Co-simulation of Matlab/Simulink with AMS Designer in System-on-Chip Design

U. Eichler, U. Knöchel, S. Altmann, Fraunhofer Institute for Integrated Circuits, Dresden, Germany
{*eichler, knoechel, altmann*}*@eas.iis.fraunhofer.de*
W. Hartong, J. Hartung, Cadence Design Systems GmbH, Feldkirchen, Germany
{*hartong, juergenh*}*@cadence.com*

With increasing complexity of Systems-on-Chips (SoC), system level design and simulation is a necessity. In an ideal top-down design flow, the system level model is used as executable specification for the block implementation, which is supported by mixed-signal simulators. This contribution describes a link between the system-level simulator MATLAB/Simulink and mixed-signal simulation in Virtuoso AMS Designer by a socket based co-simulation. The implementation of the co-simulation is described in detail, including user interface, protocol, synchronization and cross-platform support. The application of the co-simulation is illustrated by a wireless LAN system. While the RF subsystem of the WLAN receiver is modeled in Virtuoso AMS Designer, Simulink provides standard compliant testbenches and adequate visualization tools. The presented simulator coupling, as a special case of distributed simulation, provides a functional parallelization of the involved tools.

Introduction

Today's integrated circuits often contain analog and digital signal processing as well as microcontrollers and memory. These complex *Systems-on-Chips* (SoCs) provide high functionality and enable the design of smart products for a mass market. Chip designs have to be verified well before the expensive manufacturing of the first prototype in silicon is started. Simulation is the key element in chip verification.
The design flow is usually divided into different abstraction levels as shown in Figure 1. At system level functionality and performance of the whole system are specified and evaluated by system-level simulations. Afterwards, the system is partitioned into hard- and software, analog and digital parts - possibly in several steps. At circuit level these blocks are implemented as analog or mixed-signal circuits or gate netlists. Finally, a layout is designed. After each step the description of the design is more detailed than before.

In modern design flows, simulation support is available for all design levels and system parts. A wide range of tools offer dedicated solutions for specific design problems. Each tool is optimized for a specific level of abstraction and application area. Even though there is a certain range of overlap between the tools, difficulties arise when effects have to be analyzed that span different design levels. While the interfaces between block level and transistor circuit are well established by mixed-signal simulators, the link toward system level is still weak in most environments, although there are approaches e.g. for using simulator coupling in digital design flows ([5]).

MATLAB and its simulation toolbox Simulink are used in system design for many applications including communication, automotive and control systems. The visualization of results is well supported by comprehensive functions for signal analysis and monitoring.

In an ideal top-down design flow the system level model will be used as executable specification for the detailed block implementation using mixed-signal languages like VHDL-AMS or Verilog-AMS. In a following step the implementation proceeds down towards circuit level. Since most of these designs consist of analog and digital parts, their implementation is usually supported by mixed-signal simulators like Virtuoso AMS Designer or ADVance MS. Thus, a very accurate analysis of analog and mixed-signal circuits is provided. On the other hand this accuracy naturally reduces the simulation speed, so that it is often impractical to verify the whole system behavior on a very detailed level of abstraction.

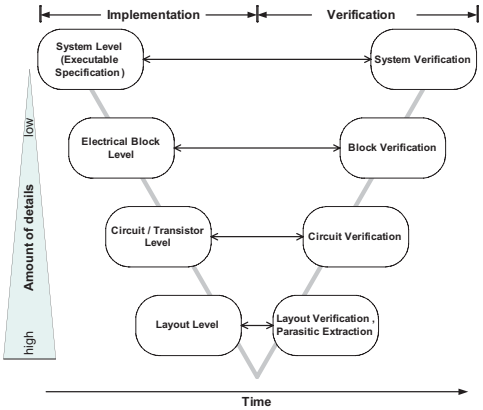

Figure 1: Design levels.

A verification of the implemented blocks against system level is difficult, due to the missing link. On the other hand, individually testing the designed circuits often does not ensure a working system, and designs fail due to problems at the interfaces. A direct link to the system environment can help to reduce interface problems and increase design efficiency and quality.
This was the main motivation for the development of the AMS Designer - Simulink co-simulation feature. Designers of analog and mixed-signal systems can evaluate their designs within a system model that can be reused from system design. The powerful Simulink model libraries simplify the design of module testbenches. System designers may include block or circuit level models of critical analog modules in the system simulation to analyze the performance impact and to adjust analog and digital parts, e.g. by digital predistortion of signals to compensate the non-linearity of a succeeding analog amplifier. Some of the existing solutions for multi-level simulation have been evaluated and improved within the project DETAILS ([*]). It is focused on an integrated simulation flow from system-level to mixed-signal and RF circuit implementation.

1 Concept and Implementation

When linking two simulators for co-simulation three main aspects of implementation have to be considered: the coupling of the different simulation algorithms, the choice of an appropriate user interface that integrates well in the simulators' handling concepts, and how both simulators should communicate. In the presented simulator coupling the mixed-signal simulator *Virtuoso AMS Designer* is used for the block and circuit level simulation. MATLAB/Simulink acts as sys-tem level simulator. Because both tools calculate the time-dependent behavior of the analyzed model (time-domain simulation), the coupling algorithm is focused on synchronization which is discussed in Section *Synchronization* below.
The simulator coupling user interface has to provide a simple way for the user to define the border between both model parts residing in the two different simulation environments. That concerns the choice of the signals to be transferred, their data types and the sampling mode to be used. Special coupler modules have been introduced for both simulation environments. The coupler module represents the model part that resides in the other simulator. Signals that should be transferred to or from the other model part are connected to the input or output ports of the coupler (see Figure 2). This approach allows to keep the modular structure of a model when splitting it for co-simulation.

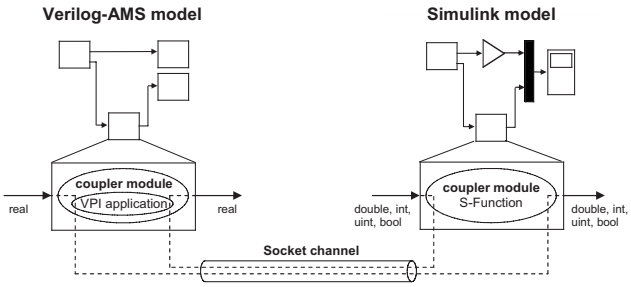

Figure 2: Co-simulation principle.

The coupler modules can be easily inserted and parameterized using the well-known graphical user interfaces of both simulation environments.

For the communication between the coupled simulators a TCP/IP network socket connection is used, allowing the co-simulation to be run on a single machine as well as on different hosts in a network. Different operating systems and platforms are supported in one co-simulation run (cross-platform simulation, see also Section *Platform Support* below). When running on different machines, the co-simulation may profit from better memory utilization because more memory is available per simulator.

1.1 Virtuoso Coupler Module

For AMS Designer the actual coupler module is written in *Verilog-AMS*. It provides port definitions, contribution statements for port access, and intermediate variables of type *real* for each input and output port. These variables are read and written by the VPI application - a dynamically loaded library which is written in C and contains the main coupling functionality. VPI, the *Verilog Procedural Interface*, provides several functions to interact with the simulation engine and to access Verilog objects like variables, modules and ports. The VPI application is started by a user-defined system task called at initialization of the coupler module:

```
initial $couple_init(CouplerToSimulink,hostName);
```

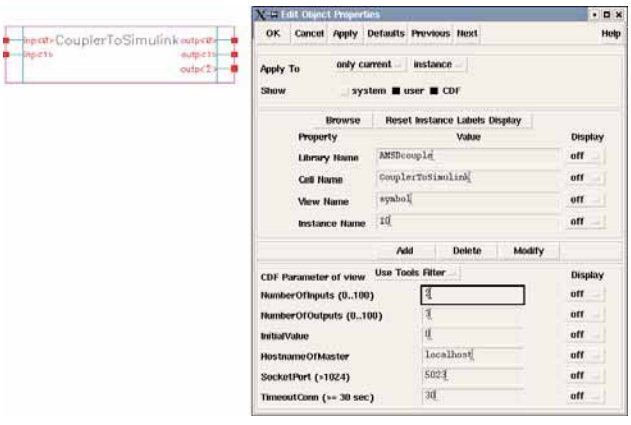

Figure 3: Virtuoso AMS Designer coupler module.

In each step, the simulation data received from Simulink is written to the Verilog module's output variables, and the input variables are read. Inside the Verilog coupler module these variables has to be mapped to the module ports. Currently, there are two coupler modules implemented. The first one is a pure digital module and maps data directly to its ports of type *wreal*. The second one is shown in Figure 3 and has analog electrical ports. It uses an interpolation algorithm to write the received data to its output ports.

Symbol and parameter dialog of the Verilog-AMS coupler module in Virtuoso AMS Designer are shown in Figure 3. The following parameters are provided:

- *NumberOfInputs, NumberOfOutputs*: The number of module ports per direction. The symbol view is changed according to these settings.
- *InitialValue:* The starting value for the inter polation algorithm. It is provided at the coupler module's output ports at simulation time t_0 (Figure 5, 6).
- *HostnameOfMaster:*The name of the local or remote host where Simulink is started.
- *SocketPort:*The TCP/IP port number for the socket connection. Both simulators must use the same setting in order to communicate.
- *TimeoutConn:* Time before terminating simulation if no data is received from the other simulator.

1.2 Simulink Coupler Module

The C-based s-function API was used to implement the coupler module on Simulink side. The advantage is that common functions for protocol and socket access could be shared by VPI application and s-function. The S-Function code is compiled to a shared library and contains the entire functionality of the coupler except the parameter dialog which was created using the *Simulink Mask Editor* (see Figure 4). The coupler module can be executed either at a possibly variable sample rate, inherited from the connected blocks, or at a constant, user-defined sample rate.

With the following three parameters the sampling mode can be controlled:

- *FrameMode*:Toggles between framed and unframed synchronization (see below).
- *FrameSize*: The number of samples per frame. If the frame size should be inherited from the connected blocks, this value is set to -1.

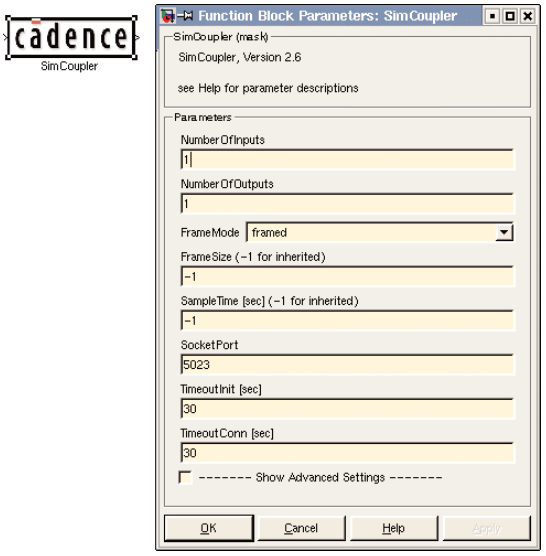

Figure 4: Simulink coupler module.

- *SampleTime*: This value defines a fixed sampling or frame period for the coupler block. If set to -1, sample time is inherited from the connected blocks.

1.3 Synchronization

When splitting a simulation task into several parts for co-simulation, the synchronization of the involved simulation tools is an important issue. This comprises the choice of an appropriate synchronization algorithm as well as its implementation (for overview, see [4]).
A synchronization algorithm should strongly depend on the scheduling schemes used by the coupled simulators. In our case these are a dataflow-like scheduling algorithm with fixed or variable time steps for Simulink and a combination of discrete-event control with a continuous-time analog solver for the mixed-signal simulator AMS Designer. Here, the set of applicable synchronization schemes is mostly determined by Simulink, because in dataflow simulation a block is not executed until all of its inputs are calculated by their drivers. This results in a fixed execution order of all blocks that is repeated for each sample period. Thus, also the Verilog-AMS model represented by the coupler block has to be executed accordingly to this order and has to calculate its outputsfor that sample period. The synchronization time points are given by Simulink. This scheme is a conservative synchronization approach ([3], [1]).
With these preconditions an implementation using the master-slave principle was the most preferable one due to its simplicity and its only small communica-tion overhead. Consequently, Simulink was chosen to take the master role for synchronization and connection setup.

The simulators exchange data frames with blocking access to the socket connection. Simulation time advance is controlled by the master simulator and is always positive for both simulators. That means, Simulink calculates the coupler module's input data for the current time step, sends this data together with the time of the next sampling point to the Verilog coupler. AMS Designer advances simulation until this time and sends back its output data to Simulink.

Inside the Verilog-AMS coupler module the discrete-timed data received from Simulink has to be mapped to the continuous time axis of the analog part. This is done by an interpolation algorithm that calculates additional values between the received samples if requested by the analog solver. The interpolation is done linearly starting at simulation time t_0 with the value of the parameter *InitialValue*. In the opposite direction the input signals of the Verilog coupler are sampled at the times given by Simulink. This can influence the co-simulation performance significantly. Thus, the sampling rate should be chosen carefully. If it is too low, signal changes with smaller time constants are lost. If the sampling rate is too high, simulation performance decreases.

With its signal processing blockset Simulink provides a special signal type - so-called frame-based signals. These compose several successive samples to a single frame and transmit them all at once. Frame-based signals can help modeling multi-rate systems and increase simulation performance significantly due to the reduced communication effort between connected blocks ([6]). This feature had to be considered also for the proposed co-simulation. Supporting Simulink's different sampling modes - from variable sample rates to frame-based signal processing - was a challenge for the co-simulation implementation.

For more flexibility two different synchronization algorithms for framed and unframed data are used. In unframed mode (Figure 5) the Simulink coupler module does not exchange data at simulation time t_0.

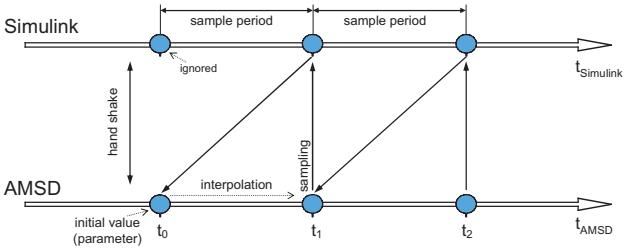

Figure 5: Sample-based synchronization.

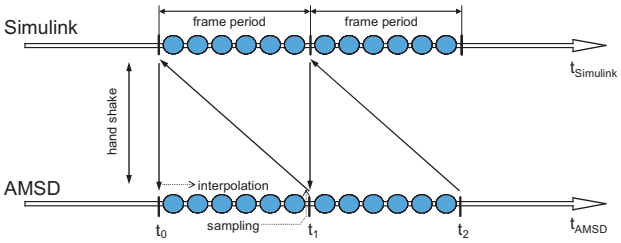

Figure 6: Frame-based synchronization.

The first input sample of the coupler module is ignored and a value of 0 is written to the outputs. The input sample of the second sampling period is then sent to AMS Designer together with the current simulation time t_1. AMS Designer advances simulation until t_1, samples the coupler module's input signals and sends these values back to Simulink. Due to the interpolation algorithm in the Verilog coupler module, the signal values received at time t_0 are not achieved until t_1 is reached. Thus, the introduced delay of one sample period is compensated, and time axes of both simulators are absolutely synchronous. Because the Simulink coupler module only needs to know the current simulation time, the synchronization scheme for the unframed mode allows using the coupler also in models with variable sample time.

In framed mode whole data frames are exchanged between the simulators at the equidistant frame sample points. It is important to note, that Simulink always generates frames for the following frame period. Thus, the first frame is sent at simulation time t_0 from Simulink to AMS Designer for the interval from t_0 to t_1 (Figure 6). The Simulink coupler module has to know the next synchronization point t_{n+1} when sending a data frame. This is only possible with fixed sampling rates as used in framed mode. Within the VPI application of the Verilog-AMS coupler the incoming data frame is split into the original data points. The Verilog simulation works subsequently on this input data. The generated output data is again collected into a frame and sent back to Simulink once the frame is complete. At sample level AMS Designer shows a delay of one sample period compared to Simulink resulting from the linear interpolation between the sample points (see above). In both, framed and unframed mode, there appears no delay between the input and output ports of the Simulink coupler module.

For the analog co-simulation, the use of the two different schemes provides the most suitable synchronization in both, framed and unframed mode. The unframed synchronization also allows variable sample times in Simulink. The occurring delays are negligible.

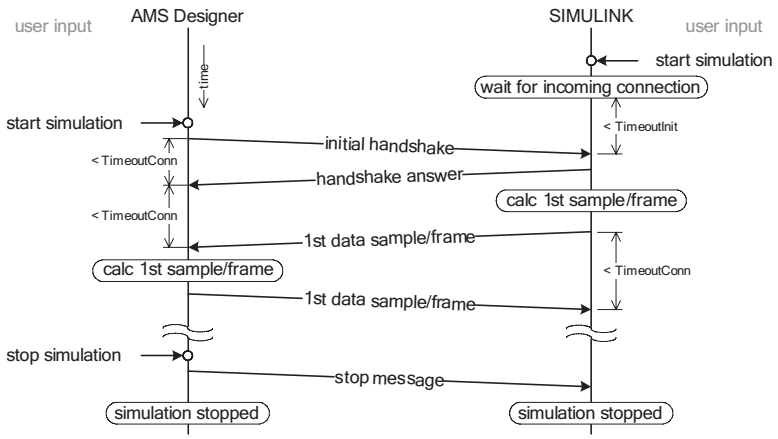

Figure 7: Protocol message flow.

1.4 Protocol

The implemented master-slave coupling principle has only few requirements concerning the underlying communication protocol. There are initial handshake messages exchanged at co-simulation setup and data messages containing simulation data (see Figure 7). When starting a co-simulation, the master simulator acts as server waiting for client requests. After the client simulator has sent an initial handshake message, the master sends a handshake reply containing information on its *Endian byte format* (see below) and the number, data types and dimensions of its coupler module's ports.

If the client simulator accepts the received settings, simulation starts with the first data frame from the master simulator. The data messages contain a flag describing the simulation status. It is mainly used to signalize errors or the end of simulation. The maximum time a simulator waits for an incoming message can be set by a parameter of the coupler module.

1.5 Platform Support

The current implementation of the simulator coupling supports the Linux, Solaris, and Windows operating systems and the Sparc and x86 processor architectures. To enable cross-platform simulation, it was necessary to consider the different Endian formats of the target platforms. Data is stored in sequences of bytes assembled of eight bits. To store numbers like integers or doubles using more than eight bits, several consecutive bytes are used. Different processor architectures use different byte orders inside those multi-byte numbers.

There are two common formats: *Little Endian*, used by Intel processors and *Big Endian* used by Sparc or Motorola processors and for protocol data in TCP/IP networks.

The cross-platform support provides an automated detection of the Endian format on both machines. A conversion is done only in the case of different formats to minimize the simulation overhead.

For *Little Endian* the least significant byte is stored at the first position (the lowest address) and for *Big Endian* the most significant byte comes first. To perform a cross-platform co-simulation, the Endian format of the master machine must be detected and - if necessary - transferred data must be converted when sent/received from the other simulator. In the current implementation the initial handshake messages contain a flag that indicates the Endian format of the simulator and are sent always in Big Endian format. The subsequent data messages are sent in the Endian format of the master simulator. That means, during simulation only the client simulator converts data if the Endian formats of client and master are different.

Endian format detection is done by casting the first byte of a long integer variable with value 1 to a one-byte character and checking whether it's value is 0 (Big Endian) or 1 (Little Endian). The data conversion simply re-orders the bytes of each double or integer in the reverse order.

2 Application Example: Wireless LAN Transceiver

In this section the application of the co-simulation for modeling a wireless LAN IEEE 802.11b physical layer transmission system on different levels of abstraction is shown.

Figure 8 shows the Simulink top-level schematic of a wireless LAN system level model. Binary random data is encoded and modulated in the transmitter part. The OFDM signal is then transferred to a channel model.

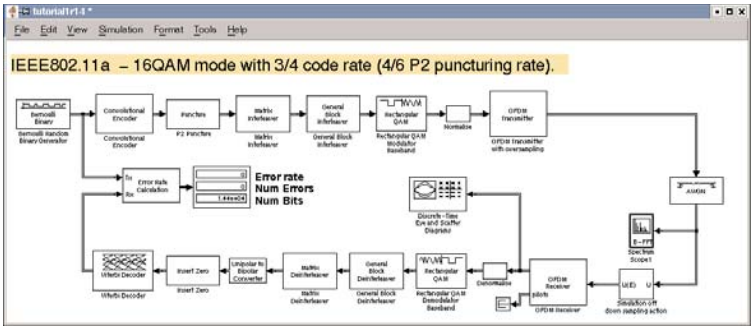

Figure 8: End-to-end system-level simulation with Simulink.

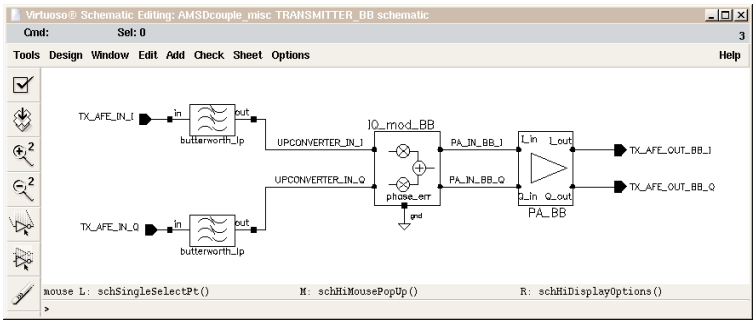

Figure 9: Behavioral model of the transmitter RF frontend.

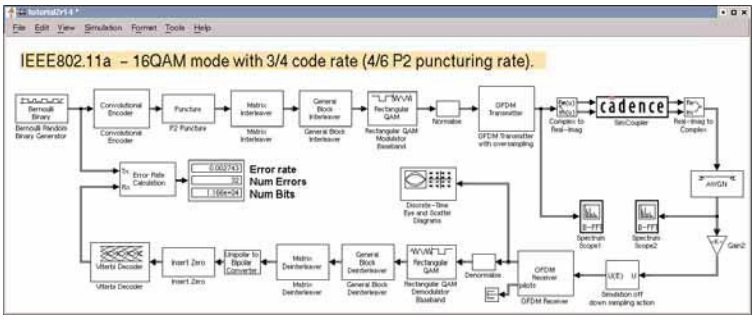

Figure 10: Simulink model with modules for co-simulation.

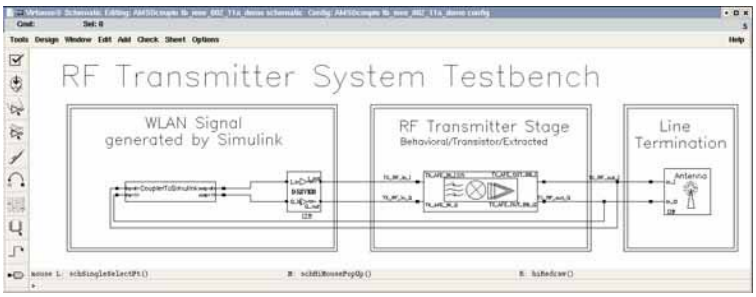

Figure 11: AMSD testbench with couple module.

In the example a White Gaussian Noise channel is used. The receiver blocks demodulate and decode the channel output. Finally the received bits are compared with the original bit stream to compute the bit error rate.

This standard compliant model of the wireless LAN link is built with modules from the Simulink communication and signal-processing toolboxes. The sample model contains only the digital parts of the transmission system. Effects originating from the analog RF parts of transmitter and receiver are not considered in the current simulation.

The RF frontend was designed using SpectreRF and AMS Designer within the Cadence Virtuoso environment. Figure 9 depicts the behavioral model of the RF transmitter module, which filters, up converts and amplifies the signal.

For simulation speed-up, the RF parts are modeled in complex baseband domain as Verilog-A behavioral models. It is possible to switch the abstraction level as far down as transistor level. However, the simulation performance will be lower in this case.

One- and two-tone sources are typically used in this environment as stimuli for the analysis of the RF sub-systems. Characteristics of the design are for example intercept points, noise figures and corner frequencies.

In most cases, it is much easier to handle more realistic stimuli, like modulated signals and corresponding DSP post-processing blocks for performance evaluation, on system level using Simulink.

With the co-simulation those tests are set up easily without modifying the environment setup too much.

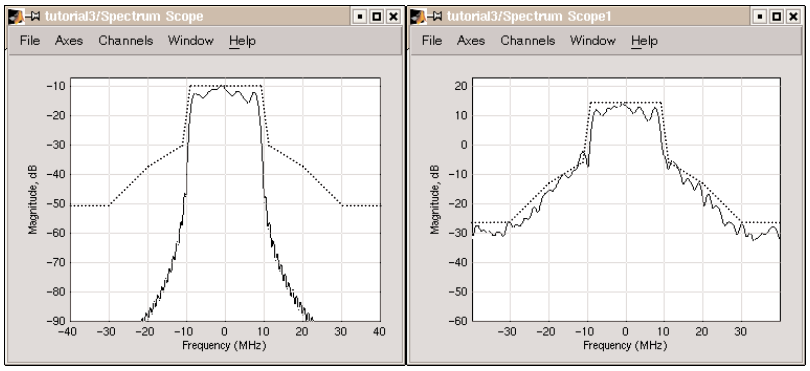

Figure 12: Co-simulation results, spectral masks.

A coupler module is used to link the RF transmitter model into the Simulink system-level schematic (Figure 10). In AMS Designer the RF frontend model is embedded in a separate testbench containing the corresponding coupler module and a simple antenna model (Figure 11). The input of this coupler module is sent through the socket connection to the output of the Simulink coupler module and is therefore connected to the output of the transmitter model.

Figure 12 depicts the frequency characteristic of the transmitted OFDM signal and the spectral mask (dotted) for IEEE 802.11a. The transmitted signal must be within this mask to fulfill the specification. The left-hand plot shows the signal generated by the digital baseband in Simulink. After passing the RF frontend some deviations can be observed, caused by nonlinear behavior (right-hand plot). The co-simulation can now be used to improve the system model by optimizing the parameters of RF front-end and DSP part.

3 Summary and Outlook

The presented simulator coupling enables the co-simulation of MATLAB/Simulink and the mixed-signal simulator Virtuoso AMS Designer. Its main advantage comprises the possibility to integrate design verification steps into system level simulation by increasing simulation accuracy of selected parts of a model - if necessary down to circuit level. Here, the general tradeoff between simulation accuracy and performance has to be taken into account. This application scenario was demonstrated by a WLAN transceiver model.
Furthermore, the coupling allows to use special features of one simulator in a co-simulation, e.g. Simulink blocks for stimuli generation and postprocessing, AMS Designer for multi-language mixed-signal simulation. Cadence Design Systems is providing this coupling feature within the current software release. It has been successfully tested by several major design companies.

References

[1] U. Donath et al.: *Parallel Multi-Level Simulation with a Conservative Approach*. J. Systems Analysis - Modelling - Simulation 21(1995), pp. 187-201

[2] R. Frevert et al.: *Modeling and Simulation for RF System Design*. ISBN 0-387-27584-3, Dordrecht, Springer, 2005

[3] D. Kim, C.-E. Rhee, S. Ha: *Combined Data-Driven and Event-Driven Scheduling Technique for Fast Distributed Cosimulation*. IEEE Trans. on VLSI Systems, Vol. 10, No. 5, pp. 672-678, Oct. 2002

[4] P. Le Marrec et al.: *Hardware, Software and Mechanical Cosimulation for Automotive Applications*. Proc. 9th IEEE Int. Workshop on Rapid System Proto-typing, pp. 202-206, Leuven, June 1998

[5] S. Wielens, S. Altmann, J. Haufe, P. Schneider: *Integration of Prototypes into the Design Flow of Digital Hardware for Applications in Mechatronics and Telecommunication*. Proc. Model-Based Design Conf. 2005, pp. 55-60, Munich, June 2005

[6] *Simulink Signal Processing Blockset*, WWW.MATHWORKS.COM/products/sigprocblockset/

[7] *Cadence RF Design Methodology Kit*. WWW.CADENCE.COM/products/kits/RF_Design/

[*] The presented work was partly funded by the project DETAILS, promoted by the German BMBF (Sign 01M3071) within the initiative ‘Mobile Internet’.

Corresponding author: U. Eichler
U. Eichler, U. Knöchel, S. Altmann
Fraunhofer Institute for Integrated Circuits (IIS),
Branch Lab Design Automation (EAS)
Zeunerstraße 38, 01069 Dresden, Germany
{*eichler, knoechel, altmann*}*@eas.iis.fraunhofer.de*
W. Hartong, J. Hartung, Cadence Design Systems GmbH
Mozartstrasse 2, 85622 Feldkirchen, Germany
{*hartong, juergenh*}*@cadence.com*

Received: May 2, 2006
Revised: June 8, 2006
Accepted: June 20, 2006