Topological Approach to Automatic Symbolic Macromodel Generation for Analog Integrated Circuits

GUOYONG SHI, HANBIN HU, and SHUWEN DENG, Shanghai Jiao Tong University

In the field of analog integrated circuit (IC) design, small-signal macromodels play indispensable roles for developing design insight and sizing reference. However, the subject of automatically generating symbolic low-order macromodels in human readable circuit form has not been well studied. Traditionally, work has been published on reducing full-scale symbolic transfer functions to simpler forms but without the guarantee of interpretability. On the other hand, methodologies developed for interconnect circuits (mainly resistor-capacitor-inductor (RCL) networks) are not suitable for analog ICs. In this work, a topological reduction method is introduced that is able to automatically generate interpretable macromodel circuits in symbolic form; that is, the circuit elements in the compact model maintain analytical relations of the parameters of the original full circuit. This type of symbolic macromodel has several benefits that other traditional modeling methods do not offer: First, reusability, namely that designer need not repeatedly generate macromodels for the same circuit even it is re-sized or re-biased; second, interpretability, namely a designer may directly identify circuit parameters (in the original circuit) that are closely related to the dominant frequency characteristics, such as dc gain, gain/phase margins, and dominant poles/zeros. The effectiveness and computational efficiency of the proposed method have been validated by several operational amplifier (opamp) circuit examples.

CCS Concepts: • **Computing methodologies → Symbolic and algebraic manipulation**; **Modeling and simulation;**

Additional Key Words and Phrases: Analog integrated circuit (IC), automatic model generation, graph-pair decision diagram (GPDD), operational amplifier (opamp), symbolic analysis

ACM Reference Format:
Guoyong Shi, Hanbin Hu, and Shuwen Deng. 2017. Topological approach to automatic symbolic macromodel generation for analog inegrated circuits. ACM Trans. Des. Autom. Electron. Syst. 22, 3, Article 47 (March 2017), 25 pages.
DOI: http://dx.doi.org/10.1145/3015782

1. INTRODUCTION

In analog integrated circuit (IC) design, macromodeling is indispensable. For example, shown in Figure 1 is a representative two-stage operational amplifier (opamp) that contains Miller compensation with nulling resistor (MCNR) [Leung and Mok 2001]. Although there is no optimal design procedure for sizing such a circuit to fulfill all design objectives, reference procedures suggested by experienced designers do exist [Palmisano and Palumbo 1995, 1999; Palmisano et al. 2001; Palumbo and Pennisi 2002]. Inspection on the suggested procedures tells us that compact small-signal models

This research is supported by the National Natural Science Foundation of China (NSFC) under grant 61176129 and grant 61474145.
Authors' addresses: G. Shi, H. Hu, and S. Deng, Department of Micro/Nano-electronics, Shanghai Jiao Tong University; emails: shiguoyong@sjtu.edu.cn, hanbinhu2016@gmail.com, swdeng.betty@outlook.com.

DOI: http://dx.doi.org/10.1145/3015782

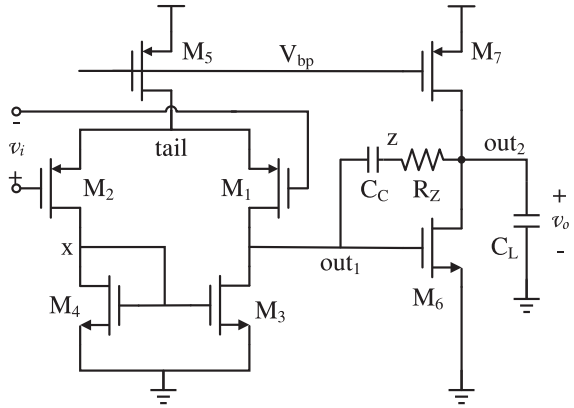


Fig. 1. Two-stage MCNR opamp.

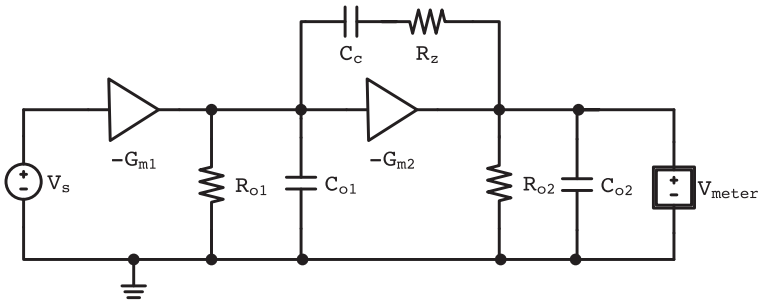


Fig. 2. Small-signal model for the two-stage MCNR opamp.

preserving the transistor-level circuit configuration (conformal abstraction in stages with retained compensation structure) are instrumental in helping designers carry out frequency-domain analysis and derive constraints for device sizing. Shown in Figure 2 is such a reduced-order small-signal model corresponding to the transistor-level circuit in Figure 1.

The main goal of frequency-domain analysis at the macro level is to derive approximate expressions on the frequency characteristics, such as dc gain, bandwidth, gain-bandwidth product (GBW), phase/gain margin, slew rate, and poles/zeros, and so on [Gray et al. 2009]. Analog designers would like to have these metrics in closed form, that is, analytical functions of the compact small-signal model parameters. These expressions then become the fundamental media for the following steps of design, including bias current allocation, constraints on gain, and pole/zero placement, which finally lead to a well-sized circuit for the layout steps.

We call the circuit in Figure 2 a *macromodel* of the original circuit. Such a macromodel has multiple uses in design and simulation, as listed below [Leung and Mok 2001]:

(1) It can be used for deriving frequency response characteristics in closed form.
(2) It can be used for studying the structural configuration of an opamp, such as the function and role of each stage, the effect of frequency compensation, the sources of noise and distortion, and the cause of slew rate limitation, and so on.
(3) It can be used for studying the advanced imperfections caused by the parasitics or nonlinearity that may contribute to secondary effects like doublet, mirror pole, and gate leakage, and so on.

(4) It can be used to explore alternative opamp topology by adding/pruning certain sections or modifying the frequency compensation structure.
(5) It can be used for the system-level simulation as a substitute of the transistor-level circuit.

However, due to the existence of numerous opamp structures, the topology of a derived compact macromodel must change with the original opamp circuit. For this reason, constructing macromodels for an arbitrary opamp structure requires both skills and design insight. Analog IC design is highly knowledge intensive; without good training, creating a proper small-signal circuit model for an arbitrary opamp structure is a challenging task for beginners, especially when connections between the model parameters and the device parameters in the original circuit need to be established.

Therefore, an automation tool that is capable of automatically generating a compact macromodel given a transistor-level opamp circuit is of great help. This art belongs to reduced-order modeling in general. However, since analog ICs are nonlinear circuits, even with small-signal linearization, there is no general methodology in the literature that can be used to generate reduced order macromodels in the form of interpretable circuits. There exist several numerical methods that can deal with nonlinear circuits by generating numerical macromodels, not necessarily in circuit form. However, such methods are mainly useful for the purpose of accelerating numerical simulation, not directly useful for analog IC design.

This article is dedicated to a systematic study on the problem of automatic macromodel generation specifically targeted at analog IC design. We review the related work in Section 2 and motivate a topological approach to the macromodel generation problem. The main contribution of this article is the proposal of a systematic topological method for automatic macromodel generation, whose unique feature consists of applying binary operations to each small-signal circuit element. We tentatively apply each of the binary operations to the relevant circuit branches. If one operation results in a smaller change of the circuit frequency response, then this branch operation is admitted for a candidate reduction operation. All circuit elements are so evaluated and scored. Then we select a portion of the circuit branches with the lowest scores to be operated by their preferred reductive operations. In the end, a reduced-scale compact circuit will be generated, which is expected to be a useful macromodel. The details are presented in Section 3. In Section 4, we formalize an automatic reduced-order model generation algorithm as the main contribution of this work. In Section 5, several representative opamp circuits are used to conduct an in-depth validation of the proposed topological reduction methodology. The validation includes aspects on interpretability of the generated models, accuracy loss in frequency response and poles/zeros, and resilience to sizing change. Finally, this article is concluded in Section 6.

2. RELATED WORK

Macromodeling is an art in many IC design subfields, including analog IC design. Creating proper macromodels for design reasoning or simulation speedup requires skills and insight from the model developers. Automatic macromodel generation for the purpose of analog IC design reasoning has hardly been addressed in the literature so far. Especially when a macromodel is required to preserve the topological feature of the original circuit (like the main stages and the frequency compensation structures in a multi-stage opamp), the existing methods for generating numerical/symbolic macromodels in the literature become less promising. To have an appreciation of the state of the art, we make a brief review on the published works that are relevant to the current work.

Macromodel generation in the domain of analog IC design is mainly to do with the practice of symbolic circuit analysis [Fernández et al. 1998]. During the decades of research on symbolic circuit analysis, generating simplified symbolic results is at the heart of this field. Typically, an exact symbolic expression auto-generated for a frequency response characteristic is too lengthy to read even for an analog IC containing only half a dozen transistors. To make such expressions interpretable, extensive research effort has been devoted to automatically generate simplified symbolic expressions (see Wambacq et al. [1995], Wambacq et al. [1996], Yu and Sechen [1997], Wambacq et al. [1998], Fernández et al. [1998], Katzenelson and Unikovski [1999], Daems et al. [2002], and Tan and Shi [2004], among others). However, hardly any of the listed publications specifically addressed the problem of automatically generating macromodels in a designer readable form. Most of the cited works considered the generation of simplified readable analytical expressions (not circuits!) by dropping a portion of insignificant terms, and the majority of them used numerical reference strategies to realize pruning. By *numerical reference*, it means that the circuit under analysis has to be sized and biased *a priori* to have numerical small-signal parameter values, which are then used to assess the dominance of symbolic terms either during or after symbolic generation. The main limitation of all those symbolic simplification methods is that the machine-generated expressions, although simplified, might not be fully suited to a designer's needs, being either not simple enough or overly simple whereby some secondary effects have been dropped (but not to designer's wish).

As relevant work, we should also mention another line of research from the perspective of model order reduction (MOR). Application of MOR to large-scale linear networks was highly regarded in the circuit community in the mid-1990s. However, a large subset of the proposed methods only applies to the interconnected circuit problems, consisting of only resistor (R), capacitor (C), and inductor (L) elements, even though some of them also consider circuit topology [Ye et al. 2008; Hao et al. 2013]. Although some other reduction methods were targeted at nonlinear analog/radio frequency (RF) circuits [Li and Pileggi 2005; Gu 2011; Liu et al. 2015; Ni et al. 2016]), some of them even considered circuit variations, and the majority of them employ extensions of the *moment matching* methodology by introducing blocked form, quadratic or tensor form, or even zonotoped form moments of dynamic systems. All such methods must rely on highly complicated linear algebra for the computation of approximating subspaces. Unfortunately, the key steps involved in these methods are hardly possible (if not impossible) to be made symbolic, which was already confirmed in the article [Shi et al. 2006].

Analog ICs, even in linearized small-signal form, typically involve dependent sources like the transconductance g_m's and others due to feedback compensation, in addition to R and C elements. Such circuits can be described by a modified nodal analysis (MNA) matrix system, which can then be reduced by a standard MOR algorithm. But this approach only produces a numerical macromodel. Transforming such an approach into a *symbolic* one seems intractable in general [Shi et al. 2006]. Now, as we desire, we would like to generate reduced circuits in interpretable circuit form, that is, topological. It seems that the traditional moment-matching-based methods are incapable of performing topological reduction. Therefore, we have to seek a novel methodology to deal with the problem at hand.

In the authors' opinion, a symbolic analysis tool, as an aid to analog IC designers, must respect designers' habit in their daily practice, because the way they design is a result of long-term training and practice. Analog designers would use a symbolic tool mainly for verifying or validating their manually derived results and intuition. If a tool generates unfamiliar analytical expressions without any discernable connection to circuit-level intuition, then designers would most likely avoid using such a tool. A potentially promising approach is to consider generating simplified results in reduced

circuit form, from which simplified symbolic expressions (if needed) can be derived subsequently. Moreover, the reduced circuits should look familiar to the designers so they can use them to carry out design reasoning.

With the ideal in mind, finding a technical means for realizing the ideal turns out to be nontrivial. Efficiency is probably the most outstanding concern. An evolutionary search strategy might be a candidate, but due to its stochastic nature, the produced macromodels might be overly dependent on the tuning details of the search engine. The preliminary idea of using a symbolic tool to generate a topological macromodel automatically was attempted in the conference article [Hu et al. 2015]. The main purpose of this work is to fully articulate the methodology and explore its potential systematically. We shall report more consolidating experimental results as the validating evidence of the new methodology. We are going to upgrade the macromodel generation method in the following aspects: (1) We fully present the philosophy of topological reduction in a pedagogical way so this methodology can be comprehended easily; (2) we present more in-depth technical details so the mysterious implementation can be better elucidated; (3) we present more convincing validations by providing greater details on circuits with advanced topologies and the model resilience to device size turning, and (4) we also validate the pole-zero property on the reduced circuit models. Most of the presented materials in this work were not available in Hu et al. [2015]. In the next section, the motivational circuit reduction mechanism is presented.

3. BINARY OPERATIONS FOR TOPOLOGICAL REDUCTION

In this work, we introduce a systematic topological network reduction method, which is based on a symbolic construction procedure by applying a sequence of binary operations to the circuit branches. The underlying symbolic construction method is called graph-pair decision diagram (GPDD) [Shi 2013], which treats a linear network as a pair of graphs. The circuit branches, that is, the graph edges, are classified and associated to each other by the symbols (representing the circuit elements). Given an order of the symbols, the graph edges are successively reduced following two deterministic operations (to be explained later). During the course of reduction, any reduced graph pairs that can be shared are identified and shared by a hash mechanism in implementation. Therefore, the GPDD method is so far one of the most efficient non-algebraic symbolic analysis methods. Since this method directly deals with the circuit topology while it structurally manages the whole circuit reduction process with the help of a GPDD data structure, GPDD naturally becomes a promising candidate for topological macromodel generation.

3.1. Motivational Example

We use a simple circuit to explain the key differences between *algebraic* reduction and *topological* reduction. Consider the resistor-capacitor (RC) circuit shown in Figure 3. The transfer function (TF) from the input current source to the output voltage is

$$H(s) = \frac{R_1(1 + R_2C_2s)}{1 + (R_1C_1 + R_1C_2 + R_2C_2)s + R_1R_2C_1C_2s^2}. \tag{1}$$

Choose an arbitrary circuit element, say, R_2. Let us consider two ways of simplifying the circuit by removing R_2. Letting $R_2 = 0$ (i.e., the R_2 branch is shorted in the circuit), we get from the transfer function (1) that

$$H(s)\big|_{R_2=0} = \frac{R_1}{1 + R_1(C_1 + C_2)s}. \tag{2}$$

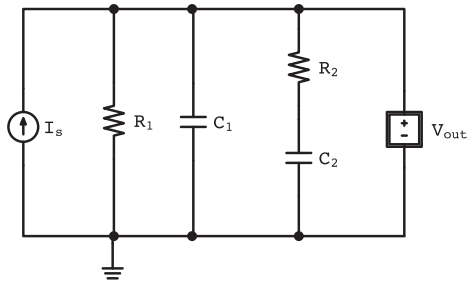


Fig. 3. Simple RC circuit used to illustrate topological reduction.

Then letting $R_2 = \infty$ (i.e., the R_2 branch is open in the circuit), we get from the transfer function (1) that

$$H(s)\big|_{R_2=\infty} = \frac{R_1}{1 + R_1 C_1 s}. \tag{3}$$

Both Equations (2) and (3) are simplification results of the full expression (1) by eliminating R_2. We conclude that, in general, there exist *two* ways of eliminating a symbol (parameter) from an expression (or a circuit).

We observe that the above simplification method requires a known symbolic expression to begin with. In the literature, there are numerous methods for deriving symbolic transfer functions for linear circuit networks. However, not all of them are suitable for implementing both of the algebraic operations as we just performed. Typically, setting a symbol to zero within a symbolic expression is much easier than setting a symbol to *infinity*. The latter operation is a limiting operation, which in general would require an overall manipulation of all symbolic terms. Be aware that in most practical applications, a symbolic tool has to deal with a huge number of symbolic product terms. Without an inherently efficient data structure construction, even sorting a large set of terms would take a significant amount of computation time and resource. Due to this difficulty, almost all of the existing symbolic simplification methods took the approach of *term elimination*, which is roughly equivalent to setting certain symbols to zero. Such operations are much easier to implement given known numerical parameter values.

In circuit theory, we know that setting a linear circuit element to infinity is equivalent to replacing the element by a nullor [Carlin 1964] (just think about an ideal voltage amplifier). As pointed out in Sánchez-López et al. [2011], a network containing nullor or other pathological elements (e.g., current mirror or voltage mirror) can be analyzed using a compact nodal analysis–(NA) based symbolic formulation. However, if we do not know at the beginning whether certain elements should be replaced by nullor or not, then later action on taking limit would become very costly to implement using the method proposed in Sánchez-López et al. [2011]. The key reason is that the NA-based method is not topological.

3.2. A Conceptual Review on GPDD

As a matter of fact, from a theoretical perspective, setting an element value to 0 or ∞ is a dual operation. This is self-evident because both values can be limiting points if we recognize that any symbol can appear either in its original form or in its reciprocal form in a symbolic expression. Therefore, we are confident that a symbolic analysis method must result that can deal with the both limiting operations with equal complexity. To the authors' knowledge, the GPDD algorithm [Shi 2013] is the only existing method that can realize such a dual symbolic construction.

The majority of symbolic methods investigated in the literature use an algebraic formulation, which with little exception adopt the MNA formulation first proposed in Ho et al. [1975]. The famous interactive symbolic analysis of analog circuits (ISSAC) [Gielen et al. 1989, 1990] and determinant decision diagram (DDD) [Shi and Tan 2000] belong to this category. Others following the signal-flow graph (SFG) approach [Lin and Alderson 1973; Daems et al. 2002; Huang et al. 2003] are still algebraic because SFG is an incarnation of an algebraic system. Such formulations have the drawback that singular elements (e.g., nullor) have to be treated in a particular way that differs from a regular element. In the parlance of matrix operation, a nullor is dealt with by row-column collapsing [Vlach and Singhal 1994]. After a matrix/determinant has been expanded like in ISSAC or DDD, subsequent row-column collapse cannot be applied anymore. Although term sorting is possible based on a data structure like DDD, extra cost is inevitable. With GPDD, all the potentially cumbersome steps can be avoided completely because GPDD inherently treats each symbol as two extremal values (0 and ∞) during each step of its construction. This philosophical perspective is the result of new interpretation on the edge operations introduced with GPDD in Shi [2013].

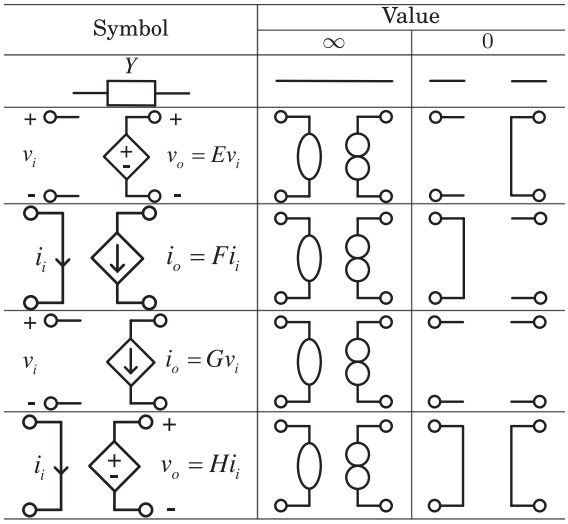
GPDD is the result of reformulation and extension of the classical topological circuit analysis method called the *2-graph* method (see Mayeda [1972]). The original algorithm was a pure enumeration method and hence not suitable for solving mildly large circuits. Moreover, the rules were restricted to g_m (transconductance) and other resistive/conductive elements, not including the E-, F-, and H-type elements in the simulation program with integrated circuit emphasis (SPICE) netlist syntax. In the GPDD work [Shi 2013], these limitations were lifted, and the most significant contribution was to transform explicit enumeration into *implicit* enumeration by applying the powerful binary decision diagram (BDD) technology [Bryant 1986]. With implicit enumeration (i.e., the sharing mechanism), much larger analog circuits can now be solved symbolically by GPDD. Most complementary metal-oxide semiconductor (CMOS) operational amplifier circuits being designed in practice are included in this category.

The process of GPDD construction is essentially an *edge decision* process or, equivalently, a *symbol decision* process. At each construction step, we generate a new pair of collapsed graphs by assuming that the current symbol either takes "0" or "∞." Specific rules map such decisions to the edge operation in the sense of "including" or "excluding" certain graph edges. When operating on one graph edge or a pair of edges, it could be either *removal of edge(s)* or *collapse of edge(s)*, depending on which of the extremal values the symbol takes.

Shown in Table I are circuit representations of the five types of generic circuit elements that are sufficient for small-signal analysis of all analog ICs. The Y element is a representative of R, C, and L that are one-port elements. (In GPDD, all resistive and impedance elements like R and L are treated in their reciprocals, that is, in admittance form.) A theoretical justification of the branch patterns for each type of element can be found in Shi [2013]. What is listed in Table I is an alternative interpretation of the graph reduction rules developed there. We note that for the dependent source elements the branch operations corresponding to the 0 value differ slightly. On the other hand, for the value ∞, the replacement by a nullor is simply equivalent to collapsing the corresponding branches for the reduction operation.

Interpreting the GPDD construction rules graphically, this process is nothing else than successively condensing the whole circuit network down to a trivial single element circuit or an invalid (disconnected) circuit by a sequence of branch operations. On top of that, if we superimpose certain control on the reduction process by adopting those "removal" operations that lead to most negligible change of frequency response, then the macromodel we obtain at the end of the reduction process would be a good approximation of the original circuit. We shall return to the detailed reduction strategy later.

Table I. Circuit Representations of Extremal Symbol Values for Generic *Y*, *E*, *F*, *G*, and *H* Elements



In summary, we have made it clear that circuit reduction essentially can be realized by two extremal branch operations on each of its elements: 0-operation and ∞-operation. While the 0-operation can be implemented in most traditional symbolic methods, implementing the ∞-operation is not necessarily as straightforward. In contrast, with the topological GPDD algorithm, implementing both 0-operation and ∞-operation is of equal complexity without incurring any extra cost. Hence, GPDD becomes our first choice for realizing automatic topological macromodel generation.

3.3. GPDD-Based Reduction

We continue on our working example, the circuit in Figure 3, this time by creating its GPDD. GPDD is a self-contained circuit analysis method that deals with the input-output (I/O) relation as a circuit element as well. When the input is a current source and the output is a voltage measurement, this I/O relation is treated in GPDD as a *voltage-controlled current source*, that is, the H-element in the SPICE netlist syntax. We assign X to be the I/O symbol. Note that the I/O transfer function H is related to X by $H = 1/X$.

In GPDD, all impedance elements are treated as immittance. In this sense, we alternatively use $G_k = 1/R_k$ to denote a resistance. GPDD also automatically lumps parallel elements to reduce the computation. In this example, we see that R_1 and C_1 are parallel. In GPDD, they are processed by a lumped immittance $Y_1 := G_1 + sC_1 = (1 + R_1C_1s)/R_1$. All GPDD symbols (or literals) are ordered. For example, the order $X < C_2 < G_2 < Y_1$ is used in the GPDD constructed in Figure 4. Here the relation "$<$" indicates the symbol precedence.

From the top down, each GPDD vertex has two arrows pointing downward, one solid and one dashed, representing the binary decisions. There are several interpretations on the decisions. In our current context, it is better suited to interpret the "solid" arrow as *the decision symbol taking the value* ∞ and the "dashed" arrow as *the decision symbol taking the value* 0. So, in the rest of the article, we shall interchangeably say that a solid arrow does the ∞-*operation* and a dashed arrow the 0-*operation*. This can be easily justified as equivalent to the "*include*" and "*exclude*" operations in the original GPDD work [Shi 2013].

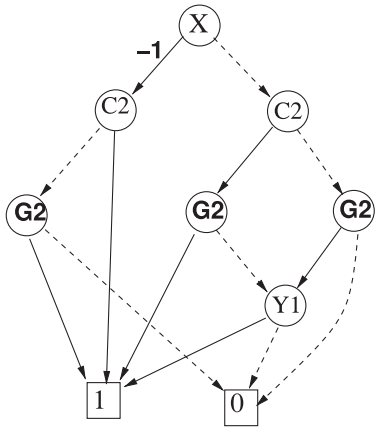


Fig. 4. GPDD created for the circuit in Figure 3 with the symbol order $X < C_2 < G_2 < Y_1$.

The GPDD arrows are attached with *signs*. For example, the solid arrow going out from "X" at the root in Figure 4 has a minus sign. All other arrows not attached with a sign are all positively signed. Readers interested in the detailed construction procedure of GPDD should consult Shi [2013] or Shi et al. [2014]. After a GPDD is constructed, reading out the product terms is straightforward. We follow the paths from the root vertex to the terminal vertex "1" (called the 1-paths). Symbols issuing solid arrows should be included in the terms. When leaving a vertex by a dashed arrow, the current decision symbol is excluded, but the sign of that dashed arrow has to be incorporated for addition or subtraction. Multiplying all the signs seen along a 1-path (terminating at the vertex 1) gives the final term sign.

Since GPDD is a shared binary data structure, it is compact and removes all repeated subexpressions. Hence, all identical subexpressions are constructed only once in a canonical GPDD. Although construction of GPDD for some large analog circuits might take several minutes or more, post-processing on such a data structure is extremely efficient. For example, if we would like to do symbolic generation by assuming that G_2 is infinity, then we have to go by the *solid* arrows only in GPDD when we arrive at that symbol during the path traversal toward the terminal "1." Since letting a symbol take infinity is a *limiting* operation, we infer that those 1-paths not involving G_2 can be ignored completely. Whether or not a 1-path goes through vertex G_2 can be checked by the symbol indexes (which are assigned during the GPDD construction). For instance, the 1-path in Figure 4, X-C_2-1, does not contain G_2, and therefore this path can be dropped during the evaluation of the ∞-operation.

As far as implementation in concerned, the implementation of an ∞-operation in GPDD is very easy. We traverse the path downward. When arriving a vertex C_2, we check whether its solid arrow points to another vertex whose symbol is ordered behind G_2; if yes, then we just terminate the solid arrow of C_2 at "0," indicating that all the sub-paths from this arrow are disabled. Note that GPDD vertices whose solid arrows point at "0" can be eliminated by an operation called *zero-suppression* [Shi 2013]. This is just an operation of redirecting the incoming arrow of C_2 to the dashed-pointer-connected child vertex of C_2. Of course, we have to combine the signs on the relevant arrows.

The ∞-operation on GPDD, though easy, has several useful applications. It can be applied to sensitivity analysis (already studied in Shi and Meng [2009]) and topological reduction in this work. We use two more figures to illustrate the detailed manipulation in GPDD. Figure 5(a) shows that when G_2 is designated an infinity-element, those

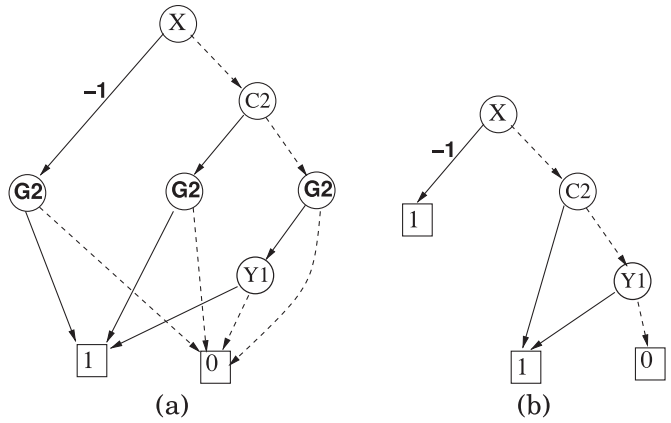


Fig. 5. Reduced GPDD after setting $G_2 = \infty$. (a) Step 1: Paths not containing G_2 are removed. (b) Step 2: Vertexes G_2 are removed.

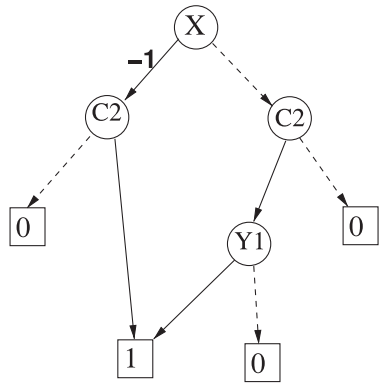


Fig. 6. Reduced GPDD after setting $G_2 = 0$.

1-paths not involving G_2 have been removed (here only one such path). At the next step, we may further remove the G_2 vertexes by redirecting the incoming pointers arriving at the G_2 vertexes to the solid-arrow-connected child vertexes of G_2 (because $G_2 = \infty$). The dashed pointers out-going from the G_2 vertexes become invalid (thus removed). The resulting GPDD is the one shown in Figure 5(b). From this simplified GPDD, we may read out the symbolic transfer function from the sum-of-product (SOP) expression $-X + C_2 s + Y_1 = 0$, that is,

$$H|_{G_2=\infty} = \frac{1}{X} = \frac{1}{C_2 s + Y_1} = \frac{R_1}{1 + R_1(C_1 + C_2)s}, \tag{4}$$

which is identical to Equation (2), because $G_2 = 1/R_2$.

Evaluating a GPDD with a symbol being zero, say, $G_2 = 0$ is much easier to comprehend. Figure 6 shows the reduced GPDD after setting $G_2 = 0$. Once again, we may read the transfer function from the reduced GPDD as follows:

$$H|_{G_2=\infty} = \frac{1}{X} = \frac{C_2 s}{C_2 s Y_1} = \frac{R_1}{1 + R_1 C_1 s}, \tag{5}$$

which is identical to Equation (3).

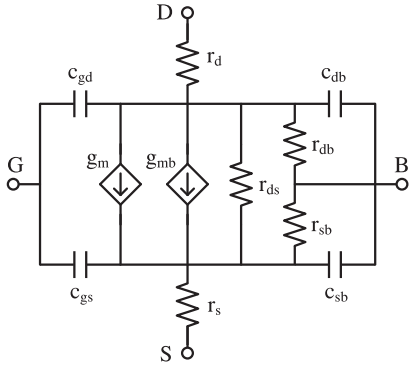


Fig. 7. MOSFET small-signal model.

The working example has demonstrated to us that GPDD is just a right means for realizing elementwise circuit reduction at both the topology and algebraic levels simultaneously. In fact, algebraic simplification is a direct consequence of topological simplification. To the authors' knowledge, there does not exist another symbolic method that can offer the same functionality.

The exposition so far leads us to the following summary on the GPDD binary topological reduction rules:

Rule 1. Reduction by the ∞-operation: When a symbol K is assigned to infinity, all paths not containing K can be early terminated at "0." In addition, the *dashed* arrow of each vertex K is terminated at "0," and, meanwhile, the symbol K is evaluated at $K = 1$.

Rule 2. Reduction by the 0-operation: The simplest way is to substitute $K = 0$ in evaluation. This is equivalent to terminating the *solid* arrow of each vertex K at "0."

We remark that GPDD is a flexible data structure that can be used for topological deduction, numerical evaluation, and other purposes with a great deal of intuition. To fully appreciate the technical details on the GPDD manipulation, we refer the reader to Shi [2013] and Shi et al. [2014] for an in-depth exposition. Here we have re-presented some key ingredients of GPDD that are most pertinent to reduced model generation.

4. AUTOMATIC MACROMODEL GENERATION

In this section, we develop a formal procedure for reducing metal-oxide semiconductor field-effect transistor (MOSFET) analog circuits and generating simplified macromodels automatically. The beginning circuit is a MOSFET analog circuit, in which each MOSFET device has been substituted by its high-frequency small-signal model shown in Figure 7. We call the substituted circuit a *full-order small-signal circuit*. We then apply our in-house GPDD program (written in C++) to construct a GPDD for this full-order circuit. This stage may consume certain amounts of computation time and memory depending on the circuit size (the number of metal-oxide semiconductor (MOS) devices it contains) and the computing equipment. Typically, for an opamp circuit containing 10 to 20 MOS transistors, the GPDD construction time ranges from seconds to minutes on a decent laptop computer. The exact computation time depends on the symbol order and the circuit structure as well. In Shi [2013] some evaluations on the timing and memory consumption were reported. For further larger circuits, hierarchical analysis procedures have been proposed that made use of port connection properties in most amplifier-type circuits [Li et al. 2011]. In this article, we shall pay more

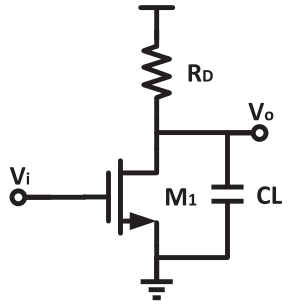


Fig. 8. Single MOSFET amplifier (the substrate of M1 is grounded).

attention to the performance of another program, called the *GPDD-Reduction* program (also written in C++), which is used to reduce a full-size GPDD in order to generate a reduced macromodel topologically.

A full-order small-signal circuit has such a large scale that it is inconvenient to be used directly for design analysis, unless it is used for computer simulation (e.g., SPICE AC analysis). Most of the design analysis and innovation work is based on a reduced-scale circuit, which, depending on the circuit architecture and design needs, can be transformed and altered by the designer [Leung and Mok 2001]. Automatic simplification can save a great amount of manual work during design reasoning and knowledge discovery.

It is well known that in a full-order small-signal circuit a large number of small-signal elements do not play the dominant role (or, in other words, directly affect the circuit response notably). Such element can be eliminated (i.e., removed by opening or shorting) without drastically affecting the frequency response in the main working frequency range. For example, most of the parasitic capacitors/resistors mainly affect the high-order dynamics, that is, those high-order poles and zeros. Higher-order poles and zeros are not the design focus in most analog IC applications. These are the main reasons why a good number of small-signal elements can be pruned to generate a condensed small-signal model without affecting the dominating circuit behavior. This technique is always at the heart of analog IC design. It is also interesting to observe that parasitic elements are *local* to each transistor. Reducing the elements like C_{gd}, C_{gs}, C_{db}, or C_{sb} (by opening) and r_d, r_s, r_{db}, or r_{sb} (by shortening) does not alter the connection of the transistor in circuit. One parasitic element in one transistor does not depend on another element in another transistor. This mutual independence provides us with a convenient way of pruning them after identifying that some of the parasitic elements do not affect the input-output response significantly. Let us illustrate this fundamental concept by use of another example containing a single MOSFET.

Shown in Figure 8 is a common-source single-MOSFET amplifier with a load resistor R_D and a load capacitor C_L. After replacing the n-channel MOSFET by its small-signal model shown in Figure 7, we obtain the linearized circuit shown in Figure 9. After running the GPDD-Reduction program, we obtain the simplified circuit shown in Figure 10. This reduced circuit, although having a smaller number of elements, still preserves the frequency response of the original circuit within the frequency range of interest, as shown in Figure 11. Clearly, the reduced circuit model is better readable and conforms to most designers' intuition.

The original GPDD data structure for the full circuit is too complicated to display, but the reduced GPDD for the simplified circuit can be viewed, see Figure 12, where $G_{mix} := R_{ds} \parallel R_D \parallel C_L$, which is the result of lumping several elements due to reduction. More details of the GPDD construction are listed in Table II. *Num. terms* and *Den. terms*

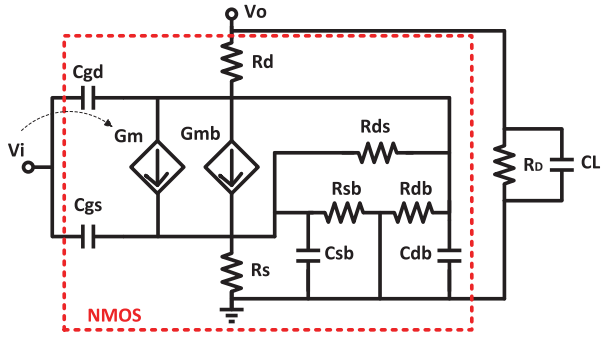


Fig. 9. Full-size small-signal model.

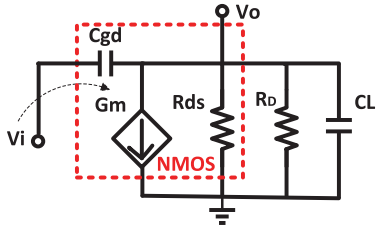


Fig. 10. Simplified circuit for the single MOSFET amplifier.

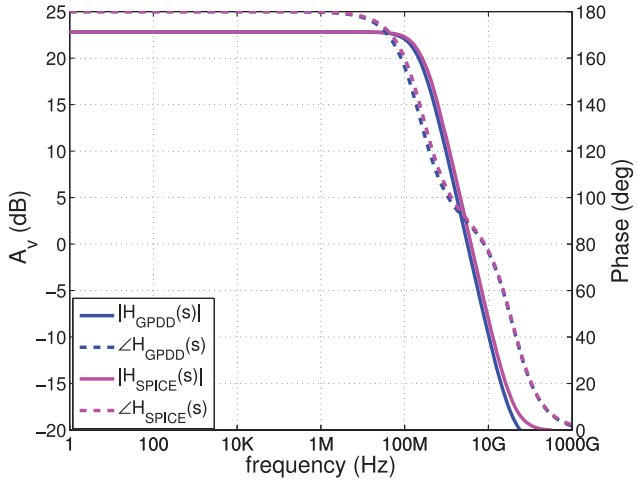


Fig. 11. Comparison of frequency responses before and after reduction.

Table II. Comparison of the GPDD Details between the Full Circuit and the Reduced Circuit

Circuit	Num. terms	Den. terms	GPDD size	Graph edges
Full circuit	6	19	5	17
Reduced circuit	2	2	3	8

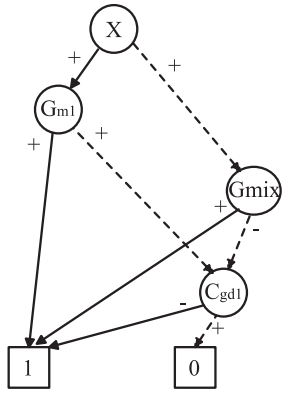


Fig. 12. GPDD structure of the simplified circuit.

stand for the numbers of terms in the numerator and denominator of the generated symbolic transfer functions. *GPDD size* is measured by the number of vertexes in the GPDD data structure. *Graph edges* measures the number of edges representing the circuit.

4.1. Proposed Simplification Algorithm

The proposed symbolic simplification method is a *supervised* truncation procedure. By supervision, we mean that whenever applying an operation (∞-operation or 0-operation) to a circuit element, we check the variation of the frequency response over a preselected frequency range. Since this evaluation is to be repeated for every operation on every element, we need to take into account of computational efficiency for practical use. Therefore, an economic strategy is to select a minimum number of anchor reference points for the evaluation of response deviation. Experiments have convinced us that the *dc gain* and *phase margin* are two characteristic measures adequate for monitoring the ac response perturbation. There are several points that can be attributed to the rationale. First, dc gain is a critical measure of opamp performance that is not allowed to change by macromodeling. Second, the phase margin is defined at the unity-gain frequency (typically referred to as the GBW frequency). Some opamp design work suggests choosing the second pole as a multiple of the GBW (called the separation factor in Palmisano et al. [2001]), meaning that fixing the frequency response up to the point GBW can more or less fix the circuit dynamics up to the second pole. Approximation up to this order of dynamics is usually adequate for analog IC design analysis and macromodeling.

Since an analog IC has to have AC response with certain performance characteristics to begin with, we assume in this article that an opamp to be treated for macromodeling should have been initially sized to exhibit a certain gain and phase margin but not necessarily optimized. This is a reasonable assumption because a condensed small-signal model is typically used to represent the functional operation of the original transistor circuit. If the original circuit has been roughly sized and biased, then transistors that are supposed to be active must have been biased in the active regions. Then insignificant parasitics of MOSFET can be differentiated from those significant small-signal elements. Only under such a condition would a condensed small-signal model make sense.

Let $H(s)$ be the frequency response of the circuit under consideration. It is evaluated at two frequency points, $f = 0$ and $f = f_u$, where f_u is the unity-gain frequency (UGF), that is, satisfying $|H(j2\pi f_u)| = 1$. We denote by $A = |H(0)|$ and $\Phi = \angle H(j2\pi f_u)$,

respectively, the reference dc gain and phase at f_u of the original circuit. The reference frequency response coordinates $(0,\ A)$ and $(f_u,\ \Phi)$ on the Bode plot curves are referred to as the *anchor points*.

Let $\mathcal{S}$ be the set of all symbols of a circuit $\mathcal{N}$. The symbols (i.e., the circuit elements) in this set are listed as $\beta_i \in \mathcal{S}$, $i = 1, \ldots, |\mathcal{S}|$, where $|\mathcal{S}|$ denotes the size of the set $\mathcal{S}$. When we apply the ∞-operation to the element β_i, we obtain a new network whose transfer function is denoted by $H_{\beta_i}(s)$. On the other hand, applying the 0-operation to the element β_i results in another new network with the transfer function $H_{\bar{\beta}_i}(s)$. (Here we slightly abuse the notation by using the symbol name β_i itself for the ∞-operation and $\bar{\beta}_i$ for the opposite.)

As long as $H_{\beta_i}(s)$ and $H_{\bar{\beta}_i}(s)$ are evaluated to normal nonzero numbers at both $f = 0$ and $f = f_u$, we denote by $A_{\beta_i} := |H_{\beta_i}(0)|$ and $\Phi_{\beta_i} := \angle H_{\beta_i}(f_u)$ (similarly for $\bar{\beta}_i$) and compare the deviations of $|A_{\beta_i} - A|$, $|\Phi_{\beta_i} - \Phi|$, and the counterparts with $\bar{\beta}_i$ to see the effect of the binary branch operations. In GPDD any abnormal values (0 or infinity or not a number) of $H_{\beta_i}(s)$ or $H_{\bar{\beta}_i}(s)$ at a frequency point can be detected by checking the GPDD vertexes of the symbol β_i to see whether it is a common factor for both numerator and denominator or either of them. The details will be explained later in the Section 4.2.

In order to be able to compare the severity of deviation after one operation is applied to a symbol, we introduce the following relative translation measure, called the *significance* (or score) of a symbol with the operation β_i or $\bar{\beta}_i$:

$$\varepsilon|_{\beta_i} := \sqrt{\left(\frac{A_{\beta_i} - A}{A}\right)^2 + \left(\frac{\Phi_{\beta_i} - \Phi}{\Phi}\right)^2}. \tag{6}$$

(Replacing β_i by $\bar{\beta}_i$ gives another formula.) The choice of the relative error form is for the convenience of adding the two errors in the Euclidean distance form. Note that both A and Φ at the two anchor points should not be zero if the circuit is working properly.

The rationale of using the two anchor points for assessing the significance of each circuit element is easily explained. If the elimination of the circuit branches corresponding to a symbol (by one of the binary operations) causes a large deviation in the frequency response, then the same effect should be observable from the dc gain and the phase margin. We also discussed that UGF is closely related to the second pole location. If the frequency response up to UGF is relatively untranslated, then it means that the dominant pole positions and hence the phase margin should not be largely perturbed.

The larger a deviation is caused, the more significant an element is. Hence, the circuit elements with higher significance or score are more critical to the circuit and thus deserve to be preserved during simplification. In contrast, those less significant elements can be eliminated, and the specific elimination operation should be the one making smaller perturbation to the AC response. For example, the two series resistors r_d and r_s at the drain and source terminals of a MOSFET (see Figure 7) cannot be opened, because if opened, then the transistor structure would be damaged. But they can be shorted if their effect on the circuit dynamics is weak. GPDD can be used to identify which reduction operation is more appropriate for a parasitic element.

The anchor point strategy was first introduced by the authors in the conference article [Hu et al. 2015], where some preliminary experiments were reported to demonstrate the effectiveness. In this article, we have made more effort on streamlining the presentation and present more test cases to consolidate the validation.

Given a small-signal network $\mathcal{N}$, Algorithm 1 describes the mechanism of the GPDD-Reduction program. This algorithm consists of two phases: the *Scoring Phase* and the *Reduction Phase*. In the first phase, all symbols are scored by evaluating their

ALGORITHM 1: Topological Reduction Algorithm

Input: Small-signal netlist $\mathcal{N}$ and the number of symbols q to be removed.
Output: Simplified circuit $\mathcal{N}_{simp}$.
Run the GPDD program to construct a GPDD for $\mathcal{N}$;
Evaluate $A := |H(0)|$; $\Phi := \angle H(j\pi f_u)$;
for *each symbol* $\beta_i \in \mathcal{S}$ **do**
  Apply the ∞-operation to symbol β_i;
  Evaluate $A_{\beta_i} := |H_{\beta_i}(0)|$;$\Phi_{\beta_i} := \angle H_{\beta_i}(j\pi f_u)$;
  If either of above evaluates to an abnormal number, $\varepsilon|_{\beta_i} := \infty$;
  Else calculate $\varepsilon|_{\beta_i}$ using (6);
  Repeat the above steps for the 0-operation to obtain $\varepsilon|_{\bar{\beta}_i}$;
  Find $\varepsilon_i(op_i) := \min(\varepsilon|_{\beta_i}, \varepsilon|_{\bar{\beta}_i})$; save the significance of β_i with the operation op_i that leads to the smaller relative error;
  Restore the circuit, that is, GPDD;
end
Sort $\mathcal{S}$ in increasing order according to $\varepsilon_k(op_i)$, $k = 1, \cdots |\mathcal{S}|$;
Remove the first q symbols in $\mathcal{S}$ from $\mathcal{N}$ in sequence from the least significance up by applying the operation op_j associated with ε_j for $j = 1, \cdots, q$. Each reduction is monitored by a GPDD evaluation of the error variation. Report error if a large error is observed;
Return the reduced network $\widehat{\mathcal{N}}$;

significance one by one. After finishing one assessment, this symbol is restored in the circuit and we begin the assessment of the next symbol, and so on. Hence, the significance assessment of each symbol is performed independently. Each symbol has two scores, and the operation with a lower score is designated to be the reduction operation in the second phase. If during the scoring stage an operation to a symbol results in a malfunctioning circuit (i.e., GPDD evaluates to an irregular number), then it means that that symbol cannot be operated that way. We assign an infinity significance to this symbol for that operation.

In the second phase, the q symbols of the lowest scores are reduced sequentially. Because of the sequential reduction, we should be cautious about any disruptive behavior, mainly caused by a malfunctioning middle circuit. Fortunately, since the q symbols have the weakest influence to the circuit behavior, we did not observe any malfunctioning middle circuit in our experiment. A reasonable explanation is that all of those lowest scored elements are insignificant parasitic elements local to each MOSFET. As long as they are operated in a proper way (with the least disruption to the whole circuit behavior), a reduction operation would not result in a malfunctioning circuit.

In the next subsection, we slightly discuss some cautions on implementation that may affect efficiency and the generation of readable circuit.

4.2. Cautions in Implementation

GPDD, as a symbolic representation, is a mathematical system that performs algebraic (more preciously polynomial) computation. In addition to generating product terms, it represents a symbolic transfer function in division form. Hence, if there exists a common factor (a symbol), say, K in the numerator, the denominator, or both, setting $K = 0$ or ∞ will cause an abnormal transfer function. Such abnormality might occur both in the scoring phase or during the reduction phase.

The GPDD representation of a symbolic transfer function is derived from the algebraic equation $N \cdot X + D = 0$ at its root, where N is the symbolic function obtained from the solid-arrow pointed child vertex from the root X and D is the other function obtained from the the solid-arrow pointed child vertex from X. If an abnormality occurs in the scoring phase, then it must be due to a symbol K whose branches in the circuit

have connection with the input/output ports. Such K could become a common factor in the function at vertex N or D, causing $N = 0, \infty$ or $D = 0, \infty$. As a result, the transfer function $H(s)$ is evaluated to 0 or ∞ as well. Such a circuit is a malfunctioning circuit and we call the TF value an abnormal value. Whichever case occurs, the associated symbol with that operation is scored infinity, meaning that this operation for that symbol cannot be adopted in reduction.

Another case is to do with a dangling element that results from circuit reduction. A circuit with a dangling element is still a well-functioning circuit, but the dangling element is redundant and should be removed from the circuit. This case can be illustrated by the first example we worked on earlier. We see in the middle step of Equation (5) that the common factor C_2 is eliminated from both numerator and denominator. It indicates that all product terms generated by GPDD must contain that symbol or, equivalently, that all 1-paths in GPDD must pass by that symbol. This is evident in Figure 6. It tells us that C_2 is a dangling element, and it is caused by the previous removal (opening) of the element R_2 from the circuit of Figure 3. Although such a circuit remains well behaved, the element C_2 is redundant and should be removed for compactness. A *common-factor* symbol in GPDD can be identified as follows. First, all paths must go through that symbol. Second, all GPDD vertexes with that symbol must have all of their dashed arrows pointing to "0." All such GPDD vertexes can be suppressed by properly repointing the incoming pointers to their solid-arrow pointed child vertexes (including arrow sign updates).

Some other specialties in implementation are worth mentioning as well. Preprocessing is a helpful means to improve the efficiency of macromodel generation and interpretability. Preprocessing can recognize (1) virtual ground and (2) constant biasing (current or voltage) sources.

Virtual ground exists in differential circuit structure. But we have to implement some rules in a computer program to automatically recognize it. To generate a human readable circuit model, we convert a virtual ground to a real ground by interactive marking in a user interface (or using some special rule). Constant biasing sources refer to those MOSFET devices playing the role of biasing currents or biasing voltages. Such devices can be marked in the user interface as well before reduction or by a simulation-based recognition method. Since these issues are implementation specific, we are not going to further expand on them in this work.

4.3. Discussion on Scalability

Scalability is a concern in all automation techniques involving symbolic analysis. All algebraic or graphical symbolic analysis methods are of exponential complexity if they are requested to produce exact results. Although using a superior algorithm or data structure like BDD can mitigate the complexity to certain degree [Shi 2010], the nature of exponential complexity does not change. Hence, symbolic construction of GPDD cannot be scalable to extremely large analog circuits. But this does not prevent us from applying this technology to analog IC design automation, because, due to the continuous dedicated research effort, current symbolic circuit analysis techniques can cope with exact analysis of a large class of analog ICs, including those most commonly seen as opamps [Shi et al. 2014]. One should not ignore the power of the current state of the art of this technology simply due to its lack of scalability.

It is well known that even for small-scale analog ICs like those most common opamps, design automation in the sense of starting from a high-level description to a well-sized circuit ready for layout is still in its infancy. Powerful scalable SPICE simulators cannot fulfill the role of the synthesis needs like automatic generation of macromodel in circuit form with symbolic model parameters. For such reasons, even algorithms with very limited scalability are of practical value, as long as they can be applied to a

certain class of analog circuits whose design effort still involves a great deal of manual work today.

Since the proposed topological reduction algorithm has to start from an *a priori* constructed GPDD, it implies that the beginning circuit must not be excessively large. The original work on GPDD [Shi 2013] has demonstrated that most commonly seen opamps (containing about 20 CMOS transistors) are within the capacity of GPDD. A later work [Li et al. 2011] has further demonstrated that *hierarchical analysis* is a feasible means for extending the GPDD capacity to even larger analog ICs.

However, for macromodel generation, it is worth mentioning that a simpler strategy can be used to address the scalability challenge. The strategy is by *divide and conquer*. Because the proposed model generation method is topological, it allows us to partition a given circuit *a priori* into several smaller parts topologically. This approach can be most beneficial for those opamps in multi-stage form, because their connections are in the form of cascading ports added with several frequency compensation paths [Leung and Mok 2001]. For topological reduction, we typically choose to keep all the compensation elements in a multi-stage circuit intact while reducing the rest of the circuit by segmented stages.

5. EXPERIMENTAL VALIDATION

This section is dedicated to a comprehensive validation on the proposed macromodel generation method. In Hu et al. [2015], we tested the reduction method on two opamp circuits, the two-stage MCNR opamp (see Figure 1) and a folded-cascode opamp. Due to space limitation, the experiment was limited to a simple verification. In this section, we plan to present a more factual report on the multiple aspects regarding the reduced macromodel generation. Interested issues include the following: (1) effectiveness when applying the method to opamp circuits with more involved structure like two-stage opamp with voltage buffer (VB) compensation and current buffer (CB) compensation, (2) robustness of the method in the sense of varying device sizes, and (3) pole-zero check before and after reduction. These facts were not reported in Hu et al. [2015]. The robustness issue is of special interest, because in real design practice MOSFET device sizes are constantly modified to tune the circuit performance. We do not want an auto-generated model to be valid with one sizing case but not with another.

The experiment is divided into three parts. In the first part, we apply the GPDD-Reduction program to the two-stage opamp with two compensation structures, one with the VB compensation shown in Figure 13 and the other with the CB compensation shown in Figure 14. Because several additional MOSFET devices are added to the circuit for compensation, we attempt to see whether the reduction program can capture the compensation structure in human readable form. In the second part, we report the test results on using the reduction program for an opamp circuit with three different sizing sets. In the third part, we present the effect of reduction from the perspective of pole-zero behavior.

5.1. Part I: Test on the VB- and CB-Compensated Opamps

Opamps with VB or CB compensation were studied in several publications [Palmisano and Palumbo 1997; Palmisano et al. 2001; Mahattanakul 2005]. Manually derived small-signal models were presented in these works. The auto-generated small-signal macromodels after running the GPDD-Reduction program are presented in Figures 15 and 17, respectively, for the VB-compensated and CB-compensated opamps. The program only generated simplified netlists. We manually drew the circuits for human reading. As can be seen, the reduced circuits preserve the original circuit structures (two stages with proper compensation). We see from the reduced models that both VB and CB compensations essentially inject current to the compensation paths, which play

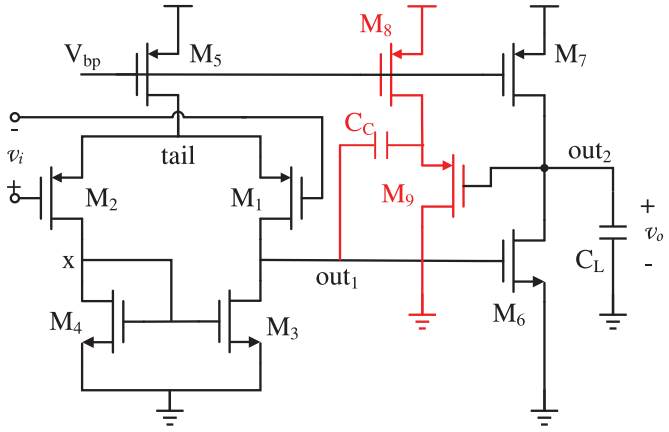


Fig. 13. Two-stage opamp with the voltage buffer (VB) compensation.

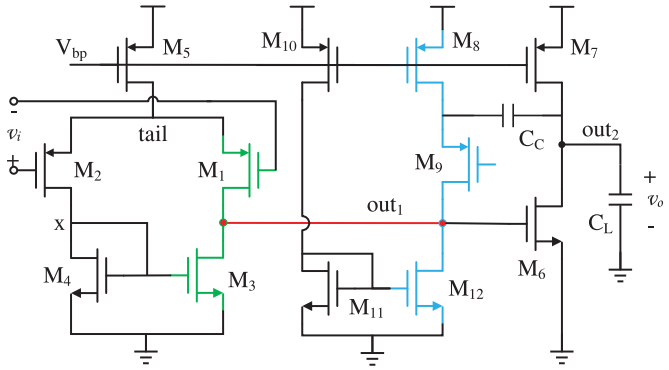


Fig. 14. Two-stage opamp with the current buffer (CB) compensation.

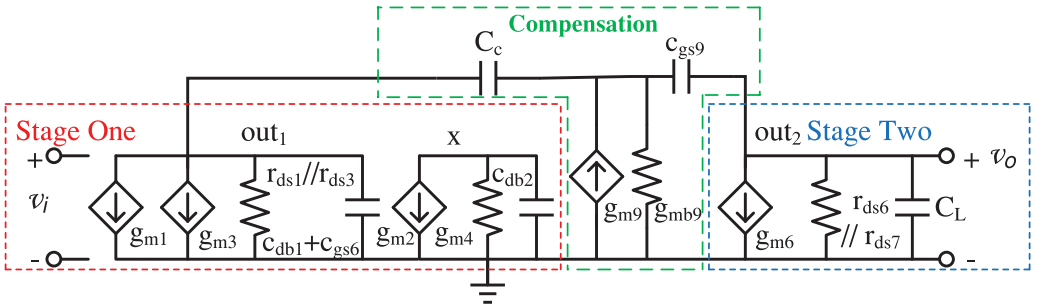


Fig. 15. Generated macromodel for the VB-compensated opamp.

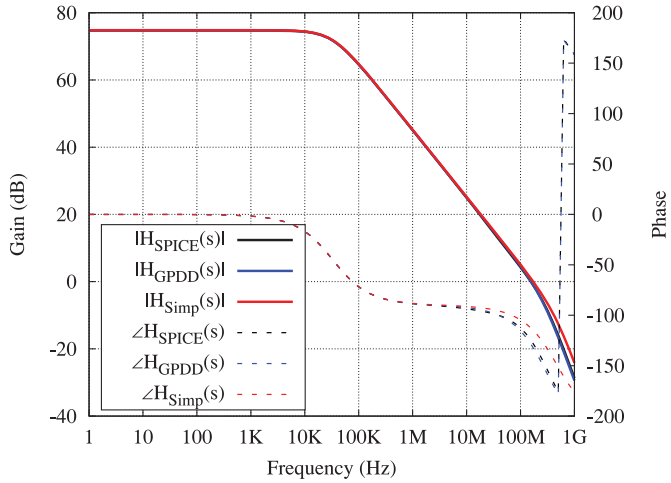


Fig. 16. Frequency response comparison for the VB-compensated opamp.

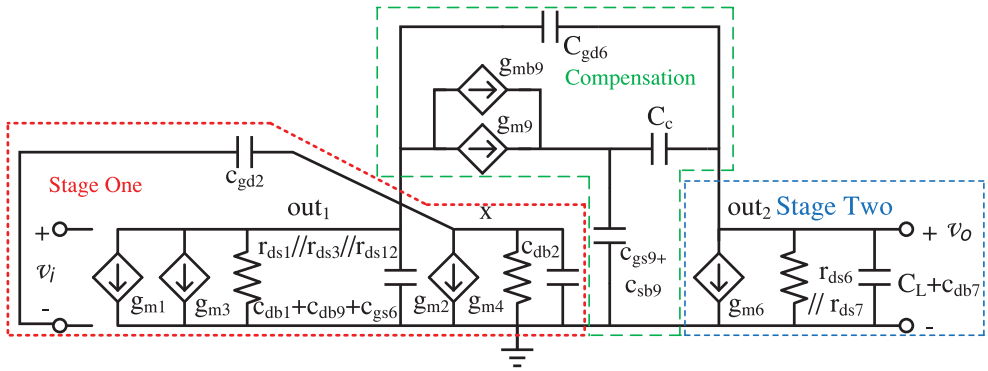


Fig. 17. Generated macromodel for the CB-compensated opamp.

the role of adjusting one of the zero paths, fulfilling the goal of relocating the dominant left-half plane zero. In addition, the annotated device parameters in the reduced models clearly show the effect of parasitics of the original circuits. When we derive the pole-zero expressions from these reduced circuits, the effect of those parasitics on the poles and zeros can be identified with great intuition.

To check the accuracy of the reduced models, we ran numerical AC analysis of both full and reduced circuits. The frequency responses are compared in Figure 16 for the VB-compensated opamp and in Figure 18 for the CB-compensated opamp. We only see minor discrepancies due to the negligence of the high-order effects in the reduced models.

As we mentioned earlier, GPDD is a shared data structure that can be used for numerical AC response evaluation. Except for relatively longer times for initial construction (seconds to a few minutes), the numerical evaluation time for sorting all small-signal parameters (including many repetitions for significance assessment) is minor, typically less than 1s. Hence, time cost is not a big concern.

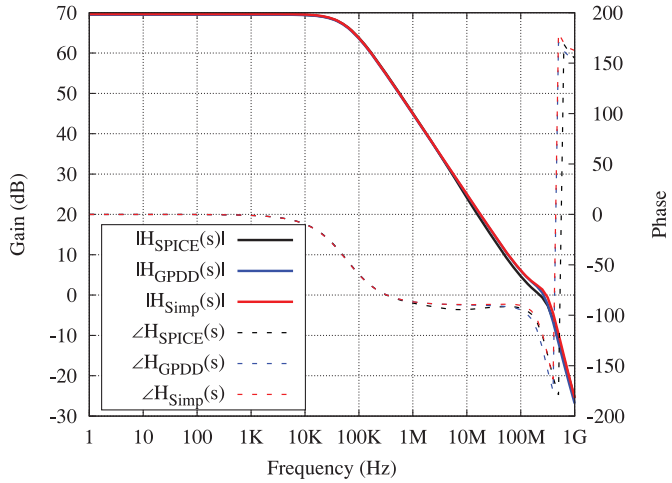


Fig. 18. Frequency response comparison for the CB-compensated opamp.

Table III. Circuit Setting for the Two-Stage MCNR Opamp

Parameter	Value
Biasing voltage V_{bp}	1.12V
Load capacitor C_L	2pF
Channel length L	0.5μm
Supply voltage Vdd	1.8V

Table IV. Three Sizing Sets for the Two-Stage MCNR Opamp

Parameter	Case 1	Case 2	Case 3
DC input voltage	0.9V	0.7V	1.1V
$W(M_1)$	20μm	40μm	25μm
$W(M_2)$	20μm	40μm	25μm
$W(M_3)$	5μm	1.5μm	1μm
$W(M_4)$	5μm	1.5μm	1μm
$W(M_5)$	48μm	12μm	10μm
$W(M_6)$	30μm	33.5μm	32μm
$W(M_7)$	144μm	144μm	144μm
R_z	1.75kΩ	1.75kΩ	1.5kΩ
C_c	350fF	350fF	250fF

5.2. Part II: Test on the Robustness of the Macromodel Generation Method

In this section, we test another important issue; that is, if a macromodel has been generated from a sized opamp circuit, then we slightly adjust the sizes of the MOS devices and regenerate the macromodel. Would the macromodel generated in the second round differ significantly from the first round? This question is relevant to the robustness of an auto-generated macromodel. As a matter of fact, the model robustness is to do with the anchor points selection during element assessment. We anticipate that, as long as resizing does not largely change the frequency response behavior, the auto-generated macromodel should not differ greatly.

To test the claim, we used the two-stage MCNR opamp shown in Figure 1 for experiment. This circuit was biased as the setting given in Table III. Three sizing and biasing sets listed in Table IV were considered. The macromodels generated by the GPDD-Reduction program are drawn in Figure 19 for the sizing cases 1 and 3 (turned out to

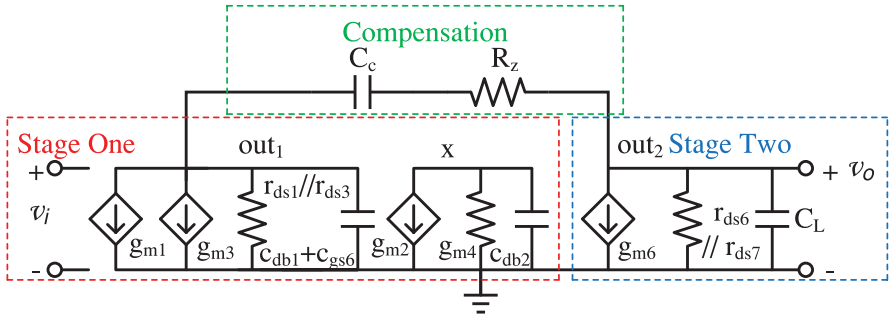


Fig. 19. Generated macromodel for the two-stage MCNR opamp with the sizing set of Case 1 in Table IV.

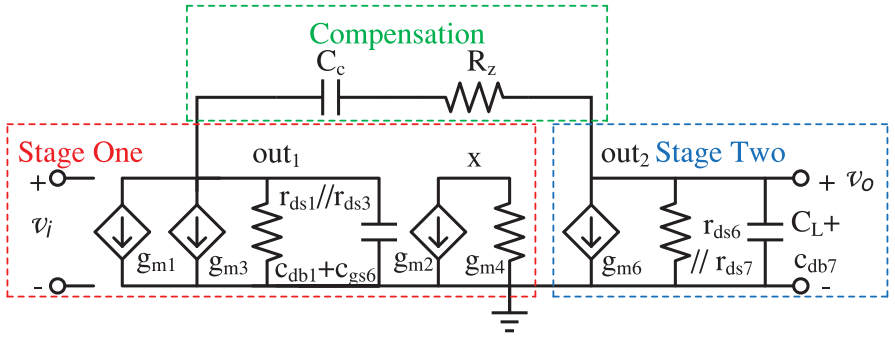


Fig. 20. Generated macromodel for the two-stage MCNR opamp with the sizing set of Case 2 in Table IV.

Table V. Circuit Reduction Effect in Terms of Poles and Zeros

Circuit	Original		Reduced	
	# Poles	# Zeros	# Poles	# Zeros
MCNR opamp	6	6	4	2
VB-comp opamp	6	6	3	2
CB-comp opamp	7	7	4	2

be identical) and Figure 20 for the sizing case 2. The majority of the circuit elements in the two reduced models are identical, except for two minor parasitic capacitors; C_{db2} is connected as the output capacitor of the first stage in Figure 19 and C_{db7} is connected as the output capacitor of the output stage in Figure 20. Such a minor difference can be ignored in design practice, unless these parasitics need special attention.

5.3. Pole-Zero Check

Since the reduced macromodels are mainly for design use, and poles and zeros are important design means, it would be of interest to inspect how a reduced model would change them.

First, Table V shows the change of the numbers of poles and zeros before and after reduction for the three test circuits. The reduction in numbers is as expected. Table VI further shows the details of pole-zero locations for the MCNR opamp with the sizing set of *Case 1* given in Table IV. We see interestingly that the reduced macromodel has automatically taken care of pole-zero cancellation and disregarded some high-order poles and zeros. The dominant poles and zeros of interest in design are captured.

Table VI. Pole-Zero Comparison for Two-Stage MCNR Opamp

	Full circuit	Reduced circuit
Poles (Hz)	$-24.62K$	$-25.64K$
	$-19.41M$	——
	$-374.1M \pm 65.11M$	$-375.8M$
	$-537.4M \pm 99.56M$	$-554.0M$
		$-631.9M$
Zeros (Hz)	$-21.25M$	——
	$-298.0M$	$-298.2M$
	$-418.8M$	——
	$-1.018G$	$-1.264G$
	$3.845G$	——
	$49.83G$	——

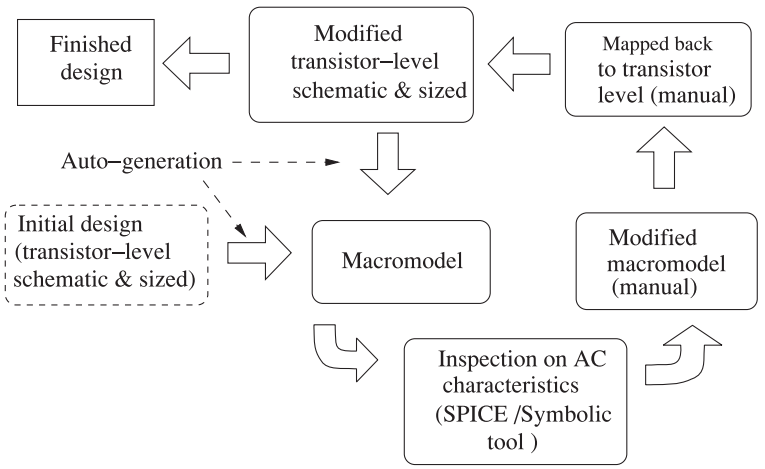


Fig. 21. Illustration of analog circuit design cycle.

5.4. Discussion on Potential Applications

Due to the space limitation, we are not able to explore further in this publication on the applications of auto-generated macromodels. We anticipate that this novel methodology can have many potential applications in analog IC design. The main benefit of using this tool is the turn-around time. A user can quickly obtain an auto-generated macromodel without going through lengthy manual procedures.

Analog IC design is most often an iterative process. A designer may start from a sketch with only a few transistors and then gradually add extra transistors to the schematic to acquire performance enhancement (so-called design reasoning). Hence, in automatic macromodel generation, we may assume that one circuit is already well sized with acceptable metrics. We generate a macromodel using GPDD-Reduction for such a circuit. If we need to add other components to this circuit, then we may manually add extra small-signal components to the auto-generated model and then go back to construct another transistor-level circuit by modifying the former one. This design cycle is illustrated in Figure 21.

In this cycle, the step of "macromodeling" is always there. Hence, undoubtedly, an automatic model generation tool can greatly speed up the process of design reasoning and knowledge discovery.

6. CONCLUSION

This article has formalized a systematic procedure for generating symbolic macromodels in topological form for analog integrated circuits. The procedure is based on directly manipulating the circuit topology by applying the GPDD algorithm, thereby the generated macromodel still takes an interpretable circuit form, which can be used for design reasoning and exploration. Analog IC design is a knowledge-intensive practice. Currently smart design automation tools are still rare, and, hence, more research is necessary to develop helpful tools to enhance learning and innovation in this area. In the upcoming publications, we shall address applications of this technology in advanced designs such as studying various mismatch effects, gate leakage issues, and subthreshold design, and so on.

ACKNOWLEDGMENT

The anonymous reviewers' comments have greatly improved the readability and technical presentation of this article.

REFERENCES

R. E. Bryant. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.* 35, 8, 677–691.

H. J. Carlin. 1964. Singular network elements. *IEEE Trans. Circ. Theory* 11, 3, 67–72.

W. Daems, G. Gielen, and W. Sansen. 2002. Circuit simplification for the symbolic analysis of analog integrated circuits. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 21, 4, 395–406.

F. V. Fernández, O. Guerra, D. J. Roddríguez-Garcia, and A. Roddríguez-Vázquez. 1998. Symbolic analysis of large analog integrated circuits: the numerical reference generation problem. *IEEE Trans. Circ. Syst. II: Analog Dig. Sig. Process.* 45, 10, 1351–1361.

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

G. E. Gielen, H. Walscharts, and W. Sansen. 1989. ISAAC: A symbolic simulator for analog integrated circuits. *IEEE J. Solid-State Circ.* 24, 6, 1587–1596.

G. E. Gielen, H. Walscharts, and W. Sansen. 1990. Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE J. Solid-State Circ.* 25, 3, 707–713.

P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer. 2009. *Analysis and Design of Analog Integrated Circuits* (5th ed). John Wiley & Sons, Inc., New York, NY.

C. Gu. 2011. QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 30, 9, 1307–1320.

Z. Hao, G. Shi, S. X. D. Tan, and E. Tlelo-Cuautle. 2013. Symbolic moment computation for statistical analysis of large interconnect networks. *IEEE Trans. VLSI Syst.* 21, 5, 944–957.

C. W. Ho, A. E. Ruehli, and P. A. Brennan. 1975. The modified nodal approach to network analysis. *IEEE Trans. Circ. Syst. CAS-22,* 6, 504–509.

H. Hu, G. Shi, A. Tai, and F. Lee. 2015. Topological symbolic simplification for analog design. In *Proceedings of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*. 2644–2647.

X. Huang, C. S. Gathercole, and H. A. Mantooth July. 2003. Modeling nonlinear dynamics in analog circuits via root localization. *IEEE Trans. Comput-Aid. Des. Integr. Circ. Syst.* 22, 7, 895–907.

J. Katzenelson and A. Unikovski. 1999. Symbolic-numeric circuit analysis or symbolic circuit analysis with online approximations. *IEEE Trans. Circ. Syst. I: Fundam. Theory Appl.* 46, 1, 197–207.

K. N. Leung and P. K. T. Mok. 2001. Analysis of multistage amplifier – frequency compensation. *IEEE Trans. Circ. Syst. I: Fundam. Theory Appl.* 48, 9, 1041–1056.

P. Li and L. T. Pileggi. 2005. Compact reduced-order modeling of weakly nonlinear analog and RF circuits. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 23, 2, 184–203.

X. Li, H. Xu, G. Shi, and A. Tai. 2011. Hierarchical symbolic sensitivity computation with applications to large amplifier circuit design. In *Proceedings of the International Conference on Circuits and Systems.* 2733–2736.

P. M. Lin and G. E. Alderson. 1973. Computer generation of symbolic network functions – a new theory and implementation. *IEEE Trans. Circ. Theory* 20, 48–56.

H. Liu, L. Daniel, and N. Wong. 2015. Model reduction and simulation of nonlinear circuits via tensor decomposition. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 34, 7, 1059–1069.

J. Mahattanakul. 2005. Design procedure for two-stage CMOS operational ampplifers employing current buffer. *IEEE Trans. Circ. Syst. I: Express Briefs* 52, 11, 766–770.

W. Mayeda. 1972. *Graph Theory*. Wiley-Interscience, New York, NY.

L. Ni, S. P. D. Manoj , Y. Song, C. Gu, and H. Yu. 2016. A zonotoped macromodeling for eye-diagram verification of high-speed I/O links with jitter and parameter variations. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 35, 6, 1040–1051.

G. Palmisano and G. Palumbo. 1995. An optimized compensation strategy for two-stage CMOS op amps. *IEEE Trans. Circ. Syst. I* 42, 3, 178–182.

G. Palmisano and G. Palumbo. 1997. A compensation strategy for two-stage CMOS op amps based on current buffer. *IEEE Trans. Circ. Syst. I* 44, 3, 257–262.

G. Palmisano and G. Palumbo. 1999. Analysis and compensation of two-pole amplifier with a pole-zero doublet. *IEEE Trans. Circ. Syst. I 46,* 7, 864–868.

G. Palmisano, G. Palumbo, and S. Pennisi. 2001. Design procedure for two-stage CMOS transconductance operational amplifiers: A tutorial. *Analog Integr. Circ. Sign. Process.* 27, 179–189.

G. Palumbo and S. Pennisi. 2002. Design methodology and advances in nested-Miller compensation. *IEEE Trans. Circ. Syst. I: Fundam. Theor. Appl.* 49, 7, 893–903.

C. Sánchez-López, F. V. Fernández, E. Tlelo-Cuautle, and S. X. D. Tan. 2011. Pathological element-based active device models and their application to symbolic analysis. *IEEE Trans. Circ. Syst. I: Regul. Pap.* 58, 6, 1382–1395.

C. J. R. Shi and X. D. Tan. 2000. Canonical symbolic analysis of large analog circuits with determinant decision diagrams. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 19, 1, 1–18.

G. Shi. 2010. Computational complexity analysis of determinant decision diagram. *IEEE Trans. Circ. Syst. II: Express Briefs* 57, 10, 828–832.

G. Shi. 2013. Graph-pair decision diagram construction for topological symbolic circuit analysis. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 32, 2, 275–288.

G. Shi, B. Hu, and C. J. R. Shi. 2006. On symbolic model order reduction. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 25, 7, 1257–1272.

G. Shi and X. Meng. 2009. Variational analog integrated circuit design by symbolic sensitivity analysis. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*. T3002–3005.

G. Shi, S. X.-D. Tan, and E. Tlelo-Cuautle. 2014. *Advanced Symbolic Analysis for VLSI Systems—Methods and Applications*. Springer, New York.

S. X. D. Tan and C. J. R. Shi. 2004. Efficient approximation of symbolic expressions for analog behavioral modeling and analysis. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 23, 6, 907–918.

J. Vlach and K. Singhal. 1994. *Computer Methods for Circuit Analysis and Design* (2nd ed). Van Nostrand Reinhold Company, New York, NY.

P. Wambacq, P. Dobrovolny, G. E. Gielen, and W. Sansen. 1998. Symbolic analysis of large analog circuits using a sensitivity-driven enumeration of common spanning trees. *IEEE Trans. Circ. Syst. II: Analog Dig. Sign. Process.* 45, 10, 1342–1350.

P. Wambacq, R. Fernández, G. E. Gielen, W. Sansen, and A. Rodriguez-Vázquez. 1995. Efficient symbolic computation of approximated small-signal characteristics. *IEEE J. Solid-State Circ.* 30, 3, 327–330.

P. Wambacq, R. Fernández, G. E. Gielen, W. Sansen, and A. Rodriguez-Vázquez. 1996. A family of matroid intersection algorithms for the computation of approximated symbolic network functions. In *Proceedings of the International Symposium on Circuits and Systems*. 806–809.

Z. Ye, D. Vasilyev, Z. Zhu, and J. R. Phillips. 2008. Sparse implicit projection (SIP) for reduction of general many-terminal networks. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 736–743.

Q. Yu and C. Sechen. 1997. Efficient approximation of symbolic network functions using matroid intersection algorithms. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 16, 10, 1073–1081.

Received August 2016; revised October 2016; accepted November 2016