Important Characteristics of VHDL-AMS and Modelica with Respect to Model Exchange

Olaf Enge-Rosenblatt, Joachim Haase, Christoph Clauß

Fraunhofer Institute for Integrated Circuits, Design Automation Division, Zeunerstr. 38
D-01069 Dresden, Germany
{Olaf.Enge, Joachim.Haase, Christoph.Clauss} @eas.iis.fraunhofer.de

Abstract. Modeling and simulation have been established as fundamental facilities in the development of analog and analog-digital systems. Essential advances have been achieved by the usage of behavioral modeling languages. These languages can be considered as a link between the technical problem and the mathematical model that can be evaluated by computational methods. The paper outlines the various possibilities that are offered by the language VHDL-AMS – standardized by the IEEE to describe analog and mixed-signal systems – and the language Modelica. The underlying modeling approaches are compared. Last but not least, the potential to transform models written in one language into models of the other language is discussed.

Keywords: VHDL-AMS, Modelica, model exchange

1 Introduction

A multitude of behavioral modeling languages has been developed during recent years [1, 2, 3, 4]. They provide among other things a link between the mathematical description of an object and a representation that can be used by a simulation engine. Currently it can not be expected that all languages are supported by all simulation tools. On the other hand, large efforts have been spent on developing model libraries. Possibilities and limitations of the transformation of models written in VHDL-AMS [1] and Modelica [2] are discussed in the paper.

VHDL-AMS is a behavioral modeling language that is an extension of VHDL for the analysis of analog and mixed-signal systems. The language is standardized by IEEE. VHDL was originally developed to describe and model digital electronic circuits. The analog and mixed signal extensions were partly based on the experiences with the program Spice [5]. This is a program to simulate Kirchhoffian networks. Spice-like simulation tools have been available for more than thirty years. The first implementations of tool-specific behavioral languages to describe the analog behavior include for instance MAST [6] which is one of the ancestors of VHDL-AMS.

Modelica is a modeling language that allows the specification of complex systems. Especially it is widely used to describe non-electrical systems. The description of analog systems is supported as well as discrete, hybrid, and concurrency modeling.

A very active community has developed many models and libraries during recent years.

From a general point of view, both languages have a lot of similarities. Modeling is primarily based on equations. A system description can be established by connecting subsystems in a hierarchical way. A system of equations that considers the restrictions of the unknowns with respect to the terminal behavior of the subsystems, the topology of the final system, and some augmentation sets (for example to describe initial conditions) are generated. Both languages support the modeling of time-continuous conservative systems (based on a network approach), non-conservative systems (described by signal-flow blocks), and time-discrete systems. Modeling of multidomain systems that consist of electrical and non-electrical components is possible.

The time-continuous part of a well-established analysis problem is described by a differential algebraic system of equations [7] of form

$$F(x(t), x'(t), t) = 0 \qquad (1)$$

$$\text{with} \qquad F : R^n \times R^n \times R^+ \rightarrow R^n \text{ and } x : R^+ \rightarrow R^n$$

The solution method is not defined by the language descriptions.

The VHDL-AMS language reference manual [1] defines characteristic equations that must be fulfilled by the solution. The structural set of these equations considers the Kirchhoff laws and represents the equality of quantities at the connection points of non-conservative ports. The explicit set is given by the simultaneous statements of the models. In the case of a conservative network, these equations describe the branch constitutive relations. Additional restrictions that must be considered in the initialization phase and at discontinuities are established by the augmentation set. It has to be considered during modeling that these equations must be fulfilled by a solution of the simulation problem [8]. A solvability check is done at the design unit level. Roughly spoken, the number of essential unknowns that are contributed by a model to the characteristic equations of the entire system must match the number of explicitly added model equations. In the case of VHDL-AMS simulators, the unknowns that fulfill the characteristic expressions are determined using a modified nodal analysis approach [9]. In this way, the dimension *n* of the system of form (1) that is solved by the simulator is reduced compared to the number of characteristic equations. The time-discrete part is solved by event-driven simulation approaches. Besides time domain simulation, small signal frequency and noise domain simulation are supported by VHDL-AMS. The language reference manual describes how to establish the characteristic expressions in these cases. They base on a linearization of (1) around the operating point.

A similar mechanism is used in Modelica. For each connection point of instances of an entire system connection equations contribute to the system (1). These equations correspond to the structural set in VHDL-AMS. The non-flow variables are set equal and the flow variables are summed to zero. The equations of each instance are included to build up (1). They represent the explicit set in VHDL-AMS. Additional requirements can be expressed to initialize unknowns in the initialization phase or to reinitialize values after discontinuities. This is similar to the augmentation set in VHDL-AMS. In contrast to VHDL-AMS, violations of these augmented equations

are handled “liberal” [10] (section 8.4.1.5). In the context of Modelica, reduction algorithms are applied to simplify the system (1). The number of unknowns and equations of the reduced system must be in accordance. A check on the unit level as in VHDL-AMS is not required. Similar to the mixed-simulation cycle in VHDL-AMS, Modelica defines how to handle hybrid differential algebraic representations that consist of differential, algebraic, and discrete equations. Table 1 compares some aspects of both languages in a general way. More details can be found in [1, 2, 10, 11].

Table 1. Comparison of some aspects of VHDL-AMS and Modelica

Aspect	VHDL-AMS	Modelica
Definition	IEEE Std. 1076.1 (2007) [1]	Modelica Specification 2.2 [2] Modelica Association
Time-continuous	Conservative (Kirchhoff networks) non-conservative	physical modeling block-oriented modeling
Time-discrete	event-driven	event-driven, but no event queue supported
Interaction	mixed-signal simulation cycle	solution of hybrid DAE’s
Model interface	**entity**	**model**, **block**
Model parameter	**generic** parameter	**parameter**
Connection point	**port** (terminal,quantity,signal)	specified by connector classes
Model behavior (general)	**architecture** (one ore more corresponding to one entity)	declarations and **equation** part of **model, algorithm**
Analog behavior	equation oriented; simultaneous statements (for instance *expr1 = = expr2*;)	equation oriented; equations (for instance *expr1 = expr2*;)
Event-driven behavior	assignment of values; concurrent statements (for instance **process**)	assignment of values; conditional equations (**when**-equations)
Analog waveform	**quantity**	dynamic variable
Connection point characterization	**nature** for terminals (**through, across, reference**)	**connector** (**flow,** non-flow)
Digital waveform	**signal**	**discrete**
Simulation	Time domain, small-signal frequency and noise analysis	Time domain analysis
Simulation phase information	DOMAIN signal	**initial**()
Initial conditions	**break** statement	**initial equation** **fixed start** values
Values after discontinuities	**break s**tatement	**reinit**()
D/A conversion	‘RAMP, ‘SLEW	**smooth**()
Vector operations	Overloading of operators	Built-in functions
Inheritance	Not supported	Widely used
Netlists	instance oriented (**port map**)	pin oriented (**connect**)

The classes of problems that can be described with both languages seem to be very similar. Thus, it is obvious to check the possibilities to use models described in both languages together or to transform from one language into the other. In the following, aspects of the transformation of Modelica models into VHDL-AMS models are especially considered.

2 Modeling Approaches in VHDL-AMS and Modelica

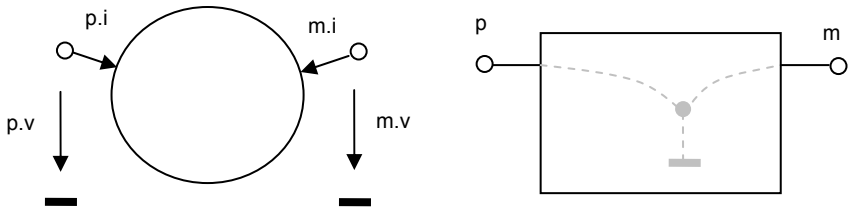


Fig. 1. Terminal behavior of conservative (electrical) systems in Modelica and VHDL-AMS (examples)

A physical Modelica model describes the relation between pin flow and pin potential variables (see for example in Fig. 1 p.i and p.v resp.) by a set of equations. Additional variables can be introduced. Usually the number of equations equals the sum of the number of pins and additional variables. This is not strictly required but usually fulfilled. It is only required that the final system (1) is well-defined. In VHDL-AMS, an internal graph structure is declared. The constitutive relations between branch across (non-flow) and through (flow) quantities are described by simultaneous statements. Additional free quantities can be declared to establish the constitutive relations. Without considering all details, it is in principle strictly required that the number of branches and free quantities equals the number of simultaneous statements.

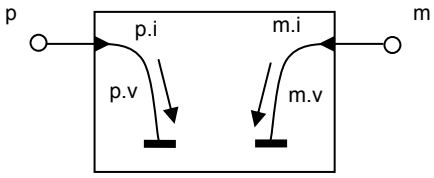
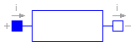


Fig. 2. Internal branch structure of a Modelica model (example)

A branch structure can be assigned to a Modelica model in a direct way - see Fig. 2. The associated model descriptions of a resistor using Modelica and VHDL-AMS are shown in Table 2.

Table 2. Example of a Modelica model and the associated VHDL-AMS model

Modelica	VHDL-AMS
	

`partial model OnePort`
`  "Component with two pins"`
`  ….Voltage v  "p.v - n.v";`
`  ….Current i  "from p to n";`
`  ….PositivePin p;`
`  ….NegativePin n;`
`equation`
`  v = p.v - n.v;`
`  0 = p.i + n.i;`
`  i = p.i;`
`end OnePort;`
 	
`model Resistor "Ideal resistor"`
`  extends ….OnePort;`
`  parameter ….Resistance R=1;`
`equation`
`  R*i = v;`
`end Resistor;`
	`library IEEE;`
	`use IEEE.ELECTRICAL_SYSTEMS.all;`
	
	`entity RESISTOR is`
	`  generic(R : RESISTANCE := 1.0);`
	`  port      (terminal P: ELECTRICAL;`
	`             terminal N: ELECTRICAL);`
	`end entity RESISTOR;`
	
	`architecture MODELICA of RESISTOR is`
	`  quantity P_V across P_I through P;`
	`  quantity N_V across N_I through N;`
	`  quantity V   : REAL;`
	`  quantity I   : REAL;`
	`begin`
	`  V   == P_V - N_V;`
	`  0.0 == P_I + N_I;`
	`  I   == P_I;`
	`  R*I == V;`
	`end architecture MODELICA;`
	
	`architecture IDEAL of RESISTOR is`
	`  quantity VNOR across INOR`
	`           through P to N;`
	`begin`
	`  VNOR == R*INOR;`
	`end architecture A0;`

The `architecture MODELICA` of the VHDL-AMS model is a direct transformation of the corresponding Modelica model on the left side of Table 2. This description is correct from a formal point of view. The better description with a reduced set of internal quantities and branches is given by the `architecture A0`. This simple example shows the limits of a formal straight forward transformation of a Modelica model into VHDL-AMS. In section 4, we will discuss a way to handle this problem.
By the way, constructing an appropriate internal branch structure is not only helpful in the model transformation process. It produces also a better understanding of existing models (in any case from an electrical engineer's point of view) - see for example Fig. 3.

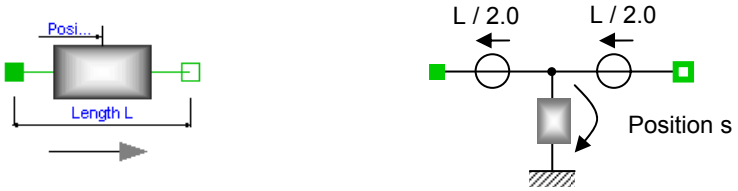


Fig. 3. Internal branch structure of the sliding mass model of the Translational Modelica Mechanics Standard Library (see [10])

Facilities to handle finite state machines are available in both, Modelica and VHDL-AMS. The associated statements allow establishing event-driven analog models. The equations used in a time-continuous analog model depend on the value of a digital STATE. The STATE is updated by evaluating the transition conditions.

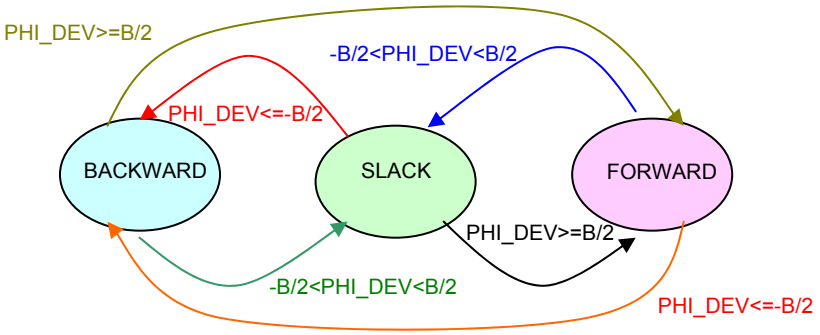
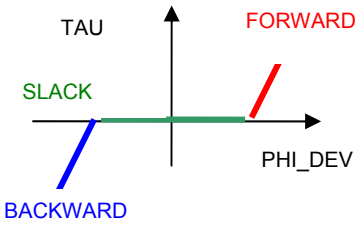


Fig. 4. Finite State Machine for event-driven analog modeling (example)

Handling finite state machines and time-discrete simulation problems is well supported in VHDL-AMS. It must be qualified how the translation of the constructs for hybrid simulation in Modelica (in VHDL-AMS named as mixed-signal simulation) into VHDL-AMS and vice versa can be done. A simple example of an event-driven analog model is shown in Fig. 5. The corresponding Modelica description can be found in [10] (chapter 13).

```
library IEEE;
use IEEE.MECHANICAL_SYSTEMS.all;

entity SIMPLE_ELASTOBACKSLASH is
 generic  (
   B         : REAL ;
   C         : REAL := 1.0E5;
   PHI_REL0  : REAL := 0.0 );
 port     (
  terminal FLANGE_A : ROTATIONAL;
  terminal FLANGE_B : ROTATIONAL);
end entity SIMPLE_ELASTOBACKSLASH;
```




```
architecture BASIC of SIMPLE_ELASTOBACKSLASH is
  -- STATE declaration for finite state automaton

  type     STATE_TYPE is (SLACK, FORWARD, BACKWARD);
  signal   STATE    : STATE_TYPE;

  quantity PHI_REL across TAU through FLANGE_B to FLANGE_A;
  quantity PHI_DEV : REAL;

begin
  -- angle deviation from zero position

  PHI_DEV == PHI_REL - PHI_REL0;

  -- finite state automaton
```

```
  STATE <= BACKWARD when not PHI_DEV'ABOVE(-B2) else
           FORWARD  when PHI_DEV'ABOVE(B2)      else
           SLACK;

  -- constitutive relations

  if       STATE = FORWARD  use
           TAU == C*(PHI_DEV - B/2.0);
  elsif    STATE = BACKWARD use
           TAU == C*(PHI_DEV + B/2.0);
  else     TAU == 0.0;
  end use;

  break on STATE;

end architecture BASIC;
```

Fig. 5. Event-driven analog VHDL-AMS model of a simple elastobackslash

3 Potential to Establish Small VHDL-AMS Models

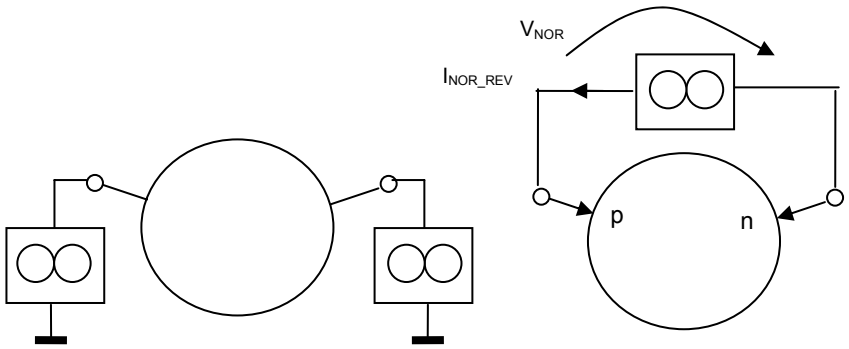


Fig. 6. Structures to determine the terminal behavior

In section 2, a direct way how to transform a physical Modelica model into a VHDL-AMS model was demonstrated. We will now draft how to simplify the structure and equations of the target model.

The terminal behavior of a conservative system can be determined by an interconnection with a tree structure of norator circuit elements [12]. A norator is a one-branch network in which branch voltage and current are completely arbitrary [13]. There are no restrictions concerning branch voltage and branch current.

If the sum of the flows into the component that has to be modeled is unequal zero, all pins have to be connected by norators to the corresponding reference node (see left part of Fig. 6). If the sum equals zero, one terminal can be used as local reference terminal and the other terminals are connected to the local reference by norator branches.

The structure of the final model is the same as the structure of the norator tree(s). The analysis of the test structure delivers the model equations that are expressed with the branch quantities of the norator branches.

Example

Let us have a look at the Modelica model from Table 2. The sum of the currents p.i and n.i is 0.0 (see the equations of the partial model OnePort). Thus, we can use a test circuit similar to the right part of Fig. 6. The network equations are given by

$$\begin{pmatrix} 1 & -1 & . & . & -1 & . & . & . \\ . & . & 1 & 1 & . & . & . & . \\ . & . & 1 & . & . & -1 & . & . \\ . & . & . & . & 1 & -R & . & . \\ 1 & -1 & . & . & . & . & -1 & . \\ . & . & 1 & . & . & . & . & -1 \\ . & . & . & 1 & . & . & . & 1 \end{pmatrix} \cdot \begin{pmatrix} p.v \\ n.v \\ p.i \\ n.i \\ v \\ i \\ V_{NOR} \\ I_{NOR\_REV} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

The first four equations represent the equation part of the Modelica model from Table 2. The last three equations result from the Kirchhoff laws. There is no additional constitutive relation that restricts branch voltage and current of the norator. Thus, there are fewer equations than unknowns.

These equations (2) can easily be transformed (for instance using a row echelon algorithm) into the following form

$$\begin{pmatrix} 1 & -1 & . & . & . & . & . & -R \\ . & . & 1 & . & . & . & . & -1 \\ . & . & . & 1 & . & . & . & 1 \\ . & . & . & . & 1 & . & . & -R \\ . & . & . & . & . & 1 & . & -1 \\ . & . & . & . & . & . & 1 & -R \\ . & . & . & . & . & . & . & . \end{pmatrix} \cdot \begin{pmatrix} p.v \\ n.v \\ p.i \\ n.i \\ v \\ i \\ V_{NOR} \\ I_{NOR\_REV} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3}$$

The constitutive relation of the model only depends on the equations that now restrict the branch voltage and current V_{NOR} and I_{NOR_REV} resp. in the norator branch

$$V_{NOR} - R \cdot I_{NOR\_REV} = 0 \tag{4}$$

This equation and the used norator structure are the basis of the `architecture A0` of the VHDL-AMS model in Table 2. ■

We will try to outline the general procedure that was applied in the example:

1. Starting point is a physical (conservative) Modelica model.
2. Combine pins in one group if the sum of the associated flows equals zero.
3. Select a local reference pin in each group. Connect all other pins of the group with the associated reference pin by norator branches. Connect the remaining pins by norator branches with the associated global reference node. In the norator branches, voltages and currents are contrarily counted.
4. Establish the network equations of the designed test circuit

$$F(x(t), x(t)', t) = 0 \tag{5}$$

with $F : R^n \times R^n \times R^+ \rightarrow R^m$ and $x : R^+ \rightarrow R^n$

5. Try to transform this equations into

$$F_1(x_1(t), x_2(t), x_3(t), x_1'(t), x_2'(t), x_3'(t), t) = 0 \tag{6.1}$$

$$F_2(x_2(t), x_3(t), x_2'(t), x_3'(t), t) = 0 \tag{6.2}$$

with $F_1 : R^{n_1} \times R^{n_2} \times R^{n_3} \times R^{n_1} \times R^{n_2} \times R^{n_3} \times R^+ \rightarrow R^{m_1}$

$F_2 : R^{n_2} \times R^{n_3} \times R^{n_2} \times R^{n_3} \times R^+ \rightarrow R^{m_2}$

$x_1 : R^+ \rightarrow R^{n_1}$, $x_2 : R^+ \rightarrow R^{n_2}$, $x_3 : R^+ \rightarrow R^{n_3}$

All norator branch voltages and currents are summarized in x_3.

6. If $m_2 = n_2 + \frac{n_3}{2}$, a VHDL-AMS model can be established. The internal branch structure of the model is given by the structure of the norator branches and their voltage directions. Additional free quantities that represent the x_2 waveforms must be declared in the architecture. The simultaneous statements of the model are given by F_2.
7. In a final step the VHDL-AMS model has to be checked against the Modelica model.

The procedure cannot only be applied to a basic model but also to an interconnection of models. It can be simplified if the transformation of the equations (5) into (6.1) and (6.2) can make use of the facilities of a simulator. This requires access to the reduced equations created by the simulator.

The transformation from VHDL-AMS to Modelica can be done in a similar way.

4 Conclusions

Some aspects of the transformation of Modelica models into VHDL-AMS descriptions were discussed.

A pure code-based transformation will not deliver the expected quality in a lot of cases. A method was prototyped how the transformation could be supported by analyzing appropriated test circuits. This could be a basis for a model transformation process if the reduced equations are available as a result of the evaluation of a Modelica description. The availability of differential algebraic equations that are

established by the simulation program would offer a lot of opportunities concerning model transformation. This is also valid for a transformation from VHDL-AMS to Modelica.

Many other aspects of model transformation could not be taken into consideration in this paper in detail. A special problem is the handling of initial values for instance. There are a lot of facilities in Modelica (for instance based on the experiences with index problems of mechanical system analysis). VHDL-AMS offers a solution based on the experiences with operating point analysis of electronic circuits.

An exchange of models between the two languages should not bring up problems in principle. However, a full automatic transformation of models seems only possible under special conditions.

Acknowledgement

The authors would like to thank the referees for helpful comments.

References

1. IEEE standard VHDL analog and mixed-signal extensions. IEEE DASC, December 1999 (revised May 2007). Online: http://www.designers-guide.org/Modeling
2. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Language Specification. Version 2.2, Modelica Association, February 2005. Online: http://www.modelica.org
3. Verilog-AMS Language Reference Manual. Version 2.2. Accellera, November 2004. Online: http://www.eda.org/verilog-ams
4. Vachoux, A.; Grimm, C.; Einwich, K.: SystemC-AMS Requirements, Design Objects and Rationale. Proc. DATE '03, Munich, March 2003, pp. 388-393. Online: http://www.systemc-ams.org
5. Nagel, L. W.; Pederson D. O: SPICE2 – Simulation program with integrated circuits emphasis. Univ. of California, ERL-Memo M520, 1975.
6. Vlach, M.: Modeling And Simulation with Saber. Proc. 3rd Annual IEEE ASIC Seminar, September 1990, pp. 17-21.
7. Brenan, K.E.; Campbell, S.L.; Petzold, L.R.: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. North-Holland, 1989.
8. Haase, J.: Rules for Analog and Mixed-Signal VHDL-AMS Modeling. In Grimm, C. (ed.): Languages for System Specification. Kluwer Academic Publishers, 2004, pp. 169 – 182.
9. Ho, C.-W.; Ruehli, A.; Brennan, P.: The modified nodal approach to network analysis. IEEE Trans. Circ. Syst. 22(1975)6, pp. 504-509.
10. Fritzson, P.: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. IEEE Press, 2004.
11. Lallement, C.; Pecheux, F.; Vachoux, A.; Prégaldiny, F. : Compact Modeling of the MOSFET in VHDL-AMS. In Grabinski, W.; Nauwelaers, B.; Schruers, D.: Transistor Level Modeling for Analog/RF IC Design. Springer-Verlag, 2006, pp. 243-269. Online: http://lsmwww.epfl.ch/models/compact/
12. Reibiger, A.: On the terminal behavior of networks. Proc. ECTD '85, Prague, 1985, pp. 224-227.
13. Carlin, H. J.; Youla, D. C.: Network synthesis with negative resistors. Proc. IRE 49 (1961) 5, pp. 907-920.