Optimized Design of SoC-based Control System for Multi-axis Drive

Qiang Gu[1], Yesong Li[1], Panqing Niu[1], Jiajiang Sun[1]
[1]*Department of Control Science and Engineering, School of Automation, Huazhong University of Science & Technology,*
1037 Luoyu Road, Hongshan District, Wuhan 430074, P. R. China
qianggu@hust.edu.cn

***Abstract*—This study presents an optimized system-on-chip (SoC) based multi-axis control system for permanent magnet synchronous motor (PMSM). The multi-axis control system becomes considerably complex with increasing controlled axes and complicated control algorithms. This paper aims to provide a balanced design considering control performance, hardware constrains and feasibility. To cope with this problem, the system design is optimized by using pipeline and time division multiplexing technology. Due to the optimized architecture and the computation capability of SoC, the performance of the whole system can be preserved. To demonstrate this, a SoC implementation of the control system is presented and comprehensively analyzed. Herein, the vector control in the current loop is explained and an anti-windup proportional-integral (PI) controller is adopted in the speed loop. The sampling frequencies of the current loop and the speed loop are 16 KHz and 4 KHz respectively. Consequently, the execution time is several microseconds with reasonable consumed resources. Experimental results are shown to prove the efficiency of the proposed SoC-based solution.**

***Index Terms*—System-on-chip (SoC), motor drives, permanent magnet motors, proportional-integral (PI) control.**

I. INTRODUCTION

For the fast progress of very large scale integration (VLSI) and electronic design automation (EDA), system-on-chip (SoC) technology is getting widely used due to its efficiency in term of control performance, cost, power consumption and development time in industrial control systems. The main advantages of SoC-based solution are the shorter execution time than fully software-solution and the less complicated development than fully hardware-solution.

Especially in the field of motor drives, such as permanent magnet synchronous motor (PMSM) [1]–[3] and brushless DC motor (BLDC) [4], [5] drives which have widely used in industrial control systems, the SoC-based solution is more efficient than the digital signal processor (DSP) based solution [6]–[8]. In order to reduce system cost and power consumption, the hardware/software co-design methodology has created an opportunity to make optimal SoC designs for increasingly sophisticated control systems [9], [10].

The research of SoC-based system for PMSM has become a popular field, which focuses on current control [11]–[14] speed control [10], [15], sensorless control [16], [17], multi-axis system [18], [19], etc. Compared with the network-based solution [20], [21], the SoC-based solution is simple, real-time and low-cost for small and medium-sized multi-axis control systems. The SoC environment can be constructed inside field programmable gate array (FPGA) architectures with embedded hard processor cores such as PowerPC or ARM, soft processor cores such as Nios II or Micro-blaze and some other dedicated blocks such as random access memory (RAM) [22], [23].

Nowadays, more and more researchers have focused on implementation of controllers based on SoC and FPGA for robot and computer numerical control machine tool (CNC) applications. Shao et al. [24] presented a motion control architecture using FPGA for servo control and DSP for trajectory generation and dynamic compensation. Cho et al. [25] presented the implementation of a FPGA-based motion controller including interpolation, kinematics calculation, servo control, etc. Martinez-Prado et al. [26] described the implementation of a multi-axis motion controller for robotic applications on low-cost FPGA. Astarloa et al. [27] presented a SoC-based multi-axis system where the intensive operations and the trajectory computation are realized by hardware and software respectively. Chen et al. [28] developed a SoC-based CNC system integrating a general-purpose processor, a motion control core, an axis control core, etc. But motor drive modules cannot be fully realized in the above solutions.

Ying-Shieh Kung et al. [7], [29] developed SoC-based motion control systems for different applications. In these solutions, current vector controller for PMSMs and interfaces can be realized by hardware and the other function modules such as position/speed controller and trajectory calculation can be realized by software. However, the hardware characteristic of parallelism cannot be fully utilized in most of the solutions for multi-axis systems. Consequently, the resource consumption and the execution time cannot be optimized effectively. These solutions lead to great negative effects with increasing controlled axes and complicated control algorithms. In order to exploit the advantages, an optimized SoC-based architecture for multi-axis drive is presented in Fig. 1, where the multi-axis timing schedule module is developed by hardware using pipeline and time division multiplexing technology.

Manuscript received July 25, 2013; accepted March 12, 2014.
This research was funded by the National Major Science and Technology Project (2012ZX04001-012).

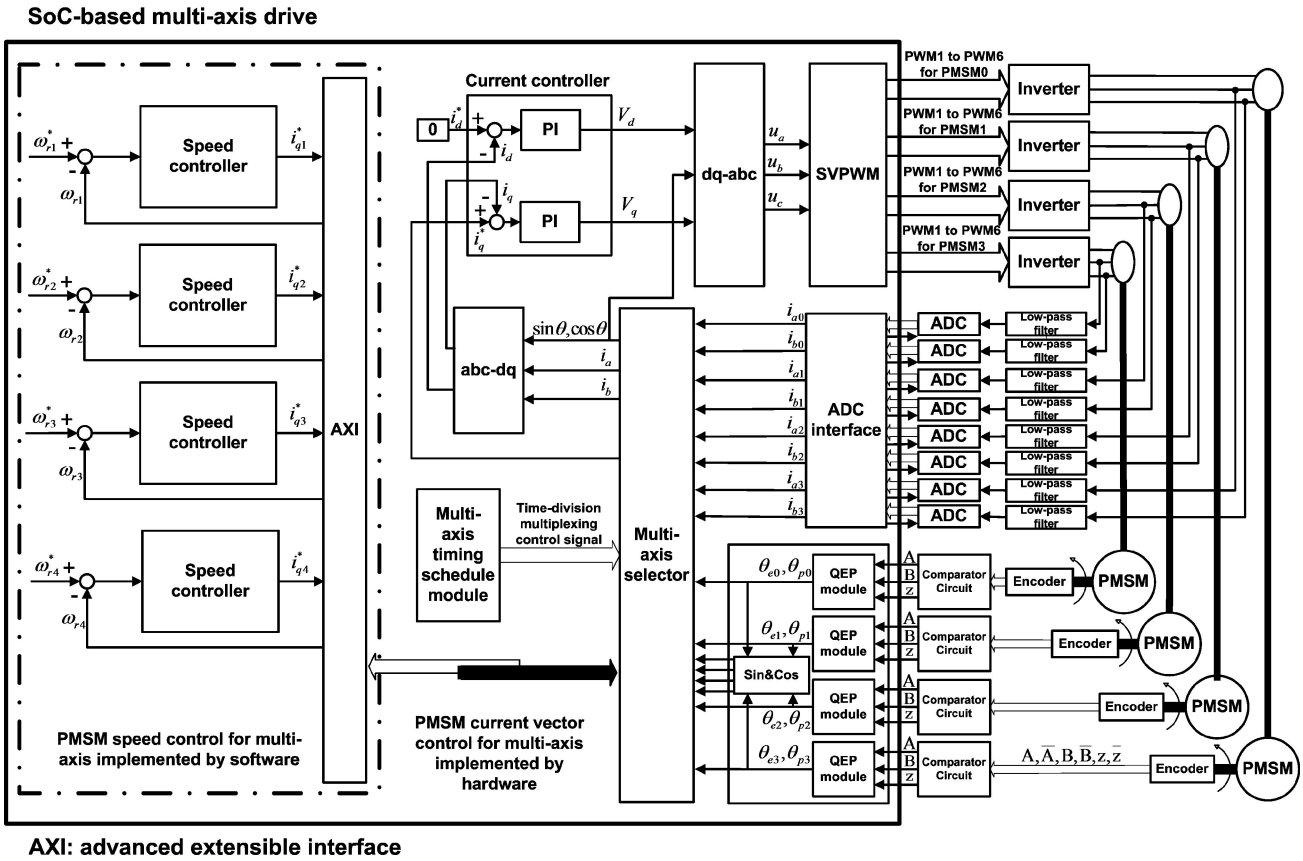

Fig. 1. Architecture of the SoC-based control system for the multi-axis drive.

This solution can be implemented in both high-performance devices and low-cost devices. Furthermore, we adopt the new-generation SoC Zynq-7000 to develop the whole system with embedded hard processor cores, which contains dual ARM Cortex-A9 processing system including hardened memory controllers and peripherals.

II. System Description of Multi-axis Drive and Its Optimized Design

The architecture of the SoC-based control system for the multi-axis drive is illustrated in Fig. 1. The current vector controller for PMSM is developed by hardware. The anti-windup proportional-integral (PI) speed controller is developed by software in the embedded processor. Fig. 1 presents four motor drives in the proposed system and it is quite easy to be reconfigured for more motor drives with a little additional resource.

A. Mathematical Model of Permanent Magnet Synchronous Motor

The typical mathematical model of a PMSM can be expressed by:

$$\frac{di_d}{dt} = -\frac{R_s}{L_d} i_d + \mu_e \frac{R_s}{L_d} i_q + \frac{1}{L_d} u_d, \quad (1)$$

$$\frac{di_q}{dt} = -\mu_e \frac{L_d}{L_q} i_d - \frac{R_s}{L_q} i_q - \mu_e \frac{f}{L_q} + \frac{1}{L_q} u_q, \quad (2)$$

where i_d and i_q are the d-axis and q-axis currents; u_d and u_q are the d-axis and q-axis voltages; L_d and L_q are the d-axis and q-axis inductances; R_s are the stator winding resistance; ω_e is the electrical angular speed; λ_f is the permanent magnet flux linkage.

The current vector control method is used in the proposed PMSM drive. The PMSM will be decoupled and controlled like a DC motor when the i_d is controlled to 0. Hence, the electromagnetic torque can be described according to the following equations:

$$T_e = K_t i_q, \quad (3)$$

$$K_t = \frac{3P}{4} \lambda f. \quad (4)$$

Finally, the dynamic equation of PMSM drive system can be written as below

$$T_e - T_L = J_m \frac{d\mu_r}{dt} + B_m \mu_r, \quad (5)$$

where T_e is the motor torque, K_t is the torque constant, P is the number of PMSM poles, T_L is the load torque, J_m is the inertial value, ω_r is the rotor speed, and B_m is the damping ratio.

The current loop of the PMSM drives shown in Fig. 1 is implemented by hardware, including current controller module (with two PI controllers), abc-dq transformation (with Clarke module and Park module), dq-abc transformation (with inverse Park module and inverse Clarke module), space vector pulse width modulation (SVPWM) module, quadrature encoded pulses (QEP) module, analog to digital converter (ADC) interface, and multi-axis timing schedule module.

B. Algorithm Modular Description and Partition

The aim of the algorithm modular partition, which was summarized in [9], is to decompose the design into functional modules with different levels of granularity. The functional modules consist of coordinate transformation, current controller and SVPWM, which are appropriate for the optimized multi-axis drive design.

The coordination transformation in Fig. 1 includes Clarke, inverse Clarke, Park, inverse Park, which can be obtained by Park:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos^- & \sin^- \\ -\sin^- & \cos^- \end{bmatrix} \begin{bmatrix} i_\sim \\ i \end{bmatrix}, \tag{6}$$

Clarke:

$$\begin{bmatrix} i_\sim \\ i \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}, \tag{7}$$

Inverse Park:

$$\begin{bmatrix} u_\sim \\ u \end{bmatrix} = \begin{bmatrix} \cos^- & -\sin^- \\ \sin^- & \cos^- \end{bmatrix} \begin{bmatrix} u_d \\ u_q \end{bmatrix}, \tag{8}$$

Inverse Clark:

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} u_\sim \\ u \end{bmatrix}, \tag{9}$$

where i_a , i_b and i_c are the stator currents respectively; i_α and i_β are the α-axis and β-axis currents; u_α and u_β are the α-axis and β-axis voltages; u_a , u_b and u_c are the stator voltages.

In Fig. 1, two digital PI controllers are presented in the current loop and described below:

$$\begin{cases} u_d(n) = K_{cdp} e_d(n) + \sum_{i=0}^{n} K_{cdi} T_c e_d(i), \\ u_q(n) = K_{cqp} e_q(n) + \sum_{i=0}^{n} K_{cqi} T_c e_q(i), \end{cases} \tag{10}$$

with

$$\begin{cases} e_d(n) = i_d^*(n) - i_d(n), \\ e_q(n) = i_q^*(n) - i_q(n), \end{cases} \tag{11}$$

where i_d^* and i_q^* are the current commands of d-axis and q-axis; e_d and e_q represent the d-axis current error and q-axis current error; K_{cdp}, K_{cqp}, K_{cdi} and K_{cqi} are the P control gain and I control gain of d-axis and q-axis current controller respectively; T_c is the sample time of the current controller. Furthermore, an anti-windup strategy is presented according to the following equations:

$$I_d(n) = \begin{cases} I_d(n-1) + K_{cdi} T_c e_d(n), & if\ (|u_d(n)| < u_{d\max}), \\ I_d(n-1), & if\ (|u_d(n)| \geq u_{d\max}), \end{cases} \tag{12}$$

$$I_q(n) = \begin{cases} I_q(n-1) + K_{cqi} T_c e_q(n), & if\ (|u_q(n)| < u_{q\max}), \\ I_q(n-1), & if\ (|u_q(n)| \geq u_{q\max}), \end{cases} \tag{13}$$

$$u_{q\max} = \sqrt{u_{\max}^2 - u_d^2(n)}, \tag{14}$$

with:

$$\begin{cases} I_d(n) = \sum_{i=0}^{n} K_{cdi} T_c e_d(i), \\ I_q(n) = \sum_{i=0}^{n} K_{cqi} T_c e_q(i), \end{cases} \tag{15}$$

where I_d and I_q denote the integrator output of the proposed d-axis and q-axis PI controllers; $u_{d\max}$ and $u_{q\max}$ are the limit value of the d-axis and q-axis PI controllers output; $u_{\max}$ is the square sum limit value of the d-axis and q-axis PI controllers output.

C. Anti-windup PI Speed Controller

The tracking back calculation is widely used to avoid integrator windup [30], [31]. The anti-windup PI speed controller proposed in the literature is shown in Fig. 2, where K_{sp}, K_{si} and K_{sa} denote the P control gain, I control gain and the anti-windup gain respectively; i_{sp} and i_{si} are the P control output and I control output; ω_r^* and ω_r are the speed commend and rotor speed; $p_{s\max}$ and $o_{\max}$ are the output limits of P control and PI control.

The difference e_s between the controller output and the saturated output is used to reduce the integrator input. The tracking back calculation is not to reset the integrator but to regulate the integrator input dynamically.

Once i_{sp} exceeds the saturation limit, the desaturation of the tracking back calculation is slow and system performance is negatively affected.

Hence, the integrator is reset in one sampling period if i_{sp} is saturated. And the sample time T_s of the PI controller is determined according to the sampling frequency of the speed loop.

The proposed anti-windup PI controller is described as follows:

$$i_q^*(n) = \begin{cases} o_{\max}, & if\ (i_{sp}(n) \geq o_{\max}), \\ p_s(n) + i_{si}(n), & if\ (|i_{sp}(n)| < o_{\max}), \\ -o_{\max}, & if\ (i_{sp}(n) \leq -o_{\max}), \end{cases} \tag{16}$$

$$p_s(n) = \begin{cases} p_{s\max}, & if\ (i_{sp}(n) \geq p_{s\max}), \\ i_{sp}, & if\ (|i_{sp}(n)| < p_{s\max}), \\ -p_{s\max}, & if\ (i_{sp}(n) \leq -p_{s\max}), \end{cases} \tag{17}$$

$$i_{si}(n) = \begin{cases} i_{si}(n-1) + K_{si} T_s e_{rs}(n), & if\ (|i_{sp}(n)| < p_{s\max}), \\ 0, & if\ (|i_{sp}(n)| \geq p_{s\max}), \end{cases} \tag{18}$$

with

$$e_{rs}(n) = \mu_r^*(n) - \mu_r(n) - K_{sa}e_s(n), \qquad (19)$$

$$i_{sp}(n) = K_{sp}\left(\mu_r^*(n) - \mu_r(n)\right). \qquad (20)$$

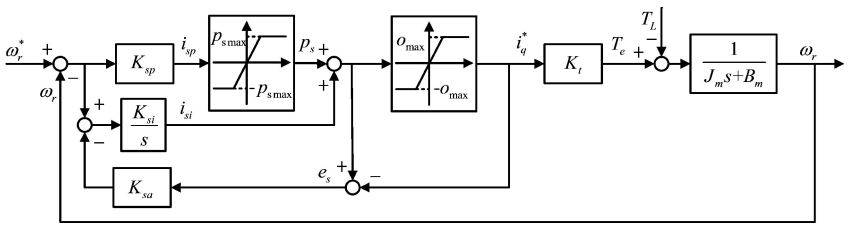

Fig. 2. Block diagram of the anti-windup PI speed controller.

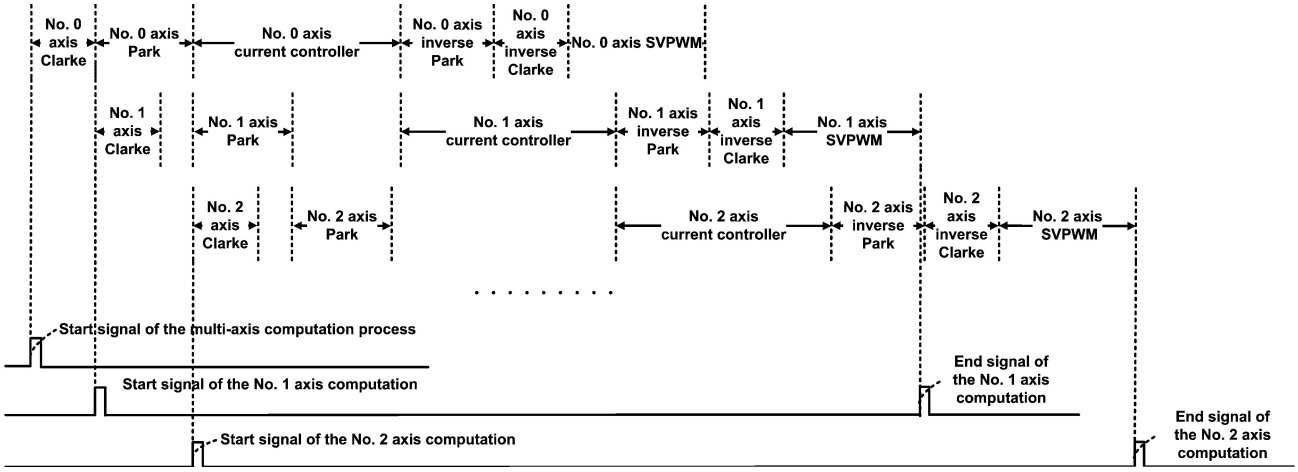

Fig. 3. Multi-axis timing diagram.

D. Optimized Multi-axis Drive Design

In order to optimize the multi-axis system design, pipeline and time division multiplexing technology are used here on the above algorithm modular partition. Compared with the single axis drive, the multi-axis drive is reliable with a little additional time and resource consumption.

Supposing the number of controlled axes is n, the multi-axis timing schedule module is utilized to complete the computation process from the No. 0 axis to the No. n-1 axis in order. And the computations, including Clarke transformation, Park transformation, current control algorithm, inverse Park transformation, inverse Clarke transformation and SVPWM, must be completed in order. The detailed multi-axis timing schedule is shown in Fig. 3. Considering the execution time of the modules is not identical, the activating conditions of each module for the No. j axis are 1) the last module for the No. j has finished and 2) this module for No. j-1 has also finished. The No. j axis denotes the motor to be controlled. The schedule scheme is summarized as follows.

Step 1: If the Clarke module for the No. j-1 axis finishes, the Clarke module for the No. j axis is activated.

Step 2: The Park module for the No. j axis is activated if 1) the Clarke module for the No. j axis finishes and 2) the Park module for the No. j-1 finishes.

Step 3: The current controller module for the No. j axis is activated if 1) the Park module for the No. j axis finishes and 2) the current controller module for the No. j-1 axis finishes.

Step 4: The inverse Park module for the No. j axis is activated if 1) the current controller module for the No. j axis finishes and 2) the inverse Park module for the No. j-1 axis finishes.

Step 5: The inverse Clarke module for the No. j axis is activated if 1) the inverse Park module for the No. j axis finishes and 2) the inverse Clarke module for No. j-1 finishes.

Step 6: The SVPWM module for the No. j axis is activated if 1) the inverse Clarke module for the No. j axis finishes and 2) the SVPWM module for No. j-1 finishes.

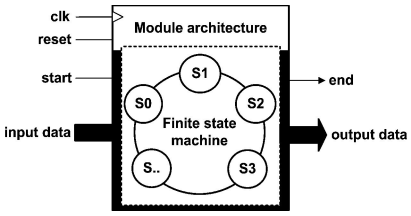

Fig. 4. General structure of the FSM module.

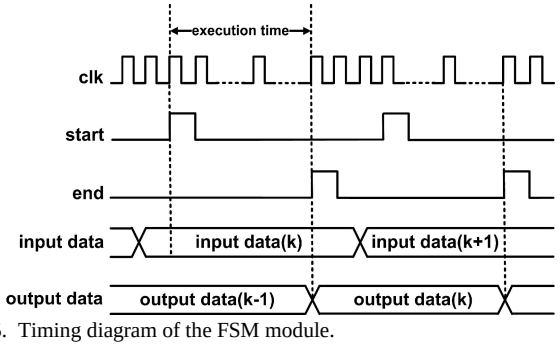

Fig. 5. Timing diagram of the FSM module.

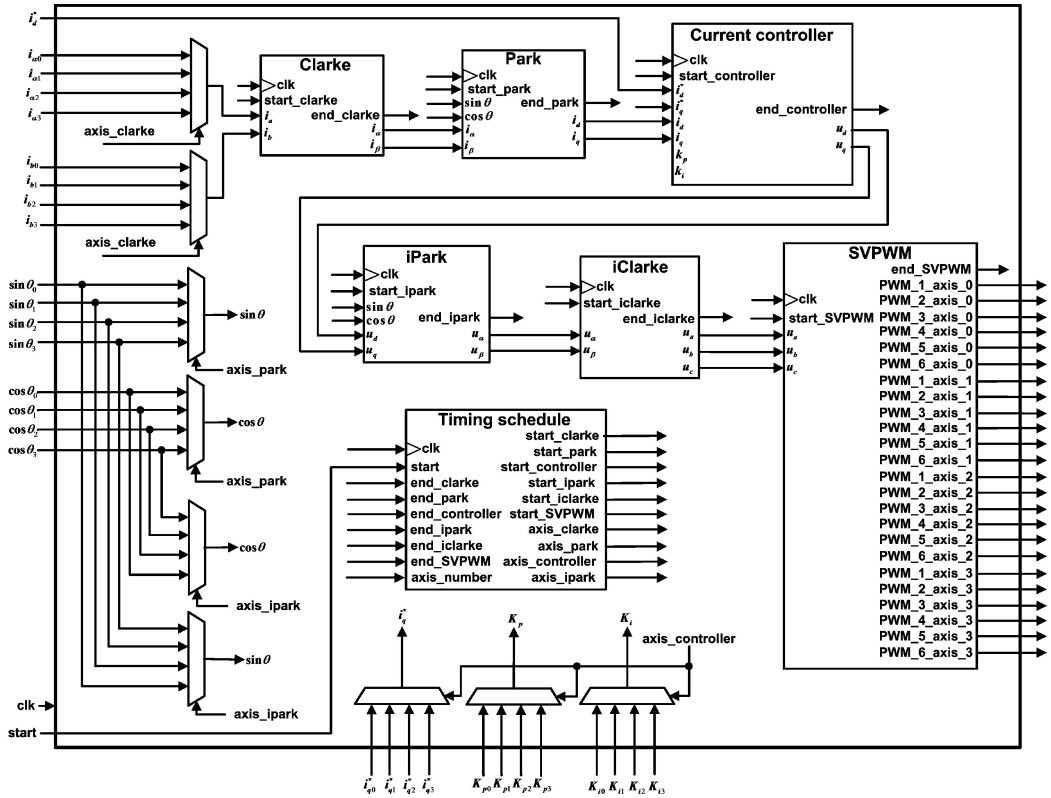

Fig. 6. Block diagram of the hardware realization circuit.

In addition, the multi-axis selector in Fig. 1 is used to transmit electrical angles and current signals to appropriate modules. Actually, the multi-axis timing schedule module and the multi-axis selector are auxiliary to each other.

III. SoC Implementation of the Proposed Design

A. Development and Analysis of the Hardware Solution

Considering performance, flexibility, power and cost, a standard module is developed. Finite state machine (FSM) method is efficient to reduce the consumed hardware resource for PMSM drive, which has been proved in [32]. Also it is advantageous for the optimized multi-axis drive design. Therefore, a standard FSM module is presented in Fig. 4, which has also been adopted in [13]. Figure 5 shows the timing diagram of the FSM module. The input signals are a start signal to activate the module and input data signals to receive algorithm data from the last module. The output signals are an end signal to be generated when the module finishes and output data signals to transfer algorithm data to the next module. The algorithm modules are realized with FSM including Clarke module, Park module, current controller module, inverse Park module, inverse Clarke module and SVPWM module. The multi-axis timing schedule module and the multi-axis selector manage the algorithm modules which are synchronized by the clock signal (clk).

Figure 6 shows the hardware realization of the multi-axis drive. The time schedule module generates axis signals, including axis_clarke, axis_park, axis_controller and axis_ipark, to control the algorithm data with the multi-axis selector.

In Fig. 7, the sequential timing diagram of the hardware solution is presented. The QEP module is synchronized with the encoders.

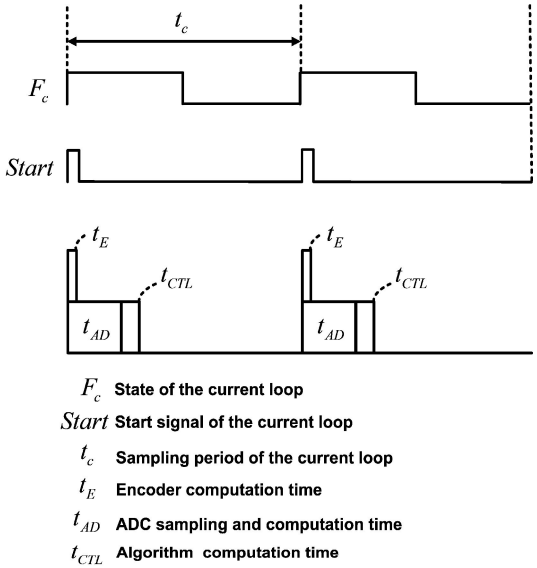

Fig. 7. Sequential timing diagram of the proposed hardware solution.

The ADC interface is managed by the current loop start signal *Start* with a sampling period t_c equal to 62.5 μs. The proposed hardware solution is realized in Xilinx Zynq-7000: XC7Z020, coded in Verilog hardware description language (Verilog HDL). Zynq-7000: XC7Z020 is based on the Xilinx all programmable SoC architecture that combines an industry-standard ARM dual-core Cortex-A9 MPCore processing system (PS) with Xilinx 7 series programmable logic (PL) containing 85120 logic cells, 220 programmable DSP slices and 560KB block ram [33]. Table I and Table II summarize the hardware area/time performance of the proposed solution.

TABLE I. HARDWARE AREA PERFORMANCE OF THE PROPOSED SOLUTION.

Module	**Consumed resource (Utilization)**			
	Slice register	Slice look-up table	Block ram	DSP48E1
ADC interface	468 (0.5 %)	762 (1.4 %)	0	0
QEP	481 (0.5 %)	610 (1.1 %)	3 (2.1 %)	0
Algorithm modules	3102 (2.9 %)	3723 (7.0 %)	1 (0.7 %)	5 (2.3 %)
AXI	8304 (7.8 %)	8606 (16.2 %)	0	0
Total	12355 (11.6 %)	13701 (25.8 %)	4 (2.8 %)	5 (2.3 %)

The Zynq-7000 device integrates advanced extensible interface (AXI) for high throughput data transfer between PS and PL. In this work, the execution time of the communication PL-to-PS t_l fixed to 1.78 μs is equal to the execution time of the communication PS-to-PL t_p.

TABLE II. HARDWARE TIME PERFORMANCE OF THE PROPOSED SOLUTION.

Module	**Latency**	**Execution time (μs)**
ADC interface	141	1.567
QEP	1	0.011
Algorithm modules	361	4.011
AXI	160	1.778

In Table I, the consumed resources of a 4-axis drive are obtained for 16-bits length and Q15 fixed-point format. In Table II, the hardware is implemented with a 90 MHz clock frequency in Zynq-7000. The execution time depending on the clock frequency has been calculated using the following equation

$$t_{ex} = Latency / f_{clk}\,, \tag{21}$$

where t_{ex} is the execution time and f_{clk} is the clock frequency. Note that the chosen ADC device in this paper is AD7265 which contains dual 12-bits, 3-channel ADCs and features throughput rates of up to 1 Msps. The ADC interface execution time denotes its SoC latency but not ADC device latency.

The results in Table II indicate that the execution time t_{ex} is very small compared with the sampling period t_c. In Table I, the consumed resources are obtained for the designed modules and interfaces. These modules and interfaces can also be implemented in low-cost devices except the AXI. Though the considerable consumed resources are needed for the AXI implementation, the AXI average throughput is over 124 Mbps. Consequently, the chosen hardware solution is efficient for reasonable consumed resources and short execution times.

B. SoC Development and Analysis of the Multi-axis Drive

The current vector controller and the proposed speed controller for multi-axis drive are implemented in PL and PS respectively. The sampling frequency of the current loop and the speed loop is designed with 16 KHz and 4 KHz. The data type in PL and PS is Q15 and Q16 fixed-point format respectively.

Figure 8 shows the sequential timing diagram of the implemented SoC-based multi-axis drive. Once signal sampling and current loop computation finish, the data is transmitted from PL to PS with communication latency. When the communication finishes, the speed loop computation starts immediately. Also, when the communication finishes from PS to PL in order to update current controller reference, the current loop computation begins with small latency. Consequently, the system performance is preserved due to the method of optimizing system latency.

IV. EXPERIMENT AND RESULTS

Experiments were carried out on a SoC-based 4-axis prototyping control system depicted in Fig. 1. The experimental setup is presented in Fig. 9 which is composed of four PMSMs with 2500 lines incremental encoders, a Zynq-7000 SoC board, an AD conversion board and a metal oxide semiconductor field effect transistor (MOSFET) drive board. The PMSM parameters are shown in Table III.

The AD conversion board is used to convert analog signals to digital signals for the measured currents and to provide the interface between the SoC board and the MOSFET drive board. A serial interface RS232 is used to send controller commends and set system parameters from the host PC to the SoC. The experimental results are plotted by Matlab software.

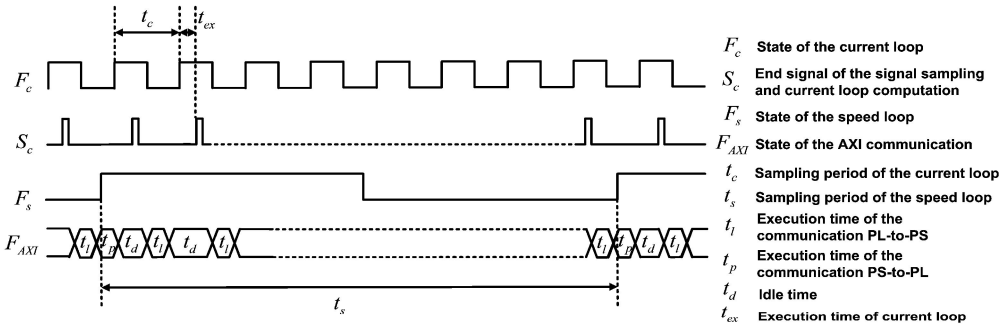

Fig. 8. Sequential timing diagram of the implemented SoC-based multi-axis drive.

TABLE III. PMSM PARAMETERS.

Rate voltage	36 V
Rate power	200 W
Rate torque	0.64 Nm
Rate current	7.6 A
Stator winding resistance	0.39 Ω
Armature inductance	0.33 mH
Inertia	0.8677×10^{-4} kgm^2

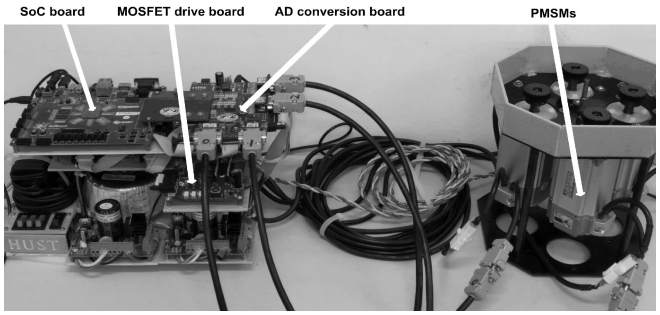

Fig. 9. Experimental setup.

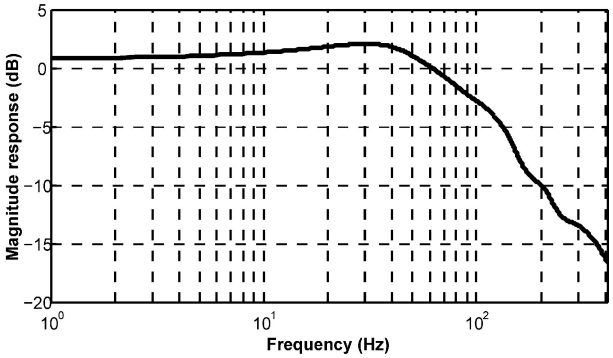

Fig. 10. Magnitude-frequency response.

Figure 10 and Fig. 11 show the magnitude-frequency response and phase-frequency response. The results are obtained by an applied sinusoidal reference (with variable frequency and magnitude 50 rpm).

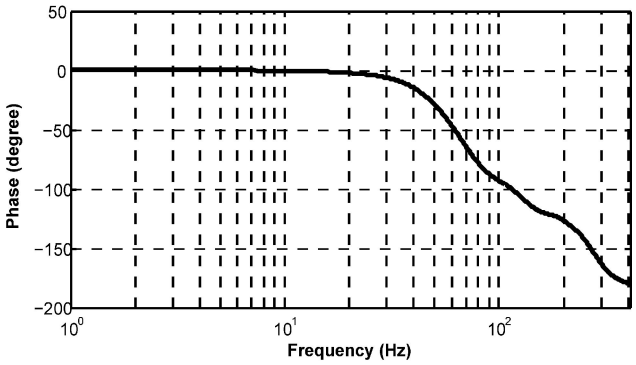

Fig. 11. Phase-frequency response.

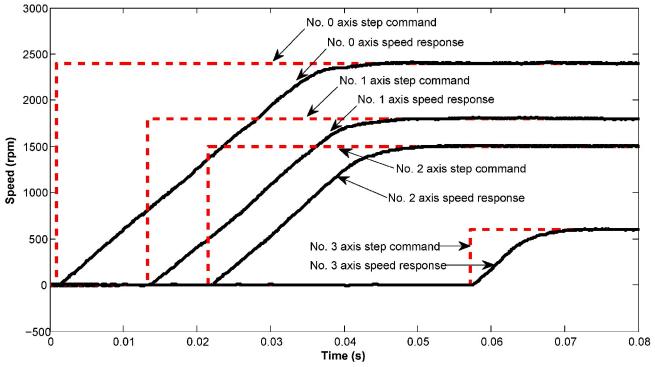

Fig. 12. The step response of each PMSM rotor speed for 0 rpm-2400 rpm, 0 rpm-1800 rpm, 0 rpm-1500 rpm and 0 rpm-600 rpm at different step time 1 ms, 13 ms, 21 ms and 57 ms respectively.

Also, Fig. 12 presents a step response. Figure 13 shows the current response of i_q^*, i_q and i_d for Fig. 12 with $i_d^* = 0$. The step response of each PMSM rotor speed is for 0 rpm–2400 rpm, 0 rpm–1800 rpm, 0 rpm–1500 rpm and 0 rpm–600 rpm at different step time 1 ms, 13 ms, 21 ms and 57 ms respectively. These results indicate the designed speed and current controllers have a good tracking performance and high control bandwidth. The SoC-based control system can be set flexibly.

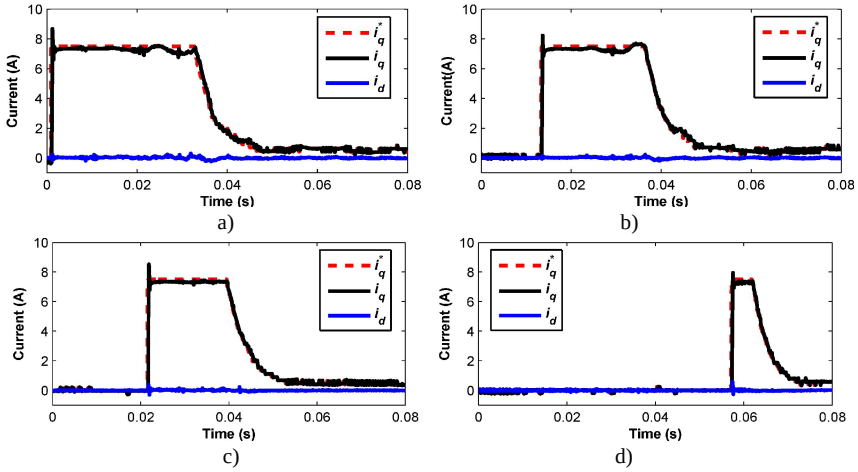

Fig. 13. Current response of i_q^*, i_q and i_d : (a) No. 0 axis, (b) No. 1 axis, (c) No. 2 axis, (d) No. 3 axis.

V. CONCLUSIONS

An optimized SoC-based control system for the multi-axis drive is demonstrated in this paper. The algorithm modules have been realized efficiently by using the pipeline and time division multiplexing technology. Thanks to the optimized architecture and the hardware computation capability, the execution time is very small compared with the sampling period of the current loop and the total resource consumption is reasonable. The experimental results have validated the

good performance of the proposed system.

The proposed design has three benefits described as follows.

1) The system parallel processing is available and efficient by hardware with the multi-axis timing schedule and the specific architecture.

2) The high performance on-chip interface AXI is used for development, supervision and control of the multi-axis system. It is more reliable and flexible than the external bus interface.

3) The system is configurable to expand to a multi-axis control system for complex applications conveniently, such as robot and CNC.

ACKNOWLEDGMENT

The authors would like to thank every reviewer for their valuable comments which improve this paper considerably.

REFERENCES

[1] L. Qin, X. Zhou, P. Cao, "New control strategy for PMSM driven bucket wheel reclaimers using GA-RBF neural network and sliding mode control", *Elektronika ir Elektrotechnika*, no. 6, pp. 113–116, 2012.

[2] N. Levin, S. Orlova, V. Pugachov, B. Ose-Zala, E. Jakobsons, "Methods to reduce the cogging torque in permanent magnet synchronous machines", *Elektronika ir Elektrotechnika*, vol. 19, no. 1, pp. 23–26, 2013.

[3] P. Brandstetter, T. Krecek, "Sensorless control of permanent magnet synchronous motor using voltage signal injection", *Elektronika ir Elektrotechnika*, vol. 19, no. 6, pp. 19–24, 2013.

[4] I. Topaloglu, F. Korkmaz, H. Mamur, R. Gurbuz, "Closed-loop speed control of PM-BLDC motor fed by six step inverter and effects of inertia changes for desktop CNC machine", *Elektronika ir Elektrotechnika*, vol. 19, no. 1, pp. 7–10, 2013.

[5] B. Karaliunas, "Study on the characteristics of electronically commutated motor", *Elektronika ir Elektrotechnika*, no. 3, pp. 31–34, 2011.

[6] L. Idkhajine, E. Monmasson, A. Maalouf, "Fully FPGA-based sensorless control for synchronous AC drive using an extended Kalman filter", *IEEE Trans. Industrial Electronics*, vol. 59, no. 10, pp. 3908–3918, 2012. [Online]. Available: http://dx.doi.org/10.1109/TIE.2012.2189533

[7] Y.-S. Kung, R.-F. Fung, T.-Y. Tai, "Realization of a motion control IC for X-Y table based on novel FPGA technology", *IEEE Trans. Industrial Electronics*, vol. 56, no. 1, pp. 43–53, 2009. [Online]. Available: http://dx.doi.org/10.1109/TIE.2008.2005667

[8] L. Idkhajine, E. Monmasson, A. Maalouf, "Extended Kalman filter for AC drive sensorless speed controller - FPGA-based solution or DSP-based solution", in *Industrial Electronics (ISIE), 2010 IEEE International Symposium*, Bari, 2010, pp. 2759–2764.

[9] I. Bahri, L. Idkhajine, E. Monmasson, M. E. A. Benkhelifa, "Optimal hardware/software partitioning of a system on chip FPGA-based sensorless AC drive current controller", *Mathematics and Computers in Simulation*, vol. 90, pp. 145–161, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.matcom.2012.06.008

[10] H.-H. Chou, Y.-S. Kung, N. V. Quynh, S. Cheng, "Optimized FPGA design, verification and implementation of a neuro-fuzzy controller for PMSM drives", *Mathematics and Computers in Simulation*, vol. 90, pp. 28–44, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.matcom.2012.07.012

[11] A. S. N. Mokhtar, M. B. I. Reaz, M. Marufuzzaman, M. A. M. Ali, "Hardware implementation of a high speed inverse park transformation using CORDIC and PLL for FOC brushless servo drive", *Elektronika ir Elektrotechnika*, vol. 19, no. 3, pp. 23–26, 2013.

[12] M. Marufuzzaman, M. B. I. Reaz, M. A. M. Ali, L. F. Rahman, "Hardware approach of two way conversion of floating point to fixed point for current dq PI controller of FOC PMSM drive", *Elektronika ir Elektrotechnika*, no. 7, pp. 79–82, 2012.

[13] M. W. Naouar, A. A. Naassani, E. Monmasson, I. Slama-Belkhodja, "FPGA-based predictive current controller for synchronous machine speed drive", *IEEE Trans. Power Electronics*, vol. 23, no. 4, pp. 2115–2126, 2008. [Online]. Available: http://dx.doi.org/10.1109/TPEL.2008.924849

[14] L. Idkhajine, E. Monmasson, M. W. Naouar, A. Prata, K. Bouallaga, "Fully integrated FPGA-based controller for synchronous motor drive", *IEEE Trans. Industrial Electronics*, vol. 56, no. 10, pp. 4006–4017, 2009. [Online]. Available: http://dx.doi.org/10.1109/TIE.2009.2021591

[15] B. Alecsa, M. N. Cirstea, A. Onea, "Simulink modeling and design of an efficient hardware-constrained FPGA-based PMSM speed controller", *IEEE Trans. Industrial Informatics*, vol. 8, no. 3, pp. 554–562, 2012. [Online]. Available: http://dx.doi.org/10.1109/TII.2012.2193891

[16] Z. Ma, R. Kennel, "System-on-Chip sensorless control of PMSM combining signal inject tion and flux observer", in *Proc. 7th Int. Power Electronics and Motion Control Conf.*, Harbin, 2012, vol. 2, pp. 1201–1205.

[17] V. D. Colli, R. D. Stefano, F. Marignetti, "A system-on-chip sensorless control for a permanent-magnet synchronous motor", *IEEE Trans. Industrial Electronics*, vol. 57, no. 11, pp. 3822–3829, 2010. [Online]. Available: http://dx.doi.org/10.1109/TIE.2009.2039459

[18] Y.-S. Kung, T.-W. Tsui, N.-H. Shieh, "Design and implementation of a motion controller for XYZ table based on multiprocessor SoPC", in *2009 Chinese Control and Decision Conf.*, Guilin, pp. 258–268.

[19] C. M. Oddo, L. Beccai, N. Vitiello, H. B. Wasling, J. Wessberg, M. C. Carrozza, "A mechatronic platform for human touch studies", *Mechatronics*, vol. 21, no. 3, pp. 604–613, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.mechatronics.2011.02.012

[20] X. Xu, Z. Xiong, J. Wu, X. Zhu, "High-precision time synchronization in real-time Ethernet-based CNC systems", *Int. J. Adv. Manuf. Tech.*, vol. 65, pp. 1157–1170, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00170-012-4246-5

[21] G. Gu, L. Zhu, Z. Xiong, H. Ding, "Design of a distributed multiaxis motion control system using the IEEE-1394 bus", *IEEE Trans. Industrial Electronics*, vol. 57, no. 12, pp. 4209–4218, 2010. [Online]. Available: http://dx.doi.org/10.1109/TIE.2010.2044127

[22] [Online]. Available: E. Monmasson, M. N. Cirstea, "FPGA design methodology for industrial control systems—a review", *IEEE Trans. Industrial Electronics*, vol. 54, no. 5, pp. 1824–1842, 2007. [Online]. Available: http://dx.doi.org/10.1109/TIE.2007.898281

[23] E. Monmasson, L. Idkhajine, M. W. Naouar, "FPGA-based controllers", *IEEE Industrial Electronics Magazine*, vol. 5, pp. 14–26, 2011. [Online]. Available: http://dx.doi.org/10.1109/MIE.2011.940250

[24] X. Shao, D. Sun, "Development of a new robot controller architecture with FPGA-based IC design for improved high-speed performance", *IEEE Trans. Industrial Informatics*, vol. 3, no. 4, pp. 312–321, 2007. [Online]. Available: http://dx.doi.org/10.1109/TII.2007.912360

[25] J. U. Cho, Q. N. Le, J. W. Jeon, "An FPGA-based multiple-axis motion control chip", *IEEE Trans. Industrial Electronics*, vol. 56, no. 3, pp. 856–870, 2009. [Online]. Available: http://dx.doi.org/10.1109/TIE.2008.2004671

[26] M. Martinez-Prado, A. Franco-Gasca, G. Herrera-Ruiz, O. Soto-Dorantes, "Multi-axis motion controller for robotic applications implemented on an FPGA", *Int. J. Adv. Manuf. Tech.*, pp. 1–10, 2012.

[27] A. Astarloa, J. Lazaro, U. Bidarte, J. Jimenez, A. Zuloaga, "FPGA technology for multi-axis control systems", *Mechatronics*, vol. 19, no. 2, pp. 258–268, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.mechatronics.2008.09.001

[28] Y. Chen, H. Wei, K. Sun, T. Wang, Y. Zou, "System-on-chip (SOC) design for CNC system", in *2009 IEEE Int. Symp. Industrial Electronics*, Seoul, pp. 690–693.

[29] Y.-S. Kung, C.-T. Hsu, H.-H. Chou, T.-W. Tsui, "FPGA-realization of a motion control IC for wafer-handling robot", in *8th IEEE Int. Conf. Industrial Informatics*, Osaka, 2010, pp. 493–498.

[30] K. J. Astrom, L. Rundqwist, "Integrator windup and how to avoid it", in *American Control Conf.*, Pittsburgh, 1989, pp. 1693–1698.

[31] J.-W. Choi, S.-C. Lee, "Antiwindup strategy for PI-type speed controller", *IEEE Trans. Industrial Electronics*, vol. 56, no. 6, pp. 2039–2046, 2009. [Online]. Available: http://dx.doi.org/10.1109/TIE.2009.2016514

[32] Y.-S. Kung, M.-H. Tsai, "FPGA-based speed control IC for PMSM drive with adaptive fuzzy control", *IEEE Trans. Power Electronics*, vol. 22, no. 6, pp. 2476–2486, 2007. [Online]. Available: http://dx.doi.org/10.1109/TPEL.2007.909185

[33] *Zynq-7000 All Programmable SoC Overview*, Xilinx Corporation, San Jose, CA, 2013.