A System-Level Co-Verification Environment for ATM Hardware Design

Guido Post, Andrea Müller and Thorsten Grötker
Institute for Integrated Signal Processing Systems
RWTH Aachen, University of Technology
D-52056 Aachen, Germany

Abstract

Common approaches to hardware implementation of networking components start at the VHDL level and are based on the creation of regression test benches to perform simulative validation of functionality. The time needed to develop test benches has proven to be a significant bottleneck with respect to time-to-market requirements. In this paper, we describe the coupling of a telecommunication network simulator with a VHDL simulator and a hardware test board. This co-verification approach enables the designer of hardware for networking components to verify the functional correctness of a device under test against the corresponding algorithmic description and to perform functional chip verification by reusing test benches from a higher level of abstraction.

1 Introduction

One of the main challenges of future design practice is the integration of point tools that address different areas of hardware and software design. Recent studies have shown that the digital system design industry loses approximately $4.5 billion each year in nonproductive time due to a lack of interoperable tools [1]. Common approaches for validation of VHDL-based hardware models [2] and the final hardware implementation are based on the creation of regression test benches to perform verification of timing and functionality by simulation [3]. Although formal verification methods evolve [4] [5] that can symbolically prove the correctness of an implementation, there is still a problem in state explosion for complex systems. Thus simulation is still the traditional way of verifying implementations of models, both at the system level and at the implementation level. The time needed to develop test benches for both, VHDL simulations and functional hardware test has proven to be a significant bottleneck (up to 50% of the design time). The increase in test vector complexity about 100x every year compared to an increase of design complexity about 4x over the same period will further aggravate the problem of test vector generation and evaluation.

The complexity inherent in telecommunication systems requires the design and evaluation of system aspects at different levels of abstraction. Furthermore, the hardware and software which make up the system have to be designed concurrently [6]. Yielding high performance of ATM equipment requires to realize most functions operating on the ATM cell stream in dedicated hardware, e.g. ASIC or FPGA. ATM is the application that drives the ASIC technology as well as state-of-the-art design and verification methodologies to its limits [7]. This is because HW functionality, that includes the largest part of ATM traffic management and physical layer tasks, is interacting with the complexity of embedded control software, that implements higher-layer functionality, such as call admission control agents and signaling protocols. Furthermore, the HW functionality itself is distributed over a number of hardware devices. The verification over several layers of functionality and different time scales is one of the greatest challenges for ATM hardware designers and has to be performed at each level of model abstraction [8]. In order to overcome the existing verification gap for hardware designs in the networking domain it becomes necessary to provide a link between state-of-the art network simulators, HDL simulation tools and the hardware prototype [9]. The main motivation is to model and reuse test benches at a higher level of abstraction in order to cope with the increasing test bench complexity.

This paper is organized as follows. In section 2 an approach for functional verification of ATM hardware against algorithmic behavioral descriptions in a telecommunication network simulator is presented. Section 3 describes the coupling of the network simulator OPNET from Mil3 with Synopsys' VHDL System Simulator (VSS) and a hardware test board that connects the prototype hardware. Major points in this section are the provisioning of abstraction interfaces and proper mechanisms for simulator synchronization. Finally, we discuss the implementation and present our conclusion.

2 ATM Co-Verification Environment

Figure 1 illustrates the co-verification environment for ATM hardware design. State-of-the-art network simulation tools provide means for modeling and simulating the operations of complete networks [10]. Tools that fall into this category include the Block Oriented Network Simulator (BONeS^{TM}) by Cadence Design Systems,

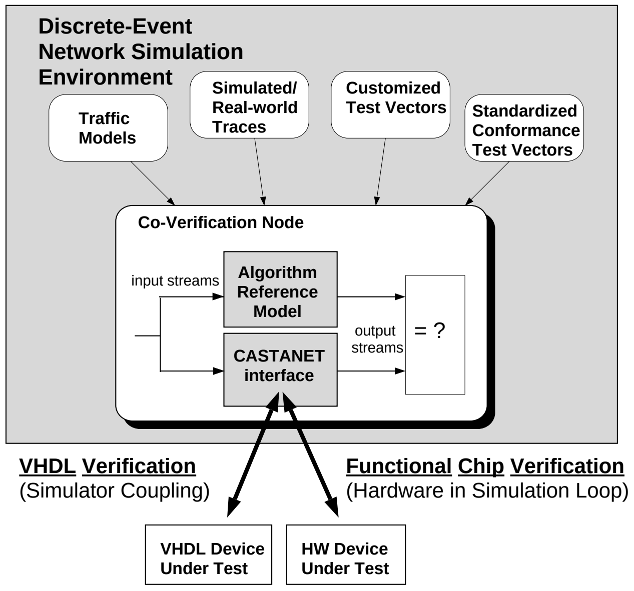

Discrete-Event Network Simulation Environment
Traffic Models
Simulated/ Real-world Traces
Customized Test Vectors
Standardized Conformance Test Vectors
Co-Verification Node
input streams
Algorithm Reference Model
CASTANET interface
output streams
= ?
VHDL Verification
(Simulator Coupling)
Functional Chip Verification
(Hardware in Simulation Loop)
VHDL Device Under Test
HW Device Under Test

Figure 1: Environment for System-Level Co-Verification

SES/Workbench by Scientific and Engineering Software and OPNET Modeler from MIL3. These tools are optimized to support the modeling of traffic sources [11] and the performance evaluation of abstracted higher-layer protocols and distributed communication devices. We chose OPNET because of its ATM model suite and library of traffic models. In this environment the algorithm design is carried out by building system level descriptions at different hierachical levels (called network-, node and process domains). The network domain specifies the topology of a networking architecture in terms of high-level devices (called nodes) such as switches and traffic sources, and communication links between them. Within the node domain each node's capability is described in terms of processing, queueing and communication interfaces. The process domain specifies the behavior of processing nodes as communicating extended FSMs. System behavior and performance can be analyzed by means of discrete event simulations. Effective traffic modeling [11] for system analysis has become crucial for the design process of networking hardware from algorithm design to implementation. Because there exists strong dependencies between decisions at the system level and hardware costs of their actual implementation there is no one way (top-down) transition from higher to lower levels of abstraction. Since both, algorithm and system architecture affect end-to-end performance of user traffic they can not be considered separately and have to be analyzed within the network context. Therefore, algorithms and architecture have to be optimized for cost, size, complexity and reliability within an interactive and iterative design process.

VHDL (VHSIC Hardware Description Language) is used for the hardware design at the behavioral and RT-level. Functional verification of VHDL models is performed using Synopsys' VHDL System Simulator (VSS). In order to preserve consistency between the more abstract system level description and the HDL-based implementation we are developing CASTANET, a **C***onfigurable* **A***TM* **S***imulation* **T***estbench* **A***pplying* **Net***work Simulations*. CASTANET establishes a coupling of the network simulator OPNET from Mil3 to the VSS and to a hardware test board for co-verification. In general, our interface models could be coupled to any discrete event based network simulator. The OPNET-VHDL simulator coupling enables the designer of hardware for telecommunication networking components to verify the functional correctness of a VHDL description against the corresponding algorithm reference model in OPNET that has been used to evaluate the performance of the system. Functional chip verification can be performed by interfacing to a hardware test board that allows to stimulate the hardware under test with real-time test patterns. These can be either stochastic traffic models or simulated real-world traces, for example MPEG traces in order to investigate the hardware's behavior under various environmental scenarios. Another category of stimuli is targeted towards testing of hardware properties through customized or standardized conformance test vectors.

This approach significantly reduces the time to construct test benches because it reuses existing test patterns and model descriptions that are available in the network simulation environment. In addition to enabling the modeling of software components at a higher algorithmic level in C or C++, it descreases simulation time compared to pure VHDL-based test benches. For example, the simulation run time for processing 10.000 ATM cells arriving at an ATM switch consisting of four ports modules, one global control unit on an UltraSparc (Solaris2.5) is approx. 30 seconds. This is equivalent to approx. 8.300 clock cycles per second. Taking the simulation time needed to simulate solely an RTL representation of the global control unit this results in approx. 300 clock-cycles per second. The proposed co-verification environment offers the following advantages:

- Functional verification of VHDL models and hardware in the system environment by reusing existing test patterns and traffic models from network simulations.
- Access to powerful analysis capabilities available in existing network simulation tools for the representation of errors and results, and in HDL simulators for depicting waveforms.
- Ability to model, simulate and analyze the system environment at the most appropriate level of abstraction.
- Support of iterations between system and implementation-level design tools to explore the design trade-offs.

3 Functional Hardware Verification

Figure 2 shows the concept and realisation of CASTANET. The coupling will be done by a special OPNET

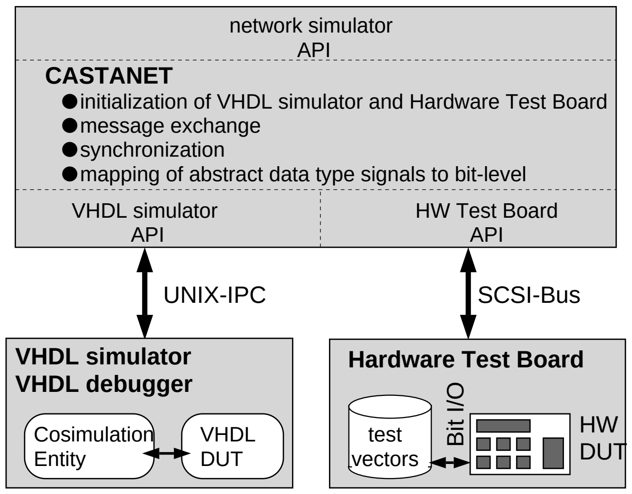


Figure 2: Functional Verification using Co-Simulation and Hardware in the Simulation Loop

interface model that steers either a VHDL simulation or the hardware test board with test-patterns from the network simulation. The CASTANET interface process in OPNET manages the proper initialization of the VHDL simulator and the hardware test board and handles the message exchange via standard UNIX inter process communication (IPC) and SCSI-bus respectively. For this task it uses API (application programming interface) functions of the CASTANET library. In the VSS simulation a C-language based co-simulation entity [12] is instantiated, that receives messages from OPNET-side interface process. It also performs signal conditioning, e.g. mapping a data structure to bit or word-level signal streams and generation of additional control signals. The responses from the device under test (DUT) are sent back to the CASTANET interface node and can be compared to the reference model's responses at the system level. Of course, it is possible to run the simulation in the background while dumping the output data into a file and to re-run previously generated test vectors.

One can identify three major points that needs to be addressed for the realization of the proposed co-verification approach. These are

- the synchronization of simulators running in parallel,
- the conversion of high-level abstract information flows to bit-level signal representations in VHDL-based hardware models, and
- the scheduling of hardware test-cycles to the hardware device under test taking into account a proper mapping of bit-level signals to the HW device pins.

3.1 Synchronization of OPNET and VHDL Simulators

The simultaneous execution of OPNET with a VHDL simulator is a special case of parallel distributed discrete-event (DE) simulation [13] [14]. A difficult problem in distributed DE simulations is the avoidance of deadlock. DE simulators manages their events via an event list that represents the event distribution over time and maintains a proper time-ordering. Events are executed in a monotone nondecreasing order of time stamps that are associated with each event. Thus, they may be generated for any future time, or the current time but never for past times. Because both simulators are event-driven and therefore both have a notion of time, one has to ensure that neither of them produce events in the past of the other simulator's time basis, see for example figure 3. Synchronization mechanisms for distributed computation or simulation can be classified in *conservative* and *optimistic* approaches. In the class of conservative methods [13] the independent advancement of the local clock is prohibited. Advancement of the local clock is granted only when it is ensured that the distributed computation is globally correct up to that point in time. Optimistic methods, on the other hand, do not exclude causality errors. Local time is allowed to advance independently until a causality error occurs. This implies that a simulator has to be resynchronized leading to a rollback of the simulation time [15]. Despite the fact that optimistic methods potentially can achieve a larger speed-up, the memory requirements for the storage of the simulator turns out to be very large.

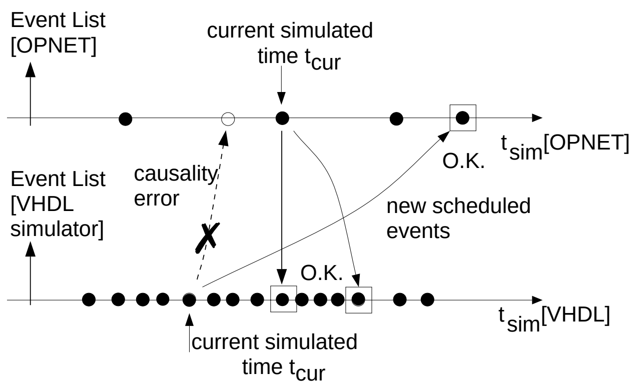


Figure 3: Scheduling of events in a parallel DE simulation

Therefore we selected a conservative synchronization protocol for the communication between OPNET and the VHDL simulator that is based on the computation of timing windows. This synchronization protocol is tailored to the application-specific co-simulation problem. Communication between both simulators is based on the exchange of time-stamped messages updating the receiving simulator with the current simulation time of the originator. For each input message type the co-simulation entity maintains a time-stamped message queue I_j. Furthermore, for each message type the maximum number of clock cycles δ_j that it takes to process the message has to be specified by the user. Because events (messages) for different input message queues arrive sequentially, the co-simulation entity has to ensure that no other VHDL port receives data before advancing its simulation time. Upon receipt of a message with a time stamp t_k for input queue I_j and $t_k > t_{cur}$ (t_{cur} -

current simulated time) the VHDL simulator is allowed to process all events with a time stamp smaller than t_k, but not equal. Following, the current simulation time is updated to $t_{cur} = t_k$. The message at queue I_j remains queued until all other input queues received messages with time stamp t_k or an event with a greater time stamp arrives at an arbitrary message queue. In the first case the local simulation time is advanced by the minimum of each message type's processing delay δ_j. Applying this strategy the simulated time of the VHDL simulator always lags behind OPNET's simulated time. The use of this specific conservative synchronization protocol resolves the possibility of deadlock.

3.2 Abstraction Interfaces

An important point which needs to be considered for the co-simulation of system level network simulators and implementation level HW simulators is the different granularity of time scales. Time units in network simulations can be derived from cell time, whereas the time unit in HW systems is fixed by the HW clock steering bit-level operations. Considering Asynchronous Transfer Mode (ATM) as an example application, one can identify time-periods where idle cells are inserted into the ATM cell (one cell comprises 53 octets) stream. This means that there is a ratio of 1:100 for a simulation time step in OPNET and VSS. Incorporating the HW-clock into the OPNET interface model would unnecessarily slow down the system-level simulation. Therefore, intelligent abstraction interfaces are needed that cope with the different granularity of time scales.

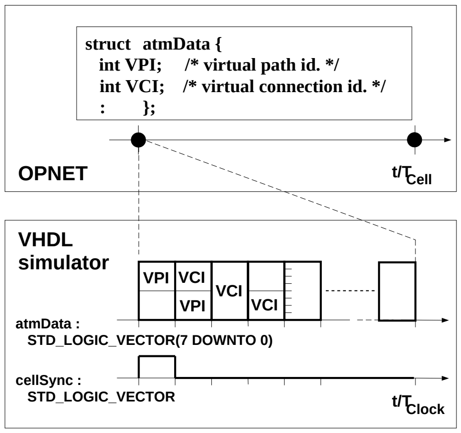

struct atmData {
int VPI; /* virtual path id. */
int VCI; /* virtual connection id. */
: };
OPNET
t/T_Cell
VHDL simulator
VPI
VCI
VCI
VPI
VCI
atmData : STD_LOGIC_VECTOR(7 DOWNTO 0)
cellSync : STD_LOGIC_VECTOR
t/T_Clock

Figure 4: Mapping of ATM packets to VHDL ports

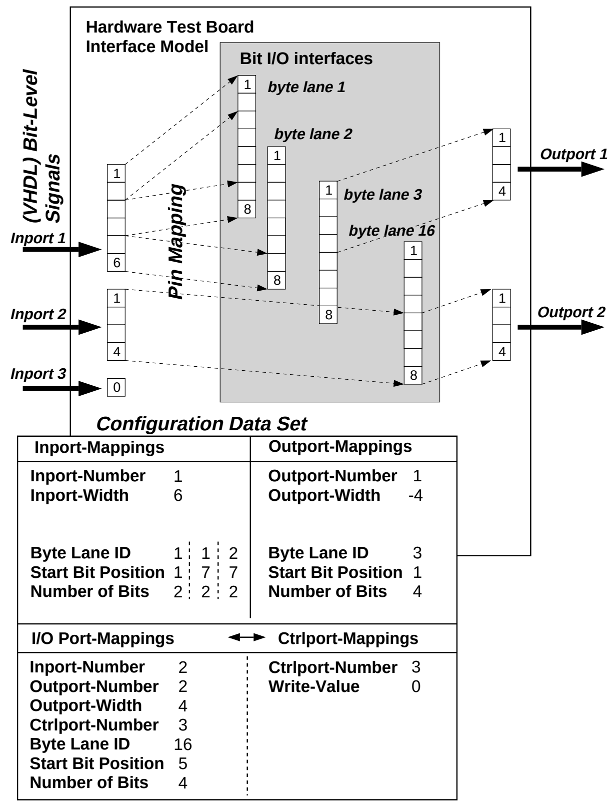
In a network simulator processes communicate through the exchange of abstracted information described for example as C-structures. The communication is instantaneous, e.g. when an event occurs the complete information is available for further processing. At the implementation level communication interfaces are represented by their structure, e.g. number of signals and pins, and their timing behavior, e.g. number of clock cycles to process input or output values and additional handshake protocols. As an example, the mapping of an OPNET packet that arrives on the input communication channel of the CASTANET interface process to a 8-bit wide VHDL port signal is illustrated in figure 4. The complete ATM cell comprises 53 bytes, therefore it takes 53 clock cycles within the hardware simulator to read the cell. Additionally, the interface model generates control signals such as a cell synchronization signal that indicates the start of a new cell. The user has to specify how high-level protocol data units and abstract data types has to be mapped to bit-level signals using appropriate conversion functions that are provided in the CASTANET library.

3.3 Functional Chip Verification

As long as one does not run the hardware at the targeted speed its behaviour can not be fully verified, because timing violations are not likely to be detected. Therefore *real-time verification* is essential before integrating components and subsystems into the target system. State of the art is to build up dedicated hardware test benches, since there is no idea of in-field debugging, because of the complexity of the verification task. This is a very error prone procedure and increases the effort spent on a design in terms of the time needed for constructing and validating the hardware test bench.

In [16] a configurable and flexible hardware test board has been presented allowing to run consecutive test cycles at real-time speed. The hardware test board consists of a control part and multiple memory units for intermediate data storage of test vector. It provides a bit stream interface and a clock interface to which the hardware device under test is connected, see figure 2. The bit stream interface consists of 128 I/O-pins, where each of 16 byte lanes is configurable in direction and speed. In the current implementation the maximum board clock is 20 MHz. The real-time verification process consists of repeated *hardware activity cycles*, interrupted by a *software activity cycle*, in which the hardware is stopped immediately. One *test cycle* contains a software activity cycles to generate stimuli, configure the board and store stimuli to the hardware test board. This is followed by a hardware activity cycles to run the hardware under test and a software activity cycle to read the results back to the simulator. Test cycles run repeatedly until the simulation is finished.

The hardware that is hooked to the hardware test board is connected to the OPNET simulation via a CASTANET interface model that is configurable with repect to the clock gating factor and the duration of one hardware test cycle. The current memory configuration of the hardware test board supports test cylce durations between 1 and 32.000 clock cycles. The hardware test board allows to interface unidirectional hardware ports as well as bidirectional ports, e.g. μP or bus interfaces. Since bit-level data flows are generated at an unidirectional single source, bus interfaces need to be modeled by three bit-level signals – input, output and a control signal indicating the direction through predefined read/write flags. The duration of each hardware test cycle is automatically calculated from the actual values at the control ports. The signal mapping of bit-level signals to the hardware test board pins is specified in a con-



Inport-Mappings					Outport-Mappings	
Inport-Number	1				Outport-Number	1
Inport-Width	6				Outport-Width	-4
Byte Lane ID	1	1	2		Byte Lane ID	3
Start Bit Position	1	7	7		Start Bit Position	1
Number of Bits	2	2	2		Number of Bits	4

I/O Port-Mappings		←→ Ctrlport-Mappings	
Inport-Number	2	Ctrlport-Number	3
Outport-Number	2	Write-Value	0
Outport-Width	4		
Ctrlport-Number	3		
Byte Lane ID	16		
Start Bit Position	5		
Number of Bits	4		

Figure 5: Configuration of Pin Mapping to HW device under test

figuration data set, see figure 5. The configuration data set collects the information in terms of byte lane ID, start bit position and number of bits, provided by the user to automatically establish the input port mapping, output port mapping, I/O port mapping and the associated control port mapping.

4 Summary and Conclusions

We have presented a co-verification environment for ATM hardware design. The main motivation is to model and reuse test benches at a higher level of abstraction to cope with the increasing test bench complexity. Our approach is to provide a link between a state-of-the-art network simulation tool and hardware simulation tools. For this task we are developing CASTANET, a **C***onfigurable* **A***TM* **S***imulation* **T***estbench* **A***pplying* **Net***work Simulations*. It implements a simulator coupling between OPNET from Mil3 and Synopsys' VHDL System Simulator (VSS) and a dedicated hardware test board respectively. Major tasks are the provisioning of synchronization protocols between parallel simulators and the mapping of high-level abstract information flows to bit-level signals and hardware device pins. With these communication and synchronization mechanisms a wide range of applications, especially in ATM traffic management sector, can be covered. We have used CASTANET for the functional verification of an ATM accounting unit. To support the development of interface modules for OPNET and VHDL simulators in the future proper interface description needs to be developed. Based on this description, core interface models can be automatically generated. Building blocks will be taken from a library of generic protocol classes and conversion routines. Because of the time scale problem, event-driven VHDL-simulators are obviously a bottlenck in the co-verification process. Moreover, the number of events that event-driven simulators have to evaluate is an order of magnitude higher compared to the system-level simulation in OPNET. Thus, the integration of cycle-based simulation techniques is required, as well the development of design methodologies that make cycle-accurate modeling sufficient.

Acknowledgements

Part of the work presented in this paper has been sponsored by the Ministerium für Wissenschaft und Forschung, NRW under contract IV A3 - 107 020 96 - and by the European Community (ACTS AC039).

References

[1] V. Madisetti, "Rapid digital system prototyping: Current practice, future challenges," *IEEE Design & Test of Computers*, pp. 12–22, Fall 1996.

[2] Institute of Electrical and Electronics Engineers Inc., 345 East 47th Street, New York, NY 10017, USA, *IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1987*, March 1988.

[3] R. Camposano and J. Wilberg, "Embedded System Design," in *Design and Automation for Embedded Systems*, vol. 1, Kluwer Academic Publ., Jan. 1996.

[4] R. Bryant, "Binary decision diagrams and beyond: Enabling technologies for formal verification," in *Proc. of the IEEE Int. Conference on Computer Aided Design*, pp. 236–243, November 1995.

[5] C. van Eijk and J. Jess, "Exploiting Functional Dependencies in Finite State Machine Verification," in *Proceedings of the European Design & Test Conference*, (Paris, France), March 1996.

[6] G. Mancini, D. Yurach, and S. Boucouris, "A Methodology for HW-SW Codesign in ATM," in *DAC'95*.

[7] J. Conesa, "Design Challenges of High Speed ATM Communication ASICs," in *Proc. of the European Design & Test Conference*, 1996.

[8] A. Sangiovanni-Vincentelli, P. McGeer, and A. Saldanha, "Verification of Electronic Systems," in *DAC'96*.

[9] G. Post, A. Müller, and H. Meyr, "High-level Validation Strategies for Broadband Network Components: A Case Study on Charging Algorithm Design," in *IEEE High-Level Design Validation and Test Workshop*, (Oakland, CA), November 1996.

[10] A. Law and M. McComas, "Simulation Software for Communications Networks: The State of the Art," *IEEE Comm. Magaz.*, pp. 44–50, March 1994.

[11] J. Ferranto, "Traffic Modeling Techniques for Discrete-Event Simulation," in *Proc. Int. Conf. on Signal Processing Application and Technology*, pp. 647–652, 1997.

[12] Synopsys, Inc., 700 E. Middlefield Rd., Mountain View, CA 94043, USA, *VHDL System Simulator Interface Manual*.

[13] K. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *IEEE Transactions on Software Engineering*, pp. 440–452, Sept. 1979.

[14] D. Jefferson, "Virtual Time," *ACM Transactions on Programming Languages and Systems*, vol. 7, pp. 404–425, July 1985.

[15] V. Madisetti and D. Hardaker, "Synchronization mechanisms for distributed event-driven computation," *ACM Transactions on Modeling and Computer Simulation*, vol. 2, no. 1, pp. 12–51, 1992.

[16] A. Müller, G. Post, M. Vaupel, and H. Meyr, "RAVEN - A Realtime Analysis and Verification Environment," Proc. of DSP Deutschland 97, October 1997, Munich.