Topological Symbolic Simplification for Analog Design*

Hanbin Hu and Guoyong Shi
Dept. of Micro/Nano-electronics,
Shanghai Jiao Tong University, Shanghai 200240, China
huhanbinnew@hotmail.com, shiguoyong@sjtu.edu.cn

Andy Tai and Frank Lee
Synopsys, Inc.
Mountain View, CA 94043, USA
{Andy.Tai,Frank.Lee}@synopsys.com

***Abstract*—Symbolically generated network functions for an analog integrated circuit are complicated in general. For this reason a variety of simplification methods have been proposed in the literature. In this work a novel topology-based symbolic simplification method is proposed, which is capable of generating a simplified symbolic network function together with a simplified small-signal circuit. The technique is developed by applying the recently proposed graph-pair decision diagram (GPDD) algorithm that generates a symbolic network function stored in a binary decision diagram (BDD). Two types of element elimination can directly be operated on such a GPDD data structure. The performance variation by eliminating each symbol from the original circuit is assessed by the means of two monitored response metrics (dc gain and phase margin). After sorting the performance loss, those circuit elements with less performance loss are eliminated, resulting in a reduced GPDD which is automatically a simplified network function. A simplified small-signal circuit is available simultaneously after reduction. Applications to two operational amplifier examples confirm the effectiveness of the proposed methodology.**

***Index Terms*—Behavioral model, binary decision diagram (BDD), graph-paired decision diagram (GPDD), topological simplification, symbolic analysis**

I. Introduction

Simplified small-signal characterization of an analog integrated circuit (IC) is widely used in the practice of analog design. It is commonly handled by manual analysis among analog designers for deriving circuit insights. A simplified circuit model also can be used for cell characterization, behavioral simulation, and hierarchical design verification.

Symbolic analysis is an existing technique that is capable of automatically generating behavioral circuit characterization [1]–[3]. Research on automatic generation of small-signal characterization of an arbitrary analog integrated circuits is still in progress. A variety of the existing simplification methods surround the idea of tailoring symbolically generated results. Pruning could be performed *before*, *during*, or *after* symbolic generation (see [4] and the references therein). Only few works [5], [6] addressed the techniques of simplification *before* generation, because it is harder to prune a circuit without any quantitative reference on the circuit. The authors of [5], [6] proposed to use nullor properties to identify a signal path (SP) from input to output. A nullor is introduced to model the g_m element in each MOSFET. Depending on a circuit in voltage mode or current mode, only one lateral branch of nullor (nullator or norator) is retained throughout the circuit for identifying a SP. This method lacks a theoretical justification, although seeming good examples are presented in [5], [6].

The method of simplification *during* generation is preferred comparing to simplification *after* generation. This is because generating a full set of symbolic terms for large circuits is highly time-consuming. Most of the published methods for simplification *during* generation (such as [7]) only are capable of generating interpretable formulas rather than creating simplified circuits. In principle, creating a simplified circuit requires symbolic manipulation on the circuit topology.

In this work the simplification problem is addressed purely topologically. We make use of a symbolic construction method that creates a symbolic network function represented by a binary decision diagram (BDD) [8]. Among the two recently proposed BDD-based methods [9], [10], the GPDD (graph-pair decision diagram) method preserves topology information in the construction [10]. The GPDD algorithm is intended for performing exact symbolic analysis, and it is fast enough even for large size circuits (typically in a few seconds [10]). Hence, using GPDD for simplification *after* generation is a feasible approach. The key advantage of this approach is that while a GPDD is simplified, simplified symbolic function and simplified circuit are obtained simultaneously. In contrast, the method in [5], [6] only simplifies circuit whereas the simplified network function has to be rederived.

Section II reviews the basic principle of GPDD. Section III presents a detailed procedure for topological simplification. The proposed method is validated by two experimental examples in Section IV. Section V concludes the paper.

II. Review on Graph-pair Decision Diagram

The GPDD algorithm is a BDD-based symbolic construction method which reformulates the two-graph method into a BDD construction process [10]. Although the two-graph method is enumeration-based, it is topological and cancellation-free. The incorporation of BDD makes the enumeration process *implicit* by enforcement of sharing [8]. Hence, the enumeration process is greatly accelerated. Moreover, the enumerated results are saved in a well-organized BDD data structure, on which a variety of post-processing functions can be implemented. By going through the paths in

*This research was supported in part by the National Natural Science Foundation of China under Grants 61176129 and 61474145, and in part by the Joint Research Project sponsored by Synopsys, Inc.

GPDD, we may inspect the effect of collapsing or removing circuit elements (or branches). This is the underlying mechanism for topological simplification.

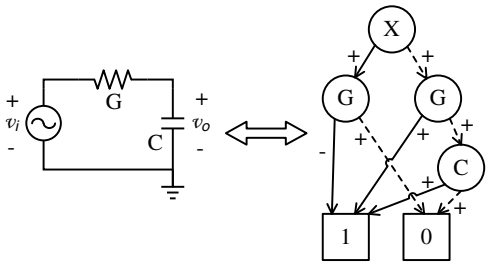

Fig. 1. GPDD Example.

We use a simple example shown in Fig. 1 to illustrate the principle of GPDD. The RC circuit on the left-hand side contains three symbols, G, C, and X, where the symbol X stands for the input-output (I/O) relation, say, $v_i = Xv_o$, where we designate v_i as the controlled source. For such a circuit setup, the GPDD is created as shown on the right-hand side in the figure. The three symbols show up as vertices in the GPDD, with the I/O symbol being at the root. The reader is referred to [10] for the details of GPDD construction.

The symbolic terms of this example are stored as the paths in the GPDD. They are traced as follows: starting from root, go by either arrow emitting from a vertex on the path until the path reaches the terminal "1". If the path goes by a *solid* arrow, the symbol issuing the arrow is *included* in the path; if the path goes by a *dashed* arrow, the symbol issuing the arrow is *excluded* in the path. The signs on the arrows passed by the path multiply together to determine the sign of the term generated by the path. Unlike BDD, in GPDD a solid arrow does multiplication while a dashed arrow does addition. The arrow signs are incorporated in multiplication or addition.

It is easy to see that, because of sharing, a GPDD is capable of storing a large number of paths (i.e., terms) by using a much smaller number of vertexes. One feature of GPDD is that it represents a symbolic function without the need of explicitly enumerating all terms; however, all terms are accessible from the root vertex. Hence, GPDD provides a compact data structure for post-processing the symbolic terms. In addition, a GPDD itself is a computation engine; it performs computation from bottom up. The numerical transfer function values are calculated by a division at the root X.

For the example in Fig. 1, the GPDD represents the following product terms and the transfer function:

$$-XG + (G + sC) = 0 \Rightarrow H(s) = \frac{1}{X} = \frac{G}{G + sC}. \quad (1)$$

III. Topological Simplification

A. GPDD vs Topology

GPDD construction is purely topological; that is, a pair of initial graphs are reduced successively by operating on the element edges. Based on a predetermined element order, a top-down construction process is executed until all circuit elements have been processed. When working on a specific device type, two sets of edge operations are exercised [10]. Typically, only four types of devices suffice for lumped circuit analysis; they are acronymed E, F, G, or H as used by SPICE element types [10]. For application to CMOS analog circuit simplification, it is sufficient to consider G-type devices only, which cover those one-port R and C elements and two-port g_m elements.

Simply speaking, the graphical edge operation associated with a *solid* arrow is just to substitute the G-type element (viewed as a two-port in general) by a nullor. When this element is to be reduced by following a *solid* arrow, the edge operation is to collapse the involved branch(es). That is, in the case of one-port (R or C) the port is short-circuited and in the case of a two-port g_m both ports are short-circuited. Henceforth, this operation is referred to as the *Short* operation.

Oppositely, the edge operation associated with a *dashed* arrow is just to remove the G or g_m element. Henceforth, this type of edge operation is referred to as the *Open* operation. Application of a sequence of *Short* and *Open* operations to a given linear(ized) circuit would result in a simplified circuit as long as the circuit remains valid.

The property described so far for element operation is a unique property owned by GPDD only; DDD [9] does not have an analogous property. Application of this property in a supervised manner leads to a symbolic simplification algorithm.

B. Proposed Simplification Algorithm

The proposed symbolic simplification method is a *supervised* sorting procedure. Several circuit performance metrics are defined and monitored as each *Short* or *Open* operation is applied to a circuit element. In analog design ac metrics are typically more informative. Therefore, we propose to use two metrics for monitoring performance deterioration: *dc gain* and *phase margin*.

The simplification algorithm is described in Algorithm 1. Given a small-signal network $\mathcal{N}$. Let L_S be the list of all symbols. Denote by $H_{\mathcal{N}}(s)$ the transfer function of the original network. Let β_i be an arbitrary symbol from L_S. When the branch(es) of the element β_i are *Short* operated, the resulting transfer function is denoted by $H_{\beta_i}(s)$; when they are *Open* operated, the resulting one is $H_{\bar{\beta}_i}(s)$, for $i = 1, \cdots, |L_S|$.

Line 1 of Algorithm 1 acquires the nominal values of the open-loop gain and the phase at the unity-gain frequency (UGF) ω_U of the original network. Line 3 calculates the open-loop dc gain A_i and phase PM_i after reducing the symbol β_i by *Short* operation, leading to the root mean square $\varepsilon|_{\beta_i}$ of the relative errors calculated in Line 4. Denote by $\varepsilon|_{\bar{\beta}_i}$ the error measure of the *Open* operation applied to the symbol β_i.

Line 6 checks whether the reduction of the current element causes an invalid circuit by inspecting the performance metric value. An invalid reduction is immediately rejected. Algorithm 1 adopts a reduced circuit that alters the monitored performance metrics as little as possible.

Algorithm 1 Algorithm for Topological Simplification

Input:

Initial circuit graph $\mathcal{N}$ and
the number of symbols (K) to be removed.

Output:

Simplified circuit $\widehat{\mathcal{N}}$.

1: $A_0 \leftarrow |H_{\mathcal{N}}(0)|$, $PM_0 \leftarrow \angle H_{\mathcal{N}}(j\omega_U)$;
2: **for all** symbol $\beta_i \in L_S$ **do**
3: $A_i \leftarrow |H_{\beta_i}(0)|$; $PM_i \leftarrow \angle H_{\beta_i}(j\omega_U)$;
4: $\varepsilon|_{\beta_i} \leftarrow \sqrt{\left(\frac{A_i - A_0}{A_0}\right)^2 + \left(\frac{PM_i - PM_0}{PM_0}\right)^2}$;
5: Repeat Lines 3 to 4 for the *Open* operation to the symbol β_i, obtaining $\varepsilon|_{\bar{\beta}_i}$;
6: Keep the symbol β_i if both $\varepsilon_i|_{\beta_i}$ and $\varepsilon|_{\bar{\beta}_i}$ are irregular (INF or NAN); otherwise, determine the operation op_i for reducing β_i leading to the lesser of the two errors, i.e., $\varepsilon_i \leftarrow \min\left(\varepsilon|_{\beta_i}, \varepsilon|_{\bar{\beta}_i}\right)$; save op_i with ε_i;
7: **end for**
8: Sort L_S increasingly by ε_k, $k = 1, \cdots |L_S|$;
9: Remove the first K symbols in L_S from $\mathcal{N}$ by the sequence of operations op_j, $j = 1, \cdots, K$, denote the reduced network by $\widehat{\mathcal{N}}$;
10: **return** $\widehat{\mathcal{N}}$;

It is worth mentioning that GPDD itself is a computation engine. The error evaluations as required by Lines 3 and 5 in the algorithm are carried out directly on an existing GPDD.

IV. Experimental Results

The proposed algorithm has been implemented. We report experimental results on two operational amplifiers (opamps). The first is a folded cascode amplifier shown in Fig. 2. After proper sizing, an operating point analysis by SPICE determines the element values in the small-signal MOSFET model shown in Fig. 3. The GPDD engine reads the small-signal netlist, then uses the numerical small-signal values as the nominal values in simplification.

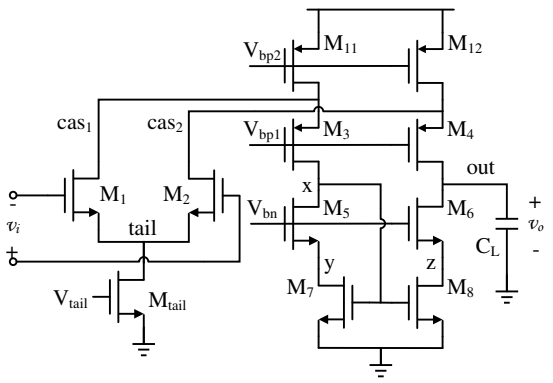

Fig. 2. CMOS folded-cascode operational amplifier.

The *folded cascode* opamp has 11 MOSFETs and 123 symbols in total. After removing 105 elements from the circuit by the algorithm, the reduced small-signal circuit is left with 18 symbols as shown in Fig. 4. It took the program about 3.9 seconds to finish the assessment of the 122 symbols (excluding the I/O symbol X) on a personal computer (Linux virtual machine installation on 64-bit Windows 7, Intel Core i5 CPU, with 2G memory allocated for Linux.)

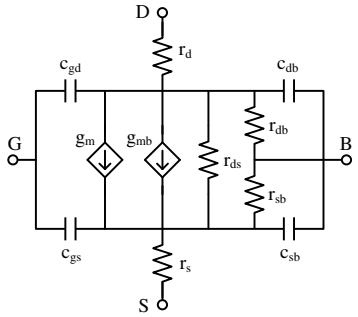

Fig. 3. MOSFET small-signal model.

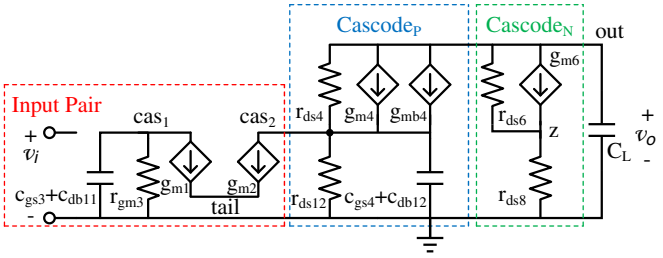

Fig. 4. Simplified circuit for the folded-cascode opamp.

Apparently, the three major blocks, input pair (marked *Input Pair* in Fig. 4), cascode transistor (marked $Cascode_P$) and current mirror (marked $Cascode_N$), are retained after symbolic simplification. The output resistance enhanced by the cascode transistors can be easily observed in the reduced circuit shown in Fig. 4. The input pair is connected to the virtual ground marked by *tail* in the simplified circuit. The dc gain for this circuit derived in [11] is duplicated here:

$$\begin{aligned} A_v &= G_m R_o = -g_{m1}\left\{R_{out,4} || R_{out,6}\right\}, \\ R_{out,4} &= \left\{g_{m4}\left(r_{ds2} || r_{ds12}\right)\right\} r_{ds4}, \\ R_{out,6} &= \left\{g_{m6} r_{ds8}\right\} r_{ds6}. \end{aligned} \tag{2}$$

We see that only r_{ds2} in Eqn. (2) is missing in the simplified circuit while the rest match exactly. Hence, the dc characteristic has been captured by the reduced circuit. A comparison on the ac responses is plotted in Fig. 5, which shows that the simplified circuit approximates the dominant ac response very well.

The second test circuit is a two-stage opamp shown in Fig. 6, which has 7 MOSFETs and 81 symbols in total. The symbol assessment time was only 0.1 second. After simplification, a small-signal circuit with 16 symbols was obtained, as shown in Fig. 7. The dc gain of the reduced circuit matches the following formula derived in [11]:

$$A_v = A_{v1} A_{v2} = -g_{m1}\left(r_{o1} || r_{o3}\right) g_{m6}\left(r_{o6} || r_{o7}\right). \tag{3}$$

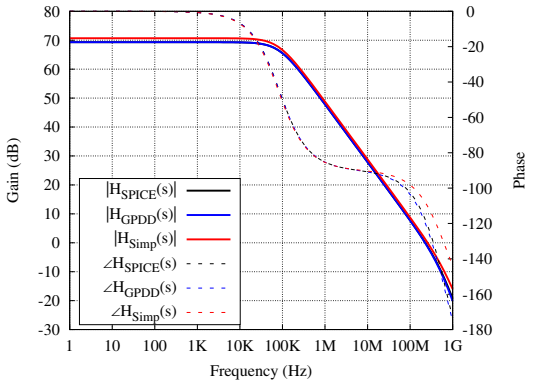

Fig. 5. Comparison of ac responses of the folded cascode amplifier.

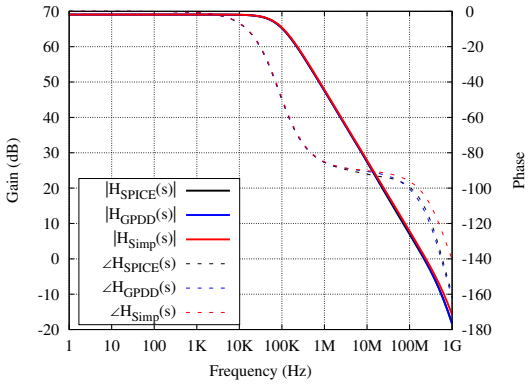

Fig. 8. Comparison of ac responses of the two-stage amplifier.

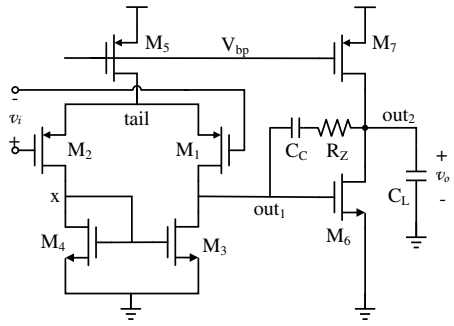

Fig. 6. CMOS two-stage opamp with compensation.

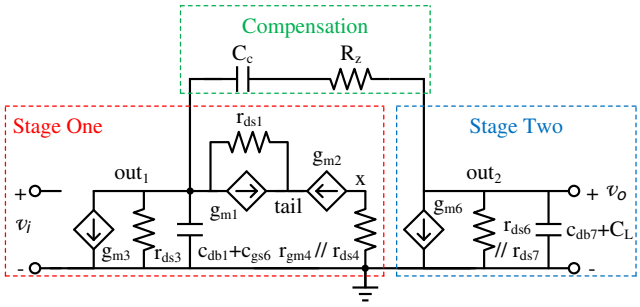

Fig. 7. Simplified circuit of the two-stage opamp.

Fig. 8 shows a comparison on the ac responses.

Some observations are: 1) The compensation elements (C_c and R_z) are retained in the simplified circuit; 2) the virtual ground in the original circuit is preserved; 3) insignificant capacitive elements have been removed; and 4) all the internal nodes in the small-signal MOSFET (Fig. 3) have been eliminated.

With a little more work we may generate the dominant symbolic poles and zeros from the simplified circuit. The complexity is greatly lower than using the method proposed in [12].

V. Conclusion

This paper proposes a new symbolic circuit simplification method that generates a simplified small-signal circuit together with its simplified symbolic transfer function. This method belongs to *simplification after generation*, but has the benefit of generating a simplified circuit topology, which distinguishes it from other traditional methods. The GPDD algorithm plays a fundamental role in the development of the simplification algorithm. Applications have demonstrated the effectiveness of this method. We may further consider auto-generation of slew-settling models by the proposed procedure.

References

[1] P.-M. Lin, *Symbolic Network Analysis*. New York: Elsevier, 1991.

[2] F. V. Fernández, A. Rodríguez-Vázquez, J. L. Huertas, and G. Gielen, *Symbolic Analysis Techniques - Applications to Analog Design Automation*. New York: IEEE Press, 1998.

[3] M. Fakhfakh, E. Tlelo-Cuautle, and F. V. Fernández, *Design of Analog Circuits through Symbolic Analysis*. Oak Park, IL, USA: Bentham Science Publishers, 2012.

[4] P. Wambacq, R. Fernández, G. E. Gielen, W. Sansen, and A. Rodriguez-Vázquez, "A family of matroid intersection algorithms for the computation of approximated symbolic network functions," in *Proc. Int'l Symposium on Circuits and Systems*, 1996, pp. 806–809.

[5] C. Sánchez-López and E. Tlelo-Cuautle, "Behavioral model generation for symbolic analysis of analog integrated circuits," in *Proc. Int'l Symposium on Signals, Circuits and Systems (ISSCS)*, Iasi, Romania, July 2005, pp. 327–330.

[6] ——, "Symbolic behavioral model generation of current-mode analog circuits," in *Proc. Int'l Symposium on Circuits and Systems (ISCAS)*, Taipei, China, May 2009, pp. 2761–2764.

[7] X.-D. Tan and C.-J. R. Shi, "Interpretable symbolic small-signal characterization of large analog circuits using determinant decision diagrams," in *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 1999, pp. 448–453.

[8] G. Shi, S. X.-D. Tan, and E. Tlelo-Cuautle, *Advanced Symbolic Analysis for VLSI Systems: Methods and Applications*. Berlin: Springer, 2014.

[9] C.-J. Shi and X.-D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 1, pp. 1–18, January 2000.

[10] G. Shi, "Graph-pair decision diagram construction for topological symbolic circuit analysis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 2, pp. 275–288, February 2013.

[11] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 4th ed. Singapore: John Wiley & Sons, Inc., 2001.

[12] G. Nebel, U. Kleine, and H. J. Pfleiderer, "Symbolic pole/zero calculation using SANTAFE," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 7, pp. 752–761, July 1995.