Energy-Based Characterization of Microelectromechanical Systems (MEMS) and Component Modeling Using VHDL-AMS

A. Dewey[1], H. Dussault and J. Hanna[2], E. Christen[3], G. Fedder[4], B. Romanowicz[5], and M. Maher[6]

[1] Duke University, [2] Air Force Research Laboratory,
[3] Analogy Inc., [4] Carnegie Mellon University,
[5] Microcosm Technologies, Inc., and [6] Tanner Research

ABSTRACT

This paper presents an approach to defining and characterizing component models that promotes the establishment of reusable intellectual property (IP) for microelectromechanical systems (MEMS), in particular, and composite microsystems, in general. System dynamical behavior is mathematically characterized as coupled sets of ordinary algebraic and differential equations (DAEs), using the basic laws of conservation of energy governing electrical and mechanical networks combined with branch constitutive relations (BCRs) governing network elements. The ordinary algebraic and differential equations are described in the behavioral analog hardware description language (AHDL) VHDL-AMS, using a common modeling methodology and associated set of constructs. An example composite system model is given of an electromechanical loudspeaker.

Keywords: MEMS, component modeling, AHDL, VHDL-AMS, composite microsystems

INTRODUCTION

Intellectual property involves codifying design knowledge in a form that can be exchanged and reused and, as such, becomes a tradeable commodity. Intellectual property captures corporate design expertise and experience in a tangible form that can be leveraged in conceptualizing and synthesizing new designs; the form is component models. Component models describe the form and function of designs and the descriptions are typically written in a computer language to facilitate electronic exchange and processing via computers and design automation applications.

Developing component models for microelectromechanical systems (MEMS) presents several new challenges concerning issues of dynamical behavior characterization and complexity. The behavior of the overall system is not the simple concatenation of separate mechanical and electrical behaviors, but the simultaneous combination of mechanical and electrical behaviors. New component modeling, analysis, and design techniques are required to address both mechanics and electronics.

This paper first discusses the use of the mathematics of ordinary differential and algebraic equations to characterize both electrical and mechanical network dynamical behavior. Then, a methodology for codifying MEMS dynamical behavior using the hardware description language VHDL-AMS and a candidate set of common constructs and guidelines is presented. Finally, conclusions and future work are discussed.

MEMS DYNAMICAL BEHAVIOR CHARACTERIZATION

In developing dynamical behavioral characterizations for MEMS intellectual property, Figure 1 illustrates the need to extend fabrication and device-level modeling and analyses to a higher level of abstraction - referred to as a *circuit* level of abstraction.

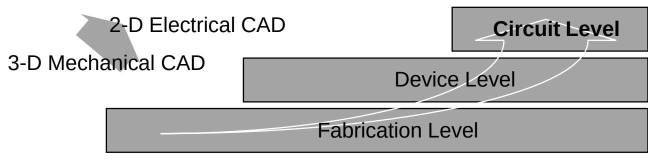

Figure 1: Moving Microelectromechanical System Dynamical Behavior to Higher Levels of Abstraction

Moving to higher levels of abstraction aligns MEMS component modeling with very large scale integration (VLSI) component modeling, thereby leveraging similar research and development, such as circuit simulation. A circuit level of abstraction of a microelectromechanical system abstracts three-dimensional electrical and mechanical effects into a two-dimensional network of lumped parameter elements governed by a system of ordinary differential and algebraic equations (DAEs). The DAEs provide a common characterization strategy that spans the multiple energy domains, allowing the simultaneous consideration of electrical and mechanical effects in determining overall system behavior as a function of time. DAEs capture the interplay between the effects of electromagnetic forces causing a change in the orientation and position of mechanical structures and conversely, mechanical structure movement causing a change in electromagnetic forces.

Microelectromechanical differential and algebraic equations are generated within a systematic framework recognizing linear independence and energy conservation because electrical networks and mechanical networks both obey basic laws of physics relating to the conservation of energy. Circuit network topology defines the global structure of the microelectromechanical equations. The behavior of circuit network elements are defined by branch constitutive relations (BCRs) that describe relationships between the applied potential gradient across an element and the resulting time derivative of state through an element.

As an example, consider the branch constitutive relations defining the relation between an excitation audio electrical signal and the movement of a speaker cone of a loudspeaker diagrammed in Figure 2.

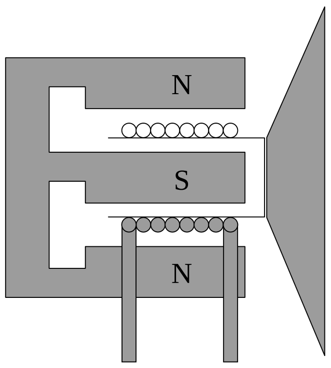

Figure 2: Electromechanical Loudspeaker

The excitation audio current passing through the magnetic field of the permanent magnet produces a force given by Equation 1

$$\boldsymbol{F} = \int \boldsymbol{I}\, \boldsymbol{dl} \times \boldsymbol{B} \quad (1)$$

and this force produces movement of the speaker cone, modeled as a simple second order harmonic oscillator with stiffness and dissipative damping. However, the movement of the speaker coil within the magnetic field of the permanent magnet, in turn, generates a voltage according to Faraday's law of magnetic induction given by Equation 2.

$$\boldsymbol{V} = \int \boldsymbol{\nu} \times \boldsymbol{B} \cdot \boldsymbol{dl} \quad (2)$$

The induced back electromotive force, along with the driving electrical network determines the excitation audio current. Hence, the coupled set of differential and algebraic equations given by Equation 1 and 2 define the branch constitutive relations for the electromechanical properties of the loudspeaker.

VHDL-AMS COMPOSITE SYSTEM COMPONENT MODELING

To use the system dynamical characterization presented in the previous section, the associated differential and algebraic equations must be cast into a computer-compatible form that can exchanged and processed; VHDL-AMS provides such a form. The VHSIC (Very High Speed Integrated Circuits) Hardware Description Language - VHDL - was originally designed to describe the structure and behavior of discrete time systems, in general, and digital systems, in particular. Recently, extensions to VHDL have been defined to enable descriptions of continuous time systems [3]. The combination of discrete and continuous time language constructs are collectively referred to as VHDL-AMS, where the suffix AMS stands for analog and mixed signal. To avoid confusion whether "mixed-signal" means coupled time-based or coupled energy-based models, we will refer to component models of coupled time behavior as "mixed system models" or "mixed-signal" and component models of coupled energy behavior as "composite system models" or "composite-signal" models.

VHDL-AMS provides constructs for defining sets of simultaneous ordinary differential and algebraic equations (DAEs). Ordinary differential and algebraic equations are described in a denotational style using "simultaneous statements" that define conditions or relations that must always hold over all time. Where VHDL uses an imperative dynamical model of loosely-coupled concurrent processes communicating by signals, VHDL-AMS uses a declarative dynamical model of tightly-coupled simultaneous relations influencing each other's solution by linked unknowns.

Unknowns are continuous, analytic functions of time, determined by repeatedly invoking an "analog solver" to solve the equations over a series of intervals denoting a period of time. The unknowns are called *quantities* and constitute a new class of objects in VHDL. Quantities are like signals in that they denote a waveform or a time series of values. However, quantities are very different from variables or signals in that they do not participate in assignment statements; quantities take their values as a result of solving the set of simultaneous ordinary differential and algebraic equations. This update mechanism is unique to quantities and is the principal reason they constitute a distinct object class.

Various operations associated with a quantity that yield related quantities, such as its derivative and integral, are defined in VHDL-AMS as predefined attributes. For example, the time derivative of a quantity representing displacement, named `X`, is denoted by `X'DOT`, where `DOT` is a predefined attribute for differentiation. The time derivative of the quantity X is automatically computed based on the sequence of values of X computed by the analog solver at various analog solution points (ASPs).

To explain these new language constructs, Figure 3 illustrates the simple branch constitutive relations defining the electromechanical loudspeaker characterized in the previous section.

```
use IEEE.MATH_REAL.all;
use IEEE.ELECTRICAL_SYSTEMS.all;
use IEEE.MECHANICAL_SYSTEMS.all;
entity LOUDSPEAKER is
 generic (
  -- mechanical properties of speaker
  constant SMASS    : MASS;
  constant SSPRING  : STIFFNESS;
  constant SDAMPING : DAMPING;

  -- electromagnetic properties of speaker
  constant NCOILS, RADIUS : REAL;
  constant MAG_FIELD : FLUX_DENSITY);
 port (
  terminal NODE1, NODE2 : ELECTRICAL);
end entity LOUDSPEAKER;

architecture ODE of LOUDSPEAKER is
  quantity V across I through NODE1 to NODE2;
  quantity X : REAL;
  constant COIL_EFFECT : REAL :=
           MATH_TWOPI*RADIUS*NCOILS*MAG_FIELD;
begin
  -- induced voltage
  V == -COIL_EFFECT*X'DOT;

  -- induced motion
  assert COIL_EFFECT /= 0.0;
  I == (1/COIL_EFFECT)*(SMASS*X'DOT'DOT
              + SDAMPING*X'DOT + SSPRING*X);
end architecture ODE;
```

Figure 3: VHDL-AMS Model of Basic ElectroMechanical Loudspeaker

The ordinary differential and algebraic equations governing electromechanical systems possess a global structure reflective of the fact that electrical and mechanical networks denote physical systems that obey laws of conservation of energy. A set of equations exhibiting such a global structure is called a *conservative* set of DAEs. To facilitate generating conservative ordinary differential and algebraic equations, VHDL-AMS provides a set of new language constructs, involving *terminals*, *natures*, and *branch quantities.*

The branch constitutive relations are defined using branch quantities and simultaneous statements. Branch quantities are quantities that possess additional semantics to reflect the additional constraints of an associated set of governing conservative simultaneous ordinary differential and algebraic equations. There are two kinds of branch quantities: *across* and *through.*

By definition, an across physical concept obeys a conservation of energy law whereby the summation of across quantities around any closed path (loop) equals zero.

$$\sum_{\text{CLOSED PATH}} \text{ACROSS QUANTITIES} = 0$$

Voltage differentials in electrical systems, displacement differentials in mechanical systems, and pressure differentials in fluidic systems are all examples of physical concepts that obey the basic conservation of energy law of summing to zero around a given closed path and, as such, are considered across quantities in VHDL-AMS. Across quantities are often gradient potentials within a given energy domain and thus, can be viewed as effort.

In a similar fashion, a through physical concept, by definition, obeys a conservation of energy law whereby the summation of through quantities at any node (point) equals zero.

$$\sum_{\text{NODE}} \text{THROUGH QUANTITIES} = 0$$

The movement of charge (current) in electrical systems, forces in mechanical systems, and heat flow in thermal systems are all examples of physical concepts that obey the basic conservation of energy law of summing to zero at a given node and, as such, are considered through quantities in VHDL-AMS. Through quantities are often the movement of state under an applied gradient potential field within a given energy domain and thus, can be viewed as flow.

The type of the branch quantities are indirectly defined via the definition of the *terminals.* Terminals provide connection points for conservative equations. Terminals hold no values; their only role is to facilitate forming conservative equation sets that, in turn, constrain the associated branch quantities. ELECTRICAL in Figure 3 is not a type; it is a *nature.*

Natures can be considered similar to types in that both types and natures define data templates or structures for objects. The nature declaration for ELECTRICAL, assumed to be supplied by the package

ELECTRICAL_SYSTEMS, is given in Figure 4. Natures can be viewed as abstract descriptions of an energy domains in that they define the fundamental effort/flow concepts of a physical discipline.

COMMON DECLARATIONS

To exchange microelectromechanical systems (MEMS) component models as intellectual property, an infrastructure is required establishing common terminology, declarations, styles, and practices [1], [2]. The infrastructure supports the common use of specialized applications of a base linguistics. This increased level of specificity is required to ensure MEMS models, or more generally composite system models, can interoperate. Candidate elements of a common VHDL-AMS modeling infrastructure for composite systems are listed below:

Common Package Architecture and Format,
Common Types and Subtypes,
Common Natures and Subnatures,
Common Physical Constants,
Common Simulation Controls, and
Common Design Unit Formats.

Package architecture refers to how the physical concepts and principles of the domain of composite systems are mapped against a set of VHDL-AMS packages. The proposed VHDL-AMS package architecture uses a hierarchical structure aligned with energy domains. A base package contains declarations common across energy domains and is used by the derived packages representing individual energy domains and disciplines. The proposed VHDL-AMS package architecture provides separate vehicles - foci - for each major engineering discipline to develop the data representations and operations relevant and pertinent to their respective domain.

In considering a common typing/subtyping structure for modeling composite systems, it is important to note that VHDL-AMS quantities are restricted to be of a floating point type so they may represent analytic functions of time. This restriction implies that across types and through types defined via nature declarations must also be floating point types.

Types for across and through branch quantities can be used to declare a common set of natures. It is important to point out that a given energy domain may not necessarily be characterized by a single nature declaration, as the definitions of across and through physical phenomena discussed in the previous section may be satisfied in multiple ways. Mechanical systems offer a simple example. In addition to displacement differentials, the time derivative of displacement also has physical significance in mechanics - namely velocity. Sample nature declarations are given in Figure 4 for the electrical and mechanical energy domains.

```
-- ELECTRICAL_SYSTEMS
subtype VOLTAGE is REAL;
subtype CURRENT is REAL;
nature ELECTRICAL is VOLTAGE across
 CURRENT through ELECTRICAL_REF reference;

-- MECHANICAL_SYSTEMS
subtype DISPLACEMENT is REAL;
subtype FORCE is REAL;
nature TRANSLATIONAL is DISPLACEMENT across
 FORCE through TRANSLATIONAL_REF reference;
```

Figure 4: VHDL-AMS Nature Declarations for Composite Systems

CONCLUSIONS

This paper presents an approach for MEMS Intellectual Property (IP) based on continuous time system dynamical characterization and behavioral analog hardware description language component modeling. Continuous time system dynamical characterization abstracts the complexities of electrical devices and mechanical structures into networks characterized by a set of conservative ordinary differential and algebraic equations. Network elements are describe by branch constitutive equations. Behavioral analog hardware description language component modeling captures the continuous time system dynamical characterization in a computer language - VHDL-AMS. Examples of common VHDL-AMS declarations are presented.

REFERENCES

[1] E. Christen and K. Bakalar, "Library Development Using the VHDL 1076.1 Language," *Proceedings of the Forum on Design Languages*, Lausanne, Switzerland, 1998.

[2] A. Dewey, et. al., *VHDL-AMS Modeling Considerations and Styles for Composite Systems*, Defense Advance Research Projects Agency (DARPA), Version 1.0, 1998.

[3] A. Vachoux and E. Moser, *VHDL-AMS as a Discrete and Continuous Time Modeling and Simulation Language*, 1995.