A Formal Approach for Specification-Driven AMS Behavioral Model Generation

Subhankar Mukherjee*, Antara Ain*, S. K. Panda*, Rajdeep Mukhopadhyay* and Pallab Dasgupta*
*Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, West Bengal, India 721302
Email: subhankar@cse.iitkgp.ernet.in,antaraain@gmail.com, {subrat,rajdeepm,pallab}@cse.iitkgp.ernet.in

***Abstract*—Behavioral models for analog and mixed signal (AMS) designs are developed at various levels of abstraction, using various types of languages, to cater to a wide variety of requirements, ranging from verification, design space exploration, test generation, and application demonstration. In this paper we present a high-level formalism for capturing the AMS design intent from the specification and present techniques for automatic generation of AMS behavioral models. The proposed formalism is a language independent one, yet the design intent is modeled at a level of abstraction which enables easy translation into common modeling standards. We demonstrate the translation into VerilogA and SPICE, which are fundamentally different standards for behavioral modeling. The proposed approach is demonstrated using a family of Low Dropout Regulators (LDO) as the reference.**

I. Introduction

Behavioral models have immense potential in shaping the design and verification practices for large scale mixed signal integrated circuits. Today, AMS behavioral models (BM) are used in various forms for various purposes. Application engineers use BMs to demonstrate new products to customers. Designers use BMs for design space exploration, stability analysis, etc. Verification engineers use BMs for modeling the environment of a component, and for integration verification. Test engineers use BMs for early development of test programs (even before the silicon becomes available). Curiously most of these models are developed independently (even if they are for the same design) by different teams who do not realize the commonality in the design intent. The models are also developed using different languages to be used in different development frameworks.

The common problem in the industry is that AMS behavioral modeling requires the expertise of analog design and the skill of dynamic systems modeling, but typically analog designers do not have the additional bandwidth to learn and develop BMs. Therefore there is a growing need for automatic generation of AMS BMs. There are broadly two approaches for doing this, namely, *bottom-up model extraction*, where the BM is extracted from the netlist by abstracting out some of the behavioral details, and *top-down model generation*, where the BM is generated from a high level specification of the design intent. The first approach is feasible for moderate sized designs, where as the second approach works well for large integrated circuits built out of previously validated units. The focus of this paper is on the second approach.

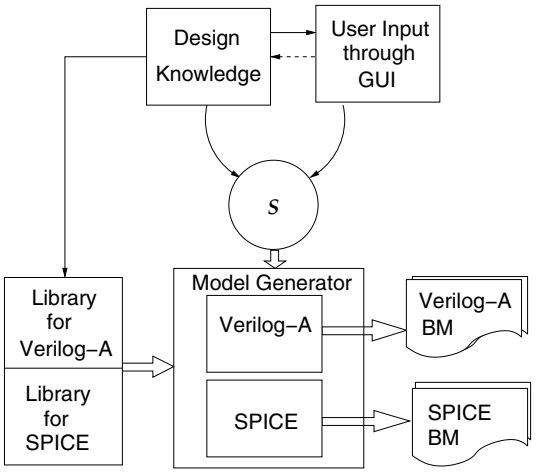


Fig. 1. Proposed BM Generation Flow

The core problem is in identifying a language independent formalism for capturing the design intent and automatically translating this intent into BMs of different languages. Typically BMs which are developed bottom-up preserve structural similarity with the circuit from which it is extracted. On the other hand models which are developed top-down can be quite generic without any specific bias towards a given topology or any specific implementation. The formalism used in this paper is derived from *hybrid automata*[1], which essentially partitions the design intent into the main functional modes in which the design operates, and specifies the real time behavior of the design (possibly as a transfer function) in each mode. Though the models generated from this formalism is not structurally similar to a circuit implementation, these models are significantly light-weight, more readable, and functionally accurate. More importantly, this formalism paves the way for automatic translation into BMs in different languages.

In this paper we present a formalism for capturing the BM intent from a template based specification. We demonstrate the methods for automatic translation of the design intent into auto-generated BMs in VerilogA and SPICE. We also report a prototype tool for reading specifications of Low Dropout regulators (LDOs) and generating BMs in VerilogA and SPICE. We show that the behaviors of these models are identical and also match the behavior of a given netlist designed from the same specs.

978-3-9810801-5-5/DATE09 © 2009 EDAA

The advantage of targetting a tool towards a family of circuits is that a lot of domain knowledge can be implicitly available in the tool. For example, in our prototype tool for LDOs, the end user does not have to specify the transfer functions in each operating mode of the LDO – rather the model is generated using existing domain knowledge and only a few specified parameters. This enables the flow shown in Figure 1. We also have a similar tool for families of buck regulators, but presenting the details of that tool is beyond the scope of this paper.

The organization of the paper is as follows. Section II presents related work. In Section III we present the formalism for capturing the design intent, which is similar to hybrid automata. We also demonstrate the top-down design intent capture by introducing a running example for a Low Dropout Regulator (LDO). Section IV demonstrates the translation mechanism for generating VerilogA and SPICE models, using the running example as a reference. In Section V we present simulation results for both types of models and compare their runtimes with that of a circuit netlist. Section VI presents concluding remarks.

II. Related Work

Behavioral modeling, traditionally known as *macromodeling* is a popular method for design analysis and verification by simulation. There exist several methods for macromodel generation of linear time invariant circuits which include *projection* based methods [2], symbolic methods [3], [4] etc. There have been several efforts in developing BMs for non linear circuits, including root localization methods [5], [6], time-series modeling [7], extension of projections [8] and Volterra kernel transfer functions[9]. The models can be represented by block diagrams and thus tools like Simulink can be used for system level simulation. The models show significant speedup with about 10% accuracy penalty [10].

The use of BMs have been reported in fault simulation [11], interaction of multiple energy domains [12], and high-fidelity noise modeling and non-linearity of analog RF circuits [13].

III. Specification Formalism

Every AMS circuit exhibits both continuous and discrete dynamics. For example, a battery charger circuit usually has five distinct modes of operation viz. *OFF, precharge, constant-current (CC), constant-voltage (CV) and maintenance* modes of operation. In each mode the circuit behaves in a certain way, that defines the behavior of the terminal voltages and currents. Depending on the state variables and the present mode of operation the next mode of operation is determined. This is similar to a hybrid system [1] specification.

A model for a block J developed following the functional partitioning based methodology is formally defined as a tuple $\mathcal{H} = \{Q, Q_0, P, V, G, E, F\}$ where:

- Q is a set of discrete modes.
- Q_0 is the initial mode.
- $P = \{p_1, ...p_n\}$ is the set of pins at the interface of J. $V_P = \{V_{p_1}, ...V_{p_n}\}$ is the set of *interface voltage* variables where $V_{p_i}(t)$ is an *interface voltage* variable that represents voltage of interface pin p_i at time instant t w.r.t ground. Similarly, $I_P = \{I_{p_1}, ...I_{p_n}\}$ is the set of *interface current* variables where I_{p_i} is an *interface current* variable that represents the electrical current flowing out of the module J through the pin p_i at time instant t.
- G is a set of constraints (*guard conditions*) that are defined over V_P, I_P and t (time).
- $E : Q \times G \rightarrow Q$ is a mode transition function.
- $F : Q \rightarrow \{\mathcal{H}_1, ...\mathcal{H}_{|Q|}\}$ is a mapping to the set of behavioral models for each mode. H_i in its simplest form is a continuous time differential equation over V_P, I_P, or it may be another model like $\mathcal{H}$.

TABLE I
Table for LDO Dynamics

MODE	DYNAMICS
SHUTDOWN	Cout*d/dt(vout) = vout/R + Cshut*(d/dt(vout))
SHORT CIRCUIT	I(vout) = shortcktlevel
DROPOUT	V(vout) = V(vin) - V_{drop}
REGULATORY	V(vout) = Vsteady - I(vout)*rout, where, Vsteady = F(Vo,Trim) and rout = (load reg/max.ld curr)
START-UP	V(vout) = V_0(1 - $e^{-t/T}$) where T = RC

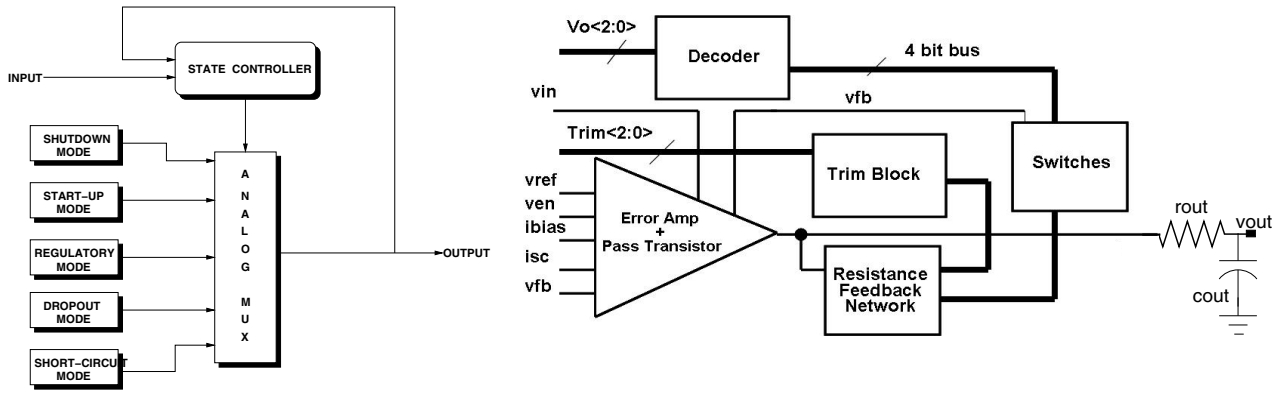
This specification gives us a uniform way of specifying any AMS circuit. Depending on the design the set of modes Q, number of interface pins P, guard conditions G etc. may vary. In fact the same design intent can be modeled with different modes and hence different dynamics at the modes.

A. Model Architecture

A typical topology of an LDO is shown in Figure 2(b). One way to develop a model for an LDO is to create individual models of the component blocks such as the error amplifier, decoder, trim block etc. In our approach, we prefer to partition the functionality of the LDO into the different operating modes in which it operates. The generic design intent of Low Dropout Voltage Regulators (LDO) may be partitioned into the following behavioral modes.

Shutdown Mode: If the enable is not high, or, the bias current, the reference voltage, or the input supply voltage is not within the range, the LDO remains in the shutdown mode of operation, where its output voltage is zero. For normal operation the bias current must be within the specified range:*ibias_val* $+/-$ *ibias_tol*, where *ibias_val* is the nominal value and *ibias_tol* is the tolerance. Similarly, *ref_val* and *ref_tol* are the specified nominal value and tolerance of the reference voltage respectively. The LDO remains in this mode when the input voltage is outside the specified range: [fall_level,up_level]. The LDO is enabled when the *V(ven)* exceeds the specified threshold: *threshold*.

Start-up Mode: When the entire enable, bias current, the reference voltage, and input supply voltage are within the specified region of operation, the LDO enters into the start-



(a) Architecure of the BM **(b) Block Diagram of the LDO**

Fig. 2. Structure of the BM and Block Diagram of the LDO

up mode of operation. The output voltage (*V(vout)*) rises according to a first order response with an user specified time constant.

Regulatory Mode: If all the conditions like enable high, bias current, reference voltage, and the input supply voltage, are within range the LDO enters into the steady state mode of operation. In this mode of operation steady output voltage is obtained. The steady output voltage *V(vout)* is dictated by a function, *F(Vo,Trim)*, where *Vo* and *Trim* are the voltage and trim selection lines respectively (see Figure 2(b)). This function is part of the specification of the LDO. At this level of abstraction the dynamics of the feedback control loop is ignored.

Dropout Mode: We know that for proper functioning of the LDO, the relation *V(vin)* $>$ *V(vout)* $+$ V_{drop} should hold, where V_{drop} is the specified minimum drop out. If the input voltage starts falling, then the output voltage will remain constant as long as the above relation holds. But if the difference between the input and the output voltage is below the dropout value then the output voltage also starts falling below its rated value to maintain their difference above the dropout value. This mode of the LDO is called the dropout mode of operation.

Short Circuit Mode: When the output current crosses a current limit, the LDO enters into the short circuit mode of operation.

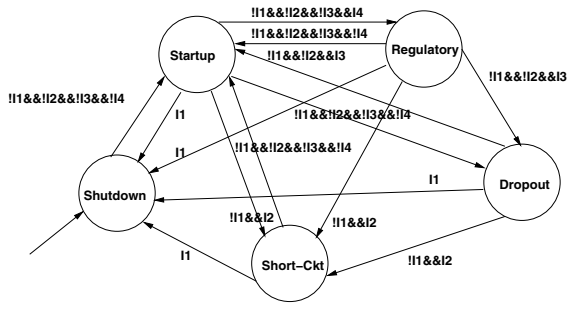


Fig. 3. LDO Mode Transition Diagram

The transitions between these modes of operation can be described in terms of a mode transition diagram as shown in Figure 3. The transition conditions between modes are defined using the constraints I1,I2, I3 and I4 which are stated as follows:

I1: (|V(vref) $-$ ref_val| $\geq$ ref_tol) ||
(V(ven) $>$ threshold) || (V(vin) $<$ fall_level)||
(V(vin) $>$ up_level) || (|I(ibias) $-$ ibias_val| $>$ ibias_tol)
I2: (I(vout) $>$ short_ckt_level) ||
((prev_mode == short_ckt) &&
(ddt(V(vout) $\geq$ 0 for 2us))
I3: (V(vin) $-$ V(vout) $>$ V_{drop})
I4: (|V(vout)$-$VSteady| $\geq$ steady_tol)
where VSteady = F(Vselect,Vtrim)

The dynamics of each mode (i.e the behavior of each mode) is formally defined in Table I. The transition conditions from a mode are all mutually exclusive and so at a time the model can be in only one particular mode.

The topology of our models are directly generated from the mode transition system and the specification of the dynamics in each mode. For a family of circuits (like LDOs) the dynamics of different circuit instances in a given mode can be captured by a set of functions developed from domain knowledge which can be tuned to match various specifations. For example, the number of voltage and trim selection pins may vary from one LDO to another and the model needs to be generated accordingly. Parameters like *ref_val, ref_tol, ibias_val, ibias_tol* may simply be specified as values that are fitted into the model directly. Therefore, our tool has separate sections for specifying the pins and parameters of the model.

The architecture of our models is shown in Figure 2(a). The mode transition conditions of the LDO are checked by the controller module which multiplexes one of the several mode outputs to the model output. The topology of the model is evidently different from that of the LDO topology shown in Figure 2(b).

IV. Model Generation

Our approach for automatic model generation has three components. The first component is capturing domain knowledge for a circuit family into a library of model intents. This library is developed once by studying the functionality of that circuit family in general and maintained implicitly as code skeletons. The second component is reading the specification of a member of the circuit family and the parameters. For example, pin information like the number of voltage and trim selection pins in a LDO is part of its specification, while *short_ckt_level* is a parameter specifying the current limit beyond which the LDO enters the short circuit mode. This part is implemented through a graphical user interface (GUI). The composition of this input with the appropriate model intent completes the internal model definition for our tool. The third component is the translation mechanism into different languages, such as SPICE and VerilogA. Figure 1 shows all three components. This section mainly describes the third component, which is called the *model generator*.

A. Verilog-A Model Generation

The Verilog-A model intent library contains some include file names, parameters, variables, and some behavioral logic which are not dependent on the user interface. There are tags or markers like //BEGINSECTION <NAME> . The model generator fits the information according to these markers. The modeling procedure mainly develops the state controller, the analog MUX and the models for each mode of the LDO (see Figure 3).

1) State Controller: The state controller monitors the guard conditions (events) of each transition from the current mode and sends control signals to the analog mux. If a condition matches then it triggers a mode change, which is recorded by the state controller in its internal state and is reflected by a change in the control signals to the analog mux so as to select the new mode. For example, in the following code, when the condition for shutdown mode is satisfied, the *mode* variable is assigned a value *zero*. The value of the *mode* variable is assigned to the output of the state controller(state). A portion of the state controller in Verilog-A is:

```
if (I(ibias,ibiasn) > 600n | I(ibias,ibiasn) < 400n |
   V(en) > thresh | V(ref) > 1.3 | V(ref) < 1.22 |
   V(vin) < 0) mode = SHUTDOWN;
else
   //CHECK FOR OTHER MODES - ASSIGN VALUES
   V(state) <+ transition(mode,0,1e-9);
```

2) Analog MUX: The state controller selects one of the inputs of the analog multiplexer for propagation to the output of the LDO.

3) Regulatory Mode: The complexity in modeling the regulatory mode is due to the user definable number of select and trim bits. For example, in case of 4 select bits, the following case statement is generated.

```
select = 8*sel[3] + 4*sel[2] + 2*sel[1] +
sel[0];
case(select)
0: V(in,GND) <+ 1.5;
1: V(in,GND) <+ 1.65;
...
15: V(in,GND) <+ 3.4;
endcase
```

The voltage values (such as 1.5, 1.65, etc) corresponding to each value of the select bits is entered as part of the specification. Trim bits are handled similarly.

The codes for the other modes are also generated in a similar way.

B. SPICE Model

SPICE models are generated using electrical components like resistors, capacitors, inductors, transistors etc. The model intent library for generating SPICE models consists of interconnections of these components. Based on the specification of an LDO instance, either the values of some of these components (like voltage source, current source etc), or the parameters of some of them are given as input to the model generator. We hierarchically build the modules (subckt). The smaller modules are instantiated within the top modules. For example to generate the analog multiplexer we instantiated transmission gates, which are created in the library using two MOS transistors. The netlist that we generate is compatible for Spectre Simulation of Cadence Virtuoso Environment. The modeling procedure is described as follows using state controller, analog mux, and the regulatory mode of LDO.

1) State Controller: The state controller comprises of two parts: event generator and state machine. The event generator produces the events from the specification conditions using analog comparators and gates. The output of the event generator is fed to the state machine. State machine takes events and previous state values as inputs and generates the next state (refer Figure 3). The state machine is implemented with D Flip-Flops and logic gates. A parity checker checks the parity of the events and produces edges whenever new events occur. These edges are used to trigger the D Flip-Flops rather than using a synchronous clock, in order to reduce the simulation time.

2) Analog MUX: There are five inputs to the analog MUX. The analog MUX is constructed using some transmission gates (which are created as one functional unit in the library using PMOS and NMOS transisors) and three select lines. According to the select lines, at one time only one transmission gate allows one input to reach the output, which thereby functions as a MUX.

3) Regulatory Mode: The functionality of the select and trim bits are implemented using a network of 2-Input analog multiplexers. For example, there will be 3 analog multiplexers instantiated by the generator, in case of a 2 bit select pin. The generator produces voltage sources and a hierarchical network of analog multiplexers preserving the interconnection in between them, connecting the select pins appropriately to the various multiplexers and connecting the input and output pins.

TABLE II
CPU TIMES: NETLIST AND BM

Simulation Time(ms)	CPU Time		
	Transistor Level	Verilog-A Model	Spice Model
1	19.1s	5.57s	18.2s
5	20.3s	11.8s	18.4s
10	43s	18.4s	18.6s

TABLE III
RUNTIMES: VERILOG-A VERSUS SPICE

Instance	CPU Time	
	Verilog-A	Spice
LDO1	10.2s	19s
LDO2	9.96s	15.6s
LDO3	9.85s	16.6s
LDO4	9.71s	17.2s

The generated Spice Netlist for a 2bit select pin is as follows:

```
//Voltage Sources
V1 (INP1 GND) vsource dc=1.5 type=dc
V2 (INP2 GND) vsource dc=1.65 type=dc
V3 (INP3 GND) vsource dc=2.5 type=dc
V4 (INP4 GND) vsource dc=3.4 type=dc
//Multilplexers
I0 (INP1 INP2 GND Inter1 sel0 VDD 0 vdd!)
analogmux
I1 (INP3 INP4 GND Inter2 sel0 VDD 0 vdd!)
analogmux
I2 (Inter1 Inter2 GND OUT sel1 VDD 0 vdd!)
analogmux
```

In this example, INP1-INP4 are input voltage sources, sel0-sel1 are select pins and Inter1, Inter2 are intermediate pins.

V. RESULTS

Table II presents the CPU time spent in simulating one LDO for various simulation times. Our experiments were performed on a 2.8GHz, 4GB RAM, Xeon Processor. For the lower simulation times (first two rows) the runtime of the SPICE model is marginally better than that for the transistor level netlist. However for a larger simulation time (third row), the SPICE model runs significantly faster. This is because in the first two cases most of the time is spent in the startup phase, which has more events than the other operating modes. The Verilog-A model returns the best runtimes in all these cases.

Figure 4, Figure 5, and Figure 6 show the simulation waveforms for the transistor level netlist, the Verilog-A model and the Spice model respectively for a simulation time of 2ms. These figures are shown to demonstrate that the models accurately mimic the behavior of the netlist.

We varied various parameters to create SPICE and Verilog-A models for different instances of LDOs. Specifically we targetted the parmeters: Vref, ibias, ven, Vo (no of pins for voltage selection) and compared the CPU times for 5ms of simulation time. Table III shows this comparison. The results

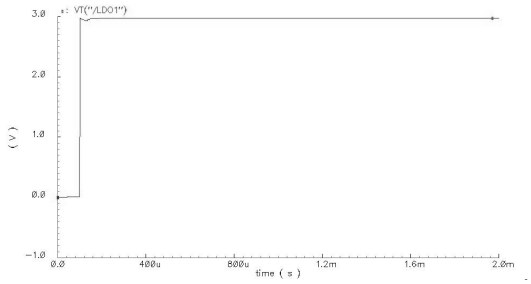


Fig. 4. Transistor netlist Simulation Results

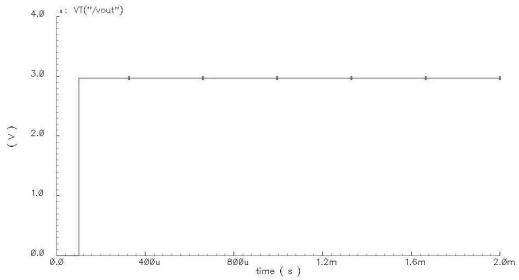


Fig. 5. Verilog-A Simulation Results

show that the simulation times do not change significantly with the instances, and therefore the difference in performance between the Verilog-A and SPICE models remain the same. It will be interesting to see whether this is a generic feature which applies to families other than LDOs.

VI. CONCLUSION

This paper shows that automatic generation of behavioral models in various languages is possible for targetted families of circuits, provided that we spend prior effort in capturing the design intent of this family in terms of domain knowledge and tune the generated models using instance specific configuration parameters. Our approach deviates from the standard practice of generating models that are structurally similar to the netlist, and provides an alternative modeling framework which appears to be more amenable to automatic translation into modeling languages.

REFERENCES

[1] R. Alur and et al., "The algorithmic analysis of hybrid systems," *Theoretical Comput. Sc.*, vol. 138, no. 1, pp. 3–34, 1995.

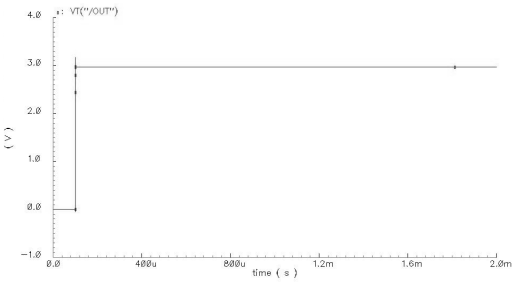


Fig. 6. SPICE Simulation Results

[2] E. Grimme, "Krylov projection methods for model order reduction, Ph.D. thesis UIUC EECS Department,," 1997.
[3] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: a symbolic simulator for analog integrated circuits," *Solid-State Circuits, IEEE Journal of*, vol. 24, no. 6, pp. 1587–1597, 1989.
[4] F. Fernández and et al., *Symbolic Analysis Techniques: Applications to Analog Design Automation*. IEEE Press, 1998.
[5] H. Mantooth and P. Allen, "A higher level modeling procedure for analog integrated circuits," *Analog Integrated Circuits and Signal Processing*, vol. 3, no. 3, pp. 181–195, 1993.
[6] X. Huang and et al., "Modeling nonlinear dynamics in analog circuits via root localization," *IEEE TCAD*, vol. 22, no. 7, pp. 895–907, 2003.
[7] D. Root, J. Wood, N. Tufillaro, D. Schreurs, and A. Pekker, "Systematic behavioral modeling of nonlinear microwave/RF circuits in the time domain using techniques from nonlinear dynamical systems," *Behavioral Modeling and Simulation, BMAS*, pp. 71–74, 2002.
[8] J. Phillips, "Projection-based approaches for model reduction of weakly nonlinear, time-varying systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 171–187, 2003.
[9] X. Li, P. Li, Y. Xu, and L. Pileggi, "Analog and RF circuit macromodels for system-level analysis," *Proceedings of the 40th DAC*, pp. 478–483, 2003.
[10] H. Mantooth, L. Ren, X. Huang, Y. Feng, and W. Zheng, "A survey of bottom-up behavioral modeling methods for analog circuits," *ISCAS*, vol. 3, 2003.
[11] A. J. Perkins, M. Zwolinski, C. D. Chalk, and B. R. Wilkins, "Fault modeling and simulation using vhdl-ams," *Analog Integr. Circuits Signal Process.*, vol. 16, no. 2, pp. 141–155, 1998.
[12] P. R. Wilson, J. R. Neil, A. D. Brown, and A. Rushton, "Multiple domain behavioural modeling using vhdl-ams," in *IEEE International Symposium on Circuits and Systems*, 2004. [Online]. Available: http://eprints.ecs.soton.ac.uk/9370/
[13] W. Yang, H. Carter, and J. Yan, "A high-level vhdl-ams model design methodology for analog rf lna and mixer," in *International Behavioral Modeling and Simulation Conference*, 2004, pp. 125–129.