Design and Prototyping of Direct Torque Control of Induction Motors in FPGAs

Sandro Ferreira, Felipe Haffner, Luís Fernando Pereira, Fernando Moraes
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - Prédio 30 / BLOCO 4 - 90619-900 - Porto Alegre – RS – BRASIL
sandro@ee.pucrs.br, jfelipe@ee.pucrs.br, pereira@ee.pucrs.br, moraes@inf.pucrs.br

Abstract

This work presents an implementation of a Direct Torque Control (DTC) strategy, which is used to control induction motors. Following a tendency in the research area, the algorithm proposed is implemented in an unique FPGA substrate, which allows for a faster validation and simplifies the control structure. A binary format is used with a variable word-size approach, which permits to reduce truncation through the calculation processes, resulting in smaller errors without increasing excessively hardware area in the prototyping step. Matlab is used to validate the algorithm. The calculation error between Matlab's double precision representation and the proposed alternative is small and can be neglected in DTC control.

1. Introduction

Induction motors have many advantages when compared to direct current (DC) motors. The formers do not necessarily present mechanical commutators, which restrict power and velocity, greatly increasing machine maintenance costs. Nevertheless, DC drives have dominated the field of adjustable speed electrical drives since the beginning of the last century due to the high dynamic performance which they permit [1].

Induction machine control was only possible after the development of efficient and powerful switching components. Besides, it involves complex equations to be solved during operation whose implementation has always been associated with microelectronic advances. For instance, the first control theory was presented in 1971 [2] and implemented ten years later with the advent of microprocessors.

Direct Torque Control (DTC) was formulated by Takahashi (1986) [3] and Depenbrock (1988) [4] and introduced as a viable alternative for the control of induction machines. Allowing for a performance even superior to DC machine drives, it was soon accepted by industry and is slowly becoming a standard, together with Field Oriented Control techniques.

DTC has usually been implemented using DSP processors associated with ASIC components [5]. This was certainly due to DSP availability as the most powerful tool at the moment.

The recent evolution in ASIC technology has given rise to the implementation of DTC in an unique integrated component ([6] – [9]), simplifying algorithm development and validation.

The main contribution of this work is an implementation of DTC drive in FPGA using a fixed-point arithmetic with a variable word-size approach. This is proposed as an alternative to 16 or 32-bit choices used before ([6] – [9]). A separation of the algorithm in functional blocks is used to simplify validation task which was accomplished in comparison with Matlab results.

The paper is organized as follows. In the next section, the algorithm is presented and discussed. Section 3 proposes an architecture to implement the algorithm. Section 4 presents an initial validation with architecture simulation results compared with Matlab results. Finally, in section 5, some conclusions are presented.

2. Algorithm

Figure 1 presents the topology of a DTC drive:

- *Induction motor* is a system to be voltage controlled. *The inverter* applies three-phase voltage signals determined by changes on the *switching keys*, which are calculated by the algorithm;
- *Switching table* calculates the switching keys according to a strategy as a function of the torque and flux comparators and the spatial sector in which the stator flux is lying at calculation time;

Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI'03)
0-7695-2009-X/03 $17.00 © 2003 **IEEE**

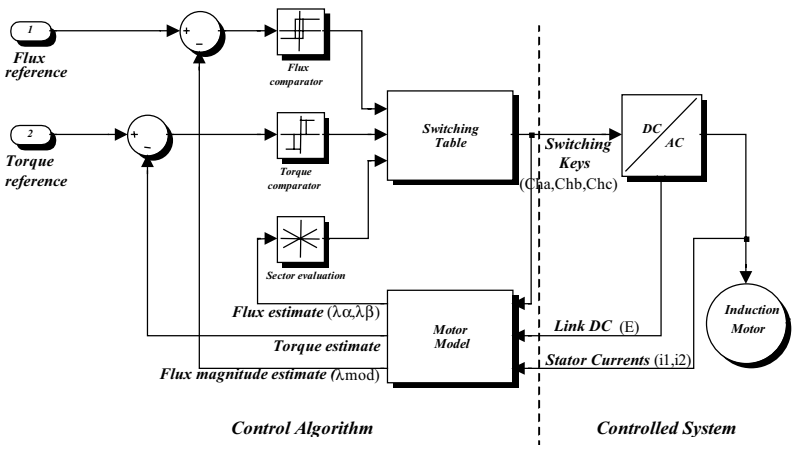


Figure 1. DTC strategy.

- *Motor model* is a mathematical model used to calculate flux, currents and voltages referred to the stator in αβ stationary coordinates and electrical torque. It is the most complex part of the algorithm, involving multiplications and a square root evaluation. These operations are computed serially in order to reduce hardware area in the prototyping step;
- *Link DC* (E) and *stator currents* (i1 and i2) are determined by three serial A/D converters, which correspond to the slower part of the algorithm, taking 25 μs to perform the operation. The conversions are executed simultaneously;
- *Torque and flux comparators* are two and three-level relays with hysteresis – they are used to compare torque and flux references with their actual values;
- *Sector evaluation* – is a function that calculates the position of stator flux vector in αβ coordinates in a plane. In this case, the plane is divided into 6 regions or *sectors* corresponding to the voltage vectors that can be applied by the inverter.

The motor model is composed by the equations presented in Figure 2.

In Figure 2, Vα, Vβ, iα, iβ, λα, λβ are coordinate components of stator voltage, current and flux respectively. $\lambda\alpha_{OLD}$ and $\lambda\beta_{OLD}$ are values of stator flux calculated in the previous instant. E is the *link DC* voltage; *i1* and *i2* are stator current values of two of the three power lines to which the motor is linked. Cha, Chb and Chc are the switching inverter commands applied in the previous instant. Rs is a stator resistance estimate evaluated off-line. Ta is the A/D converter sampling time.

Equations 1 and 2 transform current and voltage components measured in a 16-bit value by the A/D into αβ components which are adequate to DTC algorithm. Voltage components need switching commands applied in the previous instant to be calculated. Equation 3, corresponds to an integration using Forward Euler Method. Sampling time (Ts) precision is critical to DTC implementation since it affects flux estimation. Equation 4 estimates electrical torque, which is directly controlled as a main proposition of DTC technique. Equation 5 calculates flux magnitude, which is used in the flux control loop.

1. Stator Voltages in αβ coordinates

$V\alpha = E/3*(2*Cha - Chb - Chc);$

$V\beta = sqrt(3)/3*E*(Chb - Chc);$

2. Stator Currents in αβ coordinates

$i\alpha = i1;$

$i\beta = sqrt(3)/3*(i1 + 2*i2);$

3. Stator Flux estimation

$\lambda\alpha = \lambda\alpha_{OLD} + Ts*(V\alpha - Rs*i\alpha);$

$\lambda\beta = \lambda\beta_{OLD} + Ts*(V\beta - Rs*i\beta);$

4. Torque estimation

$torque = 3/4*P*(i\beta*\lambda\alpha - i\alpha*\lambda\beta);$

5. Stator Flux magnitude determination

$\lambda mod = sqrt(\lambda\alpha\text{^}2 + \lambda\beta\text{^}2);$

Figure 2. Equations modeling the motor behavior.

3. DTC Architecture

DTC algorithm is implemented in an architecture composed by five main blocks: motor model, flux comparator, sector evaluation, torque comparator and switching table (Figure 3). The available processing time is dictated by the A/D converters, corresponding to 25 μs. This time interval is partitioned into five time slots, allowing for the processing of input samples in a pipeline structure.

The motor model module uses time slots 1 to 3. In the first time slot (ld1), 16-bit samples are read from the A/D converters. Modules flux comparator, sector evaluation and torque comparator are processed in parallel in the

fourth time slot, while last time slot is used to compute the switching table (Cha, Chb, Chc).

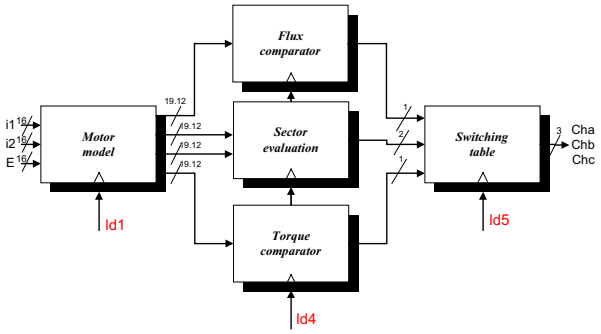


Figure 3. Architecture implementation.

Motor model module has three 16-bit inputs supplied by A/D converters: i1, i2 and E; and produces four outputs: torque, λα, λβ and λmod (Figure 4).

Equations modeling motor behavior are implemented according to the architecture presented in Figure 4. As can be observed, complex mathematical operations are performed such as multiplications and a square root. These operations are implemented using serial algorithms to reduce the final circuit area. Between the second and the third time slots (ld2 and ld3), four multiplications are executed in parallel. The same parallelism can be observed through the rest of the architecture. This inherent hardware parallelism would not be possible in software implementations showing the superior performance of the hardware solution when compared to the software one.

Flux and torque comparators are two and three-level relais with hysteresis implemented using if-then-else instructions. Flux magnitude and torque values provided by motor model module are truncated to 31 bits to be compared with reference values stored previously.

Sector evaluation is a module that receives stator flux components as inputs and determinates the position of the flux vector in a plane divided into six sectors denominated sectors 1 to 6. To determinate the position of stator flux, magnitude is compared with projection components in the axes α and β.

In the algorithm implementation, many digital properties have to be considered. Characteristics such as quantization, sampling, and adopted binary format are key performance factors in the control process.

3.1. Quantization

In ASIC implementation, the word size is critical. A larger word reduces quantization effects but increases silicon area and affects costs. On the other hand, a smaller word may affect precision, increasing control error or even unstabilizing the system. A huge silicon area may also implicate in technological constraints.

The quantization problem in DTC has already been studied by Chapuis (1998) [10]. According to him, recent DSP-based control drives for AC motors use 32-bit fixed-point format while 16-bit fixed-point format is not sufficient to achieve good DTC implementation. A different alternative was used: a fixed-point format with variable word size. Architecture was implemented with words starting with 16 bits and increasing according to the mathematical operations performed to avoid reduction of precision. This alternative was adopted due to the limited precision given by the A/D converters used to measure DC link and stator currents. Each operator in Figure 4 changes the word size to avoid precision errors. For instance, observe word size at the input of the square root block: 62 bits; and at the output of the same block: 31 bits.

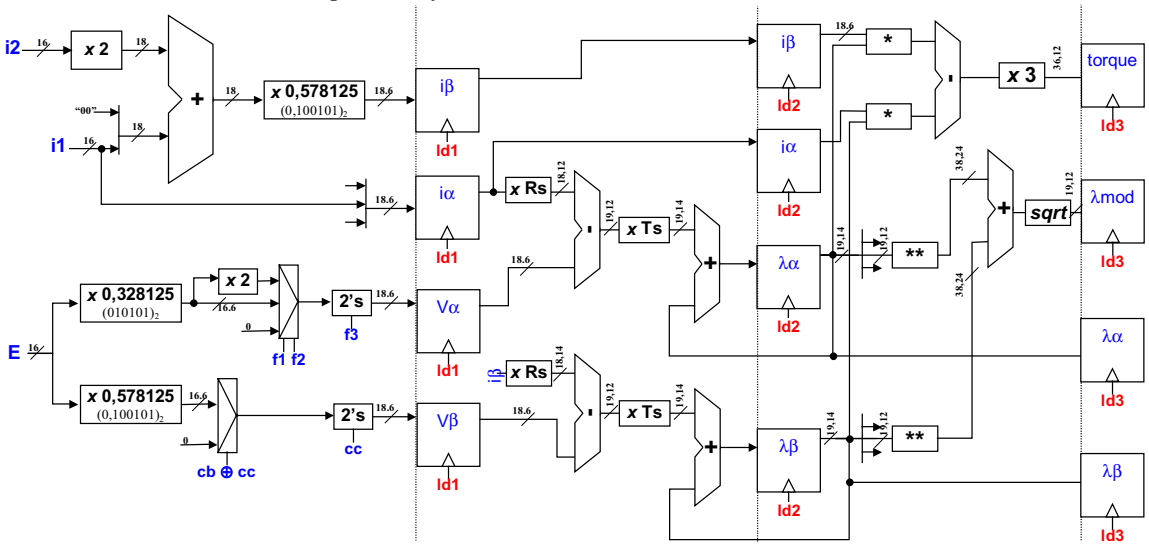
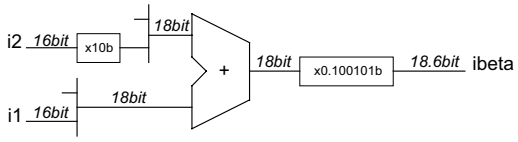


Figure 4. Motor model architecture.

In Figure 4, $f1 = ca \oplus (cb.cc)$ and $f2 = cb \oplus cc$.

Figure 5 is an extract of Figure 4 containing only the second equation of the motor model (*ibeta*). Variable word sizing can be easily observed. For instance, *i2* starts with 16 bits and is multiplied by 2 (shift operation). To avoid overflow, the result is saved in an 18-bit word. This product is added to *i1* and the result is multiplied by sqrt(3)/3 ($\approx 0.578125_{10}$, $\approx 0.100101_2$) which results in an 18.6 bit word (18-bit integer part and 6-bit fractional part). In this case, the 6 fractional bits are not discarded to avoid lost of precision, resulting in a fixed-point word with 24 bits.



Equivalent VHDL code:

```
architecture calc_is of calc_is is
        signal sm, i2_1,i1_1: STD_LOGIC_VECTOR (17 downto 0);
        signal i2_2,i2_3,i2_4: STD_LOGIC_VECTOR (23 downto 0);
begin
  -- i2*2 multiplied by 2 with signal extension
  i2_1 <= i2(15)& i2 & '0';

  -- i1 with signal extension
  i1_1 <= i1(15) & i1(15) & i1;

  -- sum
  sm <= i2_1 + i1_1;

  -- multiplication by 0.100101
  i2_2 <=sm(17) & sm(17) &sm(17) & sm(17) &sm(17) & sm(17) & sm;
  i2_3 <=sm(17) & sm(17) & sm(17) & sm(17) & sm & "00";
  i2_4 <=sm(17) & sm & "00000";
  ibeta <= i2_2+i2_3+i2_4;
……
end cal_is;
```

Figure 5. Variable word-size implementation example.

A critical part in the architecture is the $\lambda\alpha$, $\lambda\beta$ calculation, since a feedback is executed to perform the integration. This operation can induce errors and overflow can easily occur if *Ts* value is not properly scaled. Scaling factor is necessary because representing 25 µs with good precision would need a minimum of 24 bits. In the first implementation, Ts was assumed to be 0.01_2 (equals to 0.25), which is 10000 bigger than its correct value. This scaling factor proved to be inefficient because flux calculation was overflowed due to the integration process, which involves accumulation of values. Therefore, another scale factor was adopted.

The adopted scale factor was 0.025. Figure 6 presents a comparison between flux estimate errors. Relative errors obtained when compared to Matlab double precision estimates can be seen in figures (B) and (C). Two representations were tested for Ts: 0.0251464844375 (0.000001100110) and 0.02499961853027 (0.00000110011001100110011). After the accumulation, results were truncated to 33 bits (19.14b) to avoid excessive increase of fixed-point representation. It corresponds to a reduction of 10 and 17 bits respectively (Figure 4).

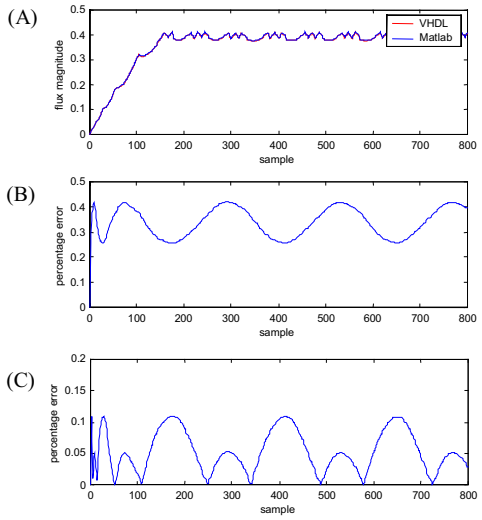


Figure 6. (A) Comparison between Matlab (double precision) and VHDL (fixed point) estimation of stator flux magnitude; (B) error with Ts=0.000001100110b ; (C) error with Ts=0.00000110011001100110011b.

3.2. Sampling

According to [3][4], sampling time (Ts) is of crucial importance to the algorithm, specially when flux estimation process is concerned. The sampling time was limited to 25 µs due to the A/D characteristics. All the mathematical operations involved in the model were performed within the sampling time.

3.3. Binary format

Two-complement fixed-point format was used since it simplifies implementation of operations like sum, subtraction and multiplication by scalar.

4. Function Validation

Function validation was performed comparing the VHDL simulation results (Active HDL – www.aldec.com) to Matlab results. Modeling equations presented in Figure 2 were simulated using Matlab with double precision (64-bit float arithmetic) and compared with VHDL simulated results using variable size words.

In Figures 7 to 9, percentage error graphs showing Matlab and VHDL comparative results are presented. Error peaks occasioned by small values of flux and torque in the error function denominator can be observed. Even in these cases, error is smaller than 0.3 per cent.

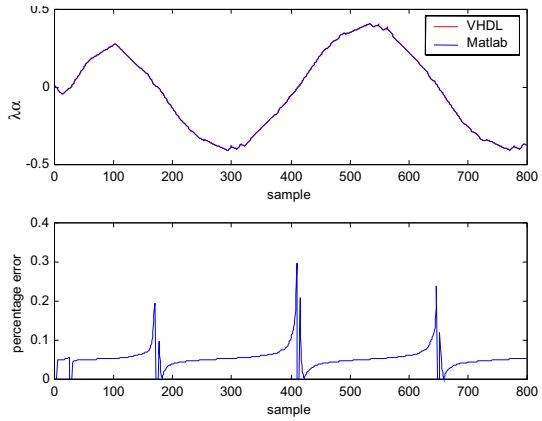


Figure 7. Comparison between full-precision λα Matlab and variable word-size VHDL estimation.

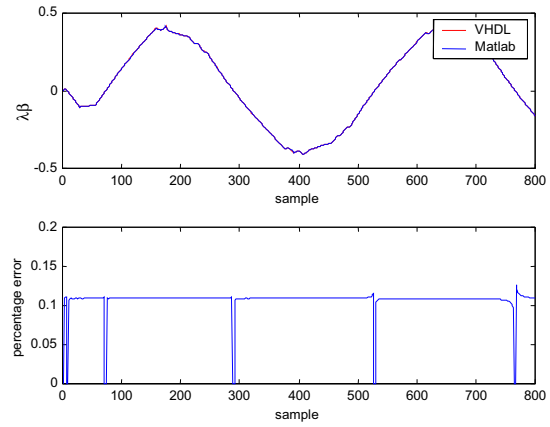


Figure 8. Comparison between full-precision λβ Matlab estimation and variable word-size VHDL estimation.

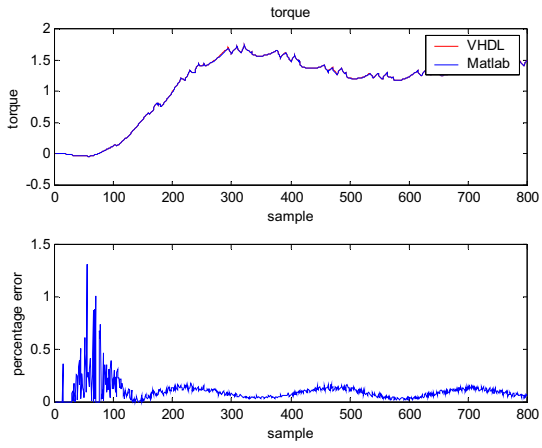


Figure 9. Comparison between double torque Matlab estimation and variable word-size VHDL estimation.

Figures 10 to 12 present the outcome of DTC algorithm, or else the switching keys that are applied to the induction motor (cha, chb, chc). Absolute error figures show the good approximation between Matlab and VHDL results.

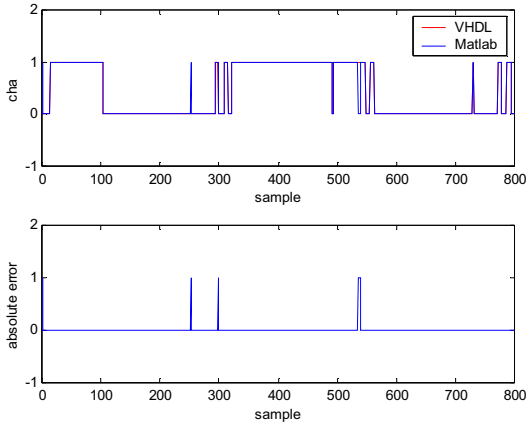


Figure 10. Comparison between Matlab and VHDL estimates of Cha command.

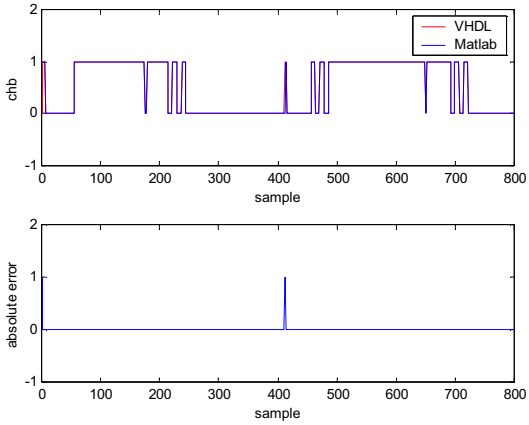


Figure 11. Comparison between Matlab and VHDL estimates of Chb command.

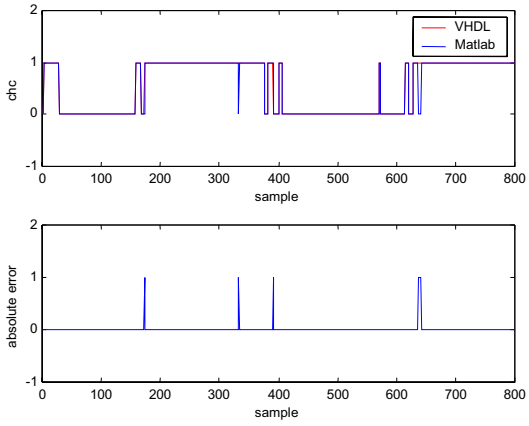


Figure 12. Comparison between Matlab and VHDL estimates of Chc command.

5. Conclusions

DTC implementation in VHDL is presented using fixed-point arithmetic to minimize quantization errors. The architecture proposed is written in synthesizable VHDL (RTL style). The resulting area usage for an EP20K200 device (Altera) is 4100 logic cells. A minimum error was obtained between the fixed-point representation proposed and Matlab's double representation used. The integrated solution does not utilize microcontrollers, which allows new algorithms to be quickly implemented, simplifying test and validation of new control strategies.

6. References

[1] Leonard, W., "30 Years Space Vectors, 20 Years Field Orientation, 10 Years Digital Signal Processing with Controlled AC-Drives, a Review", *EPE Journal*, vol.1, nº 1, pp. 13-20, July 1991.

[2] Blaschke, F., "A New Method for the Structural Decoupling of A.C. Induction Machines", in *Conf. Rec. IFAC*, Dusseldorf, Germany, Oct, 1971, pp.1-15.

[3] Takahashi, I., Noguchi, T., "A New Quick Response And High-Efficiency Control Strategy of an Induction Motor", *IEEE IAS Ann. Mtg*., pp. 496 – 502, 1985.

[4] Depenbrock, M., Baader, U., "Direct Self Control (DSC) of Inverters-Fed Induction Machine: A basis for Speed Control without Speed Measurement", *IEEE Transactions on Industry Applications,* Vol. 28, pp. 581-588, May/June 1992.

[5] Buja, G., Casadei, D., "Tutorial 2: The Direct Torque Control of Induction Motor Drives", *ISIE*, 1997.

[6] Aubépart, F., Poure, P., Girerd, C., Chapuis, Y. A., Braun, F., "Design and Simulation of ASIC-Based System Control: Aplication to Direct Torque Control of Induction Machine", *ISIE'99* - Bled, Slovenia, 1999, pp. 1250 - 1255.

[7] Poure, P., Aubépart, F., Braun, F., "A Design Methodology for Hardware Prototyping of Integrated AC Drive Control: Application to Direct Torque Control of an Induction Machine", *Proceedings. 11th International Workshop on Rapid System Prototyping, 2000. RSP 2000*, pp. 90 –95.

[8] Se Jim Kim, Ho Jae Lee, Sang Koon Kim, Young Ahn Kwon, "ASIC Design for DTC Based Speed Control of Induction Motor", *ISIE'01* - Pusan, Korea, 2001, pp. 956 - 961.

[9] Aubépart, F., Poure, P., Braun, F., "VLSI Design Approach of Complex Motor Control, Case of Direct Torque Control of AC Machine", *The 7th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2000*., Volume: 2, 2000 pp. 814 –817.

[10] Chapuis, Y. A., Girerd, C., Aubépart, F., Blondé, J. P., Braun, F., "Quantization Problem Analysis on ASIC-Based Diret Torque Control of an Induction Machine*", IECON '98.* Volume: 3, 1998, pp. 1527 - 1532.