Equation-Based Behavioral Model Generation for Nonlinear Analog Circuits

Carsten Borchers, Lars Hedrich and Erich Barke

Institute of Microelectronic Systems, University of Hanover, 30167 Hanover, Germany
Email: borchers@ims.uni-hannover.de

Abstract

A fully automatic method for generating behavioral models for nonlinear analog circuits is presented. This method is based on simplifications of the system of nonlinear differential equations which is derived from a transistor level netlist. Generated models include nonlinear dynamic behavior. They are composed of symbolic equations comprising circuit parameters. Accuracy and simulation speed-up are shown by several examples.

1 Introduction

The need to simulate large and complex mixed-signal systems has prompted the development of high-level circuit representations for analog components. These behavioral models capture certain functional properties without using specific internal representations. Two different modeling methods have been established to generate high level descriptions for analog circuits [1]. Macromodels are built up from SPICE primitives like controlled sources to describe circuit behavior. The well-known Boyle operational amplifier macromodel [2] is an excellent example for this type of modeling. With the availability of analog hardware description languages (AHDLs) a second modeling method has been introduced. Systems of differential equations are used to describe desired behavior. These models are normally referred to as behavioral models. They allow efficient behavioral descriptions on any level of accuracy. A deep insight into nonlinear dynamic behavior is necessary for the development of a behavioral model. Thus, behavioral modeling for nonlinear analog circuits is still a manual task. Up to now, automatic methods have been proposed only for linear circuits. They include state space approaches [3, 4], symbolic analysis in conjunction with several simplifications techniques [5] and model order reduction by moment matching methods like AWE [6]. Optimization is used to improve manually generated linear and nonlinear behavioral models [7, 8].

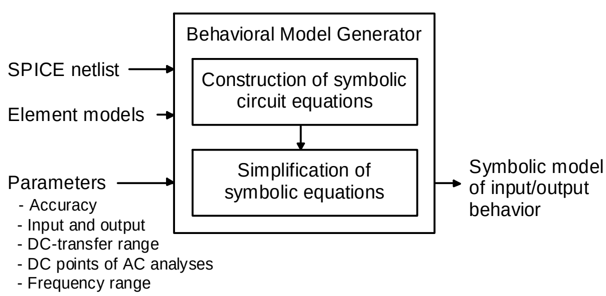

Figure 1: Concept of behavioral model generation

In this contribution a fully automatic behavioral model generation algorithm for nonlinear analog circuits is presented. The concept of this method is shown in Figure 1. Models are generated for the input/output behavior of an analog circuit with respect to the DC-transfer (DT) characteristic and several frequency responses at different operating points (multiple AC). A SPICE netlist of the circuit is taken as input. The corresponding system of nonlinear differential equations is built up symbolically using appropriate model equations for all circuit elements (e.g. Gummel-Poon model for bipolar transistors). This system is manipulated and simplified preserving desired accuracy within the operating conditions. Derived models are composed of symbolic equations comprising circuit parameters. The simplification steps are controlled by an efficient error estimation algorithm. The proposed algorithm drastically reduces the number of terms and variables. Thereby, simulation time decreases significantly. Generated models include nonlinear dynamic behavior. Thus, they are suited for transient simulations.

The remainder of this paper is organized as follows. In Chapter 2 the behavioral model generation algorithm is presented. Following, in Chapter 3 the feasibility of this approach is demonstrated by several benchmark circuits. Finally, some concluding remarks are given in Chapter 4.

2 Behavioral Model Generation

2.1 Algorithm

In Figure 2 the behavioral model generation algorithm is presented. For each simplification step the fundamental operation is printed in italics.

The modified nodal approach (MNA) is used to construct the system of symbolic circuit equations [9]. This results in a system of n time-invariant differential equations

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, e) = \mathbf{0} \ . \tag{1}$$

Construct symbolic circuit equations using MNA
Transformations by substitutions
for all variables using variable ranking **do**
  Delete all time derivatives
for all variables using variable ranking **do**
  Calculate mean value based on DT characteristic
  Set variable to mean value
Expand all equations into sum of product form
for all terms using term order **do**
  if estimate error = true **then**
    Delete term
for all nonlinear terms using term order **do**
  if estimate error = true **then**
    Linearize term about mean value of variables
for all new terms using term order **do**
  if estimate error = true **then**
    Delete term
Transformations by substitutions

Figure 2: Behavioral model generation algorithm

33rd Design Automation Conference ®

Here, $\mathbf{x} = \mathbf{x}(t)$ is the vector of n independent circuit variables at time t. The vector $\dot{\mathbf{x}}$ represents the time derivatives of $\mathbf{x}$. Since in this contribution only single-input/single-output systems are considered $e = e(t)$ is the input.

The first simplification step involves transformations on (1). If an equation can be solved explicitly for a variable the variable can be substituted and the equation itself can be removed from the system. In this way, e.g. variables which correspond to currents through voltage sources are eliminated.

The second simplification step involves the elimination of all time derivatives of each variable if the AC error (see Chapter 2.2) is not exceeded. Normally, only few variables determine the dynamic behavior of the circuit. In this case, the system of equations can be simplified drastically. A special variable ranking is used to obtain an appropriate order in which variables are processed.

Next, variables are set to their mean values calculated from the DT characteristic. This step aims at those variables corresponding to nodes of the circuit which are used for bias purposes. Since these variables are not known a priori all variables are set to their mean values whenever the maximum allowed error is not exceeded. The variable ranking mentioned above is used again.

Following, all equations are expanded into sum of product form. All further simplification steps are performed on individual terms of these equations. An error estimation algorithm (see Chapter 2.2) is used to decide whether a particular operation is allowed or not. Terms are processed in the order of increasing standard deviations calculated from the DT characteristic to achieve maximum reduction. First, terms are completely removed from their equations. Next, all nonlinear terms are linearized about the mean values of their variables. Since linearization may increase the overall number of terms a second term deletion step follows for all new terms. Finally, transformations by substitutions (see above) are performed on the system of equations again.

2.2 Error Estimation

All operations to simplify a system of equations will result in a certain change of the input/output behavior. An error prediction algorithm is necessary to decide whether a simplification will exceed the maximum error allowed. Generally, a time consuming exact calculation using DT and multiple AC simulations to obtain the DT characteristic and frequency responses should be avoided because of the high number of simplification operations that have to be performed. Thus, a special DT error estimation algorithm has been developed. This algorithm is based on the Newton-Raphson method which is normally used in SPICE-like simulators to calculate the DT characteristic [9].

The Newton-Raphson method uses the following iteration sequence to calculate the solution $\mathbf{x}_s$ of a system of nonlinear algebraic equations $\mathbf{g}(\mathbf{x}) = \mathbf{0}$.

$$\mathbf{g}(\mathbf{x}^k) + M\big|_{\mathbf{x}^k} \cdot (\mathbf{x}^{k+1} - \mathbf{x}^k) = \mathbf{0}\,. \qquad (2)$$

The superscript k denotes the k-th iteration. M is the Jacobian matrix of $\mathbf{g}(\mathbf{x})$. The solution of (2) is normally carried out by LU decomposition. The convergence rate of $\mathbf{x}^k$ towards the solution $\mathbf{x}_s$ is at least quadratic if the initial starting point $\mathbf{x}^0$ of (2) is sufficiently close to $\mathbf{x}_s$ [10]. Unfortunately, the prediction whether $\mathbf{x}^0$ is a sufficient starting point implies the knowledge of the solution $\mathbf{x}_s$. Thus, it is impossible to predict the convergence characteristic of (2) for any given starting point. Instead, the convergence of (2) is checked after each iteration using different convergence criteria like

$$\left\|\mathbf{g}(\mathbf{x}^{k+1})\right\| < \left\|\mathbf{g}(\mathbf{x}^k)\right\| \quad \text{or} \quad \left\|\mathbf{x}^{k+1} - \mathbf{x}^k\right\| < \left\|\mathbf{x}^k - \mathbf{x}^{k-1}\right\|, \qquad (3)$$

where $\|\cdot\|$ denotes an appropriate vector norm.

The DT error estimation algorithm is based on the fact that only one variable (the output x_m) of $\mathbf{x}$ defines the input/output behavior. The solution of all other variables is arbitrary as long as convergence is achieved. In Figure 3 the algorithm is presented for the system of nonlinear algebraic equations $\mathbf{g}(\mathbf{x}, e) = \mathbf{0}$ which has been obtained from (1) by setting all time derivatives of $\mathbf{x}$ to zero.

for n points of DT characteristic **do**
  Calculate 2 Newton iterations ($\mathbf{x}^0$ from DT characteristic)
  Calculate $\left|dx_m^i\right| = \left|x_m^i - x_m^{i-1}\right|$, $\left\|\mathbf{dx}^i\right\| = \left\|\mathbf{x}^i - \mathbf{x}^{i-1}\right\|$ and $\left\|\mathbf{g}^i\right\|$
  if $\left|dx_m^1\right|^2 \geq \left|dx_m^2\right|$ **and** $\left\|\mathbf{dx}^1\right\| \geq \left\|\mathbf{dx}^2\right\|$ **and** $\left\|\mathbf{g}^1\right\| \geq \left\|\mathbf{g}^2\right\|$ **then**
    Calculate new estimated DT point: $\mathbf{x} = \mathbf{x} + \mathbf{dx}^1 + \mathbf{dx}^2$
  else if $\left|dx_m^1\right| \geq \left|dx_m^2\right|$ **then**
    Calculate DT characteristic by exact DT analysis
    break
  else
    return(false)
if DT error ε **<** maximum allowed error **then**
  return(true)
else
  return(false)

Figure 3: DT error estimation algorithm

The algorithm is based on a convergence check using (3) which is performed after two Newton iterations have been calculated. If no output convergence is detected the algorithm terminates immediately. This may happen if the Jacobian is singular or if the system solution changes significantly. If quadratic output convergence and convergence of all other variables are achieved it can be expected that the simplification of the system of equations has little influence on the solution. If output convergence is detected, but the convergence rate is slow, an examination by an exact DT analysis is carried out because in general the prediction of the solution is impossible for this case. During error estimation a new estimated DT solution is calculated which is used as starting point for the next error estimation. Due to the fact that the estimated DT solution may fail even if quadratic convergence is achieved a periodic exact DT analysis has been implemented to update the estimated DT solution. This may cause a backtrack to undo simplification steps.

An AC error estimation is carried out following the DT error estimation. If the static part of the system of equations is not changed by a particular simplification operation only an AC error estimation is needed. Since the operating points of the multiple AC analyses are known from the DT error estimation the calculation of the frequency responses requires the solution of systems of linear equations only.

The DT and AC errors are calculated with respect to the reference signals using a modified relative error ε.

$$\varepsilon = \sum \frac{\left|x_{m,ref} - x_{m,simp}\right|}{\left|x_{m,ref}\right| + R},$$

x_m is the output variable and $R = |\max(x_{m,ref})| - |\min(x_{m,ref})|$ is the range of the reference signal.

2.3 Implementation

The behavioral model generation algorithm has been implemented using a symbolic computer algebra package. All equation manipulation steps mentioned above are performed symbolically resulting in a behavioral model which consists of symbolic equations. Numerical calculations based on the circuit parameter values are used during all decision steps like error estimation. Since numerical calculations are usually slow in symbolic algebra packages the analog simulator Saber® [11] was chosen to perform all DT and AC analyses. In principle, every analog simulator which incorporates an AHDL to describe arbitrary systems of differential equations can be used for this task.

To overcome convergence problems which may arise during the analysis of the system of equations special convergence aids are automatically inserted into the AHDL description of the behavioral model. For a system of equations containing strong nonlinearities like exponential functions an efficient control of the iteration step size of the Newton algorithm is necessary to achieve convergence [9]. Therefore, the system of equations is reformulated for the analysis task. Differences of variables are introduced as additional dependent variables. The choice of suitable variable differences is derived from the structure of the original circuit. For instance, in circuits containing bipolar transistors or diodes every voltage across a pn-junction is used as an additional dependent variable in the reformulated system of equations. These variables are subject to a Newton step limitation which is provided by Saber®.

3 Examples

A differential pair consisting of four bipolar transistors has been chosen to demonstrate the behavioral model generation in detail. The bipolar transistors are described using the Gummel-Poon transistor model [12]. The behavioral model is generated for the DT range of the input voltage vin from -0.25V to 0.25V and one AC analysis at vin = 0V. The maximum allowed DT and AC error is each set to 2.5%.

Table 1 shows the results of the simplification steps (see Figure 2). The number of terms is obtained from the expanded equations, static terms include no time derivatives.

		Terms	Linear		Nonlinear	
Step	Var's	Total	Static	Dyn.	Static	Dyn.
Original	16	288	98	30	32	128
Transformations	11	241	76	27	26	112
Derivatives	11	142	76	8	26	32
Set to mean	9	111	52	8	19	32
Delete terms	8	30	17	0	6	7
Linearize terms	8	33	21	4	5	3
Delete terms	8	29	21	1	5	2
Transformations	4	18	10	1	5	2

Table 1: Model generation results of differential pair

In this example the number of variables has been reduced by a factor of 4, the number of terms by a factor of 16. Overall, 124 error estimations have been carried out. 109 times (88%) the error estimation algorithm successfully decided whether a simplification operation is allowed (83 times) or not (26 times). Exact DT analyses were necessary for the remaining 12%.

The simulation results of DT and AC analysis are shown in the next two figures where the output of the original circuit (reference) is compared to that of the behavioral model (model). A DT error of ε = 0.45% and an AC error of ε = 2.3% have been achieved.

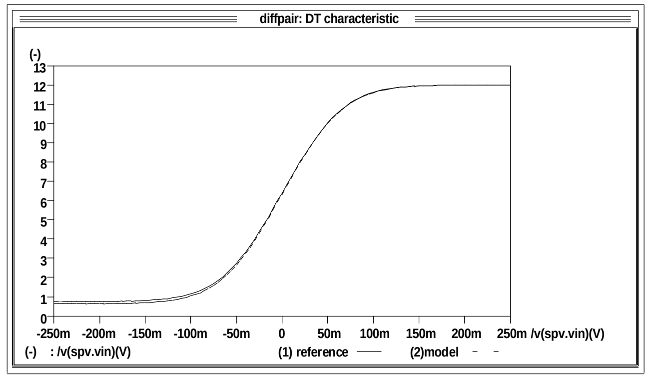

Figure 4: DT simulation results of differential pair

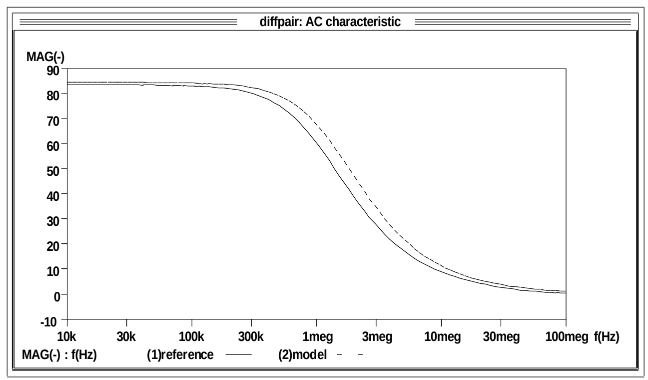

Figure 5: AC simulation results of differential pair

Figure 6 shows the transient analysis of the behavioral model compared to the original circuit. A 500kHz sine with an amplitude of 0.15V is used as input signal. Since the behavioral modeling principle is based on simplifications of circuit equations the resulting behavioral model includes nonlinear dynamic behavior although the model generation itself is solely based on DT and AC operating conditions.

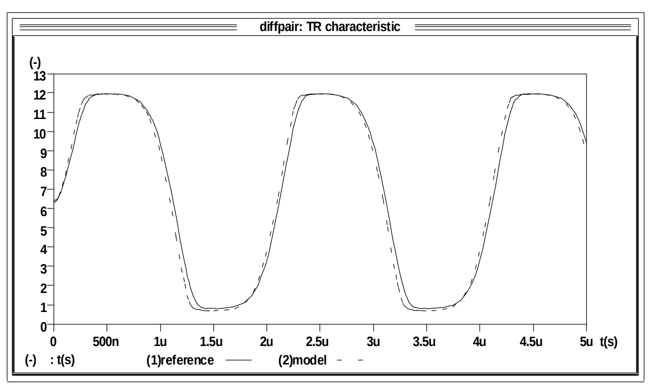

Figure 6: Transient simulation results of differential pair

Table 2 shows behavioral model generation results of other benchmark circuits. Circuits and nonlinear element models are taken from [12]. All behavioral models are generated using appropriate DT and AC operating conditions. The maximum allowed DT and AC error is each set to 5%.

	Circuit		Behavioral Model	
	Variables	Terms	Variables	Terms
rtlinv	10	94	4	16
ccsor	12	95	3	9
diffpair	16	182	4	15
eclgate	20	433	5	20
rca3040	21	740	7	38
ua733	26	205	5	18
ua709	28	1044	16	55
ua741	29	1409	18	61
ua727	39	1564	15	63

Table 2: Model generation results of benchmark circuits

Reduction factors of terms and variables are shown in Figure 7. Simulation speed-up factors for the DT analysis are presented in Figure 8.

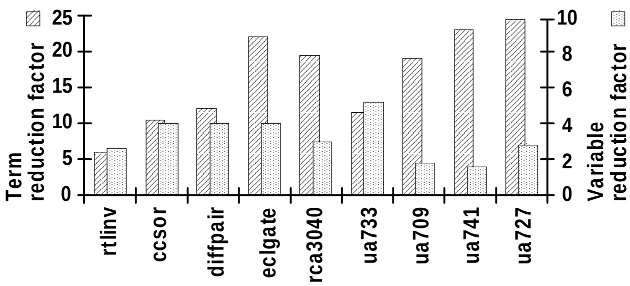

Figure 7: Reduction of variables and terms

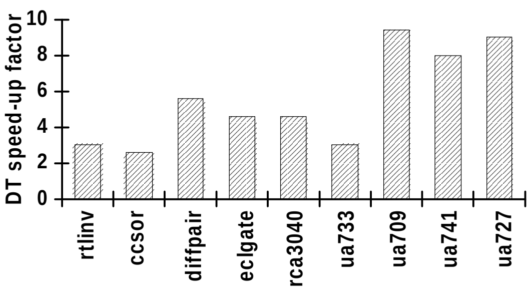

Figure 8: DT simulation speed-up

Obviously, only a small part of the circuit determines its behavior. Higher term reduction factors are achieved for larger circuits since they normally contain more parts which have no influence on their input/output behavior. The variable reduction factor decreases with increasing size of the behavioral model because less transformations by substitutions can be performed on the system of equations. The results show that simulation speed-up is mainly determined by the term reduction factor. Summarizing, large reduction of terms and variables results in a high simulation speed-up. The construction of behavioral models that only include terms which determine the input/output behavior seems to be very efficient since this can be fully automated.

Finally, in Figure 9 the overall performance of the error estimation algorithm is shown. No DT analysis is necessary for a particular simplification operation whenever the error estimation can predict its influence on the circuit behavior. Obviously, for a great number (average: 92%) of terms of the equations it can be accurately predicted that they either have no influence or have strong influence on the circuit behavior.

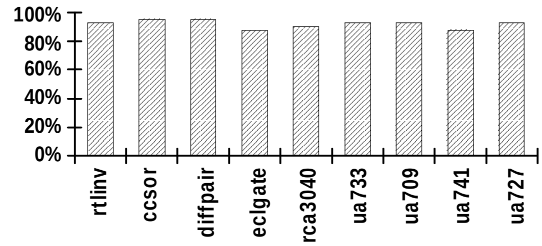

Figure 9: Simplification operations without DT analysis

4 Conclusion

A fully automatic algorithm for symbolic behavioral model generation has been presented. The algorithm starts with a system of circuit equations which is built up from a transistor level netlist using accurate physical element models. This system is successively simplified using an efficient error estimation algorithm. Derived models include nonlinear dynamic circuit behavior. The proposed method has been examined using several benchmark circuits. High reduction factors of the number of terms and variables have been achieved. This results in a large simulation speed-up of the behavioral models.

References

[1] I. Getreu and D. Teegarden, "An Introduction to Behavioural Modelling", *Microelectronics Journal*, vol. 24, no. 7, pp. 708-716, 1993

[2] G. R. Boyle, B. M. Cohn, D. O. Pederson and J. E. Solomon, "Macromodeling of Integrated Circuit Operational Amplifiers", *IEEE Journal of Solid-State Circuits*, vol. 9, no. 6, pp. 353-365, 1974

[3] B. A. A. Antao and F. M. El-Turky, "Automatic Analog Model Generation for Behavioral Simulation", *Proc. Custom Integrated Circuits Conference*, pp. 12.2.1-4, 1992

[4] C. Visweswariah, R. Chadha and C.-F. Chen, "Model Development and Verification for High Level Analog Blocks", *Proc. 25th Design Automation Conference*, pp. 376-382, 1988

[5] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits*, Kluwer Academic Publishers, 1991

[6] V. Raghavan, R. A. Rohrer, L. T. Pillage, J. Y. Lee, J. E. Bracken and M. M. Alaybeyi, "AWE-inspired", *Proc. Custom Integrated Circuits Conference*, pp. 18.1.1-8, 1993

[7] V. M. Ma, J. Singh and R. Saleh, "Modeling, Simulation and Optimization of Analog Macromodels", *Proc. Custom Integrated Circuits Conference*, pp. 12.1.1-4, 1992

[8] G. Casinovi and A. Sangiovanni-Vincentelli, "A Macromodeling Algorithm for Analog Circuits", *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 2, pp. 150-160, 1991

[9] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, 1983

[10] H. R. Schwarz, *Numerical Mathematics* (in german), B. G. Teubner, 1988

[11] Analogy® Inc., *Saber® Reference Manual*, Release 3.2, 1993

[12] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits", *Memorandum No. ERL-M520*, University of California, Berkeley, 1975