A Comparative Study of AMS Circuit Simulation in VHDL-AMS and SystemC-AMS

Rajeev Narayanan, Naeem Abbasi, Ghiath Al Sammane, Mohamed Zaki, Sofiène Tahar
Dept. of ECE, Concordia University Montreal, Quebec, Canada
Email: {r_naraya, n_ab, samian, mzaki, tahar}@ece.concordia.ca

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

I. Introduction

With the advances in embedded systems, the role of analog and mixed-signal (AMS) designs as an interface between the electronic system and the real world is becoming more and more important. However, AMS design modeling and simulation is a challenging task that requires lots of expertise and deep understanding of their behavior, when compared with digital designs.

Traditionally, AMS designs are described by netlist format in SPICE and simulated using available circuit simulators. Though, it looks simple, the circuit simulators are not capable of handling efficiently larger designs at lower levels of abstraction due to its computational intensive nature. An alternate approach, is to capture the behavior of the AMS designs at higher level of abstraction using hardware description languages (HDL). This technique (called behavioral modeling) brings down the simulation time of the design, but is less accurate when compared to low level simulation. For a tradeoff between accuracy and run-time, the designers can look at modeling AMS designs at higher levels of abstraction and HDLs provide an attractive platform to carry out such modeling.

The rest of the paper is organized as follows. We start by giving an overview of the relevant related work in Section II. In Section III, we describe the AMS simulation approaches used in VHDL-AMS [17] and SystemC-AMS [13] with emphasis on the concept of simulation cycle. In Section IV, we illustrate and compare the simulation experiments using a set of AMS benchmark circuits [8], before concluding with a discussion and outline for future directions in Section V.

II. Related Work

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 [1], the authors propose 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 [2], 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 [9], the authors developed a co-simulation environment based on SPICE and SAVANT. Another mixed-domain simulation framework was proposed in [12] 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 [10], the authors examine the applicability of SystemC-AMS for MEMS systems and do a comparison with VHDL-AMS based implementation. In [3], the authors developed a mixed-signal, functional level simulation framework based on SystemC for system-on-a-chip applications. The framework includes a C++ mixed-signal modules. They implemented a virtual clock for scheduling and synchronization between analog and digital components. In [16], 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.

All of the above mentioned papers were concerned with AMS designs high level modeling and simulation in a multi-domain environment. However, none of them compares the simulation performance of the developed AMS systems against the low level implementation, to get an insight about the tradeoffs that might occurs between accuracy and simulation performance. With the standardization of SystemC and on going standardization for its AMS extension, the aim of this paper is to extend the insight by comparing the simulation run-time of VHDL-AMS and SystemC-AMS for the analog and mixed signal benchmark circuits.

III. AMS Simulation approach

SystemC-AMS and VHDL-AMS allow the modeling of discrete-time signals 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. SystemC-AMS and VHDL-AMS allow the capture of behavior of AMS designs at higher levels of abstraction, which brings down the simulation time, while preserving the functionality of the design.

A. *VHDL-AMS*

VHDL-AMS [17] was developed as an extension to VHDL to describe and specify AMS circuits and systems. Its syntax was defined with a semantics to support conservative and non-conservatives modeling of analog part of 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. No particular technique to solve equations is specified in the standard, it only specifies what results must be achieved by a simulator, leaving the door open for creative algorithm development and their efficient implementation. VHDL-AMS standard also describes the interaction between the analog and digital parts of a model, solution of equations with discontinuities, and support for frequency domain small-signal noise simulation.

The VHDL-AMS simulation cycle [6] is shown in Figure 1 and Figure 2. The simulation starts with the initialization phase (shown in Figure 1). The initialization phase 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.

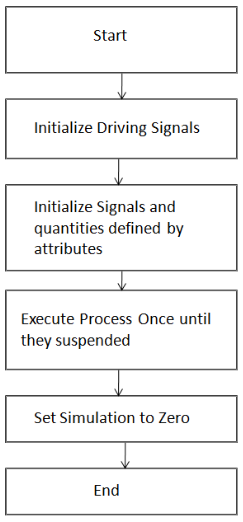
Start
Initialize Driving Signals
Initialize Signals and quantities defined by attributes
Execute Process Once until they suspended
Set Simulation to Zero
End

Fig. 1. VHDL-AMS Simulation Cycle- Initialization.

The VHDL-AMS simulation cycle (Figure 2) begins with the computation of analog solution points (see arrow 1). This continues until the next digital event is scheduled or an event occurs on the analog and digital interface (see 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 (see arrow 3). If T_n is not equal to T_c, the analog solver is called, and the next cycle begins (see arrow 4). This continues until the end of simulation is reached (see arrow 5).

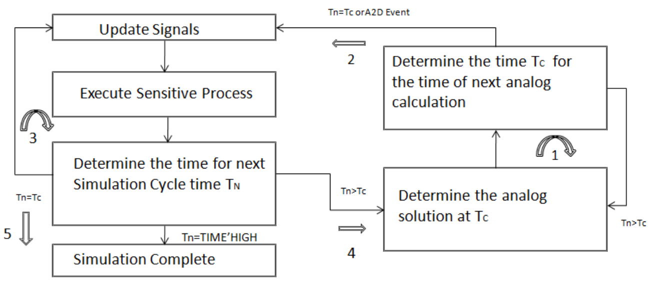
Update Signals
Tn=Tc orA2D Event
2
Determine the time Tc for the time of next analog calculation
Execute Sensitive Process
3
1
Determine the time for next Simulation Cycle time TN
Tn=Tc
Tn>Tc
Determine the analog solution at Tc
Tn>Tc
5
Tn=TIME'HIGH
Simulation Complete
4

Fig. 2. VHDL-AMS Simulation Cycle- Execution [6]

B. *SystemC-AMS*

SystemC-AMS [14] is an extension of SystemC that uses an open and layered approach [15] as shown in Figure 3.

The base layer is the existing SystemC 2.0 kernel. On top of the base layer, two sets of layers are defined:

1) Interface to the existing SystemC layers, (e.g, discrete event channels), and

2) A new set of AMS layers such as the synchronisation layer, the solver layer, and the user layers.

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) [11] model of computation for modeling and simulation [5]. 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 [15], 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

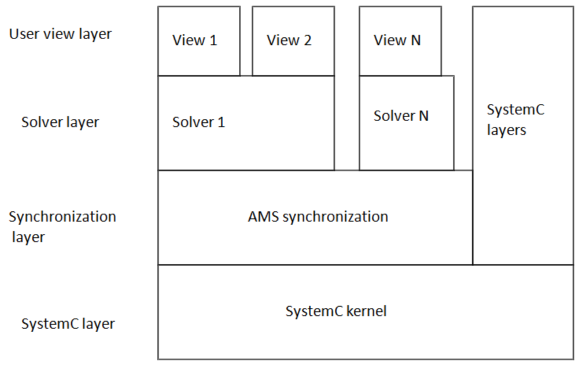

Fig. 3. SystemC-AMS Architecture

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 is shown in Figure 4 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.

IV. 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, SystemC-AMS and in HSPICE and

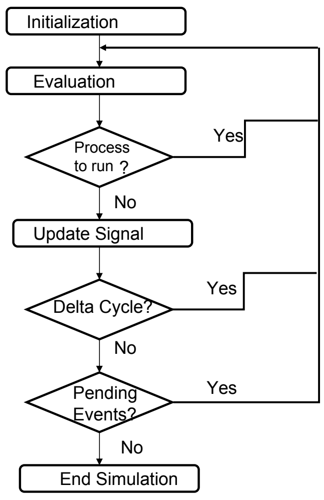

Fig. 4. SystemC-AMS Simulation Cycle

simulated them for run-time measurements. HSPICE run-time measurement results are provided as 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 a given machine to simulate the design for a specified duration. VHDL-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 [8] [7]:

1) Continuous-Time State Filter.
2) Low Pass Active Filter.
3) Leap Frog Filter.
4) First Order Switch Capacitor Filter.

The design descriptions in both VHDL-AMS and SystemC-AMS were verified through simulations and by comparing the transient and AC analysis results with those obtained from HSPICE simulations.

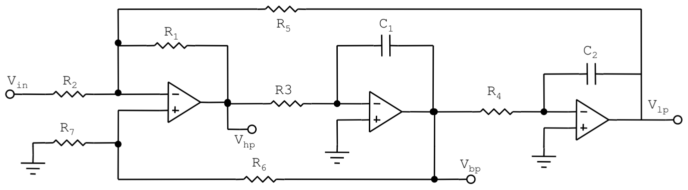

Fig. 5. Continuous-Time State Filter

A. Continuous-Time State Filter

The Continuous-Time State space Filter circuit (Figure 5) 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 I.

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

TABLE I
CONTINUOUS TIME STATE FILTER PARAMETERS.

Freq (Hertz)	VHDL AMS (Seconds)	SystemC AMS (Seconds)	HSPICE (Seconds)
100	0.07	49.20	50.8
795	0.07	48.26	50.8
1K	0.10	49.07	51.3
10K	0.38	49.71	50.9
40K	1.34	49.55	54.9

TABLE II
SIMULATION TIMES FOR 10MS SIMULATION RUN FOR CONTINUOUS TIME STATE FILTER.

For the results in Table II, Figure 6 shows a plot of the input signal frequency versus the simulation run-time for the continuous-time state space filter circuit. The thin dotted line represents SystemC-AMS, the thick dotted line represents HSPICE and solid line represents VHDL-AMS.

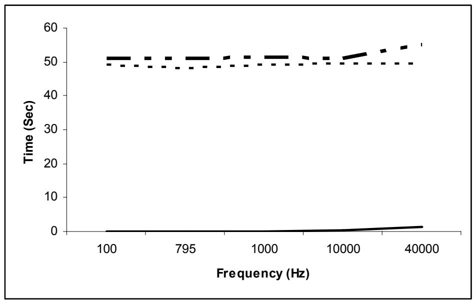

Fig. 6. Continuous Time State Filter Simulation Result

We note that for all frequency ranges, the simulation run-times for VHDL-AMS is almost negligible compared to SystemC-AMS and HSPICE. On the other hand, the simulation run-times are comparable for SystemC-AMS and HSPICE.

B. Low Pass Active Filter

The Low Pass Active Filter circuit is shown in Figure 7. The circuit design parameter and the resulting component values are summarized in Table III.

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

TABLE III
LOW PASS ACTIVE FILTER PARAMETERS.

For the results in Table IV, Figure 8 shows a plot of the input signal frequency versus the simulation time for the low pass active filter circuit. The small dotted line represents SystemC-AMS, the big dotted line represents HSPICE and bold line represents VHDL-AMS.

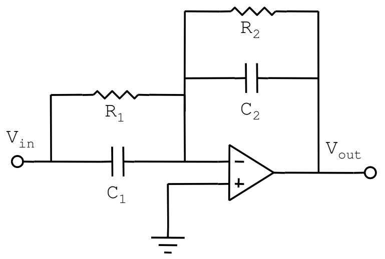

Fig. 7. Low Pass Active Filter

Freq (Hertz)	VHDL AMS (Seconds)	SYSTEMC AMS (Seconds)	HSPICE (Seconds)
1K	0.13	48.24	42.4
2K	0.17	48.45	42.9
4K	0.26	48.16	42.9
40K	0.96	48.20	43.0

TABLE IV
SIMULATION TIMES FOR 10MS SIMULATION RUN FOR LOW PASS ACTIVE FILTER.

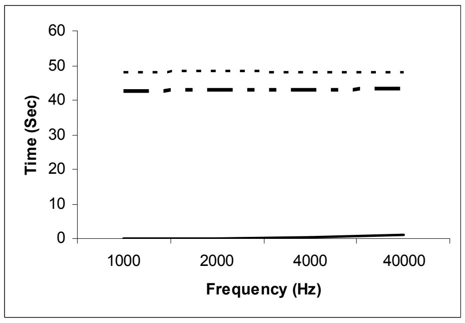

Fig. 8. Low Pass Active Filter Simulation Result

We note that, again for all frequency ranges VHDL-AMS is faster than both SystemC-AMS and HSPICE and the run-times are almost negligible. On the other hand, the run-times of SystemC-AMS and HSPICE are almost comparable with SystemC-AMS performing slightly better than HSPICE for all frequencies.

C. Leap Frog Filter

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 V.

For the results in Table VI, Figure 10 shows a plot of the input signal frequency versus the simulation time for the leap frog filter circuit. The small dotted line represents SystemC-AMS, the big dotted line represents HSPICE and bold line represents VHDL-AMS.

We note from the figure that, for all frequency ranges VHDL-AMS is faster than SystemC-AMS and HSPICE. For frequency greater than 10KHz, there is a linear increase in simulation run-times for VHDL-AMS. Also, for all frequency

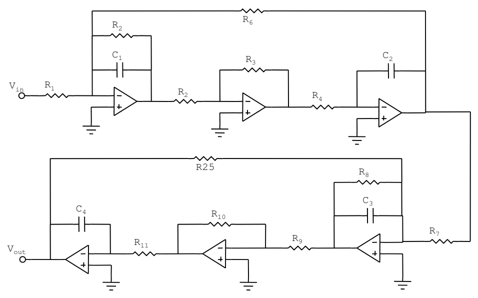
Fig. 9. Leap Frog Filter

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

TABLE V
LEAP FROG FILTER PARAMETERS.

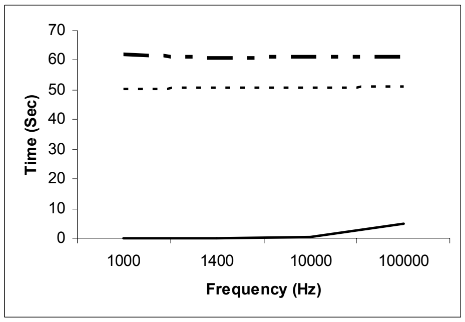

Fig. 10. Leap Frog Filter Simulation Result

ranges, SystemC-AMS is slightly faster than HSPICE simulation.

D. First Order Switch Capacitor Filter

The First Order Switch Capacitor Filter circuit is shown in Figure 11. 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 VII.

For the results in Table VIII, Figure 12 shows a plot of the input signal frequency versus the simulation time for the first order switch capacitor filter circuit. The small dotted line represents SystemC-AMS, the big dotted line represents HSPICE and bold line represents VHDL-AMS.

Freq (Hertz)	VHDL AMS (Seconds)	SYSTEMC AMS (Seconds)	HSPICE (Seconds)
1K	0.09	50.26	61.8
1.4K	0.12	50.56	60.4
10K	0.52	50.66	61.0
100K	6.92	51.27	60.9

TABLE VI
SIMULATION TIMES FOR 10MS SIMULATION RUN FOR LEAP FROG FILTER.

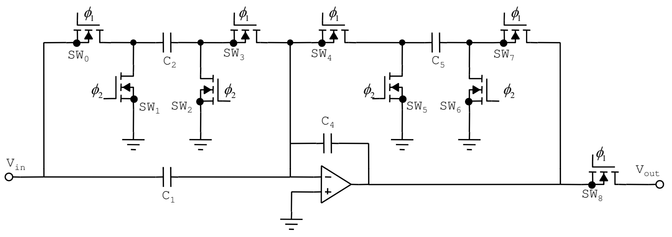
Fig. 11. First Order Switch Capacitor Filter

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 VII
FIRST ORDER SWITCH CAPACITOR FILTER PARAMETERS.

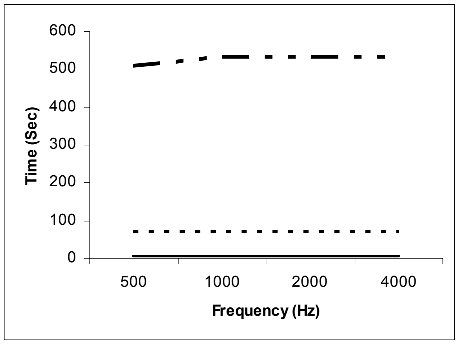

Fig. 12. First Order Switch Capacitor Filter Simulation Result

From the figure, we note that for all frequency ranges VHDL-AMS is faster than SystemC-AMS and HSPICE and the run-times are negligible. On the other hand, SystemC-AMS is faster by a factor of 7.0 compared to HSPICE for all frequency ranges.

E. Discussion

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, SystemC-AMS and HSPICE as one looks at the simulation run-time

Freq (Hertz)	VHDL AMS (Seconds)	SYSTEMC AMS (Seconds)	HSPICE (Seconds)
500	6.72	70.28	509.0
1K	6.84	70.27	532.0
2K	6.97	70.39	530.5
4K	7.06	70.40	533.0

TABLE VIII
SIMULATION TIMES FOR 10MS SIMULATION RUN FOR FIRST ORDER SWITCH CAPACITOR FILTER.

numbers starting from low frequency values to high frequency values.

However, 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.

V. 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 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 of all, the results show that for all the 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 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 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 extending the comparison to the syntactic and semantical aspects of the HDL and detailed investigation about the simulation cycle algorithms. We also need to tackle larger case studies to get a more indepth knowledge about the quantitative properties of the language simulators. We also would like to extend the comparison to include other HDL like Verilog-AMS.

References

[1] 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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] E. Markert, M. Schlegel, G. Herrmann,D. Mller. Subproject A2: Examination of the Applicability of SystemCAMS for the Description of MEMS, Technical Report, TU Chemnitz, Faculty of Electrical Engineering and Information Technology, 2004.

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

[12] 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.

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

[14] A. Vachoux, C. Grimm. Analog and Mixed Signal Modelling with SystemC-AMS. In Proc. on Circuits and Systems, IEEE, pp. 913-917, 2003.

[15] 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.

[16] M. Vasilevski, F. Pecheux, H. Aboushady, and L. de Lamarre. Modeling Heterogeneous Systems Using SystemC-AMS Case Study: A Wireless Sensor Network Node. Behavioral Modeling and Simulation Workshop, 2007. BMAS 2007. IEEE International, Sept, 2007.

[17] VHDL-AMS IEEE Standard. Website: http://www.eda.org/vhdl-ams/, 2007