Symbolic analysis of large analog integrated circuits by approximation during expression generation

F.V. Fernández[1,2], P. Wambacq[1], G. Gielen[1], A. Rodríguez-Vázquez[2] and W. Sansen[1]

[1] Departement Elektrotechniek, Katholieke Universiteit Leuven
Kardinaal Mercierlaan 94, B-3001 Heverlee, BELGIUM
Phone # 32 16 220931. FAX # 32 16 221855

[2]Dept. of Analog Circuit Design. Centro Nacional de Microelectrónica-Universidad de Sevilla
Edif. CICA, Avda. Reina Mercedes s/n, E-41012 Sevilla, SPAIN
Phone # 34 5 4239923. FAX # 34 5 4624506. Email: pacov@cnm.us.es

ABSTRACT

A novel algorithm is presented that generates approximate symbolic expressions for small-signal characteristics of large analog integrated circuits. The method is based upon the approximation of an expression while it is being computed. The CPU time and memory requirements are reduced drastically with regard to previous approaches, as only those terms are calculated which will remain in the final expression. As a consequence, the maximum circuit size amenable to symbolic analysis has largely increased. The simplification procedure explicitly takes into account variation ranges of the symbolic parameters to avoid inaccuracies of conventional approaches which use a single value. The new approach is also able to take into account mismatches between the symbolic parameters.

INTRODUCTION

Symbolic circuit analysis refers to the calculation of network functions $H(s,\boldsymbol{x})$ in the form:

$$H(s,\boldsymbol{x}) = \frac{f_0(\boldsymbol{x}) + sf_1(\boldsymbol{x}) + s^2 f_2(\boldsymbol{x}) + \ldots + s^N f_N(\boldsymbol{x})}{g_0(\boldsymbol{x}) + sg_1(\boldsymbol{x}) + s^2 g_2(\boldsymbol{x}) + \ldots + s^M g_M(\boldsymbol{x})} \quad (1)$$

where $\boldsymbol{x}^T=\{x_1, x_2, \ldots x_Q\}$ is the vector of circuit parameters which remain as symbols, and the coefficients of the s powers are polynomials in $\boldsymbol{x}$.

Symbolic analysis tools have drawn strong attention during the last few years. Their usefulness in such varied fields like automated equation-based analog sizing, fault diagnosis, etc., has been widely demonstrated [1].

Experience with symbolic simulators shows that resulting expressions increase exponentially with the size of the circuit under study. Precisely the immeasurable size of the expressions greatly limits the maximum circuit size capable to be analyzed. On the other hand, any computational application of symbolic expressions requires simplifications of these expressions, maintaining only those terms or subexpressions that are really significant. The same experience with symbolic simulators shows that, normally for real circuits, only a small part of the symbolic terms are really significant and are those which remain after the simplification process.

Thus, it is paradoxical that the analyzable size of circuits is limited by the exact symbolic expressions, which after simplification are never used again. However, in all the previous approximation techniques said expressions are indispensable to perform simplification.

The new idea reported in this paper consists in generating directly the simplified expression. If this is possible, two basic advantages arises. First, the analysis should be faster as no time is wasted in generating symbolic terms that would be pruned in the simplification procedure. Next, the memory needs will be much smaller and much larger circuits can be analyzed.

Conventional simplification approaches neglects the least significant terms based upon their relative magnitude, which is evaluated using a typical numerical estimate of the symbolic parameters, called the nominal point. This technique may yield large inaccuracies when the simplified expressions are used at points other than nominal [2]. The introduction of variation ranges instead of single numerical values has solved this problem in connection to conventional simplification *after* generation techniques. This has also been adapted to the new idea of simplification *during* generation.

APPROXIMATION DURING GENERATION

The new approach tries to generate the terms that appear at the simplified expression only, without the prior generation of the exact expression. In order to make this new approach feasible a technique able to generate symbolic terms in decreasing order of magnitude is needed. When a sufficient number of terms has been generated the algorithm must stop.

None of the classical symbolic analysis techniques is appropriate to this purpose. One possibility is to try combinations of symbols in decreasing order of magnitude and check if they are valid terms with any classical technique. But combinatorial explosion problems quickly arise with this approach.

A new analysis technique has been developed based on the undirected tree enumeration method. Although the directed tree method has been considered traditionally as advantageous over the undirected tree one, it will be shown that the latter one is much more adequate for our current problem.

Analysis technique

In the undirected tree approach a valid term corresponds to a spanning tree common to a voltage and a current graph, which are easily built from the circuit topology. Only passive elements and voltage controlled current sources (VCCS) are allowed in this technique. Any other type of controlled source must be converted to this type [3]. The voltage and current graphs are constructed with one branch between the terminal nodes of each passive element. The stamp of a VCCS is the controlled branch in the current graph and the controlling

branch in the voltage graph. Each valid term is given by the product of the admittances of every branch in the common spanning tree.

Our algorithm chooses one of these two graphs, i.e. the voltage graph, its spanning trees are generated in decreasing order of magnitude and for each generated tree it is checked if it is also a spanning tree in the current graph. The greatest inconvenience of this technique is the complicated calculation of the sign of each term in active circuits [4]. But in spite of being a tedious task in hand analysis, it can be efficiently programmed and experimental results show that the computation time is negligible compared with the other routines.

Our method has been developed starting from the algorithm proposed by Gabow to generate weighted spanning trees in order [5]. This algorithm uses a maximum weight spanning tree as a *reference tree* and exchanges branches to obtain the remaining trees. To obtain a maximum weight tree many algorithms have been reported [6]. The generation of trees in decreasing order of magnitude is based upon the concept of a *T-exchange* [5]. Given a spanning tree T of a graph G (the voltage graph in this case), a *T-exchange* is a pair of edges $[e,f]$ such that $e \in T$, $f \notin T$, and $T-e \cup f$ is a spanning tree. The weight of exchange $[e,f]$ is $w(f)-w(e)$, with $w(i)$ being the weight of edge i. So the weight of tree $T-e \cup f$ is the weight of tree T plus the weight of exchange $[e,f]$. It is obvious that the weight of every T-exchange must be negative.

Assume that the maximum spanning tree, T_1, has been obtained. For each tree branch of T_1 the T-exchange with smallest weight in magnitude is stored. The smallest T-exchange $[e,f]$ between all tree branches gives the following tree T_2. The remaining trees are conceptually splitted into two disjoints sets:

$$P_1^1 = \{T_k |\, k>1; e \in T_k\} \qquad P_2^1 = \{T_k |\, k>1; e \notin T_k\} \quad (2)$$

T_1 is reference tree for the first partition while T_2 is the reference tree for the second one.

Assume that the algorithm has already provided the first $j-1$ trees. The trees which have not yet been generated are splitted into $j-1$ disjoint partitions:

$$P_i^{j-1} = \{T_k |\, k>j-1; e_1, e_2, \ldots, e_r \in T_k; e_{r+1}, \ldots, e_s \notin T_k\} \quad (3)$$

with $e_1, e_2, \ldots, e_r, \ldots, e_s$ being edges of the undirected graph.

Among these sets the one that provides the T-exchange having the smallest weight is chosen. This gives a new spanning tree $T_j = T_i - e \cup f$, whose validity is checked in the current graph. The corresponding partition is deleted and two new partitions are generated:

$$\begin{aligned} P_i^j &= \{T_k |\, k>j; e_1, e_2, \ldots, e_r, e \in T_k; e_{r+1}, \ldots, e_s \notin T_k\} \\ P_j^j &= \{T_k |\, k>j; e_1, e_2, \ldots, e_r \in T_k; e_{r+1}, \ldots, e_s, e \notin T_k\} \end{aligned} \quad (4)$$

The other partitions remain unchanged.

We also store the set of minimum T-exchanges of each tree branch together with each partition. The sets of T-exchanges may be easily updated each time that a spanning tree is generated. Only those which have been affected by the last branch exchange performed have to be computed. This means important savings in CPU time with some additional memory consumption. The time complexity of this algorithm can be demonstrated to increase linearly with the number of spanning trees generated [5].

The described algorithm can also be adapted to the directed tree method. However, our experience shows that the number of term cancellations in the directed tree method increases exponentially with the circuit size and, hence, it is much less efficient that the undirected one.

Extension to High-Frequency Analysis

The described algorithm enables efficient calculation of symbolic terms which contain only conductances. If capacitors also appear, symbolic terms with powers of s greater than 0 have also to be generated. Given a circuit with n nodes, for each k power of s, spanning trees which contain exactly k capacitors and $n-k-1$ conductances must be enumerated.

Like the case of power 0 of s, the maximum spanning tree including k capacitors is used as reference and the rest of trees are obtained performing branch exchanges. In this case a spanning tree can be obtained from another by three kinds of T-exchanges: a T-exchange of two conductances; a T-exchange of two capacitors; and what we call a *double T-exchange*, consisting of a T-exchange of a conductance by a capacitor and a capacitor by a conductance. The smallest one among them is selected, providing the following tree in decreasing order of magnitude. If this tree is obtained by a simple T-exchange two new partitions result, as in the dc case. By the contrary, if the new tree result from a double T-exchange, $[g_n \cup c_p, c_n \cup g_p]$, then, four new partitions must be created which cover all spanning trees that have not yet been generated:

$$\begin{aligned} P_i^j &= \{T_k |\, k>j; e_1, e_2, \ldots, e_r, g_n, c_p \in T_k; e_{r+1}, \ldots, e_s \notin T_k\} \\ P_l^j &= \{T_k |\, k>j; e_1, e_2, \ldots, e_r, g_n \in T_k; e_{r+1}, \ldots, e_s, c_p \notin T_k\} \\ P_m^j &= \{T_k |\, k>j; e_1, e_2, \ldots, e_r, c_p \in T_k; e_{r+1}, \ldots, e_s, g_n \notin T_k\} \\ P_q^j &= \{T_k |\, k>j; e_1, e_2, \ldots, e_r \in T_k; e_{r+1}, \ldots, e_s, g_n, c_p \notin T_k\} \end{aligned} \quad (5)$$

STOPPING CRITERION

Obviously a stopping criterion is needed in order to know when a sufficient number of terms have been generated.

Conventional simplification criteria discards the non-significant terms in each coefficient of (1), beginning with the smallest one, until a relative error, ε_M, between the exact expression and the approximated one is reached [7,8]. In the new approach no term is discarded. Instead, for each coefficient $h_k(\boldsymbol{x})$ in numerator or denominator of (1) the R most significant terms are generated using the above described analysis technique until the following is met:

$$\frac{\left| \sum_{l=1,R} h_{kl}(\boldsymbol{x}_o) \right|}{\left| \sum_{l=1,T} h_{kl}(\boldsymbol{x}_o) \right|} > (1-\varepsilon_M) \quad (6)$$

in which the different terms are evaluated at the nominal point $\boldsymbol{x}_o$. The denominator in (6) represents the numerical evaluation of the complete symbolic polynomial at $\boldsymbol{x}_o$. The need to evaluate a priori the magnitude of the symbolic expression is obvious since the philosophy of the new approach is to not generate such expression. This can be efficiently computed using the numerical interpolation method which is the most

suitable technique to calculate the system function of a given circuit with the complex frequency s as only variable.

However, the nominal point approach does not seem to be very consistent with the own nature of symbolic analysis. The use of variation ranges instead of nominal values has been proposed [2]. In the variation range approach each symbolic parameter is assumed to take any value inside a given interval,

$$x_i \in [x_{iL}, x_{iH}] \tag{7}$$

where x_{iL} and x_{iH} are real numbers and $x_{iL} \leq x_{iH}$. Basic operations between ranges may be found in any basic book on interval analysis [9].

A conservative approach to generate the most significant terms in each coefficient of (1) taking into account variation ranges is to apply the following formula:

$$\frac{\mathcal{L}(|[G_L, G_H]|)}{\mathcal{U}(|[S_L, S_H]|)} > (1 - \varepsilon_M) \tag{8}$$

where $[S_L, S_H]$ represents the range of the sum of all the terms included in the coefficient being simplified, and $[G_L, G_H]$ denotes the range corresponding to the sum of the R most significant terms. $\mathcal{L}$ and $\mathcal{U}$ denote the lower and upper bounds of the range, respectively [2].

Notice that the denominator of (8) contains the range of the sum of all symbolic terms. Hence, this range must be calculated a priori. In previous approaches this was not a difficult problem as the complete symbolic expression was available before performing the simplification [2]. Now, it is not a trivial problem since the symbolic expression is not available to calculate its range.

The classical numerical interpolation method for real polynomials can be extended to variation ranges by substitution of real variables by interval ones and real arithmetic operations by the corresponding interval arithmetic operations. The resulting range is an interval extension of the symbolic polynomial, that is usually a too pessimistic overestimate. Such overestimate can be reduced by the application of the Skelboe algorithm [10]. Successive refinements are used in this algorithm to obtain first the lower bound and then the upper bound. Basically, it consists in the calculation of the interval extension of the polynomial in a finite subsequence of regions producing the smallest lower bounds. A successive subdivision of multidimensional intervals and the inclusion in an ordered list provides interval extensions as accurate as desired.

MISMATCHING HANDLING

Matching considerations have become a fundamental issue in analog integrated circuit design. Some second-order characteristics are mainly determined by small mismatches between nominally matched devices. Adequate mismatching handling has been a major concern in modern symbolic simulators like ASAP [8] and ISAAC [7].

If several devices are perfectly matched they are represented by the same symbolic parameter. Its main effect is the introduction of cancellations between equal terms. This means an important consumption of resources in the detection and execution of cancellations with conventional tools. This is not a problem in the new approach as terms that cancel have the same numerical value and, hence, they are generated one immediately after the other and the cancellations can be performed easily.

It is usually very interesting to express device mismatches as explicit mismatching symbols. These are represented in our approach as separate circuit elements in parallel with the corresponding nominal elements. Hence, terms containing mismatching parameters are generated, according to their relative magnitude, only when necessary.

The variation range approach is specially adequate to handle mismatches. Conventionally, mismatching parameters were assigned a sign and a value to evaluate the relative significance of symbolic terms. Instead, we assign each mismatching parameter a symmetrical interval around zero, which is much more in accordance with the philosophy of mismatching.

EXPERIMENTAL RESULTS

The advantages of the new technique over previous approaches will become obvious through different examples.

First we will consider the academic benchmark circuits of Fig.1: a resistive ladder network (extremely sparse circuit) and a fully connected network (extremely dense circuit). Fig.2 shows the CPU time as a function of the number of nodes for the symbolic computation of simplified expressions of the voltage gain with a 25% error. All CPU times have been measured on a SPARCstation 10. In both cases, the CPU time can be observed to increase dramatically using a conventional symbolic analyzer, in which the complete symbolic expression has to be generated first. The number of terms of this expression has been shown to increase exponentially [3]. However, the new technique can be observed to be several orders of magnitude more efficient to generate the simplified expressions. Moreover, the new approach is able to analyze much more complex circuits than the conventional technique.

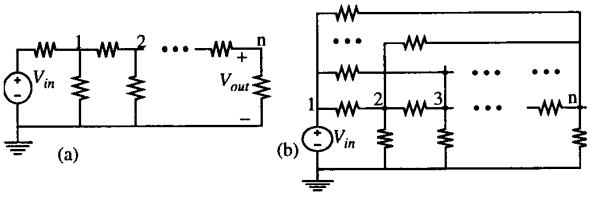

Figure 1: Benchmark circuits: (a) Resistive ladder; (b) Fully-connected network.

Let us consider now some real-life benchmark circuits. Fig.3 shows a fully-differential folded-cascode OTA with linearized input stage. ASAP spends 205s. of CPU time in providing a simplified expression of the differential voltage gain with a 25% error. The same expression is obtained with the new technique in only 0.1s.

The circuit of Fig.4 is used as a second benchmark circuit. A simplified expression of the voltage gain is provided by ASAP in 308 s. The new approach provides the same expression in 17.8s. It can be observed that the CPU time is comparatively much larger than for the circuit of Fig.3. The reason is found in the use of the variation range approach. The denominator of this circuit is very sensitive to small variations in the transconductance of the amplifiers, its range estimate includ-

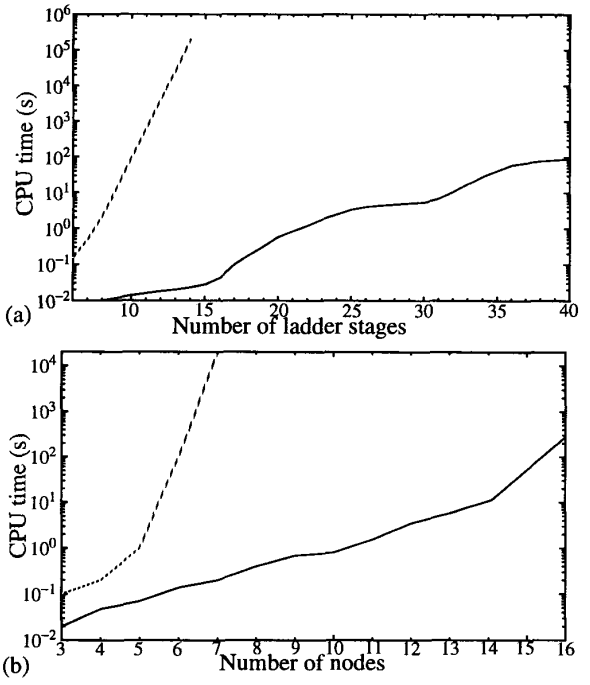

Figure 2: CPU time comparison for the calculation of the voltage gain of the circuits in Fig.1: (a) Ladder network; (b) Fully-connected network. The dotted line corresponds to ASAP and the solid line to the new approach.

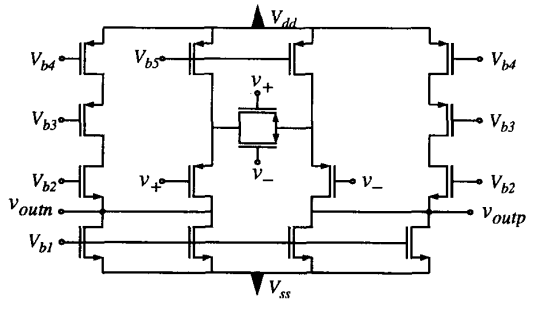

Figure 3: Fully-differential folded-cascode OTA.

ing zero. Hence, the denominator cannot be simplified safely and all its terms had to be generated.

As a final example consider the folded-cascode opamp with large output-signal swing of Fig.5. Conventional symbolic analyzers are unable to analyze a circuit of this size. However, the new approach provides a simplified expression of the voltage gain in only 54.7s.

Acknowledgements

The authors wish to thank Philips Eindhoven, The Netherlands, and the Human Capital and Mobility Program of the CEC for their support.

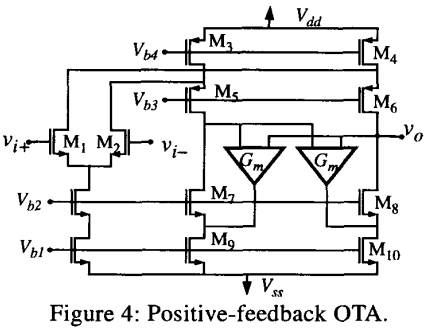

Figure 4: Positive-feedback OTA.

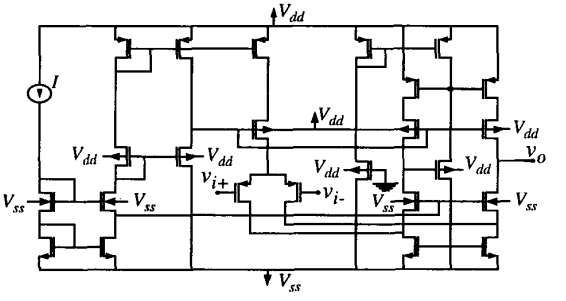

Figure 5: Folded-cascode opamp.

REFERENCES

[1] A. Rodríguez-Vázquez, F.V. Fernández and J.L. Huertas, eds., *Symbolic Analysis Techniques and Applications to Analog Design Automation*, IEEE Press, 1994

[2] F.V. Fernández, J.D. Martín, A. Rodríguez-Vázquez and J.L. Huertas, "Formula Approximation for Flat and Hierarchical Symbolic Analysis", *Analog Integrated Circuits and Signal Processing*, Vol. 3, pp. 43-58, Kluwer,1993.

[3] P.M. Lin, *Symbolic Network Analysis*, Elsevier, 1991.

[4] S.P. Chan, *Introductory Topological Analysis of Electrical Networks*, Holt, Rinehart and Winston, 1969.

[5] H.N. Gabow, "Two Algorithms for Generating Weighted Spanning Trees in Order", *SIAM J. of Computing*, Vol. 6, No. 1, pp. 139-150, March 1977.

[6] D. Cheriton and R.E. Tarjan, "Finding Minimum Spanning Trees", *SIAM J. of Computing*, Vol. 5, No. 4, pp. 724-742, Dec.1976.

[7] G. Gielen, H. Walsharts and W.Sansen, "ISAAC: A Symbolic Simulator for Analog Integrated Circuits", *IEEE J. Solid-State* Circ., pp. 1587-1597, Dec. 1989.

[8] F.V. Fernández, A. Rodríguez-Vázquez and J.L. Huertas, "Interactive AC Modeling and Characterization of Analog Circuits via Symbolic Analysis", *Analog Integrated Circuit and Signal Processing*, Vol. 1, pp. 183-208, Kluwer, Nov. 1991.

[9] R.E. Moore, *Methods and Applications of Interval Analysis*, Studies in Applied Mathematics, 1979.

[10]S. Skelboe, "Computation of Rational Interval Functions", *BIT*, No. 14, pp. 87-95, 1974.