A Low-Cost Real-Time Hardware-in-the-Loop Testing Approach of Power Electronics Controls

Bin Lu, *Member, IEEE,* Xin Wu, *Member, IEEE,* Hernan Figueroa, *Student Member, IEEE,* and Antonello Monti, *Senior Member, IEEE*

***Abstract*—Hardware-in-the-loop (HIL) testing is increasingly recognized as an effective approach in the design of power electronics controls. A high-performance real-time simulation environment is necessary to obtain high-fidelity results in HIL simulations. This paper presents the detailed implementation of a very-low-cost multisolver hard real-time simulation environment, namely the real-time extension of the virtual test bed (VTB-RT). VTB-RT is implemented completely from open-source software and off-the-shelf hardware. Using VTB-RT, this paper proposes an efficient real-time HIL testing approach for control designs in power electronics applications. VTB-RT enables the natural coupling between the simulation environment and the hardware under test and, thus, makes virtual power exchange in HIL simulation possible. For validation purposes, the proposed real-time HIL testing approach is applied in two well-known power electronics application examples, namely a boost converter and an H-bridge inverter with their respective control systems, representing a very-low-cost and a relatively advanced hardware setup, respectively. The consistency of the experimental results with the theoretical results proves the applicability of VTB-RT and the proposed testing approach. Finally, the most recent research progresses in VTB-RT are summarized.**

***Index Terms*—Distributed simulation, field-programmable gate array (FPGA), hardware-in-the-loop (HIL), power electronics, real-time simulation (RTS), virtual test bed (VTB).**

I. Introduction

TRADITIONAL software-based simulation has the disadvantage of being unable to exactly replicate real operational conditions. One way to bridge the gap between simulation and real conditions is the hardware-in-the-loop (HIL) simulation. Real-time HIL simulation replaces the emulated hardware under test or control logic in the simulation model with real hardware that interacts with the computer models. This increases the realism of the simulation and provides access to the hardware features currently not available in software-only simulation models and, hence, reduces the risks of discovering an error in the very last stage of the on-the-field testing [1].

Over the years, many real-time simulation (RTS) experiments have been proposed in the literature [2]–[9]; however, these systems are soft real time or even non real time, hardware dependent or based on costly proprietary solutions, or supported only by a single platform or solver. In order to extend the application of this technology, a very-low-cost hard RTS environment for HIL experiments is developed at the University of South Carolina, completely built on open-source software and off-the-shelf hardware. In this paper, this system is referred to as the real-time extension of virtual test bed (VTB-RT). The VTB-RT has unique properties such as multiplatform, multisolver, and hard real time. Instead of competing with commercial systems, it is designed to provide a very-low-cost alternative for real-time HIL applications while maintaining acceptable resolutions.

Real-time HIL simulation has a wide application such as power quality disturbances investigation and modern automotive electronic control unit development. This paper focuses on the power electronics applications. The application of RTS to power electronics system is, in particular, a very active field of research. In effect, power electronics can be considered a significant challenge for RTS, considering the requirement in terms of time accuracy. An interesting example of application can be found in [7], which analyzes a photovoltaic system using a commercial real-time platform. The main drawback is given by the cost of the adopted platform. On the other hand, going in the direction of limiting the cost of the platform, it is important to have procedure able to overcome the restrictions given by the time accuracy requirement. An important contribution is proposed in [8], where a mechanism of error compensation for fixed time step simulation is proposed. Other possible approaches are offered by the use of averaging methods such as reported in [9]. Still in the same direction, [10] proposes an extended method for averaging actually adopted for the platform proposed in this paper.

In the following, we will focus on the description of the main features and on the use of VTB-RT, an efficient real-time HIL testing approach for power electronics control system design. To meet with the hard real-time constraint, fast dynamics, and the ever-increasing complexity of the power electronics systems, distributed simulation using VTB-RT is also adopted in this testing approach.

The full procedure including both software simulation and hardware implementations of the proposed approach is

Manuscript received March 22, 2005; revised December 19, 2006. Abstract published on the Internet January 14, 2007. This work was supported in part by the U.S. Office of Naval Research under Grant N00014-02-1-0623 and Grant N00014-03-1-0434.

B. Lu was with the Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208 USA. He is now with the Innovation Center, Eaton Corporation, Milwaukee, WI 53216 USA (e-mail: binlu@ieee.org).

X. Wu was with the Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208 USA. She is now with Ansoft Corporation, Pittsburgh, PA 15219 USA (e-mail: xwu@ansoft.com).

H. Figueroa and A. Monti are with the Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208 USA (e-mail: figueroh@engr.sc.edu; monti@engr.sc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2007.892253

then applied to two classical power electronics application examples: a boost converter control system and an H-bridge inverter control system, representing a very-low-cost and a relatively advanced hardware setup, respectively. Since the results of these examples are well known in textbooks, they are adopted here intentionally to validate the applicability of the proposed testing approach.

This paper is organized as follows. Section II gives a brief background of VTB, the Windows-based non-real-time ancestor of VTB-RT. Section III introduces VTB-RT and its implementation. In Section IV, the proposed testing approach for real-time control of power electronics using VTB-RT is described in details. For validation purposes, two application examples are presented in Sections V and VI. Finally, in Section VII, the applicability of the proposed testing approach using VTB-RT is concluded through the comparison between the experimental results and the theoretical results. The most recent applications of VTB-RT in more complex systems are also summarized.

II. VTB Environment

The VTB project is dedicated to developing a new environment for simulation and virtual prototyping of power electronics systems. Within the context of "virtual prototyping," not only simulation of system dynamics, but also solid modeling of the system, is included. In the VTB, native models of natural components (those obeying natural conservation laws) are built based on the resistive companion modeling (RCM) method. The ubiquity of the across and the through variables across disciplinary lines allows the VTB to be used for simulation of multidisciplinary systems. As a high-level virtual prototyping tool, the VTB program addresses many challenges in the fields such as power electronics, which encompasses a wide range of disciplines, including analog electronics, digital electronics, power systems, controls, electromechanics, and mechanical systems. The VTB environment addresses these challenges by supporting:

1) Multiformalism: To enhance interdisciplinary uses, the VTB integrates simulation models defined in other environments such as Spice, MATLAB, and ACSL via translation and/or cosimulation. This allows individuals to build models using the preferred language within their discipline and, hence, makes the VTB a versatile environment for virtual prototyping.

2) Multiplatform: The VTB is developed using the C++ language. Therefore, it is portable among several platforms such as the Windows, UNIX, Linux, and Mac operating systems (OSs).

3) Multisolver: By using the VTB, one simulation system can be distributed among several computers to perform distributed simulation. Each host computer can choose different VTB solvers. This means that any part can be solved with the most appropriate integration method and step without affecting the rest of the system. Currently, two VTB solvers have been ported to Linux and are available for the VTB-RT:

a) Simulated analog computer (SAC) solver: The SAC solver is a block diagram solver. It solves a system's method of describing differential equations in the same way an analog computer would. This was the first VTB solver transplanted to the VTB-RT due to its ease of handling.

b) Signal extension resistive companion (SRC) solver: The heart of the SRC solver is its support of the RCM method, which expresses each component in a system as

$$I(t) = G(t-h) * V(t-h) - B(t-h) \tag{1}$$

where I is the through variable vector of the component, V is the across variable vector, G is the conductance matrix, B is the state source vector, t denotes time, and h is the simulation step size. The significance of the RCM method is that it models each component following the natural conservation laws. Therefore, the SRC solver supports natural coupling between simulation models and real hardware. Additionally, it supports signal coupling because of a multibackplane that allows for mixed-signal simulation.

4) Highly Interactive Environment: A user can change the system topology or parameters while a simulation is executed. This allows the user to rapidly investigate interactions between components or to explore the influence of design parameters on system performance.

5) High-Level Visualization: Visualization models of the system can be easily created and linked to live simulation data by the visualization extension engine. Visualization helps the user to rapidly comprehend the system performance. Visual outputs include the data-driven animation of the motion of the solid objects, imposition on top of the solid objects of novel representations of abstract simulation data, or simply the oscilloscope-like waveforms.

III. VTB-RT Environment

A. VTB-RT Overview

In the case that real hardware is involved in the simulation process, the software must deal with additional challenges. These challenges motivate toward the VTB-RT, the real-time extension of the VTB. Hardware-oriented simulation has to deal with more problems than software-based simulation, especially the inexorable forward progression of time. This requires the simulation environment to operate in hard real-time mode. However, OSs like Windows and Linux are unable to provide hard real-time capabilities. To address this issue, an open-source Linux kernel modification package, namely, the real-time application interface (RTAI), has been developed by the Department of Aerospace Engineering of the Politecnico di Milano. RTAI completely changes the way Linux receives and handles hardware interrupts and enables the hard real-time capability of Linux. In VTB-RT, Linux and RTAI were adopted as the underlying real-time OS. The VTB-RT shares the major parts of its architecture with the VTB. It reads the same file format created by the VTB schematic editor under Windows, thus making it convenient to export simulations from the non-real-time platform, namely the VTB, into the real-time platform, namely the VTB-RT.

B. VTB-RT Components

Many systems for HIL simulation have already been developed, such as RT-Lab, RTDS, and HyperSim. However, all of them are based on proprietary solutions. The objective of the VTB-RT is not to compete with these commercial systems, but instead, to provide a very-low-cost solution for real-time HIL applications, while maintaining acceptable speed and resolutions. The VTB-RT is completely implemented with open-source software and low-cost off-the-shelf hardware. From the software point of view, the VTB-RT consists of four free software packages:

1) Linux: Linux is selected as the OS of the VTB-RT because of its low cost, high performance, and open source. It provides the user interface, basic functions, and development tools. In the development process of the VTB-RT, various Linux distributions including Mandrake, Redhat, Caldera, and Suse have been demonstrated to be suitable as the OS. The open-source feature of Linux makes it possible to modify its kernel code to become real-time capable.

2) RTAI: Hard real time requires the OS to be preemptive and deterministic. In the VTB-RT, RTAI is used to achieve this requirement. The RTAI is a kernel modification package of Linux that permits the handling of hardware interrupts and therefore enables the hard real-time capability of Linux.

3) Comedi: A HIL simulation system requires I/O interfaces to the hardware. In the VTB-RT, this is achieved by Comedi, a freeware that develops open-source device drivers for many different data acquisition (DAQ) devices. It consists of two complementary packages: "comedi," which implements the kernel space functionality; and "comedilib," which implements the user space access to the device driver functionality. Comedi works with a standard Linux kernel as well as the RTAI.

4) VTB Solvers and Simulation Models: The SAC solver and the SRC solver are used by both the VTB and the VTB-RT. The solvers and simulation models are developed using the C++ language, and the same source codes are shared in these environments. In the VTB-RT, the source codes of the solvers and models need to be recompiled and made compatible with the real-time Linux environment. An important consideration to use the SRC solver is that it manipulates the real hardware interface, and as a result, it enables the natural coupling between the simulation environment and the hardware plant and, thus, makes possible the virtual power exchange between the simulation software and the hardware under test. At this point, the SRC solver is the main solver for both VTB and VTB-RT. The application examples in Sections V and VI use the SRC solver.

A complete insight description of the VTB-RT and its components is available in [12].

C. VTB-RT Configuration

Minimum and suggested hardware configurations for the VTB-RT host computer are listed in Table I. The minimum configuration represents the first version of VTB-RT. The suggested configuration is the one used in the application example in Section V. The speed and resolution of VTB-RT can be improved easily by choosing higher level hardware, as in the application example in Section VI.

TABLE I
MINIMUM AND SUGGESTED HARDWARE CONFIGURATIONS FOR VTB-RT

	Minimum configuration	Suggested configuration
CPU	Intel P200MHz	Intel P800MHz
Hard drive	2 Gigabytes	10 Gigabytes
RAM	64M	128M
Free PCI Slots	1	At least 1
I/O DAQ Cards	Yes (†)	Yes (†)

†: DAQ card should be listed in the Comedi supported hardware list as Appendix B of [12]. Otherwise, Comedi does not support the device and the driver has to be developed individually.

The software configuration for the VTB-RT includes the components listed in Section III: a clean Linux kernel with an RTAI patch, Comedi drivers, and VTB solvers.

A detailed VTB-RT implementation procedure is presented in the Appendix.

D. Real-Time Implementation

There are three major components in the VTB-RT real-time implementation: a real-time task, a Linux process, and a real-time first-in–first-out (FIFO) buffer.

1) Real-Time Task: The RTAI preempts the standard Linux kernel and handles hardware interrupts. In the VTB-RT, a real-time task is generated by the RTAI to manage the 8254 chip (clock generator) to generate a real-time clock, which is used as the basis for defining the simulation step. This real-time task is a loadable module in Linux; it stays in the kernel space upon being loaded. Fig. 1 shows the pseudocode of a typical real-time task.

In a typical real-time task as Fig. 1, the following code sections are included.

- Application interface between the real-time task and the real-time FIFO: The interface between the real-time task and the real-time FIFO is defined in this section.
- Real-time task initialization function: A kernel module must always contain an init_module() function. It is invoked by the "insmod" command when the module is loaded. It prepares for later invocation of the module's functions. The step size and real-time application task are defined in this section. The application task sends a real-time clock message into the FIFO at the beginning of each time step. This message indicates a new simulation time interval for the user space, where the VTB-RT solver is located.
- Real-time task cleanup function: A cleanup function cleanup_module() is also required for any kernel module. It is invoked by the "rmmod" command when the module is unloaded. It informs the kernel that the module has been removed, and none of its functions are no longer called.

2) Linux Process: VTB-RT solvers are realized by a set of standard Linux processes. They are similar to other Linux programs such as a text editor. In each step interval, the solver takes in the system input from the input channel of the DAQ device, solves the system state, and sends the system output through the output channel of the DAQ device. The Linux process is a user-space application program and, hence, has no

```
#include Header Files
<Linux module, i/o, math, rtai, rtai_sched, rtai_fifos>


#define RT_Step_Size
#define RT_Task_Priority


 RTAI_API()
{
    Send_RT_Clock_Message_to_RT_FIFO();
}
```

```
init_module()
{
    Set_RT_Step_Size();
    Initialize_RT_Task();
    Create_RT_FIFO();
    Run_RT_Task_Periodicly();
}
cleanup_module()
{
    Stop_RT_Clock();
    Destroy_RT_FIFO();
    Destroy_RT_Task();
}
```

Fig. 1. Pseudocode of a typical real-time task.

```
#include Header Files
<Standard Linux header files>
<Header files for a specific VTB-RT solver>

#define Input_Channels
#define Output_Channels

 DAQ_Input (Input_Channels)
{
    Open_DAQ_Device();
    Set_Mode&Range_of_DAQ();
    Read_Data_From_DAQ();
    Data_Scaling();
    Close_DAQ_Device();
    Return_Input_Data;
}

 DAQ_Output (Output_Data, Output_Channels)
{
    Open_DAQ_Device();
    Set_Mode&Range_of_DAQ();
    Data_Rescaling();
    Send_Data_to_DAQ();
    Close_DAQ_Device();
    Return;
}
```

```
VTB-RT_Solver (System_Inputs)
{
    System_Outputs = Solve_State(System_Inputs);
    Return_System_Outputs;
}


main()
{
    Initialize_RT_FIFO();
    Initialize_VTB-RT_Solver();

    While(!end)
    {
      Read_RT_Clock_From_RT_FIFO();
      If (RT_Clock_Updated)
      Then
        {
          System_Inputs = DAQ_Input(Input_Channels);
          System_Outputs = Solve_State(System_Inputs);
          DAQ_Output(System_Outputs, Output_Channels);
        }
    }

}
```

Fig. 2. Pseudocode of a typical VTB-RT solver.

direct communication with the real-time task. Fig. 2 shows the pseudocode of a typical VTB-RT solver.

In a typical VTB-RT solver as in Fig. 2, the following code sections are included.

- *DAQ input function.* This function reads the input data of the under test system from the input channels of the DAQ device.
- *DAQ output function.* This function sends the output data of the under test system to the output channels of the DAQ device.
- *VTB-RT solver function.* This function transplants the VTB solver from Windows environment to Linux environment. It is the "heart" of the VTB-RT platform. It solves the state of the under test system using the input data and generates the system outputs during each simulation interval.
- *Main program.* The main program incorporates the previous three functions. It polls the real-time FIFO for the real-time clock update. If an updated real-time clock is detected, it will call the DAQ input function, VTB-RT solver function, and DAQ output function in sequence.

3) Real-Time FIFO: Because in the VTB-RT the real-time clock information has to be passed to the solver, a real-time FIFO is applied as the "bridge" between the real-time task and the Linux process. The real-time FIFO is a unidirectional read/write buffer created by the RTAI. After simulation starts, it continuously records the real-time clock generated by the real-time task. Simultaneously, the Linux process polls the real-time

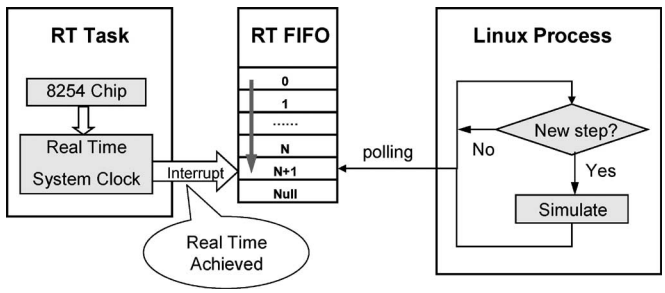

Fig. 3. Real-time implementation of VTB-RT.

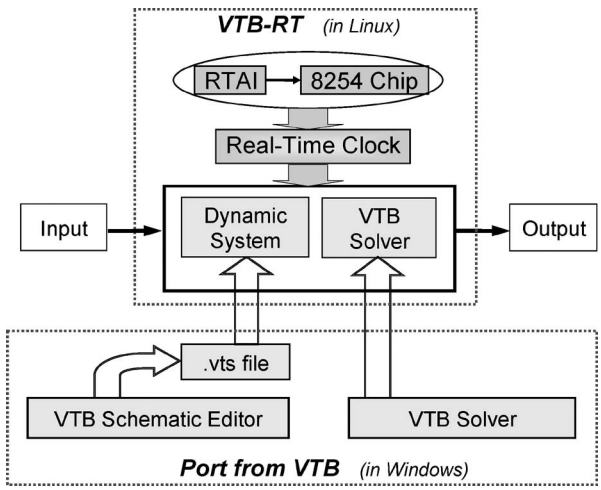

Fig. 4. Architecture of VTB-RT.

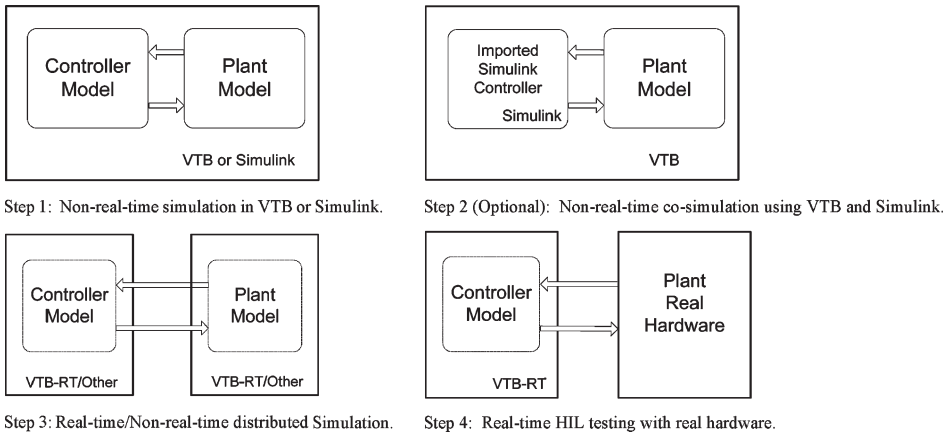

Fig. 5. Design approach of real-time control for power electronics.

FIFO, detects the real-time clock, and performs the simulation. From the programming point of view, the real-time FIFO can be divided into two parts: one part embedded in the real-time task and the other in the Linux process, as shown in Figs. 1 and 2.

Fig. 3 graphically illustrates how real time is achieved in the VTB-RT platform using the previously discussed three major components. More detailed explanations and example codes of these components are available in [12].

E. VTB-RT Architecture

Fig. 4 shows the architecture of the VTB-RT. This architecture allows the user to perform a real-time HIL testing of a system that includes real hardware. This testing phase can be considered as the very last step before the real on-line testing of power electronics controls. Through this process, the user can verify not only the algorithmic correctness of the system (e.g., a controller) in the simulator, but also its capability to meet the real-time constraints.

The simulation schematics can be created using the VTB Schematic Editor under a Windows platform. The schematic file format .vts is compatible with the VTB-RT under Linux. This enables the user to directly export a simulation from a non-real-time platform to a hard real-time platform. It is important to underline that this step does not require any compilation process. The .vts file is shared among different platforms and is an ASCII file describing the system by using an XML-based notation.

IV. Approach to Real-Time Control for Power Electronics Using VTB-RT

Power electronics control designers often use simplified models of the plant in the early stages of the system design. By doing this, they can focus on the control algorithm itself rather than the complexity of the plant. However, many factors may affect the viability of the derived controller, such as oversimplification of the model and data communication time delays. Therefore, in order to be confident of the design, the control system must be tested under more strict conditions.

TABLE II
VTB-RT Configurations in Boost Converter Control Example

Hardware configuration		Software configuration	
CPU	Intel PIII 800MHz	Linux release	Mandrake 9.0
Hard drive	10 Gigabytes	Linux kernel	2.4.20
RAM	128M SDRAM	RTAI	24.1.10
Network	10 Mbps ether net	Comedi	0.7.66
I/O DAQ	Advantech PCI1710	Comedilib	0.7.19
Devices	Advantech PCI1720	gcc version	3.2

TABLE III
Parameters of Prototype Boost Converter

Rated input voltage	12V
Rated output voltage	40V
Maximum output power	100W
Input inductance	46μH
Output filter capacitance	1.360mF
Main switch	IRF540N
Switching frequency	50kHz
Load resistance	35 Ω

For instance, in the case of the switching power converter, three different levels of models could be considered subsequently:

- averaged model;
- switching model;
- real hardware real-time HIL testing.

A. Monti *et al.* [11] gives a detailed example of using both the averaged model and the switching model of a boost converter. This paper focuses on the real-time HIL testing with real hardware. Using the VTB-RT, a very-low-cost real-time HIL testing approach for control designs in power electronics applications is proposed. This approach is summarized in four steps, as shown in Fig. 5.

Step 1) Design a high-fidelity system model for both the controller and the plant in the VTB or other non-real-time simulation environments such as Simulink. Tune the control parameters if necessary. The result of this step will be the ideal result.

Step 2) Link the controller model in Simulink developed in Step 1) into the VTB and perform the VTB–Simulink cosimulation (refer to example I). This result can be used as a comparison to analyze the HIL results in Steps 3) and 4). This step can be skipped for an even lower cost VTB/VTB-RT

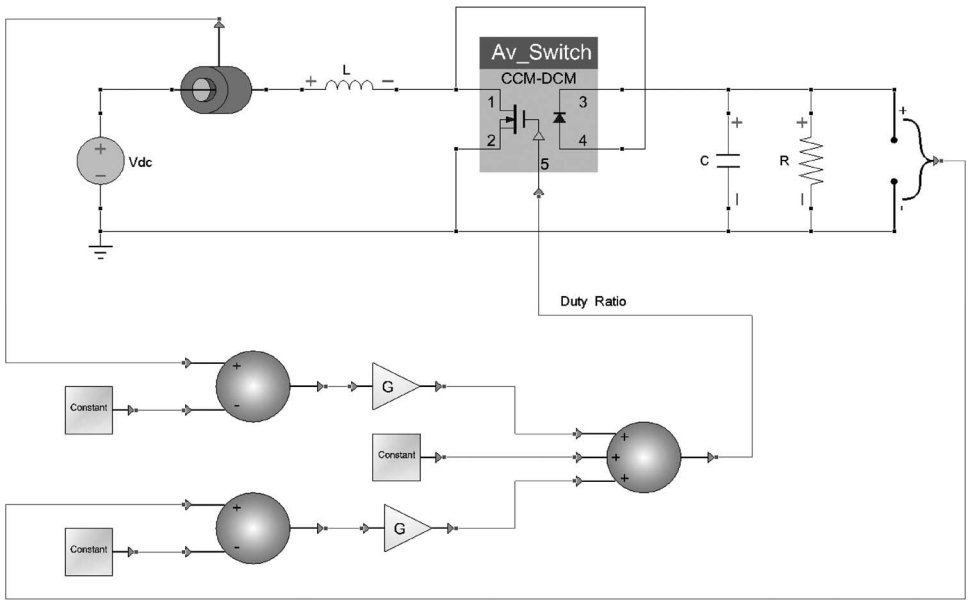

Fig. 6. Schematic of the boost converter control system in the VTB.

only design, where Simulink is not involved (refer to example II).

Step 3) This is the distributed simulation stage. The whole system is partitioned into two subsystems: one contains the model of the controller; and the other contains the model of the plant. Different simulation environments can be used to host these subsystems. This step can be either real time or not, depending on the host environments. Application example I shows the non-real-time distributed simulation using dSpace and VTB. Application example II shows the hard real-time distributed simulation using two separate VTB-RT computers.

Step 4) Keep the controller model in the VTB-RT and replace the plant model with real power electronics hardware. Perform the hard real-time HIL testing.

Following these steps, the VTB-RT provides an efficient design and testing approach for power electronics controls. The real-time HIL testing eliminates the costly hardware and greatly reduces the design cost. It should be pointed out that the limitation of VTB-RT and the proposed design approach primarily lies in the fact that, for a given hardware platform, the minimum real-time step is limited. It means only a limited bandwidth of the system under test can be used. Therefore, in the actual design stage, special attention must be paid to compromise between the VTB-RT platform cost and the bandwidth of the system under test. At this time, a minimum real-time step of 250 μs has been achieved using low-level hardware, and 50 μs has been achieved using medium-level hardware. This time resolution satisfies most power electronics controls.

V. Application Example I: Real-Time State-Space Feedback Control for a Boost Converter

This application example illustrates the process during the implementation of a state-space feedback control for a boost converter, showing all of the steps described in Section IV. Particularly, it focuses on the real-time HIL testing with very-low-cost hardware [1].

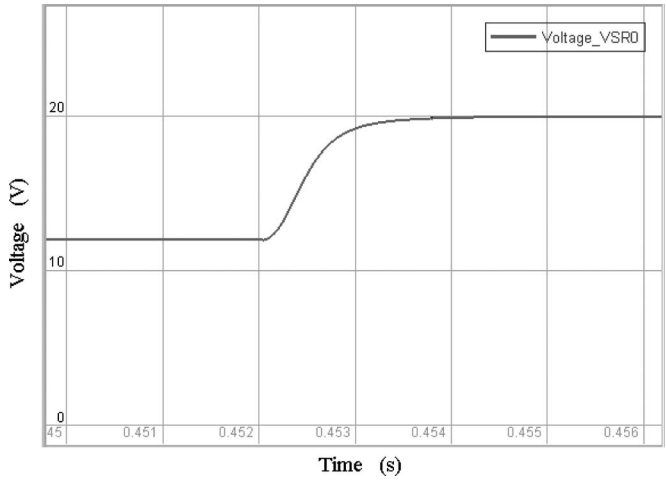

Fig. 7. Output voltage of the closed-loop system in the VTB, when reference voltage steps from 12 to 20 V (software-only simulation case).

First, the analytical control design uses pole placement on the linearized system. Poles are selected so that the closed-loop system has satisfactory dynamic behavior. The designed system is then built in the VTB to obtain the ideal results. Then, the VTB–Simulink cosimulation is performed by importing the Simulink controller into the VTB. After that, the non-real-time distributed simulation is performed on this system using dSpace and the VTB. This step is not hard real time, since the plant model is still hosted in Windows. However, it gives a general approach for the HIL testing where a hard real-time system is not available. Finally, the real-time HIL testing is performed using the VTB-RT and a real boost converter.

The hardware and software configurations of the VTB-RT computer used in this example are listed in Table II. The boost converter has the parameters given in Table III.

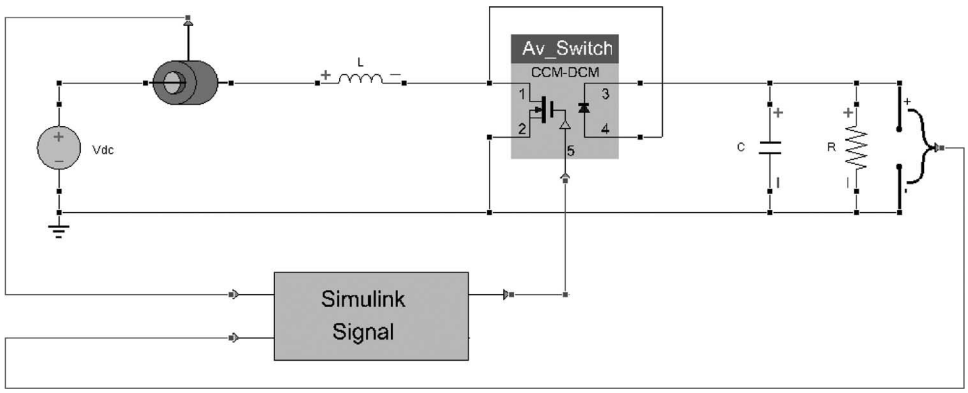

Fig. 8. VTB schematic for VTB–Simulink cosimulation.

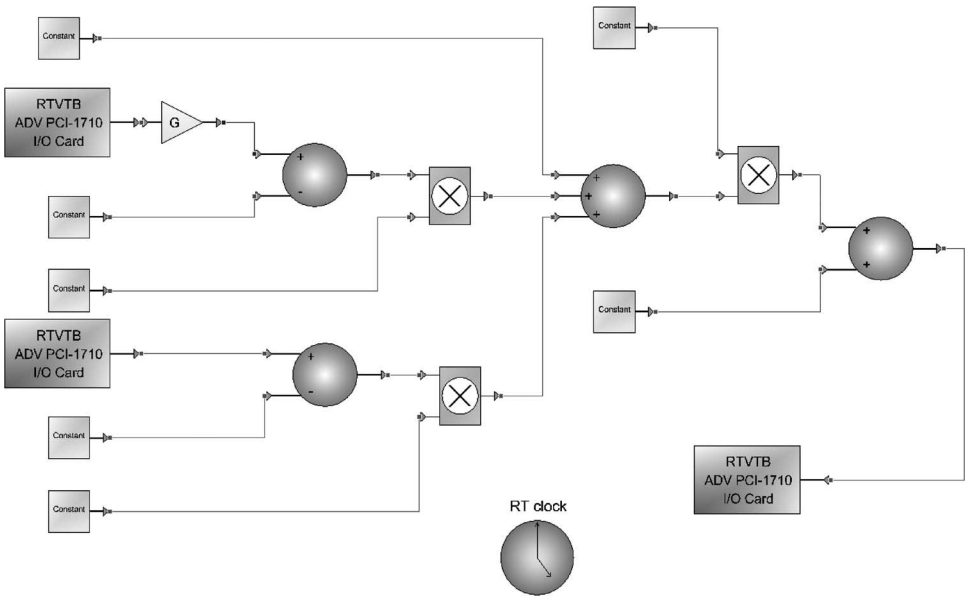

Fig. 9. State-space feedback controller model in the VTB (exported into the VTB-RT).

A. Designing and Testing Procedures

Step 1) *Non-real-time simulation in VTB.* The plant and the feedback controller are designed in the VTB as in Fig. 6. The desired closed-loop poles are chosen to be -2000 and -5000, and the desired output voltage is 20 V, which results in the following parameters of the controller: $k_1 = -0.0161$ and $k_2 = -0.0206$, according to [1]. Fig. 7 shows that when the reference output voltage steps up from 12 to 20 V, the actual output voltage follows the reference.

Step 2) *Non-real-time VTB–Simulink cosimulation.* The controller is then implemented in Simulink and imported into the VTB by using the interactive interface between the VTB and Simulink. The schematic of the cosimulation system is shown in Fig. 8.

Step 3) *Non-real-time distributed simulation.* The dSpace system supports direct compilation of the Simulink control definition, and the compiled code can be downloaded into the dSpace platform without any change. Benefiting from this, the Simulink controller model is compiled and downloaded into the dSpace platform, and non-real-time distributed simulation is performed between the VTB and dSpace. Since Steps 2) and 3) do not deal with hard real time, for simplicity, the simulation results are omitted here. Detailed results are available in [2].

Step 4) VTB-RT HIL testing with a real boost converter. The state-space feedback controller is implemented in the VTB and then exported into the VTB-RT, as shown in Fig. 9. The "RT clock" model represents the real-time clock in the VTB-RT. Three "VTB-RT ADV PCI-1710 I/O Card" interface models are used for signal exchange between the VTB-RT and the boost converter hardware.

Finally, the hard real-time HIL testing is performed with the controller in the VTB-RT and a real boost converter. Fig. 10 shows the response of the boost converter when the reference voltage steps up from 12 to 20 V. It is in good agreement with Fig. 7, the ideal result predicted in Step 1).

Notice that the steady-state output voltages in Fig. 10, namely, 11.6 and 19.2 V, are less than the ideal case in Fig. 7, namely, 12 and 20 V. This is because the simplified models in

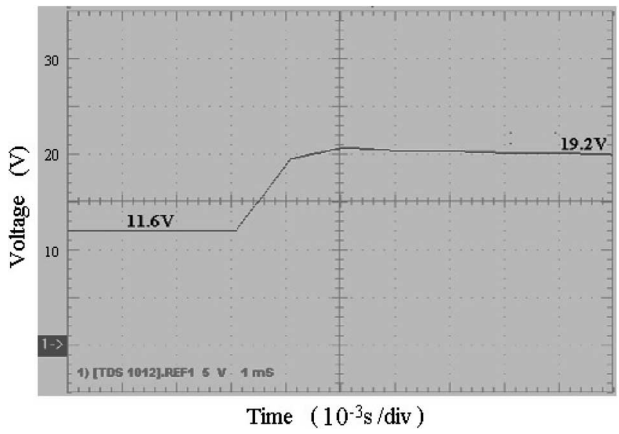

Fig. 10. Output voltage of the real boost converter, when reference voltage steps from 12 to 20 V (hard real-time HIL simulation case).

TABLE IV
CONFIGURATIONS OF A MORE ADVANCED VTB-RT PLATFORM

Hardware configuration		Software configuration	
CPU	Intel P4 2.66GHz	Linux release	RedHat 8.0
Hard drive	60 Gigabytes	Linux kernel	2.4.20
RAM	512M SDRAM	RTAI	24.1.11
Network	10 Mbps ether net	Comedi	0.7.69
I/O DAQ	Advantech PCI1710	Comedilib	0.7.20
Devices	Advantech PCI1720	gcc version	3.2

the ideal case do not take into account the losses in the actual hardware. Furthermore, it is observed that the experimental result has piecewise-like behavior caused by the RTS step size. In this case, the step size is 300 μs.

If a faster host computer is used for the VTB-RT, the simulation step size can be easily reduced, and the result will be smoother. For instance, a time step of 50 μs has been achieved for a similar example using a more advanced VTB-RT platform, whose configurations are listed in Table IV [10].

The agreement between Figs. 7 and 10 confirms that the real-time state-space feedback control of a boost converter is achieved through the proposed VTB-RT hard real-time HIL testing approach.

VI. APPLICATION EXAMPLE II: REAL-TIME FEEDBACK CONTROL FOR AN H-BRIDGE INVERTER

The second example demonstrates the design of a digital controller for an H-bridge inverter using only the VTB and the VTB-RT, eliminating any other third party design software. This example focuses on the real-time distributed simulation step.

First, a proportional–integral (PI) controller is designed analytically at desired bandwidth and phase margin. The system stability is analyzed in discrete-time domain, and a stable integration time step limit is obtained. The designed closed-loop system is then built in the VTB to obtain the ideal results. The VTB–Simulink cosimulation step is skipped, since in this example, the design capability of VTB/VTB-RT without help of third party software is a concern. After that, the whole system is partitioned into an inverter subsystem model and a controller subsystem model, hosted by two identical NI VXI-

TABLE V
VTB-RT CONFIGURATIONS IN THE H-BRIDGE INVERTER CONTROL EXAMPLE

Hardware configuration		Software configuration	
CPU	Intel PIII 1.26 GHz	Linux release	Mandrake 9.0
Hard drive	15 Gigabytes	Linux kernel	2.4.19
RAM	128M SDRAM	RTAI	24.1.10
Network	100 Mbps ether net	Comedi	0.7.66
I/O DAQ Device	NI PCI-6070E	Comedilib	0.7.19
		gcc version	3.2

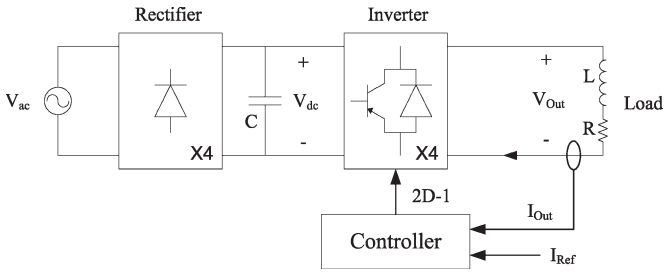

Fig. 11. Description of the H-bridge inverter system.

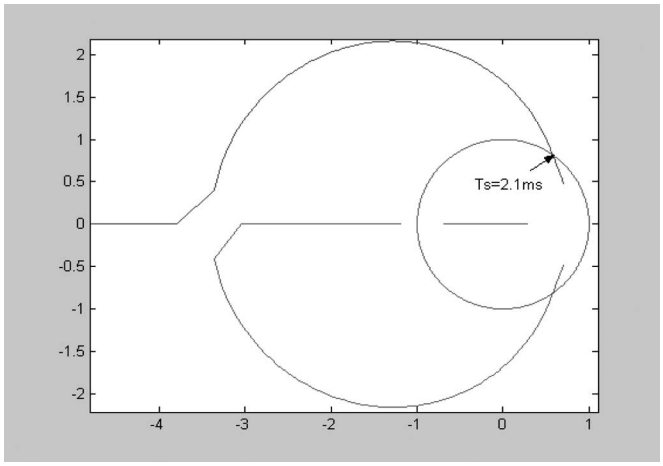

Fig. 12. Root locus of the discrete system with a one-time-step delay.

872Bpc computers. As the focus of this example, hard real-time distributed simulation is performed.

The configurations of the VTB-RT computer used in this example are listed in Table V.

The whole system is described in Fig. 11. A sinusoidal voltage source is rectified and supplies the dc voltage level for the inverter. The output current of the inverter is fed into a PI controller. This current is compared with the reference, and the output of the controller determines the duty ratio of the switching of the inverter. The load is a 1-mH inductor with an internal resistance of 0.01 Ω.

A. Analytical Design of the PI Controller

Choose the bandwidth and the phase margin of the open-loop transfer function as

$$\omega_{\mathrm{BW}} = 2\pi \times 100\ \mathrm{Hz} = 628\ \mathrm{rad/s} \quad \varphi_{\mathrm{PM}} = 50^{\circ}.$$

To compensate for the delay effect caused by the D/A and A/D conversion and by the signal transition between the plant

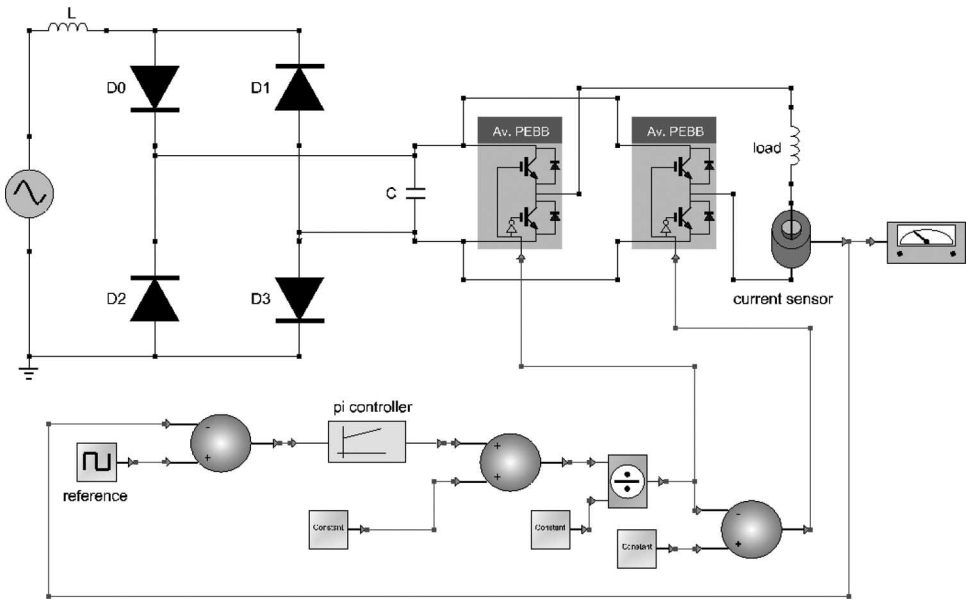

Fig. 13. Schematic of the H-bridge inverter control system in VTB.

and the controller, the phase margin is modified as

$$\varphi^*_{\mathrm{PM}} = 50^\circ + \Delta\varphi = 50^\circ + \frac{\omega_{\mathrm{BW}} \times T_s}{2} = 54.5^\circ.$$

Choosing the sampling period of the controller T_s to be 250 μs, the parameters of the PI controller can be calculated as

$$K_p = 0.0169 \quad K_i = 7.8213.$$

The open-loop transfer function of the whole system is then obtained as

$$G_O(s) = \frac{0.506s + 234.4}{0.001s^2 + 0.01s}. \tag{2}$$

B. Stability Analysis

The SRC solver uses the trapezoidal integration method. Therefore, the open-loop transfer function of the whole system can be transformed from s-domain to z-domain as

$$G_O(z) = \frac{0.506\frac{2}{T_s}\frac{z-1}{z+1} + 234.4}{0.001\left(\frac{2}{T_s}\frac{z-1}{z+1}\right)^2 + 0.01\frac{2}{T_s}\frac{z-1}{z+1}}. \tag{3}$$

The main stability issue of this system is related to the delay introduced by the signal processing, A/D and D/A conversion, and model integration. With a proper sampling frequency, the system can be approximated as a hybrid system with an internal one-step delay. Based on this estimation, the system characteristic function can be derived as

$$1 + \frac{G_o(z)}{z} = 0. \tag{4}$$

The stability analysis gives the root locus of the system with different sampling periods, as shown in Fig. 12. When the time step is greater than 2.1 ms, the system becomes unstable. Because of the system complexity and the hardware limit, the minimum time step that can be achieved while satisfying the real-time constraints is 250 μs with a system configuration shown in Table V. Therefore, the time step is chosen to be 250 μs in this example.

C. Testing Procedures

1) Non-Real-Time Simulation in VTB: The entire system is simulated using the VTB as shown in Fig. 13. A delay in the feedback loop is inserted into the closed-loop simulation to emulate the effect of the connection between the two separated machines at the distributed simulation stage. Two output current reference signals at a frequency of 10 Hz are applied separately, i.e., a square wave and a sinusoidal, with amplitudes of 1.5 and 2.5 A, respectively. Fig. 14 shows that the output currents track the current references at a typical second-order behavior.

2) Real-Time Distributed Simulation: The entire system is then partitioned into two subsystems, a plant model and a controller model, as shown in Fig. 15.

The two subsystems are exported into two separate VTB-RT computers. These computers are two identical National Instruments VXI-872Bpc controllers (system configurations in Table V). The two simulation processes communicate through a direct analog connection between the DAQ devices embedded in each VXI controller, as shown in Fig. 16.

The results of the real-time distributed simulation are shown in Fig. 17. The agreement between Figs. 14 and 17 validates that a PI controller of an H-bridge inverter is successfully designed by using the VTB-RT real-time distributed simulation. The design of the controller has already been achieved at this stage. Following this, the hard real-time HIL testing can be easily performed by substituting the plant host VTB-RT computer with real inverter hardware.

The simulation time step cannot be lower than 250 μs because of the speed limitations of the VTB-RT platform. Table VI shows the distribution of time consumed by the

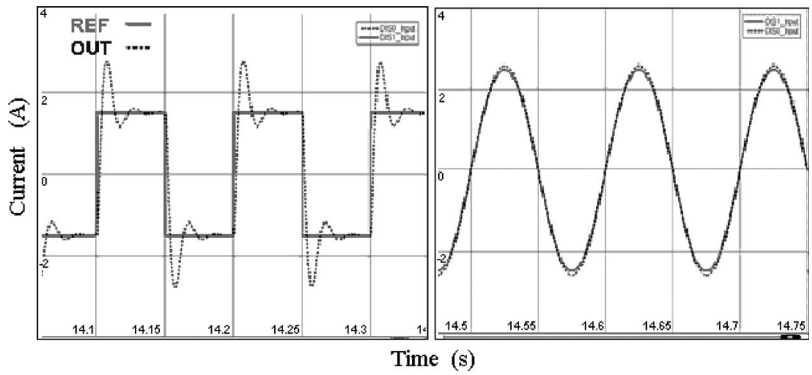

Fig. 14. Output currents and current references of the closed-loop system in VTB (non-real-time simulation case).

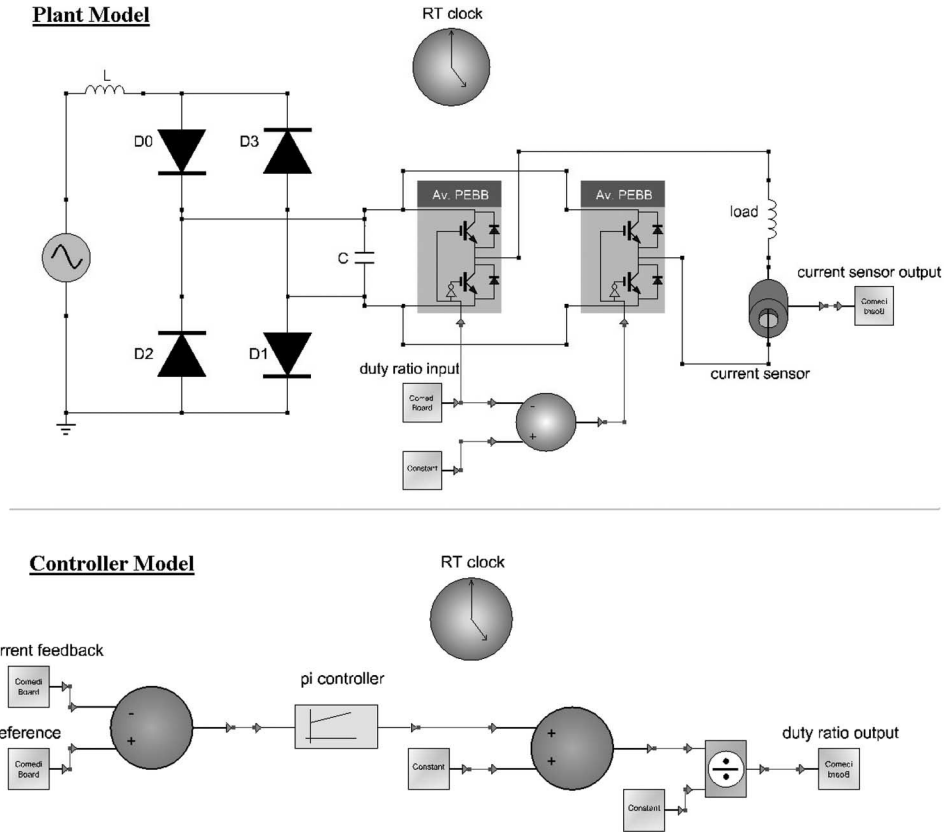

Fig. 15. Subsystem models of plant and controller in VTB (exported into two separate VTB-RT host computers).

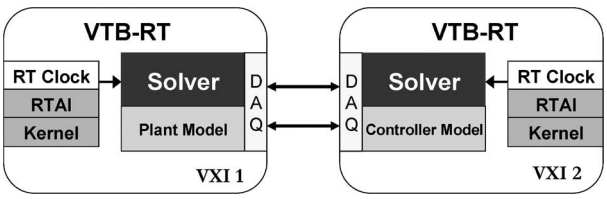

Fig. 16. Architecture of VTB-RT real-time distributed simulation.

VTB-RT in this example within each simulation time step. In this example, there are components (voltage source, diodes, switches, resistor, and inductor) that are solved using the RCM method and other components (sensors, math functions, and DAQ interface models) using the SRC method. As shown in Table VI, most of the simulation time is spent in solving and interfacing the SRC models. At this stage, the VTB-RT presents undesirable overhead that counts at least for 30 μs. In addition, 33 μs is used for outputting and logging the simulation results; this time can vary depending on how many variables the user prints in a log file at every time step. Minimizing data logging processes and using the advanced VTB-RT platform listed in Table IV, a time step of 100 μs has been achieved for this experiment.

VII. Conclusion and Recent Progress

In this paper, a multisolver hard real-time HIL simulation environment, namely, the VTB-RT, is introduced. Using this platform, a complete design and testing procedure for power electronics controls is proposed. The applicability of the VTB-RT and the proposed testing approach is validated through two application examples.

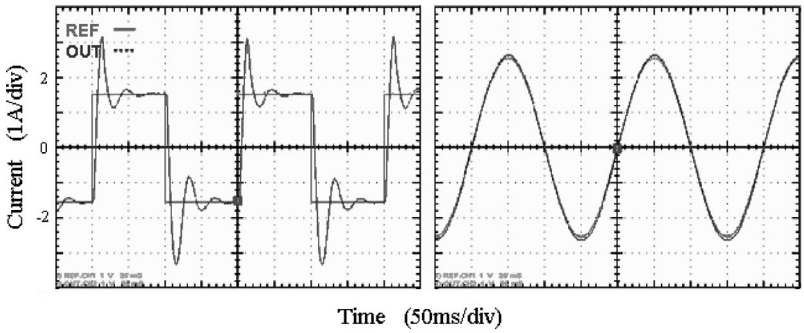

Fig. 17. Output currents and current references of the real-time distributed simulation in VTB-RT (real-time distributed simulation case).

TABLE VI
VTB-RT TIME ANALYSIS IN THE H-BRIDGE INVERTER CONTROL EXAMPLE

	Time Consumption
Solving system equation	
- RCM	71 μs
- SRC	116 μs
Output and logging	33 μs
Additional overhead	30 μs
Minimum time step	**250 μs**

The proposed real-time HIL testing approach eliminates the need of using costly hardware in the design process and, therefore, greatly reduces the overall design cost. However, for a given hardware platform, the minimum time resolution is limited and only a limited bandwidth of the system under test can be used. Consequently, in the actually design stage, special attention must be paid to compromise between the VTB-RT platform cost and the bandwidth of the system under test.

The most attractive feature of the VTB-RT is that it has a very low cost, yet it is capable of yielding comparable performance as the expensive commercial RTS software. As shown in the first example, hard real-time could be realized without advanced hardware. Certainly, however, the minimum real-time step that can be achieved depends on the hardware configuration of the VTB-RT platform (i.e., the speed of the processor, the sampling frequency of the DAQ devices, etc.), as well as the complexity of the system under test. In the first example, the VTB-RT platform is based on a standard personal computer with an Intel PIII 800 MHz CPU and 128 MB SDRAM, and the minimum step that can be achieved to simulate the boost converter control system in hard real time is 300 μs. In the second example, the VTB-RT platform is based on a National Instruments VXI-872Bpc controller with an Intel PIII 1.266 GHz CPU and 128 MB SDRAM, and the minimum time step is reduced to 250 μs for an even more complex H-bridge inverter control system. The time resolution of the VTB-RT can be improved by using faster processors and DAQ devices. If more advanced hardware is used, the minimum step can be further reduced to 50 and 100 μs, respectively, for the same experiments. These resolutions are good enough to simulate the dynamic behavior of most power electronics control systems and are comparable with the commercial real-time systems. Additionally, a high-speed device such as a field programmable gate array (FPGA) connected to the VTB-RT platform can improve the performance of the overall simulation process, providing means to accurately acquire and generate high-frequency signals [15].

The SRC solver used by the VTB-RT enables the natural coupling between the simulation environment and the hardware under test and, thus, enables the power-hardware-in-the-loop (PHIL) testing, where virtual power exchange between the simulation environment and the hardware under test is possible. The theoretical framework and experimental results of the PHIL testing using VTB-RT are reported in [13]. The most recent research advances in the VTB-RT has successfully extended VTB-RT and the proposed HIL testing approach to more complex systems such as three-phase systems and motor drive systems [14]. These applications usually include a switched circuit that requires high- and variable-frequency switching gating signals, for example, fast-switching PWM signals and encoder signals. This poses a major challenge to the simulation platform due to the high time resolution required. An FPGA-based signal conditioning hardware interface has been recently adopted in VTB-RT as the I/O interface to the hardware under test [15]. It greatly reduces the computational burden of the simulation in the acquisition and generation of fast switching signals. These recent advances have greatly extended the design and testing capability of the VTB-RT from mere control system to almost any hardware system.

APPENDIX
IMPLEMENTATION PROCEDURE OF THE VTB-RT

In the VTB-RT, the software packages need to be installed in the following sequence: Linux, RTAI, Comedi, and the VTB-RT solver. Fig. 18 shows the VTB-RT implementation procedure, including these four components.

This implementation procedure can be summarized into four major steps.

1) *Install any Linux distribution and update a clean RTHAL-enabled generic Linux kernel.* This step creates the basic software environment for the VTB-RT, including user interface and general purpose applications.
2) *Install and configure RTAI.* This step enables the hard real-time capability of the VTB-RT. It is the most important step.
3) *Install and configure Comedi and Comedilib.* This step provides the VTB-RT the ability to interface with the real hardware and, thus, makes the HIL simulation possible.

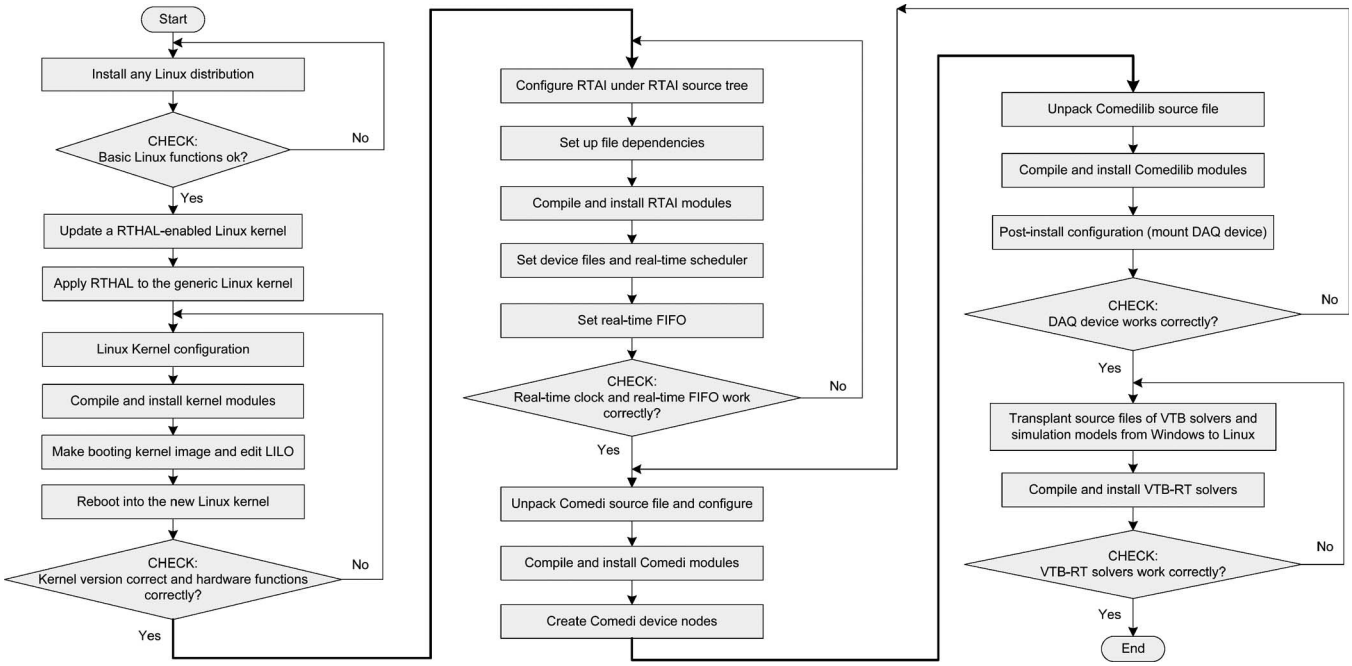

Fig. 18. Complete installation procedure of the VTB-RT.

4) *Transplant VTB solvers and simulation models from the VTB to the VTB-RT.* This step replicates all the functionality of the VTB into the VTB-RT, including the simulation solvers and models.

For simplicity, the detailed information of each block in Fig. 18 is omitted here. The complete installation procedure is minutely explained in [12].

REFERENCES

[1] B. Lu, A. Monti, and R. Dougal, "Real-time hardware-in-the-loop testing during design of power electronics controls," in *Proc. IEEE IECON*, Nov. 2003, vol. 2, pp. 1840–1845.
[2] S. Lentijo, A. Monti, E. Santi, C. Welch, and R. Dougal, "A new testing tool for power electronic digital control," in *Proc. IEEE PESC*, Jun. 2003, vol. 1, pp. 107–111.
[3] V. Dinavahi, R. Iravani, and R. Bonert, "Design of a real-time digital simulator for a D-STATCOM system," *IEEE Trans. Ind. Electron.*, vol. 51, no. 5, pp. 1001–1008, Jun. 2004.
[4] H. Li, M. Steurer, K. Shi, S. Woodruff, and D. Zhang, "Development of a unified design, test, and research platform for wind energy systems based on hardware-in-the-loop real-time simulation," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1144–1151, Jun. 2006.
[5] F. Piazza, S. Squartini, R. Toppi, M. Navarri, M. Pontillo, F. Bettarelli, and A. Lattanzi, "Industry-oriented software-based system for quality evaluation of vehicle audio environments," *IEEE Trans. Ind. Electron.*, vol. 53, no. 3, pp. 855–866, Jun. 2006.
[6] P. Baracos, G. Murere, C. Rabbath, and W. Jin, "Enabling pc-based HIL simulation for automotive applications," in *Proc. IEEE IEMDC*, Jun. 2001, pp. 721–729.
[7] M. Park and I. Yu, "A novel real-time simulation technique of photovoltaic generation systems using RTDS," *IEEE Trans. Energy Convers.*, vol. 19, no. 1, pp. 164–169, Mar. 2004.
[8] C. Dufour and J. Belanger, "Discrete time compensation of switching events for accurate real-time simulation of power systems," in *Proc. IEEE IECON*, Nov./Dec. 2001, vol. 2, pp. 1533–1538.
[9] I. Etxeberria-Otadui, V. Manzo, S. Bacha, and F. Baltes, "Generalized average modelling of FACTS for real time simulation in ARENE," in *Proc. IEEE IECON*, Nov. 2002, vol. 2, pp. 864–869.
[10] H. Figueroa, A. Monti, and X. Wu, "An interface for switching signals and a new real-time testing platform for accurate hardware-in-the-loop simulation," in *Proc. 2004 IEEE Int. Symp. Ind. Electron.*, May 2004, pp. 883–887.
[11] A. Monti, E. Santi, R. Dougal, and M. Riva, "Rapid prototyping of digital controls for power electronics," *IEEE Trans. Power Electron.*, vol. 18, no. 3, pp. 915–923, May 2003.
[12] B. Lu, "The real-time extension of the virtual test bed: A multi-solver hard real-time hardware-in-the-loop simulation environment," M.S. thesis, Dept. Elect. Eng, Univ. South Carolina, Columbia, SC, May 2003.
[13] X. Wu and A. Monti, "Methods for partitioning the system and performance evaluation in power-hardware-in-the-loop simulations—Part I and II," in *Proc. IEEE IECON*, Nov. 2005, pp. 251–262.
[14] B. Lu, X. Wu, and A. Monti, "Implementation of a low-cost real-time virtue test bed for hardware-in-the-loop testing," in *Proc. IEEE IECON*, Nov. 2005, pp. 239–244.
[15] H. Figueroa, "Novel interface based on the Firing Signal Averaging method for accurate hardware-in-the-loop testing of digital controllers for power electronics applications," Ph.D. dissertation, Dept. Elect. Eng, Univ.-South Carolina, Columbia, SC, May 2005.

Bin Lu (S'00–M'06) received the B.Eng. degree in automation from Tsinghua University, Beijing, China, in 2001, the M.S. degree in electrical engineering from the University of South Carolina, Columbia, in 2003, and the Ph.D. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 2006, respectively.

From 2001 to 2003, he was with the VTB Group of the University of South Carolina, as a Graduate Research Assistant. While there, he was involved in the development of a real-time virtual test bed for the hardware-in-the-loop testing of dynamic power systems. Since July 2004, he has been working on the application of wireless sensor networks in energy evaluation and condition monitoring of electric machines as a Graduate Research Assistant in the Power Electronics and Motor Diagnostics Group of the Georgia Institute of Technology. Since October 2006, he has been a Lead Engineer at the Innovation Center of Eaton Corporation, Milwaukee, WI. His research interests include electric machines, motor drives and diagnostics, power electronics, modeling and simulation, and application of wireless sensor networks in electric power areas. He has published over 20 conference and journal papers in these areas.

Xin Wu (S'01–M'05) received the B.S. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 1999, and the M.E. and Ph.D. degrees in electrical engineering, from the University of South Carolina, Columbia, in 2003 and 2005, respectively.

She is currently working on simulation and modeling of electromechanical systems for Ansoft Corporation, Pittsburgh, PA. Her research interests include modeling and simulation, hardware-in-the-loop, and control system designs.

Hernan Figueroa (S'98) received the B.S.E.E. degree in electrical engineering from the Universidad Nacional de Ingeniería, Lima, Perú, in 2000, and the Ph.D. degree in electrical engineering from the University of South Carolina, Columbia, in 2005.

From 2001 to 2005, he was with the Department of Electrical Engineering, University of South Carolina, as a Graduate Research Assistant. His research interests include real-time simulation, rapid prototyping of power electronics systems, and intelligent control. He is currently a Post-Doctoral Fellow at the University of South Carolina.

Dr. Figueroa received a student award at the 2004 IEEE International Symposium on Industrial Electronics.

Antonello Monti (S'88–M'94–SM'02) received the M.S. and Ph.D. degrees in electrical engineering from the Politecnico di Milano, Milan, Italy, in 1989 and 1994, respectively.

From 1990 to 1994, he was with the Research Laboratory of Ansaldo Industria of Milan, where he was responsible for the design of the digital control of a large power cycloconverter drive. In 1995, he joined the Department of Electrical Engineering, Politecnico di Milano, as an Assistant Professor. Since August 2000, he has been an Associate Professor in the Department of Electrical Engineering, University of South Carolina, Columbia. He is the author or coauthor of more than 150 papers in the field of power electronics and electrical drives.

Dr. Monti is serving as the Chair of the IEEE Power Electronics Committee on Simulation, Modeling and Control and as an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.