

# Verification and performance analysis

Performance evaluation of different MOST device and network configurations as well as verification of the used protocol stack within a device is an important but time-consuming task. By using state-of-the-art approaches, first results can be achieved only after a real hardware prototype is available. Additionally, the effort for testing and performance evaluation of different network configurations based on real hardware prototypes is very time consuming and very often incomplete. This leads to long design iterations and strongly complicates the investigation of different device and network configurations. This article presents a novel virtual prototyping approach for verification and performance evaluation of distributed embedded systems which allows early and flexible integration of the underlying hardware platform. As network configuration is done automatically, our approach provides a fast and automated network analysis.

The MOST High Protocol (MHP) provides packet data transfer in MOST networks. It is a connection-oriented protocol using some mechanisms of the Transmission Control Protocol (TCP). The MHP protocol is located between network services and the application layer of a MOST device. The communication process between devices include connection establishment, data transmission and connection termination. After data transmission, the receiver informs the transmitter if the data are correctly received or requests the data again.

For testing and performance evaluation of the MHP protocol, a simulation approach based on virtual prototypes has been developed. The virtual prototype encapsulates the MHP source code in a virtual device and provides interconnection to other virtual MHP devices. The virtual prototype is implemented using SystemC to pro-

## Different MHP configurations using virtual prototypes

Within this article a verification and performance evaluation approach based on virtual prototypes is presented. It allows early investigation of network protocols without existence of real hardware prototypes.

Different network configurations can be tested and analyzed by automated simulation runs.

From Andreas Braun, Oliver Bringmann and Wolfgang Rosenstiel

vide accurate simulation of the timing and the functional behavior. SystemC is a modeling and simulation language based on C++ that covers concurrency and timing aspects during simulation of embedded software and the underlying hardware platform. The virtual devices are composed of different modules which are connected via software interfaces. These modules implement the encapsulated MHP protocol stack, the communication modules and the application. Every virtual device can be built of different modules in order to verify the interconnection between different MHP versions during one simulation run. The devices are interconnected by using Transaction Level Modeling techniques (TLM 2.0). TLM is a high-level approach for abstract modeling of communication among modules. The verification and performance evaluation of different device and network configurations is performed using an Automatic Test Pattern Generation (ATPG) approach. The device and network models are configured using XML files. The simulation results are recorded to facilitate comprehensive analysis of the entire system.

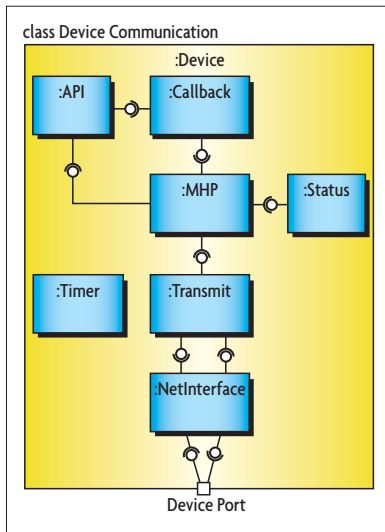
### ■ MOST network simulation including MHP

The MOST network simulation model consists of multiple devices and corresponding interconnection components.

The devices encapsulate the MHP protocol stack. All devices are modeled as concurrent processes using SystemC and C++. The device interconnection is done using TLM to accelerate simulation. Network configuration allows instantiation and simulation of multiple devices including different versions of the MHP protocol stack together with the network model. The communication among the modules



Figure 1. Structure of a virtual device.



and other status parameters are traced for analysis and automatic checking of properties during simulation. An additional host with special functionality is developed to support the MHP verification. The host can stimulate instantiated devices by sending MHP messages via the network. Received messages are stored and checked compared to defined sequences.

**Encapsulation of the MHP source code**

The provided source code of the MHP protocol stack is written in C. To allow multiple instantiation for network simulation, the C source code of the protocol stack was encapsulated in a class called MHP. The MHP class is the central element in each virtual device. To enhance the MHP code by e.g. status data or timer functionality, additional classes are developed and connected to the MHP class. The interconnection within a device is done via several interfaces explained in the following section. The main goal of the encapsulation was to fulfill the requirements for multiple instantiation with minimum modification of the pro-

vided source code. This ensures that new MHP versions can be encapsulated without much effort. The encapsulated code is adapted using two blocks which include class and additional pointer definition for interfaces together with macros for function call redefinition.

**MOST device modeling**

For simulation it is not sufficient to encapsulate the given C-Code of the MHP protocol stack. The class MHP must know the status of the nodes and an application is needed for integrating specific verification scenarios. Furthermore, a component for handling the required timers is necessary.

To deliver the node status to the MHP class, the so called status class is used. This class provides functions which can be called by the MHP class using defined interfaces. The application is integrated into a class called API which implements an interface to the MHP class. The integrated class Callback can call functions of the class API and the callback functions can be called from the MHP class. The timing of the entire node is managed by the Timer class. The MHP class and the API class are allowed to handle timers. Therefore timers can be registered, started and stopped. If a timer interrupt occurs, a defined function is called. All mentioned classes are encapsulated in the class Device. To model the interconnection with other devices the class Device includes a TLM 2.0 port (see figure 1). Two additional classes located in between the class MHP and the interconnection ports are developed, in order to simulate a possible bottleneck between the Electronic Host Controller (EHC) and the Intelligent Network Interface Controller (INIC). These classes are called Transmit and NetInterface. Figure 2 shows the structure of a virtual device including all needed classes and interfaces.

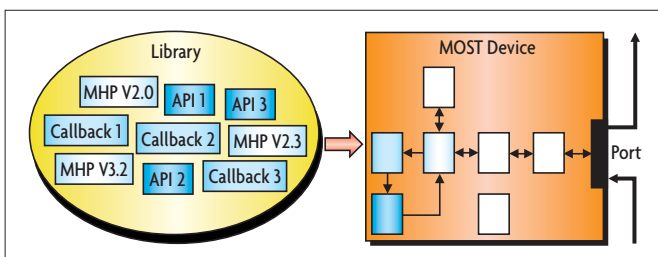


Figure 2. Virtual device configuration.

tion of one of the developed API or Callback component from a library is defined in the configuration file for each device. By using this strategy it is also possible to integrate and test different versions of the MHP protocol stack together in the virtual network.

**Device interconnection**

To communicate with other devices, every device includes a TLM 2.0 port. The data transmission via the TLM ports is abstracted. This means that not every bit of a frame is simulated; complete MHP messages are transmitted between different devices. Additionally, the transmission time of MHP messages between devices is set to zero. The ports of the devices are interconnected by a technique called many-to-many binding (see figure 3) to accelerate simulation. These abstractions should have a minimal effect on accuracy during performance evaluation

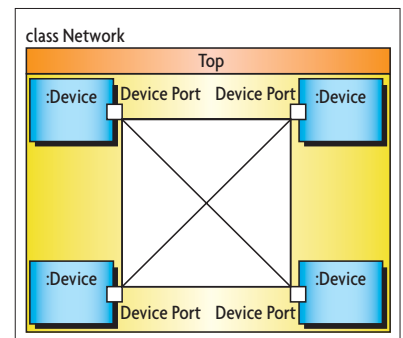


Figure 3. Simplified interconnection structure between different devices.

because of the high bandwidth of MOST networks. The main bottleneck is caused by the data transfer between the EHC and the INIC.

**Network configuration**

The simulation model is configured by XML files which are loaded before the simulation starts. The configuration file includes simulation, host and device configuration. The simulation configuration defines the simulation time, instantiates an optional host and up to 64 devices. The host configuration contains send and receive messages together with addresses, timing and sequence constraints for checking. The device configuration selects the ver-

sion of the classes API, Callback and MHP from a library and configures the classes itself. The MHP class e.g. is configured with the amount of the connections Tx and Rx as well as the packet size for data transmitting by the MHP protocol. Further class parameters are used for input and output buffer or API configuration.

### ▣ Tracing and checking

Every frame exchanged via the channels is stored together with a time stamp in a log file to monitor the communication between instantiated devices. Additionally, detected packet losses, the analyzed communication bandwidth, the number of retries and other parameters are stored for later evaluation. The model provides a better observability and testability during simulation compared to real hardware. A debugger facilitates to observe every variable of the MHP source code and to execute the simulation step by step. An optional host performs checking during simulation. The host stimulates instantiated devices by sending messages and compares the received messages to expected ones. Timing bounds for the received packets are checked additionally.

### ▣ MHP verification and performance analysis

For protocol verification, the additional host can be instantiated for stimulating the network either by sending MHP messages via the network or e.g. by stimulating the API of a device directly. For performance evaluation, no

host is used. Different applications take over traffic generation (figure 4).

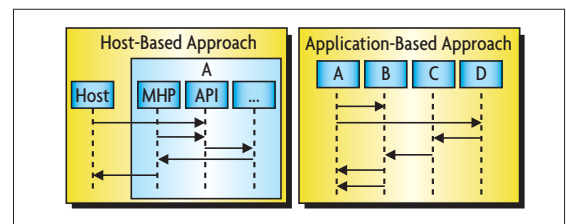
### ▣ MHP verification

The host-based approach is used for protocol verification. The stimuli and the checking constraints for the host are described by the XML configuration file. Checking includes sequence constraints under considerations of conditional expressions and timing bounds for network messages as well as function calls within a device. If an error occurs, the simulation is stopped. The error handler returns an error message and the log file ends after the incorrect frame.

The test description is modeled by UML sequence and composite structure diagrams. The composite structure diagram describes the interconnection structure of the used devices and the configuration of a device. The sequence diagram describes the communication between the defined devices which is checked during simulation. The composite structure and the sequence diagrams are modeled by using Enterprise Architect (EA) and exported to a specific XML format for data exchange called XML Metadata Interchange (XMI). The information within the XMI file is parsed and a configuration file for the simulation model is generated. To automate the verification process, several parameters of the test configuration can be randomized in defined ranges. These test cases can be executed automatically. If an error occurs, the failing test case can be marked for further analysis steps.

### ▣ MHP performance analysis

The application-based approach is used for performance analysis. Different configurable applications for traffic generation are developed. One application e.g. sends data packets with a defined size at a defined point in time, another application sends data packets with a defined size in a defined slot. The transmitting and receiving bandwidth of each device and other parameters e.g. packet loss, retries and buffer overflows are stored for analysis.



▣ Figure 4. Host-based and application-based simulation approaches.

To improve verification coverage, multiple simulation runs for performance evaluation must be analyzed. Different test scenarios, e.g. four transmitting and one receiving device, must be defined. For each of these scenarios, multiple configuration files are generated with randomized configuration parameters by an ATPG. These configurations can be executed automatically by the SystemC model. The corresponding trace files are stored after simulation. Next, these files are analyzed and statistics are generated for each scenario. Worst cases are marked and can be re-executed within the SystemC model for further analysis. *bg*



**Dipl.-Ing. Andreas Braun**

works as scientific employee in the research division Microelectronic System Design (SiM) at the FZI (Research Center for Information Technology) in Karlsruhe (Germany).



**Dr. rer. nat. Oliver Bringmann**

is divisional manager in the research division Intelligent Systems and Production Engineering at the FZI (Research Center for Information Technology) in Karlsruhe (Germany).



**Prof. Dr. Wolfgang Rosenstiel**

is professor of the Wilhelm Schickard Institute at the University Tuebingen and director at the FZI (Research Center for Information Technology) in Karlsruhe (Germany).