input.

The influence of the different number of layers and neurons on the system performance was tested. It turned out that the optimal NN structure is an MLP with 3 hidden layers of 13 neurons: 20-13-13-13-2, trained with the Levenberg–Marquardt algorithm. This NN showed that the minimal relative identification error is 0.86% [1].

5. REGULATOR ADJUSTMENT UNIT

The main task of adaptive control is rapid regulator parameters' adjustment for maintenance of unchangeable (constant) form of output signal in case of object's parameters change [1]. The regulator adjustment unit in question has to estimate the regulator parameters vector $\mathbf{b} = (\mathbf{K} \quad \boldsymbol{a})$ (Fig. 4). The regulator parameters have to be defined in such a way that the target function is minimized. The target function was defined to be a sum-square error (2) of the real output signal and desired form comparison (Fig. 5). One of the most common methods of new parameters' values calculation is the gradient descent. The condition of small argument increase or conditions of gradient smallness are defined as a criterion for search interruption.

$$\varepsilon_\Sigma = \sum_{k=1}^{N} (\varepsilon[k])^2 \qquad (2)$$

The influence of the different number of layers and neurons on the system performance was tested. It turned out that the optimal NN structure is an MLP with 3 hidden layers of 17 neurons: 2-17-17-2, trained with the Levenberg–Marquardt algorithm. This NN showed that the minimal relative identification error is 3.6% [1].

6. MODELING RESULTS

Let us test the obtained adaptive CS with the following example.

Step 1. The initial caterpillar TF is:

$$G(p) = \frac{1}{p(p+2.5)(p+5.5)}. \quad (3)$$

Accordingly, the initial regulator TF is:

$$G_c = \frac{70(p+0.6)}{p+1}. \quad (4)$$

The solid line at Fig. 4 is the transient of the initial caterpillar turn CS. The dotted line is the transient of the CS with changed object parameters. The desired transient is derermined by TFs (3) and (4).

Step 2. Let the object TF parameters change under some disturbances (Fig. 6), so the new TF is:

$$\acute{G}(p) = \frac{1}{p(p+2)(p+5)}. \quad (5)$$

Step 3. Parameters identification is processed by the first synthesized NN. The TF identificational model of $G^i(p)$ is built by the new parameters' values (vector **c**) (6).

$$G^i(p) = \frac{1}{p(p+2.1432)(p+4.986)} \quad (6)$$

As we can see, the identified parameters are sufficiently close to real.

Step 4. The received vector of object parameters is fed to the regulator adjustment unit where the second synthesized NN allows finding the new regulator parameters (vector **b)** for TF $G_c^a(p)$ (7).

$$G_c^a = \frac{63.378(p+0.3342)}{p+1} \quad (7)$$

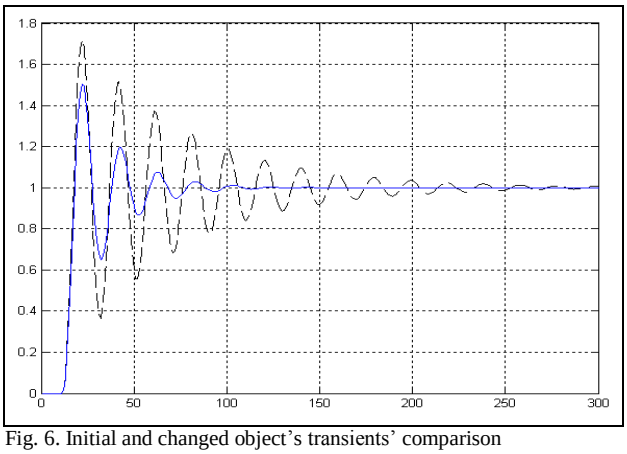

Fig. 6. Initial and changed object's transients' comparison

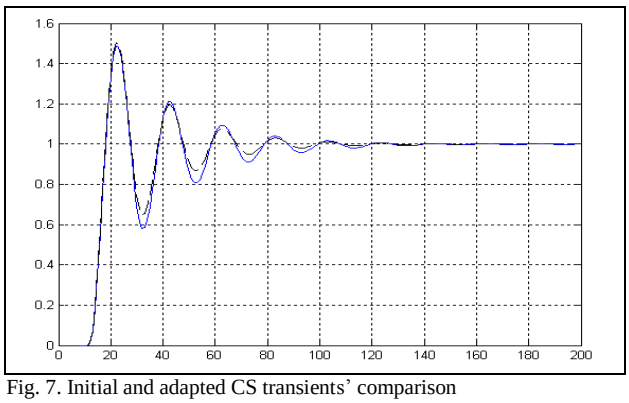

Fig. 7. Initial and adapted CS transients' comparison

The solid line at Fig. 7 is the transient of the initial erpillar turn CS, and the dotted line is the adapted CS transient.

As we can see, the transfer functions are almost identical while the identifier and the regulator does not receive prior knowledge about the initial parameters' values. It implies that the adaptive loop work is satisfactory.

7. CONCLUSION

Cascade neural networks for adaptation of the caterpillar turn CS were developed, trained and implemented with FPGA using a prototype of the cascading NN fragment developed earlier.

The created neural networks are able to restore non-stationary object parameters and calculate new values of the regulator parameters adequately using object input and output signals without taking prior knowledge about its parameters into account. Using neural approach is useful and gives us a possibility to solve adaptation problems in a minimal time with minimal error.

Results are considered to be utilized as hardware for the CS of Ecopyrogenesis plant that is being developed at Admiral Makarov National University of Shipbuilding (Ukraine).

8. REFERENCES

[1] Kondratenko, Y. P. & Gordienko, E. G. (2011). Neural Networks for Adaptive Control System of Caterpillar Turn, *Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium*, ISSN 1726-9679, ISBN 978-3-901509-83-4, Katalinic, B. (Ed.), pp 0305-0306, Published by DAAAM International, Vienna, Austria 2011

[2] Notkyn B. S. & Zmeu K. V. (2005). Effective neural network identification of inverse dynamics of objects for predicting control systems synthesis, *Available from:* http://www.reshebnik.net.ru/getfile.php?nf=pin.pdf *Accessed:* 2011-01-21 (in Russian)

[3] Rutkovska D., Pilinsky M. & Rutkovsky L. (2006). *Neural networks, genetic algorithms and fuzzy systems*, Hotline-Telekom, ISBN 5-93517-103-1, Moscow (in Russian)

[4] Dorf R.C., Bishop R.H. (2006). *Modern control systems*, Basic knowledge lab, ISBN 5-93208-119-8, Moscow (in Russian)

[5] Kazantsev P. A., Ostapenko G. P. & Galushkin A. I. Implementation of the neural network fragment on XILINX FPGA with possibility of real-time weights and transfer function changing, *Available from:* http://www.autex.spb.ru/cgi-bin/download.cgi?dspa2004_1_84 *Accessed:* 2012-03-21 (in Russian)

[6] Galushkin A. I. & Kirsanov D. V. (1999). "Digital neurochips (specialized digital LSI for neurocomputers)", *Foreign radio electronics*, No. 1, 1999, pp. 17-37 (in Russian)