IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 May 2024, accepted 24 June 2024, date of publication 2 July 2024, date of current version 10 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3422015

RESEARCH ARTICLE

FPGA-Based Adaptive PID Controller Using MLP Neural Network for Tracking Motion Systems

VAN-QUANG-BINH NGO[1], **NGUYEN KIM ANH**[2], **AND NGUYEN KHANH QUANG**[2]
[1]Faculty of Physics, University of Education, Hue University, Thua Thien Hue 530000, Vietnam
[2]Faculty of Electrical Engineering, The University of Danang-University of Science and Technology, Da Nang 550000, Vietnam

Corresponding author: Nguyen Khanh Quang (nkquang@dut.udn.vn)

This work was supported by The University of Danang, University of Science and Technology, code number of Project: T2023-02-04.

ABSTRACT In this study, the Field Programmable Gate Array (FPGA) technology is employed to integrate multi-loop controllers for motion systems. To provide precise positioning and trajectory tracking for multi-axis systems, the proportional-integral (PI) control is used in the speed control loop and adaptive PID control in the position control loop. The motion system under consideration comprises an X-Y table driven by permanent magnet synchronous motors (PMSMs), and controlled by two programmable servo systems, each designed to regulate a separate axis. Each axis of this system consists of a motion planning module, a speed PI controller in the inner loop, and an adaptive PID position controller in the outer loop. The adaptive PID controller is specifically designed using a multilayer perceptron (MLP) neural network and parameter tuning methods. The control objective is to enhance trajectory tracking accuracy, especially in the presence of dynamic variations and uncertain disturbances. The Very High-speed IC Hardware Description Language (VHDL) is utilized to implement the desirable features of the control system. The control development is based on an FPGA device using Altera's Quartus II and Nios II software environment. The VHDL designs are analyzed and synthesized within this software environment. Simulation results demonstrate that the on-chip control system can achieve accurate positioning and tracking performance for the X-Y table motion.

INDEX TERMS FPGA, PMSM drives, MLP neural network, adaptive PID controller, X-Y table.

I. INTRODUCTION

Robotic systems are increasingly used in contemporary industrial equipment assembly lines to replace human labor, particularly in those tasks that demand high performance or involve risky activities such that accurate positioning and trajectory tracking are required. However, the operational capabilities of most automated manipulators and parallel robots are often limited by their restricted range of motion and constrained operating abilities, so impeding the flexibility and precision of the production line. To address this challenge, it is essential to use advanced multiple axis control systems in the assembly industry. A typical system for that is an X-Y table driven by PMSM motors, wherein servo controllers are used to regulate the position and assure accurate trajectory tracking. Nevertheless, the movement of the X-Y table is often influenced by unmodelled dynamics, external load disturbances, and interactions across axes, causing unavoidable inaccuracy. In computer control, recent progresses have built on programmable on-chip controllers, with the development of multi-axis motion control systems that are small, flexible, and cost-effective. The FPGA technology is widely recognized for its programmable features, strong connectivity, quick time-to-market, efficient design cycles, embedded processing capabilities, low power consumption, and high density. It is particularly well-suited for digital signal processing applications, such as motion control systems and robots [1], [2], [3], [4], [5], [6], [7], [8], [9]. When developing FPGA-based systems, it is important to minimize the use of logic components and resource consumption. Therefore, intelligent control approaches are preferred over hard methods for on-chip control design. Despite their widespread usage in industrial processes, traditional PID controllers often exhibit a difficulty in

The associate editor coordinating the review of this manuscript and approving it for publication was Zhuang Xu.

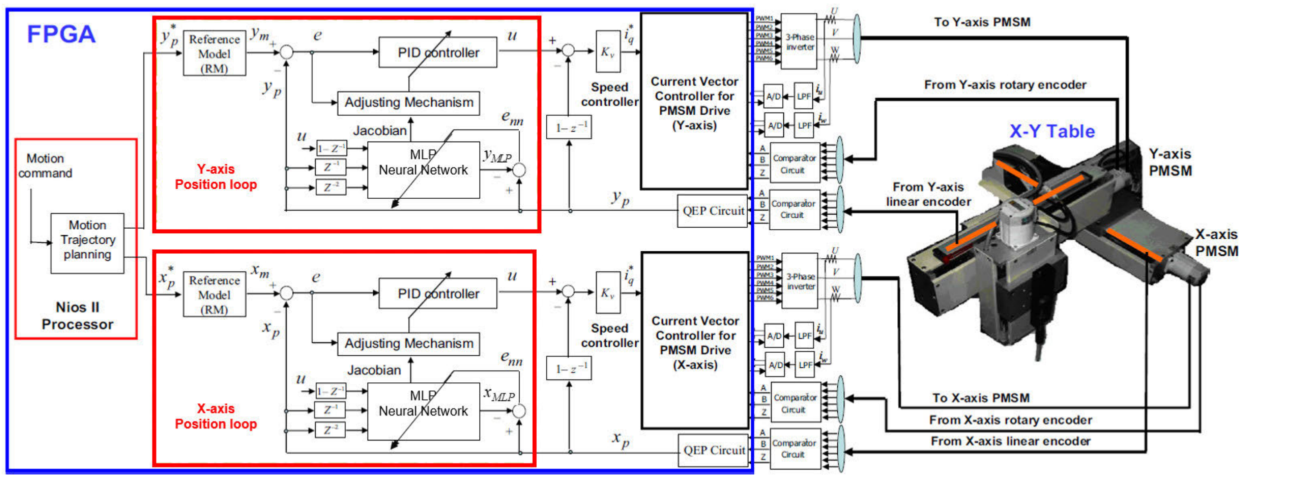


FIGURE 1. FPGA-based motion control system.

adapting to dynamic or uncertain systems owing to their set parameters [2], [4]. In order to tackle this problem, many sophisticated control strategies, including neural network control, adaptive fuzzy control, and adaptive PID control, have been devised to improve system performance [2], [4], [5], [6], [7], [10], [11], [12], [13]. Nevertheless, the implementation of these intricate control algorithms, such as neural networks, fuzzy controllers, or adaptive PID control, necessitates substantial computational resources. As a result, they are commonly implemented using fixed-point DSP and FPGA in the majority of research studies [10], [12], [13], [14], [15], [16], [17], [18], [19]. This article presents a method to improve the performance and applicability of positioning and tracking in motion control systems. It introduces the development of fine-tuning schemes for the position control design using adaptive PID control. The motion performance of the control system is tested, along with its FPGA-based execution capability. In order to maximize the efficiency of resource usage, a finite state machine is constructed, which incorporates operations such as multiplication, addition, comparison, registers, and a lookup table (LUT). The FPGA-in-the-loop simulation is conducted by using Quartus II and Nios II software environment on a single FPGA chip. The Nios II integrated development environment is used for generating motion trajectories, while the Quartus II runs VHDL code to represent the system dynamics. The simulation results validate the effectiveness of the proposed on-chip control system for positioning and tracking in executing window shape motion trajectories [11], [12], [13], [18], [19], [20], [21], [22].

II. DESCRIPTION AND DESIGN OF CONTROL SYSTEM

A. MODEL OF PMSM DRIVES

The thrust force T_e of the driving PMSM under field-oriented control (with $i_d = 0$) can be expressed as [22]:

$$T_e = K_t i_q, \tag{1}$$

and its motion equation is

$$T_e - T_L = J_m \frac{2\pi}{r}\frac{d^2 x_p}{dt^2} + B_m \frac{2\pi}{r}\frac{dx_p}{dt}, \tag{2}$$

where i_d and i_q are the motor's dq frame currents, K_t, J_m, B_m and J_t are the torque constant, inertial value, damping ratio, and external load torque.

Fig. 1 shows the axis control system designed to handle X and Y coordinates. All components of this system, including the motion planner, reference model, speed PI controller, and position neural PID controller with tuning mechanism, are integrated into a single FPGA chip.

B. DESCRIPTION AND DESIGN OF CONTROL SYSTEM

1) REFERENCE MODEL

In control systems, the reference model (RM) often takes the form of a second-order system. Therefore, the transfer function of the RM for the motion system shown in Fig. 1 is expressed as:

$$\frac{x_m(s)}{x_p^*(s)} = \frac{\omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2} \tag{3}$$

where ω_n is the natural frequency, ς is the damping ratio, x_m is the output of RM and x_p^* is the position command.

Subsequently, the use of the bilinear transformation allows for the conversion of (3) into a discrete-time model in the z-domain as,

$$\frac{x_m(z^{-1})}{x_p(z^{-1})} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}} \tag{4}$$

and the difference equation is expressed as

$$\begin{aligned} x_m(k) = &-b_1(k-1) - b_2(k-2) + a_0 x_p^*(k) \\ &+ a_1 x_p^*(k-1) + a_2 x_p^*(k-2) \end{aligned} \tag{5}$$

2) PID CONTROLLER DESIGN

The discrete-time form - representing the output of a PID controller is

$$u(k) = K_p e(k) + K_i \sum_{k=1}^{n} e(k) + K_d[e(k) - e(k-1)], \quad (6)$$

where the definition of the error is

$$e(k) = x_{ref}(k) - x_p(k) \quad (7)$$

and K_p, K_i, K_d are the gains of the PID controller.

3) MLP NEURAL NETWORK

The utilized MLP neural network in this study follows a three-layer structure, as illustrated in Fig. 2. It is composed of an input layer, a hidden layer, and an output layer. The MLP neural network consists of three input nodes as $\Delta u(k)$, $x_p(k-1)$ and $x_p(k-2)$, and its vector form is represented by

$$X = [\Delta u(k), x_p(k-1), x_p(k-2)]^T, \quad (8)$$

with $\Delta u(k) = u(k) - u(k-1)$. Additionally, in the hidden layer of the MLP NN, the sigmoid function serves as the activation function. The following is how it is formulated:

$$h_j = \frac{1}{1 + \exp\left(\sum_{i=1}^{3} c_{ij} X(i)\right)}, \; j = 1, 2, 3 \ldots m \quad (9)$$

and the output of neural network in Fig. 2 can be expressed

$$x_{MLP} = \sum_{j=1}^{m} w_j h_j \quad (10)$$

where w_j and h_j represents the weight and output of j^{th} neuron, respectively. In the MLP neural network, the goal of the learning algorithm is to determine the most appropriate cost function, aiming to minimize the designated error between the network output and the desired output. This learning algorithm, based on the method of gradient descent, provides faster computational capability compared to alternative methods, thus offering an obvious advantage. This approach involves training the MLP NN by adjusting its parameters, including the weights in the backpropagation function, to achieve favorable convergence. In the process of supervised learning based on gradually decreasing gradients, the cost function is defined as:

$$J = \frac{1}{2}(x_p - x_{MLP})^2 \triangleq \frac{1}{2} e_{nn}{}^2 \quad (11)$$

The gradient descent approach is used to develop the learning algorithm for determining the weights in the hidden layer and the output layer:

$$c_{ij}(k+1) = c_{ij}(k) + \eta e_{nn} h_j (1 - h_j) x_i(k) \quad (12)$$

$$w_j(k+1) = w_j(k) + \eta e_{nn} \sum_{j=1}^{m} w_j(k) h_j (1 - h_j) x_i(k) \quad (13)$$

where $j = 1, 2, 3 \ldots m$, $i = 1, 2, 3$ and η is a learning rate.

4) SEFT-TUNING PID CONTROLLER

By defining an instantaneous cost function

$$E(k) = \frac{1}{2} e^2(k) = \frac{1}{2}(x_{ref} - x_p)^2 \quad (14)$$

the gains of the PID controller can then be found using the gradient descent method, as shown below.

$$\Delta K_p = -\alpha \frac{\partial E}{\partial K_p} = \alpha e(k) \frac{\partial x_p}{\partial \Delta u} (e(k) - e(k-1)) \quad (15)$$

$$\Delta K_i = -\alpha \frac{\partial E}{\partial K_i} = \alpha e(k) \frac{\partial x_p}{\partial \Delta u} e(k) \quad (16)$$

$$\Delta K_d = -\alpha \frac{\partial E}{\partial K_d} = \alpha e(k) \frac{\partial x_p}{\partial \Delta u} (e(k) - 2e(k-1) - e(k-2)) \quad (17)$$

where α represents the learning rate and $\frac{\partial x_p}{\partial \Delta u}$ denotes the Jacobian transformation of the MLP neural network:

$$\frac{\partial x_p}{\partial \Delta u} \approx \frac{\partial x_{MLP}}{\partial \Delta u} = \sum_{j=1}^{m} w_j(k) h_j (1 - h_j) c_{1j} \quad (18)$$

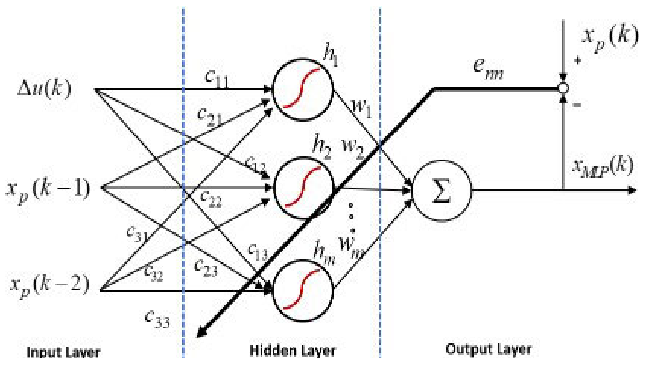


FIGURE 2. Architecture of an MLP neural network.

III. TRAJECTORY PLANNING

The trajectory planner, seen in Fig. 1, can be designed to create the model references for both axes, denoted as $x_m(k)$ and $y_m(k)$.

A. TRAJECTORY OF WINDOW FORM

Fig. 3 illustrates a trajectory representing a window as given in [6]. This trajectory is often divided into nine parts, designated as *a*- to *i*-segment. The window shape trajectory, originating at a position on the negative side of the X-axis, and the values of step S, angular increment $\Delta\theta$ are consistent throughout all four circular corners $\theta_i = \theta_{i-1} + \Delta\theta$, and their values depend on the size of the trajectory.

B. MEASURES OF TRAJECTORY PERFORMANCE

For motion tracking, the average and standard deviation of the tracking errors are computed to assess the performance of

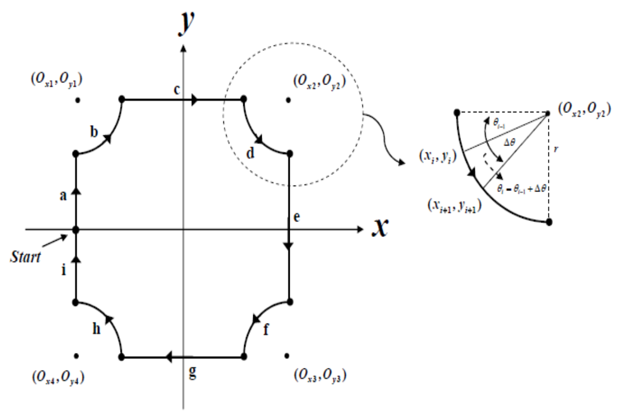
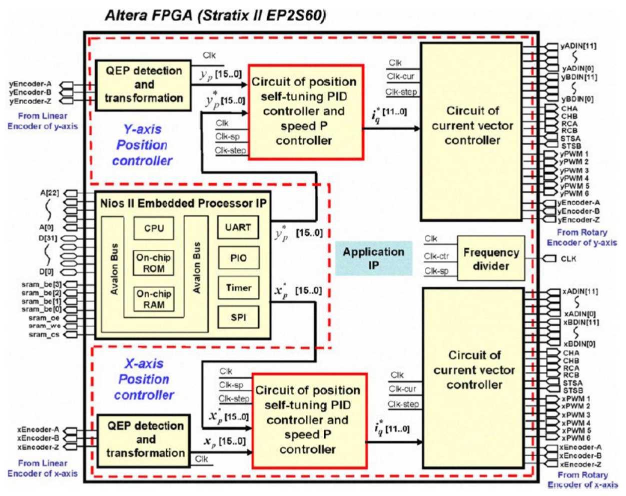


FIGURE 3. Window shape motion trajectory.



Altera FPGA (Stratix II EP2S60)
QEP detection and transformation
Circuit of position self-tuning PID controller and speed P controller
Circuit of current vector controller
Y-axis Position controller
X-axis Position controller
Nios II Embedded Processor IP
CPU
On-chip ROM
On-chip RAM
Avalon Bus
UART
PIO
Timer
SPI
Application IP
Frequency divider
From Linear Encoder of y-axis
From Linear Encoder of x-axis
From Rotary Encoder of y-axis
From Rotary Encoder of x-axis
Clk
Clk-cur
Clk-step
Clk-sp
Clk-ctr
CLK

FIGURE 4. The circuit design of the proposed control IC.

various controllers. These metrics are described as

$$\overline{e_{tr}} = \sum_{k=1}^{N} \frac{e_{tr}(k)}{N} \tag{19}$$

$$\sigma_{tr} = \sqrt{\sum_{k=1}^{N} \frac{(e_{tr}(k) - \overline{e_{tr}})^2}{N}}, \tag{20}$$

where N represents the total number of sampling periods for a trajectory cycle and $e_{tr}(k) = \sqrt{e_x^2(k) + e_y^2(k)}$ is the tracking error distance in an X-Y coordination frame.

IV. FPGA IMPLEMENTATION OF CONTROLLER

A. FPGA ARCHITECTURE DESIGN

Fig. 4 depicts the proposed architecture of a motion controller for the X-Y table, constructed utilizing FPGA technology. The FPGA used is the Altera Cyclone II EP2C70, which has a Nios II embedded CPU to provide a programming environment inside the FPGA. This FPGA board is equipped with 68,416 Adaptive Look-Up Tables (ALUTs), a total of 1,152,000 bits of Random Access Memory (RAM), and a customizable 32-bit CPU core Nios II processor. The configuration shown in Fig. 4 comprises a Nios II embedded processor IP and an application IP, which is specifically designed to generate the motion trajectory and gather response data. The IP application incorporates circuits for the position adaptive PID controller, the speed PI controller, and the current vector controller for both the X and Y axes. The position control loop is set to run at a sampling frequency of 2 KHz. A frequency divider generates clock signals with frequencies of 50 MHz (Clk), 12.5 MHz (Clk-sp), and 500 Hz (Clk-ctr) to supply all circuits.

B. FPGA ARCHITECTURE DESIGN

The control system shown in Fig. 1 is implemented in the FPGA. The generation of motion trajectories is performed by software incorporating the Nios II embedded processor. A finite state machine (FSM) is used to define the control algorithm for both the position neural PID controller and the speed PI controller. With the exception of the 24-bit data type used in the reference model, all other data have a length of 16 bits. Along with the PI controller and the more complex neural PID controller, the FSM effectively covers the system dynamics and the whole control system can be adequately represented using VHDL. The adaptive PID controller requires the execution of 90 steps. Each step operation inside the FPGA may be accomplished in 80 *ns*, resulting in a total operating time of 7.2 μs for the adaptive PID controller. The digital hardware implementation of the activation function in Equation (9) is quite complicated since it has to perform an exponential function. In order to tackle this issue, a combination of the Taylor series expansion method and the look-up table (LUT) methodology is used. The polynomial equation of the fifth order in the neighborhood of x_0 is extended as follows

$$e^{-x} \cong a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + a_3(x-x_0)^3 + a_4(x-x_0)^4 + a_5(x-x_0)^5, \tag{21}$$

where $a_0 = e^{-x_0}$, $a_1 = -e^{-x_0}$, $a_2 = e^{-x_0}/2$, $a_3 = -e^{-x_0}/6$, $a_4 = e^{-x_0}/24$, $a_5 = -e^{-x_0}/120$.

Fig. 5 depicts the internal circuit design of the sigmoid function in (9). The design of the sigmoid function in (9) which executes the overall computation utilizing seven look-up tables, two adders, two multipliers, one divider, and 14-steps machine operation. It requires 112 *ns* for computation. The VHDL code is transmitted directly to the FPGA board subsequent to the completion of hardware/software co-design. In the meantime, the software code is placed onto the external SDRAM.

V. RESULTS AND DISCUSSION

Simulation work is performed utilizing the Quartus II and Nios II IDE, and the response results are collected. Following this, MATLAB is used to perform analysis and visualization. The proposed controller for tracking motion control system is validated for accuracy and effectiveness through

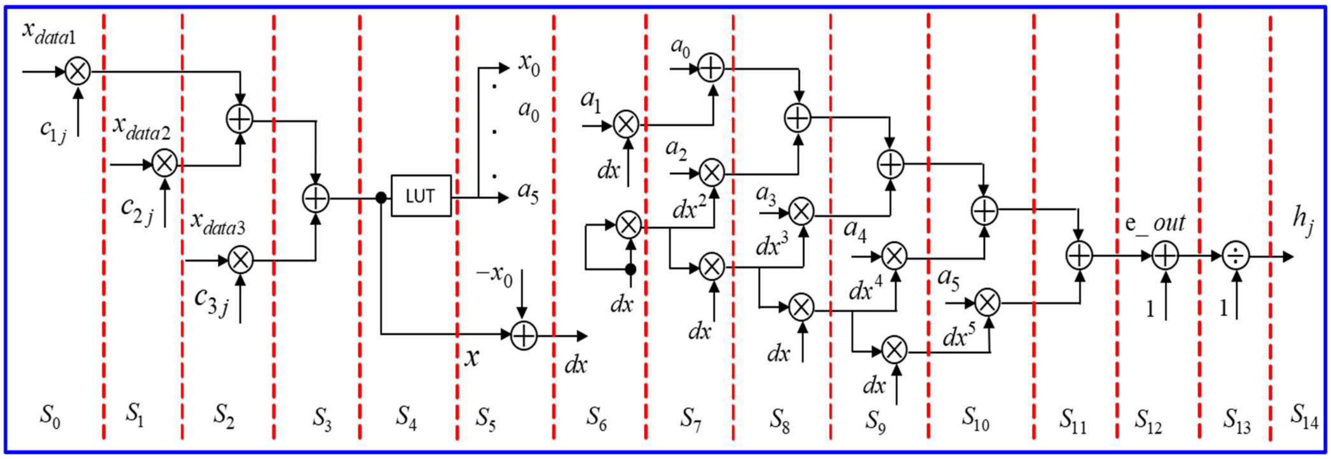


FIGURE 5. FSM design of the sigmoid function.

TABLE 1. Comparison results between PID and proposed controller.

	Average tracking error	Tracking error standard deviation	FPGA resource usage	
	$\overline{e_{tr}}(mm)$	$\sigma_{tr}(mm)$	ALUTs	(RAM bits)
PID (case 1)	0.58	0.029	28602	44542
PID (case 2)	1.25	0.068		
Proposed method (case 2)	0.32	0.014	36504	47806

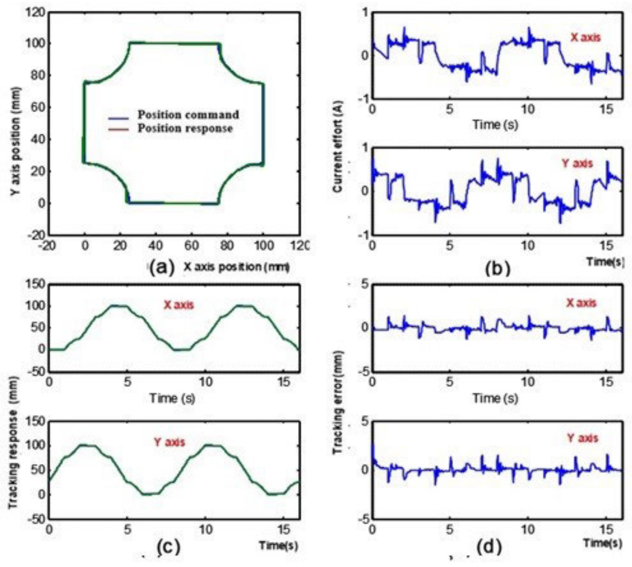


FIGURE 6. Window trajectory response by using conventional PID for Case 1: (a) Window tracking (b) Position tracking (c) Control efforts (d) Tracking errors in X- and Y-axis.

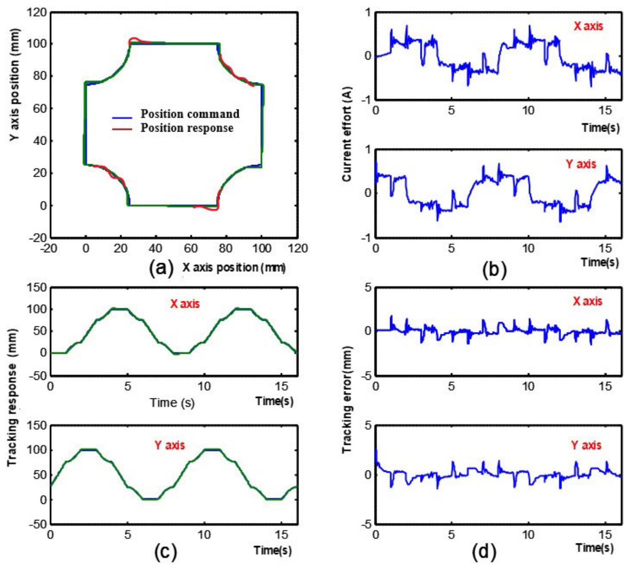


FIGURE 7. Window trajectory response by using conventional PID for Case 2: (a) Window tracking (b) Position tracking (c) Control efforts (d) Tracking errors in X- and Y-axis.

hardware-in-the-loop simulation. Performance evaluation is conducted by examining the movement paths of the window in two different scenarios: Case 1, representing normal load condition ($J_m = 0.000108$ kg.m^2, $B_m = 0.0013\ N.m.s$), and Case 2, simulating heavy load condition ($J_m = 0.000108 \times 3$ kg.m^2, $B_m = 0.0013 \times 3\ N.m.s$). The respective results are depicted in Figs. 6, 7, and 8. Initially designed under normal conditions with fixed PID control, the X-Y table exhibits satisfactory tracking responses, as illustrated in Fig. 6. However, when subjected to parameter variations in Case 2, deviations occur in the bends and corners of the trajectory, as evident in Fig. 7. To address this issue, a self-tuning PID controller employing an MLP neural network is proposed, yielding significantly reduced tracking errors thanks to the

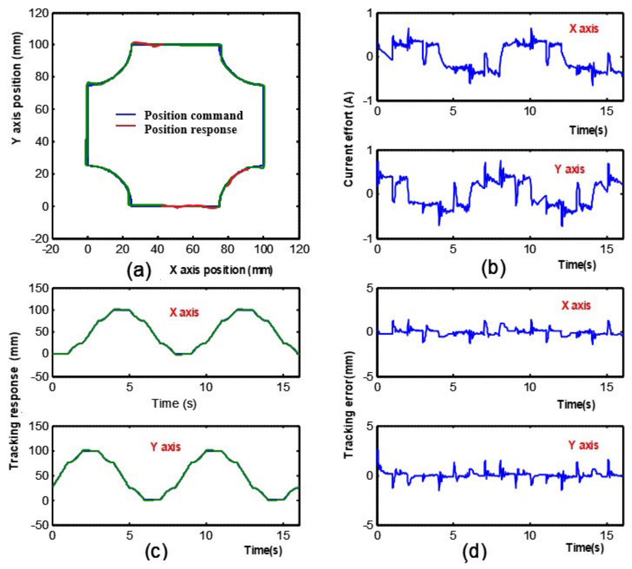


FIGURE 8. Window trajectory response by using proposed PID for Case 2: (a) Window tracking (b) Position tracking (c) Control efforts (d) Tracking errors in X- and Y-axis.

PID parameters are tuned to an adequate value, as showcased in Fig. 8. The initial gains of PID control are chosen by $K_p = 5$, $K_i = 0.06$, $K_d = 1.01$, but during the course these gains are eventually tuned to $K_p = 4.8$, $K_i = 0.7$, $K_d = 1.036$. Consequently, tracking performance can be substantially improved as can be seen in Fig. 8.

A comparison between conventional PID and the proposed adaptive PID controller is presented in Table 1, including resource usage data. The proposed method demonstrates performance enhancements in terms of significantly reduced average tracking error and standard deviation, despite a slight increase in resource consumption. These findings underscore the feasibility and effectiveness of the proposed approach for X-Y table position control using the FPGA technology.

VI. CONCLUSION

This work has presented the FPGA technology to design and implement a high performance motion control system. The main goal of the control system is to provide precise positioning and tracking of the desired trajectory of a X-Y table. Its drive system includes PMSM motors under field-oriented vector control to facilitate the regulation. The design for the control system comprises a motion planning module, a PI speed control, and a self-tuning neural PID position control. The Quartus and Nios II environment is used to execute hardware simulation inside the loop, which showcases successfully the system tracking performance. By using this programmable chip architecture, it can offer an effective solution to the requirements of high control performance and also energy efficiency. Hence, the proposed technique is promising for diverse automation applications that need multi-axis motion tracking.

REFERENCES

[1] J. Qin and J. Du, "Robust adaptive asymptotic trajectory tracking control for underactuated surface vessels subject to unknown dynamics and input saturation," *J. Mar. Sci. Technol.*, vol. 27, no. 1, pp. 307–319, Mar. 2022.

[2] S. Sanchez-Solano, A. J. Cabrera, I. Baturone, F. J. Moreno-Velo, and M. Brox, "FPGA implementation of embedded fuzzy controllers for robotic applications," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1937–1945, Aug. 2007.

[3] F.-J. Lin, L.-T. Teng, Y.-C. Hung, and C.-Y. Chen, "FPGA-based adaptive backstepping control system using RBFN for linear induction motor drive," *IET Electric Power Appl.*, vol. 2, no. 6, pp. 325–340, Nov. 2008.

[4] C. Yi-fei, X. Sen, C. Rui, and Z. Tian, "The study and simulation of PID control based on RBF neural network," in *Proc. Int. Conf. Electron. Mech. Eng. Inf. Technol.*, vol. 7, Aug. 2011, pp. 3453–3456.

[5] K. A. Abuhasel, F. F. M. El-Sousy, M. F. El-Naggar, and A. Abu-Siada, "Adaptive RCMAC neural network dynamic surface control for permanent-magnet synchronous motors driven two-axis X-Y table," *IEEE Access*, vol. 7, pp. 38068–38084, 2019.

[6] N. K. Quang, Y.-S. Kung, and Q. P. Ha, "FPGA-based control architecture integration for multiple-axis tracking motion systems," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, vol. 7, Dec. 2011, pp. 591–596.

[7] F. F. M. El-Sousy and K. A. Abuhasel, "Nonlinear robust optimal control via adaptive dynamic programming of permanent-magnet linear synchronous motor drive for uncertain two-axis motion control system," *IEEE Trans. Ind. Appl.*, vol. 56, no. 2, pp. 1940–1952, Mar. 2020.

[8] H. Tanaka, K. Ohnishi, H. Nishi, T. Kawai, Y. Morikawa, S. Ozawa, and T. Furukawa, "Implementation of bilateral control system based on acceleration control using FPGA for multi-DOF haptic endoscopic surgery robot," *IEEE Trans. Ind. Electron.*, vol. 56, no. 3, pp. 618–627, Mar. 2009.

[9] Q. N. Khanh, N. D. That, Q. N. Hong, and Q. P. Ha, "FPGA-based fuzzy sliding mode control for sensorless PMSM drive," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2012, pp. 172–177.

[10] S. Jung and S. S. Kim, "Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 265–271, Feb. 2007.

[11] F.-J. Lin and P.-H. Shen, "Robust fuzzy neural network sliding-mode control for two-axis motion control system," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1209–1225, Jun. 2006.

[12] B. Tufekci, B. Onal, H. Dere, and H. F. Ugurdag, "Efficient FPGA implementation of field oriented control for 3-phase machine drives," in *Proc. IEEE East-West Design Test Symp. (EWDTS)*, Sep. 2020, pp. 1–5.

[13] Y.-S. Kung, C.-C. Huang, and M.-H. Tsai, "FPGA realization of an adaptive fuzzy controller for PMLSM drive," *IEEE Trans. Ind. Electron.*, vol. 56, no. 8, pp. 2923–2932, Aug. 2009.

[14] F. Song, W. Xu, and J. Wang, "Design and research of automatic generation control code for dual three-phase PMSM based on FPGA," in *Proc. 26th Int. Conf. Electr. Mach. Syst. (ICEMS)*, Nov. 2023, pp. 1917–1921.

[15] A. Saware, N. Gavkar, S. Shinde, and P. Kulkarni, "Performance improvement of PID controller for PMSM using ANFIS controller," in *Proc. Int. Conf. Intell. Controller Comput. for Smart Power (ICICCSP)*, Jul. 2022, pp. 1–6.

[16] Y. Lu, J. Huang, Z. Jiang, T. Tang, H. Tang, and L. Shi, "PID adaptive feedback motor system based on neural network," *IEEE Access*, vol. 12, pp. 60149–60154, 2024.

[17] W. Tu, G. Luo, Z. Chen, C. Liu, and L. Cui, "FPGA implementation of predictive cascaded speed and current control of PMSM drives with Two-Time-Scale optimization," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 5276–5288, Sep. 2019.

[18] Q. P. Ha, Y. H. Yu, and N. K. Quang, "FPGA-based cooperative control of indoor multiple robots," *Int. J. Adv. Mech. Syst.*, vol. 4, no. 5, p. 248, 2012.

[19] B. Wang, J. Jiao, and Z. Xue, "Implementation of continuous control set model predictive control method for PMSM on FPGA," *IEEE Access*, vol. 11, pp. 12414–12425, 2023.

[20] N. K. Quang, D. D. Tung, and Q. P. Ha, "FPGA-based sensorless PMSM speed control using adaptive extended Kalman filter," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 1650–1655.

[21] Z. Ma and X. Zhang, "FPGA implementation of sensorless sliding mode observer with a novel rotation direction detection for PMSM drives," *IEEE Access*, vol. 6, pp. 55528–55536, 2018.

[22] N. K. Quang, N. T. Hieu, and Q. P. Ha, "FPGA-based sensorless PMSM speed control using reduced-order extended Kalman filters," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6574–6582, Dec. 2014.

VAN-QUANG-BINH NGO received the M.Sc. degree in automation from Da Nang University of Science and Technology, Vietnam, in 2009, and the Ph.D. degree in automation from CentraleSupelec-Paris Saclay University, France, in 2017. He was appointed a Lecturer with Hue University of Education, Vietnam, in 2004, and promoted to an Assistant Professor, in 2017. He was a Postdoctoral Researcher with the Department of Electrical Engineering, Chonnam National University, South Korea, in 2018. His research interests include multilevel converters topology, predictive control for power converters, and electrical drives and their applications in renewable energy systems.

NGUYEN KIM ANH received the Engineer's degree in electrical engineering from the University of Science and Technology-The University of Danang, in 2004, the M.S. degree in control engineering and automation from The University of Danang, in 2009, and the Ph.D. degree in systems optimization and dependability from Troyes University of Technology, France, in 2015. He is currently a Lecture with the University of Science and Technology-The University of Danang, Vietnam, where he held the Head of Automation Division Faculty of Electrical Engineering, from 2009 to 2011. His current research interests include stochastic modeling of systems deterioration, prognostic and health management techniques, maintenance and inventory decision-making, optimization, power electronics, the industrial communication networks (IIoT, SCADA), and FPGA and control systems.

NGUYEN KHANH QUANG received the Ph.D. degree from the School of Electrical, Mechanical and Mechatronic System, University of Technology Sydney, Australia, in 2015. He is currently a Senior Lecturer with the Faculty of Electrical Engineering, University of Science and Technology-The University of Danang, Vietnam. His research interests include automation, industry applications, motion control, and FPGA-based motor drives applications.