Hardware-in-the-Loop Technology Enabling Flexible Testing Processes

Andreas Himmler[1]
dSPACE GmbH, 33102 Paderborn, Germany

Hardware-in-the-loop (HIL) testing is indispensable in the control unit software development process and has been an integral part of numerous companies' development processes for years. HIL simulation is regarded as the tried-and-tested method for function, component, integration and network tests for the entire system. This holds especially for model-based design. Ever since HIL testing has started to be applied, users have continuously streamlined their development processes. This paper describes the changing requirements for HIL simulation, and how these changes need to be addressed by a new HIL technology.

Nomenclature

HIL	=	Hardware-in-the-Loop
ECU	=	Electronic Control Unit
IOCNET	=	I/O Carrier Network
I/O	=	Physical input/output channels of electronic control units, devices, etc.
CAN	=	Controller Area Network
LIN	=	Local Interconnect Network
UART	=	Universal Asynchronous Receiver/Transmitter
OS	=	Operating System

I. Introduction

TODAY, the usual case is specialization among the members of HIL teams, in other words, the division of tasks. There are tasks to design the electrical aspects of a simulator (HIL hardware, connecting real and substitute loads, cable harnesses), others to model the I/O and plant, and other to create and execute tests. Thus, today's HIL technology must enable different team members to work simultaneously on different tasks in the same project. For example, while one team member is setting up the I/O configuration of the HIL system another team member is setting up the plant model. This separation of tasks also supports confidentiality, if required.

Another key factor for today's HIL systems is the total cost of ownership, due to tight budgets and tough competition. This is especially true in the context of the ambitious technical requirements named above. A HIL simulator needs to flexible and quickly adaptable to new demands. This applies both to the hardware and to the necessary configuration software.

Other factors such as flexibility and adaptability also come into play. Flexibility is necessary so that different control units or variants can be tested on a single system, either individually or in a network. This is particularity important when entire vehicle electronics are tested because HIL systems often have to be specified and set up at a stage when parts of the final ECU specifications might still be changed. Adaptability is needed because the HIL system must be able to handle executing any additional tests for new components added to an existing vehicle (airplane, satellite, etc.). In addition, when one project has been successfully finished, it is becoming more and more common to reduce overall costs by completely converting the systems in order to test the next project. Finally, a HIL system is often used for network testing as well as for dedicated tests of single components: This occurs when a section of a HIL tester has to be isolated to intensify troubleshooting for a single ECU, and when the component testers are interconnected after successful component tests so that network tests can be run.

[1] Product Manager Hardware-in-the-Loop Simulators, Product Management Department, Rathenaustrasse 26, 33102 Paderborn, Germany.

This paper discusses the above requirements and the dSPACE HIL system setup that was designed ground-up in consideration of these needs, resulting in an architecture that provides full flexibility, adaptability and process efficiency to the embedded software development and testing process.

In section II of this paper the requirements for new HIL technology are summarized. The subsequent sections III and IV describe the hardware and software of the new HIL technology that was developed to fulfill these requirements. In sections V and VI the openness of the new technology is described by using two examples: the integration of 3rd-party hardware, and co-simulation.

II. Requirements for a New HIL Technology

A. Changed Workflows: Task Division

Previously, often just one engineer was responsible for the whole HIL workflow, from design to test execution. Now today, most companies have set up central HIL departments, though sometimes the individual engineering departments are taking care of their own HIL systems. In either case, teams solely dedicated to HIL are typical. They are responsible for planning the electrical aspects of a simulator (HIL hardware, connecting real and substitute loads, cable harnesses), for modeling the I/O and plant, and for creating and executing tests. Having dedicated teams means that the team members can specialize, and that task division is the usual case. HIL systems need to support this by allowing working practices such as:

- Early project planning and configuration, even without available simulator hardware
- Separating simulator configuration and plant modeling (also separate code generation, which saves time for individual users)
- Exchange of models between component tests and integration tests
- Automated model generation and parameterization (in-house databases can be used to put together models and parameterize vehicle data)
- Using identical HIL systems and tools by suppliers and OEMs to simplify the exchange of models, parameters, hardware configurations, tests and test results
- Worldwide use of test platforms (requiring multi-user or remote access to a distant simulator, and manual setup of the simulator for different test cases by an on-site engineer)
- Linking tests to ECU requirements: i.e., dedicated interaction between the test tool and requirements management tools such as IBM Rational Doors and MKS Integrity.

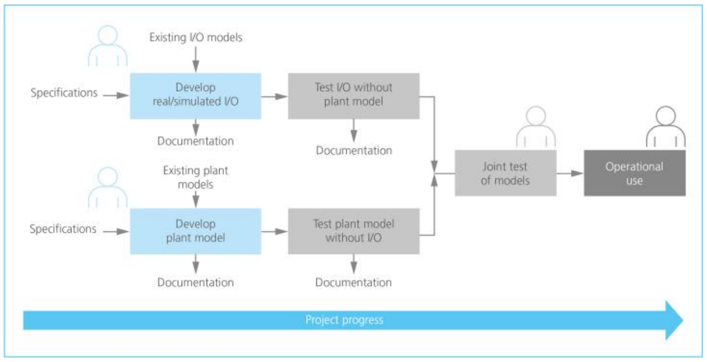


Figure 1. Division of tasks.

B. Flexibility, Adaptability and Openness

Flexibility is needed if different control units or variants have to be tested on one system, either individually or in a network. When the entire vehicle electronics are tested, it often happens that the HIL systems have to be specified and built at a point in time when parts of the final ECU specifications might still be changed. A further common requirement concerning the adaptability of a HIL system is to execute additional tests for new components added to an existing vehicle (airplane, satellite, etc.). For reasons of cost, when one vehicle has been successfully released, it is becoming more and more common to completely convert the systems in order to test the next vehicle. Moreover, a HIL system is frequently used both for network testing and for dedicated tests of single components. This may be because a section of a HIL tester has to be isolated in order to intensify troubleshooting for a single ECU, or it may be that after successful component tests, the component testers are interconnected to run network tests.

Openness is important especially for aerospace users. They require that 3rd-party hardware and in-house hardware can be used efficiently with a HIL system. This requires not only the ability to "use" such hardware with a given system. Efficient use of 3rd-party and in-house hardware of course also requires that the workflow the system enables is maintained and used by fulfilling the requirements listed above. This also requires interfacing with 3rd-party tools.

Another aspect of openness is the ability to run on the HIL system software in a different operating system than the OS of the HIL system. This holds, for instance, if certain simulation software for co-simulation is available only for the HIL systems' real-time operating system (RTOS).

C. New HIL Technology: SCALEXIO

Systematic implementation of the above requirements involves some far-reaching changes in existing hardware and software concepts. A completely new HIL technology was therefore developed, based on a completely new hardware and software architecture in order to be able to fulfill these requirements. This new HIL technology is named SCALEXIO. Its most prominent features are aspects such as channel flexibility, granular extensibility and complete software configuration.

In the subsequent sections, the hardware and software of the new HIL architecture is described, as well as the HIL architecture's capability to use 3rd-party hardware and set up co-simulation.

III. New HIL-Technology – Hardware

The new HIL technology is based on a completely new hardware and architecture developed from the ground up. Essential parts are the newly-developed high-speed serial I/O network and the new, completely software-configurable I/O boards. The real-time processor of the new HIL technology is based on an industry PC.

A. Real-Time Processor

The real-time processor is the heart of SCALEXIO. A SCALEXIO processor core is based on an industry PC with an Intel® Core™ i7 quad-core processor, a real-time operating system (RTOS), and a PCIe plug-on card designed by dSPACE for communication with the I/O and with further real-time processors. It sets up a connection to the host PC via Gigabit Ethernet, which is used to configure the entire simulator, load real-time applications and, finally, to monitor and control the HIL simulation itself.

Using standard PC technology allows a fast response to growing performance requirements and new technologies on the PC market. Standard PC technology also enables easy integration of 3rd-party hardware into SCALEXIO by utilizing standardized PC interfaces.

B. I/O Network

Communication between real-time processors and I/O boards has a considerable effect on the minimum cycle time of a simulation step. dSPACE has therefore developed the new serial I/O network IOCNET® (I/O Carrier Network), which is being used for the first time in SCALEXIO.

IOCNET is based on the physical layer of Gigabit Ethernet and is integrated into the real-time processor via the PCIe card developed by dSPACE. To guarantee dSPACE's accustomed real-time capability, network protocols for real-time communication and highly-precise time and angle synchronization were developed. Intelligence and preprocessing on the I/O boards reduce the load on the real-time processor.

IOCNET's transmission rate is higher than previously used technologies by a factor of approx. 10. More than 100 I/O nodes can be connected to a real-time processor, and each of these I/O boards can comprise considerably more than 100 I/O channels. Optical media can be used for distances of 100 m and more between the real-time processor and the I/O boards to address aspects of spatial distribution (decentralization), modularity and flexibility.

The SCALEXIO system can interact with all existing dSPACE HIL systems, such as dSPACE Simulator Mid-Size and dSPACE Simulator Full-Size. The existing systems are based on the peripheral high-speed (PHS) bus technology, which is used for communication between the processor and the I/O units. A SCALEXIO system is connected to a PHS system by processor-to-processor coupling with IOCNET Gigalink.

Figure 2 shows the scalability of HIL systems that is enabled by IOCNET and the possibility to couple SCALEXIO systems with PHS systems from dSPACE. It is also shown that I/O units can be set up without a local SCALEXIO real-time processor.

C. I/O Boards

The HIL signals can be roughly divided into four classes: signal generation (simulating sensor signals), signal measurement (measuring actuator signals), bus systems and supply signals. The SCALEXIO technology provides two different, software-configurable I/O board types for each of these four classes (see Figure 1): HighFlex I/O boards and the MultiCompact I/O unit.

What both types have in common is a local PowerPC processor for preprocessing signals and relieving the load on the real-time processor, an IOCNET interface, typical signal conditioning, converters and parts of the electrical failure simulation. The two I/O board types can be combined in any way desired and used both in small component testers and in large network test systems. Integrating the signal conditioning and failure simulation reduces internal wiring and simplifies the technical setup, which makes reuse much easier.

The outstanding features of the HighFlex I/O boards are their flexibility and performance. Each of the signal generation and measurement boards implements 10 galvanically isolated channels, and each channel provides several software-selectable physical interfaces (such as digital, analog, resistance simulation). For example, the bus board has 4 galvanically isolated bus channels. Each of these are software-configurable as CAN, LIN, FlexRay or UART channels and they provide the necessary transceivers and terminations. Thus, simulator project planning with HighFlex I/O boards has to include only the number of channels and not their types. The physical interface that is actually used is configured via software and can be changed as often as required. The result is a very high level of flexibility and reusability.

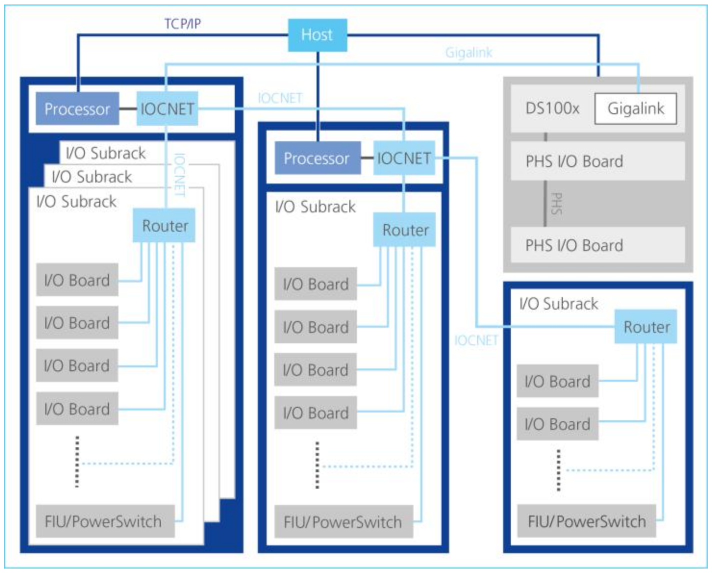


Figure 2. Scalability of HIL systems.

In contrast, the special features of the MultiCompact I/O unit are its large number of channels, its great channel density, and the attractive price per channel. The MultiCompact I/O currently available has more than 150 channels in total, and is galvanically isolated as a unit. For maximum compactness, the unit mainly has dedicated rather than multifunctional channels.

The majority of requirements can be covered by the HighFlex and MultiCompact I/O. But there is a demand to enable the user to react to even tougher requirements: for example, speeding up model parts, using new interfaces, or handling special signal conditioning. These use cases are covered by an FPGA I/O board running user- and project-specific code locally. FPGAs are especially useful for relieving the processor board of tasks such as signal preprocessing during ECU development.

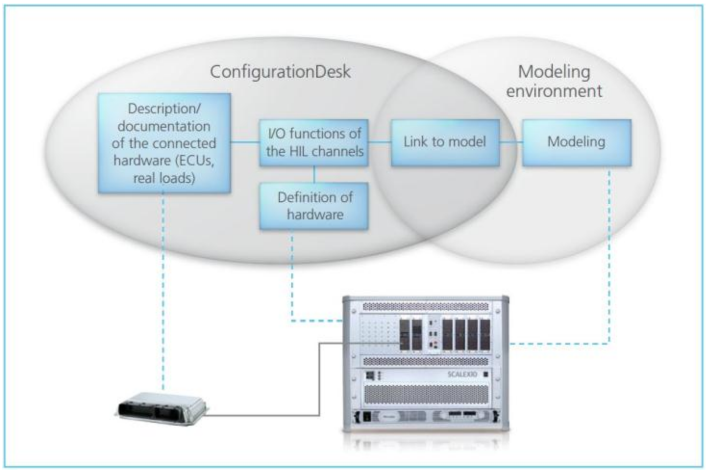

ConfigurationDesk
Modeling environment
Description/ documentation of the connected hardware (ECUs, real loads)
I/O functions of the HIL channels
Link to model
Modeling
Definition of hardware
SCALEXIO

Figure 3. Signal chain on the HIL system.

IV. New HIL Technology – Configuration Software

The versatile configuration options of the SCALEXIO hardware can be accessed via the new ConfigurationDesk® tool. The configuration process is roughly divided into three tasks: describing the externally connected devices (control units, real loads, etc.), selecting the I/O functions for each signal, and linking the I/O functions to the plant model (see Figure 3).

There is no predefined sequence to perform these three tasks. The sequence of work steps depends on the user's workflow. Moreover, not all the elements of the signal chain are required in every case. For example, information on the connected devices is not necessary for HIL simulation itself. A build process can be carried out even if only the

I/O functions and the behavior model are available. This information is primarily necessary for computing the external cable harness, and for documentation. The work steps for configuration can be performed in any order in the clearly organized 3-column display of the tool ConfigurationDesk (see Figure 4).

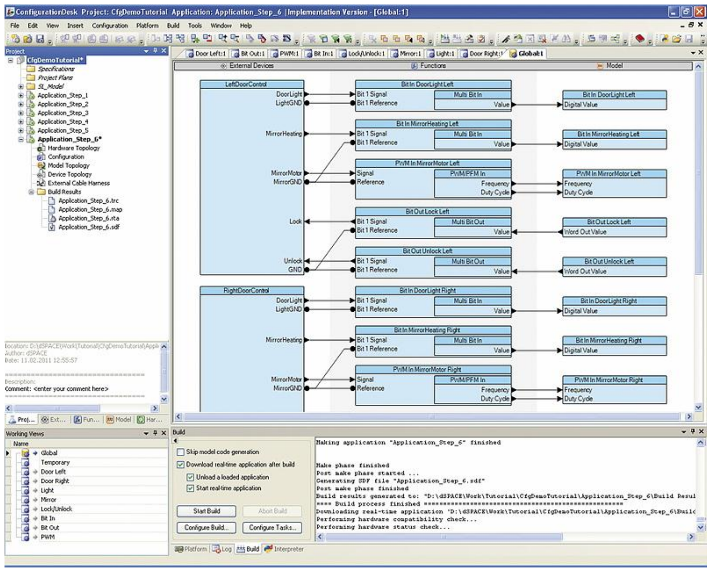
Separating I/O configuration and modeling increases the modularity and reusability of the individual configuration items. It also supports a workflow involving parallel work on different tasks. A great deal of time can also be saved: For example, modification of the I/O configuration only requires generation of new code for the I/O part, but not for the plant model.

The three steps of the configuration process are described in the subsequent subsections.

A. Specification of External Devices

Any HIL project begins with defining the electrical interface of the ECU under test. This requires information on the number of connectors and pins, and also more abstract data such as logical signal names with descriptions, the signal direction (the ECU's input or output), physical signal properties (voltage or current signal), and information on whether a specific electrical failure can be switched to an ECU pin and which loads the ECU expects on that pin.

The left column of ConfigurationDesk contains these definitions. This data can be entered in the tool manually or imported via a Microsoft® Excel® list.



```
Making application "Application_Step_6" finished


Make phase finished
Post make phase started ...
Generating SDF file "Application_Step_6.sdf"
Post make phase finished
Build results generated to: "D:\dSPACE\Work\Tutorial\CfgDemoTutorial\Application_Step_6\Build Resul
==== Build process finished ====================================================
Downloading real-time application "D:\dSPACE\Work\Tutorial\CfgDemoTutorial\Application_Step_6\Build
Performing hardware compatibility check...
Performing hardware status check...
```

Figure 4. ConfigurationDesk screenshot.

B. Interfacing Environment Models

The environment models (plant models) for HIL testing are usually available as MATLAB®/Simulink® models. To connect the I/O to a model, dSPACE provides a Simulink library to be used with ConfigurationDesk. The library contains what are called model port blocks for input, output and trigger signals (see Figure 5). Model ports are shown in ConfigurationDesk in the right column.

Model port blocks can be inserted on any level in the model. If all the signals for I/O connection are on the top model level, standard Simulink In blocks and Out blocks can be used as an alternative. Initially, the signal chain ends at these interface blocks, meaning that no hardware-specific information is placed in the Simulink model. This makes it very easy to reuse the same model in different HIL projects.

How these model port blocks are generated and when this is done during the workflow is described in the following subsection.

C. Specifying Appropriate I/O Functions

I/O functions are used in ConfigurationDesk to link the interfaces of the connected external devices (left column of the tool) to the model port blocks (right column of the tool). The I/O functions are shown in the center column of

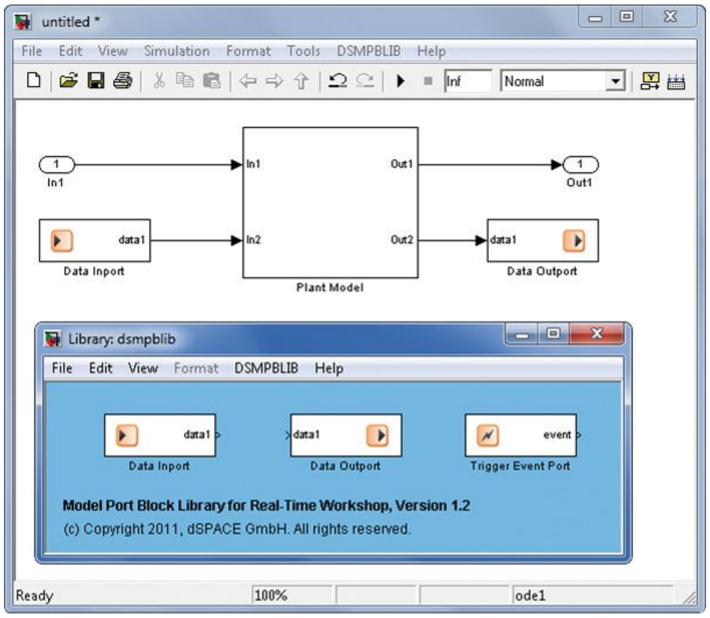

untitled *
File Edit View Simulation Format Tools DSMPBLIB Help
Inf
Normal
In1
Out1
In1
Out1
In2
Out2
data1
Data Inport
data1
Data Outport
Plant Model
Library: dsmpblib
File Edit View Format DSMPBLIB Help
data1
Data Inport
data1
Data Outport
event
Trigger Event Port
Model Port Block Library for Real-Time Workshop, Version 1.2
(c) Copyright 2011, dSPACE GmbH. All rights reserved.
Ready
100%
ode1

Figure 5. Simulink model interface.

ConfigurationDesk.

In order to connect external devices to I/O functions, ConfigurationDesk can make suggestions on suitable I/O functionalities. For example, when an ECU pin provides a voltage signal, the tool suggests an HIL I/O function for voltage measurement. Of course, it is also possible to select I/O functions from a library. The tool also warns if there are any wrong connections such as ECU outputs that are accidentally connected to HIL outputs instead of to HIL inputs.

The I/O functions are connected to the model interfaces (right column) in the next work step. There are two different ways to do this: Either ConfigurationDesk generates a model interface description that fits the I/O functions and exports it to a Simulink model, or you yourself prepare the model in Simulink by inserting model port blocks. A model analysis is then performed to transfer the interface description to ConfigurationDesk.

Initially, the I/O functions define only the functionalities that the simulator has to execute (for example, PWM signal generation), and not the SCALEXIO hardware that will provide that functionality.

In most cases, the same functionality can be provided on different hardware channels or channel types. Therefore, the necessary hardware channels are assigned to the I/O functions in a separate configuration step. This is called hardware resource assignment. This can be done manually, or automatically by ConfigurationDesk.

If signal properties such as the maximum power consumption cannot be handled by a single hardware channel, ConfigurationDesk determines the number of channels required for the channel type and assigns them to the I/O function.

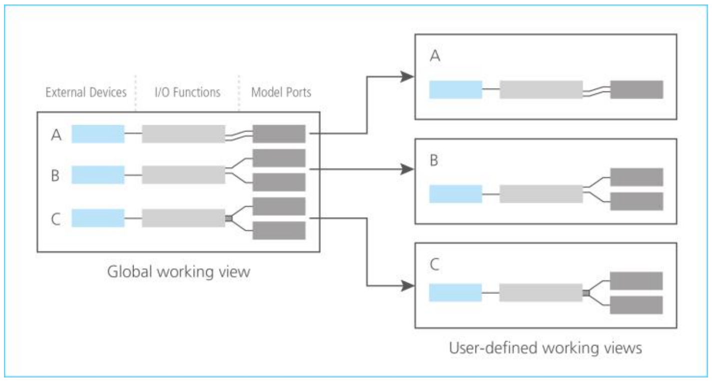


Figure 6. User-defined working views.

When performing the tasks described above users need to be able to concentrate on individual aspects. For example, they might only be interested in the I/O of a specific external device or they may be interested in all voltage measurement channels. This needs to be supported by the tool, since an HIL project rapidly accumulates large quantities of signal chains. ConfigurationDesk therefore gives users options for creating different working views (see Figure 6). For example, with just a few clicks they can display only signal chains that contain a voltage measurement or create a view of the signal chains that are connected to one specific ECU connector. Of course, a global view containing all the elements of a project is also always available.

D. Types of I/O Functions

As described above, I/O functions are configured at an abstract, logical level, and not on a specific hardware channel. This enables functionality to be reassigned to another I/O board, for example, and it is even possible to use several physical channels if the current/voltage capacity of a signal has to be increased. This abstract configuration level also allows virtual project planning while the HIL hardware setup is still evolving so that configuration work can begin very early in a project. The (incremental) build process is also run from within the tool ConfigurationDesk, resulting in an executable real-time application that can be loaded straight to the HIL simulator.

A large number of I/O functions are available in library form, from simple digital/PWM functions or wheel speed signal simulation to complex angle-based functions for internal combustion engines. In addition, there are I/O functions for the most important aerospace and automotive bus systems, such as ARINC 429, MIL-STD-1553, CAN, LIN, FlexRay and UART. Because the I/O boards are programmable, further I/O functions can be added at any time.

E. Building Real-Time Applications

All the phases of the build process described above are controlled from ConfigurationDesk. The model C code is generated by MATLAB/Simulink. For the I/O functionality, ConfigurationDesk generates extra code parts. It is also possible to switch off individual phases. For example, if only the I/O functionality has been modified, generation time can be saved by deactivating model code generation and reusing “old” model code. When all the C code parts have been generated, they are compiled and linked, and a real-time application is generated from them. The application can then be loaded to a SCALEXIO system for execution.

F. Automated Documentation

Documentation of HIL testing must cover the connected devices, (the interface description of the connected devices, like the number of connectors and real loads), the cable harness between the HIL systems and the ECUs, the HIL hardware components with their configurations, and the environment model (plant model) used for testing.

When the complete configuration process described above is run through, all this documentation is automatically available on the entire signal path from the control unit pin to the external cable harness, control unit/load connector pin, internal cable harness and I/O function, and finally to the model variable. The system configuration is therefore constantly traceable.

V. Integration of 3rd-Party Hardware

The ability to use 3rd-party hardware efficiently with a HIL system is crucial, especially for the aerospace industry. This requirement can be fulfilled by integrating standard PC interfaces (Ethernet, PCI/PCIe, etc.). These interfaces enable interfacing vast amounts of different hardware available in the market. Of course, the real-time capability of the HIL systems must be assured when using 3rd-party hardware. In addition to these hardware-related requirements the HIL technology must also enable the efficient use of 3rd-party hardware in its workflow.

In this section the integration of 3rd-party hardware in SCALEXIO systems uses the example of PMC modules from AIT (Avionics Interface Technologies, Omaha, NE, www.aviftech.com) for standard aerospace buses, like MIL-STD-1553, ARINC 429, etc. PMC modules in general are well known in the aerospace industry. Thus, it is a practical example already implemented by dSPACE. An indicator for the widespread use of PMC modules is their availability for all kinds of aerospace buses. Every common aerospace bus type is available on PMC modules from a number of vendors.

SCALEXIO technology enables the efficient integration and use of PMC modules due to its PCI/PCIe interfaces in the real-time PC. Using PMC modules on carrier cards in the real-time PC enables the same optimal bandwidth between I/O and real-time processor as with IOCNET for the SCALEXIO I/O boards. The optimal real-time performance (e.g., for testing large numbers of bus channels) is ensured by PMC device drivers specifically developed for the SCALEXIO RTOS. Thus, the hardware-related requirements are fulfilled.

The seamless integration of 3rd-party PMC modules into the SCALEXIO workflow is done with the configuration tool ConfigurationDesk. This includes the necessity to interface an external tool for the easy configuration of project-specific bus communication (e.g., based on an Interface Control Definition (ICD)).

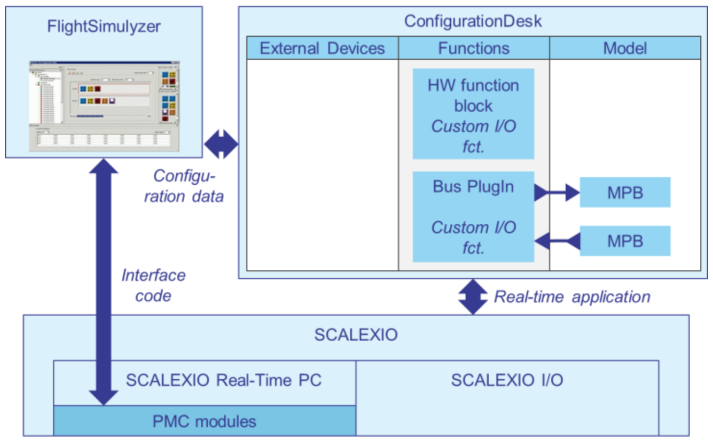


Figure 7. Integration of 3rd-party hardware and software.

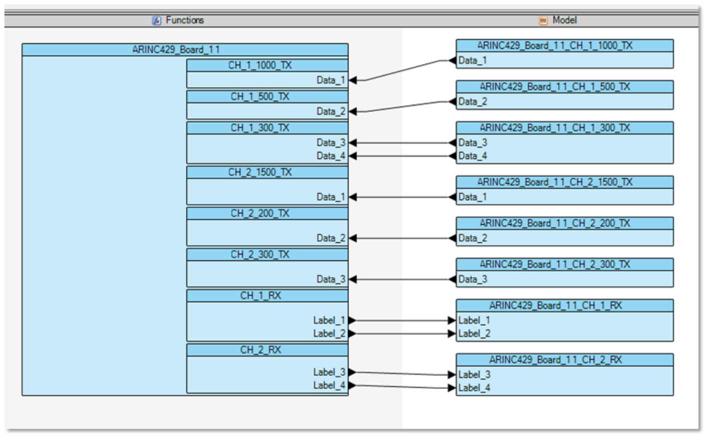


Figure 8. ARINC429 I/O functions and model port blocks.

Interface control information specified by using AIT's tool Flight Simulyzer are imported into the configuration tool ConfigurationDesk for SCALEXIO by using the XML format (see Figure 7). This information is used to auto-generate I/O functions and model port blocks, to visualize them in ConfigurationDesk (see Figure 8), and is included in the build process for SCALEXIO real-time applications. The latter was already described above.

VI. Co-Simulation – Interfacing other Simulation Environments

In order to run an application requiring a different OS on a system with a given operating system, a hypervisor can be used. It enables you to run the required application in a guest operating system.

I.e., there may be software requiring a Linux OS instead of SCALEXIO's RTOS QNX so that this application runs on one core of the quad-core real-time processor (see Figure 9).

For instance, in a pilot project the hypervisor is used to run the simulation software NPSS (Wolverine Ventures, Jupiter, FL, www.wolverine-ventures.com) on one processor core as a Linux application in order to simulate a jet turbine engine. The SCALEXIO real-time application controls the NPSS application. This means that the real-time application transfers data for the next time-discrete simulation step via a virtual Ethernet channel to NPSS and initiates the simulation step. When it has finished, the calculated data is transferred to the real-time application for further simulation steps.

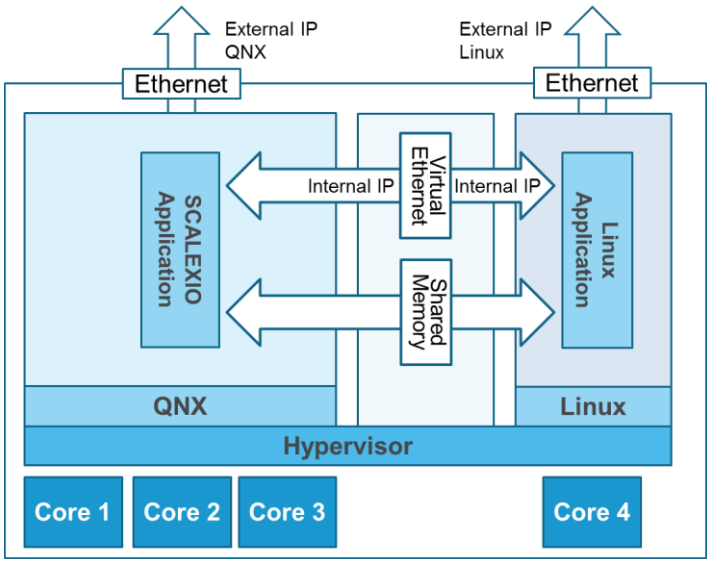


Figure 9. Applying a hypervisor for co-simulation

VII. Conclusion

The new HIL technology SCALEXIO was launched March 2011. It is already being used at a number of companies. These users definitely appreciate the system's flexibility and the ConfigurationDesk workflow. More detailed feedback indicates that users' needs are met by the system's advanced ability to configure hardware via software, the separation of plant model and I/O model, the versatility of HighFlex I/O boards, the galvanic isolation of I/O boards, the real-time processor's calculation power and PC interfaces, and the option to physically separate the I/O from the real-time processor.

References

[1]Burmester, S., Lamberg, K., "Aktuelle Trends beim automatisierten Steuergerätetest", *3rd Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik*, Berlin, Deutschland, 2007.

[2]Himmler, A., "Modular, Scalable Hardware-in-the-loop Systems," *ATZelektronik worldwide*, Vol. 5, No. 2,2012, pp. 36-39.

[3]Köhl, S., Jegminat, D., "How to Do Hardware-in-the-Loop Simulation Right", *SAE Technical Paper* [online database], Paper 2005-01-1657, URL: http://papers.sae.org/2005-01-1657 [cited 13. December 2012].

[4]Müller, B., Köhl, S., "Simulating and Testing In-Vehicle Networks by Hardware-in-the-Loop Simulation", *SAE Technical Paper* [online database], Paper 2008-01-1220, URL: http://papers.sae.org/2008-01-1220 [cited 13. December 2012].

[5]Schütte, H., Wältermann, P., "Hardware-in-the-Loop Testing of Vehicle Dynamics Controllers – A Technical Survey, *SAETechnical Paper* [online database], Paper 2005-01-1660, URL: http://papers.sae.org/2005-01-1660 [cited 13. December 2012].