An Integrated Co-simulation Interface for Mixed-Level System-on-Chip Design

Fraunhofer Institute For Integrated Circuits
Uwe Eichler, Uwe Knochel, Sven Altmann

Cadence Design Systems
Walter Hartong, Tina Najibi

Presented at

An Integrated Co-simulation Interface for Mixed-Level System-on-Chip Design

Uwe Eichler*, Walter Hartong**, Tina Najibi**, Uwe Knöchel*, Sven Altmann*

* Fraunhofer Institute for Integrated Circuits (IIS),
Branch Lab Design Automation (EAS),
Dresden, Germany
{eichler,knoechel,altmann}@eas.iis.fraunhofer.de

** Cadence Design Systems,
Munich/San Jose, Germany/USA
{hartong,tmm}@cadence.com

Abstract

The continuously increasing complexity of today's embedded electronic systems strongly requires including system level into design and verification. An ideal top-down design flow would use a system-level model as the executable specification and testbench within which analog or mixed-signal blocks can be developed and verified. This paper describes the co-simulation link between the system-level simulator, Matlab-Simulink, and a mixed-signal simulation environment, AMS Designer. After a detailed description of the implementation and the two synchronization algorithms for framed and unframed data, the co-simulation is demonstrated using the design of a wireless LAN system. Simulating such a complex system-design at the transistor level is not even possible within the design cycle time and simulating the entire design at the system level is not as accurate as transistor-level simulations. This session will demonstrate that performance and accuracy tradeoffs can easily be made using this approach and will demonstrate how to reduce interface problems and increase design efficiency and quality by allowing the different blocks to be revised and tested within the overall system.

1 Introduction

With increasing design complexity and short time-to-market demands, system level design and simulation is no longer a luxury, but a necessity. Successfully designing today's complex mixed systems not only requires new design methodologies, but also modeling languages that are suited best for the various design aspects – from high levels of abstraction down to accurate circuit implementation.

Mismatches between the system specification and the actual implementation are common if analog/mixed signal designers develop their circuits from scratch, based on paper specifications from the system engineers. A verification of the implemented blocks within the entire system is nearly impossible. 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 will help to reduce interface problems and increase design efficiency and quality by allowing the different blocks to be revised and tested within the overall system.

1.1 Design levels and languages

In modern design flows, simulation support is available for all design levels including system. A wide range of tools offer dedicated solutions for specific design problems, as shown in Figure 1. 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 need to be analyzed that span different 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.

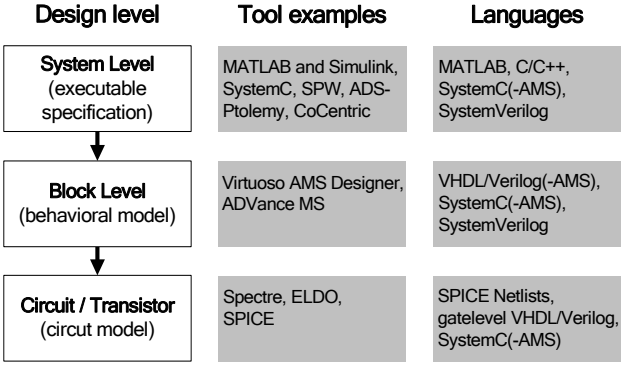

Figure 1: Design levels, tools and languages

MATLAB and its simulation toolbox Simulink are used in system design for many applications including communication, automotive and control systems. These system models often contain parts representing the environment of the device, for example wireless

transmission channels, or complex functionality such as driver behavior and car dynamics. The extensive model libraries of MATLAB and Simulink allow efficient system modeling and fast behavioral simulation for the device itself as well as for the environment. A rich set of functions for signal analysis and monitoring as well as a variety of different visualization tools are supported.

In an ideal top-down design flow the system-level model is used as the executable specification for the detailed block implementation that is modeled using mixed-signal languages like Verilog-AMS or VHDL-AMS. Once the system level simulation is complete, implementation for the individual blocks proceeds down towards circuit level. Since most of these designs consist of analog and digital parts, they are usually implemented using mixed-signal simulators like Virtuoso AMS Designer or ADVance MS. Thus, a very accurate analysis of these analog and mixed-signal blocks is possible. [1].

While it is possible to accurately simulate the individual blocks, simulating the overall system design with the same degree of accuracy is impractical. A tie from the system level to the mixed-signal blocks is required. A user can then leverage a good interface to the system-level environment by including individual mixed-signal subsystems into the system model as needed to verify a particular design step.

1.2 Co-simulation goals

Closing the gap between Simulink based system design and mixed-signal block level simulation was the primary motivation for the development of an AMS – Simulink co-simulation interface.

Designers of analog and mixed-signal systems can evaluate their designs within a system model that can be reused from the system design. The large Simulink model libraries simplify the design of module testbenches.

System designers may include lower level models of critical analog blocks in system simulation to analyze the performance impact and to adjust analog and digital parts, e.g. by digital pre-distortion of signals to compensate the non-linearity of a succeeding analog amplifier.

2 Related work

Several design tools from different application areas provide an option to integrate Simulink in their design flow using a co-simulation interface. In the area of electronic design automation (EDA) these are mainly HDL simulators using VHDL(-AMS), Verilog(-AMS) or SystemC for pure digital or mixed-signal designs.

For pure digital HDL designs Simulink interfaces are provided e. g. for ModelSim [2][3] and Active-HDL [4]. These are naturally restricted to discrete-time. In the mixed-signal and multi-physics domain Simulink can be linked with SystemVision [5], Saber [6] or Simplorer [7].

Although these integrations provide different features and application focus, this paper concentrates on the Virtuoso AMS Designer – Simulink co-simulation technology. It provides access not only to the digital but also the analog part of the mixed-signal simulator using a coupler module written in Verilog-AMS. This is not restricted to a fixed time step. Using two different synchronization schemes, Simulink's variable sample times as well as framed signals are supported. Virtuoso AMS Designer supports the standard HDL languages as well as netlist formats, using either a SPICE or FastSPICE solver.

3 Concept and implementation

The three main questions to answer before implementing the simulator coupling are the following: How can the different simulation algorithms be linked together? How should the user interface be designed? How should both simulators communicate with each other?

Because the co-simulation will operate in the time domain, the first aspect is a question of synchronization. This is discussed in the Synchronization section (3.3).

3.1 User interface

The co-simulation user interface must provide an easy and familiar way to define the signal data that will be exchanged between the two simulation environments as well as the sampling mode and some general options. These requirements were achieved by introducing special coupler modules that are placed in the design of each of the coupled model parts, the coupler representing the AMS block is placed in Simulink and the coupler representing the system design is placed in the schematic editor as shown in Figure 2.

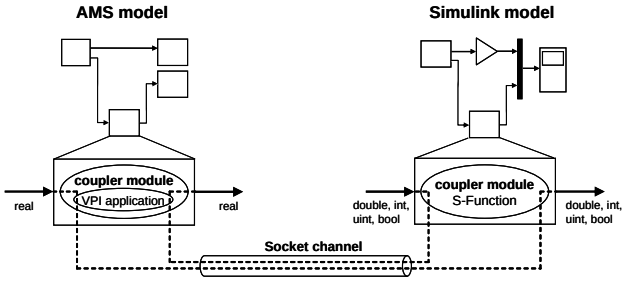

Figure 2: Co-simulation principle

The coupler module represents the design that resides in the other environment and is connected to all signals that should be transferred through the co-simulation. All

options including the number of input and output ports of the coupler module, sampling mode and network connection parameters are set via the parameter fields of the coupler modules; no other configurations or additional modules are needed. This approach allows the user to keep the modular structure of his model when splitting it for co-simulation. Moreover, it enables full support within both the GUIs and easy handling.

The AMS coupler module functionality is split into the Verilog-AMS module code and a simulator interface application that uses the Verilog Procedural Interface (VPI) to exchange data with the simulator kernel. This VPI application is initialized by the system task $couple_init within the Verilog-AMS module:

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

The coupler module also provides variables of type *real* for each input and output port that are accessed by the VPI application. Beside that, it is possible to add any additional functionality to the module.

Currently, two operation schemes are available for the Verilog-AMS coupler module. The first one is a pure digital mode that maps data directly to the module ports of type *wreal*. The second one uses analog electrical ports and a linear interpolation algorithm for writing the sampled data received from Simulink to the output ports (see also the Synchronization section).

On the Simulink side, the coupler module is implemented using the S-Function API. It is loaded at runtime as a shared library. It contains all functionality from port access to socket communication.

Because the AMS coupler module uses *real* electrical or *wreal* signals, all data is sent as 64-bit double values through the socket channel. Thus, all necessary data type conversion is done on the Simulink side. The Simulink coupler automatically detects the data type of the connected signals and converts data if necessary. No user interaction is needed for this. A further development may be to extend the co-simulation interface by supporting arbitrary length bit vectors using the digital coupler on the AMS side and Simulink's Fixed-Point feature.

3.2 Communication

Both simulators communicate using a TCP/IP network socket connection. This allows the co-simulation to be used on a single computer as well as on different hosts in a network. The platforms currently supported are Linux, Solaris, or Windows for the Simulink part and Linux, Solaris, HP, and AIX for the AMS simulator part. Any combination is also possible allowing cross-platform simulation. The necessary byte-format conversion for different processor architectures is done automatically by the cosimulation interface. When running on a single host, the socket may cause a slight overhead compared to an implementation using pipes or shared memory. This may be a point of further development. However, in most cases the main computing time shadows any time spent on communication, leaving very little room for improvement of the overall performance.

3.3 Synchronization

One of the most important tasks in the design of a timed co-simulation environment is the choice of an appropriate synchronization algorithm. This also includes the communication regime between the involved simulators [8].

An appropriate synchronization algorithm will primarily 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. 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 which is repeated for each sample period. Also the AMS model represented by the Simulink coupler block must be executed accordingly to this order and must calculate its outputs for that sample period. Thus, the synchronization time points have to be given by Simulink. This results in a conservative synchronization approach [9][10].

Consequently, the master-slave principle was chosen for synchronization and connection initialization with Simulink as the master and the AMS simulator as the slave. That means, the master simulator acts as server for the socket connection, determines the co-simulation parameters like number and data types of the coupled signals, and controls the length of the synchronization time intervals.

During co-simulation, Simulink calculates the coupler module's input data for the current time step and sends this data together with the time of the next sampling point to the AMS coupler. AMS 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 an initial value. In the opposite direction the input signals of the Verilog-AMS coupler module are sampled at the time points given by Simulink (see Figure 3). The sampling rate can influence the co-simulation performance significantly and should therefore

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 special frame-based signals [11]. These compose several successive samples into 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.

The cosimulation interface supports Simulink's different sampling modes including fixed and variable step, single sample and frame-based signal processing. For more flexibility two different synchronization algorithms for framed and unframed data were introduced. In unframed mode (Figure 3) the Simulink coupler module does not exchange data at simulation time t_0. That means that the input sample of the Simulink coupler at time t_0 is ignored and a value of 0 is written to the outputs. At time t_1 the Simulink coupler input sample is then sent to AMS together with the current simulation time. AMS advances simulation from t_0 to t_1 using the received Simulink data as coupler output. Due to the interpolation algorithm, the value of the received sample is only reached at time t_1 at the output of the Verilog-AMS coupler module. The Verilog-AMS coupler then samples its input signals and sends these values back to Simulink. Beside the initial step the time axes of both simulators are synchronous. This synchronization scheme is also applicable for variable sample times because Simulink only needs to know the current simulation time when sending data to AMS.

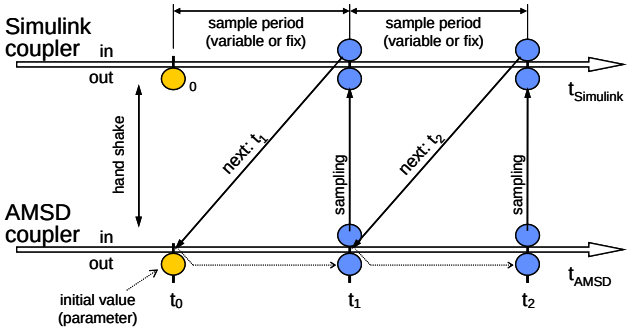

Figure 3: Sample-based synchronization

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 frame period that follows the current time. Thus, the first frame is sent at simulation time t_0 from Simulink to AMS for the interval from t_0 to t_1 (Figure 4). The Simulink coupler 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 Verilog-AMS coupler module the incoming data frame is split into the single data points. The AMS 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. Finally, Simulink applies the received data set of the first frame (t_0 to t_1) to the coupler's outputs. After finishing all remaining block evaluations for the first frame period, the simulation time is advanced to t_1.

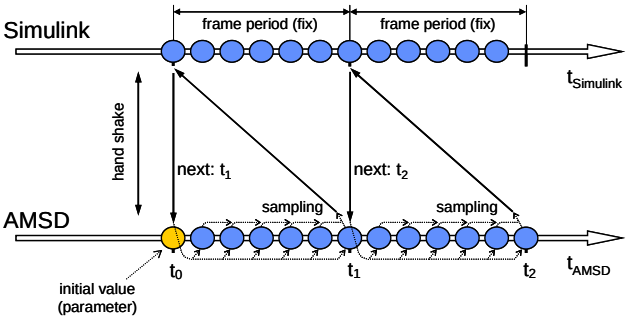

Figure 4: Frame-based synchronization

In contrast to the sample-based synchronization the very first Simulink data point at t_0 is available within the first data frame. Due to interpolation, the level of this first data sample will only be reached at the end of the first sampling period inside AMS (second circle in AMSD line in Figure 4). Comparing the two simulator time axes directly, this appears as one sample delay. However, in both, framed and unframed mode, there is no delay visible between the input and output data of the coupler blocks inside both simulation environments.

There may be additional delays inserted by the dataflow scheduling of the Simulink solver to dissolve feedback loops inside the model. This is necessary to provide the required number of samples/frames at all input ports of a block before the outputs can be calculated. If the top-level of the model resides in AMS and the outputs of the Simulink coupler are fed back into the Simulink coupler's inputs, this behavior needs to be taken into account.

4 Wireless LAN design example

This section shows the co-simulation of a wireless LAN IEEE 802.11b physical layer transmission system. The analog receiver part used is described at different levels of abstraction, namely behavioral baseband, behavioral passband and circuit level.

4.1 Design in Simulink

Figure 5 shows the top-level schematic in Simulink of the physical layer system level model, as part of the RF Kit [12]. The modulated Complementary Code Keying

(CCK) signal is generated in the left block. It contains a binary random source for payload data, CCK coding, pulse shaping filters and other signal processing. The complex signal is split into the inphase and quadrature parts (I/Q) on the baseband level. These two real signals are transferred to the schematic testbench shown in Figure 6. After the analog simulation the signals loop back into the Simulink environment for demodulation and postprocessing.

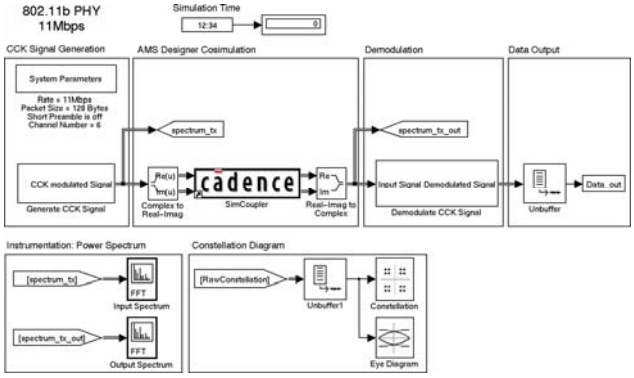

Figure 5: System-level model in Simulink

4.2 RF front-end design

The RF front-end was designed using RF and mixed-signal design and simulation techniques. Figure 7 depicts the implementation of the RF receiver module, which filters, down converts, and amplifies for the RF as well as for the analog baseband part. Beside the circuit level and passband behavioral level the RF parts are modeled in the complex baseband domain as Verilog-A behavioral models to increase simulation performance. It is possible to simulate the whole system at the transistor level, however, the simulation performance will be much slower. Replacing only the RF parts by baseband behavioral models [13] and leaving the analog baseband blocks at the transistor level increases the simulation performance by orders of magnitude.

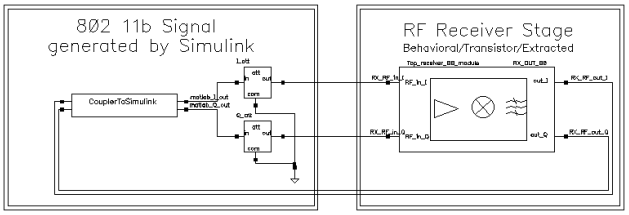

Figure 6: Schematic of RF front-end testbench

One- and two-tone sources are typically used in this environment as stimuli for the analysis of the RF subsystems. Characteristics of the design are for example intercept points, noise figure and corner frequencies. However, realistic stimuli and postprocessing algorithms like CCK, DQPSK, QAM etc. are very complex to realize on this level of abstraction. In most cases, it is much easier to implement this in a system level environment using Simulink. Stimuli could be modulated signals, and corresponding DSP post processing blocks can be used for performance evaluation. Those tests are set up easily using the co-simulation technique.

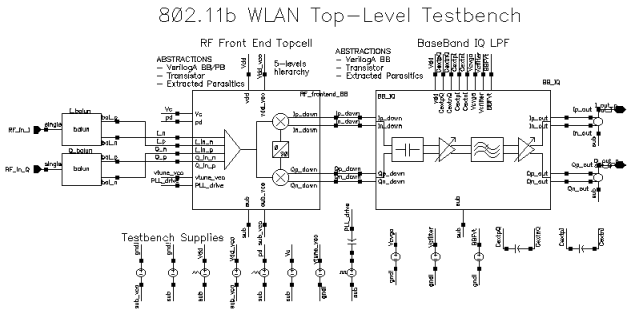

Figure 7: Schematic of RF and analog baseband design

4.3 Setting up and running co-simulation

First, the system level design and the block level design are verified in their related simulation environments. The next step is to add the coupler block into the Simulink design. On the properties form the number of input and output pins as well as some other options are specified. The same steps are necessary for the schematic design. Once these two steps are finished, the co-simulation setup is complete.

The data flow in the above example will start from the beginning of the Simulink design, into the input pins of the coupler block (Figure 5). The same data will be provided at the output pins of the corresponding coupler block in the schematic (Figure 6). The flow progresses through the design in the schematic (Figure 6 and Figure 7) back into the input pins of the schematic coupler block. Finally, the signal appears at the output pins of the coupler in Simulink where the final postprocessing occurs.

There are three different methods to run the co-simulation. The first method uses Virtuoso Analog Design Environment and is suited for designers familiar with that environment. After some minimal setup, Simulink and the co-simulation will be started automatically once the simulation is started. The second method is for system designers more familiar with Simulink. In this case the AMS simulator is started in the background while the user is working inside the Simulink environment. Finally, it is possible to bring up both the design environment and Simulink and start the cosimulation manually in both environments.

Whichever method is chosen for the example above, when the co-simulation starts, MATLAB waveform windows appear displaying the spectrums of the input and output signals and constellation and eye diagrams on the

outputs (Figure 8). The constellation diagram is very helpful to verify the quality of the received signal. The impact of noise, non-linearity, I/Q imbalance and other signal deviations is visualized directly. Depending on the transmission standard some signal processing is necessary to generate the constellation diagram. The Simulink blocksets provide dedicated functions for the generation of such testbenches. The constellation in Figure 8 shows the four symbols of the demodulated signal clearly separated with some noise. Bit errors are not detected during this simulation.

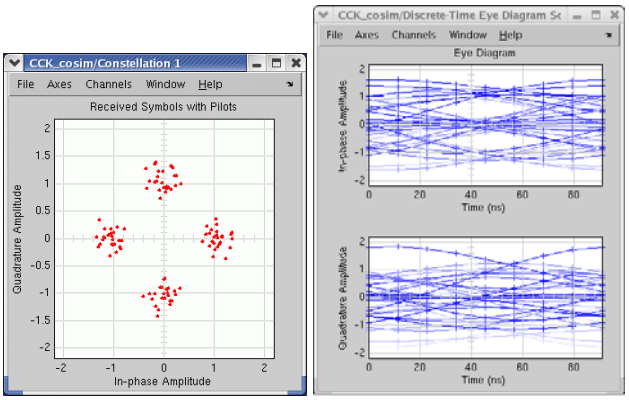

Figure 8: Simulink constellation and eye diagram

5 Summary

The co-simulation closes the gap between system level design with Simulink and mixed-signal block level design. The synchronization scheme between the signal flow simulator and the mixed-signal simulator has been described in detail for both framed and unframed modes. Due to the support of Simulink's framed signals, the co-simulation interface profits from the reduced communication overhead resulting in a performance increase depending on the frame size. This is an advantage compared to other co-simulation solutions that do not support frames.

The application example demonstrates the two main goals of the co-simulation – the integration of lower level mixed-signal blocks into the system simulation and the use of Simulink for stimuli generation, postprocessing and visualization for mixed-signal testbenches.

With the co-simulation, the designer can replace selected parts of a system-level model with more detailed behavioral or circuit-level blocks. This is helpful in a top-down design flow, where the system-level model is used as an executable specification. During concept engineering the impact of different architectures, e.g. of the RF front-end, can be evaluated within the system. Later in the design process the implemented block can then be verified within the system-level model.

The second goal is the use of system-level models as a testbench for block implementation. The co-simulation approach enables the analog/mixed-signal designer to access realistic and possibly highly complex input stimuli early in the design flow along with the related postprocessing algorithms.

The co-simulation approach makes it possible to evaluate lower level blocks inside a system-level simulation to achieve accuracy where needed yet still maintain good performance. Simulating such a complex system-design at the transistor level is not even possible within the design cycle time and simulating the entire design at the system level is not as accurate as transistor-level simulations. With the presented co-simulation approach, performance and accuracy tradeoffs can easily be made by simulating the accuracy-needed blocks at the AMS level and the other blocks at the system level.

Acknowledgements

Special thanks to Xiao Wang for his work on the Virtuoso Analog Design Environment integration, Tao Hu for his work on the Simulink driven use model, Lei Song for his help creating the tutorial, Ruilin Yue for testing the flows and creating regressions, and Loren Pahlke and Arti Warikoo for creating the documentation.

References

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

[2] 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

[3] The MathWorks: *Link for ModelSim*, http://www.mathworks.com/products/modelsim/

[4] A. Milik, A. Zytka: *HDL Simulation and Mathematical Modeling Integration.* TechOnLine, May 2006, http://www.techonline.com/community/39960

[5] Mentor Graphics: *SystemVision*, http://www.mentor.com/products/sm/systemvision/

[6] Synopsys: *Saber*, http://www.synopsis.com/products/mixedsignal/saber/

[7] R. Juchem, B. Knorr: *Complete Automotive Electrical System Design.* Proc. Vehicular Technology Conference 2003 Fall, Vol. 5, pp. 3262-3266, Orlando, Oct. 2003

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

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

[10] U. Donath, J. Haase, R. Gruschwitz, G. Kurth, P. Schwarz: *Parallel Multi-Level Simulation with a Conservative Approach.* J. Systems Analysis - Modelling - Simulation 21(1995), pp. 187-201

[11] The MathWorks: Simulink Signal Processing Blockset, http://www.mathworks.com/products/sigprocblockset/

[12] Cadence Design Systems: RF Design Methodology Kit, http://www.cadence.com/products/kits/RF_Design/

[13] M. C. Jeruchim, P. Balaban, K. S. Shanmugan: *Simulation of Communication Systems.* Second Edition, ISBN 0-306-46267-2, New York: Kluwer Academic/Plenum Publishers, 2000