Circuit Complexity Reduction for Symbolic Analysis of Analog Integrated Circuits

Walter Daems Georges Gielen Willy Sansen
Katholieke Universiteit Leuven, Department of Electrical Engineering, ESAT-MICAS
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
daems@esat.kuleuven.ac.be gielen@esat.kuleuven.ac.be sansen@esat.kuleuven.ac.be

Abstract

This paper presents a method to reduce the complexity of a linear or linearized (small-signal) analog circuit. The reduction technique, based on quality-error ranking, can be used as a standard reduction engine that ensures the validity of the resulting network model in a specific (set of) design point(s) within a given frequency range and a given magnitude and phase error. It can also be used as an analysis engine to extract symbolic expressions for poles and zeroes. The reduction technique is driven by analysis of the signal flow graph associated with the network model. Experimental results show the effectiveness of the approach.

1 Introduction

Symbolic analysis of analog electronic circuits is to generate symbolic expressions for the parameters that describe the performance of the circuit [1], [2], [3], [4]. The aim of linear symbolic analysis is to obtain expressions relating linear network performance parameters with both the frequency and the vector of small-signal values x_q retained as symbols. These expressions typically are ratios of polynomials in the Laplace variable s, with as coefficients sums of products of small-signal elements x_q of the design point vector $X = [x_1 x_2 \ldots x_q \ldots x_Q]^T$[1] as illustrated in equation (1). N and M are the number of edges respectively in a denominator and numerator spanning tree. These expressions indicate which parameters to change in order to achieve a wanted performance.

$$H(s) = \frac{\sum_{p=0}^{M} \left(s^p \sum_{k=1}^{K_p} \prod_M x_q \right)}{\sum_{i=0}^{N} \left(s^i \sum_{l=1}^{L_i} \prod_N x_q \right)} \qquad (1)$$

However, symbolic analysis by hand has proven to be very error prone and limited to fairly small circuits [5]. The automation of this straight analysis task alleviated the former problem, but not the latter one. Indeed, the expressions resulting from symbolic analysis tools tend to be very lengthy. In order to keep the obtained expressions useful for interpretation as well as manipulation, symbolic approximation is mandatory: one trades accuracy for simplicity. Such approximation is based upon the order of magnitude of the symbolic circuit parameters in a specific part of the design space.

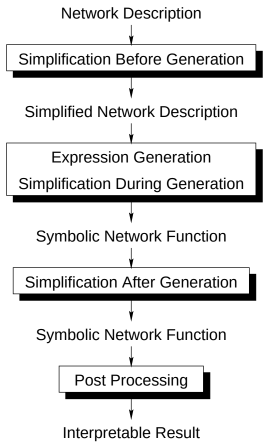

Network Description
Simplification Before Generation
Simplified Network Description
Expression Generation
Simplification During Generation
Symbolic Network Function
Simplification After Generation
Symbolic Network Function
Post Processing
Interpretable Result

Figure 1. Approximation steps during the symbolic analysis of analog electronic circuits

The approximation can occur before, during and after the generation of the symbolic expression [6]. This has been depicted in Fig. 1. The simplification after generation (SAG) technique historically was developed first and merely pruned the insignificant terms from the obtained exact expression [3], [7]. As this method became — due to memory and time constraints — infeasible for practical large circuit sizes, simplification during generation (SDG) techniques were developed that overcame the size limitation [6], [8]. About at the same time simplification before generation (SBG) techniques were developed to overcome the same limitations [9], [10]. Most likely both techniques are combined in order to obtain acceptable results.
However, the strictly "mathematical" nature of the approximation, pruning terms of the coefficient subexpressions or generating their dominant terms only, did not change: it destroys the factorability of the result and as a consequence the interpretability of the expressions. The extraction of insightful information like poles, zeroes and gain levels, thus becomes extremely difficult for higher-order polynomials and works only in case of widely spread poles and zeroes. Since simplification before generation (SBG) techniques directly operate on the circuit's network model using network transformations, the approximation inherently is less "mathematical": one expects the factorability to be less violated.
In fact, the key problem lies in the format of the generated expressions. It is observed that, in general, each pole or zero of a network is determined by only a small portion of the network's elements. These elements are often topologically very localized. More specifically, they are gathered in one or two signal paths or loops of the network. Casting the entire performance characteristic into a flat expression ruins all information concerning the topological localization of poles and zeroes. Mathematically extracting expressions for poles and zeroes from a flat expression, even when it has not been

W. Daems and G. Gielen are respectively research assistant and research associate of the Fund for Scientific Research, Vlaanderen.

[1]The variables contained in the design vector are not necessarily independent.

simplified, is almost an impossible thing to do. Factorization techniques have been developed in computer algebra, but are not fully satisfactory.
The method presented in this paper, allows being used in a standard symbolic analysis scheme as outlined in Fig. 1 and thus helps overcoming the time and memory constraints mentioned before. However, it also allows a localization analysis of poles and zeroes of a linear network model by avoiding the destructive casting into a flat expression by directly taking advantage of the topology of the circuit.
In section 2 we will describe how to translate a linear electrical network into an equivalent signal flow graph. In section 3 we will define the notion of network reduction. Section 4.1 will describe the fundamentals of the network reduction technique. How to order these network transformations will be discussed in section 4.4. The fundamentals on which this ordering is based will be explained in sections 4.2 and 4.3. Next, in section 5 we will discuss the localization of poles and zeroes. The prototype tool that implements this concept will be described in section 6. The experimental results obtained using this prototype tool will be presented in section 7. Finally, conclusions will be drawn in section 8.

2 Signal flow graph model of a network

The transformation of a linearized electrical network model in a (connected, directed) signal flow graph can be performed in numerous ways [11], [12]. The key decision to be made is the set of vertices and thus the signal entities to build the graph on. To avoid confusion, we will denote network nodes as nodes and graph nodes as vertices. As a starting point, assume using two vertices corresponding to each network node: one representing the node voltage (the so called voltage vertex) and the other representing the node voltage multiplied by the sum of all self-admittances attached to the node (the so called current vertex). Building a signal flow graph on this principle can be accomplished by replacing all network elements using the conversion table of Fig. 2. Voltage vertices have been indicated in black, current vertices in grey. The vertices to be merged are indicated by a dashed line in-between them. After straight application of this substitution, parallel graph edges can be lumped together into meta-edges by summing the admittance values. In this manner every meta-edge holds one ore more edges, partitioned using their frequency dependence, which can be proportional to s^1, s^0 or s^{-1}. We will denote such a partition as a meta-edge's order partition. As a final step the graph still needs some rearrangement in order to ensure validity and avoid redundancy, but we won't discuss this in detail here.
The signal flow graph defined that way is closely related to the Coates flow graph built on the modified nodal admittance matrix [12]. This type of graph closely resembles a designer's way of thinking, for it clearly indicates where voltages generate currents and vice versa. As an illustrative example, consider the graph representation in Fig. 4 of the single transistor amplifier depicted in Fig 3. The transistor model used has been indicated on Fig. 3 by the inset. Notice how in this case meta-edges from voltage vertices to current vertices generate open-loop poles and meta-edges from current vertices to voltage vertices generate open-loop zeroes. In general, closed loop poles/zeroes originate either by these open-loop poles/zeroes or by the interaction of signals at summing vertices. Summing vertices are vertices with a meta-edge in-adjacency larger than one.

3 Network reduction

Let's first state a definition of what network model reduction is in the scope of symbolic analysis:
Reduction of a network model N is applying a sequence of S_T network model transformations T_i in order to obtain a network model N' that exhibits a reduced complexity at the price of some error on the performance.
Formally:

$$N' = \left(\bigodot_{i=1}^{S_T} T_i \right)(N) = \left(T_{S_T} \circ \cdots \circ T_2 \circ T_1 \right)(N) \tag{2}$$

The definition contains three important notions: (1) network model transformation, (2) model complexity and (3) model performance error. They will be discussed in the next three sections (4.1-4.3). The subsequent section (4.4) will be devoted to a problem hidden somewhat deeper in the definition: the ordering of the transformation sequence.

4 Network model transformations

4.1 Definition

A network transformation can be defined as an operation that maps the original model to a new model relating the same input and output variables.
Let's call the set of candidate transformations τ:

$$\tau = \{T \mid T \text{ is a network model transformation}\} \tag{3}$$

When describing the network model as a signal flow graph, we can distinguish different useful graph transformation categories. We will illustrate these sets using the graph of Fig. 4.

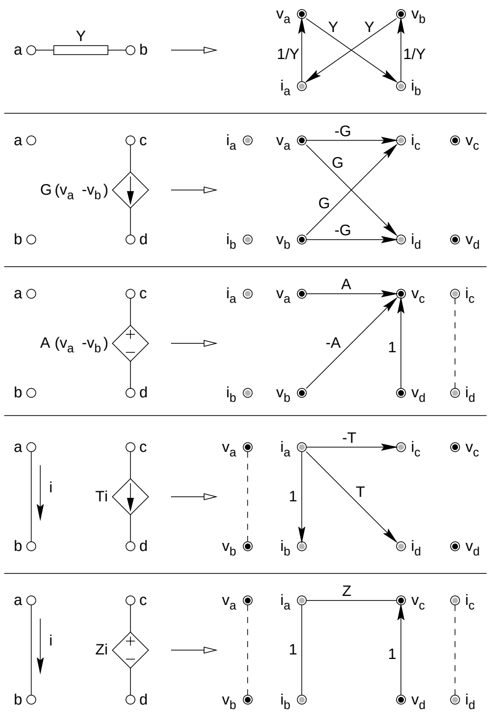


Figure 2. Network element to signal flow graph conversion table

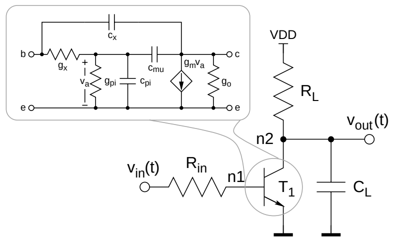


Figure 3. Schematic of the single transistor amplifier

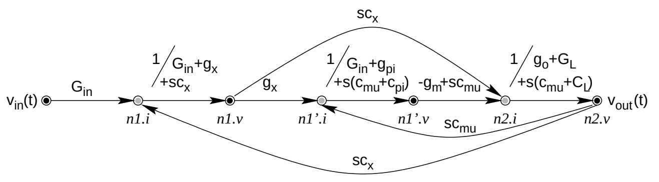


Figure 4. Signal flow graph of the single transistor amplifier

• **Removal of signal paths**: by removing an incoming meta-edge of a summing vertex one or more signal paths are removed from the graph. We will label this set τ_{RSP}.
E.g. removing the meta-edge going from vertex $n1.v$ to $n2.i$ corresponds to removing a forward path from the graph.
• **Open-loop root removal**: by removing one of the extreme order partitions of a meta edge, one or two open-loop roots are removed from the graph. We will label this set τ_{RR}.
E.g. removing the capacitive order partition of the meta-edge from vertex $n1'.v$ to $n2.i$, removes the open-loop zero $\omega_z = g_m/c_{mu}$.
• **Vertex-pair contraction**: this means shorting the voltage vertices and the current vertices associated with two nodes. We will label this set τ_{VPC}.
E.g. contracting vertex $n1'.v$ with $n1.v$ and $n1'.i$ with $n1.i$ which corresponds to shorting the transistor's base resistance r_x.
• **Subgraph substitution**: this means replacing a part of the graph by another graph without violating the topological integrity (i.e. connectedness of the graph). We will label this set τ_{SS}.
E.g. replacing the subgraph between nodes $n1.i$ and $n2.v$ by an impedance edge with value $1/sc_x$.
Of course removing signal paths, removing open-loop roots and vertex pair shorting can be regarded as subgraph substitution, but we explicitly exclude these substitutions from τ_{SS}.

4.2 Network model performance error

When applying network model transformations, the performance of the transformed network model will deviate from the original model's performance. We'd like to keep this deviation as small as possible, i.e. we'd like the model to reflect the original performance within a user specified accuracy.
When considering a vector P of performance parameters $p_i(N, X)$, we define the model performance error E_P as the weighted norm of the performance deviation vector ΔP:

$$E_P(N, T, X) = \|\Delta P\|_W \tag{4}$$
$$\text{with } \Delta P = P(N', X) - P(N, X) \tag{5}$$

and in which W represents the norm weight vector. When considering the performance error with respect to the original network model we denote it as the absolute performance error. When considering it relative to the performance of the network model obtained by the previous network model transformation, we denote it as the relative error and replace the Δ by a δ. Notice the dependence of the performance error on the design point X. This dependency introduces an additional problem. While we can assess the performance error in the design point, extrapolation outside the design point is very dangerous. We could revert to using ranges on the symbolic parameters to estimate the performance error, but this approach leads to very conservative results. Usually an in between solution is chosen, taking multiple design points that are more or less representative for the region under study, gathering their performances into one performance vector.

4.3 Network model complexity

In our approach all networks are modeled as signal flow graphs. Therefore, the problem of quantifying a network model's complexity is reduced to quantifying the complexity of a graph. In general a diversity of graph properties can be considered to contribute to the graph's complexity: the number of nodes, the number of edges combined with their type (which may yield different complexity contributions), the cardinality of the in- and out-adjacencies of the vertices, the number of paths, the number of loops, etc. All of these are valid complexity contributors, but how should we combine them into one overall complexity figure?
We've been guided by a simple observation: a graph's complexity originates from two sources: (1) the number of closed loop roots (poles and zeroes) present in the graph, and (2) the graph's connectivity. Indeed, when considering network functions of the form of equation (1), the larger the number of closed loop poles and zeroes, the higher the net order (the difference between the exponent of the highest and the lowest power of s) of the numerator and the denominator polynomial will be. The larger the connectedness of the graph, the more spanning trees will be common to voltage and current graph, and hence the more terms within one coefficient of s will be generated [6]. We will model the connectedness of the graph conveniently as the total number of forward paths and feedback loops.
One link is still missing to fully explain the chosen complexity function: how do closed loop poles and zeroes, that are not straight visible on a graph, correspond to open-loop poles and zeroes. Two cases occur: (1) a closed loop root can originate directly (or after shifting) from an open-loop root if the open-loop root occurs in a dominant path or feedback loop of the graph; (2) a closed loop root can also occur when at a certain frequency, one signal path takes over from another. One could consider the origin of such a root to be the summing point that brings both signal paths together. When this takeover occurs in a loop with signals at the summing point that are nearly in phase together with a loop gain larger than one, complex conjugate roots can occur. In a sense one could state that every summing point is a candidate for the generation of as much closed loop roots as there are meta-edges that are entering the vertex (i.e. the cardinality of the in-adjacency of meta-edges to the vertex) minus one.
In view of the above, we define as complexity function of a network graph:

$$C(N) = \underbrace{n_{fol} + n_{sum}}_{\text{candidate roots}} + \underbrace{n_{fwp} + n_{fbl}}_{\text{connectedness}} \tag{6}$$

with

• n_{fol}	: number of open-loop roots
• n_{sum}	: number of summing points
• n_{fwp}	: number of forward paths
• n_{fbl}	: number of feedback loops

One might argue that counting the number of candidate roots as representative for the closed loop roots is a huge overestimation and partly overlaps with the estimation for the connectedness of the graph. However the more summing points are eliminated also the more certainty arises concerning where to locate these closed loop roots and therefore this choice is not as strange at it seems. Yet, as we will see later in the outline of the algorithm, we've taken this criticism into account.
Some remarks concerning the calculation of this complexity number: the terms needed for the calculation of the number of candidate roots can be obtained readily by graph inspection; the number of forward paths can easily be found by a depth-first search through the graph. One problem however arises. Initially, i.e. before any simplification, the traversal of this graph for realistic circuits is unfeasible in acceptable time limits. Indeed, this is the key problem of symbolic analysis: the exponential relationship between the complexity and the number of nodes! Therefore complexity calculations

can only be carried out after a presimplification of the graph. This presimplification consists of an error controlled contraction of bias circuitry vertices with the ground vertex using a small fraction (typically 1/10 000) of the user specified error.

4.4 Network transformation ranking

The key question that hasn't been answered yet, is how to order all possible network model transformations. The components for this trade-off have been described in the previous sections. The idea is to prefer network model transformations that offer a low performance error to complexity reduction ratio. We therefore define a ranking function R_P

$$R_P\left(N, T_j, X\right) = \frac{\left[E_P\left(N, T_j, X\right)\right]^{\alpha}}{\left[Q_P\left(N, T_j\right)\right]^{1-\alpha}} \tag{7}$$

to order the possible transformations with the transformation quality Q_P defined as:

$$Q_P\left(N, T_j\right) = C\left(N\right) - C\left(T_j\left(N\right)\right) \tag{8}$$

and $0 \leq \alpha \leq 1$. The parameter α is just a tuning parameter that can shift the ranking from quality concerned ($\alpha = 0$) to error concerned ($\alpha = 1$). According to the method to calculate the performance error, we can distinguish relative ranking and absolute ranking. We will only consider network transformations exhibiting a positive ranking function and we will favor transformations with lower ranking values. In the SBG techniques reported up till now as known to the authors [9], [10], no quality measure was ever taken into account for the ranking of the transformations. The advantage of this new approach is twofold: (1) less transformations have to be applied to achieve the same level of simplification, for a simplification with a high error impact that contains several low impact simplifications can now have a lower ranking than the low impact simplifications themselves; (2) one can now steer the simplification towards a specific simplification goal, e.g. a low number of summing points, which can be helpful when localizing poles and zeroes. Especially steering the simplification in the direction of simple graph topologies, results in easier graph decomposition. We will return to this topic in sections 5 and 7.

5 Localization of poles and zeroes

The observation that a pole or a zero is only determined by elements that are active in its frequency neighborhood, leads us to the concept of root clustering. The idea is to split the error specification — that specifies a maximum magnitude and phase error within some frequency range — into separate subspecifications each valid in some part of the total error specification range. Roots that differ less than a user specified difference ε_f are put in the same root subset. These subsets form a partition on the set containing all roots. Consider having Q subsets (numbered $q = [1:Q]$) of n_q roots, $\omega_{q,i}$ each, with $i \in [1:n_q]$. Let the sets as well as the roots in the sets be ordered from low to high. The borders of the cluster ranges $[f_{q,lo}, f_{q,hi}]$ are now defined as:

$$\omega_{q,lo} = \begin{cases} 0 & \text{if } q = 1 \\ \sqrt{\omega_{q-1,n_{q-1}}\omega_{q,1}} & \text{if } q = [2:Q] \end{cases} \tag{9}$$

$$\omega_{q,hi} = \begin{cases} \sqrt{\omega_{q,n_q}\omega_{q+1,1}} & \text{if } q = [1:Q-1] \\ +\infty & \text{if } q = Q \end{cases} \tag{10}$$

Intersecting these cluster ranges with the original error specification results in the wanted subrange specifications. Analyzing the network model in these restricted areas is very beneficial for the simplicity of the resulting model. In case of widely spread poles or zeroes, it allows their full extraction by reducing the netto order of the network model to 1.

```
1. Transform circuit into linear network model $N$
2. Lump linear network model $N$
3. Transform network model $N$ to signal flow graph model $G$
4. Calculate reference performance
5. Perform root clustering
6. Split error specification using the root clusters
7. While graph not analyzable
  7.1. error controlled prereduction of the graph $G$
8. Analyze topology of $G$
9. for all subspecifications
  9.1. create subgraphs $G_j$
  9.2. for $\tau$ in $\tau_{VPC}$, $\tau_{RSP}$, $\tau_{RR}$, $\tau_{SS}$
    9.2.1. for $\forall T_i \in \tau$: calculate relative ranking values
    9.2.2. perform ranking
    9.2.3. execute $T_i(G_j)$ while error not exceeded
  9.3. Analyze topology of $G_j$ (pole/zero localization)
```

Figure 5. Circuit complexity reduction algorithm

In the subranges where more than one root is active, a closer inspection is needed. The process of root detection takes two steps.
The first step is the root identification. As mentioned before, two kinds of internal roots can occur, open-loop roots and roots originating from summing points. For the former ones the identification can be done by simple inspection of the meta-edges order partitions. For the latter ones, we revert to graph decomposition. This decomposition is guided by the topological analysis made during the circuit reduction. Cuts can be made at complementary vertices (i.e. the first vertex's topological in-adjacency equals the second vertex's topological out-adjacency and vice versa) if the edges between these vertices are not involved in any other path not occurring in the topological in-adjacency of the start vertex.[2] The subgraph from the vertex coming earlier in the graph's topology to the vertex later in the graph's topology can then be analyzed separately. Subgraphs with smaller lengths[3] are analyzed first. By this evolving analysis starting from small enclosed subgraphs to the larger enclosing subgraphs, we can detect the occurrence of poles and zeroes at summing points by comparison and intersection of the incoming signal levels at these summing points.
The second step is root observability checking. By comparing the signals at the root' frequency at the downstream summing points, we can check if the root can propagate itself to become visible at the output.
Using the above procedure a large number of poles and zeroes can be detected and localized. We will illustrate this procedure in section 7.

6 Circuit reduction prototype

A circuit reduction prototype tool has been built. The prototype implemented in C++ allows circuit reduction according to the principles described in previous sections using error control in a single design point. The prototype is however ready for future extensions, like a transition from a quality concerned ranking to an error concerned one, multiple design point error control, etc. It consists of about 10 000 lines of code. The outline of the algorithm can be found in Fig. 5. As one clearly can see, all transformations are not competing simultaneously in the ranking process. Instead we partitioned all possible transformations as indicated in section 4.1 and apply them in sequence. This partitioning avoids an unfair competition between transformations that remove open-loop roots and transformations that simplify the overall topology for the latter will cause a

[2]The notion of valid graph cuts can be extended to multiple input/output subgraphs.

[3]The length of a graph is the minimal number of edges to be traversed to reach the end vertex from the start vertex.

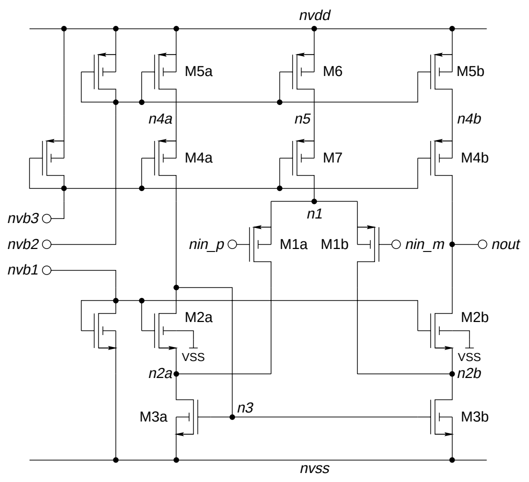


Figure 6. The operational transconductance amplifier (OTA)

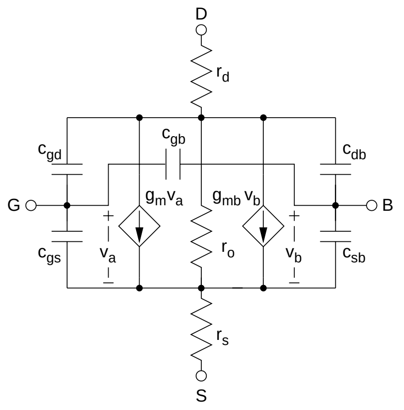


Figure 7. The MOST small-signal model

decrease in all terms of equation (6), while the former only acts on the first term.

7 Experimental results

In order to illustrate the effectiveness and usefulness of the presented method, we will present the results from the analysis of the voltage gain of the operational transconductance amplifier of Fig. 6 using the circuit reduction prototype tool.

The MIETEC C07 process was chosen as target technology and we used the small-signal MOST model of Fig. 7 in combination with the MOS level 3 model to calculate the small-signal values. In order to allow a proper biasing without disturbing the open-loop analysis, a test harness has been used as shown in Fig. 8.

We specified an error of $\pm 3dB$ and $\pm 5^o$ in a frequency range from $10Hz$ to $10GHz$. The reference performance has been plotted in Fig. 9.

Execution time of the entire simplification step using the prototype tool amounts to 75 s on a Sun UltraSparc 30 running at 250 *MHz*. Memory consumption only amounts to 8 MByte. The results in terms of complexity can be found in table I. The first data row indicates the initial complexity after an error controlled prereduction step. In a first run, we disabled the frequency splitting feature. This results in the second line labeled “whole range”. In a second run, we enabled the frequency splitting feature resulting in three subranges. For the subrange around the unity gain frequency is very important concerning phase margin, we added an intermediate subrange around that frequency. The resulting complexity figures are

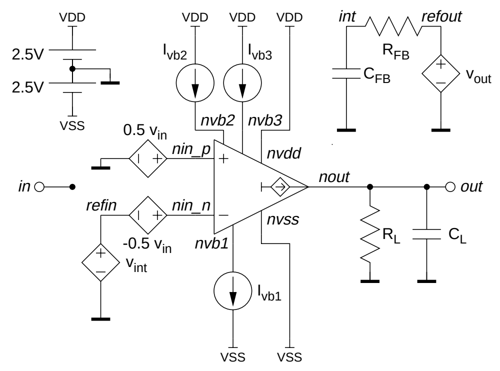


Figure 8. The test setup used to analyze the OTA

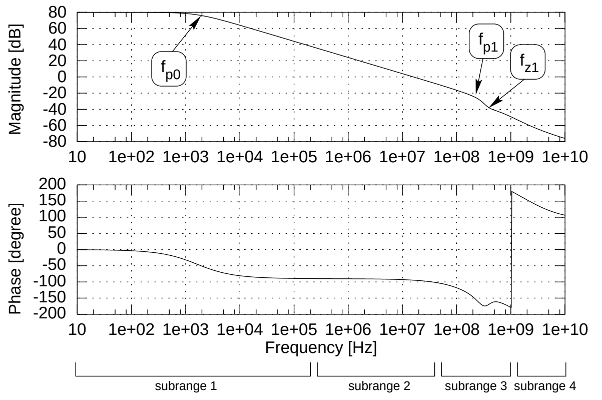


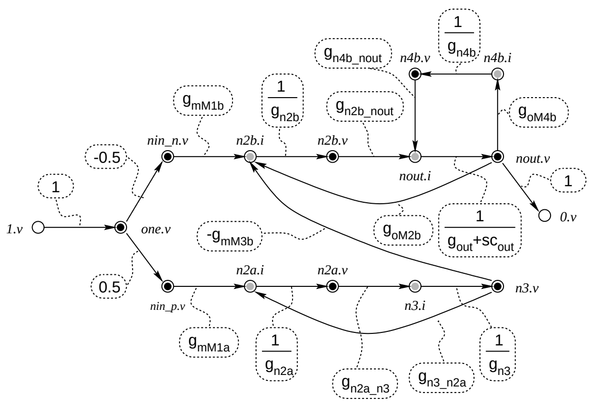
Figure 9. Bode plot of the OTA voltage gain with indication of the analyzed poles and zero and indication of the frequency subranges

indicated on the next lines. As one can see very fine results are obtained for subranges 1, 2 and 4. The results for subrange 3 are not so impressive. However, this result should not surprise us, for this cluster contains 17 closed loop roots enclosed in one and a half frequency decade! We might consider loosening the error specification for that cluster or use root clustering in the complex plane instead of on the imaginary axis.

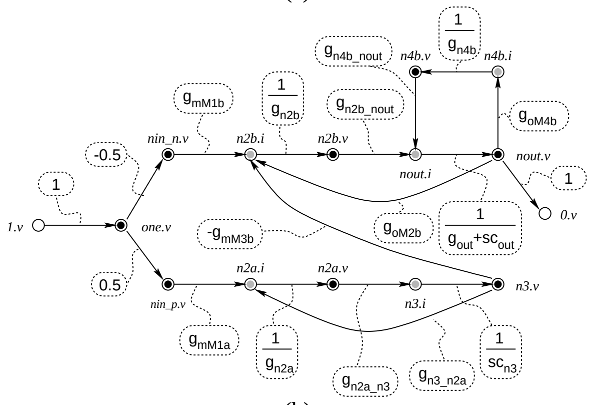
Figs. 10(a) and 10(b) clearly illustrate the informative value of the graphs generated. The first graph indicates the elements that determine the dominant pole. We clearly can locate the three dominant loops at low frequencies: the current mirror loop in the lower branch, and the two cascode loops in the upper branch. We also can assess the reason for the occurence of the dominant pole: the single frequency element c_{out} starts pulling the low frequency gain down by lowering the loop gain in the two cascode loops. We also can directly extract a symbolic expression for the low frequency gain incorporat-

	n_{fol}	n_{sum}	n_{fwp}	n_{fbl}	$C(N)$
Initial	≥ 42	≥ 36	≥ 702	≥ 1190	≥ 1970
whole range	15	13	23	12	63
subrange 1 $10Hz$-$175kHz$	1	4	2	3	10
subrange 2 $175kHz$-$42MHz$	1	4	2	3	10
subrange 3 $42MHz$-$1GHz$	13	12	24	13	62
subrange 4 $1GHz$-$10GHz$	2	0	1	0	3

Table I. Comparison of complexity figures



(a)



(b)

$$
\begin{aligned}
g_{n2a} &= g_{mM2a} + g_{mbM2a} + g_{oM2a} + g_{oM1a} + g_{oM3a} \\
g_{n2b} &= g_{mM2b} + g_{mbM2b} + g_{oM2b} + g_{oM1b} + g_{oM3b} \\
g_{n3} &= g_{oM2a} + g_{oM5a} \\
c_{n3} &= c_{gdM2a} + c_{dbM2a} + c_{gbM3a} + c_{gsM3a} + c_{gsM3b} + c_{gdM3b} + c_{gbM3b} + c_{dbM5a} \\
g_{n4b} &= g_{mM5b} + g_{oM5b} + g_{oM4b} \\
g_{n2a\_n3} &= g_{mM2a} + g_{mbM2a} + g_{oM2a} \\
g_{n3\_n2a} &= g_{mM3a} + g_{oM2a} \\
g_{n2b\_nout} &= g_{mM2b} + g_{mbM2b} + g_{oM2b} \\
g_{n4b\_nout} &= g_{mM4b} + g_{mbM4b} + g_{oM4b} \\
g_{out} &= g_L + g_{oM5b} + g_{oM2b} \\
c_{out} &= c_L + c_{dbM2} + c_{gdM2b} + c_{dbM5b} + c_{gdM5b}
\end{aligned}
$$

Figure 10. Graph resulting from the analysis of (a) the frequency range from $10Hz$ to $175kHz$ and (b) the frequency range from $175kHz$ to $42MHz$

ing the dominant pole.

$$A_{VLF} = \frac{1}{2}\left[\frac{g_{mM1a}g_{mM3b}}{g_{mM3a}+g_{oM2a}} + g_{mM1b}\right]\frac{1}{G_x}\left(\frac{1}{1+sc_{out}/G_x}\right) \quad (11)$$

$$\text{with } G_x = \frac{g_{n2b\_nout}g_{oM2b}g_{n4b} + g_{n4b\_nout}g_{oM4b}g_{n2b}}{g_{n2b\_nout}g_{n4b}} \quad (12)$$

In equation (12) we can clearly find as summing terms the parallel action of the two cascode loops. The second graph indicates the elements that determine the first non dominant pole/zero pair. Compared with the first graph, no open-loop poles or zeroes have been added. Only the transfer from $n3.i$ to $n3.v$ has changed from conductive to capacitive. For in this graph more than one closed-loop root is active, we revert to graph decomposition. We find two valid subgraphs: (1) between vertex $n2a.i$ and $n3.v$ and (2) between $n2b.i$ and $nout.v$. We already analyzed the latter in the previous paragraph. Analysis of the signal levels at the summing point $n2a.i$ of the former learns us that a pole is generated because the loop gain of this (current mirror) loop drops below $0\,dB$ (the gain bandwidth of the loop). This pole is observable at the output for it occurs in one of two equidominant forward paths at summing point $n2b.i$. A correlated topological zero occurs at summing point $n2b.i$ corresponding to the fact that the gain only can drop to 1/2 of the total gain. The results of this analysis thus become:

$$
\begin{aligned}
f_{p1} &= \frac{1}{2\pi}\frac{g_{n2a\_n3}g_{n2\_n2a}}{g_{n2a}c_{n3}} && (13) \\
f_{z1} &= 2\,f_{p1} && (14)
\end{aligned}
$$

These expressions correspond to the frequencies indicated on Fig. 9.

8 Conclusions

In this paper, we presented a method to reduce the complexity of linear or linearized analog circuits taking into account the circuit's graph complexity. This approach allows short circuit reduction times and allows generation of instructive signal flow graphs describing the circuit's behavior in a very effective way. The graphs can be efficiently analyzed using graph decomposition techniques leading to an insight in the localization of closed-loop poles and zeroes. This way the painful nature of flat symbolic analysis techniques, resulting in long uninterpretable expressions, is avoided.

Acknowledgments

The authors would like to acknowledge the support by the European Union within the ESPRIT project 21812 "AMADEUS".

References

[1] G.E. Alderson and P.M. Lin, "Computer generation of symbolic network functions: A new theory and implementation", *IEEE Trans. on Circuit Theory*, vol. 20, no. (1), pp. 48–56, January 1973.

[2] S.J. Seda, G.R. Degrauwe, and W. Fichtner, "A symbolic analysis tool for analog circuit design automation", in *Proc. IEEE/ACM ICCAD*, Santa Clara, 1988, pp. 488–491.

[3] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: a symbolic simulator for analog integrated circuits", *IEEE J. Solid-State Circuits*, vol. 24, no. 6, pp. 1587–1597, Dec. 1989.

[4] F.V. Fernández, A. Rodríguez-Vázquez, J.-L. Huertas, and G. Gielen, *Symbolic Analysis Techniques: applications to analog design*, IEEE Press, 1997.

[5] C.A. Desoer and E.S. Kuh, *Basic Circuit Theory*, McGraw-Hill, California, 1969.

[6] P. Wambacq, F.V. Fernández, G. Gielen, W. Sansen, and A. Rodríguez-Vázquez, "Efficient symbolic computation of approximated small-signal characteristics", *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 327–330, Mar. 1995.

[7] F.V. Fernández, A. Rodríguez-Vázquez, and J.-L. Huertas, "Interactive ac modeling and characterization of analog circuits via symbolic analysis", *Kluwer J. Analog Integrated Circuits and Signal Processing*, vol. 1, pp. 183–208, Nov. 1991.

[8] Q. Yu and C. Sechen, "Efficient approximation of symbolic network functions using matroid intersection algorithms", in *Proc. 3rd Workshop on Symbolic Methods and Applications to Circuit Design*, Sevilla, Oct. 1994, pp. 261–227.

[9] Q. Yu and C. Sechen, "Approximate symbolic analysis of large analog integrated circuits", in *Proc. IEEE/ACM ICCAD*, 1994, pp. 664–671.

[10] R. Sommer, E. Hennig, G. Dröge, and E.-H. Horneber, "Equation-based symbolic approximation by matrix reduction with quantitative error prediction", *Alta Frequenza - Rivista Di Elettronica*, vol. 5, no. 6, pp. 317–325, Nov. 1993.

[11] S. Mason, "Feedback theory — some properties of signal flow graphs", in *Proc. IRE*, Sept. 1953, pp. 1144–1156.

[12] C.L. Coates, "Flow-graph solutions of linear algebraic equations", in *IRE Trans. Circuit Theory*, June 1959, vol. 6, pp. 170–187.