

INVERSE PARK TRANSFORMATION USING CORDIC AND PHASE-LOCKED LOOP

ANIS SHAHIDA NIZA MOKHTAR, MAMUN BIN IBNE REAZ, MOHAMMAD
MARUFUZZAMAN, MOHAMMAD ALAUDDIN MOHAMMAD ALI

Key words: Coordinate rotation digital computer (CORDIC), Field oriented control (FOC), Inverse Park transformation, Phase-locked loop (PLL).

Faster and precise solution of inverse Park transformation is required to reach the optimal efficiency and steadiness of the servo drive. The main problem in presenting such solution lies in completing the transformation within a limited clock cycles as well as execution time. One of the options could be to implement the overall transformation algorithm into field programmable gate array (FPGA) for reducing the execution time significantly. This research presents a fully integrated solution of inverse Park transformation by incorporating Coordinate Rotation Digital Computer (CORDIC), Arithmetic and Phase-locked Loop (PLL) modules. The result shows that the proposed FPGA performance is required only 68ns of execution time for operating frequency of 30 MHz and accuracy of 99.9%, which is the lowest computational cycle for the era.

1. INTRODUCTION

In this recent era, the number of drive systems offered to designer has increased tremendously. Permanent magnet synchronous motor (PMSM) system becomes one of the most useful drive options for a wide range of applications due to their high power density, high efficiency, robust operation, lower maintenance and high mechanical reliability [1]. The vector control of the motor which is known as field oriented control (FOC) of the PMSM drive is employed to achieve high performance [2]. FOC has been proven to be a very powerful control method for motor control. It provides smooth motion and efficient operation in any speed ranges and when implemented on the induction motors, it has become an extremely important drive structure [3]. Improvements of the system will assist in reducing the developing cost and certainly will enhance the robustness of the system.

The important transformations in FOC are Park and Clarke transformation and their inverses. Park transformation is a well known mathematical

University Kebangsaan, Malaysia, Department of Electrical, Electronic and Systems Engineering, 43600 Bangi, Selangor, Malaysia; E-mail: mamun.reaz@gmail.com

transformation in synchronous machine analysis to convert two-phase stationary frame into two-phase rotating frames. The α - β frame is defined as the stationary reference frame, and the d - q frame is defined as the rotating reference frame [4].

At the output of PI iteration, there are two voltages used by the inverse Park transformation component vectors in the rotating d - q axis. The two-phase d - q is fed to a vector rotation block where it is rotated by an angle θ to follow the frame α - β . The rotation over an angle θ is given by equation (1) and (2).

$$V_\alpha = V_d \cos \theta - V_q \sin \theta, \quad (1)$$

$$V_\beta = V_d \sin \theta + V_q \cos \theta. \quad (2)$$

CORDIC proposed by Volder is used to realize this transformation [5]. It is an attractive algorithm due to the simplicity of its hardware structure, speed and resources [6]. The transformation of FOC needs very quick mathematical calculations, thus CORDIC algorithm was selected for solving FOC tasks.

Ghariani *et al.* [7] used a modified CORDIC to implement the Park transformation with a frequency of 8 MHz in Xilinx FPGA. The execution time of their design was 125 ns even solely for the CORDIC algorithm execution. Rachid *et al.* [8] used two different frequencies which were 100 MHz and 8 MHz where both of them need 6 clock cycles to produce the output. Furthermore, their research used a rotor flux estimator for calculation of $\cos \theta$ and $\sin \theta$. The calculated angle is then used in Park transformation and inverse Park transformation. So the transformation needs 6 clock cycles excluding the calculation of $\cos \theta$ and $\sin \theta$. Apart from that, a research from Ying-Shieh Kung *et al.* [9] shows that inverse Park sub-module could be accomplished in 5 steps. Finite state machine (FSM) method was proposed which interprets each step equals to 1 clock cycle. The operation of each step could be completed within 40 ns at 25 MHz, thus, execution time for whole inverse Park transformation is 0.2 μ s. But, the values of $\cos \theta$ and $\sin \theta$ were pre-calculated in the look-up table. Ricardo de Castro *et al.* [10] in their research reported that the combination of Park transformation and Clarke transformation produces 30 clock cycles with a maximum frequency of 78 MHz. This research used a CORDIC algorithm that available from Xilinx for calculating the module and angle of the vector. This CORDIC implementation takes 20 clock cycles to complete.

Infineon Technologies [11] applied FOC in their motor product. It is reported that inverse Park transformation module needed 31 clock cycles (775 ns) using frequency of 40 MHz but it is also stated that this information do not cover the complete control algorithms. The Freescale Semiconductor [12] on the other hand requires 45 clock cycles. No other description given on this transformation regarding the calculation of $\cos \theta$ and $\sin \theta$. Texas instrument [13] also provides a data on the product of TMS320C2XX which gives the performance information

where Linear Interpolation and single function with look-up table is used for calculating $\sin \theta$ and $\cos \theta$. It shows that, with 20 MHz, the inverse Park module with $\sin \theta$ and $\cos \theta$ calculation takes 101 cycles with 5.05 μs execution time in the assembly calling functions while the fully C compatible functions requires 126 cycles with 6.03 μs execution time.

Table 1 shows the summary of other research. This summary table compare in term of frequency used, the execution time and the clock cycle. The aim of the research is to present a method of getting a very quick computation in extremely short time in order to get an optimal efficiency and stability of the motor through the FOC in PMSM Motor. The completion time for the whole transformation is reduced to 4 clock cycles.

Table 1

Summary of other research

	Frequency	Execution time	Clock cycle
Ghariani <i>et al.</i> [7]	8 MHz	–	–
Rachid <i>et al.</i> [8]	100 MHz and 8 MHz	–	6
Ying-Shieh Kung <i>et al.</i> [9]	25 Mhz	0.2 μs	5
Ricardo de Castro <i>et al.</i> [10]	78 MHz	–	30
Infineon Tech [11]	40 MHz	775 ns	31
Freescall Semicond. [12]	–	–	45
Texas Instrument [13]	20 MHz	5.05 μs 6.03 μs	101 126

2. ARCHITECTURE OF INVERSE PARK TRANSFORMATION MODULE

The whole architecture of inverse Park transformation of this research is shown in Fig. 1. The architecture consists of three modules; PLL, CORDIC and ARITHMETIC. PLL is used for frequency controls where it generates an output signal based on the reference input signal. The clock output of PLL is used as clock input for all other modules. The CORDIC module that used in this research is an existing CORDIC which originated by Volder [5]. This module is used to perform the calculation of $\cos \theta$ and $\sin \theta$. This is the most important part in order to get an accurate result as it will affects the rest of the system. The ARITHMETIC module that comes after the CORDIC module is used to complete the transformation. This ARITHMETIC module consists of adder, subtractor, multiplexer and shifter. A well blend of all modules produced a complete inverse Park transformation with precise output as shown in Fig. 1.

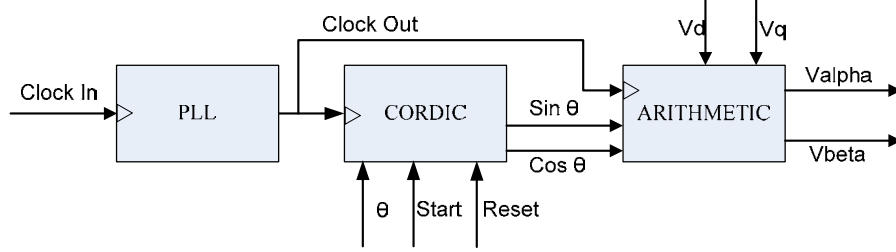


Fig. 1 – Overall block diagram of inverse Park transformation.

2.1. PHASE-LOCKED LOOP

PLL is a closed-loop frequency-control system based on the phase difference between the input and feedback clock signal of a controlled oscillator which is used on top of all modules. Phase Frequency Detector (PFD) is used in PLL circuit to align the rising edge of the reference input clock to a feedback clock [14]. The other input of PFD is from the output of a divide by N counter. Divide by N counter is used to increase the VCO frequency above the input reference frequency. The output of the phase detector is a voltage proportional to the phase difference between the two inputs. This signal is applied to the loop filter. The filtered signal controls the Voltage Controlled Oscillator (VCO). Based on the control voltage of the filter, the VCO oscillates at upper or lower frequency, which affects the phase and frequency of the feedback clock. VCO frequency (F_{out}) is equal to N times the input reference clock (F_{IN}).

2.2. CORDIC

The CORDIC algorithm normally offers the smartest solution to the problem, and extremely useful in implementation of systems. CORDIC algorithm uses only addition and shift operation which offer the opportunity to calculate the desired functions in a simple and efficient way [15]. In this paper, the rotation mode by [5] is used where the input vector is rotated by a specific angle to calculate the sine and cosine function. The rotation by an angle θ is implemented as an iterative process. The CORDIC algorithm is derived from general rotation transformation as shown in (3) and (4),

$$x' = x \cdot \cos \theta - y \sin \theta. \quad (3)$$

$$y' = y \cdot \cos \theta + x \sin \theta, \quad (4)$$

where x' and y' are the vector components after rotation through an angle θ , while x and y are the original vector components. Both equations can be arranged such that

$$x' = \cos \theta [x - y \cdot \tan \theta], \quad (5)$$

$$y' = \cos \theta [y + x \tan \theta]. \quad (6)$$

To avoid the multiplication of $\tan \theta$ as in (5) and (6), the rotation angles are restricted into a set of pre-defined angles, so that $\tan \theta = 2^{-i}$, $i = 0, 1, 2, \dots, n-1$ which can be implemented by just bit shift operations. These angles refer to the rotation of each steps which implemented as an iterative process as shown in Table 2.

Table 2
Angle used in CORDIC algorithm

i	$\tan \theta = 2^{-i}$	θ
0	$1 = 2^0$	45.000°
1	$0.5 = 2^{-1}$	26.565°
2	$0.25 = 2^{-2}$	14.036°
3	$0.125 = 2^{-3}$	7.125°
4	$0.0625 = 2^{-4}$	3.576°
5	$0.03125 = 2^{-5}$	1.789°
6	$0.015625 = 2^{-6}$	0.895°
7	$0.0078125 = 2^{-7}$	0.448°

These angle values are stored in relatively lookup table even for a high degree of accuracy where the accuracy of the result increases with the number of step. Since the θ lies between $+\pi/2$ and $-\pi/2$, then the trigonometric identity comply $\cos(\theta) = \cos(-\theta)$, which the term $\cos \theta$ in equation (5) and (6) becomes a constant. The constant depends only on the number of steps of rotation to approximate the cos and sin values as in equation (7)

$$\cos \theta = \cos(\tan^{-1} 2^{-i}). \quad (7)$$

The iterative rotation can now be expressed as [16]:

$$x_{i+1} = K_i [x_i - y_i d_i - 2^{-i}], \quad (8)$$

$$y_{i+1} = K_i [y_i - x_i d_i - 2^{-i}], \quad (9)$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}), \quad (10)$$

where $K_i = \cos(\tan^{-1} 2^{-i}) = 1/\sqrt{1+2^{-2i}}$ and $d_i = -1$ if $z_i < 0$, $+1$ otherwise.

The product of K_i 's represents the K factor.

$$K = \prod_{i=0}^{n-1} K_i . \quad (11)$$

This K factor can be calculated in advance and applied elsewhere in the system or treated as part of a system processing gain. The product approaches 0.6073 as the number of iterations goes to infinity. Therefore, the rotation algorithm has a gain, $A_n \approx 1.647$. this gain value is pre-calculated constant and used as initial value of x . Initial value of y is zero ($y_0 = 0$) because the starting point of rotation is at $y = 0$ and initial value of z is the angle in radian ($z_0 = \theta$). At each iteration, the values of x and y are updated depending on the sign of x and y . The values of z are updated based on the sign of residual angle. At the end of iterations, the resulting values for x and y are as in equation (12) and (13).

$$X_n = A_n [x_0 \cos z_0 - y_0 \sin z_0], \quad (12)$$

$$Y_n = A_n [y_0 \cos z_0 + x_0 \sin z_0]. \quad (13)$$

At the final iteration, $X_n = \cos \theta$ and $Y_n = \sin \theta$. 18 iterations is used in this work to bear with the fixed point format used.

2.3. ARITHMETIC

ARITHMETIC module is used to complete the transformation module after getting result from CORDIC module. In this research, fixed point is used in all modules since fixed point is same as integer arithmetic, fast and mostly used in FPGAs. Fixed point format Q1.14 is used for all inputs. The input angle θ is represented in radian and it ranges between -90° and 90° . The mirror properties are used for other angle values. Four multipliers are used to multiply 16bits signed input V_d and V_q with the 20 bits $\cos \theta$ and $\sin \theta$ which produced 35bits output, then adder and subtractor is used to produce V_{α} and V_{β} as in equation (1) and (2).

Inverse Park transformation plays an important role in implementing FOC of PMSM motor. The proposed architecture of inverse Park transformation is designed in such a way that it gives accurate result with very short execution time. This will influence the overall FOC performance for faster and accurate operations.

3. RESULT AND DISCUSSION

The architecture has been developed using Verilog hardware description language in Quartus II Altera environment. The simulation outcome were obtained

using ModelSim simulator. All modules as described in Fig. 1 are developed. The first left module is the PLL module. For this research, we employ the PLL from Megawizard function in Altera [14]. The goal of this research is to get 4 clock cycles for a complete inverse Park transformation by using PLL, CORDIC and Arithmetic. PLL module uses input frequency 24 MHz, 50 % duty clock cycle and it gives out output frequency of 144 MHz. The output of this PLL is used as clock input for the others modules. The calculation of $\cos \theta$ and $\sin \theta$ is done before the transformation commence where the CORDIC is applied. This CORDIC module is synchronized with *start* and *reset* signal where the values in the registers are set to zero when the *reset* signal is triggered. The *start* signal is in synchronous with clock, and the calculations begin at the first rising edge of clock when start signal is high. There is 16 bits input θ that goes through this module and it gives 20 bits output of calculated $\sin \theta$ and $\cos \theta$. This 20 bits output of CORDIC is directly sent to the multiplier to multiply with the 16 bits input of V_d and V_q . To follow the requirement and standard, these two values will be shifted right 3 bits to the right by using arithmetic shift operator. Thus, output of the arithmetic module is 32 bits of V_{α} and V_{β} . To maintain the signed values, the empty position in the most significant bit (MSB) is filled with the original MSB.

To verify the module, two different sets of 16bits random data are tested and compared with calculated values (hand calculation) shown in Table 3. They are simulated in ModelSim and are illustrated in Fig. 2. In this module, all inputs are multiplied by 16384 or 2^{14} according to fixed point format. All of these values are represented in signed decimal number. Throughout the process, the output values need to be divided by 2^{29} in order to obtain the original real data output. To calculate the accuracy of this module, equation (14) is implemented.

$$\frac{\text{Simulation Values}}{\text{Calculation Values}} \times 100\%. \quad (14)$$

From Table 3, by using this formula it shows that the accuracy of all data is 99.99 %. After synthesizing and simulating the Verilog code in Quartus and ModelSim, the result is validated by viewing the waveform generated. The number of clock cycle required to compute the output for each input is measured. Figure 2 and 3 show 2 sets of different input tested. The computation of this module is measured from the first rising edge of the clock after the start signal is activated. From these figures, it is clearly shown that the output is produced within 4 cycles. The execution time is about 160 ns for 24 MHz frequency. Figure 2 certainly shows that this module can work in an extremely fast environment.

From the synthesis result of simulation in Quartus II, this module only needs 563 out of 33,216 logic elements which consume only 2 % of total logic elements. Also, only 2 % of total combination function which is 499 / 33 126 is used. 212

logic registers have been used in this design which is less than 1 % of total logic register provided by Cyclone II. Total pin used are 115 out of 475 which is equivalent to 24 %. Only 16 units of embedded multiplier 9-bit element out of 70 are used which only consume 23 % of total 9-bits elements and 1 PLL used in this module. From these results, it shows that this design uses less logic elements which will maximize area efficiency.

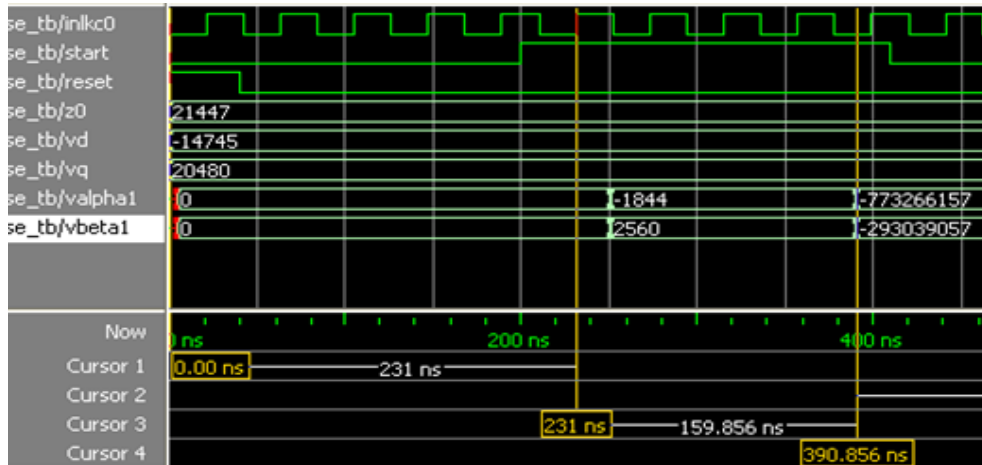


Fig. 2 – First simulation results.

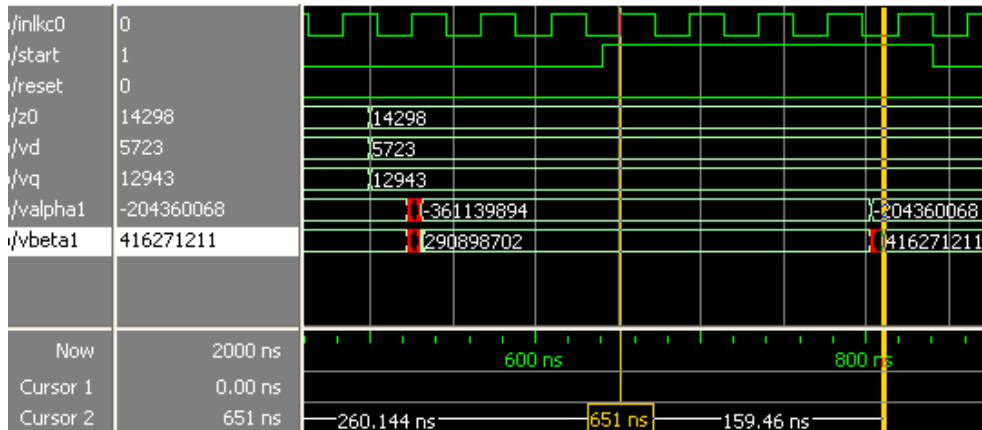


Fig. 3 – Second simulation results.

Table 3

Comparison between simulation and calculated values

Input	Simulation value	Calculation value
$V_d = -0.9$ $V_q = 1.25$ $Z_0 = 75^\circ / 1.308 \text{ rad}$	$V_\alpha = -773266157 / 2^{29}$ $= -1.440320$ $V_\beta = -293039057 / 2^{29}$ $= -0.545827$	$V_\alpha = -0.9 \cos(75) - 1.25 \sin(75)$ $= -1.440344$ $V_\beta = -0.9 \sin(75) + 1.25 \cos(75)$ $= -0.545809$
$V_d = -0.35$ $V_q = 0.79$ $Z_0 = 50^\circ / 0.873 \text{ rad}$	$V_\alpha = -204360068 / 2^{29}$ $= -0.380650$ $V_\beta = 416271211 / 2^{29}$ $= 0.775366$	$V_\alpha = -0.35 \cos(50) - 0.79 \sin(50)$ $= -0.380199$ $V_\beta = -0.35 \sin(50) + 0.79 \cos(50)$ $= 0.775918$

This inverse park transformation module have successfully completed, and in comparison with Table 1, it shows that this work can accomplish the transformation within 4 clock cycles which is the lowest among the others. Besides, this inverse park transformation implements cordic for calculating $\cos \theta$ and $\sin \theta$ as well as the transformation itself whereas others implemented only the transformation. This research also shows the lowest execution time of 160ns compared to other works.

4. CONCLUSIONS

This paper presents a complete inverse Park transformation including the calculation of $\sin \theta$ and $\cos \theta$. We have focused primarily on the existing CORDIC and integrated it with arithmetic module and PLL. After conducting a thorough evaluation and performance analysis, it was found that this type of implementation is at the great interest because it can offer fast computation and optimal efficiency with stability of the motor system. This work is a prominent contribution in this area of research on the motion movement where it is able to complete the calculation within 4 clock cycles. Thus, these fast and precise results manages to prove that the overall motor system and performance of FOC PMSM have been improved in terms of its efficiency and stability.

ACKNOWLEDGEMENTS

This research is funded by a grant from the Ministry of Science, Technology and Innovation of Malaysia (Technofund: TF1008C130).

Received on February 24, 2012

REFERENCES

1. Wenxiang Zhao, K.T Chau, Ming Cheng, Jinghua, Xiaoyo, *Remedial Brushless AC Operation of Fault-Tolerant Doubly Salient Permanent-Magnet Motor Drives*, IEEE Transactions on Industrial Electronics, **57**, pp. 2134-2141, 2010.
2. M. N. Uddin, M. A. Abido, M. A. Rahman, *Development and implementation of a hybrid intelligent controller for interior permanent-magnet synchronous motor drives*, IEEE Transactions on Industry Applications, **40**, 1, pp. 68-76, 2004.
3. S. B. Ozturk, H. A. Toliyat, *Direct torque and indirect flux control of brushless DC motor*, IEEE/ASME Transaction on Mechatronics, **16**, pp. 351-360, 2011.
4. D. Kisek, J. Hang, D. Kang, J. Kim, *Parameter identification of permanent-magnet synchronous motors for sensorless control*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **55**, 2, pp. 132-142, 2010.
5. J. E. Volder, *The Cordic Trigonometric Computing Technique*, IRE Transaction on Electronics Computer, **EC-8**, pp. 330-334, 1959.
6. P. K. Meher, J. Valls, T. B. Juang, K. Sridharan, K. Maharatna, *50 years of CORDIC: Algorithms, Architectures, and Applications*, IEEE Transactions on Circuits and Systems I: Regular Paper, **56**, pp. 1893-1907, 2009.
7. M. Ghariani, N. Masmoudi, M.W. Kharrat, L. Kamoun, *Design and Chip Implementation of Modified CORDIC Algorithm for Sine and Cosine Functions Application: Park transformation*, The Tenth International Conference of Microelectronics, pp.241-244, Monastir, 1998.
8. J.-G. M. Rachid Beguenane, Stephane Simard, Arnaud Tisserand, *Towards the System On-Chip Realization of The Sensorless Vector Controller With The Microsecond-Order Computation Time*, 19th IEEE Canadian Conference on Electrical & Computer Engineering, Ottawa, Canada, 2006.
9. K. Ying-Shieh, T. Ming-Hung, *FPGA-Based Speed Control IC for PMSM Drive with Adaptive Fuzzy Control*, IEEE Transactions on Power Electronics, **22**, pp. 2476- 2486, 2007.
10. R. de Castro, R.E. Araujo, H. Oliveira, *Control in Multi-Motor Electric Vehicle With a FPGA Platform*, Industrial Embedded Systems, IEEE International Symposium, Lausanne, 2009, pp. 219-227.
11. Infineon Technologies, *Application notes of 3-phase motor drive implementation of space vector modulation based on XC164CS/XC167CI*, Edition 2003; <http://www.infineon-technologies.com>
12. Freescale Semiconductor, *Datasheet of Motor Control Library*, in *Inverse Park Transformation*, Edition 2009; <http://www.freescale.com>.
13. Texas Instrument, *Application Reports of Clarke & Park transforms on the TMS320C2XX*, Edition 1997, pp. 46; <http://www.nalanda.nitc.ac.in/industry/appnotes/Texas/motcon/bpra048.pdf>
14. Altera Corporation, *Phase-Locked Loop Megafunction User Guide*, January 2011; Phase-locked Loop Basic. Retrieved Jun 16, 2011 from http://www.altera.com/support/devices/pll_clock/basics/pll-basics.html.
15. A. Mandal, B. K. Kaushik, B. Kumar, R.P. Ararwal, *Design and implementation of CORDIC processor for complex DPLL*, in : *Power Electronics (IICPE)*, India International Conference, India, 2011, pp. 1-5.
16. R. Andraka, *A survey of CORDIC algorithms for FPGA based computers*, ACM Sigda Sixth International Symposium on FPGAs, New York, 1998, pp. 191-200.