Hardware/Software Codesign Guidelines for System on Chip FPGA-Based Sensorless AC Drive Applications

Imen Bahri, *Member, IEEE*, Lahoucine Idkhajine, *Member, IEEE*, Eric Monmasson, *Senior Member, IEEE*, and Mohamed El Amine Benkhelifa

***Abstract*—This paper aims to provide Hardware/Software (Hw/Sw) codesign guidelines for system-on-chip field-programmable gate array-based sensorless ac drive applications. Among these guidelines, an efficient Hw/Sw partitioning procedure is presented. This Hw/Sw partitioning is performed taking into account both the control requirements (bandwidth and stability margin) and the architectural constraints (e.g., available area, memory, and hardware multipliers). A nondominated sorting genetic algorithm (NSGA-II) is used to solve the corresponding multi-objective optimization problem. The proposed Hw/Sw partitioning approach is then validated on a sensorless control algorithm for a synchronous motor based on an extended Kalman filter. Among the nondominated implementation solutions supplied by the NSGA-II, those that are considered as the most interesting are synthesized. Their time/area performances after synthesis are compared with success to their predictions. In addition, one of these optimal solutions is also tested on an experimental setup.**

***Index Terms*—Codesign guidelines, extended Kalman filter (EKF), field-programmable gate array (FPGA), hardware/software (Hw/Sw) partitioning, nondominated sorting genetic algorithm (NSGA-II), soft-core processor, system-on-chip (SoC).**

NOMENCLATURE

s, r	Stator and rotor index.
d,q	Synchronous reference frame index.
*,^	Reference quantity, estimated quantity.
a, b, c	Three-phase reference frame index.
i_{sa}, i_{sb}, I_{rd}	Stator and rotor currents.
i_{sd}, i_{sq}	d-q stator currents.
V_{DC}	DC link voltage.
V_{sa}, V_{sb}, V_{sc}	Three-phase stator voltages.
v_{sd}, v_{sq}	d-q stator voltages.
S_a, S_b, S_c	Switching signals for the VSI.
L_{sd}, L_{sq}	d-q stator inductances.
Msr	Mutual inductance.
Rs	Stator resistance.
$\theta e, \omega e$	Electrical position and speed.
x, u, y	State space vector, system input, and output vectors.
f(x,u), H	State space matrix and system output matrix.
w, v	State and output noise vectors.
k/k-1, k/k	Predicted and estimated quantities.
K, Fd	Kalman gain and Jacobian matrices.
P_n, P_0	State error and initial covariance matrices.
Q, R	Model noise and measurement noise covariance matrices.

Manuscript received July 20, 2012; revised November 17, 2012; accepted December 21, 2012. Date of publication February 08, 2013; date of current version October 14, 2013. Paper no. TII-12-0489.

I. Bahri is with the Laboratoire de Génie électrique de Paris (LGEP), University of Paris-Sud XI, 91400 Orsay, France (e-mail: imen.bahri@u-psud.fr).

L. Idkhajine and E. Monmasson are with the Systèmes et Applications des Technologies de l'Information et de l'Energie (SATIE), University of Cergy-Pontoise, 95031 Cergy-Pontoise, France (e-mail: eric.monmasson@u-cergy.fr; lahoucine.idkhajine@u-cergy.fr).

M. El Amine Benkhelifa is with the Equipes Traitement de l'Information et Systèmes (ETIS), University of Cergy-Pontoise, 95031 Cergy-Pontoise, France (e-mail: benkhelifa@ensea.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2013.2245908

I. INTRODUCTION

NOWADAYS, embedded control systems are becoming more and more sophisticated. This is a direct consequence of the rising of industrial requirements [1], [2]. These requirements are not just always limited to a higher level of performance. Indeed, the flexibility of the controller, its cost, and its time-to-market reduction are also of prime importance. To cope with all of these different challenges, a relevant use of the current digital technologies becomes crucial.

Among all of these digital technologies, field-programmable gate arrays (FPGAs) present the advantage to include both hardware and software resources. Hardware resources are usually used to accelerate the processing time by exploiting the inherent parallelism of the control algorithm to be implemented. This FPGA-based implementation has already been the focus of many researches in the field of ac drive applications [3]–[5].

More recently, it has also become typical to implement processor cores within FPGAs. Thus, FPGAs can now be considered as full system-on-chip (SoC) solutions, always allowing more flexibility for the embedded controllers. These SoC-based solutions include varied components like one or several processors, memories, hardware multiplier blocks, analog–digital con-

verters, matrix of programmable logic elements (FPGA Fabric), and busses, all in the same chip. To this purpose, FPGA manufacturers provide more open and efficient platforms introducing several RISC processor cores. The hard-core processors such as Xilinx's PowerPC 440, Microsemi's Cortex-M3, and Xilinx's dual ARM Cortex-A9 cores have a custom very large-scale integration (VLSI) layout that is integrated within the FPGA. As a general rule, these hard-core processors offer higher clock speeds with less flexibility. The soft-core processors such as Altera's Nios-II, Xilinx's MicroBlaze processors, and Microsemi's Cortex-M1 use existing FPGA logic cells to implement the processor cores [6]–[9]. The advantage of such approach is the flexibility but this implies slower clock rates. Additionally, heterogeneous multiprocessing architectures (MPSoCs) offer better performance especially in the case of complex digital applications. Indeed, MPSoC architectures provide a high level of scalability compared with monolithic cores, in particular in terms of power and performance [10].

To take full advantages of these SoC approaches and to accommodate the ever more demanding control applications, the use of hardware/software (Hw/Sw) codesign methodology is crucial. Along this line, a major issue to be addressed is to obtain an efficient partitioning of the control algorithm between software and hardware resources. During a decade, the Hw/Sw partitioning has been addressed by several methods using automated heuristic approaches. These approaches are based on optimization algorithms, such as simulated annealing algorithm, Tabu search algorithm and Genetic Algorithm (GA) [11]–[14]. These works focused on optimizing the processing time, the area and the power consumption.

The majority of these works have been proposed for signal and image processing applications but to our best knowledge, no optimized Hw/Sw partitioning strategy has been yet proposed for ac drives applications. Indeed, up to now, the partitioning between Hw/Sw resources for FPGA-based controllers dedicated to power electronics and drives applications was made only on the expertise of the designer [15]–[18], guided by engineering experience like implementing PWM and current control in hardware and position control in software.

In this paper, we propose Hw/Sw codesign guidelines for SoC FPGA-based sensorless ac drive applications. The main contribution of this work is to propose a systematic architectural exploration of the controller. Optimal architectural solutions are obtained via a nondominated sorting genetic algorithm (NSGA-II). The proposed Hw/Sw partitioning procedure takes into account heterogeneous constraints such as the control requirements (bandwidth and stability margin) and the Hw resources (available logic cells, memories, and hardware multipliers). It was tested on a sensorless speed controller of a synchronous machine using an extended Kalman filter (EKF). Indeed, this type of control strategy is representative in terms of complexity of what can be found in current ac drives. The FPGA-based SoC platform used in this project consists of a Xilinx Virtex-5 embedding a Microblaze soft-core processor.

The remainder of this paper is organized as follows. Section II presents the proposed Hw/Sw codesign guidelines. The specifications of the control application are detailed in Section III. Section IV synthesizes the algorithm development and its validation. Then, Section V presents the architectural exploration, where software, hardware, and mixed Hw/Sw SoC implementations are discussed. Section VI presents the experimental results. Finally, conclusions and perspectives are given in Section VII.

II. Hw/Sw Codesign Guidelines Presentation

The proposed Hw/Sw codesign guidelines aims to find an optimal partitioning of control algorithm between modules to be implemented in software and those to be implemented in hardware. This method defines a set of steps and rules to be followed in order to make the design process more manageable and less intuitive. It consists of a top-down design process that starts from the specifications of the control application to the final experimental validation. As shown in Fig. 1, this method is decomposed into four main steps: 1) specifications of the control application; 2) algorithm development; 3) architectural exploration; and 4) Hw/Sw integration and experimental validation. The proposed approach allows an automatic architectural exploration that integrates both the control (stability margin/bandwidth) and the resource (time/area) constraints. This automatic architectural exploration allows obtaining an optimal Hw/Sw partitioning of the functional modules with the help of an efficient evolutionary algorithm (NSGA-II). Thus, this automatic architectural exploration avoids the manual portioning that is one of the SoC design bottlenecks. In the four following sections, each step of the proposed design guidelines is explained optimal Hw/Sw partitioning of the functional modules with the help of an efficient evolutionary algorithm (NSGA-II). Thus, this automatic architectural exploration avoids the manual portioning that is one of the major SoC design bottlenecks.

III. Specifications of the Control Application

The structure of the developed sensorless control system is first overviewed and depicted in Fig. 2. Its hardware and algorithm specifications are discussed below.

A. Hardware Specifications

The power stage consists of a salient pole synchronous motor (SM) mechanically (see Table I) loaded by a powder brake and electrically fed by a Voltage Source Inverter (VSI). It consists in a SEMIKRON VSI module (IGBT modules, $\mathrm{VDC\_max} = 800$ V, $\mathrm{Imax} = 30$ A). The used electrical sensors provide measurements of stator currents and dc link voltage. The analog-to-digital converter (ADC) board is composed of four AD9221 12-b ADCs. In this study, the digital control unit is based on a Xilinx Virtex-5 FPGA (XC5VSX50T). It consists of 32640 lookup tables (LUTs) of six-input, 32640 flip-flops, 288 DSP48E (hardware multiplier and accumulator units), 4752-Kb blocks RAM. A transmission interface is scheduled so as to ensure a real-time data transfer between the controller and the Host-PC.

B. Algorithm Specifications

The developed sensorless speed controller is composed of the following subparts:

- speed controller based on Proportional-Proportional Integral (P-PI) speed regulator.

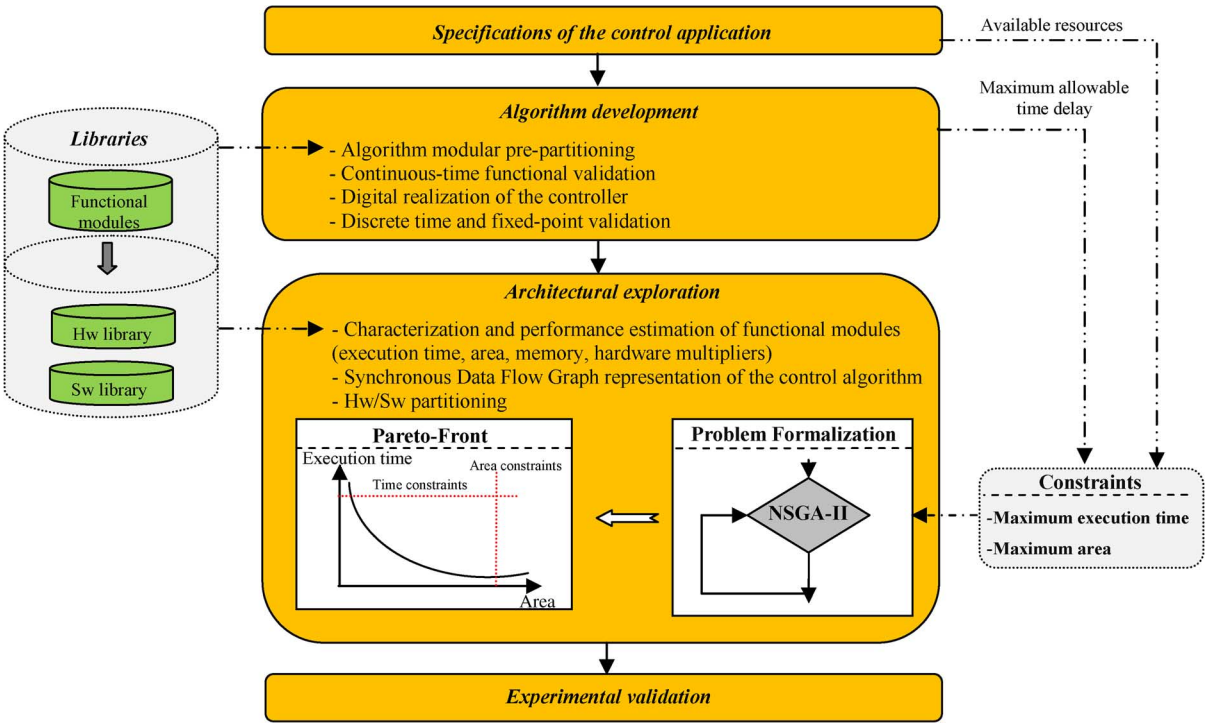

Fig. 1. Hw/Sw codesign guidelines flow

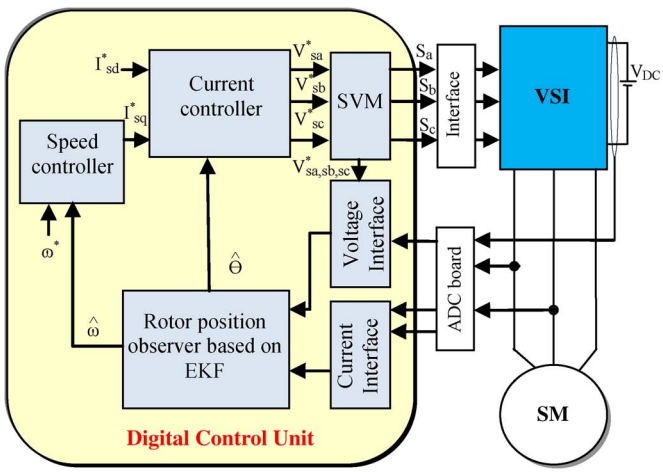

Fig. 2. Overview of the developed control application.

TABLE I
SYNCHRONOUS MACHINE PARAMETERS

0.8 KVA, 230V,1.5A, 50 Hz, 3 Phases, Y connection, 2 pole pairs

Stator resistance $R_s = 10.5\ \Omega$	Rotor resistance $R_r = 62.5\ \Omega$
d axis stator inductance $L_{sd} = 0.245$ H	Mutual inductance $M_{sr} = 0.86$ H
q axis stator inductance $L_{sq} = 0.229$ H	Nominal stator current 2.12 A

- stator current controller based on Anti-windup PI regulators, reference frame transformations and a space vector modulation (SVM) module. The control algorithm has to be synchronized with the PWM carrier avoiding the acquisition of ripples in the current data acquisitions.
- voltage interface that aims to generate the three-phase stator voltages from the duty cycles and the dc link voltage measurement.
- EKF module which estimates the rotor position and speed. The equations of this module are recalled in Table II.

IV. ALGORITHM DEVELOPMENT

During the algorithm development process, the designer makes the functional validation and prepares the algorithm for digital implementation. The different steps of this process are presented in the following subsections.

A. Algorithm Modular Prepartitioning

The algorithm modular prepartitioning aims to decompose the design into functional modules with different levels of granularity. It consists of independent and reusable modules. As shown in Fig. 3, the control algorithm can be divided into four levels of granularity. The first level corresponds to the scalar arithmetic operators such as multiplier, adder, and subtractor, for example. The second level is related to the matrix operators needed especially by the EKF treatment. The third level consists of the functional modules used by the current and the speed controllers and the EKF estimator. To provide more manageable matrix treatment, the Kalman gain module (see Table II) was split into four submodules as can be seen in Fig. 4. Finally, the highest level corresponds to the whole sensorless controller. All of the Hw modules of this library are written in VHDL.

B. Continuous-Time Functional Validation

This step aims to design the complete control system and to verify its functionality in continuous-time domain, using MATLAB/Simulink tools. The design objectives are a high

TABLE II
MAIN EQUATIONS OF THE EKF MODULE

Discrete-time system equations

$$x_k = f_d(x_{k-1}, u_{k-1}) + w = x_{k-1} + T_s . f(x_{k-1}, u_{k-1}) + w \quad (1)$$
$$y_k = h(x_k) + v$$

$$x = \begin{bmatrix} i_{sd} & i_{sq} & \omega & \theta \end{bmatrix}^T ; u = \begin{bmatrix} v_{sd} & v_{sq} \end{bmatrix}^T ; y = \begin{bmatrix} i_{sd} & i_{sq} \end{bmatrix}^T \quad (2)$$

$$f(x,u) = \begin{bmatrix} \frac{-R_s}{L_{sd}} \cdot i_{sd} + \frac{L_{sq}}{L_{sd}} \cdot \omega \cdot i_{sq} \\ \frac{-R_s}{L_{sq}} \cdot i_{sq} - \frac{L_{sd}}{L_{sq}} \cdot \omega \cdot i_{sd} - \frac{M_{sf}}{L_{sq}} \cdot I_{sd} \cdot \omega \\ 0 \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{sd}} & 0 \\ 0 & \frac{1}{L_{sq}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot u ; \quad H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \quad (3)$$

EKF equations

Prediction (4)

$$\hat{x}_{k/k-1} = f_d(\hat{x}_{k-1/k-1}, u_{k-1})$$

EKF Compensator – Kalman gain calculation (5)

Jacobian matrix : $F_{dk} = \left. \frac{\partial (f_d)}{\partial x} \right|_{x = \hat{x}_{k-1/k-1}}$ (6)

Kalman gain module:

- Covariance matrix prediction: $P_{k/k-1} = F_{dk} . P_{k-1/k-1} . F_{dk}^t + Q$, Initial condition P_0
- Kalman gain calculation: $K_k = P_{k/k-1} . H_{dk}^t . \left(H_{dk} P_{k/k-1} H_{dk}^t + R \right)^{-1}$ (7)
- Updating covariance matrix: $P_{k/k} = P_{k/k-1} - K_k H_{dk} P_{k/k-1}$ (8)

Innovation

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k \left(y_k - H \cdot \hat{x}_{k/k-1} \right) \quad (9)$$

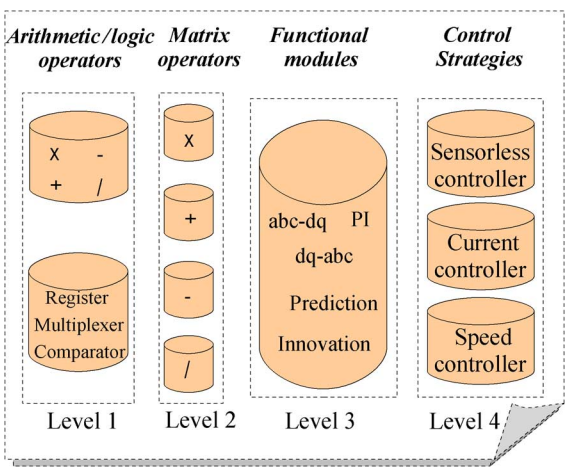

Fig. 3. Algorithm modular prepartitioning.

bandwidth, sufficient stability margin, and a good disturbance rejection. However, the stability of the controlled loop is affected by the time delay T_d. T_d is the sum of the following two terms:

- Sample and hold (S/H) delay resulting from the PWM. This introduces a time delay expressed by $G_{\text{delay}}(s) = e^{-s.T_s/2}$, where T_s is the sampling period.
- Computing time delay of the digital control T_{alg} which depends on the complexity of the algorithm and on the performance of the used digital device.

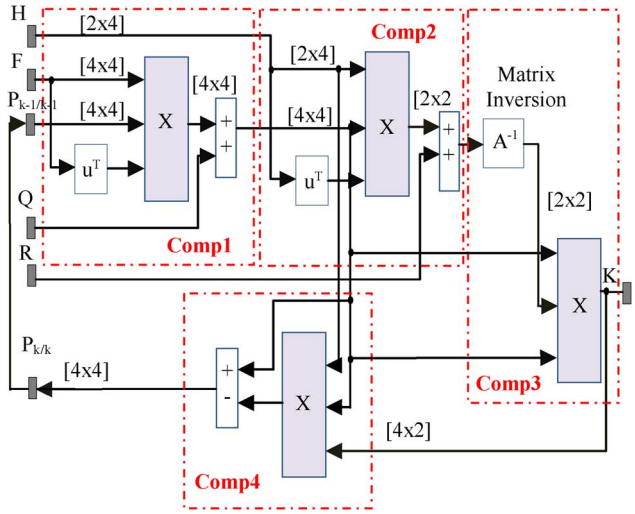

Fig. 4. Partitioning of the Kalman gain module into four submodules.

To design the PI regulators of the current loop, two metrics were taken into account: the phase margin "φ_m" and the desired closed loop bandwidth "f_c," which is more or less equal to the open-loop crossover frequency f_c.

In practice, the closed-loop bandwidth f_c must be much lower than the sampling frequency f_s as

$$f_c \leq \frac{f_s}{20}. \quad (10)$$

This rule of thumb leads to the maximum bandwidth value allowing a sufficient disturbance rejection. For the considered system, this leads to a bandwidth equal to 500 Hz. This value is used to determine the gain of the PI regulator. On the other hand, having fixed the desired bandwidth "f_c" and the phase margin "φ_m" (set to 60°), the maximum allowable time delay T_d can be derived from the phase relation of the open loop transfer function of the controlled system

$$T_d \leq \frac{\pi/2 - \varphi_m}{2.\pi.f_c}. \quad (11)$$

The exceeding of this limit decreases the desired phase margin and in some cases can cause system instability. Considering the S/H effect caused by the digital PWM, the maximum execution time devoted to the control algorithm T_{alg} can be derived from the previous expressions, yielding

$$T_{\text{alg}} \leq T_d - \frac{T_s}{2}. \quad (12)$$

As said before, the algorithm treatment is synchronized with the PWM carrier. Knowing that in this case, the sampling period and the switching period have been fixed to 100 μs, it yields that T_{alg} is equal to 116.67 μs.

C. Digital Realization of the Controller

The digital realization is the design step where the discretization, the normalization and the data quantization of the controller are achieved.

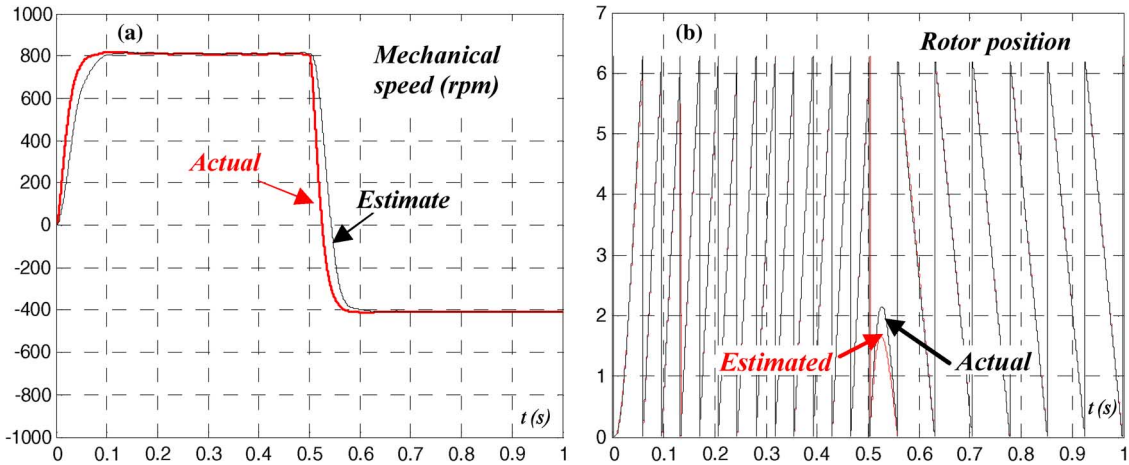

Fig. 5. Validation of the EKF Estimation of (a) speed and (b) position.

TABLE III
BIT NUMBER AND FIXED-POINT FORMATS OF THE CONTROLLER

Modules	Hw-implementation		Sw-implementation	
	Integer	Fractional	Integer	Fractional
Control modules	1	12	2	30
EKF modules	7	15	12	20

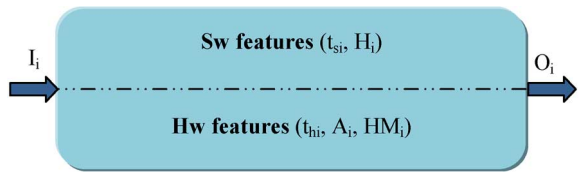

Fig. 6. Features of the functional module "M_i".

Regarding the discretization, the Forward Euler approximation is selected for the EKF predictor (see Table II). As for the controller, the Tustin approximation is used.

The fixed-point format of variables and coefficients is set based on a trial and test procedure. Several simulations, with different formats, are executed in order to find the smallest format which respects the control specifications for all quantities in terms of dynamic range and accuracy. The used representation is labelled as $s[(i+f)/Qf]$ for signed data. $(i+f)$ is the total data size, i is the number of bits of the integer part and f is the number of bits of the fractional part. In the case of a Sw implementation, the format is imposed by the chosen microcontroller (here, 32b). The chosen formats for the controller and the EKF modules are summarized in Table III.

D. Discrete-Time and Fixed-Point Validation

After having developed the digital controller, the designer has to make a final functional verification by simulating the developed algorithm in the discrete-time and fixed-point domain with the help of MATLAB/Simulink tools. In this case, the obtained simulation results are shown in Fig. 5, where the waveforms of the measured and estimated rotor speed and position are given.

V. ARCHITECTURAL EXPLORATION

In order to efficiently explore different architectures, we propose the following steps.

A. Characterization and Performance Estimation of Functional Modules

The difficulty of the architectural exploration is mainly linked to the search for an optimized Hw/Sw partitioning. This partitioning depends strongly on the number of involved modules and their corresponding granularity level. Thus, the use of a fine-grained population of modules, typically the arithmetic modules of the level-1 library (Fig. 3), expands the exploration domain but at the cost of a higher complexity of the optimization problem formulation. Indeed, the corresponding synchronous data flow graph (SDFG) can no longer be executed manually. In addition, the functional modules of level 3, which are the ones used by the control engineer for establishing the control strategy and the control specifications are no longer visible in this case. This leads to the loss of the physical meaning of the architecture. From the point of view of the authors, this loss is a real issue since the proposed design guidelines are think as a whole, allowing the control engineer himself to master the totality of the design process. Along this line, using the level-3 modules (Fig. 3) is a good compromise since the limited number of modules is more manageable and their physical meaning is preserved. In this context, the characterization of each functional module "M_i" is of prime importance. Fig. 6 presents the related metrics used for characterizing each block, both in hardware and in software, where

- A_i represents the consumed resources of module i, in the case of hardware implementation (6-b LUT & FF).
- H_i represents the used memory blocks 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 consumed hardware multiplier (DSP units) of module i, in the case of hardware implementation.
- I_i/O_i represents the number of inputs and outputs of module i.

TABLE IV
FEATURES OF CONTROL MODULES

Modules	y_i	A_i	HM_i	t_{hi} (clk cycles)	t_{si} (clk cycles)	H_i (Kbits)
M_0:Acquisition	--	262 LUTs 142 FFs	--	240	--	--
M_1:Voltage interface	1	15 LUTs 30 FFs	2	11	40	4
M_2:Jacobian matrix	0	68 LUTs 110 FFs	2	8	500	6
M_3:abc-dq	1	207 LUTs 184 FFs	2	27	380	44
M_4:abc-dq	0	207 LUTs 184 FFs	2	27	380	44
M_5:Prediction	1	314 LUTs 198 FFs	2	23	510	9
M_6:Compensator P1	0	1336 LUTs 704 FFs	8	101	4300	36
M_7:Compensator P2	0	1336 LUTs 704 FFs	8	101	1750	31
M_8:Compensator P3	0	1443 LUTs 1443 FFs	8	183	800	30
M_9:Compensator P4	0	1688 LUTs 1056 FFs	8	101	2210	42
M_{10}:Innovation	0	208 LUTs 220 FFs	2	25	525	10
M_{11}:P_PI	0	124 LUTs 160 FFs	2	21	256	5
M_{12}:abc-dq	0	207 LUTs 184 FFs	2	27	380	44
M_{13}:q-PI	1	80 LUTs 120 FFs	2	14	200	4
M_{14}:d-PI	0	80 LUTs 120 FFs	2	14	200	4
M_{15}:dq-abc	0	230 LUTs 155 FFs	2	25	380	44
M_{16}:SVM	--	390 LUTs 143 FFs	--	4	--	--

Table IV presents the set of the metrics of each functional module M_i. All the times are given as number of clock cycles. The exploitation of the concurrency between the functional modules can reduce the execution time. To this purpose, a parallelism variable "y" is introduced. This parameter depends on the scheduling and the data dependency between the different modules. The variable y is a vector of n-1 components, $y = [y_1, y_2 \dots y_{n-1}]$. Each of these components y_i is characterized by a binary value {0,1}. The component y_i is equal to 1 (resp. 0) when "M_i" can be executed in parallel (resp. in series) with "M_{i+1}."

Thus, in the case of parallel modules, the total execution time T_{ex} is the maximum execution time of the two modules. The sum is taken in the case of a serial implementation (see Fig. 7). 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 since the multiprocessors approach is not studied here.

B. SDFG Representation of the Control Algorithm

In order to prepare the partitioning task, the whole EKF-based sensorless control algorithm was translated into a SDFG. As shown in Fig. 8, the proposed SDFG is composed of a set of nodes and edges. The nodes present the functional modules

TABLE V
TIME/AREA PERFORMANCES (XILINX VIRTEX-5, XC5VSX50T)

	Hw implementation	Sw implementation
Used LUTs (6-bit LUTs)	8195 (25.1 %)	4058 (12.4%)
Used (FFs)	5790 (17.7%)	1877 (5.75%)
Used Block RAMs (Kbits)	0	319
DSP blocks (hardware multipliers)	53	4
Execution time	13.4µs	137 µs

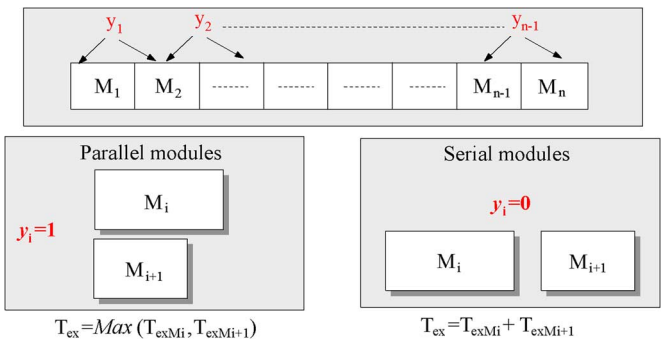

Fig. 7. Parallelism parameter.

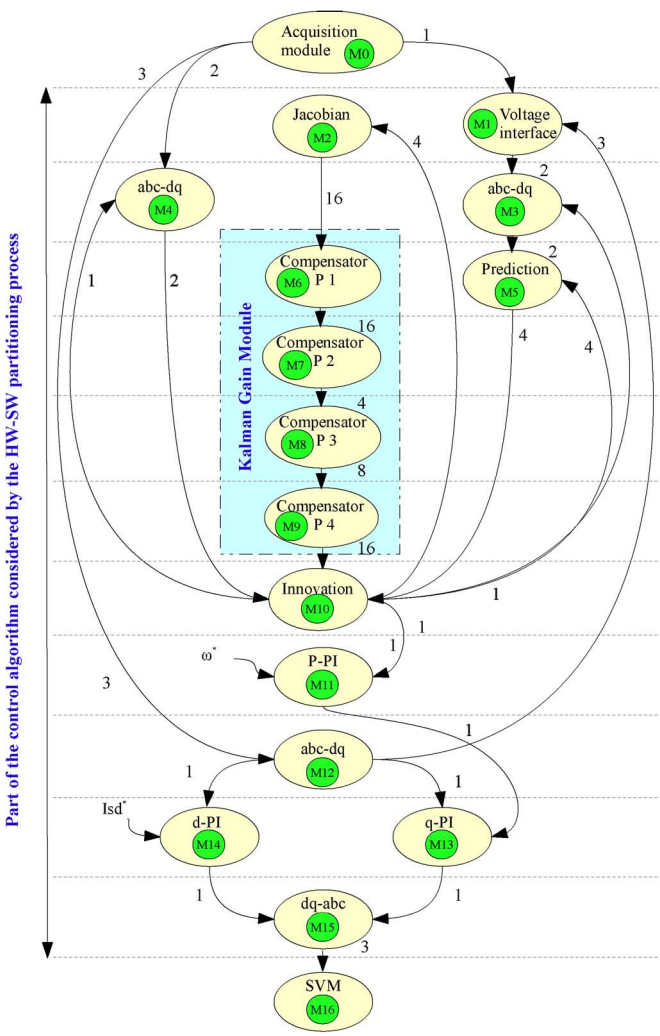

Fig. 8. SDFG of the EKF-based sensorless current controller.

"M_i" while the edges denote the data dependencies and the number of exchanged data between the different modules. To improve the execution time of the design, the concurrency between some functional modules is exploited providing an improvement of the execution time.

Each module is characterized by a number which is reflecting its natural order within the control process. Also, it has to be mentioned that the acquisition and the SVM modules (M_0 and M_{16}) are directly implemented in hardware. Indeed, the resolution and the functional performances required by these modules cannot be provided by software. Thus they are not considered in the Hw/Sw partitioning process. Consequently, the functional modules considered by the Hw/Sw partitioning process are modules M_1 to M_{15}.

C. Hw/Sw Partitioning

Once the characterization of functional modules and the SDFG representation are performed, the designer can begin the Hw/Sw partitioning process. They are three possible types of implementations:

- full software implementation of the controller using the soft-core processor.
- full hardware implementation of the controller using only the FPGA fabric.
- mixed Hw/Sw implementation using the soft-core processor and the FPGA fabric.

An in-depth study of the full hardware implementation and of the software implementation has already been proposed [19], [20]. These architectures were designed based on a dedicated design methodology beginning from the system specifications up to the experimentations. Table V summarizes the obtained time/area performances of the full software and the full hardware implementation of the sensorless control algorithm. One can note that a full hardware design takes 25.1% of the FPGA resources and the software design uses 12.4%. This should be interpreted by the fact that 12.4% is occupied by the soft-core processor and its peripherals. As for the execution time, the full hardware controller has an execution time of 13.4 μs and the full software controller has an execution time of 137 μs. It should be noted that 5 μs of the execution time of the Sw implementation is devoted to the format adaptation of data exchanged between Sw/Hw and Hw/Sw parts and the context switch. This latter presents the computing time needed to store and restore the state of the processor.

The obtained results demonstrate that a full hardware implementation provides a significant reduction of the execution time but it requires the use of much more resources. This is related to the complexity of the algorithm. A compromise is the partitioning of functional modules between the modules to be executed in software and those to be executed in hardware. This allows the controller to reach the expected level of performances

using only a reasonable fraction of the available internal resources. For the remainder, the interest of Hw/Sw partitioning to find relevant solutions is demonstrated.

Problem Formalization: The goal of the Hw/Sw partitioning procedure is to find an optimal architectural implementation among all of the design alternatives which satisfy the objectives regarding the considered constraints. In this case, these constraints are expressed in terms of control performance requirements and consumed FPGA resources. Hence, this Hw/Sw partitioning consists in a multi-objectives optimization problem. In order to formalize it, and in addition to the variables defined in Section V-A, let us define the following variables:

- $M = \{M_1, M_2, \ldots M_n\}$: the group of the n functional modules (n is here equal to 15).
- $(x_1, x_2, \ldots x_n)$: the binary individuals that represent the solution of the partitioning problem where $x_i \in \{1, 0\}$. $x_i = 1$ (resp. $x_i = 0$) means that the ith module has to be implemented in Hw (resp in Sw).
- $A_{\mu p}$: the hardware resources consumed by the Microblaze soft-core, the associated bus and the peripherals. Note that the Sw implementation of one module requires the whole use of $A_{\mu p}$.
- Area: the total consumed hardware resources.
- HM_{blocks}: the total consumed hardware multipliers (DSP units).
- $HM_{\mu p}$: the total consumed hardware multipliers (DSP units) by the processor.
- Memory: the total memory use.
- T_{ex}: the total execution time.

Area, HM_{blocks}, Memory, and T_{ex} are derived from expression (13) and Table III. $\text{A}_{\mu p}$ and $\text{HM}_{\mu p}$ are given in Table IV (Sw implementation column). Thus, for given constraints (available area, memory, multiplier blocks, and maximum allowable execution time), the optimization problem can be formalized as given by (13), shown at the bottom of the page, where S, H, and HM are, respectively, the area, memory size, and number of hardware multipliers available in the FPGA (Virtex-5, XC5VSX50T). T_{alg} corresponds to the maximum allowable execution time. This time corresponds to the maximum time delay that satisfies the control performance requirements (bandwidth and stability margin).

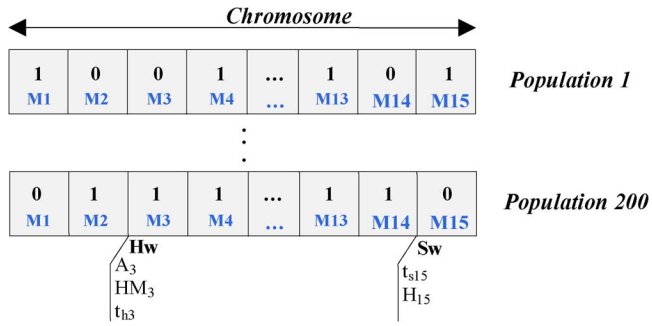

Fig. 9. Chromosome representation.

NSGA-II Algorithm: To deal with this multi-objectives optimization problem, the NSGA-II has been adopted. Such a genetic algorithm has been successfully used for solving several optimization problems in the field of VLSI systems such as the placement/routing, digital signal processor (DSP) code optimization, the area–time optimization of FPGA design [14]. In our case, the NSGA-II is used offline for the Hw/Sw partitioning under multiple objectives and constraints. It aims to find out the corresponding Pareto front. The Pareto front of a multi-objective optimization problem corresponds to the set of the nondominated solutions after completion of the optimization algorithm. These Pareto solutions present a tradeoffs (time/area): the cost of one solution cannot be improved in one dimension without being made worse in another. The NSGA-II is

$$\text{Objectives}: \quad \text{Minimize} \begin{cases} Area = \sum_{i=1}^{n} x_i \cdot A_i + \left(1 - \prod_{i=1}^{n} x_i\right).A_{\mu p} \\ HM_{\text{blocks}} = \sum_{i=1}^{n} x_i \cdot \text{HM}_i + \left(1 - \prod_{i=1}^{n} x_i\right). \cdot HM_{\mu p} \\ \text{Memory} = \sum_{i=1}^{n} (1 - x_i) \cdot H_i \\ T_{\text{ex}} = \sum_{i=1}^{n-1} \begin{bmatrix} y_i \cdot x_i \cdot x_{i+1} \cdot \max(t_{hi}, t_{\text{hi}+1}) \\ y_i \cdot x_i \cdot (1 - x_{i+1}) \cdot \max(t_{si}, t_{si+1}) \\ y_i \cdot (1 - x_i) \cdot x_{i+1} \cdot \max(t_{si}, t_{\text{hi}+1}) \\ y_i \cdot \overline{y_{i-1}} (1 - x_i) \cdot (1 - x_{i+1}) \cdot \max(t_{si}, t_{si+1}) \\ \overline{y_i} \cdot \overline{y_{i-1}} \cdot x_i \cdot t_{\text{hi}} \\ \overline{y_i} \cdot \overline{y_{i-1}} (1 - x_i) \cdot t_{si} + \overline{y_{n-1}} \left[x_n \cdot t_{hn} + (1 - x_n) \cdot t_{sn}\right] \end{bmatrix} \end{cases} \quad (13)$$

$$\text{Constraints}: \quad Subject \begin{cases} \text{Area} < \%S \\ HM_{\text{blocks}} < HM \\ \text{Memory} < H \\ T_{\text{ex}} < T_{\text{alg}} \end{cases}$$

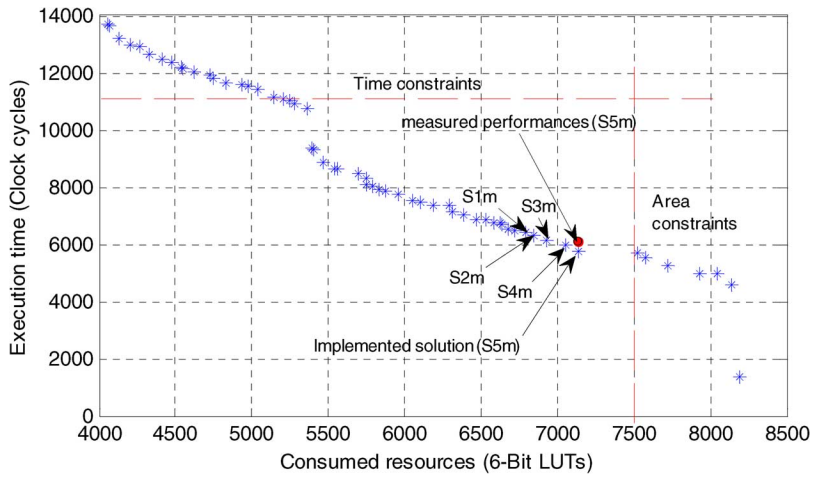

Fig. 10. Pareto-front of nondominated solutions.

characterized by a new ranking function. It classifies the candidate solutions considering all of the objectives. One important feature of the NSGA-II is the elitism which means that the best solutions within one population are conserved for the next generation. After that, the population random selection (mutation, crossover) will guarantee the diversity of the population. This procedure is iterated until finding the Pareto-optimal front. Thus, 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 $(m \cdot L)$ comparisons, where L is the number of populations and m is the number of objectives. This optimization algorithm is based on chromosome's representation. As shown in Fig. 9, each chromosome represents a specific partitioning solution. A chromosome is constituted of n genes, where n is the total number of modules to be implemented (here, n is equal to 15). Each gene x_i of the chromosome is a binary variable that represents the state of the corresponding module architecture. That is to say, when the ith gene is equal to 1 (resp. 0), the module M_i is implemented in Hw (resp. in Sw). From the genetic algorithm perspective, each chromosome is viewed as an individual of a given generation. The NSGA-II configuration was done considering the following parameters values: number of generations $= 200$, population $= 200$, crossover rate $= 1\%$, and mutation rate $= 90\%$.

Partitioning Results: The Hw/Sw partitioning tests were performed over the Xilinx FPGA (Virtex-5, XC5VSX50T) including a Microblaze soft-core. Fig. 10 presents the Pareto-Front of nondominated implementation solutions. The time taken to obtain the Pareto front for a population and a number of generations both equal to 200 is 1 mn, 38 s, and 68 ms. This was obtained with the following PC: Intel core (TM) i3 CPU 350 @2.27 GHZ processor, and 4 GB of RAM. The NSGA-II algorithm provides a space exploration covering a good number of feasible solutions. Thus, the designer can choose the appropriate allocation and scheduling which best suit his application. Considering an area/time-constrained exploration: the maximum allowable consumed resources (linked to the cost) are fixed here to 7500 six-input LUTs and the maximum execution time is set to maximum allowable time delay fixed by the control performances. The remaining SoC available resources will be used to integrate other functions such as monitoring communication process that are not presented here.

TABLE VI
EXAMPLES OF OPTIMIZATION SOLUTIONS

	Binary individuals (x1 … x15)	A	T_{ex}	Memory	25x18 HM	Speedup factor
S_{1m}	010111000101111	6788	6406	195	27	2.14
S_{2m}	111111000110011	6847	6311	151	29	2.17
S_{3m}	111111000110111	6927	6131	147	31	2.24
S_{4m}	111111000111011	7054	5959	107	31	2.30
S_{5m}	**111111000111111**	**7134**	**5768**	**103**	**33**	**2.38**

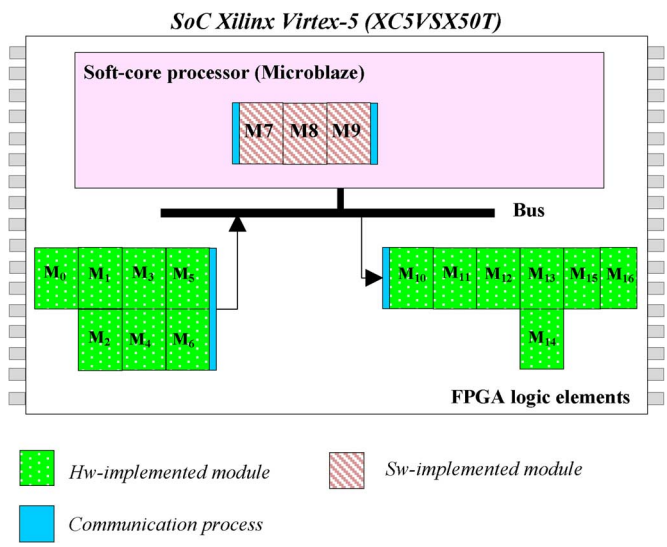

Fig. 11. Scheduling diagram of the solution "$S_{5\,\mathrm{m}}$".

Table VI presents examples of nondominated solutions. In Fig. 11 is shown the scheduling diagram of the solution $S_{5\,\mathrm{m}}$. Among all the nondominated solutions that still satisfy the area constraint, $S_{5\,\mathrm{m}}$ is the fastest. $S_{5\,\mathrm{m}}$ has a speedup factor of 2.38. The speedup factor is the ratio between the execution time of a Hw/Sw solution with regard to the execution time of the full

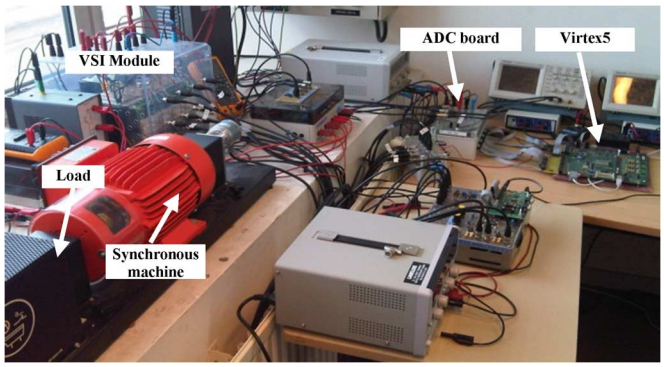

Fig. 12. Prototyping platform.

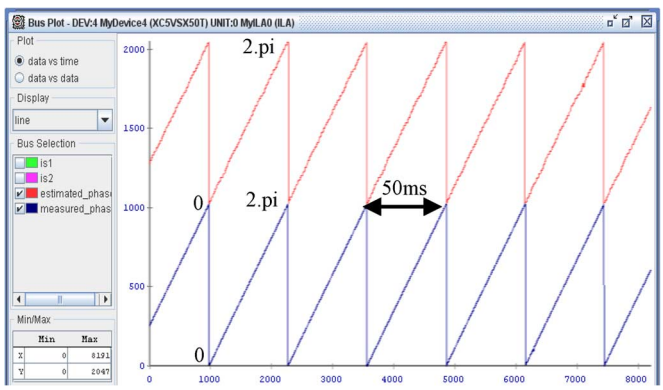

Fig. 13. Experimental results—waveforms of the estimated (red curve) and the measured rotor position (blue curve).

TABLE VII
COMPARISON BETWEEN ACTUAL AND PREDICTED HW/SW ARCHITECTURES

	Estimated area (LUTs)	Actual area (LUTs)	**Error (%)**	Estimated T_{ex} (Clk cycles)	Actual T_{ex} (Clk cycles)	**Error (%)**
S_{3m}	6927	7075	**2.15**	6131	6468	**5.6**
S_{4m}	7054	7199	**2.05**	5959	6286	**5.49**
S_{5m}	7134	7294	**2.24**	5768	5948	**3.12**

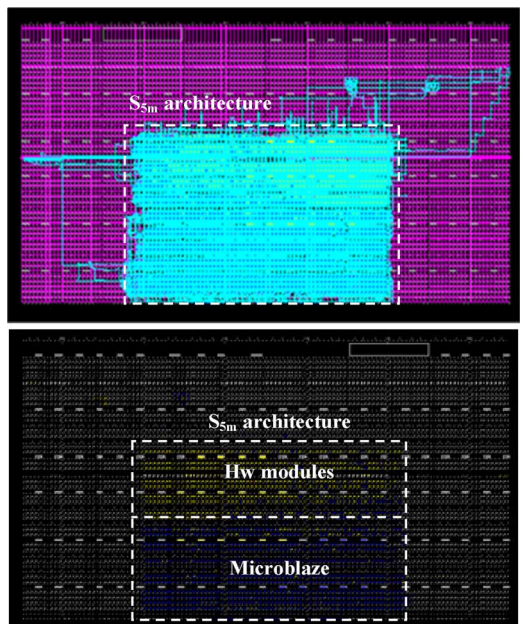

Fig. 14. $S_{5\,m}$ synthesized architecture.

Sw-implementation, taken as baseline. It can be also noticed from Table VI that there are always one or more computationally intensive modules (M6 to M9) executed in Sw. Indeed, these modules, which are used in the Kalman gain computation (Fig. 4), are consuming an important number of LUTs and hardware multipliers when implemented in Hw. Thus, in order to satisfy the resource constraint, the NSGA-II algorithm is placing some of these greedy modules in Sw while keeping the global execution time below T_{alg}.

VI. EXPERIMENTAL VALIDATION

Experimental validation of the solution "$S_{5\,m}$" has been carried out with the platform presented in Fig. 12. Fig. 13 shows the good matching of the estimated and the measured angular position at 500 rpm. The characteristics of the motor and its load are given in Table VII.

However, in this work, the most important validation concerns the quality of the performance prediction of the nondominated solutions obtained by the NSGA-II algorithm. Indeed, as shown in (13), the objective functions of the Hw/Sw partitioning optimization problem are strongly dependent of the functional module characteristics (Table III). If these characteristics are inaccurate, then the architectures proposed by the NSGA-II algorithm will be questionable. Conversely, if these characteristics are accurate, then it means that the NSGA-II has worked with correct data and as a consequence the designer can be confident in the architectural solutions proposed by the NSGA-II algorithm. Along this line, Table VII presents for three nondominated solutions ($S_{3\,m}$ to $S_{5\,m}$) the comparison between their estimated performances directly derived from the results of the NSGA-II algorithm and their actual performances obtained after synthesis.

It can be seen from these results a good matching in terms of area since the maximum error is equal to 2.3%. The error is higher in terms of timing, reaching 5.6% for the $S_{3\,m}$ solution. However, this value is still acceptable knowing that the communication times between modules have not been taken into account in the prediction. $S_{5\,m}$ gives better results compared to $S_{3\,m}$ and $S_{4\,m}$ because it is a more homogeneous solution. Indeed, as can be seen on the genes of the solutions (Table VI), $S_{5\,m}$ is concentrating all its Sw modules which is not the case of the two other solutions. Thus, the number of Sw-to-Hw and Hw-to-Sw communications is reduced which implies a better prediction.

Finally, Fig. 14 presents the $S_{5\,m}$ solution once implemented on the Virtex-5, the dotted line represents the area constraints in terms of LUTs (here 7500 LUTs). The compactness of the architecture can be observed.

VII. CONCLUSION

Hardware/software codesign guidelines for SoC FPGA-based sensorless ac drive applications were proposed. The most

salient feature of this approach lies in an automatic architectural exploration of the controller which is based on the search of the optimal solutions in terms of Hw/Sw partitioning. To this purpose, the NSGA-II was used to solve the corresponding multi-objective optimization problem. Indeed, since the early stages of the optimization process both control specifications (bandwidth and stability margin) and FPGA limited internal resources (FPGA fabric, memory, and hardware multipliers) were integrated. Thus, the set of the nondominated architectural solutions (Pareto front) was then derived. Among these solutions, the fastest which still respect the area constraint was synthesized and tested on an experimental setup. The chosen control algorithm benchmark for this study was a sensorless speed drive of a synchronous motor based on an extended Kalman filter. In addition, an important effort was also made to test the accuracy of the proposed solutions given by the NSGA-II algorithm. To this purpose, up to three different nondominated solutions were synthesized and their actual time/area performances were compared to their estimated ones. These latter are the ones proposed by the NSGA-II algorithm. Results were found satisfactory since only a maximum error of 2.3% in area and a maximum error of 5.6% in time were obtained. The inclusion of the communication time between the different functional modules of the control algorithm could help to reduce even more the timing error. Taking into account of the communication time within the partitioning problem is one of the direct perspectives to this work. Another one is to integrate to the Hw/Sw partitioning other functions that are not directly related to the control of the drive like the interface with the supervisor and the diagnosis module.

REFERENCES

[1] B. K. Bose, "Neural network applications in power electronics and motor drives—An introduction and perspective," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 14–33, Feb. 2007.

[2] S. Bolognani, L. Tubiana, and M. Zigliotto, "Extended Kalman filter tuning in sensorless PMSM drives," *IEEE Trans. Ind. Electron.*, vol. 39, no. 6, pp. 276–281, Nov. 2003.

[3] Q. N. Le and J. W. Jeon, "Neural-network-based low-speed-damping controller for stepper motor with an FPGA," *IEEE Trans. Ind. Electron.*, vol. 57, no. 9, pp. 3167–3180, Sep. 2010.

[4] T. O. Kowalska and M. Aminski, "FPGA implementation of the multilayer neural network for the speed estimation of the two-mass drive system," *IEEE Trans. Ind. Inf.*, vol. 7, no. 3, pp. 436–445, Aug. 2011.

[5] L. Idkhajine, E. Monmasson, M.-W. Naouar, A. Prata, and K. Boullaga, "Fully integrated FPGA-based controller for synchronous motor drives," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4006–4017, Oct. 2009.

[6] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and W. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Electron.*, vol. 7, no. 2, p. 224, May 2011.

[7] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domains of FPGAs," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1810–1823, Aug. 2007.

[8] M. Finc and A. Zemva, "Profiling soft-core processor applications for hardware/software partitioning," *J. Syst. Architecture*, vol. 51, no. 5, pp. 415–329, May 2005.

[9] R. Lysecky and F. Vahid, "A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning," in *Proc. DATE'05 Conf.*, 2005, pp. 18–23.

[10] R. Obermaisser and P. Gutwenger, "Model-based development of MPSoCs with support for early validation," in *Proc. IEEE IECON'09 Conf.*, 2009, CD-ROM.

[11] K. Deb, A. Pratap, S. Argarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comp.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[12] R. P. Dick and N. K. Jha, "MOGAC: A multi-objective genetic algorithm for hardware-software cosynthesis of distributed embedded systems," *IEEE Trans. Comput.-Aided Des. (CAD) Integr. Circuits Syst.*, vol. 17, no. 10, pp. 920–935, Oct. 1998.

[13] M.-L. Vallejo and J.-C. Lopez, "On the hardware-software partitioning problem: System modeling and partitioning techniques," *ACM Trans. Design Autom. Electron. Syst.*, vol. 8, no. 3, pp. 269–297, Jul. 2003.

[14] M. Savage, Z. Salcic, G. Coghill, and G. Covic, "Genetic algorithm for codesign optimization of DSP systems in FPGAs," in *Proc. IEEE ICFPT*, 2004, pp. 291–294.

[15] S. Jovanovic and P. Poure, "Design of power electronic digital controller based on FPGA/SOC using VHDL-AMS language," in *Proc. IEEE ISIE*, 2007, pp. 2301–2306.

[16] I. E. Ormaetxea, J. Andreu, I. Kortabarria, U. Bidarte, I. Martínez de Alegría, E. Ibarra, and E. Olaguenaga, "Matrix converter protection and computational capabilities based on a system on chip design with an FPGA," *IEEE Trans. Power Electron.*, vol. 26, no. 1, pp. 72–287, Jan. 2011.

[17] V. Delli Colli, R. Di Stefano, and F. Marignetti, "A system-on-chip sensorless control for a permanent-magnet synchronous motor," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3822–3829, Nov. 2010.

[18] Y. S. Kung, R. F. Fung, and T. Y. Tai, "Realization of a motion control IC for X-Y table based on novel FPGA technology," *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 43–53, Jan. 2009.

[19] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless control for synchronous AC drive using an extended Kalman filter," *IEEE Trans. Ind. Electron.*, vol. 59, no. 10, pp. 3908–3918, Oct. 2012.

[20] I. Bahri, L. Idkajine, E. Monmasson, and M. A. Benkhelifa, "Optimal hardware/software partitioning of a system on chip FPGA-based sensorless AC drive current controller," *Trans. Math. Comput. Simulation*, 2012, accepted for publication.

Imen Bahri (M'13) was born in Tunisia in 1982. She received the Ing. and M.S degrees in electrical engineering from the National School Engineers of Tunis (ENIT), Tunis, Tunisia, in 2006 and 2007, respectively, and the Ph.D. degree from Cergy-Pontoise University, Cergy-Pontoise, France, in 2011.

She is currently an Assistant Professor with Paris-XI University, Orsay, France, and a member of the LGEP laboratory. Her current research interests include the control applications applied to automotive domain, the SoPC-based controllers, FPGA-based implementation of controllers, hardware/software (Hw/Sw) partitioning, and Hw/Sw codesign methodologies.

Lahoucine Idkhajine (M'10) was born in 1983 in Massa, Agadir, Morocco. He received the "Master Professionnel GEII" and Ph.D. degree from Cergy-Pontoise University, Cergy-Pontoise, France, in 2007 and 2010, respectively.

From October 2010 to August 2011, he was an Assistant Lecturer with Arts et Métiers ParisTech, Lille, France. He is currently an Assistant Professor with Cergy-Pontoise University, Cergy-Pontoise, France, and a member of the SATIE Laboratory (SETE team). His research interests are focused on the design and the implementation (using FPGA SoC devices) of controllers and observers for power electronics and drives applications.

Eric Monmasson (M'96–SM'06) received the Ing. and Ph.D. degrees from the Ecole Nationale Supérieure d'Ingénieurs d'Electrotechnique d'Electronique d'Informatique et d'Hydraulique de Toulouse (ENSEEIHT), Toulouse, France, in 1989 and 1993, respectively. \

He is currently a Full Professor and the Head of the Institut Universitaire Professionnalisé de Génie Electrique et d'Informatique Indutrielle (IUP GEII), University of Cergy-Pontoise, Cergy-Pontoise, France. He is also with the Systèmes et Applications

des Technologies de l'Information et de l'Energie laboratory (SATIE, UMR CNRS8029). He is the author or coauthor of three books and more than 150 scientific papers. His current research interests include the advanced control of electrical motors and generators and the use of FPGAs for energy control systems.

Prof. Monmasson was the chair of the technical committee on Electronic Systems-on-Chip of the IEEE Industrial Electronics Society (2008–2011). He is also a member of the steering committee of the European Power Electronics Association and the chair of the number one technical committee of the International Association for Mathematics and Computers in Simulation (IMACS). He was the general chair of ELECTRIMACS 2011 Conference. He is an associate editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Mohamed El Amine Benkhelifa received the electronics engineer degree from the U.S.T.O., Oran, Algeria, in 1987, and the Ph.D. degree in physics from Paris-V University, Paris, France, in 1992.

He is now an Assistant Professor of electrical engineering and computer science with the ETIS Laboratory, University of Cergy-Pontoise, Cergy-Pontoise, France. His research interests include coarse-grained and fine reconfigurable arrays, domain-specific FPGAs, reconfigurable computing architectures, and operating systems for reconfigurable architectures.