Année 2011

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

Par

Imen BAHRI

Contribution of FPGA-based System-on-Chip controllers for embedded AC drive applications

---

JURY

Rapporteurs	:	Prof. Guillaume Gateau	Université de Toulouse
		Dr. Daniel Chillet	Université de Rennes
Examinateurs	:	Prof. François Verdier	Université de Nice
		Prof. Ilhem Slama Belkhodja	Université de Tunis
		Prof. Bruno Allard	Université de Lyon
		Mr. Régis Meuret	Expert Safran Power
Directeur de thèse	:	Prof. Eric Monmasson	Université de Cergy Pontoise
Co-directeur	:	Dr. Mohamed El Amin Ben khelifa	Université de Cergy Pontoise

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

Contribution of FPGA-based System-on-Chip controllers for embedded AC drive applications

Imen BAHRI

November 13, 2011

Contents

Nomenclature

List of abbreviations

A^3	: Algorithm Architecture Adequation
ATO	: Angle Tracking Observer
ADC	: Analog Digital Converter
ASIP	: Application Specific Integrated Processor
ASIC	: Application Specific Integrated Circuit
CB-PWM	: Carrier Based Pulse Width Modulation
CFG	: Control Flow Graph
DAC	: Digital Analog Converter
DSP	: Digital Signal Processor
DFG	: Data Flow Graph
EKF	: Extended Kalman Filter
ECU	: Embedded Control Unit
FPGA	: Field Programmable Gate Array
FF	: Flip-Flop
HAL	: Hardware Abstraction Layer
HF	: High Frequency
IGBT	: Insulated Gate Bipolar Transistor
LUT	: Look Up Table
MEA	: More Electrical Aircraft
NSGA-II	: Non-Dominated Sorting Genetic Algorithm
PI	: Proportional Integral regulator
PMSM	: Permanent Magnet Synchronous Machine
PLB	: Processor Local Bus
P_c	: Crossover rate
P_m	: Mutation rate
RTOS	: Real Time Operating System
RTU	: Real Time Unit
RPU	: Resolver Processing Unit
SEU	: Single Event Upsed
SEFORA	: Smart MEA For Operating in Rough Atmospheres
SPI	: Serial Peripheral interface
SVM	: Space Vector Modulation
VHDL	: Very High speed integrated Hardware Description Language
VSI	: Volatage Source Inverter

List of symbols

Start	: Start signal
θ	: Electrical position
Γ_e	: Electromagnetic torque
Γ_L	: Load torque
J	: Inertia
p	: Pairs of poles of a motor
ω_e	: Electrical speed (rad/s)
E	: DC link
R	: Resistor
V	: Voltage
I	: Current
T_s	: Sampling period
T_{sw}	: Switching period
x	: State space vector
u	: Input vector
y	: Output vector
w, v	: System and measurement disturbances
K	: Kalman matrix
P, P_0	: State error covariance matrix, Initial state error covariance matrix
Q, R	: Covariance state noise and covariance measurement noise matrices
s	: Laplace operator

Indexes

$d-q$	: Rotating reference frame indexes
$\alpha-\beta$	: Stationary reference frame indexes
a, b, c	: 3-phase reference frame index
*	: Reference quantity
$\hat{}$	: Estimated quantity
k	: Sampling index
$k/k-1$	: Predicted quantity
k/k	: Optimal estimated quantity

Abstract

Designing embedded control systems becomes increasingly complex due to the growing of algorithm complexity, the rising of industrials requirements and the nature of application domains. One way to handle with this complexity is to design the corresponding controllers on performing powerful and open digital platforms.

More specifically, this PhD deals with the use of FPGA System-on-Chip (SoC) platforms for the implementation of complex AC drive controllers for avionic applications. These latter are characterized by stringent technical issues such as environment conditions (pressure, high temperature) and high performance requirements (high integration, flexibility and efficiency).

During this thesis, the author has contributed to design and to test a digital controller for a high temperature synchronous drive that must operate at 200°C ambient. It consists on the Flux Oriented Controller (FOC) for a Permanent Magnet Synchronous Machine (PMSM) associated with a Resolver sensor. A design and validation method has been proposed and tested using a FPGA ProAsic^{Plus} board from Actel/Microsemi Company. The impact of the temperature on the operating frequency has been also analyzed.

A state of the art FPGA SoC technology has been also presented. A detailed description of the recent digital platforms and the constraints in link with embedded applications was investigated. Thus, the interest of a SoC-based approach for AC drives applications was also established.

Additionally and to have full advantages of a SoC based approach, an appropriate HW-SW Co-design methodology for electrical AC drive has been proposed. This method covers the whole development steps of the control application from the specifications to the final experimental validation. One of the main important steps of this method is the HW-SW partitioning. The goal is to find an optimal combination between modules to be implemented in software and those to be implemented in hardware. This multi-objective optimization problem was performed with the Non-Dominated Sorting Genetic Algorithm (NSGA-II). Thus, the Pareto-Front of optimal solution can be deduced. The illustration of the proposed Co-design methodology was made based on the sensorless speed controller using the Extended Kalman Filter (EKF). The choice of this benchmark corresponds to a major trend in embedded control of AC drives.

Besides, the management of SoC-based architecture of the embedded controller was allowed using an efficient Real-Time Operating System (RTOS). To accelerate the services of this operating system, a Real-Time Unit (RTU) was developed in VHDL and associated to the RTOS. It consists in hardware operating system that moves the scheduling and communication process from software RTOS to hardware. Thus, a significant acceleration has been achieved. The experimentation tests based on digital current controller were also carried out using a laboratory set-up. The obtained results prove the interest of the proposed approach.

Keywords

- Field Programmable Gate Array
- AC drive controller
- Resolver sensor
- Sensorless controller
- High temperature
- System-on-Chip
- Hardware-Software Co-design
- Optimization
- Genetic algorithm
- Operating system
- Scheduler

Résumé

La conception des systèmes de contrôle embarqués devient de plus en plus complexe en raison des algorithmes utilisés, de l'augmentation des besoins industriels et de la nature des domaines d'applications. Une façon de gérer cette complexité est de concevoir les contrôleurs correspondant en se basant sur des plateformes numériques puissantes et ouvertes.

Plus précisément, cette thèse s'intéresse à l'utilisation des plateformes FPGA System-on-Chip (SoC) pour la mise en œuvre des algorithmes d'entraînement électrique pour des applications avioniques. Ces dernières sont caractérisées par des difficultés techniques telles que leur environnement de travail (pression, température élevée) et les exigences de performance (le haut degré d'intégration, la flexibilité).

Durant cette thèse, l'auteur a contribué à concevoir et à tester un contrôleur numérique pour un variateur de vitesse synchrone qui doit fonctionner à 200 °C de température ambiante. Il s'agit d'une commande par flux orienté (FOC) pour une Machine Synchrone à Aimants Permanents (MSAP) associée à un capteur de type résolveur. Une méthode de conception et de validation a été proposée et testée en utilisant une carte FPGA ProAsic^{Plus} de la société Actel/Microsemi. L'impact de la température sur la fréquence de fonctionnement a également été analysé.

Un état de l'art des technologies basées sur les SoC sur FPGA a été également présenté. Une description détaillée des plateformes numériques récentes et les contraintes en lien avec les applications embarquées a été également fournie. Ainsi, l'intérêt d'une approche basée sur SoC pour des applications d'entrainements électriques a été démontré.

D'un autre coté et pour profiter pleinement des avantages offertes par les SoC, une méthodologie de Co-conception matériel-logiciel (hardware-software (HW-SW)) pour le contrôle d'entraînement électrique a été proposée. Cette méthode couvre l'ensemble des étapes de développement de l'application de contrôle à partir des spécifications jusqu'à la validation expérimentale. Une des principales étapes de cette méthode est le partitionnement HW-SW. Le but est de trouver une combinaison optimale entre les modules à mettre en œuvre dans la partie logicielle et ceux qui doivent être mis en œuvre dans la partie matérielle. Ce problème d'optimisation multi-objectif a été réalisé en utilisant l'algorithme de génétique, Non-Dominated Sorting Genetic Algorithm (NSGA-II). Ainsi, un Front de Pareto des solutions optimales peut être déduit. L'illustration de la méthodologie proposée a été effectuée en se basant sur l'exemple du régulateur de vitesse sans capteur utilisant le filtre de Kalman étendu (EKF). Le choix de cet exemple correspond à une tendance majeure dans le domaine des contrôleurs embraqués pour entrainements électriques.

Par ailleurs, la gestion de l'architecture du contrôleur embarqué basée sur une approche SoC a été effectuée en utilisant un système d'exploitation temps réel. Afin d'accélérer les services de ce système d'exploitation, une unité temps réel a été développée en VHDL et associée au système d'exploitation. Il s'agit de placer les services

d'ordonnanceur et des processus de communication du système d'exploitation logiciel au matériel. Ceci a permis une accélération significative du traitement. La validation expérimentale d'un contrôleur du courant a été effectuée en utilisant un banc de test du laboratoire. Les résultats obtenus prouvent l'intérêt de l'approche proposée.

Mots clefs

- Réseaux de portes programmables sur site – Field Programmable Gate Array
- Contrôle d'entraînement électrique
- Capteur de position Resolveur
- Commande sans capteur mécanique
- Haute temperature
- Système sur puce
- Co-conception matériel-logiciel
- Optimisation
- Algorithme génétique
- Système d'exploitation
- Ordonnanceur

General Introduction

During these last decades, the More-Electric Aircraft (MEA) has solicited the interest of both academic and industrial communities. The main objectives consist in increasing the scope of electrical energy sources instead of the mechanical and pneumatic ones [2], [16].

It is true that the adoption of the MEA allows numerous benefits such as the performance optimization, the decrease of maintenance costs, the reduction of CO_2 gas emission and the weight gains. However, this trend comes along serious technical issues such as environment conditions (pressure, high temperature) and high performance requirements (high integration, flexibility and efficiency) [3].

In this context, this PhD thesis deals with the development of embedded control system for embedded AC drive used in aircraft applications. The design of aircraft applications embedded control systems is an interesting challenge due to the growing of control algorithm complexity, environment conditions and the rising of aircraft requirements and specifications. Moreover, high control reactivity and large bandwidth are essential for aircraft applications. This can be ensured based on mature technologies such as Field Programmable Gate Array (FPGA) digital platforms.

The FPGAs present a great interest to implement such complex algorithms [6]-[7]. Indeed, FPGAs provide high integration density and modularity. They also offer the possibility to design very powerful dedicated parallel architectures which can dramatically reduce the execution time. For a challenging aircraft embedded control system, the development of FPGA-based Intellectual Property (IP) modules is well convenient. It provides portability of modules between different targets. Consequently, it avoids losing time in development and certification which decreases the whole cost development system. Besides, for these critical applications, where the safety is of prime importance, the configuration must be kept against the SEU (Single Event Upset) radiations and even when power is off [76]. In this sense, FPGA Flash RAM technology has also demonstrated its efficiency in terms of reliability.

The evaluation of FPGA-based AC drives for electrical applications has been the focus of many researches. Providing a well-structured design methodology was their main concern [93],[11]. The evaluation of FPGAs has been extended to see how much it can be suitable for the implementation of complex AC drive controllers. A full hardware sensorless controller for a synchronous AC drive has been implemented [113]. It is based on the Extended Kalman Filter (EKF) which estimates the rotor position and speed. Other sensorless techniques based on high frequency carrier injection have also been

tested with Brushless Synchronous Starter Generator (BSSG) and successfully tested under an actual aircraft testbench [8].

Recently, FPGAs have integrated several heterogeneous natures of cores on a single chip. This approach, called System-On-Chip (SoC), consists of varied components like one or several processors, memories, matrix of programmable logic elements and interfaces, all in the same chip. This heterogeneous system allows taking a lot of advantages by the combination of software and hardware which provides more density integration and more flexibility [12],[10]. In this case, Hardware/Software (HW/SW) Co-design methodology becomes a strategic trend to optimally design embedded control systems.

Furthermore, with the ever increasing complexity of control algorithms, the management of the SoC-based embedded controllers can take advantages using a Real-Time Operating System (RTOS). This will provide an abstracted environment in order to simplify and to coordinate the behavior of the system. However, the overhead of RTOS services cannot be neglected especially for severe time constrain applications [29].

Thesis objectives and contributions

In the frame of the more electric aircraft domain, this PhD thesis is a contribution to the design of FPGA-based controllers for AC drives embedded applications. Based on a SoC approach, this work aims to optimally design the controllers. This is performed using a HW-SW Co-design methodology and a task manager. In the following, thesis objectives and author's contribution are detailed,

- In the frame of SEFORA project (Smart MEA For Operations in Rough Atmospheres), author has contributed to design and to test a digital controller of a high temperature synchronous drive. A design and validation methodology for FPGA-based digital controller was proposed. As first stage of validation, the proposed design method was tested on the ProAsic^{Plus} board from Actel/Microsemi. The impact of the temperature on the operating frequency was also studied. In future, an ASIC board will be synthesized by another partner of the project [17],[10].

- A HW-SW Co-design methodology for AC drives has been proposed. Taken into account functional and architectural constraints, this method presents a guidance to optimally implement control modules between HW and SW. It is ranging from the early specifications of the system to the final experimental validation. The illustration of the proposed Co-design methodology was made based on the sensorless speed controller using the Extended Kalman Filter (EKF). The choice of this benchmark corresponds to a major trend in embedded control of AC drives [115], [129].

- The "Non-dominated Sorting Genetic Algorithm (NSGA-II)" was chosen to deal with the considered multi-objectives optimization. It aims to find the Pareto-Front of optimal solutions minimizing the SoC consumed resources and the execution time. Thus, a significant speedup gain has been reached compared to the pure SW solution.

- The management of the SoC-based embedded controller was provided using an efficient Real-Time Operating System (RTOS). To accelerate the services of this

operating system, a Real-Time Unit (RTU) was developed in VHDL and associated to the RTOS. It consists in hardware operating system that moves the scheduling and communication process from software RTOS to hardware. The proposed design takes only 4.3% of the FPGA available resources which is very few compared to the obtained acceleration benefits.

Thesis outline

This thesis consists of five main chapters, described as follows.

Chapter 1 presents the background of this thesis. Then, the state of art of the digital platforms (FPGAs, ASIPs, ASICs, SoCs) used in embedded control systems is described. The interest of the SoC-based approach for AC drive applications is also discussed. Thus, a HW-SW Co-design methodology for electrical AC drives is proposed.

Chapter 2 deals with the design and validation of FPGA-based synchronous drive for high temperature environment. The designed controller is based on the Field Oriented Control (FOC) principle. The controlled system consists on the Permanent Magnet Synchronous Machine (PMSM) associated to a load and a resolver position sensor. As first stage of validation, the proposed method was performed using the ProAsic^{Plus} board from Actel/Microsemi Company. These tests allow the preparation of the first-time-right corresponding silicon ASIC board.

Chapter 3 details the specifications and algorithm development steps of the Co-design methodology. Thus, hardware specifications of the controlled system, chosen switching period and control parameters are detailed. The adopted control strategy consists on the sensorless speed controller based on the Extended Kalman Filter (EKF). During algorithm development, the impact of time delay on the control bandwidth and stability margin is analyzed and time delay limits were defined.

Chapter 4 presents the architectural development step. It consists in two main steps: the performance estimation (area, time, memory size) and the HW-SW partitioning. The aim is to find an optimal partitioning of control modules between HW and SW in terms of area, execution time and memory size. The Non-dominated Sorting Genetic Algorithm (NSGA-II) was used to deal with this multi-objective optimization.

Chapter 5 describes the development of a HW-SW RTOS to provide the management of the controller tasks. To have a more deterministic RTOS, a hardware Real Time Unit (RTU) was developed using VHDL and was associated to the RTOS. It consists in hardware operating system that moves the scheduling and the communication process from software RTOS to hardware. The scheduler and the semaphore services were implemented in hardware. An experimental validation based on current controller was carried out using a laboratory set-up. The obtained results give proof of the interest of the proposed approach.

Finally, conclusions are drawn and perspectives are given.

Chapter 1 State of art of FPGA-based System-on-Chip for embedded control systems

Chapter 1

State of art of FPGA-based System-on-Chip for embedded control systems

1.1 Introduction

Nowadays, Embedded Control Systems (ECSs) are becoming more and more sophisticated. This is a direct consequence of the growing complexity of control algorithm and the rising of industrial requirements [3],[2]. These requirements are not just limited to new algorithm concepts or a high level of performances. Indeed, flexibility, cost and time-to-market reduction are also of prime importance.

More specifically, the implementation of complex AC drives controller for avionic applications, which is the target application of this PhD, is a good example of these trends. In addition to the above mentioned requirements, these applications target always more reliable, efficient and most of all compact systems. In this context, the proximity of the digital control platforms from higher temperature heating sources, like the engines, are now investigated. This presents a serious technical issue. To cope with all these challenges, the use of efficient design methodologies which take benefits of the main advantages of the current digital technologies becomes crucial.

Among these technologies, the Field Programmable Gate Arrays (FPGAs) present a great interest for implementing such complex algorithms [4]-[9]. Their high integration rate and their ability to exploit the inherent parallelism of the algorithm to be implemented make them very advantageous compared to pure software solutions such as microcontrollers or Digital signal processors (DSPs).

Besides, it is now usual to implement processor cores within FPGAs. Hence, FPGA can be considered as full System-On-Chip (SoC) solutions that make them very attractive for implementing embedded control systems. However, an optimal implementation of control algorithm based on a SoC approach requires the use of a rigorous Hardware-Software Co-design methodology. Indeed, an efficient partitioning of the control algorithm between software and hardware parts must be established [13].

Furthermore, the use of Real-Time Operating Systems (RTOS) may provide a better management of the control tasks. It allows also embedded controllers to be designed, modified and expanded more easily.

In this chapter, the background related to this thesis is presented. Firstly, aircraft

application constraints are presented. Then, the available digital technologies used for implementing embedded control systems are presented. The interest of using FPGA and SoC approaches in the AC drive domain is also established and the needs to have a Co-design expertise is emphasized. Finally, a Hardware-Software (HW-SW) Co-design methodology for AC drives is proposed. It provides a full design flow ranging from specifications to the final FPGA implementation.

1.2 Aircraft application constraints

The More Electrical Aircraft (MEA) trend aims to increase the electricity part over the other types of energy on board of an aircraft [14]-[16]. This is mainly motivated by the expected gains in terms of weight, volume and cost. The inherent financial benefits have prompted aircraft manufacturers to integrate more and more power electronics systems in recent aircrafts. Figure 1.1 illustrates the progressive evolution of the embedded electrical power in recent aircrafts. As an example, the Electro Hydraulic Actuators (EHA) project aims to replace the initial hydraulic actuators by the Electro-Hydraulic ones. This system was successfully integrated into the airbus A320. The Electrical Thrust Reverser Actuation System (ETRAS), developed by Hispano-Suiza Company, has been also successfully used in the airbus A380.

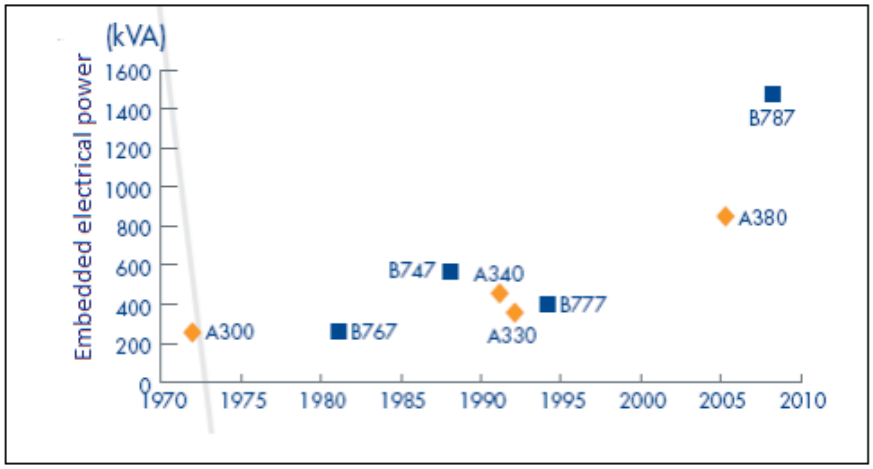


Figure 1.1: Evolution of the embedded electrical power in the aircraft domain[15].

Along this trend and focusing on digital controller, the main objective is a better integration of the digital controller. Before, complex control algorithms were performed using several Electronic Control Units (ECUs) that ensure the control of a physical plant, as depicted by Figure 1.2.

Main challenges in designing embedded control systems are:

- Flexibility: This issue is related to the possibility to adjust or to modify a function without having to re-design the whole controller.
- Modularity: It consists in dividing the whole control algorithm in functional modules. These functional modules are stored in a library, called Intellectual Property (IP) module library. The elements of this library can be re-used during the development of new projects, thus capitalizing the knowledge of the design team.

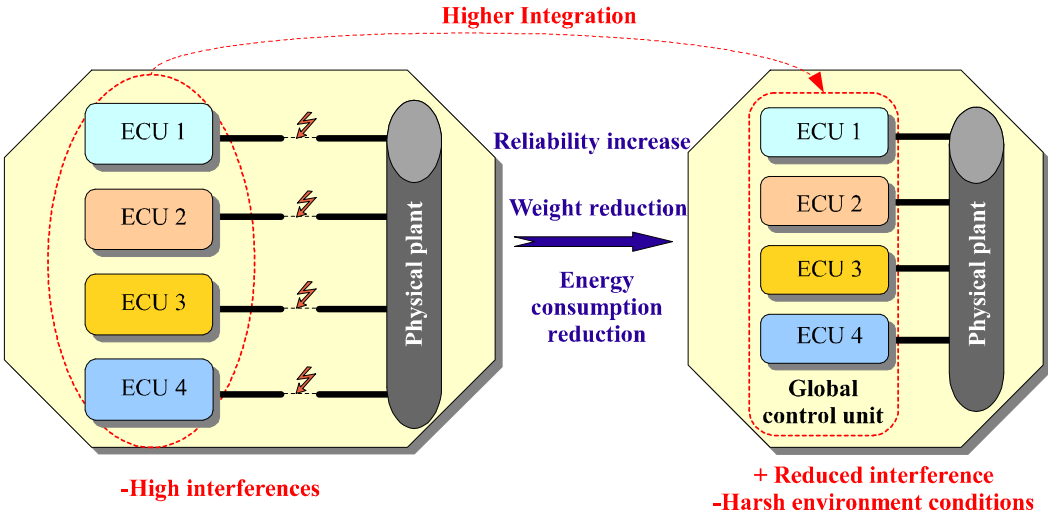


Figure 1.2: Trends of the Embedded Control Systems.

- High integration: The deployment of heterogeneous functions into one system, called global control unit, needs a high level of integration. As depicted Figure 1.2, this will lead to a more compact system with a significant reduction of wires, a reduced Electronic Magnetic Interference (EMI) impacts and a more reliable system.
- Testability: Once the controller is implemented, it is of great importance to be able to test it easily.

However and even if the assets of employing more power electronics in aircraft are demonstrated, these gains are conjointly associated to new challenges such as environment conditions (high temperature, high pressure). The search for ever more compact systems and their location in harsh environment impact seriously the reliability of the devices. This is the case of Smart MEA For Operation in Rough Atmosphere (SEFORA) project study which deals with the development of ECSs working at a range of temperatures from -55 to 200 °C[17]. Part of our thesis work deals with this high temperature challenge. More details are provided in the next chapter.

1.3 Hardware architectures

As said before, Embedded Control Systems (ECSs) are playing an increasingly important role in real-time control applications. Typical requirements are the development of small, reliable, and multi-functional system. In the same time, the continuous progress of CMOS technology and the increasing demands of new products have led to the integration of more and more transistors within a single chip, with respect of Moore's law.

Moore's law states that the number of transistors on a chip is doubling every 18–24 months. As a result, more computing capacity and higher integration level are provided to deal with the growing complexity of applications. According to the regular International Technology Roadmap for Semiconductors (ITRS)[22], Figure 1.3 presents the rythm of growth of the number of computing cores and of the memory size integrated in a single chip.

All these advances in digital electronics technologies lead to more efficient digital control units that take several forms. As shown in Figure 1.4, three main technologies are available: The ASICs, the FPGAs and the ASIPs.

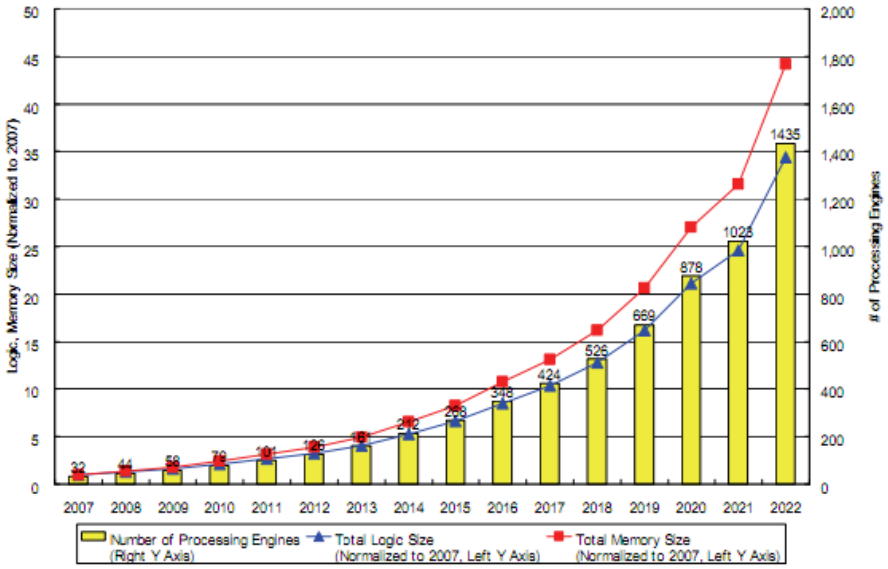


Figure 1.3: The growing of the computing cores numbers and the growing of the memory size integrated on a single chip [22].

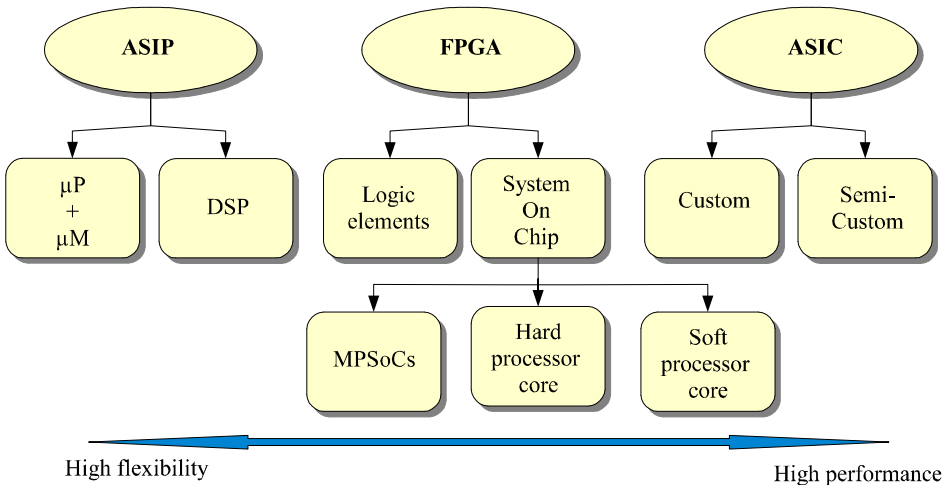


Figure 1.4: Hardware architectures used for embedded control systems

1.3.1 ASICs

The Application-specific integrated circuits (ASICs) are divided into two main groups: the circuits designed for custom applications (custom) and the pre-characterized circuits (semi-custom). ASICs are characterized by a high level of integration, predictable and good timing, reliability, high performances and low power consumption. All these characteristics make them very suitable for operation in harsh environments. However, these circuits are well known for their high design cost and long development cycle. Hence, designer must follow a strict design methodology and leads rigorous verifications to ensure

that there will be no functional nor timing problems on the final component. Moreover, these components are not used in the prototyping environment because they offer no programmability by the end user.

1.3.2 ASIPs

Application-Specific Instruction-set Processors (ASIPs) are processes dedicated to a given field of algorithm and for specific applications. ASIPs can be seen as dedicated microprocessors that also have additional peripherals (such as general purpose input/output, timer, counter...) and communication interfaces (I2C, CAN, SPI...) to interact in real time with their environment. This high integration ability makes this type components well suitable for signal processing and control systems.

The earliest ASIP's architectures were based on Von Neumann principle, shown Figure 1.5.a. They used a single bus to transfer data and instructions from the memory to the CPU. Two improvements have emerged with the Harvard architecture (shown Figure 1.5.b) : the use of two separate memories and the use of pipeline. The first memory is used for data storage and the second one to store instructions. Thanks to pipeline, parallel processing can also be performed. These features allow faster processing and high data throughputs.

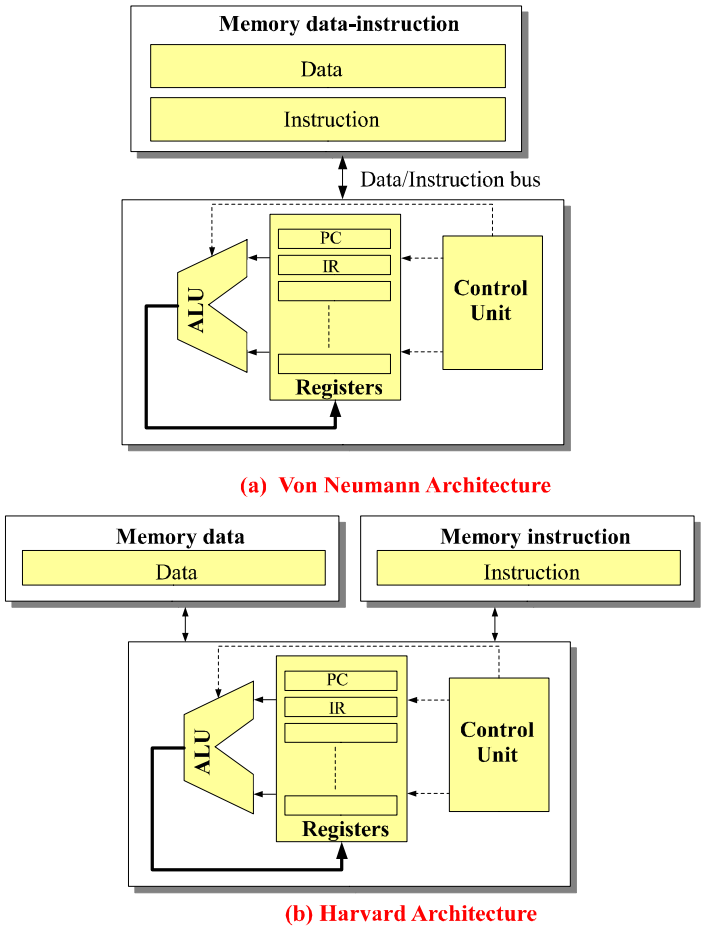


Figure 1.5: ASIPs architectures, (a) Von Neumann architecture, (b) Harvard architecture.

These ASIPs technologies have been used to perform a wide range of applications. Their main advantages are great flexibility and low cost. However, the use of these components is limited especially in applications demanding high computation performances. This is due to the fixed internal architecture which implies a serialization of the treatment.

1.3.3 FPGAs

The FPGAs technology is a good compromise between the flexibility of ASIPs and the performances of ASICs. As shown in Figure 1.6, FPGAs are based on a sea of logic blocks dedicated to treatment and on elements of interconnection between these blocks. Coarser grain hardware elements such as DSP blocks, Hard/Soft processor cores, memories and clock manager are also provided in most of the recent FPGA platforms [73].

FPGAs offer the possibility to design very powerful dedicated parallel architectures which can dramatically reduce the execution time of the control algorithm to be implemented. For embedded control system, the development of a library of FPGA-based Intellectual Property (IP) modules is convenient. This provides portability of the corresponding modules between different targets.

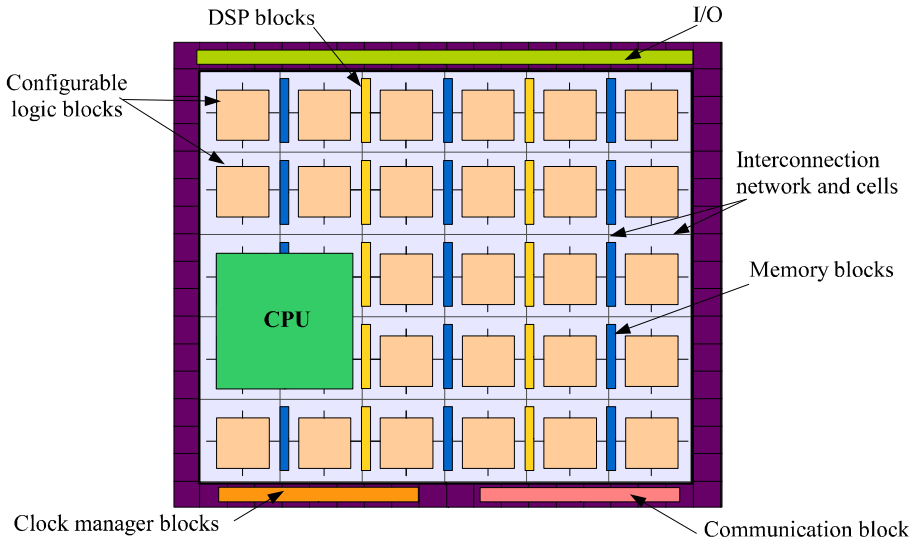


Figure 1.6: Generic structure of a FPGA.

In the following, we will present briefly the most popular FPGA technologies:

- SRAM technology: It is mainly proposed by the Xilinx and Altera Companies. The advantages of this technology are its high density and its rapid reconfiguration. Moreover, these circuits can be dynamically reconfigured partially or entirely. This process is named dynamic reconfiguration. But, the main drawback of SRAM-based FPGA is the use of standard memory loaded at initialization because they use CMOS technology. So, the use of this technology is limited in case of critical-safety applications such as aircraft and automotive fields [73],[75].

- Antifuse technology: The principle of this technology is based on the injection of a high current or a laser that heats and then melts the silicon layer between endpoints

so as to make connections. So, the configuration is maintained even after power is off. However, it is a One-Time-Programmable technology. This makes it impractical in the case of prototyping environments.

- Flash technology: The configuration of this technology is based on flash connections that keep the configuration state when the power is off. Its immunity against the Single Event Upset (SEU) radiations and its reduced static power consumption make this technology attractive for embedded systems. However, the main disadvantages of this technology are the limitation of the available internal resources and the limited number of reconfiguration cycles [76].

1.3.4 System-on-Chip

More recently, new architectures, called System-On-Chip (SoC) have emerged. A SoC architecture integrates within a single chip various components like one or more processors, analog circuitry, memories, a matrix of programmable logic elements (see Figure 1.7). By combining software and hardware, these solutions provide a better integration density, a reduced communication overheads and a good level of flexibility. All these assets prove that these solutions present a promising interest for control applications.

In SoC-based approaches, two types of processor cores can be considered, the "Hard processor core" and the " soft processor core".

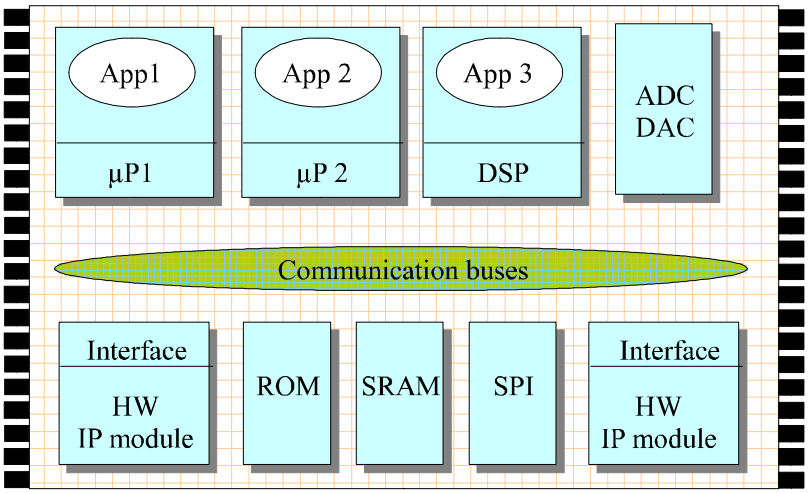


Figure 1.7: Generic System-on-Chip architecture.

1.3.4.1 Analog Device

Actel/ Microsemi Fusion family is a mixed-signal FPGA platform integrating processor and configurable analog. It offers a new level of integration by allowing the designer to implement an Analog-Digital converter (ADC) within the chip. It consists on a 12-bit ADC successive approximation. These features make these devices suitable for control applications [76], [10], [24].

1.3.4.2 Hard processor cores

The hard processor cores are non-synthesized processors. They are characterized by a custom layout using VLSI which is integrated within the FPGA with other internal resources. To communicate with its external environment, this processor needs the use of some peripherals and buses that consume resources from the FPGA matrix such as communication buses, memories, timer and so on. These processor cores offer fast processing and communication advantages. For example, Altera provides an ARM9 processor in its series EPXA10 which are marketed as Excalibur ^{TM} device [75]. Xilinx proposes also a PowerPC 440 hard-wired on-chip [73]. Recently, Actel launched a Cortex-M3 integrated into a smart-Fusion FPGA [76]. The architecture of the Cortex-M3 is presented by Figure 1.8.b.

1.3.4.3 Soft processor cores

The soft processor cores are synthesized processors. The FPGA configurable logic cells are used to implement these processor cores. The main advantage is the possibility to choose the configuration options. Hence, designer can choose the processor functionalities that correspond to the final application. An example of soft processor core is the Xilinx Microblaze. Its architecture is presented Figure 1.8.a.

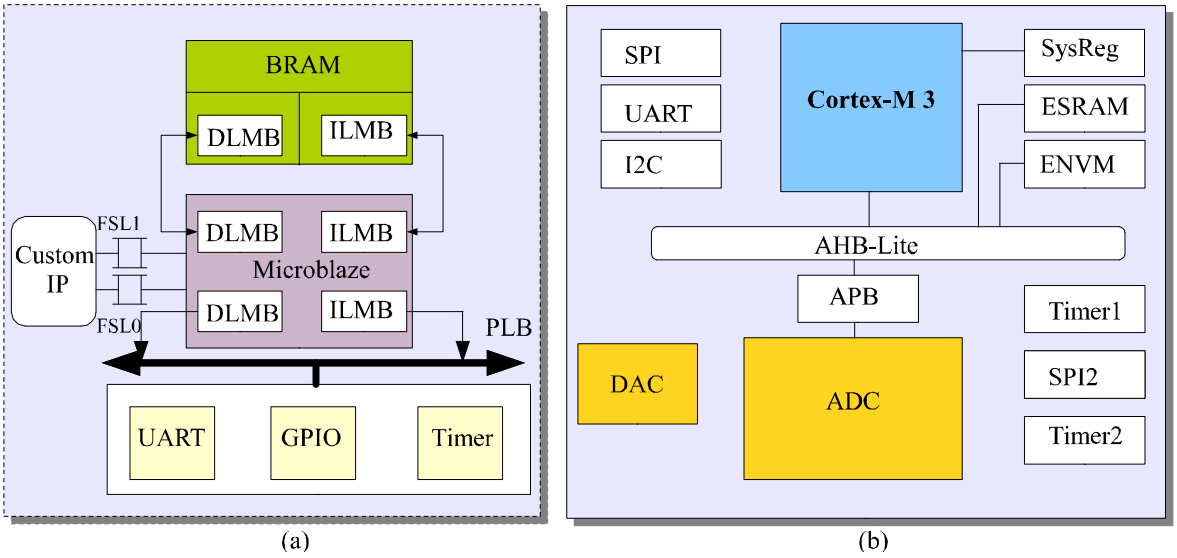


Figure 1.8: Processor architectures (a) Xilinx Microblaze soft processor core, (b) Actel/Microsemi SmartFusion Cortex-M3 hard processor core.

Table 1.1 presents a comparison between a standard DSP controller device and several SoC processor cores (Xilinx PowerPC hard core, Actel/Microsemi Cortex-M1 and Altera NIOS II/f soft processor core).

Features	**Xilinx Virtex – 5**	**Actel Fusion1**	**Altera Stratix III**	**Texas Intruments**
CPU	PowerPC 440	Cortex-M1	NIOS II/f	DSP C28x
Frequency(MHz)	400	60	290	150
Bit Number	32 b	16/32 b	32 b	32 b
Pipeline stages	*RISC superscalar* $7 - stages$	*RISC* $3 - stages$	*RISC* $6 - stages$	*DSP* $8 - stages$
32x32 Multiplier	*Hardware* (1 *cycle*)	*Synthesizable standard*(3*cycles*) *small*(33*cycles*)	*Hardware* (1 *cycle*)	*Hardware* (1 *cycle*)
ADC	—	$SAR - 12b$ $rate : 600Ksps$	—	$Pipeline - 12b$ $rate : 12.5Msps$
Logic cells Usage	—	4353	1020	—
Synthesizable	No	Yes	Yes	No
Configurability	–	++	++	–
Performance	++	+	+	++

Table 1.1: Features of processor cores for SoCs

A recent study made by Texas Instruments compares the indicated architectures to some other popular architectures [23]. Table 1.2 presents the summary of the presented report. The comparison is based on time-to-market, performance, price, development easiness, power consumption and flexibility.

	Time-to-market	**Performance**	**Price**	**Power**	**Flexibility**
ASIC	Poor	Excellent	Good	Excellent	Fair
DSP(ASIP)	Excellent	Fair	Good	Good	Excellent
FPGA	Good	Excellent	Poor	Poor	Fair
MCU(ASIP)	Excellent	Fair	Excellent	Fair	Excellent
RISC(ASIP)	Good	Fair	Fair	Fair	Excellent

Table 1.2: Architectures comparison [22]

1.3.5 Multi-Processor System-on-Chip

Real-time applications are becoming increasingly complex. This explains the motivation towards the use of Multi-Processor in System-On-Chip architectures (MPSoC). This offers a higher level of performance over a single processor, especially in terms of computation performances [25]-[26].

However, the main MPSoC design challenges are the communication infrastructure. Indeed, most of these communication models are based on dedicated channels or shared

buses. Unfortunately, scalability is limited by serialization of multiple requests to access to the bus. One promising approach is the concept of network-on-Chip (NoC). It consists in a set of routers that compose a network, which allows all the nodes connected to it (containing system resources and cores) to communicate with each others [27],[28]. This promises higher communication bandwidth than standard buses and more reusability.

1.3.6 Multi-layer software architecture

To deal with the rising complexity of control applications, the use of an operating system allowing the management of tasks and hardware resources is becoming necessary [29]. As far as the software part is concerned, different levels of abstraction between the application and the hardware that runs it can be established. Figure 1.9 shows the different layers between the physical system and the application.

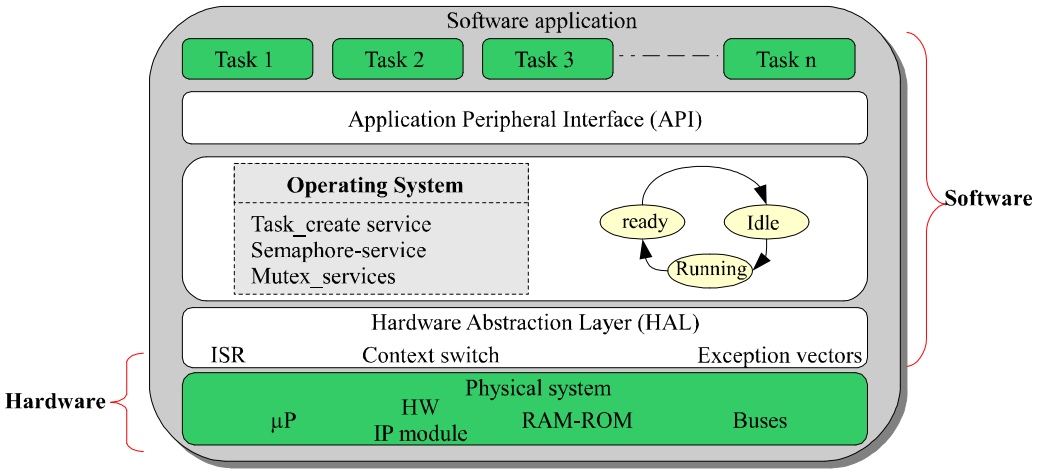


Figure 1.9: Multi-layer software architecture.

- *Software application:* It integrates one or more functional tasks to be executed. This part depends on the studied application.
- *Application peripheral interface (API):* It is also called middleware. It provides the mapping between the RTOS services and the application. The advantage of using common API is to provide the portability of the application to other standard-based operating systems.
- *Operating system:* It includes all the RTOS initialization functions and services (scheduler, semaphore_service, Mutex_service...) that allow the management of the available resources. Using a single processor, the system can implement pseudo-parallelism in the software part by multiplexing the resources.
- *Hardware Abstraction Layer (HAL):* This part was defined to overcome the disadvantages of dependency of the RTOS on the used hardware architecture. Using the HAL (including ISR, Context switch, exception vectors..), the RTOS can be used regardless the physical system on which it is executed. This can be provided using a low level interface which is dependent on the used processor. Generally, it is developed using assembly code.

- *Physical system:* It represents all the available hardware resources of the system. In the case of SoC approach, it consists in one or several processors, memories blocks, logic elements and so on.

1.4 FPGA-SoC design flow

Designing SoCs requires the use of appropriate tools. Thus, FPGA companies provide development tools for software and hardware parts. Figure 1.10 shows the standard design flow for SoC application development. This design flow consists of two main procedures: the hardware design flow and the software design flow. The first flow includes hardware design and verification tools (VHDL/Verilog editor, synthesizer, place/route and implementation and simulation tools). The second flow offers a user-friendly interface that allows the designer to customize the processor for a specific project. After its configuration, the processor core is generated in the form of an HDL file (in the case of Altera and Actel tools) or a netlist file (in the case of Xilinx tools) [73],[75],[76]. Then, this file can be associated to custom user logic and integrated within the hardware design flow to be synthesized, placed and routed. The FPGA can be configured with the resulting bitstream file. Then, the program which will be integrated on the soft processor core can be compiled with the associated library files and C header files. A C/C^{++} compiler targeted for this processor is also provided. For example, Xilinx provides the Embedded Development Kit (EDK) platform, Altera provides the Embedded Design Suite (EDS) platform and Actel/Microsemi provides the SoftConsole platform.

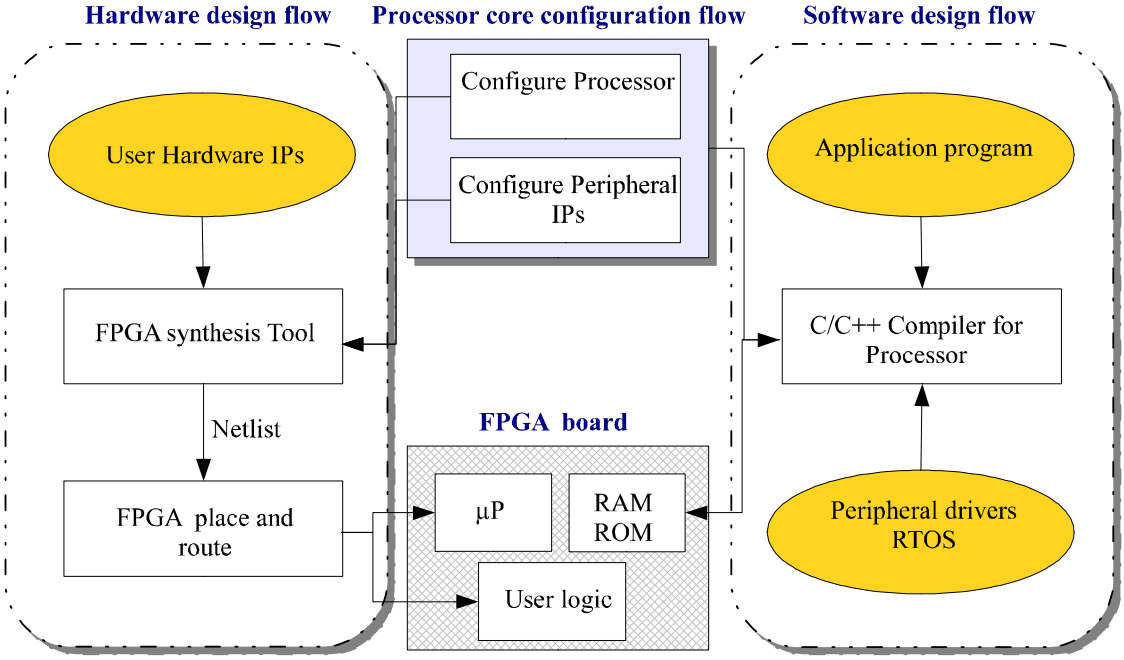


Figure 1.10: FPGA-SoC design flow.

1.5 Interest of the SoC approach for AC drive applications

As shown in Figure 1.11.(b), designing embedded control systems based on a SoC approach involves three main fields: feedback control, task scheduling and Co-design process. Hence, the whole control performance is not only related to how control algorithm was assessed, but it also relies on the scheduling policy and the Co-design procedure. Unfortunately, the development of embedded control systems is generally done by separated teams. Thus, control designers may consider that digital platforms are sufficiently deterministic to deal with the periodic treatment of control algorithms. These latter are often associated to severe timing constraints which can affect control performances once violated. Therefore, the scheduling of these real-time control applications presents a key issue since it is strongly related to the system performances.

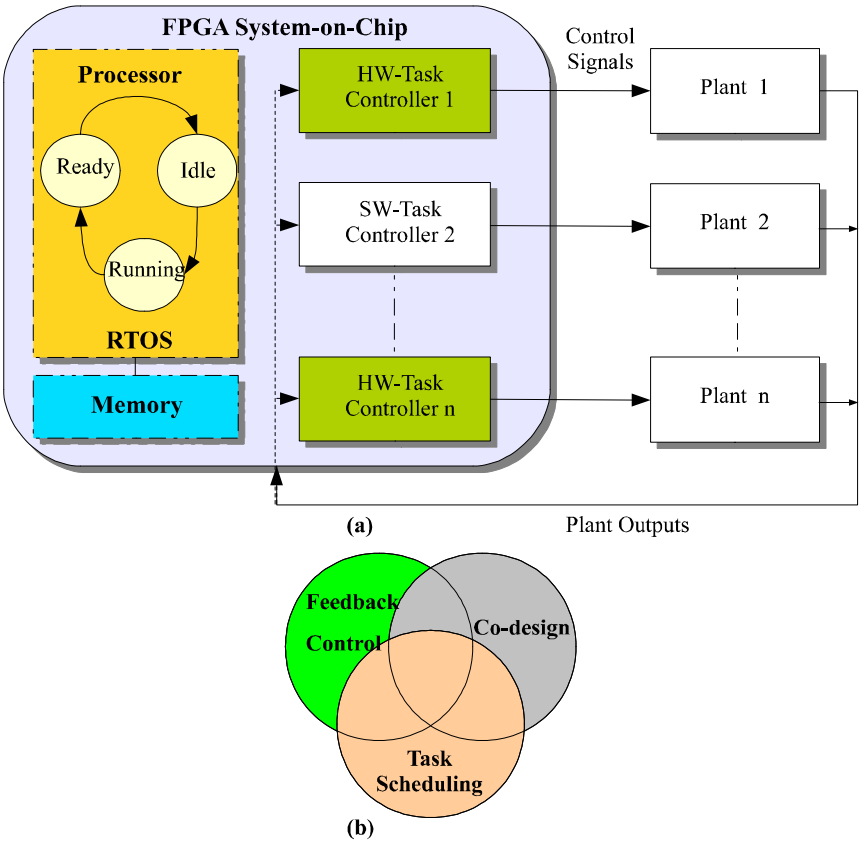


Figure 1.11: (a) SoC-based multi-tasking control application, (b) digital control fields: feedback control, task scheduling and co-design process.

One solution is to combine Feedback control scheduling and Co-design processes. Thus, the goal will be the optimal use of the digital platform resources and the maximization of the performances of the whole control system. Consequently, there is an increasing need to consider simultaneously the control performances and the digital design in order to develop cost efficient control application. This can be reached based on HW-SW Co-design methodology for AC drive which promotes the cooperative development of the control assessment and the task scheduling-design step in a single design flow.

Figure 1.11.(a) presents the use of SoC-based approaches for multitasking control applications. The principle is to control several plants by adjusting the timing attributes of controller tasks. This is a quite interesting approach because it allows the centralization of the control process for multi drive applications. The control tasks can be implemented indifferently in software or in hardware. All tasks can be scheduled using a RTOS.

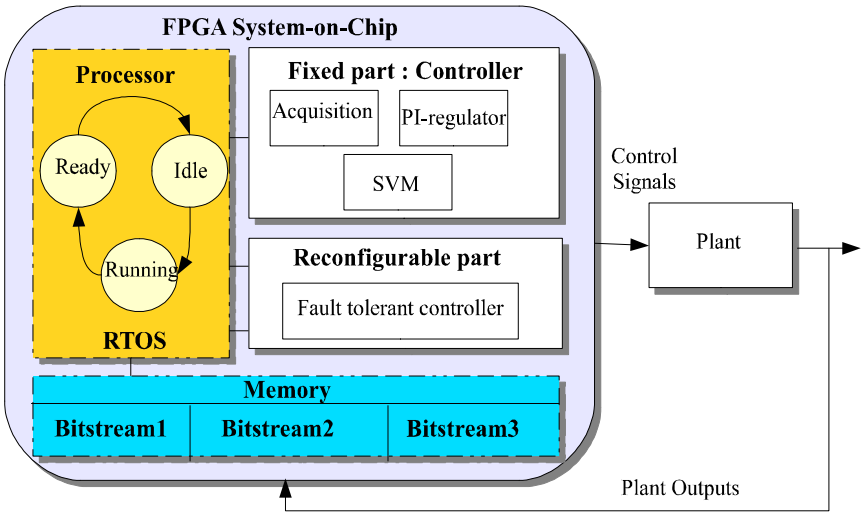


Figure 1.12: Reconfigurable control architecture.

Furthermore, reconfigurable FPGA platforms offer an interesting potential for use in the embedded control systems. It provides the possibility to be reconfigured or even better to reconfigure itself totally or partially without affecting the rest of the architecture. Therefore, the flexibility is no longer restricted to software [30]. Hence, in the case of control strategy change or monitoring process, the controller can be reconfigured in few microseconds while the system is continuing to work properly [31],[32]. This could be greatly appreciated especially for health monitoring control applications and robotics applications.

The reconfiguration can also improve FPGA fault tolerant in the case of SEU problems (case of SRAM technology). An overwriting of the existing configuration can be made while keeping the board in active operation mode [33], [34]. Figure 1.12 shows an example of FPGA partial reconfiguration. Under faulty condition, a fault tolerant controller can be downloaded in the reconfiguration part. This will allow the continuity of control function and avoid the system degradation. The reconfiguration process is provided using a RTOS and the Bitstream is saved in the memory. More than one Bitstream can be used to perform several configuration processes.

Now as the interest of SoC-based approach for embedded control systems is presented, a full HW-SW Co-design methodology for AC drives applications will be presented in this Ph.D. work. A brief description of the proposed Co-design methodology is now given. Details will be given in chapter 3 and 5.

1.6 Co-design methodology

"Hardware-Software Concurrent Design" , also called "HW-SW Co-design", is generally based on two important steps: modeling and partitioning that are strongly related.

1.6.1 Modeling

The modeling presents a key issue to handle the growing complexity of Co-design approaches. Thus, a variety of models has been developed to represent heterogeneous systems. These models are related to the nature of application domains. For example, real-time systems will be modeled on a timing basis while data base-systems will be modeled on a data exchange flow basis.

In the following, author presents briefly the main representation models that were used in the field of Co-design.

- *Model based on data flow graph :* It is often used to model systems characterized by strong data dependency. Thus, these models consist of nodes representing operations and edges representing data transfer between operations. The execution of a node is possible only if all its input data are available [37].

- *Model based on communication process competing :* This model is characterized by its ability to describe the parallelism of an application. Systems are modeled as a set of processes that run independently. Thus, all the processes are strongly decoupled to promote the parallelism and flexibility of system modeling. This type of modeling is very suitable for applications in the field of telecommunication [38].

- Model based on *Petri net :* It is a graphical representation of system behavior based on causal relationship between events affecting the system (transition, data transfer). Time is explicitly associated with these representations in order to build realistic models [36].

- *Model based on Finite state machines (FSM) :* The classical FSM is considered the best known model describing control systems. This model consists of a set of states, a set of input / output. The main disadvantage of this approach is the significant growth of states depending on the complexity of the application [35].

- *Reactive synchronous model :* This model is able to describe complex control applications characterized by strong competition between tasks. Often, the timing characteristics of these models allow the description of real-time systems using concurrent processes. This approach is based on parallel and asynchronous communications [39].

In the present work, the proposed Co-design methodology is based on the data flow graph model since it reflects accurately the characteristics of the control algorithm to be implemented (see chapter 4).

1.6.2 Partitioning

One of the main steps of Co-design methodologies is the HW-SW partitioning. Indeed, the goal is to find the optimal partitioning of application tasks between SW and HW in terms

of area allocation, time scheduling and memory use. The efficiency of the partitioning step depends on two main parts: the considered granularity level for tasks description and the used optimization algorithm.

The granularity level deals generally with two level : fine and coarse. The first one considers basic arithmetic operators such as (subtraction, addition, multiplication...). The second one presents more complex functions using several basic operators.

The optimization algorithm deals with the scheduling and the allocation of tasks. The focus here is to achieve objectives considering constraints. Several optimization algorithms were used to solve this kind of problem. These algorithms are genetic algorithm [106]-[107], branch-and-bound algorithm [98], Integer Linear Programming (ILP) algorithm [99]. More details will be given in section 4.7.

1.6.3 Modeling tools for Co-design

The literature concerning HW-SW Co-design methodologies is very rich and several approaches with different partitioning techniques were proposed. The main tools are briefly discussed below.

- *SystemC* : It consists on C $^{++}$class library. The integration of timing characteristics of the system and the competition between tasks are possible. This language was successfully used to model heterogeneous HW-SW systems. Tests based on the modeling of operating systems were also performed [40].

- *Polis* : It consists in a complete development class C (modeling, verification and simulation tool). This tool deals with control oriented applications. Thus, it is based on FSMs that are well suitable especially for the control applications. But, the use of this tool is not adapted for applications dominated by data processing [41].

- *Ptolemy* : It is a design environment developed at the University of Berkeley. It provides modeling, simulation, and design of concurrent tasks. It is mainly based on DFG and FSM modeling.[45].

- *Syndex* : This tool presents a system level CAD software based on the algorithm architecture adequation (A^3) methodology. It is generally used for rapid prototyping and aims to optimize the allocation of the available resources [43].

1.7 Proposed Co-design methodology for electrical drives

The proposed Co-design methodology aims to link the assessment of control performances and the HW-SW partitioning of control modules, at early stage of the development. As shown in Figure 1.13, this method is decomposed into four main steps: specifications, algorithm development, architectural development and HW-SW integration.

This method has been illustrated using a speed sensorless controller based on an Extended Kalman Filter (EKF) . The details of the proposed benchmark are presented in the rest of this PhD report (Chapters 3, 4 and 5) .

A-Specifications

The development of an AC drive application begins always by the specifications of the whole control application. This includes the definition of the used physical system and of the control parameters. The specification of the physical system consists in choosing the AC motor, the digital control unit, the Analog to Digital (ADC) and the Digital to Analog Conversion (DAC) interfaces. The environment conditions (high/low/ambient temperature) should be also defined since it affects the whole control performances.

In our case, a speed sensorless controller based on a EKF is considered. The EKF is used to estimate the position and speed of the rotor. It is characterized by its high level of complexity, including matrix multiplications and inversion. Two switching frequency rates are also considered: 20 kHz (presenting an example of a constrained switching frequency applications) and 100 kHz (presenting an example of an high demanding application). This part is developed in chapter 3.

B-Algorithm development

The Algorithm development process aims to the design and the validation of the control algorithm. Firstly, a modular partitioning is adopted. It consists in dividing the whole control algorithm into independent and reusable modules. In our cases, we consider functional modules (transformation, regulator, acquisition...). Then, continuous-time functional simulations are performed using Matlab-Simulink Tools. This allows the verification of the correct functionality of the considered control system.

Then, the time delay impact is quantified. Its effect on the control performances is analyzed in time and frequency domains. Thus, the maximum allowable time delay regarding the specified phase margin and bandwidth is obtained. The maximum allowable time delay will be further considered as a timing constraint during the HW-SW partitioning process.

Next step of the algorithm development consists in the discretization and the normalization of the whole control algorithm. Here, a digital re-design approach is considered. Firstly, the control is synthesized in continuous time domain. Then, a discrete version of the control algorithm is derived using Euler transformation method. Then, a normalization is applied and a fixed-point discrete equivalent control algorithm is obtained.

C-Architectural development

Once the control algorithm to be implemented has been validated, the designer can undertake the architectural development phase. It consists in optimizing the HW and SW resources allocation when implementing the control algorithm. This requires a proper space exploration method based on two main steps: the performance estimation and the HW-SW partitioning. The first step allows the estimation of area, time and memory size for each control modules. In the favorite case where the application doesn't violate the architectural constraints especially the area one, designer can go directly to the partitioning step, otherwise designer must perform a Folding step. This latter is achieved based on the A^3 methodology [44].

Then, the HW-SW partitioning of the control algorithm is performed. The target is to minimize area, time and memory size with respect to the functional constraints (T_{Alg} which is the maximum allowable execution time derived from the algorithm development step) and to the architectural constraints (available resources). It is clear that it is a multi-objectives optimization problem. To deal with, the Non-dominated Sorting Genetic Algorithm (NSGA-II) was adopted. This optimization algorithm aims to find the Pareto-optimal solutions satisfying both the functional and the architectural constraints. More details are provided in chapter 4.

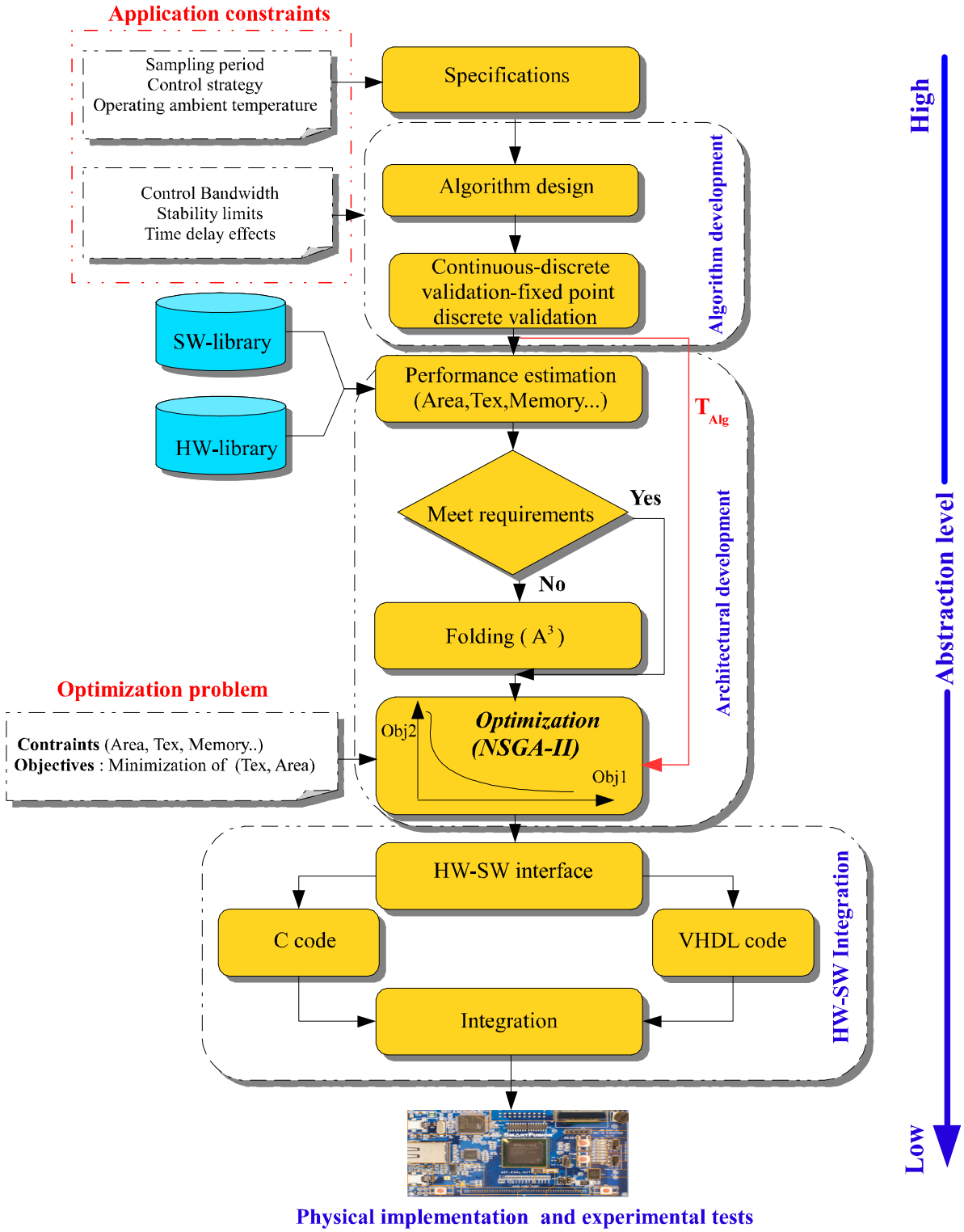


Figure 1.13: Co-design methodology for AC drive.

D-HW-SW integration

Once the HW-SW partitioning is performed, designer begins the HW-SW integration step of the chosen optimal solution. It consists in the development of the VHDL code

of the control modules to be implemented in HW and the development of the C code of the ones to be implemented on the SW. The communication between the two parts is allowed using the HW-SW communication interfaces. The integration of all these parts presents the next step. According to the design flow given in Figure 1.10, the physical implementation process can be performed and the Bitstream downloaded. Finally, the experimental validation of the chosen solution can be realized.

1.8 Conclusion

In this chapter, author has presented the background of this thesis. The new trends and the associated issues regarding embedded control systems have been given. A brief presentation of the available architectures (FPGAs, ASIPs, ASICs, SoCs) used in embedded control systems was firstly presented. Then, the interest of the SoC approach for AC drive applications has been discussed. Finally, the proposed Co-design methodology for electrical drives has been investigated. It consists on well-structured steps providing guidance for the SoC-based control applications. This method covers the whole development chain ranging from the specifications to the FPGA-based experimentation validation.

Chapter 2

Design and validation of FPGA-based motor drive for High-Temperature environment

Chapter 2

Design and validation of FPGA-based motor drive for High-Temperature environment

2.1 Introduction

Among embedded system trends, the one concerning More Electrical Aircraft (MEA) is probably one of the most challenging for electrical engineers. Indeed, even if the expected gains in terms of weight and volume due to the ever increasing part of electricity over the other types of energy on board of an aircraft are really significant, one cannot ignore the technical issues that come along [1]. The search for ever more compact systems and their location in harsh environment impacts seriously the reliability of the devices. The SEFORA project on which the author has contributed during her PhD is a good illustration of this problematic. The goal of the SEFORA project (Smart MEA For Operations in Rough Atmospheres) is to build a demonstrator of a full synchronous drive that is able to operate at 200 C° ambient. The corresponding actuator and its associated electronics are intended to be located near the reactor. As can be seen, electronics able to work at high temperature is of prime importance in this case [46]-[48].

More specifically, author has contributed to design and to test the digital controller of this high temperature synchronous drive. The designed controller is based on the Field Oriented Control (FOC) principle.

In this chapter, author proposes a design and a validation method for digital control architecture working in high temperature environment. To this purpose, two control boards were considered: an ASIC (CMOS 0.35 μm) and a ProAsic^{Plus} (AP1000) from Actel-Microsemi Company. However, as first stage of validation, the proposed design method was only tested on the ProAsic^{Plus} board, knowing that the ASIC board will be synthesized later by another partner of the project. The impact of the temperature on the operating frequency was also studied. Then, the architecture has been implemented at a frequency of 24 MHz with a junction temperature of 125 °C. Besides, a fault tolerant design strategy was also provided using segregation scenarii. This allows segregating the architecture of the controller in several operating zones. As a consequence, the risk of design issues when synthesizing the ASIC is minimized. The segregation control signals are controlled by a secure processor working at ambient temperature (soft processor core "Cortex-M1" implemented in a Actel/Microsemi Fusion-1 FPGA).

Regarding validation, a modular approach was chosen for testing the different control modules. As a final validation step, the whole controller was tested in real-time with an FPGA-based emulator of the drive.

Finally, having in mind in the future a system-on-chip solution for embedded drive control, this chapter ends by an evaluation of a full software implementation of a current controller for AC drives in the Cortex-M1. Thus, the ability of a soft processor core to implement a motor control is examined and its limits is given.

2.2 Application overview

The electrical control system considered in SEFORA project is presented Figure 2.1. It consists of a Permanent Magnet Synchronous Machine (PMSM) associated to a load and a resolver position sensor. This PMSM is supplied by a Voltage Source Inverter (VSI). The switching signals of the VSI are provided by the control algorithm which ensures two functions: the current control and the speed control of the PMSM. The position is estimated via a resolver. The corresponding treatment is achieved using a Resolver Processing Unit (RPU). The whole controller is based on the Field Oriented Control (FOC) principle.

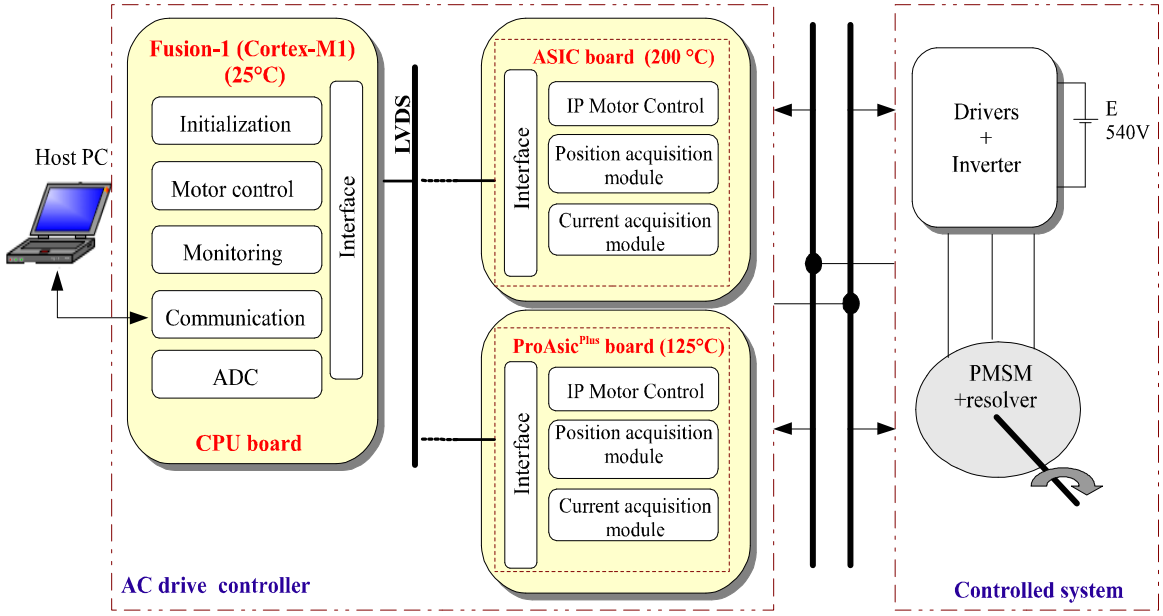


Figure 2.1: The synoptic of the actuator control board (SEFORA project).

Placed near the engine, the controller is faced to new constraints in link with high temperature. Therefore, the control algorithm was implemented using two boards: an ASIC and a FPGA. The ASIC, located in the high temperature zone (200 °C), contains the modules of motor control (abc-dq transformation, dq-abc transformation, PI regulator, SVM...) and the RPU module. In ambient temperature zone (25 °C), the SoC Fusion-1 from Actel family embedded the Cortex-M1 soft processor core was chosen. It implements the initialization, the monitoring modules and the communication process between the control boards and the Host PC.

The communication between the two boards is ensured by the use of Low Voltage Differentiate Signals (LVDS). This presents a safe method to transfer information for a long distance (separation between harsh and ambient environment) which is the case of our application.

For security reason, the control system was duplicated to ensure system redundancy and more safe control in the case of faulty conditions. Thus, the first control system is based on a FPGA Fusion-1 board associated to the ASIC one. The second control system is based on FPGA Fusion-1 board associated to the ProAsic^{plus} (APA1000) one.

In this chapter, two objectives are considered. The first one aims to the development and the validation of control algorithm under high temperature environment. The CMOS-ASIC technology 0.35 μm presents an interesting solution. However, it is widely known that ASICs are difficult to design and require significant costs and development effort [49]. Therefore, all tests were first carried out using FPGA ProAsic^{plus} APA1000 board. This was performed based on a rigorous validation methodology. Thus, these development and verification steps present the first stage of validation of the final ASIC board. This latter will be designed by IDMOS Company.

The second objective is the evaluation of SoC performances for real time control applications. To this purpose, the FPGA-Fusion1 board was chosen. It is characterized by mixed signal elements such hardware architecture (FPGA logic elements), internal Analog Digital Converter (ADC) and soft processor core (Cortex-M1). As testbench, a single current controller based on ON/OFF algorithm was considered. Such simple algorithm allows a first evaluation of the chosen SoC performances .

2.3 Design and validation methodology

The proposed methodology to validate the control algorithm is presented in Figure 2.2. This method is composed by three main steps [17].

A- Algorithm design and functional validation: It consists of designing and validating the control algorithm under Matlab-Simulink environment. The continuous-time, the discrete-time and the fixed-point discrete-time simulations are respectively performed [65].

B- Architectural design and modular verification : This step deals with the architecture development and its validation. Based on the A^3 methodology, the folding procedure is applied to the greediest operators in terms of consumed resources [44]. After that, a design verification flow, similar to the ASIC one, is used. It includes synthesis, place and route, timing analysis steps for each control block. Analysis regarding the temperature environment is also carried out to determine its impact on the designed architecture.

C- Real-time simulation : As global verification procedure, a Real-Time Simulation (RTS) of the electrical system, written in VHDL and implemented in FPGA, is used. It allows the validation of the controller before its application to the actual system allowing the analysis of the drive under several operating conditions [50], [51]. The developed emulator consists of power elements (Voltage Source Inverter "VSI", Permanent Magnet Synchronous Machine "PMSM") and AD interfaces (Analog Digital Converters "ADC", Digital Analog Converters "DAC"). Each of the indicated steps will be described in more details in the following sections.

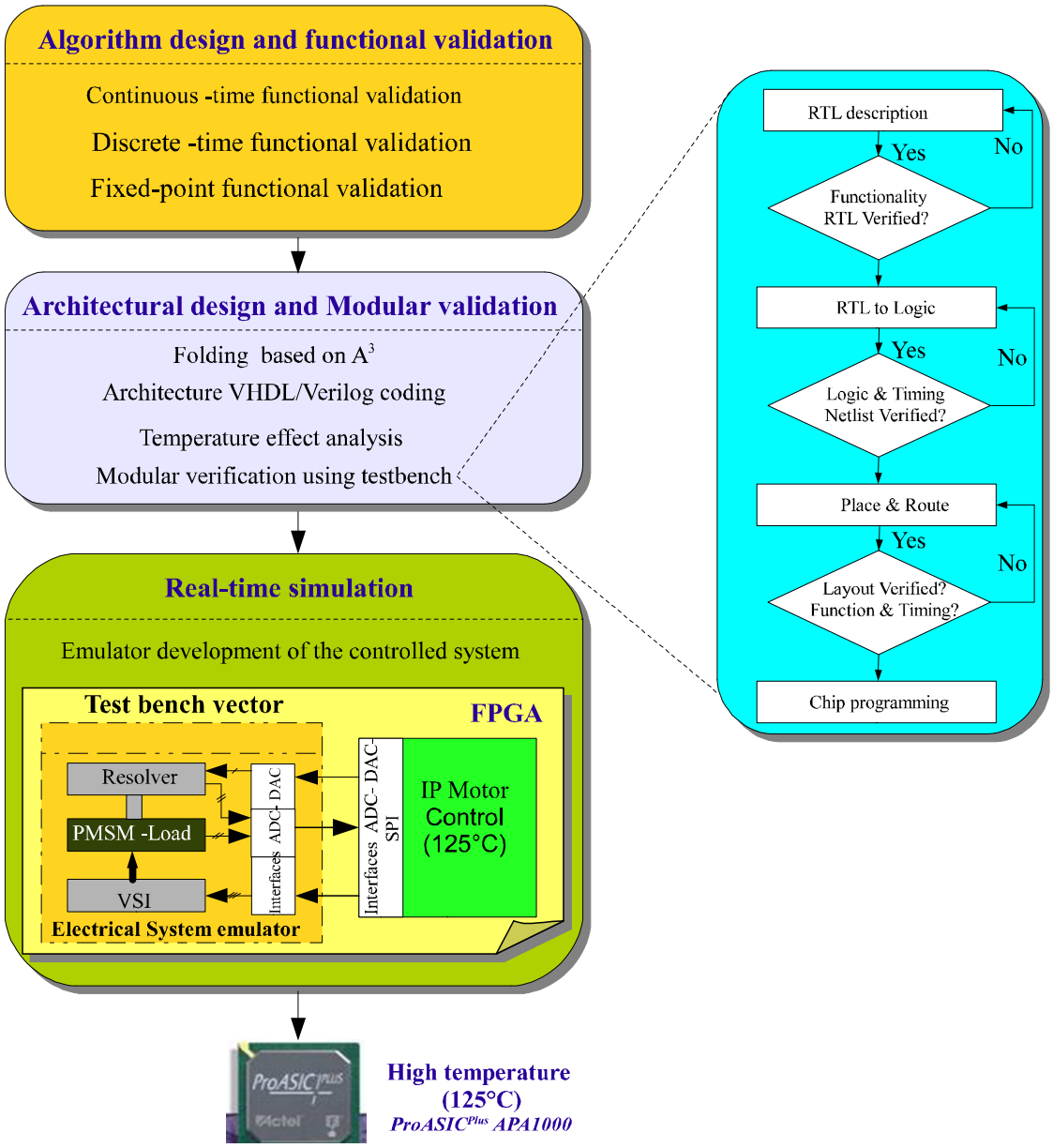


Figure 2.2: Design and validation methodology.

2.3.1 Control algorithm design and functional validation

In this section, author deals with the algorithm development and its validation. Firstly, the modular partitioning of the control algorithm into functional sub-blocks is provided to ensure reusability and modularity of the control algorithm. The synoptic of the derived modular control structure is presented in Figure 2.3. It consists on FOC using anti-windup Proportional Integral (PI) regulators and Space Vector modulation (SVM). The estimation of the speed and rotor position is provided via the RPU.

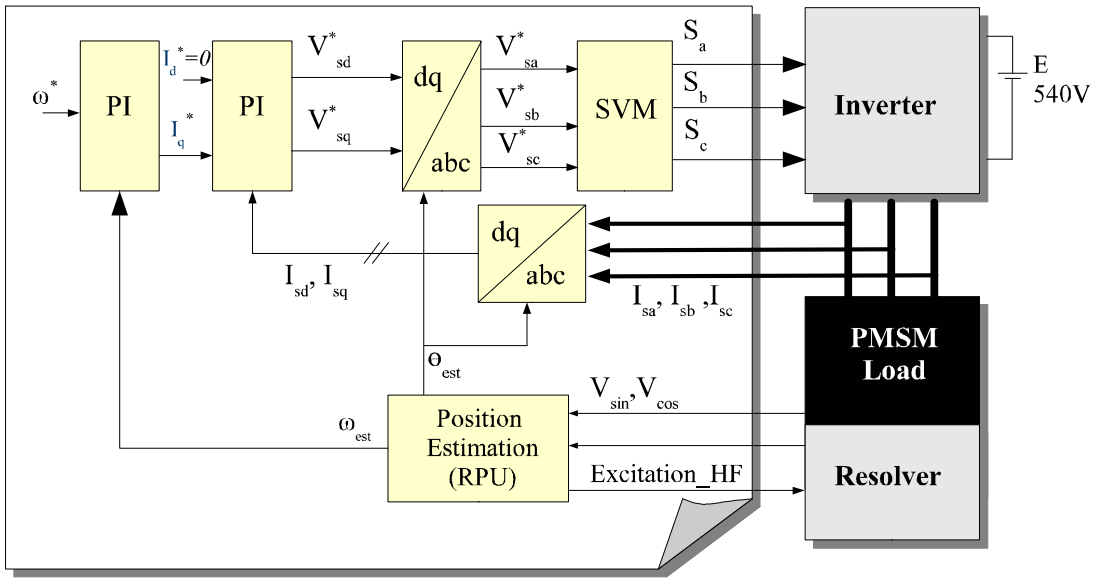


Figure 2.3: The synoptic of the control structure

- **Resolver Processing Unit (RPU)**

The resolver is a rotary transformer with one rotating reference excitation winding and two stator windings. The two stator windings are placed in quadrature and generate respectively the sine and cosine voltages [52]. The reference excitation winding is supplied by a High Frequency "Excitation_HF" square or sinusoidal voltage signal (10 kHz). The amplitude of these outputs is modulated respectively the sine and cosine of the electrical shaft angle according to the following relations,

$$\begin{aligned} V\sin &= m.E(t).\sin(\theta_r) \\ V\cos &= m.E(t).\cos(\theta_r) \end{aligned} \tag{2.1}$$

Where $E(t)$ is the high frequency square or sinusoidal excitation signal, m is the transformation ratio and θ_r the electrical rotor position.

Figure 2.4 presents the RPU principle. It is based on two steps: the synchronous demodulation (DMD) and the Angle Tracking Observer (ATO) algorithm. The first step consists in sampling the sine and cosine peaks and delivering the demodulated sine_D and cosine_D signals. Then, the ATO extracts the rotor position. It consists in a closed-loop which compares the actual position θ_r to the estimated position θ_{est}. The computation of sine and cosine of the estimated position can be ensured by a CORDIC (COordinate Rotation DIgital Computer) algorithm or via a sine table [53]. The objective is to minimize the observation error "e" given by the relation

$$e = \sin(\theta_r).\cos(\theta_{est}) - \cos(\theta_r).\sin(\theta_{est}) \tag{2.2}$$

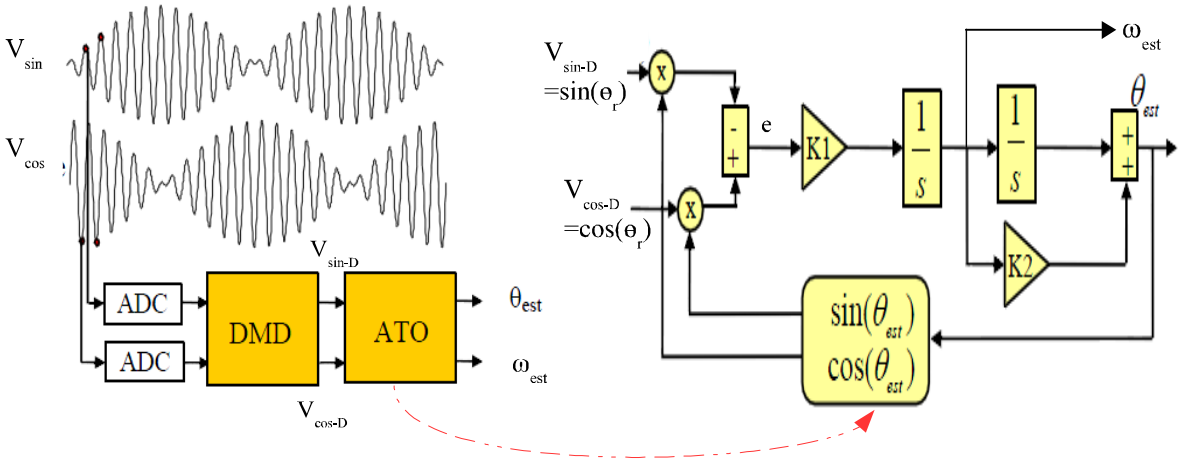


Figure 2.4: RPU principle.

Thus, for small variations of the estimated phase from the actual angle, the observation error can be linearized and be considered equal to $\theta_r - \theta_{est}$.

$$e = \sin(\theta_r - \theta_{est}) = \theta_r - \theta_{est} \tag{2.3}$$

The linearized transfer function of the ATO is then written as follow

$$H(s) = \frac{\theta_{est}(s)}{\theta_r(s)} = \frac{K1.(1 + K2.s)}{s^2 + K1.K2.s + K1} \tag{2.4}$$

It consists in a second order transfer function for which the dynamic of observation is set by $K1$ and $K2$ coefficients. These coefficients are chosen depending on the required estimation speed and on the observation error tolerance [24].

- **Space Vector Modulation (SVM)**

The SVM module consists in adding Zero-squence signal (V_{ZSS}) to the reference voltage in order to increase the linearity range of the modulation. Then, a standard Carrier Based Pulse Width Modulation (CB-PWM) is applied [54]. The aim of this latter is to deliver at each switching period an average phase voltage, to the motor, which is equal to its reference value. Figure 2.5 shows the SVM principle. The ZSS voltage is computed using the 3-phase sinusoidal voltages references (V^*_{sa}, V^*_{sb}, V^*_{sb}) and added at the same time so as to generate the corresponding voltages (V^*_{ao}, V^*_{bo}, V^*_{co}). This allows a full use of the VSI 3-phase supply voltages [131].

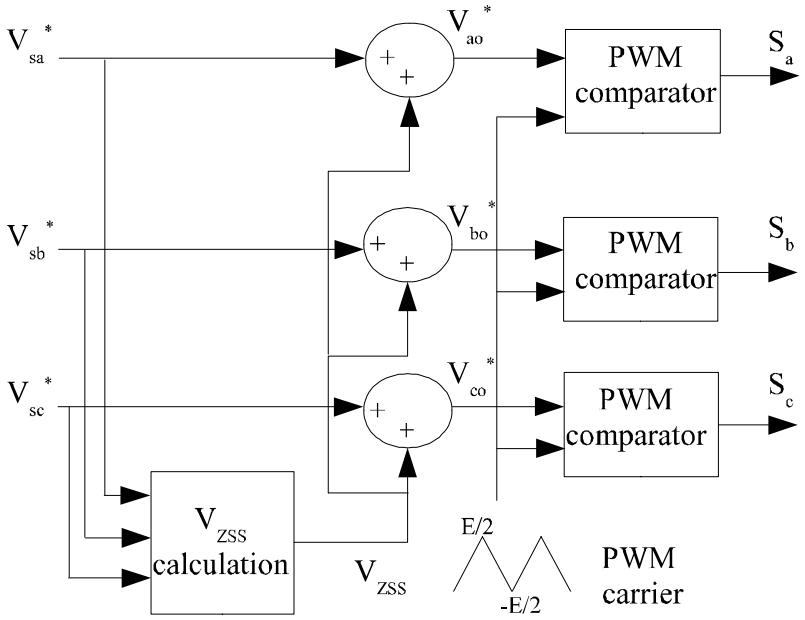


Figure 2.5: SVM principle.

The design of the PI current regulators is performed using the pole compensation method.

Once the control algorithm is designed, the continuous-time domain simulations are performed using Matlab/Simulink permitting the functional verification of the complete control system. Next step is the discretization of the controller using convenient transformation to the discrete-time domain (ZOH, Tustin and Euler). We used the first order Forward Euler approximation method, given by 2.5

$$s = \frac{z-1}{T_s} \tag{2.5}$$

It consists on redesign approach. The sampling period was fixed to 33 μs and the control treatment was executed every sampling period "T_s" which corresponds to the half of switching period.

Figure 2.6 presents the simulation results of RPU module, in discrete domain. It presents the modulated and demodulation signals. The estimated speed and rotor position are also provided. The obtained curves prove the proper functionality of the RPU module. The validation of the current controller was also carried out. The reference of q-axis current reference was fixed to 2 A and changed to 1.2 A at 0.3 s whereas the d-axis current reference was maintain to zero. The simulation results are presented Figure 2.7 to Figure 2.9. The $d-q$ axis current responses are presented showing that the measured current follows correctly its reference. The three phase stator currents and the stationary frame (α, β) based stator currents are also provided. Figure 2.9 presents the reference stator voltage generated after the addition of the zero-sequence signal. The presented results confirm the good functionality of the developed current controller.

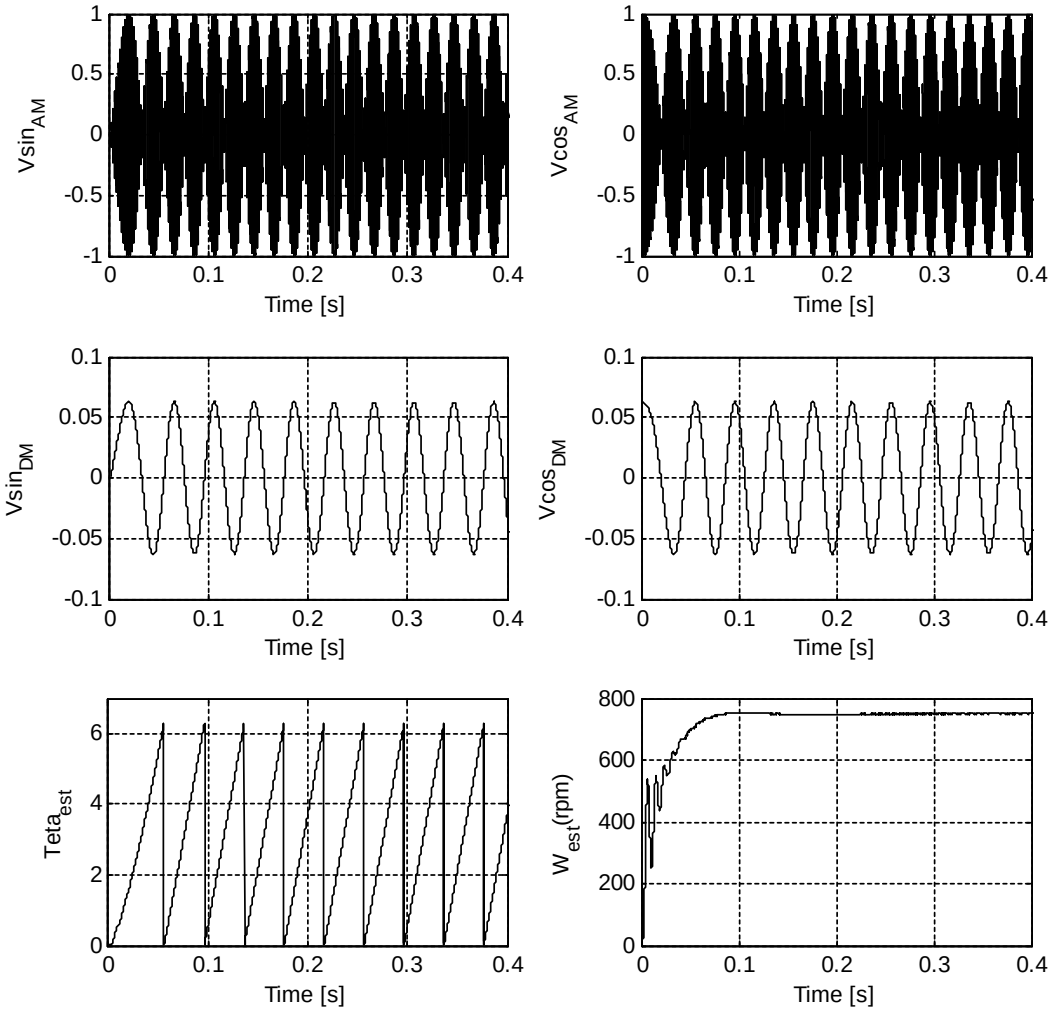


Figure 2.6: RPU simulations results.

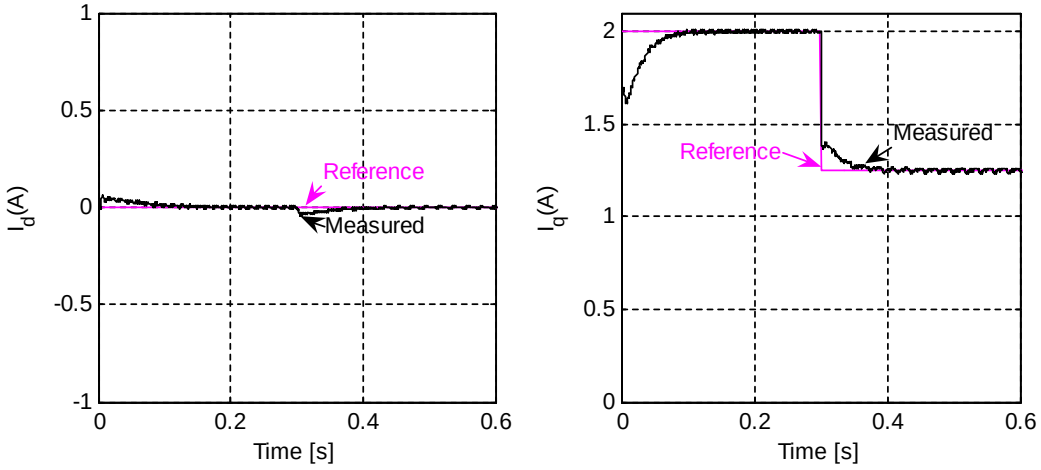


Figure 2.7: d-q axis stator currents.

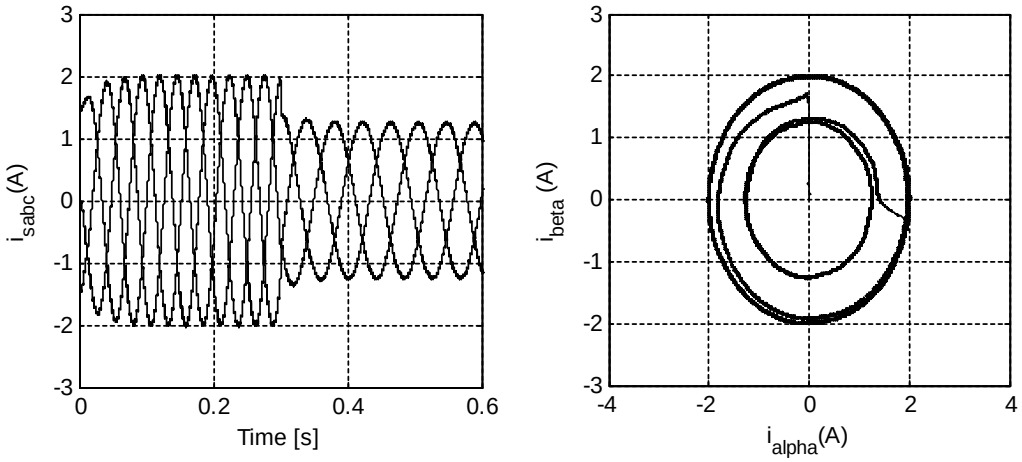


Figure 2.8: Three phase stator currents and the alpha-beta stator current vector locus.

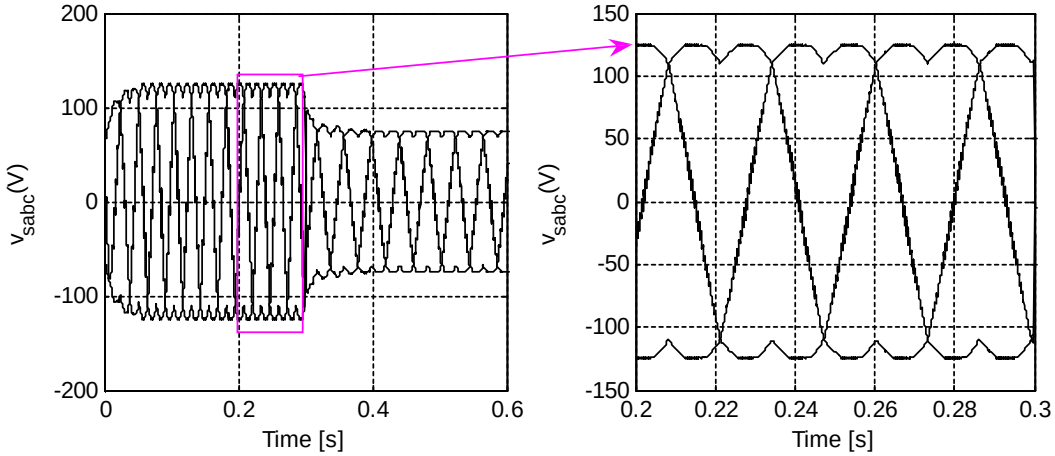


Figure 2.9: Three phase reference stator voltages.

The validation of the speed controller is presented in Figure 2.10 to Figure 2.12. Two speed references are applied "800 rpm and -800 rpm". The three current waveforms and the $\alpha - \beta$ axis currents vector locus are also presented in Figure 2.12. All these results prove that the developed speed controller ensures the desired function at steady and transient state.

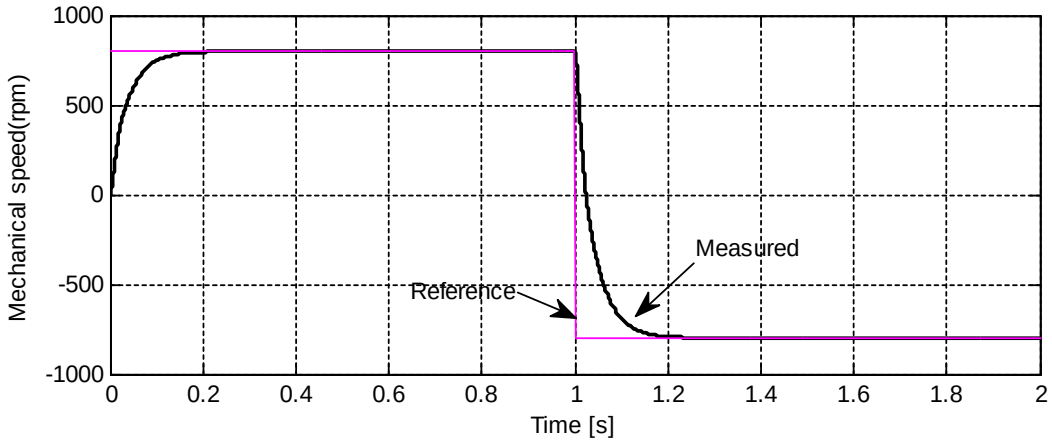


Figure 2.10: Mechanical speed of the speed controller.

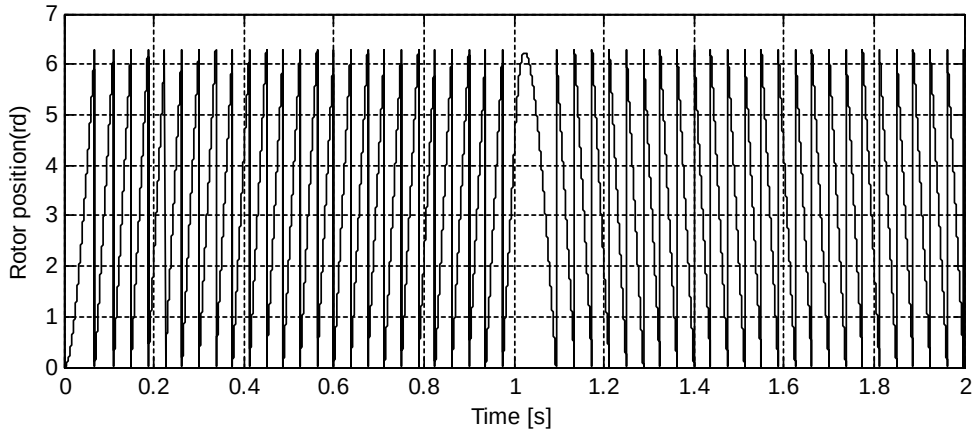


Figure 2.11: Rotor position.

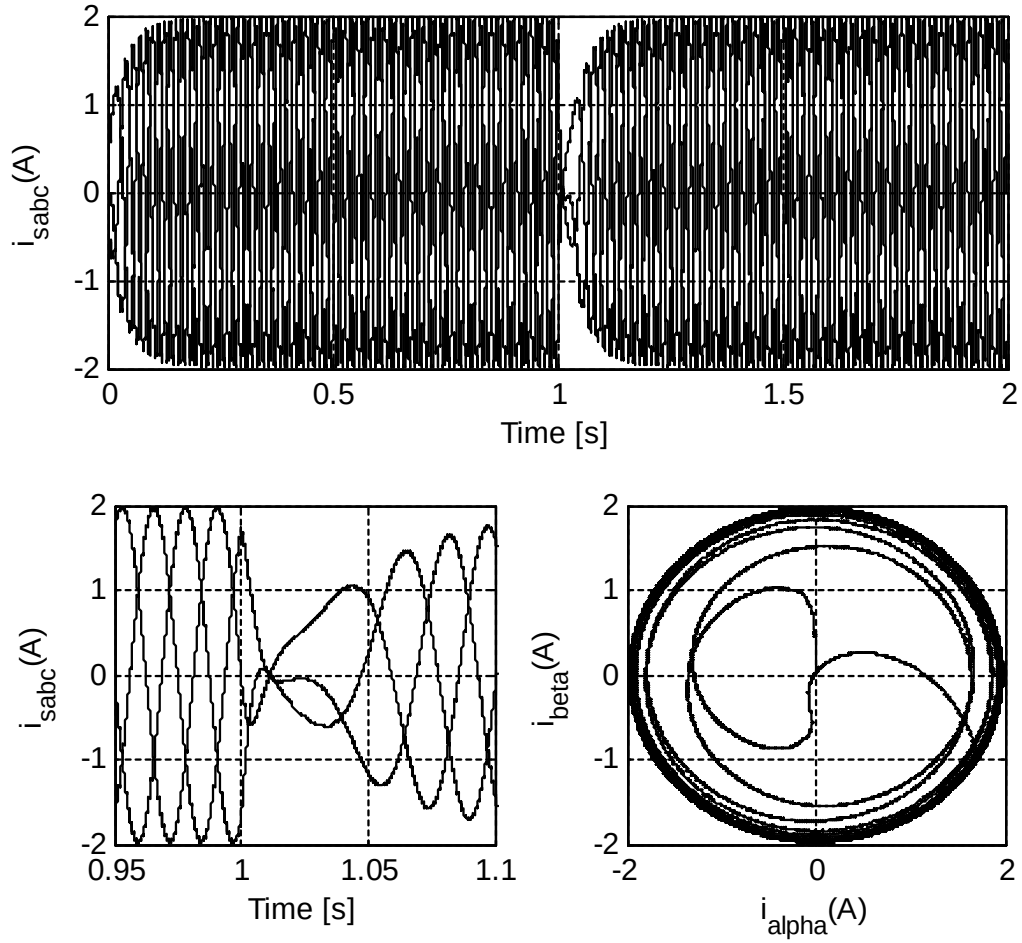


Figure 2.12: Three phase stator currents and the alpha-beta stator current vector locus.

Once the discrete time speed controller is validated, the development of the per-unit and the fixed-point control algorithm is then achieved. The variables are normalized using their base-values. These base values correspond to the rated values of each variables (current rated value "I_B", voltage rated value "V_B", speed rated value "ω_B"...). The sensor/ ADC gains are also taken into account. The used base values are presented in Appendix-D. Then, the quantification effect, on the control performance, of the

fixed-point arithmetic is analyzed. Each variable is coded in two parts: integer "i" and fractional "f" represented as a $(i+f)Q_f$ format. A trade-off between algorithm precision and an optimized use of the FPGA resources must be achieved. The fixed-point format was chosen after several fixed-point simulations tests. These simulations have been performed taken into account the precision and stability criteria. The chosen fixed-point formats for each variables and parameters are presented Figure 2.13.

2.3.2 Architectural design and modular verification of the control architecture

After having validated the control algorithm, the architectural development can be carried out. The first task consists of the development of a factorized hardware architecture derived from the previous step with the help of the A^3 methodology [44]. This folding procedure is applied to the greediest operators in terms of hardware-consumed resources (like multipliers, dividers...) to find out a factorized hardware architecture satisfying the size constraint. This step is essential for aircraft applications where the consumed resources must not exceed 60 % of the available FPGA resources (as stated by the DO-254 standard "Design Assurance Guidance for Airborne Electronic Hardware"). The modularity must be also preseved during this architectural step to guarantee the regularity and reusability of control modules. Next step is the VHDL coding and the corresponding modular verification[55].

Figure 2.13 presents the control architecture that will be implemented using the ASIC and the ProAsic^{Plus} boards. The developed control architecture is composed by the following modules:

- The control modules (SVM, dq-abc transformation, abc-dq transformation, DMD, ATO, Proportional Integral (PI) current regulator and PI speed regulator).
- The ADC-SPI (Serial Peripheral Interface) modules.
- The DAC-SPI module used to send the High-Frequency excitation (Excitation_HF) signal to the resolver.
- The SPI modules which communicate with the CPU board (used to configure the parameters and the segregation paths).

For system reliability purpose, some segregation elements ($segr_1$, $segr_2$, $segr_3$) were added to the data path of the controller in order to keep the system working in the case of any design problem, of one or more control modules. Thus, four control scenarii were provided, shown Table 2.1. The most critical modules are the SVM and SPI modules. Indeed, these 2 blocks must operate properly whatever the circonstances. Other blocks can be bypass by the CPU.

Besides, the CPU board includes the same SPI modules used to configure the ASIC and the ProAsic^{Plus} boards. The decoder module allows the decoding of data sent from/to the Cortex-M1. The communication between the processor and the decoder is ensured by the AHB-Lite and APB buses. In the following, the modular verification of each control module based on the ProAsic^{Plus} board is investigated. The impact of the temperature environment is also taken into account. We note that this is a first validation stage to prepare a first-time-right-silicon of the ASIC board. The hardware design of this latter is ensured by another partner of the SEFORA project.

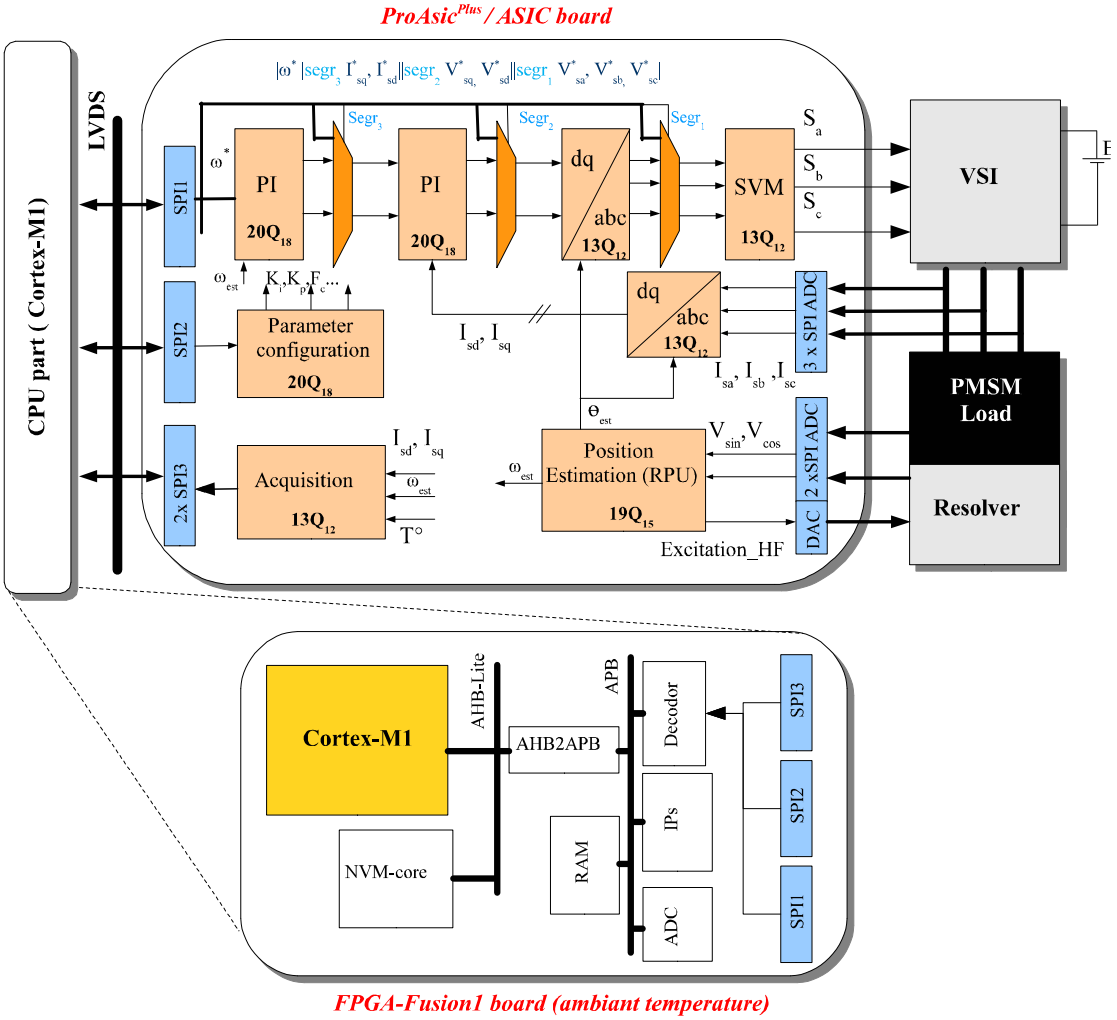


Figure 2.13: The ASIC and the ProAsic^{Plus}controller architecture.

Control scenarii	**criticity Order**	**(segr$_1$, segr$_2$, segr$_3$)**	**Data sent to ASIC / ProAsic^{Plus}**
The whole control architecture	1	(0, 0, 0)	*Speed reference (normal operation)*
$SPI + SVM + RPU$ $dq/abc + abc/dq$ $Current\ PI\ + ADC$	2	(0, 0, 1)	$d-q$ *axis current references*
$SPI + SVM + RPU$ $dq/abc + ADC$	3	(0, 1, 0)	$d-q$ *axis voltage references*
$SPI + SVM$	4	(1, 0, 0)	*Three voltage references*

Table 2.1: Segregation scenarii

2.3.2.1 ProAsic^{Plus} features

The FPGA ProAsic^{Plus} APA1000 family from Actel Company is a good candidate for aircraft applications and can be considered as an alternative to the ASIC. It is a 0.22 μm digital CMOS 4 Layer Metal Flash-Based CMOS Process. It consists on Non-volatile Flash-based FPGA which allows keeping its configuration even when power is off and requires no separate configuration memory. This provides a highly secure feature needed in typical aircraft applications. Furthermore, this technology offers immunity to SEU (Single Event Upset) radiations avoiding the loss of information leading to functional problems. All the indicated features associated to low power consumption make such FPGA device suitable for aircraft control applications [76].

The FPGA consists of a sea of Versatiles Each Versatile can be configured as a three-input logic function (3-bit Look-Up Table LUT), a D-Fip-Fop. Table 2.2 presents the features of the ProAsic^{Plus} APA1000 (208 CQFP).

	APA1000
System gates	1 Million
Versatiles (D Flip-Flops)	56320
RAM Kbits (1024 bits)	198
Max available clock frequency(Military temperature)	33 MHz
Maximum user I/Os	712
Temperature range	$-55\,^{\circ}$C to $125\,^{\circ}$C

Table 2.2: ProAsicPlus APA1000 features

2.3.2.2 Modular verification steps

An ASIC based design flow is proposed aiming to decompose the verification of large design into sub-problems [56]. This choice offers a more manageable complexity and provides less difficult verification procedures. It involves several tasks illustrated as follow:

- **RTL description:** From the factorized architecture described previously, the modules are designed through the development of a data-path and a control unit for each module. After that, the architecture is implemented using VHDL.
- **Functional simulation:** This step verifies the functionality of the RTL design using test vectors for each module. It is completed with a "presynthesis" simulation to verify that the RTL abstraction fully provides the desired functionality. The test vectors used during simulation should provide the coverage necessary to ensure that the design will meet the expected performances.
- **Synthesis:** The design is synthesized and mapped generating the corresponding netlist related to ProAsic^{Plus} library. After that, an accurate "postsynthesis" simulation is performed using the generated netlist.
- **Timing analysis:** Timing constraints (the clock periods, clock uncertainty...) can be fixed with an appropriate timing analysis tool. The most critical path for this constraint and the corresponding 'path slack' can thus be localized. The path slack is the difference between the fixed clock period and the actual path delay.

Thus a positive slack means that the design works with the desired clock frequency, whereas a negative slack indicates a potential failure in the clock synchronization of the corresponding signals.

- **Place and route:** This step implements all the desired netlist connections necessary while following the rules and the limitations of the manufacturing process. Additionally, a Standard Delay Format (SDF) file is created which contains information on the delay of the design elements and interconnects. Simulation of the gate level design using SDF files is performed as next step. This allows the functional verification of the developed architecture related to the considered timing constraints.

Since the board is intended to work in high temperature environment, the impact of temperature was also taken into account. Indeed, it influences the power consumption and the operating frequency. Figure 2.14 shows the frequency variation depending on the temperature values. We can note that the frequency decreases and has a maximum achievable value equals to 27.5 MHz, at junction temperature of 125 °C. These results were measured after the place and route process.

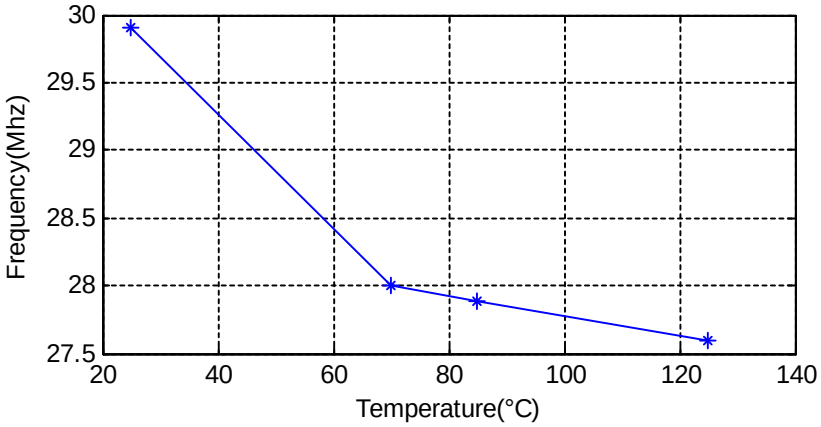


Figure 2.14: The variation of frequency over the temperature.

Clock Gating setup and hold violations, recovery and removal violations must be also verified. For more safety design, a fixed operating frequency equals to 24 MHz has been chosen. Figure 2.15 presents positive slack related with number of tested paths. This proves that the architecture works properly and respects the timing constraints even for the most critical paths. These paths include generally multiplications.

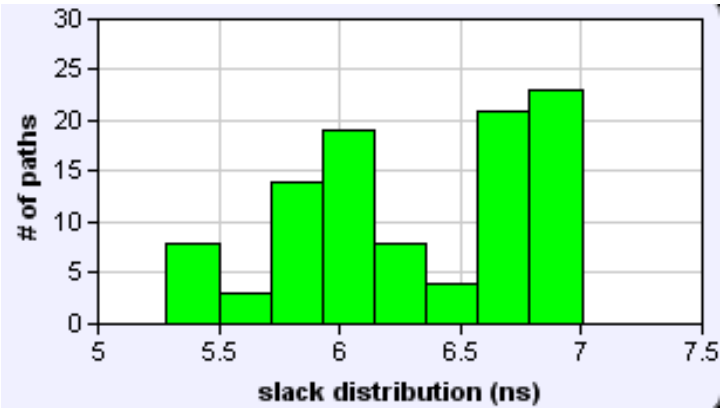


Figure 2.15:Positive slack for Clk=24MHz.

The verification was made for each control module using its own testbench. As an example, the simulation results of dq-abc transformation module are illustrated in Figure 2.16. It presents pre-synthesis, postsynthesis and place and route simulations. The same test bench vectors were used and the following constraints were considered :

- Junction temperature: 125 °C.
- Operating frequency: 24 MHz.
- Clock uncertainty: +/- 2 ns.

The transformation operates properly and the results are the same in the three cases. The glitches width, seen within the place and a route simulation results, are less than one clock period. So, they haven't any influence on the system functionality. These simulation results were performed using Modelsim Tool.

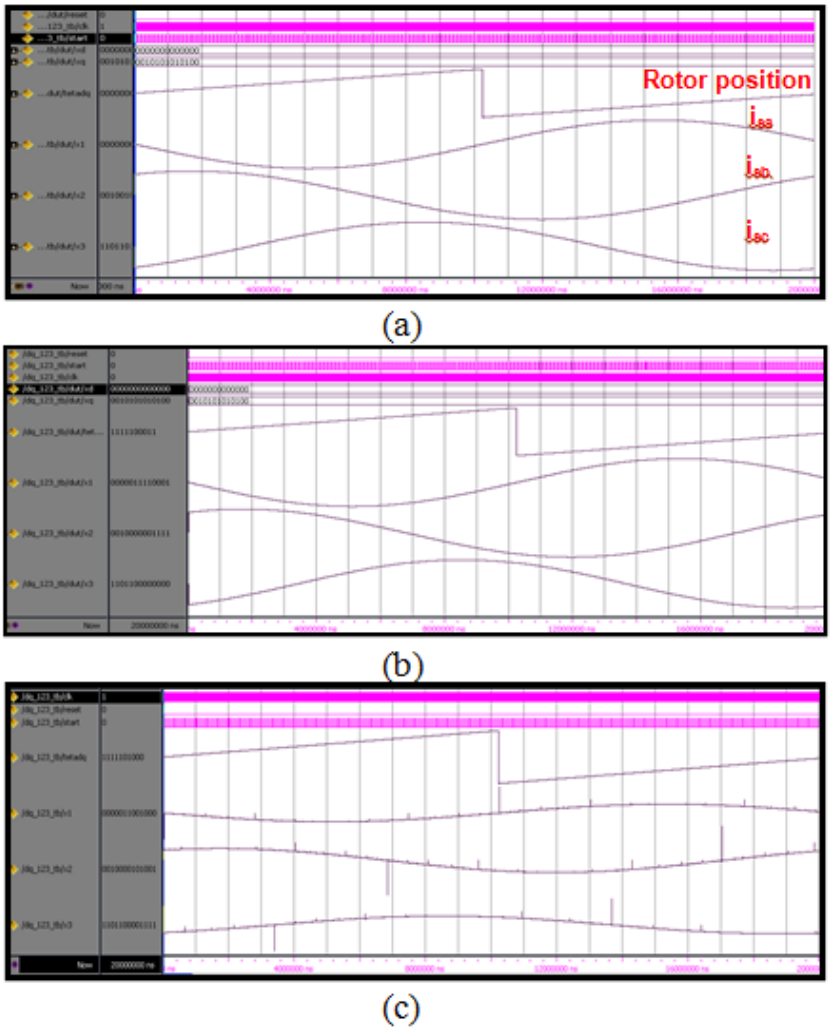


Figure 2.16: Simulation results of: (a) pre-synthesis simulation, (b) post-synthesis simulation, (c) place and route simulation.

2.3.2.3 Time and Area performances

Table 2.3 shows the FPGA time/area performances of the speed controller architecture. The consumed resources rate is equal to 63.22% of the available FPGA ProAsic^{Plus} resource (56320 Tiles). This result was obtained for a 13-bit fixed-point format for the

conversion module and transformation, 20-bit fixed-point format for PI current and speed regulator and 19-bit fixed-point format for the RTU module.

	Consumed resources (Tiles)	**Execution time (Clock cycles)**	**Number of modules**
SVM	2127	4	1
ADC – SPI	1952	16	3
dq/abc	2352	27	1
abc/dq	2730	25	1
DAC/ADC – SPI RPU(DMD + ATO)	6265	42	1
PI regulator (speed/curent)	3718	14	3
SPI Acquisition	735	16	2
SPI Configuration	1827	16	2

Table 2.3: FPGA Time/Area performances of the control architecture

The sequential timing diagram of the control architecture is shown Figure 2.17. The sampling period "T_s" is fixed to 33 μs which corresponds to half of the switching period.

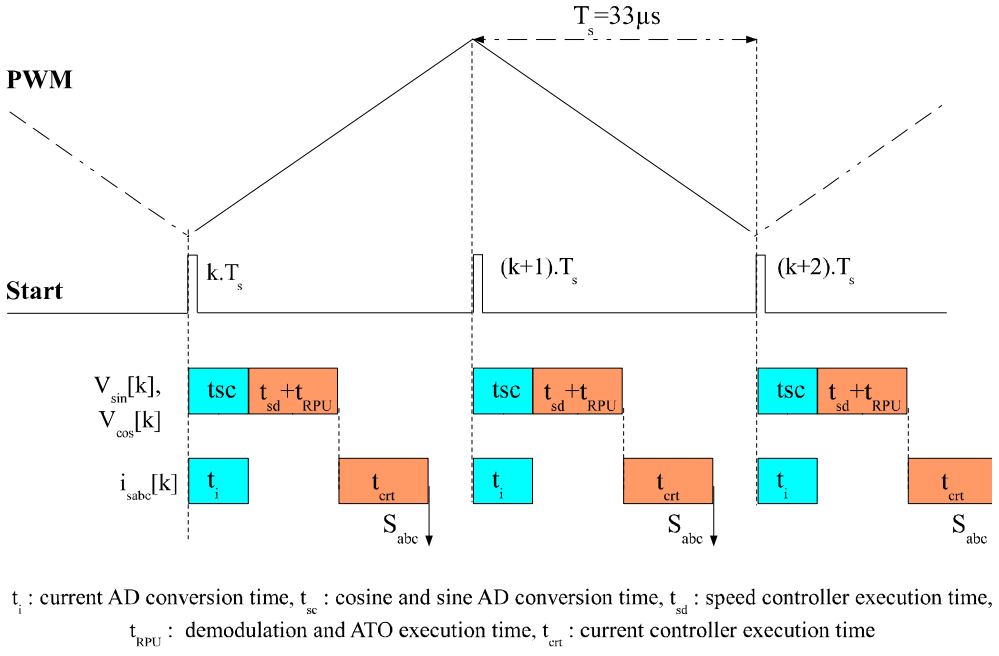


Figure 2.17: Timing diagram of controller.

2.3.3 Real-time simulation

In this section, author proposes to validate the designed controller by testing it with an emulator of the physical electrical system to be controlled. As shown in Figure 2.2, the

proposed emulator was written in VHDL and implemented also in the FPGA. Using this emulator, many validations can be performed to guarantee a first successful experimental test on the final system. The emulator is composed of the following parts:

2.3.3.1 Voltage Source Inverter (VSI) model

In order to represent the inverter in real-time, a model with high computational efficiency is of great interest. Indeed, models used for representing inverters can take various forms (ideal, average or switching models) [50]. In our approach, the inverter model is represented by switching function formulation taking into account the dead time. A delay between the two switches of the same inverter leg is introduced in order to avoid any short-circuit. Dead time is fixed here to 3 μs. The used model is described by the following expressions

$$\begin{cases} V_{aN} = \frac{1}{3}.\left(2.V_{a\_o} - \left(V_{b\_o} + V_{c\_o}\right)\right) \\ V_{bN} = \frac{1}{3}.\left(2.V_{b\_o} - \left(V_{a\_o} + V_{c\_o}\right)\right) \\ V_{cN} = \frac{1}{3}.\left(2.V_{c\_o} - \left(V_{b\_o} + V_{a\_o}\right)\right) \end{cases} \tag{2.6}$$

$$\begin{cases} V_{a\_o} = \frac{E}{2}.\left(S'_a - S''_a + \left(S'_a - S''_a - 1\right).sign(i_{sa})\right) \\ V_{a\_o} = \frac{E}{2}.\left(S'_b - S''_b + \left(S'_b - S''_b - 1\right).sign(i_{sb})\right) \\ V_{a\_o} = \frac{E}{2}.\left(S'_c - S''_c + \left(S'_c - S''_c - 1\right).sign(i_{sc})\right) \end{cases} \tag{2.7}$$

V_{i_o} , V_{iN} , i_i , E, S'_i and S''_i are respectively the pole voltage, the phase voltage, the phase current ($i = a,\ b,\ c$), the DC link voltage and the switching functions of IGBTs of the same inverter leg. The developed inverter module can be used for real-time simulation and Hardware-In-Loop. It can be also integrated under verification chains of fault tolerant control algorithms or even to test new control strategies [10].

2.3.3.2 Machine model

The emulator of the synchronous actuator is based on the normalized state space model according to the relation 2.8. This model is derived under the so-called "infinite inertia" hypothesis which means that the speed is constant regarding the dynamic of electrical quantities. The discretization and the normalization are performed respectively using the Euler approach (for T_s=33 μs) and the base-values presented in Appendix-D.

$$\begin{aligned} x_{nk} &= f\left(x_{nk-1}, u_{nk-1}\right) \\ y_{nk} &= H.x_{nk} \end{aligned} \tag{2.8}$$

Where x_n is the normalized state space vector and $u_n,\ y_n$ are respectively the normalized system input and output vectors. The $f\left(x, u\right)$ is the non-linear function that describes the model.

$$x_n = \begin{bmatrix} \frac{i_{sd}}{I_B} & \frac{i_{sq}}{I_B} & \frac{\theta_e}{I_B} & \frac{\omega_e}{I_B} \end{bmatrix}^T ; \quad u_n = \begin{bmatrix} \frac{V_{sd}}{V_B} & \frac{V_{sq}}{V_B} \end{bmatrix}^T ; \quad y_n = \begin{bmatrix} \frac{i_{sd}}{V_B} & \frac{i_{sq}}{V_B} \end{bmatrix}^T$$

$$f(x_n, u_n) = \begin{bmatrix} -\frac{R_s}{L_{sd}}.i_{sd\_n} + \frac{L_{sq}}{L_{sd}}.\omega_B.\omega_{e\_n}.i_{sq\_n} \\ -\frac{R_s}{L_{sq}}.i_{sq\_n} - (\frac{L_{sd}}{L_{sq}}.i_{sd\_n} + \frac{M_{sr}.I_{rd\_n}}{L_{sq}}).\omega_B.\omega_{e\_n} \\ 0 \\ \frac{\omega_B}{\theta_B}.\omega_{e\_n} \end{bmatrix} + \begin{bmatrix} \frac{V_B}{I_B.L_{sd}} & 0 \\ 0 & \frac{V_B}{I_B.L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} .u_n$$

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T$$

2.3.3.3 Resolver model

The resolver model was provided according to relation 2.1. The reference excitation winding is supplied by a high frequency square wave signal (10 kHz). The amplitude of the outputs is modulated respectively with the sine and cosine of the electrical shaft angle.

2.3.3.4 CAN model

The models of DAC/ADC were also developed. It consists of AD7466 (12bit, successive approximation) and DA5624 which are well-adapted to high temperature environment [57]. For the ADC, the conversion process and the corresponding data acquisition are controlled using CS (Chip Select) and the serial clock. The input signal is sampled on the falling edge of CS, and the conversion is also initiated at this point. This conversion process is controlled using a Finite State Machines (FSM).

2.3.3.5 Real-time simulation results

To demonstrate the validity of the proposed approach, real-time simulations have been performed. In Figure 2.18, curves show the modulated output signals of the resolver, the demodulation signals and the extracted rotor position. The $d-q$ axis current responses and regulated three stator currents and the estimated rotor position are provided Figure 2.19. The static and dynamic performances are validated through the reference change from 2 A to -2A. The SVM voltage references and the VSI voltage outputs are also presented in Figure 2.20 and Figure 2.21. Real-time simulation results prove the good functionality of the developed control architecture. It demonstrates also the interest of the proposed methodology to test the controller associated to the emulator of the electrical system to be controlled.

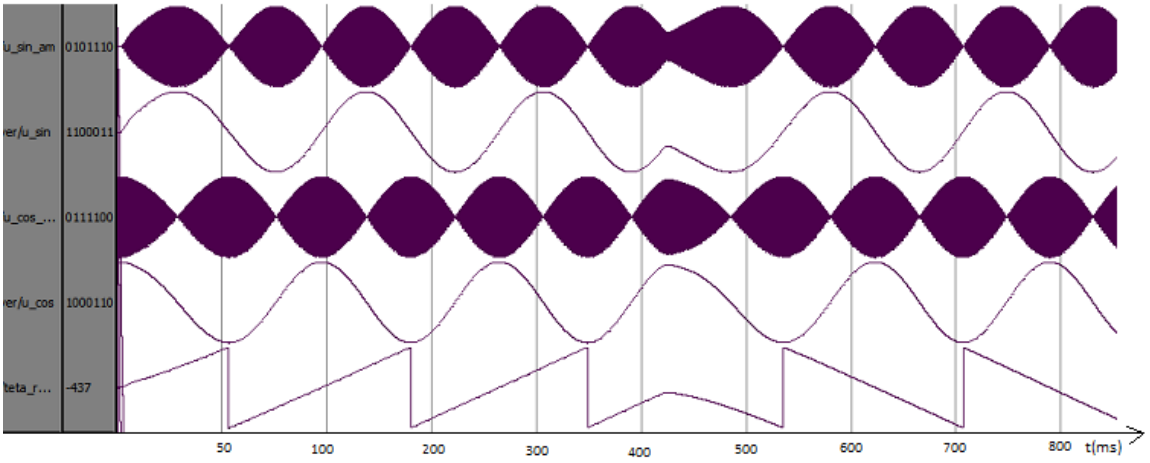


Figure 2.18: Modulated and demodulated resolver signals and the estimated rotor position.

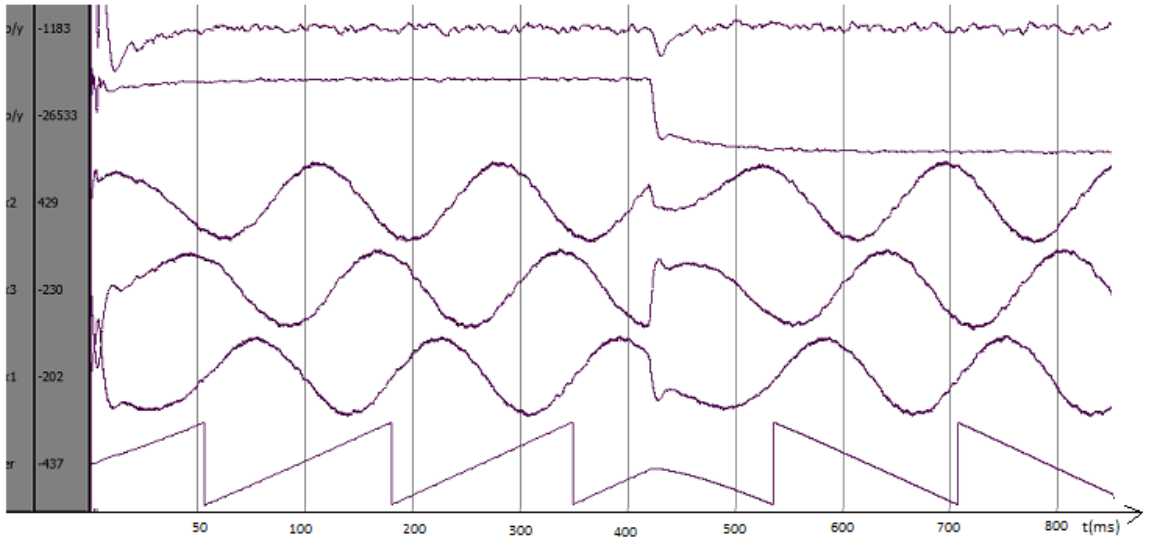


Figure 2.19: The stator currents.

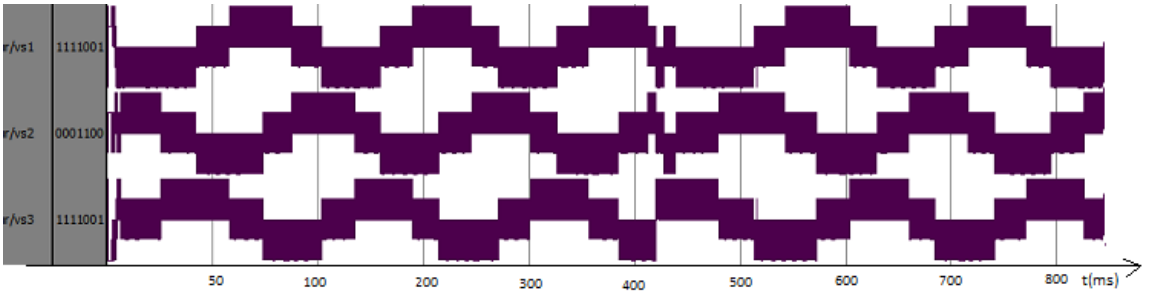


Figure 2.20: The VSI output voltages.

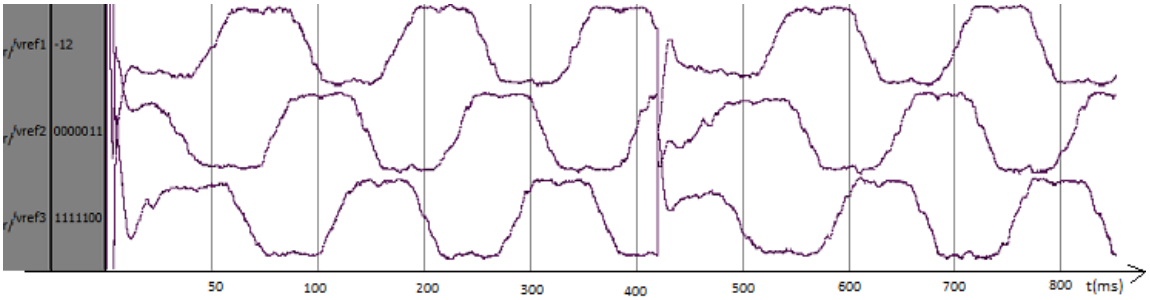


Figure 2.21: The SVM reference voltages.

2.4 SoC performance evaluation

The performance evaluation of the SoC Fusion-1 solution, embedded a Cortex-M1, was then investigated. An ON/OFF current control strategy was chosen [11]. This control strategy is simpler than the implemented in hardware (see section 2.3.2). This choice is due to the poor timing performances of the Cortex-M1. The current acquisition is performed using the integrated ADC. Whereas, the control algorithm still implemented on the Cortex-M1. Designing a SoC application requires the use of two main procedures: Standard FPGA HW Development Flow and Standard Embedded SW Development Flow Compiler. The first one consists of the traditional design flow of an FPGA including the design, synthesis, place and route steps. The second one includes the development and the verification of the software part (see chapter 1).

The Cortex-M1 is a 32-bit ARM processor designed for FPGA implementation. With a balance between size and speed, this processor operates at up to 65 MHz and can be implemented in few tiles. A streamlined three-stage pipeline solution, the Cortex-M1 runs a subset of the classic Thumb®$^{-2}$ instruction set [76].

The proposed SoC-based architecture is presented Figure 2.22. It is divided into four main parts :

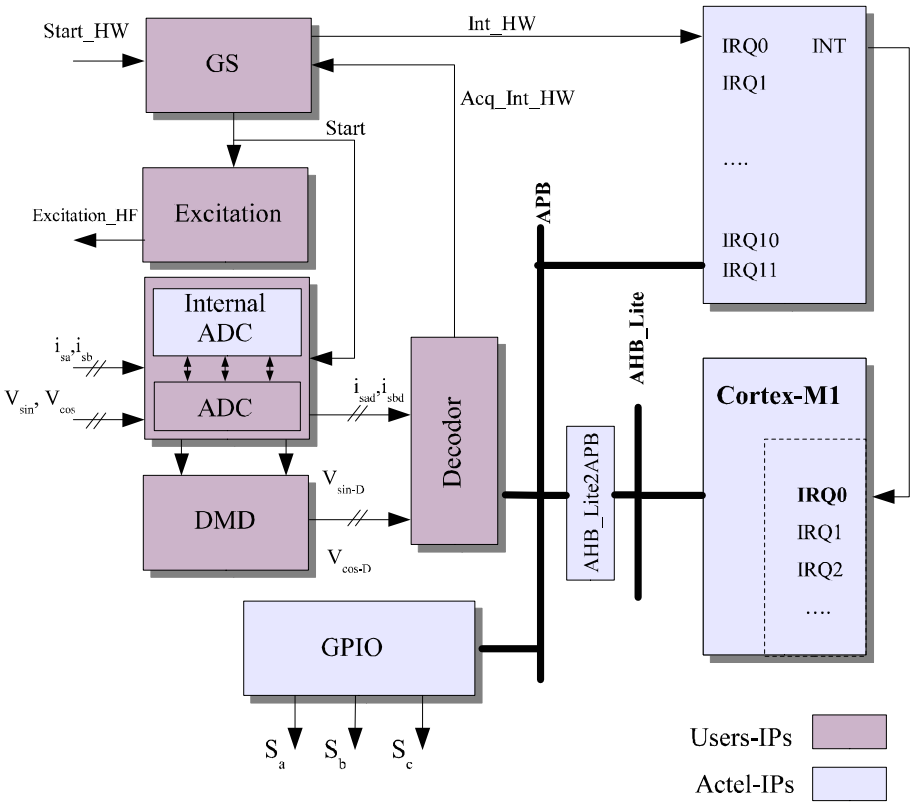


Figure 2.22: Current controller Cortex-M1-based architecture.

- **Configuration part:** This part ensures the configuration of the soft processor core and its peripherals including the following elements:

  - The core interrupt is used to manage up to 32 external interruptions.

-The AMBA AHB-Lite and APB are bus interfaces of the processor that support a single bus master and provide high-bandwidth operation. All AHB types support transfer sizes of 8, 16 and 32-bits data.

- The core GPIO allows sending data from processor to external users FPGA pins.

-The core ADC guarantees a 12-bit programmable successive conversion of analog measurement. Figure 2.23 shows the architecture of the used "ADC module". The conversion process is driven by a Finite State Machine (FSM).

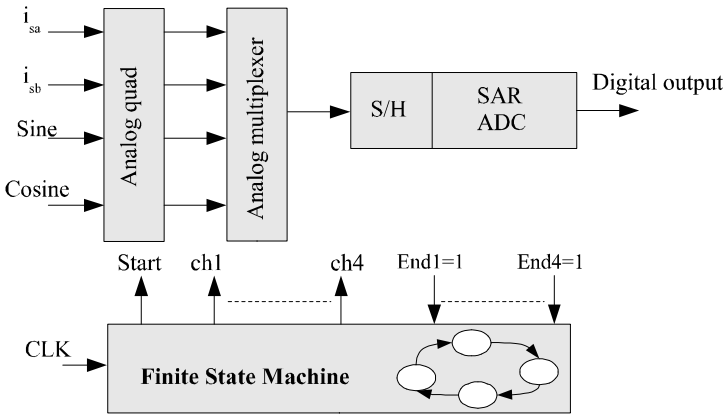


Figure 2.23: Integrated ADC module.

- **Decoder part:** The communication between the processor and the FPGA matrix is a critical point of the system, since the treatment efficiency depends on it. Therefore, a hardware/software interface was developed allowing the communication between the two parts. It is based on the 32-bit decoder mapping processor address space where the different registers of the control modules are placed. It must be consistent with the APB bus protocol in terms of bit width and corresponding signals. Figure 2.24 presents the developed decoder for In/Out exchanges between the processor and the FPGA programmable elements.

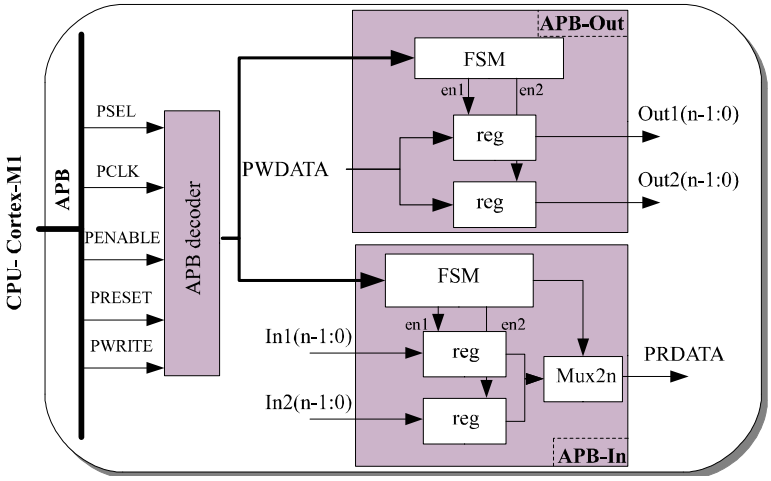


Figure 2.24: The architecture of the APB In/Out decoder (for 32-bit).

- **Software part:** This part includes the codes of the current control algorithm and ADC configuration written in C language. A fixed point format of $32Q_{30}$ was chosen, with a word length of 32-bit. A Hardware Abstraction Layer (HAL) was developed to provide an abstraction layer between the physical hardware and the software that runs on the Cortex-M1.
- **Synchronization part:** The synchronization between the processor and the FPGA matrix is made as follow. Initially, the FPGA and the processor are started in an independent way. The FPGA begins to generate the start signal "Start-HW" with a period T_s equal to 100 μs, whereas the processor, after the initialization process, remains in idle state waiting for external interruptions coming from the HW modules implemented in the FPGA matrix. In the next high level of start signal, the processor activates the Global Sequencer (GS) that sends an active start "Start" pulse to the excitation and the ADC modules. Once the measured signals are sampled, an interruption signal (Acq-Int-HW) is transferred to the GS, which in turn activates the core interruption module using signal Int-HW. This latter activates the Interrupt Service Routine (ISR) in the processor, presented Figure 2.25. Then, the program begins by the reading of data coming from the ADC and the execution of the current control modules (ATO, dq-abc transformation and ON/OFF regulator). When the computation time is over, the deleting of the interruption signal indicates that the switching signals are ready and the processor is waiting for the next interruption. The synchronization of all blocks is ensured by the general clock signal (Clk).

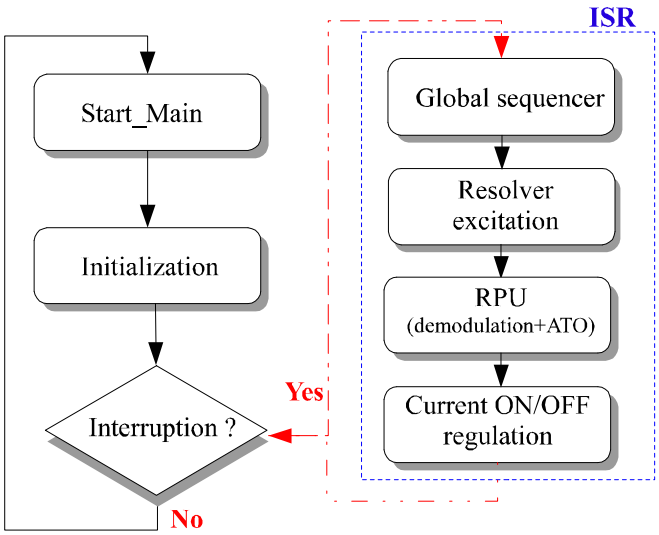


Figure 2.25: SoC execution diagram using Cortex-M1.

Time and Area performances

The time and area performances of the SoC-based architecture are presented Table 2.4. The algorithm was implemented with 32-bit fixed-point format for the current controller and the ATO process. We note that the architecture consumes 82.9% of the Fusion-1 which includes 13824 Tiles and where the Cortex-M1 takes by itself 51.6%. The global execution time is equal to 38.84 μs.

	Execution time
ADC	5.44 μs
Hysteresis regulator	2 μs
DMD+ATO	17.5 μs
Core interrupt	2.32 μs
2x Core GPIO	5 μs
Interface	1.3 μs
dq/abc transformation	8 μs
Total execution time	38.84 μs.
Total consumed resources	82.9% (51.6% for Cortex-M1)

Table 2.4: Time and area performances (Fusion M1AF600, 13824 Tiles, 50MHZ)

The developed SoC-based architecture was tested experimentally using a laboratory experimental set-up presented Figure 2.26.

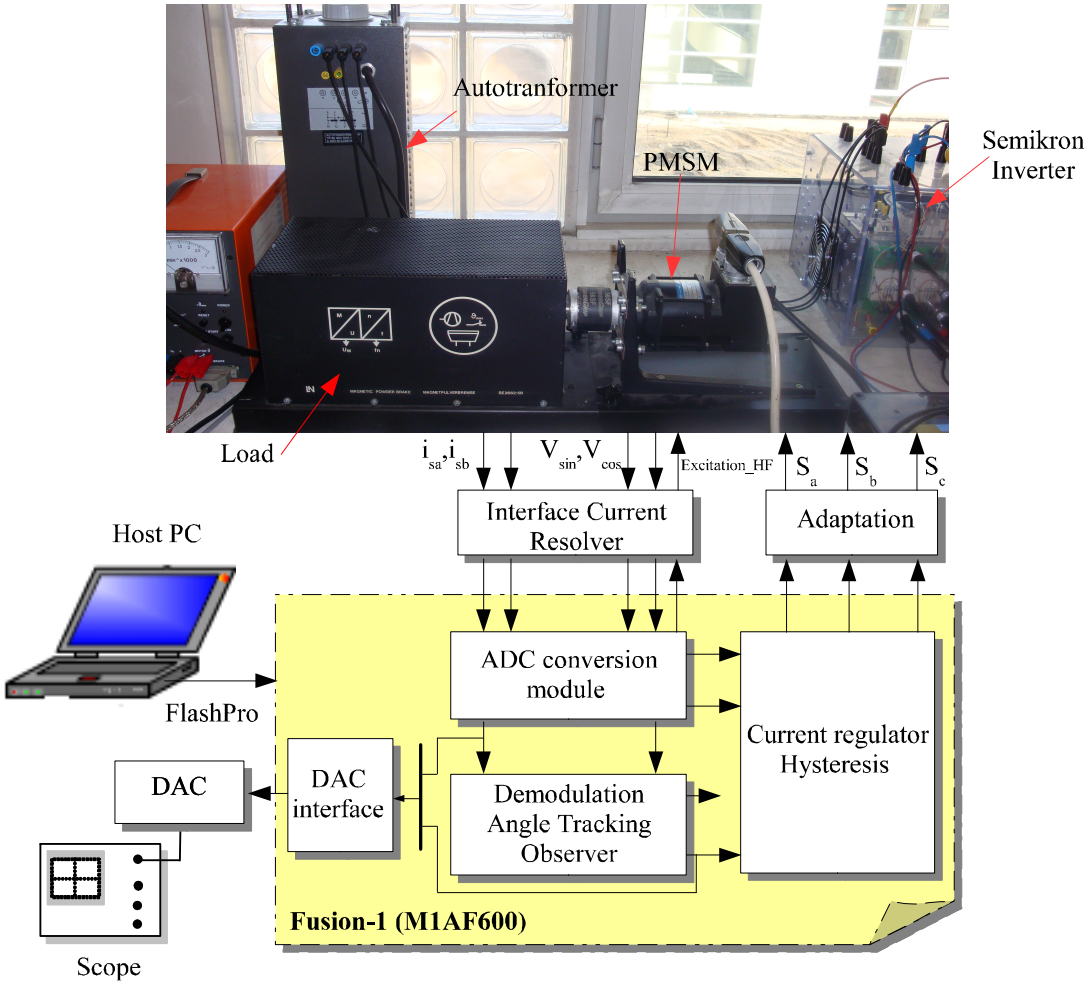


Figure 2.26: Experimental set-up.

The experimental set up is composed by:

- *Machine and load:* It consists on a PMSM : 800W, 220V, 2.5Nm, 3000 tr/min, resolver and mechanical Load (powder brake).
- *Power stage:* It includes a SEMIKRON inverter and an autotransformer.
- *Interface boards:* It consists on resolver interface boards and LEM LA-25NP sensor.
- *Control board:* It is composed by the Fusion-1 AFS600 including an integrated ADC and soft processor core "Cortex-M1".

  During experimentation, the DC-link voltage of the VSI and the hysteresis bandwidth (Bw) were set to 200V and 0A respectively. The experimental validation of the RPU is presented Figures 2.27 and 2.28. The demodulated signals have the same frequency of the resolver signals and the rotor position is extracted. Figure 2.29 presents the regulated stator currents. The experimental current THD is equal to 25.2%. This is due to the large value of the chosen sampling period, this choice being the direct consequence of the poor timing performances of the Cortex-M1.

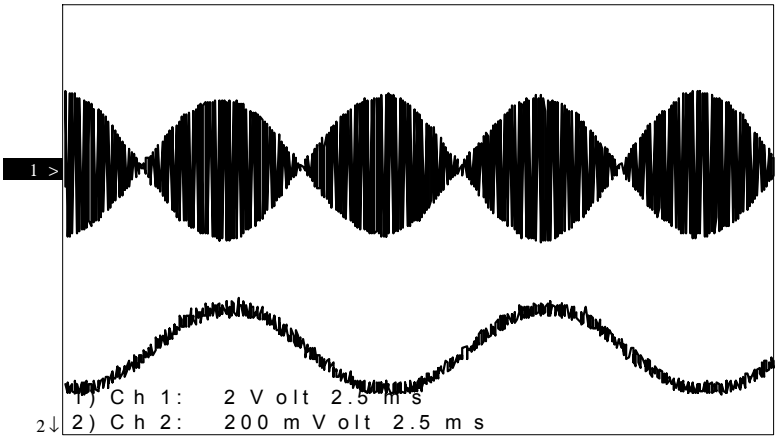


Figure 2.27: Modulated and demodulated signals of the resolver.

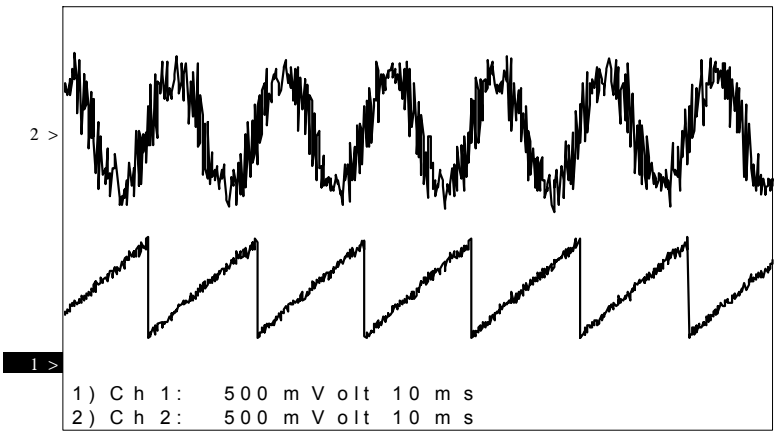


Figure 2.28: Stator current and the estimated rotor position.

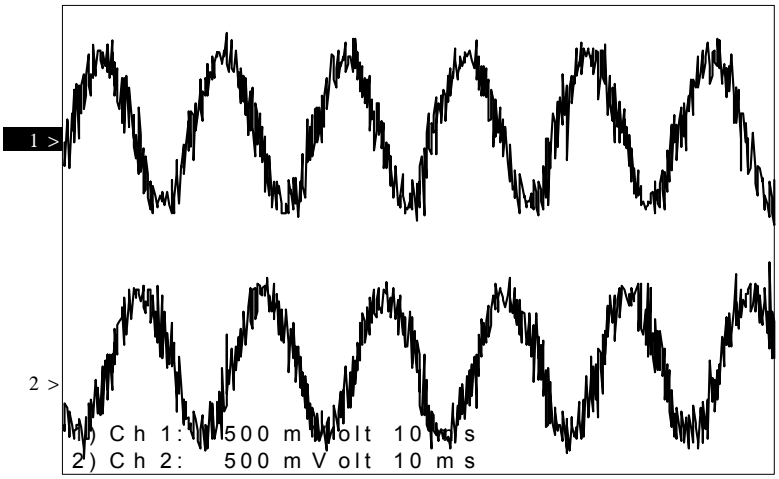


Figure 2.29: Regulated stator currents (I^*_{sd}=0A, I^*_{sq}=3A, Bw*=0A) with 250mV corresponds to 1A.

2.5 Conclusion

This chapter has dealt with the design and the validation of FPGA-based motor drives working at high-temperature (200°C). It consists in a Field Oriented Controller for a PMSM associated with a resolver. To this purpose, author has proposed a rigorous methodology based on three steps: functional validation of control algorithm under Matlab-Simulink, architectural design and its modular validation and finally real-time simulation. As first stage of validation, this method was performed using the ProAsic^{Plus} board from Actel/Microsemi Company. The temperature impact was also analyzed and the maximum system clock frequency was limited to 27.5 MHz, for a junction temperature set to 125°C. These tests allow the preparation of the first-time-right silicon ASIC board. The design of this latter is ensured by another partner of SEFORA project.

The evaluation of the SoC capability for control applications has been also focused on. It has been decided to use the SoC Fusion-1 from Actel family. It is characterized by a mixed signal elements such hardware architecture (FPGA matrix), internal Analog Digital Converter (ADC) and soft processor core (Cortex-M1). A single control algorithm was implemented. It consists in a current control algorithm based on ON/OFF regulator and a Resolver Processing Unit (RPU) that ensures the estimation of the speed and rotor position. Experimental results prove a correct functionality of the control. However, they also demonstrate the limits of a such implementation.

One interesting solution will be the use of hardware accelerators to boost the software treatment. This requires the use of a rigorous HW-SW partitioning method. The Co-design approach has to take into account all functional (control performances) and architectural (SoC available resources) constraints. Thus, the next chapters will deal with HW-SW partitioning of control modules. As a benchmark, a sensorless controller using Extended Kalman Filter (EKF) has been chosen. Although it is more complex, this kind of controller is a realistic future alternative in avionic applications to cope with reliability problems of the position sensor [114].

Chapter 3

Specifications and algorithm development-Time delay impact

Chapter 3

Specifications and algorithm development-Time delay impact

3.1 Introduction

Nowadays, the digital control presents a good alternative to the analog control. This is due to their potential advantages such as high integration, the possibility to implement more sophisticated algorithm, re-programmability... However, the digital control has some limitations, such as quantization error [58] and time delay. This time delay is mainly composed by two elements: the computation time delay and the Sample and Hold element caused by the PWM.

Many investigations have been under taken to analyze the time delay effects on the control performances [59]-[64]. It has been demonstrated that time delay affects significantly the quality of the transient response and the control performances. The majority of these analysis was qualitative and not quantitative. Explicit relationships between the time delay, the sampling period and the used digital platform to execute the control algorithm were not well investigated up to now.

In this chapter, author is focusing on the quantitative analysis of the time delay impact in the control performances. To this purpose, two AC drive systems (high and low rated speed) and two switching frequencies (high and low rated frequency: 20 kHz and 100 kHz) were considered. As for control strategy, a sensorless speed controller based on the Extended Kalman Filter (EKF) is adopted. Firstly, the source of time delay and its relationship with the used digital platform (FPGA or DSP/μC) are highlighted. The maximum control bandwidth was derived depending on the sampling period and the specified phase margin. Then, the time delay impact is analyzed in time and frequency domains. Secondly, the EKF is proposed and its algorithm complexity is discussed. Finally, the discretization and the fixed-point format setting of the sensorless speed controller are presented.

3.2 Specifications

The considered control system is presented by Figure 3.1. It consists on the power stage, the digital control unit, the sensors, the ADC boards and the Host-PC interface. The adopted control strategy consists on a sensorless speed controller based on the Extended Kalman Filter (EKF). This controller is composed by three main parts: the current

control loop, the speed control loop and the speed and rotor position observer. The development of all these parts will be discussed in the next sections.

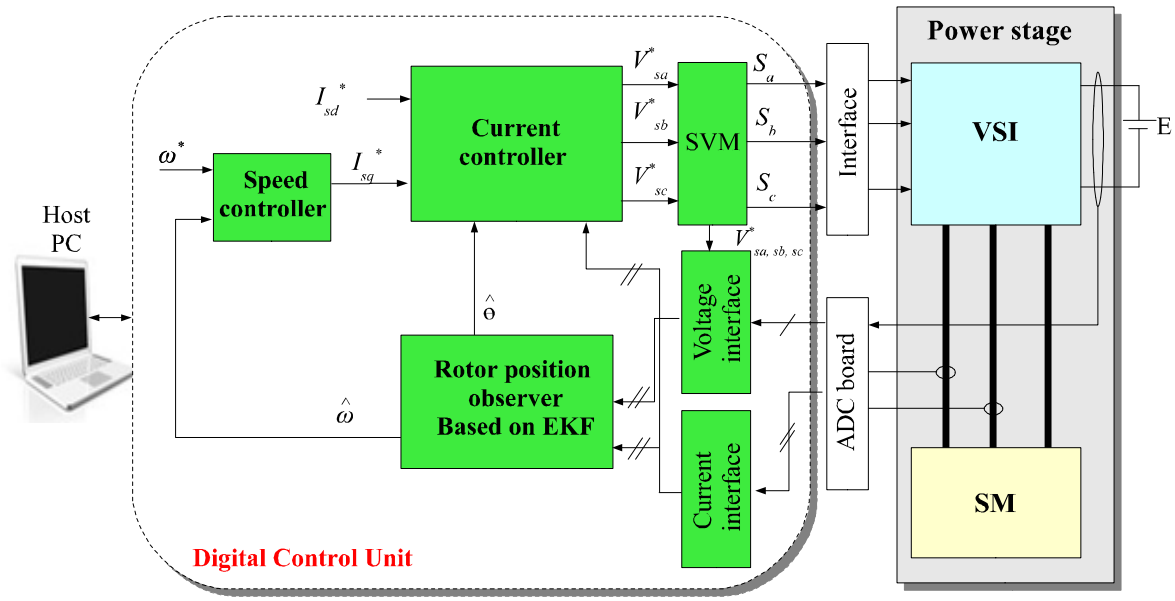


Figure 3.1: Synoptic of the sensorless control system

3.2.1 Power stage

The considered power stage is based on two case studies. It consists on two AC drive systems : low and high rated speed. The first one is a standard low rated speed synchronous machine (S_l). The range of speed goes from 0 up to 1500 rpm. The second motor is a high rated speed synchronous machine (S_h). It consists on Starter Generator that is used in aircraft applications. Its range of speed varies from 0 up to 8000 rpm. These systems were chosen to study their differences of behavior with respect to control performances, bandwidth and time delay effects. The parameters of the two considered systems are presented respectively in Table 3.1 and Table 3.2.

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 $M_{sr} = 0.86\ H$
q axis stator inductance $L_{sq} = 0.229\ H$	Nominal stator current $I_{sn} = 2.12\ A$

Table 3.1: Synchronous Machine Parameters - Low speed AC Drive 0.8 KVA, 220V,1.5A, 50 Hz, 3 Phases, Y connection, 2 pole pairs, 1500 rpm

Stator resistance $R_s = 11\ m\Omega$	Rotor resistance $R_r = 0.34\ \Omega$
d axis stator inductance $L_{sd} = 0.7\ mH$	Mutual inductance $M_{sr} = 5.8\ mH$
q axis stator inductance $L_{sq} = 0.6\ mH$	Nominal stator current $I_{sn} = 290\ A$

Table 3.2: Starter Generator Parameters - High speed AC Drive 200 KVA, 230V, 290A, 380 Hz, 3 Phases, Y connection, 3 pole pairs, 8000 rpm

Additionally, two switching frequencies "F_{sw}" are considered. The choice of the frequencies is based on the following classification [65]:

- High demanding application: As shown in Table 3.2, the inductances of the Starter Generator are very low which provide a fast current rise. This permits to operate with high bandwidth. But, these systems are generally characterized by more current ripples. To decrease these latter, high switching frequencies is a good solution. Unfortunately, the switching losses are rather high. In our case, the sampling frequency was set to 100 kHz. This requires the use of efficient digital platforms to ensure the control of the system with this severe timing constraint (T_{sw}=10 μs).

-Constrained switching frequency application: It consists in application where sampling is not critical due to switching frequency limitation. Here, the switching frequency is set to 20 kHz . This is a typical switching frequency value found in laboratory. This is the case of the low rated speed system presented in Table 3.1.

3.2.2 Measurement boards

To provide the feedback measurements to the digital control unit, two acquisition boards are used. The first one includes sensors, it is based on the ARCTU3 board. It gives the voltage signals that correspond to the stator currents and the DC-link voltage (2.5V/10A, 1V/100V). Once the measurements are achieved, the Analog-Digital (AD) conversion can begin. The used AD converters are AD9221. They are 12-bit successive approximation register components. The clock frequency and the control signal needed for these conversions are provided by the digital control unit. The total conversion time is equal to 2.4 μs.

3.2.3 Digital Control Unit

The choice of the digital platform where the controller will be implemented is of prime importance for digital control applications. Indeed, the control performances depend on it. The used digital control unit is based on a FPGA platform from Xilinx Company. It consists on the Virtex-5 (XC5VSX50T) embedding the "Microblaze" soft processor core. The design architecture and the optimal partitioning of control modules will be fully discussed in the next chapter.

3.3 Sources of time delay

3.3.1 Computation time delay

The computation time delay consists on the time needed by the controller to execute the control algorithm. Depending on the used digital platform, this time delay takes classically a sampling period "T_s" for a DSP/μC implementation and only a fraction of T_s in the case of FPGA implementation. The Laplace transfer function of this time delay is

$$G_{cp}(s) = e^{-s.T_{Alg}} \tag{3.1}$$

Where "T_{Alg}" represents the computation time delay of the control algorithm.

This time delay can be expressed quite accurately using a second order Pade approximation yielding

$$e^{-s.T_{A\lg}} = \frac{1 - \frac{T_{A\lg}}{2} + \frac{(T_{A\lg})^2}{8}}{1 + \frac{T_{A\lg}}{2} + \frac{(T_{A\lg})^2}{8}} \tag{3.2}$$

Based on equation (3.2), the magnitude and the phase of different time delays are shown Figure 3.2. The time delay implies a phase lag while no magnitude impact is noted. Thus, the addition of a pure delay in the control loop will impact mainly the stability of the controlled system.

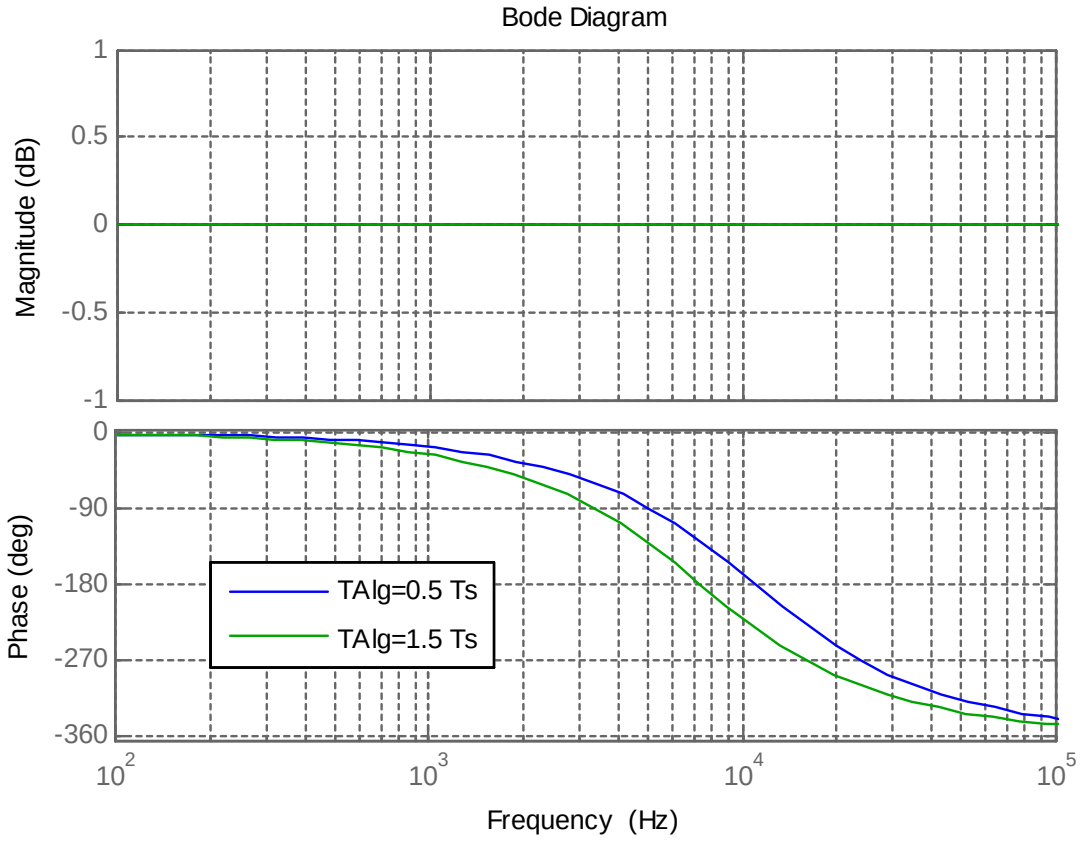


Figure 3.2: The magnitude and the phase responses of the time delay where Ts=50 μs.

3.3.2 Sample-and-Hold effect of the PWM

As depicted in Figure 3.3, the sensorless speed controller is updated on the negative vertex of the PWM carrier. Regular digital PWM modulation introduces an inherent time delay which reproduces the Sample & Hold (S&H) mechanism. Therefore, the corresponding Laplace transfer function is

$$G_{S\&H}(s) = \frac{1 - e^{-s.T_s}}{s.T_s} \tag{3.3}$$

With $s = j.\omega$, equation (3.3) yields to

$$G_{S\&H}(j.\omega) = \frac{1-e^{-j.\omega.Ts}}{j.\omega.T_s} = e^{-j.\omega.Ts/2}.\frac{e^{j.\omega.Ts/2}-e^{-j.\omega.Ts/2}}{2.j}.\frac{1}{\omega.T_s/2} \quad (3.4)$$

$$= e^{-j.\omega.Ts/2}.\frac{\sin(\omega.T_s/2)}{\omega.T_s/2} = e^{-j.\omega.Ts/2}.sinc(\omega.T_s/2)$$

From equation(3.4), it is clear that S&H element introduces an inherent time delay equal to $T_s/2$ into the control loop. It introduces also a gain equals to the magnitude of $\sin c(\omega.T_s/2)$.

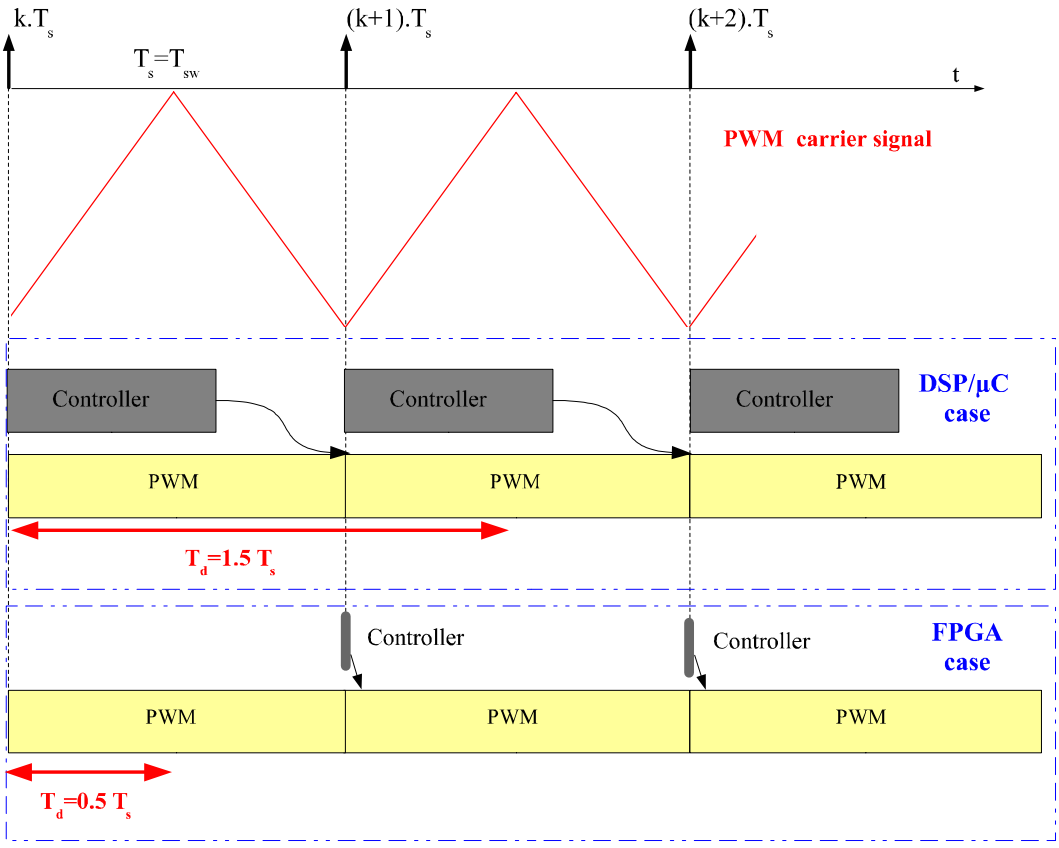


Figure 3.3: Timing diagram of the control implementation based on PWM strategy.

The bode plots of the S&H element due to the PWM, for F_s=100 kHz, are presented Figure 3.4. It shows that the magnitude of the S&H element can be approximated by a first order low-pass filter magnitude. This is not the case of the phase since the S&H element introduces a larger phase lag.

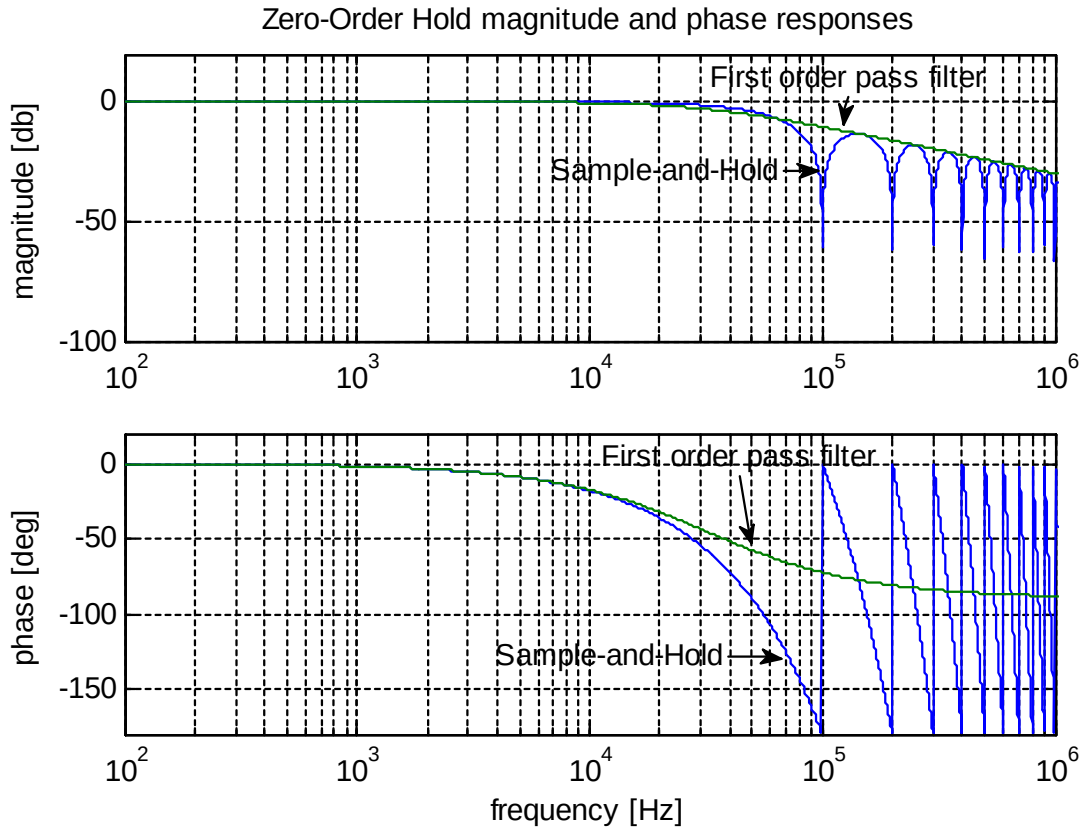


Figure 3.4: Magnitude and phase diagrams of a Sample and Hold element for Fs=100 kHz.

Thus, the global time delay transfer function of the controller is equal to

$$G_{Total}(j.\omega) = e^{-j.\omega.T_{A\lg}}.e^{-j.\omega.Ts/2}.\frac{\sin(\omega.T_s/2)}{\omega.T_s/2} = e^{-j.\omega.T_d}.\frac{\sin(\omega.T_s/2)}{\omega.T_s/2} \tag{3.5}$$

Where T_d represents the summation of the computation time delay and the time delay due to the PWM. As indicated before, this time delay depends on the digital platform and can be expressed as follow

$$T_d = \begin{cases} \frac{T_s}{2} + T_{A\lg} = \frac{3.T_s}{2} & For\ DSP/\mu C\ implementation \\ \frac{T_s}{2} + T_{A\lg} = \frac{T_s}{2} & For\ FPGA\ implementation \end{cases} \tag{3.6}$$

3.4 Stator current controller

3.4.1 Current controller synthesis

In the d-q reference frame, the voltage and flux of synchronous machine can be expressed by equations (3.7) to (3.10).

$$V_{sd} = R_s.i_{sd} + \frac{d\Phi_{sd}}{dt} - \omega_{dq}.\Phi_{sq} \tag{3.7}$$

$$V_{sq} = R_s.i_{sq} + \frac{d\Phi_{sq}}{dt} + \omega_{dq}.\Phi_{sd} \tag{3.8}$$

$$\Phi_{sd} = L_{sd}.i_{sd} + M_{sr}.i_{rd} \tag{3.9}$$

$$\Phi_{sq} = L_{sq}.i_{sq} \tag{3.10}$$

In Laplace domain, the previous relations yield to the equations (3.11) and (3.12).We suppose that the rotor current i_{rd} is set at its rated value (1 A). Thus, the d-q stator currents can be expressed

$$i_{sd} = \frac{1}{R_s + L_{sd}.s}(V_{sd} + \omega_{dq}.\Phi_{sq}) \tag{3.11}$$

$$i_{sq} = \frac{1}{R_s + L_{sq}.s}(V_{sq} - \omega_{dq}.\Phi_{sd}) \tag{3.12}$$

The regulation of the i_{sd} and i_{sq} components is ensured by PI regulators, given respectively by the equations (3.13) and (3.14).

$$G_{PI_{id}}(s) = K_{pid} + \frac{K_{iid}}{s} \tag{3.13}$$

$$G_{PI_{iq}}(s) = K_{piq} + \frac{K_{iiq}}{s} \tag{3.14}$$

Figure 3.5 (a) presents the synoptic of the current controller based on the PI regulators. We remind that the principle of the PWM is to deliver at each switching period an average voltage vector to the machine that is equal to its reference value. The S&H element and the computation time delay are taken into account. This leads to the simplified closed loops depicted in Figures 3.5(b) and 3.5(c). Thus, the open loop transfer functions of i_{sd} and i_{sq} are

$$G_{i_{sd}}(s) = \frac{1}{s}.\frac{K_{pid}.(s + K_{iid}/K_{pid})}{L_{sd}(s + R_s/L_{sd})}.e^{-s.T_{A\lg}}.\frac{1 - e^{-s.T_s}}{s.T_s} \tag{3.15}$$

$$G_{i_{sq}}(s) = \frac{1}{s}.\frac{K_{piq}.(s + K_{iiq}/K_{piq})}{L_{sq}(s + R_s/L_{sq})}.e^{-s.T_{A\lg}}.\frac{1 - e^{-s.T_s}}{s.T_s} \tag{3.16}$$

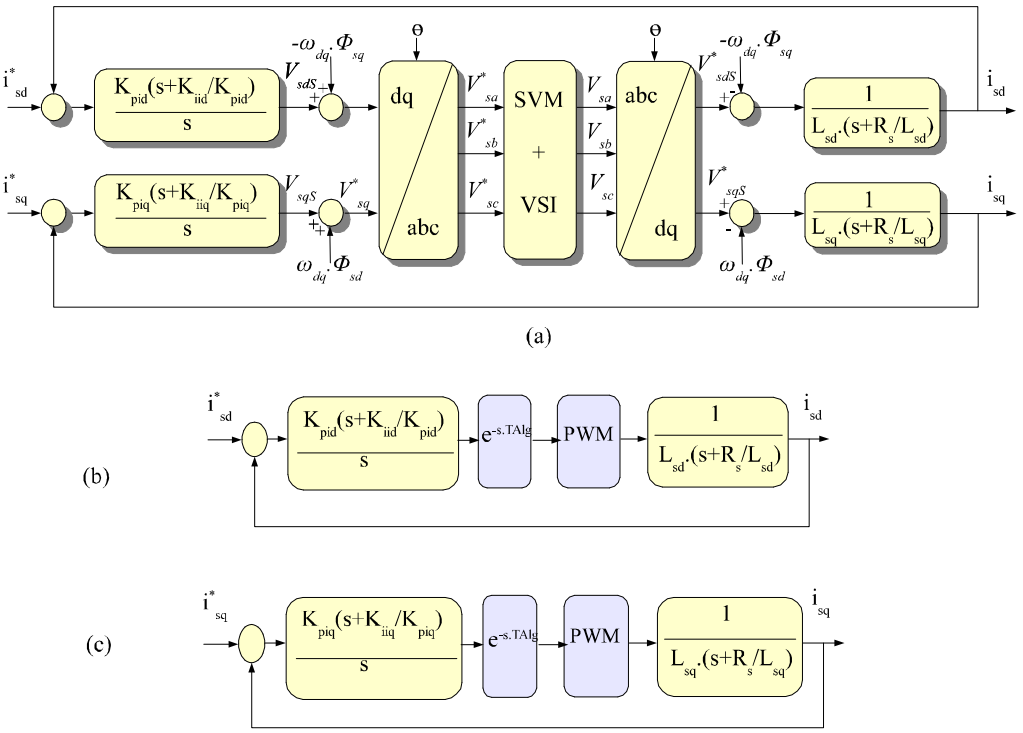


Figure 3.5: (a) closed loop of the stator current controller, (b),(c) simplified closed loops of the stator current controller with time delay.

The PI regulators were synthesized based on two design criteria: the phase margin "Φ_m" and the desired closed loop bandwidth "f_c". The pole cancellation method was also considered. It consists on imposing K_{iid}/K_{pid} to be equal to R_s/L_{sd} and K_{iiq}/K_{piq} to be equal to R_s/L_{sq}. Considering all these criteria, the magnitude and the phase relations are

$$d-axis \begin{cases} |G_{i_{sd}}(j.\omega_c)| = \left| \frac{1}{j.\omega_c}.\frac{K_{pid}}{L_{sd}}.e^{-j.\omega_c.T_d}.\frac{\sin(\omega_c.T_s/2)}{\omega_c.T_s/2} \right| = 1 \\ Arg(\frac{1}{j.\omega_c}.\frac{K_{pid}}{L_{sd}}.e^{-j.\omega_c.T_d}.\frac{\sin(\omega_c.T_s/2)}{\omega_c.T_s/2}) = -\omega_c.T_d - \frac{\pi}{2} >= -\pi + \Phi_m \end{cases} \tag{3.17}$$

$$q-axis \begin{cases} \left|G_{i_{sq}}(j.\omega_c)\right| = \left| \frac{1}{j.\omega_c}.\frac{K_{piq}}{L_{sq}}.e^{-j.\omega_c.T_d}.\frac{\sin(\omega_c.T_s/2)}{\omega_c.T_s/2} \right| = 1 \\ Arg(\frac{1}{j.\omega_c}.\frac{K_{piq}}{L_{sq}}.e^{-j.\omega_c.T_d}.\frac{\sin(\omega_c.T_s/2)}{\omega_c.T_s/2}) = -\omega_c.T_d - \frac{\pi}{2} >= -\pi + \Phi_m \end{cases} \tag{3.18}$$

where $\omega_c = 2.\pi.f_c$ is the crossover pulsation.

For practical control system, the phase condition is usually more restrictive than the magnitude condition. Thus and based on the phase relation (given by equations (3.17) and (3.18)), the maximum bandwidth can be deduced. For a specified phase margin

"Φ_m" , a fixed sampling period and a given digital platform (FPGA or DSP/μC), the maximum allowable bandwidth is

$$\omega_{c\max} = 2.\pi.f_{c\max} \leq \begin{cases} \frac{2}{3.T_s}.\left[\frac{\pi}{2} - \Phi_m\right] & For\ DSP/\mu C\ implementation \\ \frac{2}{T_s}.\left[\frac{\pi}{2} - \Phi_m\right] & For\ FPGA\ implementation \end{cases} \tag{3.19}$$

In our case, the phase margin "Φ_m" was set to 60°. Thus, the maximum bandwidth "$f_{c\max}$" can be expressed as follows

$$f_{c\max} \leq \begin{cases} \frac{f_s}{18} & For\ DSP/\mu C\ implementation \\ \frac{f_s}{6} & For\ FPGA\ implementation \end{cases} \tag{3.20}$$

Based on equation (3.20), the maximum achievable bandwidths for the two considered systems (S_h, S_l) are presented in Table 3.3.

	Maximum achievable bandwidth		**rule of thumb**
	DSP/μC	FPGA	**limits**
$\mathbf{S}_h$(100 kHz)	5.55 kHz	16.66 kHz	5 kHz
$\mathbf{S}_l$(20 kHz)	1.11 kHz	3.33 kHz	1 kHz

Table 3.3: Bandwidth limits

Once the maximum bandwidth is determined, the PI regulator gains (K_{pid}, K_{piq}, K_{iid} *and* K_{iiq}) can be computed based on the magnitude relations of the open loop transfer function, given by equations (3.17) and (3.18).

However, in practical control applications, the crossover frequency is generally fixed to one-twentieth of the sampling period (equal to the switching period in our case). This is a general "rule of thumb" in digital control. The principle is that the sampling rate, "f_s" should be higher that the closed loop bandwidth "f_c" as defined in expression (3.21) :

$$f_C \leq \frac{f_s}{20} \tag{3.21}$$

In Table 3.3, we present the maximum bandwidths for the two studied systems (S_l , S_h) according to this rule of thumb. In the following, these bandwidth values are used to synthesize the d-q current PI regulators.

The time delay effect was also analyzed in the time domain. For the two considered AC drive systems, the step responses of the closed loop are given in Figures 3.6 and 3.7.

Comparing the system without any delay and the actual system (with time delay), we note that the presence of the time delay implies an underdamped system response. Thus, a longer settling time and higher overshoot are noted.

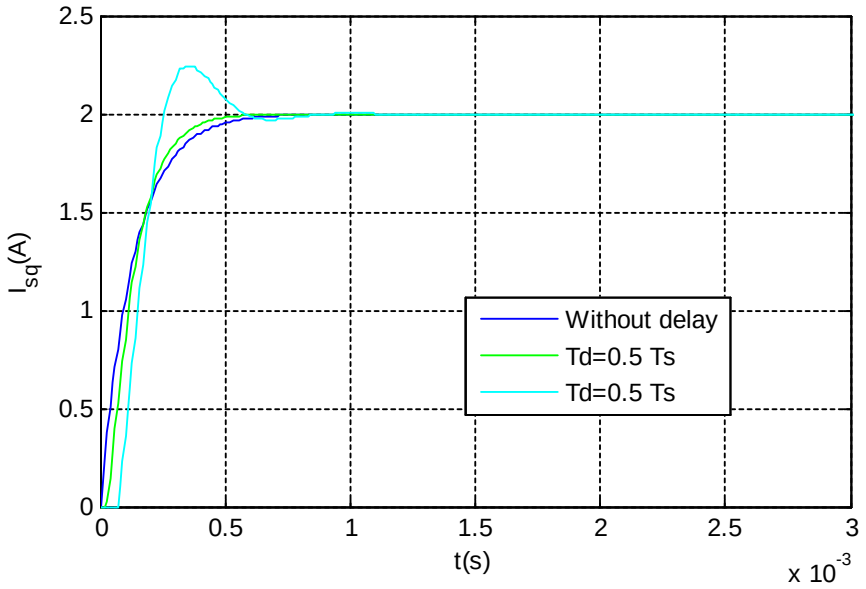


Figure 3.6: Isq step response of the low rated speed system and Ts=50μs.

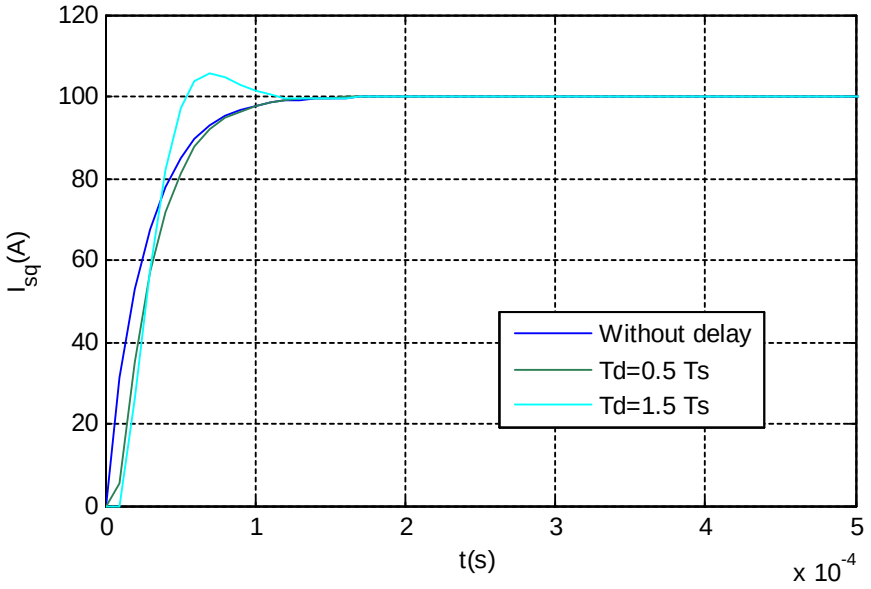


Figure 3.7: Isq step response of the high rated speed system and Ts=10μs.

The simulations in the frequency domain were also performed for the two considered systems (S_l, S_h). Figure 3.8 and Figure 3.9 present the magnitude and the phase frequency responses of the closed loop transfer functions. We can note that the phase lag, due to the time delay and S&H element, increases linearly with the frequency impacting the stability of the controlled system. On the other hand, the closed loop transfer function presents a maximum closed loop log modulus which means a poor damping factor. This is confirmed by the step responses in time domain (see Figures 3.6 and 3.7).

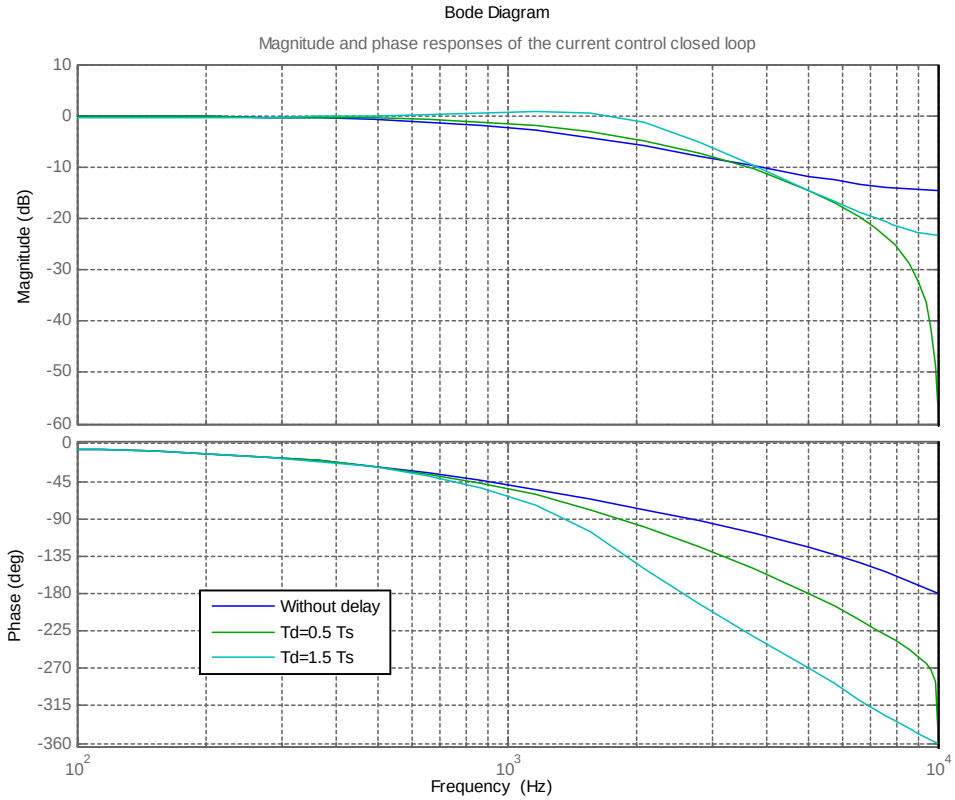


Figure 3.8: The Bode plot of the current control closed loop for different time delays (case of the low rated speed system).

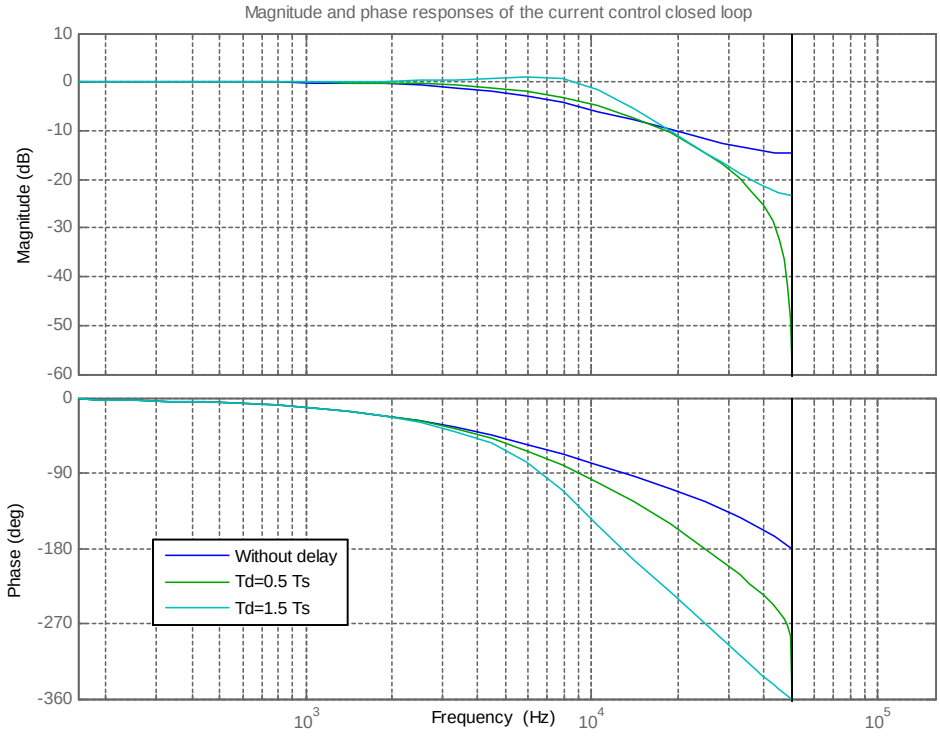


Figure 3.9: The Bode plot of the current control closed loop for different time delays (case of the high rated speed system).

On the other hand, having fixed the desired bandwidth "f_c" (fixed by the rule of thumb) and the phase margin "Φ_m" (set to 60°), the maximum allowable time delay can be easily derived

$$T_d \leq \frac{\pi/2 - \Phi_m}{2.\pi.f_c} \tag{3.22}$$

Figure 3.10 shows the set of time delay values respecting the stability condition given by the equation (3.22) and for a phase margin "Φ_m" fixed to 60°.

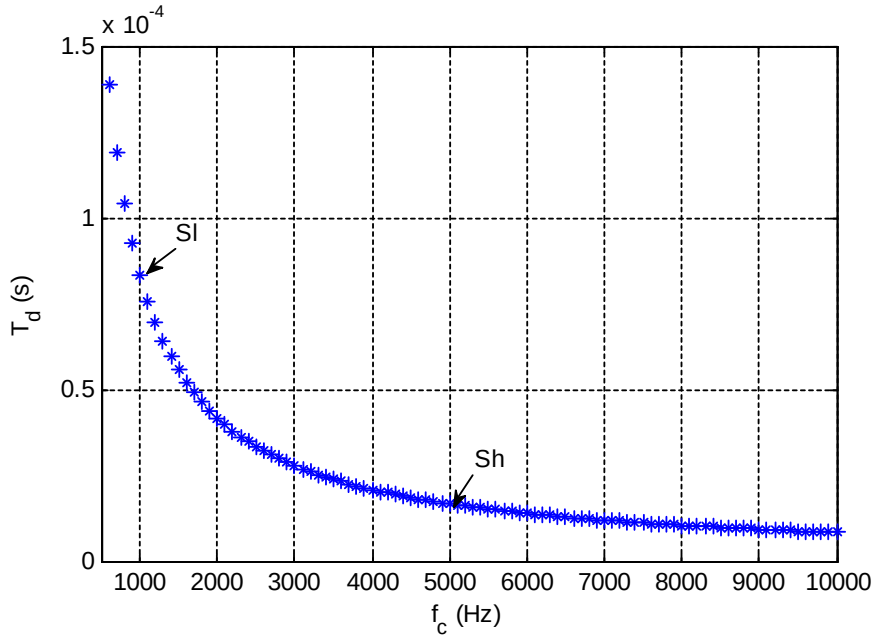


Figure 3.10 : The maximum time delay values respecting the stability condition (for Φ_m =60°).

Table 3.4 presents the time delay limits with regard to the chosen stability condition. The exceeding of these limits decreases the fixed phase margin and in some cases can cause system instability. Thus, the maximum execution time devoted to the control algorithm "$T_{A\lg}$" can be derived as follows

$$\begin{aligned} For \quad S_l : T_{A\lg} &\leq T_d - \frac{T_s}{2} = 58.33\,\mu s \\ For \quad S_h : T_{A\lg} &\leq T_d - \frac{T_s}{2} = 11.66\,\mu s \end{aligned} \tag{3.23}$$

These maximum allowable execution times will be used as functional constraints during the HW-SW Co-design procedure (see chapter 5).

System	Bandwidth "f_c"	$\mathbf{T_s}$	**Stability Condition**
S_l, F_{sw}=20 kHz	1000 Hz	50 μs	$T_d \leq 83.33$ μs
S_h, F_{sw}=100 kHz	5000 Hz	10 μs	$T_d \leq 16.66$ μs

Table 3.4: The time delay limits regarding the stability condition

3.4.2 Speed controller

The speed control loop is performed using P-PI regulator. It consists on two closed loops (inner and outer loop). The first one uses a proportional regulator (P) which aims to impose the system poles at the desired location. The second one is based on PI regulator to provide the defined steady and transient speed responses. The principle of this speed control strategy is shown by Figure 3.11.

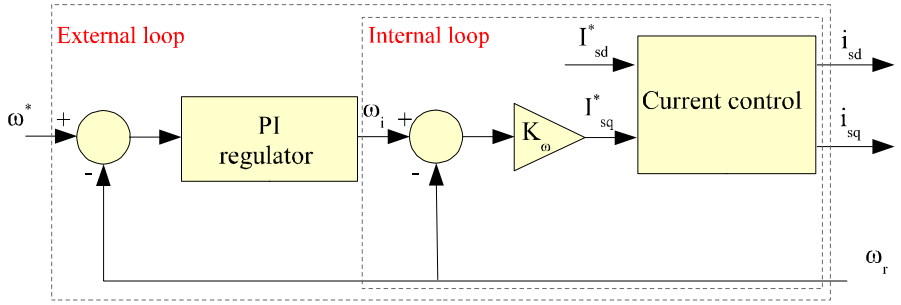


Figure 3.11: The speed control block diagram.

Assuming that the dynamic of the current controller is much faster than the one of speed controller, the expression in the Laplace domain between the electrical speed ω_r and the stator current i_{sd} and i_{sq} can be expressed as follow:

$$\omega_r(s) = \frac{3}{2}.p^2.\frac{1}{f_l + J.s}.\left(\left[L_{sd} - L_{sq}\right].i_{sq}(s).i_{sd}(s) + M_{sr}.i_{sq}(s).I_{rd}(s) - \frac{2}{3.p^2}.\Gamma_L(s) \right) \quad (3.24)$$

In this study, the direct current reference i_{sd} is set to zero. In addition, the effect of load torque "Γ_L" and the friction coefficient "f_l" are neglected. Consequently, the transfer function between ω_r and i_{sq} can be defined by

$$\frac{\omega_r(s)}{i_{sq}(s)} = \frac{3}{2}.p^2.\frac{M_{sr}.I_{rd}(s)}{J.s} \quad (3.25)$$

The regulation of the internal loop is provided using a proportional gain. The main function of this gain is to impose the controlled system poles in the desired locations. Based on the transfer function of the i_{sq} current closed loop (equation (3.26), the new transfer function is expressed by equation (3.27)

$$H_{i_{sq}}(s) = \frac{1}{1 + s.\dfrac{K_{piq}}{L_{sq}}} \quad (3.26)$$

$$\frac{\omega_r(s)}{\omega_i(s)} = \frac{\dfrac{1.5.K_\omega.p^2.M_{sr}.I_{rd}}{J.\dfrac{L_{sq}}{K_{piq}}}}{s^2 + s.\dfrac{1}{\dfrac{L_{sq}}{K_{piq}}} + \dfrac{1.5.K_\omega.p^2.M_{sr}.I_{rd}}{J.\dfrac{L_{sq}}{K_{piq}}}} \quad (3.27)$$

Assuming that I_{rd} is fixed to its rated value, a new coefficient K_ω is defined,

$$K_\omega = \frac{J}{6.\frac{L_{sq}}{K_{piq}}.p^2.M_{sr}.I_{rd}} \tag{3.28}$$

This yields to the final transfer function given by equation (3.29). It presents a second order system with double-real poles

$$\frac{\omega_r(s)}{\omega_i(s)} = \frac{\frac{1}{4.(\frac{L_{sq}}{K_{piq}})^2}}{(s + \frac{1}{2.\frac{L_{sq}}{K_{piq}}})^2} \tag{3.29}$$

From external control loop side, the PI regulator was used to provide a no steady state speed error and to impose a given dynamic. Knowing that the transfer function of the PI regulator is

$$G_\omega(s) = K_{p\omega} + \frac{K_{i\omega}}{s} \tag{3.30}$$

And imposing the ratio $\frac{K_{i\omega}}{K_{p\omega}}$ equal to $\frac{K_{piq}}{L_{sq}}$, the transfer function of the external speed loop is

$$\frac{\omega_r(s)}{\omega^*(s)} = \frac{\frac{K_{p\omega}}{4.(\frac{L_{sq}}{K_{piq}})^2}}{s^2 + \frac{1}{2.\frac{L_{sq}}{K_{piq}}}.s + \frac{K_{p\omega}}{4.(\frac{L_{sq}}{K_{piq}})^2}} \tag{3.31}$$

The new transfer function corresponds to a second order system. The computation of $K_{p\omega}$ is based on the desired overshoot and settling time.

3.4.3 Rotor speed and position observer using EKF algorithm

Extended Kalman Filter (EKF) is known for its efficiency of the online estimation of speed and rotor position especially at medium and high motor speed range. A first validation of the EKF algorithm was achieved in an associate work [8], [113]. These results encourage the author to adapt the EKF algorithm for the considered AC drive systems. Additionally, this algorithm presents an interesting benchmark for the HW-SW partitioning that will be detailed in the next chapter.

The EKF algorithm allows estimating the speed and position of the rotor. As depicted Figure 3.16, this algorithm requires the current and the voltage quantities in order to produce the states estimates. These quantities are provided by the voltage/current

interfaces and the abc-dq transformations. The interface voltage aims to the generation of the 3-phase stator voltages from the measured DC voltage and the 3-phase stator reference voltages according to the following relation (see 2.3.1).

$$V_{si} = E.V_{si}^{*} \qquad ; i = a, b, c \tag{3.32}$$

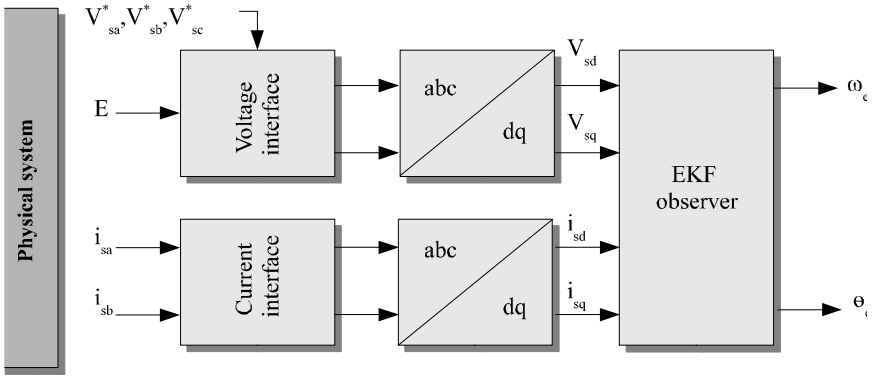


Figure 3.16: The block diagram of the EKF algorithm

The EKF observer is based on system state model. It is optimal for stochastic environment when noises can be considered as white Gaussian noises [66]-[68],[113],[8]. To start with, the discrete-time model obtained using the first order forward Euler approximation method is given equation (3.33). T_s represents the sampling period and k the sampling index.

$$\begin{aligned} x_k &= x_{k-1} + T_s.f\left(x_{k-1}, u_{k-1}\right) + w_k \\ y_k &= H.x_k + v_k \end{aligned} \tag{3.33}$$

Where x is the state space vector, u *and* y are respectively the system input and output vectors. w and v are respectively the model and the measurement disturbances.

$$x = \begin{bmatrix} i_{sd} & i_{sq} & \theta_e & \omega_e \end{bmatrix}^T \; ; \quad u = \begin{bmatrix} V_{sd} & V_{sq} \end{bmatrix}^T ; \; y = \begin{bmatrix} i_{sd} & i_{sq} \end{bmatrix}^T \tag{3.34}$$

$$f(x,u) = \begin{bmatrix} -\dfrac{R_s}{L_{sd}}.i_{sd} + \dfrac{L_{sq}}{L_{sd}}.\omega_e.i_{sq} \\ -\dfrac{R_s}{L_{sq}}.i_{sq} - \dfrac{L_{sd}}{L_{sq}}.i_{sd}.\omega_e + \dfrac{M_{sr}}{L_{sq}}.I_{rd}.\omega_e \\ 0 \\ \omega_e \end{bmatrix} + \begin{bmatrix} \dfrac{1}{L_{sd}} & 0 \\ 0 & \dfrac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} .u \; ; H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \tag{3.35}$$

It consists on the state space model using rotating d-q reference frame. The adopted model is derived under the so-called "infinite inertia" hypothesis which means that the

speed dynamic is neglected regarding the current dynamics. Namely, the state space model has been fixed to 4.

The principle of EKF treatment is described in Figure 3.17. Firstly, the prediction of the state vector $\widehat{x}_{k/k-1}$ of the system is performed. Next, the innovation step is executed. It aims to compensate the predicted vector $\widehat{x}_{k/k-1}$ using the Kalman gain K and the measurement vector y_k yielding to the estimated state vector $\widehat{x}_{k/k}$.

In the following, we detail the equations of each steps of EKF algorithm.

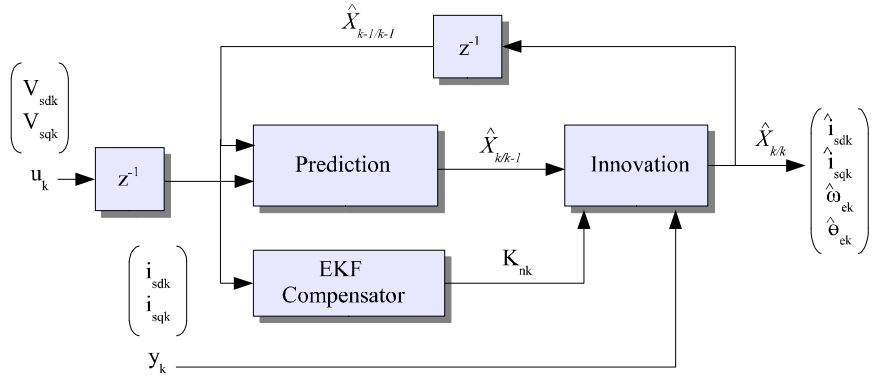


Figure 3.17: Synoptic of the EKF algorithm

-Prediction step: The prediction step can be represented by the following equation

$$\widehat{x}_k = \widehat{x}_{k-1} + T_s.f\left(\widehat{x}_{k-1}, u_{k-1}\right) \tag{3.36}$$

-EKF Compensator - Kalman gain calculation

As for the EKF compensator, it is computed using equations (3.37) to (3.40).

-The covariance matrix of the state error is expressed by

$$P_{k/k-1} = Fd_k.P_{k-1/k-1}.Fd_k^t + Q \qquad Initial\ value\ P_0 \tag{3.37}$$

where Fd_k is defined from the Jacobian matrix :

$$Fd_k = \left.\frac{\partial\left(\widehat{x}_{k-1} + T_s.f\left(\widehat{x}_{k-1}, u_{k-1}\right)\right)}{\partial x}\right|_{x=\widehat{x}_{k-1/k-1}} \tag{3.38}$$

The Kalman matrix is computed by :

$$K_k = P_{k-1/k-1}.H^t.\left(H.P_{k-1/k-1}.H^t + R\right)^{-1} \tag{3.39}$$

Where P_0, Q and R are respectively the initial state error, the model noise and the measurement noise covariance matrices. The tuning of these matrices has been done according to the methodology proposed in [66],[113]. It consists of a trial-and-error method presenting guidelines to set the EKF estimation behavior during the transient and steady state. We note that it is a common practice to assume these covariance matrices to be diagonal and invariant.

-The Updating of the covariance matrix of the error is expressed by:

$$P_{k/k} = P_{k/k-1} - K_k.H.P_{k/k-1} \tag{3.40}$$

-Innovation step: Finally, the innovation step is carried out using the measurement update as described in equation (3.41) that ensures the update of the estimated state vector.

$$\widehat{x}_{k/k} = \widehat{x}_{k/k-1} + K_k.\left(y_k - H.\widehat{x}_{k/k-1}\right) \tag{3.41}$$

After some simulation tests, the covariance matrices were fixed as provided in(3.42). This setting allows having satisfied results in terms of steady and transient state performances.

$$P_o = \begin{bmatrix} 10^{-1} & 0 & 0 & 0 \\ 0 & 10^{-1} & 0 & 0 \\ 0 & 0 & 10^{-2} & 0 \\ 0 & 0 & 0 & 10^{-3} \end{bmatrix}; Q = \begin{bmatrix} 10^{-3} & 0 & 0 & 0 \\ 0 & 10^{-3} & 0 & 0 \\ 0 & 0 & 10^{-4} & 0 \\ 0 & 0 & 0 & 10^{-4} \end{bmatrix}; \tag{3.42}$$

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

As shown before, the EKF algorithm requires matrix computations such as multiplication, subtraction, addition, and inversion... All these treatments need a high amount of arithmetic operations. Table 3.5 gives an idea about the complexity of the EKF algorithm.

	x	**+**	**-**	**inversion**
Prediction	10	6	0	0
Jacobian matrix	4	1	0	0
Kalman gain & covariance matrix	318	244	16	1
Innovation	8	8	8	0
External dq-abc transformation	12	12	0	0
Total	352	271	24	1

Table 3.5: The EKF complexity

The computation of the Kalman gain "K" is presented in Figure 3.18. It presents the module of highest complexity degree since it requires up to 318 multiplications, 244 additions, 16 subtractions and 1 inversion. This module was split into four sub-modules (compensator 1, 2, 3 and 4) which provide more manageable matrix treatment.

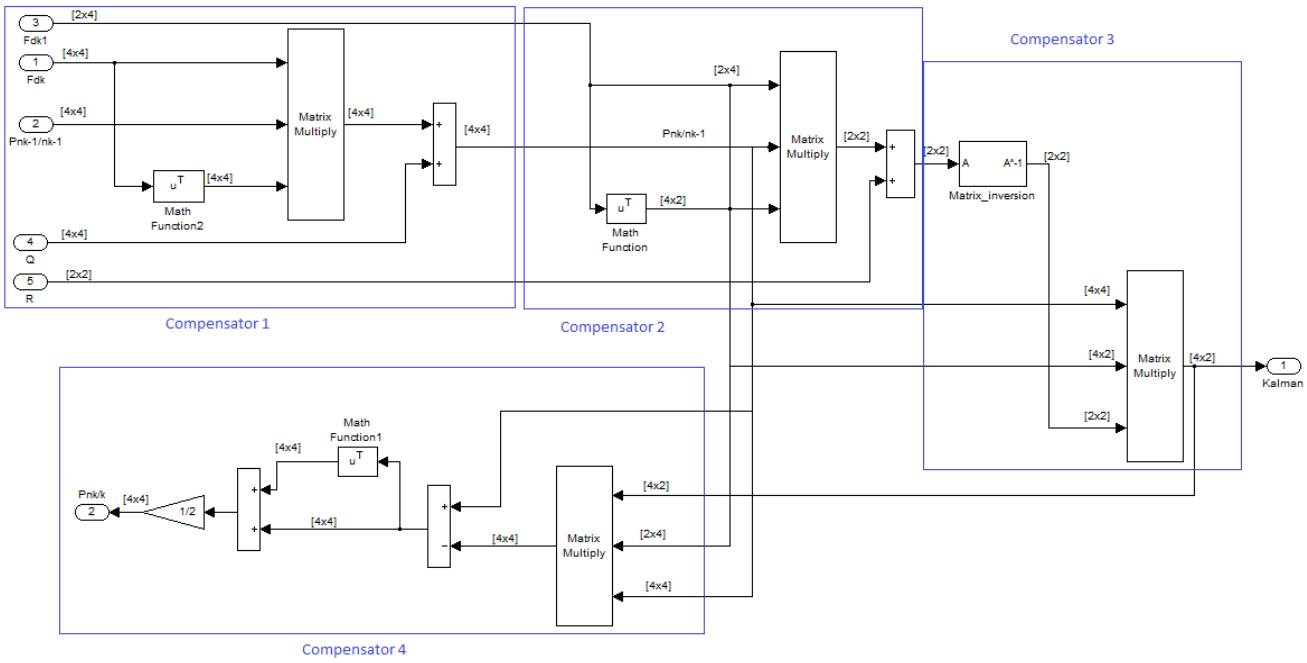


Figure 3.18: The sub-modules of the Kalman gain.

3.4.4 Discretization and fixed-point data setting

To prepare the digital implementation, some further development steps are mandatory. The first step is the discretization of the control algorithm. The approach based on the first order Forward Euler approximation was adopted. The normalization is the next step. It consists in the development of per-unit algorithm. Hence, the variables of the sensorless speed controller are normalized according to their base-values (see Appendix-D).

Once this step is achieved, designer can begin the fixed-point data setting. This latter must be chosen with respect to precision and dynamic requirements. This means that the fixed data format and the real one (floating-point data) must be closer as possible. As depicted by Figure 3.19, the fixed point representation is divided into two main parts: the integer part and the fractional part. The first part is related mainly to the dynamic range of variables and coefficients while the second one is related to the precision. This representation is denoted $(i+f)Q_f$. The i is the number of bit of the integer part, f is the number of fractional part and the $(i+f)$ is the total data size.

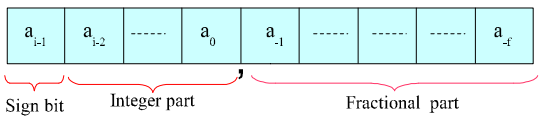


Figure 3.19: Fixed-point representation of the data

The chosen format has been set based on fixed-point simulation tests. These simulations were repeated until finding a low and acceptable error between the fixed-point data and floating-point data. Table 3.6 presents the chosen formats used for the digital implementation of the whole sensorless speed controller [113]. The corresponding simulation results are presented Figure 3.20.

Modules and fixed-point format			
Current interface	$13Q_{12}$	PI regulator	$20Q_{18}$
Voltage interface	$13Q_{12}$	P-PI regulator	$20Q_{18}$
$abc-dq$ transformation	$13Q_{12}$	EKF observer	$22Q_{20}$
$dq-abc$ transformation	$13Q_{12}$	SVM	$13Q_{12}$

Table 3.6: Bit width of control modules

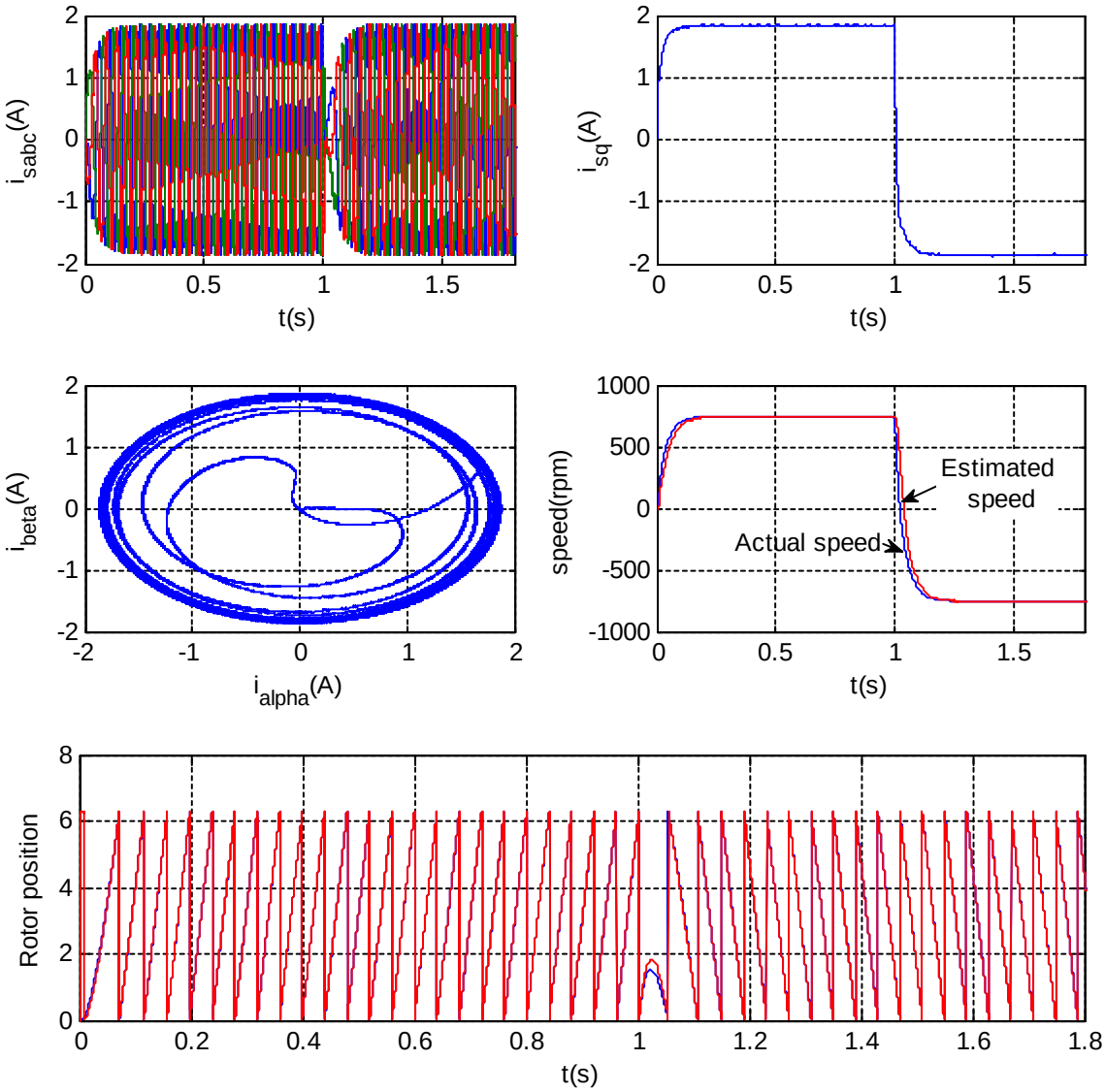


Figure 3.20: Simulations results of the fixed-point sensorless control algorithm.

These results present the case of the low rated speed system. It proves the good functionality of the developed fixed-point sensorless speed controller.

3.5 Conclusion

Digital controllers are always characterized by an inherent time delay. In this chapter, author has focused on the analysis of the time delay impact on the control performances. This time delay has been identified as the sum of the execution time needed for the control algorithm computation and the PWM. This latter is caused by the PWM process.

The adopted control strategy consists on the sensorless speed controller based on the Extended Kalman Filter (EKF). Two AC drive systems (high and low rated speed) and two switching frequencies (20 kHz and 100 kHz) were considered. The analysis of time delay impact has been performed in time and frequency domains. It proved that both the bandwidth and the stability margin are affected.

For a fixed bandwidth, a fixed phase margin and a fixed sampling period, the maximum allowable time delays have been determined for the two considered AC drive systems. These time delays will be considered as a constraint to the HW-SW partitioning procedure that will be presented in the next chapter.

Chapter 4

Co-design methodology: HW-SW partitioning

Chapter 4

Co-design methodology: HW-SW partitioning

4.1 Introduction

In nowadays embedded control systems, the digital controllers are getting increasingly sophisticated. The industrial demands in terms of control performances, reliability, integration and cost are all the more important that the implemented algorithms become complex [69]-[72]. On the other side, this considerable progress has not only been limited to the control theory, but relies also on the more and more mature digital electronic technologies.

Staying with this last point, it is commonly accepted that the FPGA System on Chip (SoC) devices are highly appropriate to reach an optimal solution, between the required performances and the controller complexity. The combination of software treatment (allowed by the on-chip processor cores) and the hardware treatment (ensured by the FPGA hardware resources) allows the designer to enhance the performances of the design and improves its flexibility.

In contrast, all these assets are hardly achievable without a rigorous and efficient Co-design methodology. Indeed, designer has to make an efficient partitioning between the software and the hardware parts. Thus, this partitioning must lean on quantitative metrics that define the objectives to achieve with regards to the constraints. To this purpose, author has developed a specific HW/SW Co-design methodology adapted to AC drive applications. For a given control algorithm, three main concepts have been defined: the modular partitioning, the performances estimation and the HW/SW partitioning. The first one aims to decompose the design into functional modules with different levels of granularity. The performance estimation aims to characterize each module in terms of computational time and FPGA resource use. Finally, the HW/SW partitioning is based on these estimated performances and on the design constraints so as to be implemented in hardware. This optimization is made with the help of the Non-Dominated sorting Genetic Algorithm (NSGA-II). To illustrate this Co-design methodology, the chosen benchmark is sensorless controller of a synchronous machine using an Extended Kalman Filter (EKF). This choice corresponds to a major trend in embedded AC drive domain. Its direct consequence is a significant increase of the complexity of the algorithm to be implemented, complexity mainly due to the EKF matrix computation.

In this chapter, author starts by presenting a full software implementation of this sensorless controller and then its experimental validation. This first implementation

is considered as a reference to the Co-design process that follows. The performance estimation is then presented, followed by the optimized HW-SW partitioning.

4.2 First stage of experimental validation

As first stage of experimental validation, a pure SW implementation of the EKF sensorless current controller was investigated. Here, a simple current controller based on the hysteresis regulator was considered. The chosen digital platform is the Virtex-5 from Xilinx Company which include the soft processor core "Microblaze". In the following, the focus is to evaluate the computing performances of the processor core to run the high computation EKF algorithm [115]. The implemented architecture is depicted Figure 4.1.

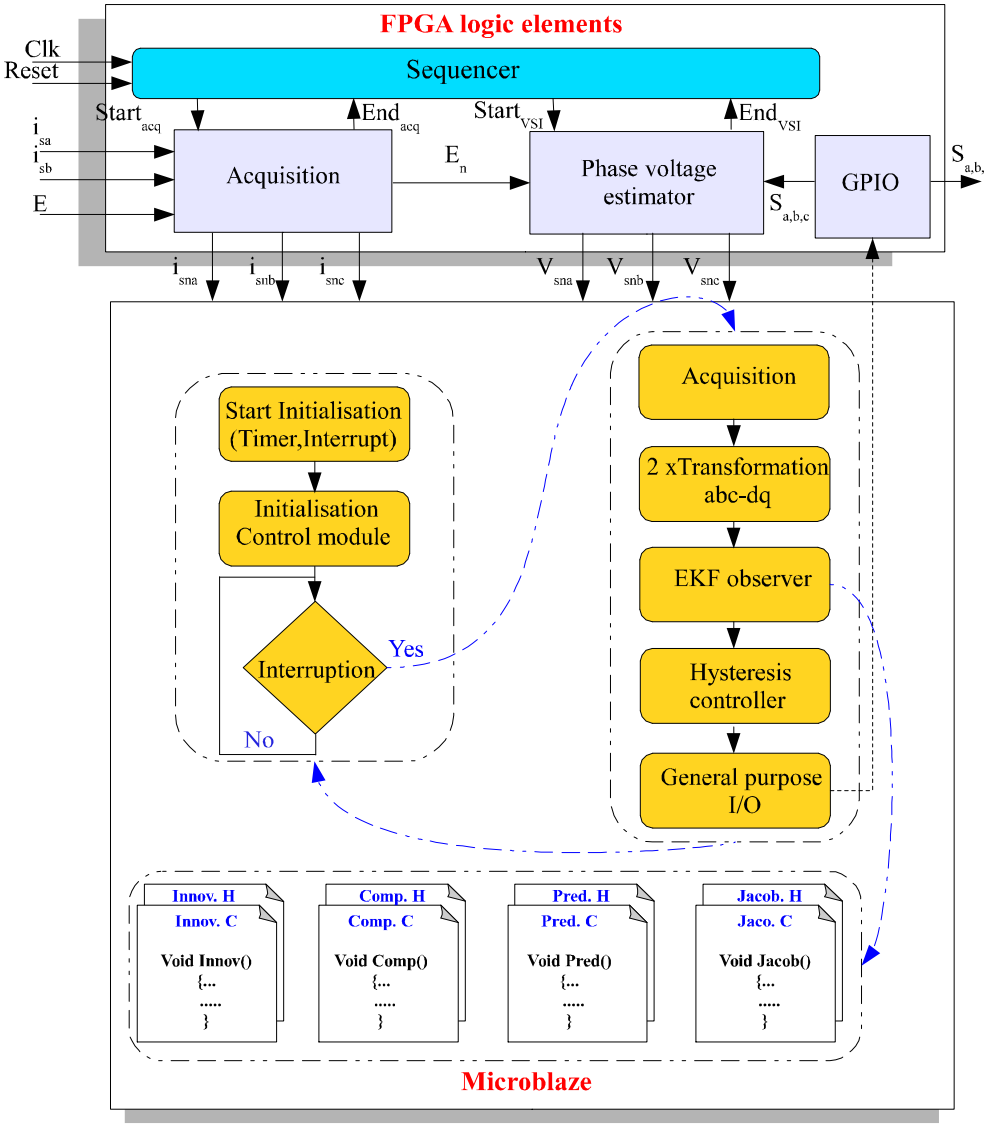


Figure 4.1: Architecture of the Microblaze-based sensorless current controller.

The structure of the testbench is divided into two main parts: the hardware design and the software design.

- **Hardware modules**

For the designed architecture, the Processor Local Bus (PLB) is used to connect peripherals. These peripherals are:

-The acquisition module of the stator current and the DC-link voltage. It is connected to the AD converters.

-The phase voltage estimator module (detailed in section 2.3.3.1). These voltages are used by the EKF sensorless treatment.

-The general Purpose Input Output (GPIO) peripheral.

-A timer, used to generate interruptions.

-The Interrupt controller peripheral, used to manage interrupts.

The sampling period is fixed to 100 μs and the frequency of the system clock is 100 MHz.

- **Software design**

The software design was carried out using assembly macros and C-coded functions. A hardware multiplier and a divider were added to the ALU in order to boost its performances. The control algorithm includes the initialization of the parameters of Microblaze, of the EKF observer and of the current controller. Once the Interrupt Service Routine (ISR) is activated, the Microblaze starts the data acquisition of the stator currents and the DC-link voltage. After that, the execution of the EKF algorithm and the current control algorithm is performed. When the execution of the control algorithm is finished, the interruption is deleted and the switching signals are ready to be send to the inverter.

The execution time of the whole control algorithm is equal to 85 μs. The code was accommodated within embedded BRAM and takes 43 KByte. This value has to be compared to the obtained execution time by a full hardware controller (Virtex-II Pro, 50Mhz) [113].

4.2.1 Digital platform

The used FPGA is based on SRAM technology from Xilinx Company. It consists in Virtex-5 (XC5VSX50T). It includes 8,160 slices, each one contains four LUTs of 6 input and four flip-flops. As showed in Figure 4.2, this FPGA integrates matrix of logic elements, configurable I/O blocks, an interconnection network, DSP48E blocks (Digital signal processing) and RAM blocks. The DSP48E block includes a 25 x 18 Hardware Multiplier (HM) and an add/subtract accumulator. It provides also a pattern detector and a pattern bar detector that can be used for convergent rounding, overflow/underflow detection for saturation arithmetic, and auto-resetting counters/accumulators [73]. Table4.1 presents the features of Virtex-5 FPGA.

Number of Slices	32640 FF 32640 $6-Bit\ LUT$
DSP48E	288
RAM blocks(Kb)	4,752
Maximum operating frequency(Mhz)	100 (without the use of DLL)

Table 4.1: Features of Virtex-5 (ML 506)

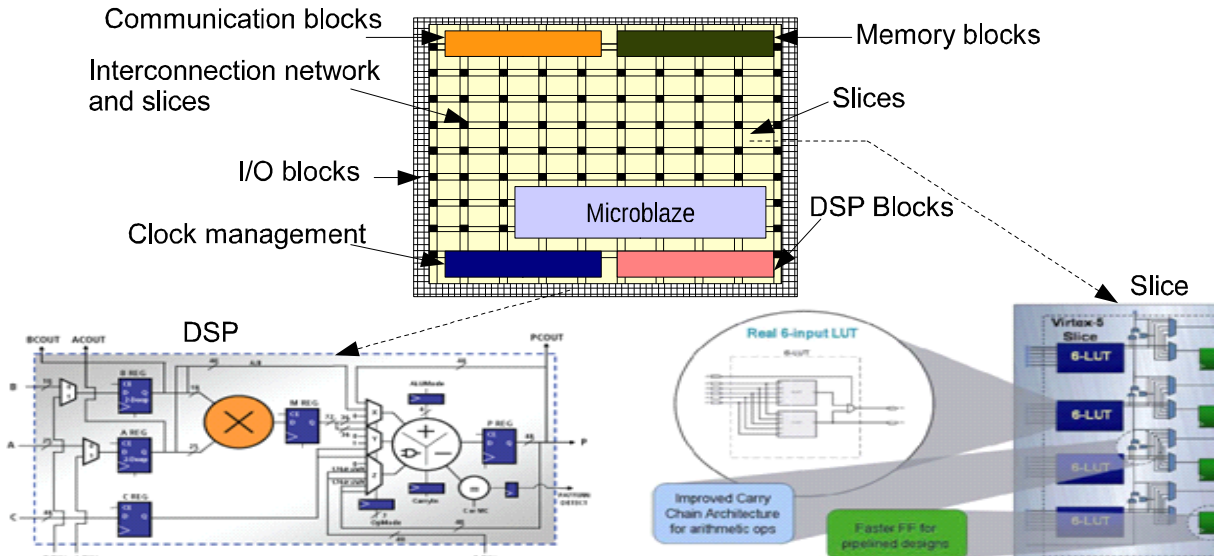


Figure 4.2: Virtex-5 Features

Additionally to the previous indicated elements, this FPGA has also the possibility to integrate a soft processor core "Microblaze". This processor has a 32-bit RISC Harvard architecture with an instruction set optimized for embedded applications. There are different processor versions: the three-stage Microblaze core which is ideal for cost-focused applications and the five stages Microblaze core which is adapted for high performance applications. Programs can be indifferently stored in external or internal SRAM blocks. Moreover, the use of precompiled arithmetic macro (multiplication-division) are not fast enough to deal with the studied real time control application. Thus, the acceleration of the Microblaze ALU was made using full wired arithmetic component (hardware multiplier, hardware division and hardware barrel shifter). These operations are executed automatically in the FPGA matrix and results can be retrieved by the processor. In the following sections, we denote "$HM_{\mu p}$" the hardware multiplier needed by Microblaze to execute multiplication.

As can be seen Figure 4.3, the Microblaze processor is organized as Harvard architecture with separate bus interface units for data and instruction.

Microblaze utilizes two Local Memory Buses for instruction (ILMB) and data (DLMB), one to store the program and the other to store data. Peripheral Local Bus (PLB) interface can be used to connect Microblaze to larger external memories. But, in this case, lower timing performances are obtained compared to the use of the LMB interface.

Microblaze has also special purpose registers such as: Program Counter (PC), Machine Status Register (MSR) to indicate the status of the processor and the enabling/disabling of interrupts. Both instruction and data interfaces of Microblaze are 32-bit wide and uses Big-Endian formats. It supports word (32 bits), half-word (16 bits), and byte accesses to data memory. Table 4.2 presents the consumed resources of a Microblaze.

	6 – Bit LUTs	FFs	$HM_{\mu p}$
Microblaze area	1911	1531	4

Table 4.2: Microblaze(v7) area occupation

The communication of Microblaze with external peripherals is provided based on two main links: Fast Simplex Link (FSL) and PLB bus. A brief description of two types of buses is given below .

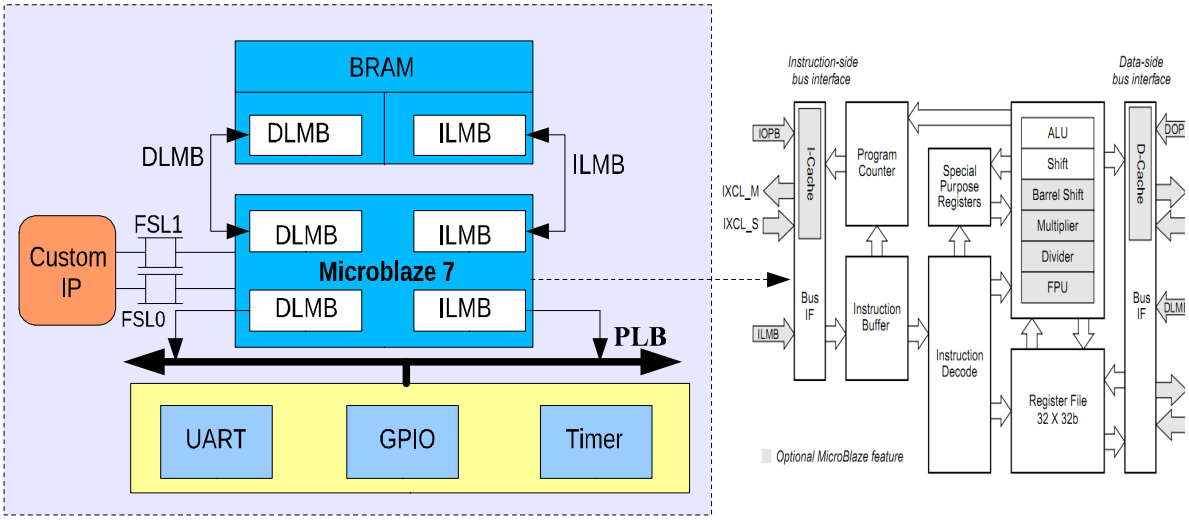


Figure 4.3: Microblaze architecture

4.2.1.1 Fast Simplex Link (FSL)

The FSL communication interface consists in an efficient and fast communication IP module provided by Xilinx. It allows a rapid communication between Microblaze and a custom IP-user(3 clock cycles in reading and writing). It is an unidirectional FIFO based communication link. For Microblaze, one can configure up to eight FSL dedicated unidirectional point-to-point data streaming interfaces. The FSL link is implemented as a 32-bit wide FIFO with configurable depth and width option. Basic software drivers are generally provided to simplify the use of FSL connection. Its interface can reach up to 300 MB/sec [73].

4.2.1.2 Processor Local Bus (PLB)

It is a communication bus which is more slowly than the FSL ones. It presents bus infrastructure for connecting PLB masters and PLB slaves within an overall PLB system. It can provide the arbitration of up 16 masters. Using Intellectual-Property Interface (IPIF) library, designer can connect safely custom IP to the PLB bus. Table 4.3 presents the resources consumed by the two kinds of communication.

	FFs	$6-Bit\ LUTs$
FSL	7	44
PLB	168	462

Table 4.3: FSL and PLB consumed resources

4.2.2 Overview of the experimental set up

The experimental validation was performed using the set up shown in Figure 4.4. This set up is composed of three main parts: the power system, the acquisition boards and the digital controller.

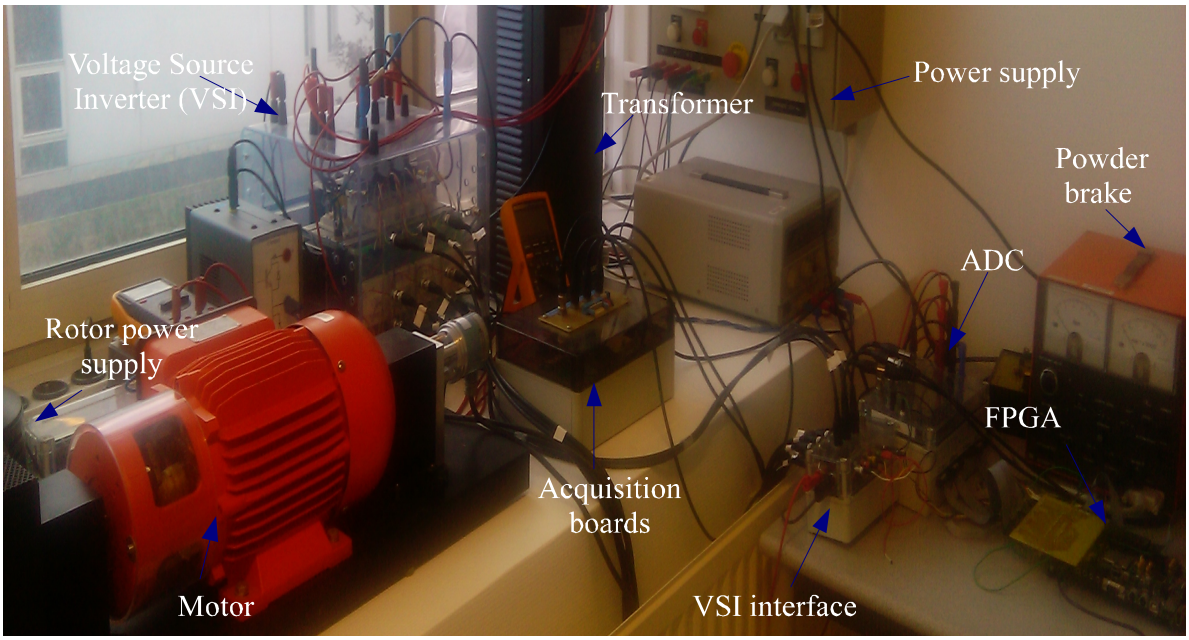


Figure 4.4: Experimental set up.

1- The power system: It is divided into three sub-parts:

- *The motor:* it consists in a Synchronous Machine (SM) with the following characteristics: 4 poles, 230/400 V, 1.52/2.66A, 0.8 kW.
- *The load:* It consists in a powder brake (Maximum torque: 25 Nm, Maximum speed: 6000 rpm).
- *The power supply:* It consists in a SEMIKRON VSI module (IGBT modules – E_{max}=800 V, I_{max}=30A).
- *The rotor power supply:* It is composed by an autotransformer (1-phase, 230 V, 50 Hz).

2- The acquisition boards: To measure the voltage and the current data, a sensor-based ARTU3I board (2.5 V/10A and 1 V/100 V) was used. The measured quantities are then sent to the ADC board which ensures the conversion from analog to digital. The chosen ADCs are AD9221, with a resolution of 12 bits and a maximum conversion rate of 1.5 Msps. The whole conversion time takes 2.4 μs.

3- The digital controller: This controller is implemented on a FPGA development board. It is based on a Virtex-5 (XC5VSX50T). The soft processor core "Microblaze" was synthesized and downloaded in the Virtex-5. The change of the current reference and the communication was ensured via a Host PC.

The experimental tests based on the EKF sensorless software controller were performed. Figures 4.5 and 4.6 present the corresponding validation results. The two measured phase stator currents and the stator voltage are presented. We note that the measured currents are well regulated and follow its reference (for I^*_{sq}= 1.5 A).The current sensors deliver voltage signals corresponding to the measured current (2.5 V/10A).

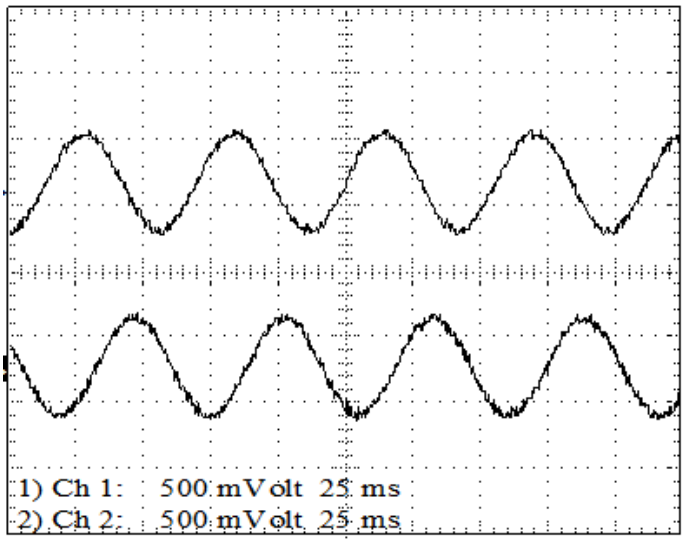


Figure 4.5: The regulated stator currents for (I^*_{sd}=0A, I^*_{sq}=1.5A, Bw*=0A) with 250mV corresponds to 1A.

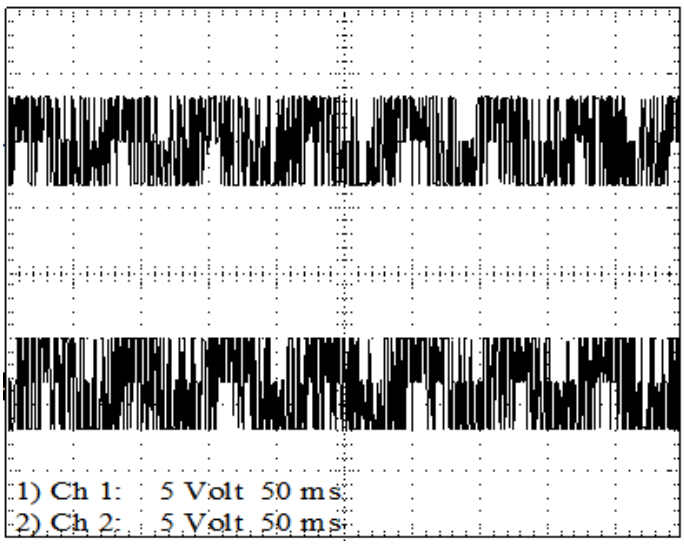


Figure 4.6: The phase voltages with 100mV corresponds to 100V.

Figure 4.7 and 4.8 present the estimated and measured rotor position. We note that the two rotor phases are identical which gives proof of the good functionality of the EKF sensorless controller.

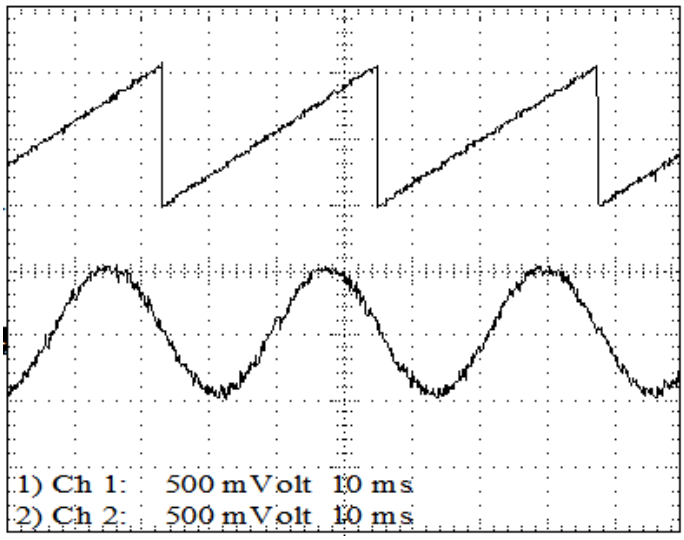


Figure 4.7: The estimated rotor position and regulated stator current with 250mV corresponds to 1A.

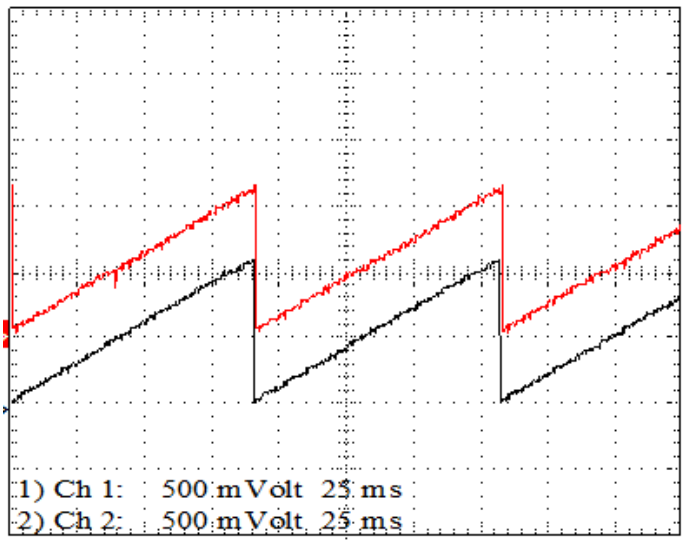


Figure 4.8: The estimated (red curve) and the measured rotor position (black curve).

These first experimental results have permitted the validation of the EKF sensorless software controller. But, the execution time is about 85μs. This important execution time is needed to perform only a simple hysteresis regulator along with the EKF observer. To implement the whole EKF sensorless speed controller based on PI regulator, a hardware accelerator seem to be necessary to boost the Microblaze treatment.

In the following, author presents a HW-SW Co-design methodology aiming to find an optimized partitioning of the control functional modules between SW and HW. The constraints linked to this optimization problem are from one hand the limited internal resources of the FPGA (Slices, DSP, memory blocks and maximum clock frequency) and on the other hand the maximum allowable execution time of the control algorithm in order to respect the specifications of the final application in terms of control performances (bandwidth, stability margin).

The proposed method is mainly composed of two steps : the estimation of occupied resources and timing performances of the functional modules included in the control algorithm and the HW-SW partitioning optimization process.

4.3 Performance estimation

An accurate performance estimation of the functional modules is a key step of the architectural exploration flow. Made at early stage of the design flow, it gives area, time and memory use estimates which allows the evaluation of different solutions. HW-SW performance evaluation was addressed by several approaches. Depending on the estimation goals, approaches were interested to power/ time/ area estimations [77]-[81] ... The performance analysis was carried out analytically or through experiments. To predict software performances, some works were interested in the use of virtual target architecture such Instruction Set Simulator (ISS) component [82]. Others propose a fast estimation model using analytical methods. Generally, these methods are based on abstract models and cost functions to predict performances. They are frequently used for high level design exploration [83],[91].

For hardware performance estimation, many techniques have been explored at different levels of design abstraction. Some approaches were based on gate-level description providing good accuracy. Others use behavioral description which leads to a less accurate estimation.

The proposed estimation method is presented Figure 4.9. It is composed of two flows : SW-estimation and HW-estimation.

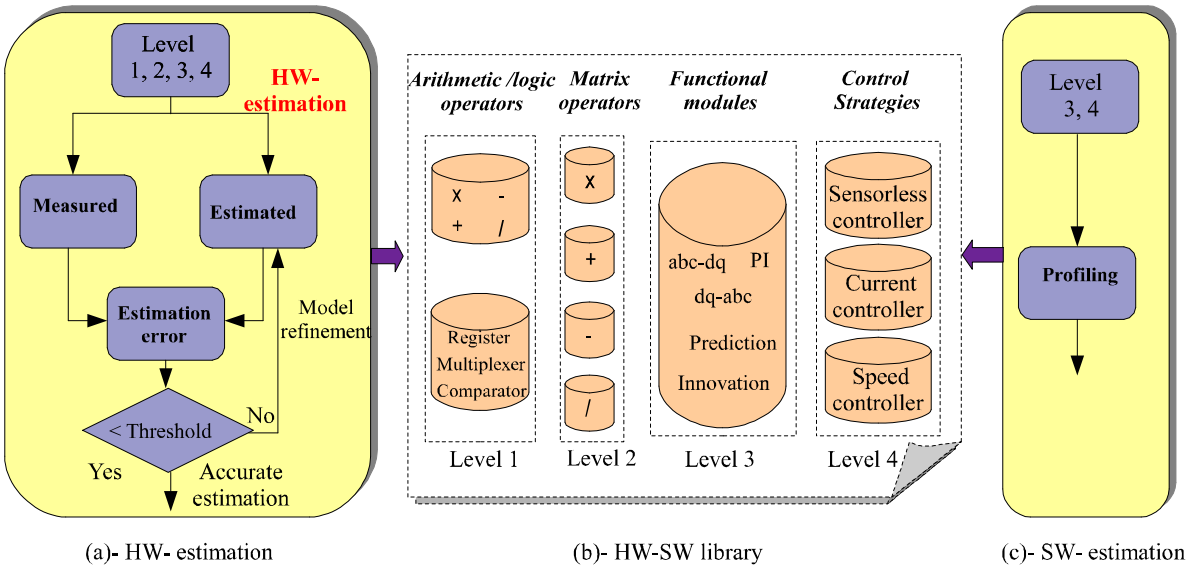


Figure 4.9: Estimation method, (a) HW-estimation, (b) HW-SW library, (c) SW-estimation.

The goal of the HW-estimation is to provide a generic flow adapted to different levels of granularity (fine, medium and coarse), different architectures (factorized or parallel) and variable bit-width of coding. The proposed flow is based on a library of control modules fully characterized in terms of area, time, memory and DSP blocks. These models were

refined several times until getting satisfied accuracy. This was performed by comparing the difference between the synthesized and the estimated value until this difference was under a low threshold value. As shown in Figure 4.9(b), the developed HW-library is composed of four levels of granularity [93]. The first level corresponds to the scalar arithmetic operators such as adder, multiplier, subtractor. The second level is related to the matrix operators. The third level consists in the functional modules used by the control (abc-dq transformation, PI-regulator, SVM...) and the functional modules used by the EKF observer (prediction, innovation, compensator...). Finally, the fourth level corresponds to functional modules of higher hierarchy level such as current controller, speed controller or the whole sensorless controller.

For SW-estimation, only the levels 3 and 4 are considered. Indeed, there are several parameters that influence the execution time of the software modules. These parameters depend on the used option for the Microblaze configuration (ex. hardware or precompiled multiplier), the used compiler, the optimization degree fixed in the compiler, the use of C or assembly code, the locality and the type of the used memory...

Therefore, by lack of precision in the development of the processor model, the SW-estimation of execution time and memory usage were made using profiling method. This method consists in the use of a timer that allows the measurement of the number of clock cycles taken by each SW module. It is worth to be noted that all the proposed measurements were made in the following conditions below:

-Custom instructions such as hardware multiplier, divider and barrel shift were configured to increase the processor timing performances.

-A Five-stages of pipeline was chosen.

-All the data and program were implemented in an embedded SRAM blocks (BRAM).

- The compiler was used without any optimization option.

4.4 Fine granularity library

As said before, the fine granulated level or the level 1 of the proposed library is constituted of the basic scalar arithmetic operators (adder, subtractor, multiplier...) and the logic functions (register, shift register, comparator...). All these operations are used frequently in control applications and digital signal processing. The focus of this estimation process is to give the area and the time features of each operators.

For the studied control application, the power consumption model was not considered. Indeed, the power consumption of digital platforms presents only a very low fraction of the drive.

As for the area estimation, the proposed models have been derived using regression approach and curve fitting. For each operator, the relationship between the consumed resources and the bit-width "N" have been defined. The proposed model is generic. We remind that all modules are coded with a fixed point-format denoted by $(i+f)Q_f$. where i and f present respectively the integer and fractional part.

The estimated area is divided into two main parts: LUTs and FFs. The first part considers all combinatory logic needed for example in arithmetic treatment. The second one is related to all sequential treatment. Table 4.4 presents the obtained results.

	Relationship Bit-width/ Consumed resources	
	6 – Bit LUTs	**FFs**
Multiplier	$Ceil(1.5 * N^2 + 3.5 * N - 31.6)$	—
Addition	$N + 2$	—
Substraction	N	—
Multiplexer	N	—
register	—	N
Shift	—	N

Table 4.4: Fine granularity library estimation results (area)

Since the multiplier is a greedy operator in terms of consumed resources, designers prefer, generally, the use of hardware multipliers. In Virtex-5 family, the proposed multipliers are 25x18 asymmetrical hardware multipliers. The relationship between bit-width of coding and the consumed hardware multiplier "HM" is expressed as below

$$HM = Ceil(\frac{N}{17}) * Ceil(\frac{N}{24}) \tag{4.1}$$

Based on the relation (4.1), the estimation results were compared to those measured after achievement from the synthesis process. Figures 4.10 and 4.11 show the measured and the estimated consumed resources for the multiplier. It shows a good fitting between the measured area and the estimated one. The results present an average and a maximum error respectively equal to 4% and 6%. It is an acceptable error of low impact in the accuracy of our estimation. We note that the small difference between the measured and the estimated values in the case of hardware multiplier is due to its asymmetric nature. These estimation results were also compared to the system generator estimator results and good accuracy was proved.

For the time estimation, only latency is considered. The time models for basic operators and Xilinx LogiCores are provided Table 4.5. We should note that the designer must choose carefully the operating frequency "Clk_{HW}" regarding the most net delay.

$$T_{Clk_{HW}} = Max(net(Op_i)) \tag{4.2}$$

Where Op_i is the operator i.

Operators	**Latency(CLK)**
Addition/substraction	1
saturation/count	1
Divider	M+R+5
25x18 Multiplier	$3+Ceil(\frac{N}{17}) + Ceil(\frac{N}{24})$

Table 4.5: Latency estimation for basic operators

where M=dividend width, R=fractional remainder width.

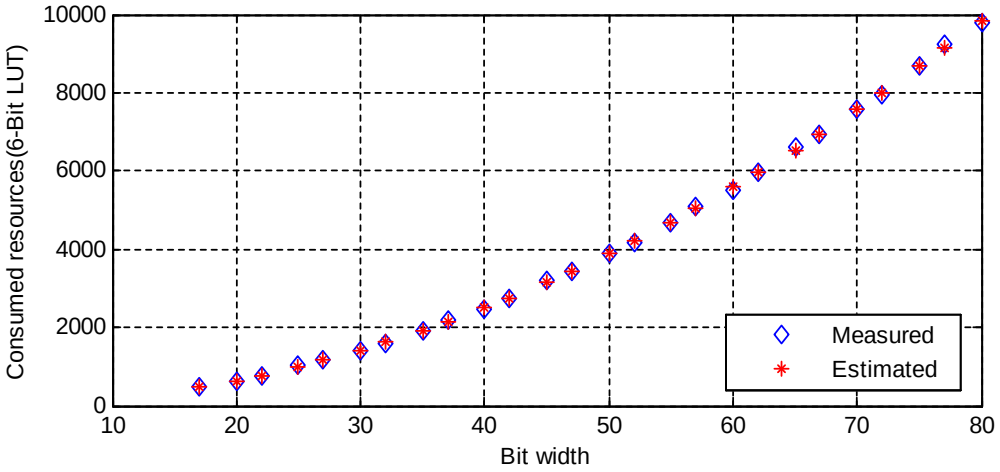


Figure 4.10: Consumed resources for a synthesized multiplier.

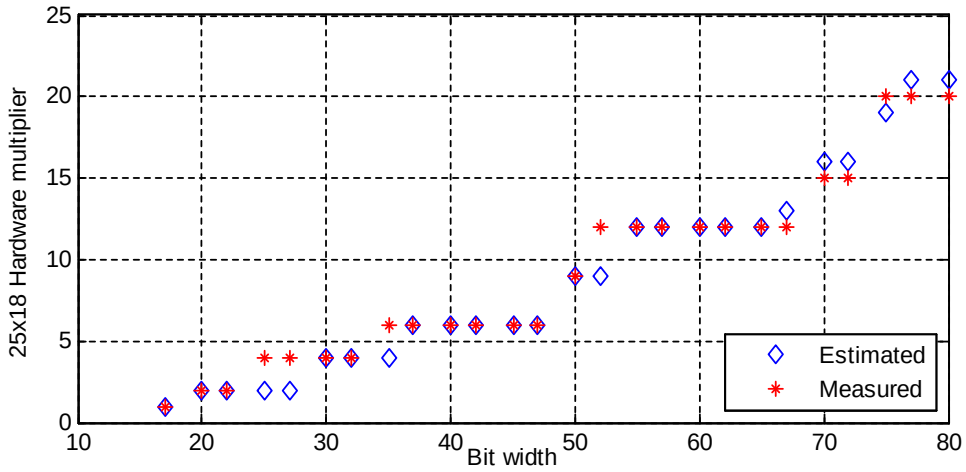


Figure 4.11: Consumed resources for a hardware multiplier

4.5 Medium granularity library

The medium granularity level 2 contains matrix operators and other modules such as saturator, counter and comparator. The area and time estimation of these modules is presented Table 4.6.

	6 – Bit LUTs	**FFs**	**Latency(Clock cycle)**
Saturator	$Ceil(3.7 * N - 1.5)$	—	1
Counter	$Ceil(12.5 * N - 8.1)$	N	1
Comparator – 3	$Ceil(3.1 * N - 4.2)$	—	1

Table 4.6: Medium granularity library

These models were verified by comparing the estimations to post-synthesis measurements. The results given Figure 4.12 proves a good fitting.

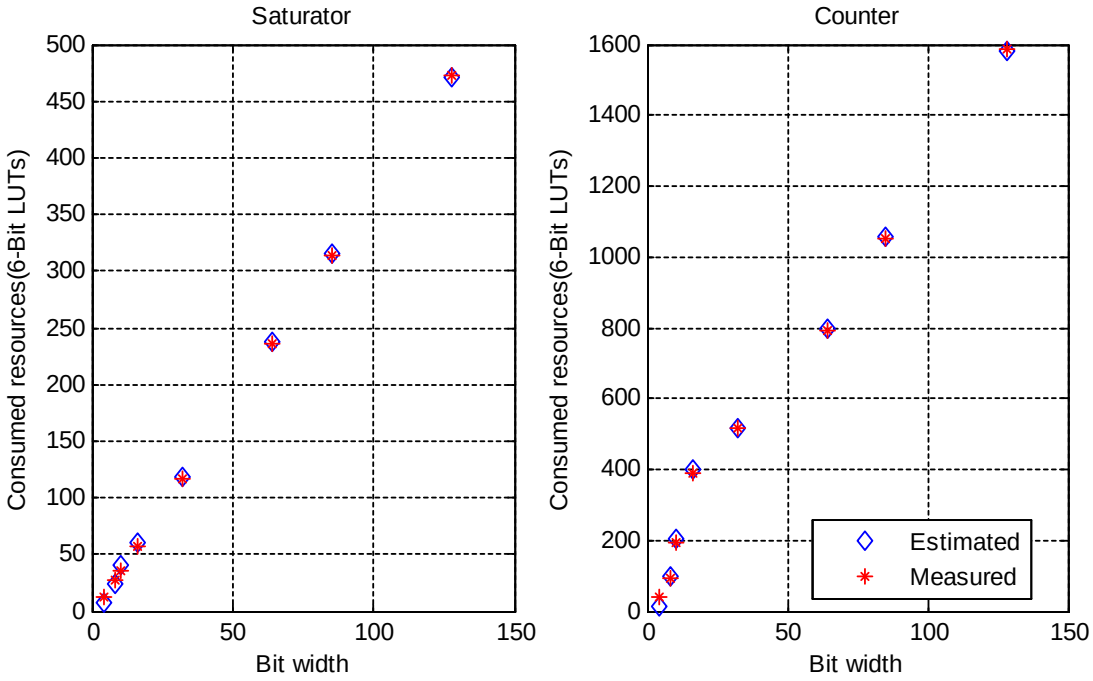


Figure 4.12: Consumed resources function of bit width.

Besides, the sensorless control algorithm based on Kalman filter uses matrix treatments. These later are often subject of optimization (area, energy, time). Table 4.7 presents a comparison between the factorized and the parallel architecture implementation of a matrix multiplication. The dimension of matrices was set to four. The factorization was based on the A^3 methodology [44]. One considers that after every operation, the result is saved via a register. More details in the developed architecture are provided in [113] and Appendix-B.

	Number of Operators	
	Parallel architecture	Factorized architecture
Multiplier	64	4
Addition	48	3
register	112	17
Multiplexer	0	10
Total Consumed resources	64 HM 1152 $6-Bit\ LUTs$ 2464 FFs	8 HM 292 $6-Bit\ LUTs$ 374 FFs
Latency (Clock cycle)	6	48

Table 4.7: Architecture performance of matrix multiplier(dimension=4 and bit-width=22)

4.6 Coarse granularity library

The coarse granularity library (level 3) is composed of functional modules such as the abc-dq and dq-abc transformations, the PI regulator...Based on the modularity principle, each of these module was developed separately allowing to be re-used in other control applications.

In the considered application, this grain level seems to be the most appropriate to be used in the HW-SW partitioning optimization process. Indeed, partitioning depends strongly on the chosen granularity level. The use of a fine-grained population of functional modules, typically the arithmetic modules, can give good results in terms of optimization but at the cost of a higher complexity of the treatment since the considered population is large. Another important issue concerns the loss of physical meaning of the functional modules to be optimized. Thus, using the level 3 modules, the number of blocks to be partitioned is significantly reduced and their physical meaning preserved. For this reason, the characterization of level 3 modules is of prime importance. Figure 3.13 presents the related metrics used for characterizing each module "Mi", both in hardware and software.

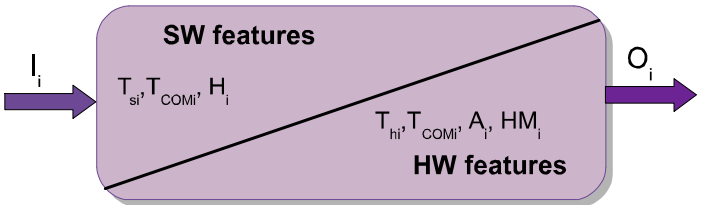


Figure 4.13: Features of the level 3 functional module Mi

Where

- A_i represents the consumed resources of module i, in the case of hardware implementation. It is expressed in terms of LUTs and FFs.
- H_i represents the size of the memory used by module i, in the case of software implementation.
- t_{hi} represents the execution time taken by the module i when executed in HW.
- t_{si} represents the execution time taken by the module i when executed in SW.
- HM_i represents the number of hardware multipliers or DSP blocks of the module i, in the case of hardware implementation.
- Ii/Oi represent the number of Inputs / Outputs of the module i.
- T_{COMi} represents the communication time between module i and the other modules.

Each of these metrics is detailed below.

4.6.1 Area estimation

The area estimation of each functional module depends mainly on two parameters: the bit-width of coding and the degree of parallelism. For the first parameter, the setting of fixed-point format is performed using fixed-point simulations as explained in section 3.4.4. While the degree of parallelism can be fixed according to the Data Flow Graphs (DFG) of the module. This degree of parallelism is directly linked to the data dependency of the corresponding DFG.

Based on the DFG, the factorization of repetitive operations can be performed using the A^3 methodology [44]. This approach targets to factorize the greediest operators (multipliers). This leads to locally serialize the treatment and then achieve a maximum of arithmetic operations with a number of operators fixed by the designer. As for the parallel approach focuses on the exploitation of all the potential parallelism contained in the studied algorithm. This leads to a reduced execution time but much more consumed resources. The data transfer within the DFGs is ensured by the Control Flow Graph (CFG).

To estimate the whole consumed resources of a given module M_i, we must estimate the area taken by the two parts: the DFG and the CFG. Thus, the number of FFs can be estimated based on the number of data registers used in the DFG and the finite state machine used in the CFG. The estimation of the LUTs is equal to the sum of the number of LUTs used in every elementary operator "j" that constitute the module M_i. A similar procedure is used to count the number of HW multiplier blocks. The consumed resources "A_i " and hardware multipliers are then evaluated using the following relations:

$$A_i(N) = \begin{cases} CFG = \begin{cases} LUTs = \sum Combinatorial\_part(CFG) \\ Flip-Flop = \sum FFs(FSM) \end{cases} \\ \\ DFG = \begin{cases} LUTs = \sum_{j=1}^{n} LUTs(operator_j(N)) \\ Flip-Flop = \sum_{j=1}^{n} FFs(register(N)) \end{cases} \end{cases} \tag{4.3}$$

$$HM_i(N) = \sum_{j=1}^{n} HM_j(N) \tag{4.4}$$

Where n is the number of elementary operators used in the functional module M_i and N the number of bits of coding.

As an example, Figure 4.14 presents the parallel and factorized DFG and CFG of a dot product of 2 vectors of dimension 3

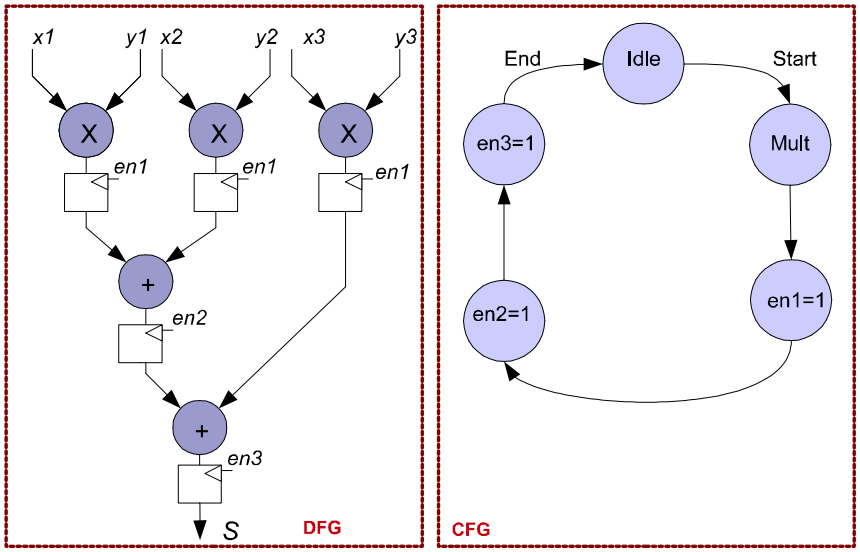
$$S(t) = x1(t).y1(t) + x2(t).y2(t) + x3(t).y3(t) \tag{4.5}$$

Thus, each functional module can be synthesized at least in two ways depending on the chosen degree of parallelism (a full parallel architecture and factorized one). The derived consumed resources metric A_i and HM_i are

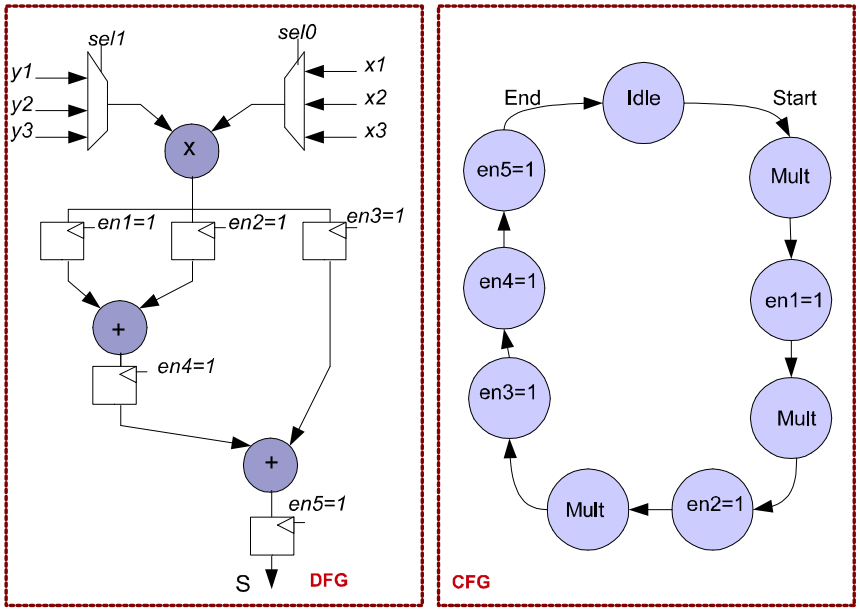
$$A_i(N) = \begin{cases} A_{if}(N) & case\ \ of\ \ Factorized\ \ DFG \\ A_{ip}(N) & case\ \ of\ \ Parallel\ \ \ \ DFG \end{cases} \tag{4.6}$$

$$HM_i(N) = \begin{cases} HM_{if}(N) \ case \ of \ Factorized \ DFG \\ HM_{ip}(N) \ case \ of \ Parallel \ DFG \end{cases} \quad (4.7)$$

The first estimated value "A_{if}" is based on a factorized architecture using the A^3 factorization methodology. The second estimated value "A_{ip}" is based on a full parallel architecture. The "HM_{if}" is estimated in the case of a factorized architecture, else this metric is expressed by "HM_{ip}".



(a) Parallel DFG and CFG



(b) Factorized DFG and CFG

Figure 4.14: Parallel and factorized DFG and CFG of the dot product example.

The proposed estimation method was validated using several benchmarks. Tables 4.8 and 4.9 present a good fitting between estimated and measured consumed resources (in

terms of LUTs and FFs) for different level 3 functional modules. It consists on factorized architectures. These results show a good accuracy of the estimation. The maximum and average error are respectively equal to 9% and 6%. The used number of bits N is presented section 3.4.4

	LUTs estimated	LUTs measured	Error(%)
$abc - dq$ transformation	207	204	+1.47
$dq - abc$ transformation	230	212	+8.49
PI regulator	80	90	-11.11
SVM	390	422	-7.58

Table 4.8: Estimation results in terms of LUTs

	FFs estimated	FFs measured	Error(%)
$abc - dq$ transformation	184	204	-9.8
$dq - abc$ transformation	155	165	-6.06
PI regulator	120	110	+9.16
SVM	143	131	+9.09

Table 4.9: Estimation results in terms of FFs

4.6.2 Time estimation

The time estimation of each functional module M_i was performed both in hardware and software. As indicated before, the time taken by the processor to execute a module was measured by profiling method. This choice was made due to the lack of precision of processor estimator model. From the hardware side, the estimation is more accurate and the whole execution time can be expressed by equation (4.8). Hence, the execution time is the result of the multiplication of system clock period "$T_{clk_{HW}}$" by the sum of the control states used in the CFG and the latency of each elementary operator "j" used in the DFG of the functional module Mi.

$$t_{hi} = \left(\sum_{j=1}^{n} latency_j + \sum control\_state \right) * T_{clk_{HW}} \tag{4.8}$$

Here again, the degree of parallelism affects the execution time depending on the considered hardware DFG (parallel or factorized). Indeed, the serialization of the treatment induces an increase of the CFG number of cycles, and consequently, the increase of the total execution time.

The execution time of a functional module can be then expressed as below

$$\begin{cases} t_{hif} \quad HW\ case \quad (Factorized\ DFG) \\ t_{hip}\ HW\ case \quad (Parallel\ DFG) \\ t_{si} \quad SW\ case \end{cases} \tag{4.9}$$

The whole execution time of the controller depends also on the communication process between all the used functional modules. In fact, the communication between software and hardware parts is very crucial. But, generally, the modeling of communication interface are frequently neglected. This can however produce an overhead on architectural performances, especially in the case of real-time applications.

To overcome this problem, we propose a communication model, as depicted Figure 4.15. It consists on four categories of communications. It is worth to be noted that in the case of hardware-software communication, all the data exchanged are extended to 32-bit. The proposed communication model is given as follows

1-Communication « C_i^{hs}» is the communication time from a hardware module « M_i » to a software module « M_{i+1}». This latter will ensure the reading of outputs data "O_i" of the module « M_i ». Thus the C_i^{hs} can be expressed as below

$$C_i^{hs} = Read\_cycle * O_i \tag{4.10}$$

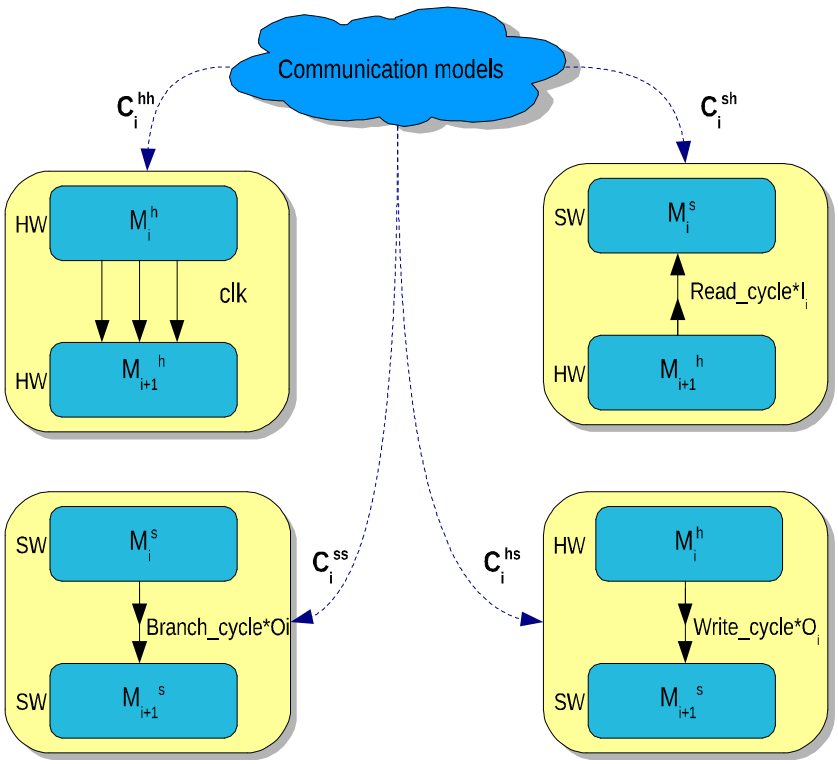


Figure 4.15: Communication models

2- Communication «C_i^{sh}» is the communication time from a software module « M_i » to a hardware module «M_{i+1}». The module « M_i» will write its data outputs on the inputs registers of module «M_{i+1}» . The C_i^{sh} is expressed as below

$$C_i^{sh} = Write\_cycle * O_i \tag{4.11}$$

3-Communication «C_i^{hh}» is the communication time from a hardware module « M_i » to a hardware module «M_{i+1}». All data are transmitted simultaneously in one clock cycle.

4-Communication «C_i^{ss} » is the communication time from the software module « M_i » to the software module «M_{i+1}». We consider that in every clock cycle a transfer of data is performed. Then, the communication execution time is defined by

$$C_i^{ss} = Clock\_cycle * O_i \tag{4.12}$$

Read/write cycle has been fixed to 3 clock cycles. This choice is based on the timing diagram of Microblaze PLB bus [73]. Author is aware that this communication model is simple that can be improved in future work.

4.6.3 Memory use

The memory use "H_i" of each functional module M_i was measured based on a profiling approach. We note that all modules were implemented using C language and with $32Q_{30}$ fixed-point format.

4.6.4 Parallelism parameter

The parallelism level is important for real time applications. The use of SoC system that includes HW resources and a processor can provide simultaneous and parallel execution of several functional modules. In this sense, we define a parallelism parameter "y" for each module. This parameter depends on the scheduling and the data dependency between modules. It is also mainly based on the designer ability to define which module «M_i» can be executed simultaneously with module «M_{i+1}». As shown in Figure 4.16, we define this parameter as a vector $y_i = [y_1, y_2...y_{n-1}]$, where $y_i \in \{1, 0\}$, with y_i=1(respectively $y_i = 0$) indicates that «M_i» can be executed in parallel (respectively sequentially) of «M_{i+1}». In this case, the maximum execution time of the two modules is only considered. A parallel execution of modules can be assigned to two modules both implemented in hardware or one in software and the other one in hardware. The case of SW-SW modules was not taken into account. Indeed, the studied control application does not requires the use of MPSoCs approach because of its medium range complexity.

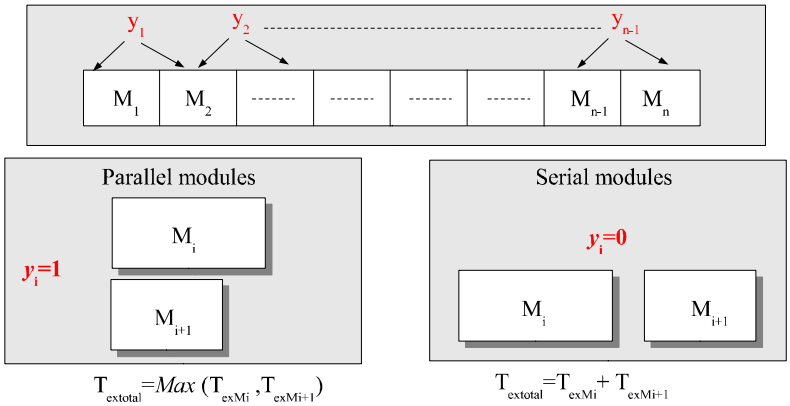


Figure 4.16: Parallelism parameter.

Finally, it has to be mentioned that the power estimation is also an important metric, especially for portable applications (with critical battery life time) and in some way for applications functioning at high temperature. This metric was not investigated in this work, but, it is a perspective.

4.6.5 Benchmark: EKF Sensorless speed controller

In order to validate the proposed methodology, a benchmark based on the EKF sensorless speed controller has been chosen. To efficiently manage this partitioning problem, the synoptic of the whole EKF-based sensorless speed controller was translated into an intermediate representation based on the Data Flow graph (DFG) .

As shown in Figure 4.17, the DFG is denoted by a set of nodes and edges. The nodes present the functional modules "M_i" while the edges denote the data dependencies between modules. Additionally, the potential parallelism between modules and the number of the exchanged data are shown. Moreover, each of the modules has a number reflecting its natural scheduling order in the control process. The ADC interface module and the SVM module are implemented in HW. Indeed, the resolution and functional performances required by these modules cannot be provided by SW. Thus they are not considered in the partitioning process.

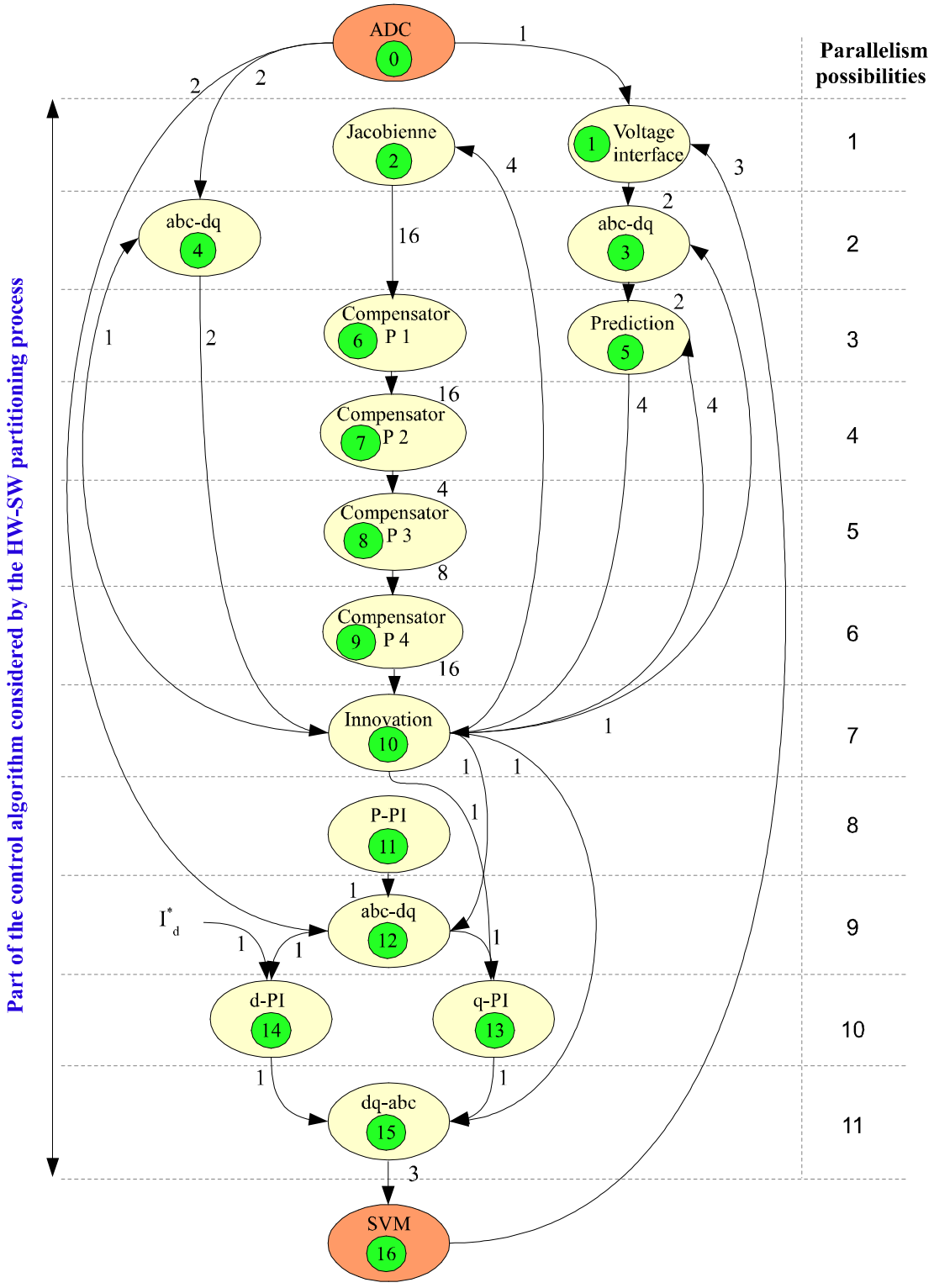


Figure 4.17: The DFG of the EKF-based sensorless speed controller.

Tables 4.10 and 4.11 present the set of the metrics of each functional module.

Modules«M_i»	$\mathbf{A_{if}}$	$\mathbf{HM_{if}}$	$\mathbf{A_{iP}}$	$\mathbf{HM_{ip}}$
ADC Interface	$54LUT$ $75FF$	$0HM$		
Voltage Interface (VI)	$15LUT$ $41\ FF$	$2HM$	$2LUT$ $44FF$	$6HM$
abc_dq	$207LUT$ $184FF$	$2HM$	$165LUT$ $92FF$	$12HM$
Prediction	$314LUT$ $198FF$	$2HM$	$144LUT$ $352FF$	$20HM$
Jacobian matrix	$68LUT$ $110\ FF$	$2HM$	$24LUT$ $110\ FF$	$8HM$
Compensator P1	$1336LUT$ $704FF$	$8HM$	$2688LUT$ $11880FF$	$256HM$
Compensator P2	$1336\ LUT$ $704FF$	$8HM$	$1632\ LUT$ $3432FF$	$192HM$
Compensator P3	$1443\ LUT$ $1443FF$	$8HM$	$1259LUT$ $2169FF$	$100HM$
Compensator P4	$1688LUT$ $1056FF$	$8HM$	$1912\ LUT$ $2574FF$	$192HM$
abc_dq	$207LUT$ $184FF$	$2HM$	$165LUT$ $92FF$	$12HM$
Innovation	$208LUT$ $220FF$	$2HM$	$236LUT$ $396FF$	$16HM$
P – PI regulator	$124\ LUT$ $160FF$	$2HM$	$84LUT$ $160FF$	$6HM$
abc_dq	$207LUT$ $184FF$	$2HM$	$165LUT$ $92FF$	$12HM$
q – PI regulator	$80\ LUT$ $120FF$	$2HM$	$64LUT$ $120FF$	$4HM$
d – PI regulator	$80\ LUT$ $120FF$	$2HM$	$64LUT$ $120FF$	$4HM$
dq_abc	$230LUT$ $155FF$	$1HM$	$191LUT$ $65FF$	$8HM$
SVM	$390LUT$ $143FF$ $0HM$			

Table 4.10: Consumed resources of the EKF-based sensorless control algorithm

Modules«M_i»	$\mathbf{y}_i$	$\mathbf{t}_{hip}$	$\mathbf{t}_{hif}$	$\mathbf{t}_{si}$	$\mathbf{I}_i/\mathbf{O}_i$	$\mathbf{H}_i$
ADC interface	0	240(100Mhz)		—	3/3	—
Voltage interface(VI)	1	5	11	256	4/2	5
Jacobian matrix	0	6	8	840	4/16	6
abc_dq	1	10	27	380	3/2	44
abc_dq	0	10	27	380	3/2	44
Prediction	1	8	23	327	6/4	9
Compensator P1	0	15	101	1100	48/16	37
Compensator P2	0	15	101	1310	24/4	31
Compensator P3	0	89	183	1385	4/8	30
Compensator P4	0	15	101	1514	32/16	42
Innovation	0	8	25	414	10/4	10
P – PI regulator	0	12	21	276	2/1	5
abc_dq	0	10	27	380	3/2	43
q – PI regulator	1	10	14	173	2/1	4
d – PI regulator	0	10	14	173	2/1	4
dq_abc	0	9	25	380	3/3	44
SVM	—	4		—	3/3	—

Table 4.11: Execution time of the EKF-based sensorless control algorithm

All the times in Table 4.11 are given in number of clock cycles.

Figures 4.18 to 4.21 provide comparison of the total consumed resources and execution time of the studied application (3 architectures are investigated (HW factorized architecture, HW parallelized architecture and SW Microblaze architecture). Regarding the two HW architecture, it is clear that the parallel solution consumes more in terms of LUTs, FFs and hardware multipliers than the factorized one. But, the execution time is twice as higher. On the other hand, the SW solution consumes less area but it takes much more execution time to compute the studied algorithm. As a conclusion, each of these extreme cases (pure software and pure hardware) have some advantages and some limits. That is the reason why, in the case of area-constrained applications, an efficient HW-SW partitioning is necessary.

Thus, all the metrics of each functional module will be integrated to the partitioning algorithm to find the optimal HW-SW solutions which respect the functional constraints

(maximum execution time "T_{Alg}") and architectural constraints (available area, hardware multipliers, memory use).

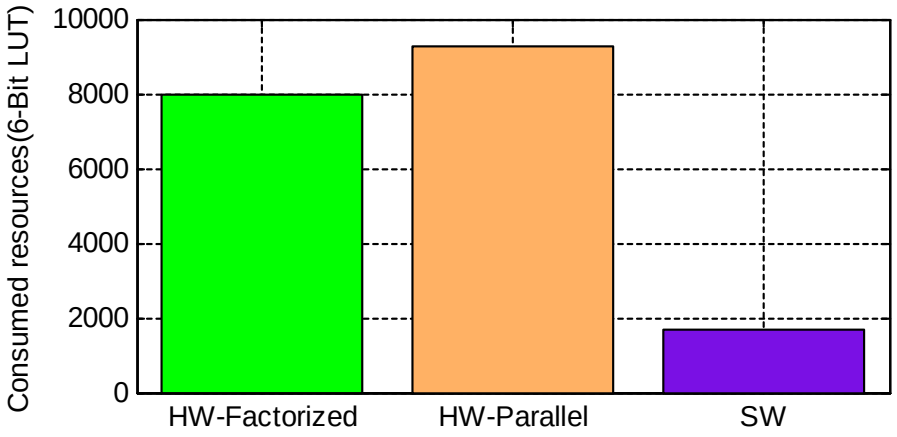


Figure 4.18: Consumed resources in terms of 6-Bit LUTs of the EKF-based sensorless speed controller.

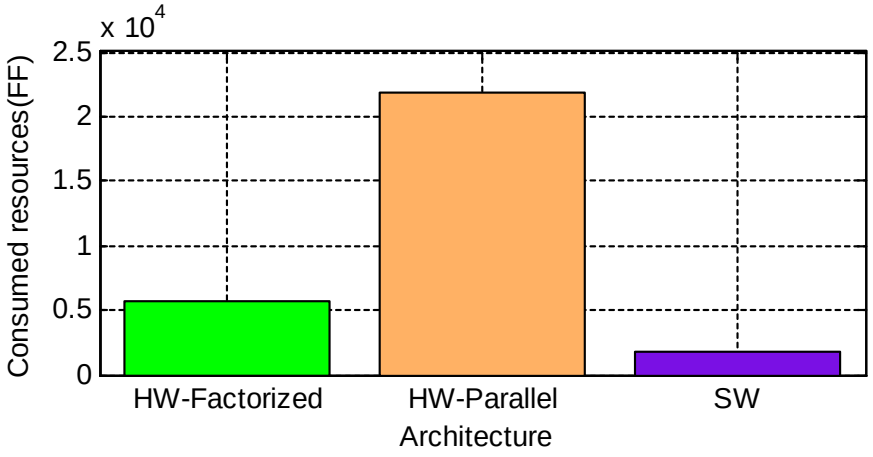


Figure 4.19: Consumed resources in terms of FFs of the EKF-based sensorless speed controller.

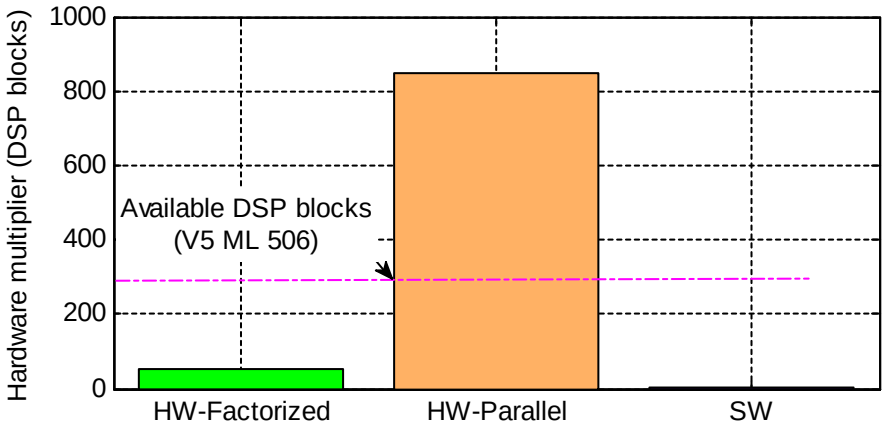


Figure 4.20: Consumed resources in terms of Hardware Multiplier (HM) of the EKF-based sensorless speed controller.

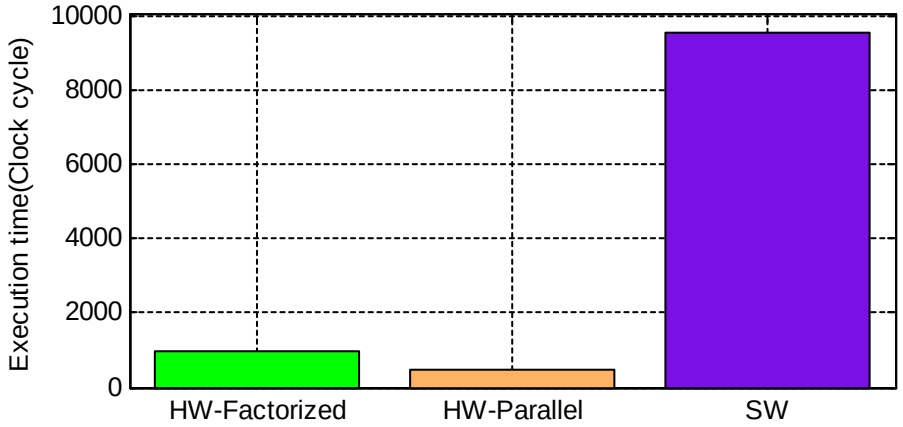


Figure 4.21: Execution time of the EKF-based sensorless speed controller.

4.7 HW-SW partitioning

The principle of HW-SW partitioning is to find optimal solutions among all the design alternatives which satisfy the requirements and the constraints of the studied application. The search for optimal partitioning is always guided by several constraints (performance, area, time...).

The partitioning problem is not new. It has been explored by numerous approaches. The most popular partitioning algorithms are heuristic [84],[88]. These approaches vary depending on several criteria as indicated below

-Granularity level

The choice of the granularity level (fine, medium and coarse grains) affects the performances of the partitioning algorithm. Indeed, the use of fine-grain provides larger number of feasible solutions. But in the same time, it presents high communication overheads leading to less accurate solutions [89], [90]. In this present work, the coarse grain (functional control modules such as PI-regulator, abc-dq transformation...) was adopted in order to respect the modularity principle and to avoid communication overhead.

-Objectives and constraints

An optimization problem focuses generally to reach one or several objectives. Additionally, the application requirements must be fulfilled without any constraint violations. The studied optimization problem consists on multi-objectives optimization problem. [91]-[97]. The goal is to minimize multiple objectives (area, time, memory use, hardware multiplier use). This optimization problem is also limited by architectural and functional constraints:

-The architectural constraints include the area, memory and HW multiplier limits inherent to the used digital platform (here the features of Virtex-5 (XC5VSX50T)).

-The functional constraints present the maximum execution time which must not be exceeded in order to respect certain stability margin to the controlled system.

-Search optimization algorithm

Numerous optimization algorithms were used for HW-SW partitioning. They can be divided into two main groups: the exact and the heuristic algorithms. As an example, the first group integrates branch-and-bound technique and integer linear programming [98],[99]. Most of the time, it deals with the resources-constrained scheduling. These methods are quite effective and suitable to solve simple problems. But, these methods

are becoming inappropriate especially for NP-hard problems [84].

The second group integrates heuristic algorithms which are well adapted to complex partitioning problem and many researchs were focused on these types of methods [100]-[102]. Among them is the Simulated Annealing (SA), the tabu search and the Genetic Algorithm (GA). In [96], simulated annealing was used to generate different partitioning solutions using fine-grained basic block level. It is based on a generic probabilistic metaheuristic for the global optimization problem of locating the global optimum. A partitioning approach based on tabu search algorithm was studied in [103] and [104]. It aims to avoid local optima by using memory structures. The aim was to find the best trade-off between the communications overhead between SW and HW parts and the reduction of the execution time. The GA has been also used widely in several HW-SW partitioning scenarii [105]-[107]. The key idea of this algorithm is to use the evolutionary genetic principle [108].

This algorithm is well-adapted to multi-objective optimization problems which is the case of our study. Thus it will be used in the following to ensure the HW-SW partitioning of control modules.

4.8 Formalization of the HW-SW partitioning problem

To handle HW-SW partitioning, a simple computing model was defined as an application composed by a set of n functional modules, denoted as $M = \{M_1, M_2, ...M_n\}$.

Figure 4.22 presents an implementation example for three functional modules. The implementation of each module can be made either in HW or in SW. All communication models are provided to ensure the data exchange between modules. We assume $(x_1, x_2, ...x_n)$ is an available solution of the HW-SW partitioning problem, where $x_i \in \{1, 0\}$, $xi = 1$ denotes a HW implementation of the module i and $x_i = 0$ denotes a SW implementation.

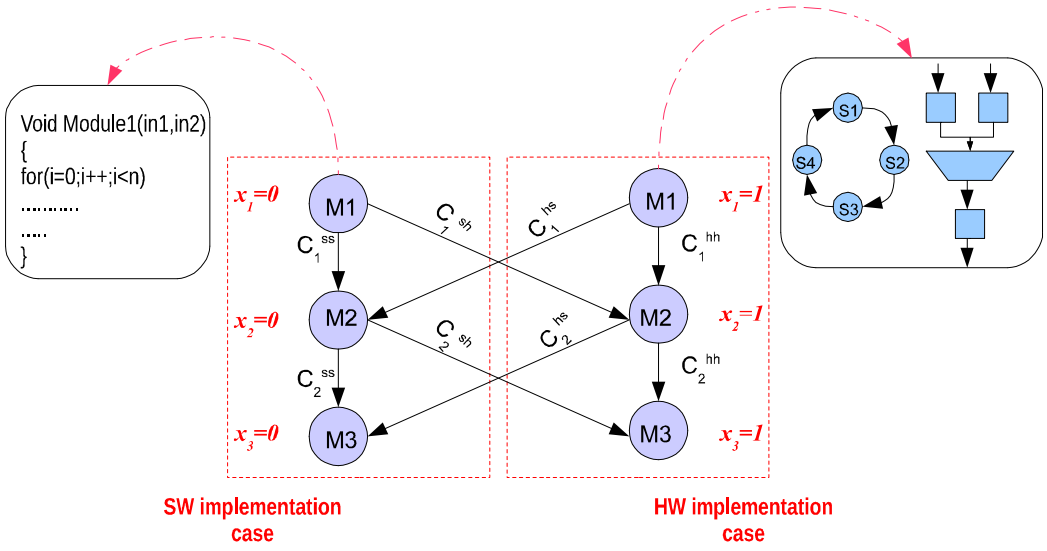


Figure 4.22: Computing model for HW-SW partitioning

The consumed resources "Area" present the total consumed resources occupied by each module implemented in HW. The use of the Microblaze soft processor core requires the consideration of the consumed resources taken by the processor. It includes the area occupied by the Microblaze itself which depends on the chosen configuration options (timer core, interrupt core, communication buses). The whole consumed resources by the software platform (Microblaze, hardware divider, buses (PLB, FSL)) are expressed by "$A_{\mu p}$". Thus the execution of only one module in SW requires the addition of processor area. The consumption of hardware multiplier blocks "HM_{blocks}" are also evaluated based on the same principle of area evaluation. Configuring hardware multiplier, in the Microblaze, leads to the use of some hardware multiplier blocks expressed by "$HM_{\mu p}$".

The total consumed resources can be formalized as

$$Area = \sum_{i=1}^{n} x_i * A_i + [(1 - x_i)\, or\, (1 - x_{i+1}) \ldots (1 - x_n)] * A_{\mu p} \tag{4.13}$$

$$HM_{blocks} = \sum_{i=1}^{n} x_i * HM_i + [(1 - x_i)\, or\, (1 - x_{i+1}) \ldots (1 - x_n)] * HM_{\mu p} \tag{4.14}$$

The total memory occupation can be expressed by equation (4.15). It corresponds to the sum of memory use for each module in the case of a software implementation. It is expressed in Kilo bit (Kb).

$$Memory = \sum_{i=1}^{n} (1 - x_i) * H_i \tag{4.15}$$

The total execution time of the control algorithm to be implemented includes execution time of modules implemented in HW and SW along with their communication times. Taken into account the parallelism parameter, the total execution time can be formalized as follows

$$T_{ex} = \sum_{i=1}^{n-1} \begin{bmatrix} y_i.x_i.x_{i+1}.\max(t_{hi}, t_{hi+1}) \\ +y_i.x_i.(1 - x_{i+1}).\max(t_{hi}, t_{si+1}) \\ +y_i.(1 - x_i).x_{i+1}.\max(t_{si}, t_{hi+1}) \\ +y_i.\overline{y_{i-1}}.(1 - x_i).(1 - x_{i+1}).(t_{si} + t_{si+1}) \\ +\overline{y_i}.\overline{y_{i-1}}.x_i.t_{hi} \\ +\overline{y_i}.\overline{y_{i-1}}.(1 - x_i).t_{si} \end{bmatrix} + \overline{y_{n-1}}\,[x_n.t_{hn} + (1 - x_n).t_{sn}] + T_{Com} \tag{4.16}$$

$$where, \quad T_{Com} = \sum_{i=1}^{n-1} \overline{y_i}. \begin{bmatrix} (1 - x_i).(1 - x_{i+1}).C_i^{ss} \\ +x_i.x_{i+1}.C_i^{hh} \\ +(1 - x_i).x_{i+1}.C_i^{sh} \\ +x_i.(1 - x_{i+1}).C_i^{hs} \end{bmatrix} \tag{4.17}$$

For a given constraints in terms of available area, memory and maximum execution time, the optimization problem of the resource allocation can be written as follows

-Objective functions :

$$Minimize \begin{cases} Area = \sum_{i=1}^{n} x_i.A_i + (1-x_i)\, or\, (1-x_{i+1}) \ldots (1-x_n)\, .A_{\mu p} \\ HM_{blocks} = \sum_{i=1}^{n} x_i.HM_i + (1-x_i)\, or\, (1-x_{i+1}) \ldots (1-x_n)\, .HM_{\mu p} \\ Memory = \sum_{i=1}^{n} (1-x_i)\, .H_i \\ T_{ex} = \sum_{i=1}^{n-1} \left[\begin{array}{l} y_i.x_i.x_{i+1}.\max(t_{hi}, t_{hi+1}) \\ +y_i.x_i.(1-x_{i+1}).\max(t_{hi}, t_{si+1}) \\ +y_i.(1-x_i).x_{i+1}.\max(t_{si}, t_{hi+1}) \\ +y_i.\overline{y_{i-1}}.(1-x_i).(1-x_{i+1}).(t_{si}+t_{si+1}) \\ +\overline{y_i}.\overline{y_{i-1}}.x_i.t_{hi} \\ +\overline{y_i}.\overline{y_{i-1}}.(1-x_i).t_{si} \end{array} \right] \\ \qquad +\overline{y_{n-1}}\left[x_n.t_{hn} + (1-x_n).t_{sn}\right] + T_{Com} \end{cases} \tag{4.18}$$

-Constraints :

$$Subject \begin{cases} Area < \%S \\ Memory < H \\ HM_{blocks} < HM \\ T_{ex} < T_{A\lg} \end{cases}$$

S, H and HM are, respectively, the area, memory size and Hardware Multiplier available on the digital platform (Virtex-5, XC5VSX50T). "$T_{A\lg}$" corresponds to the maximum allowable execution time with regard to the desired control bandwidth, sampling period and phase margin (see chapter 3).

In the following, the objective is to find the optimal HW-SW partitioning with lower execution time, area, memory and hardware multiplier use with respect to the Virtex-5 available resources. To do this, the GA was chosen for its efficiency and its recognized performances.

4.9 Genetic Algorithm : NSGA-II

The use of Genetic Algorithms (GA) in solving optimization problems was introduced, for the first time in 1975, by John Holland. This algorithm is inspired by Darwin's theory of evolution: natural selection of individual variations. Hence, it is based on biological phenomena. Such heuristics are very well suited for problems with multi-objective optimization goals. It has been successfully used for solving several problems in the field of VLSI systems such as the placement / routing, the optimization code for the DSP, the area-time optimization of FPGA design [109]-[111]. The GA is divided into two main categories [112]:

- **Approaches that use aggregating functions**: it is based on the reduction of the multi-objective problem to mono-objective one, by the combination of the objective functions. The main drawback of this approach is that it does not generate proper Pareto-optimal solutions in the presence of non-convex search spaces.
- **Pareto-based approaches**: The main principle is to find the set of individuals in the population that are Pareto non-dominated by the rest of the population of higher rank. The process is repeated until the ranking of the whole population is performed. In this sense, the Non-dominated Sorting Genetic Algorithm (NSGA-II) is one of the most promising.

The Non-dominated Sorting Genetic Algorithm (NSGA-II) is used for HW-SW partitioning under multiple objectives and constraints. It aims to find out the Pareto optimal front of the studied problem. The NSGA-II is characterized by a new ranking function. It classifies the candidate solutions considering all the objectives. Comparison between NSGA-II and Simple GA (SGA), which used the weight sum to convert the multi-objectives problem to a single objective, shows that the NSGA-II provides faster convergence. On the other hand, one important feature of the NSGA-II is the elitism. This means that the best solutions within one population are conserved for the next generation. After that, the population random selection (mutation) will guarantee the diversity of the population. This procedure is iterated until finding the Pareto-optimal front. So, the principle of NSGA-II is to compare each solution with the others in order to see whether it is dominated or not. Thus, it performs (m.N) comparison, where N is the number of populations and m is the number of objectives.

4.9.1 Principle

Genetics have revealed the existence of several operations within organism resulting in gene flow. These operations occur during the reproductive phase when the chromosomes of two organisms are merging. Figure 4.23 presents the different phases of the NSGA-II algorithm, that are explained below

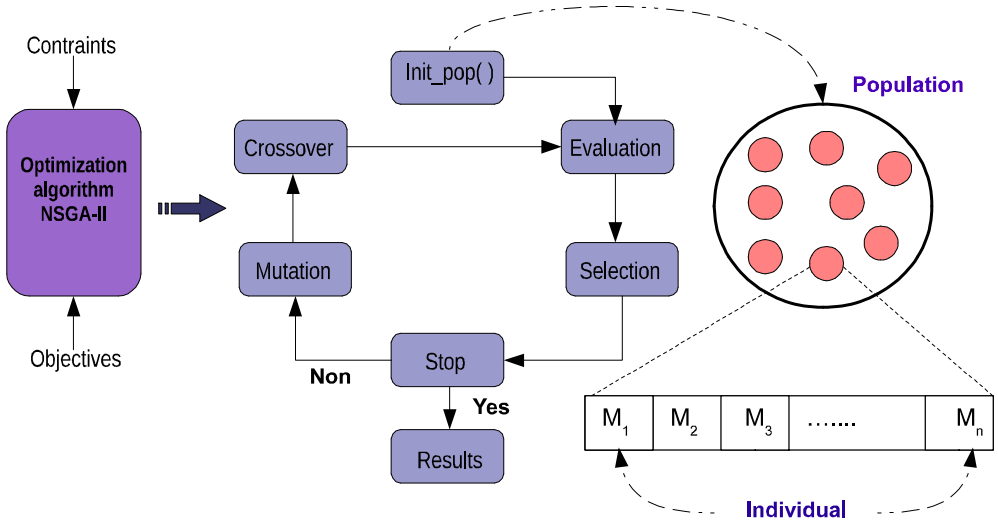


Figure 4.23: Genetic algorithm principle.

- **Initialization**

  At the beginning, the NSGA-II uses an initial population consisting of a set of solutions randomly generated. These individuals (or chromosome) will be evaluated during the evaluation step.

- **Evaluation**

  It consists in the evaluation of the performance of each individual. Thus the solutions are sorted according to the considered objectives and constraints. The aim is that a ranking selection method can highlight best solutions. Then, the non-dominated solutions present the Pareto- optimal front.

- **Selection**

  The selection is made by the determination of individuals which are more likely to get best results. This process is analogous to the process of natural selection, where strongest individuals are best suited to win the competition of reproduction, while less suitable die before breeding.

- **Crossover**

  During this operation, two chromosomes exchange some parts of their genes, as shown in Figure 4.24. There are several crossing points. The occurrence probability "P_c"of crossover between two chromosomes is an algorithm parameter. It is often applied to a high rate (90%).

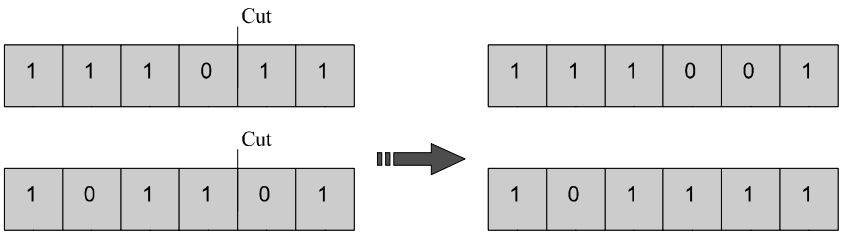


Figure 4.24: Crossover operator.

- **Mutation**

  In the same way as crossover, the mutation rate "P_m" is generally fixed between 0.1% and 1%. It aims to substitute randomly a gene, within a chromosome, by another. It is necessary to choose the rate relatively low so as not to fall into a random search and preserve the evolution principle.

4.9.2 NSGA-II configuration

The configuration parameters of the NSGA-II algorithm are presented in Table 4.12. The number of individuals corresponds to the number of functional modules. In our case, there are 15 modules (see Figure 4.17).

Parameters	value
Generations	200
Population size	200
P_c	1%
P_m	90%

Table 4.12: GA configuration

The estimated performances of each functional module were introduced to the NSGA-II optimization algorithm. This latter is running under Matlab workspace. The following considerations were taken into account:

- From area optimization point of view, only the factorized DFGs of control module were considered by the HW-SW partitioning process.
- To perform multiplication, the use of hardware multiplier was privileged leaving FPGA logic elements to perform others functions.
- Because of the intensive computational nature of the studied algorithm, only the area in terms of LUTs were considered for the HW-SW partitioning. In other words, the LUT elements present the most meaningful statics for comparing logic utilization between different technologies (Xilinx, Actel and Altera). More details are provided in the Appendix-C.

4.10 HW-SW partitioning results

As first stage of validation, the sensibility of the partitioning results with regard to the number of generations was studied. The NSGA-II algorithm was executed using the same benchmark (EKF sensorless speed controller) with 200 and 600 generations. Figure 4.25 presents the partitioning results for unconstrained resources and time. It shows the best trade-off between the four considered objectives (area, time, hardware multiplier and memory use). The dominate solutions constitute the Pareto-optimal front including all the set of optimal solutions.

It can be noted also that the number of candidate solutions increases significantly according to the iteration numbers. This is a main characteristic of the heuristic algorithms that are refined by sequential iterations. This provides more solutions but leads also to the increase of the NSGA-II execution time. In our case, the number of generations was set to 200 which ensures a good space exploration of feasible solutions and an acceptable execution time of the NSGA-II algorithm.

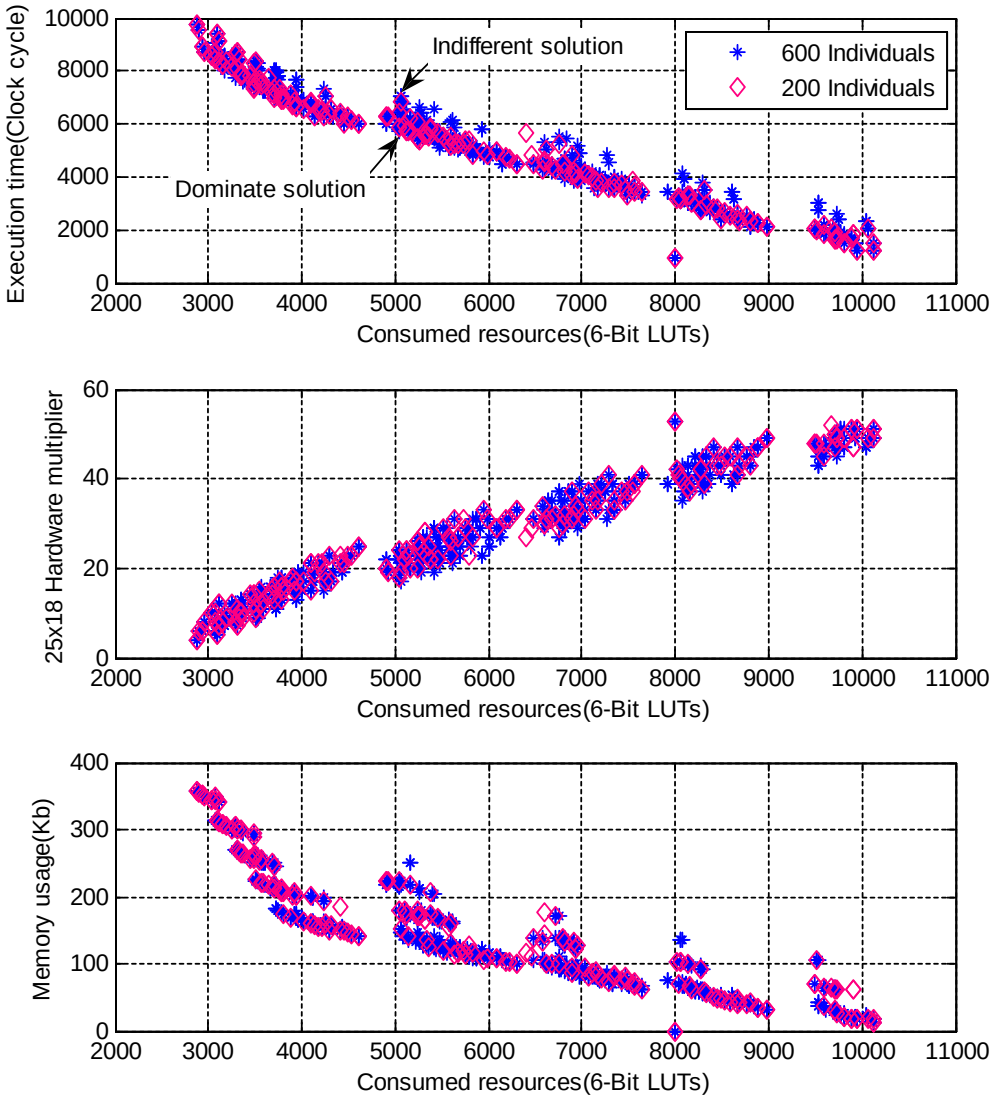


Figure 4.25: HW-SW partitioning results.

In the following, the trade-off (area-time) is taken to evaluate solutions. The impact of communication time and parallelism parameter has been presented in Figure 4.26. We note that without the consideration of communication time, the HW-SW solutions have less execution time. But, in actual embedded system, the overhead of communication is present. Thus, it must be considered in the partitioning process.

The second test highlights the advantage of using parallelism parameter which permits the reduction of execution time. This is due to the inherent parallelism of proposed SoC architecture which provides the possibility to execute in parallel two HW-HW, SW-HW, and HW-SW functional modules. We remind that SW-SW modules parallelism was not taken into account in this work.

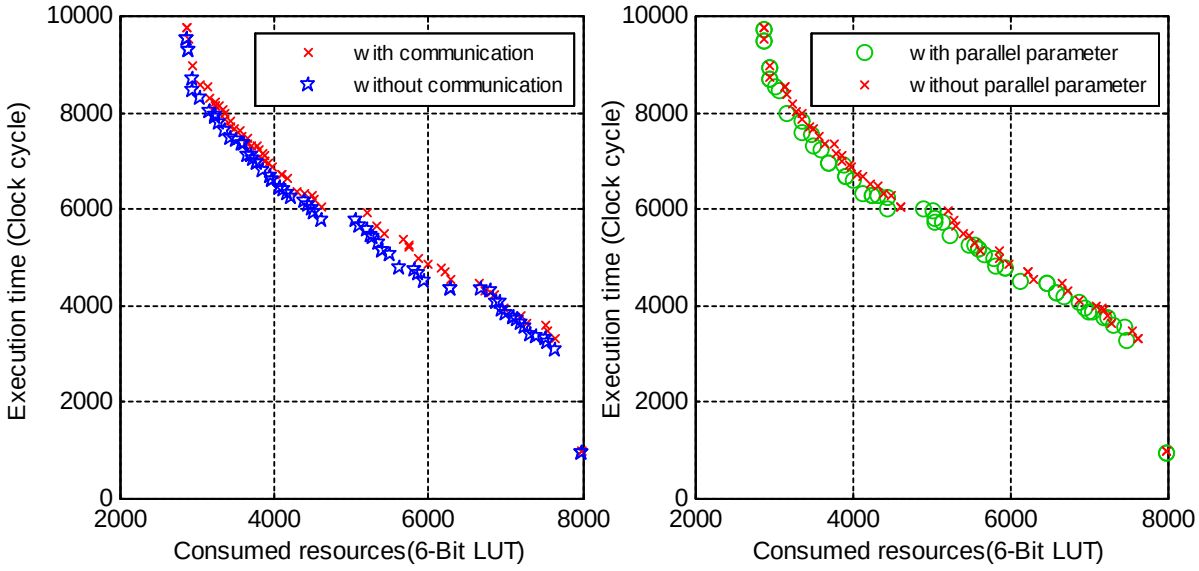


Figure 4.26: Communication and parallelism parameter impact on the HW-SW partitioning.

As far the HW-SW partitioning validation is concerned, the tests were performed based on three constraints, described below and Figure 4.27.

- *Technological constraints:* The constraints related to the use of different technology types (Altera, Xilinx and Actel) are considered. It permits the evaluation of the partitioning results respecting different technologies such as low /high cost Xilinx devices, Altera devices and Actel devices.

- *Functional constraints*: All the partitioning tests were performed for the two types of systems (presented in section 3.2) : low sampling rate and high sampling rate control systems. Thus, the maximum allowable execution time constraint varies regarding to the acceptable stability margin limits for the studied control systems. The high sampling rate control system concerns applications where time constraints are severe. In our case, we have considered a high sampling rate control system using switching frequency equal to 100 kHz. In the previous chapter, the study of time delay impact in the control performance showed that time delay "T_{Alg}" must not exceed 11.66 μs to get a stability margin higher than 60 °. For the low sampling rate control system, a switching frequency equals to 20 kHz was chosen. The stability constraints leads to maximum time delay equal to 58.33 μs.

- *Environmental constraints:* The type and the environment of the studied application have often an influence on the choice of the digital technologies and the operating Clock frequency. In typical aircraft applications, the need to use a non-volatile technology such as Actel ones is mandatory. The environment conditions (high or low temperature) impact also the choice of operating frequency. For high temperature condition which is often occurred in space and aircraft applications, designers are obliged to reduce the operating clock frequency.

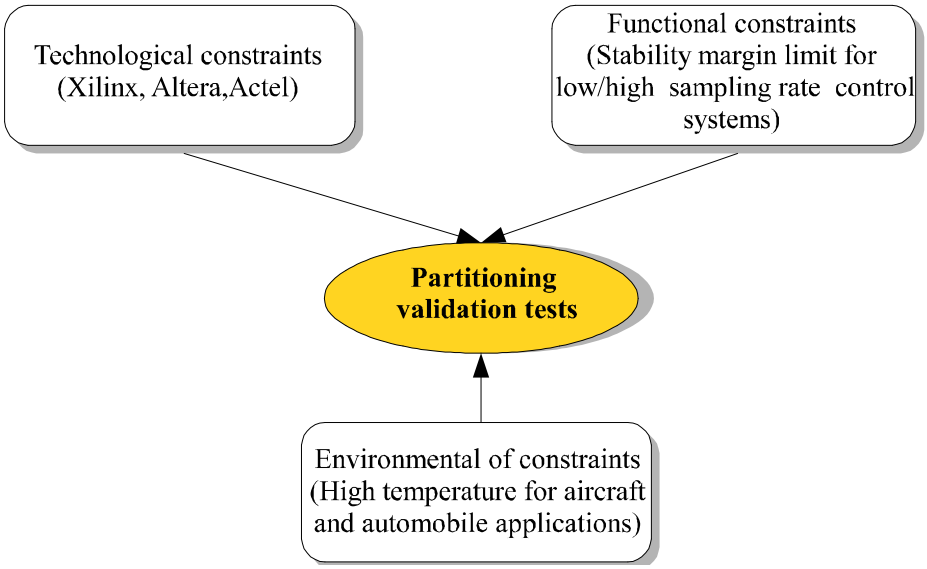


Figure 4.27: Tests strategies.

Based on Xilinx FPGA "Virtex-5", the partitioning tests were performed for two configurations : use of a soft processor core "Microblaze" and use of a hard processor core "Power-PC440". We note that the Microblaze associated to its communication bus, timer and interrupt cores consumes **2428 6-Bit LUTs**. In the case of the hard core, we consider only the consumed resources taken by the communication bus and the other peripherals.

A speedup factor was also defined allowing the evaluation of the Pareto-front solutions. This factor presents the ratio between the execution time of any HW-SW solution with regard to the execution time of a full SW implementation, taken as baseline.

Figures 4.28 and 4.29 present the HW-SW partitioning results. They present all the possible final architectures that can be realized, for the two configurations. We note that the NSGA-II algorithm provides a space exploration covering a good number of feasible solutions. Thus, designer can choose the appropriate allocation and the scheduling which best corresponds to his application.

Using Microblaze and in the case of high sampling rate control system, only one feasible solution (the pure hardware one) is possible because of the severe timing constraints. So in this case, it makes sense to use pure hardware architecture. For a less severe timing constraint, a wider range of solutions is possible.

Using Power-PC440, the partitioning results show that for the same design and in the case of high sampling rate, more results are provided. The candidate solutions are no longer restricted to the full hardware solution and more solutions can be realized which is not the case of Microblaze-based implementation.

In the following, we consider an area-constrained space exploration: the allowable consumed resources are fixed to **7000 6-Bit LUTs**. The rest of the SoC available resources will be used to integrate other functions such as monitoring, communication process that are not presented here. It is worth to be noted that this area constraint corresponds to the resources offered by a low cost Spartan-6 (XC6SLX16, 2278 Slices). In

this case, a HW-SW optimization process is mandatory to implement the speed sensorless control since a full hardware implementation can not be made by lack of area resources.

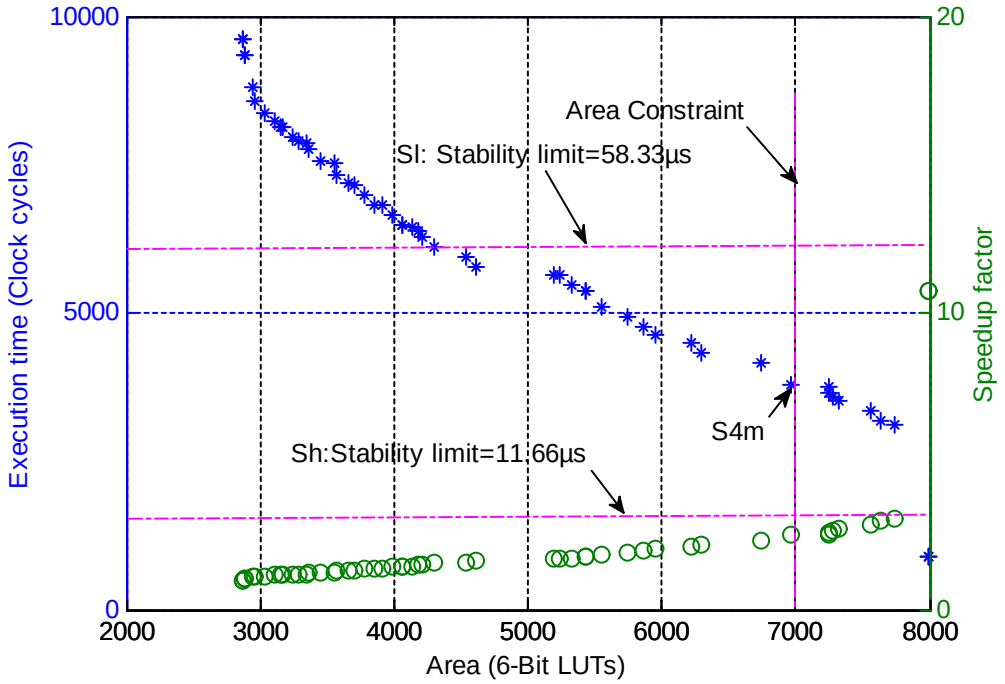


Figure 4.28: Partitioning solutions using Virtex-5 embedded Microblaze (XC5VSX50T,CLK=100MHz).

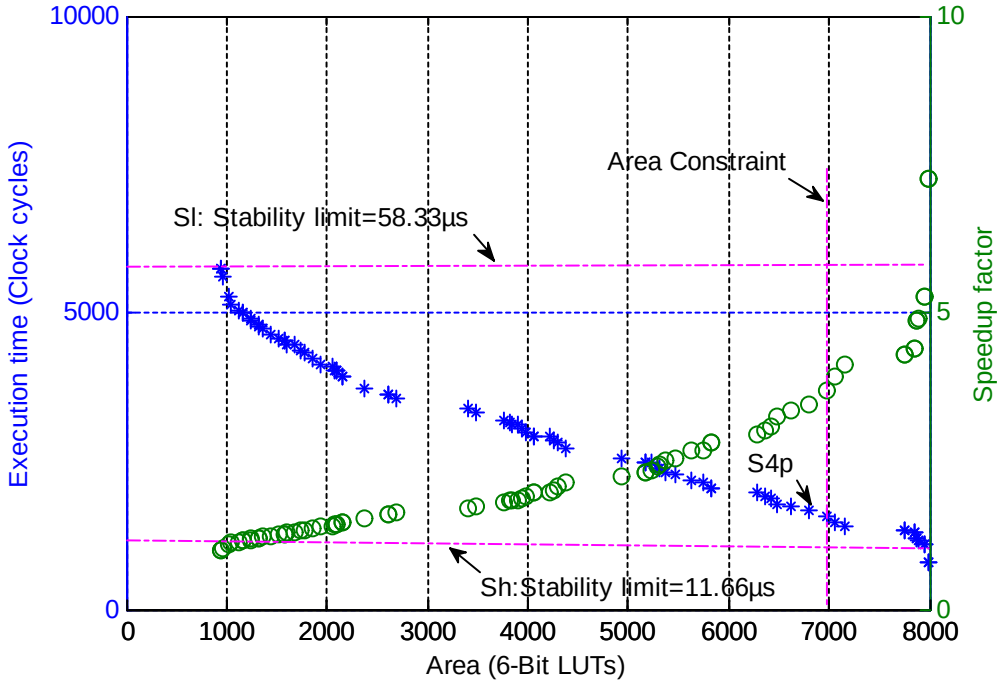


Figure 4.29: Partitioning solutions using Virtex-5 and PowerPC (XC5VFX30T, CLK=100MHz).

Tables 4.13 and 4.14 present some candidate solutions based on, respectively, Microblaze and Power-PC440. These results show that hardware accelerators can provide a

significant speedup with respect to the software baseline. The scheduling diagrams of the best solutions according to the area (7000 6-Bit LUTs) and time constraints (T_{Alg}), for low sampling rate control system, are presented Figures 4.30 and 4.31.

	Binary individual	*A (LUTs)*	*T_{ex} (Clock cycle)*	*Memory (Kb)*	*25x18 HM*	*speedup* factor
S_{1m}	11111100011111011	6220	4502	102	31	2.13
S_{2m}	11111100011111111	6300	4341	98	33	2.21
S_{3m}	11111011001111101	6740	4158	125	38	2.31
S_{4m}	11111011001111111	6970	3800	32	39	2.52

Table 4.13: Some of the solutions in the case of Microblaze (Virtex-5, XC5VSX50T)

	Binary individual	*A (LUTs)*	*T_{ex} (Clock cycle)*	*Memory (Kb)*	*25x18 HM*	*speedup* factor
S_{1p}	11111011101111111	6491	1759	51	43	3.26
S_{2p}	11111011011011111	6612	1721	45	41	3.33
S_{3p}	11111111101111111	6805	1658	42	45	3.46
S_{4p}	11111111011111011	6971	1557	35	43	3.68

Table 4.14: Some of the solutions in the case of Power-PC(Virtex-5, XC5VFX30T)

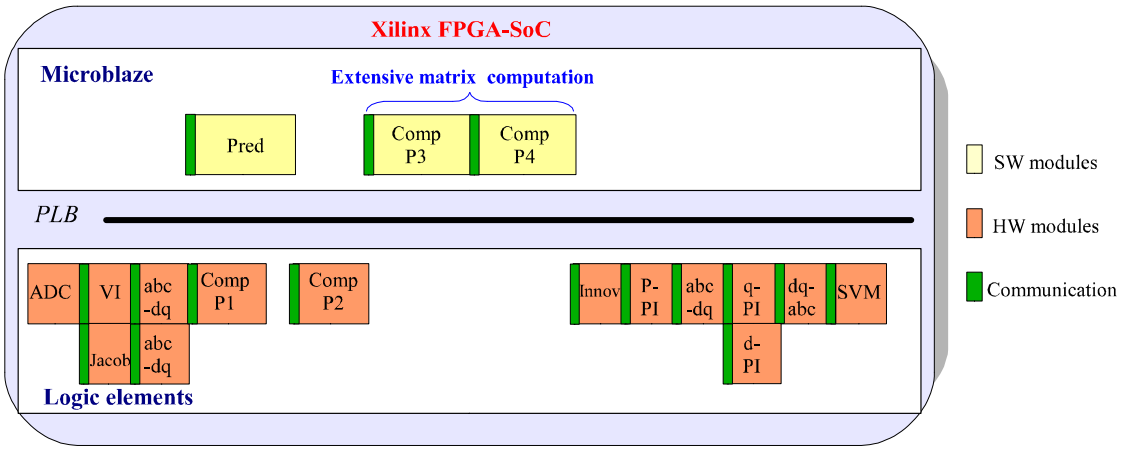


Figure 4.30: Scheduling diagram of the optimal solution "S4m" using Virtex-5 (XC5VSX50T) embedded the Microblaze.

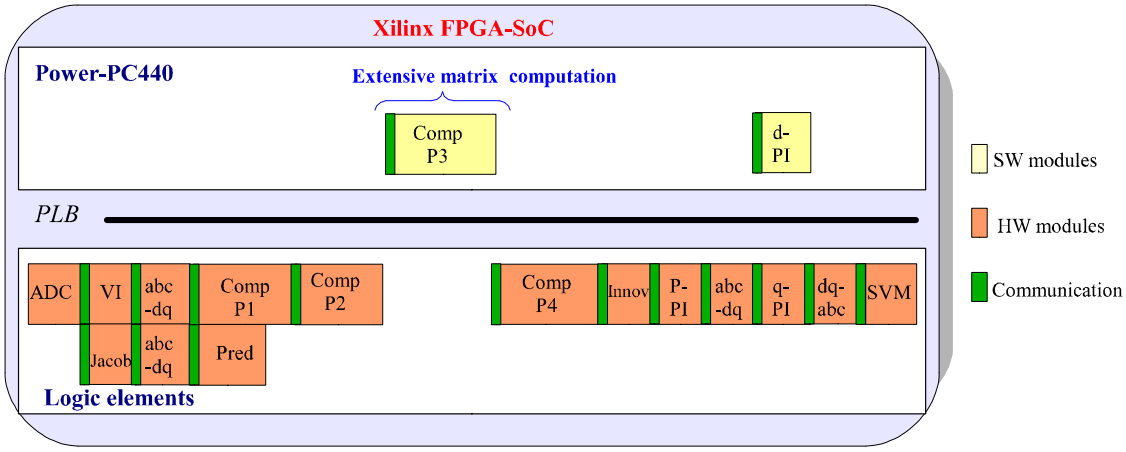


Figure 4.31: Scheduling diagrams of the optimal solution "S4p" using Virtex-5 (XC5VFX30T) and Power-PC440.

The evaluation of HW-SW partitioning algorithm was extended to other SoC platforms: low cost Xilinx platforms (Spartan-3E), low and high cost Altera platforms (Cyclone-II and Stratix-III) and Actel platforms (ProAsic-3). These tests provide to the designer an idea of the different feasible solutions using these platforms. The corresponding metrics of each functional module and the area constraints (7000 6-Bit LUTs) were recalculated, for each platform, based on the following relation (see Appendix-C)

$$\begin{cases} For\ Spartan\ \ family & Slice = \quad 1.5 \quad 6-Bit\ LUT \\ For\ Altera\ Startix-III\ \ family & ALM = \quad 1.84 \quad 6-Bit\ LUT \\ For\ Altera\ Cyclone\ \ family\ (ALM = 2.5\ LE) & LE = \quad 0.73 \quad 6-Bit\ LUT \\ For\ Actel\ ProAsic\ family & Versatile = \quad 0.25 \quad 6-Bit\ LUT \end{cases} \tag{4.19}$$

Figure 4.32 presents the partitioning results using Spartan-3E 3S1600E (14752 slices, 36 (18x18) hardware multipliers). The presented results were performed using Microblaze. We note that there is no solution for the high sampling rate control system. For the low sampling rate system, some solutions are provided. The number of hardware multiplier is limited to 36 which influences the number of the proposed partitioning results and limits the speedup factor.

The candidate solutions are presented by Table 4.15. The scheduling diagram of best solutions according to the area and time constraints is presented by Figure 4.33.

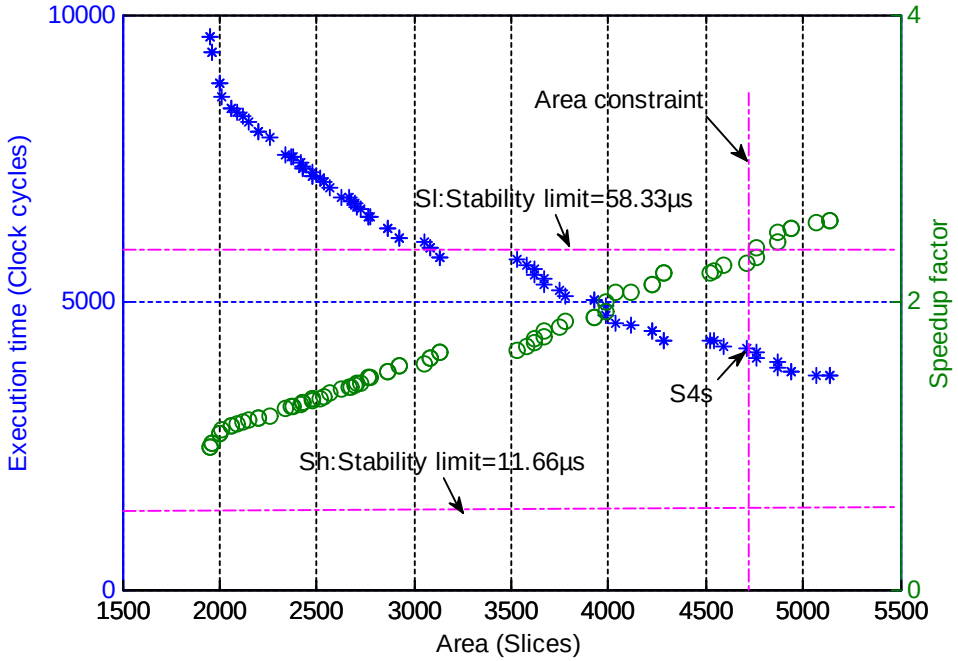


Figure 4.32: Partitioning solutions based on Spartan-3 embedded Microblaze (CLK=100MHz).

	Binary individual	*A* (*Slices*)	T_{ex} (*Clock cycle*)	*Memory* (*Kb*)	18*x*18 *HM*	*speedup* factor
S_{1s}	11111100011111111	4280	4341	99	33	2.21
S_{2s}	11111011001111001	4524	4317	130	36	2.22
S_{3s}	11110011001111011	4542	4311	129	35	2.23
S_{4s}	11111011001011011	4596	4230	91	35	2.27

Table 4.15: Some of the solutions using Microblaze with Spartan-3 FPGA

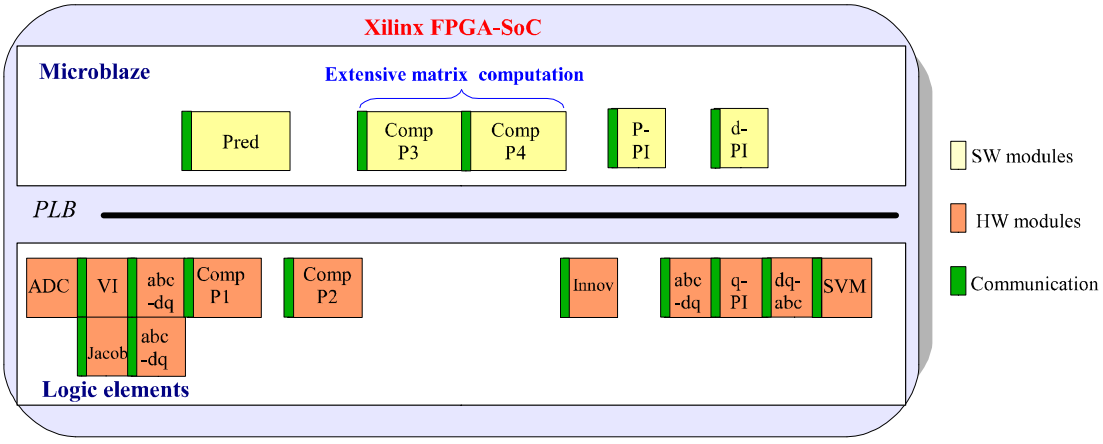


Figure 4.33: Scheduling diagram of the optimal solution "S4s" using Spartan-3 embedding the Microblaze.

Figures 4.34 and 4.35 present the partitioning results using a low cost solution from Altera company "Cyclone-II EP2C50 (50528 LEs, 86 (18x18) hardware multiplier)" and a high cost Altera solution "Startix-III EP3SL50(19000 ALMs, 216 (18x18) hardware multiplier)". The soft processor core "Nios-II/f" is used in these partitioning tests. It consumes up to 900 ALMs[75].

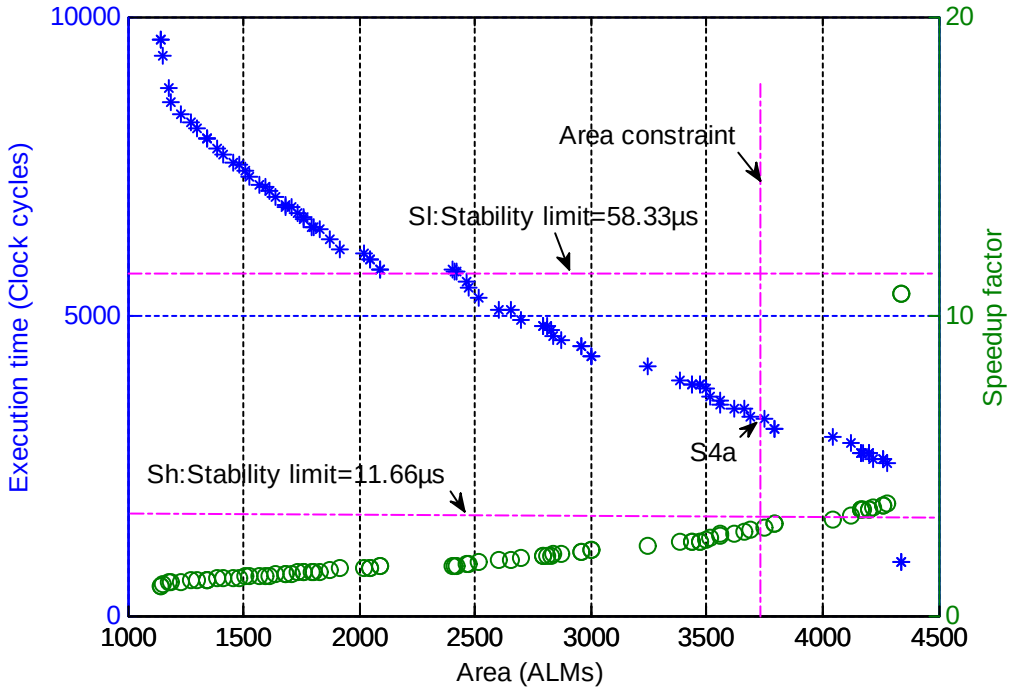


Figure 4.34: Partitioning solutions based on Stratix-III embedded Nios-II/f (CLK=100MHz).

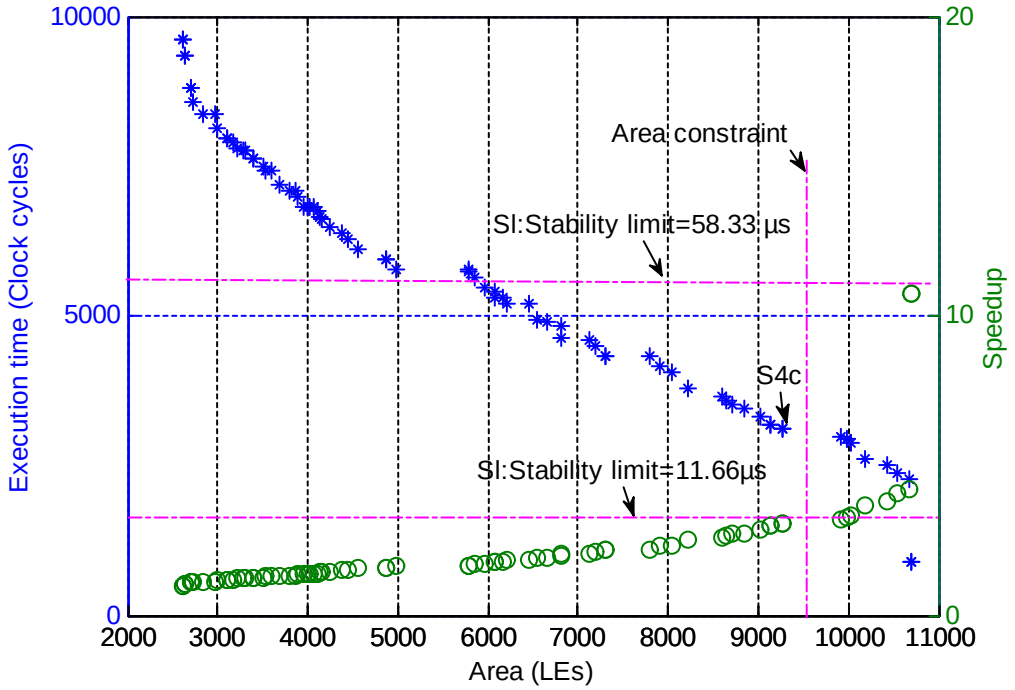


Figure 4.35: Partitioning solutions based on Cyclone-II embedded Nios-II/f (CLK=100MHz).

Some potential solutions are presented Table 4.16 and 4.17. For the two cases (Low/High digital platform), the scheduling diagrams of the best solutions according to the area and time constraints are presented Figure 4.36 and 4.37.

	Binary individual	*A* (*ALMs*)	*Tex* (*Clock cycle*)	*Memory* (*Kb*)	18*x*18 *HM*	*speedup* factor
S_{1a}	11111101011011111	3664	3440	72	41	2.79
S_{2a}	11111101011111011	3687	3332	71	41	2.88
S_{3a}	11111100111111011	3746	3285	71	41	2.92
S_{4a}	11111001011010011	3790	3124	68	43	3.07

Table 4.16: Some of the solutions based on Stratix-III

	Binary individual	*A* (*LEs*)	*Tex* (*Clock cycle*)	*Memory* (*Kb*)	18*x*18 *HM*	*speedup* factor
S_{1c}	11111000111111111	8848	3456	77	41	2.77
S_{2c}	11111101011111011	9022	3333	71	41	2.88
S_{3c}	11111101011111111	9131	3170	67	43	3
S_{4c}	11111100111111111	9278	3124	68	43	3.07

Table 4.17: Some of the solutions based on Cyclone-II

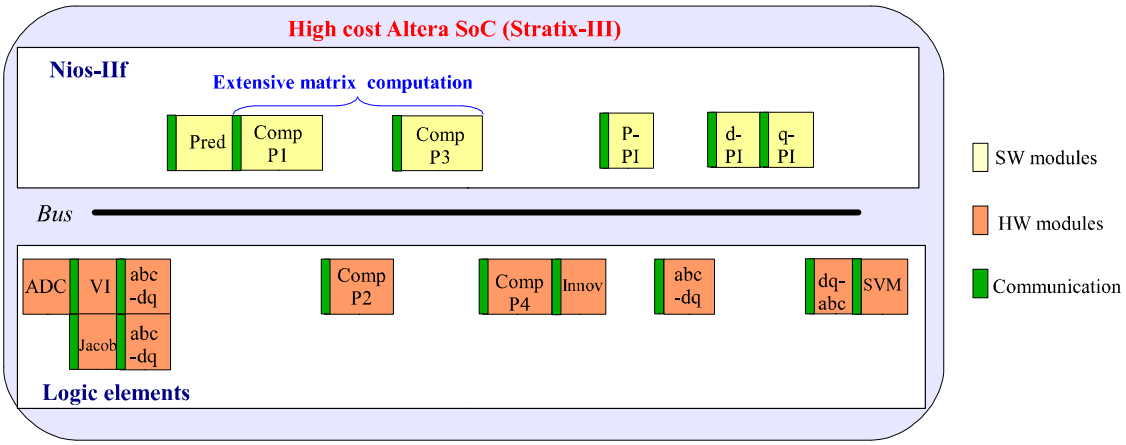


Figure 4.36: Scheduling diagram of the optimal solution "S4a" using Stratix-III embedded Nios-II/f.

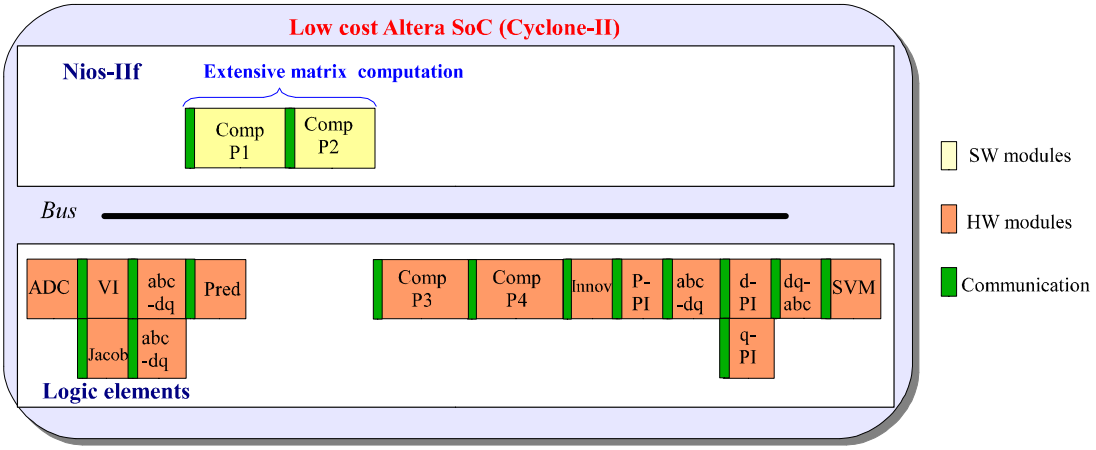


Figure 4.37: Scheduling diagram of the optimal solution "S4c" using Cyclone-II enbedded Nios-II/f.

Based on the presented timing diagrams, it can be noticed that there are always one or more extensive matrix computation modules executed in SW. These modules (Comp P1, Comp P2, Comp P3, and Comp P4), used in the EKF algorithm, are characterized by a high consumed resources in terms of area and hardware multipliers. As we deal with a resources-constrained scheduling, the NSGA-II aims to find simultaneously the optimal scheduling (minimizing the execution time) with regard to the area constraints. Placing these greedy modules in SW allows respecting the considered area constraints while not impacts too much the execution time.

Besides, the aircraft and aerospace applications need the use of Actel/Microsemi technology due to its suitable features (Non-volatile Flash technology, SEU immunity). But, comparing to other technologies such as Xilinx and Altera, Actel/Microsemi technology offers less logic resources. In addition, it does not provide hardware multiplier and hardware divider blocks. Thus, all multiplications are implemented using only logic elements which presents a significant increase of the area. This will implies also a long latency because of critical path of multiplication.

In the following, the tests were made using ProASIC3 (A3P1000) platform embedded the soft processor core "Cortex-M1". It is worth to be noted that during these tests, the operating clock frequency was fixed to 65 MHz which is the maximum allowable operating frequency for the Cortex-M1. This latter can be implemented in the FPGA matrix using 4435 tiles.

In this case, the execution time constraints of the two studied systems can be expressed as follow:

-For the first system (S_l): the stability constraint implies a maximum execution time equal to 53.33 μs which is equivalent to 3791 Clock cycles for a clock frequency of F_{clk}= 65 MHz.

-For the second system (S_h): the stability constraints is equal to 11.66 μs which is equivalent to 758 Clock cycles for a clock frequency of F_{clk}= 65 MHz.

The obtained results are presented in Figure 4.38 giving an idea of the possible solutions. We note that for the same design, the EKF sensorless speed control, the architecture consumes much more logic resources using Actel technology than those used in Xilinx and Altera technologies. For the two systems and using the ProAsic3 (A3P1000)

board, the proposed candidate solutions do not respect the functional constraints (maximum execution time). To implement this kind of complex algorithm, we need to use a board with higher density. The use of internal parallel architectures is also prefered to increase the performances in terms of execution time.

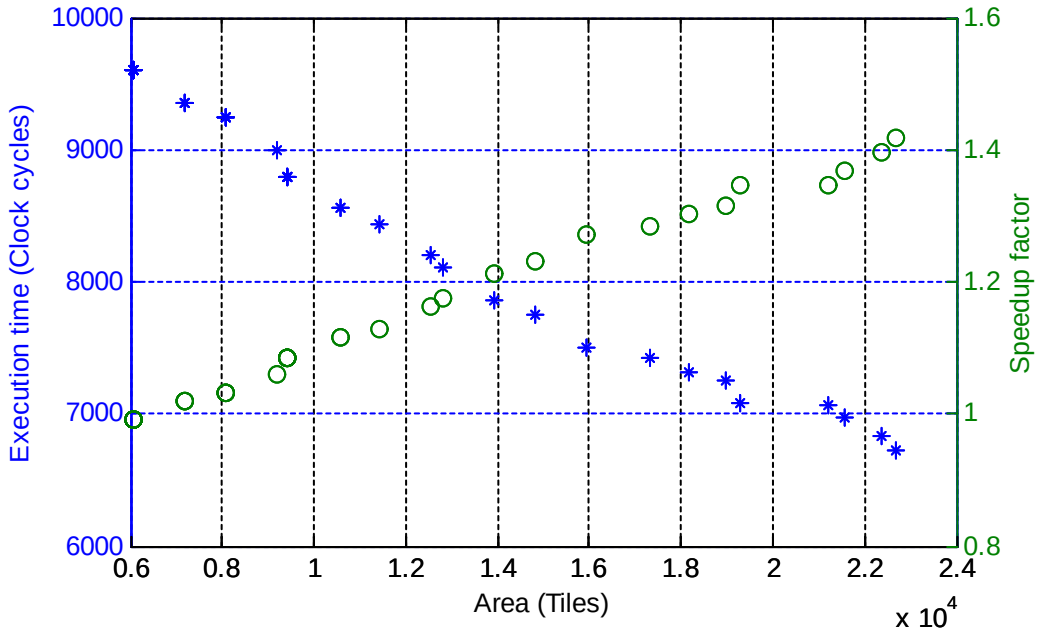


Figure 4.38: Partitioning solutions based on ProAsic3 with embedded Cortex-M1 (CLK=65MHz).

4.11 Conclusion

In this chapter, a HW-SW Co-design methodology for AC drives has been proposed. As benchmark, the EKF-based speed sensorless controller was used. To optimally implement the functional modules, a space exploration method was developed. It is composed by two main steps: the performance estimation of each functional module and the HW-SW optimized partitioning. The first step deals with the estimation of area, time, memory and hardware multiplier use for each functional module. Using the estimated performances, the second step aims to find the optimal partitioning in terms of area, execution time and memory of functional modules between HW and SW. This partitioning was performed depending on the considered constraints in terms of maximum allowable execution time and available FPGA resources. The execution time limit is based on the desired control bandwidth and the expected stability margin. To deal with this multi-objective optimization problem, the Non-dominated Sorting Genetic Algorithm (NSGA-II) was used. The partitioning tests were performed for two control systems: low and high sampling rate control systems. Besides, the proposed HW-SW partitioning algorithm was tested on Altera and Actel technologies (Low/High cost).

In future works, the experimental validation of the optimal solutions for each technology will be performed. Making tests for high temperature conditions using Actel technology is also important especially for aircraft applications.

On the other hand, the automation of the whole Co-design flow (the association of control performances estimation and the HW-SW partitioning) is a promising orientation, which could allow a significant reduction of the development time of control applications.

Chapter 5

Real-Time Operating System for HW-SW controllers: Application to the case of AC drives

Chapter 5

Real-Time Operating System for HW-SW controllers: Application to the case of AC drives

5.1 Introduction

In the previous chapter, author has demonstrated the interest of using heterogeneous approaches to deal with the growing complexity of Embedded Control Systems (ECSs). This trend is all the more relevant that the number of tasks implemented in an embedded controller is always increasing (control tasks, communications tasks, health monitoring tasks...). Thus, the use of HW-SW Co-design methodology provides an optimal HW-SW partitioning of control modules in terms of area, time performances and memory use.

But, if the HW-SW approach is a proper answer to the controller architecture design, another important issue remains to be solved to cope with this ever increasing complexity of algorithms and their corresponding architectures: it concerns the management of the designed embedded controllers.

In this context, the System-on-Chip architecture of the embedded controller can take advantages to be managed by an efficient Real-Time Operating System (RTOS) [Ref]. Indeed, RTOS is needed to ensure multi-task scheduling and services such as synchronization, communications, memory management...

But the chosen RTOS must be portable and must adapt its scalability from one platform to another, thus ensuring a maximum of flexibility to the designer. Besides the use of RTOS may add significant time overheads that can impact the execution time of the controller and as a consequence it limits the expected final performances of the controlled system. This point is critical in the case of AC drives where the dynamics are fast.

To overcome this issue, an interesting solution is the development of an hardware Real-Time Unit (RTU) in conjunction to the RTOS which provides a significant acceleration of the RTOS services and which reduces its memory use [116]-[119].

In this chapter, author deals with the design and the validation of a current controller for AC drive using a deterministic RTOS associated to hardware RTU. Firstly, author presents the related works, the problem statement and the main solutions to solve it. After that, a brief description of the chosen RTOS (MicroC/OS-II) functionalities and services are given. Thirdly, the designed RTU is presented and its time/area performances

are given. RTU validation was made on a Microblaze processor platform. Finally, the RTOS and its associated RTU are tested experimentally in the context of the current control of a synchronous drive.

5.2 Related works

Designing hardware RTU for RTOS was motivated mainly by the need of more deterministic embedded systems. One of the first works dealing with this problem, called FASTCHARD, was based on one processor associated to the RTU [120]-[121]. The focus of this work was the design and the evaluation of the RTU deterministic performances. The modifications of the RTOS due to the use of a RTU and its scalability were also studied. In [122], authors describe a hardware/software Co-design of a multithreaded RTOS kernel on a Xilinx Virtex IIPro. Tests showed tighter bounds on scheduling precision and significant jitter reduction when compared to traditionally implemented RTOS kernels. Other interesting works focused on the integration of the RTU within multiprocessor systems. Several functionalities such as scheduling, time management, semaphores and event flags were added. Among these works, the Scalable Architecture for Real- Time Applications (SARA) is a good example. It is a scalable unit which allows an easy transition from single to multiple processors. The first SARA system was used in a vision system connected to an industrial robot (ABB Robot) [123]-[125].

In [126], authors show a performance comparison among three RTOSs: the RTU hardware RTOS, the pure software Atalanta RTOS and a hardware/software RTOS composed of part of Atalanta interfaced to the System-on-a-Chip Lock Cache (SoCLC) hardware. The considered application integrates three-processor system running thirty tasks. RTU and the same system implemented within SoCLC showed respectively 36% and 19% overall speedup gain compared to the pure software RTOS implementation. Another performance comparison between pure RTOS and RTU associated to the RTOS was presented in [127]. Acceleration of about 2.6 times was achieved using the RTU. The main advantages of the RTU in this case is that the time overhead doesn't increase when adding new tasks.

The influence of RTU on the system energy consumption was also studied [128]. The obtained results demonstrate that the power consumption is independent of RTU state (running or idle). These results were provided using an energy characterization of the RTU. It was also noted that the power optimization could be performed using gated clocking techniques.

All the indicated works have shown the benefits of using a RTU over standard software RTOS. However, the main disadvantage of this approach is the use of a single bus to exchange data between the RTU and the processor. This can influence the determinism of RTOS since the RTU treatment depends on it. A lack of benchmarking and experimental validations were also noted especially in the field of stringent real time applications.

5.3 Problem statement and motivation

The introduction of the RTOS in the domain of ECSs has allowed designing more sophisticated and flexible control systems. With the adoption of multi-tasking approach, the new design presents a modularized solution which increases the code reuse, the portability and the scalability. It provides also an unrestricted access to shared memory and

CPU resources. However, the control systems are considered as time critical applications. They are characterized by severe real time constraints since they have to be executed periodically at the sampling rate. So, the scheduling of such applications is crucial when using RTOS.

As depicted Figure 5.1, the time overhead caused by the RTOS services can violate the timing constraints of the application. The introduced time delay due to RTOS in the closed loop of a control system can lead to a significant degradation of the control performances. So, the goal is to execute the control algorithm in a synchronous manner at the sampling rate as shown in relation 5.1. This means that the sum of all the control tasks associated to the RTOS services must not exceed the sampling period "T_s".

$$T_{ex\_with\_RTOS} < T_s \tag{5.1}$$

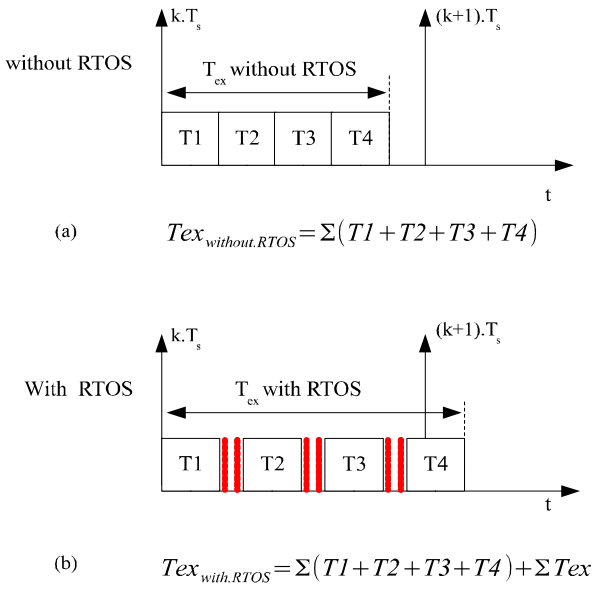


Figure 5.1: RTOS time overhead effect.

To cope with this problem, the chosen approach targets to design a deterministic RTOS associated to its corresponding hardware RTU. Thus, some RTOS services are moved from SW to HW. The designed architecture is generic, modular, flexible based on scalable modules. This will relief the processor and will increase the timing performances since this hardware accelerator will be executed in parallel with the CPU. Furthermore, the memory footprint will decrease which presents a convenient solution especially for small ECSs limited in terms of memory resources.

Figure 5.2 presents a solution based on the RTU for mono-processor. This approach presents great benefits for AC drives in terms of performances[130]. The RTU can enhance the exploitation of the FPGA reconfigurability. Indeed, the acceleration of the RTOS services allows fast response to external events and consequently allows a faster reconfiguration time. In this case, an interesting configuration using partial run-time hardware reconfiguration allows the implementation of several controllers without having to realize them all on the FPGA concurrently [31], [32]. This kind of applications is of

great importance especially in the case of faulty conditions where a new control strategy must be used.

In the present work, the communication between the CPU and the RTU is ensured by the FSL bus from Xilinx Company. But, the generic RTU module can be associated to other communication buses such as Avalon from Altera Company or AMBA from ARM Company.

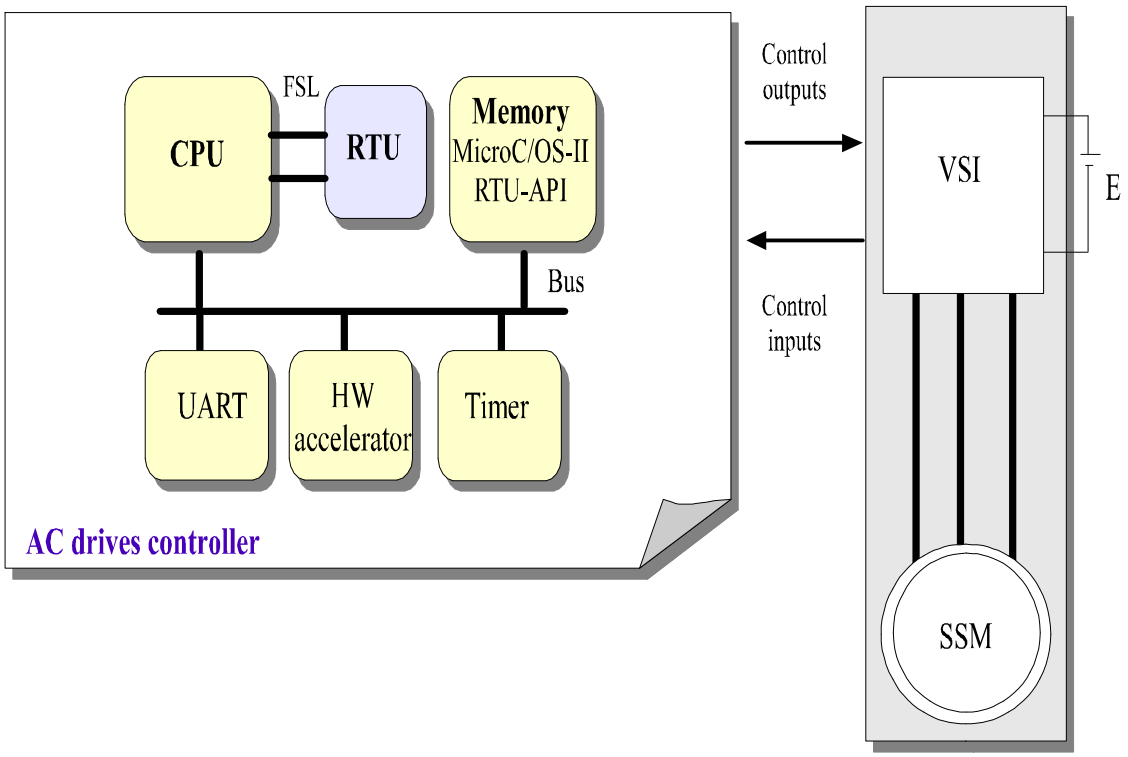


Figure 5.2 : AC drive controller based on the RTU.

5.4 Micrium: MicroC/OS-II

In this work, the chosen RTOS was the MicroC/OS-II Kernel. It is a simplified real time kernel developed by Jean Labrosse. It was certified by rigorous standards, such as RTCA DO-178B, which refers to the use of critical systems in aircraft domain. The features of MicroC/OS-II can be summarized as follows [29]:

- **Multitasking kernel** : It can manage up to 64 tasks with only 56 user tasks (unlimited task number in the new version of MicroC/OS-III). Each of the tasks has its own priority which means that it does not support round-robin scheduling. This latter is supported by the MicroC/OS-III.

- **Preemptive** : It means that the kernel executes the highest priority task that is ready.
- **RTOS services** : It provides communication services (such message mailbox, message queues..), synchronization services (semaphores, flags...), time management and memory management. Each of these services can be enabled or disabled using configuration parameters. They are written based on a systematic way and can be parameterized via compiler directives.
- **Free academic license**: The license of MicroC/OS-II services is free for academic purposes.
- **Portable** : The majority of the MicroC/OS-II code was written using highly portable ANSI C associated with only few line of assembly code. Therefore, MicroC/OS-II was already ported to a large panel of processors.
- **Code size** : It consists in a small RTOS since it takes less than 60 Kb. Its small memory footprint makes it suitable for resource-constrained embedded systems.

In the following, some important concepts of the real-time Kernel are introduced. To begin and by definition, a task is a program that considers it has all the CPU capacity for its own purpose. For each task, the designer can attribute a priority, CPU registers and a stack area. The creation and deleting of tasks is ensured respectively by *OSTaskcreate()* and *OSTaskdel()* functions.

Figure 5.3 explains the multi-task management used by the MicroC/OS-II. To save the status, the stack and the priority of a task, a data structure (Task Control Block (TCB)) is used. In the case of context switching or interruption, the TCB of the corresponding task will be saved on CPU registers to be restored once these services are achieved.

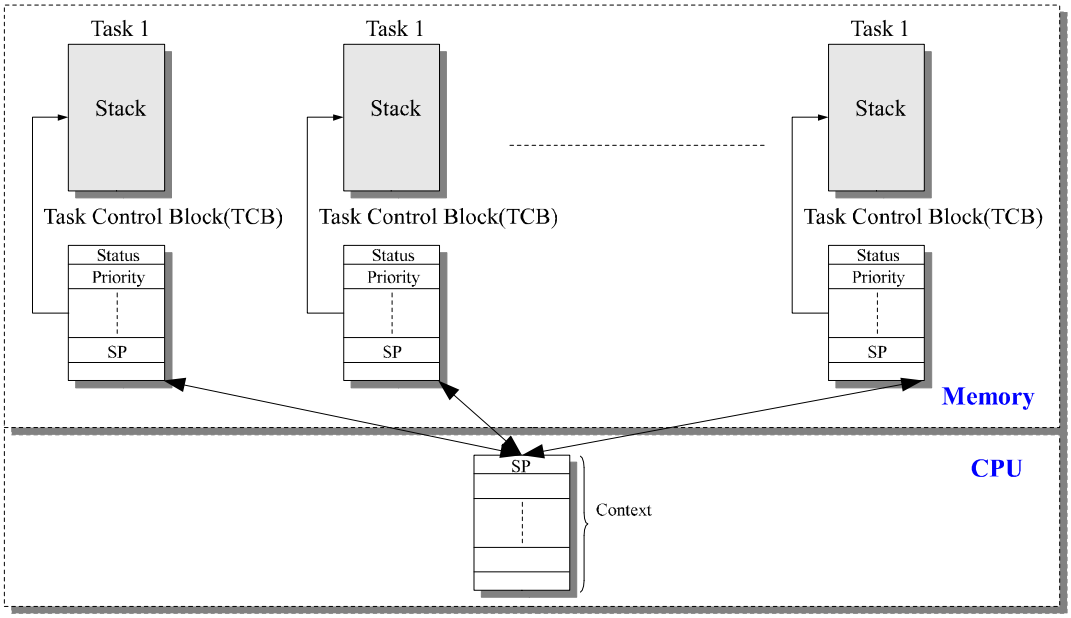


Figure 5.3: The multi-task management.

After creating tasks, the multi-tasking can be performed using *OSStart()* function. As shown in Figure 5.4, MicroC-OS/II uses the notion of state for each task, which can be either running (running in the CPU), or waiting for an event (waiting), or ready (it is ready to be executed but its priority level is less than the running task), dormant (task saved in memory but it was not called by the Kernel), or interrupted when the CPU executes an interrupt service. Some functions allowing the switching from a state to another are also indicated.

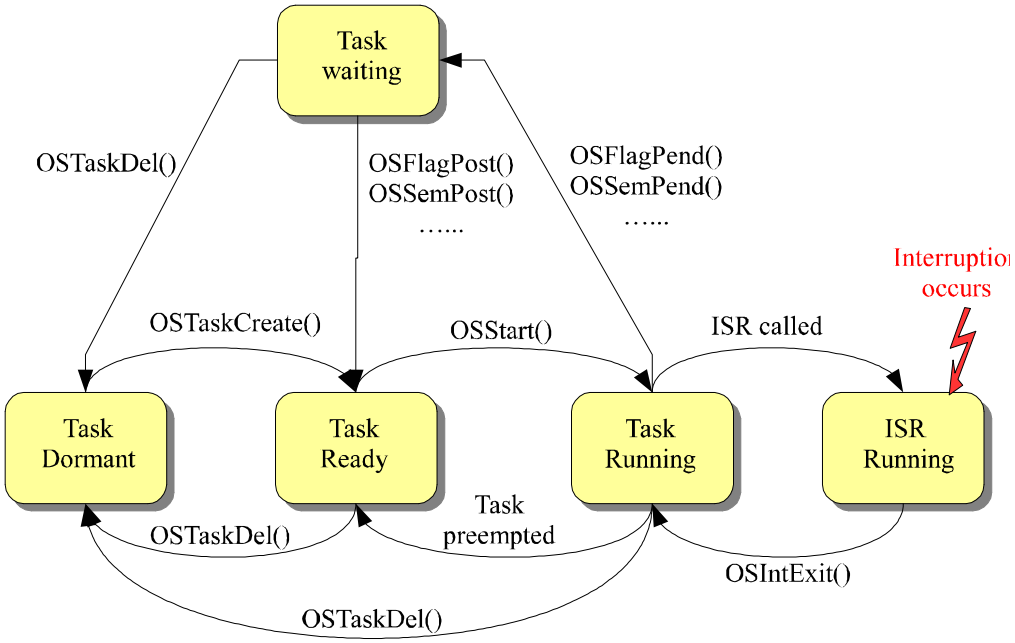


Figure 5.4: Task states.

The scheduling of tasks is provided by the scheduler. The responsibility of the scheduler is to find the next task to be run. This depends mainly on the task priority fixed by the designer when creating the tasks. MicroOS-II is a preemptive Kernel. The principle is when a higher priority task is activated, the task is preempted, and the activated task will take the control of the CPU. This is an important feature when system responsiveness is essential. To perform this multi-tasking process, the RTOS needs a periodic clock signal, called Tick. It is generated by calling the OSTimeTick() functions from Tick ISR. This later can be activated using a Timer.

5.5 Porting MicroC/OS-II

The development of an ECS using RTOS begins by the porting of MicroC/OS-II. This requires C and assembler designer skills because the RTOS porting includes independent and specific processor files. Figure 5.5 presents the different layers to handle.

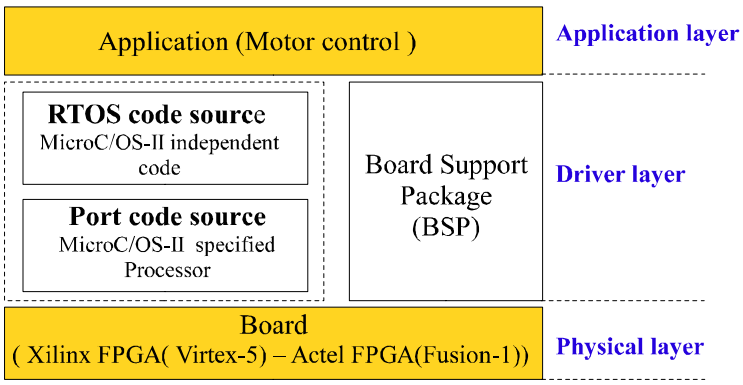


Figure 5.5: The MicroC/OS-II Porting.

- *Application layer :* It is composed by configuration functions and the application to be executed *(app.c, app_vect.c, app_cfg.h, includes.h, os_cfg.h)*. In our case, it consists on motor control application.

- Driver Layer: it includes three part:

  -*Board Support Package (BSP.c, BSP.h):* It is a set of drivers permitting the generic use of input/output, memory, timer and all the used peripherals.

  -*Port code source :* It consists on portions of code written in assembly language and depends on the used processor. It includes: os cpu.h, os cpu c.c, os cpu a.s files. These parts ensure the management of critical section. They execute the mult-tasking process (*OSStartHighRdy()*), the context switching (*OSCtxSw()*), the context switching in the case of interruption (*OSIntCtxSw()*) and the tick interruption (*OSTickISR()*).

  -*RTOS code source (os-_core.c, os_flag.c, os_mbox.c, os_mem.c, os_mutex.c, os_q.c, os_sem.c, os_task.c,os_time.c, os_tmr.c, ucos_ii.h) :* It includes all the independent RTOS code sources. They are written using ANSI C language.

- Physical layer *:* The port of MicroC/OS-II was made using two boards : The Fusion-1 from Actel/Microsemi board and the Virtex-5 from Xilinx board. The first one includes the soft processor "Cortex-M1" and the second one includes "Microblaze". Memory, timer, interrupt manager are also configured.

During this thesis, the porting was made using the two indicated boards and then this porting has been compiled, linked and tested experimentally. In order to evaluate the time taken by the RTOS services, some measurements were performed based on the Cortex-M1. The obtained results are presented in the Appendix-E. However, the RTU development and the corresponding experimental tests were made only using the Xilinx FPGA. The same tests must be made using Actel FPGA with the adaptation of RTU and CPU communication interface. The FSL bus can be replaced in this case by the AMBA one.

5.6 Description of the RTU

The proposed RTU was implemented on the Xilinx FPGA (Virtex-5, ML506). This FPGA allows the configuration of the soft processor core "Microblaze". Some MicroC/OS-II functionalities were mapped in hardware providing the parallel execution and the acceleration of the RTOS[132]. It consists on the RTU. As shown in Figure 5.6, the RTU includes the scheduler, the time manager and the semaphore modules. The communication between the Microblaze and the RTU modules is ensured by the FSL buses (FSL 0, FSL 1), the FSL interface and the decoder. One of our main concerns was the preservation of modularity principle and the scalability of the developed RTU. Therefore, these modules were designed having in mind the possibility to make extensions and reconfigurations or to adapted it to other communication buses.

The RTU was used through a specific Application Program Interface (API). It is set of functions and mapped memories that perform the message transactions between Microblaze and the RTU. As an example, we indicate the following functions: HW-SemCreate(), HW-SemPost(), HW-SemPend()....

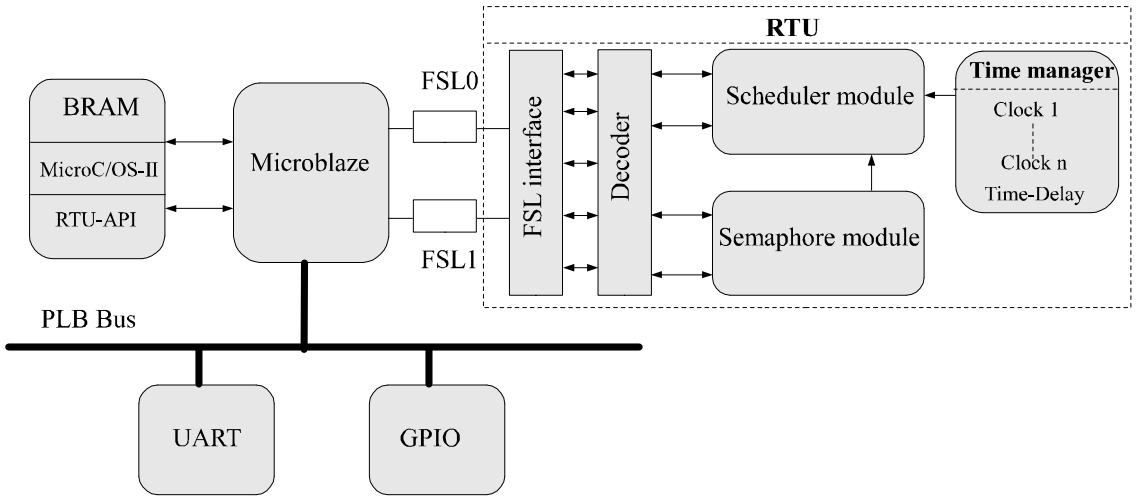


Figure 5.6: RTU architecture.

5.6.1 The FSL interface and the decoder

The FSL interface and the decoder modules ensure the synchronization and communication between the Microblaze and the RTU. The control of the FSL bus is provided using a FSM. The state transitions depend on FSL control signals and the scheduler signals. To decode the Microblaze-RTU exchanged informations, an instruction decoder was developed. It determines the primitives which are called by the RTOS and transmits informations to the scheduler and semaphore modules. The communication format for FSL-Input and FSL-Output buses is shown respectively in Tables 5.1 and 5.2.

FSL-Inputs		
Name	Width	Function
ID	8	Indicates the used primitive
Data	16	Arguments of the used primitive

Table 5.1: FSL-Input format

FSL-Outputs		
Name	Width	Function
Status	8	Indicates the context switching or the status of the semaphore
Data	16	task priority

Table 5.2: FSL-Output format

5.6.2 Scheduler and Time manager Modules

The main goal of the RTOS is the task scheduling with regard to occurred events. Indeed, the scheduler is executed when the task status is changed or an interruption has occurred. Therefore, the implementation of this module in hardware can significantly improve the RTOS performances.

Having in mind this objective, the proposed hardware scheduler module includes the scheduling algorithm, while the context switching is done in software part. The proposed time manager replaces the time delay services and the system tick. This latter is used as heartbeat of the pure software RTOS and generally set to 1 millisecond. For a stringent real time application, reactivity is crucial. If we set a short RTOS tick , the processor will be reactive but it will spend time in switching between tasks, and consequently the time devoted to running tasks will be reduced. The proposed time manager can set the period of the clock signal. This allows to have a reactive RTOS without time overhead. The time delay service allows the calling task to delay itself for a user defined time. As shown in Figure 5.7, the scheduler module is composed by a FSM and a Ready Tasks Queue (RTQ). It is designed to support preemption. From the decoder, the scheduler can determine the required primitives by using the "ID". The received "Data" will be used as next step to realize the requested function (Create task, delete task, delay task). On the other hand, the RTQ is used to determine the highest priority task in the application according to the scheduler policy: Priority (PR). When a task is inserted and depending on its PR, the queue automatically re-orders in only one clock cycle. In the case of task priority exchanges or interruptions, a context switching request is sent to the Microblaze to perform the data saving of the suspended task. The FSM ensures the synchronization between the RTQ signals and the decoder control signals. The Semaphore control signals are also taken into account in the case of suspension of a task.

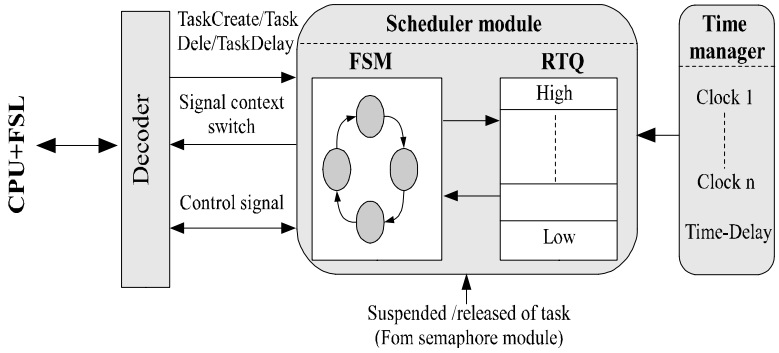


Figure 5.7: The architecture of the HW scheduler module and the clock manager.

5.6.3 Semaphore Manager

For resource access synchronization purpose, the RTU integrates a semaphore module shown in Figure 5.8. Like the scheduler module, the semaphore module uses a FSM and a Wait Queue (WQ). The proposed module allows the management of several semaphores. The FSM waits for control signals from Microblaze, indicating if a semaphore is locked or unlocked. After that, the semaphore module can select a semaphore or not. Each of these semaphores has its own WQ used to save the priority of task. Additionally, a control signal is sent to the scheduler module to prevent it when a task is suspended on a semaphore or a semaphore has released a task.

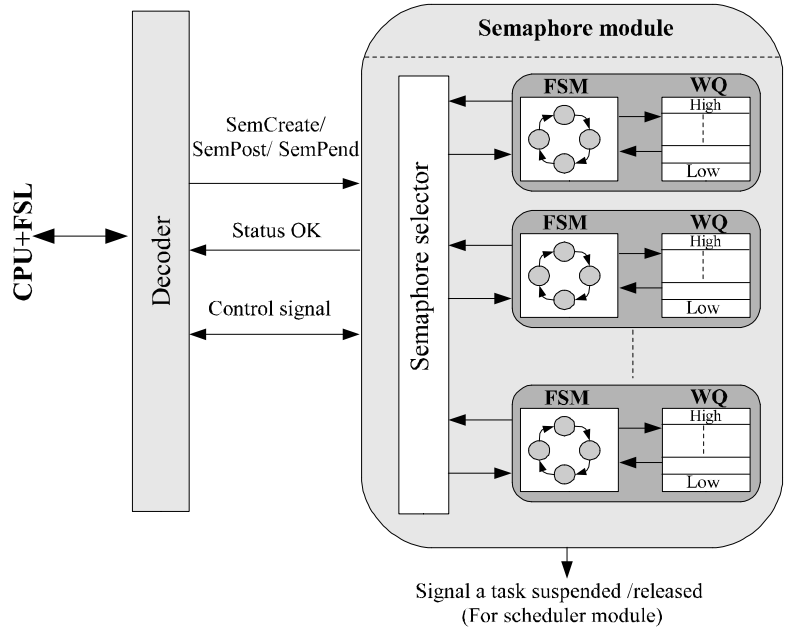


Figure 5.8: Architecture of the HW semaphore module.

5.6.4 Area and time performances

Tables 5.3 and 5.4 present respectively the area and time performances of the developed RTU. These results were performed using the Xilinx "Virtex-5 (XC5VSX50T)" and the synthesis process was performed for 16 tasks and 4 semaphores. As said before, this RTU configuration can easily be extended to integrate more tasks and semaphores. The developed RTU takes only 4.3% of the available FPGA resources which is very few compared to the acceleration benefits. The operating frequency of the Microblaze and the RTU was set to 100 MHz.

	FFs	**6-Bit LUTs**
Scheduler Module (16 tasks)	290	692
Semaphore Module (4 semaphores)	321	685
FSL interface/ Decoder	9	4
Total RTU	620	1381

Table 5.3: RTU Consumed resources

Primitives	**Number of clock cycles**
HW-Initialization	2
HW-TaskCreate	2
HW-TaskDelete	2
HW-SemCreate	2
HW-SemPost	6
HW-SemPend	6

Table 5.4: RTU Timing Evaluation

Table 5.5 presents the execution time of the developed RTU modules based on two configuration: external SRAM and BRAM (Block RAM inside the FPGA). In the case of external SRAM configuration, the comparison, between the service execution time using RTOS and using RTOS associated to RTU, shows a 50% speed-up for the second case. This is important especially in the case of complex and greedy applications that need the use of larger SRAM.

In the case of BRAM configuration, the advantages of the RTU is less visible in terms of execution time of services. But, the RTU advantages are still ensured by two manners: the elimination of the tick overhead and the reduction of memory use. Indeed, the proposed time manager used by the scheduler is activated every clock cycle which ensure high RTOS reactivity. This is not the case of classical RTOS sequenced by a tick varying between 1 ms to 10 ms.

As a conclusion, we can note that the proposed RTU offers more determinism for the studied application using MicroC/OS-II. This presents a good basis to test more complex algorithms and to prove the interest of the proposed approach for controlling power electronic systems.

	Pure software RTOS	RTOS associated to the RTU	
	SRAM	SRAM	BRAM
Context Switching	23.7 μs	23.7 μs	2.1 μs
SemCreate	26.92 μs	8.8 μs	3.59 μs
SemPend	25.99 μs	12,3 μs	2.2 μs
SemPost	23.11 μs	11,7 μs	2.4 μs
TaskCreate	85.23 μs	77.86 μs	3.23 μs

Table 5.5: Time overhead of the RTOS services

5.7 Benchmark: motor control

The synoptic of the implemented control system which includes the tested RTOS is given Figure 5.9. It consists in the same experimental set-up presented in the chapter 4. The control architecture was split in two parts: HW and SW. The HW part integrates the ADC, position and DAC interfaces. The position and ADC interfaces provide respectively the information of rotor position from the encoder and the stator current A/D conversion. The DAC module is used to send digital information to an oscilloscope. The SW part includes the reading of the acquisition registers, the dq-abc transformation and the 3-phase ON/OFF regulators. The communication between HW and SW modules is ensured using the PLB bus and the IPIF (IP Interface). This latter is a basic infrastructure to connect the PLB to HW module. It can be automatically generated using EDK tool. The data exchanges were performed using read/write command in specific software registers. The synchronization of the control parts is provided by MicroC/OS-II associated to the RTU.

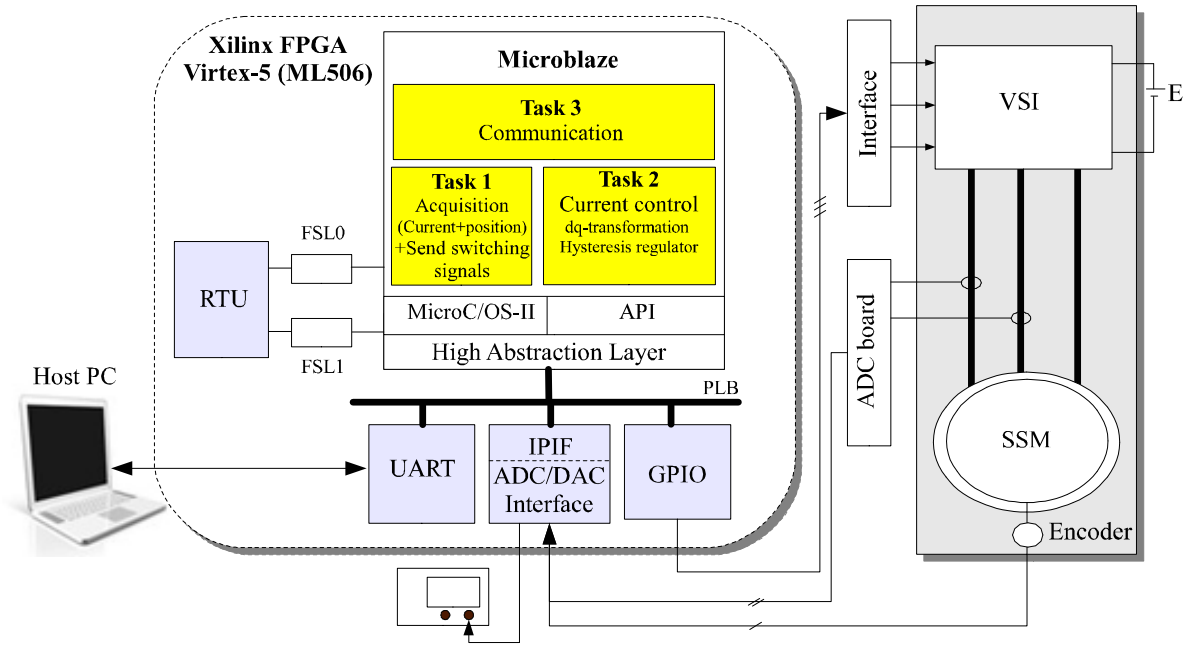


Figure 5.9: The synoptic of the control motor based on the RTU.

From the Microblaze side, the program code was divided into three tasks defined as below:

- Task 1 (Reading of acquisition data and sending the switching controls): This task provides the reading of the stator current and the rotor position data. These informations are provided by the conversion interface ADC interface performed in hardware. From other side, it send the switching signals to the Voltage Source Inverter (VSI) using the GPIO (General Purpose Input/Output).
- Task 2 (Current controller): This task includes the dq-abc transformation and the ON/OFF regulator. The hysteresis bandwidth was set to 0 A.
- Task 3 (Communication interface): It is considered as a background task. The objective is to send and receive the required information to the supervisor connected to the Host PC. This is performed by the UART module.

MicroC/OS-II is based on a fixed-priority scheduling. So, each task of the considered application has a fixed static priority. The highest priority was affected to the task 1. The second priority was attributed to the task 2. The lowest priority was affected to the communication task. All these tasks are sequenced every sampling period "T_s" equals to 50 μs.

The timing diagram of the proposed controller is presented Figure 5.10. It shows the task scheduling, number of clock cycles of each task and RTOS services (context switching (Ctx), SemPend, SemPost, Time_delay). These latter ensure the synchronization between the three tasks.

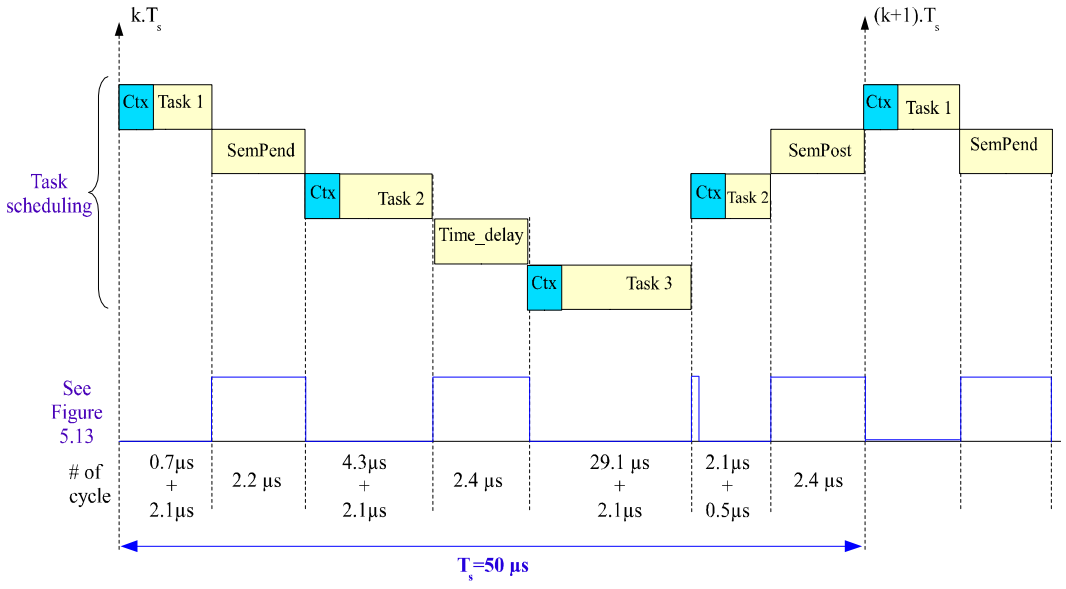


Figure 5.10: Timing diagram of the current controller using RTOS (BRAM configuration).

Figures 5.11 and 5.12 present the obtained experimental results of the ON/OFF current controller. In order to compare the control performances, the tests were performed using two memory configurations to save the program code: BRAM and external SRAM.

The minimum sampling period "T_s" of the current regulator is equal to 168 μs when using the external SRAM configuration and only 50 μs using the BRAM. As a consequence, the experimental current THD is 11,3 % for the BRAM configuration and 21,7% for the external SRAM.

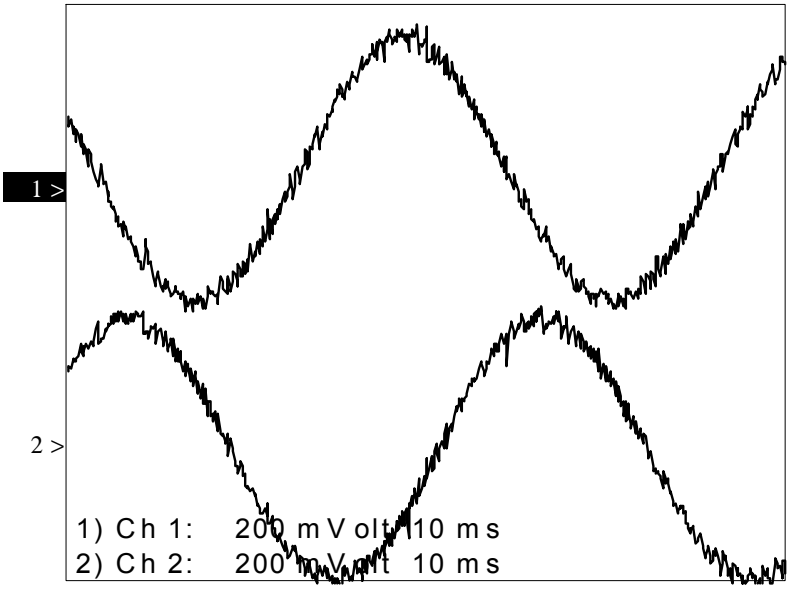


Figure 5.11: Regulated stator currents using the BRAM configuration for (I^*_{sd}=0A, I^*_{sq}=1.5A, Bw*=0A) with 250mV corresponds to 1A.

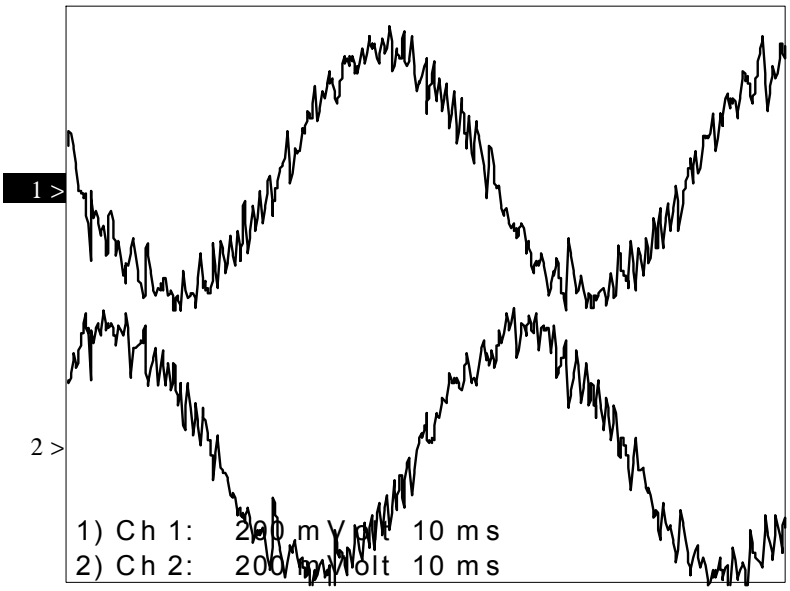


Figure 5.11: Regulated stator currents using the external SRAM configuration (I^*_{sd}=0A, I^*_{sq}=1.5A, Bw*=0A) with 250mV corresponds to 1A.

Figure 5.13 presents the measured latency time of HW-SemPost, HW-SemPend, communication process and current controller. We note that these measurements were repeated many times to ensure the correctness of the measured values.

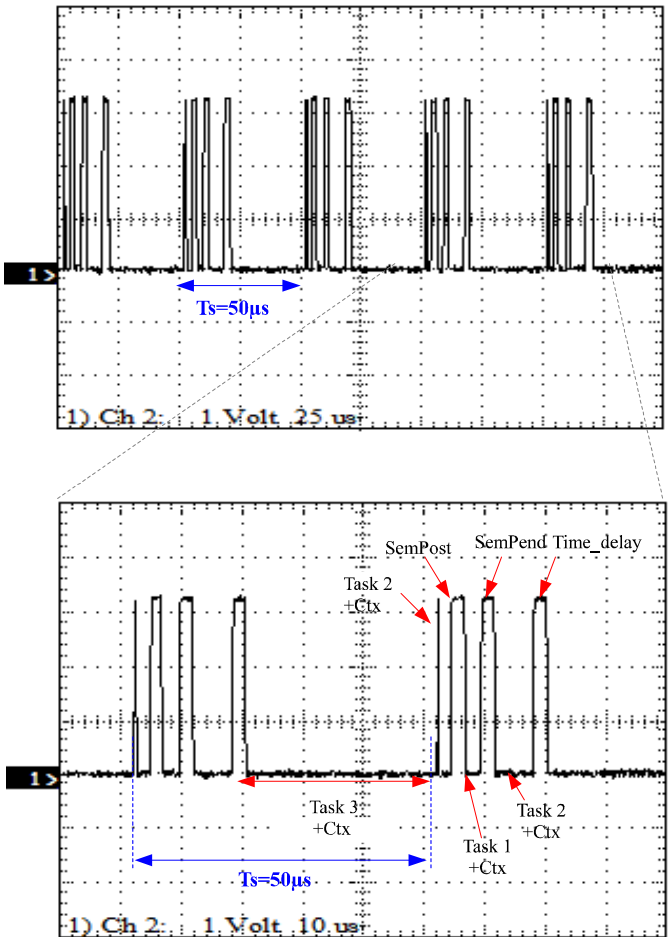


Figure 5.13: Timing chart of the current controller based on the RTOS and RTU (BRAM configuration).

5.8 Conclusion

In this chapter, a HW-SW RTOS for AC drives has been introduced. To have a more deterministic RTOS, a hardware Real Time Unit (RTU) was developed using VHDL and was associated to the RTOS. It allows the reduction of the time overhead related to the RTOS services and the decrease of memory footprint. In this approach, only the scheduler and the semaphore services were implemented in hardware. The proposed design was carried out using a Xilinx Virtex-5 (XC5VSX50T) and the communication between the "Microblaze" and the RTU was provided using the FSL bus. The evaluation of the RTU in terms of area and time performances was also discussed. It demonstrates that the proposed design takes only 4.3% of the FPGA available resources which is very few compared to the obtained acceleration benefits. To show the interest of such approach for AC drive applications, an experimental validation based on current controller was carried out using a laboratory set-up. The obtained results give proof of the interest of the proposed approach.

Chapter 6

General conclusion

This thesis has dealt with the contribution of FPGA System-on-Chip (SoC) controllers for embedded AC drives applications. More specifically, author has been interested to embedded control systems for aircraft applications. These latter are characterized by severe technical issues.

In this context and in the frame of SEFORA project (Smart MEA For Operations in Rough Atmospheres), the author has started by the description of a rigorous design and validation methodology for FPGA-based digital controller of a high temperature synchronous drive. As first stage of validation, the proposed design method was tested on the ProAsic^{plus} board from Actel/Microsemi. The impact of the temperature on the operating frequency was also studied.

Besides, a HW-SW Co-design methodology for electrical AC drive has been proposed. This methodology aimed to link the assessment of control performances and the HW-SW partitioning of control modules, at early stage of the development. It is ranging from the early system specifications to the final experimental validation of the control application. One of the main important steps of this method is the HW/SW partitioning. The goal was to find an optimal combination between modules to be implemented in software and those to be implemented in hardware. The Non-Dominated Sorting genetic Algorithm (NSGA-II) was used to find the Pareto-Front of optimal solutions. As benchmark, the speed sensorless controller using the Extended Kalman Filter (EKF) was used.

From another perspective, the management of SoC-based embedded controller can take more advantages using an efficient Real-Time Operating System (RTOS). This RTOS is needed to ensure multi-task scheduling of the controller. But, the use of an RTOS may add significant time overheads that can impact the execution time of the controller. To accelerate the treatment of this operating system, a Real-Time Unit (RTU) was developed in VHDL and associated to the RTOS. It consists in a hardware operating unit that moves the scheduling and communication process from software RTOS to hardware. Experimental results give proof of the good performances of the proposed approach.

Future work

The direct continuation of this work can be made exploring the following points:

- The experimental validation of optimal solutions proposed by the NSGA-II algorithm.
- The evaluation of partitioning algorithm results using a more regular grain regarding the functional modules.

- The comparison of the optimization results of the NSGA-II algorithm to other optimization methods such as the tabu or the ILP methods.
- The validation of HW-SW Co-design methodology with other demanding algorithms such as the predictive control.
- The integration of power consumption characterization for each functional module in the HW-SW Co-design methodology flow.
- The moving of other RTOS services to hardware such as priority inversion service, mutex services, mailbox services...
- The implementation of another scheduling policy such as round-robin policy.

The Mid term perspectives are :

- The development of automatic high-level HW-SW Co-design methodology.
- The analysis of multi-processors interest for control applications.
- The use of RTOS for partial reconfiguration control applications.

Appendix-A

In the following, we deal with the detailed features of each functional module (PI regulator, transformations...). It consists in the consumed resources in terms of Flip-Flop and Look-Up-Table and hardware multiplier blocks. The execution time is also provided. It is given in system clock cycle numbers. T_{hp} is the execution time of a parallel architecture and t_{hf} is the execution time of factorized architecture. Most of the functional modules are declined in two versions (parallel and factorized). In Table 6.1, the characteristics of each functional module are presented. Table 6.2 is devoted to the EKF modules.

	Modules	Operators	Parallel		Factorized	
Current control	**2 ∗ PI regulator**	$20-Bit\ mult$ $20-Bit\ add$ $20-Bit\ sub$ $20-Bit\ reg$ $20-Bit\ mux$	4 4 2 12 0	128 LUT 240FF 8HM $T_{hp}=2*10$	2 4 2 12 2	160LUT 240FF 4HM $T_{hf}=2*21$
	dq_abc	$13-Bit\ mult$ $13-Bit\ add$ $13-Bit\ sub$ $13-Bit\ mux$ $13-Bit\ reg$ $13-Bit\sin$	4 3 1 0 5 1	191LUT 65FF 8 HM $T_{hp}=9$	1 3 1 3 10 1	230LUT 155FF 1HM $T_{hf}=25$
	abc_dq	$20-Bit\ mult$ $14-Bit\ ad$ $13-Bit\ reg$ $20-Bit\ reg$ $14-Bit\ mux$ $13-Bit\sin$	6 2 4 2 0 1	165LUT 92FF 12HM $T_{hp}=10$	1 2 9 2 3 1	207LUT 184FF 2HM $T_{hf}=27$
	SVM	$13-Bit\ comp$ $13-Bit\ count$ $13-Bit\ sat$ $13-Bit\ reg$ $13-Bit\ mux$ $13-Bit\ add$ $1-Bit\ table$	1 1 3 10 1 3 1	390LUT 143FF 0HM $T_h=4$		
Speed control	**P − PI regulator**	$20-Bit\ mult$ $20-Bit\ add$ $20-Bit\ sub$ $20-Bit\ reg$ $20-Bit\ mux$	3 2 2 8 0	84LUT 160FF 6HM $T_{hp}=12$	1 2 3 8 2	124LUT 160FF 2HM $T_{hf}=21$

Table 6.1: Consumed resources of the control functional modules

	Modules	Operators	Parallel		Factorized	
EKF Estimator	**2x abc_dq**	$20-Bit\ mult$	12	$330LUT$	2	$414LUT$
		$14-Bit\ add$	4	$184FF$	4	$314FF$
		$13-Bit\ reg$	8	$24HM$	18	$4HM$
		$20-Bit\ reg$	4	$T_{hp}=2x10$	4	$T_{hf}=2x27$
		$14-Bit\ mux$	0		6	
		$13-Bit$ sin	2		2	
	Innovation	$22-Bit\ mult$	8	$236LUT$	1	$208LUT$
		$22-Bit\ add$	8	$396\ \ FF$	5	$220\ FF$
		$22-Bit\ reg$	18	$16HM$	10	$2\ HM$
		$22-Bit\ mux$	0	$T_{hp}=8$	2	$T_{hf}=25$
		$22-Bit\ sub$	2		2	
	Pr**ediction**	$22-Bit\ mult$	10	$144LUT$	1	$314LUT$
		$22-Bit\ add$	6	$352FF$	3	$198FF$
		$22-Bit\ reg$	16	$20HM$	9	$2HM$
		$22-Bit\ mux$	0	$T_{hp}=8$	11	$T_{hf}=23$
	Jacobian matrix	$22-Bit\ mult$	4	$24LUT$	1	$68LUT$
		$22-Bit\ add$	1	$110FF$	1	$110FF$
		$22-Bit\ reg$	5	$8HM$	5	$2HM$
		$22-Bit\ mux$	0	$T_{hp}=6$	2	$T_{hf}=8$
	Compensator P1	$22-Bit\ \ mult$	128	$2688LUT$	4	$1336LUT$
		$22-Bit\ \ add$	112	$11880FF$	19	$704FF$
		$22-Bit\ reg$	540	$256HM$	32	$8HM$
		$22-Bit\ \ mux$	0	$T_{hp}=15$	40	$T_{hf}=101$
	Compensator P2	$22-Bit\ \ mult$	96	$1632LUT$	4	$1336LUT$
		$22-Bit\ add$	68	$3432FF$	19	$704FF$
		$22-Bit\ reg$	154	$192HM$	32	$8HM$
		$22-Bit\ \ mux$	0	$T_{hp}=15$	40	$T_{hf}=101$
	Compensator P3	$22-Bit\ mult$	48	$1259\ LUT$	4	$1443LUT$
		$22-Bit\ add$	32	$2169FF$	3	$1443FF$
		$22-Bit\ \ reg$	81	$100HM$	48	$12HM$
		$22-Bit\ mux$	0	$T_{hp}=89$	40	$T_{hf}=183$
		$22-Bit\ \ \mathrm{div}$	1		1	
	Compensator P4	$22-Bit\ \ mult$	96	$1912LUT$	4	$1688LUT$
		$22-Bit\ add$	65	$2574FF$	19	$1056FF$
		$22-Bit\ \ reg$	117	$192HM$	48	$8HM$
		$22-Bit\ \ mux$	0	$T_{hp}=15$	40	$T_{hf}=101$
		$22-Bit\ \ sub$	16		16	

Table 6.2: Consumed resources of the EKF functional modules

Appendix-B

1. FPGA-based matrix multiplier

In this section, the FPGA architecture of the implemented matrix multiplier is presented[113]. M1 and M2 are the two 4x4 matrices to be multiplied.

T is the resulting matrix after a set of 64 scalar 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. The corresponding latency is then equal to 48 instead of 4 without factorization. Figure 6.1 presents the corresponding architecture.

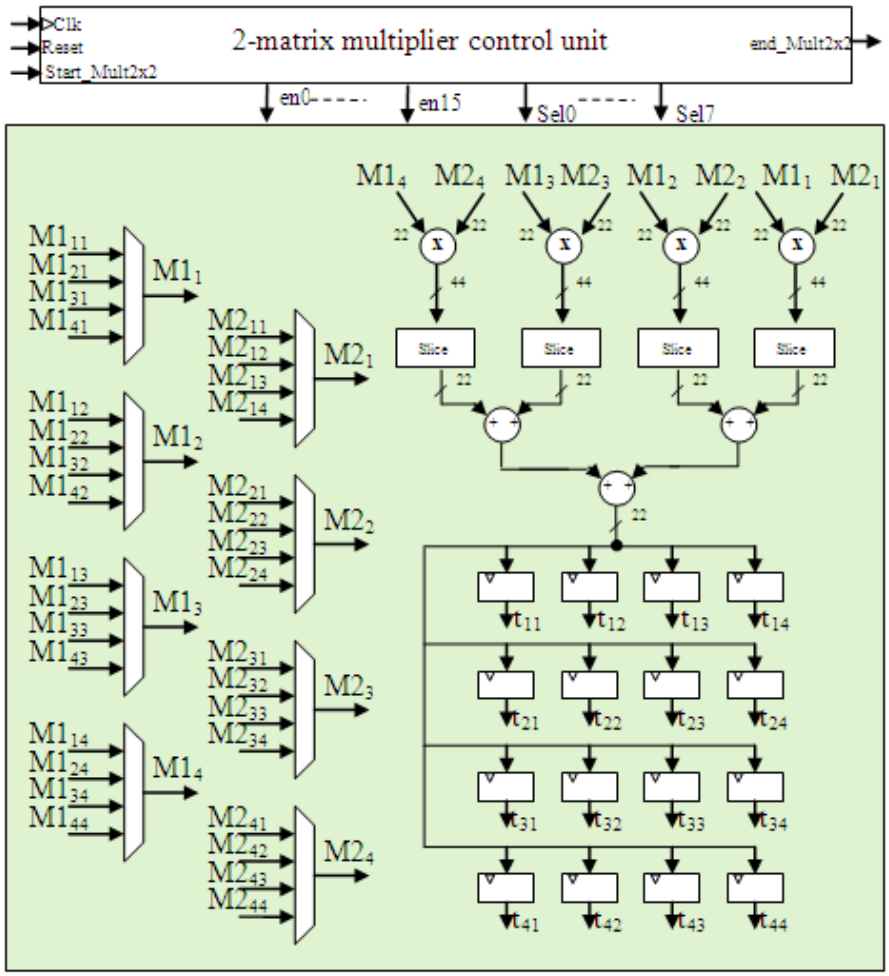


Figure 6.1 : Matrix multiplier architecture.

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 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.

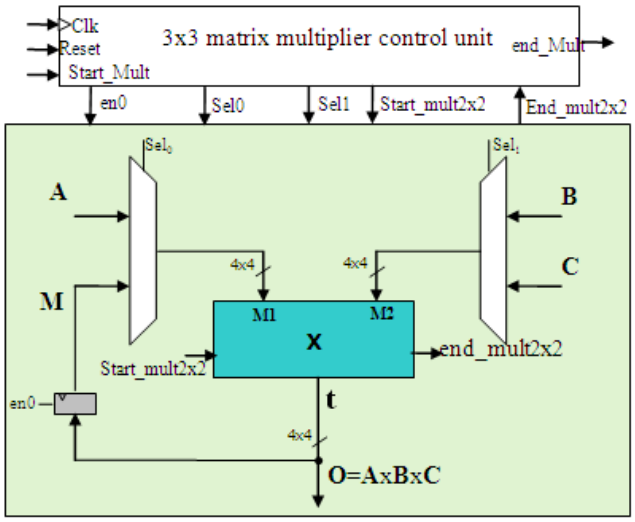


Figure 6.2 : 3-matrix multiplier architecture.

3. Matrix inversion

The implemented 2x2 matrix inversion module is defined by equation below

$$[A]^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{6.1}$$

In the case of having chosen the Xilinx FPGA solutions, the inversion of the matrix determinant is made using the Pipelined-Divider IP [73]. Figure 6.3 presents the corresponding configuration wizard.

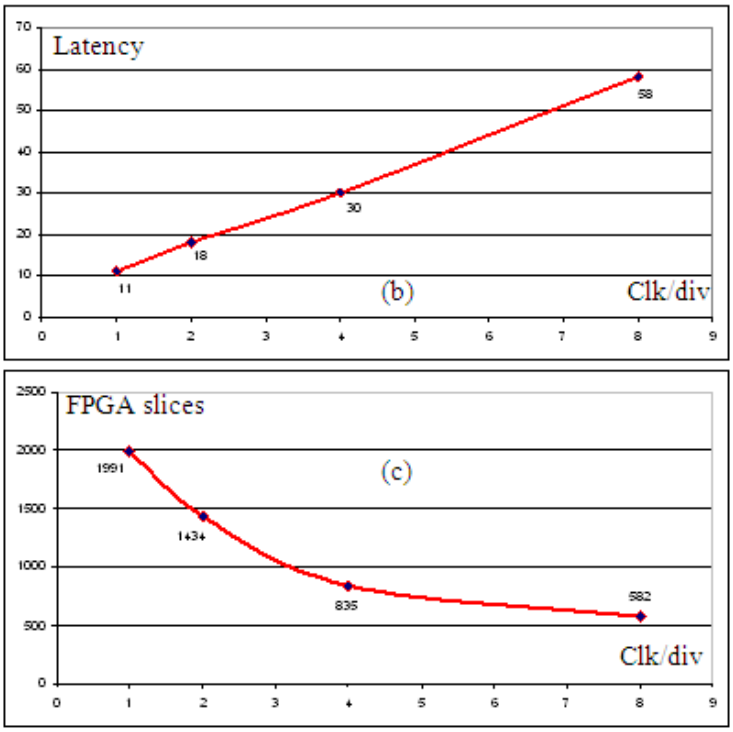


Figure 6.3: Xilinx pipelined divider IP

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 6.3 presents 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 have been analyzed in the case of clk/div set to 8.

Appendix-C

FPGA logic cells comparison

In today's market, FPGA internal architecture differs from one company to another. Having in mind the definition of a general metric, we provide an approximate relationship between different FPGA internal architecture. These relations will be used to evaluate partitioning results using different technologies. they were the results of an extensive benchmarking to determine the logic usage comparison. The proposed approximation is derived from [74]-[76]. Some validation tests were made also to confirm the validity of these coefficients.

	6 – Bit LUTs equivalent Virtex – 5
Spartan – 3E (Slice)	1.5
Altera (stratix – II, ALM)	1.84
Altera (Cyclone, LE)	0.73
Actel (Versatiles)	0.25

Table 6.3: FPGA logic cells comparison

The hardware multipliers differ also from one technology to another. Table6.3 presents the model of hardware multiplier provided by each technology.

Multiplier	Spartan-3A	Virtex-5	Altera
18x18	$Ceil(\frac{N}{17}) * Ceil(\frac{N}{17})$	—	$Ceil(\frac{N}{18}) * Ceil(\frac{N}{18})$
25x18	—	$Ceil(\frac{N}{17}) * Ceil(\frac{N}{24})$	—

Table 6.4: Relationship bit width-Hardware multiplier consumption

Processor performance comparison: Dhrystone 2.1 benchmark

The Dhrystone 2.1 benchmark was proposed in 1988. It is used to measure the performance of computer systems. Dhrystone is a benchmark written in the C language including a mix of mathematical and other operators. Based on Xilinx and Altera references, Table6.5 presents a DMIPS (the number of iterations of the main code loop per second) results of the Microblaze, Power-PC440 and Nios-II. This comparison shows that, with a hard-core processor "Power-PC440", we can have a gain of 1.68 in terms of DMIPS performance compared to Microblaze, for a clock frequency of 100 Mhz.

	Virtex – 5 Microblaze	**Virtex – 5 PowerPC440**	**Stratix – II Nios – f/II**
DMIPS Performance	119	200	117

Table 6.5: DMIPS performance comparison for CLK=100Mhz

Appendix-D

Per unit algorithm development

In order to obtain normalized model, al the variables are replaced by their corresponding per-unit values calculated using the following relation.

$$V_n = \frac{V}{V_B} \tag{6.2}$$

where V is the variable value, V_B is the base value and V_n is normalized variable value. Table6.6 presents the used base values.

	value
Base value of the voltage "V_B"	$\sqrt{6}.V_{nom} = 560V$
Base value of the current "I_B"	$G_{sensor}.G_{ADC}.\sqrt{2}.I_{nom} = 20A$
Base value of the speed "ω_B"	$2.p.\frac{\pi}{30}.N_{nom} = 628\ rd/s$
Base value of the position "θ_B"	$2^*\pi(rd)$

Table 6.6: Base values

where $V_{nom} = 230\ V$.
$I_{nom} = 1.5\ A$.
$N_{nom} = 1500\ rpm$.

Appendix-E

Time overhead of RTOS services using soft processor core Cortex-M1:

To evaluate the time overhead added by the RTOS services, some measurements were performed using the Cortex-M1. The program code was saved in an external SRAM. A timer core was used to perform the timing measurement of each service. The configuration of this timer is provided by the prescale value and the clock frequency[76]. The obtained results are provided Table 6.7.

Functions	**Measured clock cycles**
Task Management	
OSTaskCreateExt(. . .)	55
OSTaskChangePrio(. . .)	416
OSTaskDelReq(. . .)	110
OSTaskSuspend(. . .)	161
OSTaskDel(. . .)	483
OSTaskResume(. . .)	289
Semaphores	
OSSemCreate(. . .)	152
OSSemAccept(. . .)	101
OSSemPost(. . .)	115
Message Mailboxes	
OSMboxCreate(. . .)	154
OSMboxPost()	126
OSMboxAccept(. . .)	100
Time Management	
OSTimeGet(. . .)	74
OSTimeSet(. . .)	76

Table 6.7: Time overhead of the RTOS services

Personal publications

International Journals

- E. Monmasson, L.Idkhajine, M. N. Cirstea, I.Bahri, A.Tisan, M.Naouar, "FPGAs in Industrial Control Applications", IEEE Trans. On Ind. Inf, vol 7, No 2, pp 224, May 2011.
- H.Berriri, M-W.Naouar, I.Bahri, I. Slama-Belkhodja, E.Monmasson, "FPGA-Based Fault Tolerant Hysteresis Current Control for AC Machine Drives", IET Electrical Power Applications Journal, accepted for publication in 2011.
- I.Bahri, L.Idkhajine, E. Monmasson, M-A.Ben khelifa, "HW-SW partitioning of SoC based Sensorless Controller for a Synchronous Machine using an Extended Kalman Filter", submitted in MATCOM journal 2011.

National Journals

- M-W. Naouar, I. Bahri, M. Abdellatif, I. Slama-Belkhodja, E. Monmasson « Présentation d'un bureau d'études destiné à l'apprentissage des techniques de contrôle de courant usuelles d'un système électrique », La revue 3EI-N°53, Juin 2008 –pp. 11-20

International Conference

- I.Bahri, L.Idkhajine, E.Monmasson, M-A.Ben Khelifa," FPGA SoPC based Sensorless Controller for a Synchronous Machine using an Extended Kalman", Electrimacs 2011 , 6 June, Paris
- I.Bahri, A. Maalouf, L. Idkhajine, E. Monmasson, "FPGA-based implementation of sensorless AC drive controllers for embedded Electrical Systemsapplications", IEEE Sled 2011, September 1-2, Birmingham
- I.Bahri, E.Monmasson, F.Verdier, M-A.Ben khelifa,"SoPC-based current controller for permanent magnet synchronous machine drives", IEEE ISIE 2010, 4-7 July 2010-Bari, Italy.
- I.Bahri, E.Monmasson, F.Verdier, M-A.Ben khelifa, «Design and validation methodology of FPGA-based motor drive for High Temperature environment", ESARS 2010, 19-21 October 2010-Bologna, Italy.
- E. Monmasson, L. Idkhajine, I. Bahri, M-W- Naouar, L. Charaabi - Design methodology and FPGA-based controllers for Power Electronics and drive applications. The 5th IEEE Conference on Industrial Electronics and Applications (ICIEA'10), Taichung, Taiwan, 2010
- I.Bahri, I.Slama Belkhodja, E.Monmasson "FPGA-based Real-Time Simulation of Fault Tolerant Current Controllers for Power Electronics", IEEE ISIE 2009, 5-8 July 2009-Seoul, Korea
- I.Bahri, J.arbi, I.Slama-Belkhodja, E.Monmasson,"FPGA-based Fault Tolerant Current Controllers for Induction Machine", In Conf.Electromotion, 1-3 July 2009 - Lille, France
- M. Abdellatif, I.Bahri, I. Slama-Belkhodja « Comparative Study of Grid Voltage Angle Calculation for a DFIG based Wind System », SSD'07 - Conference on power electrical system system, 19-22 March 2007, Hammamet, Tunisia

Bibliography

[1] OPEC, "Annual Statistical Bulletin", Rapport, Organization of the Petroleum Exporting Countries, 2004a, pages : 1-140

[2] Cutts, S.J., "A collaborative approach to the more electric aircraft," Proc. Power Electronics, Machines and Drives International Conference, Bath, UK, April 16-18, pp. 223-228 (2002).

[3] J. Munoz-Castaner, R. Asorey-Cacheda, F.J. Gil-Castineira, F.J. Gonzalez-Castano, P.S. Rodriguez-Hernandez, "A Review of Aeronautical Electronics and its Parallelism with Automotive Electronics", IEEE Trans. On Ind. Electron., vol. 54, no. 99, April 2010

[4] Q. N. Le, J.W. Jeon, "Neural-Network-Based Low-Speed-Damping Controller for Stepper Motor with an FPGA", IEEE Transactions on Industrial Electronics, Vol. 57, Issue 9, pp. 3167–3180, September 2010.

[5] Ying-Shieh Kung, Ming-Shyan Wang, Tzu-Yao Chuang, "FPGA-based self-tuning PID controller using RBF neural network and its application in X-Y table", IEEE International Symposium on Industrial Electronics, 2009. ISIE 2009, pp. 694 – 699, July 2009.

[6] D. Kim: "An implementation of fuzzy logic controller on the reconfigurable FPGA system," IEEE Trans. on Industrial Electronics, Vol. 47, no. 3, pp.703-715, Jun 2000.

[7] C. Cecati, F. Ciancetta, P. Siano, "A FPGA/fuzzy logic - Based multilevel inverter", IEEE International Symposium on Industrial Electronics, ISIE 2009, pp. 706-711. July 2009.

[8] 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.

[9] E. Monmasson, M. Cirstea, "Guest Editorial special section on field programmable gate arrays (FPGAs) used in industrial control systems", IEEE Trans. On Ind. Electron., vol. 55, no. 4, pp. 1499–1500, April 2008

[10] I.Bahri, E.Monmasson, F.Verdier, M-A.Ben khelifa,"SoPC-based current controller for permanent magnet synchronous machine drives", IEEE ISIE 2010, 4-7 July 2010-Bari, Italy.

[11] M.-W. Naouar, E. Monmasson, A. A. Naassani, I. Slama-Belkhodja and N. Patin, "FPGA-based current controllers for AC machine drives—A review," IEEE Trans. Ind. Electron., vol. 54, no. 4, pp. 1907–1925, Aug. 2007

[12] K. Eshraghian, "SoC Emerging Technologies", IEEE Proceedings, vol. 94, no. 6, Jun. 2006, pp. 1197 – 1213.

[13] F. Salewski, S. Kowalewski, "Hardware/Software Design Considerations for Automotive Embedded Systems", IEEE Trans. On Ind. Informatics, Vol. 4, n° 3, pp. 56, August 2008

[14] J. A. Weimer, "High temperature Power Electronics for The More Electric Aircraft", Air force Research Laboratory, Mai 2004.

[15] Air&Cosmos, "Le 787 une étape majeur vers l'avion électrique", Air and Cosmos, Vol. N°2000, October 2005, pp 44-45

[16] Gong, G., Heldwein, M.L., Drofenik, U., and Kolar, J.W., "Comparative evaluation of three-phase high power factor AC-DC converter concepts for application in future more electric aircrafts," IEEE Applied Power Electronics Conference, Anaheim (CA), USA, Feb. 22-26 (2004).

[17] I.Bahri, E.Monmasson, F.Verdier, M-A.Ben Khelifa, "Design and validation methodology of FPGA-based motor drive for High Temperature environment", ESARS 2010, 19-21 October 2010-Bologna, Italy.

[18] H. P. Schöner, P. Hille, "Automotive Power – New Challenges for Power Electronics", IEEE, 2000.

[19] P. G. Neudeck, R. S. Okojie and al.; "High temperature Electronics – A rôle for Wild Bandgap Semiconductors?", Proceedings IEE, Vol. 90, Issue 6, June 2002, page : 1065- 1076

[20] O.Vermesan, R.John, M.Ottella, H.Gall and R.Bayerer,"High temperature nanoelectronics for electrical and hysbrid vehicles," In.conf.IMAPS High Temperature Electronics Network (HiTEN2009).

[21] Schlumberger, "Complete series of rotary steerable systems for reduced well construction cost", Power Drive Xtra Series, 2002.

[22] ITRS. International Technology Roadmap for Semiconductors, 2009 update system drivers.

[23] L.Adams"Choosing the right architecture for real –time signal processing designs", Texas Instruments white paper SPRA879-November 2002.

[24] L.Idkhajine, E. Monmasson, M-W. Naouar, A.Prata and K.Bouallaga, "Fully integrated FPGA-based controller for synchronous motor drives," IEEE Trans. Ind. Electron., vol. 56, n°. 10, pp. 4006-4017, Oct. 2009.

[25] R. Obermaisser, P. Gutwenger,"Model-Based Development of MPSoCs with Support for Early Validation", in Proc. IEEE IECON'09 Conf., 2009, CD-ROM.

[26] G. Martin, "Overview of the MPSoC Design Challenge". in Proc. DAC'06 Conf., 2006, CD-ROM.

[27] C. Zeferino, M. E. Kreutz, L. Carro, and A.Susin. "A study on communication issues for systems-on-chip," In Proc. SBCCI Conf., 2002, pp. 121–126.

[28] A. Brinkmann, J. Niemann, I. Hehemann, D. Langen, M. Porrmann, and U. Ruckert. "On-chip interconnects for next generation systems-on-chips," in Proc. IEEE ASIC SOC Conf., 2002, pp. 212–215.

[29] J. J. Labrosse, MicroC/OS-II: The Real-Time Kernel. CMP Books, 2002.

[30] B. Earl Wells and S.Ming Loo,"On the Use of Distributed Reconfigurable Hardware in Launch control Avionics", Digital Avionics Systems Conference, 2001.

[31] C. Paiz, J. Hagemeyer, C. Pohl, M. Porrmann, U. Ruckert, B. Schulz, W. Peters, J. B¨ocker, "FPGA-Based Realization of Self-Optimizing Drive-Controllers", In.Conf, IECON, 3-5 November 2009 at Alfandega Congress Center, Porto, 2009.

[32] B. Schulz , C. Paiz, J. Hagemeyer, S. Mathapati, M. Porrmann, J. Bocker, "Run-Time Reconfiguration of FPGA-Based Drive Controllers", In.Conf EPE, 2 - 5 September 2007, Aalborg- Denmark, 2007.

[33] Y.WANG, X.YIN, Z.ZHANG, « Monitor System for Protection Device Based on Embedded RTOS », J. Electromagnetic Analysis& Applications, 2009, 245-248.

[34] F. Berthelot, F. Nouvel, and D. Houzet, "Design methodology for dynamically reconfigurable systems," JFAAA , Dijon France, pp. 47–52, January 2005.

[35] M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vicentelli, "A Formal Specification Model for Hardware/Software Codesign," Technical Report UCB/ERL M93/48, Dept. EECS, University of California, Berkeley, June 1993.

[36] E. Stoy, "A Petri Net Based Unified Representation for Hardware/Software Co-Design," Licentiate Thesis, Dept. of Computer and Information Science, Linköping University, Linköping, 1995.

[37] T. Parks, J. L. Pino, and E. A. Lee, "A Comparison of Synchronous and Cyclo-Static Dataflow," in Proc. 29th Asilomar Conference on Signals, Systems and Computers, 1995, pp. 204-210.

[38] C. A. R. Hoare, Communicating Sequential Processes. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[39] F. Boussinot and R. de Simone, "The ESTEREL Language," in Proc. IEEE, vol. 79, pp. 1293-1304, Sept. 1991.

[40] SystemC on-line documentation. Available in : www.systemc.org

[41] on-line documentation. Available in :http://embedded.eecs.berkeley.edu/research/hsc

[42] on-line documentation. Available in :http://ptolemy.eecs.berkeley.edu/

[43] Syndex on-line documentation. Available in: www.syndex.org

[44] Grandpierre, C Lavrenne, Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessor", in Proceedings of CODES'99 7th International Workshop on Hardware/ Software Co-Design conference, 1999, CD-ROM.

[45] Ptolemy on-line documentation. Available in:www.ptolemy.berkeley.edu

[46] R.W. Johnson, J.L. Evans, P.Jacobsen, J.R.Thompson, and M.Christopher, "The Changing Automotive Environment: High-Temperature Electronics" in IEEE Trans. On electronics packaging manufacturing, vol. VOL. 27, no. 3, July 2004

[47] R. Normann, 'First High-Temperature Electronics Products Survey 2005', SAND2006, California

[48] S.Chiu, J.Chang, C.Neft, F.Morris, E.Dines, J.Hughes, "Design of motor drive for high temperature enviroment," IECEC August 2-6, 1998

[49] Y.Yuan, G.Yong, C.Lijie,"Design and Test of Novel Programmable Digital Three Phases SPWM Chip", In.Conf.Power Electronics and Motion Control, 2006. IPEMC 2006

[50] I.Bahri, I.slama Belkhodja, E.Monmasson "FPGA-based Real-Time Simulation of Fault Tolerant Current Controllers for Power Electronics", IEEE ISIE 2009, 5-8 July 2009-Seoul, Korea

[51] G.G. Parma, V.Dinavahi, "Real time Digital Hardware simulation of power electronics and Drives" IEEE Transactions on Industrial Electronics, vol.22, no.2, April 2007.

[52] L. Ben-Brahim, M. Benammar, and M. A. Alhamadi, "A resolver angle estimator based on its excitation signal," IEEE Trans. Ind. Electron., vol. 56, no. 2, pp. 574–580, Feb. 2009.

[53] J. E. Volder "The CORDIC Trigonometric Computing Technique", IEE Transactions on. Electronic Computers, Vol. EC-8, pp. 330-334, 1959.

[54] K. Zhou, D. Wang, "Relationship between space-vector modulation and three-phase carrier-based PWM: A comprehensive analysis", IEEE Transactions on Industrial Electronics, Vo. 49, no.1, pp. 186-196, February 2002.

[55] J-Y Midy, "Méthodologie de développement des tests pour une application de contrôle d'un système électrique embarqué pour l'aéronautique", Enseigner l'Electrotechnique et l'Electronique Industrielle, 3EI, Juin 2008.

[56] E.Salman, A.Dasdan, F.Taraporevala, K.Küçükçakar, and E.G.Friedman, Fellow, "Exploiting Setup–Hold-Time Interdependence in Static Timing Analysis," Ieee Transactions on computer –aided design of integrated circuits and systems, vol.26,26,no.6,june 2007.

[57] ADC-DAC devices description, available at: http://www.analog.com

[58] H.Hu, V.Yousef zadeh and D.Maksimovic,' No uniform A/D Quantization for Improved Dynamic Responses of igitally Controlled DC–DC Converters,' IEEE Trans. Ind. Electron., vol.. 23, no. 4, July.2008

[59] J.Böcker, S.Beineke, A.Bähr, "On the Control Bandwidth of Servo Drives", in Proc. IEEE, EPE 2009 - Barcelona

[60] C.Klarenbach, H.Schmirgel, J.Krah, "Design of Fast and Robust Current Controllers for Servo Drives based on Space Vector Modulation", in Conf.PCIM Europe 2011, 17.-19. May 2011, Nuremberg, Germany

[61] D. G. Holmes, T. A. Lipo, B. P. McGrath, and W. Y. Kong, "Optimized Design of Stationary Frame Three Phase AC Current Regulators," IEEE Trans. On power electronics., vol. 24, no. 11, Nov. 2009

[62] K. G. Shin and X. Z. Cui, "Computing time delay and its effects on real-time control systems ," IEEE Trans. Control Syst. Technol., Vol. 3, No. 2, pp. 218-224, Jun. 1995.

[63] D. Maksimovic and R. zane, "Small-signal discrete-time modeling of digitally controlled DC-DC converters,"in Proc. COMPEL, pp.231-235, 2006.

[64] K.JI, W-J.Kime"Optimal bandwidth allocation and QoS-adaptive control Codesign for networked control systems" in International journal of control, automation and system, vol.6, no.4, pp596-606, August 2008.

[65] E. Monmasson, L. Idkhajine, I. Bahri, M.W. Naouar, L. Charaabi, "Design methodology and FPGA-based controllers for Power Electronics and drive applications", In Proc. ICIEA'2010 Conf., pp. 2328-2338, Taichung, Taiwan.

[66] S. Bolognani, L. Tubiana, M. Zigliotto, "Extended Kalman Filter Tuning in Sensorless PMSM Drives", IEEE Trans. on Ind. Electron., vol. 39, no. 6, pp. 276 – 281, November 2003.

[67] A. Akrad, M. Hilairet, D. Diallo, "A Sensorless PMSM drive using a two stage Extended Kalman Estimator", In Proc. IECON'2008 Conf., pp. 2776-2781, Orlando, Florida, USA.

[68] Texas Instruments Europe, "Sensorless Control with Kalman Filter on TMS320 Fixed-Point DSP", TI. Literature no. BPRA057, July 1997

[69] A.Ben Salem, S.Ben Othman and S.Ben Saoud "Field Programmable Gate Array -Based System-on-Chip for Real-Time Power Process Control", American Journal of Applied Sciences, 127-139, 2010

[70] S.C. Huerta, A. de Castro, O. Garcia, J.A. Cobos, "FPGA-Based Digital Pulsewidth Modulator With Time Resolution Under 2 ns", IEEE Trans. on Power Electron., vol. 23, n°6, pp. 3135-3141, Nov. 2008.

[71] B. K. Bose, "Neural Network Applications in Power Electronics and Motor Drives - An Introduction and Perspective", IEEE Transactions on Ind. Electronics, vol. 54, no. 1, pp. 14-33, Feb. 2007.

[72] F. Salewski, S. Kowalewski, "Hardware/Software Design Considerations for Automotive Embedded Systems", IEEE Trans. On Ind. Informatics, Vol. 4, n° 3, pp. 56, August 2008

[73] Xilinx on-line documentation. Available in :www.xilinx.com

[74] Cores Technologies on-line documentation. Available: www.1-core.com

[75] Altera on-line documentation. Available in :www.altera.com

[76] Actel on-line documentation. Available in :www.actel.com

[77] J.Coburn, S.Ravi and A.Raghunathan " Hardware Accelerated Power Estimation", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)

[78] Ravi, A. Raghunathan and S. Chakradhar, , "Efficient RTL power estimation for large designs," in Proc. Int. Conf. VLSI Design, Jan. 2003.

[79] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation, and optimization," in Proc. Design Automation Conf., pp. 504–511, June 1997.

[80] Bjuréus, P. Millberg, M. Jantsch, "FPGA resource and timing estimation from Matlab execution traces", Proceedings of the international symposium on hardware/software codesign—(CODES), pp. 31–36,2002

[81] K.M. Buyuksahin and F.N. Najm. "High-Level Area Estimation",In ISLPED'02, pages 271–274, August 2002.

[82] F. Fummi, G. Perbellini, M. Loghi, and M. Poncino. ISS-centric modular HW/SW co-simulation. In Proc. of ACM GLSVLSI, pp. 31–36. 2006.

[83] Enzler, R. Jeger, T. Cottet, D. and Troster, "High-level area and performance estimation of hardware building blocks on FPGAs. In Proceedings of international workshop on field-programmable logic and applications (FPL), pp. 525–534,2000

[84] T. Wiangtong, P.Y.K. Cheung, and W. Luk. "Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware-Software Codesign". Design Automation for Embedded Systems, 6:425–449, 2002.

[85] B. Knerr, M. Holzer, and M. Rupp,"HW/SW Partitioning Using High Level Metrics",in the proceedings of the International Conference on Computing, Communications and Control Technologies (CCCT), pp. 33-38, Austin, 2004

[86] T.Wiangtong, "Hardware/Software partitioning and scheduling for Reconfigurable Systems", Ph.D thesis, Imperial College, London, 2004.

[87] J. Harkin, T. M. McGinnity, and L. P. Maguire. "Partitioning methodology for dynamically reconfigurable embedded systems". IEE Proceedings - Computers and Digital Techniques, 147(6):391–396, November 2000.

[88] Y. Le Moullec, N. Ben Amor, J-Ph. Diguet, M. Abid, and JL. Philippe, "Multi-Granularity Metrics for the Erea of Strongly Personalized SOCs," pp. 674–679, March 2003.

[89] P.Knudsen and J.Madsen,"Integrating Communication Protocol Selection with Hard-ware/Software Codesign", Ieee.Transaction on computer-aided design of integrated circuits and systems, vol.18,no.8,August 1999

[90] Henkel, J., and Ernst, R.," An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques". IEEE Transactions on Very Large Scale Integration Systems, Vol. 9, No. 2, 273 - 289. 2001

[91] G. Qu et al., "Power Minimization using System-Level Partitioning of Applications with Quality of Services Requirements", Proc of Int. conf. on CAD. pp. 343-346, 1999.

[92] F. Vahid and D. Gajski,"Incremental Hardware Estimation during Hardware/Software Functional Partitioning ",IEEE Transactions on VLSI Systems, Vol. 3, No. 3, pp. 459-464, September 1995.

[93] 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 In Proc. IECON'02 Conf Proc, Sevilla, Spain, 2002

[94] B. Knerr, M. Holzer and M. Rupp,"A Fast Rescheduling Heuristic of SDF Graphs for HW/SW Partitioning Algorithms",in proceding of the 1rst.int.on communication system software, Middleware, Jan08, New delhi.India,2006.

[95] Binh, N. N., Imai, M., Shiomi, A., and Hikichi, N. . "A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate counts". Proceedings of 33rd Design Automation Conference (Las Vegas, NV, USA). 527 - 532,1996

[96] M. L.Vallejo and J. C. Lopez,"On the Hardware-Software Partitioning Problem: System Modeling and Partitioning Techniques" ACM Transactions on Design Automa-tion of Electronic Systems, Vol. 8, No. 3, July 2003, p 269–297.

[97] J. Henkel. "A Low Power Hardware/Software Partitioning Approach for Core-Based Embedded Systems". Proceedings of the 36th ACM/IEEE Conference on Design Automation (DAC), pp. 122–127, 1999.

[98] Walter H. Kohler and Kenneth Steiglitz, "Characterization and theoretical comparison of branch-and-bound algorithms for permutation problems," J. ACM, vol. 21, no. 1, pp. 140–156, 1974.

[99] R. Niemann and P. Marewedel." An Algorithm for Hardware/ Software Partitioning Using Mixed Integer Linear Programming". Design Automation for Embedded Systems, 2(2):125–63, March 1997.

[100] K. S. Chatha and R. Vemuri, "An Iterative Algorithm for Hardware- Software Partitioning, Hardware Design Space Exploration and Scheduling,", no. 5, pp. 281–293, 2000

[101] S.Banerjee and N.Dutt,"Efficient Search Space Exploration for HW-SW Partitioning", Stockholm, Sweden,September 8–10, 2004,

[102] G.Stitt, R.Lysecky, F.Vahid ,"Dynamic Hardware/Software Partitioning: A First Approach", DAC 2003, Anaheim, California, USA,June 2-6, 2003,

[103] F. Vahid and T. Dm Le, "Extending the Kernighan/Lin Heuristic for Hardware and Software Functional Partitioning," Design Automation for Embedded Systems, pp. 237–261, 1997.

[104] T.wiangtong, P.Y.K.Cheumg, w.Luk "Comparing three heuristics methods for functional partitioning in Hardware-Software co-design", Design automation for embedded systems, 2002

[105] P.Mudry, G.Zufferey and G.Tempesti " A Dynamically Constrained Genetic Algorithm For Hardware-software Partitioning", Seattle, Washington, USA,July 8–12, 2006,

[106] M.Jagadeeswari, M.C.Bhuvaneswari,"A Fast Multi-Objective Genetic Algorithm for Hardware-Software Partitioning In Embedded System Design", ICGST-AIML Jour-nal, ISSN: 1687-4846, Volume 8, Issue II, September 2008

[107] Zou, Y. Zhuang, Z., and Cheng, H. HW-SW partitioning based on genetic algorithm. . In Proceedings of Congress on Evolutionary Computation (Anhui, China). 628- 633,2004

[108] K.S.Chatha and R.Vemuri, "An iterative algorithm for hardware-software partitioning, hardware design space exploration and Scheduling," Journal of Design Automa-tion for Embedded Systems, vol.5, pp. 281-293, 2000.

[109] M.Savage, Z.Salcic, G.Coghill, G.Covic ," Genetic Algorithm for Codesign Optimization of DSP Systems in FPGAs", ICF'PT 2004.

[110] Q.Li and J.He,"A sophisticated architecture for evolutionary multiobjective optimization utilizing high performance DSP", ICES2007,LNCS 4684,pp.415-425,2007.

[111] L.Kaouane, M.Akil, Y.Sorel, T.grandpierre,"An automated design flow for optimized implementation of real-time image processing applications on FPGA", EUROCON 2003 International Conference on computer as a tool, Ljubljana, Slovenia, 22-24 September 2003.

[112] K.Deb, A.Pratap,S.Argarwal,T.Meyarivan,"A fast and elitist multiobjective genetic algorithm:NSGA-II", IEE transaction on evolutionary compution,182-197, 2002

[113] L.Idkhajine, "FPGA-based sensorless controller for AC drives", Phd thesis.University of Cergy-Pontoise, November 2010

[114] A.M. Haddad,"Sensorless Control of Brushless Synchronous Starter Generator Including Sandstill and Low Speed Region for Aircraft Application,"Phd thesis.Ecole Normal de Cachan, March 2011

[115] I.Bahri, L.Idkhajine,E.Monmasson,M-A.Benkhelifa,"FPGA SoPC based sensorless controller for synchronous machines using an Extended Kalman Filter," In.conf Electrimacs 2011, Paris,6-8 June.

[116] Z Jin, M Sindhwani and T Srikanthan : RTOS Acceleration on Soft-core Processors Using Instruction Set Customization , 2004 IEEE International Conference on Field Programmable Technology (FPT 2004), Australia. Brisbane, Australia, pp. 371-374, Dec 2004

[117] Sindhwani, M., Oliver, T.F., Maskell, D.L. and Srikanthan, T., "RTOS Acceleration Techniques - Review and Challenges", Sixth Real-Time Linux Workshop, Singapore, pp.123-128, Nov 2004

[118] Carlos Ferreira, Arnaldo S. R. Oliveira :RTOS Co-Processor Implementation in VHDL, proceedings of IP-ESC'09 – Intellectual Property - Embedded Systems Conference, Grenoble, France, December 2009

[119] Susanna Nordström : Configurable Hardware/Software Support for Single Processor Real-Time Kernels, Susanna Nordström (former), Lars Asplund, International Symposium on System-on-Chip, p 4, IEEE, Tampere, Finland, November, 2007

[120] Lindh, L. and Stanischewski, F. "FASTCHART – A Fast Time Deterministic CPU and Hardware Based Real-Time-Kernel", 1991

[121] Lindh, L. "Utilization of Hardware Parallelism in Realizing Real Time Kernels", Doctoal Thesis, 1994.

[122] D.Andrews, W. Peck,J. Agron, K. Preston,E. Komp, M. Finley, R. Sass, "hthreads: A Hardware/Software Co-Designed Multithreaded RTOS Kernel" In. IEEE Conf. Emerging Technologies and Factory Automation, 19-22 Sept. 2005 5. ETFA 2005.

[123] Klevin, T. and Lindh, L. "Scalable Architecture for Real-Time Applications and Use of bus-monitoring", 1999.

[124] Lindh, L., Klevin, T. and Furunäs, J., "Scalable Architecture for Real-Time Applications – SARA", 1999.

[125] Enblom, L. and Lindh, L. "Adding Flexibility and Real-Time Performance by Adapting a Single Processor Industrial Application to a Multiprocessor Platform", 2001.

[126] Lee, J., Mooney, V. J., Ingström, K., Daleby, A, Klevin, T. and Lindh, L. "A Comparison of the RTU Hardware RTOS with a Hardware/Software RTOS", 2003.

[127] T. Samuelsson, M. Åkerholm, P. Nygren, J. Stärner, and L. Lindh, "A Comparison of Multiprocessor Real-Time Operating Systems Implemented in Hardware and Software", 2003.

[128] R Haukilahti. Energy Characterization of a RTOS Hardware Accelerator for SoCs. In Swedish System-on-Chip Conference, Falkenberg, Sweden, March 2002.

[129] I.Bahri, A. Maalouf, L. Idkhajine, E. Monmasson, "FPGA-based implementation of sensorless AC drive controllers for embedded electrical Systems applications", IEEE Sled 2011, September 1-2, Birmingham

[130] H.V.Poussin," Conception d'une architecture multiprocessor pour la commande des systèmes électromécaniques », Phd thesis, Univercity of Chatolique de louvain 2003.

[131] M.W.Naouar, "Commande numérique à base de composants FPGA d'une machine synchrone", PhD Thesis, UCP-ENIT, France-Tunisia, 2007

[132] M.Sadir,"Exécutif Temps Réel Embarqué : Partitionnement Matériel / Logiciel", master project september 2011.