Describing Space-Continuous Models of Microelectromechanical Devices for Behavioural Simulation

Željko Mrčarica
Institute for Precision Engineering
Technical University Vienna
Floragasse 7, 1040 Austria
zeljko@ifwtrisc.ifwt.tuwien.ac.at

Dejan Glozić
IBM Software Solutions, Toronto Lab.
1150 Englinton St. East, North York
Ontario M3C 1H7, Canada
dglozic@vnet.ibm.com

Vančo B. Litovski
Faculty of Electronic Engineering
University of Niš
Beogradska 14, 18000 Niš, Yugoslavia
vanco@earth.elfak.ni.ac.yu

Helmut Detter
Institute for Precision Engineering
Technical University Vienna
Floragasse 7, 1040 Austria
pdetter@ifwtcad.ifwt.tuwien.ac.at

Abstract

Modern behavioural simulators and their hardware description languages enable description of time-continuous and time-discrete models. In this work, a modelling technique is developed for description of space-continuous models, where partial differential equations are used. A hierarchical library of partial differential equations and boundary conditions for microelectromechanical device modelling is created. Mechanically complex devices have been modelled for system-level simulation using this technique.

1. Introduction

Behavioural simulators and appropriate analogue hardware-description languages like SABER (language MAST) [1] and ELDO (language HDL-A) [2] allow description of time-continuous models. These models are described using algebraic and ordinary differential equations (ODE), where only time derivatives are used. These so-called lumped models are available for electronic devices (diodes, transistors) and can be defined for many other devices (electromagnet, electromotor). However, it is not always possible to define a lumped model on the basis of physical laws that govern the device behaviour. For instance, many micromechanical devices can be described only by the use of partial-differential equations (PDE), because elastic bending of micromechanical part must be modelled. Such bending is preferred way of producing motion in microdimensions, since it is difficult to produce mechanical joints or rotational elements of very small size.

Derivation of the lumped, i.e. space-discrete models is possible for some micromechanical devices in special cases, where PDEs can be solved analytically. For other devices, black-box modelling techniques could be used for derivation of the lumped models from measured characteristics. However, black-box models do not satisfy condition of transparency [3], i.e. parameters of such models do not represent physical quantities that can be estimated from measurement or theoretical calculation. Besides, these models cannot be used for extrapolation outside the area that is used for model derivation, and modelling of dynamic behaviour is very difficult [4].

For that reason, description of physically based and accurate models of micromechanical devices demands usage of PDEs, where not only time-derivatives, but also derivatives with respect to spatial coordinates are used. Nevertheless, such modelling is not directly supported by the modern simulators and appropriate hardware description languages (HDL). Introduction of new language constructions into HDL is not a big problem, provided that the source-code is available. However, the simulation engine of a respective simulator creates some limits in the modelling capabilities. These simulation engines are suited for first-order ordinary differential equations, since they are based on the simulation techniques developed for circuit simulation in the well-

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00

known simulator SPICE. For that reason, partial differential equations must be reduced to the ODE form. That reduction must be done in the language itself, so that the simulation engine can be supplied with ODEs.

For description of the space-continuous models of micromechanical devices and simulation of microelectromechanical systems, we have used the behavioural simulator Alecsis that we had developed. Alecsis is similar to already mentioned simulators [1], [2], having an advantage that its HDL AleC++ is object-oriented. Different analogue systems have been simulated until now [5]. For discretization of the partial differential equations, finite difference method (FDM) is used. The discretization is performed in AleC++.

In the next section, the features that a HDL must satisfy to allow description of space-continuous models are defined. In the third section, organization of hierarchical libraries of PDEs and boundary conditions used in micromechanics is explained. Usage of such libraries makes description of space-continuous models user-friendly. Viability of this technique is proven by simulating several micromechanical devices and systems. The last section summarizes these results and outlines future plans for modelling of microelectromechanical systems using HDL.

2. Modelling with partial differential equations

As is mentioned before, elastically bent micro-machined structures are described using PDEs. Although only terminal behaviour of such devices (usually sensors) is of interest for system-level simulation, derivation of physically based lumped model is not always possible. An automatic translator of partial differential equations into the form readable by circuit simulator SPICE has already been achieved by program MEXEL [6]. Nevertheless, this program is appropriate for equations with spatial derivatives with respect to only one spatial coordinate, which is enough only in special cases. To achieve description of space-continuous model in a more flexible way, we have used our behavioural simulator Alecsis and its hardware description language AleC++.

A structure that is very often used in micromechanics is a membrane (diaphragm). Equation that describes behaviour of the micromachined membrane is [7]:

$$D\frac{\partial^4 w}{\partial x^4} + 2D\frac{\partial^4 w}{\partial x^2 \partial y^2} + D\frac{\partial^4 w}{\partial y^4} = p - k\frac{dw}{dt} - \rho h\frac{d^2 w}{dt^2} \quad (1)$$

where w is the displacement of the membrane, x and y are the spatial coordinates, D is the bending rigidity, p is the pressure, k is the damping coefficient, ρ is the material density, and h is the membrane thickness.

Boundary conditions for the eqn. (1) are determined as the model of physical connections of the membrane to other parts of the system. Membrane can be connected to other parts in different ways. If the length of the membrane is L and the edge $x=L$ is built into some rigid rim, the boundary conditions are:

$$(w)_{x=L} = 0, \qquad \left(\frac{\partial w}{\partial x}\right)_{x=L} = 0 \quad (2)$$

Boundary conditions for membrane that are also often used in micromechanics are those for the free edge. If W is the width of the membrane, and edge $y=W$ is free, boundary conditions are [7]:

$$\left(\frac{\partial^3 w}{\partial y^3} + (2-\nu)\frac{\partial^3 w}{\partial y \partial x^2}\right)_{y=W} = 0, \quad \left(\frac{\partial^2 w}{\partial y^2} + \nu\frac{\partial^2 w}{\partial x^2}\right)_{y=W} = 0 \quad (3)$$

where ν is the Poisson's ratio.

For simulation, partial differential eqn. (1) has to be discretized, i.e. partial derivatives have to be replaced by finite differences. The discretization is performed by defining some spatial mesh, and separate equation is defined for every mesh node. The same must be performed for each PDE describing boundary condition. Therefore, one PDE is replaced by an equivalent system of ordinary differential equations.

A HDL must satisfy some conditions to allow such description, i.e. discretization of PDEs. There are two main demands that are imposed. Firstly, HDL must be capable of declaring an array of nodes (signals) of variable size. In the case of eqn. (1), displacement w is represented as a two-dimensional array of local displacement of mesh nodes. The second important condition is that the language should be capable of creating an array of devices (submodels). In this way, a discretized equation representing eqn. (1) can be defined, and then copied into an array of equations over the mesh nodes. This feature of creating an array of components is known from VHDL, where array of logical gates or subcircuits is often necessary.

Both these demands are satisfied in AleC++. Dynamic allocation of arrays of signals (nodes) is possible. It should be noted that AleC++ can be used both as interpreted and as compiled language. Variable size arrays can be used in both modes; there is no need for model recompilation when size of an array is changed.

In the same way, an array of components (in this case discretized equations) can be created. A discretized PDE can be defined in a construction named *module*, which is the basic structural unit in AleC++. In a module hierarchically above this one, this construction can be declared and then multiplied in a *for loop* that covers all

mesh nodes. A keyword *clone* is used for copying a previously declared and defined component. PDEs for boundary conditions can be defined in the same way, and *cloned* in *for loops* for mesh boundary nodes.

In this way, the space-continuous model is described as the system of equations, which is a part of the complete system of equations for the whole circuit (system). Therefore, the device and the system simulation are mixed, since quantities that represent internal field distributions appear in the same system of equations as quantities on the device terminals. Although such combination of description levels might be considered inefficient, it is sometimes necessary for strongly coupled systems. When device model and system description are solved by two coupled simulators, the convergence problems can appear due to the characteristics of Gauss-Seidel method for solving the system of equations, i.e. relaxation method of simulation [8]. It should be noted that the simulation of the mechanical device behaviour together with surrounding electronics is more important in the case when coupling is strong, otherwise separate simulations can give satisfactory results. In Alecsis, the system of equations is solved as a whole, and the Newton-Raphson method is used, which has better convergence properties for strongly coupled systems than the relaxation method.

3. Hierarchical libraries for description of space-continuous models

The features of the HDL AleC++ allow the description of partial differential equations. Nevertheless, when partial derivatives are replaced by finite differences in equation like eqn. (1), the complexity of the expression grows. Therefore, it can be very difficult and error-prone to describe such discretized expression directly. In order to simplify the model description, a hierarchical library of submodels is prepared.

Firstly, a set of modules for discretization of partial derivatives can be defined. Such modules can be predefined for each form of partial derivatives that is used in eqn. (1). Of course, other forms of partial derivatives can be used for some models of micromechanical devices, too. Nevertheless, for elastic deformation PDEs are limited to the fourth order. Therefore, the number of possible variations of partial derivatives is limited, and all cases can be prepared in the library. Then, a PDE can be described by invoking submodules for partial derivatives, where they contribute to the same equation. That enables us to describe eqn. (1) nearly as it is written.

There are other libraries that can be prepared for modelling of micromechanical elements. Since the membrane is often used in micromechanics, it can be very useful to have the membrane equation also prepared in the library. Besides, different boundary conditions for the membrane (eqns. (2) and (3), etc.) are often used, and can be prepared as the library, too. These libraries are hierarchically organized. A hierarchical structure of the libraries of predefined submodels for use in micromechanics is given in Fig. 1. The membrane equation and the boundary conditions are based on the library of partial derivatives. A pressure sensor, created as the membrane with all edges built-in, can be created using previously defined membrane equation and appropriate boundary conditions.

AleC++ supports such hierarchy through the usage of *classes*. AleC++ is an object-oriented language, designed

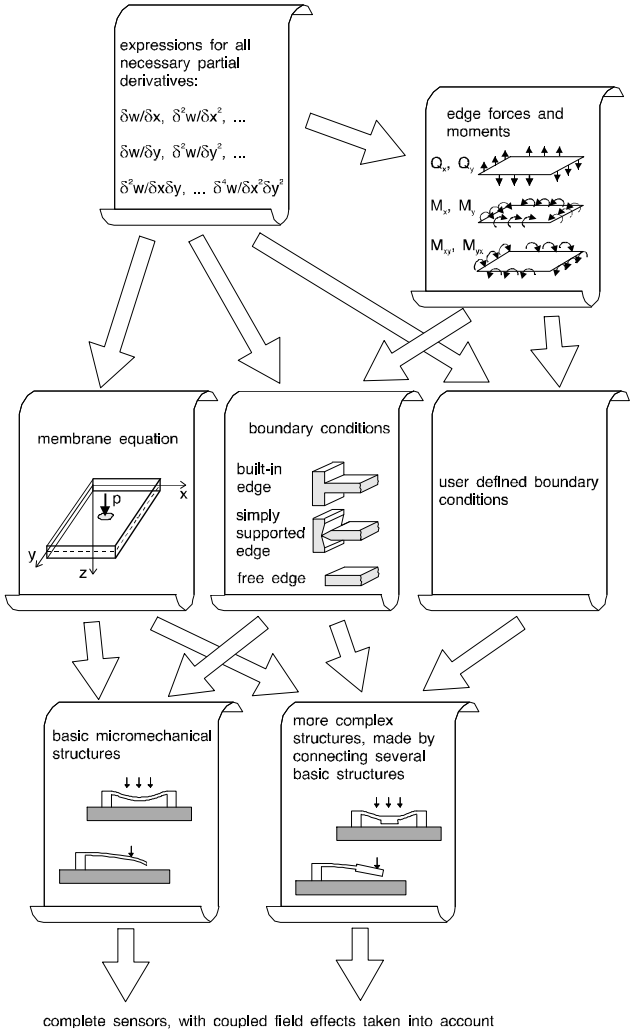

Figure 1. Hierarchical libraries of micro-mechanical submodels.

as a superset of C++. Basic construction for modelling is a *module*. The *class* is the structure that contains the module, model parameters (defining in that way a model card) and other useful C++-like functions. Usage of *derived classes* gives a straightforward way to establish hierarchy, as is depicted in Fig. 1. The hierarchy can be also defined in the HDLs that are not object-oriented, by invoking already defined submodels during definition of higher-level models. Nevertheless, usage of an object-oriented language makes this hierarchical organization much more understandable and secure, since inheritance of model properties is better controlled, as well as access rights to model parameters.

When the libraries given in Fig. 1. are used, the description of micromechanical structures is much simpler. By simple combination of predefined model equations and boundary conditions, different structures can be defined. For instance, the membrane with two edges built-in and two edges free can be modelled and simulated simply by invoking appropriate equations (modules) from the library. Such model represents a new, user-defined *derived class*, that can be also stored in the library.

The basic boundary conditions for the membrane, depicted in Fig. 1., do not cover all possibilities. The boundary conditions can be defined as the equilibrium of boundary forces and moments between two neighbouring structures. Since such boundary conditions cannot be prepared for the general case, the designer of the micromechanical device model is responsible for defining them. For that reason, expressions for shearing forces, bending moments and twisting moments at the membrane boundary [7] are also given in the library (Fig. 1). These expressions are also PDEs. They can be used for definition of complex boundary conditions, for example for connection of the membrane with the boss (thick and rigid part). Structures with bosses are often used for micromachined sensors. The designer can define the new class, containing boundary conditions that he needs, and store it in the library.

There is also a special boundary condition of symmetry, already prepared in Alecsis library. When both the micromechanical structure and the load are symmetrical, it can be very useful to model and simulate only the part of the structure, since the respective system of equation is much smaller, and therefore the simulation time is much shorter. (Due to the sparse matrix solver, simulation time does not increase drastically with the increase of the matrix size, but the benefit is still considerable.)

When the libraries given in Fig. 1. are used, modelling of micromechanical structures is simplified. Nevertheless, the discretization scheme (pattern of neighbouring mesh nodes used for discretization) would vary for different PDEs. The designer is still responsible to know these schemes and to build the consistent system of equations, where one equation corresponds to one mesh node.

4. Modelling and simulation examples

The simulation results for the membrane with different boundary conditions are given in Fig. 2. The difference in the model code between cases a) and b) is only in the boundary conditions that have been invoked in the appropriate *for loop*. In both cases membrane width and length are $L=W=1$mm, thickness is $h=5\mu$m, Young's modulus = 169GPa, Poisson's ratio = 0.3. The uniform load of 1kPa is applied.

In Fig. 3, simulation results for a more complex micromechanical structure are given. The structure is a force sensor with piezoresistive transducers [9]. The two meshed parts are micromachined membranes. Each of the membranes has one side built into the rim and two sides free, while the fourth side is built into the boss. The boss is the central part of the structure, where the force is acting. The boundary conditions for the first three mentioned edges are available in the library. For the

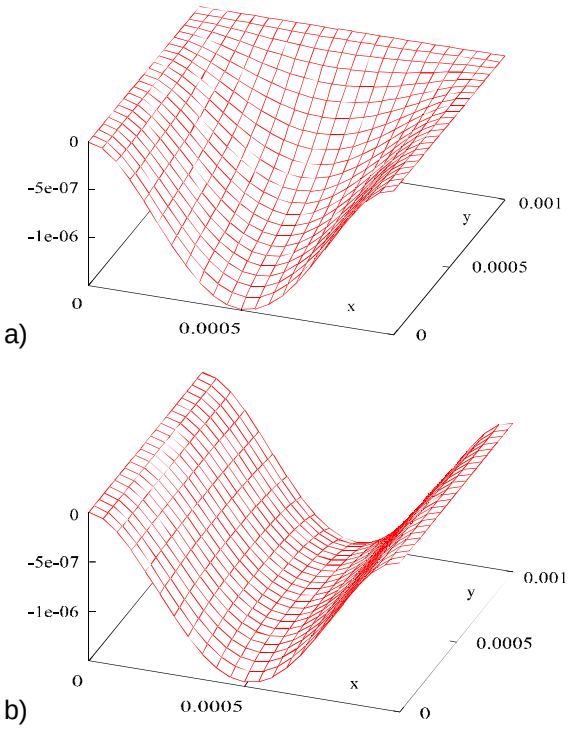

Figure 2. a) Structure with three edges built-in and one edge free. b) Bridge structure, with two edges built-in, and two edges free.

fourth edge, boundary conditions are defined from the equilibrium of forces and moments on the membrane edge (which are also available in the library) and for the geometrical conditions for the membrane built into the boss. In Fig. 3., the deflected parts are membranes 354μm wide, 133 μm long, 5μm thick. Rigid part (boss) is 354μm wide and 870.7μm long. Force F=2mN is acting at the boss centre.

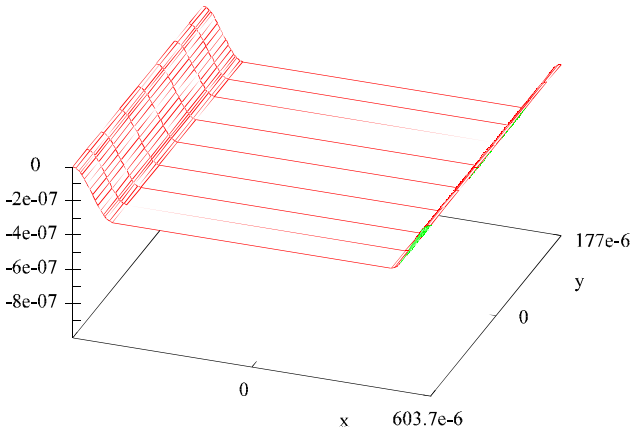

Figure 3. Displacement of the force sensor.

Fig. 2. and Fig. 3. represent simulation results for micromechanical devices. However, the main purpose of our modelling method was to enable simulation of the microelectromechanical *systems* (MEMS). When a sensor is modelled using the described technique, it is enclosed in a module that is later invoked with its terminal connections. Therefore, although such model is space-continuous, it is invoked in the same way as any other model in the system-level description (transistor, resistor, capacitor, etc.). If the model of micromechanical sensor or actuator is already prepared in the library, the system-level designer does not need to know its internal structure. The model is invoked by its name, connection terminals (which are not necessarily of electrical nature) and the model card (similar to the SPICE model card). More than one instance of the given model can be declared and used.

In Fig. 4., a capacitive pressure sensor is shown, with switched-capacitor (SC) circuit for converting the capacitance value into the output voltage. ϕ_1 and ϕ_2 are the switching phases. The sensor is the membrane of rectangular shape with all four edges built into the rim. Capacitance between the membrane and the bottom plane is changed when the pressure is applied. There are four components of the pressure - the applied (measured) pressure, the pressure caused by a temperature coefficient mismatch between different materials used in the sensor production, the electrostatic pressure, and the pressure of the gasses trapped in the sensor chamber. All these

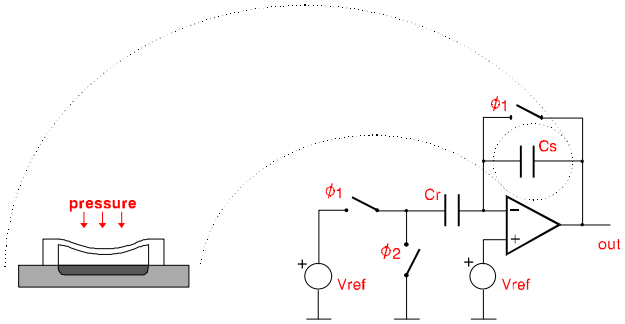

Figure 4. Capacitive pressure sensor with SC read-out electronics.

components can be analytically expressed and added to the eqn. (1).

In Fig. 5., simulation results for the system given in Fig. 4. are given. The traced quantities are the measured pressure, the displacement of the membrane centre, the capacitance of the sensor and the circuit output voltage. The effects of the electrostatic pressure change are modelled and simulated. The voltage across the sensor plates is variable, due to the switched-capacitor circuitry. The abrupt change of the electrostatic pressure causes mechanical oscillations. The oscillations are modelled by the time derivatives in the eqn. (1).

Models like this capacitive pressure sensor cannot be formulated in lumped form in general case, even by black-box modelling techniques. The influences of the applied pressure, the electrostatic pressure and other effects are distributed over the membrane surface. Besides, extraction of the lumped model from the measured characteristics is difficult because of inertia and dumping effects, i.e. time dependencies. The space-

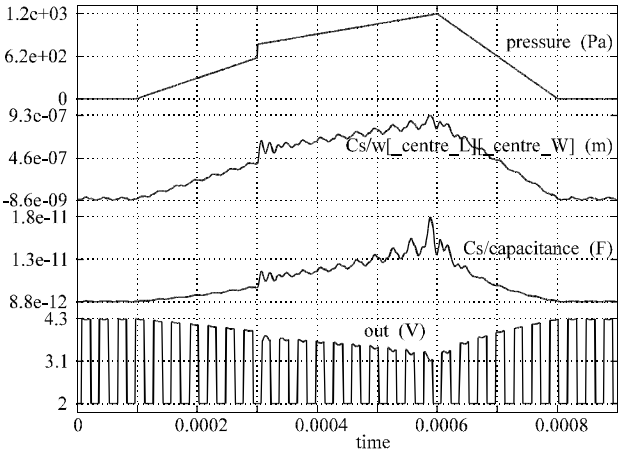

Figure 5. Simulation results for the pressure sensing system.

continuous model that is used here is physically based, with all important effects taken into account. Simulation time for the example given in Fig. 5. was 10.4 CPU minutes on a modest UNIX workstation.

This example exhibits bidirectional coupling of mechanics and electronics, since influence of the electrostatic forces is important (due to the miniaturized size of the sensor). If two specialized simulators (one for device and other for system simulation) are coupled, convergence problems can arise. In our simulation, one behavioural simulator is used, therefore one system of equation is built for the whole system. The Newton-Raphson method, which has very good convergence properties, is used for solving.

5. Conclusion

In this work, behavioural mixed-signal simulator Alecsis was used for the simulation of systems that contain space-continuous models of devices. Other simulators can be used for this purpose in the same manner, providing that their hardware description languages have two properties: allocation of the arrays of nodes (signals) of variable size; use of the arrays of subcomponents of variable size.

Although space-continuous models are usually not of interest for the system-level simulation, they are sometimes necessary. It is not always possible to define a physically based lumped model that is accurate enough for simulation.

Since in this way the internal fields in the device are also modelled and simulated, a behavioural simulator can be used for device-level simulation, too. Although finite-element simulators are usually used for device-level simulation of sensors and actuators, analytical approach of modelling can sometimes has its advantages. Limits of such models are the same as, for instance, for program SENSIM [10]. Device models described in this way can be invoked as components for system-level simulation.

Simulation times for sensors involving read-out circuitry were less than 30 min for the most complex examples. These times are acceptable when compared to times for complex finite-element device simulations.

The user-friendliness of the PDE description is facilitated by the use of libraries of submodels. This library is created for micromechanical devices. Nevertheless, usage of PDEs for modelling can be important for other problems, too. For instance, models of transmission lines are space-continuous. Also, such modelling approach can be used for simulation of some electronic devices. On the basis of libraries in Fig. 1., equivalent libraries for different purposes can be developed.

Clearly, the approach that we use allows only limited complexity of the models of micromechanical devices. Usage of finite elements instead of finite differences would enable more flexibility in the modelling, since the meshing (discretization) is more general. Usage of finite elements would be our research topic in the future. Our work will be also oriented to building of libraries of different micromechanical devices, as well as to further development of HDL AleC++ for increasing the user friendliness of the space-continuous model description.

Acknowledgement

This work was supported by project P10068 - "Development, construction and fabrication of micromechanical parts" from the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung*.

References

[1] I. E. Getreu. Behavioral modeling of analog blocks using the SABER simulator. *32nd Midwest Symposium on CAD*, Illinois, USA, 977-980, August 1989.

[2] D. Pabst. HDL-A VHDL-based analog and mixed signal model description language. *Tutorial T1 of Simulation Congress EUROSIM '95*, Vienna, Austria, September 1995.

[3] G. K. M. Wachutka. Problem-oriented modeling of coupled physical effects in microtransducers and electronic devices. *Proc. of IEEE Int. Conf. on Microelectronics MIEL '95*, Niš, Yugoslavia, 539-547, September 1995.

[4] K. Hofmann, J. M. Karam, M. Schulze, M. Theisen, B. Curtois, M. Glesner. Automatische Übersetzung von FEM-Modellen in eine Analoge Hardwarebeschreibungssprache. *Proc. of Mikro System Technik, Mikromechanik & Mikrotechnik*, Chemnitz, Germany, 86-91, October 1995.

[5] Ž. Mrčarica, T. Ilić, D. Glozić, V. Litovski, H. Detter. Mechatronic simulation using Alecsis. Anatomy of the simulator. *Proc. of Simulation Congress EUROSIM'95*, Vienna, Austria, 651-656, September 1995.

[6] G. Pelz, J. Bielefeld, F.-J. Zappe, G. Zimmer. MEXEL: Simulation of microsystems in a circuit simulator using automatic electromechanical modeling. *Proc. of Micro System Technologies '94*, Berlin, Germany, 651-657, October 1994.

[7] S. Timoschenko, S. Woinowsky-Krieger. *Theory of plates and shells*. McGraw-Hill Inc., 1959.

[8] S. Schulte, A. Maurer, H. Bungartz. Modular solution approach for simulation of coupled physical phenomena. *Proc. of Conf. on Simulation and Design of Microsystems and Microstructures MICROSIM '95*, Southampton, UK, 201-209, September 1995.

[9] N. Delic, H. Detter, G. Popovic, W. Brenner. Microgrippers and measuring forces applied to microparts. *Proc. of ECPD Int. Conf. on Adv. Robotics and Intelligent Automation*, Athens, Greece, 481-486, September 1995.

[10] K.-W. Lee, K. D. Wise. SENSIM: A simulation program for solid-state pressure sensors. *IEEE Transactions on Electron Devices*, 29(1):34-41, January 1982.