Hardware-In-the-Loop simulation of a DC-machine with INTEL FPGA boards

P. Saenger, M. Hilairet, *Member, IEEE,*
FEMTO-ST, CNRS, Univ. Bourgogne Franche-Comte, UTBM
FCLAB, CNRS, Univ. Bourgogne Franche-Comte, rue Thierry Mieg, F-90010 Belfort Cedex, France
Email: mickael.hilairet@univ-fcomte.fr

***Abstract*—This article details a low cost Hardware-In-the-Loop architecture for education and research. As application example, a speed controlled DC machine is detailed, where the emulated power converter, machine, position and current sensors are implemented on a INTEL FPGA DE1_SoC board, while the controller and peripherals are implemented on a second DE1_SoC board under test. The computed data are transferred to a personal computer and display for man-machine interface (IHM).**

I. Introduction

Hardware-In-The-Loop (HIL) or Power-Hardware-In-The-Loop (PHIL) are more and more used in the industries in order to validate the implementation of the controller on the electronic control unit (ECU) [1], [2]. In another hand, HIL and PHIL are useful simulation tools for research and education where unitary tests can be conducted on the ECU without damaging the real system under test [3]–[5].

The use of the Field Programmable Gate Array (FPGAs) for HIL and PHIL is generally mandatory because of their performances, their computing capacity and their ability to perform parallel processing of data; they are very good candidates for real-time simulation [6]. Repeating several times basic mathematical operations (multiplications, additions), parallel processing of the FPGA can perform the calculations required for real-time simulation with sample-period nearly equal to 100ns or higher [7].

However, the use of FPGA technology generally requires the use of fixed-point numbers where the precision and range are defined by the designer contrary to most processors that process floating-point data of 32 or 64 bits. It follow that the implementation of FPGA is not user-friendly and time consuming to manage the compromise between precision and the use of FPGAs resources. It's very difficult to optimize an aspect relative to the other. Also, the increase of the data streams reduces the performances in the FPGA, justifying the importance of the arithmetic operations optimization of the models under study. Therefore, the minimization of the FPGA resources is very important in order to find the best behavioural system performance [8].

Nowadays, System on Chip (SoC) offers all components of a regular processor and other peripherals into a single chip. SoC are very useful for electronics system due to their flexibility and power [9]. In fact, multiple software-processors (such as the NIOS II for INTEL FPGA boards) and/or hardware-processors (such as the ARM) can exist with regular glue-logic into a single chip offering flexibility with regular implementation with floating numbers. Moreover, high-level synthesis (HLS) tool such as Vivado HLS for Xilinx or INTEL HLS can offers an abstraction level in order to avoid VHDL or Verilog coding and where software directives lead to different software and hardware implementations [10], [11].

This article is dedicated to the study of a low cost Hardware-In-the-Loop (HIL) platform for education where VHDL blocks and soft-processor are designed. In this paper, a speed controlled DC-machine is detailed, as an example for such HIL platform, where the emulation of a DC-machine with its converter, position and current sensors are implemented on a INTEL FPGA DE1_SoC board, while the controller and the peripherals are implemented on a second DE1_SoC board. Some computed data are transferred to a personal computer and display on a man-machine interface (IHM).

In this work, the controller and associated peripherals are implemented on one INTEL FPGA DE1_SoC board and constitute the system under test. Therefore, all the control blocks have been designed into one FPGA board, while the emulated system has been designed in a totally separated FPGA board in order to have a physical separation between the controller and system that is representative of a real system. In an educational context, we could have used a single development board integrating all the command and system in a way to reduce the cost. But in such configuration the control design would need a hardware/software update in order to remove the emulated system in an industrial context. This follows by extra validation tests before a real industrial production of the board.

The paper is organized into three main sections: section two describes the controller, while the emulated system is detailed in section III. Finally, a control interface is discussed in section IV.

II. Controller

A. Introduction

The controller is implemented on an INTEL FPGA DE1_SoC board and constitute the system under test. It is based on 4 main blocks (see Fig. 1):

- A NIOS II microcontroller composed of 3 Interrupt Service Routines (ISR) based on 3 Timers with Vectored Interrupt Controller (VIC),

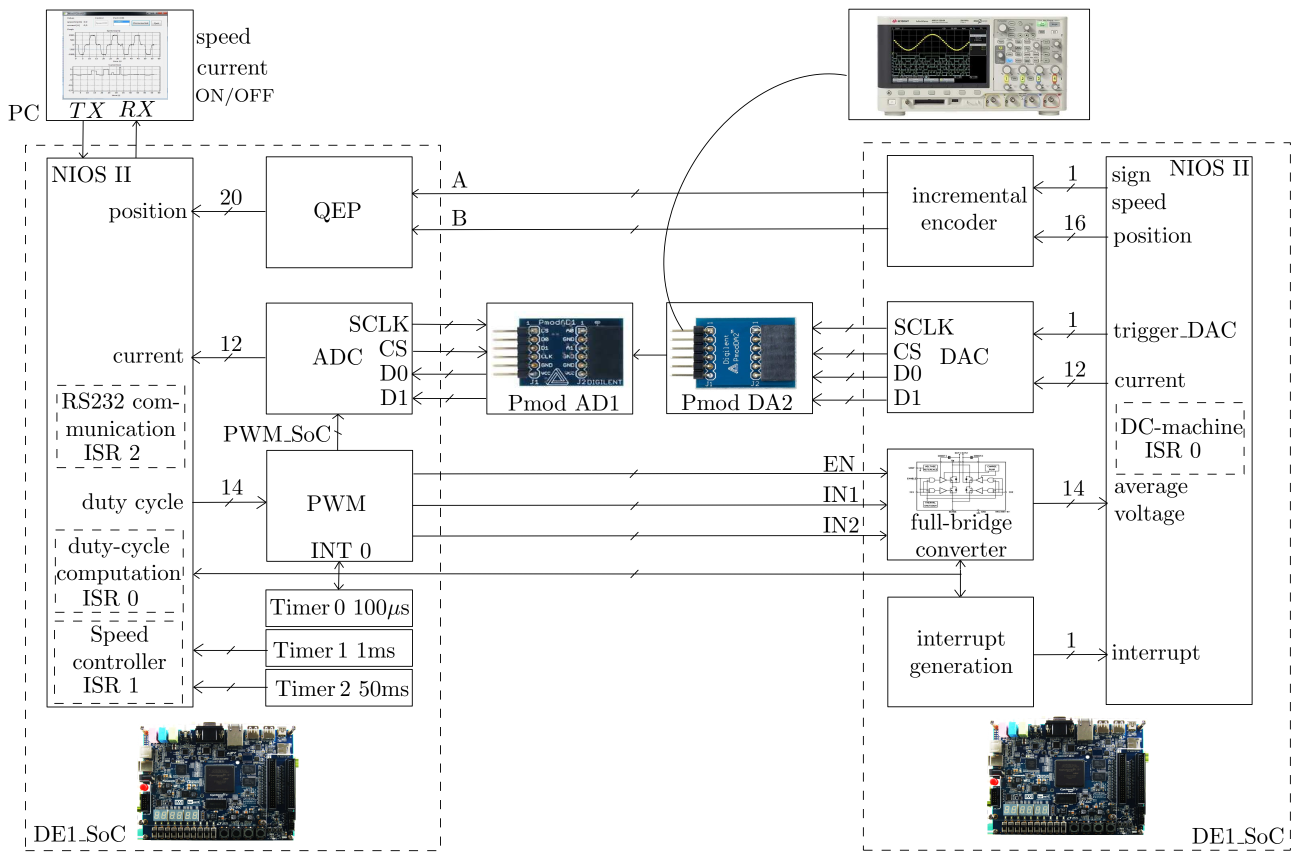


Figure 1. Controller and emulated DC-machine.

- A Quadrature Encoder Pulse (QEP) VHDL block in order to decode the two signals coming from the position sensor,
- A ADC converter to capture the current value,
- A Pulse Width Modulation (PWM) VHDL block in order to generate the command signals of the full-bridge converter that is synchronized with one ISR.

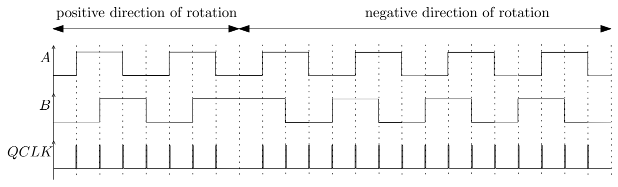


Figure 2. Quadrature-clock decoding.

B. Quadrature Encoder Pulse

The QEP module is composed of a first module that detects both edges of the A and B signals of an incremental encoder and a second module that count/countdown at each transition of the A/B signals. Fig. 2 shows the clock generation from the QEP input signals, where the frequency of the clock QCLK generated by the QEP logic is four times that of each input sequence. The QCLK signal is produced based on 3 flip-flop circuits with XOR modules. In order to exclude random glitch, the QCLK signal is produced base on the output of the first flip-flop circuit.

At each rising edge of QCLK, the state machine represented in Fig. (3) determines the direction of the counter (up or down) according to the level of A and B signals, and thus increment or decrement the counter that is an image of the relative rotor position.

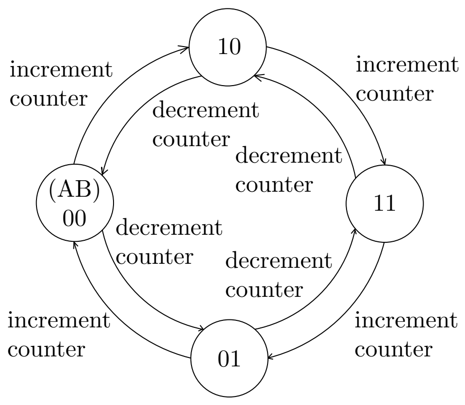


Figure 3. State machine of the Quadrature Encoder Pulse decoding.

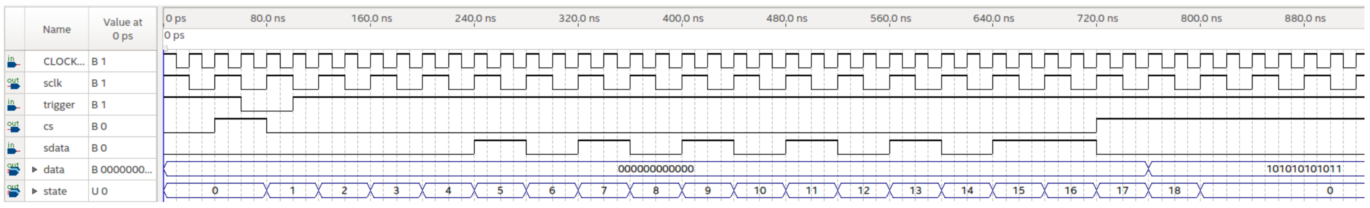


Figure 4. ADC timing.

C. Analog to digital converter

A Digilent Pmod AD1 module has been used [12]. It includes two channels, 12-bit analog-to-digital converters that features Analog Devices AD7476A with simultaneous A/D conversion at up to 1 million samples per second.

The Pmod AD1 module communicates with the main board via an SPI-like communication protocol. An acquisition start by pulling the CS low followed by four zeroes and twelve bits transmitted data, with the MSB first as shown in Fig. 4. Here, the ADC clock at been fixed at 25MHz that leads to a conversion time equal to 720ns (16 ADC clock cycles plus 2 extra clocks in order to respect the minimum quiet time and start of next conversion [13]).

The ADC conversion is triggered each 50μs using the PWM_SoC (PWM Start of Conversion) signal that it generated at the middle of the PWM signal.

D. Speed and current controllers

As an application example, a speed controlled DC machine with current control has been implemented in a NIOS II. The encoder count (position) is read once during each unit time event fixed at $T_{sw} = 1$ms base on Timer 1. Based on the measurement of the rotor position, the "measured" speed is deduced from an Euler backward differentiation:

$$\omega[k] = \frac{\theta[k] - \theta[k-10]}{10\,T_{sw}} \tag{1}$$

where a time base equal to $10\,T_{sw}$ have been opt in order to introduce a filtering of the estimated speed.

The reference current i^* is computed by a Proportional-Integral (IP) controller represented in Fig. 5.a and discretized afterwards by the Euler Backward approximation. Finally, in order to prevent from saturation of the integral term, an anti-windup scheme has been added as shown in Fig. 5.b.

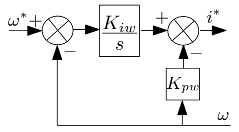
Finally, IP gains K_{pw} and K_{iw} are computed so that the closed-loop transfer function is analog to a second order transfer function, as follows:

$$\frac{\omega}{\omega^*} = \frac{1}{1 + \frac{K_{pw}}{K_{iw}}p + \frac{1}{K_{iw}}p^2} = \frac{1}{1 + \frac{2\xi_w}{\omega_{nw}}p + \frac{1}{\omega_{nw}^2}p^2} \tag{2}$$

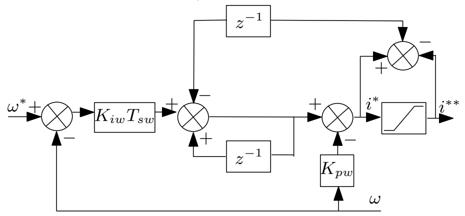
This leads to :

$$\begin{aligned} K_{iw} &= \tfrac{J}{K}\omega_{nw}^2 \\ K_{pw} &= \tfrac{2\xi_w\,J\,\omega_{nw} - f}{K} \end{aligned} \tag{3}$$

where ξ_w and ω_{nw} have been fixed at 1 and $4.8\,$rad/s respectively (response time equal to $1\,$s).



a)



b)

Figure 5. IP controller scheme.

The same procedure is adopted for the current controller design with a compensation of the back electromotive force. This leads to :

$$\begin{aligned} K_{ii} &= L\omega_{ni}^2 \\ K_{pi} &= 2\xi_i J\omega_{ni} - R \end{aligned} \tag{4}$$

where ξ_i and ω_{ni} have been fixed at $1/\sqrt{2}$ and $1500\,$rad/s respectively (response time equal to $2\,$ms). Also, the sample time has been fix at 50μs base on Timer 0 and synchronized with the PWM circuit.

A bi-polar command of the full-bridge converter has been opt. Therefore, the duty-cycle α is computed as follows:

$$\alpha[k] = \frac{1}{2}\left(\frac{u^{**}[k]}{U_{dc}} + 1\right) \tag{5}$$

where U_{dc} is the input voltage of the DC/DC converter (set at 200V).

E. Pulse Width Modulation

The PWM module retrieve the duty-cycle α that has been converted into an unsigned integer number (32 bits), i.e. multiply by $(N+1)T_b$ where the base period of the board is equal to 20ns (50MHz) and N equal to 2499 so that the PWM period is set to 50μs. The duty-cycle is compared to a synchronous counter in order to compute command signals IN1 and IN2 (IN2=/IN1 when EN=1 with the bi-polar command) of the full-bridge converter.

Timer 0 of the DE1_SoC board is based on a down binary counter with an optional periodic pulse generator feature (setting in Qsys) that outputs a pulse when timer reaches zero.

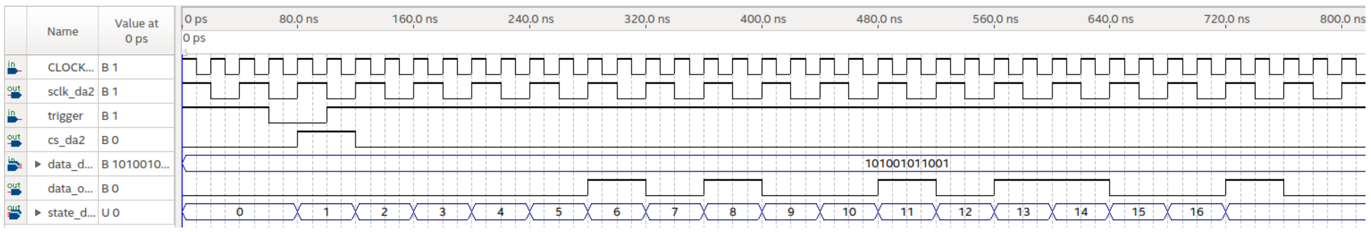


Figure 6. DAC timing.

It follows that this optional pulse-generator output is asserted for one clock period to trigger the interrupt service routine (ISR0) and reset the PWM counter, thus synchronising the PWM module and ISR0.

As noticed before, the PWM module has an additional output PWM_SoC (PWM Start of Conversion) in order to synchronize the PWM signal with the ADC conversion where the triggering signal is configurable by software.

III. Emulated system

A. Introduction

In practice, a drive is composed of three main components: the electrical machine, the converter and the sensors. In our application example, the DC-machine is speed controlled with a control of the armature current in the inner loop. It follows that an incremental position encoder and a current sensor are emulated. Finally, the machine is supply by a full-bridge converter (using hard switching) in order to work in the entire torque/speed plane.

In order that the design is friendlier, the DC-machine is represented by an average state space model that allows the implementation of the discrete state space model on a NIOS II processor [14]. Otherwise, the converter and position sensor need an accurate computation and are thus implemented with glue-logic into the FPGA. Fig. 1 shows the emulated system and provides all the input/output signals and internal signals exchanged with the INTEL NIOS II processor.

B. Full-bridge converter

Due to the fact that the machine is represented by an average model, the full-bridge converter block analyse the two PWM signals and PWM enable signal in order to retrieve the desired voltage applied to the machine. Here, dead-time and voltage drop in the converter have not been considered. The converter is therefore supposed as ideal. In such context, the armature current ripple due to the PWM signals is not modelled. We can refer to [15]–[18] in order to improve the converter modelization.

The VHDL block is synchronizing with the Timer0 interrupt coming from the controller board (i.e. DE1_SoC). At each time that Timer0 set the optional flag to one, the computed value of the duty cycle of the PWM is updated and a new computation of the PWM signals is launch.

C. Incremental encoder

The incremental encoder block generates two quadrature signals A and B based on the computed speed in order to emulate a quadrature encoder. Here, each signal generates 500 pulses per round.

D. Digital to analog converter

A Digilent Pmod DA2 module has been used [19]. It includes two channels, 12-bit digital-to-analog converters that features Texas Instruments DAC121S101 with simultaneous D/A conversion at a clock rates up to 30 MHz [20].

The Pmod DA2 module communicates with the main board via an SPI-like communication protocol. By pulling the CS signal to high (it can be set to high all along the conversion), the DAC driver send four zeroes and then twelve bits transmitted data, with the MSB first as shown in Fig. 6. Here, the DAC clock at been fixed at 25MHz that leads to a conversion time equal to 680ns (16 DAC clock cycles).

The DAC conversion is triggered each 10μs by the trigger_dac signal each time that the state vector of the DC machine is updated.

E. DC-machine

The permanent magnet DC-motor is represented by the continuous state space model:

$$\frac{d}{dt}x = Ax + B_1 u + B_2 t_l \tag{6}$$

with state space $x = [i \quad \omega]^t$ and

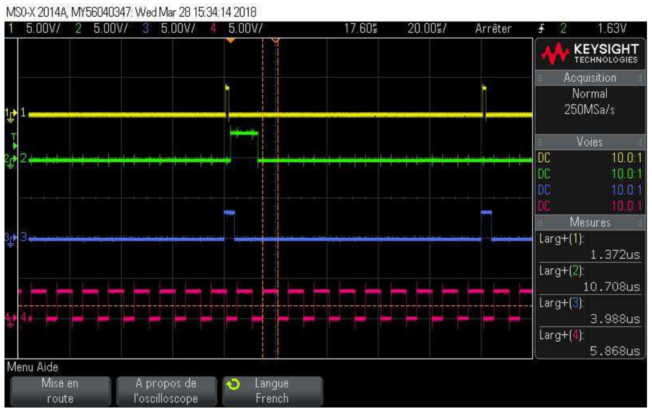
$$A = \begin{bmatrix} -R/L & -K/L \\ K/J & -f/J \end{bmatrix} \tag{7}$$

$$B_1 = \begin{bmatrix} 1/L \\ 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 \\ -1/J \end{bmatrix} \tag{8}$$

where i, ω, u and t_l are respectively the armature current, the rotor velocity, the armature voltage and load torque. R, L, J, f, K are respectively the resistance, inductance, inertia, viscous coefficient and back e.m.f. constant.

The continuous-time equations of the DC-machine are discretized with a sample time equal to the PWM period (i.e. $T_s = 100\mu$s) or at $T_s = 10\mu$s as follows:

$$x[k] = A_d x[k-1] + B_{1d} u[k-1] + B_{2d} t_l[k-1] \tag{9}$$


MS0-X 2014A, MY56040347: Wed Mar 28 15:34:14 2018

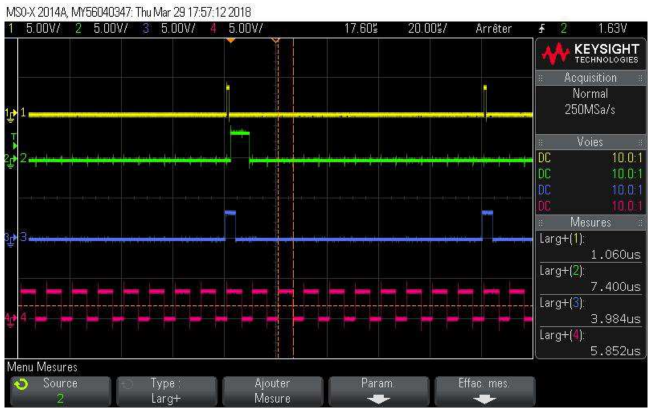
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 17.60$ 20.00$/ Arrêter 2 1.63V

Acquisition Normal 250MSa/s

Voies: DC 10.0:1, DC 10.0:1, DC 10.0:1, DC 10.0:1

Mesures
Larg+(1): 1.372us
Larg+(2): 10.708us
Larg+(3): 3.988us
Larg+(4): 5.868us

Menu Aide


MS0-X 2014A, MY56040347: Thu Mar 29 17:57:12 2018

1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 17.60$ 20.00$/ Arrêter 2 1.63V

Acquisition Normal 250MSa/s

Voies: DC 10.0:1, DC 10.0:1, DC 10.0:1, DC 10.0:1

Mesures
Larg+(1): 1.060us
Larg+(2): 7.400us
Larg+(3): 3.984us
Larg+(4): 5.852us

Menu Mesures

Figure 7. Channel 1: time duration of ISR0 of the controller board, Channel 2: time duration of ISR1 of the controller board, Channel 3: signal coming from Timer0 to synchronize the boards, Channel 4: time duration of ISR0 of the emulator board.

Figure 8. Channel 1: time duration of ISR0 of the controller board, Channel 2: time duration of ISR1 of the controller board, Channel 3: signal coming from Timer0 to synchronize the boards, Channel 4: time duration of ISR0 of the emulator board.

with

$$A_d = e^{AT_s} \tag{10}$$
$$B_{1d} = A^{-1}(A_d - I)B_1 \tag{11}$$
$$B_{2d} = A^{-1}(A_d - I)B_2 \tag{12}$$

and executed at each sample time in the NIOS II microcontroller ISR0.

F. Interrupt generation block

The simplest way to trigger the ISR0 consist in the implementation of a timer with a sample time equal to the PWM period, i.e. $T_s = 100\mu$s in this application example. However such solution leads to two drawbacks:

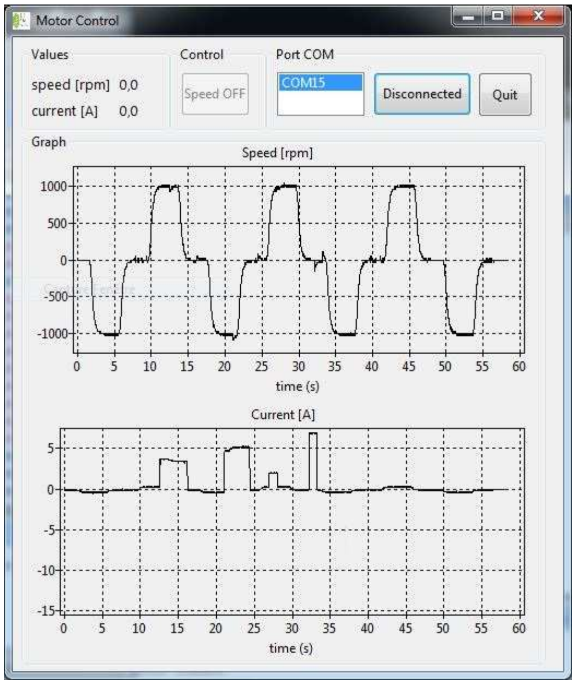
- The resolution of the discrete time state space model is not accurate because we have nearly ten computations during an electrical transient.
- The ISR0 is not synchronized with the controller board due to clock drift between the two boards (see Fig. 7-Channel 4).

One solution consist in the implementation of a own interrupt generation controller with a sample time set at $T_s = 10\mu$s (just greater that the time duration of the ISR0, i.e. 5.86μs) and re-synchronized each 100μs by using the signal coming from the controller board as shown in Fig. 8.

IV. Man-machine interface

In order to receive and send data to the controller, one RS-232 serial port between the PC and INTEL FPGA controller board has been implemented. The computed data as current and speed are transferred to a personal computer and display on an own interface as shown in Fig. (9). The interface has been design with Lazarus, a cross-platform IDE for Rapid Application Development [21]. Otherwise, the load torque is defined based on the FPGA switch position.

All the data are transferred with a sample time set at 50ms base on Timer2 of the controller board. Also base on the


Motor Control

Values
speed [rpm] 0,0
current [A] 0,0

Control
Port COM: COM15

Figure 9. Man machine interface for control and validation.

DAC converter, it is possible to capture the current and speed transients as shown in Fig. 10.

V. Conclusion

This article has detail the models and control of a DC machine for a low cost Hardware-In-The-Loop platform with INTEL FPGA boards for educational and research purpose. The emulated power converter, machine, position encoder and current sensor are implemented on a DE1_SoC board while the controller and peripherals are implemented on a second DE1_SoC board. Finally, all the computed data are transferred to a personal computer and display on a man-machine interface (IHM) for validation of the controller performances.

The perspectives of this work are numerous.

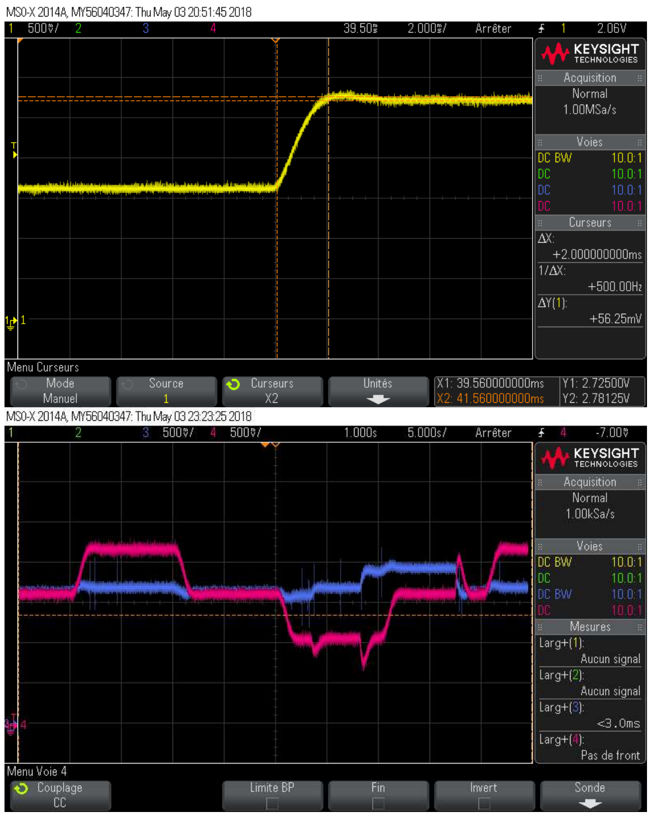


Figure 10. Current and speed response time.

- Firstly, the whole system needs development such as the design of an instantaneous model of the converter in order to capture the current ripples [6], [16]–[18], [22]–[25]. In this context, sample times nearly 100ns are mandatory so that to capture the PWM signals transition and compute precisely the current. It follows that such development needs fixed or floating point VHDL code.
- Secondly, SoC platform combine dual or quad-core ARM Cortex A9 or A53 MPCore hard processor system that offers significant computing power for bare-metal programming. Comparison between NIOS and ARM processors could be done.
- Thirdly, this system could be adapted to other electrical machines. In order to introduce the control of renewable energy system such as photovoltaic panel, it could be also interesting to adapt such work with a MPPT (Maximum Power Point Tracking) control.

References

[1] P. Fajri, V. Prabhala, and M. Ferdowsi, "Emulating on-road operating conditions for electric-drive propulsion systems," *IEEE Transactions on Energy Conversion*, vol. PP (99), 2015.

[2] H. Zhang, Y. Zhang, and C. Yin, "Hardware-in-the-loop simulation of robust mode transition control for a series-parallel hybrid electric vehicle," *IEEE Transactions on Vehicular Technology*, vol. PP (99), 2015.

[3] B. Lu, X. Wu, H. Figueroa, and A. M. ., "A low-cost real-time hardware-in-the-loop testing approach of power electronics controls," *IEEE Transactions on Industrial Electronics*, vol. 54 (2), pp. 919–931, 2007.

[4] C. Dufour, H. Blanchette, and J. Belanger, "Very-high speed control of an fpga-based finite-element-analysis permanent magnet synchronous virtual motor drive system," *34th Annual Conference of IEEE Industrial Electronics Society (IECON)*, pp. 2411–2416, 2008.

[5] J. Mahseredjian, V. Dinavahi, and J. Martinez, "Simulation tools for electromagnetic transients in power systems: Overview and challenges," *IEEE Transactions on Power Delivery*, vol. 24 (3), pp. 1657–1669, 2009.

[6] T. Ould-Bachir, C. Dufour, J. Belanger, J. Mahseredjian, and J. David, "Effective floating-point calculation engines intended for the fpga-based hil simulation," *IEEE International Symposium on Industrial Electronics (ISIE)*, pp. 1363 – 1368, 2012.

[7] Y. Chen and V. Dinavahi, "An iterative real-time nonlinear electromagnetic transient solver on fpga," *IEEE Transactions on Industrial Electronics*, vol. 58 (6), pp. 2547 – 2555, 2011.

[8] N. R. Tavana and V. Dinavahi, "A general framework for fpga-based real-time emulation of electrical machines for hil applications," *EEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, vol. 62(4), pp. 2041–2053, 2015.

[9] T. Liang and V. Dinavahi, "Real-time device-level simulation of mmc-based mvdc traction power system on mpsoc," *IEEE Transactions on Transportation Electrification*, 2018.

[10] D. Tormo, L. Idkhajine, E. Monmasson, and R. Blasco-Gimenez, "Evaluation of soc-based embedded real-time simulators for electromechanical systems," *42nd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 4772–4777, 2016.

[11] D. Tormo, R. Vidal-Albalate, L. Idkhajine, E. Monmasson, and R. Blasco-Gimenez, "Study of system-on-chip devices to implement embedded real-time simulators of modular multi-level converters using high-level synthesis tools," *IEEE International Conference on Industrial Technology (ICIT)*, pp. 1447–1452, 2018.

[12] *https://store.digilentinc.com/pmod-ad1-two-12-bit-a-d-inputs/.*

[13] *Analog Devices, AD7476A/AD7477A/AD7478A Data sheet, http://www.analog.com/media/cn/technical-documentation/evaluation-documentation/AD7476A_7477A_7478A.pdf.*

[14] K. Lian and P. Lehn, "Real-time simulation of voltage source converters based on time average method," *IEEE Transactions on Power Systems*, vol. 20 (1), pp. 110–118, 2005.

[15] P. Pejovic and D. Maksimovic, "A method for fast time-domain simulation of networks with switchesa method for fast time-domain simulation of networks with switches," *IEEE Transactions on Power Electronics*, vol. 9(4), pp. 449 – 456, 1994.

[16] H. F. Blanchette, T. Ould-Bachir, and J. David, "A state-space modeling approach for the fpga-based real-time simulation of high switching frequency power converters," *IEEE Transactions on Industrial Electronics*, vol. 59(12), pp. 4555 – 4567, 2012.

[17] M. Dagbagi, L. Idkhajine, E. Monmasson, and I. Slama-Belkhodja, "Fpga implementation of power electronic converter real-time model," *International Symposium On Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pp. 658 – 663, 2012.

[18] M. Dagbagi, A. Hemdani, L. Idkhajine, M. Naouar, E. Monmasson, and I. Slama-Belkhodja, "Fpga-based real-time hardware-in-the-loop validation of a 3-phase pwm rectifier controller," *39th Annual Conference of the IEEE Industrial Electronics Society, IECON*, pp. 5374 – 5379, 2013.

[19] "https://store.digilentinc.com/pmod-da2-two-12-bit-d-a-outputs/."

[20] *Texas Instruments DAC121S101 Data sheet, http://www.ti.com/lit/ds/symlink/dac121s101.pdf.*

[21] https://www.lazarus ide.org/.

[22] S. Hui and S. Morrall, "Generalised associated discrete circuit model for switching devices," *IEE Proceedings Science, Measurement and Technology*, vol. 141(1), pp. 57 – 64, 1994.

[23] M. Dagbagi, A. Hemdani, L. Idkhajine, W. Naouar, E. Monmasson, and I. S. Belkhodja, "Adc-based embedded real-time simulator of a power converter implemented in a low cost fpga: Application to a fault-tolerant control of a grid-connected voltage source rectifier," *IEEE Transactions on Industrial Electronics*, vol. 63(2), pp. 1179–1190, 2016.

[24] M. R. Larijani, M. Zolghadri, and M. Shahbazi, "Design and implementation of an fpga-based real-time simulator for h-bridge converter," *7 th Power Electronics, Drive Systems & Technologies Conference (PEDSTC 2016)*, 2016.

[25] M. Rezayati and M. R. Zolghadri, "Optimal down sampling for adc-based real-time simulation of basic power electronic converters," *8th Power Electronics, Drive systems Technologies Conference (PEDSTC)*, 2017.