VHDL-AMS Modeling and Compilation for Parallel Mixed-Mode Simulation

Bojan Anđelković, Marko Dimitrijević and Milunka Damnjanović

Abstract – In this paper a survey of recommended VHDL-AMS modeling practices for effective parallel simulation is given. Also, a need for extension of VHDL-AMS standard is considered in order to enable the designer to influence parallel simulation performance during the model development. Mixed-mode circuit partitioning algorithms that should be applied during the model compilation are presented.

Keywords – Parallel mixed-mode simulation, VHDL-AMS

I. Introduction

Today's high end integrated circuits may contain millions of transistors and becoming increasingly complex in the diversity of devices (embedded software, micro-electro-mechanical components). Having in mind the rapid growth of the electronic systems complexity and short time-to-market demands, the simulation has become time consuming and represents a bottleneck in the design flow. The modern simulation is based on system modeling in one of standard Hardware Description Languages (HDL), such as VHDL-AMS [1], which support an easy description of mixed-signal VLSI systems with various portions described at different abstraction levels. VHDL-AMS simulation is a compute intensive, particularly for large models. In order to accelerate the simulation process and reduce very long simulation runtimes of such HDL descriptions parallel simulation tools can be used [2]. Parallel simulation is based on splitting the circuit into several pieces and simulating them in parallel one piece per workstation. The potentially larger amount of the memory in workstation clusters will enable the execution of larger simulation models.

This paper contains some VHDL-AMS modeling recommendations necessary to achieve high-performance parallel simulation. Also, existing partitioning methods that should be performed during the VHDL-AMS model compilation are presented. A need for extension of VHDL-AMS in order to adapt it for the use in parallel simulations is considered.

A brief overview of VHDL-AMS modeling features is presented in Section II. Also, some modeling recommendations for the development of VHDL-AMS models for parallel simulations are given. Mixed-mode circuit partitioning techniques performed during the VHDL-AMS model compilation are discussed in Section III. Section IV summarizes the presented survey of parallel modelling recommendations and partitioning algorithms together with some future work directions.

Bojan Anđelković, Marko Dimitrijević and Milunka Damnjanović are with the Department of Electronics, Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia and Montenegro, E-mail: (abojan, marko, mila)@elfak.ni.ac.yu.

II. VHDL-AMS Modeling Features

A. VHDL-AMS Model Description

VHDL-AMS is an Analog and Mixed-Signal extension to the Very-High-Speed-Integrated-Circuits Hardware Description Language (VHDL). It is a combination of the base IEEE Std 1076-1993 (VHDL) standard and IEEE 1076.1-1999. (analog and mixed-signal extensions) [3]. VHDL-AMS is an internal name, widely used in the designers' community, for the combination of these standards.

VHDL-AMS provides behavioral and structural description of both discrete and continuous systems. Being a superset of VHDL it inherited all its advantages such as structural and functional decomposition, separate compilation, a powerful sequential notation, and the strong type system of a modern programming language. The discrete models are specified using component instantiations, concurrent signal assignment and the process statement. Modeling of continuous systems is based on the theory of Differential and Algebraic Equations (DAEs). DAE-based continuous systems can be modeled similar to discrete models at several hierarchical levels. A special notation for DAE's is introduced describing precisely what system of equations is implied at each simulation time. VHDL-AMS also has the ability to describe non-electrical physical phenomena. Mixed-discipline models with different domains such as electrical, physical, and thermal can be described and simulated in a single entity.

A structure of a VHDL-AMS model and an overview of the language elements and statements are given with the help of a simple RC circuit model with a pre charged capacitor shown in Fig. 1 [4].

For representing the unknown continuous variables in the system of DAEs, VHDL-AMS introduces a new class of objects, the *quantity* [5]. Quantities can also be declared as ports of the model (the points that can be connected to other models). Additional branch quantities are provided to support conservation semantics of the systems like electrical circuits. There are two kinds of branch quantities: across quantities representing effort like effects such as

voltage or temperature, and through quantities for flow like effects such as current and fluid flow rate. In the RC circuit example, the branch quantities are capacitor and resistor voltage and current. They are declared with reference to two *terminals*. Terminals can be of different *natures* that represent distinct energy domains (electrical, thermal, etc.). Using of terminals as ports of the model enables constructing nodes on hierarchical descriptions when such model is instantiated.

```
-- Set initial value (pre charged capacitor)

LIBRARY DISCIPLINES;
LIBRARY IEEE;

USE DISCIPLINES.ELECTROMAGNETIC_SYSTEM.ALL;
USE IEEE.MATH_REAL.ALL;

ENTITY RC IS
END;

ARCHITECTURE behav OF RC IS
    TERMINAL n1,n2: ELECTRICAL;
    QUANTITY v_in ACROSS i_in THROUGH  n1 TO electrical_ground;
    QUANTITY u_r ACROSS i_r THROUGH n1 TO n2;
    QUANTITY u_c ACROSS i_c THROUGH n2 TO electrical_ground;
BEGIN


    BREAK  u_c => 0.5;       --initvalue

    v_in == 1.0;      --constant  voltage source
    i_r  == u_r  / 1000.0;      --resistor  equation
    i_c  == 1.0e-6 * u_c'dot;   --capacitor equation
END;
```

Fig. 1. VHDL-AMS model of an RC circuit

The system of DAEs can be described using *simultaneous statements*. These statements express the system behaviour by specifying relationships between quantities. The language supports two quantity attributes (*'dot* and *'integ*) to specify derivatives and integrations over time, respectively. VHDL-AMS also provides two special simultaneous statements, called *simultaneous if* and *simultaneous case*, to change the set of equations. These statements include conditional expressions and depending of their satisfaction appropriate set of equations is solved.

A special construct called the *break* statement is used to represent discontinuities in VHDL-AMS model descriptions. It specifies new initial conditions and the possible occurrence of a discontinuity. The model developer should provide the necessary break statements to notify the simulator of possible discontinuities. In the example, break statement is used to setup initial value for capacitor voltage.

Since DAE solvers use numerical algorithms to solve the equation systems, VHDL-AMS enables the designer to specify individual tolerances for quantities which must be satisfied by the simulator. It gives the designer an opportunity to trade between accuracy and simulation speed, as more accurate solutions require longer simulation runtimes.

VHDL-AMS is developed as a universal and tool independent language for modeling and documentation of both analog and digital devices and physical subsystems from other domains. That enables the designer to focus on the model equations without distraction of specific simulator internals. In that way they can create models with different abstraction levels that improve significantly simulation speed.

B. Modeling Recommendations for Parallel Simulation

A set of recommended VHDL-AMS modeling practices to help designers achieve effective parallel simulation is given in [6]. Performances of VHDL-AMS parallel mixed-mode simulation depend on hardware processing environment (number of processors, interconnecting network), VHDL-AMS compiling techniques, and VHDL-AMS models used for simulation. Some model features, such as model complexity, model abstraction and stimulus influence parallel simulation performance, but the designer cannot do anything to improve them. However, there are some modeling issues that can be under model developer's control. These modeling issues include simulation time resolution, number of processes, data types, and shared variables.

VHDL-AMS gives the designer an opportunity to develop models at many levels of abstraction. In sequential simulation executing on a single processor, less abstract models increase simulation time. However, when parallel simulation is used, increasing the number of details in a model can yield more units capable of executing concurrently. In that way, when more detailed models are used, it is possible to achieve parallel simulation speedup comparing to a uni-processor. When such models are developed for parallel simulation, it is necessary to enable de-coupling between model elements in order to improve simulation performance.

The use of physically-oriented delay or parasitic information (such as SDF and VITAL) during VHDL-AMS simulation decreases parallel simulation performance. These timing models distribute evaluations at a wider range of time instants in the simulation time domain and that decreases parallelism. Better performances can be achieved by reducing the precision of delay and other timing parameters in the model.

In order to improve parallel simulation performance, VHDL-AMS models should consist of more processes, tasks, or their equivalents than there are processors executing simulation.

The transmitting events between processors on a parallel computer influence the simulation performance more than when the communication is performed in the memory subsystem of single-processor architecture. Therefore, if it is necessary to transmit more than one scalar signal value at the same time between different processors (i.e. workstations on a computer cluster), parallel simulation performance can be improved by aggregating these scalars into a single composite event. Such aggregation can be automatically implemented during the VHDL-AMS compilation process, as described in [7],

or the designer can implement such aggregation during the development of VHDL-AMS model source code.

Shared variables added in VHDL'93 standard can decrease parallel simulation efficiency, because they require communication between two or more processes in which they are used. Therefore, the model developer should avoid use of variables referenced by more than one process. If such variables must be used it is necessary to maximize the variable's locality to as few processes as possible.

Besides mentioned discrete-event modeling recommendations, there are also recommended continuous-domain modeling practices given in [6]. The parallel VHDL-AMS model developer can maximize simulation performance of analog and mixed-signal systems through appropriate use of circuit decomposition, tolerances and discontinuities.

In order to reduce communication latency during parallel simulation, large mixed-signal systems should be described as a collection of smaller continuous-domain models completely surrounded by discrete-event models.

As mentioned in the previous subsection, in VHDL-AMS quantity tolerances that must be satisfied during the equation system solving, can be specified. It is recommended to use broader tolerances, because in that way it is easier to divide the parallel solver among different cluster nodes by facilitating convergence of independent parts of the solution. However, broad tolerances give less accurate simulation results, so a compromise between parallel simulation performance and solution accuracy should be made.

Situations in which implicit or explicit value discontinuities occur in a quantity should be avoided. Use of VHDL-AMS ramp and slew quantity attributes reduces the impact of discontinuities on parallel simulation.

III. VHDL-AMS COMPILATION FOR PARALLEL SIMULATION

A. Parallel Mixed-Mode Simulation

A survey of parallel mixed-mode simulation algorithms and implementations is given in [2]. Mixed-mode designs are particularly convenient for parallel simulation, since they are by default partitioned into analog and digital parts that can be executed concurrently. Therefore, a parallel mixed-mode simulator consists of two distinct simulation kernels: continuous-time differential equation simulation kernel and discrete-event simulation kernel. In order to achieve parallel mixed-mode simulation, appropriate synchronization protocols [8] between these analog and digital simulation kernels are necessary.

Digital components in parallel discrete-event simulation are modeled as a collection of concurrently executing logical processes (LPs) that communicate via message passing. LPs can be assigned to different cluster nodes and distribute the simulation across the network of workstations.

The domain decomposition technique described in [9] is often used for parallelization of differential equation solvers.

B. Mixed-Mode-Circuit Partitioning

High-performance parallel simulation requires partitioning the simulation model into several parts in order to simulate one part per processor. Electronic circuits have natural clustering that can be used to achieve good partitioning results.

The partitioning methods are based on either parallelism in the simulation algorithm or in the circuit being simulated. Partitioning based on simulation algorithm is limited by the properties of specific algorithm used for simulation. The later method exploits the concurrency and parallelism in the circuit structure in order to minimize communication between cluster nodes and balance the nodes workload.

In parallel discrete-event simulation (PDES), the system under simulation is modeled as a collection of concurrently executing logical processes (LPs) that communicate via message passing. LPs then maybe assigned to different cluster nodes, thus distributing the simulation across the network of workstations.

Many of the partitioning algorithms for parallel logic simulation are based on a directed graph representation of the simulated circuit. In such representation the vertices of the graph denote logic components while edges represent signals. In [10] a multilevel approach to partitioning is proposed. It optimizes all factors for improving parallel logic simulation by decoupling them into separate phases.

The main objective in partitioning is to reduce the simulation time through equal load balance of processors and low communication overhead. Since, interconnecting signals between partitions cause time consuming communication, it is necessary to achieve a small number of signals connecting the partitions.

The partitioning should also provide equal workload for all slave nodes which enables optimal distribution of simulation effort. In order to estimate the workload, each circuit element can be assigned a weight according to its simulation complexity. One such partitioning method is COPART [11] implemented in TITAN parallel transistor level simulator. It reads the circuit description in SPICE and after partitioning a SPICE netlist for each partition is created, which is required for parallel simulation. In this method the structure of the circuit is mapped to an undirected graph, where the nodes represent the circuit elements and the edges are the signals connecting them. Appropriate weights for the nodes and the edges are assigned representing the simulation effort needed to simulate the element (i.e. the node in the graph). Signals connected to a lot of elements are widely distributed over the circuit and their cutting cannot be avoided during the partitioning. Therefore, such signals have lower weights

meaning lower significance during the partitioning. The partitioning process is focused on signals connected to a few elements to avoid cutting them. Similar workload for each cluster node is achieved if the element weight sums are similar for all partitions. As the partition simulations are synchronized by the master process, a master circuit description with special references to the cut nets is also created.

Mixed-signal VHDL-AMS simulator SEAMS [12] exploits component-level partitioning and *analog island* modeling to completely partition the VHDL-AMS description at the entity-architecture level and simulate the partitioned system on a workstation cluster. SEAMS analyzes the VHDL-AMS model description and generates a set of characteristic expressions (CEs) from the simultaneous statements. CEs gover the behavior of the continuous part of the mixed-signal system. Partitioning of a VHDL-AMS model, implemented in SEAMS, is based on grouping and solving for the unknowns occurring in a connected set of CEs. The set of CEs consists of equations that are sufficient to determine the unknowns in the set. The criterion for grouping depends on the characteristics of the equations in the model. A single set of CEs is called *analog island*. The objective in partitioning is that no two analog islands may communicate during simulation. Break conditions and quantities are used for the communication between the discrete-event and the continuous processes.

As it can be seen, only partitioning algorithms which guarantee a minimum communication overhead may be able to provide good speed-up on workstation cluster. The mapping of LPs to workstations can be performed automatically based on some partitioning and load-balancing heuristics. However, automatically parallelized code generally cannot achieve good speed-up. Sometimes the model developer is in much better position to know the runtime characteristics of the processes and would want to manually perform partitioning and load-balancing. Therefore special parallel simulation languages, such as Parsec [13] have been developed. They allow user to influence parallel simulation performances during the model development. Parsec is a C-based PDES language that allows the model developer to specify the specific node on which an entity will be created by using an appropriate option during entity creation. It enables that model can be partitioned in such way that message communication between nodes is minimized, with balanced computation load. Also, the language has special constructs to specify communication between entities and control the execution of all processes [14]. Each entity will execute its own tasks, and if necessary, communicate with one another by sending and receiving messages. One code example in Parsec is shown in Fig. 2 [14]. In this example the entity *worker* performs some tasks, then receives a *storeRequest* message from another entity, stores the message, and resumes working on another task.

Similar constructs can be added to VHDL-AMS in order to extend the standard language and allow the user flexibility to influence simulation performance during the model development. Desired features for PDES languages can be found in [14] and can be used as a basis in the development of VHDL-AMS extensions for parallel simulations.

```
entity worker {
  ...
  doTask();
  receive (storeRequest msg) {
     storeMessage(msg);
  }
  doMoreTask();
}
```

Fig. 2. Parsec code example

IV. CONCLUSION

VHDL-AMS is a standard mixed-signal HDL that promises to play very important role in the specification and verification of mixed-signal systems. Since it is tool and vendor independent, it has advantages over other simulator specific parallel simulation languages. However, in order to use VHDL-AMS for parallel simulation some constructs and elements from parallel simulation languages should be added into VHDL-AMS. It would give the designers an opportunity to take advantage of standard HDL together with ability to have more control over parallel simulation performance during the model development. Also, the designer should follow presented modeling recommendations to achieve effective parallel simulation. During the compilation of VHDL-AMS models, appropriate partitioning techniques should be applied to minimize communication between cluster nodes and enable equal workload.

REFERENCES

[1] Christen, E., Bakalar, K., "*VHDL-AMS – A Hardware Description Language for Analog and Mixed-Signal Applications*", IEEE Trans. CAS, Vol. 46, No. 10, Oct., 1999., pp. 1263-1272.

[2] Savić, M., Anđelković, B., Litovski, V., "*Parallel Mixed-Mode Simulation – Preliminary Study*", Proc. of V Symposium on Industrial Electronics INDEL 2004, Nov., 2004, pp. 76-79

[3] IEEE Computer Society, "*IEEE Standard VHDL-AMS Language Reference Manual*", IEEE Computer Society, 1999.

[4] http://www.hamster-ams.com

[5] Ashenden, P., Peterson, G., Teegarden, D., "*The System Designer's Guide to VHDL-AMS*", Morgan Kaufmann Publishers, San Francisco, 2003.

[6] Peterson, G., Willis, J., "*High-Performance Hardware*

Description Language Simulation: Modelling Issues and Recommended Practices", Trans. of The Society for Computer Simulation International, Vol. 16, No. 1, 1999., pp. 6-15.

[7] Willis, J., Siewiorek, D., "*Optimizing VHDL Compilation for Parallel Simulation*", IEEE Design & Test of Computers, Sep., 1992, pp.

[8] Frey, P., Radhakrishnan, R., "*Parallel Mixed-Technology Simulation*", Proc. of the 14th Workshop on Parallel and Distributed Simulation PADS'00, May, 2000, pp. 7-14.

[9] Fröhlich, N., Riess, B., Wever, U., Zheng, Q., "*A New Approach for Parallel Simulation of VLSI Circuits on a Transistor Level*", IEEE Trans. CAS, Vol. 45, No. 6, Jun, 1998, pp. 601-613

[10] Subramanian, S., Rao, D., Wilsey, P., "*Study of a Multilevel Approach to Partitioning for Parallel Logic Simulation*", 14th International Parallel and Distributed Processing Symposium, May, 2000, pp. 833-836

[11] Fröhlich, N., Riess, B., Wever, U., Zheng, Q., "*A New Partitioning Method for Parallel Simulation of VLSI Circuits on Transistor Level*", Proc. of Design, Automation and Test in Europe, 2000, pp. 679-684.

[12] Frey, P., Nellayappan, K., Shanmugasundaram, V., Mayiladuthurai, R., Chandrashekar, C., Carter, H., "*SEAMS: Simulation Environment for VHDL-AMS*", Proc. of the 1998 Winter Simulation Conference, Dec., 1998, pp. 539-546.

[13] Bagrodia, R., Meyer, R., Takai, M., Chen, Y., Zeng, X., Martin, J., Song, H., "*Parsec: A Parallel Simulation Environment for Complex Systems*", Computer, Vol. 31, No. 10, Oct., 1998, pp. 77-85

[14] Low, Y., Lim, C., Wentong, C., Huang, S., Hsu, W., Jain, S., Turner, S., "*Survey of Languages and Runtime Libraries for Parallel Discrete-Event Simulation*", Simulation, Vol. 72, No. 3, March, 1999, pp. 170-186