Interpretable Symbolic Small-Signal Characterization of Large Analog Circuits using Determinant Decision Diagrams [*]

Xiang-Dong Tan and C.-J. Richard Shi
Department of Electrical Engineering
University of Washington
Seattle, WA 98195, USA

Abstract: *A new approach is proposed to generate interpretable symbolic expressions of small-signal characteristics for large analog circuits. The approach is based on a complete, exact, yet compact representation of symbolic expressions via determinant decision diagrams (DDDs). We show that two key tasks of generating interpretable symbolic expressions — term de-cancellation and term simplification—can be performed in linear time in terms of the number of DDD vertices. With the number of DDD vertices many-orders-of-magnitude less than the number of product terms, the proposed approach has been shown to be much more efficient than other start-of-the-art approaches.*

1. Introduction

Mixed-signal (analog and digital) systems are becoming increasingly important. While automation tools exist for digital circuits, analog design is still done manually and depends heavily on designers' experience. In this paper, we present a new approach to generate interpretable analytic expressions for small-signal characteristics for typical analog building blocks.

Previous attempts to generate interpretable expressions are based on various symbolic analysis methods to generate sum-of-product representations for network functions. This area has been studied extensively in 1960s-1980s [7]. However, the resulting approaches are only feasible for very small circuits, since the number of expanded product terms grows exponentially with the size of a circuit, and resulting expressions become not interpretable by analog designers. Recently, various approximation schemes have been developed. Approximation after generation is reliable but it requires the expansion of product terms first [5, 11, 16]. Some improvement techniques based on nested expressions have been proposed [2, 12]. But they generally suffer symbolic term-cancellation and align-term problems. Approximation during generation extracts only significant product terms [3, 15, 17]. It is very fast, but has two major deficiencies: First, if accurate expressions are needed, then the complexity of the approach becomes exponential. Second, it works only for transfer functions. Other small-signal characteristics such as sensitivities, symbolic poles and zeros, cannot be extracted in general. Approximation before generations [6, 17] has also been proposed.

In this paper, we show that both term de-cancellation and dominate term generation can be performed elegantly in a new framework based on Determinant Decision Diagrams (DDDs) [9, 10]. Section 2 reviews the concepts of DDDs and s-expanded DDDs. Section 3 presents a general DDD-based framework for deriving interpretable symbolic expressions. Experimental results are described in Section 4. Section 5 concludes the paper.

[*]This work was sponsored by U.S. Defense Advanced Research Projects Agency (DARPA) under grant number F33615-96-1-5601 from the United States Air Force, Wright Laboratory, Manufacturing Technology Directorate, and by Rockwell Semiconductor Systems.

2. Determinant Decision Diagrams

2.1. Concept of DDDs

Our approach is based on a newly-introduced graph representation of symbolic matrix determinants called *Determinant Decision Diagrams* (DDDs) [9]. Consider a simple RC filter circuit shown in Figure 1. Its system equations can be written as

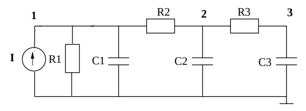


Figure 1: An example circuit.

$$\begin{bmatrix} \frac{1}{R_1} + sC_1 + \frac{1}{R_2} & -\frac{1}{R_2} & 0 \\ -\frac{1}{R_2} & \frac{1}{R_2} + sC_2 + \frac{1}{R_3} & -\frac{1}{R_3} \\ 0 & -\frac{1}{R_3} & \frac{1}{R_3} + sC_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$$

We view each entry in the circuit matrix as one distinct symbol, and rewrite its system determinant in the left-hand side of Figure 2. Then its DDD representation is shown in the right-hand side. A DDD is a signed, rooted, directed acyclic graph with two terminal vertices, namely the *0-terminal* vertex and the *1-terminal* vertex. Each non-terminal vertex is labeled by a symbolic symbol denoted by a_i, and a positive or negative sign, denoted by $s(a_i)$. It *originates* two outgoing edges, called *1-edge* and *0-edge*. Each vertex a_i represents a symbolic expression $D(a_i)$ defined recursively as follows:

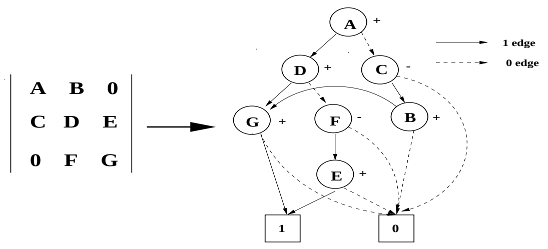


Figure 2: A matrix determinant and its DDD.

1. if a_i is the 1-terminal vertex, then $D(a_i) = 1$,
2. if a_i is the 0-terminal vertex, then $D(a_i) = 0$,
3. if a_i is a non-terminal vertex, then $D(a_i) = a_i\, s(a_i)\, D_{a_i} + D_{\overline{a}_i}$.

where D_{a_i} and $D_{\overline{a}_i}$ represent, respectively, symbolic expressions represented by the vertices pointed by the 1-edge and 0-edge of a_i.

A *1-path* is a path from the root vertex (A in our example) to the 1-terminal. A 1-path defines a product of symbolic symbols and signs of the vertices that originate all the 1-edges along the 1-path. In our example, there exist three product terms: ADG, $-AFE$ and $-CBG$. The root vertex represents the sum of these product terms and therefore the determinant.

2.2. s-Expanded DDDs

To exploit the DDD to derive interpretable small-signal characterizations, we need to directly represent circuit parameters not matrix entries. To this end, s-expended DDDs [10] can be used.

Consider the circuit in Figure 1 and its system determinant. Let us introduce a unique symbol for each circuit parameter in its admittance form. Specifically, we introduce $a = \frac{1}{R_1}$, $b = f = \frac{1}{R_2}$, $d = e = -\frac{1}{R_2}$ $g = k = \frac{1}{R_3}$, $i = j = -\frac{1}{R_3}$, $C_1 = c, h = C_2, l = C_3$. Then the circuit matrix can be rewritten as

$$\begin{bmatrix} a+b+cs & d & 0 \\ e & f+g+hs & i \\ 0 & j & k+ls \end{bmatrix}$$

The original 3 product terms will be expanded to 23 product terms in different powers of s:

$$(a+b+cs)(f+g+hs)(k+ls) \rightarrow \left\{ \begin{array}{ll} +afks^0 & +bgls^1 \\ +agks^0 & +ahks^1 \\ +bfks^0 & +bhks^1 \\ +bgks^0 & +ahls^2 \\ +cfks^1 & +bhls^2 \\ +cgks^1 & +chks^2 \\ +afls^1 & +cfls^2 \\ +agls^1 & +cgls^2 \\ +bfls^1 & +chls^3 \end{array} \right.$$

$$(a+b+cs)(-j)(i) \rightarrow \left\{ \begin{array}{l} -ajis^0 \\ -bjis^0 \\ -cjis^1 \end{array} \right.$$

$$(-e)(d)(k+ls) \rightarrow \left\{ \begin{array}{l} -edks^0 \\ -edls^1 \end{array} \right.$$

We can represent these product terms nicely using a slight extension of the original DDD, as shown in Figure 3. This DDD has exactly the same properties as the original DDD except that there are four roots representing coefficients of s^0, s^1, s^2, s^3. Each DDD root represents a symbolic expression of a coefficient in the corresponding s polynomial. Each such DDD is called a *coefficient* DDD, and the resulting DDD is called a *multi-root* DDD. The original DDD in which s is contained in some vertices is called *complex* DDD. The s-expanded DDD can be constructed from the complex DDD in a very efficient way [10].

3. A Framework for Interpretable Small-Signal Characterization

A linear(ized) analog circuit can be described by a set of linear equations in the following general form using the modified nodal analysis (MNA) approach [14]:

$$\mathbf{T}\mathbf{x} = \mathbf{w}, \tag{1}$$

where the *circuit unknown vector* **x** may be composed of node voltages and branch currents, and the *circuit matrix* **T** is a large sparse symbolic matrix.

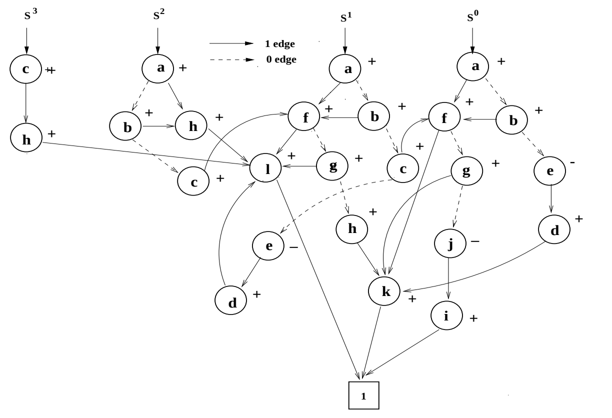


Figure 3: An s-expanded DDD.

We consider small-signal characterization as finding interpretable network functions, sensitivity expressions, symbolic pole and zero expressions. According to Cramer's rule, the kth component x_k of the unknown vector **x** is obtained as follows:

$$x_k = \frac{\sum_{i=1}^{n} w_i \, (-1)^{i+k} \, det(\mathbf{T}_{t_{i,k}})}{det(\mathbf{T})}. \tag{2}$$

Note that the numerator and the denominator are only composed of the determinant and cofactors of the circuit matrix, which can be represented effectively using DDDs. With this, interpretable small-signal characterization can be performed by DDD manipulation:

- A network function is defined as the ratio of an output unknown from **x** over an input from **w**. This can be represented as the ratio of two complex DDDs, or two s-expanded DDDs.
- Under the assumption that poles are far away from each other, the root splitting method can be applied to find symbolic expressions for poles and zeros [4]. In this case, a pole or zero can be represented as the ratio of two consecutive coefficient DDDs.
- Sensitivity of a DDD with respect to a circuit parameter amounts to finding a DDD cofactor. Sensitivities of general small-signal characteristics with respect to circuit parameters can be represented as expressions of DDDs and their cofactors.

After we obtain the exact DDD representations of small-signal characteristics, the key task is how to simplify a DDD or a ratio of two DDDs.

It turns out that DDD-based simplification can be easily performed by means of efficient DDD manipulations. Moreover, since we begin with the exact and compact DDD representation of a symbolic matrix determinant, all the error controlling mechanisms such as monitoring response magnitudes/phases, poles/zeros, can be effectively implemented in our framework. This is only feasible previously in approximation-after-generation procedures, which work only for small analog circuits [5, 16].

Our DDD-based approximation algorithm uses the following procedures: 1) *discarding insignificant terms*, 2) *removing canceling terms* and 3) *generation of dominant terms*.

3.1. Discarding Insignificant Terms

Discarding insignificant terms is to delete those terms not significant to the characteristics of interest. It consists of two methods: *device elimination* and *node contraction*.

Consider a transfer function written in the following form:

$$f(p) = \frac{N}{D} = \frac{pN_p + N_{\overline{p}}}{pD_p + D_{\overline{p}}}, \tag{3}$$

where p is a circuit element in the admittance form, N_p (D_p) is the sum of all the product terms in N (D) containing p from which p is removed, and $N_{\overline{p}}$ ($D_{\overline{p}}$) is the set of product terms in N (D) not containing p.

There are two scenarios where p can be eliminated from both N and D. First, if both $D_{\overline{p}}$ and $N_{\overline{p}}$ are compared to N and D, then f can be simplified by $f' = \frac{N_{\overline{p}}}{D_{\overline{p}}}$. This step removes those devices that are not significant in the small-signal characteristics of interest. It is called *device elimination*.

Secondly, if both pN_p and pD_p dominate N and D, then f can be simplified by $f' = \frac{N_p}{D_p}$. This is called *node contraction*. With this, the number of elements in each product term is decreased by one. Different from [17], which considers each individual device p, we consider a group of devices connected to a particular circuit node. This idea has been proven to be more effective, since for such circuits as Opamp, many devices in the bias circuity can be eliminated without affecting small-signal circuit behaviors.

Both device elimination and node contraction are performed on complex DDDs and involve mainly DDD *Cofactor* and *Remainder* operations which take liner time in the DDD size [9]. After this procedure, the simplified DDDs are expanded into multi-root DDDs which are further simplified by suppressing some coefficients of high powers of s *suppression of high powers of* s.

3.2. Elimination of Symbolic Cancellation

Term cancellation in the framework of determinant expansion comes from the MNA formulation and device matching in analog circuits. For an illustrative propose, consider the s-expanded DDD in Figure 3. Since $g = k = \frac{1}{R_3}$ and $i = j = -\frac{1}{R_3}$, term agk cancels term $-aji$ in the coefficient DDD of s^0.

Symbolic canceling terms may greatly degrades the efficiency of our dominant term generation algorithm (described in Section 3.3). In practice, a majority of expanded terms may cancel each other due to the MNA formulation. Removing canceling terms has been known to be a difficult problem in determinant-based symbolic analysis methods [5, 8].

We consider several matrix patterns shown in Figure 4. For example, case 1 may come from the rectangular appearance of a floating resistor in the nodal admittance formulation.

$$\begin{array}{c|cc} & \mathbf{l} & \mathbf{k} \\ \hline \mathbf{i} & \mathbf{p} & \mathbf{-p} \\ \mathbf{j} & \mathbf{-p} & \mathbf{p} \end{array} \qquad \begin{array}{c|cc} & \mathbf{l} & \mathbf{k} \\ \hline \mathbf{i} & \mathbf{p} & \mathbf{-q} \\ \mathbf{j} & \mathbf{-p} & \mathbf{q} \end{array} \qquad \begin{array}{c|cc} & \mathbf{l} & \mathbf{k} \\ \hline \mathbf{i} & \mathbf{p} & \mathbf{-p} \\ \mathbf{j} & \mathbf{-q} & \mathbf{q} \end{array}$$

Case 1 **Case 2**

Figure 4: Matrix patterns causing term cancellation.

Let L_1,L_2,L_3, L_4 denote, respectively, unique DDD symbols at four rectangular positions $(i, l), (i, k), (j, l), (j, k)$. Then we can prove the following lemma:

Lemma 1 *For each product term containing L_1 and L_4, there exist a corresponding canceling product term containing L_2 and L_3.*

Canceling terms caused by matrix pattern cases 1 and 2 can be removed efficiently from a DDD by using basic DDD operations: first perform two cofactoring operations with respect to either L_1 and L_4 or L_2 and L_3; then multiply the obtained DDD with both L_1, L_4, and L_2, L_3 respectively to obtain the complete canceling terms; finally subtract all the canceling terms from the original DDDs. All these operations can be done in linear time in the size of a DDD [9]. For our illustrative example, the cancellation-free DDD is shown in Figure 5 with 13 paths. Matrix patterns involving more than two rows (columns) can also cause term cancellation. But our experimental results indicate that most canceling term can be removed effectively by just considering cases 1 and 2.

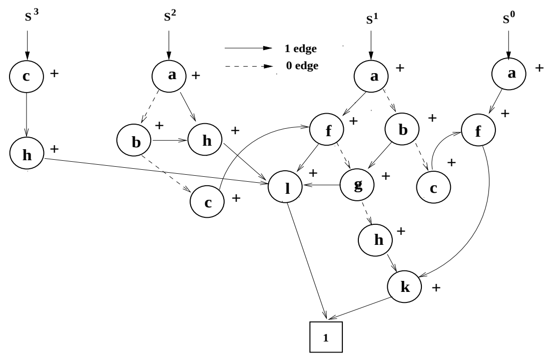


Figure 5: Cancellation-free multi-root DDD.

3.3. Generation of Dominant Terms

Many small-signal characteristics are dominated by a small number of product terms called *significant* or *dominant* terms. In our framework, the extraction of significant product terms can be transformed to the problem of finding k shortest paths in a DDD.

We need to introduce the notion of path weight in DDDs.

Definition 1 *The cost of a path in a DDD is defined to be the total cost of the edges along the path where each 0-edge costs 0 and each 1-edge costs $-log|a_i|$, and $|a_i|$ denotes the numerical value of the DDD vertex a_i that originates the corresponding 1-edge.*

We can show the following result:

Lemma 2 *The most significant product term in a symbolic determinant D corresponds to the minimum cost (shortest) path in the corresponding DDD between the root and the 1-terminal.*

The shortest path in a DDD can be found by depth-first search in time $O(V)$, where V is the number of DDD vertices [1]. A nice property of DDD is that after we find the shortest path from a DDD, we can subtract it from the DDD using a basic DDD operation [9], and then we can find the next shortest path in the resulting DDD. In this manner, we can find the k shortest paths in time $O(k \cdot V)$.

This procedure can be performed on the s-expanded DDD, after the decancellation procedure. Error controlling is carried out by enumerating the dominant terms from all the coefficient DDDs simultaneously according to certain criteria, until the generated terms, coming from different coefficient DDDs, well approximate the exact expressions in terms of magnitudes and phases.

We note that this approach also handles numerical cancellation. Since numerical canceling terms are extracted one after another, they can be eliminated by examinating two consecutive terms.

4. Experimental Results

The proposed approach has been implemented. Here we describe results for three integrated circuit examples *TwoStage*, *Cascode*, $\mu A741$ with schematics shown, respectively, in Figure 6, Figure 7, and Figure 8.

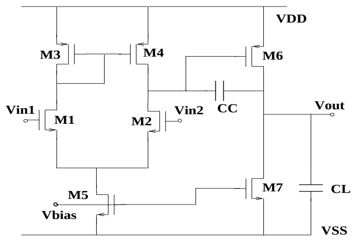


Figure 6: Simplified two-stage CMOS opamp [4].

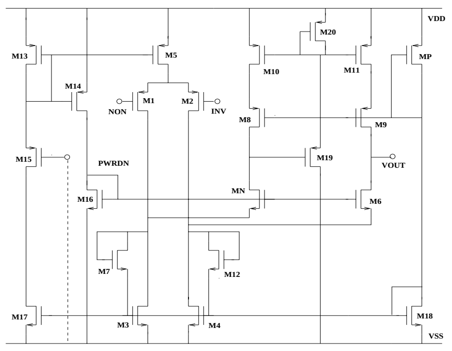


Figure 7: CMOS cascode opamp.

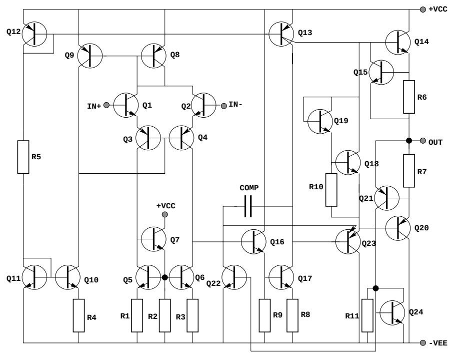


Figure 8: Bipolar $\mu A741$ opamp.

For each circuit, DC analysis is carried out using SPICE and our program reads in small-signal element values from the SPICE output. The algorithms described in [9, 10] are used to construct complex DDDs and s-expanded DDDs. Row 3 to row 9 in Table 2 summarize the statistics about the circuits, the complex $DDDs$ and s-expanded $DDDs$. We can observe that DDD is highly compact, and the number of vertices is many orders of magnitude less than the number of product terms. For example, the denominator of the s-expanded DDD for $\mu 741$ has $6.40 * 10^{19}$ product terms, but the entire DDD to represent both the denominator and numerator contains only 297117 vertices (this is for the complete and exact transfer function)!

Next, we apply the proposed approximation algorithms to derive interpretable symbolic expressions for transfer functions and poles. In each simplification step, we monitor both magnitude and phase of the simplified expressions to control the accumulated error within a given frequency range. We perform device removal and node contraction based on the complex DDD representation. The results are summarized in row 10 to row 13. We observe that this step removes devices that do not affect the small-signal characteristics of the circuits (mainly the bias circuity not in the signal path) and this reduces significantly the size of a complex DDD.

Next, we construct multi-root $DDDs$ from the simplified complex $DDDs$. Note that even after device elimination and node constriction, the number of product terms by the matrix determinant method is still in the range of millions. We then perform three simplifications: First, suppress those insignificant high order coefficients. The results are shown in row 16 to row 18. This reduces the size of each DDD by about 10 percent. Second, we perform de-cancellation, and the results are shown in row 19 to row 22. We see that over 80 percent terms are canceling terms. However, we observe that elimination of canceling terms may not necessarily reduce the size of $DDDs$, since much sharing in the original DDD may be destroyed.

Finally, we extract the significant terms from the resulting s-expanded $DDDs$. For *TwoStage*, *Cascode* and $\mu A741$, the number of product terms in the final simplified transfer functions (including both the numerator and denominator) are respectively 6, 83, and 71. In Figure 9, Figure 10, and Figure 11, we plot the voltage gain and phase responses using both the exact and simplified expressions. The results from Rainier [17] for *Cascode* and $\mu A741$ are also plotted for comparison, where Rainier's expressions for *Cascode* and $\mu A741$ contain, respectively, 784 and 89 product terms. The whole simplification process takes 11.3 seconds in Sun Ultra-I Workstation for *Cascode*, and 54.7 seconds for $\mu A741$.

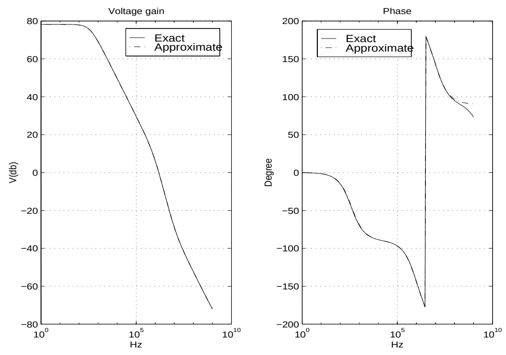
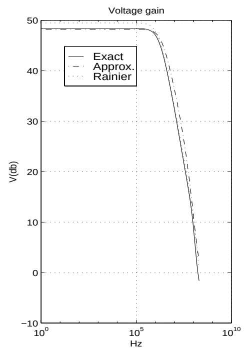


Figure 9: Accuracy comparison for *TwoStage*.

Figure 12 and Figure 13 show the distributions of number of product terms with respect to different powers of s in the exact symbolic expression, the expression after elimination of insignificant terms and the expression after de-cancellation process in the



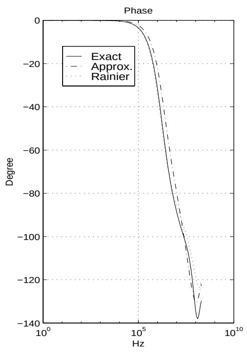
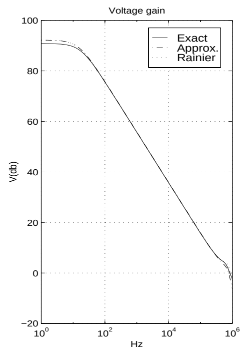


Figure 10: Accuracy comparison for *Cascode*.



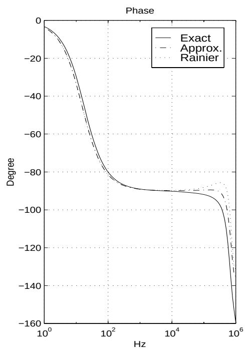


Figure 11: Accuracy comparison for $\mu A741$.

denominator of transfer functions for *Cascode* and $\mu A741$.

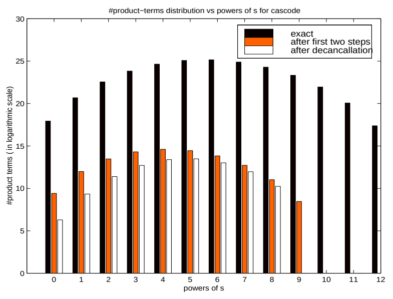


Figure 12: #terms distribution vs powers of s for cascode

The simplified voltage gain for *TwoStage* given by our program is:

$$\frac{g_{m2}g_{m6} + s^1(g_{m2}CC)}{(r_{o2}+r_{o4})(r_{o6}+r_{o7}) - s^1(g_{m6}CC) + s^2(c_{db6}+CL)CC}$$

For *TwoStage*, Table 1 shows the exact values of three zeros and three poles.

Since three poles are far away from each other, the pole splitting method can be used to find the symbolic expressions for three poles. The resulting expression of the third pole based on DDD manipulations is as follows:

$$-\frac{g_{m3}}{c_{gd1}+c_{db1}+c_{gs3}+c_{db3}+c_{gs4}} = -1.68 * 10^7.$$

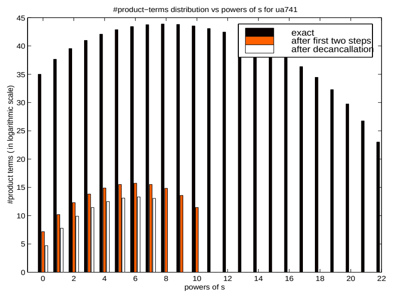


Figure 13: #terms distribution vs powers of s for $\mu A741$

Table 1: Zeros and Poles for Two Stage Opamp.

poles	$-1.68 * 10^7$	$-8.80 * 10^5$	-373.74
zeros	$3.18 * 10^9$	$-1.68 * 10^7$	$1.11 * 10^7$

This agrees with the exact third pole described in Table 1. However, we note that if we apply the pole splitting method directly on the simplified expression of the transfer function as done in many previous approaches, we cannot obtain the third pole. This is because the third pole and second zero cancel each other, and the third pole information may get lost during a simplification.

5. Conclusions

A new approach is proposed to automatically generate interpretable expressions for small-signal characteristics of large analog circuits. Unlike simplification-during-generation approaches, the proposed approach works on a complete and exact representation of transfer functions. It can monitor the errors in the magnitude, phase, poles and zeros of simplified expressions, and thus provides a reliable and robust simplification scheme. The proposed approach is based on a compact but canonical representation of symbolic expressions, and has the time and space complexity many-orders-of-magnitude less than the simplification-after-generation approaches. It provides, for the first time, a framework for deriving not only interpretable network functions, but also interpretable expressions for other small-signal characteristics such as poles, zeros and sensitivities. The proposed approach has been implemented and demonstrated a superior performance over other state-of-art tools.

References

[1] T. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts 1990.

[2] 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 and Systems*, pp. 1149–1152, 1992.

[3] F. V. Fernández,P. Wambacq, G. Gielen, A. Rodríguez-Vázquez and and W. Sansen, A. Rodríguez-Vázquez, "Symbolic analysis of large analog integrated circuits by approximation during expression generation," in *Proc. IEEE Int. Symp. Circuits and Systems.*, pp. 25–28, 1994.

[4] H. Floberg, *Symbolic analysis in analog integrated circuit design*, Kluwer Academic Publishers, Massachusetts, 1997.

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

[6] J.-J. Hsu and C. Sechen, "DC small signal symbolic analysis of large analog integrated circuits", *IEEE Trans. Circuits and Systems*, vol. 41, no. 12, pp. 817–828, Dec. 1994.

[7] P. M. Lin, *Symbolic Network Analysis*, Elsevier Science Publishers B.V., 1991.

[8] P. Sanniti and N. N. Puri, "Symbolic network analysis—An algebraic formulation", *IEEE Trans. Circuits and Systems*, vol. 27, no. 8, pp. 679–687, Aug. 1980.

[9] C.-J. Shi and X.-D. Tan, "Symbolic analysis of large analog circuits with determinant decision diagrams", in *Proc. IEEE Int. Conf. Computer Aided Design (ICCAD)*, pp. 366–373, Nov. 1997.

[10] C.-J. Shi and X.-D. Tan, "Efficient derivation of exact s-expanded symbolic expressions for behavioral modeling of analog circuits", in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, pp. 463-466, 1998.

[11] S. J. Seda, M. G. R. Degrauwe and W. Fichtner, "A symbolic analysis tool for analog circuit design automation," in *Proc. IEEE Int. Conf. Computer Aided Design (ICCAD)*, pp. 488–491, 1988.

[12] S. J. Seda, M. G. R. Degrauwe and W. Fichtner, "Lazy-expansion symbolic expression approximation in SYNAP", in *Proc. IEEE Int. Conf. Computer Aided Design (ICCAD)*, pp. 310–317, Nov. 1992.

[13] X.-D. Tan and C.-J. Shi, "Hierarchical symbolic analysis of large analog circuits with determinant decision diagrams", in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 318-321, June 1998.

[14] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, New York, 1994.

[15] P. Wambacq, G. Gielen and W. Sansen, "A cancellation-free algorithm for the symbolic simulation of large analog circuits", in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 1157–1160, May 1992.

[16] P. Wambacq, G. Gielen and W. Sansen, "A new reliable approximation method for expanded symbolic network functions", in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 584–587, 1996.

[17] Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits", *IEEE Trans. Circuits and Systems*, vol. 43, no. 8, pp. 656–669, Aug. 1996.

Table 2: Statistics of simplified symbolic expressions

#row	Circuit	Twostage		Cascode		$\mu A741$	
2		Num	Den	Num	Den	Num	Den
3	#lumped devices	14		84		111	
4	#nodes in the circuit	5		14		24	
5	#terms in exact complex DDDs	1	3	9437	28218	381526	$3.82 * 10^6$
6	#vertices in exact complex DDDs	12		1406		9572	
7	#terms in exact multi-root DDDs	16	74	$2.57 * 10^{10}$	$3.60 * 10^{11}$	$8.98 * 10^{16}$	$6.40 * 10^{19}$
8	#vertices in exact multi-root DDDs	52		18109		297117	
8	Highest power of s in exact expressions	3	3	12	12	22	22
10	#terms after device elimination in compex DDDs	1	2	800	1752	816	9338
11	#devices eliminated	6		34		68	
12	#nodes contracted in compex DDDs	1		3		8	
13	#terms after contraction in compex DDDs	1	2	33	107	84	516
14	#terms after expansion in multi-root DDDs	2	11	$4.28 * 10^5$	$8.04 * 10^6$	$2.06 * 10^5$	$2.56 * 10^7$
15	Highest power of s after expansion	1	2	7	9	9	10
16	Highest power of s after suppression	1	2	7	8	5	7
17	#terms after suppression in multi-root DDDs	2	11	$4.28 * 10^5$	$8.04 * 10^6$	$1.52 * 10^5$	$2.19 * 10^7$
18	#vertices after suppression in multi-root DDDs	18		1873		2641	
19	#terms after decancellation in multi-root DDDs	2	9	$2.00 * 10^5$	$2.43 * 10^6$	23162	$1.95 * 10^6$
20	#vertices after decancellation in multi-root DDDs	17		2500		4303	
21	#cancelling terms eliminated due to case 1	2		$3.25 * 10^6$		$1.68 * 10^7$	
22	#cancelling terms eliminated due to case 2	0		$2.58 * 10^6$		$3.31 * 10^6$	
23	#terms in final expression	2	4	21	62	12	59
24	Highest power of s in final expression	1	2	4	6	3	4
25	#lumped devices in each product term	2		9		14	