WHITE PAPER

Hardware-in-the-Loop Testing for Power Electronics Control Design

Featuring Simulink and Speedgoat real-time target machines

MathWorks®

speedgoat

real-time simulation and testing

Introduction

Growing demand for hybrid and electric vehicles, advances in renewable energy, and cross-industry calls for more efficient, higher-performance electric motors are driving the need for more complex power electronics control systems and the embedded software that enables them. The adoption of wide-bandgap semiconductor materials brings faster switching frequencies, compounding the challenge of developing embedded software for these systems. To address these challenges, engineers use system-level simulation in which the performance of closed-loop and supervisory control algorithms is evaluated using a plant model that represents the electrical behavior of the active and passive circuit components.

This white paper focuses on an important step in the engineering workflow that comes after system-level simulation but before committing to a final hardware design: real-time testing with hardware-in-the-loop (HIL) simulation. For power electronics–based systems, such as a solar inverter or motor drive, HIL simulation is used to validate the system's digital controller in real time against a deterministic model of the electrical system, including power sources and loads. Throughout the paper, references to ***Simulink***® software and ***Speedgoat*** real-time hardware systems are used to illustrate various aspects of HIL simulation. Simulink is desktop software for system-level simulation of control systems, capable of generating both C and HDL code that can be used in real-time testing. Speedgoat real-time hardware systems are computers for running code from Simulink models using ***Simulink Real-Time***™.

From Desktop Simulation to HIL Simulation

When you perform ***power electronics control design in Simulink*** using a system-level block diagram model, you can construct multilevel feedback and supervisory control algorithms that operate at multiple sample rates. To model the power electronics system to be controlled, you may choose to include components from ***Simscape Electrical***™, which builds on Simulink with a library of prebuilt sources, loads, and circuit elements (Figure 1). You can modify these components, build your own from first principles or test data, or import design constructs from other simulation software. Depending on your design objective, you may want to selectively increase model fidelity to improve simulation accuracy or decrease it to improve simulation speed.

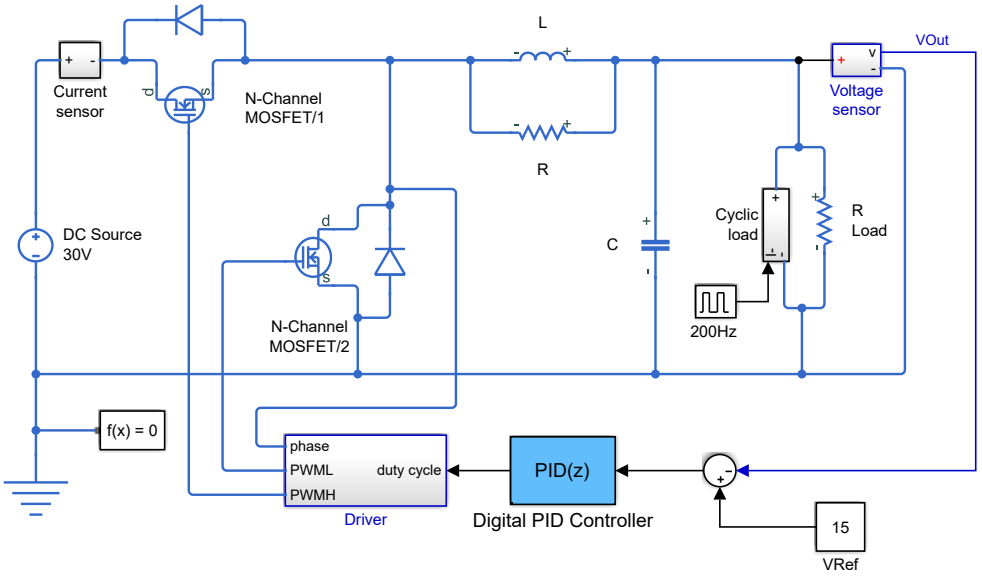

Figure 1. Simulink model of a buck converter.

MathWorks®

> *"Running simulations of the plant models we built with Simulink and Simscape Electrical enabled us to improve and optimize our hardware design before it was finalized. The simulations eliminated several iterations of pre-prototype hardware."*
>
> **— Dr. Jakub Vonkomer, VONSCH**

» ***Read story***

System-level simulation on a desktop computer is an excellent way to verify the interaction of power electronics devices and the algorithms controlling them under a wide range of scenarios, including failure situations that could potentially damage power electronics. However, it does little to validate the real-time behavior of the embedded software running on the actual controller under normal and abnormal conditions.

One option for validating the real-time behavior of the controller is to hand-write code for the control algorithms, compile the code and deploy it to a microcontroller, and test the controller as part of a real power electronics system. While this type of controller validation should always be performed, it is not advisable to delay all real-time testing until the latter stages of a development program when software-hardware integration is complete. Defects found at these late stages are notoriously more difficult and more expensive to remedy.

A second option is to validate the controller against a system model (plant) running on a real-time computer. This HIL simulation can be viewed as extending desktop simulation to real-time simulation and can be performed even before power electronics prototype hardware is available for testing.

Starting with a system model comprising a plant and controller, you generate C or HDL code from the portion of the model describing the plant (Figure 2). The code generated from the plant model is implemented on the real-time computer that executes the code fast enough to simulate the real-time behavior of the actual plant hardware. Real-time target machines from ***Speedgoat***, for example, use ***Simulink Real-Time*** to execute code generated from Simulink. With this real-time representation of your system, you can test the embedded control software running on a microcontroller, an FPGA, or another real-time test system. With HIL simulation, you can also inject faults to evaluate the controller's ability to maintain optimal and safe operation under adverse conditions.

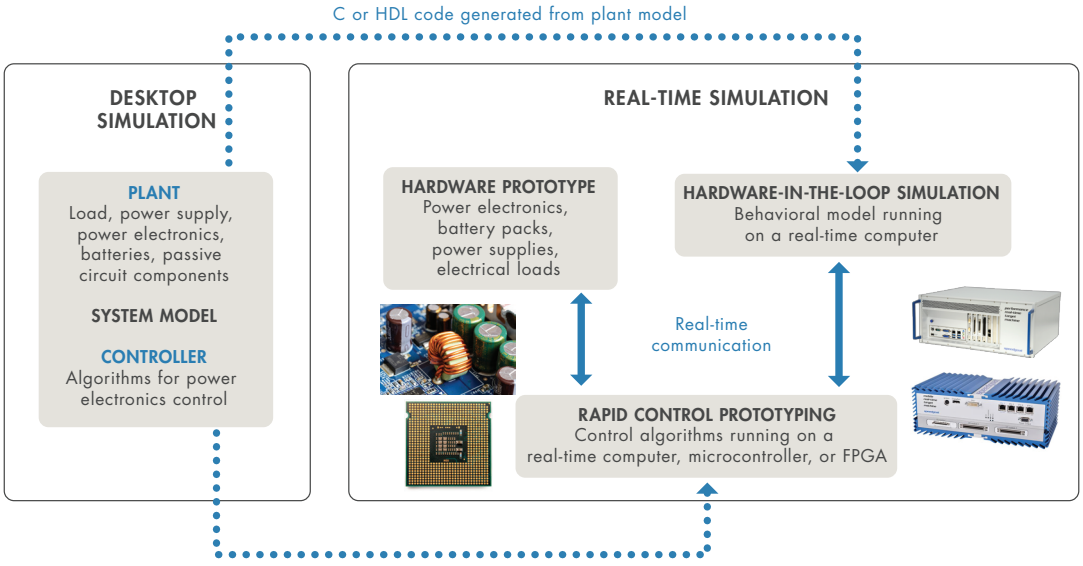

Figure 2. Using code generated from the desktop simulation model for rapid control prototyping and HIL simulation.

MathWorks®

"Simulink, Simulink Real-Time, and Speedgoat target hardware have enabled us to demonstrate the application-specific reliability of our early designs without installing the units on an actual aircraft. With Model-Based Design, we can do continuous validation and verification without waiting until all aspects of the power electrical control unit are developed."

— Shane O'Donnell, Microsemi

» ***Read story***

When developing HIL simulations for power electronics systems, take time to consider the following factors early in your project:

- Tradeoffs between simulation speed and model complexity
- Processor and latency requirements

These factors will shape how you model the power electronics control system and what real-time test hardware you choose to validate your controller performance.

Trading Off Simulation Speed and Model Complexity

As a starting point, consider the speed at which a controller must respond to ensure stable and safe operation of the system. Figure 3 shows a range of physical systems and their respective response times; note that power electronics controllers typically must sample on the order of microseconds or faster. However, the speed required for your HIL simulation will depend on the dynamics that you need to simulate in the system model.

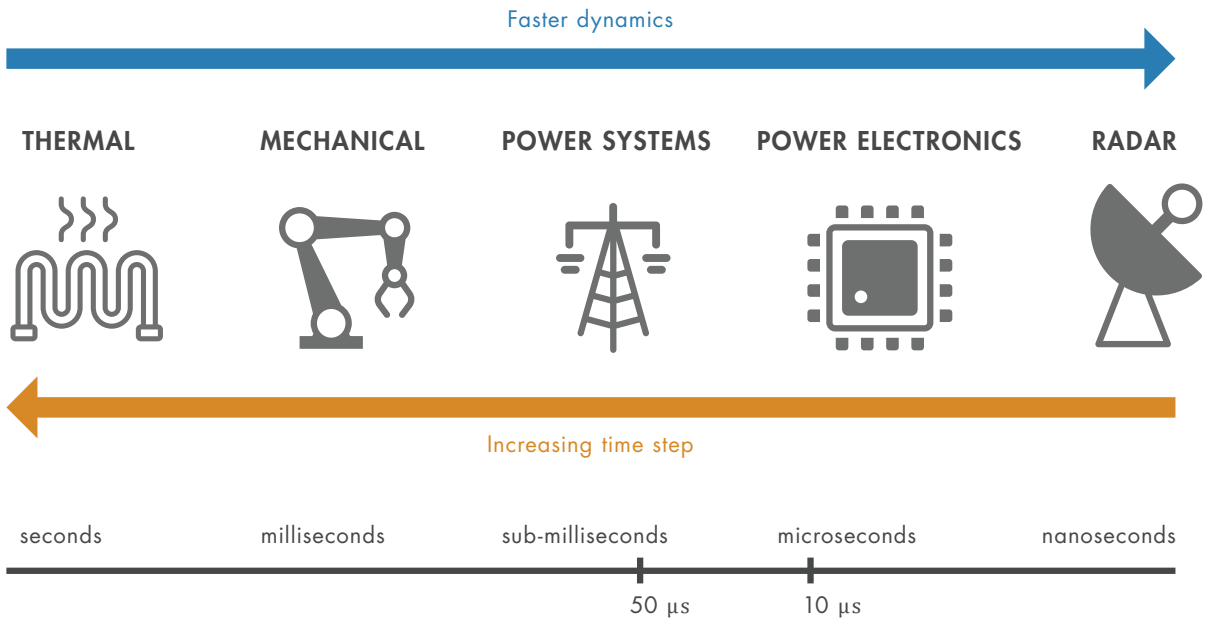

Figure 3. Relationship of dynamics and time step.

In general, a model described by simpler dynamics will simulate faster, though accuracy will not be maximized. A model with more complex dynamics will simulate slower, but will provide a more accurate representation of the true power electronics behavior. With Simulink you can ***choose from an extensive set of fixed-step or variable-step simulation solvers***. For desktop simulations, variable-step solvers adjust step size time to ensure accuracy. In models with simpler dynamics, the variable step sizes can be larger, letting the model compute faster.

For example, when designing a field-oriented control algorithm for a permanent magnetic synchronous motor (PMSM), the inverter power electronics voltage output can be modeled as an average value voltage of the power transistor duty cycle. The motor can be modeled as a set of linear differential equations with lumped parameters. The simulation will not capture the effects of switching behavior, but it will be suitable for tuning the control algorithms and testing the control under nominal operating conditions. Because it uses a simpler mathematical representation of the power electronics behavior and PMSM, the simulation will run faster than a version based on a model that reflects the detailed switching of the power electronics and a nonlinear model of the PMSM.

Now imagine that the PMSM will be used in the powertrain of an electric vehicle where all electrical losses drain charge from the battery pack and reduce range. In this case, you may want the inverter model to include the power electronics switching behavior and effects of temperature on electrical efficiency. Also, you may want to include a nonlinear model of the PMSM imported from finite element analysis (FEA) software, such as ***JMAG***, that captures magnetic saturation and spatial harmonics between the rotor and stator. This plant model is mathematically more complex than the one with the linear motor model and an average value inverter. As a result, the desktop simulation will run slower, with the solver taking small steps to capture the more detailed dynamics described in the model.

Once you have a satisfactory desktop simulation running, the next step is real-time simulation. This means generating executable code from the plant model and deploying it to a target computer. For real-time testing, Simulink can generate C code via ***Simulink Coder***™ and HDL code via ***HDL Coder***™. The code is hardware independent and can be implemented on any real-time hardware. ***Using Simulink Real-Time with Speedgoat hardware*** streamlines the process of targeting C or HDL code and makes it easier to instrument and automate real-time testing.

> *"The Speedgoat target machine provides fast and robust control of the switching semiconductors in a difficult electromagnetic environment (E = 1V/m)."*
>
> **— Piotr Dworakowski, SuperGrid Power Converters team leader**

» ***Read story***

Addressing Processor and Latency Requirements

The complexity of a system's dynamics not only affects how fast a desktop simulation will run. It also will influence your choice of hardware for HIL simulation. The model time step for a real-time simulation must be larger than the system latency of the real-time hardware to avoid task overrun errors (Figure 4). System latency depends on two factors:

- Software latency from the application running on the processor
- Hardware latency from input or output interfaces

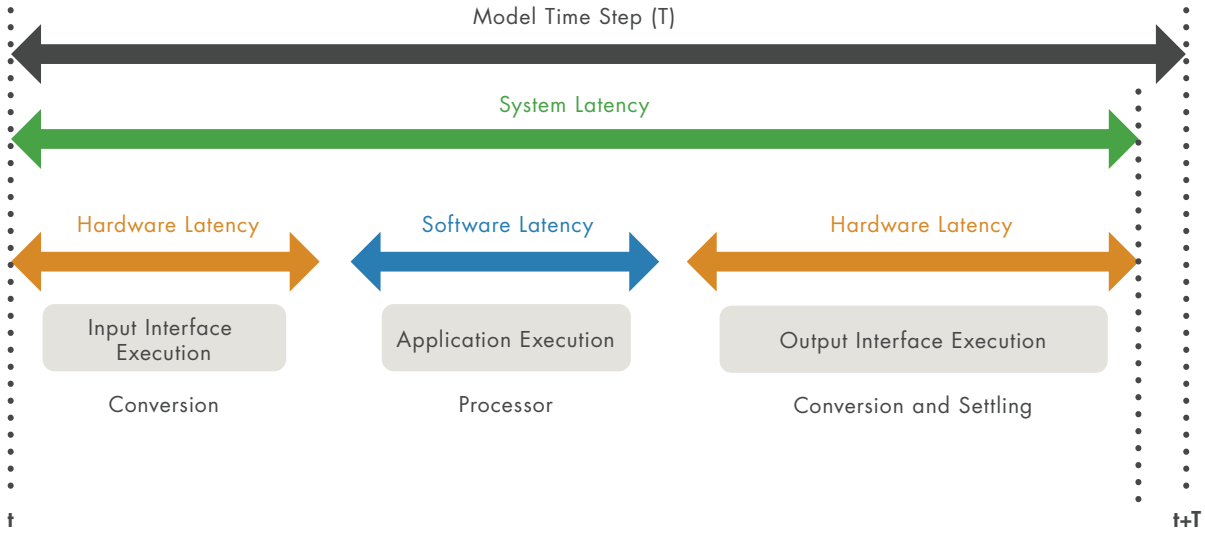

Figure 4. Factors contributing to system latency.

Software latency is driven by the computational complexity of the application and computing power of the processor used. You can reduce software latency by employing faster CPUs or FPGAs and applying concurrent execution (running tasks on multiple CPU cores). With a combination of multicore processors and FPGAs, you can partition the system to execute subsystems with slower dynamics and data logging on the CPUs while running subsystems with faster dynamics and signal conditioning on the FPGAs.

Consider the previous case of a PMSM involving power electronics switching behavior. Conservatively, you want the HIL simulation to run at a sampling frequency 100 times greater than the expected switching speed of the actual system. For example, if the PMSM will run at a switching frequency of 10 kHz, the HIL simulation needs to run at 1 MHz or faster to capture the effects caused by switching devices. For a HIL system needing to run at this frequency, you may want to consider an FPGA-based real-time system to accurately represent electronics switching behavior. While experience shows that FPGA hardware may be needed for time steps shorter than 50 microseconds (20 kHz), it may be possible to use CPU-based simulation down to 10 microseconds (100 kHz) depending on model complexity and I/O latency. The decision to use a CPU or alternatively an FPGA will be driven by the mathematical complexity of the model and the number and type of I/O channels. For power electronics HIL simulation, many organizations are moving toward FPGA-based systems for simulation of switching systems. This trend is being reinforced by the use of silicon carbide (SiC) and gallium nitride (GaN) semiconductors in power electronics systems to enable higher operating frequencies with increased control system clock speeds.

Most real-time hardware vendors, including Speedgoat, offer both FPGA- and CPU-based systems. With Speedgoat and Simulink Real-Time, you can partition your model and run the different parts on multiple CPU cores (via ***concurrent execution***) and on one or more Simulink programmable FPGAs.

Hardware latency depends on the I/O signals used. Digital I/O tends to have lower latencies than analog I/O. On an FPGA, digital I/O can have latencies below one nanosecond. For a CPU-based system, in contrast, digital I/O is in the microsecond range. When using analog signals and communication protocols, hardware latency depends on the analog-to-digital converter (ADC) technology, signal triggering options, and, for ADCs with sequential sampling, the number of channels being measured. You can expect acquisition rates to range from thousands to millions of samples per second.

Hardware latency is also affected by the communication between the processor and its I/O channels. A simulation of a full switching inverter and nonlinear motor models running on a CPU that communicates with I/O via a bus, such as PCI Express, will execute at a time step of 20 microseconds or more depending on the number of channels. A portion of the time step is caused by hardware latency associated with the PCI Express bus. In comparison, the same application running on an FPGA with onboard configurable I/O removes the bus latency and will execute with a time step of around 1 microsecond. As another comparison, connecting multiple FPGAs with a low-latency bus, such as Aurora 64B/66B serial communication, adds a communication latency between them of about 1 microsecond.

Scheduling the parallel execution of input/output and application execution can also reduce latency. For example, using direct memory access (DMA) for analog inputs and outputs can significantly reduce system latency by parallelizing the model execution, data acquisition, and signal writing (Figure 5).

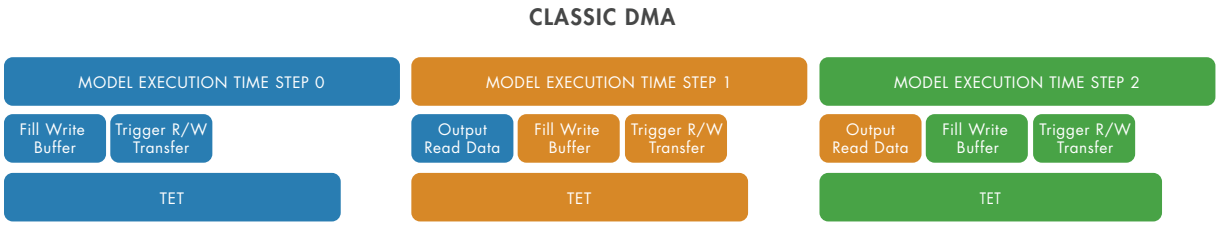

Figure 5. Using DMA to enable concurrent execution of models and I/O transfers.

Power HIL Simulation

Until this point, the HIL scenarios discussed have focused on developing a real-time model for validating a controller to be used in a power electronics–based system. The plant, consisting of active and passive circuitry, loads, and supplies, runs as a real-time model on a target computer. The controller, or device under test (DUT), is tested under normal and fault conditions, making it possible to identify most control algorithm bugs and issues at an early stage without damaging power equipment. This type of HIL simulation, sometimes referred to as *control HIL*, is illustrated in Figure 6. Here, the real-time system simulates the photovoltaic (PV) cells, inverter electronics, and grid and provides the same analog, digital, and protocol interfaces as the real plant.

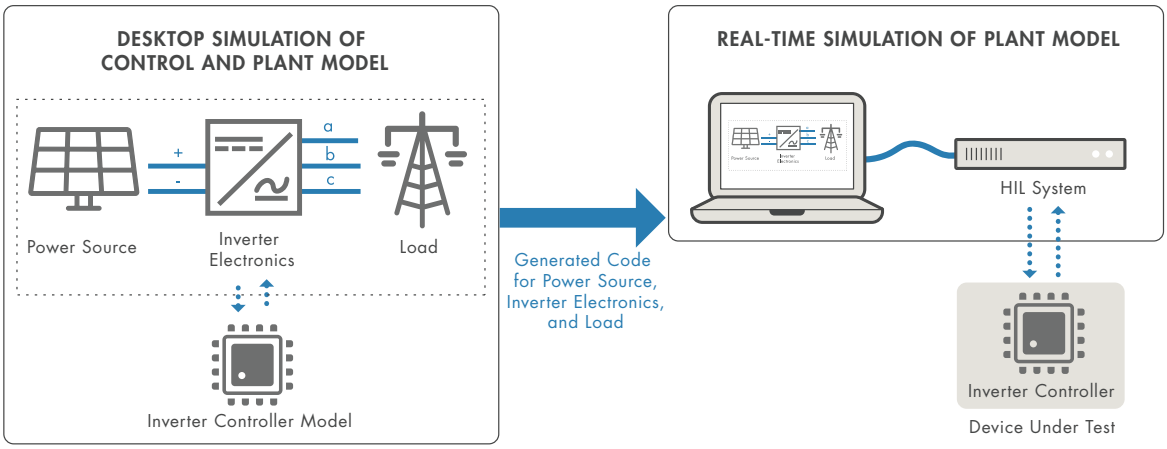

Figure 6. Control HIL using real-time simulation to test a controller.

In contrast to *control HIL* where the DUT is simply the controller, *power HIL* involves a more complex DUT that incorporates the controller as well as hardware components. In a power HIL simulation, the DUT connects to the real-time simulation via power amplifiers.

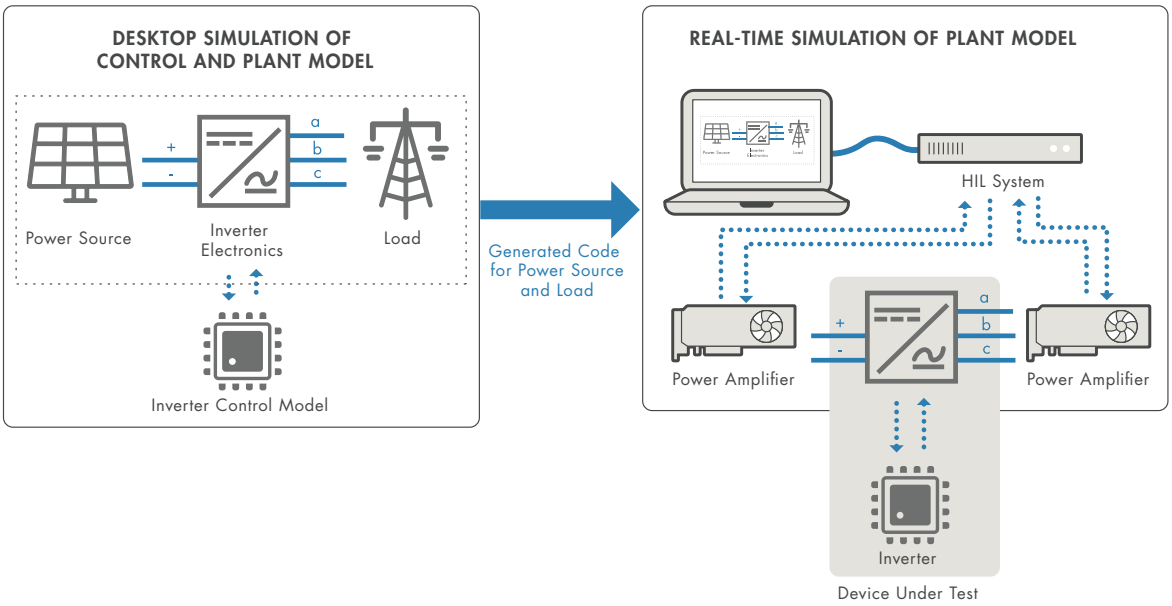

Figure 7. Power HIL using power amplifiers connected between the real-time simulation and device under test.

In the power HIL example shown in Figure 7, the DUT includes the inverter (inverter electronics and controller). The inverter interfaces are a DC voltage connection to the PV system and 3-phase AC voltage connection to the power grid. To provide the necessary DC and AC power levels, high-speed AC and DC power amplifiers are connected between the DUT and the real-time computer. The real-time computer still simulates the PV system and power grid connection, providing high-speed signals to drive the DC and AC power amplifiers.

Other power HIL applications include electrical powertrains testing, microgrid emulation, and battery management system testing.

HIL Workflow Considerations

In evaluating solutions for HIL testing of power electronics control designs, it is important to consider the complete development workflow from desktop simulation to real-time simulation. Otherwise, you may end up spending more time getting software and real-time hardware products to work together than conducting tests. Another important consideration is the fidelity of the simulation model that represents the power electronics control system. For example, the system-level simulation software should offer a comprehensive set of control design capabilities, both for feedback and supervisory control. Likewise, the electrical modeling library should provide components that can cover different levels of model fidelity. The library's insulated-gate bipolar transistor (IGBT), for example, should have model variants that represent ideal behavior on one end of the spectrum and physics-based characteristics on the other. Lastly, simulation software that generates portable code is essential if you want the option to deploy your simulation model to a variety of real-time hardware targets.

MathWorks and Speedgoat jointly develop Simulink Real-Time, the real-time operating system from MathWorks. As a result of this close cooperation, Simulink and Simscape Electrical models used for desktop simulation can be reused for real-time testing with CPUs and FPGAs using Speedgoat real-time target machines. Speedgoat products also offer a wide range of I/O connectivity and communication protocols for power electronics, such as:

- High-speed analog inputs and outputs
- Fiber-optic links
- High-speed digital inputs
- Communication protocols, including EtherCAT, CAN with Flexible Data Rate, and PROFINET

For more information on MathWorks and Speedgoat solutions for HIL testing of power electronics control designs, ***contact sales***.

© 2019 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks.
Other product or brand names may be trademarks or registered trademarks of their respective holders.