System level simulation - a core method for efficient design of MEMS and mechatronic systems

Peter Schneider, Christian Bayer, Karsten Einwich, Andreas Köhler
Fraunhofer IIS, Design Automation Division
Zeunerstr. 38, 01069 Dresden
e-mail: peter.schneider@eas.iis.fraunhofer.de

Abstract— **Design of micro systems, MEMS or mechatronic systems is dominated by the interaction of effects from different physical domains. One important approach to decrease the number of design cycles significantly is system level modeling and simulation. The main challenges for the efficient use of modeling and simulation are a systematic approach for behavioral modeling, automated model generation as well as powerful simulation frameworks. Especially for the latter flexible handling of different models of computation is crucial.**

Keywords: Modeling methodology; System Level Simulation; Model Generation; Model Order Reduction

I. INTRODUCTION

The design of micro systems, MEMS or mechatronic systems is characterized by a variety of design approaches for subsystems and components. Furthermore, it is dominated by coupled effects from different physical domains. System integration is often based on experimental setups built up from prototypes of the system components (transducer, electronics, and software). Design errors, unclear specification and incompatibilities between subsystems are therefore often identified very late in the design process and cause additional cost and design cycles.

One important approach to decrease the number of such cycles significantly is system level modeling and simulation. The main goal is to enable the designers to establish a common understanding at an early stage, to refine and adjust the specification and to provide reference models of subsystems to support the design of specific components. This basic approach requires a comprehensive view on different sub-domains and on the corresponding design flows and their mapping on an appropriate mathematical approach, which is suitable for system level simulation. Furthermore, aspects like simulation performance, model accuracy, modeling effort and model verification as well as IP protection are of particular importance. Hence the overall efficiency of a system level simulation approach is crucial for the practical applicability of design techniques like Monte Carlo analysis, parameter optimization and design centering.

Today the main challenges for efficient application of modeling and simulation in the design process of complex MEMS are systematic model generation and appropriate modeling techniques as well as the availability of powerful simulation environments.

In this paper a systematic approach for model generation and the combination of modeling methods with parametric model order reduction are presented together with a flexible simulation framework suitable for system level design of MEMS.

II. MODELING AND SIMULATION IN MICRO SYSTEM DESIGN

The current approach for micro system design is dominated by the use of a variety of simulation tools supporting specific modeling techniques for different physical domains and different levels of abstraction, e.g. PDE solvers for component design and mixed signal simulation for electronics. In the past, there were several research activities on modeling with a focus on bridging the gap between different modeling approaches and providing specific links between those different physical domains [1-6].

Today, modeling languages like VHDL-AMS, Verilog-AMS, Modelica, and Simscape are available for system level simulation. These languages are supported by powerful simulation tools (SystemVision, AMS-Designer, Dymola, SimulationX) and – to a certain degree – are complemented by substantial model libraries. Due to the increasing role of algorithms in complex sensor systems even software components have to be considered in the MEMS design process. This aspect is currently not supported sufficiently by the above mentioned tools.

Concerning behavioral models, there is still a lack of methodological approaches for systematic model generation. Behavior models are a key element for system level simulation. They can be defined for sub-domains of a system with an appropriate degree of refinement and then be connected through interfaces to resemble the whole system. As a consequence, techniques for automated generation of appropriate behavior models are required and have to be developed.

III. SYSTEMATIC APPROACH FOR MODELLING OF MICRO SYSTEMS

Efficient modeling for system level simulation of micro systems requires a unified hierarchical modeling approach derived from [7] and [8] which provides:

- a systematic modeling process,

- the exchange of models of different accuracy including a stepwise model refinement and
- an interface for different modeling approaches.

To ensure a certain amount of flexibility, the integration of special simulation algorithms is also of great interest. For example, selected techniques of co-simulation have to be supported. Basically, there are three main elements of such an approach, which will be discussed in the following.

A. System Analysis and Partitioning

For a systematic partitioning of complex dynamical systems the following strategy has been proved to be suitable:

- Partitioning based on functional aspects with a minimal number of lumped points or areas of interaction, e.g. at electrical connections, shafts in mechanics (functional partitioning),
- Further partitioning into physical domains, like electrical, mechanical, magnetic (physical partitioning),
- Partitioning within the physical domains according to different modeling approaches (network, block diagram, State chart) (mathematical partitioning) and
- Further refinement into geometrical sub components, e.g. channels in a fluid system, magnetic cores in a magnetic network model (geometrical partitioning).

This partitioning can usually be made coarse in a first step and will be refined during modeling and development, respectively.

B. Definition of Model Interfaces

The information exchange between models is enabled by connectors often called model interfaces. In reality, interactions between subsystems result from energetic coupling. For the modeling process these coupling effects between subsystems have to be represented adequately. In case of significant interactions energy flows must usually be considered. Very weak interactions can be ideally represented as directed signal flow, e.g. in block-oriented simulation of control systems. Depending on the kind of interaction conservative quantities, i.e. flow quantities ***i*** and across quantities ***u*** for each conservative connector, and non-conservative quantities, i.e. signals ***a*** for non-conservative connectors, are used. Furthermore, for discrete event simulation connectors might also carry discrete quantities ***d***.

Models should only exchange information concerning physical quantities, which in reality are present at the subsystem boundaries. During an iterative refinement further connectors might therefore be introduced to represent additional quantities. Physical quantities, which are time-invariant and apply globally, are specified as parameters.

C. Behavioral Modeling

Behavioral modeling of MEMS is usually based on a mathematical description which consists of linear or nonlinear ordinary differential equations and algebraic equations in implicit form. This set of generally underdetermined differential-algebraic equations is an implicit description of the component behavior. It is represented by the solution set for the given equations in a physically motivated area of validity. For the determination of single solutions, the model has to be connected with other models or with external elements in order to set appropriate boundary conditions. In general, such a connected or "wired" behavioral model can be represented by

$$F(x(t,p),\dot{x}(t,p),t,p) = 0 \tag{1}$$

with:

$$x : T \times \Re^{n_p} \rightarrow \Re^{n},$$

$$\dot{x} = \frac{dx}{dt},$$

$$t \in T,$$

$$p \in \Re^{n_p}.$$

x is a vector containing all variables of the "wired" behavioral model. t is the time and p is the vector of parameters which are considered to be time-invariant.

The behavioral description of a component is a set of equations which contains the dependencies between the above-mentioned interface quantities u, i and a, which are combined in the vector of terminal variables x_k. Besides those connectors, a system usually depends on a set of inner variables, which are not visible from outside. Thus, the equations of a behavioral model also contain a vector s of internal state variables, which are needed to completely describe the behavior. This description is called the "terminal behavior description" of the component and their solution set is the terminal behavior.

Often, the objective of simulation is to calculate the transient behavior. This might be expressed by the implicit form:

$$F(x_k(t),\dot{x}_k(t),s(t),\dot{s}(t),t,p_k) = 0 \tag{2}$$

with terminal variables $x_k : T \rightarrow \Re^{n_k}$,

and inner state variables $s : T \rightarrow \Re^{n_s}$,

time $t \in T$,

and parameter values $p_k \in P \subset \Re^{n_{p_k}}$.

$\dot{x}_k$ and $\dot{s}$ are the derivatives of x_k and s, respectively. Equation (2) is an under-determined set of equations as long as boundary conditions are not given. Every conservative terminal has two variables referred to as across and flow quantity. Their dependency is normally described by one equation per terminal. There are exceptions if there are constraints on variables, e.g. for Norator or Nullator from network theory [9].

For the calculation of solutions, the boundary conditions are provided either by the connection to other components with their respective equations or by assigning values to the terminal

variables. For that purpose basic elements from simulator model libraries are often used. The overall model is in most cases is therefore a mixture of structural and behavioral models. Hence, modeling conventions, e.g. interface quantities, counting directions etc. between behavioral models and library elements must be consistent.

For behavioral modeling of a component in general, it is not possible to state the needed number or structure of differential equations. The choice of internal variables or needed auxiliary quantities is also undetermined. These aspects strongly depend on the methods used for formulation of equations. In multi-body mechanics, formalisms from Lagrange or Hamilton are used, in electrical engineering e.g. modified nodal analysis is very common. For multi-physics modeling a mixture of methods is also possible.

One important requirement for (2) is that the description of a component has to be complete and consistent, i.e. the integration into equation (1) describing the complete system has to result in a set of equations, which can be solved.

In particular for multi-physical models, where equations are gathered from different sources, an exact analysis of the chains of acting elements and connections is necessary. For system level simulation with automatic partial model generation it is very useful and necessary to establish a common mathematical expression for the equations. A suitable mathematical structure arises from the modified nodal analysis of linear networks in electronics and will be used for that purpose:

$$C(p)\dot{x}_V(t) + A(p)x_V(t) - b_i(t,p) = 0 \qquad (3)$$

Considering physical properties of the system, the equations are formulated so that they fit into the mathematical structure of (3). An appropriate approach, which worked satisfactorily within numerous modeling tasks, is described in [10].

IV. Modeling Methods

For complex micro systems especially, modeling based on the described unified mathematical approach must be supported by appropriate modeling methods. Figure 1 gives a very rough overview of possible ways resulting in different kinds of models.

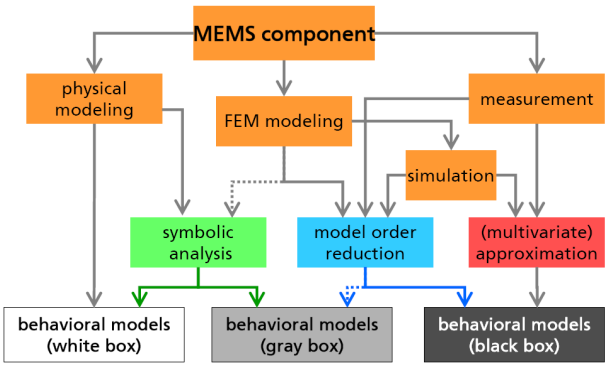

Figure 1: Modeling methods for MEMS [11]

Physical modeling based on equations from text books usually leads to so-called white box models, which have parameters with a clear correspondence to the behavior of the system. For complex sets of such equations, e.g. for a huge number of combined basic models, symbolic methods can be applied for simplification. On the other side multivariate approximation leads to models, which represent a rather fixed mapping of input and output variables. Approximation is either based on measurements or on detailed simulation, e.g. with FEM.

Model order reduction (MOR) is a method in between. The most important application for model order reduction is the integration of detailed models from component design into system level modeling. In MEMS design, this is of particular importance as finite element models for micro-mechanical structures usually reach state space dimensions N of 10^5 to 10^7. MOR is a mathematical method to reduce the number of internal states according to equation (2). Terminal variables are preserved and remain as an interface to other models. As a result, simulation times reduce dramatically, while the introduced approximation error is negligible.

Our MOR method of choice is multi-point moment matching based on ration Krylov subspace methods. During the last decades, this approach proved as an efficient way to reduce the state space dimension of large scale dynamical systems [1-6]. Exploiting the sparsity structure of the system matrices, Krylov methods have a numerical cost of only $\mathcal{O}(N^2)$. The lack of a global error bound does not carry weight in our applications where the signals have a limited bandwidth and local convergence is sufficient.

For the application of rational Krylov methods, expansion points and the number of moments per expansion point had to be selected manually by experienced users. Having a push button solution in mind, we developed an adaptive rational Krylov subspace based model order reduction algorithm AMPXT that automatically selects expansion points and moments to be matched based on an error indicator.

Recently, AMPXT has been extended for rapid parameter sensitivity computations [1]. This development was motivated by a growing demand for tools incorporating manufacturing tolerances during simulation and design for process variation, yield analysis, reliability studies and design optimization.

A. *Basic Methodology*

Our starting point is a description of the model as a linear time-invariant descriptor system. Spatial discretization of partial differential equations like the heat equation, Maxwell's equations or mass-damper-spring systems as well as RCL circuit equations fit into this framework.

The biggest challenge for the integration of Krylov-subspace based moment matching methods into Electronic Design Automation (EDA) software is the difficulty to choose parameters that determine the approximation error of the reduced order model (ROM): the set of expansion points and the number of moments to be matched per expansion point. Increasing the number of expansion points or the number of moments may or may not reduce this error, but in any case it

will increase the state space dimension of the ROM. Furthermore, unless iterative methods are used, each expansion point involves a computationally expensive matrix factorization so that the number of expansion points has a dominating influence to the computation time needed for generating the ROM. The optimal choice of these parameters requires a priori knowledge of system characteristics that will only be made available with high computational effort, comparable to that spent on simulating the full system.

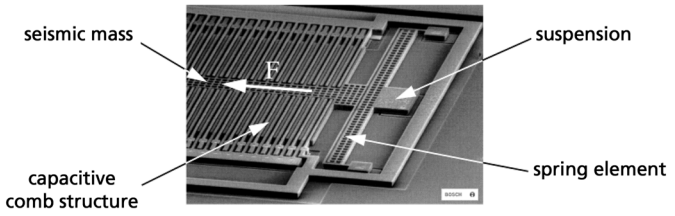

Figure 2: SEM picture of acceleration sensor (courtesy of R. Bosch GmbH)

Using this method for the acceleration sensor in Figure 2, a behavioral model was generated to enable an entire system simulation together with driving and signal processing algorithms in VHDL-AMS system simulators as well as MATLAB/Simulink. The behavioral model should have a minimal state space dimension that allows fast transient simulations, but it has to reproduce the transfer behavior of the original structure as well as possible. The descriptor system for AMPXT has been exported from an ANSYS® finite element model.

Figure 3 compares the parameter dependent frequency response of an accelerometer model with 27,225 degrees of freedom with a parameter dependent reduced order model having only 24 internal states. The considered parameter is the thickness of the spring element of the sensor, which varies between 2.7…3.3µm.

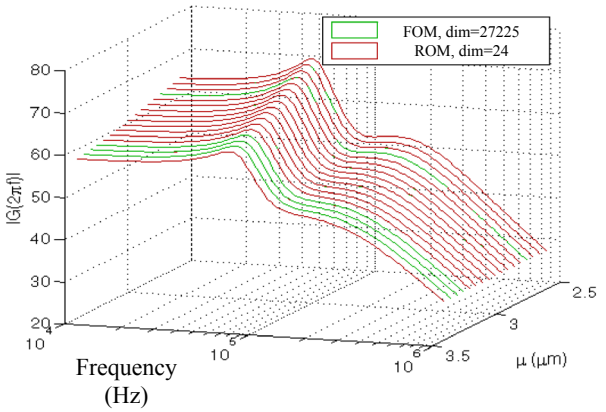

Figure 3: Transfer function of acceleration sensor as function of spring element thickness

AMPXT can be considered as an efficient push button solution for reducing the simulation time for MEMS based systems, as the algorithm manages to automatically generate very accurate ROMs while the user has to provide only the frequency range of interest. Expansion points and model dimension are selected automatically based on an error indicator that measures the degree of convergence of the Krylov iterations. Finally, ROMs can be exported to behavioral modeling languages like VHDL-AMS, Verilog-AMS, SystemC AMS etc. in order to make automated MOR based behavior modeling becomes available for system simulation of entire MEMS devices.

Furthermore, AMPXT can be used for rapid computation of transfer function sensitivities with respect to design parameters. This results in a great reduction of computation time especially in the design optimization process because MOR-based computation of sensitivities only requires a very small additional effort compared to exclusively calculating the transmission behavior.

V. Simulation Environment

Nowadays there is an ongoing trend to “digital assisted analog". In several applications more and more mechanics and analog electronics will be complemented by digital controllers and software algorithms. An example is the use of low cost transducers in combination with appropriate signal processing to compensate non-ideal behavior of the sensor element.

The consequence of those trends for the design process is that it will be ever more difficult or even impossible to consider the analog components independent of the digital parts and to design the integrated circuits without detailed knowledge of the systems' environment. This is why executable overall system level models become essential for the design of an increasing amount of system solutions.

Basically there are different possibilities to handle different models of computation in system level simulation. For the comprehensive consideration of non-electrical components, analog and digital electronics as well as software, we decided to extend SystemC. Therefore, we contributed the essential technology to the OSCI (now Accellera Systems Initiative) SystemC-AMS 1.0 standard [14].

SystemC is a C++ based hardware description language with the focus on system architectural level design of large digital hard- and software systems. It is hosted and standardized (IEEE-1666) by the Open SystemC Initiative (OSCI), a non-profit organization embracing numerous semiconductor companies and EDA vendors. OSCI provides also a so called proof-of-concept implementation on an open source basis. Numerous EDA vendors support SystemC modelling and simulation - mostly based on or derived from the OSCI implementation.

Due to the C++ nature, SystemC is very flexible and powerful. SystemC in particular supports methodologies that enable the interaction of hard- and software as well as modeling at high levels of abstraction to facilitate real system level simulations while achieving the required simulation performance.

Following the SystemC philosophy the SystemC AMS extensions focus on abstract modeling to permit overall system level simulations of “real-time" application scenarios. This requires a simulation performance which is orders of magnitude higher than achieved by models described with

"classical" hardware description languages like VHDL-AMS and Verrilog-AMS. SystemC AMS introduces several abstract models of computation which promise a very fast simulation [13]. For system level modeling the restrictions usually implied by the abstract models of computation (MoC) are not a limit. Moreover, the modeling of system level behavior becomes much easier in many cases. Thus the first version of the SystemC AMS standard includes means for timed dataflow, linear signal flow and linear network modeling. These MoCs provide enough facilities for abstract descriptions of a wide range of applications, especially for communication systems.

A. Brief SystemC AMS 1.0 Language Overview

The SystemC AMS extensions are fully compatible with the SystemC language standard as shown in Figure 4. Thus, it does not change the basic SystemC and introduces no restrictions in the usage of the available SystemC language. The introduced new models of computation cannot directly be mapped to the generic MoC of SystemC as they are not based on communication of processes. They represent an equation system instead. Thus, each AMS primitive represents a contribution to an overall equation system e.g. in the form of equation (3), which has to be set up during elaboration and solved while simulating. The infrastructure for the SystemC AMS extension must thus support these mechanisms.

The AMS language standard defines the execution semantics of the timed dataflow (TDF), linear signal flow (LSF), and electrical linear network (ELN) models of computation and gives an insight on the underlying enabling technology such as the linear solver, scheduler, and synchronization layer. The language has been designed to be extensible. However in the current 1.0 standard, the interfaces to and class definitions of this enabling technology is implementation-defined. The AMS designer (end-user) can take advantage of dedicated classes and interfaces to create TDF, LSF or ELN models by using the predefined modules, ports, terminals, signals, and nodes.

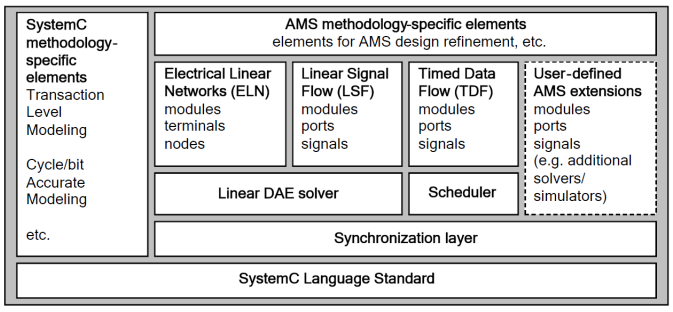

Figure 4: SystemC AMS Language architecture

Besides the time domain simulation, the SystemC AMS standard defines means of small signal modeling for frequency domain and frequency noise domain analysis.

B. Linear signal flow (LSF) and electrical linear networks (ELN)

Both models of computation consist of pre-defined elements. The LSF MoC thus defines non-conservative (connected by directed signals) modules like adder, gain, derivation and integration. The ELN MoC consists of conservative electrical elements like resistors, capacitors and inductors.

Besides these elements belonging to one model of computation several elements are available, which can be connected to other domains. E.g. for LSF, these are mux, demux or sources and for ELN these are switches, voltage and current sources. ELN and LSF models are composed in hierarchical modules in the standard SystemC way.

C. Extension of SystemC AMS to MEMS Simulation

As mentioned before, the SystemC AMS language architecture has to be designed in a way to be extensible. The current SystemC AMS standard was heavily driven by communication applications. However, in the meantime especially automotive applications have a strongly increasing demand on system level investigations and in particular to understand the interaction of the heterogeneous analog parts and the software algorithm.

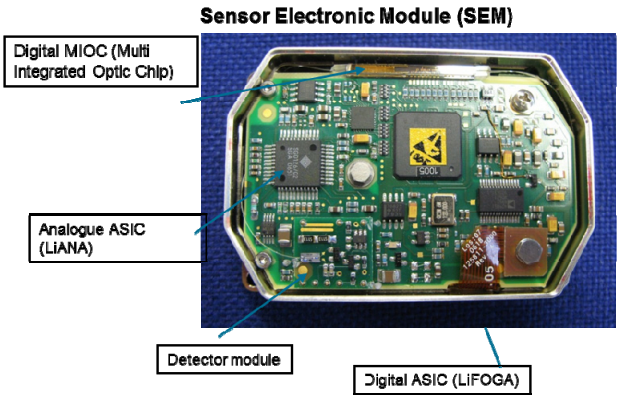

Figure 5: Fiber Optical Gyrosensor system
(Source: Northrup Grumman Litef GmbH)

Compared to communication applications, they have strongly non-linear behavior at the front- and backend – e.g. pwm (pulse width modulation) driver stages or non-linear sensor characteristics. One example may be an airbag system where the squib driver is part of a non-linear control loop with a squib equivalent circuit.

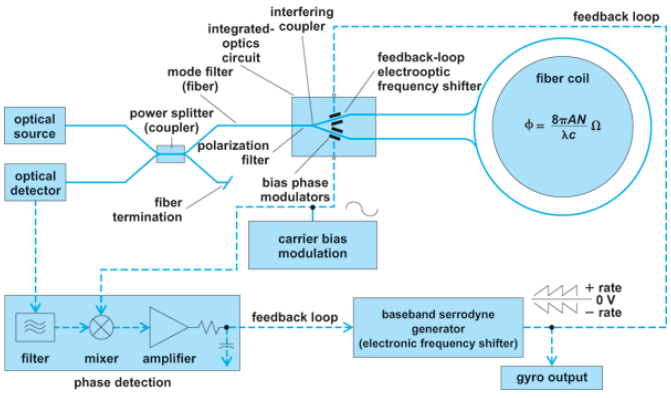

Figure 6: Principle of a fiber optical gyro sensor system [15]

Another example, where SystemC AMS was successfully applied for modeling, is the optical gyro sensor system shown in Figure 5. As shown in the block diagram in Figure 6 to electronics also important components of the optical subsystem have to be modeled to represent the specific behavior of the sensor system (see Figure 7).

Different imperfections like non-linearities in the detector and amplifier can create a bias. Due to the complex interaction with the optical part, it may result in effects like the so called "Bunny Ears" [16] – an important hindrance to increase the yield. A system level simulation will permit to understand those effects due to 100% reproducibility of simulation runs and not limited introspection and debug possibilities. Thus, they will enable the development of compensation algorithms.

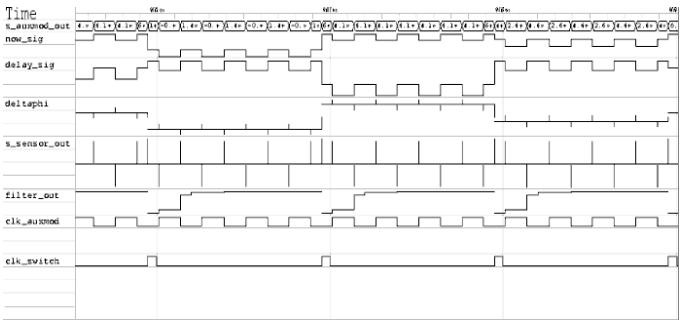

Figure 7: Simulation of the Mixed-Signal control loops

Anyway, those parts of the systems in the example that need highly detailed model descriptions are often rather small. There is usually just a small number of devices at the front- or back-end of the circuit. With languages like VHDL-AMS or Verilog-AMS, those parts can be easily modeled. However, those languages do not support modeling facilities at higher level of abstraction like dataflow or transaction level modeling (TLM). As discussed before, these facilities are required to achieve the necessary simulation performance. First investigations have thus been started to introduce modeling capabilities similar to those of languages like Verilog-AMS and VHDL-AMS. In a system level context, the non-linear parts are usually very small and independent from each other. They will not dominate the simulation performance.

Especially in the automotive context the world becomes heterogeneous. We have not only to deal with electrical signals – furthermore we have to deal with other physical domains like mechanics, magnetic and fluidic. Hence, concepts have to be developed to deal with different physical domains and dimensions.

VI. CONCLUSIONS

In the paper a systematic approach for system level simulation of heterogeneous systems was discussed. The essential elements of this procedure are partitioning, the definition of appropriate interfaces and the derivation of behavioral models. These models consist of a set of differential-algebraic equations, which have a common mathematical structure. Hence, a combination of models within one simulation environment becomes feasible. On the other hand, models can be very large or complex according to the simulation method of the sub-system. We therefore use model order reduction methods and moreover enhanced their performance. This is crucial as the simulation time for the whole system decreases significantly. For the demonstration of system level simulation we used the SystemC platform. Originally it was extended to SystemC AMS to incorporate analog devices. But the language was designed to be extensible and we finally used it as simulation environment for an optical gyro sensor. This proofs the ability of the described method to combine models of different physical domains into one model at system level. Especially in the MEMS application area this hierarchical modeling approach becomes increasingly important as electronic and mechanical behavior in many cases cannot be treated separately anymore in the design process.

REFERENCES

[1] Wachutka, G.: Tailored modeling: a way to the 'virtual microtransducer fab' ? Sensor and Actuators A 46-47 (1995), pp. 603-612.

[2] Senturia, S. D.; Aluru, N.; White, J.: Simulating the behavior of MEMS devices: computational methods and needs. IEEE Computational Science & Engineering, January 1997, pp. 30-54.

[3] Neul, R. et al.: A modeling approach to include mechanical microsystem components into system simulation. Proc. Design, Automation & Test Conf. (DATE'98), Paris, 1998, pp. 510-517.

[4] Fedder, G. K.; Jing, Q. A.: A hierarchical circuit-level design methodology for micromechanical systems. IEEE Trans. CAS-II 46(1999)10, 1999, pp. 1309-1315.

[5] Lorenz, G.: Netzwerksimulation mikromechanischer Systeme. Dissertation Universität Bremen, Shaker Verlag, Aachen, 1999.

[6] Mukherjee, T.: CAD for MEMS design. Proc. DTIP2000 - Symposium on Design, Test, Integration, and Packaging of MEMS/ MOEMS, Paris, 2000, pp. 3-14.

[7] Clauß, C.; Haase, J.; Kurth, G.; Schwarz, P.: Extended Admittance Description of Nonli¬near n-Poles. Archiv für Elektronik u.Übertragungstechnik 49(1995)2, 1995, pp. 91-97.

[8] Schneider, P.; Huck, E.; Schwarz, P.: A Modeling Approach for Mechatronic Systems - Modeling and Simulation of an Elevator System. XI. International Symposium in Theoretical Electrical Engineering, Linz, 19.-22. August 2001.

[9] Reibiger, A.: Über das Klemmenverhalten von Netzwerken, Wissenschaftliche Zeitung TU Dresden, 35, 1986, pp. 165–173.

[10] Schneider, P.: Modellierungsmethodik für heterogene Systeme der Mikrosystemtechnik und Mechatronik. Ph.D. (Dr.-Ing.) thesis Dresden University of Technology, 2010, TUDpress, Dresden.

[11] Schwarz, P.; Schneider, P.: Model Library and Tool Support for MEMS Simulation. Proc. Int. Symposium on Microelectronic and MEMS Technology, Edinburgh, Mai 2001, SPIE Proceedings Series Volume 4407, SPIE, 2001.

[12] Köhler, A.; Reitz, S.; Schneider, P.: Sensitivity analysis and adaptive multi-point multi-moment model order reduction in MEMS design. Analog Integrated Circuits and Signal Processing, Springer Netherlands, DOI 10.1007/s10470-011-9825-0 ISSN 0925-1030

[13] Einwich, K.: Virtual prototyping for smart systems for electric, safe and networked mobility. Proc. 15th International Forum on Advanced Microsystems for Automotive Applications (AMAA 2011) "Smart Systems for Electric, Safe and Networked Mobility, 29.-30.06.2011 Berlin, Springer, 2011, pp. 305-314

[14] "Standard SystemC AMS extensions Language Reference Manual", Open SystemC Initiative, March 8 2010

[15] Barbour, N.: Gyroscope. AccessScience, ©McGraw-Hill Companies, 2008, http://www.accessscience.com /content/Gyroscope/304100

[16] Handrich, G.: Fiber Optic Gyro Systems and MEMS Accelerometer. Advances in Navigation Sensors and Integration Technology, NATO Research and Technology Organisation, February 2004