Proceedings of DETC'02
ASME 2002 Design Engineering Technical Conferences
and Computer and Information in Engineering Conference
Montreal, Canada, September 29-October 2, 2002

DETC2002/CIE-34489

SUPPORTING DESIGN REFINEMENT IN MEMS DESIGN

Rajarishi Sinha[1], *Student Member, ASME*,
Institute for Complex Engineered Systems,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA.
Email: rsinha@cs.cmu.edu

Christiaan J.J. Paredis, *Member, ASME*,
Institute for Complex Engineered Systems and
Dept. of Electrical and Computer Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA.
Email: paredis@cmu.edu

Pradeep K. Khosla, *Member, ASME*,
Dept. of Electrical and Computer Engineering and
Institute for Complex Engineered Systems,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA.
Email: pkk@cs.cmu.edu

ABSTRACT

We present a framework to support design refinement during the virtual prototyping of microelectromechanical systems (MEMS). By instantiating MEMS components and connecting them to each other via ports, the designer can both configure complex systems and simulate them.

We examine design refinement in the context of ease of use and representation of the virtual prototype. We propose the use of a common, formal grammar representation for the design entities in the virtual prototype—MEMS components, behavioral models and CAD models. We show that the formal grammar approach leads to easy creation of virtual prototypes.

In this paper, we focus on ports—the fundamental building blocks of a virtual prototype. Ports mediate all interactions within and between aspects of the virtual prototype. For even moderately complex designs, there can be many interactions present. The representation and organization of all possible ports is important in the context of design refinement.

We provide a set-theoretic formalism that defines the algebra of ports. We present a formal grammar for ports that represents a port as a set of attributes, and provide a design refinement mechanism that involves adding or modifying attributes in the port.

We illustrate our framework with a MEMS example. We demonstrate that the MEMS designer can evaluate multiple design alternatives quickly and accurately with our framework.

KEYWORDS

Design methodology, MEMS, Simulation-based design, attribute grammars, port-based modeling, Modelica

INTRODUCTION AND MOTIVATION

Virtual prototyping can shorten the design cycle of MEMS products by reducing the need for expensive and time-consuming physical prototyping. The designer can evaluate more design alternatives to obtain a better quality design. In this paper, we propose to support the process of virtual prototyping of multi-disciplinary MEMS systems. We focus our attention on those aspects of virtual prototyping that are particularly important in the context of design refinement. Specifically, we further the current state-of-the-art with respect to representation and ease of use.

The system-level design process is usually top-down. The designer begins with a high-level functional description that he decomposes into sub-functions. These sub-functions are assigned to a system architecture as a configuration of components that contain both design specifications and simulation models. When further decomposition or component assignment is not desired, the designer composes the components to create a system-level configuration that is evaluated to verify the function. In this process, there are three recurring themes: *composition*, or combining subcomponents to create a compound component; *reuse*, or replacing a component

[1] Currently with IC Mechanics, 425 N. Craig Street, Suite 500, Pittsburgh, PA 15213-1154, USA. Email: raj.sinha@icmechanics.com.

with another of similar or identical function; and *refinement*, or extending the definition of a component to select or create a more specific component. It is important to define a common representation of designs entities that supports composition, reuse and refinement.

In this research, we posit that the ports, components, functional models, CAD models and behavioral models can be represented using a common framework, namely hierarchical *attributes* that represent a complete description of these entities. We define attribute grammars that govern how ports can be composed, reused or refined during the design process. We show that our approach overcomes many of the limitations of taxonomies and product hierarchies.

Instead of presenting a complete attribute-based description of all design entities, we focus on the *ports* in the design. The ports of a design entity collectively define the intended interfaces of that entity. Via the ports, the entity interacts with other entities. To represent these ports, we introduce port attribute grammars. Our attribute-based framework supports the following aspects of the design process:

Composition. In our framework, designs and simulations are created by composition of component objects [28]. Using the attribute grammars, we compose component objects by composing their constituent attributes.

Reuse through standardized representation. A library of design entities can be indexed by their attributes. Candidate entities can be selected by searching the library for particular attribute-value pairs.

Design refinement. In our framework, the design process iterates between selection and composition of components, and evaluation of their behavioral models in simulation. During each iteration, the designer uses components and models that are different in topology and/or complexity from the previous iteration. Our attributes-based framework supports transitions between design iterations through the addition or subtyping of attributes.

Organization. The *entities* used in the design are ports, component objects, component interactions, and CAD models [28], as well as functional models. They should be classified and organized so that the designer is not inundated with choices. We provide a set-theoretic formalism to organize ports.

In this paper, the focus is on laying the foundation for representing designs and models using attributes. We characterize, model and organize the interactions between components using attributes and attribute grammars for ports, and we leave the attribute representations for component objects, interaction models and functional models for future research.

We have implemented the grammars using Java and XML, and we illustrate our framework with an example of a MEMS device. Microelectromechanical systems (MEMS) have become popular in recent years as they provide low-cost, high-performance and reliable alternatives for traditional electromechanical systems. MEMS devices are manufactured using semiconductor manufacturing processes. They combine electrical and micro-scale mechanical components on the same silicon die. This allows them to enjoy the economies of scale prevalent in the semiconductor industry.

We have developed a library of system-level MEMS components that can be used for virtual prototyping of MEMS devices. The models only include the mechanical and electromechanical design aspects of the virtual prototype, and will not address the analog and digital electronics design aspects. We illustrate the use of our attribute grammars for performing design refinement of the virtual prototype.

RELATED WORK

We have created a framework known as COINSIDE (Composition In Simulation and Design) [28] where we focus on simplifying the process of model creation, by integrating form and behavior into component objects. The form, behavior and function can be represented using a common representation such as a formal grammar.

Formal grammars have been the subject of extensive research in the fields of logic programming [9] and design automation [20, 27, 30]. They have been used to formalize conceptual design [3], architectural design [32], shape generation [1, 2, 7, 8] and configuration of heat exchangers [22]. Grammars have been used to model configuration spaces of relative motions between parts [37].

Attribute grammars are extensions to formal grammars that associate attributes and rules to the tokens in the formal grammar. Due to the limitations of using formal grammars to represent semantics, attribute grammars were developed for use in the development of formal languages and software compilers [12-14, 29, 43]. These ideas have been adopted for use in the hardware design field [16, 17, 18, 19, 23, 38, 39], in VLSI design automation [4] as well as for mechanical design automation [41].

We build on these concepts and provide an attribute-based port formalism for use in both the configuration and analysis phases of the iterative design process. Although several researchers have investigated the applicability of formal grammars in mechanical design [3, 7, 20, 27], there has been little effort to formalize the representation of system-level design and simulation. In this paper, we present attribute grammars for ports in MEMS system-level design.

The modeling and design of MEMS devices has proceeded along the lines of VLSI CAD [40]. Both electrical and mechanical components are modeled as port-based objects and are connected together in a configuration. In many cases, the mechanical components exhibit behavior only in the plane of the chip, and therefore the model equations are greatly simplified.

There are several research groups and companies developing design frameworks for MEMS. CoventorWare [11] is a commercial, top-down design environment that interfaces with CAD tools like Cadence [6] and provides behavioral models and domain-specific solvers. NODAS (Nodal Design of Actuators and Sensors) [26] is a design methodology and library of hierarchical, parameterized components for simulating MEMS device behavior. SUGAR [10] is a similar tool for nodal analysis techniques that is largely implemented in MATLAB. Our simulation models are partly based on the NODAS models.

COINSIDE FRAMEWORK

The COINSIDE environment is a framework where a designer can simultaneously design and model electromechanical systems. The designer selects components from the library and connects them in a configuration. This results in a system-level design as well as a system-level behavioral model. In this section, we briefly describe the framework. For a more detailed description, see [28].

In many design processes, the target device is designed using predefined, modular components. In such processes, components are selected, configured and assembled in such a way that the design specifications are met.

In COINSIDE, components are represented by *component objects* that include complete specifications describing how they may be connected to other components in a configuration. A component object also contains CAD, behavioral and functional models. For example, a DC motor component has a shaft to connect it to a drive-train, and bolts that fasten it to a platform. The shaft and the bolts collectively form the ports or interface to this component (Figure 1).

A component is instantiated in the design by specifying *instantiation parameters*. Once instantiated, it is connected to other instantiated components via its ports. Before simulating the design, the designer selects *behavioral models* that describe its physical behavior, and *CAD models* that specify how it may be manufactured and visualized.

A configuration is created when two or more components are connected to each other via their interfaces (Figure 2). A component can itself encapsulate a configuration of components, thus allowing for the hierarchical description of systems. Multiple configurations can represent a particular component, and are bound to the configuration interface for this component. For example, a DC motor can be represented as a single component, or as a configuration of a stator and a rotor component. The candidate configurations are all equivalent specifications of the same component, and the choice of configuration is independent of the choices made for behavioral and CAD models.

Behavioral models capture the mathematical description of the physical and informational behavior of a component. For the scope of this research, we consider these models to consist of either differential-algebraic equations (DAEs) for continuous time phenomena, or discrete event systems specifications

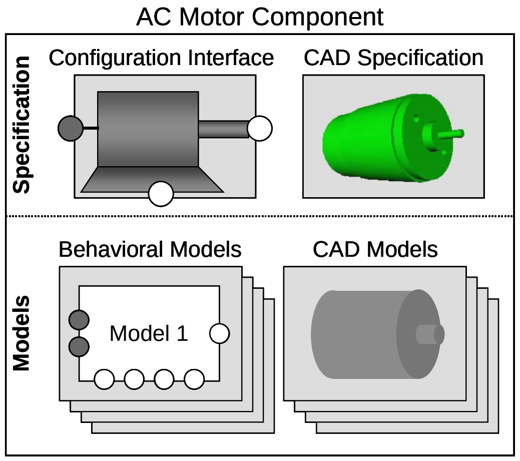

Figure 1. Component objects consist of a port-based interface for system configuration, combined with CAD and behavioral models.

(DEVS) [45]. Behavioral models can also be composed out of other behavioral models through the port-based modeling paradigm. Behavioral models in our framework are represented using the Modelica simulation language [25].

In addition to describing the internal component dynamics, behavioral models also describe the physical phenomena that act between components – the component interactions.

When a designer composes a system from components, he connects the configuration ports of components. By doing this, the designer explicitly indicates that there is an intended interaction between the connected components. The connections represent physical or information-exchange phenomena that occur at the component interfaces. These phenomena are modeled as *interaction models* for an accurate simulation.

A CAD model describes the geometry, relative positions and material properties of the parts that make up the design.

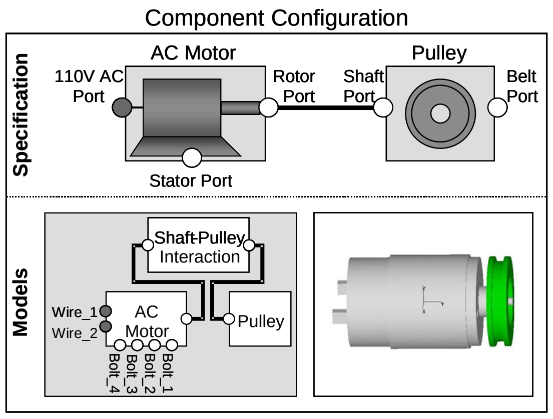

Figure 2. Component objects can be hierarchically configured into complex systems. At the same time, the behavioral and CAD models are configured also.

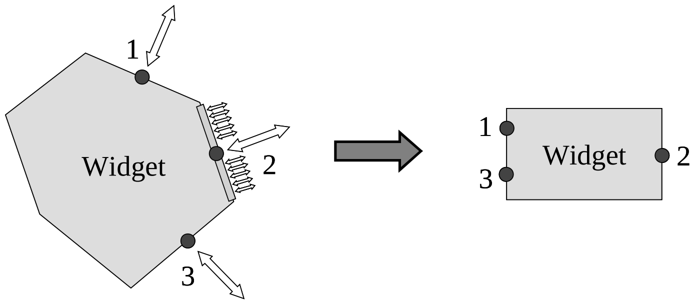

Figure 3. Ports on a component object.

Depending on the simulation experiment, the geometric representation may be simple (for bounding-box experiments), or more complex (for detailed design or manufacturing). Our framework provides the ability to use a CAD model container that contains CAD models at multiple levels of detail.

COINSIDE PORTS

A port is a descriptor for a location on the boundary of a component where the component interacts with its environment. In Figure 3, ports 1 and 3 represent point interactions, whereas port 2 represents a distributed interaction that is lumped at the port.

There are two types of ports used in COINSIDE: configuration ports and modeling ports. Configuration ports capture connection semantics between a component and its environment. For example, a DC motor component has four ports, two electrical ports, a shaft port and a stator port. The electrical ports correspond to the electrical connectors of the motor, the shaft port to the rotor and the stator port to the stator. A gear component has ports for its teeth and shaft. The gear shaft port is connected to the motor shaft port and is related to a mechanical modeling port that provides a transform for the rotational axis of the gear. The gear teeth port connects to another gear teeth port to form a gear pair, and provides information like the number of teeth and the gear pitch radius via attributes. Configuration ports can be aggregated to form more complex ports at higher levels of abstraction.

Modeling ports capture the energy exchange between behavioral models. Each connection between two behavioral models imposes some constraints on the variables of the modeling port. For a connection between simple energy or mass ports (such as mechanical, electrical, thermal and hydraulic ports), these constraints are the equivalents of the Kirchhoff voltage and current laws in electrical circuits. For signal ports, the constraint equates the value of the signal at each end of the connection.

There is a one-to-many relationship between a particular configuration port and its corresponding modeling ports. For example, a gear configuration port is related to its corresponding 3D mechanical modeling port. This relationship is captured within the component object.

Port Formalism

Ports form the basis of our framework. They are a part of the interfaces for component objects, component interactions, and behavioral models. In previous paragraphs, we defined a port as a discrete point of interaction between a component and its environment. By this definition, a port is the spatial quantum of interaction in our framework.

At a specific, *physical* location on the interface, there can be exactly one port. Therefore *location* becomes the organizing principle of a port. The geometry and material properties at the location become required attributes of every port. We model the location using a CAD feature, and material properties by a set of defining attributes such as name and physical properties (Figure 4).

Another required attribute is the *Intended Use* attribute. This attribute captures all the intended functions for the port as defined by the designer. The energy and informational domains of the port are also contained within this attribute. In addition, any special compatibility constraints on any connections to this port are specified here.

We formalize these ideas through axioms in the port formalism for a port algebra that is similar to the theory presented by Salustri and Venter [31]. Our formalism is also based on Zermelo-Fraenkel set theory [21, 36]. Here we discuss some of the important axioms (Figure 5) and their effect on the representation and use of design entities.

The Universality Axiom requires all interactions between components to occur through ports. This allows us to define the interface of a component as consisting of a set of ports.

The Uniqueness Axiom requires the ports of a component or model to be uniquely identifiable using an identifier.

The Orthogonality Axiom requires that attributes be independent of each other. This reduces the ambiguity in the grammar by not allowing an attribute to be expressed as a combination of other attributes. Therefore, a design specified using the grammar has a unique representation string.

The Finiteness Axiom states that a port consists of a finite number of attributes. Infinite attribute ports are difficult to capture using an attribute grammar. Currently, our framework permits only ports with a finite number of attributes. We leave the modeling of field interactions with potentially infinite ports

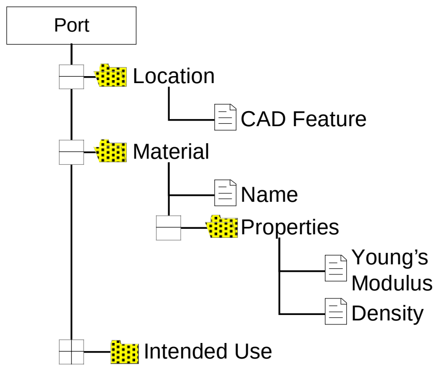

Figure 4. Attribute representation of a port.

Axiom 1:	All interactions occur through ports. (*Universality Axiom*)
Axiom 2:	The ports in the interface are ordered (*Uniqueness Axiom*)
Axiom 3:	All attributes are independent of each other (*Orthogonality Axiom*)
Axiom 4:	An attribute is modeled as a finite set of non-terminals (*Finiteness Axiom*).
Axiom 5:	Every configuration port type has at least one *compatible port type* (*Compatibility Axiom*)
Axiom 6:	If the set of attributes of two ports are identical, then the two ports are identical (*Identity Axiom*)

Figure 5. Axioms in the port formalism.

to future research.

The Compatibility Axiom states that when a connection is made between two ports, the ports must be compatible with each other. For example, the tire-road port on a tire component of a car is compatible with the running surface port of the road. Port compatibility is established as a constraint relation over the Intended Use attributes of the two ports.

The Identity Axiom establishes the equality relation over ports. Ports are equal when their constituent attributes are identical.

Using these axioms and other definitions stated in our formalism, the designer can select, instantiate, refine and connect ports.

Port Grammar

Design can be viewed as a sequence of transformations, beginning from a functional requirement and culminating in a physical device. The space of possible designs is very large—possibly infinite.

Formal grammars offer a very compact representation of a very large space such as the design space. Every point in the space is represented as a string in a language. Formal grammars are computational procedures that generate and parse the strings in the language. Several researchers have investigated the applicability of formal grammars in mechanical design [3, 7, 20, 27].

There has not been a significant effort to formalize the representation of system-level design and simulation. As a first step, we have developed attribute grammars for ports in system-level designs.

An attribute-based approach to models and design entities for simulation-based design is better than a traditional taxonomy-based approach. Firstly, instead of explicitly relating models to each other via a tree structure, we are able to describe a model by its *characteristics*. Secondly, taxonomies are not able to capture multiple-inheritance very well. In our framework, a model that is a child of two or more parents is defined as possessing at least all the attributes of all its parents. Finally, when creating new models, the designer does not have to worry about placing the models in the framework. He merely

$$port \rightarrow \texttt{location}\ \ material\ \ intended\text{-}use\ \ other\text{-}attributes$$
$$material \rightarrow property\text{-}list$$
$$property\text{-}list \rightarrow \texttt{volume-density?}\ \ \texttt{resistivity?}$$
$$intended\text{-}use \rightarrow function\text{-}list$$
$$function\text{-}list \rightarrow roll\text{-}on\text{-}surface?\ \ provide\text{-}rolling\text{-}surface?\ \ transmit\text{-}power?$$
$$roll\text{-}on\text{-}surface \rightarrow energy\text{-}domain+$$
$$provide\text{-}rolling\text{-}surface \rightarrow energy\text{-}domain+$$
$$transmit\text{-}power \rightarrow energy\text{-}domain+$$
$$energy\text{-}domain \rightarrow \texttt{mechanical}|\texttt{electrical}|\texttt{thermal}|\texttt{fluidic}|\texttt{signal}$$

Figure 6. Partial port grammar.

defines its inheritance relationships and any additional attributes that it may have, and the framework handles the storage and retrieval of the new model.

A partial grammar for ports is illustrated in Figure 6. The corresponding attributes and their meaning are presented in Figure 7. The production rules that constrain the values of the attributes are captured within COINSIDE as XML DTDs.

Refinement and Extension

Our framework supports design refinement. The designer refines the design by enriching the information content of the design entities. Intended interfaces on component objects become more detailed and elaborate, and behavioral models, interaction models and CAD models capture multiple phenomena and features of interest. The increase of detail in the interfaces to these entities is captured in the increase of detail in the attribute representation for the corresponding ports. The designer selects a particular port and adds, copies or modifies the attributes in the attribute tree of the port. The grammar maintains the consistency and correctness of the representation.

Sometimes, it may not be possible to refine a particular

Attributes	**Description**
`Location`.*name*	Name of the location of the port
`Location`.*CAD_model*	The CAD model/feature associated with the location
Material.name	The name of the material at the location
Material.description	A description of the material
`volume-density`.*symbol*	The symbol for volume density
`volume-density`.*units*	The units of volume density
`Resistivity`.*symbol*	The symbol for resistivity
`Resistivity`.*units*	The units of resistivity
roll-on-surface.name	The name of the non-terminal
roll-on-surface.description	A description of the non-terminal
Provide-rolling-surface.name	The name of the non-terminal
Provide-rolling-surface.description	A description of the non-terminal
Transmit-power.name	The name of the non-terminal
Transmit-power.description	A description of the non-terminal
Transmit-power.constraints	Constraints on the transmission of power

Figure 7. Attributes for the partial port grammar in Figure 6.

port with the attributes that are available to the designer. In such situations, the designer can extend our grammar, provided he follows certain rules in defining new port attributes. These rules are captured in the formalism. We represent the port formalism and grammar using an XML DTD [42], and the port as an XML document based on the DTD. The formalism imposes constraints on the types of ports that can be defined, as well as on the types of ports that can be connected [33].

MEMS EXAMPLE

To demonstrate CONSIDE's capabilities, we have implemented a library of MEMS components. The components represent mechanical and electromechanical building blocks commonly used in MEMS system-level design—beams, rectangular plates, anchors, springs, and comb drives. These component objects capture 2D mechanical and electromechanical behavior. Modelica is used as the behavioral modeling language and Dymola [15] is used as the solver. For more details, refer to [35]. To illustrate the concepts developed in the previous sections, we use an example of a MEMS device—a resonator.

Product Overview

The measurement of acceleration has a wide variety of applications in the industrial world [5]. For instance, acceleration sensors are used in crash detection for air bag deployment in cars, or vibration analysis in industrial machinery. Recently, traditional accelerometers have been replaced by MEMS-based accelerometers because of performance, weight and power consumption considerations.

MEMS-based resonant accelerometers (sometimes referred to as vibrating beam accelerometers) have a *proof mass* that loads two vibrating suspensions on opposite sides of the proof mass [44]. When the accelerometer proof mass is loaded, one suspension is put into tension and the other into compression. These suspensions are continually excited at frequencies of tens to hundreds of kilohertz range when unloaded. As a result, when "g" loaded, one suspension's frequency increases while the other suspension's frequency decreases. This difference in frequency is a measure of the device's acceleration. This form of accelerometer is an open-loop device, in that the proof mass is not rebalanced to its center position when force is applied. For accuracy, it relies on the scale-factor stability inherent in the material properties. Fabrication techniques result in low-cost, highly reliable accelerometers with a measurement accuracy of better than 100-μg bias error. Quartz and silicon micromechanical resonator accelerometers are being developed in the industry. They have proliferated widely into several commercial applications (e.g., factory automation).

In this section, we will follow the process of design of the resonator, and use the port grammar to demonstrate automatic selection, abstraction and design refinement of the suspension ports and electrostatic ports.

Initial System Architecture

The high-level function of the resonator can be mapped to multiple sub-functions. The sub-functions correspond to "provide inertia", "provide flexural stiffness", "actuation", and "fix to wafer". They are then mapped to one or more system architectures. The framework for this mapping is the subject of current research [24], and is not within the scope of this paper.

We will assume that a system architecture has been generated from the functional description. The system architecture under consideration is shown in Figure 8. It consists of a mass attached to the wafer substrate through a spring suspension, and actuated by an actuator. The actuator is driven by a signal, and moves the mass-spring system.

Initial Design and Modeling

Our framework supports the configuration of components in the virtual prototype by instantiating and connecting them. So the first step is to select the components that will constitute the virtual prototype.

The initial design is a configuration as shown in Figure 8. The designer decides that the springs will be placed symmetrically about the mass, and that there will be two actuators. The design is refined by adding attributes to the ports so that they become aggregate ports. The refined design is shown in Figure 9.

Once the configuration is complete and all component interfaces are connected, the designer proceeds to the modeling layer to select behavioral models.

The choice of behavioral model depends on the design stage and the requirements of the simulation. The physical behavior of the resonator structure occurs in three domains—mechanical, electrical and electrostatic. A high-level model for each of these domains captures the overall behavior of the resonator.

At this early stage of design, geometric information about the resonator is not available. The mechanical behavior of the accelerometer can be modeled as an inertial mass attached to a spring suspension. The spring model provides a restoring force with a stiffness of *k*. The input to this system is the external force on the inertial mass.

The designer considers two models for the spring: one with flexibility along 1 axis, and another more elaborate model with a 2D stiffness. In both cases, the constitutive equation is of the form:

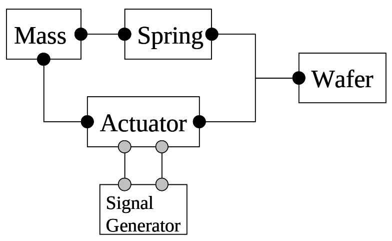

Figure 8. Resonator system architecture.

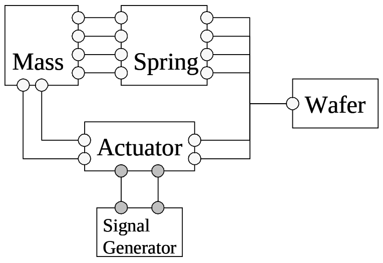

Figure 9. Design with expanded ports.

$$F = Kx \tag{1}$$

Where F is the force, x is the displacement and K is the stiffness coefficient (matrix). With this ideal model, simulation is performed to verify the function of the device.

Electrically, the proof mass and the springs behave like a resistor. Since silicon is an electrically conductive material, the structure can be modeled as a network of resistors. Applying a voltage difference between two points on the structure results in a current flow through the resistor network.

The electrostatics behavior of the resonator is confined to the comb drives. The rotor and stator are separated by an air gap. A voltage difference applied across the gap results in a translational motion. Conversely, a translation will give rise to a measurable voltage difference between the rotor and the stator of the drives.

When modeling is complete, it is possible to simulate the virtual prototype by translating the behavioral models into Modelica and evaluating them in Dymola.

Refinement of Design and Model

To obtain a more realistic simulation, the designer decides that further refinement is necessary in the components. In this scenario, the designer elaborates the mass and spring components in the configuration layer and selects refined models in the modeling layer.

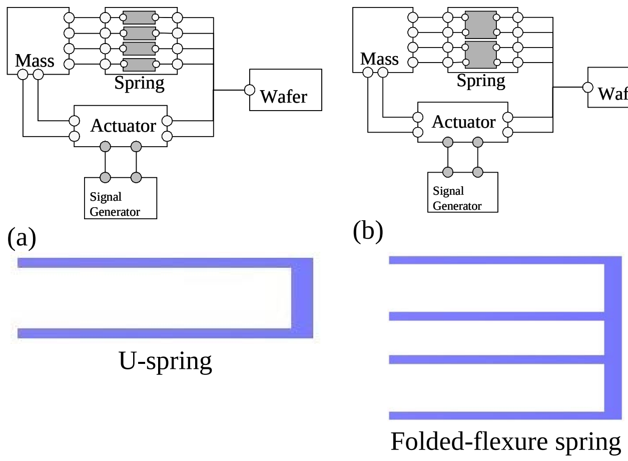

Figure 10. Two design alternatives for the resonator.

The designer considers two design alternatives: a U-spring design and a folded flexure spring design for the suspensions. For each case, he elaborates the spring component. The elaborated configurations are shown in Figure 10(a) for the U-spring and Figure 10 (b) for the folded flexure spring. He creates embodiments for each candidate spring component and connects them together in the configuration. COINSIDE automatically connects together the corresponding CAD models and behavioral models. The geometry for case (a) and (b) is shown in Figure 11 and Figure 12, respectively.

Once the configuration is complete, the designer selects behavioral models. The suspensions are now modeled as flexible beams. The mass is modeled as a combination of rigid plate models, and the comb drives are modeled as a rotor mass and a stator mass interacting through an electrostatic port.

Interaction Modeling

The electrostatic interaction between the fingers of the rotor and the stator is a dynamic phenomenon that is present at the interface. This phenomenon must also be captured in a model so that a correct system-level behavioral model is obtained [34]. In this paper, we focus on the electrostatic port that is present in the comb drive actuator.

The electrostatic port (Figure 13) is located on the fingers of the stator and on the rotor. It maps to a mechanical and an electrical modeling port. The material at the port is stored as an attribute in the attribute tree. The intended use attribute of the port indicates that it is meant to be used on a comb drive that converts mechanical energy to electrostatic energy. Based on the attribute representation, an electrostatic interaction model is chosen for the comb drive (Figure 14).

This provides all the necessary information to complete the system model and evaluate it in a Modelica solver. Both design alternatives (a) and (b) are evaluated and compared by the designer.

Results

An important characteristic of the MEMS structure is its resonant frequencies. When the structure is loaded with a sinusoidal load of a specified frequency, it vibrates about its rest position. At particular frequencies, there is energy build-up within the structure that can cause mechanical fatigue and

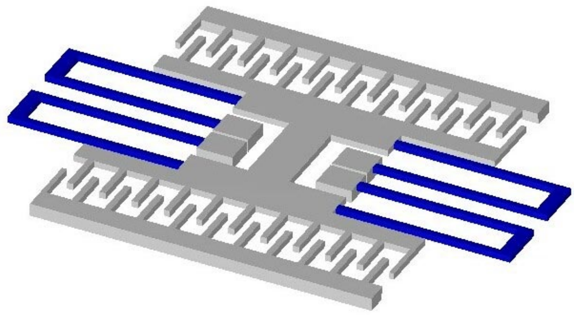

Figure 11. Resonator mechanical structure for case (a).

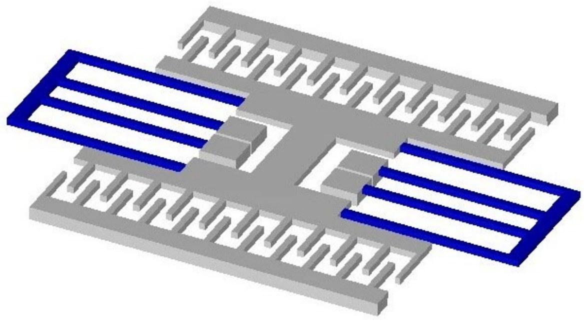

Figure 12. Resonator mechanical structure for case(b).

failure. Designers of MEMS systems design their structure such that it does not have resonant frequencies in the operating range of their device.

The designer obtained the resonant frequencies for the structure by a finite element analysis. All the frequencies are in the hundreds of KHz. The typical operating frequencies of such a resonator are tens of KHz. Therefore under normal operating modes, the resonator will not excite these resonating frequencies. The first 6 modes correspond to the $\theta_x, z, y, \theta_z, \theta_y, x$ modes respectively. The 7th and higher modes are plate bending modes and are neglected.

These results can be verified by performing an AC analysis on the system-level model of the structure. The designer obtained a θ_z resonant frequency of 389 KHz that is within 10% of the results from an Ansys finite-element simulation (340.85 KHz). The x and y resonant frequencies from were 532.1 KHz and 254.7 KHz, respectively. These values also correlate well with the Ansys results—450.77 KHz and 249.34 KHz, respectively.

Discussion

MEMS are small, intricate devices that must perform accurately and reliably, often in the hostile environments of

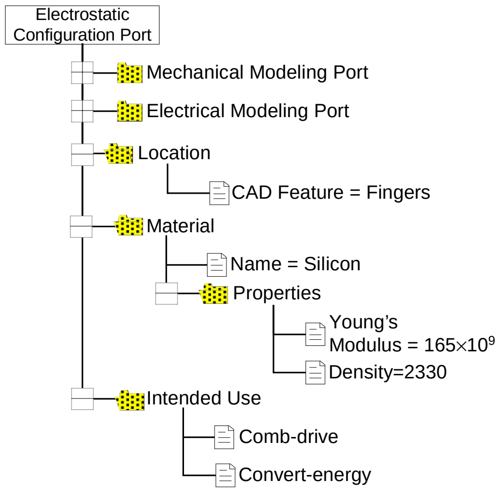

Figure 13. Electrostatic port.

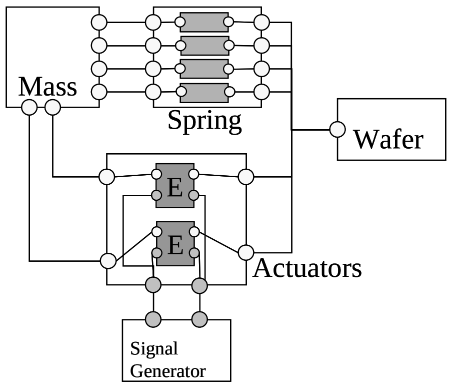

Figure 14. Behavioral model configuration with electrostatic interaction models (E) in the actuator.

vehicles and industrial machines. As a result, testing under field conditions is difficult, and MEMS engineers must rely on virtual prototyping to verify the behavior of these microstructures. Another complication in MEMS design is that many physical phenomena interact to affect the operation of the devices, including mechanical driving forces as well as resonant behaviors, thermal levels, piezoelectric effects, and electro-magnetic interference. Because many of these effects are interdependent, predicting output and performance of a MEMS device is a complex problem that requires high fidelity simulation.

Developers of MEMS also have obstacles to overcome in prototype testing. Conventional electromechanical devices may undergo several physical prototyping, test and redesign cycles. However, the initial semiconductor fabrication setup for MEMS is very costly and time-intensive and prototype testing is almost always performed to validate the final design rather than to identify design flaws at intermediate stages.

In this MEMS design scenario, we have demonstrated how a common representation and a port-based compositional approach support design refinement over the virtual prototype.

We used the port grammar to elaborate the intended interfaces of the spring component objects and the electrostatic interaction models. As the components became more detailed, the port representation was also refined to accommodate the additional information that was available at each stage of the design process.

The attribute-based representations of ports can be stored in a library. When defining intended interfaces for component objects or models, the designer can reuse previously created ports by searching the library based on the desired characteristics of the port.

The port-based modeling paradigm imposes constraints on the types of models that can be defined. In particular, all ports represent discrete locations on the interface. *Distributed interactions* can also be captured within our framework. Instead of a discrete location, a surface on the interface is involved in the interaction. The entire surface is represented by an aggregate configuration port. *Field interactions* (e.g. the gravity interaction) involve every physical location within one component object interacting with every physical location within the other component object. In this case, the interface extends to the entire mass of the component object. When one of the component objects is decomposed into subcomponents, the interaction now involves each of the subcomponents. This is difficult to represent in our framework.

SUMMARY

We presented a framework where designers can create virtual prototypes of MEM systems by configuring components, while simultaneously selecting and assigning CAD parameters and behavioral (simulation) models.

Our framework supports the designer throughout the design process by providing mechanisms for abstraction, automatic model selection and model reuse. We defined a port formalism and grammar that enables these mechanisms while defining intended interfaces for components and models.

In our implementation, we used XML to define our port grammar. Using the grammar, we represented the intended interfaces for components and models commonly used in MEMS system-level design. We captured the corresponding dynamic models in Modelica. We illustrate the usefulness of the framework with a system-level design scenario for a MEMS-based resonator that resulted in two different alternatives for the device structure.

In the future, we plan to extend our framework to incorporate distributed interactions and finite element behavioral models, and add other MEMS components to our library.

ACKNOWLEDGMENTS

We would like to thank the other members of the Composable Simulation team for sharing their insights in simulation-based design and virtual prototyping. We thank the members of the CMU MEMS group for fruitful discussions about NODAS.

This research was funded in part by the National Science Foundation under grant # CISE/115/KDI 98 73005, by the Pennsylvania Infrastructure Technology Alliance, and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

REFERENCES

[1] Agarwal, M., Cagan, J., and Constantine, K. G., "Influencing generative design through continuous evaluation: associating costs with the coffeemaker shape grammar," *(AI EDAM) Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 13, pp. 253-75, 1999.

[2] Agarwal, M. and Cagan, J., "On the use of shape grammars as expert systems for geometry- based engineering design," *(AI EDAM) Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 14, pp. 431-9, 2000.

[3] Andersson, K., "A vocabulary for conceptual design. Part of a design grammar," presented at Formal Design Methods for CAD. IFIP TC5/WG5.2 Workshop, Tallinn, Estonia, 1994.

[4] Baldwin, R. and Moon Jung, C., "Design methodology management using graph grammars," presented at Proceedings of 31st ACM/IEE Design Automation Conference, San Diego, CA, USA, 1994.

[5] Barbour, N. and Schmidt, G., "Inertial sensor technology trends," presented at 1998 Workshop on Autonomous Underwater Vehicles, Cambridge, MA, 1998.

[6] Cadence, "Cadence,",, 4.4.5 ed. San Jose, CA: http://www.cadence.com, 2001.

[7] Cagan, J., "Discussion: research issues in the application of design grammars," presented at Formal Design Methods for CAD. IFIP TC5/WG5.2 Workshop, Tallinn, Estonia, 1994.

[8] Carlson, C., "Shape grammars in design: discussion. Design space description formalisms," presented at Formal Design Methods for CAD. IFIP TC5/WG5.2 Workshop, Tallinn, Estonia, 1994.

[9] Chomsky, N., "On Certain Formal Properties of Grammars," *Information and Control*, vol. 2, pp. 137-167, 1959.

[10] Clark, V., Zhou, N., and Pister, K. S. J., "MEMS Simulation Using SUGAR v0.5," presented at Solid-State Sensors and Actuators Workshop, Hilton Head Island, SC, 1998.

[11] Coventor, "CoventorWare,". Cary, NC, 2002.

[12] Deransart, P., Jourdan, M., and Lorho, B., "Survey on attribute grammars. Part III. Classified bibliography,",, pp. 74, 1985.

[13] Deransart, P., Jourdan, M., and Lorho, B., "Survey on attribute grammars. Part II. Review of existing systems,",, pp. 140, 1986.

[14] Deransart, P., Jourdan, M. M., and Lorho, B., "Survey on attribute grammars. Part I. Main results on attribute grammars,",, pp. 54, 1986.

[15] Dynasim Ab, "Dymola,". Lund, Sweden: Dynasim AB, 1999.

[16] Economakos, G., Papakonstantinou, G., and Tsanakas, P., "AGENDA: an attribute grammar driven environment for the design automation of digital systems," presented at Proceedings Design, Automation and Test in Europe, Paris, France, 1998.

[17] Economakos, G. and Papakonstantin, G., "An attribute grammar driven high-level synthesis methodology for VHDL specifications," presented at Proceedings of SCS'99: International Symposium on Signals, Circuits and Systems, Iasi, Romania, 1999.

[18] Economakos, G. and Papakonstantinou, G., "Refinement and property checking in high-level synthesis using attribute grammars," presented at Proceedings of 10th Conference on Correct Hardware

Design and Verification Methods, Bad Herrenalb, Germany, 1999.

[19] Economakos, G. and Papakonstantinou, G., "A formal method for hardware design using attribute grammars," presented at ICECS'99. Proceedings of ICECS'99. 6th IEEE International Conference on Electronics, Circuits and Systems, Pafos, Cyprus, 1999.

[20] Fitzhorn, P. A., "Engineering design is a computable function," *(AI EDAM) Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 8, pp. 35-44, 1994.

[21] Hayden, S. and Kennison, J. F., *Zermelo-Fraenkel Set Theory*, 1 ed. Columbus, OH: Charles E. Merrill Publishing Co., 1968.

[22] Hedin, G., Ohlsson, L., and Mckenna, J., "Product configuration using object oriented grammars," presented at System Configuration Management. ECOOP'98 SCM-8 Symposium. Proceedings, Brussels, Belgium, 1998.

[23] Holmgren, F., "A Grammar-based Approach to Design and its Application to Electronics and Logic Programming," in *Department of Computer and Systems Sciences*. Stockholm, Sweden: Stockholm University, 1997.

[24] Liang, V.-C., "Ph.D. thesis proposal," in *Institute for Complex Engineered Systems*. Pittsburgh, PA: Carnegie Mellon University, 2002.

[25] Mattsson, S. E., Elmqvist, H., and Otter, M., "Physical system modeling with Modelica," *Control Engineering Practice*, vol. 6, pp. 501-510, 1998.

[26] Mukherjee, T. and Fedder, G. K., "Hierarchical mixed-domain circuit simulation, synthesis and extraction methodology for MEMS," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 21, pp. 233-49, 1999.

[27] Mullins, S. and Rinderle, J. R., "Grammatical approaches to engineering design. I. An introduction and commentary," *Research in Engineering Design*, vol. 2, pp. 121-35, 1991.

[28] Paredis, C. J. J., Diaz-Calderon, A., Sinha, R., and Khosla, P. K., "Composable Models for Simulation-Based Design," *Engineering with Computers*, vol. 17, pp. 112-128, 2001.

[29] Parisi-Presicce, F., "Modular system design applying graph grammars techniques," *Automata, Languages and Programming. 16th International Colloquium Proceedings*, pp. xi+788, 621-36, 1989.

[30] Rinderle, J. R., "Grammatical approaches to engineering design. II. Melding configuration and parametric design using attribute grammars," *Research in Engineering Design*, vol. 2, pp. 137-46, 1991.

[31] Salustri and Venter, "An Axiomatic Theory of Engineering Design Information," *Engineering with Computers*, vol. 8, pp. 197-211, 1992.

[32] Sariyildiz, S., Durmisevic, S., and Ciftcioglu, O., "Integrating pattern grammar and wavelets in architectural design," presented at 6th International Conference on Computer Graphics and Visualization 98, Plzen, Czech Republic, 1998.

[33] Sinha, R., Paredis, C. J. J., and Khosla, P. K., "Modeling of Component Interactions in Configuration Design," Carnegie Mellon University, Pittsburgh, PA, Technical Report 2001.

[34] Sinha, R., Paredis, C. J. J., and Khosla, P. K., "Interaction Modeling in Systems Design," presented at ASME Design Engineering Technical Conferences, Pittsburgh, PA, 2001.

[35] Sinha, R., "Compositional Design and Simulation of Engineered Systems," in *College of Engineering*. Pittsburgh, PA: Carnegie Mellon University, 2002, pp. 238.

[36] Spivey, J. M., *The Z notation: A reference manual*, 2nd ed. New York: Prentice Hall International, 1992.

[37] Taentzer, G. and Schween, H., "Movement of objects in configuration spaces modelled by graph grammars," *Graph Grammars and Their Application to Computer Science. 4th International Workshop. Proceedings*, pp. x+703, 660-75, 1991.

[38] Tanaka, T., "Definite-clause set grammars: a formalism for problem solving," *Journal of Logic Programming*, vol. 10, pp. 1-17, 1991.

[39] Tanaka, T. and Jain, L. C., "Circuit representation in a logic grammar," presented at Proceedings Electronic Technology Directions to the Year 2000, Adelaide, SA, Australia, 1995.

[40] Tang, W. C., "Overview of microelectromechanical systems and design processes," presented at 34th annual conference on Design automation conference, Anaheim, California, 1997.

[41] Tyugu, E., "Attribute models of design objects," presented at Formal Design Methods for CAD. IFIP TC5/WG5.2 Workshop, Tallinn, Estonia, 1994.

[42] W3c, "Extensible Markup Language (XML),".: World Wide Web Consortium, 1999.

[43] Warren, J. R. and Crosslin, R. L., "Development of a system simulation grammar for systems analysis and design," *Proceedings of the 1990 Summer Computer Simulation Conference*, pp. xix+1202, 880-5, 1990.

[44] Yazdi, N., Ayazi, F., and Najafi, K., "Micromachined inertial sensors," *Proceedings of the IEEE*, vol. 86, pp. 1640-59, 1998.

[45] Zeigler, B. P., Praehofer, H., and Kim, T. G., *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd ed: Academic Press, 2000.