DOLPHIN
MEDAL

MACROS

IST-2001-34714 WP4 – Simulation Environment

D4.2
VHDL-AMS environment with improved simulation support

Version	Date	Approved by
V1.0	03-mar-2003	Gilles DEPEYROT

Major version: $V_{i.0}$
Minor version: $V_{i.j}$ with $j \neq 0$

DOLPHIN
INTEGRATION
39, avenue du Granier
B.P. 65 ZIRST
38242 Meylan France
tel : +(33) 4 76 41 10 96
fax : +(33) 4 76 90 29 65
web : http://www.dolphin-integration.com
e-mail : medal@dolphin-integration.com

1 Introduction

SMASH™ 5.0.1 is the official release of MACROS deliverable **D4.2**.

Deliverable **D4.2** corresponds to work in **WP4 Simulation Environment** on the following tasks:
- **T4.1**: VHDL-AMS language and simulation improvements
- **T4.2**: VHDL-AMS simulation optimizations targeted at MEMS simulation

Deliverable **D4.2** also includes part of the work planned for deliverable **D4.7**, mainly the support of small-signal simulations in VHDL-AMS.

SMASH™ 5.0.1 should provide an industrial simulator with the necessary improvements needed for multi-domain simulations:
- Handling of discontinuities using the VHDL-AMS break statements,
- Support of tolerances needed to define and simulate required non-electrical variable types with absolute values that differ very much from common electrical values,
- Support of arrays, matrices and vectors, for models of 2-D and 3-D behavior with more concise equations close to the mathematical equations.

2 T4.1: Improved language support

Deliverable D4.2 implements VHDL-AMS language and simulation support in order to provide missing IEEE Std 1076.1-1999 features required by new MEMS models. The work accomplished in this task for deliverable D4.2 covers the extensive support of tolerances for multi-domain simulations, with VHDL-AMS parser improvements as well as language support improvements.

The language support improvements in deliverable D4.2 are detailed with examples demonstrating how to use the language features.

2.1 Introduction to tolerances

With multi-domain simulations, for example needed for MEMS simulation (electronic and mechanic domain), values of completely different orders of magnitude can occur in the simulated models. In capacitive pressure sensors for instance, pressures of 10^{e5} Pascal and capacity variations of 10^{e-15} Farad are usual. In order for the modeling to be done with the proven MKS unit system, this range of numbers makes excessive demands on most pure circuit simulators which do not generally take diverse tolerances into account but only provide a basic unique set of tolerance and accuracy settings. VHDL-AMS offers a way out with the possibility to use tolerance groups with individual physical quantities (with units) and its own range of values.

2.2 Tolerances in VHDL-AMS

Tolerance groups in the VHDL-AMS language definition standard are defined as names in the form of strings. The use of these names and, in particular, the mapping of these names to real tolerance values used by the simulator are vendor and implementation dependent.

IEEE Std 1076.1-1999 (12.6.6 The analog solver – note 3 – page 179)

> The criteria determining sufficiency (as used in the phrases “sufficiently close to”, “sufficiently greater than” and so on) are intended to be implementation-dependent and may vary from time to time and quantity to quantity to effect tradeoffs between simulation time and accuracy of the result.

Furthermore, the VHDL-AMS language definition standard states that the result type of the Q’DOT and Q’INTEG attributes is of the same type as Q. This means that the Q’DOT (respectively Q’INTEG) and Q quantities belong to the same tolerance group. However, Q’DOT (respectively Q’INTEG) and Q values are obviously not in the same value ranges. *If the same tolerance is applied to Q’DOT (respectively Q’INTEG) and Q, unavoidable convergence problems may occur.* Therefore, SMASH uses specific tolerances for the

Q'DOT (respectively Q'INTEG) quantity. The mapping of the integral and derivative quantities to respective tolerances groups in specified in disciplines related settings.

In SMASH, the tolerance group names are mapped to real values via:

- A default disciplines related settings file containing the default values of the tolerance groups as well as the mapping of integral and derivative quantities to respective tolerance groups (see § 2.4.3).
- A simulator directive allowing the user to override the default values of the standard tolerance groups as well as to define the values of circuit specific tolerance groups (see § 2.4.4).

2.3 Tolerance groups in VHDL-AMS statements

This section presents the different uses of tolerance group names in VHDL-AMS statements.

2.3.1 Tolerance in subtype declarations

The tolerance group of a subtype is a string value that may used by a model to group quantities that have similar requirements concerning the accuracy and tolerance of the values determined by the quantities.

Example:

```
Subtype T1 is REAL tolerance "TOL_T1";
```

The value of the string expression "`TOL_T1`" will be the *tolerance group* of each scalar of the subtype.
So each quantity which will be declared with as subtype T1, will have the tolerance "`TOL_T1`".

2.3.2 Tolerances in quantity declarations

Example:

```
Quantity Q1 :T1 tolerance "TOL_Q1";
Quantity Q2, Q3 :T1;
```

The tolerance of the quantity Q1 is "`TOL_Q1`".
The tolerance of the quantities Q2 and Q3 is "`TOL_T1`", i.e. the tolerance group of the quantities subtype.

2.3.3 Tolerances in simple simultaneous statements

Syntax:

```
[Label:] simple_expression == simple_expression [tolerance_aspect];
```

Examples:

```
EXP1: Q1  == 1.0 tolerance "TOL_EXP1";
EXP2: Q1 == Q2 + Q3 ;
EXP3: Q1 + Q2 == Q3 ;
```

If the statement has no explicit tolerance aspect, it is necessary to determine its associated ***tolerance quantity***. Indeed, if the simple expression on the left hand side of the "==" sign is a name that denotes a quantity, then this quantity is the tolerance quantity of the simple simultaneous statement. Otherwise, if the simple expression on the right hand side of the "==" sign is a name that denotes a quantity, then this quantity is the tolerance quantity of the simple simultaneous statement.

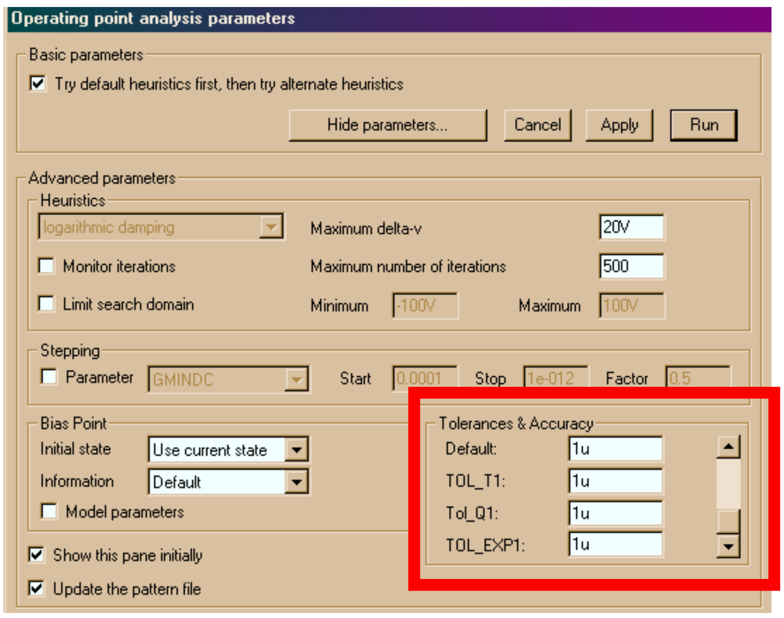
Furthermore, theoretically it is an error if a simple simultaneous statement that does not include a tolerance aspect has no associated tolerance quantity. However, SMASH accepts this kind of syntax and attributes a `DEFAULT_TOLERANCE` to the statement (§ 2.4.4 Disciplines.ini file).

2.4 Tolerance value settings

The tolerances used in the design or declared by the `.TOLERANCE` directive appear in the 'Tolerance & Accuracy' fields of the Operating Point and Transient dialog boxes.

The tolerance names displayed in the dialog correspond to the tolerances effectively used in the currently loaded circuit. The dialogs are dynamically updated to display only the useful tolerances.

2.4.1 Operating Point Analysis Parameters



Operating point analysis parameters

Basic parameters
[✓] Try default heuristics first, then try alternate heuristics

Advanced parameters

Heuristics
logarithmic damping
Maximum delta-v: 20V
[ ] Monitor iterations
Maximum number of iterations: 500
[ ] Limit search domain
Minimum: -100V
Maximum: 100V

Stepping
[ ] Parameter: GMINDC
Start: 0.0001
Stop: 1e-012
Factor: 0.5

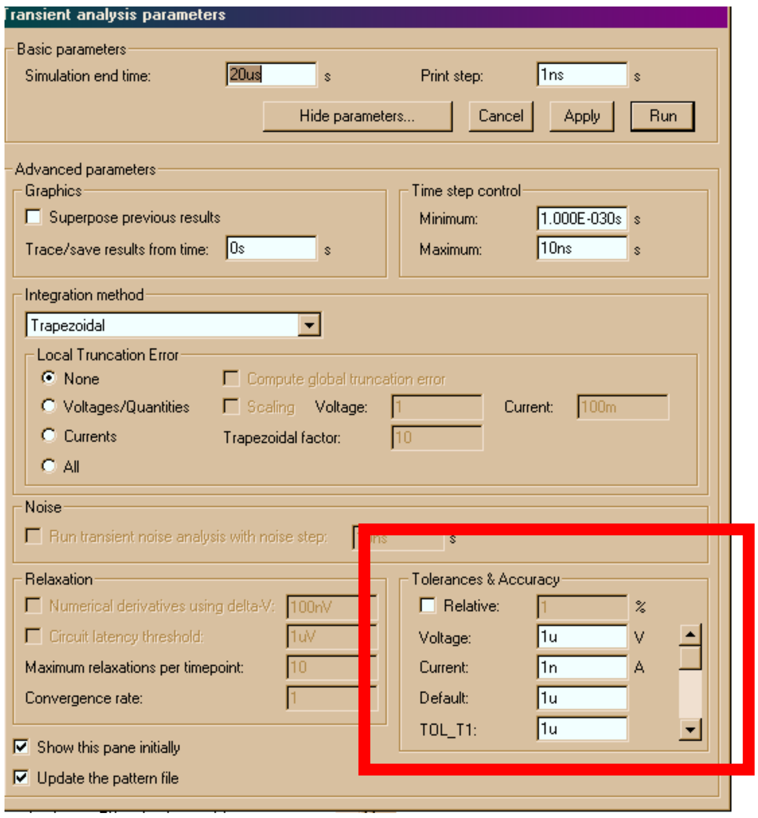
Bias Point
Initial state: Use current state
Information: Default
[ ] Model parameters

Tolerances & Accuracy
Default: 1u
TOL_T1: 1u
Tol_Q1: 1u
TOL_EXP1: 1u

[✓] Show this pane initially
[✓] Update the pattern file

Figure 1: Operating Point Analysis dialog box

2.4.2 Transient Analysis Parameters



Transient analysis parameters

Basic parameters
Simulation end time: 20us s
Print step: 1ns s

Advanced parameters

Graphics
[ ] Superpose previous results
Trace/save results from time: 0s s

Time step control
Minimum: 1.000E-030s s
Maximum: 10ns s

Integration method
Trapezoidal

Local Truncation Error
(●) None
( ) Voltages/Quantities
( ) Currents
( ) All
[ ] Compute global truncation error
[ ] Scaling Voltage: 1 Current: 100m
Trapezoidal factor: 10

Noise
[ ] Run transient noise analysis with noise step: 1ns s

Relaxation
[ ] Numerical derivatives using delta-V: 100nV
[ ] Circuit latency threshold: 1uV
Maximum relaxations per timepoint: 10
Convergence rate: 1

Tolerances & Accuracy
[ ] Relative: 1 %
Voltage: 1u V
Current: 1n A
Default: 1u
TOL_T1: 1u

[✓] Show this pane initially
[✓] Update the pattern file

Figure 2: Transient Analysis dialog box

2.4.3 Directive

In the pattern file, the `.TOLERANCE` directive is used to set a tolerance value:

```
.TOLERANCE TOL_Q1 1u
```

The directives are automatically added to the pattern file by the dialog boxes if the user changes the values.

2.4.4 Disciplines.ini file

All the tolerances which are declared in the IEEE VHDL-AMS draft packages are described in the file '`discipline.ini`' in the form of the below example:

```
[DEFAULT_VOLTAGE]
name  = Voltage
unit  = V
value = 1u
dot   = DEFAULT_DOT
integ = DEFAULT_FLUX
```

The '`name`' and '`unit`' fields are used for the tolerance display in the dialog boxes.
The '`dot`' field (respectively '`integ`' field) indicates the tolerance to use for the derivative (respectively integral) of a quantity which has this tolerance. So, in this way, the below tolerance are defined.

```
[DEFAULT_DOT]
name  = DefaultDerivative
unit  =
value = 1u
dot   = DEFAULT_DOT
integ = DEFAULT_INTEG

[DEFAULT_INTEG]
name  = DefaultIntegral
unit  =
value = 1u
dot   = DEFAULT_DOT
integ = DEFAULT_INTEG
```

Concerning the quantities or expressions which do not have an explicit tolerance, the '`DEFAULT_TOLERANCE`' is defined:

```
[DEFAULT_TOLERANCE]
name  = Default
unit  =
value = 1u
dot   = DEFAULT_DOT
integ = DEFAULT_INTEG
```

2.5 Use of tolerance values in SMASH

2.5.1 Absolute convergence check

The absolute convergence check concerns the simple simultaneous statement as follows:

```
If (Abs(Expression_Residual) < Expression_Tolerance)
   Convergence = OK
```

The ‘`Expression_Residual`’ is determined by the matrix solving. The ‘`Expression_Tolerance`’ determination is described in § 2.3.3Tolerances in simple simultaneous statements.

2.5.2 Relative convergence check

In the case of a relative convergence check the below equation is used for all the design quantities:

```
If (abs(Quantity_Value i -Quantity_Value i-1) < Quantity_Tolerance)
   Convergence = OK
```

Where `i` represents the iteration index.

2.5.3 LTE computing

The LTE is obtained by:

```
LTE = Quantity_Tolerance + Relative_Tolerance * Global
```

Where the ‘`Global`’ and ‘`Relative_tolerance`’ terms are specified in the transient parameters box.

2.5.4 Partial derivative of expression computing

The partial derivative of a simple simultaneous statement expression `f(x)` in rapport with a quantity contained in this expression, which is necessary for the matrix filling, uses the tolerance of this quantity:

```
F'(x) == f(x) - f(x + x_tolerance) / x_tolerance.
```

2.6 Future work

The work remaining to be done in task T4.1 to improve language support includes:

1. VHDL-AMS parser improvements and improved language support to provide language features not yet available in this release
   - Vector & record support in some syntax cases not yet covered by the VHDL-AMS parser
   - AMS attributes or uses no yet supported in the VHDL-AMS elaborator:
     - T’REFERENCE
     - T’CONTRIBUTION
     - T’TOLERANCE
     - Q’TOLERANCE
     - Composite types as prefixes of attributes
   - Language constructs not yet supported by the VHDL-AMS elaborator:
     - Generate statements
     - Simultaneous procedural statements
     - Interface quantity declaration modes

2. Improvements to multi-domain simulation issues and, in particular, support of multi-domain tolerances
   - Adjustments related to the use of tolerances
   - Improvements from benchmarking results as well as from partner and user feedback

3 T4.2: Simulation optimizations & improvements

The implementation of improved language support, and in particular the support of matrices and vectors, had a negative impact on simulation speed of VHDL-AMS models in SMASH™ 4.4 corresponding to deliverable D4.1. Deliverable D4.2 implements first step optimizations related to the support of matrices and vectors and provides drastic reductions in circuit load time.

3.1 Introduction

Simulation time cannot be reduced to the simple time it takes to run the simulation. The simulation time is often the main source of benchmarking. However, other times need to be taken into account and namely:

- Parsing of the VHDL-AMS source files
- Generation and compilation of the circuit description
- Elaboration and simulation of the compiled circuit

Deliverable D4.2 implements major improvements to the generation and compilation infrastructure in SMASH in order to provide faster loading of circuit descriptions, thereby reducing designer time lost when working on developing models.

3.2 Built-in standard packages

Deliverable D4.2 provides integrated precompiled built-in packages corresponding to VHDL and VHDL-AMS IEEE standard and draft[1] standard packages. The integration of these packages as precompiled libraries provides both:

- Faster loading of circuit descriptions using these standard packages since no compilation of these packages is needed
- Better interoperability with other EDA tools using the standard packages[2] thereby reducing the designer time required to switch from one tool to another.

3.3 Circuit load time optimizations

Two different approaches are generally used to obtain machine executable code from circuit descriptions:

- On the fly p-code generation
- Intermediate C/C++ code generation and compilation which is then dynamically loaded into the simulator

SMASH™ uses the second method and generates intermediate C/C++ code which is then compiled using standard C/C++ compilers. The default bundled compiler is the GNU C/C++ compiler.

Whereas previous releases of SMASH generated a single dynamic library loaded into the simulator for a given circuit description, deliverable D4.2 has concentrated on generating and compiling separate dynamic libraries for each design unit in the circuit description. This major change provides faster compilation of individual design units and faster loading time for the overall circuit.

3.4 Small signal simulation

The small-signal analysis is particularly useful to study the stability of closed-looped systems and reduce simulation time by replacing costly non-linear transient simulations with much faster linear small-signal simulations.

Deliverable D4.2 provides small-signal analysis for VHDL-AMS circuit descriptions. Frequency based simulations were originally scheduled for delivery in D4.7. However, the need and demand for small-signal

[1] Deliverable D4.2 includes the latest multiple energy domain draft IEEE packages (1.3, 30 Sep 2002 - FDL 2002 revisions).

[2] This is less true for the draft standard packages where different releases of the draft packages may be incompatible. In this case, different releases includes in different EDA tools cause compatibility problems when switching from one tool to another.

simulation, in particular in the mechanical and micro-mechanical domains, required that this analysis be available before M33.

3.5 Future work

The work remaining to be done in task T4.2 to reduce simulation time and provide simulation improvements includes:

1. Circuit compilation time
   - Generated code optimizations in order to reduce compilation time. The simpler the code is, the faster the compilation will be.

2. Circuit elaboration and simulation time
   - Generated code optimization in order to reduce simulation time. Reducing the complexity of the generated code will provide more optimized compilation and, therefore, faster simulations.
   - Jacobian matrix filling optimizations both in time domain and frequency domain in order to reduce elaboration and simulation time. The matrix is filled at every simulation step and optimizations made to the matrix filling routines will provide faster simulations.
   - Matrix partitioning to speed up analog resolution.

3. Implementation of the noise analysis which is not yet supported by SMASH for VHDL-AMS descriptions.

4 Table of Contents

1 INTRODUCTION 3

2 T4.1: IMPROVED LANGUAGE SUPPORT 3

2.1 INTRODUCTION TO TOLERANCES 3
2.2 TOLERANCES IN VHDL-AMS 3
2.3 TOLERANCE GROUPS IN VHDL-AMS STATEMENTS 4
2.3.1 TOLERANCE IN SUBTYPE DECLARATIONS 4
2.3.2 TOLERANCES IN QUANTITY DECLARATIONS 4
2.3.3 TOLERANCES IN SIMPLE SIMULTANEOUS STATEMENTS 4
2.4 TOLERANCE VALUE SETTINGS 4
2.4.1 OPERATING POINT ANALYSIS PARAMETERS 5
2.4.2 TRANSIENT ANALYSIS PARAMETERS 5
2.4.3 DIRECTIVE 6
2.4.4 DISCIPLINES.INI FILE 6
2.5 USE OF TOLERANCE VALUES IN SMASH 6
2.5.1 ABSOLUTE CONVERGENCE CHECK 6
2.5.2 RELATIVE CONVERGENCE CHECK 7
2.5.3 LTE COMPUTING 7
2.5.4 PARTIAL DERIVATIVE OF EXPRESSION COMPUTING 7
2.6 FUTURE WORK 7

3 T4.2: SIMULATION OPTIMIZATIONS & IMPROVEMENTS 8

3.1 INTRODUCTION 8
3.2 BUILT-IN STANDARD PACKAGES 8
3.3 CIRCUIT LOAD TIME OPTIMIZATIONS 8
3.4 SMALL SIGNAL SIMULATION 8
3.5 FUTURE WORK 9

4 TABLE OF CONTENTS 10