Symbolic Analysis of Large Analog Circuits Using a Sensitivity-Driven Enumeration of Common Spanning Trees

Piet Wambacq, Petr Dobrovolný, Georges G. E. Gielen, and Willy Sansen, *Fellow, IEEE*

***Abstract*—A new approach for the generation of approximate symbolic network functions is presented. This approach is used to analyze large analog integrated circuits. It is based on a matroid intersection algorithm that directly enumerates common spanning trees of a two-graph representation of a given circuit. The approximation algorithm is based on an inspection of the sensitivity of the magnitude of a network function with respect to the different coefficients of a network function. These sensitivities as a function of frequency control the enumeration process. In this way, the algorithm enumerates a minimum number of the dominant terms for each coefficient of the network function. The complete algorithm runs in $O(Kmn^3)$ time, in which K is the average number of matroid intersections that must be generated for the approximation of each coefficient of the network function, n is the number of nodes in the linearized network and m the number of circuit elements. At the end of the paper experimental results are presented. These results indicate that this new approach is superior to other approaches to generate an approximate symbolic network function from a given two-graph.**

***Index Terms*—Analog integrated circuits, circuit analysis, circuit modeling, circuit simulation, design automation, graph theory, symbol manipulation.**

I. Introduction

IN THE past few years, symbolic analysis of linearized analog integrated circuits has been a major topic of research [1], [2]. The main reason is that symbolic analysis can improve the insight and understanding of the functioning of analog circuits and therefore can accelerate the design process of analog integrated circuits. The symbolic techniques can be used for the fast automatic generation of equations that describe the circuit performance. These equations can be used either for interpretation or for repeated evaluation in design automation applications. Many symbolic analysis programs have been published during the last years: ISAAC [3], ASAP [4], SYNAP [5], SAPEC [6], SSPICE [7], SCYMBAL [8], SCAPP [9], GASCAP [10], ANALOG INSYDES [11], SANTAFE [12], ADAGIO [13], SIFTER [14], RAINIER [15], ⋯. Some of these tools are able to compute exact symbolic expressions only. Others can generate approximate expressions which is absolutely required to avoid over-lengthy expressions for large semiconductor circuits. These expressions are in fact network functions, which have the general form

$$T = \frac{f_0(\mathbf{x}) + sf_1(\mathbf{x}) + s^2 f_2(\mathbf{x}) + \cdots + s^p f_p(\mathbf{x})}{g_0(\mathbf{x}) + sg_1(\mathbf{x}) + s^2 g_2(\mathbf{x}) + \cdots + s^q g_q(\mathbf{x})}. \quad (1)$$

In this equation $\mathbf{x}^T = \{x_1, x_2, \cdots, x_Q\}$ is the vector of symbolic circuit parameters. The coefficients f_i $(i = 0, \cdots, p)$ and g_j $(j = 0, \cdots, q)$ of the different powers of the frequency variable s are sums of products of the symbolic circuit elements. The above representation is a flat, expanded representation, as opposed to a nested representation, which evolves for example from a hierarchical analysis. This paper, however, addresses the symbolic analysis of analog building blocks such as complex operational amplifiers, which cannot efficiently be partitioned into loosely coupled sub-blocks, and which therefore cannot be analyzed efficiently with hierarchical techniques.

The number of terms in the expanded representation of a network function increases exponentially with the size of the circuit. Hence, for circuits of practical size it is virtually impossible to generate the exact expression. As a result, approximation or simplification of the symbolic expressions is required. Various symbolic approximation strategies have been proposed. These strategies can be classified according to the moment where the network function is approximated compared to the moment where the symbolic terms are generated. This is illustrated with Fig. 1.

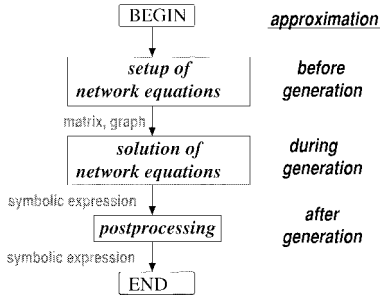

Fig. 1. Schematic representation of the flow of a symbolic analyzer with approximation capabilities.

Manuscript received November 30, 1997; revised April 19, 1998. This paper was recommended by Guest Editor M. M. Hassoun.

P. Wambacq was with ESAT-MICAS, Katholieke Universiteit Leuven, B-3001 Heverlee, Belgium. He is now with IMEC-VSDM, B-3001 Heverlee, Belgium.

P. Dobrovolný is with the Department of Microelectronics, Technical University Brno, 60200 Brno, Czech Republic.

G. G. E. Gielen is with ESAT-MICAS, Katholieke Universiteit Leuven, B-3001 Heverlee, Belgium.

W. Sansen is with ESAT-MICAS, Katholieke Universiteit Leuven, B-3001 Heverlee, Belgium.

Publisher Item Identifier S 1057-7130(98)07727-1.

This figure depicts the general flow of a symbolic analysis program. First, the network is read in, expanded, and then the network equations are set up. For an algebraic analysis method, such as the determinant expansion method used in the program ISAAC [3], this results in a matrix. For a topological method this setup results in one graph (for example for the signal-flow graph method [16]) or two graphs (for the two-graph method [16]). Then the network equations are solved with the chosen symbolic analysis technique. The resulting symbolic expressions can finally be post processed. At each of the three stages shown in Fig. 1, approximations can be made. In this way three approximation strategies can be distinguished: simplification *after* generation, *during* generation, and *before* generation.

The simplification after generation strategy has been the first approximation strategy that was used in modern symbolic analyzers [3], [4]. With this approach a network function is first generated completely, and then it is approximated afterwards by pruning the least important terms. For circuits of practical size that cannot be partitioned hierarchically in an efficient way, this technique is not useful, since it requires as an intermediate result the generation of the complete network function in expanded format. For example, for a circuit containing twenty transistors a network function can contain more than 10^{12} terms, which is a too large amount to generate.

A much more efficient strategy is the so-called simplification during generation [13], [15], [17]. With this technique, the dominant terms of every coefficient f_i or g_j in the network function [see (1)] are generated in decreasing order until the sum of generated terms is within the desired error bound of the exact numerical value of the coefficient under consideration. In this way only the desired dominant terms are generated. This technique has forced a breakthrough in the symbolic analysis of analog circuits of practical size. Implementations of this strategy are reported that can analyze symbolically circuits up to twenty transistors in reasonable CPU times of the order of hundreds of seconds or less [13], [15].

Another interesting technique is the so-called simplification before generation approach [11], [14], [15], in which the network is simplified prior to any symbolic computation. In fact the symbolic computations are performed on a simplified network, which is often much smaller than the original one. The different approaches can, of course, be combined: for example, in [15] the simplification before and during approach are coupled.

In this paper, a simplification during generation approach is presented that is much more efficient than previously presented approaches [13], [15], [18], with respect to both CPU time and memory usage. The approach presented here can again be coupled to a simplification before generation approach. This can be performed for example in the same way as described in [15].

Just as the simplification during generation approaches that have been described previously, the approach presented in this paper makes use of the two-graph method to generate the terms. This method is briefly reviewed in Section II. The underlying algorithm that generates the dominant terms with the two-graph method is a matroid intersection algorithm. The relationship between the two-graph method and matroid intersections is discussed in Section III. The use of matroid theory in this aspect has already been addressed several times with different approaches [18]–[20], which are briefly discussed in Section IV. In Section III, it will be explained that a valid term obtained with the two-graph method is a base that is common to three matroids. The enumeration in order of weight of bases common to three general matroids—corresponding to the enumeration of terms in decreasing order of magnitude—is an NP-complete problem [21]. In Section V, an approach is presented to efficiently enumerate the intersections that are common to the three specific matroids that correspond to the use of the two-graph method. The enumeration procedure, also denoted as ranking procedure, makes use of sensitivities of a network function with respect to changes in the coefficients f_i or g_j [see (1)]. Finally, experimental results obtained with this approach, are given in Section VI. Also, a comparison with other existing simplification during generation approaches is presented in that section. Conclusions are drawn in Section VII.

II. The Two-Graph Method

For the generation of symbolic terms different existing symbolic network analysis methods can be used. In [22] it has been pointed out that for the simplification during generation approach the two-graph method [16] is the most efficient one. In the two-graph method, a valid term is found as a spanning tree which is common to a weighted voltage graph and a weighted current graph. These two graphs have the same vertices and edges, but the edges are connected in a different way. The two graphs are constructed from an inspection of the circuit. The weight of each edge in these graphs is equal to the admittance of the corresponding element in the circuit. The absolute value of a term is equal to the product of the weights of the edges of a common spanning tree. The sign of the term is easily determined using topological information [23]. Hence, the two-graph method reduces to an enumeration of spanning trees that are common to two graphs. This enumeration procedure can be used to determine the dominant terms of both the numerator and the denominator, using the so-called sorting scheme [16].

The edges of the voltage or current graph can be divided in two classes: edges that correspond to conductances or transconductances on one hand, and edges that correspond to capacitors on the other hand. The weight of the latter edges contains the complex frequency symbol s since the admittance of a capacitor is of the form sC, whereas the weight of the (trans)conductance edges does not contain s. Hence, when the dominant terms of a given coefficient of power k are to be enumerated either in the numerator or denominator, then one must enumerate spanning trees that contain exactly k capacitor edges, while the rest of the edges corresponds to (trans)conductances. In graph theory it is common to assign a different color to edges belonging to different classes. Here we assign the red color to edges that correspond to (trans)conductances and the green color to capacitive edges. Using matroid theory, which is briefly reviewed in the next

section, it will be explained how to enumerate in a two-colored graph common spanning trees with k green edges while the rest of the edges are red.

III. Matroid Intersections and the Two-Graph Method

A *matroid* $M = (S, \mathcal{T})$ is a structure in which S is a finite set of *elements* and $\mathcal{T}$ is a family of subsets of S, such that the following axioms, called *independence axioms*, are satisfied [24]:

1) $\emptyset \in \mathcal{T}$.
2) All proper subsets of a set I in $\mathcal{T}$ are in $\mathcal{T}$.
3) If I_p and I_{p+1} are sets in $\mathcal{T}$ containing p and $p+1$ elements, respectively, then there exists an element $e \in I_{p+1} - I_p$ such that $I_p + e \in \mathcal{T}$.

An independent set of maximum cardinality is called a *base*. The notion of independence is made concrete for a specific matroid. A first specific matroid that is considered here is a *graphic matroid*. In this case, the set S is the set of edges of a graph. Here, an independent set is defined as a set of edges that do not contain a cycle. It is clear that a base in this matroid corresponds to a spanning tree. Another matroid considered here is a *partition matroid*, which is defined as follows. Let π be a partition which separates the finite set S into m disjoint blocks $B_1, B_2, \cdots, B_m$ and let d_i, $i = 1, 2, \cdots, m$ be m given nonnegative integers. Then for any S, π, and d_i $(i = 1, 2, \cdots, m)$ $M = (S, \mathcal{T})$ is a matroid, in which $\mathcal{T} = \{I | I \subseteq S, |I \cap B_i| \leq d_i, i = 1, 2, \cdots, m\}$. In other words, a set is independent if it does not contain more than d_i elements of partition B_i. Any matroid with such a structure is called a *partition matroid*.

It is possible to define more than one matroid on a set S. Also, it is possible that a subset of S is independent for more than one matroid. Such subset is denoted as a matroid intersection. When the cardinality of a base is the same for more than one matroid defined on a set S then a matroid intersection of maximum cardinality is a *base that is common to more than one matroid*.

For different symbolic network analysis methods the problem of simplification during generation can be formulated as the enumeration of bases that are common to three matroids. This is explained below for the two-graph method, but, as described in [20], this formalism can also be applied to other techniques to generate symbolic terms, such as the directed-tree enumeration method, the Coates flow graph method [25], and the parameter extraction method of Sannuti and Puri [26].

Let us now formulate the simplification during generation problem with the two-graph method in terms of matroid intersections. With the two-graph method two weighted graphs are considered, the voltage graph and the current graph of the circuit, that contain the same n vertices and m edges, but their topology is different. Each edge has a weight and a color, red or green. The red edges correspond to conductances or transconductances, the green edges to capacitors.

With this in mind, three matroids can be defined: a graphic matroid on the voltage graph, a graphic matroid on the current graph and a partition matroid induced by the coloring. For the latter matroid an independent set is defined as a set of edges that contains at most k green edges and at most $n-1-k$ red edges. Here k is an integer between 0 and $n-1$. A base of this partition matroid contains exactly k green edges and exactly $n-1-k$ red edges. Further, a base of any of the two graphic matroids corresponds to a spanning tree in the corresponding graph.

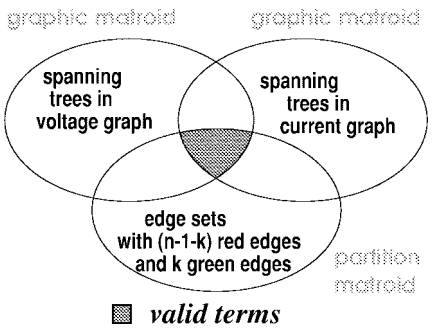

Fig. 2. A valid term of the coefficient of s^k in the numerator or denominator of a network function belongs to the intersection of the two graphic matroids with the partition matroid induced by the coloring.

Assume that—according to the simplification during generation strategy—the dominant terms of the coefficient of s^k in the numerator or denominator of a network function need to be determined. When these terms are determined with the two-graph method, then this reduces to the following problem: *enumerate in decreasing order of weight spanning trees that are common to both the voltage and the current graph and that contain exactly k green edges and $n-1-k$ red edges*. This problem can be translated to the following matroid intersection problem: *enumerate in decreasing order of weight bases that are common to the partition matroid induced by the coloring of the edges of the voltage or current graph, and to the two graphic matroids defined on the voltage graph and on the current graph, respectively*. As illustrated in Fig. 2, this problem is clearly an intersection problem in which three matroids are involved. In [21] it has been proven that for three general matroids this problem is NP-complete. An implementation of a polynomial-time algorithm that exploits the special structure of the matroids involved in our particular case has not been found either.

IV. Different Ranking Strategies

A general approach to rank the bases that are common to the three involved matroids is as follows. First, bases that are common to two matroids are enumerated in order. This problem can be solved in polynomial time with the algorithm of Camerini and Hamacher [27]. This algorithm can be used for the ranking of bases that are common to any two matroids. It runs in $O(K \cdot m \cdot R \cdot c(m))$ time, in which K is the number of enumerated bases, m is the number of elements in the matroid, R is the cardinality of the bases, also denoted as the *rank* of the matroids, and $c(m)$ is the maximum of the running time limits of the independence tests in each of the two matroids.

When an intersection is found that is common to two matroids, it is checked whether this intersection is also independent with respect to the third matroid. If not, the solution is rejected. If yes, a good base is found. With this general

strategy in mind, three options exist for the simplification during generation problem with the two-graph method:

1) Enumeration of the best intersections of the graphic matroid on the voltage graph with the partition matroid induced by the coloring. Next, it is checked whether a generated intersection contains a cycle in the current graph. If not, then a valid term is found and it is stored, after which the enumeration continues. Else, no valid term is found and the enumeration continues.
2) The same as Option 1 but with the role of the voltage and the current graph interchanged.
3) Enumeration of intersections of the two graphic matroids. This means that common spanning trees are first enumerated. Next it is checked whether the color constraint is satisfied.

The three options are now investigated in further detail.

A. Option 1

Option 1 has been implemented in [13] and [19]. Due to the simple structure of the partition matroid the running time limit of the algorithm that enumerates the intersections that are common to the graphic matroid and the partition matroid can be made lower than the running limit of the general algorithm of Camerini and Hamacher. For the two specific matroids involved in this case, the most efficient algorithm reported up until now [19] runs in $O(K_a m)$ time with $O(K_a m)$ space requirements. Here K_a is the number of generated spanning trees in the voltage graph and m is the number of edges in the voltage or current graph. This procedure is repeated for each coefficient of each power of s in the numerator or denominator of the network function. Because the total number of powers of s in the complete network function is $O(n)$, the total procedure runs in $O(K_a mn)$ time in which K_a now corresponds to the average value taken over the different powers of s, of spanning trees in the voltage graph that must be generated in order to obtain sufficient terms for the approximation of the coefficient f_i or g_j in (1). The disadvantage of this approach is that for very large transistor circuits many spanning trees of the voltage graph may have to be generated to retain only a very small number of common spanning trees. Indeed, it has been observed with this approach that the ratio of common spanning trees divided by the number of generated spanning trees in the voltage graph decreases as the circuit size increases. This will be illustrated in Section VI. Hence K_a can be much larger than the number of terms that need to be generated for f_i or g_j. Nevertheless, this approach gives good results for the symbolic analysis of circuits up to twenty transistors [13], but for larger two-graphs CPU time and memory requirements increase sharply. This will also be evidenced in Section VI.

B. Option 2

With the second option the role of the voltage and the current graph is interchanged compared to the first option. In [19] it has been shown that this option is less advantageous as the previous option, since the number of spanning trees in the current graph is usually higher than the number of spanning trees in the voltage graph.

C. Option 3

With the third option spanning trees that are common to the voltage graph and the current graph are enumerated in decreasing order of magnitude. Abstraction is made of the coloring of the edges. Hence, it must be checked afterwards, whether a common spanning tree that has been found, also contains k green edges. This approach is implemented in [18] in the following way. First, $s = j\omega$ is given a numerical value. By giving s a specific value it is possible to compare the absolute value of the admittance of a capacitor which is of the form $j\omega C$, to the value of a (trans)conductance. For the given value of $j\omega$, common spanning trees are generated. Clearly, every common spanning tree corresponds to a valid term, but it is not known in advance how many capacitors a term contains, and hence it is not known in advance to which coefficient in the network function the generated terms will belong. Hence, the terms must be classified afterwards by the number of capacitors. This procedure has to be repeated for different values of the frequency and duplicate terms that are generated at multiple frequency points have to be eliminated. In this way an approximate symbolic expression can be constructed that is valid over the frequency range between the lowest and the highest frequency that has been considered. When the terms are classified by the powers of s, then the complete approximate expression contains an approximation for each coefficient f_i or g_j that plays a role in the considered frequency band. In the implementation of this approach described in [18] the frequency points are selected by the user.

The algorithm that is used for the enumeration of common spanning trees in [18] is the general ranking algorithm of Camerini and Hamacher that has been mentioned above [27], but now applied to two graphic matroids. The running time limit for the enumeration procedure at one single frequency point reduces to $O(K_c mn^2)$, in which K_c is the number of generated intersections, m is the number of edges in each graph and n is the number of vertices in each graph. This enumeration procedure is repeated at all frequency points. As a result, the running time limit of the overall approach is $O(n_g K_c mn^2)$, in which n_g is the number of chosen frequency points and K_c is the mean value of the total number of enumerated intersections at every frequency point. It should be noted that not every generated intersection is a base: some generated intersections have a lower cardinality. However, the ratio of the total number of generated intersections to the number of bases only grows slowly with the circuit size. This will be evidenced in Section VI for practical circuits.

This option has shown to be more efficient than the approach of Option 1 for the generation of approximate expressions of the coefficients of the lowest powers of s [18]. Therefore, this option has been chosen as the starting point for the approach presented in this paper.

However, this approach has an important disadvantage. When frequency points are close to each other, then it is very likely that a term that has been generated at the previous frequency, will also be generated at the next frequency. However, in order to obtain a reasonable accuracy in the frequency range of interest, it is required to choose the

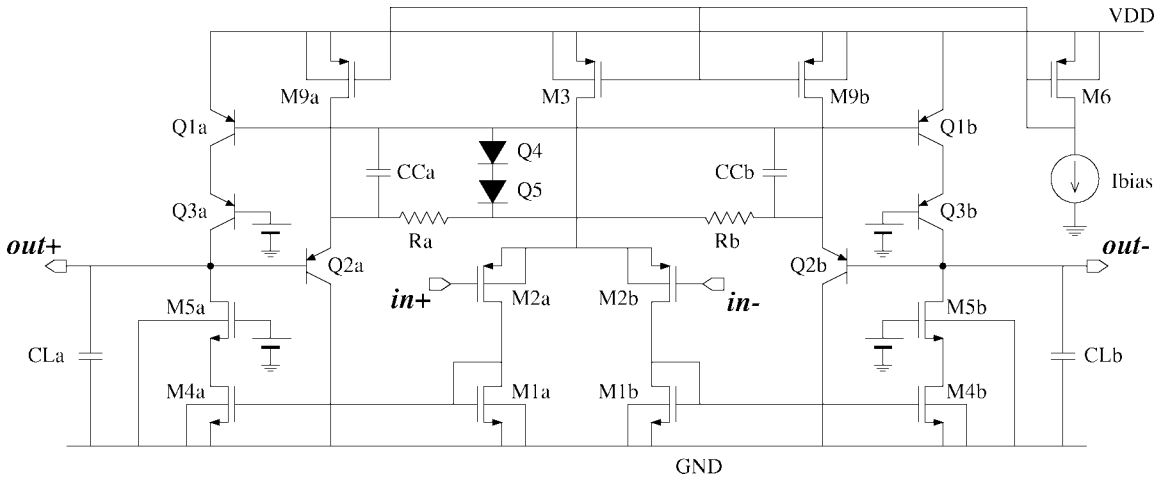

Fig. 3. A fully differential BiCMOS OTA.

frequency points close to each other. Assume, for example, that a symbolic expression for the transfer function of an operational amplifier is to be generated. The low-frequency response is quite smooth and does not require many frequency points for this approach [18]. However, in order to be able to extract from the symbolic network function an expression for the phase margin, a good accuracy is required in the vicinity of the unity-gain frequency. There the phase of the transfer function is not as smooth compared to the phase at low frequencies. As a result, a very fine grid of frequency points is required at high frequencies in order to have an acceptable accuracy. This implies that many terms will be generated more than once. In this way, the efficiency of the approach goes down.

V. Sensitivity-Based Ranking of Intersections

The approach that is proposed in this paper starts from Option 3, discussed in the previous section. The efficiency of that approach is enhanced by an intelligent choice of the frequency points. This choice is based on the sensitivities of the magnitude of the network function with respect to a coefficient f_i or g_j from (1). The normalized sensitivity of a parameter F, in this case the magnitude of the network function, with respect to coefficient h, which can be any coefficient f_i or g_j, is defined as:

$$S_h^F = \frac{\partial F}{\partial h}\,\frac{h}{F} = \frac{\partial \ln F}{\partial \ln h}. \tag{2}$$

These sensitivities can be calculated numerically as a function of frequency. For every coefficient h there is a frequency at which the absolute value of the sensitivity of the magnitude with respect to that coefficient is maximal. This is illustrated for the circuit shown in Fig. 3, which is a fully differential BiCMOS amplifier. The network function that is considered here is the differential-mode gain of this amplifier as a function of frequency. This network function is a ratio of two polynomials in s, the numerator polynomial having degree thirteen, while the denominator polynomial has a degree of fifteen. The sensitivities of the magnitude of the network function with respect to the coefficients g_j $(j = 0, 1, \cdots, 15)$ of the denominator of the network function are displayed in Fig. 4.

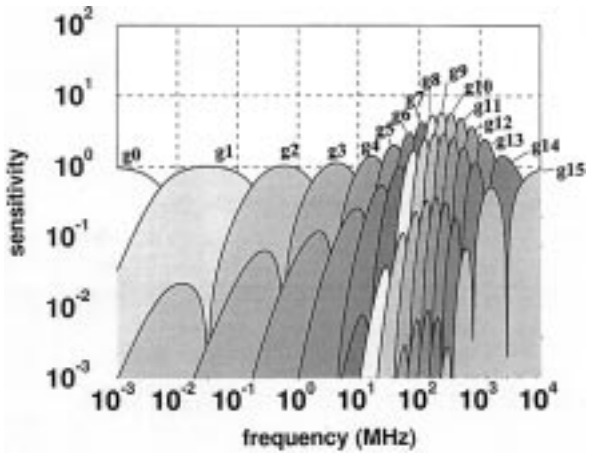

Fig. 4. Absolute value as a function of frequency of the normalized sensitivities of the magnitude of the differential-mode gain of the BiCMOS OTA with respect to the coefficients g_i $(i = 0, \cdots, 15)$ of the denominator.

In this figure the maxima of the sensitivities with respect to the different coefficients g_j can be clearly distinguished. If for a given coefficient the frequency at which this maximum occurs, is selected as the frequency at which the common spanning trees will be generated, then it is very likely that the generated dominant terms will belong to this coefficient. In this way, the need for a fine frequency grid in frequency ranges where the network function varies rapidly, is not required anymore. Instead, the number of frequency points is related to the order of the network function. As will be illustrated with the experimental results in the next section, this approach has proven to yield very little overhead in the sense that at a frequency that corresponds to a maximal sensitivity with respect to a given coefficient, the amount of generated terms that belong to another coefficient, is very low. This is also true when the frequencies of maximum sensitivity are close to each other, which is the case for example with the coefficients g_6 to g_{12} in Fig. 4. Finally, it has to be remarked that the CPU time for the determination of the frequencies of maximum sensitivity, is negligible compared to the CPU time required to generate the symbolic expressions.

The selection of the frequency points based on sensitivities of the magnitude is just a way to avoid as much as possible

that duplicate terms are generated at adjacent frequency points, while a high accuracy still can be obtained. However, this procedure needs to be controlled by a so-called error-control mechanism in order to monitor the accuracy of the magnitude and the phase of the overall network function. To this purpose, the error-control mechanism described in [28] is used. With this mechanism an initial approximation of each individual coefficient f_i or g_j from (1) needs to be generated first. Whereas the error on each coefficient individually may already be relatively low, the error on the magnitude or phase of the complete network function may be very high at some frequencies. Therefore, after the individual approximation of the coefficients, the error-control mechanism determines for which coefficients additional terms need to be generated in order to obtain a user-specified accuracy on the magnitude and phase of the network function in the given frequency range. To this purpose, the error-control mechanism makes use of sensitivities of both the magnitude and phase of the network function with respect to the different coefficients $f_i(i = 0, \cdots, p)$ and $g_j(j = 0, \cdots, q)$. More details are found in [28].

The overall algorithm now runs as follows. First, the numerical value of each coefficient is computed with the polynomial interpolation method [29]. For very large circuits, a simple application of the polynomial interpolation method can yield overflow problems or it can lead to serious errors on the coefficients. Therefore, extra precautions need to be taken as described in [30]. Next, the sensitivity of the magnitude of the network function with respect to each coefficient is computed and the frequencies are determined where the maximal (absolute) values occur. Then the algorithm of Camerini and Hamacher is applied at each of these frequencies to enumerate terms (common spanning trees of the voltage and current graph) in decreasing order. This enumeration is controlled by the error-control mechanism of [28], as described above.

The running time limit of this approach can be determined as follows. Assume that the degrees of the polynomial in the numerator and the denominator of a network function are p and q, respectively. This means that common spanning trees must be generated at $p + q + 2$ different frequencies. The enumeration of K_i common spanning trees at one frequency ω_i is performed with the algorithm of Camerini and Hamacher in $O(K_i mn^2)$ time (see above). Since $p + q + 2 = O(n)$, the running time limit of the overall approach is $O(Kmn^3)$, where K is the average number of intersections (common trees) that is generated for the approximation of each coefficient f_i or g_j. This running time limit is smaller than the running time limit $O(n_g K_c mn^2)$ of the approach described under Option 3 [18], since usually $nK \ll n_g K_c$. This will be demonstrated experimentally in the next section.

The memory usage of the new approach is $O(Kmn)$. This follows from the memory usage of the algorithm of Camerini and Hamacher.

VI. Experimental Results

Example 1: Fig. 5 depicts a BiCMOS Miller-compensated operational transconductance amplifier (OTA). This circuit has been designed in a high-voltage BiCMOS process. The low-frequency gain is 100 dB, the phase margin is 50° and the gain-bandwidth product is 5.5 MHz. The small-signal parameters that correspond to this design point have been used to obtain an approximate symbolic expression of the differential-mode gain as a function of frequency.

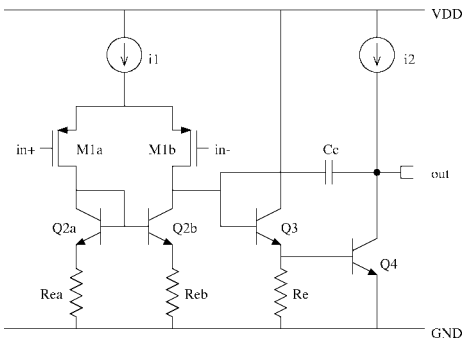

Fig. 5. A Miller-compensated BiCMOS OTA.

For a comparison between the different approaches one can concentrate on the generation of an approximate expression of the denominator only. The number of terms in the numerator is much lower.

The exact expression of the denominator contains 195 948 terms. The degree of the polynomial in the denominator is 8. Hence the approach presented in this paper has to perform the enumeration procedure of common trees nine times. If an approximate expression of the denominator is required with an error of 5% on the magnitude and a phase error of one degree in the frequency band from DC to 50 MHz (which is about ten times the gain-bandwidth) then 159 terms are required for the given numerical values. If this accuracy needs to be obtained with the approach [18] discussed in Section IV-C, then 74 frequency points are required at which common spanning trees need to be generated. The value of 74 is obtained as follows: first a very coarse grid is taken, namely 5 equally spaced frequency points between 1 Hz and 5.5 MHz, which is the frequency band of interest. Then common spanning trees are generated at those frequency points. If a numerical evaluation of the resulting symbolic expression gives a too large error on the magnitude or phase for a frequency in between two adjacent frequency points, then an extra frequency point is considered in this frequency interval and common spanning trees are generated at this new frequency point. In this way, one ends up with a non uniform frequency grid and the grid is not needlessly high in frequency regions where the network function does not change much. In spite of this efficient choice of the frequency grid, the number of required frequency points (74) is about nine times more than with the approach described in Section V. When the 159 terms are generated with this approach, then a CPU time of 1.48 s is required on a 712/100 HP workstation. The statistics of this example are summarized in Table I.

In this table the second column indicates how many intersections (common trees, not only common spanning trees) are generated at the frequency at which the sensitivity of the magnitude of the network with respect to the coefficient of the power under consideration (first column), is maximal.

TABLE I
STATISTICS OF THE SYMBOLIC APPROXIMATION OF THE DENOMINATOR OF THE NETWORK FUNCTION THAT CORRESPONDS TO THE DIFFERENTIAL-MODE GAIN OF THE BiCMOS MILLER-COMPENSATED AMPLIFIER OF FIG. 5

power of s	# intersections	# terms (any power of s)	# terms (correct power of s)	ratio (overhead)
0	7	4	4	1.00
1	5	3	3	1.00
2	23	12	7	1.72
3	55	28	12	2.33
4	107	54	23	2.33
5	187	94	39	2.44
6	199	100	42	2.38
7	101	51	23	2.22
8	11	6	6	1.00

The third column indicates how many terms, i.e., common spanning trees, are generated. As already mentioned above, not only terms are generated that belong to the power of s we are focusing on, but also terms that belong to other powers of s. The fourth column indicates how many of these terms really belong to the power we are looking at. The ratio of the data in the third column to the data in the fourth column can be seen as the generated overhead. This ratio is given in the fifth column.

For comparison purposes, the same 159 terms that have been generated with the new approach presented in this paper, have also been generated with the technique explained in Option 1, that has been implemented as described in [13]. With this technique, spanning trees in the voltage graph are generated with the correct number of capacitive edges, after which it is tested whether the same edges constitute a spanning tree in the current graph as well. It has been found that for the generation of the 159 terms, 749 spanning trees in the voltage graph need to be generated, compared to 695 intersections in our approach. The CPU time is 0.67 s on a 712/100 HP workstation. Clearly, this circuit is small enough such that it is not advantageous yet to use the new approach described in this paper, despite its lower computational complexity behavior.

The same approach can be followed for the approximation of the numerator. With the same constraints on the accuracy, namely a maximum relative error of 5% on the magnitude and a phase error of one degree in the frequency band from DC to 50 MHz, the number of terms that is generated for the numerator is 89. The final error on the magnitude and the phase as a function of frequency of the complete transfer function is shown in Fig. 6. It is seen that between DC and 50 MHz the required accuracy is achieved.

Example 2: Next, a larger example is given that illustrates the advantages of the new approach. A symbolic expression is generated for the network function that corresponds to the differential-mode gain of the fully-differential BiCMOS amplifier of Fig. 3. For this example we can restrict again to an analysis of the denominator, without loss of generality. The denominator is a polynomial of degree fifteen. The voltage and the current graph that are set up for the computation of the denominator contains 16 vertices and 70 edges, 25 of which corresponding to capacitors, while the others correspond to (trans)conductances. The total number of terms of the complete

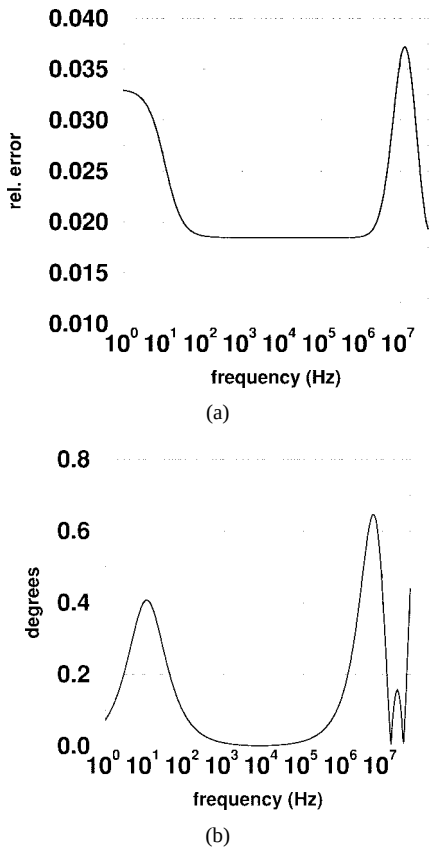

Fig. 6. Relative error on the magnitude (top) and absolute error in degrees on the phase (bottom) for the symbolic expression of the differential-mode gain of the BiCMOS Miller-compensated amplifier.

exact network function is about 10^{11}. This number can be estimated using graph theory [22].

The number of terms that have to be generated in order to obtain a reasonable accuracy, depends on the values of the small-signal parameters. With values that correspond to a realistic design point, the number of terms that has to be generated is usually much lower than if one would use rough estimates for the small-signal parameters that might not correspond to a realistic design point. In [13] an approximation of the same network function considered here has been generated, using the technique of Option 1 and with numerical values corresponding to a real design point. However, with other numerical values for the small-signal parameters, originating for example from initial guesses before the circuit effectively has been designed, this technique fails to generate approximations for all powers of s. The number of spanning trees of the voltage graph that have to be generated and stored in memory is so large, that only the first four powers of s (corresponding to five coefficients) could be generated. The CPU time required for these first four powers amounts to 1258 s on a 712/100 HP workstation. Table II lists a summary of this experiment. The total number of terms that constitute an approximation for the first five coefficients of the denominator of the network function is 4223. In order to generate these terms more than 1.1 million spanning trees of the voltage graph need to be generated. This means that on the average 269 spanning trees need to be generated for one valid term.

TABLE II
STATISTICS OF THE GENERATION OF APPROXIMATIONS OF THE FIRST FOUR COEFFICIENTS OF THE DENOMINATOR OF THE NETWORK FUNCTION OF THE FULLY DIFFERENTIAL BiCMOS OTA WITH THE TECHNIQUE DESCRIBED IN [13] (SEE SECTION IV-C)

power of s	# spanning trees in voltage graph	# terms
0	49174	25
1	51933	25
2	166402	195
3	366490	1013
4	503279	2965

TABLE III
STATISTICS OF THE SYMBOLIC APPROXIMATION OF THE DENOMINATOR OF THE NETWORK FUNCTION THAT CORRESPONDS TO THE DIFFERENTIAL-MODE GAIN OF THE FULLY DIFFERENTIAL BiCMOS AMPLIFIER OF FIG. 3

power of s	# intersections	# terms (any power of s)	# terms (correct power of s)	ratio (overhead)
0	49	25	25	1.00
1	49	25	25	1.00
2	497	249	195	1.28
3	3873	1937	1013	1.92
4	15825	7913	2965	2.70
5	35437	17719	5289	3.33
6	37801	18901	5899	3.23
7	23809	11905	4163	2.86
8	9941	4971	1971	2.50
9	5311	2656	947	2.78
10	2017	1009	593	1.69
11	8585	4293	1811	2.38
12	19455	9728	3367	2.86
13	19253	9631	3326	2.86
14	7031	3516	1607	2.17
15	585	293	293	1.00

Next, the network function is approximated with the new approach. The statistics of this example are given in Table III. The meaning of the data is the same as in Table I of the previous example.

Not only is it seen that with the new approach an approximation for all coefficients can be generated, the total CPU time of the complete simulation is 513 s, which is still lower than the CPU time required with the technique of Option 1 to generate an approximation for the first four powers only. The CPU time for the generation of approximate coefficients for the first four powers of s is 12.8 s, which is about $100\times$ smaller than with the approach of Option 1. Further, the rightmost column in Table III reveals that the choice of the frequency points that corresponds to a maximum sensitivity with respect to a coefficient g_j from (1) $(j = 0 \cdots 15)$ yields a small overhead: in the worst case, which is seen to correspond to power 6, every term of s^6 requires the generation of 3.33 terms that belong to other powers.

An even more important issue is the much lower memory usage with the new approach. This is best measured by counting the total number of matroid intersections that has been generated, since all generated intersections need to be stored for an enumeration in decreasing order. The total number of different terms that has been generated is 32 245 whereas 189 518 intersections have been generated and stored. This means that on the average 5.88 intersections need to be generated for every different valid term. Compared to the previous example, where on the average 4.37 intersections need to be generated for one valid term, it is seen that this overhead is slowly growing.

VII. CONCLUSIONS

In this paper, a new approach has been presented for the simplification during generation technique, which is known to be a very efficient technique for the generation of approximate symbolic network functions for large analog circuits. The symbolic analysis method to generate the terms is the two-graph method in which a valid symbolic term corresponds to a spanning tree that is common to the voltage graph and the current graph. The common spanning trees are generated in decreasing order, which corresponds to the generation of the dominant symbolic terms only. The algorithm for the generation of common spanning trees is a matroid intersection algorithm that is applied for different values of the frequency variable s. By an intelligent choice of the frequencies at which common spanning trees are generated, namely at the frequencies where the sensitivities of the transfer function magnitude to the individual coefficients are maximum, the running time limit of the overall approach is $O(Kmn^3)$, where m is the number of edges in the voltage or current graph, n is the number of vertices in each graph and K is the total number of generated matroid intersections. K is higher than the total number of different valid terms, such that the approach—like all other approaches—yields some overhead. However, experimental results presented on practical circuits have shown that the overhead, both in terms of memory usage and CPU time, is slowly growing with the circuit size and for large circuits this overhead is much lower than previously reported approaches.

The approach presented in this paper can be coupled now with a simplification before generation strategy, such as presented for example in [15], and with post-processing routines, such as the ones presented in [31], leading to a powerful symbolic simulator for complex large analog building blocks.

REFERENCES

[1] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits.* Dordrecht, The Netherlands: Kluwer Academic, 1991.
[2] G. Gielen, P. Wambacq, and W. Sansen, "Symbolic analysis methods and applications for analog circuits: A tutorial overview," *Proc. IEEE*, vol. 82, pp. 287–304, Feb. 1994.
[3] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: A symbolic simulator for analog integrated circuits," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1587–1597, Dec. 1989.
[4] F. V. Fernández, A. Rodríguez-Vázquez, and J. L. Huertas, "Interactive ac modeling and characterization of analog circuits via symbolic analysis," *J. Analog Integrated Circuits Signal Processing*, vol. 1, pp. 183–208, Nov. 1991.
[5] S. Seda, M. Degrauwe, and W. Fichtner, "Lazy-expansion symbolic expression approximation in SYNAP," in *Tech. Dig. IEEE/ACM Int. Conf. Computer-Aided Design*, 1992, pp. 310–317.
[6] S. Manetti, "New approaches to automatic symbolic analysis of electric circuits," *Proc. Inst. Elect. Eng.*, pt. G, pp. 22–28, Feb. 1991.
[7] G. Wierzba *et al.*, "SSPICE—A symbolic SPICE program for linear active circuits," in *Proc. Midwest Symp. on Circuits and Systems*, 1989.
[8] A. Konczykowska and M. Bon, "Automated design software for switched-capacitor IC's with symbolic simulator SCYMBAL," in *Proc. ACM/IEEE Design Automation Conf.*, 1988, pp. 363–368.
[9] M. Hassoun and P. M. Lin, "A new network approach to symbolic simulation of large-scale networks," in *Proc. Int. Symp. on Circuits and Systems*, 1989, pp. 806–809.

[10] L. Huelsman, "Personal computer symbolic analysis programs for undergraduate engineering course," in *Proc. Int. Symp. on Circuits and Systems*, 1989, pp. 798–801.
[11] R. Sommer, E. Hennig, G. Dröge, and E. H. Horneber, "Equation-based symbolic approximation by matrix reduction with quantitative error prediction," *Alta Freq.—Rivista Elettron.*, vol. 5, no. 6, pp. 29–37, Nov. 1993.
[12] G. Nebel, U. Kleine, and H. J. Pfleiderer, "Symbolic pole/zero calculation using SANTAFE," *IEEE J. Solid-State Circuits*, vol. SC-30, pp. 752–761, July 1995.
[13] P. Wambacq, F. Fernández, G. Gielen, W. Sansen, and A. Rodríguez-Vázquez, "Efficient symbolic computation of approximated small-signal characteristics," *IEEE J. Solid-State Circuits*, vol. 30, pp. 327–330, Mar. 1995.
[14] J. J. Hsu and C. Sechen, "DC small signal symbolic analysis of large analog integrated circuits," *IEEE Trans. Circuits Syst. I*, vol. 41, pp. 817–828, Dec. 1994.
[15] Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits," *IEEE Trans. Circuits Syst. I* vol. 43, pp. 656–669, Aug. 1996.
[16] P. M. Lin, *Symbolic Network Analysis*. Amsterdam, The Netherlands: Elsevier, 1991.
[17] P. Wambacq, G. Gielen, and W. Sansen, "A cancellation-free algorithm for the symbolic analysis of large analog circuits," in *Proc. Int. Symposium on Circuits and Systems*, 1992, pp. 1157–1160.
[18] Q. Yu, "Approximate symbolic analysis of large analog integrated circuits," Ph.D. dissertation, Univ. of Washington, 1995.
[19] P. Wambacq, "Symbolic analysis of large and weakly nonlinear analog integrated circuits," Ph.D. dissertation, Katholieke Univ. Leuven, Belgium, 1996.
[20] P. Wambacq, F. Fernández, G. Gielen, W. Sansen, and A. Rodríguez-Vázquez, "A family of matroid intersection algorithms for the computation of approximated symbolic network functions," in *Proc. Int. Symp. on Circuits and Systems*, 1996, pp. 806–809.
[21] M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[22] P. Wambacq, G. Gielen, and W. Sansen, "Symbolic network analysis methods for practical analog ICs: A survey," *IEEE Trans. Circuits Syst. II*, vol. 45, pp.1331–1341, this issue.
[23] S. P. Chan, *Introductory Topological Analysis of Electrical Networks*. New York: Holt, Rinehart and Winston, 1969.
[24] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.
[25] C. L. Coates, "Flow-graph solutions of linear algebraic equations," *IRE Trans. Circuit Theory*, vol. CT-6, pp. 170–187, June 1959.
[26] P. Sannuti and N. N. Puri, "Symbolic network analysis—An algebraic formulation," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 679–687, Aug. 1980.
[27] P. M. Camerini and H. W. Hamacher, "Intersection of two matroids: (Condensed) border graphs and ranking," *SIAM J. Disc. Math.*, vol. 2, no. 1, pp. 16–27, Feb. 1989.
[28] P. Wambacq, G. Gielen, and W. Sansen, "Symbolic analysis of large analog circuits using a sensitivity-driven enumeration of common spanning trees," in *Proc. Int. Symp. on Circuits and Systems*, 1996, pp. 809–812.
[29] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York: Van Nostrand Reinhold, 1983.
[30] I. García-Vargas, M. Galán, F. Fernández, and A. Rodríguez-Vázquez, "An algorithm for numerical reference generation in symbolic analysis of large analog circuits," in *Proc. Europ. Design & Test Conf.*, 1997, pp. 395–399.
[31] P. Wambacq, M. Topa, G. Gielen, and W. Sansen, "Factorization of approximate symbolic network functions of complex analog integrated building blocks," in *Proc. Europ. Conf. Circuit Theory and Design*, 1997, pp. 1464–1467.

Piet Wambacq, for photograph and biography, see this issue, p. 1341.

Petr Dobrovolný, photograph and biography not available at the time of publication.

Georges G. E. Gielen, photograph and biography not available at the time of publication.

Willy Sansen (S'66–M'72–SM'86–F'95), for a biography, see this issue, p. 1341.