ISSN 1392 – 1215 ELECTRONICS AND ELECTRICAL ENGINEERING 2012. No. 7(123)
ELEKTRONIKA IR ELEKTROTECHNIKA

MICROELECTRONICS
MIKROELEKTRONIKA

Hardware Approach of Two Way Conversion of Floating Point to Fixed Point for Current *dq* PI Controller of FOC PMSM Drive

Mohd. Marufuzzaman, M. B. I. Reaz, M. A. M. Ali, L. F. Rahman
Department of Electrical, Electronic and Systems Engineering, Universiti Kebangsaan Malaysia, Bangi, 43600 UKM, Selangor, Malaysia, phone: +603-89216316, e-mail: mohd.marufuzzaman@gmail.com

crossref http://dx.doi.org/10.5755/j01.eee.123.7.2382

Introduction

Among all of the various drive systems, a three-phase Permanent Magnet Synchronous Motor (PMSM) is widely used for accurate speed and torque control, greater efficiency, and superior torque to inertia ratio and high power density. In order to achieve the desired performances of PMSMs, direct control of stator currents is needed. Nevertheless, it is quite unattainable due to the strong coupling and nonlinear natures of the AC motors. Hence, to realize the decoupling of relevant variables, Field-Oriented-Control (FOC) algorithm is introduced [1, 2]. Due to the rapid progress in power electronics, computer and microelectronics, FOC has been used wider and wider in high performance AC drives in the latest twenty years. In FOC PMSM, the *dq*-axis current control plays an important role of determining the overall system performance [3]. Again, due to nonlinear coupling among its winding current and the rotor speed, accurate speed control of a PMSM drive becomes a complex issue. Due to magnetic saturation of the rotor core, the nonlinearity present in the electromagnetic developed torque made it more difficult to control. Therefore, the intelligent controllers claim meticulous consideration for high performance PMSM drive systems [4]. There are several controllers can be employed. Among them Proportional and Integral (PI) controller is the most suitable one to eliminate steady state error [5]. By using PI controller, exact *dq* axis reactance parameters can be obtained. Moreover, it is very sensitive to step change of command speed, parameter variations and load disturbances. Relatively simple implementation made PI controller most widely used for PMSM. Hence, real time self-automated intelligent hardware implementation of PI controller as well as FOC is desired [6].

PI controller consists of two parameters; proportional gain K_p and integral gain K_i. In order to determine its performances; K_p and K_i should be tuned properly. Different algorithms are applied to determine PI controller parameter for FOC PMSMS drive [7]. In most of the cases, K_p and K_i are less than one. Hence, for good tuning in PI controller, K_p and K_i need to be floating point numbers. Floating-point numbers can be represented in Field Programmable Gate Array (FPGA) by IEEE 754 double precision floating-point number formats. However, it requires 64bits. In addition, any arithmetic operation can be done only with another IEEE754 standard numbers. All the operands need to be 64bits. Thus, it needs more pins in FPGA that make the circuits slower and more complex.

Several researches have been done in order to overcome this issue [8]. The most convincing method is to convert the fixed-point inputs into IEEE 754 floating point numbers first. After completing the floating-point calculation, the result will be converted into fixed-point numbers for output. There are many conversion techniques [9, 10]. None of them is for 16-bit fixed-point numbers.

Hardware implementation of a FOC-PMSM controller is an essential perquisite for precise servo drives operation that will replace the expensive software and firmware modules with cheap and speedy single chip controller. The FPGA offers a potential alternative to speed up the hardware realization [11, 12]. They commonly have a large amount of resources (logic elements) available for system development and work at clock speeds up to 1 GHz. Unlike Application Specific Integrated Circuit (ASIC), FPGA is reconfigurable, that is, their internal structure is only partially fixed at fabrication, leaving the wiring of the internal logic to the application designer for the intended task. FPGA allows control over parallelism in resource utilization and also the measurement of resource utilization and power consumption. From the perspective of computer-aided design, FPGA comes with the merits of lower cost, higher density, and shorter design cycle. It comprises a wide variety of building blocks. Each block consists of programmable look-up table and storage registers, where interconnections among these blocks are programmed through the hardware description language. This programmability and simplicity of FPGA made it favourable for prototyping digital system. FPGA allows the users to easily and inexpensively realize their own logic networks in hardware. FPGA also allows modifying the algorithm easily and the design time frame for the hardware becomes shorter by using FPGA, either for prototyping or for commercial deployment.

This paper proposes hardware implementation of a simple conversion method for converting 16-bit fixed-point numbers to 64-bit IEEE 754 standard number and vice versa in Cyclone II from Altera, where, current *dq* PI controller uses 16-bit I/O bus. Hence, this conversion method is highly needed for precise current *dq* measurement.

Methodology

The d-q model has been developed on rotor reference frame as shown in Fig. 1. At any time *t*, the rotating rotor d-axis makes and angle θ_r with the fixed stator phase axis and rotating stator mmf makes an angle α with the rotor d-axis. Stator mmf rotates at the same speed as that of the rotor.

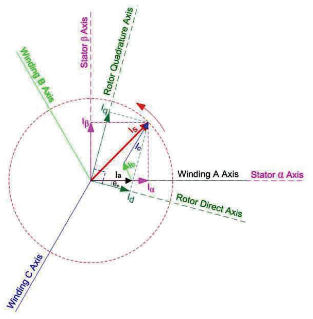


Fig. 1. d-q reference frame in FOC PMSM drive

Overall process flow of current *dq* PI controller is shown in Fig. 2. Initially input feedback current (I_d/I_q) is subtracted from the input reference current (I_{dref}/I_{qref}) to generate the error. The error signal then converted to IEEE 754 floating point number standard and passed to the PI controller. Finally, stator voltage (V_{sd}, V_{sq}) signals are calculated applying (1)

$$\mathrm{V_{outd/outq}} = (\mathrm{K_p} * \mathrm{ERR_{df/qf}}) + \left(\mathrm{K_i} * \sum \mathrm{ERR}_{\frac{\mathrm{df}}{\mathrm{qf}}} * \mathrm{t}\right), \quad (1)$$

where K_p, K_i is the proportional and integral gain that depend on the system. There is a saturation limit of V_{outd}/V_{outq} which is done after V_{outd}/V_{outq} is again converted into 16bit fixed point numbers.

This section is discussing briefly about the double precision floating-point format. A real number N is represented in following way

$$N = (-1)^s . \beta^e . S(1), \quad (2)$$

where radix is β, sign is s, exponent is e and the significant S. The IEEE standard specifies that double precision floating-point numbers are consist of 64 bits, i.e. a sign bit (bit 63), 11 bits for the exponent E (bits 62 down to 52) and 52 bits for the fraction f (bits 51 to 0). E is an unsigned biased number. The true exponent e is determined as $e = E - E_{bias}$ where $E_{bias} = 1023$. Numbers in the range (0, 1) is represented by the fraction f. The significant S is specified as $S = 1.f$ and the range of S is (1, 2). The leading 1 of the significant is known as the "hidden bit". Normalized representation means the MSB of the significant is 1 and is followed by the radix point. For double precision numbers, the range of the unbiased exponent e is [–1022, 1023], which translates to a range of only [1, 2046] for the biased exponent E. The values $E = 0$ and $E = 2047$ are reserved for particular numbers. The number zero is represented with $E = 0$, and $f = 0$. The hidden significant bit is also 0. Zero has a positive or negative sign like normal numbers. When $E = 0$ and $f \neq 0$ then the number has $e = -1022$ and a significant $S = 0.f$. The sign s of normal numbers is determined from the sign bit (bit 63). These numbers are referred as "denormalized". This part of standard is not implemented in hardware for additional complexity and cost. Another special number infinity ∞ is represented by $E = 2047$ and $f = 0$. The sign bit represents the sign of infinity as well as for normal numbers. Lastly, another special representation of "Not a number (NaN)" is defined as $E = 2047$, and a fraction $f \neq 0$. Usually NaN is produced by operations like *0/0* and $0{\cdot}\infty$. In addition the IEEE standard characterizes four rounding mode. The default mode is known as "round to nearest even (RNE)".

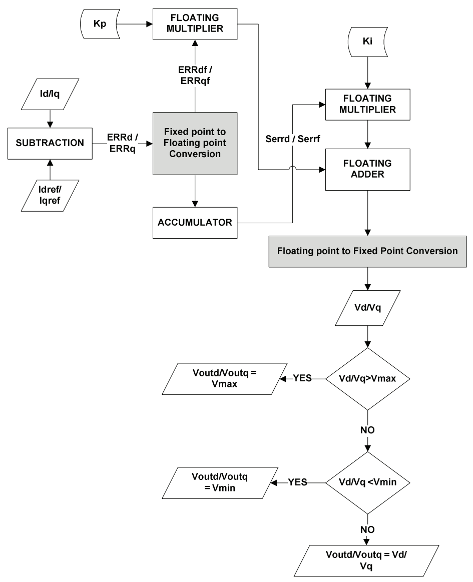


Fig. 2. Process flow chart of current *dq* PI controller

Hardware implementation of current *dq* PI controller of FOC PMSM drive is extremely depending on the success of the conversion techniques. Initially, we have designed three modules which solely focus on normalization of 16bit fixed point numbers to IEEE 754 floating point numbers, floating point calculation specially addition and multiplication, and denormalization of the result into 16bit fixed point numbers. These sub-modules are defined as *fxp_to_flp*, *flp_operation* and *flp_to_fxp*. After successful design and verification of individual module, we have integrated all three into one main module to validate the target result. There are two inputs designated for the main module. One input is 16-bit fixed-point number and the other is 64-bit IEEE 754 standard floating point number. The output of the main module is 16-bit fixed-point numbers. A test bench also designed for this main module in order to validate the results.

Hardware implementation

Initially the above-mentioned sub-modules of the system are translated to hardware description language (HDL). The HDL used in this work is Verilog HDL, as it is easy to understand and very similar to the most popular programming language, C. After preparing the Verilog design file, the Register Transfer Logic (RTL) code is generated. A testbench is also designed in Verilog for HDL-simulation. After HDL-simulation and static-logic verification, the HDL code is synthesized. If it needs any synthesis tweaks, it is done in RTL code. Finally, the FPGA is realized.

Four multiplexer and five registers are used for integrating all the three sub modules. Enable pin and clock is used for synchronizing the intermediate inputs and outputs. One of the inputs is defined as IEEE 754 double precision floating-point number and the other is 16-bit fixed-point number. After converting the 16-bit number into 64-bit number format it is then pass to second sub module, which can perform arithmetic operations on two floating-point numbers. Finally, the output is converted to 16-bit fixed-point numbers.

Table 1 shows the overall summery of the design specification of this method. This hardware implementation can be run up to 20MHz clock cycles. The proposed method has two inputs and one output. One input signal is 16bit and the other is 64bit while the output signal is 16 bit.

Table 1. Summery report of Hardware Implementation

Family	Cyclone II
Device	EP2C8Q208C7
Tool	Quartus II 9.1 Web Edition, ModelSim SE 6.3a

Total Logic Elements Used	3371(for Addition)
	4045(for Multiplication)

Total I/O Pin	98
Clock	20MHz

Results and discussion

In order to test the HDL code, some sample inputs have been fetched through testbench. The simulation is done in ModelSim SE 6.3 simulator. Fig. 3 and Fig. 4 summarizes the overall simulating result of the proposed method. This method has taken clock, reset signals as input. These signals are used for synchronizing. Two different format numbers are used as input too. *input_a* is 16bit fixed point format of decimal *4* i.e. *004h* and *input_b* is 64bit IEEE754 standard floating point number format of decimal *2.5* i.e. *4004000000000000h*. The numbers shown here is in hexadecimal for easy analysis. After doing the addition operation, the result should be decimal *6.5*. After rounding, the output shown in Fig. 3 is *0006h*. Similarly, after applying the multiplication operation the result should be (4 * 2.5 =10) i.e. decimal *10*. The output shown in Fig. 4 is also *000ah.*

Fig. 5 and Fig. 6 represents the intermediate states of this conversion method. The input set used here is different. *input_a* is decimal *6* and *input_b* is decimal *2.1*. By examining the waveform in Fig. 5, it is shown that *input_a* first converted to floating point number format and it is *4018000000000000h*, which is the IEEE 754 double precision floating-point format of decimal value *6*. *input_b* is already in IEEE 754 double precision floating-point format, which is *4000cccccccccccdh*. In second module, the converted *input_a* will add with *input_b*. the intermediate addition result is stored in *floating_out* register i.e. *4020333333333333h.*

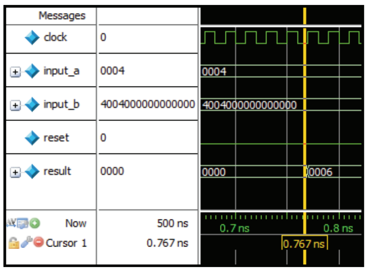


Fig. 3. Waveform generated by ModelSim SE 6.3a. applying Addition operations

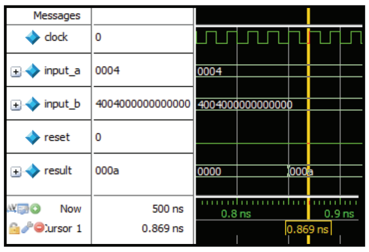
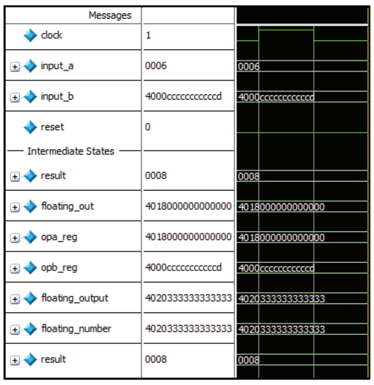
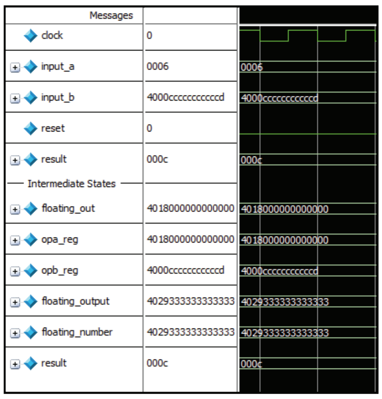


Fig. 4. Waveform generated by ModelSim SE 6.3a applying Multiplication operation



Messages
clock 1
input_a 0006 0006
input_b 4000cccccccccccd 4000cccccccccccd
reset 0
Intermediate States
result 0008 0008
floating_out 4018000000000000 4018000000000000
opa_reg 4018000000000000 4018000000000000
opb_reg 4000cccccccccccd 4000cccccccccccd
floating_output 4020333333333333 4020333333333333
floating_number 4020333333333333 4020333333333333
result 0008 0008

Fig. 5. Waveform of intermediate states generated by ModelSim SE 6.3a applying Addition operations



Messages
clock 0
input_a 0006 0006
input_b 4000cccccccccccd 4000cccccccccccd
reset 0
result 000c 000c
Intermediate States
floating_out 4018000000000000 4018000000000000
opa_reg 4018000000000000 4018000000000000
opb_reg 4000cccccccccccd 4000cccccccccccd
floating_output 4029333333333333 4029333333333333
floating_number 4029333333333333 4029333333333333
result 000c 000c

Fig. 6. Waveform of intermediate states generated by ModelSim SE 6.3a applying Multiplication operations

It is the IEEE 754 double precision floating point format of decimal *8.1. floating_out* is then passed to final module. Lastly, it converted back to fixed-point number format (*0008h*). Similarly, Fig. 6 illustrates the multiplication operations. The numerical calculation of the addition is actually decimal *8.1* and for multiplication, it is decimal *12.6.* After rounding, the result of this method is actually *8* and *12*.

Comparing with the numerical calculation, we found that the simulation producing exactly same values. Current *dq* PI controller needs only addition and multiplication for its calculation. Therefore, division operation is not necessary in this method.

Conclusions

This research implemented a simple conversion method for converting 16bit fixed point numbers to 64bit IEEE 754 double precision floating point number and vice versa. This proposed method has successfully achieved the desired result. Current *dq* PI controller uses 16-bit I/O bus. Hence, this conversion method is highly needed for precise current *dq* measurement, which is the key element of the overall FOC-PMSM drive system.

References

1. **Leonhard W.** Control of Electrical Drives. – Berlin: Springer–Verlag, 2001. – 460 p. DOI: 10.1007/978-3-642-56649-3.
2. **Vas P.** Vector Control of AC Machines. – NY, USA:Oxford University Press, 1990. – 332 p.
3. **Konghirun M., Xu L.** A dq–axis current control technique for fast transient response in vector controlled drive of permanent magnet synchronous motor // Proceedings of the 4th International Power Electronics and Motion Control Conference (IPEMC). – Xi'an, China, 2004. – No. 3. – P. 1316–1320.
4. **Dalvand F., Mardaneh M., Milimonfared J.** Development of genetic–PI based controller for interior permanent magnet synchronous motor drive over wide speed range // Proceedings of The 3rd IET International Conference on Power Electronics, Machines and Drives (PEMD). – Dublin, Ireland, 2006. – P. 291–295.
5. **Wong L. K., Leung F. H. F., Tam P. K. S.** Combination of sliding mode controller and PI controller using fuzzy logic controller // Proceedings of The IEEE International Conference on Fuzzy Systems. – Anchorage, USA, 1998. – No. 1. – P. 296–301.
6. **Marufuzzaman M., Reaz M. B. I., Rahman M. S., Mohd. A. M. A.** Hardware prototyping of an intelligent current dq PI controller for FOC PMSM drive // Proceedings of the International Conference on Electrical and Computer Engineering (ICECE'10). – Dhaka, Bangladesh, 2010. – P. 86–88.
7. **Pant M., Thangaraj R., Abraham A.** Optimal Tuning of PI Speed Controller Using Nature Inspired Heuristics // Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications. Washington DC, USA, 2008. – No. 3. – P. 420–425.
8. **Yankai X., Kai S., Shan J., Xiaoliang W.** FPGA Implementation of a Best–Precision Fixed–Point Digital PID Controller // Proceedings of the International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). – Hunan, China, 2009. – No. 3. – P. 384–387.
9. **Banerjee P., Bagchi D., Haldar M., Nayak A., Kim V., Uribe R.** Automatic conversion of floating point MATLAB programs into fixed point FPGA based hardware design // Proceedings of the 11th Annual IEEE Symposium on Field–Programmable Custom Computing Machines (FCCM). – Napa, California,USA, 2003. – P. 263–264.
10. **Daniel M. D. C., Charot F., Sentieys O.** Automatic floating–point to fixed–point conversion for DSP code generation // Proceedings of the international conference on Compilers, architecture, and synthesis for embedded systems. – Grenoble, France, 2002. – P. 270–276.
11. **Coussy P., Gajski D.D., Meredith M., Takach A.** An Introduction to High–Level Synthesis // IEEE Design & Test of Computers. – IEEE, 2009. – No. 26(4). – P. 8–17.
12. **Reaz M. B. I., Choong F., Sulaiman M. S., Mohd–Yasin F.** Prototyping of Wavelet Transform, Artificial Neural Network and Fuzzy Logic for Power Quality Disturbance Classifier // Journal of Electric Power Components and Systems. – Taylor & Francis, 2007. – No. 35(1). – P. 1–17.

Received 2011 11 28
Accepted after revision 2012 02 10

Mohd. 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 // Electronics and Electrical Engineering. – Kaunas: Technologija, 2012. – No. 7(123). – P. 79–82.

Real time self-automated hardware implementation of current *dq* PI controller for FOC PMSM is desired. Nevertheless, the controller needs some floating-point calculations, which necessitate expensive computing elements. Moreover, additional bits for implementing arithmetic operations are required for floating point calculation. This research proposed a hardware approach of 64bit floating point to 16-bit fixed-point conversion and vice versa. This conversion method will reduce the bit requirements. Any application which is required 16-bit floating-point calculation can be used this two-way conversion technique. In addition, current *dq* PI controller uses 16-bit I/O bus. Thus, hardware approach of this two-way conversion is a prerequisite for FPGA realization of current *dq* PI Controller for FOC-PMSM drive. The outcome of this research is a single chip 16bit to 64bit IEEE 754 standard number format converter for current *dq* PI controller of FOC-PMSM drive. Ill. 11, bibl. 12, tabl. 1 (in English; abstracts in English and Lithuanian).

Mohd. Marufuzzaman, M. B. I. Reaz, M. A. M. Ali, L. F. Rahman. Aparatinis slankiojo kablelio konvertavimo į fiksuotą kablelį sprendimas, skirtas FOC-PMSM variklio srovės *dq* PI valdikliui // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2012. – Nr. 7(123). – P. 79–82.

Srovės *dq* PI valdiklis skirtas FOC-PMSM varikliui, atlieka keletą slankiojo kablelio skaičiavimų, todėl reikia daug brangių skaičiavimo elementų. Be to, atliekant slankiojo kablelio skaičiavimus reikalingi papildomi bitai aritmetinėms operacijoms tobulinti. Pasiūlytas aparatinis sprendimas 64 bitų slankiajam kableliui konvertuoti į 16 bitų fiksuotą kablelį ir atvirkščiai. Konvertavimo metodas leidžia sumažinti reikiamų bitų skaičių. Kiekvienai sistemai, kuriai reikalingi 16 bitų slankiojo kablelio skaičiavimai, galima taikyti šį metodą. Be to, srovės *dq* PI valdiklis naudoja 16 bitų I/O magistralę. Todėl toks sprendimas yra būtina sąlyga srovės *dq* PI valdiklio, skirto FOC-PMSM varikliui, FPGA realizacijai. Il. 11, bibl. 12, lent. 1 (anglų kalba; santraukos anglų ir lietuvių k.).