HARDWARE IMPLEMENTATION OF A NEURALNETWORK CONTROLER WITH AN MCU AND AN FPGA FOR A NONLINEAR SYSTEM

Seul Jung and Sung-Su Kim

Intelligent Systems and Emotional Engineering Lab.
Department of Mechatronics Engineering
Chungnam National University, Daejeon, Korea

Abstract: This paper presents the hardware implementation of a neural network controller for a nonlinear system. As a learning algorithm for a neural network, the reference compensation technique has been implemented on a low cost micro-controller unit (MCU), while PID controllers with counters and PWM generators are implemented on an FPGA chip. Interface between an MCU and a field programmable gate array (FPGA) chip has been developed to complete hardware implementation of a neural controller. The neural controller has been tested for controlling the inverted pendulum as a nonlinear system. *Copyright © 2005 IFAC*

Keywords: Reference compensation technique, FPGA, VHDL, ARM, neural controller

1. INTRODUCTION

Nowadays, "Intelligence into the system" has become one of the attractive issues in the control system communities. An endless dream about creating a man-like machine has accelerated researches on developing intelligent systems. One of main tools of intelligence is a neural network system, which mimics a human brain.

Even though, for the most of motion control applications, PID controllers are used as a main controller with the merit of simplicity and easy implementation, PID controllers for nonlinear systems do not work properly as expected. It is well known that fixed PID controller gains have lack of robustness due to uncertainties and outer disturbances. So, introduction of intelligent tools to the nonlinear system will remedy this problem.

A neural network, as a nonlinear controller, is a good candidate that works quite well for a nonlinear system (Miller *et. al.*, 1991). Applications of a neural network can be found in many areas such as motion control systems, signal processing systems, and data processing systems. Specially, successful neural network applications can be found in controlling robot manipulators since they are highly nonlinear MIMO systems (Jung *et. al.*, 2000 and Miyamoto *et. al.* 1998).

Even though the neural network works well, problems arise when a real time implementation issue of a massively parallel neural network learning algorithm is required. Experiments are more difficult to be conducted since the fast computing device is required. The control-loop in the neural control system must be processed fast enough for real time control at every sampling period. Recently, with the help of high-performance and high speed hardware technology, DSPs are available for fast computation in the market.

In our previous researches, successful real time neural network applications have been achieved. The reference compensation technique has been implemented on a high cost DSP system for controlling an x-y table robot and an inverted pendulum (Jung *et. al.*, 2001 and Cho *et. al.*, 2003). As an extension of those researches, we have developed a neural network control hardware on a DSP board with an FPGA (Field Programmable Gate Array) based general purposed PID controller (Kim *et. al.*, 2003 and Jung *et. al.*, 2003). In the literature, most of real time neural network learning algorithms are implemented on DSP boards or even in the PCs. However, their cost is somewhat expensive for a certain simple application such as controlling a DC motor.

In this paper, we extend our previous researches to develop the reference compensation technique of training neural network controllers for nonlinear systems. The purpose is to develop a low cost neural network controller that can be used as a compensator in front of the PID controlled system. By using a PID controller embedded FPGA chip as a main controller for the system, the neural controller can be used as an auxiliary controller that can cancel out uncertainties in the system.

Recently, employing a concept of a system-on-chip (SOC), an FPGA chip as a controller-on-chip (COC) has been developed and used in many applications (Krips *et. al.*, 2002; Cristea *et. al.*, 2001; Abdelkrim *et. al.*, 1997; Thomas *et. al.*, 1999; Oh et al., 2002; Kongmunvattana et al., 1998). As a COC, by using the high flexibility of an FPGA, the additional hardware such as an encoder counter and a PWM generator, can also be implemented in a single FPGA device. As a result, the controller can be designed in a compact form so that it has the cost effectiveness as well as space savings. In addition, noise and power dissipation problems can be further minimized. For a neural network controller, a commercially available general purposed low cost MCU board such as an ARM board is used. The price of an ARM board is much less expensive than that of a DSP board.

Back-propagation learning algorithm has been implemented on the ARM board. Interface between an ARM board and an FPGA has been done to complete intelligent control hardware. In order to show the performance of the developed controller, it was tested for controlling an inverted pendulum. The proposed controller is required to control the angle of the pendulum and the position tracking of the cart simultaneously. Performances of position tracking of the cart while balancing the pendulum were successfully achieved.

2. OVERALL SYSTEM STRUCTURE

The reference compensation technique is known as one of on-line learning control methods of neural network applications. The control block diagram is shown in Fig. 1.

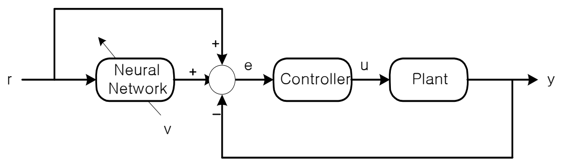

Fig.1 Reference compensation control system structure

A neural network is placed in front of the closed loop controlled system. It functions as a pre-filter to modify reference trajectories by compensating for uncertainties (Jung *et. al.*, 2000). Here we try to implement a neural network control algorithm on the ARM board. Fig. 2 shows the block diagram of interface structure between each module of the neural controller and an FPGA. The ARM board communicates with the FPGA to give compensated signals and the FPGA calculates errors to form PID controllers and then generates PWM signals to motor drivers.

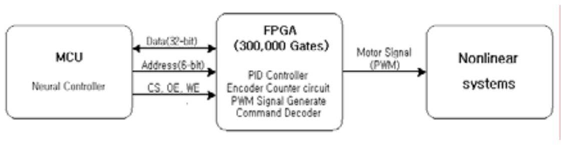

Fig.2. Overall system block diagram

A neural network is placed in front of the closed loop controlled system. It functions as a pre-filter to modify reference trajectories by compensating for uncertainties (Jung *et. al.*, 2000). Here we try to implement a neural network control algorithm on the ARM board. Fig. 2 shows the block diagram of interface structure between each module of the neural controller and an FPGA. The ARM board communicates with the FPGA to give compensated signals and the FPGA calculates errors to form PID controllers and then generates PWM signals to motor drivers.

3. PID EMBEDDED ON FPGA

The FPGA based PID controller consists of a communication block, an encoder counter block, a PID calculation block, and a PWM generation block. Input signals are 32 bit data, a 6 bit address, control signals such as CS, OE, WE, encoder signals, and a 25MHz clock. Output signals are PWM signals. Fig.3 shows the block diagram of inside the PID controller.

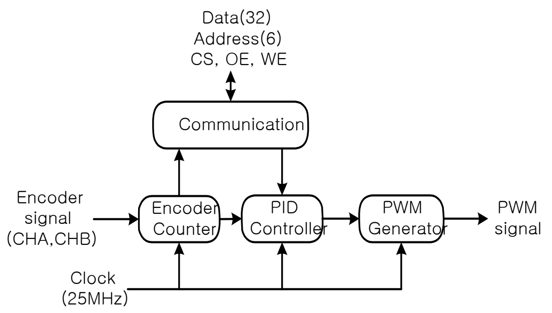

Fig 3. Inside block diagram of PID controller

The detailed description and function of each block can be found in the previous paper [7].

4. MCU BOARD

A commercially available general purposed ARM(Advanced RISC Microprocessor) board of Samsung is used for neural network controller implementation. In order for an ARM to communicate with FPGA PID controllers, a 32 bit data bus is used to share data. The ARM board has a 32 bit 66MHz RISC structured microprocessor. At every sampling time, the ARM board has to give compensation values to the FPGA so that the FPGA can add those values to PID controllers. This kind of process can be done by communication between the MCU and the FPGA. Since a whole calculation has to be done in one sample time, the ARM board and the FPGA have to be synchronized.

5. NEURAL NETWORK CONTROLLER

5.1 Reference compensation technique

In this paper, we are implementing an on-line learning algorithm for a neural network. The reference compensation technique has been proposed by Jung and Hsia [2]. The idea of the RCT is that the neural network compensates at input level by modifying input signals. The same objective function of the feedback error learning method can be minimized in on-line fashion [3].

The angle error is formed as

$$e_\theta = \theta_d - \theta \tag{1}$$

where θ_d is a desired angle and θ is an actual angle of the pendulum.

A PID controller for an angle control is defined as

$$\begin{aligned} u_\theta &= k_{P\theta} e_\theta(t) + k_{I\theta}\int e_\theta(t)dt + k_{D\theta}\dot{e}_\theta(t) \\ &+ k_{P\theta}\phi_1 + k_{I\theta}\phi_2 + k_{D\theta}\phi_3 \end{aligned} \tag{2}$$

where ϕ_1, ϕ_2, ϕ_3 are neural network outputs.

The cart position is controlled as well as the pendulum angle. The position tracking error is formed as

$$e_x = x_d - x \tag{3}$$

where x_d, x are a desired cart position and an actual cart position, respectively.

A PID control for a cart can be formed as

$$\begin{aligned} u_x &= k_{Px} e_x(t) + k_{Ix}\int e_x(t)dt + k_{Dx}\dot{e}_x(t) \\ &+ k_{Px}\phi_4 + k_{Ix}\phi_5 + k_{Dx}\phi_6 \end{aligned} \tag{4}$$

where ϕ_4, ϕ_5, ϕ_6 are neural network outputs.

The overall control input for the inverted pendulum system is to add (2) and (4) as

$$u = u_x + u_\theta \tag{5}$$

Fig. 4 shows the control block diagram for an inverted pendulum control. Inputs to the neural network are desired angle and position as well as delays of actual angle and position. Neural network outputs are added to PID errors. Those compensating signals are multiplied by PID controller gains to generate compensating torques to the system.

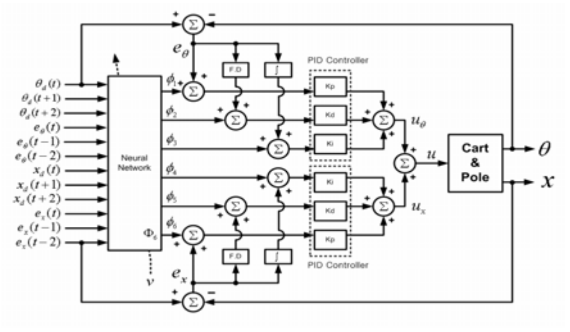

Fig.4. Control block diagram

5.2 Neural network learning

For a neural network structure shown in Fig. 5, we have used a general feed-forward structure that has an input layer, a hidden layer, and an output layer. The numbers of input layer, hidden layer, and output layer are 12, 9, and 6, respectively.

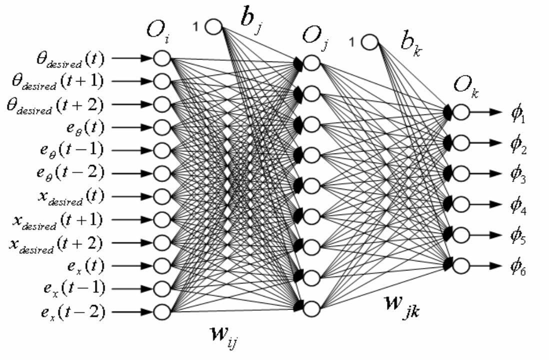

Fig.5. Neural network structure

For a nonlinear function at hidden layer and output layer we have used a hyperbolic tangent function as

$$f(x) = \frac{1-\exp(-x)}{1+\exp(-x)} \tag{6}$$

Here, the neural network learning algorithm is derived. Since we are doing on-line learning and control, selecting the training signal is very important. Neural network outputs are defined as

$$\Phi = \Phi_\theta + \Phi_x \tag{7}$$

$$\Phi_\theta = k_{p\theta}\phi_1 + k_{d\theta}\phi_2 + k_{i\theta}\phi_3 \tag{8}$$

$$\Phi_x = k_{px}\phi_4 + k_{dx}\phi_5 + k_{ix}\phi_6 \qquad (9)$$

If $f(\theta,\dot{\theta},\ddot{\theta})$ is the system dynamics, equation (2) and (4) can be represented as follows:

$$k_{p\theta}e_\theta + k_{d\theta}\dot{e}_\theta + k_{i\theta}\int e_\theta + k_{px}e_x + k_{dx}\dot{e}_x + k_{ix}\int e_x$$

$$= f(\ddot{\theta},\dot{\theta},\theta,\ddot{x},\dot{x},x) - \Phi \qquad (10)$$

If we make the left side of (10) become zero, then $\Phi \cong f(\theta,\dot{\theta},\ddot{\theta})$ can be obtained. This means that inverse dynamics control can be achieved. So here we define the training signal of neural network as the left side of (10).

$$v = k_{p\theta}e_\theta + k_{d\theta}\dot{e}_\theta + k_{i\theta}\int e_\theta + k_{px}e_x + k_{dx}\dot{e}_x + k_{ix}\int e_x \qquad (11)$$

The objective function is defined as

$$E = \frac{1}{2}v^2 \qquad (12)$$

Differentiating (12) with respect to the weight, we have

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial v}\frac{\partial v}{\partial w} = v\frac{\partial v}{\partial w} = -v\frac{\partial \Phi}{\partial w} \qquad (13)$$

Here, we have

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial v}\frac{\partial v}{\partial w} = v\frac{\partial v}{\partial w} = -v\frac{\partial \Phi}{\partial w} = -v\left(\frac{\partial \Phi_\theta}{\partial w} + \frac{\partial \Phi_x}{\partial w}\right) \qquad (14)$$

The update equation in the back propagation algorithm is

$$\Delta w(t) = \eta\frac{\partial \Phi}{\partial w}v + \Delta w(t-1) \qquad (15)$$

$$w(t+1) = w(t) + \Delta w(t) \qquad (16)$$

5.3 Hardware implementation of neural network controller

We used an ARM board of Samsung and an FPGA board of Altera APEXⅡ EP20K300EQC240. Fig. 6 shows a real figure of the designed controller.

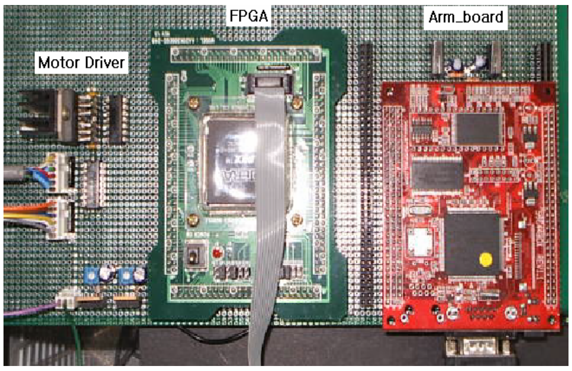

Fig. 6. MCU-FPGA Hardware

6. EXPERIMENTAL SETUP

Experiment is to control the angle of pendulum and the position of cart simultaneously. Fig. 7 shows the experimental setup of the inverted pendulum system. The system consists of an inverted pendulum, a hardware controller, and a PC.

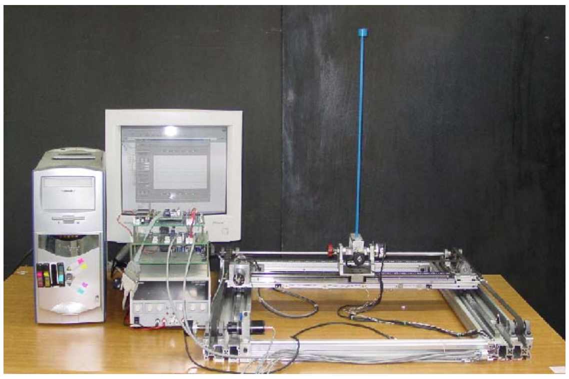

Fig. 7. Inverted pendulum system

Optimised PID controller gains and neural network parameters are listed in Table 1.

Table 1. Controller gains

Parameters		Gain
Angle	k_p	−2.3
	k_i	0.0008
	k_d	−0.542
Position	k_p	−0.625
	k_i	−0.021
	k_d	−0.542
Learning rate(η)		0.205
Momentum(α)		0.3

7. EXPERIMENTS

7.1 Balancing control

We have found that the maximum sampling time can be achieved at 10msec. This sampling rate is much slower than that of a DSP board. Even though the ARM board is fast enough, computing back-propagation algorithm can be a burden. For control application, however, 10msec sampling time is acceptable. The experimental results are shown next. Figs. 8 and 9 show the pendulum angle error and the cart position error by the controller, respectively. The pendulum is well balanced and the cart is well maintained at the desired position.

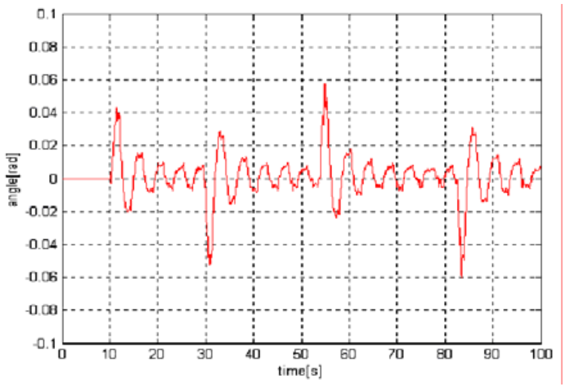

Fig. 8. Angle of the pendulum

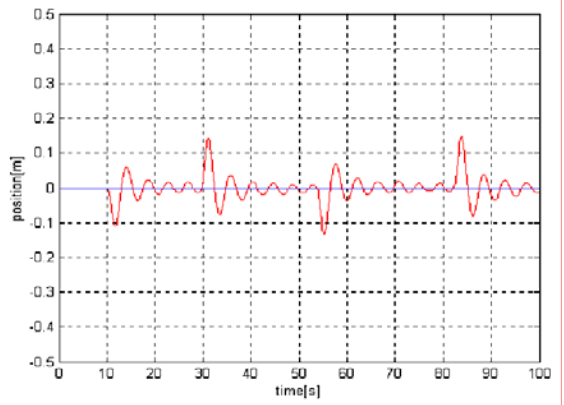

Fig. 9. Position of the cart

7.2 Position tracking

Another interesting experiment has been conducted to test the performance of the proposed controller. The cart is required to track a desired sinusoidal trajectory while balancing the pendulum. The cart is required to move along axis about 40cm in distance. Here we tested two cases: one is when a digital filter is used and the other is not.

Case 1 : Without a digital filter

We have found that derivative terms as a D controller after a finite difference computation are very noisy. Even though controlling balancing and position tracking are successful, vibration has been observed. Figs. 10 and 11 show balancing and tracking control results. We observed that about 3cm overshoot occurred in cart position tracking in Fig 11.

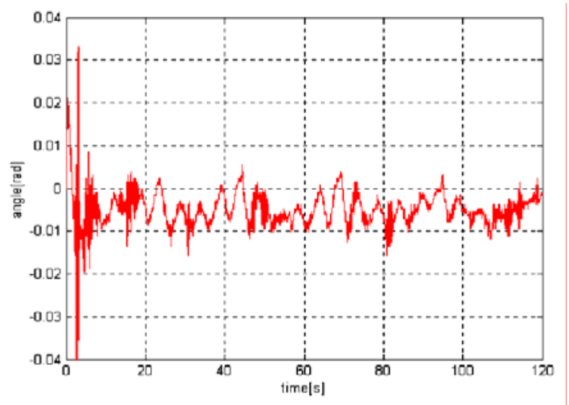

Fig.10. Pendulum angle when T= 8π sec

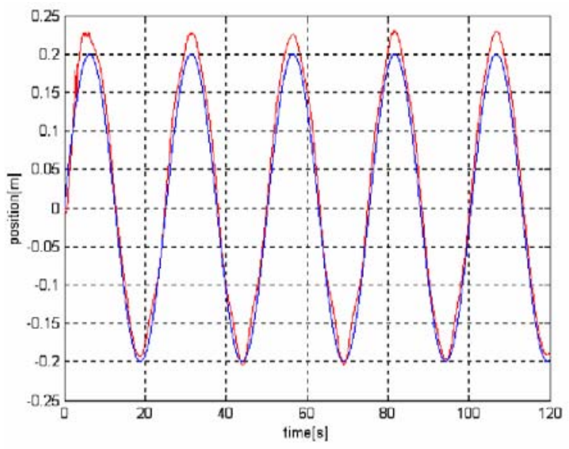

Fig.11. Position tracking of the cart when T = 8π sec . Blue line is the reference, and the red line is actual trajectory.

Fig. 12 shows the corresponding torque that has glitches to cause vibration a little bit. In order to eliminate vibration, we used a digital filter to get rid of glitches. Even though the system is stable, the pendulum keeps oscillating with a large error.

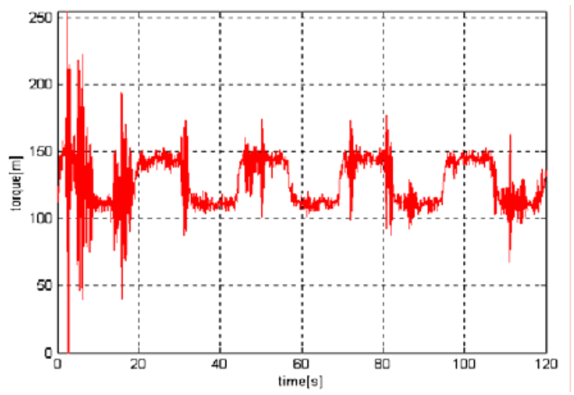

Fig.12. Control torque when T = 8π sec

Case 2 : With a digital filter

We have used a digital filter for smoothing the signal after a finite difference process. We have experimentally found that a vibrating frequency at above 5Hz. So we filter derivative terms out with a low pass filter at the cutoff frequency of 5 Hz. The designed 2nd order IIR filter is

$$H(z)=\frac{0.75596+0.511922z^{-1}+0.7559611z^{-2}}{1+0.45445z^{-1}+0.572398z^{-2}}$$

The results are shown in Figs. 13 and 14. We see that performances are much better than that of without a filter. Specially, for the cart position tracking, overshoots have been minimized very much as shown in Fig. 14. We can clearly see the reason from Fig. 15 that torques are smoother than that of case 1 shown in Fig. 12.

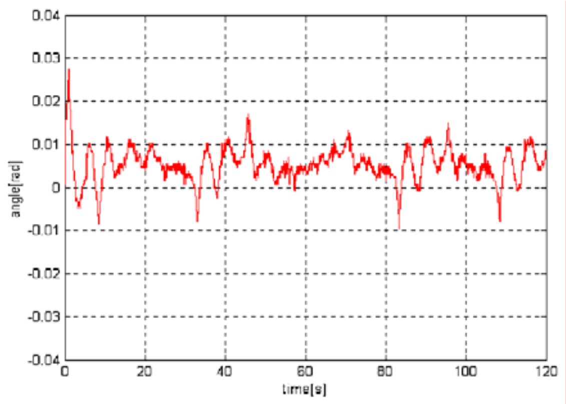

Fig.13. Pendulum angle when T= 8π sec

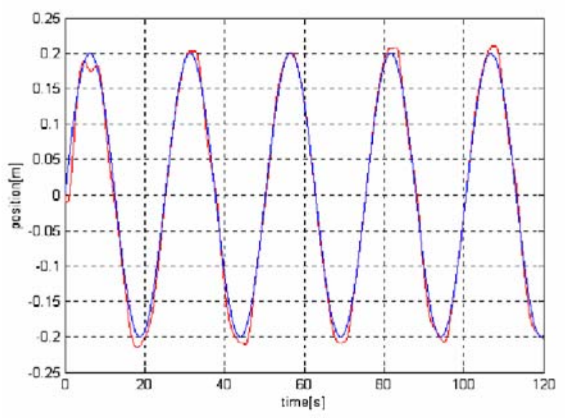

Fig.14. Position tracking of the cart when T = 8π sec (blue line : the reference trajectory and the red line :the actual trajectory).

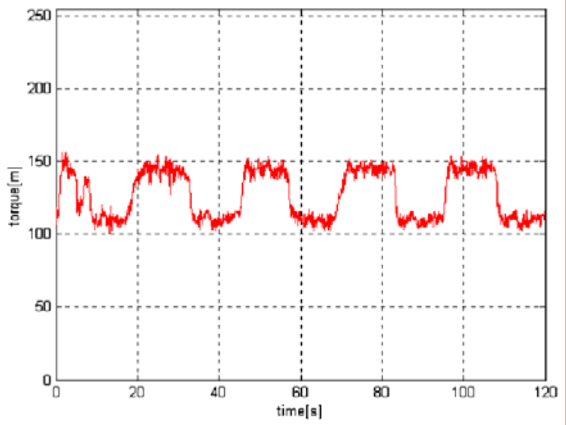

Fig.15 Control torque when T = 8π sec

In these experiments, the pendulum is balanced within a 0.015rad angle error and the cart tracks desired trajectories within the error of 1 ㎝.

8. CONCLUSIONS

This paper presented the hardware implementation of a neural network controller with an FPGA based PID controller. The proposed controller was tested by performing control of an inverted pendulum. Even though the overall sampling time is less than that of a DSP, the proposed controller improves position tracking errors of the cart while balancing of the pendulum. Experimental studies confirmed that the implemented low cost neural network controller can be used for any nonlinear system whose output signals are available since output signals are used to form errors to train a neural network on line.

However, we failed to control two degrees-of-freedom pendulum since the sampling time is not fast enough for calculating two neural networks separately. Real time control of two massively parallel neural networks was a burden for a single MCU. A faster MCU will be used in the future.

REFERENCES

Miller. W.T., R. S. Sutton, and P. J. Werbos (1991), *Neural Networks for Control*, The MIT Press

Jung S. and T. C. Hsia (2000), Neural network Inverse Control Techniques for PD Controlled Robot Manipulator, *ROBOTICA*, **vol. 19, No 3**, pp. 305-314

Miyamoto H., K. Kawato, T. Setoyama, and R. Suzuki(1988), Feedback error learning neural network for trajectory control for of a robotic manipulator, *Neural Networks*, **vol. 1**, pp. 251-265

Jung S. and S. B. Yim (2001) Experimental studies of neural network control technique for nonlinear systems, **vol. 7, no. 11**, pp. 918-926

Cho H. T. and S. Jung (2003), Balancing and Position control of an Inverted pendulum on an X-Y Plane using Decentralized neural networks, *2003 International Conference on Advanced Intelligent Mechatronics*, pp. 181- 186.

Kim S. S and S. Jung S.(2003), Implementation of an Intelligent Controller with a DSP and an FPGA for Nonlinear Systems, *ICCAS*, pp.575-580

Kim S. S and S. Jung S(2003), Development of a General Purpose Motion Controller Using a Field Programmable Gate Array, *ICCAS*, pp. 360-365

Krips M, T. Lammert, A. Kummert (2002), FPGA implementation of a neural network for a real-time hand tracking system, *First IEEE International Workshop on Electronic Design, Test and Applications*, pp. 313 317.

Cristea M., J Khor, M McCormick (2001) FPGA fuzzy logic controller for variable speed generators**,** *IEEE International Conference on Control Applications*, pp. 301 304.

Abdelkrim K. Oudjida et al(1997), A reconfigurable counter controller for digital motion control application, *Microelectronics Journal*, vol. 28, no. 6-7.

Thomas F. et al (1999), Design and implementation of a wheel speed measurement circuit using field programmable gate arrays in a spacecraft, *Microprocessors and Microsystems*, pp. 553-560

Oh et al (2002), Design of a biped robot using DSP and FPGA, *FIRA Robot World Congress*. 698-701.

Kongmunvattana A. and P. Chongstivatana (1998), An FPGA-based Behavioral Control System for a Mobile Robot, *IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 759-762, 24-27.