Design method for vector control system implementations

József Vásárhelyi[1]
Tihamér Ádám[1]
1 University of Miskolc
1 Department of Automation

H 3515 Miskolc-Egytemváros
Hungary
{vajo, adam}@mazsola.iit.uni-miskolc.hu

Mária Imecs[2]
Sergiu Nedevschi[3]
2,3) Technical University of Cluj-Napoca
2) Faculty of Electrical Engineering,
3) Faculty of Automation and Computer Sciences
26-28 G. Baritiu st., RO400020 Cluj/Napocat
Romania
{Maria.Imecs, Sergiu.Nedevschi}@{edr, cs}.utcluj.ro

Abstract – **The technology development and the software tools, which allow the design and simulation of digital systems, allows the design of new intellectual property (IP) elements. This new IP elements permit the rapid prototyping of digital signal algorithms. The vector control systems are considered complex digital control systems, from many points of view. The implementation of such a system requires run-time computing. Besides the digital signal processors, the field programmable gate arrays are the new technology, which can implement such dynamical systems. This paper describes a design methodology for vector control systems based on the previously implemented vector control library.**

I. INTRODUCTION

Traditionally, digital processing tasks were implemented in Digital Signal Processors (DSP) using software implementation methods. Since the DSP is optimized for digital signal processing, however is not optimized for the specific algorithm, which implemented in software may result in poor performance. While implementing the it in hardware, the implementation is „tailored" to the specific algorithm, resulting in fast performance. The invention of Field Programmable Gate Arrays (FPGA) has given rise to an alternative method of computing. The FPGA provides the means for achieving hardware performance and software versatility. The FPGA implementation can be optimized for the specific algorithm, and can reuse and re-optimize multiple algorithms by simply reconfiguration of the device.

However, the technology expansion (i.e. packing more transistors and increasing clock speeds) is no longer the answer to the advanced computing needs of the future. Furthermore, rapidly changing standards and the demand of new applications cannot tolerate the limitations of the conventional fixed functions and rigid design approach (RISC, DSP, ASIC). More advanced flexible processing platform is required to enable the next generation applications, which will use intensively the reconfiguration possibilities [2].

The solution lies in adaptive computing, a new processing technology, with an integrated circuit (IC) architecture that changes on demand to perform much wider variety of functions at high processing speeds, with lower consumption, reduced silicon area and at very low cost.

At the beginning of the "FPGA era" most of reconfigurable computing systems were plug-in boards made for standard computers and they acted as a coprocessor attached to the main micro-processing unit. Many of applications of reconfigurable computing were reported in image processing, digital signal processing and custom computing machines. With the evolution of FPGA technology and the associated design tools, the application field has grown rapidly. Even more, this technology became a support for the new single chip solution called System on Chip (SoC).

Kiel and *Lenze* observed the tendencies in electronics for motion control systems and traced the future trends [3]. They predicted the increasing importance of the FPGA chips in motor control system implementations.

Considering the main advantages of FPGAs, regarding their reach-hardware resources and parallel algorithm implementation and also in the run-time hardware-structure reconfiguration possibilities, we may say that it will be the basis of many future applications.

The field-orientation principle introduced by Blaschke [1] in 1971 became the most efficient method for the control of induction motors. Using digital signal processing methods it allows the most dynamic control of the speed, torque, and position of AC drives. The implementation is effective if allows minimum sampling period and high execution speed. The reconfigurable technology fulfils the former conditions.

In the last years several vector control research groups presented different FPGA implementations, but more ore less these solutions were not intended to give a general design method for rapid prototyping and design of vector control systems see [4], [5], [6], [7].

This paper will present a unitary design methodology for rapid prototyping and design of vector control systems for induction drives.

II. MODULARITY OF VECTOR CONTROL SYSTEMS

Induction motors are perhaps the most rugged and best-understood motors presently available. The main advantages of these motors are their simple maintenance and their cost effective operation. The induction machine model based on the space phasor theory has a general character. To define the space phasors no restriction was imposed considering the time variation of the phase quantities. On the same theory is based also the field orientation principle, which leads to the vector control method of the AC drives.

The generalised block diagram of such a control system with induction machine supplied from a static frequency converter (SFC) is presented in Fig. 1, where the two control loops, corresponding to the active and reactive stator-current components are split.

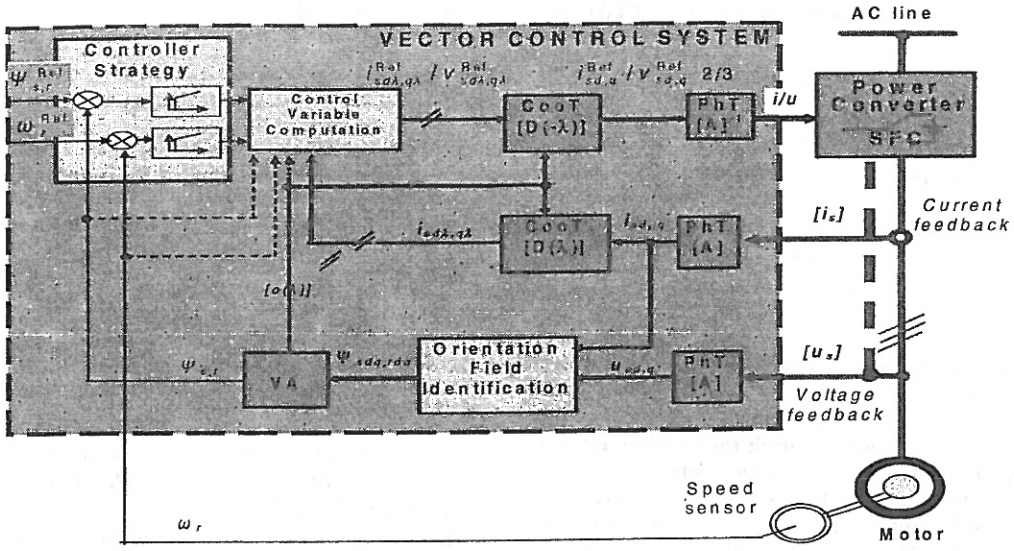

Fig. 1. Block diagram of a vector control system based on field-orientation.

The first condition of the application of the field-orientation principle is the identification of the main flux-vector position (argument).

The vector control structure of the induction machine depends on the orientation flux to be used, i.e. rotor-field (Ψ_r), air-gap field (Ψ_m) or stator-field (Ψ_s). Each type of orientation has advantages and disadvantages.

The orientation-field identification depends on the measured quantities. Considering the control loop and the sensors, generally there are two variants: *direct* (flux measurement) and *indirect* (calculated) flux sensing and also two field orientation - *direct and indirect* ones. Concerning the sensors, the direct flux measurement leads to simpler schemes in comparison with the flux calculated. However, direct flux measurement is difficult. The indirect flux sensing based on stator current and rotor speed measurement leads to the indirect filed orientation, which is parameter sensible method and needs to complex computation algorithms. For this reason, we will use the indirect flux sensing based on the stator voltage and current, which leads to direct flux orientation and fast algorithms implemented in hardware for the flux calculation.

In Fig. 1, one may observe, that in a vector control system structure there are some computing modules, which have to be part of any vector control scheme. These commonly used modules in this paper will be called ***characteristic modules***. They are:

- **PhT[A]** the direct- and **PhT[A]**$^{-1}$ reverse-Phase-Transformation module, which compute the transition from three-phase quantities (a, b, c) into two-phase ones (d, q) by applying matrix operator [A] and from two-phase quantities (d, q) into three-phase ones (a, b, c), with matrix $[A]^{-1}$, respectively;
- **CooT[D(*λ*)]** direct- and **CooT[D(*-λ*)]** reverse-Coordinate Transformation module, which rotates the two-phase axis with angle λ or $-\lambda$, i.e. computes from the d-q stator-oriented quantities the dλ-qλ stator/rotor field-oriented ones by means of the matrix operator **[D(*λ*)]**, and in the opposite direction, from the dλ-qλ stator/rotor field-oriented components computes the d-q stator-oriented magnitudes using operator **[D(*-λ*)]**, respectively;
- **VA** the Vector Analyser module, computes the amplitude (module), sine and cosine trigonometrical functions of the vector position (argument).

The structure of the vector control system depends on the field-identification method, the chosen orientation-flux (stator- or rotor resultant flux), controller type (PI, PID, adaptive, fuzzy, neural, etc.) and the static frequency converter topology, source-character (voltage- or current) and its pulse modulation procedure. The particular modules in this paper will be called ***specific modules***, as follows:

- Ψ_s**C** stator- and Ψ_r**C** rotor-flux computation module, which computes the orientation field by integration of the stator-voltage equation (Ψ_s) and compensation into Ψ_r.
- Controller Strategy block could be with **PI** character and also other types, like fuzzy-, neuro-fuzzy-, adaptive neuro-fuzzy controllers etc.
- Control variable Computation block contains the stator-voltage computation modules U_s**C** or V_s**C**.

The vector control systems require the machine dynamic model based on the space-phasor theory in order to apply the field-orientation principle in the decoupling of the control loops and in calculation of the control variables. The field-orientation principle offers the mathematical solution of the vector control systems and ensures high dynamic performance [9].

The most widespread method of the orientation-field vectorial identification is that, which utilizes the usual feedback signal (i.e. the rotor angular speed and the stator currents and voltages). The structure of the implemented control system is quite complex, but the cost is reduced due to the new technologies such as Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs) or Configurable System on a Chip (CSoC) chips.

III. DECOMPOSITION OF DIGITAL SIGNAL PROCESSING ALGORITHMS

If the digital signal-processing algorithm presents modularity, the implementation is easier. This is the reason why the vector control system was analysed and divided in characteristic and specific blocks. Also digital signal processing is *speed and algorithm critical* area where until the recent years DSP processors had the domination. Traditionally, digital signal processing algorithms are most commonly implemented using general-purpose (programmable) DSP chips for low rate applications, or special-purpose (fixed function) DSP chip-sets and application-specific integrated circuits (ASICs) for higher rates.

Technological advancements in Field Programmable Gate Arrays (FPGAs) in the past 5 years have opened new possibilities for DSP design. The FPGA maintains the advantages of the high specificity of the ASIC while avoiding the high development costs and inability to make design modifications after production. The FPGA also adds design flexibility and adaptability with optimal device utilization conserving both board space and system power consumption, which is not often the case with DSP chips. When the design is demanding higher speed and the application it is time-to-market critical or design adaptability is crucial FPGAs are the solution.

The SRAM based architecture of the Xilinx FPGA/the Triscend CSOC is well suited for *multiply and accumulate* (MAC) intensive DSP functions. Wide ranges of arithmetic functions are implemented into the FPGA and if needed they are reconfigured „on-the-fly". These functions includes, fast Fourier transforms (FFTs), convolution, filtering algorithms, as well as mathematical operations and the surrounding peripheral circuitry. Designing a digital signal processing system, using FPGA chips the data can be processed taking advantage of single chip paralleled structures and distributed arithmetic algorithms to exceed the performance of a single general-purpose DSP device.

A. Algorithm analyses and modularity

The analysis of the vector control schemes and especially the vector control structures of the reconfigurable "tandem" converter were presented in detail in [11]. The analyses for reconfigurable vector control systems were performed based on the following criteria [12]:

Given any vector control structure if exist any common modules then:

- Which are the common modules in the same position with the same function?
- Which are the common modules with different functionality?
- Which are the particular modules of each (reconfigurable) structure?
- When reconfiguration condition occurs, is it possible the output variable value transfer to the modules on the same position or no output variable value transfer allowed?
- Is the output variable value transfer of the controllers?
- Is it possible to give a general mathematical model for all of the modules?

Resulting from the analyses of the vector control structures one can do the decomposition of any vector control algorithm in elementary operations. From this decomposition and from the modularity of the algorithm was created an *intellectual property* (IP) library made especially for the implementation and rapid prototyping of vector control systems [12].

There are some major advantages of using a pre-designed parametrical IP core library when implementation is targeted. These advantages are:

- The implementation time of the simulation model is short, as the simulation model is the implementation itself. This can be done with the translation of the cores in FPGA configuration data (using the development environment).
- The computation speed increase. This results from the parallel implementation of computation algorithm of both components (d, q) and the parallel computation of each IP core. This is a significant advantage compared to the DSP sequential implementations.
- The parameters of each IP core element can be adjusted easily to any AC motor characteristics. Even the data format can be modified if necessary.
- Flexibility in implementation: each IP core can be translated separately and the vector control system can be translated as a whole.
- The targeted device can be changed if necessary.
- The optimisation of the IP elements is made for speed or/and area, which are characteristic to FPGA implementations. [11], [12]

IV. MODULE LIBRARY FOR VECTOR CONTROL SYSTEMS

After the vector control system decomposition in elementary operations a module library was created in order to help the rapid prototyping of the vector control systems.

There are some major advantages of using a pre-designed parametrical IP core library when implementation is targeted. These advantages are:

- The implementation time of the simulation model is short, as the simulation model is the implementation itself. This can be done with the translation of the cores in configuration data (using the Xilinx development environment)
- The computation speed increase. This results from the parallel implementation of computation algorithm of both components (d, q) and the parallel computation of each IP core. This is a significant advantage compared to the DSP sequential implementations.
- The parameters of each IP core element can be adjusted easily to any AC motor characteristics. Even the data format can be modified if necessary.
- Flexibility in implementation: each IP core can be translated separately and the vector control system can be translated as a whole.
- The targeted device can be changed if necessary.

The optimisation of the IP elements is made for speed or/and area, which are characteristic to FPGA

implementations. In TABLE I are presented some of the elements of the implemented library. The table presents the implementation slices consumed by each element, the time delay introduced, the maximum working frequency (were applicable) and the quantisation error of the elements.
From the implementation results, one can estimate the hardware necessities of the vector control system, the maximum delay introduced by the system, can analyse the quantisation error of each module, etc.

The Xilinx System Generator, which was used to implement the library elements, allows the estimation of the quantisation error introduced by each module. For example the quantisation error introduced by the phase transformations are:

$$\Delta\varepsilon_d \in [-1.5\cdot10^{-4}, 1.5\cdot10^{-4}], \quad (1)$$

$$\Delta\varepsilon_q \in [-1.5\cdot10^{-4}, 3\cdot10^{-4}], \quad (2)$$

where $\Delta\varepsilon_{d/q}$ is the quantisation error.

TABLE I

CHARACTERISTICS OF THE IMPLEMENTED VECTOR CONTROL LIBRARY MODULES [11], [13].

Library element or IP Module name	**Symbol Name**	**Slices needed for implementation**	**Worth path delay introduced td[ns]**	**Quantisation error**	**Max. working f[MHz]**
Direct Phase Transformation Block	PHT[A]	152	27.00	$q_e<1.5*10^{-4}$	n.a
Reverse Phase Transformation Block	$PHT[A]^{-1}$	217	4.90	~0	n.a
Stator+Rotor-flux Compensation	$\Psi_s+\Psi_r Co$	1000	41.70	$-0.02<q_e<0.1$	42,00
Vector Analyser	VA	1995	16.60	n.a	166,90
Coordinate Transformation	$CooT[D(\pm\lambda_r)]$	25	10.00	$<1.\ 10^{-4}$	n.a
Space Vector Modulation	SVM	27	3.06	n.a.	n.a
Current feedback Modulation	PWM	77	31.20	n.a	224.15
Flux Controller	Ψ-PI	24	4.00	$\sim0.6*10^{-4}$	n.a
Speed Controller	Ω-PI	24	4.00	$\sim0.6*10^{-4}$	n.a
Flux+Speed Controller	PI	135	13.73	$q_{eflux} \cong -6*10^{-3}$ $-0.1<q_{espeed}<0.16$	128.18
Reconfiguration Multiplexer	MUXr	5	3.12	0	n.a
CSI current constant multiplier	K1	79	16.99	$-2*10^{-5}<q_e<5*10^{-5}$	n.a
Total Estimated resources	Tandem	4496 slices 1058 FF 9334 LUT	48.29	n.a	38.45

Fig. 2 presents graphically the quantisation error variation for the output values of stator voltages u_{sd}-u_{sq}.

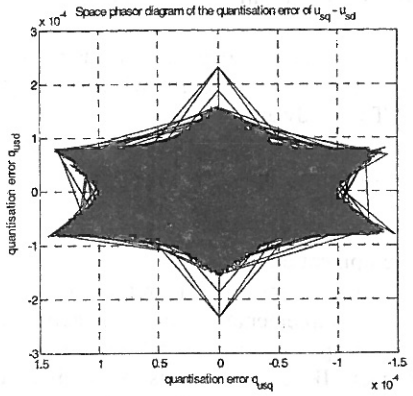

Fig. 2. Quantisation error of the block PHT[A] for the variables u_{sd} - u_{sq}

We can say that, comparing the input value ranges of the voltage/current to the quantisation error values, the obtained output values have an acceptable precision.

V. DESIGN METHODOLOGY

The design methodology is also a result of the development of the vector control IP library. Since the library is intended for implementation of any vector control system, the methodology has also a general character. In the following we present the design steps proposed to follow in order to use the module library for any vector control system implementation:

1. Development of the vector control structure for the asynchronous drive.
2. Development of the per unit model of the induction motor drive.
3. Identification of the existing modules from the module library.
4. Development of the vector control system implementation model in MATALB SIMULINK using the module library.
5. Design of the new modules using System Generator or VHDL.
6. Simulation of the constructed vector control system and/or simulation of the model calling the Modelsim simulator from Simulink environment.
7. Compilation of the developed/simulated model into VHDL Xilinx Integrated System Environment project.
8. Translation of the project into configuration bits.
9. Testing the vector control system using the prototyping board.

The methodology weak point is that not every module can be implemented using the Xilinx System Generator elements. In such cases one have to implement the new

module – if does not exist in the library – using VHDL language, or other tools such as Xilinx Core Generator. Using these development tools one can implement in VHDL the new library element, and then this implementation can be imported to the Simulink environment for further simulations.

As example we can consider the development of the so called Vector Analyzer block. This module was implemented using VHDL language. The corresponding code sequence for the VA is presented in Fig. 3. The figure shows the declaration of the square root and division functions used for the implementation of the VA.

```
library ieee;
use ieee.std_logic_1164.ALL;
library XilinxCoreLib;
entity sqrt is
        port (
        x_in: in std_logic_vector (31 downto 0);
        x_out: out std_logic_vector (15 downto 0);
        clk_a: in std_logic_vector (0 downto 0));
end sqrt;

architecture sqrt_a of sqrt is
component wrapped_sqrt
        port (
        x_in: in std_logic_vector (31 downto 0);
        x_out: out std_logic_vector (15 downto 0);
        clk_a: in std_logic_vector (0 downto 0));
end component;
-- Configuration specification
        for all : wrapped_sqrt use entity XilinxCoreLib.cordic_v2_0(behavioral)
        generic map(
                        c_has_clk => 1,
                        .....
                        c_input_width => 32,
                        ........
                        c_pipeline_mode => 0,
                        c_output_width => 16);
begin
U0 : wrapped_sqrt
        port map (
                        x_in => x_in,
                        x_out => x_out,
                        clk_a => clk_a);
end sqrt_a;

entity divider32 is
  port (
                dividend: in std_logic_vector (31 downto 0);
                divisor: in std_logic_vector (15 downto 0);
                quot: out std_logic_vector (31 downto 0);
                remd: out std_logic_vector (15 downto 0);
                c: in std_logic_vector (0 downto 0));
end divider32;

architecture divider32_a of divider32 is
component wrapped_divider32
port (....)
end component;
-- Configuration specification
for all: wrapped_divider32 use entity XilinxCoreLib.dividervht(behavioral)
                generic map(
                        dividend_width => 32,
                        signed_b => 1,
                        .........
                        divclk_sel => 1);
BEGIN
U1: wrapped_divider32
                port map (
                        dividend => dividend,
                        divisor => divisor, .....)
end divider32_a;
```

Fig. 3. Square root and division VHDL code sequences.[11]

The VA was implemented considering the following per unit equations:

$$|g| = \sqrt{g_d^2 + g_q^2}; \tag{3}$$

$$\sin\alpha = \frac{g_q}{|g|}; \quad \cos\alpha = \frac{g_d}{|g|}. \tag{4}$$

where $g_{d,q}$ are the input variables and $|g|$ is the phasor amplitude.

In the compilation process of the vector control system one have to consider the targeted chip, since the module parameters allow to use hardware specific implementation. One of these hardware specific conditions is the use of Xilinx Virtex hardware multipliers. If one considers for implementation this FPGA family, it is recommended to make use of such multipliers.

Another consideration is the board FPGA board which will be used. There is a possibility to adapt any Xilinx board to the MATLAB SIMULINK environment. At this moment this is subject of further research.

VI. SIMULATION/IMPLEMENTATION RESULTS

In the research work in all the possible tested vector control systems the simulations were performed with similar conditions in MATLAB-Simulink environment using the intellectual property library. The motor data are: 5.5 kW, 50 Hz, 220 V^{rms}, 14 A^{rms}, 720 rpm and 4 pole-pairs. The motor was started controlled by a DC link frequency converter and after some time (usually 0.5 sec or 1s after start) the control structure was reconfigured [11]. The results we obtained using this methodology are presented in the following figures.

Fig. 4 shows the implementation report of the vector control system compiled in the Xilinx ISE environment.

```
Device utilization summary:
---------------------------

Selected Device : 3s2000fg676-4

 Number of Slices:              4418  out of  20480   21%
 Number of Slice Flip Flops:    936  out of  40960    2%
 Number of 4 input LUTs:        8174  out of  40960   19%
 Number of bonded IOBs:          192  out of    489   39%
 Number of GCLKs:                  1  out of      8   12%

Timing Summary:
---------------

Speed Grade: -4

 Minimum period:                   26.005ns (Maximum Frequency: 38.454MHz)
 Minimum input arrival time before clock:  30.171ns
 Maximum output required time after clock:  43.270ns
 Maximum combinational path delay:  48.291ns
```

Fig. 4. Implementation results of the vector control system [11].

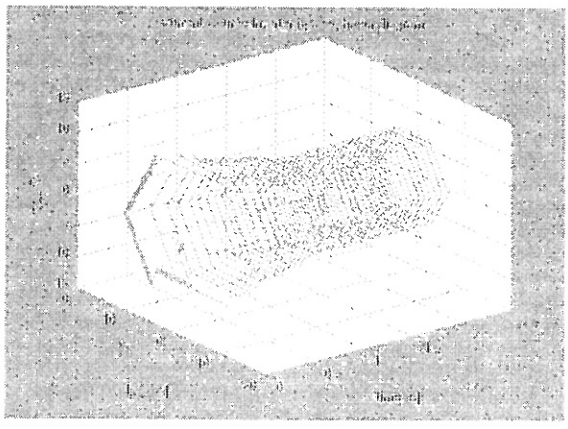

Fig. 5. Current Source Inverter space-phasor diagram.

The implementation was performed using different hardware support. One of the used boards was the Triscend Configurable System on Chip (CSoC) board and the other was the Digilent FPGA board.

The vector control system is implemented in the FPGA board. The CSoC board is used for control and configure the FPGA board, since the constructed system is intended for run time reconfigurable control systems.

VII. CONTRIBUTIONS

The paper presented a design methodology for vector control systems, which are implemented using FPGA chips. The methodology makes use of the library elements implemented in MATLAB SIMULINK environment and the System Generator toolbox for Xilinx FPGA chips.

The methodology described here is an original contribution. Using the library elements any vector control system can be implemented in FGPA system. Further research should be done to connect the FPGA board to the MATLAB environment.

VIII. ACKNOWLEDGMENT

The authors gratefully acknowledge the donations of *Xilinx Inc.* which made possible the research on some aspects of reconfigurable vector control framework.

The research is part of the Hungarian-Romanian Cooperation Project TET RO-16/2003 and HU-11/2002.

IX. REFERENCES

[1] F. Blaschke, „Das Prinzip der Feldorientirung, die Grundlage für die Transvektor-Regelung von Drehfeldmaschinen, *Siemens-Zeitshrift 45*. HEFT No 7, 1971.

[2] P. Master, "The Age of Adaptive Computing is Here," *Proceedings of the IEEE 12th International Conference Field-Programmable Logic and Applications FPL2002*, LNCS2438, Springer, ISBN 3-540-44108-5, Montpellier, France, Sept. 2-4, 2002, pp. 1-3.

[3] E. Kiel, A. Lenze, "Control Electronics in Drive Systems Micro Controllers, DSPs, FPGAs, ASICs From State-Of-Art to Future Trends," *Proceedings of the PCIM 2002 in volume Intelligent Motion*, May 14-16, 2002 Nuremberg, Germany, pp. 277 – 282.

[4] J. F. Herbert, S. Beierke, "Universal PWM-Unit with DSP-Interface realized in a TI-FPGA 1280." Texas Instruments 1994, CDROM.

[5] E. Monmasson, K. Tazi, M. Grandpierre, "Description of an entirely re-programmable hardware system dedicated to the control of a three phase PWM inverter," *Proceedings of PEMC'98 Prague*, vol. 2, September 1998, pp.179-184.

[6] M. N. Cirstea, A. Dinu, J. G. Khor, M. McCormick, *Neural and Fuzzy Logic Control of Drives and Power Systems*, Newness, ISBN 0-7506-55585, 2002, P399.

[7] P. Poure, F. Aubépart, F. Braun, "ICs for real time Motion Control: A design methodology for rapid prototyping," *Procedings of PCIM 2001 in volume Intelligent Motion*, June 19-21, 2001 Nuremberg, Germany, pp. 211-216.

[8] Leonhard W., *Control of Electrical Drives*, Spriger Verlag, Berlin, 1985.

[9] A. Kelemen, Maria Imecs, *Vector Control of AC Drives*, Volume 1: *Vector Control of Induction Machine Drives*, OMIKK Publisher Budapest, ISBN 963 593 140 9, Budapest, Hungary, 1991, p. 364.

[10] Goslin G. R., "Using Xilinx FPGAs to Design Custom Digital Signal Processing Devices," *Application Note*, Applinx Xilinx Inc, 1999.

[11] J. Vásárhelyi, *Reconfigurable Architectures for Vector Control Structures of Induction Motor Drives*, PhD Thesis, Technical University of Cluj-Napoca, 2004, p. 180.

[12] J. Vásárhelyi, Maria Imecs, J. J. Incze, Cs. Szabó "Module Library for Rapid Prototyping and Hardware Implementation of Vector Control Systems," *Proceedings of IEEE International Conference on Intelligent Engineering Systems INES 2002*, May 26-28, 2002, Opatija, Croatia, ISBN 953-6071-17-7, pp. 447-452.

[13] J. Vásárhelyi, Maria Imecs, J. J. Incze, Cs. Szabó, T. Ádám, "FPGA Implementation of the Reconfigurable Tandem Inverter Fed Control System for AC Drives," *Proceedings of the 11th International Conference on Power Electronics and Motion Control EPE-PEMC 2004*, Riga, under edition.