# VHDL-AMS Survey and Call for Participation Feedback

May 4, 2010

## *Projects Priorities*

| | Mixed Netlist | Freq. Domain Modeling | Support for PDEs | Table-driven Modeling | Vector/ matrix Operations | Unified Use of SPICE Models |
|---|---|---|---|---|---|---|
| Alin Constantin Mocanu, Infineon AG | 10 | 1 | 5 | 5 | 5 | 10 |
| Alan Mantooth, University of Arkansas | 6 | 9 | 5 | 10 | 8 | 7 |
| Rashed Rabaa, General Motors | 7 | 5 | 6 | 9 | 8 | 10 |
| Colin Marquardt, Zentrum Microelectronik Dresden AG | 9 | 6 | 6 | 2 | 2 | 2 |
| David Smith, Synopsys, Inc. | 8 | 5 | 3 | 10 | 7 | 5 |
| Zhichao Deng, Synopsys, Inc. | 3 | 9 | 10 | 7 | 10 | 8 |
| Gilles Depeyrot, Dolphin Integration | 7 | 9 | 6 | 5 | 8 | 9 |
| Lars Vosskaemper, Dolphin Integration | 1 | 9 | 10 | 6 | 5 | 8 |
| Gary L. Dare | 9 | 10 | 5 | 6 | 4 | 8 |
| Daniel Kho, Altera | 8 | 10 | 10 | 2 | 10 | 3 |
| Ernst Christen, Lynguent, Inc. | 9 | 6 | 3 | 10 | 6 | 6 |
| Lili Zhou, Orora | 7 | 9 | 8 | 10 | 10 | 5 |
| Ewald Hessel, Hella | 3 | 10 | 8 | 10 | 7 | 8 |
| Alain Vachoux, EPFL | 10 | 8 | 5 | 9 | 8 | 7 |
| Torsten Mähne, EPFL | 10 | 8 | 5 | 9 | 10 | 6 |
| Sameer Kher, Ansoft | 3 | 5 | 9 | 5 | 4 | 2 |
| Uwe Eichler, Fraunhofer | | | | 7 | | |
| Joachim Haase, Fraunhofer | 8 | 7 | 3 | 6 | 3 | 10 |
| Markus Pistauer, CISC Semi-conductor Design + Consulting | 8 | 10 | 7 | 10 | 9 | 1 |
| Erik Markert, Technical University Chemnitz | 3 | 8 | 10 | 5 | 3 | 1 |
| Michael Karner, Technical University Graz | 10 | 8 | 5 | 8 | 4 | 7 |
| Balado, Catalan Polytechnic University | 8 | 5 | 2 | 2 | 4 | 10 |
| Chris J Myers, University of Utah | 6 | 8 | 4 | 9 | 7 | 6 |
| Steve Grout | 8 | 10 | 10 | 8 | 7 | 10 |
| Marko Moch, C&S Group | 1 | 4 | 7 | 8 | 3 | 2 |
| Marie-Minerve Louerat, | 10 | 8 | 2 | 3 | 5 | 10 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Université Pierre et Marie Curie | | | | | | |
| François Pêcheux, Université Pierre et Marie Curie | 8 | 7 | 10 | 4 | 5 | 6 |
| In Hur, AZAPA Co., LTD. | 2 | 8 | 8 | 6 | 4 | 1 |
| Peter Jores, Robert Bosch GmbH | 6 | 7 | 6 | 9 | 8 | 9 |
| **Total** | **188** | **209** | **178** | **200** | **174** | **177** |

## *Conclusions*

### Priority of Optional Projects

The feedback indicates that frequency domain modeling table-driven modeling get high marks, while PDE support, uniform use of SPICE models and vector/matrix operations rank low. Mixed netlist support is in the middle.

In this revision emphasis should be given to the top 3 projects. The bottom 3 should only be pursued if there are enough resources.

### Additional Projects Proposed by Participants

A number of additional projects were proposed by participants in the survey. The proposals can be found on subsequent pages. Some projects are small enough that it should be possible to include them with relatively little effort in this revision:

- Definition of pole/zero and mixed root/coefficient forms of ‘LTF and ‘ZTF attributes
- Integration and periodic functions

The following projects should be investigated to determine whether they should be included in this revision of the language.

- Non-static coefficients for 'LTF, 'ZTF: physical meaning, algorithms
- Non-static ideal delay: physical meaning, algorithms
- Impure functions in characteristic expressions: cost of implementation
- Postponed for analog: restrictions

The following project should be delayed to a later revision as a scope should be established first, which will require extensive investigations:

- RF, Base band modeling
- Checking if equations are homogeneous
- Quantities, dimensional analysis

We do not have enough information at this time for the following proposed project:

- Sensitivity, Statistical Modeling (Note: SAE J2748 defines a package for statistical modeling)

## *Feedback Related to Projects, and Proposals for New Projects*

### Ewald Hessel

#### Frequency domain

From my perspective (and probably the perspective of AK23 and AK30) a language extension in AC mode is as important as the use of tables (e.g. behavioral modeling using measured values, insertion of standardized drive cycles, etc.). Mr. Haase has already proposed language extensions for AC mode. He also wrote a simple package for AK30 to support modeling using tabular data. The package supports reading data from ASCII files (and writing of simulation results to files).

**Follow-up**

Extensions to the AC model are important for us (Hella, AK30) because EMC simulation is gaining importance. Often the behavior of some blocks is only available as measured frequency domain data. To create models using this information there are only few choices: a direct use of the data using a table model, or an approximation of the data by complex functions or possible a substitute circuit. I believe that the corresponding functionality may already be available in many tools.

#### PDEs

I see the importance of PDEs predominantly for modeling of electrical wires and wire harnesses (e.g. vehicle wire systems)

#### Vector/Matrix

In my opinion the biggest problem with supporting vector/matrix operations is with the tools which sometimes have limited support for vectors. The workaround is often a transformation of the problem such that one-dimensional vectors can be used.

#### SPICE

For SPICE support AK30 has adopted the following solution:

1. VHDL-AMS entities were defined for the basic SPICE models (SPICE 3F5, excluding the B sources). There are also architectures that so far in general only implement level 1 functionality.
2. Tool vendors were asked to provide links to their internal SPICE models (entities and architectures) such that they could be used in VHDL-AMS context to get better simulation performance. There is a problem if such models, due to proprietary enhancements, have different parameters than the standard. In this case adaptations to the SPICE 3F5 standard by means of suitable architectures would be desirable. So far none of the tool vendors has provided such links. The currently used models work reliably - although often they are not very fast.
3. SPICE netlists are converted (so far manually) to VHDL-AMS netlists. From my perspective supporting the SPICE netlist format as part of VHDL-AMS does not make much sense.

**Follow-up**

From my perspective an acceptable solution would be to define entities for all basic SPICE models. Tool vendors could then “hide” the link to their version of the corresponding model using a suitable architecture. The effort for such an approach doesn’t seem outrageous.

### Participation

If possible with the current AK30 projects, AK30 would like to participate in extending the VHDL-AMS language (e.g. through creation or testing of packages). To this end a regular exchange of information would certainly be useful. One of our next projects will likely include modeling of a vehicle wire system (including EMC effects (AK23)). For this purpose extensions to the AC mode, support for table driven modeling and PDE support would be useful.

## Ernst Christen

### Periodicity and integration

One of the issues with periodic functions of time is that when their phase grows without limit over time. Determining the frequency by integrating the phase presents issues for the algorithms commonly used for the solution of the DAEs describing the model because the absolute tolerance becomes meaningless for large phase values. To address this issue, Verilog-AMS provides a special integration operator that is aware of the periodicity of the function. We should investigate whether a similar approach should be taken with VHDL-AMS.

### Non-static ideal delay

In system level models it is often desirable to describe a model with an ideal delay that varies as a function of some waveform. Adding this capability to VHDL-AMS would extend its usability for system level modeling. There will likely be some restrictions on the waveform describing the delay. To formulate these restrictions the algorithmic implications of supporting this functionality must be understood.

### Calls to impure functions in characteristic expressions

The VHDL-AMS LRM defines that all functions called when evaluating characteristic expressions must be cycle pure. This class includes all pure functions and impure functions that return the same value while finding an analog solution point. We should investigate whether the definition could be changed to allow any impure function to be called in a characteristic expression.

### Functionality similar to postponed process for analog solution

This functionality would enable a model writer to report quantity values after a solution has been found, which is not possible with the current language definition. This seems important when developing testbenches, but has other uses as well.

## Torsten Maehne

### Mixed netlist

In my eyes a very important topic, as it is required for various use-cases, e.g., top-level simulation of a system where various components shall be replaced by more detailled models, i.e., their interface is pin-compatibel (from the names), but the port type has to be refined. It is not desirable to be obliged to modify the top-level structural description in that case, but rather to define how the conversion shall happen at the well defined boundaries. ADVance MS proposes, e.g, some non-standard solutions in this case (especially VHDL <-> SPICE netlist coupling.)

## Frequency domain

I think there is a need to extend VHDL-AMS in this area to allow the description of non-linear behavior in the frequency domain.

#### Follow-up

With "nonlinear frequency domain support", I was thinking about support for harmonic balance to calculate the steady-state frequency response of an RF system including nonlinear components such as mixers, LNAs, and PAs. It goes in the same direction as my proposal to consider better support for baseband modeling in VHDL-AMS. Due to my work concerning the development of a model library for RF transceivers, I came across of quite some articles, where people tried to use either VHDL-AMS or Verilog-AMS to do baseband modeling. All had to fight with the limitations of current AMS-HDLs (and its support by the simulator) to basically terminals with a real-valued signal, which renders it difficult to create pin-compatible models, which could be substituted in the top-level netlist with the help of a configuration to do system simulations with varying accuracy--simulation performance trade-offs, i.e., supporting different scenarios:

- very fast transient simulation with baseband equivalent models of the RF components to test overall connectivity and functionality of the baseband part of the system.
- transient simulation with ideal behavioral models of the RF components
- transient simulation, where some RF components are described in more detail adding nonlinear effects
- transient simulation, where some critical RF components are even simulated on transistor level

I know this covers a large spectrum of abstraction, but it has to be covered during the design of an RF system. The authors of the baseband articles used all kinds of implementation tricks for their baseband equivalent models:

- Encoding of a complex value into a single real-value by scaling, truncating, and adding the real and imaginary components.
- Using the accross and through quantity of terminals independently in block diagram type models to encode the real and imaginary parts of a complex
- Splitting pins into several ports to transmit the different complex components (frequency, amplitude, phase for each harmonic) of a baseband signal between the models and modifying the top-level netlists with the help of scripts.

All these options are hacks to work around limitations of the AMS-HDL. It would be much better, if for continuos-time dataflow models (which use quantity ports), more complex datatypes could be used. I'm aware that VHDL-AMS supports quantity vectors, but more structurized datatypes could be usefule. Of course, another issue is always that tools need to support this language feature.

## PDEs

I understand the motivation to support PDEs in VHDL-AMS to allow more detailled modeling of, e.g., MEMS devices. I'm however not sure, if the proposed approach is general enough. However, I'm well aware of the usefulness to be able to use partial derivates in a behavioral model. Sometimes it is very cumbersome to have only the possibility to use directly the time-derivative/integral ('dot, 'integ). It would be nice, to be able to express other derivatives directly

(e.g., with a kind of 'dot(q), 'integ(q) syntax). If I understand correctly, the proposal goes in this direction.

**Follow-up**
There are different fields, for which PDE support in VHDL-AMS would be interesting:

- structural mechanics
- EMC problems
- thermal conductivity problems

What I'm concerned about is that users in this domain are used to tools with graphical model entry and result visualization -- even if the actual (parametrizable) model is defined in a script. It is very important to apply and check the meshing and complex boundary conditions. Additionally, these users usually don't write the rather complex element models: they just parametrize them. At least this is my experience with FEM modeling: I used for more than two years ANSYS for modeling various kinds of MEMS devices, which used mechanical, thermalresistive, electrostatic, and other effects.

I fear that the envisioned PDE support for VHDL-AMS might become too restrictive to be applicable to anything else than schoolbook problems. At least from the PDE proposal it seems to be limited to only the simplest geometries: one-dimensional line, two-dimensional rectangle, three-dimensional cuboids (and maybe their cylindrical, spherical counterparts in other coordinate systems). Maybe a more flexible solution in this case is to ensure that the definition of the VHPI-AMS language interface will allow an efficient coupling to external PDE codes -- may it be full-blown FEM tools like COMSOL, ANSYS, NASTRAN, etc. or dedicated PDE codes in form of C/C++/Fortran libraries. Current co-simulation approaches are not very performant due to insufficient interfaces between the behavioral and FE simulation tools.

Another promising approach applicable to at least some of the above stated problems are Reduced Order Modeling (ROM) methods. They extract extract accurate behavioral models from complex FE models based on systematic simulation of sweeped load cases applied to the FE model. The complexity of the behavioral model is usually smaller than a simplified PDE model (i.e., less unknowns to resolve). E.g, the modal ROM method I used during my Master's thesis allowed us to reduce the order of the equations system by a factor of more than thousand for a behavioral model of a fully coupled electrostatic-mechanical model of a MEMS gyrometer. However, the generation and implementation of such models could be much simplified by improving VHDL-AMS in the proposed directions table-based models and vector/matrix operations.

[Here is some information about] how providing direct support for partial derivatives in VHDL-AMS would simplify writing models. Well, partial derivatives appear often in the description of energy conserving behaviors, I will give two examples from my experience for which I had to develop VHDL-AMS models:

**(1) Electrostatic transducer model for MEMS applications**
Electrostatic transducers are widely used in micromechanical sensors to detect, e.g., accelerations and angular velocities. They couple the mechanical quantities force F and displacement s with the

electrical quantities voltage u and current i and are characterized by the function C(s) describing the capacitance over the displacement s of the moving electrode.

The mathematical relations describing their behavior can be derived from the energy balance:

```
dWmech + dWfield + dWelec == 0
F(s) * ds + d(0.5 * C(s) * u**2) == dQ(s) * u
```

with the charge Q(s) == C(s) * u, the force F(s) can be calculated:

```
F(s) == 0.5 * u**2 * dC(s) / ds
```

with the partial derivative of the capacitance over the displacement. Currently, this derivate needs to be calculated by hand and inserted into the equation to obtain an equation, which can be implemented in VHDL-AMS. This can be done if the function C(s) is known in advance, e.g.:

```
C(s) == c0 / s + c1 + c2 * u + c3 * u**2 + c4 * u**3
dC_ds == dC(s) / ds = -c0 / s**2 + c2 + 2 * c3 * u + 3 * u**2
F == 0.5 * u**2 * dC_ds;
```

However, this does not allow the creation of a generic electrostatic transducer model, which function C(s) is passed as a generic parameter -- a functionality, if I understand correctly, which will be available with the integration of the new features of VHDL 2008 into VHDL-AMS. Current VHDL-AMS model libraries suffer that it is not possible "to inject" new user-defined behavior into a given model using an arbitrary function, but instead to be limited to scalar/array generics.

For completeness of the example, the current i can be easily calculated as:

```
i == dQ/dt == d(C * u) / dt == dC / dt * u + C * du / dt
```

which is in VHDL-AMS directly expressible:

```
i == C'dot * u + C * u'dot;
```

What would be nice for 'dot and 'integ and other continuous time attributes to be applicable to temporaries in an equation instead of being obliged to introduce a temporary, e.g.,

```
i == (C * u)'dot;
```

instead of:

```
Q == C * u;
i == Q'dot;
```

However, I have seen cases in VHDL-AMS simulators where this doesn't yield the correct behavior defined by:

```
i == C'dot * u + C * u'dot;
```

**(2) VCO model with nonlinear varicap (**http://infoscience.epfl.ch/record/138641)

The varicap described in the referenced paper is described using a hyperbolic tangent function:

```
  -- varicap density
  d_cv(v_tune_eff) == D_CV_AMP * tanh((DDCV0_DV / D_CV_AMP) * (v_tune_eff - V_0)) +
D_CV0;
```

with the convention that lowercase identifiers are quantities and uppercase identifiers are constants.

The varicap capacitance is obtained by multiplying d_cv with the varicap surface S_CV:

```
  c_v == S_CV * d_cv;
```

The total tuning capacitance of the VCO also depends of the coupling and parasitic capacitances (C_C and C_P, respectively):

```
  c_vtot == 1.0 / (2.0 / C_C + 2.0 / (c_v + C_P));
```

The VCO frequency f_vco is calculated therefrom:

```
  f_out == 1.0 / (MATH_2_PI * sqrt(L_TANK * (C_TANK + c_vtot)));
```

An important design criteria for the VCO is the voltage sensitivity of output frequency k_vco, which is defined as:

```
k_vco == d(f_out) / d(v_tune_eff)
```

This cannot be currently expressed directly in VHDL-AMS using, e.g.:

```
k_vco = f_out'deriv(v_tune_eff)
```

but needs to be calculated lengthly by calculating all partial derivatives necessary by hand:

```
  -- d(cv) / d(v_tune_eff)
  dcv_dvtuneeff == S_CV * (1.0 - ((d_cv - D_CV0) / D_CV_AMP)**2) * DDCV0_DV;
  -- d(c_vtot) / d(v_tune_eff)
  dcvtot_dvtuneeff == 0.5 * (C_C / (c_v + C_P + C_C))**2 * dcv_dvtuneeff;
  -- d(f_out) / d(v_tune_eff)
  k_vco == 2.0 * MATH_PI**2 * f_out**3 * L_TANK * (-dcvtot_dvtuneeff);
```

The whole model looks therefore currently like this:

```
entity vco_freq_out is
```

```
  generic (  -- Primary parameters of each abstraction level
    F_OPT             : real     := 830.0e6;   -- Center frequency to be
optimized [Hz]
    K_OPT             : real     := 60.0e6;    -- Frequency sensibility [Hz/V]
    F_MIN             : real     := 750.0e6;   -- Minimum center frequency [Hz]
    F_MAX             : real     := 890.0e6;   -- Maximum center frequency [Hz]
    N_BITS            : positive := 3;         -- Number of bits for band
selection [1]
    N_BANDS           : positive := 8;         -- Number of bands
    RATIO_CC_CV0      : real     := 4.0;       -- Ratio of C_C and C_VO [1]
    RATIO_CP_CC       : real     := 0.01;      -- Ratio of C_P and C_C [1]
    L_TANK            : real     := 5.0e-9;    -- Tank inductance [H]
    R_IN              : real     := 20.0e3;    -- Input resistance [Ohm]
    -- Technology constants of the varicap
    V_0               : real     := 0.087;     -- Offset voltage [V]
    D_CV0             : real     := 6.2e-3;    -- Varicap cap. density at V_0
[F/m**2]
    DDCV0_DV          : real     := -9.3e-3;   -- Slope of D_CV at V_0 [F/(m**2
* V)]
    D_CV_AMP          : real     := 3.0e-3;    -- Cap. density modulation amp.
[F/m**2]
    -- Component parameters from real design for bottom-up verification
    S_CV_DESIGN       : real     := 3.63e-10;  -- Varicap surface [m**2]
    C_TANK_MIN_DESIGN : real     := 5.47e-12;  -- Minimum tank capacitance [F]
    C_TANK_MAX_DESIGN : real     := 8.08e-12;  -- Maximum tank capacitance [F]
    DEBUG             : boolean  := false);    -- Debug instance
  port (  -- Pin-accurate
    signal band : in std_logic_vector(N_BITS-1 downto 0)
                  := std_logic_vector(to_unsigned(0, N_BITS));  -- Band selector
    terminal power_p, power_m :     electrical;  -- Power supply terminals
    terminal tune_p, tune_m   :     electrical;  -- Tuning terminals
    quantity f_out            : out real);       -- VCO frequency [Hz]
begin
  -- Interface-level assertions
end entity vco_freq_out;

architecture td_detailed of vco_freq_out is
  -- Definition of support functions...
  -- Calculation of secondary parameters...
  -- Quantity and Signals declaration...
begin
  -- Debug code and architecture-level assertions...

  -- Behavior implementation:
  -- Power supply
  i_power == 0.0;
  -- load at input
  c_in == 2.0 * c_v + C_P + C_C;
  v_tune_eff == (1.0 / c_in) * i_tune'integ;
  v_tune == R_IN * i_tune + v_tune_eff;
  -- Band switching
  c_tank <= C_TANK_MAX - real(to_integer(unsigned(band))) * C_TANK_STEP;
  break on c_tank;
  -- Varicap capacitance
  d_cv == D_CV_AMP * tanh((DDCV0_DV / D_CV_AMP) * (v_tune_eff - V_0)) + D_CV0;
  c_v == S_CV * d_cv;
  dcv_dvtuneeff == S_CV * (1.0 - ((d_cv - D_CV0) / D_CV_AMP)**2) * DDCV0_DV;
```

```
  c_vtot == 1.0 / (2.0 / C_C + 2.0 / (c_v + C_P));
  dcvtot_dvtuneeff == 0.5 * (C_C / (c_v + C_P + C_C))**2 * dcv_dvtuneeff;
  -- Output frequency
  f_out == 1.0 / (MATH_2_PI * sqrt(L_TANK * (c_tank + c_vtot)));

  -- Compute design data (e.g. performance estimates):
  -- Voltage sensitivity of VCO frequency
  k_vco == 2.0 * MATH_PI**2 * f_out**3 * L_TANK * (-dcvtot_dvtuneeff);
end architecture td_detailed;
```

With the addition of 'deriv(q), the architecture implementation could look like this:

```
architecture td_detailed of vco_freq_out is
  -- Definition of support functions...
  -- Calculation of secondary parameters...
  -- Quantity and Signals declaration...
begin
  -- Debug code and architecture-level assertions...

  -- Behavior implementation:
  -- Power supply
  i_power == 0.0;
  -- load at input
  c_in == 2.0 * c_v + C_P + C_C;
  v_tune_eff == (1.0 / c_in) * i_tune'integ;
  v_tune == R_IN * i_tune + v_tune_eff;
  -- Band switching
  c_tank <= C_TANK_MAX - real(to_integer(unsigned(band))) * C_TANK_STEP;
  break on c_tank;
  -- Varicap capacitance
  d_cv == D_CV_AMP * tanh((DDCV0_DV / D_CV_AMP) * (v_tune_eff - V_0)) + D_CV0;
  c_v == S_CV * d_cv;
  c_vtot == 1.0 / (2.0 / C_C + 2.0 / (c_v + C_P));
  -- Output frequency
  f_out == 1.0 / (MATH_2_PI * sqrt(L_TANK * (c_tank + c_vtot)));

  -- Compute design data (e.g. performance estimates):
  -- Voltage sensitivity of VCO frequency
  k_vco == f_out'deriv(v_tune_eff);
end architecture td_detailed;
```

Sometimes partial derivatives towards various quantities need to be calculated for one model and once the base equation changes, all depending equations need to be readapted. This is quite cumbersome, as partial derivatives can be systematically derived also by a computer, and which is therefore available in any computer algebra system (CAS). Therefore, it would be nice if VHDL-AMS could offer in the future a syntax to express these partial derivatives directly in the model, e.g.,

C'deriv(s) == dC(s) / ds

Multiple application of this new attribute to express further partial derivation would be nice, e.g.,

q3(q2, q1) == ...

q3'deriv(q1)'deriv(q2) == d(dq3 / dq1) / dq2

I don't oversea all implications of the introduction of this syntax extension to VHDL-AMS. Of course it would oblige VHDL-AMS simulators to implement some symbolic manipulation of the model equations like a CAS does during the elaboration of the VHDL-AMS model. In the case of non-trivial derivatives (user-defined (non-analytical) functions, piece-wise defined derivatives, non-continuous derivatives, etc.) the simulator might even have to fall back on a numerical solution of the partial derivative at the current operation point. But if I understand correctly, this is already the case for the handling of the 'dot and 'integ attributes in certain VHDL-AMS simulators.

I hope these two examples illustrate well, why I judge the support of partial derivatives in VHDL-AMS as useful.

## Table-driven modeling

Very useful to create behavioral models of measured components or by means of reduced-order modeling.

## Vector/matrix operations

VHDL-AMS current weakness is to allow basically only the description of scalar equations, which obliges the model author often to repeat the basically same equation for each dimension. Especially for modelling mechanical systems or expressing in a compact form state-space equation systems, such an extension would be very useful. This also requires the support of the 'dot and 'integ for the quantity vector types!

#### Follow-up

I'm well aware that VHDL-AMS already offers composite quantities and terminals, but that the tools still lack behind in their implementation. To improve the situation, the VHDL-AMS standardization effort should also include the definition of standard packages, which implement the missing operations for vectors/matrices to make the currently existing language features more user-friendly, which will motivate more people to use them. It is also important to facilitate the exchange of models. There was a similar issue over many years with the interoperability of multiple physical domain models due to the lack of standardized disciplines packages, which has been thankfully finally overcome with IEEE 1076.1.1.

I'm aware that generate statements are in principle applicable to continuous-time equations, but, unfortunately, it is not well supported by the simulators.

## SPICE

I'm not sure how important it is to be able to instantiate individual SPICE elements. However, what in my experience is missing, is a standard way to instantiate SPICE subcircuits in VHDL-AMS: I think of something like declaring an equivalent component with the ports and parameters of the

SPICE subcircuits and having standardized attributes to reference the SPICE subcircuit and physical location of the SPICE netlist/libraries to be able to integrate it into the VHDL-AMS simulation.

### LTF

It would be nice to make the 'ltf, 'ztf more flexible. E.g., for digitally configurable filters it would be nice to be able to change the coefficients of the transfer functions (in a discrete way). The change could be announced through a break. It needs to be seen, if the order of the transfer function is allowed to change. It would be also nice to be able express an LTF not only in the numerator/denominator form, but also in the zero-pole form. E.g, the upcoming OSCI SystemC AMS extensions standards offers much more flexibility in this direction.

### RF

What would be very interesting for the application of VHDL-AMS in the domain of RF systems, would be to add support to the language, which facilitates the description of baseband models, i.e., the support of complex data types also for quantities; the support for records and arrays of records

to encode the tuples frequency, phase, amplitude for each harmonic, etc. It should be possible to use these new data types in block diagram style models, that is today models with VHDL-AMS models with quantity in- and out-ports.

## Sameer Kher

### Participation

I would like to try to participate/help in the working group if I can.

## Uwe Eichler

### TLU

I used the table lookup functions of the Open Source VHDL-AMS packages fundamentals_vda/tlu_vda in recent projects. Maybe these can be included in the VHDL-AMS standard. Additionally, I suggest to add similar functions that take handles instead of whole table data as arguments. This may simplify the usage of these functions and may also increase performance of the function call. The table data and the index vectors should then be passed in an initial call to a separate init function returning that handle. see also: http://fat-ak30.eas.iis.fraunhofer.de/index_en.html.

## Steve Grout

### Sensitivity, Statistical Modeling

A VERY IMPORTANT NEED - VHDL-AMS needs to directly support sensitivities as part of precisely defining non-deterministic values, value-sets, relationships, and other modeling. When I first implemented my own circuit simulator in 1966 as a working designer, I had no choice but include part tolerances, sensitivities, worstcase and other statistical results, means to directly desensitize my design, design centering using actual measured part data, and predicting real manufacturing yields. I continued implementing EDA that supplied this design information until I got stuck in companies (and MCC and SEMATECH) that had no analog/AMS tools that provided both the variational part-value data, the sensitivities to make use of the variational data, and the now-called-statistical simulation capabilites.

- It makes a person an entirely difference kind of designer when you are able to design in an analog environment providing that type of capability. Few companies have had such capabilities for their designers: IBM with the ASTAP/STAR/SAP (I think those are the names), GE Heavy Military in late 60's/early-70's, GE Audio in the 60's (my original department), Rockwell,...and then not much more.
- Sensitivities with an HDL are straight forward as you are already symbolic via it being a language. The semantics are rooted in sophomore EE and the math EE rides on. The SPICE modeling always used them but only within the 'box' of the device model.
- If there needs to be scoped plan of steps to move VHDL-AMS into this realm, I might consider same. But this capability within both analog design, the modeling of analog/AMS, simulators, and related needed variational/statistical EDA capabilities has long been needed.
- I certainly will be most glad to help write the requirements, rationale, and draft language spec, syntax, and semantics.

I respectfully submit the above for urgent consideration.

### Participation

I also would be glad to help with the writing, editing, and, as the work comes together, putting it into any chosen 'tech-pubs' format, as well as helping in any way re the lists of tasks you include below.

## François Pêcheux

### Checking if equations are homogeneous

### Quantities, dimensional analysis

I am interested in dimensional analysis and such, I would like quantities to be associated with the appropriate physical dimension.

### PDEs (Space dependent operators)

Add language constructs that allow to define PDEs without too much difficulty, like the wave equation I use to model seismic perturbations for instance.

## In Hur

### Participation

I am In Hur, interested in VHDL-AMS, and a user. I am sending this mail from Nagoya, Japan. I work for the IT company that supplies simulation models for automotive system to many Japan companies including Toyota Motors.

I have been developed simulation models based on VHDL-AMS with Toyota Motors of Japan for 2 years. My works have been related to DAE (Differential Algebraic Equation) because those were to develop simulation models based on physics, heat transfer, and mechanics. For example,

- Evaluation model for amount of air-flow in the throttle valve of engine.
- 3 dimensional analysis of vehicle indoor temperature with VHDL-AMS, etc.

This models are not only for IC but also for thermal, electrical, mechanical, and control domains.

I eagerly want to participate in the New Feature Projects because I want to know latest technological trends on VHDL-AMS and support for the development of VHDL-AMS.