Research Article

The Fuzzy PI Controller for PMSM's Speed to Track the Standard Model

Nguyen Vu Quynh

Electrical and Electronics Department, Lac Hong University, Bien Hoa, Dong Nai, Vietnam

Correspondence should be addressed to Nguyen Vu Quynh; vuquynh@lhu.edu.vn

Received 4 April 2020; Revised 2 June 2020; Accepted 15 June 2020; Published 13 July 2020

Guest Editor: Carlos Llopis-Albert

Copyright © 2020 Nguyen Vu Quynh. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a fuzzy PI controller to control the speed of a permanent magnet synchronous motor (PMSM). The structure of the system includes the speed loop controller (SLC) and the current loop controller (CLC). The speed loop controller is the fuzzy PI and standard model (SM). The CLC includes vector control and the space vector pulse width modulation (SVPWM). It compiles two closed-loop control systems for the PMSM. This research uses a very high-speed integrated circuit hardware description language (VHDL) to implement the proposed algorithm and embed it into Matlab/Simulink for simulation. Based on the PMSM parameter, this article tests the controller's correctness with some of the load cases by changing the combined inertia and viscous friction of rotor and load. After success in simulation, the system is tested again by experiment on the FPGA kit. The simulation and experiment results show that when the load changes, the PMSM speed is still stable. The novelty of this research is that it compares two kinds of controllers between simulation and experiment results.

1. Introduction

The PMSMs are commonly used in systems requiring high precision such as the robot and mechanical processing. However, there are many uncertainties, for example, noise, external load, and friction force. They affect the performance quality of the system. Many intelligent control techniques such as fuzzy, neural network, and genetic algorithm have been developed to cope with those problems [1–4]. They helped to control precisely the speed and position of the motor. The FPGA supplied a solution for the implementation of the algorithm. Especially, FPGA has a programmable hard-wired feature, fast calculation ability, low power consumption, and higher density. There were many types of research on intelligent controllers such as fuzzy control, sliding fuzzy control, artificial intelligence, and adaptive control based on the FPGA platform to control multiaxis robots for improving precision in machining and assembly [5–9]. The fuzzy control method is an intelligent control system, with an inference mechanism from experts' control experience. However, obtaining fuzzy sets and optimal membership functions are not easy. To resolve this

problem, the fuzzy knowledge base values were adjusted appropriately based on the feedback system parameter [10, 11]. This research proposes a drive system to overcome the slow response and tuning problems of the conventional PI speed controllers without any mathematical calculations [12, 13]. However, the aforementioned research studies do not check with the massive change of load. This research applied a fuzzy PI controller, current controller, and SVPWM for PMSM drive. The contents of the article include, firstly, the mathematical model of PMSM, SVPWM algorithm, vector control method, and fuzzy PI controller. Secondly, VHDL is used to program all systems. Finally, the article presents a simulation on Matlab/Simulink and experiment on FPGA kit results.

Figure 1 shows the block diagram of the proposed algorithm. The speed of the motor (ω_r) is fed back to the speed loop controller to compare with the output of the standard model (SM). The fuzzy PI controller generates the signal to minimize the error between rotor speed and SM. The adjustment mechanism keeps the motor speed stable when the load changes. The three-phase stator currents (i_a , i_b , i_c) and rotor flux angle (θ_e) are fed back to the current controller to

FIGURE 1: The simulation architecture of the fuzzy PI controller for PMSM drive.

calculate the generated voltage vector for SVPWM. The SVPWM generated the electric current to control three-phase inverter for providing power to the motor.

2. The Vector Controller Design for PMSM

2.1. The PMSM Mathematical Model.

$$\begin{aligned} \frac{di_d}{dt} &= \frac{1}{L_d}v_d - \frac{R}{L_d}i_d + \frac{L_q}{L_d}p * \omega_r * i_q, \\ \frac{di_q}{dt} &= \frac{1}{L_q}v_q - \frac{R}{L_q}i_q - \frac{L_d}{L_q}p * \omega_r * i_d - \frac{\lambda * p * \omega_r}{L_q}, \end{aligned} \quad (1)$$

in which L_q and L_d are the inductances on q -axis and d -axis; R is the resistance of the stator windings; i_q and i_d are the currents on q -axis and d -axis; V_q and V_d are the voltages on q -axis and d -axis; λ is the permanent magnet flux linkage; p is the number of pole pairs; and ω_r is the rotational speed of the rotor.

2.2. The Vector Controller. The current controller of the PMSM drive in Figure 1 is based on a vector control approach. The current controller structure includes Clark, Park, inverse Park, and inverse Clark transformation, two PID controllers, and sine/cosine of flux angle. Three-phase current is fed back through vector control, enabling controlling current $i_d \approx 0$, which helps control the three-phase motor similar to a DC motor. The torque of the PMSM is controlled via current on the q -axis (i_q). The vector control is described below.

The Clark transformation converts the stationary a - b - c frame (i_a, i_b, i_c) to stationary α - β frame (i_α, i_β).

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}. \quad (2)$$

The Park transformation converts the stationary α - β frame (i_α, i_β) to synchronously rotating d - q frame (i_d, i_q).

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos \theta_e & \sin \theta_e \\ -\sin \theta_e & \cos \theta_e \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}. \quad (3)$$

The inverse Park transformation converts a rotating reference d - q frame (v_d, v_q) to stationary α - β frame (v_α, v_β).

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} \cos \theta_e & -\sin \theta_e \\ \sin \theta_e & \cos \theta_e \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix}. \quad (4)$$

The inverse Clark transformation converts a stationary α - β frame (v_α, v_β) to stationary a - b - c frame ($v_{ref1}, v_{ref2}, v_{ref3}$).

$$\begin{bmatrix} v_{ref1} \\ v_{ref2} \\ v_{ref3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix}. \quad (5)$$

Electromagnetic moment of the motor could be calculated using the following formula:

$$T_e = 1.5p[\lambda * i_q + (L_d - L_q)i_d * i_q] \approx K_t * i_q. \quad (6)$$

The mathematical equation of the motor under the load-bearing condition is

$$\frac{d\omega_r}{dt} = \frac{1}{J_m(T_e - F\omega_r - T_m)}, \quad (7)$$

where T_e is the motor torque; J_m and F are the inertia and viscous friction of rotor and load; T_m is the shaft mechanical torque; and K_t is the torque constant.

3. The SVPWM Algorithm Design

The SVPWM is a control technique widely used in power electronic equipment control. The idea of the method is to create a continuous shift of the space vector equivalent of the voltage vector of the inverter on the circle orbit (Figure 2). With the steady shift of the space vector on a circular orbit,

FIGURE 2: (a) The eight primary vectors and switching patterns. (b) PWM at S_3 Sector and its duty cycle.

the higher harmonics are removed, and the relationship between the control signal and output voltage amplitude became linear. In this article, the SVPWM technique is used to create a vector to control the inverter using IGBT power switches.

The phase-to-phase voltage and the phase-to-neutral voltage of the motor:

$$\begin{bmatrix} V_{AB} \\ V_{BC} \\ V_{CA} \end{bmatrix} = V_{DC} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad (8)$$

$$\begin{bmatrix} V_{AN} \\ V_{BN} \\ V_{CN} \end{bmatrix} = \frac{1}{3} V_{DC} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix},$$

in which a , b , and c are switching vectors of IGBT power electronic switches.

There are six power IGBTs in the voltage-source inverter. When the upper switch is active, the corresponding lower switches must be turned off. There must be coordinated delay time (dead band) between the opening and closing of two switches on the same phase.

The eight primary space vectors ($V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7$) are shown in Figure 2(a). The output voltage (V_{ref}) is located at one of the six sectors ($S_1, S_2, S_3, S_4, S_5, S_6$) at any given time.

$$V_{ref} = \frac{T_1}{T} U_x + \frac{T_2}{T} U_{x+1} + \frac{T_0(V_0 \text{ or } V_7)}{T}, \quad (9)$$

where $x \neq 0, 7$ and T is half of the PWM period, $T_0 = T - T_1 - T_2$. For example, in the S_3 sector, the output

voltage can be calculated with V_1 and V_2 vectors (Figure 2(a)).

$$V_1 = \frac{2V_{DC}}{3} \vec{\alpha}, \quad (10)$$

$$V_2 = \frac{V_{DC}}{3} \vec{\alpha} + \frac{V_{DC}}{\sqrt{3}} \vec{\beta}. \quad (11)$$

From equations (9)–(11), the output voltage and the value of T_1 and T_2 can be obtained:

$$V_{ref} = \frac{T_1 * V_1}{T} + \frac{T_2 * V_2}{T} = \frac{T_1}{T} \frac{2V_{DC}}{3} \vec{\alpha} + \frac{T_2}{T} \left(\frac{V_{DC}}{3} \vec{\alpha} + \frac{V_{DC}}{\sqrt{3}} \vec{\beta} \right) \approx V_{\alpha} \vec{\alpha} + V_{\beta} \vec{\beta}, \quad (12)$$

$$T_1 = \frac{T(3V_{\alpha} - \sqrt{3}V_{\beta})}{2V_{DC}},$$

$$T_2 = \frac{\sqrt{3}T * V_{\beta}}{V_{DC}}.$$

Similarly, the values of T_1 and T_2 at the other sectors shown in Table 1 with T_x , T_y , and T_z are

$$T_x = \frac{\sqrt{3}T * V_{\beta}}{V_{DC}}, \quad (13)$$

$$T_y = \frac{T(3V_{\alpha} + \sqrt{3}V_{\beta})}{2V_{DC}}, \quad (14)$$

$$T_z = \frac{T(-3V_{\alpha} + \sqrt{3}V_{\beta})}{2V_{DC}}, \quad (15)$$

TABLE 1: Time value in each section.

Value	S_1	S_2	S_3	S_4	S_5	S_6
T_1	T_z	T_y	$-T_z$	$-T_x$	T_x	$-T_y$
T_2	T_y	$-T_x$	T_x	T_z	$-T_y$	$-T_z$

with the condition $T_1 + T_2 > T$; T_1 and T_2 are modified as follows:

$$T_{1(\text{cond})} = \frac{T_1 * T}{T_1 + T_2}, \quad (16)$$

$$T_{2(\text{cond})} = \frac{T_2 * T}{T_1 + T_2}. \quad (17)$$

The duty cycles (Ta_{on} , Tb_{on} , Tc_{on}) in Figure 2(b) can be calculated as follows:

$$Ta_{\text{on}} = \frac{T - T_1 - T_2}{2} = \frac{T_0}{2}, \quad (18)$$

$$Tb_{\text{on}} = Ta_{\text{on}} + T_1, \quad (19)$$

$$Tc_{\text{on}} = Tb_{\text{on}} + T_2. \quad (20)$$

PWM1, PWM2, and PWM3 with the duty time at V_1 and V_2 and zero vector (V_0 and V_7) are T_1 , T_2 , and T_0 , respectively. The values of CMP_1 , CMP_2 , and CMP_3 at each sector are shown in Table 2. For quickly determining the sector, (5) (inverse Clark transformation) is modified as follows:

$$\begin{bmatrix} v_{ref1} \\ v_{ref2} \\ v_{ref3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & \frac{\sqrt{3}}{2} \\ -1 & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_\beta \\ v_\alpha \end{bmatrix}. \quad (21)$$

We can determine the sectors S_1 – S_6 (Figure 3) according to the rule below:

$$\begin{aligned} \text{if } V_{ref1} > 0, & \quad \text{then } a = 1 \text{ else } a = 0, \\ \text{if } V_{ref2} > 0, & \quad \text{then } b = 1 \text{ else } b = 0, \\ \text{if } V_{ref3} > 0, & \quad \text{then } c = 1 \text{ else } c = 0. \end{aligned} \quad (22)$$

The sector of SVPWM is defined by this equation: sector = $a + 2b + 4c$. From equations (13)–(15) and (21), we have

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \frac{\sqrt{3}T}{V_{DC}} \begin{bmatrix} v_{ref1} \\ -v_{ref3} \\ -v_{ref2} \end{bmatrix}. \quad (23)$$

The implementation of SVPWM by VHDL can be summarized below:

- (i) From the value of V_{ref1} , V_{ref2} , and V_{ref3} , we determined the sector as (22).

TABLE 2: The values of CMP_1 , CMP_2 , and CMP_3 at each section.

Section	S_1	S_2	S_3	S_4	S_5	S_6
CMP_1	Tb_{on}	Ta_{on}	Ta_{on}	Tc_{on}	Tc_{on}	Tb_{on}
CMP_2	Ta_{on}	Tc_{on}	Tb_{on}	Tb_{on}	Ta_{on}	Tc_{on}
CMP_3	Tc_{on}	Tb_{on}	Tc_{on}	Ta_{on}	Tb_{on}	Ta_{on}

FIGURE 3: The voltage values supplying for the SVPWM.

- (ii) Calculate the value of T_x , T_y , and T_z from (23).
- (iii) Select T_1 and T_2 in Table 1.
- (iv) Calculate the duty cycle (Ta_{on} , Tb_{on} , and Tc_{on}) from equations (18)–(20).
- (v) Select comparative value (CMP_1 , CMP_2 , and CMP_3) at the duty cycles from Table 2.

4. Standard Model (SM)

The SM is the standard speed for PMSM. The output of the SM is increase and tangent to the set value at the input. To build the SM to meet the above requirements, we used the typical second-order system:

$$\frac{\omega_m(s)}{\omega_r^*(s)} = \frac{\omega_n^2}{s^2 + 2 * \eta * \omega_n * s + \omega_n^2}, \quad (24)$$

where ω_n and η are the natural frequency and damping ratio, respectively. Using the bilinear transformation, we obtained the discrete function in Z domain and difference equation:

$$\frac{\omega_m(z^{-1})}{\omega_r^*(z^{-1})} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}},$$

$$\begin{aligned} \omega_m(k) = & -b_1 * \omega_m(k-1) - b_2 * \omega_m(k-2) \\ & + a_0 * \omega_r^*(k) + a_1 * \omega_r^*(k-1) + a_2 * \omega_r^*(k-2), \end{aligned} \quad (25)$$

with coefficients: $a_0 = 0.0077$, $a_1 = 0.0153$, $a_2 = 0.0077$, $b_1 = -1.6496$, and $b_2 = 0.6803$.

The error between the output of SM and the rotor speed is supplied to the fuzzification and the fuzzy adjustment mechanism function of the fuzzy PI controller. Figure 4 shows the SM; it does not change the rising time, zero steady-state value, and overshoot when the system is running. Figures 5 and 6 show a finite state machine (FSM) and the VHDL code to describe the SM, respectively.

FIGURE 4: The response of the SM.

FIGURE 5: Seven steps to implement the SM by FSM with one adder and one multiplier.

5. Structure of Fuzzy PI Controller

The structure of the fuzzy PI controller includes fuzzification, fuzzy inference mechanism, defuzzification, fuzzy knowledge base, fuzzy adjusting mechanism, and the PI controller (Figure 1).

The rotor speed (ω_r) is fed back to SLC for comparison with the output of SM and an error signal (e) is generated, and the signal changes over time (de). Based on the e and de signal, the fuzzy PI controller generates the signal to control the motor's speed. This signal is defuzzified and then amplified by the PID controller. When the motor had a heavy load or light load, the difference between rotor speed and output of SM increased, the fuzzy adjustment mechanism (figure 1) changes the knowledge base of the fuzzy PI controller. With this method, the motor speed is still in the balance.

Fuzzy PI controller used singleton fuzzy rule, and the method of the average center solution with input linguist value includes

$$e(k) = \omega_m(k) - \omega_r(k), \quad (26)$$

$$de(k) = e(k) - e(k-1). \quad (27)$$

In Figure 7, the values e , de , and u_f are the input and output of fuzzy controller. Their linguist values E and dE are ($A_1, A_2, A_3, A_4, A_5, A_6, A_7$) and ($B_1, B_2, B_3, B_4, B_5, B_6, B_7$), respectively. During each cycle, there are only two E and dE values that are supplied to the fuzzy PI controller (Figure 7). From the values of E and dE , based on membership functions, we can calculate the values of $\mu_{A_i}(e)$ and $\mu_{B_i}(de)$ with the formula below:

$$\mu_{A_i}(E) = \frac{E_{i+1} - E}{2}, \quad (28)$$

$$\mu_{A_{i+1}}(E) = 1 - \mu_{A_i}(E), \quad (29)$$

$$\mu_{B_i}(dE) = \frac{dE_{i+1} - dE}{2}, \quad (30)$$

$$\mu_{B_{i+1}}(dE) = 1 - \mu_{B_i}(dE). \quad (31)$$

Membership function has the shape of the symmetrical triangle, and control rules have the form

$$\text{if } e \text{ is } A_m \text{ and } de \text{ is } B_n, \text{ then } u_f \text{ is } C_{m,n}. \quad (32)$$

Crisp value in the output of the fuzzy controller:

```

LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.numeric_std.all;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_signed.all;

LIBRARY lpm;
USE lpm.LPM_COMPONENTS.ALL;

ENTITY SM IS
port(CLK,CLK_40n :IN STD_LOGIC:=0';
      Wr          :IN STD_LOGIC_VECTOR(15 downto0);
      Wm          :BUFFER STD_LOGIC_VECTOR(15 downto0));
END SM;

ARCHITECTURE SM_archOF SM IS
SIGNAL a0,a1,a2,b1,b2 :STD_LOGIC_VECTOR(15 downto0);
SIGNAL Wm2,Wm1,wr1,wr2 :STD_LOGIC_VECTOR(15 downto0);
SIGNAL mula,mulb :STD_LOGIC_VECTOR(15 downto0);
SIGNAL mulr :STD_LOGIC_VECTOR(31 downto0);
SIGNAL adda,addb,addr :STD_LOGIC_VECTOR(15 downto0);
SIGNAL CNT :STD_LOGIC_VECTOR(7 downto0);

BEGIN

null: lpm_mult
generic map(LPM_WIDTHA=>16,LPM_WIDTHB=>16,LPM_WIDTHS=>16,
LPM_WIDTHHP=>32,LPM_REPRESENTATION=>"SIGNED",
LPM_PIPELINE=>1)
port map(dataa=> mula,datab=> mulb,clock=> clk,result=> mulr);

adder1: lpm_add_sub
generic map(lpm_width=>16,LPM_REPRESENTATION=>"SIGNED",
lpm_pipeline=>1)
port map(dataa=>adda,datab=>addb,clock=> clk,result=>addr);

--Q15
a0 <= X"007E";
a1 <= X"00FA";
a2 <= X"007E";
b1 <= X"966E";
b2 <= X"2B89";

GEN:block
BEGIN
PROCESS(CLK_40N)
BEGIN
IF CLK_40N'EVENT and CLK_40N='1' THEN
CNT <= CNT+1;
IF CNT=X"00" THEN
mula<= a0;
mulb<= wr;
ELSIF CNT=X"01" THEN
adda<= mulr(30 downto15);
mula<= a1;
mulb<= wr1;
ELSIF CNT=X"02" THEN
addb<= mulr(30 downto15);
mula<= a2;
mulb<= wr2;
ELSIF CNT=X"03" THEN
adda<= addr;
addb<= mulr(30 downto15);
mula<= b1;
mulb<= Wm1;
ELSIF CNT=X"04" THEN
adda<= addr;
addb<= mulr(30 downto15);
mula<= b2;
mulb<= Wm2;
ELSIF CNT=X"05" THEN
adda<= addr;
addb<= mulr(30 downto15);
ELSIF CNT=X"06" THEN
Wm <= (addr(15) & addr(13 downto0) & '0');
ELSIF CNT=X"08" THEN
Wm1 <= Wm;
Wm2 <= Wm1;
wr2 <= wr1;
wr1 <= wr;
CNT <= X"00";
END IF;
END IF;
END PROCESS;
END BLOCK GEN;
END SM_arch;

```

FIGURE 6: The example of SM computation using VHDL.

$$\begin{aligned}
u_f(E, dE) &= \frac{\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} c_{m,n} [\mu_{A_n}(E) * \mu_{B_m}(dE)]}{\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} \mu_{A_n}(E) * \mu_{B_m}(dE)} \\
&\triangleq \frac{\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} c_{m,n} * d_{n,m}}{\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} d_{n,m}} = \sum_{n=i}^{i+1} \sum_{m=j}^{j+1} c_{m,n} * d_{n,m},
\end{aligned} \quad (33)$$

where $d_{n,m} \triangleq \mu_{A_n}(E) * \mu_{B_m}(dE)$ and $c_{m,n}$ are adjustable parameters in the adaptation mechanism. Also, it can be obtained by the equations (28)–(31). Therefore, it is straightforward to obtain $\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} d_{n,m} = 1$ in (33).

The purpose of the fuzzy adjustment mechanism is to create the signal to adjust the fuzzy knowledge base of the fuzzy PI controller so that the error between the speed of the rotor and the output of the SM is the smallest. We use the method of gradient descent to determine the adaptive control law for the system.

Definition of instantaneous value function:

$$J(k+1) = \frac{1}{2} [e(k+1)]^2 = \frac{1}{2} [\omega_m(k+1) - \omega_r(k+1)]^2. \quad (34)$$

Parameter $C_{m,n}$ is determined by the variation of the instantaneous value function.

$$\Delta C_{m,n}(k+1) = -\alpha \frac{\partial J(k+1)}{\partial C_{m,n}(k)}, \quad (35)$$

in which α shows the learning rate of the system.

By taking Laplace and bilinear transform of (7), we obtained a discrete function in the Z domain of PMSM.

$$\frac{\omega_r(k)}{i_q^*(k)} = \frac{K_t}{F} \frac{(1 - e^{(-FT/J_m)})z^{-1}}{1 - e^{(-FT/J_m)}z^{-1}}, \quad (36)$$

in which T is a sampling cycle and z^{-1} is a stage of delay time. The equation describes the relationship between i_q^* and u_f .

$$i_q^*(k) = u_i(k-1) + (K_p + K_i) * u_f(k), \quad (37)$$

in which K_p and K_i are the coefficients of the PI controller and u_f is the output function of the fuzzy PI controller.

From (36) and (37), we get the function between ω_r and u_f :

FIGURE 7: The design of the fuzzy controller for PMSM drive.

$$\omega_r(k) = \Phi * \omega_r(k-1) + u_i(k-2) * \gamma + (K_p + K_i) * \gamma * u_f(k-1), \quad (38)$$

in which $\Phi = e^{(-FT/J_m)}$, $\gamma = K_t(1 - \Phi)/F$.

Variation of the instantaneous value function $J(k+1)$ from (34) is

$$\frac{\partial J(k+1)}{\partial C_{m,n}(k)} = -\alpha * e(k-1) \frac{\partial \omega_r(k+1)}{\partial u_f(k)} \frac{\partial u_f(k)}{\partial C_{m,n}(k)}. \quad (39)$$

From (35), (38), and (39), the parameters of the fuzzy PI controller $C_{m,n}$ in (33) could be adjusted through the function.

$$\Delta C_{m,n}(k) = \alpha * (K_p + K_i) * \gamma * e(k) * d_{m,n} \approx \alpha * (K_p + K_i) * \text{Sign}(\gamma) * e(k) * d_{m,n}. \quad (40)$$

6. Implementation on FPGA

The proposed algorithm has been embedded in one FPGA chip. The architecture of FPGA is shown in Figure 8. The FSM sequential programming technique was used to describe control algorithms. The internal circuit of the proposed algorithm includes one circuit of the SLC and one circuit of the CLC.

Figure 8(a) shows the implementation of FPGA for the PMSM speed controller. CLK and CLK_40ns are the input clocks (50 MHz and 25 MHz) to supply all modules. The command speed and rotor speed (ω_r^* , ω_r) with 16 bit length are the input of the SLC. The three-phase currents (i_a , i_b , i_c) and rotor flux angle (θ_e) with 12 bit length are the input of the CLC. The 6 PWMs are the output of SVPWM.

In Figure 8(b), there are 33 steps to carry out the fuzzy PI controller. The steps s_0 – s_5 compute SM, and the steps s_6 – s_7 calculate the rotor speed error with SM and error change. The steps s_8 – s_{11} compute the fuzzification. The step s_{12} looks up the fuzzy table to select four values of $c_{m,n}$ based on i , j obtained from s_{11} . The steps s_{13} – s_{21} compute the defuzzification function. The steps s_{22} – s_{32} turn the PI and turn the fuzzy rule parameters.

Figure 8(c) shows the design of SVPWM with 12 kHz frequency and 1.2 μ s dead band. The input of SVPWM is V_{ref1} , V_{ref2} , and V_{ref3} with 12 bit length.

Figure 8(d) computes the current vector controller, and it takes 24 steps. The steps s_0 – s_1 look up the table to take the value of sine and cosine functions. The steps s_2 – s_8 calculate the Clark and Park transformations. The PID controller turns the current on d and q -axis from step s_9 – s_{14} . The steps s_{15} – s_{23} calculate the inverse Park and modified inverse Clark transformations.

The data type in vector control and SVPWM for PMSM is 12 bits. The data type is designed with 16 bit length for the fuzzy PI controller and SM. Each clock pulse in the FPGA is 40 ns to complete a command. There are 33 steps (1.32 μ s) and 24 steps (0.96 μ s) to implement the SLC and the current vector controller, respectively. It does not lose any performance because the operation time of fuzzy PI with 1.32 μ s is much less than the sampling time of the speed controller (2 kHz).

Although the control algorithm of this research by using the fuzzy PI controller is quite complicated, using the FSM description in step-by-step order reduced the execution time of coding. The simulation was performed on Matlab/Simulink/ModelSim for adjusting and verifying the controller's parameters and checking the accuracy of the

FIGURE 8: The implementation of the proposed algorithm on FPGA. The FSM was used to describe the VHDL code. (a) The architecture of the FPGA-based fuzzy PI controller for PMSM drive. (b) The speed loop circuit. (c) The SVPWM architecture. (d) The vector control circuit.

FIGURE 9: The simulation system for SVPWM generation. (a) The block diagram of step 1. (b) The Simulink model of step 1. (c) The simulation result of PWM with RC circuit connection (saddle waveform).

proposed algorithm. The ModelSim software was embedded in Matlab/Simulink for simulating the whole system. The feedback signals, PMSM model, and inverter are designed on Simulink. After success in simulation, the whole VHDL code is downloaded to the FPGA board for testing again by experiment.

7. Simulation Work

The simulation is demonstrated in this section to confirm the effectiveness of the proposed architecture. In simulation work, five steps are used to confirm the effectiveness of the system. In the first step (Figure 9), the generation of

SVPWM function is firstly tested, and its block diagram includes inverse Clarke and inverse Park coordinate transformation, $\sin/\cos$ function, and SVPWM (Figure 9(a)). In Figure 9(b), the EDA simulator link for ModelSim executes the co-simulation using VHDL code, which describes the function of the block diagram in Figure 9(a). The switching frequency and dead band in SVPWM are designed with 16 kHz and 1.2 μs , respectively. The input signal i_q is set to a value varying from 0.1 to 0.4, i_d is chosen to be zero, and the address (θ_e) is generated from 0 to 360° with increment by one. The PWM1–PWM6 outputs can be generated through the computation in ModelSim. Further, PWM1, PWM3, and PWM5 are serially connected to the RC

FIGURE 10: Confirmation of the correctness of SVPWM on the PMSM model again. (a) The block diagram of step 2. (b) The rotor speed response. (c) The Simulink model.

circuit ($R = 10 \Omega$, $C = 47$). The saddle waveform of PWMs is shown in Figure 9(c).

In step 2, the inverter and PMSM are added to step 1. A real rotor flux angle replaces the address. The block diagram of this step is shown in Figure 10(a), and the Simulink model is shown in Figure 10(c). This step confirms the correctness of SVPWM on PMSM again. The rotor speed is shown in Figure 10(b), and it follows the value of i_q at the input.

In step 3, the PI speed controller is applied to the system, and the real rotor speed is fed back to the speed controller. The block diagram of this step is shown in Figure 11(a), and the Simulink model is shown in Figure 11(c). In this step, we can check the speed response of PMSM. The rotor speed can follow well the step response at the input (Figure 11(b)).

In step 4, the three-phase stator currents are fed back and applied through Clark and Park transformation. The results

FIGURE 11: The speed controller is applied to the system, and the real rotor speed is fed back to the speed controller. (a) The block diagram of step 3. (b) The rotor speed tracks the speed command perfectly. (c) Simulink model.

of that transformation are compared with i_d and i_q obtained from the motor. Then, i_d and i_q are fed back to the PI controller in the current loop to create the completed system. The block diagram of this step is shown in Figure 12(a), and the Simulink model is shown in Figure 12(b). In this step, the rotor speed also tracks the step response perfectly (Figure 12(c)). Figure 12(d) also validates the effectiveness of current vector control in Figure 1 ($i_d \approx 0$). The three-phase stator current is balanced (Figure 12(e)).

In step 5, the PI and fuzzy PI controllers are applied to check the overall system with three load cases. The Simulink model is shown in Figure 13. The VHDL code of the SLC is embedded in the ModelSim_1 block; the

VHDL code of the CLC is embedded in the ModelSim_2 block; the inverter based on IGBT and PMSM is designed on Matlab/Simulink. The fuzzy PI rule table is designed in Figure 7, and its parameters are shown in Figure 14. The learning rate is set to 0.5. The load of motor is in light-load ($J=0.000036$, $F=0.00043$) and heavy-load ($J=0.000324$, $F=0.0039$) conditions. The heavy-load condition is nine times heavier than light-load condition.

The sampling frequency of the CLC and SLC is designed with 16 kHz and 2 kHz, respectively. The clocks of 50 MHz and 12.5 MHz supply to all modules of ModelSim. The FPGA resource usage of SLC and CLC blocks is 4915 LEs

FIGURE 12: Continued.

FIGURE 12: The three-phase stator currents are fed back. (a) The block diagram of step 4. (b) The Simulink model. (c) The rotor speed tracks the speed command without overshoot. (d) $i_d \approx 0$ and the PMSM can control as DC motor. (e) The three-phase stator current balances.

FIGURE 13: The simulation model of the proposed algorithm on Matlab/Simulink.

FIGURE 14: The parameters of the fuzzy PI controller.

TABLE 3: The parameters of PMSM.

Parameters	Value
Stator resistor	1.3 Ω
Stator inductance	6.3 mH
Pole pairs	4
Inertia	$J = 0.000108 \text{ kg}\cdot\text{m}^2$
Friction factor	$F = 0.0013 \text{ N}\cdot\text{m}\cdot\text{s}$
Voltage constant	52.2 V _{peak} /1000 rpm
Torque constant	0.43169 N·m/A _{peak}

(logic elements) (0 RAM bits) and 1742 LEs (196,608 RAM bits), respectively. The designed PMSM parameters are listed in Table 3.

The article also analyzes the case that just uses the PI controller. k_i and k_p are designed at a normal load; the rotor speed tracks the output of SM perfectly (Figure 15(b)). The setting value of PI speed controller and PID on d -axis and q -axis is shown in Table 4. After that, the PMSM is changed to light-load and heavy-load conditions for testing. At the light-load condition, the response of rotor speed is overshoot to 43 rpm and the rotor runs faster than the normal-load condition (Figure 15(a)). At the heavy-load condition, the response of rotor speed is overshoot to 47 rpm and the rotor runs slower than the normal-load condition (Figure 15(c)).

The error between SM and rotor speed is maximum with 393 rpm (light-load condition) and 177 rpm (heavy-load condition).

The simulation results show that when using the fuzzy PI controller, the responding speed of the motor is obtained in the case of the light-load, normal-load, and heavy-load condition, and heavy-load (Figures 16–18). Although the load changed very largely, the rotor speed still tracked the output of the SM very well.

Also, from 0–0.1 s in Figure 19(a), the PI controller is used to control the speed of the motor in the light-load condition. The speed response has lag and overshoot. After 0.1 seconds, the fuzzy PI controller is applied, and the rotor speed tracks the SM perfectly, without overshoot. Similar to the heavy-load condition shown in Figure 19(b), from 0–0.15 s, the PI controller is used, and from 0.15 s–0.35 s, the fuzzy PI controller is used. The simulation result shows that with the proposed algorithm, the rotor speed can track the reference model well.

8. Experiment Work

After the successful simulation, we realize this code on the FPGA kit. The experimental system is shown in Figure 20, and the experimental results are shown in Figure 21–23.

FIGURE 15: The simulation results when the speed controller is the PI controller. (a) The light-load condition. (b) The normal-load condition. (c) The heavy-load condition.

TABLE 4: The parameters of PI speed controller and PID controller on d -axis and q -axis.

Parameters	Value
K_p speed controller	4600
K_i speed controller	27
K_p on d -axis	320
K_i on d -axis	5
K_d on d -axis	10
K_p on q -axis	224
K_i on q -axis	5
K_d on q -axis	10

The experimental system includes an FPGA (Cyclone IV EP4CE115F29C7N), a drive board, an encoder board, an ADC board, and a PMSM. The specifications of the motor are shown in Table 3. An encoder with a resolution of

2500 pps is attached to the motor as the rotor position sensor. The inverter has three sets of IGBT power transistors, and its specifications are shown in Table 5. The Toshiba IC (TLP250) is used in the gate driving circuit of IGBT. The PWM signals from the FPGA board are the inputs of the drive board. A Nios II embedded processor is set up in FPGA to construct a SOPC environment.

Figures 21–23 show the speed step response of the motor under an external load of 0 kg or 10 kg using the PI and fuzzy PI controller. The speed command is square waves with period of 0.5 s and magnitude variation from 0 $\Rightarrow$ 400 rpm $\Rightarrow$ 1000 rpm $\Rightarrow$ 1400 rpm $\Rightarrow$ 1000 rpm $\Rightarrow$ 1400 rpm $\Rightarrow$ 1000 rpm.

In Figure 21(a), at the light-load condition (0 kg), the speed controller is PI; the step response shows an excellent dynamic response with a rising time of 0.1 s, without

FIGURE 16: The light-load condition and the fuzzy PI controller are applied in simulation. (a) The speed response. (b) The current control.

FIGURE 17: The normal-load condition and the fuzzy PI controller are applied in simulation. (a) The speed response. (b) The current control.

FIGURE 18: The heavy-load condition and the fuzzy PI controller are applied in simulation. (a) The speed response. (b) The current control.

FIGURE 19: The PI and fuzzy PI controllers are applied in simulation. (a) The light-load condition. (b) The heavy-load condition.

FIGURE 20: The experimental system.

FIGURE 21: The PI controller without external load (light-load condition). (a) The rotor speed. (b) The current control.

overshoot, and a zero steady state. However, when heavy-load condition (10 kg) is added and the same PI controller is used, the speed dynamic response worsens and exhibits a 48 rpm overshoot in Figure 22(a). In Figure 23(a), a fuzzy PI controller is applied to solve this problem. The speed responses are highly improved without overshoot. Due to the EMI effect, the measured current in Figures 21(b), 22(b), and 23(b) have a little ripple.

Nevertheless, the experiment result still shows that the measured current could track the current command well.

Furthermore, it not only validates the function of the vector control but also could make the PMSM decouple. Therefore, the simulation and experiment results demonstrate the effectiveness and correctness of the proposed algorithm for PMSM.

9. Conclusion

The fuzzy PI controller has been studied to control the speed of PMSM. The adjustment mechanism produces a

FIGURE 22: The PI controller with external load = 10 kg. (a) The rotor speed. (b) The current control.

FIGURE 23: The fuzzy PI controller with external load = 10 kg. (a) The rotor speed. (b) The current control.

TABLE 5: The parameters of IGBT.

Parameters	Value
Collector-emitter voltage	600 V
Gate-emitter voltage	± 20 V
DC collector current (rated)	25 A
DC collector current in a short time (1 ms)	50 A

compensation signal by using (40), which is added to the fuzzy controller's knowledge base to force the system to track the SM. The simulation and experiment results confirmed the efficiency of the proposed fuzzy PI and SVPWM for PMSM.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

- [1] Y. S. Kung, N. Vu Quynh, C. C. Huang, and L. C. Huang, "Simulink/ModelSim co-simulation of sensorless PMSM speed controller," in *Proceedings of the 2011 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2011)*, pp. 24–29, Langkawi, Malaysia, September 2011.
- [2] C. Tseng, R. Jan, and J. Liaw, "Adaptive control design for permanent magnet synchronous motor with uncertain parameters: an LMI approach," in *Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics*, pp. 3647–3652, Taipei, Taiwan, October 2006.
- [3] J. Liu, S. Cao, Y. Yi, and Y. Fang, "Adaptive control for permanent magnet synchronous motor based on disturbance observer," in *Proceedings of the 2017 36th Chinese Control Conference (CCC)*, pp. 3533–3537, Dalian, China, July 2017.
- [4] A. Iqbal, H. Abu-Rub, and H. Nounou, "Adaptive fuzzy logic-controlled surface mount permanent magnet synchronous motor drive," *Systems Science & Control Engineering*, vol. 2, no. 1, pp. 465–475, 2014.
- [5] N. V. Quynh and Y. Kung, "FPGA-realization of fuzzy speed controller for PMSM drives without position sensor," in *Proceedings of the 2013 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 278–282, Nha Trang, Vietnam, November 2013.
- [6] C.-K. Lai, Y.-T. Tsao, and C.-C. Tsai, "Modeling, analysis, and realization of permanent magnet synchronous motor current vector control by MATLAB/Simulink and FPGA," *Machines*, vol. 5, no. 4, p. 26, 2017.
- [7] 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 Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 265–271, 2007.
- [8] S. T. Brassai, L. Bako, G. Pana, and S. Dan, "Neural control based on RBF network implemented on FPGA," in *Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment*, pp. 41–46, Brasov, Romania, May 2008.
- [9] 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," in *Proceedings of the 2014 International Conference on Information Science, Electronics and Electrical Engineering*, pp. 1488–1492, Sapporo, Japan, April 2014.
- [10] J. S. Kim and 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.
- [11] J. Yu, J. Gao, Y. Ma, H. Yu, and S. Pan, "Robust adaptive fuzzy control of chaos in the permanent magnet synchronous motor," *Discrete Dynamics in Nature and Society*, vol. 2010, Article ID 269283, 13 pages, 2010.
- [12] Y. M. Alsayed, A. Maamoun, and A. Shaltout, "High performance control of PMSM drive system implementation based on DSP real-time controller," in *Proceedings of the 2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pp. 225–230, Aswan, Egypt, February 2019.
- [13] A. Maamoun, Y. M. Alsayed, and A. Shaltout, "Fuzzy logic based speed controller for permanent-magnet synchronous motor drive," in *Proceedings of the 2013 IEEE International Conference on Mechatronics and Automation*, pp. 1518–1522, Takamatsu, Japan, August 2013.