Implementing a Processor-in-the-loop with a Universal Controller in the Virtual Test Bed

R. Liu and A. Monti
Department of Electrical Engineering
University of South Carolina
Columbia, South Carolina 29208
[liur, monti]@engr.sc.edu

G. Francis, R. Burgos, F. Wang and D. Boroyevich
Center for Power Electronics Systems
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
[jerryf, Rolando]@vt.edu

***Abstract*-As digital processor technology has rapidly developed, researchers have proposed various methodologies to design, develop, implement, and test digital control systems. A testing procedure, called Processor-in-the-Loop (PIL), allows designers to verify the actual control software running in a dedicated processor which controls a virtual prototype of a plant. In this paper, we focus on the demonstration of how to implement PIL technology in the Virtual Test Bed (VTB) with a hybrid DSP/FPGA-based control platform; PIL system structure and simulation results are presented. Based on this example, we can extend the PIL procedure in VTB to other control hardware platforms easily.**

I. INTRODUCTION

In the past decades, digital processor technology has been rapidly developing in the past decades and has impacted many fields. Recently, more and more processors, such as Digital Signal Processors (DSP), have been used to implement large, fast and complex digital control algorithms. Traditional software-only simulation methodology cannot often capture the control dynamics sufficiently. Ways to bridge the gap between simulation and deployment can be Processor-in-the-loop (PIL) and Hardware-In-the-Loop (HIL) methodologies. HIL is a testing procedure in which the controller runs on the actual hardware and the plant is modeled with the use of a real time simulation environment. HIL can be used to increase the realism of the simulation; however there are some hurdles in HIL procedure. For example, it may be necessary to use simplified models in order to meet the real-time constraints. Furthermore, it is usually time-consuming to modify components, parameters and circuit topology. PIL can be considered as an intermediate stage between standard simulation and HIL [1-2].

As the main point of difference from HIL, in the case of PIL the plant runs on a non-real time simulator. As a consequence there is no need to meet the real time constraint. The controller, as in the HIL case, runs on the target hardware. PIL allows controller designers to test the embedded control system even using a standard PC. In addition, the designer can quickly and conveniently modify the plant.

In order to make this solution possible, the real-time operating system on the target hardware is substituted with a communication protocol able to trigger the control software only when requested by the simulation platform. On the other hand the simulation platform must be able to operate as an event-driven system performing continuous time simulation and event triggered control execution.

The main purpose of PIL is to bridge the gap between the control design performed in simulation software and the execution of the actual control codes in the target. By means of the PIL approach, the designer can find errors that might not be found by the compiler, and the functions and performance of controllers can be debugged and verified as well. Furthermore, the PIL procedure can provide important measurements on the software system such as memory usage, control algorithm execution time. These data help designers to implement and test digital control systems and their hardware at the beginning of the whole design process, thus limiting the use of error-prone procedures [1-3].

In this paper we use a DSP/FPGA digital control system, hereafter called Universal Controller (UC) [4], as an example to illustrate how to implement a PIL procedure in Virtual Test Bed (VTB). In this PIL procedure, the plant is simulated in VTB, and the controller codes are executed on the UC. The communication channel between the two subsystems is the PCI bus. Based on this example, we can easily implement the PIL on other control hardware platforms.

II. VIRTUAL TEST BED PLATFORM

In this research, VTB serves as the software platform for the PIL simulation demonstration. VTB is a software environment that has been developed for the design, analysis, and virtual prototyping of large-scale multi-technical systems [5].

VTB allows for handling natural power flow, signal and data coupling between inter-connected devices and offers a combination of both topological and mathematical expressions in model formulation for a comprehensive and efficient modeling process. VTB also supports multiple-layer modeling, wherein each layer can describe a different model complexity or a different behavior [1-2].

As a high-level virtual prototyping tool, the VTB program addresses many challenges in a field such as power electronics, which encompasses a wide range of disciplines including analog and digital electronics, power systems, controls, electro-mechanics, and mechanical systems. The software also

1-4244-0655-2/07/$20.00©2007 IEEE

provides methods for importing models from other simulation or modeling environments (such as Spice, MATLAB, ACSL, and LabView) for co-simulation and for straightforward, object-oriented declaration of the system topology [1-4].

III. PIL IN VTB

As mentioned before, in this research, the plant is modeled and run in VTB while the controller code runs on the target hardware. The data exchange between VTB and the target hardware is performed via communication channel. During the PIL simulation process, VTB simulates the plant model by using a time step defined by the user. When a time equivalent to a control sample interval elapses, VTB exports the output data of the plant to the control system via the communication channel. The target processor executes the controller code for one sample step after the processor receives the data from the plant model. Then, after calculation, the control system returns the calculated results (the output of the controller) to VTB via the same communication bus. At this point, one simulation sample cycle is completed, and the plant model proceeds to the next sample interval. The process repeats and the simulation progresses.

To achieve PIL implementation, we should consider several aspects: the communication channel, synchronization, and data exchange protocol between VTB and the target hardware.

1) Communication Channel: Standard PCs support many communication channels, such as serial link, Universal Serial Bus (USB), parallel link, PCI bus, PCI express bus, Ethernet, and so on. The design of a specific PIL application has to be customized to the preferred communication channel.

2) Synchronization: The PIL mode involves two time domains: the simulation time domain in VTB and the real time domain in the control hardware. The processor on the control hardware cannot predict whether the desired sampling time in VTB is reached; hence, a synchronization strategy should be considered.

3) Data Exchange Protocol: A customized data exchange protocol is used to ensure the correct data exchange between VTB and the hardware.

IV. AN EXAMPLE OF PIL IMPLEMENTATION IN VTB

In this example, a UC is used to illustrate how to implement a PIL procedure in VTB. The communication channel between VTB and the UC is a PCI bus. Fig. 1 shows the system structure.

This section addresses the most important issues, including communication channel, synchronization, data exchange protocols and model setup.

A. The UC and PCI Bus

The UC (Fig. 2) is designed to be suitable for the majority of power electronics applications and topologies in industrial and naval applications. The UC is a hybrid DSP/FPGA-based control platform and its architecture exploits sequential and parallel processing of FPGA and DSP technologies. Typically, the processing speed of the FGPA can be up to four to five times faster than the DSP. This makes FPGA an optimal solution to implement power electronics specific functions, such as analog-to-digital (ADC), digital-to-analog (DAC), and PWM. On the other hand, DSP is well suited for application level controls, e.g., voltage or current loop [6].

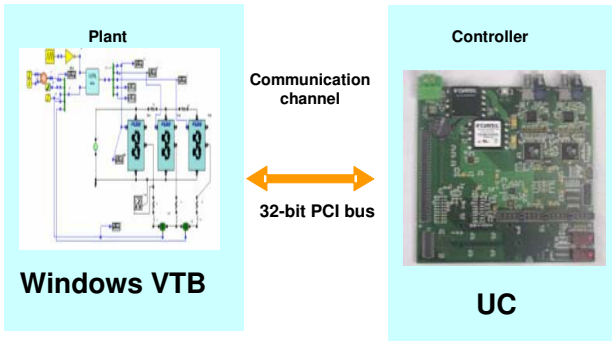

Figure 1. Structure of PIL simulation

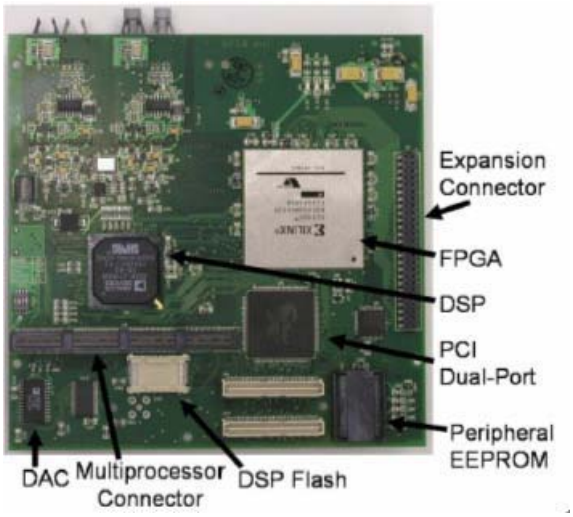

Figure 2. Universal Controller

The UC processor is a DSP (Analog Devices ADSP-21160). The UC supports a 32-bit PCI interface using a PCI interface chip, CY7C09449PV-AC provided by Cypress Semiconductor.

The PCI bus is a reliable, high performance 32-bit or 64-bit bus with multiplexed addresses and data lines. It also provides parity checking to ensure that the correct data is transferred [7]. Considering these features of PCI, no additional data checking [such as Cyclic Redundancy Check (CRC)] is employed in the data communication between VTB and the UC, thus improving the data exchange rate as well.

B. Data Exchange Protocol

The PCI interface chip (CY7C09449PV-AC) provides 128K-bit Dual-Ported shared SRAM and allows the UC to communicate with the PC during runtime. VTB and the UC exchange data through this shared memory, because it can be easily accessed from both the PCI bus and the DSP local bus

[8]. The shared memory space, whose address offset is from 0x4000H—0x7FFFH, is divided into three sections: section I (flags), section II (VTB output data), and section III (UC output data (Table I).

TABLE I

ADDRESS MAP OF THE SHARED MEMORY

SECTION	SHARED MEMORY ADDRESS OFFSET	SIZE (BYTE)	NOTES
I	0 X 4000	1	Enable
	0 X 4004	1	Ready
	0 X 4008	1	Done
II	0 X 400C–0 X 419C	100 X 4	Output data of VTB (From VTB to the UC)
III	0 X 41A0–0 X 4330	100 X 4	Output data of the UC (From the UC to VTB)

Under this architecture, the data from VTB and the UC are clearly separated; hence the challenges associated with read and write data stored in the same address are avoided. The first bits of the first three words of the shared memory are defined as flags. These three flags are used to synchronize VTB and the UC. Considering that the data type used in the DSP is a 32-bit float, the same data type is chosen in the PIL interface model. As a result, every datum occupies 4 bytes in the shared memory. The maximum output variable numbers of VTB and the UC are set to be 100.

C. Synchronization

To realize the synchronization between VTB and the UC, a mechanism is employed to ensure that the shared memory space is controlled during all of the PIL processing. Three flags (Enable, Ready and Done) are adopted. The Enable flag, written by VTB only, is used to enable the DSP to execute the control algorithm. The Ready and Done flags are changed only by the DSP and read by VTB to determine if the DSP is ready to begin and if it has finished the execution of the control algorithm respectively. Fig. 3 shows the timing of the three flags.

The states of the three flags are named according to their hexadecimal value whose most significant bit is the Enable and the Done represents the least significant bit. For example, a state 100 (translated into state 4 as well) is given by reading that the Enable is high, the Ready is low and the Done is low. The state's sequence is in order of 2-6-4-5-1-0-2. The DSP executes the control algorithm in state 4 and idles in the other state; VTB runs simulation in the state 2 and idles in the other states.

The main advantages of this mechanism are as follows:

1) This synchronization system is independent from any clock. This characteristic allows the synchronization mechanism to be independent from data transfer issues which are involved in operating systems, kernel and PCI bus. For example, data transfer might be stopped by an interrupt requirement launched by other PCI devices with higher priority.
2) This system has the ability to do self-repair. Two states which are defined as error states are not used (3 and 7). If these two states show up, it indicates that the synchronization system has crashed. The DSP will go to default states (2 and 5, respectively) and abort the current control cycle.

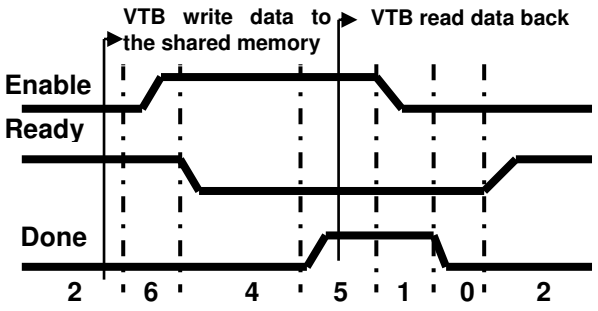

Figure 3. Synchronization Timing Diagram

In every simulation time step, VTB checks whether the set-up sampling step is reached. If not, VTB goes to the next simulation time step without calling the UC. When vice versa the sampling time is reached, VTB outputs the simulation results of the plant (the output of the plant) to section II of the shared memory and sets up the Enable flag in section I of the shared memory to indicate to the UC that input data are ready for processing. After that, VTB waits until Ready and Done show that the UC has finished the calculation and sent the computed result to section III of the shared memory. VTB successfully retrieves the updated computed results from section III, if the maximum set-up waiting time has not been reached. VTB then clears Enable, and at this point, one data exchange for a sampling interval between VTB and the UC is complete. VTB proceeds to the next simulation time step. If VTB cannot retrieve the results from the UC before the maximum set-up waiting time has elapsed, VTB reports the error information in its message window, clears Enable, and continues into the next simulation time step using the calculation results from the previous successful sampling step.

On the UC side, when the DSP sees Enable showing that the new simulated data is ready, it sets Ready to be low and reads the data in section II of the shared memory. After executing the control algorithm, the DSP sends the results back to section 3 of the shared memory and enables the flag Done to inform VTB that the updated results are ready. After VTB clears the Enable, the UC sets the Ready to be high. Then, the UC idles until it sees that the Enable changes to show the arrival of the new sampling interval.

The data flow between VTB and the UC is shown in Fig.4.

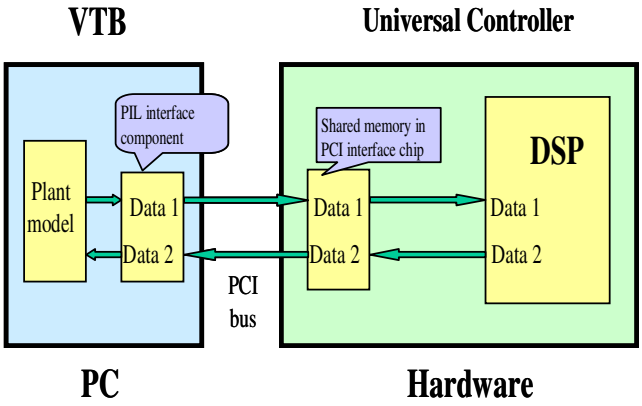

Figure 4. Data flow between UC and VTB

The operations above show that in the UC-PIL simulation, VTB operates as a master that can run or stop the control according to the timing requirement of the simulation platform.

D. UCPIL Model in VTB

A custom VTB model (UCPIL model) operates on the VTB schematic editor as interface to the UC. This model (Fig. 5) has five parameters (Table II), i.e. Input Vector Row Size, Output Vector Row Size, Sampling Time, Accuracy, Max times (for waiting).

Because the input and output of the block are vectors, users can freely decide input/output variable numbers (up to 100) depending on different applications. The features of VTB enable an independent sample rate. In the UCPIL model, users can customize a desired sampling step by setting parameters Sampling Time and Accuracy. The parameter, Max times (for waiting), is used to set up the max waiting time for one control cycle. This waiting time includes communication time spending on PCI bus and local bus between the PCI interface chip and the DSP, the calculation time for control algorithm, unexpected waiting time caused by other processing and so on.

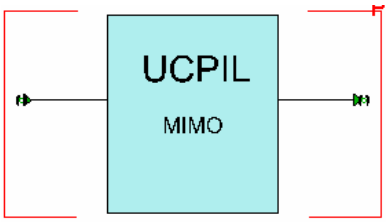

Figure 5. UCPIL Model

TABLE II

UCPIL MODEL'S PARAMETERS

Parameter	Value	Units
Input Vector Row Size	4	N/A
Output Vector Row Size	3	N/A
Sampling time	0.001	Second
Accuracy	0.00001	Second
Max Times(for waiting)	1000	N/A

Fig. 6 illustrates the flow chart of the UCPIL model.

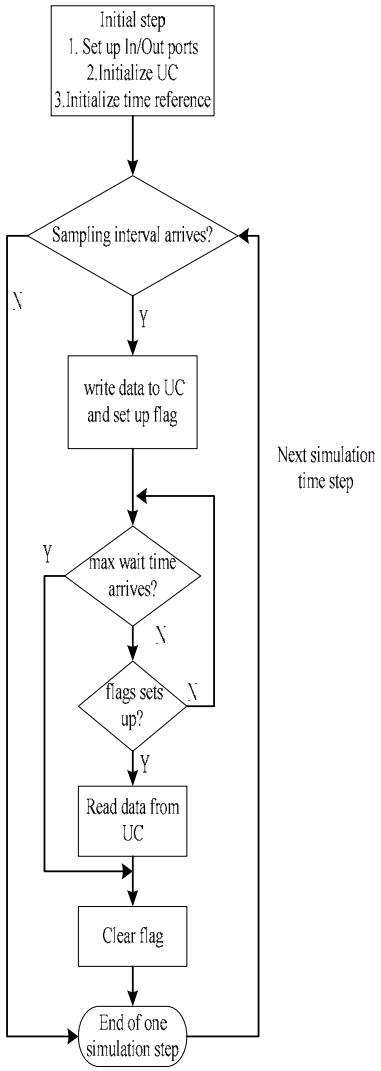

Figure 6. Flow chat of the UCPIL model

In order to exchange data between the UC and VTB, the Windows NT driver (HK-Cypr.sys) and the DLL file (HKCypr.dll) offered by Cypress Semiconductor are used. The drive is a Plug–and-Play device driver, which runs in kernel mode and can provide its services to a software application via an API (application programming interface) and a user-mode driver. The API is made up of a set of C language functions that may be invoked from C, C++, or assembly language. Functions such as OpenPCIDP, MapBaseRegister, and UnMapBaseRegister are used to access the shared memory from the PC side [9]. Fig.7 shows the basic structure of the driver application in the VTB-UC PIL.

E. An Application of the PIL Simulation with the UC

In this application, a three-phase, PEBB-based (power electronics building block) voltage source inverter is used as a

test system. Fig. 8 displays the schematic with switching PEBB and UCPIL models. The load connected to the inverter is a three-phase, passive R-L load. Two sampled line currents are fed back to the UC to execute a current loop on the d-q axes. The UC generates the phase-leg duty cycles via a minimum-loss discontinuous PWM algorithm. Fig. 9 shows a UC mounted on a PC's PCI bus. Fig. 10 and Fig. 11 show the simulation results of inverter line current and the phase-leg duty cycles using switching model. Fig. 12 shows experiment results of the inverter line currents, resistor voltages, and line-to-line switching waveforms.

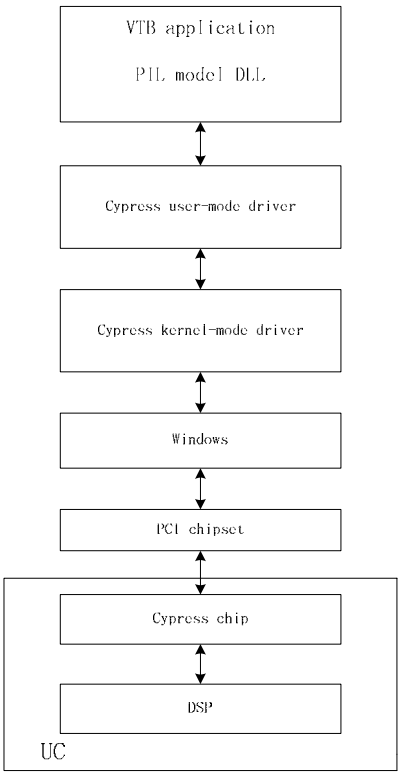

Figure 7. Basic structure of Cypress driver application

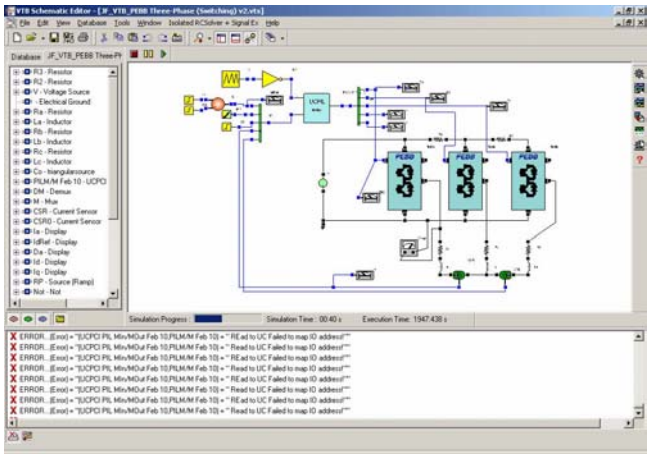

Figure 8. Switching PEBB-based converter schematic in VTB showing PEBB models and PIL model

Figure 9. A UC in the PC that is running VTB

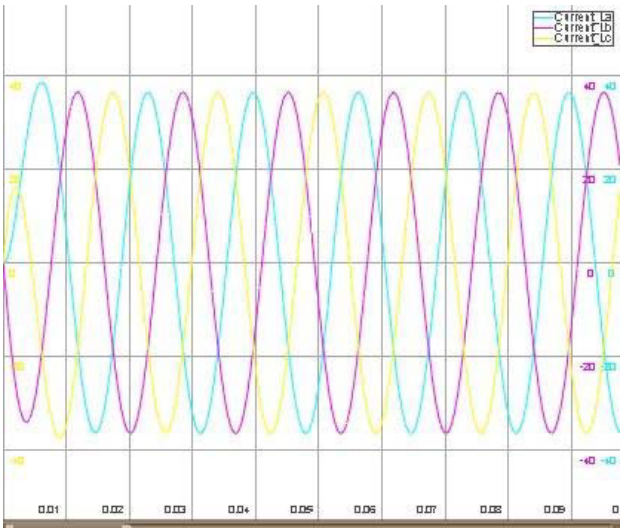

Figure 10. Line currents

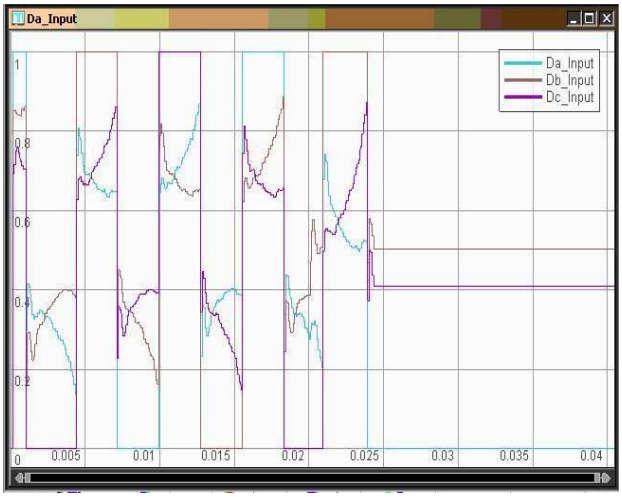

Figure 11. Phase-leg PWM duty cycles

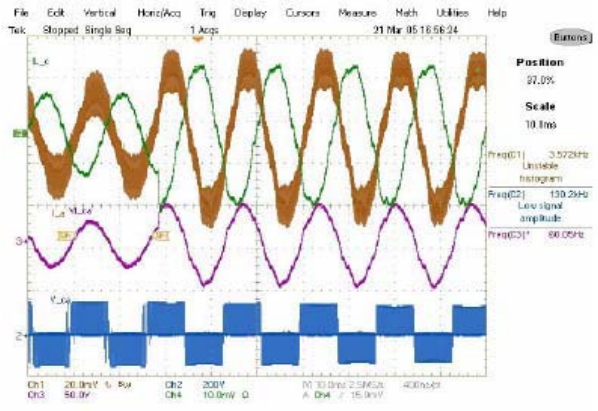

Figure 12. Inverter line voltages and currents.

V. CONCLUSIONS

This paper presented a PIL implementation with the UC in VTB. Issues such as synchronization, data exchange protocol, and model interface were addressed. PIL simulation and experiment results of the hardware system were compared. From this application, we can easily extend the PIL simulation to be used in other hardware platforms within VTB.

ACKNOWLEDGMENT

This work was supported by the Office of Naval Research under award numbers N00014-00-1-0489, N00014-03-1-0771, and N00014-02-1-0623.

This work made use of ERC shared facilities supported by the National Science Foundation under Award number EEC-9731677.

REFERENCES

[1] S. Lentijo, A. Monti, E. Santi, C. Welch and R. Dougal, "A new testing tool for power electronic digital control", IEEE Power Electronics Specialist Conference, PESC03, Acapulco, Mexico, June. 2003, vol. 1, pp. 107-111.

[2] Z. Jiang, R. A. Dougal, R. Leonard, H. Figueroa, A. Monti, "Hardware-in-the-Loop testing of digital power controllers," IEEE Applied Power Electronics Conference (APEC 2006), pp. 901-906, Dallas, TX, March 2006.

[3] S. Lentijo, A. Monti, R. Dougal , "Design of DC motor speed control through Processor-in-the-Loop approach," Hunstville Simulation Conference on CD, Huntsville, AL, October 2003.

[4] Francis, G.W.; Burgos, R.P.; Celanovic, I.; Wang, F.; Boroyevich, D.; "A universal controller for distributed control of power electronics systems in electric ships", American Control Conference, vol. 3, June 2005, pp. 1999-2004

[5] C. W. Brice, L. U. Gökdere and R. A. Dougal, "The Virtual Test Bed: An environment for virtual prototyping," Proceedings of International Conference on Electric Ship (ElecShip'98), pp. 27-31, September 1, 1998, Istanbul, Turkey.

[6] G Francis, G.; Burgos, R.; Rodriguez, P.; Wang, F.; Boroyevich, D.; Liu, R.; Monti, A., "Virtual Prototyping of Universal Control architecture systems by means of processor in the loop technology", Applied Power Electronics Conference, APEC 2007, Feb. 2007 pp. 21 – 27.

[7] *PCI Local Bus Specification Revision 3.0*, pp. 116, February 2004.

[8] *PCI-SDK/Cypress User's Manual*, pp. 3, CA, October 2004.

[9] *DEVELOPER'S GUIDE for the PCI-DP CY7C09449PV*, CA, April 2002.