electronics

Article

Design and Implementation of an Efficient Hardware Coprocessor IP Core for Multi-axis Servo Control Based on Universal SoC

Jitong Xin [1,†], Meiyi Cha [1,†], Luojia Shi [1], Xiaoliang Jiang [2], Chunyu Long [1], Qichun Lin [2], Hairong Li [1], Fangcong Wang [1] and Peng Wang [1,*]

[1] Institute of Microelectronics, School of Physical Science and Technology, Lanzhou University, Lanzhou 730000, China
[2] ZheJiang Hikstor Technology Co., Ltd., Wuxi 214000, China
* Correspondence: wangpeng@lzu.edu.cn
† These authors contributed equally to this work.

Abstract: The multi-axis servo control system has been extensively used in industrial control. However, the applications of traditional MCU and DSP chips in high-performance multi-axis servo control systems are becoming increasingly difficult due to their lack of computing power. Although FPGA chips can meet the computing power requirements of high-performance multi-axis servo control systems, their versatility is insufficient, and the chip is too costly for large-scale use. Therefore, when designing the universal SoC, it is better to directly embed the coprocessor IP core dedicated to accelerating the multi-motor vector control current loop operation into the universal SoC. In this study, a coprocessor IP core that can be flexibly embedded in a universal SoC was designed. The IP core based on time division multiplexing (TDM) technology could accelerate the multi-motor vector control current loop operation according to the hardware–software coordination scheme proposed in this study. The IP was first integrated into a universal SoC to verify its performance, and then the FPGA prototype verification for the SoC was performed under three-axis servo control systems. Secondly, the ASIC implementation of the IP was also conducted based on the CSMC 90 nm process library. The experimental results revealed that the IP had a small area and low power consumption and was suitable for application in universal SoC. Therefore, the cheap and low-power single universal SoC with the coprocessor IP can be suitable for multi-axis servo control.

Keywords: multi-axis servo control; IP core; time division multiplexing (TDM); universal SoC; hardware–software coordination scheme; FPGA prototype verification; ASIC implementation

Citation: Xin, J.; Cha, M.; Shi, L.; Jiang, X.; Long, C.; Lin, Q.; Li, H.; Wang, F.; Wang, P. Design and Implementation of an Efficient Hardware Coprocessor IP Core for Multi-axis Servo Control Based on Universal SoC. *Electronics* **2023**, *12*, 452. https://doi.org/10.3390/electronics12020452

Academic Editor: Marco Vacca

Received: 15 December 2022
Revised: 11 January 2023
Accepted: 13 January 2023
Published: 15 January 2023

1. Introduction

With the rapid development of microelectronics and corresponding control technologies, there is a need to improve the characteristics of industrial AC motors [1]. Numerous studies on AC motors have shown that permanent magnet synchronous motors (PMSMs) are critical in the field of AC motors due to their unique advantages [2]. PMSMs are extensively used in various fields, including new energy vehicles, exoskeleton systems, missile steering engines, and UAVs [3–7].

At present, there is a high demand for MCU or DSP as the motor control chip. However, they are entirely dependent on software algorithms. Thus, because they have slow calculation and response speeds, they cannot be directly used in high-speed control systems. Multi-MCU and multi-DSP have been used where high control responses are required [8]. This will undoubtedly lead to higher power consumption and cost. FPGA is suitable for medium and high-end motor control applications due to its unique parallelism [9,10]. In recent years, certain vector control algorithms have been applied to FPGA to realize its acceleration [11–15]. Loop operation of vector control through FPGA has also been

applied [16–20], greatly improving the operation speed of the current loop. Besides, the PMSM vector control ASIC has been designed to improve the integration of the system [21]. These studies demonstrate the rapid development and progress of the PMSM control field.

With the flourishing development of industry and the gradual improvement of productivity, the application of PMSM is not limited to a single main control system of a single motor, but it extends to the real-time control of multiple motors simultaneously [22], such as mechanical phased array antenna, multi-joint manipulator, multi-axis linkage CNC machine tools, and other application scenarios. In these scenarios, some motor drive systems should control multiple motors in the system synchronously or asynchronously through a single main control chip [23], which requires the superior performance of the motor control system [24]. In recent years, many researchers have devoted themselves to improving the efficiency of multi-axis servo control systems. For example, Sarayut Amornwongpeeti et al. applied the multi-axis servo control on FPGA based on time division multiplexing [6,24], whereas Qiang Gu et al. used the multi-axis servo control on zynq SoC based on hardware and software collaboration [25,26]. These findings demonstrate the application of FPGA parallelism to achieve efficient control of multi-axis servo control systems. However, FPGA or zynq SoC are unsuitable for large-scale industrial applications because of on-chip resources, their cost and system power consumption.

Therefore, in the process of universal SoC chip design, it is undoubtedly a good choice to directly embed the coprocessor IP core that can efficiently accelerate the operation of multiple PMSM vector control algorithms into the universal SoC. The coprocessor IP core greatly improves the speed of calculation of multi-PMSM Vector Control through hardware pipeline calculation and parallel computing. This facilitated multi-axis servo control at a low-cost and low-power single universal SoC. The schematic diagram of a universal SoC embedded with a coprocessor as the main control chip of a multi-axis servo control system to realize multi-axis servo control is shown in Figure 1.

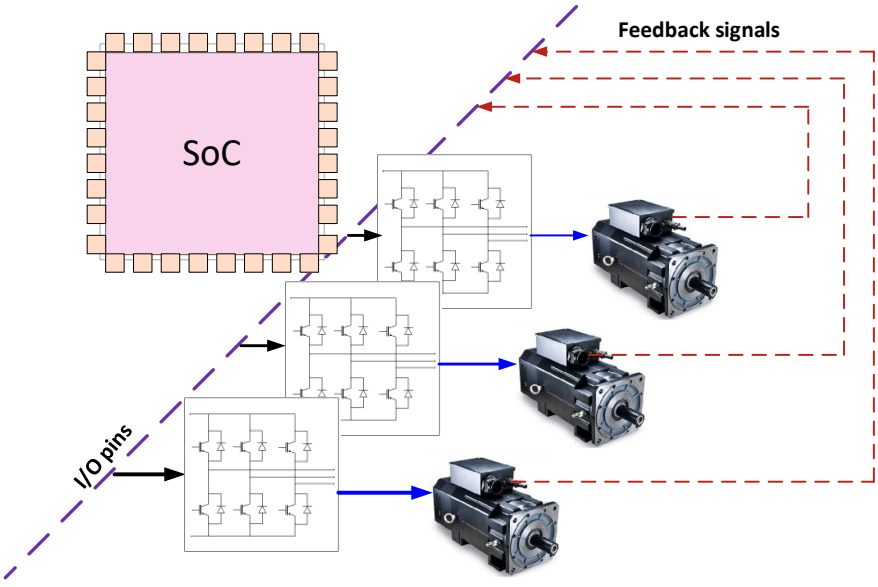


Figure 1. Multi-axis servo control system with universal SoC embedded in coprocessor as main control chip.

2. Background Research

2.1. Mathematical Model of Permanent Magnet Synchronous Motor

The voltage equation of the surface-mounted permanent magnet synchronous motor in the d-q axis coordinate system is expressed by Formulas (1) and (2).

$$u_d = R_s i_d + L_d \frac{di_d}{dt} - p\omega_e L_q i_q \quad (1)$$

$$u_q = R_s i_q + L_q \frac{di_q}{dt} + p\omega_e L_d i_d + p\omega_e \psi_f \tag{2}$$

where u_d and u_q are the voltage on d and q axes, respectively; i_d and i_q are the current on d and q axes, respectively; L_d and L_q are the inductance d and q axes, respectively; R_s is the stator resistance; ψ_f is the permanent magnet flux linkage; ω_e is the rotor electric angular velocity; and p is the number of motor poles.

The torque equation in the d-q axis two-phase rotating coordinate system is shown in Equation (3):

$$T_e = \frac{3}{2} p \left[\psi_f i_q + i_d i_q (L_d - L_q) \right] \tag{3}$$

where T_e is the output torque.

When ignoring friction, the motion equation in the d-q axis two-phase rotating coordinate system can be expressed by Equation (4):

$$J \frac{\mathrm{d}\omega_r}{\mathrm{d}_t} = -B\omega_r + \frac{3}{2} p \psi_f i_q + \frac{3}{2} p i_d i_q (L_d - L_q) - T_L \tag{4}$$

where T_L is the load torque, B is the friction coefficient, J is the rotational inertia of the axis, and ω_r is the mechanical angular velocity.

2.2. Vector Control

Vector control is also known as field-oriented control. Its control of PMSM adopts a rotor flux-oriented approach, and it is suitable for servo control and other small drive control occasions. The key to vector control is coordinate transformation, so there are numerous operations in the current loop of vector control. The current loop comprises Clarke transform, Park transform, Ipark transform, PI control, and SVPWM modulation [25]. At present, the loop control strategy of PMSM mainly includes $i_d = 0$ control method, $cos\varphi = 1$ control method, torque current maximum ratio control method, and flux weakening control method. For the surface-mounted permanent magnet synchronous motor, the current loop control strategy with $i_d = 0$ is a better choice to achieve the purpose that all the currents output by the servo system generate torque components and improve the torque output efficiency. On this basis, the speed loop and position loop are added to realize the speed and position servo control of the motor [26]. The control block diagram is shown in Figure 2.

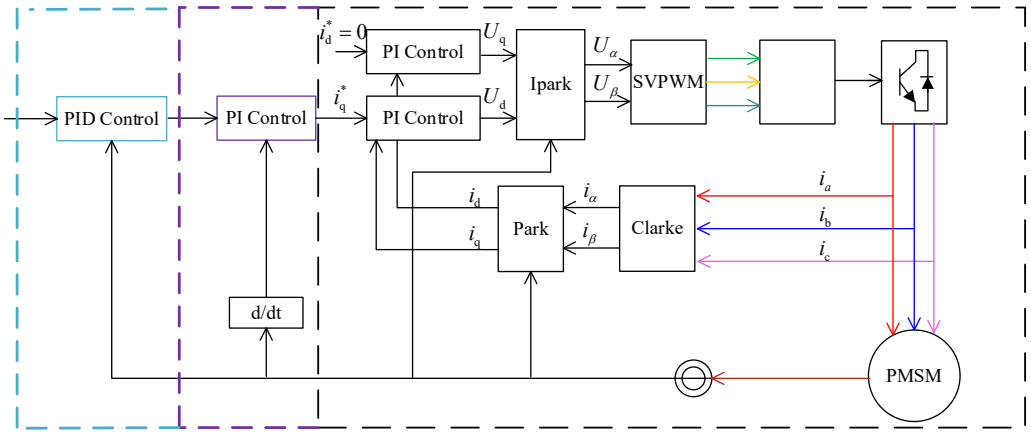


Figure 2. PMSM position servo control system.

2.3. Basic Architecture of Universal SoC

SoC can be divided into two types according to their purpose: one is dedicated SoC chips, which are the development of ASIC to system-level integration, and the other is universal SoC chips [27]. MCU, widely used in industrial control, also belongs to universal SoC [28].

The universal SoC's architecture is hierarchical [29]. The levels of the SoC system include the core: such as ARM, and RISC-V; bus interconnection: AMBA3, AMBA4, etc.; IP conforming to high-speed bus interface (high-speed IP): DMA, FLASH, SRAM, etc.; bridging IP: conversion bridge between AHB and APB; and peripheral equipment IP (low-speed IP) that meets the low-speed bus interface: UART, SPI, PWM, ADC, DAC, TIMER, I2C, WDT, GPIO, etc. Figure 3 shows the basic architecture of the universal SoC. In general, the main devices, SoC, CPU, and DMA, are connected to multiple slave devices through the AHB bus interconnection matrix. The AHB bus interconnection matrix enables different master devices to access different slave devices simultaneously, realizing parallel transmission of data of multiple master devices [30,31].

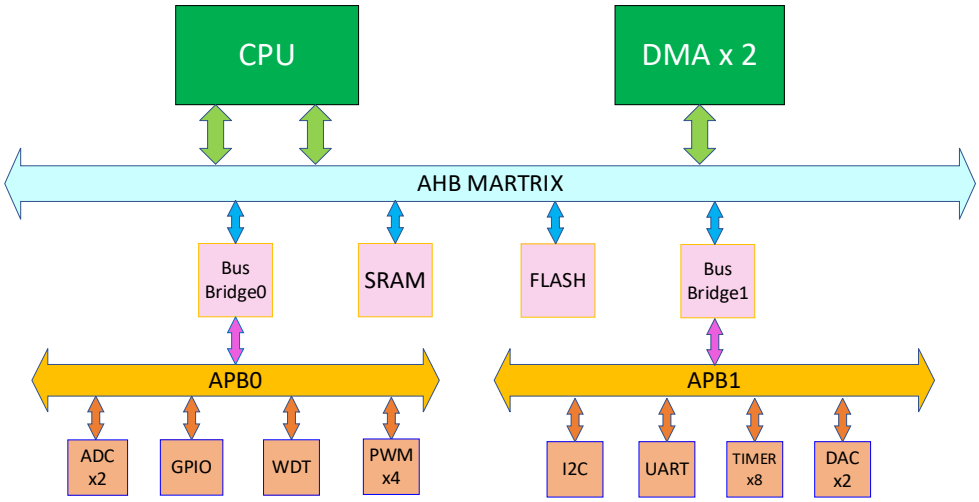


Figure 3. The architecture of universal SoC.

3. Implementation Process

Section 3.1 of this chapter focuses on the overall design architecture of the coprocessor IP and the key methods to improve the overall performance of the IP in the design process. Section 3.2 describes the SoC integration method. Section 3.3 describes the hardware–software coordination scheme of the IP.

3.1. Coprocessor IP Core Design

Figure 2 shows that the speed loop and position loop of PMSM vector control add PI operation and PID operation based on the current loop. It can be seen that the calculation of PMSM vector control is mainly focused on the current loop. Therefore, the IP core of the coprocessor involved in this study is only responsible for accelerating the current loop operation, while PI and PID operations of speed and position loops are performed entirely by the software. This not only saves the area and power consumption of the coprocessor IP core, but also increases the flexibility of PMSM vector control. Limited by the number of registers, this IP supports a six-axis servo control system at most.

The overall architecture of the IP is shown in Figure 4. The external interfaces of the IP include the AHB SLAVE interface in the AMBA bus, interrupt interface, DMA request interface, and response interface. The IP is mainly composed of an interface module and an operation core. The former is used to control the read and write of registers and the control of DMA and interrupt-related signals, and the latter is used to accelerate the current loop.

As shown in Figure 4, the interface module consists of an AHB slave controller, input register heap, output register heap, DMA, and interrupt controller.

The AHB slave controller reads and writes registers according to AHB protocol. AHB bus signals include HADDR, HWDATA, HRDATA, HSEL, etc. The read and write operations of the AHB protocol in the basic transmission mode are shown in Figure 5a,b [30].

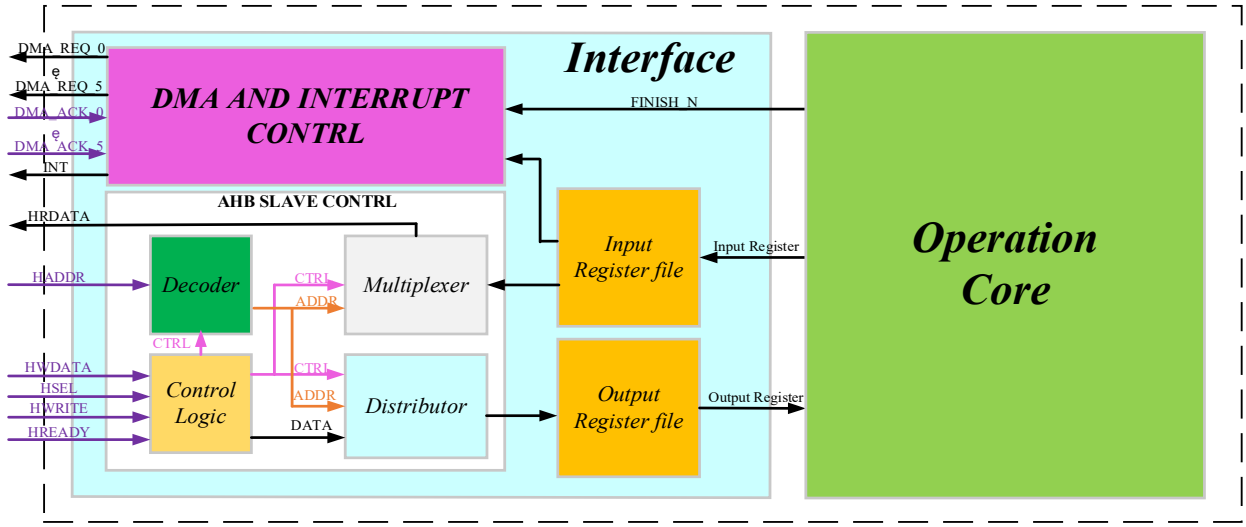


Figure 4. Overall architecture of coprocessor IP.

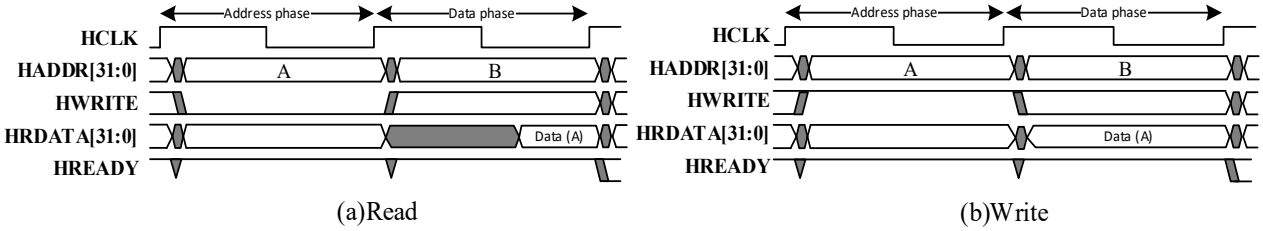


Figure 5. The read and write operations of the AHB protocol in the basic transmission mode.

The Input Register file registers the data output by the operation core, and the Output Register file registers the data configured by the host through the bus. As there are numerous input, output, and calculation parameters of multi-motor vector control current loop operation, the number of registers in the IP is also large. The registers of the IP are divided into five categories. The first category is the input variable register, which configures the input phase current value, the electric angle value, and the reference current value of the IP. These values constantly change during the PMSM current loop control process. Therefore, they need to be configured before each current loop calculation begins. The second category is the input control register, which is used to configure the motor number value (used to determine which motor in the multi-axis servo system is used in this calculation), calculate the mode value (whether to turn on the SVPWM overmodulation function), and calculate the start signal value (the signal is a pulse signal, and the IP starts an operation when it is high-level). These registers need to be configured before each current loop calculation begins; the third category is parameter registers, which include the PI control parameters and SVPWM carrier cycle parameters of all motors in the multi-axis servo system. They need to be configured during the device initialization phase; the fourth category is interruption and DMA control registers, which can not only turn on the DMA function or interruption function of the IP, but also clear the interrupt. They need to be configured in the device initialization phase. The fifth category is output class registers, which include all the calculation result values output by the operation core and the output motor number value corresponding to the calculation result value (used to determine which motor in the multi-axis servo system the calculation result is).

DMA and interrupt controller controls the DMA request signal, DMA response signal, and the IP interrupts signal. The DMA request signal and DMA response signal are related to DMA handling the calculation result data. Because the interrupt processing time is too long, the interrupt function is generally only turned on during the debugging process.

As shown in Figure 6, the operation core module consists of a Clarke unit, CORDIC unit, Park unit, Ipark unit, d-axis PI control unit, q-axis PI control unit, SVPWM unit, and data control unit. Due to the trigonometric function in the vector control, the CORDIC unit is embedded in the IP to generate the CORDIC algorithm. Compared to the look-up table method, the CORDIC unit saves the area of the circuit and improves the calculation accuracy of the trigonometric function. The data control unit transmits the intermediate calculation result data of the operation process according to the operation sequence of the IP. Therefore, a full pipeline design architecture is adopted in the overall hardware design of the operation core module to realize the high-speed operation of the multi-motor vector control current loop.

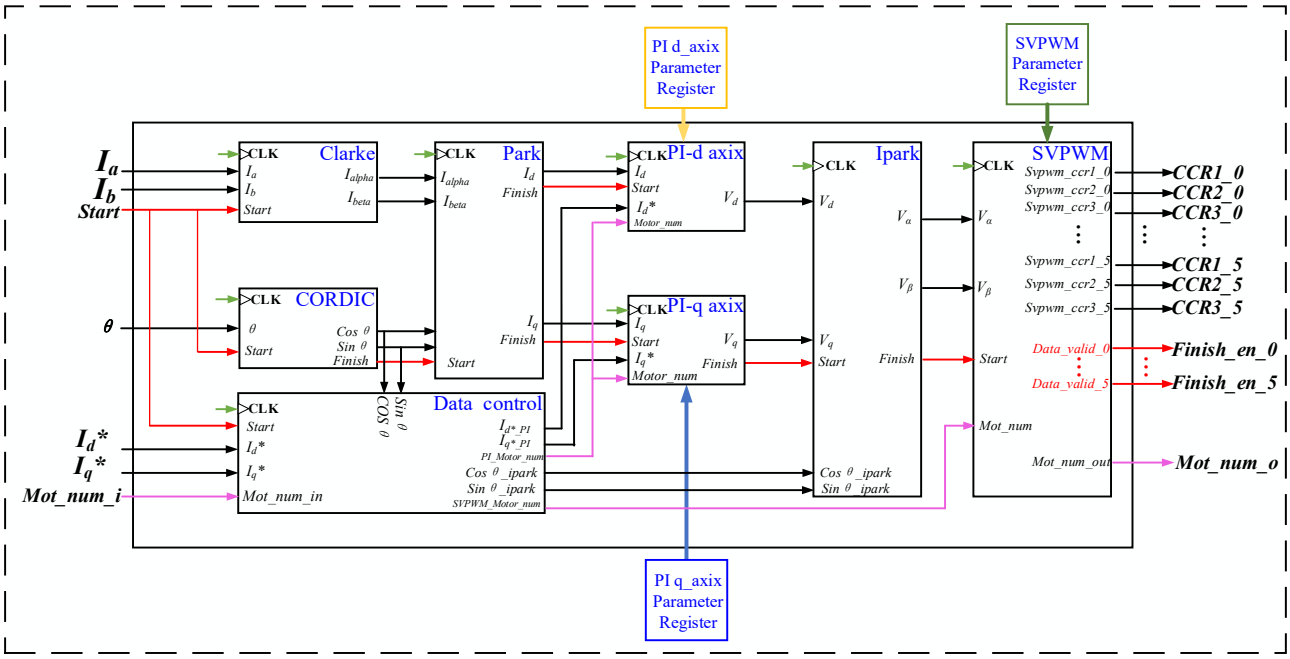


Figure 6. Overall diagram of operation core module.

Many factors must be considered when designing an IP core, such as accuracy, speed, simplicity, and flexibility. Four key methods were proposed in this paper for designing the operation core module, which simplifies the overall calculation, optimizes the hardware area, improves the speed response, and improves the DC bus voltage utilization in the vector control process.

3.1.1. Method 1: Data Normalization Processing Method

An IP data unified 16-bit fixed-point mode was adopted to facilitate the coprocessor IP for data processing. When using the fixed-point mode, it is necessary to reduce the magnitude of input data changes, so the integer data needs to be converted into decimals with higher accuracy by data normalization. In practical applications, the normalized system can adapt to various applications. Therefore, in this paper, the universality of the coprocessor IP core was improved by normalization.

For the input current value, the digital quantity output after ADC sampling was already normalized data, so no additional normalization processing was needed. However, the voltage in the vector control operation process needed normalization. Before normalization, it was necessary to select the appropriate base value. Selecting a suitable base value can not only achieve data normalization, but also reduce the calculation of vector control, thus saving circuit area and power consumption. According to the formula characteristics of SVPWM modulation calculation, this study uses Equation (5) to obtain the voltage base

value, where U_{base} is the standardized value, U_{dc} is the bus voltage value of the motor, and the size of U_{dc} is related to the motor itself.

$$u_{base} = \frac{u_{dc}}{\sqrt{3}} \tag{5}$$

To simplify the calculation, the normalization of voltage value in PI control was calculated. The PI control formula is expressed using Equation (6).

$$u(k) = k_p e(k) + k_i \sum_{n=0}^{k} e(n) \tag{6}$$

where $u(k)$ is the control quantity, that is, the voltage. $e(k)$ is the input error, which is the difference between the reference and the true current. k_p is the proportional error, and k_i is the integral gain. For the current, the functional relationship between the digital quantity $I_t(k)$ in output by the ADC after sampling and the true current value $I(k)$ was expressed as shown in Equation (7).

$$I(k) = KI_t(k) \tag{7}$$

where K is the ADC sampling amplification coefficient. The digital quantity $I_r(k)$ of the reference current and the real value $I(k)$* of the reference current is expressed using Equation (8).

$$I(k)^* = KI_r(\mathrm{k}) \tag{8}$$

As shown in Equation (9), the normalized value of the voltage could be obtained by dividing the output $U_n(k)$ in the PI expression by the reference voltage. As shown in Equation (10), the difference between Equations (7) and (8) is the input error value of PI.

$$U_n(k) = \frac{u(k)}{u_{base}} \tag{9}$$

$$E(k) = I(k)^* - I(k) \tag{10}$$

By combining Equations (6), (9) and (10), the normalization of voltage could be realized by PI control through simple transformation, as shown in Equation (11), where K_p/Ku_{base} and K_i/Ku_{base} can be used as input parameters K_p and K_i in PI control. As K_p and K_i are PI parameters, they are configured through configuration registers in the device initialization phase.

$$U_{normal}(k) = \frac{K_p}{Ku_{base}} E(k) + \frac{K_i}{Ku_{base}} \sum_{n=0}^{k} E(n) \tag{11}$$

The IP data can be normalized by mathematical transformation of the vector control formula, which limits the data to $(-1, 1)$. However, considering the SVPWM overmodulation in vector control, all IP values should be limited to $(-2, 2)$. As the IP data format is uniformly selected as 16-bit fixed-point data, the Q14 format was used to represent all the data in the IP. Therefore, 1 represents *16'h4000*, the highest bit is the symbol bit, the second high bit is the integer bit, and the remaining bits are decimals.

3.1.2. Method 2: Design Method for the Multiplication Calculation Circuit

Given that there are many multiplication calculations in the current loop operation, it is necessary to design this part of the operation circuit to improve the operational performance.

Because the PMSM vector control algorithm has a large number of multiplication calculations, all multiplication calculation outputs need to be registered inside the coprocessor IP, which is conducive to reducing the length of the combined logic chain and improving the dominant frequency of the coprocessor IP core.

Moreover, the Clarke and Ipark transformations involve the multiplication calculation of irrational constants. Taking Clarke transformation as an example, the specific Clarke transformation formulae are shown in Equations (12) and (13).

$$i_\alpha = i_a \tag{12}$$

$$i_\beta = \frac{1}{\sqrt{3}} \times (i_a + 2i_b) \tag{13}$$

There are two main methods for hardware design of multiplication of irrational number constants, the first is to directly convert the irrational constant into the fixed point number of the corresponding Q format as the multiplicand for multiplication calculation, and the other is to use the approximate value of the irrational constant as the multiplicand and split the approximate value into multiple 2^{-n} addition calculations. As shown in Equations (14) and (15), the approximate value of the irrational number constant in the Clarke transformation is decomposed into 2^{-n} continuous addition, and multiplied by the multiplier. In hardware design, the multiplication of 2^{-n} is equivalent to the right shift of the data by n bits. This is equivalent to completing Clarke transformation only through adder and register shift. The block diagram of the shift phase addition calculation circuit is shown in Figure 7. Because the error of the approximate value was 0.035%, the calculation error of the second method was equivalent to the error generated by Q14, and the calculation error could be ignored.

$$\frac{1}{\sqrt{3}} \approx 0.57715 = \frac{1}{2} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256} + \frac{1}{512} + \frac{1}{1024} \tag{14}$$

$$i_\beta \approx \left(\frac{1}{2} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256} + \frac{1}{512} + \frac{1}{1024}\right) \times (i_b + 2i_b) \tag{15}$$

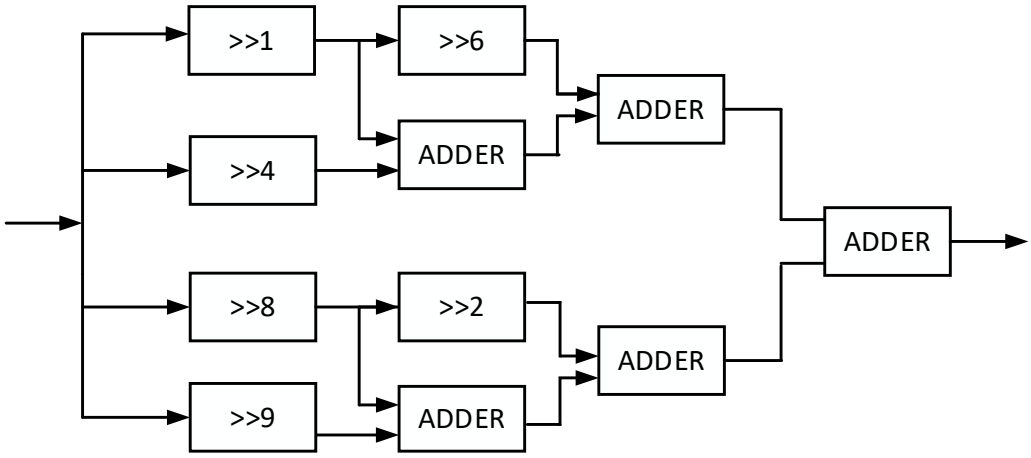


Figure 7. Circuit structure of the second method.

To respectively synthesize the above two methods, the first method required 1843 gates, and the second required 1266 gates. The resource consumption of the second method was 68.7% of the first resource consumption. In this IP, the irrational multiplication calculation in Ipark was also improved by the same method.

3.1.3. Method 3: Design Method of PI Control Unit

As the core of the coprocessor IP, the PI control unit realized the current loop PI control. Meanwhile, it used incremental PI control, in which the role of the proportional link P was to speed up the system response, and the role of the integral link I was to eliminate the

error and improve the system error. The expressions of incremental PI control are shown in Equations (16) and (17):

$$\Delta U_k = K_p(e_k - e_{k-1}) + K_i e_k \tag{16}$$

$$U_k = U_{k-1} + \Delta U_k \tag{17}$$

where e_k is the calculated deviation value and U_k is the calculated output value of PI. As shown in Figure 8, the PI control unit consists of a multi-channel control circuit, an input and output data distribution circuit, and a PI calculation circuit. The multi-channel register control circuit was used to register the PI parameter register values of different motors in the multi-axis servo system and the results of the previous PI calculation. At the beginning of each analysis, the PI control unit input the PI parameter register value of the corresponding motor and the calculation result data obtained by the corresponding motor in the previous PI calculation into the PI calculation circuit through the data distribution circuit according to the value of the input motor number register, making the PI calculation circuit start the PI calculation. When the calculation was completed, the PI data would be stored in the corresponding channel of the multi-channel control circuit through the data distribution circuit according to the value of the motor number.

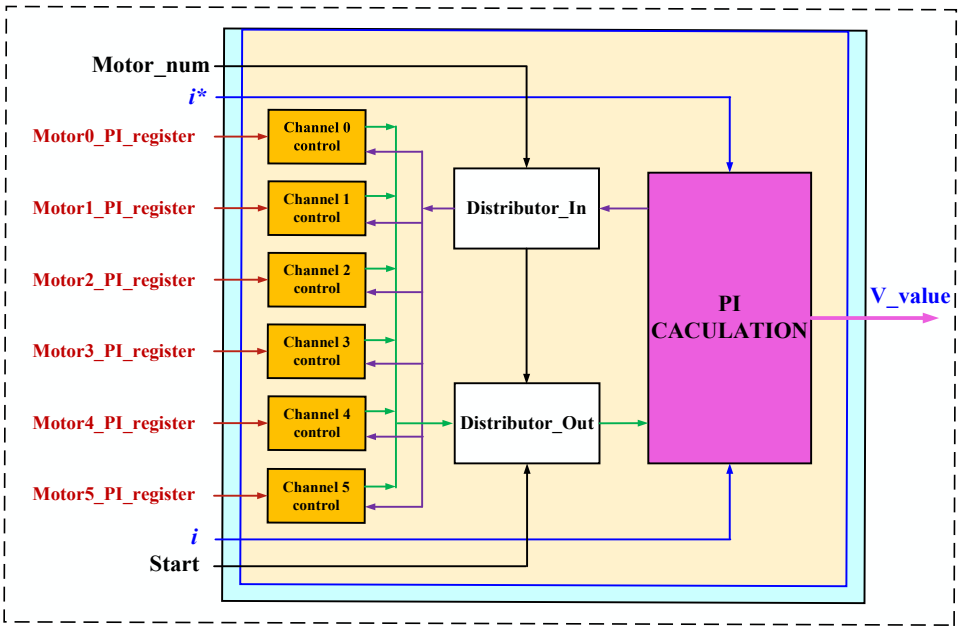


Figure 8. Structure of PI control unit.

The implementation flow of PI computing circuit is shown in Figure 9. The PI calculation circuit completed the incremental digital PI control and made the difference between the true value of the input and the expected value, that is, e_k. In this paper, multiple thresholds were set inside the PI calculation circuit to improve the response speed. In the device initialization stage, the host shaped all thresholds in the PI calculation circuit by configuring registers. The threshold e_{min} was used to determine whether the true value was close enough to the expected value. If it was close enough, PI control would not be needed. Therefore, when $|e_k| < e_{min}$, the output $U_k = U_{k-1}$. At this time, PI regulation has been stable, and there was no need for PI regulation. When the difference between the real input value and the expected value changed greatly, the integral link $K_i e_k$ in PI control would also be too large, which would cause a serious overshoot. At this point, it could be adjusted by the threshold δ. When $|e_k| > \delta$, the controller contained only the proportional link $K_p e_k$. If the difference between the real and the expected value was small, that is, $|e_k| \leq \delta$, then the controller adopted PI control, and the error was eliminated by introducing the integral link $K_i e_k$ to ensure that the system accuracy met the requirements. Therefore, the configuration threshold δ allowed the system to respond rapidly while preventing overshoot caused by

too large system response. The PI controller also had a threshold value, that is, the output limiting value U_{max}. When $|U_k| > U_{max}$, $U_k = sign(e_k) \times U_{max}$, which was equivalent to the proportional regulator with a sufficiently large proportional coefficient. At this time, the PI system would eliminate the deviation of the motor speed at the fastest speed.

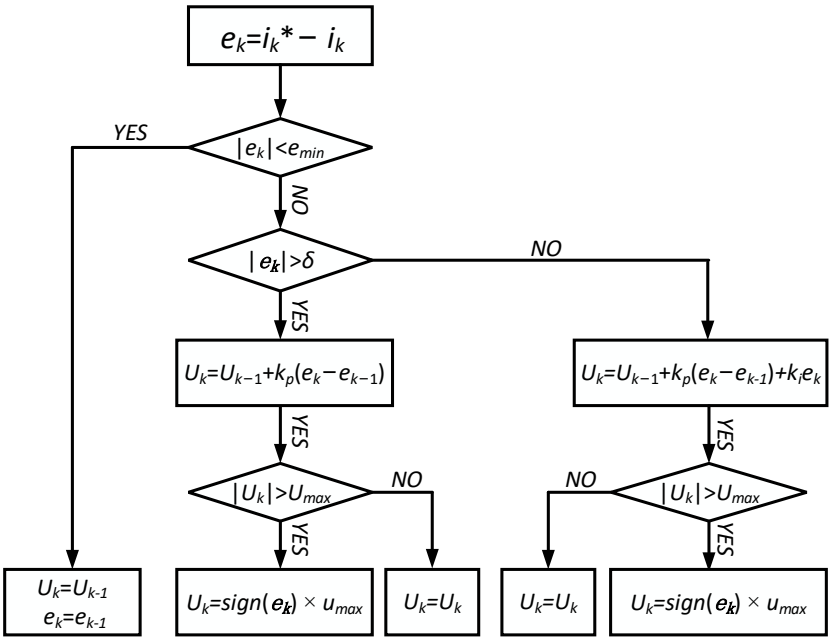


Figure 9. Flow chart of incremental PI control.

3.1.4. Method 4: SVPWM Overmodulation Design Method

The fundamental amplitude of the output voltage vector of the voltage source inverter controlled by SVPWM in the linear modulation region could be maximized as $U_{dc}/\sqrt{3}$, while the overmodulation could be maximized as $2U_{dc}/\pi$, and the output voltage was 10% higher than that in the linear modulation region, which was significant in improving the load capacity of the motor and accelerating the dynamic response speed [32,33]. Therefore, an overmodulation unit was specifically designed inside the SVPWM module in the design of the IP coprocessor to realize overmodulation operation, and this operation unit could open or close the unit by configuring the second category of registers in the device initialization stage.

The schematic diagram of SVPWM overmodulation is shown in Figure 10. Taking the first sector in SVPWM modulation as an example, it was assumed that the DC bus voltage was U_{dc}, the reference vector was V_{ref}, the switching period was T_s, the action time of vector V_4 was T_4, the action time of vector V_6 was T_6, and the action time of the zero vector was T_0. OA is the maximum amplitude of the reference voltage vector in the linear modulation region, which can be as high as $U_{dc}/\sqrt{3}$; OB and OE are the basic voltage vectors, and they have equal amplitude: $2U_{dc}/3$. Figure 11 divided the modulation range of SVPWM into three intervals, where OB is the maximum amplitude of the reference voltage vector in the over-modulation *region I*; and OC is the maximum magnitude of the reference voltage vector in the overmodulation *region II*: $2U_{dc}/\sqrt{3}$.

In the linear modulation region, the reference voltage vector amplitude range was $0 \le V_{ref} \le U_{dc}/\sqrt{3}$. At this time, the vector action time was $T_4 + T_6 \le T_s$, and there was no change in the actual vector action time.

In the overmodulation *region I*, the range of the reference voltage vector amplitude was $U_{dc}/\sqrt{3} < V_{ref} \le 2U_{dc}/3$, and the trajectory of the reference voltage vector moved in the annular region formed by OAF and OBE. At this point, the sum of the action time of adjacent

voltage vectors was greater than the switching period, that is, $T_4 + T_6 > T_s$. The action time of the actual voltage vector reduced proportionally, as expressed in Equations (18) and (19).

$$T_4' = \frac{T_4}{T_4 + T_6} T_s \tag{18}$$

$$T_6' = \frac{T_6}{T_4 + T_6} T_s \tag{19}$$

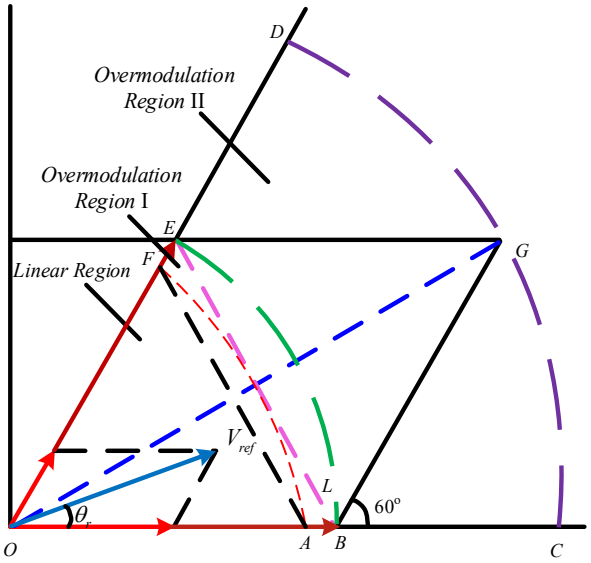


Figure 10. Schematic diagram of the first sector of SVPWM modulation.

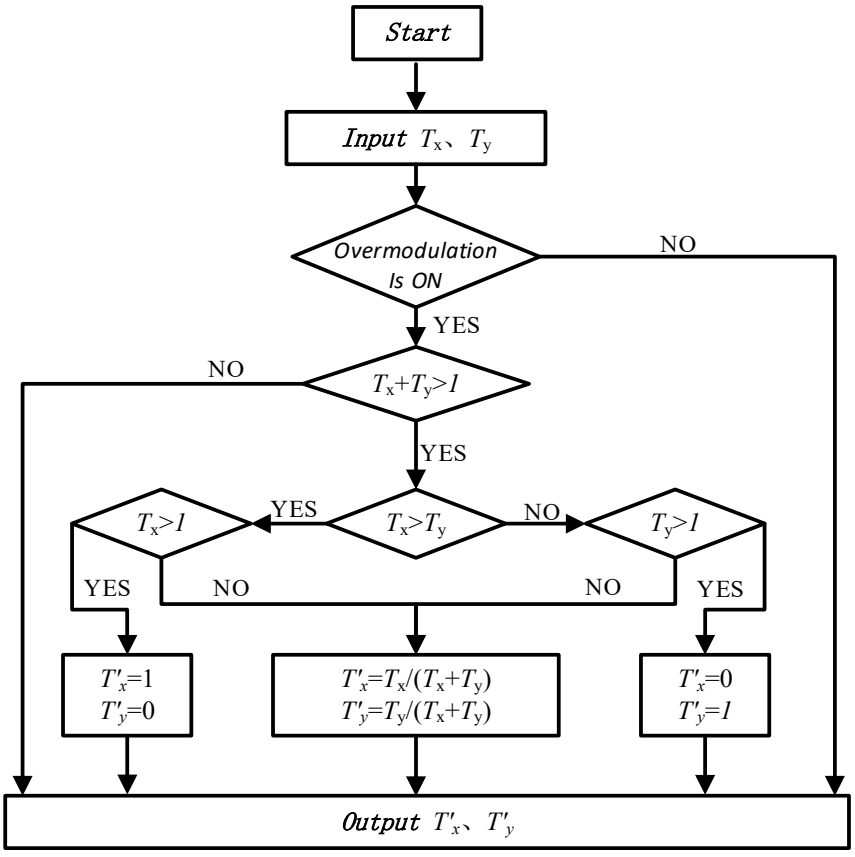


Figure 11. Flow chart of overmodulation algorithm.

In the overmodulation *region II*, the amplitude range of the reference voltage vector was $2U_{dc}/3 < V_{ref} \leq 2U_{dc}/\sqrt{3}$. At this point, the trajectory of the reference voltage vector rotated in the annular region composed of OBE and OCD. When the reference voltage vector trajectory was in the CBG region, the actual voltage vector was consistent with the basic voltage vector OB (V_4). When the reference voltage vector trajectory was in the DEG region, the actual voltage vector was consistent with the base voltage vector OE (V_6). When the trajectory of the reference voltage vector was located in the region composed of arc BE and straight lines BG and EG, the action time of the actual voltage vector would be reduced proportionally, and the reduction method was the same as the calculation method of Equations (18) and (19). When the amplitude of the reference voltage vector reached $2U_{dc}/\sqrt{3}$, the trajectory of the actual output voltage vector jumped between the six vertices of the regular hexagon, i.e., from OB to OE, with a dwell time of 1/6 period at each vertex. At this time, the motor ran in a six-step wave state [33].

There was a straightforward approach to establish which treatment the reference voltage vector corresponded to when its range was in the OCG of the overmodulation *region II*. If the endpoint of the reference voltage vector was on the straight line BG, the action time of two adjacent basic voltage vectors V_4 and V_6 was $T_4 = T_s$ and $T_6 = 0$, respectively, i.e., $T_4 + T_6 = T_s$. Therefore, the position of the reference voltage vector could be determined by the relationship between the action time of two adjacent basic voltage vectors and the switching period. When $T_4 + T_6 > T_s$ and $T_4 < T_s$, the trajectory of the reference voltage vector was inside the isosceles triangle OBG; and when $T_4 + T_6 > T_s$ and $T_4 > T_s$, the trajectory of the reference voltage vector was outside the isosceles triangle OBG, i.e., within the region CBG; this judgment method was simple, effective and easy to implement. Similarly, the processing was similar when the range of the reference voltage vector was in the ODG of the overmodulation *region II*. Therefore, this judgment method could combine the algorithm of overmodulation *region I* and overmodulation *region II* to handle the overmodulation of each sector of SVPWM. The flow chart of overmodulation processing in any sector is shown in Figure 11, and the input values T_x and T_y are the action time of two adjacent basic voltage vectors. As this IP was normalized in data processing, 1 was the switching period.

Here, a multi-cycle pipelined hardware divider was embedded in the SVPWM module to accelerate the overmodulation algorithm on the hardware; the hardware divider completes two scaling division operations in the over-modulation algorithm by a time division multiplexing method. However, in the case of multi-motor vector control, if a motor in an over-modulated state satisfied the conditions $T_x + T_y > 1$ and $T_x < 1$ & $T_y < 1$, it needed to consume multiple cycles to perform proportional scaling before subsequent calculation. On the other hand, the motor that satisfied other conditions would directly assign the results based on the algorithm input data, making the total calculation time uncertain. Therefore, it was necessary to configure the divisor and divisor data of the hardware divider based on the value of algorithm input data before performing the overmodulation algorithm calculation for the total calculation time to remain constant. Table 1 shows the hardware divider input data configuration.

Table 1. Overmodulation input data allocation table.

	$T_x + T_y > 1$			$T_x + T_y \leq 1$
	$T_x \geq T_y$ & $T_x \geq 1$	$T_y \geq T_x$ & $T_y \geq 1$	$T_x < 1$ & $T_y < 1$	
Dividend a1	a1 = 1	a1 = 0	a1 = T_x	a1 = T_x
Divisor b1	b1 = 1	b1 = T_x	b1 = $T_x + T_y$	b1 = 1
Dividend a2	a2 = 0	a2 = 1	a2 = T_y	a2 = T_y
Divisor b2	b2 = T_y	b2 = 1	b2 = $T_x + T_y$	b2 = 1

The SVPWM module was simulated by Vcs-2016 and Verdi-2017 tools to confirm the correctness of the overmodulation function. When the module turned on the overmodulation function, the input voltage gradually increased in the Testbench to make the SVPWM module reach the overmodulation state from linear modulation. As shown in Figure 12, the saddle wave output by SVPWM was gradually distorted into a six-step wave, and the simulation results were consistent with the overmodulation theory introduced above. The simulation also confirmed the correctness of the design.

Figure 12. Hardware implementation simulation of overmodulation algorithm.

3.2. SoC Integration Method of IP

This research takes the universal SoC architecture introduced in Section 2.3 as an example to introduce the coprocessor IP's SoC integration method. The core of SoC in this study is a three-stage pipelined 32-bit CPU with RISC-V instruction set architecture.

The interface of the coprocessor IP included an AHB interface signal, DMA interface signal, and interrupt interface signal, which needed to be connected with the SoC internal IP. Since the coprocessor IP was a high-speed IP, the AHB interface of the coprocessor IP was integrated into the AHB interconnection matrix for the coprocessor IP to serve as a slave device of the AHB bus matrix for CPU and two DMA access. Figure 13 shows the bus interconnection diagram of SoC. In addition to the AHB bus interface, we connected the DMA and interrupt interfaces of the coprocessor IP to the corresponding signals of the DMA and CPU in the universal SoC, respectively.

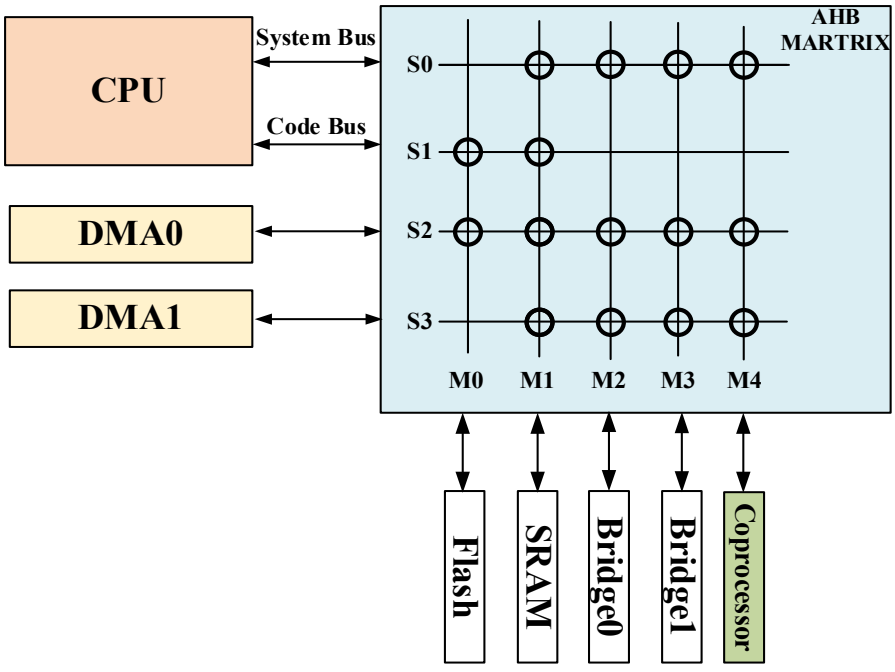


Figure 13. Bus interconnection.

3.3. Hardware–Software Coordination Scheme of IP

The SoC of the integrated coprocessor attained multi-axis servo control through hardware and software coordination. The speed loop and position loop in multi-axis servo control were completely calculated by the CPU through software due to the small amount of calculation, whereas the current loop with the largest amount of calculation was accelerated by the coprocessor IP through the hardware and software coordination scheme. The steps that took SoC to realize multi-motor vector control current loop operations through hardware and software coordination included: (a) the initialization phase, where the software initialized all devices in the SoC in turn. (b) After each current loop cycle started, the timer in the SoC would trigger the ADC to start sampling the phase current of each motor in the multi-axis servo system, whereas the CPU would calculate the electrical angle when sampling the current of the corresponding motor. (c) After the ADC sampling, the host would configure the first and second categories of registers of the coprocessor IP to the coprocessor IP based on the order of the motors to be calculated. Then, the coprocessor IP would start the vector control operation of each motor following the configuration order of the second category of registers. (d) As the vector control operation of each motor starts at different times and the hardware IP is parallel, the parallel execution of the vector operation of different motors can be realized inside the coprocessor, greatly accelerating the speed of the closed-loop operation of the current loop. After completing the SVPWM operation, the IP allocates the calculation result to the corresponding register according to the value of the motor number data and raises the corresponding completion signal. (e) After calculation, the DMA and interrupt controller would increase the DMA request signal corresponding to the motor number. After the DMA processed the DMA request, the calculation result was transported to the register of the PWM IP specified in the initialization phase. The DMA sent out a DMA response signal after the move, lowering the corresponding DMA request signal for the coprocessor core. (f) All PWM IPs simultaneously updated the duty cycle data at the beginning of the next PWM cycle and emitted PWM waves to complete a multi-motor vector control. The time sequence flow diagram of the IP when the coprocessor IP begins to perform operations is shown in Figure 14. Figure 15 shows the flow diagram of the hardware and software co-accelerated multi-motor vector control current loop operation of the coprocessor operation core.

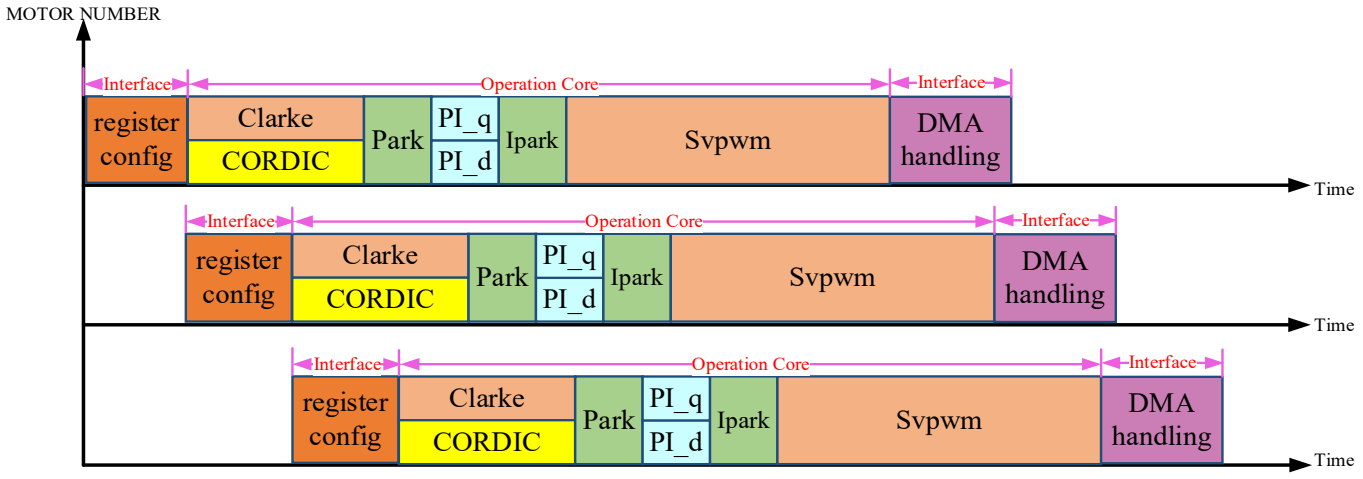


Figure 14. Coprocessor IP control timing sequence.

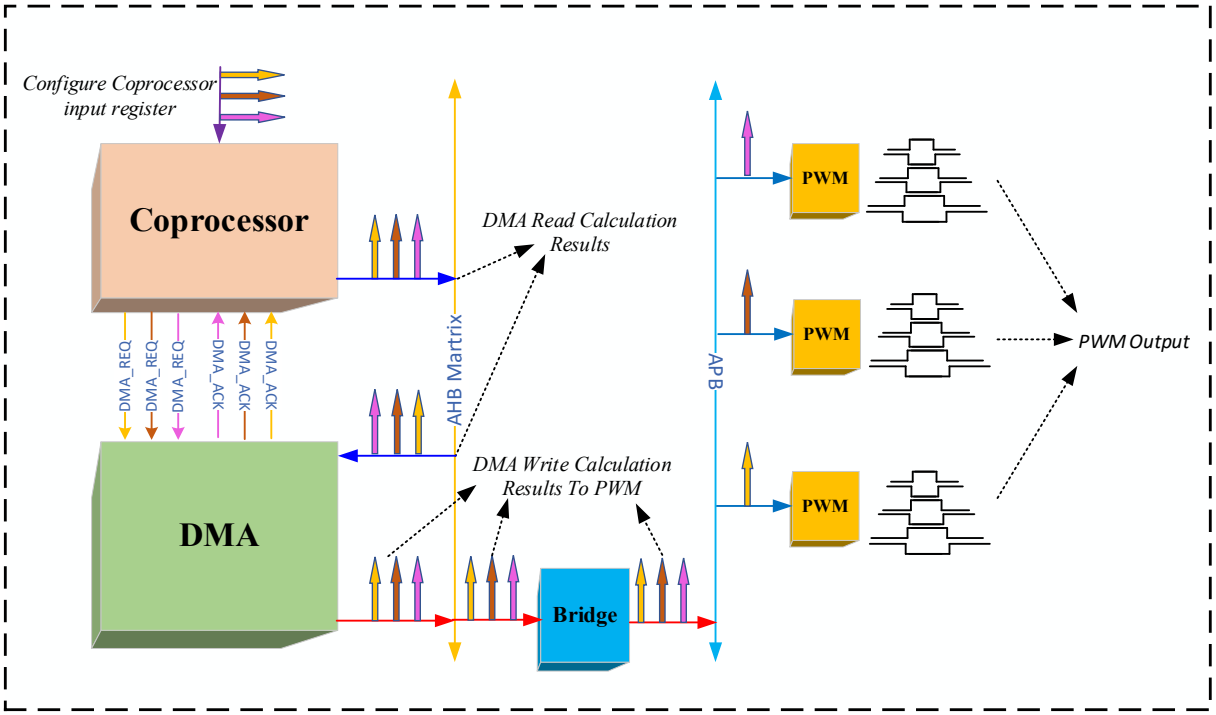


Figure 15. Hardware–software coordination scheme based on coprocessor IP.

4. Experiment and Experimental Results Analysis

This section will conduct experiments based on the IP designed in the previous section and the proposed hardware–software coordination scheme. The experiment includes FPGA prototype verification and ASIC implementation. Section 4.1 will mainly introduce the establishment and implementation of the experiment. Section 4.2 will show the resource consumption of the key IPs inside the SoC in the FPGA prototype verification system. Section 4.3 will test the function and performance of the universal SoC embedded in the coprocessor IP to realize the multi-axis servo system control. Section 4.3 also show and compare the calculation time necessary for multi-axis servo control current loop operation based on the FPGA prototype verification platform using pure software and hardware–software coordination. Section 4.4 will present the performance metrics of the IP ASIC implementation.

4.1. Establishment and Implementation of Experiments

The SoC hardware circuit embedded with the coprocessor core IP was completely described in Verilog HDL language, and the software driver code of the SoC was described in C language.

The FPGA prototype verification platform was established based on the Vivado2019.2 tool and the Digilent NEXYS Vidio Artix-7 FPGA development board, and the compilation and burning of the C program were realized by the Nuclei Studio IDE tool. Meanwhile, we constructed a three-axis servo control system based on the FPGA prototype verification platform to test the multi-axis servo control performance of the IP. The specific experimental system is shown in Figure 16. The system comprised a FPGA prototype verification platform, three PMSM motors with an encoder accuracy of 1000 lines and a rated speed of 3000 rpm, three motor drive boards, a 24 V DC power supply, and an oscilloscope. The PWM waveform output by SoC from the PAD and the phase current on the driver board was directly observed by the oscilloscope. The q-axis current value and the motor speed value were sent to the PC through the serial interface UART, and the PC drew the curve of the current data and the speed data through the Origin tool. The current loop frequency of the system was set at 20 KHZ, whereas the speed loop was set at 5 KHZ.

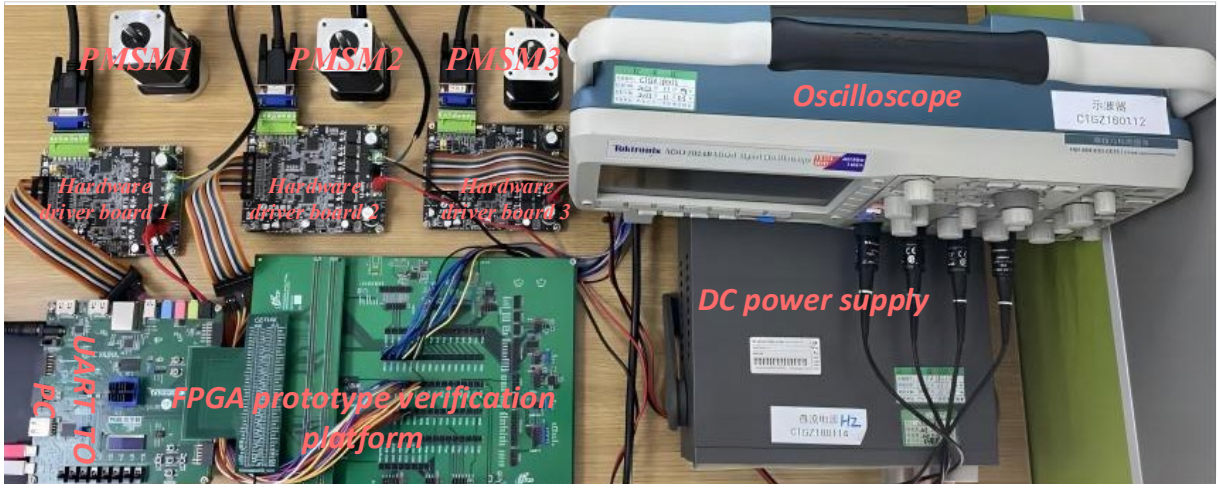


Figure 16. Diagram of the experimental platform.

The ASIC implementation was based on the CSMC 90 nm process library; Synopsys DC tool and Cadence Innovus tool were used to realize the synthesis and physical implementation of the IP.

4.2. Resource Consumption

Table 2 shows the resource consumption of the SoC coprocessor IP and the IP related to the servo control operation obtained by the Vivado tool for SoC synthesis. According to the data in the table, the SoC consumed 101,530 LUTs, 50,102 triggers, 33 RAMs, and 21 DSPs. The LUT consumed by the coprocessor IP accounted for 6.97% of SoC, the FLIP-FLOP for 13.17%, the RAMs for 0%, and the DSPs for 61.9%. Therefore, only a small amount of resources were essential to be embedded in the coprocessor IP we designed for the universal SoC chip to realize hardware acceleration of multi-axis servo control.

Table 2. SoC resource consumption.

	Resource Cost (Resource Percentage)			
	LUT	**FLIP-FLOP**	**RAM**	**DSP**
Coprocessor × 1	7074 (6.97%)	6600 (13.17%)	0 (0%)	13 (61.9%)
PWM × 4	7316 (7.21%)	4128 (8.24%)	0 (0%)	0 (0%)
Timer × 6	10,284 (10.13%)	5820 (11.62%)	0 (0%)	0 (0%)
ADC × 2	1524 (1.50%)	936 (1.87%)	0 (0%)	0 (0%)
CPU × 1	42,296 (41.66%)	13,674 (27.29%)	0 (0%)	8 (38.1%)
DMA × 2	12,530 (12.34%)	10,022 (20.00%)	0 (0%)	0 (0%)
AHB MATRIX × 1	2108 (2.08%)	269 (0.54%)	0 (0%)	0 (0%)
Bus bridge × 2	1888 (1.86%)	104 (0.21%)	0 (0%)	0 (0%)
GPIO × 6	3612 (3.56%)	4482 (8.95%)	0 (0%)	0 (0%)
Other IP	12,898 (12.70%)	4067 (8.12%)	33 (100%)	0 (0%)
SoC	101,530 (100%)	50,102 (100%)	33 (100%)	21 (100%)

4.3. Functional Verification and Performance Analysis

This section will show the function of single-axis servo control, the performance of multi-axis servo control, and the current loop operation time comparison of the software and hardware coordination scheme and software-only scheme.

4.3.1. Functional Experiment of Single-Axis Servo Control

Given a constant q axis reference current under a constant small load, the motor can run quickly and stably. Figure 17 shows the three forward PWM wave waveforms output by the PWM IP of the SoC through the GPIO. As shown, the duty cycle of the three

forward PWM waves was similar to that of the SVPWM output in different sectors in vector control. Figure 18 shows the A and B phase current waveforms of PMSM measured by two current probes of the oscilloscope, with a sinusoidal current and a phase difference of 120°. According to PMSM vector control theory, when the motor operates stably under a constant small load, the load torque is the size of electromagnetic torque at this time. Therefore, the output of electromagnetic torque is a constant value. That is, the three-phase current of A and B should be a sine wave with constant amplitude and 120° phase difference. Therefore, the test results agree with PMSM vector control theory.

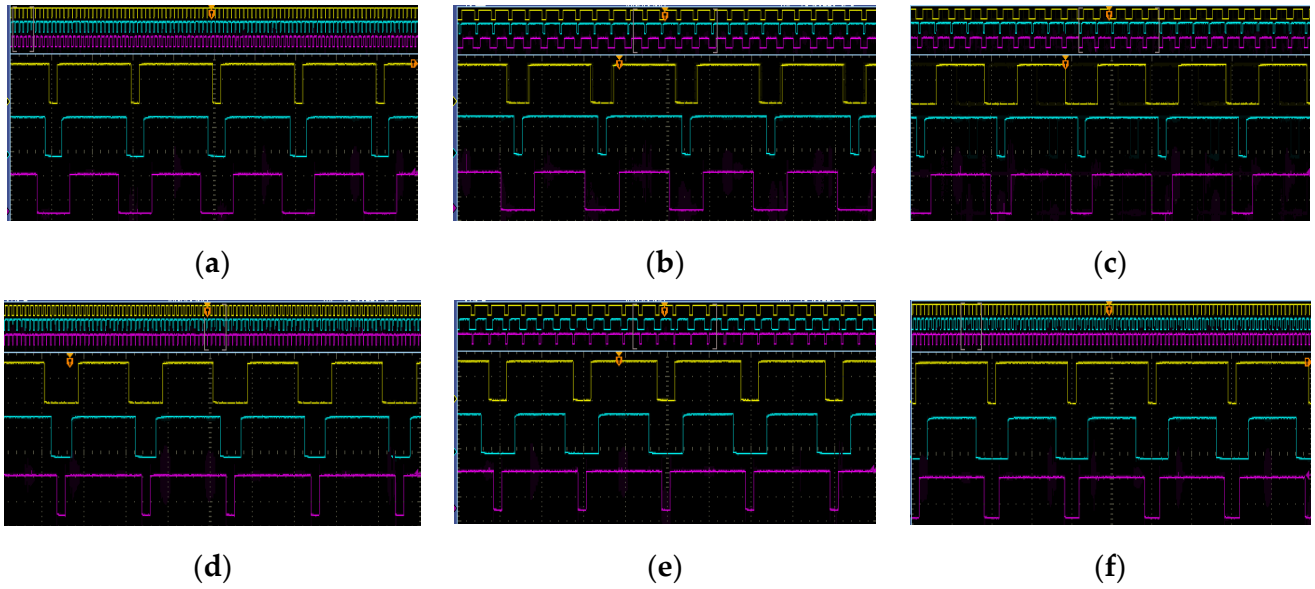


Figure 17. Schematic diagram of PWM waves in different sectors. (**a**) SVPWM first sector; (**b**) SVPWM second sector; (**c**) SVPWM third sector; (**d**) SVPWM fourth sector; (**e**) SVPWM fifth sector; and (**f**) SVPWM sixth sector.

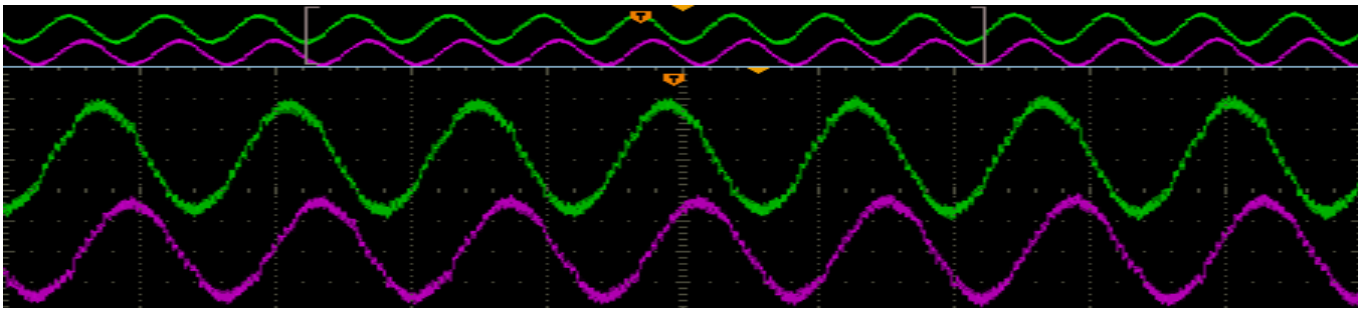
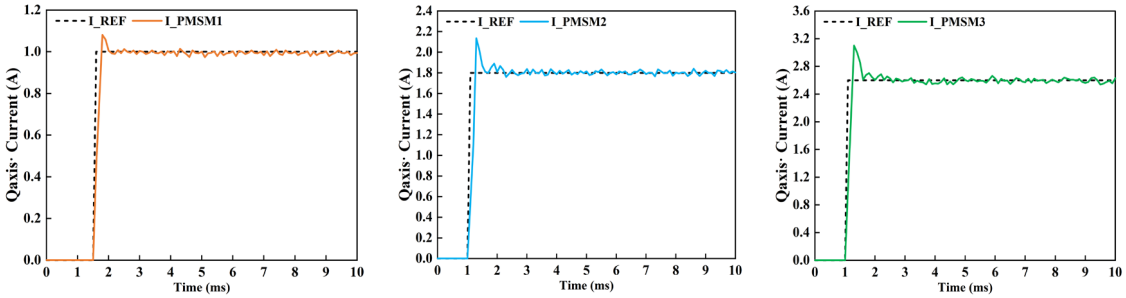


Figure 18. Schematic diagram of two-phase current of PMSM.

4.3.2. Multi-Axis Servo Control Performance Experiment

Figure 19 shows the expected and actual current curves of the three-axis servo system working in the current loop. We tested the current step effect of three PMSMs. Figure 19a shows the torque current tracking of the first-axis PMSM with a current amplitude of 1 A; Figure 19b shows the torque current tracking of the second-axis PMSM with a current amplitude of 1.8 A; and Figure 19c shows the torque current tracking of the third-axis PMSM with a current amplitude of 2.6 A. The actual current response under the step response of the current loop was fast and the overshoot was small when the PI coefficient was appropriately adjusted.



(**a**) (**b**) (**c**)

Figure 19. Schematic diagram of current response. (**a**) Current step response with step current 1 A; (**b**) current step response with step current 1.8 A; and (**c**) current step response with step current 2.6 A.

Figure 20 shows that when the three PMSM motors were running at different speeds, the speed step instruction first applied to PMSM1 was 3000 rpm, and the speed step instruction applied to PMSM1 was 600 rpm after the speed was stable for a certain period. When the PI coefficient adjustment of the speed loop was suitable, the actual speed response under the step response of the speed loop was fast and the overshoot did not appear.

The experimental results revealed that the universal SoC embedded with the coprocessor IP could independently and synchronously drive three motors, and the axes were controlled in parallel without interference. Meanwhile, the control precision was high, and the current and speed response was fast.

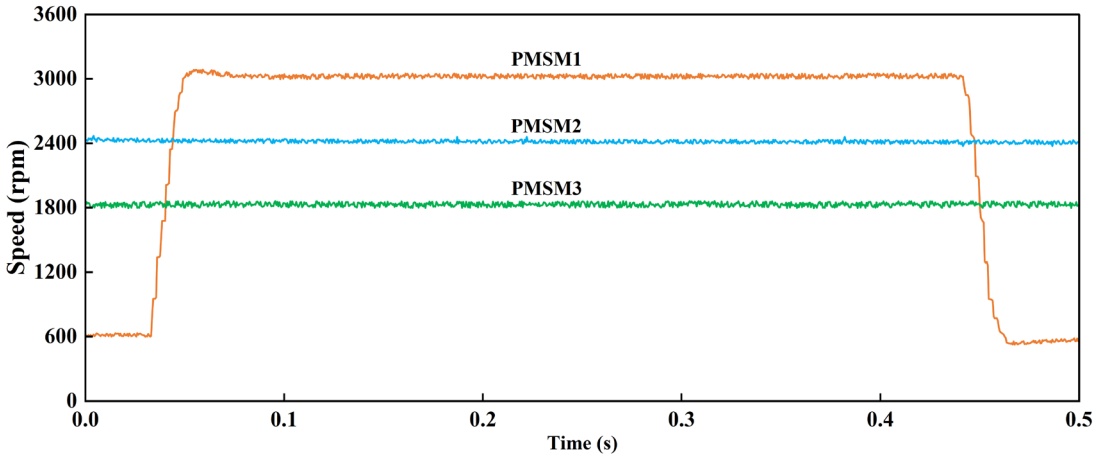


Figure 20. The diagram of test result with speed step commands.

4.3.3. Comparison of Time Consumption with Pure Software Computing

To test the computing performance of the coprocessor IP, we compared the time taken by the pure software scheme to realize the current loop operation with the time taken by the coprocessor IP to realize the current loop closed-loop operation with the hardware and software coordination scheme. At 72 MHz clock, the time taken by the universal SoC using two schemes to complete the current loop closed-loop calculation is shown in Table 3.

Based on the data in the above table, the SoC embedded with the IP would significantly improve the speed of the closed-loop operation of the current loop using the hardware and software coordination scheme. Due to the parallelism of the coprocessor IP hardware and the fact that the IP supports pipeline operation, the more motors required to compute, the more the acceleration of the IP. For multi-axis servo control, the closed-loop operation of the current loop of all motors must be completed in a current loop control cycle. The pure software scheme has a long calculation time due to serial execution, making the computing power of the universal SoC unable to meet the multi-axis servo control. The use of the coprocessor IP with the hardware and software coordination scheme significantly reduced

the closed-loop operation time of the current loop; besides, the extremely short current loop operation speed markedly improved the response speed and bandwidth of the current loop, making the single low-power universal SoC to efficiently complete the multi-axis servo control.

Table 3. Comparison of computation time.

Number of Motors	Pure Software Solution (us)	Software and Hardware Cooperation Scheme (us)
1	64.72	1.80
2	132.48	1.90
3	198.15	1.99
4	262.88	2.09
5	326.80	2.19
6	330.12	2.28

4.4. ASIC Implementation Results

The ASIC implementation of the IP was evaluated to promote rapid integration of the IP into a universal SoC. The IP was synthesized under CSMC 90 nm process library, and it could run at a frequency of 90 MHZ. In the synthesis process, the low power design method of clock gating is used to reduce the IP power consumption. The DC tool reported that the power consumption of the coprocessor IP was 4.46 mw. The final layout is shown in Figure 21. The power supply voltage of the fixed core was 1.35 V, and the core area was 0.42 mm^2. The dark-blue area in the layout was the Interface module, with an area ratio of 7.5%. The remaining areas were operation core modules. As the SVPWM unit of the operation core was embedded with an overmodulation algorithm circuit, the area accounted for the largest proportion of the operation core. The wathet area was the layout area of the SVPWM module, which accounted for 28.9% of the total IP area.

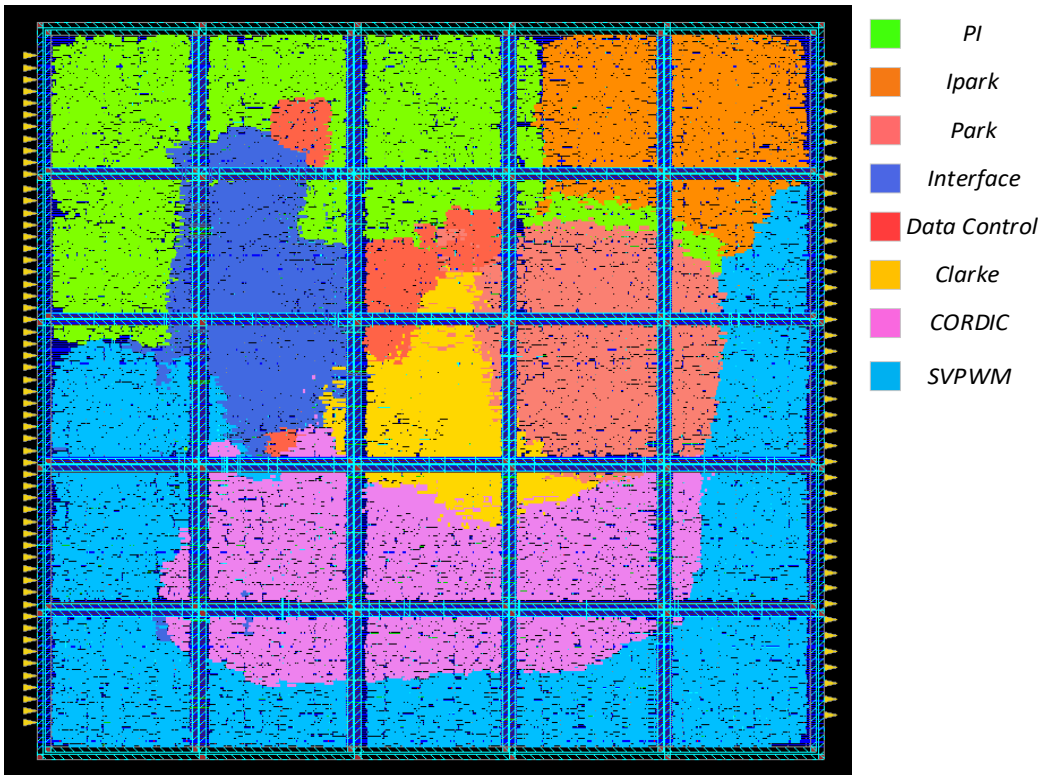


Figure 21. Layout diagram of IP.

ASIC experimental results indicated that the IP had benefits, including low area and low power consumption, and could be embedded in a universal SoC with low area overhead and low power consumption to aid in multi-axis servo control.

5. Conclusions

In conclusion, we introduced a coprocessor IP core based on an AMBA bus for accelerating the current loop operation of multi-motor vector control. As a consequence, the IP efficiently accelerated the current loop operation based on time division multiplexing technology and four key design methods. Additionally, we proposed a supporting SoC integration method and hardware–software coordination scheme for the coprocessor IP. The experimental findings highlighted the significance of this study, which include: (a) the interface of the coprocessor IP adopted AMBA bus, which can be flexibly integrated into the universal SoC; (b) the current and voltage values of the coprocessor IP were normalized data, allowing the SoC embedded in the IP to easily adapt to multi-axis servo systems with different motor types; (c) the coprocessor IP consumed fewer resources in the universal SoC, and the layout area, as well as power consumption of the IP, were lower; and (d) the operation of the coprocessor IP was quickly executed, and the hardware and software coordination scheme supporting the IP significantly reduced the execution time of the vector control current loop, making the low-cost and low-power universal SoC embedded in the coprocessor IP have a fast current and speed response when achieving multi-axis servo control.

Author Contributions: Conceptualization, J.X.; Methodology, J.X. and M.C.; Software, J.X., X.J. and Q.L.; Validation, L.S., H.L., F.W. and P.W.; Formal analysis, J.X. and M.C.; Investigation, J.X., M.C. and C.L.; Resources, X.J. and Q.L.; Data curation, J.X. and M.C.; Writing—original draft, P.W.; Writing—review & editing, J.X.; Visualization, M.C.; Supervision, P.W.; Project administration, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by Industry University Research Innovation Fund of the Ministry of Education of China "FPGA based edge detection algorithm to correct trapezoidal distortion" (June 2021–May 2022, hosted by 2020HYA04004).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mishra, I.; Tripathi, R.N.; Hanamoto, T. Synchronization and Sampling Time Analysis of Feedback Loop for FPGA-Based PMSM Drive System. *Electronics* **2020**, *9*, 1906. [CrossRef]
2. Mishra, I.; Tripathi, R.; Singh, V.; Hanamoto, T. Step-by-Step Development and Implementation of FS-MPC for a FPGA-Based PMSM Drive System. *Electronics* **2021**, *10*, 395. [CrossRef]
3. Chou, H.H.; Kung, Y.S.; Quynh, N.V.; Cheng, S. Optimized FPGA design, verification and implementation of a neuro-fuzzy controller for PMSM drives. *Math. Comput. Simul.* **2013**, *90*, 28–44. [CrossRef]
4. Banjanovic-Mehmedovic, L.; Mujkic, A.; Babic, N.; Secic, J. Hexapod Robot Navigation Using FPGA Based Controller. In *International Conference "New Technologies, Development and Applications"*; Springer: Cham, Switzerland, 2019; pp. 42–51. [CrossRef]
5. Park, J.-H.; Lim, H.-S.; Lee, G.-H.; Lee, H.-H. A Study on the Optimal Control of Voltage Utilization for Improving the Efficiency of PMSM. *Electronics* **2022**, *11*, 2095. [CrossRef]
6. Amornwongpeeti, S.; Ekpanyapong, M.; Chayopitak, N.; Monteiro, J.; Martins, J.; Afonso, J.L. A single chip FPGA-based cross-coupling multi-motor drive system. *IEICE Electron. Express* **2015**, *12*, 20150383. [CrossRef]
7. Ooi, C.P.; Hew, W.P.; Rahim, N.A.; Kuan, L.C. FPGA-based field-oriented control for induction motor speed drive. *IEICE Electron. Express* **2009**, *6*, 290–296. [CrossRef]
8. Boukaka, S.; Teiar, H.; Chaoui, H.; Sicard, P. FPGA implementation of an adaptive fuzzy logic controller for PMSM. In Proceedings of the 7th IET International Conference on Power Electronics, Machines and Drives (PEMD 2014), Manchester, UK, 8–10 April 2014; pp. 1–6.
9. Ricci, S.; Meacci, V. Simple Torque Control Method for Hybrid Stepper Motors Implemented in FPGA. *Electronics* **2018**, *7*, 242. [CrossRef]
10. Ma, Z.; Zhang, X. FPGA Implementation of Sensorless Sliding Mode Observer With a Novel Rotation Direction Detection for PMSM Drives. *IEEE Access* **2018**, *6*, 55528–55536. [CrossRef]

11. Sirisha, B.; Kumar, P.S. A simplified and generalised SVPWM method including over modulation zone for seven level diode clamped inverter-FPGA implementation. *Int. J. Power Electron.* **2019**, *10*, 350–366. [CrossRef]
12. Shu, Z.; Tang, J.; Guo, Y.; Lian, J. An Efficient SVPWM Algorithm With Low Computational Overhead for Three-Phase Inverters. *IEEE Trans. Power Electron.* **2007**, *22*, 1797–1805. [CrossRef]
13. Hu, H.; Yao, W.; Lu, Z. Design and Implementation of Three-Level Space Vector PWM IP Core for FPGAs. *IEEE Trans. Power Electron.* **2007**, *22*, 2234–2244. [CrossRef]
14. Di Benedetto, L.; Donisi, A.; Licciardo, G.D.; Liguori, R.; Piccirilli, E.; Lanzotti, E.; Rubino, A. Implementation of Hardware Architecture for SVPWM With Arbitrary Parameters. *IEEE Access* **2022**, *10*, 32381–32393. [CrossRef]
15. Tsai, M.-F.; Tseng, C.-S.; Cheng, P.-J. Implementation of an FPGA-Based Current Control and SVPWM ASIC with Asymmetric Five-Segment Switching Scheme for AC Motor Drives. *Energies* **2021**, *14*, 1462. [CrossRef]
16. Yilmaz, A.R.; Erkmen, B. FPGA-Based Space Vector PWM and Closed Loop Controllers Design for the Z Source Inverter. *IEEE Access* **2019**, *7*, 130865–130873. [CrossRef]
17. Xie, Q.; Qiu, J. Current-Loop Bandwidth Extension for PMSM Servo System Based on SiC Inverter and FPGA. In Proceedings of the 2021 IEEE 4th Student Conference on Electric Machines and Systems (SCEMS), Huzhou, China, 1–3 December 2021; pp. 1–4. [CrossRef]
18. Marufuzzaman, M.; Reaz, M.; Rahman, M.S.; Ali, M. FPGA implementation of an intelligent current dq PI controller for FOC PMSM drive. In Proceedings of the 2010 International Conference on Computer Applications and Industrial Electronics, Taichung, Taiwan, 15–17 June 2011.
19. Rogers, P.; Kavasseri, R.; Smith, S.C. An FPGA-in-the-loop approach for HDL motor controller verification. In Proceedings of the 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 4–6 December 2017.
20. Tufekci, B.; Onal, B.; Dere, H.; Ugurdag, H.F. Efficient FPGA Implementation of Field Oriented Control for 3-Phase Machine Drives. In Proceedings of the 2020 IEEE East-West Design & Test Symposium (EWDTS), Varna, Bulgaria, 4–7 September 2020; pp. 1–5. [CrossRef]
21. Liu, Y.; Li, T.; Zhou, Z.; Xu, P. Design and implementation of an ASIC-based four-axis position servo system. In Proceedings of the 2008 International Conference on Electrical Machines and Systems, Wuhan, China, 17–20 October 2008.
22. Pengfei, L.; Yuping, H.; Yanbo, W.; Qing, Z.; Zelin, Y. Design of Multi-axis Motion Control and Drive System Based on Internet. In Proceedings of the 2019 22nd International Conference on Electrical Machines and Systems (ICEMS), Harbin, China, 11–14 August 2019; pp. 1–6.
23. Jin, F.; Wei, Q.; Hua, B.; Lu, D.; Bing, C. Using one FPGA to control two high-switching-frequency PMSM drive systems through a novel time-division multiplexing method. In Proceedings of the 2018 IEEE Applied Power Electronics Conference and Exposition (APEC), San Antonio, TX, USA, 4–8 March 2018.
24. Lai, C.-K.; Chien, W.-N.; Tsao, Y.-T. An FPGA-Based Multiple-Axis Velocity Controller and Stepping Motors Drives Design. *MATEC Web Conf.* **2016**, *71*, 05002. [CrossRef]
25. Gu, Q.; Li, Y.; Niu, P.; Sun, J. Optimized Design of SoC-based Control System for Multi-axis Drive. *Elektron. Ir Elektrotechnika* **2014**, *20*, 15–22. [CrossRef]
26. Sun, Y.; Ming, Y.; Chen, Y.; He, W.; Xu, D. An SoC-based platform for integrated multi-axis motion control and motor drive. In Proceedings of the 2018 International Power Electronics Conference (IPEC-Niigata 2018-ECCE Asia), Niigata, Japan, 20–24 May 2018.
27. Wu, W.; Su, D.; Yuan, B.; Li, Y. Intelligent Security Monitoring System Based on RISC-V SoC. *Electronics* **2021**, *10*, 1366. [CrossRef]
28. Romero, J.; Cuevas, N.; Roa, E. Energy Efficient Peripheral System Buses for Low-Area Low-Power SoCApplications IEEETransactions on Circuits Systems, I.I. *Express Briefs* **2020**, *67*, 866–870.
29. Mei, K.; Zhang, B.; Ge, C. A hierarchical and parallel SoC architecture for vision procesor. *IEICE Electron. Express* **2009**, *6*, 1380–1386. [CrossRef]
30. Zheng, X.; Hu, X.; Zhang, J.; Yang, J.; Cai, S.; Xiong, X. An Efficient and Low-Power Design of the SM3 Hash Algorithm for IoT. *Electronics* **2019**, *8*, 1033. [CrossRef]
31. Sahu, N.; Kumar, A.; Kandari, R. Design and Verification of APB IP Core using Different Verification Methodologies. In Proceedings of the 2022 International Conference on Breakthrough in Heuristics And Reciprocation of Advanced Technologies (BHARAT), Visakhapatnam, India, 7–8 April 2022; pp. 150–154.
32. Aravind, M.; Bhattacharya, T. FPGA based Synchronized Sinusoidal Pulse Width Modulation with smooth transition into overmodulation and six step modes of operation for three phase AC motor drives. In Proceedings of the 2012 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES), Bengaluru, India, 16–19 December 2012; pp. 1–6. [CrossRef]
33. Jing, R.; Zhang, G.; Wang, G.; Bi, G.; Ding, D.; Xu, D. An Overmodulation Strategy Based on Voltage Vector Space Division for High-Speed Surface-Mounted PMSM Drives. *IEEE Trans. Power Electron.* **2022**, *37*, 15370–15381. [CrossRef]