Hierarchical Automatic Behavioral Model Generation of Nonlinear Analog Circuits based on Nonlinear Symbolic Techniques

Lutz Näthke, Volodymyr Burkhay, Lars Hedrich, Erich Barke
Institute of Microelectronic Systems, University of Hanover, Germany

E-mail: {naethke,burkhay,hedrich,barke}@ims.uni-hannover.de

Abstract

We present an extended method of automatic behavioral model generation for nonlinear analog circuits. The focus is on a decrease of simulation time. A procedural model formulation approach is introduced, together with a new simplification method based on the recognition of physical transistor properties of the element models. The simplification process is performed with respect to simulation time, and a hierarchical modeling approach is proposed. The result of these extensions are models with an obvious speed–up in simulation time compared to the simulation of the original netlists.

1. Introduction

One hot topic of recent discussions about the needs for analog and mixed–signal design was behavioral modeling and the demand for automated methods for the generation of behavioral models (e.g. at DATE01 in [11]). Behavioral models enable hierarchical multilevel descriptions of circuits and systems. Therefore, the simulation of large circuits and faster simulations of analog circuits are possible. Behavioral model generation plays also an important role in the development of element libraries and in the analog intellectual property field.

In [5] a method for automatic behavioral model generation for nonlinear analog circuits based on nonlinear symbolic simplification techniques was introduced. Several extensions were implemented to improve the reduction ratio of the generated models like the integration of large signal behavior into the models, term ordering [16], and term ranking [21]. While the need for simulation speed–up of the behavioral models is still evident, the only way trying to achieve this was a further improvement in model reduction.

In this paper, we present enhancements to the original method which were especially developed to assure simulation speed–up of the generated models. Therefore, we first give a short review of previous techniques of model generation in Section 2. Section 3 summarizes the main aspects of our approach of automatic model generation. Section 4 establishes the procedural model formulation approach. Section 5 describes the new simplification method and in Section 6 a new control method especially designed for simulation speed–up is introduced. In Section 7, the hierarchical model approach is discussed. Section 8 shows results for some industrial benchmark circuits before Section 9 gives the conclusions.

2. Review of Automatic Behavioral Model Generation

Black box modeling is a widely used technique in model generation. Approximation, fitting and system identification are the most common methods used. Overviews can be found in [13, 17]. Transformations of state space representations are used for linear circuits [2], neural or fuzzy networks and combinations of both for nonlinear circuits [6, 15]. The problem regarding all approximation methods is that pre–modeling knowledge is necessary or assumptions about the circuit and/or the model and its structure have to be made. The generated models are only valid for the training/experiment data, so that a new model is necessary for each new operating range.

Equivalent circuits — as another modeling method — abstract the behavior of a complex circuit by using a netlist of basic elements like resistors, capacitors and controlled sources. Two different approaches exist: In the first, the parameters of the elements of a fixed structure (so called *model template*) are fitted to the circuit's behavior [12]. In the second, the parameters and the structure have to be computed [7, 8]. The model template approach is widely used in the high frequency domain, however, it is limited to the applied circuit class because of the fixed model templates. In the second method, interaction with the user is required during the modeling process. Thus, both methods are not qualified for an automatic model generation algorithm addressing a wide class of circuits.

1530-1591/04 $20.00 (c) 2004 IEEE

Different reduction methods have also been introduced for model generation. Symbolic analysis techniques, which were introduced for linear circuits, have been extended to (weakly) nonlinear circuits. Besides the one proposed here, different methods can be found [4, 14, 25]. Padé–Approximation is another reduction technique which is mainly used for the reduction of large linear networks. An overview is given in [10]. Furthermore, piecewise linear approaches or Volterra–kernels are used to model (weakly) nonlinear circuits [20, 23].

Composite approaches use combinations of the approaches above. Best known are Wiener and Hammerstein models. They split the problem into a linear dynamic and a nonlinear static model. Most often, the first is built by reduction methods, and the latter by using an approximation technique. Several publications about recent work are available, such as [9, 3]. The models show the same limitations as black box models because a part of them is generated using the same techniques. Besides this, sometimes convergence problems have been reported due to the combinational characteristic of the models.

The only approach known to the authors which can handle strongly nonlinear behavior and which is also based on the physical description of circuit elements is described in [5]. It allows the parameterization of the models and shows no convergence problems. All circuit classes can be addressed, and the method is fully automated.

3. Automatic Generation of Behavior Models for Nonlinear Analog Circuits

The basic concept for the automatic generation of behavioral analog models is shown in Figure 1. It has been fully described in [5, 22], so that only a short outline is given here. To generate a model, the user must provide the following input data to the system:

- A netlist that describes the circuit to be modeled.
- The model operating conditions specifying the analyses. These can include one or more of DC–transfer, AC, transient, and specification based analyses. Within the model operating conditions the validity of the model is proven. They also describe which testbench to use and which control nodes are used for error calculation.
- A maximum allowed error ε_{max} for each control node of the model operating conditions to guide the simplification process. It defines the maximum deviation of the input–output behavior of the simplified model compared to the reference simulation of the model. The definition of the maximum allowed error depends on the analysis. For DT and AC, a modified relative error is used [5], while for transient analysis a modified Haudorff–Distance calculation was introduced [16].

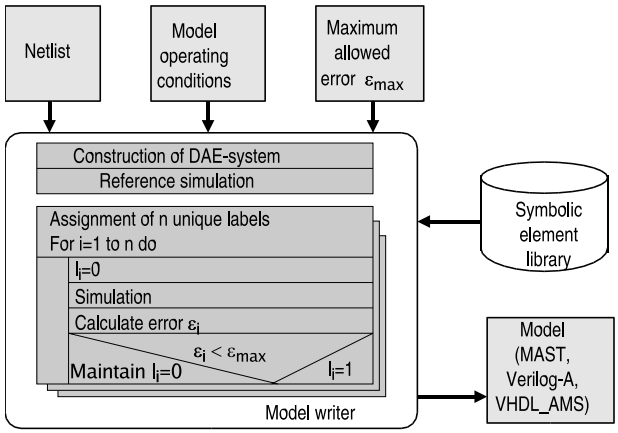

Figure 1. Method of analog model generation

First, a system of symbolic differential algebraic equations (DAE–system) is built from the netlist. The equations are derived by the modified nodal approach using a library of symbolic elements. The DAE–system is simulated within the given model operating conditions. The results are saved as the reference simulation for further calculations. Then, the DAE–system is simplified. Different simplification methods are available, addressing various properties of the DAE–system. Within each method, the first step is to assign a number n of unique labels l_i to special terms within the DAE–system. The manner of this assignment of labels depends on the addressed properties of the simplification method. One simplification method is to prepare all terms linked by an addition in the DAE–system. In (1) the early voltage V_A of a MOS transistor is shown as an example, with the inserted unique labels $l_3 \ldots l_7$.

$$V_A = l_3 V_{As} + \left(l_4 1 + l_5 \frac{P_v V_{Gt}}{E_s L_f} \right) \frac{1}{l_6 \frac{1}{V_{AC}} + l_7 \frac{1}{V_{AD}}} \quad (1)$$

All unique labels l_i are initialized with $l_i = 1$. Each simplification is performed by setting $l_i = 0$. After a simplification step is executed, a simulation is performed using the given model operating conditions. The deviation ε_i between the simulation of the simplified model and the reference simulation is calculated. If $\varepsilon_i < \varepsilon_{max}$ is true, the simplification is maintained by keeping $l_i = 0$. Otherwise, it is withdrawn assigning $l_i = 1$. After all unique labels have been checked, they are eliminated symbolically together with the connected algebraic expressions, and a new simplification method can be performed. After all simplification methods have been accomplished, the behavioral model of the circuit is written

The advantage of this approach is that only the important equations, variables and parameters for the given model

operating conditions are included in the model. Furthermore, by allowing any model operating condition and any combination of them, the user receives models which fit his needs. Parameter variations can be included by using different model operating conditions with changing parameters. Because this method is based on the analytic equations of the circuits elements, the models will also show good results when simulated in other analyses than that tested within the model operating conditions. This gentle behavior cannot be expected from empirical methods like curve fitting.

However, there also exists a drawback of this method: middle sized and large circuits cannot be modeled due to the very large size of the DAE–system at the beginning. Either the simulation time of the system is so large that the model generation time is extended to several days, or the system has convergence problems so that simulation is not possible. Four extensions have been made to address these problems: A formulation of models using procedural equations, a new simplification method which tries to perform large simplifications directly at the beginning, simulation–time–controlled simplification, and an approach to use hierarchy within the model generation.

4. Procedural Formulation of Equations

Looking at the Newton iteration performed in each simulator, the following equation has to be solved:

$$\mathbf{J}(x_i,u,p)(x_{i+1}-x_i) = f(x_i,u,p) \qquad (2)$$

x_{i+1} is the vector of N_x unknown variables at the actual iteration, x_i the vector of unknowns of the previous iteration, $\mathbf{J}(\cdot)$ the Jacobian matrix, $f(\cdot)$ the nonlinear DAE–system, u the input and p the parameter vector of the circuit. The most time consuming steps of the Newton iteration are the update of $\mathbf{J}(\cdot)$ and solving (2). The time for updating $\mathbf{J}(\cdot)$ depends on the number of entries and the method of the generation of $\mathbf{J}(\cdot)$. In case of a difference method (e.g. [1]) it could increase quadratically with its size ($O(N_x^2)$). The complexity for solving (2) is of $O(N_x^{1.5\ldots1.7})$. Hence, reducing N_x results in an decrease of simulation time.

The procedural formulation technique is based on the difference between *procedural equations* $f_p(\cdot)$ and *simultaneous equations* $f_s(\cdot)$. Using the former, a variable is explicitly calculated from parameters and already known variables (from previous $f_p(\cdot)$ or $f_s(\cdot)$ solved in the previous step). The simultaneous equations $f_s(\cdot)$ are formulated implicitly and determine the size of $\mathbf{J}(\cdot)$. $f_p(\cdot)$ cause additional contributions to $\mathbf{J}(\cdot)$, but do not increase its size. Hence, reducing the number of $f_s(\cdot)$ is equal to decreasing the size of $\mathbf{J}(\cdot)$. In most cases, also the convergence rate of the reduced DAE–system is increased dramatically, abandoning intermediate variables. In our experiments, formulating as much equations as possible in $f_p(\cdot)$, the simulation time decreases between ten and over 200. A further advantage is that $f_p(\cdot)$ can be formulated in compiled code models, if supported by the used simulator and/or AHDL. Using such compiled code models gain in another speed–up of factor two to five. We developed an algorithm to distinguish between these two types automatically during the set–up of the DAE–system. This formulation method is one major key within the generation of behavioral models of analog circuits with simulation time speed–up emphasis.

5. Pre–Labeled Transistor Models

Using standard simplification methods of the nonlinear symbolic analysis approach, the DAE–system is built before a decision about the terms to be simplified is made. In this way, all knowledge about physical effects and transfer functions of the used symbolic element models are lost. To take advantage of this knowledge, the symbolic element models must be prepared with unique labels in an intelligent way *before* the DAE–system is built. So, the pre–labeled approach works with hard coded unique labels in the equations of the elements of the symbolic element library. This is especially useful for symbolic element models with many equations as the BSIM3 MOS model. We use the BSIM3 model to demonstrate the pre–labeled transistor model approach. Unique labels are integrated into the symbolic element equations of the BSIM3 model addressing 25 physical effects as temperature, drain induced barrier lowering, operating ranges etc. Equation (3) is equivalent to (1), but now pre–labeled.

$$V_A = V_{As} + \left(1+\frac{P_v\cdot V_{Gt}}{E_s\cdot L_f}\right)\left(l_5\cdot l_7\cdot\frac{1}{\frac{1}{V_{AC}}+\frac{1}{V_{AD}}} + (l_5-1)\cdot l_7\cdot V_{AC} + (l_7-1)\cdot l_5\cdot V_{AD}\right) \qquad (3)$$

l_5 and l_7 are unique labels addressing the drain induced barrier lowering and channel length modulation effects. To allow a reformulation of a hole part of the equation with one unique label, the last row introduce some additional terms. l_5 and l_7 are also included in other equations (i_{DS} for example). Hence, one simplification step effects several parts of the DAE–system. Other unique labels for the other physical effects have been implemented in the same way.

The performance of this method is demonstrated in Figure 2. The dark bars show the number of unique labels,

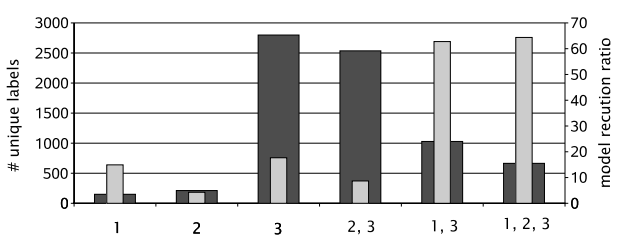

Figure 2. Model red. ratio vs. # unique labels

and the light bars present the model reduction ratio[1], both for the same model generation of an OTA example. Three different simplification methods are compared: ”1” is the pre–labeled transistor model, ”2” simplifies special transfer functions, and ”3” tries to eliminate every term in the DAE–system. Obviously, the new method achieves a simplification degree comparable to method ”2”, but with less than 10% of unique labels. In combination with the third or both others, it yields a much higher degree of simplification with less labels than the other two or the combination of them. The combination of method ”2” and ”3” yields a not as high model reduction ratio as the other two combinations, because a few simplification steps are identical.

The computational costs for the model generation is dominated by the simulation time. The number of analyses is proportional to the number of unique labels. Thus, using methods ”1” and ”3” or ”1”, ”2” and ”3” in combination, leads to the lowest computational costs for model generation. This way, not only the model reduction ratio was improved, but also the model generation time, in this example from 231 to 73 minutes.

6. Simulation Time Controlled Modeling

In the standard method, the model reduction ratio is the only goal of the simplification. However, each simplification is also causing a change of simulation time. Taking this into account, a new control criterion has been developed which takes the change of simplification time as the first control item. Only when the simulation time is equal or shorter than before, the simplification is checked whether it fulfills the error criterion. If the simulation time rises due to the simplification, it is withdrawn instantly.

Results for some examples show the impact on the simulation time. As can be seen in Figure 3, the speed–up of the simulation times for different analyses on the average equals a factor of two. Of course, this speed–up has a negative impact on the model reduction ratio resulting in models which are 10 to 30 % larger.

[1]The model reduction ratio is calculated with respect to the complexity of the DAE–system, i.e. the number and numerical complexity of the mathematical operations and the number of variables to be calculated (Refer to [22] for details). In theory, this should be approximately equal to the reduction of simulation time. In reality, it differs a lot due to computational overhead and the algorithms of the simulator.

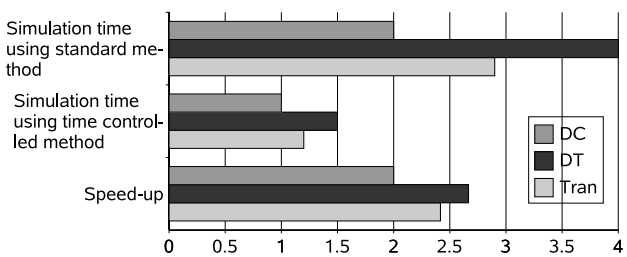

Figure 3. Simulation times and speed–up

7. Hierarchical Model Generation

Up to now the described method works for flat netlists only. Operating on hierarchical netlists should yield in better simulation time speed–up and would also allow model generation of larger circuits. Some further advantages can be expected:

1. Most industrial circuits are developed hierarchically, so their models can be generated directly without any need to flatten the netlists.
2. Identical subcircuits have to be modeled only once instead of several times. But this should be done only if different instances of one subcircuit are having almost the same operating conditions. Otherwise the model could not achieve the reduction ratio needed for a reasonable simulation time speed–up.
3. Matching of transistors like in differential pairs can be taken into account. This requires an appropriate hierarchical partitioning of the netlist.
4. The model generation times can be reduced drastically.

Several methods exists to find existing hierarchy in analog circuits. The most common method is the partitioning of the circuit by the designer himself. In [18], first attempts have been made to use automatic methods based on structural analysis of netlists in combination with symbolic analysis. Other methods exist for partitioning the DAE–system. Among them are relaxation methods [19], ε–decomposition [26], or dependence graphs [24].

The second and third methods are based on solving the Jacobian. However, they have the disadvantage that their results are based on one static solution. A repartitioning scheme would be needed for nonlinear circuits, taking dynamic solutions into account. The latter is a good alternative for finding partitions in DAE–systems. Any of these partitioning methods can be easily added as pre–modeling step. A rule based structural analysis was evaluated, but it is restricted in some respects. We prefer manual division of the netlist by the designer.

7.1. Method

The actual method of generating models from hierarchical netlists is straightforward. Instead of generating one model, a model for each part of the hierarchical netlist is generated. All instances of the same hierarchical circuit element must be identified first. Then one DAE–system is generated for all instances. In our method, the simplification process is performed as described before. This is done for all hierarchical circuit elements.

Different representations for the hierarchical circuit elements are possible: Netlists, unsimplified models, or, if available, already generated models. During the model generation of one of the hierarchical circuit elements, the representation of the others influence the simplification process. This influence is valid for the convergence of the whole hierarchical model and the error evoked by the simplification. The third case should be used to guarantee the convergence of the models at all time. Both other cases can cause convergence problems, because the simplifications already performed for one hierarchical circuit element are not recognized within the modeling process for the next one.

Another difficulty is the problem of finding the right sequence of the models to be generated and dividing the total maximum error and assign it to the single models. Three different error assignment methods have been implemented, called *identical*, *incremental*, and *individual*. Using the *identical* method, the full error value is used from the beginning without special assignments. In *incremental* mode, the error is divided by the number of models to be generated and assigned incrementally. The *individual* method allows separate values for each element.

7.2. Example

During the model generation process it is not known what role the sequence of the different hierarchical circuit elements plays, especially in combination with the different error assignment methods. Therefore, an experiment has been carried out with an OTA example. The OTA was divided into three hierarchical circuit elements. Subnetlists were chosen with taking care of matching transistors. All possible sequences have been used with the error assignment methods *identical* and *incremental*, always allowing a total maximum error of 5%. The model operating conditions were set to DC–transfer and transient. The error was calculated with respect to the output current of the OTA.

The results in terms of average values for the model reduction ratios for the three hierarchical circuit elements compared to the value of the flat netlist under the same model operating conditions are presented in Figure 4. The sum of the reduction ratios for the hierarchical models is more than twice as large as the reduction ratio for the flat netlist. This demonstrates that a properly chosen hierarchical netlist can result in smaller models and thus faster simulation times.

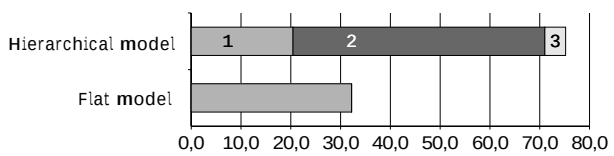

Figure 4. Model reduction ratio of the OTA

The speed–up of the simulation time is about three for the DC and the transient analysis and about two for the DC–transfer analysis. The models generated by error method *identical* simulate slightly faster than the others. A second observation is that sequences beginning with the biasing network show better speed–up values. Other examples show the same behavior, so that all of the following results will use a sequence where the N–MOS biasing network is modeled first, followed by the P–MOS biasing network, then the N–MOS active devices[2], as differential pairs and output stages, and at last the P–MOS active devices.

Because of the chosen structure of the hierarchical netlist, the model generation time of the OTA was reduced from 73 minutes to 50 minutes. The average speed–up of the hierarchical model versus the flat model is more than a factor of two, which clearly shows the advantage of the proposed hierarchical approach.

8. Industrial Examples

The new approaches have been applied to some industrial benchmark circuits: A gm–c filter (40 transistors), a differential lowpass filter (96 transistors), a charge pump (9 transistors) and a bandgap (10 transistors). Without using the hierarchical approach, the two filters could not be modeled because the DAE–system was too large. However, both have a very regular structure so that models can be generated by using appropriate hierarchies. The gm–c filter has been modeled within the AC and the transient domain, the lowpass filter within the AC domain, the charge pump and the bandgap in the transient domain. Different hierarchies have been used for the circuits, regarding their structures. The number of hierarchical circuit elements is five for the bandgap and four for the other three circuits. The error assignment method is always *identical* with a total error of 5%.

The speed–up of the different models compared to the netlist is between two and five (see Figure 5). Only the DC–analysis of the bandgap shows no speed–up. For the hierarchical circuit elements of each benchmark circuit, the model reduction ratios are summarized in Figure 6. Each hierarchical element is shown in a different color. The speed–up is not as high as could have been expected from the high

[2]Active device in this context marks an element in the signal path.

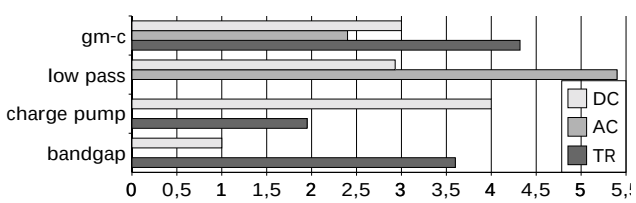

Figure 5. Simulation time speed–up

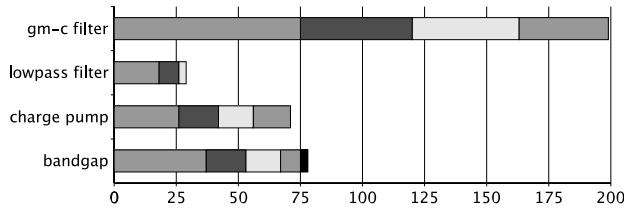

Figure 6. Model reduction ratios

reduction ratios, because the models do not contain any convergence aids and the models are not customized for a specific simulator, also. If the models could be formulated in a simulator adapted style with all convergence aids available, the simulation time speed–up would be approximately by a factor of 2 to 20 higher.

The total speed–up can be considered as the sum of an average factor of three from the new pre–labeled transistor model approach, an average factor of two from the simulation time controlled modeling, and a factor of at least three (depending on the netlist size) from the hierarchical model generation.

9. Conclusions

We presented an extended automatic method for behavioral modeling of nonlinear analog circuits based on nonlinear symbolic simplifications. For the first time, we were able to achieve simulation time speed–ups between three and five compared to the original netlist for various of industrial benchmark circuits up to 100 transistors. The models can be generated in standard behavioral languages and, thus, can be used in several different simulators.

This work has been supported by the BMBF (Federal Ministry of Education and Research, Germany) within the project ANASTASIA and ANASTASIA+.

References

[1] I. Analogy. SaberBook 5.0.1. *Release 5.0.1*, 1999.

[2] B. Antao and F. El-Turky. Automatic Analog Model Generation for Behavioral Simulation. *CICC 92*.

[3] E.-W. Bai. A Blind Approach to the Hammerstein-Wiener Model Identification. *15th Triennial World Congress 2002*.

[4] A. Bauer and W. Schwarz. Circuit Analysis and Optimization with Automatically Derived Volterra Kernels. *ISCAS '00*.

[5] C. Borchers, L. Hedrich, and E. Barke. Equation-Based Behavioral Model Generation for Nonlinear Analog Circuits. *DAC '96*.

[6] M. Bucolo, L. Fortuna, and M. Frasca. A Generalized Chua Cell for Realizing any Continuous n-Segment Piecewise-Linear Function. *International Journal of Bifurcation and Chaos*, 11(9), 2001.

[7] G. Casinovi. Automatic Generation of Models for Circuit Simulation. *MWSCAS '89*.

[8] G. Casinovi and A. Sangiovanni-Vincentelli. A Macromodeling Algorithm for Analog Circuits. *IEEE TCAD 1991*.

[9] P. Crama and J. Schoukens. First Estimates of Wiener-Hammerstein Systems Using a Random Multisine Excitation. *15th Triennial World Congress 2002*.

[10] P. Feldmann and R. Freund. Efficient Linear Circuits Analysis by Pade Approximation via the Lanczos Process. *IEEE TCAD*, 14(5), 1995.

[11] G. Gielen. Design Challenges and Emerging EDA Solutions in Mixed–Signal IC Design. *DATE '01*.

[12] J. Gomez and J. Alonso. A new approach for the extraction of SPICE compatible models from measured parameters of microwave circuits. *Microwave Symposium Digest 1998*.

[13] L. Ljung. System Identification: Theory for the User. *Prentice Hall, Inc., Englewood Cliffs, New York*, 1999.

[14] K. S. McCarvile and M. M. Hassoun. An element stamp implementation fo active device models for symbolic simulation. *Circuits and Systems 1994*.

[15] O. Mikulchenko and K. Mayaram. Neural Network Design for Behavioral Model Generation with Shape Preserving Properties. *IEEE/ACM International Workshop on Behavioral Modeling and Simulation 2000*.

[16] L. Näthke, R. Popp, L. Hedrich, and E. a. Barke. Using Term Ordering to Improve Symbolic Behavioral Model Generation of Nonlinear Dynamic Analog Circuits. *ECCTD '99*.

[17] O. Nelles. Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models. *Springer-Verlag, Berlin*, 2000.

[18] M. Olbrich, R. Popp, L. Näthke, L. Hedrich, and E. Barke. A Combined Structural and Symbolic Method for Automatic Behavioral Modeling of Nonlinear Analog Circuits. *ECCTD '01*.

[19] L. Peterson and S. Mattisson. Dynamic Partitioning for Concurrent Waveform Relaxation-Based Circuit Simulation. *ISCAS 93*.

[20] J. R. Phillips. Projection-Based Approaches for Model Recution of Weakly Nonlinear, Time-Varying Systems. *IEEE TCAD*, 22(2), 2003.

[21] R. Popp, W. Hartong, L. Hedrich, and E. Barke. Error Estimation on Symbolic Behavioral Models of Nonlinear Analog Circuits. *SMACD '98*.

[22] R. Popp, J. Oehmen, L. Hedrich, and E. Barke. Parameter Controlled Automatic Symbolic Analysis of Nonlinear Analog Circuits. *DATE '02*.

[23] M. Rewienski and J. White. A Trajectory Piecewise-Linear Approach to Model Order Recution and Fast Simulation of Nonlinear Circuits and Micormachined Devices. *IEEE TCAD*, 22(2), 2003.

[24] K. Swings, G. Gielen, and W. Sansen. An Intelligent Analog IC Design System based on Manipulation of Design Equations. *CICC 90*.

[25] P. Wambacq, G. Gielen, and W. Sansen. Symbolic Simulation of Harmonic Distortion in Analog integrated circuits with weak nonlinearities. *ISCAS '90*.

[26] A. Zecevic and N. Gacic. A Partitioning Algorithm for the Parallel Solution of Differential-Algebraic Equations by Waveform Relaxation. *IEEE TCS I*, 46(4), 1999.