# Part I
# Physical and Mathematical Fundamentals

# 1
# Introduction: Issues in Microsystems Modeling

*Gary K. Fedder and Tamal Mukherjee*

## 1.1
## The Need for System-Level Models for Microsystems

Multiphysics microsystems are having an increasing practical impact on our lives as the industry creates a wealth of new products based on microelectromechanical systems (MEMS) such as accelerometers, gyroscopes, resonant timers, microphones, radio-frequency (RF) switches, tunable RF passives, micro-optical displays, microvalves, and microfluidic total analysis systems. System-level modeling and simulation are essential tools in designing such complex multiphysics microsystems. This book provides an overview of system-level modeling methodologies and tools, drawing from experts in microsystems and tapping into their multiple perspectives. It acts as a resource for future researchers who wish to build on this foundation.

As the field continues to mature, more diverse and integrated microsystems will evolve from exploratory prototypes into tomorrow's product offerings. This high-level commercial activity is motivating the research on methodologies of accurate multiphysics system-level simulations that provide rapid analysis of iterative design and allow the transference and archiving of design knowledge. Simultaneous with these advances in multiphysics aspects, an increasing number of microsystems comprise a multiplicity of devices that are integrated with electronics. Such integrated microsystems require cosimulation of electronics and MEMS, further stimulating the need for system-level models in support of rapid and efficient development.

There is ample motivation to refine and to automate system-level modeling methodologies of multiphysics microsystems [1–3]. One can turn to the semiconductor industry to put into perspective the complexities of device modeling. In advanced complementary metal–oxide semiconductor (CMOS) electronics, sophisticated transistor models for each emerging technology node must be created before commencement of analog and digital circuit design. Foundries can justify employing hundreds of engineers and technicians to create these models in a relatively short time frame. The microsystems field does not have the ability to apply such a large amount of resources to complete multiphysics device models.

To exacerbate the issue, even in a fixed process, the multiplicity of MEMS devices renders it impractical to replicate the enormous effort given to transistor modeling. Instead, microsystems modeling efforts must leverage the CMOS design infrastructure while continuing to advance automated methodologies in order to meet the challenges.

A particularly important recent trend is the growth of the number of foundries offering custom MEMS process services and the emergence of CMOS MEMS process offerings within CMOS foundries. The ''one process, one product'' mantra that was common during the past two decades of MEMS commercial development must become a relic of the past for these latter foundry processes to be successful. A significant roadblock to process reuse is the lack of adoption of common modeling methodologies that enable designers to exploit existing MEMS processes for rapid creation of new products. Part of the solution, and the core mission of this book, is to make available a near comprehensive overview of the state of the art in corresponding system design and modeling tools and methodologies for microsystem developers.

This chapter first provides an introduction to coupled multiphysics phenomena and to multiscale modeling and simulation of microsystems. A concise glossary of system-level model terminology is next presented, followed by a short description of model order reduction (MOR) methods. The concepts of very-large-scale integration (VLSI) hierarchy and views are next presented as a means to handle complexity in the microsystem design process. Modern analog hardware description languages (AHDLs) for system-level model implementation are then introduced, followed by general attributes of AHDL models and capabilities of AHDL simulators. The chapter ends with consideration of multiphysics model libraries for microsystems design and the need for parameter extraction, model verification, and model validation to produce trusted models.

## 1.2 Coupled Multiphysics Microsystems

Microsystems are generally less than a cubic centimeter in size and have one or more critical aspects of operation that are dependent on micron-scale, or even smaller, dimensions. The small scale leads to an extremely tight coupling of multiphysics aspects arising from the processes and devices composing a microsystem. This intimate coupling sets microsystem design apart from most macroscale system design and presents subsequent challenges and opportunities for the modeling community. Microsystems pose complex problems that are at best difficult and time consuming to solve with continuum field analysis. Especially, in the case of time-stepping analyses, many problems are intractable using the currently available continuum field analysis software running on the fastest computers. Layered over these issues is the desire to perform iterative design that requires multiple sequential parametric analyses. Practical realization of rapid time-domain

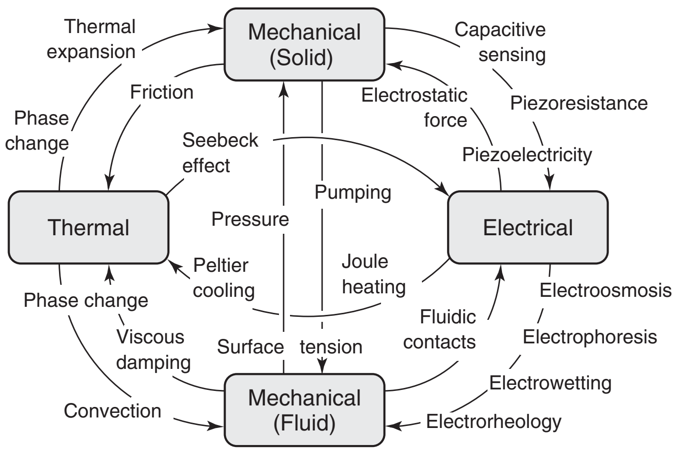


**Figure 1.1** A sampling of coupled multiphysics in microsystems.

analyses in support of the design process for complex coupled-physics systems necessitates the formation of system-level models.

The list of coupled physics in microsystems is truly inexhaustible; a short summary here of some highlight areas helps underscore this observation. Electronic and active materials used in sensors and actuators have inherent coupling between energy domains, exemplified by electromechanical, electrothermal, magnetostrictive, piezoelectric, piezoresistive and shape memory (phase change) effects. A seminal paper by Middlehoek and Hoogerwerf [4] in 1986 categorized the coupling in solid-state sensors into six signal domains – radiant, mechanical, thermal, electrical, magnetic, and chemical – with physical sensing effects classified according to the domains of the input signal, the output signal, and the auxiliary energy source that modulates the output signal. A sampling of coupling between a subset of physical energy domains of major importance in microsystems is illustrated in Figure 1.1. Examples of actuator models are given in Chapters 5, 6, and 11 and modeling of energy harvester systems are overviewed in Chapter 13. Effects of induced stress in packaging represent a very important aspect of overall system simulation, with one modeling approach outlined in Chapter 6.

MEMS inertial sensors and resonant devices have complex interactions between inertial excitations, mechanical stresses in flexures, electrostatic fields with moving walls, thermal interactions with material properties, viscous losses from the surrounding ambient, and intrinsic losses within structural materials [5]. Examples of modeling and simulation of inertial microsystems are given in Chapters 12, 15, and 16.

RF microswitches and tunable capacitors add interactions of impedance matching, wiring and substrate loss at RF frequencies, long-term creep in metallic materials, and tribology of surfaces including dielectric charging phenomena and the physics of electrical contacts [6]. Modeling and cosimulation of RF MEMS with circuits is presented in Chapters 8, 10, and 14.

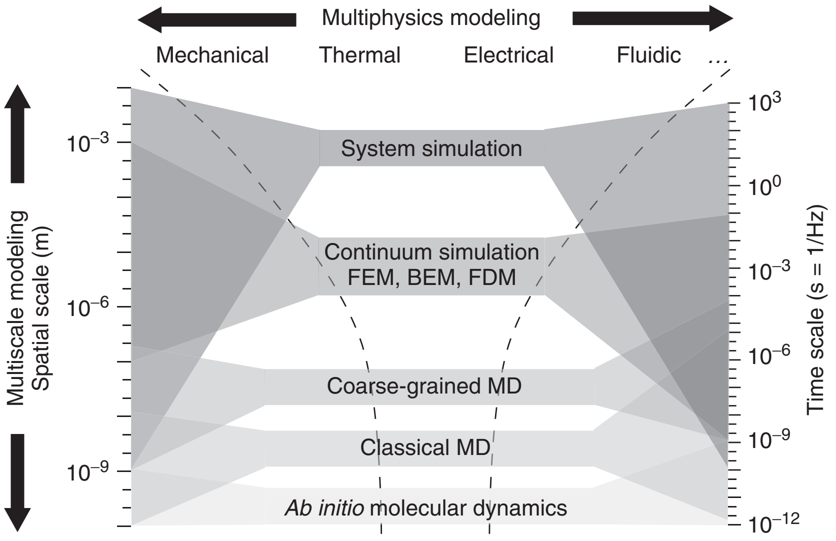


**Figure 1.2** Multiscale modeling and simulation hierarchy.

Optical-based and probe-based microsystems are two emerging areas poised for future commercialization. Optical microsystems may include arrayed micromirror, lens, and waveguide components, [7] as well as, in more complex cases, exploit interactions of optical, thermal, and mechanical forces to create optically coupled microcavities [8] and chip-scale atomic clocks and sensors [9]. In nanoprobe and nanorelay systems, scaling of mechanics down to 50 nm and below brings the need to model atomic-scale forces, such as the van der Waals and the Casimir forces [10].

Microfluidic systems are governed by a complementary set of physics: incompressible flow, diffusion, convection, two-phase flow, electroosmotic and electrophoretic forces, surface tension, electrowetting, and fluid–particle interaction [11, 12]. Chapters 10 and 11 introduce some examples of microfluidic device modeling. An approach to model fluidic damping is covered in Chapter 7. A final extensive category comprises the innumerable chemical, biological, and material interactions exploited in chemical and bioMEMS devices [13]. The modeling procedures demonstrated in this book are applicable to these systems.

## 1.3 Multiscale Modeling and Simulation

A hierarchy of multiscale modeling and simulation is illustrated in Figure 1.2. System-level tools lie at the top of the simulation hierarchy and rely on behavioral models to describe the underlying physics. A system's behavior can be simulated across a very wide range of spatial scale and timescales; however, this occurs at the expense of the granularity of the system representation. Continuum simulation, which includes the finite-element method (FEM), boundary-element method (BEM), and finite-difference method (FDM), handles physics expressed by continuum partial differential equations. Molecular dynamics (MD) simulations represent interactions between atoms in the most fine-grained *ab initio* approaches. Classical MD approaches represent interactions between molecules and coarse-grained MD deals with interactions of larger molecular units, for example, grains in polycrystalline materials.

As briefly discussed in Section 1.2, simulation of a complete system with software that solves all physics at a continuum level (or at one of the MD levels) are generally impractical when run over timescales of interest for most microsystems (e.g., over billions of time steps). It is theoretically conceivable to embed continuum numerical analyses for devices into system simulation (this is sometimes called *mixed-mode simulation*); however, the resulting simulation speed is not acceptable for design. Furthermore, the continuum approach becomes even less practical for complex microsystems with numerous interacting components that require simulation over multiple spatial scales. The flexibility in abstracting behavior over multiple time and spatial scales in system-level simulation allows for the analysis of extremely complex coupled phenomena.

A key step in implementing system-level simulation is the translation of the physical behavior of the constitutive components in a system from the more fine-grained continuum level to more abstract coarse-grained models. Building this relationship between different tools in the simulation hierarchy, where each tool has viable utility at different spatial and time scales, is known as *multiscale modeling*. Information is passed up the hierarchy from fine-grained simulation to fill in physical parameter values, material property functions, or other behavioral relationships in a more coarse-grained model.

An important challenge in system-level modeling is the preservation of accuracy from fine-grained simulation to a degree that is deemed adequate. For the simulator to run in a reasonable time, the system-level model should only include the degrees of freedom (DOFs) necessary to capture the relevant physics. Very handy in this sense are the mathematical methods of MOR, which under certain conditions enable almost automatic transfer from the continuum level simulation up to the behavioral models with minimal loss of accuracy. These methods are described in Chapter 3, with more detail on specific approaches in Chapters 9 and 10, examples of nonlinear MOR in Chapters 11 and 12, and three commercial implementations of MOR in Chapters 18, 20, and 21.

## 1.4
## System-Level Model Terminology

Identifying relevant multiscale and multiphysics phenomena in a device or system and encoding the interactions appropriately are usually a significant challenge that is very application specific. Combinations of techniques are required to build most multiphysics models for microsystem simulation. Subsequently, there is a long list of terminology used to describe system-level models and modeling approaches.

A *component*, in the context of computer-aided design tools, generally describes a functional part of a system that is represented by the combination of its system-level model and its symbol for use in system-level schematics.

A *behavioral model* is described, in whole or in part, mathematically through relationships between the model's terminals and external parameters. At least some part of a behavioral model is defined by differential and algebraic equations, but it can also incorporate interconnected subcomponents. A *primitive behavioral model* (or just *primitive model*) has no interconnected subcomponents; it is a model defined solely by differential and algebraic equations.

A *structured model* is described solely in terms of interconnected subcomponents (i.e., a schematic). At the lowest possible level in a system-level modeling hierarchy, these components must be described by primitive models.

A *circuit model* or *network model* is a structured model with potential and flow variables assigned to each terminal connection and that obeys the conservative Kirchhoffian network laws.

A *signal-flow model* (or *block-diagram model*) is a structured model where only potential variables, and no flow variables, are defined for each terminal connection.

A *physics-based model* incorporates equations derived from the physics of the problem. This is in contrast to most MOR techniques that fit to nonphysical basis functions. One potential benefit of physics-based models is their use for scaling studies and extrapolative studies.

A *compact model* is a well-established name for accurate transistor device models used for the representation and simulation in schematics. Compact models may be formed from physics-based equations, fitted to basis functions, or built from lumped components. A mix of these approaches is used for compact models of advanced-technology-node transistors. The term is often applied to complex microsystem behavioral models and, in general, encompasses both primitive and structured models.

A *reduced-order model* is a behavioral model formed by reducing the order of a high-DOF model, typically from numerical continuum simulation (e.g., using MOR).

A *lumped-element model* specifies a reduced-order modeling approach where spatially distributed physical behavior is ''lumped'' into a finite set of ''elements'' that approximate behavior at discrete points in space. Typically, a lumped-element model is implemented as a structural model with ''elements'' comprising physics-based primitive models. A common microsystem example is a 1-DOF mass-spring-damper model. A common electrical example is an inductor–capacitor network approximation of a transmission line.

The term *macromodel* has its origins from SPICE circuit modeling, where it is used to describe a ''subcircuit'' (i.e., a structured model embedded in SPICE code) comprising some combination of available primitive models such as transistors, idealized dependent sources, inductors, capacitors, and resistors [14]. In SPICE, all primitive models are built into the simulator and thus constrain designers to creating structured models (i.e., macromodels) based on the limited inventory of primitive models. The term is sometimes used to convey a compromise of accuracy somewhere between a circuit made solely of transistors and a more abstract model having no transistors. Common macromodel examples exist for operational amplifiers, comparators, timers, and other high-level analog electronic components. In the MEMS field, the term macromodel is often interchanged with reduced-order modeling; however, this association is not always synonymous with the older SPICE usage and is discouraged.

## 1.5 Automated Model Order Reduction Methods

Continuum field solvers are important tools that are used to verify system-level models that are created manually. A large swath of coupled multiphysics can now be solved for modest-sized device-level problems with commercially available software on desktop computers or on parallel computing clusters. Increasingly, these tools are also being used to generate system-level models automatically (Chapters 18, 20, and 21). Advancements in algorithms to speed up finite-element and boundary-element computations, and their greatly increased ease of use within commercial multiphysics tools has enabled their practical use in automated parameterized model generation.

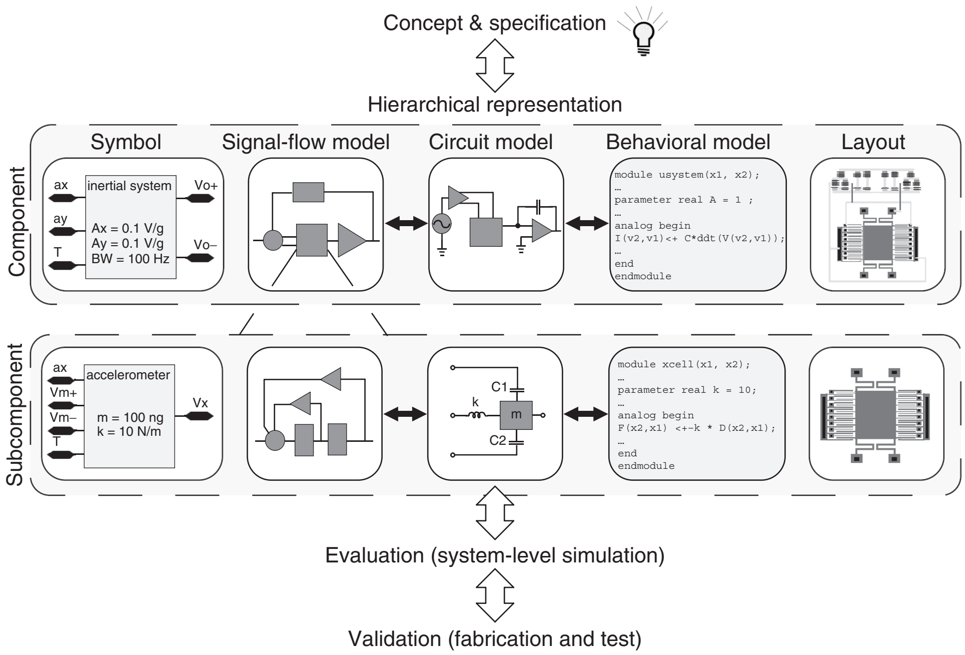
Often, system-level models are created from fine-grained simulation results by optimally adjusting coefficients to a basis function, which may be polynomials or rational functions of polynomials. Dynamic models use the fundamental differential equations governing the system, but with a reduced order. For example, the number of DOFs in a continuum model with 10 000 elements may be reduced to 10 DOFs by extracting the lower 10 eigenmodes of the simulated system. An issue with this approach can be the decision of where to truncate the matrix so that all important modes are included while excluding as many insignificant modes as possible to speed up system simulation. One example of high-order modes that could be significant are self-resonances of comb finger beams in electrostatic sensors and actuators. Modulation frequencies applied to comb transducers could conceivably stimulate resonance, yet these frequencies may be far above the first 10 modes of the overall system. A generalized automated modeling algorithm would have no way of knowing that these modes were essential to consider for proper system design. Also, systems with high nonlinearity or dynamic parametric effects may result in modal modifications that are difficult for automated modeling algorithms to predict without manual intervention. For these reasons, the Krylov-subspace methods, the accelerated grammian methods, and the parametric and nonlinear projection methods are being developed by mathematicians and increasingly used by engineers (Chapters 9–12).

## 1.6
## Handling Complexity: Following the VLSI Paradigm

The VLSI design paradigm uses both hierarchy and views to handle complexity [15]. Hierarchy is implemented by building a system or component comprising smaller subcomponents. The components have their behavior wholly encapsulated by their time-varying terminal relations and the fixed values of external parameters. This behavioral encapsulation allows components to be instantiated anywhere in the system to implement their function instead of being replicated from scratch. Partitioning within a system hierarchy may continue until the subsystems comprise only primitive behavioral models. For example, an inertial microsystem can be partitioned into a MEMS subcomponent and an electronics subcomponent. The MEMS subcomponent can be partitioned further into individual accelerometer and gyroscope components. Each accelerometer can be further partitioned into mass-spring-damper and electrostatic transduction subcomponents. As a possible final layer in the hierarchy, the mass-spring-damper component can be modeled as interconnecting beams and plates. Alternatively, mechanical and electrical subcomponents of the system can be represented by reduced-order models, gained by mathematical MOR methods and inserted into the hierarchical representation. For example, the springs in an accelerometer may be represented by an accurate reduced-order model.

However, if hierarchy was the only advantage of VLSI design then there would be little difference from the many other areas in engineering that exploit system partitioning. VLSI design is truly unique because of the combination of hierarchy with parallel model views. A *view* is a representation of a component, which may take the form of a component symbol, a layout, or any of the various model types; for example, a primitive behavioral model, a signal-flow model, a structured model (i.e., a schematic), or a schematic that includes parasitic elements extracted from layout information. The powerful implication for design is the ability to explore the hierarchy from the perspective of any of these views, as illustrated in Figure 1.3. Different model abstractions (i.e., model views) of particular components may be swapped in for evaluation at any level in the hierarchy, which allows for top-down conceptual thinking and ''what-if'' experimentation. If a subcomponent is specified by a primitive behavioral model then its detailed implementation at a lower level in the hierarchy can be decoupled from the rest of the system. It is not necessary for all views to be filled in order to evaluate performance at each level of the hierarchy. This feature allows refinement of modeling and design to occur in parallel and iteratively at any level in the hierarchy.

Microsystem design benefits directly from the VLSI paradigm; however, the supporting infrastructure for multiphysics systems is in its infancy relative to analog/digital electronic systems. The increased complexity of interacting energy domains provides a huge incentive to design microsystems with simultaneous access to both hierarchy and multiple model views. Key infrastructure needs in support of this design paradigm are fast, designer-friendly model-generation tools and comprehensive libraries of trusted system-level models.



```
module usystem(x1, x2);
…
parameter real A = 1 ;
…
analog begin
I(v2,v1)<+ C*ddt(V(v2,v1));
…
end
endmodule
```

```
module xcell(x1, x2);
…
parameter real k = 10;
…
analog begin
F(x2,x1) <+-k * D(x2,x1);
…
end
endmodule
```

**Figure 1.3** VLSI design methodology illustrating two levels in the hierarchical representation, where each component has multiple views. Several possible model views are shown, although only one model per component is used during simulation.

## 1.7 Analog Hardware Description Languages

Several AHDLs are available for model specification of physics governed by differential and algebraic equations. AHDLs are supported by commercial circuit simulators and include the open-source standards OpenMAST® [16], Verilog®-AMS [17], VHDL-AMS [18], and SystemC-®AMS [19], where AMS stands for analog mixed signal. AHDL models are decoupled from the simulation software, thereby providing users access to the algorithmic capabilities of modern simulators directly through their custom model code. System-level models written in AHDLs coupled with modern commercial electronic design frameworks provide a path to encode and document all design aspects: system architecture, device topology and sizing, process settings, multiphysics behavior with complex interactions, signal timing, and external stimuli emulating the application. Cosimulation of interconnected electronic and multiphysics devices is enabled, as the circuit simulators support system-level modeling across all energy domains.

These AHDLs allow designers to create modules that encode component models as behavioral (mathematical) descriptions and also support structural descriptions of systems by interconnecting components. Various terms used in describing multiphysics circuits are illustrated in Figure 1.4. Each terminal or *port* on a

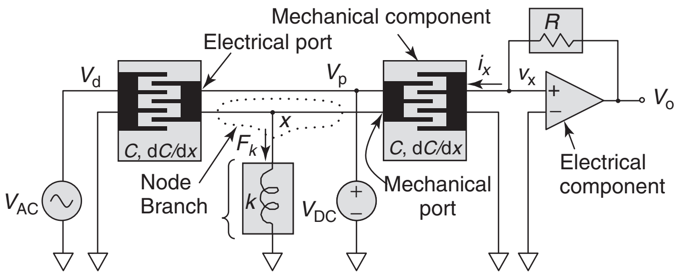


**Figure 1.4** An example of a multiphysics circuit schematic: a mechanical spring connected to two electrostatic interdigitated comb capacitors with a motional current readout circuit. Components are designated by the gray symbols. Potential variables include $v_d$, $v_p$, $v_x$, $v_o$, and $x$. Flow variables include $F_k$ and $i_x$. The ground nodes denote zero potential in all energy domains. Parameters include voltages $V_{AC}$ and $V_{DC}$, resistance, $R$, spring constant, $k$, air-gap capacitance, $C$, and motional sensitivity, $dC/dx$.

component is associated with a potential variable and a flow variable (also known as *across* and *through* variables, respectively) that are used within the model. AHDLs support both *conservative* and *signal-flow* definitions for the component ports. In a signal-flow port, only the terminal potentials are defined and used within the model. Conservative circuit interconnections follow the generalized Kirchhoff's potential law (KPL) and Kirchhoff's flow law (KFL). These laws are more generally known as *Kirchhoff's mesh rule* and *Kirchhoff's node rule*, respectively. A detailed treatment of multiphysics modeling with Kirchhoffian networks is given in Chapters 2 and 4. The definitions of KPL and KFL relate to nodes and branches in a generalized circuit. A *node* is most generally defined as any contiguous equipotential connection (i.e., wiring) between two or more component ports. A *branch* is a path of flow through a component from one of its ports to another. In KPL, the potential variable values associated with the branches around a closed loop of a circuit sum to zero. In KFL, the flow variable values flowing through all branches out of a common node of a circuit sum to zero. Most circuit simulators allow mixing of conservative and signal-flow components in a single system. In these cases, signal-flow outputs are treated as dependent potential sources and signal-flow inputs are treated as potential probes (i.e., as infinite-impedance inputs). As described in Section 1.6, this flexibility to mix component model views at any hierarchical level is especially handy for top-down design where details of a component's potential to flow relationship may be deferred to a later stage in the design process.

## 1.8
## General Attributes of System-Level Models

Interoperability is a critically important attribute of models to enable their use with other models to form systems. The potential and flow variables must be consistent in definition and in their associated reference direction. Interoperability is taken for granted in electronic circuit design. Electrical terminal standards – current as flow, voltage as potential, and the associated reference direction of current flow into the positive potential terminal to mean that power is delivered to the component – are well known and followed in device modeling efforts. Standards for macrosystem terminal relations in many energy domains have been established for AHDLs. However, multiphysics terminal relations particularly suitable for microsystems do not yet exist as accepted standards. Exacerbating adoption of standards are often significant differences in notation and approach between various microsystem designers.

Several challenges exist in appropriately and accurately translating the results of lower-level simulations to populate system-level models. Microsystems almost always involve conversion of energy across physical domains (e.g., mechanical to electrical, thermal to mechanical). Therefore, of foremost importance is the attribute that system-level models abide by principles of conservation of energy to preserve the physical integrity of the simulations in which they are used. The energy flowing into the model plus any sources internal to the model must equal the energy leaving the model plus the thermodynamic losses internal to the model.

Parameterization is a third important model attribute that enables iterative design and is supported within AHDLs. External parameters may be created to set materials properties and geometric sizing during system simulation. For example, Young's modulus and structural thickness parameter values may be set for a particular micromechanical process and then modified to allow exploration of process sensitivities. Full specification within parameterized models includes design constraints (e.g., geometric design rules) and nominal and variation values for the layer thicknesses and materials properties within a process. CMOS foundries supply this information in documents and files that comprise *physical design kits* that customize the computer-aided design environment. In an analogous manner, system-level model parameters can be formulated for compatibility with MEMS physical design kits. Examples of constraints set by MEMS design rules include the maximum proof mass size in an accelerometer, the minimum beam width in flexures, and the minimum gap in electrostatic actuators.

## 1.9 AHDL Simulation Capabilities

MEMS models in AHDLs exploit the availability of fast circuit simulators that incorporate DC, AC, and transient analysis. In general, simulations will converge if the corresponding behavioral models are formed from physical principles and if the structural models follow physically realizable interconnect rules. Simulation time is often dependent on the specific form of the model code and so can be optimized for speed. Chapter 4 addresses aspects of system-level cosimulation that have to be understood and solved if different physical energy domains are to be coupled and simulated successfully.

Most commercial circuit simulators support several additional analyses that are useful for the evaluation of microsystems designed with AHDL-based models.

Periodic ac analysis and periodic steady-state analysis provide mechanisms to simulate highly nonlinear systems that result in asymmetric periodic waveforms arising either from AC excitation or from intrinsic oscillatory behavior. In contrast, basic AC analysis linearizes around an operating point and only includes the fundamental harmonic term. Use of transient analysis for nonlinear problems may take very long to settle to a steady-state solution that displays all of the frequency content of interest. In particular, periodic analyses are helpful in simulating microsystems that employ modulation and nonlinear resonant conditions. For example, capacitive accelerometer systems that use chopper stabilization or correlated double sampling at megahertz frequencies are challenging to simulate in regular transient analysis over periods on the order of seconds needed to capture response to low-frequency input acceleration signals. Noise analyses are available to analyze the effects of stochastic disturbances originating from sources internal and external to the system. The amplitude and nature of the noise must be included in the constitutive behavioral models. In many design frameworks, these various simulation analyses are overlaid with parametric sweeps to automate exploration of the design space and with the Monte Carlo capabilities to estimate statistical distributions arising from parameter and process variations. Model support for incorporating and propagating process variations is needed to fully exploit the Monte Carlo capabilities and is essential for estimating manufacturing yield.

## 1.10 Composable Model Libraries

In MEMS, along with most other fields, the overwhelming majority of system-level designers rely on experts in device physics and in materials processing to create behavioral models. As there are few device modelers relative to system designers, a bottleneck in time and efficiency ensues. One way to help alleviate this bottleneck is to create models automatically from fine-grained simulation through MOR as described earlier. An alternate approach is to create system-level models at a low enough granularity that allows their reuse for a significantly broad design space. As long as terminal relations remain interoperable, this approach remains compatible and can be synergistic with automated MOR techniques.

Such a model reuse paradigm has existed successfully for over 40 years for circuit simulation (i.e., SPICE [20]). Widespread and free access to the evolving models in SPICE led to its adoption as a *de facto* standard. The long-term success of SPICE occurred because (i) high-fidelity models of CMOS transistors were developed, (ii) the infrastructure was created to extract model parameters from experimental process and device data, (iii) the education of designers centered on use of these models, (iv) a financial incentive for modeling activities within foundries arose to enable their propagation to external designers, and (v) there was dedication and incentive to revise models and insert new device physics as the CMOS technology advanced. It is now taken for granted by the electronics designer that models of transistors, resistors, capacitors, and inductors provide exquisite predictive accuracy regardless of how they are used in a circuit. These primitive behavioral models are *composable*, which in this context means that they are interoperable and of a basic elemental nature to enable the creation of a large number of useful structured models at higher levels in the design hierarchy.

In the mid-1990s, a series of composable and parameterized primitive models were introduced for MEMS design using handcrafted physics-based techniques. Common electromechanical composable models include straight and curved beams, plates with various geometric features, electrostatic gaps, and contact points. Examples of composable model libraries are given in Chapters 17, 19, 21 and 22, with some versions supported commercially.

Physics related to distributed effects are particularly difficult to capture in composable models. For example, electrostatic fields in principle depend on the location of all conductive and dielectric components in a microsystem (unless the components are completely shielded). This is a key reason why creation of accurate general-purpose electromechanical ‘‘gap’’ models remains an open challenge. Approximations or abstractions to far-field effects must be incorporated in the system-level models to eliminate the need for computations involving all components. Other examples of distributed effects that are challenging to model include stress, covered in Chapter 6, and damping, covered in Chapter 7. The latter damping example uses a ‘‘mixed-level’’ modeling approach that takes advantage of the hierarchical nature of structured modeling.

Current composable model libraries for MEMS are extremely useful, but they are not comprehensive; there are plenty of general-purpose models that could, should, and probably will be added in the future. Much effort is required to expand and improve composable models, as they are currently handcrafted. Model libraries will naturally evolve and improve through incorporation of additional physical phenomena, such as piezoresistance, piezoelectricity, loss, and thermal properties. Other microsystem design domains, most notably microfluidics, will benefit from analogous composable libraries.

## 1.11 Parameter Extraction, Model Verification, and Model Validation

Microsystem models can be accurate only if the parameter values incorporated within the simulation reflect the actual outcomes of the manufacturing process. Process parameters include material properties, layer thicknesses, and feature offsets from layout. Determining accurate values for these parameters is a particular challenge for MEMS. Procedures to provide parameter values from experimental test data are collectively known as *parameter extraction*. Parameter extraction for multiphysics systems is currently performed with painstaking custom test structure design and measurement. An example is described in Chapter 9, where determination of materials thermal properties is based on fitting a reduced-order model to experimental measurements. Standardizing and streamlining these kinds of activities would greatly reduce the time to market for new microsystem technologies.

*Model verification* comprises procedures to check on the mathematical and structural form of a system-level model. Models created automatically through MOR techniques are verified by construct, as they are directly derived from continuum simulations. However, handcrafted microsystem models are subject to errors in form and to errors in physical assumptions. Behavior of these kinds of models must be verified by using a system-level simulation ''test bench'' and comparing results against continuum simulation using identical boundary conditions, materials property values, and geometric sizing values in both simulations. Identifying the appropriate suite of verification problems to cover all aspects of model form is challenging. To be comprehensive, the problem set must cover static and dynamic behavior up to maximum frequencies of interest; it must cover the dynamic range of interest for the input stimuli; and it must cover the design space over all external parameters. Also, suitable criteria must be defined when comparing the simulations. An uncertainty criterion that is set too small may capture effects of numerical error that have nothing to do with the model form. Too large of an uncertainty criterion may mask detection of modeling errors.

*Model validation* describes procedures that check accuracy of a system-level model when compared with experimental results. Model simulations for use in validation exercises should be performed with extracted parameter values. Subsequent experimental validation within process uncertainty is an essential requirement for trusted models. Most of the same issues that occur in verification arise when identifying an appropriate suite of validation tests. Coverage of static responses, frequency responses, dynamic range, and external parameter space are all required for complete model validation.

The microsystem designer should be wary in trusting a system-level model to provide accurate performance predictions when its operation regime is extrapolated beyond its verified and validated range. However, physics-based composable models that operate in their verified and validated range potentially provide a basis for accurate predictive behavior when used to create new structured models. Verification of the design space for structural models that comprise composable components is an open research area. Validating individual composable models is an additional challenge, as practical test structures may need to be formed from a combination of components. For example, testing the electrostatic force generated from an air gap probably requires a flexure to be connected to the gap. Discrepancies between the simulation and experiment may then arise from the flexure and not from the gap.

## 1.12 Conclusions

System-level modeling is a core activity in support of microsystems design. Various case studies on multiphysics modeling and their use in systems design are given in various chapters throughout this book. Owing to the availability of analog hardware description languages such as Verilog-AMS, coupled multiphysics and multiscale behavior can be readily incorporated within system-level models that run on fast commercial circuit simulators that support DC, AC, periodic, noise, and the Monte Carlo analyses. AHDLs allow model formation in any combination of mathematical and structured descriptions, including primitive behavioral models, signal-flow models, and circuit macromodels (Chapters 2 and 4). Accurate translation of results from continuum field simulations into system-level models is best accomplished through model order reduction techniques (Chapters 3, 9, 10, 18, 20, and 21). Progress being made in nonlinear MOR is broadening the impact of these automated approaches to system-level modeling (Chapters 10–12). To be trusted, models for systems design constructed in whole or in part with manual techniques must be verified through comparison to continuum simulation. Regardless of their origin, all models to be considered trusted must be validated through comparison with experiment using process parameter values extracted from experimental test structures.

Leveraging the principles of the VLSI design hierarchy, multiphysics system-level modeling and simulation provides inherent capability to handle ever increasing amounts of microsystem complexity. The fast simulation speed, relative to fine-grained continuum simulation, powers iterative design by enabling simultaneous exploration of trade-offs in system architecture along with topology and sizing of multiphysics components. These capabilities become increasingly important as microsystems trend toward greater integration of multiphysics components with digital and analog electronic subsystems.

Useful models intended for rapid iterative design of multiphysics systems must be interoperable and parameterized. While interoperable model libraries for microsystem design exist (Chapters 17, 19, and 22), a significant amount of resources and effort must be mustered to build comprehensive libraries of trusted models that include the plethora of physics of interest to microsystem designers. Growing a worldwide microsystem modeling community is perhaps the most practical approach to addressing the major issues of education, adoption of standards, availability of accurate generalized models, process parameter extraction, and model verification and validation (Chapter 22).

## References

1. Wachutka, G. (1995) *Sensors and Actuators A: Physical*, **47** (1–3), 603–612.
2. Senturia, S. (1998) *Proceedings of the IEEE*, **86**, 1611–1626.
3. Mukherjee, T., Fedder, G.K., Ramaswamy, D., and White, J. (2000) *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **19** (12), 1572–1589.
4. Middelhoek, S. and Hoogerwerf, A.C. (1986) *Sensors and Actuators*, **10**, 1–8.
5. Senturia, S. (2001) *Microsystem Design*, Springer, New York, NY.
6. Rebeiz, G. (2003) *RF MEMS. Theory, Design, and Technology*, John Wiley & Sons, Inc., Hoboken, NJ.
7. Solgaard, O. (2009) *Photonic Microsystems: Micro and Nanotechnology Applied to Optical Devices and Systems*, Springer, New York, NY.
8. Vahala, K. (2004) *Optical Microcavities*, World Scientific Publishing, Singapore.

**9.** Kitching, J., Knappe, S., and Donley, E.A. (2011) *IEEE Sensors Journal*, **11** (9), 1749–1758.

**10.** Lin, W.H. and Zhao, Y.P. (2005) *Microsystem Technologies*, **11** (2–3), 80–85.

**11.** Nguyen, N.T. and Wereley, S. (2002) *Fundamentals and Applications of Microfluidics*, Artech House, Norwood, MA.

**12.** Karniadakis, G., Beskok, A., and Aluru, N. (2002) *Micro Flows: Fundamentals and Simulation*, Springer, New York, NY.

**13.** Ferrari, M. (ed.) (2007) *BioMEMS and Biomedical Nanotechnology* (**4** volume set), Springer, New York, NY.

**14.** Boyle, G.R., Cohn, B.M., Pederson, D.O., and Solomon, J.E. (1974) *IEEE Journal of Solid-State Circuits*, **9** (6), 353–364.

**15.** Weste, N.H.E. and Harris, D.M. (2011) *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th edn, Chapter 14, Addison-Wesley, Boston.

**16.** Synopsys OpenMAST Language Reference Manual, version 1.0, (2004) *http://www.openmast.org/home.html* (accessed 28 July 2012).

**17.** Accelera (2009) Verilog-AMS Language Reference Manual: Analog and Mixed-Signal Extensions to Verilog HDL, Version 2.3.1, *http://www.eda.org/verilog-ams/htmlpages/public-docs/lrm/2.3.1/VAMS-LRM-2-3-1.pdf* (accessed 28 July 2012).

**18.** IEEE (2011) Standard 1076.1.1-2011. *IEEE Standard for VHDL Analog and Mixed-Signal Extensions*, *http://ieeexplore.ieee.org/servlet/opac?punumber=5752647* (accessed 28 July 2012).

**19.** Accelera and Open SystemC Initiative, (2005) SystemC Analog/Mixed-Signal Extensions, Release 1.0. *http://www.accellera.org/downloads/standards/systemc/ams* (accessed 28 July 2012).

**20.** Nagel, L.W. and Pederson, D.O. (1973) *SPICE (Simulation Program with Integrated Circuit Emphasis)*, Memorandum No. ERL-M382, University of California, Berkeley, *http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html* (accessed 28 July 2012).