FPGA Design Methodology for Industrial Control Systems—A Review

Eric Monmasson, *Senior Member, IEEE*, and Marcian N. Cirstea, *Senior Member, IEEE*

***Abstract*—This paper reviews the state of the art of field-programmable gate array (FPGA) design methodologies with a focus on industrial control system applications. This paper starts with an overview of FPGA technology development, followed by a presentation of design methodologies, development tools and relevant CAD environments, including the use of portable hardware description languages and system level programming/design tools. They enable a holistic functional approach with the major advantage of setting up a unique modeling and evaluation environment for complete industrial electronics systems. Three main design rules are then presented. These are algorithm refinement, modularity, and systematic search for the best compromise between the control performance and the architectural constraints. An overview of contributions and limits of FPGAs is also given, followed by a short survey of FPGA-based intelligent controllers for modern industrial systems. Finally, two complete and timely case studies are presented to illustrate the benefits of an FPGA implementation when using the proposed system modeling and design methodology. These consist of the direct torque control for induction motor drives and the control of a diesel-driven synchronous stand-alone generator with the help of fuzzy logic.**

***Index Terms*—Design methodologies, field-programmable gate arrays (FPGAs), industrial control systems, programmable architectures, system-on-a-chip (SoC), (Vhsic) hardware description language (VHDL).**

I. Introduction

FAST PROGRESS of very large scale integration (VLSI) technology and electronic design automation (EDA) techniques in recent years has created an opportunity for the development of complex and compact high-performance controllers for industrial electronic systems [1]. Nowadays, the design engineer is using modern EDA tools to create, simulate, and verify a design and, without committing to hardware, can quickly evaluate complex systems and ideas with very high confidence in the "right first time" correct operation of the final product.

Speed performance of new components and flexibility inherent of all programmable solutions give today many opportunities in the field of digital implementation for industrial control systems. This is particularly true with software solutions such as microprocessors or digital signal processors (DSPs) [3]. However, specific hardware technologies such as field-programmable gate arrays (FPGAs) can also be considered as an appropriate solution in order to boost the performance of controllers. Indeed, these generic components combine low-cost development (owing to their reprogrammability), use of convenient software tools, and more and more significant integration density [4]–[8]. FPGA technology is now considered by an increasing number of designers in various fields of application such as wired and wireless telecommunications [9], and image and signal processing [10], [11], where the always more demanding data throughputs take advantage of the ever increasing density of the chips. Still, more recently, other application fields are in growing demand, such as medical equipment [12], robotics [13]–[15], automotive [16], and space and aircraft embedded control systems [17]. For these embedded applications, reduction of the power consumption [18], thermal management and packaging [19], reliability [20], and protection against solar radiation [21] are of prime importance. Finally, industrial electrical control systems are also of great interest because of the ever-increasing level of expected performance while at the same time reducing the cost of the control systems [22]. This last sector is particularly targeted by the case studies presented briefly in this review paper. Indeed, FPGAs have already been used with success in many different electric system applications such as power converter control (pulsewidth-modulation (PWM) inverters [23], [24], power-factor correction [25], multilevel converters [26], [27], matrix converters [28], [29], soft switching [30], [31], and STATCOM [32]) and electrical machines control (induction machine drives [33]–[39], switched reluctance machine (SRM) drives [40], motion control [41], [42], multimachines systems [43], neural network (NN) control of induction motors [44], fuzzy logic control of power generators [45], and speed measurement [46]). This is because an FPGA-based implementation of controllers can efficiently answer current and future challenges of this field. Among them, we can quote the following.

1) Decrease of the cost for at least three reasons. The use of an architecture based only on the specific needs of the algorithm to implement, the application of highly advanced and specific methodologies improving implementation time, also called "time to market," and the expected development in VLSI design that will allow integrating a full control system with its analog interface in a single chip, also called System-on-a-Chip (SoC).
2) Confidentiality, a specific architecture, integrating the know-how of a company, is not easily duplicable.

Manuscript received June 1, 2006; revised April 11, 2007.

E. Monmasson is with the SATIE, Cergy-Pontoise University, 95031 Cergy-Pontoise, France (e-mail: eric.monmasson@u-cergy.fr).

M. N. Cirstea is with the Department of Design and Technology, Anglia Ruskin University, Cambridge CB1 1PT, U.K. (e-mail: marcian.cirstea@anglia.ac.uk).

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/TIE.2007.898281

3) Embedded systems with many constraints as in aircraft applications, like limited power consumption, thermal consideration, reliability, and single event upset protection.
4) Improvement of control performance. For example, execution time can be dramatically reduced by designing dedicated parallel architectures, allowing FPGA-based controllers to reach the level of performance of their analog counterparts without their drawbacks (parameter drifts, lack of flexibility). In addition, an FPGA-based controller can be adapted in runtime to the needs of the plant by dynamically reconfiguring it. These points will be discussed further in Section V.

This paper aims to provide an overview of the use of FPGAs in industrial control systems. Generic FPGA architectures and a computer-aided design (CAD) environment characterizing them are presented. Benefits of using portable hardware description languages (HDLs) are discussed; then, the holistic approach is explained. It extends the traditional use of high-level programming languages and HDLs [2] to encompass the holistic modeling of industrial electronic systems. The outcome is a design environment that allows all functional aspects of the system to be considered simultaneously, therefore increasing the determinism of the system, minimizing the response time, and maximizing operational performance in order to achieve high efficiency and power quality while simultaneously allowing the rapid prototyping of digital controllers on FPGA hardware development platforms.

Major design rules are given, consisting of control algorithm refinement, application of a reuse methodology, which allows capitalizing the design efforts, and optimization of the modules in terms of performance with the help of the Algorithm Architecture “Adequation” (A^3). The authors then analyze, in the present industrial environment, the contributions and the limits of using FPGAs in electrical system controllers. A short survey on intelligent FPGA-based controllers is also presented.

Finally, two case studies are discussed to illustrate benefits of an FPGA implementation when using the proposed design methodology: 1) direct torque control (DTC) system for induction motor and 2) fuzzy logic digital controller for a diesel-driven stand-alone power generator.

II. Description of FPGAs and Their Development Tools

A. FPGA Generic Architecture Description

FPGAs belong to the wide family of programmable logic components [4]–[8]. An FPGA is defined as a matrix of configurable logic blocks (CLBs) (combinatorial and/or sequential), linked to each other by an interconnection network which is entirely reprogrammable. The memory cells control the logic blocks as well as the connections so that the component can fulfill the required application specifications. Several configurable technologies exist. Among them, only those that are reprogrammable (Flash, EPROM, SRAM) are of interest since they allow the same flexibility as that of a microprocessor. Therefore, the rest of this paper will discuss only the SRAM-based FPGA technology [6], [7], which is by far the most widespread [47]. However, the Flash-based technology [8], although it does not allow the same number of reconfiguration cycles by an order of magnitude, it is of interest for some stringent niche applications such as space and aircraft industries. Indeed, Flash technology preserves the configuration of the FPGA when the power is off, and as a consequence, the chip is ready to operate as soon as it is powered up. The generic architecture of an SRAM-based FPGA is presented in Fig. 1 [48].

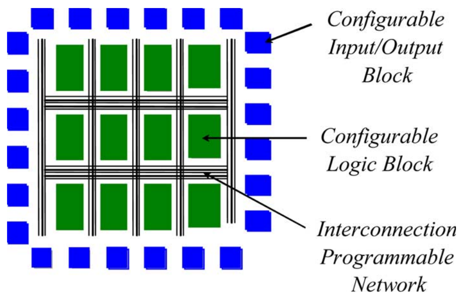
Configurable Input/Output Block
Configurable Logic Block
Interconnection Programmable Network

Fig. 1. Generic architecture of an FPGA.

The most recent FPGAs are produced using a 65-nm copper process. Their density can reach more than 10 million equivalent gates per chip with clock system frequencies of more than 500 MHz. However, it is important to note that this kind of information is only accurate for a short while as technology continues to move forward. The two main FPGA manufacturers are Altera and Xilinx [6], [7].

The FPGA generic architecture is composed of a matrix of CLBs, where the number of rows and columns is now reaching, for the largest devices, 192×116. This matrix core is bordered by a ring of configurable input/output blocks (IOBs), whose number can reach 1000 user IOBs. Finally, all these resources communicate among themselves through a programmable interconnection network.

More recently, it has also been observed inside these architectures the introduction of some dedicated blocks such as RAM, DSP accelerators (hardwired multipliers with corresponding accumulators, high-speed clock management circuitry, and serial transceivers), embedded hard processor cores such as PowerPC or ARM [6]–[8], and soft processor cores such as Nios [6] or Microblaze [7], [70]. Also, very interesting for control applications is the recent integration of an analog-to-digital converter in the fusion component from Actel [8]. However, this SoC trend does not replace the former generic architecture, but it can be seen as a complement to this original matrix.

1) Configurable Logic Blocks (CLBs): Their structures include two, four, or more logic cells, also called logic elements. The structure of a logic cell, which can be considered as the basic grain of the FPGA, is presented in Fig. 2.

It consists of a four-bit lookup table (LUT), which can be configured either as a (16×1) ROM, RAM, or a combinatorial function. A carry look-ahead data path is also included in order to build efficient arithmetic operators. Finally, a D-type flip-flop, with all its control inputs (synchronous or asynchronous set/reset, enable), allows registering the output of the logic cell. Such architecture corresponds to a microstate machine, since

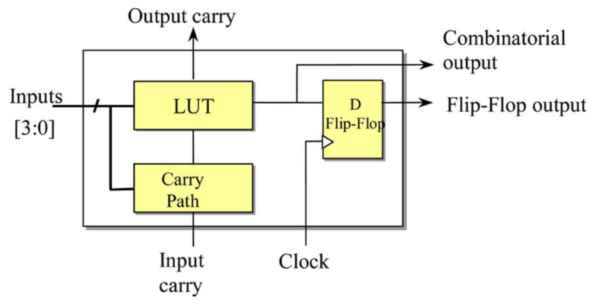

Fig. 2. Logic cell structure.

the registered output can be configured as an input of the same logic cell.

B. HDLs and FPGAs

Originally, FPGAs were only used to integrate glue logic usually devoted to TTL basic logic circuits. Applications were described with the help of simple CAD schematic tools. Today, FPGAs are more and more used to implement complex functions. For example, it is not unusual to implement in a single FPGA a complete digital system including an Arithmetic Logic Unit (ALU), memories, communication units, and so on.

This evolution has its origin in the recent advances in VLSI, but it is also due to the development of appropriate design tools and methods, which were initially reserved to the world of the application specific integrated circuits (ASICs). These tools are mostly based on HDLs such as very high-speed integrated circuits (Vhsic) HDL (VHDL) [2], [49] or Verilog [50]. The existence of IEEE standards [51] has spread the use of HDLs and has allowed the creation and the development of high-performance CAD tools in the field of microelectronics. Thus, the designer can take advantage of HDLs to build his own circuit by using a hierarchical and modular approach defined at different levels of abstraction using the design "top-down methodology" [52], [53]. The corresponding design flow is partitioned into the following four steps:

1) system level, where specifications of the circuit are given;
2) behavior level that consists in the algorithmic description of the circuit;
3) register transfer level (RTL), where the circuit is described in terms of its components;
4) physical level, where the circuit is physically described by taking into account the target hardware characteristics.

At each level of abstraction, the future integrated circuit is described in HDL, such as behavioral VHDL or synthesized VHDL. This last description gives an exact representation of the operators and variables of the final circuit.

In order to simulate and validate the digital circuit's functionality, various test benches are written and executed. Moreover, owing to the advent of analog HDLs such as Spectre HDL, VHDL-A, and VHDL-AMS [54], it is also possible to simulate at each level of abstraction the functionality of the circuit while taking into account its analog environment [55]. Another promising approach is the holistic one that promotes the use of a unique description language during the whole

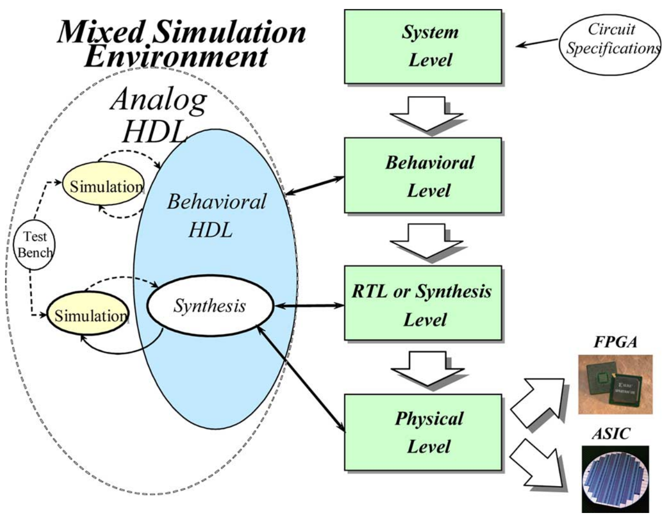

Fig. 3. Top–down design approach.

development procedure [56]. This will be described in more details in the next section. Fig. 3 presents the hierarchic flow of the top–down design method and its HDL model environment. Recently, FPGA manufacturers [6], [7] have designed software packages that enable both the simulation and the automatic translation into hardware of a design. Such software runs inside the Matlab-Simulink environment for example.

Simulation results are "bit and cycle accurate." This means results that are seen in a Simulink simulation exactly match those produced by hardware implementation. Such an approach offers an FPGA-based rapid prototyping platform [57]. It should be mentioned that the concept of automatic code generation has already been applied with success to DSP processors [58]. No doubt that this kind of solution will be more and more utilized in the near future for a rapid evaluation of new control algorithm performance. However, this approach is, so far, still limited to the applications that do not require the use of complex sequencers. Indeed, control units are still difficult to achieve with the proposed toolboxes [59]. As a consequence, the resulting hardware architectures are not area-optimized, a fact not acceptable in an industrial approach.

III. Integrated System Modeling and Design

Traditionally, mathematical models have been developed to evaluate the functionality of global engineering systems. However, the practical development of each part of the system needs then to be separately addressed. This often involves the use of other CAD tools and/or different software platforms, with the design itself being developed in a different environment. Recent advance in CAD methodologies/languages has brought the functional description of design and practical hardware implementation closer. System level modeling languages (such as Handel-C, System-C) and HDLs (such as VHDL, Verilog) enable the underpinning mathematical description and the electronic design implementation to be simultaneously addressed in a unique environment, supported by a range of major computer-aided engineering platforms. Synthesis tools can compile such designs into a variety of target technologies.

A holistic system level approach to the design and development of an electronic system enables a top–down design methodology, which begins with modeling an idea at an abstract level, and proceeds through the iterative steps necessary to further refine this into a detailed system. A test environment is developed early in the design cycle. As the design evolves to completion, the language is able to support a complex detailed digital system description, and the test environment will check in compliance with the original specification. Concepts are tested before investment is made in hardware/physical implementation. In terms of holistic modeling of complex electronic systems, the system level modeling languages offer advantages such as the following:

1) simultaneous consideration of the mathematical aspects of engineering systems (functional/behavioral description) and the detailed electronic hardware design, in the same unique environment, normally supported by a range of CAD platforms;
2) ability to handle all levels of abstraction, the system can be simulated as an overall model during all stages of the electronic controller design, which can be subsequently targeted for SoC silicon implementation;
3) fast implementation and relatively short time to market;
4) easy hardware implementation of artificial intelligence (AI);
5) versatile reusable models/design modules are generated, in accordance with modern principles of design reuse.

Simulation results are valuable to check the behavior of a model, but on many occasions, it is the hardware validation of a controller that provides significant information before the decision is taken to invest in an ASIC. The cheapest and fastest way to validate the design of an optimized digital controller is via a prototype board containing reprogrammable devices such as FPGAs. This shortens the time to correct any design problem, and it ensures an error-free design before permanent ASIC implementation. The prototype board can also be used for the hardware testing of other system components. A modern hardware-in-the-loop testing approach is also facilitated by this environment, allowing effective testing of circuit designs. This method uses a hardware-in-loop simulator (HILS) that uses the outputs of the circuit under test as inputs and produces as outputs the signals that need to be fed to the circuit under test as inputs. These signals are similar with those given by the subsystem replaced by the HILS in real-time operation. More on HILS can be found in [60]–[62].

The DK4 design suite from Celoxica, for example, allows Handel-C (high-level language similar with C) functional modeling of an electronic system. Handel-C produces an electronic design interchange format (EDIF) output when compiling the design for the hardware target. The Xilinx placement and routing tools are used to translate the EDIF format into hardware layout, enabling rapid hardware implementation onto development boards containing FPGAs. The compiler can also generate HDL format code such as VHDL, allowing combinations with other hardware elements in SoC designs. Portability without design modification of the implemented system on different PLD/FPGA/ASIC hardware target is provided using platform abstraction layer application programming interface. Thus, Handel-C can be used as a modeling tool, and then, Xilinx integrated design environment [6] enables FPGA real-time analysis.

The general benefits of holistic modeling, combined with the advantages of HDLs and FPGAs, enable novel complex but fast classical/neural/fuzzy FPGA controllers, with industrial applications, to be modeled, simulated, and evaluated with efficient use of resource.

IV. FPGA-Based Controller Design Rules

FPGA technology allows developing specific hardware architectures within a flexible programmable environment. This specific feature of the FPGAs gives designers a new degree of freedom comparing to microprocessor implementations, since the hardware architecture of the control system is not imposed *a priori*. However, in many cases, the development of this architecture is rather intuitive and not adapted to the implementation of more and more complex algorithms. Thus, in order to benefit from the advantages of the FPGAs and their powerful CAD tools, the designer has to follow an efficient design methodology. Such a methodology rests on three main principles: the control algorithm refinement, the modularity, and the best suitability between the algorithm to implement and the chosen hardware architecture. These three concepts are detailed thereafter.

A. Algorithm Refinement

Algorithm refinement is a necessary step when designing with FPGAs. It is possible to implement floating-point arithmetic on FPGAs [63], but the resources used are not optimized in this case because of FPGA sea-of-logic-cells architectures (see Fig. 2). Therefore, in order to reduce cost, manufacturers require from end-users to design controllers using fixed-point arithmetic. In this context, cost efficient architectures must result from a balance between control performances to respect and complexity of the hardware architecture to minimize. This leads to formulate two work directions:

1) Simplification of the Computation: Many authors, particularly in the early days of FPGAs, when the density of the chips was limited, proposed smart solutions to avoid including greedy operators like multipliers in their designs. Amongst the most commonly used techniques of simplification, CORDIC can be mentioned, an acronym for COordinate Rotation DIgital Computer [64]. CORDIC is a very efficient algorithm, which is only based on adders/subtractors and shifters for computing a wide range of trigonometric, hyperbolic, linear and logarithmic functions. Another interesting family of algorithms is the distributed arithmetic one [65], that can make extensive use of LUTs, which makes it ideal in implementing DSP functions in LUT-based FPGAs.

Finally, as explained thereafter, the designer can also take advantage in remodeling the target algorithm in order to reduce the number of operations to be implemented.

In order to illustrate these design rules, the authors are proposing, as example, a simple function to be implemented. It consists of an (a, b, c) to (d, q) transformation for three-phase electrical systems. The coordinate transformation is used to transform the actual quantities of a three-phase electrical system $(x_{\mathrm{a}}, x_{\mathrm{b}}, x_{\mathrm{c}})$ into a dq reference frame that is rotating at an arbitrary angle θ while keeping the instantaneous power equivalent. It gives

$$\begin{bmatrix} x_{\mathrm{o}} \\ x_{\mathrm{d}} \\ x_{\mathrm{q}} \end{bmatrix} = \sqrt{\frac{2}{3}} \cdot \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ \cos(\theta) & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ -\sin(\theta) & -\sin(\theta - 2\pi/3) & -\sin(\theta + 2\pi/3) \end{bmatrix} \cdot \begin{bmatrix} x_{\mathrm{a}} \\ x_{\mathrm{b}} \\ x_{\mathrm{c}} \end{bmatrix}. \quad (1)$$

By making the assumption that the studied three-phase system is balanced (no zero-sequence component), the transformation can be simplified and expressed as

$$\begin{bmatrix} x_{\mathrm{d}} \\ x_{\mathrm{q}} \end{bmatrix} = \sqrt{2} \cdot \begin{bmatrix} -\sin(\theta - 2\pi/3) & \sin(\theta) \\ -\cos(\theta - 2\pi/3) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} x_{\mathrm{a}} \\ x_{\mathrm{b}} \end{bmatrix}. \quad (2)$$

This first level of simplification allows reducing the number of operations to be implemented. The former expression is then converted into an n-bits fixed-point format. This format must be the result of a compromise between the required computing accuracy and the available hardware resources, as it can be seen later on. It gives

$$\begin{bmatrix} X_{\mathrm{d}} \\ X_{\mathrm{q}} \end{bmatrix} = \begin{bmatrix} -A_{11}(\theta) & A_{12}(\theta) \\ -A_{21}(\theta) & A_{22}(\theta) \end{bmatrix} \cdot \begin{bmatrix} X_{\mathrm{a}} \\ X_{\mathrm{b}} \end{bmatrix} \quad (3)$$

where each X n-bit signed fixed-point value is equal to

$$x = X \cdot Q_x \quad \text{with} \quad Q_x = \frac{x_{\max}}{2^{n-1}}. \quad (4)$$

The scale factor Q_x has to be selected with relevance by the designer so as to avoid overflow errors and in the same time keeping an acceptable dynamic range. Besides, during the conversion process, the designer can also simplify the implemented equations with an adequate choice of scale factors [34]. In this case, the original square root factor has disappeared, simplifying once again the computations. Notice also, the regularity of the operations to be executed, $A_{\mathrm{ij}}(\theta)$ variables are all sine functions, and both dot products (one per row) have exactly the same computing structure. This property gives the designer more possibilities of factorization when building the final architecture (Section IV-C).

2) Search for Optimized Fixed-Point Formats: As just mentioned earlier, when developing designs with FPGAs, a search for the best tradeoff between the size of the fixed-point format of each control variable and the respect of the control specifications is needed. To this purpose, a methodology is presented in [66], which is based on the control system L1 or l1 norms for computing the appropriate number of bits to represent each quantity of a controller (coefficients and variables). This methodology is applied with success to the implementation of a magnetic bearing FPGA-based control system. In this example, it is also proved that a delta form realization requires less hardware than a shift-form realization and provides a closer approximation to the original analog compensator.

In order to represent all the data values with a sufficient computation accuracy, Menard and Sentieys propose in [67] a methodology for an automatic determination of the fixed-point specification. First, the dynamic range of each data of the control algorithm is evaluated with the help of the interval arithmetic theory to define the minimal number of bits needed to represent the data integer part. Then, the accuracy is evaluated on the basis of an analytical approach. In the digital signal processing domain, the most commonly used criterion in evaluating the fixed-point specification accuracy is the signal-to-quantization noise ratio (SQNR). The originality of this approach is that it proposes an analytical evaluation of the SQNR expression for linear systems and nonrecursive nonlinear systems.

B. Design Methodology Based on Reuse Modules

For complex designs, modular conception is generally used to reduce the design cycle. This methodology is based on hierarchy and regularity concepts. Hierarchy is used to divide a large or complex design into subparts called modules that are more manageable. Regularity is aimed at maximizing the reuse of already designed modules [68].

With the increasing progress of CAD tools, the improvement in terms of development time reduction lies more in the capacity of the designer to know how to classify and reuse his model module than in a perfect knowledge of his CAD tools. Nowadays, the manufacturers and the designers of circuits even propose to recover in free [69] or restricted access [6]–[8] several design models, also called intellectual property (IP) modules. Besides, the complexity of some modules, such as processor cores [70], can be important. This design approach is based on the reusability of IP modules [71].

A module can be defined as an element of a library, available to the designer, which can be directly inferred without having to design it [52]. Therefore, the reuse methodology consists in selecting, throughout the synthesis process, elements of a library that are useful for the design in progress. These modules, extracted of the design flow, are distributed between various levels of abstraction. The procedure is very similar to those used in DSP developments, with soft macros [72]. Fig. 4 presents two types of reuse or IP module libraries that can be constituted, one at behavioral level and the other one at RTL level.

C. A^3 Methodology

To be efficient, the modular design approach must be based on reliable modules. However, in many cases, desired modules do not already exist, and they have to be built. It is therefore crucial, when designing them, to be helped by an efficient methodology that allows taking into account the numerous constraints of such systems.

The goal of the A^3 methodology, when applied to FPGAs, is to find out an optimized hardware architecture for a given application algorithm while satisfying time/area constraints [73]. “Adequation” is a French word meaning efficient matching. Note that it is different from the English word “adequacy” which involves only a sufficient matching. A^3 is based on graph

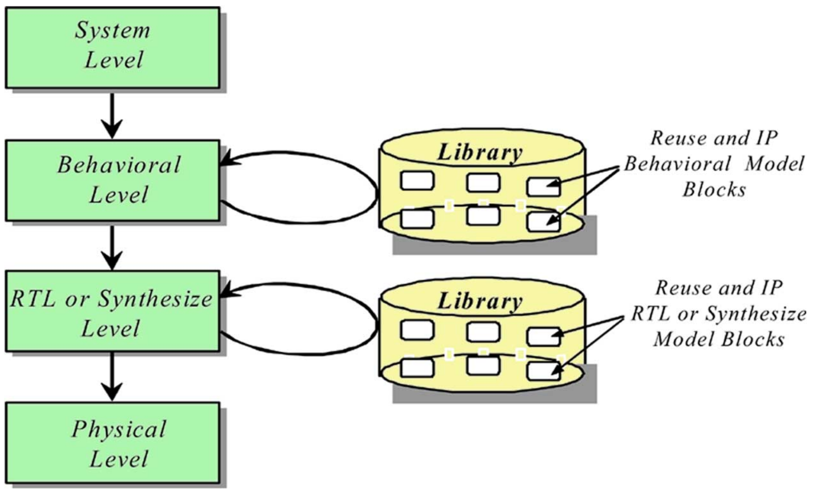
System Level
Behavioral Level
RTL or Synthesize Level
Physical Level
Library
Library
Reuse and IP Behavioral Model Blocks
Reuse and IP RTL or Synthesize Model Blocks

Fig. 4. Reuse and IP module libraries.

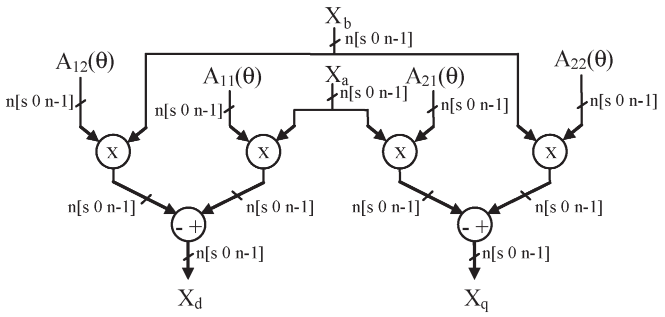

Fig. 5. DFG of the coordinate transformation algorithm.

models to exhibit both the potential parallelism of the algorithm and the available parallelism of the proposed architecture. The implementation is formalized in terms of graph transformations. The (a, b, c) to (d, q) transformation case is treated in order to illustrate the effectiveness of this methodology.

1) Data Flow Graph (DFG): Having finalized the algorithm refinement procedure, the DFG of the algorithm is directly derived from (3). The DFG establishes all the potential parallelisms of the algorithm. Fig. 5 shows the DFG corresponding to the coordinate transformation algorithm. Each node represents an operation, and each edge represents a dependence of data between two operations.

2) Design of the A^3 Optimized Architecture: The repetitive patterns of the DFG presented in Fig. 5 can then be advantageously factorized by using the A^3 methodology [73] in order to match the required hardware constraints. This leads to several data-path possibilities. Operations are now replaced by operators. Indeed, each operator included in a data path has a cost since it consumes hardware resources. In the case of the coordinate transformation algorithm, four different ALU data paths can be derived from the A^3 factorization process, as shown in Fig. 6.

Notice that, for simplicity reason, only the transformation and computation parts of this algorithm have been treated in this example. A^3 methodology could be also applied with success to sine function generation, also included in the (a, b, c) to (d, q) transformation algorithm. Then, the different data paths are compared taking into account their performance in terms of latency, speed, and size area in order to get the best tradeoff between all these constraints. In order to compare these data paths, corresponding architectures are synthesized in behavioral VHDL and implemented into the same target (Xilinx, SPARTAN 2 XC2s100 PQ208). As it is shown in Fig. 7, three resolutions were studied (12 bits, 14 bits, and 16 bits). As expected, ALU-1 is the greediest solution in terms of consumption of hardware resources, but at the same time, it is the fastest one. On the other hand, ALU-4 is the slowest data-path solution, but it presents the most optimized solution in terms of the consumption of hardware resources. In fact, it represents only one third of ALU-1 architecture. ALU-2 and ALU-3 present a compromise solution between computation time and hardware resources requirements. Therefore, the designer can choose the most suitable architecture solution according to the hardware requirements and expected control performance.

V. Contributions and Limits of FPGAs Used in Electrical System Controllers

A. Domain Use of the FPGAs

When designing industrial electronics circuits, several criteria have to be considered. Some of the most significant are the cost, the power consumption (essential in the case of embedded systems), the application performance, and above all, the suitability for the chosen hardware technology to match the requirements of the algorithm to implement.

This last point will be developed for the case of the control of electrical systems. As mentioned in the introduction, currently, the two main hardware solutions in implementing a controller are DSPs and FPGAs. Therefore, according to the nature of the algorithm to implement (i.e., its DFG), the designer has to choose between these two possibilities. The graph in Fig. 8 illustrates in a qualitative way the reasons of such a choice.

The x-axis of this graph represents timing constraints of the algorithm. These constraints mainly rely on the type of data dependence. The higher this dependence is, the more sequential the algorithm is. It is then obvious that software solution (DSPs) is perfectly adapted to this case. On the other hand, if the DFG

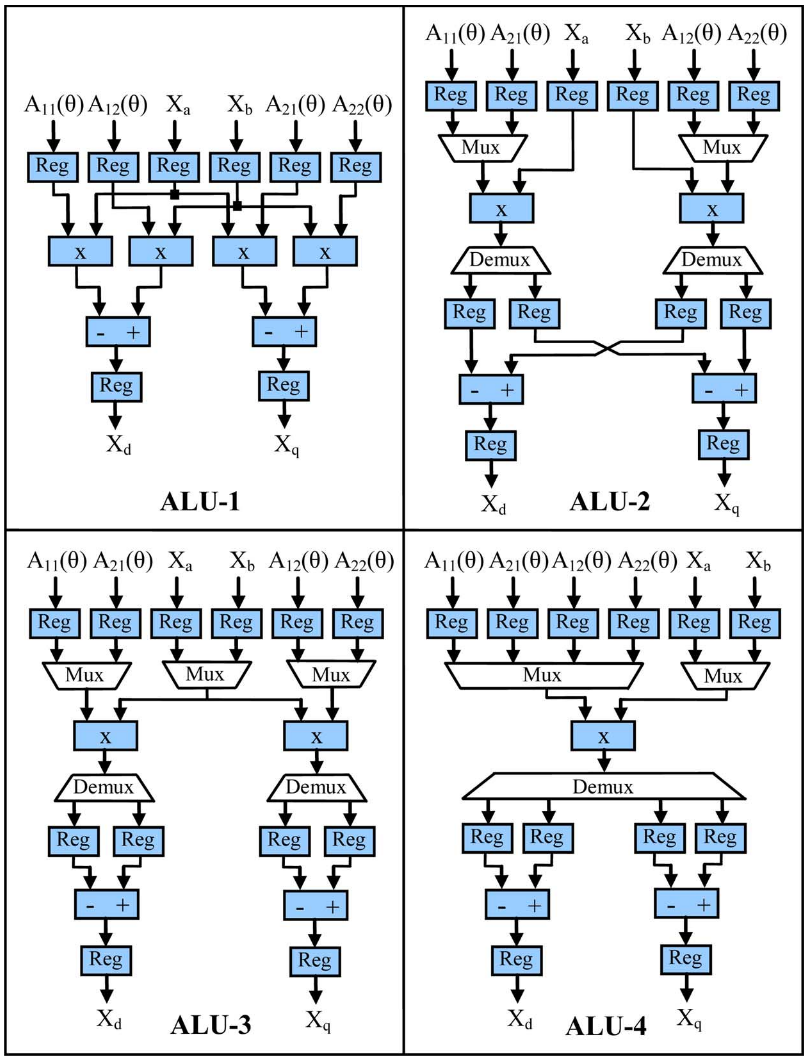
ALU-1
ALU-2
ALU-3
ALU-4
Reg
Mux
Demux
x
- +
$A_{11}(\theta)$ $A_{12}(\theta)$ X_a X_b $A_{21}(\theta)$ $A_{22}(\theta)$
X_d
X_q

Fig. 6. ALU data paths of the coordinate transformation.

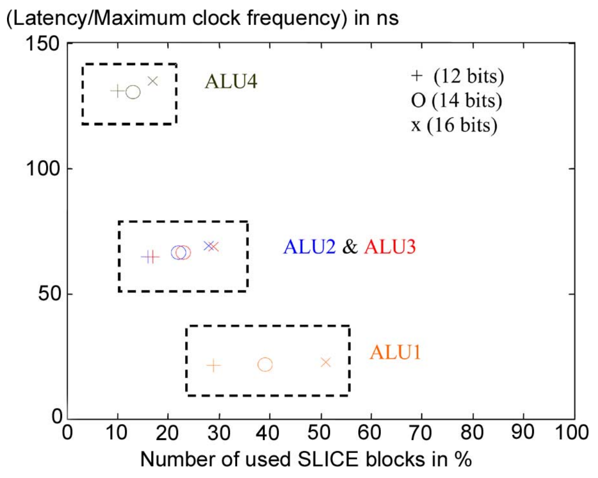

Fig. 7. Performance of the different ALU after synthesis.

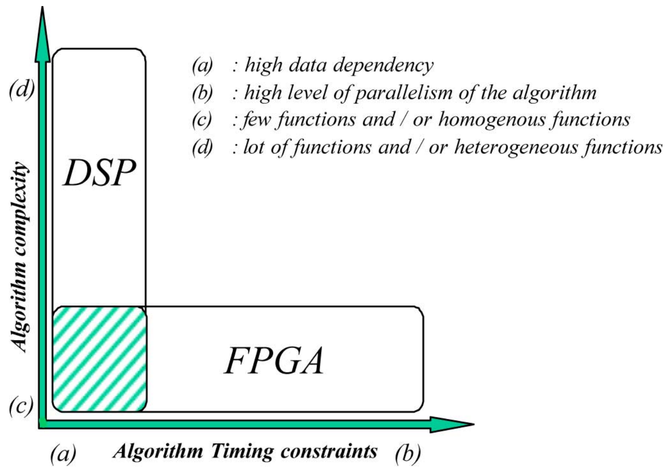
(a) : high data dependency
(b) : high level of parallelism of the algorithm
(c) : few functions and / or homogenous functions
(d) : lot of functions and / or heterogeneous functions
DSP
FPGA
Algorithm complexity
Algorithm Timing constraints

Fig. 8. DSP and FPGA domains of use.

reveals many possibilities of parallelism (low data dependence and competition between operations), it is then the hardware solution (FPGAs) which becomes the most interesting.

However, timing constraints are not sufficient to fully characterize an algorithm—its complexity is also a key element. This is the reason why it is reported on the y-axis of the graph. Algorithm complexity is evaluated in two ways: the number of operations and their regularity. Indeed, an algorithm presenting a significant number of operations is not necessarily complex if the majority of these operations are identical. It is then easy to

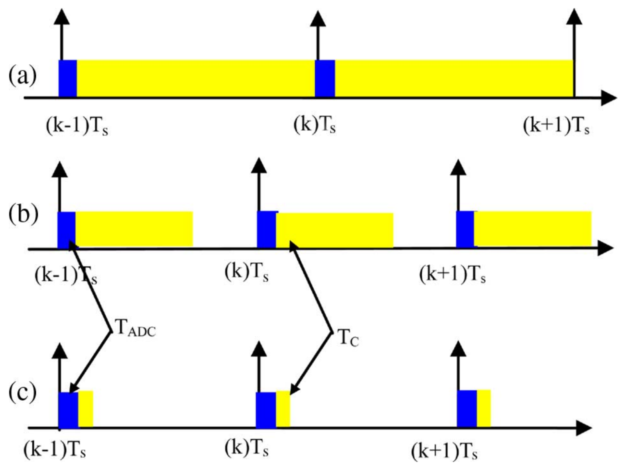

Fig. 9. Timing distribution. (a) General-purpose microcontroller, (b) DSP controller, and (c) FPGA-based controller

design an efficient ALU that is optimized for the treatment of these specific operations.

In the field of digital control of electrical systems, algorithms are almost all included in the intersection area of these two technologies. However, in many cases, the implementation in a DSP is preferred. According to us, the reason is historical. Software solutions are older, and they do not frighten the designers because they are based on programming. However, this apprehension of circuit designs is less and less founded, given the progress of design methodologies and CAD tools. When using HDLs, the FPGA implementations are also relying on portable code. Moreover, the reduction of the execution time of an algorithm in the case of a DSP implementation is only obtained by a long work of optimization of the corresponding assembler code. Such an optimization is no less consuming in terms of development time than the time needed for the design of an efficient architecture when using the A^3 methodology. However, gain in this last case is often spectacular in terms of execution time. Of course, other aspects must also be considered, such as accuracy issue.

B. Benefits of Using FPGAs for Control of Electrical Systems

As a complement of the former section, the authors are now trying to outline the benefits of using FPGAs for controlling industrial electrical systems, driven by a power converter. A typical example consists in the current control for PWM inverters. The demonstration is based on the ability of the FPGA-based controllers to execute quasi-instantaneously their tasks, as shown in Fig. 9. In this figure, T_{ADC} represents the analog-to-digital signal time conversion, T_{C} is the execution time of the control algorithm, and T_{s} is the sampling period that is usually taken equal to the switching period of the power converter or to half of it.

Fig. 9(a) corresponds to the use of a general purpose microcontroller. In this case, the main limitation factor is the computing power of this component. Sampling period is fixed according to this limit, leading to one and a half switching period of delay (one period due to the computation delay and a half-period delay which corresponds to the usual statistical delay due to the PWM signals application). This reduces the bandwidth of the closed-loop system and, in some cases, may destabilize the controlled process. Signal waveforms are also very poor [74]. Finally, it should be mentioned that direct control of power converters (sliding mode, bang-bang control) is not recommended in this case.

Fig. 9(b) corresponds to the case of a DSP controller implementation. This case gives much better results than the former one. Indeed, owing to their adapted architecture, these components allow controlling the current of an electrical machine or a load in a few dozens of microseconds. As a consequence, the lack of rapidity of the controller is no longer the limiting factor of the closed-loop system. Limitations take their origin in the switching losses of the power converter, leading to a sampling frequency of around 10–20 kHz for medium range power systems. When carefully designed, the delay can be reduced to a half switching period ($\approx 50\ \mu$s), which greatly improves the dynamic performance. Direct control of the power converters can be achieved, but expected results are of less quality than those obtained via an analog controller.

Fig. 9(c) corresponds to an FPGA-based controller. Due to their ability to transcript on the hardware architecture all the potential parallelisms of the control algorithm, FPGAs can only take a fraction of the switching period to execute in real time a full complex algorithm. A direct consequence of this extreme rapidity is the consummation of a large number of the internal resources of the chip, increasing in the same time the cost [33]. However, by using optimization techniques such as A^3 [73] and/or pipelining, the designer can easily build a balanced architecture, which respects the area limitation and preserves the rapidity of execution of the control algorithm ($\approx 1-2\ \mu$s for ac motor drive control). Therefore, the obtained computing time T_{C} for FPGA-based controllers is far below the one reached with a programmed solution.

Such instant reactions make the FPGA-based controllers very close in their behaviors to their analog counterparts. They preserve their advantages (no calculation delay, higher bandwidth) without their drawbacks (parameters drifting, poor level of integration). Hence, this quasi-analog property could be sufficient to promote this technology in implementing more and more industrial digital control systems.

However, a more careful look in Fig. 9(c) shows important time left within each sampling period, when the controller has finished its computing tasks and has only to handle the PWM signals generation. As a consequence, only a little part of the FPGA is active during this time, leaving the component largely underexploited. This important observation has been made by some authors in the past, even if very few clearly highlight it [75]. In most cases, it leads them to propose several interesting improvements that we are now trying to classify.

1) Oversampling: A first approach consists of oversampling techniques, as proposed in [25], where de Castro *et al.* integrate the current in a PFC controller at a frequency four times higher than the sampling frequency, yielding more accurate results for this current regulation. The care taken to simplify the computation (reduction of the number of multipliers) is also to be mentioned. In [76], Chapuis *et al.* propose a quasi-analog digital DTC, where the torque regulation is updated every 2 μs;

a protection module is added, which prevents switching on the same inverter leg.

2) Predictive Control: In [77], Ling *et al.* propose the implementation of a complete model predictive control strategy applied to an aircraft example. It is to be noticed that the algorithm is using floating point arithmetic, which is still rare when working with FPGAs.

3) Current Measurement Improvements: In [78], Fratta *et al.* propose an ideal PWM ripple filter, which is obtained by oversampling the measurements of the controlled current, then by computing its average value inside a time sliding window, and by reintroducing it in the current loop. Careful comparisons are also made with an alternative DSP implementation.

In [79], Blaabjerg *et al.* present an FPGA-based implementation of an SRM current control which allows avoiding antialiasing filters. Indeed, by choosing the perfect instant of sampling, a true average current may be measured.

4) Control of Multisystems With the Same Controller: In [80], Garcia *et al.* have implemented an FPGA-based controller for a 16-phases dc/dc converter. The targeted application is automotive. This kind of example is very demanding: As it is low voltage, the sampling frequency can be very high, and the controller has to manage up to 16 interleaved channels. Such power segmentation is also required in aircraft applications for reliability reasons. These examples can be considered as typical niche applications for FPGAs.

In [43], Tazi *et al.* proposed the control of up to four dc motors with the same FPGA vector current controller, with a sampling period of 50 μs.

Finally, as FPGAs can handle very fast computing and the main limitation of the power converter being the switching losses of the transistors, some authors have proposed some improvements to the classical PWM strategy in order to reduce the switching period. It consists in the single switch commutation technique that avoids the application of a deadtime when a commutation occurs. Thus, by using this technique, one can implement a faster current loop (up to 40 kHz for medium power) [81].

C. Dynamic Reconfiguration of FPGAs

Reconfigurations of control algorithms in runtime can be done by software with DSPs. Conversely, SRAM-based FPGAs allow dynamical reconfiguring of hardware architectures. This possibility has already been largely explored in computer vision applications [82]. Besides, as explained in [83], reconfiguring induction motor control algorithms in runtime, depending on the operating point of the machine, can improve significantly the performance of the whole system. Authors had also experimented with success a first dynamically reconfigurable architecture dedicated to the tests of evolving PWM strategies [84]. Another interesting use of dynamic reconfiguration consists of reconfiguring the control system when a major failure occurs [85]. The reconfiguration is then necessary to still control the plant evolution or at least to cancel the process in safety conditions.

However, the dynamic reconfiguration of hardware configuration, which can be partial or total, is still largely underexploited in the field of industrial control systems. A major reason is the poor reconfiguration speed [7]. This may change in the near future.

VI. FPGAs in Intelligent and Complex Control Systems

The use of modern EDA packages for electronic systems design facilitates easy implementation of complex control algorithms and AI into hardware. Hence, a wide range of complex and intelligent controller designs has been recently developed, with applications in industry. A significant number of them target FPGAs, due to the rapid prototyping features and the flexibility offered by FPGAs, particularly through the recent availability of microprocessor or DSP cores, allowing hardware software codesign and implementation. Some areas using FPGAs for the implementation of complex controllers are highlighted below, and a case study of an AI (fuzzy logic) controller will be dealt with in more detail in a separate section.

A. NNs Implemented in FPGA

According to a recent report of a European Network of Excellence [86], the near- and long-term future implementations of hardware-based NNs will be shaped in three ways: 1) by developing advanced techniques for mapping NNs onto FPGAs; 2) by developing innovative learning algorithms which are hardware-realizable; and 3) by defining high-level descriptions of the neural algorithms in an industry standard to allow full simulations to be carried out and to allow fabrication by the most appropriate technique and to produce demonstrators of the technology for industry.

Such designs will be of use to industry if the cost of adopting this new technology is sufficiently low for the company and if the technology is made accessible to them. The cost of implementing new technology in an ASIC falls each year. Europe lags behind Japan and the USA in the application of intelligent techniques, particularly in consumer electronics. Considerable expertise in the design of NNs and their application to industry is available in universities throughout the European Union. Strong collaboration exists in this field, particularly between universities, as expressed through existing ESPRIT programs such as NEuroNet [86].

Hardware-based NNs are important to industry as they offer a small-size and low-power consumption compared to software running on a workstation. Therefore, such NN controllers can be embedded in a wide range of systems both large and small. The benefits of NNs to industry have been recognized particularly in Japan, where a number of consumer goods are making use of this technology. A recent prominent product has been a microwave oven (Sharp), which uses a neural module developed in the U.K. Other consumer applications of related technology include fuzzy logic modules in cameras and in vacuum cleaners. Solutions should be tailored to the needs of industry by providing a choice of implementations from software modules, through FPGAs and semicustom chips to full-custom VLSI. Libraries of neural functions should be made available in software and libraries of cells (digital, mixed, and analog) for hardware. Software libraries exist for the traditional NN models, for example, for use with MATLAB.

For industry to take up university-based designs, these designs must be in an industry-standard form, for example, VHDL or C++ functional code, they should be modular and they should be parameterized to allow customization to the industry's needs. The following European companies are known to have investigated the use of hardware-based NNs: Ericsson (U.K., Sweden), Philips Research (Holland), Siemens (Germany, U.K.), 3M Laboratories Europe GmbH Neuss, XIONICS Document Technologies GmbH Dortmund, Robert Bosch GmbH Reutlingen, Spectrum Microelectronics Siek (Germany), Fiat (Italy), and Domain Dynamics Ltd. (U.K.) [86]. Specific application areas include the control of telecommunications networks, speech processing and recognition, speaker identification, and microelectromechanical systems. The industry that already applies neural technology, or is likely to benefit from it, is already pan-European. For example, Siemens has activities in both Germany and the U.K., Ericsson has activities in Sweden and in most European states, and the U.K. hosts Ericsson's VLSI Design Centre. The main areas of application are [86] as follows:

1) communications systems, demodulators, intelligent antennas, and semiconductors for the space environment;
2) object identification, image compression, HDTV, medical and biometric image analysis, thermal image processing systems, and materials analysis;
3) character recognition, speaker identification, speech recognition and enhancement, and handwriting recognition;
4) information retrieval, exploratory data analysis, quality control, function learning, automatic control, economic prediction, electrical consumption prediction, knowledge extraction, intelligent controls, and automatic verification of VLSI and wafer scale integration circuits;
5) stochastic learning algorithms, content addressable memory, massively parallel processors, and pulse-stream computation.

Some directions for implementation [86] are as follows:

1) VLSI digital and analog hardware, analog implementation of NNs, pulse-stream systems, and on-chip weight perturbation algorithms;
2) on-chip learning, reinforcement training, feedforward training, and stochastic training;
3) distributed and heterogeneous processor architectures, fault tolerant systems, and optical neural techniques;
4) analog and mixed hardware implementations of NNs using time-continuous or coherent PWM techniques;
5) massively parallel computers, silicon implementations of NNs, and neuro-fuzzy systems.

A wide range of research papers on neural-networks-based controllers was published in prestigious journals. Some (like [87]) were collated in special issues on Transactions of Industrial Electronics [88], [89]. Recently, other papers on NNs are more frequently present in regular issues of this journal ([90]–[94]).

B. Fuzzy-Logic-Based Control Systems

Today, fuzzy-logic-based control systems or simply fuzzy logic controllers (FLCs) can be found in a growing number of products, from washing machines to speedboats, from air condition units to hand-held autofocus cameras. The success of FLCs is mainly due to their ability to cope with knowledge represented in a linguistic form instead of representation in the conventional mathematical framework. Control engineers have traditionally relied on mathematical models for their designs. However, the more complex a system, the less effective the mathematical model. This fundamental concept provided the motivation for fuzzy logic and is stated by Zadeh as the principle of incompatibility [95]. There are five main elements in an FLC: fuzzification module (fuzzifier), knowledge base, rule base, inference engine, and defuzzification module (defuzzifier). Automatic changes in the design parameters of any of the five elements create an adaptive fuzzy controller. Fuzzy control systems with fixed parameters are nonadaptive. Other nonfuzzy elements, which are also part of the control system, include sensors, analog–digital converters, digital–analog converters, and normalization circuits. There are two types of normalization circuits: One maps the physical values of the control inputs onto a normalized universe of discourse, and the other maps the normalized value of the control output variables back onto its physical domain.

FPGAs constitute an appropriate target for the implementation of FLCs, which is facilitated by the flexibility of the design environment, enabling a direct implementation of the circuit's abstract model. A high number of works have been published on fuzzy-logic-based control systems. One paper presents a method employing hardware/software codesign techniques according to an "*a priori*" partition of the tasks assigned to the selected components. This feature makes it possible to tackle the control system prototyping as one of the design stages. In this case, the platform considered for prototyping has been a development board containing a standard microcontroller and an FPGA. Experimental results from an actual control application validate the efficiency of this methodology [96].

A paper by Poorani *et al.* advocates a novel approach to implement the FLC for speed control of electric vehicle by using the FPGA [97]. The speed of the motor has to be controlled, which, in turn, controls the vehicle dynamics to run the vehicle. Therefore, the main aim is to determine the motor speed, which drives the vehicle. In this respect, parameters such as acceleration, braking, energy status, gear, and terrain are considered. This system, which functions as a closed-loop system, also takes the motor speed as a reference along with the aforementioned parameters to estimate the variation of the motor speed [97].

A paper by Kim [98] presents an implementation of an FLC on a reconfigurable FPGA system. Another paper by Lago *et al.* explores the use of FPGA technologies to implement the FLCs. Two different approaches are described. The first option is based on the logic synthesis of the Boolean equations describing the controller input–output relations. The second approach uses a dedicated hardware to implement the fuzzy algorithm according to a specific architecture based on a VHDL cell library [99]. An FPGA-based fuzzy sliding-mode controller, which combines both the merits of fuzzy control and sliding-mode control, is proposed in [100] to control the mover position of a linear induction motor drive to compensate the uncertainties, including the frictional force. The uncertainties are lumped in the sliding-mode controller, and the upper bound of the lumped uncertainty is necessary in the design of the sliding-mode controller, but it is difficult to obtain in advance in practical applications. Therefore, a fuzzy sliding-mode controller is investigated, in which a simple fuzzy inference mechanism is utilized to estimate the upper bound of the lumped uncertainty. An FPGA is adopted to implement the indirect field-oriented mechanism and the developed control algorithms for possible low-cost and high-performance industrial applications.

C. Hardware Implementation of Fuzzy and NN Controllers

A paper on problems of hardware implementation of NNs in the reprogrammable structures was written by Klepaczko *et al.* [101]. New class of these devices, which integrate in one silicon wafer entire SoC, facilitates NN construction and their application. The cooperation of Micro-Controller Unit (MCU) and FPGA helps to overcome space and interconnection limitations. The paper aims to prove that large multilayer NNs are achievable by associating programmable logic array with a microcontroller, which supports space and speed-efficient designs, in comparison to systems realized only in an FPGA device or simulated only by MCU. Much attention has been devoted to the practical application of the NN in the System for European Water Monitoring (SEWING) [101].

Another work is focused on custom architectures for fuzzy and NNs controllers [102]. It presents efficient architecture approaches to develop controllers using specific circuits, using HDLs, and synthesizing them to get the FPGA configuration bitstream.

D. Intelligent Data Acquisition Devices (DAQ)

Intelligent DAQ devices use National Instruments LabVIEW reconfigurable FPGAs to implement custom high-performance DAQ on commercial off-the-shelf (COTS) hardware. Instead of a predefined subset of DAQ functionality, the intelligent DAQ uses an FPGA-based system timing controller to make all analog and digital I/Os configurable for application-specific operation. By programming the FPGA, the custom high-performance DAQ tasks can easily be implemented. Additionally, because of the parallel architectures of FPGAs, the high-performance task implementation is achieved without performance degradation [103]. With the new direct memory access (DMA) capabilities in the LabVIEW 8 FPGA Module, data from within the execution of the FPGA device can be retrieved at speeds up to 50 MB/s, depending on the target hardware and host processor. DMA provides a direct link for data on the FPGA to RAM on the host machine, improving data-logging efficiency and making data immediately available for analysis and visualization. This high-speed data transfer provides real-time visibility into parameters and variables within the FPGA [103].

E. Evolvable Hardware

Evolvable hardware offers much for the future of complex system design. Evolutionary techniques do not give the potential to explore larger solution spaces, but when implemented on hardware, these techniques allow system designs to adapt to changes in the environment, including failures in system components. Novel evolutionary algorithms are being developed and applied to intrinsic hardware evolution [104]. A major objective of this paper is to produce an evolutionary system that can be readily implemented on COTS hardware. As an example of the new system, an FPGA-based controller for a mobile robot has been developed by Prof. A. Tyrrell and his team at the University of York, U.K. The controller consists of LUTs, which perform the mapping from sensor data to actuator, evolved using an effective evolutionary algorithm. The experimental results on a Khepera robot show that the method can successfully evolve a robot controller for autonomous navigation to avoid collision in an unknown or changing environment even if sensor faults occur prior to evolution or after a successful member of a population has been evolved [104].

F. Controller Designs for Smart Structural Systems

The design of controllers for smart structural systems usually proceeds without regard for their eventual implementation, thus resulting either in serious performance degradation or in hardware requirements that squander power, complicate integration, and drive up cost. The level of integration assumed by the smart patch further exacerbates these difficulties, and any design inefficiency may render the realization of a single-package sensor-controller-actuator system infeasible. The research carried out automates the controller implementation process and relieves the design engineer of implementation concerns like quantization, computational efficiency, and device selection. FPGAs are specifically targeted as a hardware platform, because these devices are highly flexible, power efficient, and reprogrammable. The proposed controller design methodology is implemented on a simple cantilever beam test structure using FPGA hardware [105].

G. FPGAs Used in Motion Control Interface

New Ethernet-based FPGA-based controllers for motion control are reported [106]. They include all hardware functions such as timing, synchronization, and processing of cyclic and noncyclic data on the basis of two integrated Ethernet MACs. Cores for two controllers are available, based on the low-cost Spartan-3 Xilinx FPGA platform. The SERCON100 master and slave controllers are available, which are both integrated in an FT256 BGA housing so that a common hardware design can be realized. This makes a very powerful low-cost standard hardware platform available, which reduces implementation efforts and also ensures a high acceptance by suppliers [106].

VII. FPGA-Based DTC Controller

In this section, the authors present the FPGA-based implementations of direct torque and stator flux control (DTC) and

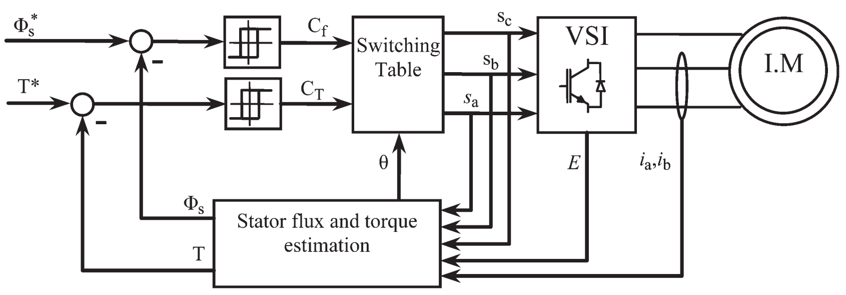

Fig. 10. Block diagram of the DTC technique.

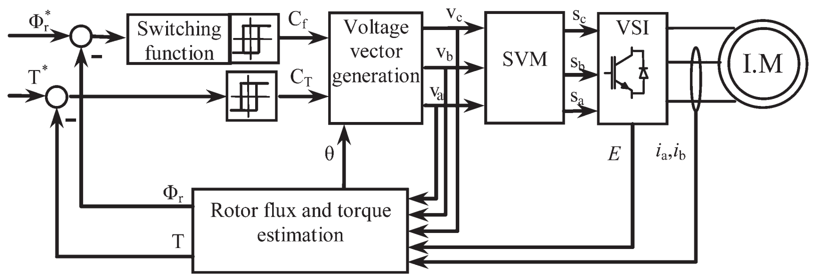

Fig. 11. Block diagram of SVM-DTRFC strategy.

TABLE I
FUNCTIONAL ALGORITHM DECOMPOSITION

	Transformation blocks	**Estimation blocks**	**Control blocks**
DTC	-(a,b,c) to (d,q)	- Stator flux (Magnitude, Angle) - Torque	- Hysteresis - Switching table
SVM-DTRFC	- (a,b,c) to (d,q)	- Rotor flux (Magnitude, Angle) - Torque - voltage vector	- Hysteresis - Switching function - SVM

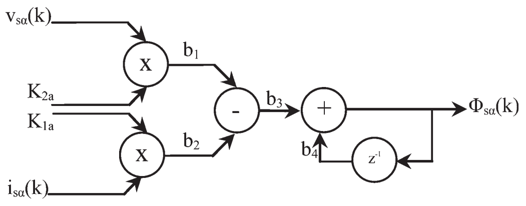

Fig. 12. DFG of the α-axis stator flux estimator.

TABLE II
OPTIMAL WORD LENGTH

Word length						SQNR
is_α	b_1	b_2	b_3	b_4	$\Phi_{s\alpha}$	(dB)
24	24	24	24	24	24	84.2
16	16	16	22	22	16	80.5

direct torque and rotor flux control (DTRFC) with the use of space vector modulation (SVM) for induction motor drives. Indeed, due to their similar structures but also their differences, these two algorithms are good examples to show the effectiveness of an FPGA-based functional modular approach to implement sensorless control induction motor drives. Therefore, the chosen solution is based on a custom hardware architecture designed by assembling a set of functional building blocks. These blocks are tested and organized in a library of IP modules for easy reuse [107]. Each block is geared toward a specific algorithm function (flux estimator, hysteresis controller, etc.). A special attention is given to the algorithm refinement, which allows finding the optimum fixed-point data word length for each internal variable of the algorithm. Finally, experimental results are shown, which validate the proposed approach.

A. Principles of the Proposed Control Algorithms

DTC and SVM-DTRFC algorithms have high torque dynamic performances. In a first approximation, the SVM-DTRFC algorithm can be considered as derived from the well-known DTC algorithm [108]. While the basic DTC technique is to directly select stator voltage vectors according to the differences between reference and actual torque and between reference and stator flux linkage, the SVM-DTRFC strategy is based on torque and rotor flux control [109]. Moreover, in this case, the voltage source inverter is controlled indirectly by using the SVM in a similar way with what was proposed in [110]. This technique allows a smoother behavior of the torque regulation at steady-state operation than basic DTC.

The block diagrams of DTC and SVM-DTRFC are presented in Figs. 10 and 11, respectively.

B. Design of Modular Architectures

The discretization of the normalized control algorithms is performed with the forward-difference approximation. A full description of these algorithms can be found in [111]. Then, the algorithm refinement procedure is carried out.

In order to increase module reutilization, a modular and standard design principle is applied. The functional algorithm decomposition leads to a set of specific subalgorithms or modules, which are summarized in Table I.

As shown in Table I, there are several common blocks used by both control algorithms. For each block, an appropriate DFG

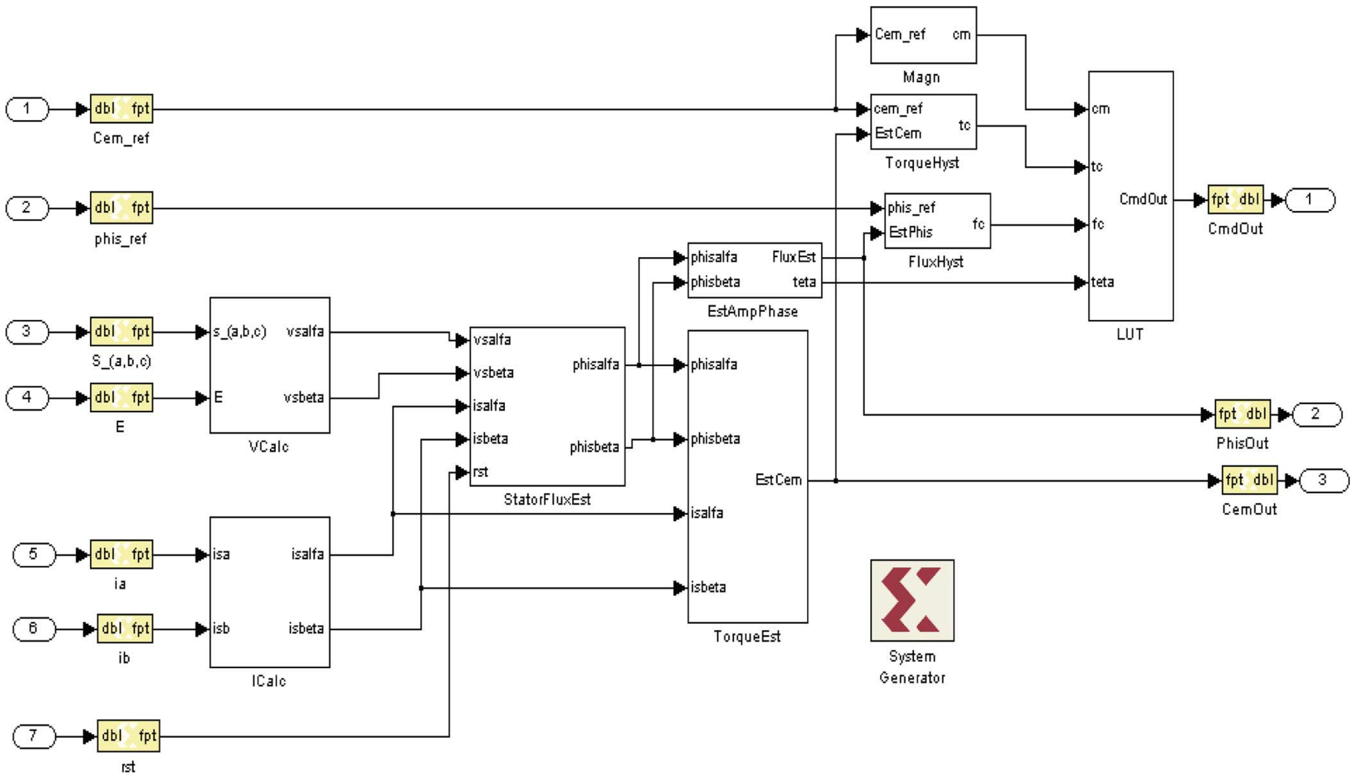

Fig. 13. Diagram of the DTC controller (built with Xilinx SG toolbox).

has been established. Fig. 12 illustrates the DFG of the α-axis stator flux estimator.

As for the magnitude and the angle of the stator or rotor flux vector, they are estimated using CORDIC [64]. A MATLAB program is used to search for the best choice in terms of accuracy and number of CORDIC steps. It has been found that ten CORDIC steps are enough for this application. As mentioned earlier, an interesting metric for evaluating the precision of the digital algorithms developed with fixed-point arithmetic is the SQNR [67].

As an example, a set of evaluations performed with the α-axis stator flux estimator is described, for which the DFG is presented in Fig. 12. The results are given in Table II for an SQNR constraint of 80 dB. It is shown that the first solution, where all the signals have the same word length (first row), can be optimized using the study in [67] while respecting the SQNR constraint (second row).

Simulation is then achieved under Simulink with the Xilinx System Generator (SG) fixed-point toolbox. Fig. 13 shows a simulated version of the DTC algorithm by SG toolbox. Good performance was obtained using the optimum fixed-point formats established by the analytic approach [67]. The sampling frequency is fixed to 20 kHz.

The development of each module in terms of architecture is based on standardization principles. These principles are regularity, understandability, and reusability of already designed components. An RTL library of standard IP blocks is developed [107]. A detailed description of the flux and torque estimator module is now given as example.

1) Description of the module: This module implements a torque and stator and rotor flux vector estimators for a three-phase induction motor. The data path is obtained with the help of the A^3 methodology [73]. As shown in Fig. 14, the factorization process is applied to the greediest operators (multipliers).
2) Module properties: Scalable module based on generic VHDL is developed. Module latency is 23 clock cycles (40 MHz). Hardware used resources are 29% of a Spartan XC2s100 FPGA [7].
3) Module interface: The resulting (RTL) model of the estimator architecture, presenting the VHDL entity, is shown in Fig. 15.

C. Experimental Results

Experiments are carried out with a 1-kW four-pole induction motor. These two algorithms have been implemented on a low-cost FPGA device (XC2S100). Table III reports the very short execution time for each FPGA-based control algorithm. Fig. 16(a) and (b) presents torque step responses using, respectively, DTC and SVM-DTRFC algorithms.

Torque dynamic is almost the same for both control algorithms (1 ms). It is to be noted that, in the case of SVM-DTRFC, torque ripples are significantly reduced. Interested readers can compare these performances to those obtained with a DSP-controller target [112]. However, direct comparisons between FPGA and DSP controllers in terms of performance must be lead carefully. Indeed, to be totally fair, among others, both targets must be of the same generation, the design effort must be similar, and control features—accuracy and sampling period—must be identical too. The authors do not aim to perform such a comparison in this paper or to open a debate about the

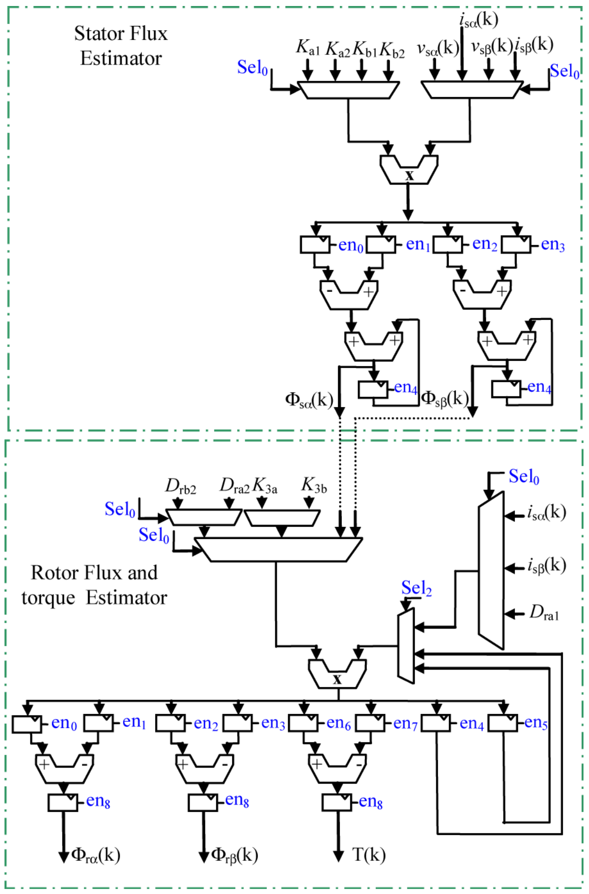

Fig. 14. Estimator architecture factorized by the use of A^3 methodology.

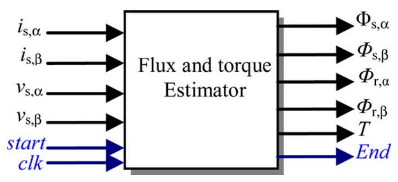

Fig. 15. Top view of the RTL model.

preference of using one or another of these technologies. This paper is focused on reviewing the use of FPGAs in industrial control.

VIII. FLC FOR STAND-ALONE SYNCHRONOUS GENERATORS

This case study describes the analysis and design of an electronic control system allowing variable speed operation of diesel-driven stand-alone synchronous generators [1], [45]. The system is shown in Fig. 17. A control scheme that can isolate the final output frequency of the system from the effects of speed variations is simulated and designed. The proposed design aims to improve the efficiency of diesel-engine-driven generators by allowing optimum speed operation. Fuel economy and environmental protection are achieved. The ac generated voltage is rectified into dc power and then converted back into ac using a PWM inverter before being applied to the load. This configuration is widely used in variable-speed wind energy conversion systems [1], [113]. A suitable fuzzy logic control system is designed to control the fuel valve of the diesel engine based on the dc link voltage input. The overall function of the control system is to ensure that the output voltage of the system maintains the desired magnitude and frequency over a range of varying rotational speed and loading conditions. Due to the difficulty in obtaining a precise model for the engine-generator set, fuzzy logic is used in the control system, as it does not require an accurate mathematical model. A typical fuzzy control system can be divided into four main sections.

TABLE III
PROCESSING TIME FOR FPGA-BASED CONTROL ALGORITHMS

FPGA-based architectures	Processing time (μs)
DTC	1.15
SVM-DTRFC	1.42

1) Fuzzifier—compares the input variables of the controller with a predefined set of membership functions and assigns the appropriate membership values. Thus, the fuzzifier converts crisp input signals into fuzzy values.
2) Fuzzy inference machine—links the controller to a set of fuzzy rules.
3) Fuzzy rule base—set of intuitive or linguistic rules, which forms the basis of the control strategy.
4) Defuzzifier—performs a function opposite to that of the fuzzifier: It converts the control system's fuzzy output into a single crisp value that can be applied as control signal.

In the actual fuzzy logic control module [45], the $V_{\rm dc}$ voltage and the rate of change of $V_{\rm dc}$ are used as input variables. The output is the fuel flow rate control signal. The steps in designing the FLC are as follows.

1) Identify the variables.
2) Formulate fuzzy rules and fuzzy associative memory table. The FVSG has 25 fuzzy rules that map the input states to 25 output conditions (C^1 to C^{25}). General form is

$$R_k : \text{IF } x_1 \text{ is } A_1^k \text{ and } x_2 \text{ is } A_2^k, \qquad \text{THEN } C^k$$

where $R_k (k = 1, 2, \ldots, 25)$ is the kth rule of the fuzzy system, and x_1 and x_2 are the input variables. $A_i^k (i = 1, 2; k = 1, 2, \ldots, 25)$ is the kth fuzzy set defined in the ith input space, and C^k is the output condition inferred by the kth rule.
3) Define membership function for input variables (fuzzification).
4) Define membership function for fuzzy output sets.
5) Defuzzification. The defuzzification process chosen for this controller is the weighted average method [45].

The FLC design is achieved in VHDL, which allows easy description of the fuzzy implication techniques. For example: PB (Positive Big) $= \max(C^1, C^2, C^6)$ is described:

$$\begin{aligned}&\text{PB} <= 0 \text{ when } c1 = 0 \text{ and } c2 = 0 \text{ and } c6 = 0\\&\text{else } c1 \text{ when } (c1 >= c2 \text{ and } c1 >= c6) \text{ else}\\&c2 \text{ when } (c2 >= c1 \text{ and } c2 >= c6) \text{ else } c6.\end{aligned}$$

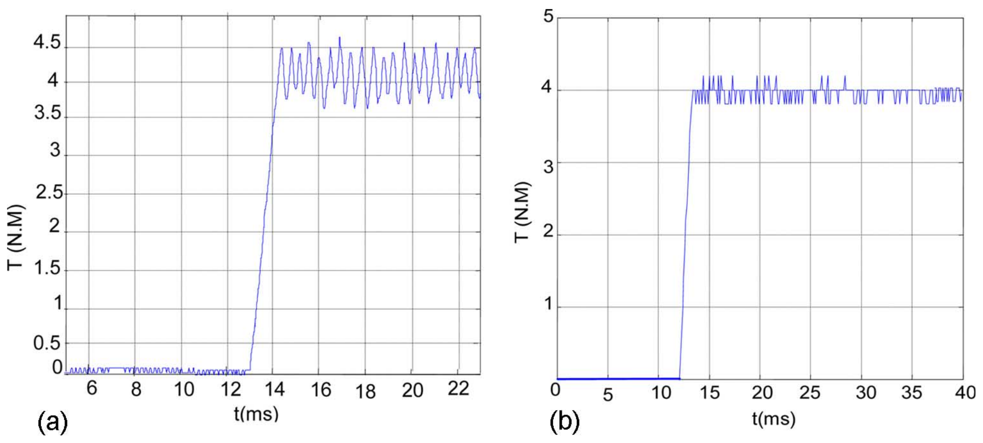

Fig. 16. Experimental results torque step response 0 to 4 nm, $T_s = 50\ \mu$s. (a) DTC. (b) DTRFC.

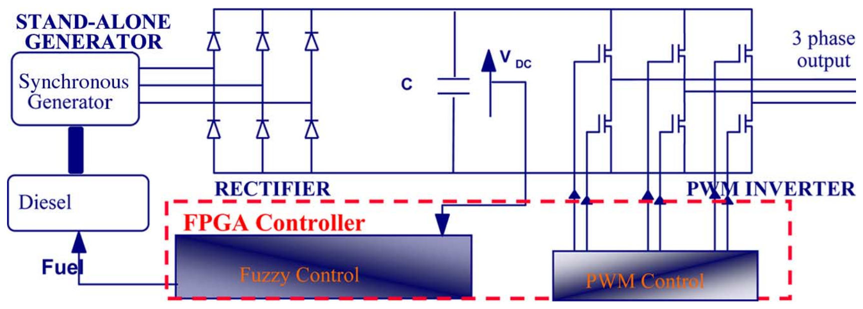

Fig. 17. Fuzzy-logic-controlled power system.

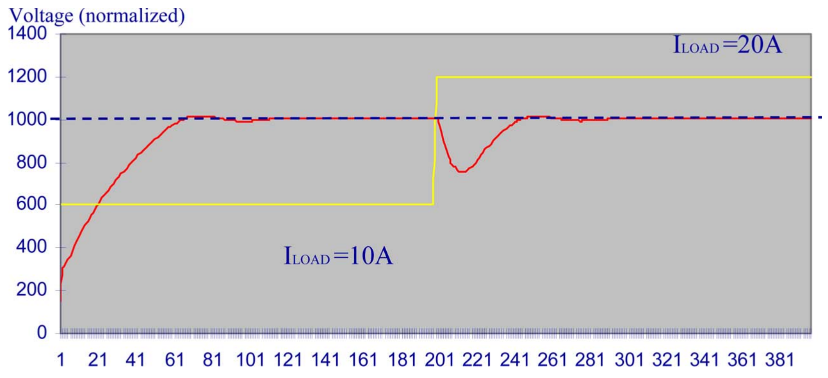

Fig. 18. Simulation results.

In this example, the rules state that the output signal u is PB when the output condition is C^1, C^2, or C^6.

The dc output voltage is simulated during a step increase of the load current. The results in Fig. 18 show that the fuzzy logic control system is successfully correcting the tendency to fall of the output voltage V_{dc}. The system is therefore able to cope with variations in V_{dc} resulting from variable load and variable speed of operation. After the complete system was modeled and simulated using VHDL, the circuit design of the controller was synthesized and implemented into a Xilinx XC4010 FPGA for rapid prototyping. By adjusting the speed of the engine to the operating conditions, fuel consumption can be reduced while the same torque can be produced. Fig. 19 shows the voltage response when the controller is connected to the system. The desired dc voltage is set at 250 V.

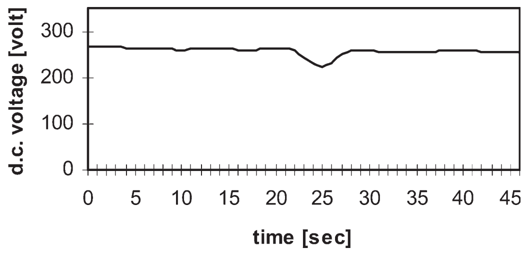

Fig. 19. Voltage response with control system.

The graph shows that the controller is successful in stabilizing the generator system. Although there is a voltage drop of about 14% when the load resistance is decreased, this effect is counteracted by the controller, and the voltage level recovers to a steady value. The main achievements of this system are as follows.

1) The configuration allows the final output voltage to be independent of the generator speed, thus allowing the system to operate at the most efficient speed at all times.
2) The control system maintains the output voltage at the desired magnitude and frequency against changes in V_{dc} which arise from changes in speed and/or load.

The system provides a suitable platform for the study of efficient diesel-engine-driven variable speed generators.

IX. Conclusion and Perspectives

The aim of this paper has been to present the contributions of FPGAs to the control of industrial systems. After a short description of FPGAs and their CAD tools, the authors have focused on the design methodology issue. Indeed, due to the simultaneous increase of the control algorithm complexity and the chip density, using an efficient design methodology is essential in this context. To this purpose, a modeling technique was proposed for the holistic investigation of power electronic systems. This is based on system-level modeling languages or HDLs and allows rapid FPGA prototyping of the control systems. The digital controller designs were developed from idea, through the design and simulation stages, to complete systems in a short time and in close interaction with the optimized holistic model of the complex engineering industrial system to be controlled. Further advantages are provided, such as multiple choices for the implementation target technology, a reliable framework for design verification, high confidence in the correct first time operation, and wide compatibility of the design (as IP block) with respect to multiple existing modern CAD tools. The latter allows the integration of digital electronic controller models in complex system models.

After that, three main design rules were presented. The main characteristics of the proposed architecture design methodology are the algorithm refinement, the modularity, and the systematic search for the best compromise between the control performances and the architectural constraints (see A^3 section). Then, an overview of contributions and limits of FPGAs were proposed, and comparisons with traditional DSP software solutions are also made. This section was followed by a short survey of FPGA-based intelligent controllers for modern industrial systems. Finally, full and timely examples were presented to illustrate the benefits of FPGA implementation when using the proposed design approach. They include the DTC for induction motor drives and the control of a synchronous stand-alone generator using fuzzy logic. It was demonstrated that in both cases, a low-cost FPGA-based controller can greatly improve the control performance, particularly due to the reduction of execution time while keeping a high level of flexibility.

In the near future, the complexity of the control systems will continue to grow. The tasks devoted to the control algorithm will no longer be limited to regulation but will have to manage diagnosis and fault-adaptive online control. In this context, the research effort on the theory and the applications of dynamic reconfiguration is crucial.

Another interesting direction of research is based on the following observation: A control algorithm, when implemented in an FPGA, can have a very short execution time due to the high degree of parallelism of its architecture. At the same time, the constraints imposed by the power electronic components imply a sampling period that is, for many applications, much higher than the execution time. The resulting “wasted time” could be advantageously employed. Several examples of relevant FPGA utilizations in this context are presented in Section V-B. They consist of predictive control, oversampling strategies, multiplants control, etc. All these very promising control paradigms must still be improved.

Another perspective on FPGA design is to propose a prototyping development system of a fully integrated controller from VLSI technology and SoC design that can include digital control and its analog interface (sensors, ADC, power drivers, etc.) [8]. Finally, the codesign [114] issue must be addressed, since the borders between software and hardware are rapidly vanishing (one can already implement up to four PowerPCs inside a single FPGA [6]). The main problem in this case is to propose automatic rules of partitioning, based on relevant quantitative indicators. However, it can be estimated that holistic modeling methodologies will be of great help for such tasks and also for rapid controller prototyping in the very near future.

Acknowledgment

The authors would like to thank Dr. J. G. Khor, Dr. L. Charaâbi, and W. Naouar for their research on some of the case studies presented.

References

[1] M. N. Cirstea, A. Dinu, J. Khor, and M. McCormick, *Neural and Fuzzy Logic Control of Drives and Power Systems*. Oxford, U.K.: Elsevier, 2002.
[2] D. L. Perry, *VHDL*. New York: McGraw-Hill, 2004.
[3] C. Cecati, “Microprocessors for power electronics and electrical drives applications,” *IEEE Ind. Electron. Soc. Newsl.*, vol. 46, no. 3, pp. 5–9, Sep. 1999.
[4] S. Brown, “FPGA architectural research: A survey,” *IEEE Des. Test. Comput.*, vol. 13, no. 4, pp. 9–15, Winter 1996.
[5] *Internet sites and on line journals dedicated to FPGAs, such as: “FPGA-FAQ”*. [Online]. Available: http://www.fpga-faq.com/, “FPGA and Structured ASIC Journal,” http://www.fpgajournal.com/
[6] *Altera data book*, 2006. [Online]. Available: www.altera.com
[7] *Xilinx data book*, 2006. [Online]. Available: www.xilinx.com
[8] *Actel data book*, 2006. [Online]. Available: www.Actel.com
[9] D. H. Lee, A. Choi, J. M. Koo, J. I. Lee, and B. M. Kim, “A wide-band DS-CDMA modem for a mobile station,” *IEEE Trans. Consum. Electron.*, vol. 45, no. 4, pp. 1259–1269, Nov. 1999.
[10] P. Pirsch, N. Demassieux, and W. Gehrke, “VLSI architectures for video compression—A survey,” *Proc. IEEE*, vol. 83, no. 2, pp. 220–246, Feb. 1995.
[11] S. J. Ovaska and O. Vainio, “Evolutionary-programming-based optimization of reduced-rank adaptive filters for reference generation in active power filters,” *IEEE Trans. Ind. Electron.*, vol. 51, no. 4, pp. 910–916, Aug. 2004.
[12] R. X. Chen, L. G. Chen, and L. Chen, “System design consideration for digital wheelchair controller,” *IEEE Trans. Ind. Electron.*, vol. 47, no. 4, pp. 898–907, Aug. 2000.
[13] K. Sridharan and T. K. Priya, “The design of a hardware accelerator for real-time complete visibility graph construction and efficient FPGA implementation,” *IEEE Trans. Ind. Electron.*, vol. 52, no. 4, pp. 1185–1187, Aug. 2005.

[14] T. N. Chang, B. Cheng, and P. Sriwilaijaroen, "Motion control firmware for high-speed robotic systems," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1713–1722, Oct. 2006.

[15] T. H. S. Li, C. Shih-Jie, and C. Yi-Xiang, "Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot," *IEEE Trans. Ind. Electron.*, vol. 50, no. 5, pp. 867–880, Oct. 2003.

[16] M. Gabrick, R. Nicholson, F. Winters, B. Young, and J. Patton, "FPGA considerations for automotive applications," in *Proc. SAE Conf.*, 2006, CD-ROM.

[17] J. J. Wang, R. B. Katz, J. S. Sun, B. E. Cronquist, J. L. McCollum, T. M. Speers, and W. C. Plants, "SRAM based reprogrammable FPGA for space applications," *IEEE Trans. Nucl. Sci.*, vol. 46, no. 6, pp. 1728–1735, Dec. 1999.

[18] H. Y. Lui, C. H. Lee, and R. H. Patel, "Power budget power estimation and thermal budgeting methodology for FPGAs," in *Proc. Custom Integr. Circuits Conf.*, 2004, pp. 711–714.

[19] S. Velusamy, W. Huang, J. Lach, M. Stan, and K. Skadron, "Monitoring temperature in FPGA based SoCs," in *Proc. Comput. Des. Conf.*, 2005, pp. 634–637.

[20] N. P. Ligocki, A. Rettberg, M. Zanella, A. Hennig, and A. L. de Freitas Francisco, "Towards a modular communication system for FPGAs," in *Proc. Electron. Des., Test and Appl. Workshop*, 2004, pp. 71–76.

[21] D. E. Johnson, K. S. Morgan, M. J. Wirthlin, M. P. Carey, and P. S. Graham, "Persistent errors in SRAM-based FPGAs," in *Proc. MAPL Conf.*, 2004, CD-ROM.

[22] E. Kiel, "Control electronics in drive systems micro controller, DSPs, FPGAs, ASICs from state-of-art to future trends," in *Proc. PCIM Conf.*, 2002, CD-ROM.

[23] Y. Y. Tzou and H. J. Hsu, "FPGA realization of space-vector PWM control IC for three-phase PWM inverters," *IEEE Trans. Power Electron.*, vol. 12, no. 6, pp. 953–963, Nov. 1997.

[24] H. Abu-Rub, J. Guzinski, Z. Krzeminski, and H. A. Toliyat, "Predictive current control of voltage-source inverters," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 585–593, Jun. 2004.

[25] A. de Castro, P. Zumel, O. Garcia, T. Riesgo, and J. Uceda, "Concurrent and simple digital controller of an AC/DC converter with power factor correction based on an FPGA," *IEEE Trans. Power Electron.*, vol. 18, no. 1, pp. 334–343, Jan. 2003.

[26] M. Aime, G. Gateau, and T. Meynard, "Implementation of a peak current control algorithm within a field programmable gate array," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 406–418, Feb. 2007.

[27] G. Gateau, A. M. Lienhardt, and T. Meynard, "Digital sliding mode observer implementation using FPGA," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1865–1875, Aug. 2007.

[28] J. Mahlein, J. Igney, J. Weigold, M. Braun, and O. Simon, "Matrix converter commutation strategies with and without explicit input voltage sign measurement," *IEEE Trans. Ind. Electron.*, vol. 49, no. 2, pp. 407–414, Apr. 2002.

[29] P. W. Wheeler, J. Clare, and L. Empringham, "Enhancement of matrix converter output waveform quality using minimized commutation times," *IEEE Trans. Ind. Electron.*, vol. 51, no. 1, pp. 240–244, Feb. 2004.

[30] R. Garcia-Gil, J. M. Espi, E. J. Dede, and E. Sanchis-Kilders, "A bidirectional and isolated three-phase rectifier with soft-switching operation," *IEEE Trans. Ind. Electron.*, vol. 52, no. 3, pp. 765–773, Jun. 2005.

[31] J. Acero, D. Navarro, L.-A. Barragán, I. Garde, J.-I. Artigas, and J.-M. Burdío, "FPGA-based power measuring for induction heating appliances using sigma-delta A/D conversion," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1843–1852, Aug. 2007.

[32] V. Dinavahi, R. Iravani, and R. Bonert, "Design of a real-time digital simulator for a D-STATCOM system," *IEEE Trans. Ind. Electron.*, vol. 51, no. 5, pp. 1001–1008, Oct. 2004.

[33] E. Monmasson, J. C. Hapiot, and M. Granpierre, "A digital control system based on field programmable gate array for AC drives," *EPE J.*, vol. 3, no. 4, pp. 227–234, Nov. 1993.

[34] Y.-A. Chapuis, C. Girerd, F. Aubépart, J.-P. Blondé, and F. Braun, "Quantization problem analysis on ASIC-based direct torque control of an induction machine," in *Proc. IEEE IECON*, 1998, pp. 1527–1532.

[35] X. Lin-Shi, F. Morel, A. Llor, B. Allard, and J. M. Retif, "Implementation of hybrid control for motor drives," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1946–1952, Aug. 2007.

[36] G. Edelbaher, K. Jezernik, and E. Urlep, "Low-speed sensorless control of induction machine," *IEEE Trans. Ind. Electron.*, vol. 53, no. 1, pp. 120–129, Dec. 2005.

[37] A. Aounis, "An investigation into induction motor vector control based on reusable VHDL digital architectures and FPGA rapid prototyping," Ph.D. dissertation, De Montfort Univ., Leicester, U.K., 2002.

[38] N. R. N. Idris and A. H. M. Yatim, "Direct torque control of induction machines with constant switching frequency and reduced torque ripple," *IEEE Trans. Ind. Electron.*, vol. 51, no. 4, pp. 758–767, Aug. 2004.

[39] H. Abu-Rub, J. Guzinski, Z. Krzeminski, and H. A. Toliyat, "Advanced control of induction motor based on load angle estimation," *IEEE Trans. Ind. Electron.*, vol. 51, no. 1, pp. 5–14, Feb. 2004.

[40] A. K. Jain and N. Mohan, "Dynamic modeling, experimental characterization, and verification for SRM operation with simultaneous two-phase excitation," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1238–1249, Jun. 2006.

[41] R. Dubey, P. Agarwal, and M. K. Vasantha, "Programmable logic devices for motion control—A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 559–566, Feb. 2007.

[42] E. Ishii, H. Nishi, and K. Ohnishi, "Improvement of performances in bilateral teleoperation by using FPGA," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1876–1884, Aug. 2007.

[43] K. Tazi, E. Monmasson, and J. P. Louis, "Description of an entirely reconfigurable architecture dedicated to the current vector control of a set of AC machines," in *Proc. IEEE IECON*, 1999, pp. 1415–1420.

[44] A. Dinu, "FPGA neural controller for three phase sensorless induction motor drive systems," Ph.D. dissertation, De Montfort Univ., Leicester, U.K., 2000.

[45] J. Khor, "Intelligent fuzzy logic control of generators," Ph.D. dissertation, De Montfort Univ., Leicester, U.K., 1999.

[46] E. Galvan, A. Torralba, and L. G. Franquelo, "ASIC implementation of a digital tachometer with high precision in a wide speed range," *IEEE Trans. Ind. Electron.*, vol. 43, no. 6, pp. 655–660, Dec. 1996.

[47] S. Trimberger, "A reprogrammable gate array and applications," *Proc. IEEE*, vol. 81, no. 7, pp. 1030–1041, Jul. 1993.

[48] W. Wolf, *FPGA-Based System Design*. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[49] P. J. Ashenden, *The Designer's Guide to VHDL*. San Mateo, CA: Morgan Kaufmann, 1995.

[50] S. Palnitkar, *Verilog HDL, A Guide to Digital Design and Synthesis*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[51] *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076-1993.

[52] A. A. Jerraya, H. Ding, P. Kission, and M. Rahmouni, *Behavioral Synthesis and Component Reuse With VHDL*. Norwell, MA: Kluwer, 1998.

[53] T. Riesgo, Y. Torroja, and E. De la Torre, "Design methodologies based on hardware description languages," *IEEE Trans. Ind. Electron.*, vol. 46, no. 1, pp. 3–12, Feb. 1999.

[54] E. Christen and K. Bakalar, "VHDL-AMS-a hardware description language for analog and mixed-signal applications," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 10, pp. 1263–1272, Oct. 1999.

[55] F. Aubepart, C. Girerd, Y. A. Chapuis, P. Poure, and F. Braun, "ASIC implementation of direct torque control for induction machine: Functional validation by power and control simulation," in *Proc. PCIM Conf.*, 1998, pp. 251–260.

[56] M. Cirstea, "Electronic systems integrated modelling and optimized digital controller prototyping—A novel (HDL) approach," *IEEE Ind. Electron. Soc. Newsl.*, vol. 52, no. 3, pp. 11–13, Sep. 2005.

[57] F. Ricci and H. Le-Huy, "An FPGA-based rapid prototyping platform for variable-speed drives," in *Proc. IEEE IECON*, 2002, pp. 1156–1161.

[58] DSPACE, *Data Book*, 2006. [Online]. Available: http://www.dspaceinc.com

[59] L. Charaâbi, E. Monmasson, and I. Slama-Belkhodja, "Presentation of an efficient design methodology for FPGA implementation of control system application to the design of an antiwindup PI controller," in *Proc. IEEE IECON*, 2002, pp. 1942–1947.

[60] S. Nabi, M. Balike, J. Allen, and K. Rzemien, "An overview of hardware-in-the-loop testing systems at visteon," in *Proc. SAE Conf.*, 2004, pp. 13–22.

[61] H. Hanselmann, "Advances in desktop hardware-in-the-loop simulation," in *Proc. SAE Conf.*, 1997, CD-ROM.

[62] M. Gomez, "Hardware in the loop simulation, embedded systems," *Design on line magazine*. [Online]. Available: http://www.embedded.com/story/OEG20011129S0054. Paper posted on 30th Nov. 2001.

[63] Z. Salcic, C. Jiaying, and N. Sing Kiong"A floating-point FPGA-based self-tuning regulator," *IEEE Trans. Ind. Electron.*, vol. 53, no. 2, pp. 693–704, Apr. 2006.

[64] R. Andraka, "A survey of CORDIC algorithms for FPGAs," in *Proc. ACM/SIGDA Conf.*, 1998, pp. 191–200.

[65] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Mag.*, vol. 6, no. 3, pp. 4–19, Jul. 1989.

[66] F. Zhengwei, J. E. Carletta, and R. J. Veillette, "A methodology for FPGA-based control implementation," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 6, pp. 977–987, Nov. 2005.

[67] D. Menard and O. Sentieys, "Automatic evaluation of the accuracy of fixed-point algorithms," in *Proc. IEEE/ACM Conf. Des., Autom. and Test Eur.*, 2002, CD-ROM.

[68] T. Trimberger, J. A. Rowson, C. R. Lang, and J. P. Gray, "A structured design methodology and associated software tools," *IEEE Trans. Circuits and Systems*, vol. 28, no. 7, pp. 618–634, Jul. 1981.

[69] *Free access IP module internet site*. [Online]. Available: http://www.opencores.org/

[70] H. Calderon, C. Elena, and S. Vassiliadis, "Soft core processors and embedded processing: A survey and analysis," in *Proc. Safe ProRisc Workshop*, 2005, CD-ROM.

[71] K. Kebbati, Y. A. Chapuis, and F. Braun, "IP modules for motor control FPGA/ASIC integration," in *Proc. IFIP Conf.*, 2001, pp. 385–390.

[72] *A Software Modularity Strategy for Digital Control System Motor*, SPRU485A, Aug. 2001, revised Oct. 2003.

[73] T. Grandpierre, C. Lavrenne, and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessor," in *Proc. CODES 7th Int. Workshop Hardware/Software Co-Design Conf.*, 1999, CD-ROM.

[74] E. Monmasson, J. C. Hapiot, and M. Granpierre, "Analysis of a current controller for AC drives entirely based on FPGAs," in *Proc. ICEM Conf.*, Sep. 1994, vol. 3, pp. 1–5.

[75] S. Berto, A. Paccagnella, M. Ceschia, S. Bolognani, and M. Zigliotto, "Potentials and pitfalls of FPGA application in inverter drives—A case study," in *Proc. IEEE ICIT Conf.*, 2003, vol. 1, pp. 500–505.

[76] Y. A. Chapuis, J. P. Blonde, and F. Braun, "FPGA implementation by modular design reuse mode to optimize hardware architecture and performance of AC motor controller algorithm," in *Proc. 11th EPE-PEMC Conf.*, 2004, CD-ROM.

[77] K. V. Ling, S. P. Yue, and J. M. Maciejowski, "Model predictive control on a chip," in *Proc. IEEE ACC Conf.*, 2006, CD-ROM.

[78] A. Fratta, G. Griffero, and S. Nieddu, "Comparative analysis among DSP and FPGA-based control capabilities in PWM power converters," in *Proc. IEEE IECON*, 2004, vol. 1, pp. 257–262.

[79] F. Blaabjerg, P. C. Kjaer, P. O. Rasmussen, and C. Cossar, "Improved digital current control methods in switched reluctance motor drives," *IEEE Trans. Power Electron.*, vol. 14, no. 3, pp. 563–572, May 1999.

[80] O. Garcia, P. Zumel, A. de Castro, J. A. Cobos, and J. Uceda, "An automotive 16 phases DC-DC converter," in *Proc. IEEE PESC*, 2004, vol. 1, pp. 350–355.

[81] J. Chen and P. C. Tang, "A sliding mode current control scheme for PWM brushless DC motor drives," *IEEE Trans. Power Electron.*, vol. 14, no. 3, pp. 541–551, May 1999.

[82] K. Bondalapati and V. K. Prasanna, "Reconfigurable computing systems," *Proc. IEEE*, vol. 90, no. 7, pp. 1201–1217, Jul. 2002.

[83] E. Monmasson, B. Robyns, E. Mendes, and B. De Fornel, "Dynamic reconfiguration of control and estimation algorithms for induction motor drives," in *Proc. IEEE ISIE Conf.*, 2002, pp. 828–833.

[84] E. Monmasson, H. Echelard, and J. P. Louis, "Reconfiguration dynamique d'algorithmes MLI," *Rev. Int. Génie Electr.*, vol. 5, no. 1, pp. 123–134, Mar. 2002. (in French).

[85] N. Chujo, T. Hashiyama, T. Furuhashi, and S. Okuma, "Reconfigurable controllers," in *Proc. IPEC Conf.*, 2000, CD-ROM.

[86] NEuroNet Roadmap consortium, *Future prospects for neural networks*, European Network of Excellence under ESPRIT III programme of the European Commission, 1994–1998. [Online]. Available: http://www.kcl.ac.uk/neuronet/about/roadmap/hardware.html

[87] B. M. Wilamowski, R. C. Jaeger, and M. O. Kaynak, "Neuro-fuzzy architecture for CMOS implementation," *IEEE Trans. Ind. Electron.*, vol. 46, no. 6, pp. 1132–1136, Dec. 1999.

[88] "Special section on 'Neural networks for robotics'," *IEEE Trans. Ind. Electron.*, vol. 44, no. 6, pp. 746–768, Dec. 1997.

[89] L. C. Jain, "Special section on "Fusion of neural nets, fuzzy systems and genetic algorithms in industrial applications"," *IEEE Trans. Ind. Electron.*, vol. 46, no. 6, pp. 1049–1050, Dec. 1999.

[90] C.-Y. Huang, T.-C. Chen, and C.-L. Huang, "Robust control of induction motor with a neural-network load torque estimator and a neural-network identification," *IEEE Trans. Ind. Electron.*, vol. 46, no. 5, pp. 990–998, Oct. 1999.

[91] J. CatalaiLopez, L. Romeral, A. Arias, and E. Aldabas, "Novel fuzzy adaptive induction motor drive," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1170–1178, Jun. 2006.

[92] F.-J. Lin and P.-H. Shen, "Robust fuzzy neural network sliding-mode control for two-axis motion control system," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1209–1225, Aug. 2006.

[93] J. Chia-Feng and C. Jung-Shing, "Water bath temperature control by a recurrent fuzzy controller and its FPGA implementation," *IEEE Trans. Ind. Electron.*, vol. 53, no. 3, pp. 941–949, Jun. 2006.

[94] Y. Chin Tsu, W. Wan-De, and L. Yen Tsun, "FPGA realization of a neural-network-based nonlinear channel equalizer," *IEEE Trans. Ind. Electron.*, vol. 51, no. 2, pp. 472–479, Apr. 2004.

[95] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 1, pp. 28–44, Jan. 1973.

[96] S. Sanchez-Solano, R. Senhadji, A. Cabrera, I. Baturone, C. J. Jimenez, and A. Barriga, "Prototyping of fuzzy logic-based controllers using standard FPGA development boards," in *Proc. 13th IEEE Int. Workshop RSP*, 2002, CD-ROM.

[97] S. Poorani, T. V. S. Urmila Priya, K. Udaya Kumar, and S. Renganarayanan, "FPGA based fuzzy logic controller for electric vehicle," *J. Inst. Electr. Eng., Singapore*, vol. 45, no. 5, pp. 1–14, 2005.

[98] D. Kim, "An implementation of fuzzy logic controller on the reconfigurable FPGA system," *IEEE Trans. Ind. Electron.*, vol. 47, no. 3, pp. 703–715, Jun. 2000.

[99] E. Lago, M. A. Hinojosa, C. J. Jiménez, A. Barriga, and S. Sánchez-Solano, "Implementation of fuzzy controllers," in *Proc. DCIS Conf.*, 1997, pp. 715–720.

[100] F. J. Lin, D. H. Wang, and P. K. Huang, "FPGA-based fuzzy sliding-mode control for a linear induction motor drive," *Proc. Inst. Electr. Eng.—Electr. Power Appl.*, vol. 152, no. 5, pp. 1137–1148, Sep. 2005.

[101] A. Klepaczko, A. Napieralski, R. Kielbik, and J. M. Moreno, "Hardware implementation of programmable neural networks," in *Proc. Nanotech*, 2003, vol. 3, pp. 115–118.

[102] A. Nelson and T. Marcelo, "Custom architectures for fuzzy and neural networks controllers," *J. Comput. Sci. Technol.*, vol. 2, no. 7, pp. 9–15, Oct. 2002.

[103] M. Trimborne, "Optimizing intelligent DAQ devices with NI LabVIEW 8," *Nat. Instrum. News*, pp. 8–9, Feb. 2006.

[104] C. Ortega and A. M. Tyrell, "A hardware implementation of an embryonic architecture using virtex FPGAs," in *Proc. Int. Conf. Evolvable Syst.: From Biol. Hardware Conf.*, 2000, pp. 155–164.

[105] J. S. Kelly *et al.*, "Design and implementation of digital controllers for smart structures using field programmable gate arrays," *Smart Mater. Struct.*, vol. 6, no. 5, pp. 559–572, Oct. 1997.

[106] Sercos North America, "Motion control interface comes as FPGA code," *Engineeringtalk*, on line design news journal, Aug. 12, 2005. [Online]. Available: http://www.engineeringtalk.com/news/ser/ser111.html

[107] W. Naouar, L. Charaabi, E. Monmasson, and I. Slama-Belkhodja, "Realization of a library of FPGA reconfigurable IP-core functions for the control of electrical systems," in *Proc. EPE-PEMC Conf.*, 2004, CD-ROM.

[108] I. Takahachi and T. Nogushi, "A new quick response and high efficiency control strategy of an induction motor," *IEEE Trans. Ind. Appl.*, vol. IA-28, no. 5, pp. 820–827, Sep./Oct. 1986.

[109] A. A. Naassani, E. Monmasson, and J. P. Louis, "Synthesis of direct torque and rotor flux control algorithms by means of sliding mode theory," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 785–799, Jun. 2005.

[110] E. Monmasson, A. A. Naassani, and J. P. Louis, "Extension of the DTC concept," *IEEE Trans. Ind. Electron.*, vol. 48, no. 3, pp. 715–717, Jun. 2001.

[111] L. Charaâbi, E. Monmasson, A. Naassani, and I. Slama-Belkhodja, "FPGA-based implementation of DTSFC and DTRFC algorithms," in *Proc. IEEE IECON*, 2005, CD-ROM.

[112] L. Charaâbi, E. Monmasson, A. Naassani, I. Slama-Belkhodja, and M. H. Belmimoun, "Choosing FPGA or DSP for control algorithms. The case of the DTC," in *Proc. Electrimacs Conf.*, 2005, CD-ROM.

[113] M. G. Simoes *et al.*, "Design and performance evaluation of a fuzzy-logic-based variable speed wind generation system," *IEEE Trans. Ind. Appl.*, vol. 33, no. 4, pp. 956–965, Jul./Aug. 1997.

[114] G. De Michell and R. K. Gupta, "Hardware/software co-design," *Proc. IEEE*, vol. 85, no. 3, pp. 349–365, Mar. 1997.

Eric Monmasson (M'97–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, Toulouse, France, in 1989 and 1993, respectively.

He is currently a Professor and Head of the Institut Universitaire Professionnalisé de Génie Electrique et d'Informatique Industrielle (IUP GEII), University of Cergy-Pontoise, Cergy-Pontoise, France. His current research interests, in the Laboratoire Systèmes et Applications des Technologies de l'Information et de l'Energie (UMR CNRS 8029), are the advanced control of electrical motors and generators and the use of field-programmable gate arrays (FPGAs) for industrial control systems. He is the author or coauthor of three book chapters and more than 90 scientific papers.

Prof. Monmasson is a member of the steering committee of the European Power Electronics Association and of the n° 1 Technical Committee of the IMACS. He is also a Referee for various IEEE TRANSACTIONS, *Institution of Engineering and Technology Proceedings*, and *Revue Internationale de Génie Electrique*.

Marcian N. Cirstea (M'97–SM'04) received the degree in electrical engineering from Transilvania University of Brasov, Brasov, Romania, in 1990, and the Ph.D. degree from Nottingham Trent University, Nottingham, U.K., in 1996.

He is currently Head of the Design and Technology Department at Anglia Ruskin University, Cambridge, U.K. He had previously been with De Montfort University, Leicester, U.K., and Transilvania University of Brasov. He has published several technical books and about 95 peer-reviewed papers. His research focus is on digital circuit design, control systems for power electronics, holistic modeling of electronic systems, field-programmable gate arrays (FPGAs), and application-specific integrated circuit design. He has delivered five international courses/tutorials on very high-speed integrated circuits hardware description language digital controller design applied to power electronic systems modeling and FPGA controller prototyping.

Prof. Cirstea is Founder and Chairman of the Technical Committee on Electronic Systems on Chip of the IEEE Industrial Electronics Society, a member of the Institution of Engineering and Technology (IET) and a Chartered Engineer in the U.K. He is also a Referee for Elsevier, IEEE TRANSACTIONS, and the *IET Proceedings*, and an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. Three of his IEEE conference papers have received awards.