A New VHDL-AMS Simulation Framework in Matlab™

M. Zorzi, N. Speciale, G. Masetti
DEIS, University of Bologna
Viale Risorgimento 2 40136 Bologna, Italy
phone: +390512093777 fax: +390512093779
e-mail: mzorzi@deis.unibo.it

Abstract

In this paper we present SAMSA, a new tool for the simulation of VHDL-AMS systems in Matlab™. The goal is the definition of a VHDL framework in which analog/digital systems can be designed and simulated and new simulation techniques can be studied, exploiting both the powerful Matlab™ functions and Toolboxes.

Keywords: VHDL-AMS, Circuit simulation, Behavioral Modeling.

1 Introduction

Although analog blocks typically constitute only a small fraction of the components in modern mixed-signal ICs and systems-on-a-chip (SoC) designs, there is a strong need for CAD tools in order to increase the design speed and productivity, improving the quality of analog integrated circuits for telecommunications, consumer, computing and automotive applications. This claim is mainly due to the increase level of integration available in silicon technology and the growing requirement for digital systems to communicate with the continuous-valued external world.

The development of analog and mixed-signal hardware description languages [1][2][3] was just intended to provide a unifying tool to link the various analog design automation tasks in a coherent framework that supports a more structured design methodology, from the first rough idea to the manufacturing stage. These languages provide a link between the analog and digital domains and allow for the definition of higher levels of abstraction to describe and simulate analog circuits, by using model paradigms and languages from the digital world. By so doing, macro, behavioral and functional simulation levels have been developed for analog circuits besides the well known circuit level and new simulators have been introduced in modern industrial design practice to cope with systems containing a mix of analog blocks described at different levels and in different domains.

To allow an easy exchange of these models across different simulators and users, standardized hardware description languages are used to describe these higher-level models and to provide a consistent way of representing and sharing design information across different design tasks and hierarchy, linking the various tools in a global analog CAD system.

Modern electronic circuits are characterized by a large number of components which can have different natures, such as MEMS systems where mechanical and electrical devices are mixed together. The key to managing this increased designed complexity while meeting the shortening time-to-market factor is certainly the use of computer-aided design and verification systems.

In this work we present *SAMSA*, a new tool developed to make possible VHDL-AMS simulations in Matlab™ [4], a powerful scientific tool for numerical analysis, along with its associated complete set of Toolboxes. The purpose of *SAMSA* is to have a sole VHDL framework where analog/digital systems can be designed and simulated and new simulation techniques can be studied as well. This is possible because the solver can be changed, or the user can implement and test its own.

The paper is organized as follows: in the next Section we introduce *SAMSA*, giving a brief description of the most important aspects of this tool. Section 3 describes two examples: a memory cell and a micro-electromechanical system. Finally, conclusions will be drawn in Section 4.

2 System Description

2.1 Overview

The simulation tool *SAMSA* is schematically described by the diagram in Figure 1. The tool is basically composed by a Java™ [5] compiler and a solver.

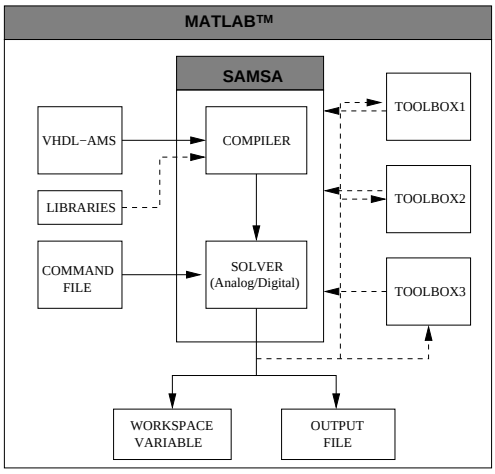

Figure 1: *SAMSA* general architecture and dependences with other Matlab™ Toolboxes.

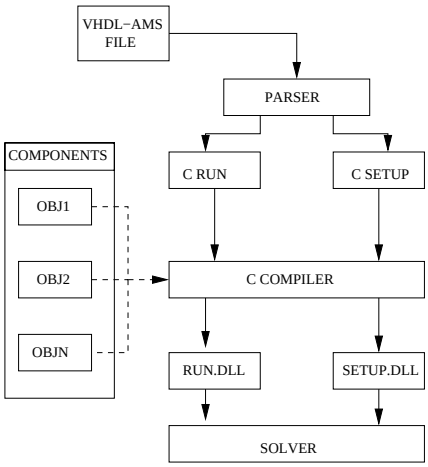

Figure 2: *SAMSA* compiler structure.

The VHDL-AMS file can be compiled and linked with models defined in an external library and Toolboxes functions, and then simulated using a command file.

Using a VHDL compiler written in Java™ and loaded as a Matlab™ java class, two functions are generated from the VHDL-AMS file:

1. a setup-function, used to initialize the simulated system;

2. a run-function, which is called during the simulation to update some workspace vectors and variables.

These functions are compiled with a C compiler, linked with other objects (i.e. instanced entities) and, finally, two dynamically linked functions are generated.

The system simulation is a three step process: first *SAMSA* reads a spice-like command file, which describes the simulation that should be performed, the variables that should be printed and some other options. Then the solver calls the setup-function for the specified design unit, and creates a structure which describes the system to be simulated in the Matlab™ workspace. Finally the run-function is called, and an output is produced as a workspace variable, or a file in the work directory. Output data can be post-processed or used within a particular Toolbox, making the system very flexible.

2.2 The compiler

We chose to develop the *VHDL-AMS compiler* depicted in Figure 2 in Java™ to exploit the capability of directly load java classes into the workspace. After the file parsing and symbols loading from included libraries, the design unit is analyzed. Two C functions are generated and then compiled as mex functions using the Matlab™ default C compiler, but the user can choose any C compiler (for example GCC from GNU). After the compilation two dynamically linked functions will be available in workspace. We use C language because these functions performs a numerically expensive task, and a purely Matlab™ implementation would be very slow. If the VHDL-AMS system have instances of different components, these will by statically linked into the run- and setup-function as shown in Figure 2.

2.3 The analog solver

The *analog solver* can solve generic problems of the form

$$G(t, y, y') = 0$$

The solver is a function call of the form $f(y_0, y'_0, C_o, F, I_a)$, where y_0 and y'_0 are the initial conditions vectors for the system of DAE being solved, C_o is an array of control options, F is the pointer to the run-function and I_a is used as temporary array for sharing information. Several control options can be set by the user: the relative tolerance, the max step allowed during transients, the simulation initial and end time. The interface allows the user to implement its own solver, making the tool very flexible and defining a framework where new simulation algorithms can be tested, leaving unchanged the way the system is

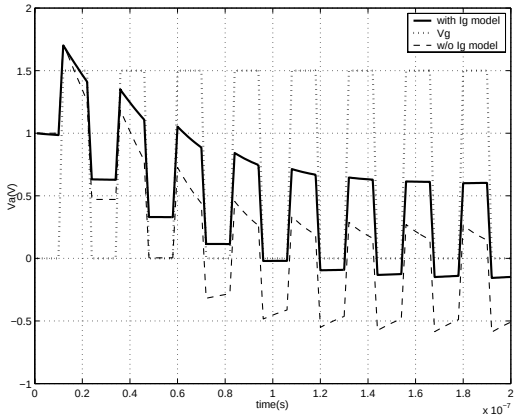

Figure 3: Comparison between transient simulation results of a memory cell with and without gate tunneling current model.

described. To perform this task, it is only necessary to understand how the setup- and run-function work.

The setup-function includes three major blocks:

1. the **system tree**, where the hierarchy information about the simulated system, free and branch quantities related to the different component instances are stored;

2. the **simulation workspace**, used to store the value of the system unknowns;

3. the **simulation flags**, to keep track of the simulation behavior, and some other variables like the `gmin` used during electrical simulation.

The run-function takes the system structure as input and modifies the simulation workspace block according to the value of simulation flags. During this phase, the user should only know what happens to the system structure and no extra informations are needed.

Finally, a basic library of components was defined: presently it includes mechanical and passive elements together with advanced MOST device models, as BSIM4, EKV and MM11.

3 Implementation examples

Exhaustive testing was performed by using test models [6] and different kind of systems were successfully simulated.

As an example, in the following paragraphs we show results concerning two different applications: an advanced compact model and a MEMS structure.

3.1 Gate tunneling current effects

Figure 3 reports electrical simulation results of a memory cell based on a advanced MOS model transistor [7]. The purpose of this simulation was to observe the effects of different gate tunneling current models on the stored voltage waveform in a memory cell.

As show in Figure, the application of a square impulse to the gate when $I_G \neq 0$ produces remarkable effects. Internal node remains at a steady voltage due to the current injection from the gate (straight line) while it gradually decreases as a function of time when gate current is ignored (dashed line).

3.2 A MEMS resonator

A beam, used to build a low frequency MEMS resonator [8] in an RF system was also successfully simulated. The beam model depicted in Figure 4 has six

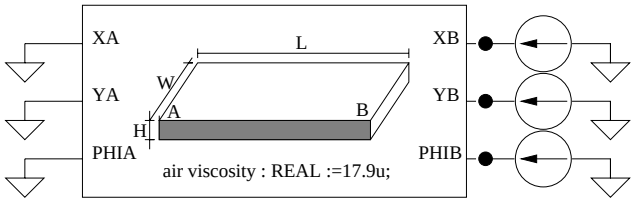

Figure 4: Low frequency MEMS resonator diagram, showing beam terminals and applied force stimuli.

terminals: a displacement and a force as across and through quantity for XA, YA, XB, YB terminals respectively, angle displacement and torque as across and through for $PHIA$ and $PHIB$ terminals. These terminals are used to represent the position at the edges A and B and to apply a force or a torque to the beam.

In the simulation we performed (Figure 5), the beam was anchored at the A edge, and a time dependent force was applied to the B edge. As the result of this stimulus, the beam starts to oscillate, with a motion that depends on different parameters, as the beam width and length and the air viscosity.

The simulation command file is shown in Figure 6: with the **UNIT** keyword we define the entity name and architecture of the system that we simulate; **LIBRARY** defines the path where unit file is placed; **TRAN** keyword tells the solver to perform a transient simulation, with a start and stop time and a print step size.

It is also possible to assign a value to the model parameters using the word **SET**, like the magnitude of the force applied and the position of the beam, and

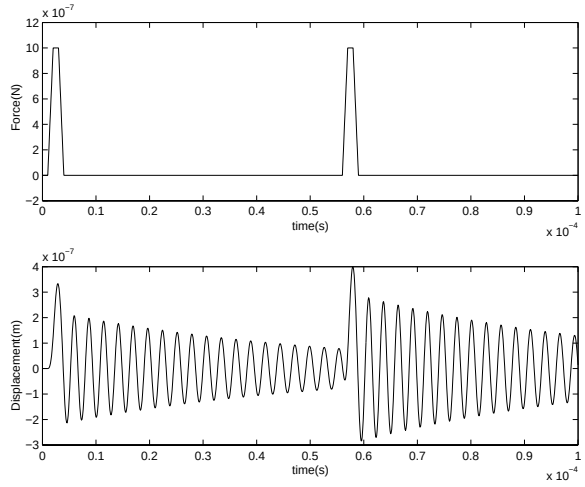

Figure 5: Simulation results showing the beam displacement due to applied force impulse.

```
* Unit name
UNIT beam behav
*       entity architecture

* Simulation type
TRAN 0     10e-5 0.01e-5
*         Start Stop  Print

* Library name, where the Unit is stored
LIBRARY work

* Setup parameters
* force_angle parameters
SET beam.f1.force = 1e-6
SET beam.f1.t_delay = 1e-6
SET beam.f1.angle = 90

* beam parameters
SET beam.t1.angle = 0
SET beam.t1.yb = 0

* trace output
TRACE beam.T1.yb beam.T1.xb beam.T1.txb
          beam.T1.tyb
```

Figure 6: Command file for the simulation of a beam with *SAMSA*.

finally the **TRACE** keyword save the specified variables to the output file or to the Matlab™ workspace. The beam structure and its physical model were entirely written in VHDL-AMS.

4 Conclusions

The introduction of behavioral description languages provides a unified tool to link different design tasks and to allow an easy exchange of information across different simulators and users. Within this scenario we propose a software for VHDL-AMS system simulation in Matlab™. The presented program is currently on-going and is part of a larger project devoted to device modeling [9] [10] and mixed-mode simulation.

We plan to extend the proposed approach to develop full VHDL-AMS models and to allow for both digital and non-digital functions to be modeled within Matlab™ environment or in an industrial IC design flow.

Acknowledgments

The authors would like to thank Dr. R. Gaddi and Dr. F. Mancarella for help, work and suggestions about the MEMS resonator and advanced compact models code.

5 References

[1] *"IEEE Standard VHDL Analog and Mixed-Signal Extensions"*, IEEE Std 1076.1-1999.

[2] *"SPECTRE HDL Reference"*, Cadence Design Systems, 1998.

[3] *"Verilog Reference Manual"*, Cadence Corporation.

[4] *"Matlab Reference Documentation"*, Ver. 6, Mathworks.

[5] *"Java 2 SDK, Standard Edition Documentation"*, Ver. 1.3.1, Sun Microsystems, 2002.

[6] `http://www.vhdl-ams.de`.

[7] C. H. Choi, K. H. Oh, J. S. Goo, Z. Yu and R. W. Dutton *"Direct Tunneling Current Model for Circuit Simulation"*, Proc. of IEDM 99, pp: 735-738.

[8] J. E. Vandemeer, M. S. Kraus, G. K. Fedder *"Hierarchical Representation and Simulation of Micromachined Inertial Sensors"*, Proc. of MSM98.

[9] M. Zorzi, N. Speciale, G. Masetti *"Automatic Embedding of a Ferroelectric Capacitor Inside the Circuit Simulator Eldo"*, Proceedings of BMAS 2001, Santa Rosa, California, 10-12 october 2001, pp. 97 -101.

[10] M. Zorzi, F. Franzè, N. Speciale *"I.M.A.GE.: a new CAD Tool for Device Modeling in Spice"*, Proc. of ECCTD01, 28-31 August 2001, Espoo, Finland, pp 241–244.