Design and implementation of Clarke and Park transforms for electric AC motors developed for FPGAs

Conference Paper · September 2010

CITATIONS
0

READS
2,550

5 authors, including:

Juan Raygoza Panduro
University of Guadalajara
56 PUBLICATIONS **151** CITATIONS
SEE PROFILE

Jorge Rivera Dominguez
91 PUBLICATIONS **653** CITATIONS
SEE PROFILE

Francisco Javier Ibarra Villegas
Grupo Bargo de México
6 PUBLICATIONS **3** CITATIONS
SEE PROFILE

JCRA Jornadas de Computación Reconfigurable y Aplicaciones
Reconfigurable Computing and Applications Conference
7-10 de Septiembre de 2010

Design and implementation of Clarke and Park transforms for electric AC motors developed for FPGAs

Juan J. Raygoza P., Susana Ortega C., Jorge Rivera D., Francisco J. Ibarra V, A de la Mora.
Departamento de Electrónica y Computación, CUCEI
Universidad de Guadalajara
Guadalajara, Jal., México
{juan.raygoza, susana.ortega, jorge.rivera, demora}@cucei.udg.mx, javier.villegas@red.cucei.udg.mx

Abstract

In this work, the direct and inverse Clarke and Park transforms (modules) that are necessary for the control of AC electric motors are designed and implemented in FPGAs. These modules are described in VHDL using mathematical functions. The mathematical functions are designed using a tool developed in Java, which converts such functions into ready to implement VHDL described files. The designed modules will be the first part of a complete library of modules for the control of AC electric machine based on the Field Oriented Control (FOC) technique. The work shows the architecture of the modules developed for the FPGA such as functionality, utilization and delay results when they're implemented in a Xilinx® Virtex II.

1. Introduction

Electric AC motors are extensively used in industrial applications, due to their characteristics as robustness, low maintenance and high performances. From the control point of view, AC motors represents a complex multi-variable nonlinear problem and constitutes one of the important areas of application for nonlinear control theory. The pioneering work of Blashke [1], the field oriented control (FOC), has become a classical technique for AC motor control which involves nonlinear state transformations like the well known Clarke and Park transforms and linearizing feedback for asymptotic decoupling of the rotor velocity and flux modulus.

On the other hand, with the advent of digital technology and its widespread use in the global market, the computer-based implementation of advanced control schemes in industry was revolutionized, resulting in a growing research activity in the area of control techniques using FPGAs. Recent FPGAs popularity have rendered considerable merit to digital computer-control systems, exhibiting relatively low operational cost, flexibility for implementation, high speed data processing, simple and functional interactive communication among several control loops [1]. Consequently, a high motivation is offered for a further development of direct digital process control methods in FPGAs. Therefore, it is interesting to emerge in the field of control systems using FPGAs. When implementing control algorithms in FPGA reconfigurable devices, an important aspect is the design of the arithmetic blocks [1]. A well known method for implementing arithmetic operations in FPGAs is based upon the CORDIC algorithm (Coordinate Rotation Digital Computer) [8], [4], [6], [10] that has become a common solution. The development of complex arithmetic functions in FPGAs results in the difficulty to implement such operations, therefore the elaboration of mathematical operations in Xilinx FPGAs are proposed trough the core generator [2]. The objective of this work consist in the development of the transforming Clarke and Park modules using a core with capabilities of performing mathematical operations such as trigonometric functions in a clock cycle, using an alternative method of the core generator suggested by the manufacturer. Such core is established by the user with software developed in Java. The maximum number of

functions that can be implemented depends on the available memory resources on the FPGA. When VHDL code generator is activated, at the beginning appears a window where the characteristics of the input and output variables are asked, at that moment the longitude of the input data indicating integer and decimal bits must be specified. What follow is the functions selection to be implemented according to the control algorithm. Finally the code generator creates a file containing the description of each module in VHDL language ready to be synthesized by the Xilinx ISE tool [11]. The functions are implemented in the embedded memory of the FPGA. The advantage presented by solving complex functions through preloaded tables can be seen in the computing time, simplifying the execution of a mathematical function to the transfer of a data value from memory to the accumulator register.

The rest of the work is organized as follows. The mathematical models of Clarke and Park transforms modules are revisited in Section 2. Section 3 presents the implementation of the modules in VHDL, while Section 4 describes the functioning of the whole system. Section 5 shows the implementation results and finally some comments conclude the work in Section 6.

2. Mathematical model of Clarke and Park transforms

Electric AC motors are divided in synchronous and asynchronous (induction motors), being the last one the more complex of all electric motors. There are various mathematical models for induction motors, one in particular is a motor related to reference frames. The (*a, b, c*) coordinate frames are the original one due to their triphasic nature of the induction motors as can be appreciated in figure 1.

The (*a, b, c*) mathematical model consist of six electrical and one mechanical time-variant differential equations that implies difficulties synthesizing a control law. It is worth to mention that the magnetic fluxes depend on time-variant inductances.

In order to eliminate this dependency, the stator and rotor circuits are referred to a common frame, this means that an random set of (*d', q'*) axis rotating at velocity $\omega'=d\theta'/dt$ are defined as shown in figure 2.

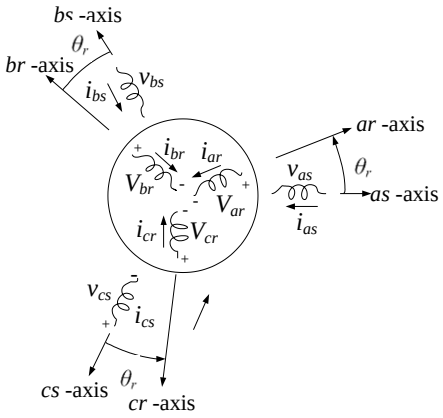

Figura 1. Induction motor windings.

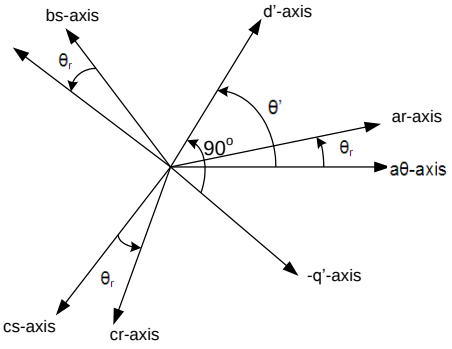

Figura 2. Location of the rotating (d', q') axis relative to the stator and rotor phases.

There are two common used reference frames for induction motors. One is obtained when $\theta'=0$, known as stationary (α,β) reference frame fixed to the stator. The second common used reference is when $\theta' = \theta_\phi$, known as rotating *(d, q)* reference frame fixed to the rotor flux vector, where θ_ϕ is the rotor flux angle in (α, β) coordinates. Control design methods performed in *(d, q)* coordinates are called FOC or vector control [5]. The *(a, b, c)* model, when transformed to (α, β) or *(d, q)* coordinates results in five time-invariant

differential equations. The main difference between both reference frames is the coordinate matrix used for transformation. The transformation matrix that takes stator voltages, currents and fluxes from *(a, b, c)* to *(α, β)* coordinates, i.e., the Clarke transform, is K_s:

$$F_{\alpha\beta} = K_s f_{abcs} \quad (1)$$

$$K_s = \frac{2}{3}\begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \quad (2)$$

Where the vector *f* can represent voltage, current and flux vectors, the subscript *s* refers to stator variables. Assuming a balanced circuit, i. e., $f_a + f_b + f_c = 0$, one can simplify [3] of the following form:

$$f_\alpha = f_a \quad (3)$$

$$f_\beta = \frac{2}{\sqrt{3}} f_b + \frac{1}{\sqrt{3}} f_a \quad (4)$$

The inverse Clarke transform that takes stator voltages, currents and fluxes from *(α, β)* to *(a, b, c)* coordinates, can easily be derived from [5] as:

$$f_a = f_\alpha \quad (5)$$

$$f_b = -\frac{1}{2} f_\alpha + \frac{\sqrt{3}}{2} f_\beta \quad (6)$$

$$f_c = -\frac{1}{2} f_\alpha - \frac{\sqrt{3}}{2} f_\beta \quad (7)$$

The transformation matrix that takes variables from *(α, β)* to *(d, q)* coordinates, i.e., the Park transform, is K_r:

$$f_{dq} = K_r f_{\alpha\beta} \quad (8)$$

$$K_r = \begin{bmatrix} \cos(\theta_\phi) & \sin(\theta_\phi) \\ -\sin(\theta_\phi) & \cos(\theta_\phi) \end{bmatrix} \quad (9)$$

Where:

$$f_d = f_\alpha \cos(\theta_\phi) + f_\beta \sin(\theta_\phi) \quad (10)$$

$$f_q = -f_\alpha \sin(\theta_\phi) + f_\beta \cos(\theta_\phi \quad (11)$$

Finally, the inverse Park transform is easily calculated as:

$$f_\alpha = f_d \cos(\theta_\phi) - f_q \sin(\theta_\phi) \quad (12)$$

$$f_\beta = f_d \sin(\theta_\phi) + f_q \cos(\theta_\phi) \quad (13)$$

2.1. Induction machine control

The induction motor drive is operated in torque mode. Indirect field oriented control (IFOC) is employed to control the torque applied to the induction machine. The basic concept of the IFOC technique involves the decoupling of the d-axis and q-axis components of stator current, the former being proportional to the flux while the latter is proportional to the electromagnetic torque [7].

The electromagnetic torque is controlled with the q-axis component of the stator current. A torque reference is generated and sent to an output by the supervisory DSP. This torque reference is sent to the motor drive inverter where it is converted to a q-axis current reference. This reference current is compared with the actual q-axis current and the resultant error is operated on by a Proportional-Integral (PI) compensator. The d-axis current command is sent directly to the motor drive as a function of the magnetization level required by the induction motor. Field

weakening is implemented at higher engine speeds. The FOC block diagram is shown in figure 3.

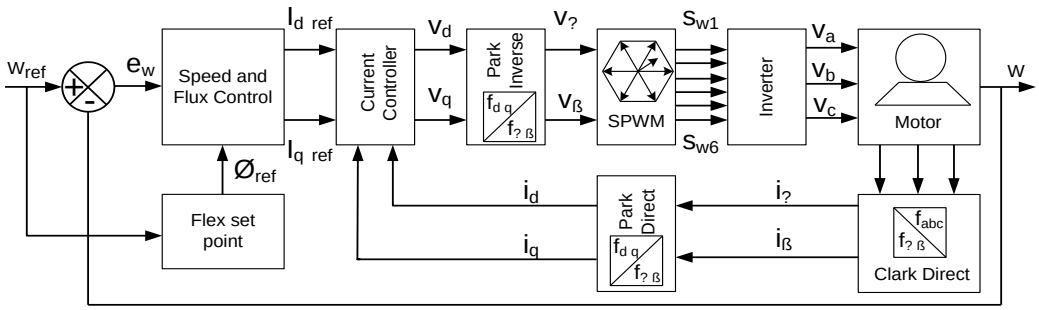

Figura 3. FOC block diagram.

3. Implementation of the transforming modules in VHDL

The blocks of the Park and Clarke transformations have been implemented in a Virtex II FPGA from Xilinx. These were designed originally in VHDL code through instantiation of RAM or ROM memories. The memories were programmed with pre-calculated values from the mathematical equations used in Clarke and Park transformations. For implementation of these equations there was software developed which was capable to generate VHDL code.

3.1. VHDL code generator

The modules proposed in this work are capable of solving trigonometric functions in a clock cycle by using pre-established data tables. To accomplish this, a program was developed in Java language that calculates the values of the trigonometric or mathematical functions within the range of values defined by the user, followed by the creation of tables with the calculated values, and implements a RAM or ROM memory to store these data values and then to translate them into the description hardware language VHDL [9]. The program defines the architecture, entity and process which automatically are added to the libraries, reducing user time and the definition of each block to only the definition of ranges and values of precision of the data of input and output in its integer and decimal part.

The software function generator reduces the computational burden to the FPGA by using a standard computer to calculate the possible results of mathematical functions that require only one parameter in the instantiation of a RAM or ROM memory.

The program creates the desired function as an entity in VHDL with an input and an output of the selected size. The VHDL has syntax standards, which are contained in libraries. The program generates the necessary lines to use the corresponding libraries. The entity block is also created at the same time, along with the input data, ready to be synthesized by the Xilinx ISE simulator.

3.2. Hardware implementation.

For the Clarke Transform there are two SIPO (Serial Input – Parallel Output) registers are used to load data in serial way to the FPGA, due that every input variable is 20 bit long. To obtain the serial output we use two PISO (Parallel Input – Serial Output) registers. The block diagram of the Clarke transform is shown in figure 5, which is constituted of 3 modules: SIPO input registers, Clarke transform module, and PISO output registers.

Inputs fa y fb, of Clarke's module (Ks1) are constituted by vectors of 20 bits (from 0 to 19), being the 19th bit the MSB, this is the sign bit.

The following 9 bits (18 down to 10) are the integer bits and the last 10 bits (9 down to 0) are the decimal part.

The variable outputs fα y fβ are output vectors of 40 bits (0 to 39) being the bit 39th the MSB and at the same time the sign bit. The following 19 bits (38 down to 20) are the integer bit and the last 20 bits are the decimal part.

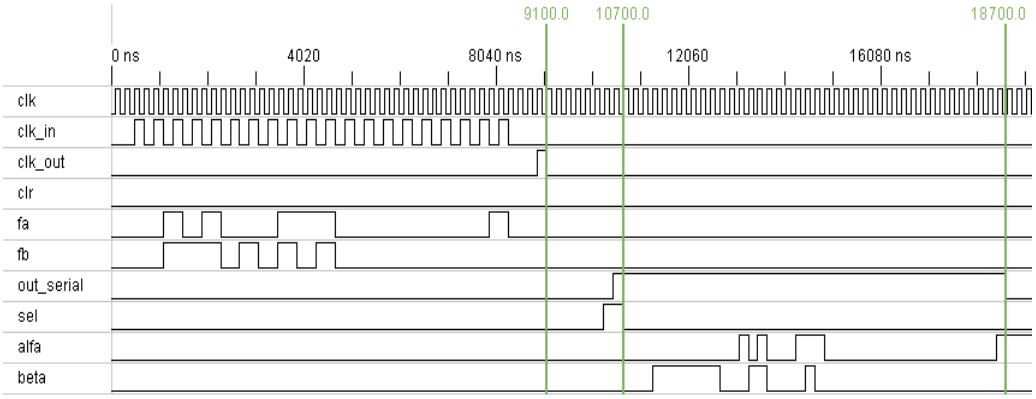

Figura 4. Clarke block diagram.

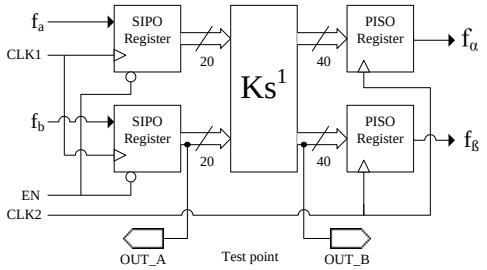

Figura 5. Clarke block diagram.

The clk bit in the registers has the function of synchronizing the data input. In each rising edge, a data is introduced to the registers and the data already in the register are shifted to the right (the data have to be introduced from the LSB to the MSB).

The bit EN (enable) enables the data to the output of the SIPO register. This signal allows every data of the registers to enter the block of equations of the Clarke in parallel to be processed.

The description of the module's equations to the Inverse Clarke Transform (Ks-1) is very similar to the direct transform, where fa and fb are output vectors and fα and fβ are input vectors. Also, there are used two SIPO registers to load the data in serial way in the FPGA and two PISO registers to obtain the output. There are also two signals to the outside of the FPGA: OUT_A and OUT_B as logic probes.

In the Park Transform three SIPO registers were used to load the data in serial to the FPGA and two PISO registers to obtain the serial output. One of the SIPO registers has a 14 bit output to introduce the angle θ. The other two registers have 20 bits outputs to introduce fα and fβ. The two PISO registers to the output store the results of fd and fq that may be displayed in serial, as shown in figure 6.

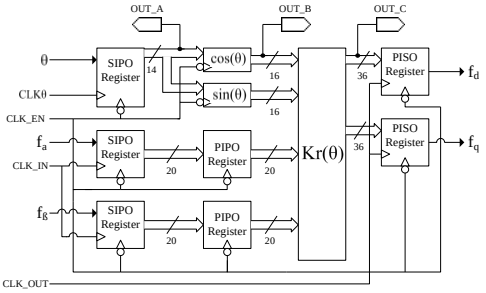

Figura 6. Park block diagram.

4. System functioning

The results of the functioning of the modules of the Clarke transforms are shown in the timing diagram in figure 4. This shows an example of the behavior of the transform where 10 signals may be observed. The signal "clk" is a clock used to obtain data in serial to the output. The "clk_in" is a clock signal that allows loading of data in serial to the SIPO registers, which is constituted of 20 pulses the size of the input word (20 bits).

The "clk_en" is a signal that enables the SIPO registers and loads the data in parallel to be processed in the Clarke transformation block. The signals "fa" and "fb" are the two 20 bit input vectors, the "fa" signal in the graph of the figure 4 has a negative value due to MSB being in '1' (aprox. 8040ns). The signal "out_serial" indicates the beginning of the displacement of the result in the PISO register, is the signal that enables the output of the result of the module. The signal "sel" is a selector that allows loading the result of the module of mathematical equations to the PISO register (value "1"). When "sel" has the value of "0" it starts to pull out the 40 data in serial way each time there is a rising edge in "clk" and this presents the signal "out_serial". The signals "α" and "β" are the outputs fα and fβ respectively.

In figure 4 three cursors are shown. The first one (9100ns) indicates the beginning of the mathematical processing of the input signals and the second cursor (10700ns) shows the beginning of the displacement of the results in the PISO register. The third cursor (18700ns) indicates the end of the displacement of the 40 bits corresponding to the "α" and "β".

It may be appreciated the MSB "α" (18600ns aprox.) is in high state, which indicates that the result is a negative value, and the MSB "β" is in low state (18600ns aprox.), which indicates that the result is a positive value.

5. Implementation results

The Clarke & Park transformation modules were implemented in a Virtex II Xilinx FPGA. The results of the utilizations are shown in the graphs of the figures 7 and 8. These show the slices, LUTs, flip flops, RAMs and multipliers. The figure 7 has the results of the implementation of the transformed of Clarke and his inverse, the figure 8 presents the results of the transformed of Park and the inverse of this one.

The results of the processing of the Clarke module implemented in the FPGA are showed in the figure 9, the variables of entry f_a and f_b have a signal shift of 120^o, and the θ angle grows in linear form and decreases to zero. The results obtained with the modules done in the FPGA were very similar to calculated with Matlab, the error of calculation is between a range of ± 0.01 for f_β and of ± 0.008 for f_α in case of Clarke transformed as is shown in the graph of the figure 10.

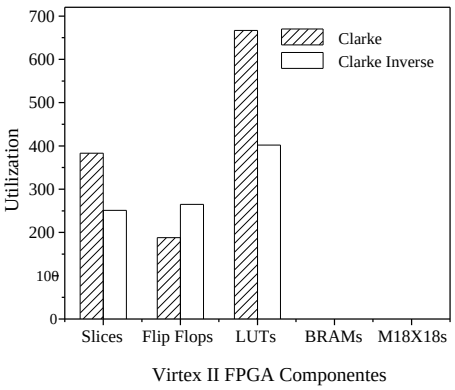

Figura 7. Clarke utilization in Virtex II.

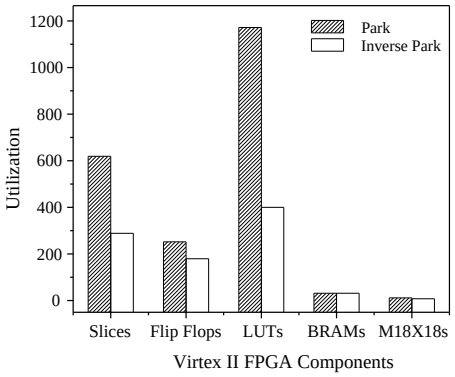

Figura 8. Park utilization in Virtex II.

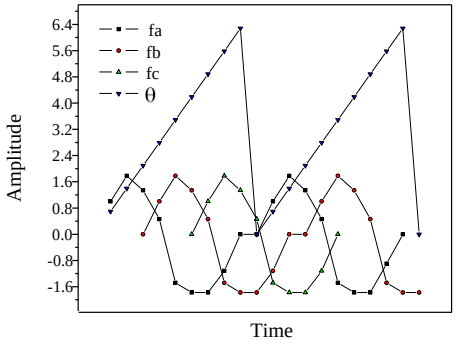

Figura 9. Results of the processing of Clarke's module.

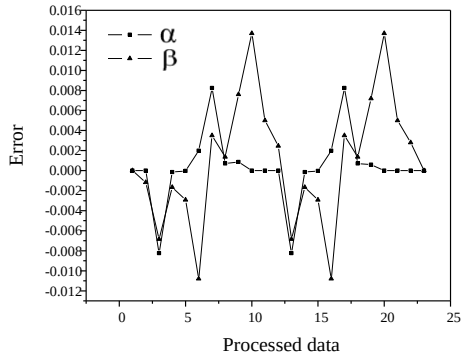

Figura 10. Error of calculation of the processing of Clarke's module.

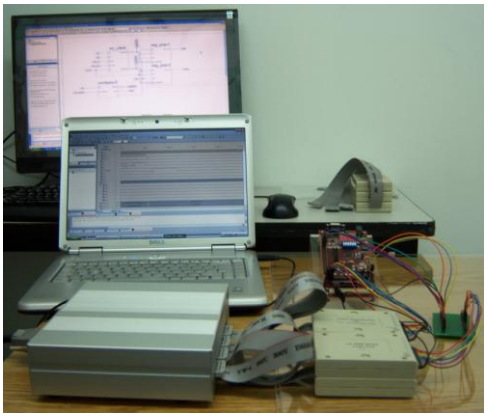

Figura 11. Work space and equipment used.

The figure 11 shows the equipment used in the laboratory for this work. This image shows a Virtex II Xilinx FPGA on which the equations of Clarke and Park were implemented, also shows a logic analyzer LA5000 to which all digital measurements were made using pattern generator pod that send the data to the system, while the logic pods monitored the inputs and take the reading of results.

The figure 11 shows a laptop that is used for taking measurements of the logic analyzer to be compared with the simulation done in Xilinx ISE 9.1.

A desktop computer is shown in the figure 11 which is used to program the Virtex II through the JTAG communication protocol.

6. Conclusions

This work presents the implementation of the Park and Clarke transformations in reconfigurable devices, as well as the application of the VHDL code generator, developed in language Java, designed to implement the mathematical equations in FPGAs of Xilinx.

The Park and Clarke modules are used for solving algorithm of control of engines and diverse applications.

In this work is demonstrated that the implementation in hardware satisfies the necessary requirements of speed and precision to work. The VHDL code generation tool allows implementing blocks of complex operations which can be grouped in the same memory allowing realizing the operations in a pulse of clock, based on the calculation of functions by means of pre-established tables. The implementation of this type of equations with this method is perfectly applicable in systems where there is known the range of work of the system.

References

[1] B.C. Toledo, A.G. Loukianov, Gennaro Di., J.R. Dominguez, "Output regulation for induction motors", *Control Conference, 5th*

Asian, 2004, ISBN: 0-7803-8873-9, Volume: 1, 2005-05-09, pp. 623- 628.

[2] Bravo, I. Jiménez, P. Mazo, M. Lázaro, J.L. Gardel, A. Marron, M., "Evaluation and selection of internal parameters of a CORDIC-unit for a specific application based on FPGAS". *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, ISBN: 978-1-4244-0829-0, 3-5 Oct. 2007, pp. 1 - 6.

[3] Chattopadhyay, S. Madhuchhanda Mitra Sengupta, "Power quality assessment in V-V, Clarke and Park domain", *42nd International Conference on Power Engineering, 2007. UPEC 2007.*, ISBN: 978-1-905593-34-7, 4-6 Sept. 2007, pp. 669 - 676.

[4] Circuits and Systems I: Regular Papers, IEEE Transactions on [*see also Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]*, Vol 52, Issue 11, Nov. 2005,Pages: 2385-2396.

[5] Dufour, C. Lapointe, V. Belanger, J. Abourida, S., "Closed-loop control of virtual FPGA-coded permanent magnet synchronous motor drives using a rapidly prototyped controller", *Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th*, ISBN: 978-1-4244-1741-4, 1-3 Sept. 2008, pp.1077 – 1083.

[6] Jie Zhou Yazhuo Dong Yong Dou Yuanwu Lei, "Dynamic Configurable Floating-Point FFT Pipelines and Hybrid-Mode CORDIC on FPGA", *Embedded Software and Systems, 2008. ICESS '08. International Conference on*, ISBN: 978-0-7695-3287-5, 29-31 July 2008, Pag 616 - 620.

[7] Kim, N.H. Kim, M.H. Toliyat, Hamid A. Lee, S.H. Choi, C.H. Baik, W.S., "Rotor fault detection system for inverter driven induction motor using currents signal", *Power Electronics, 2007. ICPE '07. 7th Internatonal Conference on*, ISBN: 978-1-4244-1871-8, 22-26 Oct. 2007 pp. 724 – 728.

[8] Martin Kuhlmann Keshab K. Parhi, "P-CORDIC: A Precomputation Based Rotation CORDIC Algorithm". *EURASIP Journal on Applied Signal Processing* , vol. 2002, Issue 9, pp. 936-943.

[9] R. Ortega. "Diseño e implementación de una unidad matemática, descrita en VHDL, para dispositivos reconfigurables", CONCIBE 2007.

[10] Vachhani, L. Sridharan, K. Meher, P. K., "Efficient CORDIC Algorithms and Architectures for Low Area and High Throughput Implementation", *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ISSN: 1549-7747, Vol. 56, Issue 1, Jan. 2009, Pp. 61 - 65.

[11] Xilinx, "www.xilinx.com", 2007.