HAL
open science

Towards the System-on-Chip Realization of a Sensorless Vector Controller with Microsecond-order Computation Time

Rachid Beguenane, Jean-Gabriel Mailloux, Stephane Simard, Arnaud Tisserand

To cite this version:

Rachid Beguenane, Jean-Gabriel Mailloux, Stephane Simard, Arnaud Tisserand. Towards the System-on-Chip Realization of a Sensorless Vector Controller with Microsecond-order Computation Time. CCECE'06: Canadian Conference on Electrical and Computer Engineering, May 2006, pp.908-912, 10.1109/CCECE.2006.277332 . lirmm-00125485

HAL Id: lirmm-00125485

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00125485

Submitted on 19 Jan 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards the System-on-Chip Realization of a Sensorless Vector Controller with Microsecond-order Computation Time

Rachid Beguenane*, Jean-Gabriel Mailloux*, Stéphane Simard*, and Arnaud Tisserand†
*Groupe ERMETIS, Département des Sciences appliquées
Université du Québec à Chicoutimi
Chicoutimi (QC), G7H 2B1, CANADA
rbeguena@uqac.ca, jean-gabriel_mailloux@uqac.ca, s.simard@ieee.org
† LIRMM, CNRS-Univ. Montpellier II
161 rue Ada. F-34392 Montpellier, FRANCE
arnaud.tisserand@lirmm.fr

***Abstract*— The aim of this research is to implement sensorless vector control algorithms on a single, eventually reconfigurable, chip, with a computation timing constraint of, at most, 1-6 microseconds, and a concern for implementation cost. In this article, we discuss the implementation problems and tradeoffs involved in meeting these goals on Field-Programmable Gate Arrays (FPGAs). To be able to fit a complete induction motor vector controller on a single, inexpensive FPGA chip, we estimate the area/time requirements of each module involved in sensorless vector control. We discuss, in particular, the tradeoffs of implementing the key modules, the speed and flux observers and the Clarke and Park transformations. The speed and flux observers here under consideration are extended Kalman filter-based.**

I. Introduction

Technological progress depends more and more on miniaturization and increasingly fast and powerful computing systems, and we witness a true revolution in microelectronics. In almost every field, significant progress now greatly depends on advances in nanotechnologies and their applications. The design of lighter, less cumbersome, and more economic application-specific computing processors becomes required in order to reach increasingly demanding performances.

The very fast evolution of CMOS integrated circuit fabrication technologies already makes it possible to design complete digital systems integrated on the same chip.

The reconfigurable chips known as FPGAs (Field-Programmable Gate Arrays) currently on the market are manufactured at a level of integration of around 90 nanometers or less, and usually comprise several millions of gates on the same chip. They offer considerable advantages for accelerating the time to market, and reduce the development and production costs.

In field of electric motor control, with which we are concerned in this research, the real-time computing capacity of traditional approaches using PC computers and off-the-shelf digital signal processors (DSPs), is largely superseded. One must now turn to nanotechnologies in order to obtain adequate hardware acceleration. FPGAs, light and relatively inexpensive, are usually more powerful than traditional devices, and appear therefore ideal for implementing real-time control systems without involving the considerable costs traditionally related to the design and fabrication of application-specific integrated circuits (ASIC).

The semiconductor industry is trying to design digital signal controllers (DSCs) having a computing time of only a few microseconds for the precise and robust control of electric motors. It would actually be possible of increasing the current operating efficiency of electric motors, now in the order of approximately 40-60%, up to 90%.

The high cost and complexity of the required electronics always constituted a significant impediment to the implementation of complex algorithms within a dynamic of only a few microseconds. The DSC technology, traditionally composed of DSPs coupled to a microcontroller unit or a microprocessor, is presently reaching its physical limits, with a minimal computing time about 6 microseconds for the most minimalistic implementation of vector control using a speed sensor. The challenge to which we attack ourselves here is to realize on a single chip, without using a speed sensor, and in an even shorter lapse of time, the most sophisticated vector control algorithms, where the speed and the flux will not be measured, but estimated by hardware. We will see the complexity of these estimates in the following discussion.

II. Induction Motor Vector Contol

The characteristics of the induction motor are basically non-linear. Vector control, also called *flux directed control*, is the first method which makes it possible to artificially give a certain linearity to the torque control of the induction motor. Speed sensors are however necessited, in general, for the implementation of vector control. This does not pose any problem as long as the induction motor is used for regular motion control using a position or speed encoder, but whenever

1-4244-0038-4 2006 IEEE
CCECE/CCGEI, Ottawa, May 2006

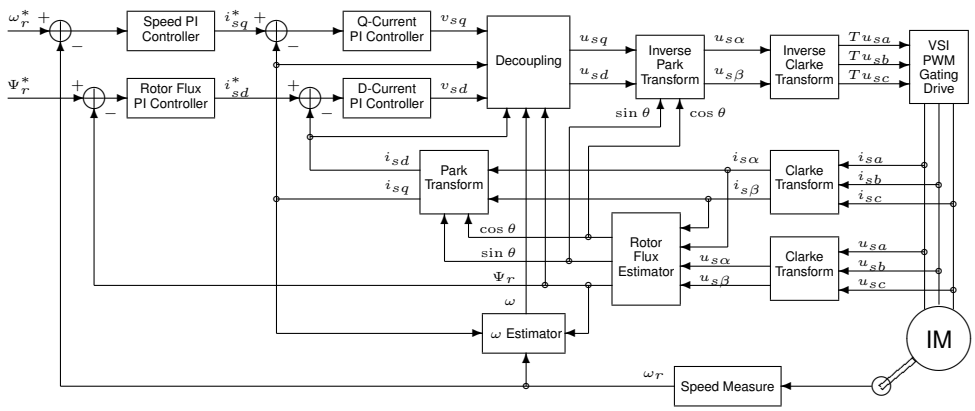

Fig. 1. IM Vector Control Scheme

it is impossible to connect a sensor to the motor shaft, the implementation proves difficult. It is then necessary to carry out vector control without using a speed sensor, with all the difficulties that that poses.

Precise induction motor control requires the independent control of the components of its input current producing the field and the couple, as it is the case with the DC motor. The only theoretical solution which makes it possible to realize such an independence of control consists in breaking up the stator current into its components in a domain or a suitable frame of reference. By taking a synchronously revolving frame of reference, with the space vector of rotor flux as phasor of reference, it is mathematically possible to separate the stator current in two independent components i_{sd} and i_{sq}, respectively controlling the field and the torque of the motor. Decoupling between the effects of the components of the current then makes it possible to simplify the control of the mechanical variables of the drive, and to impose on the motor fast variations of the rotor flux at starting and at constant power.

A. Induction Motor Model in Park Domain

The electromechanical model of the induction motor in the Park reference frame (d, q), known as Park domain, synchronously revolving at speed ω, is stated as follows:

$$u_{sd} = R_s i_{sd} + \sigma L_s \frac{\mathrm{d}}{\mathrm{d}t} i_{sd} \underbrace{-\sigma L_s \omega i_{sq} + \frac{M}{L_r}\frac{\mathrm{d}}{\mathrm{d}t}\Psi_r}_{D_d} \quad (1)$$

$$u_{sq} = R_s i_{sq} + \sigma L_s \frac{\mathrm{d}}{\mathrm{d}t} i_{sq} \underbrace{+\sigma L_s \omega i_{sd} + \frac{M}{L_r}\omega\Psi_r}_{D_q} \quad (2)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\Psi_r = -\beta_r \Psi_r + M\beta_r i_{sd} \; ; \quad \beta_r = \frac{R_r}{L_r} \quad (3)$$

$$\omega = P_p \omega_r + \frac{M\beta_r}{\Psi_r} i_{sq} \quad (4)$$

$$\frac{\mathrm{d}\omega_r}{\mathrm{d}t} = \frac{3}{2} P_p \frac{M}{J L_r} \Psi_r i_{sq} - \frac{f}{J}\omega_r - \frac{T_l}{J} \quad (5)$$

with

u_{sd}, u_{sq}	Stator voltage of d-axis and q-axis
i_{sd}, i_{sq}	Stator current of d-axis and q-axis
Ψ_r	Rotor flux modulus
ω	Angular speed of the (d, q) reference frame
L_s, L_r	Stator and rotor inductances
M	Mutual inductance
R_s, R_r	Stator and rotor resistances
σ	Leakage coefficient of the motor
β_r	Constant: L_r/R_r
P_p	Number of pole pairs
ω_r	Rotor speed, or *angular frequency* (measured or estimated)
J	Inertial momentum
f	Friction coefficient
T_l	Torque load

B. System Block Diagram

In the following, we develop the mathematical expressions for the blocs in Fig. 1. The starred (*) variables are the input references for the PI controllers. In general, ϵ denotes the error signal, and k_p, k_i, the PI controller parameters. The α and β subscripts denote the components of the corresponding variables in the stationnary (α, β) reference frame. The a, b, and c subscripts denote the components of the corresponding variables in the stationnary (a, b, c) reference frame. θ is the angular position of the rotor flux vector, with $\theta = \int \omega \, \mathrm{d}t$.

a) Speed PI Controller:

$$i_{sq}^* = k_{p_v}\epsilon_v + k_{i_v}\int \epsilon_v \, \mathrm{d}t \; ; \quad \epsilon_v = \omega_r^* - \omega_r$$

b) Rotor Flux PI Controller:

$$i_{sd}^* = k_{p_f}\epsilon_f + k_{i_f}\int \epsilon_f \, \mathrm{d}t \; ; \quad \epsilon_f = \Psi_r^* - \Psi_r$$

c) Rotor Flux Estimator:

$$\Psi_r = \sqrt{\Psi_{r\alpha}^2 + \Psi_{r\beta}^2}$$

$$\cos\theta = \frac{\Psi_{r\alpha}}{\Psi_r} \; ; \quad \sin\theta = \frac{\Psi_{r\beta}}{\Psi_r}$$

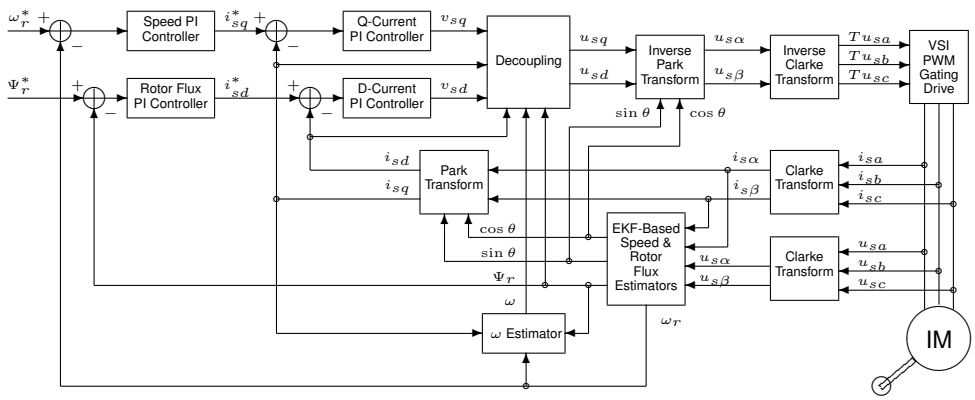

Fig. 2. EKF-based Speed-sensorless IM Vector Control Scheme

with

$$\Psi_{r\alpha} = \frac{L_r}{M}\left(\Psi_{s\alpha} - \sigma L_s i_{s\alpha}\right)$$

$$\Psi_{r\beta} = \frac{L_r}{M}\left(\Psi_{s\beta} - \sigma L_s i_{s\beta}\right)$$

$$\Psi_{s\alpha} = \int \left(u_{s\alpha} - R_s i_{s\alpha}\right)$$

$$\Psi_{s\beta} = \int \left(u_{s\beta} - R_s i_{s\beta}\right)$$

d) Current PI Controller:

$$v_{sd} = k_{p_i}\epsilon_{i_{sd}} + k_{i_i}\int \epsilon_{i_{sd}}\ \mathrm{d}t; \quad \epsilon_{i_{sd}} = \imath^*_{sd} - i_{sd}$$

$$v_{sq} = k_{p_i}\epsilon_{i_{sq}} + k_{i_i}\int \epsilon_{i_{sq}}\ \mathrm{d}t; \quad \epsilon_{i_{sq}} = \imath^*_{sq} - i_{sq}$$

e) Decoupling:

$$u_{sd} = \sigma L_s v_{sd} + D_d\ ; \quad u_{sq} = \sigma L_s v_{sq} + D_q$$

with

$$D_d = -\sigma L_s \omega i_{sq} + \frac{M}{L_r}\frac{\mathrm{d}}{\mathrm{d}t}\Psi_r$$

$$D_q = +\sigma L_s \omega i_{sd} + \frac{M}{L_r}\omega\Psi_r$$

f) Omega (ω) Estimator:

$$\omega = P_p\omega_r + \frac{M\beta_r}{\Psi_r}i_{sq}$$

g) Clarke Transformations:

$$i_{s\alpha} = i_{sa}$$

$$i_{s\beta} = \frac{1}{\sqrt{3}}i_{sa} + \frac{2}{\sqrt{3}}i_{sb}$$

and

$$u_{sa} = u_{s\alpha}$$

$$u_{sb} = -\frac{1}{2}u_{s\alpha} + \frac{\sqrt{3}}{2}u_{s\beta}$$

$$u_{sc} = -\frac{1}{2}u_{s\alpha} - \frac{\sqrt{3}}{2}u_{s\beta}$$

h) Park Transformation:

$$\begin{pmatrix} i_{s_d} \\ i_{s_q} \end{pmatrix} = PT \begin{pmatrix} i_{s_\alpha} \\ i_{s_\beta} \end{pmatrix}$$

with

$$PT = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

i) Inverse Park Transformation:

$$\begin{pmatrix} u_{s_\alpha} \\ u_{s_\beta} \end{pmatrix} = \mathrm{PT}^{-1} \begin{pmatrix} u_{s_d} \\ u_{s_q} \end{pmatrix}$$

with

$$\mathrm{PT}^{-1} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

When it is necessary to carry out vector control without using a speed sensor, the speed can be calculated from the values of the current and voltage of an AC motor. Open-loop solutions give a certain speed estimate, but inherently bear a large error. For better results, it is necessary to design an estimator or a filter. The Kalman filter has a good dynamic behavior, a good resistance to perturbations, and it can function at stand still. Designing a filter for an AC motor remains however a very complex problem which requires the calculation of the motor model in real time. Moreover, it is necessary to calculate the filter equations, which normally implies several matrix multiplications as well as one matrix inversion. These requirements can nevertheless be satisfied by a high-performance computing processor. Fig. 2 shows the speed-sensorless induction motor vector control scheme including the speed and rotor flux estimator based on an extended kalman filter (EKF). The tradeoffs involved in the implementation of this estimator constitute the core of the present research and require in-depth analysis and careful considerations which will make the object of a subsequent study.

III. Implementation Analysis

The basic IM vector control scheme (Fig.1) comprises 24 multiplications (including multiplication by a constant and squaring), 3 divisions, and only one square root operation. We further observe that the division and square root operators are localized in close mutual coupling inside the rotor flux and ω estimators. Because of the fact that $\cos\theta = \Psi_{r\alpha}/\Psi_r$ and $\sin\theta = \Psi_{r\beta}/\Psi_r$, no actual sine or cosine computation is involved.

Most modern FPGAs embed several tens, even up to a couple hundreds, of small, ultra-fast, 18x18 VLSI multipliers, which can readily be used in signal processing applications. All multiplications involved in an implementation of the IM vector control scheme can efficiently be implemented using these embedded multipliers.

The division and square root operators, thanks to their small number and the modest operand width required, can efficiently be implemented by traditional hardware modules without being overcostly in area. An extensive comparative study of divider implementations on FPGAs, including all kinds of restoring, non-restoring, and SRT dividers has been presented in [1]. It is even possible to implement dividers based on the small embedded 18x18 multiplier blocks [3], [4].

The matrix operations involved in implementing the EKF-based estimator are outside the scope of the present article.

We analysed the system of Fig.1 using the Xilinx blockset in Simulink. This analysis revealed that the required internal precision would be of at least 32 bits. We will show in the next section the results of a straightforward FPGA implementation of this design using System Generator.

IV. Area and Time Estimates

We did a worst case analysis of the system implementation as a network of on-line modules following the methodology proposed in [2]. Table II shows our area cost estimates for a 16-bit on-line implementation. The arithmetic modules implementation data presented in Table I have been taken from [2] for the on-line (ol-XXX) modules and SRT-DIV, while NR-SQRT is our home implementation of a non-restoring parallel-sequential square root. We see from Table III that the on-line delay on the bottleneck path of the system is about 116 clock cycles. One lap of the whole control loop therefore takes about 164 clock cycles to complete, taking into account the required conversions back and forth between standard binary and redundant number representations. Estimating that this on-line design could be clocked at 100 MHz on a Virtex-II FPGA, its total computation time would therefore be of around 1.6 microseconds.

For a crude, but rather convincing, comparision with a 32-bit parallel arithmetic implementation, the synthesis results of the VHDL code generated using System Generator, targetting

TABLE I

16-bit Arithmetic Modules Implementation Results (mostly from [2])

Module	δ	CLB	LUT	FF	Freq. (MHz)
ol-ADD	2	3	4	5	-
ol-cMUL	1	14	24	18	89
ol-cMAC	2	16	27	19	86
ol-MUL	4	93	149	139	79
ol-Div	5	115	187	122	57
SRT-DIV	N/A	36	58	34	78
NR-SQRT	N/A	25	28	43	–

TABLE II

16-bit On-line Implementation Area Estimates

	LUT	FF
4 Error Differences	4	5
Speed PI Controller	55	65
Rotor Flux PI Controller	55	65
Q-Current PI Controller	55	65
D-Current PI Controller	55	65
Decoupling	584	540
Park Transform	604	566
Inverse Park Transform	604	566
Clarke Transform 1	53	42
Clarke Transform 2	53	42
Inverse Clarke Transform	58	48
Rotor Flux Estimator	1039	791
ω Estimator	241	165
TOTAL	3460	3025

TABLE III

Estimated On-line Delay of the Bottleneck Path

Module	ol-delay
Clarke Transformation	19
Rotor Flux Estimator	38
Park Transformation	23
ω Estimator	36
TOTAL	116

TABLE IV

System Generator Synthesis Results for Virtex II xc2v2000-4

Slices	6484
Flip-flops	4700
LUTs	11086
MULT18x18	50

TABLE V

Number of Clock Cycles on the Bottleneck Path for the System Generator Design

Module	N Cycles Variant A	N Cycles Variant B
Clarke Transformation	3	3
Rotor Flux Estimator	77	15
Park Transformation	6	6
ω Estimator	34	3
TOTAL	120	27

a Virtex II xc2c2000-4 FPGA, are presented in Table IV. The dividers and square root operators used in this design are all sequential, based on the non-restoring algorithm. On lap of the complete control loop of an implementation of this design using the minimum possible number of registers takes 131 clock cycles at a maximum frequency of 50 MHz. The total computation time is therefore of about 2,5 microseconds.

In addition, two variants of this design have been evaluated where each arithmetic module outputs are registered. The first, Variant A, uses sequential dividers and square root operators, while the second, Variant B, uses a 1-clock version of the same operators. Variant A can be clocked up to around 100 MHz and takes 120 clock cycles to complete one lap of the whole control loop. The maximum frequency of Variant B is about 8 MHz, and one lap takes 27 clock cycles. From this data, a simple calculation tells us that the computation time of Variant A is about 1.2 microseconds, while that of Variant B is about 3.4 microseconds.

V. Conclusion

For the basic IM vector control scheme using a speed sensor, our analyses and estimations showed that, thanks to the absence of actual sine and cosine functions computations, and to the localized, small number of dividers (3 of them) and square root operators (only 1) involved, an FPGA implementation in parallel arithmetic using the embedded multipliers present in modern FPGAs has the potential to outperform one in on-line arithmetic, while not being overcostly in area. Even at twice the number of bits of precision than used in the on-line design, the parallel designs compare advantageously with the on-line one, both in area cost and computation time.

On-line arithmetic might reveal superior characteristics, however, when implementing the sensorless, EKF-based, vector control scheme, because of the intrinsic complexity of the matrix operations involved and the consequently increased system size. This will make the object of a further study.

Acknowledgments

This research is funded by a grant from the National Sciences and Engineering Research Council of Canada (NSERC).

CMC Microsystems provided development tools and support through the System-on-Chip Research Network (SOCRN) program.

References

[1] G. Sutter, G. Bioul, and J.-P. Deschamps, "Comparative Study of SRT-Dividers in FPGA," FPL 2004, LNCS 3203, pp. 209–220, 2004.

[2] R. Galli and A. Tenca, "A Design Methodology for Networks of Online Modules and Its Application to the Levinson–Durbin Algorithm," *IEEE Trans. on VLSI,* Vol. 12, No. 1, Jan. 2004.

[3] B.R. Lee and N. Burgess, "Improved Small Multiplier Based Multiplication, Squaring and Division," *Proc. 11th Annual Symposium on Field-Programmable Custom Computing Machines* (FCCM'03), 2003.

[4] J.-L. Beuchat and A. Tisserand, "Small Multiplier-Based Multiplication and Division Operators for Virtex-II Devices," FPL 2002, LNCS 2438, pp. 513–522, 2002.