Available online at www.sciencedirect.com
ScienceDirect
INTEGRATION, the VLSI journal 40 (2007) 193–208
INTEGRATION the VLSI journal
www.elsevier.com/locate/vlsi

Compiled code simulation of analog and mixed-signal systems using piecewise linear modeling of nonlinear parameters: A case study for $\Delta\Sigma$ modulator simulation ☆

Hui Zhang[a], Simona Doboli[b], Hua Tang[a], Alex Doboli[a,*]

[a]Department of Electrical and Computer Engineering, State University of New York at Stony Brook, NY 11794-2350, USA
[b]Department of Computer Science, Hofstra University, Hempstead, NY 11549, USA

Received 12 May 2005; received in revised form 19 September 2005; accepted 22 September 2005

Abstract

This paper presents a methodology for fast time-domain simulation of analog systems with nonlinear parameters. Specifically, the paper focuses on $\Delta\Sigma$ analog-to-digital converters (ADC). The method creates compiled-code simulators based on symbolic analysis. Code is optimized using loop invariant elimination and constant folding, well-known compiler optimization methods. Circuits are described as structural macromodels. Nonlinear parameters are expressed using piecewise linear (PWL) models. The paper presents a technique for automatically creating PWL models through model extraction from trained neural networks. As compared to existing behavioral simulation methods for $\Delta\Sigma$ ADC, this technique is more systematic and accurate. In our experiments, compiled-code simulation was significantly faster than numerical simulation. Hence, the methodology is very useful in analog and mixed-signal system synthesis, which is known to require a large number of simulation steps.

Keywords: Simulation; Analog and mixed-signal systems; Nonlinear modeling; Piecewise linear; Neural networks

1. Introduction

One bottleneck in system-on-chip (SOC) development is presently the design of RF and analog IP cores, as well as the integration and verification of final designs [1]. Existing research offers remarkable solutions to synthesis of analog circuits, like OpAmps, operational transconductors (OTA), comparators, and so on. Please refer to [1] for a recent overview. Commercial tools, including Virtuoso NeoCircuit and Virtuoso NeoCell from Cadence and Circuit Explorer from Synopsys, are available for transistor sizing and layout design of analog circuits. Obviously, the next step is to tackle synthesis of more complex analog and mixed-signal systems, like analog to digital and digital to analog converters, phase-locked loop circuits, and transceivers. For this endeavor, however, it has been reported that one of the main challenges is the large number of optimization variables that must be simultaneously tackled [2–6]. Existing analog and mixed-signal circuit simulators [7], which are the core of exploration-based synthesis, are still too slow for being used inside the SOC synthesis loop [8,1,9], experience numerous stability problems [8], and are unable to exploit the specifics of circuits and systems [10,11]. Slow system simulation poses additional challenges to exploration-based analog synthesis, which needs a very long time to complete its search or might even not converge towards constraint satisfying solutions.

Behavioral models are used for speeding-up simulation and analysis inside the analog and mixed-signal system synthesis loop. For example, in the methodologies proposed by Dhanwada et al. [2] and Vancoreland et al. [6], behavioral models are used for most of the time to quickly

☆Parts of this paper were published in the Proceedings of the 2003 International Joint Conference on Neural Networks (IJCNN) and 2004 Behavioral Modeling and Simulation Workshop (BMAS).

*Corresponding author. Tel.: +16316321611; fax: +16316328494.
E-mail addresses: huizhang@ece.sunysb.edu (H. Zhang), adoboli@ece.sunysb.edu (A. Doboli), Simona.Doboli@hofstra.edu (S. Doboli).

0167-9260/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2005.09.001

find the performance of the analyzed designs. Periodically, detailed circuit simulation is performed to correct the inaccuracies introduced by behavioral models. Behavioral circuit and system models are of two kinds: structural (physical) and mathematical models. Please refer to [12] for a presentation of the most recent advancements in behavioral modeling and simulation. Structural modeling methods, in general, simplify a circuit to a reduced subcircuit that includes only the dominant devices. Recently, several approaches are reported to automate this process using the signal-path approach [13], root localization method [14], and behavioral model decoupling [15]. In spite of these promising results, structural models are obtained most of the time through manual analysis based on the designer's expertise and experience. Mathematical models capture quantitative relationships between the parameters and performances of a circuit. Mathematical modeling includes linear and nonlinear regression, Volterra series, Pade approximations, wavelet functions, and neural networks (NNs) [12]. Nonlinear regression is traditionally used to produce mathematical models [16–18]. The most important limitations of existing structural and mathematical modeling techniques include difficulties in handling nonlinear parameters as well as in tackling large circuits. Also, large amounts of sampling data are needed for accurate and complete modeling.

In this paper, we propose a novel technique for fast simulation of analog systems with nonlinear parameters. We focus on time-domain simulation even though the method is valid for the frequency domain too. The main idea is to generate optimized code for a simulator that is customized to each individual system. The code generation methodology relies on calculating symbolic expressions for the output voltages and currents, and the state variables of a system. To avoid the large memory requirements specific to symbolic analysis, the suggested method exploits the structural regularities present in the topology of a system netlist [3]. Each kind of interconnection structure *IST* between two blocks b_i and b_j is captured as a separate C++ class $C_{\mathrm{IST}}(b_i, b_j)$ with the related methods encapsulating the symbolic composition rule (SCR) of the two blocks. All instances of structure *IST* present in the system topology are formulated as objects of class C_{IST}, for which the identity of blocks b_i and b_j is set as the two blocks actually appearing in the structural instance. This representation is applied bottom-up, so that blocks b correspond to building blocks and composed blocks in the system. Code generation utilizes detailed structural macromodels for the building blocks, such as OTA, OpAmp, comparators and so on, including non-idealities, like finite gain, poles and zeros, CMRR, phase margin, fall and rise time. Nonlinear parameters are described using piecewise linear (PWL) models. Code optimization identifies and eliminates loop invariants, and propagates constant sub-expressions present in the simulation loop. As a case study, experiments present simulation results for $\Delta\Sigma$ analog to digital converters (ADCs) [19]. As compared to numerical simulation, the presented technique achieves significant simulation time reductions, and has few stability problems, whereas losses in accuracy are minor.

The paper also discusses a new algorithm for extracting PWL models from trained NN. NN are capable to learn any type of nonlinear mapping based on their well-known property of being universal approximators [20]. The proposed method addresses the need of automatically creating PWL models for nonlinear parameters [21]. The model generation techniques starts with NN training. A back-propagation algorithm is used for training until the desired accuracy is obtained at the output of the network. Next, a pruning method is applied to eliminate the neurons and weights with insignificant contributions. Then, the sigmoidal activation function of each hidden neuron is approximated with a PWL function with a variable number of segments. The number of segments, its limits, and the linear approximation on each segment are automatically determined by a clustering algorithm. Finally, the PWL functions for the hidden neurons are composed together to generate the PWL functions of the model output. The regions, where each linear output model is active, are found by iteratively solving a linear system of inequalities and adjusting its limits.

Compared to other fast simulation methods for $\Delta\Sigma$ ADCs [8,9], our technique requires less designer input, and uses more detailed circuit models. Hence, it offers the benefit of more accurate simulation, and thus, potentially, the advent of faster analog design closure. ADC simulation in [8,9] relies on behavioral models, which are reported to be imprecise [8]. Finally, our simulation technique belongs to the class of compiled-code simulators. To the best of our knowledge, this is the fist attempt to develop a general methodology for compiled-code simulation of continuous-time systems with nonlinear parameters. Existing compiled-code simulators [22–26] are for discrete-time and event-driven systems. As opposed to this work, we showed how symbolic analysis can be used to generate simulator code for tightly coupled systems (such as analog and mixed-signal systems), and how nonlinearities can be systematically addressed for simulation. Being customized to a certain application, compiled-code simulators are much faster than their counter-part numerical simulators, and can be optimized to reduce numerical instability.

This paper is organized as six sections. Section 2 details the simulation methodology. Section 3 presents PWL modeling of nonlinear parameters. Next, we discuss the structure of the C++ code for customized simulators, and simulation results are given in Section 5. Finally, Section 6 summarizes our conclusions.

2. Proposed simulation methodology

Fig. 1 shows the proposed compiled-code simulation methodology and the structure of single-loop $\Delta\Sigma$ ADCs [19], used as illustrating examples in our paper. The methodology produces C++ code for customized simulators generated from the system netlist used as input