Chapter 9
An Extension to VHDL-AMS for AMS Systems with Partial Differential Equations

Leran Wang, Chenxu Zhao and Tom J. Kazmierski

Abstract  This paper proposes VHDL-AMS syntax extensions that enable descriptions of AMS systems with partial differential equations. We named the extended language VHDL-AMSP. An important specific need for such extensions arises from the well known MEMS modelling difficulties where complex digital and analogue electronics interfaces with distributed mechanical systems. The new syntax allows descriptions of new VHDL-AMS objects, such as partial quantities, spatial coordinates and boundary conditions. Pending the development of a new standard, a suitable pre-processor has been developed to convert VHDL-AMSP into the existing VHDL-AMS 1076.1 standard automatically. The pre-processor allows development of models with partial differential equations using currently available simulators. As an example, a VHDL-AMSP description for the sensing element of a MEMS accelerometer is presented, converted to VHDL-AMS 1076.1 and simulated in SystemVision.

Keywords  Hardware description language, VHDL-AMS, mixed-technology modelling, partial differential equations, MEMS

9.1 Introduction

VHDL-AMS is a hardware description language designed to support modelling at various abstraction levels in mixed, electrical and non-electrical physical domains as well as mixed, digital and analogue components [1]. These features make it straightforward for VHDL-AMS to be used as the modelling language in MEMS design. Since MEMS systems are combinations of subsystems from both the electrical and mechanical domains, the field of MEMS design is interdisciplinary in nature. Several VHDL-AMS based MEMS models have already been reported in literature, such as a yaw rate sensor [2] and a vibration sensor array [3].

School of Electronics and Computer Science, University of Southampton, UK
Email: {lw04r, cz05r, tjk}@ecs.soton.ac.uk

E. Villar (ed.) Embedded Systems Specification and Design Languages, 123
© Springer Science+Business Media B.V. 2008
Although VHDL-AMS is a very powerful and flexible mixed physical domain modelling tool, it faces a challenge in MEMS related applications. The current VHDL-AMS (IEEE 1076.1) can only describe the continuous parts of a system by using differential and algebraic equations (DAEs). Support for partial differential equations (PDEs) was intentionally left out in the development of VHDL-AMS standard due to the complexity [4]. This limits accurate modelling of system blocks that include distributed physical effects [5]. However, simulation of single-domain characteristics of micro devices is usually performed by solving PDEs with geometry-related boundary conditions [6]. Such blocks are currently modelled in VHDL-AMS mainly by reduced-order models (ROMs) [2, 3]. Because of the size of a MEMS device, distributed effects are not negligible and may even play vital roles, for which reduced-order MEMS models are often not accurate enough. Thus an implementation of PDEs in VHDL-AMS is in demand. Suggestions have been made to extend other AMS-HDLs, such as Modelica [7] and Verilog-AMS [8], to add PDE support.

Some attempts have already been made to implement PDEs within the existing limits of VHDL-AMS. A transmission line example [5] and a system with electro-thermal coupling [9] are modelled using VHDL-AMS 1076.1. The way is to discretize the equations with respect to spatial variables and leave the time derivatives to be handled by VHDL-AMS [5]. The problem with this approach is that the discretization is done manually. When some modifications are made to the system, a series of equations have to be rewritten which makes the modelling very inefficient. New language extensions for PDE support have also been raised [5, 9] but currently no simulator can handle the new operators.

The work presented in this paper implements PDEs in VHDL-AMS in such a way that pending the development of a corresponding standard, PDEs can be written directly but no new simulators are needed. Necessary language constructs have been adopted from previous work [5, 9] and some improvements have been made. A translation pre-processor has been developed to convert the extended language (VHDL-AMSP) into VHDL-AMS 1076.1 automatically so that models with PDEs can be simulated using currently available simulators. Using this new method VHDL-AMS models that describe systems with distributed physical effects can now be built and simulated more efficiently.

The proposed methodology is expected to have particular advantages in mixed mechanical-electrical systems with tight control feedback loops, of which the MEMS block is an integral part. For example, the work presented in a recent paper [10] intends to develop new and innovative control and interface systems, technologies and circuits for MEMS physical sensors. The primary methodology is based on the incorporation of micro-mechanical sensing elements (e.g. for accelerometers and gyroscopes) in high-order $\Sigma\Delta$ modulator (SDM) loops. The loop filter consists of mechanical and electronic integrators; the former is constituted by the micromachined sensing element which is, to a first order approximation, a second order transfer function. The tools currently used for simulating such a complex and highly coupled system are primarily system level tools, such as Matlab/Simulink. The lumped parameter model of the sensing element captures only the first mechanical mode.
However, when designing higher-order electro-mechanical SDM loops, higher order mechanical modes may well be of considerable significance for the stability and performance of the control loop. Consequently, having a distributed mechanical model using partial differential equations would be a significant breakthrough for the design of such devices. To demonstrate the efficiency of our approach, the sensing element of such a MEMS accelerometer in SDM loop has been modelled in VHDL-AMSP, translated to VHDL-AMS 1076.1 and simulated. Simulation results show that high-order behaviour of the cantilever beam has been captured, which is not possible in conventional methodologies.

### 9.2 VHDL-AMS Extensions for PDE Support

The extensions outlined below support equations that may contain high-order partial derivatives describing systems in a multidimensional space.

#### 9.2.1 Partial Quantity

With the keyword `partial`, a partial quantity is defined as a physical variable which has a continuous value not only over a period of time but also over a hypercube in a multidimensional space. It is declared as:

```
partial quantity q : real;
```

The corresponding BNF (Backus-Naur Form) notation is:

```
partial_quantity_declaration ::= partial quantity identifier_list : subtype_indication;
```

Partial quantities may act as interfaces between entities as well as appear in architecture bodies.

#### 9.2.2 Spatial Coordinate

With the keyword `coordinate`, spatial coordinate is declared over which a partial quantity is distributed. Multiple coordinate declarations will form a hypercube in space. The declaration can define a range in space and the discretization step size.
The range is obligatory as it defines the hypercube, but the step size is optional. It is up to the designer to decide whether to use default step size or to give a fixed value. The following is an example of a spatial coordinate declaration:

```plaintext
coordinate x : real range 0.0 to 10.0 step 0.1;
```

Two new grammar productions have been added to the language BNF:

```plaintext
coordinate_declaration ::= coordinate identifier_list : subtype_indication;
step_size ::= step simple_expression
```

The existing `range` construct is extended by the new `step` construction as:

```plaintext
range ::= range_attribute_name [step_size]  
       | simple_expression direction simple_expression [step_size]
```

### 9.2.3 Partial Derivatives

As suggested in the papers by Nikitin et al. [5, 9], a new language attribute name is introduced as `$\dot{\text{q}}(x)$`. If $q$ is a partial quantity and $x$ is a coordinate, $q^{\prime}\dot{\text{q}}(x)$ represents the derivative of $q$ with respect to $x$. Unlike the example given in the paper [9] where a high-order derivative is represented by multiple ticks, e.g. $q^{\prime\prime\prime\prime}\dot{\text{q}}(x)$ for the second order, VHDL-AMSP uses the same notation as VHDL-AMS, namely $q^{\prime\prime\dot{\text{q}}}(x)$.

This kind of representation is in the spirit of the existing VHDL-AMS 1076.1 standard and $q^{\prime\prime\dot{\text{q}}}(x)$ as a whole is still a partial quantity. A partial quantity can also have a derivative with respect to time, using the attribute `$\dot{}$`, so items like $q^{\prime\dot{\text{q}}}(x)$ are valid. Multidimensional derivatives are supported, such as $q^{\prime\dot{\text{q}}}(x)\dot{\text{y}}(x)$ where $x$ and $y$ are two coordinates. Since there is no predefined attribute name and attribute designator in VHDL-AMS, this extension does not affect the language BNF.

### 9.2.4 Simultaneous Statement with Partial Derivatives

A simple example is:

```plaintext
q^{\prime\dot{\text{q}}}(x) == A * q^{\prime\dot{\text{q}}};
```
which represents \( \frac{\partial q}{\partial x} = A \frac{\partial q}{\partial t} \)

Partial differential equations can also appear in simultaneous if or case statements. High-order derivatives or derivatives of more than one spatial coordinate can also be described in a simultaneous statement.

### 9.2.5 Boundary Conditions

A boundary condition is defined as a special simultaneous statement as shown below. The expression after the keyword at specifies the spatial boundary where the conditions should apply. Conditions are written in the form of simultaneous statements. An example is:

```
boundary x at 0.0 is
begin
  q == 0.0;
  q' dot(x) == 0.0;
end BOUNDARY;
```

The corresponding production in the language BNF is:

```
simultaneous_boundary_statement ::= [boundary_label:]
boundary coordinate_name at simple_expression is begin
  simultaneous_statement {simultaneous_statement}
end boundary [boundary_label];
```

### 9.3 Translation to VHDL-AMS 1076.1

We have developed a translation pre-processor to automatically convert VHDL-AMSP models into VHDL-AMS 1076.1. The pre-processor can be used as a tentative measure to implement PDEs in VHDL-AMS pending the development of an appropriate standard. The translation pre-processor uses a modified version of a VHDL-AMS parser [11] where the modifications incorporate the new syntax into the parser and allow syntax analysis by recursive scanning of the parse tree. During the scanning, new language constructs can be identified and replaced by necessary VHDL-AMS 1076.1 constructs. How the new constructs are converted into existing constructs is demonstrated below, using the examples from Section 9.2.
In the declaration part of the model, a partial quantity is converted into a quantity vector by the same name. The vector size is determined by the coordinate’s range and step, i.e. \( \text{range/step} \). The coordinate won’t appear in the output file but a differential coefficient (\( dx \) in the example) will be declared as a constant, which has the value of the step size. The declaration part will therefore contain:

```plaintext
quantity q : real_vector (0 to 100);
constant dx : real := 0.1
```

In the architecture part, a PDE will be replaced by a series of DAEs. Finite difference approach [12] is used as the discretization method. Note that the discretization only applies to the middle part of a hypercube space while the borders will be described by boundary conditions. The PDE in Section 9.2.4 will be discretized as:

```plaintext
(q(2) - q(1)) / dx == A * q(1)'dot;
(q(3) - q(2)) / dx == A * q(2)'dot;
(q(4) - q(3)) / dx == A * q(3)'dot;
...
```

The boundary statements in Section 9.2.5 are translated into simple simultaneous statements:

```plaintext
q(0) == 0.0;
(q(1) - q(0)) / dx == 0.0;
```

These DAEs are solvable by a VHDL-AMS 1076.1 simulator.

### 9.4 MEMS Accelerometer in a Sigma Delta Control Loop

Figure 9.1 shows the block diagram of a MEMS accelerometer in fifth-order SDM control loop [10]. Like most conventional modelling approaches, the micro-mechanical sensing element is modelled as a second-order spring damping system:

\[
M \ddot{z}(t) + C \dot{z}(t) + K z(t) = F(t)
\]  

(9.1)

where \( M \) is the proof mass, \( C \) and \( K \) are effective damping and spring factor respectively, \( z(t) \) is the relative displacement and \( F(t) \) is the feedback force. The frequency response of the lumped model is shown in Fig. 9.2.
The proof-mass displacement is converted to electronic signal by differential capacitive position sensing. The electronic signal is then passed through a third-order low-pass filter, which is implemented with distributed feedback structure. The filtered signal is digitized by a 1-bit quantizer and the output is the digital signal. The electrostatic feedback force is generated by a DAC. Such a SDM control loop has the advantages of increased dynamic range, linearity and bandwidth [10] thus it has attracted great research interests.

In actual situation, the sensing element consists of a MEMS cantilever beam located between two plate electrodes (Fig. 9.3). Instead of moving as a lumped mass, the cantilever beam itself vibrates and has higher frequency modes. It has
been proved that higher-order resonant frequencies can affect the performance of an SDM loop [13]. However, as shown in Fig. 9.2, such behaviour cannot be captured by the conventional lumped model.

9.5 VHDL-AMSP Model of the Sensing Element

9.5.1 Model Description

Figure 9.3 shows the sensing element of an accelerometer in SDM control loops. The feedback force is acting on the base of the cantilever (non-collocated dynamics) [13] and the cantilever beam is only deformed by distributed electrostatic force. The governing equation of this model is:

\[
EI \frac{\partial^4 y(x,t)}{\partial x^4} + c_d \frac{\partial^2 y(x,t)}{\partial x^2 \partial t} + \rho S \frac{\partial^2 y(x,t)}{\partial t^2} = F_e(x,t)
\]  

(9.2)

where \(y(x,t)\) is the relative displacement at position \(x\) and time \(t\), \(E\) is the Young's modulus, \(I\) is the moment of inertia, \(c_d\) is the damping factor, \(\rho\) is the material's density, \(S\) is the cross sectional area and \(F_e(x,t)\) is the electrostatic force.

The boundary conditions at the clamped end and the free end are shown in Eqs. 9.3 and 9.4 respectively [14],

\[
y(0,t) = z(t)
\]

\[
\theta = \frac{\partial y(0,t)}{\partial x} = 0
\]

(9.3)
9 An Extension to VHDL-AMS for AMS Systems

\[ M = -EI \frac{\partial^2 y(L,t)}{\partial x^2} = 0 \]

\[ Q = -EI \frac{\partial^3 y(L,t)}{\partial x^3} = 0 \]  

(9.4)

where \( \theta, M \) and \( Q \) denote the slope angle, the bending moment and the shear force respectively, \( L \) is the length of the beam.

The initial condition is simply:

\[ y(x,0) = 0 \]  

(9.5)

The electrostatic force \( F_e(x, t) \) is given by:

\[ F_e(x, t) = \frac{1}{2} \varepsilon A \left[ \frac{V_0^2}{(d_0 - y(x,t))^2} - \frac{V_0^2}{(d_0 + y(x,t))^2} \right] \]  

(9.6)

where \( \varepsilon \) is the permittivity of the gap, \( A \) is the area of the electrode, \( d_0 \) is the spacing between the beam and the electrode and \( V_0 \) is the amplitude of the applied AC voltage.

The distributed capacitance between the cantilever and the electrode is given by:

\[ C_{11} = \frac{\varepsilon A}{d_0 - y(x,t)}, \quad C_{12} = \frac{\varepsilon A}{d_0 + y(x,t)} \]  

(9.7)

The output voltage can be calculated as:

\[ V_{out}(t) = \frac{C_{11} - C_{12}}{C_{11} + C_{12}} V_0 \sin(\omega t) \]  

(9.8)

For small displacement cases, it can be assumed that \( y^2 \ll d_0^2 \). The above equation could be simplified as:

\[ V_{out}(t) = -\frac{\bar{y}(t)}{d_0} V_0 \sin(\omega t) \]  

(9.9)

where \( \bar{y}(t) \) is the average beam position.

9.5.2 VHDL-AMSP Code

The VHDL-AMSP model of the cantilever beam presented below provides an example of how the elements discussed in Section 9.2 are implemented. \( y \) is the partial quantity which represents the deflection of the beam and \( FE \) is also a partial quantity which represents the electrostatic force. \( x \) is the spatial coordinate.
Boundary conditions have been applied and typical values are used for the constants.

library IEEE;
use IEEE.ENERGY_SYSTEMS.all;
use IEEE.ELECTRICAL_SYSTEMS.all;
use IEEE.MECHANICAL_SYSTEMS.all;
use IEEE.MATH_REAL.all;
entity COMB_DRIVE is
  generic(E:real; --Young’s modulus
           I:real; --moment of inertia
           rou:real; --density
           L:real; --length of beam
           d0:real; --gap spacing
           K:STIFFNESS; --effective spring stiffness
           D:DAMPING; --effective damping
           S:real; --cross sectional area
           C:real; --cantilever damping
           A:real; --electrode area
           ep0:real; --permittivity
           M:MASS);
  port(terminal PROOF_MASS:TRANSLATIONAL);
end entity COMB_DRIVE;
architecture BCR of COMB_DRIVE is
  constant N:real:=5.0;
  partial quantity y:real;
  partial quantity FE:real;
  coordinate x:real range 0.0 to L step L/N;
  quantity z across F0 through PROOF_MASS to TRANSLATIONAL_REF;
begin
  M*z’DOT’DOT+D*z’DOT+K*z=F0;
  --movement of proof mass
  E*I*y’dot(x)’dot(x)’dot(x)’dot(x)+ROU*S*y’dot’dot+
  C*y’dot(x)’dot(x)’dot(x)’dot(x)’dot=FE;
  --dynamics of cantilever
  FE=0.5*ep0*A*(1.0/((d0-y)**2)-1.0/((d0+y)**2));
  --electrostatic force
  BOUNDARY x at 0.0 is
  begin
    y=z;
    y’dot(x)=0.0;
  end BOUNDARY;
--boundary condition at clamped end
BOUNDARY x at L is
begin
  y’dot(x)’dot(x)==0.0;
  y’dot(x)’dot(x)’dot(x)==0.0;
end BOUNDARY;
--boundary condition at free end
end architecture BCR;

9.5.3 Output from the Translation Pre-Processor –VHDL-AMS

In the output from the translator shown below, partial quantity $y$ and $FE$ each has been replaced by a quantity vector. The beam is discretized into five sections where the number of sections is calculated as range/step. The differential coefficient $dx$ represents the step size. From the PDE and the boundary conditions, two sets of six DAEs are created to describe the distributed behaviour of the beam. The comments in the code below were added manually for clarity.

library IEEE;
use IEEE.ENERGY_SYSTEMS.all;
use IEEE.ELECTRICAL_SYSTEMS.all;
use IEEE.MECHANICAL_SYSTEMS.all;
use IEEE.MATH_REAL.all;
entity COMB_DRIVE is
generic(…);
port(terminal PROOF_MASS:TRANSLATIONAL);
end entity COMB_DRIVE;
architecture BCR of COMB_DRIVE is
constant N:real:=5.0;
constant dx:real:=L/N;
quantity y:real vector(0 to 5):=(others=>0.0);
quantity FE:real vector(0 to 5):=(others=>0.0);
quantity z across F0 through PROOF_MASS to TRANS-
LATIONAL_REF;
begin
  M*z’DOT’DOT+D*z’DOT+K*z==F0;
  --movement of proof mass
  FE(0)==0.5*ep0*A*(1.0/((d0-y(0))**2)-1.0/((d0+y(0))**2));
--electrostatic force for clamped end
FE(1)==0.5*ep0*A*(1.0/((d0-y(1))**2) -1.0/((d0+y(1))**2));
--electrostatic force for section 1
FE(2)==0.5*ep0*A*(1.0/((d0-y(2))**2) -1.0/((d0+y(2))**2));
--electrostatic force for section 2
FE(3)==0.5*ep0*A*(1.0/((d0-y(3))**2) -1.0/((d0+y(3))**2));
--electrostatic force for section 3
FE(4)==0.5*ep0*A*(1.0/((d0-y(4))**2) -1.0/((d0+y(4))**2));
--electrostatic force for section 4
FE(5)==0.5*ep0*A*(1.0/((d0-y(5))**2) -1.0/((d0+y(5))**2));
--electrostatic force for section 5
y(0)==z;
--dynamics of clamped end
E*I*(y(3)-4.0*y(2)+6.0*y(1)-3.0*y(0))/dx**4 +ROU*S*y(1)'DOT'DOT+C*(y(3)'DOT-4.0*y(2)'DOT +6.0*y(1)'DOT-3.0*y(0)'DOT)/dx**4==FE(1);
--dynamics of section 1
E*I*(y(4)-4.0*y(3)+6.0*y(2)-4.0*y(1)+y(0))/dx**4 +ROU*S*y(2)'DOT'DOT+C*(y(4)'DOT-4.0*y(3)'DOT +6.0*y(2)'DOT-4.0*y(1)'DOT+y(0)'DOT)/dx**4==FE(2);
--dynamics of section 2
E*I*(-2.0*y(5)+5.0*y(4)-4.0*y(3)+y(2))/dx**4 +ROU*S*y(3)'DOT'DOT+C*(-2.0*y(5)'DOT+5.0*y(4)'DOT -4.0*y(3)'DOT+y(2)'DOT)/dx**4==FE(3);
--dynamics of section 3
E*I*(y(5)-2.0*y(4)+y(3))/dx**4+ROU*S*y(4)'DOT'DOT+C*(y(5)'DOT-2.0*y(4)'DOT+y(3)'DOT)/dx**4==FE(4);
--dynamics of section 4
E*I*(y(5)-2.0*y(4)+y(3))/dx**4+ROU*S*y(4)'DOT'DOT+C*(y(5)'DOT-2.0*y(4)'DOT+y(3)'DOT)/dx**4==FE(5);
--dynamics of section 5
end architecture BCR;
9.5.4 Simulation Results

The VHDL-AMS 1076.1 description generated by the translation pre-processor has been simulated by SystemVision from Mentor Graphics [15] and simulation results showing the frequency response of average beam position are presented in Fig. 9.4. It is clear that higher-order resonant modes have been captured.

9.6 Conclusion

This paper proposes extensions to efficiently implement general partial differential equations in VHDL-AMS. The current version of VHDL-AMS (IEEE 1076.1) can only support ordinary derivatives with respect to time and faces difficulties when applied to the modelling of distributed systems. In the proposed VHDL-AMSP language, new constructs are introduced to describe PDEs in a direct form. A translation pre-processor has been developed to convert VHDL-AMSP models into VHDL-AMS 1076.1 automatically, such that models with PDEs can be simulated using currently available simulators. The added PDE support enhances the ability
of VHDL-AMS to model MEMS systems where distributed behaviour is essential. The efficiency of this new approach has been investigated by VHDL-AMSP based modelling and simulation of the sensing element of a MEMS accelerometer in high-order SDM loop. Simulation results show that VHDL-AMSP model could describe the distributed behaviour of a system which is not possible in current VHDL-AMS 1076.1 language.

References