International Journal of Electric and Power Engineering 2 (4): 262-270, 2008
ISSN: 1990-7958
© Medwell Journals, 2008

VLSI Design Methodology for Hardware Prototyping of Integrated Direct Torque Control of Induction Motor Drives

[1]Sarat Kumar Sahoo, [2]G. Tulasi Ram Das and [3]Vedam Subrahmanyam
[1]Department of EEE, M.G.R. University, Chennai, 600037, India
[2]Department of EEE, J.N.T.U. College of Engineering, Hyderabad, India
[3]Department of EEE, R.M.K. Engineering College, Chennai, India

Abstract: This study presents a Top-down methodology for hardware rapid prototyping of integrated Direct Torque Control of Induction motor drive control, based on Hardware Description Languages (HDL's). This methodology is a set of procedures and Computer Aided Design tools to optimize development time, final product cost and reusability of the digital electronic system design. The use of HDL's provides continuous checking of functional validation of the control and makes the digital design independent of the target technology. More, HDL's are common languages throughout all design stages. The three stages of the methodology are discussed. The behavioral stage defines control functionality at system level and allows the study of Integrated Circuit digital properties. In the structural stage, the design is defined as a set of components and interconnections. Finally, in the physical stage implementation technologies, FPGA is considered. Matlab/Simulink is used to validate the algorithm. This study presents an application of the methodology to the integration of Direct Torque Control (DTC) of an induction machine. First of all, the DTC principle is described. Then, step by step, the behavioral, structural and physical stages of digital control representation are discussed and results are validated.

Key words: Field Programmable gate arrays (FPGAs), Hardware Description Language (VHDL), design methodology, Direct Torque Control (DTC)

INTRODUCTION

With the growing complexity of motor and motion control applications, it becomes apparent that a Field Programmable Gate Array (FPGA) offers significant advantages in the area of performance, flexibility and inventory control. With an FPGA, calculations that would normally consume large amounts of CPU time when implemented in software may be hardware accelerated. In digital electronics systems the basic kinds of devices are memory, microprocessor and logic. Memory devices stores random information, microprocessors execute software instructions and logic devices provides specific functions including device-to-device interfacing, data communication, signal processing, data display, timing and control operations and almost every other function a system must perform (Altera Corporation, 2003; Xilinx Corporation, 2006). Logic devices can be classified into two broad categories- fixed and programmable. Fixed logic device circuits are permanent, once manufactured they can not be changed. Programmable logic devices (PLDs) are standard and these devices can be changed at any time to perform any number of functions. The two major types of Programmable logic devices are Field Programmable Gate Arrays (FPGAs) and Complex.

Programmable Logic Devices (CPLDs). Of the two, FPGAs offer the highest amount of logic density, the most features and the best performance. The advanced devices also offer features such as built-in hardwired processors, substantial amount of memory, clock management systems and support for many of the latest, very fast device to device signaling technologies. FPGAs are used in a wide variety of applications ranging from data processing, data storage, instrumentation, telecommunication and digital signal processing. To make the design of complex system manageable, the design team has to be coordinated throughout the complete development period. Communication among them should be made using a common format or language, which makes possible understanding. Hardware Description Languages (HDL's) have come to solve some of the problems previously encountered (IEEE Std, 1076-1993; Cirstea *et al*., 2000).

Corresponding Author: Sarat Kumar Sahoo, Department of EEE, M.G.R. University, Chennai, 600037, India

In the last few years, the number of paper presented at conferences and targeted for journal publication reflected a significant increase in the interest paid to the use of FPGAs in a wide range of electronics systems. This study focuses on FPGA usage in the realm of industrial drives. This paper is organized as follows.

FPGA DESIGN METHODS

Field Programmable Gate Array (FPGA) is a good compromise between the advantage of the flexibility of a programming solution and the efficiency of a specific architecture with a high integration density. These characteristics are quite appropriate for controller design. Thus, in order to benefit from the advantages of the FPGA and their powerful CAD tools, the designer has to follow an efficient design methodology (Wolf, 2004). Such a methodology has two main principles: the modularity and the best suitability between the algorithm to implement and the chosen hardware architecture.

For very complex designs, modular conception is generally used to reduce design cycle. This methodology is based on hierarchy and regularity concepts. Hierarchy is used to divide a large or complex design into sub-parts called modules that are more manageable. Regularity is aimed to maximize the reuse of already designed modules (Trimberger *et al*., 1981). Nowadays, the manufacturers and the designers of circuits even propose to recover in free (http://www.opencores.org) or restricted access (Altera Corporation, 2003; Xilinx Corporation, 2006), several design models, also called Intellectual Property IP modules. Besides, the complexity of some modules can be important as for the processor-cores (http://www. fpga_cpu .org). This design approach is then based on the reusability of IP modules (Kebbati *et al*., 2001).

To be efficient, the modular design approach must be based on reliable modules. However, in many cases, the desired modules do not already exist and they have to be realized. It is therefore crucial when designing them to be helped by an efficient methodology that allows taking into account the numerous constraints of such systems. In order to be more explicit, a simple algorithm example called the coordinate transformation is treated.

The coordinate transformation is used to transform the actual quantities of a three-phase electrical system (x_1, x_2, x_3) onto a d-q reference frame that is rotating at an arbitrary angle θ. By making the assumption that the studied three-phase system is balanced and the instantaneous power is preserved, the transformation can be simplified and expressed as:

$$\begin{bmatrix} X_d \\ X_q \end{bmatrix} = \sqrt{2}\begin{bmatrix} -\sin(\theta - 2\pi/3) & \sin(\theta) \\ -\cos(\theta - 2\pi/3) & \cos(\theta) \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (1)$$

The former expression is then converted to a n-bit fixed-point format. This format must be the result of a compromise between the required computing accuracy and the available hardware resources. It gives:

$$\begin{bmatrix} X_d \\ X_q \end{bmatrix} = \sqrt{2}\begin{bmatrix} -A_{11}(\theta) & A_{12}(\theta) \\ -A_{21}(\theta) & A_{22}(\theta) \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (2)$$

Where, the X_n -bit signed fixed-point value is equal to

$$x = XQ_X \text{ with } Q_X = \frac{x_{max}}{2^{n-1}}$$

Scale factor Q_X, has to be selected with relevance by the designer so as to avoid overflow errors. Besides, during the conversion process, the designer can also simplify the implemented equations from an adequate choice of scale factors (Chapuis *et al*., 1998; Tazi *et al*., 2000) and highlight the nature, the order and the frequency of the operations to be executed.

USE OF FPGA IN MOTOR CONTROL

FPGAs for industrial drives started appearing since a decade. There has been a surge of developmental activity in FPGA-based industrial drives as FPGAs prices have come down in the last five years. A prototyping system is required at every stage of the development cycle for a Variable Speed Drives (VSD) since, it provides the designer an environment and necessary tools to edit, compile, test and debug the control code under operating conditions similar to that of the actual VSD. The complexity of a prototyping system depends largely on the controller chip used (Dubey *et al*., 2007).

The design process can be divided into two successive phases: simulation and implementation. During the simulation phase, the VSD is modeled and simulated with the best accuracy possible to represent the operation of the drive under different operating conditions. The performance of the controller is evaluated and some design iterations are required to optimize its performance. The following phase is the implementation of the simulated controller on actual hardware to verify its real-time performance.

In a conventional design approach, simulation and implementation phases are performed on two different platforms. Simulation is usually done on workstation or

PC using programs or a system simulator (for example Simulink, Saber, etc.). For the implementation phase, the controller C program (which is directly written for the simulation and modified for the implementation or automatically generated by a specialized tool such as the Real-Time Workshop of Simulink) is modified to suit the hardware. The compiled code is then downloaded to the hardware (DSP or microcontroller board) for execution and debugging. Some iteration between the simulation and implementation phases are required to fine tune the controller. This approach tends to be somewhat lengthy due to the code modification and adaptation required to move from the simulation platform to the implementation platform. It is possible to accelerate the design process by using a unique platform for both simulation and implementation.

In the simulation phase, the VSD controller is represented by hardware models of the actual elements which will be implemented on FPGA. In fact, the simulation model represents in every detail the controller to be implemented in FPGA. When the debugging has been completed, the simulated controller can be converted directly into hardware without further debugging or adaptation. This is the reason of the reduced design time which this approach can provide as compared to conventional approach where debugging is needed in both simulation and implementation phases. Control algorithm is no longer a limiting factor for its implementation. Another advantage of the proposed approach concerns the execution speed of the implemented controller. Since, all operations in FPGAs are in parallel in hardware, the control algorithm can be executed very fast and a high sampling rate can be sent for the controller. As a result, the complexity of the control algorithm is no longer a limiting factor for its implementation.

The general structure of a control process of induction motor, as shown in Fig. 1, can be divided into the control algorithm and the tuning process interface. The power system can also be divided into three parts: the voltage source inverter (VSI), the machine and the feedback sensors (Kiel *et al*., 2002). Digital controllers have predominated over their analog counterparts. Advantages such as the implementation of complex more control algorithms, immunity to noise or flexibility to modify the parameters of the control laws are causing the migration from analog to digital controllers (Pimentel and Huy, 2000). The digital control algorithms have many characteristics related to the power environment. In fact, the power interface introduces a set of limitations in terms of time response and stability related to: sampling rate limited by the switching losses of the VSI, delay and

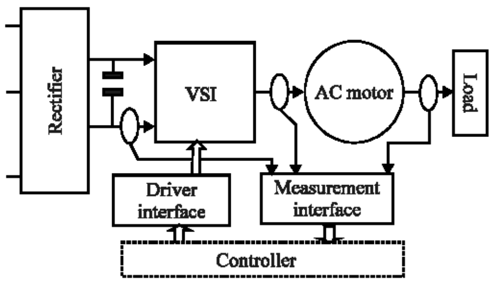


Fig. 1: General structure of a controlled induction motor

quantization. Furthermore, the control laws are constituted of heterogeneous functions because of the diversities in algorithmic operations (memory, arithmetic operations…) and differences in rate of computation. Besides, the control algorithms are always recursive which complicates the algorithmic partitioning and the digital data coding. The complexity of the control algorithms makes the Algorithmic study an important phase to design an efficient digital control solution. It begins by simulating the algorithms in Matlab from Mathworks®. This brings to focus the problems of data algorithm dependencies in order to find independent sub-algorithm functional blocks as well as analyzing fixed point refinement for variable coding (including the influence of word length, overflow and sampling frequency).

DIRECT TORQUE CONTROL PRINCIPLE

In direct torque control method the stator flux magnitude is controlled. In the stationary frame the stator flux can be represented by Eq. 3.

$$\left.\begin{aligned} \overline{\psi_{ds}} &= \int (\overline{V_{ds}} - \overline{i_{ds}} R_s)dt \\ \overline{\psi_{qs}} &= \int (\overline{V_{qs}} - \overline{i_{qs}} R_s)dt \end{aligned}\right\} \tag{3}$$

In DTC, voltage vector selection is limited to the eight available from the inverter. Figure 2 shows the available voltage vectors superimposed on the space vector diagram.

The architecture of Direct Torque Control Principle is shown in Fig. 3. The stator flux can be controlled by selecting one of the available voltage vectors which will move the flux in the desired direction. The magnitude of the flux is controlled by selecting a voltage vector which will either increase or decrease flux magnitude.

In DTC, torque control is achieved by controlling the quadrature component of the stator flux relative to the rotor flux. One of the available voltage vectors is

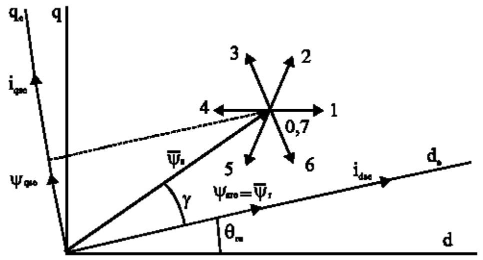


Fig. 2: Space vector diagram showing available voltage vectors for DTC

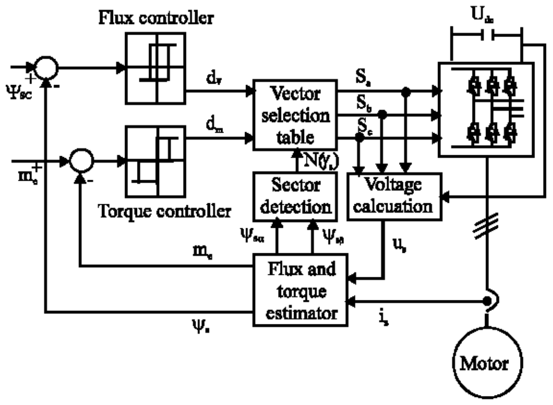


Fig. 3: Direct torque control architecture

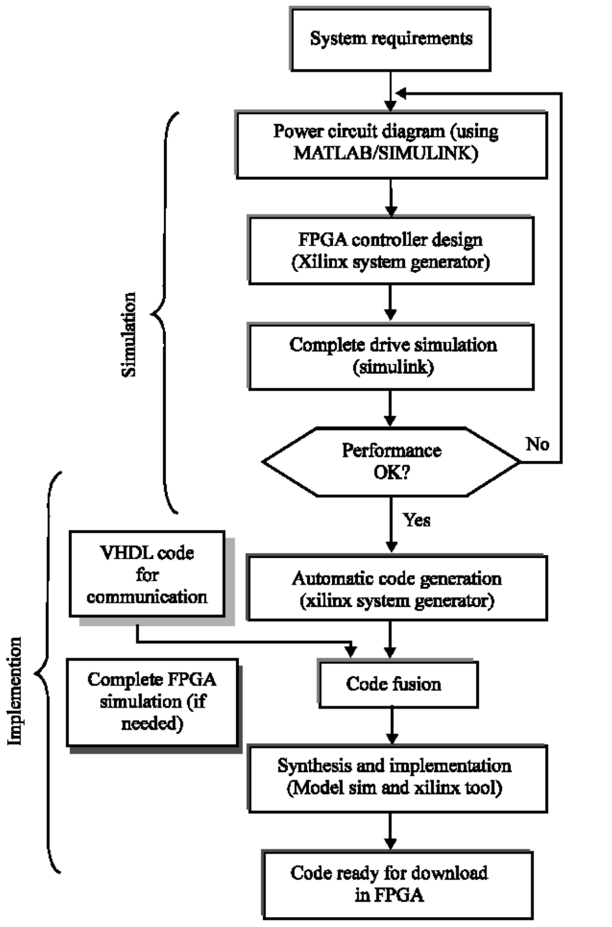
Table 1: Direct torque control switching table

Torque error $(T_{ref}-T_e)$	Flux error $(\psi_{ref} - \psi)$	Voltage vector selection
+ve	+ve	n+1
+ve	-ve	n+2
-ve	+ve	n+5
-ve	-ve	n+6
Null	Don;t care	Zero

selected which will increase or decrease the quadrature component of the flux in order to increase or decrease torque. Selection of the appropriate vector is based upon a hysteresis controller for each of the flux and torque control loops and a look-up table which utilizes the stator flux position. The stator flux position is divided into six 60° regions, denoted as n, where ((2n-3) /6 < θ < (2n-l) π/6), giving a voltage vector selection table shown in Table 1.

METHODOLOGY FOR HARDWARE RAPID PROTOTYPE

Generality: The design process of a digital FPGA consists of the following stages: logic design and simulation, placement and routing, design rule check and finally prototype production. For these reasons, the advance of FPGA technologies in industrial drive control applications seems relatively slow because of the industrial perceived disadvantages of long lead times and initial cost when classical design method is used (Le-Huy, 1994). In this study, presented a Top-down methodology, particularly adapted to integration of drive control. However, this methodology is based on classical design method, but takes advantages of HDL's development in IC simulation and FPGA evolution in terms of integration capabilities, low cost and re-programmability using static SRAM technology for prototyping. Optimized in terms of time and cost, the methodology is presented in Fig. 4.



System requirements
Power circuit diagram (using MATLAB/SIMULINK)
FPGA controller design (Xilinx system generator)
Complete drive simulation (simulink)
Performance OK?
No
Yes
Simulation
VHDL code for communication
Automatic code generation (xilinx system generator)
Complete FPGA simulation (if needed)
Code fusion
Synthesis and implementation (Model sim and xilinx tool)
Code ready for download in FPGA
Implemention

Fig. 4: Top-down HDL's based methodology for rapid prototyping of digital integrated drive control

The design of IC is divided in two parts called "HDL software development" and "hardware prototyping". The HDL software development includes two stages: behavioral stage and structural stage. The hardware prototyping is a physical stage.

Behavioral stage: In the behavioral stage, the studied drive control specifications are first written at system level, using an HDL. The most common used HDL's are Verilog or Very high speed integrated circuit Hardware Description Language (VHDL). We have chosen VHDL for digital control block description of drive up to physical stage. VHDL allows the possibility of modeling the digital control behavior, regardless of target technology and implementation (Riesgo *et al*., 1999). However, in drive control case, power and analog elements of the AC drive (Fig. 1) must be described in order to functional validation of the VHDL described digital control, by analog-digital simulation. VHDL is not suited for modeling or describing power or analog elements, where continuous time is required. MATLAB/ Simulink Software tool is used for the modeling and simulation of control algorithms.

Power System Blockset are used for the modeling and simulation of power system .The same models will be reused all over this methodology to check the IC functionality.

After functional validation at system level description, integrated control block digital properties must be studied and chosen by the designer. Thus, data binary format (word length, fixed or floating point), mathematical and numerical calculation methods (mathematical operators, integration) or quantification effects due to used binary format and external Analog Digital Converters (ADC's) format, must be defined. In drive control, these choices are based on accuracy of estimated digital values. These parameters must be also defined to minimize digital estimation errors, which could decrease control performances. At end of this behavioral stage, all digital properties must be defined, after functional validation by mixed (analog-digital) simulations.

Structural stage: At this stage, the digital control must be partitioned into functional blocks. Each block is described into a lower abstraction level as in precedent stage, taking into account the digital properties previously defined. The description of the digital IC is closer to hardware implementation. The final goal of this stage is to obtain a Register Transfer Logic (RTL) model, which must be checked against by mixed simulation, reusing high level analog MATLAB/ Simulink models written in behavioral stage for power and analog elements. The major results of this stage are, first, the functionality of the VHDL described control and secondly, the additional information, such as length of the operations in term of clock cycles number. The final RTL digital control model must be written according to the synthesis aspect, optimizing hardware resources and timing of the final digital control IC (Bhasker, 1996).

Physical stage: The RTL VHDL model, obtained at the end of structural stage, is independent of the target technology. The rapid hardware prototyping used method changes depending on the aim of the final implementation. The evolution of FPGA in term of integration capabilities, low cost and re-programmability using static SRAM technology leads the designer to think of FPGA prototyping. For these reasons, to reduce lead time and initial prototyping cost, we propose in methodology to implement and validate, first, on an experimental test bench, a FPGA prototype, programmed with the RTL final description of the structural stage. Another interest of FPGA prototyping is component re-programmability.

VLSI DESIGN

A DTC controller for an induction motor drive has been designed using the proposed prototyping platform in order to evaluate the performance of the approach. Figure 5 shows a Simulink diagram that represents the drive system.

The power section on the top is built with PSB blocks. The controller is built with Xilinx SG blocks. The channels of the A-to-D converter are represented by 4 special blocks. Figure 6 shows the diagram of the DTC controller built entirely with SG blocks. Each function on the original control algorithm has been rewritten to be suited for FPGA. This procedure is usually straight forward because SG functions are replicas of Simulink functions. However, in order to exploit the full advantage of FPGAs, some functions in the control algorithm have to be modified. For example, the operation *arctan* is traditionally used in DSP implementation to determine the rotor flux vector position. In an FPGA, this operation would require a lot of silicon space so it was replaced by several comparisons which can be implemented using much less resources on the FPGA. For sine or cosine functions, it is simpler to implement using lookup tables in FPGAs (Dubey *et al*., 2007).

Since, the FPGA's structure is completely configurable, the prototyping platform can be readily modified to suit new requirements (addition of measured variables, modification of communication system, modification of control configuration, etc.). The modifications are done at high level (Xilinx SG blocks and VHDL language) (Perry, 2004). The presented control strategy was implemented in Xilinx Spartan 3E and the target device is XC3S100E. For simulations the Mentor Graphics ModelsimXE III 6.2 g were used. The overall simulation of the DTC strategy is illustrated in Fig. 7.

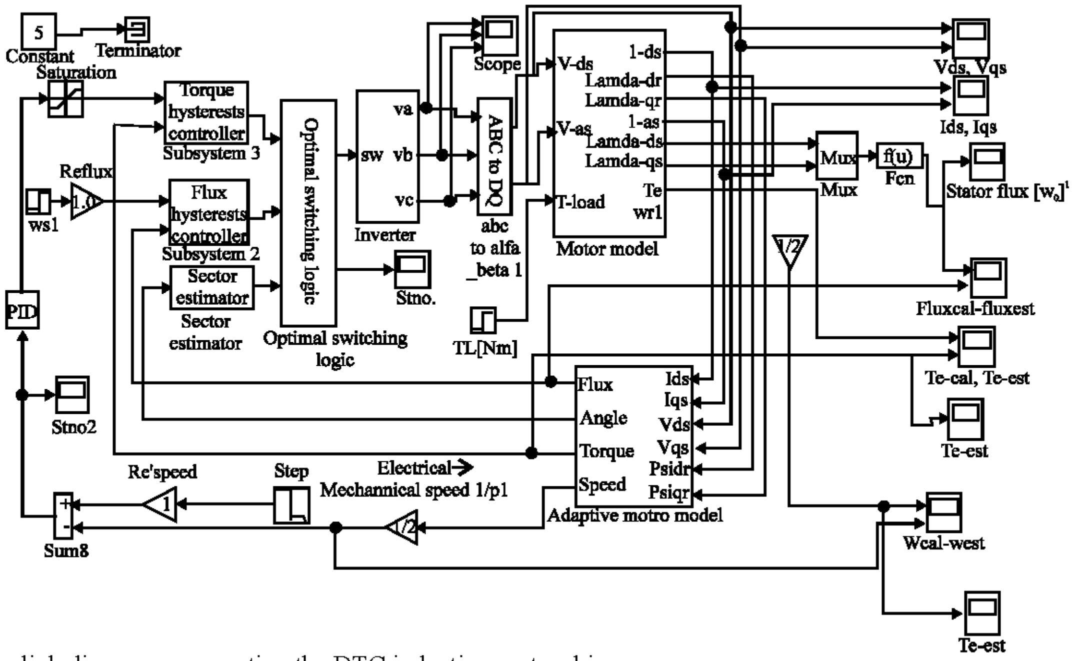


Fig. 5: Simulink diagram representing the DTC induction motor drives

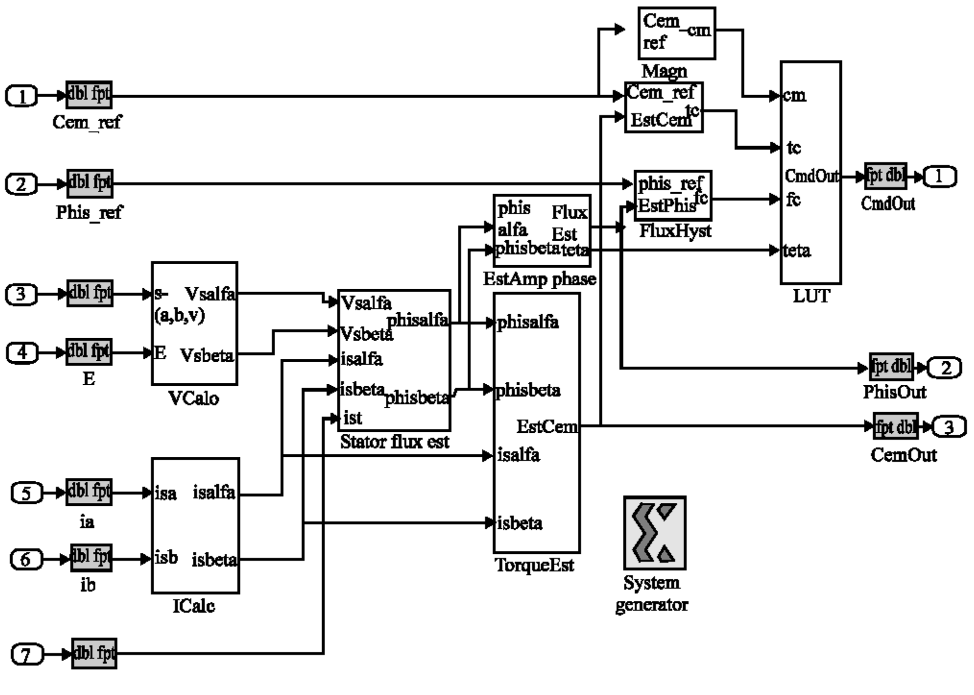
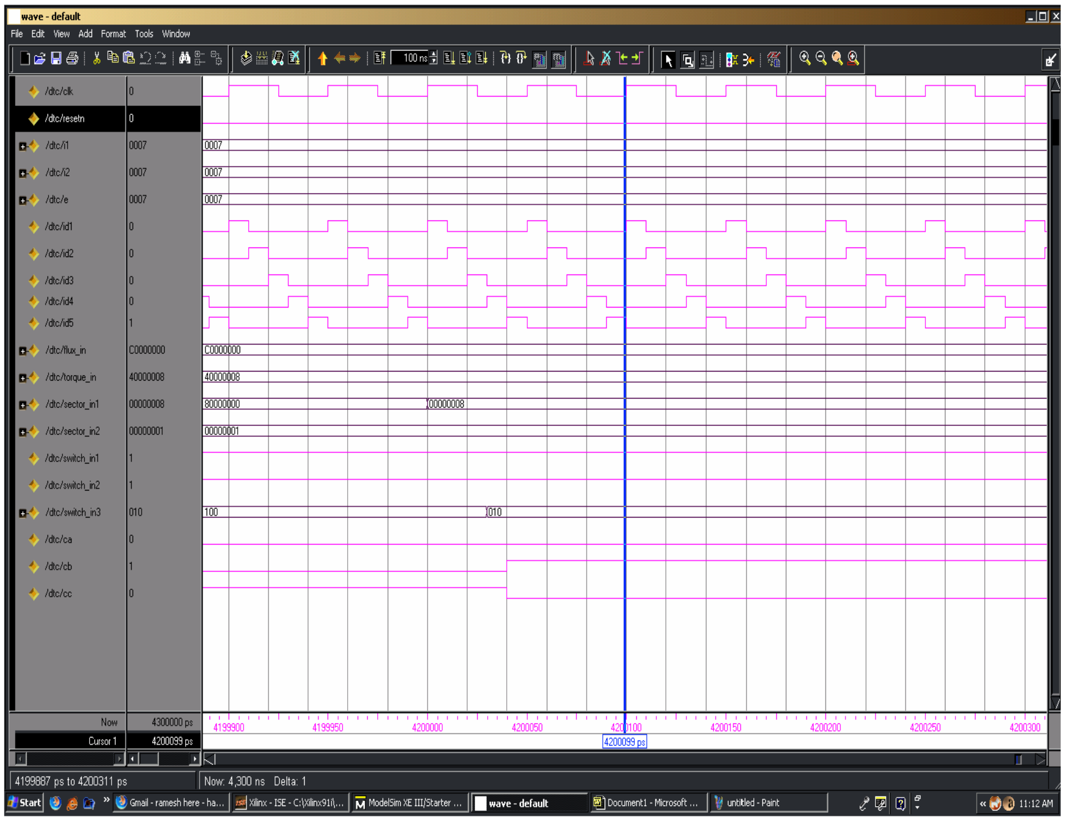


Fig. 6: Diagram of the DTC controller (built with Xilinx SG blocks)

The resulting (RTL) model of estimator architecture, presenting the VHDL entity, is shown in Fig. 8.

VHDL code for the DTC module:

```
library ieee;
use ieee.std_logic_1164.all;
entity dtc is --external view of entire Direct
 torque control circuit
port(i1,i2,E:in std_logic_vector(15 downto 0);
torque,flux:in std_logic_vector(30 downto 0);
        resetn,clk:in std_logic;
        ca,cb,cc:inout std_logic);
```



/dtc/clk	0
/dtc/resetn	0
/dtc/i1	0007
/dtc/i2	0007
/dtc/e	0007
/dtc/id1	0
/dtc/id2	0
/dtc/id3	0
/dtc/id4	0
/dtc/id5	1
/dtc/flux_in	C0000000
/dtc/torque_in	40000008
/dtc/sector_in1	00000008
/dtc/sector_in2	00000001
/dtc/switch_in1	1
/dtc/switch_in2	1
/dtc/switch_in3	010
/dtc/ca	0
/dtc/cb	1
/dtc/cc	0

```
Now      4300000 ps
Cursor 1 4200099 ps
4199887 ps to 4200311 ps     Now: 4,300 ns   Delta: 1
```

Fig. 7: Overall simulation of the DTC strategy

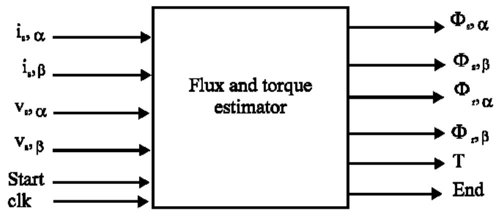


Fig. 8: RTL model of the estimator architecture

```
end dtc;
architecture x of dtc is
component control
       port(clk,resetn:in std_logic;
       id1,id2,id3,id4,id5:out std_logic);
end component ;
component motor
port (i1,i2,E:in std_logic_vector(15 downto 0);
   resetn,id1,id2,id3:in std_logic;
 end component;
component torque_compare
port(torque_in,torque:in  std_logic_vector(30  downto  0);
                                          id4:in std_logic;
comparator_out:inout std_logic);
end component;
component flux_compare
port(flux_in,flux:in std_logic_vector(30 downto 0);
                          id4:in std_logic;
                          comparator_out:inout std_logic);
end component;
component SWITCH_TABLE
PORT(flux_in,torque_in,id5:IN STD_LOGIC;
    sector:IN STD_LOGIC_VECTOR(2 DOWNTO 0);
    ca_out,cb_out,cc_out:inOUT STD_LOGIC);
END component;
component sector6
port(lambda_alpha,lambda_beta:in std_logic_vector
(30 downto 0);
    id4:in std_logic;
                   sector:inout std_logic_vector(2 downto 0));

end component;
signal switch_in1,switch_in2:std_logic;
signal switch_in3:std_logic_vector(2 downto 0);

signal id1,id2,id3,id4,id5:std_logic;
```

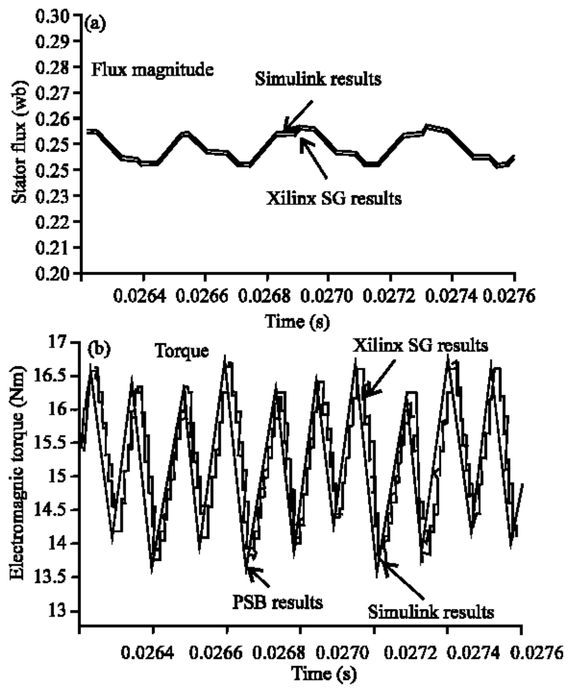


Fig. 9: Comparison between Xilinx SG and Simulink simulation results

```
begin
motor_model:motor port map
(i1=>          i1,i2=> i2,E=> E,resetn=> resetn,id1  =>
id1, id2
=> id2,id3       =>id3,ca=> ca, cb=> cb, cc=>cc,
torque=> torque_in,lambda_mod     => flux_in,
lambda_alpha=> sector_in1,lambda_beta => sector_in2);
……………..
controlling:control port map          (clk=>clk, resetn=>
resetn,id1       =>id1,
id2=>id2,id3=>id3,id4=>id4,          id5=>id5);
end;
```

Figure 9 shows the results (stator flux and motor torque) obtained with Xilinx SG model compared to that obtained with standard Simulink model. A slight difference was noted which is due to time step delay in the ADC and fixed point format used in FPGA. This difference is considered negligible and does not affect the controller performance. Figure 10 shows the simulation results of the electromagnetic torque simulated in the power block of the global behavioral simulation. The designed controller was simulated to control 3 kW, 2 pole, 230/400 V, 50Hz induction motor fed by an IGBT inverter.

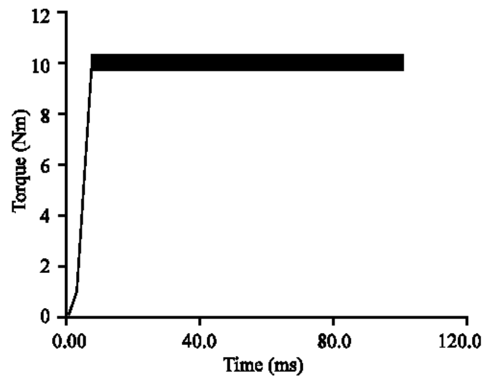


Fig. 10: Estimated torque simulation result

CONCLUSION

The design process based on HDL's has been presented in this study for motor drive application. The main steps in the design process have been pointed out, as well as the tools involved in each stage. In spite of the drawbacks that are found in the method, the HDL-based design is good solution to deal with the complexity of current drive system. So far, it is restricted to digital systems, but some effort is being made to extend some of these concepts to analog electronics. The forecast for the use of this process is very promising. A large group of designers is moving from old methodologies to those based on HDL's. Some companies and institutions are starting to demand VLSI documentation for all their designed systems. The FPGA-based development brings the possibility to add more stringent control strategy, in terms of processing time, to algorithms such as over sampling control or predictive control in order to increase performances.

ACKNOWLEDGMENT

The authors would like to thanks Xilinx Corporation and Altera Corporation for providing the technical support during the simulation stage. We would also like to thank the management of Dr. MGR University, Chennai, India for its financial support.

REFERENCES

Altera Corporation, 2003. Avalon Bus Specification reference Manual, Document Version 23. http://www.altera.com/literature/manual/mnl_avalon_bus.pdf.

Bhasker, J., 1996. A VHDL synthesis prime. Ed. Star Galaxy Publishing.

Chapuis, Y.A., C. Girerd and F. Aubepart, J.P. Blonde and F. Braun, 1998. Quantization problem analysis on ASIC-based Direct Torque Control of an induction machine. In: Proc. IEEE-IECON. Conf., pp: 1527-1532.

Cirstea, M., A. Dinu, M. McCormick and D. Nicula, 2000. A VHDL success story: Electric drive system using neural controller. In: Proc. VHDL Int. Users Forum Fall Workshop, pp: 118-122.

Dubey, R., P. Agarwal and M.K. Vasantha, 2007. Programmable logic devices for motion control-A review. IEEE Trans. Ind. Electron., 54 (1): 559-566.

Free access IP Module Internet site http://www. opencores.org.

Gray, J., FPGA CPU Links. http://www. fpga_cpu .org.

IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1993.

Kebbati, K., Y.A. Chapuis and F. Braun, 2001. IP modules for motor control FPGA/ASIC integration. In: Proc. IFIP. Conf., pp: 385-390.

Kiel, E. Lenze and Aerzen, 2002. Control Electronics in Drive Systems Micro Controller, DSPs, FPGAs, ASICs from State-Of-Art to Future Trends. PCIM, CD-ROM.

Le-Huy, H., 1994. Microprocessors and digital IC's for Motion Control: Experience of the use of ASIC methods in a motor control application. Invited Paper, Proc. IEEE., 82 (8): 1140-1163.

Perry, D.L., 2004. VHDL. New York: McGraw-Hill.

Pimentel, J.C.G. and H. Le Huy, 2000. A VHDL-Based Methodology to Develop High Performance Servo Drivers. IEEE-IAS Conf. Proc., Roma, Italy, pp: 1505-1512,

Riesgo, T., Y. Torroja and E. de la Torre, 1999. Design methodologies based on hardware description languages. IEEE. Trans. Industrial Elect., 46 (1): 3-12.

Tazi, K.K., E. Monmasson and J.P. Louis, 2000. The use of FPGAs to build an AC drive specific hardware-software library in the case of Park's transformation. In: Proc. EPE-PEMC. Conf., 7: 181-185.

Trimberger, T., J.A. Rowson and C.R. Lang and J.P Gray, 1981. A structured design methodology and associated software tools. In: Proceeding of the IECS.

Wolf, W., 1993. FPGA-Based System Design. Englewood Cliffs, NJ: Prentice-Hall, 2004.

Xilinx Corporation, 2006. Programmable, Logic Design, Quick Start Handbook. http://www.xilinx.com/ company/about/programmable.html.