EVS27
Barcelona, Spain, November 17-20, 2013

A new FPGA based PMSM torque controller for hybrid and electric vehicles powertrain

J. Ricardo Soares[1], Tiago Sá, Armando S Araújo, Adriano S. Carvalho
[1]J. Ricardo Soares (corresponding author), Tiago Sá FEUP, DEEC, Rua Dr. Roberto Frias, Porto, Portugal, jrsoares@fe.up.pt

Abstract

This paper presents a new approach to vector PMSM control implemented on a FPGA platform with a soft processor core integration for floating point computation. An advanced vector control is developed, according to PMSM vector control state-of-the-art, where the torque reference is a direct input to the system, dynamically controlling the injection of stator current flux component, in order to direct control of the torque angle. This approach provides a more robust solution for electric and hybrid electric vehicles as the controlled variable is the torque and the controller actuates simultaneously on the flux and torque current components and their angle. State-of-the-art presents two different control methods based on stator current flux component injection and the authors discuss another two methods, considering different relations between air-gap flux, rotor flux and stator current vectors. These methods are described and one is shown as appropriate to electric vehicle powertrain applications, regarding performance, efficiency and motor parameters. Concerning motor position information feedback, a PLL-based sensorless control technique is presented and compared with a solution based on position measurement. A hardware and software prototype based on a Xilinx® Spartan-6 FPGA and an electric vehicle powertrain set up are developed and tested for performance validation and experimental results are presented. The results show a robust controller with a high dynamics response.

Keywords: FPGA, PMSM, Powertrain, Torque Angle, Torque Control

1 Introduction

In the last few years, several papers have discussed electric and hybrid electrical vehicle architectures [1], [2]. Researches were carried out comparing the different types of electric machines for vehicular applications. In [3], [4] authors present a comparative study where the advantages and disadvantages of different kinds of electrical machines are highlighted in a vehicular application point of view.

This paper establishes the permanent magnet synchronous machine as the most appropriated machine for electric vehicle applications. The absence of brushes and copper losses in the rotor constitutes an advantage for high efficiency and reliability. There are two different types of PMSM depending on flux orientation: [5] presents a comparison between the two types and [6] discusses the advantages of axial flux motor type for electric vehicle applications. In this paper an axial flux PMSM is considered.

In PMSM, stator current produces flux and torque. Current flux component is in phase with rotor flux created by permanent magnets. On the other hand, current torque component is orthogonal to rotor flux linkage vector created by the PM [7].

Considering the dq reference frame, the flux and torque current components - i_d and i_q, respectively - have to be directly controlled considering the coupling effect between them.

Considering the development of vector control methods, information of the rotor position is required. In this paper a PLL-based sensorless technique is tested being the results compared with the ones obtained by measuring the rotor position, as some applications demand sensorless techniques.

The complexities of vector control have led to implementations based on Digital Signal Processors (DSP) as floating point computation is required. However, its sequential processing architecture provides non-deterministic latency which is a drawback for applications alike. FPGA based solutions allow parallel processing capabilities and deterministic latency, so some authors have already presented FPGA implementation of vector control approaches for driving the electric motor as [8] and [9], with fixed point algorithm computation.

A fixed point algorithm computation for vector control can be of very complex development. Having this in consideration, the authors have designed a FPGA based system that integrates the design of soft processor core units that run floating point operations in order to simplify the implementation of the vector control algorithm. So, the proposed solution gathers the advantages of the FPGA to the convenience of the float point support of a DSP.

2 PMSM Modelling

The permanent magnet synchronous motor is a particular type of synchronous motor, which runs in synchronism with the source. When three-phase currents flow in the windings, magnetic poles are created on the surface of the stator. These created pole pairs move along the stator surface, producing a rotating magnetic field as the phase currents alternate in time with a displacement of 120º between them [10].

The PMSM can be mathematically modelled by their dq voltage equations as follows [11]:

$$\begin{cases} V_d = R_S i_d + \partial\lambda_d - \omega_e \lambda_q \\ V_q = R_S i_q + \partial\lambda_q + \omega_e \lambda_d \end{cases} \tag{1}$$

$$\begin{cases} \lambda_d = L_d i_d + \Lambda_m \\ \lambda_q = L_q i_q \end{cases} \tag{2}$$

Where V_{dq} is the voltage vector in dq reference frame, R_s is the phase resistance, i_{dq} is the current vector in dq frame, ∂ is the differential operator, λ_{dq} is the air-gap flux vector in dq frame, ω_e is the rotor electrical angular speed, L_{dq} is the inductance in dq frame and Λ_m is the rotor flux.

The electromagnetic torque is, by definition, proportional to the cross product between the air-gap flux and stator current vectors. Hence, it can be written as follows [11]:

$$T_e = \frac{3}{2} p(\vec{\lambda} \times \vec{I}) \tag{3}$$

Where p is the number of pole pairs.

The electromagnetic torque can also be calculated from the current and flux linkages in the rotating reference frame, dq. Thus, it can be written as presented in [12] and transcribed here for convenience:

$$T_e = \frac{3}{2} p\Lambda_m i_q + \frac{3}{2} p\left(L_d - L_q\right) i_d i_q \tag{4}$$

The first component is the excitation torque, which exists due to the influence of the rotor flux linkage and constitutes the main component of the produced torque. The second component is called reluctance torque. It exists due to the effect of the salient poles and so it naturally depends on the saliency ratio L_d/L_q [13]. This component may be zero if $L_d = L_q$ which is the case of surface mounted PM rotors.

Concerning torque angle, it is defined as the angle between stator current vector and air-gap flux vector, as presented in the electromagnetic torque definition in equation (3). Therefore, torque can be expressed as a function of torque angle, θ_T, as:

$$T_e = \frac{3}{2} p\left|I_S\right|\left|\lambda_S\right| \sin(\theta_T) \tag{5}$$

3 System Controller Design

3.1 Torque Control Loop

In this paper is proposed a control design solution that aims to directly control the produced torque having a torque reference as the input of the system. In this situation neither speed reference nor speed controller are used. The control is based on processing the torque error generated from the difference between produced and reference torque given by the driver through the throttle input. Without the presence of the speed controller the

driver is the responsible of closing the speed loop acting as a speed controller by himself. This approach solves the problem of loss of synchronism in case of dynamic load variation, which could cause serious problems. In fact, with this approach, load variations are reflected in speed variation but never affect the produced torque.
To illustrate how the torque controller designed works, it is represented in Fig. 1 a block diagram of the developed control algorithm.

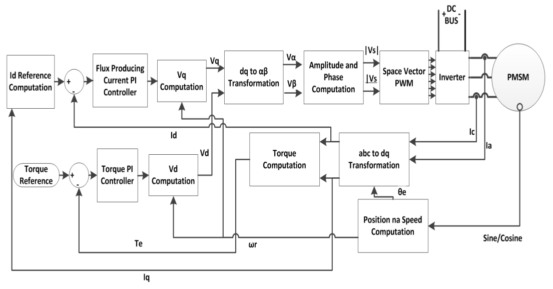

Figure 1: Block diagram of the developed control algorithm.

The control loop is based on two PI controllers, running in parallel, that compute the error of torque and flux producing current components in order to calculate values for dq axis voltages according to equation (1). Those are transformed in αβ reference frame quantities and then computed by the space vector modulation algorithm that generates the gate signals for the IGBTs.
The *Id Reference Computation* block includes a control method that aims to optimize motor performance. These methods are discussed in the next section.
The feedback quantities are the phase currents and rotor position. The latter can be either measured or estimated and the performance of both are presented in this paper. Only two of the three phase currents are needed to be measured as the third one can be obtained from the other two.

3.2 Id Current Injection Control Methods

The controller presented is constituted by two main loops, each one including one PI controller. The torque loop is responsible for calculating the torque error and generating the corresponding i_q; the flux loop is responsible for applying one of the following control methods and generating i_d.

- Stator current vector (I_s) at 90º;
- Maximum Torque Per Ampere (MTPA);
- Air-gap flux vector (λ_s) at 90º;
- Torque angle (θ_T) at 90º;

The first method is adapted from [12] and is based on controlling the angle between stator current and rotor flux vectors, θ_i. Keeping this angle equal to 90º, the reluctance torque component is nulled and thus the torque is proportional to the stator current amplitude. This method has a really smooth start-up since the stator current amplitude is equal to i_q which increases proportionally to torque.
The second method, based in [11], pretends to maximize the developed torque with the least current usage. The implementation is based on finding the zeros of the differentiation of the torque expression in order to the torque/current ratio as follows:

$$\frac{d}{dt}\left(\frac{Te}{I_S}\right) = \frac{3}{2}p(\Lambda_m \cos(\theta_i) + (L_d - L_q)I_S \cos(2\theta_i)) = 0 \qquad (6)$$

Hence, it is possible to find the expression that defines i_d magnitude in order to minimize stator current. The result has two solutions, being chosen the one that gives the lowest value:

$$i_d = -\frac{\Lambda_m}{2(L_d - L_q)} - \sqrt{\frac{\Lambda_m^2}{4(L_d - L_q)^2} + i_q^2} \qquad (7)$$

This control is the best regarding the amount of current needed for the developed torque. The start-up is really smooth because the air-gap flux vector phase is slowly increased due to generated i_d reference, which reduces the instantaneous current per torque unit. This fact makes this control method very suitable for this kind of application when PMSM saliency ratio is different of the unity.
The third method is one of the suggested control approaches by the authors of this paper. It pretends to control the air-gap flux angle, θ_λ, based on the description of [11]. Keeping the d current component equal to $-\psi_m/L_d$ it is possible to keep the air-gap flux angle at 90º resulting in the maximum value of the torque in these conditions. Under the same conditions of the first method, start-up current is much higher, which is a consequence of the constant reference of i_d that is needed to null the effect of the rotor flux linkages right from the beginning. On the other hand, it results to need less current amplitude in steady state in the same conditions for motor with a low ψ_m/L_d ratio.
The last control method presented is another suggested control approach by the authors, which pretends to directly control the torque angle, θ_T. It aims to keep this angle at 90º in order to control the torque proportionally to the product of air-gap flux and stator current amplitudes according to equation (5).

To keep this angle at 90º the following equation is solved in order to obtain a value of i_d to be injected.

$$\tan^{-1}\left(\frac{i_q}{i_d}\right) - \tan^{-1}\left(\frac{i_q L_q}{i_d L_d + \Lambda_m}\right) = \frac{\pi}{2} \tag{8}$$

The solution is presented in equation (9).

$$i_d = \frac{\sqrt{{\Lambda_m}^2 - 4L_q L_d {i_q}^2}}{2L_d} - \frac{\Lambda_m}{2L_d} \tag{9}$$

However, this method has a mathematical problem. The i_d expression solution is complex for high values of i_q, which makes this approach only valid for limited values of produced torque and thus not a very suitable control considering all the torque range. As this is a method with i_d current injection, it allows the production of maximum torque for higher speed range, since it weakens the PM magnetic field.

Each proposed control method has different performance behavior that may have advantages or disadvantages depending on the constructive characteristic of the PMSM and its application. For this paper, since a vehicular application is presented, the method chosen is MTPA.

3.3 Position and Speed Feedback

3.3.1 Position measurement

Regarding of position acquisition, a resolver sensor is considered. A resolver is a device that produces a sine and cosine wave analog output that corresponds to the sine and cosine wave of the rotor angle relative to the stator.

Note that the motor and resolver poles number have to be considered. If the resolver does not have the same number of poles of the motor then a multiplicative factor has to be taken into account.

Dividing the resolver sine signal by the cosine signal, the tangent of the resolver position is obtained. Therefore, the absolute rotor electrical position may be obtained as follows:

$$\frac{\sin(\theta)}{\cos(\theta)} = \tan(\theta) \Rightarrow \theta_e = \tan^{-1}\left(\frac{\sin(\theta)}{\cos(\theta)}\right) k_{poles} \tag{10}$$

Where k_{poles} is the multiplicative factor that depends on the relation of motor and rotor poles. k_{poles} is then given by the ratio between the number of the motor poles and resolver poles as follows:

$$k_{poles} = \frac{n_{mp}}{n_{rp}} \tag{11}$$

Where n_{mp} is the number of motor poles and n_{rp} is the number of resolver poles.

From the electrical angular position, θ_e, is then possible to compute electrical angular speed, ω_e. It is calculated by computing the discrete derivative of the electrical position as follows:

$$\varpi_e = (\theta_e - \theta_{e\_previous})\frac{1}{\Delta t} \tag{12}$$

Where Δt is the derivative time constant and $\theta_{e\_previous}$ is the last value of the rotor position stored Δt seconds before.

Position information calculated from resolver signal acquisition is computationally less complex than when using sensorless methods and usually more reliable at low speeds. However, it requires more hardware as in the sensor by itself and all the electronics for signal conditioning and acquisition, which can increase the overall cost of the system.

3.3.2 Position estimation

Many sensorless techniques have been developed along the years. They can be separated on the following categories:

- Back-EMF estimators [14];
- State observers [15] and [16];
- Sliding-mode observers [17] and [18];
- High-frequency signal injection [19];
- PLL-based estimators [20] and [21];

Regarding the complexity of each one, the mathematical effort and the results obtained, PLL-based estimator is chosen for this application.

This algorithm is based on [21] and pretends to estimate rotor position by synchronizing the estimated reference frame with the real rotor reference frame, on which the rotor flux vector is aligned with *d-axis*. The vectors and referential frames at issue are represented in Fig. 2 where the superscript '^' refers to estimated. It is clear that when $\hat{e}_d$ amplitude is zero the rotor back-EMF is along *q-axis*, overlapping the two referential frames.

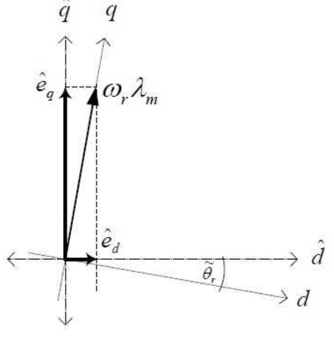

Figure 2: Estimated and real *dq* reference frames [21].

The block diagram presented in Fig. 3 shows the principle of operation of the PLL estimator implemented.

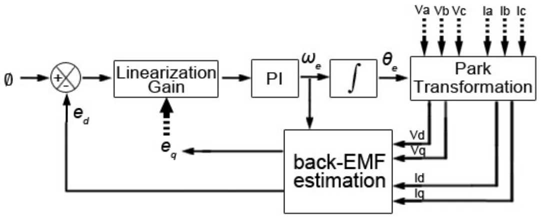

Figure 3: Block diagram of PLL-based sensorless algorithm.

Back-EMF *d* component, $\hat{e}_d$, is compared with a zero reference as explained before. Then, the error enters on a PI controller, which output is the estimated speed. The integral of that speed is calculated in order to obtain the estimated position. This data is used to apply the Park Transformation to the three-phase voltages and currents measured. Their *dq* components are then used to calculate *dq* back-EMF components as in (13) and (14), (where the derivative components are neglected admitting a fast processing controller on which dt is very small and so are current variations in that time) from which $\hat{e}_d$ enters the loop, as explained. This means that when back-EMF *d* component is zero, the position used in Park Transformation is accurate and so it is the speed.

$$\begin{aligned} \hat{v}_d &= R_S \hat{i}_d + L_S \frac{d}{dt}\hat{i}_d - \hat{\omega}_e L_S \hat{i}_q - \hat{e}_d \\ \Leftrightarrow \hat{e}_d &= -(\hat{v}_d - R_S \hat{i}_d + \hat{\omega}_e L_S \hat{i}_q) \end{aligned} \tag{13}$$

$$\begin{aligned} \hat{v}_q &= R_S \hat{i}_q + L_S \frac{d}{dt}\hat{i}_q + \hat{\omega}_e L_S \hat{i}_d + \hat{e}_q \\ \Leftrightarrow \hat{e}_q &= \hat{v}_q - R_S \hat{i}_q - \hat{\omega}_e L_S \hat{i}_d \end{aligned} \tag{14}$$

With this technique, the PMSM can be started from zero speed without any changes or additional considerations, except the non-zero *q* back-EMF component that is settled when there is no speed and no torque reference. This is mandatory because of the Linearization Gain, which cancels the speed-dependent gain nature of the PLL-loop, linearizing the position controller dynamics throughout the entire range [21]. Since it is defined as (15), assuming the correct functioning of the controller, $\hat{e}_d$ is settled to zero thus simplifying this gain as in (16).

$$G_{Lin} = \frac{-1}{\sqrt{\hat{e}_d^2 + \hat{e}_q^2}} \tag{15}$$

$$G_{Lin} = \frac{-1}{\hat{e}_q} \tag{16}$$

3.4 Space Vector PWM

The SVPWM technique proposed in this paper is based on the one presented in [22] but an adaptation is developed considering the computational simulation software used and mainly the hardware processing target, in order to be easily implemented. To produce the necessary voltage vector by means of inverter state vectors, information about the times for the application of each boundary vector is needed. [22] presents the following equations to calculate the duration times of each boundary vector:

$$t_k = \frac{3}{2}\frac{T_m}{V_{dc}}\left|V_{ref}\right|\left[\cos(a) - \frac{\sin(a)}{\sqrt{3}}\right] \tag{17}$$

$$t_{k+1} = 3\frac{T_m}{V_{dc}}\left|V_{ref}\right|\left[\frac{\sin(a)}{\sqrt{3}}\right] \tag{18}$$

$$t_0 = T_m - t_{k+1} - t_k \tag{19}$$

Where T_m is the switching frequency, t_k is the lower vector time, t_{k+1} is the upper vector time and t_0 is the zero voltage time.

The next stage of the algorithm is to generate a switching sequence with the expected duty cycles for each vector in a center-weighted PWM sequence over the period T_m. It is on this stage that is important to develop a solution possible and simple to be implemented in the target control platform (FPGA). To achieve this, it is created a solution based on a triangular wave with period T_m, amplitude $T_m/2$ and duty cycle of 0.5. The objective is to compare the time values with the amplitude of the triangular wave in order to generate a pulse with the width necessary to implement the expected sequence.

Note that this technique only requires 3 different vectors in 5 transitions because only one of the zero vectors is used. This way the number of commutations is reduced and consequently there are less switching losses. The only vector with zero value used is the vector V_0, which corresponds to the switching combination where the 3 top transistors are off.

The final sequence will have different intermediate times which depend on whether the sector is odd or even and thus the sequence times will be called hereafter: T_{center}, for the vector located in the center of the sequence and T_{wrap}, for the vector around it. These times are obtained from t_k and t_{k+1} as equations (20) and (21).

$$T_{center} = \begin{cases} T_m - \dfrac{t_{k+1}}{2} & for \quad odd \quad sector \\ T_m - \dfrac{t_k}{2} & for \quad odd \quad sector \end{cases} \qquad (20)$$

$$T_{wrap} = \begin{cases} T_{center} - \dfrac{t_k}{2} & for \quad odd \quad sector \\ T_{center} - \dfrac{t_{k+1}}{2} & for \quad odd \quad sector \end{cases} \qquad (21)$$

The difference between even and odd vectors is just a swap in which boundary vector is generated first. This fact is just to keep the center waited pulse sequence.

The sequence is generated comparing the reference time values with the triangular wave, resulting in true when the wave is higher than the reference values and false when it is not. To better understand the process described above it is represented in Fig. 4, an illustration of all the steps of the technique that generates the pulses.

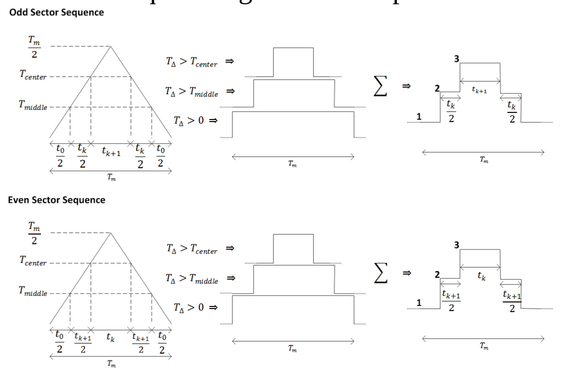

Figure 4: Sequence timing generation stages.

The result from the comparisons when summed result in a signal with 3 step values. Each step has the expected width for the vector. The three step values 1, 2 and 3 correspond to the vectors V_0, V_k and V_{k+1} respectively. This technique facilitates the process because this signal together with the number of the sector are then used as inputs of a lookup table that does the selection of the respective switching vectors and converts it in the gating signal with pulses with the width equal to the width of the correspondent step.

Each switch has to have a lookup table with the correspondent states of the respective switch in order to turn it on or off as function of the sector number and the boundary vector selector signal. The lookup tables of the bottom switches are the inverse of the respective top switch.

The space vector modulation technique is very suited for three phase electric machine control because the space vectors describe the machine in both steady and transient state operation. Also, this technique generates low THD, the unexpected low order harmonics are reduced and the voltage gain are increased in comparison to sinusoidal PWM.

4 Computational Simulation

With help of PSIM® software it is possible to model the system from power stage until the software controller and analyze the behavior of the developed controller for different conditions.

Using PSIM® tools it is possible to model the PERM 156M motor, which is a 22kW motor with a nominal current of 255A$_{rms}$ and a nominal torque of 35 Nm. More information about this motor is presented in Table 1. PERM motor has a $L_d = L_q$, thus applying the MTPA control method results to be the same of injecting zero i_d current.

The system was modeled as close as possible to the implemented prototype, except for the time interval (that is kept small to decrease the processing time required for the simulation), in order to analyze the dynamic response of the controller close to the real conditions.

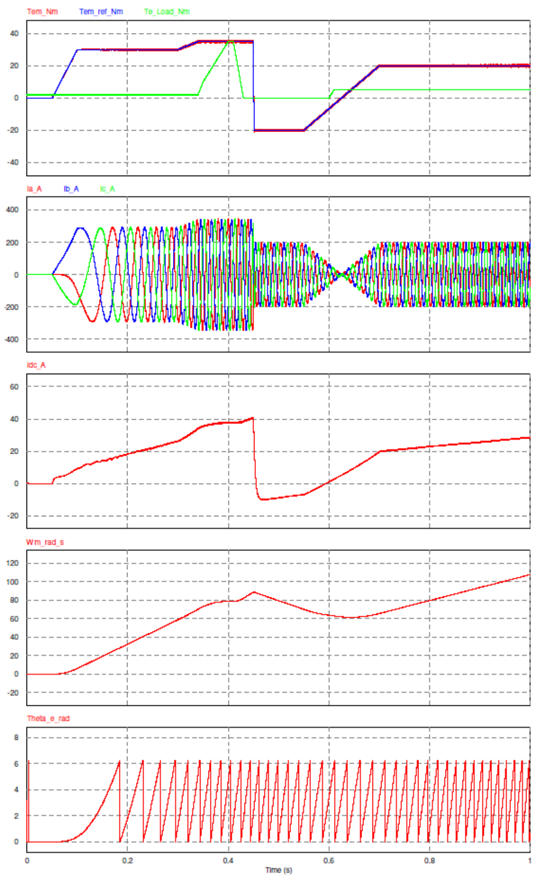

Figure 5: PSIM® simulation results with resolver.

A variable torque reference is given to the controller to observe the response of the torque controller for different conditions. It is presented, in this order, the results for torque in *Nm*, phase currents in *A*, mechanical speed in *rad/s*, position of the electric system in *rad* and DC bus current in *A* during 1 *s* of simulation, for a variable mechanical load with a moment of inertia of 0.1 Kgm^2. Fig. 5 presents the results based on position feedback through position measurement and Fig. 6 presents the results based on position estimation.

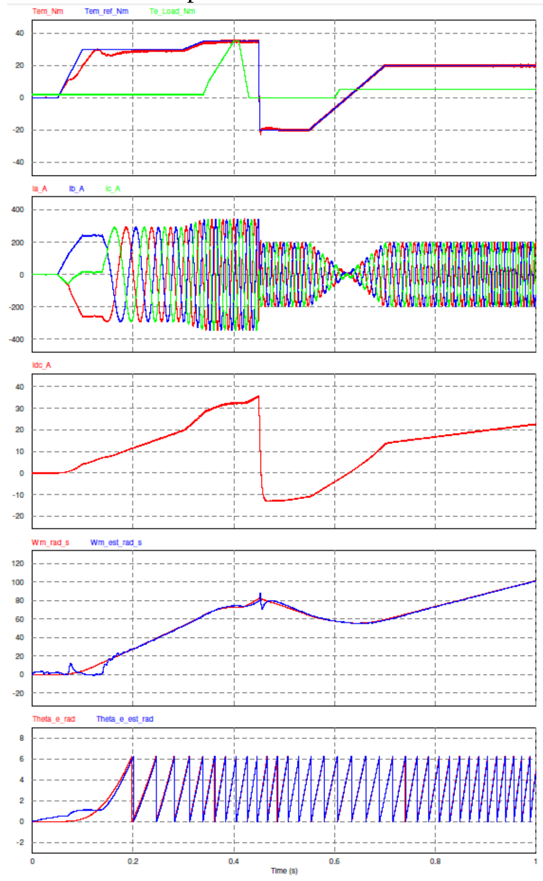

Figure 6: PSIM® simulation results with position estimation.

Attending to Fig. 5, it is possible to see that the controller is very dynamic, the generated torque follows the torque reference almost instantly. Note that between 0.43s and 0.62s an order is given for regenerative braking and hence the speed slows down. The negative DC current indicates that the motor is working as a generator, recovering energy to the batteries.

Comparing both results it is possible to conclude that both methods have good dynamic response. However, in the case of position estimation, the produced torque have some perturbations due to estimation errors at low speeds.

5 Implementation

In this paper, an Avnet® Spartan-6 LX75T Development kit is used as main core for the controller platform. This board is designed and assembled by Avnet® and is based on a Xilinx® Spartan-6 LX75T FGG676A. It includes on board JTAG communication, RS-232 over USB, power supplies and SPI flash memory integrated with the Spartan-6 FPGA core running at *100 MHz*.

In this section, all the hardware development and integration for the electric power train controller is described, as well as the digital design of the developed controller in the Xilinx® FPGA through the Xilinx® EDK and SDK software development tools.

5.1 Hardware architecture

The hardware architecture can be divided in two parts: the *Power Stage* and the *Control Stage*. The Power Stage is constituted by a 100V nickel-metal hydride battery pack connected to a three-phase IGBT inverter module by Infineon®. To the DC input of the inverter is connected a 20µF MKP capacitor to filter voltage transients introduced by cable parasitic inductances. The inverter feeds a PERM 156M motor which is the last element of the powertrain. Table 1 presents the summary of relevant characteristics of the motor and inverter.

Table1: Inverter and motor characteristics

Infineon® FS400R07A1E3 Inverter	Rated Current	400A
	Rated Voltage	600V
PERM 156M Motor	R(u-v)	5 $m\Omega$
	L(u-v)	47.333 μH
	Pole pairs (p)	4
	Rated Power	22 *kW*
	Rated Voltage	65.6 *Vrms*
	Rated Current	255 *Arms*
	Rated Torque	35 *Nm*
	Rated Speed	6000 *rpm*
	Inertia	58.6 $Kg.cm^2$
	Weight	29.6 *Kg*

The *Control Stage* is responsible to acquire relevant system variables, compute the torque control loop and generate the gate signals for the IGBTs.

The current acquisition is done by open loop linear output *Hall Effect* transducers which provide galvanic isolation from the *Power Stage* and produce a voltage signal proportional to the current being measured. The output signal from the current sensor of phase *a* and phase *c* is then converted by two 12 bit analog-to-digital converters that provide the digital value of the conversion to the FPGA.

The position feedback is provided by the resolver sine and cosine analog outputs that are converted by two 12 bit ADCs as well, or computed by PLL-based estimation technique described before.
The throttle and brake sensors are based on resistive variation sensors that produce a voltage signal proportional to the position of the throttle and brake. The PERM motor and the Infineon® inverter have thermistor sensors that allow the acquisition of a voltage signal proportional to temperature. For the 4 resistive based sensors a conditioning circuit is developed where the signals are buffered and then converted by 12 bit analog-to-digital converters.
The switching signals produced by the FPGA are conditioned and drive by Concept® 2SC0108T dual channel drivers together with the 2BB0108T basic board. These devices are responsible for interfacing the PWM signals with the gates of the IGBTs, providing galvanic isolation from the power stage, high driving current up to 8A and short circuit monitoring.
A discrete logic circuit is also implemented for protection purposes. The fault signals from the drivers are monitored as well as the DC bus current, which is also measured with a *Hall Effect* current sensor and compared with safe limit references in order to shut down the system in case of fault.
The power supply for the controller circuitry is obtained from a DC/DC converter fed by the batteries that provide 15V. The 15V supplies the protection and driver boards. An isolated converter transforms the 15V in 12V to supply the FPGA and digital circuitry and a second isolated converter feds the analog circuitry.
To better understand the developed hardware architecture, it is presented in Fig. 7 a diagram that presents the main parts of the prototype.

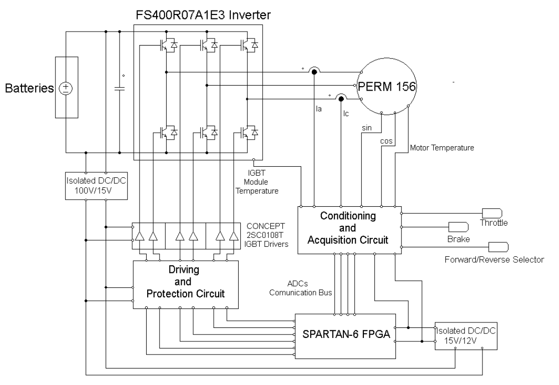

Figure 7: Hardware diagram of the control prototype developed.

5.2 FPGA Architecture

For FPGA design, a hardware description language is needed. In the development of this controller, VHDL language is used.
The first step in the design is the creation of VHDL IP (Intellectual Property) cores which are independent hardware processing units that allow to distribute the processing tasks among several of these units (these IP only support fixed point operations), achieving parallel processing. In this point the advantages of FPGAs start to highlight since each of the ADC IP core will run in parallel sampling the current, temperature and position at the same time.
In FPGA IP design, floating point operations cannot be used due to its technology. As is mentioned above, vector control and position/speed computation require complex mathematical efforts, which can be simpler to implement if floating point operations are available. To overcome this drawback, it is proposed in this paper a distributed processing architecture that aims to integrate the parallel processing of several VHDL IP cores with soft processor cores that allow using floating point math. Xilinx® EDK development tools allow the design of MicroBlaze® soft processor cores that can afterwards be programed in C language having the same behaviour of a physical microprocessor.
The architecture developed integrate 4 IP cores for ADC interface, for current, motor and inverter temperature, throttle and brake, and position acquisition. These IPs are prepared to communicate with the analog-to-digital converters through its proprietary communication protocol and write the conversion value into memory registers that are available for the higher layer of MicroBlaze® processors.
Another important IP core developed is the SVPWM IP. It will compute the gate signals for the inverter according to the information that is given by the control loop computations of the MicroBlaze® cores, with a fixed switching frequency of 10 *kHz*.

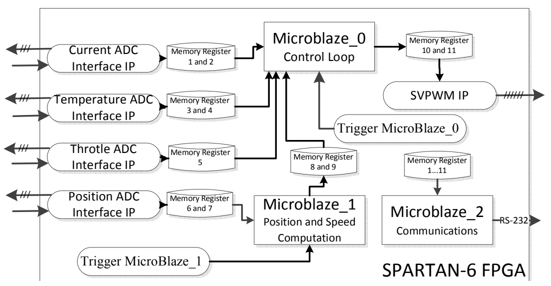

Figure 8: Diagram FPGA developed architecture

In order to better understand the developed architecture, it is presented in Fig. 8, a schematic of the developed FPGA architecture.

To ensure a periodic execution of the algorithm running in the MicroBlaze® processors, 2 trigger IP cores are developed: *Trigger MicroBlaze_0* and *Trigger MicroBlaze_1*. These IP cores are developed to generate a periodic signal of 100 *kHz* frequency that triggers an interrupt in the MicroBlaze® cores that allows a periodic execution of the code. This provides a deterministic latency for code execution of the Microblaze® processors since it is ensured that the execution of the code does not take more time than 10µs. Thus, with this strategy a sampling frequency of 100 *kHz* is fixed, and time dependent variables and tasks are correctly computed.

The architecture integrates a 3-core processing, where one MicroBlaze® core computes position and speed either from resolver position feedback or sensorless algorithm. The second computes the control loop and the third one is responsible for manage communications over RS-232 for debugging and variable logging. The MicroBlaze® cores are configured to allow floating point operation in a 32 bit format and to allow integer division and multiplication. All of the IP and Microblaze® cores are connected to an axi4lite communication bus to interface data with memories, ADC and PWM IP cores and FPGA primary clock.

6 Experimental Results

After prototype assembly, laboratorial tests are performed in order to validate the traction control system. The electric motor is mechanically connected to the transmission of a four wheeled vehicle though a gear chain with a fixed ratio of 2:1. The final relation of the vehicle transmission is 3.69, thus the total fixed ratio from the motor axis to the wheel axis is 7.38:1.

The tests are conducted under load conditions, the motor produces a rated torque of 35 Nm so for the test the applied load is 35% of the motor rated load, approximately 12 Nm, in the beginning, at the 5th second the load is reduced to 10 Nm and at the 10th second the load is again reduced to 8 Nm. Considering the gear ratio, to apply 12 Nm, 10 Nm and 8 Nm to the motor axis, it has to be applied ≈89 Nm, ≈74 Nm and ≈60 Nm to the wheel axis respectively. Note that this load have a high static friction allowing a good analysis of the controller and motor response from standstill.

The results of the tests are obtained from a log file generated by the controller. The Microblaze core dedicated to the communications sends the value of the produced torque, torque reference and mechanical speed every 50 ms thought RS-232 protocol to the computer where the information is stored. The data obtained from the log file is plotted in Fig. 9.

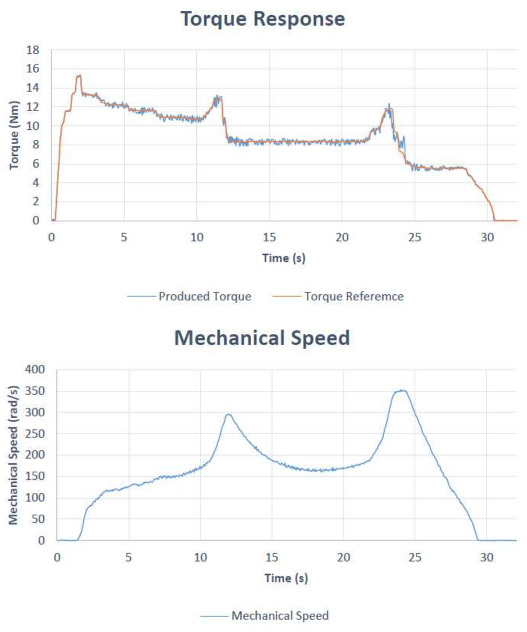

Figure 9: (Top) Torque reference profile vs torque response and (bottom) mechanical speed, of the developed prototype.

During the tests, the current of phase *a* is measured, Fig. 10 presents the phase current waveform in steady state.

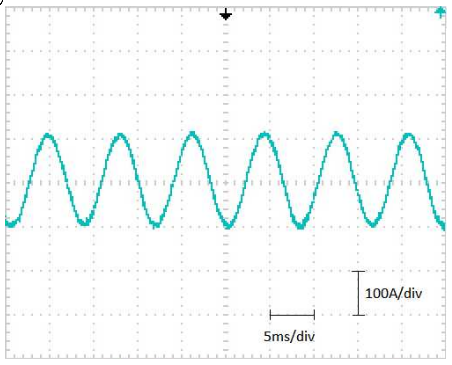

Figure 10: Phase current waveform.

As it can be seen, in steady state the measured phase current has a sinusoidal shape as it is expected.

As matter of torque dynamic response it can be seen that the produced torque follows the torque reference profile introduced through throttle sensor measurement, simulating a real reference torque profile introduced by a vehicle driver.

From the FPGA performance point of view, the consumed resources and primary clock delay are analised.

The processing time of the control cycle managed by *MicroBlaze_0* core is analysed and Fig. 11 presents the maximum execution time of the control cycle through an output signal that has a logic high value during the execution of the control algorithm.

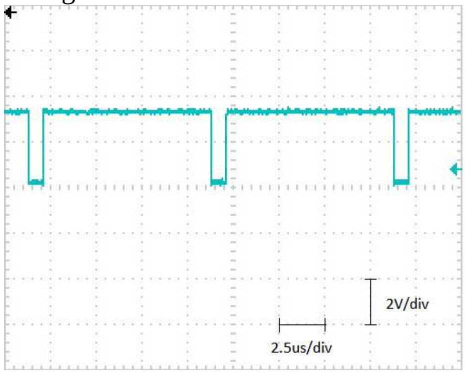

Figure 11: Maximum execution time of the control loop.

As it can be seen the maximum execution time is around 1us less than the period of the interrupt signal, which guaranties a fixed frequency for the control algorithm and hence a fixed and known latency for time dependent calculations.

Table 2 presents the resources consumed and the maximum delay time of the FPGA primary clock.

Table2: Spartan-6 FPGA performance

clk 100MHz	Max delay	1.866 *ns*
Resources Consumed	Number of Slices	3923 (33 %)
	Number of Slice LUTs	10277 (22 %)
	Number of BRAMs	102
	Memory RAM	29 %

The analysis of the consumed resources shows that global usage of the Spartan-6 FPGA is around 30% of its total capabilities. Considering this fact, the designed controller may be developed in an inferior FPGA reducing the overall cost of the system.

7 Conclusion

In this paper, the state-of-the-art in PMSM torque control is analysed and an advanced FPGA torque controller is fully designed for traction control of hybrid and electric vehicles. The parallel processing of FPGA hardware architecture is gathered with floating point computation capabilities of its software processor cores, introducing many advantages as multitasking, floating point computation and fixed processing latency allowing time critical tasks execution.

It is presented an analysis of the PMSM working principle and 4 i_d injection methods are described. The torque controller is designed according to MTPA as it is the proper method for the application requirements. This method is implemented using Space Vector PWM for it has the better dynamic response and allows a better DC voltage usage in comparison to other techniques known. Regarding of position feedback, two methodologies are compared, one with position measurement and another with position estimation.

The simulation tests performed in PSIM® software corroborate the initial assumptions of the theoretical concept. Both methodologies performance are analysed in total speed range, including operation from standstill. All the torque range is achieved, including a torque extension with a maximum produced torque of 46 Nm in all the speed range of the motor. Also, energy regeneration is achieved. However, for low speeds, the sensorless technique presents estimation errors that affect the produced torque. Therefore, it is suggested by the authors the usage of this technique only for higher speed ranges, using the position measurement for start and low speeds. An integration of both methodologies constitutes an added value for the overall system performance and increases its fault tolerance.

The experimental results match entirely the expected results which proves that the developed controller is a very robust software and hardware platform, very versatile and mainly with a good dynamic response which allows the control of different permanent magnet synchronous machines with just an adjustment of few parameters.

References

[1] M. Ehsani, Y. Gao, and A. Emadi, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design*: CRC PressINC, 2010.

[2] C. C. Chan, "The state of the art of electric and hybrid vehicles," *Proceedings of the IEEE*, vol. 90, pp. 247-275, 2002.

[3] N. Hashemnia and B. Asaei, "*Comparative study of using different electric motors in the electric vehicles*," in *Electrical Machines, 2008*. ICEM 2008. 18th International Conference on , vol., no., pp.1,5, 6-9 Sept. 2008.

[4] Z. Q. Zhu and D. Howe, "*Electrical Machines and Drives for Electric, Hybrid, and Fuel Cell Vehicles*," *Proceedings of the IEEE*, vol. 95, pp. 746-765, 2007.

[5] A. Cavagnino, M. Lazzari, F. Profumo, and A. Tenconi, "*A comparison between the axial flux and the radial flux structures for PM synchronous motors*," *Industry Applications, IEEE Transactions on*, vol. 38, pp. 1517-1524, 2002.

[6] Z. Ping, Z. Jing, L. Ranran, T. Chengde, and W. Qian, "*Magnetic Characteristics Investigation of an Axial-Axial Flux Compound-Structure PMSM Used for HEVs*," *Magnetics, IEEE Transactions on*, vol. 46, pp. 2191-2194, 2010.

[7] B. K. Bose, *Power Electronics And Motor Drives: Advances And Trends*: Elsevier Science & Tech, 2006.

[8] Marufuzzaman, M.; Reaz, M. B I; Ali, M. A M, "*FPGA implementation of an intelligent current dq PI controller for FOC PMSM drive*", Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference on , vol., no., pp.602,605, 5-8 Dec. 2010.

[9] Alecsa, B.; Cirstea, M.N.; Onea, A., "*Simulink Modeling and Design of an Efficient Hardware-Constrained FPGA-Based PMSM Speed Controller*", Industrial Informatics, IEEE Transactions on, vol.8, no.3, pp.554,562, Aug. 2012.

[10] G. McPherson, *An introduction to electrical machines and transformers*: Wiley, 1981.

[11] B. Nicola, "*Permanent Magnet Synchronous Motors*," in *Power Electronics and Motor Drives*, ed: CRC Press, 2011, pp. 1-46.

[12] R. Krishnan, *Electric motor drives: modeling, analysis, and control*: Prentice Hall, 2001.

[13] Z. Yan, "*Control and Observation of Electric Machines by Sliding Mode*," Ohio State University, Ohio 2002.

[14] Nahid-Mobarakeh, B.; Meibody-Tabar, F.; Sargos, F.-M., "*Back-EMF estimation based sensorless control of PMSM: robustness with respect to measurement errors and inverter irregularities*," Industry Applications Conference, 2004. 39th IAS Annual Meeting. Conference Record of the 2004 IEEE , vol.3, no., pp.1858,1865 vol.3, 3-7 Oct. 2004

[15] S Taghavi, S.M.; Jain, M.; Williamson, S.S., "*A comparative study of sensorless control techniques of interior permanent magnet synchronous motor drives for electric vehicles*," Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE , vol., no., pp.1,7, 6-9 Sept. 2011

[16] Jun Sung Park et al. "*State Observer with Stator Resistance and Back-EMF Constant Estimation for Sensorless PMSM,*" TENCON 2010 - 2010 IEEE Region 10 Conference , vol., no., pp.31,36, 21-24 Nov. 2010

[17] Hongryel Kim; Jubum Son; JangMyung Lee, "*A High-Speed Sliding-Mode Observer for the Sensorless Speed Control of a PMSM,*" Industrial Electronics, IEEE Transactions on , vol.58, no.9, pp.4069,4077, Sept. 2011

[18] Changsheng Li; Elbuluk, M., "*A sliding mode observer for sensorless control of permanent magnet synchronous motors*," Industry Applications Conference, 2001. Thirty-Sixth IAS Annual Meeting. Conference Record of the 2001 IEEE , vol.2, no., pp.1273,1278 vol.2, Sept. 30 2001-Oct. 4 2001

[19] Hao Zhu; Xi Xiao; Yongdong Li, "*A simplified high frequency injection method for PMSM sensorless control,*" Power Electronics and Motion Control Conference, 2009. IPEMC '09. IEEE 6th International , vol., no., pp.401,405, 17-20 May 2009.

[20] Preindl, M.; Schaltz, E., "*Sensorless Model Predictive Direct Current Control Using Novel Second-Order PLL Observer for PMSM Drive Systems*," Industrial Electronics, IEEE Transactions on , vol.58, no.9, pp.4087,4095, Sept. 2011.

[21] Burgos, R.P.; Kshirsagar, P.; Lidozzi, A.; Jihoon Jang; Wang, F.; Boroyevich, D.; Rodriguez, P.; Seung-Ki Sul, "*Design and Evaluation of a PLL-Based Position Controller for Sensorless Vector Control of Permanent-Magnet Synchronous Machines,*" IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on , vol., no., pp.5081,5086, Nov. 2006.

[22] K. Samir, L. José, F. Leopoldo, R. José, and W. Bin, "*DC-AC Converters,*" in Power Electronics and Motor Drives, ed: CRC Press, 2011, pp. 1-50.

Authors

J. Ricardo Soares got M.Sc.Eng. degree in electrical and computers engineering, major Automation, in 2012, from Faculty of Engineering, Oporto University, Porto, Portugal. Currently, he is an Assistant Professor for Industrial Electronics course, in the Department of Electrical and Computers Engineering and a Research Engineer in traction control systems for electric and hybrid electric vehicles since September 2012.

Tiago Sá graduated with M.Sc.Eng. degree in 2012 in electrical and computers engineering, major Automation, from the Faculty of Engineering, Oporto University, Porto, Portugal. He is currently an Assistant Professor for Industrial Electronics course, in the Department of Electrical and Computers Engineering. Also, he is a Research Engineer in traction control systems for electric and hybrid electric vehicles since September 2012.

Armando Araújo graduated in electrical and computers engineering in 1984 and he got M.Sc.Eng. and Ph.D. degrees in 1990 and 1998, all from the Faculty of Engineering, Oporto University, Porto, Portugal, where he has been since 1984. Currently, he is an Assistant Professor in the Department of Electrical Engineering and Computers, where he is engaged in research activities in power electronics semiconductors and converters modelling, control, and simulation.

Adriano Carvalho (M’89) received the Graduate degree in electrical engineering in 1976, the Ph.D. degree in electrical and computers engineering in 1989, and the title of Aggregate Professor in 1997, from the Faculty of Engineering, Oporto University, Oporto, Portugal. He started a full-time academic career in1976, and currently, he is an Associate Professor in the Department of Electrical Engineering and Computers, where he is engaged in research activities in automation and control.