1. Introduction

Electronic circuit simulation constantly attracts attention of the scientific community. Algorithms and methods that proved to be efficient for circuits with 100000 gates, run out of steam when faced with 1 million gates, while others cannot cope with heterogeneous circuits with micro-mechanical and/or optical devices and systems, or hardware/software co-simulation problems. It is well known that circuit simulation is very resource-intensive and that requirements for simulation of modern electronic circuits are always one step ahead of the state-of-the-art memory and CPU capabilities. Modern ASIC (Application Specific Integrated Circuits) more and more frequently contain both analog and logic subsystems, embedded software, and are sometimes used with optical and/or micro-mechanical devices. To be able to handle an entire system, a modern simulator must have the ability to express all these kinds of sub-systems in the most efficient way, with a desired level of accuracy. As an expected consequence, it is likely that the simulator itself will incorporate several different algorithms and concepts, resembling the heterogeneous nature of simulation problems.

If we assume that non-electrical devices may be expressed using sets of nonlinear differential equations, the problem of modern VLSI ASIC chip simulation essentially requires two different algorithms - one for solving the sets of

nonlinear differential equations and one for discrete-event simulation. Thus, analog electrical and non-electrical portions of a system are analyzed using accurate and detailed, but quite slow and resource-intensive SPICE-like algorithms [Nage75], while logic subsystems are expressed using logic values and states, with reverse accuracy and efficiency figures. Besides, simulator should be able to interpret software routines during the simulation run, to enable description of systems with embedded software (SoC – system-on-a-chip). It is therefore necessary to have a behavioral simulator that handles mixed-mode systems, and a powerful hardware description language that enables description of both hardware modules and software routines..

There are several simulators available that can handle different classes of circuits, levels of accuracy and generality. In the presence of well-defined and loosely coupled analog and digital domains, it is possible to simulate the system using two separated simulators synchronized with some sort of kernel that can handle inter-process communication [Corm88]. Many commercial CAD frameworks have the option of coupling several different analog and logic simulators for that purpose [Smit92]. However, the approach is not general, since it cannot handle circuits with strong feedback loops. More appropriate is the algorithm that integrates analog and logic subroutines into one simulator. Integrated mixed-signal simulators like DIANA [DeMa80], SPLICE [Newt78], MOTIS [Chen84], SAMSON [Saka85], all combine analog system solving techniques and event-driven, selective trace methods controlled by one central agent to simulate mixed-mode circuits. In this way, large digital portions of ASIC systems may be efficiently simulated using fast logic simulation techniques, while accuracy and details are reserved for isolated analog blocks. It was shown, however, that those analog pieces, being modeled at the transistor level, use most of the simulators' time and resources and degrade the overall performance. It is therefore essential to have behavioral analog macro-modeling capabilities to resolve this bottleneck. Some of the analog and hybrid simulators offer these capabilities through their analog modeling languages, usually named hardware description languages (HDL). Such languages as M of Lsim [Odry86], MAST of SABER [Getr89], ALFA [Kazm92], S++SDL [Brow92], Verilog-AMS [Kund96] and HDL-A [Pabs95, Roch96], which is the working version of the standard VHDL-AMS [Vach95], [IEEE99], [Chri99]. Languages MAST, ALFA, Verilog-AMS and VHDL-AMS also have the ability to model non-electrical devices by expressing their behavior using algebraic and/or ordinary differential equations (ODE). Development of the language MHDL was sponsored by the MDHL Study Group, a part of the IEEE Standards Coordinating Committee - 30 (SSC-30). Verilog-AMS is proposed analog extension to a co-standard IEEE 1364 Verilog language [Thom91], [OVI], and VHDL-AMS is a standard developed by the IEEE 1076.1 committee as the analog extension of VHDL [Lips89]. An overview of this

standardization activity can be found in [Rhod96]. General idea of all the solutions that are under standardization is separate development of the HDL and the simulation engine (engines). The languages are designed to be tool independent. They should support not only the simulation, but also other design tasks such as synthesis, layout generation, etc., which is already the case with digital HDLs (VHDL and Verilog). Nevertheless, automated design of analogue and mixed-signal circuits is on much lower level then of digital, and these languages are still used exclusively for simulation.

In the industrial community, it is already clear that standardization of the analog and mixed-signals extensions of discrete-event languages is not going to solve all problems. There is already a SystemC initiative [SysC], governed by a need to develop a language that is suitable for describing both hardware and software needed for a system-on-a-chip (SoC) designs. Importance of having C/C++ based modeling environment is recognized by the industry [Bore99].

Languages as VHDL-AMS and Verilog-AMS are intended to be universal tools for modeling and documentation of both analog and digital devices and physical models from other domains (mixed-domain simulation). Nevertheless, they are both regarded as suited for design representation of "big-D-little-A" systems, i.e. for systems with more digital than analog content [Anta96]. They are developed as analog extensions of digital (discrete-event) languages. Moreover, they cannot be used for harware/software co-simulation.

Our approach is essentially different. We have developed our mixed-signal simulator Alecsis (**A**nalog and **L**ogic **E**lectronic **C**ircuit **SI**mulation **S**ystem), and the object-oriented HDL named AleC++ as a united system. Alecsis is an integrated **mixed-mode** simulator, which can handle arbitrarily complex combinations of different kinds of devices and subsystems with no limitations concerning closed feedback loops. It can handle various kinds of quantities that appear as physical connections between devices (**mixed-signal**), from analog nodes and logic discrete signals, to non-electrical quantities such as pressure, light, etc. It also enables both analog and logic blocks to be described at any level of abstraction, according to required precision and efficiency. It is not used for electronic circuits only. The generality of the language AleC++ has enabled behavioral description of different physical or abstract systems. Different kinds of systems can be simulated (**mixed-domain** simulation).

AleC++ is designed to be a superset of C++, hence its object-oriented features. However, it can be still used just as another C++ compiler in which case the produced object code is immediately executed or stored into a simulation library to be linked later. One of the main qualities of our simulation system is that AleC++ is an integrated part of Alecsis. The organization of the language will be

explained in more details in the next chapter. It integrates the power of C++, the ability to model concurrent processes and VHDL-like typed signals, and compatibility with the SPICE device models. It also upgrades the SPICE model card concept and proposes a general method for hybrid block parametrization in an object-oriented manner. All HDLs have some characteristics of the programming languages - if-else branches and basic computing is necessary for describing even the simplest models. However, features of a powerful object-oriented language like C++ can be used to advantage in the modeling process. As pointed out in the description of a C++ based simulator Sframe [Melv92, Melv93a, Moin94, Caba99, Li00], inheritance properties and polymorphism can be very helpful in modeling. In AleC++, a model can be derived as a derived class of a base model, which gives an efficient way of model code reuse, and helps significantly in the modeling process. As Alecsis can interpret C/C++ routines while executing hardware models, it is very suited for descriptions of systems with embedded software.

We have developed new HDL, but we could not neglect the fact that the standard for digital system description already exists. In order to enable use of rich VHDL model libraries, we decided to support simulation of digital systems described in VHDL using Alecsis. Alecsis-VHDL co-simulation is achieved using the simulation kernel of Alecsis simulator (virtual processor). We have developed a compiler that converts VHDL source code into the object code for Alecsis virtual processor. Alecsis object code is designed for mixed-mode simulation, and it supports almost all VHDL modeling mechanisms.