BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 62 (66), Numărul 1, 2016
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

MODELLING AND SIMULATION OF AN E-BIKE APPLICATION WITH A PMSM FIELD-ORIENTED CONTROL BY USING SYSTEMC-AMS LANGUAGE

BY

ALEXANDRA IOSUB[1,2,*], JEROME KIRSCHER[2], MONICA RAFAILĂ[2], ANDI BUZO[2], GEORG PELZ[2] and LIVIU GORAŞ[1,3]

[1]Technical University “Gheorghe Asachi” of Iaşi,
[2]Infineon AG Neubiberg, Germany,
[3]Institute of Computer Science, Romanian Academy, Iaşi

Received: March 11, 2016
Accepted for publication: March 31, 2016

Abstract. The paper presents an electric bicycle (E-Bike) application developed with SystemC-AMS, a C++ based language for system-level description, model refinements and simulation improvements. The mathematical model of PMSM is analysed and implemented using Electrical Linear Network (ELN) and Time Data Flow (TDF) formalisms from AMS extension. A principle Field Oriented Control (FOC) algorithm is proposed and developed with Transaction Level Modelling (TLM) from SystemC.

We show that it is possible to model, validate and optimize a complex system using only one description language, SystemC-AMS, an open-source C++ library.

Key words: SystemC-AMS; PMSM; E-Bike; PI; current sensors; TLM; ELN; TDF.

1. Introduction

Permanent magnet (PM) synchronous motor has gained an increasing popularity in applications such as electric vehicles due to its following assets: high torque/inertia ratio, high power density and high efficiency. In this paper

[*]Corresponding author: *e-mail*: aiosub@etti.tuiasi.ro

we investigate an E-Bike application (Fig. 1). An E-Bike is a bicycle with an integrated electric motor which can be used for propulsion. Modeling such an application implies dealing with heterogeneous simulations in terms of signal type (digital and analog) as well as operating frequencies and a long simulation time which has to be spent in order to reach the steady-state operation (after the transients characterized by large time constants due to system inertia have passed). A possible way to decrease simulation time is to change the abstraction level in which the system is defined, such that the system performs overall mostly in the same way but with less simulation complexity.

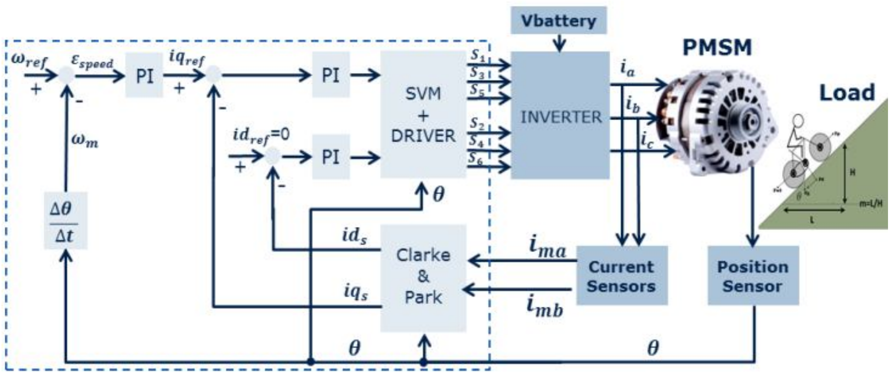

Fig. 1 – Diagram block of an E-Bike.

Given that some modeling languages like VHDL-AMS or Verilog-AMS reach rapidly their limits regarding simulation performance and interoperability, it is interesting to experiment such complex architectures with SystemC-AMS (http://www.systemc-ams.org.). The SystemC-AMS modeling language offers means to describe a system as a full virtual prototype, with software simulation capability, which can provide a thorough view of the system physical characteristics (electrical, mechanical etc.) that can be simulated with reduced license costs (Ting-Ting, Liu; Fan, Xuan; Lee Stephens; Dave Wilson; Jingyu Liu, Xueli Li, et al). It introduces two models of computation, SystemC-AMS synchronous dataflow (SDF) and SystemC-AMS conservative. In addition to the SystemC event-driven model of computation (for digital description), we used such SystemC-AMS views to model the PMSM, the sensors used in the application and the Inverter Block.

The paper offers an overview of the components: a PMSM, a microcontroller, PI controllers, sensors and discusses the implementation of the PMSM ("ABC" model) and the microcontroller model.

2. Application Description

Apart from the electric motor, the electrical part of an E-bike contains: a controller block, a power stage consisting of a transistor driver and an inverter circuit, sensors for electrical signals (three phase currents) and mechanical signals (motor position). All these components are presented in Fig.1. The PMSM motion is controlled by applying the Field-Oriented Control (FOC) algorithm (Veltman *et al.*, 2007) that has greatly improved the performance of AC motor drives. These control strategies use pulse width modulated (PWM) switching strategies aimed at producing a precisely controlled current to the windings of the motor.

2.1. Application Overview

The FOC aims to control both torque and flux to force the motor to accurately track the desired values. This control is performed by regulating the motor current, being mandatory to have the angle information for this. To apply the FOC algorithm the electrical equations are projected from a 3 phase non-rotating frame into a 2 co-ordinate rotating frame by using mathematical transformations. Such transformations are often used to decouple variables, to facilitate the solution of difficult equations with time varying coefficients (M. Trzynadlowski). The concept behind FOC is to control the important electrical variables (V_d, V_q, I_d, I_q) which are turning with the rotor. Thus the currents measured at the stator ($I\alpha$, $I\beta$) have to be transformed into the rotor coordinates (I_d, I_q). The controller for the currents is done in the rotating system as PI-controller, whereas the field exciting d-component and the torque exciting q-component are controlled separately. Clarke transformation converts a three phase system into a two phase system with orthogonal axes in the same stationary reference frame (α, β plane) while Park transformation changes a three phase system into one stationary reference frame into a two phase system with orthogonal axes in a different rotating reference frame (Fig. 2). Thus the PMSM control is equivalent to that of the dc motor.

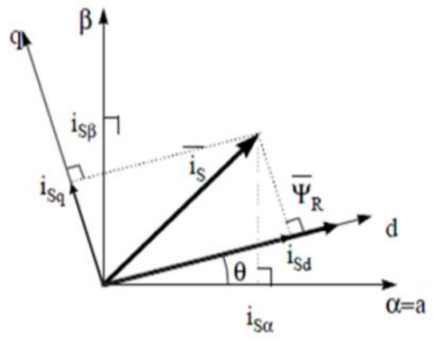

Fig. 2 – Stator Current Space Vector and the α, β components in the d,q rotating reference frame.

The field oriented control of the PMSM is derived from its dynamic model. The stator currents that, by using Park’s transformation, must be transformed to the rotor reference frame are:

$$\begin{cases} i_a = I_s \sin\left(\omega_r t + \theta_0\right), \\ i_b = I_s \sin\left(\omega_r t + \theta_0 - 2\pi/3\right), \\ i_c = I_s \sin\left(\omega_r t + \theta_0 + 2\pi/3\right), \end{cases} \tag{1}$$

where: θ_0 is the angle between the rotor field and stator current phasor and ω_r is the electrical rotor angular speed. Note that the q and d axis currents are constant in the rotor reference frames since θ_0 is constant for a given resistive (load) torque.

The electrical and mechanical equations of the PMSM in the rotor reference (d-q) frame are (Pragasan & Krishnan, 1988):

$$v_q = R_q i_q + L_q \frac{\mathrm{d}i_q}{\mathrm{d}t} + L_d \omega_e i_d + \lambda \omega_e, \tag{2}$$

$$v_d = R_d i_d + L_d \frac{\mathrm{d}i_d}{\mathrm{d}t} - L_q \omega_e i_q. \tag{3}$$

The (d-q) fluxes are:

$$\lambda_d = L_d I_d + \lambda,\ \lambda_q = L_q I_q.$$

The principle in controlling a PMSM (Fan Xuan, 2010; Hughes *et al.*, 2013) drive is based on field orientation. Because the magnetic flux generated from the permanent rotor is fixed in relation to the rotor shaft position, the flux position can be determined by the shaft position sensor. If the reference of d-axes current is set to 0 (I_d = 0), then, the d-axis flux linkage λ_d is fixed. Since flux on q-axis is constant for a PMSM (λ = ct.), the electromagnetic torque is proportional to the q-axis current I_q which is determined by closed-loop control (Grellet & Clerc, 2000):

$$T_e = \frac{3}{2} p \left[\psi_m - \left(L_q - L_d\right) i_d \right] \xleftrightarrow{i_d = 0,\ L_q = L_d} \frac{3}{2} p \lambda i_q, \tag{4}$$

where: p represents the number of pole pairs.

The above equation is used in steady state region in order to check if the amplitude of the currents is correct. The rotor flux is produced only in the q-axis while the current vector is generated in the axis in the field-oriented control. Since the generated motor torque is linearly proportional to the q-axis current, as the d-axis rotor flux is constant, a maximum torque per ampere can be achieved. Three PI regulators have been used in the control system: one for the mechanical loop (the loop that regulates the angular speed) and two for the electrical loops (the loops that regulate the d and q currents). The corresponding outputs of the PI controllers (that represent the voltages to be applied to the motor, referenced to the stationary frame) are then passed through an inverse Park and Clarke transformation to convert the variables back to the 3-phase stator reference. The 3-phase voltages (which are continuous sine waves) are finally converted to PWM signals using the Space Vector Modulation (SVPWM) technique (Trzynadlowski, 1994). The main objective is to approximate the reference voltage vector utilizing eight switching patterns. In SVPWM, the three phase variables are expressed as space vectors. Each desired voltage vector can be simulated by an averaging effect between two adjacent active vectors and a zero vector. The six active vectors divide the space vector plane into six equal sized sectors of 60º with equal magnitude which form an origin centered hexagon, and two zero space vectors found at the origin as shown in Fig. 3.

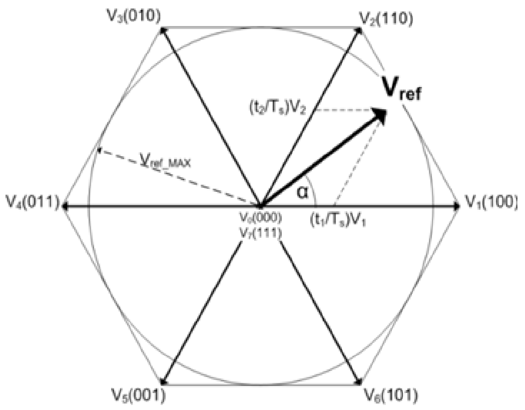

Fig. 3 – Six voltage vectors within the generator resulting from state 1 to 6 of the inverter.

The length of the six active space-vectors, denoted by V_1 to V_6 is $2V_{DC}/3$, each separated by 60°. The phase angle and voltage reference vector are calculated based on input voltage components. The modulation index is calculated based on 3 time periods (T_a, T_b, T_0). The reference voltage vector section is found from the adjacent space vectors based on sector angle. The time intervals T_a, T_b, T_0 are computed based on T_{carry} and phase angle. Fig. 3 illustrates directions and amplitudes of the space vector voltages applied to the motor corresponding to the six active states. These vectors convey information about the direction the inverter is attempting to establish a magnetic field inside the motor. To control the torque output of the generator, a smoothly rotating vector should be created (Fig. 3). The PWM signals (designed to be right-aligned) are applied to an inverter block which feeds power into the machine according to the control. In order to close the current loops, two currents (the current from the A winding and the current from the B winding) are measured by taking two samples at the beginning of the PWM period, every FOC cycle. Once the samples are taken, the FOC begins the ADC conversion of these samples (with 14 bit resolution), and the currents are reconstructed. The angular speed is calculated as the derivative of the angle with respect to time: $\omega = \mathrm{d}\varphi/\mathrm{d}\tau$.

To implement the FOC, the value of the currents passing through the three windings of the motor must be precisely known. There are different methods to sense the current passing through the windings of the motor: direct measurement on the phase line by using Hall sensors, measurement on the low side of the inverter block and measurement on the bus voltage path. In the presented E-Bike system we have followed the second approach. Putting a shunt resistor on the low side path generates a voltage drop proportional to the current which can then be amplified by standard operational amplifiers and fed to the microcontroller's analog to digital converter.

In this paper we have done simulations with both type of sensors (based on shunt resistor and based on Hall efect). Both sensors can be automatically selected in the system.

For motor position detection a sensor based on the Giant Magnetic Resistor (GMR) effect is used. The sensor converts the motor angle to two voltages that represent the sine and the cosine of the actual motor angle with respect to the stator frame. After analog to digital conversion, the angle information is reconstructed inside the control algorithm by using the arctangent function which calculates the digital angle value as: $\theta = \tan^{-1}\left(V_Y/V_X\right)$. The motor position is required for the SVPWM technique and also for the motor speed reconstruction.

In our investigation we have considered a motor that has appropriate specifications for an E-Bike which requires higher torque and a lower top speed. The motor parameters are presented in Table 1 (http://www.ebike.heinzmann.com/en/systems/directpower/motor).

Table 1
Motor Parameters

DC Voltage, [V]	36
Rated power, [W]	250
Speed limit, [Km/h]	25
Impulse torque, [Nm]	60
EMF constant, [Wb]	0.125
Moment of inertia, [Kg]	0.0102
Resistance phase, [mΩ]	88
Winding inductance, [mH]	0.36

2.2. System transfer functions

The goal of tuning of the PI coefficients of the speed loop is maximizing responsiveness and avoiding instability (Wilson, 2009; Jingyu Liu, *et al.*, 2011). The PID controller is the most common form of feedback. The output of a PID controller, equal to the control input to the plant, in the time-domain is as follows:

$$u(t) = K_p e(t) + K_i \int e(t)\mathrm{d}t + K_p \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \tag{5}$$

where: u is the control signal and e is the error from the reference value. The reference value is often called the set point. The transfer function of a PID controller is found by taking the Laplace Transform:

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}, \tag{6}$$

where: K_p is the proportional gain, K_i – integral gain, K_d – derivative gain.

If the derivative part of a PID is 0 (K_d = 0 the controller is PI. We use a first-order approximation of the motor winding to be a simple series circuit containing a resistor, an inductor, and a back-EMF voltage source. Assuming that the back-EMF voltage is a constant for now (since it usually changes slowly with respect to the current), we can define the small-signal transfer function from motor voltage to motor current as:

$$\frac{I(s)}{V(s)} = \frac{1/R}{1 + s\dfrac{L}{R}}. \tag{7}$$

Small signal tuning has been done by taking into account the entire system transfer function. The parameters of the PI controllers in current loop are obtained by using the pole placement method and the pole of the current PI controller was compensated by introducing a pole-zero cancellation (Dave Wilson). Two factors from the set of control parameters are selected to be adjusted in order to explore the system performances: δ (“damping” factor) and τ_{LPF} (time constant of the low pass filter used to filter the speed response). Note that this is a digital filter and τ_{LPF} gives the system. The total moment of inertia seen by the motor (J) is another critical parameter as it depends on the load and it determines the gain of the speed loop. The open and closed loop currents transfer functions are:

$$H_{\mathrm{crt_{OL}}}(s) = PI_{\mathrm{crt}}(s)\frac{I(s)}{V(s)} = \frac{1}{s}\left(\frac{K_p}{L}\right);$$

$$H_{\mathrm{crt_{CL}}}(s) = \frac{1 + \dfrac{s}{K_p/K_i}}{\left(1 + s\dfrac{R}{K_i}\right)\left(1 + \dfrac{s}{K_p/K_i}\right)}.$$

The open loop transfer function for speed is:

$$H_{\text{speed}_{\text{OL}}}(s)=\frac{1}{1+s\tau_{\text{LPF}}}\cdot\frac{K_{iw}\left(1+\dfrac{s}{K_{pw}/K_{iw}}\right)}{s}\cdot\frac{1}{1+s\dfrac{L}{K_{pw}}}1.5p\lambda_r\frac{1}{sJ},$$

where: K_{pw}, K_{iw} are proportional and integral coefficients of the speed PI controller.

3. SystemC and SystemC-AMS Models of the PMSM and the Microcontroller

All the components of the motor drive system were modeled in SystemC and SystemC AMS (SystemC-AMS, http://www.systemc-ams.org.):

a) the microcontroller was modeled as an algorithmic block, by using the Transaction Level Modeling (TLM) formalism;

b) the Driver block modeled with the Discrete Event (DE) formalism;

c) the Inverter circuit was modeled with the Discrete Event (DE) and Electrical Linear Networks (ELN) formalism;

d) Timed Data Flow (TDF) and ELN were used for the motor model;

e) ELN + TDF were used for modeling the sensors.

The components are generic. Their behavior and internal structure is parameterized to a reasonable degree to allow their adaptation to different specifications.

3.1. The PMSM Model in SystemC-AMS

The motor is modeled in SystemC-AMS based on the theoretical “ABC” equations of a PMSM (Veltman *et al.*, 2007). The model describes a star connection of three windings modeled as a series connection of R, L and Back EMF controlled voltage sources. The simplified electromechanical circuit of the PMSM model is shown in Fig. 4. The electro-mechanical part is described with Laplace Transfer Functions in TDF (Timed Data Flow), while the rest of the model is described using ELN (Electrical Linear Networks).

The modeling of electrical networks is supported by instantiating predefined linear network primitives such as resistors and capacitors, which are used as models for describing the continuous-time relations between voltages and currents. A restricted set of linear primitives and switches is available to model the electrical energy conserving behavior. The execution semantics based on TDF introduce discrete-time modeling and simulation without the overhead of the dynamic scheduling imposed by the discrete-event kernel of SystemC. Simulation is accelerated by defining a static schedule, which is computed before simulation starts, and which executes the processing functions of the scheduled TDF modules according to the stream direction of the dataflow.

The phase voltages of each line to ground are:

$$v_{a,b,c} = Ri_{a,b,c} + L\frac{\mathrm{d}i_{a,b,c}}{\mathrm{d}t} + e_{a,b,c}(\theta), \tag{8}$$

where: $i_{a,b,c}$ are the three-phase currents flowing through the windings of the motor. If we suppose that there are no mutual inductances between the motor windings, the rotor flux through each winding is:

$$\begin{bmatrix}\Psi_a\\ \Psi_b\\ \Psi_c\end{bmatrix} = \begin{bmatrix}L_{aa} & L_{ab} & L_{ac}\\ L_{ba} & L_{bb} & L_{bc}\\ L_{ca} & L_{cb} & L_{cc}\end{bmatrix}\begin{bmatrix}i_a\\ i_b\\ i_c\end{bmatrix} + \Psi_m\begin{bmatrix}\cos\theta_r\\ \cos\left(\theta_r - 2\pi/3\right)\\ \cos\left(\theta_r - 4\pi/3\right)\end{bmatrix}. \tag{9}$$

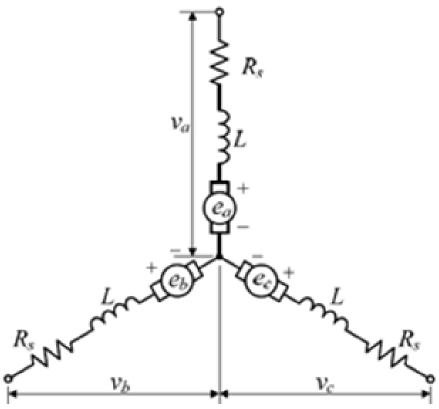

Fig. 4 – Electrical model of PMSM.

The back EMF voltage of each branch is:

$$\begin{bmatrix}e_a(\theta)\\ e_b(\theta)\\ e_c(\theta)\end{bmatrix} = -\omega_e\Psi_m\begin{bmatrix}\sin\theta_r\\ \sin\left(\theta_r - 2\pi/3\right)\\ \sin\left(\theta_r - 4\pi/3\right)\end{bmatrix} = -\omega_e\Psi_m\left[K(\theta_e)\right], \tag{10}$$

where: $K(\theta_e)$ is the flux linkage of each of the stator windings, as a function of motor angle, and Ψ_m is the amplitude of flux linkage.

The electromagnetic torque of each phase is:

$$\begin{cases} T_a = K_T\cos\left(\theta_e\right)i_a,\\ T_b = K_T\cos\left(\theta_e - 2\pi/3\right)i_b,\\ T_c = K_T\cos\left(\theta_e + 2\pi/3\right)i_c.\end{cases} \tag{11}$$

The total electromagnetic torque is the sum of torques for each phase:

$$T_e = T_a + T_b + T_c. \tag{12}$$

The equations above are used in the SystemC models by using ELN and TDF approach. Under sinusoidal steady-state conditions in the rotor reference frame, all variables are constant. Thus we may set all derivative terms in the voltage equation equal to zero. The rotational angle denoted as θ_r is the electrical angle θ_e, where $\theta_e = p\theta_m$ (θ_m is the mechanical angle).

The mechanical behavior is described by the equation:

$$J_m \frac{\mathrm{d}\omega_r}{\mathrm{d}t} = T_e - F\omega_r - T_{\mathrm{Load}}. \tag{13}$$

The rotor mechanical speed can thus be deduced as:

$$\omega_r = \int \frac{T_e - F\omega_r - T_{\mathrm{Load}}}{J_m}\,\mathrm{d}t, \tag{14}$$

where: J_m is the moment of inertia, F is the friction coefficient, T_{env} is the resistive torque applied to the shaft (the load of the system) and T_e is the electrical torque developed by the motor. Finally, the motor angle is given by:

$$\theta = \int \omega_r \mathrm{d}t, \tag{15}$$

3.2. The Microcontroller Model in SystemC

SystemC supports the refinement of HW/SW systems down to cycle-accurate behavior by providing a discrete-event simulation framework called Transaction-level modeling (TLM). TLM is a system-level abstraction level which describes the system in terms of initiators, targets and intermediate channels. It allows modeling the transmission of signals between electronic components at an abstract level.

The TLM modelling of the microcontroller offers the possibility to generate signals and describe blocks with different simulation time steps thus making possible a considerable improvement of the simulation time.

The FOC algorithm used in order to spin a PMSM is implemented within the microcontroller block. The model of the microcontroller (Fig. 5 *a*) contains:

a) the ADC peripheral (for reading the sensors that produce analog signals);

b)the PWM peripheral (to control power converters, resistive loads, motors, etc.);

c) the TLM timer and a PWM clock;

d) interrupts (signal the processor to suspend processing the current instruction sequence and to begin an interrupt service routine);

e) an algorithmic block.

The algorithmic block contains the following sub-blocks: a scheduler block, an application routine (the software part, where all FOC functions are implemented) and a Hardware Abstraction Layer (HAL) block used to call functions such as "read_register" and "write_register", functions used to initiate a TLM read or write access. HAL operates as a kind of interface between the software parts and the hardware model of the microcontroller.

A scheduler process is sensitive to a rising clock edge. The main task of the scheduler is to run the function (called "DO_FOC") which computes the values of the duty cycles of the PWM signals which are fed to the inverter circuit. The duty cycle values are computed according to the Field Oriented Control algorithm (Fig. 5 *b*). This function will run at each period of time given by the PWM frequency (f_{PWM} = 20 KHz, τ_algorithm = 50 μs). The choice of the PWM frequency depends on the motor electrical constant L/R. If the PWM frequency is too low, audible noise can be heard from the motor. In general, PWM frequencies are above 20 kHz, however, the higher the frequency, the higher the switching losses will be in the Inverter block. The PWM switching frequency has to be much higher than the value that would affect the load (the device that uses the power), which is to say that the resultant waveform perceived by the load must be as smooth as possible.

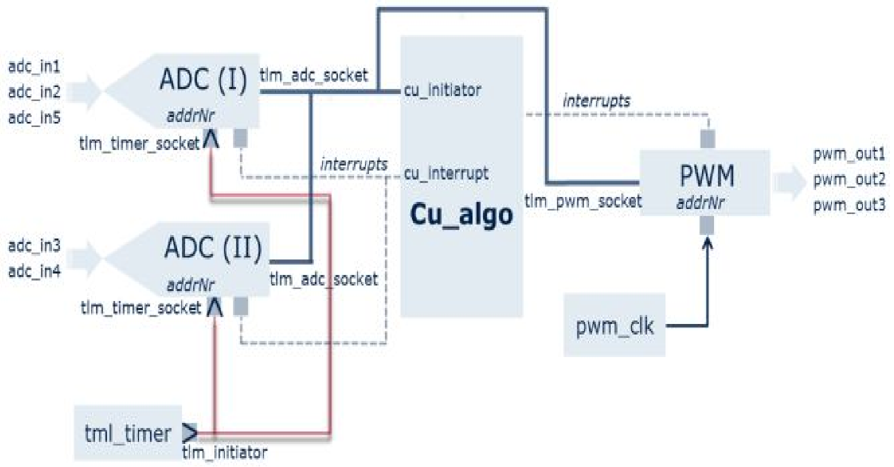

Fig. 5 *a* – Microcontroller blocks.

In addition, digital PWM generators can suffer from time quantization effects. This is a direct consequence of digital systems being driven by a clock of finite period (frequency), usually in the range of a few MHz to 100 MHz. This directly dictates the time-step granularity offered by DPWM generators, and can be no finer (better) than the highest running clock, for example a 50 MHz clocked DPWM can offer no better than 20 ns time-step adjustment of duty cycle. This quantization effect can lead to serious problems in a control loop and is often known as Limit Cycling, the PWM resolution must be at least

2 times better than the resolution used. Because of the time simulation constraints, further simulations will be done by using the averaged model of the inverter block. Therefore, the requirement of a better resolution for PWM that ADC is not fulfilled (the equality between PWMs and ADCs resolutions is chosen).

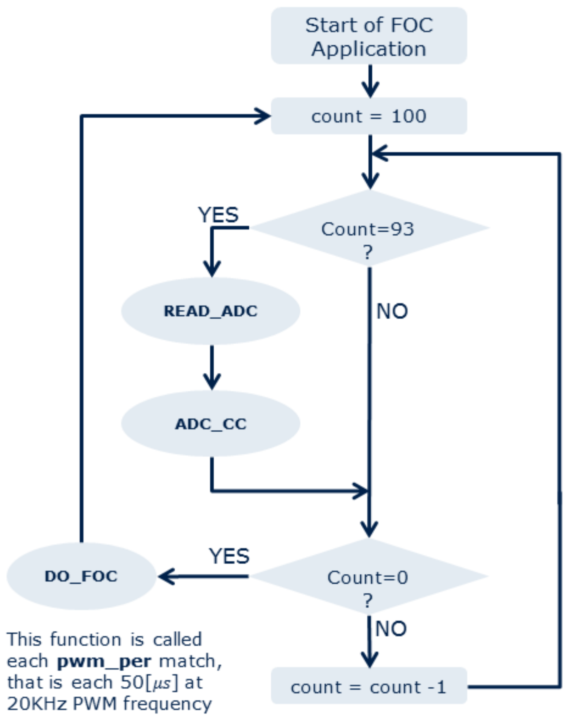

Fig. 5 *b* – Flow chart of the microcontroller states.

The carrier signal of the PWM is generated by using an internal clock. The resolution of this signal is 500 ns. Because the “right aligned sequence” is used, the counter corresponding to the internal clock will count down from 100 (*pwm_per*/*pwm_res* = 50 µs/ 500 ns) to 0. The scheduler block has also a timer which is set in down count mode and generates a two types of periodical interrupt events: first interruption called “READ_ADC” which is needed to ensure the synchronization between ADC and PWM blocks (this interrupt is mandatory when shunt resistor is used); the second interruption, called “ADC_CC” is used to indicate the end of conversion performed by ADC.

The flow chart of the microcontroller states is shown in Fig.5b where the main function of “FOC Application” includes the two loops: first loop (which ends when the value of internal counter matches 93) is needed for PWM-ADC synchronization and the second loop (which ends when the value of internal counter matches 0) is the needed for duty cycle computation.

The main steps of “DO_FOC” function are:

1. Read the i_a and i_b phase currents (from current sensors).
2. Read the motor position (from angle sensor).
3. Compute the angle information by using atan function: θ = tan⁻1(V_Y/V_X), where V_Y and V_X are sine and cosine analog output voltages of the angles sensor.
4. Compute the speed information: $\omega = \mathrm{d}\theta/\mathrm{d}t$.
5. Filter the speed (by using a low pass filter).
6. Apply Clarke transformation to determine the stator current projection in a two co-ordinate non-rotating frame
7. Apply Park co-ordinate transformation in order to obtain the currents projection in the (d, q) rotating frame.
8. Regulate the electrical speed feedback signal in order to obtain the desired reference speed.
9. Regulate the stator phase currents i_d and i_q, compared to their reference ($I_{d\mathrm{REF}} = 0$, $I_{q\mathrm{REF}} = T_e/1.5p\lambda$) in order to obtain the corresponding amplitude given by the electrical torque feedback.
10. The outputs of the current controllers are passed through the inverse Park transformation.
11. A new stator voltage vector is applied to the motor using the Space Vector Modulation technique (Jingyu Liu & Xueli Li *et al.*).

The presented flow is not generally valid. The sampling rate of the outer speed loop is often slower than that of the inner current loop. Even if different sampling rates are supported by the controller design, we used the assumption (which is valid in our case) that the speed is sampled as often as the currents loop. In a real application, different sampling rates are used depending on the targets.

The simulation time step of the SystemC-AMS modules (apart from the microcontroller) is set to 1 MHz (when the Shunt resistor is used as a current sensor), or equal with the PWM frequency, 20 KHz (when the Hall sensor is used for current sampling). It is possible to use different simulation time steps for digital blocks (the microcontroller works at 2 MHz) and analogue parts (the motor, sensors, Invertor works at 1 MHz) when TLM is used. Even though the modeled analog blocks do not respect the Nyquist frequency, a 1 MHz frequency was preferred because of the long simulation time, but more important, because there was not a significant improvement on the results when the simulation time step of those modules was set at 4 MHz. Therefore, the advantage of a fast simulation was preferred.

Next, simulations are done in the nominal case, where all components are ideal (no sensor errors); also the motor has an ideal flux (without harmonics). This ideal test scenario is necessary in order to check if the control algorithm is working properly. A step is applied in the speed reference signal from 0 to W_{ref} = 23.3 rad/s (see Fig. 6 *a*), while the human torque is kept constant (applied from the beginning of the simulation). The load inertia is set to J_L = 6 Kg.m^2.

In Fig. 6 *b* it can be seen that the electrical torque developed by the motor matches the resistive torque in steady state ($T_{\text{res}} = T_{\text{env}}$). The electrical torque response is a smooth waveform in steady state (Fig. 6 *b*). For these simulations, the average model of the Inverter block is used in transient phase for faster simulations and the switching model of the Inverter in steady state. The acceleration time is small (2.2 s) as long as in the first step of calibration we followed to obtain a small rising time of the speed response. The three phase currents through the motor windings from in Figs. 7 and 8 show that PI controllers are well tuned: $I_d = 0$ (while the reference is set to be 0), $I_q = I_{q\text{REF}}$ and V_d , V_q are not saturated. Under sinusoidal steady-state conditions in the rotor reference frame, all variables are constant. The amplitude of these currents is:

$$I_s = \sqrt{I_q^2 + I_d^2} \tag{16}$$

where: $I_d = 0$ A (under FOC control) and $I_q = T_e/1.5p\lambda$.

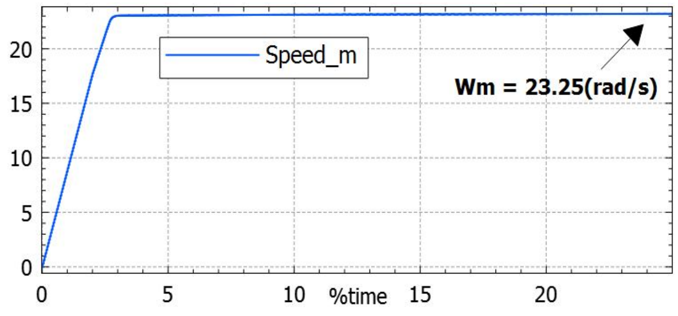

Fig. 6 *a* – Mechanical speed.

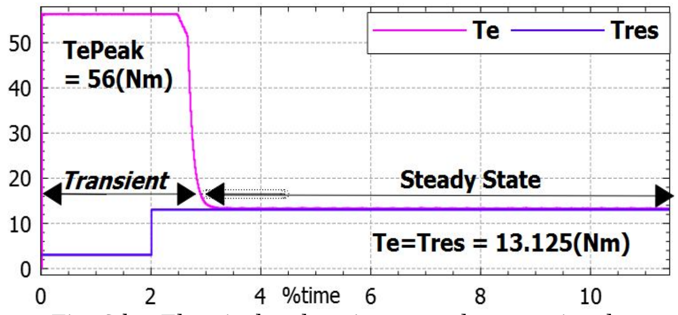

Fig. 6 *b* – Electrical and environmental torque signals.

If we consider that the friction $F = 0$ and $T_{\text{env}} = 0$, then, the electrical torque is: $T_e = d\omega_e/dt$. For a resistive torque of about $T_{\text{env}} = 13$ N.m, the torque developed by the motor is:

$$T_e = \frac{\mathrm{d}\omega_e}{\mathrm{d}t} + T_{\mathrm{env}}. \qquad (17)$$

If the current control loops are working properly, then, in steady state operation, where the motor speed is constant, the i_d and i_q currents will be also constants. If $i_d = 0$, the amplitude of three phased currents will be given by:

$$I_q = \frac{T_e}{1.5p\lambda} = \frac{T_{\mathrm{res}}}{1.5p\lambda} = \frac{13}{1.5\times 4\times 0.125} = 17.333 \text{ A}.$$

The amplitude of the three phase currents is:

$$I_{a,b,c} = \sqrt{I_d^2 + I_q^2} \xleftrightarrow{I_d=0} I_q = 17.333 \text{ A}.$$

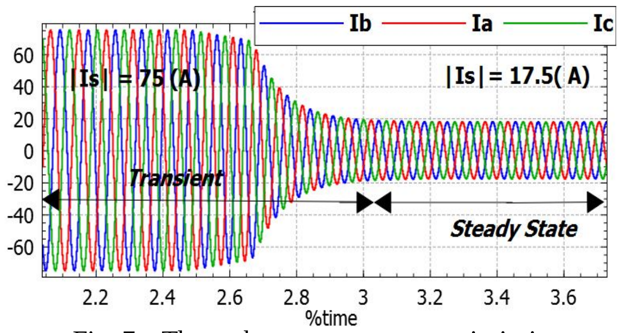

Fig. 7 – Three phase motor currents: i_a, i_b, i_c.

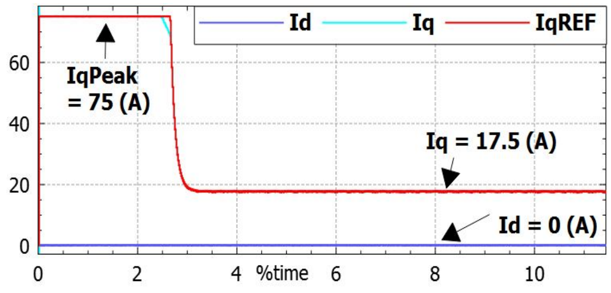

Fig. 8 – Control signals: I_d, I_q currents; V_d, V_q voltages.

5. Conclusions

Simulation results are according to the performance characteristics of a PMSM, which proves the accuracy of the FOC algorithm and the motor model.

In this paper we have presented an analog mixed signal application using SystemC and SystemC-AMS language. We show that it is possible to model, validate and optimize a complex system using only one description language, SystemC-AMS (an open-source C++ library). The simulation of one run in SystemC-AMS takes between 5 to10 min compared to about 1h in VHDL-AMS. Moreover, we are able to simulate both digital and analog blocks simultaneously (because in Matlab/Simulink, for example, is more difficult to perform microcontroller models).

Our intention for the future work is to do some refinements in the microcontroller algorithm for design closer model specifications, such as the sensors bandwidth.

Acknowledgments. The current thesis is a project developed with the support of Infineon Technologies MUC, for which I am very thankful.

REFERENCES

Ting-Ting Liu *et al.*, *Simulation of PMSM Vector Control System Based on Matlab/Simulink*, Measuring Technology and Mechatronics Automation, ICMTMA'09, **2**, IEEE (2009).

Fan Xuan, *Intelligent Power Assist Algorithms for Electric Bicycles*, 2010.

Lee Stephens, *The Significance of Load to Motor Inertia Mismatch*, Senior Motion Control Engineer Kollmorgen, www.kollmorgen.com.

Dave Wilson, *Teaching Your PI Controller to Behave*, Blogs: Motor Drive & Control, Motion Products Evangelist, Texas Instruments, 2009.

Jingyu Liu, Xueli Li *et al.*, *Steering (EPS) System Based on ADAMS*, Electric Information and Control Engineering (ICEICE), 2011 Internat. Conf., 2011.

Grimm C., Barnasconi Vachoux A.M., Einwich, K., *An Introduction to Modeling Embedded Analog/Mixed-Signal Systems Using SystemC AMS Extensions*, http://www.systemc.org/downloads/, 2008.

Hughes Austin, Bill Drury, *Electric Motors and Drives: Fundamentals, Types and Applications*, Newnes, 2013.

* * * http://www.ebike.heinzmann.com/en/systems/directpower/motor

* * * SystemC-AMS, http://www.systemc-ams.org

Veltman A., Pulle D.W.J, de Doncker R.W, *Fundamentals of Electrical Drives*, Springer Netherlands, 2007.

Pragasan P., Krishnan R., *Modeling of Permanent Magnet Motor Drives*, IEEE Trans. Industrial electronics, **35**, *4* (1988).

Grellet G., Clerc G., *Actuators Electric Principe/Model/Control* (2nd ed.), Eyrolles, 2000, pp. 298.

Trzynadlowski A.M., *The Field Orientation Principle in Control of Induction Motors*, Kluwer Academic Publishers, 1994.

PROIECTAREA LIMITĂRILOR ÎN REGULATOARELE PI PE BAZA EŞANTIONĂRII CURENȚILOR ÎN APLICAȚII DE CONTROLUL MOTOARELOR

(Rezumat)

Este prezentată o aplicație de semnal mixt analogic folosind limbajul de modelare SystemC şi SystemC-AMS. Formalismele de modelare utilizate pentru acest

sistem sunt specifice extensiei AMS: ELN (Electrical Linear Network) şi TDF (Time Data Flow), în timp ce modelul complet al microcontrolerului a fost realizat folosind biblioteca TLM (Transaction Level Modelling) disponibilă în SystemC.

De asemenea, se arată că este posibilă modelarea, validarea şi optimizarea un sistem complex, folosind doar un singur limbaj de descriere comportamentală, SystemC-AMS (bibliotecă C++ open-source). O simulare a sistemului în SystemC-AMS durează între 5 to10 min., comparativ cu circa 1 oră în VHDL-AMS. Mai mult decât atât, se poate modela şi simula simulatn atât blocuri digitale cât şi analogice (în Matlab/Simulink, de exemplu, este mult mai dificil de a realiza modele de microcontroler).