ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20231122001137
Original scientific paper

FPGA Implementation of Master-Slave Servo On-Chip Control with Active Disturbance Rejection

Dechun ZHENG, Jiliang XU*, Li XU

Abstract: In order to improve the anti-disturbance ability of the system and the running speed of PID controller, this paper proposes a design method of master-slave servo on-chip control system with active disturbance rejection control. The active disturbance rejection control (ADRC) technology is introduced into the system, and the friction and interference factors in the system are summed up as disturbances, which are equivalent to the estimation problem of disturbances, and the anti-disturbance ability of the system is improved. The current loop vector control algorithm is normalized, and the parallel computing unit composed of multiplier and adder is used to complete the current loop vector control algorithm under the control of the state machine, which not only improves the running speed of the current loop vector controller, but also reduces the requirement of multiplier resources for FPGA. The experimental results show that the PID controller with ADRC can improve the noise suppression ability of the system to a certain extent. The whole system can be integrated in a low-end FPGA system to reduce the cost of the system, and the current loop vector control algorithm can be completed in 2 us.

Keywords: disturbance rejection control; FPGA; on-chip systems; parallel processing units; vector controller

1 INTRODUCTION

Servo motors, characterized by a wide controllable range, high reliability, excellent flexibility, outstanding stability, and high power factor, have found extensive applications in various industries such as military, aerospace, precision assembly, and textiles. Currently, various control algorithms, such as PID algorithm, fractional-order PID algorithm, fuzzy control, sliding mode control, adaptive control, etc., have been applied in the field of servo motor control [1-8]. In industrial field, the PID control algorithm is widely adopted due to its simplicity, ease of implementation, and good reliability. In particular, some PID controllers implemented on FPGA implement a specific hardware architecture in a programmable environment according to the control performance indicators, so that the PID control algorithm can operate in a very short time and retain the potential parallelism of the selected algorithm [9-11]. One of the more classic is the FPG-based AC motor current controller developed by Monmasson and his team. They have implemented a switching current controller, a proportional-integral (PI) current controller and a predictive current controller for synchronous motors on FPGA. The PID control algorithm, however, has its limitations. Firstly, although PID control performs well in controlling linear systems, its effectiveness is compromised when dealing with nonlinear systems or systems subjected to strong disturbances. Secondly, the PID control algorithm's parameters cannot be adjusted in real-time, and it fails to promptly respond to changes in system parameters caused by environmental variations or factors such as friction. With the continuous advancement of intelligent technology, the PID algorithm has undergone theoretical upgrades, and the application of intelligent optimization algorithms to online tuning of PID parameters has achieved significant improvements in performance. Researchers like Zhu and Wu [12] proposed a stable speed control system for a direct current servo motor based on an incremental PID algorithm, which was implemented on an AT89S52 processor. Hong et al. [13] designed a self-driving precision compass based on a BP neural network and PID control, achieving precise control of a micro DC servo motor on STM32F103C8T6. Wang et al. [14] utilized a serial PID controller to manage flight attitudes and realized control requirements for takeoff, hovering, and landing modes of a rotorcraft on an STM32F1 processor. Although the aforementioned system has been implemented on a general-purpose processor and achieved certain results, challenges arise in further optimizing and enhancing the algorithm due to the serial computation nature and limited computational power. With the development of large-scale integrated circuit design technology and FPGA devices, the hardware implementation of control algorithms has become feasible. Liu et al.[15] utilized Verilog language to design a neural network PID controller, and applied it to the motion control of service robots, thereby enhancing the control stability and accuracy of service robots. Wang et al. [16] proposed a method for adaptive online tuning of PID controller parameters using neural networks, and this system was implemented on FPGA [17]. Rosa et al.[18] utilized a genetic algorithm to optimize PID controller parameters, implementing both genetic algorithm and PI regulator on a Nois II soft processor. The experimental outcomes indicate a significant improvement in the system's stability [19]. While many control systems combining various intelligent algorithms with PID control have been not only successfully implemented and validated in laboratory experiments, but also effectively enhanced the performance of PID controllers, their large-scale industrial application remains limited. The traditional PID controller and some improved PID controllers can be implemented by DSP, DSP + FPGA and FPGA [20, 21]. Because DSP is a serial execution mode, its execution efficiency is low, and it can only be used in some occasions with low real-time requirements. Although DSP+FPGA implements the system to combine the advantages of the two, there is also a need to consume more FPGA IO resources, and there is also a communication loss between the two, which also affects the real-time performance of the system. By using master-slave architecture to implement the system on FPGA, the loss of IO resources and communication is avoided, so that the real-time performance of the system is the best. The traditional PID

controller and some improved PID controllers can be implemented by DSP, DSP+FPGA and FPGA. Because DSP is a serial execution mode, its execution efficiency is low, and it can only be used in some occasions with low real-time requirements. Although DSP+FPGA implements the system to combine the advantages of the two, there is also a need to consume more FPGA IO resources, and there is also a communication loss between the two, which also affects the real-time performance of the system. By using master-slave architecture to implement the system on FPGA, the loss of IO resources and communication is avoided, so that the real-time performance of the system is the best. This system proposes a method for implementing PID controllers using a master-slave dual-core processor. The PID controller is designed with a three-tier closed-loop structure, where the velocity inner loop and position outer loop are implemented by software in the Nois II soft processor. The current loop is implemented by the slave processor. The current loop is configured with a PI control method. However, due to the manual tuning requirement of the PI controller and its limited disturbance rejection capability, an active disturbance rejection controller is introduced in this design. During the design and debugging process, to overcome the constraints posed by the FPGA's multiplier module resources, a state machine approach was utilized to design the slave processor module. In summary, the workflow of this paper unfolds as follows: the second section provides a detailed description of the on-chip system composition of the servo motor controller; the third section introduces the active disturbance rejection controller technology; the fourth section elucidates the normalization of vector control in the current loop; the fifth section designs the core unit for vector control in the current loop; the sixth section presents the experimental results, and finally, the seventh section presents the conclusions drawn from this study.

2 COMPOSITION OF THE SYSTEM-ON-CHIP PID CONTROLLER

The hardware of the PID controller on-chip system primarily consists of the Nios II soft core processor, current loop vector control unit, phase current sampling, motor rotor position and speed analysis, and communication interface with the upper computer, as illustrated in Fig. 1. In terms of structure, the entire system can be regarded as a dual-core processing system with a master-slave architecture. The Nios II soft core serves as the master processing system. Apart from handling real-time control tasks related to outer loops comprised of slower-paced velocity and position control, it is also responsible for communication modules with the upper computer, overall system real-time monitoring, and computation for active disturbance rejection control. On the other hand, the spatial vector control functions as a high-speed slave processing system, primarily executing the current loop vector control algorithms.The current loop vector control adopts PI control, but the parameters of the PI controller cannot be automatically tuned. Therefore, when system parameters such as friction coefficient change, it definitely affects the system's performance. Introducing active disturbance rejection control (ADRC) technology into the current loop, the uncertain variations of system parameters are abstracted as disturbances. Consequently, the control of time-varying systems is equivalently transformed into estimating disturbances. This is achieved by employing an extended state observer for observation and implementing an error feedback controller. Spatial vector control and various peripherals of the system are mounted onto the Nios II main processor via high-speed Avalon bus.

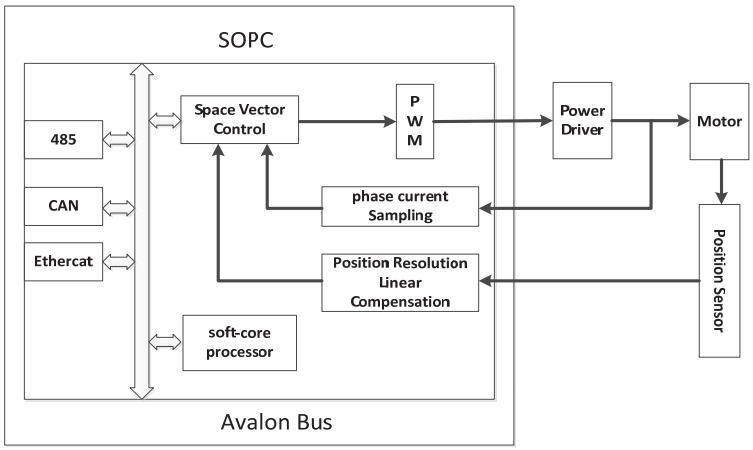

Figure 1 Architecture diagram of servo control system based on system-on-chip (SoC)

3 ACTIVE DISTURBANCE REJECTION CONTROL AND CURRENT LOOP VECTOR DESIGN

3.1 Active Disturbance Rejection Control

Due to the presence of uncertainties in the system such as friction coefficients and variations in system parameters, coupled with inter-axis coupling between the dq axes even after the motor is in operation, higher demands are imposed on the performance of the current loop vector controller. In this system, active disturbance rejection control technology is introduced, treating factors such as friction and disturbances as disturbances. This approach equivalently transforms the control problem of time-varying systems into an estimation problem of disturbances. The extended state observer is employed for disturbance observation, and ultimately, error feedback control is implemented. The block diagram of the system is depicted in Fig. 2. The equations of the extended state observer in Fig. 2 are represented as Eq. (1), and its output is connected to the input of the q-axis reference current.

$$\begin{cases} e_2 = z_1 - \omega \\ \dot{z}_1 = z_2 - \beta_1 e_2 + bu \\ \dot{z}_2 = -\beta_2 e_2 \end{cases} \quad (1)$$

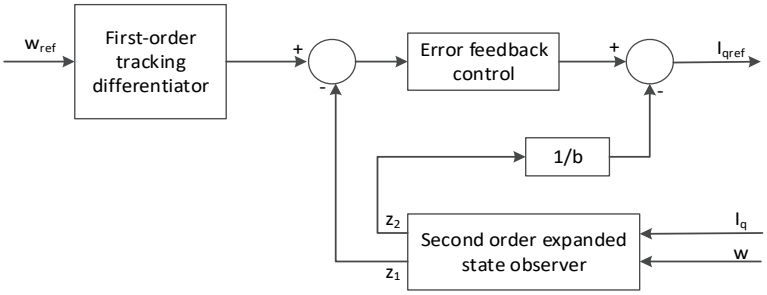

Figure 2 Block diagram of active disturbance rejection control

3.2 The Normalization of Vector Control

A commonly used motion control block diagram for servo motors is shown in Fig. 3. The control algorithms for the outer loop (velocity loop and position loop) are implemented in software using the Nios II soft core processor. A detailed discussion of this implementation is beyond the scope of this paper. The vector control unit is implemented on an FPGA with dedicated multipliers. In the diagram, $\theta_{e,n}$ and $w_{e,n}$ represent the motor rotor angle and speed respectively. P and I are the proportional and integral regulators of the current loop. The current vector control unit first acquires phase currents, then undergoes Clarke and Park transformations for flux decoupling. Thus, after achieving i_{sdref} = 0 rotor flux-oriented control, i_{sdref} linear control of motor torque is attained. However, during the actual operation of the motor, there may still exist coupling effects between the dq axes. Additionally, certain system parameters of the motor undergo slight variations during operation, particularly factors such as friction coefficients. In this complex scenario, a single PI controller cannot effectively handle the situation. Therefore, this system incorporates the active disturbance rejection control technology. The technique, based on given parameters such as speed, collects real-time information about the system's speed and current. Through an extended state observer and error feedback control, it generates the reference current for the q-axis, which is then fed into the q-axis PI controller in the vector control. The calculation is implemented by the Nios II soft core program.

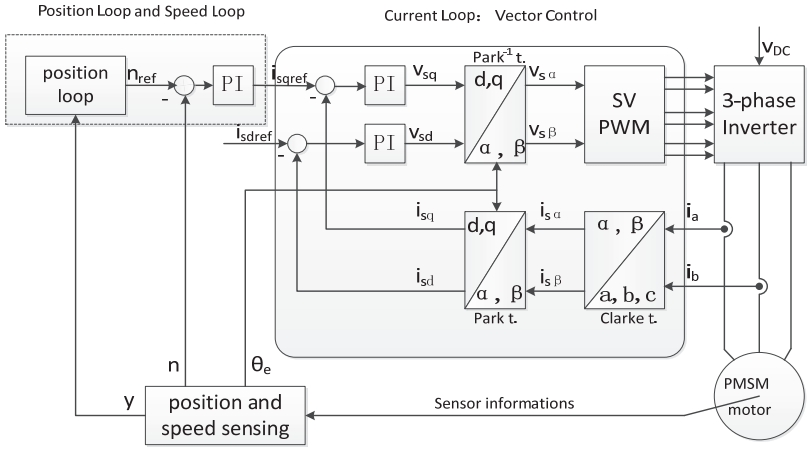

Figure 3 Block diagram of servo motion control system

The current vector control algorithm is primarily composed of the Clarke transformation module, Park transformation module, PI regulator module, Park inverse transformation module, and space vector pulse width modulation (SVPWM) in sequence. According to reference [19], the calculations for the Clarke transformation module, Park transformation module, PI regulator module, and Park inverse transformation module are accomplished through four independent multiplication operations and two parallel addition operations, which can be implemented using the multiplication-accumulators available in FPGA resources. However, SVPWM is relatively complex and requires further elaboration for its implementation. The calculation process of SVPWM can be divided into computational units, including SVPWM spatial sector computation, SVPWM theoretical switch computation, and SVPWM practical pulse width modulation computation, as demonstrated in Eq. (2) to (4).

1) SVPWM Sector Identification:

$$\begin{aligned} V_a &= V_{s\beta} \\ V_b &= \sqrt{3}V_{sa} - V_{s\beta} \\ V_c &= -\sqrt{3}V_{sa} - V_{s\beta} \end{aligned} \quad (2)$$

$$A = \begin{cases} 1, V_a > 0 \\ 0, V_a < 0 \end{cases}; B = \begin{cases} 1, V_b > 0 \\ 0, V_b < 0 \end{cases}; C = \begin{cases} 1, V_c > 0 \\ 0, V_c < 0 \end{cases}$$

2) Calculate the theoretical switching times for SVPWM in each phase

$$
\begin{aligned}
&V_{DC\_intT} = \frac{PWMPRD}{V_{dc}} \\
&X = \sqrt{3} \cdot V_{DC\_intT} \cdot V_{s\beta} \\
&Y = \frac{\sqrt{3}}{2} \cdot V_{DC\_intT} \cdot V_{s\beta} + \frac{3}{2} \cdot V_{DC\_intT} \cdot V_{sa} \\
&Z = \frac{\sqrt{3}}{2} \cdot V_{DC\_intT} \cdot V_{s\beta} - \frac{3}{2} \cdot V_{DC\_intT} \cdot V_{sa} \\
&t_1 = \begin{cases} Z & Sect\_No = \ = 1 \\ Y & Sect\_No = \ = 2 \\ -Z & Sect\_No = \ = 3 \\ -X & Sect\_No = \ = 4 \\ X & Sect\_No = \ = 5 \\ Y & Sect\_No = \ = 6 \end{cases} \\
&t_2 = \begin{cases} Y & Sect\_No = \ = 1 \\ -X & Sect\_No = \ = 2 \\ X & Sect\_No = \ = 3 \\ Z & Sect\_No = \ = 4 \\ -Y & Sect\_No = \ = 5 \\ -Z & Sect\_No = \ = 6 \end{cases}
\end{aligned} \tag{3}
$$

PWMPRD represents the pulse width modulation period and V_{dc} corresponds to the bus voltage.

3) Calculate the actual pulse width modulation times for each phase in SVPWM:

$$
\begin{aligned}
&t_{1SAT} = t_1 \cdot \frac{PWMPRD}{t_1 + t_2};\ t_{2SAT} = t_2 \cdot \frac{PWMPRD}{t_1 + t_2} \\
&t_{1s} = \begin{cases} t_1;\ t_1 + t_2 \le PWMPRD \\ t_{1SAT};\ t_1 + t_2 > PWMPRD \end{cases} \\
&t_{2s} = \begin{cases} t_2;\ t_1 + t_2 \le PWMPRD \\ t_{2SAT};\ t_1 + t_2 > PWMPRD \end{cases} \\
&t_{aon} = \frac{PWMPRD - t_{1s} - t_{2s}}{2} \\
&t_{bon} = t_{aon} + t_{1s} \\
&t_{con} = t_{bon} + t_{2s}
\end{aligned} \tag{4}
$$

t_{aon}, t_{bon} and t_{con} are the conduction times of phases U, V, and W respectively, denoted as A, B, and C respectively, and are illustratedin Tab. 1.

Table 1 Modulation time of each phase in different sectors

Sect_No	1	2	3	4	5	6
CMPA	t_{bon}	t_{aon}	t_{aon}	t_{con}	t_{con}	t_{bon}
CMPB	t_{aon}	t_{con}	t_{bon}	t_{bon}	t_{aon}	t_{con}
CMPC	t_{con}	t_{bon}	t_{con}	t_{aon}	t_{bon}	t_{aon}

In order to enhance the implementation of the current loop vector control unit on FPGA, it is necessary to perform computations for modules including Clarke transformation, Park transformation, PI regulator, Park inverse transformation, Space Vector Pulse Width Modulation (SVPWM) sector calculation, SVPWM theoretical switching calculation, and actual pulse width modulation calculation, according to the multiply-accumulate architecture as depicted in Eq. (5). Furthermore, the algorithms for each step need to undergo normalization, as illustrated in Eq. (6) to Eq. (12).

$$
\begin{aligned}
&y_1 = c_1 \cdot x_1 + c_2 \cdot x_2 \\
&y_2 = c_3 \cdot x_3 + c_4 \cdot x_4
\end{aligned} \tag{5}
$$

(1) Clarke Transformation:

$$
\begin{aligned}
&c_1 = 1;\ c_2 = 0;\ x_1 = i_a;\ x_2 = 0; \\
&c_3 = \frac{1}{\sqrt{3}};\ c_4 = \frac{2}{\sqrt{3}};\ x_3 = i_a;\ x_4 = i_b; \\
&i_{sa} = y_1;\ i_{s\beta} = y_2
\end{aligned} \tag{6}
$$

In this transformation, i_a and i_b represent the currents flowing through the U-phase and V-phase of the motor respectively.

(2) Park Transformation:

$$
\begin{aligned}
&c_1 = \cos(\theta_e);\ c_2 = \sin(\theta_e);\ x_1 = i_{sa};\ x_2 = i_{s\beta} \\
&c_3 = -\sin(\theta_e);\ c_4 = \cos(\theta_e);\ x_3 = i_{sa};\ x_4 = i_{s\beta} \\
&i_{sd} = y_1;\ i_{sq} = y_2
\end{aligned} \tag{7}
$$

In this transformation, θ_e denotes the motor angle.

(3) PI Controller (Implemented in Incremental Mode):

$$
\begin{aligned}
&c_1 = K_{d_{new}};\ c_2 = K_{d_{old}};\ x_1 = e_d(k); \\
&x_2 = e_d(k-1);\ V_{sd}(k) = V_{sd}(k-1) + y_1 \\
&V_{sq}(k) = V_{sq}(k-1) + y_2
\end{aligned} \tag{8}
$$

In this context, K_{d_new}, K_{d_old}, K_{q_new} and K_{q_old} are K_p parameters of the PI controller, determining by $K_i.V_{sd}(k)$, $V_{sd}(k)$, $e_d(k)$ and $e_q(k)$ represent the current control cycle values, while $V_{sd}(k-1)$, $V_{sq}(k-1)$, $e_d(k-1)$, $e_q(k-1)$ denote the values from the previous control cycle.

(4) Park inverse transformation:

$$
\begin{aligned}
&c_1 = \cos(\theta_e);\ c_2 = -\sin(\theta_e);\ x_1 = V_{sd};\ x_2 = V_{sq}; \\
&c_3 = \sin(\theta_e);\ c_4 = \cos(\theta_e);\ x_3 = V_{sd};\ x_4 = V_{sq}; \\
&V_{sa} = y_1;\ V_{s\beta} = y_2
\end{aligned} \tag{9}
$$

(5) SVPWM Sector in Spatial Domain:

$$
\begin{aligned}
&c_1 = \sqrt{3};\ c_2 = -1;\ x_1 = V_{sa};\ x_2 = V_{s\beta}; \\
&c_3 = -\sqrt{3};\ c_4 = -1;\ x_3 = V_{sa};\ x_4 = V_{s\beta}; \\
&V_a = V_{s\beta};\ V_b = y_1;\ V_c = y_2
\end{aligned} \tag{10}
$$

The logic judgment and Sect_No are implemented in additional logic units.

(6) computation of the theoretical switching times for SVPWM phases:

$$
\begin{aligned}
&c_1 = \frac{\sqrt{3}}{2}\cdot V_{DC_{int}T};\ c_2 = \frac{3}{2}\cdot V_{DC_{int}T};\\
&c_3 = \frac{\sqrt{3}}{2}\cdot V_{DC_{int}T};\ c_4 = -\frac{3}{2}\cdot V_{DC_{int}T};\\
&x_3 = V_{s\beta};\ x_4 = V_{sa};\\
&X = y_1 + y_2;\ Y = y_1;\ Z = y_2
\end{aligned}
\tag{11}
$$

(7) calculation of the actual pulse width modulation times for SVPWM phases:

$$
\begin{aligned}
&c_1 = \frac{PWMPRD}{t_1 + t_2};\ c_2 = 0;\ x_1 = t_1;\ x_2 = 0;\\
&c_3 = \frac{PWMPRD}{t_1 + t_2};\ c_4 = 0;\ x_3 = t_2;\ x_4 = 0;\\
&t_{1SAT} = y_1,\ t_{2SAT} = y_2
\end{aligned}
\tag{12}
$$

$\frac{PWMPRD}{t_1 + t_2}$ is computed independently using a dedicated divider, whereas t_{aon}, t_{bon} and t_{con} along with logical assessments, are computed separately in additional dedicated logic units.

3.3 The Design of Vector Control Core Unit

The operational core unit for vector control is designed as depicted in Fig. 4. Due to the sequential nature of operations in the current vector controller, the operational core unit is time-multiplexed under the control of a state machine. It performs computations step by step according to the designated sequence. This process continues until all seven steps of vector control calculations are completed within one clock cycle. In the state machine timing diagram shown in Fig. 5, the state variables S[2..0] are illustrated. State 0 is designated for the conversion of motor rotor angle to electrical angle and compensation of magnetic pole angle. Triggered by the control clock cycle, the states S[2..0] corresponding to the 7 steps of vector control are generated, along with the latch signals for storing intermediate variables. The state variables are utilized by the control selector to choose inputs corresponding to the current state for rapid vector computations of the current step. Upon the falling edge of the latch signal, the current computation results are latched and stored in the register selected by the state variables, serving as intermediate variables required for the subsequent clock cycle. In state four, the C selector and the X selector apply operation $c_1 = \cos(\theta_e);\ c_2 = -\sin(\theta_e);\ x_1 = V_{sd};\ x_2 = V_{sq};$ $c_3 = \sin(\theta_e);\ c_4 = \cos(\theta_e);\ x_3 = V_{sd};\ x_4 = V_{sq}$; thus, upon completion of state two, $V_{sa} = y_1$; $V_{s\beta} = y_2$ is obtained and stored in the corresponding register to serve as the input for the subsequent computation in state five.

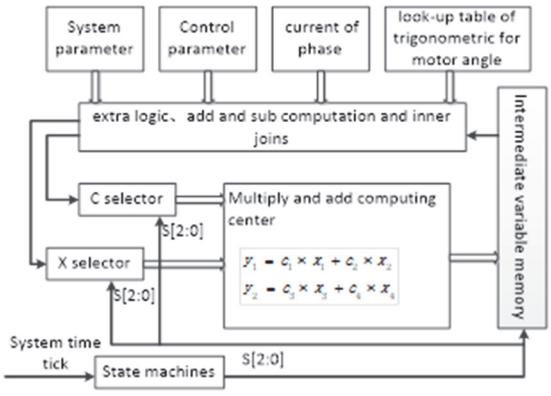

Figure 4 Block diagram of vector control core unit

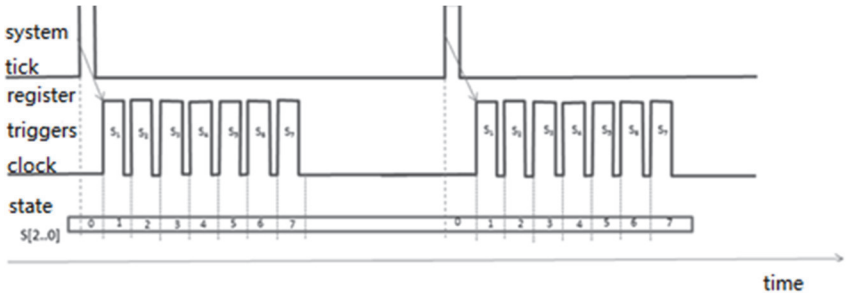

Figure 5 Timing state of vector control core unit

4 EXPERIMENTAL RESULTS

The vector control of the current loop and the vector control of the current loop with active disturbance rejection control were simulated separately using MATLAB. The simulation results of the ADRC velocity controller are illustrated in Fig. 6. In the figure, the yellow waveform represents the load torque; the blue waveform represents the observed disturbance; the brown waveform represents the desired velocity; and the green waveform corresponds

to the velocity feedback. The feedback speed reached 90% of the desired speed, achieving a settling time of less than $5e^{-3}$ second. At 0.05 seconds, the desired speed transitioned from 20 to 40 in a step response. The feedback speed approximately reached the desired speed within $5e^{-3}$ second. Despite disturbances affecting the torque, the system quickly converged to the load torque, and the steady-state speed error was maintained below 0.01 rad/s.

Figure 6 The simulation results of active disturbance rejection speed controller

In order to reduce the cost of the system and meet the needs of system design, EP4CE30F23C8 produced by altera company is used as the core of the system in this design. The chip contains 28848 logic cells, 594 kbit embedded memory cells, 66 18 × 18 multiplication cells and 4 PLL resources. The current loop vector controller is designed according to the common design method, which needs more than 20 multiplication units, while the method combining state machine and normalization only needs 4 multiplication units. The system was implemented on an Altera FPGA based on EP4CE30F23C8. The experimental platform is illustrated in Fig. 7, which consists of three layers. The first layer comprises the FPGA core board, the second layer includes peripheral devices and signal conditioning, and the third layer involves power supply and power drive components. The servo control system, with the servo motor as the controlled object, has a phase resistance of 3.4 Ω, a phase inductance of 10.8 mh, rated speed 3000 rpm, rated torque 1.9 Nm, and a maximum power of 850 W. The motor feedback encoder adopts a 2500-line incremental encoder. The entire servo control system (Nios II with current loop vector control unit) consumes 21041 logic gates and 417306 storage units. Among them, the current vector control unit consumes 9621 logic gates, as illustrated in Fig. 8.

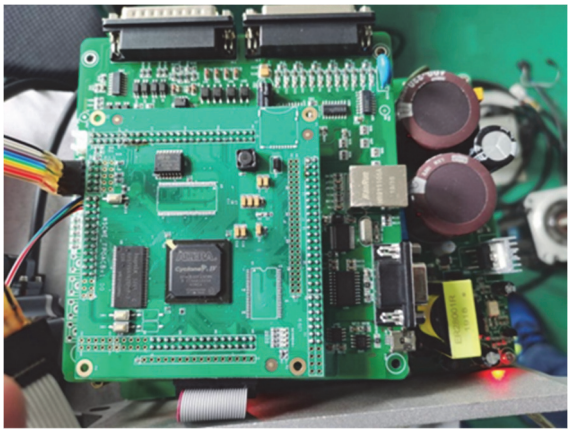

Figure 7 Experimental platform

Entity/Instance	Logic Cells	Dedicated Logic Registers	I/O Registers	Memory Bits	M9Ks	DSP Elements	LUT-Only LCs	ister-Only	/Register
Cyclone IV E: EP4CE30F23C8									
NOSII_TOP	21041 (53)	7491 (0)	64 (64)	417306	59	44	13550 (53)	3080 (0)	4411 (0)
sld_hub:auto_hub	161 (1)	85 (0)	0 (0)	0	0	0	76 (1)	14 (0)	71 (0)
Core_S:inst	4675 (2)	2733 (0)	0 (0)	317568	44	6	1942 (2)	602 (0)	2131 (0)
Single_Current_Loop:inst1	9621 (2)	1520 (0)	0 (0)	98304	12	32	8101 (2)	762 (0)	758 (0)
CLP_Core_Pack:inst	3368 (2)	1053 (0)	0 (0)	0	0	32	2315 (2)	712 (0)	341 (0)
stage4:inst1	195 (0)	0 (0)	0 (0)	0	0	0	134 (0)	0 (0)	61 (0)
stage5:inst3	288 (0)	0 (0)	0 (0)	0	0	0	218 (0)	0 (0)	70 (0)

Figure 8 SoC servo control consumption of resources

During the operation of the current loop vector control unit with active disturbance rejection control, one complete cycle of current loop vector control involves transitions between State 1 to State 7, as illustrated in Fig. 9. The high-level portion of signal 1 in the figure represents the operational time, with a grid interval of 500 ns. Therefore, the duration for one complete cycle of current loop vector control is approximately 2 us, significantly less than 5 us. The red segments denote the transition signals of the state machine. State 0 commences from the falling edge of the red signal and State 7 initiates from the last rising edge of the red signal.

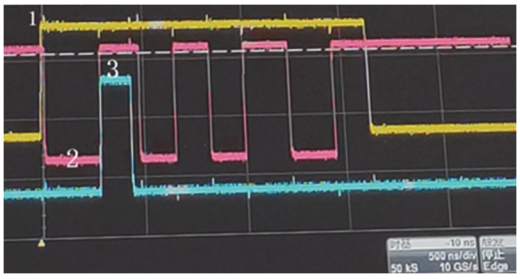

Figure 9 Actual sequence diagram of current loop vector control

A series of tests at different speeds and torques was carried out. In order to better illustrate the speed and position control characteristics of the motor, the motor royal mansion motion test is used. Fig. 10 to Fig. 14 present the experimental data collected in real-time during the reciprocating motion of the servo motor. In Fig. 10 and Fig 11, the tracking performance of currents on the d-axis and q-axis under directional magnetic field control is demonstrated. From Fig. 11, it can be observed that the decoupling of measured currents and positions results in I_{d_cal} close to zero, enabling I_{q_cal} rapid tracking of reference values I_{d_cal}. This indicates excellent linearity in the

system's magnetic linkage after decoupling and demonstrates superior dynamic performance in vector control. The position and velocity tracking curves shown in Fig. 12 and Fig. 13 demonstrate excellent dynamic performance and repeatable positioning accuracy in the servo motion control. It can be seen from the figure that the forward and reverse speed are accurately controlled at 0 ~ 3000 rpm, the starting position of the reciprocating motion position is at 145, and the ending position is at 6000, indicating that the speed control and position control characteristics of the motor are good. Fig. 14 displays real-time sampled U and V phase currents during acceleration and deceleration, indicating the vector controller's smooth commutation capability using SVPWM. Furthermore, it illustrates the effective noise suppression capability of the digital current sampling filter.

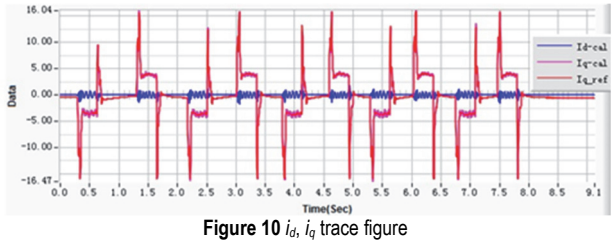

Figure 10 i_d, i_q trace figure

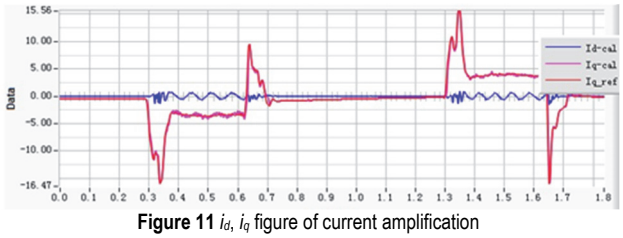

Figure 11 i_d, i_q figure of current amplification

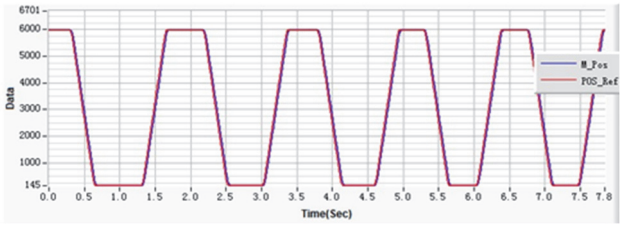

Figure 12 Figure of position trace

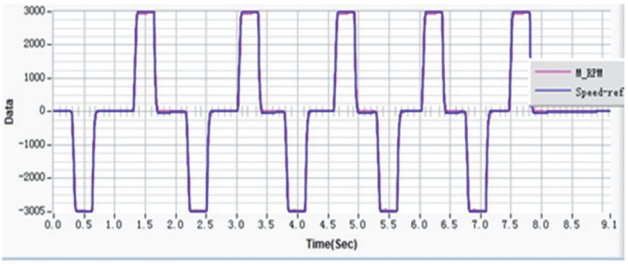

Figure 13 Speed trace figure

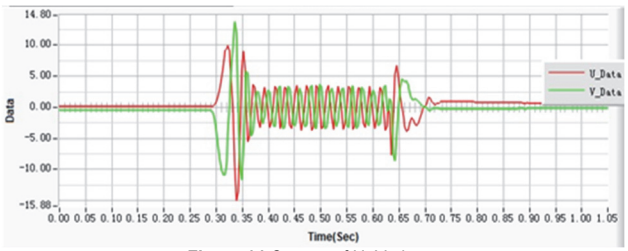

Figure 14 Current of U, V phase

5 CONCLUSIONS

The study has successfully implemented a complete servo motor control system with active disturbance rejection control on Altera Cyclone series FPGAs. The stability, accuracy, and speed of the entire servo drive system meet the design requirements. This System-on-Chip (SoC)-based control system integrates a microprocessor, digital filtering for current sampling, vector control, encoder decoding, and communication with the host computer. Not only does it ensure high-performance motion control, but it also exhibits excellent scalability and adaptability. Moreover, it is compatible with various digital encoders. By integrating the features of FPGA parallel computing and the precise matching algorithm, the time-division multiplexing technique not only conserves FPGA resources but also achieves outstanding performance in completing the vector control algorithm within 2us. Experimental results demonstrate that the high-speed current vector control unit based on a state machine effectively enhances the system's dynamic performance, suppresses current noise, and mitigates interference between control cycles. However, the system still has some shortcomings: when the noise such as current or external interference is large, the dynamic performance of the system will be affected to a certain extent. In the next research plan, the neural network online learning is implemented in the main processor, and the PID parameters are adjusted in real time to further improve the system performance.

Acknowledgement

This work was supported by Basic public welfare research in Zhejiang (LGG19F030007), supported by Natural Science Foundation of Zhejiang Province (LY19F020008), supported by scientific research project of Zhejiang Education Department (Y201635575), supported by the teachers scientific research funds of Zhejiang University City College (JYB18003).

5 REFERENCES

[1] Youness, H., Moness, M., & Khaled, M. (2014). MPSoCs and multicore microcontrollers for embedded PID control: A detailed study. *IEEE Transactions on Industrial Informatics, 10*(4), 2122-2134. https://doi.org/10.1109/TII.2014.2355036

[2] Roy, A., Sharma, L., Chakraborty, I., Panja, S., Ojha, V. N., & De, S. (2019). An FPGA based all-in-one function generator, lock-in amplifier and auto-relockable PID system. *Journal of Instrumentation, 14*(05), P05012. https://doi.org/10.1088/1748-0221/14/05/P05012

[3] Ghane, R. G. & Hassan, M. Y. (2023). Advanced hybrid nonlinear control for morphing quadrotors. *Mathematical Modelling of Engineering Problems*, *10*(4), 1216-1224. https://doi.org/10.18280/mmep.100414

[4] Ray, P. K., Paital, S. R., Mohanty, A., Foo, Y. E., Krishnan, A., Gooi, H. B., & Amaratunga, G. A. (2019). A hybrid firefly-swarm optimized fractional order interval type-2 fuzzy PID-PSS for transient stability improvement. *IEEE Transactions on Industry Applications, 55*(6), 6486-6498. https://doi.org/10.1109/TIA.2019.2938473

[5] Özyurt, F., Mira, A., & Çoban, A. (2022). Face mask detection using lightweight deep learning architecture and raspberry Pi hardware: An approach to reduce risk of

coronavirus spread while entrance to indoor spaces. *Traitement du Signal*, *39*(2), 645-650. https://doi.org/10.18280/ts.390227

[6] Ali, H. I. & Ibrahim, I. H. (2022). An optimal quantitative PID controller design for ball and beam system. *Journal Européen des Systèmes Automatisés, 55*(3), 323-329. https://doi.org/10.18280/jesa.550304

[7] Wei, B., Xia, X., Yu, F., Zhang, Y., Xu, X., Wu, H., Gui, L., & He, G. (2020). Multiple adaptive strategies based particle swarm optimization algorithm. *Swarm and Evolutionary Computation, 57*, 100731. https://doi.org/10.1016/j.swevo.2020.100731

[8] Maghfiroh, H., Ma'arif, A., Adriyanto, F., Suwarno, I., & Caesarendra, W. (2023). Adaptive linear quadratic gaussian speed control of induction motor using fuzzy logic. *Journal Européen des Systèmes Automatisés, 56*(4), 703-711. https://doi.org/10.18280/jesa.560420

[9] Lin-Shi, X. F., Morel, F., Llor, A. M., Allard, B., & Retif, J. M. (2007). Implementation of hybrid control for motor drives. *IEEE Trans. Ind. Electron, 54*(4), 1946-1952. https://doi.org/10.1109/TIE.2007.898303

[10] Chan, Y. F., Moallem, M., & Wang, W. (2007). Design and implementation of modular FPGA-based PID controllers. *IEEE transactions on Industrial Electronics, 54*(4), 1898-1906. https://doi.org/10.1109/TIE.2007.898283

[11] Kadhim, N. N., Abood, L. H., & Mohammed, Y. A. (2023). Design an optimal fractional order PID controller for speed control of electric vehicle. *Journal Européen des Systèmes Automatisés, 56*(5), 735-741. https://doi.org/10.18280/jesa.560503

[12] Zhu, R. & Wu, H. (2017). Dc motor speed control system based on incremental PID algorithm. *Instrum. Tech. Sens, 7*, 121-126.

[13] Hong, H., Ni, L., & Sun, H. (2021). Design and simulation of a self-driving precision compass based on BP+ PID control. *Mech. Des, 38*, 78-84.

[14] Wang, H., Jiang, H., & Cheng, X. (2015). A design of flight control system for four-rotor micro aerial vehicle. *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 1447-1451. https://doi.org/10.1109/CICN.2015.279

[15] Liu, J., Liu, M., Pei, D., & Sun, H. (2019). FPGA implementation of family service robot based on neural network PID motion control system. *2019 International Conference on Electronic Engineering and Informatics (EEI)*, 304-308. https://doi.org/10.1109/EEI48997.2019.00073

[16] Wang, J., Li, M., Jiang, W., Huang, Y., & Lin, R. (2022). A design of FPGA-based neural network PID controller for motion control system. *Sensors, 22*(3), 889-917. https://doi.org/10.3390/s22030889

[17] Yin, M. K., Guo, S. J., Sun, L., & Zhang, J. J. (2021). Fuzzy adaptive PID control of the lower-limb walking-assistance exoskeleton robot. *Journal of Machine Design, 38*(9), 38-44.

[18] Rosa, N., Arantes, M. D. S., Toledo, C. F. M., & Lima, J. M. G. (2022). Implementation on FPGA of neuro-genetic PID controllers auto-tuning. *Intelligent Information Management, 14*(5), 165-193. https://doi.org/10.4236/iim.2022.145012

[19] Naouar, M. W., Monmasson, E., Naassani, A. A., Slama-Belkhodja, I., & Patin, N. (2007). FPGA-based current controllers for AC machine drives-A review. *IEEE Transactions on Industrial Electronics, 54*(4), 1907-1925. https://doi.org/10.1109/TIE.2007.898302

[20] Ghani, N. M. A., Othman, A., Hashim, A. A. A., & Nasir, A. N. K. (2023). Comparative Analysis of PID and Fuzzy Logic Controllers for Position Control in Double-Link Robotic Manipulators. *Journal of Intelligent Systems and Control, 2*(4), 183-196. https://doi.org/10.56578/jisc020401

[21] Jin, L. C., Hashim, A. A. A., Ahmad, S., & Ghani, N. M. A. (2022). System Identification and Control of Automatic Car Pedal Pressing System. *Journal of Intelligent Systems and Control, 1(*1), 79-89. https://doi.org/10.56578/jisc010108

Contact information:

Dechun ZHENG, PhD Student & Associate Professor
School of Electronic and Information Engineering,
Ningbo University of Technology,
Ningbo 315016, China
E-mail: zdc@nbut.edu.cn

Jiliang XU, Senior Engineer
(Corresponding author)
Ningbo Airport Group Co. LTD,
Ningbo 315016, China
E-mail: jl_xu@nbairport.com

Li XU, PhD Student & Associate Professor
School of Electronic and Information Engineering,
Ningbo University of Technology,
Ningbo 315016, China
E-mail: xuli@nbut.edu.cn