Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

SEAMS: SIMULATION ENVIRONMENT FOR VHDL-AMS

Peter Frey
Kathiresan Nellayappan
Vasudevan Shanmugasundaram
Ramesh S. Mayiladuthurai,
Chetput L. Chandrashekar
Harold W. Carter

University of Cincinnati
Dept. of ECECS
PO Box 210030
Cincinnati, OH 45221–0030, U.S.A.

ABSTRACT

VHDL-AMS is an Analog and Mixed-Signal extension to the Very High Speed Integrated Circuit Hardware Description Language (VHDL). With the standardization of VHDL-AMS, capable and efficient simulators are in demand for exercising complex analog and mixed-signal models. The simulation of the language requires the ability to handle several levels of design hierarchy, the combination of multiple domains of modeling and the synchronization of continuous and discrete-event simulation. The expressive power of VHDL-AMS is also conducive for creating large simulation models. Large models have high resource demands especially on memory and execution time making parallel simulation no longer an option but a requirement. This paper introduces the issues involved in the design of a VHDL-AMS simulator and illustrates the simulation approach provided by SEAMS a parallel VHDL-AMS simulator. A performance study is presented to analyze the effectiveness of mixed-signal simulation using SEAMS.

1 INTRODUCTION

Mixed-signal simulation combines two different domains, namely the discrete and the differential equation models (Zeigler, 1976). This combination is becoming increasingly important as models of both domains demand functionality provided by the other. In the domain of electronic design automation, digital circuits are modeled using discrete models, whereas analog circuits are modeled with differential equations. The need to merge the two modeling domains can be established by the following facts. Today's digital circuits are designed to operate at ever increasing clock frequencies to the degree that a purely discrete representation of the circuit can no longer be considered accurate. On the other hand, chip designers would like to model and simulate entire systems at the analog level to obtain the fine accuracy produced by analog simulation. But physical limitations prevent the sole use of analog simulation. The introduction of mixed-signal simulation enables the designer to choose between model accuracy and simulation performance.

The need for a mixed-signal simulation capability has resulted in the birth of several proprietary languages like MAST, ABCDL, FAS to capture contemporary designs using the two modeling domains. As these languages and simulation environments are completely independent, the use and inter-operability of these domains is highly restricted. Moreover, models developed for one system are not readily usable in other environments. Efforts to overcome these limitations have resulted in the development and standardization of VHDL-AMS which is a mixed-signal language that promises to play an important role in the specification and verification of mixed-signal systems.

2 INTRODUCTION TO VHDL-AMS

VHDL-AMS provides behavioral modeling capability for both discrete and continuous systems. It is a minimal extension to VHDL and hence enjoys all the advantages already inherent in VHDL. The discrete models are specified using component instantiations and concurrent behavioral specifications. The continuous systems are modeled using Differential Algebraic Equations (DAEs). DAE-based continuous systems can be modeled similar to discrete models at several hierarchical levels. VHDL-AMS also provides mixed-discipline modeling, wherein different

```
−− Bouncing Ball
entity bouncing_ball is
end entity bouncing_ball

architecture simple of bouncing_ball is
    −− Declarations
    QUANTITY velocity: real;
    QUANTITY displacement: real;
  −− acceleration due to gravity
    CONSTANT G: real := 9.81;

begin
    −− specify initial conditions
    b1: break velocity => 0.0, displacement => 30.0;

    −− announce discontinuity & reset velocity value
    b2: break velocity => − 0.7 * velocity
       WHEN NOT(displacement'above(0.0));

    velocity_eqn: velocity == displacement'dot ;
    acceleration: velocity'dot == − G;
end architecture simple;
```

Figure 1: VHDL-AMS model of Bouncing ball

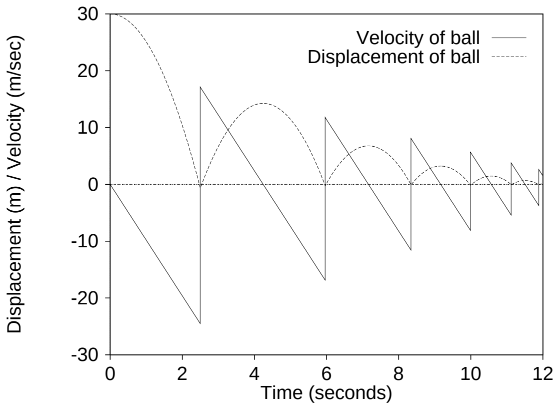

Figure 2: Simulation of a Bouncing ball

domains such as electrical, physical, and thermal, can be described and simulated in a single entity.

An overview of the features provided by VHDL-AMS is given with the help of the classical bouncing ball model (see Figures 1 & 2). This model is of special interest as it demonstrates the impact of design and performance of the simulation environment. The model describes the physics of a ball dropped from a certain height. Two variables are of interest viz - velocity and displacement. The initial conditions are set to zero velocity at a displacement representing the release point of the ball. The physical behavior is represented by two differential equations - first relates velocity and displacement, and the second relates velocity and acceleration. The physics of the problem dictates that when the ball hits the ground, velocity changes direction instantaneously and the magnitude is reduced due to elasticity. This change in velocity manifests as a discontinuity during simulation.

"Quantities" are the unknown continuous variables in the system, and their relationships expressing the system behavior are specified by DAEs. In the bouncing ball example, the quantities are *velocity* and *displacement*. VHDL-AMS uses simultaneous statements to represent general DAEs. Systems may also be formulated using signal-flow semantics which do not have any conservation semantics associated. This is true for the quantities in the example. VHDL-AMS provides additional branch quantities and terminals to support conservation semantics inherent to systems like electrical circuits. The language supports an attribute (*'dot*) to be able to specify time derivatives.

To represent discontinuities, VHDL-AMS provides a special construct called the *break* statement. The *break* statement indicates the possible occurrence of a discontinuity and also provides new initial conditions. In the example, two break statements are provided. The first break statement (`b1`) represents the initial conditions. New values for the quantities are assigned once at the beginning of the simulation. The second break statement (`b2`) models the impact of the ball on the ground. At impact, the ball displacement is zero. Thus, a new velocity (and direction) has to be calculated depending on elasticity.

Another form of discontinuity may result from a changing set of equations in the model. VHDL-AMS provides two statements, called *simultaneous if* and *simultaneous case*, to alter the set of equations. These statements specify conditional equations, which are solved if and only if certain conditions are satisfied. It is the task of the modeler to provide the necessary break statement to notify the simulator of possible discontinuities. DAE-based simulation engines involve numerical algorithms to solve the equations. However, as digital computers provide only finite accuracy, tolerances have to be introduced. VHDL-AMS enables the user to specify individual tolerances which must be satisfied by the simulator. The use of tolerances enables the designer to trade between accuracy and simulation speed, as more accurate solutions require more computation power.

3 MIXED-SIGNAL SIMULATION

This section presents the key components of a mixed-signal simulator - discrete event kernel & DAE-based solver. In addition, communication between the simulation paradigms must be provided. The communication in the model (Saleh et al. 1994) and in the simulator (Schmerler et al. 1995) is of critical importance as it affects correctness and performance. Physical processes represent the elementary

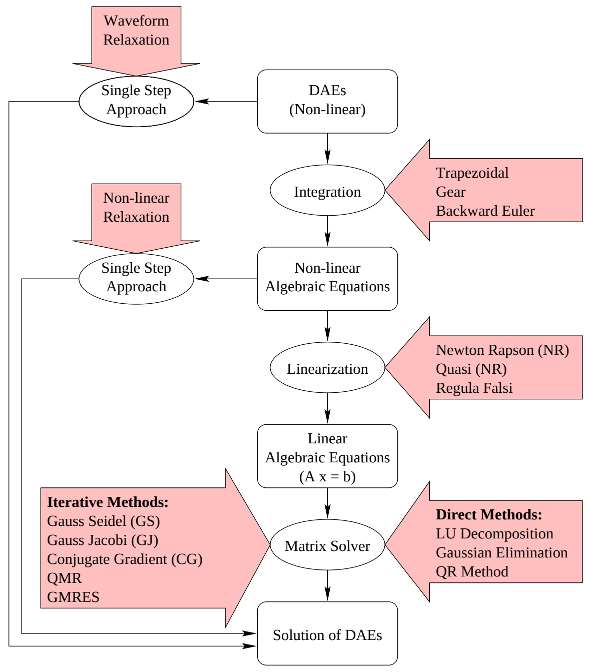

Figure 3: Differential Algebraic Equation Solver

unit of execution in a discrete-event simulation paradigm. Execution speed and memory requirements of such models have led to increased research activity in the parallel discrete-event simulation domain. The parallel discrete-event simulation domain is split into two major areas, namely the *conservative* and the *optimistic* simulation approach (Fujimoto 1990). A large set of parameters is responsible for the performance of a parallel discrete-event simulator, including process granularity, message size, and memory requirements. Perhaps most critical is the memory consumption in case of a mixed-signal simulation environment, which favors the optimistic approach (Soulé and Gupta 1991).

DAE-based solvers cannot easily be classified by their properties. Figure 3 provides an overview of the general structure of a DAE solver. The main execution flow shows the three major tasks (integration, linearization, and matrix solving) required during simulation. The shaded arrows contain just a few of the algorithms developed to perform the task. Single step approaches can be applied at several stages, resulting in larger set of simulation algorithms, but not all are applicable to the complete set of DAEs. Highly optimized algorithms provide significant performance improvements, but are restricted in use. For example, waveform relaxation (Lelarasmee et al. 1982) in circuit simulation is only guaranteed to converge if all nodes exhibit a capacitive load.

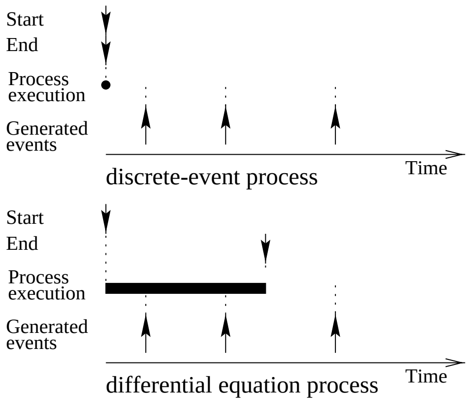

Figure 4: Mixed-signal Simulation Processes

The interface between the differential equation solver and the discrete event simulator is the most critical component of a mixed-signal simulator. Both paradigms have their own notion of time and the interface is responsible for coordinating simulation. The different notions of times are displayed in Figure 4. The discrete-event processes are instantaneous and events are scheduled into the future. In contrast, differential equation solvers always advance over simulation time and new events may influence the mixed-signal simulation immediately or in the future. The next time step in a differential equation solver is always influenced by tolerance requirements. To achieve performance improvements, dynamic time step adjustment has to be considered. Based on these properties of the individual processes and solvers, an interface specification has been established suitable for mixed-signal simulation (Frey et al. 1998).

With the identification of interface functionalities, mixed-signal simulation can be represented as a directed graph. Communication is represented by edges and vertices represent either differential equation solvers or discrete-event processes. Many differential equation solvers may be present in a single simulation model. Individual differential equation solvers compute a solution for an independent equation set. These independent “islands” (vertices of the graph) are common in large mixed-signal designs and hence are amenable for natural partitioning.

4 ELABORATION IN SEAMS

SEAMS (Carter 1998) is an implementation of a mixed-signal simulator supporting VHDL-AMS. Before a VHDL-AMS model can be simulated, sufficient information about it must be captured by the front-end analyzer and transformed into a suitable form for the simulator to execute. This transformation process is termed as *elaboration* and the result of elaboration may be a program to be interpreted by the simulator (interpreted simulation). The model description may also be compiled into object code and linked into the simulation kernel and directly executed (compiled simulation). The latter approach is used in SEAMS.

The elaboration of a design hierarchy provided by the VHDL-AMS model creates a collection of processes interconnected by signal nets, quantities and characteristic expressions. Characteristic expressions (henceforth referred to as CEs) represent constraints on the values of the quantities and are specified by DAEs. A set of CEs govers the behavior of the continuous portion of the mixed-signal system. SEAMS analyzes the VHDL-AMS model description first and then generates a hierarchical C++ representation during code generation. After code generation, elaboration takes place during the execution of the compiled model. Hence it is referred to as Run Time Elaboration of VHDL-AMS (RTEMS). RTEMS is a combination of both top-down and bottom-up approaches for constructing the set of processes, signal nets, quantities and CEs from the design. In this approach, the various tasks that are performed during elaboration of a design are: (a) Phase 1 - Construction of design units and objects and (b) Phase 2 - Construction of the Continuous system.

In Phase 1, the elaboration of a design hierarchy takes place in three steps: the *instantiation*, the *signal net list updation*, and the *connection*. After identifying the top-most design entity, objects representing the entire design hierarchy are created in the first step. In the second step, all information related to discrete signals is recorded. This includes fanout calculation and creation of drivers for the signals, and association of processes with signal drivers. Finally, the signal information is passed on to the instantiated components and processes. This information is required to record data such as signal source tree and up-down type conversion functions specified in the model.

The objective of Phase 2 is to identify the different unknown quantities in the design and to form CEs in order to accurately simulate the behavior of the continuous system. In this phase, the design is processed in four steps. In the first step, the unknown quantities are identified across the design entities. Solvability checks to verify the correctness of the model are applied. Secondly, the interface terminals and quantities in the design are connected following the semantics specified in (IEEE Computer Society 1997). In the next step, all the break statements are processed to identify discontinuity augmentation sets in the system. This is performed to accurately detect and process discontinuities in a mixed-signal system (Mayiladuthurai 1998). Lastly, the CEs are formed from the simultaneous statements.

5 PROCESS BASED CONTINUOUS SIMULATION

Continuous simulation finds for every time point, a solution for the unknowns using the set of CEs in the simulation model. At each time point, the basic set of CEs and possibly an augmentation set are evaluated by the analog solver to determine the quantity values(Bakalar and Christen 1997). A number of numerical approaches exploit matrix properties and rely on matrix data structures to perform this task (Banerjee 1994; Buchanan and Turner 1992). These methods have sufficient accuracy, but their solution time is of the order of N^3 (where 'N' is the number of unknowns in the system). Sparse matrix methods can be applied to reduce memory requirements, however this does not affect the complexity of the algorithm. An approach to reduce the computation for conserved systems is the Modified Nodal Analysis (Vlach and Singhal 1994), wherein the number of unknowns is reduced.

Reducing the size of matrix is the goal of recent research oriented towards improving the efficiency of analog and mixed-signal simulation. For example, the matrix can be partitioned and each partition can solved independently. Considerable simulation execution speedup could be obtained by parallel execution of these independent matrices. On the other hand, the theoretical gains obtained by parallelizing the analog simulation is mitigated by 1) the amount of process concurrency present in the model, and 2) the communication overhead imposed by sharing variables and events between parallel processes. Several partitioning schemes have been analyzed to study these factors (Smith et al. 1987). Of these natural partitioning is of special interest in VHDL-AMS as the improvement is significant and the required model properties are frequently available. It is based on grouping and solving for the unknowns occurring in a connected set of CEs. The set of CEs represent the DAEs that are necessary and sufficient to solve for the unknowns in the set. The critieria for grouping depends entirely on the characteristics of the equations represented in the model. A single connected CE set is referred to as an *analog island* in the perspective of a designer and as a continuous process in the context of the simulator. The union of all analog islands represents the complete continuous model, which may or may not communicate with discrete-event processes. Partitioning is performed such that no two analog islands may communicate during simulation.

The formation of analog islands is performed after elaboration when all model equations are available. Besides the CEs forming an analog island, the interface functions, viz break conditions and quantities, are responsible for the communication between the discrete-event and the continuous processes. The design of this simulation environment results in a low-communication overhead simulation kernel suitable for executing on clusters of workstations. Communication is reduced to only discrete events; no inter-CPU communication is required during the numerical computation. The island approach provides similar simulation efficiency on shared memory machines as well.

6 PROPERTIES OF THE EQUATION SET

The set of VHDL-AMS equations assigned to an analog island have three attributes that are critically important for efficient simulation: generality, completeness and conditionality.

$$F(\vec{x}, \frac{d\vec{x}}{dt}, t) = 0 \quad \text{for} \quad t \geq 0 \tag{1}$$

VHDL-AMS enables the modeler to state arbitrary equations in the form of Equation 1. SPICE-like solvers support only a fixed set of equations. To support dynamic equation sets, some form of symbolic differentiation is required during elaboration, and the fixed set simulator must be augmented to form CE sets prior to the solution process. SEAMS takes advantage of an automatic differentiation package called ADOL-C (Griewank et al. 1996) to provide the time derivatives during runtime. Though computationally intensive, it provides flexibility for solving any type of equation set.

A set of equations is considered to be complete if they constitute a solvable system (IEEE Computer Society 1997). Finally, conditionally changing equation sets enable the modeler to dynamically modify the topology of the model over time. For example, a transistor operates in one of three different regions of operation depending on the gate-source voltage. A conditional equation is used to identify which operating region is in effect during simulation. Further, the transition from one region to another may be discontinuous in time. If such is the case, the operating point of the system is reset by performing a DC analysis, and then the transient simulation continues using the modified DAE set. The expression evaluation in the condition may depend on quantities or discrete events.

The representation of discontinuous conditions is easily accommodated in the analog island approach. As with conditional statements, the break condition might depend on quantities and/or discrete events. However, on the occurrence of a break condition, discontinuity processing as stated earlier must be performed. The concept of analog islands provides a very efficient environment for break statement evaluation. Since all equations necessary for a simulation process are gathered in a single island, the simulation of only this island is interrupted enabling other islands to continue their execution.

7 MULTIPLE SOLUTION METHODS

The Multiple Solution Methods (MSM) approach provides the framework for an optimized mixed-signal simulation paradigm. In the previous sections, we established the need for a general differential equation solver for VHDL-AMS. However, such a solver is slow compared to others which are applicable only to a restricted form of designs. This method selects efficient numerical analysis algorithms and solution approach at each time step based upon the characteristics of the DAE set. General algorithms are a part of the MSM algorithm suite thereby providing a solver for even the most generic DAE set.

Figure 3 illustrates the standard steps of the direct simulation method in the main column. The arrows indicate some of the numerous algorithms for the individual steps. Each one has it's specific attributes and may be favorable under certain conditions of the equation set. Special relaxation based approaches provide single step solutions if the characteristics of the DAE set satisfies their needed properties. The DAE set is statically and/or dynamically evaluated and a classification is determined. Based upon the classification, a suitable solver or set of numerical methods is chosen. It is partially based on matrix properties such as symmetry, positive definiteness, and diagonal dominance and other numerical properties that guarantee convergence. In addition, very efficient algorithms apply if the equation set is linear. Finally, characteristics such as memory requirements and computation efficiency are used to assist in dynamic algorithm selection. Barret et al. (1994) discuss various methods and some mathematical conditions for selecting a given algorithm.

Two different paradigms to formulate the matrix are provided viz MNA and Tableau approach. The manual selection is an example of static evaluation of the solver algorithm to be applied. However, it can be automated in the elaboration phase. The present implementation of MSM automatically determines at each time step if the dynamic DAE set is linear or non-linear. In the former case, only a linear solution method is used which greatly improves the simulation time. In the latter case, the Newton-Raphson method is employed and automatic differentiation is used to determine the Jacobian values. If the DAE is non-linear, the entire MNA or Tableau method is used.

The effectiveness of the MSM approach is illustrated in the following section. The MSM approach is not yet complete and is the subject of ongoing research. We intend to expand the simulation environment to provide a wide set of solution methods and DAE classification metrics. Since the synchronization protocols of SEAMS provides complete independence from the parallel distributed domain, parallel

Table 1: Characteristics of examples

Adder Model	Total Number of Design Units	CEs	Unknowns	max. Partitions
Single stage	1	7	15	1
Two stage	3	14	30	2
Three stage	4	21	45	3
Four stage	5	28	60	4

Table 2: Elaboration and simulation times

Model Descripn (Adder)	Without Partitioning		With Partitioning	
	Elab. Time(sec)	Exec. Time(s)	Elab. Time(sec)	Exec. Time(s)
Single stage	0.43	81	0.43	81
Two stage	0.66	1163	1.03	474
Three stage	1.11	4337	1.23	1098
Four stage	0.92	8100	1.19	2295

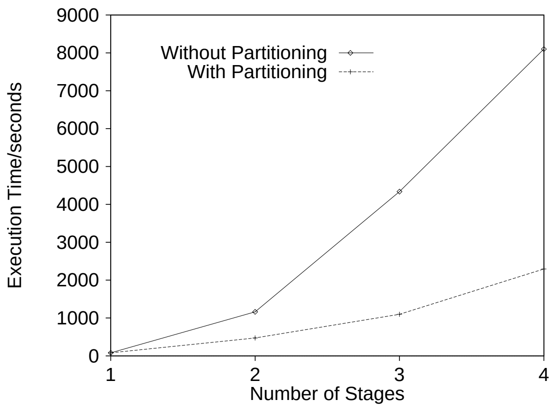

Figure 5: Improvement from design partitioning

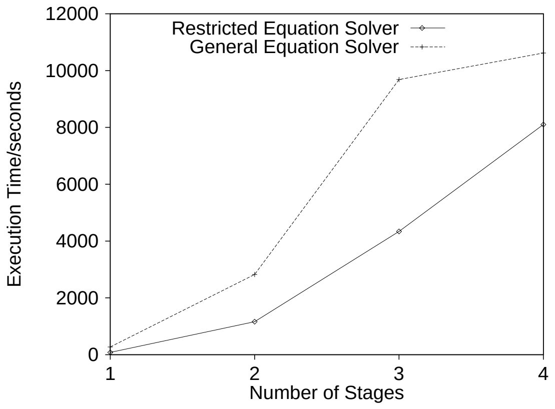

Figure 6: Changing the Solvers

forms of the solver algorithms are also under investigation to enhance the execution speed.

8 PERFORMANCE EVALUATION

The issue of resource requirements to perform simulation of today's large simulation models is addressed through the capability of running SEAMS in parallel on a network of workstations. As networks exhibit rather high communication costs, the simulator is designed to reduce communication. This design issue restricts the execution of DAE solvers onto a single node in the network. Local parallelization in case of a shared memory node is possible, but not supported yet. In addition, discrete-event processes can be grouped on a single node to increase granularity and reduce communication overhead. The optimistic distributed discrete-event simulation kernel WARPED (Martin et al. 1995) enables the distribution of objects onto the machines provided. Synchronization protocols allow the differential equation solvers to be treated the same way as discrete-processes. Two intermediate levels, TyVIS and a set of differential equation solvers provide the functionality to support VHDL-AMS. To evaluate the performance of a VHDL-AMS simulator, three fundamental properties have to be considered - viz correctness, efficiency and the capability to perform the simulation. Correctness is defined by the semantics of VHDL-AMS language and has to be ensured by the implemented algorithms. Efficiency is measured in terms of simulation speed. Performance capability includes issues such as resource requirements and language support. The performance improvement gained by taking advantage of natural partitioning is illustrated by executing an adder circuit model in which several wires have an additional delay introduced by a simple analog delay line. These lines have the property of being independent of each other and the complete circuit is scalable by increasing the number of adder stages. The characteristics of the model for the individual stages are provided in Table 1. The model was simulated from 0 to 10 microseconds. SEAMS was executed four times for the model described in Table 1 , once for each of the adder configurations. The results of this simulation are shown in Table 2 and plotted in Figure 5. SEAMS was executed on a single workstation and each analog island was executed in a separate process. Considerable improvement was obtained by partitioning into analog islands. The elaboration times show that partitioning contributes little overhead.

To show the impact of providing MSM approach, the following two performance studies are described. In the first case, two different solvers are used - a MNA solver and a Tableau solver. The MNA solver restricts the form of the DAE set but is known to execute faster than the more general Tableau method. Table 3 and Figure 6 show that even this slight restriction in the equation set

Table 3: Performance influence of changing solvers

Model Description (Adder)	Without Partitioning	
	Restricted Solver Time (sec)	General Solver Time (sec)
Single stage	81	271
Two stage	1163	2824
Three stage	4337	9685
Four stage	8100	10619

Table 4: Performance changing algorithms

Solver	Simulation Time (seconds)	Execution Time (seconds)	Speedup
Non-MSM	5e-4	8.69	24.15 %
MSM	5e-4	6.59	

enables much faster simulation. Currently, the specific solver is selected manually in SEAMS. Another method is the possibility of reducing the complexity of the general simulation algorithm. Table 4 shows such an optimization approach (available in the MSM environment). In this case, a simple model containing a diode is simulated. As the diode is modeled in one region as a linear model, whereas it is non-linear in another region the general approach must have the Newton-Raphson method applied to handle the equation set. Taking advantage of the changing properties is infact improving the system considerably.

9 CONCLUSIONS

Providing a simulation environment which is able to simulate complete VHDL-AMS models is a challenging task. The modeling capacity of VHDL-AMS, especially in case of differential equations and the mixed-signal interface functionalities, requires a general simulation paradigm. As shown, general differential equation solvers exhibit poor performance. The MSM approach proposed in SEAMS, enables the use of more optimized algorithms, which can be selected during run-time or statically to increase simulation performance.

SEAMS and it's elaboration schemes take advantage of natural partitioning of the input VHDL-AMS design. The reduction in the the matrix size is considerable and with a complexity $O(n^3)$ of differential equation solvers not neglectable. In addition, as SEAMS provides the capabilities of parallel simulation the distribution of memory requirements is important. Since parallel execution is always affected by network latency the approach in SEAMS is designed to reduce network communication. SEAMS provides the flexibility and extensibility to support all issues involved in VHDL-AMS simulation. The question of efficiency concerning a VHDL-AMS simulator remains open as language coverage and simulation performance needs to be considered.

ACKNOWLEDGEMENTS

This work is supported in part by the DARPA Contract grant J-FBI-93-116.

Bakalar, K. and E. Christen (1997). VHDL 1076.1: The Design of a Language Extenstion for VHDL. In *VHDL: The Next 10 Years*, pp. 119–127.

Banerjee, P. (1994). *Parallel Algorithms for VLSI Computer-Aided Design*. Prentice Hall - Englewood Cliffs, New Jersey.

Buchanan, J. L. and P. R. Turner (1992). *Numerical Methods and analysis*. McGraw Hill.

Carter, H. W. (1998). Seams: A simulation environment for vhdl-ams. (available on the www at `http://www.ece.uc.edu/~mistie/`).

Frey, P., R. Radhakrishnan, H. W. Carter, and P. A. Wilsey (1998). Optimistic synchronization of mixed-mode simulators. In *1998 International Parallel Processing Symposium, IPPS'98*.

Fujimoto, R. (1990). Parallel discrete event simulation. *Communications of the ACM 33*(10), 30–53.

Griewank, A., D. Juedes, and J. Utke (1996, September). *ADOL-C: A Package for the Automatic Differentiation of Algorithms written in C/C++*. Version 1.7.

IEEE Computer Society (1997). *IEEE Draft Standard VHDL-AMS Language Reference Manual*. IEEE Computer Society.

Lelarasmee, E., A. E. Ruehli, and A. L. Sangiovanni-Vincentelli (1982). The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems CAD-1*(3), 131–145.

Martin, D. E., T. McBrayer, and P. A. Wilsey (1995). WARPED: A time warp simulation kernel for analysis and application development. (available on the www at `http://www.ece.uc.edu/~paw/warped/`).

Mayiladuthurai, R. S. (1998). Processing Discontinuities and SPICE modeling in VHDL-AMS. Master's thesis, University of Cincinnati.

R.Barrett, M.Berry, T.F.Chan, J.Demmel, J.Donato, J.Dongarra, V.Eijkhout, R.Pozo, C.Romine, and H. V. der Vorst (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Philadelphia, PA: SIAM.

Saleh, R. A., S. Jou, D. Overhauser, X. Xu, and Y. Wang (1994, February). Benchmark circuits for mixed-mode simulators. In *IEEE Custom Integrated Circuits Conference*, pp. 21.1.1–21.1.8.

Schmerler, S., Y. Tanurban, and K. D. Müller-Glaser (1995). A backplane for mixed-mode cosimulation. In *EUROSIM '95*, pp. 499–504.

Smith, S. P., B. Underwood, and M. R. Mercer (1987). An analysis of several approaches to circuit simulation for parallel logic simulation. In *Proceedings of the International Conference on Computer Design*, pp. 664–667.

Soulé, L. P. and A. Gupta (1991, October). An evaluation of the Chandy-Misra-Bryant algorithm for digital logic simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 1*(4), 308–347.

Vlach, J. and K. Singhal (1994). *Computer Methods for Circuit Analysis and Design*. New York, NY: Van Nostrand Reinhold.

Zeigler, B. P. (1976). *Theory of Modelling and Simulation*. New York: John Wiley & Sons, Inc.

AUTHOR BIOGRAPHIES

PETER FREY Dipl.-Ing.(FH) is a senior Ph.D. student in the Distributed Processing Laboratory at the University of Cincinnati. He received his Dipl.-Ing.(FH) degree in Computer Engineering from Fachhochschule Ulm, Ulm, Germany. His research interests include: Mixed-Mode Simulation, Parallel Discrete Event Simulation, Formal Software Specification, and Hardware Description Languages.

KATHIRESAN NELLAYAPPAN is a software development engineer in the Analog/Mixed Signal simulation group at Mentor Graphics Corporation, USA. He received his B.E degree in Computer Science & Engineering from the University of Madras, India and his MS degree in Computer Engineering from the University of Cincinnati, USA.

VASUDEVAN V. SHANMUGASUNDARAM is currently with Intel Corporation, DuPont, USA as a Component Design Engineer. He received his Masters degree in Computer Engineering from the University of Cincinnati, and Bachelors degree in Electronics and Communication Engineering from Bharathiar University, Coimbatore, India.

RAMESH S. MAYILADUTHURAI is currently with Cadence Design Systems (Massachussetts) as a Member of Technical Staff. He received his Masters in Computer Engineering from the University of Cincinnati and B.E. in Computer Science from the University of Madras, India.

CHANDRASHEKAR L. CHETPUT is currently a member of technical staff with Cadence Design Systems. He received his Masters in Computer Engineering from the University of Cincinnati in 1998. He got his B.E in Computer Science from Coimbatore Institute of Technology, India in 1995.

HAROLD W. CARTER, Ph.D.(EE), MSEE, BSEE, is a professor of Electrical and Computer Engineering at the University of Cincinnati where he also serves as Associate Head for Computer Engineering. He directs the Electronic Design Automation Research Center and the Computer Engineering Resesarch Consortium in Ohio. With his students, he performs and coordinates research in mixed-technology simulation, hardware/software co-design, and adaptive computing algorithms. He is a member of the IEEE Computer Society.