A Hardware Software In the Loop architecture for control units

Mossaad Ben Ayed
National Engineering School of Sfax
University of Sfax, Tunisia
mossaad_benayed@yahoo.fr

Faouzi Bouchhima
National Engineering School of Sfax
University of Sfax, Tunisia
f_bouchhima@yahoo.fr

Mohamed Abid
National Engineering School of Sfax
University of Sfax, Tunisia
mohamed.abid@enis.rnu.tn

Abstract—**Design of continuous/discrete systems is based on tools coming from mathematical model of the plant. The industrial tool Matlab/Simulink is widely used in such systems. The main advantage of this tool is its ability to model in a common formalism the software and its physical environment. Unfortunately, Matlab/Simulink still suffers from many limits in modeling and verification. Due to the multidisciplinary nature of advanced systems and to overcome these limits in modeling and verification, several tools based on combined language are adopted. This paper describes Hardware Software In the Loop technique for Control Units. A synchronization model between Matlab/Simulink and a real board is presented.**

Keywords- HIL; Co-design; Simulator; Emulator; Verification; Simulink.

I. INTRODUCTION

As the number and the complexity of mechatronics components increases, tools and technologies for developing and verification of the Control Units (CU) are required. Simulink presents the widely tool used for continuous/discrete systems and it is a good target for design and verification on the earlier stage of the design. Matlab/Simulink is not only used in all the steps of the cycle of development but also played a crucial role in the numerical simulation of CU. The cycle development based on Matlab/Simulink, especially in automotive industry, can be divided on three steps.

First, the Model In the Loop (MIL) [1] refers to the kind of testing performed to verify the expected performance and robustness of a control algorithm in model form in a closed loop environment. This step concerns the definition of a mathematical model of the plant and the control law. This model is validated using a numerical simulation.

Then, the Software In the Loop (SIL) [2] step concerns the implementation of the control algorithm in a low language such as C.

Finally, the Hardware In the Loop (HIL) [3] step concerns the compilation of the controller implementation into an executable running on a particular hardware.

HIL involves connecting the actual CU to the real time simulation models, in which the CU in hardware is integrated with virtual models of the devices and systems being controlled.

The goal of this paper is to investigate the needs and the possibilities concerning a combined usage of Matlab/Simulink and a real architecture based on co-design development, implemented in the board. A Hardware Software In the Loop is announced in this paper.

Section 2 presents the related works for design and verification method in Simulink. Section 3 presents the conventional approach and the synchronization scheme of the Hardware Software In the Loop technique. Section 4 describes the different steps for implementation and the experimental results. Section 5 concludes the paper with a discussion.

II. RELATED WORK

There are mainly tow ways in literature to combine Simulink with another environment.

A. Integration

There have been several studies regarding the integration of different environments and enabling different modeling frameworks to interact with each other. [4] is made to integrate SystemC in Matlab/Simulink environment using the S-function bloc. Support the different abstraction layers for embedded systems in Simulink environment is the aim of the last work.

The work [5] integrates the Processor Expert tool in Simulink to use different kinds of microcontroller. The integrated environment has to follow the Simulink solver. This fact is the major problem. Indeed, the discrete simulation progresses with the respect of the integration step imposed by the solver in the continuous simulation.

B. Co-simulation

The co-simulation methodology is based on different simulation tools running simultaneously and exchanging information in a collaborative manner for verification reasons. We can cite especially Simulink/Modelsim, Simulink/SystemC and HIL.

Simulink/Modelsim is adopted when the Hardware Description Language (HDL) is used to describe the behavior of the control algorithms. HDL Verifier automates verification by using Simulink to stimulate the HDL code and analyze its response [3].

Simulink/SystemC is adopted when the hardware description is modelled using SystemC. As we know, SystemC is a standardized modelling language intended to enable system level design and intellectual property integration at multiple abstraction layers, for systems containing both software and hardware components. As an example, CODIS (COntinuous DIscrete Simulation) [6] is

a tool which can automatically produces co-simulation instances for continuous/discrete systems simulation using SystemC and Simulink simulators. This is done by generating and providing co-simulation interfaces and the co-simulation bus. To evaluate the performances of simulation models generated in CODIS, they measured the overhead given by the simulation interfaces [7].

Hardware In the Loop:

The more traditional application of the HIL concept is controller design and testing, in which a CU in hardware is integrated with virtual models of the devices and systems being controlled. Most famous techniques of HIL are MathWorks' solution xPC Target [8] [9] and Real-Time Windows Target [10], where the model is executed on a dedicated system or on a windows system, respectively. The last consists of synchronizing the clocks of the virtual subsystems with the clocks of the real subsystems and achieving determinism in the overall system.

The HIL is adopted in verification and testing for many advantages:

- Control and regulation functions can be tested in early stages of development, even before a test carrier.

- Typical test drives under low conditions (ambient, snow, ice) can be performed repeatedly.

- Failures and errors that could have devastating effects in a real system can be simulated and tested systematically.

- The experiments performed in the HIL system can be reproduced precisely, and automatically repeated as often as required.

However, the modeling language provided and the different advantages of the HIL, these solutions have not been designed for hardware/software co-design purposes. There are several weaknesses that motivate us to develop a new standard target based on FPGA board [11].

- Only few targets exist and therefore far from all CU families and derivates are supported.

- Each CU target has its own block set. This fact prevents the reusability and the portability of the model using these HW specific blocks.

- The way in which the peripheral HW is handled by the generated code is predefined by the target developers and it can not be changed by the user.

- Wiring harness HIL simulation platform needs to be redone each time the hardware interface of a CU changes.

The next section presents the Hardware Software In the Loop (HSIL) proposed.

III. CONVENTIONAL APPROACH

Control embedded systems are mostly heterogeneous devices. Their design is based on hardware and software components. Each part needs to be aware of the characteristics of other parts, in order to provide optimized components. The best strategy adopted is co-design, since it allows us to develop HW/SW components concurrently. The main idea is to set the HSIL to overcome the limits of the HIL.

Our method described in the next section, improves the HIL, and has many advantages such:

➢ Expand the HIL to attend the Co-design strategy.

➢ Use of one and same S-function for every CU. This fact lets the reusability and the portability of the model.

➢ Able to verify multiple CU in the same system and to modify one without modify the architecture.

Figure 1 describes the global idea of HSIL.
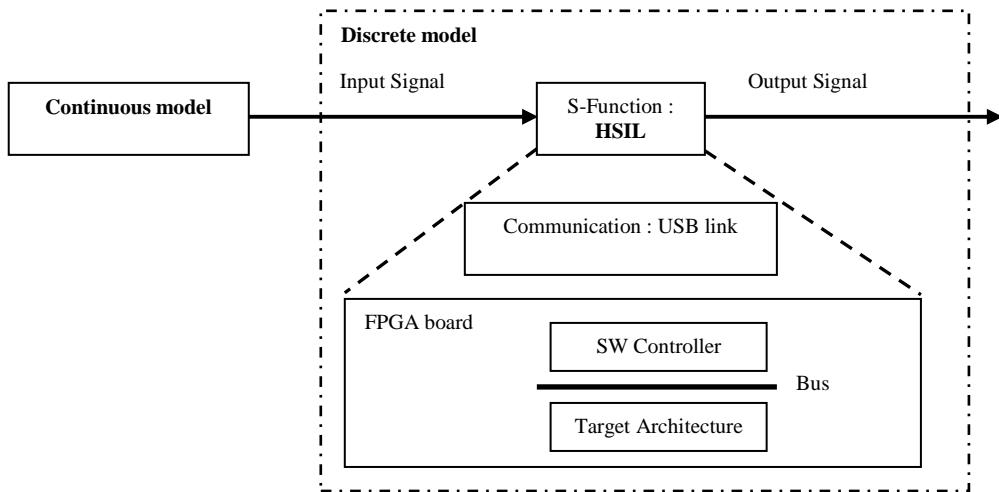
Figure 1. Hardware Software In the Loop architecture

Two issues are essential for the HSIL architecture: the communication and the synchronization models.

A. Communication model

This section gives a brief introduction to the communication model. A USB link is used in the communication between Simulink and FPGA because this kind of communication has better speed than PCI which it adopted in emulation [12].

This communication is based on packets which are constructed by the communication interface using S-Function between Simulink and board. An S-Function is a computer language description of a Simulink block. It uses syntax of call allowing us to interact with Simulink solvers.

A Data packet, figure 2, is used to perform the synchronization scheme between the Simulink simulator and the emulator.

| Routine Number (RN)	Size (S)	Data (D)	Time stamp (TS)

Header   Body
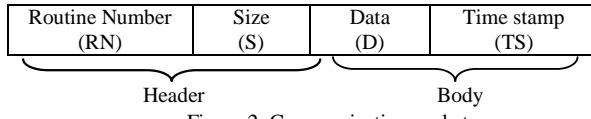
Figure 2: Communication packet

Data packet comprises a header and the Body. The header contains the routine number and the data size. The routine number corresponds to the CU that will be executed in the target architecture. The body is composed with data and the time stamp to synchronize when it is necessary.

Note that any packet received by the board side generates an USB interruption that can be exploit in the implementation phase to interrupt the target processor each time a packet is received.

B. Synchronization model

Because the heterogeneity of systems, analog-digital (AD) and digital-analog (DA) converters are used. These converters are integrated in synchronization bloc (S-function), see figure 3.
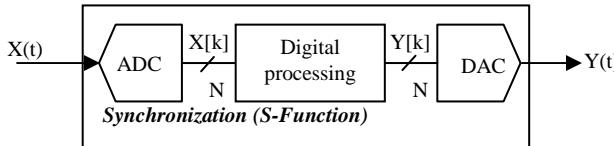

Figure 3: Synchronization bloc

The ADC is used to transform the analog input to digital signal based on equation (1).

$$X[k] = k \frac{X(t)}{2^N} \qquad (1)$$

Where:

X(t) = Analog input

X[k] = Digital output code

N = Number of digital input bits (resolution)

$k \in \{1,.....,N\}$

The DAC is used to transform the digital signal after the processing step to analog output based on the transfer function shown in (2).

$$Y(t) = \frac{Y[k]}{2^N} X(t) \qquad (2)$$

Where:

Y(t) = Analog output

Y[k] = Digital input code

N = Number of digital input bits (resolution)

X(t) = Reference Value (full-scale)

A key issue of the proposed approach is the time synchronization between the Simulink simulator and the processor emulated on the FPGA board. The verification method is based on the following synchronization scheme which respects the interaction style that can be involved between continuous and discrete model, figure 4.
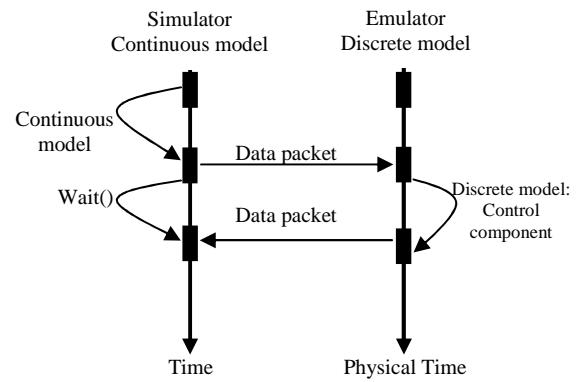

Figure 4: Synchronization scheme for HSIL

> **The Continuous model waits the end of the hardware/software task.**

When a hardware/software components is emulated by the board, the continuous model uses a waiting loop for data (see figure 4). Once controller task is finished, the emulator sends data to the simulator and a switch context from the board to Simulink simulator is taken. At this time, the continuous model receives data and resumes the execution. Note that the Simulink and the emulator need to usually exchange information about the time.

The next section drafts the implementation and her details.

IV. EXPERIMENTAL RESULTS

The application example, figure 5, consists in a DC Motor Speed Modeling in Simulink with a PID regulator. The following equation (3) in open-loop transfer function is

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2} \qquad \left[\frac{rad/\sec}{V}\right] \quad (3)$$

Where:

- The rotational speed is considered the output and the armature voltage is considered the input.
- (J) moment of inertia of the rotor
- (b) motor viscous friction constant
- (Ke) electromotive force constant

- (Kt) motor torque constant
- (R) electric resistance
- (L) electric inductance

Then, a PID regulator is added to control the DC motor speed. Figure 6 shows the diagram of the global system.
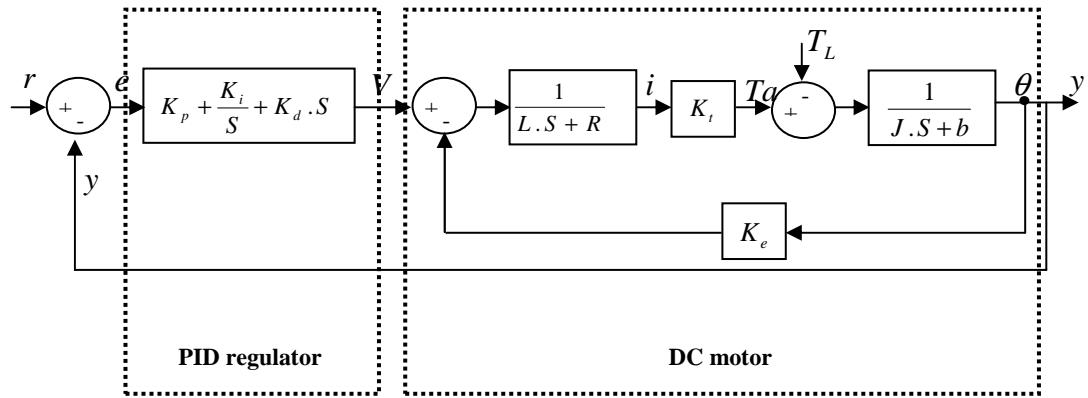

Figure 6: Diagram of DC motor speed regulator
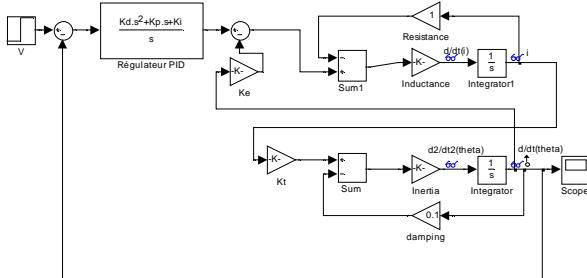
Figure 5: DC motor speed regulator

The aim as mentioned bellow is to verify the controller units. In this case, the model has two integrators and one PID that will be considered as SW applications. Three steps are essential for the implementation.

Step 1: Target architecture

The verification idea is based on combined tools to satisfy continuous and discrete models. For the discrete model an FPGA type ALTERA DE2-70 is used as a board and QuartusII, NIOSII IDE as tools. The first step is to set the architecture model. Figure 7 shows the architecture chosen. It contains the NIOSII processor [13], Avalon bus, memory and the ISP1362 USB controller [14].

The integrator and the PID controller are described as SW applications that will be executed in the last architecture.

Step 2: Integration of synchronization bloc

A C++ S-function bloc is used to implement the synchronization scheme between Simulink simulator and the target architecture in the FPGA board.
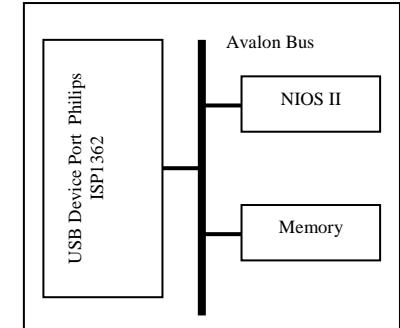

Figure 7: Architecture target

The Synchronization bloc is an interface that creates break points which must be reached accurately by a solver. These points are the time stamps of the input signal from continuous model. When a signal is received, this interface blocks the solver and makes a switch context to the board. At first, the last activates the USB link with interruption mode and sends the data packet. Then a wait() function is executed until the reception of data from the board. After resuming execution the interface Synchronization sets the next activation time.

Step 3 : Simulation results

The simulation was performed using the synchronization scheme, since the continuous model generates state events and the signals update events are not periodic.

At first, the new DC motor speed scheme, figure 8, is described using the following parameter: J=0.01; b=0.1; K=0.01; R=1; L=0.5.
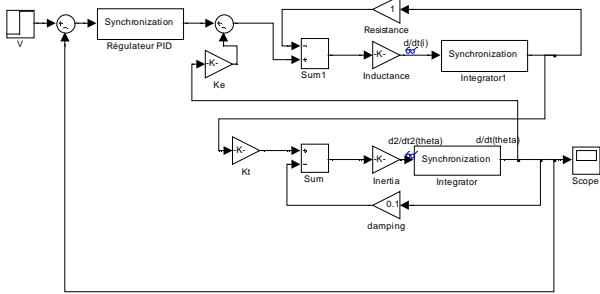
Figure 8: New DC motor speed model

Figure 9 shows the whole operation between Simulink and the NIOSII IDE environment.
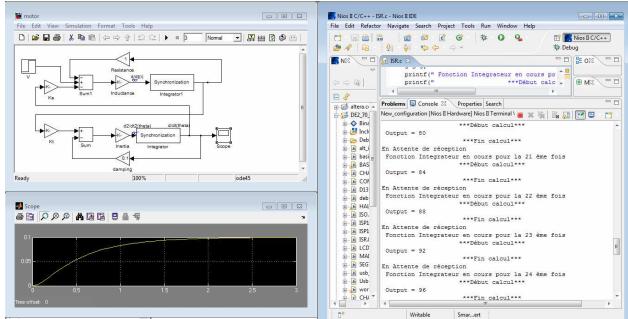
Figure 9: Simulink/NIOSII simulation/emulation

To verify the efficiency of the synchronization model, three figures issued by each synchronization bloc are used as shown in figure 10.

For this sample the Simulink simulator performs 24 switch context to the board. The simulation time is 1 second calculated by the simulator.
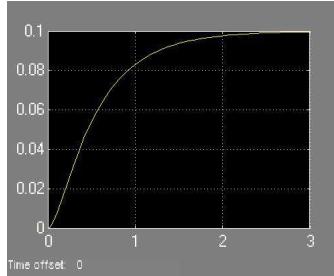
a) Signal of motor speed
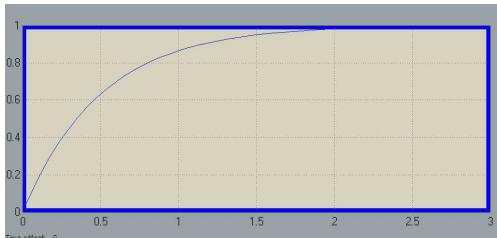
b) Signal of current i

c) Output signal of the regulator

Figure 10: critical signal

V. CONCLUSION

This paper proposed a Hardware Software In the Loop technique using a simulation / emulation framework for continuous/discrete systems, based on generic interface. The simulation / emulation integrated currently in this framework is based on FPGA board (for the discrete parts) and Simulink (for the continuous parts).

The HSIL architecture improves not only the HW/SW design in CU but also ensure the reusability and the portability of CU. To evaluate the proposed approach, simulation / emulation results for a discrete/continuous application were illustrated.

REFERENCES

[1] Melih Çakmakc, Yonghua Li and Shuzhen Liu."Model-in-the-Loop Development for Fuel Cell Vehicle", 2011 American Control Conference on O'Farrell Street, San Francisco, CA, USA June 29 - July 01, 2011

[2] Vitalijs Osadcuks, Ainars Galins. "Software In the Loop simulation of autonomous hybrid power system of an agricultural facility", Engineering for rural development, Jelgava, 24.-25.05.2012. p 500-505

[3] M. Bacic, "On hardware-in-the-loop simulation," in 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05, 2005, pp. 3194–3198

[4] Walid Hassairi, Moncef Bousselmi, Mohamed Abid and Carlos Valderrama, "Matlab/SystemC for the New Co-Simulation Environment by JPEG Algorithm" , MATLAB – A Fundamental Tool for Scientific Computing and Engineering Applications – Volume 2, p 120-138.2012

[5] Roman Bartosinski, Zdeněk. Hanzálek, Petr Stružka, and Libor Waszniowski, "Integrated Environment for Embedded Control Systems Design", Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International

[6] F. Bouchhima, M. Brière, G. Nicolescu, M. Abid, E. M. Aboulhamid, "A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation", Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International

[7] Dogan Fennibay_y, Arda Yurdakuly and Alper Seny, "Introducing Hardware-in-Loop Concept to the Hardware/Software Co-design of Real-time Embedded Systems", The 7th IEEE International Conference on Embedded Software and Systems, ICESS 2010.

[8] MathWorks. (2013) xpc target. [Online]. Available: http://www.mathworks.com/products/xpctarget/

[9] Hosam K. Fathy, Zoran S. Filipi, Jonathan Hagena and Jeffrey L. Stein , "Review of Hardware-in-the-Loop Simulation and Its Prospects in the Automotive Area" , Modeling and Simulation for Military Applications, edited by Kevin Schum, Alex F. Sisti Proc. of SPIE Vol. 6228, 62280E, (2006).

[10] MathWorks. (2010) Real-time windows target. [Online]. Available: http://www.mathworks.com/products/rtwt/

[11] Roman Bartosinski, Zdeněk. Hanzálek, Petr Stružka, and Libor Waszniowski, "Integrated Environment for Embedded Control

Systems Design", Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, p 1-8

[12] Soha Hassoun, Murali Kudlugi, Duaine Pryor, and Charles Selvidge, "A Transaction-Based Unified Architecture for Simulation and Emulation", IEEE transactions on very large scale integration (VLSI) systems, vol. 13, no 2, 2005.

[13] NIOS II Processor Reference. (2013) [Online]. Available: http://www.altera.com

[14] ISP1362 Single-chip Universal Serial Bus On-The-Go controller. (2013) [Online]. Available: http://www.cs.columbia.edu