Efficient Symbolic Computation of Approximated Small-Signal Characteristics of Analog Integrated Circuits

Piet Wambacq, Francisco V. Fernández, Georges Gielen, Willy Sansen, and Angel Rodríguez-Vázquez

Abstract—**A symbolic analysis tool is presented that generates simplified symbolic expressions for the small-signal characteristics of large analog integrated circuits. The expressions are approximated while they are computed, so that only those terms are generated which remain in the final expression. This principle causes drastic savings in CPU time and memory, compared with previous symbolic analysis tools. In this way, the maximum size of circuits that can be analyzed, is largely increased. By taking into account a range for the value of a circuit parameter rather than one single number, the generated expressions are also more generally valid. Mismatch handling is explicitly taken into account in the algorithm. The capabilities of the new tool are illustrated with several experimental results.**

I. Introduction

CURRENT tools for small-signal symbolic analysis of analog integrated circuits, like for instance ISAAC [1] and ASAP [2], are able to evaluate network functions in the s-domain with the complex frequency variable and the circuit parameters (capacitances, resistances, transconductances, etc.) kept as symbols. These functions are typically given as an expanded cancellation-free sum of products,

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

in which $\boldsymbol{x}^T = \{x_1, x_2, \cdots, x_Q\}$ is the vector of symbolic circuit parameters and the f_i and g_j are sums of products.

Since these expressions are calculated automatically, analog designers are released from the involved calculations needed to get insight into the ac behavior of circuits. Also, analog cells can be automatically sized for given ac specifications through the iterative optimization of the symbolic equations generated for their gain, poles, zeros, terminal impedances, PSRR, CMRR, etc. Other potential applications of symbolic analyzers, for synthesis, statistical optimization, testability, etc., exploit also the computational advantages to perform repetitive evaluations of precalculated models [3]. However, these applications can be realized at fully only if the automatic generation of symbolic expressions runs parallel to the automatic pruning of insignificant terms in these expressions—similar to what expert analog designers do when they analyze circuits by hand.

Although existing analyzers like ISAAC and ASAP incorporate such simplification feature, their algorithms have two important drawbacks: a) simplifications are performed only after the exact symbolic expression is generated in an expanded sum-of-product format; and b) the significance of each term in the sums-of-products is assessed on the basis of numerical evaluations using typical values of the circuit parameters, at a single point of the design parameter space. Since the size of the exact symbol expressions increases exponentially with the number of nodes and elements in the circuit, the first drawback puts an upper limit on the complexity of analyzable circuits; around ten transistors if each transistor is represented by a high-frequency model containing about nine circuit elements. On the other hand, approximating symbolic expressions by considering just a single point of the parameter space does not seem to be consistent with the very nature of the symbolic analysis procedure, where the exact numerical value of the parameters is, by definition, unknown *a priori*. Even in the case symbolic analysis is used to study critical parameter variations in an already sized schematic, simplifying by using just information about a nominal point may lead to important inaccuracies in mismatch-sensitive characteristics, as for instance PSRR or CMRR of operational amplifiers.

This paper presents a simplification algorithm to overcome both drawbacks above. First of all, the complexity limits of analyzable circuits are extended by generating only the dominant terms, without first computing the complete exact expression. In this approach, which is denoted as *simplification during generation*, the dominant terms are generated until the accuracy falls within a given user-supplied accuracy. As shown in Fig. 1, this is a much more efficient approach, both in terms of memory usage and CPU time, than the classical approach followed in [1], [2]. The idea of simplification during generation was first mentioned in [4], and later found also in [5] and [6]. However, simplifications in [4] and [6] are performed at a nominal point of the design space, and, consequently, any evaluation of the expressions in another operating point might cause large errors. To reduce these errors, this paper further elaborates the approach in [5] to combine the concept of simplification during generation with the use of ranges [7], instead of single values, for the circuit design parameters. This increases the compliance of generated expressions, while keeping the computation time and the memory resources needed for symbolic analysis of large analog circuits bounded. Also, the combination of both

Manuscript received July 13, 1994; revised November 4, 1994

P. Wambacq, G. Gielen, and W. Sansen are with Katholieke Universiteit Lueven, Dep. Elektrotechniek, ESAT-MICAS, B-3001 Heverlee, Belgium.

F. V. Fernandez and A. Rodríguez-Vázquez are with Department of Analog Circuit Design, Centro Nacional de Microelectrónica, Edif. CICA, E-41012 Sevilla, Spain.

IEEE Log Number 9408739.

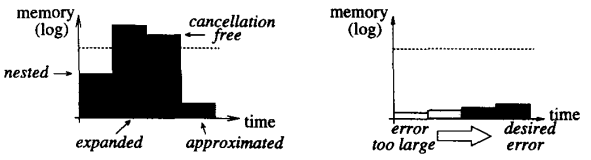

Fig. 1. Schematic representation of the memory usage during simulation time with the classical approach (a) and with the proposed approach of simplification during generation (b). Classically, a symbolic expression is first generated in a nested format. For a reliable approximation, the expression is then expanded and the cancelling terms are elaborated. This expansion can lead to a huge number of terms whose storage exceeds the memory limits (dashed line), while only a few terms are retained after approximation. This problem is circumvented if the expression is simplified when it is generated: the memory usage increases with the required accuracy or the number of terms of the symbolic expression.

techniques, simplification during generation and the use of ranges, demonstrate better results than previous approaches for the handling of matching between symbolic parameters and mismatches.

Section II presents the concept and outlines the algorithms used for simplification during generation. Section III explains how intervals are incorporated in the stopping criterion that controls the generation of terms. In Section IV it is explained how matching elements and corresponding mismatches are handled in the new approach. Finally, Section V presents examples that demonstrate the suitability of the techniques presented for analog cells containing more than 20 transistors, which approaches the size of practical circuits used in todays IC designs.

II. Simplification During Generation

The idea of simplification during generation needs a term by term generation mechanism, which finds the terms in decreasing order of magnitude, without skipping any term. This can be achieved with the undirected tree enumeration method [8]. This is a topological method that operates on two weighted graphs, the voltage graph and the current graph, which are easily derived from the given (small-signal) network. A term is valid only if its corresponding branches constitute a spanning tree in both graphs. The symbolic term is given by the product of the branch weights (admittances) in any of the graphs. The sign of a term is determined separately, using topological information of both graphs. By augmenting the network in a special way with fictitious elements, it is possible to generate the terms for both numerator and denominator at the same time [8].

The number of trees increases exponentially with the circuit size. Since we are interested only in the dominant terms and therefore not in all trees, the new algorithm enumerates spanning trees in the voltage graph in decreasing order. For every spanning tree, it is checked whether the corresponding branches in the current graph constitute a spanning tree as well. If so, a valid term is found and its sign is determined.

This technique is performed for every power of the frequency variable s in both the numerator and denominator of the network function. For a nonzero power of s, say the k-th power, spanning trees in decreasing order must be generated containing exactly k capacitance branches. This can be formulated as the following graph-theoretical problem: given a graph with n nodes and with red (corresponding to (trans)conductances) and blue (corresponding to capacitances) weighted branches, enumerate in decreasing order the spanning trees that contain exactly k blue branches and $n-k-1$ red branches, in which k can have a value between zero and $n-1$. For this problem, an algorithm [9] has been developed whose time complexity and memory requirements increase linearly with the number of generated spanning trees.

III. Generation of the Numerical Reference and Approximation Over Ranges

The tree enumeration procedure described above obviously needs a stopping criterion to know when enough terms have been generated. The generation of terms for a certain power k of s in the enumerator or denominator can stop when

$$\frac{|\sum \textit{num. evaluation of generated terms}|}{|\textit{num. value of coefficient of } s^k|} > (1-\epsilon_k). \quad (2)$$

In this equation, the numerical evaluation is performed in a nominal operating point of the circuit. The denominator in (2) represents the numerical value of the coefficient of s^k in either the numerator or denominator of the network function. The complete coefficients are never generated and, hence, their numerical value must be calculated in advance (without knowing the symbolic expressions). This is efficiently performed using the polynomial interpolation method [10].

For the extension of the stopping criterion of (2) to intervals, it is assumed that a symbolic parameter x can take a value inside a given interval determined by its lower bound x_L and its upper bound x_H.

The introduction of intervals for the symbolic circuit parameters gives rise to multidimensional intervals for the value of the coefficients f_i and g_j from (1). These are computed by an interval extension of the polynomial interpolation method. The resulting interval for a coefficient is usually a pessimistic overestimate. Therefore, intervals are narrowed using the algorithm described in [11].

Intervals for the small-signal circuit parameters are either determined by specifying a relative variation around a given nominal value, or they can be derived from intervals of the bias values and technological parameters. Intervals for symbolic terms or sums of products are then determined using the direct interval extension [11], or, more accurately, with the mean value form [11].

The stopping criterion given in (2) can now be reformulated as:

$$\frac{\mathcal{L}(|[G_L, G_H]|)}{\mathcal{U}(|[S_L, S_H]|)} > (1-\epsilon_k). \quad (3)$$

In this equation $[S_L, S_H]$ represents the interval of the coefficient of s^k obtained as described above. The interval $[G_L, G_H]$ denotes the interval of the sum of the significant terms that have already been generated. The symbols $\mathcal{L}$ and $\mathcal{U}$ denote the lower and upper bound of an interval, respectively.

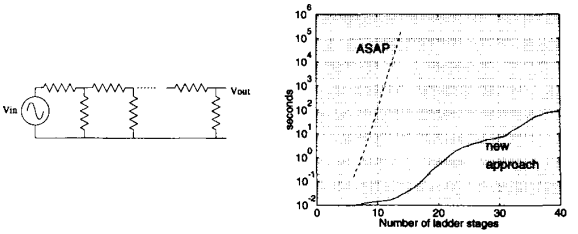

Fig. 2. CPU time on a SUN SPARC 10 for the symbolic computation with a 25% error of the voltage gain V_{out}/V_{in} of the resistive ladder network. The dotted line corresponds to times measured with ASAP (conventional symbolic analyzer). The solid line corresponds to the new approach.

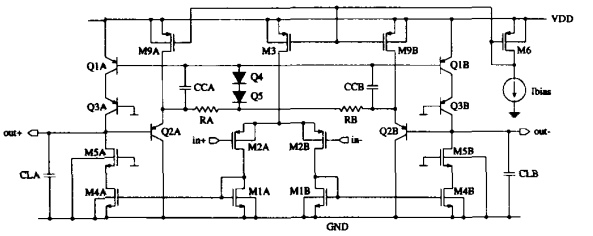

Fig. 3. A fully differential BiCMOS operational transconductance amplifier with common-mode feedback.

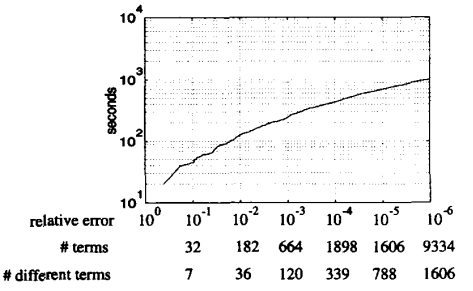

Fig. 4. CPU time (in seconds on a SUN SPARC 10 workstation) versus the relative error of the generated symbolic expression for the denominator of the low-frequency differential-mode gain of the BiCMOS amplifier (Fig. 3). The number of terms that corresponds to the accuracy is indicated as well.

The use of intervals provides a very good trade-off between accuracy and complexity. Obviously, more terms appear in the final result than when using fixed values, and if the intervals are taken too wide, then the interpretation of results can become complicated again.

IV. Matching Elements

Matching elements play an important role in analog and especially in differential integrated circuits. In symbolic calculations they are represented by the same nominal symbol. After doing so, product terms can occur with exactly the same symbols, so that they cancel or add, depending on their sign. The detection of matching terms requires a lot of overhead in CPU time and memory consumption in conventional symbolic analyzers [1], [2]. With the new technique, however, matching terms are easily detected: since they are equal in magnitude, they are generated one immediately after the other. Hence, the cancellations can be elaborated by looking only at the last few generated terms that have the same magnitude as the last generated term.

Mismatches are modeled explicitly by adding a small symbolic mismatch element in parallel with the nominal element. From that moment, both elements are handled independently, and with their own numerical magnitude. For example, the transconductances of two matching transistors M_{1A} and M_{1B} are written as $g_{m_{M1}}$ and $g_{m_{M1}} + \Delta g_{m_{M1BA}}$, respectively. In this way, product terms containing mismatch symbols are generated much later than the corresponding nominal terms and only when necessary.

In techniques previously used [1], [2] in symbolic analyzers, mismatch terms were always given a magnitude (the maximum deviation) and a sign. This is not realistic, since their sign is not known in advance. This problem is overcome here by representing a mismatch term by a symmetric interval around zero.

V. Examples

The new technique not only exceeds the limits of a conventional symbolic analyzer, it can also—for smaller circuits—generate an approximate expression in a CPU time that is up to several orders of magnitude smaller than with conventional analysis. This is shown with the symbolic analysis of the resistive ladder network shown in Fig. 2, which is often used as a benchmark circuit for symbolic analyzers. The CPU time is shown as a function of the number of stages for the symbolic computation of the voltage gain with a 25% error. The dramatic increase in CPU time with the number of stages for conventional symbolic analysis is due to the fact that the exact expression must be generated.

The efficiency of the simplification during generation technique in terms of CPU time is illustrated with the symbolic computation of the system determinant of the BiCMOS amplifier of Fig. 3. This circuit, containing twenty transistors, is far too complex to be analyzed with classical symbolic analyzers. Fig. 4 indicates how with the new technique the CPU time increases with the accuracy of the generated symbolic expression and hence with the number of terms, just as with the principle idea shown in Fig. 1. In contrast with the conventional symbolic analysis approaches, the less terms are generated (the larger the error), the less CPU time is required, which is a very "natural" way of generating terms that constitute a large expression.

For large circuits complicated expressions may be generated. This is illustrated in Fig. 5, which shows the symbolic expression of the low-frequency differential-mode gain of the amplifier of Fig. 3. This expression has been generated in 58 s on a SUN Sparc 10. The expression, however, can be further simplified without increasing the error by a symbolic postprocessing procedure, that factorizes the expressions and that takes advantage of the fact that the error on a ratio of two coefficients of a network function is often much smaller than the error on the coefficients individually. Doing so, the

$$
\begin{aligned}
&\Big(4\,gm_{Q2}^2\,gm_{Q3}^2\,gm_{M6}\,gm_{M2}^2\,gm_{M5}^2\,gm_{M1}\,gm_{Q5}\,gm_{M4}\,gm_{Q4}\,Ga\,gm_{Q1} \\
&+ 8\,gm_{Q2}^2\,gm_{Q3}^2\,gm_{Q5}\,gm_{M2}^2\,gmb_{M5}\,gm_{M5}\,gm_{M1}\,gm_{M6}\,gm_{M4}\,gm_{Q4}\,Ga\,gm_{Q1} \\
&+ 4\,gm_{Q1}\,gm_{Q3}^2\,gm_{Q5}\,gm_{M2}^2\,gm_{Q2}^2\,gmb_{M5}^2\,gm_{M1}\,gm_{M6}\,gm_{M4}\,gm_{Q4}\,Ga \\
&+ 4\,gm_{Q2}^2\,gm_{Q3}^2\,go_{M1}\,gm_{M2}^2\,gm_{Q5}\,gm_{M5}^2\,gm_{M6}\,gm_{M4}\,gm_{Q4}\,Ga\,gm_{Q1}\Big) \Big/ \\
&\Big(8gm_{Q2}^2\,gm_{Q3}^2\,go_{M1}\,gm_{M2}\,gm_{M1}\,go_{M4}\,gm_{Q5}\,Ga\,gm_{M5}\,gm_{M6}\,go_{M5}\,gm_{Q4}\,gm_{Q1} \\
&+ 4\,gm_{Q2}^2\,gm_{Q3}^2\,Ga\,gm_{M1}^2\,go_{M4}\,gm_{Q5}\,go_{M5}^2\,gm_{M6}\,gm_{M2}\,gm_{Q4}\,gm_{Q1} \\
&+ 8\,g\pi_{Q2}\,gm_{Q2}\,gm_{Q3}^2\,Ga^2\,gm_{M2}\,gm_{M1}^2\,gmb_{M5}\,gm_{Q5}\,gm_{M5}\,gm_{M6}\,gm_{Q4}\,gm_{Q1} \\
&+ 4\,g\pi_{Q2}\,gm_{Q2}\,gm_{Q3}^2\,Ga\,gm_{M1}^2\,gm_{Q5}\,gm_{M5}^2\,gm_{M6}\,geq2\,gm_{M2}\,gm_{Q4}\,gm_{Q1} \\
&+ 4\,gm_{Q2}^2\,gm_{Q3}^2\,gm_{M1}^2\,gm_{M2}\,go_{M4}\,gm_{Q5}\,Ga\,gm_{M5}\,gm_{M6}\,go_{M5}\,gm_{Q4}\,gm_{Q1} \\
&+ 4\,gm_{Q2}^2\,gm_{Q3}^2\,Ga\,gm_{M1}^2\,go_{M4}\,gmb_{M5}\,gm_{Q5}\,gm_{M6}\,go_{M5}\,gm_{M2}\,gm_{Q4}\,gm_{Q1} \\
&+ 4\,g\pi_{Q2}\,gm_{Q2}\,gm_{Q3}^2\,Ga^2\,gm_{M2}\,gm_{M1}^2\,gm_{Q5}\,gm_{M5}^2\,gm_{M6}\,gm_{Q4}\,gm_{Q1} \\
&+ 4\,gm_{Q2}^2\,gm_{Q3}^2\,gm_{M2}\,gm_{M1}^2\,go_{M4}^2\,gm_{Q5}\,Ga\,gm_{M6}\,go_{M5}\,gm_{Q4}\,gm_{Q1}\Big)
\end{aligned}
$$

Fig. 5. Approximated expression ($\epsilon = 20\%$) for the low-frequency differential-mode gain of the circuit of Fig. 3. The terms are sorted in decreasing order. Due to matching, several product terms occur more than once, which explains the occurance of integer coefficients 4 and 8. The element $geq2$ is a lumped element [1] consisting of the parallel conductances go_{M9} and go_{Q2}. Elements from the bias circuitry (like gm_{M6}) or from the common-mode feedback circuitry (like gm_{Q4}) that don't influence the differential-mode gain at all, disappear after factorization.

differential-mode gain reduces to

$$\frac{gm_{M2}gm_{M4}}{gm_{M1}\left(\frac{go_{M4}go_{M5}}{gm_{M5}+gm_{M5}}+\frac{Ga+geq2}{\beta_{Q2}}\right)}. \tag{4}$$

It is found that the output conductance of the bipolar cascode is too small, even with the inclusion of intervals, to contribute significantly to the conductance seen at the output node.

An even more complex circuit is the commercial μA741 opamp. This circuit contains 23 nodes, 22 transistors, and 13 resistors. The generation of a symbolic expression for the amplifier's transfer function with an error of 0.1% (110 terms) requires 38 seconds on a SUN Sparc 10 workstation.

VI. Conclusion

A new program has been presented that generates approximated symbolic expressions for small-signal characteristics of analog circuits. The approximation is performed during the generation of the expression. In this way, only the necessary terms of the simplified symbolic expressions are generated, which contrasts to approaches of conventional symbolic analyzers which require a lot of over head for the generation of the exact symbolic expression, which is then pruned. The new approximation technique also takes into account a range for the value of the symbolic circuit parameters rather than one single value. This extends the range of validity of the generated symbolic expressions. Moreover, the new technique allows an accurate control of the approximation error. The interpretability of the expressions can be enhanced by further postprocessing. Several examples have demonstrated that this approach enables the symbolic analysis of large analog integrated circuits of the size of practical analog cells, which were impossible to analyze properly before.

Acknowledgment

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

References

[1] G. Gielen and W. M. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits.* Norwell, MA: Kluwer Academic, 1991.
[2] F. V. Fernández, A. Rodríguez-Vázquez, and J. L. Huertas, "Interactive ac modeling and characterization of analog circuits via symbolic analysis," *Kluwer Journal on Analog Integrated Circuits and Signal Processing*, vol. 1. Norwell, MA: Kluwer, 1991, pp. 183–208.
[3] G. Gielen, P. Wambacq, and W. M. Sansen, "Symbolic analysis methods and applications for analog circuits: A tutorial overview," *Proc. IEEE*, vol. 82, pp. 287–304, Feb. 1994.
[4] P. Wambacq, G. Gielen, and W. M. Sansen, "A cancellation-free algorithm for the symbolic analysis of large analog circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1992, pp. 1157–1160.
[5] P. Wambacq, F. V. Fernández, G. Gielen, and W. M. Sansen, "Efficient symbolic computation of approximated small-signal characteristics," in *Proc. CICC 1994*, 1994, pp. 21.5.1–21.5.4.
[6] Q. Yu and C. Sechen, "Generation of color-constrained spanning trees with application in symbolic circuit analysis," in *Proc. 4th Great Lakes Symp. VLSI*, Mar. 1994, pp. 252–255.
[7] F. V. Fernández, J. D. Martín, A. Rodríguez-Vázquez, and J. L. Huertas, "On simplification techniques for symbolic analysis of analog integrated circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1149–1152, May 1992.
[8] P.-M. Lin, *Symbolic Network Analysis.* Amsterdam, The Netherlands: Elsevier, 1991.
[9] P. Wambacq, F. V. Fernández, G. Gielen, W. M. Sansen, and A. R odríguez-Vázquez, "An algorithm for efficient symbolic analysis of large analogue circuits," *IEE Electron. Lett.*, vol. 30, no. 14, pp. 1108–1109, July 1994.
[10] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and design.* New York: Van Nostrand Reinhold, 1983.
[11] R. Moore, *Methods and Applications of Interval Analysis*, Studies in Applied Mathematics, Philadelphia, 1979.