Design, Development and Characterisation of a FPGA Platform for Multi-Motor Electric Vehicle Control

Ricardo de Castro, Rui Esteves Araújo, Hugo Oliveira
Faculty of Engineering, University of Porto
Porto, Portugal
{sirpdc, raraujo, hugo.oliveira}@fe.up.pt

***Abstract*—Two three-phase squirrel-cage induction motors are used as a propulsion system of an electric vehicle (EV). A simple XC3S1000 FPGA is used to simultaneously control both electric motors, with field oriented control and space vector modulation techniques. To electronically distribute the torque between the two electric motors, a simple, yet effective, strategy based on a uniform torque distribution has been implemented. Experimental results obtained with a multi-motor EV prototype demonstrate the proper operation of the proposed system.**

Keywords: Electric Vehicles; Motor Drives; Field-Programmable Gate Arrays(FPGAs); Motion Control.

I. INTRODUCTION

Since the early of 1990's there has been resurgence in Electric Vehicles (EVs) research. This trend has been stimulated by various factors including rising in oil prices, the environmental concerns, the cost reduction of power electronics and motors drives, and with the constant performance improvement in lithium-ion battery technology. It is expected that in the near future EVs could have a leading role in the transport sector and, in this context, it is necessary to study and develop new advanced control systems (ACS) that take advantage of all opportunities made available by electric propulsion. The electric motors used by EVs, compared with internal combustion engines, offer significant advantages: (i) the reduced volume and weight of an electric motor allows the inclusion of the motor inside the wheel, offering a new degree of freedom, which can be exploited to improve vehicle handling and safety using active torque distribution [1]; (ii) the response times of torque generated by electric motor are much faster (up to 10 times) than internal combustion engines and hydraulic systems, which could increase the performance of traction and anti-skiding systems [2]; (iii) other advantages with motor inside the wheel it frees up space at the front of the vehicle, which could be used to improve the absorption of crash impact energy.

The typical structure used in an ACS for a multi-motor EV is based on 3 control layers (see Fig.1):

1) The top layer is composed by two controllers, one for lateral motion and the other for longitudinal movement. The first ensures that the EV meets certain handling and safety criteria involving the direct control of yaw rate and side-slip (typical examples: electronic stability control and active torque distribution). The second controller (longitudinal) maps in a longitudinal force the driver requests for acceleration/braking, taking into account the information received from navigation and driving pattern, and including features such as cruise control and collision avoidance from the active safety systems. The output of the longitudinal controller represents the total force that must be applied to the vehicle, while the lateral controller generates a yaw moment reference that must be translated into differential forces in the left/right wheels. The integration of these two controllers, which must produce the reference force for each wheel, is one of the main challenges in multi-motor EVs [3].

2) To ensure the proper application of longitudinal forces, taking into account the conditions of adhesion and the tire/road non-linearities, a traction control layer is inserted (the Anti-lock

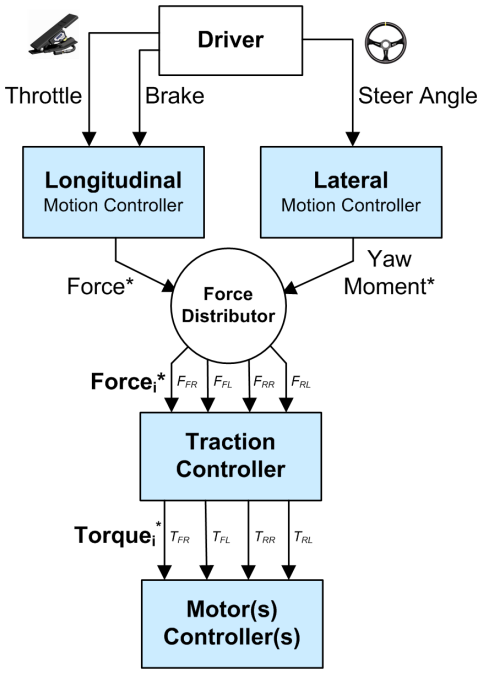


Figure 1. Typical control layers implemented in advanced control system for multi-motor EV.

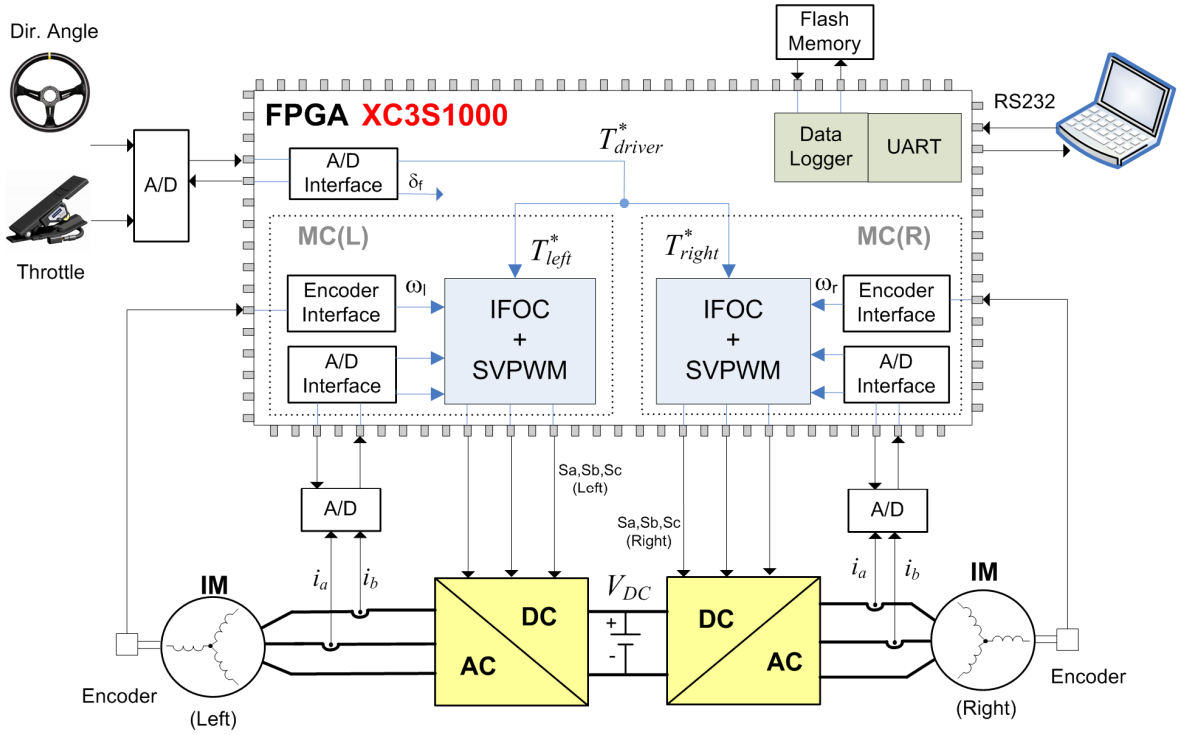


Figure 2. Architecture of propulsion control chip, implemented on a FPGA.

Braking System (ABS) is a typical example). The traction control layer manipulates the torque applied to each motor in order to prevent the wheel lock during braking (or wheel slip during acceleration), which could compromise the vehicle steerability and increase the vehicle braking distance [4].

3) The bottom layer, motor controller, is composed by the power electronics and pulse-width modulators that ensures the generation of electromagnetic torque produced by electric motors.

The integration of various control systems (motor, traction and motion control) on a single and compact chip is the ultimate goal of this work. The motor controller is the control level that requires the most processing power (PWM modulators, coordinates transformation, PI controllers, etc.). If the EV has several electrical motors, for example one per wheel, the processing requirements increases proportionately. Traditional solutions, such as DSPs (Digital Signal Processors), have some difficulties to control more than 2 motors simultaneously, a consequence of its sequential processing, which normally leads to the distribution of various DSPs (1 per motor). The FPGAs (Field-Programmable Gate Arrays) do not have these kind of limitations, and offer attractive features, such as parallel processing, high calculation capacity, modularity, etc., allowing the merging, in a single chip, of all motors controllers (running in parallel) and, possibly, of all control layers (motor, traction and motion). Furthermore, in recent years FPGAs have been successfully applied to motor controller applications, and, by reducing the execution times, have improved the quality of the controlled variables [5]. This technology has also received attention by some industrial manufacturers, highlighting the Accelerator™ platform developed by International Rectifier [6], oriented to control position in industrial applications, which requires high-bandwidth control of torque and speed. This platform subsequently has become an Application Specific Standard Product (ASSP) [7]. Due to its high processing capacity a FPGA platform was elected to implement ACS in a Multi-motor EV. In this paper, only the motor controller (bottom layer in Fig. 1) is described in detail, because it is the only layer that is in an advanced stage of development. The other control layers (traction and motion controller) are in an early phase and will be addressed in futures works.

II. Description of Control System

In Fig. 2 the propulsion system configuration is presented, which is based on a single FPGA chip capable of simultaneously controlling two induction motors. The energy applied to the motors is regulated by a set of DC/AC converters, supplied by four electrochemical batteries, and controlled by a Xilinx Spartan 3 FPGA (XC3S1000) [8]. The most important IP (Intellectual Property) Cores in the FPGA are the two Motor Controllers (MC), coded in Verilog and running in parallel, responsible for generating the PWM signals for the inverter in order to track the torque demanded by the driver. Additional modules for the interface with external peripherals, like the encoders and the Analog to Digital Converters, were also included. While the top level controllers, such as traction and yaw rate control (see Fig. 1), are not implemented, a uniform torque distribution strategy has been used, with both motor controllers receiving the same torque reference ($T_{left} = T_{right}$), defined by the throttle position. 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 engine 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 prototype, presented in the final section. Although simple, this "mechanical differential emulation" strategy has some weaknesses. For instance, it does not exploit all the merits offered by the multi-motor configuration, like handling and safety benefits that the yaw rate and side-slip control presents [3]. However, it remains a valid approach, which allows us to experimentally validate the motor controller layer, while the top level algorithms are not fully developed.

III. IMPLEMENTATION DETAILS

In Fig. 3 it is represented a block diagram that characterizes the interconnection of all modules implemented in the control chip and gives an idea of the system complexity. The main modules developed were:

1) Motor Controller - the classic Field Oriented Control (FOC) theory, using the indirect method for rotor flux orientation (λ_r), with the current PI controllers formulated on the synchronous frame (*dq* coordinates) and the inverter voltage modulation performed by a Space Vector PWM (SVPWM) algorithm [9], was used as the main motor control strategy. The FPGA chip includes 2 motor controllers, one for the left motor and the other for the right. Note that the parallelism offered by the FPGA allows additional motor controller to be included in the chip, without compromising the existing controllers already in place. The maximum numbers of motors that the FPGA can handle are only restricted by the area and multipliers available in the FPGA. In the current work a simple XC3S1000 chip [8] was enough for simultaneously control the 2 motors installed in the Electric Vehicle prototype.

2) ThrottleMap - this module implements a function that translates the throttle signal, defined by the driver, in a current reference (i_q current, proportional to the torque) applied to the motor controller. The ThrottleMap module is very useful to improve the driving experience, for instance, generating electric braking torque when the driver partially releases the throttle, making the driving more pleasant and predictable.

3) Electronic protections - the controller has a set of electronic protection against overcurrent, over and under-

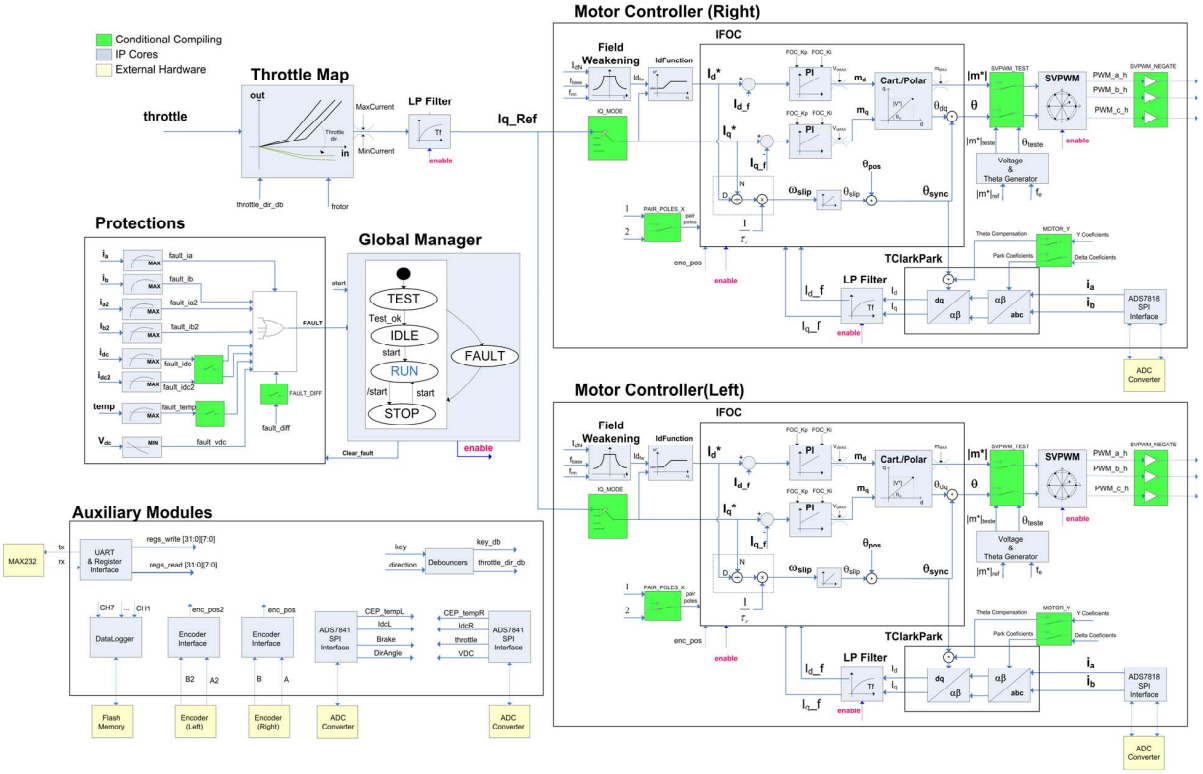


Figure 3 – Detailed view of the propulsion control chip, including the main IP Cores developed.

battery voltage and thermal overloads. Traditional DSP based controllers detects the overcurrent faults through an external analog comparison, activating an interruption in the DSP to stop the control process. In the FPGAs, the parallel processing capabilities make the implementation of protections quite effective: a dedicated protection module is continuously monitoring the current levels, comparing the digital current signal with the current preset limits, without disturbing the performance of other control modules. Actually, the bottleneck of the protection system is in the bandwidth of the current sensor and the ADC conversion, and not in the FPGA chip (the fault detection can be generated in a few tens of nanoseconds).

4) Global Manager - to manage the various modules of the system, the controller has a global state machine. At the start of system the module performs a series of initial validation tests (check current and voltage sensors, throttle signal, etc), and subsequently enables / disables the motor controller depending on the mode of operation selected and the faults detected.

5) Auxiliary Modules – Additional modules are used to make the interface with external peripherals: modules for counting the pulses sent by the encoder, SPI communications with the Analog to Digital converters, input debouncing, etc. To configure the parameters in the propulsion chip, a distributed structure of an 8-bit registers (32 registers for write and 32 register for read) are accessible by a RS232 link. The evolution of the variables in the propulsion chip is stored in Flash memory, accessible via SPI by a datalogger module developed for this purpose. The data stored during the system operation are useful to characterize the performance of the vehicle and the controller itself.

IV. "POWER IP CORE" DESIGN EXAMPLE

In this section the Space Vector PWM module, which is reused by the two AC Motor Controllers instantiated in the FPGA (see Fig. 3), is described in detail. The SVPWM modulator is one of the components that can benefit from the flexibility and parallel capabilities offered by the FPGAs. Compared with other platforms (such as DSPs), the implementation of the modulator in FPGA can facilitate the incorporation of advanced techniques, such as dead-time compensation [11] and overmodulation operation [12], and allow multiple-motor control with a single chip [13]. In this work, the last feature was explored to control a multi-motor electric vehicle.

A. Introduction

The SVPWM is one of the main techniques used to control three phase inverters, allowing a 15% increase in the linear zone of operation and a low current distortion, compared with carrier-based modulation technique, and was the modulation method used in this work. The module receives the normalized voltage vector reference, defined in polar coordinates (m, θ), and generates six PWM signals to be applied to the power semiconductors. The modulus of the normalized voltage vector is normally defined as modulation index, given by:

$$m = \frac{v_{1m}}{\frac{2}{\pi}V_{dc}} \quad (1)$$

where v_{1m} is the fundamental output voltage and V_{dc} is the DC Bus voltage. The three-phase inverter is capable of generating 8 voltage combinations (100, 110, 010, 011, 001, 101, 000 and 111 where 1 represents that the top switch is on and lower switch is off, and 0 mean the opposite) . These 8 vectors, 6 active and 2 null vectors, form a hexagon (see Fig. 4a) on the stationary reference frame and can be defined as:

$$m_n = \begin{cases} \frac{\pi}{2\sqrt{3}} e^{j\frac{\pi}{3}(n-1)}, & n = 1,\ldots,6 \\ 0, & n = 0,7 \end{cases} \quad (2)$$

The principle behind the SVPWM is based on the modulation of adjacent space vector for each sector. For instance, if the voltage vector reference lies in the first sector, during a switching period of time the inverter must apply the vector m_1 and m_2 during t_1 and t_2 and m_0 and m_7 during t_0 and t_7 times. A more detailed description of the SVPWM techniques can be found on reference [9].

The detailed view of SVPWM implementation is presented on Fig 4b. The SVPWM module starts by detecting the sector in which the voltage vector lies and then rotate it to the first sector. Subsequently, the times t_1, t_2 and t_0 are calculated based

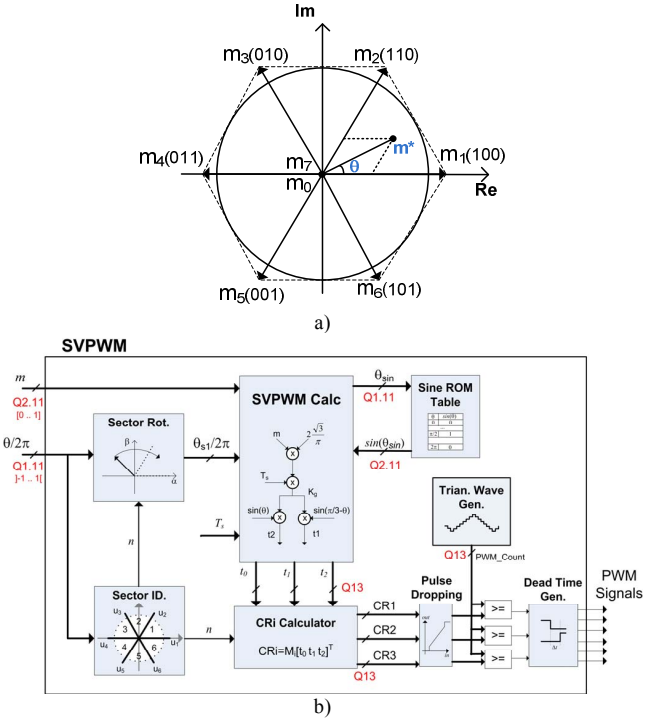


Figure 4 - a)Space Vector representation of three-phase converter; b) Space Vector PWM (SVPWM) implementation details (all variables have fix point (signed) format, represented by the notation Qx.y - x bits for the integer part and y bits for the fraction part).

on simple trigonometric relations, which are then used to determine the duty cycles values for each arm of the inverter. Finally, the duty cycles are adjusted by a pulse dropping function, compared with a triangular wave to generate the PWM signals, and corrected with the insertion of dead-times. In the next sections, the sub-modules of the SVPWM block are briefly described.

B. *Sector Identification and Vector Rotation*

To simplify the calculations, only the formulas valid in the first sector are implemented. The identification of the sector (1 to 6) in which the voltage vector lies is straightforward because the voltage angle (θ) is known and the sector can be easily detected, comparing the angle of the voltage vector with the 6 sector limits:

$$n = \begin{cases} 1 \quad , \ 0^o < \theta \le 60^o \\ 2 \quad , 60^o < \theta \le 120 \\ \quad\quad \ldots \\ 6 \quad , 300^o < \theta \le 360^o \end{cases} \tag{3}$$

Next, an equivalent first sector voltage vector (m_{1s}, θ_{1s}) is calculated, rotating the input vector to the first sector, which is performed subtracting the original voltage angle by *(n-1)*60* degrees, and maintaining the voltage modulus:

$$\begin{aligned} m_{1s} &= m \\ \theta_{1s} &= \theta - (n-1)60^o \end{aligned} \tag{4}$$

Note that, because the voltage vector is in polar coordinates, both the sector identification and voltage rotation are much simpler to perform than if the voltage vector was defined in the Cartesian coordinates.

C. *Times Calculation*

With the voltage vector in the first sector, the times t_1, t_2 and t_0, which defines the "on time" of the 2 active voltage vectors and the zero voltage, are calculated with the help of a simple trigonometric relationships, valid for the first sector [9]:

$$\begin{aligned} K_g &= m_{1s}\frac{2\sqrt{3}}{\pi}T_s \\ t_1 &= K_g sin(60^o - \theta_{1s}) \\ t_2 &= K_g \sin(\theta_{1s}) \\ t_0 &= t_7 = (T_s - t_1 - t_2)/2 \end{aligned} \tag{5}$$

where K_g is an auxiliary variable, T_s = 1/(2f_s) and f_s is the switch frequency (in practice, the T_s variable is replaced by a digital equivalent value, identified as PWM_TOP, and calculated as $f_{clk}/2f_s$, where f_{clk} is the FPGA clock value). The formulas presented in (5) need four multiplications: 2 for K_g and 2 for t_1 and t_2 calculations. Because the FPGA used in this work (XC3S1000) has a limited number of multipliers (24), compounded by the fact that multiple instances of SVPWM module are implemented to simultaneously control several

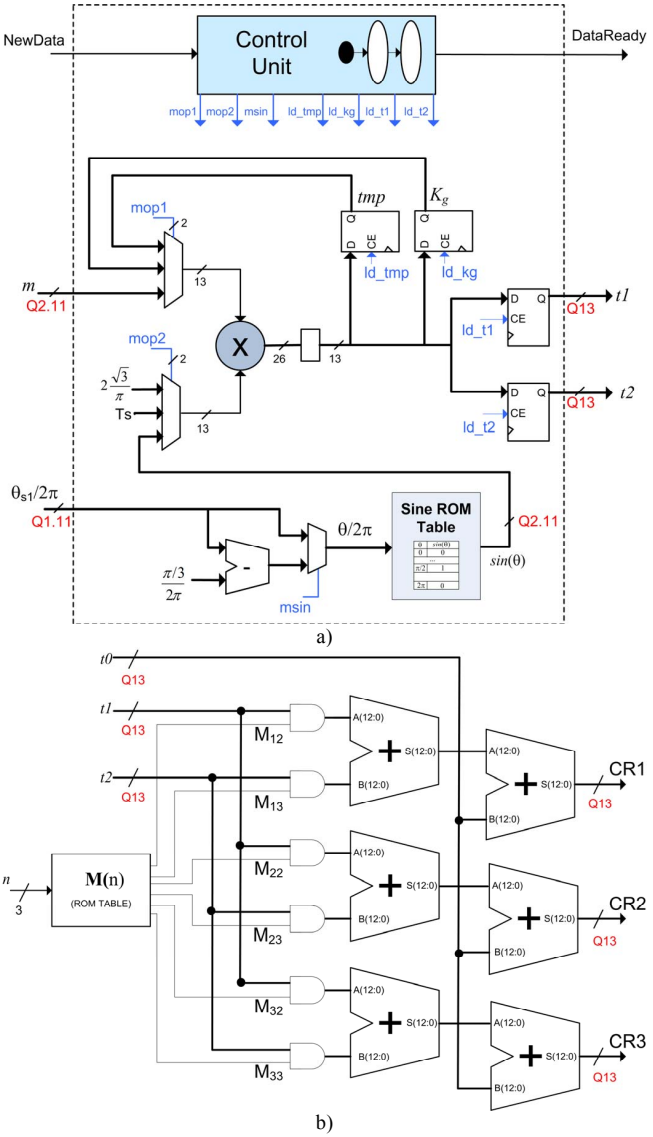


Figure 5 - Detailed view of: a) times calculation; b) duty cycles (CRi) calculator module. (all variables have fix point (signed) format, represented by the notation Q*x.y* - *x* bits for the integer part and *y* bits for the fraction part)

TABLE 1 MATRIX M DEFINITION

	Sector					
	1	**2**	**3**	**4**	**5**	**6**
M (n)	$\begin{bmatrix}1&0&0\\1&1&0\\1&1&1\end{bmatrix}$	$\begin{bmatrix}1&0&1\\1&0&0\\1&1&1\end{bmatrix}$	$\begin{bmatrix}1&1&1\\1&0&0\\1&1&0\end{bmatrix}$	$\begin{bmatrix}1&1&1\\1&0&1\\1&0&0\end{bmatrix}$	$\begin{bmatrix}1&1&0\\1&1&1\\1&0&0\end{bmatrix}$	$\begin{bmatrix}1&0&0\\1&1&1\\1&0&1\end{bmatrix}$

electric motors, a multiplier sharing strategy was developed to multiplex the use of this resource (Fig. 5a). The sharing strategy is based on a single multiplier, managed by a control unit, which is "time-shared" to execute the 4 multiplications needed by (5). For instance, to obtain the gain K_g, two steps are performed. First, the control unit selects the m and $2\sqrt{3}/\pi$ values as the multiplier inputs and stores the results in a temporary register (*tmp*). Second, the temporary register and the *Ts* values are used as the multiplier inputs, producing the K_g

value at the multiplier output, which is stored in an internal register to be used in next calculations (the times t_1 and t_2 are obtained in the same manner and described for the K_g variable). This sharing strategy enables us to reduce the use of dedicated multipliers from 4 to only 1, for each SVPWM instantiation. The sine function needed in (5) was implemented with a simple ROM (Read Only Memory) table, with the address bus specifying the sine angle, and the data bus returning the function result.

D. Duty Cycle Calculation and PWM generation

The t_1, t_2 and t_0 times must be transformed in duty cycles to be applied to each arm in the inverter. The duty cycles, which depends on the t_i times, but also on the voltage vector sector, are stored in the Compare Registers (CRi), compactly defined by:

$$\begin{bmatrix} CR_1 \\ CR_2 \\ CR_3 \end{bmatrix} = \mathbf{M}(n)\begin{bmatrix} t_0 & t_1 & t_2 \end{bmatrix}^T \quad (6)$$

Because the coefficients of matrix **M**(n) are restricted to zeros and ones (see Table 1), (6) can be efficiently implemented with 6 conditional sums (see Fig. 5b). The values of t_1 and t_2 are added, or not, depending on the value of the element M_{ij} (i row, j column of **M**). The conditional sum is controlled by introducing a AND function in the adder inputs: if the element M_{ij} is 0, ANDing t_1 (or t_2) with M_{ij} generates a zero value in the adder entry, disabling the summation. On the other hand, if M_{ij} is 1, the AND output is equal to t_1 (or t_2), enabling the summation. A small ROM table stores the various versions of the matrix **M** as a function of the voltage vector (the address bus specifies the sector and the data bus contains the elements of the matrix **M**). Note that the elements M_{i1} are always 1 (see Table 1), which means that t_0 is added to all Compare Registers, represented by the second group of adders in Fig. 5b.

Before generating the PWM signals, the CRi registers are shaped with a pulse elimination method [10], dropping pulses less than a minimum width (2 times the inverter deadtimes) to ensure the proper operation of the inverter when high modulation indexes are used. The final step in the SVPWM is comparing the CRi registers with a triangular wave, generated with an Up/Down counter, and apply the dead-times to the PWM signals.

E. Simulation Results

To minimize the number of resources used in the FPGA, it was decided to use variables with reduced resolution. All the calculations performed in the module use fixed point arithmetic with 13 bits resolution. The modulation index m is normalized between 0 and 1 and has a resolution of 13 bits, 2 for the integer part and 11 bits for the fraction part, i.e. Q2.11. The vector angle θ is normalized with 2π, in Q1.11 format. The registers [CR1 CR2 CR3] and [t_0 t_1 t_2] have 13 bits of resolution, interpreted as integers and without signal, and the sine ROM table has a $2\pi/2^{11}$ radians resolution in the angle, and generates the sine function with Q2.11 format.

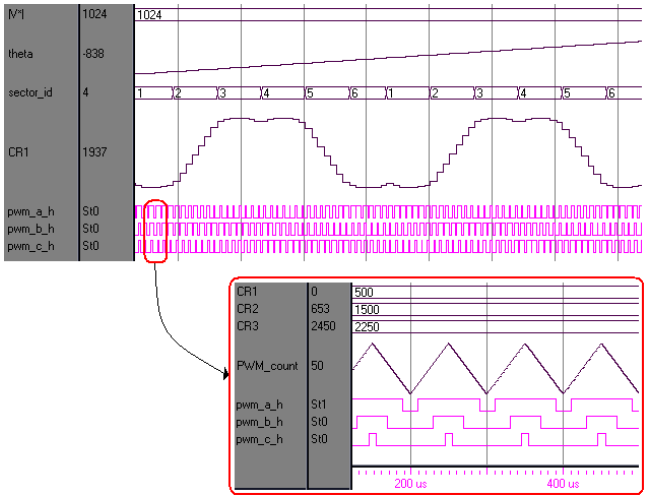


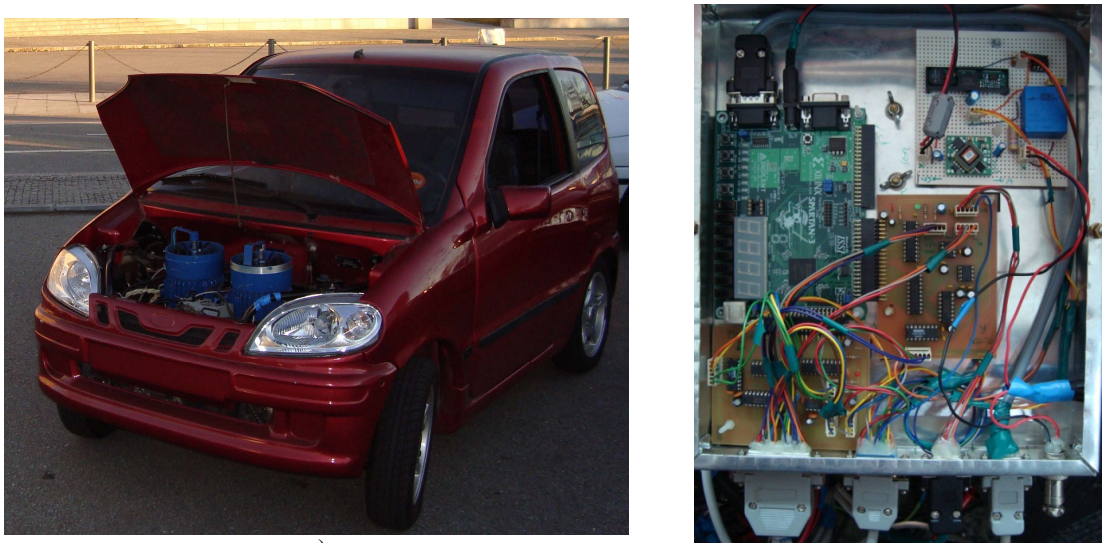
Figure 6. SVPWM simulation results, with fixed modulation index (m=0.5) and variable theta (bottom).

In Fig. 6 some simulations results for this module are presented. It can be seen the evolution of the CR1 register for a fixed value of the voltage vector module m (defined as $|V^*|$) and a variation from 0 to 4π in the vector angle θ (defined as *theta*). In the zoom box it is shown a fragment of the triangular wave, operating at 10kHz, and the output PWM signals, which demonstrates the correct operation of the developed SVPWM. These simulation results were obtained with the ModelSim program, which provides a useful environment to develop test-benches and conduct several tests to validate the correct function of the developed modules.

V. Experimental Results

A. Platform Description

The Faculty of Engineering of the University of Porto, in cooperation with some Portuguese firms, and under financial support of FCT (Foundation for Science and Technology), developed a multi-motor electric vehicle, named uCar (Fig. 7a). In the context of this research a conventional MicroCar, Virgo model, was transformed in a multi-motor electric vehicle, composed with two, low voltage, 2.2 kW three-phase cage induction motors (26V, Δ, 63A and 1410 rpm), coupled to the front wheels through fixed ratio (7:1) transmission. The energy source consists of 4 lead-acid batteries, connected in series (48V), with a stored energy of 5.28kWh. Without passengers, the vehicle weighs 600kg, with most of the mass concentrated at the rear of the vehicle (45% front, 55% to rear), motivated by 200kg of batteries installed in the trunk of the vehicle. To control the 2 DC/AC converters a single FPGA XC3S1000, based on the Digilent Spartan 3 Starter Kit, has been used (Fig. 7b). The FPGA board contains a set of useful peripherals (50MHz clock, expansion pins, Flash and RAM memory, etc.) and was expanded with 2 additional circuit boards. These boards contain ADCs peripherals (TIADS7818 and TIADS7848) to digitalize analog signals, such as currents, voltage, throttle signal, etc., which are essential for the control



a) b)

Figure 7 - Multi-Motor prototype: a) Chassis overview with, 2 AC Motors coupled to front wheel; b) FPGA Control System based on a simple XC3S1000.

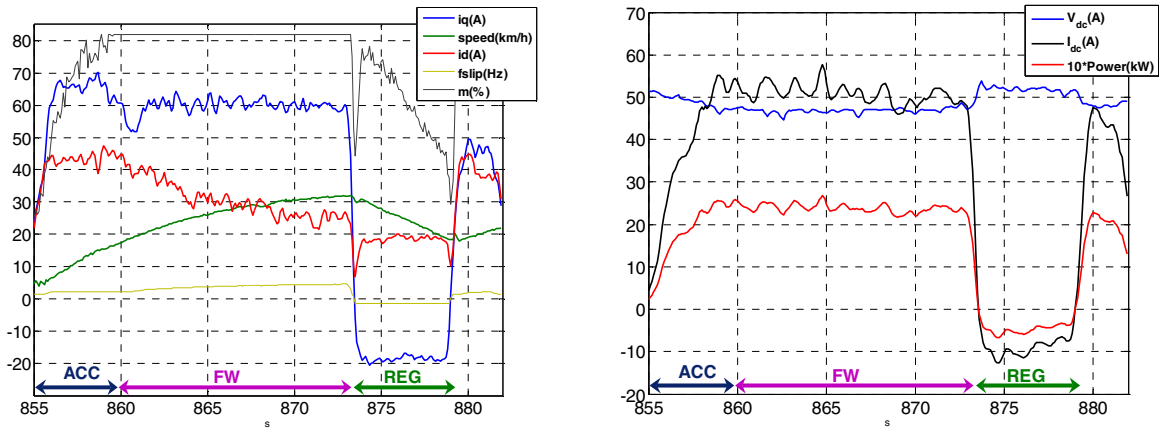


Figure 8. Experimental results during acceleration, field weakening and regenerative braking (left motor only);

algorithms, and logic converters for 3.3V/5.0V and 5.0V/3.3V conversion. Experimental results, demonstrating the basic operation of the motor controller and the uniform torque distribution strategy, were acquired with the FPGA embedded datalogger, which records the evolution of mechanical variables (motor speed), energy source status (voltage and current) and the FOC controller variables (i_q, i_d, motor slip, and modulation index m) throughout the tests.

B. *Acceleration/Braking Performance*

In Fig. 8 it can be seen the motor controller performance during a straight line test (shown only results for one motor, the other has similar results). During the initial acceleration the driver requests maximum torque and the currents i_q and i_d are maintained in their maximum values, producing an acceleration of 2.2km/h/s. When de EV reaches 18km/h the motor voltage saturates at 83% and i_d current ("flux" current) is reduced to allow the vehicle to operate in the constant power zone. During this period each motor consumes, approximately, 2.5kW. After reaching 30km/h the driver requests a reduction in the vehicle speed and a negative iq current ("torque" current) is applied to produce regenerative braking. During this period, each motor convert 500W from kinetic to electrical energy, emphasizing one of the most promising features in EV: energy recovering during braking.

C. *Cornering Performance*

The basic operation of the uniform torque distribution strategy, with both motors receiving the same torque reference (i_q current), is depicted in Fig. 9. During straight line maneuvers, which can be identified when the vehicle steer angle δ is close to zero, the wheels speeds are equal, but during the cornering maneuvers different wheel speeds emerge. A simple kinematic model (see (9) in Appendix) was used to predict the speed difference of the front wheels. Figure 9 shown that the kinematic model output $\Delta\omega_{model}$ is almost

overlapped with the experimental measure, Δω, confirming the correct operation of the proposed uniform torque distribution.

VI. Conclusions

In this paper an FPGA platform was used to control a multi-motor EV. The high processing capabilities and reduced processing times make the FPGAs an attractive solution for control several motors with a single chip. A simple XCS31000 was used to implement 2 induction motor controllers, base on Field Orientation Control and Space Vector PWM techniques. To validate the developed motors controllers a simple uniform torque distribution strategy was implemented. Experimental results obtained with a multi-motor EV prototype demonstrates the basic operation of the propose propulsion system during accelerating, braking, straight line and cornering manoeuvres. In future works the developed FPGA platform will be used to implement higher control layer, targeting improvements in vehicle safety and handling, with traction control systems and active torque distribution algorithms.

Appendix - Vehicle Kinematic Model

The vehicle behavior at low speed operation was modeled using a simple kinematic model, assuming zero slip angles at all wheels and a small vehicle side slip angle. In this situation, the vehicle yaw rate ($\dot{\psi}$) can be approximated by [14]:

$$\dot{\psi} \approx \frac{v}{L}\tan\delta_F \tag{7}$$

where v is the vehicle longitudinal speed, L is the vehicle wheelbase and δ_F is the front wheel steering angle. Using the additive superposition of the yaw rate and vehicle speed, the speed of the front left and right wheels (v_{FL} and v_{FR}) can be obtained through the "differential radii" method [15]:

$$\begin{cases} v_{FL} \approx v - \dot{\psi}\dfrac{C}{2} \\ v_{FR} \approx v + \dot{\psi}\dfrac{C}{2} \end{cases} \tag{8}$$

where C is the vehicle track width. Finally, combining (7) and (8), the speed difference of the front wheels can be obtained:

$$\Delta\omega_{model} = \frac{v_{FR}-v_{FL}}{r} = \frac{v}{r}\left(\frac{C}{L}\tan\delta_F\right) \tag{9}$$

where r is the radius of the wheel. Feeding the vehicle speed v and the steering angle δ_F in (9), a prediction for the speed difference of the front wheels can be obtained and compared with the experimental results. (Note: the electric vehicle uCar (Fig. 7a) has the following parameters: r = 0.26m; C =1.2m and L = 1.6m).

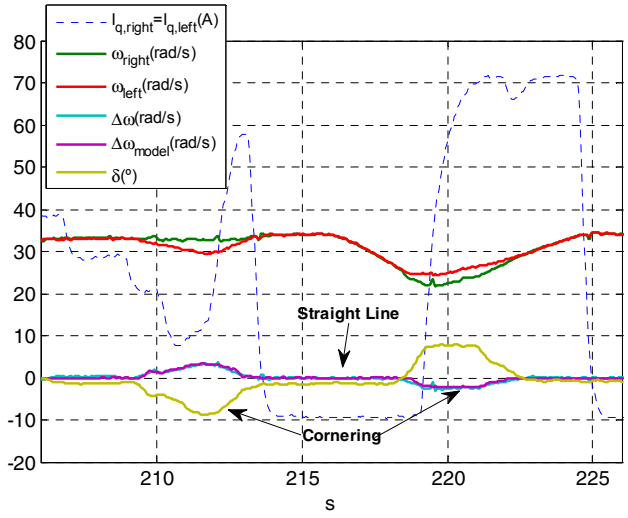


Figure 9. Experimental results during straight line straight line and cornering manoeuvres. (ω_{right} ω_{left} = angular speed of the right and left front wheels; $\Delta\omega = \omega_{right}-\omega_{left}$; δ = front wheel steering angle)

References

[1] Y. Furukawa and M. Abe, "Advanced Chassis Control Systems for Vehicle Handling and Active Safety", Vehicle System Dynamics, Vol. 28. No. 2, 1997

[2] Y. Hori, "Future Vehicle Driven by Electricity and Control—Research on Four-Wheel-Motored 'UOT Electric March II'", IEEE Transactions on Industrial Electronics, Vol. 51, No. 5, Oct, 2004

[3] A. Goodarzi and E. Ezmailzadeh, "Design of a VDC System for All-Wheel Independent Drive Vehicles", IEEE Transactions on Mechatronics, Vol. 12, No. 6, Dec. 2007

[4] L. Li, F. Wang and Q. Zhou, "Integrated longitudinal and lateral tire/road friction modelling and monitoring for vehicle motion control", IEEE Transactions on Intelligent Transportation Systems, Vol. 7, No. 1, 2006

[5] M. Naouar, E. Monmasson, A. Naassani, I. Slama-Belkhodja and N. Patin, "FPGA-Based Current Controllers for AC Machine Drives – A Review", IEEE Transactions on Industrial Electronics, Vol. 54, No.4, Aug. 2007

[6] "Accelerator Drive Design Platform - IRACS201 datasheet", International Rectifier, 2003

[7] T. Takahashi and J. Goetz, "Implementation of complete AC servo control in a low cost FPGA and subsequent ASSP conversion", Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition, 2004

[8] "Spartan 3 FPGA Family –Complete Datasheet", Xilinx, April, 2008

[9] M.P. Kazmierkowski, R. Krishnan and F. Blaabjerg, Control in Power Electronics – Selected Problems. Academic Press, 2002

[10] A.M. Hava, R.J. Kerkman and T.A.Lipo, "Carrier-based PWM-VSI overmodulation strategies: analysis, comparison and design", IEEE Transactions on Power Electronics, Vol. 13, No. 4, 1998

[11] S. Berto, S. Bolognani, M. Ceschia, A. Paccagnella and M. Zigliotto, "FPGA-based random PWM with real-time dead time compensation", IEEE 34th Annual Power Electronics Specialist Conference, 2003

[12] Z. Zhaoyong, L. Tiecai, T. Takahashi and E. Ho, "Design of a universal space vector PWM controller based on FPGA," Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition, 2004

[13] 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", The 25th Annual Conference of the IEEE Industrial Electronics Society, 1999.

[14] R. Rajamani, Vehicle Dynamics and Control. Springer, 2006

[15] U. Kiencke and L. Nielsen, Automotive Control Systems For Engine, Driveline, and Vehicle, 2nd Edition. Springer, 2005