Received: 25 August 2017 | Revised: 20 February 2018 | Accepted: 27 March 2018

DOI: 10.1002/cta.2495

RESEARCH ARTICLE

WILEY

Symbolic analysis of electric networks with higher order summative cofactors and parameter decision diagrams

Sławomir Lasota

Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Institute of Electronics, ul.Akademicka 16, 44-100 Gliwice, Poland

Correspondence
Sławomir Lasota, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Institute of Electronics, ul. Akademicka 16, 44-100 Gliwice, Poland.
Email: slasota@polsl.pl

Funding information
Polish Ministry of Science and Higher Education, Grant/Award Number: 8686/E-367/S/2016

Summary

The paper introduces the concept of higher order summative cofactors (HOSCs) to the circuit analysis. Although the concept is not new, it is not well known. In the paper, some mathematical background of HOSCs is presented. The further development of the concept of HOSC will yield computer implementation arithmetic of HOSC. A cancellation-free symbolic analysis technique, which is based on HOSC arithmetic, is presented. This technique allows results to be created directly from a netlist in the form of a binary decision diagram, which is called a parameter decision diagram. Additionally, HOSC arithmetic allows the calculation to be started in many places (sometimes distant) simultaneously. The techniques of *rolling up* the already analyzed parts of a circuit, which is built into HOSC arithmetic, result in a novel multilevel hierarchical analysis method that is called hierarchical parameter decision diagram (HPDD). Unlike in most hierarchical methods, the results that are obtained based on the subcircuit representation in HPDD always maintain a cancellation-free form. The HPDD always represents the sum of the product form, which is heavily compressed due to the self-similarities of the actual circuit. The time that is required for any recalculation of the transfer functions is greatly reduced. Analysis of models that are based on pathological components is also a natural consequence of using HOSC arithmetic.

KEYWORDS

hierarchical analysis, higher order summative cofactors arithmetic, MNA method, pathological components, symbolic analysis, 2-graph method

1 | INTRODUCTION

Symbolic analysis derives the analytical characterization of the behavior of a circuit in terms of the circuit's parameters. Symbolic simulation results are more instructive to designers than numerical ones. However, the number of symbolic terms in a circuit function increases in an exponential manner with an increasing number of nodes and branches in a circuit.[1] This is true for the methods described in Lin,[1] if one expects to have the exact results in the form of the sum of products. For the past few years, research on symbolic analysis has been dedicated to creating different methods to circumvent this intrinsic disadvantage of symbolic analysis, and some progress has been made. However, no one has proposed any new analysis engine that could be added to those described in Lin.[1] Although in Gielen et al,[2] Wambacq et al,[3] and Guerra et al,[4] the authors introduced some simplification methods, and they used the classical 2-graph method proposed in Mayeda.[5] In Shi and Tan,[6-8] a determinant decision diagram (DDD) was introduced that was based

on the modified nodal admittance (MNA) matrix (MNAM) method that had been introduced in Ho et al.[9] In previous studies,[10-15] graph pair decision diagrams (GPDDs) were presented, which are also based on the 2-graph method improved by the admissible tree-pairs concept from Yin.[16] In previous works,[17-21] the authors developed a simplified symbolic analysis with an active devices model that was based on the pathological components: nullators, norators, and current and voltage mirrors. Nevertheless, they still used classical analysis methods. When reading the literature on symbolic analysis (eg, Shi[22]), one has the impression that researchers today are divided into proponents of the 2-graph method or the MNAM.

Both methods were created to solve the same problem, and it is obvious that they are related to each other. This paper will demolish this artificial division between them by introducing a new approach to symbolic analysis—creating a parameter decision diagram (PDD) with higher order summative cofactors (HOSCs) (hereafter called a PDD with an HOSC). The method is derived from the genuine nodal admittance matrix (NAM) method to become a topological one, which is related to the 2-graph method with some modifications. In the presented method, no matrices need to be created. The result can be obtained directly from the netlist of the components and their topological position in the circuit. The analysis consists of extracting the components from the netlist one by one and considering them in the result until the netlist is empty. Unlike the well-known DDD,[6-8] which operates on the MNAM entries and naturally suffers from cancellations, the PDD operates directly on the actual parameters of the components and are always cancellation free. This method links together topological methods that are based on the 2-graph approach with methods that are based on NAMs into one. Moreover, the presented method is free of some inconvenience of the MNA and graph pair decision diagram (GPDD) methods.

In contrast to the genuine NAM approach, no components are incompatible with the method. We can freely use any controlled source. Any 2-terminal can be represented as an admittance or impedance. Applying the pathological components in the analysis is a natural extension of this method. This always reduces the size.

Generally, selecting the components in an analysis can be done in an arbitrary order. We can start an analysis with all of the capacitors, coils, and coupled coils to obtain the transfer function as the rational function of the *s*-expanded polynomials directly. We could also start with some frequently changing parameters, as well, to accelerate the numerical recalculation of network functions. Each recalculation is done from the top of a PDD until only the intact components are left. Generally, if an operation has already been performed, it is not repeated. The results from a specially tailored system of *caches* are reused.

Creating a PDD with an HOSC produces a hierarchical method in a natural way. The hierarchical method is not similar to any other method that is presented in the literature. In the hierarchical PDD (HPDD) analysis method, the subcircuit representation is created in almost the same manner as the PDD for the transfer function. The unified representation of the PDD that is in progress and the components means that any piece of the circuit can also be treated as a *meta-component*. In any analysis, it is considered in a way that is similar for a primitive component. An analysis can start from more than one part of a circuit, even those that are distant. It can also be performed in a parallel manner as separate threads of a program after which the parts can be linked together to form a representation of the larger circuit, just as primitive components can be done. Although the hierarchy can also be multilevel, each result that is in the form of PDD always represents the sum of terms, although in a compressed form and is still cancellation free.

The results presented here are extensions of the results from Lasota and Malcher[23] and Lasota.[24-29] However, the algorithms that are presented here have been significantly modified and improved.

The rest of paper is organized as follows. In Section 2, some mathematical background of HOSC and HOSC arithmetic are defined. The application of HOSC in multi-input multi-output (MIMO) circuit analysis is presented in Section 3. Section 4 is devoted to the symbolic determination of HOSC and the creation of the PDDs. In Section 5, the application of PDDs and some unusual analysis methods of large circuits are presented such as the uncommon hierarchical method of analysis (HPDD) and the direct generation of transfer functions in an *s*-expanded form. Both solutions are not easy to discover and implement without HOSCs and PDDs. The algorithms and methods are illustrated using numerous examples. During algorithm creation, the impact for the future implementation in parallel and/or distributed computations was emphasized. In Section 6, there are some conclusions.

2 | THEORETICAL FOUNDATIONS

2.1 | Higher order cofactors

In contrast to numerical methods, each classical symbolic analysis problems is reduced to a determination of some set of transfer functions.[1] Furthermore, nowadays, both of the most popular method, which are based on the MNAM or the

2-graph approach,[22] in practice are reduced to the application of Cramer rule to an (M)NA set of equations. To determine the jth nodal voltage component that is stimulated by a current that is forced to the node i, one should determine $v_{i\to j} = \Delta_j^i/\Delta{\cdot}I_j$, where Δ is a determinant of the whole NAM, while Δ_j^i is the so-called cofactor of the matrix, ie, the determinant of the matrix with the ith row and jth column removed (*deleted*) and multiplied by $(-1)^{i+j}$. However, quite often, the output signal can be different from the nodal voltage drop, and the input stimulus signal can be different from the current that is forced between a node and the reference node. The difference between the methods of the symbolic analysis is a consequence of the different algorithms that are used to determine the cofactors and the preprocessing and postprocessing that are done to obtain the desired transfer functions. Cofactors in the form Δ_j^i are the most commonly known type of cofactors. However, more generalized form of cofactors exists in mathematics.

Cofactors were introduced to allow the Laplace rule of determinant expansion to be written along the r_1th row or the c_1th column in a compact form: $\Delta = \sum_{c_1=1}^{n} a_{c_1 r_1}{\cdot}\Delta_{r_1}^{c_1} = \sum_{r_1=1}^{n} a_{c_1 r_1}{\cdot}\Delta_{r_1}^{c_1}$. On the other hand, a cofactor is a signed minor, and a minor is a determinant of the submatrix. Thus, each cofactor in the form $\Delta_{r_1}^{c_1}$ can be developed along row $r_2 \neq r_1$ or column $c_2 \neq c_1$ according to the Laplace rule: $\Delta_{r_1}^{c_1} = \sum_{c_2=1, c_2\neq c_1}^{n} a_{c_2 r_2}{\cdot}\Delta_{r_1,r_2}^{c_1,c_2} = \sum_{r_2=1, r_2\neq r_1}^{n} a_{c_2 r_2}{\cdot}\Delta_{r_1,c_2}^{c_1,c_2}$. The third-order cofactor is obtained in a similar recurrent manner. Generally, such a recurrent procedure yields the kth-order cofactors. The formal definition can be found in Gantmacher.[30] However, for the practical application, the following rules of thumb can be considered:

1. If in $\Delta_{j_1,j_2,\ldots,j_k}^{i_1,i_2,\ldots,i_k}$ there are *multiple deletions* (ie, some row or column is removed more than once), then a cofactor must always be equal to 0, because such a cofactor is illegal.
2. If in $\Delta_{j_1,j_2,\ldots,j_k}^{i_1,i_2,\ldots,i_k}$ both *sets of deletions* $i_1, i_2, \ldots, i_k$ and $j_1, j_2, \ldots, j_k$ are in an ascending order (or both are in a descending order), then $\Delta_{j_1,j_2,\ldots,j_k}^{i_1,i_2,\ldots,i_k} = (-1)_{j_1+j_2+\cdots j_k}^{i_1+i_2+\cdots i_k}{\cdot}\boldsymbol{M}_{j_1,j_2,\ldots,j_k}^{i_1,i_2,\ldots,i_k}$, where $\boldsymbol{M}_{j_1,j_2,\ldots,j_k}^{i_1,i_2,\ldots,i_k}$ is the determinant of the matrix with rows $i_1, i_2, \ldots, i_k$ and columns $j_1, j_2, \ldots, j_k$ removed (so-called minor). The minor of the matrix with all rows and columns removed is always equal to 1.
3. If removed rows or columns are not already ordered, they should be ordered by exchanging the order pair by pair; however, each swapping affects the sign, ie, $\Delta_{j_1,\underline{j_2},\ldots,\underline{j_k}}^{i_1,\underline{i_2},\ldots,\underline{i_k}} = -\Delta_{j_1,\underline{j_k},\ldots,\underline{j_2}}^{i_1,\underline{i_k},\ldots,\underline{i_2}}$. Rather than the expensive reordering, a determination of the sign of permutation can be performed. Another useful method is inserting the *pairs of deletions* pair by pair to always maintain the descending order. If the distance of the inserted *deletions* for the rows and columns is odd, the sign must be changed.

Additionally, the higher order cofactors have 2 additional features that are crucial for the method presented here:

1. The order of the *pairs of deletions* can be arbitrary, ie, $\Delta_{j_1,\underline{j_2},\ldots,\underline{j_k}}^{i_1,\underline{i_2},\ldots,\underline{i_k}} = \Delta_{j_1,\underline{j_k},\ldots,\underline{j_2}}^{i_1,\underline{i_k},\ldots,\underline{i_2}}$.
2. The relation that is true for $\Delta_{j_1,j_2,\ldots}^{i_1,i_2,\ldots}$ is true as well for $\Delta_{j_1,j_2,\ldots,j_k}^{i_1,i_2,\ldots,i_k}$.
3. The cofactors with a number of *pairs of deletions* that exceed the size of the matrix are always 0.

The concept of higher order cofactors is not well known among the scientists who are working on the problem of symbolic analysis. In Chen,[31] the author proposed the second-order cofactors. However, there is no simple rule to generalize his definition for a cofactor that has a higher order than 2. In Tan et al[32] and Tan,[33] the authors tried to use the higher order cofactors to determine the cancellation-free parameters of the subcircuits from an MNAM in a DDD form. However, it was too involved and was abandoned by the authors themselves. The higher order cofactors operate on the matrix entries. They cannot explore the specific structure of the NAM that describes the electric network. We need to introduce *summative cofactors*. Although they were introduced in Sigorskij,[34] the book has never been translated into English. Thus, this concept is still not very well known. There are only a few papers on the subject in English. It can be found in Lasota and Malcher,[23] Lasota,[24-29] Sigorskij and Petrenko,[34] and in some papers by Russian scientists.[35,36] Removing rows or columns is a special case of a more advanced operation.

2.2 | Higher order summative cofactors

Let us have 2 cofactors of the same matrix. Let these cofactors have the almost same *set of deletions* except for one pair. Let these pairs differ only with the *deletion* of rows. We can present them as $\Delta_C^{R',a}$ and $\Delta_C^{R',b}$, and a and b are the only

deletions in rows that are different. Let us find the result of the addition or subtraction of these cofactors. By expanding $\Delta_C^{R',a}$ along row b, we obtain $\Delta_C^{R',a} = \sum_{i=1}^{n} a_{bi} \cdot \Delta_{C,i}^{R',a,b}$, and $\Delta_C^{R',b}$ along row a, we obtain $\Delta_C^{R',b} = \sum_{i=1}^{n} a_{ai} \cdot \Delta_{C,i}^{R',b,a}$. Adding or subtracting both cofactors yields $\Delta_C^{R',a} \pm \Delta_C^{R',b} = \sum_{i=1}^{n} a_{bi} \cdot \Delta_{C,i}^{R',a,b} \pm \sum_{i=1}^{n} a_{ai} \cdot \Delta_{C,i}^{R',b,a} = \sum_{i=1}^{n} a_{bi} \cdot \Delta_{C,i}^{R',a,b} \mp \sum_{i=1}^{n} a_{ai} \cdot \Delta_{C,i}^{R',a,b}$ $\sum_{i=1}^{n} (a_{bi} \mp a_{ai}) \cdot \Delta_{C,i}^{R',a,b} = \Delta_C^{R',(a \mp b)}$. Instead of adding or subtracting the 2 cofactors, we can determine only one cofactor. A *deletion* in the form $(a + b)$ means "add column a to column b and then remove column a," while a *deletion* in a form $(a - b)$ means "subtract column a from column b and then remove a". The same relations can be formed for the columns, as well.

$$\Delta_C^{R',a} \pm \Delta_C^{R',b} = \Delta_C^{R',(a \mp b)}, \tag{1}$$

$$\Delta_{C',a}^{R} \pm \Delta_{C',b}^{R} = \Delta_{C',(a \mp b)}^{R}. \tag{2}$$

Hereafter, *deletions* in the form $(a \pm b)$ will be called *positive/negative summative deletions*. Let a and b be called a *source* and *target* row/column, respectively. Any higher order cofactor with at least one *summative deletion* will be called a *higher order summative cofactor*—an HOSC in short.

An HOSC can be used for any kind of square matrix. However, henceforth, we are going to focus only on an NAM. A complete NAM is always singular.[1,31] Thus, it cannot be used to solve any circuit in a direct way. We should choose one node to be the reference node and remove the row and column that is associated with this node. Then a determination of the voltage drop in relation to the reference node (the nodal voltage) can be made. Usually, the reference node obtains the number zero, and it is the same as the grounded node. However, it can be chosen arbitrarily, if there is some specific reason. Thus, row and column number 0 can be treated as existing in the matrix, but are always deleted. This makes the *deletion* of row or column number 0 acceptable. However, such a cofactor should always be equal to zero (ie, $\Delta_{C,0}^{R,a} = \Delta_{C,b}^{R,0} = 0$).

An HOSC has the following main features:

1. Any summative *pair of deletions* can be developed according to the rules:

$$\Delta_{C,(c+d)}^{R,(a+b)} = \Delta_{C,c}^{R,a} + \Delta_{C,d}^{R,b} - \Delta_{C,d}^{R,a} - \Delta_{C,c}^{R,b}, \tag{3}$$

$$\Delta_{C,(c+d)}^{R,(a-b)} = \Delta_{C,c}^{R,a} - \Delta_{C,d}^{R,b} - \Delta_{C,d}^{R,a} + \Delta_{C,c}^{R,b}, \tag{4}$$

$$\Delta_{C,(c-d)}^{R,(a+b)} = \Delta_{C,c}^{R,a} - \Delta_{C,d}^{R,b} + \Delta_{C,d}^{R,a} - \Delta_{C,c}^{R,b}, \tag{5}$$

$$\Delta_{C,(c-d)}^{R,(a-b)} = \Delta_{C,c}^{R,a} + \Delta_{C,d}^{R,b} + \Delta_{C,d}^{R,a} + \Delta_{C,c}^{R,b}. \tag{6}$$

2. Each cofactor, even basic one, can be presented as an HOSC, eg,

$$\Delta_{C,(b+0)}^{R,(a+0)} \equiv \Delta_{C,(c+0)}^{R,(a-0)} \equiv \Delta_{C,(c-0)}^{R,(a+0)} \equiv \Delta_{C,(c-0)}^{R,(a-0)} \equiv \Delta_{C,b}^{R,a}. \tag{7}$$

3. The order of the *pairs of deletions* can be arbitrary. However, the exchange of some *deletions* only between rows or only between columns changes the sign

$$\Delta_{\ldots,(e+f)(g+h)}^{\ldots,(a+b)(c+d)} = -\Delta_{\ldots,(e+f)(g+h)}^{\ldots,(c+d)(a+b)} = -\Delta_{\ldots,(g+h)(e+f)}^{\ldots,(a+b)(c+d)} = \Delta_{\ldots,(g+h)(e+f)}^{\ldots,(c+d)(a+b)}. \tag{8}$$

4. A *complete* HOSC is always equal to 1

$$\Delta^{(1+0)(2+0)(3+0),\ldots,(n+0)}_{(1+0)(2+0)(3+0),\ldots,(n+0)} \equiv 1, \tag{9}$$

where n is the number of nodes except the reference node and 0 is the reference node.

5. In each *summative deletion*, the *source* and *target* can be exchanged. However, for a *positive summative deletion*, it changes the sign:

$$\Delta^{R,(a+b)}_{\ldots} = -\Delta^{R,(b+a)}_{\ldots}, \tag{10}$$

$$\Delta^{R,(a-b)}_{\ldots} = \Delta^{R,(b-a)}_{\ldots}. \tag{11}$$

6. If 2 HOSCs only differ by one *deletion* in the rows or in the columns, we can transform their sum into

$$\Delta^{R,(a+b)}_{C} \pm \Delta^{R,(c+d)}_{C} = \Delta^{R,(a\mp c)}_{C} - \Delta^{R,(b\mp d)}_{C}, \tag{12}$$

$$\Delta^{R,(a+b)}_{C} \pm \Delta^{R,(c-d)}_{C} = \Delta^{R,(a\mp c)}_{C} - \Delta^{R,(b\pm d)}_{C}, \tag{13}$$

$$\Delta^{R,(a-b)}_{C} \pm \Delta^{R,(c+d)}_{C} = \Delta^{R,(a\mp c)}_{C} + \Delta^{R,(b\pm d)}_{C}, \tag{14}$$

$$\Delta^{R,(a-b)}_{C} \pm \Delta^{R,(c-d)}_{C} = \Delta^{R,(a\mp c)}_{C} + \Delta^{R,(b\mp d)}_{C}. \tag{15}$$

Formulae (12) to (15) are generalization of (1). They play a crucial role in the structural synthesis of circuits and the creation of the idealized models of modern active circuits.[29]

Manipulating an HOSC can be more involved and difficult than manipulating ordinary cofactors. If the number of the *pairs of deletions* is large, the decision, whose rows and columns remains, which is added or subtracted to remaining ones, and even the detection of *multiple deletions* are not so obvious. Quite often, an HOSC that has *multiple deletions* at the first glance can be transformed into a valid form. Even the detection of the same HOSC, but in a different form, is not straightforward. For example, for any 6×6 matrix, there are *multiple deletions* in the HOSC $\Delta^{(4+0)(6+0)(1+3)(3+4)(2+6)(5+6)}_{(1+2)(3+5)(1+3)(3+4)(2+6)(5+6)}$, and therefore, it is equal to 0. However, the slightly different HOSC $\Delta^{(4+0)(6+0)(1+3)(3+4)(2+6)(5+6)}_{(1+2)(3+5)(1+3)(3+4)(2+0)(5+6)}$ is identical to $-\Delta^{(1+0)(2+0)(3+0)(4+0)(5+0)(6+0)}_{(1+0)(2+0)(3+0)(4+0)(5+0)(6+0)}$, and as a complete cofactor, it is equal to -1. Although, we obtain the correct results by applying Formulae (3) to (7), such an approach is much too involved and impractical. Fortunately, an alternative method can be proposed.

2.3 | HOSC arithmetic and its computer implementation

For the uniqueness of seemingly different HOSCs, let us assume that each HOSC is stored in a so-called *canonical* form.

Definition 1. The HOSC $\pm\Delta^{(a_1\pm b_1)(a_2\pm b_2)\ldots(a_k\pm b_k)}_{(c_1\pm d_1)(c_2\pm d_2)\ldots(c_k\pm d_k)}$ has a *canonical* form, if

1. both lists of the number of *deleted* rows $a_1, a_2, \ldots, a_k$ and columns $c_1, c_2, \ldots, c_k$ are in a descending order and each a_i and c_i are unique;
2. for each i, $b_i < a_i$, and there is no j that $b_i = a_j$; similarly for each i, $d_i < c_i$, and there is no j that $d_i = c_j$.

The HOSC $\Delta_{(1+2)(3+5)(1+3)(3+4)(2+0)(5+6)}^{(4+0)(6+0)(1+3)(3+4)(2+6)(5+6)}$ is not *canonical*. However, $-\Delta_{(1+0)(2+0)(3+0)(4+0)(5+0)(6+0)}^{(1+0)(2+0)(3+0)(4+0)(5+0)(6+0)}$ is its *canonical* equivalent. To avoid any ambiguity during an analysis, we should reduce each HOSC to a *canonical* form as soon as it is possible. For an effective transformation, some additional rules should be defined for an HOSC.

The following expressions can be proved:

$$\Delta_{\dots}^{\dots,(\underline{a}+\underline{a})} = 0, \tag{16}$$

$$\Delta_{\dots}^{\dots,(\underline{a}-\underline{a})} = 2\cdot\Delta_{\dots}^{\dots,(a+0)}, \tag{17}$$

$$\Delta_{\dots}^{\dots,(a+b)(a+c)} = \Delta_{\dots}^{\dots,(a+c)(b+c)}, \tag{18}$$

$$\Delta_{\dots}^{\dots,(a+b)(c+a)} = \Delta_{\dots}^{\dots,(a+b)(c+b)}, \tag{19}$$

$$\Delta_{\dots}^{\dots,(a+b)(a-c)} = \Delta_{\dots,}^{\dots,(a-c)(b-c)}, \tag{20}$$

$$\Delta_{\dots}^{\dots,(a+b)(c-a)} = \Delta_{\dots}^{\dots,(a+b)(c-b)}, \tag{21}$$

$$\Delta_{\dots}^{\dots,(a-b)(a+c)} = -\Delta_{\dots}^{\dots,(a+c)(b-c)}, \tag{22}$$

$$\Delta_{\dots}^{\dots,(a-b)(c+a)} = \Delta_{\dots}^{\dots,(a-b)(c-b)}, \tag{23}$$

$$\Delta_{\dots}^{\dots,(a-b)(a-c)} = -\Delta_{\dots}^{\dots,(a-c)(b+c)}, \tag{24}$$

$$\Delta_{\dots}^{\dots,(a-b)(c-a)} = \Delta_{\dots}^{\dots,(a-b)(c+b)}. \tag{25}$$

Their proofs are trivial. Formulae (16) and (17) are direct applications of (1). For the formulae (18) to (25), both the left and right sides should be developed according to (3) to (6), and then any HOSC with *multiple deletions* should be deleted. Both sides are equal.

Remark 1. Although (18) to (25) are 6 formulae, they can be reduced to one simple rule. Let us present it with Equation (24). The left side is $\Delta_{\dots}^{\dots,(a-b)(a-c)}$. Node a is apparently removed twice. Thus, the form is not *canonical*. Considering *deletion* $(a - c)$, we know from the previous pair that a has been subtracted from b. We substitute $a := -b$, and we obtain $\Delta_{\dots}^{\dots,(a-b)(-b-c)}$. The form with $-b$ is formally illegal, because removing a row or column with a negative number is meaningless. However, this is a side effect of the addition and subtraction of HOSCs. Thus, we can remove the "−" sign before the HOSC, and we obtain $-\Delta_{\dots}^{\dots,(a-b)(b+c)}$. It is still *noncanonical*, because a is subtracted from b, which was moved to c and removed in the next step. We should substitute $b := c$ in the rest of the *deletions*. Finally, we obtain $-\Delta_{\dots}^{\dots,(a-c)(b+c)}$. This rule allows us to construct a simple algorithm for the transformation from a *noncanonical* to *canonical* form of the HOSC.

Algorithm 1. Always-canonical HOSC maintenance

Let $sgn = 1$. Let us declare 2 lists—$\boldsymbol{R}$ for rows and $\boldsymbol{C}$ for columns, respectively. Each element of the list is a pair (s_i, t_i), which means that s_i has been moved to t_i. If t_i is a negative number, it means that s_i has been moved into t_i with the opposite sign. The list is always ordered in ascending order according to s_i. To determine $[s_j]$ ($\boldsymbol{C}[s_j]$), the algorithm looks for pair $s_i = s_j$ and returns t_i. If there is no $s_i = s_j$, it simply returns s_j.

For any HOSC, the *pair of deletions* $\binom{p+r}{k+l}$ is added one by one. The algorithm for maintaining the canonicity of an HOSC is

1. Determine from lists $\boldsymbol{R}$ and $\boldsymbol{C}$ $a := \boldsymbol{R}[p]$, $b := \boldsymbol{R}[r]$, $c := \boldsymbol{C}[k]$ and $d := \boldsymbol{C}[l]$.
2. If $a = b$ or $c = d$ then the result is 0. Break the further analysis.
3. If $a = -b$ then $b := 0$, $sgn * = 2$.
4. If $c = -d$ then $d := 0$, $sgn * = 2$.
5. If $a<|b|$ change $a \Leftrightarrow b$ and $sgn = -sgn$.
6. If $c<|d|$ change $c \Leftrightarrow d$ anfd $sgn = -sgn$
7. If $a< 0$ then $[a, b, sgn] * =(-1)$.
8. If $c< 0$ then $[c, d, sgn] * =(-1)$.
9. Store (a, b) in $\boldsymbol{R}$ and (c, d) in $\boldsymbol{C}$ to preserve the ascending order at position p_r and p_c.
10. Enumerate *deletions* from p_r (p_c for columns) to the end of the list to exchange each a (c) into b (d).
11. $sgn * = (-1)^{pr-pc}$.

The implementation of lists **R** and **C** requires some remarks because they are specifically designed to perform their duties. The most optimal implementation of ordered lists is a balanced search tree. Both searching and adding components to such a list have complexity $O(log\ N)$. However, the skipped list has the similar complexity, while maintaining the advantages of an ordinary linked list, such as the rapid enumeration of a certain range of elements.[37] The list is a type of linked bidirectional list. The removal and insertion before or after of an element are immediate. However, random access and indexing are more complex. The index of the first element is always 0, while the index of the last element is always the current number of elements minus 1. Additionally, the pointer to the last inserted element is stored with its index. The list that represents HOSC is usually short. The number of *deletions* never exceeds the number of nodes. Searching the existing *deletions* or places where a new *deletion* should be inserted can be accelerated by exploring the features of a canonical HOSC. The scanning can start from the beginning, end, or the last inserted position depending on which number is closer. These determine the scanning direction. In a canonical HOSC, in *deletion* $(a \pm b)$, b is always lower than a and cannot play the role of a in other *deletions*. This helps to decide where to search. Additionally, the operation $a := \boldsymbol{R}[p]$ always returns a pointer where a new *deletion* should be inserted.

Algorithm 1 presents the general rules for obtaining the canonical form. In the method presented here, a *complete* HOSC plays a special role. In a *complete* HOSC, the number of the *pairs of deletions* is equal to the number of nodes. Regardless of the content of the matrix, they can have only 3 values ± 1 or 0. A *valid* HOSC has the common *canonical* form (9) with a "$\pm$" sign. However, most HOSCs are *invalid* due to of *multiple deletions*. In practice, a *valid complete* HOSC forms a common spanning tree in a 2-graph representation, whereas *deletions* in an *invalid complete* HOSC form loops at least for rows or columns. Thus, they these cannot form any spanning tree. A *valid* HOSC always reduces to the same *canonical form* (9). *Complete* HOSCs are the tools that link the 2-graph method and methods that are based on NAMs to one common representation.

By operating only on *complete* HOSCs, we can improve Algorithm 1. If each *pair of deletions* contains some node n_r that has already been analyzed, we can "forget" any *deletion* with n_r. The information that is contained in such a *deletion* is no longer necessary. We can remove them from tables $\boldsymbol{C}$ and $\boldsymbol{R}$ to conserve memory and accelerate further operations. However, to do this without affecting the result, we should do it according to *HOSC arithmetic* rules ((16)-(25)). Additionally, if we detect that a node that had just separated has not yet appeared in the *deletions* for the rows or columns, it will no longer appear. We can be sure that this *complete* HOSC cannot form a common spanning tree. Thus, we can assume that it is equal to zero, without further analysis.

The HOSC implementation from Lasota[26-28] is different. Each HOSC was represented by 3 vectors of a length equal to the number of nodes in the circuit. While manipulations on a single HOSC was faster, it required much more memory. The application of the operation described in the next 2 algorithms was impossible. Thus, the early detection of invalid HOSCs was difficult. In practice, the entire analysis consumed much more time and memory.

Algorithm 2. Removing node n_s in a canonical HOSC

1. Try to find $(n_s \pm r)$ at position p_r in the rows.
2. If it does not exist, find the first appearance of $(r \pm n_s)$ by looking forward from the previous unsuccessful position. Let its position be p_{rr}.
   2.1. If it also does not exist, the result is 0. Break the analysis.
   2.2. Otherwise, change (r $\pm$ n_s) into $(n_s \pm r)$ with $sgn * = (-1)$ in the case of $(n_s \pm r)$.
   2.3. Replace each n_s with $\pm r$ in the *deletions* starting from p_r.
   2.4. $p_r := p_{rr}$.
3. Perform a similar operation for the *deletions* in the columns. Determine p_c
4. Remove the deletions that are found and $\text{sgn}^* = (-1)^{p_r - p_c}$.

However, the removed nodes are usually one (or more) node(s) in the added *deletions* at the same moment. In this case, the number of operations can be limited. We can usually avoid adding and removing the same *deletions*.

Algorithm 3. Adding deletion $(n_s \pm r)$, while n_s is removed.

1. First, if a *deletion* has the form $(r \pm n_s)$, it should be inverted to $(n_s \pm r)$, with the proper sign changing.
2. Try to find $(n_s \pm a)$ at some position p_a.
3. If it does not exist, enumerate each *deletion* from p_a to the end replacing each r with $\pm n_s$.
4. Otherwise, replace it with $(\pm r \pm a)$, extract the sign, and make it canonical by changing r to a, if $a > r$ and moving it to the correct position p_r. Let it be represented by $(p_n \pm r_n)$.
5. Enumerate each *deletion* from position p_r and substitute $p_n := \pm r_n$.

Example 1. Let us present Algorithms 1 to 3 with the HOSC $\Delta^{(4+0)(6+0)(2+6)(1+3)(3+4)(5+6)}_{(1+2)(3+5)(2+0)(1+3)(3+4)(5+6)}$ and $\Delta^{(4+0)(6+0)(2+6)(1+3)(3+4)(5+6)}_{(1+2)(3+5)(2+6)(1+3)(3+4)(5+6)}$ presented above. The order of the *pairs of deletions* was changed a little to emphasize some features of the method, but the order of the pairs does not matter. Let us start with empty row and column lists: $C = []$, $R = []$ and $sgn = 1$. In an HOSC form, it is simply Δ.

1. The first pair is $^{(4+0)}_{(1+2)}$. The lists are empty. Thus, both *deletions* remain intact, and they should be inserted into position 0. Because $1 < 2$, we should invert the *deletion* (1 + 2) into (2 + 1) and change the sign. Both *deletions* are inserted in the same position, and the difference is even. There is no additional sign change. We obtain $-\Delta^{(4+0)}_{(2+1)}$.
2. The next pair is $^{(6+0)}_{(3+5)}$. It does not interfere with the previous pair. Both pairs remain intact; $3 < 5$, and we should revert it to (5 + 3) and change the sign. Both *deletions* must be inserted as the last one at position 1. The difference is even. We obtain $\Delta^{(4+0)(6+0)}_{(2+1)(5+3)}$.
3. Adding the next pair $^{(2+6)}_{(2+0)}$, we can remove nodes 2 and 0 simultaneously. They do not appear in the subsequent pairs. Because node 2 is common for both *deletions*, we start by adding the pair and removing node 2 at the same moment (Algorithm 3). In C, there is no 2, and its probable position is 0. There is nothing to change. For the columns, we can find (2 + 1) at position 0. We force 2 to be 0. We obtain $\Delta^{(4+0)(6+0)}_{(0+1)(5+3)}$. The HOSC is not canonical. However, we need to remove 0, as well. This uncanonical (0 + 1) is well prepared for removing 0. We do not normalize it. To remove 0 in the rows, we should find the first *deletion* with 0. Here, it is (4 + 0). It should be reverted to obtain $-\Delta^{(0+4)(6+0)}_{(0+1)(5+3)}$ and then by enumerating the rest of the *deletions* $-\Delta^{(0+4)(6+4)}_{(0+1)(5+3)}$. Both of removed *deletions* are at the same 0 position. Finally, we obtain $-\Delta^{(6+4)}_{(5+3)}$.
4. Adding the pair $^{(1+3)}_{(1+3)}$ and removing node 1 is trivial. There is no node 1 in $-\Delta^{(6+4)}_{(5+3)}$. Thus, we just add and remove the new pair with no influence on the result.
5. Adding $^{(3+4)}_{(3+4)}$, we can remove both nodes 3 and 4. Let us assume the new pair is added and 3 is removed as the first. The choice is arbitrary. We can remove 4 as well. Both *deletions* should be inserted in the 0 position, and each 3 should be replaced by 4 in the rest of the *deletions*. We obtain $-\Delta^{(6+4)}_{(5+4)}$. To remove node 4, we should transform it into $-\Delta^{(4+6)}_{(4+5)}$, and finally, we obtain $-\Delta$.

The addition of the last *pair of deletion* is trivial. We add $\binom{(5+6)}{(5+6)}$ removing 5 (or 6) at the same time. The pair has no influence for result. We obtain −1.

The analysis of $\Delta^{(4+0)(6+0)(2+6)(1+3)(3+4)(5+6)}_{(1+2)(3+5)(2+6)(1+3)(3+4)(5+6)}$ is the same in the first point. In the second one, we could additionally remove the node 0. There is no 0 in the columns in $\Delta^{(4+0)(6+0)}_{(2+1)(5+3)}$. This HOSC will be invalid, anyway. Frankly, we could have detected this at the very beginning. There is no 0 in the columns. However, in some of the algorithms that will be described later in the paper, we do not know the complete HOSC in advance. We obtain information about the following *pairs of deletions* and nodes that can be removed on the run. Nevertheless, we should dispose of the redundant data as soon as it is possible.

Algorithms 1 to 3 permit an HOSC with *negative summative deletions*, which have no direct interpretation in 2-graph representation, to be analyzed. However, there are some exceptions. Removing *hidden* node 0 creates some ambiguity. In this case, it is better to develop an HOSC according to (11) and to remove 0 separately, which yields a multilevel HOSC. Fortunately, a multilevel HOSC usually reduces in the next few steps.

Example 2. For 4 × 4 matrix, determine the HOSCs: $\Delta^{(4+0)(2-1)(3+2)(4+1)}_{(3+0)(3-1)(3+2)(4+1)}$ and $\Delta^{(3+1)(4-2)(3+2)(4+1)}_{(3+0)(3-4)(3+2)(4+1)}$.
For $\Delta^{(4+0)(2-1)(3+2)(4+1)}_{(3+0)(3-1)(3+2)(4+1)}$, we obtain the following:

1. Adding *pair* $\binom{(4+0)}{(3+0)}$, we obtain $\Delta^{(4+0)}_{(3+0)}$.
2. Adding *pair* $\binom{(2-1)}{(3-1)}$ and removing 0, we have $\Delta^{(4+0)(2-1)}_{(3+0)(3-1)} = \Delta^{(4+0)(2-1)}_{(-1+0)(3-1)} = -\Delta^{(4+0)(2-1)}_{(1+0)(3+0)} = \Delta^{(2-1)(4+0)}_{(1+0)(3+0)}$. Because we remove the *hidden* 0, it is better to develop it into the form $\Delta^{(2+0)(4+0)}_{(1+0)(3+0)} + \Delta^{(1+0)(4+0)}_{(1+0)(3+0)} = \Delta^{(0+2)(4+2)}_{(0+1)(3+1)} + \Delta^{(0+1)(4+1)}_{(0+1)(3+1)} \xrightarrow{1} \Delta^{(4+2)}_{(3+1)} + \Delta^{(4+1)}_{(3+1)}$.
3. Adding *pair* $\binom{(3+2)}{(3+2)}$ and removing both nodes 3 and 2 yield $\Delta^{(4+2)(3+2)}_{(3+1)(3+2)} + \Delta^{(4+1)(3+2)}_{(3+1)(3+2)} = \Delta^{(4+2)(3+2)}_{(2+1)(3+2)} + \Delta^{(4+1)(3+2)}_{(2+1)(3+2)} \xrightarrow{3} \Delta^{(4+2)}_{(2+1)} + \Delta^{(4+1)}_{(2+1)} = -\Delta^{(2+4)}_{(2+1)} + \Delta^{(4+1)}_{(2+1)} \xrightarrow{2} -\Delta$.
4. Finally, after adding *pair* $\binom{(4+1)}{(4+1)}$ and removing node 4, we obtain $-\Delta^{(4+1)}_{(4+1)} \xrightarrow{4} -1$.

HOSC $\Delta^{(3+1)(4-2)(3+2)(4+1)}_{(3+0)(3-4)(3+2)(4+1)}$ seemingly has no 0 node in the rows. However, it is hidden in (4 − 2). The first step is trivial. Lets us start with the second one.

2. Adding *pair* $\binom{(4-2)}{(3-4)}$ and removing 0, we should develop the HOSC into the sum of 2 HOSCs: $\Delta^{(3+1)(4-2)}_{(3+0)(3-4)} = \Delta^{(3+1)(4-2)}_{(3+0)(4+0)} = \Delta^{(3+1)(4+0)}_{(3+0)(4+0)} + \Delta^{(3+1)(2+0)}_{(3+0)(4+0)} = \Delta^{(3+1)(0+4)}_{(0+3)(4+3)} + \Delta^{(3+1)(0+2)}_{(0+3)(4+3)} \xrightarrow{0} -\Delta^{(3+1)}_{(4+3)} - \Delta^{(3+1)}_{(4+3)} = -2\cdot\Delta^{(3+1)}_{(4+3)}$.
3. Adding *pair* $\binom{(3+2)}{(3+2)}$ and removing nodes 3 and 2, we obtain $-2\cdot\Delta^{(3+1)(3+2)}_{(4+3)(3+2)} = -2\cdot\Delta^{(2+1)(3+2)}_{(4+2)(3+2)} \xrightarrow{3} -2\cdot\Delta^{(2+1)}_{(4+2)} = 2\cdot\Delta^{(2+1)}_{(2+4)} \xrightarrow{2} 2\cdot\Delta$.
4. Adding *pair* $\binom{(4+1)}{(4+1)}$ and removing node 4, finally, we have $2\cdot\Delta^{(4+1)}_{(4+1)} \xrightarrow{4} 2$

Thus, $\Delta^{(3+1)(4-2)(3+2)(4+1)}_{(3+0)(3-4)(3+2)(4+1)} = 2$.

However, another question arise, why does $\Delta^{(4+0)(6+0)}_{(2+1)(5+3)}$ have no *hidden* 0 for columns. Of course, we can develop it as $\Delta^{(4+0)(6+0)}_{(2+1)(5+3)} = \Delta^{(4+0)(6+0)}_{(2+0)(5+3)} - \Delta^{(4+0)(6+0)}_{(1+0)(5+3)} = \Delta^{(0+4)(6+4)}_{(0+2)(5+3)} - \Delta^{(0+4)(6+4)}_{(0+1)(5+3)}$. After removing node 0 we obtain $\Delta^{(0+4)(6+4)}_{(0+2)(5+3)} - \Delta^{(0+4)(6+4)}_{(0+1)(5+3)} \xrightarrow{0} \Delta^{(6+4)}_{(5+3)} - \Delta^{(6+4)}_{(5+3)} \equiv 0$. We always obtain 0.

3 | CIRCUIT ANALYSIS WITH HOSC

Let us assume that a circuit can be represented by NAM $\boldsymbol{Y}$ (Figure 1). Let us assume that the circuit is driven by known input current I_I that is connected between nodes a and d. The output value can be a voltage drop between nodes b and c or current that flows from nodes b to c. According to the superposition rule, the component with the nodal voltage V_b, which comes from the current that is forced into a, can be determined by $V_{ba} = \Delta^{(a+0)}_{(b+0)}/\Delta\cdot I_a = \Delta^{(a+0)}_{(b+0)}/\Delta\cdot I_I$.[1] Similarly, the component that comes from the current that flows into node d is $V_{bd} = \Delta^{(d+0)}_{(b+0)}/\Delta\cdot I_a = -\Delta^{(d+0)}_{(b+0)}/\Delta\cdot I_I$. Adding both

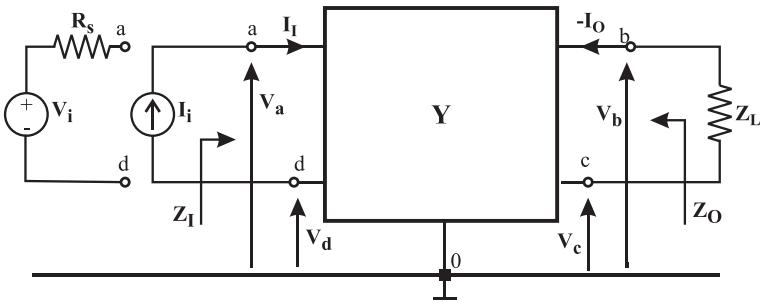

FIGURE 1 A circuit as a 2-port

components and applying (1), one can obtain $V_b = V_{ba} + V_{bd} = \Delta_{(b+0)}^{(a+d)}/\Delta \cdot I_I$. Similarly, $V_c = \Delta_{(c+0)}^{(a+d)}/\Delta \cdot I_I$. Because $V_{bc} = V_b - V_c$, we finally obtain

$$J = \frac{V_{bc}}{I_I} = \frac{\Delta_{(b+c)}^{(a+d)}}{\Delta}. \tag{26}$$

Instead of 4 basic cofactors for the numerator, only one summative cofactor can be determined. Quite often, both the input and output signals have a common reference node. If $d = c = 0$, Formula (26) reduces to the well-known classical one.[1] However, for an advanced analysis (eg, noise analysis[38-40] or weekly nonlinear circuit analysis[41]), the stimulus signal may be connected between 2 nodes that are different than the reference node, and the output signal can be a voltage drop between the nodes that are different than the reference node.

Any impedance that is seen between nodes a and b can be derived directly from (26)

$$Z = \frac{V_{ab}}{I_{ab}} = \frac{\Delta_{(a+b)}^{(a+b)}}{\Delta}. \tag{27}$$

In an actual circuit analysis, the current input and voltage output are not the only desirable configurations. The voltage input and the current output are also required. Additionally, an analysis of MIMO is desirable. The superposition rule can be applied. However, removing the voltage inputs generates short circuits, which change the topology of a circuit. The key to the solution is the extraction of the components from the NAM.

Theorem 1. (parameter extraction theorem)
Any HOSC Δ_C^R, which is a function of (trans)conductance G, can be determined as $\Delta_C^R(G) = \Delta_C^R(G = 0) + G \cdot \Delta_{C,(k+l)}^{R,(p+r)}$, where p, r, k, l are the nodes to which a component is connected. For simplification, we will use the form

$$\Delta_C^R \overset{G}{\rightarrow} \Delta_C^R + G \cdot \Delta_{C,(k+l)}^{R,(p+r)}, \tag{28}$$

where a (trans)conductance G above the arrow means that it is being extracted.
The interpretation of the nodes for the transconductance (VCCS) is presented in Figure 2. For each 2-terminal, $p = k = a$, and $r = l = b$.

The theorem was originally presented in Alderson and Lin[42] to determine NAM. However, a form with an HOSC is more convenient to use and can be generalized for any HOSC in a natural way. The proof is obtained by applying the Laplace theorem and relation (3).

A short circuit between nodes a and b can be considered by adding a *pair of deletions* $\begin{smallmatrix}(a+b)\\(a+b)\end{smallmatrix}$ to each cofactor, instead. That is,

$$\Delta_C^R \overset{a \leftrightarrow b}{\longrightarrow} \Delta_{C,(a+b)}^{R,(a+b)}. \tag{29}$$

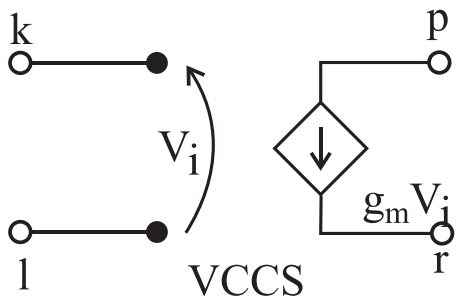

FIGURE 2 VCCS—transconductance

The proof is trivial. A short circuit can be modeled as resistance $R = 0$ ($G \to \infty$) connected to nodes a and b. By applying Theorem 1, one obtains $T \xrightarrow{G\to\infty} \frac{\Delta^{RN,(a+b)}_{CN,(a+b)}}{\Delta^{RD,(a+b)}_{CD,(a+b)}}$.

The genuine NAM approach assumes that the input signals are currents. To circumvent this, the authors use some special tricks such as the application of an additional transconductance[1,10-15,22,43] or using a special equivalent circuit that consists of the current source, unit resistor, and nullator.[17-21] The former approach can cause redundancy for a MIMO analysis. The latter one increases the size of NAM and the number of components unnecessarily.

However, it can always be determined directly with HOSC. Let us assume that the circuit is driven by a voltage source with a resistance R_S that is connected in a series (see Figure 1). According to the Norton rule, this could be transformed into a parallel connection of the conductance $G_s = 1/R_S$ and the current source $i_I = v_I \cdot G_s$. By inserting them into (26), applying Theorem 1 and removing terms that have *multiple deletions*, one obtains

$$V_{bc} = \frac{\Delta^{(a+d)}_{(b+c)} + G_S{\cdot}\Delta^{(a+d)(a+d)}_{(b+c)(a+d)}}{\Delta + G_s{\cdot}\Delta^{(a+d)}_{(a+d)}} G_s{\cdot}V_{ad} = \frac{G_s{\cdot}\Delta^{(a+d)}_{(b+c)}}{\Delta + G_s{\cdot}\Delta^{(a+d)}_{(a+d)}} V_{ad} = \frac{\Delta^{(a+d)}_{(b+c)}}{R_s{\cdot}\Delta + \Delta^{(a+d)}_{(a+d)}} V_{ad}.$$

For $R_s = 0$, it reduces to

$$H_u = \frac{V_{bc}}{V_{ad}} = \frac{\Delta^{(a+d)}_{(b+c)}}{\Delta^{(a+d)}_{(a+d)}}. \tag{30}$$

Similarly, if the output signal is the short-circuit current flowing through load Z_L (Y_L), the current-to-voltage and current-to-current transfer functions can be determined indirectly:

$$K = \frac{-i_O}{v_I} = H_u{\cdot}Y_L = \frac{Y_L{\cdot}\Delta^{(a+d)}_{(b+c)}}{\Delta^{(a+d)}_{(a+d)} + Y_L{\cdot}\Delta^{(a+d)(b+c)}_{(a+d)(b+c)}}, \tag{31}$$

$$H_i = \frac{-i_O}{i_I} = J{\cdot}Y_L = \frac{Y_L{\cdot}\Delta^{(a+d)}_{(b+c)}}{\Delta + Y_L{\cdot}\Delta^{(b+c)}_{(b+c)}}. \tag{32}$$

However, since the output current is a short-circuit current and $Y_L \to \infty$, it yields

$$K = \left.\frac{-i_O}{v_I}\right|_{Y_L\to\infty} = \frac{\Delta^{(a+d)}_{(b+c)}}{\Delta^{(a+d)(b+c)}_{(a+d)(b+c)}}, \tag{33}$$

$$H_i = \left.\frac{-i_O}{i_I}\right|_{Y_L\to\infty} = \frac{\Delta^{(a+d)}_{(b+c)}}{\Delta^{(b+c)}_{(b+c)}}. \tag{34}$$

In most papers on symbolic analysis, authors determine only one voltage transfer function, whereas an actual circuit analysis can be solved, if we know the response of many values that comes from many sources. The analysis of linear circuits with many input signals can be done by applying the superposition rule. If short-circuiting of some voltage sources is needed, we simply apply (29) instead. In practice, we can forget about Formulae (26) to (34) and apply the following rule of thumb:

Algorithm 4.

1. Open circuit in each branch with any output short-circuit current to determine.
2. Number the nodes in order to associate each open-circuited current branch with 2 different numbers.
3. To form the common denominator of each transfer function, add the pairs of deletions $\binom{a_{V_I}+b_{V_I}}{a_{V_I}+b_{V_I}}$ from each input voltage source and the pairs of deletions $\binom{a_{I_O}+b_{I_O}}{a_{I_O}+b_{I_O}}$—from each temporarily open-circuited output current.
4. For the numerator of the transfer function from the input gates $I_h - I_l$ to the output gates $O_h - O_l$, copy the denominator *deletion* by *deletion*, except for *pairs* $\binom{I_h+I_l}{I_h+I_l}$ and $\binom{O_h+O_l}{O_h+O_l}$, if they exists. Then, add *pair* $\binom{I_h+I_l}{O_h+O_l}$.

Example 3. Let us determine the network functions to determine I_O and U_O for the circuit in Figure 3. There is one voltage source and one current output in the circuit. To determine the short-circuit current I_O, node 3 was split into two nodes—3 and 4. The common denominator is $\Delta^{(1+0)(3+4)}_{(1+0)(3+4)}$. The numerators are as follows: V_I-to-$V_O - \Delta^{\cancel{(1+0)}(4+3)(1+0)}_{\cancel{(1+0)}(4+3)(2+3)}$; V_I-to-$I_O - \Delta^{\cancel{(1+0)(4+3)}(1+0)}_{\cancel{(1+0)(4+3)}(3+4)}$; I_I to V_O, $\Delta^{(1+0)(4+3)(2+3)}_{(1+0)(4+3)(2+3)}$; and I_I-to-$I_O - \Delta^{(1+0)\cancel{(4+3)}(2+3)}_{(1+0)\cancel{(4+3)}(3+4)}$. Summarizing the results, we obtain $V_O = \dfrac{\Delta^{(1+0)(4+3)}_{(2+3)(4+3)} \cdot V_I + \Delta^{(1+0)(4+3)(2+3)}_{(1+0)(4+3)(2+3)} \cdot I_I}{\Delta^{(1+0)(4+3)}_{(1+0)(4+3)}}$ and $I_O = \dfrac{\Delta^{(1+0)}_{(3+4)} \cdot V_I + \Delta^{(1+0)(2+3)}_{(1+0)(3+4)} \cdot I_I}{\Delta^{(1+0)(3+4)}_{(1+0)(3+4)}}$.

To determine the output values, 5 cofactors should be calculated. This can be done by NAM manipulation and proper determinant computation. However, one component, VCVS (E), makes it difficult with a genuine matrix. To circumvent this, a MNAM can be used. Nevertheless, neither the determination of 5 cofactors separately nor the manipulation of the matrices is effective. Additionally, methods based on the admittance matrix are not cancellation free and require the additional operation of de-cancellation. We will determine the cofactors directly from the netlist simultaneously.

4 | CANCELLATION-FREE SYMBOLIC HOSC ANALYSIS

There are 6 types of primitive components that can be found in linear circuits: 2-terminals as impedance (Z) or admittance (Y), VCCS (G-type in Spice), VCVS (E-type), CCCS (F-type), and CCVS (H-type). Only Y and G can be directly applied to a genuine NAM. To circumvent this inconvenience, a MNAM[9] or some tricks that use the pathological

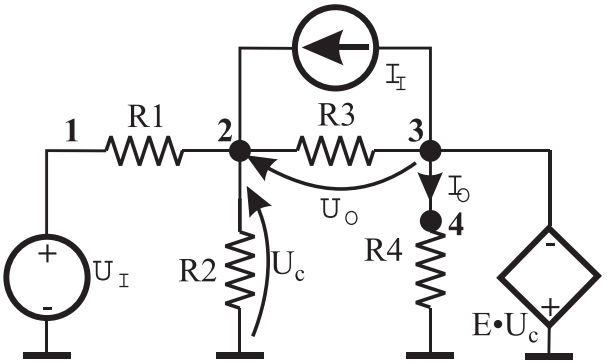

FIGURE 3 Example of a multi-excitation multi-output circuit

components can be used.[17-21] In the MNAM approach, each element that is incompatible with the NAM increases the size of a problem. In an approach with pathological components, one usually increases and then decreases the size of a problem. Thus, the size of the problem to be solved is usually moderately lower. Fortunately, HOSC arithmetic allows each type of component to be considered without any redundancy, even for a genuine NAM.

4.1 | Parameter extractions

In (28), the extraction rule for any conductance and transconductance is presented. Although the rest of the primitive components cannot be put into a genuine NAM, we can form a similar rule of extraction. Assuming that each extraction rule has a general form

$$\Delta_C^R = P \cdot \Delta_{C,c_1}^{R,r_1} + \Delta_{C,c_0}^{R,r_0}, \tag{35}$$

where P is the extracted parameter, Δ_{C,c_1}^{R,r_1} is the divisor, and Δ_{C,c_0}^{R,r_0} is the remainder. $\begin{smallmatrix} r_1 \\ c_1 \end{smallmatrix}$ and $\begin{smallmatrix} r_0 \\ c_0 \end{smallmatrix}$ are additional *pairs of deletions* for the divisor and remainder, respectively. These *pairs* for 6 primitive components are presented in Table 1. An ellipsis means that there were no additional *deletions*. Each 2-terminal described as admittance or impedance is connected between nodes a and b. The controlled sources are connected as is presented in Figure 4. k and l are always voltage nodes, and k points the higher voltage than l. p and r are always the current nodes and the current flows from nodes p to r. Thus, for the pathological components, nodes k and l are the same for the voltage domain components (the nullator or the voltage mirror) and p and r for the current domain components (the norator or the current mirror). Any valid pathological pair should have pathological components from both domains. Thus, the symbol $\varnothing$ is used to show a *pair of deletions* from the complementary component. The pathological components have no parameter. That is why they do not generate any 1-descendant. The main advantage of the pathological components is that the *pairs of deletions* that are associated with them should be added obligatorily to each HOSC and this can be done at the very beginning.

The proofs of the rules are simple. Rules 1 and 3 are direct applications of Theorem 1. Rule 2 can be derived from rule 1 by substituting $G = 1/Z$. The proofs of rules 4 to 6 are similar. Let us prove rule 6. Assuming that a CCVS has some finite and nonzero input and output resistances, we can model it as is presented in Figure 5.

This model consists of 3 G-type components, and they can be extracted easily. The cofactor in the numerator yields $\Delta_{RN}^{\mathrm{CN}} \xrightarrow{G_i,G_O,G_iHG_O} \Delta_{RN}^{CN} + G_i\Delta_{RN,(k+l)}^{CN,(k+l)} + G_o\Delta_{RN,(p+r)}^{CN,(p+r)} + G_iHG_o\Delta_{RN,(k+l)}^{CN,(p+r)} + G_iG_o\Delta_{RN,(p+r),(k+l)}^{CN,(p+r),(k+l)}$. By performing the same operations for the denominator and returning to the ideal case when $G_i,\ G_o \to \infty$, we obtain $T = \dfrac{H\Delta_{RN,(k+l)}^{CN,(p+r)} + \Delta_{RN,(p+r)(k+l)}^{CN,(p+r)(k+l)}}{H\Delta_{RD,(k+l)}^{CD,(p+r)} + \Delta_{RD,(p+r)(k+l)}^{CD,(p+r)(k+l)}}$.

TABLE 1 Extraction rules for different primitive components

No	Elem. Name	Elem. Type P_i	Divisor *Deletions* $\begin{smallmatrix} r_1 \\ c_1 \end{smallmatrix}$	Reminder *Deletions* $\begin{smallmatrix} r_0 \\ c_0 \end{smallmatrix}$
1.	Admittance	Y	$\begin{smallmatrix} (a+b) \\ (a+b) \end{smallmatrix}$	$\begin{smallmatrix} \cdots \\ \cdots \end{smallmatrix}$
2.	Impedance	Z	$\begin{smallmatrix} \cdots \\ \cdots \end{smallmatrix}$	$\begin{smallmatrix} (a+b) \\ (a+b) \end{smallmatrix}$
3.	VCCS	G	$\begin{smallmatrix} (p+r) \\ (k+l) \end{smallmatrix}$	$\begin{smallmatrix} \cdots \\ \cdots \end{smallmatrix}$
4.	VCVS	E	$\begin{smallmatrix} (p+r) \\ (k+l) \end{smallmatrix}$	$\begin{smallmatrix} (p+r) \\ (p+r) \end{smallmatrix}$
5.	CCCS	F	$\begin{smallmatrix} (p+r) \\ (k+l) \end{smallmatrix}$	$\begin{smallmatrix} (k+l) \\ (k+l) \end{smallmatrix}$
6.	CCVS	H	$\begin{smallmatrix} (p+r) \\ (k+l) \end{smallmatrix}$	$\begin{smallmatrix} (p+r)(k+l) \\ (p+r)(k+l) \end{smallmatrix}$
7.	Nullator			$\begin{smallmatrix} \varnothing \\ (k+l) \end{smallmatrix}$
8.	Norator			$\begin{smallmatrix} (p+r) \\ \varnothing \end{smallmatrix}$
9.	VM			$\begin{smallmatrix} \varnothing \\ (k-l) \end{smallmatrix}$
10.	CM			$\begin{smallmatrix} (p-r) \\ \varnothing \end{smallmatrix}$

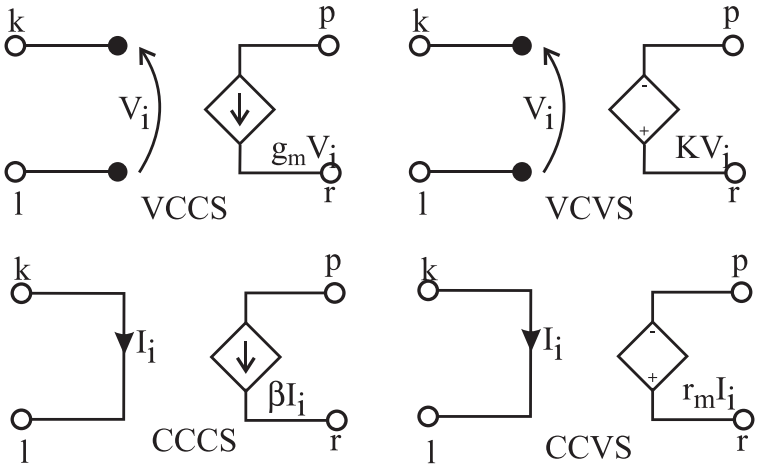

FIGURE 4 Four types of controlled sources

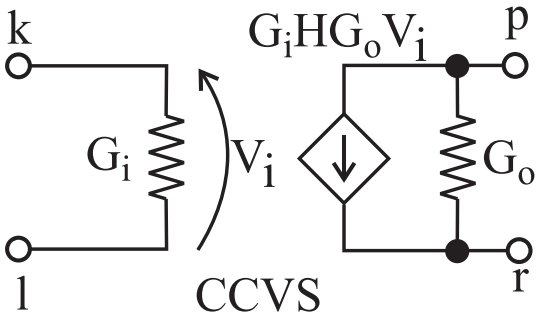

FIGURE 5 The equivalent of CCVS with a nonzero input and output resistance

4.2 | Special case of pathological components

The pair nullator-norator can be obtained from any of rules 3 to 6 assuming that parameters G, E, F, $H \to \infty$. In a similar way, the rest of the pathological pair combinations can be derived by using the models from Soliman and Saad[44] and applying (1).

Pathological components are special cases. Unlike the other primitive components, they do not have any parameters, and they never split an HOSC into 2 descendants. That is why they are obligatorily added to each HOSC from the network functions, usually at the beginning. Then the HOSCs should be reduced to a canonical form. If there are nodes that have only pathological components connected to them, the nodes must be removed, and the number of nodes will be reduced. Zeros and the same HOSC should be detected. This dramatically simplifies the further analysis. Such a procedure is reasonable. However, if there are some other crucial reasons, in contrast to the procedures presented in previous studies,[17-22] the addition of a pathological component can be delayed until some moment in the future.

On the other hand, an inverted operation is always possible. We can create a new node with a *pair of deletions*. If such a pair is common for each HOSC, it can be implemented as a pathological pair. By choosing the proper pairs, it is possible to suppress some of the unwanted terms, thus leaving only the desirable one. This feature of pathological components and HOSCs can be the basis of a synthesis method of circuits that have an active modern component.

Although the problem is generally out of the scope of this paper, the proofs of some of the pathological component equivalents become trivial and evident with HOSCs. For example, in Wang et al,[45] there are a few "new" equivalences that can be useful when searching for new active circuits on the basis of the existing ones. In Figure 6A, 3 equivalences for the current mirrors and norators are presented. A similar equivalence can be formed for the voltage mirrors and nullators. By applying HOSC arithmetic, we obtain $\Delta_{\dots}^{\dots(c-a)(a-b)} \overset{a\to -b}{=} \Delta_{\dots}^{\dots(-b-c)(a-b)} = -\Delta_{\dots}^{\dots(b+c)(a-b)} = \Delta_{\dots}^{\dots(c+b)(a-b)}$, or $\Delta_{\dots}^{\dots(c-a)(a-b)} = \Delta_{\dots}^{\dots(a-c)(a-b)} \overset{a\to c}{=} \Delta_{\dots}^{\dots(a-c)(-c-b)} = -\Delta_{\dots}^{\dots(a-c)(c+b)} = \Delta_{\dots}^{\dots(a-c)(b+c)}$. We obtain the other 2 configurations. Note that there is some additional effort due to changing the sign. Actually, the norators and nullators are directed

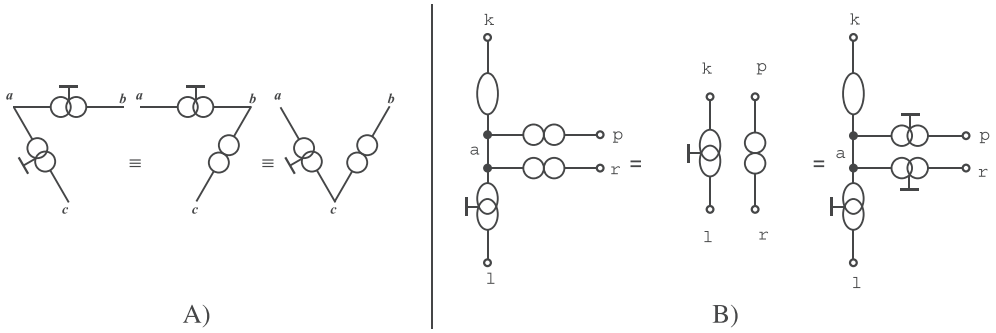

FIGURE 6 Representative pathological equivalence from Wang et al[45]

components, while the voltage and current mirrors are not. Most authors ignore this fact[17-22] because changing the direction of norator or nullator usually changes the sign in each HOSC simultaneously and the sign is reduced. However, there are some exceptions in specific applications in which the sign becomes essential, eg, Filaretov and Gorshkov[35] and Filaretov et al.[36]

In Figure 6B, other apparently new equivalences are presented, which are trivial, as well. This time, there is an internal node a, which is inaccessible from the outside. For the HOSC in the left circuit, we have $\Delta^{\cdots(p+a)(a+r)}_{\cdots(k+a)(a-l)} = \Delta^{\cdots(p+r)(a+r)}_{\cdots(k-l)(a-l)} \xrightarrow{a\rightarrow} \Delta^{\cdots(p+r)}_{\cdots(k-l)}$. Similarly, for any HOSC in a circuit with the left component configuration, we obtain $\Delta^{\cdots(p-a)(a-r)}_{\cdots(k+a)(a-l)} = \Delta^{\cdots(p+r)(a-r)}_{\cdots(k-l)(a-l)} \xrightarrow{a\rightarrow} \Delta^{\cdots(p+r)}_{\cdots(k-l)}$. Both configurations are actually the same and are equal to the middle configuration. Any equivalence from Wang et al[45] can be proven in a similar simply manner. Furthermore, less obvious equivalences, which are not presented in Wang,[45] are also possible.

4.3 | Parameter decision diagrams

Due to Formula (35), each primitive component can be represented by $P\left(\begin{smallmatrix}R_0, R_1\\ C_0, C_1\end{smallmatrix}\right)$, where $\begin{smallmatrix}R_0\\ C_0\end{smallmatrix}$ and $\begin{smallmatrix}R_1\\ C_1\end{smallmatrix}$ are additional *pairs of deletions* for the remainder and divisor, respectively. On the other hand, Formula (35) describes a vertex in the structure of the binary decision diagrams (BDDs),[46] or rather a zero-suppressed BDD.[47] The zero-suppressed BDD is a base of the symbolic analysis method with DDD[6-8] and GPPD.[10-15] The former method as a parameter P uses the entries of MNAM, while the latter uses as the parameters P the primitive components. Although Formula (35) comes from a NAM analysis, it represents a vertex of a GPDD. A GPDD is a topological analysis method that is based on an improved 2-graph method. The method to create BDD-like structure based on Formula (35) was presented in Lasota[24] as a component decision diagram. In Lasota,[25-28] it was renamed the PDD to avoid misunderstandings for some components that are described with more than one parameter. Both GPDD and PDD for the same circuit with the same prefixed order give the same diagram. Both can be called PDD. The name GPDD emphasizes the origin of the method too much. The difference is only in the method for validating the subbranches and determining the sign. A sign is a natural attribute of HOSC arithmetic and unlike the 2-graph approach requires no additional efforts to be determined. The 2-graph approach starts with a large complete 2-graph, which is reduced systematically component by component. In the HOSC approach, we start with small lists of *pair of deletions* that describes the HOSC to be determined. The size of the lists systematically increases with each component that is considered, but decreases with each separated node that is removed. At the end of the analysis, we always obtain a single HOSC with an empty list of *pairs of deletions*.

Formula (35) is the basis of the presented method. To obtain PDD, we should define the criterion when the 0- and 1-terminal vertices are reached and make some attempts to obtain a *canonical* decision diagram.[46,47]

To solve the first problem, let us assume that we extract each component. Then we have only an HOSC of an empty (**0**) matrix left. They can be equal to 1 only if their *canonical* form is (9). Otherwise, it is always equal to 0.

Any decision diagram is *canonical* if it is (*a*) *ordered* and (*b*) *unique*.[46,47] The algorithm based on extraction component by components ensures fulfillment of an *ordered* condition in a natural way. No parameter can be extracted twice. The fulfillment of *uniqueness* is a more serious problem. However, not as serious like in a GPDD. In the paper[15] and in the book[43] in chapter 7.4.1., the author presents the possibility of obtaining 2 different graphs that have the same branches forming the spanning tree. This means that although these 2 graphs are different, they represent the same term. The authors claim the need to use an extra procedure called a *canonical GPDD reduction* to make a GPDD

canonical, which comes from Bryant[46] with the name *reduce*. Using an *always-canonical* HOSC, we can avoid such a situation. The remaining parameters are placed in the common 2-graph virtually, and they never move, except when being removed. Higher order summative cofactors are description of branches that are the equivalent of *collapsed* branches in the GPDD version. The same set of *collapsed branches* that comes from the different origins is always represented by the same *canonical* HOSC. However, there can be another source of *nonuniqueness*. Many different HOSCs can yield the value 0. According to Algorithms 1 to 3, there is more than one reason for obtaining 0, and there is a possibility of obtaining vertex with all of the descendants being equal to 0. This means that 0 is detected late. Fortunately, such a case can be eliminated by advanced, but standard protocol for creating vertices. Before a new vertex is created, the algorithm first checks to see whether the same vertex exists in the *cache* (it has the same associated parameter and the same descendants). If so, the existing vertex is returned; otherwise, a new vertex is created and *cached*.

To determine an HOSC, we are not going to create a matrix or graph. The input form of a circuit is a list of the components with the number of nodes to which the component is connected, eg, netlist in the Spice style. By counting the number of appearances of a node in the list, we can create an incidence vector with entries that indicates the number of components that are connected to a node. The analysis extracts the component one by one until the list is empty. Each time a component is removed, we decrease the numbers in the incidence vector. If the number reaches zero, the node becomes a separated one, and then it should be removed.

Any graph algorithm can be a breath-first search (BFS) or depth-first search (DFS). Creating a GPDD, like many BDD-like algorithms, is only DFS. Depth-first search is considered to be an algorithm that consumes less memory (only one 2-graph at each level) that can be represented by a simple recursive procedure. The DFS version for creating a PDD is very similar to creating a GPDD algorithm. The complexity of the algorithm presented below is at least the same as creating a GPDD, because a *reduction* operation is not necessary.

Algorithm 5. DFS PDD creation

Let $SL\{P_i\}$ be a set of m primitive components. Let $I_R[0..n]$ and $I_C[0..n]$ be an incidence table that is initiated by zeros for the rows and columns, respectively. Let us enumerate each P_i in $SL\{P_i\}$. Let n_{RP_i} and n_{CP_i} be the set of nodes that forms the *deletions* for the rows and columns, respectively. Let us increase the numbers in $I_R[n_{RP_i}]$ and $I_C[n_{CP_i}]$. Let $PL[P_i]$ be the preordered list of parameters. To obtain a reasonable size of PDD (GPDD) that is created, the heuristic described in Shi[15] can be used. To maintain the uniqueness of the PDD, 2 *caches* based on the *hash table* architecture are needed and both are common for implementing most of the BDD-like structure. A *vertex cache* (VC) is used to avoid nonunique vertices creation. A so-called *operation cache* (OC) is used to avoid repeating the same operation. If some kth HOSC $\Delta_{C_k}^{R_k}$ with ith component P_i has already been analyzed and we obtained vertex V_j, we do not analyze it again, thus returning V_j. The most important part of the code is the recursive function *GetPDD*. As formal parameters we have $\Delta_{C_k}^{R_k}$—the HOSC, P_i—the next parameter to extract, and n_i—the list of just separated nodes.

1. **function** $GetPDD\left(\Delta_{C_k}^{R_k}, P_i, n_i\right)$
2. **if** P_i **is invalid and** $\Delta_{C_k}^{R_k} = \Delta$ **return** *OneTerminalVertex*;
3. **f** P_i **and** $\Delta_{C_k}^{R_k}$ **is in** *OC* **as** V_j **return** V_j;
4. **let** $\begin{smallmatrix} r_0 \\ c_0 \end{smallmatrix} = GetZeroDecendantDeltions(P_i)$;
5. **let** $\Delta_{C_{k0}}^{R_{k0}} = AddDeletionsAndRemoveNodes\left(\Delta_{C_k}^{R_k}, \begin{smallmatrix} r_0 \\ c_0 \end{smallmatrix}, n_i\right)$;
6. **if** $\Delta_{C_{k0}}^{R_{k0}}$ **is valid**
7. $V_0 = GetPDD\left(\Delta_{C_{k0}}^{R_{k0}}, P_i.\text{next}, n_i.next\right)$;
8. **else** $V_0 = ZeroTerminalVertex$;
9. **let** $\begin{smallmatrix} r_1 \\ c_1 \end{smallmatrix} = GetOneDecendantDeltions(P_i)$;
10. **let** $\Delta_{C_{k1}}^{R_{k1}} = AddDeletionsAndRemoveNodes\left(\Delta_{C_k}^{R_k}, \begin{smallmatrix} r_1 \\ c_1 \end{smallmatrix}, n_i\right)$;
11. **if** $\Delta_{C_{k1}}^{R_{k1}}$ **is valid**
12. $V_1 = GetPDD\left(\Delta_{C_{k1}}^{R_{k1}}, P_i.next, n_i.next\right)$;
13 **else** $V_1 = ZeroTerminalVertex$;
14. **let** $V = GetVertex(P_i, V_0, V_1)$;
15. **store** V **with** $\Delta_{C_k}^{R_k}$ **in** *OC*;
16. **return** V;

Let $\Delta_{C_1}^{R_1}, ..., \Delta_{C_m}^{R_m}$ be the set of m HOSCs to be determined. Let us assume that there are $2p$ pathological component—p current dependant (CM or norators), which are represented by the *set of deletions* c_p and p voltage dependant (VM or

nullators), which are represented by the *set of deletions* v_p. Thus, we have to determine $\Delta_{C_1,v_p}^{R_1,c_p},...,\Delta_{C_m,v_p}^{R_m,c_p}$. To do this, we simply call the *GetPDD* function for each HOSC in the set $\Delta_{C_1,v_p}^{R_1,c_p},...,\Delta_{C_m,v_p}^{R_m,c_p}$ with the first component P_1 from the ordered component list *PL* that contains the set of just separated nodes n_1.

However, the DFS approach has a few disadvantages from the point of view of modern computer science. Depth-first search recursive algorithms are difficult to rebuild for any parallel or distributed infrastructure. The idea that the DFS approach consumes less memory is also false when creating a (G)PDD. The most memory consuming structure is the *operation cache*, and we must keep all of the data until the (G)PDD is created. Algorithm 5 jumps up and down removing and restoring components. It is not easy to predict which already determined value could be reused. Additionally, the DFS algorithm has to have a preordered list of parameters that cannot be changed during the creation of a PDD. A typical preordering heuristic is based on the topological adjacency, in which parallel or series branches are lumped together. The analysis is done from the input to the output (or in the opposite direction), and the most adjacent component to the previous one is chosen. Eliminating the adjacent components one by one finally results in the creation of separated nodes and yields simplifications. This heuristic works well for single-input single-output circuits that are compatible with a genuine NAM. For other primitive components, including a pathological one, it sometimes fails. When choosing the next component, we would like to have the one that generates the minimal number of unique descendants and the minimal number of vertices. Components that suppress to zero at least 1-descendants are preferred. However, such an optimization is only possible in the BFS approach. In the BFS algorithm, analysis is made by testing the influence of a component for each unique HOSC from the previous level, and then the component is definitely eliminated. Because the analysis is done component by component, the algorithm can change the order of the remaining components on the run, if there are some reasons. The BFS version for creating a PDD is

Algorithm 6.

Similar to Algorithm 5, let us assume that we have the set of parameters $P = [P_1, P_2, ..., P_m]$. Let $\Delta_{C_1,v_p}^{R_1,c_p},...,\Delta_{C_m,v_p}^{R_m,c_p}$ be a set of HOSC to determine, with additional *deletions* from the pathological components. Let $I_R[1..n]$ and $I_C[1..n]$ be the vectors of the incidence counters for the rows and columns respectively, which are initialized by the components P_1, P_2, ..., P_m.

1. Make the cofactors $\Delta_{C_1,v_p}^{R_1,c_p},...,\Delta_{C_m,v_p}^{R_m,c_p}$ canonical, using Algorithms 1 to 3 and unique using the *cofactor cache* CL_0. The nonzero, unique cofactors are sent to the hash-oriented list of the HOSC to be analyzed in the next step L_0 and their unambiguous identifiers to the list Id_0. Let $i = 0$.
2. While P is not empty
   2.1. Choose the next component P_i. Generate the *deletions* for the descendants $D = \binom{R_{0_i}\ R_{1_i}}{C_{0_i}\ C_{1_i}}$ and the list of just separated nodes $sn = \{n_{1_i}, ...\}$ based on the parameter P_i and incidence vectors I_R and I_C.
   2.2. For each HOSC $\Delta_{C_j}^{R_j}$ in the list L_i
      2.2.1. For each descendant $d \ni \{0, 1, ...\}$
         2.2.1.1. Add $\begin{matrix}R_{d_i}\\C_{d_i}\end{matrix}$, remove nodes $sn = \{n_{1_i}, ...\}$ and make $\Delta_{C_j,...,C_{d_i}}^{R_j,...,R_{d_i}}$ canonical.
         2.2.1.2. If $\Delta_{C_j,...,C_{d_i}}^{R_j,...,R_{d_i}}$ is valid, find items that already exist in the hash-oriented *local HOSC cache* CL_{i+1}. If $\Delta_{C_j,...,C_{d_i}}^{R_j,...,R_{d_i}}$ is new, send it to the list of HOSCs to be analyzed in the next step L_i and their unambiguous identifier to the list Id_i.
   2.3. $i := i + 1$
3. For $k = m$ to 0
   3.1. Based on unambiguous identifiers Id_m, recreate the vertices to make them unique using the *vertices cache*.

The *local cofactor cache* CL_{i+1} is a local list that is only necessary to avoid any repetition of $\Delta_{C_j,...,C_{d_i}}^{R_j,...,R_{d_i}}$ and can be released with another component in the next step of analysis, whereas the list of HOSC L_i that is sent to the next level analysis can be released when the current level is equal to $i + 2$. The list of identifiers and the association with descendents are enough to generate a PDD. Of course, the standard for creating BDD-like structures with unique vertices with a *vertices cache* is necessary, because the canonicity of an HOSC is not a guarantee of the canonicity of the vertices in a few cases. In the middle stage of the analysis, the L_i lists can be quite large. However, the L_i lists are always only a sublist of the HOSC that should be stored in the *operation cache* in the DFS approach. On the other hand, in modern

multicore computer architecture, rather than enumerating L_i HOSC by HOSC, parallel or even distributed techniques can be used. Another place for the application of parallel computing methods is the procedure for selecting the next component.

Example 4. In Example 3, we needed to determine 5 HOSCs for the circuit in Figure 3. The diagram is repeated in Figure 7A for convenience. The common denominator is $D = \Delta^{(1+0)(4+3)}_{(1+0)(4+3)} = A1$. The numerators in the canonical form are $N_{V\to I} = -\Delta^{(1+0)}_{(4+3)} = -A2$, $N_{V\to V} = -\Delta^{(1+0)(4+3)}_{(3+2)(4+2)} = -A3$, $N_{I\to I} = \Delta^{(1+0)(3+2)}_{(1+0)(4+3)} = A4$, and $N_{I\to V} = \Delta^{(1+0)(3+2)(4+2)}_{(1+0)(3+2)(4+2)} = A5$. Let us use the BFS approach. All 5 HOSCs $A1$ to $A5$ are sent to L_0. By extracting component $G3$ first, we obtain the minimum number of descendants. $G3$ adds a *deletion* (3 + 2) for both the rows and columns in the 1-descendant. The 0-descendant remains intact. Thus, we obtain $A1_1 = \Delta^{(1+0)(4+3)\underline{(3+2)}}_{(1+0)(4+3)\underline{(3+2)}} = \Delta^{(1+0)(3+2)(4+2)}_{(1+0)(3+2)(4+2)} = B1$, $A1_0 = \Delta^{(1+0)(4+3)}_{(1+0)(4+3)} = B2$, $A2_1 = \Delta^{(1+0)\underline{(3+2)}}_{(4+3)\underline{(3+2)}} = -\Delta^{(1+0)(3+2)}_{(3+2)(4+2)} = -B3$, $A2_0 = \Delta^{(1+0)}_{(4+3)} = B4$, $A3_1 = \Delta^{(1+0)(4+3)\underline{(3+2)}}_{(3+2)(4+2)\underline{(3+2)}} = 0$, $A3_0 = \Delta^{(1+0)(4+3)}_{(3+2)(4+2)} = B5$, $A4_1 = \Delta^{(1+0)(3+2)\underline{(3+2)}}_{(1+0)(4+3)\underline{(3+2)}} = 0$, $A4_0 = \Delta^{(1+0)(3+2)}_{(1+0)(3+4)} = B6$, $A5_1 = \Delta^{(1+0)(3+2)(4+2)(3+2)}_{(1+0)(3+2)(4+2)(3+2)} = 0$, and $A5_0 = \Delta^{(1+0)(3+2)(4+2)}_{(1+0)(3+2)(4+2)} = B1$. The subscript 0 means a 0-descendant, whereas the subscript 1 means a 1-descendant. Note that HOSCs $B1$ that appear twice are treated as one. There are 6 HOSCs $B1$ to $B6$ in L_1. In the next step, we extract VCVS $E\left(\begin{smallmatrix}(3+0)\\(3+0)\end{smallmatrix}, \begin{smallmatrix}(3+0)\\(2+0)\end{smallmatrix}\right)$, and we can remove node 3 because there is no component left that is connected to node 3. We obtain $B1_1 = \Delta^{(1+0)(3+2)(4+2)\underline{(3+0)}}_{(1+0)(3+2)(4+2)\underline{(2+0)}} = \Delta^{(1+0)(3+0)(4+0)(2+0)}_{(1+0)(3+0)(4+0)(2+0)} \xrightarrow{3} \Delta^{(1+0)(2+0)(4+0)}_{(1+0)(2+0)(4+0)} = C1$, $B1_0 = \Delta^{(1+0)(3+2)(4+2)\underline{(3+0)}}_{(1+0)(3+2)(4+2)\underline{(3+0)}} \xrightarrow{3} \Delta^{(1+0)(2+0)(4+0)}_{(1+0)(2+0)(4+0)} = C1$, $B2_1 = \Delta^{(1+0)\underline{(4+3)}(3+0)}_{(1+0)(4+3)\underline{(2+0)}} \xrightarrow{3} \Delta^{(1+0)(4+0)}_{(1+0)(2+0)} = C2$, $B2_0 = \Delta^{(1+0)(4+3)(3+0)}_{(1+0)(4+3)(3+0)} \xrightarrow{3} \Delta^{(1+0)(4+0)}_{(1+0)(4+0)} = C3$, $B3_1 = \Delta^{(1+0)(3+2)\underline{(3+0)}}_{(3+2)(4+2)\underline{(2+0)}} \xrightarrow{3} \Delta^{(1+0)(2+0)}_{(2+0)(4+0)} = C4$, $B3_0 = \Delta^{(1+0)(3+2)\underline{(3+0)}}_{(3+2)(4+2)\underline{(3+0)}} \xrightarrow{3} \Delta^{(1+0)(2+0)}_{(2+0)(4+0)} = C4$, $B4_1 = \Delta^{(1+0)\underline{(3+0)}}_{(4+3)\underline{(2+0)}} \xrightarrow{3} \Delta^{(1+0)}_{(2+0)} = C5$, $B4_0 = \Delta^{(1+0)\underline{(3+0)}}_{(4+3)\underline{(3+0)}} \xrightarrow{3} \Delta^{(1+0)}_{(4+0)} = C6$, $B5_1 = \Delta^{(1+0)\underline{(4+3)}(3+0)}_{(3+2)(4+2)\underline{(2+0)}} \xrightarrow{3} -\Delta^{(1+0)(4+0)}_{(2+0)(4+0)} = -C7$, $B5_0 = \Delta^{(1+0)(4+3)(3+0)}_{(3+2)(4+2)\underline{(3+0)}} \xrightarrow{3} -\Delta^{(1+0)(4+0)}_{(2+0)(4+0)} = -C7$, $B6_1 = \Delta^{(1+0)(3+2)(3+0)}_{(1+0)(4+3)(2+0)} \xrightarrow{3} -\Delta^{(1+0)(2+0)}_{(1+0)(2+0)} = -C8$, and $B6_0 = \Delta^{(1+0)(3+2)(3+0)}_{(1+0)(4+3)(3+0)} \xrightarrow{3} -\Delta^{(1+0)(2+0)}_{(1+0)(4+0)} = -C2'$. $C2$ and $C2'$ have the *deletions* in the rows and columns exchanged. There are only 2-terminals left, which always have the same *deletions* in the rows and columns. We can assume that they are similar. In the next steps, only the nonzero descendants are presented. In the following step, we extract $G4$ and remove node 4. As a result, we only have 3 nonzero HOSCs left: $C1_0 = C8_1 = \Delta^{(1+0)(2+0)}_{(1+0)(2+0)} = D1$, $C3_0 = \Delta^{(1+0)}_{(1+0)} = D2$, $C5_1 = C7_0 = \Delta^{(1+0)}_{(2+0)} = D3$. In the penultimate step, we extract $G2$ with the 0 node removed. Node 1 can now play the role of the reference node. We obtain $D1_0 = D2_1 = \Delta^{(2+1)}_{(2+1)} = E1$ and $D2_0 = D3_0 = \Delta = E2$. The last component is $G1$ and node 2 is removed. The last step yields $E1_0 = E2_1 = \mathbf{1}$. To obtain the result in PDD form, we start from the last HOSC, creating vertices from the terminal vertex to the root. The vertices are represented in the form $P(D_0, D_1)$, where P is a parameter and D_0 and D_1 are the 0- and 1-descendants. Thus, $P(D_0, D_1) = D_0 + P \cdot D_1$. We obtain the following: $E2 = G1(0, \mathbf{1}) = V_2$, $E1 = \mathbf{1}$, $D3 = V_2$, $D2 = G2(V_2, \mathbf{1}) = V_3$, $D1 = \mathbf{1}$, $C8 = G4(0, \mathbf{1}) = V_4$, $C7 = V_2$, $C6 = 0$, $C5 = G4(0, V_2) = V_5$, $C4 = 0$, $C3 = V_3$, $C2 = 0$, $C1 = \mathbf{1}$, $B6 = E(0, -V_4) = -V_6$,

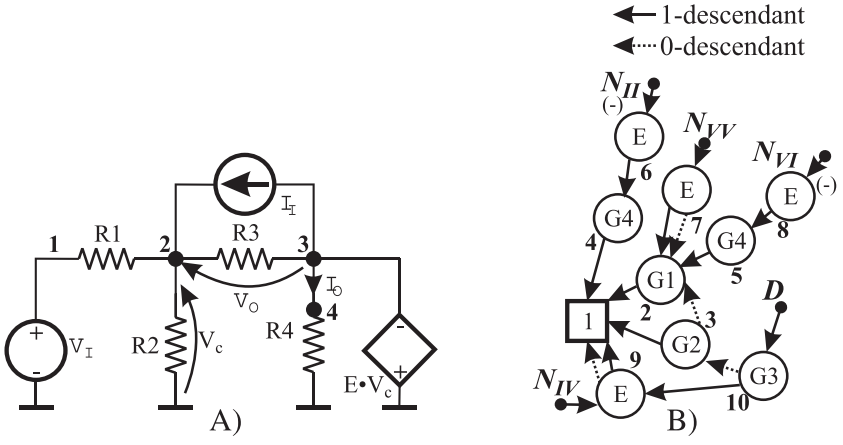

FIGURE 7 The trivial multi-input multi-output circuit: A, the diagram; B, parameter decision diagram

$B5 = -E(V_2, V_2) = -V_7$, $B4 = E(0, V_5) = V_8$, $B3 = 0$, $B2 = V_3$, $B1 = E(\mathbf{1}, \mathbf{1}) = V_9$, $A5 = V_9$, $A4 = V_6$, $A3 = -V_7$, $A2 = V_8$, and $A1 = G3(V_3, V_9) = V_{10}$. The PDD is presented in Figure 7B. This PDD presents a set of formulae $V_O = \frac{(E+1)\cdot(I_I + G1\cdot V_I)}{(E+1)\cdot G3 + G2 + G1}$ and $I_O = \frac{-E\cdot G4\cdot(I_I + G1\cdot V_I)}{(E+1)\cdot G3 + G2 + G1}$. However, the PDD form is more compact, and it usually requires fewer mathematical operations to obtain the numerical result.

To obtain the complete result, we needed 54 steps. However, in DFS approach, the same number of steps is also required. In a DFS approach, we use the common *operation cache*, and at the end, it stores 26 elements. In a BFS, the *local cache* needs to store $|L_0| = 5$, $|L_1| = 6$, $|L_2| = 8$, $|L_3| = 3$, and $|L_4| = 2$. The maximum number of HOSCs that should be stored is at the moment at which level 2 (extraction of E) is completed. Then we have 14 HOSCs in memory—6 from origin level 1 and 8 from newly created level 2. However, starting with level 3, we can immediately release 6 HOSCs from level 1.

4.4 | PDD vs GPDD

By directly applying (28) into a 2-graph, we can discover that adding a *pair of deletion* $\binom{(p+r)}{(k+l)}$ is the equivalent of *collapsing* the branches between nodes p and r in the current graph (L-graph in GPDD) and between nodes k and l in the voltage graph (R-graph in GPDD). From the remainder, we only remove a component, which is the equivalent of *opening* branches. The successive extraction of components yields the Minty algorithm,[48] which is the basic algorithm of the common spanning trees that are determined in a GPDD.[10-15,22,43]

A PDD with HOSCs and GPDD was created more or less in the same time independently.[10,24] A GPDD as the base uses the better-known 2-graph method, while a PDD uses practically unknown HOSC arithmetic. In practice, a PDD with HOSC is not a concurrent method to a GPDD. The presented method here is a supplement that allows the symbolic analysis to be looked at from a different point of view. The results are similar. The differences are mainly in the intermediate results and algorithms that can be used in the parallel computing, as well. The performance of both methods should be compared with some benchmarks. However, this is only possible with the cooperation with the author of GPDD. Despite the large number of papers and books on GPDD,[10-15,22,43] some details of their implementation are not clear enough. Furthermore, the author does not provide his program code. In practice, a DFS implementation of a PDD with HOSCs can be treated as an alternative version of a GPDD. The single-thread version of the BFS approach saves memory, with a very similar time of performance as the DFS approach. Additionally, an order optimization algorithm was added to the BFS version. The optimization algorithm tests the influence of each remaining component on the size of the next level. This is the main bottleneck of the whole analysis. Modern computer systems are usually multicore or even multiprocessor one. Thus, the ability for parallel computation is an important asset of each modern algorithm. In contrast to a 2-graph GPDD or DFS PDD, creating a BFS PDD algorithm permits parallel analysis. The list of unique HOSCs to analyze in the current level L_i, which contains a set of HOSCs (often large) that should be modified by extracting of a component P_i can be analyzed independently and can be shared between different threads. Furthermore, testing the remaining components for the next level for the order optimization can be shared between threads, or even distributed in a computer cluster.

5 | SOME ADVANCED ANALYSIS WITH HOSC AND PDD

Considering the similarity of PDD and GPDD, a method that is only an improvement of well-known GPDD would not be worth of special presentation, unless it yielded some novel solutions that are not available using other methods. There are a few of them. We will concentrate on two—the novel approach to the hierarchical decomposition of large circuits and the determination of only selected coefficient of *s-expanded* transfer functions. Appling HOSC arithmetic to the former problem results in the method, which, similar to a plain PPD (or GPDD), maintains a *strongly compressed* form of the sum of products and can easily be applied in parallel or distributed computation. The method allows universal macromodels of the frequently repeated subcircuits to be produced as a piece of the PDD that can be reused many times as so-called meta-vertices. The application of HOSC arithmetic to the last problem allows us to determine only a few of the most crucial coefficient at powers of s terms in the numerator and/or denominator.

5.1 | Novel HPDDs

The hierarchical approach to the symbolic analysis is one of the basic methods that are used to circumvent a rapid increase in the size of the result vs the size of the problem to be solved.[7,11-15,32,33,43,49-53] However, the main drawback of these classical hierarchical methods is the representation of a subcircuit as a set of external parameters, which usually hides the internal features of the circuit. Even in analysis with GPDD,[11-15,43] the authors treat the subcircuit as a MIMO one, and they describe it as a set of the transadmittances $y_{ij} = i_i/v_j$. Then these parameters are put into an MNAM, and the DDD is determined to analyze the whole circuit. An analysis using a GPDD is cancellation free, whereas an analysis using DDD is not. On the other hand, subcircuit modeling with a set of transadmittances is not itself cancellation free. This can be proved with a trivial example.

Let us treat the 2-stage voltage divider in Figure 8 as a cascade connection between 2 simple voltage dividers. The single stage can be represented by parameters $y_{11} = G1$, $y_{12} = -G1$, $y_{21} = -G1$, and $y_{22} = G1 + G2$. The transfer voltage function of the whole circuit with y_{ij} parameters modeled is $H_u = \frac{y_{21a} \cdot y_{21b}}{(y_{22a} + y_{11b}) \cdot y_{22b} - y_{12b} \cdot y_{21b}}$. Both the transfer function and subcircuit parameters are cancellation free. By substituting y_{ij} with their actual representations, we obtain cancellations in the denominator. Such cancellations are often deeply hidden and not trivial to detect.

An additional drawback of the GPDD and DDD combination is the lack of a possibility to perform a multilevel hierarchical analysis. Circuits consist of transistors (MOS, BJT), which have a complicated but always the same small-signal equivalent. Transistors with primitive components usually form some functional blocks that can also be repeated. A multilevel hierarchical analysis is desirable and a natural approach to a modern analysis. Additionally, the results should have the form of an easy-to-reuse universal macromodel, which only needs to be determined once after which it can be *recycled* on demand. Such an approach permits an analysis to be scalable. In most papers, a hierarchical approach is treated only as a remedy for a large size of results in plain circuit. Do we have to reanalyze the same parts of circuits many times?

The main reason of the cancellations in example above is that it ignores the topology of a subcircuit. We naturally assumed that the local reference node was the global reference node at the same moment. It causes that conductance $G1$ appeared in each of the 4 parameters of y_{ij}. It is necessary to create an alternative approach. In Fontana,[54] the author proposes the generalized substitution theorem, which has a difficult hard-to-follow mathematical background. The author's aim is similar to that of the results presented below—to be able to combine larger circuits from blocks without having to recalculate everything from the very beginning. A PDD with an HOSC allows the same to be done in a simpler way. In practice, we should do nothing new to expand a PDD to its hierarchical version (HPDD). Everything is a natural consequence of the features of a PDD. We can start an analysis by determining Δ. We usually start an analysis by defining the correct HOSC to determine a function. This is usually more effective because the initial *deletions* force simplifications. However, nothing stands in the way to do it in the end. In a transfer function analysis, we remove any nodes that have become separated, until we have only one left. Then the result reaches terminal vertex **1**. However, there are a few boundary nodes in a subcircuit, which cannot be removed until we know what is connected to them. In this way, we have a PDD in progress, with a set of descendants. Each descendant is described by an HOSC. There are no HOSCs with internal nodes. The number of HOSCs is limited because descendants with the same canonical form are joined to one another. Let us present it with the following trivial example.

Example 5. In Figure 9A, a subcircuit is presented that consists of 2 resistors and 1 op-amp as an example. The boundary nodes are *i*1, *i*2, and *o*, and they are numbered 1, 2, and 3, respectively. There is one internal node *int*, and it has the number 7. In HOSC arithmetic, the nodes do not have to be assigned subsequent

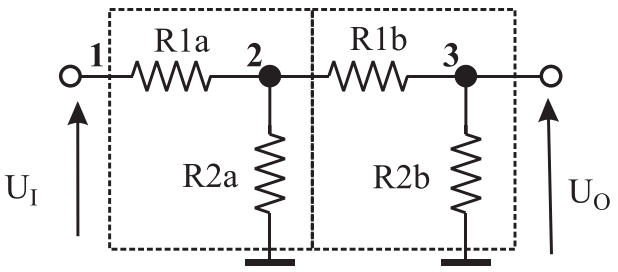

FIGURE 8 Two-stage voltage divider as a cascade connection between 2 subcircuits

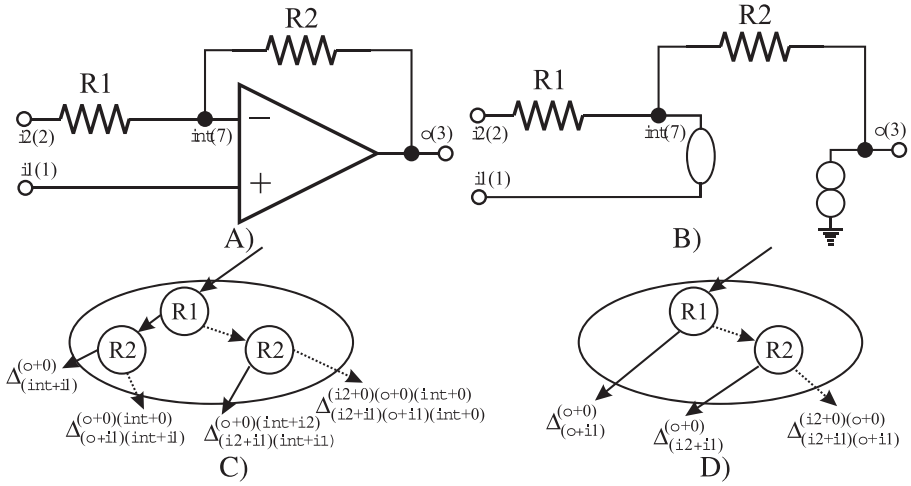

FIGURE 9 Hierarchical circuit: A, a subcircuit; B, a simplified model of the subcircuit; C, a meta-vertex that represents the subcircuit; D, a meta-vertex that represents the subcircuit, while *int* is an inaccessible internal node

numbers. On the contrary, they can be assigned by any kind of literals, but the order should be fixed. For illustrative simplicity, let us assume that the op-amp is ideal and can be modeled as a pathological pair nullator-norator called a nullor (Figure 9B). The subcircuit is small, but not so trivial. There are 4 boundary nodes, and according to previous studies,[11-15,43] the stamp matrix should be 3×3. However, it does not exist at all. Creating the stamp yields a singularity. The equation that describes current I_o is undefined. The norator allows for an arbitrary simultaneous current and voltage drop. The subcircuit has a physical meaning. By grounding either $i1$ or $i2$, we obtain the classical noninverting or inverting amplifier, respectively. Even if we assume that the op-amp has a finite gain and it is modeled as a VCVS, the output current I_o is still undefined.

The analysis begins with the HOSC with the *deletions* from the nullator $A = \Delta^{(3+0)}_{(7+1)}$. By extracting $R1\left(-, \begin{smallmatrix}(7+2)\\(7+2)\end{smallmatrix}\right)$, we obtain $A_1 = \Delta^{(3+0)}_{(7+1)} = B1$ and $A_0 = \Delta^{(3+0)(7+2)}_{(7+1)(7+2)} = \Delta^{(3+0)(7+2)}_{(2+1)(7+1)} = B2$. By extracting the last component $R2\left(-, \begin{smallmatrix}(7+3)\\(7+3)\end{smallmatrix}\right)$, we obtain $B1_1 = \Delta^{(3+0)}_{(7+1)} = C1$, $B1_0 = \Delta^{(3+0)(7+3)}_{(7+1)(7+3)} = \Delta^{(3+0)(7+0)}_{(3+1)(7+1)} = C2$, $B2_1 = \Delta^{(3+0)(7+2)}_{(2+1)(7+1)} = C3$, and $B2_0 = \Delta^{(3+0)(7+2)(7+3)}_{(2+1)(7+1)(7+3)} = \Delta^{(3+0)(3+2)(7+2)}_{(2+1)(3+1)(7+1)} = \Delta^{(2+0)(3+0)(7+0)}_{(2+1)(3+1)(7+1)} = C4$. In Figure 9C, the piece of a PDD that was obtained is presented. The numbers of the nodes were replaced by the symbolic literals. It is packaged in an ellipse because it can be treated as a generalized form of a vertex—a meta-vertex with more than 2 descendants. The internal part of the meta-vertex contains the complete information about the subcircuits, whereas the external circuits can treat such a subcircuit as a black box, called a *meta-component*, and need only to know the *pairs of deletions* that are associated with the descendants. This can be represented as $M_C\left(\begin{smallmatrix}(o+0)\\(int+i1)\end{smallmatrix}, \begin{smallmatrix}(o+0)(int+0)\\(o+i1)(int+i1)\end{smallmatrix}, \begin{smallmatrix}(o+0)(int+i2)\\(i2+i1)(int+i1)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)(int+0)\\(i2+i1)(o+i1)(int+0)\end{smallmatrix}\right)$. To include the whole subcircuit, we can consider them component by component by testing the 6 descendants or we can consider the whole structure by testing the 4 descendants simultaneously. We have assumed that node *int* should be internal and inaccessible from the outside. A *int* node can be useful, eg, in a noise analysis. However, if we are sure that this node no longer needs to be accessible, we can remove it as a separated one. This can be done during the creation of the meta-vertex, and then we obtain the simplified meta-vertex that is presented in Figure 9D with only 3 descendants because $M = M_C \xrightarrow{7} M\left(0, \begin{smallmatrix}(o+0)\\(o+i1)\end{smallmatrix}, \begin{smallmatrix}(o+0)\\(i2+i1)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)\\(i2+i1)(o+i1)\end{smallmatrix}\right)$. On the other hand, leaving *int* accessible is not a complication. Although the meta-vertex is larger, it is universal. In contrast to the hierarchical method with stamps,[11-15,43] there is no problem to leave some boundary nodes unconnected. These nodes simply become separated ones. Thus, they should be removed from the HOSCs that are associated with the descendants. This is how the meta-vertex in Figure 9C becomes the meta-vertex in Figure 9D. None of hierarchical methods that the author is aware of[7,11-15,32,33,43,49-53] deal with subcircuits with unconnected nodes. The approach to hierarchical analysis presented here allows a universal subcircuit representation to be created and then reduces it by suppressing some descendants to zero, when there are some nodes that have been short-circuited or left unconnected. Detecting the descendants that should immediately be suppressed to zero is a trivial operation that is based on HOSC arithmetic.

Determining any transfer function that contains only one meta-component is also trivial. We should define the correct HOSC for the denominators and numerators and treat them as the input of a meta-vertex. Next, we should determine the descendants in the same manner as for any primitive component. The common denominator for both inverting and noninverting transfer functions is $\Delta^{(i1+0)(i2+0)}_{(i1+0)(i2+0)} = M\left(\begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(o+i1)(i1+0)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(i2+i1)(i1+0)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)(i1+0)(i2+0)\\(i2+i1)(o+i1)(i1+0)(i2+0)\end{smallmatrix}\right) = M\left(\begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(o+0)(i1+0)(i2+0)\end{smallmatrix}, 0, 0\right) \xrightarrow{o,i1,i2} M(1,0,0) = R1$. The numerator for an inverting amplifier is $\Delta^{(i1+0)(i2+0)}_{(i1+0)(o+0)} = M\left(\begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(o+i1)(i1+0)(o+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(i2+i1)(i1+0)(o+0)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)(i1+0)(i2+0)\\(i2+i1)(o+i1)(i1+0)(o+0)\end{smallmatrix}\right) = M\left(\begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(o+0)(i1+0)(0+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(i2+0)(i1+0)(o+0)\end{smallmatrix}, 0\right) = M\left(0, \begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(i2+0)(i1+0)(o+0)\end{smallmatrix}, 0\right) \xrightarrow{o,i1,i2} M(0,-1,\ 0) = -R2$. Similarly, the numerator for a noninverting amplifier can be derived as $\Delta^{(i1+0)(i2+0)}_{(o+0)(i2+0)} = M\left(\begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(o+i1)(o+0)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(i2+i1)(o+0)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)(i1+0)(i2+0)\\(i2+i1)(o+i1)(o+0)(i2+0)\end{smallmatrix}\right) = M\left(\begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(o+0)(i1+0)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i1+0)(i2+0)\\(i2+0)(o+0)(i1+0)\end{smallmatrix}, 0\right) \xrightarrow{o,i1,i2} M(1,1,0) = R1 + R2$. These results yield $H_u = -\dfrac{R2}{R1}$ and $H_u = \dfrac{R1+R2}{R1}$ for inverting and noninverting amplifiers, respectively. As a further example, let us assume that $i1$ is grounded and that we control the circuit by the current source that flows into $i2$. Then the denominator becomes $\Delta^{(i1+0)}_{(i1+0)} = M\left(\begin{smallmatrix}(o+0)(i1+0)\\(o+i1)(i1+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i1+0)\\(i2+i1)(i1+0)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)(i1+0)\\(i2+i1)(o+i1)(i1+0)\end{smallmatrix}\right) \xrightarrow{o,i1,i2} M(0,0,1) = 1$ and $J_2 = -R2$. By forcing the current into $i1$, having i_2 grounded, we obtain denominator $\Delta^{(i2+0)}_{(i2+0)} = M\left(\begin{smallmatrix}(o+0)(i2+0)\\(o+i1)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(o+0)(i2+0)\\(i2+i1)(i2+0)\end{smallmatrix}, \begin{smallmatrix}(i2+0)(o+0)(i2+0)\\(i2+i1)(o+i1)(i2+0)\end{smallmatrix}\right) \xrightarrow{o,i1,i2} M(0,0,0) = 0$. The transfer function tends to infinity. Yet we try to force a current into the nullator.

A circuit can be decomposed into subcircuits in an arbitrary way. There are no preliminary rules except that each primitive component must be completely included in the subcircuit, eg, separating the pairs of pathological components between 2 subcircuits is not permitted. In both a circuit and a subcircuit, the number of voltage-dependent pathological components should be equal to the current-dependent ones.

Connecting the subcircuits that are modeled as meta-vertices always gives the correct transfer function, if it only exists. Let us connect the 2 subcircuits in Figure 9A into the structure in Figure 10A. We obtain an amplifier with feedback that is made from other op-amp sections. However, recognizing the topology and knowledge about the internal structure is not necessary to determine any transfer function. Only the descendants from the meta-vertices are needed.

The circuit is only made from 2 subcircuits that can be described as follows: $M_a\left(\begin{smallmatrix}(3+0)\\(3+5)\end{smallmatrix}, \begin{smallmatrix}(3+0)\\(1+5)\end{smallmatrix}, \begin{smallmatrix}(1+0)(3+0)\\(1+5)(3+5)\end{smallmatrix}\right)$ and $M_b\left(\begin{smallmatrix}(5+0)\\(5+0)\end{smallmatrix}, \begin{smallmatrix}(5+0)\\(3+0)\end{smallmatrix}, \begin{smallmatrix}(3+0)(5+0)\\(3+0)(5+0)\end{smallmatrix}\right)$. Let us determine $H_u = \Delta^{(1+0)}_{(3+0)}/\Delta^{(1+0)}_{(1+0)}$, $J = \Delta^{(1+0)}_{(3+0)}/\Delta$. We have 3 HOSCs to determine $\Delta = \mathrm{A1}$, $\Delta^{(1+0)}_{(3+0)} = A2$, and $\Delta^{(1+0)}_{(1+0)} = A3$. As the first component, M_a will be extracted along with eliminating node 1. We obtain $\Delta = M_a\left(\begin{smallmatrix}(3+0)\\(3+5)\end{smallmatrix}, \begin{smallmatrix}(3+0)\\(1+5)\end{smallmatrix}, \begin{smallmatrix}(1+0)(3+0)\\(1+5)(3+5)\end{smallmatrix}\right) \xrightarrow{1} M_a\left(0, 0, \begin{smallmatrix}(3+0)\\(3+5)\end{smallmatrix}\right) = M_a(0,0,B1) = B1$, $\Delta^{(1+0)}_{(3+0)} = M_a\left(\begin{smallmatrix}(3+0)(1+0)\\(3+5)(3+0)\end{smallmatrix}, \begin{smallmatrix}(3+0)(1+0)\\(1+5)(3+0)\end{smallmatrix}, \begin{smallmatrix}(1+0)(3+0)(1+0)\\(1+5)(3+5)(3+0)\end{smallmatrix}\right) \xrightarrow{1} M_a\left(0, -\begin{smallmatrix}(3+0)\\(3+0)\end{smallmatrix}, 0\right) = M_a(0,B2,0) = R2_a \cdot B2$, and $\Delta^{(1+0)}_{(1+0)} =$

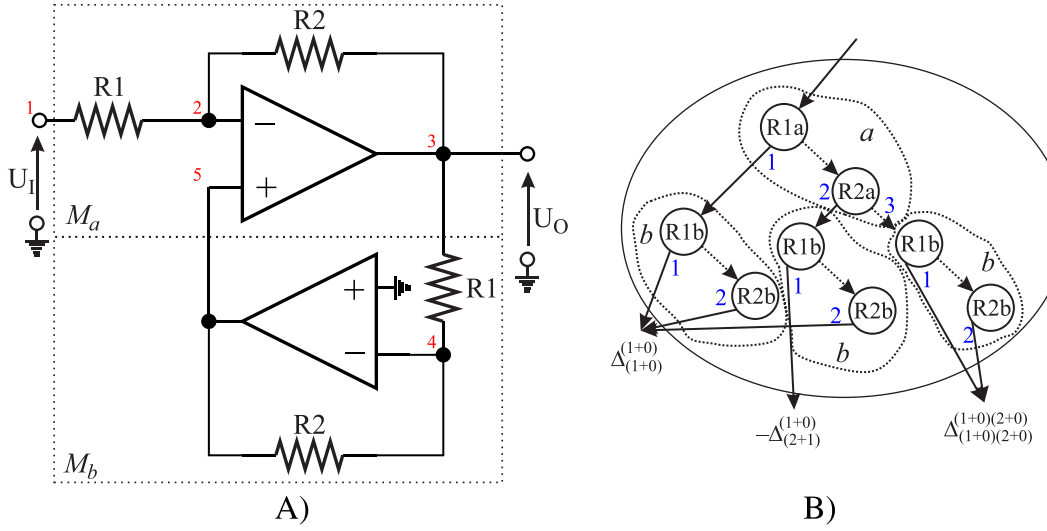

FIGURE 10 A circuit made of 2 subcircuits: A, schematic diagram; B, meta-vertex [Colour figure can be viewed at wileyonlinelibrary.com]

$M_a\left(\begin{smallmatrix}(3+0)(1+0)\\(3+5)(1+0)\end{smallmatrix},\begin{smallmatrix}(3+0)(1+0)\\(1+5)(1+0)\end{smallmatrix},\begin{smallmatrix}(1+0)(3+0)(1+0)\\(1+5)(3+5)(1+0)\end{smallmatrix}\right)\xrightarrow{1} M_a\left(\begin{smallmatrix}(3+0)\\(3+5)\end{smallmatrix},-\begin{smallmatrix}(3+0)\\(5+0)\end{smallmatrix},0\right)=M_a(B1,B3,0)=R1_a\cdot B1+R2_a\cdot B3$. In the next step, we have 3 unique nonzero descendants, and we will extract M_b and eliminate nodes 3 and 5. We obtain $B1=\Delta^{(3+0)}_{(3+5)}=M_b\left(\begin{smallmatrix}(5+0)(3+0)\\(5+0)(3+5)\end{smallmatrix},\begin{smallmatrix}(5+0)(3+0)\\(3+0)(3+5)\end{smallmatrix},\begin{smallmatrix}(3+0)(5+0)(3+0)\\(3+0)(5+0)(3+5)\end{smallmatrix}\right)\xrightarrow{3,5}M_b(1,1,0)=R1_b+R2_b$, $B2=-\Delta^{(3+0)}_{(3+0)}=M_b\left(-\begin{smallmatrix}(5+0)(3+0)\\(5+0)(3+0)\end{smallmatrix},-\begin{smallmatrix}(5+0)(3+0)\\(3+0)(3+0)\end{smallmatrix},-\begin{smallmatrix}(3+0)(5+0)(3+0)\\(3+0)(5+0)(3+0)\end{smallmatrix}\right)\xrightarrow{3,5}M_b(-1,0,0)=-R1_b$, $B3=-\Delta^{(3+0)}_{(5+0)}=M_b\left(-\begin{smallmatrix}(5+0)(3+0)\\(5+0)(5+0)\end{smallmatrix},-\begin{smallmatrix}(5+0)(3+0)\\(3+0)(5+0)\end{smallmatrix},-\begin{smallmatrix}(3+0)(5+0)(3+0)\\(3+0)(5+0)(5+0)\end{smallmatrix}\right)\xrightarrow{3,5}M_b(0,1,0)=R2_b$. Thus, $\Delta=R1_b+R2_b$, $\Delta^{(1+0)}_{(3+0)}=-R2_a\cdot R1_b$, and $\Delta^{(1+0)}_{(1+0)}=R1_a\cdot(R1_b+R2_b)+R2_a\cdot R2_b$.

An alternative approach is to create a meta-vertex that is composed of smaller subcircuits, which are described as meta-vertices M_a and M_b. We obtain a new meta-vertex $M_{ab}=M_a\oplus M_b$ (the notation has been taken from Fontana[54]), which contains both subcircuits. As a result, we obtain the HPDD that is presented in Figure 10B. The graph is larger than in the case of the direct determination of the transfer function. Nevertheless, it is appropriate in any case and is always true. It can be used for larger subcircuit once again, or it can be reduced to determine the direct transfer function.

In Figure 10B, a drawback of a hierarchical analysis, even in the form of an HPDD, is presented—a redundancy. There are 2 vertices that are associated with $R2_b$ that have common descendants: 0 and $\Delta^{(1+0)}_{(1+0)}$. These vertices are nonunique. However, they belong to different instances of the same meta-vertex. Although eliminating such nonuniqueness is possible, it requires communication between the instances. The main assumption of an HPPD is that the instances do not interfere with one another. The main postprocessing should be performed *inside* the meta-vertices in parallel or distributed threads. Moreover, the uniform structure inside of each instance of the same meta-vertex is an advantage, eg, the numerical value of each instance is the same function of the descendants and can be determined once as a function of the values, which are obtained from the descendants. This accelerates the calculations significantly and compensates for any possible redundancy. In contrast to subcircuits that are modeled as stamps,[11-15,43] a redundancy never yields cancellations, even in a multilevel hierarchical structure.

The circuit from Example 5 is very simple and does not require any hierarchical analysis, so in this case, HPDD application is moderately profitable. However, it is small enough to present in a graphical form and to explain the method. To show the advantages of this approach, let us look at a small-signal model of some actual device.

Example 6. In Figure 11, the submicron small-signal MOSFET model is presented. The model has 4 boundary nodes (*G*, *D*, *S*, and *B*) and 2 internal nodes (D' and S'). In Figure 12, the small-signal MOSFET model is presented in the form of a list of vertices. A graphical representation would be too large to be clear. Frankly, the list representation is closer to the form that is stored in a computer's memory. However, rather than symbols of the vertices as descendants, a direct reference to them is used.

The *bolded descendants* in the *EPi* form represent the boundary HOSCs—the only HOSCs that are accessible to the master circuit and the only HOSCs that are required to connect this structure to a larger one. They are presented in Figure 13.

The main vertex is V_2. To determine V_2 as a function of each *EPi*, we can use the superposition rule. We should assume $\forall j\neq i\ EPj=0$ and simplify the internal structure by removing the zero vertices. In this way, we can find that, eg, $V_2=...+g_m\cdot EP17+...+g_m\cdot(sC_{db}+G_{db})\cdot R_D\cdot EP19+...+g_m\cdot(sC_{db}+G_{db})\cdot EP18+...+g_{ds}\cdot sC_{gb}\cdot EP25$. Only a few components are presented. Most of them are long and complicated. However, the form presented above is usually less effective than the PDD. It is better to determine the influence of all of the values at the same moment simultaneously from the PDD. However, the HPDD can be transformed into cancellation free and can be as effective as PDD the sequence of expressions. The structure can be redundant in some applications as was shown in the previous example. However, it is complete. Generally, each application of such a structure may reduce the number of vertices (a fast operation), but will never create any new vertices (an advanced and slow operation).

The small-signal MOSFET model has 4 boundary nodes, and its HPDD model generates 26 descendants. The classical transadmittance stamp (like in GPDD[11-15,43]) needs only $4\times4=16$ parameters. However, the stamp consists of 16 parameters, which sometimes are quite complicated transfer functions and these should be included in the master circuit like other primitive components. This decreases the number of nodes by 2, but increases the number of parameters from 12 to 16. In a combined GPDD + DDD, such approach is beneficial,[11-15,43] but a GPDD + GPDD can be controversial. Unfortunately, the GPDD + DDD approach can be difficult to generalize for an analysis of a multilevel hierarchy. By contrast, the model in Figure 12 does not need any parameters to be generated, and there is no parameter

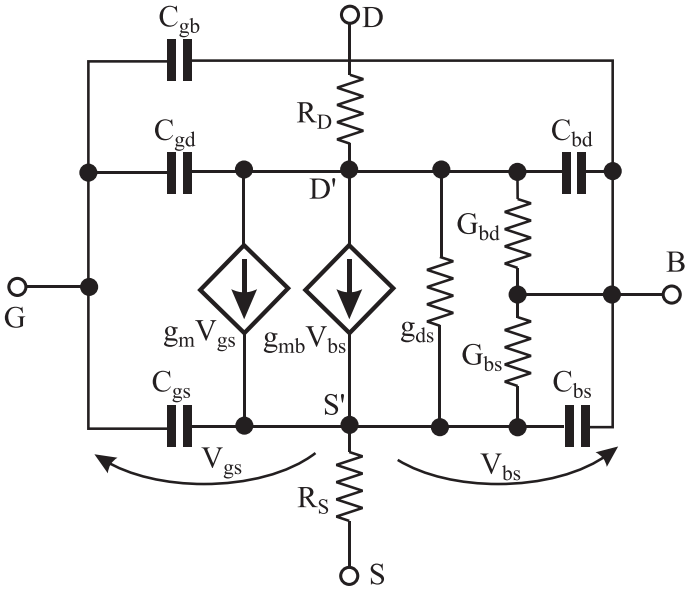

FIGURE 11 A small-signal MOSFET model

V_2: $g_m(V_3, V_6)$
V_3: $g_{mb}(V_4, V_{14})$
V_4: $g_{ds}(V_5, V_7)$
V_5: $C_{sb}(V_8, V_{15})$
V_6: $C_{sb}(V_9, V_{45})$
V_7: $C_{sb}(V_{10}, V_{24})$
V_8: $G_{sb}(V_{11}, V_{15})$
V_9: $G_{sb}(V_{12}, V_{45})$
V_{10}: $G_{sb}(V_{13}, V_{24})$
V_{11}: $C_{db}(V_{16}, V_{26})$
V_{12}: $C_{db}(V_{17}, V_{37})$
V_{13}: $C_{db}(V_{18}, V_{24})$
V_{14}: $C_{db}(V_{19}, V_{24})$
V_{15}: $C_{db}(V_{20}, V_{24})$
V_{16}: $G_{db}(V_{21}, V_{26})$
V_{17}: $G_{db}(V_{28}, V_{37})$
V_{18}: $G_{db}(V_{22}, V_{24})$
V_{19}: $G_{db}(V_{23}, V_{24})$
V_{20}: $G_{db}(V_{25}, V_{24})$
V_{21}: $C_{gs}(V_{27}, V_{36})$
V_{22}: $C_{gs}(V_{29}, V_{35})$
V_{23}: $C_{gs}(V_{30}, V_{48})$
V_{24}: $C_{gs}(V_{32}, V_{44})$
V_{25}: $C_{gs}(V_{33}, V_{42})$
V_{26}: $C_{gs}(V_{34}, V_{31})$
V_{27}: $C_{gb}(V_{63}, V_{62})$
V_{28}: $C_{gb}(V_{38}, V_{47})$
V_{29}: $C_{gb}(V_{39}, V_{52})$
V_{30}: $C_{gb}(V_{40}, V_{47})$
V_{31}: $C_{gb}(V_{41}, V_{44})$
V_{32}: $C_{gb}(V_{43}, V_{44})$
V_{33}: $C_{gb}(V_{46}, V_{42})$
V_{34}: $C_{gb}(V_{56}, V_{71})$
V_{35}: $C_{gb}(V_{49}, V_{44})$
V_{36}: $C_{gb}(V_{50}, V_{42})$
V_{37}: $C_{gb}(V_{51}, V_{44})$
V_{38}: $R_S(V_{61}, V_{63})$
V_{39}: $R_S(V_{60}, V_{63})$
V_{40}: $R_S(V_{58}, V_{63})$
V_{41}: $R_S(V_{57}, V_{56})$
V_{42}: $R_S(V_{55}, V_{62})$
V_{43}: $R_S(V_{54}, V_{56})$
V_{44}: $R_S(V_{72}, V_{71})$
V_{45}: $R_S(V_{59}, V_{53})$
V_{46}: $R_S(V_{65}, V_{63})$
V_{47}: $R_S(V_{66}, V_{62})$
V_{48}: $R_S(\boldsymbol{EP21}, \boldsymbol{EP22})$
V_{49}: $R_S(V_{78}, V_{79})$
V_{50}: $R_S(V_{67}, V_{63})$
V_{51}: $R_S(V_{64}, V_{56})$
V_{52}: $R_S(V_{68}, V_{62})$
V_{53}: $C_{gd}(\boldsymbol{EP5}, V_{71})$
V_{54}: $C_{gd}(V_{73}, V_{72})$
V_{55}: $C_{gd}(\boldsymbol{EP0}, V_{72})$
V_{56}: $C_{gd}(V_{74}, V_{71})$
V_{57}: $C_{gd}(V_{75}, V_{72})$
V_{58}: $C_{gd}(\boldsymbol{EP14}, V_{70})$
V_{59}: $C_{gd}(\boldsymbol{EP15}, V_{72})$
V_{60}: $C_{gd}(V_{76}, V_{78})$
V_{61}: $C_{gd}(\boldsymbol{EP17}, V_{78})$
V_{62}: $C_{gd}(\boldsymbol{EP7}, V_{71})$
V_{63}: $C_{gd}(\boldsymbol{EP11}, V_{79})$
V_{64}: $C_{gd}(V_{77}, V_{72})$
V_{65}: $C_{gd}(\boldsymbol{EP2}, V_{69})$
V_{66}: $C_{gd}(\boldsymbol{EP20}, V_{72})$
V_{67}: $C_{gd}(\boldsymbol{EP13}, V_{78})$
V_{68}: $C_{gd}(V_{80}, V_{72})$
V_{69}: $R_D(\boldsymbol{EP8}, \boldsymbol{EP2})$
V_{70}: $R_D(\boldsymbol{EP3}, \boldsymbol{EP4})$
V_{71}: $R_D(\boldsymbol{EP6}, \boldsymbol{EP7})$
V_{72}: $R_D(\boldsymbol{EP1}, \boldsymbol{EP0})$
V_{73}: $R_D(\boldsymbol{EP9}, \boldsymbol{EP2})$
V_{74}: $R_D(\boldsymbol{EP10}, \boldsymbol{EP11})$
V_{75}: $R_D(\boldsymbol{EP12}, \boldsymbol{EP13})$
V_{76}: $R_D(\boldsymbol{EP16}, \boldsymbol{EP11})$
V_{77}: $R_D(\boldsymbol{EP18}, \boldsymbol{EP19})$
V_{78}: $R_D(\boldsymbol{EP23}, \boldsymbol{EP13})$
V_{79}: $R_D(\boldsymbol{EP24}, \boldsymbol{EP11})$
V_{80}: $R_D(\boldsymbol{EP25}, \boldsymbol{EP7})$

FIGURE 12 The internal structure of the small-signal MOSFET model

$\boldsymbol{EP0} = \Delta^{(G+B)(S+B)}_{(G+B)(S+B)}$
$\boldsymbol{EP1} = \Delta^{(G+B)(S+B)(D+B)}_{(G+B)(S+B)(D+B)}$
$\boldsymbol{EP2} = \Delta^{(S+B)}_{(S+B)}$
$\boldsymbol{EP3} = \Delta^{(S+G)(D+G)}_{(S+B)(D+G)}$
$\boldsymbol{EP4} = \Delta^{(S+G)}_{(S+B)}$
$\boldsymbol{EP5} = \Delta^{(D+B)}_{(G+B)}$
$\boldsymbol{EP6} = \Delta^{(G+B)(D+B)}_{(G+B)(D+B)}$
$\boldsymbol{EP7} = \Delta^{(G+B)}_{(G+B)}$
$\boldsymbol{EP8} = \Delta^{(S+B)(D+G)}_{(S+B)(D+G)}$
$\boldsymbol{EP9} = \Delta^{(S+B)(D+B)}_{(S+B)(D+B)}$
$\boldsymbol{EP10} = \Delta^{(D+B)}_{(D+B)}$
$\boldsymbol{EP11} = \Delta$
$\boldsymbol{EP12} = \Delta^{(S+G)(D+B)}_{(S+G)(D+B)}$
$\boldsymbol{EP13} = \Delta^{(S+G)}_{(S+G)}$
$\boldsymbol{EP14} = -\Delta^{(D+S)}_{(S+B)}$
$\boldsymbol{EP15} = -\Delta^{(S+B)(D+B)}_{(G+B)(S+B)}$
$\boldsymbol{EP16} = \Delta^{(D+S)}_{(D+S)}$
$\boldsymbol{EP17} = -\Delta^{(D+S)}_{(S+G)}$
$\boldsymbol{EP18} = \Delta^{(S+B)(D+B)}_{(S+G)(D+B)}$
$\boldsymbol{EP19} = \Delta^{(S+B)}_{(S+G)}$
$\boldsymbol{EP20} = -\Delta^{(G+B)(D+S)}_{(G+B)(S+B)}$
$\boldsymbol{EP21} = \Delta^{(S+G)(D+G)}_{(G+B)(S+B)}$
$\boldsymbol{EP22} = -\Delta^{(D+G)}_{(G+B)}$
$\boldsymbol{EP23} = \Delta^{(S+G)(D+G)}_{(S+G)(D+G)}$
$\boldsymbol{EP24} = \Delta^{(D+G)}_{(D+G)}$
$\boldsymbol{EP25} = \Delta^{(G+B)(D+S)}_{(G+B)(D+S)}$

FIGURE 13 The boundary higher order summative cofactor for the small-signal MOSFET model

to include in the master circuit. The MOSFET model can be analyzed in the circuit's primitive component by component, or ... en bloc. In the former approach, there is a risk of obtaining 65 vertices and testing about 130 descendants. In the latter approach, 24 descendants are tested.

In Hu et al[55] and Shi et al,[56] the authors proposed a concept for automatically generating simplified symbolic macromodels. To obtain simplified models, they include a complex model of MOSFET transistors to get a large 1-level plain circuit. Then the simplification is performed by assuming that some primitive components are 0 or tend to infinity. In a GPDD and PDD, these 2 operations are trivial by removing each vertex that is associated with the parameters and replacing them by their 1- or 0-descendants. Depending on the role of the MOSFET, their model reduces in different ways. Sometimes, the entire transistor can be eliminated. From papers,[55,56] it is not clear whether the author used their hierarchical approach from Song and Shi[13] and Shi et al.[43] The meta-vertices presented here allow the influence of any internal parameter or values that come from the descendants on the value of the whole meta-vertex being tested. We can test what the change of a vertex V_j value is, if some parameter g_i becomes 0 ($V_j(g_i = 0)$) or if it becomes very large ($V_j(g_i \to \infty)$), or we can even determine the sensitivity of vertex V_j to a variation in parameter g_i as $S_{g_i}^{V_j} = \frac{g_i \cdot V_j(g_i = 0)}{V_j}$. The denominator is the original value of a vertex, and the numerator is its value with the 0-descendants of the vertices that are associated with g_i suppressed to zero; see Figure 14 for sensitivity $S_{g_m}^{V_2}$ for the MOSFET model in Figure 11. Because each meta-vertex can be supported by a separated thread or node in a distributed system, the size of the circuit can be larger than the one tested in Hu et al[55] and Shi et al.[56]

Example 7—a large actual circuit. On the website,[57] there is a benchmark that is based on the actual structure of the op-amp μA741. The circuit contains 20 BJTs, 11 resistors, and 1 capacitor. Each BJT model consists of 1 transconductance, 4 resistors, and 2 to 3 capacitors depending on the type of transistors—n-p-n, lateral p-n-p, or substrate p-n-p. Including each model of transistors into the circuit, we obtain 217 primitive components and a circuit with 75 nodes. The circuit is challenging. The detailed results cannot be presented in a reasonable form in the paper. In Shi,[15] a competitive analysis of μA741 with a GPDD was performed. However, the author declared only 24 nodes and 165 edges. The model of transistors was simplified. There are no serial parasitic resistances in the base or the collector. In our experiments, there were no simplifications. The circuit was analyzed in the exact form as in the benchmark. The algorithms were implemented in C++ and compiled with a full optimization of the resulting code. The test was performed on a computer with an Intel Xeon CPU E5-2630 v3 @ 2.40 GHz processor, with 8GB RAM, in Windows 7. In the 1-level analysis with the order optimization, we obtained 435 470 vertices. The generation time was 2359.27 seconds. However, the bottleneck of the implementation is heuristic for optimizing the order. By switching it off, we reduced the time to about 59 seconds. However, a prefixing operation is needed, and we obtained more than million vertices. In the future implementation, a parallel version of optimization will be used, and the time is expected to be smaller. Anyway, such a large PDD structure is not very effective for a numerical recalculation. The semisymbolic exact values in the form of s polynomials were obtained in 20.13 seconds.

For the hierarchical analysis, the whole circuit was divided into 4 subcircuits: the biasing reference network, the input stage, the middle stage, and the output stage, like in the benchmark in Rodanski.[57] Each transistor was treated as a subcircuit. The same type of transistors generates the same meta-vertex. Although the values that

$$S_{g_m}^{V_2} =$$

$V_2: g_m(0, V_6)$	$V_{38}: R_S(V_{61}, V_{63})$	$V_{59}: C_{gd}(\boldsymbol{EP15}, V_{72})$	$V_{71}: R_D(\boldsymbol{EP6}, \boldsymbol{EP7})$
$V_6: C_{sb}(V_9, V_{45})$	$V_{44}: R_S(V_{72}, V_{71})$	$V_{61}: C_{gd}(\boldsymbol{EP17}, V_{78})$	$V_{72}: R_D(\boldsymbol{EP1}, \boldsymbol{EP0})$
$V_9: G_{sb}(V_{12}, V_{45})$	$V_{45}: R_S(V_{59}, V_{53})$	$V_{62}: C_{gd}(\boldsymbol{EP7}, V_{71})$	$V_{74}: R_D(\boldsymbol{EP10}, \boldsymbol{EP11})$
$V_{12}: C_{db}(V_{17}, V_{37})$	$V_{47}: R_S(V_{66}, V_{62})$	$V_{63}: C_{gd}(\boldsymbol{EP11}, V_{79})$	$V_{77}: R_D(\boldsymbol{EP18}, \boldsymbol{EP19})$
$V_{17}: G_{db}(V_{28}, V_{37})$	$V_{51}: R_S(V_{64}, V_{56})$	$V_{64}: C_{gd}(V_{77}, V_{72})$	$V_{78}: R_D(\boldsymbol{EP23}, \boldsymbol{EP13})$
$V_{28}: C_{gb}(V_{38}, V_{47})$	$V_{53}: C_{gd}(\boldsymbol{EP5}, V_{71})$	$V_{66}: C_{gd}(\boldsymbol{EP20}, V_{72})$	$V_{79}: R_D(\boldsymbol{EP24}, \boldsymbol{EP11})$
$V_{37}: C_{gb}(V_{51}, V_{44})$	$V_{56}: C_{gd}(V_{74}, V_{71})$		

FIGURE 14 The numerator of the sensitivity $S_{g_m}^{V_2}$

are associated with different transistors are different, they are the same in different instances of the meta-vertices that are associated with the same component. In the resulting HPDD structure, 4608 meta-vertices were obtained (including 1764 for the input stage and 2059 for the output stage). Determining the complete result took only 7.23 seconds (including the optimization of the component order in the single-thread version). This time, the semisymbolic exact values in the form of s polynomials were obtained in 2.39 seconds. However, the first HOSC as a semisymbolic polynomial was determined in 2.15 seconds, the second in 0.16 seconds, and the rest in 0.08. The longer the operations are performed, the faster the results are obtained. The results from earlier operations are reused.

The above example shows that despite the redundancy of a hierarchical analysis, the results can be obtained faster and more efficiently. Additionally, the HPDD structure maintains the sum of the products form of values, although it is strongly compressed. It is possible to determine the denominating terms with a simplification after generating method.

5.2 | *s*-expanded transfer functions

When a transfer function is a rational function of 2 polynomials of complex frequency variable s, it is called an s-expanded transfer function.[8] The main advantage of this form is that it permits the dynamic futures of a circuit analysis such as the stability of a circuit, the position of zeroes and poles on the complex surface, and the transient response. In Shi and Tan,[8] the authors created a compact DDD, and then by postprocessing of the original diagram, they obtained an s-expanded form. The main disadvantage of such approach is the size of the results. An s-expanded form is usually much larger than a compact one and we obtain coefficients at s in the higher order, which are less crucial than that in the lower one. For example, the exact voltage transfer function from the input to the output of op-amp μA741 in an open-loop configuration is described as the rational function of 2 polynomials of s. The power of the denominator is 68, and the power of the numerator is 65. However, the coefficients at the highest power are extremely small. The ratio of the coefficients at the highest power to the zero power for the numerator is $n_{65}/n_0 = 3.784 \cdot 10^{-602}$ and for the denominator is $d_{68}/d_0 = 2.667 \cdot 10^{-609}$; see example 7. Considering the highest order terms is unnecessary and even difficult with a double-precision floating number. It is more reasonable to obtain the first few dominating coefficients. In Pierzchala and Rodanski,[58] the authors used a special representation of the terms in the result, which they called a *road map*, to avoid generating terms with the power of s that were different from those that were desirable. In Fontana,[59] the author uses the extended Cochrun-Grabel algorithm. The former approach is similar to a BDD-like structure. The last one is cognitively very interesting, but complicated. By applying a PDD and an HOSC, the solution to the problem is natural and simple.

A PDD with an HOSC analysis can begin from any component, and it can be performed in an arbitrary order. Any primitive component can represent some passive value and have the form $s \cdot P_i$, only if it has some physical sense. Thus, we can have, eg, susceptance sC (capacitance), reactance sL (inductance), and transreactance sM (mutual inductance). If the algorithm for creating a PDD starts with s-dependent components first, after they are extracted, a set of descendants is obtained. A path from the root leads to each descendant. The number of 1-descendants from the associated s-dependent components determines the power of s in the boundary descendant. This operation is similar to creating an HPDD model of a subcircuit. However, the subcircuit is now a *distributed* subcircuit that contains all of the s-dependent components. If we want to determine only one coefficient at ith power (s^i) or a few of them, we should suppress the rest of descendants with the unwanted power to zero. Then we should simplify the remaining part of a PDD by removing the vertices that are zero. To determine the coefficients, we should only determine the values of the remaining descendants. Let us check this with the following illustrative example.

Example 8. Let us recalculate example 1 from Fontana[59]; see Figure 15. The aim of author example was to determine only the numerator of the voltage transfer function $H_u(s) = \dfrac{V_O(s)}{V_{S(s)}}$ in its s-expanded form. It is obvious from (30) that we should only determine $\Delta_{(O+0)}^{(S+0)}$. The circuit contains 3 s-dependent components—three capacitors $C1\binom{\cdots,\ (1+0)}{\cdots,\ (1+0)}$, $C2\binom{\cdots,\ (2+0)}{\cdots,\ (2+0)}$, and $C3\binom{\cdots,\ (S+2)}{\cdots,\ (S+2)}$. Generally, it yields a PDD structure like the one in Figure 16A where the descendants are $\#1 = \Delta_{(S+0)(1+0)(2+0)}^{(S+0)(1+0)(2+0)}\{3\}$, $\#2 = \Delta_{(S+0)(2+0)}^{(S+0)(2+0)}\{2\}$, $\#3 = \Delta_{(S+2)(1+0)}^{(S+2)(1+0)}\{2\}$,

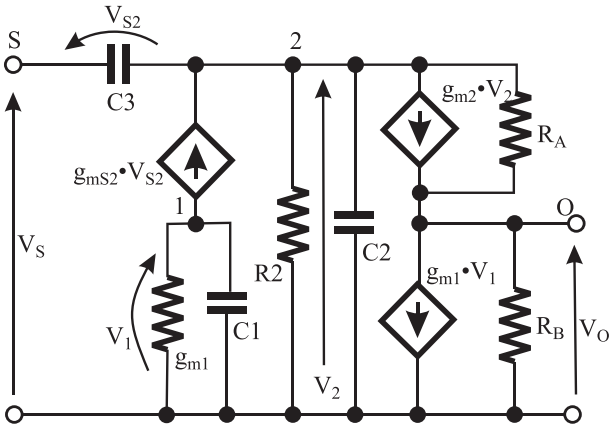

FIGURE 15 Example of the circuit from Fontana[59]

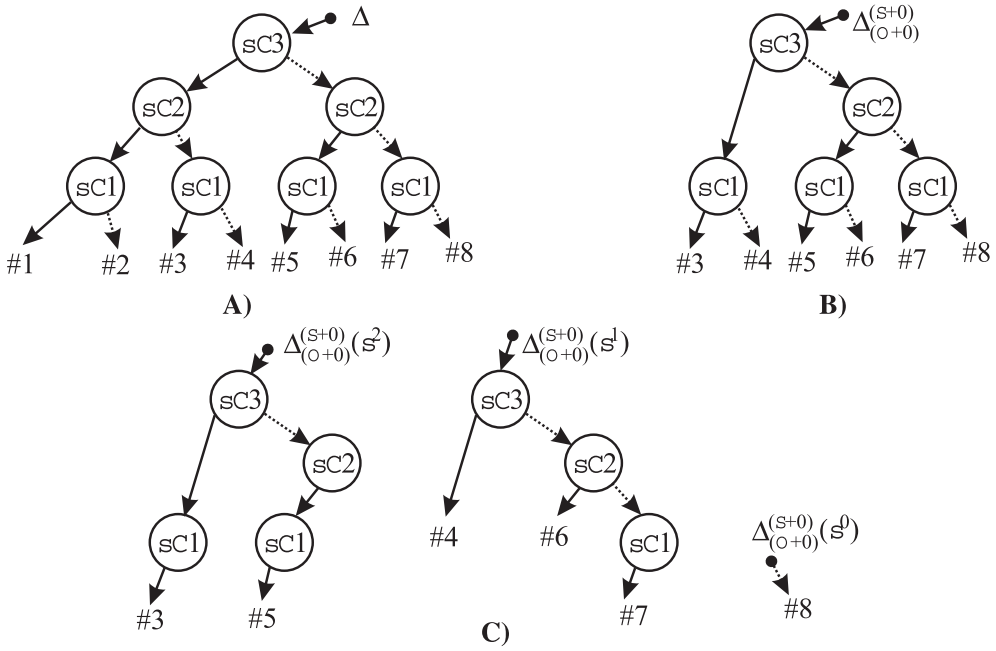

FIGURE 16 s-expansion of the cofactor $\Delta^{(S+0)}_{(O+0)}$

$\#4 = \Delta^{(S+2)}_{(S+2)}\{1\}$, $\#5 = \Delta^{(1+0)(2+0)}_{(1+0)(2+0)}\{2\}$, $\#6 = \Delta^{(2+0)}_{(2+0)}\{1\}$, $\#7 = \Delta^{(1+0)}_{(1+0)}\{1\}$, and $\#8 = \Delta\{0\}$. The power of s is presented in the braces. After adding the *deletions* from the numerator $^{(S+0)}_{(O+0)}$, 2 descendants become suppressed to 0 (Figure 16B). Only $\#3 = \Delta^{(S+0)(1+0)(2+0)}_{(S+2)(1+0)(O+0)}\{2\}$, $\#4 = \Delta^{(S+0)(2+0)}_{(S+2)(O+0)}\{1\}$, $\#5 = \Delta^{(S+0)(1+0)(2+0)}_{(1+0)(2+0)(O+0)}\{2\}$, $\#6 = -\Delta^{(S+0)(2+0)}_{(2+0)(O+0)}\{1\}$, $\#7 = -\Delta^{(S+0)(1+0)}_{(1+0)(O+0)}\{1\}$, and $\#8 = \Delta^{(S+0)}_{(O+0)}\{0\}$ are left.

To determine the coefficients at a selected power of s, we can separate them by suppressing to 0 descendants with the unwanted power. In the example, this leads to a structure like the one in Figure 16C. However, instead of a PDD with one root, we obtain 3-root PDD, and the descendants that need to be determined are the same as in the 1-root case. The complete result is presented in Figure 17.

Creating s-expanded numerators and denominators for any circuit is natural with PDD. Only a few minor modifications in the order of component extraction are required to get the s-expanded form directly. In contrast to the methods described in Fontana,[59] there are no special cases of degeneration in the circuits (see example 4 in Fontana[59]).

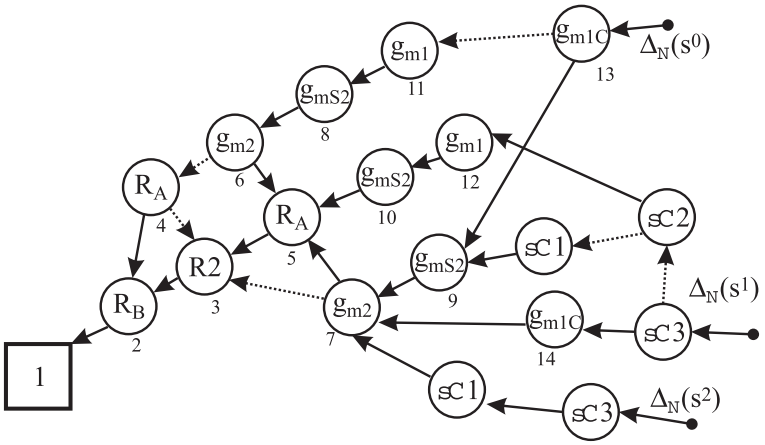

FIGURE 17 *s*-expanded numerator of the voltage transfer function in example 8

s^3 ·

V_2: $C_{sb}(V_4, V_3)$	V_5: $C_{gb}(V_8, V_{10})$	V_{15}: $C_{gd}(\mathbf{0}, V_{57})$	V_{64}: $R_S(\boldsymbol{EP19}, \boldsymbol{EP20})$
V_3: $C_{gs}(V_5, V_{10})$	V_8: $C_{db}(\mathbf{0}, V_{15})$	V_{57}: $R_D(V_{70}, V_{64})$	V_{70}: $R_S(\boldsymbol{EP10}, \boldsymbol{EP9})$
V_4: $C_{gs}(V_6, V_5)$	V_{10}: $C_{db}(V_{15}, V_{57})$		

FIGURE 18 The coefficient at s^3 in a macromodel in Figure 11

Example 9. The *s*-expanded form can also be used in a multilevel hierarchical analysis. Then we obtain a *meta-vertex* with more than one root with different powers of *s*. Such a *meta-vertex* is a function of the descendants and *s*: $M(s, d_0, d_1, d_2, ..., d_m)$. Let us return to the MOSFET transistor model in Figure 11. The subcircuit contains 5 capacitors. However, the coefficients at *s* at the fourth and fifth powers are zeros. This macromodel generates the maximum third power of *s*; see Figure 18.

This means that the coefficient at s^3 of V_2 is $(sC_{sb} \cdot sC_{gs} \cdot (sC_{db} + sC_{gd}) \cdot V_{57} + (sC_{sb} + sC_{gs}) \cdot (sC_{gb} \cdot (sC_{db} + sC_{gd}) + sC_{db} \cdot sC_{gd})) \cdot (R_D \cdot (R_S \cdot EP7 + EP0) + R_S \cdot EP6 + EP1)$ and depends on only 4 descendants.

6 | CONCLUSIONS

In this paper, a new concept for the symbolic analysis of circuits that is based on the HOSC with a PDD was presented. The mathematical background of an HOSC is not commonly known. Although the concept of HOSC is derived from the classical nodal analysis method, it is rather relative to the topological methods such as the 2-graph method. The HOSC combines these 2 classical methods into one method. Another feature of the HOSC application is that each aspect of an analysis always reduces to the manipulation on some *pairs of deletions*. The numerator and the denominator of some circuit functions are represented by an HOSC that comes from the topology of the input and output gates of the signals in the circuit. The consideration of each component in the analysis also leads to additional *pairs of deletions* that are associated with the gates to which the components are connected. The arithmetic of manipulating an HOSC and creating a PDD was developed for a computer application, with the strong impact for the future parallel and distributed implementation. On the other hand, handwritten HOSC manipulations also permit many problems in the modern symbolic analysis of electric networks to be solved. Some of these solutions are presented in this paper only as a draft. Others, like an application of an HOSC with a PDD in the structural synthesis of new circuits, were completely omitted. Some of the more advanced applications, which were only presented here as a draft or which were omitted, could be presented in separate papers. The main aim of this paper was to popularize an HOSC with PDD as a universal tool in circuit analysis. The author has been using them successfully in handwritten and computer symbolic analysis and in the synthesis of actual electronic circuits for years. Many typical and peculiar problems from linear and weekly nonlinear circuit problems have been solved. The paper only presents a brief overview of them. The author believes and hopes that the method presented here will become one of the standard methods of symbolic analysis, which joins 2 other commonly known methods into one and inherits the advantages of both of them.

ACKNOWLEDGEMENTS

This research was supported by statutory funds (BK-232/RAu-3/2017) of the Institute of Electronics, Silesian University of Technology granted by the Polish Ministry of Science and Higher Education no. 8686/E-367/S/2016.

ORCID


Sławomir Lasota http://orcid.org/0000-0001-8179-7653

REFERENCES

1. Lin PM. *Symbolic Network Analysis*. Amsterdam, the Netherlands: Elsevier Science; 1994.
2. Gielen G, Wambacq P, Sansen WM. Symbolic analysis methods and applications for analog circuits: a tutorial overview. *Proc IEEE*. 1994;82(2):287-304. https://doi.org/10.1109/5.265355
3. Wambacq P, Dobrovolny P, Gielen GGE, Sansen W. Symbolic analysis of large analog circuits using a sensitivity-driven enumeration of common spanning trees. *IEEE Trans Circuits Syst II Analog Digit Signal Process*. 1998;45(10):1342-1350. https://doi.org/10.1109/82.728847
4. Guerra O, Roca E, Fernández FV, Rodríguez-Vázquez A. Approximate symbolic analysis of hierarchically decomposed analog circuits. *Analog Integr Circuits Signal Process*. 2002;31(2):131-145. https://doi.org/10.1023/A:1015094011107
5. Mayeda W. *Graph Theory*. New York: Wiley-Interscience; 1972.
6. Shi C-JR, Tan X-D. Canonical symbolic analysis of large analog circuits with determinant decision diagrams. *IEEE Trans Comput Des Integr Circuits Syst*. 2000;19(1):1-18. https://doi.org/10.1109/43.822616
7. Tan X-D, Shi C-JR. Hierarchical symbolic analysis of analog integrated circuits via determinant decision diagrams. *IEEE Trans Comput Des Integr Circuits Syst*. 2000;19(4):401-412. https://doi.org/10.1109/43.838990
8. Shi C-JR, Tan X-D. Compact representation and efficient generation of s-expanded symbolic network functions for computer-aided analog circuit design. *IEEE Trans Comput Des Integr Circuits Syst*. 2001;20(7):813-827. https://doi.org/10.1109/43.930996
9. Ho C-W, Ruehli A, Brennan P. The modified nodal approach to network analysis. *IEEE Trans Circuits Syst*. 1975;22(6):504-509. https://doi.org/10.1109/TCS.1975.1084079
10. Shi G, Chen W, Shi C-JR. A graph reduction approach to symbolic circuit analysis. *2007 Asia South Pacific Des. Autom. Conf.* 2007:197-202. https://doi.org/10.1109/ASPDAC.2007.357985
11. Xu H., Shi G., and Li X. Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps. *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC*, 2011; 19–24. https://doi.org/10.1109/ASPDAC.2011.5722183.
12. Li X, Xu H, Shi G, Tai A. Hierarchical symbolic sensitivity computation with applications to large amplifier circuit design. *Proc - IEEE Int Symp Circuits Syst*. 2011;22:2733-2736. https://doi.org/10.1109/ISCAS.2011.5938170
13. Song Y, Shi G. Hierarchical graph reduction approach to symbolic circuit analysis with data sharing and cancellation-free properties. *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC* 2012; 541–546. https://doi.org/10.1109/ASPDAC.2012.6165012.
14. Shi G. A survey on binary decision diagram approaches to symbolic analysis of analog integrated circuits. *Analog Integr Circuits Signal Process*. 2013;74(2):331-343. https://doi.org/10.1007/s10470-011-9773-8
15. Shi G. Graph-pair decision diagram construction for topological symbolic circuit analysis. *IEEE Trans Comput Des Integr Circuits Syst*. 2013;32(2):275-288. https://doi.org/10.1109/TCAD.2012.2217963
16. Yin Z. Symbolic network analysis with the valid trees and the valid tree-pairs. *Circuits Syst 2001 ISCAS 2001 2001 IEEE Int Symp*. 2001;5:335-338. https://doi.org/10.1109/ISCAS.2001.922053
17. Tlelo-Cuautle E, Sánchez-López C, Martínez-Romero E, Tan SX-D. Symbolic analysis of analog circuits containing voltage mirrors and current mirrors. *Analog Integr Circuits Signal Process*. 2010;65(1):89-95. https://doi.org/10.1007/s10470-010-9455-y
18. Sánchez-López C, Fernández FV, Tlelo-Cuautle E. Generalized admittance matrix models of OTRAs and COAs. *Microelectron J*. 2010;41(8):502-505. https://doi.org/10.1016/j.mejo.2010.06.010
19. Tlelo-Cuautle E, Sánchez-López C, Moro-Frías D. Letter to the editor: symbolic analysis of (MO)(I)CCI(II)(III)-based analog circuits. *Int J Circuit Theory Appl*. 2010;38(6):649-659. https://doi.org/10.1002/cta.582
20. Sánchez-López C, Fernández FV, Tlelo-Cuautle E, Tan SXD. Pathological element-based active device models and their application to symbolic analysis. *IEEE Trans Circuits Syst I Regul Pap*. 2011;58(6):1382-1395. https://doi.org/10.1109/TCSI.2010.2097696
21. Sánchez-López C. Pathological equivalents of fully-differential active devices for symbolic nodal analysis. *IEEE Trans Circuits Syst I Regul Pap*. 2013;60(3):603-615. https://doi.org/10.1109/TCSI.2013.2244271
22. Shi G. Two-graph analysis of pathological equivalent networks. *Int J Circuit Theory Appl*. 2015;43(9):1127-1146. https://doi.org/10.1002/cta.1997

23. Lasota S, Malcher A. A cancellation-free algorithm for the symbolic analysis of circuits containing current conveyors. *Int. Conf. Signals Electron. Syst. ICSES '04*., 2004; 171–174.
24. Lasota S. A direct and fully cancellation-free approach to the analysis of large analog circuits with decision diagrams. *Int. Conf. Signals Electron. Syst. ICSES '06*., 2006; 149–157.
25. Lasota S. Parameter decision diagrams in the symbolic analysis and the structural synthesis. Part I, II & III. *X-th Int. Work. Symb. Numer. Methods, Model. Appl. to Circuit Des. SM2ACD '08* 2008; 149–157, 172–179, 180–187.
26. Lasota S. Fast multilevel & always cancelation-free method for large electric networks exact symbolic analysis. *IFAC Proc Vol*. 2012;45(7):214-219. https://doi.org/10.3182/20120523-3-CZ-3015.00042
27. Lasota S. A new always cancellation-free approach to the multilevel symbolic analysis for very large electric networks. Int Conf Int Conf Signals Electron Syst (ICSES), 2012, 2012; 1–6. https://doi.org/10.1109/ICSES.2012.6382244
28. Lasota S. Multilevel hierarchical always cancellation-free symbolic analysis method for large electric networks. *Electron - Constr Technol Appl Ther*. 2013;54(2):51-57.
29. Lasota S. Models of modern active devices for effective and always cancelation-free symbolic analysis. *IFAC-PapersOnLine*. 2016;49(25):80-85. https://doi.org/10.1016/j.ifacol.2016.12.014
30. Gantmacher F.R. The theory of matrices, 1977. https://doi.org/10.1007/978-3-642-99234-6.
31. Chen W. *Applied Graph Theory*. Amsterdam: North-Holland Pub. Co.; 1971.
32. Tan SX-D, Guo W, Qi Z. Hierarchical approach to exact symbolic analysis of large analog circuits. *IEEE Trans Comput Des Integr Circuits Syst*. 2005;24(8):1241-1250. https://doi.org/10.1109/TCAD.2005.850812
33. Tan SX-D. A general hierarchical circuit modeling and simulation algorithm. *IEEE Trans Comput Des Integr Circuits Syst*. 2005;24(3):418-434. https://doi.org/10.1109/TCAD.2004.842815
34. Sigorskij VP, Petrenko AI. *Algorithms of the Analysis of Electronic Circuits*. Kiev: Tehnika; 1971.
35. Filaretov V, Gorshkov K. Topological analysis of active networks containing pathological mirror elements. *IEEE Int Sci Conf Electron Nanotechnol*. 2013;460-464. https://doi.org/10.1109/ELNANO.2013.6552078
36. Filaretov V, Gorshkov K, Kurganov S. A cancellation-free symbolic sensitivity technique based on network determinant expansion. *Adv Electr Eng*. 2015;2015(January 2015):1-13. https://doi.org/10.1155/2015/328517
37. Pugh W. Skip lists: a probabilistic alternative to balanced trees. *Commun ACM*. 1990;33(6):668-676. https://doi.org/10.1145/78973.78977
38. Sánchez-López C, Tlelo-Cuautle E. Symbolic noise analysis in analog integrated circuits. *Proc - IEEE Int Symp Circuits Syst*. 2004;5:245-248. https://doi.org/10.1109/ISCAS.2004.1329508
39. Martínez-Romero E, Tlelo-Cuautle E, Sánchez-López C, Tan SXD. Symbolic noise analysis of low voltage amplifiers by using nullors. *Symb Numer Methods, Model Appl to Circuit Des (SM2ACD), 2010 XIth Int Work*. 2010;1:1-5. https://doi.org/10.1109/SM2ACD.2010.5672322
40. Fakhfakh M, Tlelo-Cuautle E, Fernández FV. *Design of Analog Circuits Through Symbolic Analysis*. Oak Park: Bentham Science Publishers; 2012.
41. Wambacq P., and Sansen W. *Distortion Analysis of Analog Integrated Circuits*, 1998. https://doi.org/10.1007/978-1-4757-5003-4.
42. Alderson GE, Lin PM. Integrating topological and numerical methods for semi-symbolic network analysis. *Proc. 8th Allert. Conf. Circuits Syst*., 1970; 646–654.
43. Shi G., Tan S.X.-D., and Tlelo Cuautle E. *Advanced Symbolic Analysis for VLSI Systems-Methods and Applications*, 2014. https://doi.org/10.1007/978-1-4939-1103-5_1.
44. Soliman AM, Saad RA. The voltage mirror-current mirror pair as a universal element. *Int J Circuit Theory Appl*. 2010;38(8):787-795. https://doi.org/10.1002/cta.596
45. Wang H-Y, Chiang N-H, Nguyen Q-M, Chang S-H. Circuit synthesis using pathological elements. *Adv Mater Physics, Mech Appl*. 2014;317-328. https://doi.org/10.1007/978-3-319-03749-3_26
46. Bryant RE. Graph-based algorithms for Boolean function manipulation. *IEEE Trans Comput*. 1986;C-35(8):677-691. https://doi.org/10.1109/TC.1986.1676819
47. Minato S. Zero-suppressed BDDs for set manipulation in combinatorial problems. *Proc. 30th Int. Des. Autom. Conf*., 1993; 272–277.
48. Minty G. A simple algorithm for listing all the trees of a graph. *IEEE Trans Circuit Theory*. 1965;12(1):120. https://doi.org/10.1109/TCT.1965.1082385
49. Starzyk J, Konczykowska A. Flowgraph analysis of large electronic networks. *IEEE Trans Circuits Syst*. 1986;33(3):302-315. https://doi.org/10.1109/TCS.1986.1085914
50. Hassoun MM, McCarville KS. Symbolic analysis of large-scale networks using a hierarchical signal flowgraph approach. *Analog Integr Circuits Signal Process*. 1993;3(1):31-42. https://doi.org/10.1007/BF01239191

51. Hassoun MM, Lin P-M. A hierarchical network approach to symbolic analysis of large-scale networks. *IEEE Trans Circuits Syst I Fundam Theory Appl*. 1995;42(4):201-211. https://doi.org/10.1109/81.382473

52. Pierzchala M, Rodanski B. Generation of sequential symbolic network functions for large-scale networks by circuit reduction to two-port. *IEEE Trans Circuits Syst I Fundam Theory Appl*. 2001;48(7):906-909. https://doi.org/10.1109/81.933334

53. Doboli A, Vemuri R. A regularity-based hierarchical symbolic analysis method for large-scale analog networks. *IEEE Trans Circuits Syst II Analog Digit Signal Process*. 2001;48(11):1054-1068. https://doi.org/10.1109/82.982361

54. Fontana G. Revisited generalized substitution theorem and its consequences for circuit analysis. *Int J Circuit Theory Appl*. 2017;45(9):1249-1298. https://doi.org/10.1002/cta.2285

55. Hu H, Shi G, Tai A, Lee F. Topological symbolic simplification for analog design. *Proc - IEEE Int Symp Circuits Syst*. 2015;2644-2647. https://doi.org/10.1109/ISCAS.2015.7169229

56. Shi G, Hu H, Deng S. Topological approach to automatic symbolic macromodel generation for analog integrated circuits. *ACM Trans Des Autom Electron Syst*. 2017;22(3):1-25. https://doi.org/10.1145/3015782

57. Rodanski B. Symbolic analysis. http://rodanski.net/ben/work/symbolic/, accessed Jan. 2018.

58. Pierzchala M, Rodanski B. Road map representation of s-expanded symbolic network functions. *2007 18th Eur. Conf. Circuit Theory Des*., 2007; 858–861. https://doi.org/10.1109/ECCTD.2007.4529732.

59. Fontana G. A novel method of network function analysis based on the Andreani-Mattisson extension to the Cochrun-Grabel algorithm. *Int J Circuit Theory Appl*. 2015;43(5):579-612. https://doi.org/10.1002/cta.1961

How to cite this article: Lasota S. Symbolic analysis of electric networks with higher order summative cofactors and parameter decision diagrams. *Int J Circ Theor Appl*. 2018;46:1796–1826. https://doi.org/10.1002/cta.2495