Neuro-fuzzy Control of a Position-Position Teleoperation System Using FPGA

Hocine Khati, Rabah Mellah, Hand Talem
Faculty of Electrical and Computing Engineering
University Mouloud Mammeri of Tizi-Ouzou
Tizi-Ouzou, Algeria.
hoc.khati@gmail.com

***Abstract*— This paper presents an ANFIS (adaptive neuro-fuzzy inference system) controller for a teleoperation system using FPGA (Field Programmable Gate Array). The proposed controller allows adapting to the dynamic variations of the master and slave models by adjusting the output parameters of the neuro-fuzzy network using a learning algorithm, while taking advantage of the benefits of the FPGA computing power and its high sampling frequency. The ANFIS controllers are developed in MATLAB-Simulink environment and implemented using Simulink's Fixed point tool and HDL Coder. These features provide a fast and accurate control algorithm while optimizing the hardware resources used by the FPGA. The proposed controllers are implemented on a teleoperation system with one degree of freedom. The experimental position tracking results clearly show that the proposed control algorithm guarantees better performance compared to conventional control methods (PID).**

Keywords*— *Teleoperation, ANFIS, Neuro-fuzzy, FPGA, HDL Coder

I. INTRODUCTION

A teleoperation system enables a human operator to manipulate a machine in a remote environment while keeping it away from the risks associated with hazardous work [1]. A teleoperation system is composed of two robots called master robot and slave robot [2]. The human operator manipulates the master robot while the slave reproduces his movements [3]. In a teleoperation system, positions and forces are transmitted from the master and the slave and vice versa. The contact force of the slave with the environment is sent back to the operator via the master station and will then allow him to evaluate the situation of the slave robot [4]. The basic architectures of teleoperation systems have been presented in [4]. The position-position control architecture was the first method implemented on teleoperation systems [5]. Its operation relies on the exchange of position information only on the communication channel. Its main advantage is that no force sensor is used at the remote site [6], the force feedback is then created on the basis of the position error between the master station and the slave station [5]. In a bilateral teleoperation, it is desirable that the system is stable and transparent [7] so that the slave reproduces the movements of the master with fidelity and that the operator feels directly in contact with the distant environment, nevertheless, master and slave robots represent nonlinear and complex systems, especially when the slave is in contact with the remote environment. Therefore, the control algorithms must be robust and efficient. For this, this paper proposes an adaptive control based on the methods of artificial intelligence using an FPGA. The techniques of artificial intelligence through the concepts of fuzzy logic, neural networks, and neuro-fuzzy networks, ensure high performance without resorting to a mathematical model. The combination of fuzzy logic and artificial neural networks allows taking advantage of the inference capabilities of fuzzy logic with the learning ability of neural networks [8]. Neuro-fuzzy and adaptive techniques used in the control of bilateral teleoperation systems were studied in [8,9,10,11], the aim of which was to improve teleoperation performances by compensating for the uncertainties dynamics of the master-slave system. This paper proposes an adaptive ANFIS controller for a master-slave system with unknown dynamic. ANFIS controllers are used in a position-position control architecture to adapt to master-slave system dynamics, by adjusting the output parameters of the neuro-fuzzy network using a learning algorithm based on the extended Kalman filter, by taking advantage of computing power of the FPGA and its high sampling frequency [12].

The main advantages of FPGAs are the high computing speed and parallel data processing [3]. It can run multiple processes at the same time with a high operating frequency, contrary to ordinary processors that can only run one process at a time, with a lower frequency [13], the FPGA is therefore an appropriate circuit to perform the teleoperation control [14], because it allows to accelerate the control algorithm, which can increase performances. In [3,12], FPGA implementations of sliding mode control for bilateral teleoperation using a LabVIEW-based design methodology for rapid prototyping is studied. This paper proposes an implementation on Zynq-7000 FPGA development board of an ANFIS controller by using the MATLAB-Simulink functionalities. The fixed point tool allows automating the conversion of Simulink models (position controllers) developed in double-precision floating point to fixed-point models, and HDL Coder will generate the appropriate VHDL code, optimized in terms of hardware resources, and execution time [15] that will be implemented on the FPGA.

The structure of paper is organized as follows: In Section II, the position-position architecture is presented. In section III, the ANFIS controller is developed and control stability is also studied. Section IV describes FPGA implementation of two controllers, ANFIS and PID, on a master-slave system. Section V shows the realization of our teleoperation system. Experimental results justifying the effectiveness of the proposed controller are presented in section VI. Finally, section VII concludes this paper.

II. POSITION-POSITION CONTROL ARCHITECTURE

Historically, this is the first control method implemented on haptic teleoperation schemes in 1950 [5]. It is based on the exchange of position measurements between the master and the slave. In this control strategy, the position of the master arm is returned to the slave arm and inversely. In this work, we will control the master and the slave in position using two ANFIS controllers (Fig.1).

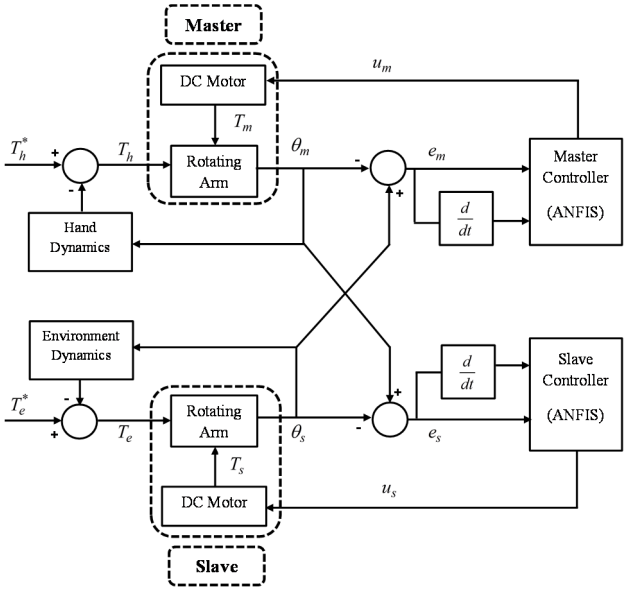

Fig. 1. Position-position control architecture with ANFIS controller.

In this scheme, θ_m and θ_s are the positions of the master and the slave respectively, T_e is the interaction torque of the slave with the environment and T_h is the torque produced by the user on the master. u_m and u_s are the control signals delivered by the actuators, to control the movement of the master and the slave respectively.

III. ADAPTIVE NEURAL FUZZY CONTROLLER

The combination of neural networks and fuzzy logic makes it possible to build robust neuro-fuzzy controllers. The neuro-fuzzy system can automatically optimize and adjust fuzzy rules and membership functions by a self-learning algorithm [16]. One of the most popular neuro-fuzzy systems is the adaptive neuro-fuzzy inference algorithm (ANFIS).

A. Adaptive neuro fuzzy inference system (ANFIS)

The ANFIS structure was proposed by Jang in 1993 [17]. It is a class of adaptive networks that has the advantage of automatically optimizing and adjusting the membership functions using a learning algorithm. In this work, we consider a neuro-fuzzy network with a first-order fuzzy inference model of Sugeno, with two inputs x_1 and x_2, and a single output f, as shown in Fig.2. In this subsection, we develop the position controller for the master, and the development is the same for the slave system.

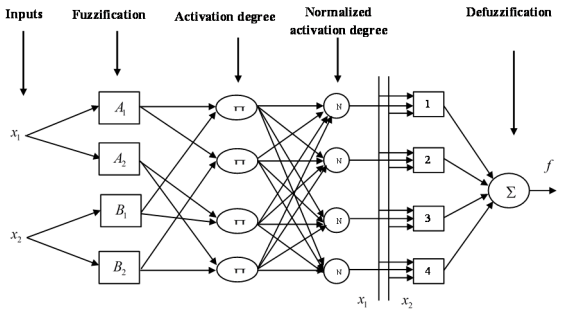

Fig. 2. ANFIS controller architecture.

In order to obtain a precise control signal to have a real output very close to the desired output, we used the error and its derivative (respectively x_1 and x_2) as inputs of the ANFIS controller.

The fuzzy rules are defined as follows:

Rule 1:If x_1 is A_1 and x_2 is B_1 then $f_1 = p_1x_1 + q_1x_2 + r_1$ (1)

Rule 2:If x_1 is A_1 and x_2 is B_2 then $f_2 = p_2x_1 + q_2x_2 + r_2$ (2)

Rule 3:If x_1 is A_2 and x_2 is B_1 then $f_3 = p_3x_1 + q_3x_2 + r_3$ (3)

Rule 4:If x_1 is A_2 and x_2 is B_2 then $f_4 = p_4x_1 + q_4x_2 + r_4$ (4)

Where $f_j = p_jx_1 + q_jx_2 + r_j$ for $j = 1,2,\ldots,4$, A_i and B_i are fuzzy subsets, p_j, q_j and r_j are the consequences parameters of the rule j.

We associate two fuzzy sets for each of the inputs x_1 and x_2 namely *N (Negative)* and *P (Positive)*. μ_N and μ_P are the degrees of membership of the variables x_i to the fuzzy subsets A_i and B_i, defined by the membership functions illustrated in Fig.3, and defined as follows:

For $i = 1, 2$

$$\mu_P(x_i) = \begin{cases} 0 & if \quad x_i < -1 \\ 0.5x_i + 0.5 & if \quad -1 < x_i < 1 \\ 1 & if \quad x_i > 1 \end{cases} \tag{5}$$

$$\mu_N(x_i) = \begin{cases} 1 & if \quad x_i < -1 \\ -0.5x_i + 0.5 & if \quad -1 < x_i < 1 \\ 0 & if \quad x_i > 1 \end{cases} \tag{6}$$

Using the defuzzification method by the center of gravity, the numerical value of the output u is given by [18]:

$$u = \frac{\sum_{j=1}^{4} f_j w_j}{\sum_{j=1}^{4} w_j} \tag{7}$$

Where:

$$\begin{cases} w_1 = \mu_P(x_1).\mu_P(x_2) = \mu_{A_1}(x_1).\mu_{B_1}(x_2) \\ w_2 = \mu_P(x_1).\mu_N(x_2) = \mu_{A_1}(x_1).\mu_{B_2}(x_2) \\ w_3 = \mu_N(x_1).\mu_P(x_2) = \mu_{A_2}(x_1).\mu_{B_1}(x_2) \\ w_4 = \mu_N(x_1).\mu_N(x_2) = \mu_{A_2}(x_1).\mu_{B_2}(x_2) \end{cases} \tag{8}$$

B. Learning algorithm

In this work, we consider that the parameters of the premises are fixed, whereas those of the consequences are adjusted by the minimization of the following objective function [9]:

$$J = \frac{1}{2}(y_d - y)^2 \tag{9}$$

Where y_d and y are respectively the desired and actual outputs of the master system (respectively θ_s and θ_m).

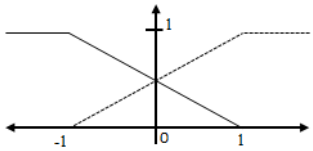

Fig. 3. Membership functions.

Let be Φ_j the vector of the consequences parameters. The objective is to find the parameters p_j, q_j and r_j of the vector Φ_j using the gradient descent method [19] as follows:

$$\Phi_j(k+1) = \Phi_j(k) - \alpha(k)\frac{\partial J}{\partial \Phi_j} \tag{10}$$

$$\Phi_j(k+1) = \Phi_j(k) - \alpha(k).\left(-(y_d - y)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j}\right) \tag{11}$$

$$\Phi_j(k+1) = \Phi_j(k) + \alpha(k)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j}e \tag{12}$$

Where $e = y_d - y$.
The term $\frac{\partial y}{\partial u}$ cannot be calculated, but it can be approximated using the extended Kalman filter equations.

Equation (12) can be written as:

$$\Phi_j(k+1) = \Phi_j(k) + K_j'(\Psi_j)^T e \tag{13}$$

Where:

$$(\Psi_j)^T = \frac{\partial u}{\partial \Phi_j} = \begin{bmatrix} \frac{\partial u}{\partial p_j} \\ \frac{\partial u}{\partial q_j} \\ \frac{\partial u}{\partial r_j} \end{bmatrix} \tag{14}$$

$$K_j' = \alpha(k)\frac{\partial y}{\partial u} \tag{15}$$

Equation (13) can be identified to the extended Kalman filter equation [20]:

$$\Phi_j(k+1) = \Phi_j(k) + K(k)e \tag{16}$$

Where $K(k)$ is the Kalman gain defined as follows:

$$K(k) = \frac{P(k)H^T(k)}{H(k)P(k)H^T(k) + R(k)} \tag{17}$$

Where $H(k)$ is the observation matrix of the system, $P(k)$ is the covariance estimation matrix of the error and $R(k)$ is the covariance matrix of the process noise.
Taking $H^T(k) = (\Psi_j)^T$, $P(k) = \lambda_1$ and $R(k) = \lambda_2$, the Kalman gain $K(k)$ can be written :

$$K(k) = \frac{\lambda_1}{(\Psi_j)\lambda_1(\Psi_j)^T + \lambda_2}(\Psi_j)^T \tag{18}$$

Consequently, (16) becomes:

$$\Phi_j(k+1) = \Phi_j(k) + \frac{\lambda_1}{(\Psi_j)\lambda_1(\Psi_j)^T + \lambda_2}(\Psi_j)^T e \tag{19}$$

By identification between (13) and (19), we have:

$$K_j' = \frac{\lambda_1}{(\Psi_j)\lambda_1(\Psi_j)^T + \lambda_2} \tag{20}$$

Finally, the vector of parameters Φ_j can be adjusted using the following formula:

$$\Phi_j(k+1) = \Phi_j(k) + K_j'(\Psi_j)^T e \tag{21}$$

By considering a very short sampling period T_e, we have:

$$\dot{\Phi}_j = \frac{\Phi_j(k+1) - \Phi_j(k)}{T_e} = K_1(\Psi_j)^T e \tag{22}$$

Where $K_1 = \frac{K_j'}{T_e}$.
This implies that:

$$\dot{\Phi}_j = K_1(\Psi_j)^T e = (\Psi_j)^T e_u \tag{23}$$

Where $e_u = K_1 e$.
Let be $\tilde{\Phi}_j = \Phi_{jd} - \Phi_j$, where Φ_j is the vector of the parameters and Φ_{jd} the vector of the desired parameters. That implies :

$$\dot{\tilde{\Phi}}_j = \dot{\Phi}_{jd} - \dot{\Phi}_j = -(\Psi_j)^T e_u \tag{24}$$

Where $e_u = u_d - u$ is the control error, u_d is the controller's desired output and u is the controller's actual output. It is defined by:

$$\begin{aligned} e_u &= u_d - u = \sum_{j=1}^{4}\left((\Psi_j)\Phi_{jd} - (\Psi_j)\Phi_j\right) \\ e_u &= \sum_{j=1}^{4}\left((\Psi_j)(\Phi_{jd} - \Phi_j)\right) = \sum_{j=1}^{4}\left((\Psi_j)\tilde{\Phi}_j\right) \end{aligned} \tag{25}$$

C. Stability Analysis of the control algorithm

Consider the following Lyapunov function [21]:

$$V_j = \frac{1}{2}\sum_{j=1}^{4}\left((\tilde{\Phi}_j)^T(\tilde{\Phi}_j)\right) \tag{26}$$

Differentiating V_j with respect to time yields, we obtain:

$$\dot{V}_j = \sum_{j=1}^{4}\left(\left(\dot{\tilde{\Phi}}_j\right)^T(\tilde{\Phi}_j)\right) \tag{27}$$

By using (24) and (25), we can obtain:

$$\begin{aligned} \dot{V}_j &= \sum_{j=1}^{4}\left(\left(-(\Psi_j)^T e_u\right)^T(\tilde{\Phi}_j)\right) \\ \dot{V}_j &= -(e_u)^T\sum_{j=1}^{4}\left((\Psi_j)(\tilde{\Phi}_j)\right) = -(e_u)^T(e_u) \end{aligned} \tag{28}$$

Consequently:

$$\dot{V}_j = -\left(e_u\right)^T \left(e_u\right) \leq 0 \qquad (29)$$

Since $\dot{V}_j \leq 0$, so we conclude that the system is asymptotically stable in the sense of Lyapunov [8,22].

IV. FPGA IMPLEMENTATION

A. Design methodology of the control algorithm

FPGA designs may be based on the fixed-point or floating-point data type [23], however, fixed-point implementations are still more efficient than their floating-point counterparts because they consume fewer hardware resources [15] with shorter calculation time.

The controllers of master and slave are developed in the MATLAB-Simulink environment using HDL Coder supported Simulink blocks, which are in the form of double precision floating point data, then they are converted to fixed-point models by the fixed point tools tool. The implementation on FPGA is done via the Simulink Workflow Advisor, where HDL Code generation generates the appropriate VHDL code to program the FPGA. The last step is to create the project via system integration through Xilinx-Vivado to create the bitstream file that will be loaded on the FPGA via the JTAG cable.

B. Resources consumption

When generating VHDL code, HDL Coder enables several types of optimizations that allow you to share FPGA resources to reduce the area occupied by the design [15].

Table I and Table II describe the FPGA hardware resources consumed when implementing PID and Neuro-Fuzzy control algorithms, with a sampling frequency of 1 MHz. As can be seen, the hardware resources used are optimized thanks to the adopted design methodology.

TABLE I. FPGA RESOURCES USAGE WITH PID CONTROLLERS

Resources	Utilization	Available	Utilization %
FF (Flip-Flop)	726	106400	0.68
LUT (Look Up Table)	898	53200	1.69
Memory LUT	64	17400	0.37
I/O (Input/Output)	16	200	8.00
DSP48 (Digital Signal Processor)	10	220	4.55
BUFG (Global Buffer)	3	32	9.38
MMCM (Mixed-Mode Clock Manager)	1	4	25.00

TABLE II. FPGA RESOURCES USAGE WITH ANFIS CONTROLLERS

Resources	Utilization	Available	Utilization %
FF (Flip-Flop)	1064	106400	1.00
LUT (Look Up Table)	5064	53200	9.52
Memory LUT	64	17400	0.37
I/O (Input/Output)	16	200	8.00
DSP48 (Digital Signal Processor)	134	220	60.91
BUFG (Global Buffer)	3	32	9.38
MMCM (Mixed-Mode Clock Manager)	1	4	25.00

C. Electronic circuit of the system

The positions of master and slave robots are measured through hall effect encoders integrated into DC motors. The FPGA allows the acquisition of encoder signals A and B to calculate the positions of the two robots, which are injected into the position controllers to calculate PWM control signals, and identify the directions of rotation S1 and S2 of each motor. Master and slave motors are driven by L298N drivers, which are composed of a double bridge H, designed to control DC motors. The electronic circuit of our system is shown in Fig.4.

V. EXPERIMENT

The system consists of two identical devices, master and slave. Each device consists of an arm actuated by a DC motor 12V (DFRobot model 28PA51G).These motors are equipped with integrated hall encoders and encoder adapter cards (FIT0324) which allow recovering the position signal of the motor shaft. The hall encoder produces 2700 pulses per turn and can detect 0.54 degrees rotation of the shaft.

In order to confirm the robustness of ANFIS controllers, we compare its performances with those of conventional PID controllers. Therefore, master and slave regulators for each PID and ANFIS control strategy, encoders, and PWMs have been programmed with blocks of Simulink supported by HDL Coder. The two controllers were implemented via the Workflow Advisor from Simulink on a Zedboard development board with an FPGA Xilinx Zynq-7000 AP-SoC-XC7Z020-CLG484[24], by considering sampling frequency of 1MHz. The master-slave experimental system is illustrated in Fig.5.

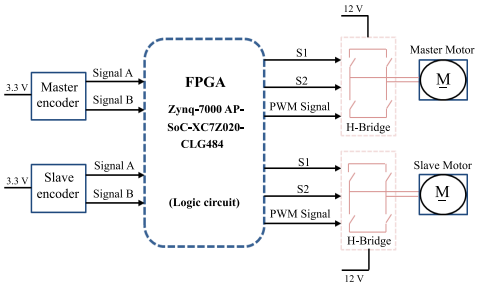

Fig. 4. Electronic circuit of the teleoperation system.

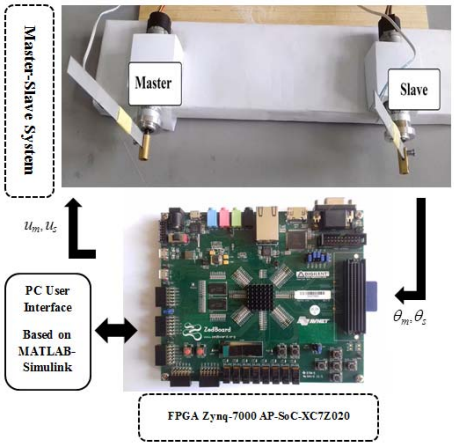

Fig. 5. The master-slave experimental system.

VI. EXPERIMENTAL RESULTS

In order to evaluate the performances of the proposed control strategy, we compared in identical conditions the performances of neuro-fuzzy controllers with those of PID controllers.

First, the PID position controllers are implemented with the parameters P = 20, I = 2, D = 0.1 Experimental results showing position tracking in free motion are shown in Fig.6 and Fig.7. Fig.6 represents the time evolution of the positions of the master and the slave. From this figure, we can see that the trajectory of the slave position follows the variations of the master position; however, a small tracking error can be noticed through Fig.7.

Secondly, neuro-fuzzy position controllers are implemented. Fig.8 represents the time evolution of the position of the master and that of the slave. From this figure, we can see that the slave device follows correctly the master device with a very small error as shown in the Fig.9. This control strategy has very good tracking performance and robustness compared to conventional PID control. From these figures, we can see that neuro-fuzzy controllers (ANFIS) offer good tracking performances because the slave reproduces position movements of the master with fidelity. The teleoperation system using ANFIS controllers offers better performances of tracking and robustness compared to the teleoperation system with conventional PID controllers.

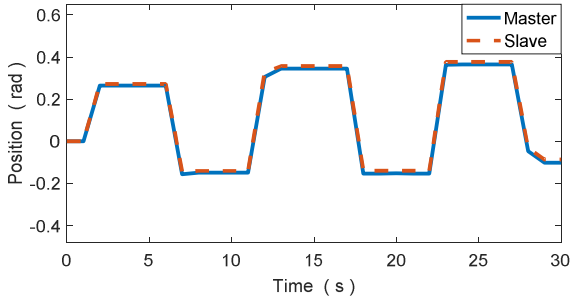

Fig. 6. Motion tracking behavior with PID controllers.

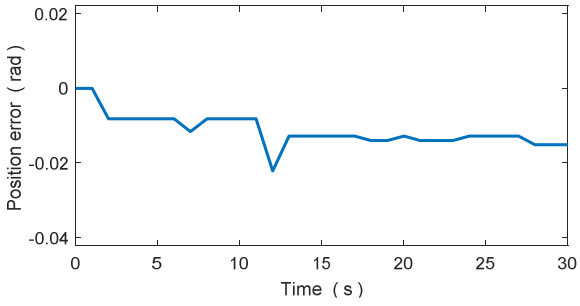

Fig. 7. Position error behavior with PID controllers.

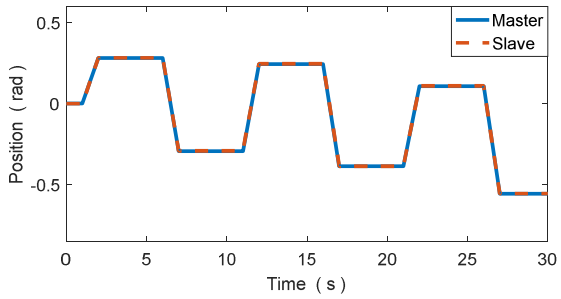

Fig. 8. Motion tracking behavior with ANFIS controllers.

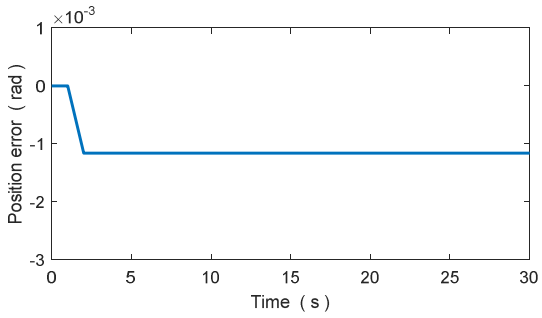

Fig. 9. Position error behavior with ANFIS controllers.

These results can be explained by the two ANFIS controllers designed in this work, which apply an intelligent control algorithm that can adapt to the variation dynamics of master and slave systems, and this by updating in a short time the consequence parameters of the neuro-fuzzy network, thanks to the FPGA's computing power and its high sampling frequency. In addition, the adopted design methodology, used in this paper to design the controllers, allowed us to have good results in our teleoperation system in terms of precision and stability. The MATLAB-Simulink environment with its Fixed-Point Tool and HDL Coder functionalities allowed us to design a robust algorithm with great precision by optimizing the execution time, the design frequencies, and the hardware resources used in the FPGA.

VII. CONCLUSION

In this paper, an FPGA implementation of an adaptive neuro-fuzzy control for a teleoperation system was presented. The purpose was to design a robust and adaptive controller to compensate for the non-linearity of master and slave robots, by combining fuzzy logic inference capabilities with the learning capabilities of neural networks, while taking advantage of the computing capacity of the FPGA and its high sampling rate.

The controllers were developed in MATLAB-Simulink environment and were implemented on FPGA using Simulink's fixed point tool and HDL Coder. Such a design methodology aims to create optimal VHDL code in terms of hardware resources and design time. The effectiveness of the control adopted has been demonstrated experimentally by a teleoperation system with one degree of freedom. Position tracking results showed that ANFIS controllers offer better performances than those obtained by conventional PID controllers.

REFERENCES

[1] B. Siciliano, and O. Khatib, Springer handbook of robotics. Springer Science & Business Media, 2008.

[2] I. Aliaga, A. Rubio, and E. Sanchez, "Experimental quantitative comparison of different control architectures for master-slave teleoperation," IEEE Transactions on Control Systems Technology, Vol. 12, no. 1, pp. 2-11, 2004.

[3] A. Hace, and M. Franc, "FPGA implementation of sliding-mode-control algorithm for scaled bilateral teleoperation," IEEE Transactions on Industrial Informatics, Vol. 9, no. 3, pp. 1291-1300, 2013.

[4] PF. Hokayem, and MW. Spong, "Bilateral teleoperation: An historical survey," Automatica, Vol. 242, no. 12, pp. 2035-2057, 2006.

[5] Letier P, Exosqulette haptic arm: Design and control, Doctorate thesis of Engineering Sciences, University of Brussels, 2010. [Bras exosqulette haptique: Conception et contrôle, Doctorat en Sciences de l'Ingénieur, Université Libre de Bruxelles (ULB). (In french.)]

[6] G. De Gersem, Kinaesthetic feedback and enhanced sensitivity in robotic endoscopic telesurgery, Doctorate Thesis on the katolic university of Leuven. Belgium, 2005.

[7] A. Alfi, and M. Farrokhi, "A simple structure for bilateral transparent teleoperation systems with time delay," Journal of Dynamic Systems, Measurement, and Control, Vol. 3, 2008.

[8] R. Mellah, S. Guermah, and R. Toumi, "Adaptive control of bilateral teleoperation system with compensatory neural-fuzzy controllers," International Journal of Control, Automation and Systems, Vol. 15, pp. 1-11, 2017.

[9] R. Mellah, and R. Toumi, "Control bilateral teleoperation by compensatory ANFIS," Advanced Mechatronics Solutions, Vol. 393, pp. 167-172, 2016.

[10] Y. Yang, C. Ge, H. Wang, X. Li, and C. Huaa, "Adaptive neural network based prescribed performance control for teleoperation system Under input saturation," Journal of the Franklin Institute, Vol. 352, pp. 1850-1866, 2015.

[11] S. Liu, X. Zhang, W. Zheng, and B. Yang, "Adaptive control for time-delay teleoperation systems with uncertain dynamics," Journal of Physics: Conf. Series 887. pp.1-8, 2017

[12] M. Franc, and A Hace, "A study on the FPGA implementation of the bilateral control algorithm towards haptic teleoperation," Journal for Control, Measurement, Electronics, Computing and Communications, Vol. 54, no. 1, pp. 49–61, 2013.

[13] E. Ishii, H. Nishi, and K. Ohnishi, "Improvement of performances in bilateral teleoperation by using FPGA," IEEE Transactions on Industrial Electronics, pp. 317- 322, 2007.

[14] E. Monmasson, L. Idkhajine, MN. Cirstea, I. Bahri, A. Tisan, and MW. Naouar, "FPGAs in industrial control applications," IEEE Transactions on Industrial Informatics, Vol. 7, no. 2, pp. 224–243, 2011.

[15] HDL coder user's guide, Mathworks. MATLAB & SIMULINK, r2018a, 2018.

[16] Y. Q. Zhang, and A. Kandel, " Compensatory neuro-fuzzy systems with fast learning algorithms," IEEE Transactions on Neural Networks, Vol. 9, no. 1, pp. 83-102, January 1998.

[17] JSR. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, no. 3, pp. 665-685, 1993.

[18] JSR. Jang, CT. Sun, and E. Mizutani, Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence. Prentice Hall. 1997.

[19] S. Ruder, " An overview of gradient descent optimization algorithms," Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin. pp.1-14, 2016.

[20] S. Hayki, Kalman filtering and neural networks, John Wiley & Sons, Inc; 2001.

[21] L. Peng, and PY. Woo, "Neural-fuzzy control System for Robotic Manipulators," IEEE Control Systems Magazing, pp. 53-63, February 2002.

[22] FL. Lewis, CT. Abdallah, and DM. Dawson, Control of robot manipulators, Macmillan, New York, 1993.

[23] A. Finnerty, and H. Ratigner, Reduce power and cost by converting from floating point to fixed point. XILINX All Programmable, WP491 (v1.0), 2017.

[24] ZedBoard (Zynq evaluation and development), Hardware user's guide, Version 2.2, 2014.