Short Papers

Hierarchical Approach to Exact Symbolic Analysis of Large Analog Circuits

Sheldon X.-D. Tan, Weikun Guo, and Zhenyu Qi

***Abstract*—This paper proposes a novel approach to the exact symbolic analysis of very large analog circuits. The new method is based on determinant decision diagrams (DDDs) representing symbolic product terms. But instead of constructing DDD graphs directly from a flat circuit matrix, the new method constructs DDD graphs in a hierarchical way based on hierarchically defined circuit structures. The resulting algorithm can analyze much larger analog circuits exactly than before. The authors show that exact symbolic expressions of a circuit are cancellation-free expressions when the circuit is analyzed hierarchically. With this, the authors propose a novel symbolic decancellation process, which essentially leads to the hierarchical DDD graph constructions. The new algorithm partially avoids the exponential DDD construction time by employing more efficient DDD graph operations during the hierarchical construction. The experimental results show that very large analog circuits, which cannot be analyzed exactly before like $\mu A725$ and other unstructured circuits up to 100 nodes, can be analyzed by the new approach for the first time. The new approach significantly improves the exact symbolic capacity and promises huge potentials for the applications of exact symbolic analysis.**

***Index Terms*—Behavioral modeling, circuit simulation, symbolic analysis.**

I. Introduction

Symbolic analysis calculates the behavior or the characteristics of a circuit in terms of symbolic parameters. It can be used to derive parametric behavioral models of analog modules. As illustrated in [4], simple yet accurate symbolic expressions can also be interpretable for analog designers to gain insight into circuit behavior, performance, and stability, and are important for verification and synthesis applications in analog circuit design automation [5]. Symbolic analysis, however, suffers from the well-known circuit-size limitation problem as the number of product terms increases exponentially with the complexity of a circuit. Traditional symbolic analyzers can only analyze analog circuits with about ten nodes [4]. With the introduction of the determinant decision diagram (DDD) concept, symbolic approaches based on DDD graphs have improved the exact symbolic analysis capacity, but the size of typical unstructured analog modules that can be analyzed exactly is still less than 25–30 nodes in general [12].

In addition to the DDD graph-based approaches, modern symbolic analyzers also rely on two techniques, symbolic simplification and hierarchical decomposition [3], to cope with larger analog blocks. Symbolic simplification discards those insignificant terms based on the relative numerical magnitude of symbolic parameters and the frequency defined at some nominal design points or over some ranges. It can be performed before/during the generation of symbolic terms [1], [8], [11], [23] or after the generation [2], [4], [21]. The simplified expressions, however, only have sufficient accuracy at some points or over some frequency ranges. Even worse, simplification often loses certain information, such as sensitivity with respect to parasitics, which is crucial for circuit optimization and testability analysis. For instance, mismatched device parameters typically have very small or no impact on circuit performance for well-matched devices, but their sensitivities may not be small, which can lead to significant impacts on circuit characteristics in the presence of no trivial device mismatch.

Hierarchical decomposition generates symbolic expressions in the sequence-of-expression form [7], [14]. There are three methods known as topological analysis [14], network formulation [7], and DDD-based approach [19]. The first two methods are based on the symbolic Gaussian node elimination concept to obtain circuit transfer functions in the sequence-of-expression form, and the last one performs subcircuit reduction and generates hierarchical DDD graphs, which can be viewed as a special form of sequence of expressions as some DDD nodes themselves represent symbolic expressions (DDD graphs) coming from subcircuits. The major drawback for all those hierarchical approaches is that symbolic manipulations other than evaluation of the resulting sequence of expressions, which are critical for behavioral modeling and approximation, are known to be difficult and complicated, and often require dedicated efforts, e.g., sensitivity calculation in [9] and lazy approximation in [11].

This paper presents a novel approach to exact symbolic analysis based on DDD graphs. The new approach is based on the observation that exact symbolic expressions are cancellation-free expressions during the hierarchical decomposition process of a circuit [16]. The authors propose to perform the decancellation in a symbolic way, which equivalently leads to the hierarchical construction of flat DDD graphs. The main difference between the new method and the existing DDD-based hierarchical approach [19] is that the resulting DDD graphs are simple flat DDD graphs where each DDD node represents a simple admittance from stamping of devices. In other words, no sequence of expressions is used.

With flat DDD graphs, all symbolic manipulations such as computing s-expanded DDDs [13], finding dominant terms via the shortest path search [18], cofactoring and sensitivity calculation can be performed easily [12]. The authors also develop an efficient clustering algorithm to automatically define the circuit hierarchy to minimize DDD sizes and construction time. Experimental results show that mesh-structured resistance–capacitance (*RC*) filter circuits with up to 100 nodes and tree-like structured analog circuits with more than 1000 nodes can be exactly analyzed for the first time. The new algorithm promises the huge potential for exact symbolic analysis and highly accurate behavioral modeling of many analog blocks whose sizes are typically less than 100 nodes.

The rest of the paper is organized as follows. Section II reviews how DDD graphs are constructed in the existing methods and shows the difference of the new method from the existing method. Section III provides an overview of the hierarchical decomposition process and cancellation issues during the process. Section IV gives some theoretical results for cancellation-free symbolic expressions in hierarchical circuit decomposition. Section V presents the new approach to constructing cancellation-free DDDs during the hierarchical decomposition process. Section VI presents the experimental results and the com-

Manuscript received June 13, 2004; revised October 7, 2004. This work was supported by NSF CAREER Award CCF-0448534, NSF Grant OISE-0451688, and UC Regent's Faculty Fellowship (04-05). This paper was recommended by Associate Editor S. Sapatnekar.

S. X.-D. Tan and Z. Qi are with Department of Electrical Engineering, University of California, Riverside, CA 92521 USA (e-mail: stan@ee.ucr.edu; zhenyu@ee.ucr.edu).

W. Guo was with Department of Electrical Engineering, University of California, Riverside, CA 92521 USA (e-mail: wguo@ee.ucr.edu).

Digital Object Identifier 10.1109/TCAD.2005.850812

parison with the existing DDD-based approach. Section VII concludes the paper.

II. Review of the Existing Laplace Expansion-Based DDD Construction Algorithm

For a flat circuit, once the circuit matrix has been built, the authors can construct its DDDs directly from the circuit matrix using the Laplace expansion rule [12]. The idea is to expand the matrix and cache all the minors for reuse in the later expansion process. So the time complexity for the constructing process is roughly proportional to the number of unique minors and the time to construct each minor plus the time for storing and retrieving them. The number of unique minors is roughly related to the number of unique product terms. As a result, the authors may end up with an exponential construction time even though the final DDD size grows very slowly (polynomially) with the size of the circuits. So Laplace expansion-based symbolic expression generation can be viewed as an explicit term generation algorithm where product terms are explicitly generated and then put into DDD graphs for compact representation.

In the proposed hierarchical construction algorithm, which will be discussed in detail below, the authors only need to construct DDDs for the size-reduced subcircuits or the top-level circuit by Laplace expansion. The final DDDs are constructed via DDD graph operations with the DDDs from each subcircuit. The key difference is that DDD operations are proportional to the size of the DDD graph and not to the number of paths (product terms) in the DDD graph. As a result, the new method partially avoids the explicit generation of symbolic product terms, which may grow exponentially with the size of circuits.

III. Hierarchical Circuit Analysis and Symbolic Cancellation

This section briefly reviews how nodes in a subcircuit can be suppressed in matrix form and symbolic cancellation in the hierarchical circuit decomposition.

Consider a circuit with some internal structures and terminals. The circuit unknowns—node voltage variables and branch current variables—can be partitioned into three disjoint groups x^I, x^B, and x^R, where the superscripts I, B, R stand for, respectively, internal variables, boundary variables, and the rest of the variables. With this, the system equation set $\mathbf{Ax} = \mathbf{b}$ can be rewritten in the form (Schur decomposition)

$$\begin{bmatrix} A^{II} & A^{IB} & 0 \\ A^{BI} & A^{BB} & A^{BR} \\ 0 & A^{RB} & A^{RR} \end{bmatrix} \begin{bmatrix} x^I \\ x^B \\ x^R \end{bmatrix} = \begin{bmatrix} b^I \\ b^B \\ b^R \end{bmatrix}. \tag{1}$$

The matrix A^{II} is the internal matrix associated with the internal variable vector x^I. Assume that row indices for A^{II} are from 1 to m, row indices for A^{BB} are from $m+1$ to $m+l$, and row indices for A^{RR} are from $m+l+1$ to n, where m and l are the sizes of submatrices A^{II} and A^{BB}, respectively.

Subcircuit suppression (Schur decomposition) eliminates all the variables in x^I and transforms (1) into the reduced set of equations

$$\begin{bmatrix} A^{BB} - A^{BI}(A^{II})^{-1}A^{IB} & A^{BR} \\ A^{RB} & A^{RR} \end{bmatrix} \begin{bmatrix} x^B \\ x^R \end{bmatrix} = \begin{bmatrix} b^B - A^{BI}(A^{II})^{-1}b^I \\ b^R \end{bmatrix}. \tag{2}$$

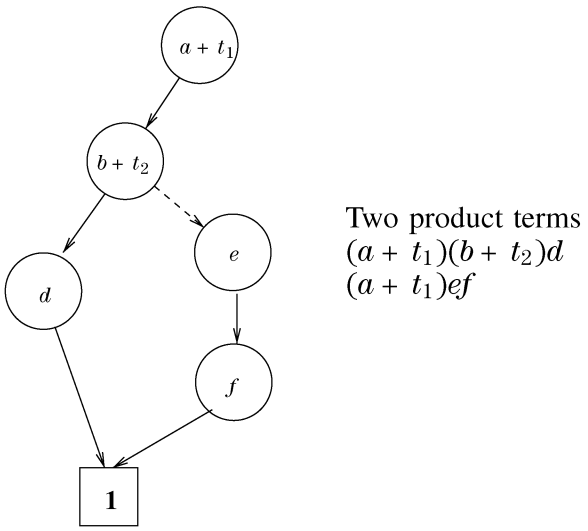

Fig. 1. DDD graph for a 3 × 3 determinant.

Since the number of internal variables is m and the number of boundary variables is l, each matrix element in the modified A^{BB} and b^B can then be written in the expanded form

$$a_{u,v}^{BB*} = a_{u,v}^{BB} - \frac{1}{\det(A^{II})} \sum_{k_1,k_2=1}^{m} a_{u,k_1}^{BI} \Delta_{k_2,k_1}^{II} a_{k_2,v}^{IB} \tag{3}$$

where $u, v = m+1, \ldots, m+l$ and $a_{u,k_1}^{BI} \Delta_{k_2,k_1}^{II} a_{k_2,v}^{IB} / \det(A^{II})$ is called the composite admittance due to modified nodal analysis (MNA) formulation in the sequel, and

$$b_u^{B*} = b_u^B - \frac{1}{\det(\mathbf{A}^{II})} \sum_{k_1,k_2=1}^{m} a_{u,k_1}^{BI} \Delta_{k_2,k_1}^{II} b_{k_2}^I \tag{4}$$

where $u = m+1, \ldots, m+l$ and $\Delta_{u,v}$ is the first-order cofactor of $\det(\mathbf{A})$ with respect to $a_{u,v}$. Notice that the modified submatrix A^{BB} is a full $l \times l$ matrix.

A nice thing about Schur decomposition is that a subcircuit can be suppressed independent of the other subcircuits at a time. The interaction between two subcircuits is automatically taken care of by the composite admittances generated at common boundary nodes (common x^B) between the two subcircuits (if the two subcircuits have common nodes) during the decomposition process. The difference is that (3) and (4) may consist of composite product terms from more than one subcircuit if the corresponding x^B is common to all the involving subcircuits.

Specifically, suppose the authors have two subcircuits, named subckt 1 and subckt 2. The two subcircuits have common boundary nodes $x_{B,12}$, then the corresponding $a_{u,v}^{BB*}$ and b_u^{B*} for the common boundary nodes $x_{B,12}$ can be written as

$$a_{u,v}^{BB*} = a_{u,v}^{BB} - \frac{1}{\det(A^{II,1})} \sum_{k_1,k_2=1}^{m} a_{u,k_1}^{BI,1} \Delta_{k_2,k_1}^{II,1} a_{k_2,v}^{IB,1} - \frac{1}{\det(A^{II,2})} \sum_{k_1,k_2=1}^{m} a_{u,k_1}^{BI,2} \Delta_{k_2,k_1}^{II,2} a_{k_2,v}^{IB,2} \tag{5}$$

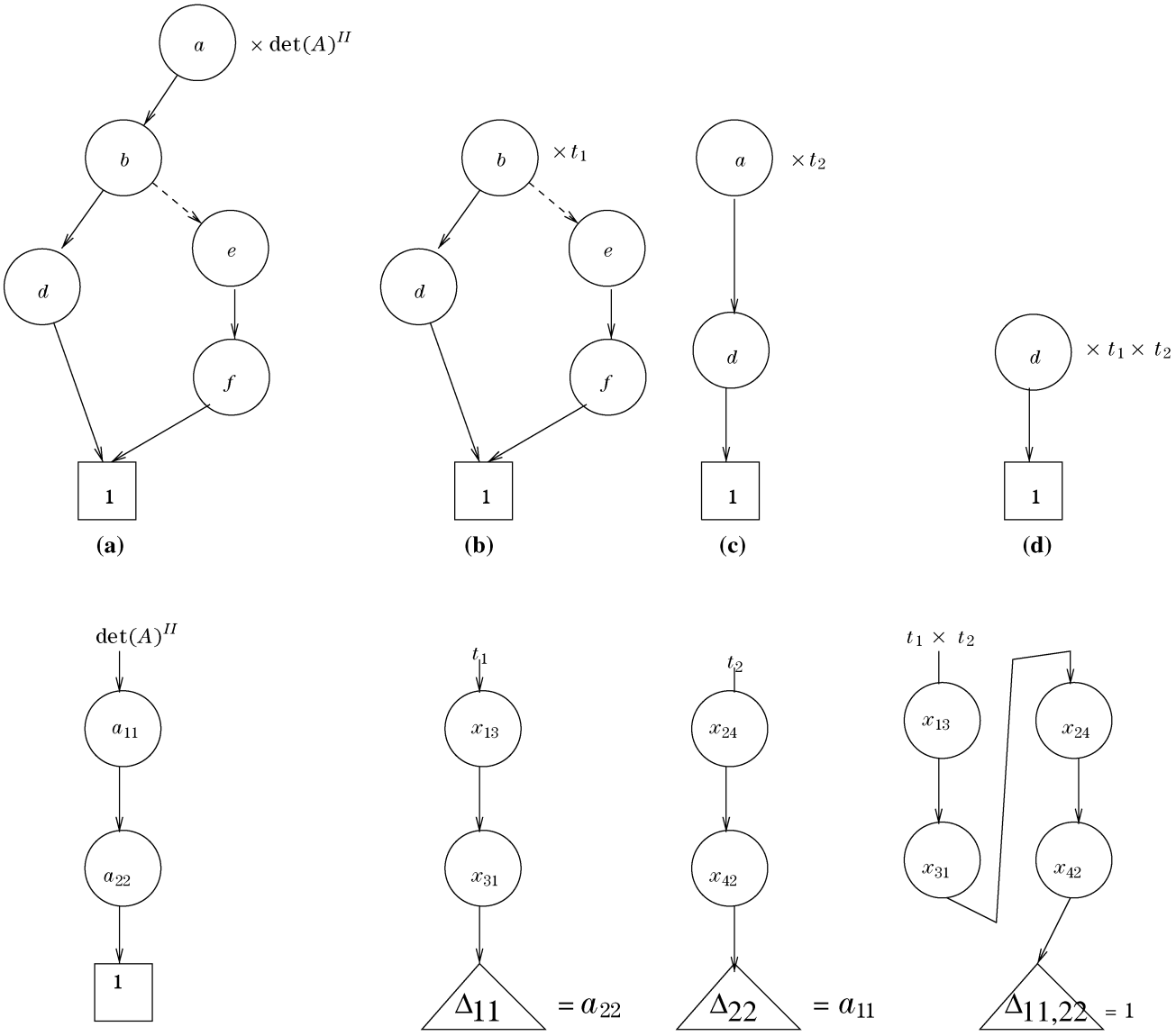

Fig. 2. Illustration of PDDD graphs and compDDD graphs.

where $\det(A^{II,1})$ and $\det(A^{II,2})$ are the internal matrices for subckt 1 and subckt 2, respectively. The same notations go for $a_{x,x}^{BI,t}$, $a_{x,x}^{IB,t}$, and $\Delta_{x,x}^{II,t}$, $t = 1, 2$

$$b_u^{B*} = b_u^B - \frac{1}{\det(\mathbf{A}^{II,1})} \sum_{k_1,k_2=1}^{m} a_{u,k_1}^{BI,1} \Delta_{k_2,k_1}^{II,1} b_{k_2}^{I,1} - \frac{1}{\det(\mathbf{A}^{II,2})} \sum_{k_1,k_2=1}^{m} a_{u,k_1}^{BI,2} \Delta_{k_2,k_1}^{II,2} b_{k_2}^{I,2}. \quad (6)$$

Let us go back to (3) and (4). The composite admittances $-a_{u,k_1}^{BI} \Delta_{k_2,k_1}^{II} a_{k_2,v}^{IB} / \det(A^{II})$ in the modified A^{BB} submatrix will generate symbolic cancellations if they are multiplied with other composite admittances when the reduced matrix is further reduced and final transfer functions are computed [16], [17]. Specifically, two kinds of cancellations will happen: term cancellation, where two product terms consisting of composite admittances cancel out (their sum equals zero), and common-factor cancellation, where the numerator and the denominator in the resulting product terms consisting of the composite admittances have a common factor. It was shown in [17] that common-factor reduction happens when all row and column indices of first-order cofactors in a product term are unique. Otherwise, it will lead to term cancellation and the product will always cancel out with another term. The authors call this term cancellation rule in the sequel. For common-factor cancellation $\det(A^{II})^m$, $m \geq 1$ will become a common factor if the number of first-order cofactors in a product term is $m + 1$ [16] due to the identity

$$\begin{vmatrix} \Delta_{i_1 j_1} & \cdots & \Delta_{i_1 j_k} & \cdots & \Delta_{i_1 j_m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \Delta_{i_k j_1} & \cdots & \Delta_{i_k j_k} & \cdots & \Delta_{i_k j_m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \Delta_{i_m j_1} & \cdots & \Delta_{i_m j_k} & \cdots & \Delta_{i_m j_m} \end{vmatrix} = \det(A)^{m-1} \Delta_{i_1 j_1, \ldots, i_m, j_m} \quad (7)$$

where A is an $n \times n$ matrix with its $m \times m$ elements at rows $i_1, \ldots, i_m$ and columns $j_1, \ldots, j_m$, and $\Delta_{i_1 j_1, \ldots, i_m, j_m}$ is the mth order cofactor with rows $i_1, \ldots, i_m$ and columns $j_1, \ldots, j_m$ removed from $\det(A)$. A general s-domain reduction algorithm was proposed in [16], where common-factor decancellation is carried out numerically in the s-domain. In this paper, the authors show that both term and common-

factor cancellations can be removed symbolically. Such a decancellation process essentially leads to the hierarchical construction of DDD graphs.

IV. Cancellation-Free Expressions Versus Exact Expressions

This section shows that cancellation-free expressions in the hierarchical circuit decomposition will lead to the exact symbolic expressions obtained from the original flatten circuit, which lays the foundation for the hierarchical DDD construction algorithm.

Before theoretical results are presented, the authors define some terms used in this paper.

Definition 1: Exact symbolic expressions are symbolic expressions obtained directly from the expansion of a determinant without any approximation.

Definition 2: Cancellation-free expressions are symbolic expressions that are free of symbolic cancellations. Symbolic cancellations can be termed cancellation or common-factor cancellation and come from the hierarchical Schur decomposition only.

Let A^{II} be the submatrix to be reduced. The authors then give the following theorem concerning cancellation-free symbolic rational expressions.

Theorem 1: For the reduced matrix shown in (2), the numerator of a cancellation-free symbolic rational expression of the determinant of the reduced matrix is the exact symbolic expression obtained from the determinant of the original flat matrix.

A detailed proof of this theorem can be found in [15]. This theory basically states that if we manage to remove the cancellations during the symbolic Schur decomposition, exact symbolic expressions from the original circuit matrix can be obtained. This leads to the hierarchical DDD construction algorithm shown in the next section.

V. Hierarchical DDD Graph Construction Algorithm

This section presents the new hierarchical DDD construction algorithm. The basic idea is to construct cancellation-free DDDs for one subcircuit at a time from the hierarchically defined circuit structure. The authors call such a process as expansion of a subcircuit. The construction proceeds until all subcircuits are expended.

A. Basic Idea

For any product term in the reduced matrix in (3) and (4), if the number of first-order cofactors is larger than one, both term and common-factor cancellations may happen. According to the term cancellation rule [17], if any two first-order cofactors share the same row or column indices in the subcircuit matrix A^{II} (in other words, their row or column indices are the same), this term will cancel out with another term. Otherwise, this product will generate a common factor (when added with other similar product terms with the same number of first-order cofactors) between the numerator and the denominator.[1]

This suggests an efficient symbolic decancellation strategy based on the DDD graph [12], which is a compact graph representation of determinants and cofactors. The authors would like to illustrate the symbolic decancellation process using an example. Consider a 5×5 matrix

$$\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{bmatrix} a_{11} & 0 & x_{13} & 0 & 0 \\ 0 & a_{22} & 0 & x_{24} & 0 \\ x_{31} & 0 & a & 0 & 0 \\ 0 & x_{42} & 0 & b & e \\ 0 & 0 & 0 & f & d \end{bmatrix}. \tag{8}$$

[1]The product terms from the reduced matrix become rational function of s due to the presence of composite admittances.

```
HierConst (P, subckt)
For all the subckts, visit each subckt in a BFS order {
  D_res = P;
  for subckt i, do {
    build its internal determinant det(A_i^II);
    D_res = ConstPDDDR(D_res, subckt);
    Multiply each PDDD with its compDDDs;
    Add all the resulting DDD graphs into one DDD graph;
  }
}
```

```
ConstPDDDR(P, subckt)
if(P = one or has been visited)
  return;
if(P.index < parent_min(P))
  return;
ConstHDDDListR(P.child1, subckt);
ConstHDDDListR(P.child0, subckt);
L_1 = MultiplyHDDDList(P, HDDDList(P.child1), subckt);
L_res = UnionHDDDList(L_1, HDDDList(P.child0));
HDDDList(P) = L_res;
```

```
MultiplyHDDDList(P, HL_1, subckt)
if (P has device admittances or composite elements
which is not created from the given subckt)
  L_res = Multiplication of P with each PDDD in the HL_1;
if (P have composite admittance which is created from
the given sucbkt)
  for each composite admittance (CA){
    for each PDDD T_i in HL_1, do {
      r = row of the first-order cofactor in the CA;
      c = column of the first-order cofactor in the CA;
      if (r ∉ T_i's CRowSet and c ∉ T's CColSet){
        Add r and c into T_i's CRowSet and CColSet respectively;
        Add the row and column indices of P into T_i's PRowSet
        and PColSet respectively;
      } else {
        Remove the T_i from HL_1;
      }
    }
  }
```

Fig. 3. Hierarchical DDD construction algorithm.

After reducing variables v_1 and v_2, the authors have a reduced 3×3 matrix

$$\begin{array}{c} v_3 \\ v_4 \\ v_5 \end{array} \begin{bmatrix} a+t_1 & 0 & \\ 0 & b+t_2 & e \\ 0 & f & d \end{bmatrix} \tag{9}$$

where $t_1 = -x_{13}\Delta_{11}x_{31}/\det(A^{II})$, $t_2 = -x_{24}\Delta_{22}x_{42}/\det(A^{II})$, $\det(A^{II}) = a_{11}a_{22}$, $\Delta_{11} = a_{22}$, and $\Delta_{22} = a_{11}$. The corresponding DDD representation of the determinant of the reduced matrix is shown in Fig. 1.

Before the decancellation process is discussed, two special DDDs are defined first.

Definition 3: A partial DDD or PDDD graph is a DDD graph that consists of product terms, which are cancellation-free with respect to one or a number of expanded subcircuits, and the symbolic elements in each of the product term in a PDDD come from device admittances of the current top-level circuit, which include all the expanded subcircuits and the original top-level circuit, and the composite admittances from unexpanded subcircuits.

Note that a PDDD is only partially free of cancellations as the cancellations are only removed from the expanded subcircuit. The resulting PDDD may not be cancellation-free with respect to other unexpanded downstream subcircuits.

Definition 4: A complementary DDD or compDDD is a DDD graph that consists of product terms that are cancellation-free with respect to

TABLE I
CONSTRUCTION STATISTICS ON DIFFERENT ANALOG CIRCUITS

Circuit	#nodes	$\|DDD\|$	#path	#IntNode	CPU Time(hier)	CPU Time(flat)
CMOS_Opamp	14	8803/1891	79 643	5	2.22	0.44
$\mu A741$	24	2314/4985	108 032	10	0.26	0.61
$\mu A725$	32	11 373	1.21×10^8	5	4.22	–
low_pass_fliter	34	68 362	5.10×10^7	-	8.43	–
stat_var_filter(1)	15	682	1070	-	0.03	0.02
stat_var_filter(2)	29	63 623	1.24×10^6	-	5.31	–
stat_var_filter(3)	43	2 325 927	1.45×10^9	-	4710.25	–
p5x20x2	100	20 402	5.53×10^{20}	10	2.91	–
p5x20x3	100	112 913	7.3×10^{20}	10	40.6	–
p5x20x4	100	392 326	2.10×10^{21}	10	202.63	–
p10x100	1000	67 840	N/A	50	13.72	–
p10x150	1500	276 340	N/A	50	73.66	–

one or a number of expanded subcircuits. The symbolic elements in each product term in a compDDD come from the subcircuit to be expanded and the composite admittances from unexpanded downstream subcircuits.

The idea of decancellation is to build a number of PDDD graphs from a hierarchical DDD graph where each of the PDDD is cancellation-free with respect to all the expanded subcircuits. The new algorithm removes cancellations caused by one subcircuit at a time and repeats the process for all the subcircuits until the resulting DDDs are cancellation-free for all the subcircuits.

To remove the cancellation caused by the reduction of the given subcircuit, the authors need to remove all the DDD nodes that have composite admittances from the given subcircuit. Since the depth of some DDD graphs will be reduced (or the product terms in the PDDD are not complete), the authors have to find the missing DDD nodes in the reduced subcircuits and associated boundary nodes. All missing DDD nodes from one particular PDDD graph will become a DDD graph again, which is its compDDD graph with respect to the PDDD graph. The decancellation process is better illustrated in Fig. 2

In this figure, the authors construct four cancellation-free PDDD graphs out of the original DDD graph [subfigure (a)–(d)]. Each of the PDDD graphs has a unique compDDD graph, which is also displayed below each PDDD graph. If the authors multiply each PDDD graph with its compDDD graph and add (union operation) all the resulting DDD graphs together, the resulting DDD graph, which has six product terms, will be the same DDD graph constructed directly from the flat matrix in (8), which has the same six product terms.

So the critical issue left is how to construct the PDDDs and their compDDDs from a given hierarchical DDD graph. Notice that a compDDD mainly consists of composite admittances whose row and column indices are not covered by the nodes in its corresponding PDDD graph in the reduced matrix. Suppose that there is only one composite admittance $a_{ik}\Delta_{ij}a_{tj}/\det(A^{II})$ in any product term of the determinant of the reduced circuit. For those product terms, their common PDDD consists of product terms without the composite admittance while their common compDDD consists of the product term $a_{i,k}a_{tj}$ times the cofactor Δ_{ij} as shown in Fig. 2(b)–(c).

When there are more than two composite elements in the compDDDs, the authors need to consider cancellations. 1) If all the first-order cofactors have unique row and column indices, then common-factor cancellation will take place. As a result, the compDDD will consist of a higher order cofactor predicted by (7), as shown in the $t_1 \times t_2$ case in Fig. 2(d). The higher-order cofactor is the cofactor obtained from A^{II} when all the rows and columns of the boundary admittances (like x_{ij} in Fig. 2) are removed. 2) If there are two first-order cofactors sharing a common row or column index in A^{II}, term cancellation will happen. That means that the PDDD and its corresponding compDDD should be removed. In summary, the authors only keep compDDDs, whose first-order cofactors in any path have unique row and column indices.

Notice that the compDDD and PDDD construction rules are consistent with the definition of a determinant [6]: the row and column indices of elements in each product term in the determinant of a matrix will cover all the row and column indices of the matrix exactly once. Since the boundary DDD nodes in compDDD will occupy some of the rows and columns in A^{II}, the authors have to find all elements in the uncovered row and columns in A^{II}, which is actually the operation to get the higher order cofactors.

The authors have the following theoretical result regarding the maximum number of PDDDs (compDDDs) constructed from a hierarchical DDD graph.

Theorem 2: If the size of $\mathbf{A}^{II}$ is m and the size of $\mathbf{A}^{BB}$ is l, then the maximum number of PDDD graphs is bounded by $\sum_{i=0}^{p}(C_{l,i})^2(C_{m,i})^2$, where $C_{n,k} = n!/(k!(n-k)!)$ and $p = \min(l, m)$.

Proof: The authors compute the number of compDDDs since the number of compDDDs equals to the number of PDDDs. For each compDDD, its boundary DDD nodes will cover i unique row and column indices in A^{II} and A^{BB}, respectively, where $i \in [0, p]$, $p = \min(l, m)$. The reason to take the minimum of l and m will be clear soon. The authors take any i rows out of m rows from A^{II}, which is an $m \times m$ matrix, so the number of combinations is $C_{m,i}$. Once the i rows from A^{II} are selected, i columns need to be selected from A^{II}, the number of combinations is $C_{m,i}$. So the total number of choices is $C_{m,i}^2$. After this, the authors continue to select i rows and columns from A^{BB}, which is an $l \times l$ matrix, so the number of choices is $C_{l,i}^2$ and the total number of compDDDs is $(C_{l,i})^2(C_{m,i})^2$ for i. The maximum number of rows or columns that can be selected is bounded by l, for $l > m$. If $i > m$, according to the term cancellation rule, some first-order cofactors must share the same row or column indices in A^{II} as the number of first-order cofactors in any path is larger than the number of rows and columns in A^{II}. As a result, all those compDDDs will cancel out and i is limited to $[0, p]$. The total number of compDDDs is $\sum_{i=0}^{p}(C_{l,i})^2(C_{m,i})^2$. ■

Theorem 2 tells that the number of PDDDs can be limited by either the number of boundary nodes or the number of internal nodes of a subcircuit. So for very strongly coupled circuits, the authors can choose smaller subcircuit sizes to control the number of PDDDs. While the side effect is that the authors get more subcircuits for smaller subcircuits.

B. The Hierarchical DDD Construction Algorithm

Both PDDDs and their compDDDs can be constructed by traversing the given hierarchical DDD graph in a bottom-up fashion. The authors expand one subcircuit at a time from the top to the bottom in a breath first search (BFS) order until all subcircuits are expanded.

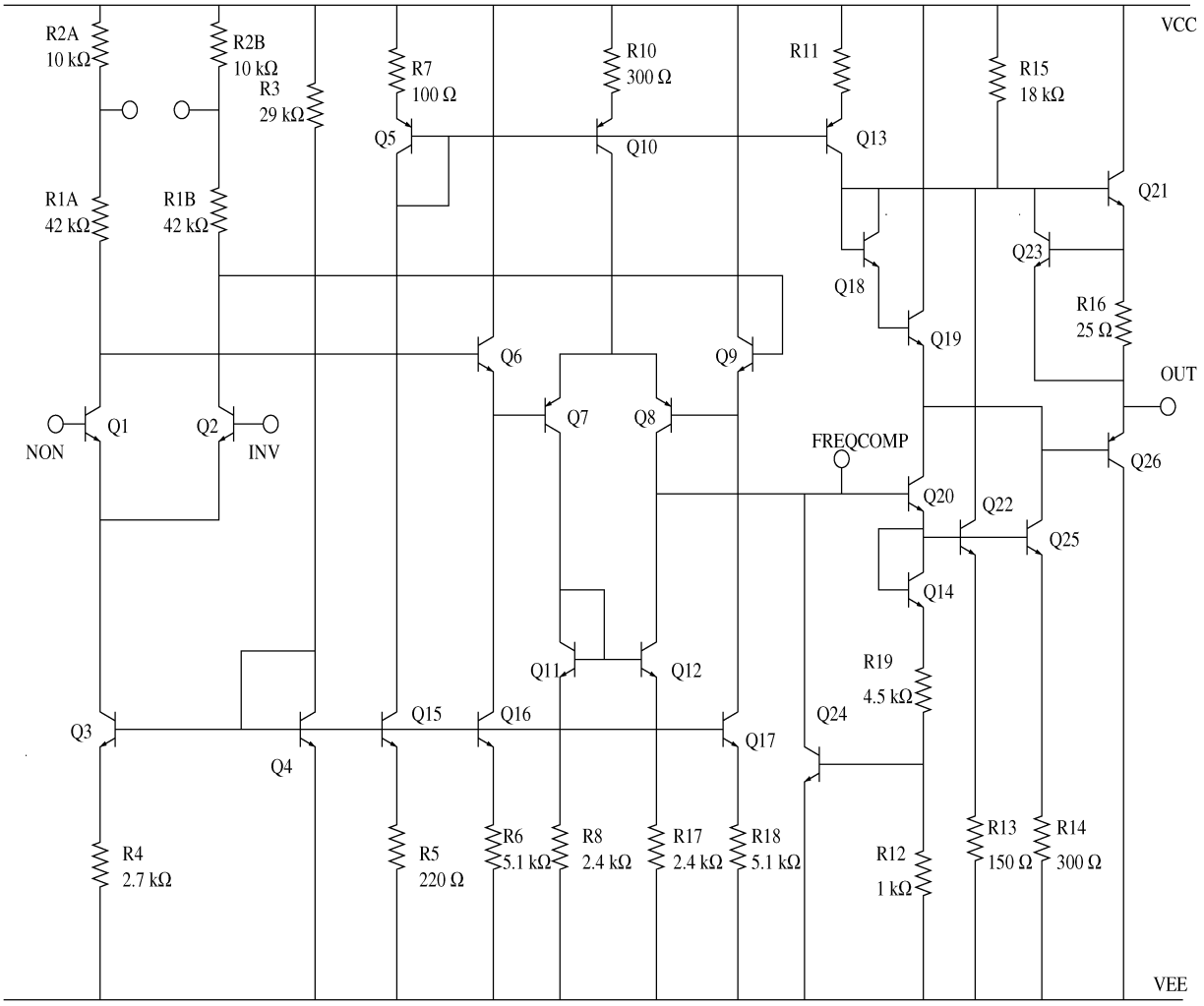

Fig. 4. $\mu A725$ OPAMP circuit.

The authors create a hier-DDD (HDDD) entry (structure) for each DDD node during the construction process. They maintain four index sets: the row index set (PRowSet) and the column index set (PColSet) for the DDD node in the PDDD graph and the row index set (CRowSet) and the column index set (CColSet) for the DDD node in the compDDD graph in $\det(A^{II})$ in each HDDD. The authors also keep track of two DDD graphs in each HDDD structure: PDDD, which is the partial DDD graph for the HDDD, and compDDD, which is the complementary DDD graph for this HDDD. A HDDD entry list will be created at each DDD node when the authors visit each DDD in a bottom-up way. The whole hierarchical construction algorithm is shown in Fig. 3.

UNIONHDDDLIST[L_1, HDDDList(P.child0)] performs the union of two lists of HDDDs. Two HDDDs can be joined if their PRowSet, PColSet, CRowSet, and CColSet are the same. Union operations are done for both the PDDD and the compDDD graphs inside the HDDD.

The complete traversal of the given DDD graph P can actually be avoided if the authors know where the composite admittances from a given subcircuit are located in the resulting $D_{\rm res}$. For a subcircuit, it is noticed that its composite admittances are created in its parent circuits. If the authors know the minimum DDD index in its parent circuits, they can stop further searches when the DDD index just visited is smaller than the minimum DDD index.[2] This reflects the statement "if $(\mathrm{P.index} < \mathrm{parent\_min}(P))$ return;" in the function

[2]In DDD implementation, ordering is enforced such that the parent's index is always larger that its children indices.

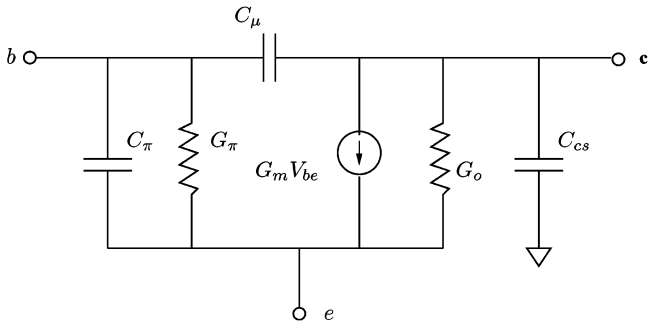

Fig. 5. Bipolar transistor model.

CONSTHDDDR(). In this case, the resulting HDDD list has just one HDDD entry and its PDDD is just the original DDD graph rooted at the current DDD node.

Consequently, the hierarchical construction algorithm will have the best efficiency when sharing of DDD nodes between different subcircuits is minimized. This implies that boundary nodes between different subcircuits should be minimized. In this way, the new DDDs can be created at or near the top portion of the new $D_{\rm res}$ graph for expanding each new subcircuit and the construction (DDD graph traversal) is localized. Unfortunately, only tree-structured circuits have such an extremely localized property. This feature also suggests that if the given circuit structure is like a tree, the construction will be quite efficient.

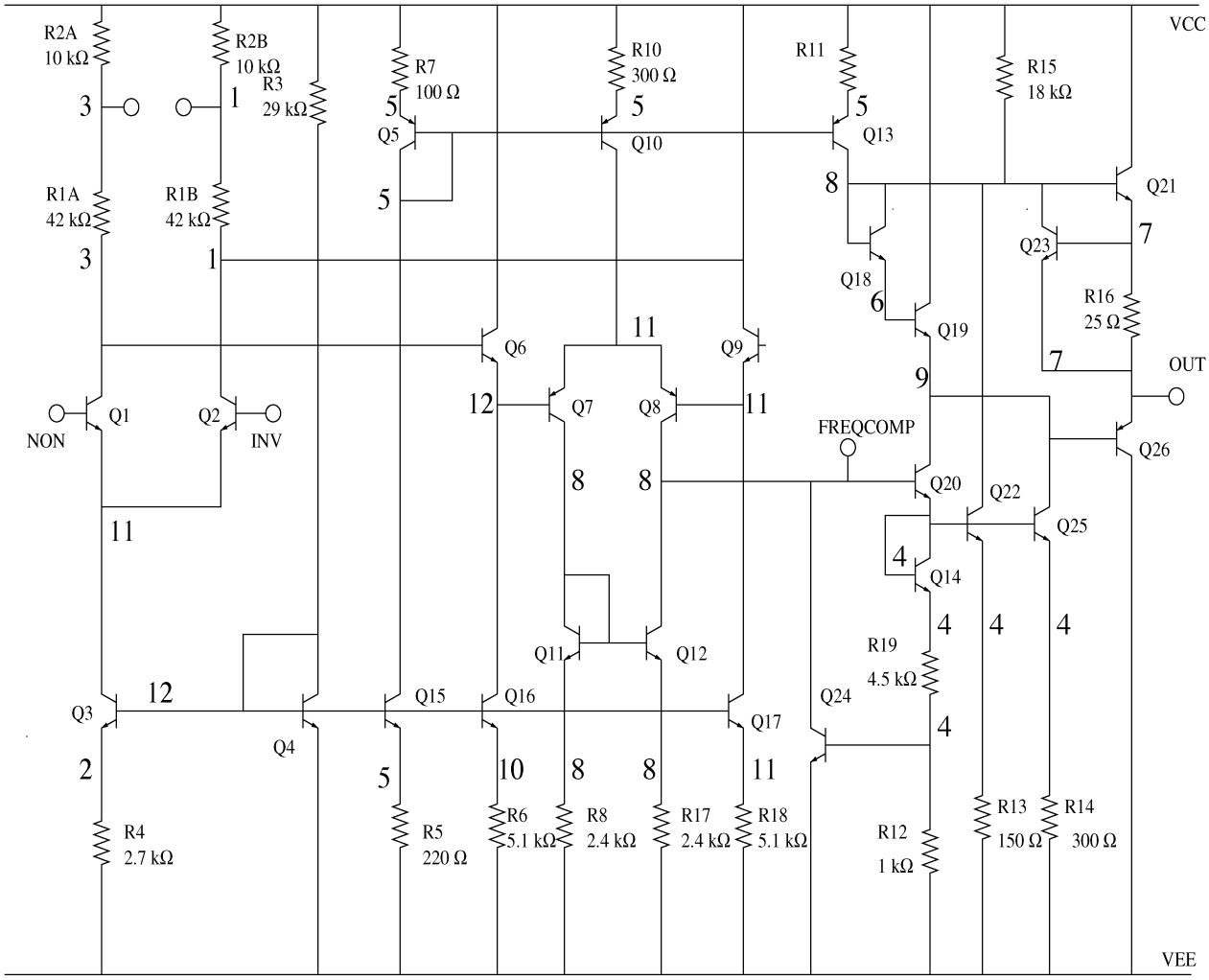

Fig. 6. $\mu A725$ OPAMP circuit with reduction order (the order indices are marked around each node).

One important issue for hierarchical DDD construction is how to build the circuit hierarchy to reduce the resulting DDD size and construction time. As studied in [12], ordering is critical for the DDD size. In hierarchical decomposition, ordering reflects which nodes are reduced first as subcircuits. The best ordering for Gaussian elimination in terms of minimum fill-ins is also the best ordering for DDD graphs in terms of DDD size. As a result, the authors apply the wildly used multiple minimum degree (MMD) algorithm [10] to find the best ordering. The MMD algorithm can find an ordering close to the real minimum degree ordering efficiently.

For the problem, once the authors have a node ordering, they group nodes into a number of subcircuits based on the constraints on the user-specified limits on internal nodes for each subcircuit. At the same time, the authors want to reduce the boundary nodes among different subcircuits for construction efficiency, so they use the connectivity-based clustering scheme where nodes are selected for clustering sequentially following the MMD ordering. Also for input and output nodes, which must be in the top level circuit, the authors explicitly avoid putting any adjacent devices of those nodes into any subcircuit to prevent them from crossing many circuit hierarchical levels.

VI. Experimental Results

The proposed algorithm has been implemented. A number of analog circuits ranging from large operational amplifier (OPAMP) circuits, active filters, to mesh-structured *RC* filters are analyzed symbolically. All results are collected on a Linux workstation with a dual 1.6-GHz AMD Athlon MP 2000+ CPUs and 2 GB of memory.

The authors first test the program on a number of small analog circuits for which the nonhierarchical method can still be used for construction. Both algorithms give DDDs with the same number of product terms, but with different DDD sizes. Table I summarizes the results. In this table, $|\mathrm{DDD}|$ is the number of DDD nodes for the denominator of a transfer function. If two numbers $n1/n2$ are given, $n1$ is the DDD size from the hierarchical method and $n2$ is the DDD size from the flat method. #path is the number of product terms in the denominator of the transfer function and #IntNode is the limit for the number of internal nodes for any subcircuit. This number may be absent when the circuits are predefined hierarchically. CPU time(hier) gives the CPU time using the hierarchical construction method and CPU time(flat) gives the CPU time using the flat Laplace DDD construction method.

In general, the authors find that the new construction method will lead to larger DDD sizes than the nonhierarchical method in general. This is a not surprise as ordering is done for each subcircuit separately. Although in the subcircuit level, it follows the minimum degree ordering.

Then, the authors test a number of circuits that cannot be analyzed by the nonhierarchical method. The first one is the $\mu A725$ OPAMP circuit, which has 32 nodes as shown in Fig. 4.

For completeness, the small signal model used for bipolar transistors is shown in Fig. 5.

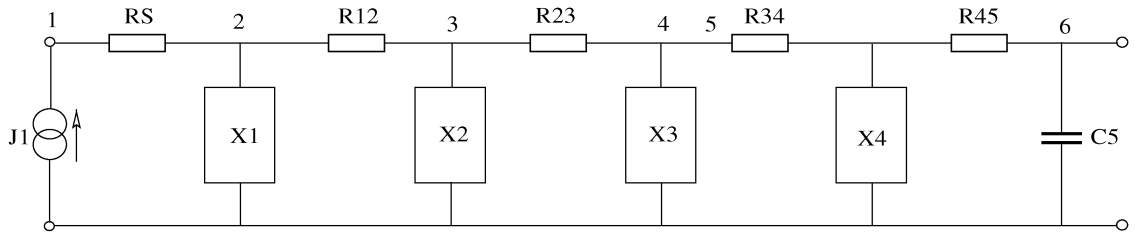

Fig. 7. Active low-pass filter.

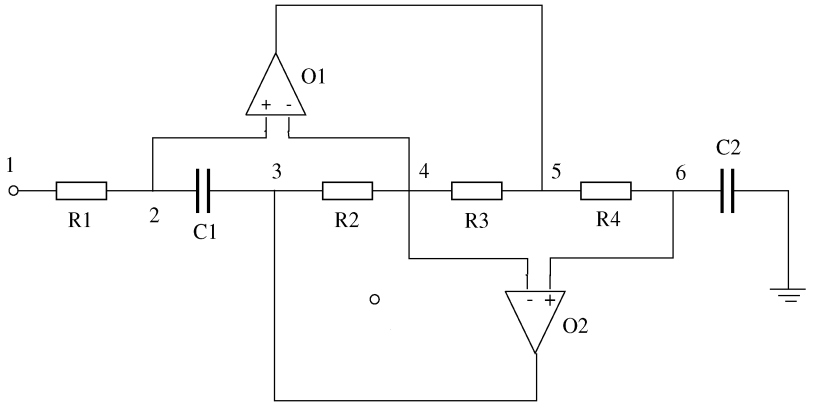

Fig. 8. Frequency dependent negative resistance (FDNR) subcircuit.

By using the new approach, it takes less than 5s to finish the construction. The number of internal nodes for each subcircuit is set to 5. This is the first time that analog circuits with this size can be exactly analyzed symbolically.

Fig. 6 shows the reduction order used for $\mu A725$. There are 11 subcircuits with internal node number limited to five. The numbers marked along the nodes are the subcircuit indices. A subcircuit with index 1 will be suppressed first and so on.

Then, the authors test the new method on a hierarchically defined active low pass filter circuit shown in Fig. 7. The filter circuit is a ninth-order Cauer-parameter low-pass filter [20], which has four identical subcircuits, named $X1$ to $X4$. Fig. 8 shows the detailed structure of a subcircuit. Each subcircuit contains two OPAMP subcircuits. The authors have tested the program on an implementation of OPAMP subcircuits: a linear model of 741 OPAMP circuit shown in Fig. 9. For this circuit, no clustering is used as the circuit has a predefined circuit hierarchy. The new algorithm takes 8.43 s to finish the construction.

The third example is a state variable filter circuit shown in Fig. 10. It consists of four identical OPAMPs, which in turn are implemented by a linear model of 741 OPAMP in Fig. 9. The authors use the existing hierarchical definition of the circuit for hierarchical analysis. To increase the circuit complexity, the authors cascade two- and three-state variable filter circuits into one circuit where each state variable filter circuit is treated as a subcircuit. The results are reported in Table I, where stat_var_filter(1) is the original circuit, stat_var_filter(2) is the circuit with two state variable subcircuits, and stat_var_filter(3) is the circuit with three-state variable subcircuits. As the circuit complexity increases, the construction time also increases significantly as shown for circuit stat_var_filter(3).

Finally, the authors perform the DDD construction on a number of mesh-structured *RC* circuits. Circuits p5x20x{2–4} have five rows and each row is an *RC* ladder circuit with 20 *RC* segments. The last digit is the number of vertical lines for connecting the five ladder circuits at evenly distributed points. In Table I, as the number of vertical lines increases, so do the number of product terms and construction time.

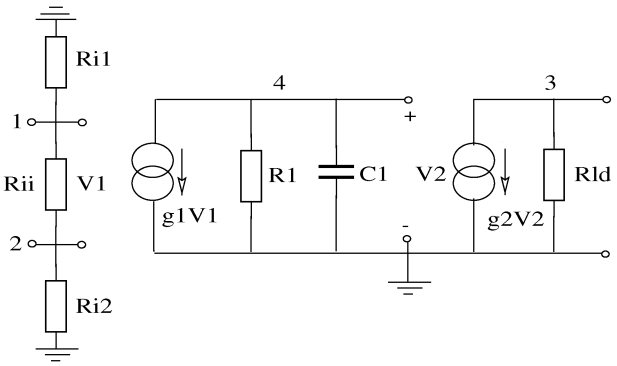

Fig. 9. Linear model of 741 OPAMP.

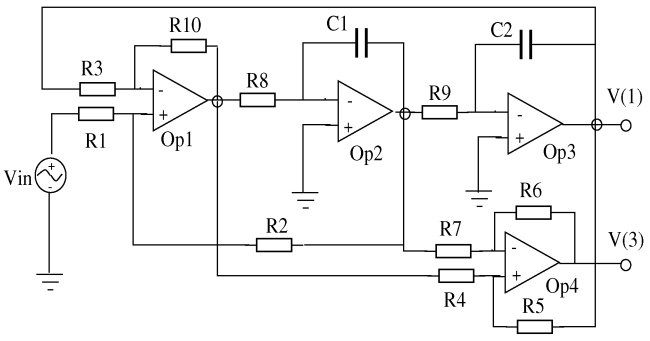

Fig. 10. State variable filter circuit.

For tree-like structured circuits p10x100 and p10x150, which have 10^3 nodes, the new algorithm constructs DDD graphs in a very short time. Actually, the authors can analyze almost arbitrarily large tree-structured circuits due to their localized nature given enough memory. N/A here means the number of paths is out of numerical range of floating point numbers in the computer.

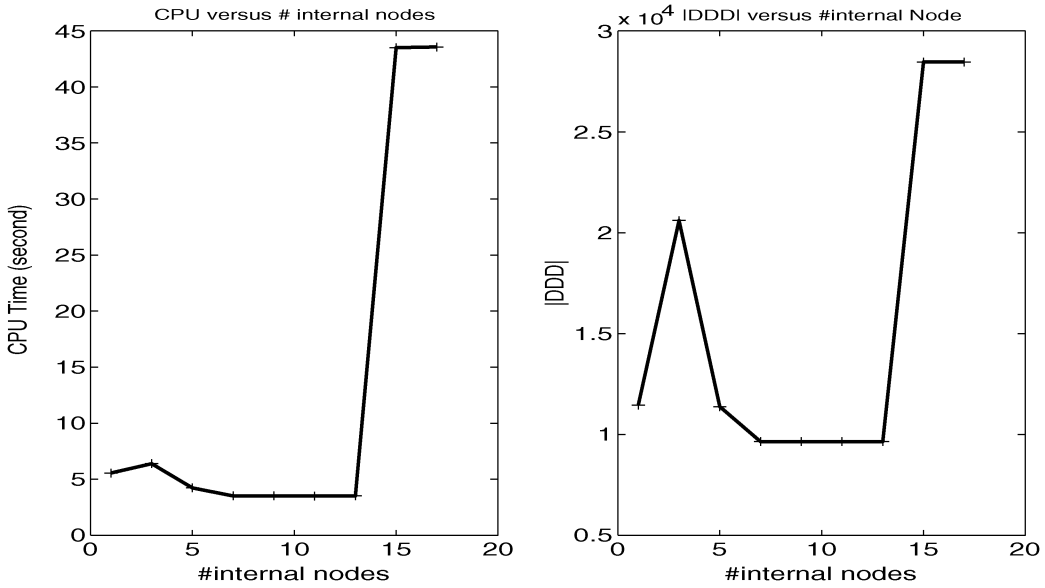

Fig. 11. CPU time and DDD sizes versus #internal nodes.

Finally, the authors study how clustering affects the construction time and final DDD sizes. Fig. 11 shows the CPU time and final DDD sizes for the denominator of a transfer function of $\mu A725$ circuit clustered under different numbers of internal nodes.

Typically, the authors find that a number of internal nodes between five and ten give the best results in terms of both construction time and DDD sizes. This is true for other circuits. The difference in construction time and final DDD sizes can be one or several orders of magnitude different between different numbers of internal node limit. The jump around 13 in Fig. 11 clearly shows the fact that the DDD construction time for the subcircuit may increase exponentially with the size of subcircuits by using the Laplace expansion method as mentioned before.

The authors also notice that as the construction process continues, the DDD sizes grow larger. This will slow down the construction process as in the worst case the process needs to visit every node once in existing DDD graphs to build a new one. So the time complexity of the construction process depends on the final DDD size and the hierarchical structure of the circuit. The actual sizes of circuits that can be analyzed are subject to the structure of the circuits, the computer memory, and the efficiency of DDD graph operations (with or without using garbage collection to improve memory efficiency).

VII. Conclusion and Future Work

In this paper, the authors have proposed a novel approach to the exact symbolic analysis of large analog circuits. The new method is based on the hierarchical circuit decomposition and determinant decision diagrams (DDDs) for representation of circuit matrix determinants. The authors have proposed a novel hierarchical DDD graph construction algorithm based on symbolic decancellation. The authors showed that exact symbolic expressions of a circuit are cancellation-free expressions in the hierarchical decomposition of the same circuit. The experimental results have shown that large analog circuits like $\mu A725$ and mesh-structured *RC* filter circuits up to 100 nodes and tree-like structured circuits up to 1000 nodes, which cannot be analyzed exactly before, can be analyzed by the new approach for the first time. Such symbolic simulation capacity can be used for the symbolic analysis and behavioral modeling of many practical analog blocks, which are typically less than 100 nodes. The new approach has substantially improved the capacity for exact symbolic simulation.

To further improve the full symbolical hierarchical analysis capacity, dynamic vertex ordering techniques [22] will be applied, which will further reduce the DDD size significantly during the hierarchical construction and increase the symbolic analysis capacity.

References

[1] S.-M. Chang, J.-F. MacKey, and G. M. Wierzba, "Matrix reduction and numerical approximation during computation techniques for symbolic analog circuit analysis," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, San Diego, CA, 1992, pp. 1153–1156.

[2] F. V. Fernandez, J. Martin, A. Rodriguez-Vazquez, and J. L. Huertas, "On simplification techniques for symbolic analysis of analog integrated circuits," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, San Diego, CA, 1992, pp. 1149–1152.

[3] F. V. Fernandez and A. Rodriguez-Vazquez, "Symbolic analysis tools-the state of the art," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Atlanta, GA, 1996, pp. 798–801.

[4] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits*. Norwell, MA: Kluwer, 1991.

[5] G. Gielen, P. Wambacq, and W. Sansen, "Symbolic analysis methods and applications for analog circuits: A tutorial overview," *Proc. IEEE*, vol. 82, no. 2, pp. 287–304, Feb. 1994.

[6] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1989.

[7] M. M. Hassoun and P. M. Lin, "A hierarchical network approach to symbolic analysis of large scale networks," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 42, no. 4, pp. 201–211, Apr. 1995.

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

[9] P. M. Lin, "Sensitivity analysis of large linear networks using symbolic program," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, San Diego, CA, 1992, pp. 1145–1148.

[10] J. W. Liu, "Modification of the minimum degree algorithm by multiple elimination," *ACM Trans. Math. Softw.*, vol. 11, no. 2, pp. 141–153, 1985.

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

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

[13] ——, "Compact representation and efficient generation of s-expanded symbolic network functions for computer-aided analog circuit design," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 20, no. 7, pp. 813–827, Apr. 2001.

[14] J. A. Starzky and A. Konczykowska, "Flowgraph analysis of large electronic networks," *IEEE Trans. Circuits Syst.*, vol. 33, no. 3, pp. 302–315, Mar. 1986.
[15] S. X.-D. Tan, "A general hierarchical circuit modeling and simulation algorithm," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, no. 3, pp. 418–434, Mar. 2005.
[16] ——, "A general s-domain hierarchical network reduction algorithm," in *Proc. Int. Conf. Computer Aided Design (ICCAD)*, San Jose, CA, 2003, pp. 650–657.
[17] S. X.-D. Tan, Z. Qi, and H. Li, "Hierarchical modeling and simulation of large analog circuits," in *Proc. Design, Automation and Test Europe Conf. (DATE)*, Paris, France, Feb. 2004, pp. 740–741.
[18] S. X.-D. Tan and C.-J. Shi, "Efficient approximation of symbolic expressions for analog behavioral modeling and analysis," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 23, no. 6, pp. 907–918, Jun. 2004.
[19] X.-D. Tan and C.-J. Shi, "Hierarchical symbolic analysis of large analog circuits via determinant decision diagrams," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 19, no. 4, pp. 401–412, Apr. 2000.
[20] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York: Van Nostrand, 1995.
[21] P. Wambacq, G. Gielen, and W. Sansen, "A new reliable approximation method for expanded symbolic network functions," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Atlanta, GA, 1996, pp. 584–587.
[22] J. Yang and S. X.-D. Tan, "An efficient algorithm for transient and distortion analysis of mildly nonlinear analog circuits," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Vancouver, BC, Canada, May 2004, pp. V129–V132.
[23] Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 43, no. 8, pp. 656–669, Aug. 1996.

A Study of a Hybrid Phase-Pole Macromodel for Transient Simulation of Complex Interconnects Structures

Bing Zhong, Tao Hu, Dawei Fu, Steven L. Dvorak, and John L. Prince

***Abstract*—An overview of standard macromodeling techniques (i.e., employ poles but no phase shifts) for transient simulation of high-speed interconnects is first presented. Then, the limitations of these standard macromodeling techniques (e.g., high model order and slow convergence) are discussed. In order to overcome these limitations, generalized method of characteristics (MoC) techniques include the physical phenomenology of phase shift (time delay) in addition to the system poles, thereby making it possible to model single and coupled transmission lines using far fewer terms than when standard macromodeling techniques are employed. Since MoC techniques incorporate the time delay into the model, causality is also guaranteed. In this paper, the MoC idea is extended by developing a hybrid phase-pole macromodel (HPPM) for the modeling of more complex interconnects with embedded discontinuities. Unlike other generalized MoC techniques that have only been applied to single and coupled transmission lines, the HPPM macromodel can be applied to larger portions of the system that contain multiple cascaded transmission lines and discontinuities. The HPPM parameters can be extracted from either measured or simulated transient data. Comparisons between a standard macromodel and the HPPM show that the HPPM has significant advantages in terms of reduced macromodel orders.**

***Index Terms*—Hybrid phase-pole macromodel, interconnect, macromodels, method of characteristics, transmission line.**

I. Standard Macromodeling Techniques

Twenty-first century package designs require wideband frequency simulations. In packaging design procedures, system performance under various input conditions needs to be evaluated, especially during the optimization cycle. Therefore, simulations involving complex interconnects (like long lossy frequency dependent lines) have to be performed during each optimization cycle. Macromodeling, or so-called model-based parameter estimation (MBPE) [1], reduces the model order by replacing the first-principle model or measurement data with a fitting model. Such models can be used in simulations for high-speed interconnect structures, provided they have sufficient accuracy. When employing macromodeling or model order reduction techniques, the response of a system can be expressed in the time domain and frequency domain as

$$f(t) = f_{\rm p}(t) + f_{\rm np}(t) = \sum_{\alpha=1}^{M} R_\alpha \exp(s_\alpha t) u(t) + f_{\rm np}(t) \quad (1)$$

$$F(s) = F_{\rm p}(s) + F_{\rm np}(s) = \sum_{\alpha=1}^{M} \frac{R_\alpha}{s - s_\alpha} + F_{\rm np}(s) \quad (2)$$

where $f_{\rm np}(t)$ and $F_{\rm np}(s)$ are the nonpole components in the time and frequency domains, respectively, and $u(t)$ denotes a unit step function. The time-domain fitting model involves a superposition of a series of decaying exponential waves that all turn on at zero time. Since every wave propagates at a finite velocity, the terms in the series must sum

Manuscript received August 22, 2002; revised November 26, 2002, August 2, 2003, and May 27, 2004. This paper was supported by the Semiconductor Research Corporation under Task ID 956.001. This paper was recommended by Associate Editor C.-J. R. Shi.

B. Zhong, D. Fu, S. L. Dvorak, and J. L. Prince are with the Electronics and Communications Engineering (ECE) Department, University of Arizona, Tucson, AZ 85721 USA (e-mail: bingzh@ece.arizona.edu).
T. Hu is with Sigrity, Shanghai, China.

Digital Object Identifier 10.1109/TCAD.2005.850817