

Testing with Virtual Prototypes



(Source: A. Dreher/pixelio.de)

Test description and automation for software verification on virtual prototypes using the TTWorkbench

The complexity and the safety-critical requirements of distributed embedded software used in modern vehicles are constantly growing. This leads to a high verification effort and cost-inefficient design iterations across different development phases to fulfil the quality requirements in a traditional automotive system design flow. In order to allow the verification of distributed software in early design phases, flexible and efficient virtual prototyping approach to perform module and integration tests for protocol or application software is proposed.

By Andreas Braun, Dr. Oliver Bringmann and Prof. Dr. Wolfgang Rosenstiel

One possible way to reduce the verification costs is software testing in early development phases. Even though early verification requires additional testing and modelling effort, costs can be reduced by avoiding cost-intensive design iterations through different development phases. In particular, early verification applies to complex distributed software.

Well known techniques for early verification are model-in-the-loop (MIL) and software-in-the-loop (SIL). Both verification techniques are used with an environment simulation to activate the software. This allows testing of the software functionality without the use of real hardware. The main disadvantages of these techniques are that hardware characteristics and timing behaviour are not considered sufficiently

to find hardware-related software errors. To close this gap between MIL, SIL, and hardware-in-the-loop (HIL), an additional verification step using virtual hardware-in-the-loop (VHIL) is proposed.

The VHIL approach has been applied to the MOST High Protocol

(MHP) [4, 5]. MHP provides packet data transfer in MOST networks. It is a connection-oriented protocol using some mechanisms of the transmission control protocol (TCP). The virtual prototype provides precise timing information and the functional behaviour of the final hardware implementation

at the software level. Test description, execution, and analysis are done using the TTWorkbench from Testing Technologies. TTWorkbench is a TTCN-3-based test modelling and automation environment. This approach allows a fast test description and ensures reuse of the developed test cases for regression or hardware tests by interfacing the virtual and the real network. In order to include the timing and functional behaviour of the MOST network, synchronization mechanisms between TT-

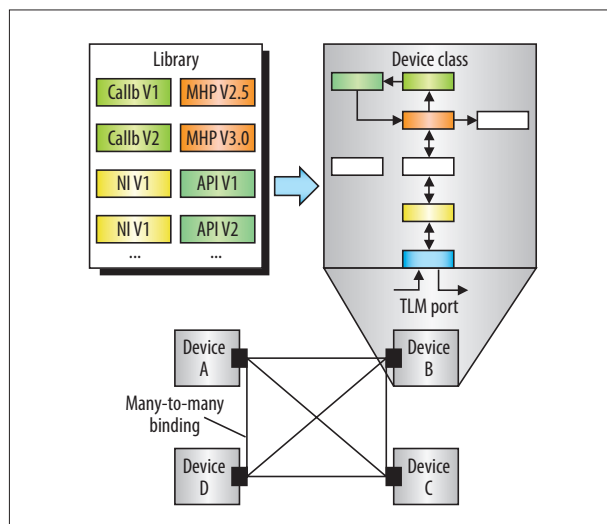


Figure 1. Structure of a virtual network.

Workbench and the virtual prototype have been implemented.

Virtual prototype for MHP verification

The MOST MHP virtual prototype consists of a configurable number of devices and corresponding interconnection components. The devices encapsulate the original MHP protocol stack and are modelled as concurrent processes using SystemC and C++ [6]. SystemC is a C++-based modelling and simulation language that covers concurrency and timing aspects during simulation of embedded software and the underlying hardware platform. The device interconnection is done using Transaction Level Modelling techniques (TLM-2.0) to accelerate the system simulation. TLM is a high-level approach for abstract modelling of communication among modules. The messages, which are transferred via the communication channels and other status parameters, are traced for analysis and automatic checking of properties during simulation.

To provide timing and functional behaviour of the hardware to the simulation model, it is necessary to introduce different elements presenting either hardware or software components. The software components are the encapsulated MHP protocol stack, the call-back functions, and the Application. Hardware components e.g. are the NetInterface representing the INIC and the timer component, which provides timing functionality to the protocol stack.

In order to support testing of different protocol versions or applications, some components of the virtual device can be exchanged. Therefore, a library that can be easily extended with new hardware or software components is introduced. These elements are selected by the network configuration before the simulation starts.

For communication with other devices, every device includes a TLM-2.0 port (figure 1). The data transmission via the TLM ports is abstracted. This

means that not every bit of a frame is simulated, but the complete MHP messages are transmitted between different devices. The ports of the devices are interconnected by a technique called many-to-many binding. Both abstractions lead to an increasing simulation performance and have a minimal effect on timing accuracy during the simulation.

The simulation model is configured by .xml files that are loaded before the simulation starts. The configuration file includes simulation and device configuration. The transaction description for transaction-based protocol verification is defined by the connected test tool TWorkbench. The simulation configuration defines the simulation time, an instance of an optional host, and up to 64 devices. The device configuration selects the elements such as the application or the protocol stack from the library and configures the elements itself. For example the protocol

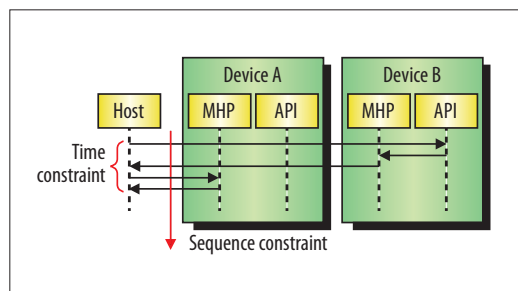


Figure 2. Host based transaction tests.

stack is configured with the number of Tx and Rx connections and the packet size for data transmission by the MHP protocol.

Different elements of the virtual prototype include monitors to trace e.g. state changes, packet losses, and exchanged messages via the communication channel during simulation. This data is stored in files together with time stamps for later analysis. Therefore, different file formats are supported, e.g. a text format and a format that can be processed by the SMSC OptoLyzer Suite or the MOST System Radar. This ensures that the simulation results of the virtual prototype can be compared to results using the real hardware in later phases.

Checking can be done either by introduced assertions using the tracing information or by an optional host that

has access to the communication channels. The host can trigger instances of devices by sending MHP messages, and also check received messages compared to defined sequences and timing constraints (figure 2). Additionally, the virtual host can trigger the application of a device directly and is able to simulate multiple sources and sinks. This allows control of the whole test from the host. If these features are supported by the real hardware, tests are reusable.

TTCN-3-based test description with TWorkbench

The Testing and Test Control Notation version 3 (TTCN-3; [8]) is a modern language for test specification and test execution. It supports different variants of black box tests, e.g. transaction based tests, and is especially useful for communication-based systems. Test specification can be done using sequence diagrams or a textual description similar to a modular programming language. Besides different data types and other typical programming language elements, special elements for timer definitions or test results are available.

To execute TTCN-3-based test definitions, an interpreter or compiler is needed. TWorkbench from Testing Technologies is an eclipse-based test development and execution environment that provides a TTCN-3 debugger and compiler. It supports test automation and analysis and provides interfaces for connecting either hardware or virtual prototypes as system under test (SUT).

The virtual network can include a host which is able to send and receive messages for verification. The connection to TWorkbench is done via a software interface that is included in the host. This allows TWorkbench to stimulate the network and to check received messages. Furthermore, the timing of the virtual prototype and the timing of TWorkbench are synchronized via this software interface.

The connection of TTCN-3 within TWorkbench up to the SUT is done via the codec and the test adapter (figure 3). The function of the codec is to translate defined messages in TTCN-3

to messages that can be processed by the virtual prototype. The test adapter is responsible for communication with the SUT, supporting a number of different languages. This ensures full portability to any platform implementation, e.g. if real MOST hardware is connected to TTWorkbench, no changes in the codec or the TTCN-3-based test description are required.

The link between the virtual prototype and TTWorkbench allows testing in early development phases. To execute the communication sequences, send, receive, and wait functions are available that can control the host of the virtual prototype. In addition to the sequence definition, the platform of the virtual prototype must also be defined. But TTCN-3 does not consider the platform configuration yet. This problem is solved by calling an external configuration file via the software interface of the host.

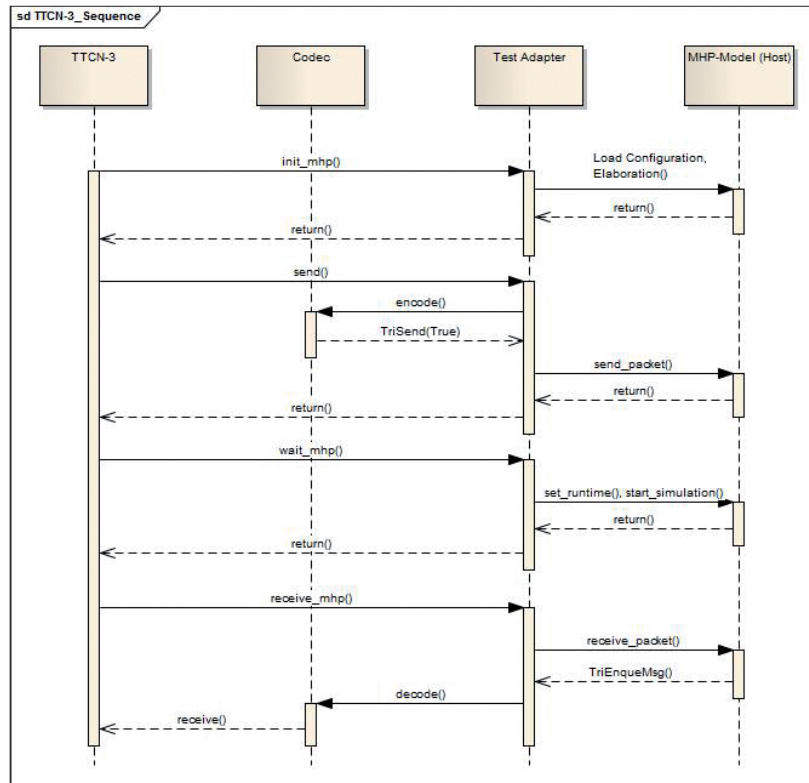


Figure 3. Communication of the TTWorkbench with the virtual prototype.

Early verification can improve test coverage

This article shows an approach for early software verification of distributed embedded systems that consider functional and timing behaviour of the hardware. If some restrictions are followed, it is possible to reuse test cases from the virtual prototype to test the software on the real hardware without any modifications. Analysis of the test results can be done with the common tools used for the hardware or with additional tools e.g. a software debugger

(step by step execution). Furthermore, investigations have shown that with the early verification using virtual prototypes, the test coverage can be improved because of high simulation performance and simple test automation. sj

[2] A. Grzempa, A.: MOST The Automotive Multimedia Network. Franzis Verlag, 2006.
 [3] MOST Specification Rev 3.0. MOST Cooperation, 2008.
 [4] MOST High Protocol Service. SMSC, 2009.
 [5] MOST High Protocol Specification Rev 2.2. MOST Cooperation, 2005.
 [6] Open SystemC Initiative, www.systemc.org
 [7] Liggesmeyer, P.: Software Qualität: Testen, Analysieren und Verifizieren von Software. Spektrum Akademischer Verlag, 2009.
 [8] TTCN-3, www.ttcn-3.org

Literature + Links

[1] MOST Cooperation. www.mostcooperation.com



Dipl.-Ing. Andreas Braun
 works as scientific employee in the research division Microelectronic System Design (SiM) at the FZI (Forschungszentrum Informatik) in Karlsruhe (Germany).
abraun@fzi.de



Dr. rer. nat. Oliver Bringmann
 is division manager in the research division Intelligent Systems and Production Engineering at the FZI (Forschungszentrum Informatik) in Karlsruhe (Germany).
bringmann@fzi.de



Prof. Dr. Wolfgang Rosenstiel
 is professor of the Wilhelm Schickard Institute at the University Tuebingen and director at the FZI (Forschungszentrum Informatik) in Karlsruhe (Germany).
rosentiel@fzi.de