Issues in MEMS Macromodeling

Gary K. Fedder
Department of Electrical and Computer Engineering
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA USA 15213

Abstract

AHDL modeling of MEMS components for system simulation is slowly maturing, however many issues remain regarding interoperability and composability. A discussion of MEMS natures, hierarchy and model partitioning helps to introduce the issues. Specific conventions for interoperability include choice of component reference frame and associated reference directions. A composite discipline extension to AHDLs is proposed to enable a single-port representation of micromechanical connection points in 2-D and 3-D space.

I. Introduction

Microelectromechanical Systems (MEMS) are components with micron-scale moving parts made from the materials and processes of microelectronics fabrication. They convey the advantages of miniaturization, multiple components and microelectronics, and enable on-chip integration of electronics, microstructures, microsensors and microactuators. MEMS are increasingly playing a key role in linking information technology more directly with the physical world. However, time-efficient component design is a key bottleneck to realizing the commercial potential of MEMS.

Accurate dc simulation of MEMS requires accurate modeling of the effects from mechanical forces, electrostatic forces and intrinsic stresses. Transient and ac simulations require additional modeling of inertial and damping forces. Other physical effects, such as heat transfer, thermal expansion, piezoelectric stresses and piezoresistive effects, are needed for specific applications. The list of possible effects is so long that many are deemed negligible and not included in MEMS models, even at final design verification. Poor design decisions may result from a reliance on this ad hoc modeling approach.

One of the central concepts of electronic circuit design is hierarchy, which also applies well to MEMS. MEMS designers partition microstructures into functional elements, such as flexural springs, electrostatic actuators and capacitive sensors. Today, pre-formed models for the exact topology of interest rarely exist, and so the modeling effort usually accounts for the greatest amount of time within a design cycle. Design reuse with verified model libraries that contain all physical effects is rarely used, but sorely needed.

Taking the analogy to electronic circuit design further, the next generation of MEMS system designers are starting to use composable MEMS models. That is, models that provide physically correct and accurate simulation results when connected together in an arbitrary manner. The most current work in forming composable MEMS models is SUGAR from UC Berkeley [3], Coventor ARCHITECT [1] [2] and NODAS from Carnegie Mellon [4][5]. ARCHITECT and NODAS use Analog hardware description language (AHDL) descriptions, while SUGAR has its models written in MATLAB.

To take full advantage of AHDL descriptions, conventions must be adopted by the designer to ensure interoperability between module instances so that design by structural composition is reduced to interconnection of MEMS cells from pre-made libraries. Unlike electric circuits, the first-order behavior of MEMS usually depends on the shape and location of the devices and on geometrical interactions between the devices and moving boundaries. Incorporating these effects into models in a self-consistent and comprehensive way is necessary.

II. Natures and Disciplines

MEMS models involve mechanical disciplines, where the choice of the natures are a potential source of confusion. The potential term in simulations of micromechanical translational motion can be defined as velocity, absolute position or displacement from a reference (“mechanical ground”) state. The product of the potential (voltage) and flow (current) in the electrical discipline provides a conservative physical quantity (power). This same logic motivates the use of velocity as potential and force as flow, where the velocity×force product, mechanical power, is conserved by the simulator. However, it is equally valid to choose displacement as potential and force as flow so that mechanical work (i.e., energy) is conserved. Therefore, this choice of mechanical discipline for model ports is primarily dependent on ease in interpretation and analysis of simulation results.

In a majority of cases, translational and angular displacements are the critical variables of greatest interest in analysis of MEMS. Access to only translational and angular velocities as the across variables would require integration and combining of across variables to determine displacements, which is not likely to be a user-friendly post-simulation step. The displacement calculations end up having to be imple-

mented by the model in any case in order to calculate spring forces.

Moving micromechanical structures are displaced with respect to the initial positions that are described by the layout. Displacement, rather than absolute position, is a more natural choice for across variable because:

- The layout location of a microstructure is a natural reference, with displacement from the layout position describing its motion. Displacements will be zero when the system is at rest, i.e, when all external and internal forces are turned off, as shown in Figure 1(a). In contrast, positions will not be zero at rest, leading to an extra step in extracting the motion from the simulation output if they are chosen as the across variables.
- If position is used, a reference coordinate frame needs to be defined. If an off-chip reference is used, and the chip itself is moved, then the potential difference between two displaced structures on the same chip may involve the subtraction of large numbers leading to numerical accuracy limitations, as shown in Figure 1(b). A more practical example is the simulation of a MEMS accelerometer in a car that moves several kilometers, while the beams within the MEMS accelerometer may move less than a nanometer.

The standard discipline definitions in Verilog-AMS do not fully account for the above reasoning for MEMS applications. MEMS models can be implemented using the existing standards, however the naming of the natures and access functions do not precisely correspond to the modeling intent, and may lead to misunderstanding if code is propagated to other groups for reuse. Therefore, a proposed set of MEMS disciplines and natures that are consistent with the modeling intent is given in Listing 1. *Disp* and *Phi* are used as access functions for translational and angular displacements, respectively, to distinguish from and not conflict with the

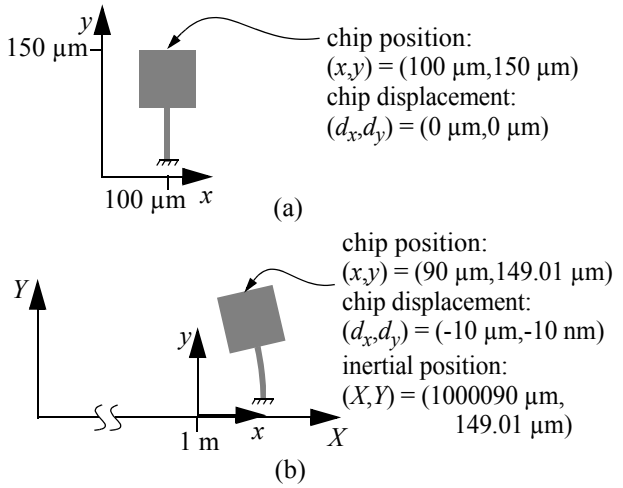

Figure 1: Reference frames of microstructures. (a) Position and displacement at rest in chip frame of reference (*x*, *y*). (b) Position and displacement in inertial (*X*, *Y*) and chip frames of reference after acceleration moving chip 1 m in *X*.

```
nature Displacement
  units = "m";
  access = Disp;
  ddt_nature = Velocity;
  abstol = 1e-12;
endnature

nature Force
  units = "N";
  access = F;
  abstol = 1e-12;
endnature

nature Velocity
  units = "m/s";
  access = Vel;
  ddt_nature = Acceleration;
  idt_nature = Displacement;
  abstol = 1e-9;
endnature

nature Angular_Displacement
  units = "rad";
  access = Phi;
  ddt_nature = Angular_Velocity;
  abstol = 1e-6;
endnature

nature Angular_Force
  units = "N/m";
  access = Tau;
  abstol = 1e-12;
endnature

nature Angular_Velocity
  units = "rad/s";
  access = Omega;
  ddt_nature =
        Angular_Acceleration;
  idt_nature = Angular_Velocity;
  abstol = 1e-6;
endnature

discipline kinematic_translational
  potential Displacement;
  flow Force;
enddiscipline

discipline kinematic_rotational
  potential Angular_Displacement;
  flow Angular_Force;
enddiscipline

discipline velocity
  potential Velocity;
enddiscipline

discipline angular_velocity
  potential Angular_Velocity;
enddiscipline
```

Listing 1: Abbreviated list of disciplines and natures proposed for use in composable MEMS models.

standard position and angle natures. The *velocity* and *angular_velocity* disciplines are created to handle the signal flow state variables internal to the MEMS models.

Absolute tolerances (abstol) in Listing 1 are set to values appropriate for a majority of MEMS applications. "Large" displacements in MEMS are usually on the order of microns, while sub-nanometer motions are common in inertial sensors and RF resonators. Therefore, a displacement abstol of 1 pm is appropriate. Although this size of motion is smaller than the atomsmaking up a microstructure, time-averaged picometer displacements are possible to detect. The design "sweet spot" for MEMS flexural-mode resonators is in the kHz range, so the velocity abstol is set at 1 nm/s, which is 1000 times higher than the displacement abstol. Although mN scale forces are achievable in MEMS, most actuators deliver less than several µN. In surface probe applications, nN level forces are of interest. Therefore, a 1 pN abstol is appropriate. Angular displacements and forces are not common MEMS design variables, except in rotational actuators (e.g., secondary actuators for disk-drives) and in angular accelerometers. The abstol values are somewhat arbitrarily set to 1 µrad and 1 µrad/s and 1 pN/m.

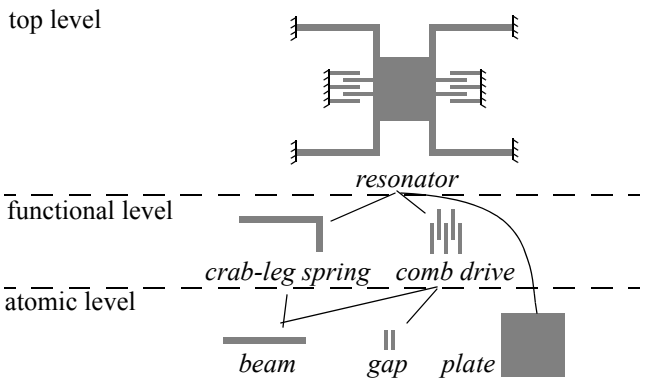

Figure 2: Design hierarchy of a resonator with a crab-leg suspension and symmetric electrostatic comb drives. Components are shown in layout view.

III. Hierarchy

MEMS hierarchy is exemplified in Figure 2 for a microstructure with a crab-leg suspension and two electrostatic comb-drives. Mass-spring microstructures, similar to this crab-leg example, are commonly used for material property test structures, microresonators, inertial sensors and gravimetric vapor sensors. The resonator may be partitioned into functional-level elements including a shuttle mass, crab-leg springs and electrostatic comb drives, then further decomposed into "atomic-level" elements including beams, plates and electrostatic gaps. Atomic-level beams and plates are similar to the kinds of lumped elements supported by commercial finite-element tools. Models at this lowest level have broad applicability at the expense of creating larger system matrices for simulation.

The number of degrees of freedom (DOF) required for accurate simulation usually will affect the choice of the behavioral model. As an example, an electrostatic comb microactuator can have multiple vibrational modes, even vibrations of the individual comb fingers. In this latter case, the mechanical DOF relative to finger deflection must be included. An atomic-level representation of the microactuator may be built through structure using interconnected beam models and electrostatic gap models, and includes DOF for motion of each finger. An equivalent functional-level behavioral model is possible, but requires custom encoding of equations of motion for all DOF.

IV. Crab-Leg MEMS Model Example

Symmetric "crab-leg" suspensions, such as those illustrated in Figure 3 are commonly used in resonators and simple platforms. The crab-leg design alleviates nonlinear stiffening and buckling from axial stress found in fixed-fixed suspensions.

A simple 1-D model for the *y*-directed center displacement of the topology in Figure 3(a) is shown in Listing 2. It is straightforward to implement the additional kinematic DOF for the center point to form a 2-D or 3-D model. Effects of higher-order vibrational modes and nonlinear elastic effects can be added to the model, of course with increased effort in figuring out the physics.

Although the model in Listing 2 has geometric parameters, it is limited in design reuse to a single topology. Likewise, the model as a MEMS building block is restricted to mechanical connections to the center point of the plate that maintain the simple *y*-directed motion. For example, if an electrostatic actuator was to be attached to the left and right sides of the plate, as in Figure 2, the resonator model would need to be overhauled to accommodate connections on its sides, or a special actuator model would have to be made.

In an attempt to provide more model reuse, the individual crab-leg spring, one located at each of the plate's four corners in Figure 3, can be modeled at the corresponding "functional" level. However, external forces in the *x* direction and moments around the *z* axis affect the behavior in the *y* direction. Therefore, a *y*-directed spring model that allows arbitrary boundary conditions at its ends is impossible to implement if only the single DOF is accessible. A reusable micromechanical model that accounts for arbitrary boundary conditions must include access to all generalized displacements and forces (for 2-D in-plane: d_x, d_y, ϕ_z and F_x, F_y, M_z).

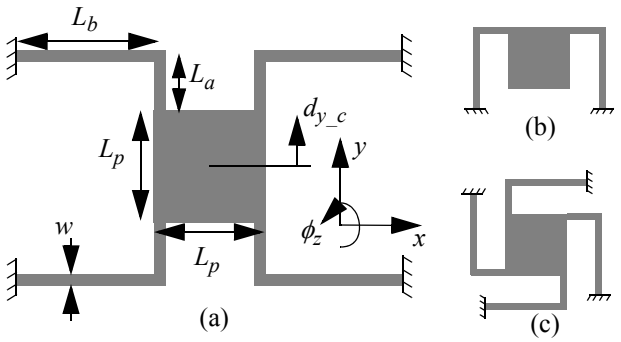

Figure 3: Three selected layout topologies of crab-leg suspended microstructures.

```
module crableg1D (dy);
inout dy;
kinematic_translational dy;
velocity vy;
parameter real La = 1e-5;
parameter real Lb = 1e-4;
real density = 2330, E = 165e9, viscosity = 1.79e-5, deff = 2e-6;
real h = 2e-6, w = 2e-6, Lp = 1e5;
real m, b, k;
analog begin
  m = density*h*(Lp^2+4*w*La);
  b = viscosity*Lp^2/deff;
  k = E*h*(w/Lb)^3*(4*Lb+La)/(Lb+La);
  Vel(vy) <+ ddt(Disp(dy));
  F(dy) <+ m*ddt(Vel(vy)) + b*Vel(vy) + k*Disp(dy);
  end
endmodule
```

Listing 2: 1-D, 1-DOF behavioral model of a crab-leg resonator.

An example functional-level schematic of a 2-D crab-leg resonator is given in Figure 4 with partial model code in Listing 3 Each 2-D spring element has three terminals on each of its two ends, contributing six DOF to the system matrix. (A 3-D spring element would have six terminals on each end, corresponding to d_x, d_y, d_z, ϕ_x, ϕ_y, ϕ_z and 12 DOF.) In micromechanics, branch flow is not necessarily equal on both sides of the element. For example, the flow variables corresponding to moment (*Tau(phi_za)* and *Tau(phi_zb)* in Listing 3) will normally not be equal and opposite on the two sides of a spring element. Although the sum of moments acting on an element must be zero, any translational forces acting over a distance contribute to this sum. Unless the external forces are imposed in a way that creates zero moment, the moment flow variables on the two sides will have different values that do not sum to zero. Contributions to each side must be assigned separately in the model code.
The 2-D crab-leg spring model may be reused to form model definitions of other topologies such as the crab-leg platforms in Figure 3(b) and (c). The crab-leg model can be further decomposed into a structured model from two interconnected atomic beam elements. The composable beam model provides the maximum design reuse for springs. No matter what level in the hierarchy is modeled behaviorally, proper port connection characteristics between module instances must be enforced, as discussed next.

V. Interoperability

Port interoperability at the functional and atomic levels is essential for general composition of devices. A meaningful interconnection between ports must meet three basic requirements. First, the disciplines must match, as is required of all port connections in AHDLs. Second, the physical directions

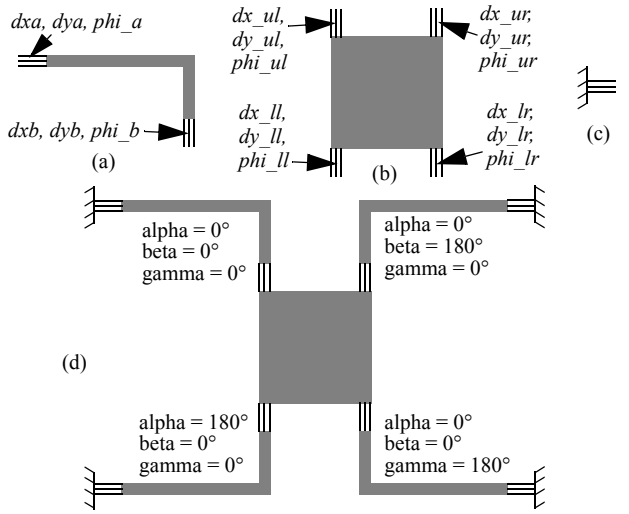

Figure 4: Example of MEMS composition. (a) crableg_spring2D symbol. (b) plate_2D symbol. (c) anchor (ground) symbol. (d) Schematic of a crab leg microstructure. Alpha, beta, and gamma are Euler angles to specify component oreientation. The crableb_spring_2D symbols are flipped, corresponding to the angle parameters, to make a more readable schematic.

associated with each connected port must match. That is, the global reference frames of the models must match their intended orientation when connected. Third, the models must adopt a uniform convention for associated reference directions for potential and flux of each port. This attention to interoperability is perhaps the most critical, and sometimes the most frustrating, issue in MEMS behavioral modeling, assuming the model physics is already understood.

A. Model Reference Frame

Consider a structured 2-D model for the crab-leg platform in Figure 3(c). Four crab-leg spring models are to be connected to the central plate, however, the orientation of each adjacent

```
module crableg_spring2D (dxa, dya, phi_za, dxb, dyb, phi_zb);
inout dxa, dya, phi_za, dxb, dyb, phi_zb;
kinematic_translational dxa, dya, dxb, dyb;
kinematic_rotational phi_za, phi_zb;
  ...
// Layout rotation angles of component
parameter real alpha = 0.;
parameter real beta = 0.;
parameter real gamma = 0.;
parameter real La = 1e-5;
parameter real Lb = 1e-4;
analog begin
  ...
cos_a = cos(alpha*`M_PI/180.);
cos_b = cos(beta*`M_PI/180.);
cos_g = cos(gamma*`M_PI/180.);
  ...
// Transform displacements in chip frame to component local frame
dx_l = cos_b*cos_g*(Disp(dxb)-Disp(dxa)) +
       cos_b*sin_g*(Disp(dyb)-Disp(dya));
dy_l = -cos_a*sin_g*(Disp(dxb)-Disp(dxa)) +
       cos_a*cos_g*(Disp(dyb)-Disp(dya));
dphi = Phi(phi_zb) - Phi(phi_za);
//Calculate spring forces and moments
Fkx_l = 3/(La+Lb)*
       ((4*La+Lb)/La^3*dx_l-3/La/Lb*dy_l-(2*La+Lb)/La^2*dphi);
Fky_l = 3/(La+Lb)*(-3/La/Lb*dx_l+(La+4*Lb)Lb^3*dy_l+3/b*dphi);
Mkz_b = 1/(La+Lb)*
       (-3*(2*La+Lb)/La^2*dx_l+3/Lb*dy_l+(4*La+3*Lb)/La*dphi);
  ...
Fx_bl = Fmx_bl+Fbx_bl+Fkx_l; // Fm are inertial forces
Fy_bl = Fmy_bl+Fby_bl+Fky_l; // Fb are damping forces
Fx_al = Fmx_al+Fbx_al-Fkx_l;
Fy_al = Fmy_al+Fby_al-Fky_l;
// Transform forces to chip frame of reference
F(dxb) <+ cos_b*cos_g*Fx_bl - cos_a*sin_g*Fy_bl;
F(dyb) <+ cos_b*sin_g*Fx_bl + cos_a*cos_g*Fy_bl;
F(dxa) <+ cos_b*cos_g*Fx_al - cos_a*sin_g*Fy_al;
F(dya) <+ cos_b*sin_g*Fx_al + cos_a*cos_g*Fy_al;
Tau(phi_zb) <+ Mm_b + Mb_b + Mkz_b;
Tau(phi_za) <+ Mm_a + Mb_a - Mkz_b - La*Fkx_l - Lb*Fky_l;
end
endmodule
```

Listing 3: Partial code for a linear 2-D behavioral model of the crab-leg spring.

spring instance changes by 90°. How one correctly connects the mechanical ports will depend on the underlying reference frame in the model code. The two logical choices for model reference frame are 1) a local frame of reference or 2) a "chip" frame of reference.

The local frame refers to a coordinate system aligned with the internal geometric features of the component, regardless of layout orientation. For example, the 2-D crab-leg spring model may have its local x axis oriented along the beam of length L_b and the y axis oriented along the beam of length L_a as defined in Figure 3(a). For the platform in Figure 3(c), the d_x, d_y and ϕ_z ports will be connected directly together when connecting the top and bottom spring module instances to the plate module. However, the d_x and d_y ports will be swapped when connecting the left and right spring module instances. With external ports referenced to the local frame, the burden of knowing when to swap connections is placed upon the user during schematic generation, and is thus prone to error. Furthermore, the simple swapping of ports does not work if the force-displacement behavior of the spring is different in its positive and negative directions (e.g., from nonlinearities under large deflection).

A better solution is to create models that use a chip frame of reference, named after the assumed rigid chip upon which the microstructures are fabricated. All modules using the chip frame as their coordinate system have a consistent definition for the direction of their mechanical ports. Therefore, the connections between module instances will always be d_x to d_x, d_y to d_y and ϕ_z to ϕ_z, eliminating a source of confusion for the user. The internal physical equations for the model behavior are still written in a local frame of reference. However, any static rotation of the element layout, such as the 90° rotation of the crab-leg spring instance, must now be accomplished internally within the model.

Layout orientation around the chip frame's x, y, and z axes may be expressed as Euler angle parameters (α, β, γ). In MEMS created in the plane of the substrate, the first two Euler angles, α and β, equal 0° or 180° to flip the component. Using this implementation, the layout position and Euler angle orientation for each module instance are specified by the user as fixed parameter values, and are not dynamically changing during simulation. The rotation matrices that relate the local frame variables internal to the element to the chip frame port signals have static values.

B. Associated Reference Directions

The Verilog-AMS LRM indicates that the HDL uses associated reference directions such that a positive flow enters a branch through the port marked with the plus sign and exits the branch through the port marked with the minus sign. Mechanical systems require additional interpretation, to map the sign of a potential or flow in a schematic to the coordinate direction of the 2-D or 3-D mechanical system. Consistency of the interpretation of associated reference directions across component libraries is critical for interoperability.

In our group's work, translational and rotational displacements, which are across variables, are interpreted as follows:

- positive valued displacements are in the positive axial direction
- positive valued angular displacements are counterclockwise around the axis

Force and moments, which are through variables, are interpreted as follows:

- positive valued force flowing *into* a pin acts in the positive axial direction
- positive valued moment flowing *into* a pin acts counterclockwise around the axis

The across variable conventions are relatively straightforward. However, the flow conventions can take some time to master. Some physical interpretations of several examples are given in Figure 5. A beam in tension has positive flow (force) going from right to left, while a beam in compression has flow going from left to right. Flow going into both left and right ports results in a net positive external force acting on the beam causing it to accelerate to the right. A flow through the moment ports from left to right, with no transla-

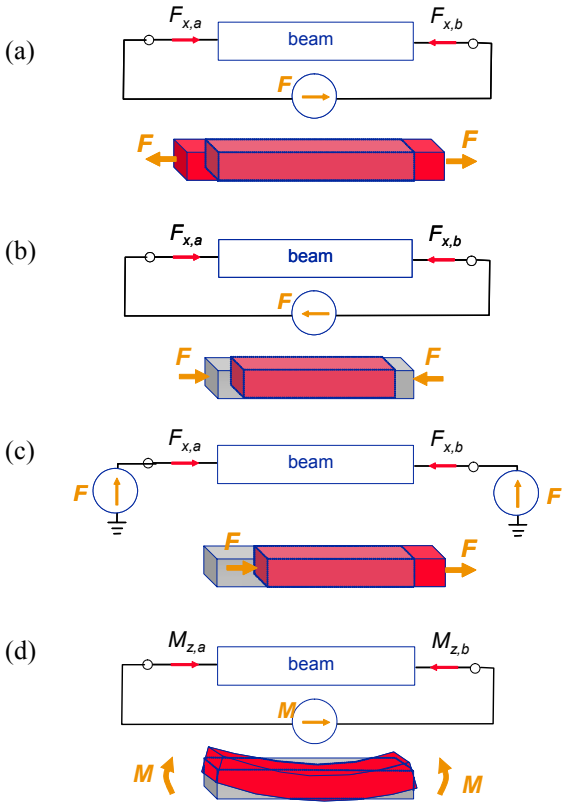

Figure 5: Examples of mechanical nodal conventions. F and M are positive valued. (a) Beam in tension, $F_{x,a} = -F_{x,b} = -F$. (b) Beam in compression. $F_{x,a} = -F_{x,b} = F$. (c) Beam accelerating in x. $F_{x,a} = F_{x,b} = F$. (d) Moment bending beam with positive curvature in y. $M_{z,a} = -M_{z,b} = -M$.

tional forces applied, causes the beam to bend up, while a flow from right to left causes the beam to bend down.

VI. Extension for MEMS

Each symbol pin in a schematic is normally associated with a single potential/flow port. A pin can be assigned to an analog array, used to group potential/flow ports of the same discipline, but there is currently no syntax for grouping ports of different disciplines in a single pin.

In MEMS, a single physical connection point has ports for its translational, rotational, electrical and thermal DOF. The location of the MEMS symbol and pins on the schematic can (and usually do) contain important contextual information for the user. Grouping of all the DOF of a single connection point into a single pin would simplify schematic construction by reducing pin count. It also would eliminate the multiple inout variables in the model, which currently creates code that can be difficult to read. It also reduces the possibility of incorrectly connecting pins corresponding to different DOF. For example, the pin corresponding to *x* displacement on one element could be inadvertently connected to *y* displacement on another element without generating any warning to the user.

The desire for grouping port signals along physical connection points motivates the ability in AHDLs to define multi-dimensional, composite, disciplines. One possible proposed extension of Verilog-AMS for MEMS is shown in Listing 4. In the example, a *mems2D* discipline is defined as a grouping of port signals declared by *kinematic_translational* and *kinematic_rotational* disciplines. With this multi-dimensional discipline, the signals associated with a particular physical connection point may be declared in a single variable. In Listing 4, new disciplines *dx*, *dy* and *phi* are derived from existing base disciplines, where the syntax extension is borrowed from that of derived natures. Although it is illegal in Verilog-AMS to change the access function for derived natures, this is done in the proposed code to create unique access functions for each DOF. The resulting model code is compact and simple to understand.

```
multidiscipline mems2D
  discipline dx : kinematic_translational
        potential.access = "Dx";
        flow.access = "Fx";
  enddiscipline
  discipline dy : kinematic_translational
        potential.access = "Dy";
        flow.access = "Fy";
  enddiscipline
  discipline phi : kinematic_rotational
  enddiscipline
endmultidiscipline

// Example declaration and use
inout a, b;
mems2D a, b;
        ...
real kxx, kyy, ktaa, ktab, ktbb, kxy, kxt, kyt;
analog begin
        ...
  Fx(a, b) <+ kxx*Dx(a,b) + kxy*Dy(a,b) + kxt*Phi(a,b);
  Fy(a, b) <+ kxy*Dx(a,b) + kyy*Dy(a,b) + kyt*Phi(a,b);
  Tau(a) <+ kxt*Dx(a,b) + kyt*Dy(a,b) + ktaa*Phi(a) + ktab*Phi(b);
  Tau(b) <+ kxt*Dx(a,b) + kyt*Dy(a,b) + ktab*Phi(a) + ktbb*Phi(b);
end
```

Listing 4: Proposed Verilog-AMS extension that allows composite disciplines to support MEMS modeling.

If such a structured discipline definition were possible, it is likely that custom disciplines for MEMS would emerge that would group various DOF of interest. For example, a *mems3D* discipline would include *z* translation and ϕ_x and ϕ_y rotation. Other disciplines may group the 2D or 3D kinematics with electrical and thermal signals that are associated with the same connection point.

VII. Conclusion

MEMS behavioral modeling has advanced greatly with the maturation of AHDLs, however the MEMS community has yet to accept AHDL simulation as the preferred starting point for MEMS design. MEMS 2-D and 3-D mechanical models in AHDLs are proven to be as accurate as their finite-lumped-element counterparts. There are still research issues in forming composable atomic-level electrostatic models, however sufficiently accurate models do exist for many design spaces of importance to MEMS.

A open standard cell library of 2-D and 3-D MEMS models, particularly at the atomic level in the hierarchy, will help promote advantages of behavioral modeling to the MEMS designers and provide templates upon which to extend the models or to build custom models. Standard conventions of disciplines and associated reference directions will help promote interoperability across the entire MEMS community.

VIII. References

[1] Coventor Architect: http://www.coventor.com/coventorware/architect/

[2] G. Lorenz, A. Morris, I. Lakkis, "A top-down design flow for MOEMS," Proc. of the SPIE - 4408, Design, Test, Integration, and Packaging of MEMS/MOEMS, April 25-27, 2001, Cannes, France, pp. 126-37.

[3] J.V. Clark, A. Agogino, et al., "Addressing the Need for Complex MEMS Design," Proc. MEMS '02, Jan. 20-24, 2002, Las Vegas, NV, USA, pp. 204-209.

[4] Q. Jing, *Modeling and Simulation for Design of Suspended MEMS*, Ph.D thesis, Dept. of ECE, Carnegie Mellon University, 2003.

[5] G. K. Fedder and Q. Jing, "A hierarchical circuit-level design methodology for microelectromechanical systems," *IEEE Trans. on Circuits and Systems-II*, vol. 46, no. 10, Oct. 1999, pp. 1309-1315.