# A Framework for Hardware-in-the-Loop Testing of an Integrated Architecture

Martin Schlager[1], Roman Obermaisser[2], and Wilfried Elmenreich[2]

[1] TTTech Computertechnik AG
Schoenbrunner Strasse 7, 1040 Vienna, Austria
martin.schlager@tttech.com
[2] Vienna University of Technology
Treitlstrasse 3, 1040 Vienna, Austria
{romano,wil}@vmars.tuwien.ac.at

**Abstract.** In this paper we present a distributed Hardware-in-the-Loop (HiL) simulation approach that supports the verification and validation activities in an integrated architecture as recently developed in DECOS (Dependable Embedded COmponents and Systems), an integrated project within the Sixth Framework Programme of the European Commission. Focusing on the interconnection between the simulated environment and the Integrated System Under Test (ISUT), our approach involves the concept of a Smart Virtual Transducer (SVT) that replaces the physical transducers of the ISUT without a probe effect on the ISUT. Our approach enables a complexity reduction for setting up an HiL simulation and supports a well-designed scalable interface to an integrated architecture. Furthermore, we support non-intrusive, deterministic interaction between the environment simulation system and the ISUT in order to guarantee reproducible test-runs. We show an exemplary application of the proposed concept by tailoring the generic components of the proposed simulation approach to an automotive park assistant system.

## 1 Introduction

The increasing number of electronic functions in future automobiles requires a change from the traditional "one function – one ECU (Electronic Control Unit)" concept to integrated architectures that support bundling several functions in one ECU. Such an *integrated system architecture* must provide means to handle the complexity of distributed applications while supporting efficient integration of functions into the shared hardware.

An example for an integrated system architecture is the DECOS Integrated Architecture [1], which builds upon the validated architectural services of a time-triggered core architecture. A distributed time-triggered computer system provides a physical network as a shared resource for the communication activities of more than one application subsystem. Other integrated architectures are AUTOSAR [2] and IMA [3].

Integrated architectures pose also a challenge to the HiL test procedure, a standard method for testing of an embedded controller before its deployment [4].

HiL simulation is a technique where parts of a real system are replaced by a simulation, i. e., a mathematical model of these real system parts [5]. HiL simulation offers increased realism of the simulation because access to hardware features is provided that would not be available in a pure software simulation. In an integrated system, applying the HiL test procedure requires finding adequate interfaces between the simulator and the Integrated System Under Test (ISUT).

In this paper we present a distributed HiL simulation approach for the DECOS Integrated Architecture. The interaction between the simulated environment and the ISUT involves the concept of a smart virtual transducer (SVT) that replaces the physical transducers of the ISUT without a probe effect on the ISUT. Thus, an ISUT as part of an integrated architecture can be connected to the HiL simulator in a non-intrusive way. Each SVT communicates with other components of a distributed environment simulator via a standardized time-triggered digital interface. Furthermore, an SVT emulates a transducer-specific interface. The proposed concept enables a complexity reduction for setting up a HiL simulation and supports a well-designed scalable interface to an integrated architecture.

The rest of the paper is structured as follows: Section 2 reviews related work in the area of HiL simulation. Section 3 describes structure and features of the integrated system architecture that is used in our approach. Section 4 elaborates on the architecture of the environmental simulation system. We present a case study based on an exemplary prototype application in Section 5 and discuss the implications on reproducibility of simulation results in Section **??**. The paper is concluded in Section 6.

## 2 Related Work

In the literature, a number of approaches can be found that aim at testing distributed real-time applications with a pure software simulation. For instance, a testing method of distributed algorithms that includes a simulation model of the communication subsystem is presented in [6]. Furthermore, in [7], an investigation on pre-validation of system properties of a safety-critical distributed real-time system by a simulation environment can be found. Another approach is given in [8], where a generic MATLAB/Simulink toolbox has been introduced for simulation of distributed real-time control systems. This toolbox includes a real-time kernel module for task activation and a network module that can be tailored to a particular network model.

In contrast to a pure software simulation, HiL simulation involves physical hardware components, i. e., nodes, of a real-time system. Hence, HiL simulation requires the construction of an environment simulator in order to emulate the environment of these nodes [9]. In case only a subset of nodes of a distributed real-time system exist, non-existing nodes must be simulated by a cluster simulator as discussed in [10–12].

HiL simulators are constructed for a wide range of different applications. For instance in [13] real-time HiL simulation of vehicle and mobile robots is proposed

to avoide extensive formal analysis of these systems. In the traffic control domain, system integrators are confronted with frequent changes of signal timing plans implemented in traffic controllers. These signal timing plans are provided by sub-suppliers as closed intellectual property (IP) software modules. Hence, HiL simulation is proposed in order to fine-tune these signal timing plans while at the same time protecting the intellectual property (IP) of the individual sub-suppliers [14].

Commercially available HiL simulation systems range from simple simulators that target at testing a single ECU to complex simulators that are capable of testing large distributed real-time systems. *DSP Builder* [15] by Altera[3] and *Tanto2 Test* by Hitex[4] are examples for simple HiL simulators, where a single hardware target (i. e., an FPGA, or a single ECU) is directly connected to a development PC that executes an environment simulation.

Several vendors offer solutions for more complex HiL simulators. Regarding such complex HiL simulators, we can basically distinguish between monolithic and distributed HiL simulators.

A modular, component-based, monolithic HiL simulator, uses a single device that is configured to offer all required interfaces for a particular SUT. Monolitic HiL simulators are offered for instance by dSpace[5] (*Simulator Mid-Size, Simulator Full-Size*), The Mathworks[6] (*xPC Target*[16]), National Instruments[7] (*LabVIEW*), and Pi Technology[8] (*Pi Autosim*). These simulator products can be equipped with a range of modular I/O boards and processor boards in order to be tailored to a certain HiL simulation system. I/O hardware solutions include analog and digital I/O, CAN, PWM, dynamic signals, motion control, image acquisition as well as FPGA modules.

In contrast to a monolitic HiL simulator, a distributed HiL simulator consists of several interacting nodes that are capable of executing a distributed simulation model. Each of these nodes can be equipped with application-specific I/O hardware. Distributed HiL simulators are provided by Applied Dynamics International (ADI)[9] (*ADI rtX simulator*), Opal-RT[10] (*RT-LAB*), and RTDS Technologies[11] (*RTDS Simulator*). These distributed simulators interact either by the exchange of data that is visible at the interfaces of the SUT (*emulated electronic interfaces*), or by the exchange of data that is part of the simulation model and that is not visible at the SUT's interfaces (*virtual interfaces*) [17]. Communication via the virtual interfaces, i. e., interaction between different nodes of a distributed simulator is either realized by the implementation of an event-triggered protocol (e. g., Ethernet, SCRAMNet, FireWire, or INFINIBAND) or

[3] http://www.altera.com
[4] http://www.hitex.de
[5] http://www.dspace.com
[6] http://www.mathworks.com
[7] http://www.ni.com
[8] http://www.pitechnology.com
[9] http://www.adi.com
[10] http://www.opal-rt.com
[11] http://www.rtds.com

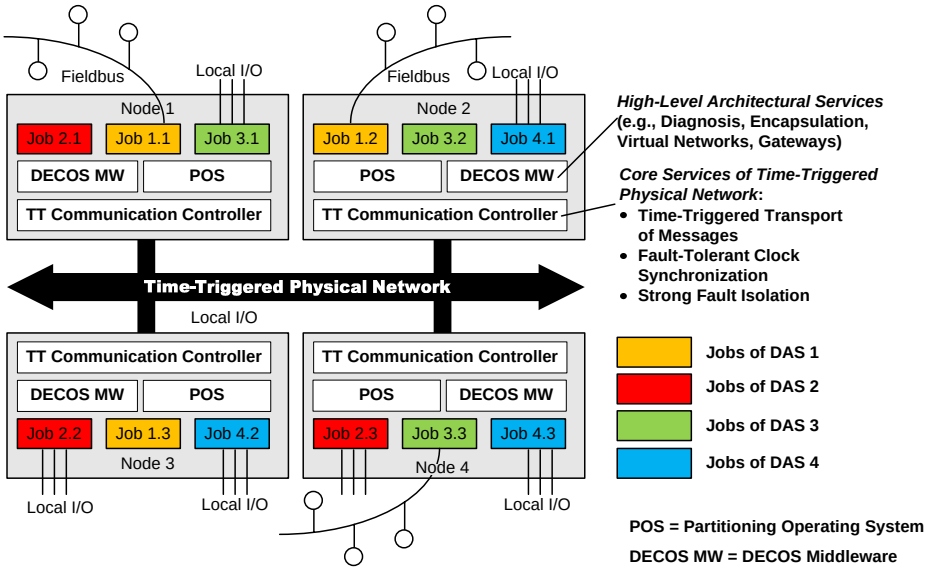


**Fig. 1.** Distributed System in the DECOS System Architecture

by a common communication backplane as for the RTDS Simulator, that links all processing nodes in parallel.

Although all HiL simulators are designed for real-time execution of a simulation model, the existing solutions lack a scalable approach for deterministic interaction between HiL simulator components. Moreover, none of the existing solutions target at HiL simulation in an integrated architecture.

## 3 Integrated System

Many large applications (e. g., in the automotive or aerospace domain) consist of a number of nearly independent application systems. We call such an application subsystem a Distributed Application Subsystem (DAS). A DAS provides a major part of the overall application and is composed of smaller functional elements called *jobs*. In the automotive domain, the powertrain subsystem, the comfort subsystem, and the multimedia are examples for DASs. Examples of DASs in a present-day avionic application are the cabin pressurization system, the fly-by-wire system, and the in-flight entertainment system.

The proposed framework for HiL simulation is designed for integrated architectures, i. e., a single distributed computer system serves as the execution platform for multiple DASs. Each node computer of the distributed computer system contains jobs of one or more DASs (cf. Figure 1). Likewise, the communication network that interconnects the node computers serves the transport of messages between jobs of more than one DAS.

In the following, we will discuss the structural elements of the DECOS architecture (i. e., network, nodes, environment), because this system architecture will be used for the construction of the framework for HiL simulation.

### 3.1 Communication Network

The communication network of the integrated architecture executes a time-triggered protocol (e. g., TTP [18], FlexRay [19]). The rationale behind choosing a time-triggered communication protocol is the suitability for ultra-dependable systems [20]. Time-triggered communication protocols are characterized by a guaranteed message transport with low jitter, error containment between node computers, and a fault-tolerant distributed global clock service.

### 3.2 Node Computers

A node computer provides an execution environment for multiple collocated jobs of one or more DASs as shown in Figure 1. Each job implements a part of the application functionality and is within the responsibility of a single organizational entity (e. g., a specific supplier).

The allocation of computational resources (e. g., memory, CPU time) to jobs occurs using a partitioning operating system with support for fault isolation and modular certification [21, 22]. The partitioning operating system implements mechanisms for spatial and temporal partitioning in order to encapsulate the individual jobs. The scheduling of jobs needs to ensure that a timing failure of a job, such as a worst-case execution time violation, does not affect the CPU time available to other jobs. In analogy, the spatial partitioning mechanisms of the partitioning operating system enforce memory protection between jobs (e. g., with a memory management unit).

The interaction with other jobs occurs through the services provided by the DECOS middleware. The DECOS middleware offers high-level architectural services, which serve as a baseline for the development of applications. These services constitute the interface for the jobs to the underlying platform. Among the high-level services are gateway services, virtual network services, encapsulation services, and error detection services. On top of the time-triggered physical network, different kinds of virtual networks are established and each type of virtual network can exhibit multiple instantiations. Gateway services selectively redirect messages between virtual networks and resolve differences with respect to operational properties and naming. The encapsulation services control the visibility of exchanged messages and ensure spatial and temporal partitioning for virtual networks in order to obtain error containment.

Below the DECOS middleware, each node computer in Figure 1 contains the communication controller. The communication controller executes a time-triggered communication protocol as required for accessing the network. It provides so-called core architectural services (i. e., time-triggered transport of messages, fault-tolerant clock synchronization, strong fault isolation), which are used

as the basis for the implementation of the high-level architectural services in the DECOS middleware.

The rationale for distinguishing between the core architectural services and the high-level architectural services is the ability to exploit existing time-triggered communication protocols for the construction of an integrated architecture. For example, it has been demonstrated by formal analysis [23] and experiments [24] that the Time-Triggered Protocol (TTP) is appropriate for the implementation of applications in the highest criticality class in the aerospace domain according to RTCA DO-178 B Level A.

### 3.3 Input/Output

In order to perform integration tests that involve the interaction between a given distributed computer system and its environment, the framework needs to simulate the physical surroundings of the computer system, i. e., the controlled object(s) and the operator. In a real-world system, the interaction between the computer system and the environment occurs via transducers, i. e., sensors and actuators. These transducers can either be connected directly or interfaced via a fieldbus. The latter approach simplifies the installation from a logical and a physical point of view and is extendable but might introduce higher cost and increased latency of sensory information and actuator control values.

## 4 Environmental Simulation

### 4.1 Simulator Architecture

HiL simulation of an integrated system involves a simulation of the environment of this integrated system. Thereby, a HiL simulation system consists of the following two major building blocks:

**Integrated System Under Test (ISUT):** The ISUT is either an integrated system as outlined in the previous chapter (refer to figure 1) or a part thereof. The ISUT interacts with its environment across the so-called Controlled Object Interface (COI) [26].

**Environment Simulator:** The aim of an environment simulator is to substitute the environment or parts of the environment of the ISUT. An environment simulator is constrained by the properties of the ISUT, i. e., the interfaces of an environment simulator that are relevant for the interaction with the ISUT must resemble the interfaces of the ISUT in the temporal and the functional domains. The environment simulator executes a simulation model of the process under control of the ISUT and a model of the behavior of the transducers. The input and output from these models is fed to the ISUT via the COI.

The COI that links the ISUT and the environment simulator can either be a standardized digital transducer interface or an arbitrary transducer-specific

interface (e. g., an analog interface). Deterministic interaction between the HiL simulator and the ISUT across the COI is an essential aspect for any kind of interface.

In the following we introduce a structured development approach with generic components that can be tailored to establish the coupling between an HiL simulator and a specific ISUT. Hence, we separate between those components that emulate the COI, e. g., via a 4-15mA interface, a fieldbus, or direct I/O, and those components that are used to execute part of a distributed simulation model but do not directly interact with the ISUT.

Following this separation, our HiL simulation framework involves a distributed environment simulator consisting of a set of so-called *frontend simulation components* that control the physical interaction between the environment simulation and the ISUT, as well as a set of so-called *backbone simulation components* that are used to execute (part of) the environment simulation model. Additionally, a time-sync master component is employed in the HiL simulation framework. The time-sync master component is part of the environment simulator, i. e., it triggers the individual backbone and frontend simulation components according to a pre-defined schedule. Furthermore, the time-sync master is a (passive) member of the ISUT, i. e., it synchronizes its time-base with the time-base of the ISUT. Hence, the time-sync master establishes synchronism between the ISUT and the environment simulator without a probe effect with respect to the ISUTs execution.

As depicted in figure 2, the interaction of nodes of an integrated system with their environment is realized via an arbitrary transducer interface including value/time-dependent analog and/or digital direct I/O as well as standardized fieldbus interfaces. A frontend simulation component connects to nodes of the integrated system for the purpose of interacting with these nodes via a particular transducer interface. Frontend simulation components and backbone simulation components collectively execute the distributed simulation model of the environment of the integrated system.

A frontend simulation component requires updates of simulation values that are provided by one or several backbone components. Based on these simulation values, the frontend simulation component determines the I/O signal that is to be provided to the ISUT. Both the control logic that calculates the required I/O signal based on the simulation values and the physical wiring are part of the frontend simulation component. Thus, a change in the interface specification of the ISUT directly affects the frontend simulation component, but not necessarily the backbone simulation component as long as the frontend simulation components can be provided with all relevant simulation values in time.

The availability of separate frontend simulation components in an HiL simulation is particularly advantageous when it comes to incremental testing of an integrated system. Starting with a single node, a stepwise inclusion of jobs of the integrated system in the HiL simulation is required. At each step, the environment model of the real-time system is simulated (by backbone simulation components) and the coupling between this simulation and the actual ISUT is

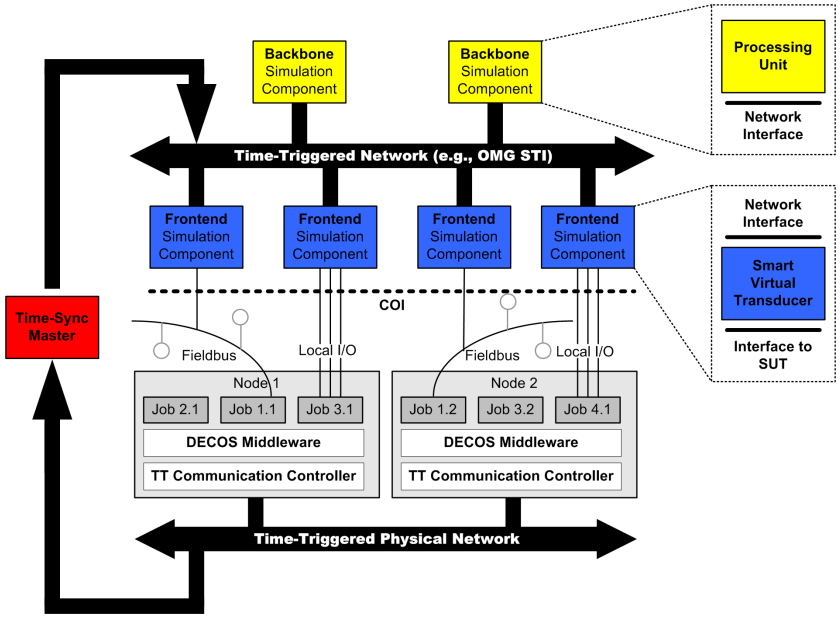


**Fig. 2.** HiL simulation with an Integrated System

established with frontend simulation components. With separate frontend simulation components it is possible to scale the HiL simulation from a small ISUT (e. g., a single node with only one job) up to a complete integrated system by adding additional frontend simulation components as required.

For the realization of the frontend simulation components of the environment simulator, we use *Smart Virtual Transducers (SVTs)* [27]. An SVT implements two interfaces – a standardized digital interface to a time-triggered transducer network (e. g., the Smart Transducer Interface of the Object Management Group [28]) and a transducer-specific interface. The digital interface is used to interact with the backbone simulation components and with other SVTs (i. e., frontend simulation components). The transducer-specific interface resembles the interface of a sensor or actuator element for coupling the SVT with direct I/O of the ISUT. Furthermore, an SVT can implement a certain fieldbus interface. In that case, the SVT would act as a gateway between the environment simulator and a fieldbus of the ISUT.

As depicted in figure 3, an SVT consists of a processor core, memory, a UART, as well as the digital and analog I/O necessary to emulate a specific transducer of the ISUT. The prototype given in figure 3 includes an Atmel ATMega168 microcontroller and an Analog Devices 8-Bit DA converter (AD5330).

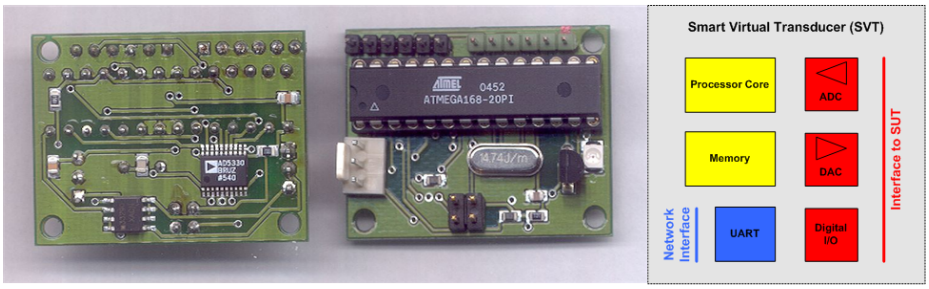


**Fig. 3.** Smart Virtual Transducer (SVT)

### 4.2 Reproducibility of Simulation Results

Deterministic interaction between the environment simulator (i. e., the network of backbone and frontend simulation components) and the respective ISUT (i. e., the integrated system) is a major concern in order to guarantee reproducible results of an HiL simulation run. Thereby, deterministic interaction relates to the functional (i. e., message value or signal size) and the temporal domain (i. e., instant of interaction).

Given two simulation runs $(sim_1, sim_2)$ with an environment simulator, this environment simulator offers deterministic interaction with an ISUT in the temporal domain if it can be guaranteed that a set of inputs from the ISUT to the environment simulator at defined instants[12] cause the same set of outputs from the environment simulator to the ISUT at the same instants during both simulation runs. More precisely, deterministic interaction means that:

- if the simulation runs $sim_k$ $(k = 1, 2)$ start at instants $t_{begin}[sim_k]$ and finish at instants $t_{end}[sim_k]$, and
- the environment simulator receives exactly the same inputs from the ISUT at all instants $t_{begin}[sim_k] + d$ $(0 \leq d \leq t_x[sim_k] - t_{begin}[sim_k])$ during the interval $[t_{begin}[sim_k], t_x[sim_k]]$,
- given that $t_{begin}[sim_k] \leq t_x[sim_k] \leq t_{end}[sim_k]$ and $t_x[sim_1] - t_{begin}[sim_1] = t_x[sim_2] - t_{begin}[sim_2]$,
- then it follows that the (interface) state $S[sim_1]$ of the environment simulator during simulation run $sim_1$ equals the (interface) state $S[sim_2]$ of the environment simulator during simulation run $sim_2$, at all instants $t_{begin}[sim_k] + i$ $(0 \leq i \leq t_x[sim_k] - t_{begin}[sim_k])$, i. e., $S[sim_1]_{t_{begin}[sim_1]+i} = S[sim_2]_{t_{begin}[sim_2]+i}$ (refer to figure 4).

In order to achieve reproducible results in our proposed architecture, the following requirements have to be fulfilled:

[12] These instants relate to a common time-base that is established by synchronizing the time-base of the environment simulator to the time-base of the ISUT.

X

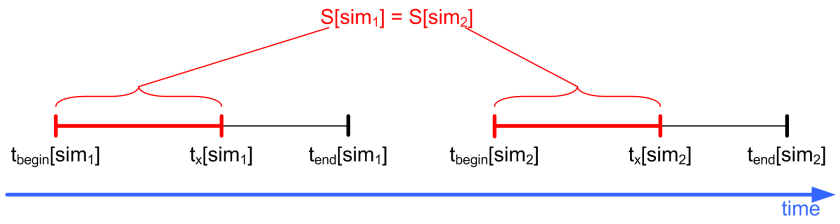


**Fig. 4.** Interface State of HiL Simulator

1. The HiL simulator must share a common time base with the ISUT and have *a priori* knowledge about the time when a sensor is read or an actuator is set by the ISUT.
2. The values exchanged across interfaces between HiL simulator and ISUT must be deterministic.
3. The ISUT and the HiL simulator may not exhibit intrinsic sources of indeterminism, e. g., by suffering from race conditions.

The proposed architecture can satisfy the first requirement by sharing its existing global timebase with the HiL simulator. Furthermore, the DECOS architecture supports a time-triggered action model the allows the prediction of the instants of accessing a sensor's or actuator's value.

The second requirement depends on the employed interfaces. While the digitalization of a pure analog value, e. g., by an ADC, always constitutes a possible source of indeterminism, a DAC – ADC system may behave deterministically, when (i) there is no sampling while the current value is changing to a new one and (ii) each value generated by the DAC can be interpreted by the ADC in a non-ambiguous way. (i) is already solved by the synchronization mechanisms and the temporal determinism of our architecture while (ii) in general requires a careful design of the analog path. For sensor types with only few detection results, e. g., a binary on/off detector, (ii) can be easily fulfilled.

Regarding the HiL simulator, we can establish deterministic behavior due to the usage of a time-triggered communication and execution scheme. Deterministic construction of the ISUT lies outside the sphere of control of the HiL simulator and requires a deterministic architecture. Our proposed case study builds on a time-triggered architecture that avoids sources of indeterminism by design and thus fully satisfies the third requirement.

## 5 Case Study

### 5.1 Exemplary Application Using the Integrated Architecture

The case study used to exemplify the HiL simulation environment includes two automotive DASs (which are part of a larger automotive electronic system):

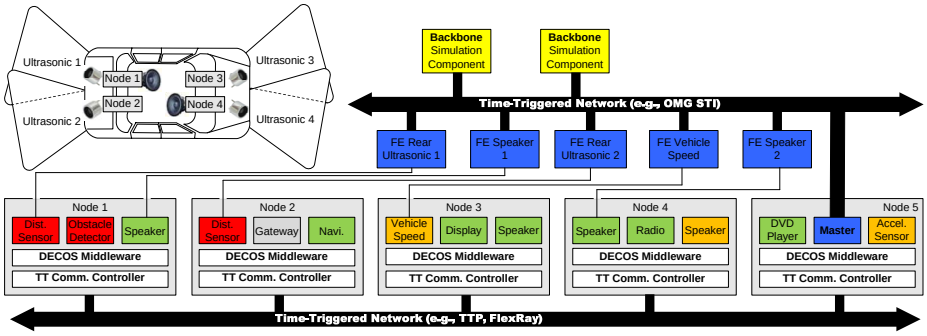


**Fig. 5.** Exemplary Integrated System with Environmental Simulation

– **Multimedia DAS.** Car drivers are no longer satisfied with cars being simple means of transportation. For this reason, today's luxury cars contain multimedia functionality such as DVD players, high-end audio systems, and GPS navigation systems. In addition, voice control and hands-free speaker phones relieve the driver of concentrating on multimedia devices instead of traffic.
– **Park assist DAS.** This DAS implements a parking aid with ultra-sonic sensors. In case a threshold for a minimum distance is exceeded, the DAS produces an acoustic alarm signal. Therefore, the park assist DAS encompasses four jobs reading inputs from ultra-sonic distance sensors. In addition, the DAS contains an obstacle detector job, which reads the distance measurements from the four other jobs and determines whether an alarm signal should be produced. In this case, the acoustic alarm signal is transferred via a gateway to the speaker jobs of the multimedia DAS.

Figure 5 depicts a possible realization of these DASs using the DECOS architecture. Each node computer hosts multiple jobs, which can belong to different DASs (such as the multimedia or park assistant DAS).

### 5.2 Exemplary Environmental Simulation

In the scope of the case study we exemplarily focused on two kinds of transducers, namely ultra-sonic sensors for distance measurement of the park assist DAS and loudspeakers of the multimedia DAS. Hence, the interaction between the environment simulation and the integrated system (i. e., the ISUT) across the COI involves SVTs that emulate the behavior of an ultra-sonic sensor as well as SVTs that capture and process the signals provided by the audio system jobs of the ISUT.

As depicted in figure 5, the setup of the environment simulation system additionally involves a frontend simulation component that receives the actual vehicle speed from the ISUT (i. e., *FE vehicle speed*) and a master node that

controls the operation of the involved SVTs (i. e., *Master*) and that synchronizes the time-base of the environment simulation to the time-base of the ISUT.

Within the prototypical realization of the environment simulation system, we use TTP/A [29] to interconnect the deployed SVTs. The time-triggered fieldbus protocol TTP/A is an implementation of the OMG ST interface standard, including the time-triggered transport service. TTP/A is a round based master slave protocol where multiple nodes of a TTP/A cluster arbitrate a shared bus according to a *time division multiple access (TDMA)* scheme.

In the current implementation we prototypically realized an SVT with a simplified interaction pattern that consists of digital samples for acoustic pressure. This SVT can be used to emulate a loudspeaker of the multimedia DAS. For the ultra-sonic sensors we realized SVTs that emulate a Polaroid 6500 series sonar ranging transducer [30]. A Polaroid 6500 ultra-sonic transducer can be instrumented to operate in single-echo mode. In this mode of operation the INIT input of the transducer is set to high in order to start the transmission of an acoustic signal (16 pulses at 49.4 kHz with 400 volt amplitude). As soon as the echo of this acoustic signal is received back, the ECHO output of the transducer is set to high. The interval between INIT high and ECHO high is proportional to the distance to the measured object.

Each ultrasonic SVT is periodically provided with the actual distance value from a backbone simulation component. Regarding the physical interconnection to the ISUT, an SVT offers an INIT and an ECHO port (digital I/O of the SVT). The ultrasonic SVT polls the INIT input with high frequency. As soon as the input is set to high, a timer is set in accordance with the current distance value. This timer is used to set the ECHO signal of the SVT to high after a specified amount of time.

## 6 Conclusion

In this paper we outlined a distributed HiL simulator that consists of so-called frontend simulation components (FSCs) and backbone simulation components (BSCs). An FSC connects to an Integrated System Under Test (ISUT) across a well-defined interfaces which can either be a fieldbus interface, an arbitrary transducer interface or a physical transducer. A BSC is used for the execution of parts of a distributed environment simulation. For the interconnection of FSCs and BSCs we propose a standardized digital transducer interface, e. g., the OMG STI.

Besides showing an exemplary application of the proposed concept in the automotive domain (i. e., park assistant system), we discussed the prerequisites to achieve reproducible results in our proposed architecture. These prerequisites are: (a) synchronous operation of the HiL simulator and the ISUT, (b) deterministic exchange of values across the COI, and (c) no sources of indeterminism within the ISUT and the HiL simulation system.

Our approach supports the verification and validation activities in an integrated architecture, e. g., DECOS, IMA, AUTOSAR, by realizing a generic

interface for an FSC. Such a generic interface is provided by the concept of a Smart Virtual Transducer (SVT) that replaces the physical transducers of the ISUT. Hence, we support non-intrusive, deterministic interaction between an HiL simulator and an ISUT in order to guarantee reproducible test results. Moreover, this approach offers the possibility to test an integrated system at the physical interface. Hence, it is possible to perform non-intrusive (black box) tests which is particularly important for an integrated system where different vendors provide closed intellectual property software or hardware/software components.

## Acknowledgments

This work has been supported in part by the European IST project ARTIST2 under project No. IST-004527, the European IST project DECOS under project No. IST-511764, and DOC [DOKTORANDENPROGRAMM DER ÖSTERREICHISCHEN AKADEMIE DER WISSENSCHAFTEN]. We would like to thank Bernhard Wenzl for proofreading an earlier version of this paper.

## References


1. R. Obermaisser, P. Peti, B. Huber, and C. El Salloum. Decos: An integrated time-triggered architecture. *e&i journal (Journal of the Austrian professional institution for electrical and information engineering)*, 3, March 2006.
2. AUTOSAR GbR. *AUTOSAR – Technical Overview V2.0.1*, June 2006.
3. Aeronautical Radio Incorporated (ARINC), Annapolis, MD, USA. *ARINC Specification 651: Design Guide for Integrated Modular Avionics*, November 1991.
4. National Instruments Corporation. LabVIEW FPGA in hardware-in-the-loop simulation applications, July 2003.
5. X. Wu, S. Lentijo, A. Deshmuk, A. Monti, and F. Ponci. Design and implementation of a power-hardware-in-the-loop interface: a nonlinear load case study. In *Applied Power Electronics Conference and Exposition (APEC) 2005*, pages 1332–1338. IEEE, March 2005.
6. R. Pallierer. *Validation of Distributed Algorithms in Time-Triggered Systems by Simulation*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2000.
7. J. Ehret. *Validation of Safety-Critical Distributed Real-Time Systems*. PhD thesis, Technische Universität München, Fakultät für Elektrotechnik und Informationstechnik, Arcisstrae 21, 80333 München, Germany, 2003.
8. D. Henriksson, A. Cervin, and K.E. Årzén. TrueTime: Real-time control system simulation with MATLAB/Simulink. In *Proceedings of the Nordic MATLAB Conference*, Copenhagen, Denmark, October 2003.
9. W. Schütz. Testing distributed real-time systems: An overview. Research Report 12/1995, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 1995.
10. W. Fleisch, Th. Ringler, and R. Belschner. Simulation of application software for a TTP real-time subsystem. In *European Simulation Multiconference (ESM)*, Istanbul, Turkey, June 1997.
11. T. Galla. *Cluster Simulation in Time-Triggered Real-Time Systems*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 1999.
12. M. Schlager. A simulation architecture for time-triggered transducer networks. In *Proceedings of the First Workshop on Intelligent Solutions for Embedded Systems (WISES'03)*, pages 39–49, Vienna, Austria, June 2003.
13. Z. Papp, M. Dorrepaal, and D.J. Verburg. Distributed hardware-in-the-loop simulator for autonomous continuous dynamical systems with spatially constrained interactions. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, Nice, France, April 2003.
14. Z. Li, M. Kyte, and B. Johnson. Hardware-in-the-loop real-time simulation interface software design. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pages 1012–1017, Washington, D.C., USA, October 2004.
15. Altera Corporation. DSP Builder – user guide. Available at http://www.altera.com, April 2006.
16. D.J. Burns and A.A. Rodriguez. Hardware-in-the-loop control system development using MATLAB and xPC. Report, Department of Electrical Engineering, Center for System Science and Engineering, Arizona State University, May 2002.
17. Applied Dynamics International. Distributed HIL simulation. Available at http://www.adi.com, 2005.

18. TTTech Computertechnik AG, Schönbrunner Strasse 7, A-1040 Vienna, Austria. *Time-Triggered Protocol TTP/C – High Level Specification Document*, July 2002.
19. FlexRay Consortium. BMW AG, DaimlerChrysler AG, General Motors Corporation, Freescale GmbH, Philips GmbH, Robert Bosch GmbH, and Volkswagen AG. *FlexRay Communications System Protocol Specification Version 2.1*, May 2005.
20. N. Suri, C.J. Walter, and M.M. Hugue. *Advances In Ultra-Dependable Distributed Systems*, chapter 1. IEEE Computer Society Press, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1264, 1995.
21. M. Schlager, W. Herzner, A. Wolf, O. Gründonner, M. Rosenblattl, and E. Erkinger. Encapsulating application subsystems using the DECOS core OS. In *The 25th International Conference on Computer Safety, Security and Reliability (SAFECOMP)*, pages 386–397, Gdansk, Poland, September 2006.
22. B. Huber, P. Peti, R. Obermaisser, and C. El Salloum. Using RTAI/LXRT for partitioning in a prototype implementation of the DECOS architecture. In *Proc. of the Third Int. Workshop on Intelligent Solutions in Embedded Systems*, May 2005.
23. J. Rushby. An overview of formal verification for the time-triggered architecture. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 2469 of *Lecture Notes in Computer Science*, pages 83–105, Oldenburg, Germany, September 2002. Springer-Verlag.
24. A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin. Evaluation of fault handling of the time-triggered architecture with bus and star topology. In *Proc. of Int. Conference on Dependable Systems and Networks*, pages 123–132, 2003.
25. ATIS Committee T1A1, American National Standards Institute, Inc. Telecom glossary 2000, February 2001. Available at http://http://www.atis.org/tg2k/.
26. H. Kopetz, E. Fuchs, D. Millinger, and R. Nossal. An interface as a design object. *2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '99), 2-5 May 1999*, May 1999.
27. M. Schlager, W. Elmenreich, and I. Wenzel. Interface design for hardware-in-the-loop simulation. In *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE'06)*, pages 1554–1559, Montréal, Canada, July 2006.
28. OMG. Smart Transducers Interface. Specification ptc/2002-05-01, Object Management Group, May 2002. Available at http://www.omg.org/.
29. H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2):71–77, March 2001.
30. B. Wirz. Technical specifications for 600 series instrument grade electrostatic transducer. Available at http://controls.ae.gatech.edu/gtar/electronics/6500.pdf, 1997.