7

FPGA Based Powertrain Control for Electric Vehicles

Ricardo de Castro, Rui Esteves Araújo and Diamantino Freitas
Faculdade de Engenharia da Universidade do Porto
Portugal

1. Introduction

Current legislation in European countries, as well as in other parts of the world, is putting stricter limits on pollutant emissions from road vehicles. This issue, in conjunction with the increase awareness of consumer for the environmental problems, will require the development of new clean propulsion systems in disruption with the current mobility solutions based on internal combustion engine. Electric, hybrid and fuel-cells vehicles (Chan, 2007) are now recognized as an indispensable mean to meet the challenges associated with sustainable mobility of people and goods. In this paradigm shift, the electric motor (EM) will assume a key role in the propulsion of future vehicles and, unlike vehicles based on internal combustion engines, the high energy and power densities will facilitate the development of new powertrains configurations. In particular, multi-motor configurations, where several EMs are allocated to each driven wheel of the vehicle, represent an attractive configuration for electric vehicles (EVs), due to the independent wheel torque control and the elimination of some mechanical systems, like the differential. These features, allied with the fast dynamics of EMs, are being explored to increase the vehicle maneuverability and safety (Geng et al., 2009) and improve the performance of the EV traction system (Hori, 2004).

To cope with this rise in functional and computational complexity that current (and future) EVs require, in this work we explore the new Field-programmable Gate Array (FPGA) platform to address the powertrain control, i.e. involving the electric motor and the power converters control, of multi-motor EVs.

In the past two decades, motor control applications have been dominated by software based solutions implemented in DSPs (Digital Signal Processors), due to the low cost and ease of programming (Cecati, 1999). However, these DSP solutions are facing increasing difficulties to respond to the ever-increasing computational, functional and timing specifications that modern industrial and vehicular applications require (Monmasson & Cirstea, 2007). For instance, when single-core DSP based solutions needs to incorporate complex and time-critical functions, e.g. multi-motor control, the sequential processing of this approach decrease the controller bandwidth (see Fig. 1), which may compromise the application timing specification. Multi-core DSPs are a possible alternative to address this concern, but they also add costs and interconnection complexity. Consequently, in the last years, FPGAs received an increased interest by the academy and industry as an option to offload time-critical tasks from the DSPs (Lopez et al., 2008; Rahul et al., 2007; Ying-Yu & Hau-Jean, 1997), or even replace the DSPs control platform by a System on Chip (SOC) based on FPGAs (Idkhajine et al., 2009).

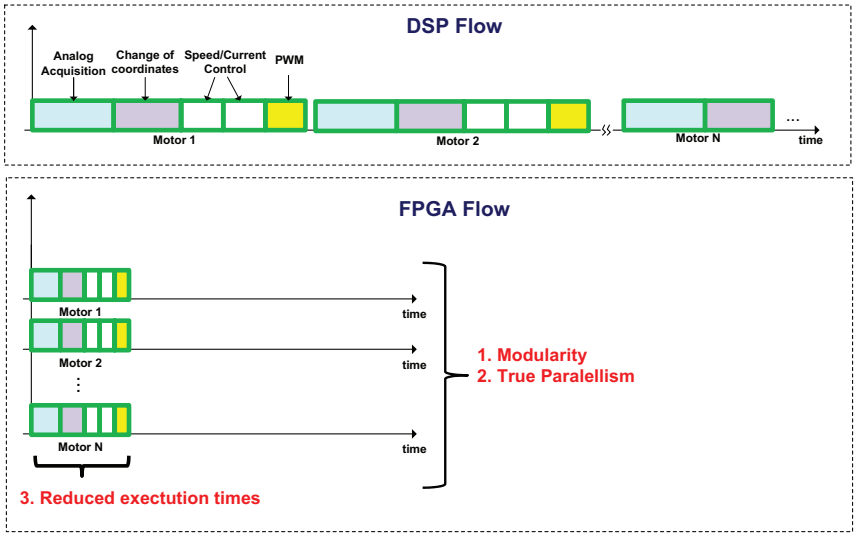


Fig. 1. Some advantages of FPGA use when controlling multiple electric motors.

The main motivation behind the FPGA introduction lies in the following properties: *i*) high processing speed; *ii*) modularity and parallel capabilities (see Fig. 1) and *iii*) hardware reconfigurability. The first feature, high processing speed, enables a reduction in execution times of the motor control algorithm, which can been explored to increase the bandwidth of torque control (Takahashi & Goetz, 2004) and decrease the discretization effects of some control and estimation techniques (Delli Colli et al., 2010; Naouar et al., 2007). Similarly, the FPGA parallelism and modularity features open up new possibilities to incorporate multi-motor control in a single chip, a useful property in multi-axis robotic manipulator arm (Jung Uk et al., 2009) and in process control applications (Tazi et al., 1999), and develop fault tolerant control systems with physical redundancy (Seo et al., 2010). Furthermore, even software based solution implemented in DSPs relies on some special hardware peripherals to boost time critical tasks, e.g. the pulse with modulators (PWM). Although these fixed hardware peripherals can be parameterized to address a wide variety of application, there are some cases where this architecture is unsuitable. The special modulation hardware blocks that multi-level converters demand is a paradigmatic example of the limitations present in the fixed DSP architecture. In this case, the FPGA hardware reconfigurability property is pivotal to effectively solve the multi-level modulation problem by allowing the development of custom PWM blocks in FPGA logic (Lopez et al., 2008). As additional benefits, the FPGAs offers the possibility to migrate, if large production volumes are needed, to ASICs - application specific integrated circuits (Fasang, 2009) - and avoid components obsolescence by emulating in FPGAs the behaviour of discontinued products, such as processors and others digital circuits (Guzman-Miranda et al., 2011).

On the other hand, FPGAs also present some potential difficulties and pitfalls that must be carefully considered before adopting this technology. Firstly, the FPGA unit cost have been the main obstacle to achieve a wider penetration of this technology, but,

owing to the continuous cost reduction in the fabrication processes, this issue is being attenuated (Rodriguez-Andina et al., 2007). In second place, the designing times and the learning curve associated with the FPGA tools can also pose some concerns. The most common approach to design these types of systems is to code directly in hardware description languages, like Verilog or VHDL. This approach has potential to produce very efficient and optimized code, but the developing time, although inferior to ASICs (Winters et al., 2006), is still very high when compared with DSPs. To overcome these difficulties, graphical based design tools, such as Simulink HDL Coder (MathWorks, 2010) or the System Generator (Xilinx, 2005), were included on the top of the FPGA tool-chain, allowing faster developing times and a jump start for designers without previous knowledge on hardware design. Another factor that helps to reduce the developing effort is the integration of processors (hard or soft) within the FPGA. As a result, the designer has an additional degree of freedom to partitioning the system project into two components, namely the software and hardware components. The former runs inside the processor and is responsible for non-time critical tasks, giving easier and faster develop environments (Barat et al., 2002), while the later is implemented in FPGA logic and handles the faster algorithms.
Finally, another traditional limitation in the majority of the FPGAs is the absence of analog peripherals, like analog to digital converters, forcing the inclusion of additional external components on the control board, which may poses undesirable limitations on embedded system with high-level of integration requirements. A notable exception to this trend is the Actel Fusion family (Actel, 2010) that supports mixed-signal processing, and is being used to develop complete SOC solutions for motor drives applications (Idkhajine et al., 2009).
To sum up, the FPGAs cost reductions made in the last years, coupled with the new graphical oriented design tools and the increasingly computational demands in industrial and vehicular electronics, are making these reconfigurable system a competitive alternative to ASICs and DSPs. For that reason, in this work, we explore the FPGAs properties to design and implement an advanced motion control library for EVs, addressing the most relevant control needs in modern powertrains, i.e. (multi) motor control, energy loss minimization and vehicle safety functionalities.
The remaining of the article is structured as follows. Section 2 presents an overview of the library developed to control the powertrain of multi-motor EV's. Next, in Section 3, this powertrain library is used to build a FPGA control system for the uCar EV prototype, including experimental validation. And finally, Section 4 discusses the conclusions and future work.

2. Overview of the powertrain IP core library

Inspired by the vector control signal processing blockset for use with Matlab - Simulink (Araujo & Freitas, 1998), in recent years our team developed a set of Intellectual Properties (IP) cores, targeting the advanced and efficient powertrain control of EV's with multiple motors (Araujo et al., 2009; 2008; de Castro et al., 2010a;b; 2009a;b). This IP core library, whose main components are depicted in Fig. 2, can be divided in 3 main classes. Firstly, a low level class was created containing basic control blocks, like Proportional and Integral controllers, modulators, mathematical transforms, etc., that were efficient and carefully designed to be reused in the control of power electronics applications and other higher level functions. Secondly, a middle layer incorporating motor control and estimation blocks was developed, in order to regulate the electric motor torque and flux, while

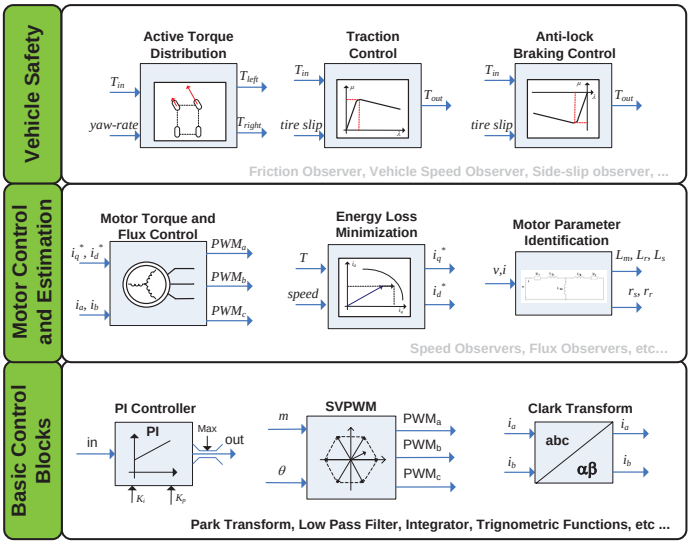


Fig. 2. Powertrain IP Core library for the control of Electric Vehicles.

minimizing the energy losses in the process. On top of that, a control layer related with the vehicle safety functions is also available, which aid the driver avoiding high tire slips during acceleration (traction control) and braking (ABS) manoeuvres and to perform active torque distribution in multi-motor EVs.

One of the main strengths in this library is that the system designer has a great flexibility to specify the controller architecture that best fits the design goals, ranging from basic powertrain controllers (just the motor control) to more advanced solutions with energy minimization and driving aid systems. The FPGA parallel features also allow the multiple instantiation of the same IP core, without degrading the control response time, which is very beneficial for multi-motor EVs where many blocks may need to be replicated in the same control unit (de Castro et al., 2010b). Moreover, the library is implemented in a generic hardware description language (Verilog), so it can be easily ported between different FPGA vendors, introducing more freedom in the hardware selection for the controller implementation.

It is worth pointing out that the IP core blocks represent "out-of-box" solutions to specific powertrain control problems, without requiring extensive development times, so the designer can focus more in the application problems and tuning and less in the technology development. Although not explicitly represented in Fig. 2, there is also available other auxiliary IP cores related with communications (UART, SPI, CAN, etc.), memory controllers, quadrature decoder, soft processors etc., which are normally supplied from the FPGA manufactures and allow the interface with external peripherals. In the remaining of this section, a brief overview of the main functionalities of the powertrain library will be discussed.

2.1 Vehicle safety

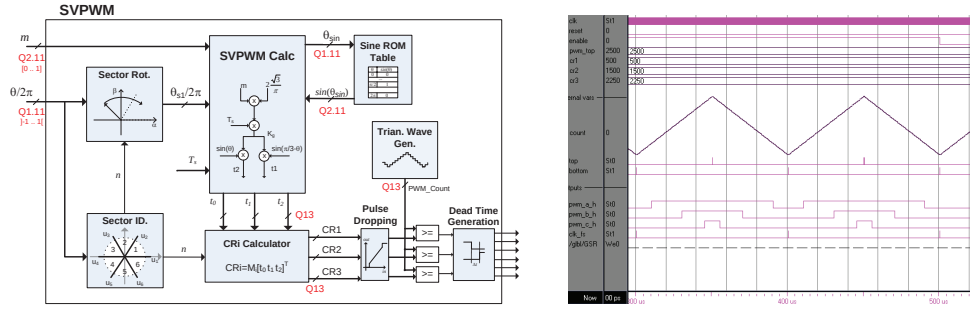
Active safety systems are of paramount importance in modern transportation applications and represent an unavoidable functionality to reduce and prevent road accidents (van Zanten, 2002). Therefore, the powertrain control of EVs must address this concern by offering driving aid mechanisms to mitigate the effects of a loss of the vehicle longitudinal and/or

lateral controllability. Such cases may appear due to aggressive driving patterns or, more importantly, by unavoidable external conditions that limit the vehicle operation range and make driving more difficult. For example, when the road present low grip conditions, such as wet tarmac, snow or ice, it is easy for the driver to apply excessive torque to the wheels and generate high tire slips. As a consequence, this excessive slip may produce tire wear, reduce the longitudinal force transmitted to the vehicle and compromise the tire capability to generate lateral forces, i.e. the steerability. In order to improve the longitudinal EV safety our powertrain control library contains IP cores for performing the traction controller (TC) and anti-lock braking functions. These control blocks are based on the sliding mode control framework and, when excessive tire slips are detected, they reduce the wheel torque magnitude to levels where the tire slip is constrained to a safe range, i.e. to a point where the tire longitudinal force is maximized. The main merit of this approach is the implementation simplicity and robustness to the model's parametric variations, such as the grip levels present on the road. These IP cores were experimental verified in our EV prototypes, demonstrating satisfactory competence in preventing excessive tire slip, particularly under low friction conditions. For further details about these control blocks, the interested reader is referred to (de Castro et al., 2010a).

A second important safety function that the powertrain control must address is the torque allocation/distribution strategy for EV's driven by multiple motors. Unlike traditional vehicles, based on internal combustion engine, the high specific power and energy densities offered by the electric motor opens up new possibilities for the powertrain configuration, in particular the distribution of several motors by the EV wheels. With these new configurations, normally composed by 2 or 4 electric motors, the traditional mechanical differentials are eliminated from the powertrain, as well as its energy losses, and the torque transmitted to each driven wheel can be independently controlled. Accordingly, this new degree of control can be explored to perform torque allocation based on the vehicle yaw-rate and side-slip control, which improve the vehicle handling (He et al., 2005) and the lateral safety (Geng et al., 2009). Although important, this block is still under internal development by our team (de Castro, 2010) and we are planning to incorporate it, as soon as possible, in the powertrain library. Nevertheless, while this block is not completely developed we are employing a simple constant torque allocation (de Castro et al., 2009b), whose features will be discussed in a later section.

2.2 Motor control, identification and energy minimization

To address the control of EVs based on induction motors, the well known indirect field oriented method (Araujo, 1991; Novotny & Lipo, 1996) was incorporated in the library, achieving a decoupled control of torque and flux by regulating the motor currents, which are formulated in the synchronous reference frame and will be briefly discussed in the next section. Besides the decoupled control, the current feedback loops, on which the vector control builds, also increases the system robustness against load overloads, peak current protection and compensation of non-linear effects (e.g. semiconductor voltage drop, dead-times, DC-link voltage variations, etc.), making this approach very attractive for EV applications. Furthermore, this module can also be used, with little modifications, to control other types of motors frequently employed in the EV applications, such as permanent magnet and brushless DC motors (Araujo et al., 2009), and be easily replicated inside the control unit to address EVs with multi-motor configurations (de Castro et al., 2010b).



(a) (b)

Fig. 3. Space Vector PWM implementation (a) and simulation validation (b).

Another important feature of the powertrain library is the energy loss minimization block of induction motors (IMs). It is a well known fact that IMs, although robust and cheap, are less efficient than other electric motors, like the permanent magnets motors (Zeraoulia et al., 2006). To partially attenuate this drawback, our IP core library incorporates a loss minimization algorithm (Araujo et al., 2008) which, based on the IM losses model, finds the flux current setpoint that minimizes the energy losses for each operating point (torque, speed) of the motor. Experimental studies conducted in our prototypes showed that, in urban driving cycles, energy savings between 10% and 20% can be achieved, which contribute to increase the vehicle efficiency and range per charge metric (Araujo et al., 2008). Since the motor control and the energy minimization are model based, the effectiveness of these algorithms relies on good estimates of the motor parameters. To address this limitation, our team developed an additional IP core for performing the identification of the IM model parameters, whose output data is employed to tune the motor controller and the loss minimization algorithm (Cerqueira et al., 2007).

2.3 Basic control blocks

The majority of the control blocks that comprise the low layer depicted in Fig. 2 are intrinsic connected with the operations needed in the motor control algorithms, such as the pulse-width modulators, mathematical transformations and the linear PI controllers. To start with, the analysis and controller design of electric motors, in particular induction and synchronous, is greatly simplified if the equations of the motor model are represented in a rotating coordinate systems. These change of coordinates are normally known as the Park and Clark transformations and lays the mathematical foundations on which the vector control of induction and synchronous motors builds (Novotny & Lipo, 1996). Therefore, these transformations were added to the powertrain library, and, due to the high amount of multiplication operation involved, special attention was taken in the efficient use of the multiplier operator, employing time-sharing methodologies (de Castro et al., 2009a). Along with these change of coordinates, the pulse-width modulators (PWM) represent another fundamental cornerstone to regulate the energy flow in modern power electronics (Kazmierkowski et al., 2002). Since three phase electric motors are the most common type in EVs applications, three-phase voltage source converters are a natural choice to feed the motor. To control these converters we implemented the Space Vector PWM (SVPWM) method which, compared with the carrier-based modulation

techniques, allows a 15% increase in the linear zone of operation and a low current distortion (Kazmierkowski et al., 2002). The SVPWM implementation, depicted in Fig. 3, was designed using polar coordinates, represented in the stationary reference frame (α, β), having as inputs the normalized magnitude (m) and angle (θ) of the desired voltage vector to be applied to the motor. Based on this information, and employing simple trigonometric relations (de Castro et al., 2009b), the SVPWM calculates the duty cycles to be applied to each arm in the inverter. Before generating the final PWM signals, dead-times are inserted in the switching signals, as a mean to prevent upper and lower arm short-circuits, and pulses less than a minimum width (2 times the inverter dead-times) are dropped to ensure the proper operation of the inverter when high modulation indexes are requested.
The modulators and the mathematical transformations described above are usually employed in conjunction with Linear PI controllers, i.e. the motor voltage vector is defined by current PI controllers that operate in a synchronous reference frame. This class of linear controllers are widely used, not only in the motor controller, but also in many control application due to the implementation simplicity and robustness to constant disturbances. For that reason, a discrete version of the PI controller was coded in Verilog and reused in several high-level control blocks. Other functions, like low pass filters to attenuate the noise effects in the feedback loops, the calculation of trigonometric functions and inverse numbers were also included in the library.
And finally, it is worth point out that, even though these low level modules have been implemented in the context of powertrain control, they can also be reused in other industrial and power electronics applications, e.g. with three phase rectifiers and grid interface, a feature to be explored in future works.

3. The uCar case study

After describing the main components of the powertrain library, we will discuss in this section how these modules can be assembled in order to build a complete controller for an bi-motor EV prototype, named uCar. In order to reduce the FPGA resource usage, as well as the costs, we will focus our attention on designing a minimal EV control system, comprising only essential functions to operate the vehicle. As a consequence, the high-value functions associated with the energy minimization and vehicle safety will not be considered in the current controller design.
The architecture of this minimal control system, depicted in Fig. 4, is built around a single FPGA XC3S1000 unit, and has as main features the ability to control two inductions motors, thanks to the double instantiation of the motor control IP core, and a soft processor, the PicoBlaze (Xilinx, 2010), used to manage the global operation of the EV controller. To complement the operation of the motor control IP core and the soft processor, the FPGA unit has a set of peripheral interfaces (UART, SPI, ADC interface, quadrature decoder, etc.), enabling the FPGA to interact with the external entities, like sensors, driving commands and host systems (computer).
The processing flow in the EV controller evolves as follows. In first place, the soft processor acquires the analog commands requested by the driver, such as throttle, brake and also digital signals, like the key and vehicle direction, using for this purpose the I/O and ADC peripherals, which are accessed by the soft processor via a dedicated data bus. Based on this information a state machine (task 2), responsible for defining the EV operating mode (run, stop, fault, etc.), is updated; if the EV is in run mode, then the soft processor executes the

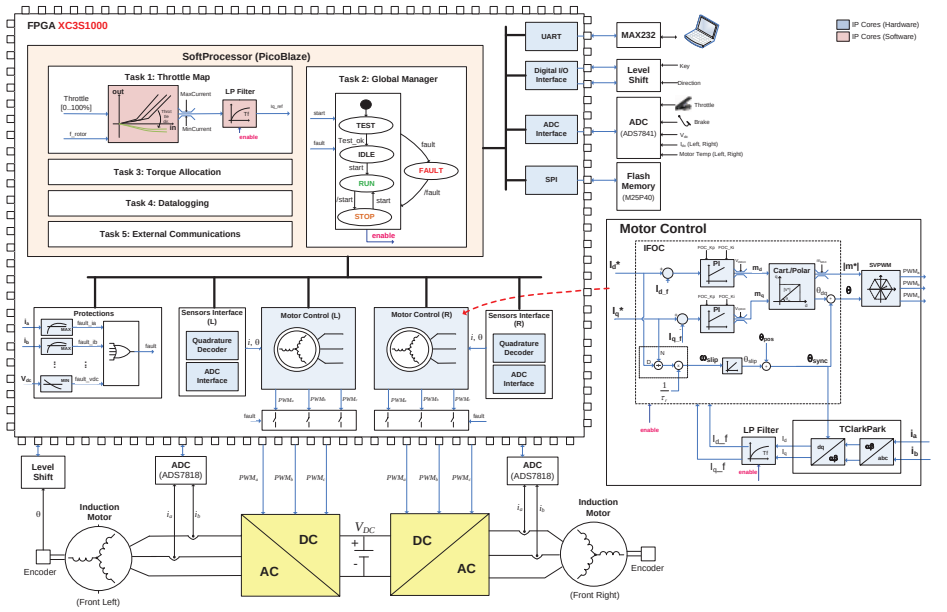


Fig. 4. Diagram of the uCar EV controller.

throttlemap module, where the throttle signal is mapped on a current reference (task 2), and the torque allocation task (3) to define the current (or torque) level that must be imposed to each electric motor in the uCar. After completing task 3, the soft processor sends the current references to the motor control IP cores that command the power converters to achieve the motor torque and flux regulation.

An interesting feature that is worth mention is the hardware/software partitioning on which the EV controller relies. On one hand, the algorithms that needs a fast response times or parallel execution, such as the motor control and the protection module, are directly coded in FPGA logic with boosts the system performance. On the other hand, for implementing slow algorithms and designing control sequences based on state machines, like the EV global state machine or the communication protocols, the soft processor offers a better solution, easier to program and with faster developing times (Xilinx, 2010).

3.1 Soft processor tasks

The soft processor, i.e. a processor implemented in FPGA logic, employed in the current system was the 8-bit PicoBlaze processor, which was selected due to the low resource usage and the free access to the soft processor VHDL source-level code. This processor is used to perform 5 non time-critical tasks (throttlemap, global manager, torque allocation, datalogging and external communications), which are described in this section.

3.1.1 Task 1: ThrottleMap

The ThrottleMap task implements a static function that translates the throttle signal (t_i), defined by the driver, in a current reference (in our case i_q, which is proportional to the motor torque) to be applied to the motor controller. This mapping is normally composed

by a initial dead-zone, to avoid the pedal offset, a liner gain, followed by a saturation to limit the maximum torque, and is described by the following relation:

$$t_1 = \begin{cases} 0, & \text{if } t_i - \underline{t} < 0 \\ t_i - \underline{t}, & \text{if } 0 \leq t_i - \underline{t} \leq 100 \\ 100, & \text{if } t_i - \underline{t} > 100 \end{cases} \quad (1)$$

where $t_i, t_1 \in [0\%, 100\%]$ are the normalized input and output throttle signals and $\underline{t}$ represents the throttle deadzone. A second feature that needs to be addressed by the throttlemap is the "engine braking" emulation: in vehicles based on internal combustion engine, when the driver release the throttle pedal the vehicle experience a deceleration, caused by "engine braking". In EVs this behaviour must be emulated by the control unit, which, in our controller, is performed by the ThrottleMap module and described by:

$$t_2 = \begin{cases} -t_{brk}, & \text{if } t_i - \overline{t} < -t_{brk} \\ t_i - \overline{t}, & \text{if } -t_{brk} \leq t_i - \overline{t} \leq 0 \\ 0, & \text{if } t_i - \overline{t} > 0 \end{cases} \quad (2)$$

where $-t_{brk}$ is the minimum value for the equivalent "engine braking" torque and $\overline{t}$ the throttle point where the braking begins. Putting (1) and (2) together, and considering a scale factor k_{t2i} that translates the throttle signal to current, the final motor current reference (i_q^*) is defined as:

$$i_q^* = (t_1 + t_2 i_{move}(v))\, k_{t2i} \quad (3)$$

where $i_{move}(v) \in \{0, 1\}$ is a flag that is activated when the vehicle is moving above a given speed threshold. This flag intents to disable the "engine braking" when the vehicle is close to a complete stop.
Based on several experimental roadtests conducted in our EV prototypes, it was verified that the throttlemap module is a fundamental tool to improve the driving experience, and, in particular, the "engine braking" contributes to a more pleasant and predictable driving.

3.1.2 Task 2: Global state machine

To keep track of the EV operating mode, the soft processor also runs a simple state machine (task 2), whose simplified behaviour is shown in Fig. 4. When the controller is first initiated, the state machine goes to the TEST state, where a series of validation tests (check current and voltage sensors, throttle signal, etc) are performed; if these checks do not shown any anomaly, then the state machine jumps to the IDLE state, where it waits for the start signal generated by the driver. After performing this initialization phase, the EV switches between the RUN and STOP mode, depending on the start signal defined by the driver, and if an EV protection becomes active, the FAULT mode is enabled. Since the switching between the state machine depends on external digital signals, which are subject to be bouncing and other fast transient disturbances, a preprocessing was also included in the task 2 in order to filter these signals.

3.1.3 Task 3: Constant torque allocation

While the active torque allocation method discussed in Section 2.1 is not implemented, a uniform torque distribution strategy has been used, with both motor controllers receiving the same torque reference ($i_{q,left}^* = i_{q,right}^* = i_q^*$), defined by the throttle position and throttlemap.

This strategy emulates the basic features of a single axis mechanical open differential, widely used in conventional vehicles. Typically, the open differential has 2 objectives: *i*) transfer the motor power to the driven wheels, applying the same torque to both wheels; *ii*) allow the driven wheels to rotate at different speeds (critical feature during the vehicle cornering). In the case of multi-motor EV, the first feature can be easily emulated by applying the same current/torque reference to both motor controllers, and assuming that both motors have similar characteristics. The second problem addressed by the mechanical differential, related with different wheel speeds, is not an issue in a multi-motor EV configuration. Note that in a multi-motor configuration each motor is free to rotate at any speed, and can be seen as an independent system: all motors receive equal value of acceleration/braking torque, but the load torque experience by each motor is different, especially during cornering manoeuvres, which naturally leads to different wheel speeds. These observations are corroborated by the experimental results obtained in the multi-motor uCar prototype (de Castro et al., 2009b) and allow us to employ a simple and low cost torque allocation to operate the EV.

3.1.4 Task 4 and 5: Datalogging and external communications

The soft processor also run a datalogger task (4) that stores, in an external 4 Mbit flash memory (M25P40) accessed by an SPI interface, the evolution of several EV variables, helpful for debug purposes and characterization of the vehicle behaviour. Finally, a communication task (5) is responsible for implementing a simple communication protocol (Oliveira et al., 2006), on top of a RS232 link, to enable the interaction between the FPGA controller and a host system, e.g. a computer.

3.2 Motor Controller (MC)

In order to address the motor controller needs, which are the most time-critical and computational intensive in the EV controller, we employed some of the IP cores from the powertrain library described in Section 2. As a result, the SVPWM, PIs and the mathematical transformations were assembled together to build the indirect field oriented control of induction motors, represented in the right part of Fig. 4, offloading much of the processing demands from the soft processor to the FPGA logic. Due to the parallel features of the FPGA, the second motor in the uCar prototype can be straightforwardly handled by instantiating a second motor control (MC) module, which does not degrade the bandwidth and response time of modules already in place. In addition, each MC has a dedicated module to acquire the feedback information from the motor position, though an incremental encoder, and motor currents, as well as a protection module to detect and process faults produced by overcurrents, over and under-battery voltage and thermal overloads.

3.3 Latency and resource analysis

Figure 5 shows the latency cycles introduced by the most time-critical IP core in the EV controller: the MC sub-modules. Before beginning the mathematical calculations, the MC must acquire the motor currents. Due to the absence of an internal ADC in the FPGA, the currents measurements is done through ADCs (TI ADS7818) external to the FPGA and its value transmitted by a high-speed serial protocol. This acquisition process takes 250 latency cycles and represents the largest delay in the MC (73% of the total time). The MC computational blocks (Clark and Park transformation, PIs and SVPWM), introduces a latency of just 90 cycles. In total, the MC control cycles is performed in less than 340 cycles (6.8us),

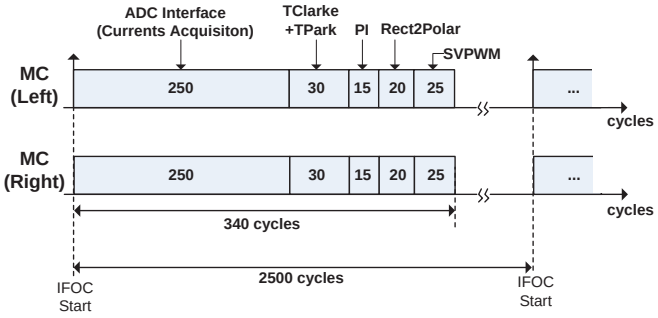


Fig. 5. Latency introduced by the MC sub-modules (the main clock in the FPGA has a frequency of 50MHz, thus 2500cyces ⇔ 50us)

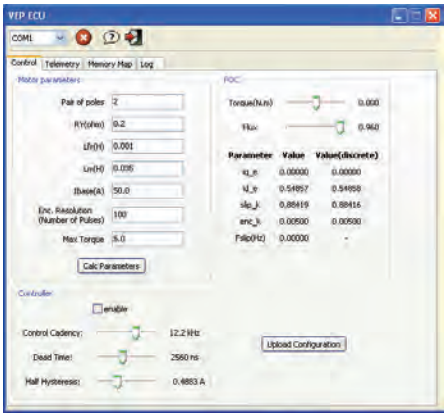
Type	Module	Slices	Mul.	BRAM.	FMax(MHz)
Motor Control	SVPWM	316	1	1	86
	TClark+TPark	212	2	1	78
	2xPI's + Cart2Polar	1012	6	1	92
	Field Weakening	59	2	1	125
Sensor Interface	ADC Interface (ADS7818)	47			190
	Quadrature Decoder	37			134
Protections	Protections	75			183
Soft Processor	PicoBlaze + SPI + UART + ...	501		3	85

Table 1. Resource utilization of the main IP cores (Note: the design tool was the ISE WebPack 8.2.03i, FPGA family: Spartan 3, Speed Grade: -5).

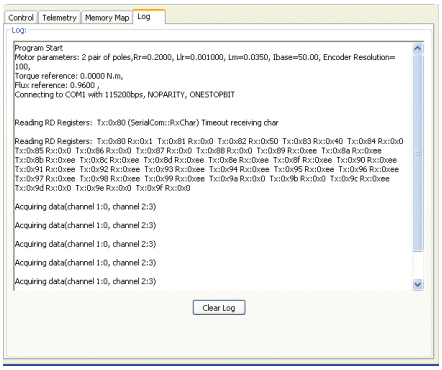
Module	Num. Instances	Slices	Mul.
Motor Control(MC)	2	3198	22
Sensor Interface	2	168	
Protections	1	75	
Soft Processor	1	501	
Others	1	789	
Total		4731 (61%)	22 (92%)

Table 2. Resource utilization of the XC3S1000 FPGA used to control the uCar prototype (Note: the design tool was the ISE WebPack 8.2.03i, FPGA family: Spartan 3, Speed Grade: -5).

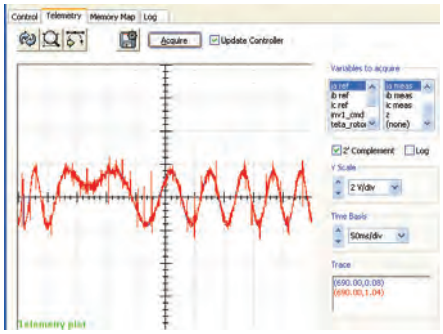
representing 14% of the 2500 cycles associated with the MC minimum execution rate (20kHz). This minimum rate is the result of the energy dissipation limits in the power semiconductors, which, in hard-switching, high current traction applications, is normally constrained to a maximum of 20kHz switching frequency. Albeit the MC modules have been specifically developed for electric traction applications, with the 20 kHz update rate limit, the low value of latency permits a higher execution rate, up to 147 kHz. This feature enables the MC modules to be reused in other industrial applications, where a high-bandwidth control of torque and



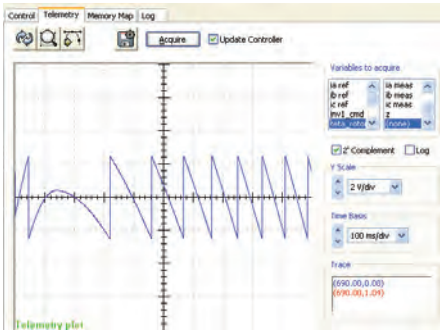
(a) Motor controller and SVPWM configuration



(b) Debug screen



(c) Telemetry plot for current regulation



(d) Telemetry plot for motor position

Fig. 6. User interfaces of the software developed to configure and debug the EV controller.

speed is necessary. Figure 5 also shows the parallel processing capabilities of FPGA, which allows multiple instantiations of the MC to run simultaneously, independently and without compromising the bandwidth of other modules.

A summary of the resource utilization in the IP cores implementation, such as slices, dedicated multipliers and Block Ram (BRAM), is presented in Tables 1 and 2. The two Motor Controllers instantiated in control unit are the most demanding on the FPGA resources, requiring 44% of the slices and 92% of the dedicated multipliers available on the chip. Although there are a considerable number of slices available (39%), the low number of free multipliers prevents the inclusion of additional MC, presenting a restriction for future improvements in this FPGA; in other words, such improvements would need an FPGA with more computational resources, thus more costly. In addition to the MC, there are also others modules to perform auxiliary functions (sensor interface, protections, soft processor), described in the previous section, and which consume 17% of the FPGA area.

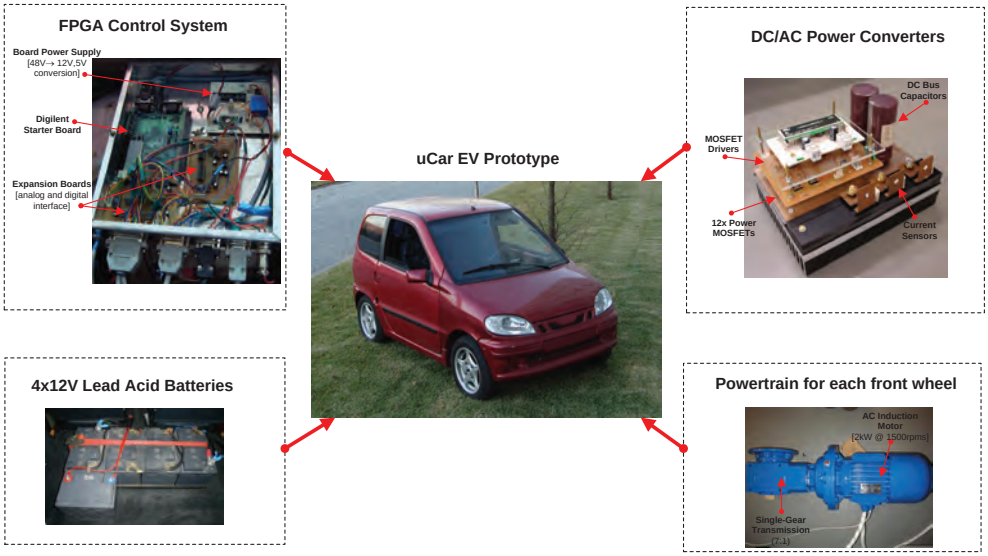


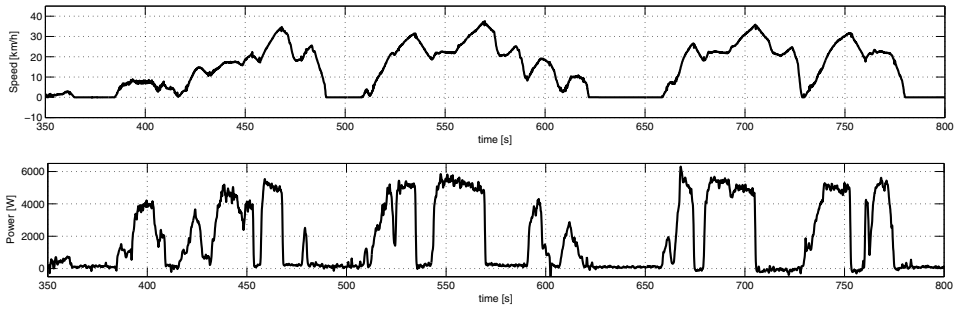
Fig. 7. uCar electric vehicle prototype.

3.4 Configuration software

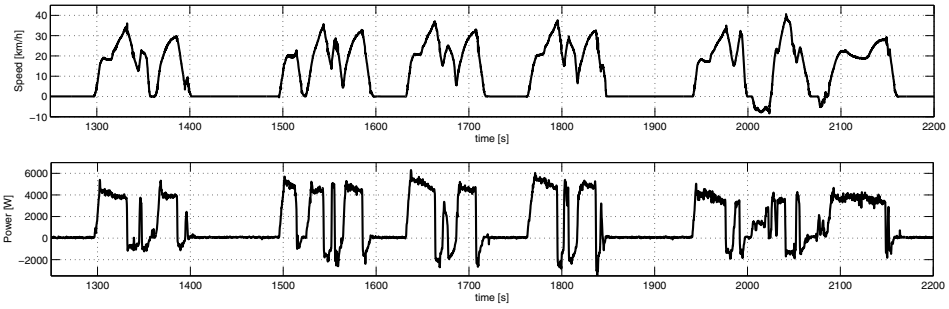
During the EV development, it is necessary to exchange configuration and debugging data with the FPGA control unit. To this aim, we built a graphical application based on the cross-platform wxWidgets library, whose main user interfaces are depicted in Fig. 6. This application, running on a convention computer, establishes a communication channel with the tasks 4 and 5, briefly described in Section 3.1.4. Based on this interface, the EV designer has the possibility to change the EV control parameters associated with the motor controller (current and flux limits, pair of poles, etc.), peripherals (encoder pulses), modulation (switching frequency, dead-times, etc.), among other modules. For debugging the controller we also have a datalogger interface (Fig. 6(c)), which enables the real-time acquisition of the EV controller variables, like the motor currents, voltages and mechanical position, providing an effective mechanism to inspect the performance of the control loops during fast transients and aid the controller tuning process.

3.5 Experimental results

In order to evaluate the control system discussed in the previous sections, an EV prototype, named uCar, was built to accommodate the electric powertrain (see Fig. 7). The vehicle is based on a two-seater quadricycle, manufactured by the MicroCar company, and is very popular among elderly people of southern Europe, mainly due to non-compulsory driving license. The original propulsion structure, based on the internal combustion engine, was replaced by a new electric powertrain composed by two electric motors (26 Vrms, 2.2 kW @ 1410 rpm), each one coupled to the front wheels by single gear (7 : 1) transmissions. Due to low cost, lead acid batteries ($4x12V@110Ah$) were selected as the main energy storage of the EV, providing a range of 40 km per charge, a sufficient autonomy for urban driving. After the conversion, the uCar prototype weights 590 kg and reaches a top speed of 45 km/h.



(a) regenerative braking OFF



(b) regenerative braking ON

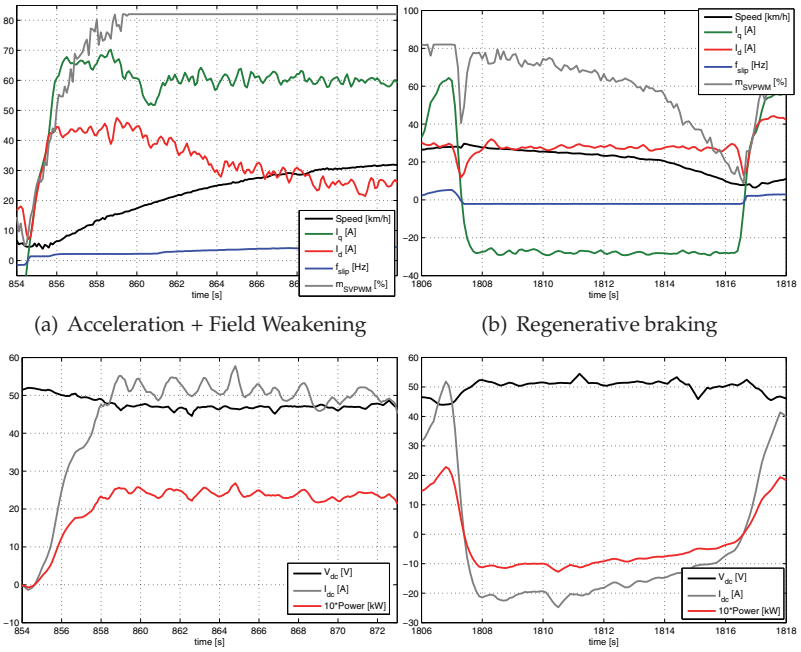
Fig. 8. Experimental results of a typical driving cycle performed by the uCar inside the university campus, with and without regenerative braking active.

All the powertrain control functions of the EV are concentrated on the Digilent Spartan 3 Start Board, containing, besides the XC3S1000 FPGA, several useful peripherals such as flash memory (2 Mbit) for storing data, serial interface for communications and 4 expansion ports for I/O with the FPGA. To extend the functionalities of these main peripherals, two additional boards were constructed and connected to the main board, containing analog to digital converters (TIADS7818 and TIADS7848) to allow the acquisition of analog variables, and voltage level shifters (3.3 $\leftrightarrow$ 5.0V) to perform the interface with the external digital I/O. This EV controller interacts with two DC/AC power converters, featuring 120Arms@30Vrms and 20kHz switching frequency, in order to regulate the current and voltage delivered to the electric motors, as discussed in the previous sections.

To validate the experimental performance of the uCar, several roadtests were conduced inside the FEUP university campus, characterized by low speed driving cycles, similar to urban conditions (see Fig. 8). From these roadtests, we selected two representative cycles for assess the influence of the regenerative braking in the energy consumption of the uCar. In the first situation, with the regenerative braking disabled, the vehicle travels approximately 2.36 km and shows consumption metrics close to 100 Wh/km (see Table 3). On the other hand, when the reg. braking is active the EV consumption decreases 13.2%, to 86.8 Wh/km, representing an important contribute to increase the EV range per charge.

Mode	Distance	Energy Delivered	Energy Regenerated	Consump.	Max. Power	Min. Power
Reg. OFF	2.37km	236.7 W.h	0W.h	99.9 Wh/km	6.3 kW	0 kW
Reg. ON	4.26km	417.6 W.h	48.3W.h	86.8 Wh/km	6.3 kW	-3.5 kW

Table 3. Performance metrics of the uCar over the driving cycles described in Fig. 8.



(a) Acceleration + Field Weakening (b) Regenerative braking

(c) DC Bus variables (acceleration) (d) DC Bus variables (reg. braking)

Fig. 9. Detailed view of the uCar (left motor) results during accelerating, field weakening and regenerative braking.

To further validate the EV control unit performance, Fig. 9 show the detailed results of the left motor controller for tree different operating modes: acceleration, field weakening and regenerative braking. The data depicted in these figures was acquired with the controller internal datalogger, which enable us to keep track of the most relevant EV variables, such as: mechanical (motor speed), energy source (voltage, current and power) and the motor controller (torque (i_q) and flux (i_d) currents, modulation index (m) and the slip frequency (f_{slip})) variables. During the acceleration mode (Fig.9(a), 9(c)), performed with the throttle at 100%, the i_q and i_d currents are set at the maximum value in order to extract the maximum motor torque and vehicle acceleration (2.2km/h/s). When the EV reaches 18km/h the motor voltage saturates at 83% and the flux current is reduced to allow the vehicle to operate in the field weakening area, with a power consumption of 2.5kW per motor. In fact, analyzing the evolution of the power supplied by the batteries during the experimental driving cycles (Fig. 8), it is interesting to note that the electric motors spend most of the time operating in this field weakening zone. Fig. 9(b) and 9(d) shows the detailed results of third EV operation

mode: the regenerative braking. In the depicted manoeuvre, the driver is requesting a torque current of -25A to decelerate the vehicle from 30 km/h to 5 km/h in 10s, which enable a conversion of 1kW peak power and emphasizing one of the most promising features in EVs: energy recovering during braking.

4. Conclusion

In this article an FPGA based solution for the advance control of multi-motor EVs was proposed. The design was build around a powertrain IP Core library containing the most relevant functions for the EV operation: motor torque and flux regulation, energy loss minimization and vehicle safety. Due to the parallel, modularity and reconfigurability features of FPGAs, this library can be reused in the development of several control architectures that best suits the EV powertrain configuration (single or multi-motor) and functional requirements. As proof of concept, the powertrain library was employed in the design of minimal control system for a bi-motor EV prototype and implemented in a low cost Xilinx Spartan 3 FPGA. Experimental verification of the control unit was provided, showing reasonable consumption metrics and illustrating the energy benefits from regenerative braking.

In future works, we are planning the inclusion, in the powertrain library, of active torque methods in order to improve the handling and safety of multi-motor EVs. On the technological level, we also intent to validate the library on EV prototypes with 4 in-wheel motors.

5. References

Actel (2010). Fusion Family of Mixed Signal FPGAs datasheet.

Araujo, R. E. (1991). *Control System of Three-phase Induction Motor based on the Principle of Field Orientation*, Master thesis, Faculdade de Engenharia da Universidade do Porto.

Araujo, R. E. & Freitas, D. S. (1998). The Development of Vector Control Signal Processing Blockset for Simulink: Philosophy and Implementation, *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*.

Araujo, R. E., Oliveira, H. S., Soares, J. R., Cerqueira, N. M. & de Castro, R. (2009). Diferencial Electronico. Patent PT103817.

Araujo, R. E., Ribeiro, G., de Castro, R. P. & Oliveira, H. S. (2008). Experimental evaluation of a loss-minimization control of induction motors used in EV, *34th Annual Conference of IEEE Industrial Electronics*, Orlando, FL, pp. 1194–1199.

Barat, F., Lauwereins, R. & Deconinck, G. (2002). Reconfigurable instruction set processors from a hardware/software perspective, *IEEE Transactions on Software Engineering* 28(9): 847–862.

Cecati, C. (1999). Microprocessors for power electronics and electrical drives applications, *IEEE Industrial Electronics Society Newsletter* 46(3): 5–9.

Cerqueira, N. M., Soares, J. R., de Castro, R. P., Oliveira, H. S. & Araujo, R. E. (2007). Experimental evaluation on parameter identification of induction motor using continuous-time approaches, *International Conference on Power Engineering, Energy and Electrical Drives*, Setubal, Portugal.

Chan, C. C. (2007). The State of the Art of Electric, Hybrid, and Fuel Cell Vehicles, *Proceedings of the IEEE* 95(4): 704–718.

de Castro, R. (2010). Main Solutions to the Control Allocation Problem, *Technical report*, Universidade do Porto.

de Castro, R., Araujo, R. E. & Freitas, D. (2010a). A Single Motion Chip for Multi-Motor EV Control, *10th International Symposium on Advanced Vehicle Control (AVEC)*, Loughborough, UK.

de Castro, R., Araujo, R. E. & Freitas, D. (2010b). Reusable IP Cores Library for EV Propulsion Systems., *IEEE International Symposium on Industrial Electronics*, Bari, Italy.

de Castro, R., Araujo, R. E. & Oliveira, H. (2009a). Control in Multi-Motor Electric Vehicle with a FPGA platform, *IEEE International Symposium on Industrial Embedded Systems*, Lausanne, Switzerland, pp. 219–227.

de Castro, R., Araujo, R. E. & Oliveira, H. (2009b). Design, Development and Characterisation of a FPGA Platform for Multi-Motor Electric Vehicle Control, *The 5th IEEE Vehicle Power and Propulsion Conference*, Dearborn, USA.

Delli Colli, V., Di Stefano, R. & Marignetti, F. (2010). A System-on-Chip Sensorless Control for a Permanent-Magnet Synchronous Motor, *IEEE Transactions on Industrial Electronics* 57(11): 3822–3829.

Fasang, P. P. (2009). Prototyping for Industrial Applications, *IEEE Industrial Electronics Magazine* 3(1): 4–7.

Geng, C., Mostefai, L., Denai, M. & Hori, Y. (2009). Direct Yaw-Moment Control of an In-Wheel-Motored Electric Vehicle Based on Body Slip Angle Fuzzy Observer, *IEEE Transactions on Industrial Electronics* 56(5): 1411–1419.

Guzman-Miranda, H., Sterpone, L., Violante, M., Aguirre, M. & Gutierrez-Rizo, M. (2011). Coping With the Obsolescence of Safety - or Mission-Critical Embedded Systems Using FPGAS, *IEEE Transactions on Industrial Electronics* 58(3): 814 – 821.

He, P., Hori, Y., Kamachi, M., Walters, K. & Yoshida, H. (2005). Future motion control to be realized by in-wheel motored electric vehicle, *31st Annual Conference of IEEE Industrial Electronics Society*.

Hori, Y. (2004). Future vehicle driven by electricity and control - Research on four-wheel-motored UOT Electric March II, *IEEE Transactions on Industrial Electronics* 51(5): 954–962.

Idkhajine, L., Monmasson, E., Naouar, M. W., Prata, A. & Bouallaga, K. (2009). Fully Integrated FPGA-Based Controller for Synchronous Motor Drive, *IEEE Transactions on Industrial Electronics* 56(10): 4006–4017.

Jung Uk, C., Quy Ngoc, L. & Jae Wook, J. (2009). An FPGA-Based Multiple-Axis Motion Control Chip, *IEEE Transactions on Industrial Electronics* 56(3): 856–870.

Kazmierkowski, M., Krishnan, R. & Blaabjerg, F. (2002). *Control in Power Electronics: Selected Problems*, Academic Press.

Lopez, O., Alvarez, J., Doval-Gandoy, J., Freijedo, F. D., Nogueiras, A., Lago, A. & Penalver, C. M. (2008). Comparison of the FPGA Implementation of Two Multilevel Space Vector PWM Algorithms, *IEEE Transactions on Industrial Electronics* 55(4): 1537–1547.

MathWorks (2010). Simulink HDL Coder 2.0 User Guide.

Monmasson, E. & Cirstea, M. N. (2007). FPGA Design Methodology for Industrial Control Systems-A Review, *IEEE Transactions on Industrial Electronics* 54(4): 1824–1842.

Naouar, M. W., Monmasson, E., Naassani, A. A., Slama-Belkhodja, I. & Patin, N. (2007). FPGA-Based Current Controllers for AC Machine Drives-A Review, *IEEE Transactions on Industrial Electronics* 54(4): 1907–1925.

Novotny, D. & Lipo, T. (1996). *Vector control and dynamics of AC drives*, Oxford University Press.

Oliveira, H. S., Soares, J. R., Cerqueira, N. M. & de Castro, R. (2006). *Veiculo Electrico de Proximidade com Diferencial Electronico*, Licenciatura thesis, Universidade do Porto.

Rahul, D., Pramod, A. & Vasantha, M. K. (2007). Programmable Logic Devices for Motion Control-A Review, *IEEE Transactions on Industrial Electronics* 54(1): 559–566.

Rodriguez-Andina, J. J., Moure, M. J. & Valdes, M. D. (2007). Features, Design Tools, and Application Domains of FPGAs, *IEEE Transactions on Industrial Electronics* 54(4): 1810–1823.

Seo, K., Yoon, J., Kim, J., Chung, T., Yi, K. & Chang, N. (2010). Coordinated implementation and processing of a unified chassis control algorithm with multi-central processing unit, *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 224(5): 565–586.

Takahashi, T. & Goetz, J. (2004). Implementation of complete AC servo control in a low cost FPGA and subsequent ASSP conversion, *Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition*.

Tazi, K., Monmasson, E. & Louis, J. P. (1999). Description of an entirely reconfigurable architecture dedicated to the current vector control of a set of AC machines, *The 25th Annual Conference of the IEEE Industrial Electronics Society*.

van Zanten, A. T. (2002). Evolution of electronic control systems for improving the vehicle dynamic behavior, *Proceedings of the International Symposium on Advanced Vehicle Control (AVEC)*, Hiroshima, Japan, pp. 7–15.

Winters, F., Nicholson, R., Young, B., Gabrick, M. & Patton, J. (2006). FPGA Considerations for Automotive Applications, *SAE 2006 World Congress and Exhibition*, Detroit, MI.

Xilinx (2005). System Generator User Guide.

Xilinx (2010). PicoBlaze 8-bit Embedded Microcontroller User Guide.

Ying-Yu, T. & Hau-Jean, H. (1997). FPGA realization of space-vector PWM control IC for three-phase PWM inverters, *Power Electronics, IEEE Transactions on* 12(6): 953–963.

Zeraoulia, M., Benbouzid, M. E. H. & Diallo, D. (2006). Electric Motor Drive Selection Issues for HEV Propulsion Systems: A Comparative Study, *IEEE Transactions on Vehicular Technology* 55(6): 1756–1764.