On the Simulation Performance of Contemporary AMS Hardware Description Languages

Rajeev Narayanan, Naeem Abbasi, Mohammed Zaki, Ghiath Al Sammane, and Sofiène Tahar
Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada
Email: {r_naraya, n_ab, sammane, mzaki,tahar}@ece.concordia.ca

Technical Report

April, 2008

Abstract

Mixed-Signal extensions to VHDL, Verilog, and SystemC languages have been developed in order to provide a unifying environment for the modeling and verification of Analog and Mixed Signal (AMS) designs at different levels of abstraction. In this paper, we model the behavior of a set of benchmark designs in VHDL-AMS, Verilog-AMS and SystemC-AMS and compare the simulation performance with HSPICE. The various experimental results observed for the benchmark circuits show the superiority of VHDL-AMS and Verilog-AMS against SystemC-AMS and HSPICE in terms of simulation run-times at lower level of abstraction.

1 Introduction

Verification of Analog and Mixed Signal (AMS) circuits and systems is a challenging task because it requires both an accurate model of the system and an efficient method of simulation. For a simulator, a tradeoff exists between accuracy of the results and the simulation speed.

Traditionally, circuit simulators are used to simulate and analyze the AMS design described as a netlist in SPICE. Circuit simulators face a bottleneck of long simulation run-times for complex circuits. An alternate approach would be to capture the behavior of AMS designs at higher level of abstraction using AMS hardware description languages (HDLs). This approach brings down the simulation run-times, but is less accurate compared to SPICE simulation. For a tradeoff between accuracy and run-time, designers can look at modeling AMS designs at higher levels of abstraction. This paper compares the performance of different AMS HDLs in terms of simulation run-times. Figure 1 shows the methodology used for comparing the simulation run-times of three contemporary AMS HDLs namely VHDL-AMS, Verilog-AMS and SystemC-AMS against HSPICE.

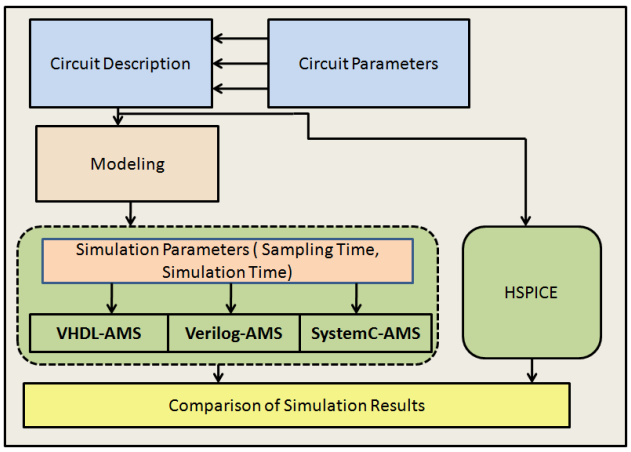

Figure 1: Modeling and Simulation Environment

During the past few decades, several work in the Computer-aided design (CAD) literature were concerned with studying possible frameworks for the simulation of mixed signal designs. For instance, in [6], the authors discusses a new methodology for the Jiles-Atherton model of ferromagnetic core hysteresis using mixed-domain SystemC and VHDL-AMS implementation to ensure numerically reliable integration of the magnetisation slope. In [1], the authors proposed a SystemC/Simulink co-simulation framework for embedded system that relies on Simulink for the continuous simulation and SystemC for the discrete simulation based on one or more synchronization model. While in [2], the authors provides a co-simulation environment based on SPICE and SAVANT. Another mixed-domain simulation framework was proposed in [3] based on VHDL and ELDO. The commercial tool Nexus-PDK [4] supports co-simulation of cycle accurate C/C++ with

SystemC, MATLAB/Simulink, and VHDL/Verilog simulators. In [5], the authors implemented a mixed-signal, functional level simulation framework based on SystemC for system-on-a-chip applications. The framework includes a C++ mixed-signal modules. In [15], the authors presents a preliminary approach for the modeling and simulation of a simple but complete Wireless Sensor Network with two nodes using SystemC-AMS. The paper also explains the advantage of SystemC-AMS over other HDL's in modeling and simulation of such network. In [7], the author focuses on commonalities and differences between the two mixed-signal hardware description languages, VHDL-AMS and Verilog-AMS, in the case of modeling heterogeneous or multidiscipline systems.

The rest of the paper is organized as follows. In Section 2, we describe the AMS simulation approaches used in Verilog-AMS, VHDL-AMS and SystemC-AMS with emphasis on the concept of simulation cycle. In Section 3, we illustrate and compare the simulation experiments using a set of AMS benchmark circuits [11], before concluding with an outline for future directions in Section 4.

2 AMS Simulation approach

VHDL-AMS, Verilog-AMS and SystemC-AMS allow the modeling of discrete and continuous-time signals, or a combination of both in a single design. Connecting functional and behavioral models is accomplished with the help of terminals and quantities. VHDL-AMS, Verilog-AMS and SystemC-AMS can capture of behavior of AMS designs at higher levels of abstraction, which brings down the simulation time, while preserving the functionality of the design.

2.1 VHDL-AMS

VHDL-AMS [9] was developed as an extension to VHDL to describe and specify AMS circuits and systems. The analog parts are modeled as lumped systems and can be described by ordinary differential and algebraic equations. Systems in both electrical and non-electrical domains can be described and specified at various levels of abstraction.

The VHDL-AMS simulation cycle starts with the initialization phase [Figure 2], which consists of four main steps. The analog system equations are determined from the analog part of the VHDL-AMS model. The initial conditions for the equations are determined from the initial values of the quantities, their attributes and also from the break statements. The initial values of the driving signals, and quantities defined by attributes are first computed. The processes are then executed once until they suspend. At the end of the processes execution the simulation time is set to zero. Both Verilog-AMS and SystemC-AMS follow a similar initialization technique as above.

The actual VHDL-AMS simulation cycle (Figure 3) begins with the computation

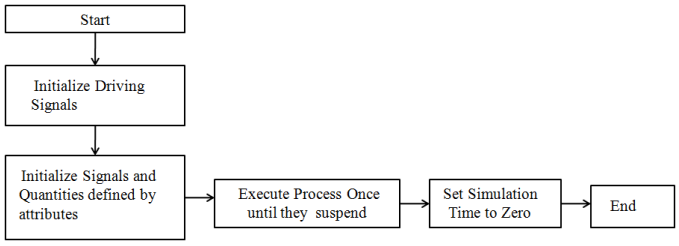

Figure 2: VHDL-AMS Simulation Cycle- Initialization [9].

of analog solution points (arrow 1). This continues until the next digital event is scheduled or an event occurs on the analog and digital interface (arrow 2). To compute a digital evaluation point, signals are updated first. After that, any triggered processes are executed until they settle. If the time for the next digital evaluation T_n is equal to current time T_c, the digital simulator is called again (arrow 3). If T_n is not equal to T_c, the analog solver is called, and the next cycle begins (arrow 4). This continues until the end of simulation is reached (arrow 5).

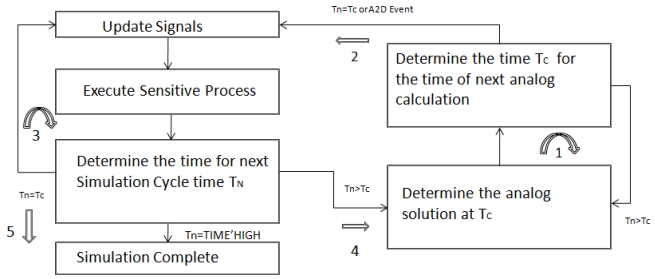

Figure 3: VHDL-AMS Simulation Cycle- Execution [9].

2.2 Verilog-AMS

Verilog-AMS HDL [16] is derived from IEEE 1364-1995 Verilog HDL. It allows designers to capture the behavior of AMS designs at different levels of abstraction. Verilog-AMS HDL [17], can also be used to describe discrete (digital) systems (per IEEE 1364-1995 Verilog HDL) and mixed-signal systems using both discrete and continuous descriptions as defined in this Language Reference Manual (LRM). Verilog-AMS HDL consists of the complete IEEE 1364-1995 Verilog HDL specification, an analog equivalent for describing analog systems, and extensions to both for specifying the full Verilog-AMS HDL.

Figure 4, illustrates a typical Verilog-AMS HDL simulation cycle which includes:

1. *Initialization:* The initialization phase of a transient analysis is the process of initializing the circuit state before advancing time.

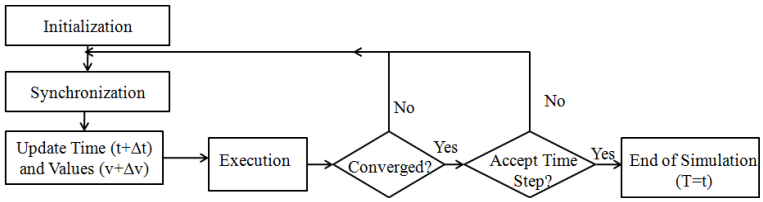

Figure 4: Verilog-AMS Simulation Cycle [16]

2. *Synchronisation:* A Verilog-AMS simulation consists of a number of analog and digital processes communicating via events, shared memory and conservative nodes. All conservative nodes (macro process) are represented by matrices and solved jointly.

3. *Evaluation:* The equation describing the design is differential and nonlinear, which makes it impossible to solve directly. All approaches discretize time and solve the nonlinear equations iteratively.
   When an analog macro process is evaluated, the analog engine finds a potential solution at a future time, and it stores values for all the process nodes up to that time. A *wake up* event is scheduled for the acceptance time of the process, and the process is then inactive until it is either woken up or receives an event from another process.

4. *Update:* If a process is woken up by its own *wake up* event, it calculates a new solution point and deactivates. If it is woken up prior to acceptance time by an event that disturbs its current solution, it will cancel its own *wake up* event, recalculate its solution and schedule a new *wake up* event for the new acceptance time. The process may also wake itself up early for reevaluation by use of a timer (which can be viewed as just another process). If the analog process identifies future analog events such as crossings or timer events then it will schedule its wake-up event for the time of the first such event rather than the acceptance time. If the analog process is woken by such an analog event it will communicate any related events at that time and de-activate, rescheduling its wake-up for the next analog event or acceptance. Events to external processes generated from analog events are not communicated until the global simulation time reaches the time of the analog event. If the time to acceptance is infinite then no wake-up event needs to be scheduled. Analog processes are sensitive to changes in all variables and digital signals read by the process unless that access is only in statements guarded by event expressions.

5. *Convergence:* In the analog kernel, the behavioral description is evaluated iteratively until the Newton-Raphson method converges. On the first iteration, the signal values used in expressions are approximate and do not satisfy Kirchhoffs Laws. As the iteration progresses, the signal values ap-

proach the solution. Iteration continues until two convergence criteria are satisfied.

2.3 SystemC-AMS

SystemC-AMS [13] is an extension of SystemC that uses an open and layered approach [14]. The base layer is the existing SystemC 2.0 kernel as shown in Figure 5. On top of the base layer, two sets of layers are defined: Interface to the existing SystemC layers, (e.g, discrete event channels), and a new set of AMS layers such as the synchronisation layer, the solver layer, and the user layers.

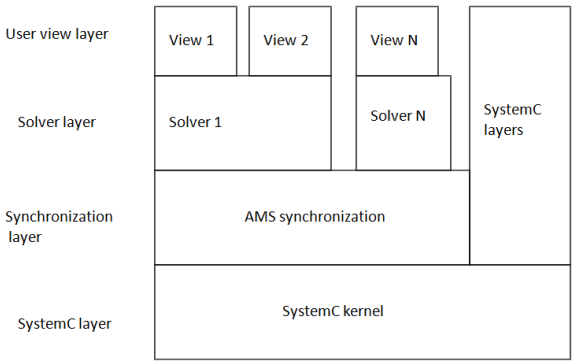

Figure 5: SystemC-AMS Architecture

The *user view layer* provides methods to describe the continuous-time models in terms of procedural behavior, equations, transfer functions, state-space formulations, and as netlists of primitives. Due to its open source architecture, the user can add additional features to the simulator depending on their application. SystemC-AMS uses a Synchronous Data Flow (SDF) [12] model of computation for modeling and simulation [8]. The *solver layer* provides different implementations of solvers (such as linear solver to solve electrical network) that are required to simulate specific AMS descriptions. The *synchronization layer* implements a mechanism to organize the simulation of a SystemC-AMS model that may include different continuous-time and discrete-event parts. SystemC-AMS defines a generic interface for various continuous-time solvers and provides methods to synchronize analog solvers and the discrete kernel of SystemC.

In [14], the authors describe the semantic model of SystemC-AMS and propose changes to the SystemC 2.0 simulation cycle to extend its capabilities to support the execution of dataflow clusters. A dataflow cluster (or a cluster process) consists of one or more continuous-time modules embedded inside a discrete-event process which is managed by a coordinator. An elaborated AMS design in SystemC-AMS consists of a set of interconnected cluster processes and discrete-event SystemC processes. The cluster process simulation runs at a constant time step determined by a coordinator based on the sampling rates of the signals in the

dataflow cluster and is generally much higher than the minimum required Nyquist rate. Discrete-event models are simulated using delta cycle mechanism which allows emulation of concurrent behavior.

The SystemC-AMS simulation cycle [14] is shown in Figure 6 and is summarized below:

1. *Initialization:* The initialization methods registered in SystemC-AMS modules are executed including the initial condition definitions.
2. *Evaluation:* Processes are only executed at delta 0 in the order defined by the static scheduling (delta cycles provide a standard way to emulate concurrency when simulating discrete-event models). The cluster processes will be reactivated, always at delta 0, at every time step defined for the cluster.
3. Repeat step 2 while there are still processes ready to run, else go to step 4.
4. *Update:* Signals are updated with their new values.
5. Go to step 2 if the signal updates generated events with zero delay (delta cycle), else go to step 6.
6. Finish simulation if there are no more pending events, else go to step 7.
7. Advance the time to the earliest pending event.
8. Determine ready to run processes and go to step 2.

A SystemC model consists of a hierarchical network of parallel processes, which exchange messages under the control of the simulation kernel process and concurrently update the value of signals and variables. Signal assignment statements do not affect the target signals immediately, but the new values become effective in the next simulation cycle. The kernel process resumes when all user-defined processes become suspended either by executing a wait statement or upon reaching the last process statement. On resumption, the kernel updates the signal and variable and suspends again when the user-defined process starts. If the time of the next earliest event T_n is equal to the current simulation time T_c, the user processes execute a delta cycle.

3 Comparison and Simulation Results

For the comparison, we have chosen four small to medium sized analog and switch capacitor circuits. We modeled those circuits in VHDL-AMS, Verilog-AMS, SystemC-AMS and in HSPICE and simulated them for transient and AC analysis run-time measurements. HSPICE run-time measurement results are provided as reference since it is still the dominant and widely accepted simulator for analog circuits to-date. We define the simulation run-time as the time taken by

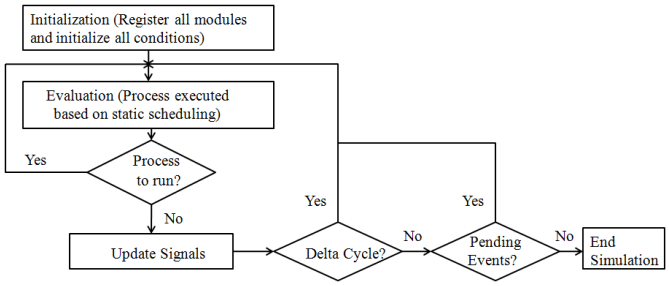

Figure 6: SystemC-AMS Simulation Cycle [14]

a given machine to simulate the design for a specified duration. VHDL-AMS, Verilog-AMS, and HSPICE designs were simulated using Mentor Graphics Tools on an ULTRA SPARC-IIIi machine (177 MHz CPU, 1024 Mbyte memory). The SystemC-AMS design descriptions were also compiled and executed on the same workstation.

The four circuits selected for the simulation are:

1. Continuous-Time State Filter [11].
2. Low Pass Active Filter [10].
3. Leap Frog Filter [11].
4. First Order Switch Capacitor Filter [10].

The Continuous-Time State space Filter circuit (Figure 7) has three outputs; the low pass output V_{lp}, the high pass output V_{hp}, and the band pass output V_{bp}. The circuit design parameter and the resulting component values are summarized in Table 1.

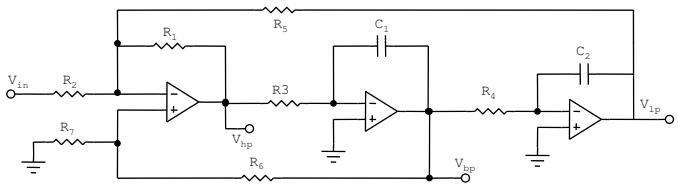

Figure 7: Continuous-Time State Filter

Table 1: Continuous Time State Filter Parameters.

Circuit Parameters	Fc=795Hz, G_{dc}=1, Q=1.11
Resistors	R_1=R_2=R_3=R_4=R_5=10kΩ, R_6=7kΩ, R_7=3kΩ
Capacitors	C_1=20nF, C_2=20nF

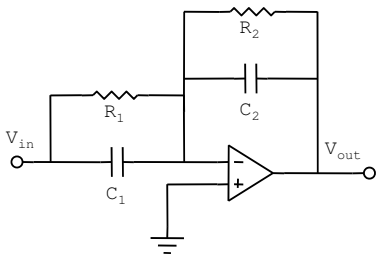

Figure 8: Low Pass Active Filter

The Low Pass Active Filter circuit is shown in Figure 8. The circuit design parameter and the resulting component values are summarized in Table 2.

Table 2: Low Pass Active Filter Parameters.

Circuit Parameters	G_{dc}=1, F_{lp}=1kHz
Resistors	R_1=398Ω, $R2$=3.98kΩ
Capacitors	C_1=100pF, C_2=10nF

The low pass Leap Frog Filter circuit is shown in Figure 9, whereas the design parameters and the resulting component values are given in Table 3.

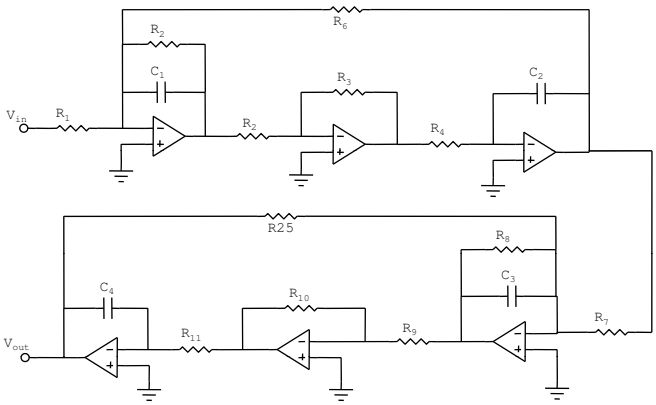

Figure 9: Leap Frog Filter

The First Order Switch Capacitor Filter circuit is shown in Figure 10. The circuit is modeled at component level using ideal switches and operational amplifiers. The design is simulated using ideal two-phase non-overlapping clock. The circuit design parameter and the resulting component values are summarized in Table 4.

Table 3: Leap Frog Filter Parameters.

Circuit Parameters	F_{lp}=900Hz, G_{dc}=1
Resistors	R_1=R_2=R_3=R_4=R_5=10kΩ R_6=R_7=R_8=R_9=R_{10}=R_{11}=10kΩ
Capacitors	C_1=10nF, C_2=20nF, C_3=20nF, C_4=10nF

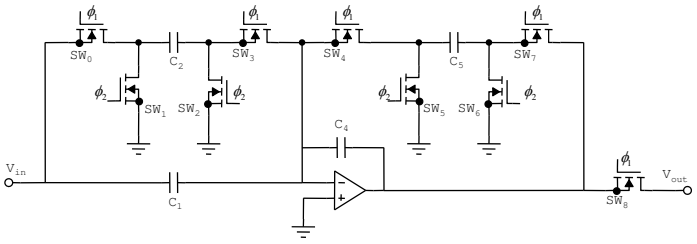

Figure 10: First Order Switch Capacitor Filter

Table 4: First Order Switch Capacitor Filter Parameters.

Circuit Parameters	G_{dc}=1, F_s=64kHz, F_p=1kHz, T_s=15.635μs
Capacitors	C_1=0pF, C_2=1.032pF, C_3=1.032pF, C_4=10pF

Table 5: Simulation Times for 10ms Simulation run.

Circuit	Frequency (Hertz)	VHDL-AMS (Seconds)	Verilog-AMS (Seconds)	SystemC-AMS (Seconds)	HSPICE (Seconds)
Low	1k	0.13	0.12	48.24	48.72
Pass	2k	0.17	0.21	48.45	48.73
Active	4k	0.26	0.26	48.16	48.74
Filter	40k	0.96	1.32	48.20	48.75
First	500	6.72	21.04	70.28	184.34
Order	1k	6.84	21.94	70.27	185.65
Switch	2k	6.97	19.98	70.39	186.13
Capacitor	4k	7.06	18.77	70.40	185.38
Continuous	100	0.07	0.09	49.20	57.24
Time	795	0.07	0.07	48.26	56.61
State	1k	0.10	0.13	49.07	56.62
Filter	10k	0.38	0.50	49.71	56.61
	40k	1.34	1.95	49.55	56.66
Leap	1k	0.09	0.22	50.26	66.85
Frog	1.4k	0.12	0.15	50.56	66.89
Filter	10k	0.52	0.82	50.66	66.70
	100k	4.99	6.92	51.27	66.73

3.1 Discussion

Table 5 summarises the experiment results. The first and second column represents the circuit and the frequency of operation. The next columns represent the simulation run-times of, respectively, VHDL-AMS, Verilog-AMS, SystemC-AMS and HSPICE in seconds. From the table statistics, we note that for all frequency ranges, the simulation run-times for VHDL-AMS and Verilog-AMS are almost comparable and in some cases negligible. Both VHDL-AMS and Verilog-AMS outperforms SystemC-AMS and HSPICE in their simulation run-times. On the other hand, the simulation run-times are comparable for SystemC-AMS and HSPICE with SystemC-AMS performing slightly better in some cases.

For higher frequency inputs the simulation run time is slightly higher than for low frequency inputs. This is because when the input signal changes at a faster rate (higher frequency) the analog solver requires more iterations to converge to an analog solution point for a given accuracy requirements and hence results in a slight increase in simulation time. This is seen for each circuit described in the VHDL-AMS, Verilog-AMS, SystemC-AMS and HSPICE as one looks at the simulation run-time numbers starting from low frequency values to high frequency values.

The circuit simulation times of the first order switch capacitor filter are larger because of the non-linear switches in the filter circuit which cause the simulator to iterate more often at the instants of time when the switches change states from ON to OFF or vice versa. Since the switches are turned ON and OFF a fixed number of times in a 10ms simulation the simulation run-time is independent of the input signal frequency but rather depends on the clock signal frequency used for controlling the switches.

4 Conclusion

The simulation of analog and mixed signal circuits is both memory and CPU intensive. The simulation speed depends on the complexity of the circuit, the length of simulation, and the frequency of the input signals. In this paper, we give an overview about the simulation cycles of VHDL-AMS, Verilog-AMS and SystemC-AMS. Four benchmark circuits were described, simulated and their run-times were compared with that of HSPICE simulation.

Our experience can be summarised as follows: First, the results show that for all filter circuits, the simulation run-times increase as the input signal frequency increases. This is again due to the fact that the simulator requires more iterations for each analog solution point if the input signal changes faster as compared to a slowly varying signal for a given time resolution and accuracy requirements. We observe the superiority of VHDL-AMS and Verilog-AMS against SystemC-AMS and HSPICE simulation runtimes. However, the HSPICE and SystemC-AMS run times are comparable for all filter circuits.

Unfortunately, SystemC-AMS is still in its development phase, so there is a lack of available libraries that would have allowed to explore more complex case studies. We believe that with growing user and developer community for SystemC-AMS such library would be available allowing us to conduct more experimental results on the language.

Future plans include a detailed investigation about the simulation cycle algorithms and also to tackle larger case studies to get a more indepth knowledge about the quantitative properties of the language simulators.

References

[1] F. Bouchhima, M. Brirel, G. Nicolescu1, M. Abid, E. M. Aboulhamid. A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation, In Proc. Behavioral Modeling and Simulation, IEEE, pp. 1-6, 2006.

[2] D.E. Martin, P.A. Wilsey, R.J. Hoekstra, E.R. Keiter, S.A. Hutchinson, T.V. Russo, L.J. Waters. Integrating Multiple Parallel Simulation Engines for Mixed-technology Parallel Simulation, In Proc. Simulation Symposium, IEEE, pp. 45-52, 2002.

[3] H. El Tahawy, D. Rodriguez, S. Garcia-Sabiro, J.J. Mayol. VHD_ELDO: A new mixed mode simulation, In Design Automation Conference, IEEE/ACM, pp.546-551, 1993.

[4] Celoxia *Website: http://www.celoxica.com/,* 2008.

[5] T.E. Bonnerud, B. Hernes, T. Ytterdal. A Mixed-signal Functional Level Simulation Framework based on SystemC for System-on-a-Chip Applications, In Proc. Custom Integrated Circuits, IEEE, pp. 541-544, 2001.

[6] H. Al-Junaid, T.Kazmierski. HDL Models of Ferromagnetic Core Hysteresis Using Timeless Discretisation of the Magnetic Slope. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 12, pp. 2757-2764, 2006

[7] Franois Pcheux, Christophe Lallement, Alain Vachoux. VHDL-AMS and Verilog-AMS as Alternative Hardware Description Languages for Efficient Modeling of Multidiscipline Systems. IEEE Transactions on Computer-Aided Design of Integerated Circuits and Systems,Vol. 24, No. 2, pp.204-225, February 2005.

[8] K. Einwich, C. Clauss, G. Noessing, P. Schwarz, and H. Zojer. SystemC Extensions for Mixed-Signal System Design. In Proc. Forum on Design Languages, pp.1-6, 2001.

[9] W. Haas, U. Heinkel, H. Braisz, T. Gentner, M. Padeffke, T. Buerner, G. Alexander, F. Alexander. The VHDL Reference: A Practical Guide to Computer-Aided Integrated Circuit Design Including VHDL-AMS, Wiley, 2000.

[10] D. A. Johns, K. Martins. Analog Integrated Circuit Design, Wiley, 1996.

[11] B. Kaminska K. Arabi, I. Bell, P. Goteti, J.L. Huertas, B. Kim, A. Rueda, M. Soma. Analog and Mixed-Signal Benchmark Circuits-First Release, In Proc. Test Conference, IEEE, pp. 183-190, 1997.

[12] E. A. Lee, D.G. Messerschmidt. Synchronous Data Flow, In Proc. of the IEEE, Vol.75, Issue.9, Sept, 1987.

[13] SystemC-AMS USER Community Website: http://www.systemc-ams.org, 2008

[14] A. Vachoux, C. Grimm, K. Einwich. Towards Analog and Mixed-Signal SOC Design with SystemC-AMS. In Electronic Design, Test and Applications, IEEE, pp.97-102, 2004.

[15] M. Vasilevski, F. Pecheux, H. Aboushady, and L. de Lamarre. Modeling Heterogeneous Systems Using SystemC-AMS Case Study: A Wireless Sensor Network Node. In Proc. Behavioral Modeling and Simulation Workshop, IEEE, pp.1-6, Sept, 2007.

[16] Verilog Analog and Mixed Signal Language Reference Manual (2004). Available: http://www.eda.org/verilog-ams/

[17] Accellera, Verilog-AMS Language Reference Manual Analog & Mixed-Signal Extensions to Verilog-HDL. Version 2.1, January 2003. Avaialbe: http://www.designers-guide.org/