Année 2010

UNIVERSITE DE CERGY PONTOISE

THESE

Présentée pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE CERGY PONTOISE

Ecole doctorale: Sciences et Ingénierie
Spécialité: Génie Electrique

Soutenue publiquement le 24 Novembre 2010

Par

Lahoucine IDKHAJINE

Fully FPGA-based Sensorless Control for Synchronous AC Drive using an Extended Kalman Filter

---

JURY

Président	:	Prof. Benoît ROBYNS	HEI Lille
Rapporteurs	:	Dr. François AUGER	Université de Nantes
		Prof. Serge PIERFEDERICI	ENSEM Nancy
Examinateurs	:	Prof. Marcian CIRSTEA	Anglia Ruskin University – Royaume Uni
		Dr. Josep GUERRERO	Universitat Politècnica de Catalunya – Espagne
Directeur de thèse	:	Prof. Eric MONMASSON	Université de Cergy Pontoise

Laboratoire SATIE - UCP/UMR 8029, 1 rue d'Eragny, 95031 Neuville sur Oise France

Je dédie ce travail à toute ma famille, à ma mère, à mon père, à ma sœur et à mon frère pour leur soutien et leur affection.
Une pensée à mon grand père AGNTAF M'Barek, que Dieu ait son âme !

Résumé

L'objectif du travail réalisé dans le cadre de cette thèse est de montrer l'intérêt d'utiliser les FPGAs (Field Programmable Gate Array) comme support pour l'implantation d'algorithmes complexes dédiés à la commande de machines électriques. Pour ce faire, une commande sans capteur mécanique utilisant un filtre de Kalman étendu et basée sur FPGA est réalisée. Cette commande est destinée à piloter une machine synchrone à pôles saillants. Le modèle d-q de la machine basé sur l'approximation d'inertie infinie est implanté. L'ordre du Filtre de Kalman est donc égal à 4 et la complexité totale de la boucle de régulation est évaluée à près de 700 opérations arithmétiques (dont plus de 53% de multiplications). Les apports des solutions FPGAs en termes de performances de contrôle et en termes de capacité d'intégration sont quantifiés.

En termes de performances de contrôle, il a été démontré qu'en utilisant de telles solutions matérielles, le temps de calcul est très réduit (de l'ordre de 5µs, 5% de la période d'échantillonnage). Cette rapidité de calcul permet d'avoir un contrôle quasi-instantané ce qui améliore la bande passante de la boucle de régulation. A ce sujet, une comparaison avec les performances obtenues avec une solution logicielle telle que le DSP est effectuée. Dans les deux cas, le comportement dynamique de la boucle de régulation sans capteur est quantifié.

En termes de capacité d'intégration, il est possible de développer une architecture commune qui peut être adaptée à plusieurs systèmes. A titre d'exemple, il est possible de développer un filtre de Kalman sur un même FPGA capable d'estimer les grandeurs de plusieurs systèmes sans pour autant affecter les performances de contrôle.

En outre, une méthodologie de développement dédiée à de tels algorithmes complexes est proposée. Il s'agit là d'une adaptation des méthodologies proposées dans des travaux de thèse précédents, [62] et [63]. En effet, une étape de spécification préliminaire du système ainsi que des procédures d'optimisation supplémentaires y sont introduites. Ces dernières sont particulièrement nécessaires dans le cas de commandes complexes et permettent une adéquation entre l'algorithme développé et l'architecture FPGA correspondante. De plus, cette méthodologie a été organisée de façon à distinguer l'étape du développement de l'algorithme et l'étape du développement de l'architecture FPGA.

Un état de l'art sur les technologies FPGA est également proposé. La structure interne des FPGAs récents est décrite. Leur contribution dans le domaine de la commande des machines électriques est quantifiée. Les différentes étapes de la méthodologie de développement sont présentées.

Le développement d'une commande numérique (basée sur FPGA) d'une machine synchrone à aimants permanents associée à un capteur de position Resolver est par la suite traité. Cette application s'inscrit dans un contexte avionique où l'objectif était d'avoir une solution FPGA hautement intégrée. Pour cela, le FPGA Actel Fusion est utilisé. Ce composant intègre un convertisseur analogique numérique. La commande, le traitement des signaux Resolver ainsi que la conversion analogique numérique sont implantés sur le même composant.

En ce qui concerne la commande sans capteur basée sur le filtre de Kalman étendu, il a été décidé de structurer les chapitres correspondants à travers la méthodologie de développement proposée. Ainsi, la phase de spécification préliminaire du système, la phase du développement de l'algorithme, la phase du développement de l'architecture FPGA et la phase d'expérimentation sont séparément traitées. Durant la phase d'expérimentation, la procédure *«Hardware-In-the-Loop (HIL)»* est incluse afin de valider le fonctionnement de l'architecture développée une fois la phase d'implantation physique achevée.

Mots clefs

- Réseaux de portes programmables – Field Programmable Gate Array
- Méthodologie de développement
- Commande sans capteur mécanique
- Filtre de Kalman Etendu
- Machine synchrone à pôles saillants
- Machine synchrone à aimant permanent
- Capteur de position Resolver
- Traitement des signaux Resolver

Abstract

The aim of this thesis is to present the interest of using Field Programmable Gate Array (FPGA) devices for the implementation of complex AC drive controllers. The case of a sensorless speed controller using the Extended Kalman Filter (EKF) has been chosen and applied to a Salient Synchronous Machine (SSM). The d-q model based on the infinite inertia hypothesis has been implemented. The corresponding EKF order is then equal to 4 and the complexity of the whole sensorless controller is equal to 700 arithmetic operations (more than 53% of multiplications). The contribution of FPGAs in this field has been quantified in terms of control performances and in terms of system integration.

In terms of control performances, the proposed FPGA-based solution ensures a short execution time which is around 5µs (5% of the sampling period). This treatment fastness ensures a quasi-instantaneous control which improves the control bandwidth. To this purpose, a comparison with a software DSP-based solution is made. The dynamic behavior and the influence of the execution time, in both cases, on the control bandwidth have been quantified.

In terms of integration capacity, it is possible to implement a generic FPGA architecture that can be adapted to the control of several systems. Thus, it is possible to develop a common EKF architecture that is able to estimate variables from many systems without affecting the control performances.

In addition, a design methodology adapted to such complex controllers has been proposed. The particularity of this updated methodology, compared to the previous ones ([62], [63]), is to provide an enlarged set of steps starting from the preliminary system specification to the ultimate experimentation. Optimization procedures have also been introduced. These optimizations are necessary in case of complex controllers and lead to the adequation between the developed algorithm and the corresponding hardware FPGA architecture.

A state of the art FPGA technology is also presented. The internal structure of the recent devices and their corresponding technology are discussed. Their contribution in the field of AC drive applications is quantified. An in-depth presentation of the proposed design methodology is made.

Besides, the development of a fully integrated FPGA-based controller for a Permanent Magnet Synchronous Machine (PMSM) associated with a Resolver sensor is presented. This controller has been developed for an aircraft application where the main objective was to develop a fully integrated FPGA solution. The Actel Fusion FPGA device has been used. This device integrates an Analog to Digital Converter (ADC). The current controller, the Resolver Processing Unit (RPU) and the analog to digital conversion are implemented within the same device.

When it comes to the sensorless controller, the corresponding chapters have been structured according to the presented design methodology: the preliminary system specification, the algorithm development, the FPGA architecture development and finally the experimentation. The latter includes Hardware-In-the-Loop (HIL) tests and the final experimental validation.

Keywords

- Field Programmable Gate Array
- Design methodology
- Sensorless controller
- Extended Kalman Filter
- Salient Synchronous Machine
- Permanent Magnet Synchronous Machine
- Resolver sensor
- Resolver Processing Unit

Remerciements

Les travaux de thèse présentés dans ce mémoire ont été réalisés au sein de l'équipe SETE (Systèmes d'Energies pour les Transports et l'Environnement) du laboratoire SATIE (Systèmes et Applications des Technologies de l'Information et de l'Energie), antenne de l'université de Cergy-Pontoise.

Ces quelques remerciements témoignent de la reconnaissance que je porte à toutes celles et tous ceux qui ont contribué de près ou de loin à la réussite de ce travail.

Je ne puis commencer sans remercier mon directeur de thèse Eric MONMASSON, Professeur des Universités à l'université de Cergy-Pontoise et directeur de recherche de l'antenne du laboratoire SATIE à Cergy-Pontoise. J'ai eu la grande chance d'avoir été sous sa direction depuis mon premier stage de Master. J'ai pu découvrir un homme aux grandes qualités humaines et scientifiques. Je ne lui exprimerai jamais assez ma profonde reconnaissance pour tous ses conseils, ses encouragements, sa confiance et les conditions de travail excellentes qu'il m'a procurées.

Je voudrais également exprimer mes sincères et inoubliables remerciements à Sandrine LEBALLOIS, Maître de Conférences à l'université de Cergy-Pontoise. En effet, elle a été à l'origine de mon parcours et ma réussite puisque c'est elle qui m'a conseillé, encouragé et soutenu pour continuer mes études et m'orienter vers le domaine de l'enseignement et la recherche.

Je tiens aussi à remercier Wissem NAOUAR, Enseignant Chercheur à l'ENIT (Ecole Nationale d'Ingénieurs de Tunis) pour son incomparable soutien, ses qualités humaines et ses précieux conseils qu'il m'a prodigués tout au long de cette période.

Mes remerciements s'adressent aussi à mes très chères collègues Amira MALOUF, Imene BAHRI et Herie PARK. J'ai eu le grand plaisir de travailler avec elles et j'ai particulièrement apprécié leur amitié, leur soutien et leur esprit d'équipe.

Je profite également de cette occasion pour exprimer ma profonde reconnaissance à Isabelle COLLET, Aude BREBANT, Don Abasse BOUKARI et Kamel BOUALLAGA pour leur amitié, leur contribution et leur soutien sans limite tout au long de cette période.

Mes vifs remerciements s'adressent également à Jean-Yves LEHUEROU, Bruno BUSSO, Lionel VIDO, Dejan VASIC ainsi que tout le corps professoral de l'Institut Universitaire Professionnalisé (IUP-GEII) de l'université de Cergy-Pontoise.

Je ne peux terminer sans avoir une pensée à mon grand père M'Barek (que Dieu ait son âme), mon grand père Lahcen, mes parents Mohamed et Yamina, mon frère Youssef, ma sœur Nadia et toute ma famille, pour leurs soutien et encouragements. Je voudrais souligner que ma réussite est d'abord et avant tout leur réussite. Je voudrais en outre témoigner mes sincères reconnaissances à la famille BIZOUNKAD et plus particulièrement à Lahcen BIZOUNKAD qui n'a cessé de m'assurer tous les moyens moraux et matériels pendant mes études en France.

Table of content

General Introduction

1. Thesis objectives and author contributions 12
2. Outline 13
3. Nomenclature 14

Chapter 2: State of the art FPGA technology

1. Introduction 17
2. Generic structure of an FPGA 19
   2.1. Logic Blocks 20
   2.2. Interconnection network 20
   2.3. Clock manager blocks 21
   2.4. I/O blocks 22
   2.5. Arithmetic (DSP) blocks 23
   2.6. Memory blocks 23
   2.7. Communication blocks 23
   2.8. Embedded processor cores 23
   2.9. Configuration technology 24
3. Case studies 24
   3.1. SRAM based technology 25
   3.2. Antifuse based technology 26
   3.3. Flash based technology 27
   3.4. Feature summary 28
4. Design tools 29
5. Contribution of FPGAs in complex AC drive applications 30
   5.1. Evaluation in terms of control performances 30
   5.2. Evaluation in terms of system integration 32
   5.3. FPGA implementation constraints 34
6. FPGA design methodology for control applications 35
   6.1. Preliminary system specification 36
   6.2. Algorithm development 36
   6.3. FPGA-based architecture development 38
   6.4. Experimentation 42
7. Conclusion 43

Chapter 3: Fully integrated FPGA-based controller for a PMSP associated with a resolver sensor

1. Introduction 45
2. Description and implementation of the FPGA integrated ADC 47
3. Resolver Processing Unit 48
   3.1. Resolver sensor description 48
   3.2. RPU principle 49
   3.3. Synchronous demodulation 49
   3.4. Angle Tracking Observer (ATO) 49
   3.5. Compensation of the ADC Sampling Synchronization Error for resolver signals 51
   3.6. Evaluation of the noise rejection 53
   3.7. Experimentation 54
4. FPGA-based controller description 54
   4.1. Controller timing diagram 55
   4.2. Compensation of ADC Sampling Synchronization Error for PMSM currents 56

4.3. Experimental results 58
4.4. Time/Area performances 58
4.5. Influence of the execution time on the control quality 59
5. Conclusion 59

Chapter 4: FPGA-based sensorless control for synchronous AC drive - Preliminary system specification

1. Introduction 61
2. Sensorless control system - Hardware specification 62
2.1. Power stage 62
2.2. Electrical sensors, ADC and DAC boards 63
2.3. FPGA-based digital control unit 63
2.4. Host-PC interface 63
3. Stator current controller and speed controller 63
3.1. Stator current controller 63
3.2. PWM specifications 64
3.3. Compensation of VSI nonlinearities 64
3.4. Speed controller 65
3.5. Voltage interface 65
4. Estimation of the rotor position and speed - sensorless methods 65
4.1. Sensorless methods based on signal injection 66
4.2. Sensorless methods based on the motor model 66
4.3. Choice of the sensorless method 67
5. Extended Kalman Filter basics 67
6. System state space modeling 69
6.1. Modeling in (d-q) rotating frame 69
6.2. Modeling in (α-β) stationary frame 71
6.3. Choice of the model 73
7. Conclusion 75

Chapter 5: FPGA-based sensorless control for synchronous AC drive - Algorithm development

1. Introduction 77
2. Modular partitioning 78
3. Continuous-time functional simulation 78
4. Digital realization 82
4.1. Discretization and sampling period setting 82
4.2. Algorithm normalization 83
4.3. Fixed-point data setting 84
5. Algorithm optimization 87
5.1. EKF complexity pre-evaluation 87
5.2. Optimization of the EKF algorithm 87
5.3. Complexity post-evaluation 88
6. Discrete-time, fixed-point simulation 89
6.1. Validation of the stator current controller 89
6.2. Validation of the speed controller 90
6.3. Validation of the EKF observer 92
6.4. Validation of the EKF-based sensorless speed controller 96
7. Conclusion 99

Chapter 6: FPGA-based sensorless control for synchronous AC drive - FPGA architecture development

1. Introduction 101
2. Architecture optimization 102
2.1. Optimization strategy 102
2.2. Optimization of the EKF prediction module 105
2.3. Optimization of the EKF compensator 107
2.4. Optimization of the EKF innovation module 111
3. FPGA architecture design 112
4. Architecture VHDL-coding 115
5. Architecture functional simulation 115
6. Design synthesis and time/area performances analysis 116
7. Conclusion 119

Chapter 7: FPGA-based sensorless control for synchronous AC drive - Experimentation

1. Introduction 121
2. Overview of the experimental platform 122
3. Hardware in The Loop validation 124
4. Experimental validation 130
4.1. Validation of the stator current controller 130
4.2. Validation of the sensor-based speed controller 131
4.3. Validation of the EKF observer 132
4.4. Validation of the whole sensorless speed controller 132
5. Conclusion 135

General conclusion and perspectives

1. General conclusion 137
2. Perspectives 137
2.1. Algorithm perspectives 138
2.2. FPGA development perspectives 138

Appendix A - Compensation of the VSI non-linearities 139

Appendix B - Salient Synchronous Machine - Modeling and parameter identification 145

Appendix C - Tuning of the current and speed regulators 151

Appendix D - EKF for AC drive applications - FPGA-Based solution or DSP-Based solution 155

Appendix E - FPGA-based matrix multiplication and inversion 159

Bibliography 161

Chapter 1

General Introduction

During these last years, the interest of power electronics and drive applications has been constantly rising. They have encountered a significant progress in terms of power management, by using high efficiency magnetic and power electronic materials, and in terms of control technology by implementing sophisticated control solutions.

When focusing on these control solutions, it is commonly accepted that digital-based controllers are the systematic option. A wide range of them are mostly carried out using software solutions such as microcontrollers and Digital Signal Processors (DSPs). The increasing interest to such solutions is due to their low cost, their design flexibility and their ability to implement complex control algorithms. However, their use remains limited in some industrial applications where high control performances are required. This is typically true for aircraft applications where high control reactivity and large bandwidth are key-issues. The latter are mainly related to the computing time of the used digital controller. The use of software solutions is then limited because of their fixed internal architecture which leads to fully serialize the treatment. The more complex the control algorithm, the longer the execution time. As a consequence, delays are introduced in the control closed loop which affects the control bandwidth.

To achieve high control performances, the use of fast digital solutions is then essential. Many researches and industrial applications have proved that Field Programmable Gate Array (FPGA) solutions are good candidates. Indeed, FPGAs are outperforming today's software solutions by exploiting the inherent algorithm parallelism. Consequently, implementing such hardware solution gives the possibility to develop an architecture that is fully dedicated to the control algorithm. Thus, allying today's FPGA high speed performances with parallelism, leads to a drastic reduction of the execution time. Consequently, in terms of control performances, a quasi-instantaneous control is ensured which enhances the control reactivity and bandwidth.

In addition to these control performances, the system integration is also another criterion that justifies the use of FPGAs. Indeed, their increasing integration density allows the implementation of several independent control algorithms within the same device. When exploiting the control rapidity, it

is also possible to implement a unique algorithm that can control quasi-simultaneously several systems. Along this integration way, recent FPGAs give as well the possibility to implement software treatment since they can integrate processor cores. This makes them true System on Chip (SoC) solutions. Furthermore, a novel SoC approach consists in integrating mixed signal elements such as Analog to Digital Converter (ADC) that has been encountered in niche FPGA devices.

With all these assets, the use of such hardware solutions has successfully allowed the implementation of advanced AC drive control algorithms, [24], [62]. In addition to the standard control strategies, advanced control strategies such as oversampling control, multi-level multi phase control, predictive control and re-configurable control have been proposed.

As far as the design process is concerned, FPGA solutions have been the focus of many researches. All these researches share the same objective which is to provide a design methodology that makes the design process more manageable and less intuitive, [33]-[37], [63].

1. Thesis objectives and author contributions

The proposed thesis work is a prolongation of the previously discussed researches and applications. The objective is to evaluate how FPGAs can also be suitable for the implementation of complex AC drive controllers. The case of a sensorless controller for a synchronous AC drive has been chosen. The chosen sensorless method is based on the Extended Kalman Filter (EKF) which estimated the rotor position and speed from the current and voltage quantities. Due to the EKF complexity, such sensorless controller is systematically implemented in DSP solutions. With these solutions, the execution time is frequently evaluated to several tens or hundreds of microseconds.

The aim is then to evaluate how, with only a few microseconds of computing time (less than 5µs), an FPGA solution can boost the control performances and how they can increase the system integration. In the following, more details about the thesis objectives and author's contributions are listed,

- Before starting the development of the sensorless controller, a first evaluation of the hardware FPGA solutions in terms of system integration is achieved. A sensor-based controller for a Permanent Magnet Synchronous Machine (PMSM) is developed. This PMSM is associated with a resolver position sensor. This first task belongs to an avionic research program and the objective was to develop a fully integrated FPGA-based controller. To reach this integration aim, the used target is the Actel Fusion FPGA. In addition to the digital features, this device integrates also an ADC. The PMSM controller, the resolver signal treatment and the analog to digital conversion are all ensured by the same device. This development has led to the following publications: [18]-[23].

The next points are related to the development of the FPGA-based sensorless controller using the EKF. This controller is applied to a Salient Synchronous Machine (SSM). Here again, the developed design is intended to be used in an aircraft application which aims to develop a sensorless controller for a Brushless Synchronous Starter Generator (BSSG) [38], [65], [92].

- Adaptation of the design methodology proposed in [63] to complex control applications. The proposed methodology is reorganized and is divided into four main phases: (i) the preliminary system specification, (ii) the development of the algorithm, (iii) the development of the FPGA architecture and (iv) the experimentation. This methodology is applied to the development of the EKF-based sensorless controller, [25].
- Insertion of algorithm optimization procedures to the design methodology. The objective here is to optimize the complexity of the algorithm by reducing its computational cost. When applied to the developed sensorless controller, this optimization is to be balanced with the precision and the dynamic behavior, [69].

- Insertion of an FPGA architecture optimization procedure. The latter takes into account implementation constraints. The objective is to develop an FPGA architecture that is able to process the algorithm with the consideration of timing and area constraints, [67].
- Analysis and quantification of the control performances. To this aim, the control bandwidth of the developed FPGA-based sensorless controller has been quantified. The same analysis is done with a DSP-based sensorless controller. Both of these solutions have been developed and compared, [68].
- Analysis and quantification of the system integration. With the obtained time/area performances, author states the possibility to implement a common architecture that can be adapted and used quasi-simultaneously to drive many systems without downgrading their control performances.
- Hardware-In-the-Loop validation of the developed FPGA-based sensorless controller, [24], [66].
- Experimental validation of the developed FPGA-based sensorless controller, [24], [25], [65]-[69].

2. Outline

This thesis report is basically divided as follows.

In chapter 2, a state of the art FPGA technology is presented. In this chapter, author starts by presenting the evolution of FPGAs and their value-added to the nowadays industrial electronics. A generic structure of the recent devices is provided and then case studies are achieved. In the latter, the features of recent FPGAs from each technology are given. Their contribution in the field of power electronics and drive applications is presented. To this aim, we have focused on the case of complex AC drive control applications. A quantitative analysis in terms of control performances and system integration is achieved. Finally, the proposed design methodology is presented.

In chapter 3, author deals with the fully integrated FPGA-based controller for the PMSM associated with the Resolver sensor. The used digital control unit is based on the Actel Fusion FPGA. At first, the presentation and the implementation of the integrated Analog to Digital Converter (ADC) is achieved. Then the principle of the Resolver sensor is presented. The extraction of the rotor position and speed is ensured by the FPGA-based Resolver Processing Unit (RPU). This module is associated with a Hysteresis PMSM current controller. The introduced Sampling Synchronization Error (SSE) by the implemented ADC is also discussed and compensation procedures are proposed.

Now, the development of the FPGA-based sensorless speed controller is initiated. Chapters 4 to chapter 7 are organized according to the proposed design methodology.

In chapter 4, the preliminary system specification is discussed. The objective of this step is to make a hardware specification of the system in the one hand, and make an algorithm benchmarking in the other hand. For the first case and depending on the mechanical load conditions, the AC drive and the supply conditions are chosen. The algorithm benchmarking consists in choosing the control strategy, the sensorless method and the system state space model.

In chapter 5, the development of the whole sensorless algorithm is achieved. During this phase, the modular partitioning, the continuous-time simulation, the digital realization, the algorithm optimization and the necessary discrete-time and fixed-point simulations are achieved.

Chapter 6 treats the development of the corresponding FPGA architecture. Taking into account the defined implementation constraints, the optimization of this architecture is made with the help of the Algorithm Architecture Adequation methodology. Then the design of the architecture, its VHDL-coding and the time/area performances analysis are made.

Finally, chapter 7 presents the experimentation step. In order to make a first experimental operating guarantee, the Hardware-In-the-Loop (HIL) validation is performed. Then the ultimate experimental validation using the presented experimental platform is achieved.

3. Nomenclature

3.1. Symbols

Clk : Clock signal
Reset : Reset signal
En : Enable signal
Init_done : ADC configuration flag – Fusion FPGA
Start : Start signal
DATAVALID : ADC result ready – Fusion FPGA
CALIBRATE : ADC calibration flag – Fusion FPGA
V_{in} : ADC analog voltage input – Fusion FPGA
CHNumber : Analog multiplexed selection input – Fusion FPGA
xxx_RDY : ADC conversion flag – Fusion FPGA
t_{conv} : ADC Conversion time – Fusion FPGA
ε_{SSE} : ADC sampling synchronization error – Fusion FPGA
θ_e, ω_e : Electrical angular position and speed estimated from the Resolver Processing Unit
θ_r, ω_r : Actual position and speed, from Resolver sensor
θ_{offset} : Position offset
V_{cos}, V_{sin} : Amplitude modulated Resolver sensor outputs
V_{cos_D}, V_{sin_D} : Demodulated Resolver signals
E : Resolver Excitation signal
ω_{ex} : Resolver Excitation pulsation
m : Resolver transformation ratio
ε_θ : RPU angular position error
H_θ : RPU position-position transfer function
H_ω : RPU speed-position transfer function
K_1 : RPU closed-loop coefficient
K_2 : RPU closed-loop coefficient
T_s : Sampling period
Bw : Hysteresis bandwidth
i_{sa}, i_{sb}, i_{sc}, I_{rd} : 3-phase stator currents and the rotor currents
ε_{isa}, ε_{isb}, ε_{isc}, : 3-phase stator current errors
i_{sd}, i_{sq} : d-q stator currents
ε_{id}, ε_{iq} : d-q current errors
$i_{s\alpha}$, $i_{s\beta}$: α-β stator currents
θ, ω : Electrical angular position and speed
V_{DC}, V_{ZSS} : DC link voltage, Zero Sequence voltage
V_{sa}, V_{sb}, V_{sc} : 3-phase stator voltages
v_{sd}, v_{sq} : d-q stator voltages
$v_{s\alpha}$, $v_{s\beta}$: α-β stator currents
S_a, S_b, S_c : VSI switching signals
V_a, V_b, V_c : 3-phase power supply voltages
V_{ao}, V_{bo}, V_{co} : 3-phase voltages – VSI voltages
T_e, T_L : Electromagnetic torque, Load torque
R_s, R_r : Stator, rotor Resistances
L_{sd}, L_{sq} : d-q stator inductances
M_{sr} : Stator-rotor mutual inductance
f_L, J : Viscous friction coefficient, Rotor inertia
p : Pole pairs number
N : Mechanical speed
Ω : Angular mechanical speed
θ_m : Mechanical rotor position
V_{bmf} : Back-EMF voltage
Ψ : Flux

x, u, y	: State space vector, System input and output vectors
w, v	: System disturbances
f, h	: continuous-time state space matrix, System output matrix
f_d, h_d	: Discrete-time State space matrix, Discrete-time System output matrix
F_d, H_d	: Jacobian matrices for linearization
K	: Kalman gain matrix
P, P_0	: State error covariance matrix, Initial covariance matrix
Q, R	: Model noise and measurement noise covariance matrices
K_p	: PI-regulator proportional gain
K_i	: PI-regulator integral gain
V_{nom}	: Nominal voltage
I_{nom}	: Nominal current
N_{nom}	: Nominal speed (rpm)
V_{Bcc}	: Voltage base value for current controller
V_{Bekf}	: Voltage base value for the EKF
I_B	: Current base value
ω_B	: Angular speed base value
θ_B	: Angular position base value
$G_{vsensor}$	: Gain introduced by the voltage sensor
$G_{isensor}$	: Gain introduced by the current sensor
G_{ADC}	: Gain introduced by the ADC

3.2. Indexes

s, r	: Stator and rotor index
d, q	: Rotating reference frame indexes
α, β	: Stationary reference frame indexes
$*$, ^	: Reference quantity, Estimated quantity
k	: Sampling index
B, n	: Base quantity for normalization, Normalization index
a, b, c	: 3-phase reference frame index
$k/k-1, k/k$	: Predicted quantity, Estimated optimal quantity
$k-1/k-1$	: Estimated optimal quantity at the previous sampling period

3.3. Abbreviations

A^3	:	Algorithm Architecture Adequation
ADC	:	Analog to Digital Converter
ASIC	:	Application Specific Integrated Circuit
ATO	:	Angle Tracking Observer
CAD	:	Computer Aided Design
CAM	:	Computer Aided Manufacture
CAN	:	Control Area Network
CAT	:	Computer Aided Test
CB_PWM	:	Carrier Based PWM
CB_SPWM	:	Carrier Based Sinusoidal PWM
CB_ZSS_PWM	:	Carrier Based PWM with Zero Sequence Signal
CCC	:	Clock Conditioning Circuit
CLB	:	Configurable Logic Block
CMT	:	Clock Management Tile
CORDIC	:	COordinate Rotation Digital Computer
CPLD	:	Complex Programmable Logic Device
DAC	:	Digital to Analog Converter
DCM	:	Digital Clock Manager
DFG	:	Data Flow Graph
DSP	:	Digital Signal Processor
EDA	:	Electronic Design Automation
EDK	:	Embedded Development Kit

EDS	:	Embedded Design Suite
EEPROM	:	Electrically Erasable Programmable Read Only Memory
EKF	:	Extended Kalman Filter
EMF	:	Electromotive Force
EPROM	:	Erasable Programmable Read Only Memory
FDFG	:	Factorized DFG
FIFO	:	First In First Out
FPGA	:	Field Programmable Gate Array
HIL	:	Hardware In the Loop
I/O	:	Input / Output
I2C	:	Inter Integrated Circuit
IC	:	Integrated Circuit
ICON	:	Integrated CONtroller
IGBT	:	Insulated Gate Bipolar Transistors
ILA	:	Integrated Logic Analyzer
IOE	:	I/O Element
IP	:	Intellectual Property
LAB	:	Logic Array Block
LB	:	Logic Block
LC	:	Logic Cell
LE	:	Logic Element
LM	:	Logic Module
LUT	:	Look Up table
MMCM	:	Mixed Mode Clock Manager
OTP	:	One Time Programmable
P_PI	:	Proportional – Proportional – Integral
PCI	:	Peripheral Component Interconnect
PI	:	Proportional – Integral
PLL	:	Phase Locked Loop
PMSM	:	Permanent Magnet Synchronous Machine
PWM	:	Pulse Width Modulation
RAM	:	Random Access Memory
RDC	:	Resolver to Digital Converter
ROM	:	Read Only Memory
RPU	:	Resolver Processing Unit
SAR	:	Successive Approximation Register
SEU	:	Single Event Upset
SIA	:	Satellite Industry Association
SMO	:	Sliding Mode Observer
SoC	:	System on Chip
SPI	:	Serial Peripheral Interface
SR	:	Shift Register
SRAM	:	Static Random Access Memory
SSE	:	Sampling Synchronization Error
SSM	:	Salient Synchronous Machine
SVPWM	:	Space Vector PWM
THD	:	Total Harmonic Distortion
USB	:	Universal Serial Bus
VCO	:	Voltage Controlled Oscillator
VHDL	:	Very high speed integrated Hardware Description Language
VIO	:	Virtual Input/Output
VSI	:	Voltage Source Inverter
VT	:	VersaTile

Chapter 2

State of the art FPGA technology

1. Introduction

A few decades ago, digital electronic designs were mainly based on basic Integrated Circuits (ICs). Each of them was fully specified by the manufacturer and has a specific function. The end-user had to make a collection of the necessary ICs to develop a digital circuit that performs the desired function. Generally, digital systems were easier to develop, smaller in terms of complexity, heavier in terms of board size and ran at low speed performances. No Design tools existed which made the design processes more intuitive.

The increasing success of these digital systems and their value-added to the early everyday's life made customers starting to ask for more sophisticated device solutions. Device solutions that are manufactured and specified to the developed application and that are smaller, faster, more complex, low cost and low power consuming. All these challenges made designers to compete and propose many solutions that led to the expansion of today's electronic industry.

Driven by these demands, new materials, new fabrication processes, new design tools and device technologies have been proposed and new markets for digital circuits have evolved. Indeed, with the increasing resort to informatics and their associated computing resources, a set of efficient software tools are proposed [8]. Therefore, Electronic Design Automation (EDA) and Computer Aided Design (CAD) tools are provided for the design process. For the fabrication process, Computer Aided Manufacture (CAM) and Computer Aided Test (CAT) tools are proposed.

When it comes to device technologies, many solutions are nowadays available and an additional benchmarking effort has to be made by the end-user so as to choose the best one that suits the desired performances and also the expected level of flexibility, re-programmability, cost and power consumption. This leads to many start-up questions that are asked during system specification: do we have to use a set of standard ICs, a rapid-prototyping reprogrammable solution (Semi-Custom Application Specific Integrated Circuit-ASIC) or a fully custom solution (Full custom ASIC)? Do we have to use fully software devices (Digital Signal Processor-DSP, µProcessor, µController …), fully

hardware devices (Complex Programmable Logic Device-CPLD, Field Programmable Gate Array-FPGA …) or mixed software/hardware devices (FPGA System on Chip-SoC)?

Among this diversity of digital solutions and since their first introduction to the market in 1985 by the Xilinx Company, FPGA hardware technologies have attracted an always increasing interest. Indeed, FPGAs belong to the semi-custom ASIC family. The latter low cost devices consist of pre-designed (by the manufacturer) elementary cells and interconnections that can be programmed and interconnected by the user in order to realize a specific function for a specific application. In addition to their low manufacturing cost, these devices distinguish themselves by a high integration density, a notable computation rapidity, a high level of flexibility and a rapid prototyping.

Nowadays, the always increasing integration density, speed and the low power consumption make FPGA solutions suitable for complex applications in different domains such as, digital signal processing, power electronics and drive applications, communication, aerospace, defense, etc. This is also made possible with the considerable progress in terms of process technology which has reached down to 40nm (28nm has been recently announced by Xilinx and Altera vendors) [2], [3]. To give a visual evolution of FPGA capacities, Figure 2.1 overviews the evolution of FPGAs in terms of density (logic cells), speed, and process technology. It is worth noticing that these waveforms have been obtained by comparing the available commercial FPGAs. Also the measurement of speed is based on the maximum available clock frequency that can be reached within the FPGA.

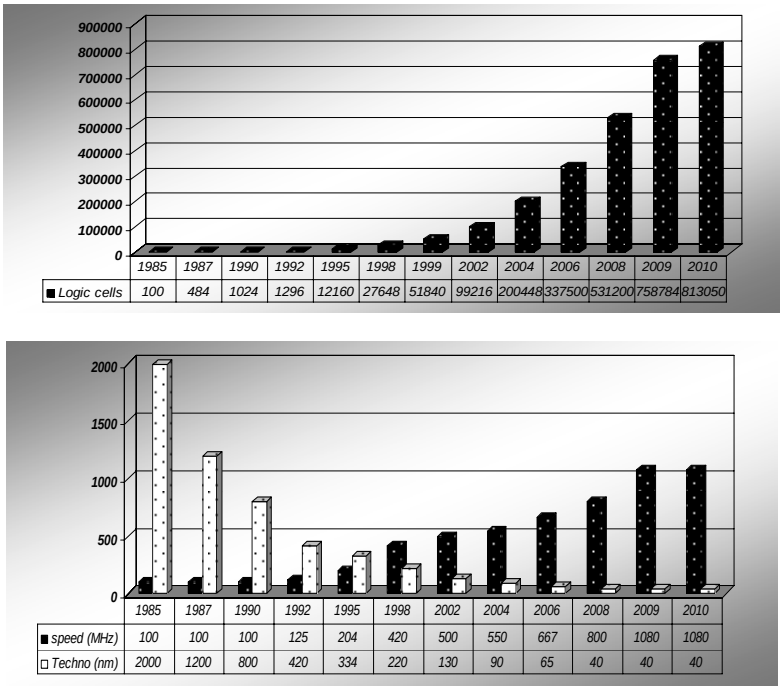


Figure 2.1: Evolution of FPGA performances

From a more financial point of view, many market analyses have been achieved and the conclusions in each case gave promising financial value-added of FPGAs. For example, according to an *In-Stat market analyst* documentation article (published in 2006) where an estimation of the FPGA market evolution was made [1], the value of worldwide FPGA market would increase from $1.9 billion in 2005 to $2.75 billion by 2010. Another precise example of FPGA market is given in Figure 2.2 where the evolution of Dollar consumption by Major (Satellite Industry Association) SIA defined regions is stressed.

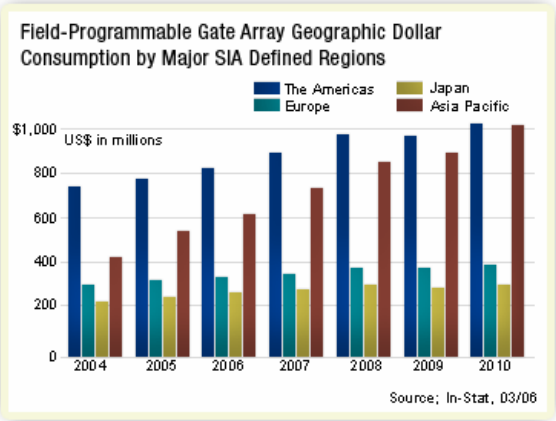


Figure 2.2: FPGA market evolution in SIA regions (source: In-Stat market analyst)

The goal and the challenge of this chapter are to introduce a large part of FPGA facets. Thus, the next part introduces the generic structure of an FPGA where the most important and relevant FPGA elements are presented. The case studies provided in part 3 aim to make a quantitative description of FPGAs by focusing on precise FPGA devices from each technology. Part 4 is devoted to the contribution and the application of FPGAs in the field of power electronics and drive applications especially in the case of complex control algorithms. Advantages of using FPGAs in this field and also implementation constraints to manage are both focused on. Finally, an FPGA design methodology dedicated to power electronics and drive applications is discussed.

2. Generic structure of an FPGA

As presented in Figure 2.3, the basic structure of an FPGA consists of a matrix of logic blocks, an interconnection network and configurable I/O blocks [2]-[17]. To ensure high level of integration, today's FPGA devices also include coarse-grain hardwired elements such as memory blocks, arithmetic (DSP) blocks, clock manager blocks and communication blocks. Furthermore, FPGA solutions give the possibility to implement embedded processor cores which makes them true System on Chip (SoC) solutions [26]-[29]. Also, a novel SoC approach consists in integrating mixed signal elements such as Analog to Digital Converters (ADCs) that has been encountered in the Actel Fusion FPGA [10].

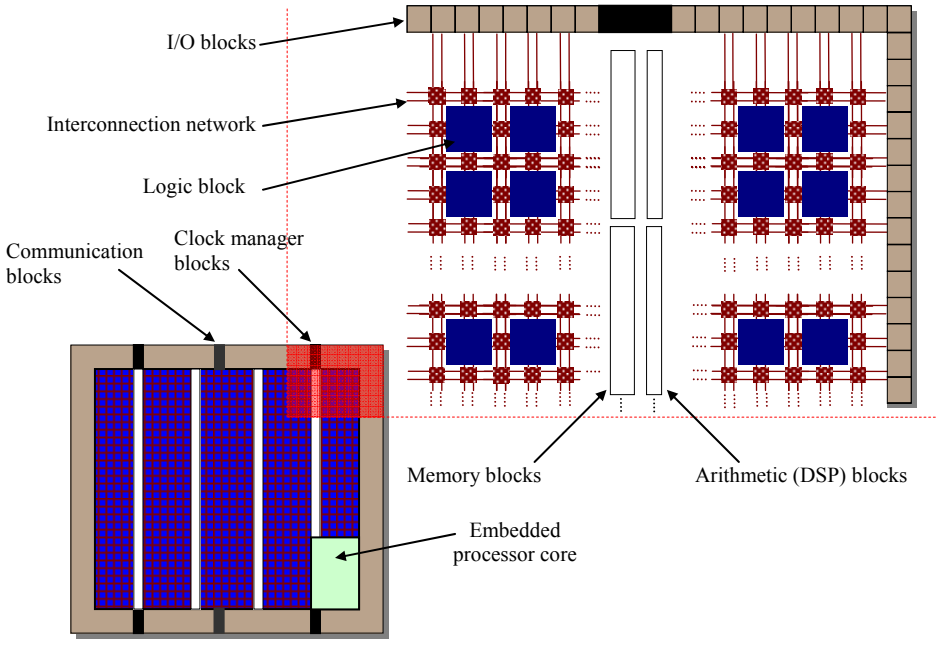


Figure 2.3: Generic structure of an FPGA (Island topology)

2.1. Logic Blocks

First of all, it is worth noticing that the naming of Logic Blocks (LBs) has been intentionally chosen so as to keep the generic structure as independent on FPGA families as possible. In fact, to anticipate the discussion made in the case studies (part 3); different appellations are adopted by FPGA vendors with different levels of granularity. For instance, Slice and Configurable Logic Block (CLB) appellations are used by Xilinx vendor, [2]. Logic Element (LE) and Logic Array Block (LAB) are used by Altera Vendor, [3] and VersaTile is used by Actel, [4].

Depending on the expected function to implement, each LB is configured to perform combinatorial and/or sequential operations. An LB is generally composed of a set of Look-Up-Tables (LUTs) dedicated to combinatorial operations and a set of D-Flip-Flops for sequential operations. In addition to this basic operating mode, an LB is also able to perform a local storage function (distributed RAM memory), shift register (SR), multiplexer, and adder/subtractor operations.

Although the internal structure of LBs differs from an FPGA family to another, a normalized FPGA density metric has been accepted. Indeed, a common Logic Cell (LC) has been defined which is composed of a 4-bit LUT, a D-Flip-Flop, a carry chain (for arithmetic operations) and a multiplexer. Roughly speaking, Altera FPGAs include LEs, each of them is equivalent to one LC [3]. A Xilinx Spartan-6 FPGA CLB is equivalent to 12.8 LCs [12] and a Virtex FPGA CLB is equivalent to 4.5 LCs [13]. Figure 2.4 highlights the internal structure of an elementary LC.

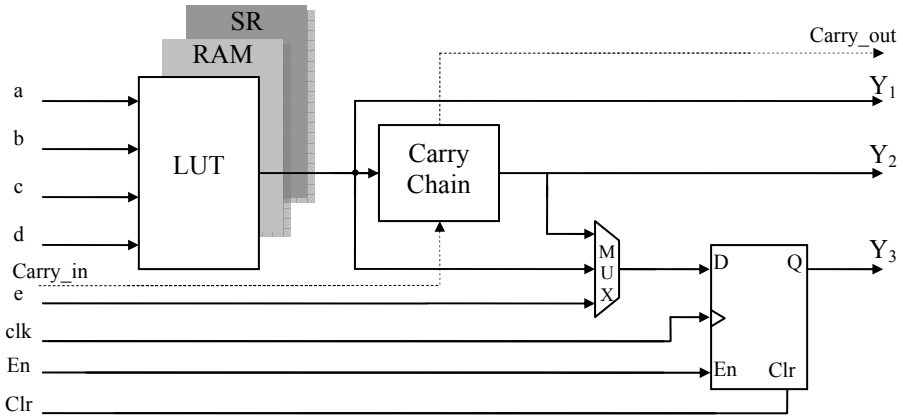


Figure 2.4: Internal structure of an elementary LC

2.2. Interconnection network

The programmable interconnection network is the backbone of the FPGA logic resources. Where the FPGA fabric performs arithmetic and logical computations, the programmable interconnection network makes the necessary connections between the necessary elements so as to develop the architecture that performs the expected user function.

In many today's FPGAs, three typical interconnection techniques can be encountered: the nearest-neighbor technique, the segmented technique and the hierarchical technique [6]. By considering the distance between the FPGA elements to be interconnected and the complexity of the developed architecture, a combination of these techniques is made so as to optimize signal propagation delays.

2.2.1. Nearest-neighbor technique

This technique consists in directly routing each logic block with each of its immediate neighbors using local interconnection [6]. Although, this technique has the credit to be simple, it has a lack severe propagation delay and connectivity issues. This is especially true when the distance and the complexity increase. This is the reason why this technique is combined with segmented and hierarchical techniques.

2.2.2. Segmented technique

The routing is, in this case, made using specific switch matrices. As described in Figure 2.5, the interconnection between logic blocks is segmented and the switch matrices ensure the connectivity between these segments, allowing long distance routing to be accomplished and then ensuring optimized propagation delays [6]. As an example, this topology is realized in Xilinx and Altera FPGAs.

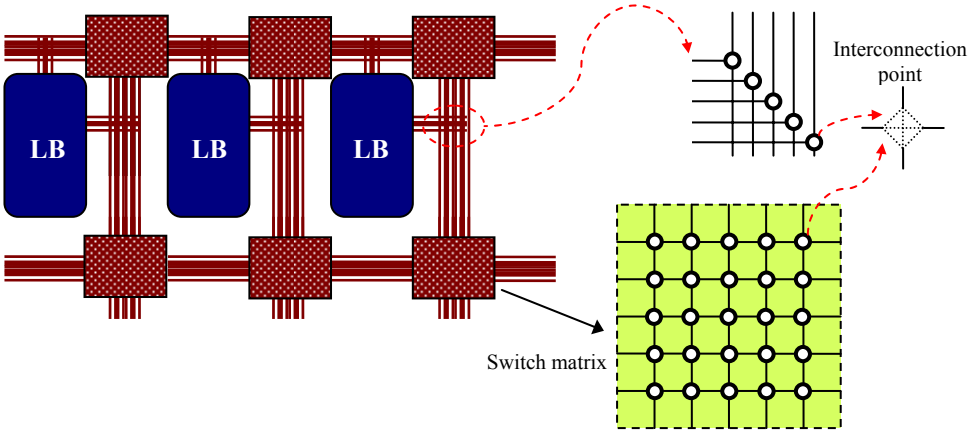


Figure 2.5: Segmented interconnection technique

2.2.3. Hierarchical technique

This is a slightly different approach to reducing propagation delays of long wires. In fact, as depicted in Figure 2.6, at the lower level of hierarchy, local neighbor logic blocks are grouped together as a single cluster. Within each cluster, local nearest-neighbor routing is made. Then, at the immediate higher level of hierarchy, adjacent clusters are grouped and form another cluster in the higher level of grouping [6]. This is repeated at higher levels of hierarchy, with larger clusters (super-clusters) and longer wires. As in the segmented technique, the connection points that connect one level of routing hierarchy to another are ensured by switch matrices. This interconnection technique is typically used by Actel Antifuse FPGAs where 3 levels of hierarchy are available, [4].

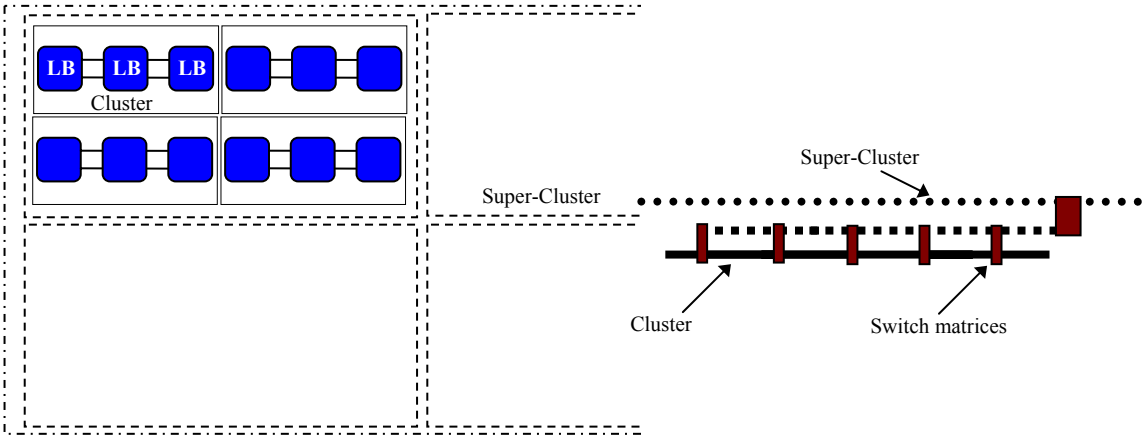


Figure 2.6: Hierarchical interconnection technique

2.3. Clock manager blocks

The integrated clock manager blocks allow the management of the clocking resources within the FPGA. They are commonly based on high frequency Phase-Locked-Loops (PLLs) that support several features for general purpose clock management such as frequency multiplication/division, phase shifting, propagation delay compensation and duty cycle correction. For instance, Xilinx Virtex-6 FPGA includes up to 18 clock manager blocks (called Mixed Mode Clock Manager MMCM, [14]) and Altera Stratix-4 FPGA provides up to 12 PLLs, [16].

As far as clock distribution is concerned, a dedicated global clock network is available so as to wire clock signals to the FPGA elements. In addition, specific FPGA pins and buffers are provided so as to ensure a high speed clock signal transmission.

2.4. I/O blocks

The Input/Output blocks provide a programmable bidirectional interface between the internal FPGA fabric and the external environment. They are usually organized as banks and can reach up to 1200 I/O blocks (Xilinx Virtex-6 FPGA). Each bank can be dedicated to different I/O standards including single-ended I/Os, differential I/Os, voltage referenced I/Os and high speed interfaces (PCI and memory interfaces). A simplified diagram of an I/O block internal structure is presented in Figure 2.7. There are three main signal paths, the output path, the input path and the 3-state path. For synchronization purpose, each path contains a set of storage elements that act as registers or latches.

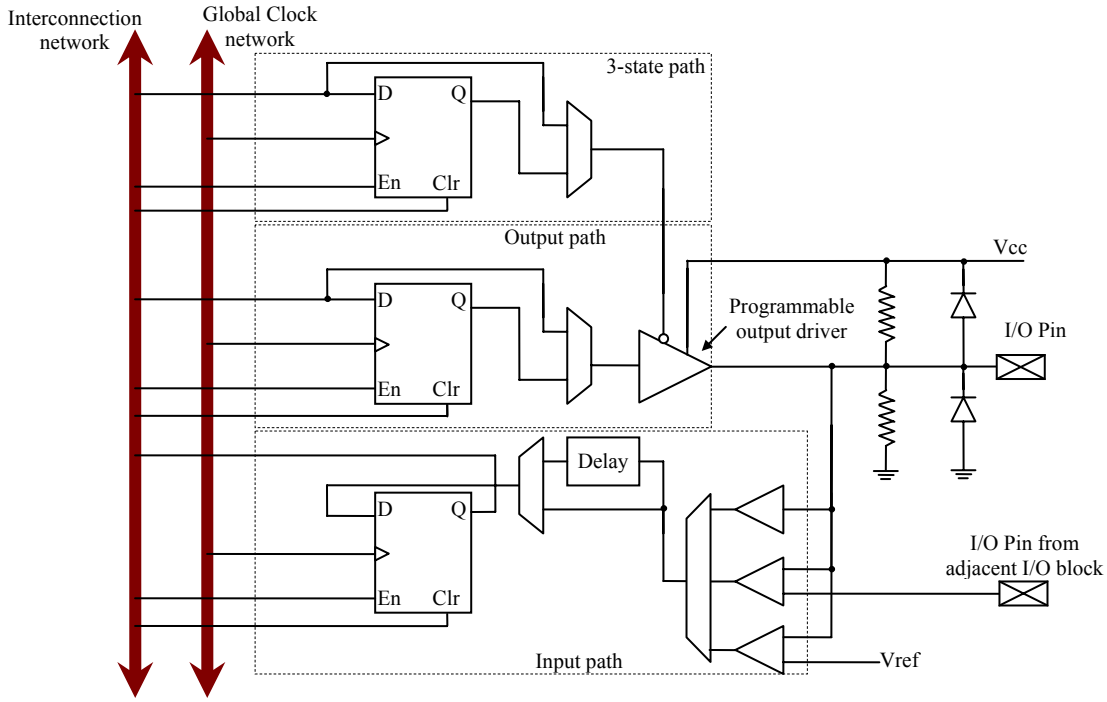


Figure 2.7: General structure of an I/O block

2.4.1. Input path

This path carries data from the external FPGA pin to the internal logic. A programmable delay and storage elements are introduced in order to control the synchronization of data with the clock signal. The input path can be configured to ensure standard input, differential input with another input path from an adjacent I/O block or voltage referenced input.

2.4.2. Output path

This path carries data from the internal logic to the external FPGA pin. Storage elements can also be used to synchronize the data transfer with the clock signal.

2.4.3. Three-state path

This path sets the FPGA pin in a high impedance state. A programmable output driver is used to select either the output path (output mode) or the 3-state path (high impedance mode).

2.5. Arithmetic (DSP) blocks

In order to suit the high demand of resources for complex applications, recent FPGAs give the possibility to implement arithmetic operations using hardwired arithmetic blocks. In most of the cases, the latter consist of a large amount of hardwired multipliers, e.g. up to 2000 multipliers (25x18 bits) are included in Virtex-6 SX Xilinx FPGA series. In addition, these arithmetic blocks are pipelined using a set of registers in order to enhance speed performances.

Furthermore, in the advanced recent FPGAs, more complex arithmetic blocks are provided: DSP blocks. Indeed, these pipelined blocks consist of a combination of a multiplier, an adder/subtractor and an accumulator. Implementing such blocks allows more complex arithmetic operations in high computational and high frequency demanding applications such as filtering, image processing, video treatment, signal transmission … Examples of FPGAs that provide DSP blocks are: Xilinx Virtex-6, Xilinx Spartan-6 and Altera Stratix-4.

2.6. Memory blocks

In addition to logic blocks and hardwired arithmetic blocks, FPGA devices support a large amount of embedded memory blocks to increase hardware resource availability and speed performance. Indeed, in large systems that often require data storage, implementing on-chip memories has a value-added especially by increasing the system integration density and allowing faster read/write operations.

In nowadays FPGAs, most of these on-chip data storage blocks consist of configurable blocks which allow the implementation of various memory structures including RAMs, ROMs, FIFOs and shift registers. As an example, the Xilinx Virtex 6 FPGA provides up to 1064 RAM blocks that correspond to up to 38 Mb RAM capacity and the Altera Stratix-4 FPGA provides up to 33 Mb embedded RAM memories.

In addition to these dedicated blocks, distributed RAM memories are also available. This is especially true when talking about SRAM-based FPGAs where the integrated logic blocks can run as distributed RAM and/or shift registers.

For the same purpose and in order to provide a nonvolatile memory feature, niche FPGA families such as Flash-based FPGAs support also Flash memory blocks. For example, Actel Fusion FPGA supports up to 8 Mb flash memory resources, 1 Kb FlashROMs and up to 270 Kb RAMs, [10].

2.7. Communication blocks

The current FPGA devices include also communication blocks that consist generally of transmission and reception buffers. Various communication protocols (standard or user-defined) are supported, including among others USB, Ethernet, CAN, PCI, SPI and I2C protocols. For this aim, dedicated transceivers are provided so as to support many of these protocols and ensure high data transmission rates, e.g. up to 11 Gbps with Stratix-4 and Virtex-6 FPGA.

2.8. Embedded processor cores

Besides the evolution of the discussed FPGA elements and in order to meet more flexibility and higher integration capability, the recent FPGA devices give the possibility to implement an increasing diversity of processor cores. They are then considered as System-On-Chips (SoCs) or System-on-Programmable-Chips (SoPCs) solutions [26]-[29].

In such SoPC approach, two categories of processor cores can be encountered, the “non-synthesizable” cores and the “synthesizable” cores.

The non-synthesizable (also called hard processor) cores are hardwired within the FPGA, in addition to the previously discussed FPGA elements. As a general rule, a hard processor core offers higher clock speeds with less flexibility. For example, Altera provides an ARM9 processor core embedded in its EPXA10 series that is marketed as an Excalibur™ device [3]. The Xilinx Virtex-5

integrates also a hardwired PowerPC 440 processor cores on-chip [2]. Recently, Actel has provided the first hardwired Cortex-M3 processor core integrated into its Fusion FPGA family [10].

The synthesizable (Soft cores), such as Altera Nios II, Xilinx MicroBlaze processors and Actel ARM7 or Cortex-M1, use existing FPGA configurable elements to implement the processor core, [2]-[4]. The particularity of such approach is the flexibility that allows the designer to configure and specify the number, the types of peripherals, the memory width… However, these cores have slower clock rates.

2.9. Configuration technology

The internal structure, the operating mode and the configuration of the discussed FPGA elements differ depending on the device family and technology. There are various configuration methods and technologies including, SRAM, EPROM, EEPROM, Fuse, Antifuse and Flash technologies. Because of their popularity, only the SRAM, the Antifuse and the Flash technologies have been here investigated.

2.9.1. SRAM technology

The most widely used method for storing configuration data in available FPGAs is volatile static RAM, or SRAM. The configuration is entirely made using a set of dedicated SRAM blocks. These blocks are organized as a specific configuration layer. The most popular SRAM-based FPGA families are Xilinx and Altera families. The popularity of this technology is manly due to their fast and infinite reconfiguration cycles [6]. Drawbacks of such technology are power consumption and data volatility. Indeed, compared to the other technologies, an SRAM-based connection point is based on high number of transistors (6 transistors) and dissipates significant static power because of leakage current. Another significant drawback is that SRAM does not maintain its contents after power is off, which means that at power-up the FPGA is not configured and must be programmed using off-chip logic and storage, [6].

2.9.2. Antifuse technology

The Antifuse technology is based on the so-called Antifuse connections that are based on metal link. The latter behaves in the opposite of a Fuse. In other words, an Antifuse link is normally unconnected and a specific programming procedure is required to make the connection (i.e short circuit between Antifuse endpoints). This procedure consists in injecting a high current or a Laser that heats and then melts the silicon layer between endpoints so as to make the connection [6]. The main drawback of such technology is that FPGAs in this case are One-Time-Programmable (OTP) which limits significantly their flexibility and make them useless for prototyping environments.

2.9.3. Flash technology

This technology is definitely an interesting gap between SRAM technology and Antifuse technology since the configuration is based on flash connections that keep the configuration state when the power is off. Furthermore, a flash-based connection point uses less number of transistors than its SRAM counterpart (2 transistors sharing a floating gate) [4]. Consequently, this yields to lower current leakage and then to lower static power consumption. Also this technology is useful in aircraft and space systems since it guaranties the configuration against the Single Event Upset (SEU) radiations. In contrast, the main drawbacks of such technology are slow configuration rate and a limited number of reconfiguration cycles.

3. Case studies

In order to move from an abstract presentation of FPGAs and see how exactly they look like, single examples have been chosen from the available and most recent FPGAs. The selection is based on configuration technology criterion. Then, a deeper investigation of each example for each configuration technology is made.

3.1. SRAM based technology

The most popular SRAM-based FPGA families are Xilinx and Altera families. Among the commercialized FPGA devices, one can stress the high performance VIRTEX (Xilinx) and STRATIX (Altera) FPGAs and the low cost SPARTAN (Xilinx) and CYCLONE (Altera) FPGAs. As it will be discussed afterwards, in the field of power electronics and drive applications, cost is a key-issue. As a consequence, it has been chosen to present only the low cost FPGA families. As examples of illustration, the latest Xilinx SPARTAN-6 and Altera Cyclone-4 FPGA families will be investigated.

3.1.1. Xilinx Spartan-6 FPGA

A Spartan-6 SRAM-based FPGA incorporates a combination of the previously discussed FPGA elements and is based on 45-nm process technology. There are up to 11519 CLBs (equivalent to 147443 LCs) and, as seen in Figure 2.8, each single CLB contains a pair of slices: SliceX and SliceL/SliceM (globally, a Spartan-6 contains 50% of SliceX, 25% of SliceM and 25% of SliceL). Each slice can be configured to perform combinatorial functions using four 6-bit LUTs and sequential functions using eight D-Flip-Flops. Furthermore, the SliceM can operate as a distributed RAM block, as a shift register, as a multiplexer or as a carry chain that performs arithmetic additions and subtractions. A SliceL supports all the SliceM features except the memory and shift register functions. To have a deeper idea about the structure of each slice, refer to [12]

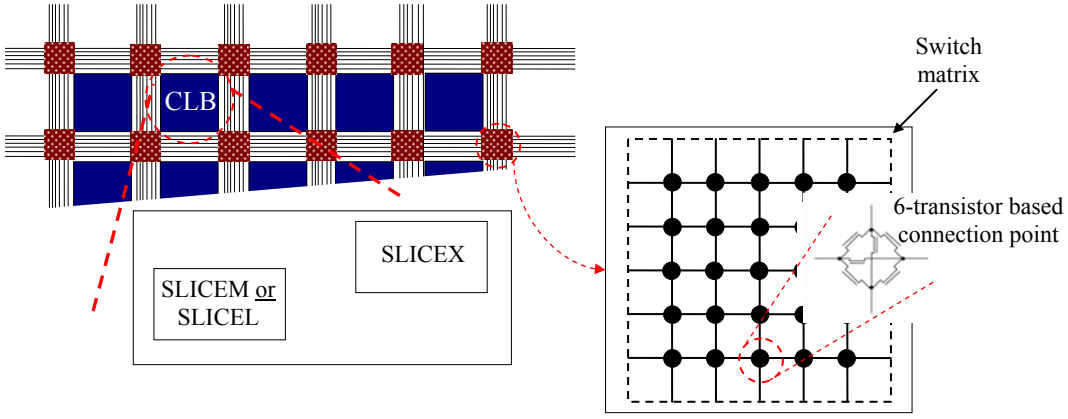


Figure 2.8: SRAM-based SPARTAN-6 FPGA fabric

The interconnection network is composed of wires and a sea of switch matrices (with 6-transistor based connection points). The interconnection of FPGA elements in this case is based on segmented technique and the routing resources are physically located in horizontal and vertical routing channels. Various types of routing are ensured depending on the distance between elements to interconnect (Fast interconnect, Single interconnect, Double interconnect and Quad interconnect).

As for the hardwired arithmetic blocks, Spartan-6 FPGA includes up to 180 DSP blocks named DSP48A1 slices. Each one supports many independent functions, including 18-bit multiplier, 48-bit accumulator, 18-bit adder/subtractor, a wide bus multiplexer, magnitude comparator and a wide counter.

The number of I/O blocks varies from 102 to 576 depending on the series and device size. All of them operate as bidirectional interfaces. They are organized as banks and can support a large number of single-ended, differential and voltage referenced standards.

Spartan-6 FPGAs contain a set of clocking resources and Clock Management Tiles (CMTs). The provided clocking resources consist of dedicated clock inputs, buffers and routings. There are 8 dedicated clock inputs, 32 global clock inputs that can operate as general purpose I/Os and 16 clock buffers. There are up to 6 CMTs, each single one contains 2 Digital Clock Managers (DCMs) and one PLL. The role of a DCM is primarily to eliminate clock skew and distribution delays. It can also ensure phase shifting and clock multiplication and division. A PLL is based on voltage controlled oscillator (VCO) that operates from 400 MHz to 1080 MHz.

As far as memory blocks are concerned, every Spartan-6 FPGA supports between 12 and 268 dual port RAM blocks in addition to the distributed RAM within CLBs. This corresponds to a total of 6179 Kb memory capacity.

3.1.2. Altera Cyclone-4 FPGA

A Cyclone-4 FPGA is based on 60nm process technology and integrates up to 150000 LEs [15]. As seen in Figure 2.9, these latter are gathered in 16-group blocks called Logic Array Blocks (LABs). Each LE consists of a 4-bit LUT that can perform either combinatorial or arithmetic operations and a D-Flip-Flop for sequential operations. Also, an LE can operate in arithmetic mode and perform a 2-bit full adder and also a basic carry chain.

The interconnection network is organized in 2 levels (Figure 2.9); *local interconnect* and *global interconnect* [15]. The first one is used to transfer signals between LEs in the same LAB. The global interconnect is organized in column and row lines that drive signals between LABs and also between the other FPGA elements. Direct link interconnect is also provided to connect adjacent elements so as to minimize the use of columns and rows, providing higher flexibility and higher speed performances.

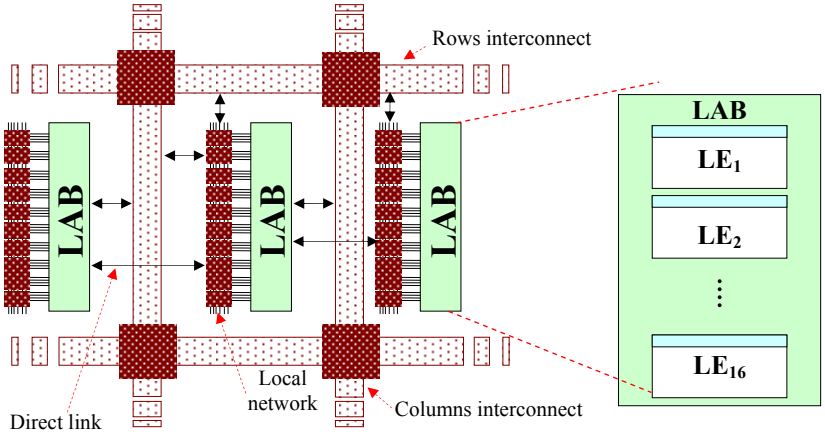


Figure 2.9: SRAM-based CYCLONE-4 FPGA fabric

Cyclone-4 FPGAs include up to 360 embedded multipliers organized as columns. Each one can be configured to perform as one 18x18-bit multiplier or two 9x9-bit multipliers. In order to reach high speed performances, each multiplier is pipelined and associated with input/output registers.

The number of I/Os can reach up to 532 and up to 11 banks are available. Each I/O is associated with an IO Element (IOE) that contains a set of bidirectional buffers, registers and programmable delays. Various single-ended, differential and voltage referenced standards are supported.

As for the clocking resources, Cyclone-4 provides up to 15 dedicated clock pins and up to 30 global clock networks that feed the whole FPGA elements. Besides, cyclone-4 includes up to 8 PLLs that perform general purpose clock management such as multiplication, division, phase shifting and programmable duty cycle. Each PLL is based on a VCO that operates from 600 MHz to 1300 MHz.

As for the on-chip storage elements, Cyclone-4 integrates up to 6480 Kb embedded RAM blocks that can be configured to provide various memory modes such as RAM, shift registers, ROM and FIFO modes.

3.2. Antifuse based technology

The Antifuse technology is based on a one-time non volatile FPGA configuration. The Actel Axcelerator, EX and SX-A series belong to the most recent Antifuse FPGA families [4]. The internal architecture of the Axcelerator FPGA is discussed, [11].

An Axcelerator FPGA is based on 150nm process technology and consists of a sea of Logic Modules (LMs) and Antifuse interconnection elements. There are up to 32000 LMs and two types of

LMs are available: the Register cell (R-cell) and Combinatorial cell (C-cell). An R-cell consists in a D-Flip-Flop. A C-cell can implement up to 4000 combinatorial functions with up to 5 inputs.

The arrangement of these modules and the interconnection network are based on hierarchical approach where combinations of two C-cells and one R-cell (C-C-R) form a cluster. In a higher level of hierarchy, two clusters form a super-cluster. A set of 336 super-clusters and 4 memory blocks form, at the next level, a Core Tile. The interconnections are also ensured with respect to the level of hierarchy. Thus, *local direct interconnects* are used for elements inside a cluster, *fast interconnects* are user to link adjacent super-clusters, horizontal and vertical *tracks* are used to link Core Tiles. Figure 2.10 gives an idea of module and interconnection arrangements.

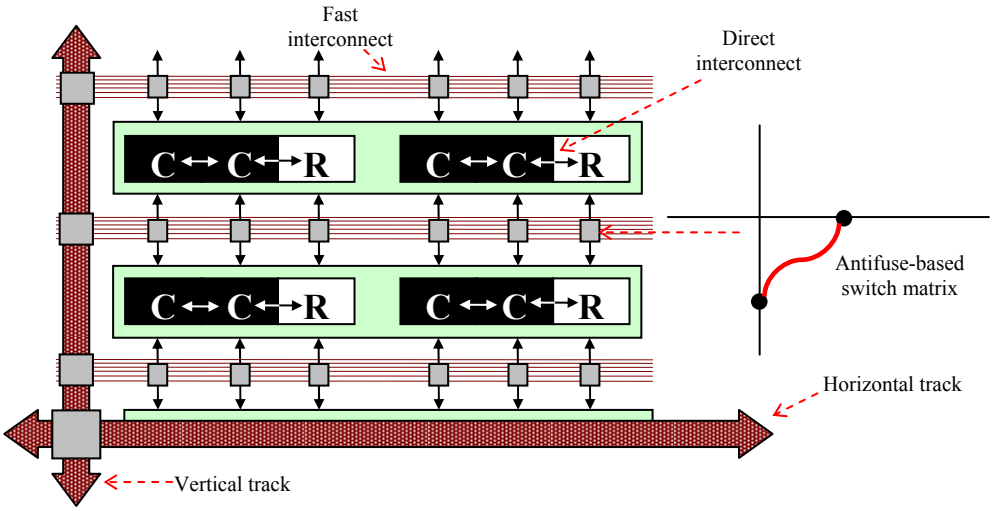


Figure 2.10: Axcelerator Antifuse FPGA fabric

Depending on the device profile, the number of I/Os in an Axcelerator FPGA can reach up to 684 I/Os that support at least 14 different single-ended, differential, voltage-referenced standards. They are organized into banks, with 8 banks per device. Each I/O block contains input, output and enable registers and combination of 2 I/O blocks form an I/O cluster.

As for the clocking resources, each device contains 8 PLLs that perform functions such as frequency multiplication, division, programmable delays and skew minimization. The input frequency of each PLL ranges from 14 MHz to 200 MHz and its output from 20 MHz to 1 GHz.

An Axcelerator FPGA provides up to 295 Kb embedded memory blocks that are based on RAM/FIFO memories. No arithmetic (DSP) bocks are included.

3.3. Flash based technology

The Flash based technology ensures a non volatile reprogrammable configuration mode and leans on specific flash-based configuration switches. In the following, the Actel Fusion Flash-based FPGA is presented [10].

Based on a 130nm process technology, its internal FPGA fabric contains up to 38400 VersaTiles (VTs). Due to the flash connections and compared to SRAM based FPGAs, each VT can implement either a combinatorial (3-bit LUT) *or* a sequential (D-Flip-Flop) functions. Figure 2.11 gives an overview of the Fusion FPGA fabric.

Within this FPGA, 4 levels of routing hierarchy are available. Thus, at the lower level, *ultra-fast local lines* make the connection between the output of each VT to every inputs of the eight surrounding immediate VTs. At the next level, e*fficient long line* resources ensure the routing of longer distances spanning vertically and horizontally up to 4 VTs. Then, *very long line* resources ensure the routing of high distance and very long nets. Finally, at the highest level, *VersaNet* global networks are used to drive global signals such as clocks, reset signals or other signals that require low skew.

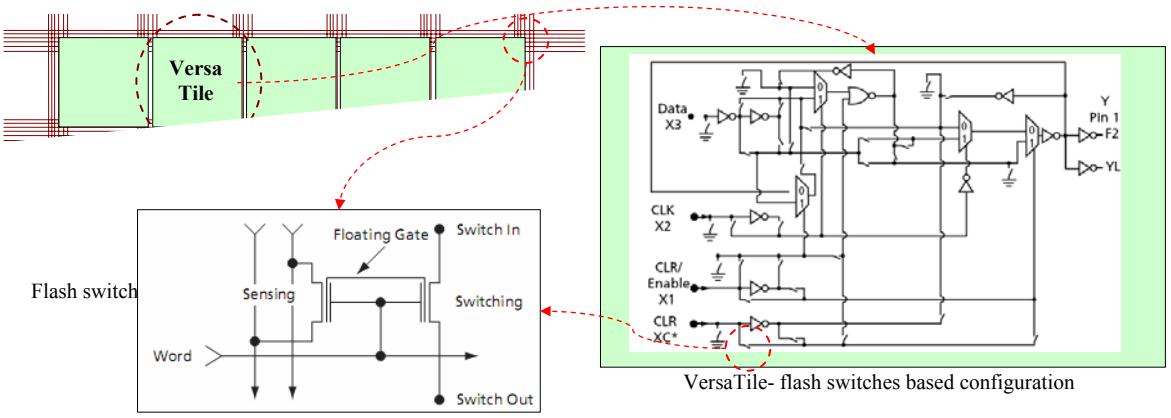


Figure 2.11: Flash-based Fusion FPGA fabric

Fusion devices provide up to 252 I/Os and include a set of I/O tiles, organized as I/O banks. Each tile supports a large number of standards including single-ended, differential and voltage referenced I/Os.

As for the clock resources, Fusion FPGA integrates a collection of on-chip resources that create, manipulate and distribute the clock signals. To this aim, an internal RC oscillator and a Crystal oscillator are integrated and can generate up to 100 MHz clock source without external component. To manipulate the clock signals, 2 PLLs are integrated for multiplication, division, synchronization and phase shift. The input frequency of each PLL ranges from 1.5 MHz to 350 MHz and its output from 0.75 MHz to 350 MHz. The distribution within the global *VersaNet* lines is ensured by 6 Clock Conditioning Circuit (CCCs) that can also perform as PLLs.

The embedded memories consist of Flash blocks (up to 8 Mb), SRAM blocks that can perform as FIFOs (up to 270 Kb), FlashROM blocks (1 Kb).

Another particularity of Fusion FPGA is the integrated mixed signal peripherals. Indeed, this FPGA integrates a 12-bit programmable successive approximation ADC. Associated with an analog multiplexer, it can convert successively up to 30 analog signals [10]. Analog quads are integrated and used to precondition and to adapt the analog inputs to the ADC voltage range. Applications and implementations of this integrated module are reported in [18]-[23]. In chapter 3, these analog peripherals will be deeply discussed.

3.4. Feature summary

The following table summarizes the main FPGA characteristics that have been discussed previously. The provided values correspond to the maximum available values depending on the FPGA series.

Table 2.1: Overview of the discussed FPGA characteristics

Technology	SRAM		Antifuse	Flash
Family / Elements	Xilinx Spartan-6	Altera Cyclone-4	Actel Axcelerator	Actel Fusion
Process technology	**45 nm**	**60 nm**	**150 nm**	**130 nm**
Logic blocks	**11519 CLBs (147443 LCs)**	**150000 LEs**	**32000 LMs**	**38400 VTs**
Clocking performances – PLL output frequency range	**400-1080 MHz**	**600-1300 MHz**	**20-1000 MHz**	**1.5-350 MHz**
I/Os	**576**	**532**	**684**	**252**
Arithmetic (DSP) blocks	**180 DSP slices**	**360 multipliers**	**-**	**-**
Memory blocks	Distributed RAM: **1355 Kb** RAM blocks: **4824 Kb**	RAM blocks: **6480 Kb**	RAM blocks: **295 Kb**	RAM blocks: **270 Kb** Flash blocks: **8000 Kb** Flash ROM blocks: 1 **Kb**
				Analog peripherals: ADC, Analog Mux, Analog quads

4. Design tools

As FPGA features are becoming more and more sophisticated and the diversity of integrated elements is increasing, CAD software tools have become mature as well. Today, FPGA vendors provide a fairly complete set of design tools that allow high quality design process starting from the hardware description, using VHDL or Verilog languages, to the final bitstream generation [5], [17], [24], [25]. An overview of a typical FPGA design process is presented in Figure 2.12.

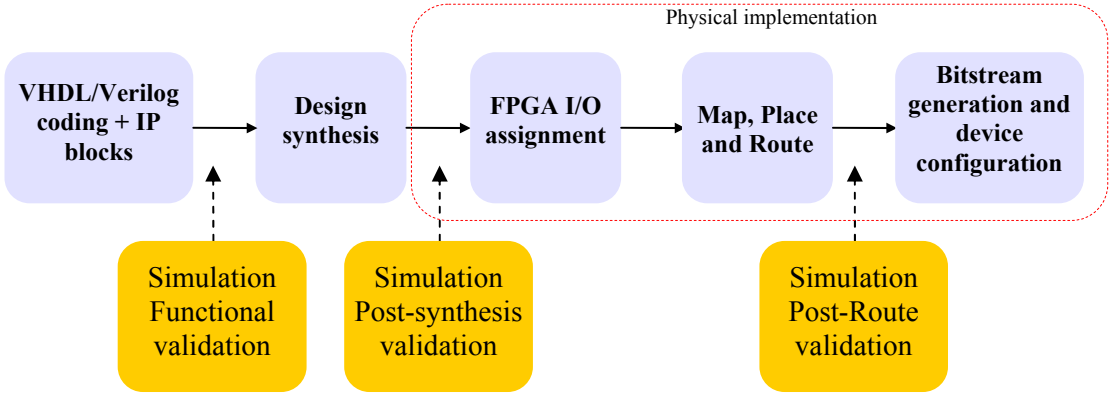


Figure 2.12: Simplified synoptic of FPGA design process

Generally, design tools include hardware design and verification tools (VHDL/Verilog editor, synthesizer, place/route and physical implementation tools), vendor libraries (IP cores) in addition to simulation and debugging tools. Some examples are the Integrated Software Environment (ISE) tools from Xilinx, Quartus tools from Altera and Libero Integrated Design Environment (LiberoIDE) tools from Actel. All of them provide flexible and complete design features with additional associated tools for simulations (e.g. ModelSim tools) and for debugging (e.g. ChipScope tools from Xilinx).

Furthermore, to suit SoC trends, FPGA vendors provide software tools that include software development tools (editor, compiler, assembler, linker and debugger), software vendor IPs and processor customization tools. For example, Xilinx provides Embedded Development Kit (EDK) platform, Altera provides Embedded Design Suite (EDS) platform and Actel provides SoftConsole platform [25].

5. Contribution of FPGAs in complex AC drive applications.

Nowadays, it is commonly accepted in the field of power electronics and drive applications, that digital based control solutions are becoming the natural and systematic resort. Indeed, compared to their analog counterparts, the implementation of a digital based controller has several advantages such as flexibility, re-programmability, reduced time-to-market and the possibility to implement complex control algorithms.

However, to achieve high performances and try to compete with performances provided by analog solutions, the selection of the appropriate digital solution is based on many criteria and implementation demands. Some of the most challenging criteria are: high performance in terms of control reactivity and bandwidth, the implementation of algorithms that insure complex treatments, high level of integration (use of the same digital solution to process many heterogeneous tasks), reduced time-to-market, system confidentiality and low cost.

In this section, a deep understanding of how FPGAs are highly adapted to these applications and these demands is proposed. This has been already made in a wide range of applications (e.g. [17]-[51], [53]-[59]) where FPGAs have successfully accomplished their role with regards to the expected performances. In these previous evaluations, it has been demonstrated that FPGAs have a significant contribution and allow the implementation of advanced high performance control strategies. For instance, in [24] and [25], authors give a clear classification of applications depending on the switching frequency. Two categories are then defined: *high demanding* applications and *constrained switching frequency* applications.

The first category corresponds to high switching frequency applications (above 100 kHz) where a high level of parallelism and short execution time are required. The use of FPGAs is then mandatory in this case. As for the second category where the switching frequency is limited (less than 100 kHz due to power switch losses), it has been stated that by exploiting the rapidity and the high integration density of FPGAs, it is possible to implement advanced high performance control strategies with the possibility to include complementary tasks such as health-monitoring and diagnosis. Relevant examples are oversampling controllers [32], multi-level multi-phase PWM controllers [45] and multi-system controllers [54], [55].

Additionally, FPGAs have their value-added to implement many other advanced controllers like, dynamically reconfigurable controllers [60], [62] and Predictive controllers [56], [61]. Another challenging field where FPGAs are one of the final resorts is the Hardware-In-the-Loop (HIL) applications. Here again, many promising FPGA based developments are achieved for Real time simulation purpose (e.g. [49]).

When it comes to time-to-market, it is clear that FPGAs have their own value-added since they allow rapid-prototyping solutions in the one hand, and they allow optimized design process in the other hand. This is possible with the help of the significant progress in terms of design tools and also design methodologies. Also, in terms of system confidentiality, today's FPGA technologies ensure highly secure designs avoiding any bad-intentioned duplication.

To make a connection with the context of the proposed work, it has been chosen to evaluate quantitatively the FPGA solutions in the case of complex control applications such as the developed sensorless AC drive application using the Extended Kalman Filter. In the following, two evaluation axes have been emphasized: the evaluation in terms of control performances and in terms of system integration. Finally, in the same context of complex control applications, some of the most important FPGA implementation constraints to manage have been discussed.

5.1. Evaluation in terms of control performances

To start with, it is fair to mention that a wide range of these complex control applications are mostly carried out with software solutions such as DSP controllers. Main reasons of this statement are: a good software flexibility, a rapid-prototyping and an easy way of coding (the familiar C/C++ coding). All these reasons allow an easy implementation of complex tasks. However, it is more than

important to stress that in some cases, their timing performances can be easily a severe drawback. This is definitely due to DSP-based architectures which are fixed leading to serialize the treatment. Consequently, the more complex the implemented controller, the longer the execution time.

For example, in [93]-[106] the total execution time has been evaluated from 90µs to 500µs. In order to make a quantitative comparison to the FPGA solution, author has tested the EKF-based sensorless speed controller on a DSP solution. This sensorless controller is applied to a synchronous AC machine. Note that the complexity is the same in both hardware and software cases. More details and in-depth studies are provided in [Appendix D].

Figure 2.13 presents the performances obtained with the DSP solution. The case of a elementary PI-regulator, the case of an AC drive current controller and the case of the sensorless controller are presented. These results are obtained with a TI TMSF2808 DSP [64] (100MHz, 32Bit, 12-bit ADC, 2x16-bit multiplier, 16Ko RAM memory). They indicate that for different levels of complexity and for a fixed architecture, the execution time increases.

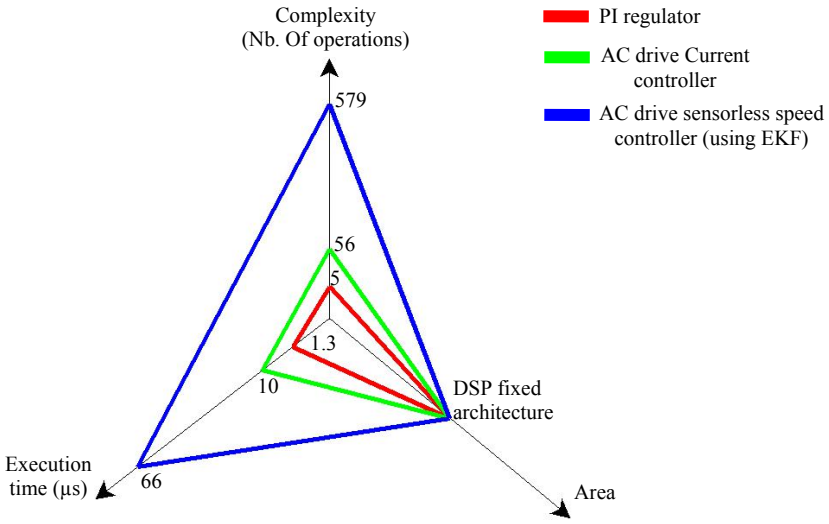


Figure 2.13: Performances of the DSP solution in case of complex sensorless application

Now as the performances of the used DSP are measured, it is important to analyze how they act and influence the control performances. In fact, from the control point of view, such long execution time imposes the use of lower sampling frequency in the one hand and introduces a delay in the closed loop. Consequently, the bandwidth and the quality of the implemented controller are downgraded. This is especially true in case of high speed AC drives such as avionic systems [65] where high frequencies are operated and high control performances are required. With the developed DSP solution, the obtained frequency response has been measured and plotted in Figure 2.15.

In order to overcome these limits, let's analyze the appropriate solutions. In fact, staying always with software solutions, many possibilities can be of great interest in order to speed up the timing performances. For example, the first reaction could be the reduction of the complexity of the algorithm which leads to make compromises with the algorithm performances. In a more systemic point of view, the use of a high clocking frequency DSP can be a simple and relevant resort but the problem that matters is the availability of DSP which is dedicated to control applications (i.e. fast DSP controller). Also a parallelization of tasks using multi-DSP structures is possible which, in contrast, downgrades the system integration. Also, the way of C-coding and the use of Assembly-coding have their own contribution. Finally, a completely different interesting alternative to these key-solutions is the use of hardware solutions such as FPGAs.

In fact, FPGAs are outperforming today's software solutions by exploiting the inherent algorithm parallelism. Consequently, implementing such hardware solution give the possibility to develop architecture that are fully dedicated to the control algorithm. Thus, allying today's FPGA high speed performances with parallelism, leads to a drastic reduction of the execution time. Consequently,

in terms of control performances, a quasi-instantaneous control is ensured which speeds up the control reactivity and bandwidth.

With the same purpose as before, the EKF-based sensorless speed controller has been implemented on an FPGA. The development process is described in chapters 4, 5, 6 and 7. This development is organized according to the design methodology that will be presented afterwards. The FPGA architecture is synchronized with a 50 MHz clock frequency which gives the timing performances in Figure 2.14. It can be seen that an additional area degree of freedom allows the reduction of the execution time.

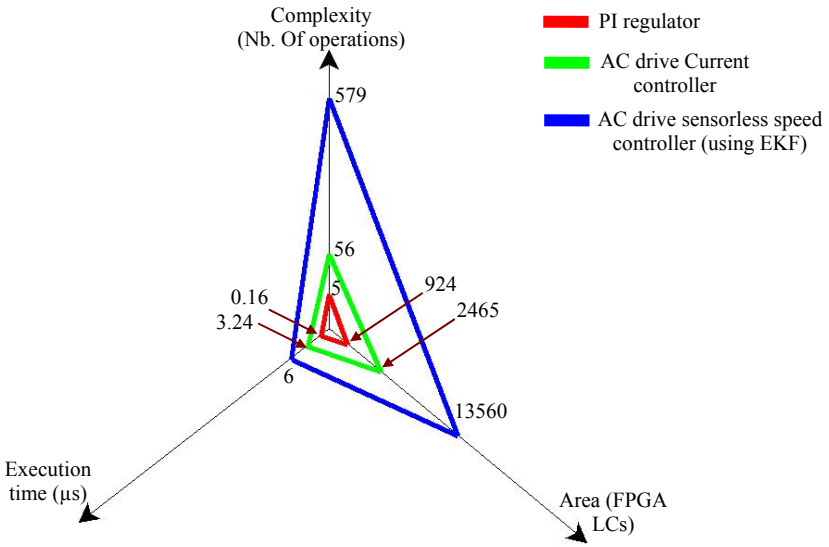


Figure 2.14: Performances of the FPGA solution in case of complex sensorless application

The impact of the execution time on the control performances and bandwidth (in both software and hardware cases) has been quantified in the case of the developed application. The obtained results are deeply presented in [Appendix D]. Figure 2.15 shows the obtained frequency responses. They indicate that a short execution time improves the control bandwidth.

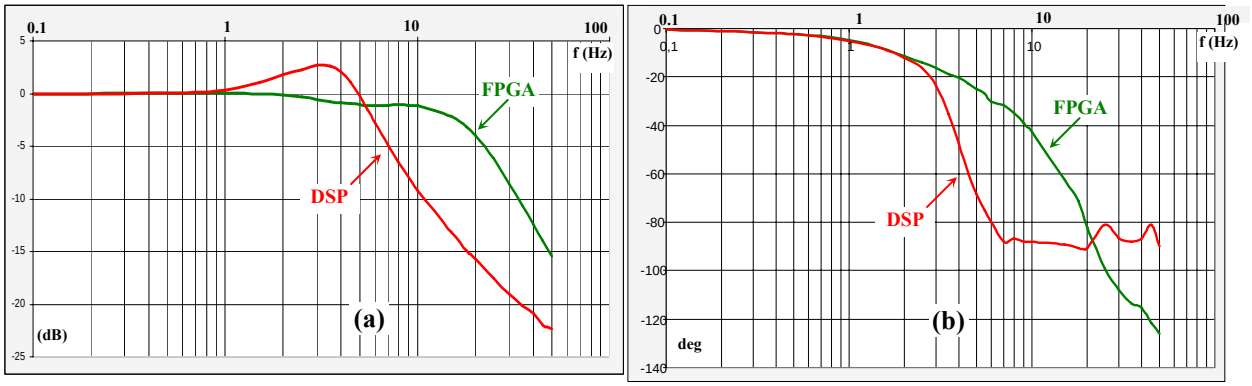


Figure 2.15: Evaluation of the control bandwidth in case of FPGA-based controller and DSP-based controller – Case of high speed AC drive **(a)** : Frequency response (magnitude) **(b)** : Frequency response (phase)

5.2. Evaluation in terms of system integration

As far as the integration criterion is concerned, the available huge number of elements within today's FPGAs, allows them being perfectly suited to high complexity algorithms. Furthermore, allying high integration and high speed performances allow the implementation of algorithms that can be used quasi-simultaneously in different applications. For example, when talking about sensorless

control application, it is possible to implement an EKF that can be used to observe the state space vector of many systems. Let's take for example the control system presented in Figure 2.16 where a sensorless control for an AC drive ([65]-[69]) and a sensorless control of a Rectifier [70] are made. Normally, two different EKF modules are to be implemented for each application but, when using an FPGA solution, it is possible to develop a common architecture that is adapted for both applications almost without impacting the execution time and the used FPGA resources. Thus, using a set of multiplexers, it is possible to drive the right signals to the EKF so as to estimate the right vector of the right system.

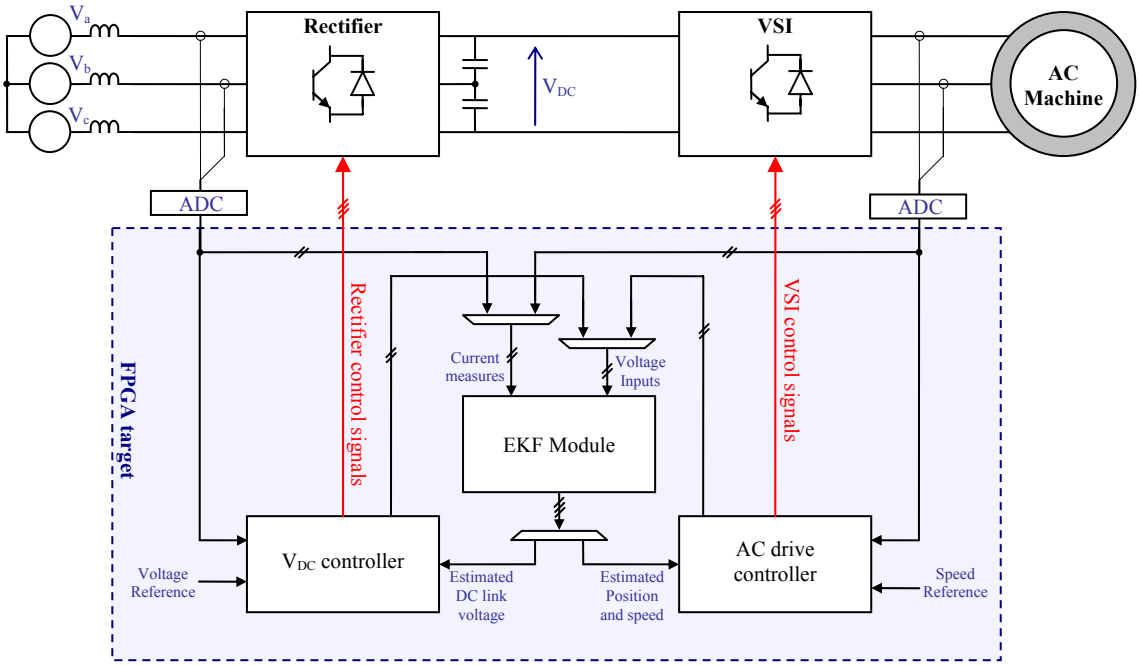


Figure 2.16: Example of sensorless control for mixed-system application

Another interesting example is the sensorless control of multi AC drives that is presented in Figure 2.17. Here again, both the EKF and controller can be gathered (or factorized) so as to control all the systems without impacting their corresponding performances.

A slightly different integration aspect is the possibility (that is ensured by today's FPGAs) to implement mixed hardware and software treatments in the same device. Indeed, with the available processor cores, designer is able to partition the algorithm and decide which treatment is to be made in hardware and which treatment is to be made in software. To do so, a deeper investigation in terms of control performances has to be achieved and co-design approaches and methodologies are to be adopted [73]. This aspect in not covered in the proposed work and is the main subject of an associate thesis work [74].

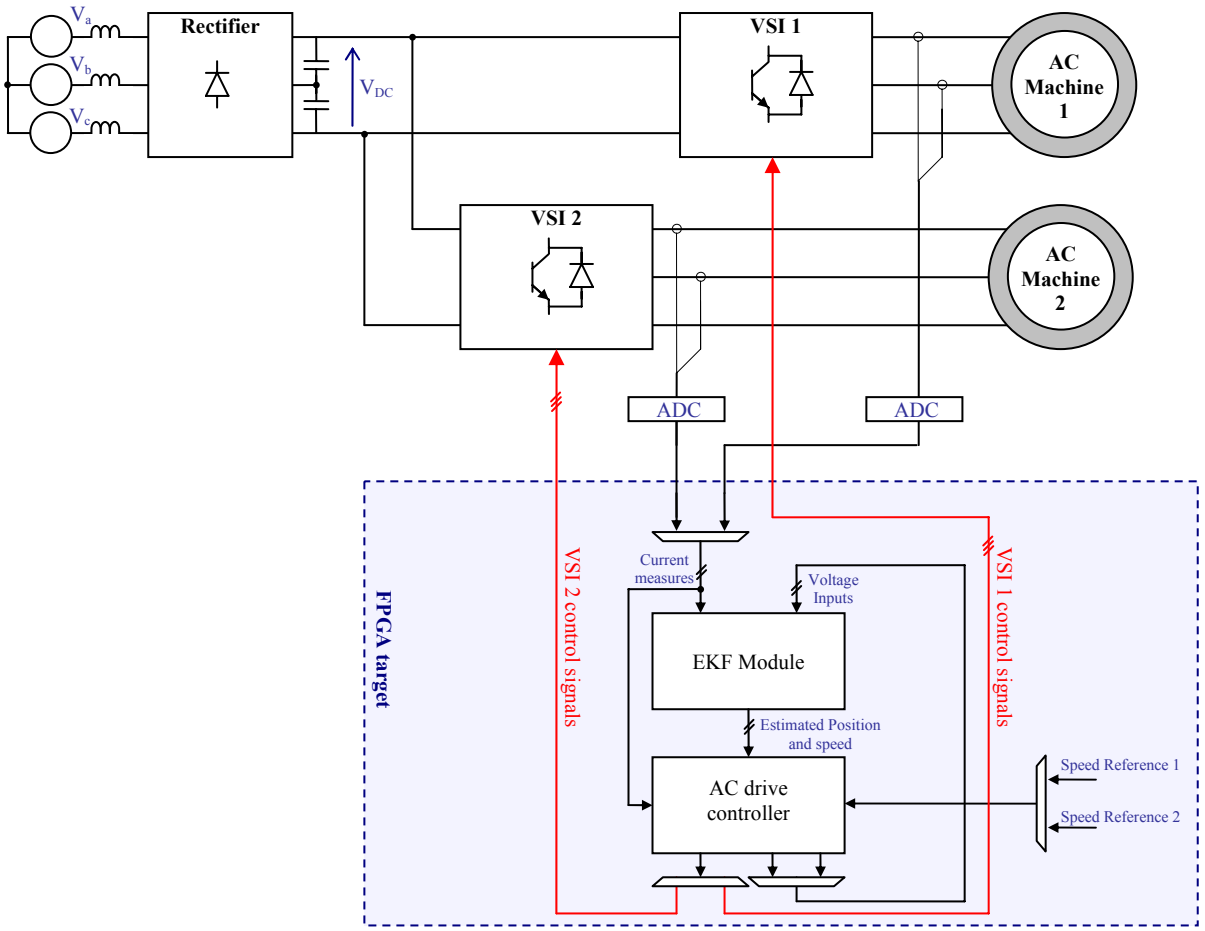


Figure 2.17: Example of sensorless control for multi-system application

5.3. FPGA implementation constraints

Now, having in mind these advantages, there are anyhow some implementation constraints to manage when using FPGA solutions to implement complex digital controllers. At first, the increasing complexity of the algorithm induces the need of numerous FPGA resources. This may seem confusing with the previous argumentation, but in the case where the system constraints impose the use of a specific FPGA (with limited resources) in a specific application, the implementation of complex algorithms can be easily inappropriate. As an example, in an avionic actuator control application where a specific Actel Flash based Fusion FPGA is needed [18]-[23], the implementation of complex algorithms (for example an Extended Kalman Filter) is difficult since the available resources are limited and no hardwired arithmetic blocks are provided within the device. Consequently, a specific care has to be taken so as to design optimum FPGA architectures that use a minimum of operators for implementing the necessary operations in a short execution time.

The data word-length in FPGAs is also another significant concern. In the case of a DSP solution, the data size doesn't matter since it is fixed (32 bits or 64 bits depending on the used device) and both fixed-point and floating point formats are allowed. This is clearly not the case of FPGAs since the word-length is fully customized. Thus depending on the complexity of the implemented algorithm, the larger is the data size, the heavier is the corresponding FPGA architecture. For this reason, designer has to make the choice of word-length that suits the available FPGA resources without downgrading the control stability and precision. This is typically true when manipulating fixed point data format. Otherwise, in the case of manipulating floating point data format, the consumed FPGA resources depend only on how many floating point operations are processed. As a reminder, recent VHDL standards offer the possibility to implement floating point arithmetic using the VHDL 2008 libraries [71].

Another concern when using FPGAs is the operating clock performances. Indeed, a placed and routed complex algorithm, where many successive arithmetic operations are done, introduces many propagation delays. Consequently, the operating clock frequency is highly limited. To overtake this issue, the pipelining of the developed architecture is essential. In fact, a fully pipelined architecture where registers are placed between operators makes sequential the signal routing and leads to low propagation delays. This has definitely the credit of ensuring a high operating clock frequency.

To end with, the contribution of FPGAs and the induced implementation constraints are to be considered and balanced with the expected level of control performances. These design considerations have to be taken into account during the design process. Thus, it is quite mandatory to use a rigorous and well-structured design methodology that allows designing efficiently the FPGA-based controller that suits all the process and performance demands.

6. FPGA design methodology for control applications

Now as the advantages of FPGA solutions and their implementation constraints are both initiated, the main challenge is how to reach efficiently the demanded control performances. In fact, a well-developed FPGA solution has to be based on a perfect adequation between the control algorithm and its corresponding FPGA architecture without loosing the potential parallelism. Thus preserving the inherent parallelism of the algorithm allows high timing performances which enhances the control performances.

A well developed FPGA solution has also to satisfy the implementation constraints. To this aim, it is clear that designer has to manage many design considerations (optimization of the complexity, choice of the data word-length, pipelining of the architecture…) at different stages of the design process. Consequently, these considerations require from designer to master several different knowledges and qualifications such as micro-electronics, control, signal processing and electrical system theories. This is particularly true when implementing complex control applications that need a narrow link between control engineering and FPGA design domains.

For these reasons, the use of a rigorous and well-structured design methodology is of prime necessity. This methodology should consist of a set of steps and rules to be followed in order to optimize the time-to-market and make efficiently the design process more manageable and less intuitive. By this way, several authors have proposed and formalized interesting design methodologies [33]–[37], [62]–[63], having always in mind to reach similar objectives.

As it can be seen in Figure 2.18, the particularity of the presented design methodology consists in providing an enlarged design process that starts from the preliminary system specification to the final experimental validation. In addition, a notable distinction between the development of the algorithm and the development of the FPGA architecture is made. This distinction has the credit of making the algorithm as independent as possible on the used digital device. For instance, once the developed algorithm is achieved, either a hardware solution (FPGA) or a software solution (DSP) can be chosen for the digital implementation. Furthermore, this distinction can lead to a separation between the needed designer qualifications. For example, the algorithm development may be realized by control engineers and the FPGA development by a micro-electronics expert.

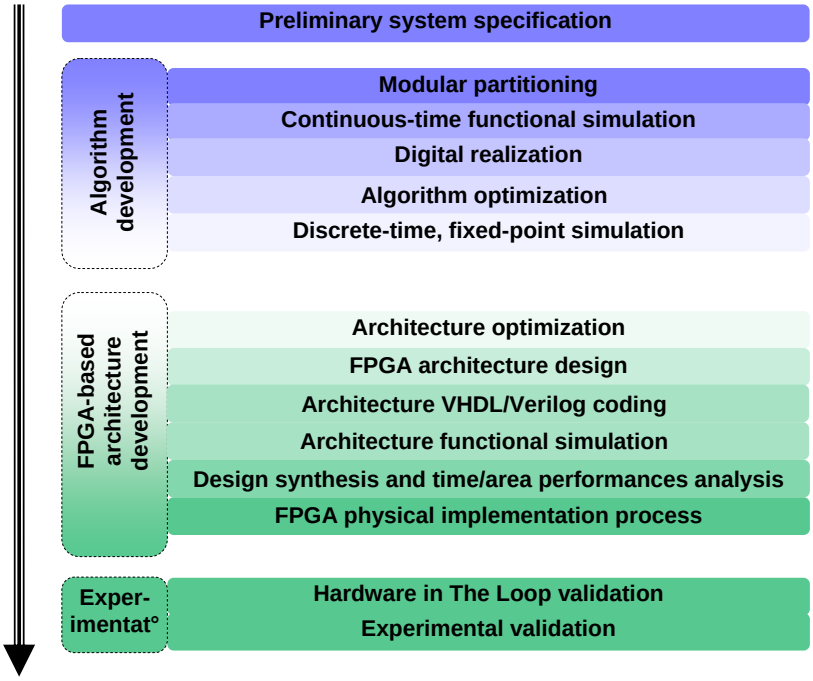


Figure 2.18: The proposed FPGA Design methodology

From a more technical point of view, the proposed design methodology includes optimization assumptions that ought to be achieved so as to adapt the algorithm complexity to the available FPGA resources. As it will be discussed later, this optimization is done during the algorithm development process and during the FPGA architecture development one. For the first case, this consists in reducing the computational cost of the algorithm (reduction of the number of processed operations). For the second case, this consists in studying the data dependency of the algorithm and finding out the potential factorizations that lead to the use of a minimum of operators that process a maximum of operations. The aim here is to develop an FPGA architecture that satisfies the timing and area constraints. This optimization can be achieved by applying for example the so-called Algorithm Architecture Adequation (A^3) methodology [62], [63], [72].

The proposed methodology has been overviewed in [25] and has been deeply illustrated with the developed sensorless application that is covered in this thesis report [chapters 4, 5, 6 and 7]

6.1. Preliminary system specification

First of all, designer makes a preliminary system specification regarding the whole control application. In the case of an AC drive application; this consists in making a physical specification of the control system and an algorithm benchmarking. The physical specification consists in choosing, depending on the load conditions, which AC motor is to be controlled and on which supply conditions. The physical characteristics of the final experimental platform are defined including the selection of the digital control unit, the used ADC and interface boards.

The algorithm benchmarking, consist in choosing the control strategy, the sensorless method (case of a sensorless control) and the convenient system model.

6.2. Algorithm development

The algorithm development process consists of a set of steps during which designer makes the functional validation and prepares the algorithm for the digital implementation.

6.2.1. Modular partitioning

This step is very important, especially in case of complex algorithms, and leads to strategic choices regarding the reusability and the modularity of the developed algorithm. Indeed, this step is

based on hierarchy and regularity concepts. Hierarchy is used to divide a large or complex design into subparts called modules that are more manageable. Regularity is aimed to maximize the reuse of already designed modules [34], [36], [62], [63]. As a result, the extracted reusable modules are organized in different levels of granularity and added to a specific library of control for electrical systems [34], [36], [62], [63].

6.2.2. Continuous-time functional simulation

Once the control system is designed and the algorithm partitioning is made, a continuous-time (s-domain) functional simulation is achieved using Matlab/Simulink tools. This step is aimed to simulate and verify the functionality of the complete control system.

6.2.3. Digital realization

During this step, the first task consists in making a digital synthesis of the aimed control closed loop and choosing the right sampling frequency. Two approaches are considered, the direct synthesis approach and the digital re-design approach. The first one consists in configuring the controller and synthesizing the used regulators in a fully discrete-time z-domain. This approach is suitable for high switching frequency applications.

In most cases, power electronics applications are using limited switching frequency. Thus, the re-design approach can be adopted. The latter, consists in synthesizing regulators in the continuous s-domain and then making the convenient transformation to the discrete-time domain (ZOH, Tustin, Euler…). This is typically the case of the developed sensorless control application.

The obtained digital controller (or observer) can be then considered as a digital filter that is now to be realized. The corresponding structure is then specified (direct form, cascade form, transpose form …). Then the normalization is processed. It consists in developing a per-unit algorithm where variables are replaced by their corresponding per-unit counterparts with the introduction of base-values. To this purpose, the base-value of each variable is determined according to variable nominal value and also according to the gains that are introduced by the sensors and the ADC board.

The following task is the choice of the fixed point data format. This choice can be made in two stages [40], [41]. The first one is the choice of the fixed-point format of the coefficients by studying the stability of the closed-loop. The second stage concerns the choice of the fixed-point format for the variables. To this purpose, the limit-cycle at steady state and the signal-to-noise ratio are both considered.

A simpler but more intuitive method for choosing the fixed-point format is by trial-and-error fixed-point simulations. Indeed, designer can develop the fixed-point model and then make a comparison with the floating point initial model. The format that leads to a minimum quantification error is then maintained. Another and still more intuitive way to choose the format is the use of Matlab/Simulink fixed-point tool. At the end of each simulation, this tool collects information about the processed data and displays their maximum, minimum values. It also indicates when overflows occur. Then, these data ranges help designer to choose the appropriate fixed point format.

6.2.4. Algorithm optimization

As mentioned before, an optimization is to be performed in order to reduce the number of operations. This optimization is quite mandatory in the case of the FPGA solution since the size of the developed architecture is conditioned with the complexity of the algorithm. For instance, a complex control algorithm, where many greedy operations like multiplications have to be processed, needs a rigorous and smart simplification without loosing the required performances. Another optimization example is to perform complex functions with the use of elementary operators. This is the case of CORDIC (COordinate Rotation Digital Computer [39]) algorithms where trigonometric, hyperbolic, linear and logarithmic functions are performed with the use of elementary adders, subtractors and shifters.

6.2.5. Discrete-time, fixed-point simulation

After having developed the aimed digital control algorithm and having specified the suitable sampling frequency and the data fixed-point format, designer makes a final functional verification. The latter is made by simulating the developed algorithm in the discrete-time and fixed-point domain with the help of the appropriate Matlab/Simulink tools.

6.3. FPGA-based architecture development

In the case of having chosen the FPGA target to implement the developed algorithm, designer initiates the development of the corresponding FPGA-based architecture. To make the design process less constraining in terms of time-to-market, an interesting solution consists in generating automatically the FPGA-based architecture from Matlab/Simulink, using dedicated toolboxes proposed by the FPGA manufacturers [43]. In contrast, besides the cost of these toolboxes and in the case of complex algorithms, such solution can lead to un-optimized architecture that may be inadequate to the available FPGA resources. This is the reason why, in the proposed design methodology, designer has to develop and code himself the FPGA-based architecture with the help of the following steps.

6.3.1. Architecture optimization

The optimization here consists in finding the appropriate FPGA architecture that is able to perform the developed algorithm with the respect of a set of implementation constraints. These constraints can be summarized and classified according to the Figure 2.19(a).

Algorithm constraints: They are related to the structure of the developed algorithm including its complexity and the chosen data word-length that preserves its performances. Thus, the higher are the complexity and the data word-length, the higher are the needed FPGA resources.

Design constraints: They include the algorithm modularity preservation constraint and the FPGA architecture pipelining constraint. The more modular is the algorithm, the higher is the corresponding FPGA architecture. The more pipelined is the developed architecture, the higher is the operating clock frequency and then the shorter is the execution time.

Power consumption constraints: These constraints concern typically applications such as portable and power-conscious embedded systems where low power consumption is a key-issue. Here again the higher are the clock frequency and the used hardware resources, the higher is the consumed power. This point is not maintained in the following discussion since in the case of AC drive applications, the power consumption of the digital control unit doesn't matter compared to the power consumed by the drive itself.

Area/cost constraints: They concern the hardware FPGA resources. In our case, this consists in limiting the used FPGA solutions to the low cost FPGA families. Indeed, most of the available FPGAs are classified into two categories: the low cost category and the high performance category. These categories distinguish themselves with the quantity of the integrated resources and thus their cost. As an example, the low cost Xilinx Spartan 6 FPGA contains up to 180 DSP blocks and the high performance Xilinx Virtex6 FPGA contains up to 2016 DSP blocks.

Timing constraints: These constraints concern the operating clock conditions and the execution time of the developed FPGA architecture. In the case of a control application, the control performances are influenced by the execution time. The shorter is the execution time, the larger is the control bandwidth. These timing constraints consist then in defining an execution time limit that leads to acceptable control performances. A short execution time limit induces the increasing of the parallelism and then the increasing of the needed hardware resources.

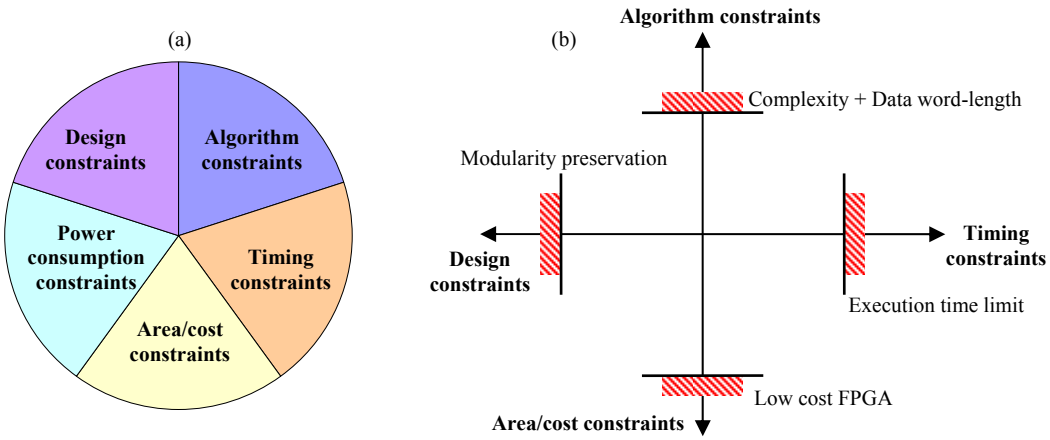


Figure 2.19: Architecture optimization – Implementation constraints
(a) Classification of the implementation constraints (b): chosen constraints

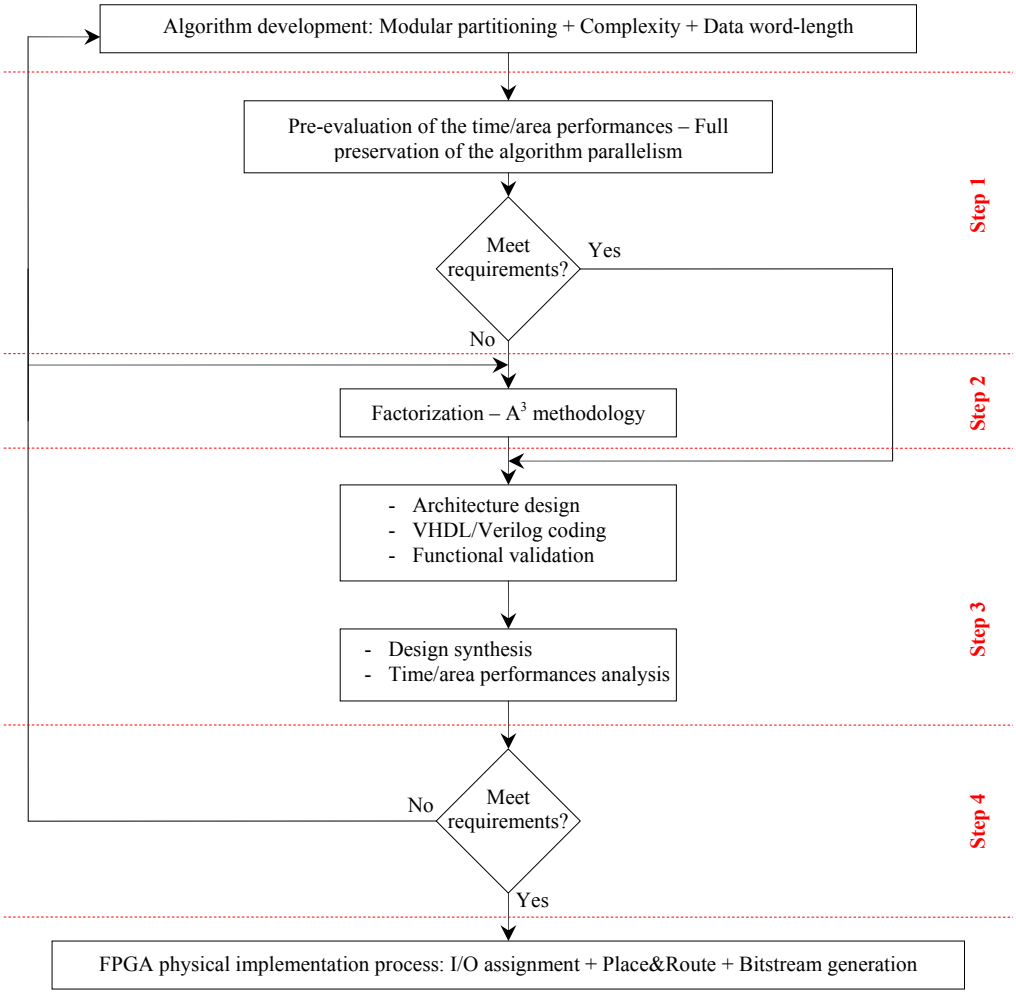


Figure 2.20: Architecture optimization procedure

In order to develop an optimization procedure, we are going to focus on the implementation constraints presented in Figure 2.19(b). In the following, a general purpose optimization procedure is then presented. Its concrete illustration can be found in chapter 6.

We first assume that the complexity of the algorithm and the data word length have been already defined during the algorithm development step. The objective then is to find an optimum between the area and the timing constraints with the preservation of the algorithm modularity.

The optimization procedure can be summarized in Figure 2.20. Four main steps can be noticed. The first one consists in making a first evaluation of the time/area analysis of the developed algorithm with the full preservation of its parallelism. This aims to verify if the corresponding FPGA architecture can be directly designed without any optimization. The second step consists in factorizing the architecture using the A^3 methodology. The third step is the design, the VHDL/verilog-coding and the functional validation of the factorized architecture. Finally, the design synthesis and the time/area analysis are made. Once the obtained time/area performances satisfy the corresponding constraints, the FPGA physical implementation process is initiated. A deeper presentation of these steps is made in the following sections.

When it comes to the A^3 methodology, the aim is to find out an optimized FPGA architecture for a given application algorithm, while satisfying area and timing constraints, [62], [63], [72]. In other words, it consists in studying the data dependency of the algorithm in order to find the potential factorization that leads to the use of the minimum number of operators that process a maximum of operations.

The A^3 methodology consists of three steps:

Design of the DFG (Data Flow Graph): this consists in making a graphical representation of the algorithm to be implemented.

Data dependency evaluation: according to the obtained DFG, the data dependency is evaluated and the potential parallelism is determined, which leads to the choice of the factorization strategy. Note that the latter reduces the hardware resources but increases the computation time since it serializes locally the treatment. The higher is the level of factorization, the longer is the execution time. A compromise between computation time and consumed hardware resources is then mandatory. The factorization is generally applied to the greediest operators in terms of consumed hardware resources like multipliers. Also, in an upper level of granularity, this factorization can also be applied at a functional module scale (thicker grain operator).

Design of the FDFG (Factorized DFG): this is a graphical representation of the factorized algorithm on which the development of the FPGA-based architecture is directly derived. This graphic introduces specific nodes called, F ("Fork"), J ("Join"), D ("Diffuse") and I ("Iterate"). These nodes are used to delimit the factorization borders [32].

A "dot product" module has been chosen as an example of illustration (relation 2.1). Figure 2.21(a) shows its DFG.

$$o(t) = x_1(t) \cdot y_1(t) + x_2(t) \cdot y_2(t) + x_3(t) \cdot y_3(t) \tag{2.1}$$

As it can be seen in the corresponding DFG, the multiplications can be performed in parallel mode which is not the case of additions. Thus, the factorization process can be applied to the multiplier operator and the obtained FDFG is presented in Fig. 21(b).

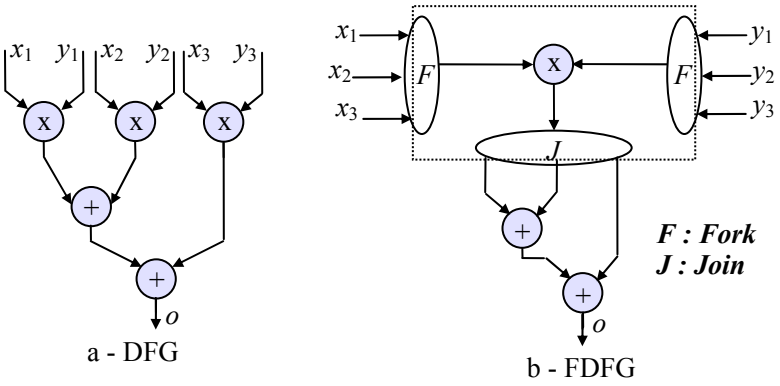


Figure 2.21: Developed DFG and FDFG – factorization of dot product operator

6.3.2. Architecture design

According to the obtained FDFG, the FPGA-based architecture is designed by replacing the FDFG nodes (F, J and I) by there corresponding operators. Thus, the node F is replaced by a multiplexer, J and I replaced by registers.

The hardware architecture of each of the developed modules (according to the adopted modular partitioning) is then composed of a data path and a control unit that are both synchronized with the global clock signal. The data path contains the used operators and data buses between them. The treatment scheduling is ensured by the control unit which is a simple Finite State Machine (FSM). The latter is activated via a Start pulse signal. When the computation time process is over, an End pulse signal indicates the end of the treatment. As an example, Figure 2.22 presents the FPGA architecture corresponding to the FDFG of Figure 2.21(b).

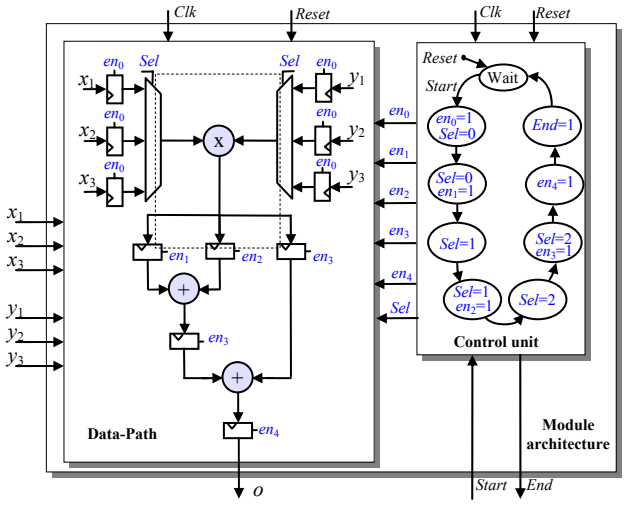


Figure 2.22: Example of a designed FPGA architecture – case of the dot-product operator

6.3.3. Architecture VHDL/Verilog coding

At this stage, the designed FPGA architecture of each module is ready to be described. The VHDL or Verilog description languages can be used.

6.3.4. Architecture functional simulation

As for the functional validation, the objective is to make the necessary simulation in order to validate functionally the developed VHDL/Verilog design. This can be achieved using dedicated tools such as ModelSim tools. The obtained simulation results can also be compared to those obtained during the fixed-point simulation in Matlab/Simulink environment.

6.3.5. Design synthesis and time/area performances analysis

The VHDL-designed and functionally validated architecture is now ready to be synthesized so as to analyze and translate it to a gate level design (using the dedicated tools, e.g. Xilinx synthesizer, Synplify synthesizer …). During this synthesis, timing and area analyses are made.

The timing analysis consists in quantifying the propagation delays between the used FPGA elements. Depending of these delays, the operating clock maximum frequency is estimated. This estimation is made with regards to the used FPGA target but also on the quality of the designed architecture. A fully-pipelined architecture leads to short propagation delays between the used FPGA elements and then to a high clock frequency. From this analysis, designer chooses the appropriate clock frequency and then estimates the execution time of the whole design.

In addition to the timing analysis, the synthesis process makes an area analysis and establishes the connection between the used operators and the available FPGA resources.

From the obtained time/area performances, four scenarios are possible:

The timing constraint and area constraint are both satisfied: The design is then appropriate for the FPGA physical implementation process.

The timing constraint is satisfied but the area constraint is not satisfied: In this case, designer has to make a choice: (i) Use of another FPGA target (with more resources but if the low cost constraint is still respected). (ii) Re-design of the architecture so as to increase the level of factorization. (iii) Revision of the modular partitioning so as to gather potential modules and then factorize the new architecture of the new module. (iv) Implementation of the optimized versions of the algorithm. (v) Reduce the data word length which leads to re-validate the algorithm development step.

The timing constraint is not satisfied but the area constraint is satisfied: In this case, designer can increase the operating clock frequency and/or decrease the level of factorization.

Both timing and area constraints are not satisfied: In this case it is mandatory to decrease the level of factorization and change the FPGA target.

6.3.6. FPGA physical implementation process

According to the design flow presented in Figure 2.12, the physical implementation process includes the I/O assignment, the place&route and then the generation of the final Bitstream that is ready to be transferred to the chosen FPGA target. These steps are commonly used by the FPGA development tools (e.g. ISE, Quartus, Libero) provided by FPGA manufacturers (e.g. Xilinx, Altera, Actel). It is also possible to achieve post-synthesis and post-route simulations that lead to validate the functionality of the obtained gate-level design.

6.4. Experimentation

6.4.1. Hardware In the Loop (HIL) validation

In order to verify a first experimental operating guarantee, an HIL procedure is recommended. The latter is an interesting realistic validation technique since it can be considered as an intermediate between a fully computer-based development validation (Matlab/Simulink, ModelSim, FPGA design tools) and a fully experimental validation (actual system platform).

The HIL procedure is carried out through a physical implementation of the developed FPGA-based architecture to be validated. The latter has to be associated with an emulation of the plant [49], [62]. In addition, a communication controller has to be implemented in order to transfer the stimuli and the probed data. This communication is to be achieved with a Host-PC in which a comparison is made between the obtained HIL results and the simulation results. Figure 2.23 highlights the synopsis of the achieved HIL test. When using a Xilinx FPGA target, the HIL procedure can be made using the ChipScope analyzer [2]. The latter is used to probe the internal signals in one hand and to configure the design in the other hand. The data transfer is made using the JTAG interface.

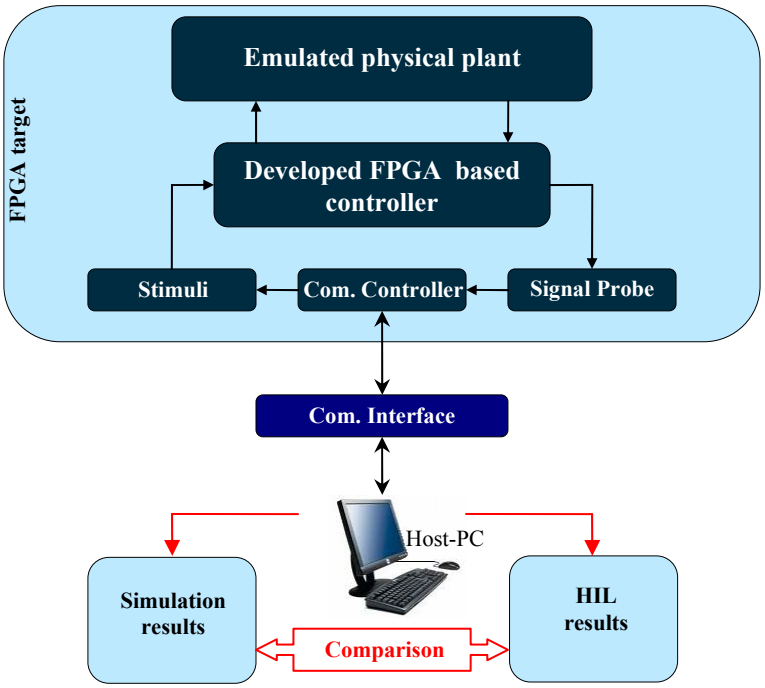


Figure 2.23: Synoptic of the HIL procedure

6.4.2. Experimental validation

The experimental validation is the final step of the design methodology and aims to make a final validation of the developed FPGA-based controller.

7. Conclusion

This chapter has made an in-depth presentation of FPGAs starting by a general purpose investigation of their relevant elements. This is followed by a deeper investigation of specific FPGA devices from each family and technology. Then an analysis has been made in order to state how FPGA-based solutions are useful in the field of power electronics and drive applications. The case of complex sensorless applications has been focused on and a discussion about the FPGA implementation constraints is done. Finally, a design methodology is provided. The latter is dedicated to power electronics and drive applications and consists in a set of steps and guidelines that help designer to develop the expected application starting from the preliminary system specification to the ultimate experimentation.

Now that this presentation is made, the next challenge is to present how to build an FPGA-based controller that is dedicated to the control of a synchronous motor drive. Chapter 3 will deal with the development of a standard FPGA based controller for a synchronous AC machine. Standard means that the implemented control system uses the measured rotor position (from a sensor) in the control closed loop. The development of a fully integrated FPGA-based controller for a Permanent Magnet Synchronous Machine (PMSM) associated with a Resolver sensor will be presented. As for the sensorless controller that uses the EKF, a deeper discussion about the development of the corresponding FPGA solution will be made. The case of a Salient Synchronous Machine (SSM) will be presented. This has been made in 4 stages according to the design methodology: system specification of the sensorless controller (chapter 4), development of the algorithm (chapter 5), development of the FPGA architecture (chapter 6) and finally the experimentation (chapter 7).

Chapter 3

Fully integrated FPGA-based controller for a PMSP associated with a resolver sensor

1. Introduction

Before the development of the aimed sensorless controller, it has been decided to start with the implementation of a fully integrated FPGA-based controller for a Permanent Magnet Synchronous Machine (PMSM) associated with a resolver position sensor [22], [23]. This start up thesis activity belongs to an aircraft application where the main objective is to develop a fully integrated FPGA solution. To suit this high system integration demand, it has been decided to use the Actel Fusion FPGA as a prototyping platform. Indeed, as presented in chapter 2, this Flash-based FPGA is considered as a System on Chip (SoC) device since it allows a mixed signal processing: analog processing using integrated analog peripherals (ADC, analog pre-scalers, analog multiplexer) and digital features such as FPGA matrix. This last can support either hardware architectures or software architectures with a processor unit (CoreMP7, CortexM1). This mixed treatment approach offers a new level of integration by allowing the use of heterogeneous functions in the same device. In addition to this features, these Flash-based FPGAs are well known for their low power consumption and their immunity to Single Event Upset (SEU) radiations. Some preliminary motor control implementations have been achieved using this SoC device and promising results have been obtained [18]-[23].

The developed controller implements the whole necessary functions in the presented Fusion FPGA SoC. The analog to digital conversion is ensured by the integrated ADC. The whole control closed loop including the treatment of resolver signals is implemented in the FPGA matrix. The sine patterns used for the coordinate transformation are stored in the integrated Flash memory block.

At this stage, it can be noticed that this controller has been fully implemented in hardware. In order to provide a fully SoC solution where software treatment is also ensured (using the processor unit), an associate thesis work has been fixed [74]. The main objective of this work is to analyze,

develop, validate and propose a hardware/software solution combining the hardware FPGA treatment and software processor treatment. Thus, additional co-design approaches are to be initiated in order to decide which part of the treatment is to be implemented in software and which part of the treatment is to be accomplished in hardware.

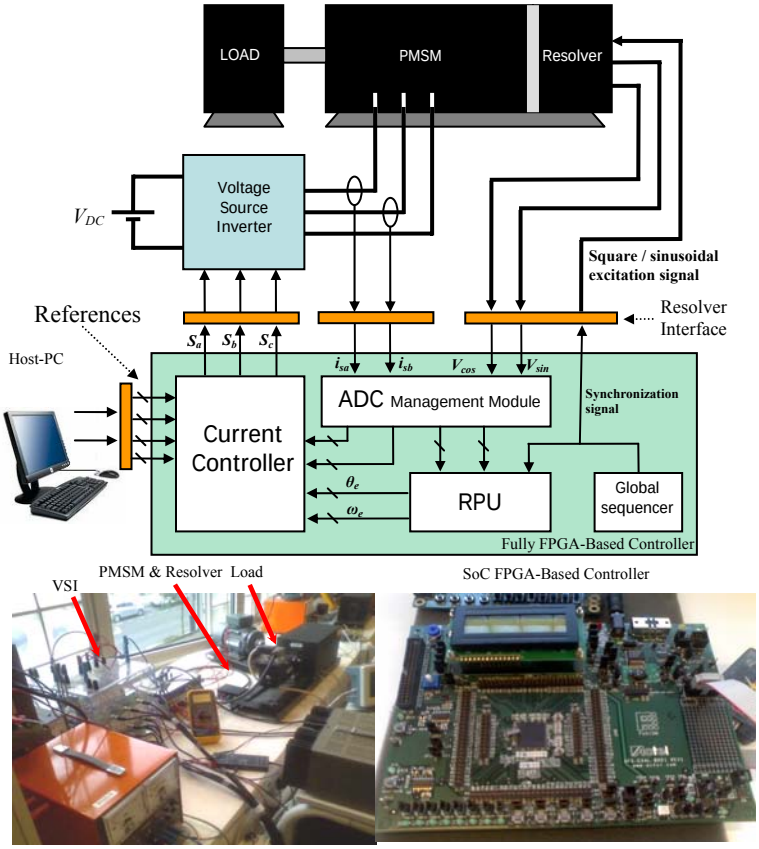


Figure 3.1: Synoptic of the proposed control system

The evaluation of the FPGA-based solution capabilities, in such AC drive application, is achieved using the experimental set up presented in Figure 3.1. It consists of a PMSM associated with a load and a resolver position sensor. This PMSM is supplied by a Voltage Source Inverter (VSI). The switching signals are provided by the FPGA-based controller. The latter includes an ADC management module, a Resolver Processing Unit (RPU) and a current controller. These modules are sequenced by a global sequencer unit which is also used to generate a synchronization signal. From this signal, the resolver interface processes either square or sinusoidal excitation signal for the resolver. The ADC management module converts the resolver signals and the stator currents using the integrated ADC. The RPU generates the rotor position and speed using the Angle Tracking Observer (ATO) algorithm. From this position, the converted currents and the reference currents, the VSI switching signals are computed by the current controller. The development of these modules has been achieved with the help of the design methodology that is presented in the previous chapter.

In addition to these tests, a compensation of the ADC limitations is made. Indeed, the used Fusion FPGA integrates only one ADC which, associated with an analog multiplexer, is able to convert up to 30 analog signals. Consequently, the conversion is serialized and these analog signals are not sampled at the same time which introduces a Sampling Synchronization Error (SSE). The impact of this error has been measured and compensation procedures have been presented.

This chapter is organized as follows. At first, a brief presentation of the integrated ADC and its implementation is discussed and the problem of synchronization is positioned. The third part presents the resolver sensor and deals with the FPGA-based RPU. In this part, a presentation of the chosen ADC SSE compensation method for resolver signals is made in addition to the evaluation of the RPU noise rejection. Some experimental results are then provided so as to validate this RPU treatment. The following part presents the FPGA-based controller which uses an ON/OFF current control strategy

[62], [75]. This part includes the implemented ADC SSE compensation method for PMSM stator currents. In order to prove the efficiency of this fully integrated solution, some experimental results are provided, followed by time/area performance analysis. Finally, author makes a comparison, in terms of execution time delay, between the FPGA-based controller and its corresponding software-based one. The influence on the control quality is discussed in both cases. This comparison aims to stress the importance of using FPGAs to enhance the control reactivity. This last point has to be taken into account when designing a SoC solution. Indeed, designer has to think about the right hardware/software partitioning that increases the integration density without loosing the control performances [74].

2. Description and implementation of the FPGA integrated ADC

The Fusion FPGA integrates a 12-bit programmable successive approximation ADC (SAR-ADC). Associated with an analog multiplexer, it can convert successively up to 30 analog signals, [10]. Analog quads are used to precondition and adapt the analog inputs to the ADC voltage range. Figure 3.2 gives an idea about the organization of the analog module.

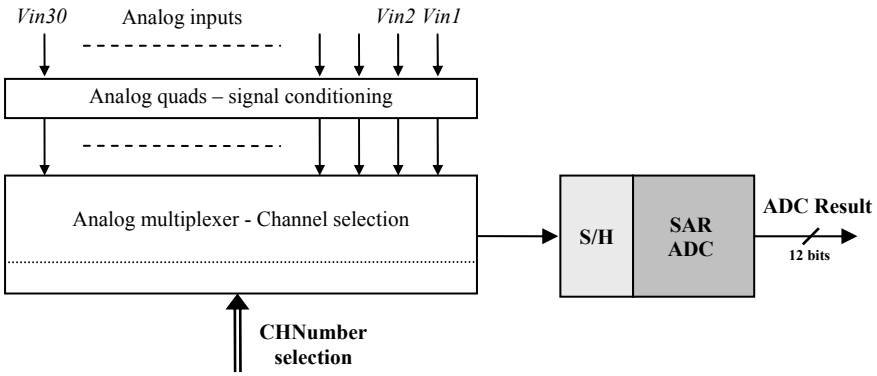


Figure 3.2: Analog module overview

The resolution of the ADC is adjustable and can be set to 8, 10 or 12 bits mode. The conversion time is programmable and depends on the global system clock, the conversion resolution and the sample and hold (S/H) time.

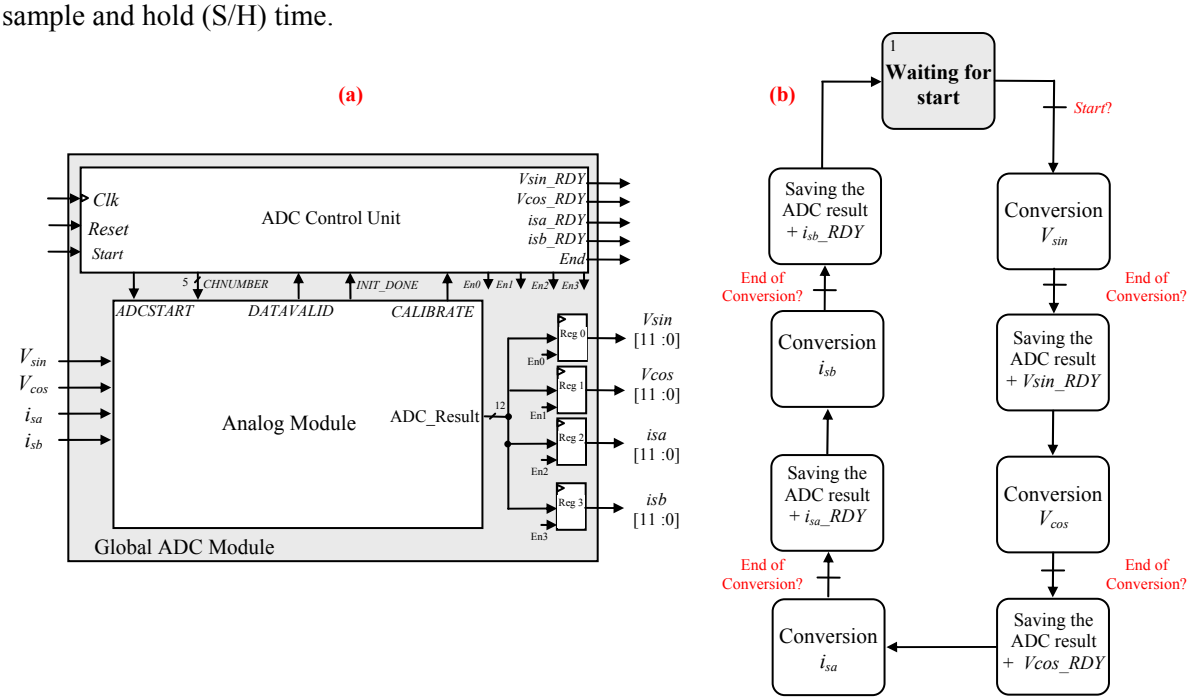


Figure 3.3: Synoptic of the implemented Global ADC Module
(a): FPGA architecture (b): State diagram

Figure 3.3 highlights the operating principle of the implemented *Global ADC Module*. It includes the analog module (Figure 3.3(a)) and four registers in which is stored the ADC result corresponding to each analog input. In the proposed application, the ADC is used to convert successively resolver signals (V_{sin}, V_{cos}) and the PMSM stator currents (i_{sa}, i_{sb}). The conversion control is ensured by the ADC control unit. The corresponding state diagram is shown in Figure 3.3(b). At the end of each conversion cycle, the ADC result is stored in the corresponding 12-bit register and at the same time, a corresponding *xxx_RDY* flag signal is activated.

The main limit of the proposed ADC structure is the sequential conversion process. Indeed, since there is only one ADC, the conversion is serialized. Consequently, this induces a global computational delay, which increases proportionally with the number of converted signals and depends on the conversion time. Furthermore, with the proposed structure, analog signals are not sampled at the same time, so, a Sampling Synchronization Error (SSE) is introduced. In the proposed application, the influence and the way-to-compensate this error have been studied for resolver signals in the one hand and for the stator PMSM currents in the other hand.

3. Resolver Processing Unit

3.1. Resolver sensor description

A resolver sensor is an electromagnetic position sensor similar to a rotary transformer with a rotating excitation winding (also called reference winding) and two 90 degrees shifted stator windings, [76]-[80]. Figure 3.4(a) gives an overview of resolver construction scheme. Often used in niche applications like space and aircraft industry systems, resolver position sensors are more efficient compared to optical sensors (absolute and incremental encoders), thanks to their large operating speed range, their low volume and their ability to work in noisy and harsh environments (rugged construction, noise rejection, operating in a large temperature range).

The excitation winding is supplied by a high frequency square or sinusoidal voltage signal (generally 1 kHz to 20 kHz). When the rotor shaft turns, two voltage signals are induced in the stator windings. The amplitude of these outputs is modulated respectively with the sine and cosine of the electrical shaft angle according to relation (3.1).

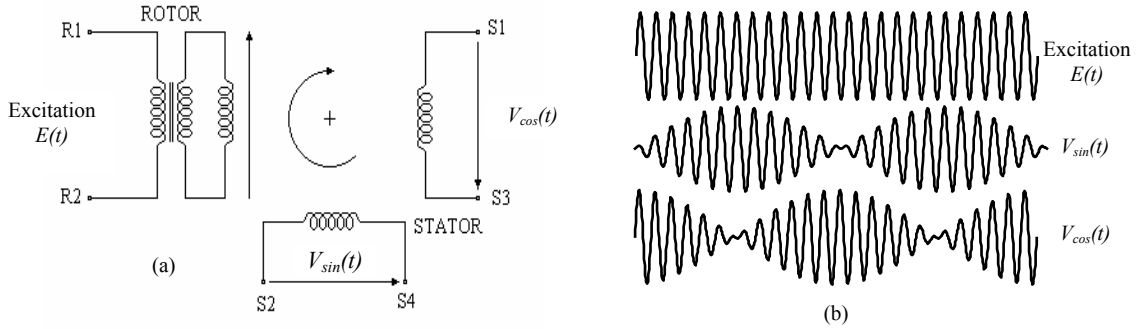


Figure 3.4: (a): Resolver Construction scheme (b): Resolver Input/output waveforms

$$\begin{cases} V_{sin}(t) = m \cdot E(t) \cdot \sin(\theta_r) \\ V_{\cos}(t) = m \cdot E(t) \cdot \cos(\theta_r) \end{cases} \quad (3.1)$$

Where $E(t)$ is the high frequency square or sinusoidal excitation signal, m the transformation ratio and θ_r the electrical rotor position. In case of sinusoidal excitation signal, resolver input/output waveforms are presented in Figure 3.4(b).

To ensure high system integration, the treatment and the extraction of the rotor position and speed from the resolver outputs are ensured by an FPGA-based Resolver Processing Unit (RPU), instead of using an external off-chip Resolver to Digital Converter (RDC), [79].

3.2. RPU principle

The RPU treatment is done in two steps, as shown in Figure 3.5. Firstly, a synchronous demodulation is achieved. The latter consists in demodulating resolver signals so as to extract the sine and cosine of the rotor position. There are various demodulation techniques, all with their advantages and limits in terms of demodulation quality and also in terms of algorithm complexity. In [19], authors have reported and studied three typical methods: PLL-based method, Quadrature method and peak detection method. The last one has been chosen in the case of the developed application for its simplicity. It consists in using the ADC to detect and convert the V_{sin} and V_{cos} peak values. The second step of the RPU treatment is the extraction of the position and speed from these demodulated signals. The so-called Angle Tracking Observer is implemented for this aim.

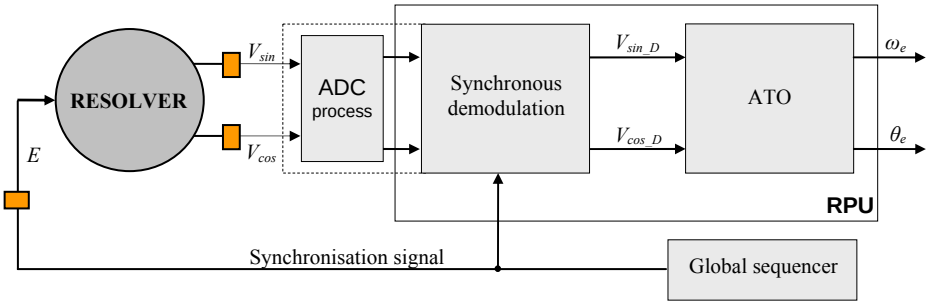


Figure 3.5: RPU principle

3.3. Synchronous demodulation

The implemented synchronous demodulation method, presented in Figure 3.6, consists in sampling the V_{sin} and V_{cos} peaks and delivering the demodulated V_{sin_D} and V_{cos_D} signals. In order to increase the demodulation precision, the sign of the negative peaks is reversed depending on the synchronization signal.

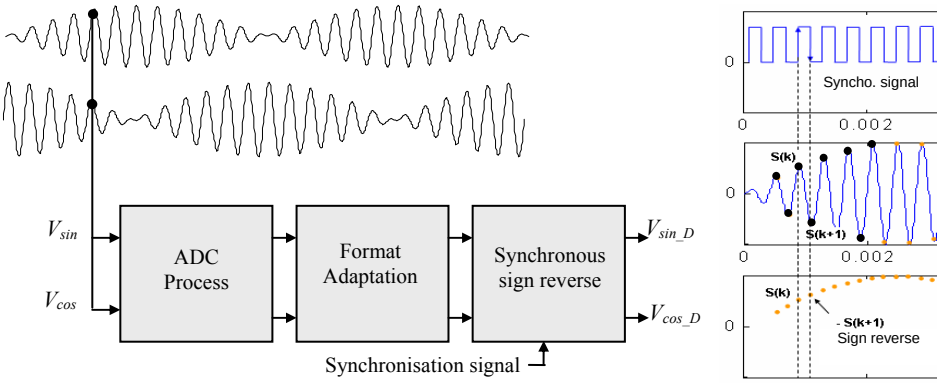


Figure 3.6: Synoptic of synchronous demodulation process

3.4. Angle Tracking Observer (ATO)

This is a closed-loop system which estimates accurately both speed and position, compared to other solutions such as trigonometric extraction methods [19], [79]. Indeed, this observer consists in a second order closed-loop system which compares permanently the actual angular position θ_r to the estimated angular position θ_e. The objective is to minimize the angular position error.

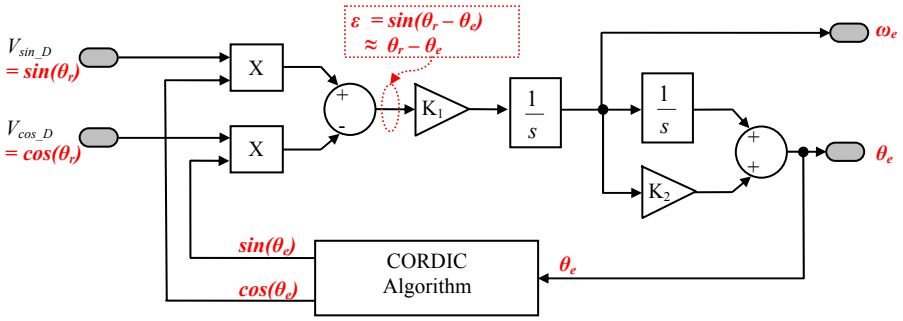


Figure 3.7: ATO closed-loop

Figure 3.7 presents the implemented ATO closed-loop system, for the proposed application. In this closed-loop, the computation of sine and cosine of the estimated position is ensured by a CORDIC algorithm [39]. This last allows performing trigonometric functions, with an iterative algorithm and using only simple operations like shifts and additions.

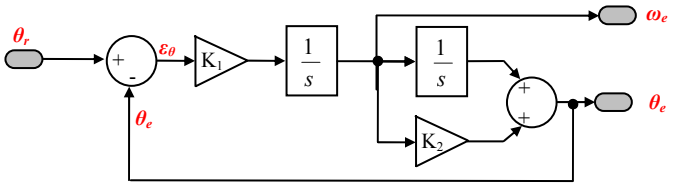


Figure 3.8: Linearized ATO closed-loop

The angular position error between θ_r and θ_e can be expressed as in relation (3.2). For small variations of the error, this relation can be linearized and rewritten according to relation (3.3). Figure 3.8 presents the closed-loop system after the application of the linearization assumption. Based on this simplified closed-loop, the ATO transfer functions are expressed in relations (3.4) and (3.5).

$$\varepsilon_\theta = sin(\theta_r)\cdot cos(\theta_e) - cos(\theta_r)\cdot sin(\theta_e) = sin(\theta_r - \theta_e) \tag{3.2}$$

$$\varepsilon_\theta = sin(\theta_r - \theta_e) \approx \theta_r - \theta_e \tag{3.3}$$

$$H_\theta(s) = \frac{\theta_e(s)}{\theta_r(s)} = \frac{k_1(1 + k_2 s)}{s^2 + k_1 k_2 s + k_1} \tag{3.4}$$

$$H_\omega(s) = \frac{\omega_e(s)}{\theta_r(s)} = \frac{k_1 s}{s^2 + k_1 k_2 s + k_1} \tag{3.5}$$

These are second order transfer functions for which the observation dynamic is set using K_1 and K_2 coefficients. These coefficients are chosen depending on the required estimation speed and on the observation error tolerance. As for the digital realization of the ATO, the adopted discretization assumption is based on Forward Euler approximation. Namely, *1/s* is replaced by $T_s/z\text{-}1$ where T_s is the sampling period which has been set to 50µs in our case. Figure 3.9 shows the block diagram of the discrete-time ATO closed loop. When it comes to the FPGA implementation, the choice of the appropriate fixed-point data format is essential. In the proposed ATO algorithm and with the achieved dynamic setting, the optimum data format that preserves the treatment precision is set to 19Q15 (see chapter 5 section 4.3 for the fixed-point notation). The obtained fixed-point simulation results are highlighted in Figure 3.10. They are obtained in the case of a sinusoidal resolver excitation with a frequency set to 10 kHz.

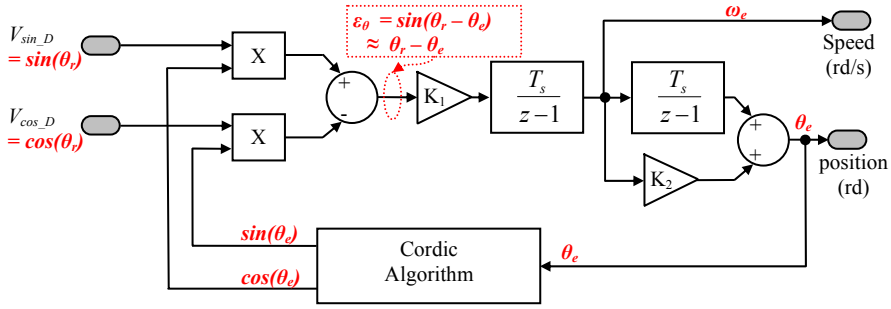


Figure 3.9: Discrete-time ATO closed-loop

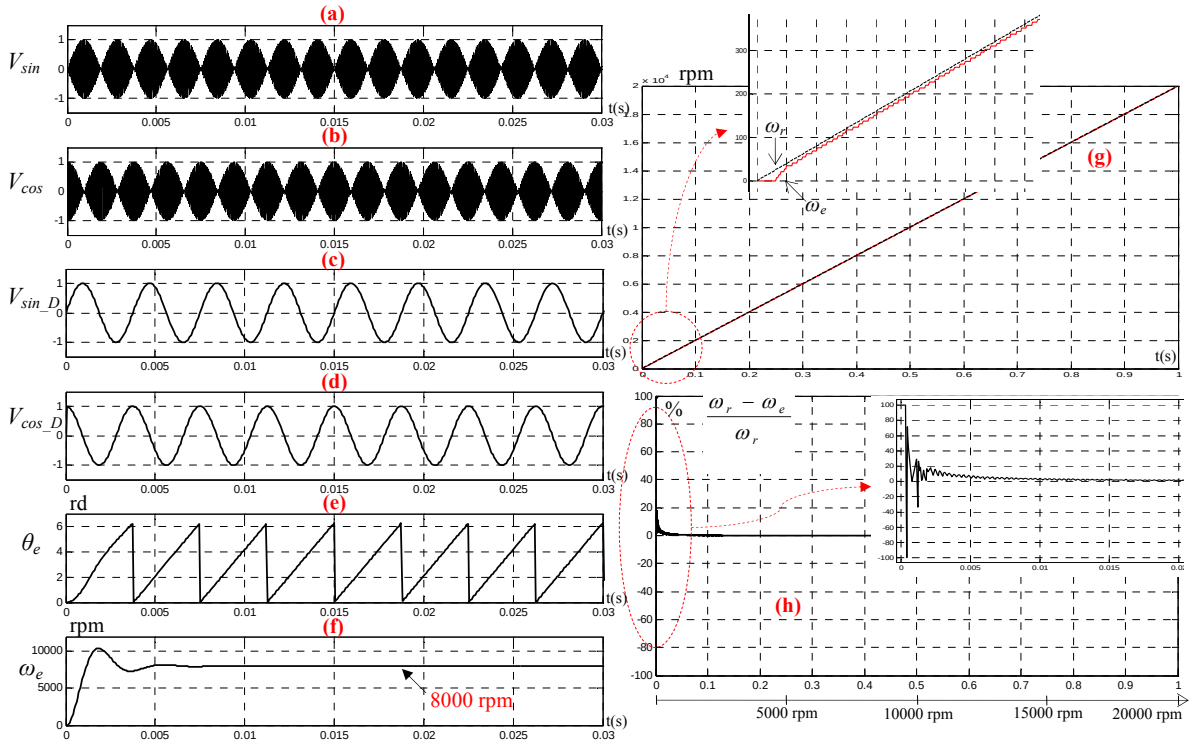


Figure 3.10: Fixed-point simulation results of the RPU –
(a): waveform of V_{sin} **(b)** : waveform of V_{cos} **(c)** : demodulated V_{sin_D} **(d)** : demodulated V_{cos_D}
(e) : estimated rotor position θ_e **(f)** : estimated rotor speed ω_e for 8000 rpm mechanical speed
(g) : estimated speed for an acceleration 0-20000 rpm mechanical speed **(h)** : relative speed estimation error (%)

3.5. Compensation of the ADC Sampling Synchronization Error for resolver signals

During the synchronous demodulation process, the resolver signals V_{sin} and V_{cos} are sampled and converted by the single integrated ADC of the used Fusion FPGA. As a consequence, they aren't sampled at the same time, as illustrated in Figure 3.11. At each sampling period kT_s, the conversion process is launched and the conversion results are respectively ready at $kT_s + t_{conv}$ and $kT_s + 2t_{conv}$, where t_{conv} corresponds to the conversion time.

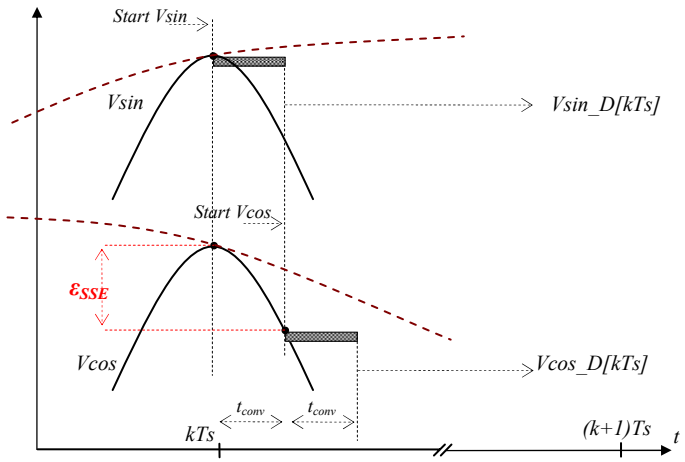


Figure 3.11: Conversion process for resolver signals

Taking into account the amplitude modulated signals, the sinusoidal excitation signal and the ADC SSE, the demodulated signals V_{sin_D} and V_{cos_D} are expressed in relation (3.6), where θ_r, ω_r and ω_{ex} are respectively rotor position, rotor speed and the excitation signal pulsation.

$$\begin{cases} V_{sin_D[kT_s]} = sin(\frac{\pi}{2}) \cdot sin(\theta_{r[kT_s]}) \\ V_{\cos_D[kT_s]} = sin(\frac{\pi}{2} + \omega_{ex} t_{conv}) \cdot cos(\theta_{r[kT_s]} + \omega_r t_{conv}) \end{cases} \quad (3.6)$$

Assuming that ω_{ex} and t_{conv} are constant and known, the compensation procedure of the ADC SSE for resolver signals can be made according to Figure 3.12. This figure shows the z-domain modified ATO closed-loop. α and β variables are expressed in relation (3.7).

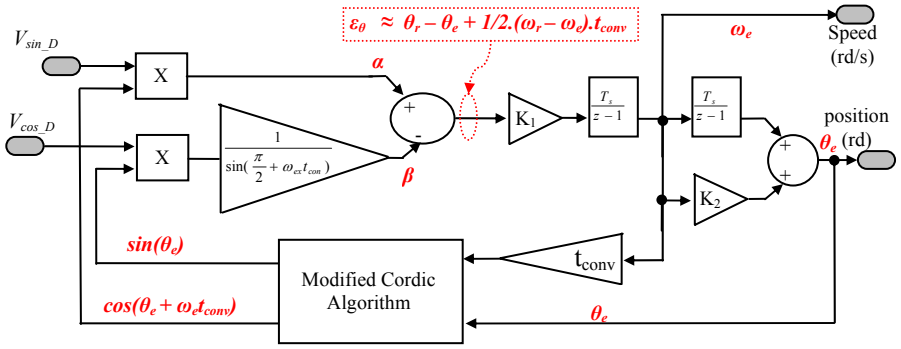


Figure 3.12: Modified ATO closed-loop

$$\begin{cases} \alpha = \frac{1}{2} \cdot [sin(\theta_r + \theta_e + \omega_e t_{conv}) + sin(\theta_r - \theta_e - \omega_e t_{conv})] \\ \beta = \frac{1}{2} \cdot [sin(\theta_e + \theta_r + \omega_r t_{conv}) + sin(\theta_e - \theta_r - \omega_r t_{conv})] \end{cases} \quad (3.7)$$

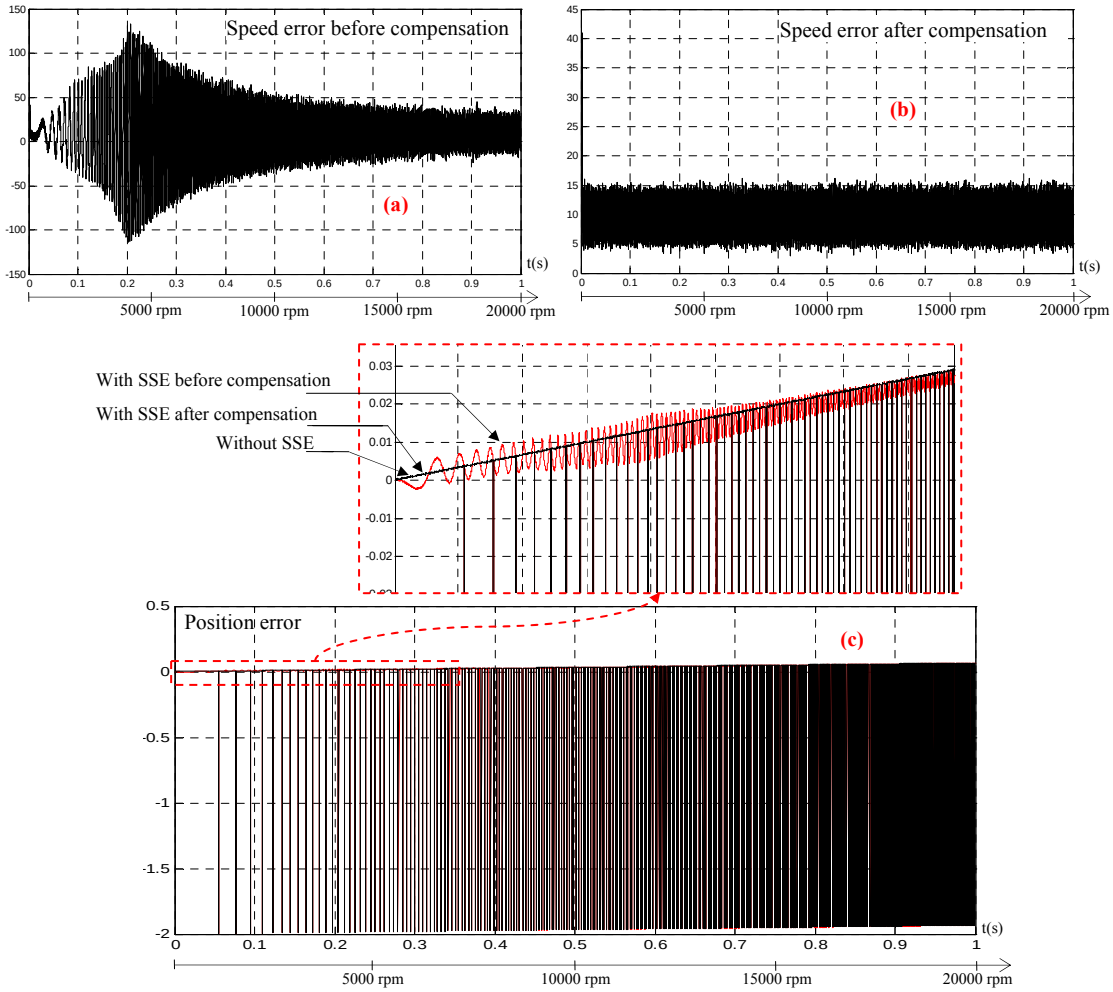


Figure 3.13: RPU Fixed-point simulations before and after SSE compensation
(a,b): speed estimation error (acceleration 0-20000 rpm-mechanical) before (a) and after (b) SSE compensation
(c): position estimation error (acceleration 0-20000 rpm mechanical)

Assuming that the rotor speed is correctly estimated ($\omega_r=\omega_e$), relations (3.8) and (3.9) present the new observation error respectively before and after linearization. The same expression as in (3.3) is finally obtained.

$$\varepsilon_\theta = \alpha - \beta = \frac{1}{2}\cdot[sin(\theta_r - \theta_e - \omega_e t_{conv}) - sin(\theta_e - \theta_r - \omega_r t_{conv})] \tag{3.8}$$

$$\varepsilon_\theta \approx \theta_r - \theta_e + \frac{1}{2}(\omega_r - \omega_e)t_{conv} \approx \theta_r - \theta_e \tag{3.9}$$

Figure 3.13 shows the fixed-point simulation results of the RPU unit before and after ADC SSE compensation. These results prove that the serialization of the analog to digital conversion has an impact on the observation quality.

3.6. Evaluation of the noise rejection

An evaluation of the noise rejection of the developed RPU has also been done. A noise at the PWM frequency (10 kHz) was injected to Resolver signals (V_{sin} and V_{cos}) and the behavior of the obtained position and speed was observed. Figure 3.14 shows the waveform of the relative error between the theoretical speed and the estimated speed after having injected the PWM noise. This fixed-point simulation result has been obtained for an acceleration from 0 to 20000 rpm mechanical

speed. This indicates that the injected noise has a limited impact on the estimation quality either for position or speed.

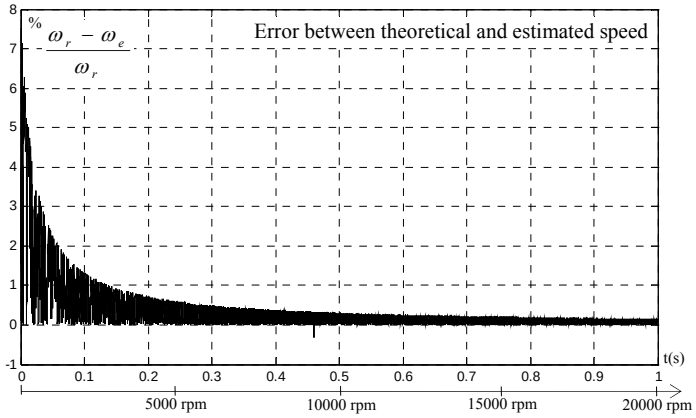


Figure 3.14: Evaluation of the RPU noise rejection

3.7. Experimentation

Figure 3.15 presents the experimental results obtained after implementing the RPU in an AFS600 Fusion FPGA (13824 FPGA tiles, 1 ADC, 40 analog I/Os, 4 Mb Flash memory, 108Kbits RAM memory). Resolver shaft is linked to the used PMSM that turns at 800 rpm mechanical speed. The computation is synchronized by a 50MHz clock signal. The sampling period is synchronized with the resolver excitation and is fixed to 50µs and the frequency of the resolver excitation signal is set to 10 kHz. The used internal tiles present 25% of the FPGA matrix. These results are obtained for a 19-bit fixed point data format. The obtained position and speed waveforms prove a good FPGA-based treatment in terms of quality and accuracy.

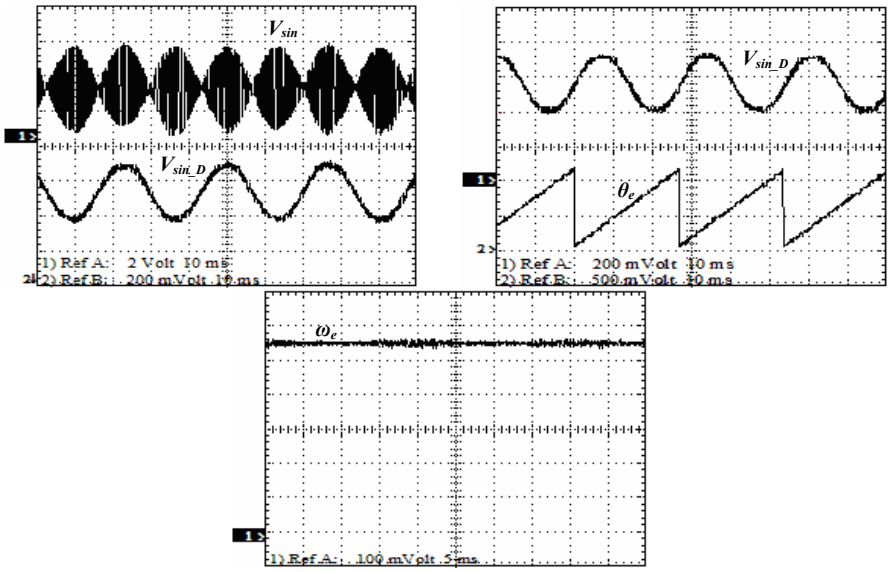


Figure 3.15: RPU - Experimental results

4. FPGA-based controller description

The goal of the proposed control system consists in controlling the stator currents of the PMSM. Among the various control strategies that have been proposed in [62] and [75], it has been chosen to implement an Hysteresis ON/OFF controller. The latter consists of a dq-abc coordinates transformation module and a 3-phase Hysteresis regulator. The dq-abc module generates the 3-phase

current references with regards to the expected dq current references and the rotor position. These references are then compared to the measured currents by hysteresis regulators in order to generate the appropriate switching signals. Figure 3.16 shows the corresponding FPGA architecture. The ADC management module is composed of the previously discussed Global ADC module and AD interfaces. The latter adapt the conversion results to the appropriate data format, 19-bit fixed point format (19Q15) for RPU treatment and 13-bit fixed point format (13Q12) for current controller.

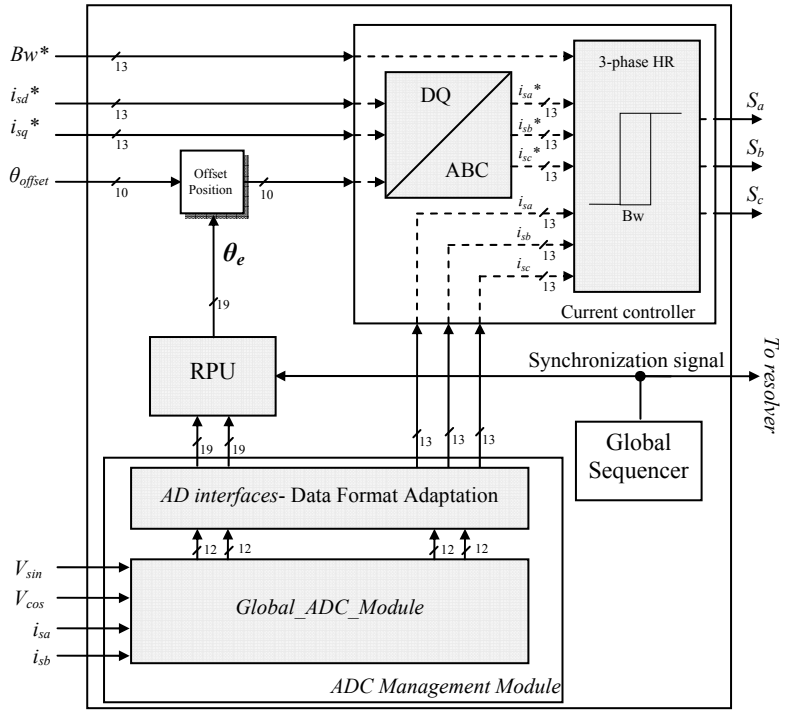


Figure 3.16: FPGA-based controller architecture

4.1. Controller timing diagram

Figure 3.17 highlights the control system timing diagram. At the beginning of each sampling period T_s (set to 50µs), the Resolver position shift (t_{phi}) is taken into account. This shift (measured experimentally) corresponds to the delay between the synchronization signal edge and the peak of resolver signals. This delay is mainly due to the resolver interface electronic circuits and the resolver windings. After this shift, the ADC management module is activated. It converts, sequentially, the resolver and current sensor analog outputs. At the end of each conversion cycle, the *xxx_RDY* flag corresponding to each input is activated. Thus, as soon as $V_{cos_D}_RDY$ is activated (Resolver signals are both converted), the RPU starts. Simultaneously, the conversion of the other analog signals (PMSM stator currents) is realized. Finally, the current controller is then activated after the Resolver processing and at the end of the conversion process.

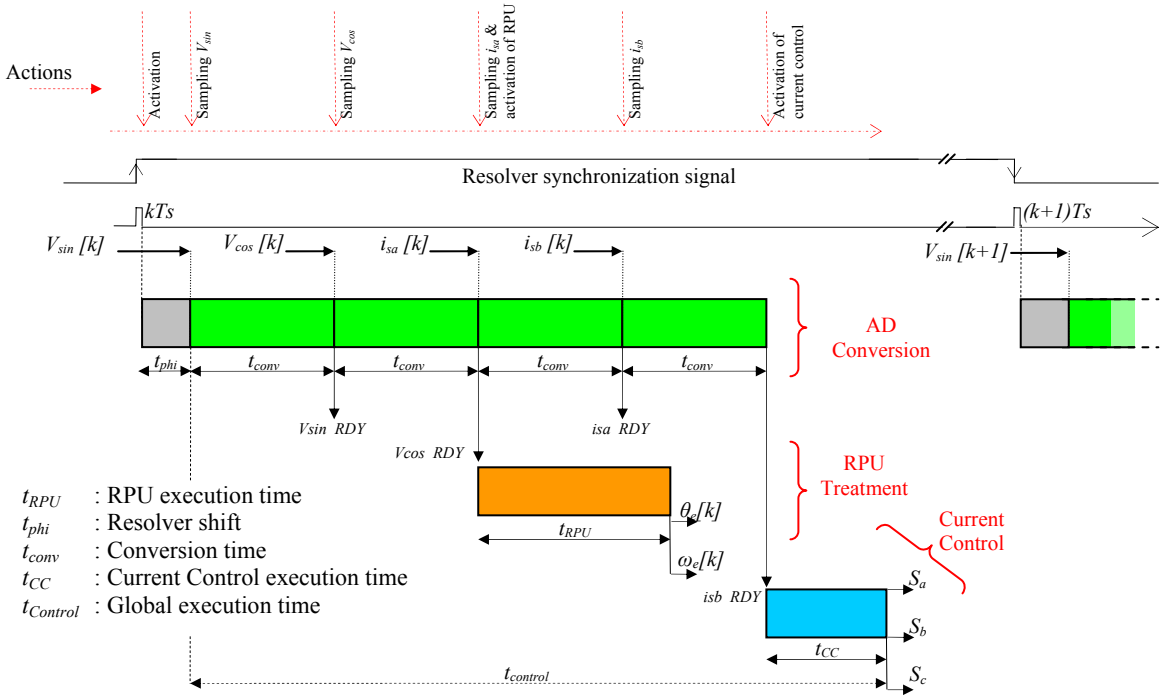


Figure 3.17: Control system timing diagram

4.2. Compensation of ADC Sampling Synchronization Error for PMSM currents

Figure 3.18 presents the conversion process of the measured currents i_{sa} and i_{sb} which is launched at each sampling period (kT_s) after converting the resolver signals V_{sin} and V_{cos}. This conversion is made sequentially. As consequence sampling synchronization errors are introduced (ε_{SSEa} for i_{sa} and ε_{SSEb} for i_{sb}). Assuming that the stator currents are sinusoidal, these sampling errors correspond to position errors. The conversion results of i_{sa} and i_{sb} at each sampling period are then expressed in (3.10). This relation includes the third current i_{sc} obtained by considering that the 3-phase system is balanced.

$$\begin{cases} i_{sa[kTs]} + \varepsilon_{SSEa[kTs]} = \hat{I}_s \, sin(\theta_{e[kTs]} + 2\omega_e.t_{conv}) \\ i_{sb[kTs]} + \varepsilon_{SSEb[kTs]} = \hat{I}_s \, sin(\theta_{e[kTs]} - \frac{2\pi}{3} + 3\omega_e.t_{conv}) \\ i_{sc[kTs]} = -(i_{sa[kTs]} + i_{sb[kTs]} + \varepsilon_{SSEa[kTs]} + \varepsilon_{SSEb[kTs]}) \end{cases} \quad (3.10)$$

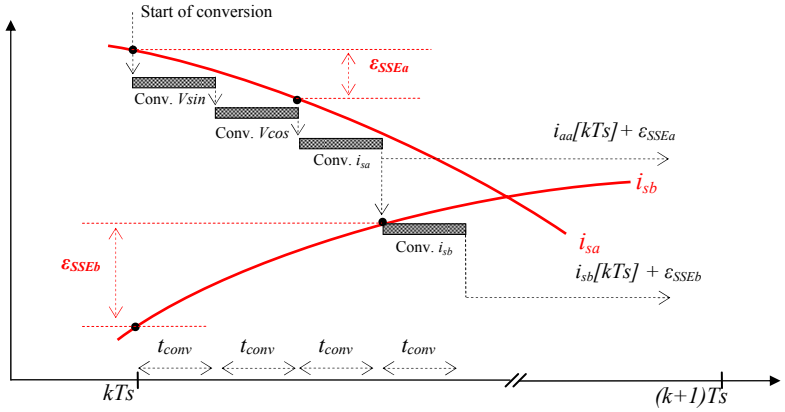


Figure 3.18: Conversion process for stator currents

These measured currents are compared by the hysteresis regulator to their corresponding reference values i_{sa}^*, i_{sb}^* and i_{sc}^*. The current errors are then written as,

$$\begin{cases} \varepsilon_{i_{sa[kTs]}} = i_{sa}^{*}[kTs] - (i_{sa[kTs]} + \varepsilon_{SSEa[kTs]}) \\ \varepsilon_{i_{sb[kTs]}} = i_{sb}^{*}[kTs] - (i_{sb[kTs]} + \varepsilon_{SSEb[kTs]}) \\ \varepsilon_{i_{sc[kTs]}} = i_{sc}^{*}[kTs] + i_{sa[kTs]} + i_{sb[kTs]} + \varepsilon_{SSEa[kTs]} + \varepsilon_{SSEb[kTs]} \end{cases} \quad (3.11)$$

The compensation of the impact of the sequential conversion is made by adding the sampling errors ε_{SSEa} and ε_{SSEb} respectively to the reference values i_{sa}^* and i_{sb}^* in order to have the same current errors as in an ideal case (3.12).

$$\begin{cases} \varepsilon_{i_{sa[kTs]}} = (i_{sa}^{*}[kTs] + \varepsilon_{SSEa[kTs]}) - (i_{sa[kTs]} + \varepsilon_{SSEa[kTs]}) = i_{sa}^{*}[kTs] - i_{sa[kTs]} \\ \varepsilon_{i_{sb[kTs]}} = (i_{sb}^{*}[kTs] + \varepsilon_{SSEb[kTs]}) - (i_{sb[kTs]} + \varepsilon_{SSEb[kTs]}) = i_{sb}^{*}[kTs] - i_{sb[kTs]} \\ \varepsilon_{i_{sc[kTs]}} = i_{sc}^{*}[kTs] - i_{sc[kTs]} \end{cases} \quad (3.12)$$

In case of sinusoidal signals and assuming that the conversion time t_{conv} is constant and known, the compensation is made through the position θ_e and the electrical speed ω_e, both estimated by the RPU. Figure 3.19 summarizes the proposed compensation procedure. The ADC SSE is introduced through the modified Park transformation. The 3-phase current references are then expressed according to the following relation,

$$\begin{cases} i_{sa}^{*}[kTs] + \varepsilon_{SSEa[kTs]} = \hat{I}_s^{*} \sin(\theta_{e[kTs]} + 2\omega_e.t_{conv}) \\ i_{sb}^{*}[kTs] + \varepsilon_{SSEb[kTs]} = \hat{I}_s^{*} \sin(\theta_{e[kTs]} - \frac{2\pi}{3} + 3\omega_e.t_{conv}) \\ i_{ss}^{*}[kTs] - \varepsilon_{SSEa[kTs]} - \varepsilon_{SSEb[kTs]} = -(i_{sa}^{*}[kTs] + i_{sb}^{*}[kTs] + \varepsilon_{SSEa[kTs]} + \varepsilon_{SSEb[kTs]}) \end{cases} \quad (3.13)$$

Figure 3.20 shows the evolution of the ADC SSE when increasing the current frequency and the conversion time. This result proves that the serialization of the analog to digital conversion has an impact on the control quality. In this application the electrical speed is low, so this error can be neglected. However, it cannot be so when increasing the speed, the ADC conversion time and the number of analog inputs.

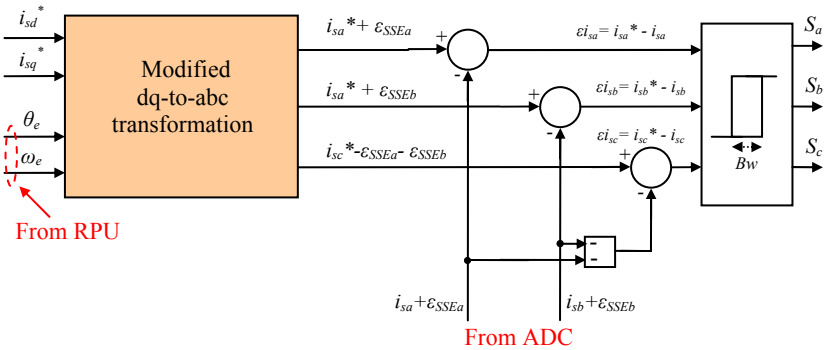


Figure 3.19: Compensation of the ADC SSE for stator currents

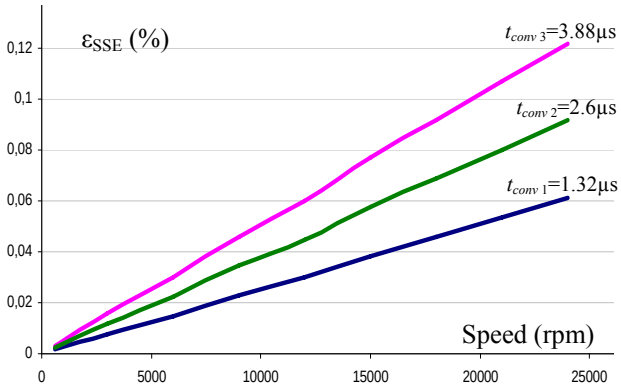


Figure 3.20: SSE = f (speed, t_{conv})

4.3. Experimental results

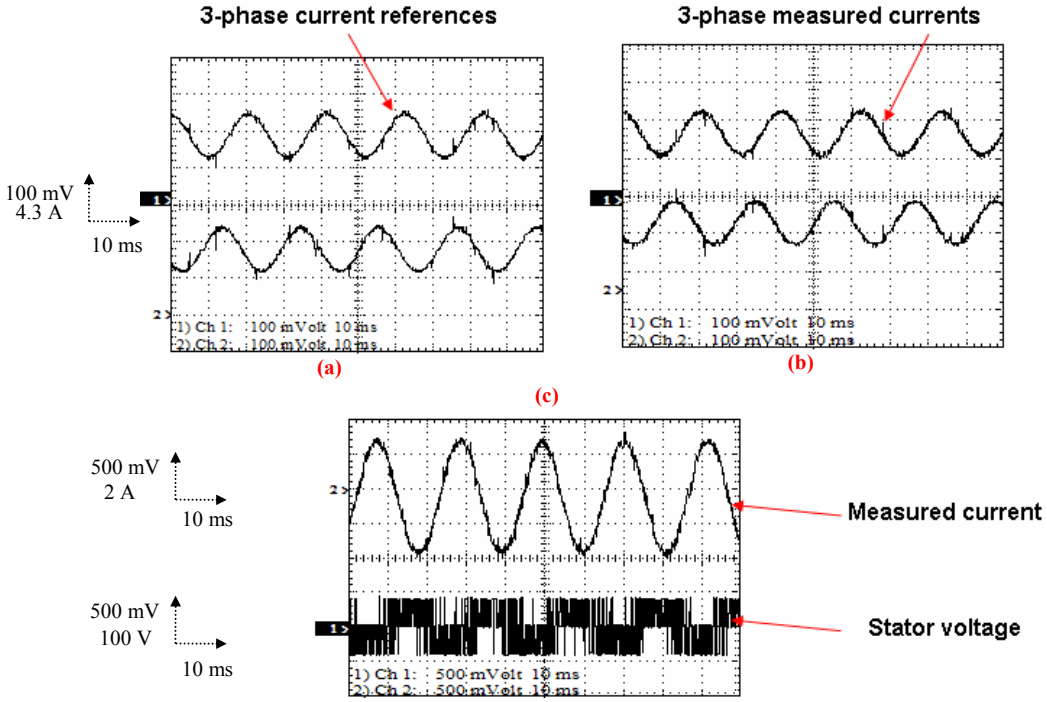


Figure 3.21: Current controller experimental results – Obtained for i_{sd}*=0A; i_{sq}*=3A; Bw*=0A

The experimental set up presented in the first section (Figure 3.1) is composed of a 0.8kW PMSM associated with a Resolver sensor, current sensors and a controlled load (powder brake). The current sensors deliver voltage signals corresponding to the measured currents (2.5V/10A). The sampling period is synchronized with the resolver excitation and is fixed to 50µs. Figure 3.21 presents the stator current and voltage instantaneous waveforms.

4.4. Time/Area performances

Table (3.1) presents the time/area resources of the implemented motor controller. This full FPGA-based solution was based on an AFS600 Fusion FPGA (13824 FPGA VersaTiles, 1 ADC, 40 analog I/Os, 4 Mb Flash memory, 108Kbits RAM memory) [10]. The consumed resources were obtained for a 13-bit fixed-point format for the current controller and 19-bit fixed-point format for the RPU treatment. The obtained area performances show that the global implemented architecture takes 47% of the Fusion FPGA matrix. This is mainly due to the greediest operators like multipliers. Indeed, these multipliers are synthesized in the FPGA matrix because they are not hardwired in the chip. As

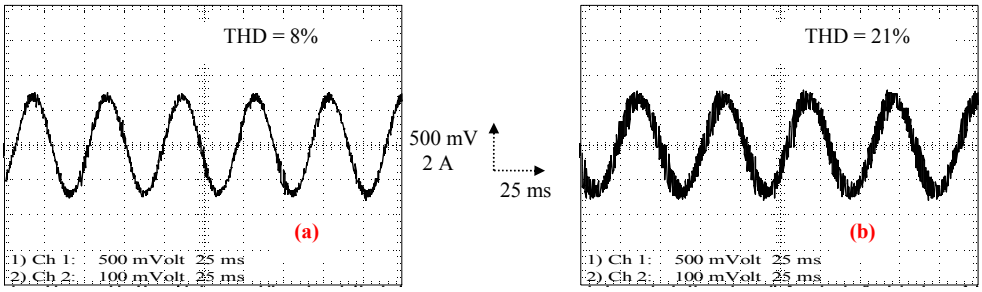
for the time performances, the global execution time (6.5µs) remains short, but it depends mainly on the sequential analog to digital conversion time.

Table 3.1: Time/Area resources

Time/area resources Modules	*% of hardware resources (over 13824 cells)*	*Execution time CLK = 50MHz*
ADC Management module 1 Flash memory block + 4 analog quads + 12-bit ADC	1.48 % (205 cells)	1 internal ADC t_{conv}=1.36µs
RPU Synchronous demodulation Angle Tracking Observer Cordic treatment	25.16 % (3478 cells)	t_{RPU}= 1.56 µs
Hysteresis current control DQ-abc transformation 3-phase hysteresis regulator	16 % (1242 cells)	t_{CC} = 1.06 µs
Global control architecture Total	**47.01 % (6499 cells)**	$t_{control} = 4 * t_{conv} + t_{CC}$ **= 6.5 µs**

4.5. Influence of the execution time on the control quality

In order to illustrate the influence of execution time on the control quality, the obtained experimental results using the hardware fully FPGA controller, are compared to those obtained with a slower solution like a software controller. In the first case, the controller processes and generates the switching signals in 6.5µs delay (including AD conversion time). In the second case, in order to reproduce a typical software-based controller, the execution time was deliberately delayed and the switching signals are generated one sampling period (50µs) later. Figure 3.22 shows the influence of the execution time on the current waveform in case of hysteresis-based control strategy. In this figure, it can be seen that current ripples are significantly larger when execution time is equal to a sampling period (embedded software controller case, Figure 3.22(b)) compared to the case of a fully hardware controller (Figure 3.22(a)).



*isd** = 0A ; *isq** = 3A ; *Bw** = 0A

Figure 3.22: Comparison between a 6.5µs control delay (a) versus a 50µs control delay (b)

5. Conclusion

This chapter has dealt with the development of a fully integrated FPGA-based controller for a PMSM. This PMSM is associated with a resolver position sensor. For a high integration purpose, it has been decided to implement the controller within an Actel Fusion FPGA. The latter is characterized by mixed-signal elements that combine analog elements (e.g. ADC) in addition to the digital features ensured by the FPGA matrix.

The developed controller includes a Resolver Processing Unit (RPU) which calculates rotor position and speed and a current controller which processes the switching signals for the VSI. The conversion process is ensured by the integrated ADC. These modules have been described and propped up with fixed-point simulation and experimental results. Also the Sampling Synchronization

Error (SSE) which is introduced by the ADC structure has been discussed and compensation methods have been presented.

Now, as the utility of using FPGAs to suit high integration demands is discussed, the next step is how they bring their value-added for more complex controllers. Thus, the next chapters deal with the development of an FPGA-based sensorless AC drive application using an Extended Kalman Filter (EKF). Besides the importance of sensorless controllers especially for avionic applications, the performances and the way-to-develop the corresponding FPGA solution will be covered.

Chapter 4

FPGA-based sensorless control for synchronous AC drive - Preliminary system specification

1. Introduction

In this chapter, author presents the preliminary development step of an FPGA-based sensorless controller for a synchronous AC drive. According to the design methodology presented in chapter 2 (Figure 2.18), this step consists in making a preliminary system specification regarding the whole sensorless control application.

Thus, at the beginning, a hardware specification is made. The power stage, the electrical sensors, the analog/digital interfaces and the digital control unit are specified. For the power stage, this consists in choosing, depending on the load conditions, which AC motor is to be controlled and which supply conditions. The specification of the electrical sensors that are used to measure voltages and currents is also important at this step since the introduced gains are taken into account during the algorithm normalization (chapter 5). The specification of the analog/digital interfaces consists in choosing the Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC) boards. Here again the specification of the ADC is important since the introduced gain and the conversion time are taken into account respectively during the normalization and during the analysis of the controller timing performances. Finally, the appropriate FPGA-based digital control unit is defined.

Beyond this physical specification, the algorithm benchmarking is made. This consists in specifying the appropriate control strategy and the appropriate sensorless method for the estimation of the rotor position and speed. In the case of the implemented Extended Kalman Filter (EKF), the selection of the appropriate system model that allies model precision and complexity is to be achieved.

This chapter is organized as follows. The hardware specification is discussed in the following part. The third part deals with the chosen speed and stator current control strategies. Then, part 4 presents some widespread sensorless methods for the estimation of the rotor position and speed. This

investigation leads to the choice of the sensorless method for the developed application. Thus the principle and the algorithm of the chosen EKF are presented in part 5. Finally, the development and the selection of the system model for the EKF treatment are dealt with in part 6.

2. Sensorless control system - Hardware specification

The structure of the proposed sensorless control system is overviewed in Figure 4.1. It highlights mainly the power stage structure, the FPGA-based digital control unit, the electrical sensors, the ADC board, the DAC board and the Host-PC interface.

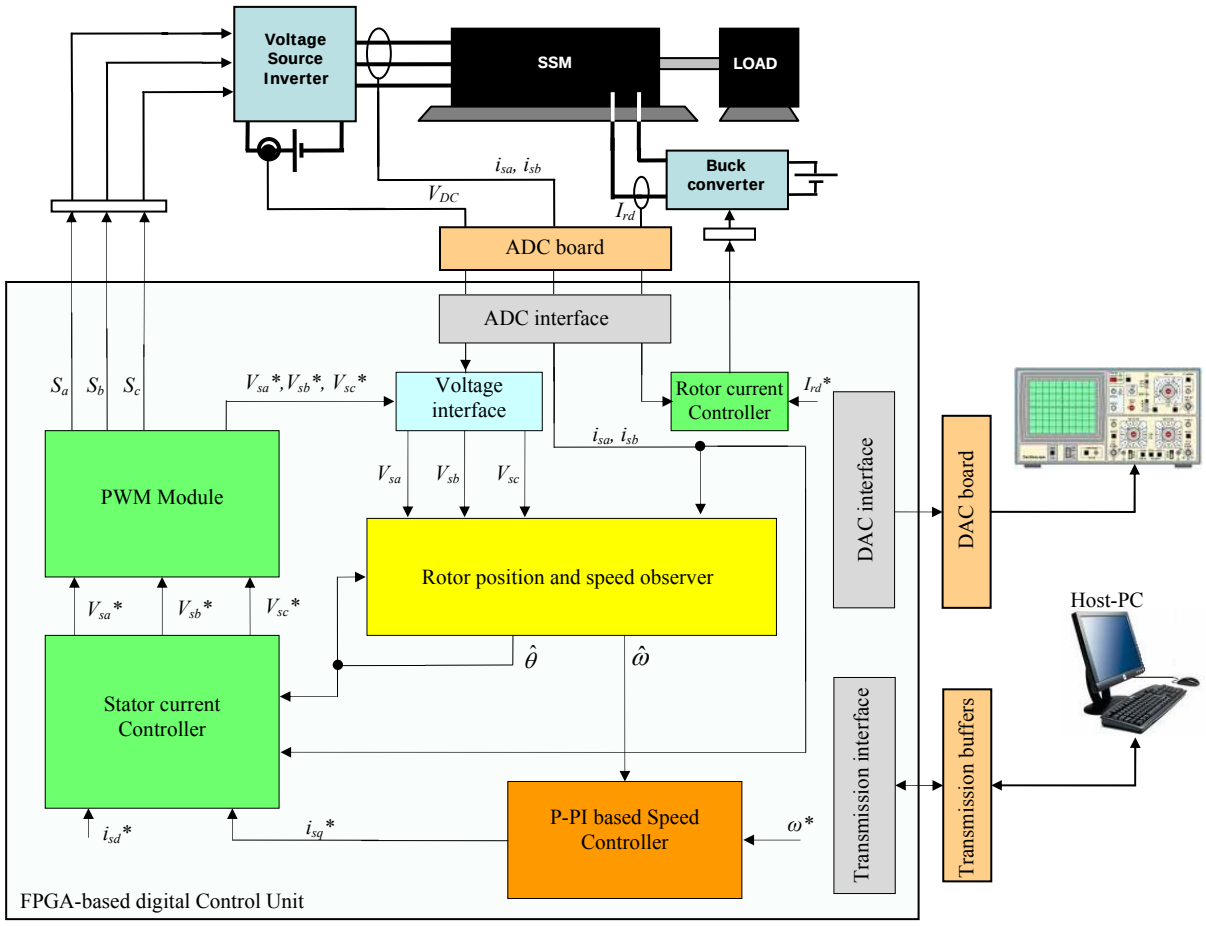


Figure 4.1: Synoptic of the developed control system

2.1. Power stage

The power stage is composed of a Salient Synchronous Machine (SSM) associated with a controlled load (i.e. a powder brake). The characteristics of this power stage are presented in chapter 7. The modeling and the parameter identification of the SSM are presented in [Appendix B].

This machine has a wounded rotor which needs an additional supply source. This last consists of a buck converter which, associated with a hysteresis-based current controller, aims to maintain the rotor current to the desired reference. The stator is fed by a 3-phase 2-level Voltage Source Inverter (VSI) based on Insulated Gate Bipolar Transistor (IGBT) modules. In [Appendix-A] and [21], a characterization of this VSI is made. A measurement of the dead time between switching signals, the turn-on/off delays and the voltage drop introduced by the power switches is done. The influence on the control quality and the way-to-compensate these nonlinearities are presented.

2.2. Electrical sensors, ADC and DAC boards

The used electrical sensors deliver voltage signals corresponding to the measured stator currents and the measured DC link voltage (2.5V/10A, 1V/100V). The implemented measurement system is based on the ARCTU3 board which is deeply described in [62].

For the developed platform, the available ADC board is composed of four AD9221 ADCs. These are successive approximation register components that provide conversion results as 12-bit signals. The corresponding ADC interface adapts the converted signals to the chosen data format. The effective conversion time (ADC conversion time + the post signal adaptation) is equal to 2.4 µs.

The implemented Digital to Analog Converter (DAC) board consists of a set of 10-bit AD9760 devices that are used to convert the internal signals for display purposes.

2.3. FPGA-based digital control unit

In the proposed application, the digital control unit is based on a Xilinx FPGA board. As it will be discussed in chapter 6, the objective is the possibility to implement the developed design within a low cost FPGA device. Thus with the Xilinx family, this concerns the low cost Spartan-3, Spartan-3E and Spartan-6 series.

2.4. Host-PC interface

This interface ensures a real-time transfer of data between the digital control unit and the user Host-PC. In the proposed work and in the case of Xilinx FPGA device, the ChipScope tool has been used and the transmission is based on USB-JTAG protocol. This tool ensures the setting of the sensorless controller in the one hand and the probing of the internal signals in the other hand. This latter point will be deeply investigated during the Hardware-In-the-Loop validation (chapter 7).

3. Stator current controller and speed controller

3.1. Stator current controller

Depending on the aimed synchronous AC drive application, various control strategies are available. All these strategies are composed of an inner current control closed loop. The latter consists in maintaining the measured stator currents of the motor as close as possible to the desired current references. To this purpose, several current control techniques are available [62], [75] including for example ON/OFF, Predictive, PI-based and sliding mode control techniques.

The evaluation of each of these control techniques and the selection of the appropriate one is based on which one is able to satisfy the expected control performances (for example high precision, robustness toward parameter variations, large bandwidth, low switching power losses and low THD in current waveforms). To this aim, these current control techniques have been studied and analyzed in [62] and an in-depth comparison in terms of control performances has been made. Also [62] provides a deep understanding and a relevant argumentation of how the use of FPGAs as digital solutions is important for the implementation of such current control techniques.

In our case, the implemented FPGA-based stator current controller is based on an Anti-windup PI regulator [81], [82]. Figure 4.2 summarizes the corresponding principle. The d-q current regulators calculate the d-q voltage references according to the measured and reference currents. The axes decoupling has been included so as to make independent the regulation in the d and q axes. After a coordinate transformation, the 3-phase voltage references are processed. Note that this Park transformation is processed after a Clark transformation with the amplitude-conservation. The used Pulse Width Modulation (PWM) module generates the corresponding switching signals for the VSI. The tuning of the d-q current regulators is discussed in [Appendix C].

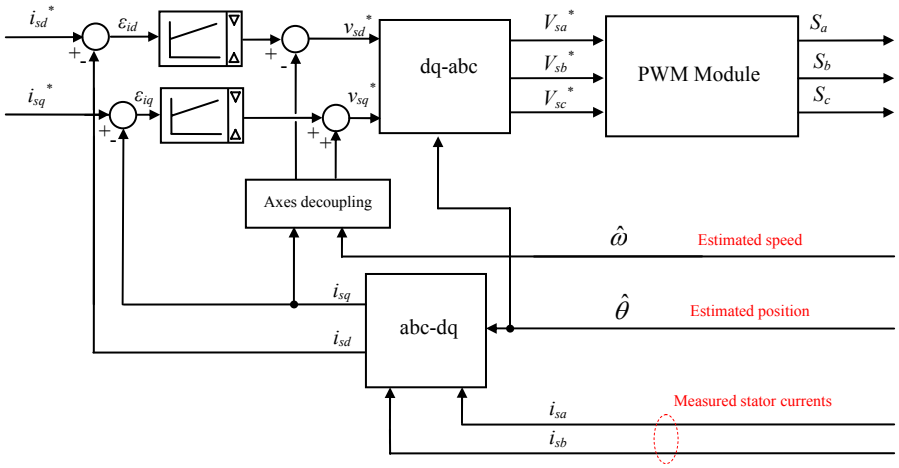


Figure 4.2: Structure of the stator current controller

3.2. PWM specifications

The implemented synchronous machine is fed by a 3-phase, 2-level VSI. The switching signals of the latter are generated by the PWM module. There are various PWM techniques; each of them is specified depending on the implementation approach (analog or digital), on the operating conditions (e.g. rated power and switching frequency) and on the expected performances (e.g. linearity range, switching losses and THD). According to [62] and [83], where various PWM techniques are analyzed, two main categories are suggested. The first one is the carrier based PWM which includes the basic sinusoidal PWM technique (CB-SPWM) and the PWM with zero sequence signal (CB-ZSS-PWM). The second category is composed of the well-known Space Vector PWM (SVPWM).

In the proposed FPGA-based controller, it has been chosen to implement the CB-ZSS-PWM technique. Figure 4.3 presents the corresponding principle scheme. The ZSS voltage (between *n* and *o* neutral points) is processed from the 3-phase sinusoidal voltage references (V_{sa}^*, V_{sb}^*, V_{sc}^*) and added at the same time so as to generate the corresponding voltages (V_{ao}^*, V_{bo}^*, V_{co}^*). The choice of such PWM technique is motivated by its low algorithm complexity (easy FPGA implementation) and also because it ensures, as an SVPWM, a full use of the VSI 3-phase supply voltage [62], [83].

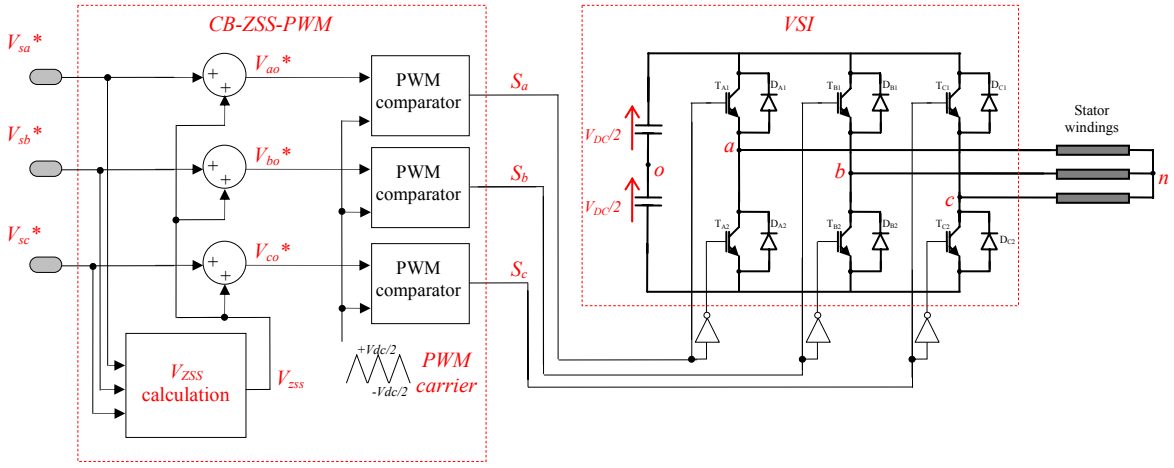


Figure 4.3: Principle of the CB-ZSS-PWM

3.3. Compensation of VSI nonlinearities

One of the key-issues that influence the motor control quality is the performance of the used VSI. Indeed, the latter introduces nonlinearities on its output voltage, mainly due to the dead time of the switching signals (added to avoid short-circuit), to the turn-on/off delays of the used IGBTs and to

the voltage drop across these power switches. This output voltage error has a non-negligible influence on the motor current and as consequence on the developed torque. Various compensation methods have been proposed to improve the VSI output voltage waveform such as pulse-based methods [84], [85] and voltage average value based methods [85], [86]. In our case, the proposed method consists in compensating the VSI non-linearity through the reference voltage average value applied to the CB-ZSS-PWM. To this aim, an in-depth study of this compensation assumption is made in [Appendix A] and [21].

3.4. Speed controller

The developed speed controller is made up using a Proportional-Proportional Integral (P-PI) regulator. As deeply studied in [81], this controller is characterized by double speed feedback loops, an internal loop and an external loop. As depicted in Figure 4.4, the first one consists of a Proportional regulator which imposes the poles of the controlled system. The external speed control loop is performed via a PI regulator in order to ensure a zero steady-state error and set the response dynamic. The tuning of this P-PI regulator is discussed in [Appendix C].

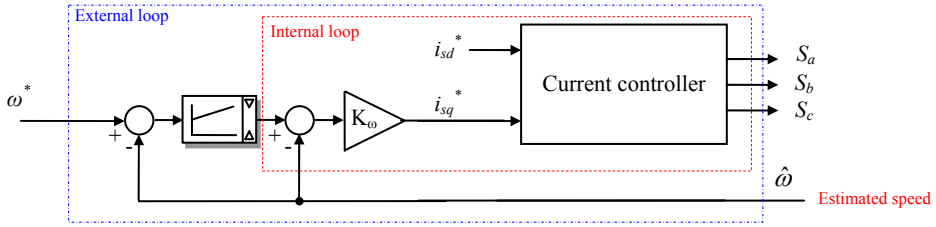


Figure 4.4: Structure of the speed controller

3.5. Voltage interface

In order to reduce the number of voltage sensors, the voltage interface has been developed. This interface generates the 3-phase stator voltages from the measured DC voltage and 3-phase stator reference voltages according to relation (4.1). These voltages are used for the estimation of the rotor position and speed that will be discussed thereafter.

$$V_{si} = V_{DC} \cdot V_{si}^* \quad ; \quad i = a,b,c \tag{4.1}$$

4. Estimation of the rotor position and speed - sensorless methods

In the literature, many research investigations in the field of sensorless controllers dedicated to AC drives have been achieved. Several sensorless methods for the estimation of the rotor speed and position have been implemented. Each of them has its own advantages and limits according to the operating conditions. In the following, some examples of the frequently encountered sensorless methods are presented. To make a clear classification, they can be divided in two categories; the model based category and the signal injection based category (according to [88]). Figure 4.5 highlights this classification.

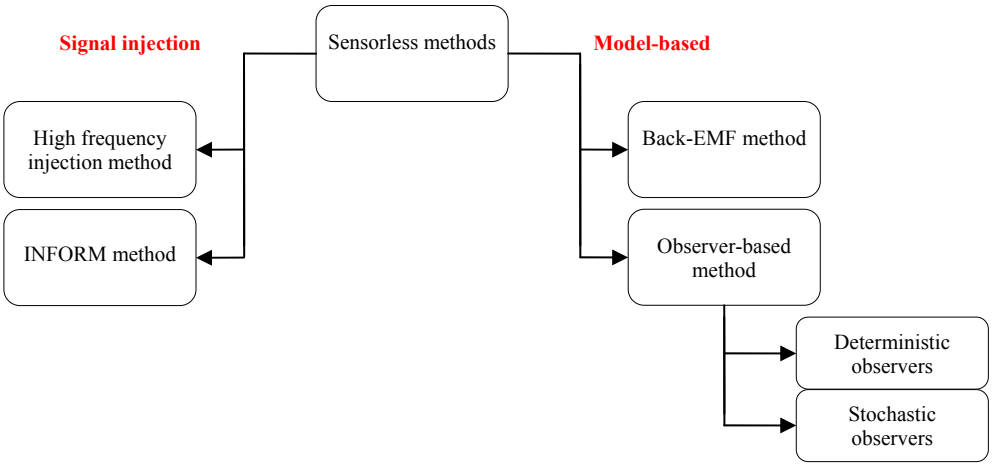


Figure 4.5: Examples of the frequently encountered sensorless methods

4.1. Sensorless methods based on signal injection

This category exploits the machine anisotropy. When implementing an SSM, the estimation approach exploits typically the saliency and relies on the dependence between the rotor position and the inductances. The principle is based on signal injection through voltages and on the measurement of currents. The main advantage of these sensorless methods is their ability to estimate rotor position at standstill and operate at very low speed.

4.1.1. High frequency injection method

This sensorless method consists in injecting a high frequency voltage signal in the voltage reference generated by the controller (e.g. d-q voltages in the case of a PI-based controller). Thus the extraction of the rotor position and speed is achieved by measuring the stator currents and making the appropriate signal filtering so as to extract the harmonic that contains the position information. Examples of implementations can be found in [88] and [89].

4.1.2. INFORM method

Another sensorless approach is the INFORM (Indirect Flux Detection by Online Reactance Measurement) method. This method has been introduced by [52] and consists in applying two opponent voltage phasors in order to track the saturations and geometric saliency of the motor. In other words, this consists in applying two consecutive voltage sequences (two switching voltage states) in opposite direction. Two consecutive current variations are then measured. When calculating the difference between the applied voltages, it is possible to eliminate the resistive voltage and the back-EMF. The obtained equation is then based only on the inductance which is extracted and used to calculate the rotor position.

4.2. Sensorless methods based on the motor model

These sensorless methods are based on the motor model. They are adapted to high speed operating conditions. However, their main limit is their estimation failure at standstill and very low speed. This is because the system becomes unobservable and the amplitude of voltages is very low (zero at standstill).

4.2.1. Back-EMF method

This approach consists in extracting the back-EMF from the implemented machine electrical equations (e.g. [90]). The implemented motor model is usually based on the stationary (α,β) frame. The extraction of the rotor position can be made using *Arctan* function and a post treatment is to be done to extract the speed. Another extraction method can be based on specific Phase Locked Loop (PLL) which has the same treatment as the ATO (studied in chapter 3).

4.2.2. Observer-based method

In this case, the extraction of the rotor position and speed is done with the use of observers. These lasts lean on the state space model of the implemented AC motor and the main goal is to minimize the observation error between the measured and the estimated quantities. There are two observation approaches: deterministic approach and stochastic approach.

The first one consists in using deterministic observers such as Luenberger observer or Extended Luenberger observer (for non-linear systems) [91]. Along the same line, other sensorless methods using Sliding Mode Observers (SMO) have been proposed and a typical example is given by [90]. These deterministic observers lean on the system model without considering measurement noises and modeling errors.

The second approach is based on stochastic observers such as Kalman filter for linear systems and the Extended Kalman filter (EKF) for non linear systems [93]-[106]. These observers consist of a set of mathematical equations that implement prediction/innovation optimal estimation process. The goal is to minimize the covariance of the error between the real state space vector and the estimated one by considering measurement noises and modeling errors.

4.3. Choice of the sensorless method

As discussed during the general introduction (chapter 1), the developed sensorless control system is going to be adapted to an industrial aircraft research program. The main objective of this program is to develop an FPGA-based sensorless controller for a Brushless Synchronous Starter Generator (BSSG) [92], [38]. Two sensorless methods will be implemented, one for low and high speed and one for very low speed and standstill.

In the proposed work, the chosen FPGA-based sensorless method for low and high speed operating conditions is the EKF-based method. This choice is based on two main motivations.

The first one is related to its performances. Indeed, the EKF is well-known for its inherent robustness towards random noisy environment (typically the case of avionic systems). This has been proved in wide range of sensorless applications. Such observer presents, anyhow, a major disadvantage which is the intensive mathematical operations compared to other sensorless methods. As it will be seen in the next part, this is due to intensive matrix operations such as multiplications and divisions that demand high computational resources. In the proposed thesis work, this drawback has been transformed to a challenge. This is in fact the second motivation of having chosen the EKF because it is the best candidate to prove that FPGA solutions are suited to such complex algorithms.

As for the case of very low speed and standstill, the high frequency injection method is implemented. This activity is not covered here and is the main subject of the associate thesis work [92].

5. Extended Kalman Filter basics

In the literature, a wide range of papers have discussed the basics the EKF theory. In addition to the conventional structure, many other structures have been proposed (for example, Adaptative EKF, Neural EKF, Two-stage EKF, Reduced order EKF, Unscented Kalman filter …). In this work, the conventional structure of the discrete-time EKF algorithm has been implemented. A brief recall of the corresponding principle is made.

To start with, relation (4.2) presents the discrete-time stochastic state space model of the observed system.

$$\begin{aligned} x_k &= f_d(x_{k-1}, u_{k-1}) + w_{k-1} \\ y_k &= h_d(x_k) + v_k \end{aligned} \tag{4.2}$$

Where x is the state space vector, f_d is the discrete-time state space matrix, u is the input matrix, y is the output (measurement) vector and h_d is the discrete-time output matrix. The model and measurement disturbances are statistically described by the zero-mean Gaussian noises w and v respectively characterized by covariance matrices Q and R.

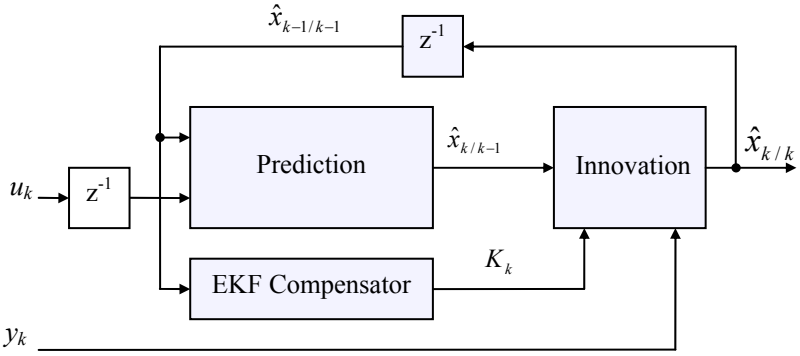


Figure 4.6: Synoptic of the EKF algorithm

Figure 4.6 summarizes the EKF treatment. The latter is done in two main steps, prediction step and innovation step. The first one consists in predicting the state vector $\hat{x}_{k/k-1}$ of the used system model from u_{k-1} and $\hat{x}_{k-1/k-1}$. Once this step is achieved, the innovation procedure is launched. This one consists in compensating the predicted vector $\hat{x}_{k/k-1}$ using the performed Kalman gain K_k and the measurement vector y_k and generates the estimated optimal vector $\hat{x}_{k/k}$ .The order of the EKF corresponds to the length of the state space vector x.

Table 4.1 shows the processed EKF equations within each step. Note that, with regards to the conventional principle, the proposed algorithm has been reorganized so as to gather the intensive matrix operations. For the chosen non linear model, the calculation of the Kalman gain requires a linearization assumption. The most commonly used linearization method is the first order Taylor approximation [93]-[95]. By this way, the computation of the derivative of f_d and h_d, also called Jacobian matrices F_d and H_d, is needed (relations (4.4) and (4.5)). The covariance matrices P_0, Q and R represent respectively the initial state error, the model noise and the measurement noise.

The stability and robustness of the EKF in the proposed sensorless application is going to be discussed in the next chapter. This is the same for the discussion about the digital implementation tradeoffs.

As for the tuning of the EKF, the covariance matrices have been set according to methodology proposed in [95]. The latter consists of a trial-and-error procedure which gives some guidelines to be followed in order to set the EKF estimation behavior during the transient and at steady state. Then varying the matrix P_0 yields different transient amplitudes. Varying Q and R yields the setting of the transient duration and the steady state behavior. It is in a common practice to assume these covariance matrices to be diagonal and invariant.

Table 4.1: Discrete-time EKF Algorithm

Prediction Step	
$\hat{x}_{k/k-1} = f_d(\hat{x}_{k-1/k-1}, u_{k-1})$	(4.3)
EKF Compensator – Kalman gain calculation	
Jacobian matrices :	$F_{dk} = \left.\frac{\partial(f_d)}{\partial x}\right\|_{x=\hat{x}_{k-1/k-1}}$ (4.4)
	$H_{dk} = \left.\frac{\partial(h_d)}{\partial x}\right\|_{x=\hat{x}_{k-1/k-1}}$ (4.5)

Covariance matrix prediction :	$P_{k/k-1} = F_{dk}.P_{k-1/k-1}.F_{dk}^{t} + Q$; **Initial value P_0**	(4.6)
Kalman gain calculation :	$K_k = P_{k/k-1}.H_{dk}^{t}.\left(H_{dk}P_{k/k-1}H_{dk}^{t} + R\right)^{-1}$	(4.7)
Updating covariance matrix :	$P_{k/k} = P_{k/k-1} - K_k H_{dk} P_{k/k-1}$	(4.8)
Innovation step		
$\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k\left(y_k - h_d \cdot \hat{x}_{k/k-1}\right)$		(4.9)

6. System state space modeling

The use of an EKF in a sensorless control application demands a reliable and accurate motor model considering the whole physical phenomena (saliency, saturations, core losses, nonlinearities, skin effects …). In contrast, designer has to be aware that the more complete the system, the higher the complexity and the order of the EKF and thus the more constraining is its digital implementation. Then optimizations and hypotheses regarding the simplification of the system model have to be achieved in order to find an optimum between the precision of the model and its complexity.

In this part, the development of the used SSM model is discussed. The first hypotheses regarding model optimization are listed below:

- The SSM has a 3-phase Y connected stator windings
- The 3-phase coordinate system is balanced
- Saturations are neglected
- Core losses and skin effects are neglected
- The friction torque is assumed to be linear regarding the speed
- The rotor current is constant

As far as the coordinate system is concerned, the SSM model can be developed either in stationary frame (α,β) or in rotating frame (d,q). In the following, the two cases have been studied and different versions of the system model are presented. At the end, a quantitative comparison between these models is achieved and the fixed criteria are: the EKF estimation behavior and the model complexity. From the obtained results, the choice of the appropriate system model is made.

6.1. Modeling in (d-q) rotating frame

In the literature, several implementations of sensorless applications have been achieved using a rotating d-q reference frame (e.g. [99], [100]). Various structures of the used state space model have been proposed. Depending on the controlled AC drive, the expected level of performances and the digital implementation constraints, model simplification assumptions have been achieved. To start the state space modeling, let's recall at first the d-q based SSM equations (Table 4.2). These equations have been developed in [Appendix B]. Note that the Park transformation is processed after a Clark transformation with the amplitude-conservation.

Table 4.2: SSM model in d-q reference frame

Electrical equations	
$v_{sd} = R_s i_{sd} + L_{sd}\frac{d(i_{sd})}{dt} - \omega L_{sq} i_{sq}$ $v_{sq} = R_s i_{sq} + L_{sq}\frac{d(i_{sq})}{dt} + \omega L_{sd} i_{sd} + \omega M_{sr} I_{rd}$	(4.10)

Electromagnetic torque

$$T_e = \frac{3}{2} p\left[L_{sd} - L_{sq}\right] i_{sd} i_{sq} + \frac{3}{2} p M_{sr} I_{rd} i_{sq} \tag{4.11}$$

Mechanical equation

$$\frac{J}{p}\frac{d\,\omega}{dt} = T_e - \frac{f_L}{p}\omega - T_L \tag{4.12}$$

The corresponding state space model is then expressed in relation (4.13):

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x) \end{aligned} \tag{4.13}$$

With,

$$x = [i_{sd} \quad i_{sq} \quad \omega \quad \theta]^t \; ; \; u = [v_{sd} \quad v_{sq} \quad T_L]^t \; ; \; y = [i_{sd} \quad i_{sq}]^t$$

$$f = \frac{d}{dt}\begin{bmatrix} i_{sd} \\ i_{sq} \\ \omega \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_{sd}} \cdot i_{sd} + \frac{L_{sq}}{L_{sd}} \cdot \omega \cdot i_{sq} \\ \frac{-R_s}{L_{sq}} \cdot i_{sq} - \frac{L_{sd}}{L_{sq}} \cdot \omega \cdot i_{sd} - \frac{M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \omega \\ \frac{3p^2}{2J}\left[L_{sd} - L_{sq}\right] \cdot i_{sd} \cdot i_{sq} + \frac{3p^2}{2J} M_{sr} \cdot I_{rd} \cdot i_{sq} - \frac{f_L}{J} \cdot \omega \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sd}} & 0 & 0 \\ 0 & \frac{1}{L_{sq}} & 0 \\ 0 & 0 & -\frac{p}{J} \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{sd} \\ v_{sq} \\ T_L \end{bmatrix} \tag{4.14}$$

$$h(x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot x$$

For the observation purpose and in the case where the motor is connected to an unknown load torque (or inaccurately known), this variable is generally estimated. Namely, the value of the load torque has to be added to the state space vector. In most of the cases, it is assumed to be constant because its dynamic can be neglected with regard to the dynamic of the electrical quantities. The new formulation of the sate space model for the observation is expressed in relation (4.15).

$$x = [i_{sd} \quad i_{sq} \quad \omega \quad \theta \quad T_L]^t \; ; \; u = [v_{sd} \quad v_{sq}]^t \; ; \; y = [i_{sd} \quad i_{sq}]^t$$

$$f = \frac{d}{dt}\begin{bmatrix} i_{sd} \\ i_{sq} \\ \omega \\ \theta \\ T_L \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_{sd}} \cdot i_{sd} + \frac{L_{sq}}{L_{sd}} \cdot \omega \cdot i_{sq} \\ \frac{-R_s}{L_{sq}} \cdot i_{sq} - \frac{L_{sd}}{L_{sq}} \cdot \omega \cdot i_{sd} - \frac{M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \omega \\ \frac{3p^2}{2J}\left[L_{sd} - L_{sq}\right] \cdot i_{sd} \cdot i_{sq} + \frac{3p^2}{2J} M_{sr} \cdot I_{rd} \cdot i_{sq} - \frac{f_L}{J} \cdot \omega - \frac{p}{J} \cdot T_L \\ \omega \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sd}} & 0 \\ 0 & \frac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{sd} \\ v_{sq} \end{bmatrix} \tag{4.15}$$

$$h(x) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot x$$

It can be seen in this relation that the order of the state space model is equal to 5. With this model, the whole EKF complexity is equal to 1200 equivalent arithmetic operations. This number can be reduced almost by 35% if the Infinite Inertia hypothesis is adopted. Indeed, this assumption consists in assuming that the dynamic of mechanical quantities is slower than the dynamic of the electrical ones. Namely, the variation of the speed is neglected. Consequently, the obtained model does not include the mechanical equation which makes it independent on the used mechanical load. The obtained state space matrix is then rewritten in relation (4.16). It can be noticed that in the following sections, the presented models are all based on the Infinite Inertia hypothesis.

dq model with external transformation →

$$f = \frac{d}{dt}\begin{bmatrix} i_{sd} \\ i_{sq} \\ \omega \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_{sd}} \cdot i_{sd} + \frac{L_{sq}}{L_{sd}} \cdot \omega \cdot i_{sq} \\ \frac{-R_s}{L_{sq}} \cdot i_{sq} - \frac{L_{sd}}{L_{sq}} \cdot \omega \cdot i_{sd} - \frac{M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \omega \\ 0 \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sd}} & 0 \\ 0 & \frac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{sd} \\ v_{sq} \end{bmatrix} \tag{4.16}$$

The input and output matrices of this model include the (d-q) based voltages and currents. They are obtained after a coordinate Park transformation which uses the estimated position. This is unfortunately the main drawback of using such (d-q) model since the inputs and outputs are conditioned by a quantity which is supposed to be estimated. This issue can be bypassed when the whole necessary Park transformations are included in the system model according to relation (4.17). In this case the input and output matrices include terms of position which are taken into account in the EKF treatment. The input and output vectors are composed of the stationary α-β based voltages and currents.

$$x = [i_{sd} \quad i_{sq} \quad \omega \quad \theta]^t ;\ u = [v_{s\alpha} \quad v_{s\beta}]^t ;\ y = [i_{s\alpha} \quad i_{s\beta}]^t$$

dq model with internal transformation →

$$f = \frac{d}{dt}\begin{bmatrix} i_{sd} \\ i_{sq} \\ \omega \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_{sd}} \cdot i_{sd} + \frac{L_{sq}}{L_{sd}} \cdot \omega \cdot i_{sq} \\ \frac{-R_s}{L_{sq}} \cdot i_{sq} - \frac{L_{sd}}{L_{sq}} \cdot \omega \cdot i_{sd} - \frac{M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \omega \\ 0 \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sd}} \cdot cos(\theta) & \frac{1}{L_{sd}} \cdot sin(\theta) \\ -\frac{1}{L_{sq}} \cdot sin(\theta) & \frac{1}{L_{sq}} \cdot cos(\theta) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{s\alpha} \\ v_{s\beta} \end{bmatrix} \tag{4.17}$$

$$h(x) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \end{bmatrix} \cdot x$$

6.2. Modeling in (α-β) stationary frame

In the case of α-β based model, the state space inputs and outputs are position independent. Thus, these variables are not affected by the estimated position. In the literature, numerous sensorless applications using EKF and based on (α-β) frame have been proposed, [93]-[95]. In most of the cases, the implemented control system is based on non-salient synchronous motors. When it comes to a salient motor, the state space modeling in (α-β) may seem to be a tricky task since the saliency is to be taken into account. To prop up this statement, the state space modeling is discussed and some simplification assumptions that allow reducing the model complexity are proposed. Here again we start by presenting the α-β based equations (obtained after Clark transofmations) where the saliency is taken into account (Table 4.3). These equations have been developed in [Appendix B]

Table 4.3: SSM model in α-β reference frame

Electrical equations

$$v_{s\alpha\beta} = R_s[i_{s\alpha\beta}] + L_1\frac{d[i_{s\alpha\beta}]}{dt} + 3L_2\omega\begin{bmatrix} -\sin(2\theta) & \cos(2\theta) \\ \cos(2\theta) & \sin(2\theta) \end{bmatrix}[i_{s\alpha\beta}] + \frac{3}{2}L_2\begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix}\frac{d[i_{s\alpha\beta}]}{dt} + M_{sr}I_{rd}\omega\begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \quad (4.18)$$

Electromagnetic torque

$$T_e = \frac{3}{2}pM_{sr}I_{rd}\left(-i_{s\alpha}\cdot\sin(\theta) + i_{s\beta}\cdot\cos(\theta)\right) + \frac{9}{2}pL_2\cdot i_{s\alpha}\cdot i_{s\beta}\cdot\cos(2\theta) - \frac{9}{4}pL_2\cdot(i_{s\alpha}{}^2 - i_{s\beta}{}^2)\cdot\sin(2\theta) \quad (4.19)$$

Mechanical equation

$$\frac{J}{p}\frac{d\omega}{dt} = T_e - \frac{f_L}{p}\omega - T_L \quad (4.20)$$

To develop the corresponding state space model, the electrical equations can be rewritten so as to gather terms in stator currents and terms in the derivative of currents as follows.

$$v_{s\alpha\beta} = [F(\omega,\theta)]\cdot[i_{s\alpha\beta}] + [G(\theta)]\cdot\frac{d[i_{s\alpha\beta}]}{dt} + [E(\omega,\theta)] \quad (4.21)$$

Where,

$$[F] = R_s\cdot[I_2] + 3L_2\omega\cdot\begin{bmatrix} -\sin(2\theta) & \cos(2\theta) \\ \cos(2\theta) & \sin(2\theta) \end{bmatrix} \quad (4.22)$$

$$[G] = L_1\cdot[I_2] + \frac{3}{2}L_2\cdot\begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix} \quad (4.23)$$

$$[E] = M_{sr}\cdot I_{rd}\cdot\omega\cdot\begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \quad (4.24)$$

From this settlement, the state space model based on the infinite inertia hypothesis is given in the following relation,

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x) \end{aligned} \quad (4.25)$$

With,

$$x = [[i_{s\alpha\beta}] \quad \omega \quad \theta]^t;\ u = [v_{s\alpha} \quad v_{s\beta}]^t;\ y = [i_{s\alpha} \quad i_{s\beta}]^t$$

$$f = \frac{d}{dt}\begin{bmatrix} [i_{s\alpha\beta}] \\ \omega \\ \theta \end{bmatrix} = \begin{bmatrix} -[G]^{-1}\cdot([F]\cdot[i_{s\alpha\beta}] + [E]) \\ 0 \\ \omega \end{bmatrix} + \begin{bmatrix} [G]^{-1} & 0 \\ 0 & 0 \end{bmatrix}\cdot[v_{s\alpha\beta}] \quad (4.26)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

This relation shows clearly that this model is quite complex. This is mainly due to the matrix treatments in which matrix multiplication and inversion have to be processed. Furthermore, according to the EKF algorithm, the Jacobian matrix for linearization seems to be even more complex and constraining for digital implementation. An alternative to overtake this issue consists in rewriting the whole electrical equations so as to expand them as follows [101],

$$\begin{aligned} v_{s\alpha\beta} &= R_s \cdot \left[i_{s\alpha\beta}\right] + L_{sq} \cdot \frac{d\left[i_{s\alpha\beta}\right]}{dt} + \omega (L_{sd} - L_{sq}) \cdot i_{sd} \cdot \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \\ &+ (L_{sd} - L_{sq}) \cdot \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \cdot \frac{d\left[i_{s\alpha\beta}\right]}{dt} + M_{sr} I_{rd} \cdot \omega \cdot \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \end{aligned} \tag{4.27}$$

In this equation, the direct current i_{sd} has been introduced. It has been obtained after doing the appropriate transformation of the former equations. L_{sd} and L_{sq} are the inductances expressed in rotating frame according to the following relations,

$$L_1 = \frac{L_{sd} + L_{sq}}{2} \quad ; \quad L_2 = \frac{L_{sd} - L_{sq}}{3} \tag{4.28}$$

From these rewritten equations two potential simplifications are possible, [101].

- The first one consists in assuming that the current controller has usually its direct current reference set to zero. By this way, the terms containing i_{sd} and its derivative can be eliminated. The state space model is then,

$$\xrightarrow{\substack{\alpha\beta \text{ model with} \\ i_{sd}=0 \\ \text{approximation}}} \quad f = \frac{d}{dt}\begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ \omega \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_{sq}} & 0 & \frac{M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \sin(\theta) & 0 \\ 0 & \frac{-R_s}{L_{sq}} & \frac{-M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ \omega \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sq}} & 0 \\ 0 & \frac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{s\alpha} \\ v_{s\beta} \end{bmatrix} \tag{4.29}$$

This assumption is adapted to flux oriented AC drive controllers where the direct current reference is maintained to zero. However, this is not adapted for controllers that require a variation of this current, for example in over-speed applications.

- The second assumption consists in assuming that i_{sd} is not equal to but can be considered as constant (slow variation). In this case the electrical equations are rewritten by maintaining the term with i_{sd} and eliminating the term with its derivative. The new state space model is then,

$$\xrightarrow{\substack{\alpha\beta \text{ model with} \\ di_{sd}/dt=0 \\ \text{approximation}}} \quad \frac{d}{dt}\begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ \omega \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{L_{sq}} & 0 & \frac{M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \sin(\theta) + (L_{sd} - L_{sq}) \cdot i_{sd} \cdot \sin(\theta) & 0 \\ 0 & \frac{-R_s}{L_{sq}} & \frac{-M_{sr}}{L_{sq}} \cdot I_{rd} \cdot \cos(\theta) - (L_{sd} - L_{sq}) \cdot i_{sd} \cdot \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ \omega \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sq}} & 0 \\ 0 & \frac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{s\alpha} \\ v_{s\beta} \end{bmatrix} \tag{4.30}$$

6.3. Choice of the model

In order to make a comparison between each of the developed SSM models, two criteria are taken into account. The first one consists in studying the dynamic performances and the behavior of the EKF estimation for each model. To this purpose, the dynamic of the speed estimation is studied.

The second criterion consists in evaluating the complexity of the model with regards to the digital implementation.

Figure 4.7 highlights the behavior of the EKF in terms of speed estimation dynamic for each of the previously discussed models. These models are all based on the infinite inertia approximation. The order of the EKF is then equal to 4 for each case. The values of P_0, Q and R matrices remain the same in both cases. The waveforms are obtained after a functional simulation using Matlab/Simulink environment. During this evaluation the SSM is supplied by the VSI and a sensor-based stator current controller is implemented. Thus, the obtained estimation responses concern the case of an open loop EKF estimation. Open loop means that the estimated speed and position are not injected to the controller.

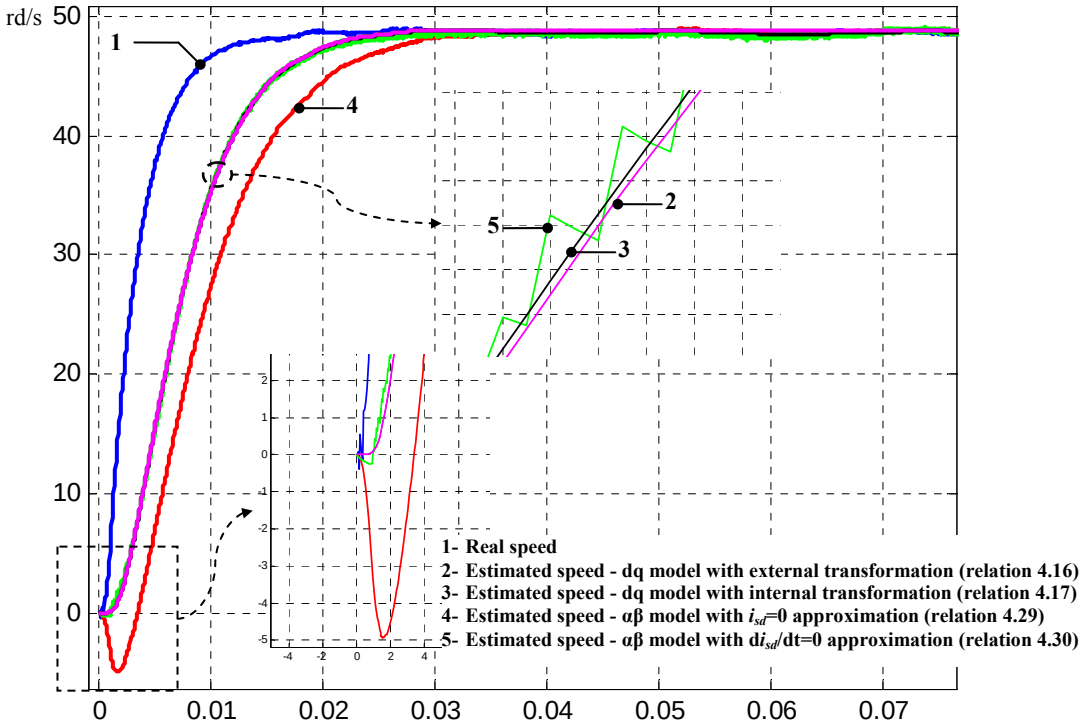


Figure 4.7: EKF speed estimation behavior

The obtained results indicate that, in terms of estimation dynamic and depending on the motor saliency, the α-β based model with i_{sd} equal to zero has the slowest dynamic. However, the evaluation of the complexity indicates that this model has the lowest computational cost. Figure 4.8 presents the evaluation results. In this case, the complexity corresponds to the number of arithmetic operations. As a reminder, this complexity concerns only the system model (i.e. EKF prediction) and its linearization (Jacobian matrix calculation). The complexity of the EKF compensator remains the same since the EKF order is the same in all cases. The trigonometric functions ($sin\theta$, $cos\theta$) are approximated to a fifth degree polynomial function, which corresponds to 10 arithmetic operations.

In the proposed application and according the these results, the chosen model that allies low complexity and acceptable estimation dynamic is the d-q based model with external Park transformations.

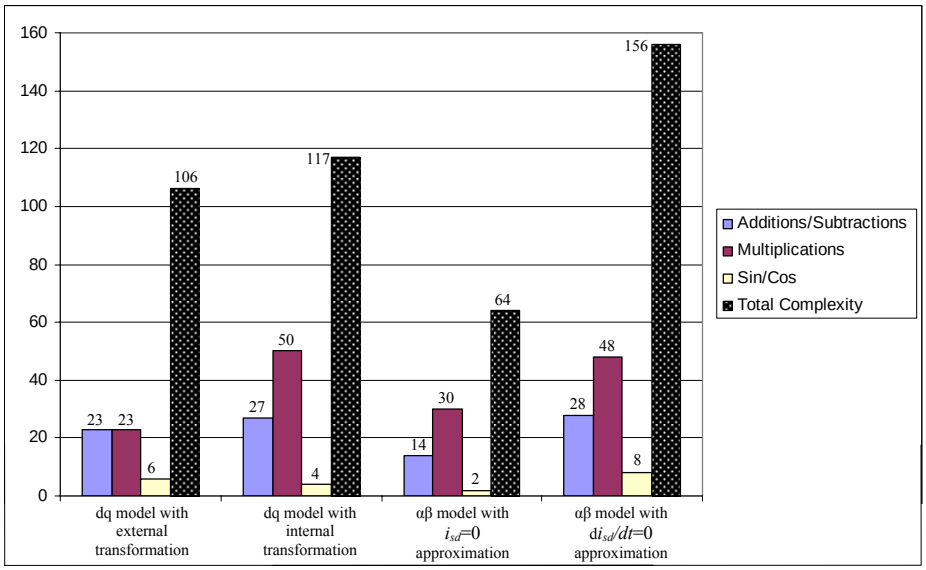


Figure 4.8: Evaluation of the model complexity: EKF prediction + Jacobian matrix

7. Conclusion

This chapter aimed to present the preliminary system specification of the developed sensorless control application. According to the design methodology, presented in chapter 2, the main objective of this specification is to choose the AC drive that will be controlled, the appropriate control strategy and the suitable sensorless method.

As a summary, the chosen AC drive consists of an SSM associated with a controller mechanical load and fed by a 3-phase 2-level VSI. The chosen stator current controller is based on PI-regulator control technique. The implemented PWM module leans on a CB-ZSS-PWM technique. The developed speed controller is based on a P-PI regulator. The chosen sensorless method is the observer method which uses the EKF. Finally, the implemented system model for the EKF treatment is the d-q based model with infinite inertia hypothesis.

Once this preliminary development step is achieved, the next task (chapter 5) is the development of the corresponding algorithm. This process consists of a set of steps during which designer makes the functional validation and prepares the algorithm for the FPGA digital implementation.

Chapter 5

FPGA-based sensorless control for synchronous AC drive - Algorithm development

1. Introduction

In this chapter, author deals with the algorithm development step, during which the whole sensorless control algorithm is developed and validated. After having made a preliminary system specification, the main task here is to make the necessary functional validation. The whole control system is then simulated and validated and the developed algorithm is prepared for the digital implementation. According to the design methodology (chapter 2, Figure 2.18), the listed-below steps have been followed.

Modular partitioning (Part 2) which aims to divide the algorithm into independent and reusable modules of different levels of granularity.

Continuous-time functional simulation (Part 3) which aims to validate the functionality of the closed-loop system using the friendly Matlab/Simlink simulation tools. Also, the configuration of the implemented regulators is achieved during this step.

Digital realization (Part 4) during which designer makes the appropriate discretization assumption and chooses the sampling period. The realization structure is then designed. This is followed by the normalization of the algorithm and the choice of the appropriate fixed-point data format.

Algorithm optimization (Part 5) which is a necessary assumption especially in the case of complex algorithms such as the implemented EKF. This step aims to simplify the processed equations so as to reduce the computational cost.

Discrete-time, fixed-point simulation (Part 6) which is the ultimate algorithm validation step. It aims to make a final verification of the developed discrete-time fixed-point sensorless controller.

2. Modular partitioning

The modular partitioning consists in dividing the whole sensorless algorithm into independent and reusable modules with different levels of granularity. In the case of the developed sensorless controller, five levels of hierarchy have been defined (Figure 5.1). At the lowest level, the basic operators such as logic and arithmetic operators are listed. From a functional point of view, these operators can be considered as fine-grain operators [62], [36]. Then at a higher level of granularity, the matrix operators (used in EKF module) are defined. The third level contains functions dedicated to the control (PI-regulator, P-regulator, Anti-windup PI-regulator, Park transformations, PWM module and Hysteresis regulator) and functions dedicated to the EKF observation (Prediction module, EKF compensator and Innovation module). The interfacing modules are also included here since they are at the same level of hierarchy. At the fourth level, coarser grain modules such as the P-PI speed controller, the stator and rotor current controllers and the EKF module are listed. Finally, at the highest level, the global sensorless controller is defined.

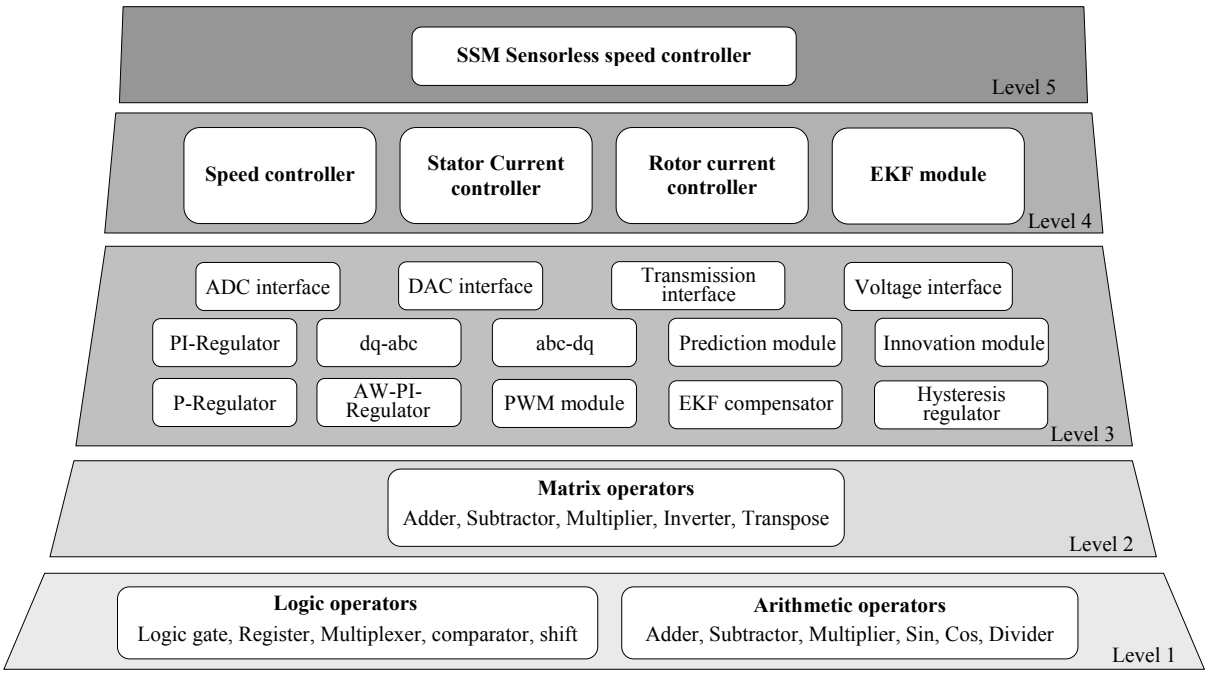


Figure 5.1: Sensorless controller – Modular partitioning

3. Continuous-time functional simulation

The continuous-time functional simulation aims to verify the functionality of the whole control application. This is made with the use of the friendly Matlab/Simulink environment where the functional model of the control system is designed using Simulink continuous-time blocks.

In the proposed sensorless controller the implemented EKF module is based on the discrete-time EKF algorithm. Thus, the presented continuous-time functional simulation concerns only the validation of the speed and current controllers. The objective here is to validate the speed and current responses according to the desired dynamic performances. During this simulation, the rotor current is assumed to be constant and the PWM switching frequency has been set to 10 kHz. Also, it is worth noticing that this simulation has been achieved at non-zero mechanical load conditions. The chosen load torque T_L is proportional to the mechanical speed N (rpm): T_L=0.15*N. This load torque remains the same for the whole simulations that are presented in this chapter. Figure 5.2 presents the corresponding Matlab/Simulink block diagram.

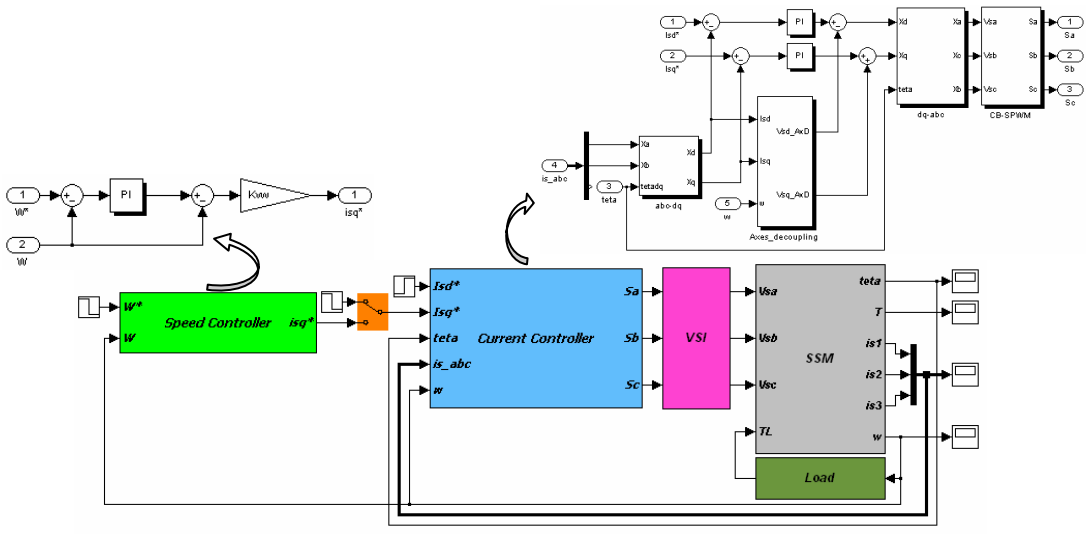


Figure 5.2: Sensorless controller – Modular partitioning

At this stage, the directly measured rotor position and speed are used. The tuning of the developed regulators is made according to [62] and [81] and summarized in [Appendix C]. The whole current control closed-loop (with axes decoupling) is approximated to a first order transfer function where the time constant is user-defined. The whole speed control closed-loop is approximated to a second order transfer function where the controller coefficients are determined with regards to the expected overshoot and settling time.

First of all, a validation of the stator current controller is achieved. The direct current reference has been set to zero. At start up, a 2A step reference has been applied to the quadrature current reference and at 0.5s a -1A negative step value is applied so as to validate the functionality at the opposite rotating direction.

The obtained simulation results are presented in Figure 5.3. The i_{sd}, i_{sq} current step responses are respectively shown in Figure 5.3(a) and Figure 5.3(b). The stationary frame (α,β) based stator current waveform is presented in Figure 5.3(c). The 3-phase stator currents are highlighted in Figure 5.3(d). The stator voltage V_{sa} is shown in Figure 5.3(e) and the corresponding post-filtered waveform (voltage fundamental) is shown in Figure 5.3(f). The obtained waveforms attest the good functionality of the developed stator current controller and the observed static and dynamic responses correspond exactly to the expected behavior.

As for the functional validation of the speed controller, Figure 5.4 presents the obtained simulation results. A 750-rpm step reference (mechanical speed) has been applied at the beginning and the opposite value is applied at 2s. The speed response and the rotor position are shown in Figure 5.4(a,b). The waveform of the developed torque is shown in Figure 5.4(c). The d-q, 3-phase and (α,β) based stator currents are presented in Figure 5.4(d-g). Finally, the waveforms of the stator voltage and its fundamental are presented in Figure 5.4(h,i). Here again, the expected operating conditions and the desired response dynamic have been successfully obtained.

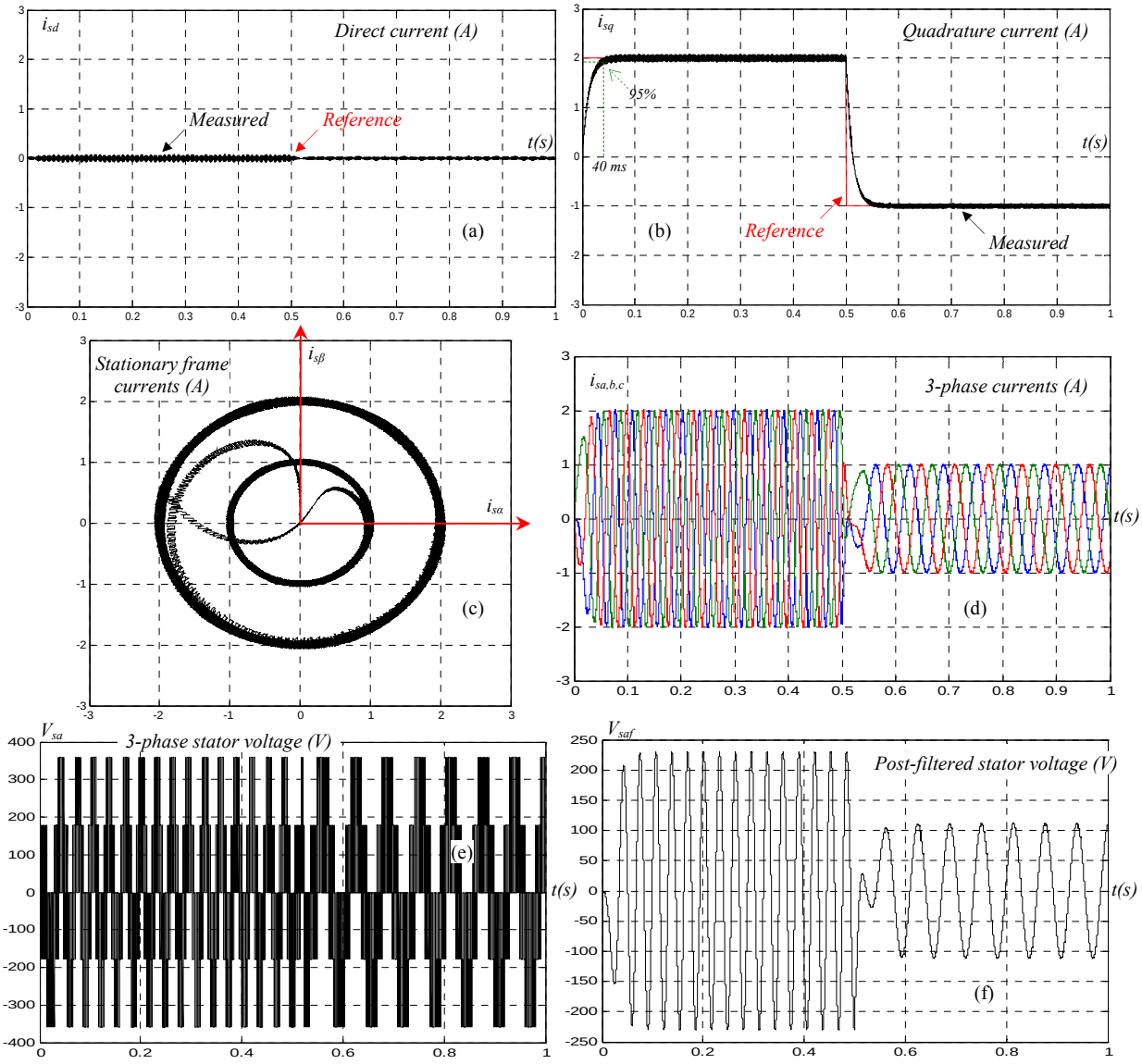


Figure 5.3: Continuous time simulation results – Stator current control validation

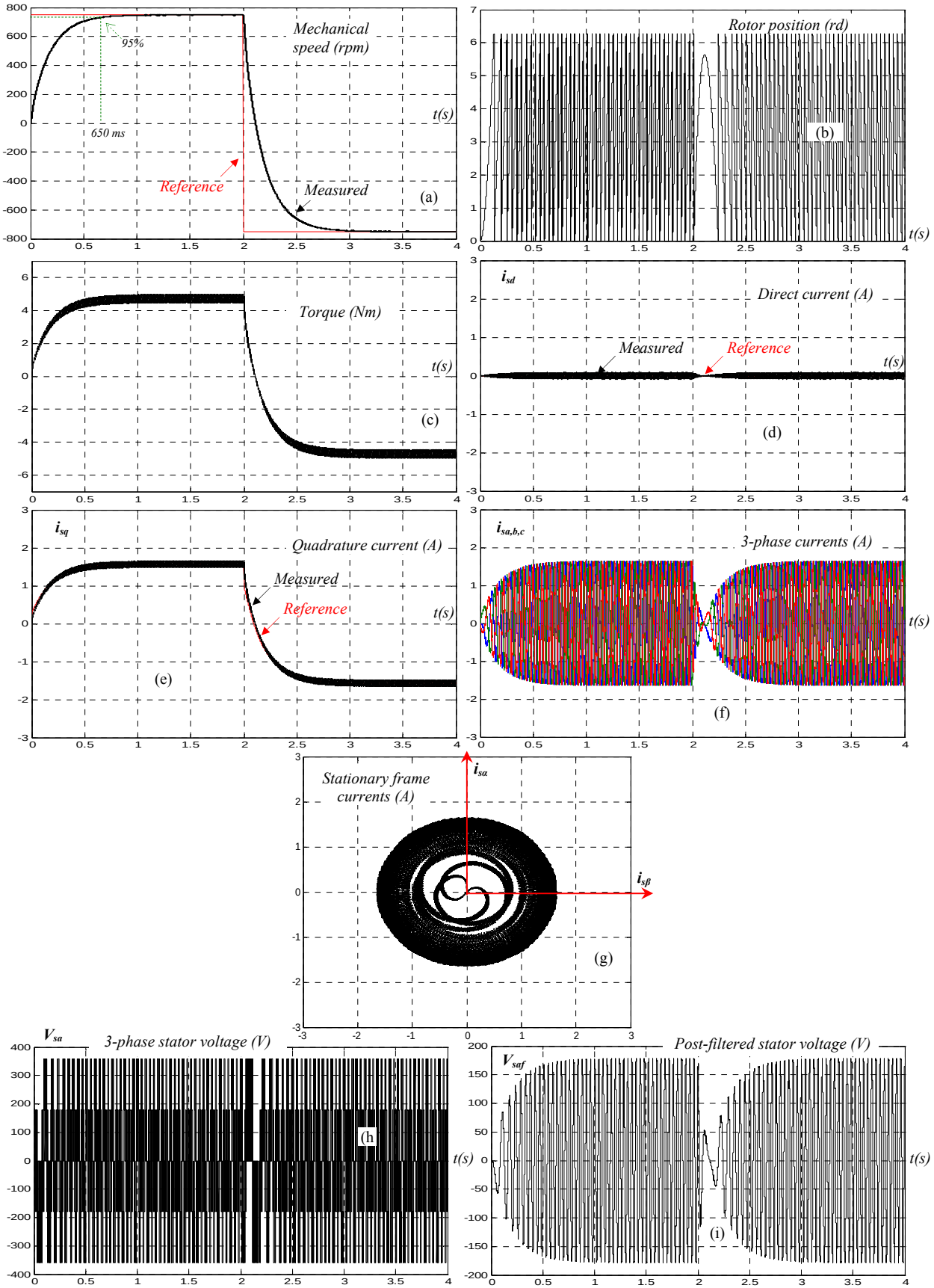


Figure 5.4: Continuous time simulation results – Speed control validation

4. Digital realization

4.1. Discretization and sampling period setting

For the digital implementation purpose, the discretization of the controller is mandatory. In the following, the discretization of the PI-regulator and the discretization of the system model for the EKF treatment are both presented. This is made with the adoption of the digital re-design approach (chapter 2, §6.2.3).

To start with, relation (5.1) presents the s-domain transfer function of a standard PI-regulator.

$$PI(s) = \frac{o(s)}{\varepsilon(s)} = K_p + \frac{K_i}{s} \tag{5.1}$$

The Tustin discretization method has been adopted. The s-domain integrator has been transformed to its z-domain counterpart according to relation (5.2).

$$\frac{1}{s} \rightarrow \frac{T_s}{2}\frac{z+1}{z-1} \tag{5.2}$$

T_s is the sampling period. The obtained discrete-time recurrence equation is expressed in the following relation.

$$o_k = (K_p - \frac{K_i T_s}{2}) \cdot \varepsilon_k + K_i T_s \cdot \sum_{\rho=0}^{k} \varepsilon_\rho \tag{5.3}$$

The developed anti-windup structure is based on the conditional integration (integration clamping) principle [82], [107]. Indeed, depending on the output saturation conditions, the integration is switched on or switched off. Figure 5.5 presents the corresponding block diagram.

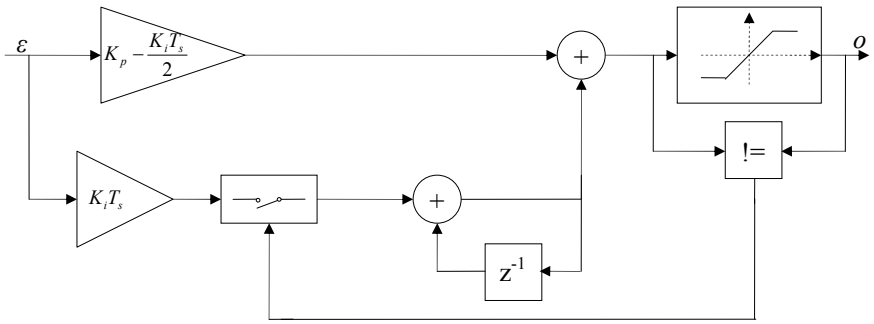


Figure 5.5: Block diagram of the implemented Anti-windup structure

When it comes to the EKF algorithm (chapter 4, part 5), the discretization of the used non linear system model (relation (5.4)) is also required.

$$\begin{aligned} \dot{x} &= f(x,u) + w \\ y &= h(x) + v \end{aligned} \tag{5.4}$$

In the literature, the most commonly used discretization method is the the first order Forward Euler approximation. The latter consists in moving from the s-domain to the z-domain according to the following substitution,

$$s \rightarrow \frac{z-1}{T_s} \tag{5.5}$$

The obtained discrete-time stochastic model is then expressed in relation (5.6).

$$\begin{aligned} x_k &= x_{k-1} + T_s.f(x_{k-1}, u_{k-1}) + w_{k-1} \\ y_k &= h(x_k) + v_k \end{aligned} \tag{5.6}$$

As far as the sampling period is concerned, the choice of the appropriate value is made according to the following settlement. With the used CB-ZSS-PWM technique (chapter 4 §3.2), the generated V_{io}^* voltage must be refreshed at each PWM carrier peak (high or low) and must be constant within each PWM switching period. This is the reason why the chosen sampling period has been set equal to the switching period (100µs in our case). Also, as we are talking about a synchronous PWM control, the fact that the treatment starts at each carrier peak, the value of the sampled stator current can be assumed to be the instantaneous average value within a sampling period.

4.2. Algorithm normalization

The normalization consists in developing a per-unit algorithm where variables are replaced by their corresponding per-unit counterparts with the introduction of base-values, relation (5.7).

$$\gamma_n = \frac{\gamma}{\gamma_B} \tag{5.7}$$

To this purpose, the base-value of each variable is determined according to variable nominal value and also according to the gains that are introduced by the sensors and the ADC board. In our case, the defined base-values are: V_{Bcc} and V_{Bekf} (for the voltage), I_B (for the current), ω_B (for the speed) and θ_B (for the position). V_{Bcc} is used in the stator current controller and V_{Bekf} is used in the EKF module. The corresponding numerical values are listed below,

$$\begin{cases} V_{nom} = 230V \\ I_{nom} = 1.5A \\ N_{nom} = 1500rpm \\ V_{Bcc} = \sqrt{6} \cdot V_{nom} = 563V \\ V_{Bekf} = G_{vsensor} \cdot G_{ADC} \cdot V_{DC} = G_{vsensor} \cdot G_{ADC} \cdot \sqrt{6} \cdot V_{nom} = 936V \\ I_B = G_{isensor} \cdot G_{ADC} \cdot \sqrt{2} \cdot I_{nom} = 20A \\ \omega_B = 2 \cdot p \cdot \frac{\pi}{30} \cdot N_{nom} = 628rd/s \\ \theta_B = 2\pi\ rd \end{cases} \tag{5.8}$$

Then, relation (5.9) presents the normalized discrete-time equation of the direct and quadrature current PI-regulators.

$$\begin{cases} v_{snd\,k} = (K_{pd} - \frac{K_{id}T_s}{2}) \cdot \frac{I_B}{V_{Bcc}} \cdot \varepsilon_{snd\,k} + K_{id}T_s \cdot \frac{I_B}{V_{Bcc}} \cdot \sum_{\rho=0}^{k} \varepsilon_{snd\,\rho} \\ v_{snq\,k} = (K_{pq} - \frac{K_{iq}T_s}{2}) \cdot \frac{I_B}{V_{Bcc}} \cdot \varepsilon_{snq\,k} + K_{iq}T_s \cdot \frac{I_B}{V_{Bcc}} \cdot \sum_{\rho=0}^{k} \varepsilon_{snq\,\rho} \\ \varepsilon_{nid} = i_{snd}^* - i_{snd} \\ \varepsilon_{niq} = i_{snq}^* - i_{snq} \end{cases} \tag{5.9}$$

As for the EKF algorithm, relation (5.10) gives the expression of the normalized system model. As a reminder, the chosen model is based on the d-q SSM equations with the infinite inertia approximation (Chapter 4, relation (4.16)).

$$\begin{aligned} x_{nk} &= f_{dn}(x_{nk-1}, u_{nk-1}) + w_{k-1} = x_{nk-1} + T_s.f(x_{nk-1}, u_{nk-1}) + w_{k-1} \\ y_{nk} &= h_{dn}(x_{nk}) + v_k = h(x_{nk}) + v_k \end{aligned} \tag{5.10}$$

Where x_n is the normalized state space vector. u_n and y_n are respectively the normalized system input and output vectors. The normalized input and output matrices are,

$$x_n = \left[\frac{i_{sd}}{I_B} \quad \frac{i_{sq}}{I_B} \quad \frac{\omega}{\omega_B} \quad \frac{\theta}{\theta_B}\right]^T ; u_n = \left[\frac{v_{sd}}{V_{Bekf}} \quad \frac{v_{sq}}{V_{Bekf}}\right]^T ; y_n = \left[\frac{i_{sd}}{I_B} \quad \frac{i_{sq}}{I_B}\right]^T$$

$$f(x_n,u_n) = \begin{bmatrix} \frac{-R_s}{L_{sd}} \cdot i_{snd} + \frac{L_{sq}}{L_{sd}} \cdot \omega_B \cdot \omega_n \cdot i_{snq} \\ \frac{-R_s}{L_{sq}} \cdot i_{snq} - (\frac{L_{sd}}{L_{sq}} \cdot i_{snd} + \frac{M_{sr} \cdot I_{rnd}}{L_{sq}}) \cdot \omega_B \cdot \omega_n \\ 0 \\ \frac{\omega_B}{\theta_B} \cdot \omega_n \end{bmatrix} + \begin{bmatrix} \frac{V_{Bekf}}{I_B \cdot L_{sd}} & 0 \\ 0 & \frac{V_{Bekf}}{I_B \cdot L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot u_n ; \quad h(x_n) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^t \cdot x_n \tag{5.11}$$

4.3. Fixed-point data setting

Also, for the digital implementation where the fixed-point data are manipulated, the choice of the appropriate format is of great concern. Indeed, the chosen fixed-point format must have a high-enough data precision so as to preserve the algorithm precision as closed as possible to its real data format version (i.e. floating point algorithm version). However, in the case of an FPGA solution, this must be balanced with the available FPGA resources. The longer is the data word length, the heavier is the FPGA architecture. A compromise is then mandatory so as to choose a format that preserves the algorithm performances and that allows an efficient use of the FPGA resources. This last point will be deeply discussed in the next chapter.

In this section, author is presenting the chosen fixed-point format for the whole sensorless controller. To start with, let's recall the general purpose fixed-point representation (Figure 5.6). The latter is divided into two parts: an integer part and a fractional part. This representation is labeled as *s[(i+f)/Qf]* for signed data and *u[(i+f)/Qf]* for unsigned data, *(i+f)* is the total data size, *i* is the number of bits of the integer part, *f* is the number of bits of the fractional part.

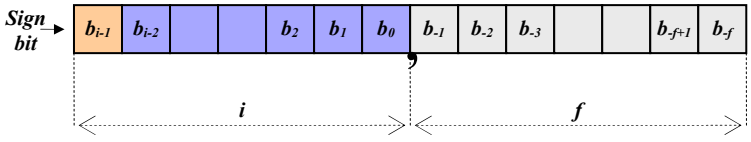


Figure 5.6: Fixed-point representation

The choice of the format has been made after a set of fixed-point simulation tests. The precision and the stability of the fixed-point algorithm (for each module, Figure 5.1) are evaluated and compared to the floating-point algorithm version. Table 5.1 lists the formats that have been chosen for the final digital implementation.

Table 5.1: Fixed-point format setting – speed and current controllers

Modules		Fixed-point format
Speed controller	**PI-regulator**	**s[17Q16]**
	P-regulator	**s[17Q16]**
Rotor current controller	**Hysteresis regulator**	**s[14Q12]**
Stator current controller	**dq-abc**	**s[13Q12]**
	AW-PI-Regulator	**s[20Q18]**
	abc-dq	**s[20Q18]**
	PWM module	**s[13Q12]**
ADC interface		**s[13Q12]**
DAC interface		**u[10Q9]**

As far as the EKF module is concerned, the same comparative simulations have been achieved. The choice of the format is based on the evaluation of the EKF estimation error. The format that leads to the minimum error is then chosen. Figure 5.7 shows the estimation error in the case floating point and fixed-point simulation and for different formats. From the obtained results, the final fixed point format that has been maintained for the FPGA implementation is *s[22Q20]*.

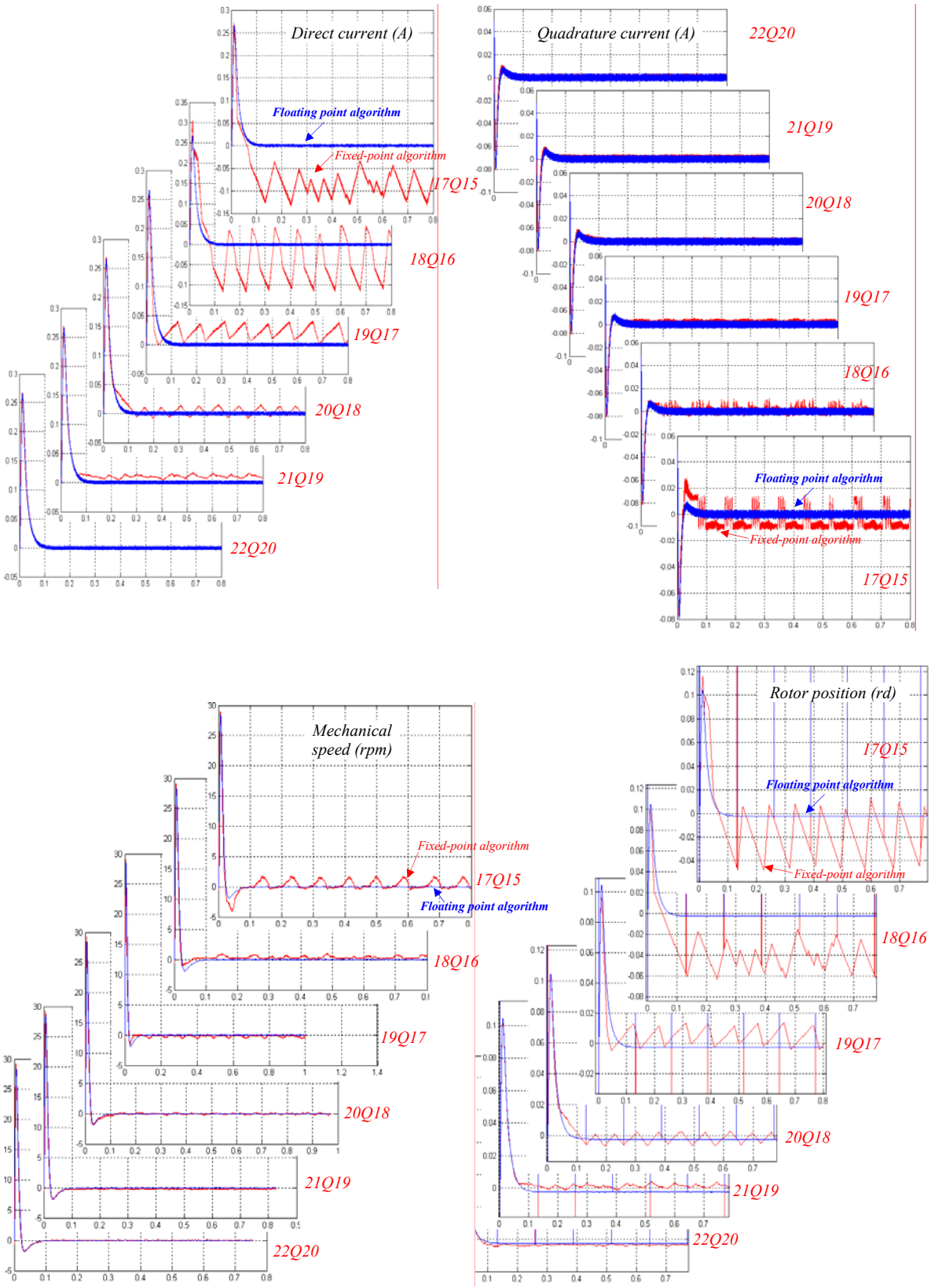


Figure 5.7: Estimation error for different fixed-point data formats

5. Algorithm optimization

Mainly due to the intensive EKF treatment, the development of the whole sensorless controller demands optimization assumptions so as to reduce the computational cost of the algorithm without loosing the required performances. This optimization is especially mandatory in the case when implementing the algorithm in an FPGA with limited hardware resources. This is due to the size of the developed architecture which is also conditioned with the complexity of the algorithm.

In order to prop up this statement, the developed EKF module has been focused on. Then, according the EKF theory and its practical tradeoffs, many algorithm optimizations can be adopted, all with the same objective: the reduction of the computational cost. Examples of algorithmic solutions have been studied by [98] including the so-called Chandrasekhar equation, multi-level EKF and Interlaced EKF. These solutions consist in modifying the conventional structure of the EKF so as to simplify the operated equations and then reduce its intensity. Additional and simpler solutions can also be adopted. For example the infinite Kalman gain K(∞) [98] optimization procedure and the Matrix symmetrization procedure can be good candidates. These two last solutions have been chosen in our case. Before developing them, let's make, at first, a complexity pre-evaluation of the EKF algorithm.

5.1. EKF complexity pre-evaluation

Table 5.2 shows the complexity of the developed EKF module in terms of arithmetic operations. It indicates clearly the intensity and the hugeness the treatment.

Table 5.2: Complexity of the initial EKF algorithm

Operations / Modules		x	+	-	1/x
Prediction		10	6	0	0
Kalman compensator	Jacobian matrices	4	1	0	0
	Kalman gain & covariance matrix	318	244	16	1
Innovation		8	8	8	0
External abc_dq transformations		12	12	0	0
Total		**352**	**271**	**24**	**1**

5.2. Optimization of the EKF algorithm

5.2.1. Offline K(∞) based optimization procedure

This assumption concerns especially the matrix-based Kalman compensator. The principle is to assume that the optimal Kalman gain remains constant at steady state. Then, this gain can be set to offline pre-calculated values avoiding the whole matrix treatment, as shown in Figure 5.8. In this case, only the prediction (based on the normalized SSM model) and the innovation steps are performed.

With this assumption, it can be noticed that the complexity of the developed algorithm has been reduced. However, this approximation has a functional drawback since such algorithm does not ensure a good estimation dynamic, as it will be confirmed during the fixed-point discrete-time simulations.

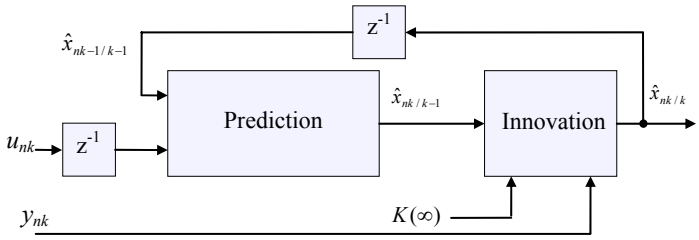


Figure 5.8: Synoptic of the EKF with K(∞) assumption

5.2.2. Matrix symmetrization procedure

Also based on the EKF compensator equations, the principle of this procedure consists in assuming that the initial covariance matrix P_0, the noise matrix Q and the measurement matrix R are diagonal (relation (5.12)).

$$P_0 = \begin{bmatrix} P_{011} & 0 & 0 & 0 \\ 0 & P_{022} & 0 & 0 \\ 0 & 0 & P_{033} & 0 \\ 0 & 0 & 0 & P_{044} \end{bmatrix}; Q = \begin{bmatrix} Q_{11} & 0 & 0 & 0 \\ 0 & Q_{22} & 0 & 0 \\ 0 & 0 & Q_{33} & 0 \\ 0 & 0 & 0 & Q_{44} \end{bmatrix}; R = \begin{bmatrix} R_{11} & 0 \\ 0 & R_{22} \end{bmatrix} \tag{5.12}$$

In this case, it can be easily demonstrated by iterative reasoning that the processed covariance matrices $P_{nk/k-1}$ and $P_{nk/k}$ are symmetrical (relations (5.13) and (5.14)).

$$P_{nk/k-1} = \begin{bmatrix} P_{11}^{nk/k-1} & P_{12}^{nk/k-1} & P_{13}^{nk/k-1} & P_{14}^{nk/k-1} \\ P_{12}^{nk/k-1} & P_{22}^{nk/k-1} & P_{23}^{nk/k-1} & P_{24}^{nk/k-1} \\ P_{13}^{nk/k-1} & P_{23}^{nk/k-1} & P_{33}^{nk/k-1} & P_{34}^{nk/k-1} \\ P_{14}^{nk/k-1} & P_{24}^{nk/k-1} & P_{34}^{nk/k-1} & P_{44}^{nk/k-1} \end{bmatrix} \tag{5.13}$$

$$P_{nk/k} = \begin{bmatrix} P_{11}^{nk/k} & P_{12}^{nk/k} & P_{13}^{nk/k} & P_{14}^{nk/k} \\ P_{12}^{nk/k} & P_{22}^{nk/k} & P_{23}^{nk/k} & P_{24}^{nk/k} \\ P_{13}^{nk/k} & P_{23}^{nk/k} & P_{33}^{nk/k} & P_{34}^{nk/k} \\ P_{14}^{nk/k} & P_{24}^{nk/k} & P_{34}^{nk/k} & P_{44}^{nk/k} \end{bmatrix} \tag{5.14}$$

Consequently, a significant reduction of the computational cost is possible. The matrix treatment can be replaced by scalar treatment with a significant reduction of the number of performed arithmetic operations and processed variables. Furthermore, this assumption does not downgrade the estimation dynamic since a real-time online EKF calculation is still ensured. However, this solution does not prevent designer from the inversion operator which is quite-tricky to implement digitally (especially in the case of an FPGA solution).

5.3. Complexity post-evaluation

Table 5.3 highlights how the presented optimization procedures can significantly reduce the complexity of the EKF algorithm, compared to table 5.2. It can be seen that a reduction of 50% is obtained with the matrix symmetrization assumption and a reduction of 94 % is obtained with the offline K(∞) assumption. However, depending on the used FPGA solution (number of the available hardware resources), an additional effort is, in some cases, quite mandatory so as to adapt the algorithm complexity to the available FPGA resources. To this aim, additional optimization assumptions ought to be achieved during the development of the FPGA architecture. This is covered in the next chapter.

Table 5.3: complexity of the EKF after algorithm optimization

Modules \ Operations		**x**	**+**	**-**	**1/*x***
Prediction	Matrix Symmetrization	10	6	0	0
	K(∞)	10	6	0	0
Kalman compensator	Matrix Symmetrization	149	107	11	1
	K(∞)	0	0	0	0
Innovation	Matrix Symmetrization	8	8	8	0
	K(∞)	8	8	8	0
External abc_dq transformations		12	12	0	0
Total	Matrix Symmetrization	**179**	**133**	**19**	**1**
	K(∞)	**30**	**26**	**8**	**0**

6. Discrete-time, fixed-point simulation

Now as the whole sensorless algorithm is developed and the corresponding digital realization is made, the discrete-time and fixed-point simulation is to be achieved so as to make an ultimate algorithmic validation. Figure 5.9 shows the developed Matlab/Simulink block diagram where the fixed-point toolbox has been used.

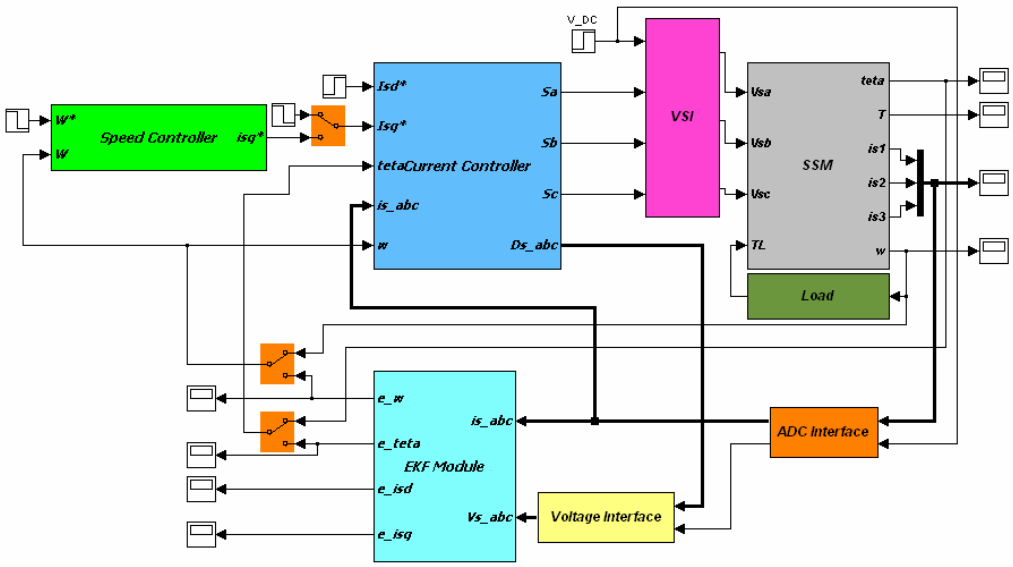


Figure 5.9: Sensorless speed controller – Matlab/Simulink block diagram

The following sections present the obtained results. At first, the validation of the stator current controller is made, followed by the validation of the speed controller. Then, an open-loop validation of the EKF estimation is made. Open-loop means that the estimated position and speed are not injected to the speed and current controllers. The objective of this activity is to validate the estimation dynamic and the robustness of the EKF. Finally, the sensorless controller, where the estimated data are injected to the controllers, is validated.

6.1. Validation of the stator current controller

Figure 5.10 presents the fixed-point simulation results of the stator current controller. The measured rotor position and speed are used. The same operating conditions (current references and load conditions) as during the continuous-time simulation are maintained. The obtained waveforms validate the choices made during the digital realization. Also the obtained static and dynamic performances are as expected.

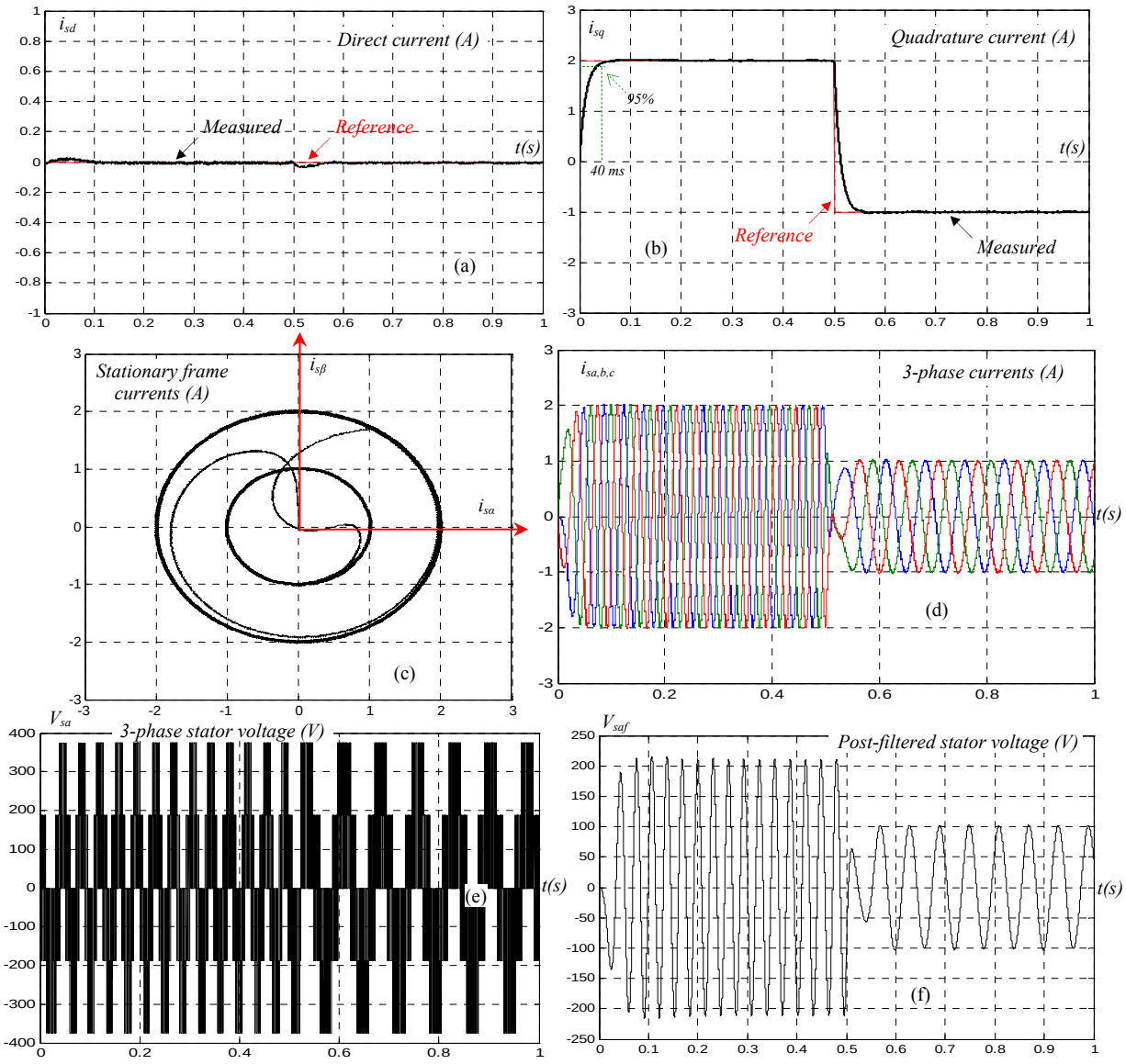


Figure 5.10: Stator current controller – Fixed-point discrete-time simulation results

6.2. Validation of the speed controller

The speed controller is validated in the same motor operating conditions. The obtained fixed-point results are highlighted in Figure 5.11. Here again, the same performances as in the case of continuous-time validation are obtained.

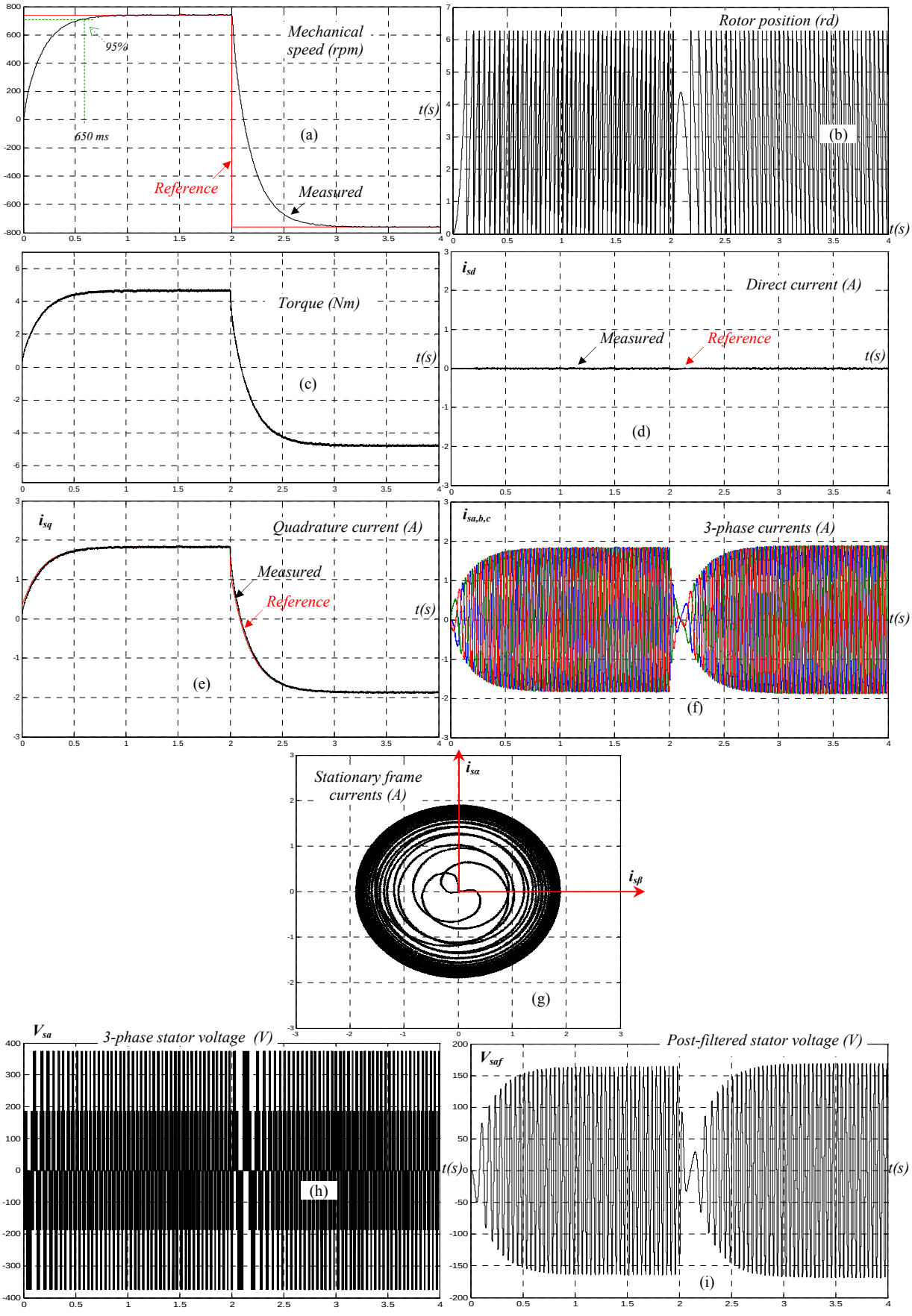


Figure 5.11: Speed controller – Fixed-point discrete-time simulation results

6.3. Validation of the EKF observer

As explained right before, an open-loop validation of the EKF is presented in this section. Indeed, the measured position and speed are injected to the speed and current controllers and the measured and estimated data are compared.

At first, the estimation process of the EKF is validated. The waveforms of the estimated space vector are presented and compared to their measured counterparts. Then, a discussion about how to improve the EKF treatment is made. The robustness against parameter variation is dealt with. Also, the analysis of the EKF robustness against a random initial position error is made. Finally, a comparison in terms of estimation behavior is made between the initial EKF algorithm and its optimized versions (§5.2).

6.3.1. Validation of the estimation process

Figure 5.12 shows the estimation behavior of the implemented EKF observer. The waveforms of the measured and estimated quantities are shown. Figure 5.12(a) presents the estimated direct current $\hat{\imath}_{sd}$ and the actual current i_{sd}. This is the same for the quadrature current (Figure 5.12(b)). The waveforms of the measured and estimated mechanical speed and electrical position are respectively depicted in Figure 5.12(c) and Figure 5.12(d). The obtained results prove the proper EKF estimation behavior. This will also be confirmed during the robustness analysis. The obtained estimation dynamic has been obtained for the following EKF setting,

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; Q = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}; R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.15}$$

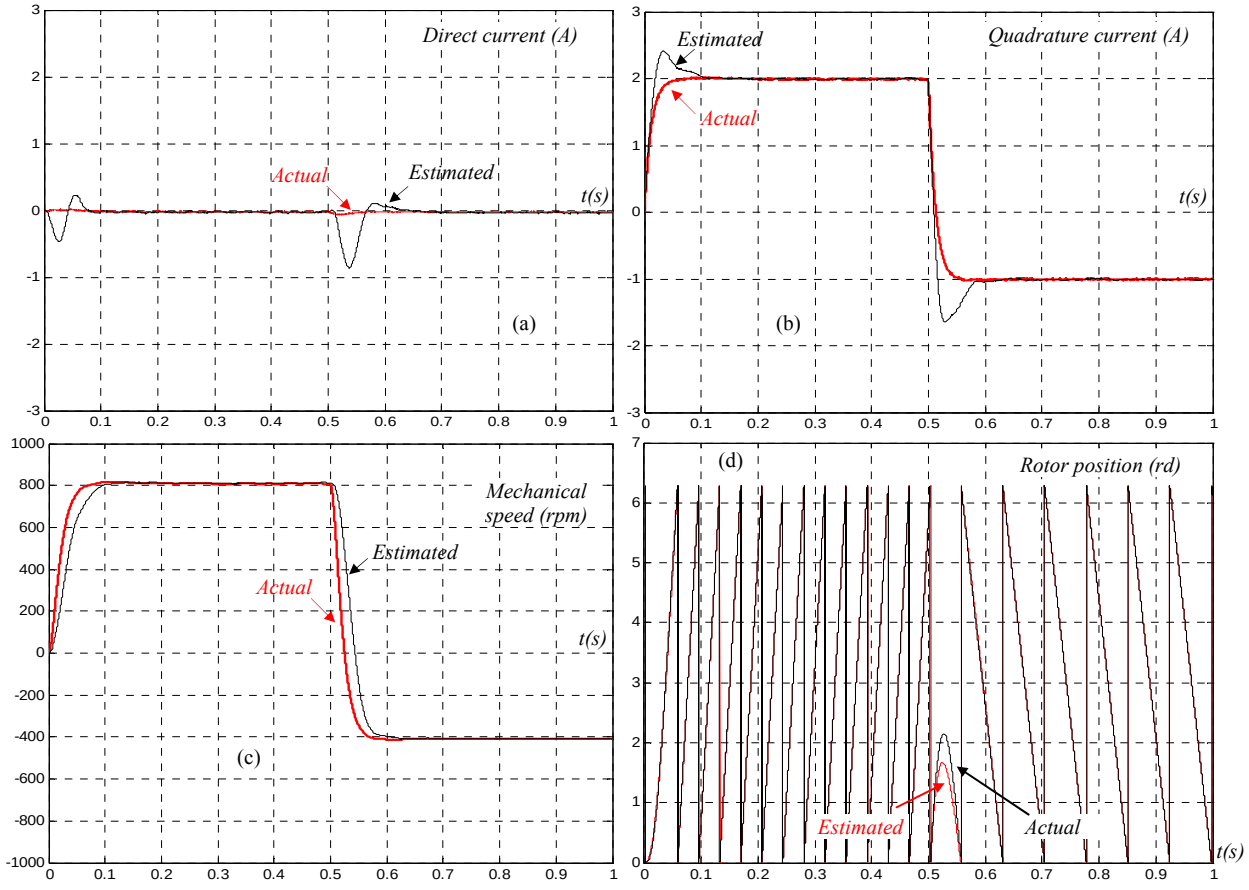


Figure 5.12: Validation of the EKF estimation – Fixed-point discrete-time simulation results

6.3.2. Improvement of the EKF estimation

The manipulation of fixed-point data in such intensive EKF algorithm can be easily a source of convergence problems which affects the EKF stability. Indeed, the limited precision of the data can typically lead to a non-symmetrical covariance matrix P, which is processed after intensive matrix multiplications and a matrix inversion. A simple and efficient way to improve the EKF treatment is to force the covariance matrix P to be symmetrical using the following matrix-update, relation (5.16) [98]. Figure 5.13 presents the estimation error in the case of a standard EKF treatment and the case of the added matrix-update. It can be seen that this matrix-update improves the estimation dynamic and minimizes the precision errors.

$$P_{nk/k} = \frac{P_{nk/k} + P^{t}_{nk/k}}{2} \tag{5.16}$$

Other algorithmic solutions that can improve the EKF treatment and manage its stability are the so-called algorithm decompositions (also called algorithm factorizations). Among the most widespread solutions are the *Square-Root* and *UD* factorizations. They are presented and deeply studied in [98].

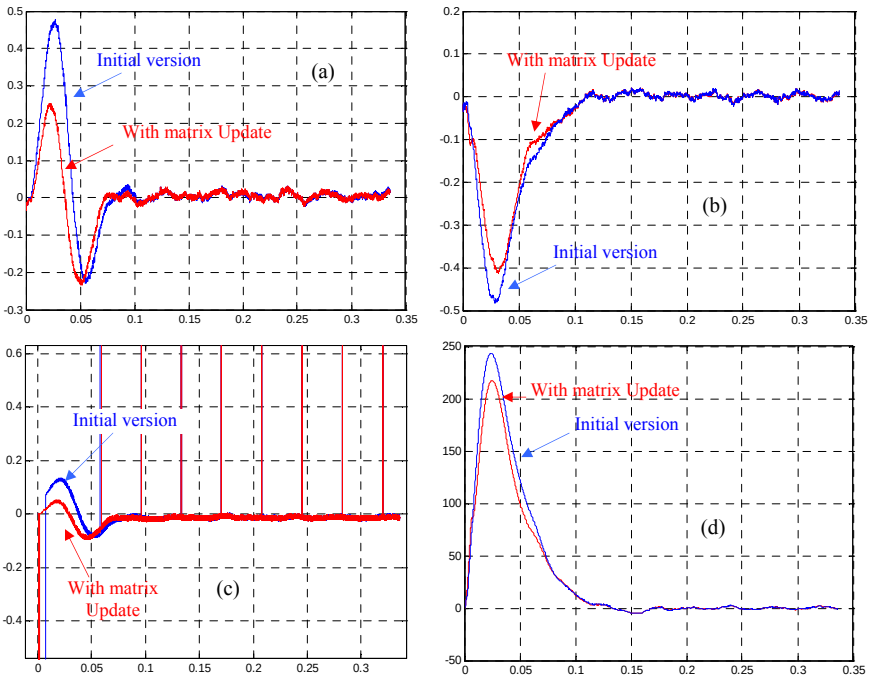


Figure 5.13: Improvement of the EKF estimation – waveforms of the estimation error
(a) Direct current (A); (b): Quadrature current (A); (c): rotor position (rd); (d): mechanical speed (rpm)

6.3.3. Robustness against parameter variation

Up to now, the developed EKF algorithm has been validated in the case where the parameters of the used salient synchronous machine are perfectly known, which is not exactly the case in practice. For this reason, it is necessary to study and evaluate how the EKF is robust when the parameters are varying. For this aim, the influence of the stator resistance R_s variation is evaluated. The following figures (5.14 and 5.15) present the speed and rotor estimation error (at steady state) in the case when the resistance is overestimated (+ 20% and +50%) and also the case when it is underestimated (-20% and -50%). From these simulation results, we can notice a smooth estimation error in both cases, which confirms the robustness of the developed EKF with the chosen tuning values. The same simulations when varying the values of the inductances have been done and the obtained estimation error remains negligible.

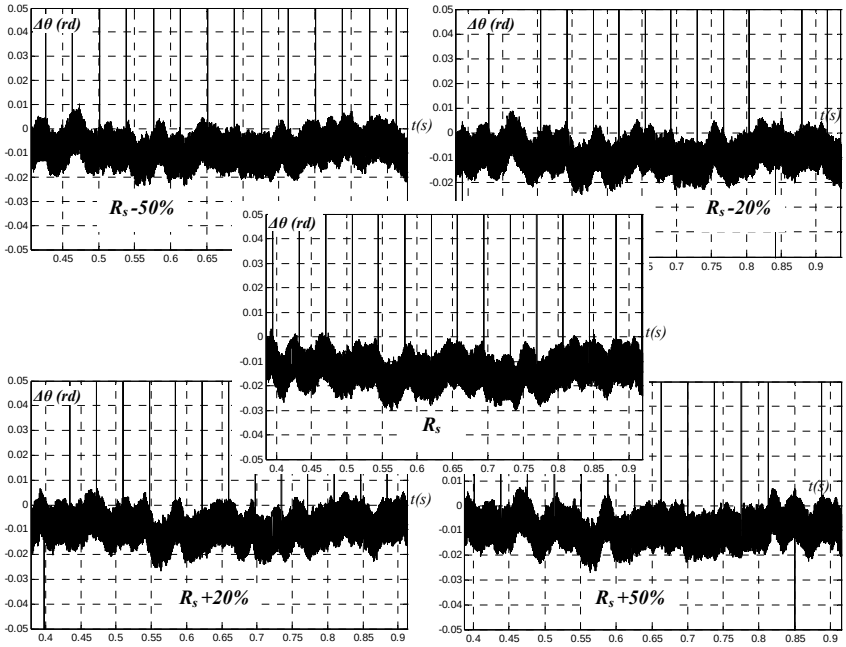


Figure 5.14: Rotor position estimation error – For R_s, R_s-50%, R_s-20%, R_s+20% and R_s+50%

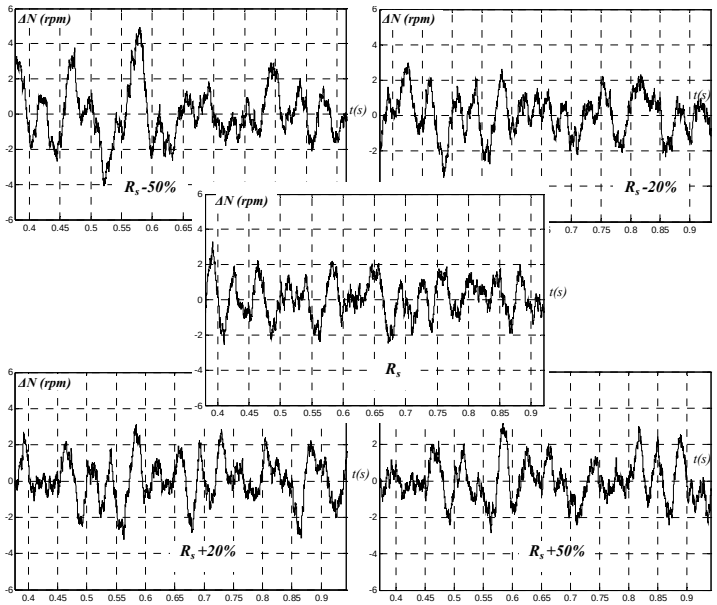


Figure 5.15: Mechanical speed estimation error – For R_s, R_s-50%, R_s-20%, R_s+20% and R_s+50%
Operating mechanical speed: N=800 rpm

6.3.4. Robustness against random initial position error

Another robustness evaluation of the developed EKF is its ability to converge properly and reach the desired motor operating point when the initial position is unknown. Here again, fixed-point simulations have been achieved with random initial rotor position. Figure 5.16 presents the obtained results for *0.18 rad (10°)*, *0.88 rad (50 °)*, *1.745 rad (100 °)* and *3.14 rad (180 °)* initial values. In all cases, a zero steady state position estimation error is ensured.

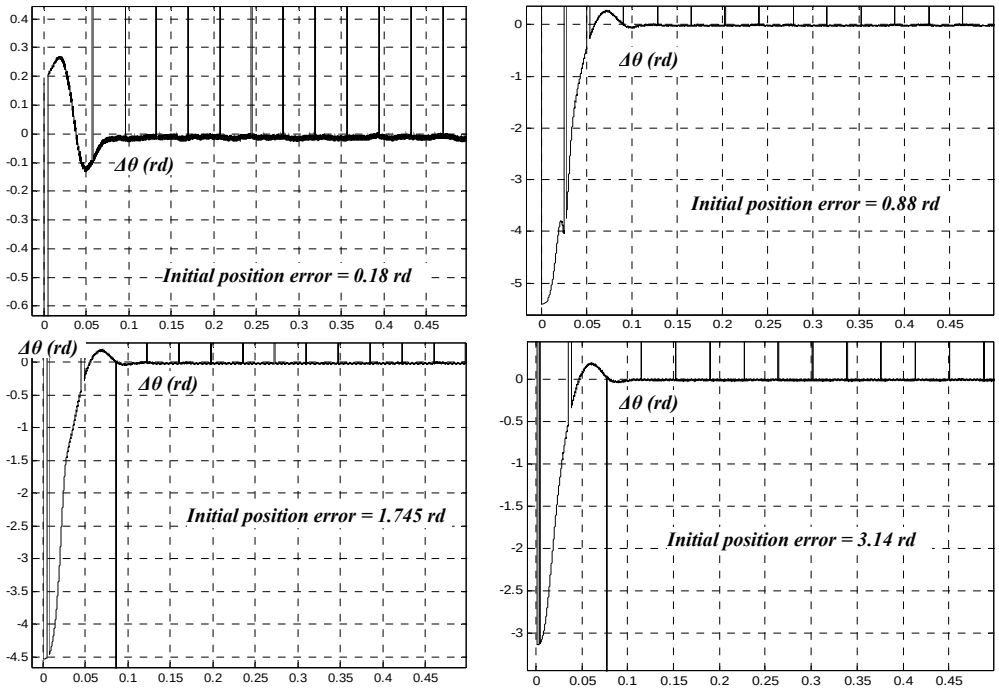


Figure 5.16: Position estimation error for an unknown initial rotor position

6.3.5. Validation of the optimized EKF algorithm

The optimization of the EKF algorithm (reduction of the computational cost) has to be balanced with the ability to keep an acceptable estimation performance. Figure 5.17 presents the obtained speed estimation behavior with the adopted optimization procedures (§5.2). These simulation results indicate that a zero steady state error is ensured by both optimized EKF versions. As for the estimation dynamic, the matrix symmetrization of the EKF algorithm does not affect the behavior. This is not the case for the *K(∞)* version since a good dynamic behavior is not guaranteed.

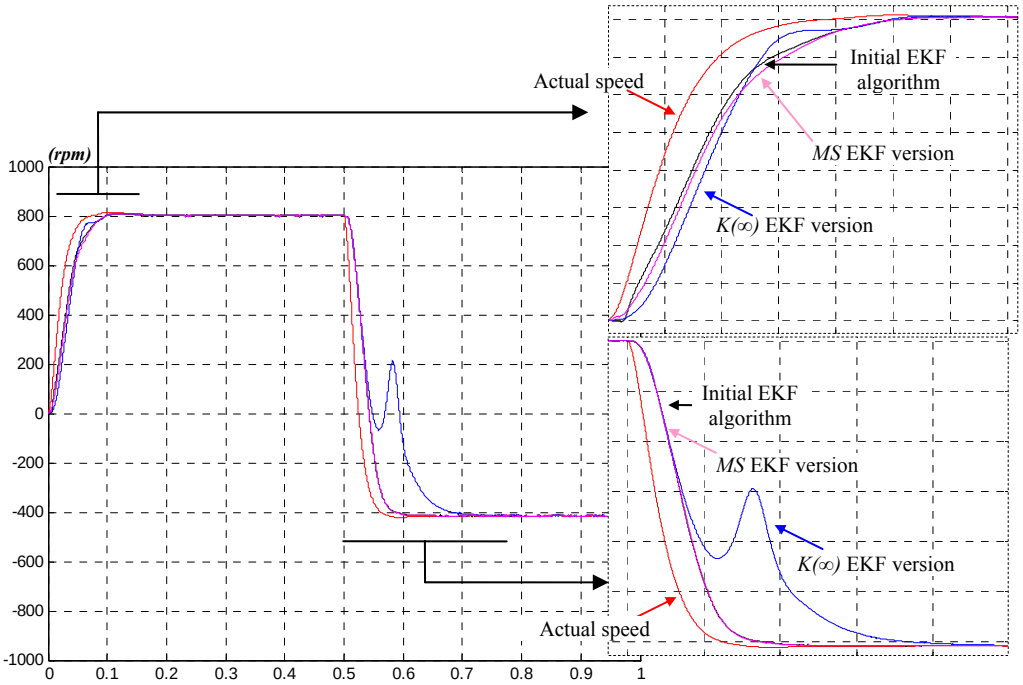


Figure 5.17: Speed estimation dynamic when implementing the optimized EKF algorithms – Mechanical speed

6.4. Validation of the EKF-based sensorless speed controller

Now as both stator current controller, speed controller and the EKF estimation are validated, the ultimate fixed-point simulation aims to validate the whole sensorless controller. The estimated rotor speed and position are then injected to the controllers.

Figure 5.18 presents the obtained results with the same operating conditions as previously (a 750-rpm step reference at 0s, -750 rpm step reference at 2s and the same mechanical load conditions). The speed response and the rotor position are shown in Figure 5.18(a,b). The waveform of the developed torque is shown in Figure 5.18(c). The d-q, 3-phase and (α,β) based stator currents are presented in Figure 5.18(d-g). Finally, the waveforms of the stator voltage and its fundamental are presented in Figure 5.18(h,i).

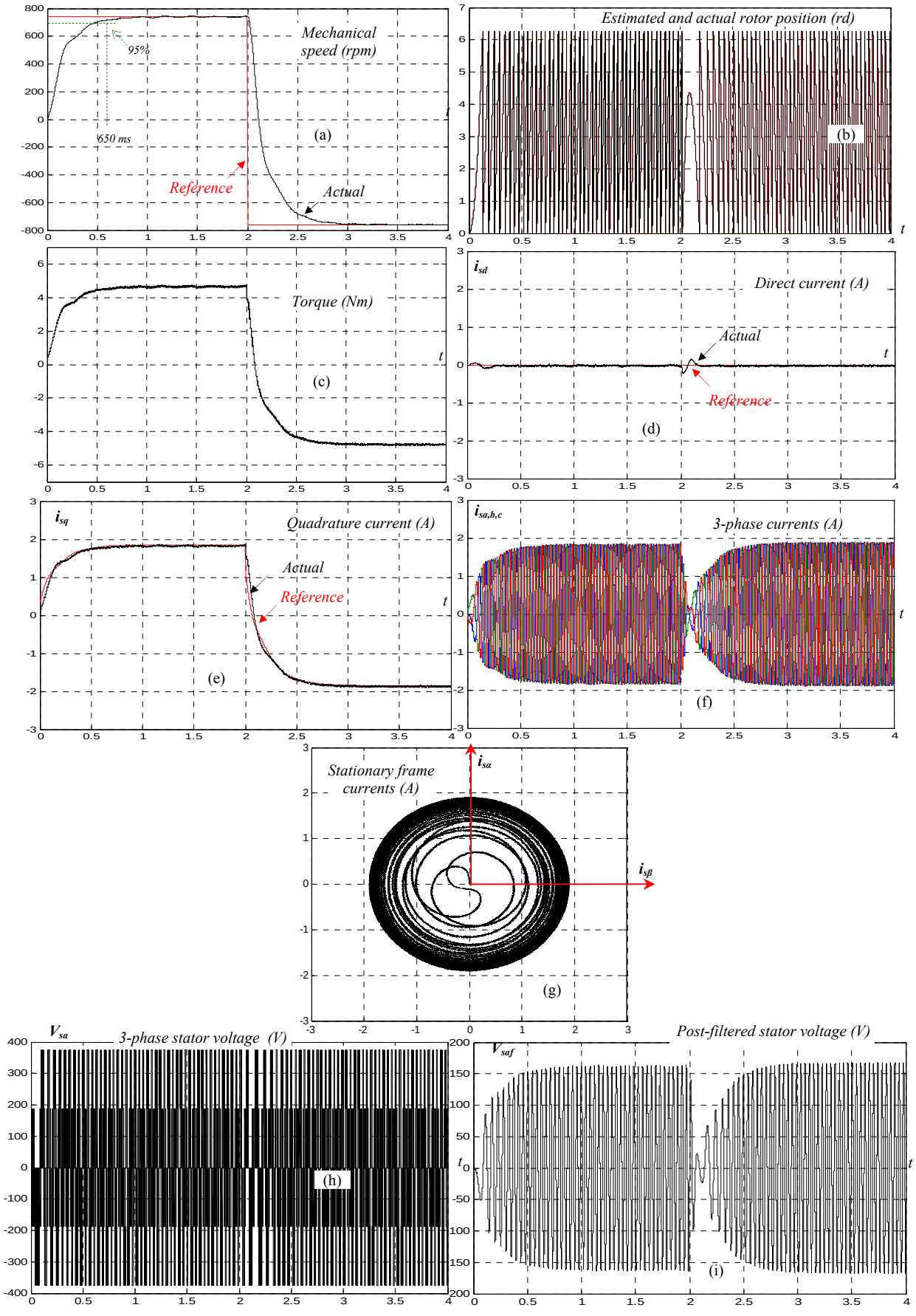


Figure 5.18: Validation of the EKF-based sensorless speed controller

6.4.1. Evaluation of the sensorless control robustness

Figure 5.19 confirms how the developed sensorless controller is robust towards a variation of the load torque. A 750-rpm step reference is applied to the sensorless controller and a variation of 1Nm is applied at 1.6s. The waveform of the speed response and the waveform of the developed torque are respectively shown in Figure 5.19(a) and Figure 5.19(b).

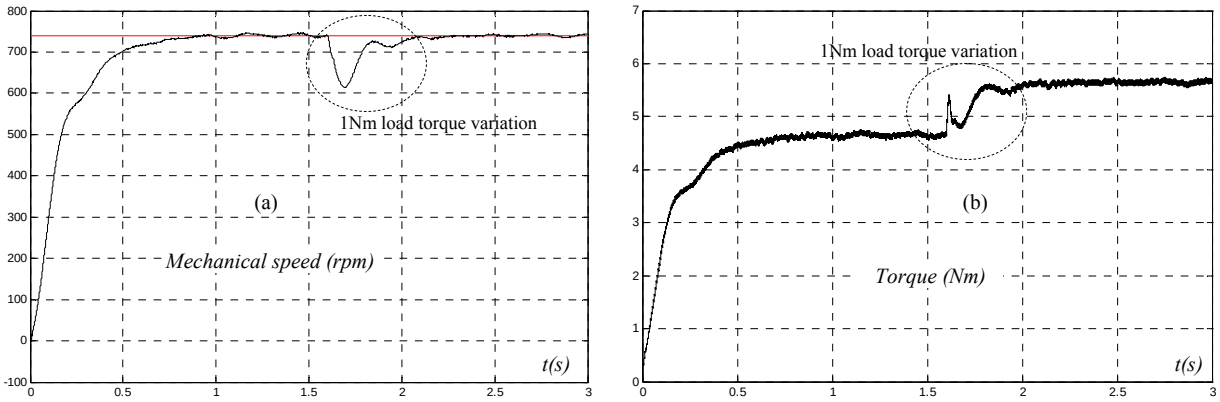


Figure 5.19: Robustness evaluation of the developed EKF-based sensorless speed controller

6.4.2. Expansion to a large operating speed range

In addition to the achieved validation, the evaluation of the developed sensorless controller at an expansive operating speed range is necessary. To this aim, simulations have been made respectively at very low speed, at medium speed and at high speed. Also over-speed conditions are tested. The obtained results indicate that, with the used 1500-rpm SSM, the developed EKF-based sensorless speed controller is able to operate up to 200% of the nominal speed (3000rpm) and down to 8% of the nominal speed (125 rpm). Indeed, at very low speed and at standstill, the EKF does not guarantee a zero steady state estimation error since the back-EMF is very low (zero at standstill). Consequently, this attests the necessity to implement specific estimation method that is dedicated to this operating range. Thus the high frequency injection method has been chosen to this aim. This method is not covered in this work and is the main topic of the associate thesis work [92].

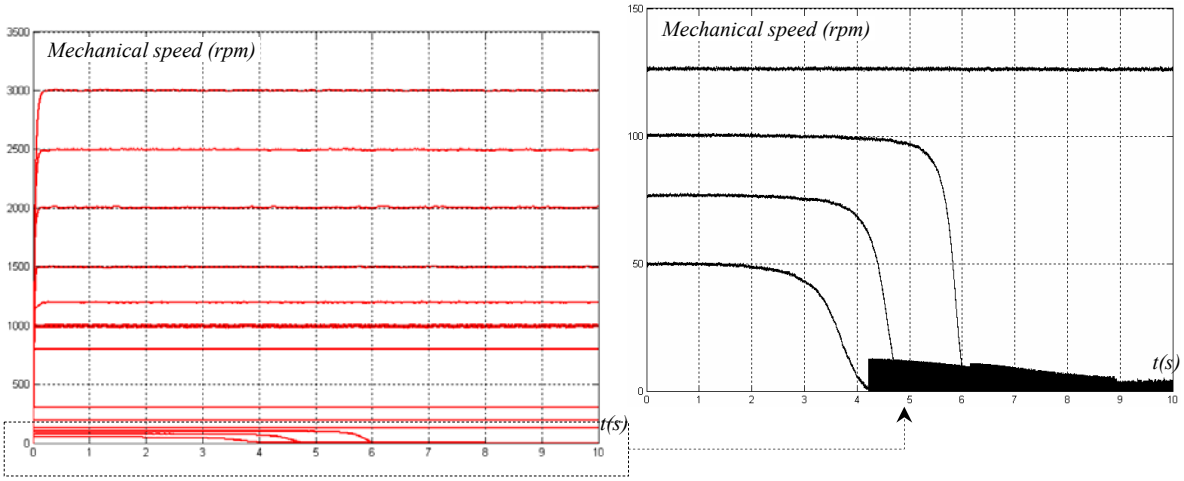


Figure 5.20: Expansion of the developed EKF-based sensorless controller to large operating speed range

7. Conclusion

The objective of this chapter was to present the development of the EKF-based sensorless speed controller. This algorithm development step aims to validate the functionality of the algorithm and prepare it to the digital implementation.

This chapter was organized according to the design methodology. At first, a modular partitioning is achieved. Then, the continuous-time functional validation is made. The configuration and the validation of the speed and current closed-loops are made. This is followed by the digital realization where the discretization, the normalization, the data quantification and the algorithm optimization are both realized. Finally, Fixed-point and discrete time simulations are made in order to validate the whole sensorless controller with regards to the achieved digital realization.

Chapter 6

FPGA-based sensorless control for synchronous AC drive
- FPGA architecture development

1. Introduction

This chapter presents the FPGA architecture development of the proposed sensorless speed controller. According to the design methodology (chapter 2, Figure 2.18), this development takes place after having validated the corresponding algorithm.

The first task of this development is the optimization of the FPGA architecture. This is made according to the optimization procedure, presented in chapter 2. The objective of this procedure is to find the appropriate FPGA architecture that is able to perform the developed algorithm with regards to implementation constraints. In our case, the latter are: the modularity preservation constraint, the area constraint and the timing constraint.

Assuming that the modular partitioning of the algorithm (chapter 5 part 2) has been maintained, a first evaluation of the time/area performances is made. This pre-evaluation is achieved when the parallelism of the algorithm is fully preserved. The objective here is to verify if the corresponding FPGA architecture can be directly designed without any optimization. Otherwise, the Algorithm Architecture Adequation (A^3) methodology is applied to each module so as to factorize the developed architecture and then reduce the consumed FPGA resources.

The optimization is followed by the design of the architecture, the VHDL coding and then the functional validation. During the VHDL synthesis, a time/area evaluation is achieved. Once the obtained time/area performances satisfy the corresponding constraints, the designed FPGA architecture is then ready to be physically implemented.

2. Architecture optimization

2.1. Optimization strategy

Up to now, the EKF-based sensorless control algorithm has been developed and validated. The appropriate fixed-point data format has been chosen and the necessary algorithm optimization has been achieved. Note that in order to enlarge the case studies, both of the EKF versions (initial and optimized versions) have been taken into account during the FPGA architecture development.

The optimization of the corresponding FPGA architecture consists in designing the appropriate architecture that is able to process this algorithm. However, the following FPGA implementation constraints ought to be respected (see chapter 2, §6.3.1).

2.1.1. Modularity constraint

The first development constraint is to preserve the whole modular partitioning that was made during the algorithm development (chapter 5 part 2). According to this partitioning and depending on the level of hierarchy, the FPGA architecture of each module is to be developed. In the case where the area constraint is not satisfied, this partitioning must be revised so as to gather potential modules that lead to the reduction of the FPGA resources.

2.1.2. Area constraint

This constraint imposes the development of a global FPGA architecture that can be easily implemented in a low cost FPGA solution. In the case of having chosen the Xilinx FPGA family, the objective is the possibility to use the available FPGA Spartan series (e.g. Spartan3, Spartan3E and Spartan6).

2.1.3. Timing constraint

The timing constraint here consists is defining an execution time limit, depending on the required control performances. For this aim, the influence of execution time on the control bandwidth of developed EKF speed sensorless controller is evaluated. The case of a high speed synchronous motor has been chosen since the influence of the execution time is more visible at high operating speed [Appendix D]. Figure 6.1 presents the obtained simulation results for different execution times (0%, 10%, 30%, 50%, 70%, 90% and 100% of the sampling period, T_s=100µs). The shorter is the execution time, the larger is the control bandwidth. Finally, the chosen execution time limit is set to 10% of the sampling period.

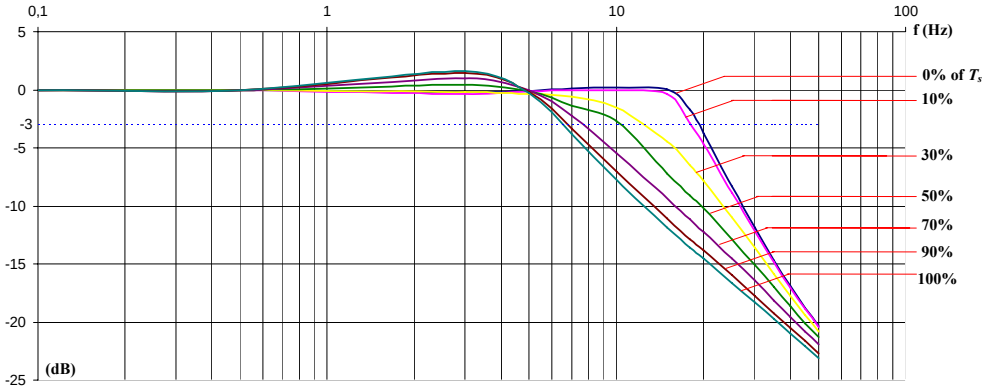


Figure 6.1: Influence of the execution time on the bandwidth of the EKF-based sensorless speed controller

2.1.4. Optimization procedure

The optimization procedure is done with regards the diagram presented in chapter 2, Figure 2.20. The listed-below four steps have been processed:

1- The first task is to evaluate the execution time and estimate the used FPGA resources when the parallelism of the whole algorithm is fully preserved. As an example, the case of the non-optimized EKF version is discussed.

Figure 6.2 presents the timing diagram of the whole sensorless speed controller. The treatment is synchronized with the PWM carrier peaks. Furthermore, when the timing constraint is satisfied, it is possible to launch the treatment before and then refresh the switching signals at the PWM peaks. With the used ADC devices, the necessary conversion time is equal to 2.4µs. Thus, in the case of a full preservation of the algorithm parallelism, the execution time of the whole sensorless controller is expressed according to relation (6.1). With a fully pipelined structure, this execution time can be rewritten as a function of the latency and the operating clock period. In our case, the new expression is given by relation (6.2).

It can be seen from this relation that when using a high clock frequency, the obtained total execution time satisfies the timing constraint. However, when estimating the necessary FPGA resources to implement this fully parallel algorithm, the obtained results (Table 6.1) indicate that the corresponding architecture cannot be supported by any of the available low cost FPGA solutions. Consequently, the area constraint is not satisfied which leads to the second optimization step.

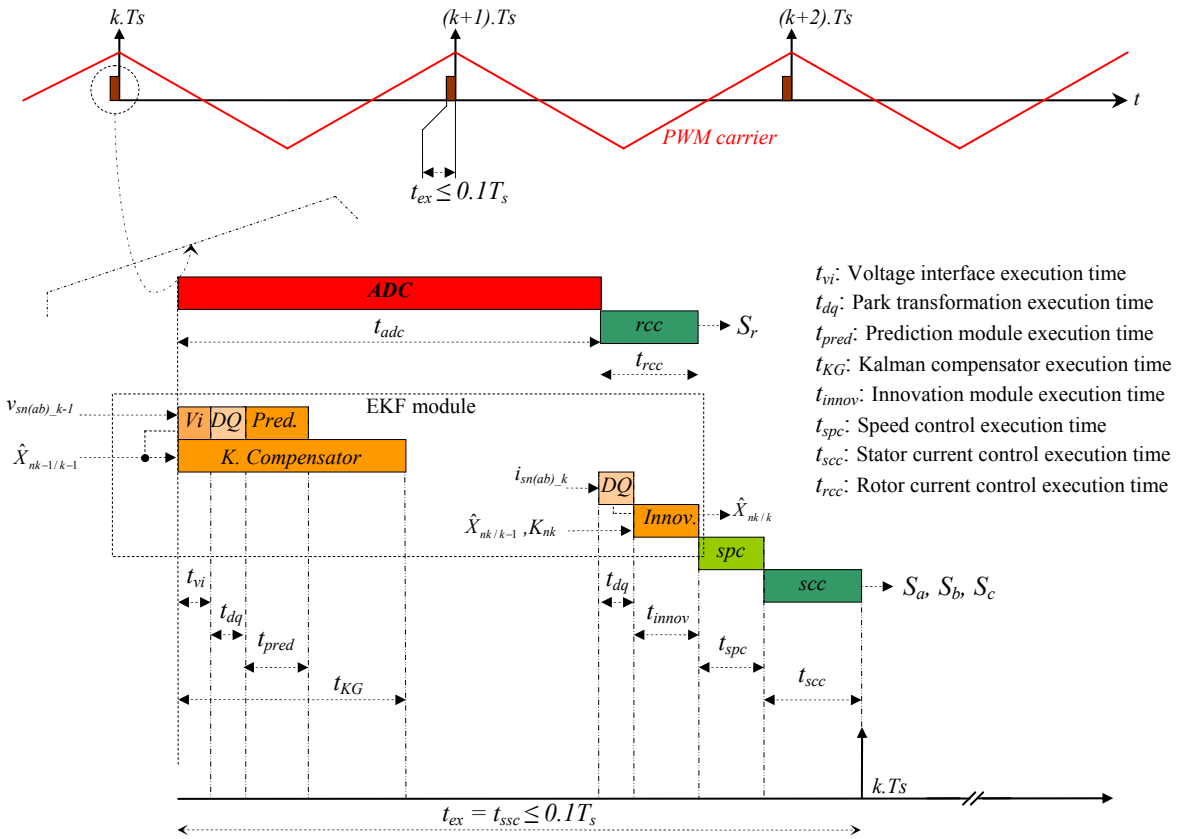


Figure 6.2: Timing diagram of the developed sensorless speed controller

$$\begin{cases} t_{ex} = t' + t_{dq} + t_{innov} + t_{spc} + t_{scc} \\ t' = t_{adc} \quad (if \quad t_{adc} > t_{KG}) \\ t' = t_{KG} \quad (if \quad t_{adc} < t_{KG}) \end{cases} \tag{6.1}$$

$$\begin{cases} t_{ex} = t' + (N_{dq} + N_{innov} + N_{spc} + N_{scc}) \cdot T_{clk} = t' + 65 \cdot T_{clk} \\ t_{adc} = 2.4\mu s; \quad t_{KG} = N_{KG} \cdot T_{clk}; \quad N_{KG} = 30; \quad N_{innov} = 7; \quad N_{dq} = 4; \quad N_{spc} = 6; \quad N_{scc} = 18 \end{cases} \tag{6.2}$$

Table 6.1: Estimation of the FPGA resources when the algorithm parallelism is fully preserved – case of the non-optimized EKF version.

Modules	Operators	Number	Corresponding FPGA hardware resources
EKF Module	**22-bit multiplier**	**352**	**1408** equivalent 18-bit multipliers (DSP blocks)
	22-bit adder	**271**	**7181** 6-bit LUTs
	22-bit subtractor	**24**	**6456** 6-bit LUTs
	22-bit register	**1932**	**42504** Flip-Flops
	22-bit inverter	**1**	**1303** 6-bit LUTs **and 3426** Flop-flops
Speed controller	**17-bit multiplier**	**3**	**3** equivalent 18-bit multipliers (DSP blocks)
	17-bit adder	**2**	**38** 6-bit LUTs
	17-bit subtractor	**2**	**38** 6-bit LUTs
	17-bit register	**21**	**357** Flop-flops
Stator & rotor current controllers	**20-bit multiplier**	**16**	**16** equivalent 18-bit multipliers (DSP blocks) + **164** 6-bit LUTs
	13-bit multiplier	**4**	**4** equivalent 18-bit multipliers (DSP blocks)
	20-bit adder	**13**	**286** 6-bit LUTs
	13-bit adder	**6**	**90** 6-bit LUTs
	20-bit subtractor	**2**	**44** 6-bit LUTs
	14-bit subtractor	**1**	**16** 6-bit LUTs
	13-bit subtractor	**1**	**15** 6-bit LUTs
	20-bit register	**93**	**1860** Flop-flops
	14-bit register	**3**	**42** Flop-flops
	13-bit register	**33**	**429** Flop-flops
Total			**- 1431** equivalent 18-bit multipliers (DSP blocks) **- 15631** 6-bit LUTs **- 46944** Flop-flops **100% of the 180 DSP blocks** (Sp6 xc6slx150) **704% of 6-bit LUTs** (Sp6 xc6slx150)

2- This second step aims to cope with the area mismatch and design an FPGA architecture that needs fewer resources. To reach this objective, the A^3 methodology can be adopted (chapter 2, §6.3.2). When applied to each module, this methodology consists in finding the potential parallelism and then making the necessary factorization. This concerns mainly the greediest operators such as multipliers. However, factorizing an architecture leads to serialize the treatment. The higher is the level of factorization, the longer is the execution time.

For example, when factorizing the EKF compensator, the corresponding execution time t_{KG} will increase and then can exceed t_{ADC} (Relation (6.1)). Because of its complexity, this module has then a significant timing criticality and can lead to the non respect of the timing constraint. Consequently, a compromise is to be found so as to apply the appropriate factorization that keeps the timing constraint satisfied.

In sections 2.2, 2.3 and 2.4, the application of the A^3 methodology to the EKF module is presented. The case of the prediction module (section 2.2), the EKF compensator (section 2.3) and the innovation module (section 2.4) are focused on. For each case, the Data Flow Graph (DFG), the factorization procedure and the Factorized DFG are discussed.

3- Once the appropriate factorization level and the FDFG of each module are defined, the third task is the design of the corresponding FPGA architecture. This design has to respect the hierarchy levels according to the modularity constraint. This is followed by the architecture VHDL-coding and the architecture functional validation. For the developed sensorless controller, these points are respectively discussed in parts 3, 4 and 5.

4- Once the global FPGA architecture is functionally validated, the next step is the analysis of the time/area performances. This is obtained after having synthesized the developed design (using the dedicated synthesis tools). This synthesis indicates the consumed FPGA resources and the maximum frequency of the operating clock. This maximum frequency allows the calculation of the global execution time. For the developed FPGA-based sensorless controller, the analysis of the time/area performances is made in part 6.

2.2. Optimization of the EKF prediction module

As a reminder, the prediction module calculates the non-compensated state space vector using the salient synchronous machine model. Relations (6.3-6.6) represent the implemented discrete-time normalized equations and Figure 6.3 highlights the corresponding DFG.

$$\hat{i}_{sndk/k-1} = G_{1isd} \cdot \hat{i}_{sndk-1/k-1} + G_{2isd} \cdot \hat{i}_{snqk-1/k-1} \cdot \hat{\omega}_{nk-1/k-1} + G_{3isd} \cdot v_{sndk-1} \tag{6.3}$$

$$\hat{i}_{snqk/k-1} = G_{1isq} \cdot \hat{i}_{snqk-1/k-1} + G_{2isq} \cdot \hat{i}_{sndk-1/k-1} \cdot \hat{\omega}_{nk-1/k-1} + G_{3isq} \cdot \hat{\omega}_{nk-1/k-1} + G_{4isq} \cdot v_{snqk-1} \tag{6.4}$$

$$\hat{\omega}_{nk/k-1} = \hat{\omega}_{nk-1/k-1} \tag{6.5}$$

$$\hat{\theta}_{nk/k-1} = G_{\theta} \cdot \hat{\omega}_{nk-1/k-1} + \hat{\theta}_{nk-1/k-1} \tag{6.6}$$

Where,

$$G_{1isd} = 1 - T_s \frac{R_s}{L_{sd}} \quad ; \quad G_{2isd} = T_s \frac{L_{sq}}{L_{sd}} \omega_B \quad ; \quad G_{3isd} = T_s \frac{V_{Bekf}}{I_B L_{sd}}$$

$$G_{1isq} = 1 - T_s \frac{R_s}{L_{sq}} \quad ; \quad G_{2isq} = -T_s \frac{L_{sd}}{L_{sq}} \omega_B \quad ; \quad G_{3isq} = -T_s \frac{M_{sr} I_{rd}}{L_{sq}} \omega_B \quad ; \quad G_{4isd} = T_s \frac{V_{Bekf}}{I_B L_{sq}}$$

$$G_{\theta} = T_s \frac{\omega_B}{\theta_B}$$

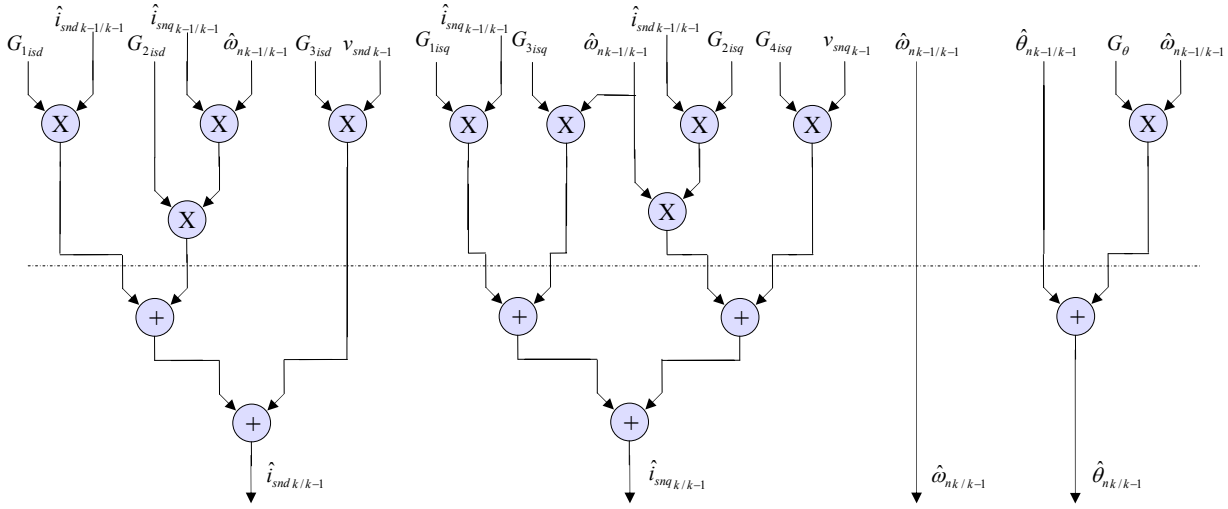


Figure 6.3: DFG of EKF prediction module

From this associated DFG, it can be seen that the equations are processed independently. For each equation (except the speed equation), the treatment can be divided in two stages. The first stage is dedicated to multiplications and the second one is dedicated to additions. Also, the DFG indicates that there are multiplications that can be performed in parallel. Thus they can be factorized and processed using only one multiplier. With an iterative structure, this multiplier can also perform the multiplication which is not made in parallel to the others (case of the i_{sd} and i_{sq} equations). Figure 6.4 shows the FDFG that represents the factorized i_{sq} equation. This graphic introduces specific nodes called, F ("Fork"), J ("Join") and I ("Iterate") that are used to delimit the factorization borders.

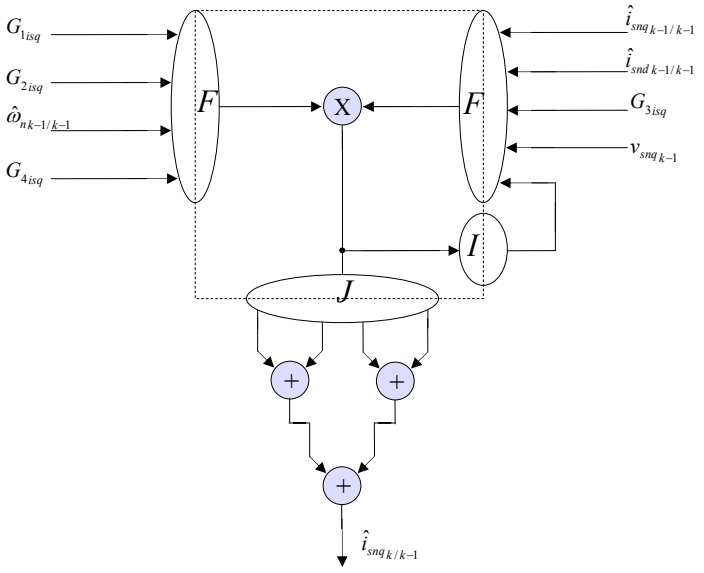


Figure 6.4: FDFG of i_{sq} equation

When it comes to the whole prediction module, it can be seen from the defined timing diagram (Figure 6.2), that the corresponding treatment can be fully serialized without affecting the execution time of the whole sensorless controller. This means that the prediction equations can be gathered and then processed by the same factorized architecture. The advantage is the use of a minimum of operators without downgrading the timing performances. Figure 6.5 presents the corresponding FDFG.

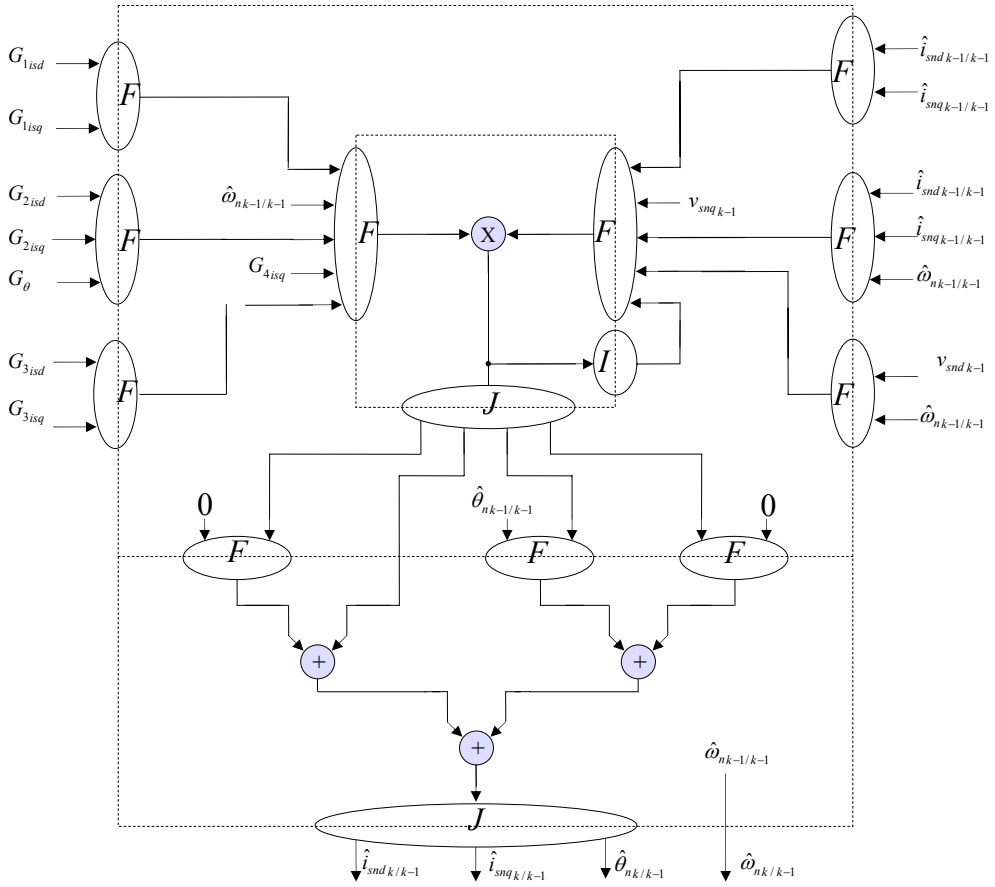


Figure 6.5: FDFG of EKF prediction module

2.3. Optimization of the EKF compensator

The EKF compensator is definitely the most constraining module in terms of complexity as well as the FPGA resources. In order to satisfy the low cost FPGA constraint, the factorization of the corresponding architecture is mandatory. Because this module presents a high timing criticality, this factorization can lead to the non respect of the timing constraint. A compromise is then necessary so as to balance the level of factorization with the corresponding execution time. In the following, it has been decided to present the factorization of the initial Kalman compensator (non-optimized version) and also the optimized version that is based on the matrix symmetrization (chapter 5 §5.2.2).

2.3.1. Jacobian matrices

Relation (6.7) and (6.8) present the processed Jacobian matrices for the model linearization.

$$F_{dnk} = \left.\frac{\partial(f_{dn})}{\partial x_n}\right|_{x=\hat{x}_{nk-1/k-1}} = \begin{bmatrix} G_{1isd} & G_{2isd}\cdot\hat{\omega}_{nk-1/k-1} & G_{2isd}\cdot\hat{i}_{snq\,k-1/k-1} & 0 \\ G_{2isq}\cdot\hat{\omega}_{nk-1/k-1} & G_{1isq} & G_{2isq}\cdot\hat{i}_{snd\,k-1/k-1}+G_{3isq} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & G_\theta & 1 \end{bmatrix} \quad (6.7)$$

$$H_{dnk} = \left.\frac{\partial(h_{dn})}{\partial x_n}\right|_{x=\hat{x}_{nk-1/k-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (6.8)$$

Figure 6.6 shows the DFG for F_{dnk}. It can be noticed that the time-varying matrix elements are processed in parallel. Thus the multiplication can be factorized and processed by only one multiplier. The obtained FDFG is then depicted in Figure 6.7.

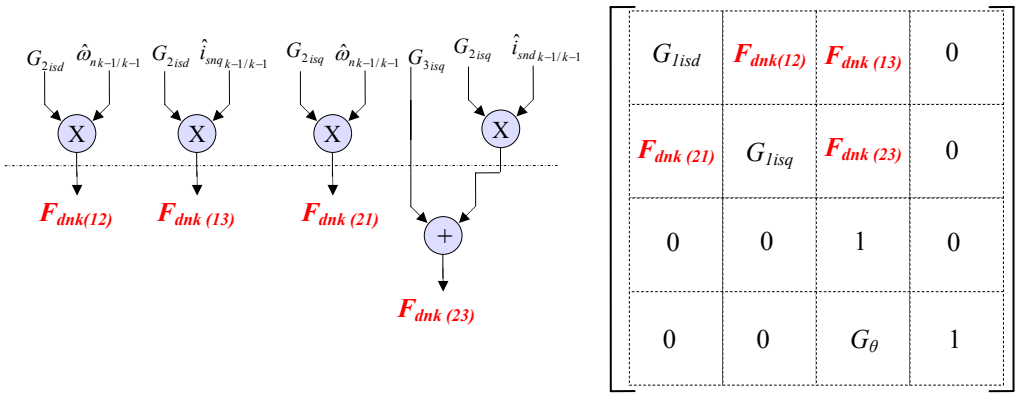


Figure 6.6: DFG for Jacobian matrix F_{dnk}

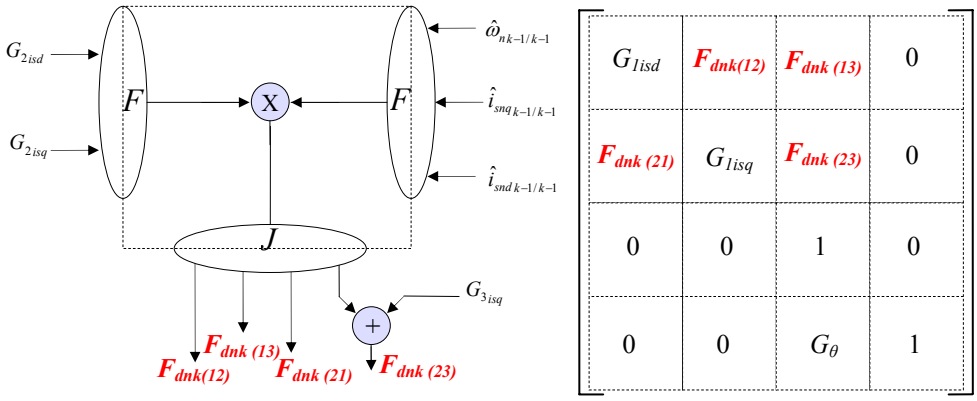


Figure 6.7: FDFG for Jacobian matrix F_{dnk}

2.3.2. Kalman gain and covariance matrix

The non-optimized computation of K_{nk} and $P_{nk/k}$ (respectively the Kalman gain and the covariance matrix of the estimation error, chapter 4, relations (4.6-4.8)) is based on matrix treatment including matrix multiplications, additions, subtractions, transpose and a matrix inversion. For figure-view simplicity, the corresponding DFG has been extracted from the Simulink block diagram (Figure 6.8).

The factorization of this module has been done according to the following settlement. The whole matrix multiplications have been factorized using only one 3-matrix multiplier (AxBxC). Because the matrix dimensions are not the same everywhere, it has been decided to make uniform the matrix multiplications. Consequently, the matrix dimensions have been extended (when necessary) to 4x4. At the end of each multiplication, the result is re-adapted (when necessary) to the initial dimension. Figure 6.9 overviews the developed FDFG.

In the proposed FPGA architecture, the 3-matrix multiplier is also factorized. To process the 128 multiplications, only four multipliers have been used. As for the matrix inversion, the needed 6 multiplications are processed with one multiplier. To anticipate the FPGA architecture design, the implementation of the inverter (matrix determinant inversion) is made with the help of a dedicated Xilinx IP. In [Appendix E], an in-depth presentation of the developed 3-matrix multiplier and the developed matrix inverter is made.

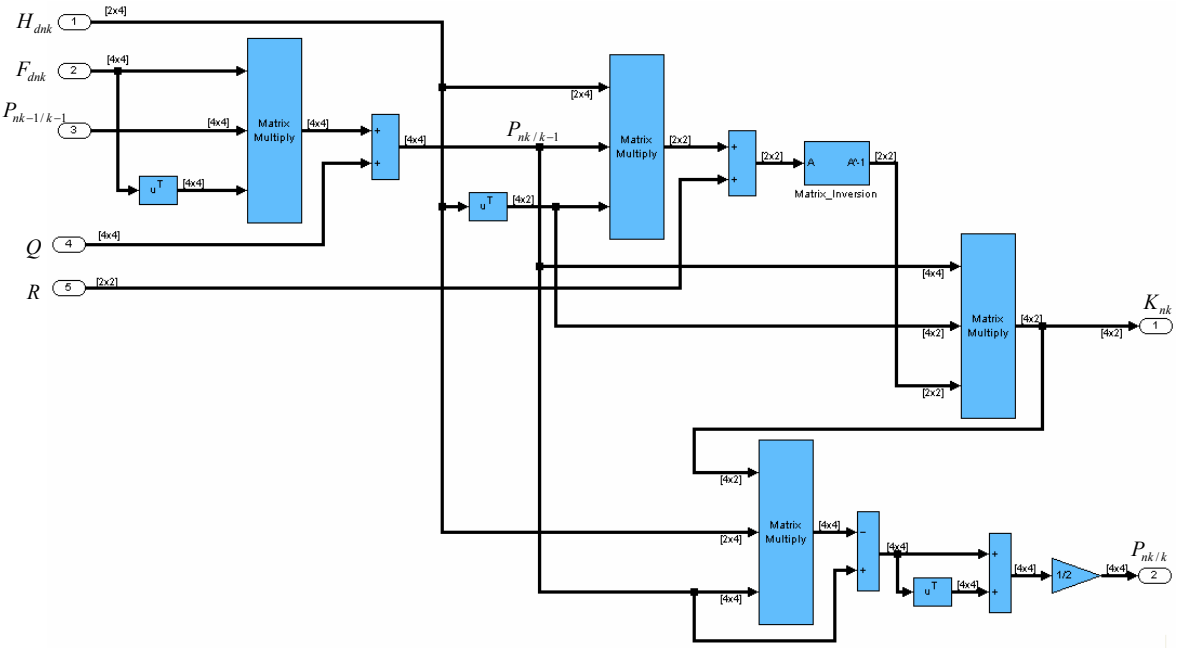


Figure 6.8: DFG for non-optimized EKF compensator

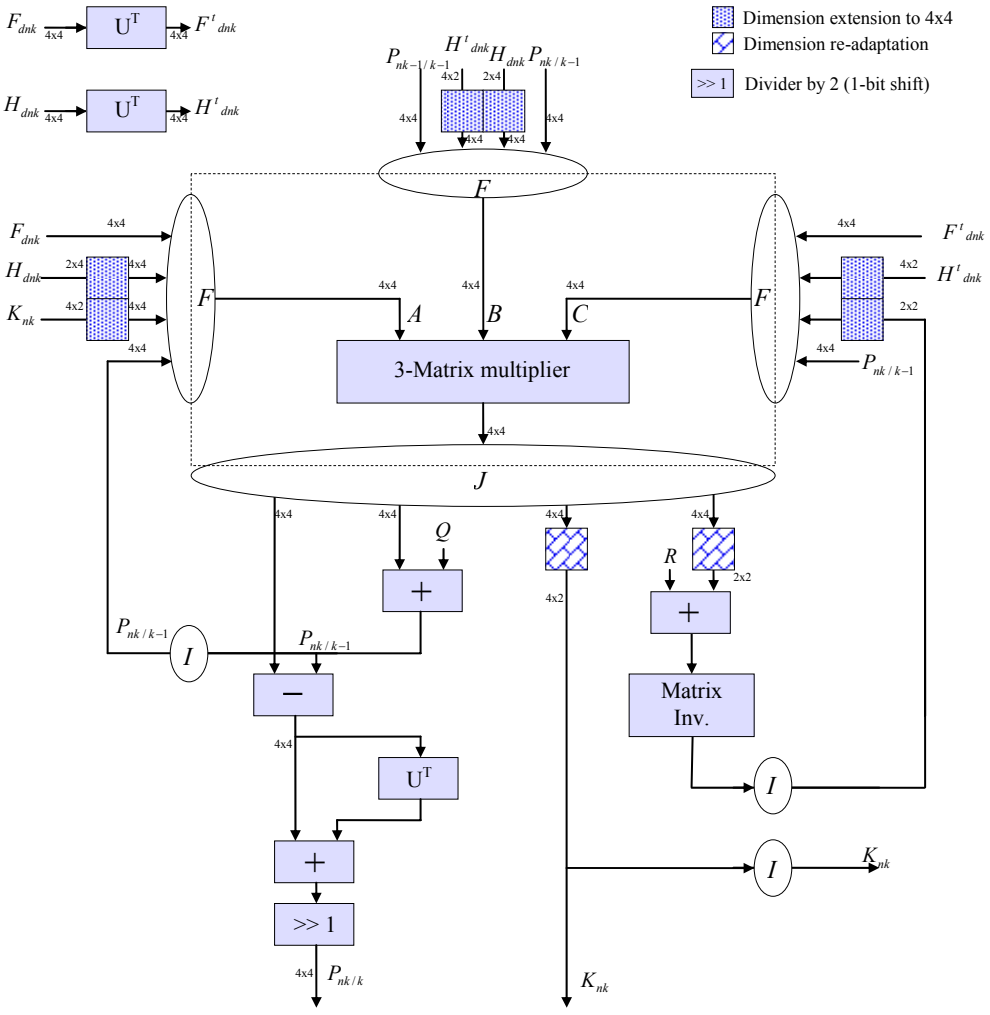


Figure 6.9: FDFG for non-optimized EKF compensator

As far as the optimized version of the EKF compensator is concerned (Matrix symmetrization version), the treatment is mainly based on scalar operations (except the matrix inversion). Figure 6.10 presents the DFG that concerns the calculation of the covariance matrix (extracted from Simulink).

When looking closely to the processed equations, it can be seen that the calculation of the matrix elements can be made using only a dot-product (see the example in chapter 2 §6.3.1). It is then possible to factorize the whole treatment around this dot-product. Figure 6.11 shows the corresponding FDFG.

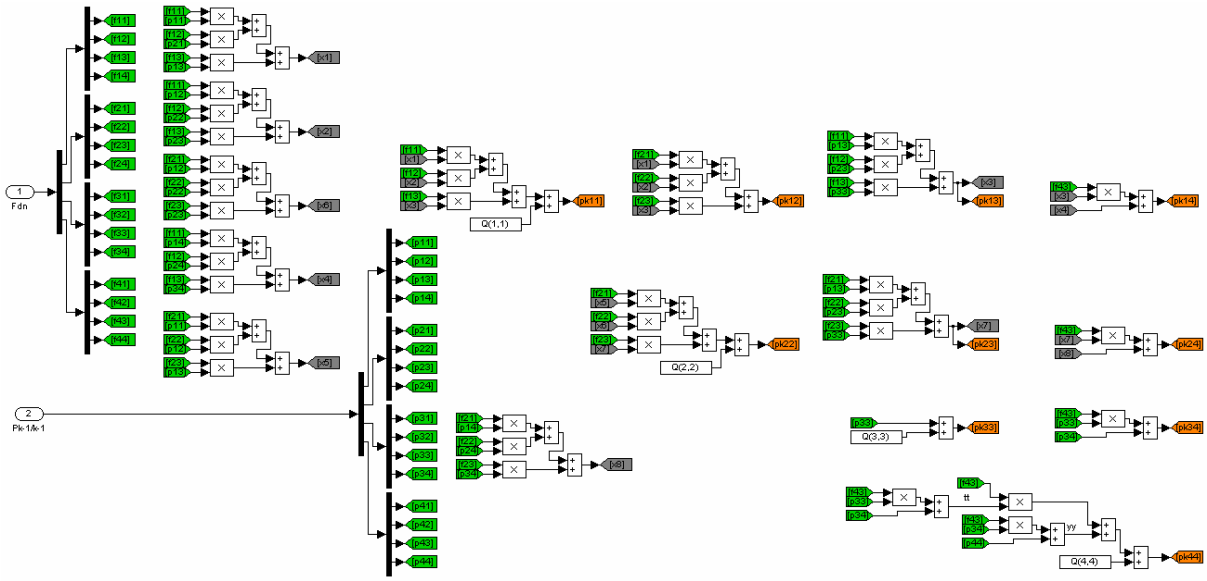


Figure 6.10: DFG for MS-optimized EKF compensator

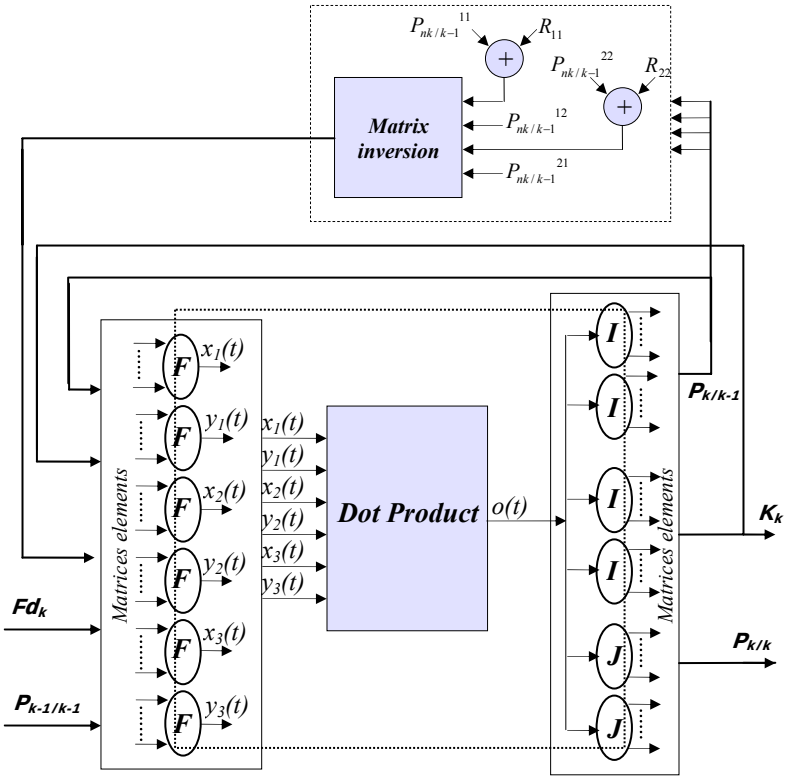


Figure 6.11: FDFG for MS-optimized EKF compensator

2.4. Optimization of the EKF innovation module

The innovation module is launched ones the EKF compensator has generated the Kalman gain. Relations (6.9-6.12) present the processed equations and Figure 6.12 highlights the corresponding DFG. Here again the processed parallel multiplications can be factorized and only one multiplier can be used. Figure 6.13 presents the obtained FDFG.

$$\hat{i}_{sndk/k} = \hat{i}_{sndk/k-1} + K_{11} \cdot (i_{sndk} - \hat{i}_{sndk/k-1}) + K_{12} \cdot (i_{snqk} - \hat{i}_{snqk/k-1}) \quad (6.9)$$

$$\hat{i}_{snqk/k} = \hat{i}_{snqk/k-1} + K_{21} \cdot (i_{sndk} - \hat{i}_{sndk/k-1}) + K_{22} \cdot (i_{snqk} - \hat{i}_{snqk/k-1}) \quad (6.10)$$

$$\hat{\omega}_{nk/k} = \hat{\omega}_{nk/k-1} + K_{31} \cdot (i_{sndk} - \hat{i}_{sndk/k-1}) + K_{32} \cdot (i_{snqk} - \hat{i}_{snqk/k-1}) \quad (6.11)$$

$$\hat{\theta}_{nk/k} = \hat{\theta}_{nk/k-1} + K_{41} \cdot (i_{sndk} - \hat{i}_{sndk/k-1}) + K_{42} \cdot (i_{snqk} - \hat{i}_{snqk/k-1}) \quad (6.12)$$

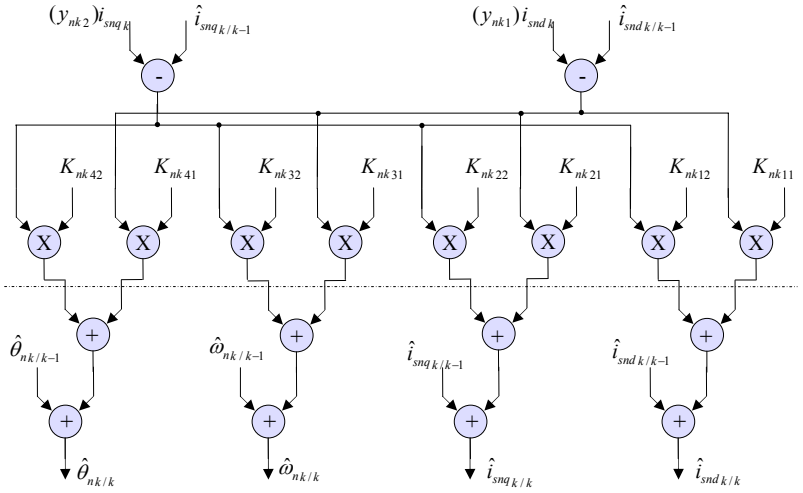


Figure 6.12: DFG of the innovation module

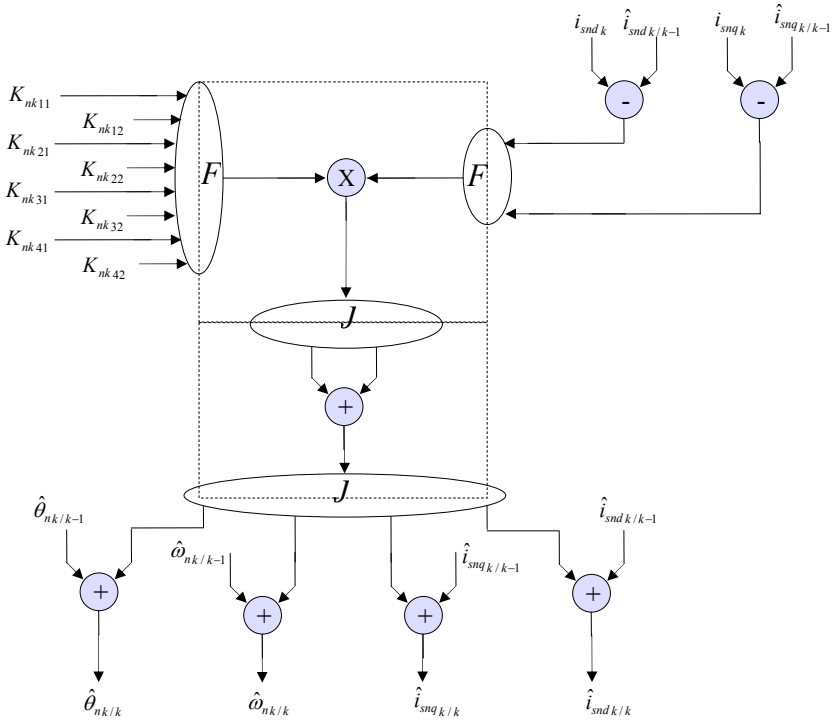


Figure 6.13: FDFG of the innovation module

3. FPGA architecture design

With regards to the modular partitioning and the level of hierarchy, the hardware FPGA architecture of each module is designed. It is composed of a data path and a control unit that are both synchronized with the global clock signal. According to the obtained FDFG, the architecture is designed by replacing the FDFG nodes (F, J and I) by there corresponding operators. Thus, the node F is replaced by a multiplexer, J and I replaced by registers.

The data path contains the used operators and data buses between them. The treatment scheduling is ensured by the control unit which is a simple Finite State Machine (FSM). The latter is activated via a Start pulse signal at each sampling period. When the computation time process is over, an End pulse signal indicates the end of the treatment.

Once the architecture of each module is developed, the global data path (4th and 5th level of hierarchy of the modular partitioning, chapter 5, Figure 5.1) is then designed by associating the needed modules. A global control unit is also implemented so as to synchronize the global treatment.

In the following, author is focusing on the architectures of the prediction module and the EKF compensator (3rd level of hierarchy, chapter 5, Figure 5.1). Figure 6.14 presents the developed FPGA architecture of the prediction module and Table 6.2 lists the signal configuration depending on which prediction equation is processed. Figure 6.15 presents the architecture of the EKF compensator, especially the module that calculates the Kalman gain and the covariance matrix.

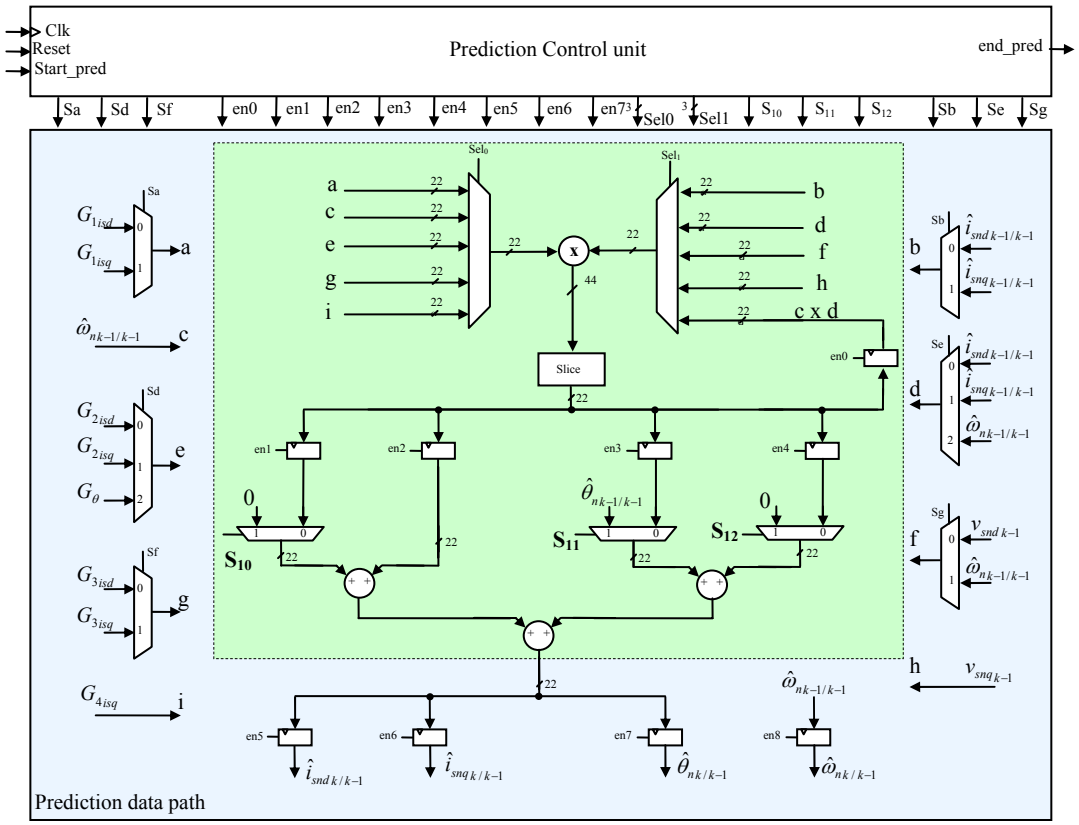


Figure 6.14: FPGA architecture of the innovation module

Table 6.2: Prediction module - Configuration of the multiplexers

	$\hat{\imath}_{sndk/k-1}$	$\hat{\imath}_{snqk/k-1}$	$\hat{w}_{nk/k-1}$	$\theta_{nk/k-1}$
a	G_{1isd}	G_{1isq}	--	--
b	$\hat{\imath}_{sndk-1/k-1}$	$\hat{\imath}_{snqk-1/k-1}$	--	--
c	$\hat{w}_{nk-1/k-1}$	$\hat{w}_{nk-1/k-1}$	--	--
d	$\hat{\imath}_{snqk-1/k-1}$	$\hat{\imath}_{sndk-1/k-1}$	--	$\hat{w}_{nk-1/k-1}$
e	G_{2isd}	G_{2isq}	--	G_{θ}
f	v_{sndk-1}	$\hat{w}_{nk-1/k-1}$	--	--
g	G_{3isd}	G_{3isq}	--	--
h	--	v_{snqk-1}	--	--
i	--	G_{4isq}	--	--
S_{10}	*0*	*0*	--	*1*
S_{11}	*0*	*0*	--	*1*
S_{12}	*0*	*0*	--	*1*

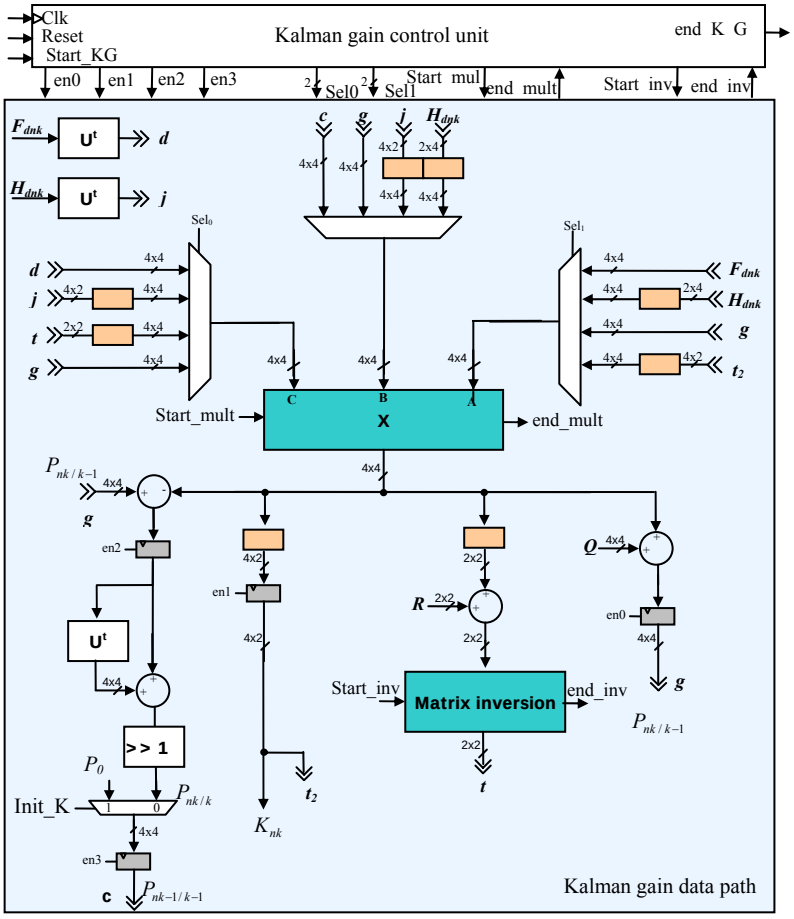


Figure 6.15: FPGA architecture of the Kalman gain and covariance matrix calculator

Figure 6.16 presents the FPGA architecture of the EKF module which is located at the 4th level of hierarchy (chapter 5, Figure 5.1). This module includes Park transformation (abc-dq) modules. The used sine pattern has been stored in a RAM memory block and the corresponding total size is 13312 Kbits. As for the final level, Figure 6.17 presents the architecture of the global sensorless speed controller.

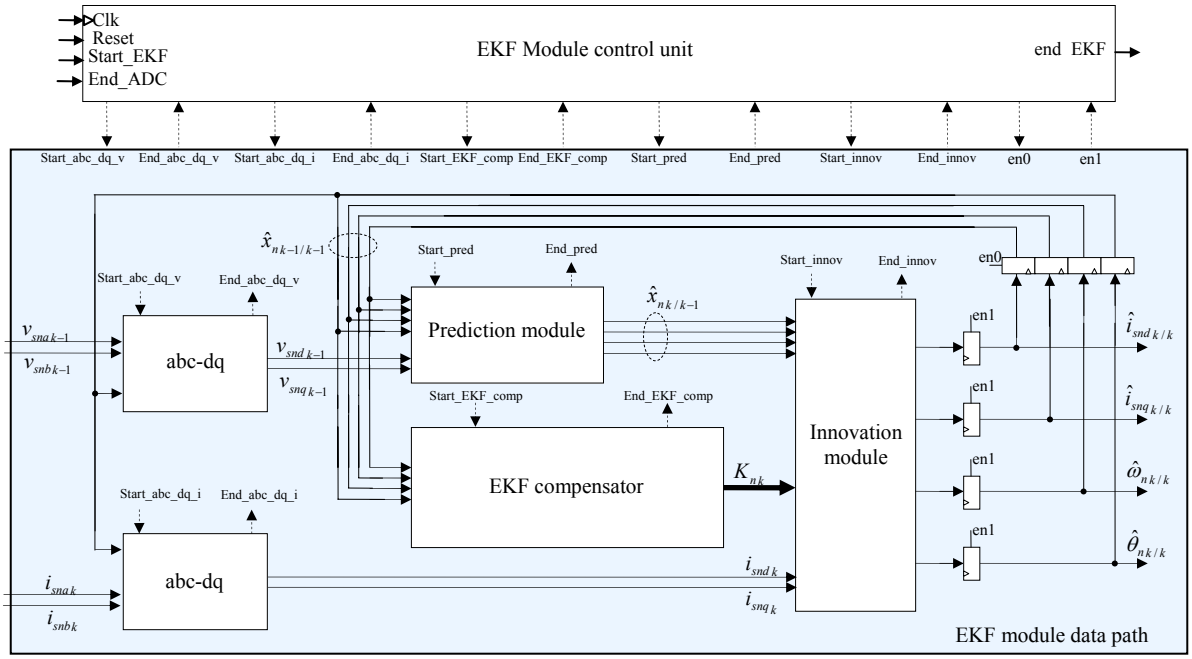


Figure 6.16: FPGA architecture of the EKF module

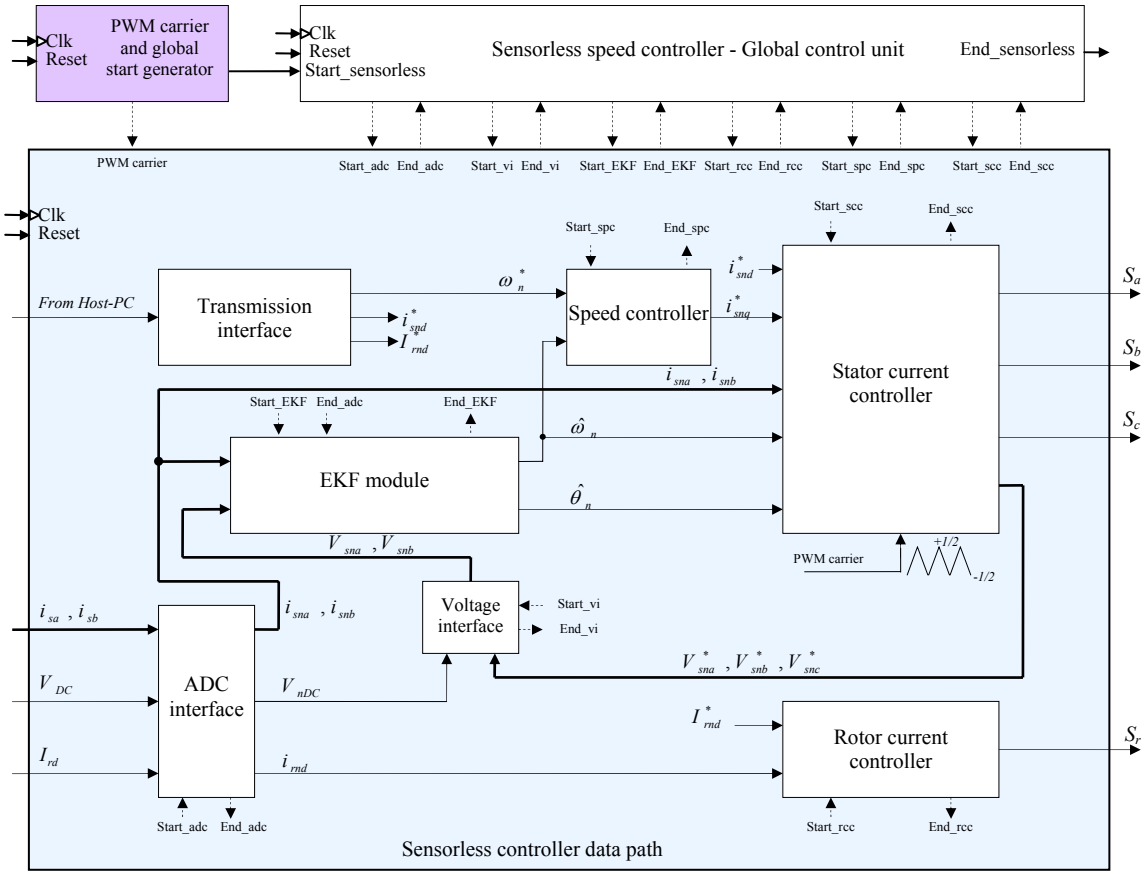


Figure 6.17: FPGA architecture of the global sensorless speed controller

4. Architecture VHDL-coding

At this stage, the designed FPGA architecture of each module is ready to be programmed. The VHDL description language is used in our case. Also, with the used Xilinx FPGA solutions, the Xilinx ISE design tool is used. In order to preserve the modularity and make the design easy to manage, hierarchical and multi-file VHDL programming approach is adopted (e.g. Figure 6.18). Each module is coded as a unique VHDL file and then used as a component at a higher level of hierarchy.

The implemented matrix inversion uses a divider. The latter is used to invert the input matrix determinant. To this aim, the Xilinx Pipelined Divider IP has been implemented. An overview of this IP has been made in [Appendix E].

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Module_i is
    Port ( start_module_i  : in  STD_LOGIC;
           input           : in  STD_LOGIC;
           output          : out  STD_LOGIC;
           end_module_i    : out  STD_LOGIC;
           clk             : in  STD_LOGIC;
           reset           : in  STD_LOGIC);
end Module_i;

architecture Behavioral of Module_i is

      component Module_i_Data_Path
          Port ( input     : in  STD_LOGIC;
                 output    : out  STD_LOGIC;
                 eni       : in  STD_LOGIC;
                 clk       : in  STD_LOGIC;
                 reset     : in  STD_LOGIC);
      end component;

      component Module_i_CTRL
          Port ( start  : in  STD_LOGIC;
                 end_m  : out  STD_LOGIC;
                 clk    : in  STD_LOGIC;
                 reset  : in  STD_LOGIC);
      end component;

signal en : std_logic;
begin
      Inst_Module_i_Data_Path: Module_i_Data_Path
         PORT MAP(input => input,output => output,eni => eni,clk => clk,reset => reset );

      Inst_Module_i_CTRL: Module_i_CTRL
         PORT MAP(start => start_module_i, end_m => end_module_i, clk => clk,reset => reset  );
end Behavioral;
```

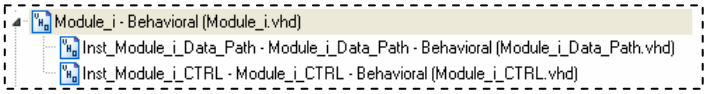


Figure 6.18: Architecture VHDL-coding – Hierarchical Multi-file approach

5. Architecture functional simulation

The developed VHDL design has been functionally simulated using dedicated tools such as the well-known ModelSim. The test has been done by applying Testbench waveforms to the inputs of the developed sensorless speed controller. These waveforms have been extracted from their correspondent Matlab/Simulink counterparts.

Figure 6.19 show the ModelSim results obtained after having tested the EKF module. Sinusoidal voltage and current waveforms have been applied to the inputs. The estimated electrical rotor position and speed are displayed.

In order to confirm the good functionality of the EKF module, the same simulation process has been made in Matlab/Simulink environment with the same input waveforms. The obtained simulation results are then compared to those obtained with ModelSim. Figure 6.20 shows the waveforms of the electrical rotor position and speed in both cases.

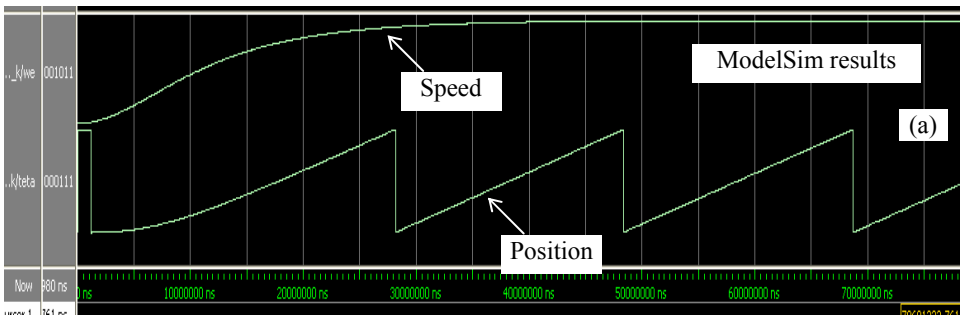


Figure 6.19: VHDL design functional validation – ModelSim results

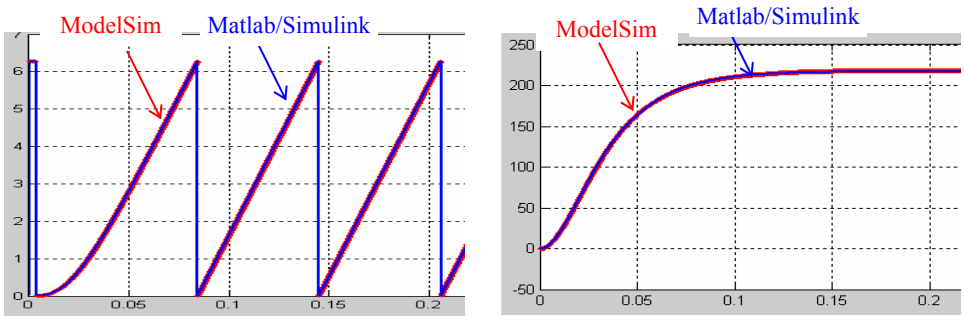


Figure 6.20: VHDL design functional validation – Comparison between ModelSim and Matlab/Simulink results

6. Design synthesis and time/area performances analysis

In the case of the developed FPGA-based sensorless controller, Tables 6.3, 6.4 and 6.5 summarize the obtained synthesis results in the case of the non-optimized and the optimized EKF versions. They highlight the estimated maximum frequency of the operating clock and show the global hardware resource consumption. All these data are listed for different FPGA device solutions including low cost and high performance FPGAs.

Table 6.3: Synthesis results for the initial sensorless speed controller (non-optimized EKF)

	Low cost FPGA			**High performance FPGA**	
	Spartan 3 xc3s5000	**Spartan 3E** xc3s1600E	**Spartan 6** xc6slx150	**Virtex 2P** xc2vp30	**Virtex 6** Xc6vsx475
Max. clk Frequency	**67 MHz**	**44 MHz**	**82 MHz**	120 MHz	226 MHz
Global resources use	- **20 %** (8320 CLB) - **66%** hw 18-bit multipliers (over 104) - **1** 18-Kbit RAM block	- **63%** (3688 CLB) - **100%** hw 18-bit multipliers (over 36) - **1** 18-Kbit RAM block	- **11%** (11519 CLB) - **38 %** hw 18-bit DSP blocks (over 180) - **1** 18-Kbit RAM block	- **49%** (3424 CLB) - **50%** hw 18-bit multipliers (over 136) - **1** 18-Kbit RAM block	- **3.4%** (37200 CLB) - **3.4%** hw 18-bit DSP blocks (over 2016) - **1** 18-Kbit RAM block

Table 6.4: Synthesis results for the optimized sensorless speed controller (k(∞) EKF)

	Low cost FPGA			High performance FPGA	
	Spartan 3 xc3s5000	**Spartan 3E** xc3s1600E	**Spartan 6** xc6slx150	**Virtex 2P** xc2vp30	**Virtex 6** Xc6vsx475
Max. clk Frequency	**67 MHz**	**65 MHz**	**97 MHz**	120 MHz	252 MHz
Global resources use	**-10 %** (8320 CLB) **-25%** hw 18-bit multipliers (over 104) - **1** 18-Kbit RAM block	**-21%** (3688 CLB) **-80%** hw 18-bit multipliers (over 36) - **1** 18-Kbit RAM block	**-3.4%** (11519 CLB) **-12 %** hw 18-bit DSP blocks (over 180) - **1** 18-Kbit RAM block	-25% (3424 CLB) -19% hw 18-bit multipliers (over 136) - 1 18-Kbit RAM block	-1.4% (37200 CLB) -1.24% hw 18-bit DSP blocks (over 2016) - 1 18-Kbit RAM block

Table 6.5: Synthesis results for the optimized sensorless speed controller (MS-EKF)

	Low cost FPGA			High performance FPGA	
	Spartan 3 xc3s5000	**Spartan 3E** xc3s1600E	**Spartan 6** xc6slx150	**Virtex 2P** xc2vp30	**Virtex 6** Xc6vsx475
Max. clk Frequency	**120 MHz**	**54 MHz**	**97 MHz**	167 MHz	253 MHz
Global resources use	**-16 %** (8320 CLB) **-50%** hw 18-bit multipliers (over 104) - **1** 18-Kbit RAM block	**-45%** (3688 CLB) **-100%** hw 18-bit multipliers (over 36) - **1** 18-Kbit RAM block	**-5.8%** (11519 CLB) **-26 %** hw 18-bit DSP blocks (over 180) - **1** 18-Kbit RAM block	-39% (3424 CLB) -38% hw 18-bit multipliers (over 136) - 1 18-Kbit RAM block	-2.5% (37200 CLB) -2.5% hw 18-bit DSP blocks (over 2016) - 1 18-Kbit RAM block

From these synthesis results, it can be seen that with the fixed factorization level, both sensorless controller versions (Optimized and non-optimized EKF versions) can be implemented in low-cost FPGA devices. According to the optimization strategy, the area constraint is consequently satisfied.

As for the timing constraint, Tables 6.6, 6.7 and 6.8 list the minimum execution time depending on the clock frequency and for each of the chosen FPGA devices. This execution time has been calculated using relation (6.1). The conversion time of the used ADC devices is equal to 2.4µs. The obtained results indicate that both of the chosen FPGA solutions and the fixed factorization level lead to short execution times which all satisfy the timing constraint (maximum execution time limit). It is also important to notice that in some cases (typically table 6.7), the obtained execution time depends strongly on the ADC conversion time. The use of high speed ADC devices is then necessary in order to enhance, if needed, the obtained timing performances.

Table 6.6: Minimum execution time (µs) for the initial sensorless speed controller (non-optimized EKF)

	Low cost FPGA			High performance FPGA	
	Spartan 3 xc3s5000	**Spartan 3E** xc3s1600E	**Spartan 6** xc6slx150	**Virtex 2P** xc2vp30	**Virtex 6** Xc6vsx475
Maximum clock Frequency (MHz)	**67**	**44**	**82**	120	226
t_{ADC}	2.4				
t_{KG} (latency :230)	3.43	5.23	2.8	1.91	1.02
t_{dq} (latency :9)	0.134	0.204	0.11	0.075	0.04
t_{innov} (latency :18)	0.269	0.41	0.22	0.15	0.08
t_{spc} (latency :9)	0.134	0.204	0.11	0.075	0.04
t_{scc} (latency :43)	0.642	0.977	0.524	0.358	0.19
$\boldsymbol{t_{ex_min}}$	**4,612**	**7,023**	**3,768**	**3,058**	**2,75**

Table 6.7: Minimum execution time (µs) for the optimized sensorless speed controller (k(∞) EKF)

	Low cost FPGA			High performance FPGA	
	Spartan 3 xc3s5000	**Spartan 3E** xc3s1600E	**Spartan 6** xc6slx150	**Virtex 2P** xc2vp30	**Virtex 6** Xc6vsx475
Maximum clock Frequency (MHz)	**67**	**65**	**97**	120	252
t_{ADC}	2.4				
t_{KG} (latency :0)	0	0	0	0	0
t_{dq} (latency :9)	0,134	0,138	0,093	0,075	0,036
t_{innov} (latency :18)	0,269	0,277	0,186	0,15	0,071
t_{spc} (latency :9)	0,134	0,138	0,093	0,075	0,036
t_{scc} (latency :43)	0,642	0,662	0,443	0,358	0,171
$\boldsymbol{t_{ex_min}}$	**3,579**	**3,615**	**3,214**	**3,058**	**2,713**

Table 6.8: Minimum execution time (µs) for the optimized sensorless speed controller (MS EKF)

	Low cost FPGA			High performance FPGA	
	Spartan 3 xc3s5000	**Spartan 3E** xc3s1600E	**Spartan 6** xc6slx150	**Virtex 2P** xc2vp30	**Virtex 6** Xc6vsx475
Maximum clock Frequency (MHz)	**120**	**54**	**97**	167	253
t_{ADC}	2.4				
t_{KG} (latency :198)	2.955	4.5	2.414	1.65	0.876
t_{dq} (latency :9)	0,075	0,167	0,093	0,054	0,036
t_{innov} (latency :18)	0,15	0,333	0,186	0,108	0,071
t_{spc} (latency :9)	0,075	0,167	0,093	0,054	0,036
t_{scc} (latency :43)	0,358	0,796	0,443	0,257	0,17
$\boldsymbol{t_{ex_min}}$	**3,058**	**5,130**	**3,214**	**2,873**	**2,712**

7. Conclusion

This chapter was dedicated to the development of the FPGA architecture of the EKF-based sensorless speed controller. This development has been done according to the design methodology which includes the proposed optimization procedure.

As a summary, at the beginning of the FPGA architecture development, a first estimation of the time/area performances has been made. This estimation concerned the whole sensorless speed controller where the parallelism has been fully preserved. Then, it has been observed that the corresponding architecture is too heavy to be implemented in a low cost FPGA solution. To reduce the needed FPGA resources, the A^3 methodology has been applied so as to factorize each of the developed modules. Although this assumption leads to serialize the treatment, the final FPGA architecture has respected both of the timing and area constraints. Thus, the architecture is ready for the FPGA implementation process (chapter 2 §6.3.4).

Chapter 7

FPGA-based sensorless control for synchronous AC drive - Experimentation

1. Introduction

This chapter discusses the final step of the FPGA-based sensorless speed controller design process: the experimentation. The latter has been divided into two steps, the Hardware-In-the-Loop (HIL) validation and the ultimate experimental validation.

The HIL gives the possibility to make an intermediate validation between a full software development (Matlab/Simulink, Xilinx ISE, ModelSim …) and a full experimental validation using the experimental platform. The advantage is to guarantee the functionality of the implemented FPGA-based controller when applied to the actual system.

This chapter is organized as follows. First of all, the features of the experimental platform are presented. This section is complementary to the system specification made in chapter 4.

The HIL validation is then discussed starting by recalling the principle synoptic and by describing the implemented emulation modules. This is followed by the presentation of the obtained results. It can be observed that these results are organized in the same way as the Matlab/Simulink simulation results. We start by presenting the validation of the stator current controller, then the speed controller, then the EKF estimation and finally the validation of the whole sensorless controller.

As far as the experimental validation is focused on, the proposed experimental results are once again organized as previously discussed. Additional tests have been, anyhow, proposed and deal with the evaluation of the robustness of the sensorless speed controller against load torque variation. Furthermore, an expansion to a larger operating speed range is achieved and the corresponding results are presented.

2. Overview of the experimental platform

Figure 7.1 presents the power stage of the implemented experimental platform. According to the hardware system specification (chapter 3), the features of this platform are listed below.

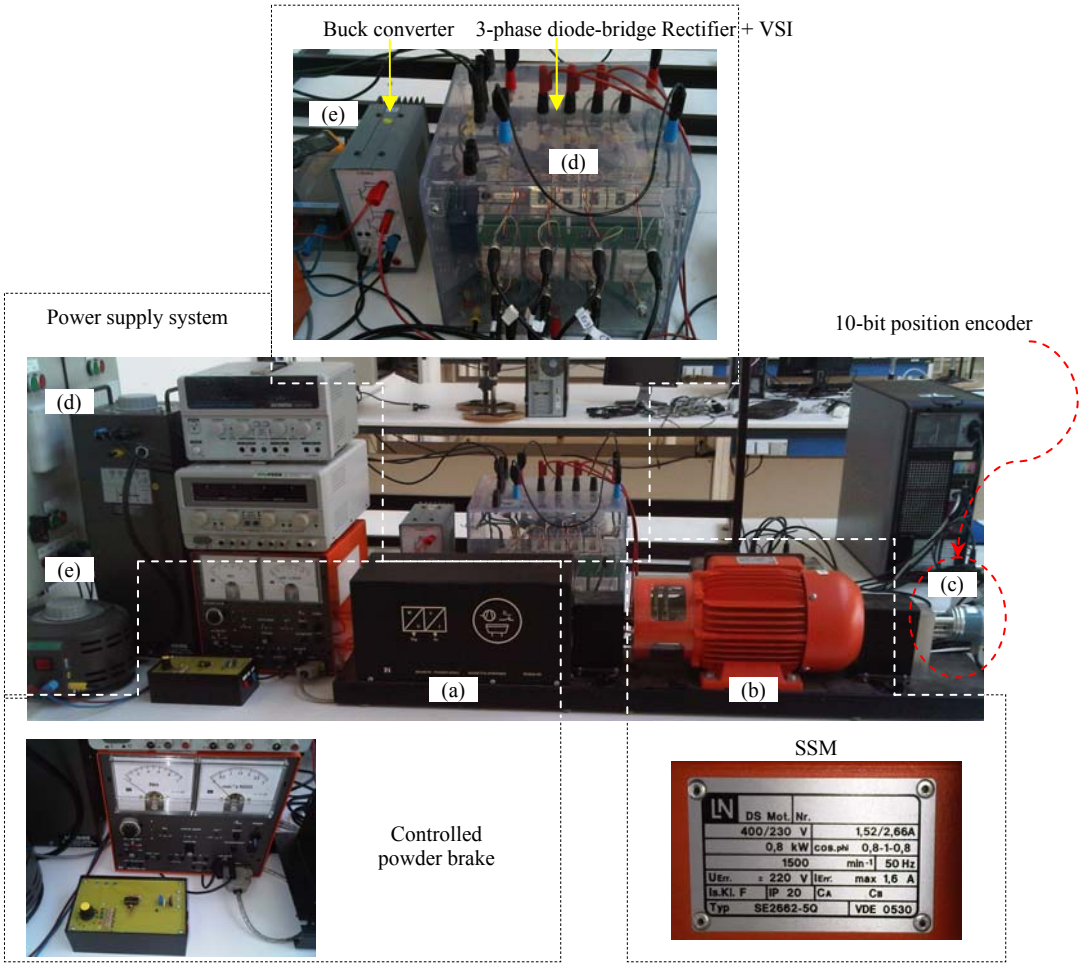


Figure 7.1: Experimental platform – Power stage

Mechanical load (Figure 7.1(a)):

- Controlled powder brake: Maximum torque:25Nm, Maximum speed: 6000 rpm

Salient Synchronous Machine (Figure 7.1(b)):

- 230/400V, 1.52/2.66A, 4-poles, 0.8kW, 1500 rpm, 5Nm

Position sensor (Figure 7.1(c)): Used to compare estimated and measured rotor positions.

- Absolute encoder: 10-bit resolution

Stator power supply (Figure 7.1(d)):

- Autotransformer: 3-phase, 230/400V, 50Hz
- AC-DC converter: SEMIKRON 3-phase, diode-bridge rectifier, 1100µF/800V capacitors
- DC-AC converter: SEMIKRON VSI – CM50DY IGBT modules – V_{DCmax}=800V, I_{max}=30A

Rotor power supply (Figure 7.1(e)):

- Autotransformer: 1-phase, 230V, 50Hz
- AC-DC converter: 1-phase, diode-bridge rectifier, 2200µF/450V capacitor
- DC-DC converter: Buck converter – ARCEL-2RDV-22 IGBT module – V_{DCmax}=800V, I_{max}=30A

Figure 7.2 overviews the implemented voltage and current sensor board, the ADC board and the FPGA board. Their features are listed below. It can be noticed that the FPGA board is based on Xilinx XUP Virtex_2P board. The used XC2VP30 FPGA belongs to the high performance FPGA family. This may seem confusing with the analysis made in the previous chapter where the objective was the use of low cost FPGAs. However, such FPGA board was used for two reasons. The first one is related to experimental constraints where a high number of signals are wired (89 signals), mainly due to the position encoder, to the ADC and to the DAC boards. The second reason is related to the Hardware In the Loop (HIL) tests where the emulation of the physical plant is added to the developed design. Additional FPGA resources are consequently needed. Furthermore, during these HIL tests, a high number of RAM memory blocks is required for data storage.

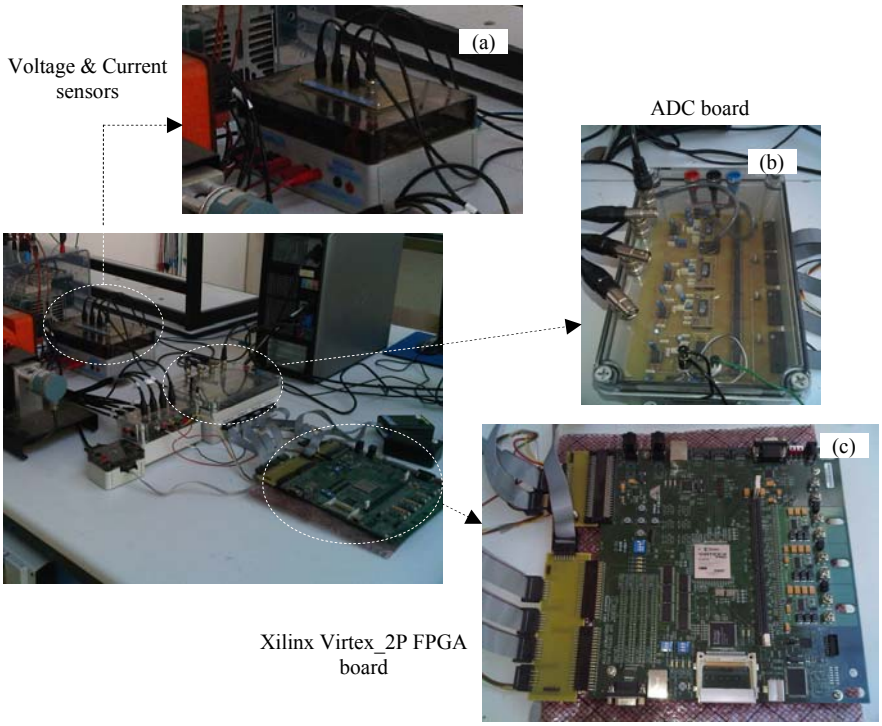


Figure 7.2: Experimental platform – Digital control unit and interface boards

Voltage and current sensor board (Figure 7.2(a)):

- Based on the ARCTU3 board: 2.5V/10A and 1V/100V

ADC board (Figure 7.2(b)):

- AD9221 ADCs
- Resolution : 12 bits
- Voltage reference V_{ref}: 1V, 2.5V
- Input voltage range : 0-2V (with V_{ref}=1V), 0-5V (with V_{ref}=2.5V)
- Maximum conversion rate: 1.5Msps
- Maximum clock frequency: 1.5MHz
- Effective conversion time : 3 clock cycles; t_{conv_min}=2µs
- Used clock frequency: 1.25MHz ➔ t_{conv}=2.4µs

Xilinx XUP Virtex_2P board (Figure 7.2(c)):

- XC2VP30 FPGA
- 6 expansion connectors (80 user I/O pins)
- 1 high speed expansion connector (40 user I/O pins)
- 32MHz, 75MHz and 100MHz clocking resources
- Up to 2Gb Synchronous Dynamic RAM
- USB/JTAG configuration port; transmission rate: 3MHz
- Dedicated peripherals (switches, push-buttons, LEDs, RS232 port, PS-2 ports, Audio Codec, SATA ports, transceiver ports …)

3. Hardware in The Loop validation

In order to verify a first experimental operating guarantee of the developed FPGA-based sensorless controller, an HIL validation has been achieved. The latter is an intermediate validation step between a fully computer-based development (Matlab/Simulink, ModelSim, FPGA design tools) and a fully experimental test (actual system platform). The developed design has to be associated with an emulation of the plant. In addition, a communication controller has to be implemented in order to transfer the stimuli and the probed data. This communication is made with a Host-PC. A comparison between the obtained HIL results and the simulation results is achieved.

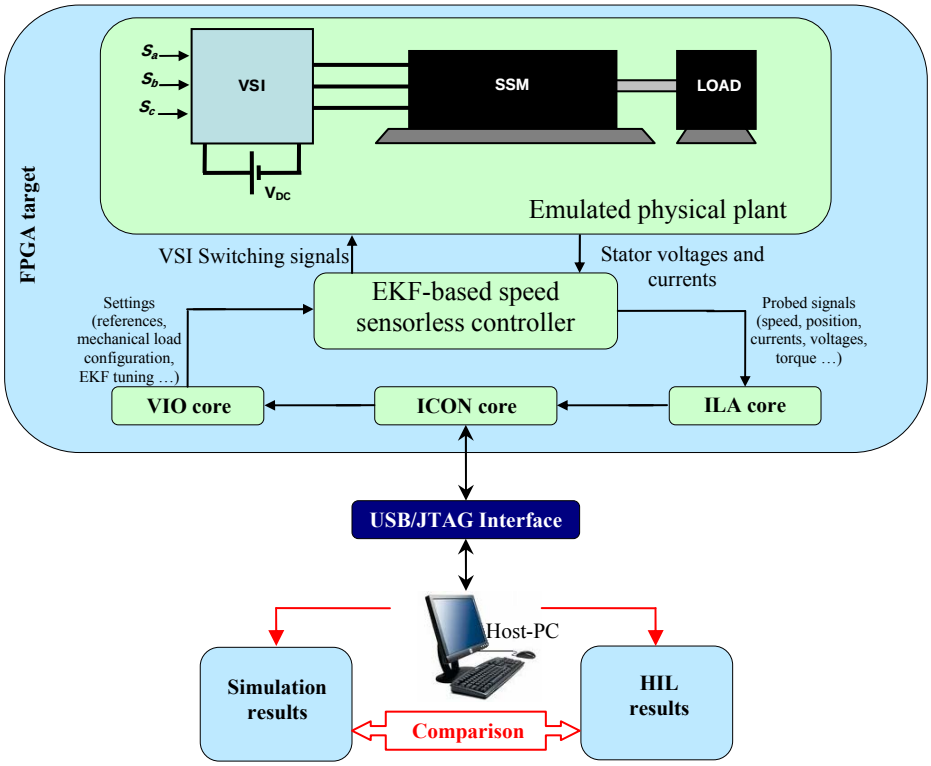


Figure 7.3: HIL validation of the developed FPGA-based speed sensorless controller

Figure 7.3 presents the corresponding synoptic. When using a Xilinx FPGA target, the HIL procedure can be achieved using the ChipScope analyzer [2]. The latter is used to probe the internal signals on the one hand and to configure the design on the other hand. The data transfer is made using the USB/JTAG interface. Depending on the used configuration cable, the supported synchronization clock ranges from 750 kHz to 24 MHz. Also, depending on the used configuration cable and the available memory resources, the transmission rate can reach up to 24 Mbps. In our case, the frequency of synchronization clock is 3MHz and the corresponding transmission rate is 3Mbps.

The implemented design must be associated with the following cores:

- *Integrated CONtroller (ICON)*: This core aims to control the communication between the Host-PC (via the JTAG boundary scan port) and the FPGA target.
- *Integrated Logic Analyzer (ILA)*: The main function of this core is to probe and store data information on the RAM block resources. These data are then sent and displayed by the ChipScope analyzer.
- *Virtual Input/Output (VIO)*: This core aims to monitor and drive internal FPGA signals in real time. In our case, this core is then used to set the speed reference, the direct current reference, the load torque and the EKF covariance matrices.

The emulation of the SSM leans on the developed model that is expressed in chapter 4, in relation (4.14). The Forward Euler discretization method has been adopted. The chosen sampling

period has been set to 1µs and the chosen fixed-point format is *s[30Q28]*. To anticipate the obtained HIL results, acceptable performances have been obtained with this digital realization. However, it is worth noticing that the Euler discretization method has a major drawback in terms of stability. Indeed, the stability of the system is influenced by the value of the sampling period and the data word-length. Thus, with a limited data precision and when the sampling period decreases, the poles of the system move closer to |z| = 1, which move them to the stability limit (Figure 7.4(a)).

To cope with this limitation, it is highly recommended to implement a system model that is based on the delta-operator (δ-operator) [44]. The main advantage of such model remains in the possibility to use a very small sampling period without affecting the stability. Indeed, with the delta approach, when the sampling period decreases, the stability zone increases and becomes closer to the stability zone of the s-domain (Left-hand plane, Figure 7.4(b)). An example of the SSM delta-operator based is given in [108]. As a perspective, this model will be systematically included to the future HIL tests.

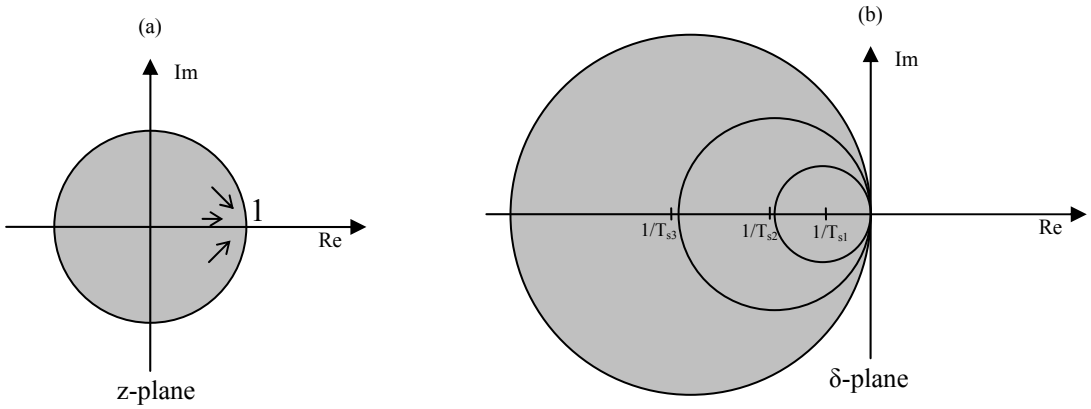


Figure 7.4: Stability regions for discrete z-plane and δ-plane

As far as the VSI is concerned, the implemented emulation model is presented in [109]. The latter includes the non-linearities introduced by the power switches dead time. The buck converter has not been implemented and the SSM rotor current is assumed to be constant (I_{rd}=1.5A).

The following HIL results have been organized in the same way as the fixed point discrete time simulations (chapter 5 part 6). We start by highlighting the validation of the stator current controller, followed by the validation of the speed controller. An open loop validation of the EFK estimation is made and finally the whole sensorless speed controller is validated.

Figure 7.5 presents the HIL results of the stator current controller. The same references settings have been chosen: the direct current is set to 0A and for the quadrature current a step of 2A at start up and a negative step (-1A) at 2s are applied. As shown in Figure 7.5(a) and Figure 7.5(b), the current responses behave as expected with a settling time equal to 40ms. The corresponding 3-phase stator currents are presented in Figure 7.5(d). The stator spatial current vector is plotted in the stationary α-β frame as shown in Figure 7.5(c). Finally, the waveforms of the voltages (VSI output voltage and its fundamental) are presented in Figure 7.5(e) and Figure 7.5(f) .

As for the validation of the speed controller, Figure 7.6 presents the obtained results. Note that in this case, the actual speed and position (from the plant) are injected to the controllers. The same reference settings as during Matlab/Simulink simulations are achieved. Thus, a step of 750 rpm (mechanical speed) is applied at start up and a -750 rpm at 2s (opposite rotor direction). The direct current here is maintained to zero. The expected settling time (650ms) has been obtained and the dynamic and steady state behavior are both the same as those of fixed point discrete time simulations. The waveforms of the position, torque, currents and voltages are also presented.

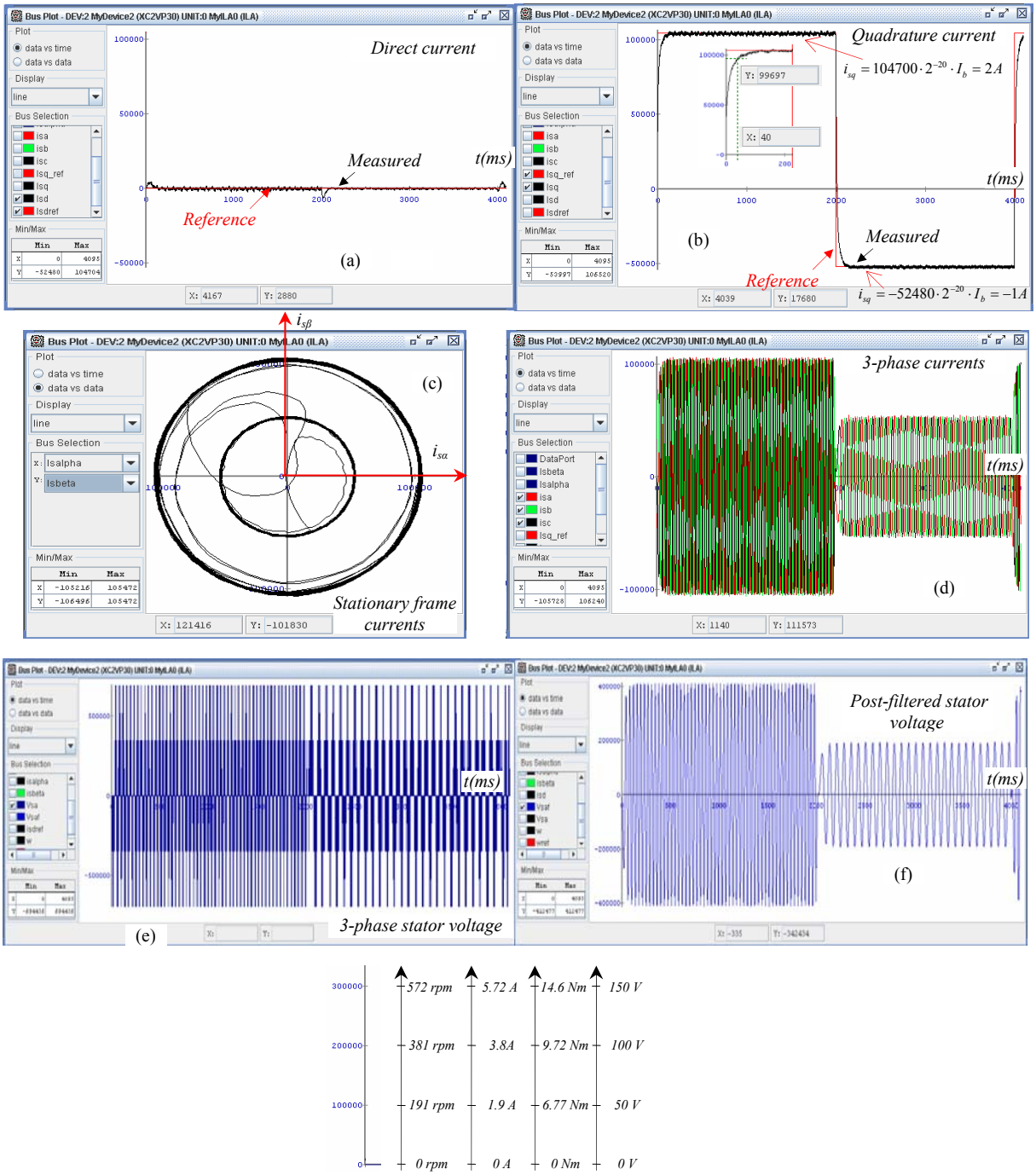


Figure 7.5: HIL validation of the stator current controller

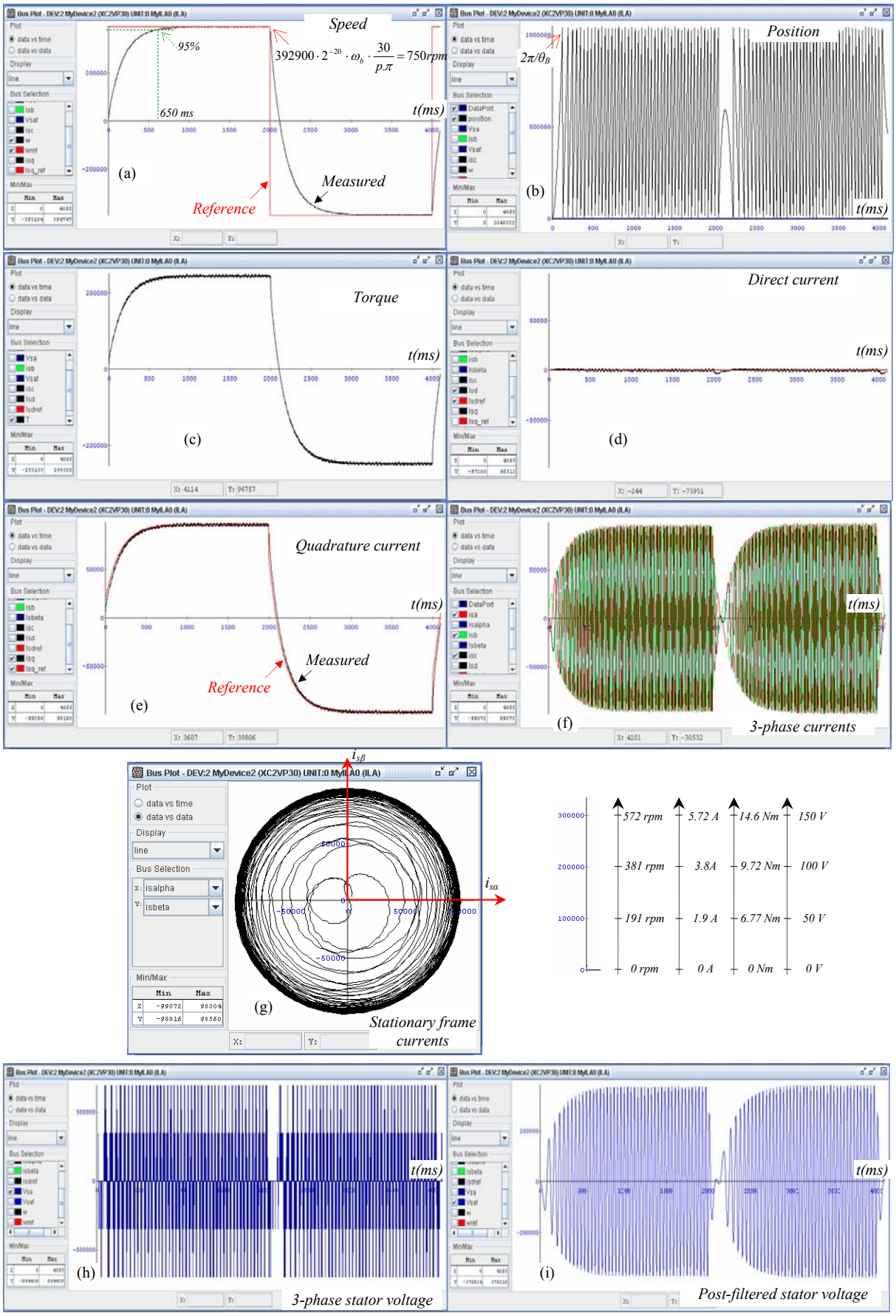


Figure 7.6: HIL validation of the speed current controller

As for the estimation of the EKF, Figure 7.7 highlights the waveforms of the rotor speed and position. The obtained HIL results are compared with the obtained fixed-point discrete-time simulation results. Both sensorless controllers (HIL and Matlab/Simulink versions) operate in the same operating conditions. With the chosen covariance matrices settings (relation (7.1)), the obtained results attest that the EKF converges properly in both rotor directions.

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; Q = \begin{bmatrix} 0.005 & 0 & 0 & 0 \\ 0 & 0.005 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}; R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (7.1)$$

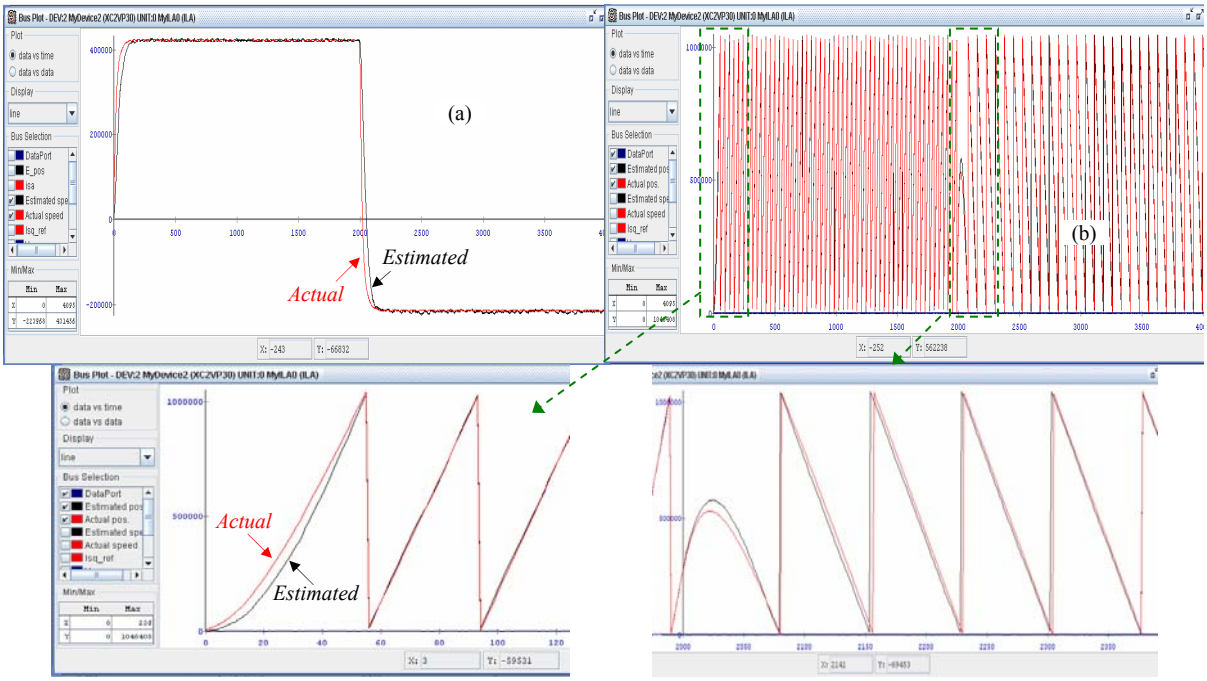


Figure 7.7: HIL validation of the EKF estimation

Finally, when it comes to the HIL validation of the whole EKF-based sensorless speed controller, Figure 7.8 presents the obtained results. Here again, with the same operating conditions as during Matlab/Simulink simulations, the same behavior has been obtained. The estimated speed and position are injected to the speed and stator current controllers. A step of 750 rpm (mechanical speed) is applied and a negative -750 rpm is applied at 2s. The objective here is to validate the functionally at both rotor directions. The direct current is set to 0A. The waveforms of the rotor position, the developed torque, currents and voltages are highlighted.

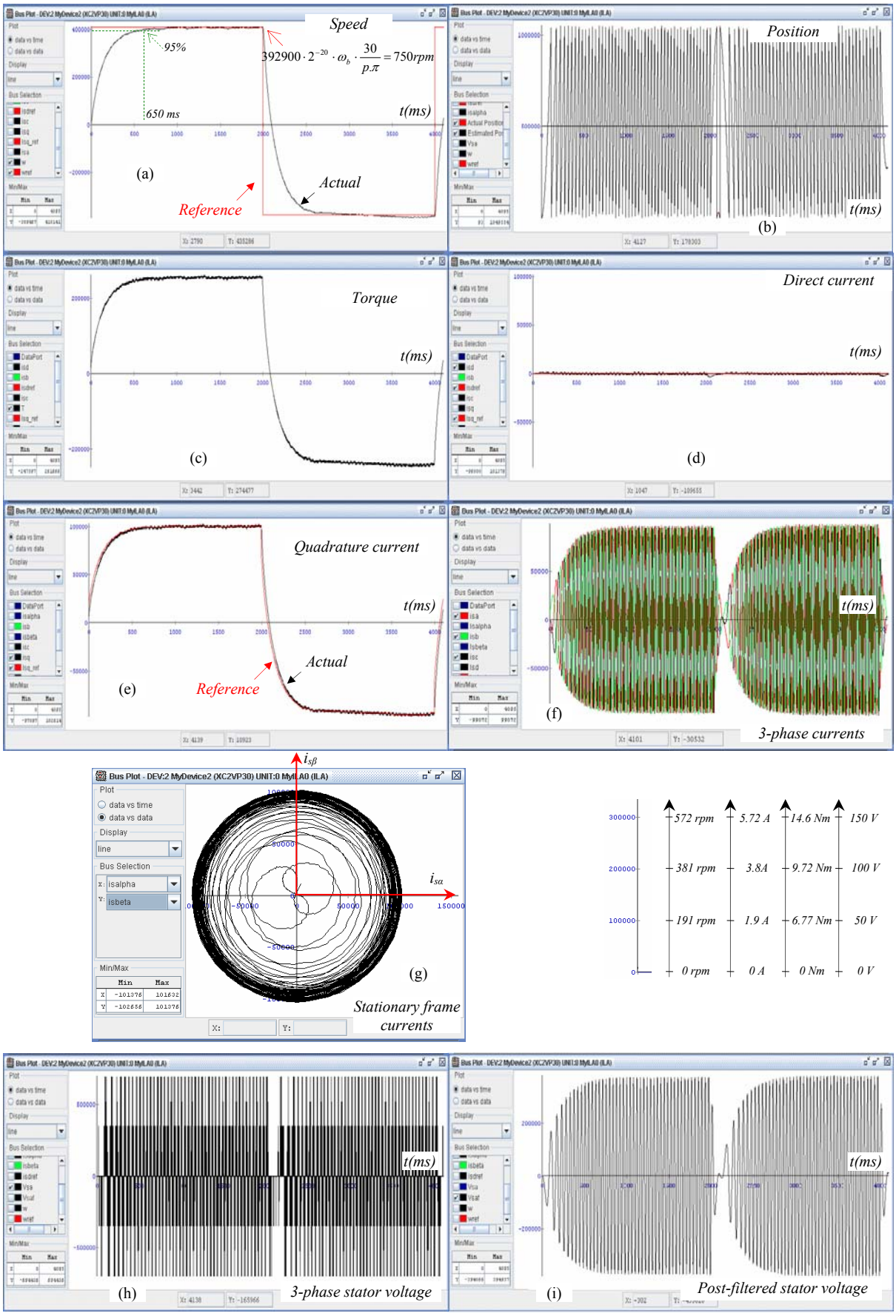


Figure 7.8: HIL validation of the FPGA-based sensorless speed controller

4. Experimental validation

4.1. Validation of the stator current controller

In this section, we are presenting the obtained experimental results after having implemented the FPGA based stator current controller. During this test, the rotor position is provided by the used position absolute encoder (10-bit resolution encoder). From this position, the rotor speed (used for axes decoupling) is performed by the speed estimator, discussed in [81]. The principle of this estimator is based on the position variation. Indeed, the duration between two position increments is calculated using a counter. Then at each increment, the value of this counter is registered and used to address the memory where the corresponding pre-calculated speed is stored. The corresponding mathematical equation is expressed in relation (7.2).

$$\omega = \frac{2\pi}{1024} \cdot \frac{\theta[k] - \theta[k-1]}{N_{counter} \cdot T_{clk}} \tag{7.2}$$

Where ω is the electrical speed and θ is the electrical 10-bit coded position. $N_{counter}$ is the value of the counter at each position variation and T_{clk} is the clock period.

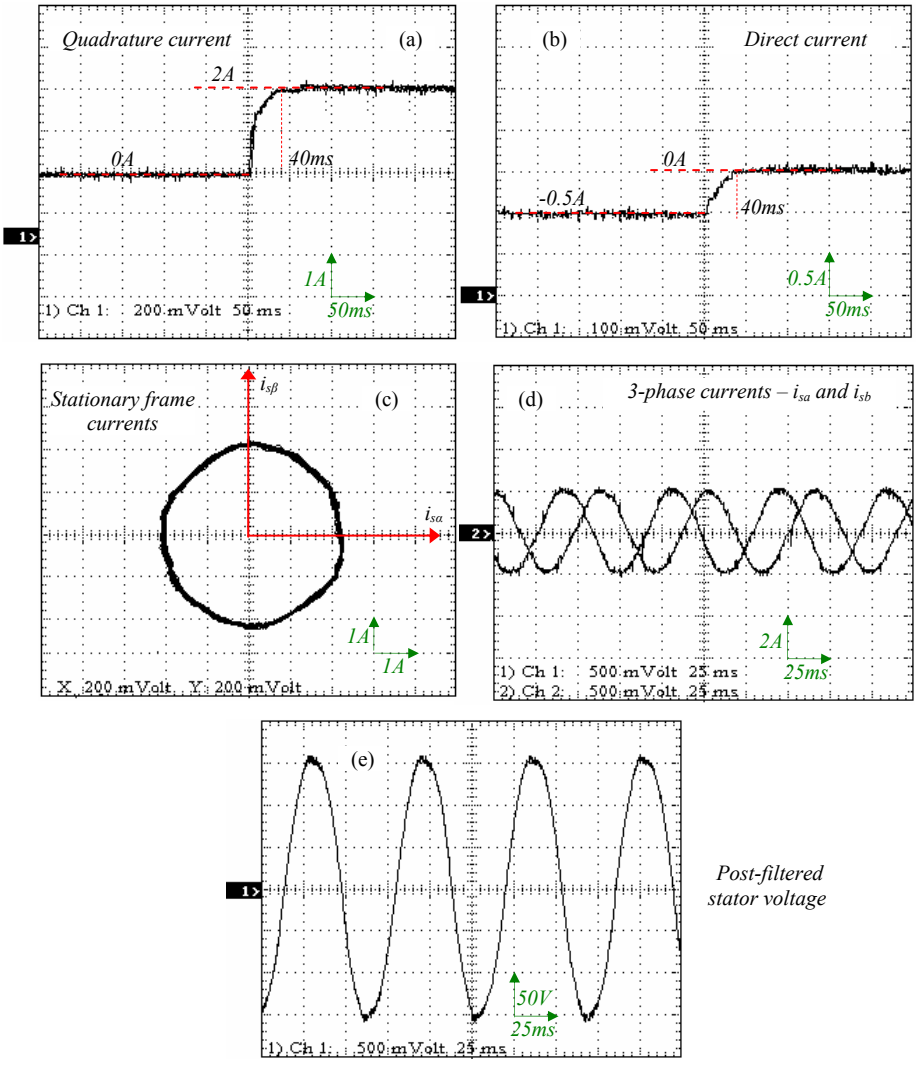


Figure 7.9: Experimental validation of the stator current controller

Figure 7.9 presents the obtained experimental results. A 0-to-2A step reference is applied to the quadrature current and a -1A-to-0A step reference is applied to the direct current. Figure 7.9(a) and Figure 7.9(b) show respectively the quadrature i_{sq} and direct i_{sd} current waveforms. These data are processed internally and converted by the used DAC boards. First order responses have been obtained and the settling time is equal to 40ms which is the expected value. Figure 7.9(c) presents the steady state α-β currents and Figure 7.9(d) presents the 3-phase i_{sa} and i_{sb} current waveforms. The steady state post-filtered (output of a low-pass RC filter, cutoff frequency: 300 Hz) stator voltage is shown in Figure 7.9(e).

4.2. Validation of the sensor-based speed controller

Here again, the speed controller has been experimentally validated using the absolute 10-bit position encoder. The previously discussed speed estimator generates the rotor speed. The achieved test consists in applying a progression of positive and negative 750rpm steps. The direct current has been set to 0A. As it can be seen in Figure 7.10 where the obtained experimental results are highlighted, the expected performances (response dynamic and settling time) have been successfully reached: first order response and 650ms settling time. Figure 7.10(a) and Figure 7.10(b) present the rotor speed and position. Figure 7.10(c) presents the waveform of the stator current. The amplitude of the steady state current is equal to 1A and an over-current is observed at each rotor direction variation which is due to the mechanical load. Finally, the waveform of the post-filtered (fundamental) stator voltage is depicted in Figure 7.10(d).

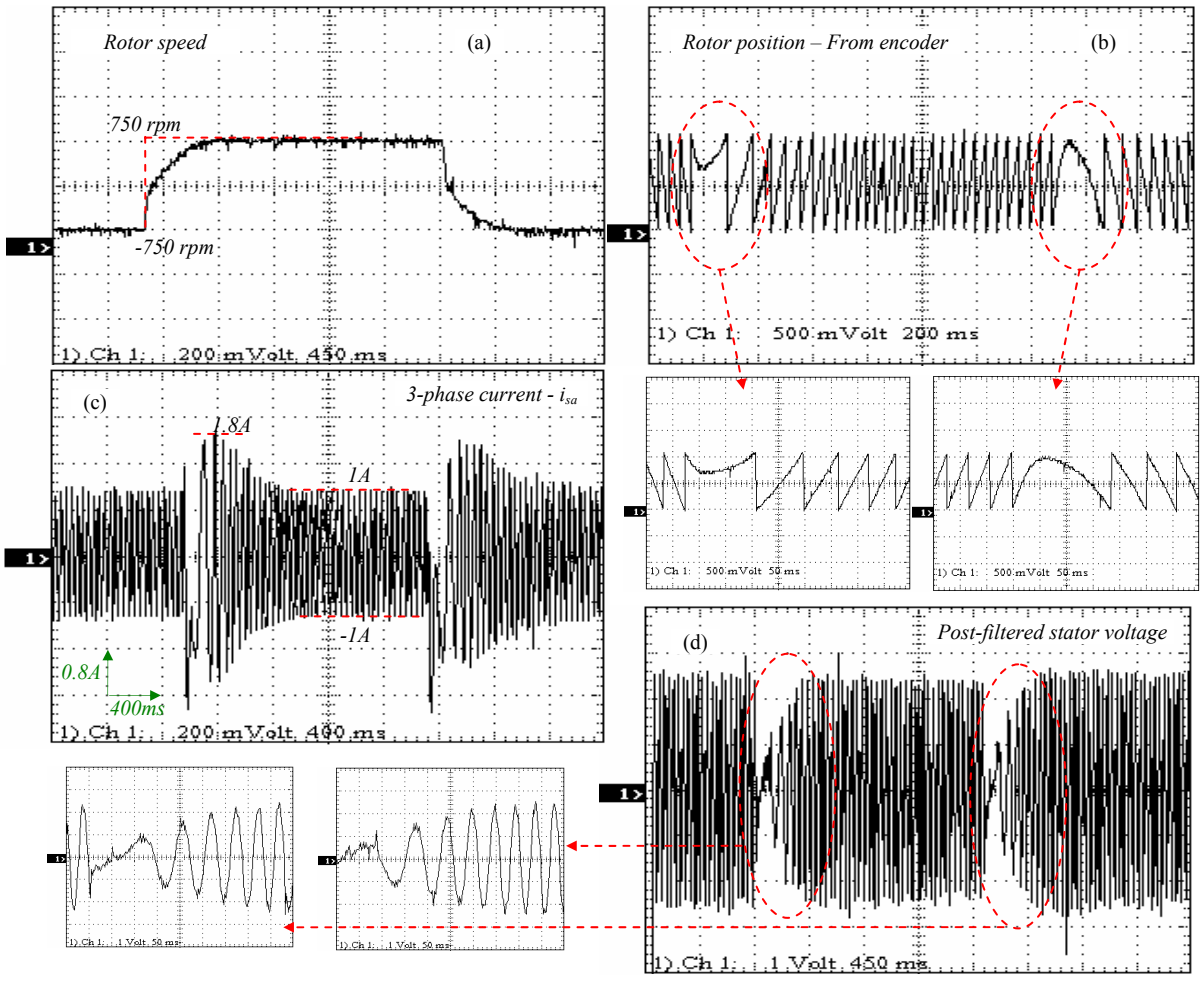


Figure 7.10: Experimental validation of the sensor-based speed controller

4.3. Validation of the EKF observer

This experimental test aims to make an open loop validation of the EKF. The measured position and the processed speed (by the speed estimator) are used. To do so, the stator current controller has been implemented where a 0-to-2A step is applied to the quadrature current reference and the direct current is set to zero. Figure 7.11(a) shows the waveform of the actual and estimated speed and Figure 7.11(b) shows the waveform of the actual and estimated electrical position. As a reminder, the d-q based SSM model has been implemented with the infinite inertia approximation (chapter 4, part 6). The identification of the SSM parameters has been discussed in [Appendix B].

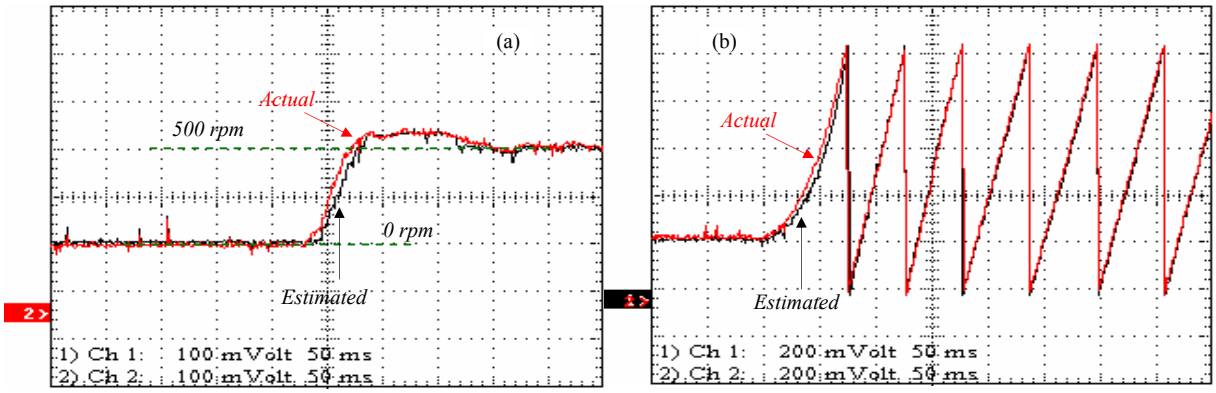


Figure 7.11: Waveforms of the estimated rotor speed and position

As we are dealing with the experimental validation of the EKF, it is worth presenting the obtained estimation behavior of the optimized EKF versions. As discussed in chapter 5 during the algorithm optimization, the chosen optimized EKF versions are: the infinite Kalman gain version and the matrix symmetrization EKF version. It has been observed during simulations that the first EKF version (K∞) doesn't preserve the estimation dynamic. This has been confirmed during the experimentation (Figure 7.12(b)). As for the matrix symmetrization EKF version, the experimentation confirms that the estimation dynamic is preserved and no miss-convergence is observed at start up (Figure 7.12(a)).

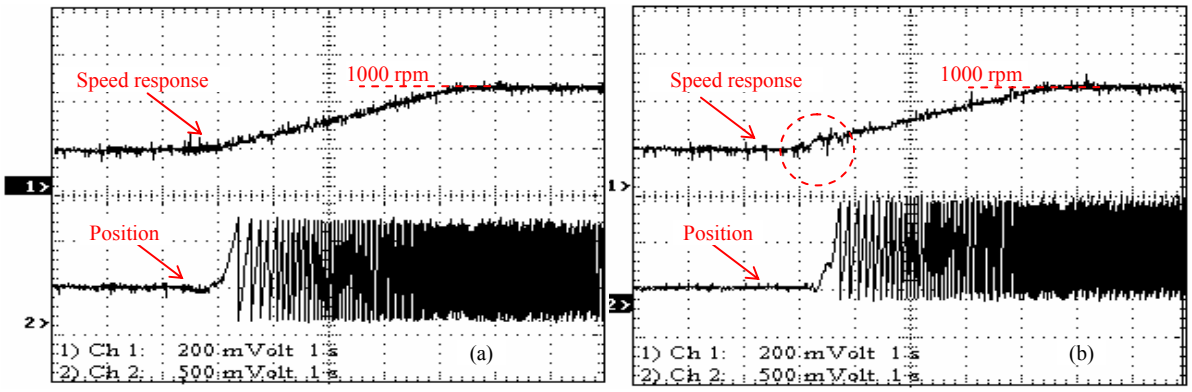


Figure 7.12: Validation of the optimized EKF versions

4.4. Validation of the whole sensorless speed controller

The estimated rotor position and speed are now injected into the speed and stator current controllers. Figure 7.13 presents the obtained experimental results. Figures 7.13(a-d) correspond to the test where a progression of positive and negative 750rpm steps are applied as mechanical speed references. The direct current is still maintained to zero. Figures 7.13(e-f) present the speed ramp

response where a 1000 rpm mechanical speed reference ramp has been applied. It is to be noted that the observed convergence at start up has been obtained after having determined the standstill rotor position. Indeed, if the motor is started without the a priori knowledge of the initial position, it cannot be guaranteed that this motor will rotate in the expected direction during start up. An additional algorithm is then to be implemented so as to estimate the standstill position (e.g high frequency signal injection algorithm, [92]). During primarily test procedure, the initial position has been measured using the position encoder. Another alternative to cope with this start up miss-convergence is to implement an open-loop controller that operates during start up and imposes to rotor position. With this procedure, a transition algorithm is to be added to ensure the transition between the imposed position and the estimated position.

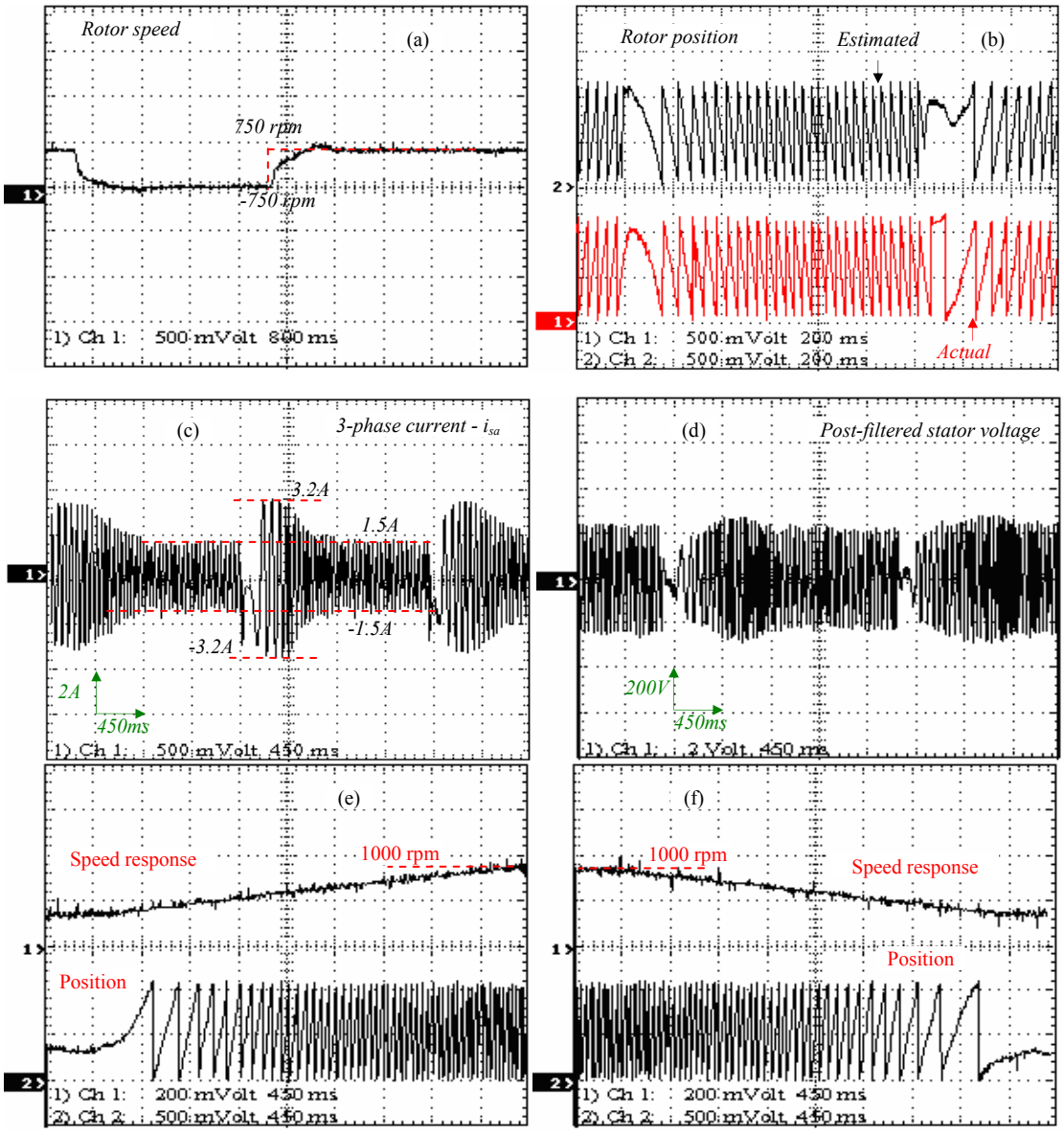


Figure 7.13: Experimental validation of the whole sensorless speed controller

In order to expand the sensorless controller tests, the robustness evaluation against load torque variation has been achieved. Figure 7.14 highlights the speed waveform in the case of a 2Nm and 4Nm load torque variation.

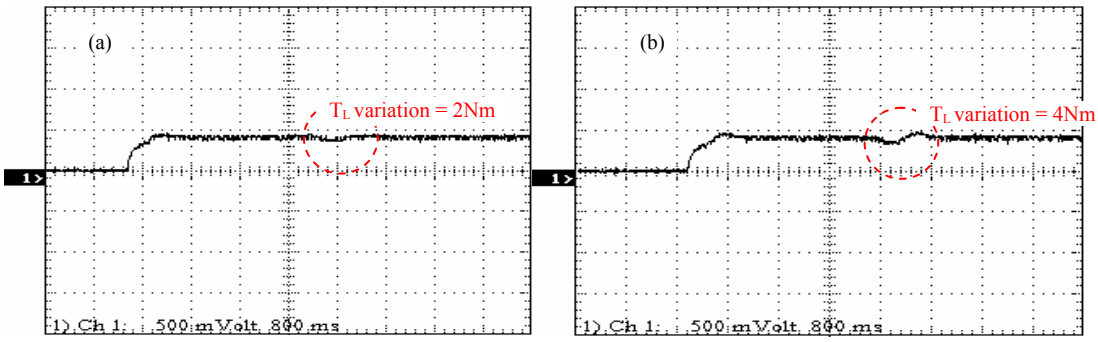


Figure 7.14: Sensorless speed controller: Experimental evaluation of the robustness against load torque variation – waveforms of the rotor speed

In addition, the evaluation of the developed sensorless controller on a wide operating speed range was also made. Figure 7.15 presents the waveforms of the rotor position from 125 rpm to 2500 rpm. With the used 1500-rpm SSM and in order to reach over-speed operating conditions, the value of the applied rotor current is decreased and a negative direct current reference is applied to the stator current controller.

The obtained results indicate that, with the used 1500-rpm SSM, the developed EKF-based sensorless speed controller is able to operate at over-speed operating conditions (up to 170%) and down to 8% of the nominal speed (125 rpm). Indeed, at very low speed and at standstill, the EKF does not guarantee a zero steady state estimation error since the back-EMF is very low (zero at standstill). Consequently, this attests the necessity to implement a specific estimation method that is dedicated to very low speed operating range.

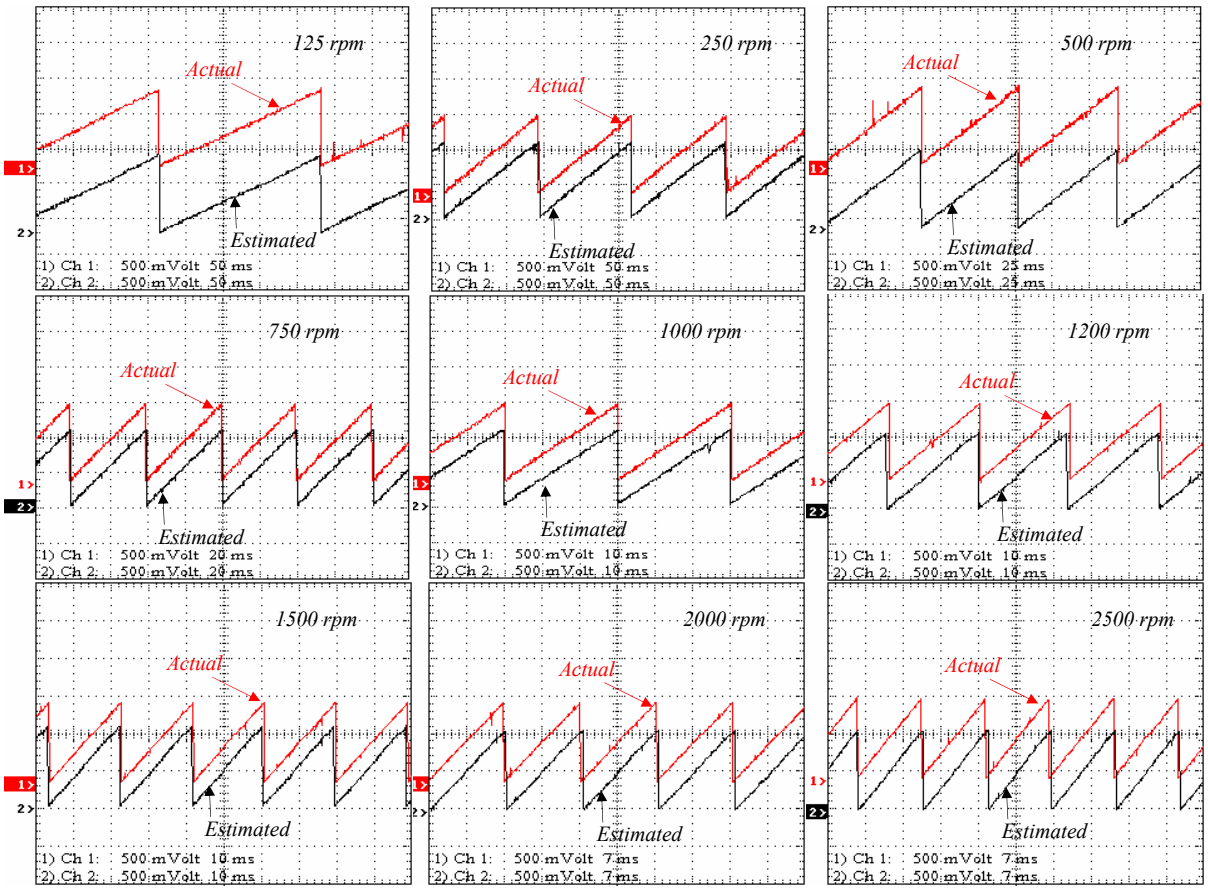


Figure 7.15: Experimental validation of the sensorless speed controller – expansion to a large operating speed range – waveforms of the measured and estimated rotor position.

5. Conclusion

This chapter was aimed to present the experimental validation of the developed fully FPGA-based sensorless speed controller. At the beginning, the features of the experimental platform were listed. This was followed by the Hardware-In-the-Loop (HIL) validation. The HIL results of the stator current controller were firstly presented. They were followed by those of the sensor-based speed controller. Then the estimation of the developed EKF was validated before the ultimate HIL validation of the whole sensorless speed controller. At next, experiments were carried out. Here again, the provided results are organized similarly to the HIL validation. In addition, the robustness of the sensorless speed controller against load torque variation was checked and a wide range operating speed range was tested.

General conclusion and perspectives

1. General conclusion

The objective of this thesis was to analyze and emphasize the contribution of FPGA devices in the field of complex AC drive applications. This contribution was quantified in terms of control performances and in terms of system integration. The chosen application was the sensorless speed controller of a Salient Synchronous Machine (SSM) based on the Extended Kalman Filter (EKF).

Such EKF based sensorless controllers are systematically implemented using software devices such as DSPs. Due to the fixed architecture of these solutions, the treatment is fully serialized. Consequently, the implementation of complex control algorithm leads to long execution time. This has a significant impact on the control quality and bandwidth since the sampling frequency is limited and delays are introduced in the control closed-loop. Our idea trough this thesis is to use hardware devices such as FPGAs. With these solutions, it is possible to exploit the parallelism of the algorithm and then reduce significantly the execution time. Consequently, the control quality and bandwidth are not affected. Furthermore, the use of FPGAs can improve the system integration. For example, it is possible to develop a generic EKF that is able to estimate quasi-simultaneously variables of many heterogeneous systems. Two typical examples were treated in chapter 2.

To make the design more manageable and less intuitive, the development of the FPGA-based sensorless speed controller was achieved by following a dedicated design methodology. The latter consists of a four major phases. The first one is the preliminary system specification. This consists in making a hardware specification of the control system and making the algorithm benchmarking. The second phase is dedicated to the development of the algorithm where the necessary modular partitioning, digital realization, algorithm optimization and simulations are achieved. The third phase is the development of the FPGA architecture. This development starts with the optimization of the architecture and ends with the physical implementation. Time/area performances are also analyzed and compared to the expected time/area constraints. The last step is the experimentation which includes the Hardware-In-the-Loop (HIL) and the experimental validations.

At the beginning of this thesis report, a state of the art FPGA technology was made. In this chapter, author described the structure of the recent FPGA devices and discussed their contribution in the field of power electronics and drive applications, especially the case of complex control applications. Also, the previously discussed design methodology was presented. This chapter was followed by the description of a fully integrated FPGA-based controller for a Permanent Magnet Synchronous Machine (PMSM) associated with a resolver sensor. This development was firstly made so as to evaluate the system integration of a sensor-based FPGA-based controller. Then the following chapters (4, 5, 6 and 7) dealt with the development of the FPGA-based sensorless speed controller. It can be noticed that these chapters were organized according to the design methodology.

2. Perspectives

During these researches, it has been proved that when combining the speed and the integration capacity of the recent FPGAs, it is possible to implement complex AC drive controllers. With the developed EKF-based sensorless controller, it is possible to use a low cost FPGA device and at the same time ensure short execution time. These promising results made us think about different possible perspectives. We have established them into two groups: perspectives related to the development of the algorithm and perspectives related to the development of the FPGA architecture.

2.1. Algorithm perspectives

During the specification of the system model, we had chosen the state space model based on the infinite inertia hypothesis. This led to a 4^{th} order EKF. As a perspective, it is interesting to evaluate the performances of FPGAs with a more complex system model which includes for example, the load torque, rotor excitation equation, or also saturations. Furthermore, using the EKF, it is interesting to extend the system model for online identification of the system parameters.

Other interesting investigations are related to the digital realization of the filter. Indeed, the discretization of the system model is based on Euler approximation. In fact this corresponds to a first order approximation of the exponential of the continuous-time state space matrix. Although this approximation allows a simple model, it is important to test a higher order approximation.

Also the choice of the fixed-point data format is a promising research field. Indeed, in this thesis work, this choice has been done intuitively during simulations and using Matlab/Simulink fixed-point tools. In the next future, we are going to apply the methodology discussed in chapter 2 and presented in [40], [41]. This is a less intuitive methodology that aims to choose separately the fixed-point format of coefficients and variables by studying the stability and the steady state behavior of the EKF.

Another important topic is the implementation of the continuous-time EKF algorithm [112]. This is possible with the obtained short execution time with FPGAs. In this context, the use of delta-transform could be of great interest.

2.2. FPGA development perspectives

Additional perspectives related to the FPGA architecture development are to be focused on. At first, improvements are to be brought to the proposed architecture optimization procedure (chapter 2, Figure 2.20). As proposed in this work, the first and the second steps of this procedure are made manually. The idea is to develop an optimization algorithm that calculates the appropriate level of factorization according to the defined implementation constraints.

Finally, along these FPGA developments, it is also important to evaluate the value-added of FPGAs when exploiting the possibility to implement Hardware/Software treatments. Indeed, the recent FPGAs give the possibility to implement high performance processor cores. To this purpose, co-design approaches and methodologies are to be adopted so as to make the efficient partitioning between the hardware and software treatments. This is the subject of the associate thesis work [74].

Appendix A

Compensation of the VSI non-linearities

1. Voltage Source Inverter Characterization

Figure A.1 presents the topology of a Voltage Source Inverter (VSI). The outputs are applied to the machine stator windings. The latter are assumed to be Y-connected and balanced. In the following, we are going to start the development of the VSI linear model before studying the introduced non-linearties.

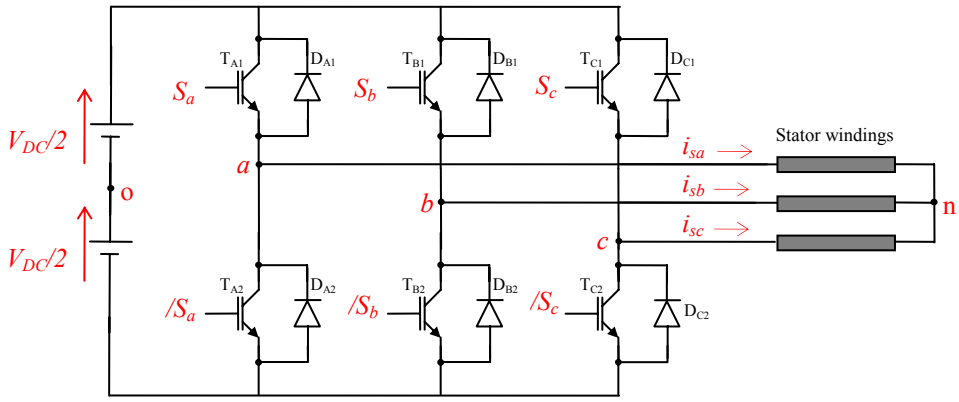


Figure A.1: 3-phase VSI topology

1.1. VSI linear model

The VSI linear model is developed without considering any dead time and turn-on/off delays introduced by the switching signals. Thus, the ideal switching function can be written as a simple arithmetic relation $Si + \overline{Si} = 1$ (i: a, b or c). As consequence, the voltages V_{ao}, V_{bo}, V_{co} can be expressed as follows,

$$\begin{aligned} V_{ao} &= (2 \cdot S_a - 1) \cdot \frac{V_{DC}}{2} \\ V_{bo} &= (2 \cdot S_b - 1) \cdot \frac{V_{DC}}{2} \\ V_{co} &= (2 \cdot S_c - 1) \cdot \frac{V_{DC}}{2} \end{aligned} \tag{A.1}$$

Assuming that the stator windings are Y-connected and balanced, the 3-phase output voltages V_{san}, V_{sbn} and V_{scn} are written according to relation (A.2).

$$\begin{bmatrix} V_{san} \\ V_{sbn} \\ V_{scn} \end{bmatrix} = \frac{V_{DC}}{3} \cdot \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} S_a \\ S_b \\ S_c \end{bmatrix} \tag{A.2}$$

1.2. Analysis of VSI non-linearities

In this section, the non-linear VSI modeling is achieved. It describes how the model was developed by taking into account the effects of the dead time between switching signals, turn-on/off delays and voltage drop introduced by the power switches (IGBTs). As a first step, the model of one single VSI leg is achieved as shown in Figure A.2. Then, it is duplicated for the other two legs with the same manner.

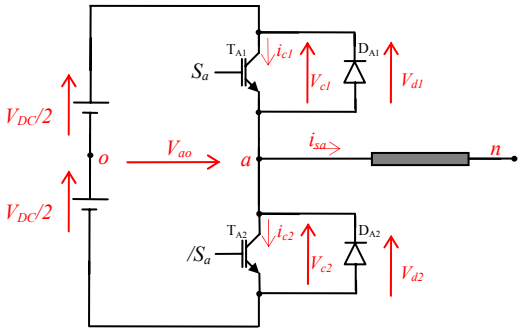


Figure A.2: One single VSI leg

For this leg, depending on S_a and $/S_a$ states, four possible combinations are considered. As shown in Figure A.3, these combinations are obtained after introducing the necessary dead time t_{DT} to avoid a simultaneous conduction of the switches.

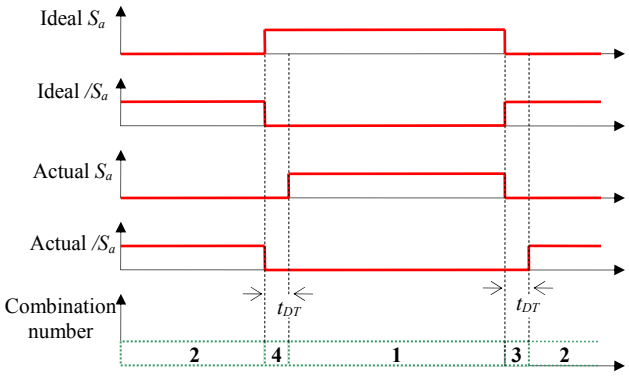


Figure A.3: Dead time introduction and the four possible combinations

In addition to this dead time, the listed-below time delays are introduced by these switches. They are to be considered to determine accurately the V_{ao} voltage waveform.

t_{on} : turn-on propagation time
t_{off} : turn-off propagation time
t_r : rise time (from 10%ic to 90%ic)
t_f : fall time (from 90%ic to 10%ic)

Figure A.4 shows an idealized timing characteristic of an IGBT during turn-on and turn-off.

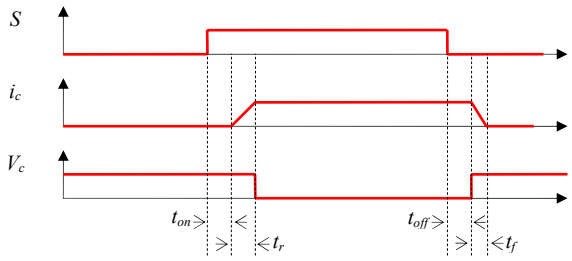


Figure A.4: Idealized IGBT timing diagram

For each combination, the value of V_{ao} voltage is determined according to the direction of the output current i_{sa}. The later is used to determine whether the current is flowing through IGBT or the anti-parallel diode. Figure A.5 shows the V_{ao} waveform when i_{sa} is positive or negative.

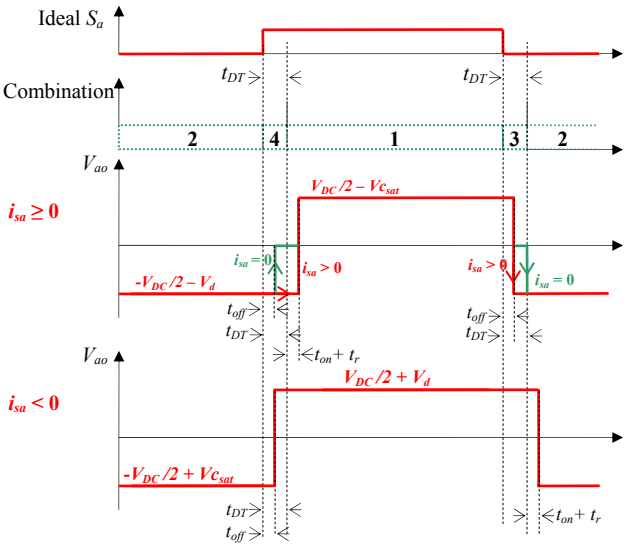


Figure A.5: VSI output voltage waveform depending on current direction

2. Compensation of the VSI non-linearities

In the developed application, switching signals are processed using a Carrier Based PWM with Zero Sequence Signal (CB-ZSS-PWM) [62], [75]. As explained in Figure A.6, the purpose of this PWM method is to fix the instantaneous average value of VSI output voltage $<V_{io}>$ (i: a, b or c) and to make it equal to an instantaneous reference voltage $<V_{io}>^*$. This later is obtained after adding, to $<V_{in}>^*$ an additional ZSS which occurs between N and O points (Figure A.1).

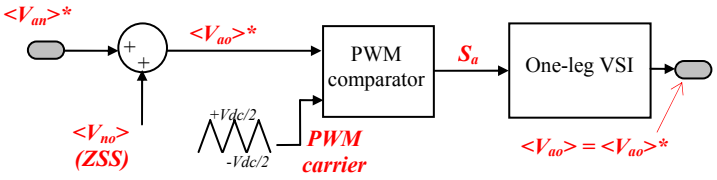


Figure A.6: CB-ZSS-PWM principle

For example, for the first VSI leg, the output average voltage $<V_{ao}>$ expression, extracted from Figure A.5, is given in relation (A.3),

$$< V_{AO} >= D \cdot V_1 - V_2 + D \cdot V_2 + V_1 \cdot \left(\frac{t_2 - t_1}{T} \right) - V_2 \cdot \left(\frac{t_2 + t_1}{T} \right) \tag{A.3}$$

Where T and D are, respectively, the PWM period and duty cycle. V_1 and V_2 are the positive and negative voltage levels and t_1 and t_2 the time delays defined as,

$$\begin{cases} t_1 = t_{DT} + t_{on} + t_r \quad ; \quad t_2 = t_{off} \quad when \quad i_{sa} > 0 \\ t_1 = t_{off} \quad ; \quad t_2 = t_{off} \quad when \quad i_{sb} = 0 \\ t_1 = t_{off} \quad ; \quad t_2 = t_{DT} + t_{on} + t_r \quad when \quad i_{sc} < 0 \end{cases} \tag{A.4}$$

Without considering VSI non-linearities, the ideal output voltage can be written as,

$$< V_{AO} >_{ideal} = D \cdot V_1 - V_2 + D \cdot V_2 \tag{A.5}$$

From these two last relations, the introduced error is extracted and expressed as follows,

$$< \varepsilon_{VSI} >= V_1 \cdot \left(\frac{t_2 - t_1}{T} \right) - V_2 \cdot \left(\frac{t_2 + t_1}{T} \right) \quad \text{(A.6)}$$

The chosen compensation method in this application consists in subtracting this error from the PWM reference voltage. As consequence, the ideal average voltage will be applied to the VSI output. Figure A.7 summaries the adopted compensation principle.

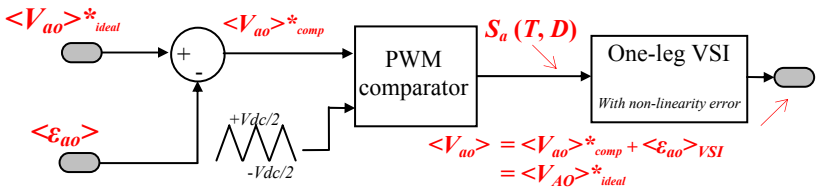


Figure A.7: CB-ZSS-PWM unit with the compensation of VSI non-linearities

3. Application to stator current controller

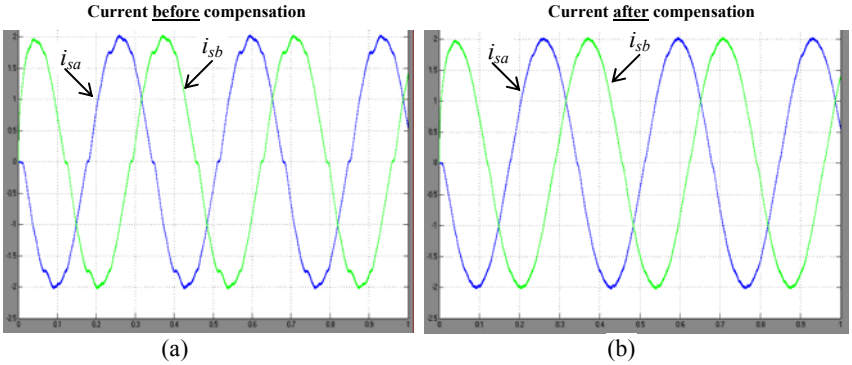
A characterization of the implemented VSI is firstly achieved. This consists in measuring the dead time and IGBT turn on/off delays. These timing performances are listed below,

- $t_{off} = 2.12$ µs
- $t_{DT} + t_{on} + t_r = 5.7$ µs

The simulation results presented in Figure A.8 show the stator currents i_{sa} and i_{sb} before (a) and after (b) applying the VSI non-linearies compensation. These results have been obtained for the following settings:

- $i_{sd}^* = 0$A
- $i_{sq}^* = 2$A
- *PWM frequency* = 6 KHz

Before the compensation, the observed current distortions near 0A are mainly due to the effect of VSI time delays. Indeed, the effect of these delays over the switching signal duty cycle increases when value of the current is low.



(a) (b)

Figure A.8: Simulation results before and after compensation

Figure A.9 shows the variation of Total Harmonic Distortion (THD) of stator currents over the electrical rotor speed before and after applying compensations.

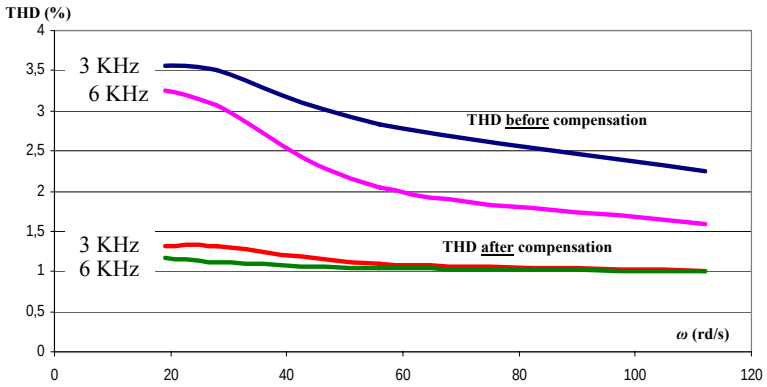


Figure A.9: THD = f (ω, PWM frequency)

4. Experimental results

The Experimental set up is presented in Figure A.10. It is composed of a 0.8kW synchronous motor associated with a 1024 points absolute encoder, current sensors and a controlled mechanical load. The SM is supplied by a SEMIKRON VSI module which implements SKM 50GB123D IGBT modules. Timing performances of this VSI are listed below,

- $t_{off} = 2.12$ µs
- $t_{DT} + t_{on} + t_r = 5.7$ µs

The used FPGA target is the Fusion AFS600 FPGA which implements a stator current controller.

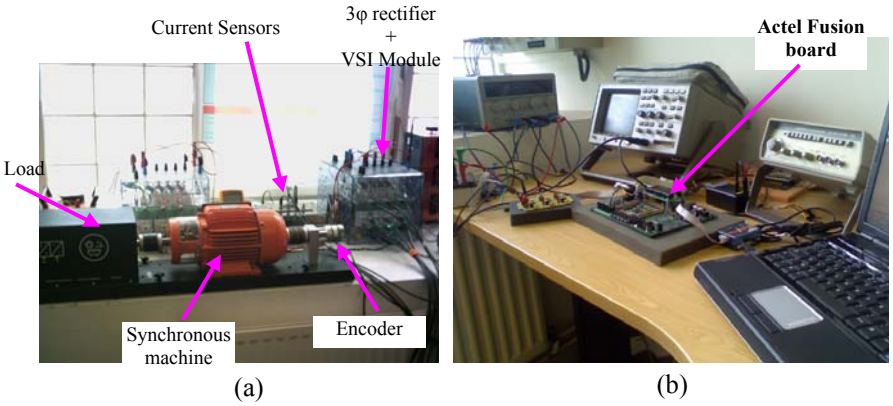


Figure A.10: Prototyping platform
(a) : Power circuit (b) : Fusion FPGA Control circuit

Figure A.11 presents the stator current instantaneous waveforms after and before the compensation of the VSI non-linearities. Before compensation, distortions appear in the current waveform (Figures A.11(a,c)). These distortions are clearly reduced by the chosen compensation method as shown in Figures A.11(b,d).

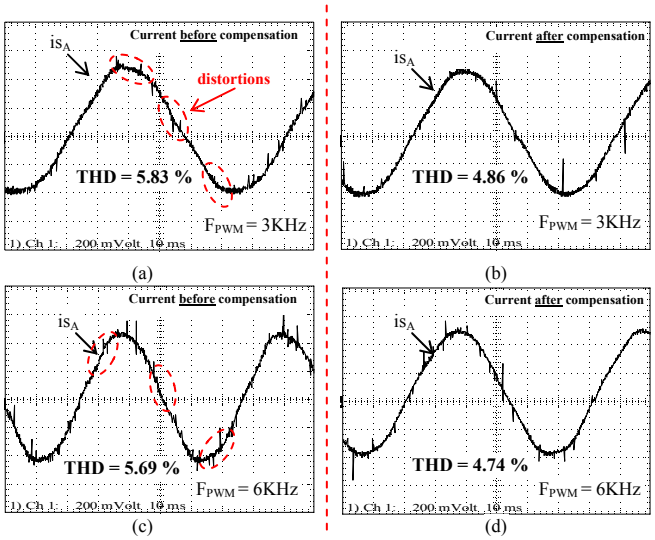


Figure A.11: Stator current waveforms before and after compensation
i_{sd}* = 0A; i_{sq}* = 2A; F_{PWM} = (3kHz, 6kHz)

Appendix B

Salient Synchronous Machine: Modeling and parameter identification

1. d-q modeling of the Salient Synchronous Machine (SSM)

In this section, the modeling in the rotating d-q reference frame of the implemented rotor wounded SSM is made. The latter is composed of 3-phase stator windings and an excitation rotor winding. As a first assumption, the stator windings are Y-connected and balanced. Figure B.1 overview the 3-phase representation of the SSM, [62], [111].

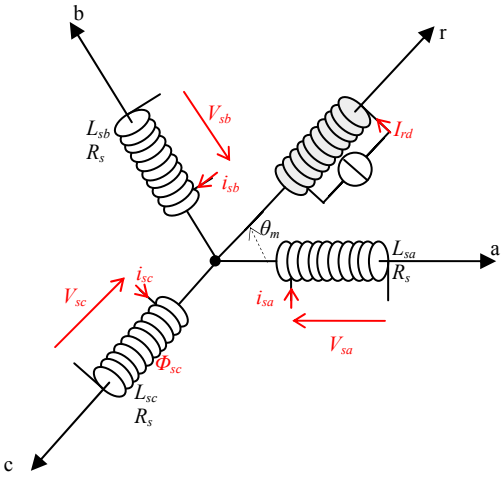


Figure B.1: 3-phase representation of an SSM

The angle θ_m corresponds to the mechanical rotor position. The latter is linked to the mechanical angular rotating speed Ω_m according to relation (B.1).

$$\Omega_m = \frac{d\theta_m}{dt} \tag{B.1}$$

With regards to the number of pole pairs p, relations (B.2) and (B.3) express respectively the relationship between the electrical and the mechanical positions and speeds.

$$\theta = p \cdot \theta_m \tag{B.2}$$

$$\omega = p \cdot \Omega_m \tag{B.3}$$

To start the d-q modeling of the SSM, let's recall the principle and the equations of the used coordinate transformations.

1.1. Coordinate transformations

In this section, the principle and the equations of the Clark transformation and the Park transformation are presented.

1.1.1. Clark transformation

The Clark transformation aims to transform a 3-phase reference frame *(X_a, X_b, X_c)* to a stationary 2-phase (α-β) reference frame *(X_α, X_β)* according to the following relation. Figure B.2 gives the corresponding vector representation.

$$\begin{bmatrix} X_\alpha \\ X_\beta \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \cdot \begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\sqrt{3} & 2/\sqrt{3} \end{bmatrix} \cdot \begin{bmatrix} X_a \\ X_b \end{bmatrix} \quad \text{(B.4)}$$

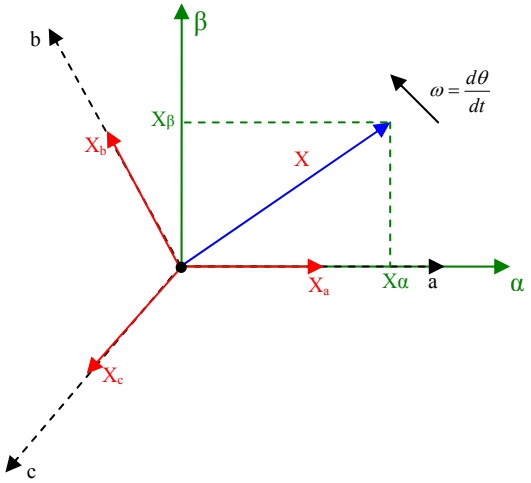


Figure B.2: vector-representation of the Clark transformation

1.1.2. Park transformation

The Park transformation aims to transform a 3-phase reference frame *(X_a, X_b, X_c)* to a rotating (d-q) reference frame (X_d, X_q). This frame is composed of a direct axis d and a quadrature axis q. The direct axis d is linked to the rotor axis. Figure B.3 gives the corresponding vector representation.

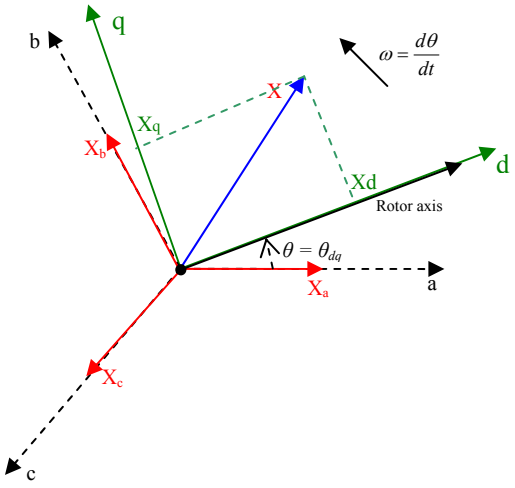


Figure B.3: vector-representation of the Park transformation

The (d-q) based components are obtained by multiplying their stationary based counterparts by the rotation matrix *[R(-θ)]*. This last is expressed in relation (B.5) and the final Park transformation equation is expressed in relation (B.6).

$$[R(\theta)]=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix} \quad ; \quad [R(-\theta)]=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix} \tag{B.5}$$

$$\begin{bmatrix}X_d\\ X_q\end{bmatrix}=[R(-\theta)]\cdot\begin{bmatrix}X_\alpha\\ X_\beta\end{bmatrix}=\frac{2}{3}\cdot\begin{bmatrix}\cos\theta & \cos(\theta-2\pi/3) & \cos(\theta-4\pi/3)\\ -\sin\theta & -\sin(\theta-2\pi/3) & -\sin(\theta-4\pi/3)\end{bmatrix}\cdot\begin{bmatrix}X_a\\ X_b\\ X_c\end{bmatrix} \tag{B.6}$$

1.2. Model equations

1.2.1. Electrical equations

Relation (B.7) gives the 3-phase electrical equations of the SSM. Its corresponding matrix-based formulation is given in relation (B.8).

$$\begin{aligned} V_{sa} &= R\cdot i_{sa}+\frac{d}{dt}[\Psi_{sa}]\\ V_{sb} &= R\cdot i_{sb}+\frac{d}{dt}[\Psi_{sb}]\\ V_{sc} &= R\cdot i_{sc}+\frac{d}{dt}[\Psi_{sc}] \end{aligned} \tag{B.7}$$

$$[V_{sabc}]=R\cdot[i_{sabc}]+\frac{d}{dt}([\Psi_{sabc}]) \tag{B.8}$$

When applying the Park transformation to these 3-phase vectors, the following d-q based equations can be extracted.

$$\begin{aligned} v_{sd} &= R\cdot i_{sd}+\frac{d(\Psi_{sd})}{dt}-\omega\cdot\Psi_{sq}\\ v_{sq} &= R\cdot i_{sq}+\frac{d(\Psi_{sq})}{dt}+\omega\cdot\Psi_{sd} \end{aligned} \tag{B.9}$$

Where,

$$\begin{aligned} \Psi_{sd} &= L_{sd}\cdot i_{sd}+M_{sr}\cdot I_{rd}\\ \Psi_{sq} &= L_{sq}\cdot i_{sq} \end{aligned} \tag{B.10}$$

1.2.2. Electromagnetic torque

Relation (B.11) gives the expression of the electromagnetic torque.

$$T_e=\frac{3}{2}p\cdot(\Psi_{sd}\cdot i_{sq}-\Psi_{sq}\cdot i_{sd})=\frac{3}{2}p\left[L_{sd}-L_{sq}\right]i_{sd}i_{sq}+\frac{3}{2}pM_{sr}I_{rd}i_{sq} \tag{B.11}$$

1.2.3. Mechanical equation

Relation (B.12) gives the expression of the mechanical equation.

$$\frac{J}{p}\frac{d\omega}{dt}=T_e-\frac{f_L}{p}\omega-T_L \tag{B.12}$$

2. α-β modeling of the Salient Synchronous Machine

The expression of the stator flux is given in the relation (B.13). It is expressed as function of the stator inductance matrix, stator currents, mutual inductance matrix and the rotor current. Since the machine is salient, the inductances are position dependent, [62], [111].

$$[\Psi_{sabc}] = [L_{sabc}(\theta)] \cdot [i_{sabc}] + [m_{sr}(\theta)] \cdot I_{rd} \tag{B.13}$$

Where,

$$[L_{sabc}(\theta)] = \begin{bmatrix} L_0 & M_0 & M_0 \\ M_0 & L_{s0} & M_0 \\ M_0 & M_0 & L_0 \end{bmatrix} + L_2 \cdot \begin{bmatrix} \cos(2\theta) & \cos(2\theta - 2\pi/3) & \cos(2\theta + 2\pi/3) \\ \cos(2\theta - 2\pi/3) & \cos(2\theta + 2\pi/3) & \cos(2\theta) \\ \cos(2\theta + 2\pi/3) & \cos(2\theta) & \cos(2\theta - 2\pi/3) \end{bmatrix} \tag{B.14}$$

And,

$$[m_{sr}(\theta)] = M_{sr} \cdot \begin{bmatrix} \cos(\theta) \\ \cos(\theta - 2\pi/3) \\ \cos(\theta + 2\pi/3) \end{bmatrix} \tag{B.15}$$

From the 3-phase electrical equations (relation B.8), and after a Clark transformation, the (α-β) electrical equations are,

$$[V_{s\alpha\beta}] = R \cdot [i_{s\alpha\beta}] + \frac{d}{dt}([\Psi_{s\alpha\beta}]) \tag{B.16}$$

Where,

$$[\Psi_{s\alpha\beta}] = [L_{s\alpha\beta}(\theta)] \cdot [i_{s\alpha\beta}] + [m_{sr\alpha\beta}(\theta)] \cdot I_{rd} \tag{B.17}$$

$$[L_{s\alpha\beta}(\theta)] = \begin{bmatrix} L_0 - M_0 + \frac{3}{2}L_2\cos(2\theta) & \frac{3}{2}L_2\sin(2\theta) \\ \frac{3}{2}L_2\sin(2\theta) & L_0 - M_0 - \frac{3}{2}L_2\cos(2\theta) \end{bmatrix} = \begin{bmatrix} L_1 + \frac{3}{2}L_2\cos(2\theta) & \frac{3}{2}L_2\sin(2\theta) \\ \frac{3}{2}L_2\sin(2\theta) & L_1 + \frac{3}{2}L_2\cos(2\theta) \end{bmatrix} \tag{B.18}$$

$$[m_{sr\alpha\beta}(\theta)] = M_{sr} \cdot \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{B.19}$$

The (α-β) electrical equation are then rewritten as follows,

$$\begin{aligned} v_{s\alpha\beta} = {} & R_s[i_{s\alpha\beta}] + L_1\frac{d[i_{s\alpha\beta}]}{dt} + 3L_2\omega\begin{bmatrix} -\sin(2\theta) & \cos(2\theta) \\ \cos(2\theta) & \sin(2\theta) \end{bmatrix}[i_{s\alpha\beta}] \\ & + \frac{3}{2}L_2\begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix}\frac{d[i_{s\alpha\beta}]}{dt} + M_{sr}I_{rd}\omega\begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \end{aligned} \tag{B.20}$$

As it will be discussed afterwards, the identified stator inductances are the d-q L_{sd} and L_{sq} inductances. Their relationship with the inductances L_1 and L_2 is then,

$$L_1 = \frac{L_{sd} + L_{sq}}{2} \quad ; \quad L_2 = \frac{L_{sd} - L_{sq}}{3} \tag{B.21}$$

Finally, the expression of the electromagnetic torque is,

$$T_e = \frac{3}{2}pM_{sr}I_{rd}\left(-i_{s\alpha}\cdot\sin(\theta) + i_{s\beta}\cdot\cos(\theta)\right) + \frac{9}{2}pL_2\cdot i_{s\alpha}\cdot i_{s\beta}\cdot\cos(2\theta) - \frac{9}{4}pL_2\cdot(i_{s\alpha}{}^2 - i_{s\beta}{}^2)\cdot\sin(2\theta) \tag{B.22}$$

3. Parameter identification

3.1. Identification of the stator resistance R_s

The identification of the stator resistance can be made either by measuring directly the resistance of the stator winding or by applying a DC voltage and then measuring the current. In our case the first method has been used. The obtained value is 10.5Ω.

3.2. Identification of the mutual inductance M_{sr}

The identification of the mutual inductance is made when the implemented SSM runs as a no-load generator. The amplitude of the measured Back-EMF, the rotating speed and the rotor current are then measured. From these data, the mutual inductance M_{sr} is extracted according to the following relation.

$$M_{sr} = \frac{V_{bmf_{\max}}}{\omega \cdot I_{rd}} \tag{B.23}$$

During the whole tests (simulations and experimentations), the excitation rotor current is maintained to 1.5A. With this value, the obtained Mutual inductance is equal to 0.85H

3.3. Identification of the stator inductances L_{sd} and L_{dq}

The identification of the stator inductances is based on the method deeply investigated in [110]. This method is performed via a hysteresis based current control of the dq components of the stator current vector. With a locked rotor, the hysteresis based current controller allows the application of step reference to one component (e.g q component) of the stator current vector while the current component of the other axis (e.g. d component) is set to zero vice versa. From the sampled current responses, the dq-based stator inductances are then deduced. Note that, with the proposed identification method, the saturation is neglected and the cross saturations between the d and q axis are not considered. Figure B.4 gives an example of the current responses in both cases.

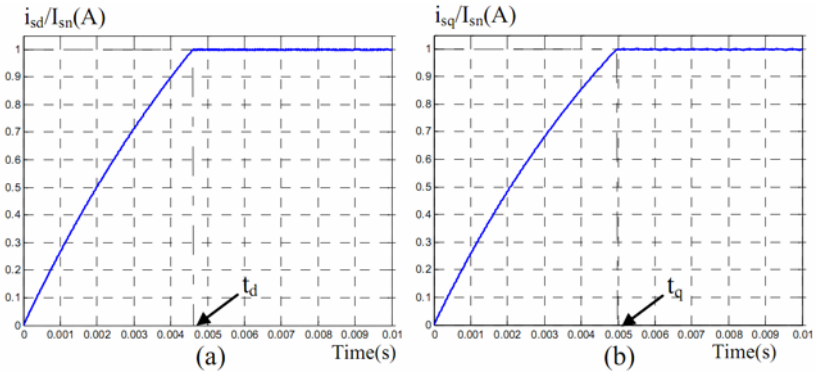


Figure B.4: (a) Response to a step $i_{sd}^*=I_{sn}$ and $i_{sq}^*=0A$
(b) Response to a step $i_{sq}^*=I_{sn}$ and $i_{sd}^*=0A$
(Figure extracted from [110])

From the measured settling times t_d and t_q, and depending on the DC-link voltage V_{DC}, the stator resistance R_s and the value of the current I_{sn}, the stator inductances can be calculated as follow,

$$L_{sd} = \frac{-R_s \cdot t_d}{\log\left(1 - \frac{3R_s \cdot I_{sn}}{2V_{DC}}\right)} \quad ; \quad L_{sq} = \frac{-R_s \cdot t_q}{\log\left(1 - \frac{\sqrt{3}R_s \cdot I_{sn}}{V_{DC}}\right)} \tag{B.24}$$

Having in mind that the stator inductances vary depending on the stator current, the average numerical values are then calculated and used by the implemented EKF. In our case they are: L_{sd}=0.245H and L_{sq}=0.229H.

Appendix C

Tuning of the current and speed regulators

1. Tuning of the stator current d-q PI regulators

To start with, it is necessary to recall the d-q based electrical equations of the implemented SSM (relation (C.1)).

$$\begin{aligned} v_{sd} &= R_s \cdot i_{sd} + L_{sd} \cdot \frac{d(i_{sd})}{dt} - \omega \cdot L_{sq} \cdot i_{sq} \\ v_{sq} &= R_s \cdot i_{sq} + L_{sq} \frac{d(i_{sq})}{dt} + \omega \cdot L_{sd} \cdot i_{sd} + \omega \cdot M_{sr} \cdot I_{rd} \end{aligned} \tag{C.1}$$

Relations (C.2) and (C.3) are the equivalent d and q transfer functions.

$$i_{sd} = \frac{V_{sd} + \omega \cdot L_{sq} \cdot i_{sq}}{R_s + L_{sd} \cdot s} \tag{C.2}$$

$$i_{sq} = \frac{V_{sq} - \omega \cdot L_{sd} \cdot i_{sd} - \omega \cdot M_{sr} \cdot I_{rd}}{R_s + L_{sq} \cdot s} \tag{C.3}$$

These d-q currents are controlled with the help of PI regulators, whom the transfer functions are expressed as follows,

$$PI_d = K_{pd} + \frac{K_{id}}{s} \tag{C.4}$$

$$PI_q = K_{pq} + \frac{K_{iq}}{s} \tag{C.5}$$

Figure C.1 highlights a simplified block diagram of the control closed loops of i_{sd} and i_{sq} stator currents.

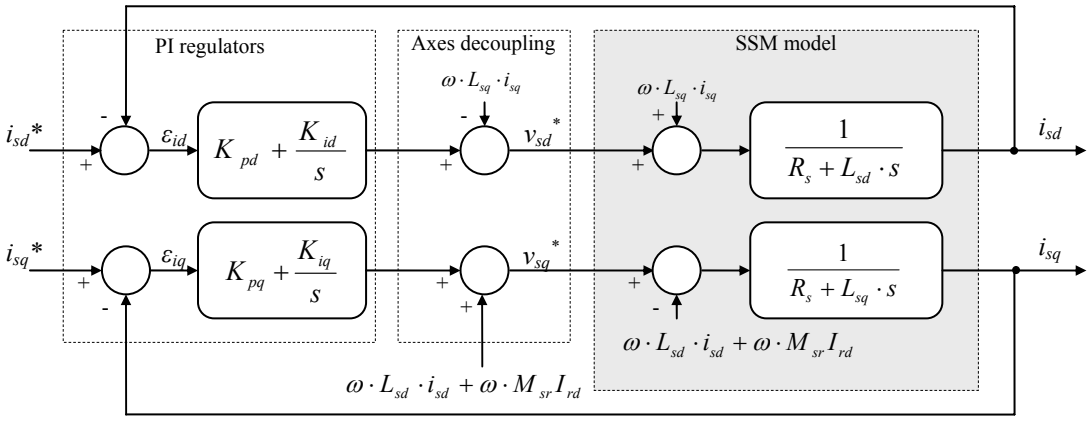


Figure C.1: Current control block diagram

After having decoupled the d-q axes, the global d-q transfer functions are then extracted,

$$\frac{i_{sd}}{i_{sd}^*} = \frac{(K_{pd} + \frac{K_{id}}{s}) \cdot \frac{1}{R_s + L_{sd} \cdot s}}{1 + (K_{pd} + \frac{K_{id}}{s}) \cdot \frac{1}{R_s + L_{sd} \cdot s}} = \frac{\frac{1}{s} \cdot \frac{K_{pd}}{L_{sd}} \cdot \frac{s + \frac{K_{id}}{K_{pd}}}{s + \frac{R_s}{L_{sd}}}}{1 + \frac{1}{s} \cdot \frac{K_{pd}}{L_{sd}} \cdot \frac{s + \frac{K_{id}}{K_{pd}}}{s + \frac{R_s}{L_{sd}}}} \tag{C.6}$$

$$\frac{i_{sq}}{i_{sq}^*} = \frac{(K_{pq} + \frac{K_{iq}}{s}) \cdot \frac{1}{R_s + L_{sq} \cdot s}}{1 + (K_{pq} + \frac{K_{iq}}{s}) \cdot \frac{1}{R_s + L_{sq} \cdot s}} = \frac{\frac{1}{s} \cdot \frac{K_{pq}}{L_{sq}} \cdot \frac{s + \frac{K_{iq}}{K_{pq}}}{s + \frac{R_s}{L_{sq}}}}{1 + \frac{1}{s} \cdot \frac{K_{pq}}{L_{sq}} \cdot \frac{s + \frac{K_{iq}}{K_{pq}}}{s + \frac{R_s}{L_{sq}}}} \tag{C.7}$$

The tuning of the PI regulator gains is based on the poles compensation. This consists in making K_{id}/K_{pd} and K_{iq}/K_{pq} respectively equal to R_s/L_{sd} and R_s/L_{sq}. With this assumption, the final transfer functions of the control closed loops are,

$$\frac{i_{sd}}{i_{sd}^*} = \frac{1}{1 + \frac{L_{sd}}{K_{pd}} \cdot s} = \frac{1}{1 + T_{isd} \cdot s} \tag{C.8}$$

$$\frac{i_{sq}}{i_{sq}^*} = \frac{1}{1 + \frac{L_{sq}}{K_{pq}} \cdot s} = \frac{1}{1 + T_{isq} \cdot s} \tag{C.9}$$

These are first order transfer functions and T_{isd} and T_{isq} are respectively the time constants of i_{sd} and i_{sq} current control closed loops. These constants can then be set according to the desired dynamic performances. The values of the PI regulators gains are,

$$K_{pd} = \frac{L_{sd}}{T_{isd}} \quad ; \quad K_{id} = \frac{R_s}{T_{isd}} \tag{C.10}$$

$$K_{pq} = \frac{L_{sq}}{T_{isq}} \quad ; \quad K_{iq} = \frac{R_s}{T_{isq}} \tag{C.11}$$

2. Tuning of the P-PI speed regulator

The speed control strategy that has been implemented in the proposed work is based on P-PI speed regulator. This controller is characterized by two closed loops: an inner closed loop and an outer closed loop. The first one is based on a proportional regulator (P) which aims to impose the system poles at the user-defined location. The outer loop is based on a PI regulator which ensures the desired static and dynamic speed responses. Figure C.2 summarizes the principle of this regulator.

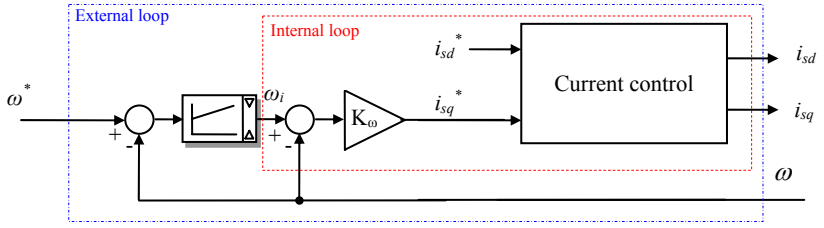


Figure C.2: Speed control block diagram

Assuming that the current controller is well-established and its dynamic is much faster than the speed controller, it is possible to make a link between the electrical speed ω and the stator currents i_{sd} and i_{sq} according to the following relation:

$$\omega = \frac{3}{2} \cdot p^2 \cdot \frac{1}{f_L + J \cdot s} \cdot \left(\left[L_{sd} - L_{sq} \right] \cdot i_{sd} \cdot i_{sq} + M_{sr} \cdot I_{rd} \cdot i_{sq} - \frac{2}{3 \cdot p^2} \cdot T_L \right) \tag{C.12}$$

Assuming that in case of a flux oriented control application, the direct current i_{sd} is set to zero. Also, the effect of the load torque T_L and the friction coefficient f_L are neglected. This leads to the following transfer function between ω and i_{sq},

$$\frac{\omega}{i_{sq}} = \frac{3}{2} \cdot p^2 \cdot \frac{M_{sr} \cdot I_{rd}}{J \cdot s} \tag{C.13}$$

The internal speed control loop is made up of a proportional controller, which is aimed to impose the controlled system poles at the desired locations. The corresponding transfer function is expressed in relation (C.14). This relation has been obtained with the consideration of the i_{sq} transfer function (relation (C.9)).

$$\frac{\omega}{\omega_i} = \frac{\dfrac{1.5 K_\omega p^2 M_{sr} I_{rd}}{J T_{isq}}}{s^2 + \dfrac{1}{T_{isq}} \cdot s + \dfrac{1.5 K_\omega p^2 M_{sr} I_{rd}}{J T_{isq}}} \tag{C.14}$$

When assuming that,

$$K_\omega = \frac{J}{6 T_{isq} p^2 M_{sr} I_{rd}} \tag{C.15}$$

The final transfer function of the internal loop can be expressed as a second order system with double-real poles according to relation (C.16).

$$\frac{\omega}{\omega_i} = \frac{\dfrac{1}{4 \cdot T_{isq}^2}}{(s + \dfrac{1}{2 T_{isq}})^2} \tag{C.16}$$

The external speed control feedback loop is performed via a PI controller in order to ensure zero steady-state speed error and to impose the response dynamic. The transfer function of the PI speed controller is given by,

$$PI_w = K_{pw} + \frac{K_{iw}}{s} \tag{C.17}$$

Imposing the ratio K_{iw}/K_{pw} equal to $1/2T_{isq}$, the final transfer function of the external speed loop is:

$$\frac{\omega}{\omega^*} = \frac{\dfrac{K_{pw}}{4 T_{isq}^2}}{s^2 + \dfrac{1}{2 T_{isq}} s + \dfrac{K_{pw}}{4 T_{isq}^2}} \tag{C.18}$$

Here again a second order transfer function has been obtained. The value of K_{pw} is determined with regards to the expected overshoot and settling time.

Appendix D

Extended Kalman Filter for AC Drive Sensorless Speed Controller FPGA-Based solution or DSP-Based solution

1. Problem statement

The performances of the developed FPGA-based sensorless speed controller have been compared to a fully software solution based on a DSP, [68]. The aim is to study and discuss the influence of the necessary execution time of the whole sensorless algorithm on the control bandwidth and quality. In fact the use of a software solution allows a long execution time which introduces significant time delays in the controller closed-loop. Consequently, in case of a high speed AC drive where high control performances are required, the controller bandwidth and quality are downgraded. This is mainly due to the DSP fixed architecture leading to serialize the treatment.

In contrast, the use of a hardware solution such as FPGA ensures the possibility to implement a fully user-defined parallel architecture and then provides parallel treatment. In this case the execution time is dramatically reduced which leads to a quasi-instantaneous control (see chapter 6). This particularity provides the possibility to implement high bandwidth and high quality sensorless controllers.

In order to illustrate and quantify the influence of the execution time, simulation and experimental tests have been achieved. In the following subsections, a comparison in terms of dynamic behavior is achieved. The used digital controller consists then on a Xilinx Virtex_2P FPGA in the one hand and on a TI TMSF2808 DSP (100MHz, 32Bit, 12-bit ADC, 2x16-bit multiplier, 16Ko RAM memory) [64] in the other hand.

To start with, let us present the adopted timing diagram in both cases, Figure D.1 In the first case (Figure D.1(a)) and having in mind that the whole algorithm including the presented EKF equations have been implemented in the same DSP device, the obtained execution time is evaluated to 66µs. Consequently, the necessary sampling period of the digital treatment has been set to 100µs, which is the same as the PWM switching period.

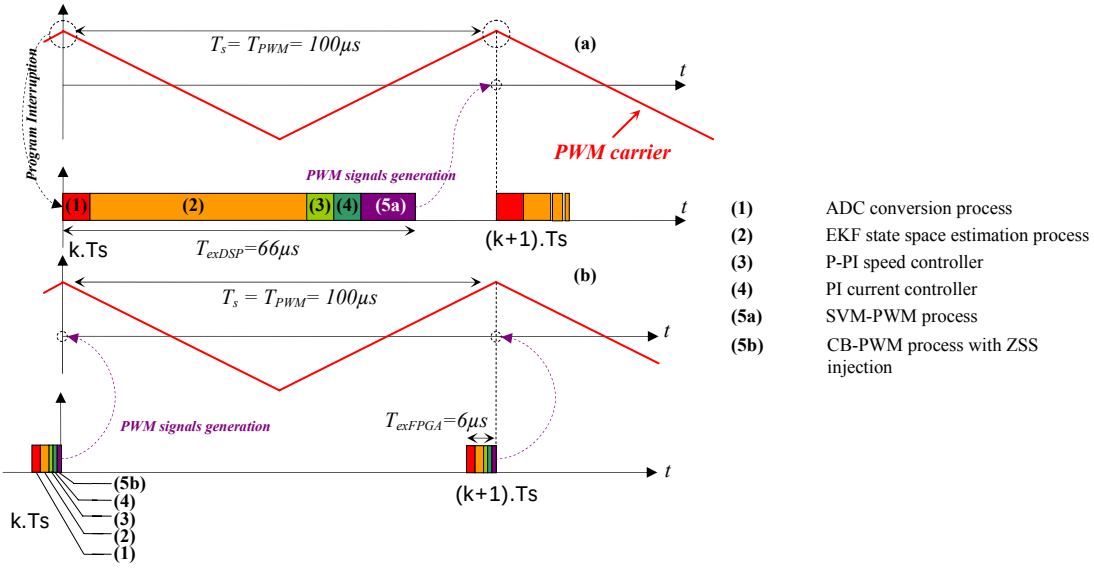


Figure D.1: Timing diagram
(a): case of a DSP sensorless controller (b): case of an FPGA sensorless controller

As far as the hardware FPGA based solution is concerned and when operating at a 50MHz clock frequency, the corresponding execution time is evaluated to 6µs which is eleven times less then the one obtained with the software solution. This fast treatment ensures an important flexibility regarding the timing diagram. For example, as shown in Figure D.1(b), the short execution time gives the possibility to launch the controller right before the PWM carrier peak so as to update the necessary PWM signals at the same switching period which ensures a quasi-instantaneous controller.

Figure D.2 shows the synoptic of the normalized speed sensorless control closed-loop. In order to illustrate the impact of the execution time on the dynamic behavior, a time delay is introduced and the sensorless controller tuning (EKF, speed and current regulators) remains the same in both cases as well as the motor load conditions.

As it can be noticed in this figure, the introduced time delay has an impact on the estimated speed and position. For the latter, this time can be considered as a phase shift and then influences Park transformations. Consequently, erroneous values are processed which affects the behavior of speed and current controllers. In other words, the control bandwidth is then influenced.

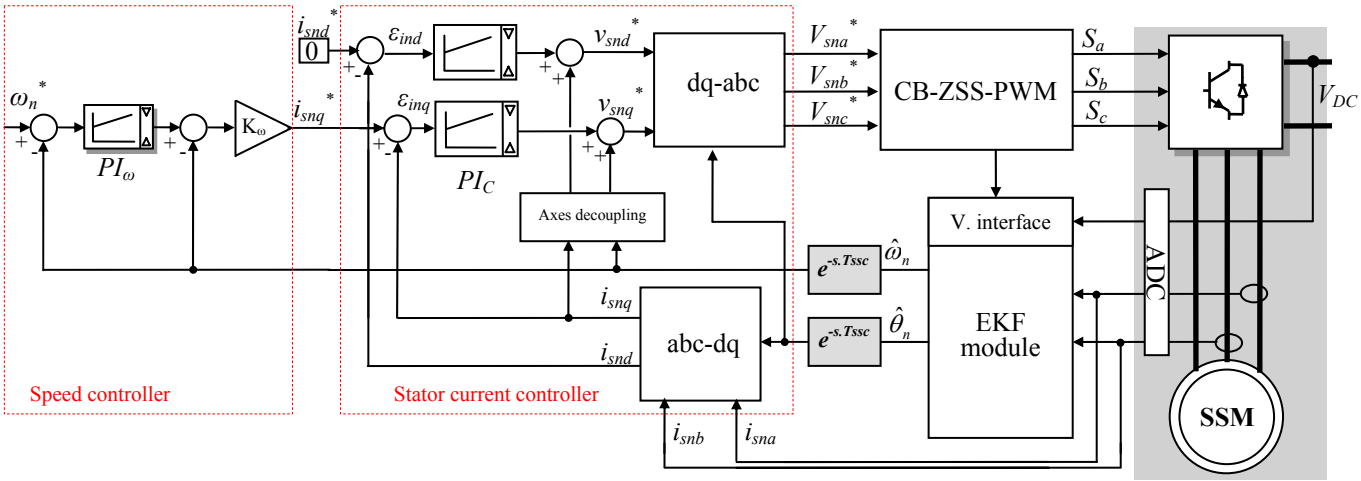


Figure D.2: Sensorless speed control closed loop

2. Influence of the time delay on Park transformations

Relation (D.1) presents the equation of the *abc_dq* transformation module, where *[R(θ)]* is the rotation matrix. This module generates the d-q stator currents from the measured currents and the estimation position with the consideration of the time delay (phase shift). The latter is set to $T_{ssc}=T_s=T_{pwm}=100\mu s$ for the DSP-based solution and to $T_{ssc}=T_{exFPGA}=6\mu s$ for the FPGA one.

$$\begin{bmatrix} i_{sndq} \end{bmatrix} = \begin{bmatrix} \cos(\omega T_{ssc}) & -\sin(\omega T_{ssc}) \\ \sin(\omega T_{ssc}) & \cos(\omega T_{ssc}) \end{bmatrix} \cdot [R(\theta)] \cdot \begin{bmatrix} 1 & 0 \\ 1/\sqrt{3} & 2/\sqrt{3} \end{bmatrix} \cdot \begin{bmatrix} i_{sna} \\ i_{snb} \end{bmatrix} \quad \text{(D.1)}$$

From this relation, it can be noticed that when the operating speed increases, erroneous values are processed. For instance, in case of a 1256 rd/s electrical speed (corresponding to a 4000 rpm mechanical speed, case of a high speed AC drive, Table D.2), the processed currents are written as,

$$\begin{bmatrix} i_{sndq} \end{bmatrix}_{DSP} = \begin{bmatrix} 0.992 & -0.1256 \\ 0.1256 & 0.992 \end{bmatrix} \cdot \begin{bmatrix} i_{sndq} \end{bmatrix}_{\text{withoutdelay}} \quad \text{(D.2)}$$

$$\begin{bmatrix} i_{sndq} \end{bmatrix}_{FPGA} = \begin{bmatrix} 0.9999 & -0.0075 \\ 0.0075 & 0.9999 \end{bmatrix} \cdot \begin{bmatrix} i_{sndq} \end{bmatrix}_{\text{withoutdelay}} \quad \text{(D.3)}$$

As for the *dq_abc* transformation module, relation (D.4) presents the corresponding equation.

$$\left[V^*_{snabc}\right]=\frac{2}{3}\cdot\begin{bmatrix}\cos(\omega T_{ssc}) & \cos(\omega T_{ssc}-\frac{2\pi}{3}) & \cos(\omega T_{ssc}+\frac{2\pi}{3})\\ \cos(\omega T_{ssc}+\frac{2\pi}{3}) & \cos(\omega T_{ssc}) & \cos(\omega T_{ssc}+\frac{4\pi}{3})\\ \cos(\omega T_{ssc}-\frac{2\pi}{3}) & \cos(\omega T_{ssc}-\frac{4\pi}{3}) & \cos(\omega T_{ssc})\end{bmatrix}\cdot\begin{bmatrix}\cos\theta & -\sin\theta\\ \cos(\theta-2\pi/3) & -\sin(\theta-2\pi/3)\\ \cos(\theta-4\pi/3) & -\sin(\theta-4\pi/3)\end{bmatrix}\cdot\left[V^*_{sndq}\right] \tag{D.4}$$

Here again the influence of the introduced phase delay has an increasing impact with regards to the operating speed. In case of a 1256 rd/s operating speed, the obtained 3-phase voltages are expressed in relations (D.5) and (D.6),

$$\left[V^*_{sabc}\right]_{DSP}=\frac{2}{3}\cdot\begin{bmatrix}0.992 & -0.387 & -0.604\\ -0.604 & 0.992 & -0.387\\ -0.387 & -0.604 & 0.992\end{bmatrix}\cdot\left[V^*_{sabc}\right]_{without\ \ delay} \tag{D.5}$$

$$\left[V^*_{sabc}\right]_{FPGA}=\frac{2}{3}\cdot\begin{bmatrix}0.9999 & -0.493 & -0.506\\ -0.506 & 0.9999 & -0.493\\ -0.493 & -0.506 & 0.9999\end{bmatrix}\cdot\left[V^*_{sabc}\right]_{without\ \ delay} \tag{D.6}$$

Thus, in case of high speed operating condition, the introduced error is less important in the case of FPGA solution than in the case of DSP solution.

3. Influence of the time delay on the controller bandwidth

As mentioned before, the introduced errors on park transformations and the time delay introduced in the speed controller have a significant impact on the behavior of the corresponding regulators.

The speed controller processes the i_{sndq}* reference for the current controller. With the consideration of the time delay, the corresponding mathematical equation is then

$$\left[i^*_{sndq}\right]=\begin{bmatrix}0\\ K_\omega\cdot PI_w(s)\cdot(\omega^*_n-\omega_n\cdot e^{-s.T_{ssc}})-K_\omega\cdot\omega_n\cdot e^{-s.T_{ssc}}\end{bmatrix} \tag{D.7}$$

The dq-voltage references generated by the PI-based regulator (with the dq axes decoupling assumption) are expressed in relation (D.8). According to the latter, the influence of time and phase delay can be stressed.

$$\left[V^*_{sndq}\right]=\left[PI_{cdq}(s)\right]\cdot\left[\varepsilon_{indq}\right]+\begin{bmatrix}0 & -L_{sq}\\ L_{sd} & 0\end{bmatrix}\cdot\left[i_{sndq}\right]\cdot\omega_n\cdot e^{-s.T_{ssc}}+\begin{bmatrix}0\\ M_{sr}\cdot I_{rnd}\end{bmatrix}\cdot\omega_n\cdot e^{-s.T_{ssc}} \tag{D.8}$$

Figure D.3 highlights the obtained simulations and experimental results. These results are obtained for a low speed 1500 rpm AC Drive, Table D.1. It can be noticed, from the frequency behavior (Figure D.3 (a) and (b)) and the step response (Figure D.3 (c) and (d)) that for this low operating speed, the influence of the time delay is of less importance since the error in Park transformations and the delay in the speed controller are negligible.

This is not the case for a high speed AC drive (Table D.2), [65], where a notable difference on the dynamic behavior is indicated. Figure D.4 presents the obtained simulation results in this case. The fast execution time ensured by the FPGA solution provides a greater bandwidth than the DSP one.

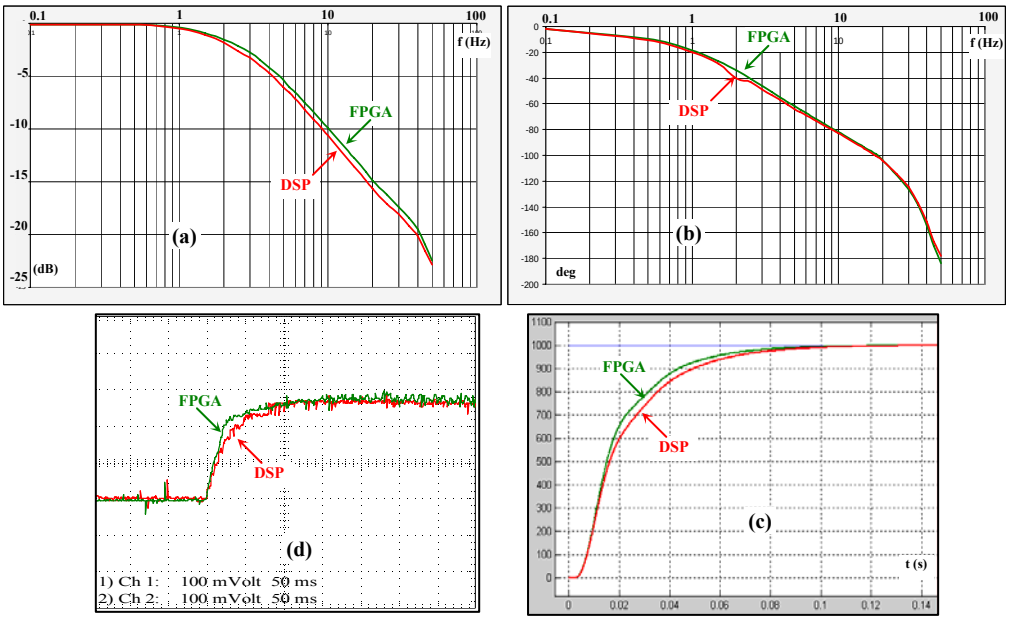


Figure D.3: Evaluation of the control bandwidth – Case of low speed AC drive
(a) : Frequency response (magnitude) –Simulation **(b)** : Frequency response (phase) –Simulation **(c)** : Step response (1000 rpm) –Simulation **(d)** : Step response (1000 rpm) –Experimentation

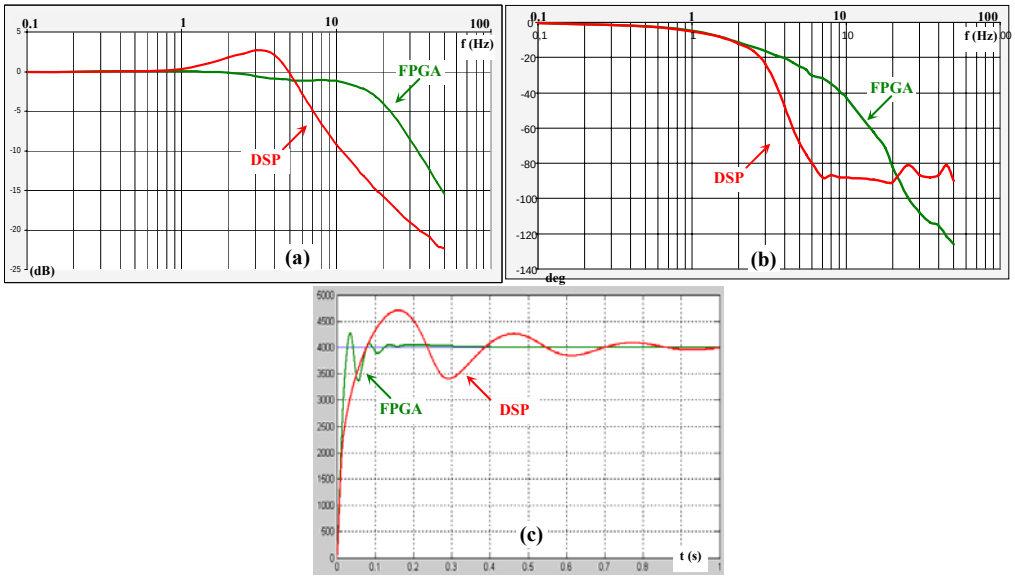


Figure D.4: Evaluation of the control bandwidth – Case of high speed AC drive
(a) : Frequency response (magnitude)– Simulation **(b)** : Frequency response (phase) –Simulation
(c) : Step response (4000 rpm) –Simulation

Table D.1: Synchronous Machine Parameters
Low speed AC Drive
0.8 KVA, 220V,1.5A, 50 Hz,
3 Phases, Y connection, 2 pole pairs

Stator resistance $R_s = 10.5\ \Omega$	Rotor resistance $R_r = 62.5\ \Omega$
d axis stator inductance $L_{sd} = 0.245$ H	Mutual inductance $M_{sr} = 0.85$ H
q axis stator inductance $L_{sq} = 0.229$ H	Nominal stator current $I_{nom} = 1.52$ A

Table D.2: Main Starter Generator Parameters
High speed AC Drive
200 KVA, 230V, 290A, 380 Hz,
3 Phases, Y connection, 3 pole pairs

Stator resistance $R_s = 16$ mΩ	Rotor resistance $R_r = 0.34\ \Omega$
d axis stator inductance $L_{sd} = 0.45$ mH	Mutual inductance $M_{sr} = 3.6$ mH
q axis stator inductance $L_{sq} = 0.35$ mH	Nominal stator current $I_{nom} = 290$ A

Appendix E

FPGA-based matrix multiplication and inversion

1. FPGA-based 2-matrix multiplier

In the case of the developed non-optimized EKF observer, matrix multiplications are processed. As discussed in chapter 6, the dimensions of processed matrices have been extended to 4x4 matrices which have allowed the factorization of matrix multiplications. In this section, we are presenting the FPGA architecture of the implemented 2-matrix multiplier.

To start with, let's define *M1* and *M2* as the two 4x4 input matrices and *T* the output 4x4 matrix. They are related as expressed in relation (E.1).

$$\forall i, j \quad T_{ij} = \sum_{k=1}^{m_d} M1_{ik} \cdot M2_{kj} \;\; ; \;\; m_d = 4 \tag{E.1}$$

The output matrix is processed after a set of 64 multiplications and 48 additions. In order to reduce the needed FPGA resources, it has been decided to factorize the multiplications and the additions. Then only four 22-bit multipliers and three 22-bit adders are used. With this assumption, the treatment is consequently serialized. The corresponding latency is then equal to 48 instead of 4 without factorization. Figure E.1 presents the designed FPGA architecture.

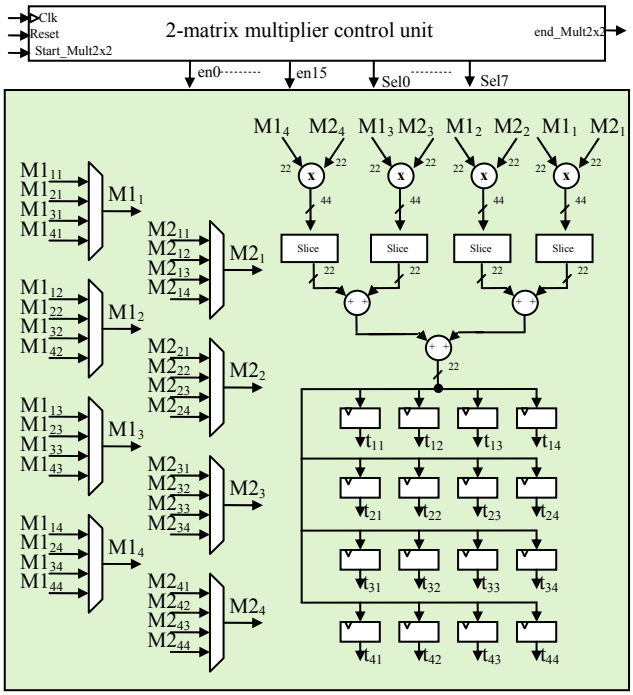


Figure E.1: FPGA architecture of the 2-matrix multiplier

2. FPGA-based 3-matrix multiplier

When it come the multiplication of three matrices, the development of the corresponding FPGA architecture is based on the factorization of the previously discussed 2-matrix multiplier. Additional matrix multiplexers and a matrix register have been introduced. Figure E.2 presents the designed FPGA architecture where A, B and C are the input matrices and O is the output matrix. With this

configuration, the total latency of the 3-matrix multiplier is equal to 96 instead of 8 without factorization.

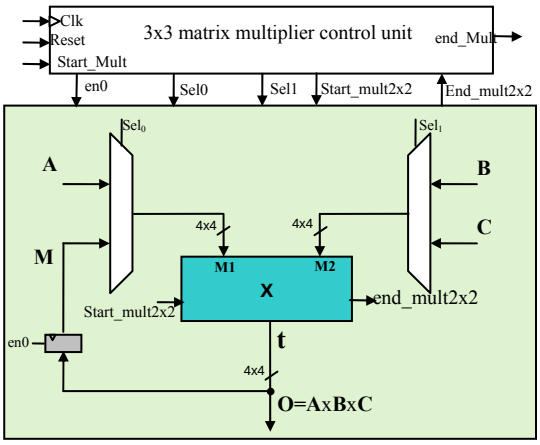


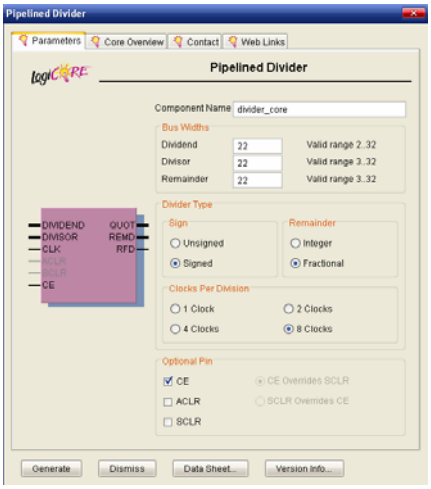
Figure E.2: FPGA architecture of the 3-matrix multiplier

3. Matrix inversion

The implemented 2x2 matrix inversion module is based on relation E.2.

$$[A]^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{E.2}$$

In the case of having chosen the Xilinx FPGA solutions, the inversion of the matrix determinant is made using the Pipelined-Divider IP [2]. Figure E.3(a) presents the corresponding configuration wizard.



(a)

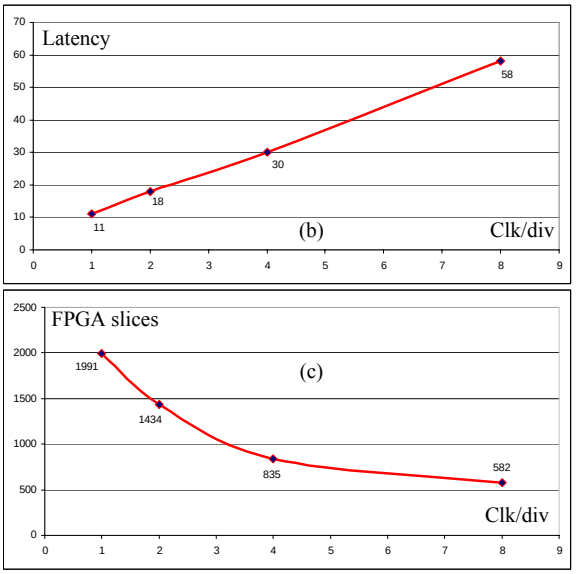


Figure E.3: Xilinx Pipelined Divider IP

The latency of the divider and the consumed FPGA resources are all conditioned by the chosen clock per division value (1, 2, 4 or 8). These values mean that the input data is sampled at each 1st, 2nd, 4th of 8th clock rising edge. Figure E.3(b) and Figure E.3(c) present the relationship between the divider latency, consumed resources and the clock per division value. In the case of the developed application, the time/area performances (presented in chapter 6) have been analyzed in the case of clk/div set to 8.

Bibliography

[1] In-Stat market analyst website: www.instat.com

[2] Xilinx on-line documentation. Available in www.xilinx.com

[3] Altera on-line literature. Available in www.altera.com

[4] Actel on-line documentation. Available in www.actel.com

[5] J. J. Rodriguez-Andina, M. J. Moure, M. D. Valdes, "Features, design tools, and application domains of FPGAs", *IEEE Transactions On Industrial Electronics*, vol. 54, no. 4, pp. 1810–1823, August 2007.

[6] S. Hauck, A. Dehon, "Reconfigurable Computing, the theory and practice of FPGA-based computing", University of Pennsylvania, Philadelphia and Pennsylvania USA, Elsevier and Morgan Kauffman publishers, 2008.

[7] I. Grout, "Digital Systems Design with FPGAs and CPLDs", Elsevier and Newnes, 2008.

[8] Katarzyna Leijten-Nowak, "Template-Based Embedded Reconfigurable Computing", PhD thesis, Library Technische Universiteit Eindhoven, Philips Research Laboratories, 2004

[9] Wai-Kai Chen, "The VLSI Handbook, second edition", University of Illinois Chicago USA, CRC Press, 2007.

[10] Actel Fusion FPGA product specification. www.actel.com/products/fusion/

[11] Actel Axcelerator FPGA product specification. www.actel.com/products/axcelerator/

[12] Xilinx Spartan-6 FPGA product specification. www.xilinx.com/products/spartan6/

[13] Xilinx Virtex FPGA product specification. www.xilinx.com/products/virtex/

[14] Xilinx Virtex-6 FPGA product specification. www.xilinx.com/products/virtex6/

[15] Altera Cyclone-4 FPGA product specification. www.altera.com/products/devices/cyclone-iv

[16] Altera Stratix-4 FPGA product specification. www.altera.com/products/devices/stratix-iv

[17] E. Monmasson, Y. A. Chapuis, "Contributions of FPGAs to the control of Electrical systems, a Review", *IEEE Industrial Electronics Society Newsletter*, vol. 49, no. 4, pp. 8-15, December 2002.

[18] L. Idkhajine, M-W. Naouar, E. Monmasson, A. Prata, "Fully FPGA-Based System on Chip Solution for Current Control of AC Machine", *In Proceedings EPE'2007 Conference*, CD-ROM, Aalborg, Denmark.

[19] K. Bouallaga, L. Idkhajine, E. Monmasson, A. Prata, "Demodulation Methods on Fully FPGA-Based System for Resolver Signals Treatment ", *In Proceedings EPE'2007 Conference*, CD-ROM, Aalborg, Denmark.

[20] L. Idkhajine, M-W. Naouar, E. Monmasson, A. Prata, "Standard FPGA-based or Full FPGA-based Controllers for Electrical systems, two viable solutions". *In Proceedings ISIE'2007 Conference*, pp. 2332-2337, Vigo, Spain.

[21] L. Idkhajine, E. Monmasson, A. Maalouf, "AC drive System on Chip Controller with non-linearity errors compensation". *In Proceedings IECON'2008 Conference*, pp. 2381-2386, Orlando, Florida, USA.

[22] L. Idkhajine, A. Prata, E. Monmasson, M-W. Naouar, "System on Chip Controller for Electrical Actuator". *In Proceedings ISIE'2008 Conference*, pp. 2481-2486, Cambridge, UK.

[23] L.Idkhajine, E. Monmasson, M-W. Naouar, A.Prata, K.Bouallaga, "Fully integrated FPGA-based controller for synchronous motor drives", *IEEE Transactions On Industrial Electronics*, vol. 56, n°. 10, pp. 4006-4017, October 2009.

[24] E. Monmasson, M. W. Naouar, L. Idkhajine, “FPGA-based Controllers for Power Electronics and Drive Applications”, *IEEE Industrial Electronics Magazine,* Accepted for publication.

[25] E. Monmasson, L. Idkhajine, I. Bahri, M.W. Naouar, L. Charaabi, “Design methodology and FPGA-based controllers for Power Electronics and drive applications”, *In Proceedings ICIEA’2010 Conference,* pp. 2328-2338, Taichung, Taiwan.

[26] K. Eshraghian, “SoC Emerging Technologies”, *IEEE Proceedings*, vol. 94, n° 6, June 2006, pp. 1197 – 1213.

[27] A.K. Ben-Salem, S. Ben-Othman, S. Ben-Saoud, N. Litayem, “Servo Drive System Based on Programmable SoC Architecture”, *In Proceedings IECON’2009 Conference*, pp. 2961-2966, Porto, Portugal.

[28] A. Das , K. Banerjee, “Fast Prototyping of a Digital PID Controller on a FPGA based Soft-Core Microcontroller for precision control of a Brushed DC Servo Motor”, *In Proceedings IECON’2009 Conference*, pp. 2825-2830, Porto, Portugal.

[29] G. Martin, “Overview of the MPSoC Design Challenge". *In Proceedings DAC’2006 Conference*, pp. 274-279, Yokohama, Japan

[30] A. Tjondronugroho, A. Al-Anbuky, S. Round, R. Duke “Evaluation of DSP and FPGA based digital controllers for a single-phase PWM inverter”, *In Proceedings AUPEC’2004 Conference*, Brisbane, Australia

[31] E.J. Bueño,, Ã. Hernandez, F.J. Rodriguez, C. Giron and R. Mateos and S. Cóbreces,. “A DSP and FPGA based industrial control with high-speed communication interfaces for grid converters applied to distributed power generation systems”, *IEEE Transactions On Industrial Electronics*, vol. 56, n° 3, pp. 654-669, March 2009,

[32] A. Fratta, G. Griffero, and S. Nieddu, “Comparative analysis among DSP and FPGA-based control capabilities in PWM power converters,” *In Proceedings IECON’2004 Conference*, pp.257-262, Busan, Korea**.**

[33] S. N. Murthy, W. Alvis, R. Shirodkar, K. Valavanis, W. Moreno, “Methodology for implementation of unmanned vehicle control on FPGA using system generator”, *In Proceedings ICCDCS’2004 Conference*, CD-ROM.

[34] E. Monmasson, M. Cirstea, “FPGA design methodology for industrial control systems – A review,” *IEEE Transactions On Industrial Electronics*, vol. 54, no. 4, pp. 1824–1842, August 2007.

[35] J. S. Beeckler, W. J. Gross, “A Methodology for Prototyping Flexible Embedded Systems”, *In Proceedings CCECE’07 Conference,* CD-ROM

[36] M. W. Naouar, E. Monmasson, A. A. Naassani, I. Slama-Belkhodja, N. Patin, “FPGA-based current controllers for AC machine drives – A review,” *IEEE Transactions On Industrial Electronics*, vol. 54, no. 4, pp. 1907–1925, August 2007.

[37] Y. A. Chapuis, J. P. Blonde, F. Braun, “FPGA implementation by modular design reuse mode to optimize hardware architecture and performance of ac motor controller algorithm,” *In Proceedings EPE-PEMC’2004 Conference*, pp. 134–142, CD-ROM.

[38] A. Maalouf, L. Idkhajine, S. Le Ballois, E. Monmasson, “FPGA-based Sensorless Control of Brushless Synchronous Starter Generator for Aircraft Application”, *IET Electric Power Applications Journal*, Accepted for publication in 2011.

[39] R. Andraka, “A survey of CORDIC algorithms for FPGAs”, *In Proceedings ACM/SIGDA’1998 Conference*, pp. 191–200.

[40] D. Menard, O. Sentieys, “Automatic evaluation of the accuracy of fixed-point algorithms,” *In Proceedings ACM Des., Autom. and Test Eur. Conference, 2002,* pp. 529–535, CD-ROM.

[41] F. Zhengwei, J. E. Carletta, R. J. Veillette, “A methodology for FPGA-based control implementation”, *IEEE Transactions On Control Systems Technology*, vol. 13, no. 6, pp. 977–987, November 2005.

[42] T. Grandpierre, C. Lavrenne, Y. Sorel, “Optimized rapid prototyping for real-time embedded heterogeneous multiprocessor,” *In Proceedings CODES’1999, 7th International Workshop on*

Hardware/ Software Co-Design Conference, pp. 74-78.

[43] F. Ricci and H. Le-Huy, "An FPGA-based rapid prototyping platform for variable-speed drives", in *In Proceedings IECON'2002 Conference*, pp. 1156–1161, Sevilla Spain.

[44] M. J. Newman and D. G. Holmes, "Delta operator digital filters for high performance inverter applications", *IEEE Transactions On Power Electronics*, vol. 18, no. 1, pp. 447–454, January 2003.

[45] O. Lopez,, J. Alvarez, J. Doval-Gandoy, F.D. Freijedo, "Multilevel Multiphase Space Vector PWM Algorithm", *IEEE Transactions On Industrial Electronics*, vol. 55, no 5, pp 1933 – 1942, May 2008.

[46] B. J. Patella, A. Prodic, A. Zirger, and D. Maksimovic, "High-frequency digital controller IC for dc–dc converters", *IEEE Transactions On Power Electronics*, vol. 18, no. 1, pp. 438–446, January 2003.

[47] S.C. Huerta, A. de Castro, O. Garcia, J.A. Cobos, "FPGA-Based Digital Pulsewidth Modulator With Time Resolution Under 2 ns", *IEEE Transactions On Power Electronics*, vol. 23, n°6, pp. 3135-3141, November 2008.

[48] A. V. Peterchev and S. R. Sanders, "Quantization resolution and limit cycling in digitally controlled PWM converters", *IEEE Transactions On Power Electronics*, vol. 18, no. 1, pp. 301–308, January 2003.

[49] A. Myaing, V. Dinavahi, "FPGA-Based Real-Time Emulation of Power Electronic Systems With Detailed Representation of Device Characteristics," *IEEE Transactions On Industrial Electronics*, Accepted for publication in 2010.

[50] S. Karimi, A. Gaillard, P. Poure, S. Saadate, S. "FPGA-Based Real-Time Power Converter Failure Diagnosis for Wind Energy Conversion Systems", *IEEE Transactions On Industrial Electronics*, vol. 55, n°12, pp. 4299-4308, December 2008.

[51] F. Blaabjerg, P.C. Kjaer, P.O. Rasmussen, C. Cossar, "Improved digital current control methods in switched reluctance motor drives", *IEEE Transactions On Power Electronics*, vol. 14, n°3, pp. 563-572, May 1999.

[52] M. Schroedl, "Sensorless control of AC machines at low speed and standstill based on the "INFORM" method," *In Proceedings IEEE-IAS'1996 Conference*, vol.1, pp. 270–277.

[53] D. Samuelsen, "AC machine control Robust and sensorless control by parameter independency," PhD Thesis, Narvik University College, Norway, 2009.

[54] K. Tazi, E. Monmasson and J.P. Louis, "Description of an entirely reconfigurable architecture dedicated to the current vector control of a set of AC machines," *In Proceedings IECON'1999 Conference*, pp. 1415-1420.

[55] Y.-S. Kung, R.-F. Fung, and T.-Y. Tai, "Realization of a motion control IC for x-y table based on novel FPGA technology", *IEEE Transactions On Industrial Electronics*, vol. 56, no. 1, pp. 43–53, January 2009.

[56] M. W. Naouar, A. A. Naassani, E. Monmasson, I. Slama-Belkhodja. "FPGA-based predictive current for synchronous machine speed drive," *IEEE Transactions On Power Electronics*, vol. 23, no. 4, pp.2115–2126. July 2008.

[57] B. Miao, R. Zane, D. Maksimovic, "System identification of power converters with digital control through cross-correlation methods", *IEEE Transactions On Power Electronics*, vol. 20, n°5, pp. 1093-109, Sept. 2005.

[58] A. Ordaz-Moreno, R. De Jesus Romero-Troncoso, J.A. Vite-Frias, J.R. Rivera-Gillen, A. Garcia-Perez, "Automatic Online Diagnosis Algorithm for Broken-Bar Detection on Induction Motors Based on Discret Wavelet Transform for FPGA Implementation", *IEEE Transactions On Industrial Electronics*, vol. 55, n°5, pp. 2193-2202, May 2008.

[59] P. Simi Valsan and K. Shanti Swarup, "High-Speed Fault Classification in Power Lines: Theory and FPGA-Based Implementation", *IEEE Transactions On Industrial Electronics*, vol. 56, no. 5, pp. 1793–1800, May 2009.

[60] E. Monmasson, H. Achelard, J.P Louis, "Dynamically reconfigurable architecture dedicated to the test of PWM algorithms", *In Proceedings EPE'1999 Conference*, CD-ROM, Lausanne, Suisse.

[61] S. Mariethoz, A. Domahidi, M. Morari, "A model predictive control scheme with torque ripple mitigation for permanent magnet motors", *In Proceedings IECON'2009 Conference*, pp. 2943-2948, Porto, Portugal.

[62] M. W. Naouar, "Commande numérique à base de composants FPGA d'une machine synchrone", PhD Thesis, UCP-ENIT, France-Tunisia, 2007.

[63] L. Charaabi, "Conception des architectures matérielles dédiées à la commande de systèmes électriques", PhD Thesis, UCP-ENIT, France-Tunisia, 2006.

[64] Texas Instrument technical documents. Available in www.ti.com

[65] A. Maalouf, S. Le Ballois, L. Idkhajine, E. Monmasson: "Sensorless control of brushless exciter synchronous starter generator using extended Kalman filter", *In Proceedings IECON'2009 Conference*, pp. 2581-2586, Porto, Portugal.

[66] L. Idkhajine, E. Monmasson, A. Maalouf: "FPGA-Based Sensorless Controller for Synchronous Machine using an Extended Kalman Filter", *In Proceedings EPE'2009 Conference,* CD-ROM, Barcelona Spain.

[67] L. Idkhajine, E. Monmasson, A. Maalouf: "Fully FPGA-based Sensorless Control for AC Drive using an Extended Kalman Filter", *In Proceedings IECON'2009 Conference*, pp. 2925-2930, Porto, Portugal.

[68] L. Idkhajine, E. Monmasson, A. Maalouf: "Extended Kalman Filter for AC Drive Sensorless Speed Controller - FPGA-Based solution or DSP-Based solution", *In Proceedings ISIE'2010 Conference*, Bari Italy.

[69] L. Idkhajine, E. Monmasson: "Optimized FPGA-based Extended Kalman Filter Application to an AC Drive Sensorless Speed Controller ", *In Proceedings SPEEDAM'2010 Conference*, pp. 1012-1017 Pisa Italy.

[70] N. Youssef, K. Al-Haddad, "Sensorless nonlinear control of a three-phase switch level Vienna rectifier based on a numerical reconstruction of DC and AC voltages", *In Proceedings MELECON'2008 Conference*, pp. 560-566, Ajaccio, France.

[71] P. J. Ashenden, J. Lewis, "VHDL-2008 Just the New Stuff", Elsevier and Morgan Kauffman publishers, 2008.

[72] T. Grandpierre, C Lavrenne and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessor", *In Proceedings* CODES'1999 *7th International Workshop on Hardware/ Software Co-Design Conference*, CD-ROM, pp. 74-78.

[73] Sang-Il Han, Soo-Ik Chae, Lisane Brisolara, Luigi Carro, Katalin Popovici, Xavier Guerin, Ahmed A. Jerraya, Kai Huang, Lei Li, Xiaolang Yan, "Simulink-based heterogeneous multiprocessor SoC design flow for mixed hardware/software refinement and simulation", Integration, the VLSI journal, Elsevier, 2008

[74] I. Bahri, "SW/HW Co-Design methodology for AC Drive applications", PhD Thesis, in process, to be defended in 2011.

[75] M. P. Kazimierkowski and L. Malesani, "Current Control Techniques for Three-Phase Voltage-Source PWM Converters", *IEEE Transactions On Industrial Electronics,* Vol. 45, n°5, pp. 691-703, October 1998.

[76] R. Hoseinnezhad, A. Bab-Hadiashar, P. Harding, "Calibration of Resolver Sensors in Electromechanical Braking Systems: A Modified Recursive Weighted Least-Squares Approach", *IEEE Transactions On Industrial Electronics,* Vol. 54, n°2, pp. 1052-1060, April 2007.

[77] Santanu Sarma, V. K. Agrawal, Subramanya Udupa, "Software-Based Resolver-to-Digital Conversion Using a DSP", *IEEE Transactions On Industrial Electronics,* Vol. 55, n°1, pp. 371-379, January 2008.

[78] L. Ben-Brahim, M. Benammar, Mohd A. Alhamadi, "A Resolver Angle Estimator Based on Its Excitation Signal", *IEEE Transactions On Industrial Electronics,* Vol. 56, n°2, pp. 574-580, February 2009.

[79] M. Mienkina, P. Pekarek, F. Dobe "DSP56F80x Resolver Driver and Hardware Interface", Motorola, Inc., 2002

[80] R. Hoseinnezhad, P. Harding, “A Novel Hybrid Angle Tracking Observer for Resolver to Digital Conversion”, *In Proceedings CDC-ECC’2005 Conference*, pp. 7020-7025, Seville, Spain.

[81] W. Naouar, A. Naassani, E. Monmasson, I. Slama Belkhodja, “FPGA-Based Speed Control of Synchronous Machine using a P-PI Controller”, *In Proceedings ISIE’2006 Conference*, pp 1527-1532, CD-ROM.

[82] L.Charaabi, E.Monmasson, I.S.Belkhodja, “Presentation of an efficient design methodology to develop IP-Core Functions for Control Systems: Application to the Design of an Antiwindup PI Controller”, *In Proceedings IECON’2002 Conference,* pp. 1942 - 1947 , CD-ROM.

[83] M. P. Kazimierkowski, “Teaching of Pulse Width Modulation Methods for Three-Phase Converter Using Internet”, *IEEE Transactions On Industrial Electronics*, Vol. 51, No 2, 2004.

[84] F. Blaabjerg, J. K. Pedersen, and P. Thoegersen, “Improved modulation techniques for PWM-VSI drives”, *IEEE Transactions On Industrial Electronics*, vol. 44, pp. 87–95, February 1997.

[85] D. Leggate and R. Kerkman, “Pulse based time compensator for PWM voltage inverters”, *In Proceedings IECON’1995 Conference*, pp. 474–481.

[86] J.-W. Choi and S.-K. Sul, “Inverter output voltage synthesis using novel dead time compensation”, *IEEE Transactions On Power Electronics*, vol. 11, pp. 222-227, Mars 1996.

[87] C. Wang, L. Xu, “A Novel Approach for Sensorless Control of PM Machines Down to Zero Speed Without Signal Injection or Special PWM Technique” *IEEE Transactions On Power Electronics*, vol. 19, N° 6, pp. 1601–1607, November 2004.

[88] J. Holtz, “Sensorless Control of Induction Machines – With or Without Signal Injection?”, *IEEE Transactions On Industrial Electronics,* Vol. 53 No 1, pp. 7-30, February 2006.

[89] H. Kim, K. K. Huh, R.D. Lorenz, T. M. Jahns “A novel method for initial rotor position for IPM synchronous machine drives” *IEEE Transactions On Industry Applications,* vol. 40, n° 5, pp. 1369-1378, October 2004.

[90] M. Fadel, R. Ruelland, G. Gateau, JC Hapiot, P. Brodeau, JP. Carayon, “Commande sans mécanique des actionneurs embarqués” *Journées de la section électrotechnique du club EEA*, Cergy Pontoise, France, March 2004.

[91] J.A. Solsona, M.I. Valla, “Disturbance and nonlinear Luenberger observers for estimating mechanical variables in permanent magnet synchronous motors under mechanical parameters uncertainties”, *IEEE Transactions On Industrial Electronics*, Vol. 50, N° 4, pp. 717-725, 2003.

[92] A. Maalouf, “Sensorless control for a Brushless Synchronous Starter Generator for Aircraft Applications”, PhD Thesis, in process, to be defended in 2010.

[93] S. Bolognani, M. Zigliotto, M. Zordan, “Extended-Range PMSM Sensorless Speed Drive Based on Stochastic Filtering”, *IEEE Transactions On Industrial Electronics*, Vol. 16, N° 1, pp. 110-117 January 2001

[94] S. Bolognani, M. Zigliotto, M. Zordan, “Rotor position detection for Sensorless PM synchronous motor drives”, *In Proceedings PEMC’1998 Conference,* pp. 883-888.

[95] S. Bolognani, L. Tubiana, M. Zigliotto, “Extended Kalman Filter Tuning in Sensorless PMSM Drives”, *IEEE Transactions On Industry Applications*, Vol. 39, No. 6, pp. 1741-1747, November 2003.

[96] A. Akrad, M. Hilairet, D. Diallo, “A Sensorless PMSM drive using a two stage Extended Kalman Estimator”, *In Proceedings IECON’2008 Conference*, pp. 2776-2781, Orlando, Florida, USA.

[97] J. Person, “Innovative Standstill Position Detection Combined with Sensorless Control of Synchronous Motors”, PhD Thesis, N° 3221, EPFL, Lausanne, 2005

[98] M. Hilairet, “Application des outils de traitement de signal à la commande des machines tournantes”, PhD Thesis, N° 0366-038, EPUN, St Nazaire, 2001

[99] Y-H. Kim , Y-S. Kook, "High Performance IPMSM Drives without Rotational Position Sensors Using Reduced-Order EKF", *IEEE Transactions On Energy Conversion,* Vol 14, No. 4, pp. 868-873, December 1999.

[100] S. Bolognani, L. Tubiana, M. Zigliotto, "EKF-Based Sensorless IPM Synchronous Motor Drive for Flux-Weakening Applications", *IEEE Transactions On Industry Applications*, Vol. 39, No. 3, pp.112-119, May/June 2003.

[101] M. Kosaka, H. Uda, "Sensorless IPMSM drive with EKF estimation of speed and rotor position", *In Proceedings CDC'2003 Conference*, pp. 5915-5920, Hawaii USA.

[102] Texas Instruments Europe, "Sensorless Control with Kalman Filter on TMS320 Fixed-Point DSP", Literature Number: BPRA057, July 1997

[103] Babak Nahid-Mobarakeh, Farid Meibody-Tabar, and François-Michel Sargos, "Mechanical Sensorless Control of PMSM With Online Estimation of Stator Resistance" *IEEE Transactions On Industry Applications*, Vol. 40, N° 2, pp. 457-471, March/April 2004

[104] Z. Peroutka : "Development of Sensorless PMSM Drives: Application of Extended Kalman Filter" *In Proceedings ISIE'2005 Conference*, pp. 1647-1652, Dubrovnik, Croatia

[105] Pavel Brandstetter, Martin Kuchar, David Vinklarek: "Estimation Techniques for Sensorless Speed Control of Induction Motor Drive", *In Proceedings ISIE'2006 Conference*, pp. 154-159, Montreal, Quebec, Canada

[106] Bendjedia, M.; Ait-Amirat, Y.; Walther, B.; Berthon, A : "Sensorless control of hybrid stepper motor" *In Proceedings EPE'2007 Conference*, CD-ROM, Aalborg, Denmark.

[107] J-W Choi, S-C Lee, "Antiwindup Strategy for PI-Type Speed Controller", *IEEE Transactions On Industrial Electronics*, vol. 56, no. 6, pp. 2039–2046, June 2009.

[108] M. Dagbagi, "Implantation numérique sur cible FPGA d'un émulateur temps réel d'une machine alternative", Master's report, July 2010, ENIT Tunisia.

[109] I. Bahri, M-W. Naouar, E. Monmasson, I. Slama-Belkhodja, L. Charaabi**, "**Design of an FPGA-based Real-Time Simulator for electrical system**",** *In Proceedings EPE-PEMC'2008 Conference,* CD-ROM, September 2008.

[110] W. Naouar, E. Monmasson, I. Slama Belkhodja, "Identification of Synchronous Machine Parameters Using Hysteresis Based Current Controller", In Proceedings IECON'2006 Conference, pp 1357-1362.

[111] J-P. Louis, "Modélisation des machines électriques en vue de leur commande", Traité EGEM Electronique - Génie électrique - Microsystèmes.

[112] M. Labarrere, J.P. Krief, B. Gimonet, "LE FILTRAGE et ses applications", CEPADUES editions 1995.