Standard VHDL 1076.1.1 Packages for Multiple Energy Domain Support

Peter R. Wilson & Andrew D. Brown
Department of Electronics and Computer Science
University of Southampton
Southampton, UK

H. Alan Mantooth
Department of Electrical Engineering
University of Arkansas
Fayetteville, AR, USA

Abstract

This paper describes the set of packages providing a standard for the declaration of the most frequently used constants and types required for multiple energy domain modeling. Use of these packages with their defined types, constants and attributes is intended to provide a mechanism for writing VHDL models (compliant with IEEE Std 1076.1-1999) that are portable and interoperable with other VHDL models adhering to this standard. The standard serves a broad class of applications including electronics, thermal, magnetic, optical, fluidic and mechanical systems.

The work described is the culmination of efforts by groups with the same basic goals working over a period of over two years. During this time many individuals have made valuable contributions to the development of this standard from the academic, industrial and EDA communities.

In this paper, rather than describe every aspect of the standard in detail (impractical in a short paper), we have highlighted the key elements of the proposed standard.

Introduction

The packages are intended for use primarily in the modeling of multiple energy domain systems. The range of operation of the packages is not defined in this standard, but is intended to be valid across a wide range of disciplines and applications. The range and scope of applications have been essentially defined by the interests and requirements of the existing user community of VHDL-AMS (defined by the IEEE Std 1076.1 [1]). Christen, et. al. [2],[3] and Vachoux [7] each provide an overview of the rationale behind the language especially with regard to the specific extensions addressed by the new set of packages, namely natures, quantities and terminals. This is required because the extensions to VHDL encapsulated in the 1076.1 standard only state the mechanism of mixed-signal systems and does not address the specific packages required to support the practical implementation of such systems.

In the "conventional" electronics arena, the nature of the VHDL-AMS language is designed to support "mixed-signal" systems (containing digital elements, analog elements and the boundary between them) with a focus on IC design. Typical examples of this kind of application are described in [4-6] & [8-11]. There are of course many other similar applications using VHDL-AMS that can be found in the literature.

Where the strengths of the VHDL-AMS language have really become apparent however, is in the multi-disciplinary areas of mechatronic and micro electro mechanical systems (MEMS) [12-16].

Being able to represent these multiple disciplines and technological interfaces in a single modeling framework is incredibly powerful, however some pitfalls present themselves to the unwary. The first potential problem is defining a common, standard, interface such that models created within different simulators will have consistent definitions of natures, quantities and terminals to allow them to be connected together. If this is not the case, then problems can obviously occur when models and libraries undergo deployment. Another related problem is in the underlying assumptions made in models with regard to constants. If the same name is used, but different value (or accuracy for example) is used, then erroneous results can occur in simulations of ostensibly the same circuit. Taking these issues into account and building on the pioneering work of the early adopters of the VHDL-AMS modeling language allows a clear requirement for a set of multiple domain packages that define basic physical constants and interface conventions.

The packages proposed in IEEE standard 1076.1.1 were therefore chosen for two purposes. The first was to define a set of basic physical constants (either with or without default values) so that models written using these packages could have a common basis for modeling physical systems. The second purpose was to define a set of physical types that would provide a common framework for modeling physical systems across a range of commonly used energy domains. These enable models written using this proposed standard not only use the same fundamental physical constants, but also ensure that the interfaces are consistent, correct and maintain interoperability between users and computer simulation programs.

New Definitions

In order to provide supplementary information required for multiple domain systems such as units and symbols, two new attributes are defined in the proposed standard **UNIT** and **SYMBOL.**

A. Unit

The UNIT attribute is defined as a string and is used to define the name of the fundamental unit of the declared type. For example, the name of the unit for voltage in electrical systems is defined as "Volt". The convention adopted in this standard is for the initial letter to be capitalized if the unit is named after an individual, otherwise the unit is in lowercase.

B. Symbol

The SYMBOL attribute is defined as a string and is used to define the abbreviation of the fundamental unit of the declared type. For example, the name of the symbol for voltage in electrical systems is defined as "V". The convention adopted in this standard is for the initial letter to be capitalized if the symbol is named after an individual, otherwise the symbol is in lowercase.

Constants Packages

The constants have been divided into two categories, fixed and user-definable. The fixed constants are fundamental physical constants that have an accepted value and do not generally vary. The user-definable constants are defined without a default value, allowing them to be overwritten in the model by the user. This allows the package to contain names of commonly used constants, without restricting the value to specific cases.

The fundamental constants and their default values are given in Table 1 in the proposed standard package: **FUNDAMENTAL_CONSTANTS**.

TABLE 1
FUNDAMENTAL CONSTANTS

Constant	Unit	Name	Default
Electron charge	C	PHYS_Q	602_176_462e-19
permittivity of vacuum	F/m	PHYS_EPS0	8.854_187_817e-12
permeability of vacuum	H/m	PHYS_MU0	4.0e-7 * MATH_PI
Boltzmann's constant	J/K	PHYS_K	1.380_6503e-23
Acceleration due to gravity	ms^{-2}	PHYS_GRAVITY	9.806_65
Conversion between Kelvin and degree Celsius	-	PHYS_CTOK	273.15
Velocity of light in a vacuum	m/s	PHYS_C	299_792_458.0
Planck's constant	-	PHYS_H	6.626_068_76e-34
Planck's constant divided by 2pi	-	PHYS_H_OVER_2_PI	PHYS_H/MATH_2_PI

It has been considered appropriate to include standard scaling factors from *YOCTO* (1.0e-24) to *YOTTA* (1.0e+24) in this package.

The user-definable constants have been placed in a package separately from these fundamental constants in a package called **MATERIAL_CONSTANTS**. The purpose of this package is to defined standard names for constants in common use, without necessarily defining a fixed value. This is especially useful for material characteristics that may be highly dependent on environmental conditions such as temperature. The list of defined material constants is given in Table 2.

TABLE 2
MATERIAL CONSTANTS

Constant	Unit	Name
Relative permittivity of silicon	-	PHYS_EPS_SI
Relative permittivity of silicon dioxide	-	PHYS_EPS_SIO2
Young's Modulus for silicon	Pa	PHYS_E_SI
Young's Modulus for silicon dioxide	Pa	PHYS_E_SIO2
Young's Modulus for polysilicon	Pa	PHYS_E_POLY
Poisson's Ratio for silicon	100 orientation	PHYS_NU_POLY
Density of Polysilicon	Kg/m^3	PHYS_RHO_POLY
Density of Silicon-Dioxide	Kg/m^3	PHYS_RHO_SIO2
Ambient Temperature	K	AMBIENT_TEMPERATURE
Ambient Pressure	Pa	AMBIENT_PRESSURE
Ambient Luminance		AMBIENT_LUMINANCE

Mixed Energy Domain Packages

VHDL-AMS requires that analog variables be defined using quantities. Quantities can be free (not with respect to a specific reference) or defined using a combination of a through and across variable. Different energy domains use

basic definitions of through and across variable types that are defined in these packages. For example, in the electrical domain, through variables may be defined using currents and across variables as voltages. The approach taken with the proposed standard is to define a package for each major energy system grouping (electrical, mechanical, thermal, etc). Each package defines the names of variable types, tolerances, units and symbols for use within a single energy domain.

The list of proposed packages is as follows:

ENERGY_SYSTEMS: This package contains basic definitions of energy and power. There is also a definition of a generic "no unit" system for control system modeling. The basic subtypes defined in this package are:

```
ENERGY
POWER
PERIODICITY
REAL_ACROSS
REAL_THROUGH
```

ELECTRICAL_SYSTEMS: This package contains the basic definitions for electrical and magnetic systems. The basic subtypes defined are as follows:

```
VOLTAGE
CURRENT
CHARGE
RESISTANCE
CAPACITANCE
MMF
FLUX
INDUCTANCE
FLUX_DENSITY
FIELD_STRENGTH
```

MECHANICAL_SYSTEMS: This package contains the basic definitions of rotational and translational mechanical systems. The basic subtypes defined are as follows:

```
DISPLACEMENT
FORCE
VELOCITY
ACCELERATION
MASS
STIFFNESS
DAMPING
MOMENTUM
COMPLIANCE
ANGLE
TORQUE
ANGULAR_VELOCITY
ANGULAR_ACCELERATION
MOMENT_INERTIA
ANGULAR_MOMENTUM
ANGULAR_STIFFNESS
ANGULAR_DAMPING
```

RADIANT_SYSTEMS: This package contains definitions for basic optical system modeling. The basic subtypes defined are as follows:

```
ILLUMINANCE
LUMINOUS_FLUX
LUMINOUS_INTENSITY
IRRADIANCE
```

THERMAL_SYSTEMS: This package contains the definitions for thermal systems modeling. The basic subtypes defined are as follows:

```
TEMPERATURE
HEAT_FLOW
THERMAL_CAPACITANCE
THERMAL_RESISTANCE
```

FLUIDIC_SYSTEMS: This package contains the definitions for fluidic (hydraulic) systems modeling. The basic subtypes defined are as follows:

```
PRESSURE
VFLOW_RATE
VOLUME
DENSITY
VISCOSITY
FRESISTANCE
FCAPACITANCE
INERTANCE
```

If the electrical systems package is taken as an example (proposed standard package is given in appendix A) it can be seen that the subtypes appropriate for electrical and magnetic systems are defined as type real and include voltage, current, mmf and flux. These are the variables that allow basic quantities to be defined with appropriate through and across variables. Also defined are useful derivative types including capacitance, resistance, inductance and charge. Also defined are the magnetic variables B (FLUX_DENSITY) & H (FIELD_STRENGTH) as they are commonly used in magnetic system modeling and simulation.

As can be seen from the package definition, each subtype has a corresponding UNIT and SYMBOL defined for post-processing, display and unit checking purposes and as well as the basic nature definitions for electrical and magnetic systems, vector types are also defined.

The final element defined in the package is the name of the default reference (in this case GROUND for electrical systems).

This pattern is repeated for the other packages, and these can be accessed through the working group web page [17].

Example

As an example of how the packages can be used in practice a simple electro-magnetic transformer is used to illustrate some of the key concepts involved. If a simple two winding transformer (schematic shown in figure 1) is implemented using a mixed –technology approach, the simple structure consists of two winding models (interfacing between the electrical and magnetic domains and a magnetic model of the core.

Using the models defined in this way, structural models of magnetic components can be built up that encompass the multiple domain capability in the VHDL-AMS language.

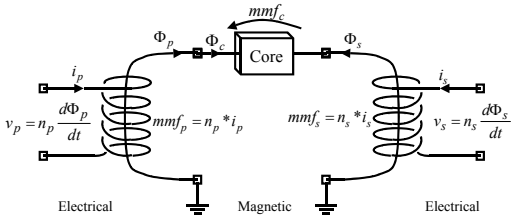

Figure 1: Electro-Magnetic model of transformer

Using this approach, the winding model is defined using the listing below:

```
1  use work.electrical_systems.all;
2
3  entity winding is
4     generic (r : real := 0.0;
5              n : real := 1.0);
6     port (
7       terminal ep,em : electrical;
8        terminal mp,mm : magnetic
9     );
10 end entity winding;
11
12 architecture simple of winding is
13  quantity h across f through mp to mm;
14  quantity v across i through ep to em;
15 begin
16   h == i*n;
17   v == - n*f'dot + i*r;
18 end architecture simple;
```

Note that in order to use the electrical and magnetic definitions, the electrical_systems package must be referenced using the use statement in line 1. Note that the terminals are defined with electrical or magnetic types and the resulting quantities (*v* & *i* in the electrical domain and *h* & *f* in the magnetic domains) have the correct units, symbols and types defined by the referenced package (see appendix A for the complete electrical_systems package listing).

The same approach is used for the core model as shown below:

```
1 use work.energy_systems.all;
2 use work.electrical_systems.all;
3
4 entity core_linear is
5     generic (ur   : real := 1.0;
6              len  : real := 1.0e-2;
7              area : real := 1.0e-4
8              );
9    port (terminal p,m : magnetic);
10 end entity core_linear;
11
12 architecture simple of core_linear is
13 constant mg : real
14       := PHYS_MU0*ur*area/len;
15 quantity mmf across f through p to m;
16 begin  -- architecture linear
17    assert len /= 0.0
18        report "len should not be 0!"
19       severity error;
20    f == mg * mmf;
21 end architecture simple;
```

As for the winding model the *electrical_systems* package must be referenced using the use statement in line2, but the *energy_systems* package must also be included because of the use of the **PHYS_MU0** constant.

Using this approach, the electrical and magnetic effects are modeled in the same overall system simulation as shown in figure 2. As can be seen from the figures, the flux and mmf values are correctly differentiated from the electrical voltage signal applied to the transformer. In this simple simulation a voltage source was applied to the primary of the transformer, with a resistive load applied to the secondary.

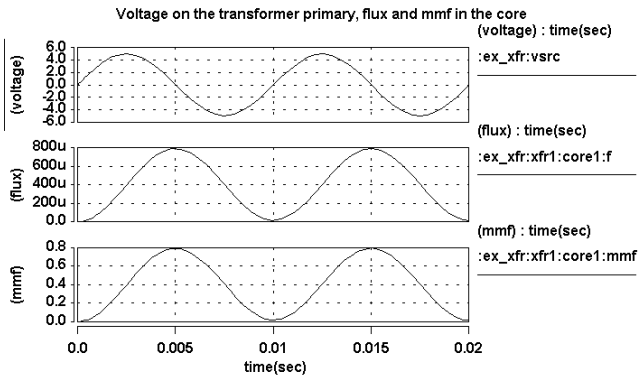

Figure 2: Electrical and Magnetic test waveforms

Conclusions

The emergence of VHDL-AMS as a standard modeling language for describing multiple energy domain systems has produced a requirement for a clear and unambiguous definition of constants and interface types for a wide variety of technology types.

The proposed IEEE standard 1076.1.1 as outlined in this paper seeks to address these issues and will provide a solid foundation for the development of portable and interoperable models.

Acknowledgement

It is only possible for standardization efforts to be successful with the hard work of many individuals who volunteer time

and effort as part of the process. An acknowledgement is therefore given to the members of the 1076.1.1 working group who have contributed in many ways to the development of this work.

References

[1] "Definition of Analog and Mixed-Signal extensions to IEEE Standard VHDL", IEEE Standard 1076.1-1999

[2] Christen, E. and Bakalar, K., "VHDL-AMS-a hardware description language for analog and mixed-signal applications", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Volume 46, Issue 10 , Oct. 1999, pp 1263 -1272

[3] Christen, E. and Bakalar, K., "VHDL 1076.1-analog and mixed-signal extensions to VHDL", *Design Automation Conference*, 1996, Proceedings EURO-DAC '96, 16-20 Sept. 1996, pp 556 -561

[4] Sabiro, SG, "Mixed-mode system design: VHDL-AMS", *Microelectronic Engineering*, 2000, Vol 54, Iss 1-2, pp 171-180

[5] Godambe NJ and Shi CJR, "Behavioral level noise modeling and jitter simulation of phase-locked loops with faults using VHDL-AMS", *JOURNAL OF ELECTRONIC TESTING-THEORY AND APPLICATIONS* 1998, Vol 13, Iss 1, pp 7-17

[6] Perkins AJ; Zwolinski M; Chalk CD; Wilkins BR,"Fault modeling and simulation using VHDL-AMS",*ANALOG INTEGRATED CIRCUITS AND SIGNAL PROCESSING* 1998, Vol 16, Iss 2, pp 141-155

[7] Vachoux A, "Analog and mixed-signal extensions to VHDL", *ANALOG INTEGRATED CIRCUITS AND SIGNAL PROCESSING* 1998, Vol 16, Iss 2, pp 185-200

[8] Boccuni I; Gulino R; Palumbo G, "Behavioral model of analog circuits for nonvolatile memories with VHDL-AMS", *ANALOG INTEGRATED CIRCUITS AND SIGNAL PROCESSING* 2002, Vol 33, Iss 1, pp 19-28

[9] Mongellaz B; Marc F; Milet-Lewis N; Danto Y, "Contribution to ageing simulation of complex analogue circuit using VHDL-AMS behavioural modelling language", *MICROELECTRONICS RELIABILITY* 2002, Vol 42, Iss 9-11, pp 1353-1358

[10] Sida M; Ahola R; Wallner D, "Bluetooth transceiver design and simulation with VHDL-AMS", *IEEE CIRCUITS & DEVICES* 2003, Vol 19, Iss 2, pp 11-14

[11] Vogels, M.; De Smedt, B.; Gielen, G, "Modeling and simulation of a sigma-delta digital to analog converter using VHDL-AMS", *2000 IEEE/ACM International Workshop on Behavioral Modeling and Simulation*, (2000), p 5-9

[12] Endemano A; Desmulliez MPY; Dunnigan M,"System level simulation of a double stator wobble electrostatic micromotor ", *SENSORS AND ACTUATORS A-PHYSICAL* 2002, Vol 99, Iss 3, pp 312-320

[13] Gibson D; Carter H; Purdy C, "The use of hardware description languages in the development of microelectromechanical systems ", *ANALOG INTEGRATED CIRCUITS AND SIGNAL PROCESSING* 2001, Vol 28, Iss 2, pp 173-180

[14] Bontoux P; O'Connor I; Gaffiot F; Letartre X; Jacquemod G, "Behavioral modeling and simulation of optical integrated devices", *ANALOG INTEGRATED CIRCUITS AND SIGNAL PROCESSING* 2001, Vol 29, Iss 1-2, pp 37-47

[15] Zhang TH; Cao F; Dewey AM; Fair RB; Chakrabarty K, "Performance analysis of microelectrofluidic systems using hierarchical modeling and simulation", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II-ANALOG AND DIGITAL SIGNAL PROCESSING* 2001, Vol 48, Iss 5, pp 482-491

[16] Voigt P; Schrag G; Wachutka G, "Microfluidic system modeling using VHDL-AMS and circuit simulation", *MICROELECTRONICS JOURNAL* 1998, Vol 29, Iss 11, pp 791-797

[17] http://mixedsignal.eleg.uark.edu/stdpkgs.html.

Appendix A: Electrical Systems Package Listing

```
package ELECTRICAL_SYSTEMS is
   -- subtype declarations
   subtype   VOLTAGE            is   REAL   tolerance
"DEFAULT_VOLTAGE";
   subtype   CURRENT            is   REAL   tolerance
"DEFAULT_CURRENT";
   subtype   CHARGE             is   REAL   tolerance
"DEFAULT_CHARGE";
   subtype    RESISTANCE       is    REAL    tolerance
"DEFAULT_RESISTANCE";
   subtype    CAPACITANCE    is    REAL    tolerance
"DEFAULT_CAPACITANCE";
   subtype  MMF                 is  REAL  tolerance
"DEFAULT_MMF";
   subtype  FLUX                is  REAL  tolerance
"DEFAULT_FLUX";
   subtype    INDUCTANCE       is    REAL    tolerance
"DEFAULT_INDUCTANCE";

   -- attribute declarations
   -- Use of UNIT to designate units
   attribute UNIT of VOLTAGE         : subtype is
"Volt";
   attribute UNIT of CURRENT         : subtype is
"Ampere";
   attribute UNIT of CHARGE          : subtype is
"Coulomb";
   attribute UNIT of RESISTANCE      : subtype is
"Ohm";
   attribute UNIT of CAPACITANCE     : subtype is
"Farad";
   attribute UNIT of MMF             : subtype is
"Ampere";
   attribute UNIT of FLUX            : subtype is
"Weber";
   attribute UNIT of INDUCTANCE      : subtype is
"Henry";
   attribute UNIT of FLUX_DENSITY    : subtype is
"Tesla";
   attribute UNIT of FIELD_STRENGTH  : subtype is
"Amperes per meter";

   attribute SYMBOL of VOLTAGE       : subtype is
"V";
   attribute SYMBOL of CURRENT       : subtype is
"A";
   attribute SYMBOL of CHARGE        : subtype is
"C";
   attribute SYMBOL of RESISTANCE    : subtype is
"Ohm";
   attribute SYMBOL of CAPACITANCE   : subtype is
"F";
```

```
   attribute SYMBOL of MMF           : subtype is
"A";
   attribute SYMBOL of FLUX          : subtype is
"W";
   attribute SYMBOL of INDUCTANCE    : subtype is
"H";
   attribute SYMBOL of FLUX_DENSITY  : subtype is
"T";
   attribute SYMBOL of FIELD_STRENGTH : subtype is
"A/m";

   -- nature declarations
   nature ELECTRICAL is
      VOLTAGE        across
      CURRENT        through
      ELECTRICAL_REF reference;
   nature ELECTRICAL_VECTOR is array (NATURAL range
<>) of ELECTRICAL;

   nature MAGNETIC is
      MMF          across
      FLUX         through
      MAGNETIC_REF reference;
   nature MAGNETIC_VECTOR is array (NATURAL range
<>) of MAGNETIC;

   -- vector subtype declarations
   subtype     VOLTAGE_VECTOR                    is
ELECTRICAL_VECTOR'across;
   subtype     CURRENT_VECTOR                    is
ELECTRICAL_VECTOR'through;
   subtype   CHARGE_VECTOR           is   REAL_VECTOR
tolerance "DEFAULT_CHARGE";
   subtype    RESISTANCE_VECTOR    is    REAL_VECTOR
tolerance "DEFAULT_RESISTANCE";
   subtype    MMF_VECTOR                        is
MAGNETIC_VECTOR'across;
   subtype    FLUX_VECTOR                       is
MAGNETIC_VECTOR'through;
   subtype    INDUCTANCE_VECTOR    is    REAL_VECTOR
tolerance "DEFAULT_INDUCTANCE";

   -- attributes of vector subtypes
   attribute UNIT of VOLTAGE_VECTOR       : subtype
is "Volt";
   attribute UNIT of CURRENT_VECTOR       : subtype
is "Ampere";
   attribute UNIT of CHARGE_VECTOR        : subtype
is "Coulomb";
   attribute UNIT of RESISTANCE_VECTOR    : subtype
is "Ohm";
   attribute UNIT of CAPACITANCE_VECTOR   : subtype
is "Farad";
   attribute UNIT of MMF_VECTOR           : subtype
is "Ampere";
   attribute UNIT of FLUX_VECTOR          : subtype
is "Weber";
   attribute UNIT of INDUCTANCE_VECTOR    : subtype
is "Henry";

   attribute SYMBOL of VOLTAGE_VECTOR     : subtype
is "V";
   attribute SYMBOL of CURRENT_VECTOR     : subtype
is "A";
   attribute SYMBOL of CHARGE_VECTOR      : subtype
is "C";
   attribute SYMBOL of RESISTANCE_VECTOR  : subtype
is "Ohm";
   attribute SYMBOL of CAPACITANCE_VECTOR : subtype
is "F";
   attribute SYMBOL of MMF_VECTOR         : subtype
is "A";
   attribute SYMBOL of FLUX_VECTOR        : subtype
is "W";
   attribute SYMBOL of INDUCTANCE_VECTOR  : subtype
is "H";

   alias GROUND is ELECTRICAL_REF;
end package ELECTRICAL_SYSTEMS;
```