TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 18, No. 6, December 2020, pp. 3346~3356
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v18i1.17277

Co-simulation of self-adjusting fuzzy PI controller for the robot with two-axes system

Nguyen Vu Quynh, Pham Van Toan
Department of Electrical and Electronics, Lac Hong University, Vietnam

Article Info

Article history:

Received May 12, 2020
Revised Jun 6, 2020
Accepted Jul 20, 2020

Keywords:

FPGA
Fuzzy PI controller
Radial basis function neural network
Simulation
XY-axis

ABSTRACT

This paper presents the co-simulation of the self-adjusting fuzzy PI controller to control a two-axes system. Each axis was driven by a permanent magnet linear synchronous motor (PMLSM). The position and speed controller used the fuzzy PI algorithm with parameters adjusted by a radial basis function neural network (RBFNN). The vector control was applied to the decoupled effect of the PMLSM. The field programmable gate array (FPGA) was used to control both axes of the system. The very high-speed integrated circuit-hardware description language (VHDL) was developed in the Quartus II software environment, provided by Altera, to analyze and synthesize designs. Firstly, the mathematical model of PMLSM and fuzzy PI was introduced. Secondly, the RBFNN adjusted the knowledge base of the fuzzy PI. Thirdly, the motion trajectory was introduced for testing the control algorithm. Fourthly, the implementation of the controller based on FPGA with the FSM method and the structure of co-simulation between Matlab/Simulink and ModelSim were set up. Finally, discussion about the results proved the effectiveness of the control system, determining the exact position and trajectory of the XY axis system. This research was successful in implementing a two-motor controller within one chip.

Corresponding Author:

Nguyen Vu Quynh,
Department of Electrical and Electronics,
Lac Hong University,
10 Huynh Van Nghe Street, Buu Long Ward, Bien Hoa City, Dong Nai Province, Vietnam.
Email: vuquynh@lhu.edu.vn

1. INTRODUCTION

Nowadays, robots are increasingly used to replace people in repetitive work or dangerous jobs. Robots are operated on a particular trajectory. However, most robots have a limited operating space, which affects the flexibility and accuracy of their actions. Many types of research for the multi-axis controller were designed to resolve these problems. Typically for this control method are two XY axes, driven by permanent magnet linear synchronous motor (PMLSM). The design of a specialized IC to make the controller compact, flexible, and low-cost has been studied by many experts to control the two XY-axes system. The two-axis XY, each axis has a PMLSM, the moving mechanism uses a ball screw nut. The contact between screw and screw nut is a layer of steel balls bearings to minimize friction, which helps smooth movement and achieve high accuracy.

Field programmable gate array (FPGA) technology has also been used in this case, with the hardware description programming feature, fast response time, short design cycles, embedded processors, low power consumption. The paper [1] designed the current vector control by Matlab/Simulink then convert it into Verilog hardware description language (HDL) code to speed up the calculation and attain a fast response. The vector

control for permanent magnet synchronous motor (PMSM) was coding by very high-speed integrated circuit-hardware description language (VHDL) in this research [2]. The paper [3] is using pipeline and resource sharing methods of FPGA to implement the sensorless PMSM drives. The BLDC was controlled by fuzzy sliding mode based on FPGA of Xilinx [4]. There were many types of research on intelligent controllers such as fuzzy control, sliding fuzzy control, artificial intelligence, adaptive control to control the speed and position of the motor for improving precision in machining and assembly. The fuzzy controller was applied to control the tennis ball robot for better training [5], modified direct torque control gives better performance of surface-mounted PMSM [6]. The scalar and field-oriented control techniques controlled the three-phase induction motor drive [7]. The fuzzy and sensorless methods were used to increase the exact speed of PMSM without encoder [8]. A sliding-mode fuzzy controller was used to strengthen the dynamic performance for the discrete-time uncertain system [9]. The back-propagation and radial basis function networks were used for PMSM to get the quick parallel speed and high torque response [10]. The adaptive fuzzy controls the rotor field-oriented of surface mount PMSM, speedy and precise control can be achieved [11]. The adaptive controller with simple estimator equations and the absence of the voltage probe depends on direct and quadrature reference current only [12]. The paper [13] showed an adaptive fuzzy supervisor controller for the PMSM sensorless control. The fuzzy control method is an intelligent control system, with an inference mechanism from experts' control experience, but obtaining fuzzy sets and the optimal membership functions are not easy. To resolve this problem, the PID gain or the fuzzy knowledge base were adjusted appropriately based on the feedback system parameter. In this paper, an adaptive fuzzy controller is implemented in DSP to cope with the dynamic uncertainty and external load effect to the PMLSM [14]. The neural network can perform real-time control of the back-propagation learning algorithm [15-17]. The adaptive filter and fuzzy logic controller are applied for torque ripple minimization [18]. The fuzzy logic controller and genetic algorithm are designed to reduce the selected time for the optimized error signal gain values and, as a result, enhance the controller and system performance [19]. This research designed a non-linear controller and observer for a PMSM drive [20]. The co-simulation work by electronic design automation (EDA) Simulator Link has been gradually applied to verify the effectiveness of the Verilog and VHDL code in the motor drive system [21-24]. The EDA Simulator Link [25] provides a co-simulation interface between MATLAB/Simulink and HDL simulators-ModelSim [26]. Using it, you can verify a VHDL, Verilog, or mixed-language implementation against your Simulink model or MATLAB algorithm [25]. Therefore, EDA Simulator Link enables the use of MATLAB code and Simulink models as a test bench that generates stimulus for an HDL simulation and analyzes the simulation's response [25].

This research focuses on improving the control method by using the self-adjusting fuzzy PI controller to control a two-axes system. The controller of both X and Y-axis is implemented on one chip. This research also presents a mixed-method with parallel and sequential processing to implement the self-adjusting fuzzy PI controller. The radial basis function neural network (RBFNN) has three neural networks; they are computed in an identical manner. The others are calculated in a sequential manner. The co-simulation method is applied to check the result.

2. DESIGN OF THE SELF-ADJUSTING FUZZY PI CONTROLLER

The control diagram for the XY system is shown in Figure 1. It consists of two loops, in which the current controller is an inner loop; the position and speed controller is an external loop. The external loop uses RBFNN to adjust the parameter for the fuzzy PI controller. The VHDL code programmed the whole structure of the controllers. RBFNN has three layers, the input layer, the hidden layer, which is a set of Gaussian functions, and the output layer is a set of total linear functions. There are two phases of network training. Firstly, the weight functions were calculated from the input layer to the hidden layer $u(k), x_p(k-1), x_p(k-2)$; then the weights from the hidden layer to the output layer were calculated to adjust the $c_{j,i}$ values of the fuzzy PI controller to make the fuzzy PI controller more accurate. Each axis has a controller, so the system consists of two self-tuning controllers.

2.1. Mathematical model of PMLSM

The torque of motor F_e under the effect of field-oriented control (FOC) ($i_d = 0$) is:

$$F_e = K_t i_q \tag{1}$$

motion equation of motor:

$$F_e - F_L = M_m \frac{d^2 x_p}{dt^2} + B_m \frac{dx_p}{dt} \tag{2}$$

with i_d and i_q as the motor's current on the d-q axis, K_t, M_m, B_m and F_L are the motor's coefficients, the total weight of the rotor, the coefficient of friction, and the load of the motor, respectively.

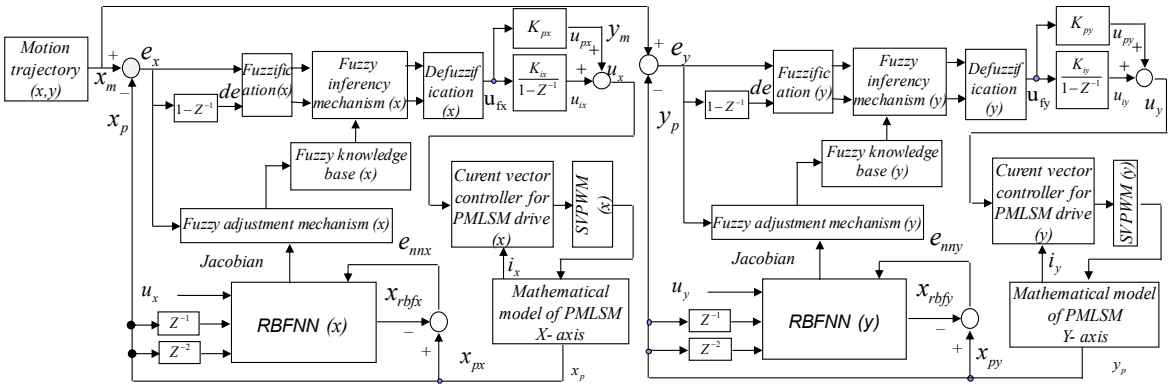

Figure 1. Block diagram of self-adjusting fuzzy PI controller to control the PMLSM of both X and Y-axes

2.2. Fuzzy PI controller design

The fuzzy PI controller's input has two values: the difference (E) between the desired position x_m and current position x_p of the motor on the X or Y-axis, which returned from the encoder and which is derivative by the time (dE):

$$E_x(k) = x_m(k) - x_p(k) \tag{3}$$

$$dE_x(k) = e(k) - e(k-1) \tag{4}$$

This equation controlled the position of the motor:

$$u_x(k) = u_{ix}(k-1) + \left(k_{px} + k_{ix}\frac{T}{2}\right)u_{fx}(k) + k_{ix}\frac{T}{2}u_{fx}(k-1) \tag{5}$$

where $k_{px}\ and\ k_{ix}$ are the coefficients of the PI controller at the external loop, U_{ix} is the output of the integral controller (I), T is the sampling time, and u_{fx} is the output of the fuzzy PI controller. The fuzzy PI controller can be summarized as follows:

The membership functions were the symmetrical triangles to calculate the input variables E and dE_x. During each cycle, only two values of E_x and dE_x were supplied to the system as shown in Figure 2. From the values of E_x and dE_x, based on membership functions, we can calculate the values of $\mu_{A_i}(E_x)\ and\ \mu_{B_i}(dE_x)$ with:

$$\mu_{A_i}(E_x) = \frac{E_{x,i+1} - E_x}{2}, \tag{6}$$

$$\mu_{A_{i+1}}(E_x) = 1 - \mu_{A_i}(E_x) \tag{7}$$

$$\mu_{B_i}(dE_x) = \frac{dE_{x,i+1} - dE_x}{2}, \tag{8}$$

$$\mu_{B_{i+1}}(dE_x) = 1 - \mu_{B_i}(dE_x) \tag{9}$$

Where A_i is the linguistic value of E_x, B_i is the linguistic value of dE_x. The fuzzy rules described as follows:

IF $E_x = A_i$ and $dE_x = B_i$ THEN $u_{fx} = c_{j,i}$ (10)

where $c_{j,i}$ is real value; i, j values from 1-49 are the number of fuzzy rules. The RBFNN adjusted the value of $c_{j,i}$. Now, there are only four fuzzy rules triggered. The fuzzy controller uses the product inference rule, singleton for fuzzifier, and central average method for defuzzification, the output signal can be calculated by:

$$u_{fx} = \frac{\sum_{n=i}^{i+1}\sum_{m=j}^{j+1} c_{m,n}(\mu_{A_n}(E_x)\mu_{B_m}(dE_x))}{\sum_{n=i}^{i+1}\sum_{m=j}^{j+1}(\mu_{A_n}(E_x)\mu_{B_m}(dE_x))} \approx \sum_{n=i}^{i+1}\sum_{m=j}^{j+1} c_{m,n}d_{n,m} \tag{11}$$

where $d_{n,m} \triangleq \mu_{A_n}(E_x)\mu_{B_m}(dE_x)$ and $\sum_{n=i}^{i+1}\sum_{m=j}^{j+1} d_{n,m} = 1$

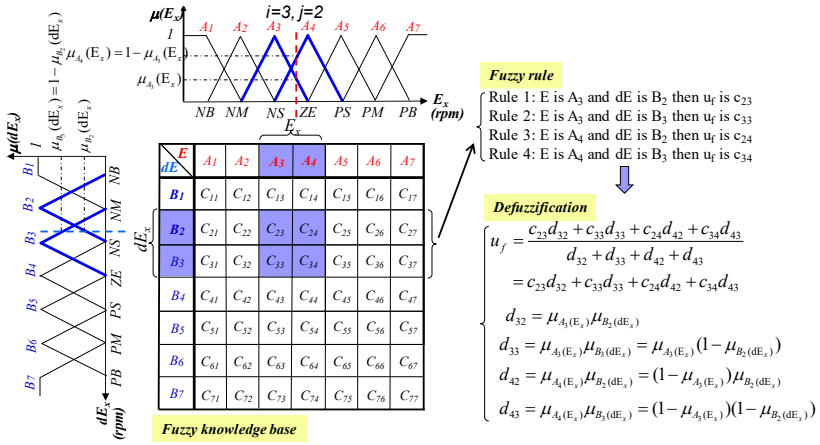

Figure 2. Structure of fuzzy controller with two input variables E and dE

2.3. Using RBFNN to adjust parameters of the fuzzy controller

During operation, the parameters of the system change, so it is necessary to adjust the $c_{j,i}$ (knowledge base of the fuzzy controller) in order to get a better output response. $c_{j,i}$ is adjusted according to the criterion of root mean square. The value function:

$$J_{ex} = \frac{1}{2}e_x^2 = \frac{1}{2}(x_m - x_p)^2 \tag{12}$$

Derivative (12) we get the minimum value:

$$\Delta c_{m,n} = -\varphi \frac{\partial J_{ex}}{\partial c_{m,n}} \tag{13}$$

where $\varphi > 0$ shows the convergence rate of the value function. From (11) and (12) we have:

$$\frac{\partial J_{ex}}{\partial c_{m,n}} = \frac{\partial J_{ex}}{\partial e}\frac{\partial e}{\partial x_p}\frac{\partial x_p}{\partial u_{fx}}\frac{\partial u_{fx}}{\partial c_{m,n}} = -e_x d_{n,m}\frac{\partial x_p}{\partial u_{fx}} \tag{14}$$

RBFNN was used to adjust the parameters of the fuzzy controller ($c_{j,i}$). RBFNN has one input layer, one hidden layer, and one output layer. Figure 3 (a) shows the diagram of RBFNN. The input signal of RBFNN:

$$X = [u(k), x_p(k-1), x_p(k-2)]^T \tag{15}$$

The output of RBFNN is:

$$x_{rbf} = \sum_{rbf=1}^{q} w_{rbf} h_{rbf} \tag{16}$$

where w_{rbf} is the weight function, and h_{rbf} is the activation function in the hidden layer of the neural network. The activation function is the Gaussian:

$$h_{rbf} = \exp\left(-\frac{\|X - c_{rbf}\|^2}{2\delta_{rbf}^2}\right), \text{rbf} = 1, 2, \ldots . \text{q} \tag{17}$$

where $c_{rbf} = [c_{r1} c_{r2} c_{r3}]^T$ and σ_{rbf} are respectively central and spread. To adjust the system parameters, the instantaneous value function is defined:

$$J_n = \frac{1}{2}(x_p - x_{rbf})^2 = \frac{1}{2}e_{nn}^2 \tag{18}$$

From (18) the weight, central, and spread function were updated:

$$w_{rbf}(k_{+1}) = w_{rbf}(k) + \eta e_{nn}(k) h_{rbf}(k) \tag{19}$$

$$c_{rs}(k+1) = c_{rs}(k) + \eta e_{nn}(k) w_{rbf}(k) h_{rbf}(k) \frac{X_s(k) - c_{rs}(k)}{\sigma_{rbf}^2(k)} \tag{20}$$

$$\sigma_{rbf}(k+1) = \sigma_{rbf}(k) + \eta e_{nn}(k) w_{rbf}(k) h_{rbf}(k) \frac{\left\| X(k) - c_{rbf}(k) \right\|^2}{\sigma_{rbf}^2(k)} \tag{21}$$

where s=1, 2, 3 and $\eta > 0$ is a learning rate of RBFNN. In (5) is transformed as follows:

$$\frac{\partial x_p}{\partial u_f} = \frac{\partial x_p}{\partial_u} \frac{\partial_u}{\partial u_f} \cong (k_p + k_i T) \frac{\partial x_p}{\partial_u} \tag{22}$$

where the Jacobian $\frac{\partial x_p}{\partial_u}$ was calculated from (17) and through the relationship $e_{nn} = x_p - x_{rbf}$ of RBFNN on Figure 3 (a):

$$\frac{\partial x_p}{\partial_u} = \frac{\partial x_{rbf}}{\partial_u} = \sum_{rbf=1}^{q} w_{rbf} h_{rbf} \frac{c_{r1} - u(k)}{\sigma_{rbf}^2} \tag{23}$$

From (13), (14), (22), and (23) the parameter $c_{m,n}$ of the fuzzy PI controller was updated as follows:

$$\Delta c_{m,n}(k) = \propto e(k)(k_p + k_i T) d_{n,m} \sum_{rbf=1}^{q} w_{rbf} h_{rbf} \frac{c_{r1} - u(k)}{\sigma_{rbf}^2} \tag{24}$$

with m = j, j+1 and n = i, i+1

2.4. The motion trajectory of the system

A reference equation generated the motion trajectory of the system with $x_m(k)$ and $y_m(k)$ on both axes. The motion trajectory is a star shape as shown in Figure 3 (b). It has five segments from a to e. The equation for each segment is described as follows. Where S, x_i, y_i are respectively moving positions, coordinates on the X and Y-axis.

Segment a: $x_i = S + x_{i-1}, y_i = y_{i-1}$ (25)

Segment b: $x_i = -S * sin54^0 + x_{i-1}, y_i = -S * sin54^0 + y_{i-1}$ (26)

Segment c: $x_i = S * sin18^0 + x_{i-1}, y_i = S * sin72^0 + y_{i-1}$ (27)

Segment d: $x_i = S * sin18^0 + x_{i-1}, y_i = -S * sin72^0 + y_{i-1}$ (28)

Segment e: $x_i = -S * sin54^0 + x_{i-1}, y_i = S * sin36^0 + y_{i-1}$ (29)

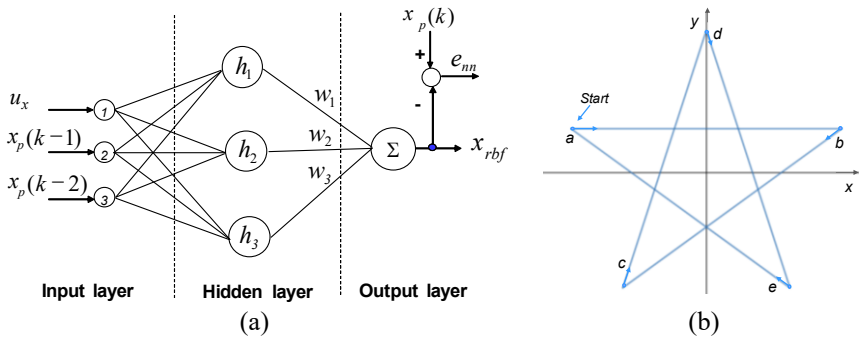

Figure 3. (a) RBFNN adjusts the parameters of the fuzzy PI controller, (b) The motion trajectory of the XY axes

3. IMPLEMENT THE CONTROLLER BASED ON FPGA AND USING MATLAB/SIMULINK/MODELSIM FOR SIMULATIONS

3.1. FPGA controller structure and Matlab simulation

FPGA technology was used to implement the control algorithm of this paper. One of the crucial advantages of FPGA is that it allows the completion of the product quickly and makes it easy to use design tools. This research used Altera Cyclone II EP2C115 to implement all algorithms of this research. Altera Cyclone II EP2C115 has 114,480 logical Elements (LEs) and 3.9 Mbits of RAM. With the co-simulation of Simulink/ModelSim, the evaluation task of the VHDL code becomes more convenient. We can also change the input signal for examining VHDL code behavior, which is a challenging job in some VHDL development tools, such as Quartus II. The Simulink/ModelSim model for evaluation of the proposed algorithm is shown in Figure 4. The two motors, two IGBT inverters, are executed by the function of Simulink. The four ModelSim models execute the VHDL code of whole proposed algorithms. In Figure 4, the ModelSim model named ModelSim_1 and ModelSim_3 performs the function of position and speed controller of the X- and Y-axis. The ModelSim model named ModelSim_2 and ModelSim_4 performs the function of the current vector controller of the X- and Y-axis. The rotor speed of the two motors was fed back to ModelSim_1 and ModelSim_3. The rotor position, three-phase stator currents of two motors, were fed back to ModelSim_2 and ModelSim_4. The m-file is applied to develop the program of star shape motion trajectory. The following simulation will test the result of the proposed algorithm. Firstly, the decoupled system is tested. Secondly, self-adjusting is tested. The single-axis speed response was checked and compared between typical fuzzy PI and self-adjusting fuzzy PI controllers. Finally, the two axes simultaneous motion within the star trajectory was checked. Figure 5 shows the architecture of the proposed algorithm. It implements FPGA technology for the PMLSM position and speed controller. The CLK and CLK_40ns are the input clocks 50MHz and 25MHz to supply all modules of the proposed algorithm. The command speed and rotor speed x_m, x_p are the input of the self-adjusting fuzzy PI controller for the X-axis. The three-phase currents (Iax, Ibx, Icx) and rotor flux angle θex are the input of the vector control for the X-axis. The 6 PWMs are the output of SVPWM on the X-axis. Similar for the Y-axis.

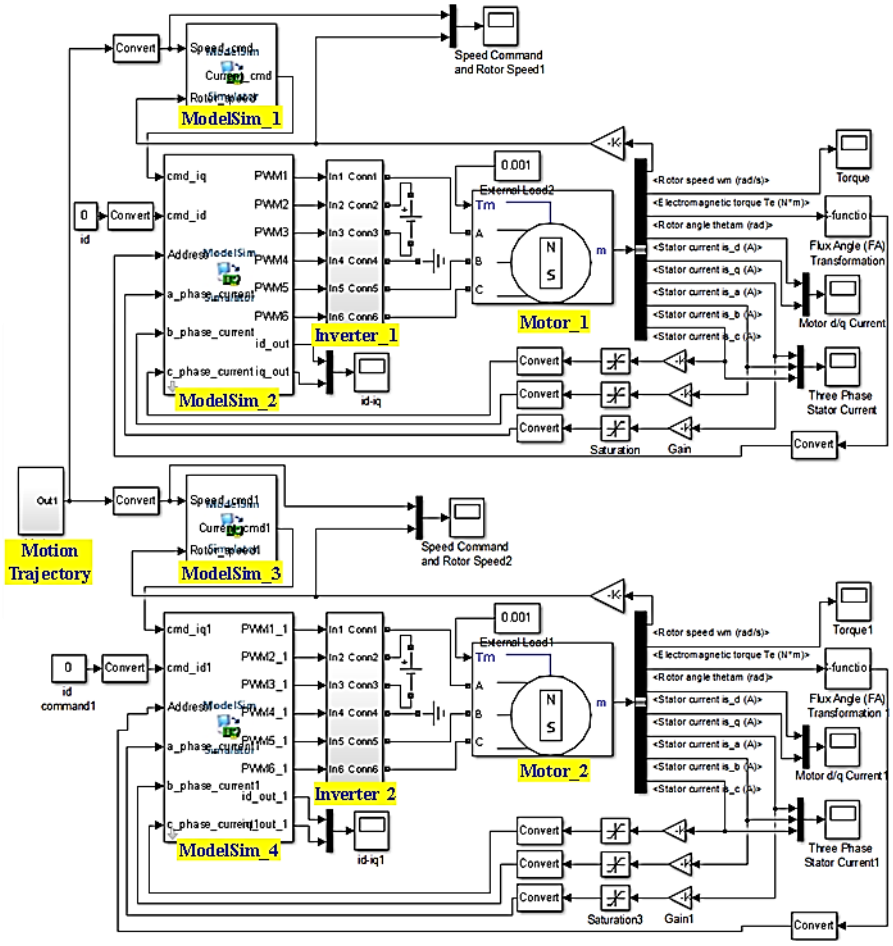

Figure 4. The Simulink/ModelSim model for evaluation of the proposed algorithm

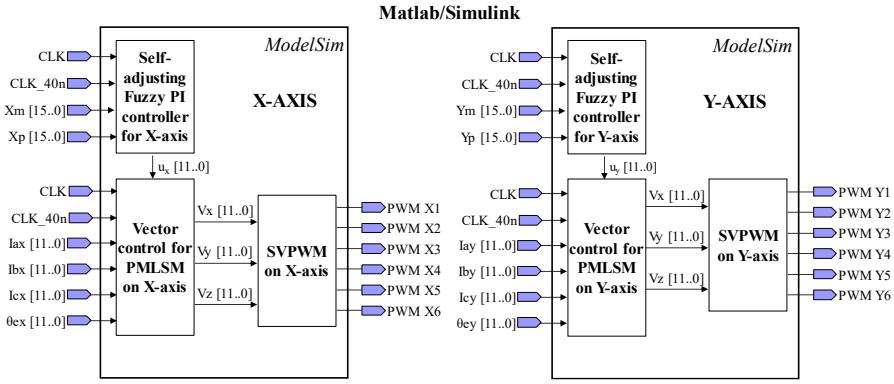

Figure 5. The FPGA based a speed and position controller for PMLSM drive

3.2. Implementation of the control algorithms

The transfer function in the discrete domain of the motion equation was described as follows:

$$\frac{x_p(z^{-1})}{i_q^*(z^{-1})} = \frac{\theta z^{-1}}{(1-\emptyset z^{-1})(1-z^{-1})} \tag{30}$$

where $\emptyset = \exp\left(-\frac{B_m T}{M_m}\right), \theta = \frac{K_t(1-\emptyset)}{B_m}$. Hence the control equation at the output is:

$$x_p(k) = \theta i_q^*(k-1) + (1+\emptyset)x_p(k-1) - \emptyset x_p(k-2) \tag{31}$$

4. RESULTS AND DISCUSSION

The speed and position controller of both X and Y-axes have been embedded in one FPGA chip. The proposed algorithms in the main block diagram are shown in Figure 6. The FSM sequential programming technique was used to describe control algorithms. The FSM method with one multiplier, one adder, one look-up table, has been used to implement the proposed algorithm. In the speed and position control loop, the fuzzy PI controller is applied. The RBFNN adjusted the knowledge base of the fuzzy PI controller. The internal circuit of the main proposed algorithm, included one circuit of position and speed fuzzy PI controller, one circuit of RBFNN, one circuit of the current controller, and coordinate transformation.

In Figure 6 (a), there are 85 steps to carry out the fuzzy PI controller. The steps s_0-s_1 compute the rotor speed error and error change. The steps s_2-s_5 compute the fuzzification. Step s_6 looks up the fuzzy table to select four values of c_{mn} based-on i and j obtained from s_5. The steps s_7-s_{15} compute the defuzzification function; those are (6-11). The steps s_{16}-s_{18} compute the current command, and it is the PI function. The steps s_{19}-s_{70} and s_{71}-s_{73} calculate three neural networks; the FPGA is parallel processing; Therefore, it needs only 52 steps for completion. The steps s_{74}-s_{84} turn the fuzzy rule parameters. To compute one neural takes 52 steps as shown in Figure 6 (b). The steps s_0-s_5 compute the norm value. The steps s_6-s_{33} calculate the exponential function (17). The steps s_{34}-s_{35} compute the output of neural and Jacobian at r^{th} neural. The steps s_{36}-s_{51} update the weight, variance, and node center value of the neural network.

The data type in vector control and SVPWM for PMLSM are 12-bits. The data type is designed with a 16-bits length for the self-adjusting fuzzy PI controller. Each clock pulse in the FPGA is 40ns to complete a command. There are 85 steps (3.4μs), 52 steps (2.08μs) to implement the fuzzy PI and one neural network, respectively. It does not lose any performance because the operation time of the self-adjusting fuzzy PI with 3.4μs is much less than the sampling time of the position and speed controller (2 kHz). Although the control algorithm of this research in using RBFNN to adjust the parameters of the fuzzy PI controller is quite complicated, using the FSM description in step-by-step order, it simplified the coding. The simulation was performed on Matlab/Simulink/ModelSim for adjusting and verifying the controller's parameters and checking the accuracy of the proposed algorithm. The ModelSim software was embedded in Matlab/Simulink for simulating the whole system. Motion trajectories, feedback signals, PMLSM engine models were designed on Simulink. VHDL codes of position control and speed control algorithms for XY axes were embedded in ModelSim.

The sampling frequency of the position and speed controllers was 2 kHz; the vector control was 16 kHz. The FPGA resources used for the whole algorithm are 73,402 LEs and 348,947 RAM bits. Firstly, the step response is set up, the step signal with period of 0.5s and magnitude variation from 0 => 500 rpm => 0rpm => 1000 rpm =>

1500 rpm => 2000 rpm => 1500 rpm => 2000 rpm => 1500 rpm => 2000 rpm were adopted as test conditions. The simulation results are presented in Figures 7-9. It presents a good speed response as shown in Figure 7 and the i_d, i_q currents complete decoupled effect as shown in Figure 8, the i_d current approached zero. The PMLSM can control as DC motor, and the speed is dependent on the i_q current. The speed response in Figure 7 without overshoot, the rising time is about 0.2 s, and steady-state value is zero. The three-phase stator currents in Figure 9 are balanced.

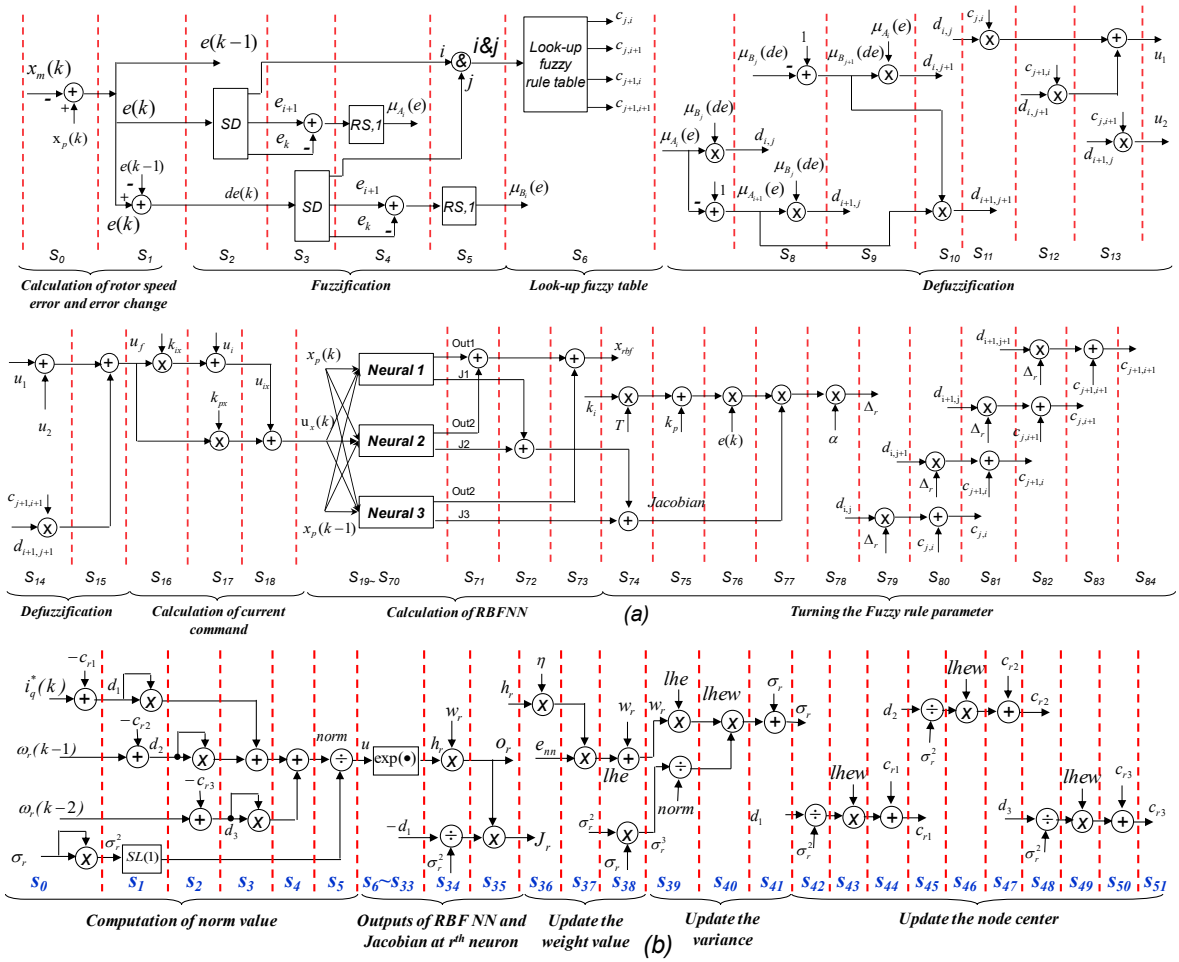

Figure 6. The detail of the internal circuit design of the main proposed algorithm, (a) There are 85 steps to carry out the fuzzy PI controller, (b) There are 52 steps to compute one neural network

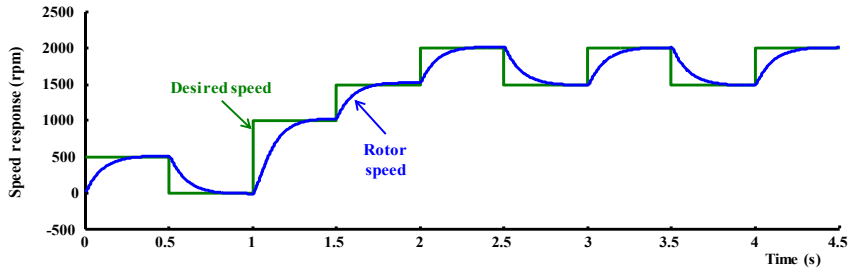

Figure 7. The speed response of the motor for checking the decoupled effect of the PMLSM, the rotor speed can follow the desired speed at the input.

After checking the system's decoupling, the load condition is changed to evaluate the system's effectiveness. The load of the motor is in light-load ($J = 0.000036, F = 0.00043$) and heavy-load ($J = 0.000324, F = 0.0039$) condition. The heavy-load is nine times heavier than the light-load. The reference model was applied to check the load condition. The simulation result of the heavy-load condition is shown in

Figure 10. In the beginning, the fuzzy controller was used to control the speed of the motor. The speed response has lag and overshoot. After one second, the self-adjusting fuzzy PI affects the $c_{m,n}$ which is turning to reduce the error speed between rotor speed and reference model. The result shows that with the proposed algorithm, the rotor speed can track the reference model well. Similar to the light-load condition shown in Figure 11.

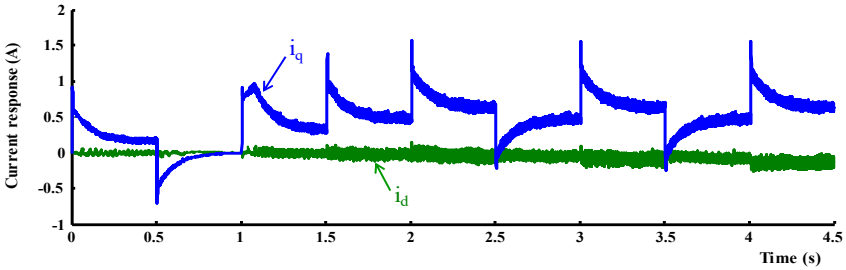

Figure 8. The i_q and i_d of the motor, decoupled effect, helps to control the servo motor as to control the DC motor

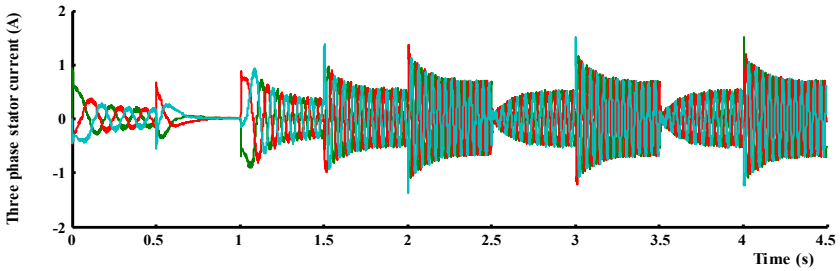

Figure 9. The three phase-currents of the motor are balance

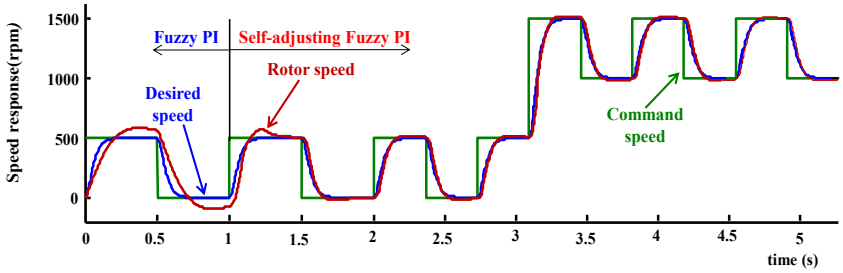

Figure 10. The rotor speed of the motor at the fuzzy controller and self-adjusting fuzzy PI controller are compared at the heavy-load condition

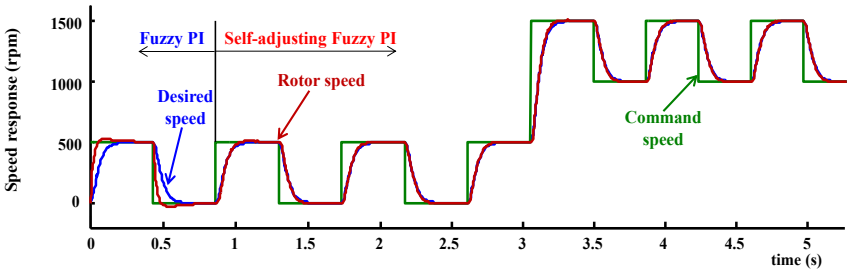

Figure 11. The rotor speed of the motor at the fuzzy controller and self-adjusting fuzzy PI controller are compared at the light-load condition

Finally, the star shape is considered. The input signal with five segments from a to e (25 to 29) was stimulated from the Simulink. The simulation results are shown in Figures 12 and 13. The tracking performance by using fuzzy PI and self-adjusting fuzzy PI is compared. Figure 12 shows the performance of fuzzy PI; The error in

X-axis is a maximum of about 3.5mm; the error in the Y-axis is a maximum of approximately 10mm. Figure 13 shows the performance of the self-adjusting fuzzy PI, the output motion of the X and Y-axis can draw the star shape well. The position output response followed the motion star trajectory very well. The error in X-axis is less than 2 mm; the error in the Y-axis is less than 8mm. It still has a different error between the two axes. However, objective control has been achieved. The self-adjusting fuzzy PI controllers with the knowledge base are turned by RBFNN; the results proposed algorithm are better than the fuzzy PI controller.

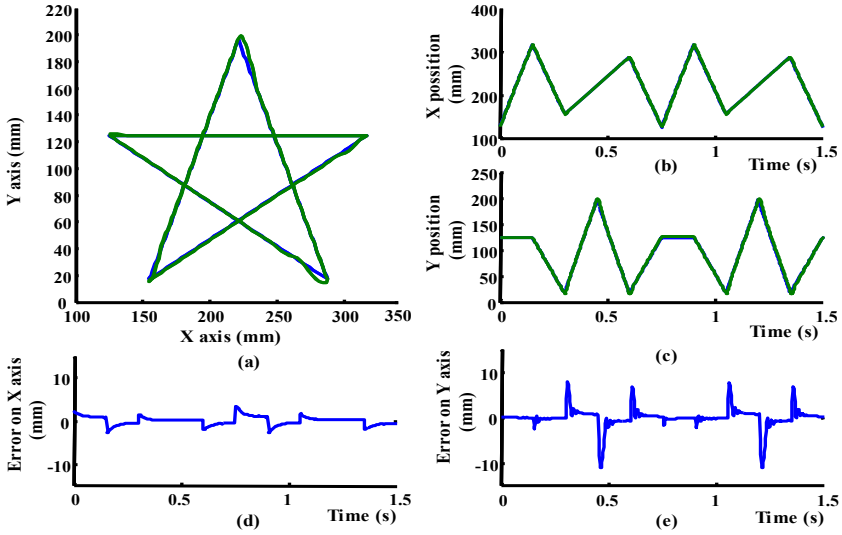

Figure 12. (a) The combination of two axes motion in case fuzzy PI controller, (b) the X-axis position, (c) the Y-axis position, (d) The error on X-axis is a maximum of about 3.5mm, (e) The error on Y-axis is a maximum of approximately 10mm

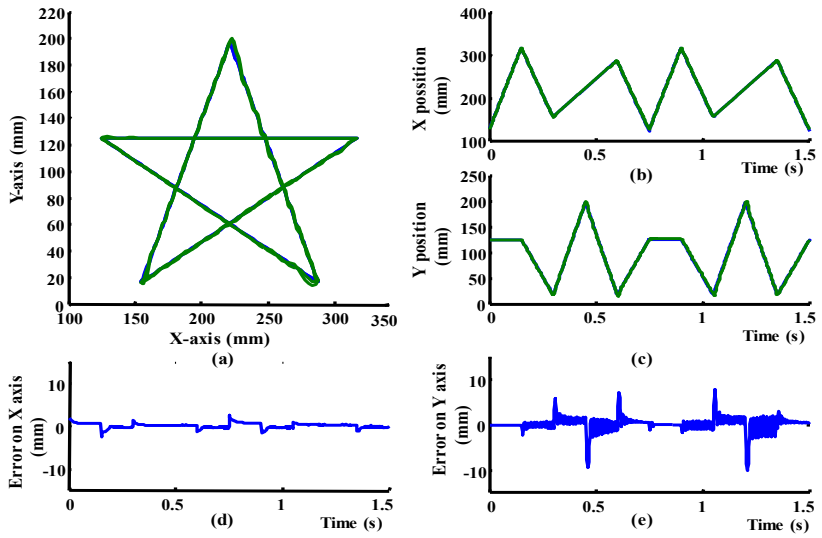

Figure 13. The output position response faithfully follows the input star trajectory signal in the case of a self-adjusting fuzzy PI controller; (a) the combination of two axes motion, (b) the X-axis position is less than 2 mm, (c) the Y-axis position is less than 8 mm, (d) The error on X-axis, (e) The error on Y-axis

5. CONCLUSION

This paper successfully applied FPGA technology to control the movement of the two-axes XY system. The proposed structure includes motion trajectory, position, and speed self-adjusting fuzzy PI controller. The RBFNN adjusted the fuzzy PI controller during the operation. The controller of both X and Y-axis was implemented on only one chip. The RBFNN is computed in a parallel method; it gets the fast response to adjust the fuzzy parameter when the system changes. Besides, the FSM method can help to control the implementation of the exact whole system. The system has been successfully simulated in Matlab/Simulink

and ModelSim environments. The simulation results show that the proposed algorithm's tracking performance is better than the fuzzy PI controller. The motion controller is sufficient and correct.

REFERENCES

[1] Lai, C.-K.; Tsao, Y.-T.; Tsai, C.-C, "Modeling, Analysis, and Realization of Permanent Magnet Synchronous Motor Current Vector Control by MATLAB/Simulink and FPGA," *Machines,* vol. 5, no. 26, 2017.

[2] Quynh, N. V., Kung, Y. S., Van Dung, P., Liao, K. Y., & Chen, S. W., "FPGA-Realization of Vector Control for PMSM Drives*," Applied Mechanics and Materials*, vol. 311, pp. 249-254, 2013.

[3] Z. Ma and X. Zhang, "FPGA-based sensorless control for PMSM drives using the stator/rotor frame extended Kalman filter," *2018 Chinese Control And Decision Conference (CCDC),* Shenyang, pp. 102-107. 2018.

[4] Arun Prasad K. M., Usha Nair, "Intelligent fuzzy sliding mode controller based on FPGA for the speed control of a BLDC motor," *International Journal of Power Electronics and Drive System (IJPEDS)*, vol. 11, no. 1, pp. 477-486. March 2020.

[5] Nguyen V. Q., Tran H. T. and Huynh D. M. T., "Fuzzy Controller for Better Tennis Ball Robot," *J. Fundam. Appl. Sci*., vol. 9, no. 7S, pp. 667-677, 2017.

[6] D. Kiran Kumar, G. Tulasi Ram Das, "Simulation and Analysis of Modified DTC of PMSM," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 8, no. 5, pp. 2894~2903, October 2018.

[7] B. M. Badr, A. M. Eltamaly and A. I. Alolah, "Fuzzy controller for three phases induction motor drives," *2010 IEEE Vehicle Power and Propulsion Conference*, Lille, pp. 1-6, 2010.

[8] N. V. Quynh and Y. Kung, "FPGA-realization of fuzzy speed controller for PMSM drives without position sensor," *2013 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 278-282, 2013.

[9] Liu, Yan-Jun, Yan, T. H., Wu, B., He, B., Li, W. H., Wang, R. B., "A Novel Fuzzy Sliding-Mode Control for Discrete-Time Uncertain System" *Hindawi Publishing Corporation, Mathematical Problems in Engineering*, vol. 2016, 2016.

[10] Zhang Chunmei, Liu Heping, Chen Shujin, Wang Fangjun, "Application of neural networks for permanent magnet synchronous motor direct torque control," *Journal of Systems Engineering and Electronics*, vol. 19, no. 3, pp. 555-561, 2008.

[11] Atif Iqbal, Haitham Abu-Rub & Hazem Nounou, "Adaptive fuzzy logic-controlled surface mount permanent magnet synchronous motor drive," *Systems Science & Control Engineering,* pp. 465-475, 2014.

[12] Jurifa Mat Lazi, Zulkifilie Ibrahim, MHN Talib, Azrita Alias, Ainain Nur, Maaspaliza Azri, "Speed and position estimator of for sensorless PMSM drives using adaptive controller," *International Journal of Power Electronics and Drive System (IJPEDS),* vol. 10, no. 1, pp. 128~136, March 2019.

[13] Larbi M'hamed, Adjir Roufaida, Ait Amer Meziane Nawa, "Sensorless control of PMSM with fuzzy model reference adaptive system," *International Journal of Power Electronics and Drive System (IJPEDS)*, vol. 10, no. 4, pp. 1772~1780, December 2019.

[14] Y. Kung, "Design and Implementation of a High-Performance PMLSM Drives Using DSP Chip," *in IEEE Transactions on Industrial Electronics*, vol. 55, no. 3, pp. 1341-1351, March 2008.

[15] 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. Industrial Electronics*, vol. 54, no. 1, pp. 265-271, 2007.

[16] S.T. Brassai, L. Bako, G. Pana, S. Dan, "Neural control based on RBF network implemented on FPGA," *Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment,* pp. 41-46, 2008.

[17] J.S. Kim, S. Jung, "Implementation of the RBF neural chip with the back-propagation algorithm for on-line learning," *Applied Soft Computing*, vol. 29, pp. 233-244, 2015.

[18] Yasser Ahmed, Ayman Hoballah, "Adaptive filter-FLC integration for torque ripples minimization in PMSM using PSO," *International Journal of Power Electronics and Drive System (IJPEDS),* vol. 10, no. 1, pp. 48-57, March 2019.

[19] Ahmed Jadaan Ali, Ziyad Farej, Nashwan Sultan, "Performance evaluation of a hybrid fuzzy logic controller based on genetic algorithm for three phase induction motor drive," *International Journal of Power Electronics and Drive System (IJPEDS),* vol. 10, no. 1, pp. 117-127. March 2019.

[20] Ramana Pilla, Santukumari Killari, K.B.Madhu Sahu, "Design and Simulation of the Control System for Inverter-fed Permanent Magnet Synchronous Motor Drive," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 12, no. 3, pp. 958-967, December 2018.

[21] Y. Kung, J. Lin and N. V. Quynh, "ModelSim/Simulink co-simulation of sensorless PMSM speed control system with EKF estimator and adaptive fuzzy controller," *2014 International Conference on Information Science, Electronics and Electrical Engineering*, Sapporo, pp. 1488-1492, 2014.

[22] Hsin-Hung Chou, Ying-Shieh Kung, Nguyen Vu Quynh, Stone Cheng, "Optimized FPGA design, verification and implementation of a neuro-fuzzy controller for PMSM drives," *Mathematics and Computers in Simulation*, vol. 90, pp. 28-44, 2013.

[23] Sandre-Hernandez, O., Rangel-Magdaleno, J., Morales-Caporal, R. *et al.,* "HIL simulation of the DTC for a three-level inverter fed a PMSM with neutral-point balancing control based on FPGA," *Electr. Eng.,* vol. 100, pp. 1441-1454, 2018.

[24] Ying-Shieh Kung, Nguyen Vu Quynh, Nguyen Trung Hieu, and Jin-Mu Lin, "FPGA Realization of Sensorless PMSM Speed Controller Based on Extended Kalman Filter," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[25] The Mathworks, Matlab/Simulink User's Guide, Application Program Interface Guide, 2004.

[26] Modeltech, ModelSim Reference Manual, 2004.